TestQuest: An Open Bug Reporting Platform with Developer-Driven Rewards

Joker P. Panuelos
Bachelor of Science in Information Technology

Jacob Andrew V. Masip
Bachelor of Science in Information Technology

Senior Thesis submitted to the faculty of the
Department of Computer Science
College of Computer Studies, Ateneo de Naga University
in partial fulfillment of the requirements for their respective
Bachelor of Science degrees

Project Advisor: Kurt Gilbert D. Tapel
Adrian Leo T. Pajarillo
Kevin G. Vega
Paul Angelo S. Silvestre

14 June 2025
Naga City, Philippines

Keywords: Testers, CST, Motivation

The Senior Project entitled

TestQuest: An Open Bug Reporting Platform with Developer-Driven Rewards

developed by

Joker P. Panuelos

Bachelor of Science in Information Technology

Jacob Andrew V. Masip

Bachelor of Science in Information Technology

and submitted in partial fulfillment of the requirements of their respective Bachelor of Science degrees has been rigorously examined and recommended for approval and acceptance.

Adrian Leo T. Pajarillo
Panel Member
Date signed: _____

Kevin G. Vega
Panel Member
Date signed: _____

Paul Angelo S. Silvestre
Panel Member
Date signed: _____

Kurt Gilbert D. Tapel
Project Advisor
Date signed: _____

The Senior Project entitled

TestQuest: An Open Bug Reporting Platform with Developer-Driven Rewards

developed by

Joker P. Panuelos

Bachelor of Science in Information Technology

Jacob Andrew V. Masip

Bachelor of Science in Information Technology

and submitted in partial fulfillment of the requirements of their respective Bachelor of Science degrees is hereby approved and accepted by the Department of Computer Science, College of Computer Studies, Ateneo de Naga University.

Lowie Vincent S. Bisana
Chair, Department of Computer Science
Date signed: _____

Joshua C. Martinez
Dean, College of Computer Studies
Date signed: _____

Declaration of Original Work

We declare that the Senior Project entitled

TestQuest: An Open Bug Reporting Platform with Developer-Driven Rewards

which we submitted to the faculty of the

Department of Computer Science, Ateneo de Naga University

is our own work. To the best of our knowledge, it does not contain materials published or written by another person, except where due citation and acknowledgement is made in our senior project documentation. The contributions of other people whom we worked with to complete this senior project are explicitly cited and acknowledged in our senior project documentation.

We also declare that the intellectual content of this senior project is the product of our own work. We conceptualized, designed, encoded, and debugged the source code of the core programs in our senior project. Also duly acknowledged are the assistance of others in minor details of editing and reproduction of the documentation.

In our honor, we declare that we did not pass off as our own the work done by another person. We are the only persons who encoded the source code of our software. We understand that we may get a failing mark if the source code of our program is in fact the work of another person.

Joker P. Panuelos
4 - Bachelor of Science in Information Technology

Jacob Andrew V. Masip
4 - Bachelor of Science in Information Technology

This declaration is witnessed by:

Kurt Gilbert D. Tapel
Project Advisor

TestQuest: An Open Bug Reporting Platform with Developer-Driven Rewards

by

Joker P. Panuelos and Jacob Andrew V. Masip

Project Advisor: Kurt Gilbert D. Tapel

Department of Computer Science

ABSTRACT

Accomplishing effective software testing continues to be a problem in areas like Tanzania, even with the ever-evolving software development technologies. The software testers' scarcity problem, absence of adequate training opportunities, as well as little to no encouragement for novice testers, have all been associated with poor software reliability and project timelines. This research suggests creating a CST (Crowdsourced Software Testing) platform that allows developers to submit their software for testing, while novice testers are able to find bugs and be monetarily compensated. In addition to CST, the platform utilizes the features of game design like points, badges, or even leaderboards to increase participation. The study aims to achieve the outlined goals utilizing the Agile Scrum, accepting iterative stakeholder feedback in designated sprint cycles as the basis of continuous enhancement. This methodology incorporates user interface/user experience design methodologies including prototyping, feature implementation, usability tests with evaluations using the survey, observation, and interview techniques. The CST platform is designed to enable various types of testing including usability, performance, and compatibility testing along with mechanisms for bug reporting, test case description, and automation scripts submission.

TABLE OF CONTENTS

Chapter 1

Introduction

In order to guarantee the dependability and quality of software products, software testing is essential. However, putting into practice efficient software testing procedures presents a number of difficulties for software development companies, like those in Tanzania. These difficulties may result in the supply of poor software, higher expenses, and project delays, among other serious repercussions [1]. One emerging solution to these challenges is Crowdsourced Software Testing (CST). The term 'crowdsourced software testing' describes the application of crowdsourcing methods in the software testing domain. It's a new field, with usage that has increased dramatically during the past ten years. It has been utilized in numerous software testing tasks, including functional testing, usability testing, performance testing, and compatibility testing [2]. Furthermore, they can collect feedback which truly reflects usage as end users are the best for usability testing. Usability testing crowdsourcing benefits greatly from the user's authentic experiences and specific knowledge related to the area . For instance, feedback from a teacher of the learning application will be more useful than from a tester with no experience or background in education [1]. In addition, crowdsourcing can also represent the training ground for those who want to become testers. In fact, novices participating in actual testing projects acquire reporting skills and gain familiarity with various devices, environments, and industry-specific testing. Creating platforms with elements of game design, feedback systems, and ranking features can motivate people to get involved and help them chart their development throughout the platform. This means that rather than only increasing software reliability and accuracy, these platforms are preparing a new generation of experts in software quality assurance.

Gamification has been used extensively to promote participation, develop motivation, and maintain long term engagement through the use of points, badges, and other game-like elements. Therefore, CST platforms that use these strategies not only solve the issue of tester motivation, but also function as productive learning environments for prospective testers [3].

1.1 Purpose and Description

This capstone project's purpose is to develop TestQuest, a crowdsourced software testing platform that integrates developers with a pool of testers which includes students, freelance QA professionals, and those in the early stages of their careers. Through the platform, developers are able to upload software applications to be tested, define evaluation metrics, and compensate testers through a points-based system that translate into monetary payment. Testers, in turn, engage in a practical learning environment where they can report bugs, enhance their skills, build reputation, and earn rewards based on the accuracy and severity of their reports. To keep the testers engaged, badges, leaderboards, as well as reputation tracking are incorporated as gamified features on the platform. Moderators assist in resolving disputes between users and in validating bug reports to maintain fairness in the platform. Global transactions will be simulated within a PayPal Sandbox environment, automating the distribution of rewards while preserving a seamless financial system devoid of real monetary funds. Moreover, TestQuest seeks to enhance the software testing experience by building a collaborative ecosystem where testers and developers engage openly, learn continuously, and derive value together through the testing processes and payments.

1.2 Objectives

This study's main goal is to develop a crowdsourced testing platform that enables developers to post systems they want tested and allows testers to identify faults within them. The platform will incorporate a reward system, where developers can award credit points to testers who successfully find these faults. These points can then be exchanged for actual money. This approach not only helps improve the quality of the submitted software but also provides aspiring testers with an opportunity to develop their skills in an environment that rewards their contributions.

Main Objectives:

- To develop a collaborative testing platform

  This objective is centered on creating a comprehensive web-based platform that enables collaborative work among software testers in real and simulated testing environments, which is also applicable for aspiring testers who wish to hone their skills early on.

- To enhance the quality of software testing by creating a system that encourages wide participation, feedback sharing, and collaboration.

  This objective seeks to enhance the results of software testing by engaging a wide array of users through an crowdsourced collaboration platform. The system fosters and supports communication and critique between testers and developers, thus contributing diverse perspectives which bolster comprehensive bug detection.

  This objective focuses on integrating gamified elements like points, badges, leaderboards, and challenges into the system. These components aim to enhance interactivity within learning and practicing software testing, stimulating sustained engagement and improving the learning experience.

Specific Objectives:

- To create an environment to enable aspiring testers practice their skills

  This objective concentrates on providing a practical workspace where students and novice testers can practice different types of testing activities. The platform will cover all types of software testing, including functional, usability, and performance testing, and provide feedback so that learners can identify gaps, enhance their skills, and transition from novice to industry-ready software testers.

- To implement a feedback system that allows testers to give feedback on the developers posts. This objective will help the testers know more about the posted software before testing them, and will also give the testers a chance to tell about the other testers how the developers are handling their testers achievements. This will discourage developers from exploiting the testers, as they will now have a reputation to uphold, in order to bring more testers.

- To develop methods within the platform that increase user motivation and engagement through gamification

This objective focuses on integrating gamified elements like points, badges, leaderboards, and challenges into the system. These components aim to enhance interactivity within learning and practicing software testing, stimulating sustained engagement and improving the learning experience.

1.3 Statement of the Problem

Putting into practice efficient software testing procedures presents a number of difficulties for software development companies, like those in Tanzania. These difficulties may result in the supply of poor software, higher expenses, and project delays, among other serious repercussions [1]. Moreover, Software testing is undervalued by students in software and computer engineering, according to studies that were done in Canada, China, India, and Malaysia. Some of the findings also suggests the need to change how testing careers are viewed by students. Cultural and social factors influence perception, so more replications are needed to generalize the results [4]. At the same time, the software industry struggles because of an insufficient number of skilled and motivated software testers. Although a few universities have begun teaching software testing, students still perceive testing as an unappealing profession. There is still a shortage of platforms and environments that allow aspiring testers to collaboratively perfect their craft in a motivating and engaging way [5].

1.4 Needs Assessment

Currently, the software industry is experiencing a considerable shortage of qualified quality assurance (QA) professionals and software testers. Recent industry reports indicate that organizations are unable to fill testing roles due imbalance in supply and demand [6]. The taleng gap is more noticeable at the entry level, where QA teams are continuously struggling to attract and keep skilled testers [7]. Such inadequacies dramatically impede the assurance of software quality, prolong the development phase, and, to a greater extent, amplify the possibility of unrefined products being released to end users. In the context of software testing education, the traditional approaches have had limitations on motivating students to enter the testing field. In this context, gamification has appeared as a potential solution. Empirical studies show that the incorporation of gamified elements in the curriculum of testing courses enhances the students' level of engagement and performance relative

to the courses with no gamified elements [8]. Moreover, reviews at the tertiary level on the use of gamification in software engineering education also show that testing and software quality are some of the domains most frequently impacted by gamified instructional design. In this context, competitive and collaborative frameworks are used as some of the most frequent motivational strategies [8]. This evidence highlights the potential of gamification in alleviating some of the motivational issues that are part of the shortage of testers available. Simultaneously, the industry demand revolves around the adoption of more efficient and cost-effective testing solutions that can scale. For small software companies and startups, the financial burden of holding large inventories of devices and having full-time in-house QA teams is increasingly unsustainable. As a result, companies have been selling crowdtesting as a cost-effective solution, offering a globally distributed workforce that can conduct testing in different environments without the architecture costs of pre-crowdtesting solutions. These platforms offer the ability to scale testing capabilities based on the specifications and size of a project, while also providing extensive platform and configuration coverage due to the flexibility afforded by crowdsourced participation anywhere in the world. The conclusions drawn indicate a distinct set of interlinked needs in the software testing market and its educational ecosystem: providing globally accessible software testing, advancing educational practices in engagement and motivation, and building comprehensive cost-effective testing solutions that incorporate educational infrastructures. The TestQuest platform is conceived to meet the aforementioned needs which include combining a gamified educational approach and crowdsourced testing to integrate the aspirational software testers, the software developers, and the educators into one cohesive solution.

1.5 Significance of the Study

CST helps in creating high quality software by bringing in thousands of people to test under a wide range of conditions. This differs from traditional software testing which relies on a handful of testers from the software development company, but it complements it [6]. CST has proven effective across several types of testing including but not limited to usability testing , functional, performance and compatibility testing [9,10]. Complete compatibility testing in-house is often difficult because of the large number of devices and operating systems, and the limited time and budget available. Once devices age, the cost of acquiring them solely for testing purposes greatly outweighs their value [11]. A promising solution to increase tester motivation is gamification, which involves integrating game

elements into non-game contexts. Gamification techniques offer an opportunity to engage students in testing, even when they perceive it as boring or tedious. By incorporating elements commonly found in games, gamification has been successfully applied to various areas of software engineering, including software construction, requirements engineering, and software process management [12,13]. In our platform's case, the researchers are using a reward system, which will reward testers who have successfully found faults in the systems that the developers posted in the platform. This will enhance the testers motivation, especially those who are aspiring

1.6 Scope and Limitations

1.6.1 Scope

Target Users

For the purpose of this capstone project, platform testing and evaluation will be limited to fellow Computer Studies capstone takers.The platform is tailored to accommodate three primary user roles:

- Developers who will post software projects for testing. They can also establish specific evaluation criteria for measuring bugs that are reported, as well as set a reward structure, assigning credit points to testers according to the severity and quality of reports submitted.
- Testers who will test available software projects, submit comprehensive bug reports, and earn rewards (which can be monetized) based on validated contributions. Testers may include aspiring QA professionals, students or freelance testers who seek to enhance their skills and professional standing.' elaborates more on the developers
- Moderators act as a medium to resolve any misunderstandings between the developers and testers, regarding payments, or the validity of the submitted bug reports. In the earlier stages of the platform, they may be trusted QA professionals, or students. But once the testers has established their reputation and reached a certain criteria, they may be selected and invited to become a moderator.

Platform Accessibility

TestQuest is a web-based platform designed for crowdsourced software testing. Users will be able to access the system via desktops, laptops, or smartphones. Important functionalities including

extensive bug report submission (screenshot attachment, step documentation and files).

Testing Capabilities

The platform will allow different levels of software testing to accommodate different types of users. Functional testing will enable testers to check whether the software features operate correctly, and usability testing will focus on assessing ease of use and interface ergonomics. Compatibility testing across multiple browsers, devices, and operating systems, will ensure functionality. All discovered bugs will be classified into severity categories: Critical bugs that hinder essential operations, Minor bugs that do not impact primary functions, and Major bugs, such as visual or user experience problems framed as design issues.

Gamification and Engagement

To boost user motivation and engagement, TestQuest will integrate gamification features throughout the platform. A point system will reward testers according to the severity and validity of the bug that they reported, while accomplishment badges will recognize testers for achieving specific milestones, like finding their first bugs or maintaining high accuracy rates. Public leaderboards will feature the best testers whose performance will motivate others to learn and compete in healthy ways. A reputation system will monitor every tester's credibility based on reported accuracy, and trust from developers through their feedbacks, which assists in building confidence within the community.

PayPal Sandbox Transactions for Testing Purposes

The platform will implement global accessibility by incorporating secure payment processing through PayPal. Automated tracking systems will be employed to maintain complete transaction records of all earnings, point conversions, and payments. Automated reward distribution will instantly allocate points when developers confirm submitted bug reports, while flexible payout options enable testers to convert earned points into actual currency at their convenience.

1.6.2 Limitations

Identity and Bug Report Validation Limitation

The system's capabilities for international payments will be enabled through reliable PayPal Sand-

box Facilitated transactions. With this configuration, developers can perform validation tests of the platform's payment processing and reward systems without working with physical funds. Full logs of all simulated transactions such as earnings, point conversions, payouts and others will be recorded by automated tracking systems. After all verified bug reports are processed, rewards in the form of currency are distributed automatically on test account which enables testers to seamlessly go through the process of converting points into virtual currency in a controlled environment.

Platform Management Limitations

Human moderators will have to manually resolve conflicts between developers and testers regarding points or bug differences. Although this method provides an unbiased way to resolve conflicts, it might slow them down when traffic is heavy. The platform will only offer basic reporting features that require manual import into workflows, and it will not integrate with professional development tools such as GitHub or JIRA. The framework is designed for solo practitioners and micro teams, as opposed to large corporations with complex hierarchies.

Security and Data Protection

While the platform will employ basic password safeguards as well as HTTPS encryption, more advanced features such as multi-factor authentication, biometric login options, or military-grade encryption will not be implemented. Industry standards for data protection will be observed with routine backups and secure storage of sensitive information, but features such as comprehensive audit logs or sophisticated data anonymization techniques will not be offered.

Excluded Features

A number of key features may be removed from the scope of this project in order to maintain focus and maintain feasibility. TestQuest will not include automated testing systems, algorithms for test script creation, automated educational testing AI applications, academic professional qualifications, integration of enterprise software with legacy development ecosystems, diagnostic or evaluative bug analysis employing artificial intelligence technologies, proposed automated unit test applications for mobile platforms on iOS and Android, interface design beyond English language for users in other countries, or higher-level business analytics with embedded reporting modules and customized dashboards. By removing these features the project is able to focus on the comprehensive crowdsourced

testing functionality while still being deliverable within existing resource and schedule constraints.

Chapter 2

Review of Related Systems and Literature

This chapter offers a close look at studies and existing platforms that inform the creation and rollout of a crowdsourced bug-hunt site. Coverage is grouped into three key areas: software testing routines and techniques , tester motivation and incentive models , and adding games and crowds to keep quality high [1–3].

2.1 Software Testing Frameworks

Software testing plays an important role in the development process of software applications. One of the foundational sources makes a note on the need for systematic and controlled test design at every level, beginning with unit tests up to full system evaluation to confirm that the software works correctly and all features are implemented [14]. Other studies focus on different areas like automation, coverage within agile frameworks, as well as seamless integration of testing into agile processes. Other adaptive frameworks that utilize a combination of automated and manual techniques are also supported [15]. Issues concerning motivation, quality control, social dynamics, along with project scalability in relation to tester involvement reveal complexity from both technical and social angles in crowdsourced software testing [2]. Furthermore, context-driven strategies were also adviced by some studies when compared to other industrial methods since they will always be more successful

when aligned with organizational objectives rather than using one-size-fits-all approaches [5].

2.2 Motivation and Incentives for Testers

Active participation in crowdsourced testing platforms requires efficient engagement strategies and ongoing maintenance of testers' involvement. As noted in motivation theory, performance peaks when recognition is commensurate with milestones achieved and the amount of work invested [16]. This also applies to woven functionality, in a distributed testing setting, it was found that reward mechanisms, clarity of the given instructions, their defined role within the work structure as well as appreciation of efforts play a crucial role in influencing tester's participation and the quality of their output [4, 9]. Further supporting this point is a gamification research showing that incorporating game mechanisms like points, leaderboards, and achievement badges improves user engagement [17]. Furthermore, gamification has also been described as a means to enhance organizational productivity through improvement in motivation by aligning performance against a set of key results [1, 18]. The basic premise upon which such platforms operate, is that appropriately structured rewards correlate positively with activity levels and outcomes delivered.

2.3 Effects of Gamification Elements to Apps

Gamification focuses on the incorporation of game-like features into non-gaming scenarios to enhance user interest , motivation, and performance [19]. It has been employed with varying degrees of success within educational software for purposes like improving learning in software engineering and testing. Applications typically use points, badges, and leaderboards to incentivize participation and shape behavior [20]. Points give instant feedback while indicating progress towards a goal, enhancing measurable actions. Badges reward users visually offering recognition for achievement through goals reached. Users gain a competitive edge through competing which motivates improvement when ranking occurs as seen in leaderboards [20]. Beyond advanced gaming features, the MDA (Mechanics-Dynamics-Aesthetics) framework can also be used to explore gamification. This framework combines structural design (mechanics), user interaction (dynamics), and emotional engagement (aesthetics) in regard to the experience being gamified. With this approach , we can ensure that gamification serves a purpose, but it is also done in an enjoyable and immersive manner—functional as well

as enjoyable. Other elements such as integrated narratives through visual designing and rewards contribute towards appreciation points from users such as social validation or pleasure [21]. Engaging students within Software testing courses through gamified approaches showed students were able to grasp the more complex concepts revolving around testing with ease. However most of these attempts are under-researched in the long term; limited scope focused on short-term results have crow-found undone work resulting in an incomplete picture [20]. As highlighted by various studies, there are benefits to applying gamification; however, some of these studies may have other shortcomings like small-scale implementations or absence of comparison with non-gamified counterparts. When implemented carefully and sustained over longer periods of time, there is strong evidence suggesting that user experience and performance in highly diverse contexts including educational settings or brand facing consumer applications can be enhanced considerably using gamification-describedby frameworks [21, 22]].

2.4 Crowdsourcing and Platform Governance

The benefits and drawbacks of utilizing crowdsourcing for managing open collaboration in software quality assurance are evident. Some major concerns were raised by research, including controlling the range of contributor skills, avoiding submit multiple reports for identical bugs, and compensating test service providers fairly [3, 12]. Integrating partially automated workflows with human supervision improves efficient task distribution. According to some researchers, automation via AI-driven triage systems can streamline assignment through skill-set schedule-output mapping [13]. In violation of overwhelmingly important social norms, platforms narrowed access and excluded whole segments of their user bases with troubling opaqueness regarding how funds held by developers and testers got disbursed. Centralized reward allocation models often disregard important criteria like fairness, responsibility, possible trust violations or abuse, rampant inequality among contributors, or even authoritarian centralized control structures without real checks [10, 11].

2.5 Related Systems

2.5.1 HackerOne

HackerOne, one of the leading platforms for crowdsourced security testing, makes use of vulnerability disclosure and bug bounty schemes. The platform connects businesses with ethical hackers around the world to find fixable flaws in systems. HackerOne is known for its comprehensive approach to vulnerability management. It offers an all-in-one workflow tool that integrates all tasks from triage submission to activity closing, thus enabling firms to handle program resources, disclosures, and centralized bounties payment processing effortlessly [23]. Besides providing a systematic solution, the platform features AI in the form of Hai which automates ghastly tasks like crafting detailed summaries of noted vulnerabilities and summarizing workflows which are monotonous on their own [24]. Such automation improves efficiency as well as accuracy on these workflows. Moreover, the site features a vibrant and engaged international community with more than two million registered security researchers. It provides both private and public programs which helps organizations customize the scope and visibility of their engagements. With regard to ecosystem transparency and benchmarking, HackerOne supports peer leaderboards as well as many analytic performance metrics. They support organization's self-assessment relative to others in the platform's ecosystem [23]. HackerOne has a special team for triage services that focuses on critical impactful vulnerabilities. This team receives close to 16,000 submissions a month, effectively sifting through lesser problems to attend to the most important ones. This enables the platform to reach a milestone average first response time of 11 hours [25, 26], highlighting its responsive and proactive approach in managing vulnerabilities. Although HackerOne showcases exceptional scalability of AI assist features and support, it remains focused on cybersecurity in particular. The absence of beginner-friendly frameworks as well as gamified features is limiting its accessibility to aspiring or new testers. While this focus can be considered beneficial in specialized environments, it may limit the effectiveness of more generalized testing ecosystems that seek to engage a broader range of contributors.

2.5.2 Bugcrowd

Bugcrowd is a platform for security testing that allows an organization to run managed bug bounty programs, coordinated vulnerability disclosures, penetration testing-as-a-service and attack surface management activities . The platform offers advanced researcher-to-project matching and submis-

sion validation tools to streamline the operations of comprehensive expert-driven cyber security on a large scale [27]. One of Bugcrowd's key features is its proprietary CrowdMatch AI, which automatically matches security researchers with relevant programs. In order to create a comprehensive security knowledge graph, the system uses a wide range of selection criteria, like accuracy, researcher's background, prior performance metrics that are recorded during submissions [28]. The security knowledge graph was developed over a decade long period of gathering and refining data. The security knowledge graph was developed over a decade long period of gathering and refining data. It integrates numerous information, such as digital assets, prior test results, security holes, and researcher profiles. This graph fuels triage analytics as well as the CrowdMatch system to yield sustained insight-driven performance analysis that improves strategic decisions [29]. Moreover, Bugcrowd facilitates the integration of security into the DevSecOps workflows within development processes. It offers APIs and connectors for popular development tools, thus helping companies to seamlessly execute security assessments during their software development lifecycle [30]. Although Bugcrowd's infrastructure and automation systems showcase powerful frameworks concerning scalable, expert-level cybersecurity testing, they are still oriented towards professional users [27]. The absence of gamification elements for beginners highlights the need for mechanisms that would widen participation, boost sustained engagement, and assist lay persons in diverse testing situations not limited to specific purposes.

2.5.3 Open Bug Bounty

Open Bug Bounty is a nonprofit and crowdsourced platform that focuses on coordinated vulnerability disclosure. The platform on its part has been operational since June, 2014 and helps ethical security researchers to report web exploits using non-intrusive methods which is inline with ISO29147 standards for responsible disclosure practices [31]. Open Bug Bounty is distinctive due to its non-commercial, altruism-focused framework. Unlike most platforms, there are no mandatory bounties or fees, and participation is entirely voluntary. Researchers share their findings freely and in return for their goodwill, receive acknowledgment. The researchers' contributions to the platform are validated through community-driven processes that do not charge for their services by monetizing the work of either researchers or website owners [31,32]. It contains flexible reporting options which allows a researcher to opt for either public or private disclosures. Moreover, participants may choose to display badges denoting their participation. However, these are optional and have no cash value [32]. In

order to preserve ethical standards, Open Bug Bounty has enforced a constrained scope of activity. Only low-level web vulnerabilities may be filed; both manual and automated intrusive scans, along with hacking techniques, are expressly prohibited. This limitation ensures that all testing performed via the platform is compliant with responsible disclosure and safe [33].

2.5.4 Test'Em All

Test'em All is a mobile application created by JustDice GmbH, marketed as a rewards system in which users can amass points redeemable for monetary compensation or gift cards. The app's availability on Android and iOS devices allows its users to complete numerous tasks such as playing mobile games, completing various offers and surveys, or even partaking in mystery shopping [34]. Users can accumulate points through engaging in a variety of activities which may include installing and interacting with advertised applications, reaching specific milestones in games, answering in surveys, or fulfilling specific shopping tasks. These points can be exchanged for monetary compensation using PayPal, or converted into gift cards from retail partners. Besides the primary earnings from tasks, the application offers additional rewards via referral bonuses as well as through designated advertising campaigns [35]. Nonetheless, there were concerns that have been raised about the payment experience on the platform. Some users have reported problems with payments being delayed or lower amount than expected, as well some inconsistencies with bonus rewards. There are further highlighted issues relating to customer service, particularly in relation to resolving payments and payout issues [35]. Although Test'em All offers an easy way to get into game testing and provides straightforward, task-based earnings for the more casual users, its effectiveness as a dependable platform is quite limited.

Related Systems	**Features**	**Limitations**	**TesQuest Advantage**
HackerOne	- Comprehensive vulnerability management with triage-to-remediation processes - Tools augmentation via AI - Over two million researchers worldwide - Analytics of peer transparency and benchmarks - Triage team specializing in over 16 thousand submissions per month	- Only as pertains to cybersecurity - Intended for expert participants - lacks beginner progression or gamification	- Incorporates points and badges, which are game mechanics. - Covers usability and performance testing in addition to security testing. - It does not require advanced experience and can be used by novices who are not experts.

Table 2.1 continued from previous page

Related Systems	**Features**	**Limitations**	**TesQuest Advantage**
Bugcrowd	- CrowdMatch AI enables skill-based pairing. - Knowledge Graph of cybersecurity built on more than a decade of collected data. - Engineered triage with recursive function validation. - Integrations through APIs.	- Only targets advanced ethical hackers. - No learning systems or gamified elements. - Not very interesting for novice users.	- Includes more comprehensive testing (functional, usability, performance). - Gamifies engagement by adding level-based incentive systems. - Aims to reach more people than just security testers.
Open Bug Bounty	- Free platform with altruistic and nonprofit motives. - Ability to select between public, private disclosure options. - Crowdsourced validation of non-malicious vulnerabilities. - Concentrates on lower risk web problems that are less intrusive. - Over a million coordinated fixes and participation from over twenty thousand contributors	- Neither payment nor rewards are offered. - Absence of DevSecOps or other tools related integration. - Non-intrusive testing only. - Lack of support for automated or severe high-priority bug types.	- Awards prizes using payment systems already in place. - Applies gamification to reports with throttled scoring based on severity. - Provides defined levels of bugs, reputations, and ranked submissions.
Test'em All	- Completing tasks in mobile games testing to earn money - Participation in surveys, recommendations, mystery shopping, and deals. - Reward redemption via gift cards or PayPal.	- Payout reliability and earnings are low. - Inadequate feedback on support - Unsuitable for game testing professional workflows	- Documents bug reports officially with images and tags indicating severity. - Combines updates with the ability to modify documents in real-time. - Focuses on projects as opposed to casual interaction with the application.

Chapter 3

Technical Framework

3.1 System Architecture

TestQuest is a platform built on a three-tier web application model that consists of the presentation layer (frontend), application layer (backend), and data layer. These layers are essential for maintaining scalability and sustainability in a crowdsourced testing platform, as well as for separating concerns.

Presentation Layer (Frontend):

- Dynamic interfaces developed through React.js.
- Responsive designs implemented with Bootstrap CSS framework.

Application Layer (Backend):

- Node.js runtime environment
- Express.js web application framework
- RESTful API for client-server communication

Data Layer:

- Relational data structured into mongodb database.
- Sequelize ORM used for executing database operations.
- Payment services rendered through PayPal API.

3.1.1 System Architecture Diagram

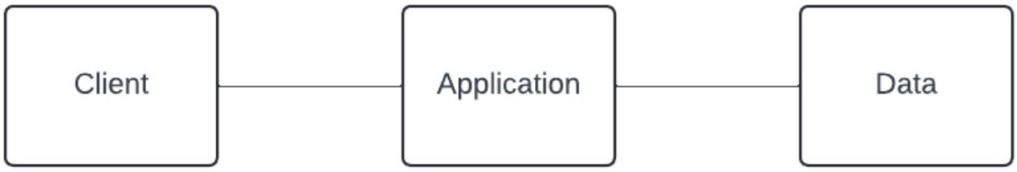


Figure 3.1: *System Architecture*

3.2 Technology Stack and Development Tools

3.2.1 Frontend Technologies

The user interface for platform is developed using React.js. The reason React.js was chosen is because it has a component-based structure which enables development of reusable UI components critical for the diverse parts of a testing platform like test submission forms, project cards, user profiles and testing dashboards. Equally important, the entire system built using react is very easy to debug and maintain, especially because of how controllable and predictable the declarative nature of react makes the codebase.The extensive support community that React boasts also comes in handy for development with its myriad resources and libraries.

Bootstrap is utilized as the CSS framework for the platform, enhancing it with responsive design features along with UI elements that are made ready to use. This selection helps the platform maintain a coherent visual design across devices and screen sizes, ranging from desktops for developers to multiple device types used by testers. As always, accessibility remains critical. With its grid system and responsive utilities, Bootstrap enables the consistent delivery of an optimal experience regardless of how users access the platform.

3.2.2 Backend Technologies and Database

The backend structure is organized with a Node.js framework which uses JavaScript as the server side runtime. This selection provides a cohesive development experience for the group going through the entire application stack while lowering context switching for developers, enabling shared code between frontend and backend parts of the system. The I/O model based on events in Node.js is appropriate for a Call Center workload with many simultaneous submissions of tests, file uploads, and payment processing from different users.

Express.js acts as a web application framework, as it serves up minimal yet flexible building blocks for the platform's API endpoints. Along with implementing production-ready applications, authentication, request validation and verification, error handling, and other essential cross-cutting concerns can be handled using middleware support offered by Express.js. Because the framework is lightweight, rapid development is supported while giving room for scaling the platform or adding features.

The platform uses MongoDB as its NoSQL database for managing dynamic data such as user accounts, reports on system errors, and transactional data. The database structure it's offering provides great flexibility in how data is modeled. Mongodb's efficient querying of the database aids the platform in maintaining high performance and dependability even as the platform and the amount of data that it's handling continues to grow.

Handling of the reward system on the platform is managed via PayPal API integration. It allows developers to pay testers with a seamless payment processing system. Selection of PayPal eases adoption of the platform as it is well known and trusted by users. Within the API integration, both sandbox and production environments are supported, which facilitates comprehensive testing of payment workflows prior to deployment in the live environment. Webhooks alert users regarding transactions and point credits in addition to monetary transfers over automation ensuring reliable and timely earnings updates.

3.2.3 Development and Deployment

The development environment will use Visual Studio Code as the main IDE because of its support for JavaScript and React, debugging features, and wide range of extensions. For version control, the team relies on Git. As a hosted repository solution, GitHub provides collaboration tools necessary for team work.

GitHub is used for managing collaborative development work and version control. It centralizes code, allowing storage, review, pull requests, and issue tracking. It fosters collaboration by letting multiple developers work at the same time. Moreover, GitHub works with deployment pipelines, like Vercel, to help with Continuous Deployment, which lets codebase changes be quick, and also to allow it to be automatically deployed and tested throughout various stages of the development lifecycle.

The deployment approach utilizes a Node.js backend with a MongoDB database to manage and persist data. The front end React app is provided by a host with global reach and integrated SSL and CDN capability. The backend provides RESTful APIs to MongoDB and is designed for optimal

data return and scale on simultaneous users. Integration and deployment is CI/CD via GitHub, which simplifies changes and manages development history and streams.

3.2.4 Programming Languages

JavaScript is the core language that underpins both Express.js's server-side logic as well as React components. Moreover, HTML5 provides an appealing semantic framework for webpage automation. With multimedia capabilities along with modern file validations, HTML5 caters to a dynamic tester's needs.

3.3 Database Design

3.4 Gamification Framework

3.4.1 Point System Architecture

The platform implements a comprehensive point system to motivate user engagement:
Bug Severity Scoring:

- Critical Bugs: 500 points
- Major Bugs: 300 points
- Minor Bugs: 100 points

3.4.2 Reputation System

User reputation is calculated based on multiple factors:

- Successful bug reports (positive weight)
- Report accuracy rate (positive weight)
- Community ratings (positive/negative weight)
- Platform activity level (positive weight)

3.4.3 Achievement Badge

Digital badges are awarded for specific milestones:

- "First Blood": First verified bug report
- "Bug Hunter": 10 verified bug reports
- "Elite Tester": 100 verified bug reports

Chapter 4

Methodology

4.1 Research Design

This study will use Agile methodology, but in the first stages, user interface and workflow designs will only have their prototypes (figma, draw.io). This method will allow the stakeholders to give their input, so that we can improve the platform on each iteration. Feedback from people like software developers and aspiring testers can be used to achieve our objectives, this will help refine them throughout the development process, ensuring that the platform will meet out expectations. Through short cycles of development, also referred to as iterations, Agile facilitates the continuous delivery of valuable software. With these iterations, the software development team has the ability can gather data, that will address the users' concerns, and improve users' overall satisfaction [36].

4.2 Development Methodology

Agile Scrum will be used as a platform-specific methodology in developing the system. Agile Scrum is a framework that propagates repeated delivery, incremental delivery through teamwork and fast feedback cycles. Enables teams to effectively switch quickly to rapid process improvements and the adaptability of software development to customer requirements [37]. The project will be divided into 2-week sprint cycles, with five specifications being included in each:

- Sprint Planning: Identifying and prioritizing features of the platform to be implemented in the sprint [38].
- UI/UX Design: Designing user interfaces and user journeys to enhance usability and user engagement.
- Coding and Integration: Developing selected priority features for the platform.
- Testing: Conducting internal, and pilot tests, to ensure quality.
- Sprint Review and Retrospective: Gathering the feedbacks, and use them to make the platform better on each iteration.

This iterative process, a process of adaptation termed "inspect and adapt," is clearly in line with the very core of Scrum and is adopted whereby the feedback from using and stakeholder educates the developments and product adjustments respectively.

4.3 Functional Modules

4.3.1 Bug Submission and Review

- Testers submit bug reports
- Moderators evaluate validity.
- Verified reports are assigned credit based on category.

4.3.2 Bounty pool handling

- Developers must deposit bounty before a post is made visible.
- Central wallet is used to ensure fair escrow.
- Admin cannot withdraw pooled funds arbitrarily—requires a transparent ledger system.

4.3.3 Incentive and Credit System

- Fixed credit per bug severity (e.g., Critical: 500, Major: 300, Minor: 100).
- Developers may add bonuses to encourage deep testing.
- Reputation increases with verified reports.

4.4 Process Flow

4.4.1 User Registration & Authentication

User Registration Flow

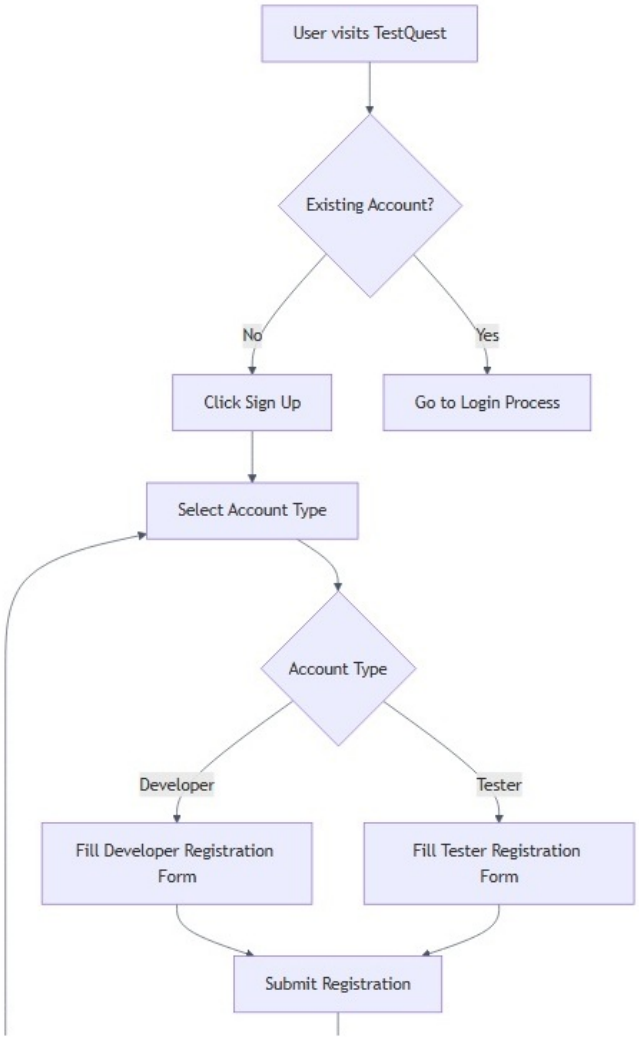


Figure 4.1: *User Registration Process Flow*

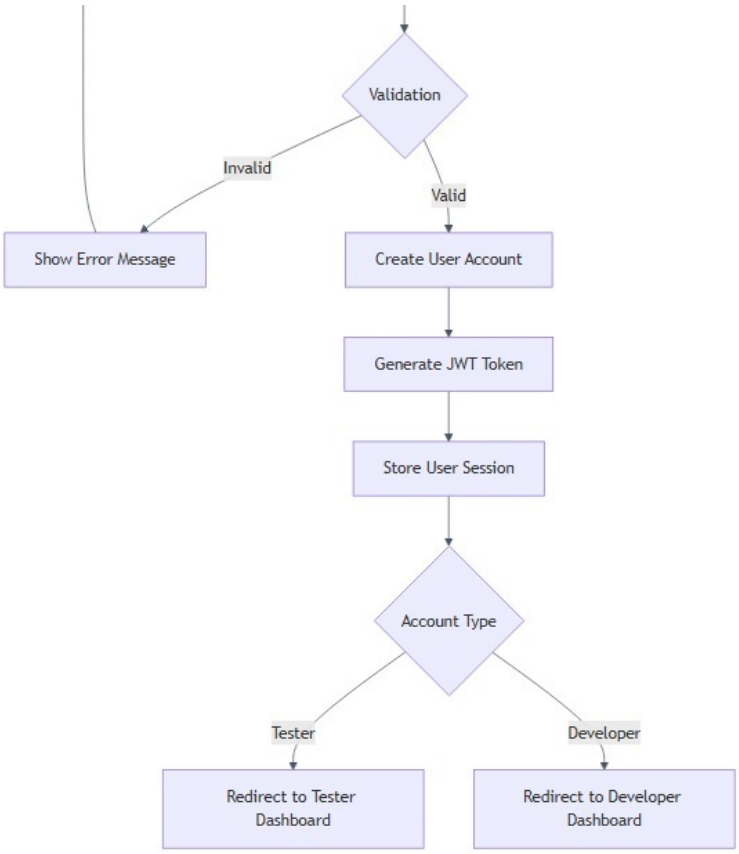


Figure 4.2: *User Registration Process Flow*

The authentication features of the platform are designed to ensure secure and role-specific access for all users. Role-based account creation allows individuals to register either as testers or developers, ensuring that each user is granted appropriate permissions tailored to their responsibilities. To streamline access and enhance security, the system integrates OAuth with widely used providers such as Google and GitHub. Session management is handled through JSON Web Tokens (JWT), which provide secure and efficient authentication across user interactions. Upon login, users are automatically redirected to dashboards specific to their designated roles, improving usability and workflow efficiency. In addition, form validation and error handling mechanisms are implemented to prevent incorrect inputs, reduce vulnerabilities, and enhance the overall reliability of the login process

OAuth Social Login Flow

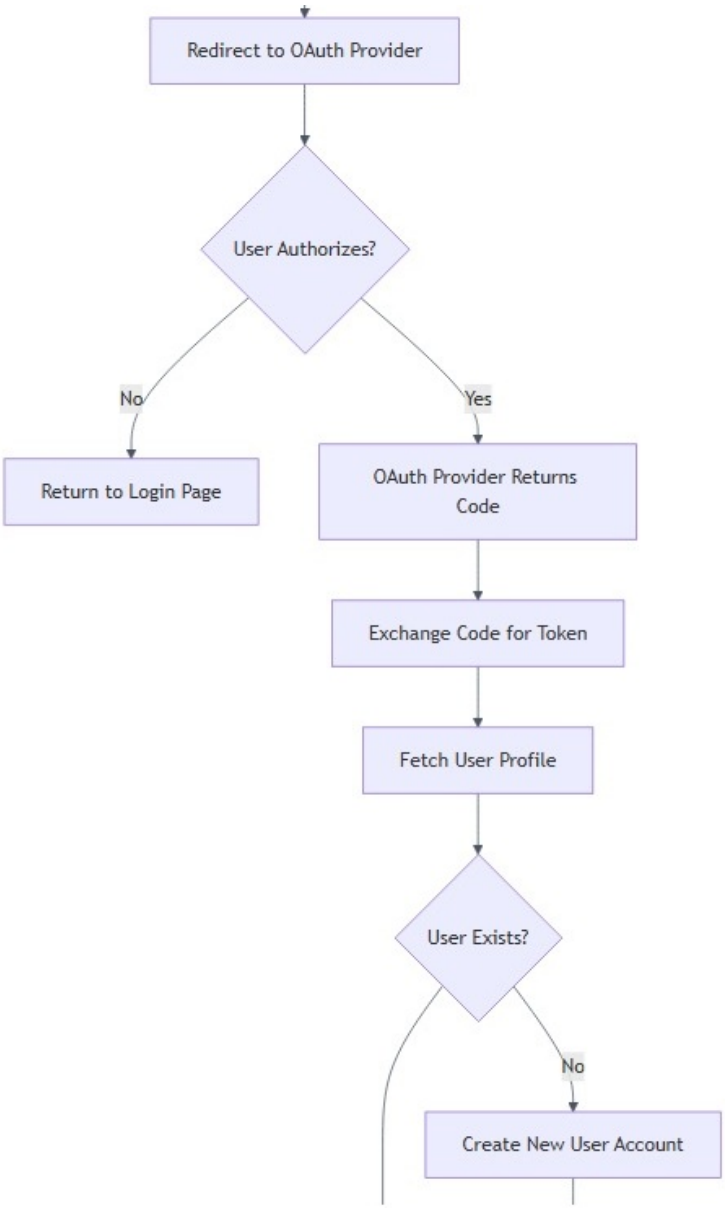


Figure 4.3: *OAuth Process Flow*

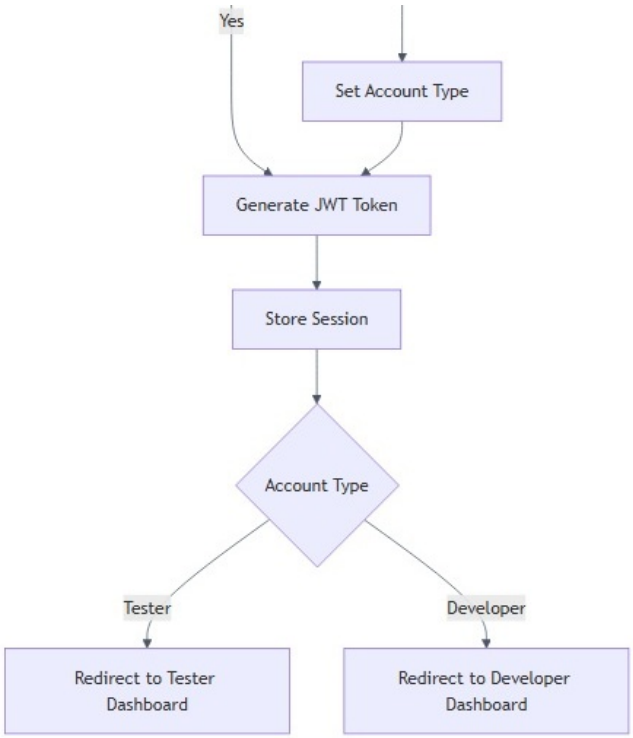


Figure 4.4: *OAuth Process Flow*

Key Authentication Features:

- Role-based account creation (Tester/Developer)
- OAuth integration with Google and GitHub
- JWT token-based session management
- Automatic dashboard redirection by role
- Form validation and error handling

4.4.2 Project Management Process

Project Post and Submission Flow

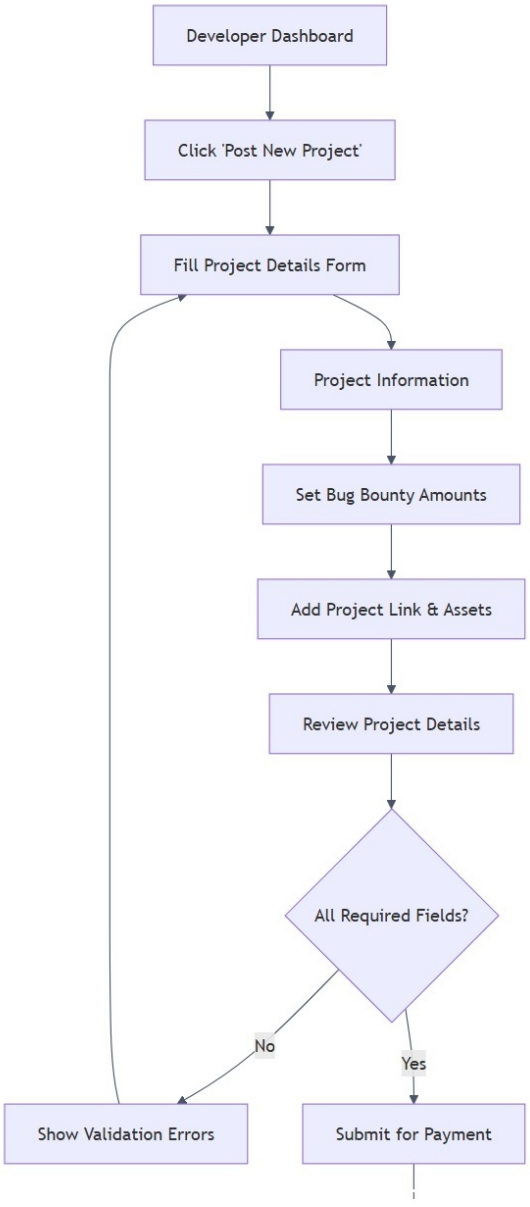


Figure 4.5: *Developer Project Post Process Flow*

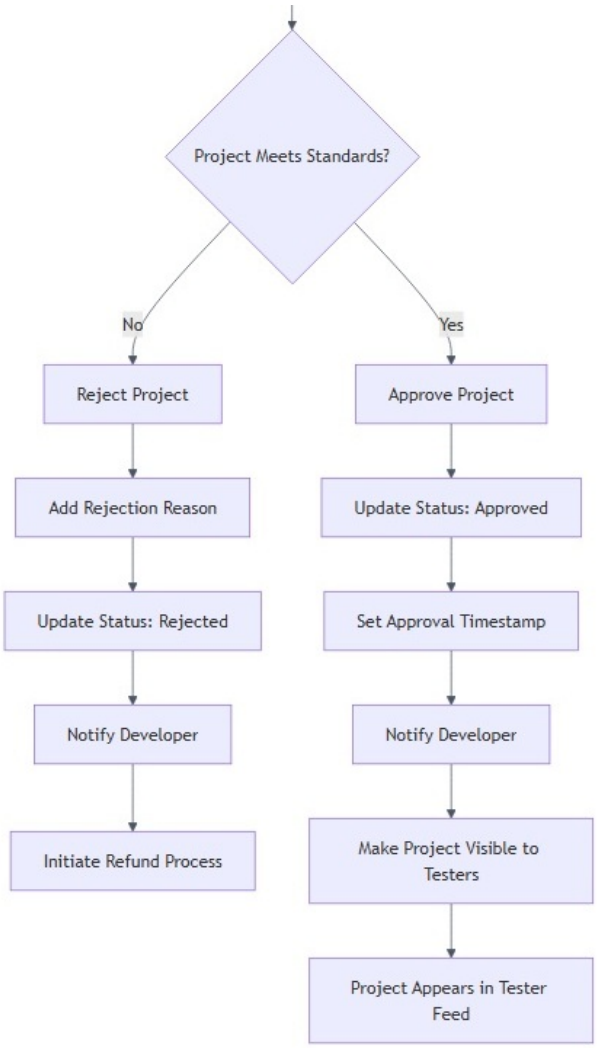


Figure 4.6: *Developer Project Post Process Flow*

In order to post a bug bounty, the developer have to click on the "Create Bounty" button. Then they enter the project title, description, and test scope. They also select the platform type which could be a web, mobile, or desktop application. Developers also can set criteria for bugs and set a reward structure such as Critical: 500 credits, Major: 300 credits, and Minor: 100 credits. There is also a 15% platform fee that is deducted. Upon completion, the bounty post is automatically published and becomes visible on the platform timeline.

Project Approval Process

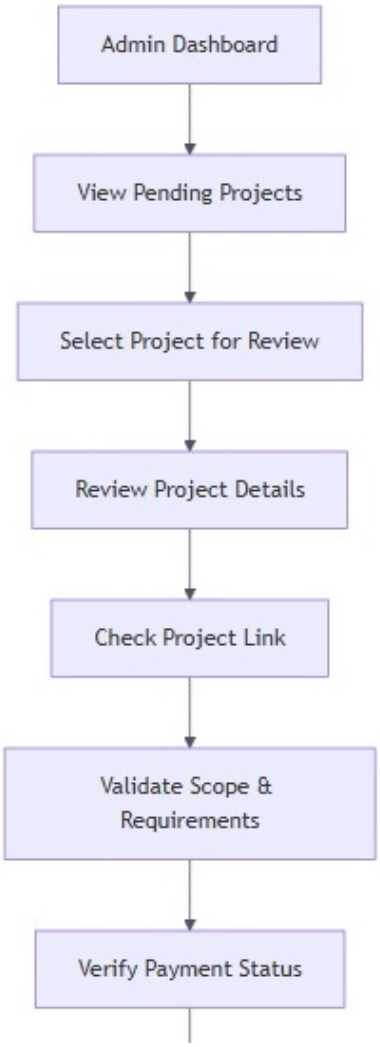


Figure 4.7: *Project Post Management Process Flow*

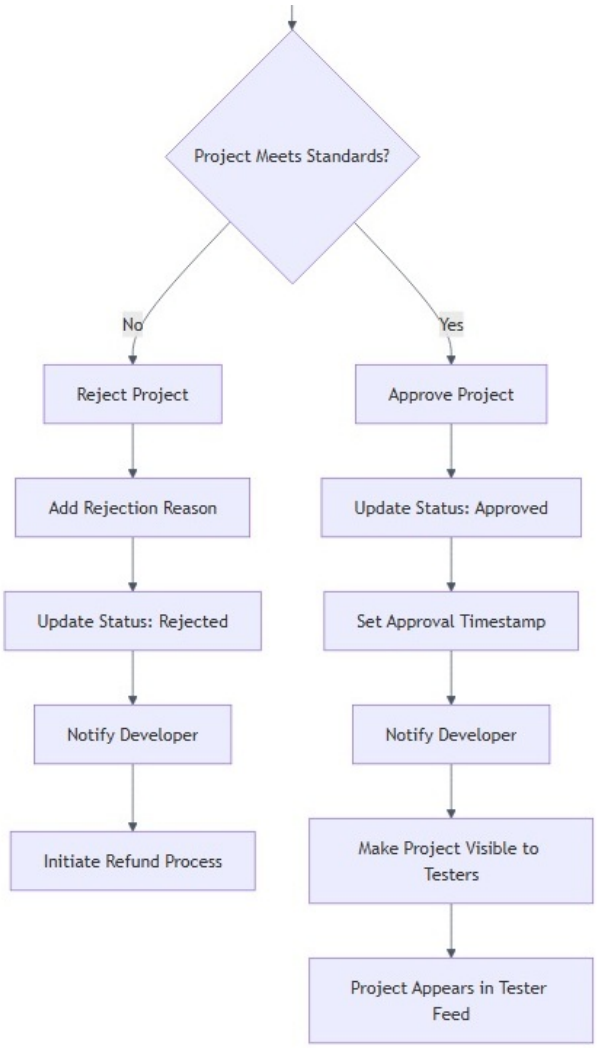


Figure 4.8: *Project Post Management Process Flow*

The platform offers complete project management tools intended to organize and control the bug bounty process. A guided project setup workflow allows developers to outline project parameters and ensure consistency across submissions. For project funding, integration with PayPal provides a financially safe and convenient solution. Automated refund processing for rejected projects also offers streamlined financial handling. Lastly, status tracking and notification systems provide stakeholders with project insights, promoting the efficiency, accountability, and trust of the platform ecosystem.

4.4.3 Bug Testing & Reporting Process

Bug Discovery & Report Submission

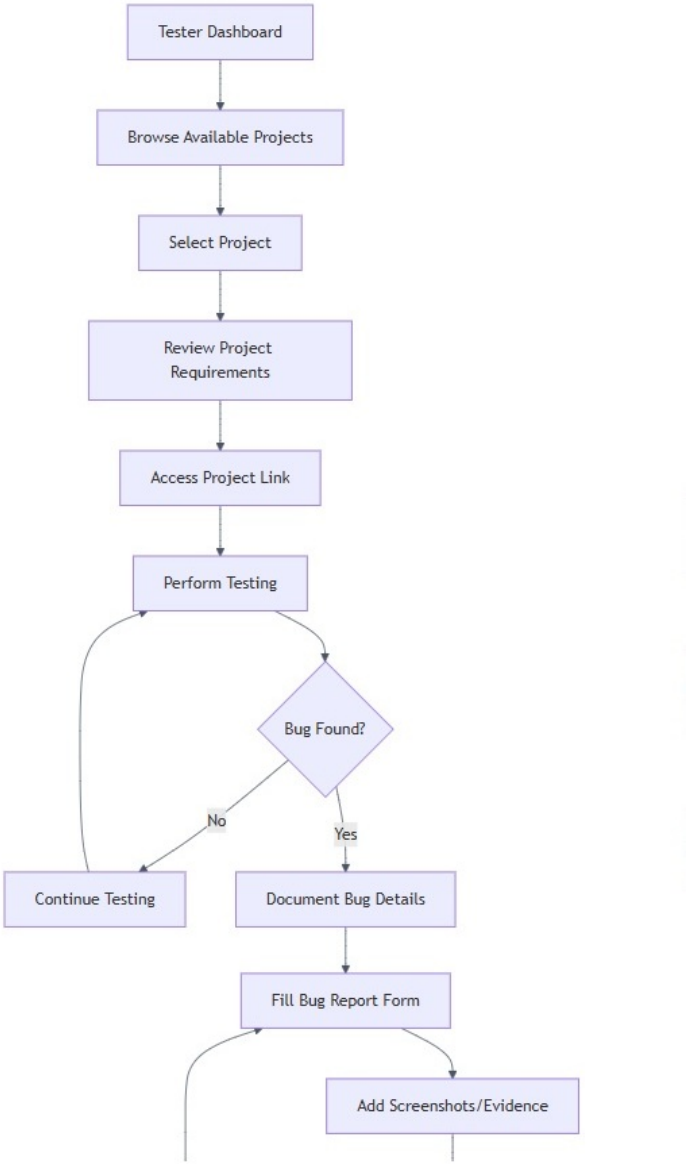


Figure 4.9: *Bug Report Submission Process Flow*

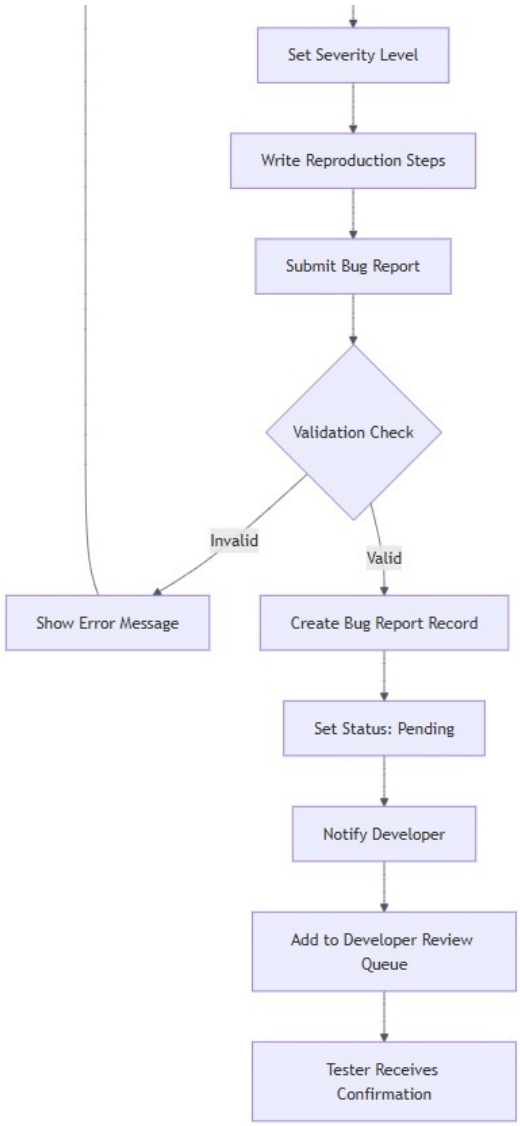


Figure 4.10: *Bug Report Submission Process Flow*

After downloading or accessing the project from the listing, the tester begins testing. After that, they can carry out the manual testing within the scope that the developer has specified. For every bug that is reported, the tester creates a structured bug report that includes the replication procedures, a comparison of the expected and actual outputs, and a classification of the bug's severity. They are also required to provide supporting screenshots or videos, in order to complete the report.

Bug Report Review Process

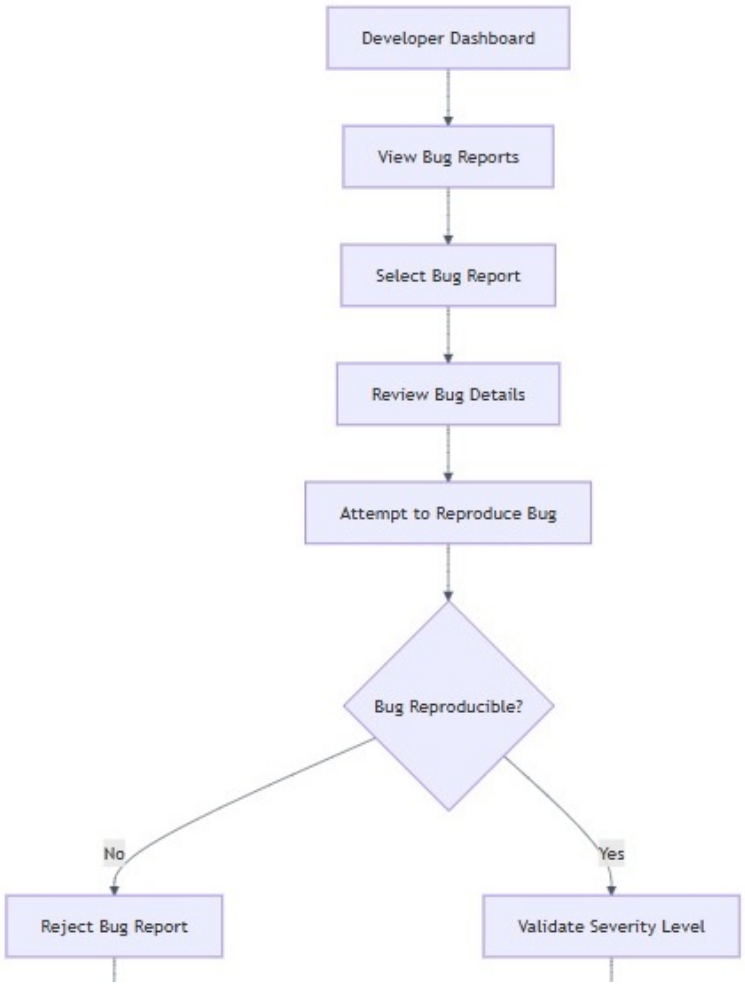


Figure 4.11: *Bug Report Review Process Flow*

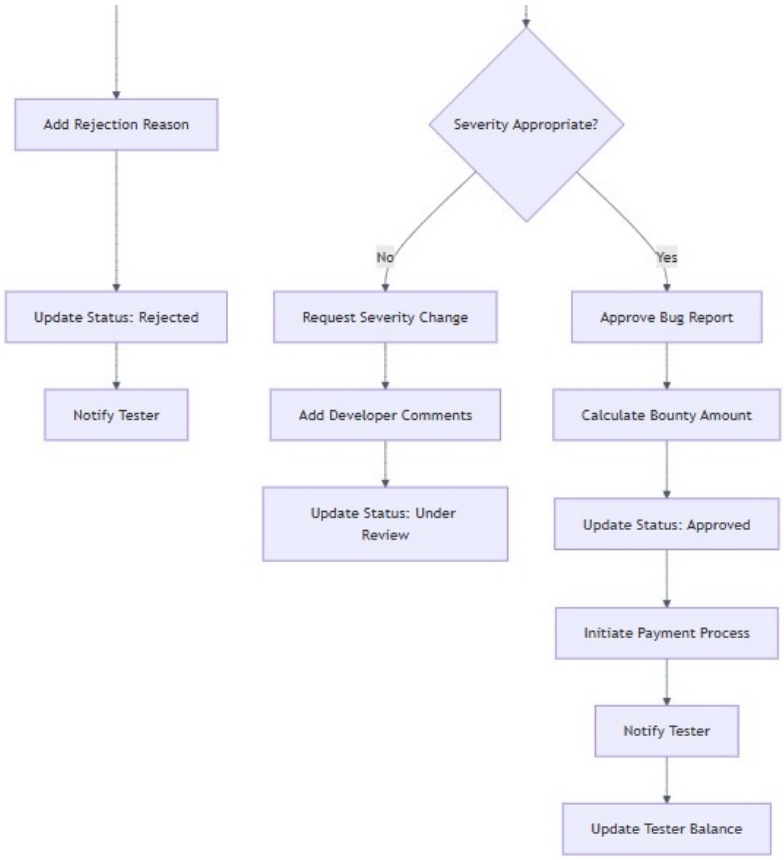


Figure 4.12: *Bug Report Review Process Flow*

The features pertaining to bug testing in the platform enhance accountability and efficiency in the reporting and testing process. Testers produce structured reports along with evidence that may include screenshots and log files. This makes the reports reproducible and verifiable. In order to fairly compensate the testers, the system adopts a severity bug bounty system which correlates the testers bounty to the potential impact of validated issues. This condition is specifically designed to enhance engagement. Testers receive automatic payment upon bug validation and processing by the system, while designed real-time notifications are sent. This condition is specifically designed to enhance engagement. Testers receive automatic payout in their balance upon bug validation and processing by the system, while designed real-time notifications are sent. These notifications keep the testers apprised of system status. All of this is designed to enhance seamless engagement in the

reporting and bug bounty process.

4.4.4 Dispute Resolution Process

Dispute Submission Flow

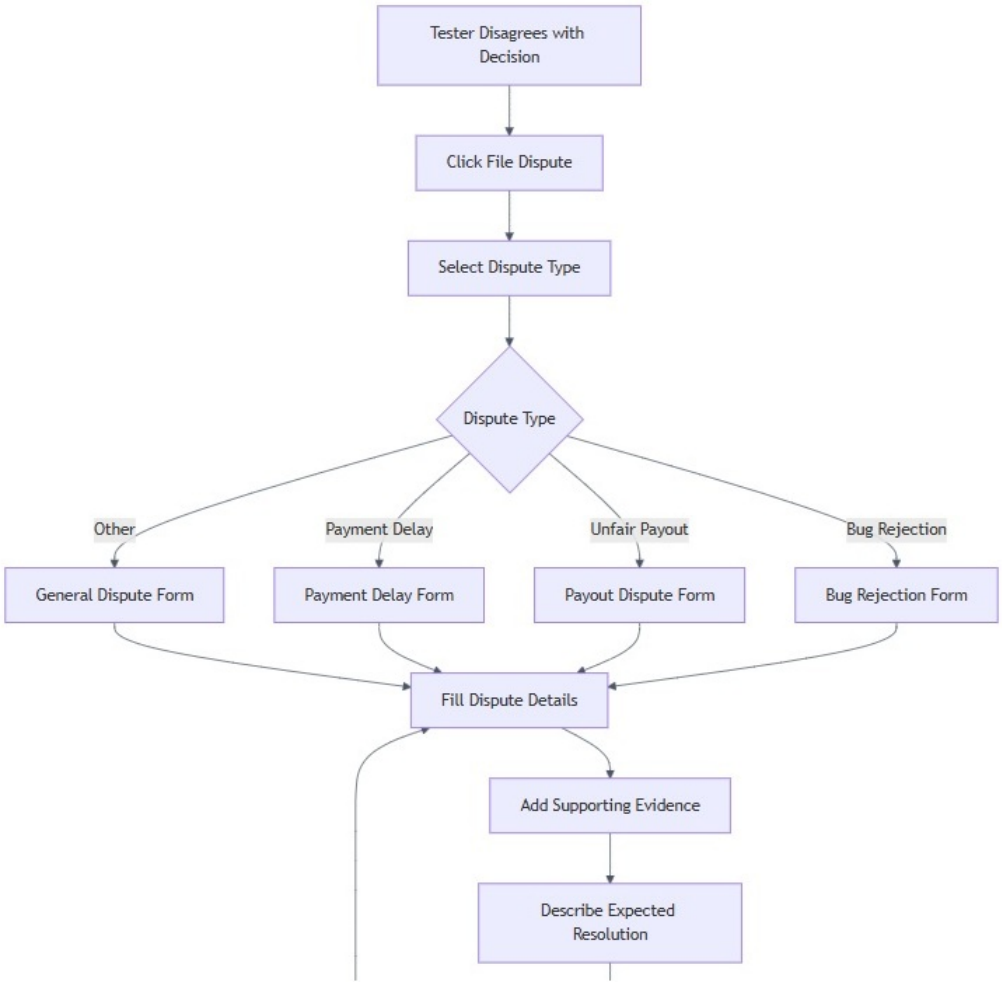


Figure 4.13: *Dispute Submission Process Flow*

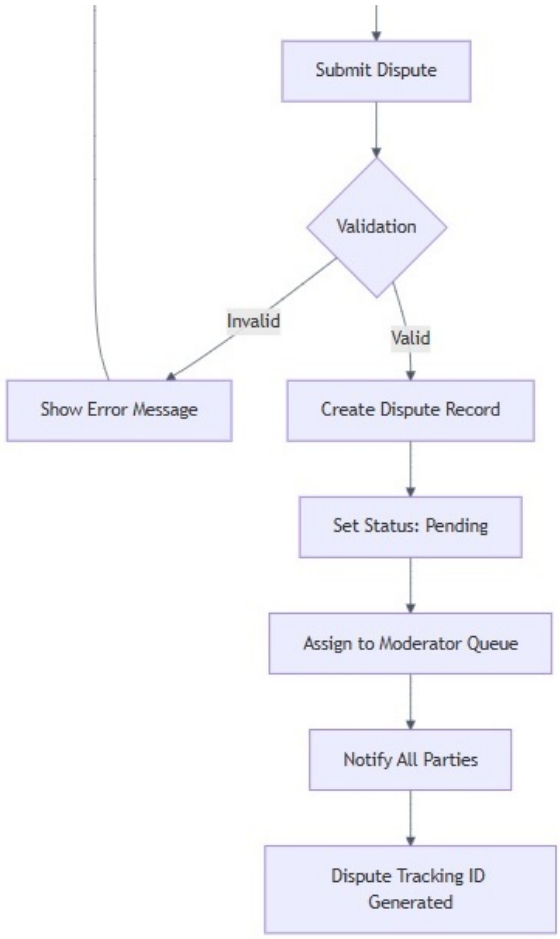


Figure 4.14: *Dispute Submission Process Flow*

Moderator Review Process

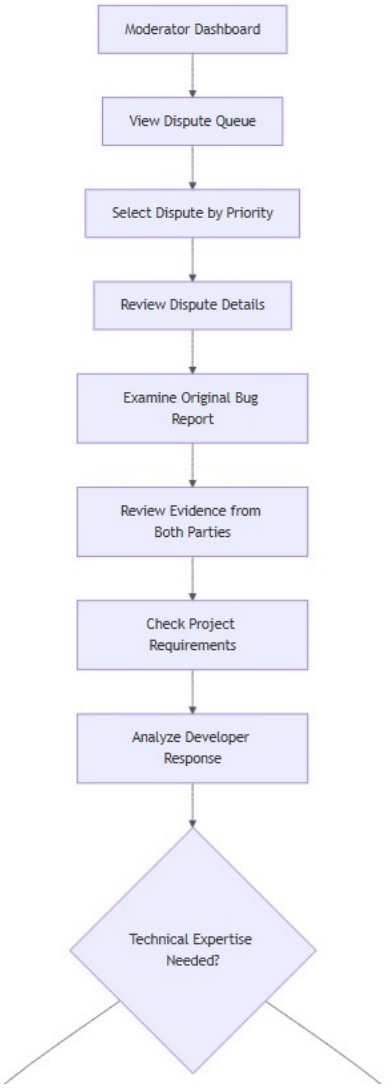


Figure 4.15: *Moderator Dispute Report Process Flow*

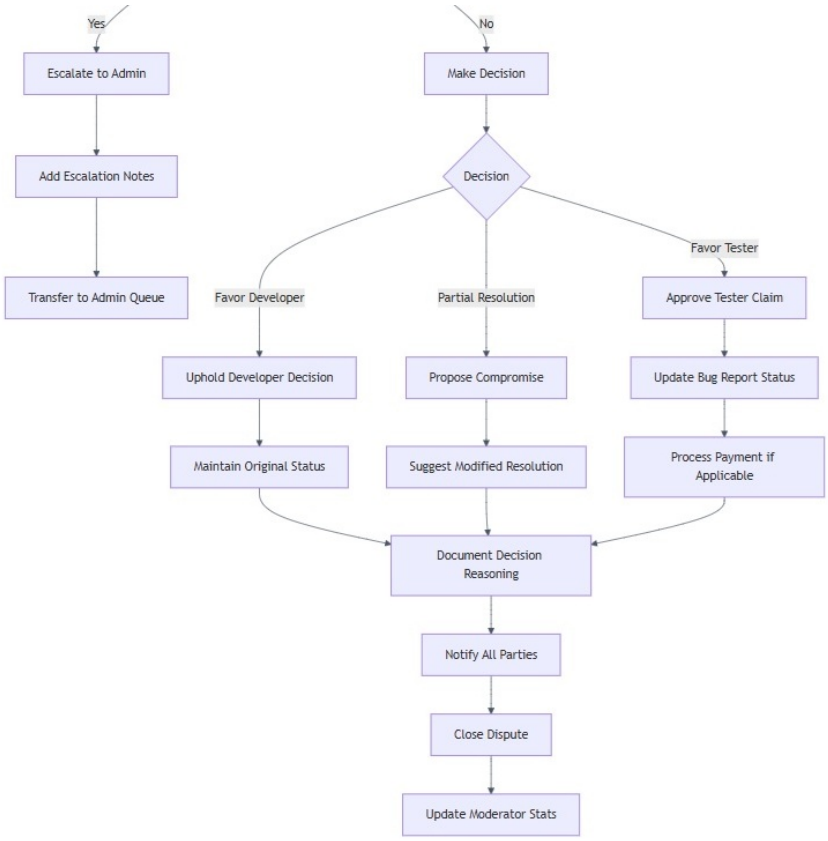


Figure 4.16: *Moderator Dispute Report Process Flow*

Features dealing with disputes on the platform promote equity, openness, and accountability. Through tailored forms, users can submit details for a range of dispute types. Each dispute is reviewed on a value basis and the relevant documentation is analyzed to assess the credibility of the claim. Moderators, who receive specialized training, assist in the mediation and resolution of disputes; those disputes which continue to be complicated or unresolved are sent to the admin for upper-level judgement. Due to the importance of documenting each resolution for a complete record, the system encourages trust among the developers, testers, and system overseers. This is done by providing extensive documentation for each resolution.

4.4.5 Payment Processing Flow

Project Funding Process

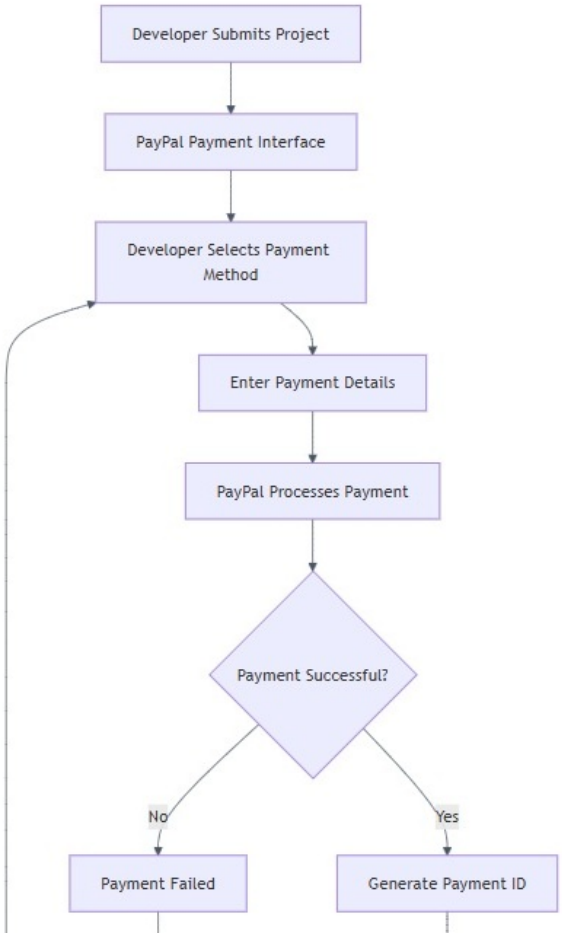


Figure 4.17: *Payment Funding Process Flow*

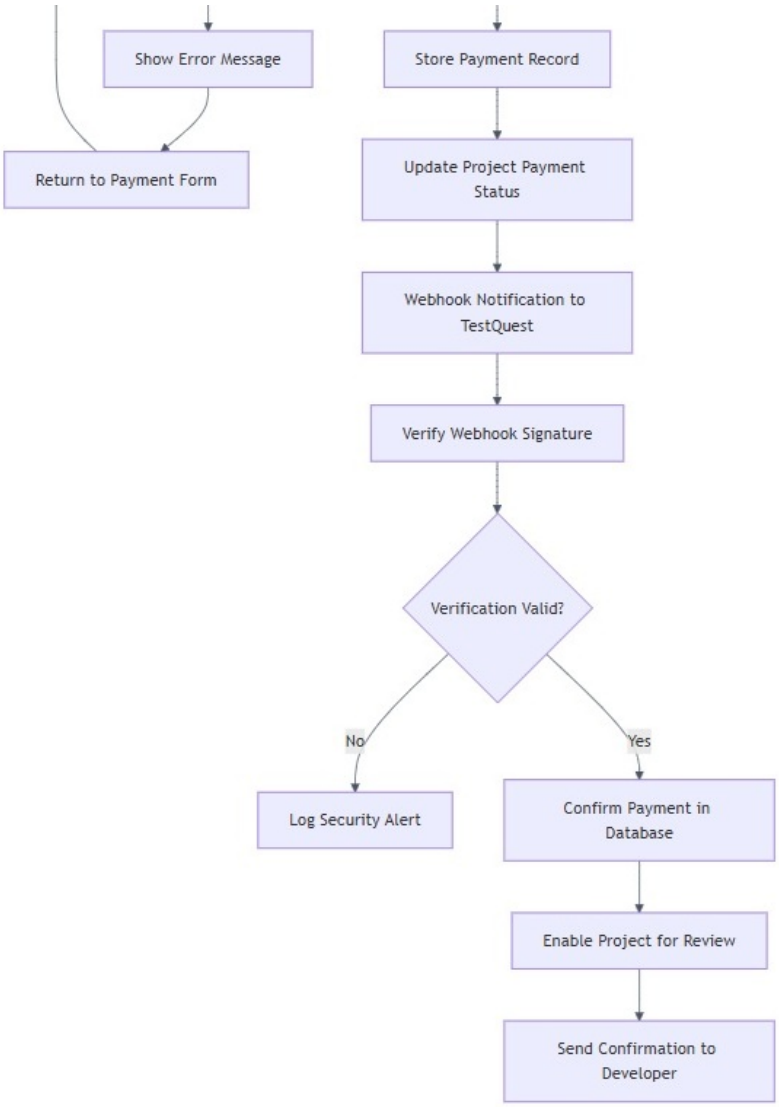


Figure 4.18: *Payment Funding Process Flow*

Bug Bounty Payout Process

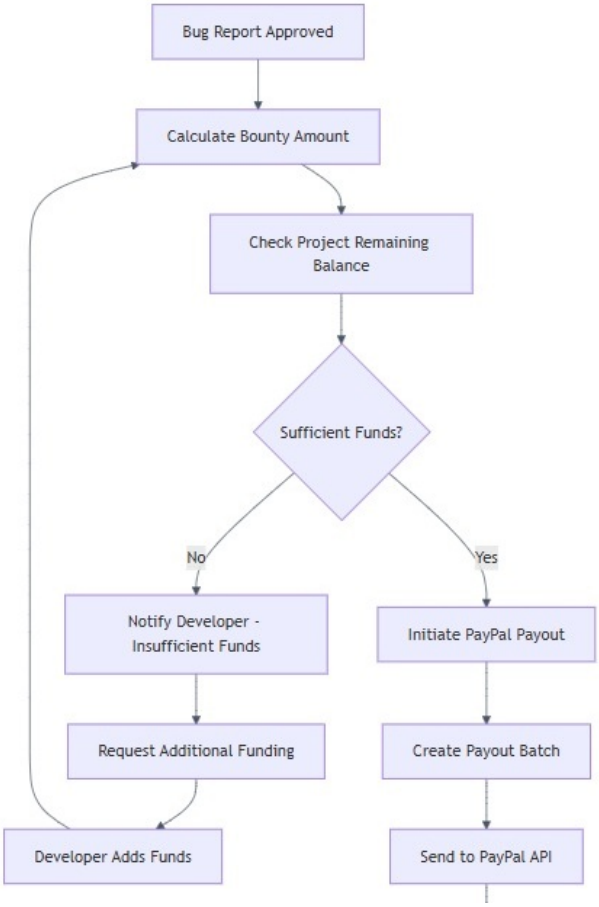


Figure 4.19: *Bug Bounty Payout Process Flow*

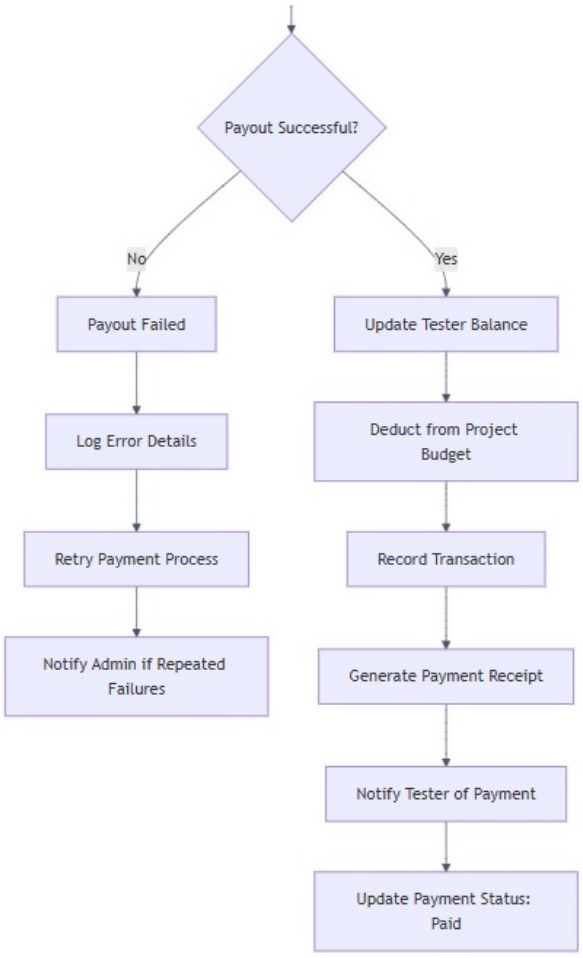


Figure 4.20: *Bug Bounty Payout Process Flow*

The platform includes strong security features for payment processing to guarantee the safety and dependability of all financial transactions. PayPal is integrated for the funding of projects and paying testers, as it is a highly trustworthy and globally used monetary transfer option. Moreover, all transactions are verified through webhooks to check the validity of the payment and to protect against fraudulent transactions. For the purpose of expediting rewards, the system features automatic calculation of bounties certification and distribution that is based on the severity level of validated bugs. Furthermore, automated error processing and the retry of payment transactions are designed for the possible gaps during transaction processing, thus enhancing uninterrupted functionality. The

system also features extensive transaction logging and automated processing to guarantee payment continuity. Finally, all financial activities undertaken on the platform are documented for auditing purposes and to ensure accountability.

4.5 Timeline

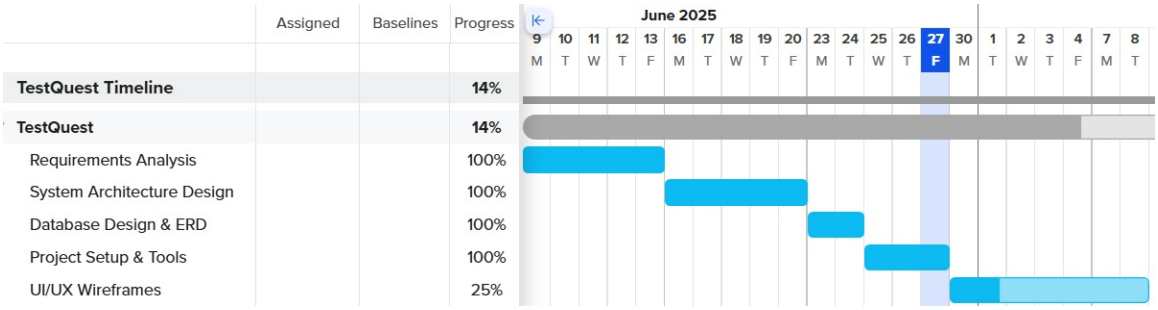


Figure 4.21: *Planning and Design*

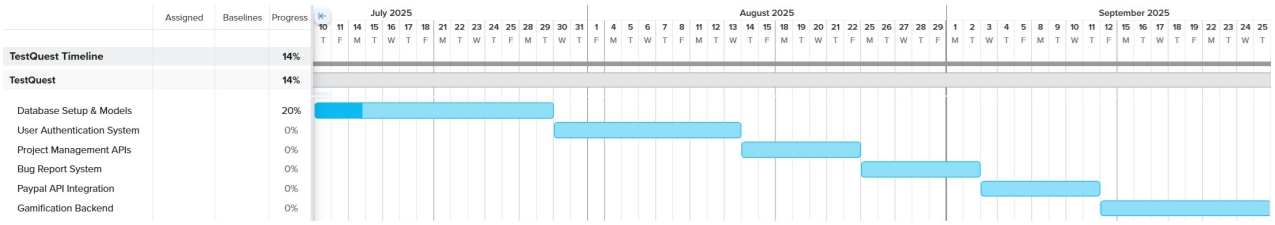


Figure 4.22: *Backend Development*

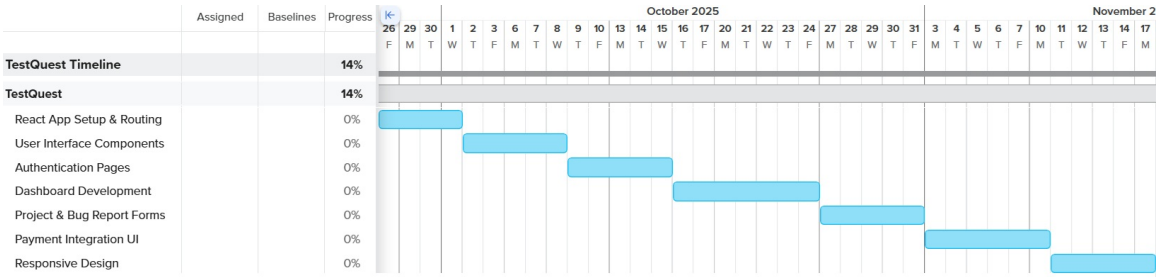


Figure 4.23: *Frontend Development*

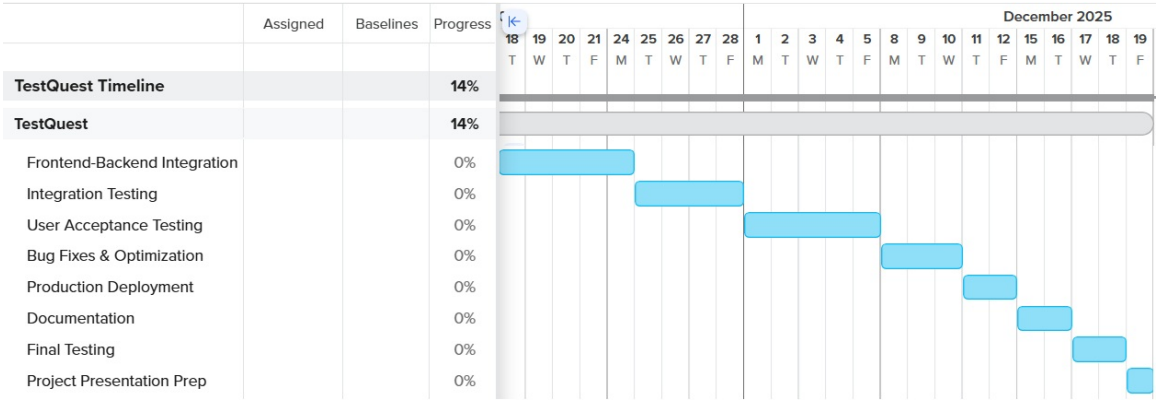


Figure 4.24: *Integration, Deployment, and Documentation*

4.6 Business Model

4.6.1 Value Proposition

For Developers

- Access to a larger testing community without the burden of long-term contracts.
- Get their software tested by a wider range of devices.

For Testers

- Monetary compensation for testing softwares.

- Gamified engagement through points, badges, and leaderboards.
- Build profile and reputation with validated bug reports, and get a chance to become a moderator.
- Progress from beginner to well-prepared tester for the industry.

4.6.2 Revenue Stream

To ensure fair pay for testers while sustaining the company's operations, TestQuest employs a commission-based revenue model. Such a model, however, requires developers to pay a project fee on the platform. The revenue model/cash flow system employs platform commission fee payment, fee allocation comprising an operational fee and an 85 percent bounty pool set aside for tester payment. The fee commission system maintains a platform project budget commission fee of 15 percent and a bounty pool of 85 percent for testers, which ensures that every budget member receives a payment. It maintains a minimum project fee of $20 USD (or around ₱1,120). At the project minimum, TestQuest keeps 15 percent, while 85 percent goes to the bounty pool. Operational and administrative costs including payment processing expenses (2.9-3.9% per transaction), hosting and infrastructure costs, customer support including dispute resolution, fraud and security services, among others, will be covered by the 15% platform fee. Also, the fee serves for sustained profit which facilitates the platform's future expansion and development. This revenue model, as a free-standing ecosystem, serves the purpose of incentivising and rewarding testing at scale commensurate to the level of testing the developers conduct. This balance facilitates the mission of the platform which is to promote a collaborative ecosystem that serves the interests of all the stakeholders.

Component	Description
Minimum Project Budget	$20 USD (approximately ₱1,120)
TestQuest Platform Fee	15% of total project budget (e.g., $3 from a $20 project)
Bounty Pool Allocation	85% of total project budget (e.g., $17 from a $20 project) – distributed to testers based on validated bug severity
Bug Reward System	Rewards vary by severity: Critical ($10–50), Major ($5–20), Minor ($1–5)
Point System and Monetization	1 point = $0.01 USD. Points can be converted into real money via PayPal. Minimum withdrawal threshold: $5 (500 points).

Bug Reward System

The Bug Reward System assigns compensation based on the classification of the bug's severity. System crashes, security breaches, and data losses incur the most serious consequences, and thus are compensated most highly, at between $10–50 USD per bug. Functionalities that are broken and issues of high usability are classed as Major bugs and receive between $5–20 USD per bug. Finally, typographical errors and other such Minor bugs receive $1-5 USD per bug. The point based system assigns one point for every one cent earned, which translates to $0.01 USD. Withdraws from the platform which is set at the minimum of $5 USD (500 points) is done to ensure fast and efficient payments. PayPal is also used as the payment system since it is flexible with currency conversion for overseas users, while platform transactions are currency based in USD for standard global alignment.

4.6.3 Target Market

Primary Users:

- Developers: Independent developers, small teams, startups
- Testers: CS students, freelance QA professionals, career changers

4.6.4 Competetive Advantage

Unlike competitors such as HackerOne and Bugcrowd that primarily focus on security testers, TestQuest offers a more inclusive and versatile testing environment. By fostering a learning environment, aspiring and beginner testers can gain practical experience even if they do not have advanced credentials. This strategy helps develop a supportive community and prepares professionals for the industry.

In order to enhance engagement , TestQuest incorporates such gamification features as:

- Points and rankings that increase engagement
- Recognition through badges and milestones
- Competitive leaderboards

Lastly, the platform guarantees a transparent and fair compensation. Testers are compensated for the severity of the bugs reported according to set payout thresholds. This helps in building trust and motivates consistent, high-quality contributions.

4.6.5 Cost Structure

Platform Development and Maintenance

These center on the continuing expenses related to building, updating, and maintenance of a given platform's capabilities and its distinguishing components of hardware and software. Developer costs, maintenance and feature development, system updates, and regular patches are all included.

Mongodb Database

Vercel is used to host the frontend of the platform because it offers quick deployment and automatic scaling with global delivery of front-end content. MongoDB is used to service the database to handle dynamic data which includes user data, bug reports, and other transaction data. The combination of these services provide cost challenges in terms of hosting, database, bandwidth, and other processing resources for the platform to function and scale as the user base grows.

Payment Processing Fees (PayPal)

As the payment processor, PayPal has to deal with the transaction fees of every payment processed within the system. These fees are charged per transaction, along with potential fixed charges too.

Marketing and User Acquisition

These are the costs associated with promoting the platform and attracting new users, including advertising, promotional activities, and other marketing initiatives designed to grow the user base.

4.6.6 Business Rules

User Registration and Verification

Developer Registration:

- Verification of email address.
- Linking a valid PayPal account is required for payment processing.
- Submission of valid id.

Tester Registration:

- Complete the email verification process
- Setting up a PayPal account is compulsory to withdraw rewards.
- Identity verification through valid id.
- Acceptance of the platform's terms of service and privacy policy is mandatory.

Moderator Selection:

- At least 6 months as an active tester on the platform.
- Maintaining an reputation score of 85% or higher.
- Completion of moderator training program.
- Subject to background checks, ongoing performance metrics, and other reviews.

Dispute Resolution Framework

Disputes between developers and testers are addressed using a framework which ensures equity, speed, and transparency. Unlike other platforms, does not divide the resolution process among the community. Unlike community-driven approaches, the resolution process is handled exclusively by designated moderators to ensure impartiality and consistency of outcomes.

Dispute Resolution Distribution:
Stage 1: Developer Response (within 48 hours). As soon as a tester contests the rejection of a submitted bug report, a notification is sent to the developer. Within 48 hours, the developer is supposed to provide a detailed evidence-based justification for the rejection, supported with the relevant paperwork. If the developer does not answer on the issue in the indicated timeframe, the case is forwarded directly to a moderator. This is to ensure that no conflict goes unresolved, and the submissions made by testers are given due consideration in a timely fashion.

Stage 2: Moderator Arbitration (within 72 hours).
When a case is escalated, whether it is due to a lack of action on the developer's part or a justification has been submitted, it is then the responsibility of a designated moderator. The moderator

is granted full access to the complete evidence set, which includes the bug report, developer response, supporting documents, and the project history. All evidence is then used to arrive at a binding and conclusive decision which includes the moderator's written justification. All decisions made are recorded and saved within the system to enhance system accountability, and auditability. In all other cases, appeals may only be made if there is proven evidence for moderator's bad conduct.

Dispute Distribution Mechanism

In order to achieve impartiality with regards to bias, a platform employs a policy of conflict-free assignment moderation. In the event of a dispute, a moderator will be automatically disengaged from the dispute if they have the potential of a conflict of interest, or if they have worked on the project previously or have prior disputes with either party or have a stake in the dispute. This preserves the integrity of the dispute resolution process. The system employs a simplified balancing mechanism when it comes to workload distribution. Moderators with fewer active cases are prioritized until workloads are balanced. If workloads are balanced, disputes are rotated in the system to minimize fairness issues regarding work distribution. Moderators' inactivity for 24 to 48 hours leads to automatic case reassignments, which is a way to prevent stagnation of disputes. In these cases, the inactive moderator suffers a reliability score penalty, while the newly assigned moderator is granted full access to all dispute record copies. This system fosters activity, averts standstills, and promotes uninterrupted workflow.

Moderator Accuracy Evaluation

The reliability of resolving disputes is enhanced with the evaluation of moderator performance. The system uses a form of audit sampling, where a sample of moderator outcomes (10%) is randomly selected and periodically assessed. Correct outcomes are determined based on adherence to the platform rules, while discrepancies are classified as errors. Furthermore, incidents as outcomes of incorrectly made decisions which are subsequently overturned of a certain moderator are also captured to monitor for policy violations which may be attributed to chronic error repetition.

Alongside audits, the system also has a form of feedback tracking which allows system developers and testers to attribute a single fairness ratings(positive or negative) to a dispute resolution. Although, feedback is not the principal form of accuracy assessment, this serves as a potential indi-

cator for reliability concerns in the system.

The combination of these diverse assessment options in a single framework promotes both operational robustness and administrative efficiency. This means that attribution of moderator accountability is a standard procedure.

Moderator Reliability Framework

There are three core components that govern moderator reliability: performance tracking, transparency, and progressive responsibility.

Performance Tracking:
Each moderator is assigned a reputation score which is representative of their compliance with the standards of the platform alongside their timeliness. The resolution deadlines set for moderators is one aspect. The other is the consistency of the moderator's decisions with the established dipute resolution rules.

Transparency:
Every decision that is made by a moderator is justified and archived as a decision log. Those logs are subject to random sampling for periodic auditing by the senior moderators or admins as an independent check and balance to avoid micromanagement.

Progressive Responsibility:
Moderators are part of a system that has varying levels of responsibilities. Every moderator starts by managing disputes that are of a lower value and with each subsequent dispute they resolve, they are able to ascend to more casious disputes as a result of met reliability indicators. This protects critical cases from the consequences of novice level inexperience while rewarding steady performance.

This balance of elements ensures that the platform is able to incorporate dispute resolution that is both practical and systemized. It systemizes dispute resolution by embracing a blend of both straightforwardness and complexity. The result is fair, streamlined, and transparent adjudication of disputes without overcomplicating the processing system.

Project Posting Management

Project Requirements:

- Minimum budget of $20 is required for posting a project
- Detailed project description including scope, testing needs, as well as acceptance criteria
- Defined bug severity levels with clear definitions and a corresponding reward system
- Projected testing period (not exceeding 100 days per project)

Project Approval Process:

- All projects subject to admin review within 24 hours
- Projects involving sensitive data require additional security clearance
- Rejection reasons must be clearly communicated to developers
- Appeals process available for rejected projects

Standards for Reporting Bugs:

- Must contain clear step-by-step instructions to reproduce the bug
- It is necessary to define expected output and actual output observed during the process.
- It is required to provide a screenshot or video for UI discrepancies.

  Assignment of severity: Critical, Major, Minor, or Minor

Overview of the Validation Process:

- The developer has 48 hours for assessment and validation of report claims.
- Duplicate reports are ignored.
- Invalid claims demand an explanation with evidence from the developers.
- Claims labeled as disputed are sent to a moderator for assessment after 24hrs, if the developer doesn't take action.

Quality Assurance:

- Testers are punished for submitting duplicate reports: 15minutes on the first offense, 1hr on the 2nd, and 24hours on the 3rd.
- Sustained pattern of imprecise reporting may lead to suspension of access.
- Valid bug reports are recognized and rewarded.

Payment and Reward Distribution

Point-to-Currency Conversion:

- Fixed rate of conversion: 100 points equals $1.
- Minimum withdrawal limit: 1000 points which is equivalent to $10.
- Maximum daily withdrawal limit: 5,000 points which translates to $50.
- Processing time: 1-3 business days.

Payment Processing:

- All payments IS processed through PayPal integration
- The platform will not be responsible for any transaction fees incurred via PayPal.
- Payments that fail to process will automatically attempt to process again after 24 hours.

4.6.7 NDA Compliance

The Non-Disclosure Agreement (NDA) will be used as a crucial legal document in the context of this capstone project in order to maintain confidentiality and protect sensitive information pertaining to software testing and development.

Platform Confidentiality Agreement

All testers is required to sign a confidentiality agreement for the platform. This agreement will ensure that any proprietary information regarding the platform's software and business operations, source code, and other sensitive details shall not be disclosed.

Project-Specific NDAs

Testers will need to sign project-specific NDas for software features which are considered overly sensitive. These NDAs are designed to impose certain confidentiality obligations to be kept, and affirmative duties in relation to the project such as not revealing technical details, features undergoing development, or any relevant testing data. Thus, these measures ensure that more sensitive components of the project are kept safe.

Legal Action and Termination

The NDA will stipulate both immediate termination of platform access and legal action in the event of a violation to these agreements. This measure will stop confidential information from being shared further and ensure that any breaches of confidentiality will have legal repercussions.

4.7 Diagrams

4.7.1 Database Schema

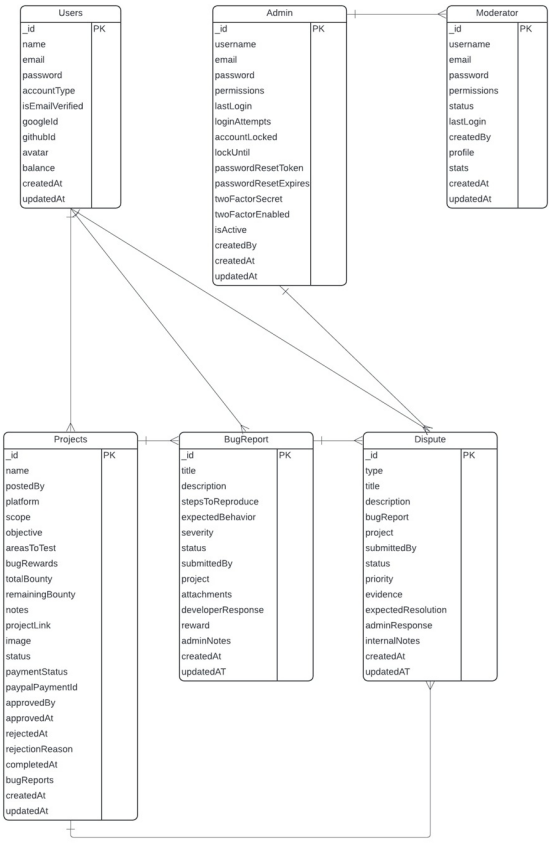


Figure 4.25: *NoSQL Diagram*

The architecture for database systems for TestQuest platform's operations is composed of six essential entities. At the lowest level, User entities capture both testers and developers, all differentiated by the accountType field. All users have stored credentials for authentication along with verification status for emails, OAuth integration IDs for Google and GitHub, account balance as a wallet, and varying timestamps. This is the only one entity that relates to all other entities as users and system actors integrate to projects they post, bug reports they submit, and developers' feedback they reply to along with disputes they raise. The inactivity of the system is captured by encapsulat-

ing Admin and Moderator as separate entities. Admin has system designed security for two factor authentication, account locking, tokens for password resets, and controlled permissions for users, projects, payments, system settings, and analytics for overall resource management. Moderators have a narrower scope for permissions relating to dispute moderation which consists of the ability to view, resolve, delete disputes, and ban users as well as access to analytics. All entities have their login sessions and profile data monitored while moderators have additional sample control for the analytics of their dispute resolution to capture system inactivity. At the operational core are the Project and BugReport entities. Projects hold all information about a bounty campaign, including reward structures for different bug severities, the total and remaining bounty amounts, specifications of the platform, the scope of testing, and workflows of multi-stage approvals, which have pending, approved, rejected, and completed workflows. Each project connects through PayPal integration to its poster, approver, related bug reports, and payment information. Bug reports hold all detailed testing information, including steps of reproduction, expected versus actual behaviors, severity levels, attachments, responses, and feedback about rewards. Lastly, the Dispute entity handles the resolution of conflicts between the testers and developers, linking to a bug report and project, which outlines the various dispute types, including bug rejection, unfair payout, and payment delay, tracking dispute levels and status priority, holding evidence and internal notes, and responses from the admin. The whole system is made of ObjectId references within MongoDB, and all entities hold timestamps for audit tracking. Careful enumeration of statuses is applied to manage workflows through the testing lifecycle, especially auditing for control and integrity.

4.7.2 Level 0 Diagram

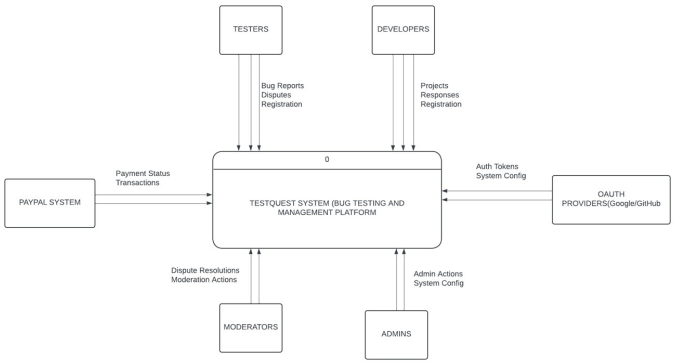


Figure 4.26: *Level 0 Diagram*

The Level 0 Data Flow Diagram shows the system as a single, centralized, and self-contained process and having multiple unified linked interactions with external entities. These entities consist of external TestQuest system users such as PayPal payment processing, authentication third-parties like Google and GitHub, testers, developers, administrators, and moderator. Testers and developers provide inputs like bug reports, project submissions, and login and payment information, whereas admins and moderators manage the actions as system oversight, and resolve disputes. PayPal processes payment confirmations, and the OAuth providers facilitate authentication tokens and user information. Project listings, bug report updates, payment information, dispute details, analytics, and transaction requests are responses that the system outputs based on the entity providing the inputs. TestQuest operates on multiple internal data stores including user, project, bug report, dispute, admin, and payment databases. It was designed with system boundaries which consist of a frontend built with React, a Node.js/Express backend, a MongoDB database layer, authentication services, integration with PayPal, and secure file storage. OAuth2 and JWT authentication, encrypted user data, and controlled access with strict admin functions provide the necessary security. This designed system fosters a well-organized and secure structure while facilitating interactions with all external stakeholders.

4.7.3 Level 1 Diagram

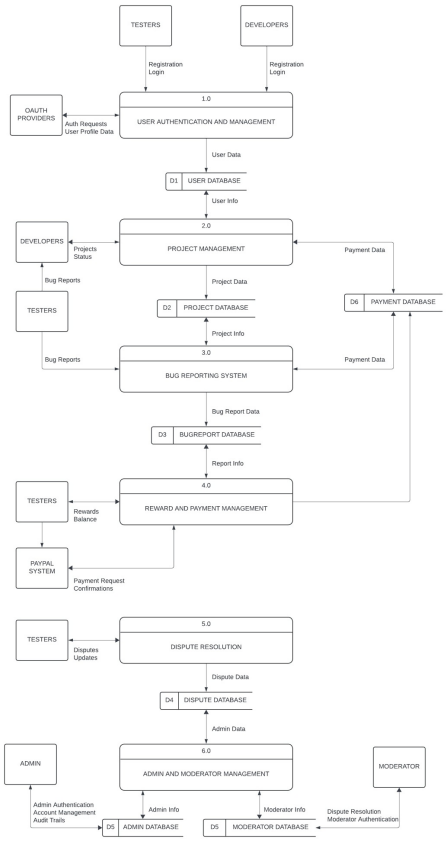


Figure 4.27: *Level 1 Diagram*

The Level 1 DFD TestQuest has identified the system components into six core areas: user authentication, project management, bug reporting, reward and payment handling, dispute resolution, and admin/moderator management. Each of these processes involves external actors like testers, developers, PayPal, and OAuth and uses user, project, bug report, dispute, payment, and moderator databases. User workflows are designed to move efficiently across the modules. For instance, the bug reporting and payment processing systems are directly connected to payment processing rewards. Moderators and admins manage escalated disputes, while system management functions autonomously control security, permissions, and audit trails. All of these combined offer a well-ordered and consistent approach to managing the entire system.

4.7.4 Use Case Diagrams

Tester's Perspective

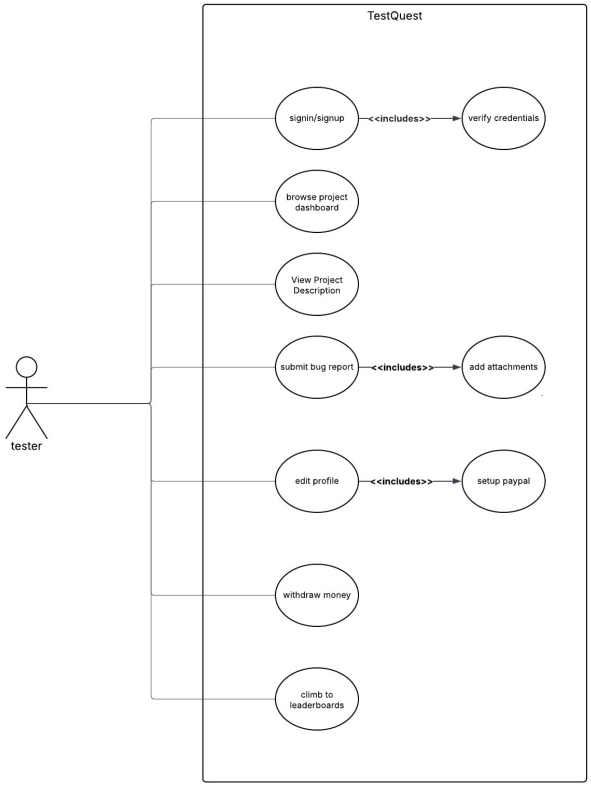


Figure 4.28: *Use Case Diagram - Tester*

This use case diagram shows TestQuest from the Tester's perspective. A tester can perform several key activities within the system: they can sign up or sign in (which includes credential verification), browse available projects on the dashboard, and view detailed project descriptions to understand testing requirements. When they discover bugs, testers can submit comprehensive bug reports that include file attachments for evidence like screenshots or logs. The platform also handles the business side by allowing testers to edit their profiles, set up PayPal integration for payments, withdraw earned money from successful bug discoveries, and climb leaderboards to track their ranking among other testers. This creates a complete ecosystem where security researchers can find testing opportunities, report vulnerabilities, and get compensated for their findings.

Developer's Perspective

Figure 4.29: *Use Case Diagram - Developer*

This use case diagram shows TestQuest from the developer's perspective, illustrating how project owners interact with the bug bounty platform. Developers can sign up and verify their credentials to access the system, then browse the project dashboard to manage their listings. They can post new projects with attachments like documentation or test cases, and view detailed project descriptions to monitor their submissions. The system allows developers to view project report logs to track testing activity and see the current bounty pool allocated to their project. For financial management, developers can edit their profiles, set up PayPal integration for payments, and deposit funds into bounty pools to incentivize testers. An admin user oversees the platform by approving project posts and rewarding testers for valid bug discoveries. This creates a comprehensive ecosystem where developers can crowdsource security testing by posting projects, funding bounties, and managing the entire bug discovery process through a centralized platform

Moderator's Perspective

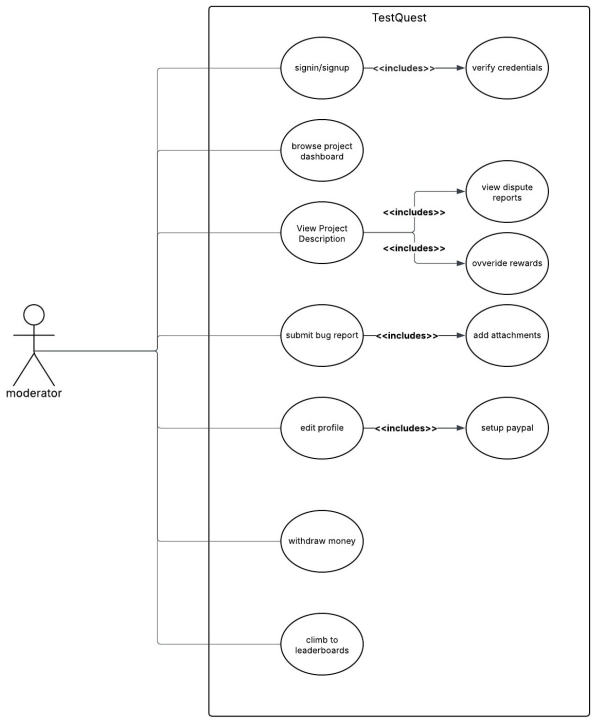


Figure 4.30: *Use Case Diagram - Moderator*

This use case diagram depicts the moderator role within TestQuest, showing how platform administrators manage and oversee the bug bounty ecosystem. Moderators have comprehensive access to the system, starting with standard authentication through signin/signup and credential verification. They can browse the project dashboard and view detailed project descriptions to understand what's being tested. Crucially, moderators handle dispute resolution by viewing dispute reports when conflicts arise between testers and developers over bug validity or bounty payments. They also have the authority to override rewards, allowing them to adjust compensation when standard automated systems may not capture the full value or complexity of a discovered vulnerability. Like other users, moderators can submit their own bug reports with attachments, manage their profiles, set up PayPal integration, withdraw earnings, and compete on leaderboards. This role essentially serves as the judicial and administrative backbone of the platform, ensuring fair play, resolving conflicts, and

maintaining the integrity of the bug bounty process between developers and testers.

Admin's Perspective

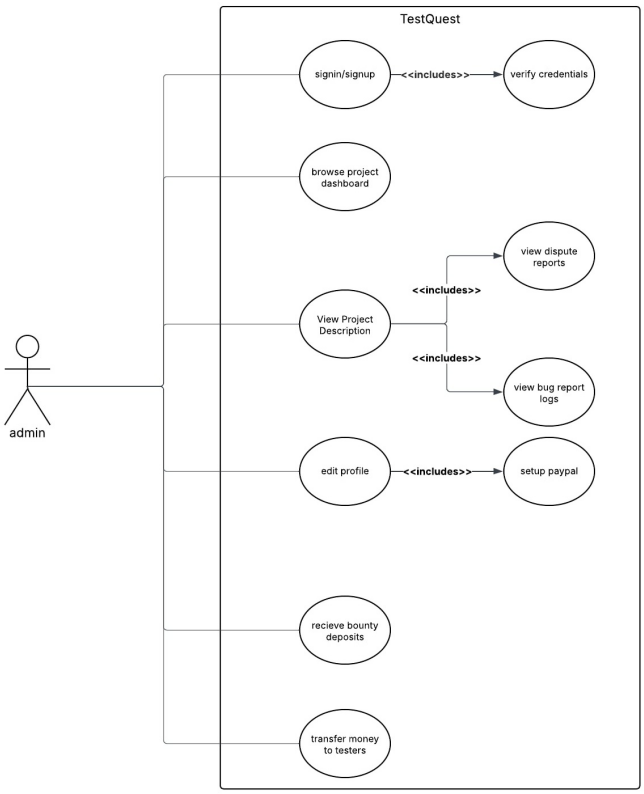


Figure 4.31: *Use Case Diagram - Admin*

This use case diagram illustrates the admin role within TestQuest, highlighting the highest level of administrative control over the bug bounty platform. The admin has foundational access through signin/signup with credential verification and can browse the project dashboard to oversee all platform activity. They possess comprehensive monitoring capabilities by viewing project descriptions, dispute reports, and bug report logs to maintain oversight of all interactions between testers and developers. The admin manages their own profile and PayPal setup for platform-related financial transactions. Most importantly, the admin controls the financial backbone of the system through two critical functions: they can recieve bounty deposits from developers who want to fund their

projects, and they have the authority to transfer money to testers when bugs are validated and rewards are due. This role essentially serves as the central financial hub and ultimate authority of the platform, ensuring proper fund management, dispute resolution, and overall platform governance while maintaining the trust and integrity of the bug bounty ecosystem.

4.8 Deployment Plan

Environment Setup

Vercel will provide the hosting platform as well as a CDN for the frontend application. MongoDB will be the main database for handling user accounts, bug reports, and transaction records. Role-based authentication and access control rules will secure the database eliminating unauthorized access. Protected data will only be reachable to users with a valid purpose. Also, the environment setup will provide distinct development and production environments to preserve data integrity and permit innovation safely.

Backend Deployment

Create separate Cloud Functions for individual Express.js routes. Set PayPal's production API keys as sensitive data environment variables and ensure they are properly configured. Conduct functionality tests of every API endpoint in the serverless environment to confirm proper operation. For logging and monitoring systems, implement supervision on vulnerable thresholds that are critical to production use.

Frontend Deployment

Develop the React application for production to generate static files, which will be optimized. Deploy these files to vercel, which takes care of global proxying as well as caching and compression automatically. All API calls will be directed to the production backend, which connects to the MongoDB database for handling dynamic data and transactions.

Feature Testing

Examine the end-to-end user workflows for registration, project posting, bug reporting, and payment completion. Verify that the gamification features function as intended with points, reputation systems, and badge awards calibrating accurately. Conduct user testing to ensure that the features are working as intended.

4.8.1 Security Implementation

Hidden Admin URL

To maintain a clean and user-focused interface for developers and testers, a separate login page will be created exclusively for administrators. This admin login will be accessible only to authorized users, such as those with pre-approved email addresses or from whitelisted IP addresses, ensuring a higher level of security and role segregation. By isolating the admin access point—such as through a dedicated URL like testquest.io/admin/login—the main user interface remains streamlined for the platform's primary users, while administrative functions are kept securely separate.

Create a separate login page for admins:

Authentication Security

The authentication and authorization is managed the backend interface with node.js, integrated with MongoDB. Email verification is required during the registration/sign up to confirm the users' accounts.

API Security

Express.js middleware inspects every incoming request and sanitizes input fields, blocking attempts at SQL injection or script embedding. In addition, rate-limiting throttles excessive calls, defending the API from abuse and denial-of-service floods.

Payment Security

Payments are processed through a PayPal sandbox, where tokenized credentials protect sensitive card data. No financial information is written to the application database, and all wallet balances are calculated on the server to prevent tampering.

Data Protection

All user data is encrypted as it is stored, thanks to mongodb atlas. During transmission, HTTPS protects all traffic moving between clients and servers, preventing data interception or tampering during transmission.

Access Control

Permissions are assigned according to roles such as developer, tester, moderator, and admin, granting each group only the access it needs. Actions reserved for administrators call for extra proof of identity, and all such changes are logged for auditing purposes.

4.9 Testing and Evaluation

The effectiveness and user experience of the platform will be evaluated through internal and user testing. Each sprint, the project team will conduct an internal test that consists of functional and unit tests. The computer studies capstone students who will act as developers, testers, and moderators will pilot the platform through simulated workflows. The reliability of bug reporting will be evaluated based on validity, reproducibility, and actionability within certain boundaries from gamified testing-based frameworks [39].

Implementation of internal testing:
At the end of each sprint, the project team will conduct internal tests based on the features added during that sprint. This includes:

- Conducting unit tests on new modules, such as bug submission form, for basic operational accuracy.
- Evaluating interaction workflows such as submitting a bug report with attachment submission and severity level assignment or point withdrawal via PayPal Sandbox as functional tests.
- Cross module integration flow checks if submitted reports trigger reputation scoring or how confirmation from developers affects reward calculation when done manually.
- Recording problems in error tracking tools which are analyzed after assigning next sprint tasks related to those bugs to give time for resolution.

For internal tests, all results will be stored in issue tracking spreadsheets and all test cases executed prior to deploying any fixes so no changes can affect results inaccurately.

Implementation of Pilot User Testing:
Computer studies students, who are currently also taking their capstone projects, will be assigned roles as testers, developers, and moderators to create user testing workflows in the platform. This includes:

- Testers will execute tests by identifying and submitting bug reports based on a provided checklist or through exploratory testing.

- Developers will review submitted bug reports within the system; approving, rejecting, or processing reward-triggering steps.
- Moderators assessing the disputes tending to disagreements along with simulated compensation confirmation.

Pilot sessions will occur at the end of each sprint. Each session has defined tasks for participant assignment within controlled test environment setups. Reports generated in this phase will be analyzed alongside set criteria benchmarks to assess the reliability of the system being evaluated.

Evaluation Metrics:
Every reported bug will be evaluated for:

- Assessment – Applicability of the bug in the defined testing scope is determined.
- Reproducibility – The issue can be consistently replicated for the same results.
- Actionability – the report has enough details for the developers to act upon.

In addition to the above quality indicators, all bugs will be categorized into one of three severity levels, which will affect both tester rewards and developer prioritization: The test results along with

Severity Level	**Definition**	**Examples**
Critical	Bugs that break core functionality, cause system crashes, or result in data loss.	App not launching, user unable to log in, payment function not working.
Major	Bugs that impair significant features but do not halt primary operations.	UI layout failure on some screens, incorrect calculations, broken links.
Minor	Cosmetic issues or non-disruptive bugs that do not affect primary use.	Spelling errors, icon misalignment, non-blocking tooltip glitches.

participant comments as well as flagged problems will be assessed for usability to direct adjustments in future sprints. To evaluate level of engagement and ease of use, needs assessment and attitudinal surveys will be conducted after each pilot session.

4.9.1 Platform Testing Framework

The assessment of the platform will focus on the evaluation of the various components to determine its value as a senior project.

Phase I: Assessment of Functional Testing at the Component Level

Tests:

- Authentication modules, the user registration, and login)
- Core functionalities such as project posting, bug submission, payment calculations.

The objective is to confirm each system component meets established functional requirements. Acceptance criteria: Functional requirements are met and no open critical defects.

Phase II: Integration and Workflow Testing

To test:

- The entire bug lifecycle: submission → validation → payment distribution.
- Dispute resolution workflow: rejection → appeal → moderator intervention → resolution.

The objective is to check communication between each component and the end-to-end process flows.Acceptance Criteria: Each workflow runs without any data inconsistency or stops in process flows.

Phase III – User Acceptance Testing

Participants: 10-15 Senior Project Computer Science students.
The objective is to assess usability and functionality of the system with a designated group of users. This willl include task-based testing scenarios, Usability Scaling, and post-test interviews.

Metrics:

- Task completion rate (goal: ¿ 80
- Error rate per task
- Time taken per task

Collection Methods: Screen recordings, observation notes, and Google Forms submitted surveys.

Test Documentation

All testing phases formalize test logging as follows:

- Test Case ID and Description
- Expected and actual results
- Defect Severity classification: Critical, Major, Minor
- Closure status with results of re-testing

4.10 Data Collection Methods

For quantitative feedback to be obtained from testers and developers, questionnaires and surveys will be used. While the contrasting pattern will be looked into with the assistance of short structured interviews to gain qualitative insights. User requirements can be further analyzed and understood using these mixed methods [40]. Observing user interactions on the platform will complement the results of the survey as it will show real-time behaviors in usage; such methods are supported by mixed-method guidelines to integratively analyze qualitative and quantitative data. Short structured interviews will also be conducted with the participants, which will give them an opportunity to gain deeper perspectives about their experiences. This is in accordance with best practices for pairing survey and interview data in qualitative analysis platforms. Usability assessment will implement task-driven methods noting defined user goals while measuring key indicators such as completion rate alongside time to assess use effectiveness and efficiency. A multi-method approach using direct observation, surveys, and an all-in usability evaluation will be conducted. Of particular note for this survey is gathering peer feedback across the roles of developer, tester, and moderator which makes it holistic in nature with regards to role-based feedback on usability.

4.11 Ethical Considerations

This study will guarantee:

- Informed Consent: All participants will be informed about the purpose of their involvement, and what data will be gathered from them.

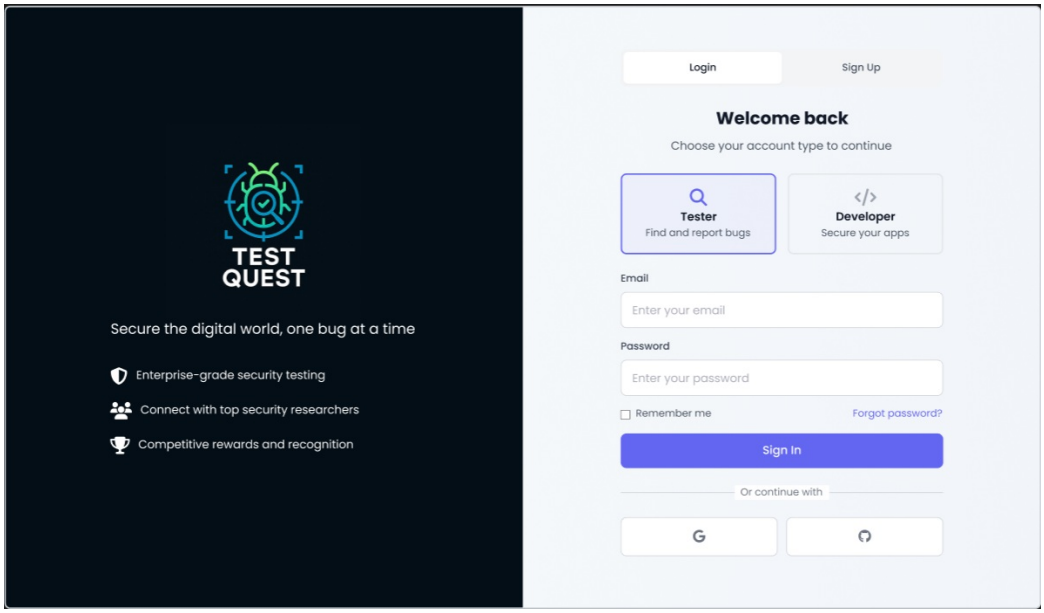
- Confidentiality: All participant data will be kept private and anonymous, and will only be used for research purposes.
- Voluntary Participation: Participants will not be forced to participate.
- No Exploitation: To stop abuse or exploitation, the platform will have equitable methods for compensating testers.

Appendix A

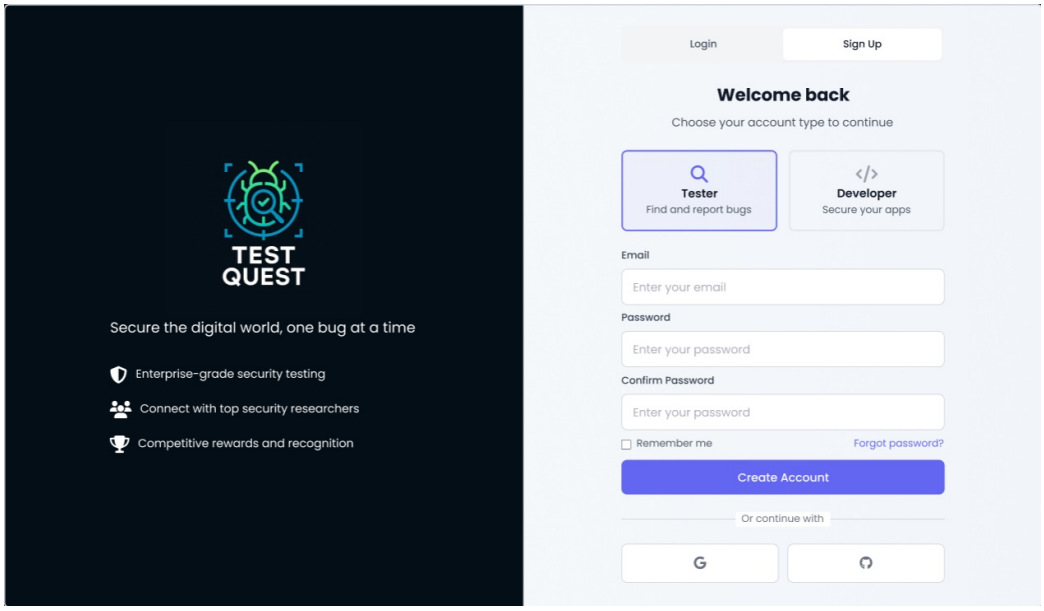
User Interface Designs

A.1 Authentication Screens

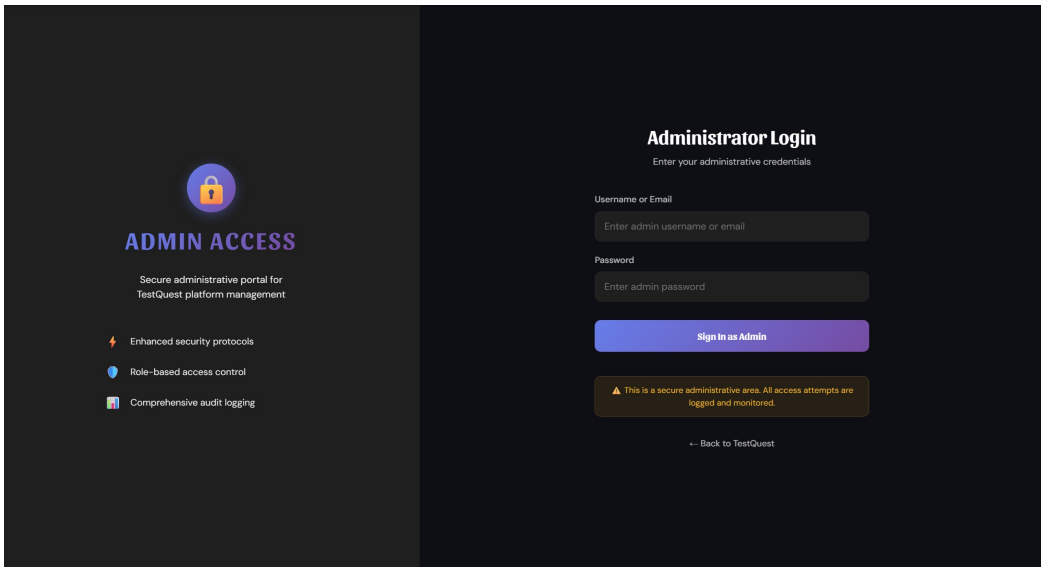
A.1.1 Tester/Developer Sign In



A.1.2 Tester/Developer Sign Up

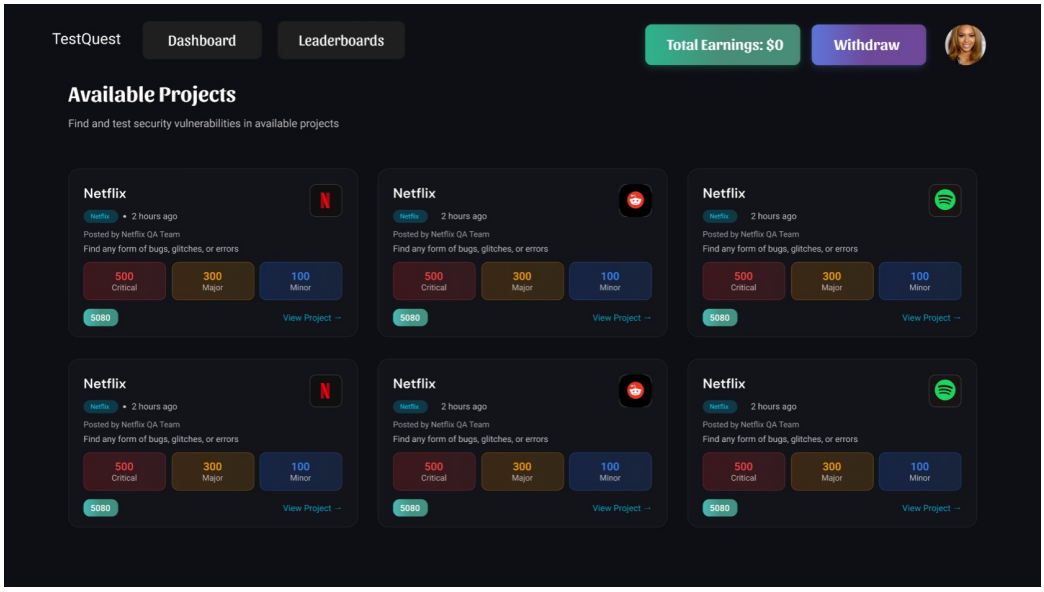


A.1.3 Admin Sign In

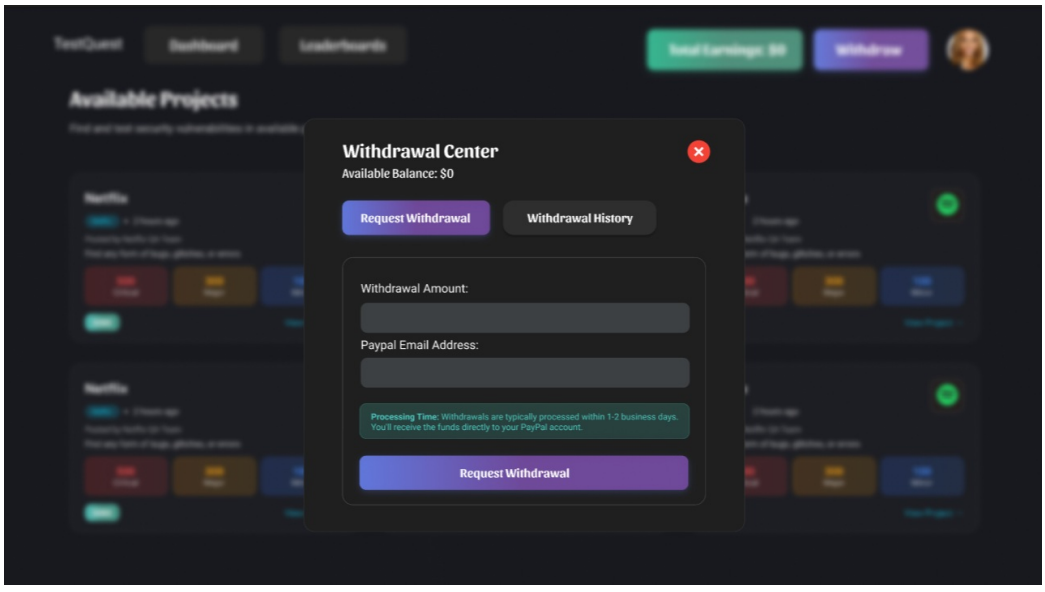


A.2 Tester Screens

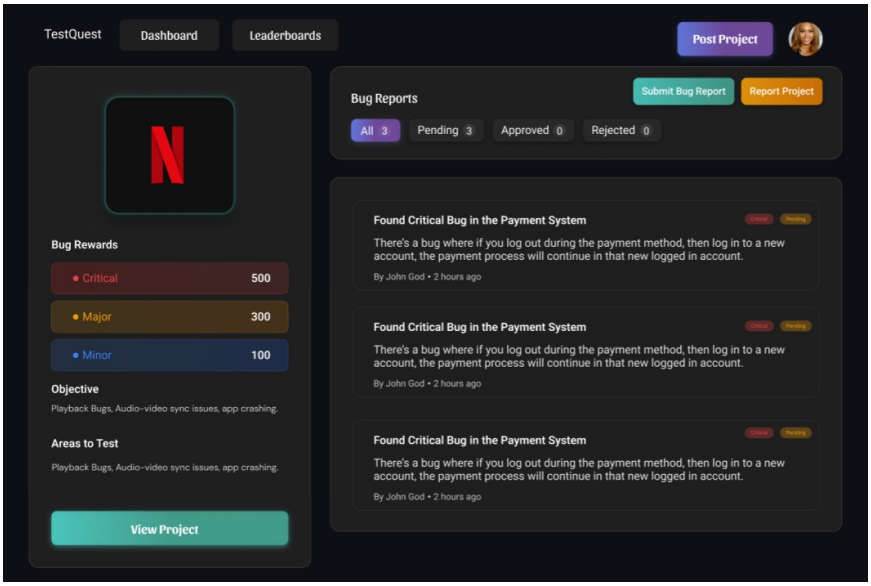
A.2.1 Tester Dashboard



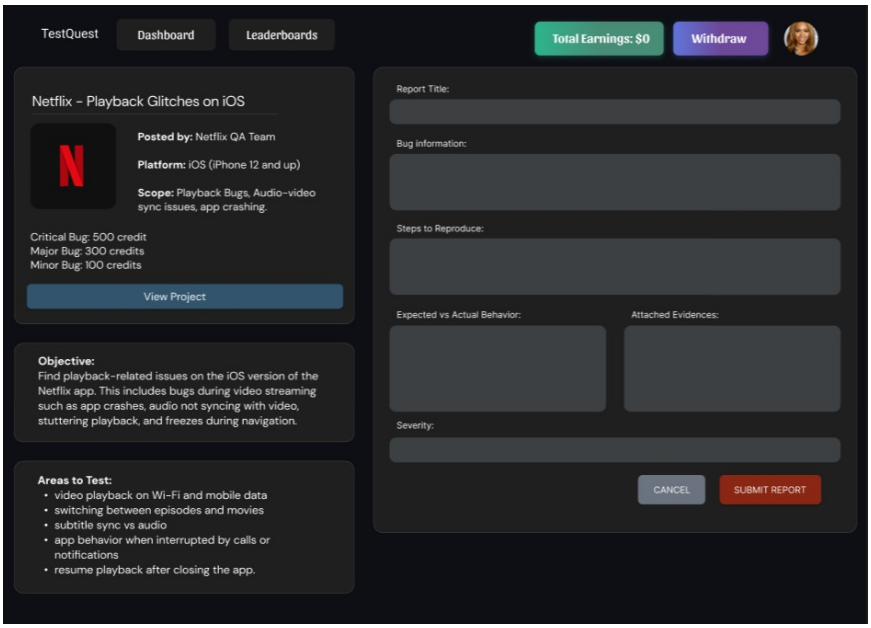
A.2.2 Withdraw Pop-up



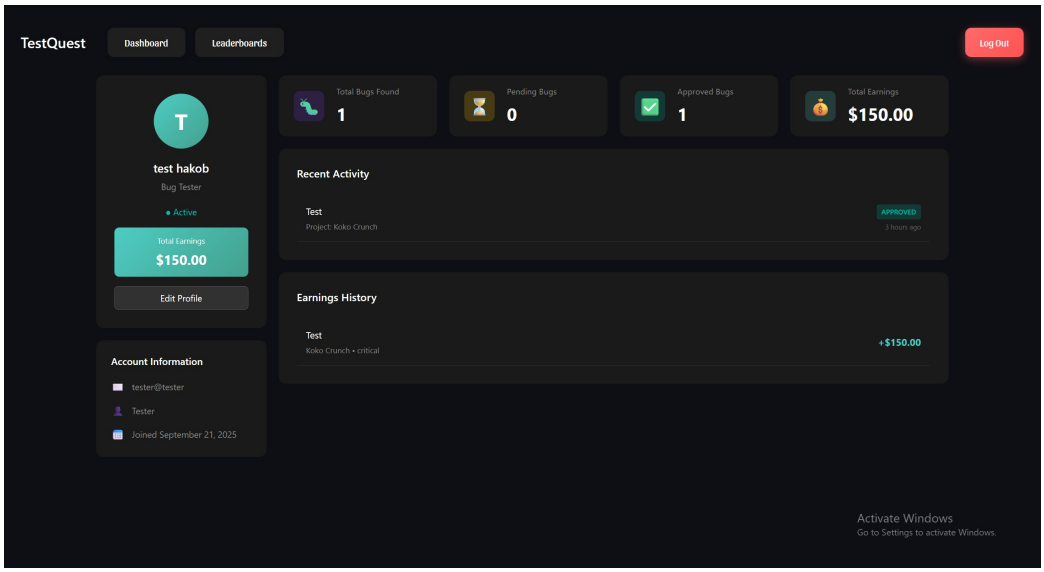
A.2.3 Bug Report Review



A.2.4 Bug Report Screen

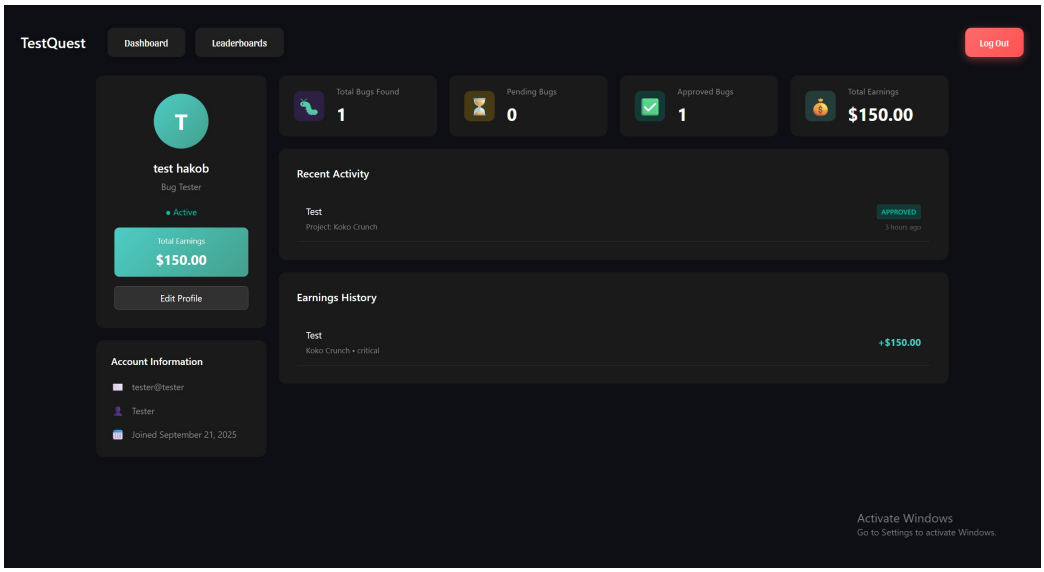


A.2.5 Tester Profile

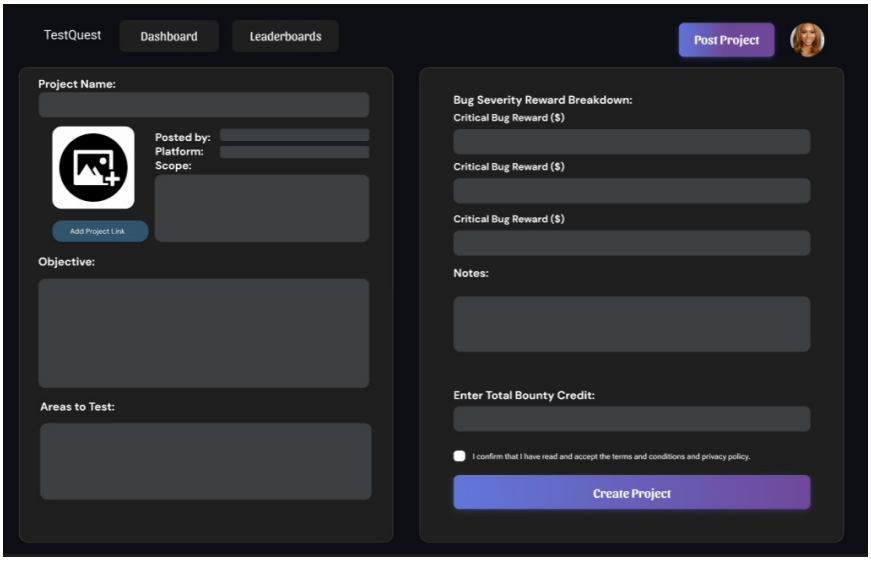


A.3 Developer Screens

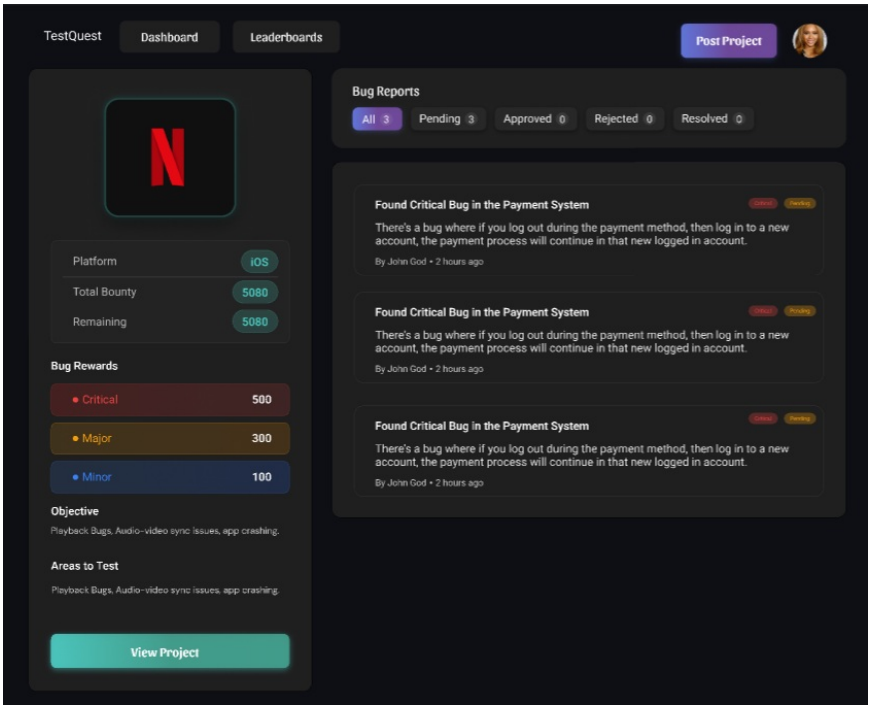
A.3.1 Developer Dashboard



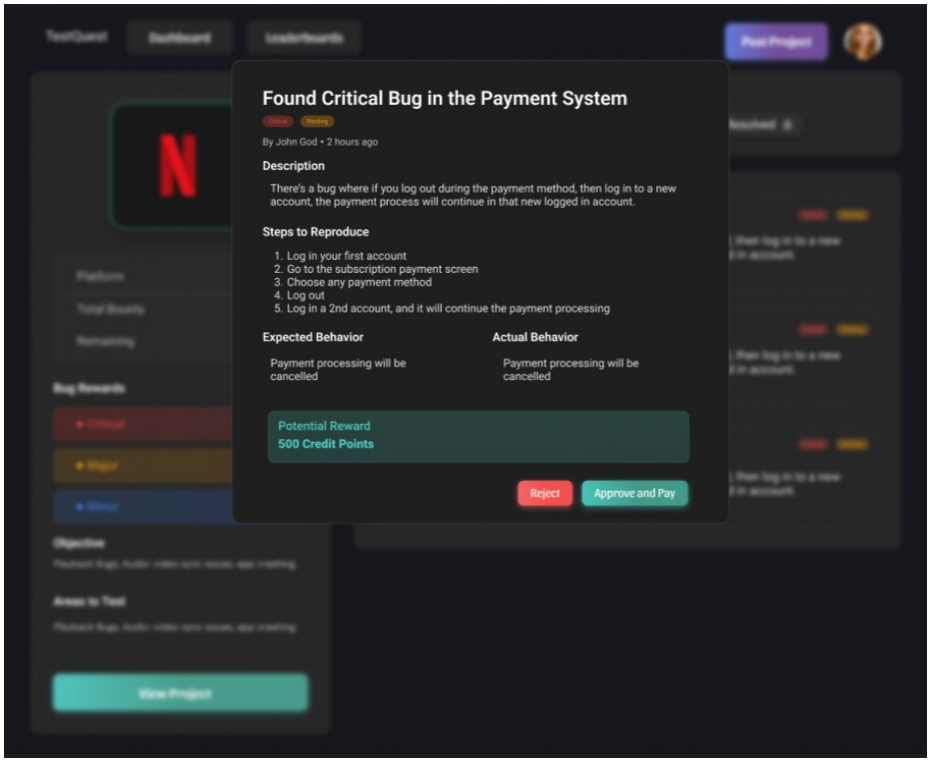
A.3.2 Project Post



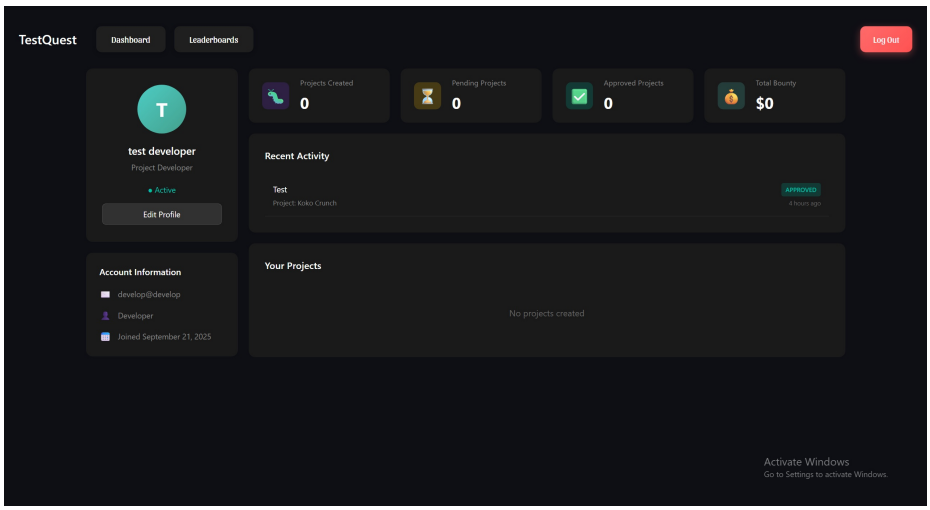
A.3.3 Project View



A.3.4 Project View - Bug Report Pop-up

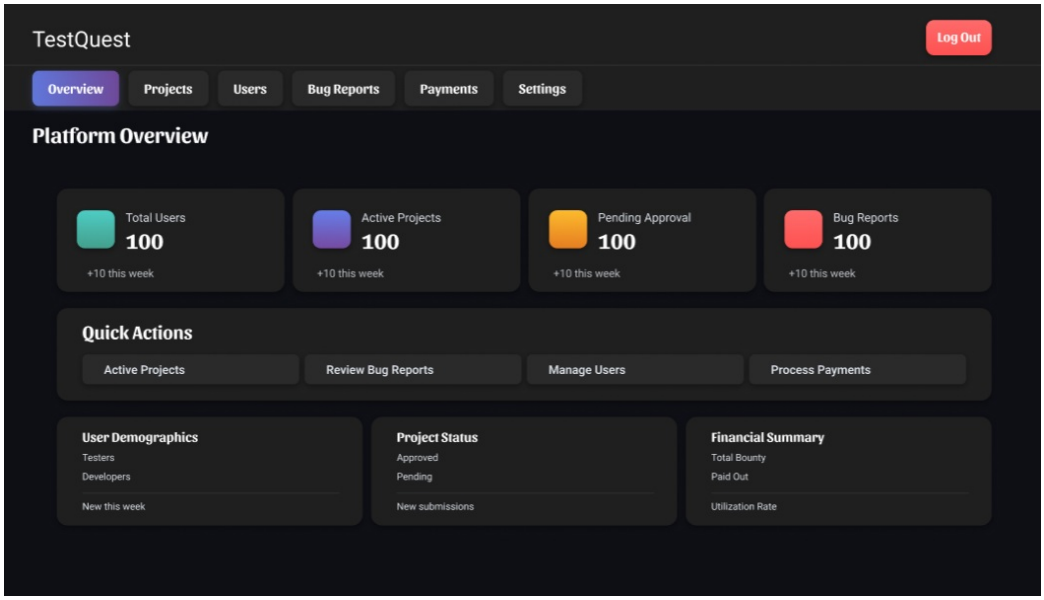


A.3.5 Developer Profile

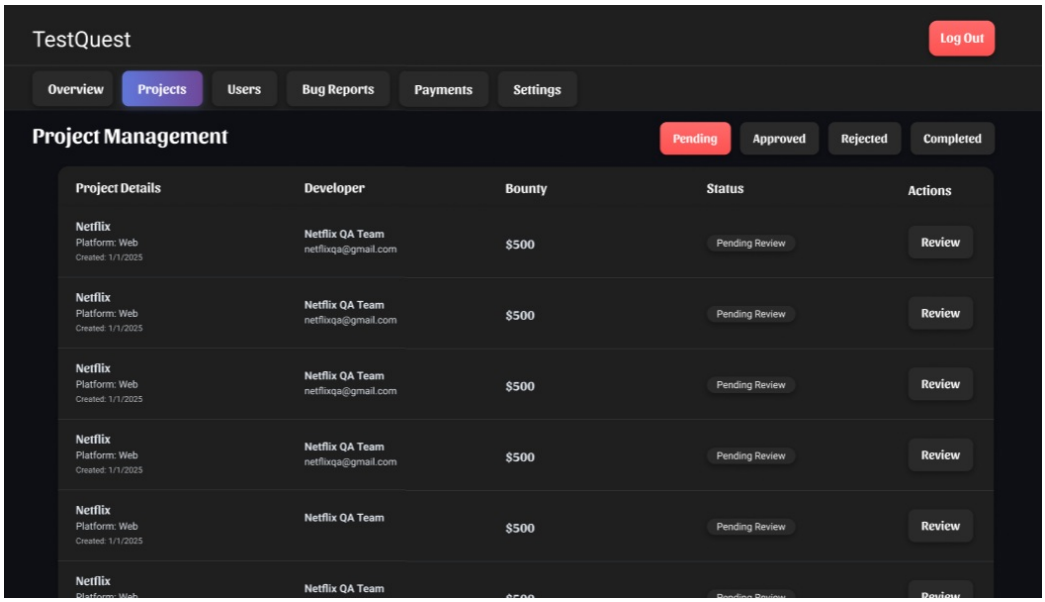


A.4 Admin Screens

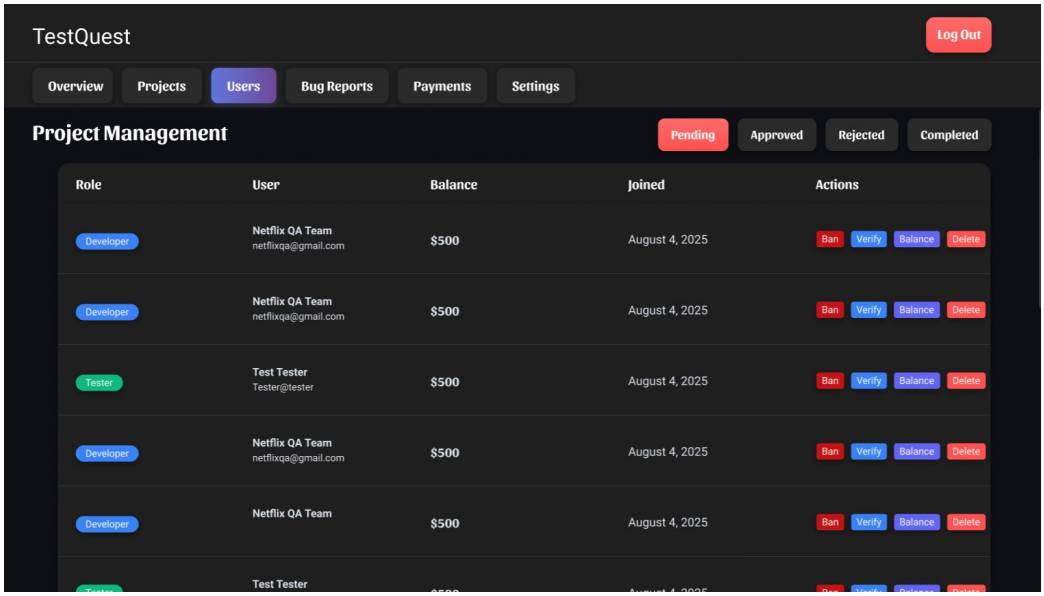
A.4.1 Admin Dashboard



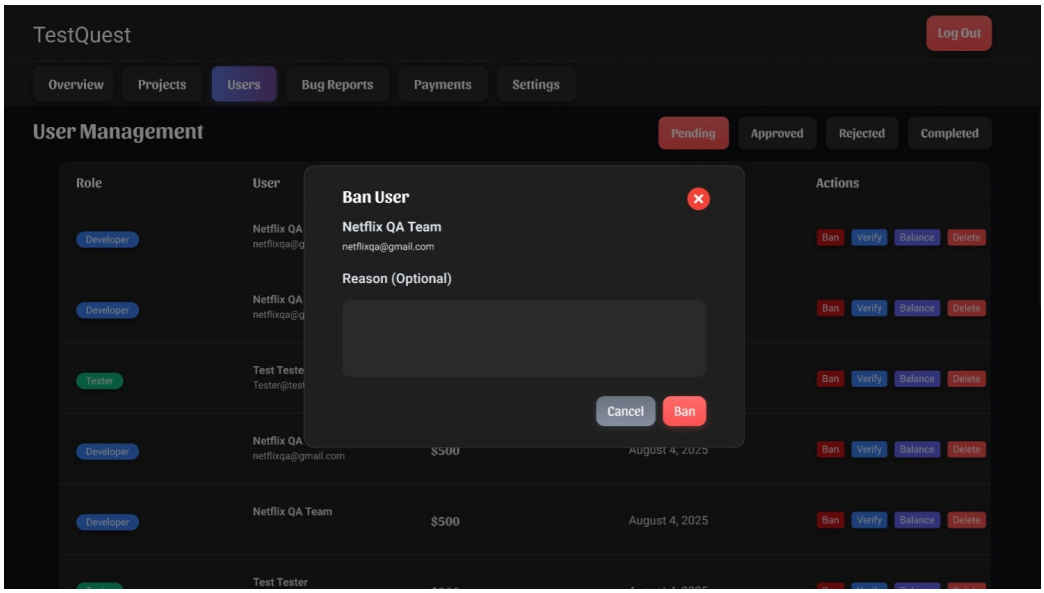
A.4.2 Project Management



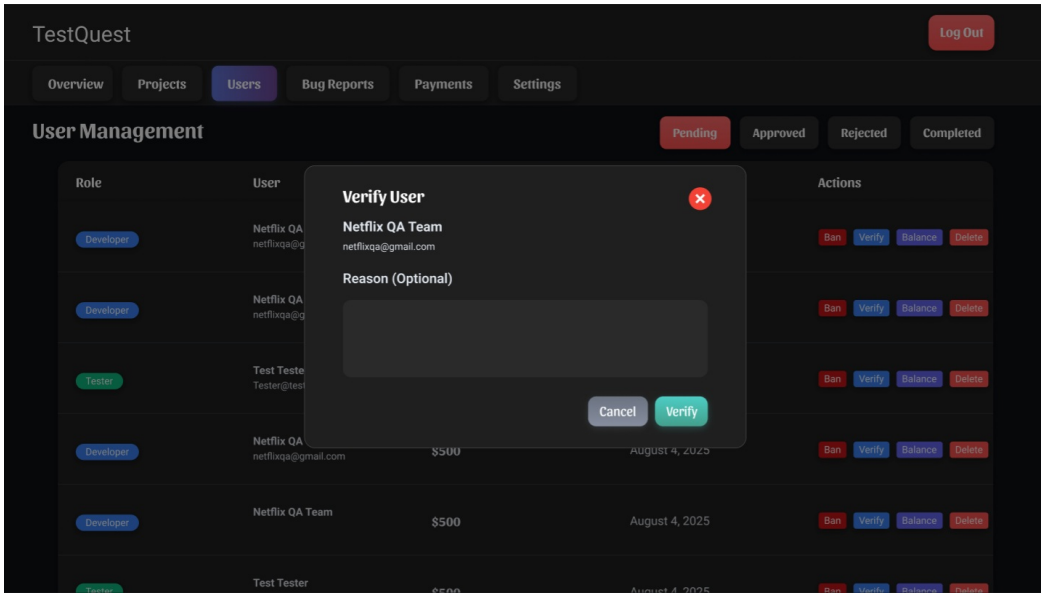
A.4.3 User Management



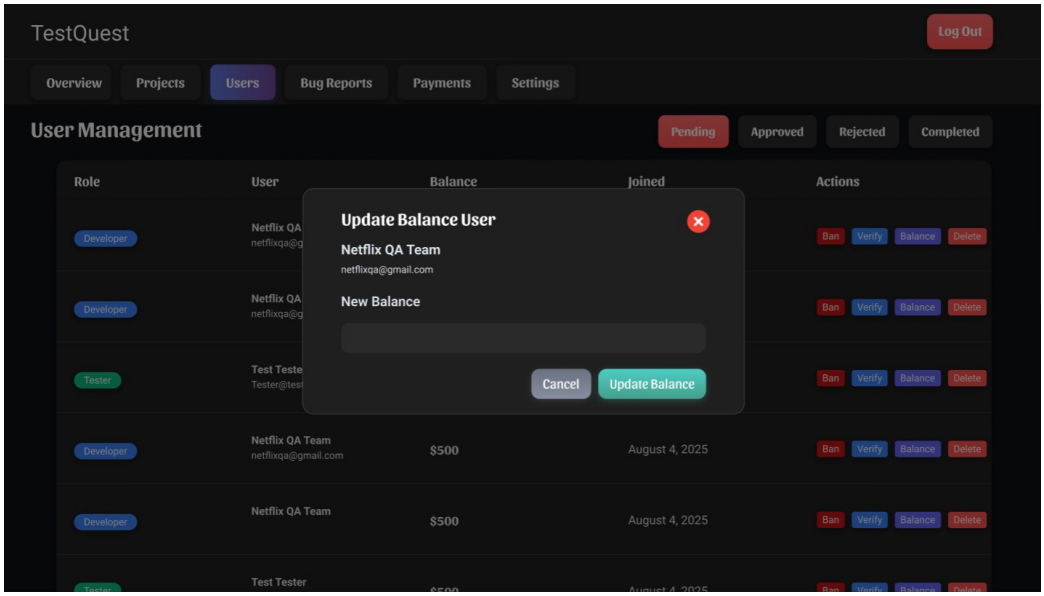
User Management Ban Pop-up



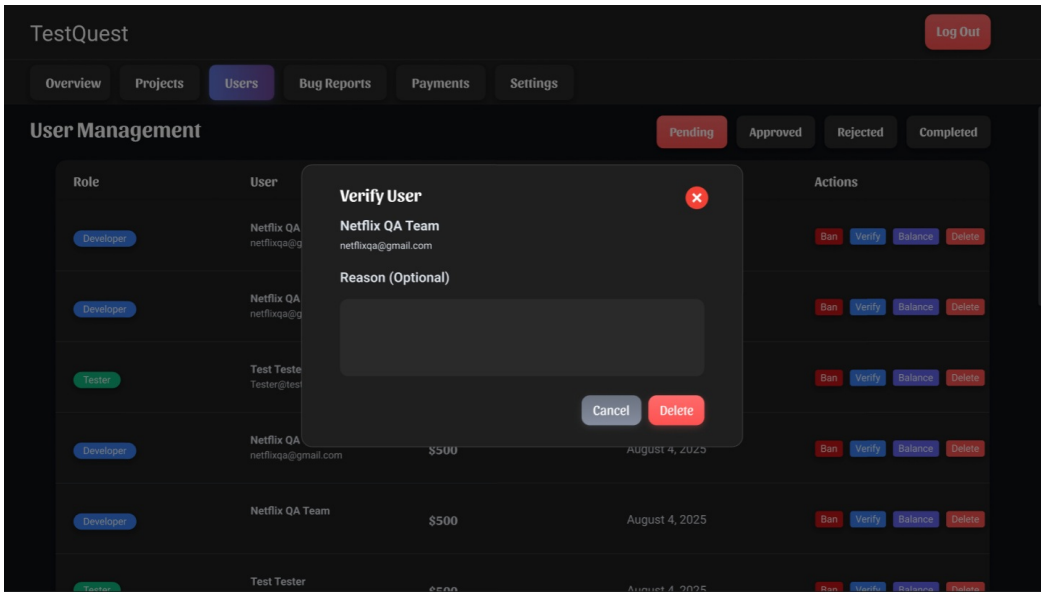
User Management Verify Pop-up



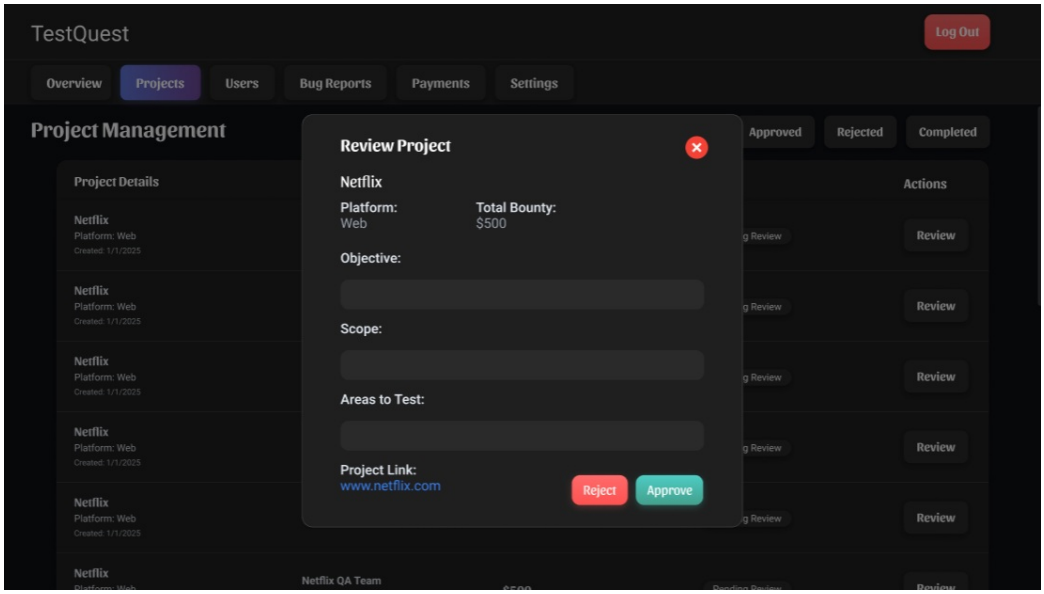
User Management Update Pop-up



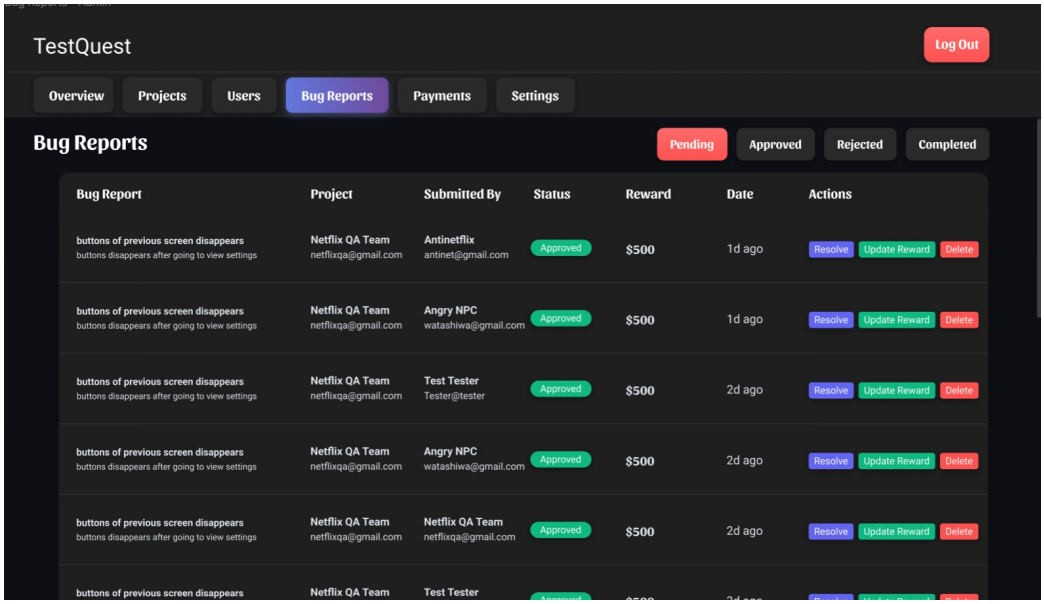
User Management Delete Pop-up



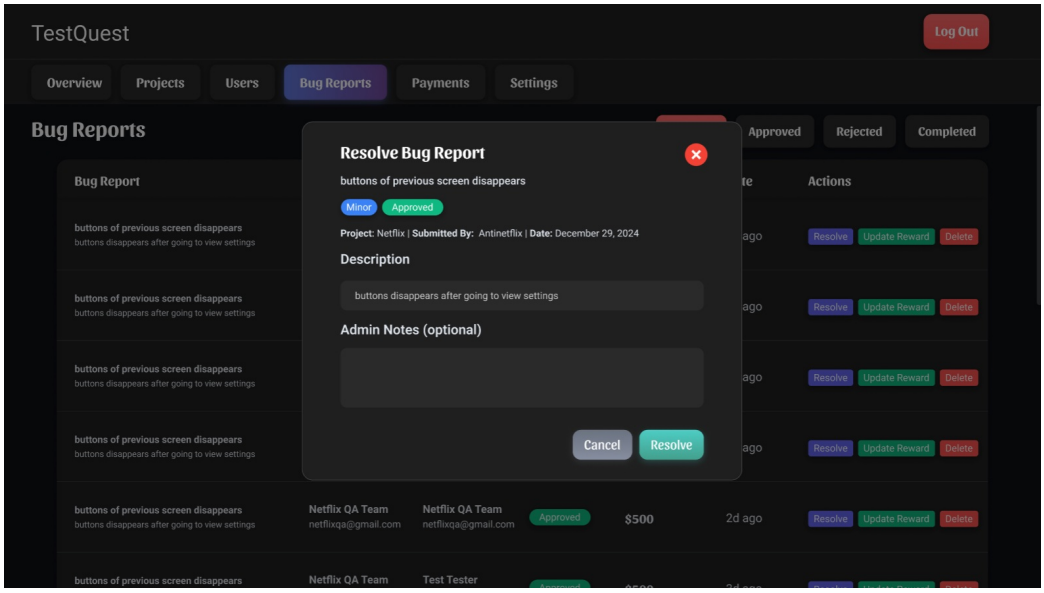
Project Management - Project Review Pop-up



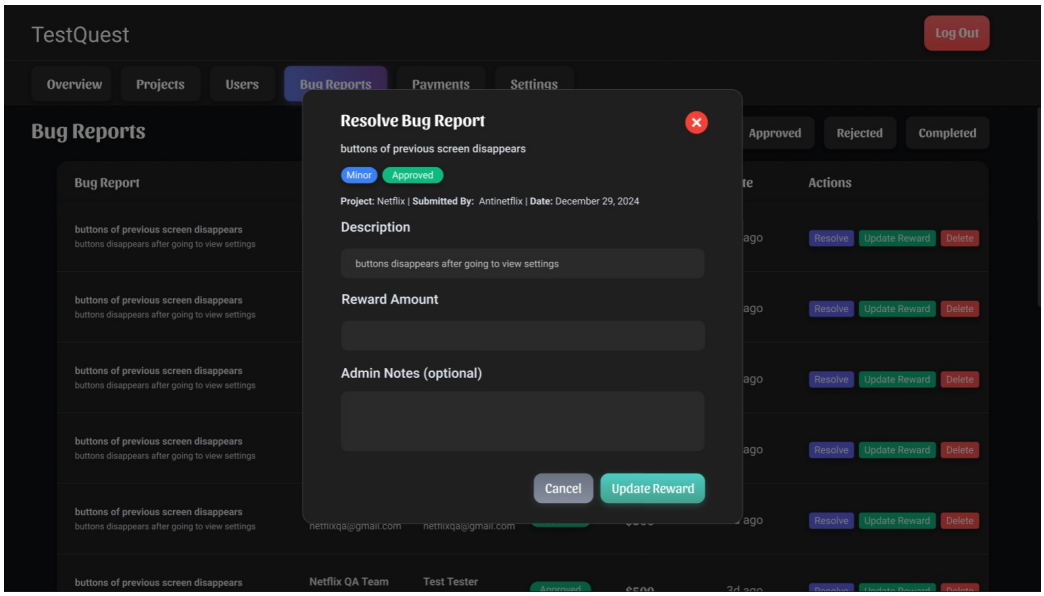
A.4.4 Report Management



Report Management Resolve Pop-up



Report Management Update Pop-up



Appendix B

TestQuest Overview & Policies

The objective is to articulate the understanding of the mission of TestQuest platform and the moderation role in advocating fair collaboration between developers and testers and the basic dispute resolution model integrated into the platform.

Key Policies of the Platform

Neutrality First – Moderators need to be indifferent in each and every case to maintain the platform's integrity. Moderators need to neither favour developers nor testers.

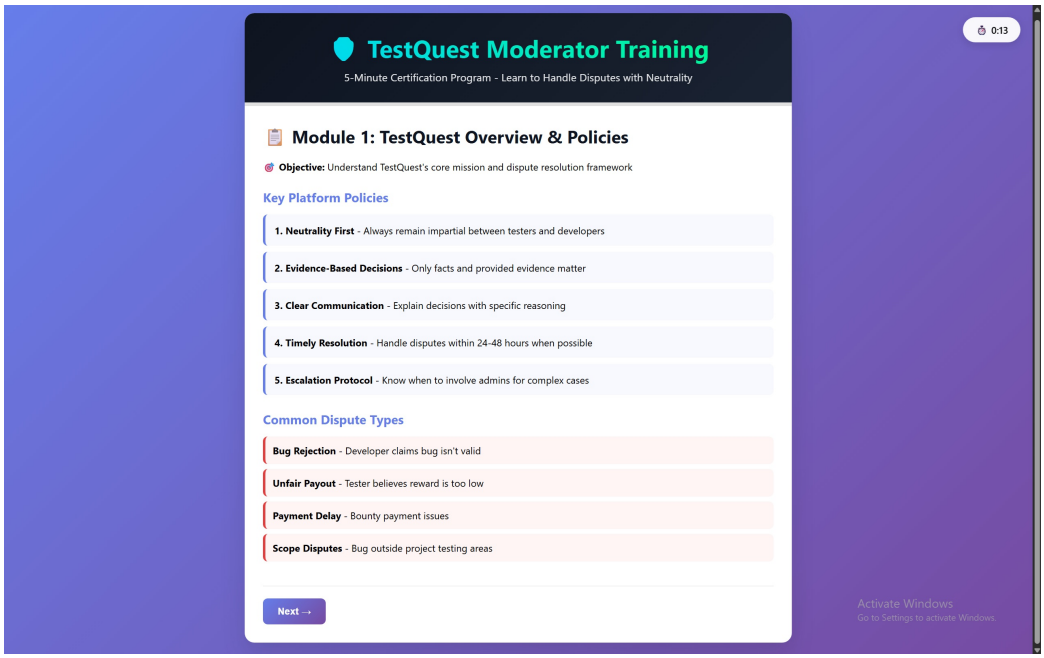
Evidence Based – Each case ought to be decided based on the documentation provided by each party. The final resolution cannot be driven by personal estimations.

Clear Communication – Each decision must be accompanied by a written account detailing each rationale along with supporting data. This is to guarantee accountability and foster trust on the platform by the users.

Timely Resoltion – Each case must be resolved within 24-48 hours. This is to guarantee efficiency in case management and foster trust on the platform.

Proper Escalation – There are cases when the moderators must escalate to the administrator for further review. These are the high value cases, cases involving possible collusion and cases with other signs of moderator misconduct.

B.1 Moderator Training Program



A knowledge training/assessment is part of the moderator training program. This lets us gauge the extent to which the moderators understand the policies of TestQuest and the framework to resolve disputes. The knowledge assessment consists of 15 questions taken from a bank of 25 questions. These questions cover neutrality, evidence-informed decision-making, standards of communication, scenarios of disputes, and other relevant issues. This helps limit rote memorization while fairly testing every moderator across various topics. The system, which requires a moderator to attain a certain score before they can start managing disputes, offers a clear sign of readiness. The format's randomization and retrieval from various topics foster real comprehension of platform rules, which helps to cement the reliability and consistency of moderators in real case resolutions.

REFERENCES

[1] M. Alphonce, "Challenges and solutions in software testing practices: A systematic review in tanzanian software development companies," *SSRN Electronic Journal*, 2023. [Online]. Available: https://doi.org/10.2139/ssrn.4607901

[2] S. Alyahya, "Crowdsourced software testing: A systematic literature review," *Information and Software Technology*, vol. 127, p. 106363, 2020. [Online]. Available: https://doi.org/10.1016/j.infsof.2020.106363

[3] G. Richter, D. R. Raban, and S. Rafaeli, "Studying gamification: The effect of rewards and incentives on motivation," in *Gamification in Education and Business*, T. Reiners and L. C. Wood, Eds. Cham, Switzerland: Springer, 2015, pp. 21–46. [Online]. Available: https://doi.org/10.1007/978-3-319-10208-5_2

[4] R. E. C. Souza, R. E. d. S. Santos, L. F. Capretz, M. A. S. d. Sousa, and C. V. C. d. Magalhães, "Roadblocks to attracting students to software testing careers: Comparisons of replicated studies," in *Quality of Information and Communications Technology (QUATIC 2022)*. Cham, Switzerland: Springer, 2022, pp. 127–139. [Online]. Available: https://doi.org/10.1007/978-3-031-14179-9_9

[5] V. Garousi, A. Rainer, P. Lauvås, and A. Arcuri, "Software-testing education: A systematic literature mapping," *Journal of Systems and Software*, vol. 165, p. 110570, 2020. [Online]. Available: https://doi.org/10.1016/j.jss.2020.110570

[6] D. Journal. (2023) Closing the qa talent gap: Test management tools as a solution. [Online]. Available: https://www.devprojournal.com/software-development-trends/software-testing/closing-the-qa-talent-gap-test-management-tools-as-a-solution/

[7] Testlio. (2023) The qa challenge facing 2023: A talent shortage. Accessed: 2025-09-06. [Online]. Available: https://testlio.com/blog/qa-talent-shortage/

[8] A. Santos, A. Paiva, R. Brito, and J. Alves, "Gamification in software engineering education: a tertiary study," *arXiv preprint arXiv:2405.05209*, 2024. [Online]. Available: https://arxiv.org/html/2405.05209v1

[9] S. Alyahya, "Collaborative crowdsourced software testing," *Electronics*, vol. 11, no. 20, p. 3340, 2022. [Online]. Available: https://doi.org/10.3390/electronics11203340

[10] Q. Wang, J. Wang, M. Li, Y. Wang, and Z. Liu, "A roadmap for software testing in open collaborative development environments," *arXiv preprint*, 2024, arXiv:2406.05438. [Online]. Available: https://arxiv.org/abs/2406.05438

[11] N. Leicht, N. Knop, I. Blohm, C. Müller-Bloch, and J. M. Leimeister, "When is crowdsourcing advantageous? the case of crowdsourced software testing," in *Proceedings of the 24th European Conference on Information Systems*, ser. ECIS '16, 2016, pp. 1–17. [Online]. Available: http://dx.doi.org/10.2139/ssrn.3159166

[12] P. Straubinger and G. Fraser, "Gamekins: Gamifying software testing in jenkins," in *Proceedings of the 44th International Conference on Software Engineering Companion (ICSE '22 Companion)*. Pittsburgh, PA, USA: ACM, 2022, pp. 85–89. [Online]. Available: https://doi.org/10.1145/3510454.3516862

[13] ——, "Gamifying a software testing course with continuous integration," in *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET '24)*. Lisbon, Portugal: ACM, 2024, pp. 1–12. [Online]. Available: https://doi.org/10.1145/3639474.3640054

[14] G. J. Myers, *The Art of Software Testing*, 3rd ed. John Wiley & Sons, 2011.

[15] A. Bertolino, "Software testing research: Achievements, challenges, dreams," in *Proceedings of the 2007 Future of Software Engineering (FOSE '07)*. IEEE Computer Society, 2007, pp. 85–103. [Online]. Available: https://doi.org/10.1109/FOSE.2007.25

[16] E. A. Locke and D. M. Schweiger, "Toward a theory of task motivation and incentives," *Organizational Behavior and Human Performance*, vol. 3, no. 2, pp. 157–189, 1970. [Online]. Available: https://doi.org/10.1016/0030-5073(68)90004-4

[17] D. R. Mavridis and A. Dimitrakopoulou, "Gamification of business processes: Re-designing work in production and service industry," *Procedia Manufacturing*, vol. 3, pp. 3424–3431, 2015. [Online]. Available: https://doi.org/10.1016/j.promfg.2015.07.616

[18] M. Z. U. Arif, "Gamification: A strategic tool for organizational effectiveness," *International Journal of Business and Technopreneurship*, vol. 6, no. 2, pp. 285–302, 2016. [Online]. Available: https://www.researchgate.net/publication/308097517

[19] M. Sailer, J. U. Hense, S. K. Mayr, and H. Mandl, "How gamification motivates: An experimental study of the effects of specific game design elements on psychological need satisfaction," *Computers in Human Behavior*, vol. 69, pp. 371–380, 2017. [Online]. Available: https://doi.org/10.1016/j.chb.2016.12.033

[20] L. Jaramillo-Mediavilla, A. Basantes-Andrade, M. Cabezas-González, and S. Casillas-Martín, "Impact of gamification on motivation and academic performance: A systematic review," *Education Sciences*, vol. 14, no. 6, p. 639, 2024. [Online]. Available: https://doi.org/10.3390/educsci14060639

[21] J. Kaur, R. Lavuri, R. Parida, and S. V. Singh, "Exploring the impact of gamification elements in brand apps on the purchase intention of consumers," *Journal of Global Information Management*, vol. 31, no. 1, pp. 1–30, 2023. [Online]. Available: https://doi.org/10.4018/JGIM.317216

[22] Y. B. Mohammed and F. Ozdamli, "Motivational effects of gamification apps in education: A systematic literature review," *BRAIN. Broad Research in Artificial*

Intelligence and Neuroscience, vol. 12, no. 2, pp. 122–138, 2025. [Online]. Available: https://doi.org/10.18662/brain/12.2/196

[23] HackerOne, "Bug bounty platform," 2025. [Online]. Available: https://www.hackerone.com/product/bug-bounty-platform

[24] N. Bozdemir and C. Collins, "Gain actionable, data-backed insights withhackeronerecommendations," 2025. [Online]. Available: https://www.hackerone.com/blog/gain-actionable-data-backed-insights-hackerone-recommendations

[25] HackerOne, "Hackerone's in-depth approach to vulnerability triage and validation," 2024. [Online]. Available: https://www.hackerone.com/blog/hackerones-depth-approach-vulnerability-triage-and-validation

[26] ——, "Hackerone triage101," 2025. [Online]. Available: https://www.hackerone.com/sites/default/files/2025-02/HackerOne-Triage-101.pdf

[27] Bugcrowd, "The bugcrowd platform overview," 2023. [Online]. Available: https://www.bugcrowd.com/wp-content/uploads/2023/11/Bugcrowd-Platform-Overview-Data-Sheet.pdf

[28] ——, "Crowdmatch ai for better security testing outcomes," 2023. [Online]. Available: https://www.bugcrowd.com/products/platform/crowdmatch/

[29] ——, "Get to know the bugcrowd security knowledge graph," 2024. [Online]. Available: https://www.bugcrowd.com/blog/get-to-know-bugcrowds-security-knowledge-graph/

[30] ——, "Platform integrations: Connect bugcrowd with your devsecops toolchain," 2023. [Online]. Available: https://www.bugcrowd.com/products/platform-integrations/

[31] Cyber Magazine, "The open bug bounty community fixes over $1m," n.d. [Online]. Available: https://cybermagazine.com/articles/the-open-bug-bounty-community-fixes-over-1m

[32] Open Bug Bounty, "About," n.d. [Online]. Available: https://www.openbugbounty.org/about/

[33] P. Dutta, "Open bug bounty: 100,000 fixed vulnerabilities and iso 29147," 2024. [Online]. Available: https://www.techworm.net/2018/02/open-bug-bounty-100000-fixed-vulnerabilities-iso-29147.html

[34] The Budget Diet, "Test em all review," n.d. [Online]. Available: https://www.thebudgetdiet.com/test-em-all-review

[35] Paid From Surveys, "Testem all app review," n.d. [Online]. Available: https://paidfromsurveys.com/testem-all-app-review

[36] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband, "A systematic literature review on agile requirements engineering practices and challenges," *Computers in Human Behavior*, vol. 51, pp. 915–929, 2015. [Online]. Available: https://doi.org/10.1016/j.chb.2014.10.046

[37] C. C. Ekechi, C. D. Okeke, and H. E. Adama, "Enhancing agile product development with scrum methodologies: A detailed exploration of implementation practices and benefits," *Engineering Science & Technology Journal*, vol. 5, no. 5, pp. 1542–1570, 2024. [Online]. Available: https://doi.org/10.51594/estj.v5i5.1108

[38] D. Geszten, B. P. Hámornik, and K. Hercegfi, "Team usability testing: Development and validation of a groupware usability evaluation method," *Cognition, Technology & Work*, vol. 26, no. 3, pp. 487–506, 2024. [Online]. Available: https://doi.org/10.1007/s10111-024-00759-5

[39] N. K. D. Hariyanti, I. F. P. Sudhana, I. Sanjaya, and K. Elfarosa, "Implementation of usability testing in measuring the effectiveness and efficiency of mobile application," in *Proceedings of the 5th International Conference on Applied Science and Technology on Engineering Science (iCAST-ES 2022)*. SciTePress, 2022, pp. 836–842. [Online]. Available: https://doi.org/10.5220/0011892900003575

[40] S. McLeod, "Mixed methods research guide with examples," Global Academic Journal's Research Consortium, June 2024. [Online]. Available: https://doi.org/10.13140/RG.2.2.31329.93286

[41] Testlio. (2023) What is crowdsourced testing? [Online]. Available: https://testlio.com/blog/what-is-crowdsourced-testing/

VITA

Joker P. Panuelos is BS Information Technology student of the Department of Computer Science at the Ateneo de Naga University.

Jacob Andrew V. Masip is BS Information Technology student of the Department of Computer Science at the Ateneo de Naga University.