INSTITUTO
UNIVERSITÁRIO
DE LISBOA

# Investigating the Attributes of Software Testers to Avoid Human Bias in Software Testing

Tran Nguyen Nhu Ngoc

Master's in Management of Services and Technology

Supervisor:

João Carlos Rosmaninho de Menezes, Associate Professor, Department of Marketing, Operations and General Management (IBS)

May, 2023

BUSINESS
SCHOOL

Department of Marketing, Strategy and Operations

# Investigating the Attributes of Software Testers to Avoid Human Bias in Software Testing

Tran Nguyen Nhu Ngoc

Master's in Management of Services and Technology

Supervisor:

João Carlos Rosmaninho de Menezes, Associate Professor

May, 2023

**Acknowledgement**

The completion of this Master's thesis marks a significant milestone in my academic journey, one that would not have been possible without the support and guidance of many.

I wish to first express my profound gratitude to Professor João Carlos Menezes, my thesis supervisor. His patience, wisdom, and mentorship have been invaluable over the past two years. He has shaped not only my thinking in this work but also my development as a scholar. For that, I am deeply indebted.

I also wish to thank the many experts who generously contributed their time and insights through interviews. Access to their hard-won knowledge and experience was crucial in informing my research questions and analysis. Their willingness to share reflects a spirit of intellectual generosity that I hope to emulate.

This thesis also would not have been completed without the support of treasured friends, especially Kim, Maria, Renata and Esmeralda. They have sustained me through challenges and self-doubts, providing empathy, care, and humor that lifted my spirits when most needed. True friendship is a gift, and I am grateful for theirs.

My colleagues in the SC team have also inspired and motivated me in this work. Interacting with their passionate minds has fueled my own curiosity and excitement for this research. I am proud to work among them.

Most of all, I wish to thank my family for their unceasing love and sacrifices. My partner, Robin, especially, has endured and supported me steadfastly through long hours and preoccupation. His patience and care humble me. My family's faith in me keeps me determined to push forward.

Finally, I owe thanks to myself for persevering in the pursuit of this goal despite any obstacles or setbacks. Completing this thesis is evidence that perseverance pays off and a reminder that self-belief is key to overcoming self-doubt.

My deepest gratitude to you all. This thesis is a testament to what we can achieve together.

[This page was intentionally left blank]

**Resumo**

Esta tese investiga os atributos necessários aos testadores de software para mitigar vieses humanos na prática de testes. A pesquisa concentra-se na identificação de atributos essenciais para testadores de software e propõe um framework para apoiá-los na eliminação de vieses. A metodologia empírica utilizou abordagens qualitativas, incluindo entrevistas individuais com nove especialistas internacionais. Os principais resultados revelam atributos cruciais para mitigar vieses humanos, como proficiência técnica, habilidades de comunicação e colaboração, adaptabilidade, atenção aos detalhes, gestão do tempo, neutralidade e habilidade na elaboração de casos de teste. Contribuições adicionais da revisão da literatura incluem autonomia, flexibilidade, curiosidade, capacidade de tomada de decisão e disposição para aprender.

Além disso, a tese propõe um framework abrangente que inclui: conscientização e educação, planejamento, estratégia, elaboração e execução de testes; coleta e análise de dados; revisão por pares e colaboração; verificação e validação independentes; aprendizagem e melhoria contínua; considerações éticas. A adoção deste framework, juntamente com frameworks de teste comuns, possibilita que testadores de software melhorem consideravelmente a qualidade e confiabilidade dos produtos de software, minimizando vieses humanos. A implementação dos resultados pode aprimorar significativamente a qualidade dos testes, reduzir erros humanos, diminuir defeitos do software e aumentar a confiança das partes interessadas. Em suma, esta tese contribui teórica e praticamente, identificando atributos essenciais para mitigar vieses em testes de software e propondo um framework abrangente para que testadores de software abordem esta questão. Os resultados podem beneficiar pesquisadores, profissionais e organizações envolvidas em testes de software.

**Palavras chave:** Testes de Software, Fatores humanos, Testador de Software, Gestão de Operações

**Códigos de Classificação JEL:** Programas de Computador (L86), Gestão da Tecnologia da Informação (M15)

[This page was intentionally left blank]

**Abstract**

This thesis examines the attributes of software testers to mitigate human bias in software testing. The research aims to identify the essential attributes required for software testers to effectively address bias and proposes a framework to support them in eliminating bias during testing. The study employs qualitative research methods, including individual interviews with nine experts from diverse countries. The findings highlight crucial attributes for mitigating human bias, including technical proficiency, communication and collaboration skills, adaptability, attention to detail, time management, neutrality, and test case design skills. The literature review contributes additional insights such as autonomy, flexibility, curiosity, decision-making abilities, and eagerness to learn. Moreover, the thesis presents a comprehensive framework consisting of awareness and education, test planning and strategy, test design, test execution, data collection and analysis, peer review and collaboration, independent verification and validation, continuous learning and improvement, and ethical considerations. By adopting this framework and integrating some common testing frameworks, software testers can enhance the quality and reliability of software products while minimizing human bias. The proposed framework provides a holistic approach to support software testers in addressing bias and promoting unbiased software testing practices. Implementing the findings from this research can significantly improve testing quality, reduce human error, decrease software defects, and enhance stakeholder confidence in software products.

**Keywords:** Software Testing, Human factors, Software Tester, Operations Management

**JEL Classification System:** Computer Software (L86), IT Management (M15)

[This page was intentionally left blank]

**Table of Contents**

1. Introduction ..... 1
1.1. Motivation and Problems statement ..... 1
1.2. Research questions ..... 2
1.3. Thesis outline ..... 3
2. Literature review ..... 5
2.1. Overview of Software Testing and Various Testing Approaches ..... 6
2.2. Software Tester Background and Essential Attributes ..... 8
2.3. Negative Factors Impacting Testing Quality and Human Bias ..... 11
2.4. Framework For Software Testers ..... 13
2.5. Conceptual Framework: Testing and Bias ..... 14
2.6. Summary ..... 15
3. Methodology ..... 17
3.1 Empirical Research Methodology ..... 17
3.1.1. Selecting a Methodology for Research ..... 17
3.1.2. Research Method description ..... 18
3.2. Interview design ..... 19
3.2.1. Analysis unit ..... 19
3.2.2. Data collection ..... 19
3.2.3. Data analysis ..... 20
4. Result ..... 23
4.1. Interview Respondents ..... 23
4.2. Results Summary ..... 27
5. Discussion ..... 43
6. Conclusion ..... 51
6.1. Implications ..... 51
6.2. Limitations ..... 53
6.3. Future work ..... 54
7. Bibliography ..... 57
8. Appendices ..... 63
Appendix A – Interview questions and guidelines ..... 63
Appendix B – Interview transcript ..... 64

[This page was intentionally left blank]

**List of Tables**

Table 2.1. Characteristics of Software Testers from Scientific Data Summary ..... 11
Table 4.1. Summary of respondents ..... 24
Table 4.2. Key Aspects of the Impact of Bias ..... 41

[This page was intentionally left blank]

**List of Figures**

Figure 2.1. Summary of types of human bias ..... 13
Figure 2.2. Conceptual Framework of Testing and Bias..... 15

[This page was intentionally left blank]

**Abbreviations**

| | |
|---|---|
| BA | Business Analyst |
| BDD | Behavior-Driven Development |
| Dev | Developer |
| FMCG | Fast-Moving Consumer Goods |
| ISTQB | The International Software Testing Qualifications Board |
| IT | Information Technology |
| PM | Project Manager |
| PO | Product Owner |
| QA | Quality Assurance |
| TaaS | Testing as a Service |
| UA | User Acceptance |
| UI | User Interface |
| UX | User Experience |
| WWII | World War II |

# 1. Introduction

## 1.1. Motivation and Problem Statement

In the present fourth industrial revolution, software testing and development are important components that play a significant role in ensuring software quality. As digitalization continues to boost human productivity, advances in artificial intelligence have changed how software is assessed. However, the capabilities of AI are still restricted to the algorithms or programming included in its internal circuit, which is why humans are still recognized as a reliable source for software testing.

Software testing is recognized as a labor-intensive activity, a socio-technical activity rather than a simple technological one. As a result, the quality of a product is strongly dependent on the knowledge of the specialists involved in the process (Mäntylä et al., 2012). Software testing requires more than just understanding user requirements and conveying them to developers. As a result, software testers must understand the customer's business requirements in order to ensure that the program meets the user's expectations.

In areas such as healthcare and banking, the software is vital for storing patient data and performing critical activities such as transaction management. Any flaws or inconsistencies in the program might have serious effects, making the function of a human tester even more vital. Because organizations prioritize validation from the end user's perspective, the role of a human tester becomes increasingly crucial when goods have sophisticated user interfaces for end users (Mäntylä et al., 2012).

The testing life cycle is divided into numerous stages, such as test planning, design, and analysis. Most studies, however, concentrate on the overall picture of a tester's character rather than the precise qualities required for each step. Even though software testing, particularly human testing, is a well-researched issue with scientific publications and conferences, there are numerous substantial gaps between academic research and the practical challenges that individuals working in this sector encounter (Glass et al., 2006). A skilled software tester is inquisitive, capable of seeing the bigger picture, and thinks innovatively. (Stray et al., 2021). While human testing helps find and fix undiscovered defects, bug-free software is not necessarily flawless.

In addition, testing as a service (TaaS) has emerged as an important operations and quality practice for companies. TaaS involves "the outsourcing of software testing activities to a third-party service provider" (Software Testing Outsourcing- QA Outsource, n.d.). This allows companies to leverage specialized testing expertise and resources on demand and at a lower cost. Implementing TaaS as part of wider operations and total quality management practices

has been shown to improve software quality, reduce costs, and increase customer satisfaction (Ogbari & Borishade, 2015).

Numerous studies have been conducted to investigate how team members' talents impact software quality, teamwork performance, and the competitive market benefits such skills provide (Ahmed et al., 2012), as well as the importance of general human factors in the software tester's function (Deak et al., 2016; Kanij et al., 2015). However, little study has been conducted on the human bias of software testers. Many of the studies reviewed for this study often concern the performance of some testing methodologies, the benefits of automation, or variances in procedures employed in software factories that utilize traditional or agile development methods. In addition, there are no clear guidelines for this activity, and there are no defined character qualities that testers must possess, which leads to human bias and prejudice in the testing process. As a result, a framework that supports testers in overcoming human bias and prejudice throughout the testing process is required.

In summary, the primary aim of this study is to contribute actionable insights by investigating the vital attributes of software testers that effectively mitigate human bias during the software testing process. The motivation for this research, which emerged from the article by Calikli and Bener (2015) emphasizing the need to explore how testers can acquire desired characteristics and utilize them efficiently, aligns with the imperative of identifying factors that have a significant impact on bias, including confirmation bias. By gaining insight into these impactful elements, companies can reduce the adverse consequences of confirmation bias and also provide direction in hiring new software engineers and strategically assigning current ones to roles where they can operate most productively.

### 1.2. Research Questions

The goal of this study is to look at the necessary attributes that software testers in the software quality assurance industry should have to reduce human bias in the testing process. The study will concentrate on software testers, who are in charge of assessing the functionality and performance of software programs. It will investigate how characteristics like critical thinking, attention to detail, and open-mindedness might aid unbiased decision-making during testing.

An important outcome of this research will be the creation of a comprehensive framework that provides guidance for software testers on making fair and unbiased decisions. However, the development of this framework will occur towards the end of the study, building upon the insights and findings uncovered during the research process. The research will be conducted

through open-ended interviews with subject-matter experts from various industries. To help address the overall research problem, the following two questions will steer the investigation:

*RQ1: What are the essential attributes required for a software tester to mitigate human bias during software testing?*

*RQ2: What framework can be deployed to support software testers in eliminating human bias during software testing?*

To establish a robust theoretical basis for the research inquiries, an exhaustive analysis of relevant prior work will be performed. The goal of reviewing the literature is to scrutinize current understanding and pertinent scientific advances in the domain of software testing. It will help situate this study within a broader context and highlight the implications of prior research on the current work. The literature review will support and inform the overall research design and analysis.

### 1.3. Thesis outline

This part describes the overall organization of the research report as well as a quick overview of each chapter.

*Chapter 1* serves as an introduction to the research study, presenting the author's motivation and objectives. It also provides a brief outline of the research approach, which includes two research questions derived from the problem statement and the structure of the thesis to address the problem.

*Chapter 2* of this thesis presents a comprehensive literature review that covers three main study areas: the general attributes of the software tester and types of bias in software testing. The chapter provides an in-depth analysis of research papers and other relevant literature in each of these areas to provide a conceptual framework.

*Chapter 3* employs a qualitative research strategy based on semi-structured interviews to elicit significant insights from experts on many areas of software testing, such as techniques, procedures, strategies, and current difficulties. This chapter will explain the study methodology and its unique technique, as well as interview instructions, data-gathering processes, and analytic procedures. The research findings will give a more in-depth understanding of the key characteristics required by software testers to eliminate human bias and identify aspects required for establishing a complete framework to assist them in making unbiased conclusions.

*Chapter 4* presents the findings obtained from the qualitative research conducted in *Chapter 3* by analyzing the data collected from the interviews and identifying key themes and patterns.

*Chapter 5* covers the paper's contribution to current research and discusses any constraints that were discovered throughout the study process. It offers management advice to software testers by answering the research questions defined at the beginning.

*Chapter 6* summarizes the results and provides managerial recommendations for software testers. It also discusses the paper's contribution to existing research, limitations, and suggestions for future work.

## 2. Literature Review

In this study, a structured review technique was employed to perform a methodical evaluation of the current literature on the attributes and human bias of software testers. The review process involved several phases, including issue conceptualization, literature search, data assessment, data analysis and synthesis, interpretation, presentation, and reflection.

An extensive review of the literature was conducted to explore English and German academic sources on the topic comprehensively. A broad range of scholarly databases were consulted, such as Google Scholar, ResearchGate, ScienceDirect, and Enterprise Science. The search encompassed a variety of formats, including peer-reviewed journals, academic books, papers from professional conferences, doctoral dissertations, and subject-specific websites. The aim was to gain deep insight into the current state of research on the subject and identify gaps left unaddressed in prior work. By casting a wide net across languages and types of publications, this literature review sought to build a foundational understanding of the topic to inform further study.

To identify relevant secondary sources, specific keywords related to software testers and testing were employed, such as “software tester characters”, “human factor of a software tester”, “challenges in software testing”, and “types of bias in software testing”. These keywords were carefully selected to capture different aspects of the topic and retrieve relevant sources for the research paper.

The identified sources were documented in an organized manner, including information such as themes, authors, years of publication, abstracts, and notable discoveries. Thorough reviews were conducted to assess the relevance of the sources to the research topics, excluding those that did not significantly contribute to the study's work.

To ensure the validity and currency of the literature, backward and forward searches were performed. Backward reference searching involved examining the references cited in the publications, while forward searching included reviewing publications that cited the current references to identify any developments on the topic. These searches enabled the identification of additional sources and ensured the research paper was based on up-to-date literature.

During the interpretation phase, the acquired literature was synthesized by categorizing the material into appropriate categories and sorting and synthesizing it according to the study's established structure. This phase considered both supportive and conflicting remarks on the issue, with a specific focus on software testers and their distinguishing characteristics.

Finally, the findings were reported in a clear and systematic structure within the research paper, highlighting the most important articles related to the research issues, including the

history of software testing, the profile of a tester, and the negative repercussions on testing quality.

By following these rigorous processes, the literature review conducted in this study aimed to provide a comprehensive understanding of the attributes and human bias of software testers, ensuring the inclusion of relevant and up-to-date information in the research paper.

### 2.1. Overview of Software Testing and Various Testing Approaches

In order to raise the quality of software products and improve user experiences, software testing is an essential part of software development. The process includes several tasks, such as test planning, analyzing, generating, and implementing tests, reporting test progress and results, and assessing the quality of a test object (Dosaj, n.d.). Test execution encompasses conducting the tests themselves as well as verifying the results. For maximum benefit, quality assurance like testing should commence early on in software development before avoidable costs and delays build up (Dosaj, n.d.). However, research on software testing lags behind research on other professions within software development teams, and the occupations of tester and maintainer in software development are the least popular, which might be a disadvantage for collaborative software development (Capretz et al., 2015; Deak et al., 2013; Florea & Stray, 2019; Shah & Harrold, 2010).

From a socio-technical standpoint, organizational realities and constraints have a substantial effect on software testing (Martin et al., 2007). Testing has been studied in the real world as a social and organizational process (Rooksby et al., 2009). Andersson and Runeson (2002) found that priorities around verification and validation processes—closely tied to software testing—are unclear, as teams frequently have to choose between meeting deadlines or upholding quality standards. They also noted that organizations tend not to consider inexperienced employees able to handle verification and validation tasks adequately. This indicates that software testing is often not considered a top priority in the software development process, and the testing team may not receive the same level of recognition as the development team.

Experts point out that software testing roles offer substantial opportunities for learning and development (Capretz et al., 2019). Capretz et al.'s (2019) research found that most Indian testing professionals valued the cognitive challenges of testing and saw it as well-suited for "newcomers" to the field. However, others viewed the repetitiveness and limited career progression in testing jobs as a downside. These varied perspectives suggest that impressions

of software testing as a career path differ, requiring organizations to work to attract and keep top testing talent.

Program testing is likely critical because it ensures that the program satisfies the necessary quality requirements and performs as intended. If software programs are launched without being tested, the repercussions can be devastating, including financial loss, harm to a company's reputation, and legal liability (Beizer, 1990). The testing process aids in the detection of faults and mistakes in software, which might result in crashes, security breaches, or other types of software failures.

To make sure the program works as intended, functional testing is done. It is the most popular kind of software testing, and it involves assessing each software program's function against specified standards to make sure it performs as intended. Myers et al. (2012) note that functional testing may be performed either manually or through automation.

Performance testing assesses the scalability and speed of a software program across different settings. The goal is to determine how well the software functions under different volumes of work, network states, and other situations. This type of testing can identify potential obstacles that hinder performance as well as ways to improve. To ensure software applications satisfy expected performance standards, Myers et al. (2012) emphasize the importance of performance testing.

Software that has been subjected to security testing is examined for vulnerabilities that might be used by attackers. It consists of evaluating the software program's security measures to make sure that they comply with security requirements and safeguard user data. Security testing is essential for identifying security problems and guaranteeing software security, according to Felderer et al. (2016).

Usability testing assesses how easy it is to use a software program for the intended users. The goal is to ensure the application is straightforward, intuitive, and user-friendly. This type of testing can uncover issues with the user interface design, accessibility, and other usability problems. According to J. A. Whittaker and Thompson (2004), usability testing is vital for guaranteeing that software applications are easy to use and satisfy the needs of the target users.

Several widely used techniques for software testing have been developed to improve their effectiveness and efficiency.

Agile testing emphasizes collaboration and adaptability throughout the testing process, allowing for iteration and continuous input. It is an incremental, iterative process for creating software that includes regular testing and feedback to ensure the final result meets the required quality standards. As Crispin et al. (2008) point out, agile testing plays a key role in enabling

software to be developed and assessed in an iterative, adaptable process that yields results efficiently.

In order to find defects early and frequently, continuous testing involves testing across the whole software development lifecycle, from development to deployment. To ensure that the program is thoroughly tested and verified throughout the development process, it also uses continuous integration. The importance of continuous testing in identifying mistakes early and ensuring thorough testing of software throughout the development lifecycle is highlighted by Haertl et al. (2016).

Exploratory testing is a testing strategy that involves doing tests without following a predetermined test plan. This method enables testers to find new flaws and investigate the capabilities of the product. It requires a more innovative and impromptu method of testing software, which might help find flaws that might otherwise go undiscovered. The importance of exploratory testing in identifying issues that conventional testing methods miss is highlighted by Whittaker (2009).

Risk-based testing, model-based testing, and acceptability testing are a few other methods for testing software. The method selected depends on the particular specifications and characteristics of the software program because each methodology has advantages and disadvantages of its own.

In conclusion, software testing is an essential final step in software development. Through rigorous testing, developers can verify that the software achieves the right level of quality, includes the necessary features, and operates as intended. The careful application of different testing types, like functional, performance, security, and usability testing, evaluates the software from multiple angles. Combined with approaches optimized for efficiency and adaptability, such as agile, continuous, and exploratory testing methodologies, comprehensive testing gives software teams the insights needed to deploy solutions that fulfill users' requirements. By dedicating attention to testing, developers greatly increase the chances of building software upon which end users can rely. Overall, testing is pivotal in crafting software that is dependable, robust, and fit for purpose.

### 2.2. Software Tester Background and Essential Attributes

A tester is defined in a document as a qualified expert who does component or system testing (ISTQB Glossary, n.d.). Software testers play an integral role in software development by designing and executing test cases based on requirements (Kaner et al., 1999). They meticulously document and report any issues found, working with teams to resolve them and

improve quality (Kaner et al., 1999). Effective communication is also key, as testers provide updates on testing progress and results (Kaner et al., 1999). In addition to finding issues, testers have diverse responsibilities beyond basic checking. These include evaluating compliance with standards, handling high load conditions, detecting unauthorized access, and evaluating usability (Bertolino, 2007). By performing these multifaceted tasks, testers contribute to the overall success and reliability of software development (Bertolino, 2007). The presence of individuals with diverse educational backgrounds in the field of software testing highlights the importance of diversity in the software testing industry, which is critical to ensuring high-quality software development (Kaner et al., 1999).

Recent studies show that software testing jobs require workers with a wide range of advanced skills. According to a survey by Florea and Stray (2019), testers require an average of ten separate testing skills, five programming skills, and two soft skills to be successful in the software testing field. These include skills such as test planning, quickly learning new things, knowing different programming languages, and some other abilities. The increasingly complex technological solutions and systems that testers evaluate call for multifaceted abilities and knowledge. Testers must not only have deep expertise in quality assurance testing methods and tools but also possess complementary skills in areas like software engineering, user experience, and communication. The demanding and interdisciplinary nature of software testing points to the need for "highly qualified employees" with a versatile blend of technical, practical, and soft skills (Florea & Stray, 2019). Many academics also noted that software testers work on a wide range of issues since their employment necessitates a comprehensive understanding of the product (Hernandez & Marsden, 2014; Stray et al., 2021). This finding pointed out that the tester's capability to view the entire picture may bridge the gap between the domain and the technical aspects of the product while also advising on what to keep in mind for the other components during execution. This has the potential to improve team transparency, resulting in more successful cooperation while reducing misunderstandings.

Communication skills are extremely crucial for software testers (Florea & Stray, 2018; Matturro, 2013), and disclosing flaws constructively is recommended to build a collaborative work atmosphere. Software testers have been described as the software development team's biggest adversary; hence, they must have strong interpersonal skills, which were highlighted in earlier studies (Ahmed et al., 2012; Deak, 2014). According to a recent study, when a disagreement is anticipated, individuals are more cautious in their communication (Gren, 2017). Software testers must communicate in a way that avoids causing disagreement within the team. They claimed that it became easier to obtain assistance and comments from other testers.

According to the latest study, one of the primary reasons why software testers do not want to pursue a testing profession is the fear that other project members will be offended when the testers disclose the findings of grading their work (Lizama et al., 2020). In addition, testers are increasingly collaborating with developers, operations engineers, and even customer representatives much more intimately (Cunningham et al., 2019).

Although creativity is typically important in software development, its relevance varies substantially among positions (Li et al., 2020). According to studies by Stray et al. (2021) and Paruch et al. (2020), creativity allows software testers to identify uncommon bugs and resolve issues by generating various alternative solutions. Old research also found that testers need to be creative (Burnstein, 2006) because they must be able to picture the many ways a software item should be tested and form hypotheses about the various sorts of flaws that may exist and the various ways the program could fail.

Others, however, discovered that software testers were more organized, disciplined, and hardworking (Kanij et al., 2015; Paruch et al., 2020). The software testers underlined that they were organized and structured and that they enjoyed making and utilizing checklists; they stated that they generally recorded everything in lists, notes, or calendars (Stray et al., 2021). Organizational ability was also defined as the capacity to manage several activities efficiently and on time without wasting resources (Florea & Stray, 2018).

Studies have found that strong attention to detail helps software testers in key areas like evaluating user experience (Stray et al., 2021), gaining technical expertise (Kanij et al., 2014), and building domain knowledge (Stray et al., 2021). There were a limited number of documents accessible for the persons conducting the testing, or the information that was provided could not always be believed to be correct. As a result, domain expertise was deemed critical for understanding requirements and acting as a test. Knowledge of the product's implementation was deemed valuable during testing to test boundary values and improper input data types. High-performing testers, in particular, should be well-versed in the product under test, not just in how it is intended to be used but also in how it is executed (Iivonen et al., 2010).

Some of the characteristics determined to be significant for software testers are autonomy, flexibility, curiosity, decision-making abilities, critical thinking, and eagerness to learn (Florea & Stray, 2018; Iivonen et al., 2010; Li et al., 2020; Paruch et al., 2020; Stray et al., 2021).

Table 2.1. Characteristics of Software Testers from Scientific Data Summary

| Attribute | Citation |
|---|---|
| Attention to detail | Stray et al. (2021); Kanij et al. (2014) |
| Autonomy | Stray et al. (2021); Paruch et al. (2020) |
| Communication skills | Florea and Stray (2018); Matturro (2013); Deak (2014); Ahmed et al. (2012); Gren (2017); Lizama et al. (2020); Cunningham et al. (2019) |
| Creativity | Stray et al. (2021); Paruch et al. (2020); Burnstein (2006); Li et al. (2020) |
| Critical thinking | Stray et al. (2021); Paruch et al. (2020); Li et al. (2020) |
| Curiosity | Crispin et al. (2008); Paruch et al. (2020); Li et al. (2020) |
| Decision-making abilities | Stray et al. (2021); Paruch et al. (2020) |
| Diverse activities | Bertolino (2007) |
| Domain expertise | Iivonen et al. (2010) |
| Eagerness-to-learn | Stray et al. (2021); Paruch et al. (2020); Li et al. (2020); Florea and Stray (2018); Iivonen et al. (2010) |
| Flexibility | Stray et al. (2021); Paruch et al. (2020) |
| Organizational ability | Kanij et al. (2015); ); Florea and Stray (2018); Paruch et al. (2020); Stray et al. (2021) |
| Qualified expert | ISTQB Glossary. (n.d.) |
| Required skills | Florea and Stray (2019) |
| Time management | Kaner et al. (1999) |
| Working on wide range of issues | Stray et al. (2021); Hernandez and Marsden (2014) |

### 2.3. Negative Factors Impacting Testing Quality and Human Bias

The quality of the final software product is heavily influenced by various factors that can have a negative impact on software testing (Ahmed et al., 2012; Cruz et al., 2015). These factors include the cognitive, organizational, and operational aspects that affect the work of software testers as well as their working environment (Sánchez-Gordón et al., 2020). Testers' cognitive abilities can be affected by regular exposure to variables like heavy mental workload, stress, and psychological strain, which can impair their perception, processing, and recall of information (Beecham et al., 2008; Fernandez-Sanz et al., 2009; Gonçalves et al., 2017; Shah et al., 2014). Operational issues, such as task scheduling, team interaction, and employee

autonomy, can also influence the quality of testing (Gonçalves et al., 2017). Furthermore, organizational concerns like a lack of proper activity division, management, and personnel selection and training can contribute to the negative impact on software testing quality (Correia & Silveira, 2009).

In addition to these negative factors, human bias can significantly affect software testing. Confirmation bias, which is the tendency to favor information that confirms existing beliefs, is particularly relevant in the context of software (Mohanani et al., 2020; Salman et al., 2019). Testers may choose test cases that align with their expectations, neglecting alternative scenarios and contradicting evidence (Graham & Fewster, 2012). This bias can also influence test suite design and result interpretation (Graham & Fewster, 2012). Another form of bias is congruence bias, where testers rely primarily on documented expected behaviors and overlook unanticipated scenarios, potentially leading to issues after major releases or changes (Shenk & Shenk, 2022).

Negativity bias, the tendency to give more weight to negative information than positive information, can also affect software testing (Nikolopoulou, 2023). Testers may focus more on finding and reporting problems than recognizing product strengths and benefits (Shenk & Shenk, 2022). Survivorship bias, which occurs when only successful outcomes are analyzed, can create an incomplete view of testing effectiveness (Frost, 2022). Examining only successful tests or desired results can skew perspectives on testing quality and coverage.

Irving Janis, a social psychologist, popularized the term groupthink in the 1970s. Groupthink refers to the tendency of groups to conform to a dominant viewpoint in decision-making scenarios, leading to the suppression of individual critical thinking and a disregard for new information. This phenomenon often goes unnoticed as group members prioritize maintaining group cohesion and minimizing conflict (Janis, 1972). Despite the potential presence of dissenting opinions, individuals may still conform to the group opinion, possibly due to the influence of a strong authority figure or an initial consensus that is formed with limited information. Even as additional information is introduced and opinions evolve, individuals may remain hesitant to contradict the group's opinion. Groupthink may happen in essential and insignificant decision-making situations, it's crucial to remember that. Groupthink may have detrimental effects, including poor decision-making, a lack of inventiveness, and the failure to take into account opposing viewpoints. In view of this, it's essential to spot the symptoms of groupthink and take preventative measures to avoid it while making judgments in groups.

In-group prejudice, the tendency to show preference for one's own group, can also impact perspectives or decision-making in testing teams (Tajfel & Turner, 2004). This can foster dismissive or discriminatory treatment of perceived outsiders (Mullen et al., 1992). Within testing teams, in-group prejudice may negatively impact perspectives or decision making, though impacts can be subtle.

To ensure the quality of the final software product, it is important for software development teams to be aware of and address these negative factors and human biases in software testing. By understanding and mitigating these influences, teams can improve the effectiveness and accuracy of their testing processes.

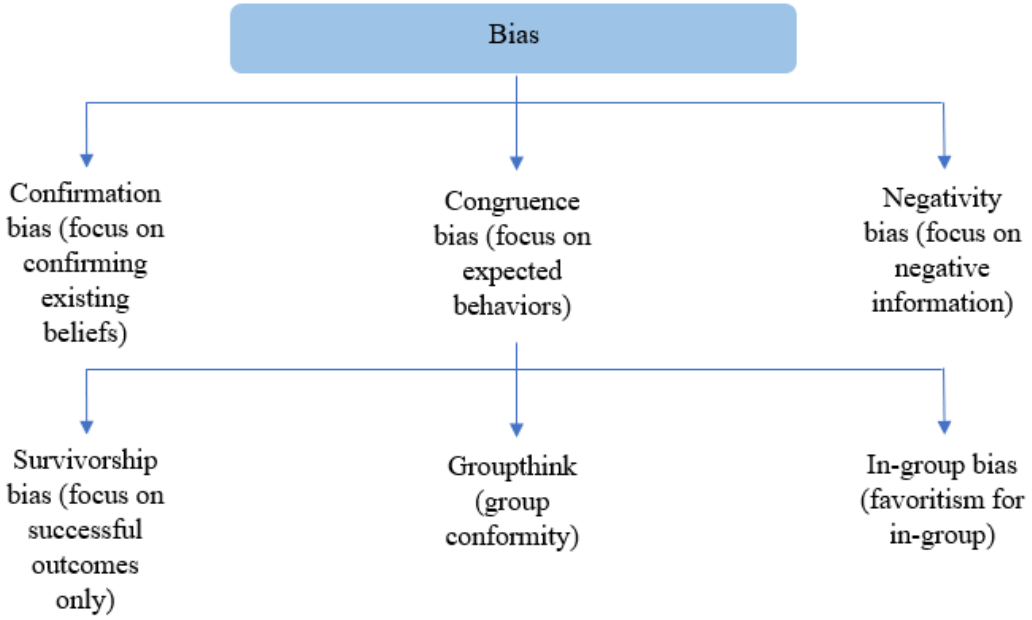



Figure 2.1. Summary of types of human bias

## 2.4. Framework For Software Testers

A framework for software testing is a standardized approach that provides guidelines, protocols, and procedures for conducting testing activities efficiently and effectively. It is a reusable set of components, tools, and libraries that simplify the testing process and allow testers to concentrate on the most important areas of testing. A testing framework, according to Kaner et al. (1999), is a set of assumptions, concepts, values, and practices that constitute a way of viewing testing. Organizations may increase the consistency and accuracy of testing, minimize testing time and effort, and improve the overall quality of the software product by employing a testing framework. There are different types of software testing frameworks that testers can choose based on project requirements and objectives. In a keyword-driven framework, testing actions are represented by keywords that are used to create test scripts. Because this method

requires minimal programming knowledge, non-technical testers can perform testing more easily. In contrast, data-driven frameworks separate test data from test scripts and use external data sources, such as spreadsheets and databases, to execute tests. Test maintenance is simplified, and testing coverage is more comprehensive with this approach. In addition to keyword-driven and data-driven frameworks, a hybrid framework provides greater flexibility and control than either alone.

Testing frameworks are used by testers today in a variety of ways. Selenium, an open-source web application testing framework that automates web browsers across different platforms and programming languages, is one of the most popular frameworks. Using Selenium, you can write test scripts in any programming language and run them against any browser, which increases test coverage while reducing test time, according to Myers et al. (2012). Appium is another popular framework for mobile application testing that supports multiple platforms, including iOS and Android, as well as programming languages such as Java, Python, and Ruby. In addition to test configuration, test grouping, and reporting, TestNG is another popular framework for testing Java applications.

In conclusion, a testing framework is an essential tool for software testers to ensure that testing is carried out efficiently and effectively. Frameworks provide a standardized approach to testing, which can help improve the quality and reliability of software products, reduce testing time and effort, and simplify test maintenance. The use of popular testing frameworks, such as Selenium, Appium, and TestNG, has become increasingly widespread, allowing testers to take advantage of powerful testing tools and libraries to achieve their testing goals.

### 2.5. Conceptual Framework: Testing and Bias

According to the theory of building cause-and-effect relationships, a moderating variable can affect the strength or direction of the relationship between two other variables (Baron & Kenny, 1986). In the context of software testing, the attributes of a software tester can be considered an independent variable, while software testing quality can be seen as a dependent variable. However, the presence of human bias in software testing can complicate this relationship by acting as a moderating variable.

Baron and Kenny (1986) also address the fact that a moderating variable can either strengthen or weaken the relationship between the independent and dependent variables. In the case of software testing, human bias can moderate the relationship between the attributes of a software tester and software testing quality. For example, a software tester who possesses strong technical and soft skills may still produce lower-quality tests if they are influenced by human biases. Conversely, a software tester who has weaker technical and soft skills may still produce higher-quality tests if they are not affected by human biases. Therefore, the presence or absence of human bias can moderate the relationship between the attributes of a software tester and software testing quality (Figure 2.2).

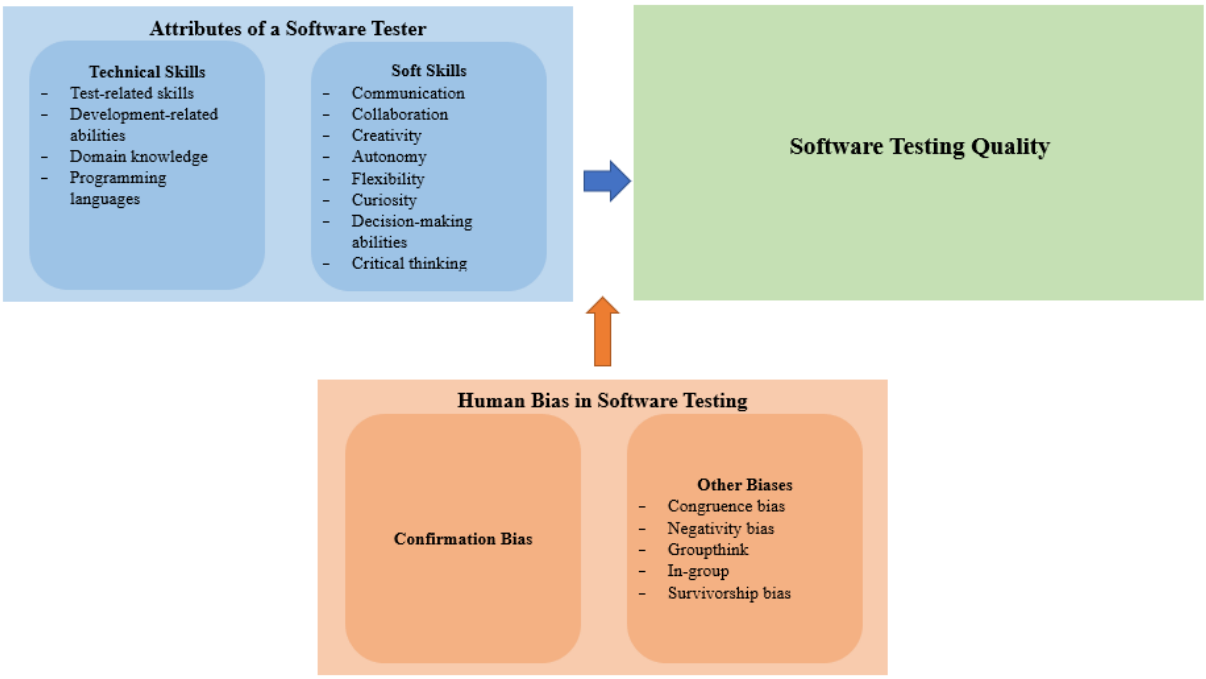



Figure 2.2. Conceptual Framework of Testing and Bias

## 2.6. Summary

Software testing is a crucial aspect of software development aimed at enhancing product quality and user experiences. It encompasses various approaches, including test planning, analysis, generation, implementation, and reporting. Early adoption of quality assurance practices, such as testing, is vital to preventing costly delays and ensuring effective software development. However, research on software testing is relatively limited compared to other software development professions, leading to a lack of recognition for testing teams.

Organizational factors significantly influence software testing, with priorities often conflicting between meeting deadlines and upholding quality standards. Inexperienced employees may not be adequately considered for verification and validation tasks, undermining

the importance of testing in the development process. Nonetheless, software testing roles provide opportunities for learning and development, attracting individuals with diverse perspectives on the cognitive challenges and career progression in testing jobs.

Various testing types play a crucial role in ensuring software quality. Functional testing verifies that software functions as intended, while performance testing assesses scalability and speed. Security testing identifies vulnerabilities and safeguards user data, and usability testing ensures software is user-friendly. To enhance testing effectiveness and efficiency, different techniques have been developed, including agile testing, continuous testing, exploratory testing, and risk-based testing.

Software testers possess essential attributes and skills to fulfill their roles successfully. Effective communication is crucial for providing progress updates, reporting issues, and fostering collaboration. Testers need to exhibit creativity to identify uncommon bugs and generate alternative solutions. They should also possess organizational abilities, attention to detail, domain expertise, and autonomy. Other critical attributes include flexibility, curiosity, decision-making abilities, critical thinking, and eagerness to learn.

However, various negative factors can impact testing quality. Testers may face cognitive challenges due to workload, stress, and psychological strain. Operational factors like task scheduling, team interaction, and autonomy can also influence testing outcomes. Understanding these factors is vital for creating an environment conducive to effective software testing.

In conclusion, software testing is an essential step in software development to ensure quality, functionality, performance, security, and usability. Testing approaches and techniques, combined with diverse tester attributes and skills, contribute to the success and reliability of software products. Recognizing the importance of testing and addressing negative factors can significantly enhance software testing outcomes, ultimately leading to dependable and fit-for-purpose software solutions.

# 3. Methodology

In this section, the methodology used for the study is discussed. First, the decision to do qualitative research will be examined. Following that, it will be detailed how the data was collected and, finally, how it was analyzed.

## 3.1. Empirical Research Methodology

### 3.1.1. Selecting a Methodology for Research

Quantitative techniques are generally established approaches for quantifying social phenomena and evaluating hypotheses using fixed variables. Because of their standardized measurements, they may be used in relatively large samples (Silverman, 2006) and aid in the discovery of generalizable data (Patton, 2002). However, critics of quantitative approaches may claim that studies in this field have little or no contact with participants and that variables are often defined at random (Silverman, 2006). Furthermore, certain phenomena or social processes are just not quantifiable with numbers, statistics, or random sampling. In such circumstances, quantitative approaches may limit the possibility of detecting some characteristics of these events (Silverman, 2006).

Qualitative research is defined as any type of study that delivers results that are not obtained using statistical techniques or other methods of quantification (Strauss & Corbin, 1992). As a result, rather than figures, the emphasis is on in-depth comprehension of words, ideas, and experiences. Furthermore, qualitative methodologies focus on the individual rather than the generic (Mayring P, 2008). The majority of qualitative research is inductive. Although supported by a theoretical framework, the facts, not the theory, should guide the investigation (Taylor et al., 2015). Because of the researcher's engagement in a very open study, qualitative research may be seen as rather subjective. Furthermore, because qualitative research focuses on individual situations, generalization may be limited, and samples are rarely chosen at random. However, analyzing and examining a specific example and setting is the primary purpose of qualitative research, rather than generalizing to a population (Bryman, 2010).

The research approach chosen is mostly determined by the nature of the research issue. Qualitative approaches appear to be a good alternative for exploratory investigations like this thesis (Strauss & Corbin, 1992). Instead of using data to quantify the phenomenon of software testing, this thesis uses open-ended questions to investigate the viewpoints of software testers. Aside from the exploratory nature of the study, it nonetheless employs current theories to comprehend the notion of software testing. However, the primary focus of this research is inductive, with the goal of developing an understanding of software testers' views by

investigating the process and issues of software testing. These claims were not predictable for the researcher; therefore, it was not considered prudent to test predicted variables, as doing so may result in the omission of important concerns connected to the subject matter.

### 3.1.2. Research Method description

The thesis uses qualitative approaches to investigate research issues in an exploratory manner. Exploratory research, according to Popper (2005), is used to illustrate how effectively a theory can explain notions. Furthermore, it should attempt to give a new perspective on the problem or a new explanation that may have been ignored in the theoretical study (Popper, 2005). Turner (2014) defines three types of interviews: casual conversation interviews, generic interview guide approaches, and standard open-ended interviews. It is difficult to compare different replies since the casual conversation technique necessitates the interviewer's considerable flexibility in constructing questions as the interview progresses (Turner, 2014). The general interview guide method and standardized interview, on the other hand, are more organized in that the questions are prepared ahead of time. The distinction between the two is the degree of latitude afforded to the interviewer (Turner, 2014). To explain, Gall et al. (2007) state that the questions in the standardized technique are fully organized, whereas the material acquired in the general interview approach is more conversational (Turner, 2014). In terms of interview design forms, the general interview guide method and the standardized open-ended interview employed in this research work are mixed based on the interviewees' replies and time constraints. As a consequence, this semi-structured technique aims to contribute to the literature study and inquiry from the present empirical situation as well as include new perspectives from competent experts in the area. The preparation of question guidelines, investigation and collection of interviewees, documentation, and data analysis are all part of the interview process.

In some circumstances, group interviews (e.g., focus group sessions) can be useful. However, this seemed less suited for the research questions, as variables may be a very personal topic that informants would be less willing to discuss in the company of coworkers. Therefore, individual interviews with a semi-structured approach were chosen. Another explanation is that data from completely unstructured interviews may have been more difficult to integrate, but findings from fully structured interviews may have overlooked crucial informant perspectives that did not fit under any of the interview guide's items.

Following consent, all interviews were videotaped and transcribed verbatim. The transcripts were occasionally emailed to each participant for confirmation, along with clarifying

questions. This procedure allowed participants to alter the transcripts or make necessary corrections. The final transcripts were utilized for data analysis later.

### 3.2. Interview design

#### 3.2.1. Analysis unit

In research techniques, the unit of analysis is an important factor. The unit of analysis in this study is software testing professionals. The researchers used a qualitative study technique to investigate the important characteristics of a software tester, with an emphasis on comprehending the experiences and viewpoints of software testing professionals. Individual, in-depth interviews were employed to collect data for the study, largely through semi-structured interviews with participants. The interviews were designed to elicit information about the interviewees' testing experiences, current work, and challenges in the tester position, as well as opinions on testing processes and strategies, bug analysis techniques, and solutions for negative outcomes. Semi-structured interviews provided us greater flexibility in asking and probing while still ensuring that the major study topics were covered. This method yielded extensive and detailed data for analysis, allowing the researchers to delve into the experiences and opinions of software testing professionals, the study's unit of analysis.

#### 3.2.2. Data collection

As previously stated, data were gathered through individual in-depth interviews, primarily semi-structured interviews. The semi-structured interviews allowed for flexibility in the questioning and probing of participants while still ensuring that the main research questions were addressed. In addition to the semi-structured interviews, unstructured questions were utilized to gather additional information and insights. The use of unstructured questions allowed participants to freely express their opinions and perspectives on the topic, providing rich and nuanced data for analysis. Due to the geographic dispersion of the study participants, all interviews were conducted online. Eight interviews were conducted via online video conference, using platforms such as Google Meet and Zoom, to provide a more personal touch to the interviews. One interview was conducted via audio call using the online application, which allowed for audio recording of the interviews. One interview was conducted in person, allowing for face-to-face interaction between the researcher and the participant. In total, nine interviews were conducted over the course of seven hours, with each interview lasting an average of 45 minutes. The interview design included four major components: an introduction and warm-up period; semi-structured interview questions; unstructured questions; and a closing

and debriefing period. The interview design was altered once after the first two interviews to improve the flow and effectiveness of the interview process. Overall, the data-gathering process employed in this study allowed for a thorough exploration of the research questions while also providing ample opportunities for participants to express their perspectives and opinions. The use of online interviews enabled access to a diverse range of participants, while the interview design facilitated in-depth and nuanced responses. The adjustments made during these updates will not impact the basic substance of the script, but the organization and sequence of the interview questions will be adjusted and enhanced.

To ensure that the interview process was conducted ethically and with the participants' consent, a short introduction and terms of confidentiality were prepared, outlining the purpose of the study and seeking the participants' permission to record the interviews. The introduction included a brief overview of the topic and the research questions that the study sought to explore. The interview questions were designed to elicit information about the interviewees' experiences with the testing process, their current work and challenges with the tester position, their opinions on testing processes and strategies, their techniques for bug analysis, and their solutions for negative outcomes. The interview structure began by seeking to understand the backgrounds of the interviewees and their past and current experiences with the tester position. This was followed by the main questions that sought to understand their opinions regarding the challenges they have faced during work, particularly in relation to human bias.

While the questions and order were pre-defined, the interview was flexible, allowing the participants to feel at ease and freely share their thoughts. The interview questions were adjusted according to the answers given by the interviewees to ensure that the study could capture rich and nuanced data. The participants' responses were recorded and transcribed for analysis. The final part of the interview included a thank-you message and a feedback session. The interviewees were given the opportunity to share any additional thoughts they may have had and provide feedback on their experience with the interview process. Any concerns or questions that arose during the interview were addressed in this feedback session. The latest updated interview script can be found in *Appendix A*. Overall, the interview process was designed to gather in-depth insights into the experiences and opinions of testers while also ensuring that the process was conducted ethically and with the participants' consent. During the course of interviews conducted for this research, it was observed that the respondents expressed varying levels of comfort with the languages used, with English, Vietnamese, and parts of German being the most commonly used languages. However, it should be noted that,

as this research is intended to be conducted and presented in English, all of the collected data was subsequently processed in English.

### 3.2.3. Data analysis

The dataset utilized for this study focuses on the knowledge and experiences of quality assurance engineers and software testers. Each interview serves as one data point, with individuals serving as the analysis's unit of analysis.

The recorded interviews were done with nine participants and were transcribed verbatim, one by one, immediately following the completion of each session. The participants were chosen based on their experience and competence in software testing and quality assurance and were recruited through industry networks. Notes and memoranda obtained during the interview were resorted to during the transcription process to ensure accuracy. Although a few minutes of recording were lost due to internet outages, the basic concepts were well covered.

It is vital to highlight that transcribing audio records is a difficult operation, as proven by DiCicco-Bloom and Crabtree (2006), who also suggest that the length of the recording may affect data accuracy.

Thematic analysis was chosen as the analytic approach in the second phase of dataset preparation because it was best suited to identifying material linked by a similar theme or concept (Gibbs, 2007; Guzmán et al., 2014). As a result, the interviews were re-read, examined for trends or patterns, and categorized according to the interview criteria.

Following that, the technique adopted began with sorting and rearranging the facts to match the two research objectives for discussion in Chapter 1. Any new ideas or unusual facts that surfaced during the process were analyzed and used in the debate and conclusion. Overall, the dataset contains a wealth of qualitative data on the experiences and viewpoints of software testers and quality assurance engineers, which may be used to drive future study and practice in this sector.

[This page was intentionally left blank]

## 4. Findings

This section's goal is to present an overview of the interviewees' histories as well as a synopsis of their important points based on the primary issues discussed in the interview guide. Although the questions were deliberately created to focus on the topic of software testing, some IT jargon was unavoidable owing to the technical nature of the subject. Certain sections of the interview have been modified for clarity to ensure that the major ideas are available to readers. *Appendix B* contains the whole interview transcript.

### 4.1. Interview Respondents

In-depth interviews, according to DiCicco-Bloom and Crabtree (2006), should be fairly homogeneous and share critical similarities related to the research question. As a result, the purpose is to collect ideas from both junior and senior testers who have experience with both manual and automated testing and are now working in the software industry, including individuals with job titles such as "software tester" or "quality assurance engineer". While their titles may differ, both individuals conduct comparable software testing duties, and hence both viewpoints are useful for this research. However, owing to time constraints and industry specifications, it is not feasible to discover all samples against the same backdrop. As a consequence, there are three factors to consider while looking for and hiring respondents for this empirical research.

First and foremost, the individual must be familiar with the software testing system and procedures. Second, the individual should have at least two years of job experience. The last criterion was to select interviewers who had prior expertise with both manual and automated testing. The interviewers are not required to work for an IT firm if their product is connected to the software. They should, however, have a firm grasp of testing concepts. The goal of widening the requirements is to gain additional viewpoints from all sides of the company and, as a result, to build a generic framework for testers. The respondents were recruited using these criteria through two major channels: LinkedIn and the network. Three respondents from the network were asked to participate in the qualitative research. LinkedIn is regarded as a powerful social network tool for reaching out to a wider range of individuals through personal weak relationships and invisible ties that meet the standards. To search for prospective profiles on LinkedIn, keywords such as "software tester", "quality assurance engineer", and "quality assurance tester" were utilized. After reviewing their profiles, the individuals were contacted on LinkedIn to introduce the aim of the thesis and request an interview. Out of the fifteen individuals contacted, six expressed a strong interest in the subject matter. Once the agreement

was obtained to record and use the interview for research purposes, an invitation was sent to each participant, detailing the interview purpose and including the interview questions and instructions. Due to the rigorous selection criteria and the challenge of finding suitable candidates who satisfy all requirements while living in various locations, only nine individuals were interviewed. Interviews took place during the peak holiday season of January to May 2023, adding to the recruiting challenges. The interviews were conducted anonymously, with only the linked firm's industry and the respondents' working positions recorded. All data has been filtered by deleting names and characteristics such as family concerns, desire for specific employment locations, or any other sort of data that may lead to the identification of the interviewee. An overview of their profiles and interview information can be found in Table 4.1.

Table 4.1. Summary of respondents

| ID | Position | Affiliated Company | Country base | Date | Length | Format |
|---|---|---|---|---|---|---|
| Speaker 1 | QA Engineer | FMCG Company A | Vietnam | 31/01/2023 | 44m | Audio call |
| Speaker 2 | Automation Test Engineer | Finance Company B | Vietnam | 10/02/2023 | 53m | Video call |
| Speaker 3 | QA Engineer | Tech Company C | Portugal | 16/02/2023 | 47m | Face-to-face meeting |
| Speaker 4 | Software Tester | Tech Company D | Germany | 15/03/2023 | 30m | Video call |
| Speaker 5 | Junior QA Engineer | Tech Company E | Germany | 20/03/2023 | 1h15m | Video call |
| Speaker 6 | Software QA Tester | Tech Company F | Canada | 02/04/2023 | 47m | Video call |
| Speaker 7 | QA Engineer Lead | E-Commerce Company G | The United States | 11/05/2023 | 46m | Video call |

Table 4.1. Summary of respondents (continue)

| ID | Position | Affiliated Company | Country base | Date | Length | Format |
|---|---|---|---|---|---|---|
| Speaker 8 | Junior QA Engineer | Tech Company H | Portugal | 12/05/2023 | 35m | Audio call |
| Speaker 9 | Senior Software Tester | Energy Solutions Company I | Vietnam | 15/05/2023 | 34ms | Video call |

Speaker 1 is a highly experienced QA engineer who is now employed at A, a reputable FMCG (fast-moving consumer goods) firm in Ho Chi Minh City, Vietnam. Speaker 1 boasts a lot of knowledge and skill in guaranteeing the quality and operation of diverse goods and systems, having over six years of professional experience in both manual and automated testing. Company A is one of the world's leading consumer packaged goods firms, and it initially opened its doors in Vietnam in 1995.

Speaker 2 is an accomplished automation test engineer with extensive experience in frontend and backend automation testing for online and mobile apps. Speaker 2 has four years of professional experience and has worked in a variety of industries in Ho Chi Minh City, Vietnam, establishing a thorough grasp of the automated testing scene. Speaker 2 is presently employed at Company B, a well-known Australian financial organization. The corporation is regarded as one of Australia's top four major financial institutions, with a substantial market capitalization, earnings, and customer base.

Speaker 3 is a highly talented QA engineer who is now employed by a world-renowned technology company situated in the United States. Search engine technology, online advertising, cloud computing, computer software, quantum computing, e-commerce, artificial intelligence, and consumer electronics are among company C's specialties. Speaker 3 has three years of professional experience in the technology business, working only as a QA engineer for numerous IT organizations in Lisbon, Portugal.

Speaker 4 is a highly skilled software tester with three years of experience in the sector, having worked in Germany, Australia, and India. Speaker 4's professional experience includes both manual and automated testing, demonstrating a diverse set of abilities and competence. Speaker 4's academic credentials are similarly excellent, as she holds both a bachelor's and a master's degree in computer science. Speaker 4 is currently employed by Organization D, a

respected and well-established organization that has been creating, developing, and producing sensor solutions for the traffic control and automotive sectors for over 24 years.

Speaker 5 is a very driven junior QA who is now employed in a small tech firm in Berlin, Germany. Prior to this, Speaker 5 spent two years as a software tester, refining skills that have proven helpful in her present capacity. Speaker 5 has also earned the ISTQB Foundation Certificate, indicating a thorough grasp of industry-accepted QA procedures. Company E provides a varied assortment of items in a variety of categories, providing Speaker 5 with new and challenging testing possibilities. Speaker 5 is dedicated to ensuring the company's goods exceed the highest standards of excellence, with a passion for quality and a commitment to always improve.

Speaker 6 is an experienced testing specialist with five years of experience working in both Vietnam and Canada. Speaker 6 has extensive experience with a variety of testing approaches, including manual, automated, and performance testing. Speaker 6 has developed a sharp eye for detail and a thorough knowledge of the necessity of QA in software development as a result of this breadth of expertise. Speaker 6 is currently employed as a software QA tester for an award-winning SaaS startup based in Toronto, Canada. With over 500 workers, company F specializes in assisting businesses grow by delivering creative software solutions.

Speaker 7 is a QA engineer lead at Company G, a Texas-based e-commerce business. He leads the agile software development process, troubleshoots failing acceptance criteria, and tests production deployments using Jenkins. Speaker 7 also builds out Postman collections for end-to-end sales funnel flows and utilizes Python BDD with Cucumber for automated scripts. Company G is committed to creating cutting-edge digital solutions and providing exceptional customer service, with activities spread across the United States and over a thousand employees.

Speaker 8 is a junior QA engineer who has been working at Tech Company H in Lisbon, Portugal, for 2 years. Prior to this role, she worked as a technician for 3 years; therefore, she is good at networking, operating system, and hardware troubleshooting. She holds an ISTQB Foundation Level certification. In her current role at Tech Company H, she performs manual testing following the waterfall methodology. Tech Company H is a US-based technology company that provides various software solutions and testing services, including automated testing, functional testing, non-functional testing, security testing, and managed testing. They serve customers in the banking, insurance, aerospace, and entertainment industries globally.

Speaker 9 is a senior software tester at Company I, an energy solutions firm based in the United States with an office in Ho Chi Minh City, Vietnam. He has over 8 years of experience

working in software quality assurance and testing. Specifically, he is proficient in developing test cases, performing functional testing, software testing, test execution, manual testing, and API testing using tools like Postman. Company I helps customers manage and navigate the complexities of the wholesale and retail electricity and natural gas markets. With hundreds of customers, Company I provides industry-leading cloud-based energy trading and risk management platforms and services.

### 4.2. Results Summary

The summary follows the primary format of the question guidelines, beginning with an exploration of the respondents' backgrounds and experiences with the organization. Following that, strategies and techniques for analyzing bugs are covered. Finally, issues concerning obstacles and unfavorable elements are addressed.

Although not all respondents were able to answer every question owing to various degrees of knowledge and interest, some gave excellent insights that contributed to a thorough grasp of the issues.

- *An overview of the typical duties and responsibilities of a team.*

Team responsibilities, including usual duties and assignments, vary among different organizations and projects. However, the interviews of nine speakers show that testing and QA are crucial and complex aspects of team responsibilities. From the transcripts, reveals several noteworthy trends and patterns within the context of software development teams, as discussed by the various speakers:

- Diverse Team Structures (Speaker 1, Speaker 8): The organizational structure of software development teams exhibits considerable variation across different companies. Speaker 1 emphasizes the presence of a dedicated QA team working alongside developers. In contrast, Speaker 8 mentions a team without developers where QA responsibilities are assigned to team members who review and approve changes made by developers.
- Emphasis on Collaboration and Communication (Speaker 2, Speaker 7): Effective collaboration and communication between QA teams and developers emerge as crucial factors. Speaker 2 highlights the need for communication and collaboration with developers to gain a comprehensive understanding of the project and ensure a balanced workload. Speaker 7 emphasizes the importance of regular meetings and inclusive test case reviews where input is sought from various stakeholders.

- Test Case Development as a Core Responsibility (Speaker 1, Speaker 2): The development of test cases assumes a prominent role among multiple speakers. Speaker 1, belonging to a QA team, is responsible for developing test plans, test cases, and testing documents. Speaker 2, also from a QA team, is involved in preparing test data, designing test cases, and performing automation testing.
- Integration of Manual and Automated Testing (Speaker 2, Speaker 7): The adoption of a combined approach comprising both manual and automated testing is prevalent. Speaker 2 mentions the initial manual testing phase to verify tickets before proceeding to automation testing. Speaker 7 discusses the process of manual testing followed by automation, highlighting the conversion of manual test cases into scripts.
- Focus on Regression Testing (Speaker 5): Speaker 5 specifically mentions running regression tests to ensure that new features do not introduce issues or disrupt existing functionalities. Regression testing is highlighted as an integral part of the QA process to maintain the stability of the software.
- Adoption of Agile Methodologies (Speaker 6, Speaker 7): Several speakers indicate adherence to Agile methodologies, particularly Agile with Scrum, to guide their software development and QA processes. Speaker 6 mentions following an Agile methodology with Scrum, while Speaker 7 refers to their team as an Agile team.
- Development of Internal Tools (Speaker 8): Speaker 8 discusses the development of internal tools tailored to the team's specific needs for bug tracking and feature request management. The team relies on these internal tools, built on proprietary platforms, to streamline the quality assurance process.
- Multi-Phased Testing Approaches (Speaker 9): Speaker 9 describes a four-phase testing process, including the unique UA testing phase, which involves additional verification before releasing the product to customers for final testing. This phase focuses on meeting customer acceptance criteria.

The identified trends and patterns, as elucidated by the respective speakers, underscore the significance of fostering collaboration, meticulous test case development, integration of manual and automated testing approaches, regression testing, adherence to Agile methodologies, and the development of tailored internal tools to enhance the quality assurance process within software development teams.

- *Challenges encountered by software testers in their work.*

Testing software applications presents a multitude of challenges for software testers, as indicated by the interviewees' perspectives. These challenges can be classified into technical and non-technical obstacles that affect the efficiency and effectiveness of the testing process.

From a technical standpoint, understanding application requirements emerges as a significant challenge. Speakers 2 and 3 highlight the complexity of comprehending intricate requirements, particularly when working with larger teams. Speaker 9 emphasizes the importance of comprehensive documentation and project-specific requirements to facilitate newcomers' understanding of the system.

Regression testing and workload pose additional technical challenges. Speaker 5 notes the need for regression testing in agile development, which increases the testers' workload. Speaker 7 adds that testing often uncovers issues that require collaboration with data experts or menu setup specialists for resolution.

Another technical challenge is the requirement of coding skills for automation testing, as mentioned by Speaker 4. This implies that testers need to possess adequate programming proficiency to effectively engage in automation testing.

On the non-technical front, effective communication and collaboration emerge as critical factors. Speakers 2, 6, 7, and 8 emphasize the significance of seamless communication and collaboration between testers, developers, and other teams. Speakers 2 and 6 encounter difficulties in working with developers and resolving conflicts, emphasizing the need for regular meetings and the ability to persuade stakeholders regarding identified issues. Speaker 7 emphasizes seeking assistance within the team during Agile meetings, while Speaker 8 faces challenges in communicating with developers located in different time zones.

The lack of cross-transferrable knowledge from other teams is highlighted by Speaker 3 as a non-technical challenge. Additionally, Speaker 9 mentions the time pressure faced by testers, wherein identifying and reporting bugs close to the release deadline requires swift testing and reliance on developers for bug fixes.

Balancing workload and quality is another non-technical challenge discussed by Speaker 1 and Speaker 5. Speaker 1 emphasizes the need to manage test cases within a two-week sprint cycle, which impacts the tester's ability to design effective test cases. Speaker 5 further stresses the challenge of maintaining a balance between timely product releases and ensuring quality.

To tackle the various difficulties and guarantee the prompt and high-quality completion of software projects, software testers must make clear and thorough documentation, effective communication and teamwork, and suitable resource allocation to handle workloads and time restrictions a priority. These observations altogether emphasize the multifarious nature of the

challenges experienced by software testers in their daily work, emphasizing the necessity for tactical methods and support structures to prevail over these impediments.

- *An overview of the test organization utilized in the software application testing process*

The analysis of the transcripts from the nine interviewees highlights the test organization utilized in the software application testing process. Several key trends and patterns emerged, revealing insights into the practices employed by the speakers.

Firstly, agile methodologies, particularly Scrum, were prominently mentioned by several speakers. Speaker 1 underscored the use of the Scrum framework, while Speaker 7 made references to Sprint meetings, encompassing sprint planning, retrospectives, and demos. Speaker 9 explicitly stated the adoption of Agile and the application of Scrum within their team. These findings indicate the widespread popularity and relevance of agile methodologies in the industry.

Communication and collaboration were consistently emphasized by multiple speakers. Speaker 2 stressed the significance of effective communication and collaboration within development teams. Speaker 7 mentioned the discussion of improvements and learning during Sprint meetings, reflecting the importance placed on collaborative problem-solving. Furthermore, Speaker 8 highlighted the regular stand-up meetings conducted to provide updates on release status, fostering communication among team members.

Diverse team structures were evident in the discussions. Speaker 2 specifically mentioned the presence of cross-functional teams, while Speaker 6 discussed the existence of sub-teams with distinct functions. This highlights the recognition that there is no one-size-fits-all approach to team structures in software development. Tailoring team structures to meet the specific needs of the project appeared to be a common practice among the interviewees.

Continuous improvement and learning were recurring themes in the discussions. Speakers 3, 4, and 5 emphasized the importance of investing in training and skill development for team members. These findings align with the principles of Agile methodologies, which emphasize valuing individuals and cultivating a culture of continuous learning and improvement.

Test case and documentation management practices varied among the speakers. Speaker 7 mentioned utilizing a wiki or Confluence-like tool to store test cases, providing convenient access through links embedded within individual tickets. This approach facilitates easy reference and promotes efficient test case management within the testing process.

It is worth noting that Speaker 8 stood out from the majority, as their team followed the waterfall methodology rather than agile. This finding illustrates the existence of different methodologies employed within the software development industry.

In conclusion, the analysis of the interview transcripts reveals the predominance of Agile methodologies, particularly Scrum, in the test organization utilized for software application testing. Effective communication, collaboration, diverse team structures, continuous improvement, and learning were highlighted as essential aspects of the testing process. Test case and documentation management approaches varied, with the use of wikis or Confluence-like tools mentioned. The existence of a team adhering to the waterfall methodology further underscores the diversity of practices within the field. These findings contribute to the academic understanding of the test organization utilized in the software application testing process and provide insights for future research and practice in the field of software development.

- *Quality assurance processes implemented to ensure high-quality software testing*

The analysis of interview transcripts reveals several recurring trends and patterns concerning the strategies implemented to ensure high-quality software testing. These findings shed light on effective approaches embraced by software development teams across various domains.

Collaborative testing emerges as a prominent trend among the interviewed professionals. A shared commitment to teamwork is evident, as highlighted by Speaker 1, who underscores the involvement of the entire team in reviewing test cases and plans. This collective effort fosters a collaborative environment, minimizing individual biases and enhancing defect detection capabilities. Furthermore, Speaker 2 emphasizes the significance of inter-team collaboration, particularly in the context of background testing, which entails comprehensive evaluations of an application's backward compatibility with previous versions. This collaboration ensures the thoroughness and accuracy of testing efforts.

Another noteworthy trend relates to the establishment of robust frameworks and processes. Speaker 1 and Speaker 3 draw attention to the importance of well-defined frameworks for achieving optimal software quality. Speaker 3 advocates for breaking down processes and steps, thereby enabling a systematic approach to testing. Such meticulous decomposition ensures adherence to established standards and facilitates comprehensive result analysis. In a similar vein, Speaker 1 highlights the implementation of the Scrum framework, which promotes team involvement throughout the testing process. By leveraging this framework, organizations foster transparency, mitigate biases, and bolster defect detection capabilities.

Both manual and automated testing are recognized as critical components of an effective testing strategy. Speaker 4 underscores the significance of manual testing, advocating for its prioritization prior to automation. This sequential approach ensures that bugs are identified and rectified before transitioning to automated testing. Speaker 9, on the other hand, favors a manual testing approach, rewriting test cases to align them with specific business requirements. This tailored approach enables meticulous examination and validation of software functionality.

The involvement of multiple testers is yet another discernible pattern. Speaker 5 emphasizes the importance of collaborative testing efforts involving multiple individuals. This multi-faceted approach guarantees comprehensive software examination, enhances defect detection rates, and ensures that issues are addressed proactively before software release.

Workload management emerges as a challenge in the testing process, as highlighted by Speaker 6. Late deliveries from developers often burden the QA team, necessitating effective workload distribution. Speaker 6 proposes a solution involving the division of assignments and incremental delivery of functionalities. This approach aims to alleviate the strain on the QA team and enhance efficiency in software testing endeavors.

In addition, the significance of integration testing is highlighted by Speaker 7, who underscores the integration of features from developers with other components. Although not explicitly emphasized by other speakers, integration testing plays a pivotal role in ensuring seamless software functionality and compatibility across various system elements.

Furthermore, the involvement of a quality assurance lead is found to be influential in decision-making processes. Speaker 8 highlights the pivotal role of the quality lead in assessing critical issues and making informed decisions regarding release blocking. The quality lead's extensive product knowledge and expertise serve as valuable resources, aiding in the overall management of software quality.

In summary, the analysis of the interview transcripts reveals several trends and patterns within the realm of software testing. Collaboration among team members, the establishment of robust frameworks and processes, the integration of manual and automated testing approaches, the involvement of multiple testers, effective workload management, integration testing, and the guidance of a quality assurance lead are identified as key elements in ensuring high-quality software testing. These findings provide valuable insights for software development teams seeking to optimize their testing practices and enhance software quality assurance processes.

- *Software testing techniques for bug identification, analysis, and documentation.*

The interview transcripts encompass a range of methodologies and approaches for effective bug identification, analysis, and documentation during software testing. The summarized

information from the previous interviews highlights various techniques employed by different speakers, and the newly added transcripts contribute additional perspectives to this discussion.

Speaker 1 emphasizes the significance of testing early and frequently, including unit testing and integration testing, to catch defects at the initial stages of development. They also advocate for the use of automation tools to expedite the testing process.

Speaker 2 introduces exploratory testing as a valuable technique that enables testers to explore the software without predefined test plans. This method allows for greater flexibility and creativity, helping uncover flaws that traditional testing techniques might overlook.

Speaker 3 underscores the importance of security vulnerability testing, which involves the use of tools such as code analysis and penetration testing to identify potential software flaws. They highlight the need to address security concerns and mitigate vulnerabilities.

Speaker 4 contributes the concept of manual testing, which involves comparing expected outputs with the actual results to identify discrepancies and potential bugs. This approach applies to both manual and automated testing.

Speaker 5 highlights performance testing as a critical technique for ensuring software functions well under various loads and stresses. Scaling testing is recommended to assess the software's capability to handle increasing demands as the user base expands.

Speaker 6 discusses testing methodologies such as black box testing, white box testing, and gray box testing. They recommend combining these methods to gain a comprehensive understanding of the software and identify possible flaws. Specific black box testing techniques, like equivalence partitioning and boundary value analysis, are mentioned.

Speaker 7 introduces the notion of "black box testing" emphasizing the importance of validating the expected results based on user requirements. The focus is on ensuring successful user interactions with the software without concern for the underlying code.

Speaker 8 provides insights into their team's bug identification and analysis techniques. Their approach involves manual testing using Excel for reviewing and documenting issues, with automation tools aiding in gathering information. Collaboration takes place through Google group chat, and all data is stored in Google Drive for evaluation and performance assessment. They highlight the team's focus on quality and understanding how features should trigger from an end-user perspective.

Speaker 9 reinforces the use of techniques such as equivalence partitioning, boundary value analysis, and decision table testing for writing test cases. Their team relies on Jira as a tool to store test cases and related information.

Taking into account the collective insights from all speakers, it is evident that a thorough and iterative testing process is necessary to identify, analyze, and document software bugs effectively. Various techniques and methodologies, including early testing, exploratory testing, security vulnerability testing, performance testing, and a combination of black box, white box, and gray box testing, contribute to this comprehensive approach. Additionally, the importance of considering end-user perspectives, leveraging automation tools, and employing structured test case design methodologies is underscored.

- *Strategies for accurate bug analysis and reporting.*

The interview transcripts cover several strategies for ensuring accurate bug analysis and reporting during software testing. The previous summarized information highlighted four key areas: test planning, testing techniques, communication, and documentation. The newly added transcripts contribute additional perspectives to this discussion.

Speaker 1 emphasizes the importance of developing a comprehensive test plan that encompasses various testing methodologies, such as white box testing, black box testing, mobile device or web testing, unit testing, functional and non-functional testing, regression testing, exploratory testing, smoke testing, sanity testing, ad-hoc testing, user interface testing, automation testing, performance and load testing, and security testing. Defining and applying these techniques in the test plan facilitates the accurate identification and reporting of bugs.

Speaker 2 introduces the monkey test strategy as a valuable technique for validating significant features. This approach involves testing the app with the mindset of a user who knows nothing about technology, randomly interacting with the app to uncover potential issues.

Speaker 3 emphasizes the importance of neutrality in terms of localization and cultural differences during bug analysis and reporting. They highlight the need to document bugs with clear and understandable details, describing issues in the simplest form to ensure clarity for everyone involved.

Speaker 4 emphasizes the necessity for effective communication between testers and developers. They suggest establishing a philosophical relationship between testers and developers to foster understanding and efficient communication.

Speaker 5 underscores the significance of collaborating with the product owner (PO) and project manager (PM) to determine the criticality of bugs. Additionally, they recommend documenting and addressing minor bugs that may not be critical but are important to the tester.

Speaker 6’s insights contribute to both testing techniques and documentation strategies. They suggest following the main flow of a function and dividing the equivalence area to define

test cases accurately. Furthermore, they highlight the importance of relying on expected results from the requirements to define test cases, ensuring that the function works as intended.

Speaker 7 introduces the concept of limiting test coverage and seeking input from developers regarding areas of focus during testing. This collaborative approach allows testers to align their testing efforts with the specific areas of code that developers have worked on. It helps in refining the testing strategy and ensuring comprehensive coverage.

Speaker 8 emphasizes the significance of following the guidelines provided by the company, which offer step-by-step instructions and troubleshooting guidance. When necessary, they recommend consulting with seniors or quality leads to ensure the correctness of decisions made during bug analysis and reporting.

Speaker 9 highlights the importance of reducing subjectivity when writing test cases by following the mentioned techniques. They emphasize the need for self-research and learning, including topics like API testing, to enhance testing skills. They also stress the importance of being careful and meticulous in their work.

Considering the collective insights from all speakers, the strategies for accurate bug analysis and reporting encompass test planning, testing techniques, communication, and documentation. Test planning involves creating a comprehensive test plan that incorporates various testing methodologies. Testing techniques involve following the main flow of a function, dividing the equivalence area, and relying on expected results from requirements. Communication entails effective collaboration between testers, developers, and stakeholders. Documentation involves clear and detailed bug reporting that is understandable to others. Furthermore, the importance of neutrality, collaboration with the PO and project manager, following guidelines, and continuous learning are emphasized to enhance bug analysis and reporting processes.

- *Challenges encountered in mitigating human bias during software testing*

After analyzing the interview transcripts of the nine speakers, a comprehensive understanding of the challenges encountered in mitigating human bias during software testing has emerged. The trend reveals that software testing involves addressing the diverse needs of different users, dealing with technical limitations and personal biases, ensuring clear communication and understanding between different stakeholders, and fostering collaboration and communication within the software development team.

Speaker 1 underlined the importance of studying and considering various types of users during software development to ensure the program satisfies their demands and offers a satisfying user experience. Speaker 2 highlighted the challenge of identifying bugs related to

user interfaces, which can be missed due to differences in screen ratios on different devices and cross-platform web browsers.

Speaker 3 discussed the barriers faced by software testers in mitigating human bias, including technical and personal obstacles. They mentioned the difficulty of addressing colleagues' biases and the challenge of changing emotions or personalities that can impact their work. Additionally, Speaker 3 emphasized the need to be aware of one's own biases and frames of reference that may influence testing.

Speaker 4 emphasized the importance of clear communication and understanding between different stakeholders involved in software development. They highlighted the need to consult managers rather than developers when customer requirements arise to ensure effective collaboration and alignment of goals.

Speaker 5 pointed out the risk of having only the product owner read the test cases and decide on the project's status without involving the entire team. This highlights the importance of collaboration and communication within the software development team to ensure that diverse perspectives are considered.

Speaker 6 acknowledged the challenge of covering all possible cases for a function, even with extensive experience in creating test cases. They emphasized the need for thorough testing to mitigate biases and ensure comprehensive software quality.

Speaker 7 discussed the challenge of putting oneself in the customer's shoes during software testing. They mentioned the importance of simulating real-world scenarios and considering user actions based on visual cues and user interface elements.

Speaker 8 highlighted the difficulty in understanding user expectations due to the large user base. They emphasized the need to compare and consider user feedback to align the product with user needs. Speaker 8 also mentioned the challenge of following common-sense solutions, which may not be documented, leading to different debugging approaches and perspectives within the team.

Speaker 9 shared their experience of acting as a customer during software testing and mentioned that it has become less difficult for them over the years.

In summary, the challenges encountered in mitigating human bias during software testing encompass addressing the diverse needs of different users, dealing with technical limitations and personal biases, ensuring clear communication and understanding between stakeholders, fostering collaboration within the team, simulating real-world scenarios, and aligning user expectations. Effective communication, collaboration, and awareness of biases are crucial for mitigating human bias and ensuring comprehensive software quality.

- *Strategies for mitigating human bias in software testing*

The findings of the interview analysis conducted with nine software testing professionals shed light on the strategies employed to mitigate human bias in software testing. Through a systematic examination of the transcripts, several prominent themes emerged, providing valuable insights for promoting unbiased and objective testing practices. The identified themes encompass essential aspects that testing practitioners should consider to minimize the influence of human bias in their testing endeavors. These themes are outlined as follows:

- Rigorous testing process: One salient strategy highlighted by Speaker 1 emphasizes the implementation of a rigorous testing process that encompasses multiple stages and types of testing. This approach aims to ensure that software products meet desired quality standards and deliver a reliable user experience throughout their lifecycle. By adopting a systematic and comprehensive approach to testing, the inherent biases of testers can be mitigated, fostering a more objective evaluation of software systems.
- Input from diverse sources: Speakers 3, 6, and 9 underscored the significance of seeking input from diverse sources to combat human bias. By actively incorporating diverse perspectives and opinions, testing professionals can counterbalance their own biases and gain a comprehensive understanding of the software being evaluated. Engaging with various individuals and organizations broadens the scope of insights, facilitating a more objective and inclusive assessment of software quality.
- Supportive team culture: Speaker 4 emphasized the role of a supportive team culture in minimizing human bias during software testing. Collaborating with experienced developers and fostering an environment conducive to open communication enables new testers to learn from the collective experiences of their team members. This knowledge exchange helps mitigate the impact of individual biases, contributing to more objective testing outcomes.
- Promoting transparency: Speaker 5 highlighted the significance of promoting transparency within the testing team. This entails ensuring that all team members have equal access to pertinent information through transparent communication channels. By creating an atmosphere of open dialogue and shared information, potential biases stemming from information asymmetry can be mitigated. Promoting transparency facilitates a collective understanding of issues and aids in minimizing the influence of individual biases on testing outcomes.

- Regression testing: Another crucial strategy mentioned by Speaker 1 involves conducting regression testing after bug fixes. This practice aims to minimize the introduction of biases and errors during the software testing process. By systematically validating the software's stability and functionality after bug resolution, testing professionals can safeguard against potential biases that may arise from the fixing process itself. Regression testing serves as a critical safeguard, ensuring the ongoing objectivity and reliability of testing outcomes.

While the insights provided by Speakers 2, 7, and 8 did not neatly align with the previously identified themes, they contributed valuable perspectives to the overall understanding of strategies for mitigating human bias in software testing. Speaker 2 emphasized the value of user feedback through beta releases; Speaker 7 emphasized the importance of perseverance; and Speaker 8 stressed adherence to guidelines. Although these insights did not fit directly into the established themes, they offer supplementary viewpoints on strategies that complement the overarching themes identified in the interviews.

In conclusion, the analysis of the interview transcripts has elucidated significant strategies for minimizing human bias in software testing. By embracing the identified themes within their testing practices, professionals can strive towards enhanced objectivity, reduced biases, and ultimately improved software quality and reliability. These findings provide a foundation for further research and serve as valuable guidance for practitioners seeking to promote unbiased .

- *Potential barriers to eliminating human bias in software testing*

It is apparent that there are potential barriers that can prevent software testers from eliminating human bias during software testing. The most commonly mentioned barriers were time constraints, limited exposure to other opinions, and subjective tendencies.

Speaker 1 stated that the number of bugs and issues encountered during use can require QA to re-evaluate their testing approach, indicating that time constraints may hinder software testers from fully eliminating human bias during software testing. Speaker 2 also mentioned time as a potential barrier, citing that conducting testing with multiple parties and receiving feedback from non-tech departments can be time-consuming and create pressure.

Speaker 3 highlighted the importance of exposure to other opinions and avoiding echo chambers as potential barriers. Without exposure to diverse perspectives and frames of reference, software testers may not have the open mind and objectivity needed to eliminate their own biases.

Additionally, Speaker 2 mentioned subjective tendencies that may be affected by age and reduced curiosity. This suggests that subjective tendencies may also be a potential barrier to eliminating human bias during software testing.

To make a long story short, the potential barriers that can prevent software testers from eliminating human bias during software testing include time constraints, limited exposure to other opinions, subjective tendencies, and reduced curiosity due to age. It is important for software testers to be aware of these potential barriers and actively work to overcome them in order to effectively eliminate human bias during software testing.

- *Best practices for testers for reducing human bias in software testing*

After analyzing the statements made by the speakers, it can be concluded that effective testing is crucial for ensuring the quality of software. Speaker 1 emphasizes the importance of evaluating the effectiveness of the testing approach and making necessary adjustments to minimize the risk of bugs and issues. This is achieved by running the software through a trial period and conducting various tests. Speaker 2 highlights the significance of identifying the root cause of issues and addressing them promptly to prevent them from recurring. This involves analyzing data and using various tools to investigate the issue.

Speaker 3 emphasizes the necessity of testing automation in improving productivity, accuracy, and dependability. Automation also frees up testers' time to focus on more difficult testing jobs rather than repetitive activities.

Speaker 5 recommends using a risk-based approach to testing, which entails identifying and prioritizing the most crucial sections of software that require testing. This ensures that testing efforts are directed toward the areas most likely to have an influence on the user experience. Speaker 6 highlights the need for collaboration among stakeholders like developers, testers, and business analysts. Effective cooperation may assist in ensuring that everyone is on the same page with the project's goals and that difficulties are discovered and remedied as soon as possible.

In summary, effective testing necessitates a variety of methodologies, such as assessing the testing strategy, discovering the root cause of issues, automation, a risk-based approach, and stakeholder participation. Software testing may assist in guaranteeing that software satisfies the necessary quality requirements and offers a pleasant user experience by applying these procedures.

- *The impact of bias on decision-making*

Upon thorough examination and analysis of both the prior summary and the additional transcripts, it becomes evident that the impact of bias on decision-making encompasses

discernible trends and patterns. These trends shed insight into the consequences of bias, the influence of testing procedures, managerial implications, the release process, and the acknowledgment of human imperfection.

Firstly, all speakers unanimously recognize the substantial ramifications associated with allowing bias to permeate decision-making processes. Such consequences encompass an augmented likelihood of faults within the product or service, which invariably compromises overall quality. Moreover, instances of bias influencing decisions in the context of financial transactions, as exemplified by Speaker 2, can lead to the loss of assets, customer dissatisfaction, and even legal repercussions, thereby amplifying the significance of mitigating bias in decision-making.

Secondly, the pivotal role of testing procedures in decision-making is underscored by Speaker 1. The acknowledgment that bias can substantially impede the efficacy of testing and thereby compromise the quality of the end product or service underscores the imperative nature of identifying and addressing bias during the testing phase. This recognition accentuates the significance of impartiality and accuracy in testing to ensure reliable outcomes.

Thirdly, Speaker 7 illuminates the notion that the severity of consequences resulting from mistakes or biases may hinge upon managerial discretion and the perceived sensitivity of the product or service. Organizations exhibit varying degrees of leniency or strictness in response to errors, with some taking stringent measures, such as terminations for minor infractions, while others foster an environment conducive to learning and growth. Managerial influence thus plays a pivotal role in determining the ramifications associated with biases in decision-making.

Fourthly, Speaker 8 delves into the implications of blocking releases due to minor issues. Such actions engender disruptions, necessitate additional explanations, and impact diverse departments within the organizational framework. Striking a delicate balance between preventing issues and ensuring a seamless release process thus becomes a critical factor in decision-making processes.

Finally, Speaker 9 astutely acknowledges the inherent fallibility of human beings. Minor bugs that evade detection by customers may be deemed tolerable manifestations of human imperfection. However, errors pertaining to critical functions or acceptance criteria have far-reaching implications, including performance evaluations, erosion of credibility, and dissatisfaction among customers. This acknowledgement underscores the need to strike a judicious balance between acknowledging human limitations while maintaining high standards of performance and customer satisfaction.

In summary, these discernible trends and patterns accentuate the imperative for organizations to cultivate awareness regarding potential biases in decision-making processes, particularly within the testing phase. It becomes incumbent upon organizations to implement strategies that minimize biases and their adverse consequences. This may encompass initiatives such as creating consciousness among decision-makers, fostering a culture of learning from mistakes, and accounting for variances in the ramifications of errors based on the nature of the product or service being delivered. The information related to the key aspects of the impact of bias is gathered and presented in Table 4.2.

Table 4.2. Key Aspects of the Impact of Bias

| Aspect | Summary of Key Points |
|---|---|
| Consequences | • Bias in decision-making leads to compromised quality of products or services, increasing the likelihood of faults. |
| | • Bias in financial transactions can result in loss of assets, customer dissatisfaction, and potential legal repercussions. |
| Testing Procedures | • Bias significantly hinders the effectiveness of testing, leading to compromised quality in the end product or service. |
| | • Impartiality and accuracy in testing are crucial to ensure reliable outcomes and mitigate the influence of bias. |
| Managerial Impact | • The severity of consequences resulting from biases depends on managerial discretion and the perceived sensitivity of the product or service. |
| | • Organizations exhibit varying levels of leniency or strictness in response to errors, influencing the impact of biases in decision-making. |
| Release Process | • Blocking releases due to minor issues can disrupt operations, necessitate additional explanations, and affect multiple departments within the organization. |
| | • Striking a balance between preventing issues and ensuring a seamless release process is essential for effective decision-making. |
| Human Imperfection | • Human fallibility is acknowledged, and minor bugs may be considered tolerable manifestations of imperfection. |
| | • The critical errors related to essential functions or acceptance criteria have significant consequences, including performance evaluations and credibility. |

Table 4.2. Key Aspects of the Impact of Bias (continue)

| Aspect | Summary of Key Points |
|---|---|
| Human Imperfection | • Human fallibility is acknowledged, and minor bugs may be considered tolerable manifestations of imperfection. |
| | • The critical errors related to essential functions or acceptance criteria have significant consequences, including performance evaluations and credibility. |

## 5. Discussion

This chapter will analyze the findings from the literature review in Chapter 2 and the qualitative research in Chapter 3. The objective is to assess whether the thesis addresses the research questions posed initially. In addition to addressing the two research questions, this chapter will summarize and present other relevant perspectives from the interview participants on the topic of attributes for software testers in software testing. At the end of the chapter, a general framework will be proposed based on the key findings.

### RQ1: What are the essential attributes required for a software tester to mitigate human bias during software testing?

To answer the research question, "What are the essential attributes required for a software tester to mitigate human bias during software testing?" the insights are compared from the literature review and the interview summary. This discussion aims to identify the essential attributes and their alignment between the two sources while also acknowledging additional insights from the literature review.

Firstly, it is important to note that software testers need to possess technical proficiency and a comprehensive understanding of the software application and its technical aspects (Florea & Stray, 2018). This point aligns with the interview speakers' emphasis on the challenges of understanding complex application requirements and the importance of comprehensive documentation (Speaker 2; Speaker 3; Speaker 4; Speaker 9).

Effective communication and collaboration skills are essential attributes mentioned in both the literature and the interview. The literature emphasizes the need for software testers to have strong communication skills to build a collaborative work atmosphere and avoid conflicts (Ahmed et al., 2012; Deak, 2014). This aligns with the interview speakers' emphasis on seamless communication and collaboration between testers, developers, and other stakeholders (Speaker 2; Speaker 4; Speaker 6; Speaker 7; Speaker 8).

Adaptability and continuous learning are emphasized in both the literature and the interview. Testers need to be adaptable to different methodologies and techniques and have a willingness to upgrade their skills to keep up with evolving industry practices (Florea & Stray, 2019). This aligns with the interview speakers' emphasis on the importance of continuous improvement, learning, and investing in training for team members (Speaker 3; Speaker 4; Speaker 9).

Meticulousness and attention to detail are recognized as essential attributes for software testers in both the literature and the interview. The literature emphasizes the importance of

attention to detail in areas such as evaluating user experience and gaining technical expertise (Kanij et al., 2014; Stray et al., 2021). This is consistent with the interview speaker's emphasis on the need for meticulousness in test case design and bug identification (Speaker 9).

Time management skills and workload balancing are identified as important attributes for software testers in both the literature and the interview. The literature highlights the need for strong time management skills and effective task prioritization (Kaner et al., 1999), which aligns with the interview speakers' discussions on managing time pressure and balancing workloads (Speaker 1; Speaker 5; Speaker 6).

Neutrality and objectivity are crucial attributes mentioned in both sources. Testers should strive to maintain objectivity and avoid letting personal preferences or biases influence their testing process (Salman et al., 2019). This aligns with the interview speaker's emphasis on neutrality in bug analysis and reporting, considering cultural differences, and avoiding personal biases (Speaker 3).

Test case design skills are recognized as an essential attribute for software testers in both the literature and the interview. The literature emphasizes the importance of designing effective test cases for comprehensive testing coverage (Florea & Stray, 2019), which is consistent with the interview speakers' emphasis on structured test case design techniques (Speaker 6; Speaker 9).

In addition to the common themes, the literature review introduces additional insights into essential attributes for software testers. Autonomy, flexibility, curiosity, decision-making abilities, and eagerness to learn are mentioned as important attributes for testers. However, these attributes were only briefly mentioned in the discussion and would benefit from further elaboration with examples and evidence from the literature.

Furthermore, it is important to acknowledge that the interview speakers provided practical insights based on their experiences, offering a more specific and firsthand perspective. While the discussion primarily focuses on comparing the common themes between the literature and the interviews, it is also essential to highlight any contrasting views or additional points made in the interviews that were not covered in the literature. Exploring these differences can provide a more comprehensive analysis of the essential attributes required to mitigate human bias during software testing.

In conclusion, the comparison between the interview summary and literature review reveals several essential attributes for software testers to mitigate human bias during software testing. Technical proficiency, communication and collaboration skills, adaptability, attention to detail, time management, neutrality, and test case design skills are consistently emphasized in both

sources. The additional insights from the literature review, including autonomy, flexibility, curiosity, decision-making abilities, and eagerness to learn, further enhance our understanding of the essential attributes. Recognizing and cultivating these attributes can help software testers address the various forms of human bias discussed in the literature and contribute to more effective and unbiased software testing practices.

**RQ2: What framework can be deployed to support software testers in eliminating human bias during software testing?**

The research question addressed in this thesis is: "What framework can be deployed to support software testers in eliminating human bias during software testing?" To answer this question, we conducted interviews with nine speakers who provided valuable insights into the challenges encountered in mitigating human bias during software testing and identified the key factors that contribute to bias in the testing process.

Based on the interview analysis, several challenges related to human bias in software testing were identified. First, addressing the diverse needs of different users emerged as a significant challenge. Speakers emphasized the importance of studying and considering various types of users during software development to ensure that the program satisfies their demands and offers a satisfying user experience (Speaker 1).

Another challenge mentioned was the identification of bugs related to user interfaces, which can be missed due to differences in screen ratios on different devices and cross-platform web browsers. This highlights the technical limitations faced by software testers and the need for comprehensive testing strategies (Speaker 2).

The interviews also revealed personal biases as a significant obstacle to mitigating human bias during software testing. Testers face challenges in addressing colleagues' biases and the impact of emotions or personalities on their work. It is crucial for testers to be aware of their own biases and frames of reference that may influence their testing approach (Speaker 3).

Effective communication and understanding between different stakeholders involved in software development emerged as an essential factor in eliminating human bias. Clear communication channels and alignment of goals between testers, developers, and managers are crucial to ensuring that customer requirements are effectively addressed (Speaker 4).

Collaboration and communication within the software development team were also emphasized as critical to mitigating human bias. The involvement of the entire team in test case reviews and decision-making processes helps to ensure that diverse perspectives are considered, minimizing the risk of biased decision-making (Speaker 5).

The challenge of simulating real-world scenarios and considering user actions based on visual cues and user interface elements was discussed as an important aspect of mitigating bias. By incorporating realistic scenarios into testing, testers can better understand and evaluate the software's performance from the user's perspective (Speaker 7).

Furthermore, aligning user expectations was identified as a challenge due to the large user base and varying expectations. Testers need to consider and compare user feedback to ensure that the product meets user needs. Additionally, following common-sense solutions, even if not explicitly documented, can help eliminate biases and encourage diverse perspectives within the testing team (Speaker 8).

Considering the challenges identified, a framework can be deployed to support software testers in eliminating human bias during software testing. The literature review highlighted the significance of testing frameworks in providing standardized approaches, guidelines, and procedures for conducting testing activities efficiently and effectively (Kaner et al., 1999). Testing frameworks can help increase consistency and accuracy, minimize testing time and effort, and improve the overall quality of software products.

Different types of testing frameworks were discussed, including keyword-driven, data-driven, and hybrid frameworks. Each framework has its advantages and can be chosen based on project requirements and objectives. For example, keyword-driven frameworks simplify testing for non-technical testers by representing testing actions with keywords. Data-driven frameworks separate test data from test scripts, enabling comprehensive test coverage and simplified test maintenance. Hybrid frameworks offer greater flexibility and control by combining the strengths of both keyword-driven and data-driven approaches.

In practice, popular testing frameworks such as Selenium, Appium, and TestNG have gained significant traction. Selenium, an open-source web application testing framework, enables test scripts to be written in any programming language and executed across different browsers, increasing test coverage while reducing test time (Myers et al., 2012). Appium, on the other hand, is a popular framework for mobile application testing that supports multiple platforms and programming languages. TestNG is widely used for testing Java applications and provides features such as test configuration, grouping, and reporting.

In conclusion, the identified challenges in mitigating human bias during software testing encompass addressing diverse user needs, technical limitations, personal biases, clear communication between stakeholders, collaboration within the software development team, simulating real-world scenarios, and aligning user expectations. To address these challenges, software testers can leverage testing frameworks such as Selenium, Appium, and TestNG,

which provide standardized approaches and tools to support efficient and effective testing (Kaner et al., 1999; Myers et al., 2012). By adopting appropriate testing frameworks and integrating them into their testing processes, testers can enhance the quality and reliability of software products while minimizing human bias. However, it is important to note that while testing frameworks can provide valuable support, they should be used in conjunction with effective communication, collaboration, and awareness of biases to achieve comprehensive software quality.

**Developing a Framework: Mitigating Bias in Software Testing**

Based on the analysis of the challenges in mitigating human bias during software testing and the literature review on testing frameworks, a general framework can be proposed to support software testers in addressing bias and promoting comprehensive software quality. This framework incorporates the key factors identified in the discussion section and leverages the advantages of testing frameworks.

The proposed general framework for software testers to eliminate human bias during software testing includes the following components:

I. *Awareness and Education:*

1. Familiarization with Biases:
    - Acquire knowledge of various types of biases that can influence software testing.
    - Explore relevant research, articles, and resources focused on biases in testing.
2. Continuous Education:
    - Stay updated on the latest developments in the field of bias in testing.
    - Participate in training sessions or workshops specifically designed to recognize and address biases in testing.

II. *Test Planning and Strategy:*

1. Understanding Project Requirements:
    - Develop a thorough understanding of project requirements and objectives.
2. Collaboration with Stakeholders:
    - Engage in collaborative discussions with stakeholders to establish clear and unambiguous acceptance criteria.
3. Comprehensive Test Strategy:

- Formulate a comprehensive test strategy that takes into account potential biases and proposes methods to mitigate them.

III. *Test Design:*

1. Systematic Test Case Design:
    - Utilize systematic approaches, such as equivalence partitioning, boundary value analysis, or decision table testing, for test case design.
2. Scenario Coverage:
    - Ensure that test cases cover a wide range of scenarios, including edge cases and negative scenarios.
3. Avoid personal assumptions:
    - Guard against relying solely on personal assumptions or prior knowledge when designing test cases.

IV. *Test Execution:*

1. Consistency in Test Execution:
    - Follow predefined test scripts or test cases to maintain consistency and minimize personal biases during execution.
2. Avoid personal beliefs and preferences:
    - Refrain from making assumptions or drawing conclusions based on personal beliefs or preferences.
3. Accurate Test Result Documentation:
    - Document test results accurately, including any deviations from expected outcomes.

V. *Data Collection and Analysis:*

1. Objective data gathering:
    - Collect objective data and metrics throughout the testing process.
2. Minimize subjective bias:
    - Employ automated testing tools to minimize subjective bias in test execution and result collection.
3. Statistical Analysis:
    - Analyze the data objectively using statistical analysis techniques to derive valid conclusions.

VI. *Peer Review and Collaboration:*

1. Regular peer reviews:

- Engage in regular peer reviews of test plans, test cases, and test results.

2. Encourage Diverse Perspectives:
    - Foster a collaborative environment that encourages open discussions and diverse perspectives.
3. Seek Feedback:
    - Solicit feedback from team members to identify potential biases and areas for improvement.

VII. *Independent Verification and Validation:*

1. Involvement of External Parties:
    - Consider involving external parties, such as third-party testers or auditors, for an unbiased evaluation of the software.
2. Leverage external expertise:
    - Leverage the expertise of external parties to identify blind spots or biases that internal teams might overlook.

VIII. *Continuous learning and improvement:*

1. Stay Updated:
    - Continuously update knowledge of testing methodologies, tools, and industry trends.
2. Address Emerging Biases:
    - Stay informed about new biases that emerge in the field and strategies to address them.
3. Reflection and Improvement:
    - Regularly reflect on testing processes, identify areas where biases may have influenced decisions, and implement improvements.

IX. *Ethical Considerations:*

1. Professional Ethics and Standards:
    - Adhere to professional ethics and standards throughout the testing process.
2. Fair Treatment:
    - Avoid conflicts of interest and ensure fair treatment for all stakeholders.
3. Transparency and Integrity
    - Maintain transparency and integrity in reporting test results.

By following this general framework, software testers can effectively address the challenges associated with human bias during software testing and ensure comprehensive

software quality. The framework emphasizes collaboration, communication, bias awareness, and leveraging testing frameworks to streamline and enhance the testing process.

## 6. Conclusion

In conclusion, the research highlights the essential attributes required for software testers to mitigate human bias during software testing. These attributes include technical proficiency, communication and collaboration skills, adaptability, attention to detail, time management, neutrality, and test case design skills. By cultivating these attributes, software testers can contribute to more effective and unbiased software testing practices.

The discussion also reveals the challenges associated with mitigating human bias in software testing, such as addressing diverse user needs, overcoming technical limitations, managing personal biases, fostering clear communication, promoting collaboration, simulating real-world scenarios, and aligning user expectations. Understanding and recognizing these challenges is crucial for testers to effectively address bias in their testing processes.

Testing frameworks, such as Selenium, Appium, and TestNG, are valuable tools that can support software testers in eliminating human bias. These frameworks provide standardized approaches, guidelines, and procedures for efficient and effective testing, enhancing consistency, accuracy, and overall software quality.

Based on the insights from the discussion, a general framework is proposed to support software testers in addressing bias and promoting comprehensive software quality. This framework includes components such as requirement analysis, test strategy development, test planning and design, test execution, collaboration and communication, bias awareness and mitigation, user feedback analysis, and continuous improvement. By following this framework, testers can effectively address bias and ensure comprehensive software quality.

Based on the discussions, a general framework is proposed to support software testers in addressing bias and promoting comprehensive software quality. This framework encompasses components such as requirement analysis and user research, test strategy development, test planning and design, test execution, collaboration and communication, bias awareness and mitigation, user feedback analysis, and continuous improvement. Adhering to this framework can enable testers to effectively address bias and ensure comprehensive software quality.

Recognizing and cultivating the essential attributes, understanding the challenges, and implementing a comprehensive framework can help software testers mitigate human bias during software testing, leading to improved software quality and user satisfaction.

### 6.1. Implications

The discussions presented important implications regarding the essential attributes required for software testers to mitigate human bias during software testing. The identified attributes include

technical proficiency, communication and collaboration skills, adaptability, attention to detail, time management, neutrality, and test case design skills. These attributes play a crucial role in ensuring comprehensive and unbiased testing practices (Deak, 2014; Florea & Stray, 2018; Graham & Fewster, 2012; Li et al., 2020; Salman et al., 2019).

Moreover, the discussions shed light on the challenges associated with mitigating human bias in software testing. These challenges encompass addressing diverse user needs, overcoming technical limitations, managing personal biases, fostering clear communication between stakeholders, promoting collaboration within the development team, simulating real-world scenarios, and aligning user expectations (Speakers 1–8). Understanding and recognizing these challenges is pivotal for software testers to effectively address bias in their testing processes.

Testing frameworks emerged as valuable tools to support software testers in eliminating human bias during software testing. Frameworks such as Selenium, Appium, and TestNG provide standardized approaches, guidelines, and procedures for efficient and effective testing (Kaner et al., 1999; Myers et al., 2012). By leveraging appropriate testing frameworks, testers can enhance consistency, accuracy, and overall software quality while minimizing human bias.

Based on the discussions, a general framework can be proposed to support software testers in addressing bias and promoting comprehensive software quality. This framework encompasses components such as requirement analysis and user research, test strategy development, test planning and design, test execution, collaboration and communication, bias awareness and mitigation, user feedback analysis, and continuous improvement. Adhering to this framework can enable testers to effectively address bias and ensure comprehensive software quality (Deak, 2014a; Florea & Stray, 2018; Graham & Fewster, 2012; Kaner et al., 1999; Li et al., 2010; Myers et al., 2012; Salman et al., 2019).

Implementing the insights from the discussions, including the essential attributes, challenges, and proposed general framework, holds the potential to enhance the effectiveness and objectivity of software testing practices. By recognizing and cultivating the identified attributes, software testers can mitigate bias and contribute to the delivery of high-quality software products. Additionally, the adoption of appropriate testing frameworks, along with effective communication, collaboration, and bias awareness, can further augment the comprehensive elimination of human bias during software testing.

### 6.2. Limitations

While the proposed framework and attributes provide valuable insights into mitigating human bias during software testing, several limitations constrain the generalizability and application of these findings. A nuanced understanding of these limitations is critical for interpreting and utilizing the results effectively.

Firstly, the reliance on qualitative data from interviews and literature limits quantitative measurement of the framework's effectiveness or impact. Qualitative methods yield rich insights but cannot determine quantifiable outcomes or changes. Incorporating quantitative data through surveys or standardized metrics may yield more robust evidence to substantiate the benefits and value of this approach. A mixed-methods design integrating qualitative and quantitative data could provide a comprehensive understanding of both the depth of the challenges involved in bias mitigation as well as the measurable impacts of the proposed solutions.

Secondly, the attributes identified represent only a subset of all characteristics that may influence a tester's ability to mitigate bias. The selection and framing of these attributes are subject to the researchers' own biases and perspectives, introducing potential gaps or oversights. A more systematic identification of attributes through expanded literature reviews, Delphi studies, focus groups, or secondary analyses of related frameworks could elicit a wider range of relevant characteristics. Regular reviews of new and emerging attributes are also needed to maintain currency. Researchers must practice reflexivity to limit the effect of their own biases when developing and applying new frameworks.

Finally, the complex and dynamic nature of bias mitigation necessitates continuous evaluation and adaptation of any proposed framework. As new technologies emerge, definitions of bias evolve, and industry standards change, the components and attributes required to support unbiased software testing may need adaptation. An iterative, cyclical approach emphasizing constant review of new tools, research, and work practices as well as soliciting stakeholder feedback allows a framework to remain responsive and optimal. Without diligent monitoring and flexibility, there is a risk of the framework becoming static, outdated, or irrelevant.

While limitations exist, this discussion aims to inform a balanced and critically thoughtful application of the proposed framework. By triangulating the data, expanding reviews of relevant attributes, and iteratively revising the framework as needed, researchers and practitioners can work to comprehensively address limitations over time. Overall, developing a nuanced understanding of constraints and reflexively enhancing solutions will support the

delivery of unbiased and ethical software systems. Continued progress requires diligent and iterative work.

### 6.3. Future Work

To enhance the rigor and applicability of bias mitigation frameworks in software testing, further research is warranted. The current study relied primarily on qualitative data, limiting quantifiable measures of the framework's outcomes and impacts. Future research should incorporate quantitative evaluation methods, such as surveys and standardized metrics, to assess the effectiveness of bias mitigation efforts more systematically. A mixed-methods approach integrating qualitative and quantitative data would provide a holistic understanding of the challenges of mitigating bias and the effects of proposed solutions.

The attributes identified in this study represent a subset of characteristics that can influence a tester's ability to mitigate bias, selected and framed according to the researchers' perspectives. To improve generalizability, future research should employ more systematic methods to elicit a wider range of relevant attributes, such as expanded literature reviews, Delphi studies, focus groups, or analyses of adjacent frameworks. Regular reviews of new and emerging attributes are also needed to ensure currency. Researchers must apply reflexivity to limit the influence of their own biases when developing and revising frameworks.

Reflexivity necessitates acknowledging and critically reflecting on one's own biases and preconceptions. Future research should explore strategies to enhance reflexivity, such as including diverse perspectives, seeking external evaluation, or implementing rigorous review processes. By minimizing the impact of researchers' biases, proposed frameworks may become more objective, comprehensive, and robust. Researchers should continuously evaluate and monitor their own biases to develop ethical and unbiased frameworks.

The dynamic nature of bias mitigation requires continuous evaluation and adaptation of proposed frameworks. As technologies emerge, definitions evolve, and standards change, framework components and attributes must remain responsive. Future research should emphasize an iterative, cyclical approach, continuously reviewing tools, findings, practices, and stakeholder feedback to inform framework revisions. Without diligent monitoring and flexibility, frameworks risk becoming outdated, static, or irrelevant.

In summary, future research directions should address limitations in the current study by: incorporating quantitative methods; expanding identification of attributes through systematic methods and continuous review; enhancing reflexivity to minimize researchers' biases; and emphasizing an iterative, responsive process to adapt frameworks to changes. Such research

may substantively contribute to delivering unbiased, ethical software systems. Continued progress will require persistent and collaborative efforts across disciplines.

[This page was intentionally left blank]

## 7. References


Ahmed, F., Capretz, L. F., & Campbell, P. J. (2012). Evaluating the Demand for Soft Skills in Software Development. *IT Professional*, *14*(1), 44–49. https://doi.org/10.1109/mitp.2012.7

Andersson, C. D., & Runeson, P. (2002). *Verification and validation in industry - a qualitative survey on the state of practice*. https://doi.org/10.1109/isese.2002.1166923

Baron, R. M., & Kenny, D. A. (1986). The moderator–mediator variable distinction in social psychological research: Conceptual, strategic, and statistical considerations. *Journal of Personality and Social Psychology*, *51*(6), 1173–1182. https://doi.org/10.1037/0022-3514.51.6.1173

Beecham, S., Baddoo, N., Hall, T., Robinson, H. P. C., & Sharp, H. (2008). Motivation in Software Engineering: A systematic literature review. *Information & Software Technology*, *50*(9–10), 860–878. https://doi.org/10.1016/j.infsof.2007.09.004

Beizer, B. (1990). *Software Testing Techniques*. Van Nostrand Reinhold Company.

Bertolino, A. (2007). *Software Testing Research: Achievements, Challenges, Dreams*. https://doi.org/10.1109/fose.2007.25

Bryman, A. (2010). Social Research Methods. In *Taylor & Francis eBooks* (pp. 157–184). https://doi.org/10.4324/9780203381175_chapter_9

Burnstein, I. (2006). *Practical Software Testing: A Process-Oriented Approach*. Springer Science & Business Media.

Calikli, G., & Bener, A. (2015). Empirical analysis of factors affecting confirmation bias levels of software engineers. *Software Quality Journal*, *23*(4), 695–722. https://doi.org/10.1007/s11219-014-9250-6

Capretz, L. F., Varona, D., & Raza, A. (2015). Influence of personality types in software tasks choices. *Computers in Human Behavior*, *52*, 373–378. https://doi.org/10.1016/j.chb.2015.05.050

Capretz, L. F., Waychal, P. K., Jia, J., Varona, D., & Lizama, Y. (2019). *Studies on the Software Testing Profession*. https://doi.org/10.1109/icse-companion.2019.00105

Case Study Research in Software Engineering: Guidelines and Examples. (2012, March 7). John Wiley & Sons. http://ci.nii.ac.jp/ncid/BB15466883

Correia, S., & Silveira, C. (2009). A ergonomia cognitiva, operacional e organizacional e suas interferências na produtividade e satisfação dos colaboradores. Encontro Nacional de Engenharia de Produção, 29.

Crispin, L., Lisa, & Gregory. (2008). Agile Testing: A Practical Guide for Testers and Agile Teams.

Cruz, S. S. J. O., Da Silva, F. Q. B., & Capretz, L. F. (2015). Forty years of research on personality in software engineering: A mapping study. *Computers in Human Behavior*, *46*, 94–113. https://doi.org/10.1016/j.chb.2014.12.008

Cunningham, S. P., Gambo, J., Lawless, A., Moore, D., Yilmaz, M., Clarke, P., & O'Connor, R. J. (2019). Software Testing: A Changing Career. In *Communications in computer and information science* (pp. 731–742). Springer Science+Business Media. https://doi.org/10.1007/978-3-030-28005-5_57

Deak, A. (2014a). A Comparative Study of Testers' Motivation in Traditional and Agile Software Development. In *Lecture Notes in Computer Science* (pp. 1–16). Springer Science+Business Media. https://doi.org/10.1007/978-3-319-13835-0_1

Deak, A. (2014). What Characterizes a Good Software Tester? – A Survey in Four Norwegian Companies. In *Springer eBooks* (pp. 161–172). https://doi.org/10.1007/978-3-662-44857-1_11

Deak, A., Stålhane, T., & Cruzes, D. S. (2013). *Factors Influencing the Choice of a Career in Software Testing among Norwegian Students*. https://doi.org/10.2316/p.2013.796-032

Deak, A., Stålhane, T., & Sindre, G. (2016). Challenges and strategies for motivating software testing personnel. *Information & Software Technology*. https://doi.org/10.1016/j.infsof.2016.01.002

DiCicco-Bloom, B., & Crabtree, B. F. (2006). The qualitative research interview. *Medical Education*, *40*(4), 314–321. https://doi.org/10.1111/j.1365-2929.2006.02418.x

Dosaj, C. R. (n.d.). ISTQB Foundation Sample Exam Questions Certified Tester Foundation Level (CTFL) 2018 Syllabus.

Felderer, M., Büchler, M., Johns, M., Brucker, A. D., Breu, R., & Pretschner, A. (2016). Security Testing. In *Advances in Computers* (pp. 1–51). Elsevier BV. https://doi.org/10.1016/bs.adcom.2015.11.003

Fernandez-Sanz, L., Villalba, M., Hilera, J. R., & Lacuesta, R. (2009). Factors with Negative Influence on Software Testing Practice in Spain: A Survey. In *Springer eBooks* (pp. 1–12). https://doi.org/10.1007/978-3-642-04133-4_1

Florea, R., & Stray, V. (2018). Software Tester, We Want to Hire You! an Analysis of the Demand for Soft Skills. In *Lecture notes in business information processing* (pp. 54–67). Springer Science+Business Media. https://doi.org/10.1007/978-3-319-91602-6_4

Florea, R., & Stray, V. (2019). The skills that employers look for in software testers. *Software Quality Journal*, *27*(4), 1449–1479. https://doi.org/10.1007/s11219-019-09462-5

Frost, J. (2022). Survivorship Bias: Definition, Examples & Avoiding. *Statistics by Jim*. https://statisticsbyjim.com/basics/survivorship-bias/

Gall, M. D., Gall, J. P., & Borg, W. R. (2007). *Educational Research: An Introduction*. Pearson.

Glass, R. L., Collard, R. M., Bertolino, A., Bach, J., & Kaner, C. (2006). Software testing and industry needs. *IEEE Software*, *23*(4), 55–57. https://doi.org/10.1109/ms.2006.113

Gonçalves, W., De Almeida, C. C., De Araujo, L. S., Ferraz, M. S., Xandu, R. B., & De Farias, I. H., Junior. (2017). *The influence of human factors on the software testing process: The impact of these factors on the software testing process*. https://doi.org/10.23919/cisti.2017.7975873

Graham, D., & Fewster, M. (2012). *Experiences of Test Automation: Case Studies of Software Test Automation*. Addison-Wesley Professional. pp. 38-40.

Gren, L. (2017). *The Links Between Agile Practices, Interpersonal Conflict, and Perceived Productivity*. https://doi.org/10.1145/3084226.3084269

Guzmán, L. J., Lampasona, C., Seaman, C., & Rombach, D. (2014). *Survey on research synthesis in software engineering*. https://doi.org/10.1145/2601248.2601273

Gibbs, G. R. (2007). Thematic Coding and Categorizing. In *SAGE Publications Ltd eBooks* (pp. 38–55). https://doi.org/10.4135/9781849208574.n4

Haertl, R., Thomsen, M. S., & Ostermayer, M. (2016). Continuous testing: A practical guide with concepts and tools. *IEEE Software*, 33(3), pp. 80-86.

Shah, H., & Harrold, M. J. (2010). *Studying human and social aspects of testing in a service-based software company*. https://doi.org/10.1145/1833310.1833327

Iivonen, J., Mäntylä, M. V., & Itkonen, J. (2010). *Characteristics of high performing testers*. https://doi.org/10.1145/1852786.1852862

ISTQB, (2015) Glossary of testing terms. ISTQB Glossary https://glossary.istqb.org/en/search/. Accessed Feb 2017

*ISTQB Glossary*. (n.d.). https://glossary.istqb.org/en/search/

Janis, I. L. (1972). *Victims of Groupthink: A Psychological Study of Foreign-policy Decisions and Fiascoes*. Houghton Mifflin.

Kaner, C., Falk, J., & Nguyen, H. Q. (1999). *Testing Computer Software*. John Wiley & Sons.

Kanij, T., Merkel, R., & Grundy, J. (2014). *A Preliminary Survey of Factors Affecting Software Testers*. https://doi.org/10.1109/aswec.2014.32

Li, P. C. H., Ko, A. J., & Begel, A. (2020). What distinguishes great software engineers? *Empirical Software Engineering*, *25*(1), 322–352. https://doi.org/10.1007/s10664-019-09773-y

Lizama, Y., Varona, D., Waychal, P. K., & Capretz, L. F. (2020). The Unpopularity of the Software Tester Role Among Software Practitioners: A Case Study. In *Springer eBooks* (pp. 185–197). https://doi.org/10.1007/978-3-030-36518-9_7

Mäntylä, M. V., Itkonen, J., & Iivonen, J. (2012). Who tested my software? Testing as an organizationally cross-cutting activity. *Software Quality Journal*, *20*(1), 145–172. https://doi.org/10.1007/s11219-011-9157-4

Martin, D. C., Rooksby, J. H., Rouncefield, M., & Sommerville, I. (2007). *"Good" Organisational Reasons for "Bad" Software Testing: An Ethnographic Study of Testing in a Small Software Company*. https://doi.org/10.1109/icse.2007.1

Matturro, G. (2013). *Soft skills in software engineering: A study of its demand by software companies in Uruguay*. https://doi.org/10.1109/chase.2013.6614749

Mayring P. (2008). Qualitative Inhaltsanalyse : Grundlagen und Techniken. *CiNii Books*. http://ci.nii.ac.jp/ncid/BA89966921

Mohanani, R., Salman, I., Turhan, B., Rodriguez, P., & Ralph, P. (2020). Cognitive Biases in Software Engineering: A Systematic Mapping Study. *IEEE Transactions on Software Engineering*, *46*(12), 1318–1339. https://doi.org/10.1109/tse.2018.2877759

Mullen, B., Brown, R., & Smith, C. (1992). Ingroup bias as a function of salience, relevance, and status: An integration. *European Journal of Social Psychology*, *22*(2), 103–122. https://doi.org/10.1002/ejsp.2420220202

Myers, G. J., Sandler, C., & Badgett, T. (2012). The Art of Software Testing (3rd ed.). John Wiley & Sons.

Nikolopoulou, K. (2023). What Is Negativity Bias? | Definition & Examples. *Scribbr*. https://www.scribbr.com/research-bias/negativity-bias/

Ogbari, M. E., & Borishade, T. T. (2015). Strategic Imperatives of Total Quality Management and Customer Satisfaction in Organizational Sustainability. *International Journal of Academic Research in Business & Social Sciences*, *5*(4). https://doi.org/10.6007/ijarbss/v5-i4/1538

Paruch, L., Stray, V., & Blindheim, C. (2020). *Characteristic Traits of Software Testers*. https://doi.org/10.1145/3383219.3383270

Patton, M. Q. (2002). *Qualitative Research & Evaluation Methods*. SAGE.

Hernandez, T. P. R. Y., & Marsden, N. (2014). *Understanding Software Testers in the Automotive Industry - A Mixed-method Case Study*. https://doi.org/10.5220/0004992503050314

Popper, K. (2005). The Logic of Scientific Discovery. In *Routledge eBooks*. https://doi.org/10.4324/9780203994627

Rooksby, J., Rouncefield, M., & Sommerville, I. (2009). Testing in the Wild: The Social and Organisational Dimensions of Real World Practice. *Computer Supported Cooperative Work*, *18*(5–6), 559–580. https://doi.org/10.1007/s10606-009-9098-7

Salman, I., Turhan, B., & Vegas, S. (2019). A controlled experiment on time pressure and confirmation bias in functional software testing. *Empirical Software Engineering*, *24*(4), 1727–1761. https://doi.org/10.1007/s10664-018-9668-8

Sánchez-Gordón, M., Rijal, L., & Colomo-Palacios, R. (2020). *Beyond Technical Skills in Software Testing*. https://doi.org/10.1145/3387940.3392238

Shah, H., Harrold, M. J., & Sinha, S. (2014). Global software testing under deadline pressure: Vendor-side experiences. *Information & Software Technology*, *56*(1), 6–19. https://doi.org/10.1016/j.infsof.2013.04.005

Shenk, G., & Shenk, G. (2022). The Impact of Cognitive Bias on Software Testing. *www.functionize.com*. https://www.functionize.com/blog/the-impact-of-cognitive-bias-on-software-testing#:~:text=Software%20testing%20is%20subject%20to,been%20colored%20by%20past%20experience.

Silverman, D. (2006). *Interpreting Qualitative Data: Methods for Analyzing Talk, Text and Interaction*. SAGE.

*Software Testing Outsourcing- QA Outsource*. (n.d.). https://www.appsierra.com/blog/software-testing-outsourcing

Strauss, A. L., & Corbin, J. (1992). Basics of Qualitative Research: Grounded Theory Procedures and Techniques. *Contemporary Sociology*, *21*(1), 138. https://doi.org/10.2307/2074814

Stray, V., Florea, R., & Paruch, L. (2021). Exploring human factors of the agile software tester. *Software Quality Journal*. https://doi.org/10.1007/s11219-021-09561-2

Tajfel, H., & Turner, J. A. (2004). The Social Identity Theory of Intergroup Behavior. In *Psychology Press eBooks* (pp. 276–293). https://doi.org/10.4324/9780203505984-16

Kanij, T., Merkel, R., & Grundy, J. (2015). An empirical investigation of personality traits of software testers. In *International Conference on Software Engineering* (pp. 1–7). https://doi.org/10.5555/2819321.2819323

Taylor, S. J., Bogdan, R., & DeVault, M. (2015). *Introduction to Qualitative Research Methods: A Guidebook and Resource*. John Wiley & Sons.

Turner, D. J. (2014). Qualitative Interview Design: A Practical Guide for Novice Investigators. *The Qualitative Report*. https://doi.org/10.46743/2160-3715/2010.1178

Whittaker, J. A., & Thompson, H. H. (2004). How to Break Software Security: Effective Techniques for Security Testing. Addison-Wesley.

Whittaker, J. (2009). Exploratory Software Testing (1st ed.). Pearson. Retrieved from https://www.perlego.com/book/883831/exploratory-software-testing-tips-tricks-tours-and-techniques-to-guide-test-design-pdf (Original work published 2009)

## 8. Appendices

### Appendix A – Interview script

| | |
|---|---|
| 1. Introduction | – Thank you for agreeing to work with me on this project. Hi, my name is Ngoc Tran. Today, I'll be in charge of our interview. As I said in my previous e-mail, I am collecting data to work on my in-company project for my Master's degree at ISCTE in Lisbon, Portugal. |
| | – The interview should last between 30 and 45 minutes. |
| | – Let me explain this project in more detail: I am now working on a project to learn more about the characteristics that a tester should possess to assist them throughout the bug analysis phase and eliminate human bias while making decisions. Therefore, I'd like to study how you overcame this difficulty and whether there is any opportunity for me to design a framework to assist your job. |
| | – Before we begin the interview, I'd like to inform you of the following privacy policies and optional participation:<br>• The information obtained from the interviews will be utilized to address the research question.<br>• Any personal information will be kept strictly secret.<br>• All participants will be kept anonymous.<br>• Participation in the interview is entirely optional, and participants may withdraw their agreement at any moment.<br>• The entire interview will be recorded, and the interviewee may refuse if they are uncomfortable. |
| 2. Background | 2.1. Could you tell me a bit about your experience as a software tester? |
| | 2.2. Could you please describe your team's responsibilities, including usual duties and assignments? And what challenges have you faced when testing software applications? |
| 3. Testing organization | 3.1. Could you explain the test organization you use when testing software applications? |

| | | |
|---|---|---|
| | 3.2. | What processes do you follow in order to ensure the highest quality of software testing? |
| 4. Bug analysis | 4.1. | What techniques do you use to identify, analyze, and document bugs during software testing? |
| | 4.2. | What strategies do you use to ensure accurate bug analysis and reporting? |
| 5. Barriers or negative aspects | 5.1. | What challenges have you encountered when trying to mitigate human bias during software testing? |
| | 5.2. | What strategies can be used to reduce human bias during software testing? |
| | 5.3. | What best practices can software testers follow in order to reduce human bias in software testing? |
| | 5.4. | What will be the consequences if you let bias affect your decision? |
| 6. Closing | | Thank you so much for your time. Before we finish, do you have any questions for me or anything else you want to discuss more? |

**Appendix B – Interview transcripts (only Q&A)**

**Speaker 1 - QA Engineer at FMCG A**

- *Background*

*Q: Could you tell me a bit about your experience as a software tester?*

*A:* It is a job that is easy to imagine but hard to be professional.

*Q: Why did you say so? Could you please explain me more?*

*A:* This has been true for me since I went to work with my colleague. He's the type of person who thinks you can learn QA on your own so just being a developer or someone who knows about IT without much knowledge can do it. People will think so at first. Because I only need to access the system and then I use it as a user and then I can test it to see if there is an error. But actually, in order for me to get to better levels, I have to have more knowledge. Instead of just testing by looking at the document and then looking at my visualization, the cases that I think it will have to test are wrong, I have to rely on the specialized knowledge, I have to learn based on the testing techniques. I'm sure in each test method in each different test strategy I

will use what testing techniques and then I can combine my experiences in my long-term processes. And understanding the business of the system is important.

*Q: What challenges have you faced when testing software applications? Describe your team's responsibilities, including usual duties and assignments.*

*A:* It is the challenge of work progress. We have to balance the timely release of the product and ensuring the quality of the product meets requirements, being usable with no major bugs. Regarding the team's responsibilities, the QA team is responsible for ensuring the completion and quality of the project when released. Detecting and consulting on resolution of issues during development and maintaining during user usage. In a project, a test plan, test case, and testing document will be established and the progress of development and testing will be monitored to promptly address any issues that arise in the project.

- *Testing organization*

*Q: Could you explain the test organization you use when testing software applications?*

*A:* Both manual testing and automation testing will be used, with testing level varying by project stage and type. API testing with postman, automation testing with selenium and cucumber.

*Q: What processes do you follow in order to ensure the highest quality of software testing?*

*A:* Scrum team. I am a QA and I report to my boss in my scrum, scrum master I have a commitment that this scrum team I will be writing test cases and test plans. After I finish writing, I will give it to the whole team to review together to see if it is enough or not. As well as I will reduce my subjectivity again. After each work, every day when I report to him, that person will know if his situation is okay, if there are any problems. Then at the end of each ring we will have a meeting to report on whether we did the right thing, whether the work is over, ready for the next rings or not.

*Q: With black box testing, how could you handle it?*

*A:* If I have a clear knowledge of test techniques, it will help me know what test cases need tweaking. If I'm subjective, I only do the cases which I could think about and can miss them. But when I can apply testing techniques, I will classify, group, and write separately for each module, that project I will test out, list all types of test cases, and test list I have to do. It's also very rare that it doesn't make mistakes or something. If it's too difficult or in cases where users rarely use it, it won't happen. And if I, if I really do it and now we do it, the subjective part is no longer there, then I have to follow the test case.

*Q: Who will create test cases?*

*A:* Yes, I have to make my own test plan, then I write my own test cases, I will prepare for it, then I will follow that test case. The part that subjectively only plays a small part in my testing.

After I have written all the things I have done, finished the test cases, then in my subjective opinion there are cases where I think it will fail, I will test it myself or I will pass it through special cases. But no longer for subjective testing, it is very rare.

*Q: How could you create test case by yourself?*

*A:* This test case I will be based on technical requirements. The first is to have the document of the finished product, then they will transfer it to the IT document. I could receive that request and analyze it, and the developer and QA side can talk to each other to come up with and write the cases.

- *Bug analysis*

*Q: What techniques do you use to identify, analyze, and document bugs during software testing?*

*A:* There are a lot. For e.g.:

- Equivalence partitioning: a software testing technique that involves dividing the input data of a system into partitions of equivalent data from which test cases can be derived.
- Boundary value: a value that represents the point between two adjacent partitions.
- Decision table: a table that summarizes the possible inputs, output and conditions of a system, used in functional and logic testing.
- Debugging: the process of identifying and fixing errors or bugs in software.
- Value pair: a pair of values used in testing, often representing the minimum and maximum values for a range of inputs.

*Q: What strategies do you use to ensure accurate bug analysis and reporting?*

*A:* The project will apply different testing strategies, which will be analyzed and listed in the test plan. The following testing techniques will be used:

- Define white box testing, black box testing, and mobile device or web testing
- Define unit testing
- Define functional and non-functional testing
- Define regression testing, exploratory testing, smoke testing, sanity testing, ad-hoc testing, etc.
- Define user interface testing
- Define automation testing
- Define performance and load testing
- Define security testing.

*Q: Which bugs are acceptable? And what is objective and subjective mistake?*

*A:* Acceptable bugs are usually small errors like the UI it's dangerous and it's not right for each type of device so it's dangerous but it's just a small error, not a mistake or you are I can fix it or not. As for subjective errors, unacceptable errors, if you make another part blocked or you can't display the information obtained from the parts you do, it is a subjective error, an objective error usually depends on the system or environment is different. Every time I pretend to be a guest, I just go to that environment to get an error, then I know, that's also called an error, I will have to fix it immediately. I will have to fix it immediately, not QA's fault. In the process of working, in the past, I also encountered some errors. Just made my own system especially making this equivalent module, I confirm it won't affect the APIs of other equivalent modules. Then I no longer have to double-check those parts. Then there are subjective errors that I check for my parity model and then I just check briefly for the side but I don't do it. But I do not check carefully in other cases. So when it's on the live environment, other equivalent modules will fail, people will call me, it will say that my equivalent module is affected. It's like I thought it didn't matter much so I thought it was okay so it was a subjective error

- *Barriers or negative aspects*

*Q: What challenges have you encountered when trying to mitigate human bias during software testing?*

*A:* It's important to analyze and consider multiple types of users when developing software:

- The users who request the software
- Users who have prior knowledge of the software
- Users who are unfamiliar with the product
- New users who are not yet familiar with technology.

This can help to ensure that the software meets the needs of a diverse range of users and provides a positive user experience for everyone.

*Q: What strategies can be used to reduce human bias during software testing?*

A: Apply a rigorous testing process, covering multiple stages and types of testing, from the initial project analysis to the final maintenance of the project. This can help to ensure that the software meets the desired quality standards and provides a reliable user experience throughout its lifecycle. When I found the error, I tested it again. I'm very subjective, there I just think I've tested it once and then I just need to retest this bug to make it right. But really, when I test or fix it, it's often wrong in other places. After I retest all those errors, I have to use regression test again. You can imagine that I will do it in a way that I will test the first time with the cases, the bugs, I will retest the developer's bugs once. After that time, I confirmed it was ok, all developer

bugs were dead, now I will do regression for the last time and I have to continue doing it for the last time.

*Q: Are there any potential barriers that can prevent software testers from eliminating human bias during software testing?*

*A:* The number of bugs and issues encountered during use will require QA to review and re-evaluate their testing approach to identify any areas that were not adequately covered and to improve the overall quality of the software.

*Q: What best practices can software testers follow in order to reduce human bias in software testing*?

A: Based on the test cases and the successful performance of the software during the trial period, QA can evaluate the effectiveness of their testing approach and make necessary adjustments to ensure that the software meets the desired quality standards. This can help to minimize the risk of future bugs and issues, and provide a reliable and positive user experience. I'm actually using it myself, before giving it to QA, the Dev also has to test the tests below the demo. Meanwhile, the demo is given to QA as a tissue map that it has to test all its bugs and key test cases that QA requires and then give it to QA. So when it's given to QA, QA only needs to run the flow end to end of QA and ad hoc tests of QA. It also has minor errors. So it will also be saved, it will run through the place of the Dev test, to the QA test and then to the PO verify. The BA is the one who makes the request, so they verify their request to see if it is correct, need to edit anything, or have any features, then they will go live.

*Q: What will be the consequences if you let bias affect your decision?*

*A:* Of course, the product will have many defects and bugs because when there are many users, each person will have a unique approach to the experience that is different from the QA team, resulting in a higher risk of defects.

**Speaker 2 - Automation Test Engineer at Finance Company B**

- *Background*

*Q: Could you tell me a bit about your experience as a software tester?*

A: My background is studying Electronics and Communication, then after I graduate from university, the one with the major is Electronics and Communication, I will go to work in accordance with production models in factories and companies. the company where they move production and either work on embedded systems, if you choose to work as QA for embedded systems, the first company for me to work after graduation is robotics is a company that there are many fields but my field is making embedded systems that the project I am involved in is

about the car airbag system. After I worked there for about a year, I decided that I would switch to QA for the web application field, since then I have worked for nearly 4 years in the web application field. and Fintech domains and it also has domains with Banking. For the domain of Banking, I made a product. It is an internet Tool for the banker side of the bank to help them solve and collapse customer loans quickly. As for the fintech project, I mainly do tax declaration through the main Open Apple network. During the process of taking care of those working processes, I will test both the Firmware and the Software loaded into the microcontroller. In addition, I also test the controller's board, but for the thing that in making a web application, I will also test the database, which means that the data in the system has valid or not about me. Backend activity I always test for UI activity. Most of my work, I will apply the black box technique mainly. which means I will not consider the structure of the software but I will rely on the system definition of the BA and I define the test case. But for the microservice model, it requires me to look at a part of the code of the system. Let me check how this system works and from there create some fake data to test because of it. It will be difficult to create real data in practice. For example, if there are trees where the server is down, I can't turn off the server. The method, I modified it, it's called gray Box, which means knowing a part of the code and applying black box testing to test

*Q: What challenges have you faced when testing software applications? Describe your team's responsibilities, including usual duties and assignments.*

A: Hmm. For example, when you receive a new feature, the first difficulty you need to meet is transferring the requirement. Requirement is a basic idea from the Project owner they push down to BA and BA they write into more detailed tickets. then the thing that understands the idea of my project owner, it is passed by a BA. Sometimes it will have to have a lot of meetings to be able to close the requirement because I have to involve verifying the design of the requirement from the very beginning. In some cases, if you are a new member of the team, it means that you have absolutely no prior knowledge about the product of your team that you enter. When the product is being built in the middle, it is very time consuming because I myself don't know what I need to learn. Therefore, I will usually book a private meeting with my BA so that I can help me. Write down the related things I need to learn so that I can do more of this feature. And in the process of starting the test, one difficulty is that I need to create the correct test data for it because in some cases the test data comes from another team. It means that my team is not able to create that, then I have to request another team but the owner of the other team, but they will also have urgent tickets from their side, so sometimes they can't support I can. At that time, I will raise to the team and Scrum master they will come out and arrange for

me to do it. Another difficulty is that when I raise a bug, in some cases when the requirement clears from the beginning, everything is very easy, because I know what it is but it will have cases. The requirement is that it is unclear how to handle it, so when I raise that bug, the developer side rejects it. People say this isn't a bug but in my view I see it as a bug. So my treatment at that time was that I would book a meeting to discuss and I would invite the BA and the developer and other people on the team to come in. I put one together so I can close. There was once a case where the argument still happened even after the meeting. At this point, my next action is that I invite more PMs or Scrum Master to come in, that person will decide whether it is a bug or not because they have influence.

*Q: How will you handle if the final decision is not like your decision?*

A: In terms of the way the bug works, they will have to go into that problem deeply. With that, they will only have the question of whether to fix it now or not to fix it. If I fix it, I accept it as a bug, and if I don't fix it, I don't recognize it as a bug. So if I go to fix it, what advantages do I get and what harm do I get? There will be some cases where they decide it's a bug but they don't fix it right away because they have to secure the release and they feel like everyone's sticking together is a minor issue. It is Ok I can fix it later. And if on the developer side They say it's not an issue then why isn't it an issue.

*Q: Could you please  your team responsibilities?*

A: To put it simply, the quality is that you have to make sure of it. If you talk more about the basic jobs, you can imagine the following. Dev has to report how much time and effort this ticket takes from the dev side and he also has to report how much time the ticket takes to test, in order for me to get a full picture, which tickets will I add to this scrum so that it will ensure my right workload. That was the first responsibility I did. The second responsibility is that while Dev is preparing the ticket, he has to prepare test data first. Test data comes in two forms. One, I can create it myself. The other two are created by another team in the same department. There is also a case where test data is created by a team outside the department. In that case you have to announce as soon as possible so that everyone can notify the outside team so they can prepare for you. The worst case is that the outside team is too busy, then QA has to set up a fake server so that they send fake data to me for me to simulate. Also, after I have the test data, I will design the test case and I will prepare the automation test for me. If possible if I don't have enough capacity then I have to Run half automation and half manual. After I test the tickets, I will have a step where I collect all those tickets. That means I tested each ticket individually before. After I collect it to create a version to deploy and I will test all of them again to see if there are any issues. Also there is a step that is user acceptance test which is the PO one will do. So I'll have

to book a meeting with the PO. Because the step it requires is what outsiders have to do, but surely the PO will not know how to test, then I will have to book a meeting and I test it for them to see it quickly, but I will show them which test case I need to run. They are there to see if I have finished testing, then they will confirm that it is ok and qualified to release to production. This way to ensure transparency. After I have the confirmation from the PO, I will one day run the regression Test which means I run the test cases of the features in the past, this is what I run internally. After all those steps are done, I will proceed to deploy. After deploying, I will go to production environment, I run a few cases called main case. Let me check if I am deploying the correct version or not. Because I will have a case where I collect tickets but when I push, I lack tickets. After everything Test is ok, then it is considered that I have finished that cycle, I prepared the scrums

- *Testing organization*

*Q: I think what you just described is also considered like test organization, right?*

A: Yes. Exactly!

*Q: What processes do you follow in order to ensure the highest quality of software testing?*

A: In addition to the process testing I often use, I can apply one more testing step they call background testing. It means that today my application is up to version 4.0, but other applications it is version 3.1 because each team has a different cycle of release, so I will have to revert to the old version of other teams. That means I will have to notify other teams And they revert back to their old model and I deploy my new version so that I can consider it even though I'm deploying the new version, but if other teams use the old version, they can bed affected or not.

- *Bug analysis*

*Q: What techniques do you use to identify, analyze, and document bugs during software testing?*

A: Usually people will use root cause analysis. For example, I have used a model called the herringbone model, which is often used in my team, in addition, it also has some other models 5Y, 8D. However, 8D is usually not used by my brother very often. Also, I don't always apply models. When it happens a bug, it's QA's duty to record how to reproduce the bug and create a ticket to assign to the developer. Also, if they request some test data for them to check, I will also provide them. If QA they don't have urgent tasks, they can put them in a common meeting to support developers debug. The first investigation is that it will come from BA's side because BA will read how to reproduce that bug first or not. If the requirement is not broken, then I will start with the next step. The next step is I will use that test data to reproduce the bug. Normally developers will usually use a method for them to run the code and while running the code it

checks in that the code it is running is running. There will be some tools to print the block line of the code, so they can check where this code runs and how much it returns the variable of the value and whether it returns false or not. which is not the right song. So it's also rare to have to apply a lot of techniques before but there will be cases when it's necessary.

*Q: Which factors always play an important role in these methods/techniques which you mentioned before?*

A: There is a common element about that environment. For example, many times there is a bug. It happens because another team is also using the same test environment, but they deploy something and they don't notice it, that's a factor. The new environment comes from the outside from the human factor. In addition, the element of its test environment is not stable. you very often say that often happens often those two things don't happen. As for the subjective factor of the tester, it is often due to the lack of cases by QA testing. And this instance is detected by users. Many times the design of UI / UX makes people not understand their design intentions, leading to them walking around the road and accidentally it has some threads that I don't cover, it is bugs.

*Q: What strategies do you use to ensure accurate bug analysis and reporting?*

A: Normally, on the QA side, they should always have a mindset that users they don't know anything about technology and don't know how to use the app. I should not have the subjective idea that people will always go the usual route. There's a test strategy it's called Monkey Test which means a strategy that I'm going to test with a kind of mindset that I don't know anything. I opened that app and I used it randomly in there. Usually monkey test is used for large features. Because this is the final step to validate a big feature. And usually Monkey tests are done by experts because they have been doing it for a long time and they have enough experience so that they don't suffer from the subjective factor. The testing process is that I will go up and use the app by all means and then I will have a meeting to.

*Q: How could you work without bias when you run Monkey test?*

A: That, first of all, comes mostly from my experience. The second one is because I also watch what other seniors do and I follow. Normally, testers need to be curious. They simply want to try this and they try these and they don't follow any procedure.

- *Barriers or negative aspects*

*Q: What challenges have you encountered when trying to mitigate human bias during software testing?*

A: It always challenges no matter how long I work as a QA. Bugs that are in the workflow, I can cover but some bugs are related to UI. Because web application is that it is used on cross-

platform web browsers like Fire fox, chrome, so I won't be able to cover that. In addition, it also goes with the screen ratio of different devices so it will fail but I can be sure that it is my fault that I let it pass through me after I do that test it's only minor issues.

*Q: What strategies can be used to reduce human bias during software testing?*

A: If the company has a big budget, the company can release a beta version for users. And users can report so they can receive a promotion or some kind of subsidy

*Q: Are there any potential barriers that can prevent software testers from eliminating human bias during software testing?*

A: I find the factor that hinders the most is that time. Because it's like when I conduct my testing I will need multiple parties if I feel that feature needs a lot of feedback from different non-tech departments. If the deal is too close, it will cause great pressure. In addition, the health factor of the person who does it is also important. I think its age will also be affected by the tendency it's subjective because they've done this too many times so the curiosity has been reduced.

*Q: What best practices can software testers follow in order to reduce human bias in software testing?*

A: Internal review. Two QA will cross-review the test case against each other. I can also book a meeting where I demo all the flows to other teammates. This gives them a chance to raise different ideas. Usually demo sessions like this are very useful if time is not too tight.

*Q: What will be the consequences if you let bias affect your decision?*

A: For domain Banking, the company can lose quite a lot in terms of assets because if for example there are two cases. One is that I transfer the missing money to the customer, it will affect the customer's property and the customer can sue me. The second case is that I transfer the balance to the customer, then I will suffer a loss and I cannot manage that amount of damage.

**Speaker 3 - QA Engineer at Tech Company C**

- *Background*

*Q: Could you tell me a bit about your experience as a software tester?*

A: Yes. And my background is in international business and management. So, I mean, I've been exposed to general best practices in terms of how companies operate, how things work. But my personal experience as a software tester, I'd say that comes in the last three years where I started working for tech companies before I was working for.

*Q: What challenges have you faced when testing software applications? Describe your team's responsibilities, including usual duties and assignments.*

A: My team is in charge of doing releases for software orientated tech products and services. So essentially what my team does is it reviews something that is in, as we call production, which means it's already rolled out to the user. The user has access to this software and we compare the results against new software, new patches, new code, and we compare that against what is already in production. So we basically analyse something old against something new, and we try and take into account the user journey and how much it impacts the user and what will their experience be. So we try to analyse from many different angles and that's our main responsibility, um, in terms of the challenges that we face when doing that, there's always many challenges and there's technical challenges such as, you know what, you know, fixing code and understanding code, understanding the technical part of it, especially when we don't have exposure, we don't have full visibility because it is a very big company. Mm hmm. So often we don't have the knowledge that, um, we're working on something that is created by another team, so we don't have the knowledge to tackle any problems that come up, etc., etc.. So yeah, I think having. Cross transferrable knowledge from other teams. That's one of the biggest problems we face. And because we just don't have the visibility. And then there's also the human element and such as disagreements inside the team and people having differing opinions. One person can think A is right or the best option for the user, whereas someone else can think B is the best option for the user. So I'd certainly say that that is one of the biggest challenges we face.

- *Testing organization*

*Q: Could you explain the test organization you use when testing software applications?*

A: And when we're doing the test organisation yeah. Generally I would describe this as the kind of process that we follow or the kind of framework that we use to analyse, you know, what is our release doing, Is it good or is it bad? So generally know we have to look at it, we have to make our own independent decision on something. And then if there is, if it's good, okay, no problem. We just take a box. If it's bad, then we have to kind of use our own independent thinking and objectivity to raise this issue to whether it be a software engineer or someone else. We have to basically escalate the issue. Um, so yeah, you first come in, you compare the results, you analyse the results, and then you, you escalate the results if they need to be escalated. And then of course you have to follow up. If you escalate something, then you have to follow up and you have to come back to it to see. And was this a worthwhile escalation? Should it have been escalated in the first place? Um, and if anything else needs to be done, so you follow up on that review process and that's, that's a very high level overview of how I work and the team works.

*Q: What processes do you follow in order to ensure the highest quality of software testing*?

A: We're talking about quality of software testing. So one thing we have is a quality lead or a quality analyst, and essentially they are in charge of making sure that the team's quality is good, making sure that we're delivering good quality for our customers. And um, essentially they have their own framework to help us achieve that quality. So many parts of our workflows are broken down into smaller parts, and then those smaller parts are um, described. So for example, if we're doing a release, as I said in the test organisation stage, we need to compare it to results so that um, that stage would be broken down even further. So it goes from comparing to results to did the and did the software tester compare these things, Did they look at all of these things. So that's, that's one way that our quality analyst and ensures quality. It's by breaking down each of our processes and steps and outlining what is a good way to analyse something and then comparing it to what we did. So essentially we established a framework of the best way to analyse our software and then that's what the quality analyst does. They try to ensure that we follow that process.

*Q: Do you believe this process is the most ideal?*

A: I think so. As long Yeah. As long as yeah. There have to be certain things in place to make sure that that happens, such as the appropriate communication from the quality analyst to the software tester, um, to make sure if what they're doing is correct, that they should continue to do that or if what they're doing is incorrect, then you need to have a conversation about what you should do and how you can test better. Um, so yeah, I think it does help.

- *Bug analysis*

*Q: What techniques do you use to identify, analyze, and document bugs during software testing?*

A: Yeah. Good question. Um, I think the techniques that I use are quite different to the techniques that someone else would use simply because we're all humans. Everyone has a different view of the world and everyone has a different way of working for me. Um, I try to use empathy because often we're using it things from we're looking at things from a user perspective. So I need to be and I do use our product every day. So from that sense, it's easy for me to step into someone else's shoes. But what I'm trying to do is put myself in the user's position. And so I'm identifying things from a user perspective. It is it better? Is it worse? Um, do I have greater visibility over? Um, sorry, I was about to say something I shouldn't, but yeah, the. Is it better or not. From a user perspective, using empathy. Okay, then if I'm analysing things, I need to try and analyse them in the most neutral way possible. So, you know, I'm from Ireland and it's possible that my own cultural bias may creep in.

*Q: What strategies do you use to ensure accurate bug analysis and reporting?*

A: Analytic and kind of judgement. So I need to try and remove that judgement and analyse things from a purely neutral perspective. So being very neutral in terms of localisation, in terms of cultural differences, I think that's quite important. And yeah, if I'm documenting bugs. Again, very much trying to put myself in the position of someone else. So if I'm documenting a bug, I'm documenting it for the purpose of someone else and it's going to be given to someone else so that they can take action on it. So I need to very much describe the issue in clear details, make sure it's understandable. So someone just like I have no idea about some of the code entries, this person may have no idea about some of our processes. So I will need to be very clear and thorough in documenting my findings in the book. And the best way you can do that is, you know, it's the cliché. Explain it to me like I'm five. It's, you know, something that people say and it's basically saying, ask me to explain it as if I'm five years old. Explain it as if I have no idea.

So I think the best way to do that is explain something like you're like the person you're expanding to is five years old, like they're a child. So explain it in the easiest and most simplest form. Don't overcomplicate things. And that's how I would document my book.

- *Barriers or negative aspects*

*Q: What challenges have you encountered when trying to mitigate human bias during software testing?*

A: Somewhat, but not too much. Um, yeah, Building. Building on what I said earlier. Often we have the technical barriers, and we often have the human barriers, which are kind of, you know, people's own biases or their their own frame of reference creeping into their work. And so, yeah, certainly those, I would say, are the biggest things. But, um, when trying to mitigate human bias, um, I can be doing two things. I can be trying to mitigate my own human bias or I can be trying to mitigate my employees or my, um, my colleagues human bias. So if I'm trying to mitigate my own human bias. Again, I'm just trying to think as neutrally as I can. But sometimes that's not always possible, cause sometimes you're tired, sometimes you're hungry, and sometimes lunches in 10 minutes. So you, you don't want to focus just as hard. Um, and then, for example, the hand you're not focusing, and then your bias can creep into your work because you're just not using the same rampart to stay as neutral as possible. Um, and that extends to when you're trying to mitigate the bias in your colleagues, because they can also be tired. They're also humans, they're, they can be hungry or whatever. But yeah, then you can also have, you know, kind of emotional blockers with, with colleagues, um, cultural blockers with colleagues. Um, so for example, if emotional blocker, I would describe that as their attitude perhaps. So um, I think some people are unwilling to change their frame of reference or to think

that other frame of reference does exist outside of their own. And that's, yeah, that's, I think that's just on some, that's dependent on someone's own ability to think neutrally to have an open mind and um, yeah, just be receptive to new ideas. So if you're not receptive to new ideas, then it's very difficult to mitigate your bias. So yeah, for me, it's difficult to change someone's emotions or to change someone's personality. So from that stance, I think it's very difficult to mitigate somebody else's human bias. Um, yeah, sorry. I've kind of, yeah, lost my train of thought, but yeah, definitely working with my own bias. Yeah, it can be emotional factors, working with someone else's bias, trying to mitigate that. It can be their their own emotional barriers or their own cultural barriers.

*Q: What strategies can be used to reduce human bias during software testing?*

A: Um, so we're talking about strategies to reduce human bias. Yeah. Then I think there's several strategies. Um, one of the best ones can be to add another opinion, to add some diversity. Um, because if two people are saying. One thing and you add a third opinion, then it may, because these two people are there agreeing. They're in there in a room looking at pieces of software and they notice a bug and they think, Oh, yes, this is a very big bug. But then you add someone else to the conversation. We're not three people in a room and someone else says, Oh, hey, did you look at this? And then the two people say, No, I had no idea. I never considered that. So I think adding an extra opinion or an extra voice can add the diversity that's needed to stimulate and basically a full perspective of the situation. So that can be one method to reduce human bias, adding another opinion.

*Q: But how do you think there's more and less?*

A: Good point. Um, I think it's up until a certain point. Um, so more and more voices can be too much. Um, so I would say this is. Yeah, it's not a linear skew. It can be. Yeah. If there's too much input, then there's too much noise and there will be no output. So I don't know personally, for me, I would like one or two extra opinions if I'm doing my work, if I'm getting too many opinions, I know that some of those will be good points, but I also know that there's going to be a lot of noise and irrelevant discussion which will occur, and that's going to probably detract from the total output and basically the quality of my work. So too many. There's an expression in English called too many cooks spoil the broth. And I think that applies here. Yeah, you're right. Too many people, it's.

*Q: Are there any potential barriers that can prevent software testers from eliminating human bias during software testing?*

A: Um, that's a good question. So I think it's certainly a potential barrier is if you don't have any exposure to other opinions. So if you're locked inside an echo chamber where you're only

saying the same thing again and again and again, then you're not going to have the open mind and the objectivity to think of, um, how to eliminate your own bias. So if you're not seeing new perspectives or new frames of references and your bias is probably going to stay where it is and also being too busy, I think if you're.

*Q: What best practices can software testers follow in order to reduce human bias in software testing?*

A: I'm just thinking. Yeah. So in terms of what best practices software testers can follow to reduce their own bias, and it's quite difficult because I'm not sure what are the best practices for reducing your own bias. So it's difficult to answer. Um, but in my own experience, I think the best practices is certainly remaining neutral in your thoughts, using empathy to just always put yourself in someone else's position, in somebody else's shoes. So if you're testing something for a user, you must always be trying to think from their perspective and have the empathy to relate with them and think of what their thoughts are going to be at this time. And also diversity. So empathy is the first one. Um, I think diversity is the second most important thing. Um, not too much diversity like we spoke about, but there should be extra opinions, second opinions because that can challenge your own bias. And so I think certainly empathy and diversity are the two best practices from my experience.

*Q: What will be the consequences if you let bias affect your decision?*

A: Yes. I think consequences are. Yeah. That's a tricky one. And I think consequences are a mirror. So yeah, the consequences are going to be whatever you see, whatever you, whatever you're looking at, it's going to reflect. So essentially the consequences are on. And if you're applying your own bias. For example, if I'm applying my own cultural bias and then there's things in other cultures which may be very, very bad. And for example, in my culture it's it's not problem. Words, a a phrase, a gesture. Things like this can be extremely offensive in another culture. So the consequences can be unknown to you. And that's why it's so important to probably remove bias because it can have such a great impact that you're not even aware of. So I think that's yeah, that the consequences are endless, basically

**Speaker 4 - Software Tester at Tech Company D**

- *Background*

*Q: Could you tell me a bit about your experience as a software tester?*

A: Okay, so my name is Divya. I came to Germany since 2021, after I've done my bachelor's and master's both in India. I have been masters of computer applications. I did my internship in India. Then I got my first job back in India, that is Project Coordinator. So in this job, I have

done almost all the tasks like developing, just being managing the project. And after coming here in Germany I studied German language while did freelancing for an Australian company. So that I did manual testing actually and when I was about to quit this job, then I jumped into automation and that company with me. Now I'm more in automation. I'm not doing manual anymore and our current product is mobile applications.

*Q: What challenges have you faced when testing software applications? Describe your team's responsibilities, including usual duties and assignments*.

A: Hm. I can say manual testing is quite a hard job. Automation is quite easy. Because it is getting with automation, those things, it's more fun. If you are interested in coding, then it's good to do automation because you have all been like everything. And every day you learn something new. Actually, I'm the only automation engineer in my company and I belong to testing team. So we have a team of five people and we don't use Scrum because I am doing only automation test and it's completely a new project. Oh, so we have a basic application. I'm doing automation of that application, and that is a stock business application. And we have a meeting every week and I write the test specification and test cases and then I execute it. And also for the continuous integration.

- *Testing organization*

*Q: Could you explain the test organization you use when testing software applications?*

A: It is very simple. Whenever I find a bug I reported it on TestRail. But as an automation test, I mostly report my bugs that I can go live for via email.

*Q: So that means from the beginning until the end of the test process, only you work on this one and nobody join you and give you any feedback?*

A: I just need to send it to my leader and then receive the feedback from him on every Monday.

*Q: Do you have any product owner? And in case you find a bug but you are not sure, what would you do?*

A: If if you have found any bugs, you defected it and you could not accept it you have to get the support because it should be regarded as reportable. So most important thing because it can be happen like now developers are not accepting the bug, but the bug is there and a future customer will report that bug and you can say yeah, you have already reported that bug but nobody listened to you so you should always report any bug you find

*Q: So it means that you could not do anything more?*

A: You cannot just disable or ignore it. You'll have to report it. But if nobody fixes and still release the versions, then it's okay to read the history.

*Q: What processes do you follow in order to ensure the highest quality of software testing?*

A: The automation, also manual testing is needed. So if something is under development, one manual testing should be done. Then it comes to automation. Automation cannot be happen without manual testing, otherwise there will be bugs and those all fit and you will automate the software with all the so many of those fingers must. So there should be always amendment as to.

- *Bug analysis*

*Q: What techniques do you use to identify, analyze, and document bugs during software testing?*
A: For manual testing, I think the process is like you get some requirements and what should be the output. And then you check what is actually done and what is expected. The same thing happens in automation. Also, I get that those cases and the output is there, or if it matches them or doesn't match, then there's a bug.
*Q: What strategies do you use to ensure accurate bug analysis and reporting and you can convince developers that it's a bug and they should fix it?*
A: That depends upon how you are talking to developer. If you are a to talk like and this is the bug, might be I can be wrong on my view. I have understood something wrong. There should be always a philosophy psychology between the tester and the developers. So you have to understand how to communicate
*Q: Oh, so this is more or less about communication, right?*
A: Exactly. Exactly!

- *Barriers or negative aspects*

*Q: What challenges have you encountered when trying to mitigate human bias during software testing? Such as if the requirement causes any difficulty for you to understand and define if it is a bug?*
A: Yeah, this happens. You cannot go to developers. You should always go to your managers because they can answer, they can ask you about the department's requirements. The requirement is always from the customer side. So go to the manager could ask customer and they will tell you.
*Q: What strategies can be used to reduce human bias during software testing?*
A: I work alone now. But when I was new in the company other time when I found some difficulties to a couple developers, so my team was very supportive. That's how I learned, actually.

**Speaker 5 - Junior QA Engineer at Tech Company E**

- *Background*

*Q: Could you tell me a bit about your experience as a software tester?*

A: Previously, I worked as a software tester. Now, I'm working as a QA junior at my current company. Full time just started in February. Before that, I worked for 2 years as a tester. Actually, my team was a scrum team but without... We didn't have that much capacity back then, because a scrum team is usually small. So my company wants to keep it as small as possible. So we have to be both testers and in QA, which means doing many things at the same time. Because my company is also a startup, so I do many types at the same time.

*Q: What challenges have you faced when testing software applications? Describe your team's responsibilities, including usual duties and assignments.*

A: For example, a normal model is that the tester will have time to design test cases. But what we do has to be released every 2 weeks so people can't focus on one thing. Because we still have to write automation tests and have to maintain framework to automate tests. So it is also a very big work and at the same time has to do manual test cases. And we have to Drive the test circle for the whole team's sprint in order for the whole team to join. So those jobs add up, we can't have enough time to design test cases that it's like a common development people do. And the challenge of Scrum is that normally developers create features and near the end of the project the tester will start testing right? Because I follow Scrum, so I have to test once every 2 weeks. And what does test once mean? That is, in Agile Development, every sprint I have to re-test all the features once to see if the new changes affect the existing ones. It's called regression testing. Whereas with Common Development, I will do final testing before giving the customer all the features. This simply means that you understand that I am not only testing new things, but I will also test my product to see if the new things have any effect. That is the biggest challenge of Agile Development so far. Actually, it is the problem of many projects and many teams. Not just each of his companies. The time of a sprint can't be much, but you have to test everything that your product has so far plus new ones. So we focus a lot on automation testing a lot, that's very important in Agile development. First, at the end of the sprint, we will have to plan for the team to run regression tests which means testing new features and features that we have just developed in that sprint. That means re-running all the tests that I have so far to make sure it has no effect, not broken anywhere. That is the biggest and most important task that I have to do. I have to prepare and how all the teams on the test day which is the last day of the sprint can test as much and as fast as possible. We have to try to find as many bugs as possible because we can't find all bugs, we can only find as many as we can. The normal model is that the tester is someone who has a lot of experience and specializes in that problem but in Agile teams they often have teams without testers. All roles in the team are done by a tester.

Why? Since time is running out, they use different perspective of each person finding bugs, which means the developer will test the case it's different from the tester testing the case it's different and the PO testing this one. Different roles, different people test. Test customers are also different. Because people have their own criteria to look at a function for people to test. So this model focuses on that to find bugs as quickly as possible. That is the first task that will have to be planned for the team so that everyone can join that day of testing. Everyone test and can align at the end of the day whether we can deploy or not.

We have to discuss whether these bugs are critical or not. Then I see if there is any blocker. Either I can fix it that day or I can't fix it. That's the decision that was made at the end of the day of testing. That's the process that I have to plan from the beginning to the end of the sprint. The second responsibility is to handle automated testing. The automation test it needs a separate framework, it needs a separate repository. Testers must also have the code to run automated tests for themselves. It's a bit like developers. Then it is also a very big task when I have to maintain a large framework and have to check if the automation tests work, if it is green or it is red, it means it fails or passes. A lot of times it fails not because the test case it failed but because the system I built it on has a problem, I will have to fix it. So that's another task besides this. The next task is to manage what kind of test cases? Where do I save? How do I manage test cases? The sprints, the new test cases that I put in, I have to add in my regression test. But I didn't add them all, because not all of them are important. I have to filter it out and I put it in there. And that is test case management as I think. And those are the three main missions. Besides, I have to prepare documentation for the team so that they can support the team.

Sometimes there are developers who don't like testing. Then I have to explain to them that in the AI Development it is like that. Since I don't have much time, I don't have much capacity. And that is the strength of Agile Team. Which I have to use. Those are many different roles. I can find as many bugs as I can. And the end goal is just that I want to make sure my product is released as best as possible. That is the common goal that everyone should think about.

- *Testing organization*

*Q: Could you explain the test organization you use when testing software applications?*

A: That was the old days when I used Jira to leave test cases. It's actually just like Excel and Word. Everyone is very manual. But when you have to save that, it's very manual. And not maintainable, because it is very difficult. The second is how you save, because each sprint repeats itself. It needs something to support it. But you can't save it all to any page. Very difficult. For each test case you have to describe it, so you need to include an image, describe

test text, and everything else. So it must have a dedicated one for that. We use the test management tool to do that. Jira has the correct test plan, test execution to use to manage that. But convincing people to switch to it is not as easy as I thought. There are views that will be different from everyone else's and everyone has to understand why they need that. For example a test plan. Every sprint I will create a test plan. Test plan is actually just a place where you put all your test cases and you say these are the test cases that we want this sprint we will test it. For example, you are testing, and I don't know, I tested your correct one. So how do you know who is testing and run it? The test management will help me know that. Yes, you are testing that, I will see that you are testing, I will not test anymore, I will test another. And save the team a lot of time. And save those results. And I am the one who will create the plans. I am the one who will create the execution if the team wants to.

I will follow, I prepare the team to follow and do those things. I also have to create a dashboard on Jira so I can keep track of it all. For example, the current progress is how everyone is testing and there are more or less, more or less, I will have to ask the team now to do it faster. Or is it ok that today everyone works earlier than usual so I can meet earlier. Example to manage the sprint testing.

*Q: How do you hold the testing day?*

A: I will have to train the team on what the process of that day will be like, what the turns are like, and when it's about to kick off, everything is ready for testing then I will announce everyone on Slack as Started testing today. Then everyone will automatically know how to open the boxes that we have prepared and choose the test case to test.

*Q: Did you hold any meetings before to prepare for the testing day?*

A: That is, on the dev side, they write unit tests. To implement the features. One had to test one's features when one already had it. The tests they have written, usually a tester should know what they already have so that I can avoid it, let me write other tests. Because in order to avoid me testing too much time and while this is already there. Well, now that my company has it, people see that my test case has many things that people have put in their code, then I really don't need it anymore. And that is our next topic is to sit back and see if what the Dev side already has, I will remove and add the necessary things on my side. That will probably need regular meetings to be able to talk about it. As for the fact that before the test, we didn't need a meeting. No need why? Because we're made of Jira. It is quite transparent so I can follow Dev and Dev can also Communicate with me very clearly through that ticket. That test day is regression test and test the new feature one last time. Actually, our old features have to be tested before. Must test right after it just merged. Every feature that was merged into one we had to

test immediately and one of those people had to be the PO. That is, in our team, the rule is that a feature, a task of a developer will be tested by at least two people. So there will have to be someone who is PO. That means one of them must be the PO. The other person might be a tester, or a developer, or a project manager, or whatever. That's how we let PO join our testing. And must test first, as soon as possible, to see if the requirements are met. Each tester only needs to complete those acceptance criteria. Your task is called done accepted when you complete all the criteria. So when a person, anyone on the team tests that, they can rely on this Acceptance Criteria for them to test like a PO. And when they find bugs, what they can ask the PO are these bugs, critical bugs? And the rest is up to the PO to decide.

*Q: What processes do you follow in order to ensure the highest quality of software testing?*

A: A feature must be tested by at least 2 people and 2 different people and actually in my team it is tested 3 times. Until the end of that testing day is the day when all the features are merged together and tested immediately to see if it has any errors so that the devs have time to fix it. It is not simply testing every single feature but it is testing all the features at the same time going back and forth before releasing.

So in the process that a feature is created, the other 2 people have already tested it, meaning the feature is tested for the first time and on testing day will be tested by the whole team again, ie three times testing 1 feature to We are as sure as possible that it is ok and this is a very normal thing when the first person to test it passes it is green and the second person sees it and fails. Because maybe I won't check that but someone else will check it.

- *Bug analysis*

*Q: What techniques do you use to identify, analyze, and document bugs during software testing?*

A: Like I said pass or fail, it's actually simply the test description there. They wrote it down and when I tested it, it wasn't the same as what they described, it was a fail. The thing is you describe a test case, you say "Ok, this when I click here, it should show me this". So now you follow that step and you see the result it doesn't look like in the description or it doesn't look like in the screenshot people left like the "expected result" there then you know for sure is it fail. And I need 2 testers because of that. But some people test it through speakers, ie test it quickly. Actually, I haven't met many people like that. But such a case happened. In addition, testers also need to follow experience and time and the team it works together will understand each other better. Everyone will gradually think very similar and follow the same flow. So everyone will automatically understand whether this is a pass or a fail. Because for example something that is too small, people just document it and create a new task "Please fix this small bug" and still let this test pass. Because it's that test case, people test the functionality as the

main thing, but for example, the design has a slightly wrong format, they won't beat the test case as fail, but people still leave it as the pass but they don't. will make a new task to fix this small format error. That's what one can do. That's what it depends on the context that I'm meeting for me to make a decision. And its team learns each sprint together when we have a meeting at the end of the sprint usually we will go through all the bugs that we found that day and start talking to each other then everyone will always understand how if in the team we will usually solve the problem. And while designing test cases, I will usually divide the test cases according to the severity of the test cases. For example, I write a test case about price. For me, what is related to price is blocker. It that fails means it is blocker. As for the rest of the tests, which are examples of design or typefaces that do not appear or are missing, for me, if the information is not too important, it is also a "minor bug". That's how I can decide. Because I determined it from the beginning, it was easy for me to decide in the end. That's also the strategy I came up with for the team. I am the one to estimate the severity of the test case. The person who decides whether it is important or not is the PO. I'm just helping POs simplify their work. But they're supposed to do it. I'm not the one making the decision. I am a support person. I'm the one who defines it according to what I think it is.

*Q: What strategies do you use to ensure accurate bug analysis and reporting?*

A: Usually the QA person and the PO and Project Manager, they will be the ones who will decide which bugs are critical. And the developers, actually the whole team. All teams will also decide. Actually, in my team, it's PO's right now. But if QA in the future does well, we will have the right to say "no". You say this doesn't matter, to me it's important, it's a blocker. But you say no. Then I don't agree. I always have to handle the minor bugs and document them so that P0 will decide with the team when to fix the minor bugs because it's been noted in the backlogs. For example, the next sprint I don't have many new features, I will fix the little guys or the interns can fix the small issues. That's how it works on its own. In addition, it is still based on the acceptance criteria. It's like when you talk to a customer and you work with a customer and you know how much the customer wants this feature. And if this feature works but the design is a bit bad but this spring I can't fix it, the PO has to make a decision whether to release it or not. In fact, they wonder. Sometimes they don't even know if they should or not. That's why they need the support of the team because of that. Everyone will decide to release or not. Usually the dev will join and fix if it feels too necessary. Business Analyst is the coordinator of both parties. They know what customers usually do. And what should I focus on and what is important. They can discuss it with the team or ask the customer directly.

- *Barriers or negative aspects*

*Q: What challenges have you encountered when trying to mitigate human bias during software testing?*

A: If the PO reads the test cases by himself, reads the results, and decides by himself, but the team does not know the status of the project and only knows when the PO asks, it is very dangerous.

*Q: What strategies can be used to reduce human bias during software testing?*

A: The strategy is that all teams will have to attend the meeting. Everyone sits there and transparent about the problem. It is very clearly stated in that it is agreed with all that it is a blocker or not, fix or not, release or not. And if it is fixed, who will fix it? And after fixing, who checks it and approves it will be approved. So that's how we make it transparent as much as possible. The group chat is very important. I advise never to chat with each other privately about an issue, but to chat in a group so that everyone can read the information. I used to work with two different teams in the past. A team is always transparent, really like it. But the other team doesn't like that and prefers to talk privately. But I don't like that, I just made it transparent and people told me not to post like that. But I said "No". I find these transparent to save time as well.

*Q: What best practices can software testers follow in order to reduce human bias in software testing*?

Because if you should learn more about testing techniques then it will help you to see more clearly. And don't understand what the Dev is doing anymore. People have different small components, usually they will test those components already. Then I will be the last person to test whether the components work well together. That's something they can't test when they're Dev, they implement the feature. That's what I should focus on is the flow. You go from the step of adding the items to the cart, then you go until you choose the payment method. Each of these steps is a step, it's a small package that Devs have to create. But the thing about testing how all that works together, it's that I should focus on that process, I test as a customer too. What customers like to do. Plus you will know which test cases will be needed and which test cases it matches, which means that developers have already tested small components, small packages. I think yes and should listen to the PO. People know customers well, so they stick to customer criteria more closely. In fact, PO people test more about customers than themselves. Because I use test techniques to test if the features of the dev developers are good or not.

**Speaker 6 - Software QA Tester at Tech Company F**

- *Background*

*Q: Could you tell me a bit about your experience as a software tester?*

A: I graduated from university in 2014, then returned to Vietnam to work every year from 2015 to 2017 and then I continued to study further in Canada until 2019 and then I continued to work again. Working for the position of Software Tester until now, I think I have 5 years of experience in this field. Because during the time that I studied for higher education, I worked as a Freelancer, also a Software Tester. I do both Automation and Manual. Sometimes there are projects where they require 70% Manual and 30% Automation. There are projects they claim 50/50. In general, not every testing team does both, there are testers who only specialize in manual and there are testers who only specialize in automation, but because I know how to do both, I was assigned both.

*Q: What challenges have you faced when testing software applications? Describe your team's responsibilities, including usual duties and assignments.*

A: There are also challenges in the work, but in my opinion, it is that although there is no specific for any position, but for Tester/QA is probably the relationship between developer and QA. Because sometimes I also have to do something so that I can convince the deverloper to fix bugs. Because many times in their mindset they don't think it's bugs. It's hard to describe now because before my company was a waterfall, but later I started to adapt and gradually moved to Agile, I also started applying Agile with Scrum but it wasn't 100%. In the company they are divided into many different teams and each sub-team has a different function and has its own number of developers and testers. Our team will do it from A to Z, end to end, that is to say, from the step is requirement, then analyze the requirement, discuss with each other, then do planning, then start to code, prepare test plan, test case, Test products of all kinds and fix bugs and then release the product.

- *Testing organization*

*Q: Could you explain the test organization you use when testing software applications?*

A: I will start with BA. They will provide the requirement and I have a backlog containing all the requirements. The whole team will discuss with each other what tickets are needed for which functions. I will divide the points for each ticket and the overall price of the ticket's complexity is about how much and the time plan is for the developers to see if they will start. from when and when will they deliver the ticket to QA. During that time, QA will start planning for tests, making test plans, preparing test cases. After developers assign tickets to themselves by deploying to test environment, QA will start executing test cases to test tickets and log bugs.

Then there will be a time to fix the bug and I test verification so I can close the ticket. Next will come another time after everything is done, then I start to do sanity testing for the whole thing.

In that sanity test there is also an integration test when I integrate the function it into what is a main product because currently the company is in the form of production. Product is already available and working so the new functions are only added so our team has to do integration first for all and then sanity test then if everything is ok it will go to the release step.

*Q: What processes do you follow in order to ensure the highest quality of software testing?*

This company's process has just been re-teamed. Before that, the company had a different process. Usually, after a period of 2 to 3 years, their company will improve the process. Right now, I think it's pretty good. The only thing that is a bit difficult is that this process easily causes overload for the QA team because most of the developers are late, so they will accumulate tickets in one go. While asking my team if developers can split the number of assignments. For example, in 1 month, the first 2 weeks they will deliver the function A, then I test all the function A and then the last week they will deliver the function B, then I will have time to test the function B but Usually they deliver late, they deliver both functions at the same time, which will easily cause overload.

- *Bug analysis*

*Q: What techniques do you use to identify, analyze, and document bugs during software testing?*

A: Well it depends most of the time it will be black box testing but later on, they recommend that I combine both with White Box Testing and also with gray box testing. That means they now require to go into the request and have to read the code to know which parts have changed. And in that code what parts will be affected and let me know before I test. There are also different techniques in black box testing, like equivalence partitioning or Boundary Value Analysis or cause and effect and so on. Those are the same applications that I use. Black box testers are testers usually they will be the ones who have to act as the end user of the product and they will not write information about the product's algorithms in advance.

*Q: What strategies do you use to ensure accurate bug analysis and reporting?*

A: For example, in that function, I will know what the function is, usually I will follow the main flow that goes from A to Z. That means I will do everything for that function, go. From A to Z. Then I will apply techniques like I divide for example, I divide the equivalence area of one of the functions so that I can define test cases for that function. In the function's requirement it will have the so-called expected results of the function and I have to rely on the expected results for me to do. Expected results are available from the requirement of the BA party. When I make a test case for that, I have to determine that my test case must cover that expected result. Because only when we cover the expected result can we guarantee that the function works, it works, and

then we will expand to other cases. In addition to those expected results, I have to imagine myself as a user to see if they have any other flows.

- *Barriers or negative aspects*

*Q: What challenges have you encountered when trying to mitigate human bias during software testing?*

A: Based on my work experience to create cases beyond the expected results. Usually people with a lot of experience can think of many cases. But it can be said frankly that it will never happen that I cover 100% of all cases for that function. Even my manager straight up said that there will never be a thing that you will cover everything in that function. So I just try to do my best as much as I can create as many cases, I just do it. The rest is that I will never fail because I will never know what the user will do.

*Q: What strategies can be used to reduce human bias during software testing?*

A: I can avoid creating test cases that are not close to reality by asking many other people and companies that use an application to manage test cases like Jira, if I go to Jira, I research on it, I will Reviewing the history, old bugs, I will learn quite a lot of experience so that I know what users think and I can predict what users will do. I have to study hard all the time. I have to ask a lot because in this industry I have to ask, not ask, but just silently doing it myself, I will never improve.

*Q: What best practices can software testers follow in order to reduce human bias in software testing?*

A: Everything works will be based on the requirement, I just need to rely on the requirement I work on and I give my evidence in a logical way based on the requirement, then no one will be able to refute me. . For example, sometimes when I test, there will be cases where I catch cases that easily lead to controversial cases, but usually, I will give them to BA and related people so that dicuss and BA are the ones to come up with. final decision. If the BA agrees this is not a bug, then I just ignore it, but if the BA sees it as a bug, the BA will say no, this is a bug, the developers will have to fix it.

Normally when I find bugs, I will specify in which version these bugs will appear. Depending on the priority and severity of bugs, it will decide whether to fix it immediately or not. If the priority is low, the developers will just leave it there, they fix the ones with higher priority. After a while, then they come back, they will start they fix smaller bugs, now they will look at that version and then they will have to get the correct code of that version for them to fix. This will save a lot of time

**Speaker 7 - QA Engineer Lead at E-Commerce Company G**

- *Background*

*Q: Could you tell me a bit about your experience as a software tester?*

A: Mm hmm. So, yeah. So I graduate, you know, with a bachelor's degree in computer science. Pretty much like, you know, coding. After my graduation, it was that. Yeah, I start out with a software developer first, or, like, maybe about two years. And then, um, you know, so with software development, you know, you pretty much like now in long term piece of code at that point, which is the system, the overall system as much as Tester. So that's when I started moved to, to become a tester. Um, so back then, so my first job was telecom industry. Yeah I learned about like how to set up the switch and stuff like that and make calls to it, you know, like simple because it was like a call, like a user, stuff like that or, you know, doing the local call or, you know, a long distance call in natural call. So yeah, so that's, that's my first exposure to testing. And then after that, you know, the company would want to start doing the automation. Automation with those, you know, is don't make a call. You pick up the, you know, that phone and, and try to get dial tone and stuff like that. They have a simulation, you know, so go to automation then and we start using those and you know like uses simulator that you know, great script, you know with that test group for it was that you know, using C++ and back then so yeah so it was a little more a little more exposed to the automation was so I mean people ask me what is automated. I would say that's, um, that's a picture of a manual tester and a developer that what do you do? You go hire a hybrid in between. So you, you got knowledge of, you know, testing, you know, manual testing, but you do coding as well. So it's a good combo. I mean it's not heavily coding at is like a software development. Do some coding. You know what you know after in the US for about two years and then after that I changed it to like health care, um, stock trading company, the country station and uh, and the last one I've been with, well, even now is that Pizza Hut for the last four years, so pretty much, yes. Yeah. So I get to like, e commerce. No. Yeah, yeah. Then again it's still is this manual testing and then automation again. So you know it's fun and interesting

*Q: Describe your team's responsibilities, including usual duties and assignments.*

A: Agile team we got like six developers six seven because that one like been moving in and out a little bit. So that's got like seven developers and three. Okay. Um, you know with that like so much like we got tickets coming, and you know, developer work on them and then you know, what is available, what can we as a key ways to go in and look at those ticket and you know, start come up with like call test and arrow and test case you know for manual. So you know what we have done, we meet up with the developers and other stakeholder. You know what that

like all the team will ever want to call it forward to view out those test case. So whoever want to give input or come see, you know that we by a lot of people off your people, but mainly the people on a team. So they go, I mean, we come in and reviewed the test case that been read by us and you know, to look at it, well, you know, we got enough courage, you know, for that take or for the whole project itself, you know, and to get the input and you know, we can they update the test case or, you know, like even, you know, like some, you know, bring up discussion, you know, Well, I don't think, you know, this will we don't need to test it because that was cool. You know so those have been know we discussed at that meeting. The developer is done with a ticket so they give that question to us, you know that either put that color to like sort of testing environment, you know, we call it like staging or you know, was that something else? And then so we test the ticket manually for you, you know, make sure you pass. And then so one the ticket pass everything. Then we start. Okay, start, you know, look at the automation. You know, they're like, okay, let's say like we got five test, you know, pick and we can automate all five. That's great. If not, we pick, you know, whichever has the most coverage of the, the feature we are to make them. Yeah. Convert them to script.

*Q: It looks like your team has to do every activity of the part testing?*

A: Yeah. Before we release, we have to push it to a note of production that, you know, we call it up, which is simple. Like the production. Exactly like production.

*Q: Is it a testing environment?*

A: Yes. So we run our test case, you know. Yeah, it could be mostly. Yeah, like one day. So what, that like we run all the manual testing, anything that's not covered by automation, we run them. And then was that and run everything, every script into automation on that day, on, you know, on the environment as well. So yes, so that as the let's say you know, we release by not the reason we wanted to release by next week, right? So we're like, okay, we have to run our test, right. Write Friday later. So we call Yeah, I like that person you mentioned was like this one day that we dedicate to, like full force testing. Yeah.

*Q: What challenges have you faced when testing software applications?*

A: Tested issue coming up, you know, was that like. Yeah. And this data can be like was that this call or that. Yeah. I mean a lot of issue you know was that always, you know all the time so normally was that we you know issue it can be either you know some data we're missing so you know I always you know it's always a collaboration, right? So there are people, you know, like if I run through data issues, I always like, you know, how people on the data side that I can, like, reach out to and, you know, ask questions, stuff like that. And yeah, I mean, or setup, you know, was that like, hey, sometimes, you know, you know something new feature coming in

that requires special setup like on the menu side stuff like that for example. So yeah, there'll be like a menu that, you know, I go reach out, Hey, I'm trying to set up this menu item right now and I'm stuck. Can you get me some help? Yeah. So there's always issue. And you know what? It's just sometimes it is a part of Find out who you can get help from. So, you know, if I can or if I don't know who to get help from. I always bring up in the So for Agile you have I sprint meeting every day so you know any roadblock or anything that issue or you know any concerns you have always ask in that meeting.

*Q: But do you have any challenge with communication to developers?*

A: Not really. I mean, you know, with that, I mean, you know what that like to all the company I work with, I mean, like I, I don't, I never have problem with developer because, I mean like, you know some people have that separation between the developer and QA, not me. I mean, like, you know, a little I get along. The thing is that yeah, yeah, they develop a way to, to, to test the, you know, the code stuff like that. But others I really need developer to push those feature out. Well yeah. So you know, I mean like one time so you know, we have a like a new openness, stuff like that. So they asked me to help out with interview so, well, not just me but like other so it can they can be like, oh you know what that like, and you know what I always argue with developer. I go, Excuse me, I mean, I agree with you. I mean like there's a, there's a line between developer, but, but here's the thing so you upfront you know that they, they need to test right but yeah we need them also in the both way. So I mean, I mean I don't like the idea of coming here to attract developer you know what they're like. Right. Okay. Just me.

- *Testing organization*

*Q: Could you explain the test organization you use when testing software applications?*

A: So I think you have to do that so that they call sprint virtual. So yeah, so pretty much I had every spring we got for a meeting, you know, and then uh, you know, sprint planning was that and then, you know, they start up and then sprint retro and demo for. So for the sprint, that's all, you know, everything we learn and we can improve on, we discuss it there, right. And then, you know, if you ask about like where I mean, where I put my test case on test and stuff like that, normally we the way that the my company right now, or most of you know, every company I've been with too is that you know we operate on recalled zero. Oh yes normally I put my testing into that or it or you know not completely which is like a wiki. So I put it there and you know, if I put it on the wiki then I put a link to that wiki inside individual tickets. So that way, you know, when you look at the ticket, you can like go to that wiki page or look at a test plan there, stuff like that.

*Q: What processes do you follow in order to ensure the highest quality of software testing?*

A: I mean, was that, you know, that different way of testing? You know, it wasn't like, you know, that for me, um, I don't use my job well, I mean, exploit, exploit testing, you know, was I only have time, but most of time, like, my role is both like, um, like integration testing. So, you know, integrate, you know, like the feature from developer with, like, other components, stuff like that.

- *Bug analysis*

*Q: What techniques do you use to identify, analyze, and document bugs during software testing?*

A: okay, let's say, for example, write the order and you know, use an order pizza. And you know what? I like this piece as opposed to cost by $10, right? So the expected result would be like $10. No, no less, no more and something like that. Right. So that something like that, we get that you know, we get that for requirements like, okay, um, like if this order, if this piece are being ordered, you know, by the customer is to see like $10 check out. Okay. Right. I mean like you know with that, like it could be much more of a black box testing more because I think the white box is more done on the developer side because they, they, they see but we don't see the code either like, you know, the proper, you know, impression. I think it's pushy. Come over. We don't care what you do with the code. We just want, you know, like make sure, okay this is they coming in use the order the pizza successfully. It'll check out stuff like that that that's all we can. So yeah, it's more of a black context.

*Q: What strategies do you use to ensure accurate bug analysis and reporting?*

A: Mm hmm. Um, so normally when, you know, like, um, is a new feature or existing, we, we try to limit coverage that we can or, you know, we would ask the developer that, hey, this is not the plan for my test that, you know, I plan to cover, you know, like the, the menu part and then the check out part. You think that's enough? You know, what's that? Because I mean, like developer know which piece of code they touch more. So, you know, like my developer case, they will come back out. I want you to test, like, the menu with that and then check out. But also it'll make and make sure you can able to check out check out the big card more or what we call localization, you know, when you want to use it. So like at search store, you know, those errands so that the developer give me more input on you know, where to where to touch my testing.

*Q: So your strategies is about to communicate to developers?*

A: Yeah. Okay. I mean they, the one like they, they know which you know which error code and you know they need to make the change into, you know they would make a change to like what you call a card area or like a with that payment area. So like hey touch you know these like focus on these are more for me, you know they, they would tell us you know like.

- *Barriers or negative aspects*

*Q: What challenges have you encountered when trying to mitigate human bias during software testing?*

A: Customer will very likely do that on the app. Yeah. So I mean like could you put yourself into customer's shoes. You know, like they did the usual trying to order a pizza for e.g. or you know, or modify an item, stuff like that. So, you know, you have to lay out stuff like that. Yeah. We just dealt with to like, okay, I see this, you know, there's a button below very high chance that that the user going to like do something and it did click that but you know so yeah.

*Q: Do you have any challenge while you try to imagine all scenarios?*

A: Uh, not really. I mean, you know, what's that like? I don't see, you know, with any system I don't see. Yeah. I mean like sometimes you buy a hard initially, you know what I like, but I mean, you know, if you keep doing it, you know, keep doing it more. This again, you know, with that become ill with it. And, you know, after a while, like, oh, it's just another way of doing it. Right. So yeah, so like people go like, oh, you know, why is it so hard? And like, I don't get why is it so hard? Okay, I mean, like, if you keep do it, you know, like if you don't do it yet, you would say you find hard, but, you know, like I just trying to explain people that like, okay, if you want to learn about it. Okay. I mean, I understand like people just new to company, right? Like, oh, I joined this company last month for ex. Okay, I understand that you do. But here's the thing. How hard is it? You know, quickly, just that once a customer go to the website and, you know, try to place an order, that's simple as is, you know, I, I mean, I have I agree with you that I have seen like people are like I think is more an excuse because, I mean, it is a complex system then I agree yet as this is hard but you know how simple my product is.

*Q: What strategies can be used to reduce human bias during software testing?*

A: So for challenge I always say, you know, just keep doing it. And the more you do it, the, the more you can build with it does not mean you cannot find it. Challenge in and I mean like not difficulty anymore. You know what in terms of one challenging. It's going to take time to, to, to learn that. Understandable you know, nothing wrong with that.

*Q: What will be the consequences if you let bias affect your decision?*

A: Not all companies are that flexible and not every organization that flexible. So, um, I have a piece that I have the like, you know what I like. So I won't totally because we do deployment stuff like that. So I will close it. When they called was that site after it. I know you heard that side of reliability with that so they, they handle like all the incident or production disasters, stuff like that. Right. So with them, I think they are a little harsh on their employee because I have seen like they fired an employee. Was that like because, you know, one mistake that they make.

Oh, I forgot to like, erase the key or I've forgotten dead and forgot that, you know. So if you ask me that question, I think it come down to it. Manager So what they should do so would play with me my group, um, I have seen you know, case where, you know, we would make mistake and forgive and stuff like that but you know I think it depends. It depend on the manager and how sensitive the products are to customers such as finance and so on.

**Speaker 8 – Junior QA Engineer at Tech Company H**

- *Background*

*Q: Could you tell me a bit about your experience as a software tester?*

A: Right now, I'm junior quality assurance engineer at a tech company in Lisbon, Portugal. Um, I had three years working as a technical support for a hardware company before. Therefore, I have some knowledge about network. Uh, hardware and software, um, and operations system. After that, I decide to move to another new field. And luckily, I got an offer, uh, and work as a junior quality assurance engineer until now, for two years. Um, basically, uh, I don't have any, um, background like, uh, developers before because in my company, the process doesn't require a lot of knowledge of coding. That's why it is, is a little bit easier for me. And I also have knowledge about network and operation operations system. Um, this is my experience as a software tester.

*Q: Could you please describe your team's responsibilities, including usual duties and assignments?*

Um. Uh. Okay, so right now I am working as a junior quality assurance and in my team and my team is QA team. We have the same role. We don't have any developer; therefore we just review the changing of products before release. Developers change or update the features, we review it and we approve. In case we see any issues we have to inform to the engineer, I mean, developer by escalating. If the issue is critical, we will block the bug. Sometimes we have to release from Monday to Thursday. Uh, sometimes we do not have to review releases. It depends on the schedule of each release. Besides, we have some small projects to improve our internal tool for our quality assurance because we don't usually use third party tools and we build our internal tools which are based on some platforms for tracking bugs and feature requests during product development.

*Q: Do you not have to create test plan or test case?*

A: No no. Everything is very simple. We receive the evaluating report and we review and analyze it to see if there is any critical issue to block the release. I think that's why we have

release every day because if we have to create test plan and cases, we won't have enough time to do it.

*Q: What challenges have you faced when testing software applications?*

A: Oh uh, the challenges in software testing, I think it's mostly about our product. It is a little bit special because we have a huge amount of releases and changes each day. Therefore we have a lot of work and we have to review every day, nearly every day, sometimes to really tricky to us to decide if we should log this bug or block the release. And we don't have any close contact to developers because only our team work on quality assurance and while the engineer I'm the developer department is located in another country with another time zone. So that's why it's really hard to communicate to them . Also, some issues could be just minor issues to them but based on our human review it is not to us. So again this is the challenges - workload and communication that in our organization.

- *Testing organization*

*Q: Could you explain the test organization you use when testing software applications?*

A: Um. Our team belongs to the Waterfall methodology. We do not use Agile framework. We have more than 10 quality assurance engineers with 2 team operation leads and 1 quality lead. We have rotation schedule which will assign the member to each release per week. One release has 2 assignees. One will be the primary contact while another one is the backup in case the primary one needs to discuss the bugs before they raise the issue to the whole team. So with the whole process, we just manage exactly the part of quality. We don't really meeting with, uh, engineer or talk with engineer. Uh, when they have any plan for changing any feature they will do it by themselves and, and every day during our working time the review will come and we review analyze and approve. Uh, and every week we also have stand up meeting where we update the status of our releases last week.

*Q: What processes do you follow in order to ensure the highest quality of software testing?*

A: Um. Um. So for the process, actually it's really simple. The process really simple. As I told before the developers create the environment where they put the change of the feature. And then when their parts of are done, it is will be ready for us to reveal. We will just see if there's any information that it is strange and compare to the current version to see if these changes maybe lost important information or functions. Uh, and then we will take note this one to the report and we will take a look to see if there are a lot of examples for that issues issue and is looks like will affect our user journey then we decide to block the release. Of course not all issues will block our release. Just some of them. Based on how critical that issue. Mostly we run the demo to make sure if it still reproduces and then escalate it to the developers. That's

why we, we are, we're careful when we decide to block to the release or not because blocking release is important to the developers and us. So we will talk with our backup first and we will talk with the whole team who have more experience on that issue before. Also, quality lead also help us  check up and consult us when we need help in case if the whole team could not help us because she has more knowledge about the product and she's also managed the quality of our performance.

- *Bug analysis*

*Q: What techniques do you use to identify, analyze, and document bugs during software testing?*
A: Um. All techniques are used to identify, analyze and document are not same like other companies. Oh, actually, right now, because our team, uh, just focus on one part - quality. We don't really have any communications with developer and we have to be our internal tool. So first of all, we use excel to perform our review and document the issue. It could be called manual testing. Of course, we still have some automation tools which do help us to gather the information like, like changes that we should review but all tools will transfer the information to excel sheet. So and then we, we analyze this on Excel and when we need to discuss we use google group chat to communicate with other teammates. All excel sheets will be stored in google drive where we could look back if we need to evaluate the performance. It could say we use Google work space. Um, so actually the techniques, we don't really use black box or white box or gray box testing and we just, you know, our knowledge, like we know exactly how the features should trigger and should not trigger. And when we compare, we try to put ourselves to end user 'shoes to see if that triggering of feature meets their expectation, um, sometimes there are some rules for some specific releases and we just need to focus on the requirement of the release. So it will be easier for us because we will have is we have a specific requirement to follow.
*Q: wow it sounds really simple, right?*
A: it is and sometimes it is not. Because we don't really have a strict process comparing to other companies.
*Q:What strategies do you use to ensure accurate bug analysis and reporting?*
A: I think mostly it should be followed the guideline. Because during the guideline, it shows step by step, and it also shows some troubleshooting. For e.g. when the system is down or we could not be able to extract the report. And it sometimes shows how the features should trigger which is very important to us. If the documents from the company do not have the information we need, we could check the reports in the past but it does not mean to show everything. So

sometimes you have to talk with your senior or your quality lead to be sure that what you decide is correct. Uh, that is my strategy.

- *Barriers or negative aspects*

*Q: What challenges have you encountered when trying to mitigate human bias during software testing?*

A: The challenge is that we need to think about our user's expectation. And so it really happens because there are million people using our product every day and it is hard to understand what they want. Maybe what we think they need is not what they need and what we think they don't need actually is what they need. So we have to compare and consider most of time. To follow the minor or the the major isuse, potentially there's something that we can do is common sense. And this could not be documented because the above-mentioned documents are about operation of system. Sometimes information is discussed but no one take note it. And every time we face that issue, we might debug differently. And each of us also has different perspectives which cause some arguments during discussion.

*Q: What strategies can be used to reduce human bias during software testing?*

A: A: And it's like I told you before. Following provided guidelines and documentation, consulting senior team members, learning from past mistakes, considering end user needs, and not overcomplicating the testing approach. While there are no major consequences if releases go out with issues, preventing problems and monitoring product quality are still priorities. In the end striking a balance between major and minor issues is important for success in this role.

*Q: What will be the consequences if you let bias affect your decision?*

A: First of all, that, um, actually without our team the release is still automatically released after a certain time. If we review and find a critical bug, we could block the release but without our work, the release is still out. And of course the next release there is still these problem but if the customer reports, it would not be a big deal. So actually in no matter if we approve we skip, it would not cause any big problem. But of course, with our work, we will prevent issues and monitor the reality of our product. The second thing is that if we block the release because of an unimportant issue, it will cause a mess to everyone, including our department and the developer department, because when you block to release we need to explain why and so on. So that's it from our company.

**Speaker 9 - Senior Software Tester at Energy Solutions Company I**

- *Background*

*Q: Could you tell me a bit about your experience as a software tester?*

A: Currently, I have been working since 2015, then by 2023, I will have 8 years of experience. He have worked for 3 companies. At the first company, I worked for 5 years and then 2 years. And now, I am working for an American company with the position of Senior Software Tester. The main job is manual testing. Testing will have manual and automation testing but he mainly does manual work.

*Q: Could you please describe your team's responsibilities, including usual duties and assignments*

A: Currently, it is UA testing which means a circle, the developer phase is phase 1, where I receive and learn about business requirements from customers. I'll break down into small pieces and convert them to tickets for me to do. Phase 2 is that I will develop. Then developers will read those requirements and code them into functions or products. After they are done, move on to the third phase of testing environment. Normally, companies only have 3 phases, that is, after testing is completed, they will be given to customers. However, in my company, there will be one more phase, UA Testing. That is, when the testers in phase 3 ensure that they have passed the acceptance criteria given by the customer, we will put the product into another environment, which is UA environemnt. Then now he will verify the UA key criteria. After he has made sure everything is up to his expectations, he will release it and give it to the customer for final testing. This position will require the most understanding of the customer's criteria and then I could talk to customers after releasing.

*Q: And what challenges have you faced when testing software applications?*

A: Hmm the first and most important challenge is when testers are assigned a project and they do not lack requirements or documents about that project or system. For example, if a system has been running for 10 years without documents, how can a new tester understand how the system works? Testers can learn by themselves but it will be very time consuming and will miss a lot of information. So writing requirements and documents for a project is the most important. In addition, testers have time pressure. Whenever I find a bug or an issue near the release, I have to wait for the developer to fix it while the deadline is coming to an end. At that time, testers have to test quickly to make sure they will release on time.

- *Testing organization*

*Q: Could you explain the test organization you use when testing software applications?*

A: As I said, testing will have 2 environments: test enviroment and UA enviroment. Then UA environment will have 2 teams in Vietnam and USA. The team in Vietnam is that I will handle only test environment. After finishing, the Vietnam team has 2 people, me and 1 other colleague, will coordinate with the US team to test UA. The first round of UA testing in Vietnam

and a second round of UA testing in the US. But American experts will know more about the business and system of the product than test techniques.

*Q: Oh based on what you explained, is your team using Agile methodology?*

A: Yes, exactly! We are using Scrum Team. Every sprint takes 4 weeks which is longer than normal process. I know many companies only have 2 weeks per sprint.

*Q: What processes do you follow in order to ensure the highest quality of software testing?*

A: I don't use test cases or test plans of the testing environment. I have to rewrite the test cases in your way. Even though they have given test cases, the UA Testing team does not follow, but will use their own mindset to test. Of course also based on the business requirement and perform those test cases yourself. First, we also attended intro meetings, sprint planning, daily scrum. During sprint planning sessions, his team will receive customer requests. Based on acceptance criteria, we will write test cases ourselves. And run those test cases ourselves in UA environment. If everything goes well, I will capture those test evidence and give it to the customer.

- *Bug analysis*

*Q: What techniques do you use to identify, analyze, and document bugs during software testing?*

A: I also use the same techniques as in the testing environment. For example is . Equivalence Partitioning, Boundary Value Analysis or Decision Table Testing. I use it to write test cases. Our team only uses Jira to store test case and so on.

*Q: Are these black box testing?*

A: Yes it is! Our team only use black box testing for manual testing.

*Q: What strategies do you use to ensure accurate bug analysis and reporting?*

My strategy is to use traceability matrix. Because it allows you to trace back from a test case to the requirement it is covering, and vice versa. In addition, I use more personal experience that I have accumulated to know where there will be errors or where it is easy to be missing.

- *Barriers or negative aspects*

*Q: What challenges have you encountered when trying to mitigate human bias during software testing?*

A: When I have to act as a customer to use the product. But for me, it's not too difficult after working for years.

*Q: What strategies can be used to reduce human bias during software testing?*

Just follow the techniques I mentioned above, I will reduce my subjectivity when writing test cases. I can also do my own research to learn more about testing. Learn more about technology.

For example API testing and so on. In general, I research and learn by myself. In addition, I also need to be careful and meticulous when doing this work

*Q: How about learning from our teammates or seniors?*

A: It is also a great way to do it.

*Q: What will be the consequences if you let bias affect your decision?*

A: If I give it to the customer but the customer doesn't detect it, it's okay. If it's just minor bugs, it's ok because I'm human. Can't be 100% perfect. However, if there is an error from the acceptant criteria, they will generate a report and evaluate the team's performance. A few times it's okay but too many times making errors related to main functions they will escalate and the team will lose credibility and customers.