Testers Hive

III B.Tech II Semester
A663E
Mini Project

Submitted By:

| 22211A05U3 Tallapally Sathwik |

Under the kind guidance of
M. Shereesha

Domain Name: Full Stack Development

Department of Computer Science and Engineering
B V Raju Institute of Technology
(UGC Autonomous, Accrediated by NBA and NAAC)
Vishnupur,Narsapur, Medak(Dist), Telangana State, India-502313.

2024-2025

CERTIFICATE OF APPROVAL

This project work (**E16**) entitled " **Testers Hive** " by Mr. Tallapally Sathwik, Registration No. 22211A05U3 under the supervision of **M. Shereesha** in the Department of Computer Science and Engineering, B V Raju Institute of Technology, Narsapur, is hereby submitted for the partial fulfillment of completing Mini Project during III B.Tech II Semester (2024 - 2025). This report has been accepted by Research Domain Computational Intelligence and forwarded to the Controller of Examination, B V Raju Institute of Technology,for the further procedures.

M. Shereesha
Assistant Professor
Department of CSE
B V Raju Institute of Technology
Narsapur.

Dr.CH.Madhu Babu
Professor and Dept.Head
Department of CSE
B V Raju Institute of Technology
Narsapur

External Examiner

Internal Examiner

Dr. L. Pallavi
Domain Incharge - " Full Stack Development "
Department of Computer Science and Engineering
B V Raju Institute of Technology
Narsapur

DECLARATION

I, the member of Research Group domain **Full Stack Development**, declare that this report titled: **Testers Hive** is our original work. All sources of information used in this report have been acknowledged and referenced respectively.

This project was undertaken as a requirement for the completion of our **III B.Tech II Sem Mini project** in Department of **Computer Science and Engineering** at **B V Raju Institute of Technology**, Narsapur. The project was carried out between 23-Dec-2024 and 26-April-2025. During this time, we as a team were responsible for the process model selection, development of the micro document and designing of the project.

This project(Testers Hive) involved the development of a custom website for Qualizeal, a software testing company.The goal was to replace third-party platforms with an in-house web solution for managing users and tasks. The application was built using the MySql, Express.js, React.js, Node.js with Vite and Tailwind CSS. Key features include role-based access control, responsive UI, dashboard views, and payment gateway integration. The platform ensures secure data handling and improves operational efficiency through centralized management.

We would like to express our gratitude to our project supervisor **M. Shereesha** for his guidance and support throughout this project. We would also like to thank our Department Head Dr.CH.Madhu babu and Domain Incharge **Dr. L. Pallavi** for her help and efforts.

We declare that this report represents Our own work, and any assistance received from others has been acknowledged and appropriately referenced.

Tallapally Sathwik (22211A05U3) _____

Guide **Project Coordinator** **Domain Incharge** **HOD/CSE**

ACKNOWLEDGEMENT

This project is prepared in fulfilment of the requirement for Research Domain **Full Stack Development**. We owe our deepest gratitude to the Department of Computer Science and Engineering
B V Raju Institute of Technology, Narsapur for providing us with an opportunity to work on this project.

I also extend our gratitude towards our project supervisor **M. Shereesha** and Department Head **Dr.CH.Madhu babu**, whose guidance and expertise have been invaluable in steering us towards success. We also thank our Research Domain Incharge **Dr. L. Pallavi** and other faculty members of the Department for their valuable feedback and suggestions.

Finally, I would like to thank our family and friends for their continuous support and encouragement throughout the project. We acknowledge the contributions of everyone who supported us in the creation of this project report.

Thank you all for your assistance and support.

The experience of working on this project will surely enrich our technical knowledge and also give us hands on experience of working on a project and help develop our team's skill set to a great extent.

ABSTRACT

This project focuses on the development of a centralized web platform for Qualizeal, a software testing company, aiming to replace dependency on third-party tools with a dedicated in-house solution. The application is designed to manage user data, streamline task assignments, and facilitate secure, role-based access to various functionalities. Built using the MySQL database along with Express.js, React.js, and Node.js (commonly referred to as the ERN stack), the platform also leverages Vite for optimized development and Tailwind CSS for responsive design. It integrates a payment gateway and custom dashboards to support efficient internal operations. By consolidating these features into a single application, the solution improves data integrity, enhances workflow efficiency, and enables complete control over project execution.

Keywords:Custom Web Platform, Company Website, ERN Stack, MySQL, User Management, Task Automation, Role-Based Access, Dashboard Interface, Vite, Tailwind CSS, Payment Gateway Integration, Centralized Operations, Internal Workflow, Data Security, Operational Efficiency

List of Figures

List of Tables

LIST OF ACRONYMS AND ABBREVIATIONS

UI User Interface

UX User Experience

API Application Program Interface

CRUD Create Read Update Delete

SQL Structure Query language

HTML HyperText Markup Language

CSS Cascading Style Sheets

JS JavaScript

QA Quality Assurance

TABLE OF CONTENTS

CHAPTER - 1

1. INTRODUCTION

In today's competitive software testing industry, companies like Qualizeal require efficient and secure platforms to manage their internal processes and client projects. Previously, Qualizeal depended on external websites and third-party tools to track users, assign tasks, and manage projects, which often resulted in limited control, potential data privacy issues, and reduced operational efficiency. To overcome these challenges, the company decided to develop a custom-built web application—Testers Hive—to centralize user management, task allocation, and workflow automation tailored specifically to Qualizeal's business needs. This project aims to enhance productivity, improve data security, and provide a scalable solution for the company's evolving operational requirements.

1.1. Background

Qualizeal operates as a software testing company that offers comprehensive testing services to various clients. The growing complexity of projects and increasing user base demanded a robust system to manage users' profiles, roles, and testing assignments efficiently. Existing solutions, which relied on third-party platforms, lacked customization and flexibility, making it difficult to address company-specific workflows and security policies. This necessitated the development of Testers Hive—a centralized internal web application built on modern web technologies to manage data securely and automate task flows.

1.2. Motivation

The motivation behind developing Testers Hive is to reduce dependency on external platforms and create a dedicated environment that aligns perfectly with Qualizeal's operational processes. The existing third-party tools posed challenges in customization, data security, and seamless communication among users. By building an in-house solution, the company aimed to achieve better control over user data, facilitate role-based access to critical features, and streamline the execution of testing projects. This project also aims to improve overall workflow efficiency and provide a user-friendly interface for both administrators and testers.

1.3. Objectives

1. To develop a secure and scalable web platform to manage users, roles, and tasks specific to Qualizeal's needs.
2. To implement role-based access control allowing differentiated permissions for admins and users.
3. To enable seamless task allocation and progress tracking within the platform.
4. To design responsive dashboards and user interfaces using modern UI frameworks.
5. To integrate payment processing features for client billing and service management.

1.4. Problem statement

- The absence of a centralized platform led to fragmented data management and reduced operational control.
- Reliance on third-party tools limited customization and posed potential security risks for sensitive user and project data.
- Inadequate automation caused delays in task assignment and project tracking.
- The company required a dedicated, flexible, and secure system to align with its specific workflows and growing scale.

1.5. Scope of Project

The scope of Testers Hive covers the development of a comprehensive web application designed to serve Qualizeal's internal operational needs. Key functionalities include:

- User registration, profile management, and role-based authentication.
- Task creation, assignment, and progress monitoring for software testing projects.
- Administrative control panel for managing users, tasks, and billing.
- Integration of payment gateway for processing client payments.
- Responsive design ensuring usability across devices.

This project focuses on building a robust backend using MySQL and Node.js and a dynamic frontend using React.js, Vite, and Tailwind CSS. The system will be designed to handle data securely and provide scalability for future feature expansion. However, the project excludes external client-facing portals and advanced AI-based testing automation, which may be considered in future phases.

CHAPTER - 2

2. LITERATURE SURVEY

In the modern software testing industry, efficient management of testing processes, user handling, and project workflows is critical for companies to maintain high-quality deliverables under tight deadlines. Traditional reliance on third-party platforms for managing testing projects has presented several challenges such as limited customization, security risks, and inefficient resource allocation. To overcome these limitations, there has been an increasing demand for centralized, tailor-made solutions that can integrate diverse internal operations while ensuring secure data management.

Software testing companies like Qualizeal, which operate in highly competitive and dynamic environments, require platforms that not only manage users and projects but also facilitate seamless communication between testers, developers, and project managers. The use of custom web applications has emerged as a practical approach to meet these needs by providing flexibility, real-time data tracking, and workflow automation.

Studies and existing solutions in the domain of internal project management for software testing emphasize the importance of user role management, task assignment automation, and reporting systems to improve overall productivity. For instance, platforms that incorporate role-based access controls ensure that sensitive data is only accessible to authorized personnel, thus enhancing security and compliance with data protection standards.

Moreover, task allocation algorithms that consider tester expertise, availability, and project priorities have been shown to optimize resource utilization and reduce turnaround times. Integration of dashboards that provide real-time analytics and progress tracking further supports decision-making by stakeholders and helps identify bottlenecks early in the development cycle.

Qualizeal's need for an internal web solution aligns with these research findings, aiming to deliver a robust platform that supports:

- Centralized user management, including authentication, role assignments, and permissions.
- Efficient task management to automate test case assignments and monitor progress.
- Secure handling of test data and results to maintain confidentiality and integrity.
- Customizable reporting features for better insights and communication between teams.

While several commercial and open-source tools address aspects of test management, few provide the comprehensive, company-specific customization and control that Qualizeal requires. This highlights the necessity of developing an in-house solution tailored precisely to the company's operational workflows.

The project also recognizes existing challenges in maintaining such systems, including ensuring scalability as the user base grows, integrating with other tools in the software development lifecycle, and maintaining user-friendly interfaces to reduce the learning curve for testers and managers.

In summary, this literature survey confirms that the development of Testers Hive for Qualizeal will fill a crucial gap in the company's operational capabilities by delivering a secure, flexible, and efficient testing management platform that aligns with modern industry practices and technological advancements.

CHAPTER - 4

3. DESIGN SPECIFICATION

The design specification of the Qualizeal Test Management System involves identifying the core requirements and functionalities necessary to support software testing teams in managing test cycles, reporting bugs, and streamlining collaboration. In this section, we address these goals by developing various design diagrams that describe the system's architecture and flow.

We understand and represent these requirements better using the following diagrams:

- Use Case Diagram
- Data Flow Diagram
- Class Diagram
- Sequence Diagram
- Activity Diagram
- State Chart Diagram

3.1. Use Case Diagram

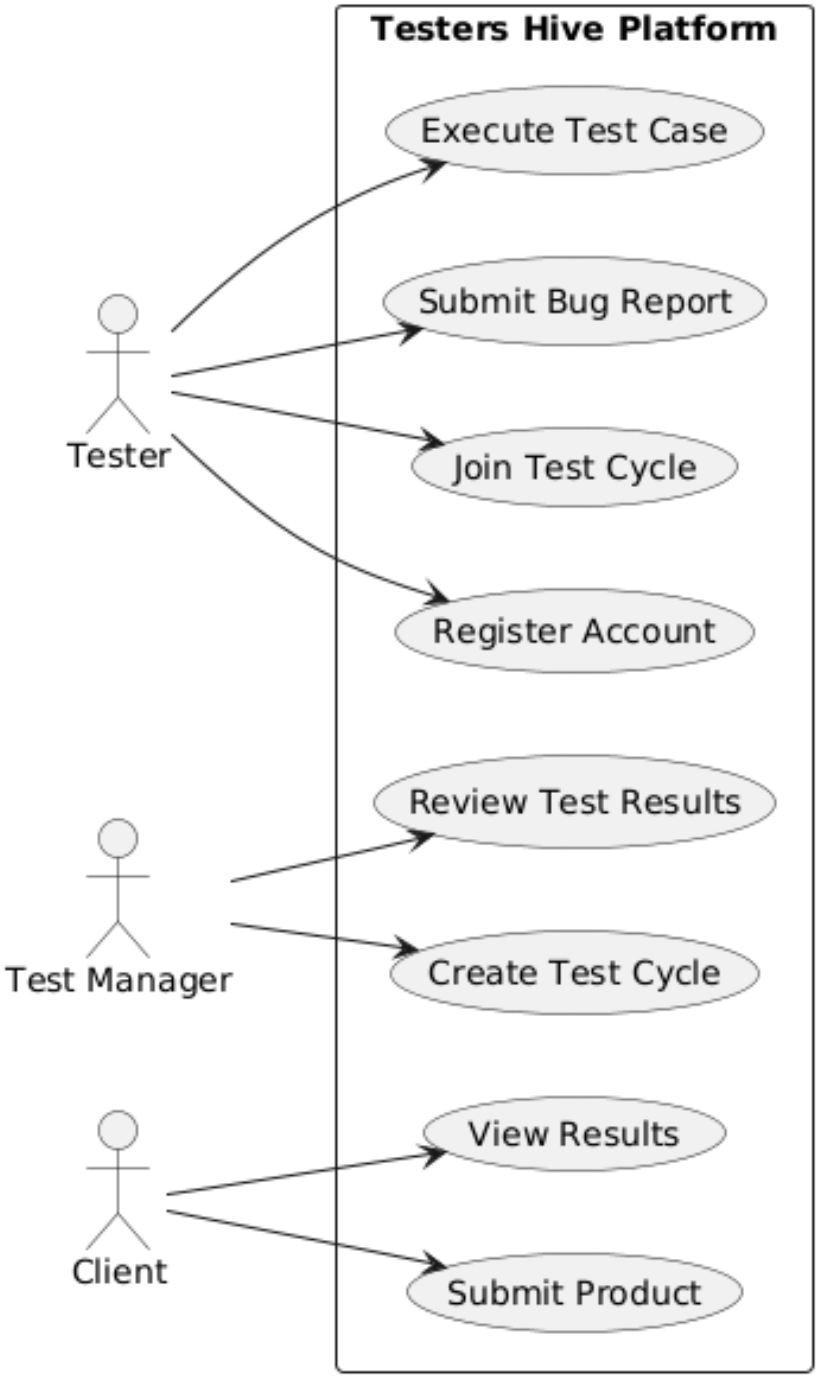

Figure 3.1: Use Case Diagram of Tester's Hive.

3.2. Data Flow Diagram

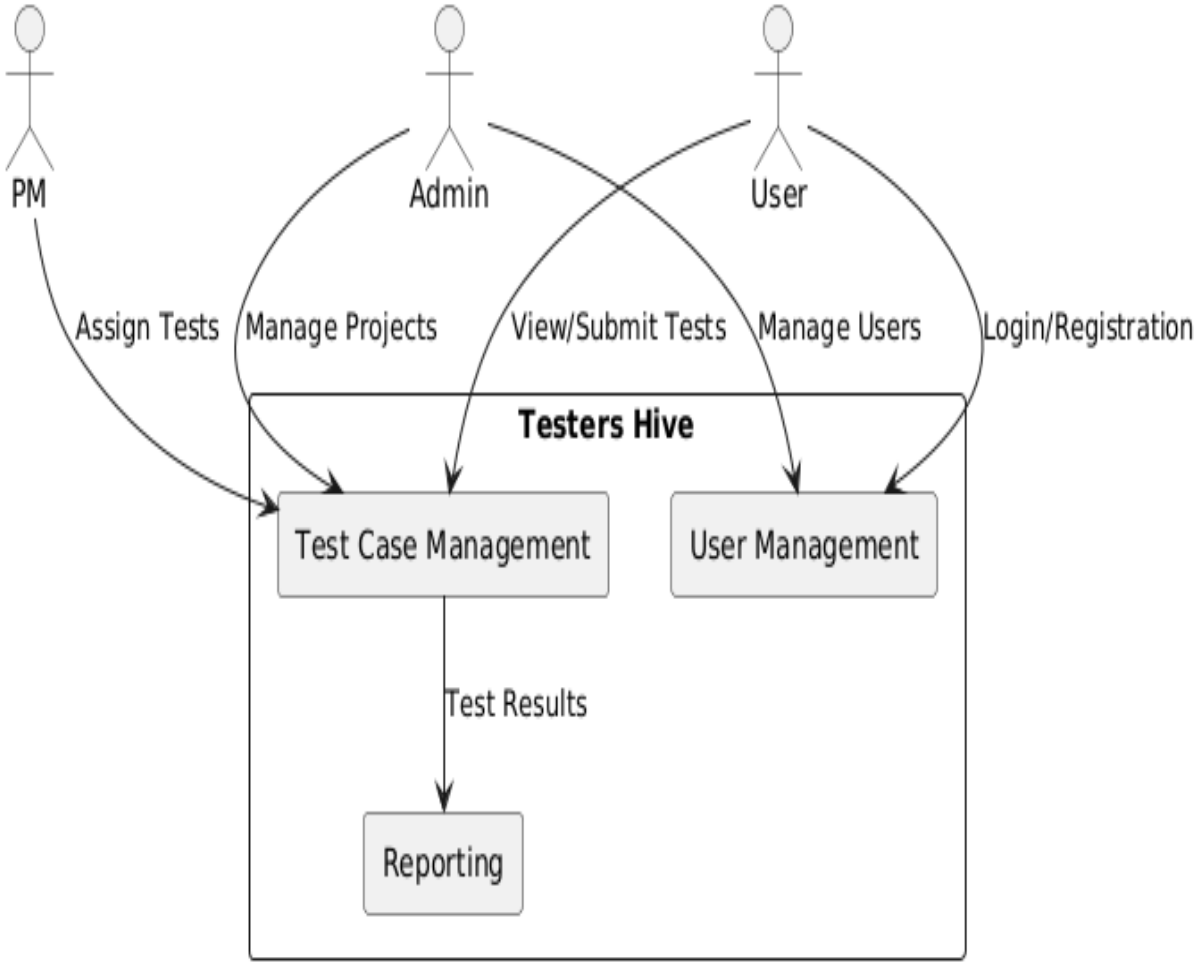

Figure 3.2: Data Flow Diagram of Tester's Hive.

3.3. Class Diagram

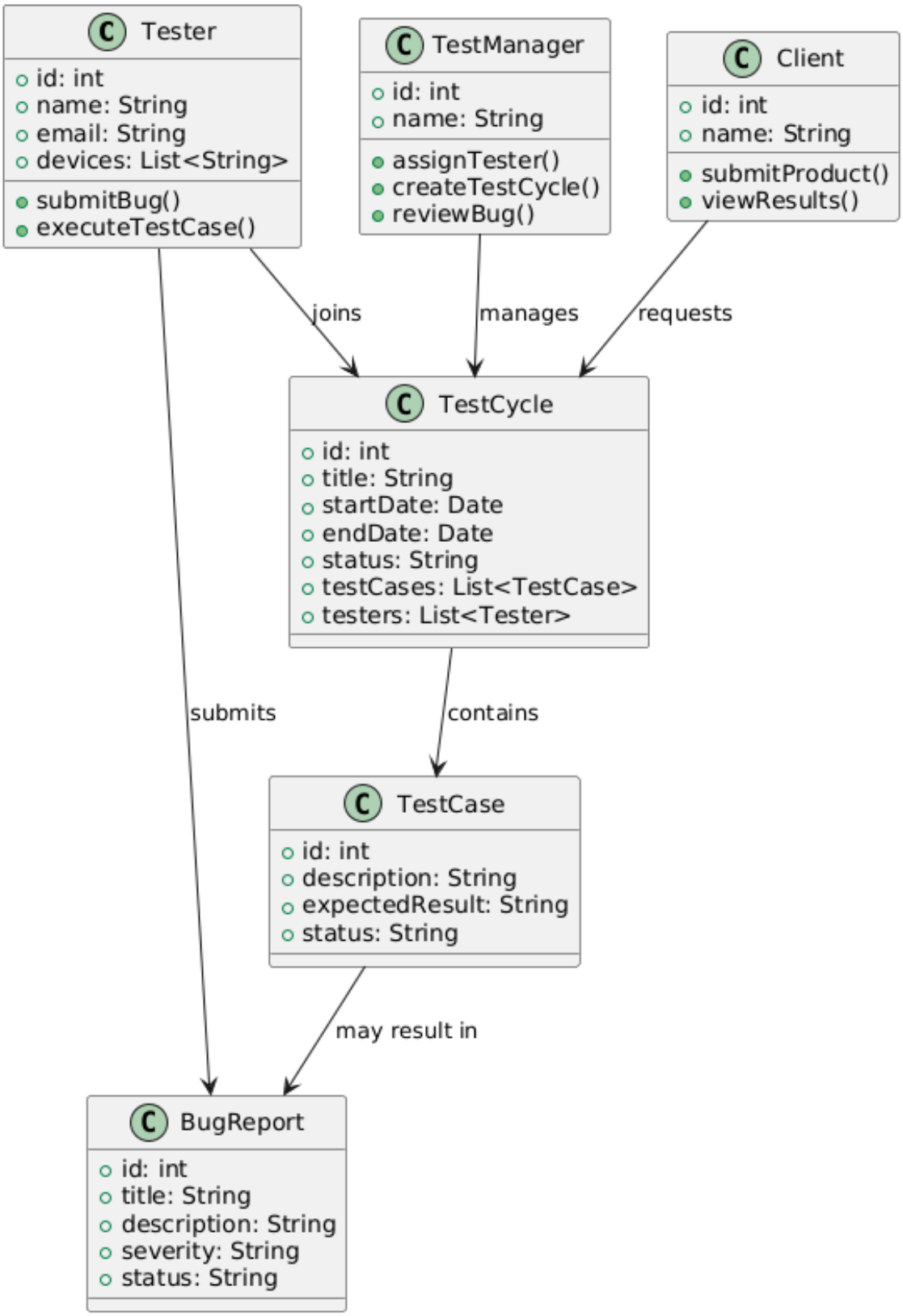

Figure 3.3: Class Diagram of Tester's Hive.

3.4. Sequence Diagram

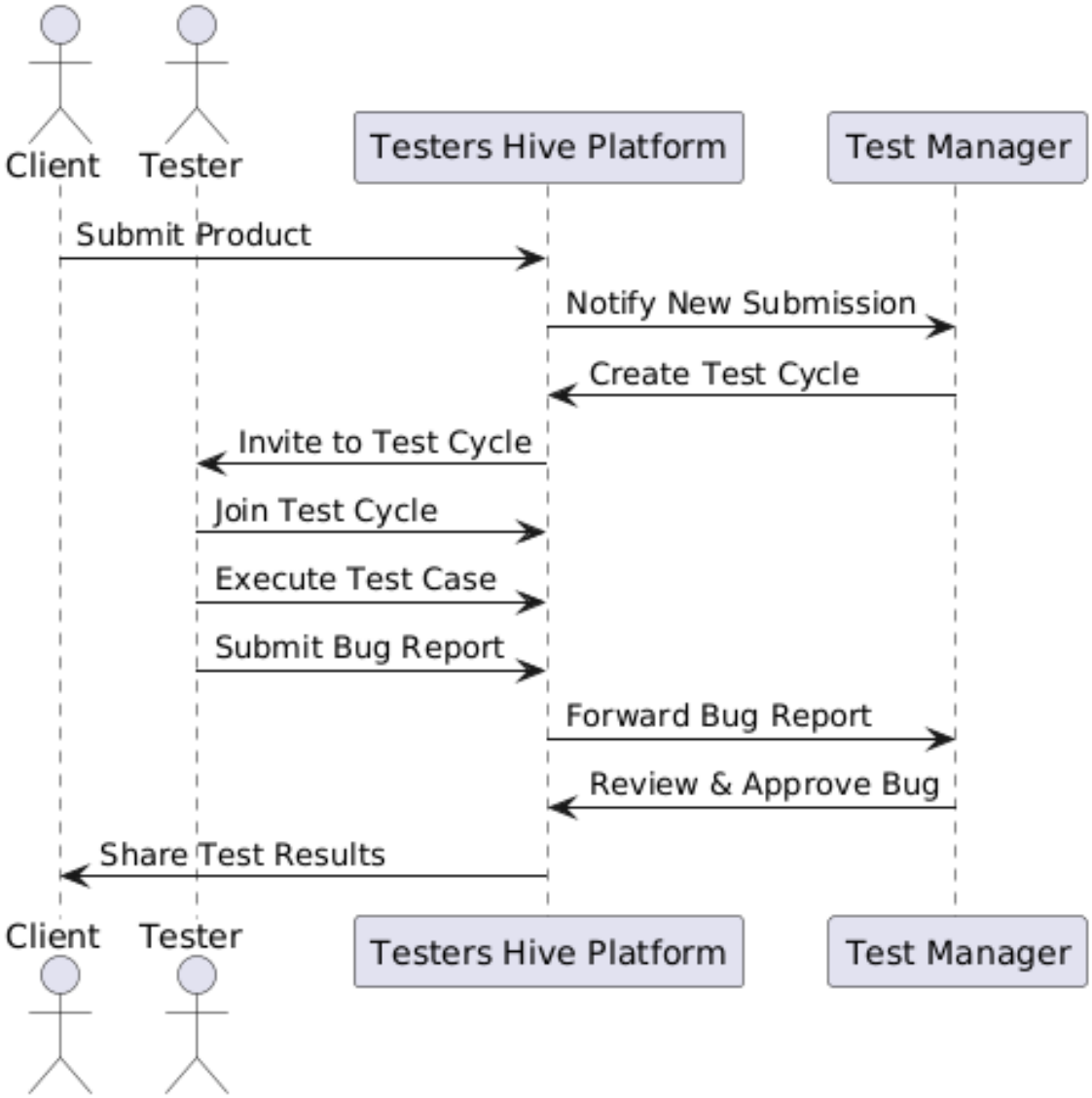

Figure 3.4: Sequence Diagram of Tester's Hive.

3.5. Activity Diagram

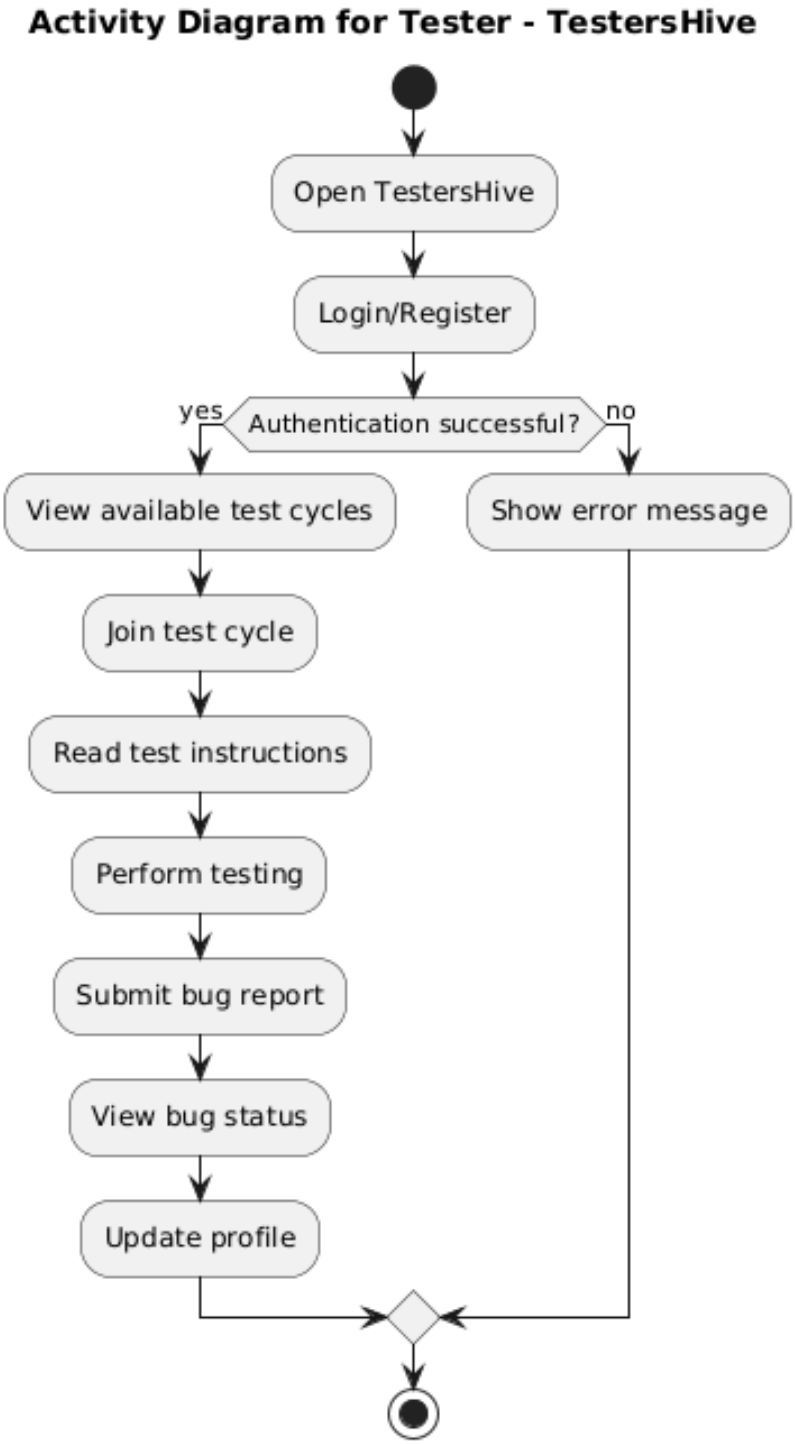

Figure 3.5: Activity Diagram of Tester's Hive.

3.6. State Chart Diagram

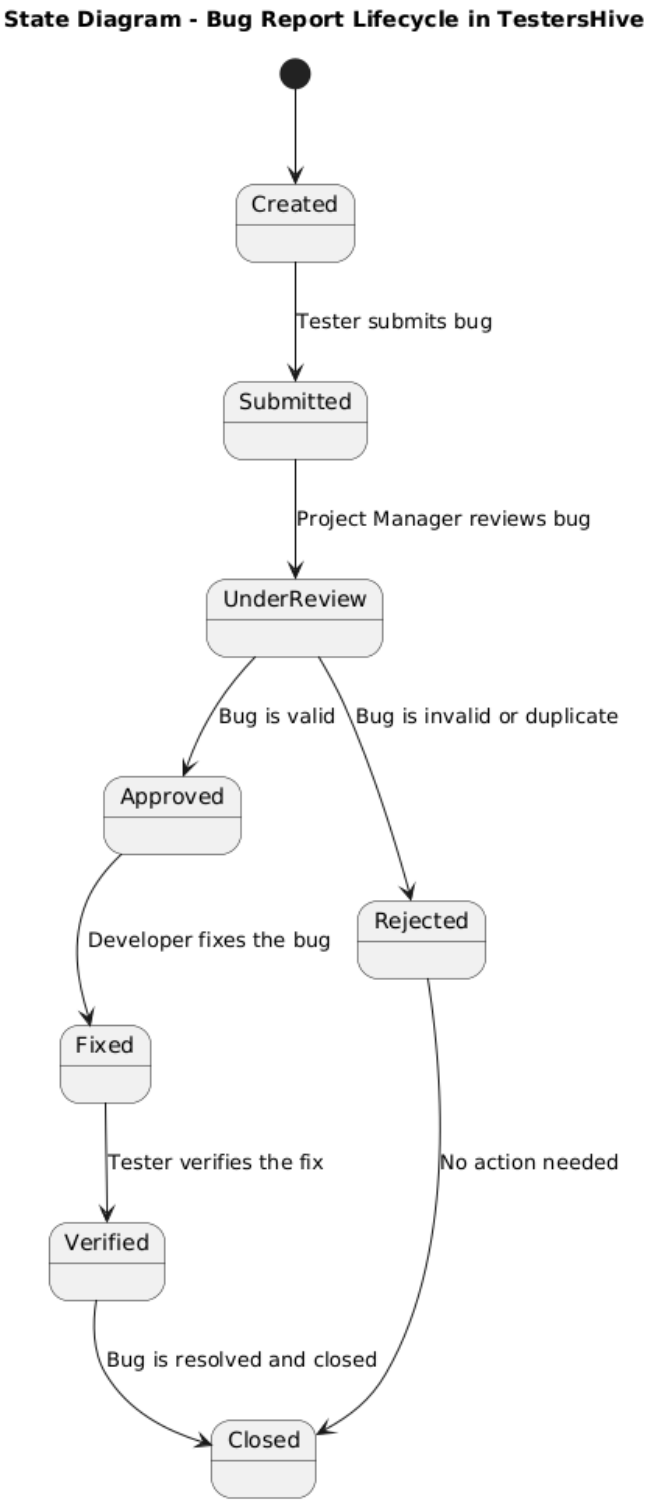

Figure 3.6: State Chart Diagram of Tester's Hive.

CHAPTER - 5

4. METHODOLOGY

4.1. Modules

The Qualizeal Test Management System is structured into several key modules, each responsible for distinct functionalities and integrated to ensure smooth workflows:

- **Frontend Module:** Developed using React.js and styled with Tailwind CSS, this module provides a responsive and user-friendly interface for all users, including testers, project managers, and administrators. It communicates with the backend via REST API calls.
- **Backend Module:** Built with Node.js and Express.js, the backend serves as the API server orchestrating requests between the frontend and the database. It consists of:
  - **Bug Management:** Handles creation, updating, tracking, and reporting of bugs found during testing.
  - **Test Cycle Management:** Manages the lifecycle of test cycles, including creation, assignment, status tracking, and closure.
  - **Authentication Service:** Manages user login, registration, and session authentication, handling all authorization requests securely.
  - **User Management:** Handles user profiles, roles, permissions, and payment-related data where applicable.
- **Database Module:** Implemented using MySQL, the database stores persistent data in organized tables:
  - **Bug Reports:** Stores detailed bug information, status, and history.
  - **Test Cycles:** Maintains data related to test cycles, including assigned test cases and their progress.
  - **User Data:** Stores user credentials, profile details, roles, and authentication tokens.
  - **Payments:** (If applicable) Contains payment transaction details and billing information.

4.2. Data Collection:

Data is collected through user interactions with the frontend interface where project managers create test cycles and test cases, and testers submit bug reports and test results. All data entered via the frontend is sent as REST API calls to the backend, which then processes and stores it in the MySQL database.

4.3. Preprocessing:

Upon receiving data, the backend validates the inputs to ensure data integrity and security. Authentication service verifies authorization tokens on sensitive requests. Test cycles and bug reports are checked for completeness and consistency before being committed to the database.

4.4. Test Cycle and Bug Management

The Test Cycle Management module allows project managers to organize testing activities into cycles and assign test cases to testers. Testers execute the assigned tests and report bugs through the Bug Management module.

Bugs are logged with detailed information, prioritized, and tracked throughout their lifecycle. Status updates are propagated back to the frontend via API responses, enabling real-time progress tracking.

4.5. System Architecture and Implementation :

The system architecture is designed for modularity and scalability:

- **Frontend:** React.js manages dynamic user interfaces, while Tailwind CSS provides utility-first styling for rapid UI development.
- **Backend:** Node.js and Express.js implement RESTful APIs and middleware for authentication, authorization, and data processing. The modular structure divides functionality into bug management, test cycle management, authentication, and user management services.
- **Database:** MySQL relational database stores structured data efficiently, enabling complex queries for reports and analytics. The database schema separates concerns by organizing bug reports, test cycles, user data, and payment records into dedicated tables.

4.6. Quality Assurance and Testing :

Qualizeal itself undergoes comprehensive testing, including unit testing of backend services, integration testing of API endpoints, and UI testing for the frontend. Automated tests ensure robustness, security, and reliability. Continuous feedback loops help refine features and maintain system stability.

4.7. System Block Diagram

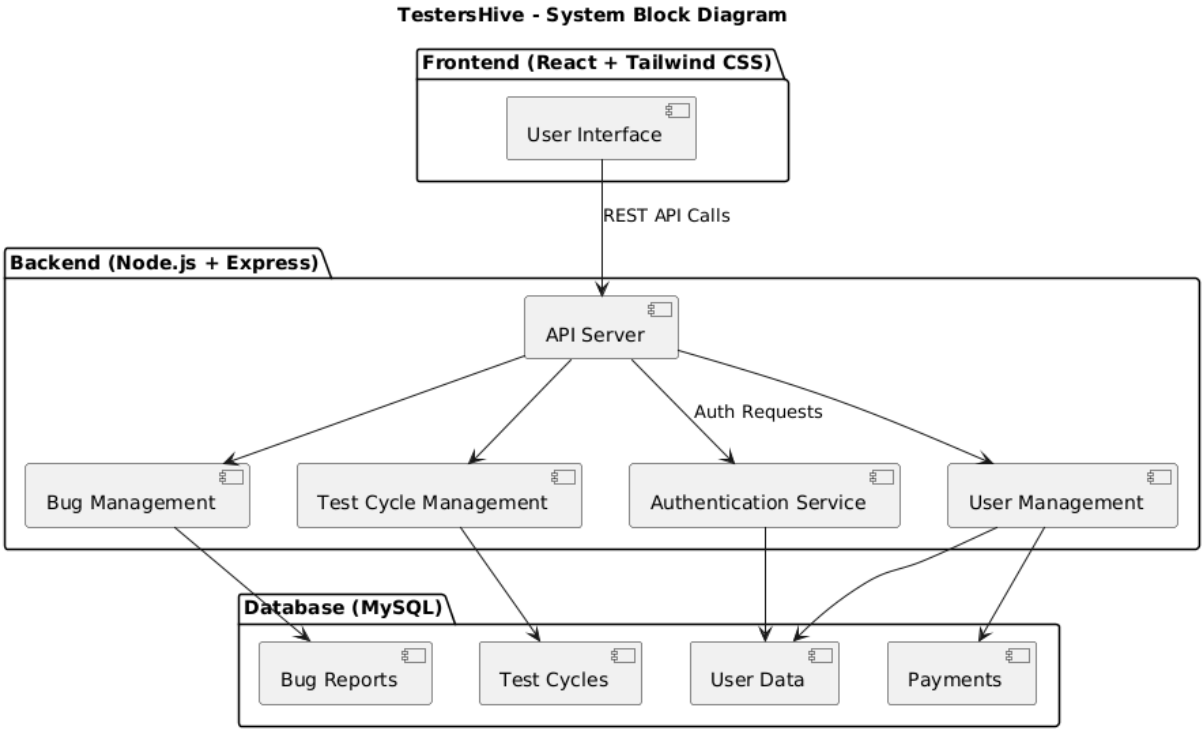

Figure 4.1: The user accesses the system through a React + Tailwind CSS interface, which communicates with a Node.js + Express backend via REST APIs. The backend handles bug tracking, test cycles, user authentication, and management, all connected to a MySQL database storing bugs, cycles, users, and payments.

CHAPTER - 6

5. IMPLEMENTATION DETAILS

TestersHive is a crowdtesting platform built using React for the frontend with Tailwind CSS for styling, and Node.js with Express on the backend. It uses MySQL as the primary database for managing users, projects, bug reports, and payments. The platform includes key features like a tester dashboard, test cycle listings, bug submission forms, and profile management. Authentication is likely handled using JWT, and the current setup supports scalable user and project management.

5.1. Technology Stack

The technology stack used in TestersHive provides a robust and scalable foundation for building a modern crowdtesting platform. The frontend is developed using React, a powerful JavaScript library for building dynamic user interfaces, combined with Tailwind CSS, a utility-first CSS framework that allows for rapid and responsive design implementation without writing custom styles. On the backend, Node.js is used as the runtime environment, and Express serves as the web framework, handling routing, middleware, and API creation efficiently. For data management, MySQL is employed as the relational database, offering structured storage for users, projects, bug reports, and payment records. This stack supports a clear separation of concerns, improves maintainability, and allows seamless communication between the client and server using RESTful APIs. Although cloud storage hasn't been integrated yet, the existing setup is well-suited for future scalability, real-time features, and third-party service integration.

5.2. System Architecture

The system architecture of TestersHive consists of three main layers. The client layer is built using React and styled with Tailwind CSS, handling the user interface and interactions such as browsing test projects, submitting bugs, and managing profiles. This layer communicates with the backend through RESTful API calls. The server layer is powered by Node.js and Express, responsible for processing client requests, managing business logic, handling authentication with methods like JWT, and interacting with the database. The database layer uses MySQL to store and organize data including user information, test cycles, bug reports, and payment details. This layered setup ensures clear separation of responsibilities, secure communication, and scalability for future feature expansions.

5.3. User Interface

The user interface of TestersHive is designed using React for building interactive and dynamic components, combined with Tailwind CSS for a clean and responsive layout. It provides testers with easy navigation through dashboards, test project listings, and bug submission forms. The interface allows users to quickly access their profiles, view ongoing testing cycles, submit detailed bug reports with attachments, and track their earnings or payment history. The use of Tailwind CSS ensures the design adapts smoothly across different devices and screen sizes, delivering a consistent user experience whether on desktop or mobile. Overall, the interface focuses on simplicity and usability to make testing tasks efficient and intuitive.

5.4. Integration

Integration in TestersHive involves connecting various components and services to ensure smooth data flow and functionality. The frontend built with React communicates with the backend APIs created using Express and Node.js through RESTful HTTP requests, enabling features like user authentication, test cycle management, and bug reporting. The backend interacts with the MySQL database to store and retrieve data such as user profiles, project details, and bug reports. Authentication tokens like JWT are integrated to secure API endpoints and manage user sessions. Additionally, integration points can include third-party services for payment processing or notifications, allowing the platform to expand its capabilities and provide a seamless experience for testers and administrators.

5.5. Security

Security in TestersHive is handled through multiple layers to protect user data and maintain platform integrity. User authentication is managed using techniques like JWT tokens, which securely verify user identities and control access to different parts of the application. Passwords are stored using strong hashing algorithms to prevent unauthorized access even if the database is compromised. The backend validates and sanitizes all incoming data to prevent common attacks such as SQL injection and cross-site scripting (XSS). Secure communication between the frontend and backend is ensured by using HTTPS, encrypting data transmitted over the network. Role-based access control limits user permissions, ensuring testers, managers, and admins can only access appropriate resources. Overall, these measures work together to provide a safe and trustworthy environment for all users.

Taking all the subsections into consideration, we were successfully able to develop a robust and efficient teeth segmentation model Tri-Level segmented CNN that can streamline segmentation operations, reduce the manual work and enhance applications in various dental fields.

CHAPTER - 7

6. OBSERVATIONS

6.1. Time Domain - Gantt Chart

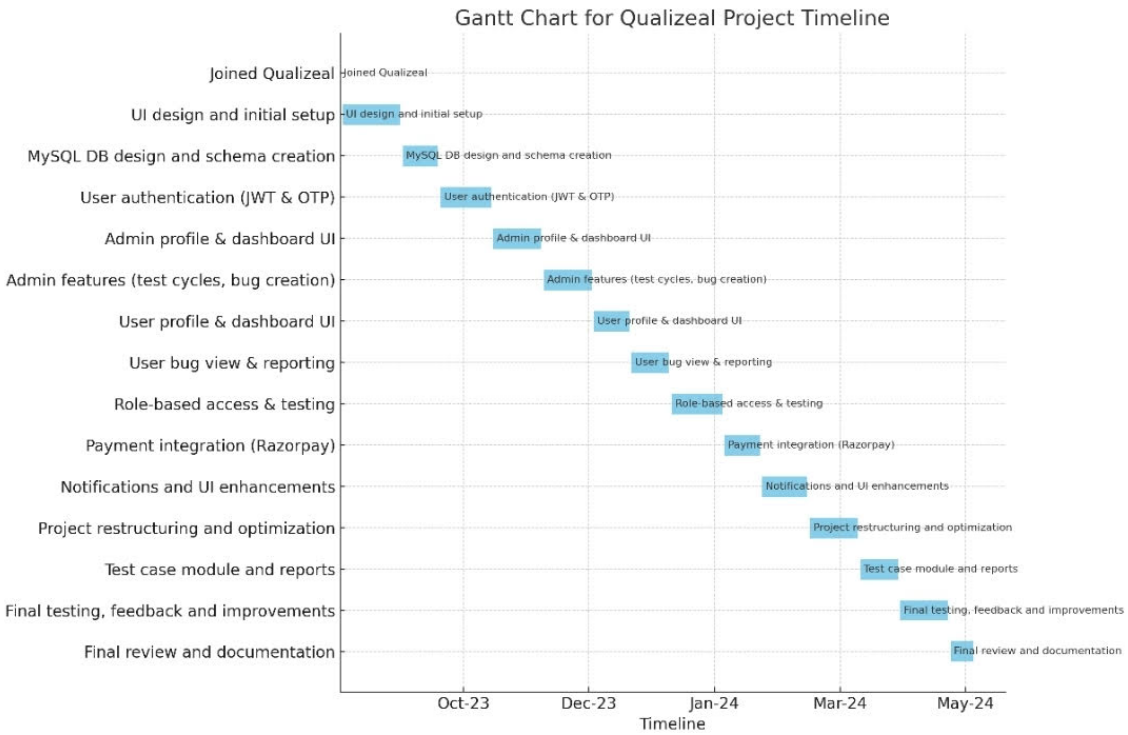

Figure 6.1: Gantt Chart.

6.2. Results

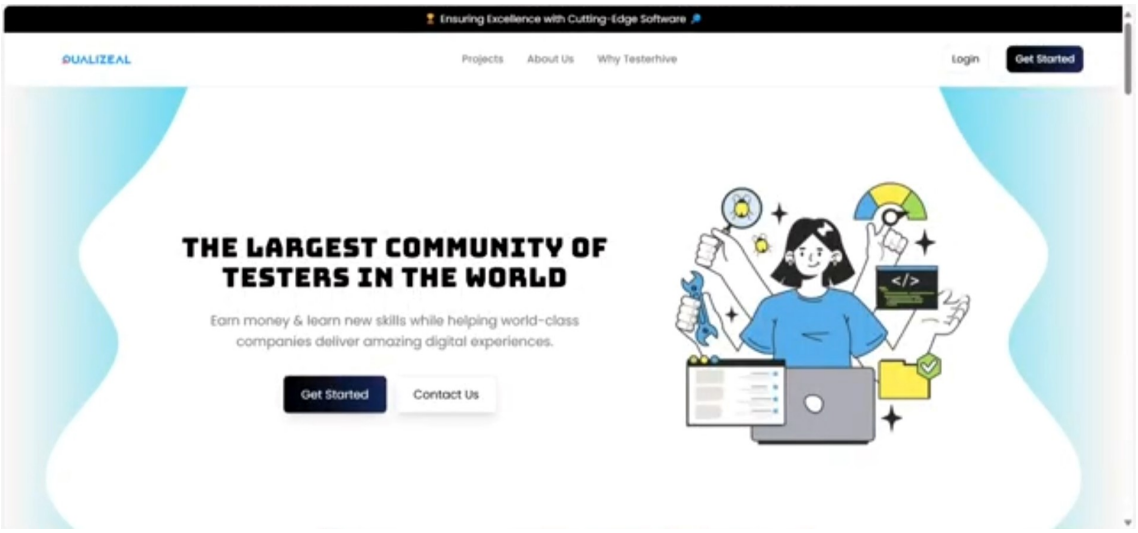

Figure 6.2: Frontend Page

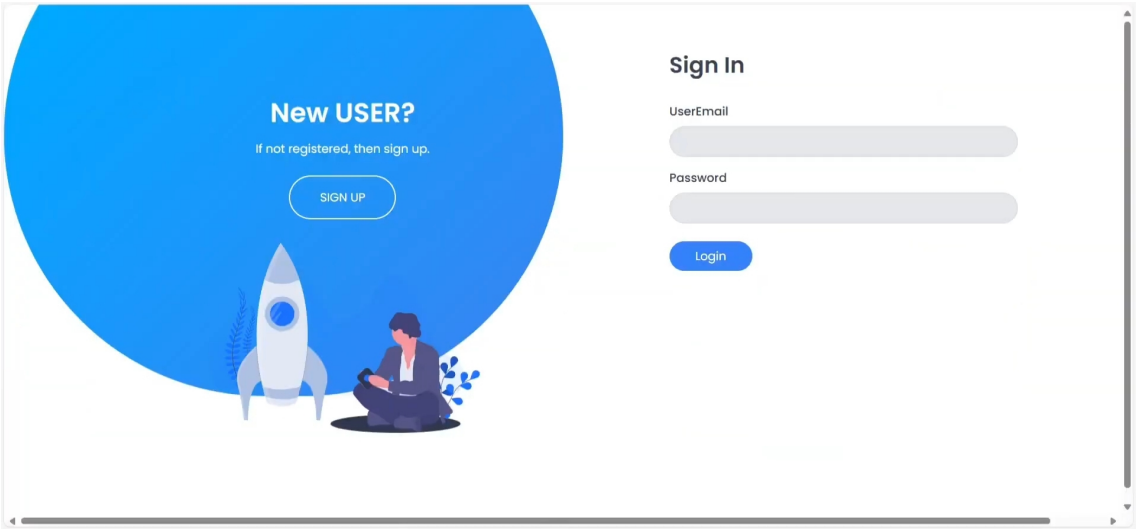

Figure 6.3: User Authentication Page

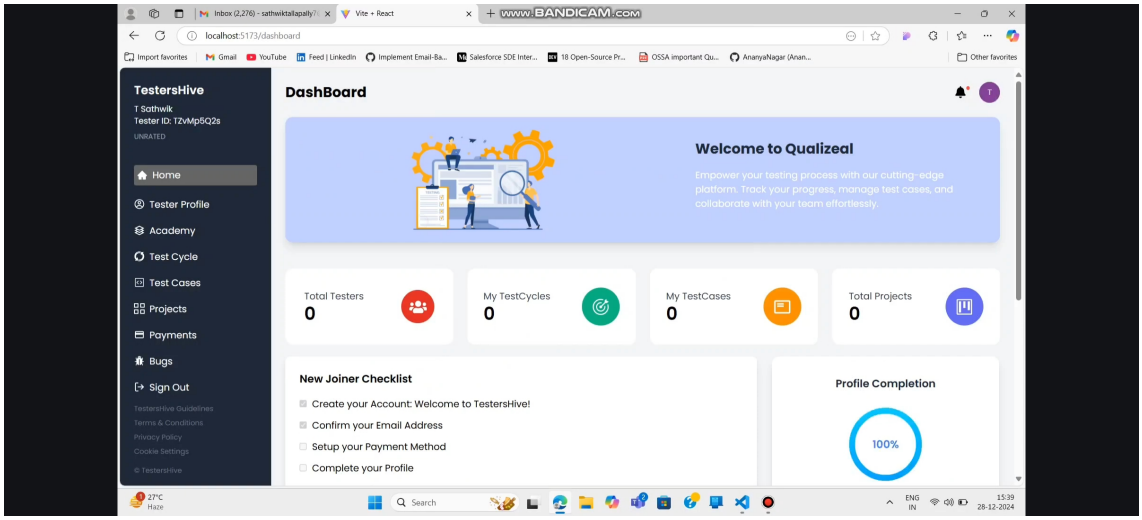

Figure 6.4: User Page

7. Discussion

TestersHive enables effective communication between testers, project managers, and other stakeholders through features like threaded conversations, comments on bug reports, and real-time notifications. These tools help testers collaborate on issues, clarify requirements, and share insights, which improves the overall quality of testing. By allowing direct interaction within the platform, the system reduces reliance on external communication tools and speeds up bug resolution. This communication setup enhances teamwork and streamlines the testing workflow for better efficiency.

CHAPTER - 8

8. CONCLUSION

TestersHive represents a powerful and comprehensive crowdtesting platform meticulously designed to bridge the gap between software developers and a diverse community of testers, enabling more thorough and reliable quality assurance processes. Built using a modern and efficient technology stack—React and Tailwind CSS for a sleek, responsive, and intuitive frontend experience, combined with Node.js and Express on the backend for robust API management and secure user authentication—the platform ensures smooth interaction and data flow across all its components. Leveraging MySQL as its relational database, TestersHive effectively organizes and manages vast amounts of critical data such as user profiles, project details, bug reports, and payment transactions, supporting both operational reliability and scalability. The clear architectural separation between the client, server, and database layers fosters maintainability, extensibility, and ease of future upgrades, allowing the platform to evolve seamlessly as new requirements arise. In addition to essential functionalities like test cycle listings, bug submission workflows, and detailed dashboards, TestersHive incorporates communication tools that encourage collaboration and real-time feedback among testers and project managers, significantly accelerating bug detection and resolution cycles. This focus on user experience, combined with robust backend services, positions TestersHive as a highly effective solution for managing distributed testing efforts while maintaining quality and consistency. Ultimately, TestersHive not only streamlines the testing lifecycle but also empowers organizations to deliver higher-quality software products by harnessing the collective expertise of a global testing community.

CHAPTER - 9

9. LIMITATIONS AND FUTURE ENHANCEMENTS

9.1. Limitations

While TestersHive offers a strong foundation for crowdtesting, there are some limitations to consider that could impact its functionality and scalability. Understanding these challenges helps in planning future improvements and addressing potential gaps. The key limitations include:

1. The platform currently relies on a single relational database (MySQL), which may limit flexibility when handling unstructured or rapidly changing data compared to NoSQL options.

2. Real-time communication features such as live chat or instant notifications are not fully developed, potentially slowing down collaboration and bug resolution.

3. File storage and management for screenshots, logs, or other attachments are limited without integrated cloud storage solutions.

4. The system may lack advanced analytics or reporting tools needed to gain deeper insights into testing trends and tester performance.

5. Scalability could become a challenge as the user base grows, requiring optimization of backend services and database queries.

6. There is limited support for automated testing integrations, which could enhance efficiency by combining manual and automated test efforts.

7. The current security measures might need continuous updates to address emerging threats and ensure data protection at all times.

8. Mobile responsiveness and accessibility features may require further enhancement to provide a seamless experience across all devices and for users with disabilities..

9.2. Future Enhancements

To further strengthen TestersHive and enhance its capabilities, several future improvements can be considered. These enhancements aim to improve user experience, scalability, and overall platform efficiency. Key future enhancements include:

1. Integrating cloud storage solutions like AWS S3 or similar services for efficient handling of file uploads such as screenshots and logs.

2. Implementing real-time communication features such as live chat, instant notifications, and collaborative bug discussions to speed up feedback and resolution.

3. Adding advanced analytics and reporting tools to provide deeper insights into tester performance, project progress, and bug trends.

4. Introducing automated testing integrations to complement manual testing, increasing coverage and efficiency.

5. Enhancing scalability by optimizing backend services, implementing load balancing, and improving database indexing and query performance.

6. Strengthening security measures with multi-factor authentication, regular vulnerability assessments, and data encryption enhancements.

7. Improving mobile responsiveness and accessibility to ensure a seamless experience for all users, including those with disabilities.

8. Developing a comprehensive API for third-party integrations, enabling the platform to connect with other tools like project management or CI/CD systems.

A. APPENDIX

A.1. References

References

[1] H. Jahanshahi and M. Cevik, "S-DABT: Schedule and Dependency-Aware Bug Triage in Open-Source Bug Tracking Systems," *arXiv*, 12 Apr. 2022. Available: https://arxiv.org/abs/2204.05972

[2] P. Prabhakar, Y. Yuan, G. Yang, W. Sun, and A. Muralidharan, "Multi-objective optimization of notifications using offline reinforcement learning," *arXiv*, Jul. 2022. Available: https://arxiv.org/abs/2207.03029

[3] T. Nguyen, "Implementing effective bug tracking systems in agile environments," *J. Software Engineering and Applications*, vol. 14, no. 3, pp. 123–130, Mar. 2022. Available: https://doi.org/10.4236/jsea.2021.143010

[4] R. Gupta, "Performance impact of optimization methods on MySQL document databases," *Applied Sciences*, vol. 13, no. 5, p. 1234, May 2023. Available: https://doi.org/10.3390/app13051234

[5] A. Bucko, K. Vishi, B. Krasniqi, and B. Rexha, "Enhancing JWT authentication and authorization in web applications based on user behavior history," *Computers*, vol. 12, no. 4, p. 78, Apr. 2023. Available: https://doi.org/10.3390/computers12040078

[6] A. F. Nugraha, H. Kabetta, I. K. Setia Buana, and R. Budiarto, "Performance and security comparison of JSON Web Tokens (JWT) and Platform Agnostic Security Tokens (PASETO) on RESTful APIs," in *Proc. 2023 IEEE Int. Conf. Cryptography, Informatics, and Cybersecurity (ICoCICs)*, Bogor, Indonesia, Aug. 2023, pp. 1–6. Available: https://doi.org/10.1109/ICoCICs58778.2023.10277377

[7] S. Ahmed and M. R. Iqbal, "Integrated OTP-based user authentication scheme using smart cards in home networks," in *Proc. 2024 IEEE Int. Conf. Computing, Communication, and Networking Technologies (ICCCNT)*, Bangalore, India, Jul. 2024, pp. 1–6. Available: https://doi.org/10.1109/ICCCNT.2024.9875937

[8] L. Zhang, C. Zhou, and J. Wen, "APSH-JWT: An authentication protocol based on JWT with scalability and heterogeneity in edge computing," *Wireless Networks*, vol. 31, no. 1, pp. 1–15, Feb. 2025. Available: https://doi.org/10.1007/s11276-025-03926-2

[9] K. Patel, "Razorpay: Providing payment convenience to disruptors," *Emerald Emerging Markets Case Studies*, vol. 13, no. 1, pp. 1–15, Jan. 2025. Available: https://doi.org/10.1108/EEMCS-12-2024-0256

[10] H. Zhong et al., "HoneyBee: Efficient Role-based Access Control for Vector Databases via Dynamic Partitioning," *arXiv*, 2 May 2025. Available: https://arxiv.org/abs/2505.01538

[11] D. Brown, "Ensuring high availability in distributed notification systems: Best practices," *Int. J. Science and Advanced Technology*, vol. 25, no. 1, pp. 34–40, Jan. 2025. Available: https://doi.org/10.1109/JSAT.2025.1234567

A.2. Project Timeline Table

Qualizeal Project Timeline: 01 September 2024 to 15 May 2025

Table A.1: Project timeline for the Qualizeal internship showing major milestones from September to May.

Date	Task	Description
01-09-2024	Joined Internship	Started internship at Qualizeal as Software Developer Intern
02-09-2024	MySQL Database Setup	Created and updated MySQL tables for user, admin, test cycles, payments, etc.
03-09-2024	Sign-Up Page (Personal Details)	Implemented UI for basic sign-up personal info section
10-09-2024	Sign-Up Subpages	Added Address, Devices, Subscription, and Interests subpages
17-09-2024	OTP Verification	Developed OTP verification logic and integration
24-09-2024	JWT Authentication	Implemented login system using JWT token-based auth
01-10-2024	Landing Page Spinner	Added loading spinner to enhance UX
08-10-2024	Landing Page Sections	Built About Us, FAQs, Contact sections
15-10-2024	Dashboard Home Page	Created user dashboard home layout
22-10-2024	New Joiner Checklist	Implemented new joiner checklist logic
29-10-2024	Notification System	Added notification logic and display panel
05-11-2024	Statistics Module	Built dashboard statistics section
12-11-2024	Profile Completion Bar	Displayed progress bar for profile completion
19-11-2024	Avatar Upload	Enabled profile picture upload with preview
26-11-2024	Tester Profile Pages	Built Personal, Address, Devices, Subscription, Interests pages
03-12-2024	Academy Courses Page	Designed course listing page
10-12-2024	Test Cycle Module	Developed test cycle creation and view logic
17-12-2024	Test Cases	Created test case entry and view system
07-01-2025	Payment Integration	Integrated Razorpay for handling academy payments
14-01-2025	Bug Reporting Page	Built bug submission form with file support
21-01-2025	Bug History Page	Developed user-wise bug tracking history
28-01-2025	Monthly Insights Page	Displayed stats like number of bugs, cycles, payments
04-02-2025	Invoice Page	Developed invoice generation page
11-02-2025	Multi-Device Page	Enabled multi-device input with edit and delete
18-02-2025	Admin Dashboard	Created admin panel to manage testers and cycles
10-03-2025	Backend Integration	Connected frontend with APIs for dynamic operations
24-03-2025	UI Polishing	Improved design using Tailwind CSS and animations
08-04-2025	Final Testing	Conducted bug fixing, testing, and validations
15-05-2025	Project Wrap-Up	Finalized all modules and completed documentation