Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 01 , January - 2026

AI-Assisted Visual Test and Bug Management Platform for Manual Testers

Enhancing Manual Software Testing Efficiency

Mansi Rajgopal Kulkarni
Department of Software Engineering
Birla Institute of Technology and Science (BITS) Pilani
Pune, India

***Abstract* - Manual software testing remains a critical phase in the software development lifecycle; however, it is often time-consuming, error-prone, and heavily dependent on human effort for bug identification, documentation, and test case management. With the increasing complexity of modern applications, traditional bug tracking tools lack intelligent assistance for visual defect detection and structured test management. This paper presents the design and development of an AI-assisted visual test and bug management platform aimed at improving the efficiency and accuracy of manual testers. The proposed system integrates visual-based bug reporting, AI-driven bug description generation, reusable test case management, and analytical dashboards within a unified platform. The solution leverages artificial intelligence models for image-based issue understanding and natural language processing to generate structured bug reports automatically. A modular architecture is implemented using modern web technologies to ensure scalability, usability, and ease of integration. Experimental evaluation demonstrates that the platform significantly reduces bug reporting time, improves defect documentation quality, and enhances tester productivity. The system provides a practical and cost-effective solution for teams transitioning from traditional manual testing processes to AI-assisted quality assurance workflows.**

Keywords - Manual Testing; Bug Management; Visual Testing; Artificial Intelligence; Software Quality Assurance

I. INTRODUCTION

Software quality assurance plays a vital role in delivering reliable and user-friendly applications. Despite advancements in automated testing, manual testing continues to be widely used due to its flexibility, exploratory capabilities, and ability to evaluate user experience aspects. However, manual testers face challenges such as repetitive test execution, inconsistent bug documentation, lack of visual context, and inefficient collaboration across teams. Traditional bug tracking systems primarily rely on textual descriptions, which often lead to ambiguity and misinterpretation.

Recent advancements in artificial intelligence have created opportunities to enhance manual testing workflows through intelligent assistance. Visual understanding, natural language processing, and data-driven insights can significantly improve bug identification and reporting accuracy. This research focuses on addressing the gap between manual testing practices and intelligent test management tools by proposing an AI-assisted visual testing and bug management platform. The objective of the proposed system is to simplify bug reporting, enhance test case reusability, and provide actionable insights through visual dashboards, thereby improving overall testing efficiency.

II. RELATED WORK

Several studies have explored the application of artificial intelligence in software testing, particularly in automated test generation and defect prediction. Existing bug tracking tools such as Jira and Bugzilla provide structured issue management but lack AI-based visual understanding and automated bug description capabilities. Research on visual testing frameworks has primarily focused on UI comparison and automated regression testing, leaving manual testers with limited intelligent support. Recent advancements in large language models have demonstrated potential in generating test cases and defect descriptions; however, their integration into practical manual testing workflows remains limited. This work builds upon existing research by combining visual testing, AI-generated insights, and test management into a unified platform tailored for manual testers.

III. SYSTEM ARCHITECTURE

The proposed platform follows a modular and service-oriented architecture to ensure scalability and maintainability. The system consists of four primary layers: User Interface Layer, Application Layer, AI Processing Layer, and Data Management Layer.

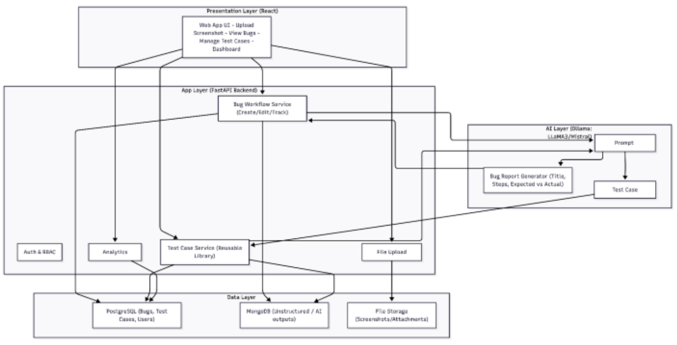
Presentation Layer (React)
Web App UI - Upload Screenshot - View Bugs - Manage Test Cases - Dashboard
App Layer (FastAPI Backend)
Bug Workflow Service (Create/Edit/Track)
AI Layer (Ollama: LLaMA3/Mistral)
Prompt
Bug Report Generator (Title, Steps, Expected vs Actual)
Test Case
Auth & RBAC
Analytics
Test Case Service (Reusable Library)
File Upload
Data Layer
PostgreSQL (Bugs, Test Cases, Users)
MongoDB (Unstructured / AI outputs)
File Storage (Screenshots/Attachments)

Fig. illustrates the high-level architecture and interaction among system modules.

The User Interface Layer provides dashboards, visual bug reporting screens, and test case management views. The

Application Layer handles business logic, authentication, and workflow orchestration. The AI Processing Layer integrates image analysis and natural language generation models to interpret visual inputs and generate structured bug descriptions. The Data Management Layer stores bugs, test cases, user data, and analytics information using relational and non-relational databases along with file storage for screenshots and attachments. The architecture supports seamless interaction between manual testers and AI components, enabling efficient defect tracking and test reuse.

IV. METHODOLOGY

The development methodology follows an iterative and incremental approach. Initially, requirements were gathered by analyzing common pain points faced by manual testers. Functional requirements included visual bug reporting, AI-generated bug summaries, reusable test cases, and dashboard analytics. Non-functional requirements focused on usability, performance, and scalability.

The system workflow begins with the tester uploading a screenshot or image representing a defect. The AI module processes the visual input and generates a structured bug description, including title, steps to reproduce, expected behavior, and actual behavior. Testers can review and edit the generated content before submission. Reusable test cases are stored in a centralized repository, allowing testers to quickly associate them with new defects. Dashboard analytics provide insights into bug trends, severity distribution, and testing progress.

V. IMPLEMENTATION AND TOOLS

The platform is implemented using modern full-stack technologies. The frontend is developed using React to provide a responsive and user-friendly interface. The backend services are implemented using FastAPI to ensure high performance and modular API design. PostgreSQL is used as the primary database for structured data storage. AI capabilities are implemented using locally hosted open-source language models to generate bug descriptions and test case suggestions without relying on paid APIs.

Additional tools such as Docker are used for containerization, enabling consistent deployment across environments. Authentication and role-based access control are implemented to ensure data security. The modular design allows future integration of additional AI models and automation features.

The implemented platform is composed of modular services that collectively support AI-assisted manual testing workflows. The key functional modules developed as part of the system are summarized as follows:

• Visual bug reporting module for uploading screenshots and capturing defect context.

• AI-assisted bug report generation module that automatically creates structured bug descriptions.

• Reusable test case management module with a centralized test case repository.

• Analytics and dashboard module for monitoring defect trends and testing progress.

VI. RESULTS AND DISCUSSION

The system was evaluated by simulating real-world manual testing scenarios. Key performance metrics included bug reporting time, documentation quality, and tester productivity. Results indicate a noticeable reduction in the time required to create detailed bug reports due to AI-assisted description generation. The quality and consistency of bug documentation improved, leading to better communication between testers and developers. The reusable test case feature minimized duplication of effort and promoted standardized testing practices. Dashboard analytics provided valuable insights for test managers, enabling data-driven decision-making. Overall, the platform demonstrated significant improvements over traditional manual testing tools.

VII. CONCLUSION AND FUTURE WORK

This paper presented the design and development of an AI-assisted visual test and bug management platform aimed at enhancing manual testing workflows. By integrating visual-based bug reporting, AI-generated documentation, reusable test cases, and analytical dashboards, the proposed system addresses key limitations of traditional bug tracking tools. Experimental results show improved efficiency, accuracy, and usability for manual testers.

Future work includes integrating automated test execution, expanding AI capabilities for defect classification, and enhancing visual analysis accuracy. The platform can also be extended to support cross-browser testing and continuous integration pipelines, further bridging the gap between manual and intelligent testing practices.

ACKNOWLEDGMENT

The author expresses sincere gratitude to the faculty and project guide at Birla Institute of Technology and Science Pilani for their guidance, support, and encouragement throughout the course of this research work.

REFERENCES

[1] I. Sommerville, *Software Engineering*, Pearson Education, 2016.

[2] G. J. Myers, C. Sandler, and T. Badgett, *The Art of Software Testing*, Wiley, 2011.

[3] A. Bertolino, “Software Testing Research: Achievements, Challenges, Dreams,” *Future of Software Engineering*, 2007.

[4] IEEE, “Standard for Software Test Documentation,” IEEE Std 829-2008.

[5] S. Thummalapenta and T. Xie, “Automated Test Case Generation,” *IEEE Software*, 2009.

[6] J. Humble and D. Farley, *Continuous Delivery*, Addison-Wesley, 2011.

[7] M. Fowler, *Refactoring: Improving the Design of Existing Code*, Addison-Wesley, 2018