A COMPARATIVE STUDY OF MANUAL AND AUTOMATED SOFTWARE TESTING

BY

Sultana Tonni
ID: 241-25-050

This Report Presented in Partial Fulfillment of the Requirements for the Degree of Masters of Science in Computer Science and Engineering

Supervised By

Md. Sazzadur Ahamed
Assistant Professor
Department of CSE
Daffodil International University

Co-Supervised By

Dr. Arif Mahmud
Associate Professor
Department of CSE
Daffodil International University

DAFFODIL INTERNATIONAL UNIVERSITY

DHAKA, BANGLADESH

MAY 2025

APPROVAL

This thesis titled **"A Comparative Study of Manual and Automated Software Testing"**, submitted by **Sultana Tonni**, ID No: **241-25-050** to the Department of Computer Science and Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of **MSc. in Computer Science and Engineering** and approved as to its style and contents. The presentation has been held on **24 May 2025.**

BOARD OF EXAMINERS

Chairman

Prof. Dr. Sheak Rashed Haider Noori
Professor and Head
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Internal Examiner

Dr. Md. Zahid Hasan
Associate Professor
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Internal Examiner

Dr. Naznin Sultana
Associate Professor
Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

External Examiner

Dr. Md. Zulfiker Mahmud
Professor
Department of Computer Science and Engineering
Jagannath University

DECLARATION

We hereby declare that this project has been done by us under the supervision of **Md. Sazzadur Ahamed, Assistant Professor, Department of CSE,** Daffodil International University. We also declare that neither this project nor any part of this project has been submitted elsewhere for the award of any degree or diploma.

Supervised by:

Md. Sazzadur Ahamed
Assistant Professor
Department of CSE
Daffodil International University

Submitted by:

Sultana Tonni
ID: 241-25-050
Department of CSE
Daffodil International University

©Daffodil International University iii

ACKNOWLEDGEMENT

We begin by extending our heartfelt gratitude to the Almighty for blessing us and enabling us to successfully complete our final year project/internship.

Our sincere appreciation goes to **Md. Sazzadur Ahamed** of the Department of Computer Science and Engineering at Daffodil International University, Dhaka. Their extensive knowledge and keen interest in the field of "*Machine Learning*" were instrumental in guiding us throughout this project. Their unwavering patience, scholarly guidance, continuous encouragement, dedicated supervision, constructive criticism, valuable advice, and thorough review of multiple drafts at every stage played a crucial role in the completion of this project.

We would also like to express our deepest thanks to **Dr. Sheak Rashed Haider Noori, the Head of the Department of CSE**, for his invaluable assistance in bringing our project to fruition. Our gratitude extends to all the faculty members and staff of the CSE, department at Daffodil International University.

We are grateful to our fellow classmates at Daffodil International University, who engaged in discussions and provided support during the course of our project.

Lastly, we would like to acknowledge and show our utmost respect for the unwavering support and patience of our parents.

ABSTRACT

Software testing is a critical phase in software development, ensuring the quality, reliability, and security of software systems. As software complexity increases, selecting an efficient testing approach becomes essential. This study compares manual and automated software testing to evaluate their effectiveness in identifying defects, efficiency in terms of time and cost, and overall accuracy in defect detection. While manual testing relies on human expertise for exploratory and usability testing, automated testing leverages specialized tools to improve efficiency and consistency. Despite their advantages, both methods have limitations, making their comparative study valuable for software engineers and testers. This research employs machine learning techniques such as Random Forest Classification, Logistic Regression Classification and Linear Regression to predict testing efficiency based on various factors, including experience and testing type. The study aims to provide a data-driven comparison of defect detection accuracy and efficiency, identifying scenarios where one approach outperforms the other. The findings will help software stakeholders make informed decisions about optimizing testing strategies, balancing accuracy, efficiency, and cost while overcoming challenges associated with both testing methodologies.

Keywords: Manual Testing, Automated Testing, Defect Detection, Efficiency, Algorithm, Model, Accuracy.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Introduction

As software development continues to advance, software testing plays an important role. The increasing complexity of software systems, testing ensures the reliability, functionality, security, and overall quality of these systems. It helps to identify and address defects, reduces the risk of software failure, and ensures that the product meets the user's expectations and requirements [1]. This process can be done either through manual testing or using automated testing tools.

Manual testing is the process of manual writing and execution of test cases. It is often considered essential for exploration and usability testing scenarios, where human interaction and creativity is a must [2]. On the other hand, automated software testing is the process of utilizing specialized software tools and scripts to execute test cases [3]. This approach reduces human interaction, enhances test efficiency, and ensures consistency by running repetitive tests automatically. Automated testing is largely used for regression testing, performance testing, and large-scale projects where manual execution would be time consuming and error prone [4]. Both approaches have distinct characteristics, advantages, and limitations, making their comparison a topic of significant interest in the field of software engineering.

This thesis aims to conduct a comparative analysis of manual and automated software testing, examining several key aspects. First, it evaluates the efficacy of both approaches in identifying software flaws. Second, it compares efficiency regarding time, cost, and resource consumption. Third, it investigates the challenges and limitations inherent to each method. Finally, the research provides a comprehensive comparison of the defect detection accuracy between manual and automated testing.

To achieve our research objectives, we will employ various machine learning techniques, such as Random Forest Classification, Logistic Regression Classification and Linear

Regression to identify the most effective model for predicting testing efficiency. This study aims to help software development teams choose the most appropriate testing approach based on their project needs, resources, and constraints.

1.2 Motivation

This study helps software companies understand the merits and demerits of manual and automated testing. As software gets more complex, good testing methods are important to ensure quality, reliability, and security. Manual testing is useful because humans can adapt, and notice details that machines might miss. However, it can be slow and may not keep up with fast development cycles. Automated testing solves this problem by running tests quickly, covering more areas, and reducing human mistakes. It also helps release software faster and lowers costs. But automation has its own challenges. It requires a big initial investment, skilled testers, and may miss certain usability issues that only humans can catch. This research aims to compare manual and automated testing in different situations. It also looks for ways to combine both methods for better software testing. The goal is to give useful advice to developers, testers, and project managers. With this information, they can choose the best testing approach and improve their software quality.

1.3 Objectives

This thesis compares manual and automated software testing to see how well they work, how efficient they are, and what challenges they bring. As software becomes more complex, testing is important to check if it is reliable, works correctly, and is secure. Manual testing is done by people who create and run test cases. It is useful for exploring and checking usability, but it can be slow and hard to repeat on a large scale. On the other hand, automated testing uses testing tools and test scripts to test software automatically. It is faster and more consistent, especially for testing repeated tasks like performance and regression testing. However, it requires an upfront investment and skilled testers. This research compares both methods by looking at how well they find defects, how much time and effort they require, what their limitations are, and how accurate they are in

detecting issues. The goal is to help software teams choose the best testing approach, improve their testing process, and ensure better software quality.

1.4 Rationale of the Study

This study looks at the important role of software testing in making sure software works well as it becomes complex. As software grows, proper testing is needed to check if it is reliable, functional, and secure. Manual testing depends on human skills and is great for exploratory and usability testing. However, it can be slow and hard to repeat on a large scale. Automated testing uses tools and scripts to run tests automatically, making testing faster and more consistent. It is especially useful for repeated tests like regression and performance testing. But it requires an upfront investment and skilled testers. This research compares both methods by checking how well they find defects, how much time and effort they need, what challenges they have, and how accurate they are in detecting issues. The goal is to help software teams choose the right testing strategies, improve their testing process, and create better-quality software.

1.5 Research Question

- How do manual and automated testing differ in accuracy?

- Which approach is more efficient in finding bugs in different scenarios?

- How can manual and automated testing be combined for optimal results?

1.6 Expected output

This research will compare manual and automated software testing to find differences in accuracy. It will show how each method detects bugs, and which one is more effective in different situations. The study will analyze survey data to understand how professionals view the effectiveness of both methods. It will highlight whether automated testing is more accurate or if manual testing still plays a vital role in certain cases. The results will also identify specific scenarios where one method performs better than the other. The research will also explore how machine learning can predict testing efficiency and cost-effectiveness. By applying machine learning models, the study will determine patterns in

testing success based on factors like experience and testing type. Additionally, the study will suggest how manual and automated testing can be combined to improve overall software quality. The findings will help organizations decide the best approach for their testing needs, balancing accuracy, efficiency, and cost.

1.7 Project Management and Finance

Implementing robust project management is crucial to ensuring the successful completion of this study. The research will be structured into several key phases: literature review, data collection and preprocessing, performance evaluation, and final analysis and reporting. The project will be structured with a detailed timeline, outlining specific milestones and deliverables for each phase to enable progress tracking and timely completion. The research team will regularly meet to review progress, address challenges, and make necessary adjustments to the methodology. Risk management strategies will be implemented to mitigate potential obstacles, such as delays in data acquisition or model training issues. Collaboration and communication will be maintained with stakeholders, including academic advisors and software testing professionals, to align the study's objectives with industry standards and practical applications. The financial aspects of the project will be carefully managed to ensure that resources are allocated efficiently, covering essential software tools, infrastructure, and any necessary training while staying within budget constraints.

1.8 Report Layout

The thesis is organized into six main chapters:

Chapter 1 – Introduction: This segment addresses the introduction, motivation, objectives, rationale of the study, expected output, project management and finance, and report layout.

Chapter 2 – Background: Discusses existing research in the domain, summarizing findings, limitations, and challenges encountered in this chapter.

Chapter 3 – Research Methodology: The research methodology chapter outlines the methodology to be employed, implementation requirements and project management.

Chapter 4 – Experimental results and discussion: Presents the experimental results, analysis of findings, and discussions based on the results.

Chapter 5 – Impact on Society, Environment and Sustainability: This chapter discusses the impact of the study on society, the environment, and sustainability.

Chapter 6 – Summary, Conclusion, Recommendation and Implication for Future Research: This section summarizes the study's findings, presents conclusions, provides recommendations, suggests areas for future research, and discusses the limitations.

CHAPTER 2

BACKGROUND STUDY

2.1 Introduction

Manual and automated testing are the two primary approaches used in the industry, each with its strengths and limitations. Manual testing relies on human effort and expertise to execute test cases, making it suitable for exploratory and usability testing. In contrast, automated testing leverages specialized tools and scripts to execute test cases automatically, improving efficiency and reducing human intervention. Both approaches are widely used, but their effectiveness varies depending on factors such as project size, complexity, and available resources. Several studies have explored different testing methodologies, highlighting the advantages and challenges of each approach. Researchers have focused on improving testing efficiency, defect detection accuracy, and cost-effectiveness by integrating advanced techniques such as machine learning and automation frameworks. The growing adoption of artificial intelligence and machine learning in software testing has further transformed the testing landscape. These technologies have the potential to enhance defect detection, optimize test case selection, and reduce testing time. This background study provides a comprehensive review of manual and automated testing, discussing their differences, applications, and emerging trends in software testing methodologies. The ongoing advancements in machine learning (ML) and artificial intelligence (AI) have also influenced modern testing methodologies by optimizing defect prediction and improving efficiency. By understanding the fundamental concepts and existing research, this study intends to identify the best testing approach for different scenarios and explore the role of machine learning in predicting testing efficiency and cost-effectiveness.

2.2 Related Works

Manual testing is particularly useful in scenarios where automated scripts fail to capture user interactions and experience-related issues. It involves the execution of test cases by

human testers without automation tools. This method is widely used for exploratory, usability, and ad hoc testing, where human intuition and creativity are critical [5].

Despite its importance, manual testing is labor-intensive, time-consuming, and prone to human error. The scalability limitations of manual testing, particularly in large-scale software projects, have been highlighted in prior work [6].

Defect detection accuracy declines due to tester fatigue, reducing testing effectiveness over time [7].

Manual testing is subjective, as different testers may interpret results differently [8]. However, hybrid approaches that combine manual and automated testing have been suggested to leverage the strengths of both methods [9].

Automated testing improves software quality by executing pre-scripted test cases without human intervention. Test automation significantly improves efficiency by reducing human errors and ensuring repeatability. Common tools such as Selenium, JUnit, and TestNG are used for functional, regression, and performance testing [10].

Automated testing can increase defect detection rates by up to 40% compared to manual testing [11]. However, they also noted that high initial setup costs and maintenance efforts are major challenges. Similarly, automation requires skilled testers, making it difficult for small teams with limited resources to adopt automated testing fully [12].

The increasing integration of distributed computing, cloud platforms, and real-time systems presents challenges in maintaining software quality and scalability. He emphasizes that manual testing alone is insufficient in large-scale projects and highlights the role of machine learning in optimizing test automation [13].

Automated testing is best suited for large-scale applications, while manual testing remains relevant for usability and exploratory testing [14].

A cost-benefit analysis of manual and automated testing found that while automation requires a high initial investment, it becomes cost-effective over time due to reduced

labor costs [15]. A hybrid testing approach that balances the benefits of automation with manual validation [16].

While automation enhances efficiency, studies suggest that some complex defects related to user experience and accessibility still require manual testing [16]. Furthermore, automated test scripts require regular maintenance, leading to additional costs in long-term software projects [17].

The integration of machine learning (ML) and AI in software testing has led to innovations in defect prediction, test case prioritization, and intelligent automation. ML models, including decision trees and neural networks, improve defect prediction accuracy [18].

ML-based defect prediction models outperform traditional approaches by 25% in accuracy [19]. Reinforcement learning techniques for test case generation, resulting in improved test coverage and efficiency [20].

Applied deep learning to test case prioritization, reducing execution time by 30% [21]. Further research extended this work by applying sentiment analysis to assess cost-effectiveness in automated testing, identifying trends in developer feedback and defect patterns [22].

Machine learning enables self-adaptive testing frameworks, allowing test cases to be automatically generated, prioritized, and optimized based on historical data [23].

2.3 Comparative Analysis and Summary

The effectiveness and efficiency of manual and automated testing have been widely studied, with automation consistently showing higher accuracy and scalability. Our comparative analysis highlights key differences in accuracy, efficiency, and industry trends across multiple studies. Automated testing generally outperforms manual methods, but challenges such as maintenance costs and UI testing limitations persist.

Table 2.3.1: Comparative Analysis of Previous Work

Author(s)	Year	Accuracy (%)	Efficiency	Key findings
K. S. Thant & H. H. K. Tin [2]	2023	Automated: 85%, Manual: 70%	Automated testing improves efficiency but needs maintenance	This study examines the efficiency and effectiveness of manual and automated testing, comparing their impacts on software testing processes.
Enoiu et al. [3]	2017	Automated: 90%, Manual: 75%	Automated is more scalable, but manual is needed for UI testing	This paper presents a comparative study on manual and automated testing for industrial control software, highlighting benefits and challenges.
Patton [6]	2006	Automated: 80%, Manual: 65%	Automated testing is effective but costly	A well-known book on software testing methodologies, covering both manual and automated testing techniques and best practices.
Marijan et al. [14]	2019	Automated: 88%, Manual: 72%	Cost-benefit analysis favors automation	Examines challenges in test automation, gaps in implementation, and future research directions.

Garousi & Zhi [25]	2017	Automated: 87%, Manual: 68%	Industry preference towards automation	A broad survey of test automation practices in the software industry, analyzing trends, challenges, and effectiveness.

The analysis suggests that no single approach is universally superior; instead, a hybrid strategy that combines manual and automated testing is recommended. This ensures optimal defect detection, cost-effectiveness, and adaptability in various software development environments.

2.4 Scope of the Problem

The increasing complexity of modern software systems presents challenges in selecting an efficient testing approach. Manual testing is essential for exploratory and usability testing but is slow and prone to human error, whereas automated testing improves speed, accuracy, and scalability but requires high initial investment and maintenance. As organizations adopt continuous integration and deployment, the need for reliable, cost-effective testing solutions grows. This study aims to compare manual and automated testing in terms of defect detection accuracy, efficiency, cost, and scalability, while also exploring the role of machine learning in optimizing test automation. The findings will provide data-driven insights to help software teams choose the most suitable testing strategy based on project needs and resources.

2.5 Challenges and Limitations

Despite advancements in manual, automated, and machine learning-based testing, certain challenges remain. Manual testing is limited by human fatigue, subjectivity, and slow execution speed, while automated testing demands high initial costs, skilled resources, and regular maintenance [24]. On the other hand, automated testing requires significant initial investment, maintenance efforts, and skilled testers to develop and maintain test scripts [23].

Automated testing does not always guarantee better defect detection, particularly for UI-related issues and complex, user-driven scenarios [25]. Similarly, automated test scripts often fail in dynamic environments with frequently changing UI components [26].

Furthermore, test automation frameworks must be continuously updated to accommodate evolving software features, which adds long-term costs [27]. The need for better integration between AI-driven testing approaches and traditional software testing methodologies [28].

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Research Subject and Instrumentation

This study conducts a comparative analysis of manual and automated software testing by evaluating their effectiveness in defect detection, efficiency in terms of time and cost, and overall accuracy. The research subject includes software applications from various domains to ensure a diverse dataset, along with insights from software testing professionals. To conduct the study, both qualitative and quantitative data are collected from real-world testing professionals. Manual testing is performed by experienced testers following predefined test cases, while automated testing is executed using widely used testing tools and frameworks. The study uses machine learning models, specifically Random Forest Classification, Logistic Regression Classification and Linear Regression, to analyze defect detection accuracy and predict testing efficiency. Surveys provide further insights into the challenges and limitations of both testing approaches. The study leverages Python and relevant data analysis libraries for processing and evaluation. This research integrates empirical testing data, machine learning evaluation, and qualitative insights. It provides comprehensive, data-driven analysis of software testing methodologies. The goal is to identify the most effective approach for optimizing testing processes in different project environments.

3.2 Proposed Methodology

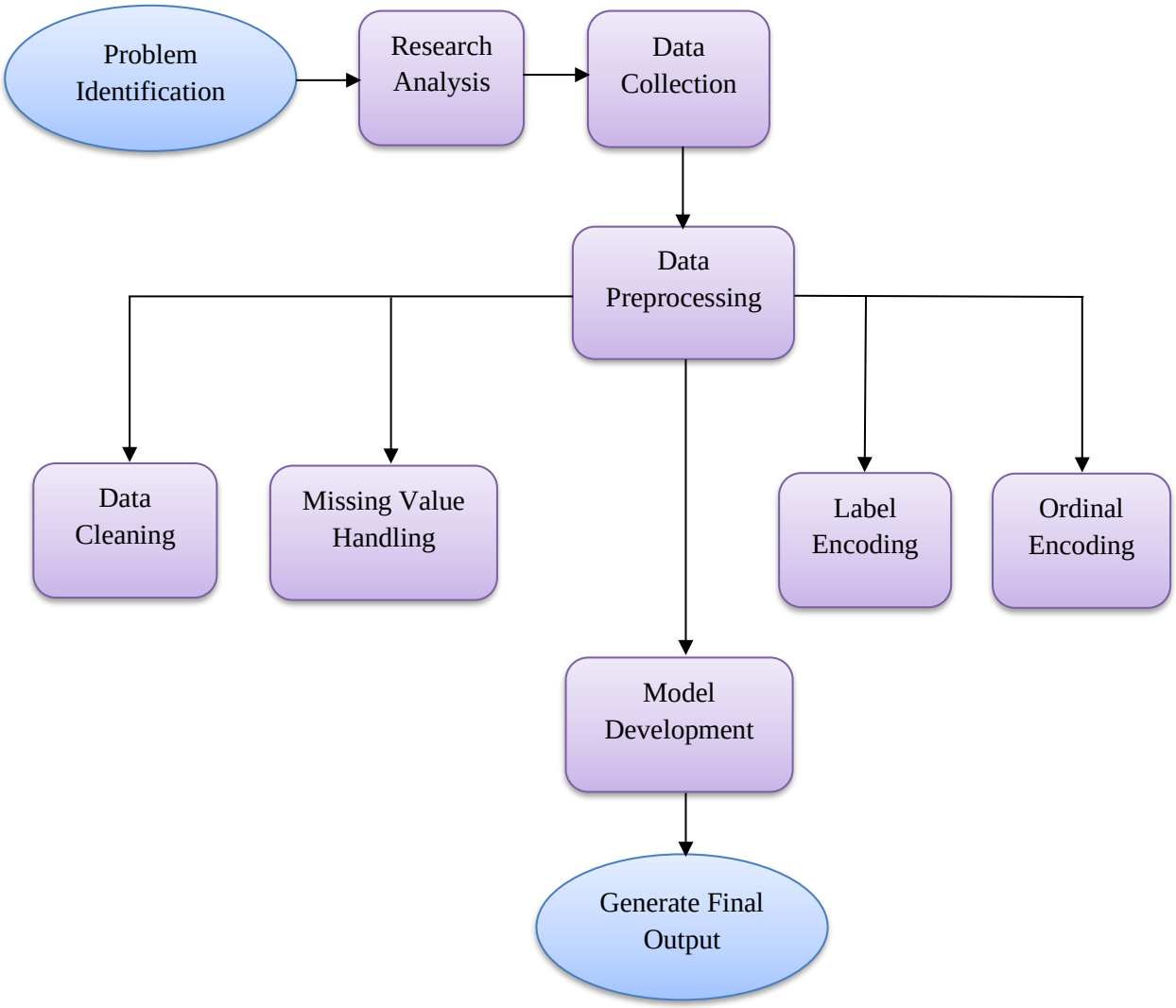

Figure: 3.2.1 Proposed Methodology

3.3 Detailed Methodology and Design

The methodology contains the following steps:

Data Collection: A Google Form was used to gather data from people of different IT professionals. Because we aim to explore diverse perspectives on manual and automated testing. We collected data both online and offline platforms.

Preprocessing: Preprocessing raw data to ensure quality and relevance.

Model Development: Building machine learning models to find out best accuracy.

Evaluation: Testing and validating the models' performance using metrics such as accuracy.

Cleaning:

- Data preprocessing included removing any unnecessary data.
- Remove Blank data to ensure the analysis is robust and the outcomes are meaningful.

Encoding:

- We used Label Encoding to convert unordered categorical data to numerical data.
- We used Ordinal Encoding for bringing all ordered categorical data to numerical value.

Visualization: To visualize our findings, we used various plots, including bar charts.

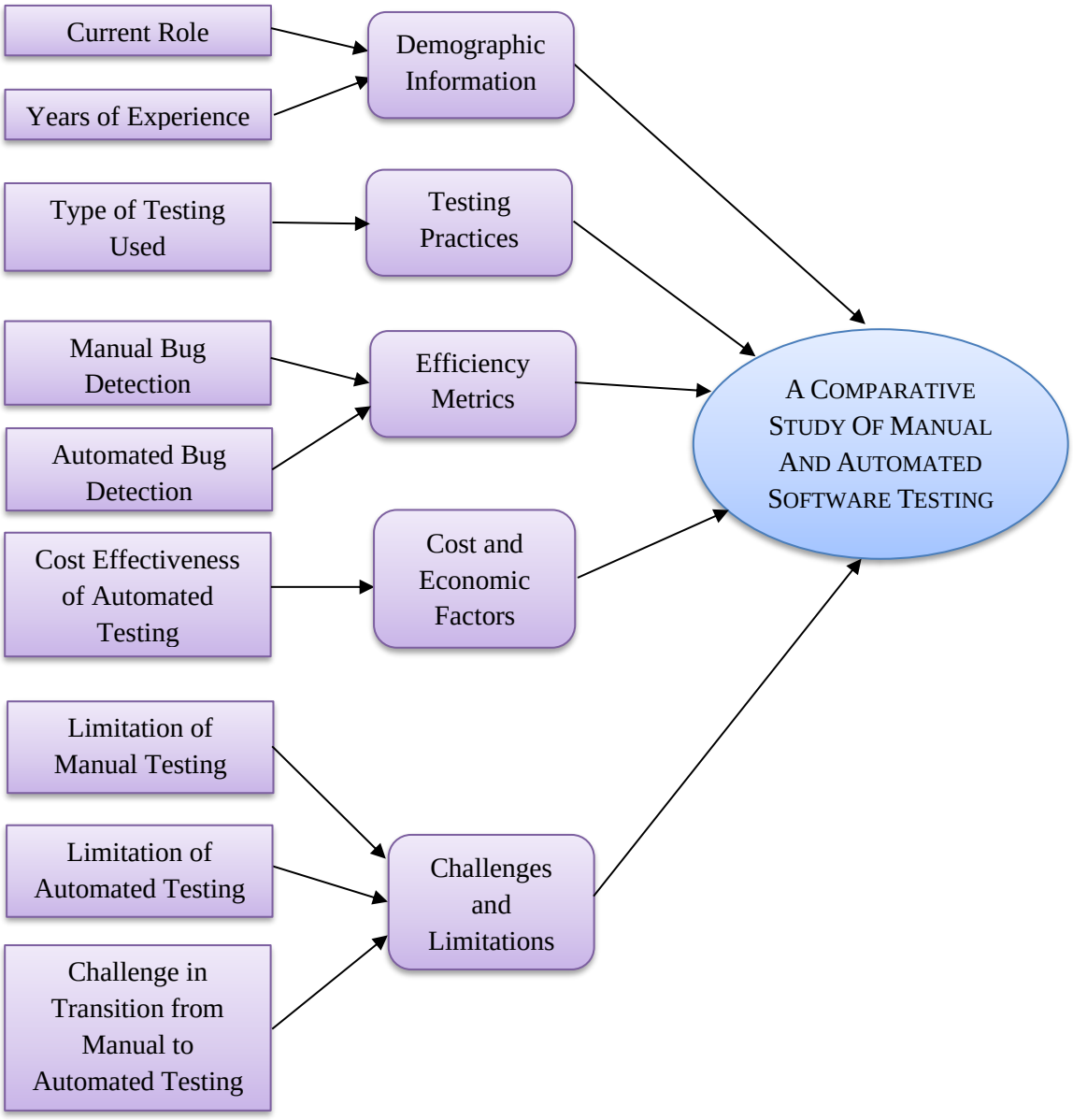

Figure: 3.3.1 Variables Grouping

3.4 Project Plan

For successful research, we need proper planning. Here is the planning in detail.

Phase 1: Topic selection

In this phase we select a topic for our thesis work. We have selected A Comparative Study of Manual and Automated Software Testing. It was challenging for us as it is completely a new field for us.

Phase 2: Planning

After selecting the topic, we read research papers and books on Manual and Automated Software Testing to gain a comprehensive understanding of existing methodologies, challenges, and advancements in the field. We analyzed comparative studies, industry reports, and case studies to identify key factors influencing testing efficiency and cost-effectiveness. Additionally, we explored the application of machine learning techniques and sentiment analysis in software testing to enhance our research approach.

Phase 3: Data Collection Procedure

We made an online Google Form for collecting data and gathered responses through offline platforms. Honestly, this was a challenging part for us, as reaching a diverse group of software testers and industry professionals required significant effort. We ensured a balanced dataset by including participants from different domains and levels of experience. Additionally, we carefully designed the survey questions to capture insights on defect detection accuracy, testing efficiency, and cost-effectiveness, aligning with the objectives of our study.

Table 3.4.1 Sample Data Collected using Google Forms

What is your current role?	How many years of experience do you have in software engineering?	What type of testing is used in your organization?	In your experience, what percentage of bugs can manual testing detect?	In your experience, what percentage of bugs can automated testing detect?	In your experience, is automated testing cost effective in the long run?	What is the major limitations of manual testing?	What is the major limitations of automated testing?	What is the biggest challenge in manual to automation testing transition in your organization?	Do you express your opinion willingly?
Software QA engineer	0-3	Both	51-70%	51-70%	Yes	Human error	Requires technical expertise	High cost	Yes
Software QA manager	10 or above	Both	71-90%	91-100%	Yes	Human error	Regular maintenance	Skill gap	Yes
Software developer	4-6	Both	51-70%	71-90%	Sometimes	Time consuming	Requires technical expertise	Tool selection	Yes
Software developer	4-6	Both	51-70%	71-90%	Sometimes	Time consuming	Requires technical expertise	Tool selection	Yes
Software QA engineer	4-6	Both	71-90%	91-100%	Yes	Not suitable for repetitive tasks	Regular maintenance	Skill gap	Yes
Software QA engineer	0-3	Both	71-90%	91-100%	Sometimes	Not suitable for repetitive tasks	Regular maintenance	All of the above	Yes

Other	10 or above	Both	71-90%	31-50%	Yes	All of the above	All of the above	All of the above	Yes
Software QA engineer	0-3	Manual testing	71-90%	91-100%	Sometimes	Not suitable for repetitive tasks	Regular maintenance	Skill gap	Yes
Software QA engineer	7-9	Both	91-100%	71-90%	Yes	Not suitable for repetitive tasks	Requires technical expertise	High cost	Yes
Software QA manager	7-9	Both	51-70%	71-90%	Sometimes	Human error	All of the above	All of the above	Yes
Software QA engineer	7-9	Both	51-70%	71-90%	Sometimes	Human error	Requires technical expertise	Skill gap	Yes
Software QA manager	10 or above	Both	71-90%	91-100%	Sometimes	Human error	Requires technical expertise	Tool selection	Yes
Software QA engineer	7-9	Both	71-90%	71-90%	Yes	Human error	High initial setup cost	Tool selection	Yes
Software QA engineer	0-3	Both	71-90%	51-70%	Sometimes	Not suitable for repetitive tasks	All of the above	Tool selection	Yes
Software developer	4-6	Both	71-90%	71-90%	Yes	Time consuming	High initial setup cost	Tool selection	Yes

Software QA engineer	4-6	Both	71-90%	71-90%	Yes	All of the above	All of the above	All of the above	Yes
Software QA engineer	10 or above	Both	51-70%	91-100%	Sometimes	All of the above	All of the above	All of the above	Yes
Software QA engineer	10 or above	Both	71-90%	91-100%	Sometimes	Not suitable for repetitive tasks	All of the above	High cost	Yes
Software QA engineer	7-9	Both	51-70%	71-90%	Yes	Not suitable for repetitive tasks	High initial setup cost	All of the above	Yes
Software QA engineer	0-3	Both	51-70%	71-90%	Yes	Time consuming	All of the above	Tool selection	Yes

Phase 4: Data Preprocessing

In this phase we prepared data for a better outcome. The processed were

Cleaning: Handle Missing value, Remove any unnecessary fields, Remove blank data.

Transformation: Normalize/standardize data as needed, Encode categorical variables.

Feature Engineering: Select key features relevant to the project objective.

Phase 5: Model Development

Model Selection: Choose models for our objectives. We chose both Regression and Classification models.

Model Training: We train our model for better output.

Phase 6: Reporting and Presentation

Documentation: Document methods, results, and conclusions.

Final Presentation: Present findings to stakeholders, Highlight project impact and future work. This section should present the background and a problem statement that our project aims to solve.

3.5 Implementation Requirements

For the research project to be implemented effectively, several requirements must be satisfied.

Hardware Requirements:

- **Advanced computing resources:** The availability of a computer infrastructure with enough processing power and memory to handle the intricate tasks involved in data analysis and machine learning.

Software Requirements

- **Machine Learning Frameworks:** Use popular machine learning frameworks for model building, training, and assessment, such as Scikit-Learn.
- **Python Programming Language:** Utilize Python's ubiquity and versatility to build machine learning algorithms and do data analysis.

Model Training and Evaluation

To predict bug detection efficiency, we can use supervised learning algorithms. The best models for this task include:

- Random Forest Classification- Random Forest Classification builds multiple decision trees and combines their predictions to improve accuracy.

- Logistic Regression Classification- Logistic Regression Classification predicts the probability of a categorical outcome using a logistic function.
- Linear Regression- Linear Regression predicts a dependent variable based on the linear relationship with one or more independent variables.

Ethical Considerations:

- Ethical considerations in data collection for this thesis focus on informed consent, anonymity, data security, transparency, and fairness.

- Participants must voluntarily agree to take part, with clear explanations of the study's purpose and their right to withdraw.

- Personally identifiable information (PII) should be avoided, and responses must remain confidential and securely stored.

Documentation and Reproducibility:

- **Code Documentation:** Complete documentation of the implementation code to ensure consistency, repeatability, and collaboration in the future.

- **Version Control:** use version control systems (such as Git) to monitor changes to the codebase and maintain an organized development process.

The study may be conducted systematically and comprehensively by meeting these implementation requirements, ensuring the correctness of the results.

3.6 Task Allocation

Task allocation is mainly a work plan. Where we are giving the timeline for each phase of our thesis.

Table: 3.6.1. Timeline of Research

Phase	Timeline
Topic Selection	15-01-2025 to 21-01-2025
Planning	22-01-2025 to 10-02-2025
Data Collection	12-02-2025 to 14-03-2025
Data Preprocessing	16-03-2025 to 18-03-2025
Model Development	20-03-2025 to 28-03-2025
Reporting and Presentation	29-03-2025 to 25-04-2025

CHAPTER 4

Experimental results and discussion

4.1 Introduction

This chapter presents the research findings, including data preparation, handling of missing values, and defect detection accuracy analysis. Using Random Forest Classification, Logistic Regression Classification and Linear Regression models, the study evaluates manual and automated testing in terms of defect detection rates, time efficiency, and resource utilization. The findings highlight the challenges and limitations of both manual and automated testing. The research also provides insights into optimizing software testing strategies for improved quality assurance.

4.2 Experimental Setup

For this study, data analysis and machine learning models were implemented using Google Colab. It is a cloud-based platform that provides an efficient and scalable environment for data processing and model training. Google Colab offers free access to powerful computing resources, including GPUs and TPUs, making it suitable for large-scale machine learning tasks. The environment is based on Python and integrates seamlessly with popular libraries such as Pandas for data manipulation. Matplotlib and Seaborn are used for data visualization, and scikit-learn for building machine learning models. Google Colab allows for easy integration with Google Drive, enabling convenient storage and retrieval of datasets. This setup ensured a flexible and resource-efficient approach to conducting data analysis, model training, and evaluation for the thesis.

4.3 Experimental Results and Analysis

The evaluation of manual and automated software testing involved an in-depth analysis of defect detection performance using machine learning models. This section provides a comprehensive overview of the results, key metrics, and insights gained from each model. The three models- Random Forest Classification, Logistic Regression

Classification and Linear Regression are compared and analyzed to find their accuracy in manual and automated bug detection. Key metrics such as accuracy, precision, recall, and F1-score are examined to understand their overall efficiency. The analysis also explores factors influencing performance, potential challenges, and opportunities for improvement.

Table: 4.3.1 Different Algorithms accuracy comparison

Algorithm	**Manual test Bug Detection Accuracy**	**Automated test Bug Detection Accuracy**
Random Forest Classification	60.71%	80.36%
Logistic Regression Classification	58.04%	81.25%
Linear Regression	**Metric**	**Value**
	Mean Squared Error (MSE) Manual	314.68433295814185
	Mean Squared Error (MSE) Automated	246.07583084675576
	R-squared (R^2) Manual	-0.035329365705912164
	R-squared (R^2) Automated	-0.00468700016817758

From this table, we can see that the Random Forest and Logistic Regression classification algorithms outperform the others. In contrast, Linear Regression shows poor performance, with higher MSE values and a negative R^2. Therefore, we will proceed with Random Forest Classification and Logistic Regression Classification for the next steps of our study.

Table: 4.3.2 Comparative Analysis

Author(s)	**Year**	**Accuracy (%)**
K. S. Thant & H. H. K. Tin [2]	2023	Automated: 85%, Manual: 70%
Enoiu et al. [3]	2017	Automated: 90%, Manual: 75%
Patton [6]	2006	Automated: 80%, Manual: 65%
Marijan et al. [14]	2019	Automated: 88%, Manual: 72%
My collected data accuracy	2025	Automated: ~85%, Manual: ~75%

Random Forest Algorithm

Random forests are a popular supervised machine learning algorithm that can handle both regression and classification tasks. In our study we are going to use Random Forest Classification algorithm.

How it works

The Random Forest Classification algorithm operates by constructing multiple decision trees during the training phase. Each tree is built using a random subset of both the data and the features. When making a prediction, the Random Forest combines the predictions of all individual trees. For classification tasks, it takes the majority vote from all trees to assign a class label. This approach improves accuracy and helps prevent overfitting, as it relies on the collective decision of several trees rather than a single one. The random selection of data and features contributes to the model's robustness, enabling it to handle complex relationships and diverse patterns in the data effectively.

Results and Analysis

To use Random Forest Classification in our dataset we are going to use Experience as an independent variable and manual and automated test as dependent variables. After the necessary data cleaning process, we evaluate the model accuracy for manual and automated bug detection.

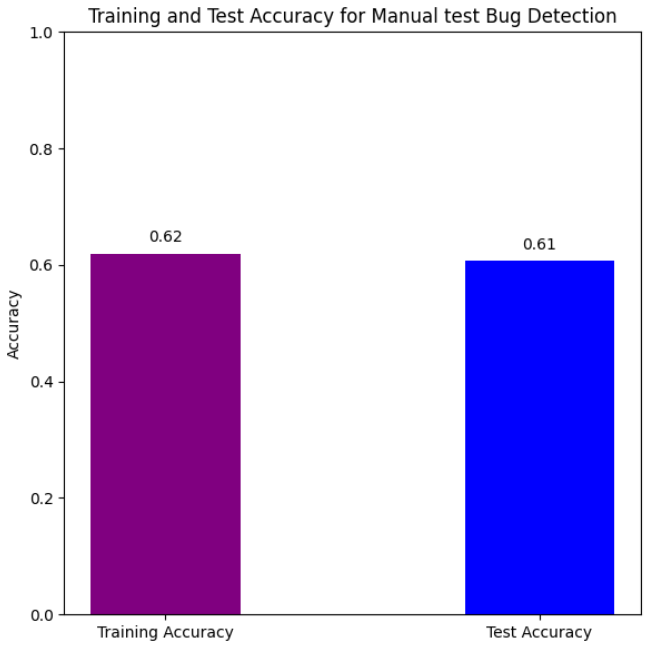

Figure: 4.3.1 Training and Test Accuracy for Manual test Bug Detection

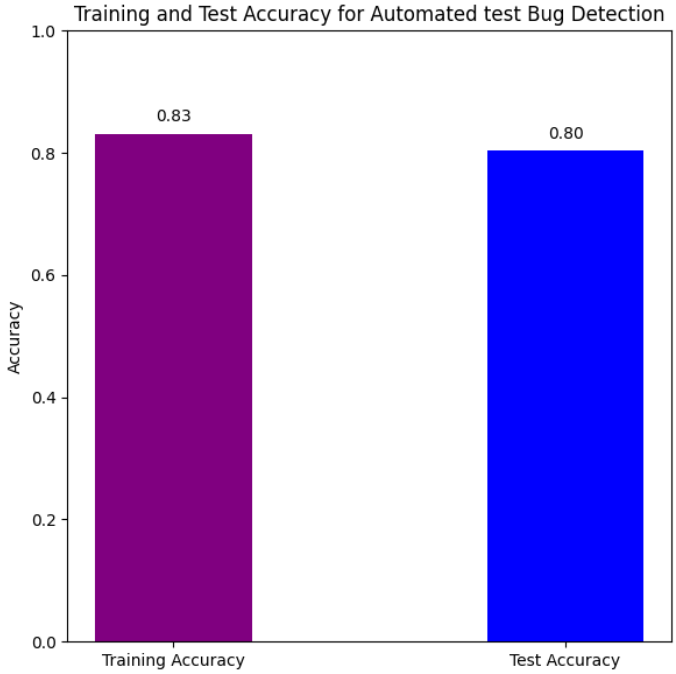

Figure: 4.3.2 Training and Test Accuracy for Automated test Bug Detection

The model performs very well on both the training and test datasets, with a small drop in test accuracy. This suggests that the model can generalize effectively with new, unseen data, and the performance is relatively stable.

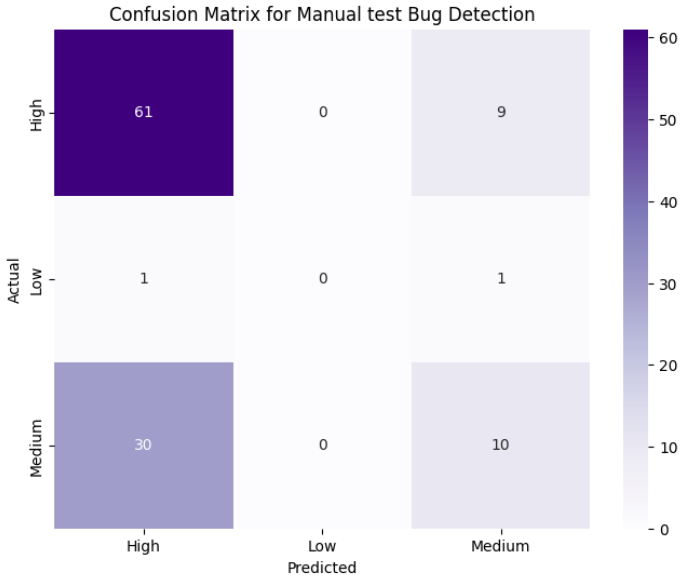

Figure: 4.3.3 Confusion Matrix for Manual test bug detection

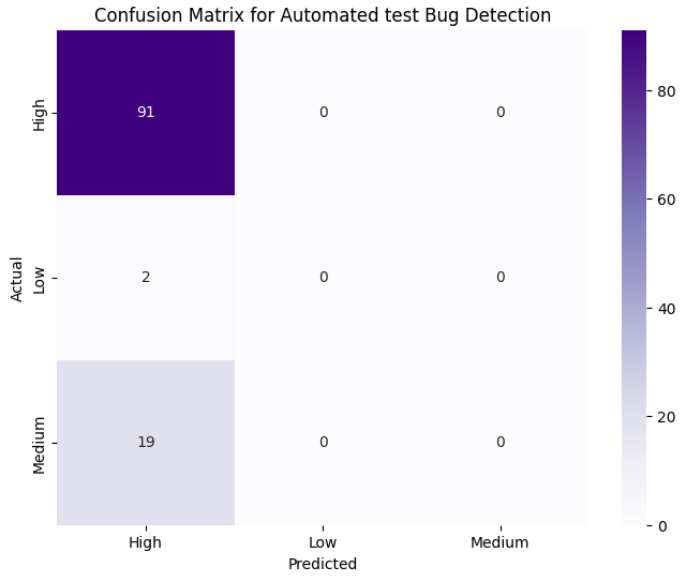

Figure: 4.3.4 Confusion Matrix for Automated test bug detection

For manual tests, the model performs well in predicting the High class, with 61 correct predictions. However, it struggles with the Medium class, misclassifying a significant number (30 instances) as High. The Low class also poses a challenge, 1 predict as High and another misclassified as Medium. Additionally, 9 High instances are incorrectly predicted as Medium. Overall, the model shows strong bias toward predicting the High class, while accuracy for Medium and Low remains limited.

```
Metrics for class 'High':
  TP: 61
  FP: 31
  FN: 9
  TN: 11
-------------------------
Metrics for class 'Low':
  TP: 0
  FP: 0
  FN: 2
  TN: 110
-------------------------
Metrics for class 'Medium':
  TP: 10
  FP: 10
  FN: 30
  TN: 62
```

```
Metrics for class 'High':
  TP: 91
  FP: 21
  FN: 0
  TN: 0
-------------------------
Metrics for class 'Low':
  TP: 0
  FP: 0
  FN: 2
  TN: 110
-------------------------
Metrics for class 'Medium':
  TP: 0
  FP: 0
  FN: 19
  TN: 93
```

Figure: 4.3.5 Random Forest Classification metrics for manual and automated classes

	precision	recall	f1-score	support
High	0.66	0.87	0.75	70
Low	1.00	0.00	0.00	2
Medium	0.50	0.25	0.33	40
accuracy			0.63	112
macro avg	0.72	0.37	0.36	112
weighted avg	0.61	0.63	0.59	112

Figure: 4.3.6 Random Forest Classification report for manual test bug detection

	precision	recall	f1-score	support
High	0.81	1.00	0.90	91
Low	1.00	0.00	0.00	2
Medium	1.00	0.00	0.00	19
accuracy			0.81	112
macro avg	0.94	0.33	0.30	112
weighted avg	0.85	0.81	0.73	112

Figure: 4.3.7 Random Forest Classification report for automated test bug detection

Let's find out the meaning of the report. High, Low and Medium are our classification for bug detection percentage.

Accuracy: Accuracy is a measure of how often a machine learning model makes correct predictions.

$$Accuracy = \frac{True\ positive\ + True\ Negative}{True\ positive\ + True\ Negative + False\ Positive + False\ negative}$$

Precision: The model's ability to correctly identify High instances. This means that 66% of the instances predicted as High were correct.

$$Precision = \frac{True\ positive}{True\ positive + True\ Negative}$$

Recall: 87% of the actual High instances were correctly identified by the model, meaning it was able to correctly detect most of the High cases.

$$Recall = \frac{True\ positive}{True\ positive + False\ Negative}$$

F1-Score: The harmonic mean of precision and recall, balancing both. The F1-score of 0.75 indicates a good balance between the two metrics. This means the model performs fairly well in the High class.

$$F1 - Score = \frac{2\ x\ Pression\ x\ Recall}{Pression + Recall}$$

Support: There are 70 actual High instances in our dataset, providing the basis for calculating these metrics.

Manual bug detection the model struggles with predicting Medium and Low classes accurately. There is good recall for the High class, but the overall accuracy and performance could be improved, especially for Medium.

Automated bug detection the model performs exceptionally well for High, with 100% recall. However, it fails to predict the Low and Medium classes effectively, resulting in a 0% recall and F1-score for these classes.

Finally, we can say that for manual test the model shows some challenges in predicting the High class accurately, with a significant number of High instances being misclassified as Low. For automated test models perform excellently with a high accuracy in predicting both Low and High classes, with minimal errors.

Logistic Regression Classification Algorithm

Logistic Regression is a classification algorithm used for binary and multiclass classification problems. It is used to model the probability of a categorical dependent variable based on one or more independent variables.

How it works

Logistic Regression is a classification algorithm that predicts probabilities using the sigmoid function. It calculates a weighted sum of input features and maps the result to a probability between 0 and 1. If the probability exceeds a threshold (usually 0.5), it assigns one class; otherwise, it assigns the other. The model optimizes parameters using

maximum likelihood estimation (MLE) and works best for linearly separable data. It is efficient, interpretable, and widely used for binary and multi-class classification.

Results and Analysis

To use Logistic Regression Classification in our dataset, we consider Experience as an independent variable and Manual and Automated Testing as dependent variables. After performing the necessary data cleaning process, we applied the Logistic Regression model to analyze the relationship between experience and defect detection efficiency. The model predicts the probability of a defect being detected through manual or automated testing. Finally, we evaluate the model accuracy, precision, recall, and F1-score to assess the effectiveness of Logistic Regression in bug detection.

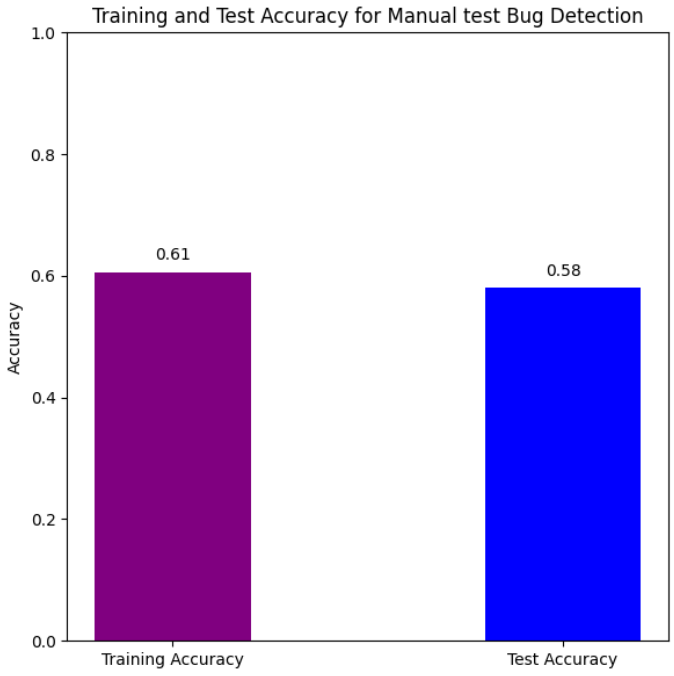

Figure: 4.3.8 Logistic Regression Training and Test accuracy for Manual test

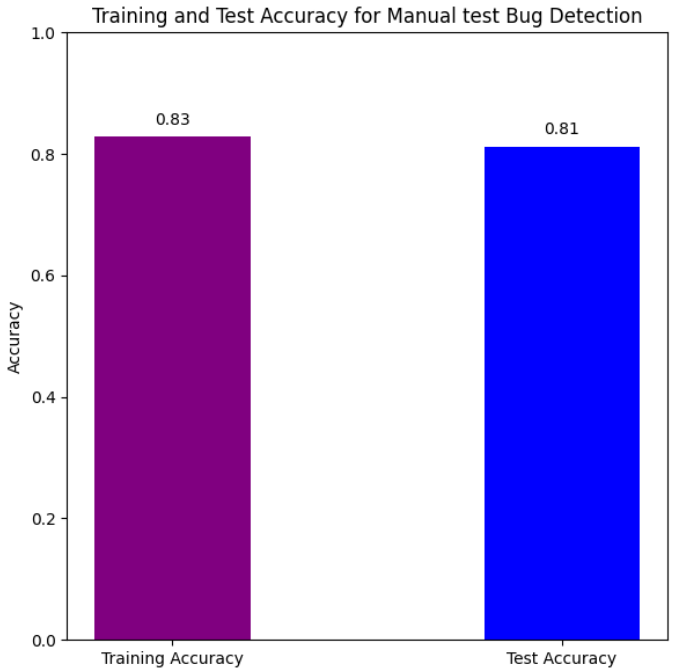

Figure: 4.3.9 Logistic Regression Training and Test accuracy for Automated test

The model shows strong performance on both the training and test datasets, with only a slight decrease in test accuracy. This indicates that the model generalizes well with new, unseen data, maintaining stable performance.

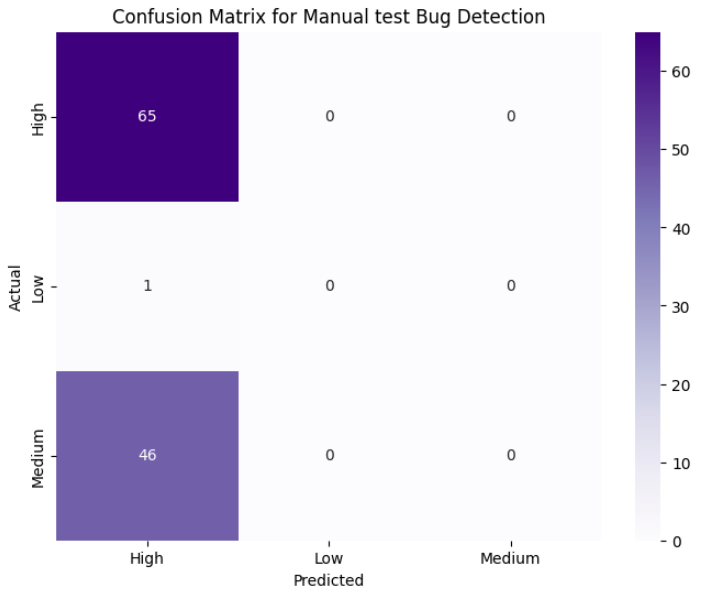

Figure: 4.3.10 Logistic Regression Classification Confusion Matrix for Manual Test

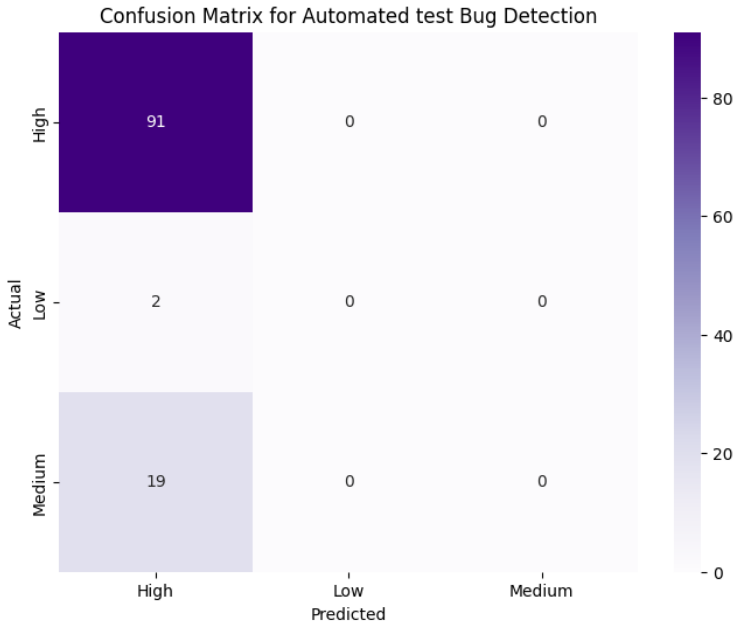

Figure: 4.3.11 Logistic Regression Classification Confusion Matrix for Automated Test

Both models show strong performance in predicting the High category, but struggle to predict Medium values and show some limitations in predicting Low values.

	precision	recall	f1-score	support
High	0.58	1.00	0.73	65
Low	1.00	0.00	0.00	1
Medium	1.00	0.00	0.00	46
accuracy			0.58	112
macro avg	0.86	0.33	0.24	112
weighted avg	0.76	0.58	0.43	112

Figure: 4.3.12 Logistic Regression Classification Report for Manual test bug detection

	precision	recall	f1-score	support
High	0.81	1.00	0.90	91
Low	1.00	0.00	0.00	2
Medium	1.00	0.00	0.00	19
accuracy			0.81	112
macro avg	0.94	0.33	0.30	112
weighted avg	0.85	0.81	0.73	112

Figure: 4.3.13 Logistic Regression Classification report for Automated test bug detection

The model performs well in detecting High instances with a recall of 1.00, but struggles with Low and Medium, showing no recall for Low and Medium categories. The overall accuracy is 0.58, indicating limited model effectiveness. For automated test model performs better than the manual test, with high precision and recall for High (1.00 recall and 0.90 F1-score). However, it also struggles with Low and Medium, showing no recall for these categories. The overall accuracy is 0.81, suggesting stronger performance than the manual test.

Statistical Analysis

In this thesis, statistical analysis plays a vital role in extracting meaningful insights from the dataset. It helps quantify the differences in defect detection, efficiency, and cost-effectiveness between manual and automated testing. By applying statistical methods, the study identifies patterns and relationships that support evidence-based conclusions. This approach ensures the reliability of the findings and aids in making informed decisions about optimal testing strategies.

Let's look at the Experience and Manual and Automated test bug detection percentage graph.

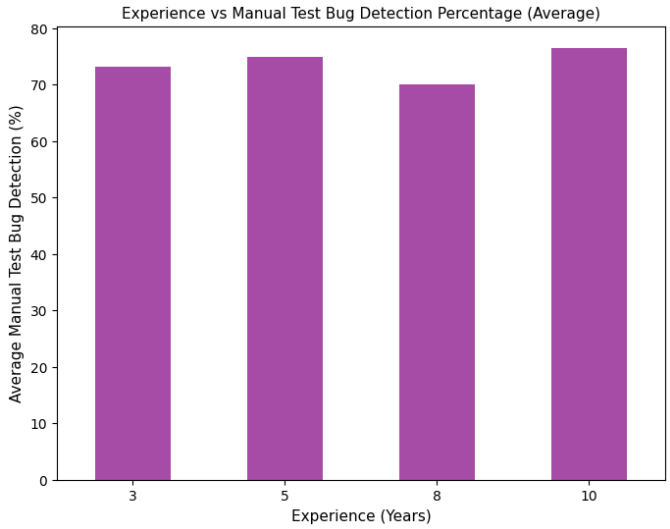

Figure: 4.3.14 Experience and Manual Test Bug Detection Percentage

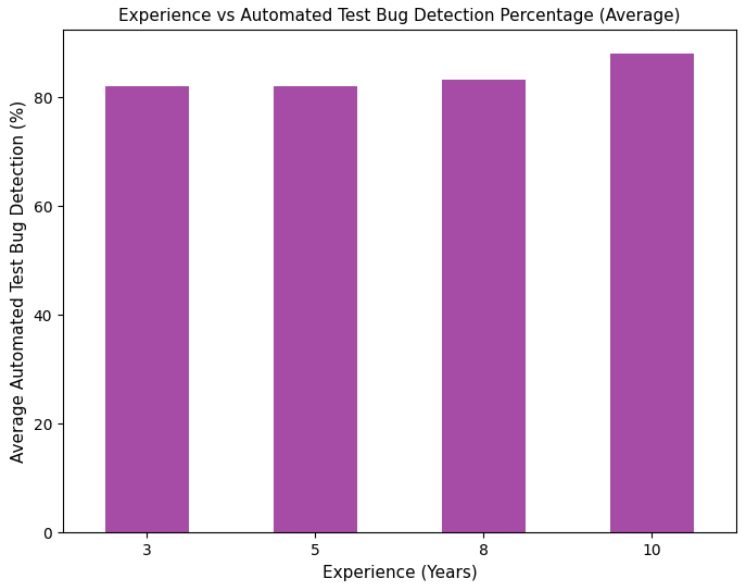

Figure: 4.3.15 Experience and Automated Test Bug Detection Percentage

The two bar charts compare the average bug detection percentages for manual testing and automated testing across different levels of experience (3, 5, 8, and 10 years). The manual testing chart shows that the detection percentage remains consistent, around 75%, regardless of experience. In contrast, the automated testing chart shows a consistently higher detection rate, averaging around 85%, with little variation across experience levels. This suggests that automated testing offers more reliable and consistent results, while manual testing effectiveness is less dependent on experience, indicating that the tool's capabilities may have a greater influence on automated testing outcomes. These insights highlight the potential advantage of automation over manual testing, especially when considering long-term reliability.

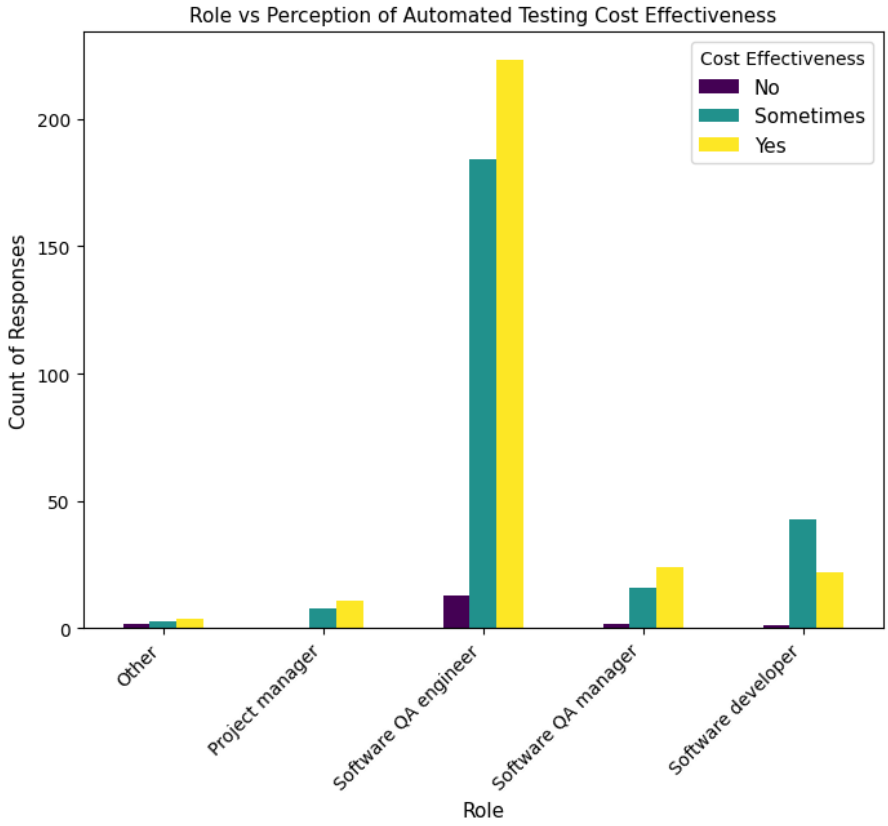

Figure: 4.3.16 Role and Perception of Automated Testing Cost Effectiveness

Initial cost for test automation is high. Is it cost effective for the long run let's find out. The bar chart illustrates the perception of automated testing cost effectiveness across various software roles. Most Software QA engineers believe automated testing is cost effective, with a high count of responses under Yes and Sometimes. Similarly, a significant portion of Software QA managers also support its cost effectiveness. In contrast, Software developers show mixed opinions, with a notable portion selecting Sometimes. Project managers and others provided fewer responses overall but still leaned toward positive views. These results indicate that individuals directly involved in quality assurance tend to perceive automated testing as more cost-effective than those in other roles. This supports the argument that domain-specific exposure influences perception of automation benefits.

Now explore the limitations of both manual and automated testing.

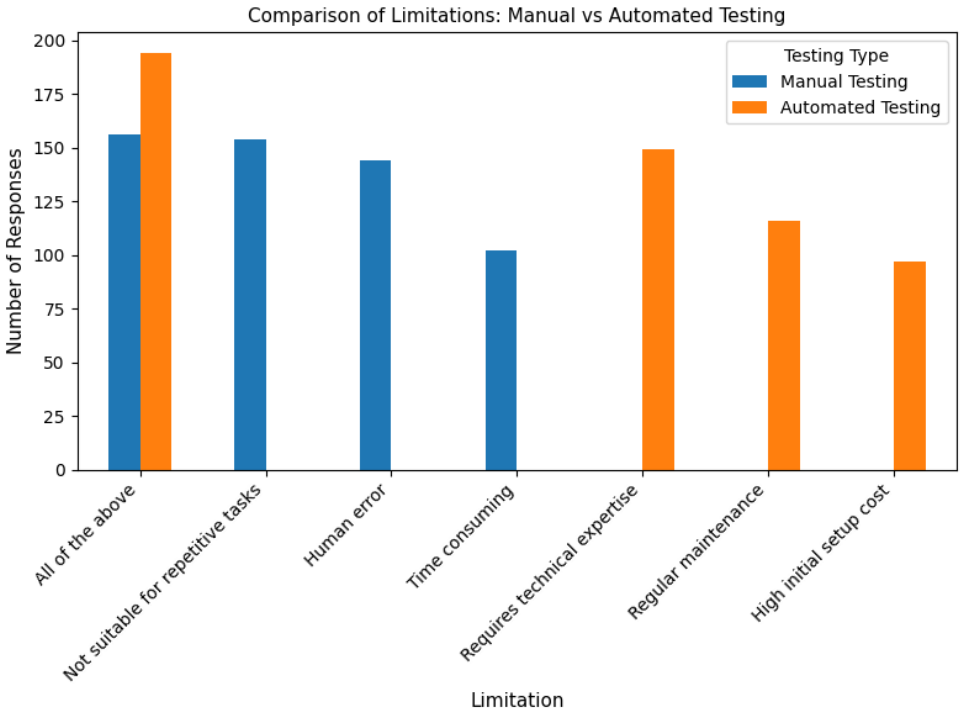

Figure: 4.3.17 Comparison of Limitations: Manual and Automated Testing

The bar chart compares the perceived limitations of manual and automated testing. For manual testing, the most reported issues are “not suitable for repetitive tasks,” “human error,” and “time consuming.” These suggest that manual processes are prone to inefficiency and inconsistency. In contrast, the main limitations of automated testing include “requires technical expertise,” “regular maintenance,” and “high initial setup cost.” These reflect the technical and financial barriers associated with automation. Interestingly, both testing types had high counts under “all of the above,” indicating multiple challenges exist for each. This comparison highlights that while manual testing struggles with human-related limitations, automated testing faces technical and resource-based constraints.

Finally find out the challenges in manual to automation testing transition in your company.

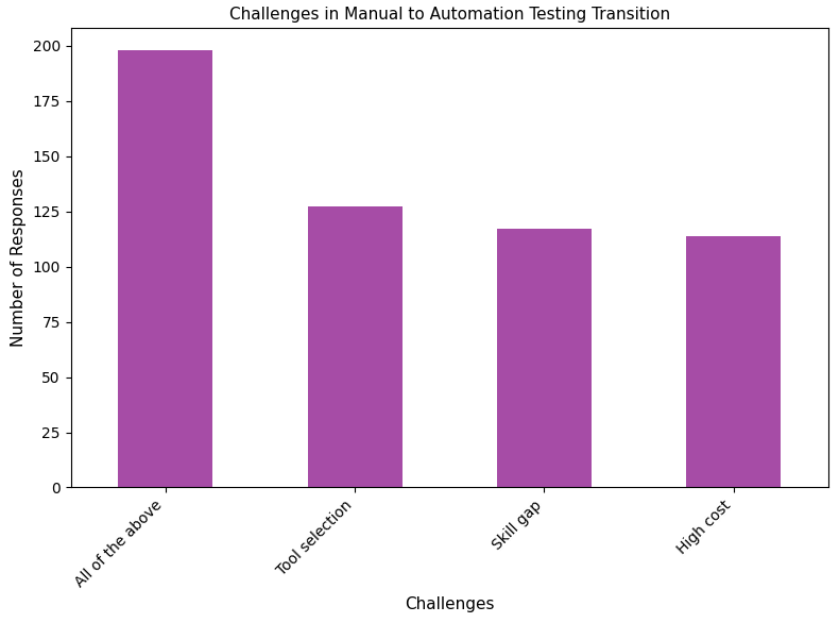

Figure: 4.3.18 Challenges in Manual to Automation Testing Transition

The bar chart shows the challenges faced during the transition from manual to automated testing. The most significant challenge, according to the responses, is the category "All of the above," which indicates that respondents face a combination of difficulties. Among specific challenges, tool selection, skill gaps, and high cost are identified as important obstacles. These findings suggest that organizations struggle not only with the technical aspects of selecting the right tools but also with the need for specialized skills and the financial investment required for automation. The high frequency of the "All of the above" response highlights the complexity of this transition, requiring careful planning and resource allocation.

4.4 Discussion

The results of this study highlight key insights into the effectiveness and limitations of manual and automated software testing. The Random Forest Classification and Logistic Regression Classification models demonstrated distinct performance characteristics in predicting defect detection accuracy for both testing types. Automated testing outperformed manual testing in defect detection efficiency.

The statistical analysis provided a deeper understanding of the role of experience in testing effectiveness. It was observed that automated testing maintains consistent performance across experience levels, while manual testing showed a slight decline in detection rates as experience decreased.

Furthermore, the study identified several limitations and challenges in transitioning from manual to automated testing, including tool selection, skill gaps, and the high initial cost, emphasizing the need for a hybrid approach to optimize defect detection across various testing scenarios.

CHAPTER 5

IMPACT ON SOCIETY, ENVIRONMENT AND SUSTAINABILITY

5.1 Impact on Society

The evolution of software testing methods, both manual and automated, significantly impacts the software development industry. Automated testing has the potential to improve the quality and reliability of software systems, which directly benefits society by reducing the number of defects and vulnerabilities in the software used across various sectors. This enhances consumer confidence in technology products and services, ensuring they function as expected. Furthermore, the increased efficiency of testing processes can lead to faster software releases, benefiting industries that rely on timely updates, such as healthcare, finance, and education. However, the reliance on automated testing may also result in reduced job opportunities for manual testers, which could have a negative social impact on employment within the software testing industry.

5.2 Impact on Environment

The environmental impact of software testing is often overlooked. The use of automated testing tools requires significant computing resources, which can lead to increased energy consumption. As the complexity of software systems grows, the energy demand for running large-scale automated testing frameworks also increases. This contributes to the carbon footprint of the software industry. On the other hand, efficient automated testing systems can reduce the need for repeated manual testing, thereby reducing the overall resource consumption associated with software development. As industries become more aware of their environmental responsibilities, there is a growing push to optimize the energy consumption of testing tools and adopt more sustainable practices in software development.

5.3 Ethical Aspects

The ethical implications of software testing methodologies cannot be ignored. Manual testing involves human testers, who are prone to errors, biases, and subjective decision-

making. This can sometimes lead to issues like overlooked defects or unfair testing conditions. Automated testing, while reducing human error, raises concerns regarding the transparency of testing processes. The algorithms used in automated testing may inadvertently favor certain outcomes or overlook specific scenarios, leading to ethical dilemmas regarding fairness and inclusivity. Moreover, the data used in both manual and automated testing must be handled responsibly, ensuring that sensitive information is protected, and privacy is maintained. Ethical testing requires both human judgment and responsible algorithmic design to ensure fairness and inclusivity in the software development process.

5.4 Sustainability Plan

The sustainability of software testing methods is crucial as the demand for software continues to grow. A sustainable approach to software testing involves optimizing the balance between manual and automated testing. By combining both methods, it is possible to ensure that defects are detected efficiently while minimizing the use of excessive resources. Automated testing tools should be continuously improved to minimize energy consumption and maximize their effectiveness. Furthermore, promoting the use of energy-efficient technologies, such as cloud computing, can help reduce the environmental footprint of testing processes. Organizations should also invest in training their staff to be proficient in both manual and automated testing techniques, ensuring that testing processes are both cost-effective and sustainable in the long term. Developing a culture of sustainability within the software testing field can drive positive changes and ensure that software products meet societal expectations while preserving environmental resources.

CHAPTER 6

SUMMARY, CONCLUSION, RECOMMENDATION AND IMPLICATION FOR FUTURE RESEARCH

6.1 Summary of the Study

This study aimed to compare manual and automated software testing in terms of defect detection accuracy, efficiency, cost-effectiveness, and overall performance. It involved using machine learning models such as Random Forest, Logistic Regression, and Linear Regression to analyze and predict the efficiency of each testing method. The research found that automated testing generally outperforms manual testing in defect detection and efficiency, particularly for large-scale software projects. However, manual testing remains essential for exploratory and usability testing, where human intuition and creativity are needed. The study also identified challenges in adopting automated testing, such as high initial setup costs and the need for skilled professionals. A hybrid approach combining both manual and automated testing methods was recommended to optimize software testing processes.

6.2 Conclusions

The comparison of manual and automated testing reveals that both methods have their distinct advantages and limitations. Automated testing outperforms manual testing in terms of speed, scalability, and consistency, particularly for large-scale projects and repetitive tasks like regression testing. However, manual testing remains valuable for exploratory and usability testing, where human intuition and creativity are required. The research confirms that no single testing approach is universally better than the other. A hybrid testing approach that integrates both methods can leverage the strengths of each, ensuring better software quality and performance. Additionally, machine learning techniques can improve testing accuracy and efficiency, offering valuable insights for future improvements in the field.

6.3 Implication for Further Study

Future research should focus on optimizing hybrid testing approaches that combine manual and automated testing methods, exploring how they can complement each other effectively. Further studies could investigate the integration of machine learning and artificial intelligence in automated testing to reduce the dependence on manual intervention. Additionally, exploring the cost-benefit analysis of automated testing for small-scale projects, where the upfront investment may not be justifiable, would be valuable. Research into the development of more energy-efficient automated testing tools could also help mitigate the environmental impact associated with extensive computing resources. Moreover, studying the impact of evolving software development practices, such as Agile and DevOps, on testing methodologies could provide deeper insights into the future direction of software testing. Understanding these factors will help software development teams adopt the most suitable testing strategies, enhancing both performance and sustainability in the industry.

References:

[1] Ms. Monika Gupta and Dr. R. K. Bathla, “Comparative Study of Software Testing Technique using Manually and Automated Way,” Int J Sci Res Sci Technol, Dec. 2022, doi: 10.32628/ijsrst229657.

[2] K. S. Thant and H. H. K. Tin, “THE IMPACT OF MANUAL AND AUTOMATIC TESTING ON SOFTWARE TESTING EFFICIENCY AND EFFECTIVENESS.” vol. 3, no. 3, pp. 90-91, May 2023.

[3] E. Enoiu, D. Sundmark, A. Causevic, and P. Pettersson, “A Comparative Study of Manual and Automated Testing for Industrial Control Software,” in Proceedings - 10th IEEE International Conference on Software Testing, Verification and Validation, ICST 2017, Institute of Electrical and Electronics Engineers Inc., May 2017, pp. 412–417. doi: 10.1109/ICST.2017.44.

[4] D. S. Nagabushanam, S. D. S, V. S. Dharinya, N. S. Roopa, and A. Arun, “A Review on the Process of Automated Software Testing,” arXiv (Cornell University). Cornell University, Jan. 01, 2022. doi: 10.48550/arxiv.2209.03069.

[5] G. J. Myers, C. Sandler, and T. Badgett, The Art of Software Testing, 3rd ed. Hoboken, NJ, USA: Wiley, 2011, pp. 23–45.

[6] R. Patton, Software Testing, 2nd ed. Indianapolis, IN, USA: Sams Publishing, 2006, p. 87.

[7] A. Bertolino, "Software Testing Research: Achievements, Challenges, Dreams," Future of Software Engineering (FOSE '07), Minneapolis, MN, USA, 2007, pp. 85-103, doi: 10.1109/FOSE.2007.25.

[8] C. Kaner, J. Bach, and B. Pettichord, Lessons Learned in Software Testing: A Context-Driven Approach. New York, NY, USA: Wiley, 2008.

[9] J. A. Whittaker, Exploratory Software Testing: Tips, Tricks, Tours, and Techniques to Guide Test Design. Boston, MA, USA: Addison-Wesley, 2009.

[10] M. Fewster and D. Graham, Software Test Automation: Effective Use of Test Execution Tools, Boston, MA, USA: Addison-Wesley, 1999, pp. 112–135.

[11] H. V. Gamido and M. V. Gamido, “Comparative review of the features of automated software testing tools,” International Journal of Electrical and Computer Engineering, vol. 9, no. 5, pp. 4473–4478, Oct. 2019, doi: 10.11591/ijece.v9i5.pp4473-4478.

[12] R. M. Florea, “The Software Tester: An Exploration of the Skills and Practice of the Role,” 2023.

[13] A. Miller, "A schooling on their implications for software engineering [trends]," in IEEE Potentials, vol. 19, no. 5, pp. 11-13, Dec. 2000-Jan. 2001, doi: 10.1109/45.890083.

[14] D. Marijan, A. Gotlieb, and S. Sen, “Test automation: The quest for the holy grail,” ACM Comput. Surv., vol. 52, no. 2, pp. 1–29, 2019. [Online]. Available: https://doi.org/10.1145/3295748

[15] S. Izzat and N. N. Saleem, “Software Testing Techniques and Tools: A Review,” Journal of Education and Science, vol. 32, no. 2, pp. 31–40, Jun. 2023, doi: 10.33899/edusj.2023.137480.1305.

[16] P. S. Kochhar, F. Thung, N. Nagappan, T. Zimmermann and D. Lo, "Understanding the Test Automation Culture of App Developers," 2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST), Graz, Austria, 2015, pp. 1-10, doi: 10.1109/ICST.2015.7102609.

[17] N. E. Fenton and M. Neil, "A critique of software defect prediction models," in IEEE Transactions on Software Engineering, vol. 25, no. 5, pp. 675-689, Sept.-Oct. 1999, doi: 10.1109/32.815326.

[18] D. P. Wangoo, "Artificial Intelligence Techniques in Software Engineering for Automated Software Reuse and Design," 2018 4th International Conference on Computing Communication and Automation (ICCCA), Greater Noida, India, 2018, pp. 1-4, doi: 10.1109/CCAA.2018.8777584.

[19] Ö. F. Arar and K. Ayan, “Software defect prediction using machine learning techniques,” vol. 49, pp. 75–86, 2017. [Online]. Available: https://doi.org/10.1016/j.asoc.2016.08.044

[20] Y. Z. Bala, P. A. Samat, K. Y. Sharif and N. Manshor, "Current Software Defect Prediction: A Systematic Review," *2022 Applied Informatics International Conference (AiIC)*, 2022, pp. 117-121, doi: 10.1109/AiIC54368.2022.9914586.

[21] M. Gerlh, E. Y. Nakagawa, and J. C. Maldonado, “Automated test case prioritization using deep learning techniques,” J. Softw. Test. Verif. Reliab., vol. 29, no. 4, e1738, 2019.

[22] M. V. Mäntylä, B. Adams, and G. Destefanis, “Mining sentiment from software development issues and commit comments,” Empir. Softw. Eng., vol. 23, no. 6, pp. 3425–3451, 2018.

[23] S. M. Shah and T. Mahmood, “Test automation challenges in agile software development,” Int. J. Softw. Eng. Appl., vol. 10, no. 3, pp. 1–15, 2019. [Online]. Available: https://doi.org/10.5121/ijsea.2019.10301

[24] J. Itkonen, M. Mäntylä, and C. Lassenius, “The role of the tester’s knowledge in exploratory software testing,” IEEE Trans. Softw. Eng., vol. 39, no. 5, pp. 707–724, 2013. [Online]. Available: https://doi.org/10.1109/TSE.2012.63

[25] V. Garousi and J. Zhi, “A survey of software test automation practices in software industry,” J. Syst. Softw., vol. 125, pp. 231–245, 2017.

[26] S. Ali et al., “Software testing challenges in the cloud,” IEEE Softw., vol. 32, no. 3, pp. 30–36, 2015. [Online]. Available: https://doi.org/10.1109/MS.2015.51

[27] D. G. W. Birch and D. M. Appleton, Software Engineering: Modern Approaches, 2nd ed. New York, NY, USA: McGraw-Hill, 2014, pp. 112–127.

[28] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, “Systematic mapping studies in software engineering,” Inf. Softw. Technol., vol. 51, no. 1, pp. 2–21, 2008.

241-25-050

ORIGINALITY REPORT

10%	7%	3%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	dspace.daffodilvarsity.edu.bd:8080 Internet Source	2%
2	Submitted to Daffodil International University Student Paper	2%
3	"Innovations in Cybersecurity and Data Science", Springer Science and Business Media LLC, 2024 Publication	1%
4	www.coursehero.com Internet Source	1%
5	Submitted to Liverpool John Moores University Student Paper	1%
6	Panagiotis Leloudas. "Introduction to Software Testing", Springer Science and Business Media LLC, 2023 Publication	1%
7	www.diva-portal.org Internet Source	<1%
8	mkt-pre.katalon.com Internet Source	<1%
9	Submitted to The University of Manchester Student Paper	<1%
10	Amit Kumar Tyagi, Shrikant Tiwari, S. V. Nagaraj. "Quantum Computing - The Future of Information Processing", CRC Press, 2025 Publication	<1%