Test Case Management & Analysis Engine

Group Members

Muhammad Saqib Aziz (01-131192-057)

Zain Ahmed (01-131192-037)

Supervisor

Dr. Tamim Ahmed

A final Year Project submitted to the Department of Software Engineering, Faculty of Engineering Sciences, Bahria University, Islamabad in the partial fulfillment for the award of degree in Bachelors of Software Engineering

June 2023

DECLARATION

Certified that this project report "Test Case Management & Analysis Engine" is the bona fide work of "Muhammad Saqib Aziz and Zain Ahmed" who carried out the project work under my supervision.

_____

(Signature of Supervisor)

THESIS COMPLETION CERTIFICATE

Student Name: Muhammad Saqib Aziz Enrolment No: 01-131192-057
Student Name: Zain Ahmad Enrolment No: 01-131192-037

Programme of Study: Bachelor of Software Engineering
Project Title: Test Case Management & Analysis Engine

It is to certify that the above students' project has been completed to my satisfaction and to my belief, its standard is appropriate for submission for evaluation. I have also conducted plagiarism test of this thesis using HEC prescribed software and found similarity index at **15%** that is within the permissible limit set by the HEC. I have also found the thesis in a format recognized by the department.

Supervisor's Signature: _____

Date: ___02-June02023_____ Name: _____ _____

Bahria University Islamabad
Department of Software Engineering

Dated: 6th June 2023

CERTIFICATE

We accept the work contained in the report titled **Test Cases Management & Analysis Engine** as a confirmation to the required standard for the partial fulfillment of the degree of BSE -8.

Dr. Raja M. Suleman
Project Coordinator
Date: 6/6/2023

Dr. Tamim A. Khan
Supervisor

Date: 6/6/2023

Engr. Bilal A. Awan
Internal Examiner

Date: 6/6/2023

Dr. WaqasAhmad
External Examiner
Date: 6/6/2023

Dr. Awais Majeed
Head of Department (SE)
Date: _____

ACKNOWLEDGEMENT

I would like to take this opportunity to express my sincere gratitude to my supervisor, Dr. Tamim Ahmed Khan, for his invaluable guidance, support, and encouragement throughout this research project. His expertise, insightful feedback, and unwavering dedication have been instrumental in shaping my research and helping me achieve my goals.

Dr. Tamim Ahmed Khan's unwavering commitment to excellence, his depth of knowledge in the field, and his willingness to share his expertise and insights have been an inspiration to me. I have learned a great deal from his guidance, and his mentorship has been instrumental in developing my research skills.

I would like to express my sincere appreciation to Dr. Tamim Ahmed Khan for his support and encouragement throughout this project. Without his guidance, this research would not have been possible.

ABSTRACT

The software testing industry is rapidly growing, with more and more companies relying on the expertise of software testers to ensure the quality of their products. However, there is a lack of platforms that effectively connect software testers with companies in need of their services.

We introduce a web application marketplace that offers remote testing job opportunities to software testers. The platform allows testers to register and work on assigned projects, and they are paid according to their progress. Companies in need of testing services can also register and find suitable testers for their projects.

Front end development is in React.js whereas backend is node.js We used GIT repository for version control and centralized code repository, and implemented CI/CD pipeline using GitHub actions written using YAML. We prepare test cases for the front end using React JS unit testing and integration test cases using Postman. We identified 50 test scenarios for acceptance testing and 101 test cases using Selenium.

The application provides a wide range of services and also offers test case management and traceability metrics to measure the testing process effectiveness. Our developed platform aims to bridge the gap between software companies and testers by providing a centralized marketplace that streamlines the testing process and offers remote job opportunities to testers and can assist in attracting testing resources in an online manner reducing testing costs, improving resulting quality and managing time.

Table of Contents

List of figures

CHAPTER - 1

INTRODUCTION

1 Motivation

The software testing industry is growing rapidly, and there is a need for a platform that effectively connects software testers with companies in need of their services. Currently, there is a lack of centralized marketplaces that provide support to testers and enable effective testing. This project aims to bridge this gap and provide a solution that benefits both software testers and companies.

In this document we will also compare our system with the traditional system. We present the core features of this project and discuss its importance. We also cover the descriptive analysis of these features and their benefits for the organizations and associated stakeholders.

1.1 Problem Statement

In the aftermath of Covid-19 and current prevailing tight financial situations, companies tend to keep physical presence of human resources as less as possible. This leads to either hiring human resources that are working from home of completing depending upon the outsourcing models. We provide a centralized marketplace for software testing services so that the software houses and IT companies find suitable testers for their projects who are working online or from home and providing testing services. The purpose of this project is to develop a web application marketplace that connects software testers with companies in need of their services. The platform aims to offer remote job opportunities to software testers and streamline the testing process for companies, resulting in improved product quality.

1.2 Objectives

The main goal of this project is to develop a web application marketplace that connects software testers with companies in need of their services. The objectives of the project are:

- To provide remote job opportunities to software testers.
- To streamline the testing process for companies by providing a centralized marketplace.
- To enable effective testing through input test data generation, test case management, and traceability metrics.
- To improve the quality of the final product by providing comprehensive testing services.

1.3 Contributions

This project offers the following contributions:

- A centralized marketplace for software testing services that benefits both software testers and companies.
- Remote job opportunities for software testers, enabling them to work from anywhere.
- Support for effective testing through input test data generation, test case management, and traceability metrics.
- Improved product quality through comprehensive testing services.

1.4 Existing System

The existing systems in the software testing industry are limited in their ability to connect software testers with companies that need their services. While there are some job portals and freelancer platforms available, they lack features specifically designed for software testers. Additionally, many of these platforms do not offer support for test data generation, test case management, and traceability metrics, which are critical for effective testing.

1.5 Proposed System

Our proposed web application marketplace offers several unique features that address the existing gaps in the software testing industry.

- **Remote Testing Job Opportunity:** Our platform is designed exclusively for software testers, offering them remote job opportunities that match their skill sets. This specialization ensures that companies can find the right testers for their projects, and testers can find suitable assignments that match their expertise.
- **Input Test Data Generation:** Our platform provides additional support to testers through input test data generation, which enables them to generate relevant and effective test data for their projects. This feature is particularly useful for complex software systems that require a large amount of data to be tested comprehensively.
- **Test Case Management:** Our platform offers test case management and traceability metrics, which provide testers with an efficient and organized way to manage their testing activities. The test case management feature enables testers to create, edit, and manage test cases, while the traceability metrics feature provides an easy way to track the progress and coverage of test cases.

- **Test Case Traceability:** With our platform, both testers and clients have visibility into the mapping of each requirement of the testing project to specific test cases. This traceability ensures that all project requirements are adequately addressed and tested. The clear mapping between requirements and test cases enhances transparency and collaboration between testers and clients.

1.6 Project Execution

We conducted a detailed requirements identification, specification and preparation of design specification of our proposed system. The system architecture of our web application follows a standard Model-View-Controller (MVC) design pattern. In our system, the Model layer is built using MongoDB, a NoSQL where the View layer is developed using React, and the Controller layer is implemented using Node.js. The data layer is built on top of the Node.js runtime environment and uses the Mongoose Object Data Modeling (ODM) library to interact with the database. The processing layer of our system is built using Node.js, a JavaScript runtime built on Chrome's V8 JavaScript engine. It provides an event-driven architecture and non-blocking I/O capabilities, making it a good fit for building scalable and efficient server-side applications. Finally, the representation layer is implemented using the React.JS library, which is a popular JavaScript library for building user interfaces.

We implemented a Continuous Integration/Continuous Deployment (CI/CD) pipeline using GitHub Actions. We defined separate workflows for the frontend and backend components of our project. We automated the deployment process for our MERN stack application. We used React JS unit testing environment for development of unit test cases and we used Postman and Node JS libraries mocha and chai for the integration testing purposes. We developed 47 test cases pertaining to our methods calls in frontend and backend. We prepared test scenarios and conducted system testing creating test cases considering our test scenarios. We developed 101 test cases using Selenium during this process.

1.7 Report Organization

The rest of the document is organized as follows: We present Literature Review in Chapter 2 whereas the Requirements Specification is presented in Chapter 3. We present System Design in Chapter 4 and the Implementation is presented in Chapter 5. Finally, System Testing is presented in Chapter 6 and the Conclusions in Chapter 7 of this document.

CHAPTER - 2

BACKGROUND/ LITERATURE REVIEW

2 BACKGROUND/ LITERATURE REVIEW

2.1 Introduction

Making software that satisfies corporate needs is a problem for the software industry. The fact is we are to deliver software that is free of bugs. If they are not fixed prior to delivery, software problems can result in significant losses for the IT and software organizations. Software testing is a crucial component in creating software that is free of errors and flaws. Software testing is carried out to assist quality control [1].

Despite the growing importance of software testing, there is a significant gap in the market for platforms that effectively connect software testers with companies in need of their services. Many software testing companies struggle to find the right testers with the necessary expertise and skills to ensure high-quality software products. Additionally, companies often face challenges in managing the testing process, tracking progress, and ensuring that their testing efforts align with their business objectives. This gap in the market highlights the need for a centralized platform that streamlines the testing process and connects software testers with companies in need of their services.

2.2 The Importance of Software Testing

Software testing plays a vital role in the process of developing a high-quality software. Testing is necessary because we all make mistakes. Some of those mistakes are unimportant but some of them are expensive and dangerous. Therefore, there is a need to check everything that we produce.[2]

Releasing faulty software can have negative consequences for software companies, including financial losses, damage to reputation, and decreased user satisfaction. For example, a 2018 study by Tricentis found that software failures cost companies an average of $1.7 trillion in lost revenue and productivity worldwide. Another study by IBM found that the cost of fixing defects increases exponentially the later they are detected in the software development life cycle.

In addition to financial losses, releasing faulty software can damage a company's reputation, resulting in loss of trust from users and stakeholders. For instance, the 2019 Boeing 737 Max

crisis, which was caused by a software glitch that resulted in two plane crashes, led to significant financial losses and reputational damage for Boeing.

These negative consequences underscore the importance of software testing in ensuring software quality and preventing costly errors. Effective software testing can identify and address potential issues before they become larger problems, saving companies time and resources.

Furthermore, numerous studies demonstrate the importance of software testing in ensuring software quality. For example, a 2017 report by Cap Gemini found that over 50% of software defects could be prevented through proper testing. Additionally, a study by NIST found that the cost of fixing software defects increased significantly when they were detected later in the software development life cycle, underscoring the importance of testing throughout the development process.

2.3 Remote Testing

Remote testing is a type of software testing that allows testers to work on software systems from anywhere in the world, without being physically present in the same location as the software development team. This differs from traditional testing methods, where testers are typically co-located with the development team and testing is done on premise.

Advantages of Remote Testing

Remote testing offers several advantages over traditional testing methods, including:

On-Demand Testing:

Flexibility: Remote testing allows testers to work from anywhere, at any time, providing more flexibility and convenience than traditional testing methods. This can result in increased productivity and faster turnaround times for testing tasks.

Cost Savings: Remote testing can also result in cost savings, as companies do not need to provide physical office space, equipment, or other resources to testers. Additionally, remote testing can reduce travel expenses for both testers and development teams.

2.4 Competing Systems

There are several online marketplaces for software developers like Fiverr, upWork and Toptal etc, but particularly for software testers, the options are limited. Some of them include:

Testlio: Testlio is a remote testing platform that provides a network of professional testers to help companies test their software products. Testers are vetted and trained to ensure high-quality testing, and the platform offers features such as bug tracking and test case management.

Rainforest QA: Rainforest QA is a remote testing platform that leverages crowdtesting to provide on-demand testing services. The platform allows companies to submit testing tasks and receive results within hours, leveraging a global network of testers.

uTest: uTest is a remote testing platform that provides a community of professional testers to help companies test their software products. The platform offers a range of testing services, including functional testing, usability testing, and security testing.

2.5 Limitations in Existing Systems

Following are the limitations and shortcomings of the existing systems in the software testing marketplace. Understanding these limitations is crucial in identifying opportunities for improvement and differentiation in our platform.

Lack of Test Case Traceability

One of the significant limitations of the existing systems is the absence of comprehensive test case traceability. Testers and clients often struggle to track the mapping of requirements to specific test cases, resulting in a lack of transparency and accountability in the testing process.

Lack of Test Case Proof of Concept

A significant limitation is the absence of a feature where testers can provide proof, such as pictures, videos, or other forms of evidence, to validate the execution and outcomes of test cases. This limitation affects the transparency, credibility, and clarity of the testing process for both testers and clients.

Inadequate Input Test Data Generation

Many existing systems do not provide robust support for input test data generation. This limitation can lead to a lack of comprehensive and diverse test scenarios, potentially impacting the thoroughness and effectiveness of the testing process.

2.6 Proposed Solution

We overcame some of the issues highlighted in existing systems and our platform differs from those in several ways.

Test Case Traceability

With our platform, both testers and clients have visibility into the mapping of each requirement of the testing project to specific test cases. This traceability ensures that all project requirements are adequately addressed and tested. The clear mapping between requirements and test cases enhances transparency and collaboration between testers and clients.

Annotated Test Case Outputs

Our platform generates test case outputs in a format that can be utilized by AI systems. This feature enables the generation of structured data sets and facilitates classification tasks. By leveraging AI systems, clients can further analyze and interpret the test results, leading to more efficient and accurate decision-making in the software testing process.

Remote Testing Focus

We exclusively focus on remote testing, expanding the pool of qualified testers available to companies seeking testing services. By embracing remote testing, our platform provides flexibility and accessibility to both testers and clients. Testers can work from anywhere, while clients can tap into a global network of skilled testers, thereby increasing the chances of finding the right expertise for their projects.

Test Case Management

Our platform offers robust test case management capabilities. Testers and clients can efficiently create, assign, and track test cases throughout the testing process. The comprehensive test case management system ensures that all necessary tests are executed, progress is monitored, and potential issues are identified and addressed promptly.

Test Case Proof of Concept

In our platform, every time a tester logs a test case, they provide proof in the form of supporting materials such as pictures, videos, or any other relevant evidence to confirm the validity of that test case. This feature enhances the transparency and credibility of the testing process by providing concrete evidence of the tests performed and their outcomes.

By allowing testers to attach visual or multimedia proof to each test case, clients can gain a deeper understanding of the testing process and validate the results. This documentation serves as a valuable resource for stakeholders, aiding in bug identification, analysis, and resolution.

The inclusion of a Test Case Proof of Concept feature adds an extra layer of assurance and confidence in the testing process, enabling both testers and clients to have a comprehensive overview and verification of the executed test cases.

2.7 Conclusion

Our proposed web application marketplace is designed to revolutionize the software testing industry by exclusively offering remote testing job opportunities to software testers. By leveraging the advantages of remote testing, we aim to provide a wider pool of qualified testers to companies seeking their services, ensuring enhanced flexibility and accessibility. The platform's key features include streamlined test case management and traceability, facilitating efficient collaboration and fostering accountability between testers and clients. Our primary objective is to improve the overall quality of the software testing process by ensuring thorough testing and delivering high-quality software products to companies. Additionally, by offering remote job opportunities, our platform provides testers with increased flexibility, enabling them to work on projects from anywhere. This not only meets the market demand for remote testing but also offers a seamless experience for both testers and clients. Through improved communication, better project understanding, and effective decision-making, our solution aims to connect testers with companies, elevate software quality, and create valuable job opportunities in the software testing industry.

CHAPTER - 3

SYSTEM REQUIREMENT SPECFICATION

3 SYSTEM REQUIREMENTS

We present functional and non-functional requirement in this section.

3.1 Functional Requirements

3.1.1 Functional Requirements #1: User Registration

User will be able to make his account on the platform in order to access the features provided.

Output:

The User credentials stored in database

Error:

The User Already exists.

3.1.2 Functional Requirements #2: User Login

User will be able to login to his account so that he can perform the operations related to his position.

Output:

User navigated to Dashboard/home screen.

Error:

Invalid Credentials.

3.1.3 Functional Requirements #3 Request Project

Tester will be able to make a request for a project from a list of projects maintained by Admin.

Output:

Project is marked requested in database

Error:

The project does not exist in Database.

3.1.4 Functional Requirements #4 Monitor Progress

Admin, tester and Clients will be able to see the progress of the project/system under test. The progress may include test suites, their results and bugs encountered etc.

Output:

Progress of the project is shown in the form of tables, charts and PDF report.

Error:

No project is under progress.

3.1.5 Functional Requirements #5 Provide Project Artifacts

Client will provide Project Artifacts to Admin that will be used to test the system accordingly. The artifacts may include executables, links, and documentation etc.

Output:

The artifact is stored in the files of the project.

Error:

User selects an unsupported file from device.

3.1.6 Functional Requirements#6 Sign NDA

Tester will sign an NDA while he is registering for the platform.

Output:

Tester can now officially request for project testing.

Error:

Tester did not agree to the terms and conditions.

3.1.7 Functional Requirements #7 Manage Project Requests

Admin will get requests made by testers for project allocation and clients for project initiation. Admin will accept or reject the requests.

Output:

Project Request accepted or rejected.

Error:

The project doesn't exist.

3.1.8 Functional Requirements #8 Request Project Initiation

Clients will ask admin to initiate a project. They will provide all the necessary information like project name, start date, end date and budget etc.

Output:

Project initiation request saved in Database.

Error:

If Client left a field empty.

3.1.9 Functional Requirements #9 Make Payment

Admin will get payments from clients and make payments to testers.

Output:

Payment released.

Error:

Network issues.

3.2 Use Cases

Use Case Diagram

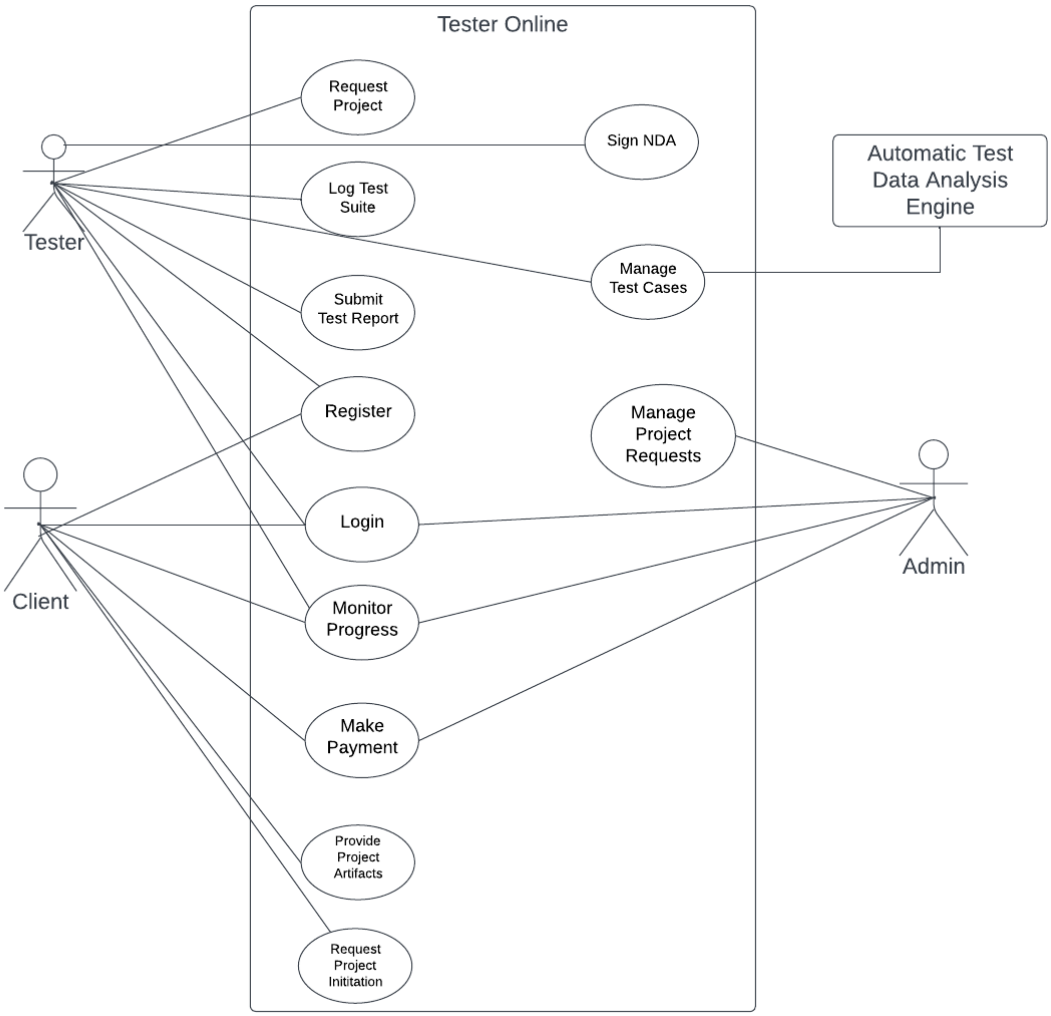

Figure 3-1 use case diagram

3.3 Use Case Descriptions

The use case descriptions for the use cases given in Figure 1 use case diagram are presented in this section

3.3.1 User Management

3.3.1.1 User Registration

Name	Register
Description	Users who do not have an existing account will be able to create a new account to access the platform's features.
Actors	Tester, Client
Priority	Medium
Pre-condition	Not an existing user
Post-condition	User credentials stored to database
Basic Flow	User can register account if he is not an existing user.
Actor Action	**System Response**
	1. System shows registration page
2. User Enters credentials 3. User presses Register Button	4. Data saved into database
Alternate Course of Action	
Actor Action	**System Response**
	4.a If user is an existing user, error message is shown and user stays at registration page (continue at step1). **4.b** If any field is left empty, error message is shown and user stays at registration page (continue at step1).

3.3.1.2 User Login

Name	Login
Description	User will be able to login to his account so that he can perform the operations related to his position.
Actors	Tester, Admin, Client
Priority	medium
Pre-condition	Valid User
Post-condition	User navigated to Dashboard/home screen.
Basic Flow	Authenticated user is able to login to his account.
Actor Action	**System Response**
	1. System shows Login page
2. User enters his credentials 3. User Presses Login Button	4. System redirects user to home screen/dashboard.
Alternate Course of Actions	
Actor Action	**System Response**
3.a User entered invalid credentials	3b. "Invalid User" Error is shown and user stays at login screen (continue at step 1).

3.3.2 Project Management

3.3.2.1 Project Request

Name	Request Project
Description	Tester will be able to make a request for a project from a list of projects maintained by Admin.
Actors	Tester
Priority	medium
Pre-condition	• Tester is signed in • Projects are available
Post-condition	Project is marked requested in database
Basic Flow	Tester makes project request if project is available.
Actor Action	**System Response**
1. Tester selects Available Projects option from Home Screen	2. Tester is navigated to Available Projects Screen and list of Projects available is Shown
3. Tester presses request project button for a particular project	4. Project marked requested in database
Alternate Course of Actions	
Actor Action	**System Response**
	2.a If no project is available, nothing is shown
	3.a If tester doesn't press any button, nothing happens (continue at step 2)

3.3.2.2 Monitor Progress

Name	Monitor Progress
Description	Admin, tester and Clients will be able to see the progress of the project/system under test. The progress may include test suites, their results and bugs encountered etc.
Actors	Admin, Client, Tester
Priority	High
Pre-condition	• Project is under progress • Tester has submitted test suites
Post-condition	Progress of the project is shown
Basic Flow	User can monitor progress of the project if the project is under progress.
Actor Action	**System Response**
1. User selects Monitor Progress option from Home Screen	2. User is navigated to Project Progress Screen 3. Data/Progress from database is shown
Alternate Courses	
Actor Action	**System Response**
	2.a If user doesn't have a project under progress, "No Project Available" Error is shown and user stays at home screen (continue at step1) 3.a If Tester has not submitted test suites, a message is shown "no record available" and user stays at same screen (continue at step2).

3.3.2.3 Providing Project Artifacts

Name	Provide Project Artifacts
Description	Client will provide Project Artifacts to Admin that will be used to test the system accordingly. The artifacts may include executables, links, and documentation etc.
Actors	Client
Priority	Medium
Pre-condition	NDA is signed
Post-condition	Admin will enlist the project in database under category of available projects.
Basic Flow	Client comes to chat box and provides project artifacts to the admin.
Actor Action	**System Response**
1. User selects send artifacts option from Dashboard	2. User is navigated to send artifacts form.
3. User selects clip icon (to send project files) 4. User selects files from Device. 5. User presses send button.	6. Project Files/Artifacts are sent to Admin.
Alternate Course of Actions	
Actor Action	**System Response**
4a. User selects an unsupported file from device.	4b. “File not Supported” Error shown, continue from 3.

3.3.2.4 Sign NDA

Name	Sign NDA
Description	Tester will sign an NDA to be able to request project.
Actors	Tester
Priority	Medium
Pre-condition	none
Post-condition	NDA is signed
Basic Flow	User can tick the sign NDA form
Actor Action	**System Response**
1. User presses Sign NDA Button from Dashboard	2. A form Opens with terms and Conditions
3. User ticks I agree check box 4. User clicks submit button	5. NDA signed Message shown
Alternate Course of Action	
Actor Action	**System Response**
3.a User does not tick check box	4a. Submit Button is not clickable.

3.3.2.5 Manage Project Requests

Name	Manage Project Request
Description	Admin will get requests made by testers for project allocation and clients for project initiation. Admin will accept or reject the requests.
Actors	Admin
Priority	Medium
Pre-condition	Project is available to be tested
Post-condition	Project Request accepted or rejected
Basic Flow	Admin accepts or rejects the project requests.
Actor Action	**System Response**
1. Admin selects Project Requests option from home screen	2. Admin is navigated to Project Requests Screen
	3. A list of project requests along with two buttons; accept and reject is shown.
4. Admin presses accepts or rejects button.	5. The project is marked accepted or rejected in database
Alternate Course of Action	
Actor Action	**System Response**
	3.a If there is no project request, “No Requests” message is shown.
4.a Admin presses no button	**5.a** Admin stays on the same screen (continue at sep3).

3.3.2.6 Request Project Initiation

Name	Request Project Initiation
Description	Clients will ask admin to initiate a project. They will provide all the necessary information like project name, start date, end date and budget etc.
Actors	Client
Priority	medium
Pre-condition	Valid User
Post-condition	Project initiation request saved in Database.
Basic Flow	Client makes project request.
Actor Action	**System Response**
1. Client selects request project from Dashboard	2. Client is navigated to request project form.
3. Client enters information about project 4. Client Enters Request Project Button	5. Project initiation request saved in Database.
Alternate Course of Actions	
Actor Action	**System Response**
	3.a If Client left any field empty, Request Project Button is not pressable. 4a. If Client does not press Request Project Button, nothing happens.

3.3.2.7 Make Payment (Admin to Tester)

Name	Make Payment
Description	Admin will make payments to the testers according to their contributions in the testing process.
Actors	Admin
Priority	Low
Pre-condition	• Project test report has been submitted • Payment is received from client.
Post-condition	Payment is made to testers.
Basic Flow	Admin makes payment to tester.
Actor Action	**System Response**
1. Admin Selects Payment Option from Dashboard	2. Admin is navigated to Payment Screen
	3. A list of pending payments is shown.
4. Admin presses release payment button against tester.	5. The payment is made to tester. 6. The transaction is saved in database.
Alternate Course of Action	
Actor Action	**System Response**
	3.a If no tester has payment in pending, "No payment in Pending" message is shown. **4.a** If admin doesn't press release payment button, nothing happens and admin stays on the same screen (continue at step3). 4b. If there is a network problem, "Network issue" message is shown.

3.3.2.8 Make Payment (Client to Admin)

Name	Make Payment
Description	Client will make payment to the admin after the project is tested and the test report is received.
Actors	Client
Priority	Low
Pre-condition	Project test report has been submitted
Post-condition	Payment is made to the Admin.
Basic Flow	Client makes payment to Admin.
Actor Action	**System Response**
1. Client Selects Payment Option from Dashboard	2. Client is navigated to Payment Screen
	3. Pending payments are shown.
4. Client presses release payment button against tester.	5. The payment is made to Admin. 6. The transaction is saved in database.
Alternate Course of Action	
Actor Action	**System Response**
	3.a If there is no payment in pending, "No payment in Pending" message is shown. **4.a** If client doesn't press release payment button, nothing happens and client stays on the same screen (continue at step3). 4b. If there is a network problem, "Network issue" message is shown.

3.3.3 Test Case Management and Traceability

3.3.3.1 Log Test Suite

Name	Log Test Suite
Description	Tester will be able to log test suites and the results of those test suites into the database. That data will be used to produce test report at the end of testing activity.
Actors	Tester
Priority	High
Pre-condition	Tester is assigned a project.
Post-condition	Test suit and result is stored into database.
Basic Flow	Tester logs test suites and results into database if he has a project assigned.
Actor Action	**System Response**
1. Tester selects Log test suite option from dashboard.	2. Tester is navigated to log test suite screen
3. Tester enters test suite 4. Tester presses the log button	5. Test suite saved to database
Alternate Course of Actions	
Actor Action	**System Response**
	2.a If Tester is not already assigned a project, error message is shown and tester stays at home screen. **2.b** If tester is assigned more than 1 project, list of projects is shown and tester can select 1 (continue at step3)
	5.a If any field is left empty, an error message is shown (continue at step2).

3.3.3.2 Test Report Submission

Name	Submit Test Report
Description	Tester will be able to submit test report after testing the system under test. The report would include test suites and their results, along with bugs encountered.
Actors	Tester
Priority	High
Pre-condition	Tester is assigned a project
Post-condition	• Test report is stored into database • Admin and client can see the report from database
Basic Flow	Tester submits test report to database if he is assigned a project.
Actor Action	**System Response**
1. Tester selects Submit Test report option from dashboard	2. Tester is navigated to Submit Test Report screen
3. Tester Selects File to submit 4. Tester presses the submit button	5. Test Report is saved to database
Alternate Course of Actions	
Actor Action	**System Response**
	2.a If Tester is not already assigned a project, error message is shown and tester stays at home screen. **2.b** If tester is assigned more than 1 project, list of projects is shown and tester can select 1 (continue at step3).
	5.a If any field is left empty, an error message is shown (continue at step3).

3.3.3.3 Manage Test Cases

Name	Manage Test Cases
Description	Tester will be able to manage test cases. Tester can perform CRUD operations on test cases.
Actors	Tester
Priority	High
Pre-condition	At least 1 project is under progress
Post-condition	Operation is performed on test cases.
Basic Flow	tester performs CRUD operations on test cases.
Actor Action	**System Response**
1. Tester selects Manage Test Cases option from home screen	2. tester is navigated to Manage Test Cases Screen 3. Traceability matrices containing test cases is shown
4. tester can perform CRUD operations 5. Admin presses Update Button	6. Test cases are updated in database
Alternate Course of Action	
Actor Action	**System Response**
	3a. If no project is under progress, tester is only shown a message “No Project Under Progress” and he stays on the same screen. **3b.** If there are more than 1 project under progress, list of projects is shown and tester can select 1 (continue at step3). **6a.** If tester doesn’t press update button, no change happens.

3.4 Non-Functional Requirements

3.4.1 Performance Requirements

- System response should be fast.
- Execution time should be good for report generation.
- Storage Capacity should be enough to store test suites and reports.

3.4.2 Safety Requirements

- Payment module should be secure to make safe transactions.
- NDA is signed to make sure non-disclosure of the project artifacts.

3.4.3 Security Requirements

- Authorized Access to Users.
- Each User performs their own functionalities assigned.
- Payment transactions are secured by Banks.

3.4.4 Software Quality Attributes

Requirement #1:

Title: Robustness Description:

The system should handle exceptions when the function / program fails.

Requirement #2:

Title: Usability Description:

The interaction of the system should be user friendly or more attractive.

• Requirement #3:

Title: Performance Description:

The system response time should be less than 3 seconds, so that the reports should be generated quickly as per need.

• Requirement #4:

Title: System Reliability Description:

The system should perform exact functionalities all the time.

• Requirement #5:

Title: Application extensibility Description:

The application should be easy to extend. The code should be written in a way that it favors implementation of new functions.

• **Requirement #6:**

Title: Application testability Description:

Test environments should be built for the application to allow testing of the application's different functions.

• **Requirement #7:**

Title: Application Portability Description:

The application should be portable with iOS and android. It should support all types of web browsers.

• **Requirement #8:**

Title: Communication Security Description:

Security of the communication between the system and server.

• Requirement #9: Title: Internet Connection Description:

The application should be connected to the Internet.

• **Requirement #10:**

Title: System Availability Description:

The availability of the system when it is used

3.5 Resource Requirement

3.5.1 Client End

A web browser is required for the computer system to run this project. The system must have Operating system running on it.

CHAPTER - 4

SYSTEM DESIGN

4 SYSTEM DESIGN

4.1 Sequence Diagrams

In the design phase, we have employed Sequence Diagrams to illustrate the flow of interactions at the System Design Specification (SDS) level. These diagrams demonstrate the various use cases identified in the initial Use Case Diagram.

To ensure a well-structured design, we have incorporated the Model-View-Controller (MVC) architectural pattern in the Sequence Diagrams. This pattern separates the application logic into three components: the Model, View, and Controller.

The Model represents the data and business logic of the application. It stores and manages the data, ensuring consistency and integrity throughout the system.

The View represents the user interface through which users interact with the application. It presents the data to the user and captures user input.

The Controller, as the central component, takes responsibility for handling user input and coordinating the interactions between the View and Model. It receives input from the user (View), queries the Model for necessary data or updates, and updates the View accordingly. This ensures a clear separation of concerns and promotes maintainability and flexibility in the design.

By employing the MVC pattern in our Sequence Diagrams, we have established a structured and organized approach to handle user interactions, data retrieval, and updates. This design promotes modularity, extensibility, and reusability, enhancing the overall quality and maintainability of the software system. This is presented in Figure 2-14

4.1.1 Login

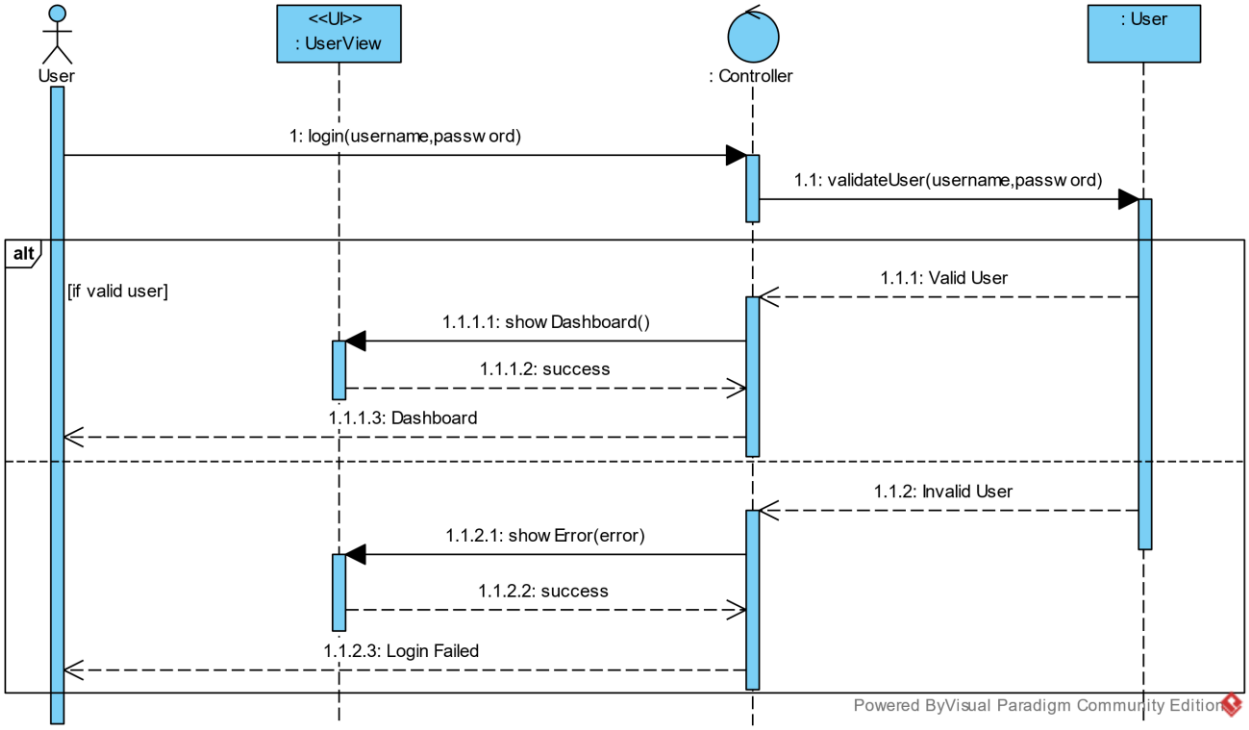

Figure 4-1 login sequence

4.1.2 Register

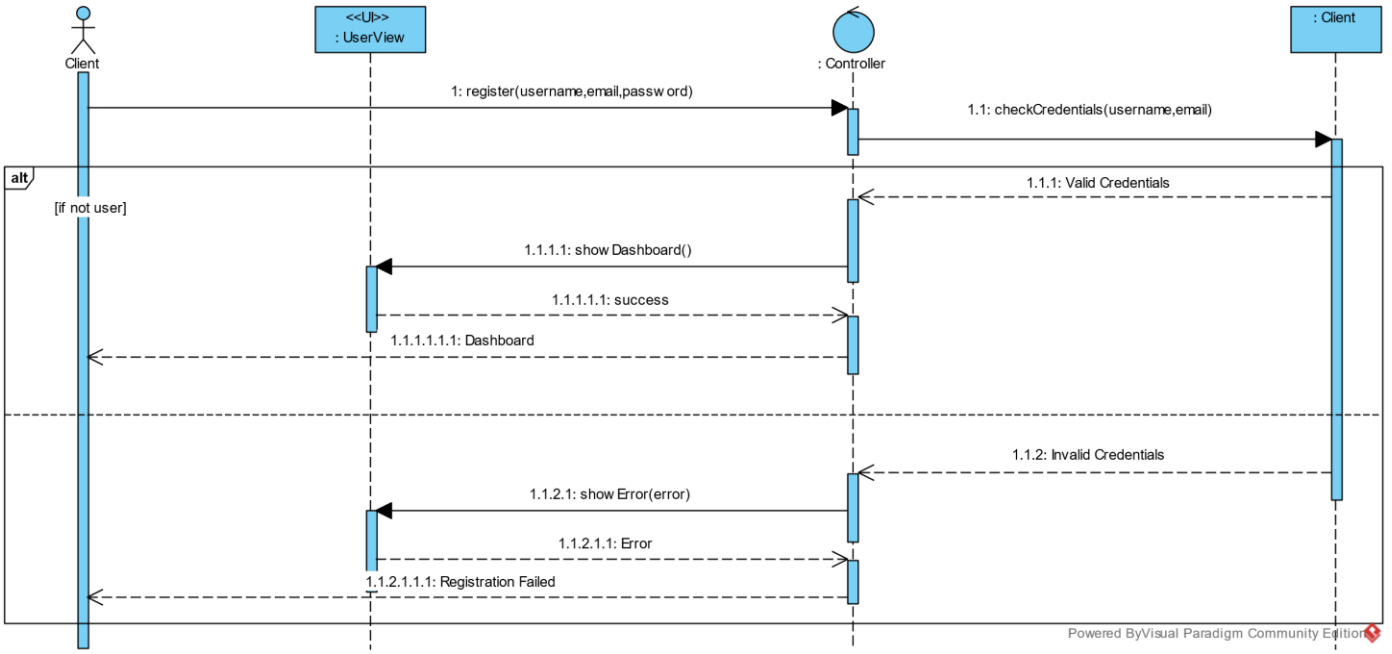

Figure 4-2 Client Signup sequence

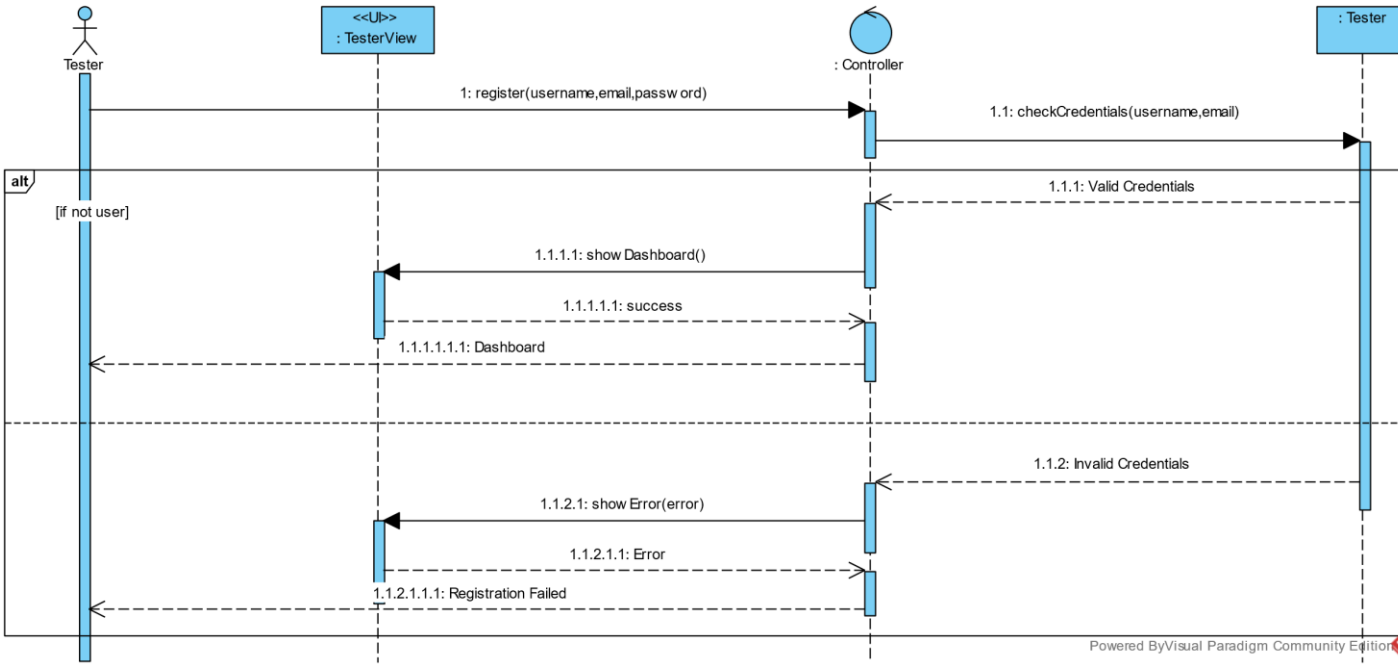

Figure 4-3 Tester Signup sequence

4.1.3 Monitor Progress

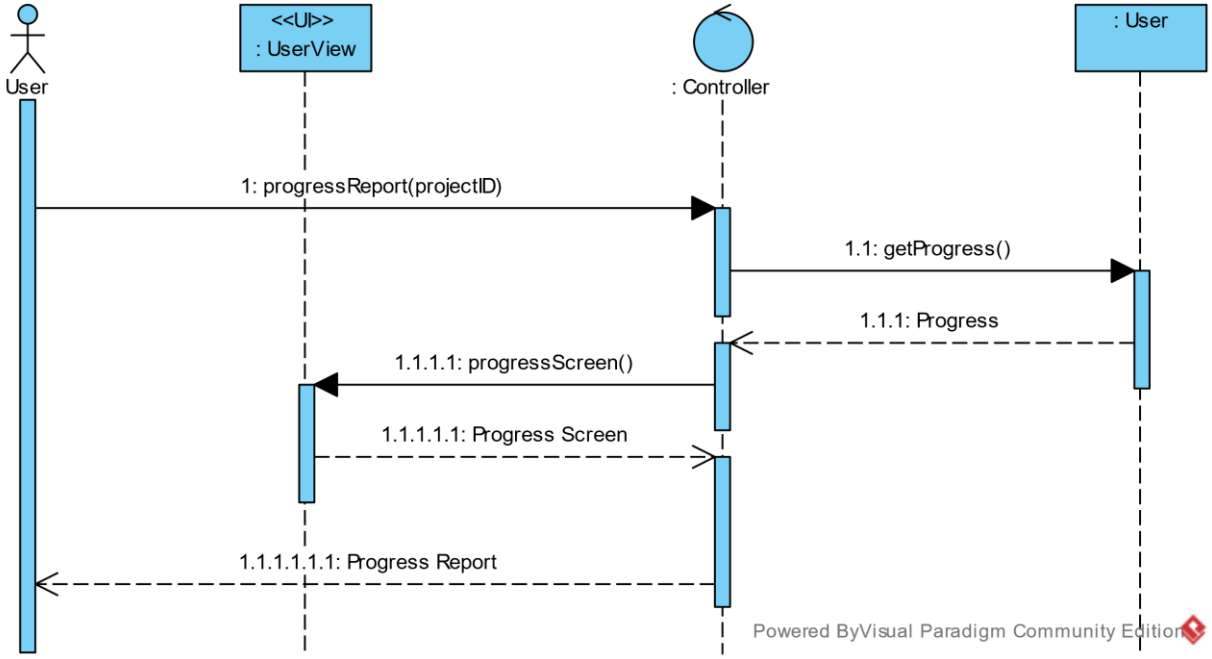

Figure 4-4 Monitor Progress sequence

4.1.4 Manage Test Cases

4.1.4.1 Add test case

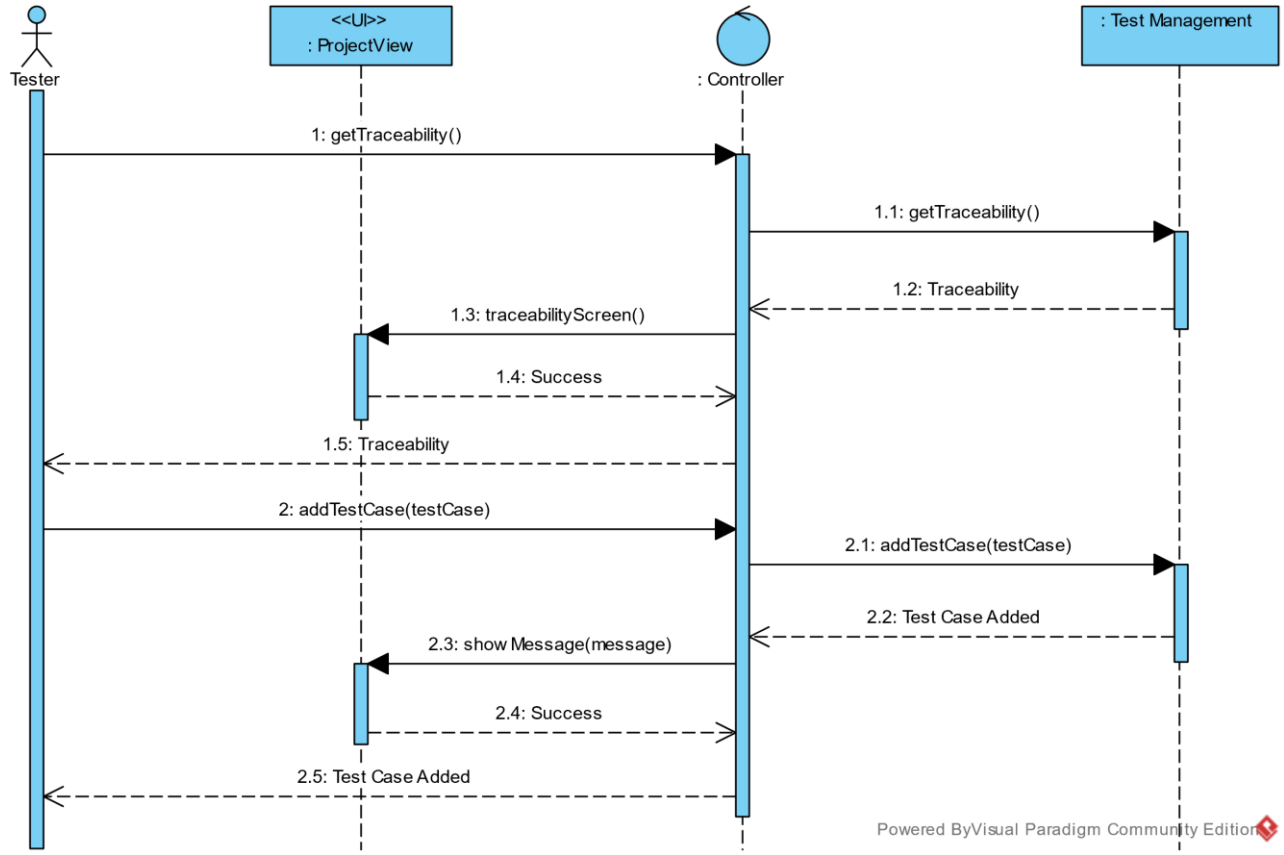

Figure 4-5 Add Test Case sequence

4.1.4.2 *Update Test Case*

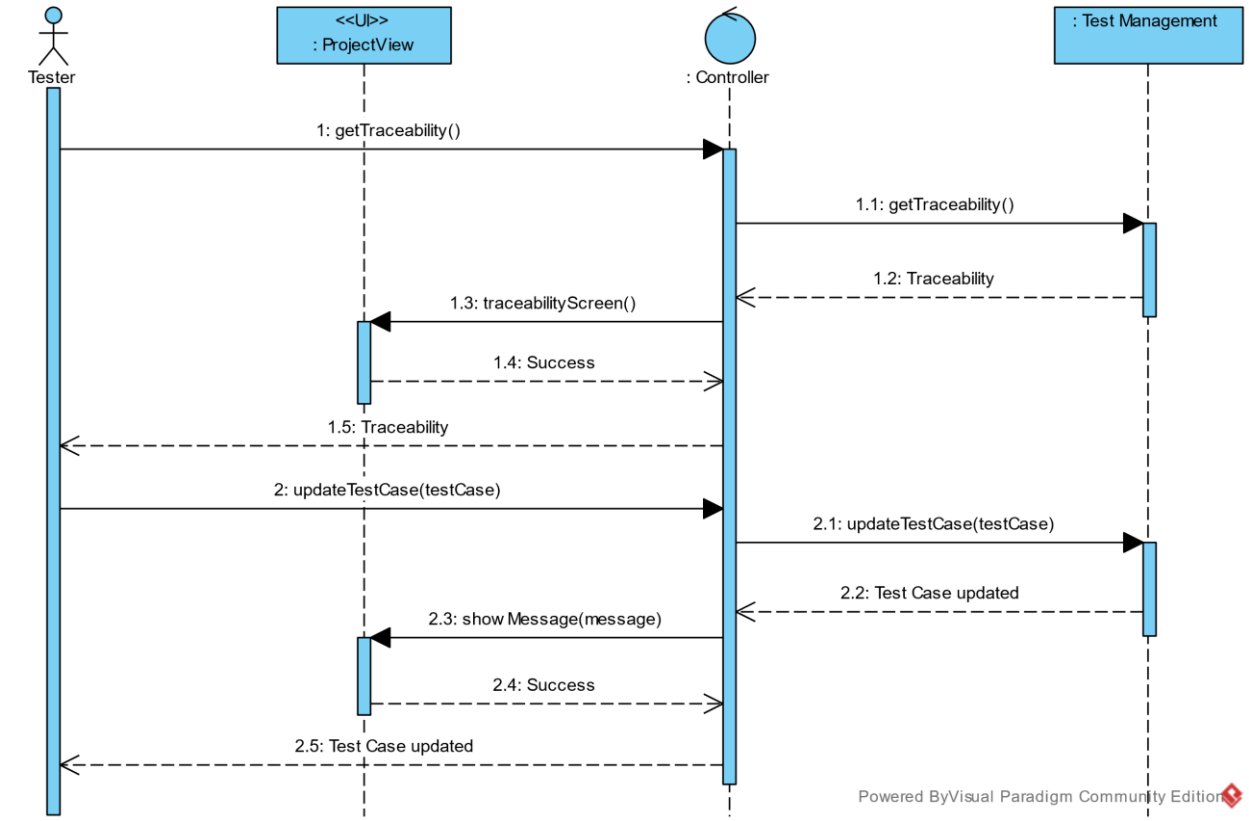

Figure 4-6 update test case sequence

4.1.4.3 *Delete Test Case*

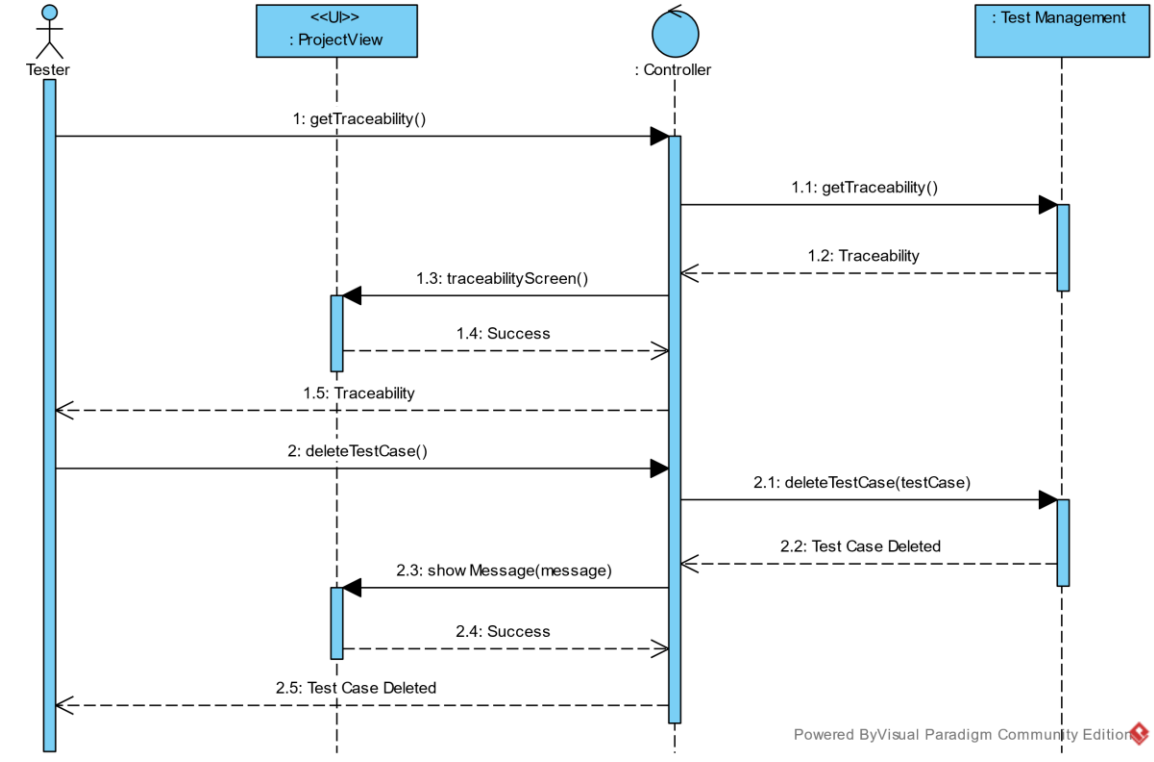

Figure 4-7 delete test case sequence

4.1.5 Submit Test Report

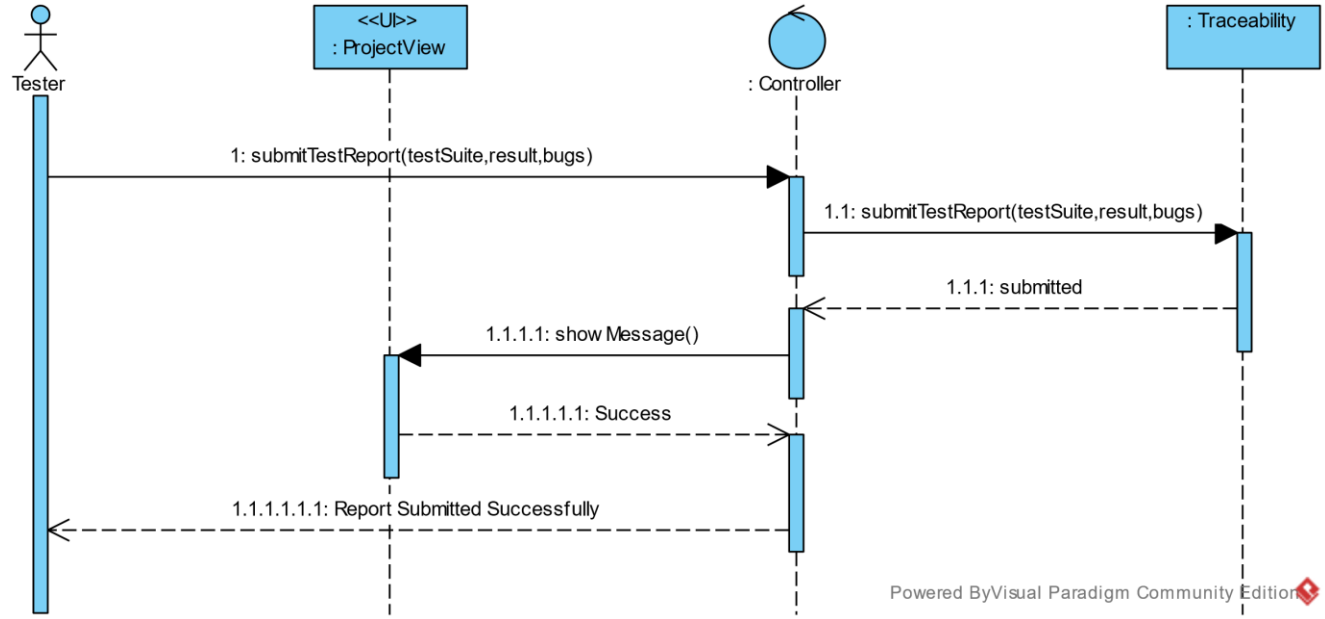

Figure 4-8 Submit Test Report sequence

4.1.6 Manage Project Request

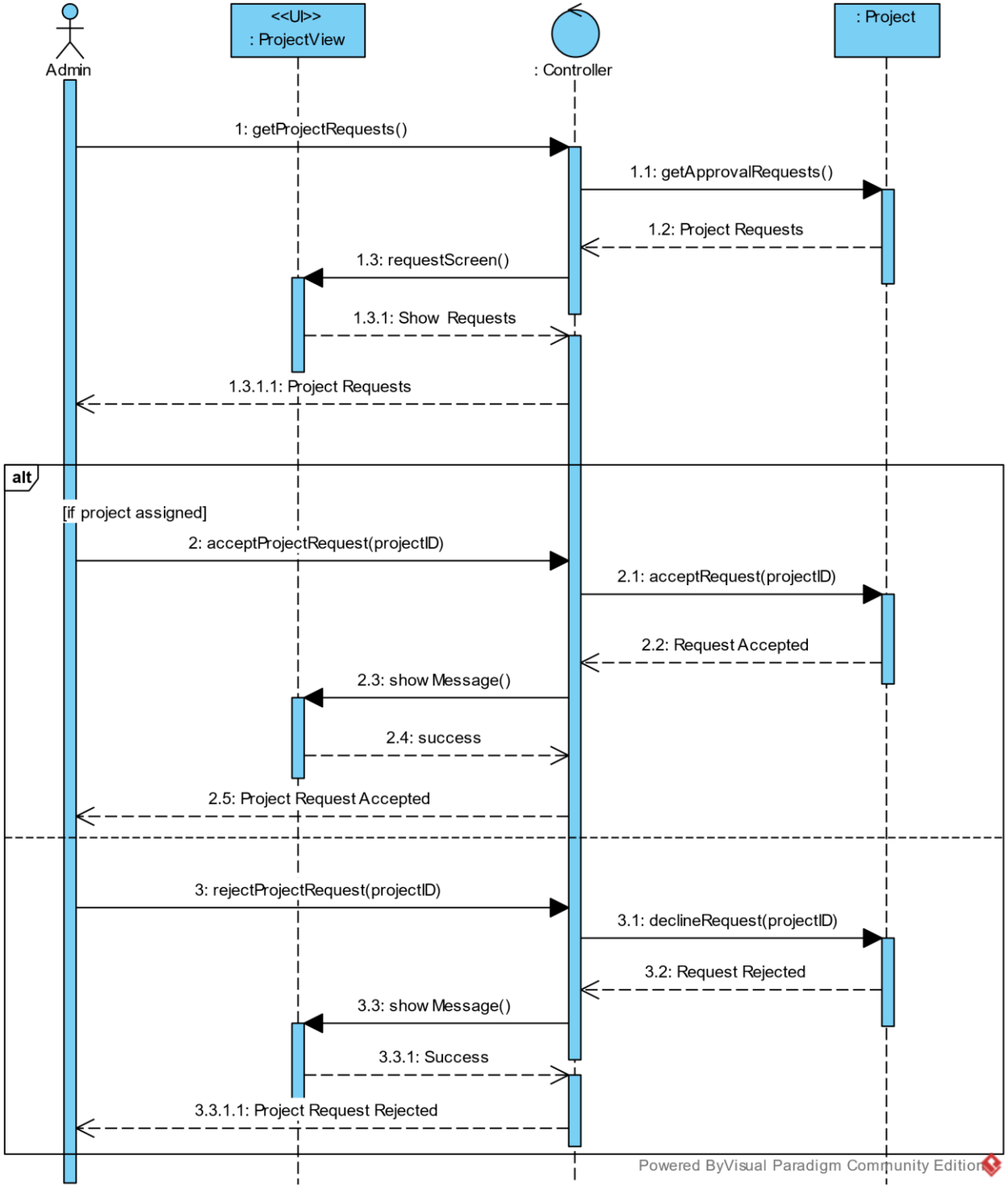

Figure 4-9 Manage Project Request sequence

4.1.7 Request Project

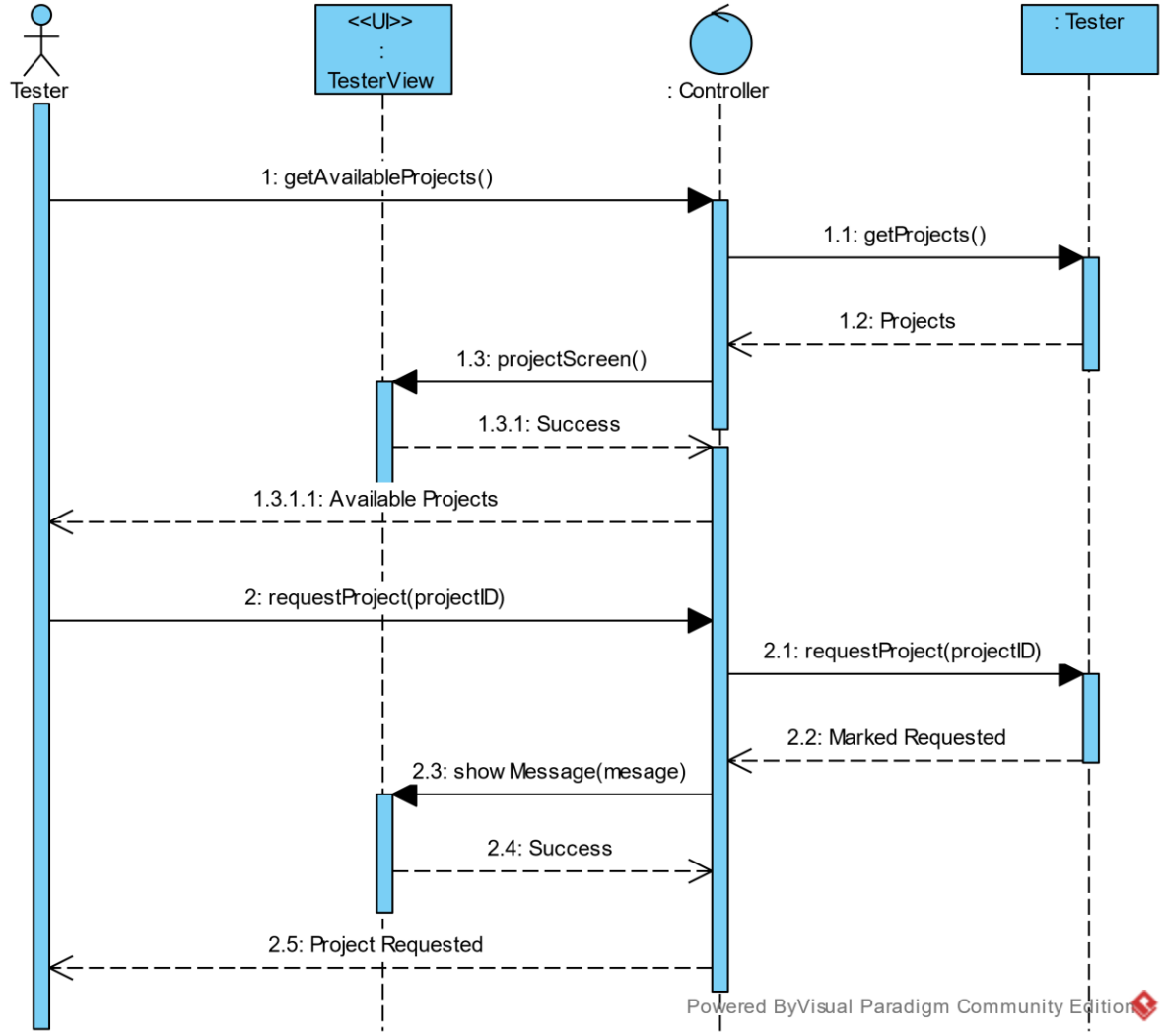

Figure 4-10 Request Project sequence

4.1.8 Request Project Initiation

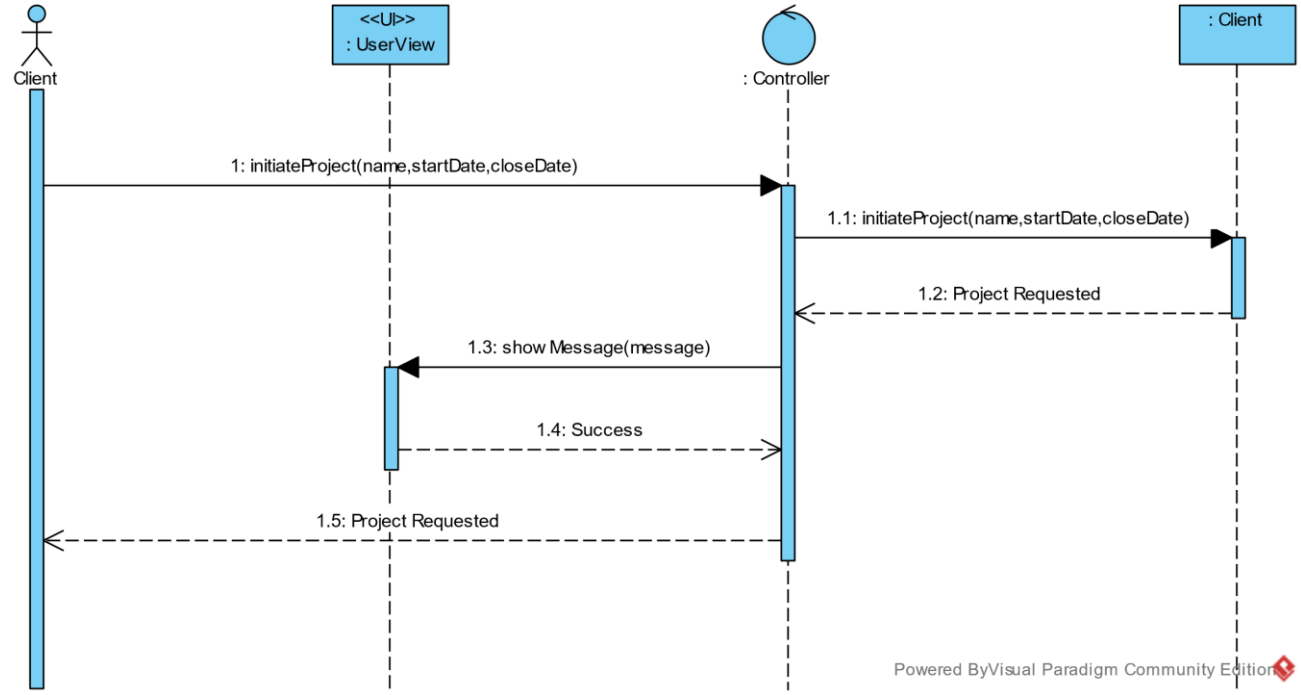

Figure 4-11 Project Initiation sequence

4.1.9 Make Payment

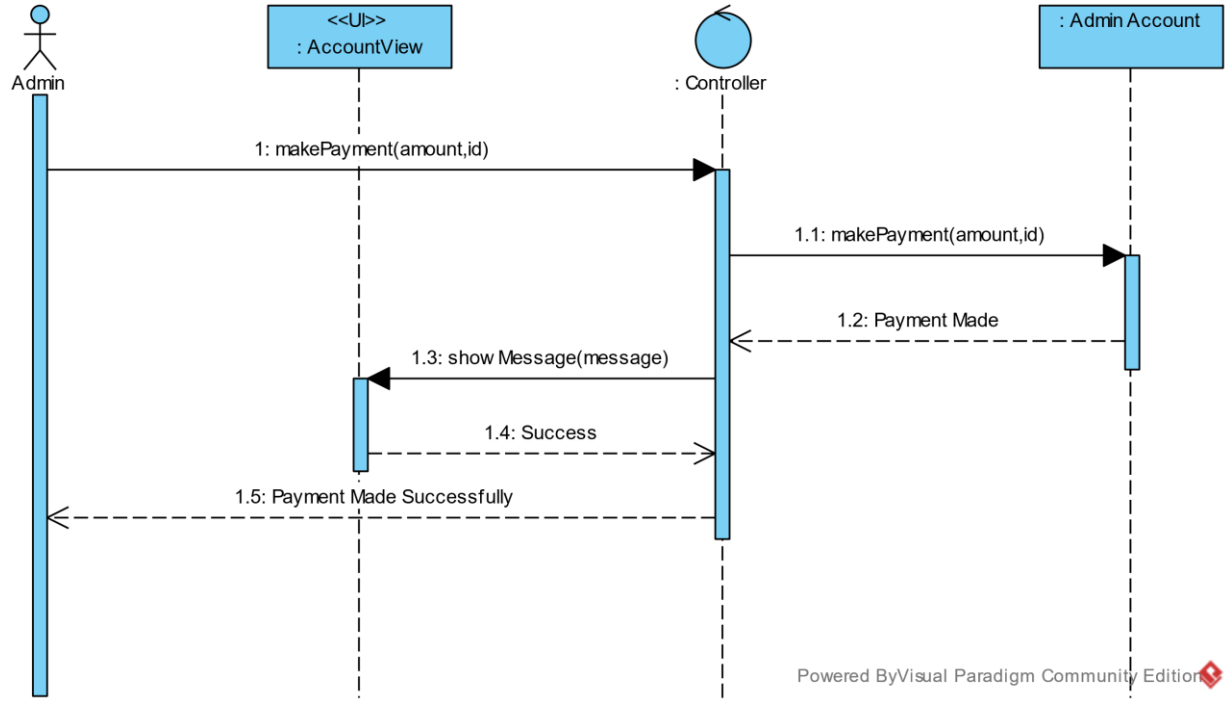

Figure 4-12 Admin Payment sequence

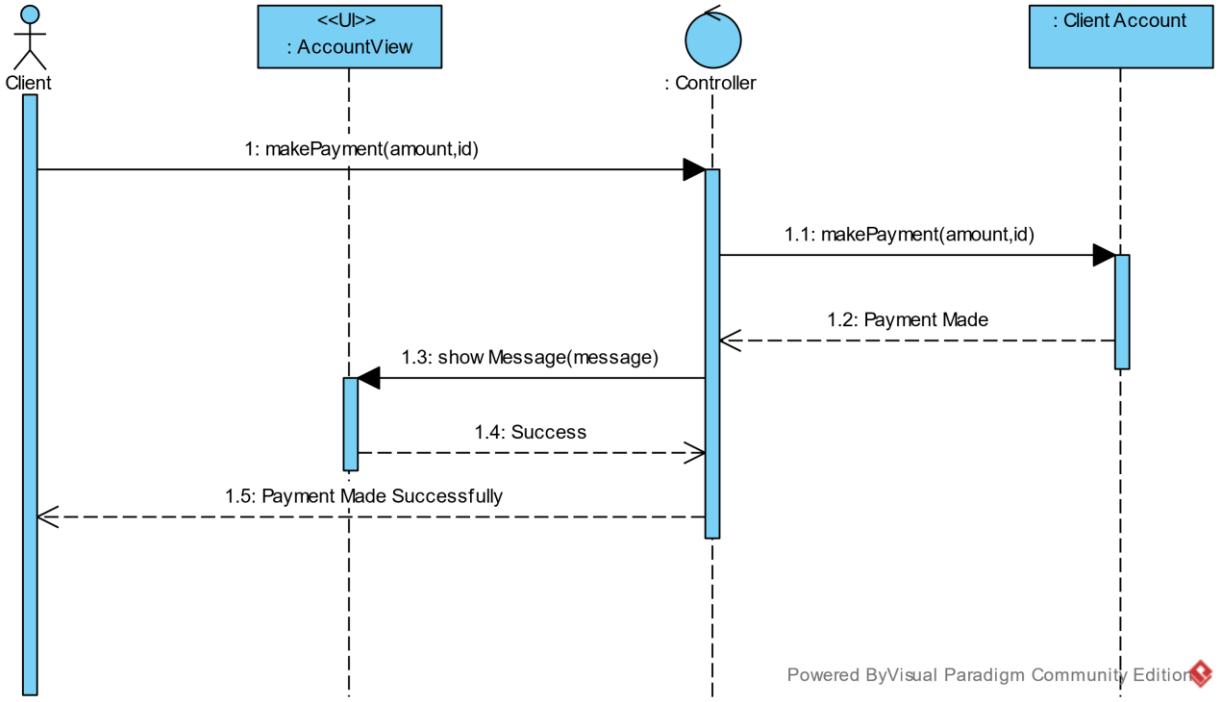

Figure 4-13 Client Payment sequence

4.2 Class Diagram

In the Class Diagram, we have identified and depicted the various model classes that represent the data and business logic of the application. These classes encapsulate the essential functions and properties required to manage and manipulate the data effectively.

Additionally, the Class Diagram includes the controller class, which acts as the central component responsible for coordinating the interactions between the View and Model. The controller class encapsulates the necessary functions to handle user input, query the Model for data, and update the View accordingly.

The Class Diagram and Sequence Diagrams in our system demonstrate a strong alignment, showcasing the implementation of the Model-View-Controller (MVC) architectural pattern.

By mapping the Class Diagram and Sequence Diagrams onto each other, we ensure a cohesive and consistent design. The alignment emphasizes the clarity and organization of the system's structure and functionality, showcasing how the MVC architectural pattern is effectively implemented. This is presented in Figure 15

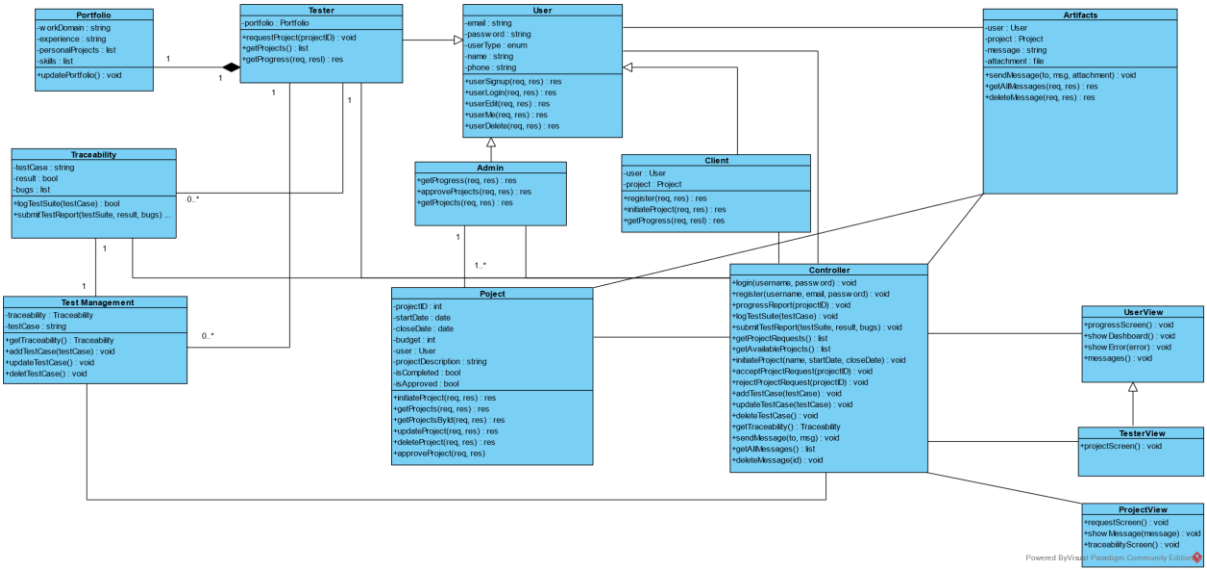

Figure 4-14 Class Diagram

4.3 Component Diagram

The Component Diagram in our system provides a comprehensive overview of the different components and technologies employed in our application.

At the core of our design, we have utilized the Model-View-Controller (MVC) architectural pattern, which divides the application into three primary components: Model, View, and Controller. This architectural approach promotes separation of concerns, modularity, and code reusability.

The Model component represents the data and business logic of the application. It includes the model classes depicted in the Class Diagram, which encapsulate data management and manipulation operations. Additionally, our chosen database technology is MongoDB, which serves as the persistent storage for the application's data.

The View component encompasses the user interface (UI) part of the application. We have employed JavaScript and React, a popular JavaScript library, to build interactive and dynamic UI components. React's component-based approach allows for efficient UI development, reusability, and a seamless user experience.

The Controller component acts as the intermediary between the Model and View, facilitating the flow of data and user interactions. It receives input from the View, processes it, interacts with the Model to retrieve or update data, and updates the View accordingly. This enables effective communication and coordination between the different components. This is presented in Figure 16

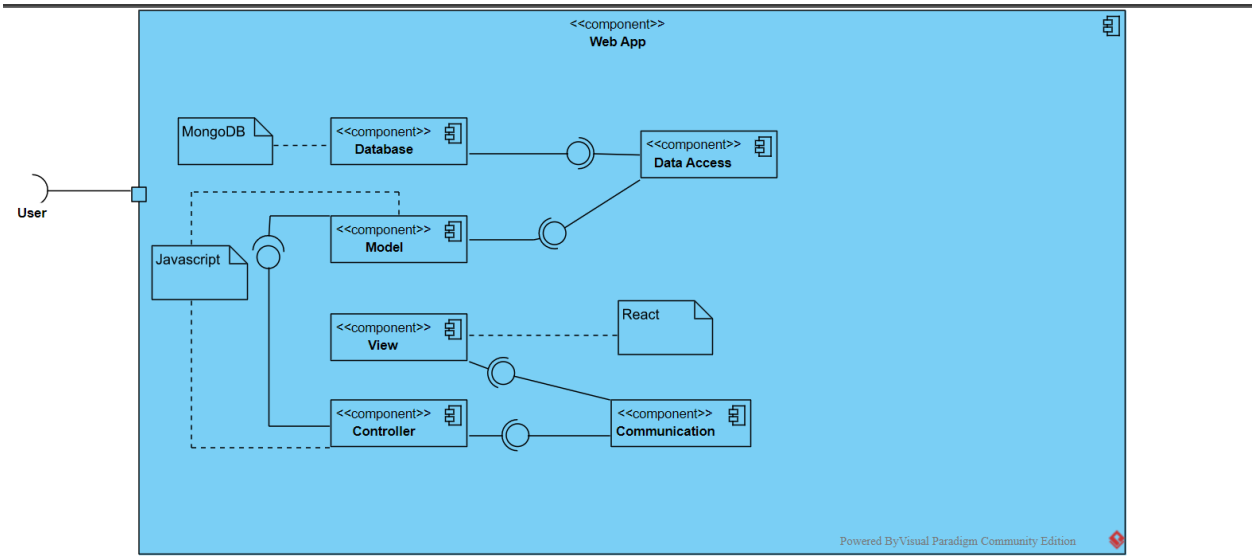

Figure 4-15 Component Diagram

4.4 Deployment Diagram

The Deployment Diagram illustrates the physical deployment of our system's components and technologies. It provides an overview of how the system is distributed across different nodes or servers.

In our deployment configuration, the frontend components, developed using React and JavaScript, are deployed on a web server. React enables the creation of dynamic and interactive user interfaces, while JavaScript enhances the functionality and interactivity of the frontend.

On the other hand, the database, implemented using MongoDB, is deployed on a separate server or cluster. MongoDB is a popular NoSQL database that offers flexibility, scalability, and efficient data storage and retrieval.

The Deployment Diagram shows the connection between the frontend components and the database, representing the flow of data between them. The frontend communicates with the MongoDB database for retrieving and storing data, ensuring seamless integration between the user interface and the underlying data storage. This is presented in Figure 17

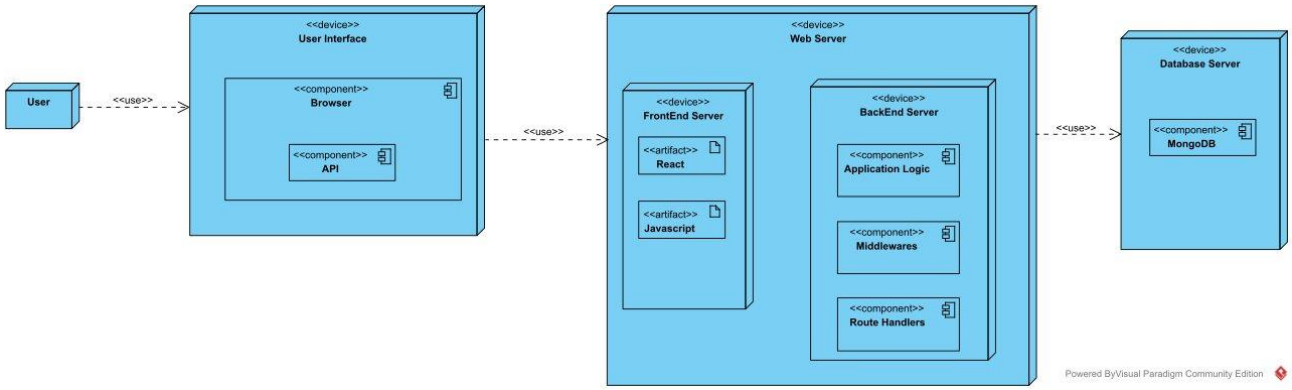

Figure 4-16 Deployment Diagram

4.5 Process Workflow Diagram

The Process Workflow Diagram illustrates the step-by-step workflow of our system involving the client, admin, and tester. This diagram demonstrates the sequence of actions and interactions among the different users throughout the process.

The workflow begins with the client initiating a test project request, providing the necessary details and requirements for the project. The request is then received by the admin, who evaluates and decides whether to accept or reject the request based on various criteria such as resource availability and project feasibility.

If the admin accepts the request, the workflow moves forward, and the tester, who is qualified and available, requests to acquire the project for testing. The admin then assigns the project to the tester, providing them with the necessary information and access.

The tester proceeds with the testing process, making test cases, and documenting the test results. Once the testing is complete, the tester submits the test report, which includes details of the test cases and identified issues.

Finally, the client is notified that the test report is available. The client can then access and download the test report. This is presented in Figure 18

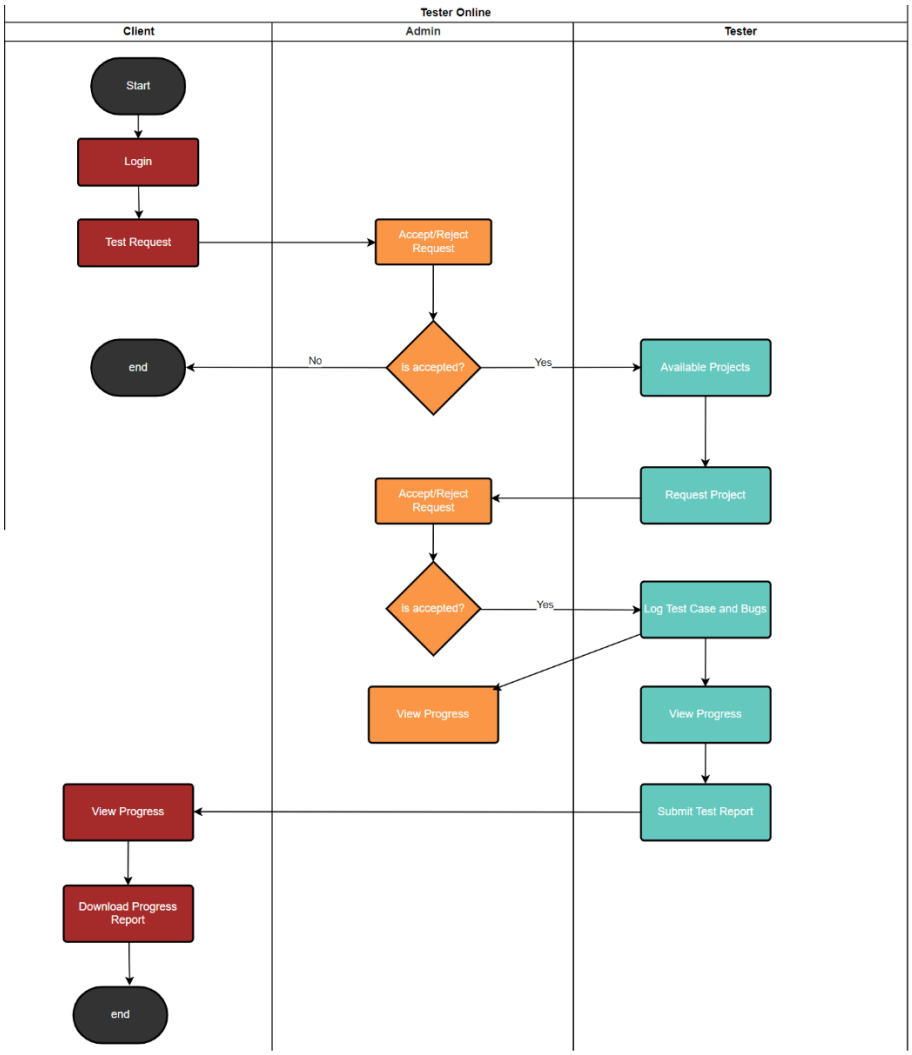

Figure 4-17 Process Workflow Diagram

4.6 Interface Design

4.6.1 Login

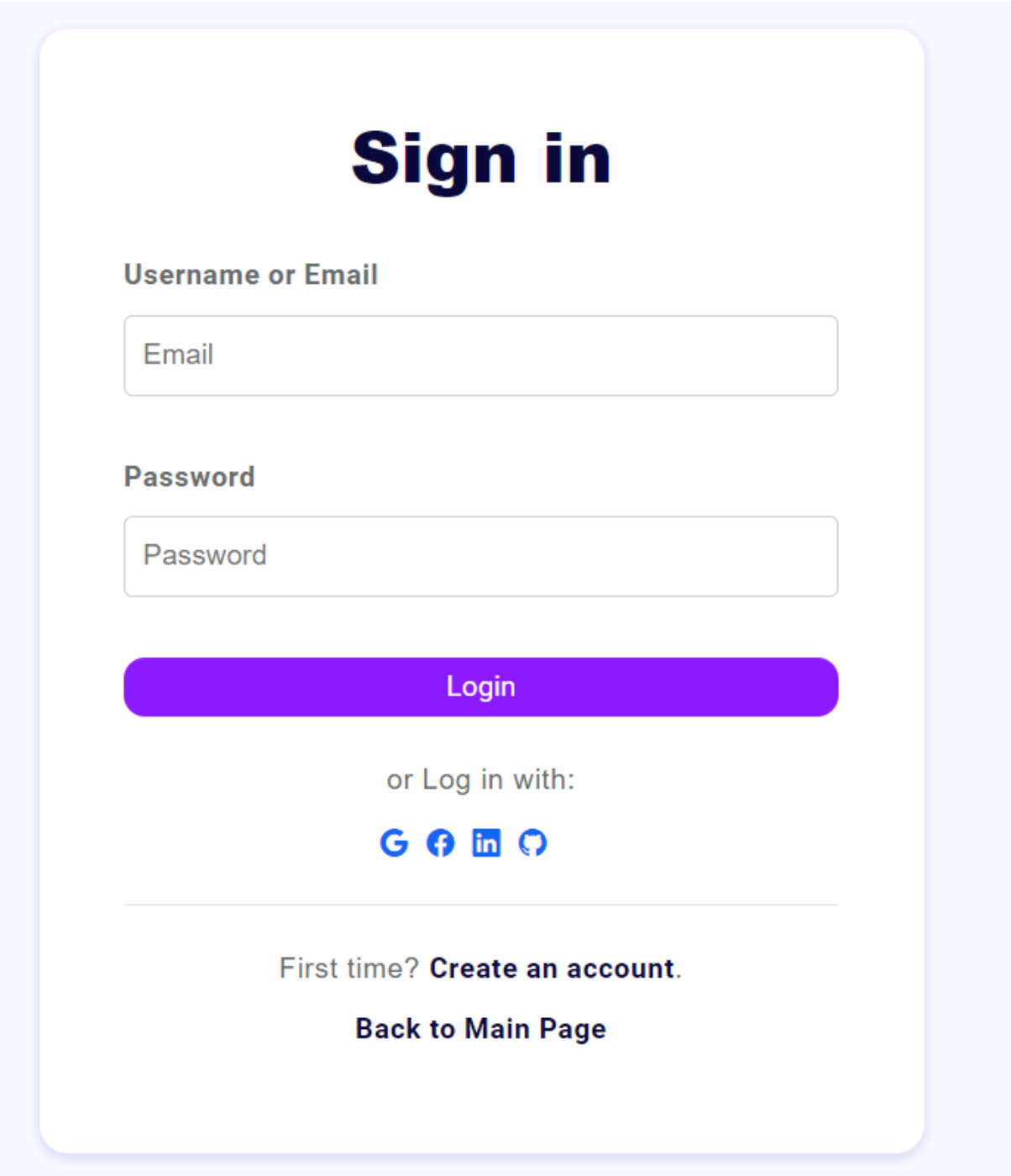

Figure 4-18 login interface

4.6.2 Register

Tester

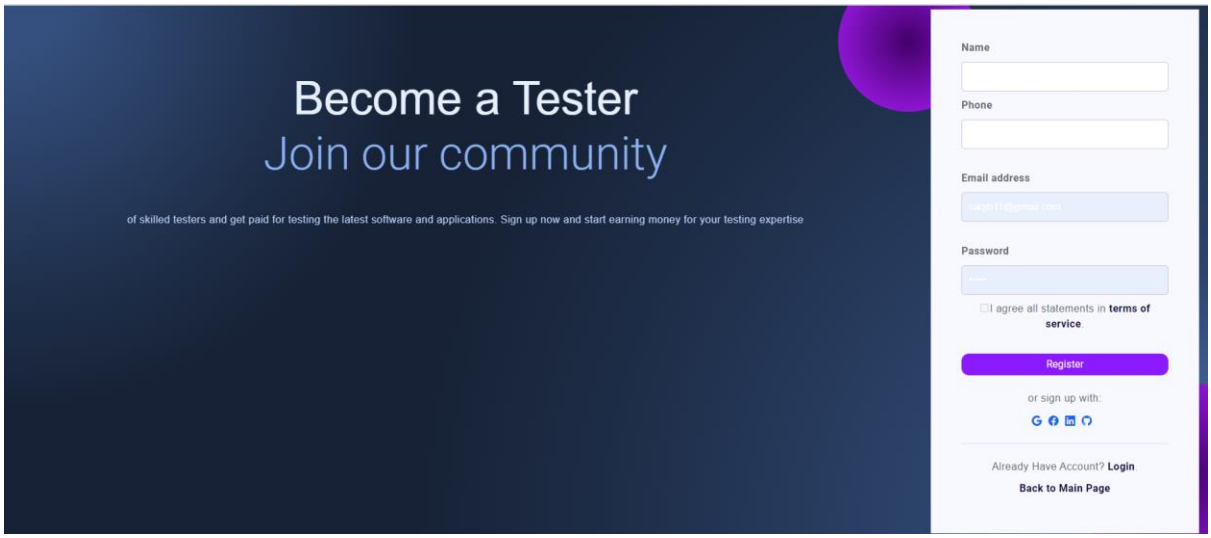

Figure 4-19 Tester Sign up interface

Client

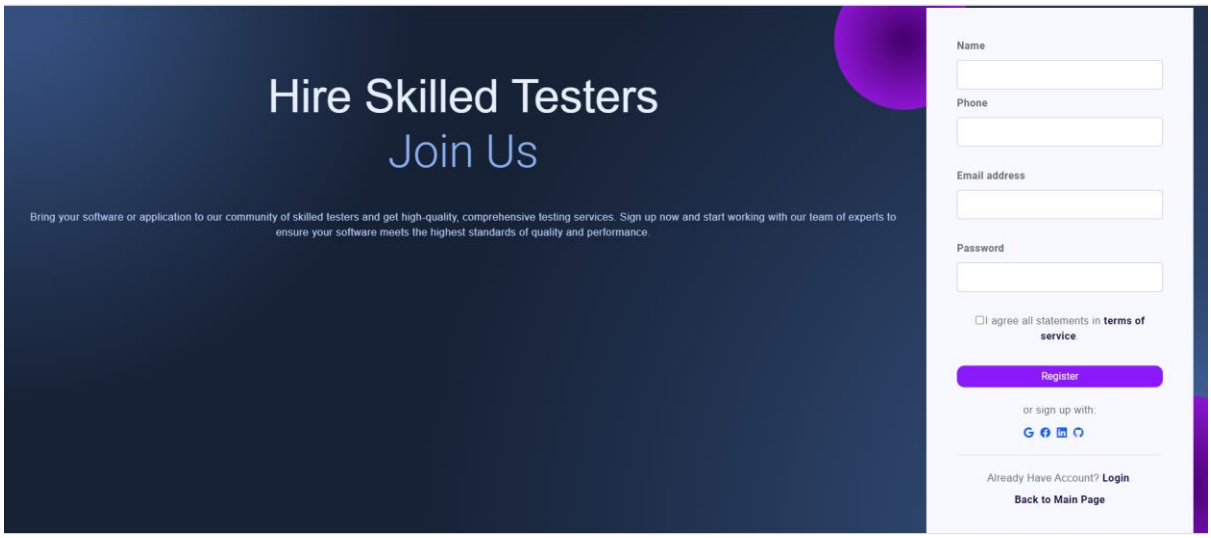

Figure 4-20 client sign up interface

4.6.3 Client Dashboard

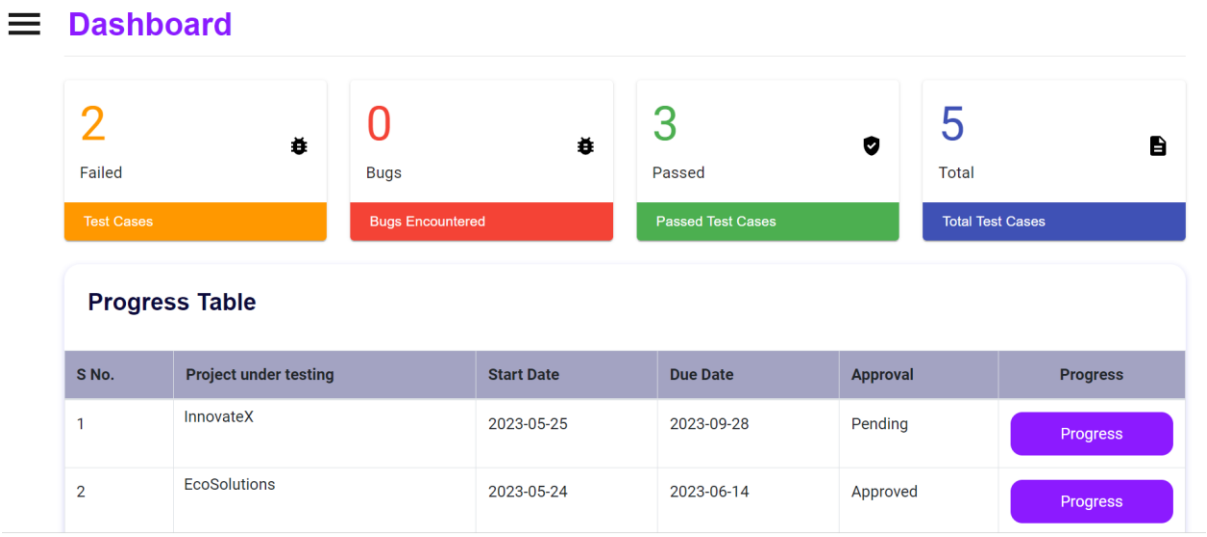

Figure 4-21 Client Dashboard interface

4.6.4 Initiate Project

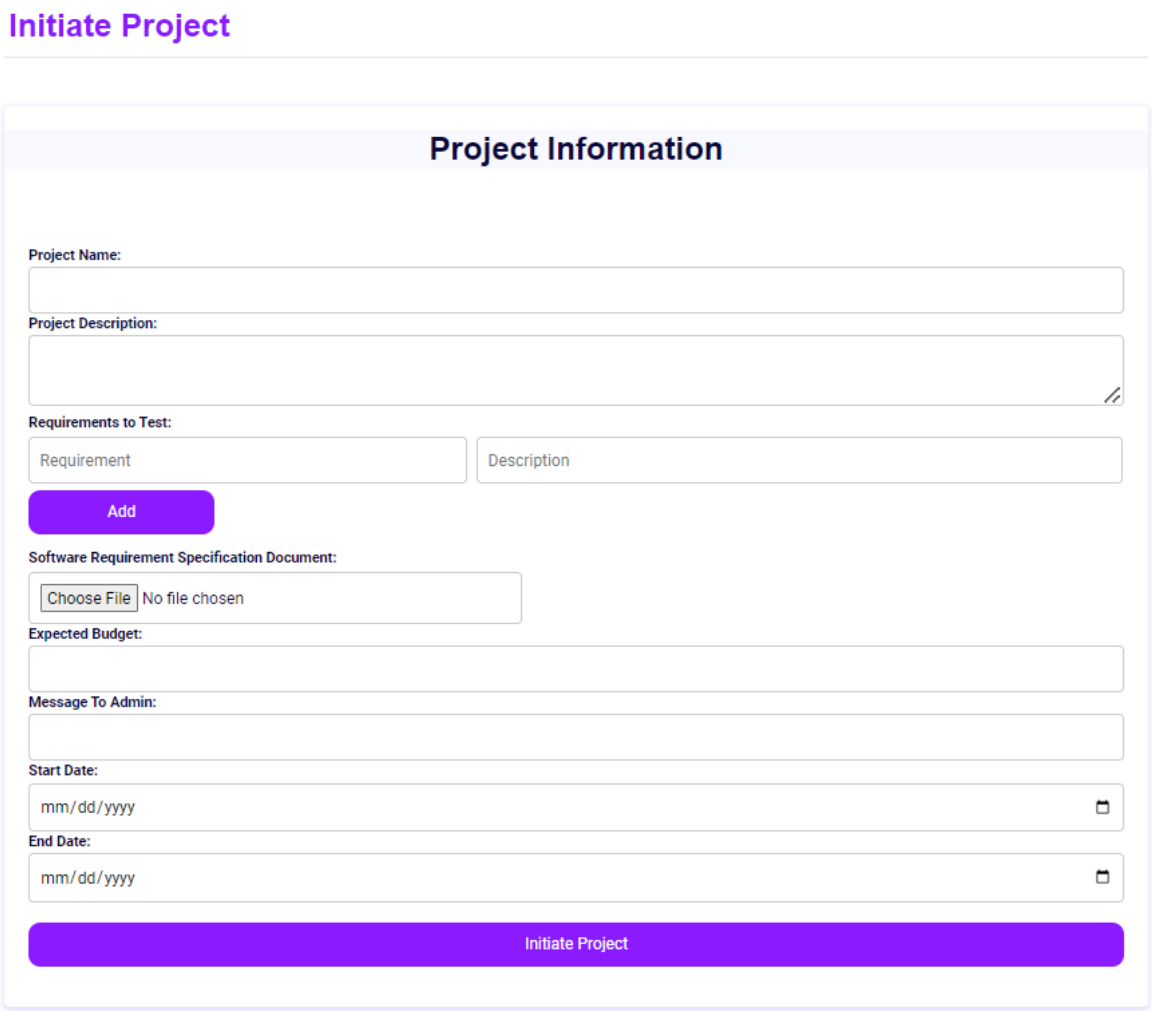

Figure 4-22 Initiate project interface

4.6.5 Progress Report

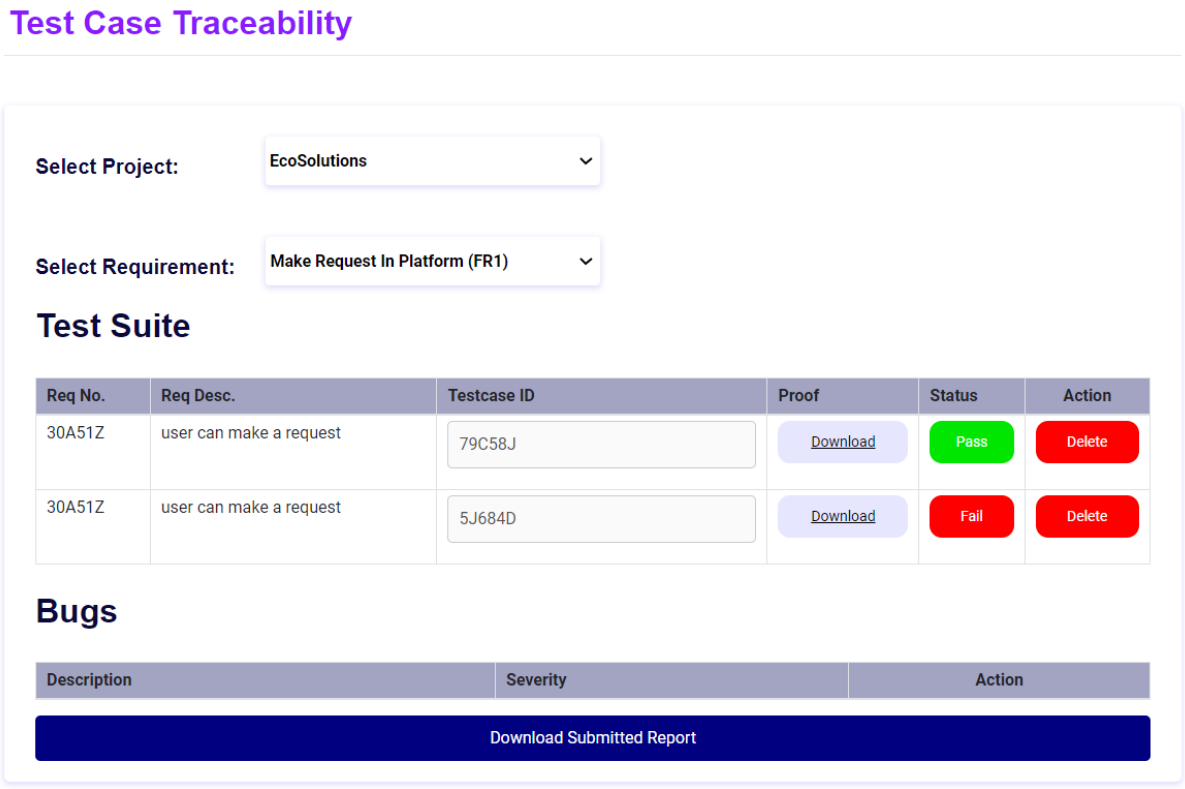

Figure 4-23 Test Case Traceability interface

4.6.6 Log Test Suite

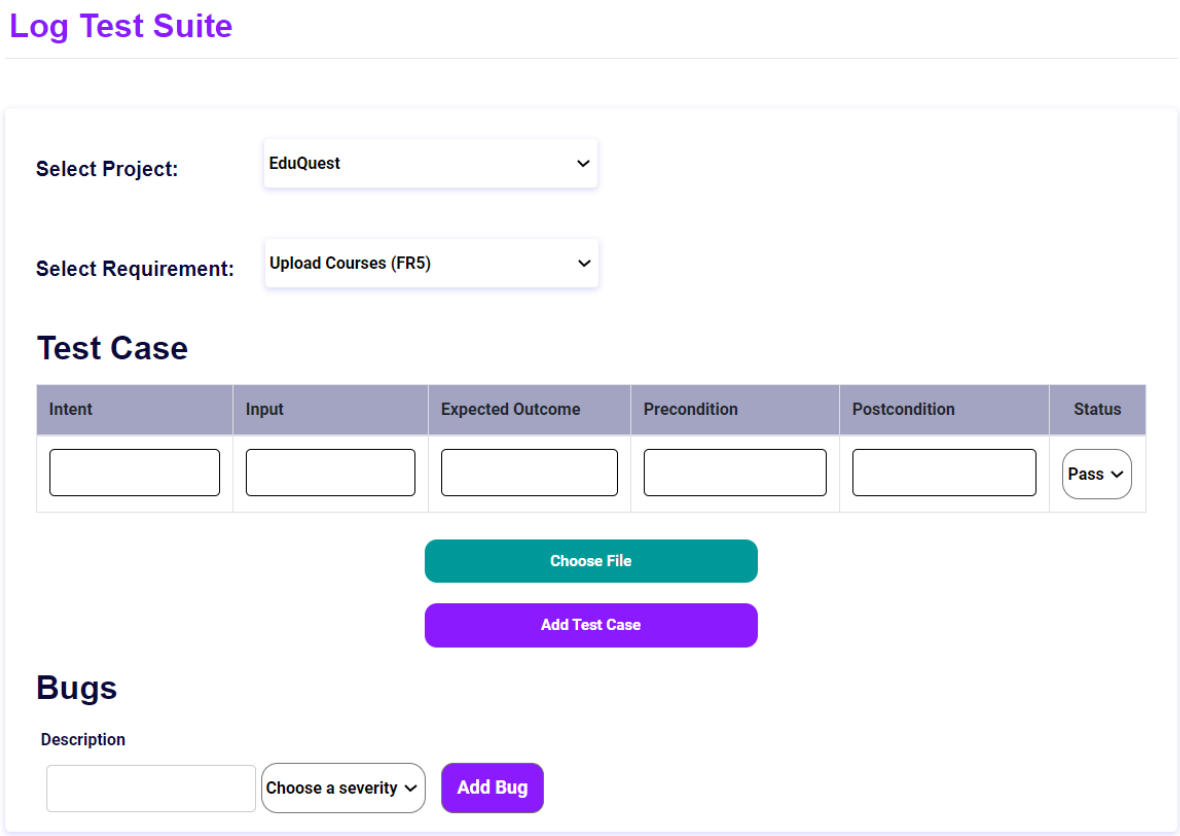

Figure 4-24 Add test Case interface

4.6.7 Traceability

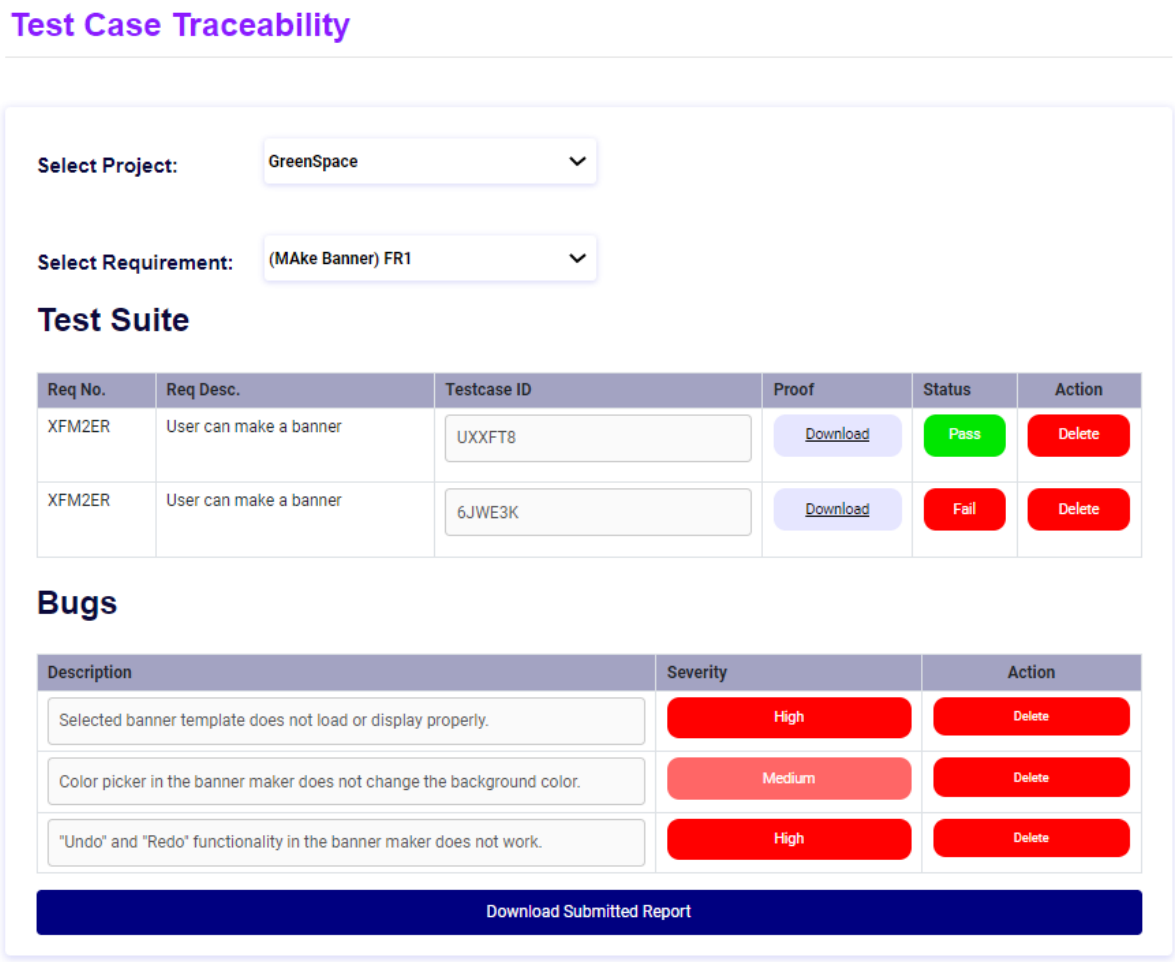

Figure 4-25 Test Case Traceability interface

CHAPTER - 5

SYSTEM IMPLEMENTATION

5 SYSTEM IMPLEMENTATION

5.1 Strategy

Our web application project followed a well-planned strategy for successful implementation. We divided the effort into two main components, namely software development and project management.

Before starting the actual development, we created a detailed plan for the complete project lifecycle, including a work breakdown structure using Microsoft Project. We also utilized GitHub for efficient collaboration among group members, and Microsoft Visual Code IDE to manage and sync our code seamlessly.

The initial phase of the project was the system architecture design, which we found to be crucial for the success of the project. We carefully planned the design, ensuring it met all project requirements. Although we faced some challenges in identifying the best design approach, we were able to overcome them through consulting with our project supervisor.

The second phase of the project was the actual implementation of the design. We started by creating a database design and proceeded to build the web application's software components. To maintain quality, we broke down the development process into smaller, manageable units and conducted various unit tests.

Once all the software components were built and tested, we integrated them into a single module and conducted multiple test cases for quality control and to check performance. We followed an iterative model throughout the development cycle to maintain quality and improve the workflow of the project.

5.2 Tool Used

During the development of our web application, we used a variety of tools and technologies to ensure the successful implementation of the project. These included:

- Visual Studio Code (VS Code) as our Integrated Development Environment (IDE) for code editing, debugging, and testing.
- Node.js as the server-side runtime environment and runtime engine.
- MongoDB as our NoSQL database for storing and managing data.
- React for building the user interface and components on the client-side.
- Amazon Web Services (AWS) for hosting and deploying the application.

- Postman for testing and validating APIs.
- GitHub as our primary code repository for centralized code sharing and version control.

5.3 CI/CD Implementation

We successfully implemented a Continuous Integration/Continuous Deployment (CI/CD) pipeline using GitHub Actions. This CI/CD pipeline played a crucial role in automating the build, testing, and deployment processes for both the frontend and backend components of our MERN stack project. To ensure efficient collaboration and version control, we chose GitHub as the hosting platform for our project's source code. GitHub Actions, the integrated CI/CD solution provided by GitHub, served as the foundation for automating our development workflow.

In our CI/CD pipeline, we defined separate workflows for the frontend and backend components of our project. This approach allowed us to manage and test each component independently, ensuring a more granular and efficient development process.
Within the frontend workflow, we utilized npm, the package manager for JavaScript, to run the unit tests for our frontend codebase. By including the appropriate commands, such as npm run test, in the workflow configuration file, GitHub Actions automatically executed these tests whenever changes were made to the frontend repository. This enabled us to validate the functionality and integrity of our frontend codebase.

Similarly, in the backend workflow, we again employed npm to run the unit tests for our backend code. By including the necessary test scripts, such as npm run test, in the workflow configuration file for the backend repository, GitHub Actions executed the tests upon detecting changes. This ensured that our backend code was thoroughly tested for any potential issues or bugs.

As part of the CI/CD pipeline, we also automated the deployment process for our MERN stack application. Once the unit tests passed successfully, GitHub Actions initiates the deployment of our application. The deployment of our application is on AWS (amazon web services).

CI/CD workflow:

1. Setup Job: In this step, the CI/CD workflow sets up the job configuration, specifying the necessary environment and resources required for the subsequent steps.
2. Checkout Code: The workflow checks out the source code from the repository, ensuring that the latest version of the code is obtained for further processing.
3. Setup Node.js 16.x: To ensure compatibility and consistency, the workflow sets up the Node.js environment with version 16.x, which is required for running the backend and frontend components.
4. Install Frontend/Backend Dependencies: The workflow installs the necessary dependencies for both the frontend and backend components. This step ensures that all required libraries, modules, and packages are available for building and testing the code.
5. Run Backend/Frontend Tests: The workflow executes the unit tests for both the backend and frontend components. This step verifies the functionality, correctness, and reliability of the code, ensuring that any issues or bugs are identified and addressed.
6. Setup CLI Environment for AWS: In order to deploy the application on AWS (Amazon Web Services), the workflow sets up the Command Line Interface (CLI) environment, ensuring that the necessary AWS credentials and configurations are in place for the deployment process.

7. Deployment on AWS: After the tests have passed successfully, the workflow triggers the deployment process on AWS. This step involves uploading the built artifacts and deploying the application to the specified AWS environment, ensuring its availability for end-users.

8. Post Setup Node.js 16.x: This step performs any necessary post-configuration tasks related to the Node.js environment after the tests and deployment have been completed. It ensures that the environment is properly cleaned up and prepared for subsequent CI/CD workflows or stages.

9. Post Checkout Code: After the deployment and post-configuration steps are complete, this final step performs any required post-processing tasks related to checking out the code. It ensures that the code repository and workspace are in a consistent state for future development and testing.

Below are the screen shots of CI/CD pipeline.

Frontend:

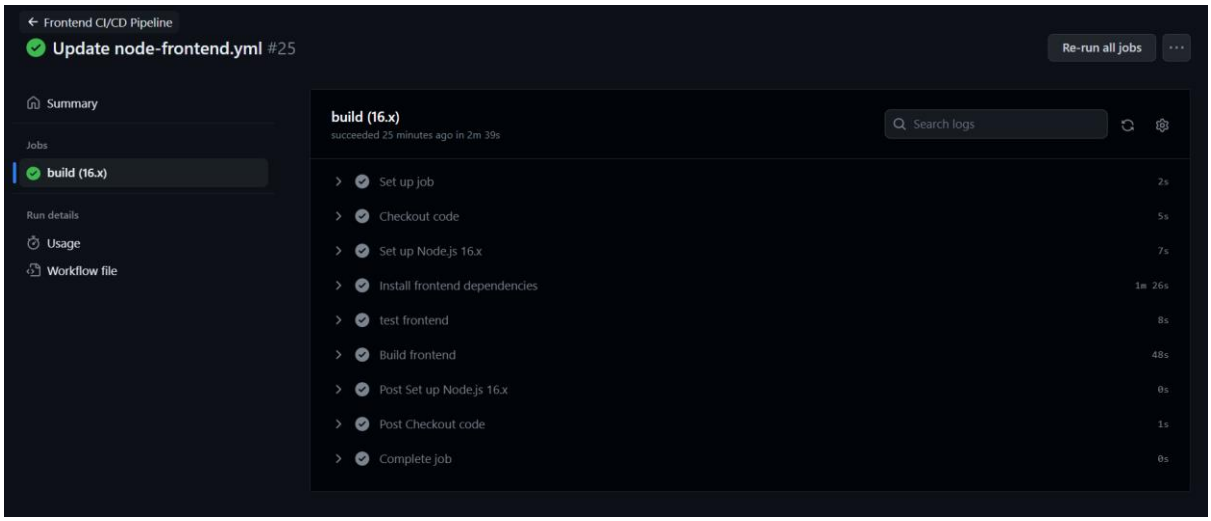

Figure 5-1 CI/CD pipeline for Frontend

Backend:

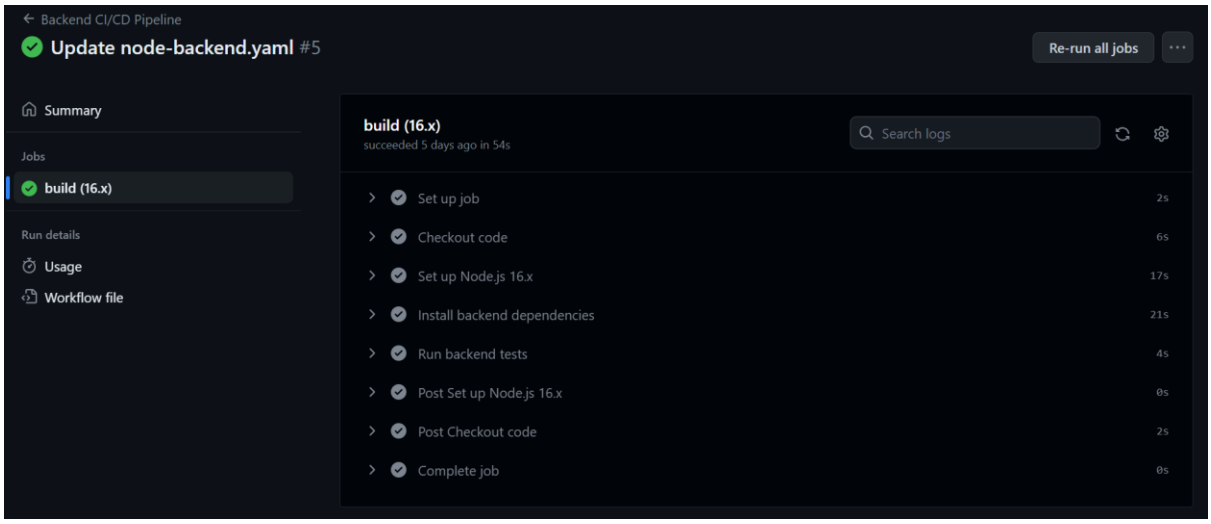

Figure 5-2 CI/CD pipeline for Backend

5.4 Methodologies

The system is developed using iterative model of Software development cycle. The main objective during the development was to build the project in chunks and test that chunks again and again to get high performance. The focus was on real time syncing and key performance elements which take lot of time. Several versions of the project are built, and each latest version add new key features to previous one.

Then module testing is also performed as the module is completed the test cases were performed and the bug report is generated. Then integration of these chunks results in the whole product.

5.5 System Architecture

The system architecture of our web application follows a standard Model-View-Controller (MVC) design pattern, with a client-server architecture. This approach separates the application into three distinct layers: the Model, which handles data storage and retrieval; the View, which presents data to the user in a user-friendly format; and the Controller, which acts as an intermediary between the Model and the View, processing user inputs and updating the Model accordingly.

In our system, the Model layer is built using MongoDB, a NoSQL database that allows for efficient and scalable storage and retrieval of large amounts of data. The View layer is developed using React, a popular front-end JavaScript library that enables the creation of dynamic, interactive user interfaces. The Controller layer is implemented using Node.js, a JavaScript runtime that allows for fast and efficient server-side processing of user inputs and application logic.

5.5.1 Data Layer

The data layer of the system is responsible for managing the persistent storage and retrieval of data. The system uses MongoDB as its primary database technology, which provides a NoSQL document-based storage system. This technology was selected because it provides a flexible and scalable solution for managing the large volumes of data generated by the system.

The data layer is built on top of the Node.js runtime environment and uses the Mongoose Object Data Modeling (ODM) library to interact with the database. The Mongoose ODM provides a simple and intuitive way to define the data schema and query the database, which reduces the amount of code required to perform CRUD (Create, Read, Update, Delete) operations on the data.

5.5.2 Processing Layer

The processing layer of the system is responsible for the logic and business rules of the application. This layer communicates with the data layer to retrieve and manipulate data, and with the presentation layer to send the processed data to the user interface.

The processing layer of our system is built using Node.js, a JavaScript runtime built on Chrome's V8 JavaScript engine. It provides an event-driven architecture and non-blocking I/O capabilities, making it a good fit for building scalable and efficient server-side applications.

To manage the business logic of the application, we used the Model-View-Controller (MVC) architecture. This architecture separates the application logic into three interconnected components: the model, which represents the data and business logic; the view, which displays the data to the user; and the controller, which handles user input and updates the model and view accordingly.

The processing layer includes the implementation of these three components. The controller component handles the incoming requests and delegates them to the appropriate model and view components. The model component handles the data and business logic, and the view component renders the response to the user.

5.5.3 Representation Layer

The representation layer of the system is responsible for presenting data and functionalities to the end-users through the user interface. It is implemented using the ReactJS library, which is a popular JavaScript library for building user interfaces.

The representation layer is designed to be responsive, user-friendly, and easily navigable. The system provides a modern and intuitive user interface, which is consistent across all the pages. The layout of the pages is designed to be simple and straightforward, allowing users to quickly find what they are looking for.

CHAPTER - 6

SYSTEM TESTING

6 SYSTEM TESTING

6.1 Test Strategy

The testing is the important part of system or a software completion as it controls quality and assured quality engineering. Different strategies can be applied to the test the quality of software. Testing must be done before system deployment and it make the system bug free and become easy to use. It also becomes trust worthy. We applied unit testing on each part while building and then we applied component or module testing. And after integration of the system, we applied a complete testing mechanism by declaring some test cases and then applied it on our system and create a report for that.

6.2 Unit Testing

Unit testing is an essential aspect of software development, involving the writing of tests for small code units. In our project, we utilized the React.js environment and the Jest test library to write and run these unit tests. By testing individual units within the React.js framework, we could ensure the accuracy of our code and identify any logical errors or issues related to business logic. This approach allowed us to validate the behavior of our software components in an isolated and controlled manner, enabling early detection and resolution of potential problems. We identified 22-unit test cases and ran these on react.js environment using jest library.

6.3 Component Testing

When all the units are created then the units are testing with each other this create a component or module. Testing of such system leads to component testing. It is easier task and short in length.

6.4 Integration Testing

In integration of a system, several components are joined together to complete the project. The testing at the time of combining these components is called as integration test. We used Postman and Node JS libraries mocha and chai for the integration testing purposes. We identified 9 integration test cases. Some of the screen shots are shown in Figure 27 - 30

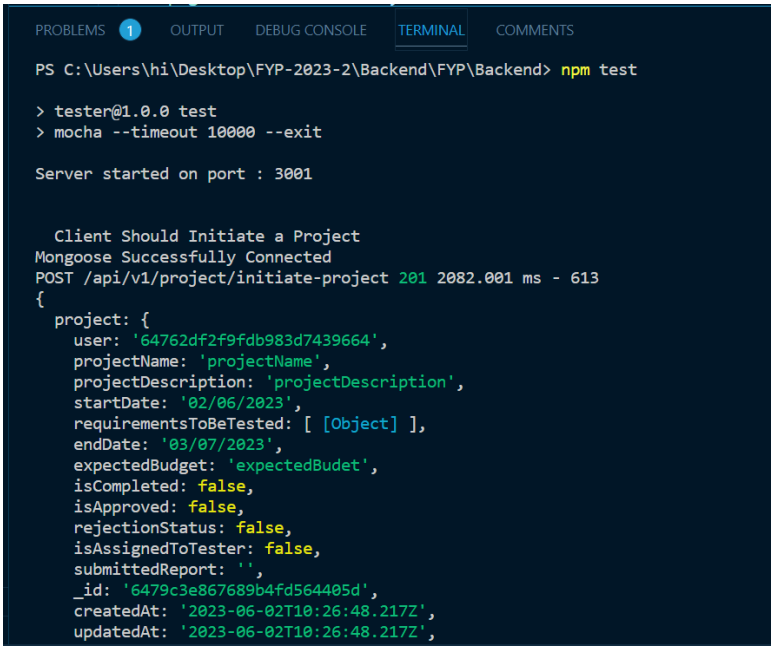

Figure 6-1: Integreation level test case using Node.js

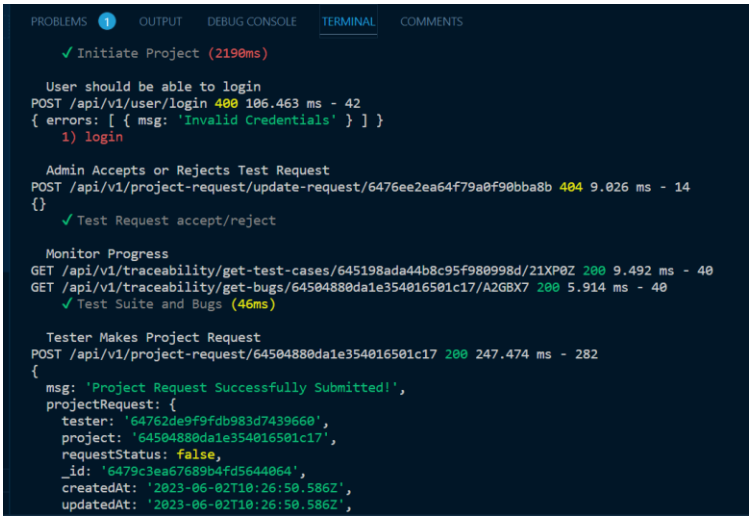

Figure 6-2: Output of Integration level test case using Node.js

Figure 6-3: Code example of Integration level test case using Node.js

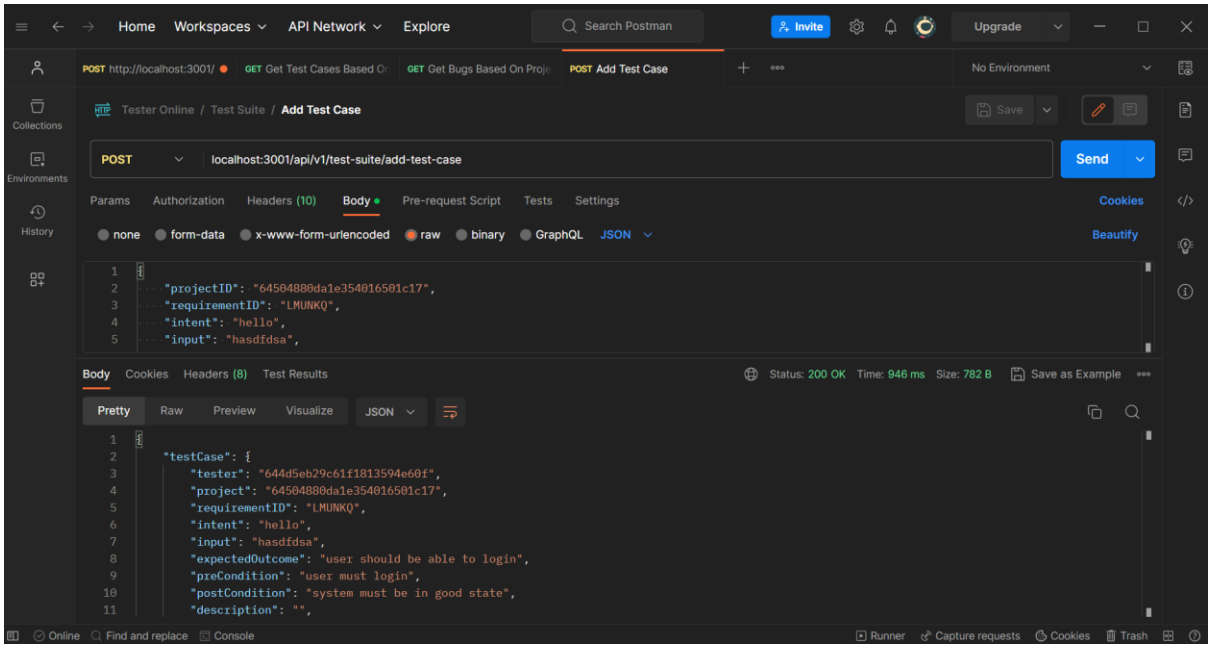

Figure 6-4: Postman test case example

6.5 System Testing

System testing is a crucial phase in the software testing process that focuses on evaluating the entire system as a whole. It involves testing the system's functionalities, interactions, and performance to ensure its compliance with the specified requirements and to identify any potential defects or issues.

In our project, we conducted thorough system testing using Selenium, a widely-used automation framework for web application testing. Selenium allowed us to simulate user interactions and validate the system's behavior across different browsers and platforms.

During system testing, we followed a structured approach to validate the system's functionalities and ensure its reliability, stability, and compatibility. This involved executing a series of test scenarios that covered a wide range of scenarios and user interactions. We identified 50 test scenarios for acceptance testing and 101 test cases using Selenium. Some of the screen shots are shown in Figure 31 - 32

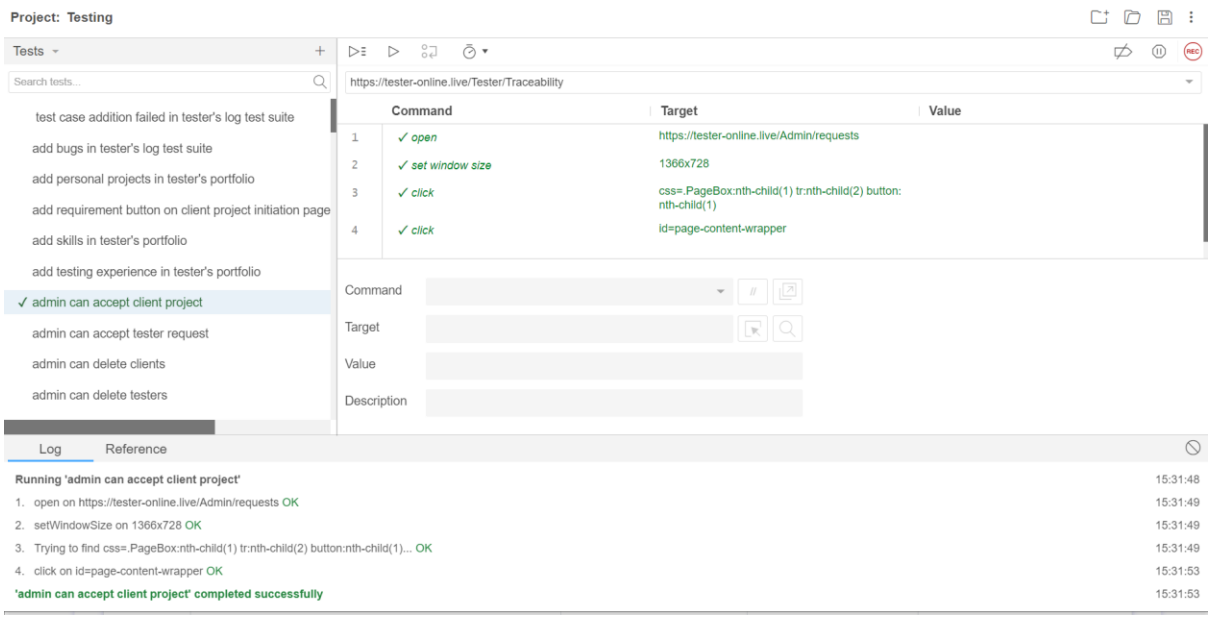

Figure 6-5: Acceptance level test case using Selenium for Admin Panel

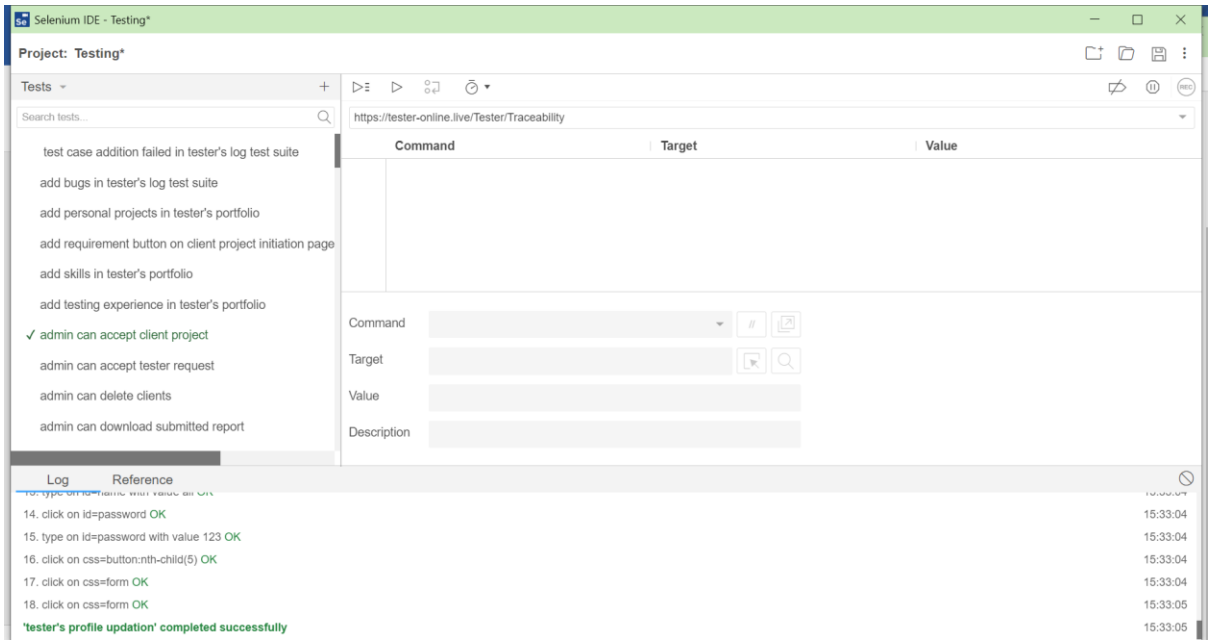

Figure 6-6 Acceptance level test case using Selenium for Client Project Intiiation

Test Case Scenarios:

6.5.1 Tester Login

T Sen ID	TS-001	**T C ID**	TC-001
T C Desc	Tester login positive test case	**T Prior**	High
Pre	Valid User Account	**Post**	Tester Logged in

Test Execution Steps:

Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Enter valid email and password and press login button.	Email: ayesha@gmail.com Password: 12345	Login Successful	Login Successful	Pass

T Sen ID	TS-001	**T C ID**	TC-002
T C Desc	Tester login negative test case	**T Prior**	High
Pre	Invalid User Account	**Post**	Login Unsuccessful

Test Execution Steps:

Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Enter invalid email and password and press login button.	Email: sadfr Password: 1234	Login unsuccessful	Login unsuccessful	Pass

6.5.2 Client Login

T Sen ID	TS-002	**T C ID**	TC-003
T C Desc	Client login positive test case	**T Prior**	High
Pre	Valid User Account	**Post**	Client Logged in

Test Execution Steps:

Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Enter valid email and password and press login button.	Email: anza@gmail.com Password: 9999	Login Successful	Login Successful	Pass

T Sen ID	TS-002	**T C ID**	TC-004
T C Desc	Client login negative test case	**T Prior**	High
Pre	Invalid User Account	**Post**	Login Unsuccessful

Test Execution Steps:

Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Enter invalid email and password and press login button.	Email: aefh@gmail.com Password: 68uf	Login unsuccessful	Login unsuccessful	Pass

6.5.3 Admin Login

T Sen ID		TS-003	**T C ID**		TC-005
T C Desc		Admin login positive test case	**T Prior**		High
Pre		Valid User Account	**Post**		Admin Logged in
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Enter valid email and password and press login button.	Email: admin@tester-online.live Password: 12345	Login Successful	Login Successful	Pass

T Sen ID		TS-002	**T C ID**		TC-006
T C Desc		Admin login negative test case	**T Prior**		High
Pre		Invalid User Account	**Post**		Login Unsuccessful
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Enter invalid email and password and press login button.	Email: abcd@gmail.com Password: Abc123	Login unsuccessful	Login unsuccessful	Pass

6.5.4 Tester Registration

T Sen ID	TS-004	**T C ID**	TC-007
T C Desc	Tester Registration positive test case	**T Prior**	High
Pre	Not a User	**Post**	Tester Account Created

Test Execution Steps:

Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Enter name, phone, email, password and press register button.	Name: ayesha Phone: 1234567 Email: ayesha@gmail.com Password: 12345	Registration Successful	Registration Successful	Pass

T Sen ID	TS-004	**T C ID**	TC-008
T C Desc	Tester Registration negative test case	**T Prior**	High
Pre	Not a User	**Post**	Tester Account Not Created

Test Execution Steps:

Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Enter name, phone, email, password and press register button.	Name: asdfg Phone: 1234 Email: 12345 Password: 12345	Show Error	Error shown	Pass

6.5.5 Client Registration

T Sen ID		TS-005	**T C ID**		TC-009
T C Desc		Client Registration positive test case	**T Prior**		High
Pre		Not a User	**Post**		Client Account Created
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Enter name, phone, email, password and press register button.	Name: anza Phone: 123476 Email: anza@gmail.com Password: 1234	Registration Successful	Registration Successful	Pass

T Sen ID		TS-005	**T C ID**		TC-010
T C Desc		Client Registration negative test case	**T Prior**		High
Pre		Not a User	**Post**		Client Account Not Created
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Enter name, phone, email, password and press register button.	Name: ali Phone: 1234 Email: okh78 Password: 12345	Show Error	Error shown	Pass

6.5.6 Tester's Portfolio

T Sen ID		TS-006	**T C ID**		TC-011
T C Desc		Add personal projects in tester's portfolio	**T Prior**		medium
Pre		Tester logged in	**Post**		Project added
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Enter project name in input field and press Add button.	Web app testing	Project Added Successfully	Project Added Successfully	Pass

T Sen ID		TS-006	**T C ID**		TC-012
T C Desc		Add skill in tester's portfolio	**T Prior**		medium
Pre		Tester logged in	**Post**		Skill added
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Enter skill name in input field and press Add button.	Integration Testing	Skill Added Successfully	Skill Added Successfully	Pass

T Sen ID	TS-006	**T C ID**	TC-013
T C Desc	Add experience in tester's portfolio	**T Prior**	medium
Pre	Tester logged in	**Post**	Experience added

Test Execution Steps:

Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Slide experience slider.	10 years	Experience Added Successfully	Experience Added Successfully	Pass

T Sen ID	TS-006	**T C ID**	TC-014
T C Desc	Write description and delivery in tester's portfolio	**T Prior**	low
Pre	Tester logged in	**Post**	Description and delivery added

Test Execution Steps:

Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Input Description in description field and delivery in delivery field	Description: Hi I am a Tester. Delivery: I deliver Test reports	Description and delivery added Successfully	Description and delivery added Successfully	Pass

T Sen ID	TS-006	**T C ID**	TC-015
T C Desc	Upload Resume/CV in tester's portfolio	**T Prior**	low
Pre	Tester logged in	**Post**	Resume Added

Test Execution Steps:

Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Press upload button, choose file from device and press enter.	Select file from device	Resume Added Successfully	Resume Added Successfully	Pass

6.5.7 Tester Adds Test Case

T Sen ID	TS-007	**T C ID**	TC-016
T C Desc	Tester Adds test case positive test case	**T Prior**	High
Pre	Tester logged in and tester has a project assigned	**Post**	Test Case Added

Test Execution Steps:

Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Select Project, select requirement, input intent, inputs, expected outcome, pre-condition, post -condition and status. Select proof file from device. Press Add test case button	Project: ortho website testing Requirement: Pages Intent: check pages are working Inputs: abc expected outcome: bcd pre-condition: efg post -condition: hij status: pass file selected: img.jpg	Test case added successfully	Test case added successfully	Pass

T Sen ID	TS-007	**T C ID**	TC-017
T C Desc	Tester Adds test case negative test case	**T Prior**	High
Pre	Tester logged in and tester has a project assigned	**Post**	Test Case not Added

Test Execution Steps:

Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Select Project, select requirement, input intent, inputs, expected outcome, pre-condition, post - condition and status. Not Select proof file from device. Press Add test case button	Project: ortho website testing Requirement: Pages Intent: check pages are working Inputs: abc expected outcome: bcd pre-condition: efg post -condition: hij status: pass file selected: none.	Error Shown	Error Shown	Pass

6.5.8 Tester Adds bug

T Sen ID	TS-008	**T C ID**	TC-018
T C Desc	Tester Adds bug	**T Prior**	High
Pre	Tester logged in and tester has a project assigned	**Post**	Bug Added

Test Execution Steps:

Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Select Project, select requirement, Enter Bug description, select bug severity and press Add Bug button.	Project: ortho website testing Requirement: Pages Bug Description: Page not found Severity: High	Bug Added Successfully	Bug Added Successfully	Pass

6.5.9 Tester's Profile Updating

T Sen ID		TS-008	**T C ID**		TC-019
T C Desc		Tester updates his profile	**T Prior**		medium
Pre		Tester logged in	**Post**		Profile Updated
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Change Input field you want to update. Press Update button.	Before: Name: amna After: Name: ali	Profile Update Successfully	Profile Update Successfully	pass

6.5.10 Tester Downloads Project Artifact

T Sen ID		TS-009	**T C ID**		TC-020
T C Desc		Tester Downloads Project Artifact	**T Prior**		High
Pre		Tester logged in and Tester is assigned a project	**Post**		Artifact downloaded
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Click Project Name. Click file icon.	Project Name: ortho website testing	File Downloaded Successfully	File Downloaded Successfully	pass

6.5.11 Tester Submits Test Report

T Sen ID		TS-010	**T C ID**		TC-021
T C Desc		Tester Submits Test Report	**T Prior**		High
Pre		Tester logged in and Tester is assigned a project	**Post**		Report Submitted
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Select Project and Press Submit Test Report Button.	Select Project: ortho website testing	Report Submitted Successfully	Report Submitted Successfully	Pass

6.5.12 Tester Requests Test for Available Project

T Sen ID		TS-011	**T C ID**		TC-022
T C Desc		Tester Requests for Available Project	**T Prior**		Medium
Pre		Tester logged in and project is not already requested by same tester	**Post**		Test Project Requested
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Click Request button on available project.	NA	Requested Successfully	Requested Successfully	Pass

6.5.13 Tester Downloads Submitted Report

T Sen ID	TS-012	**T C ID**	TC-023
T C Desc	Tester downloads submitted report	**T Prior**	Medium
Pre	Tester logged in and test report is submitted	**Post**	Report Downloaded

Test Execution Steps:

Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Select Project and Press download Test Report Button.	Select Project: ortho website testing	Report downloaded Successfully	Report downloaded Successfully	Pass

6.5.14 Tester Deletes bug from Traceability

T Sen ID	TS-013	**T C ID**	TC-024
T C Desc	Tester deletes bug from traceability	**T Prior**	Medium
Pre	Tester logged in and bug is in traceability	**Post**	Bug deleted

Test Execution Steps:

Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Select Project, Select Requirement and Press delete Button in front of bug.	Select Project: ortho website testing requirement: buttons	Bug deleted successfully	Bug deleted successfully	Pass

6.5.15 Tester Deletes Test Case from Traceability

T Sen ID	TS-014	**T C ID**	TC-025
T C Desc	Tester deletes test case from traceability	**T Prior**	Medium
Pre	Tester logged in and test case is available in traceability	**Post**	Test Case deleted

Test Execution Steps:

Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Select Project, Select Requirement and Press delete Button in front of test case.	Select Project: ortho website testing requirement: buttons	Test Case deleted successfully	Test Case deleted successfully	Pass

6.5.16 Tester Updates Test Case

T Sen ID	TS-015	**T C ID**	TC-026
T C Desc	Tester Updates test case	**T Prior**	High
Pre	Tester logged in and test case is available	**Post**	Test Case updated

Test Execution Steps:

Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Select Project, select requirement, Click T C ID. Update the field desired.	Project: ortho website testing Requirement: Pages Before: Intent: check pages are working After: Intent: check pages are not working.	Test case updated successfully	Test case updated successfully	Pass

6.5.17 Client Initiates Project

T Sen ID	TS-016	**T C ID**	TC-027
T C Desc	Client initiates project positive test case	**T Prior**	High
Pre	Client logged in	**Post**	Project Initiation Requested

Test Execution Steps:

Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Enter Project Name, Project Description, Requirements to Test, expected budget, startDate, endDate, message and select file from device.	Project Name: ortho website testing Project Description: test functionality of all buttons and pages Requirements to be Tested: buttons, pages Expected Budget: 150000 Message to Admin: Hi startDate: 23/6/2023 endDate: 27/7/2023 file: SRS.docx	Project Initiated Successfully	Project Initiated Successfully	Pass

T Sen ID		TS-016	**T C ID**		TC-028
T C Desc		Client initiates project positive test case	**T Prior**		High
Pre		Client logged in	**Post**		Project not Initiation Requested
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Enter, Project Description, Requirements to Test, expected budget, startDate, endDate, message and select file from device. (no project name)	Project Description: test functionality of all buttons and pages Requirements to be Tested: buttons, pages Expected Budget: 150000 Message to Admin: Hi startDate: 23/6/2023 endDate: 27/7/2023 file: SRS.docx	Project Can't be Initiated	Project Can't be Initiated	Pass

6.5.18 Client Deletes Test Case from Progress

T Sen ID		TS-017	**T C ID**		TC-029
T C Desc		Client deletes test case from Progress	**T Prior**		Medium
Pre		Client logged in and test case is available in progress	**Post**		Test Case deleted
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Select Project, Select Requirement and Press delete Button in front of test case.	Select Project: ortho website testing requirement: buttons	Test Case deleted successfully	Test Case deleted successfully	Pass

6.5.19 Client Deletes bug from Progress

T Sen ID		TS-018	**T C ID**		TC-030
T C Desc		Client deletes bug from traceability	**T Prior**		Medium
Pre		Client logged in and bug is available in progress.	**Post**		Bug deleted
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Select Project, Select Requirement and Press delete Button in front of bug.	Select Project: ortho website testing requirement: buttons	Bug deleted successfully	Bug deleted successfully	Pass

6.5.20 Client Downloads Project Artifact

T Sen ID		TS-019	**T C ID**		TC-031
T C Desc		Client Downloads Project Artifact	**T Prior**		Medium
Pre		Client logged in	**Post**		Artifact downloaded
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Click Project Name. Click file icon.	Project Name: ortho website testing	File Downloaded Successfully	File Downloaded Successfully	pass

6.5.21 Client Downloads proof of test case

T Sen ID		TS-020	**T C ID**		TC-032
T C Desc		Client downloads proof of test case	**T Prior**		High
Pre		Client logged in and proof is available in progress.	**Post**		Proof downloaded
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Select Project, Select Requirement and Press download Button in front of test case.	Select Project: ortho website testing requirement: buttons	Proof Downloaded successfully	Proof Downloaded successfully	Pass

6.5.22 Client Profile Update

T Sen ID		TS-021	**T C ID**		TC-033
T C Desc		Client updates his profile positive test case	**T Prior**		medium
Pre		Client logged in	**Post**		Profile Updated
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Change Input field you want to update. Press Update button.	Before: Name: Sarah After: Name: Zoha	Profile Update Successfully	Profile Update Successfully	pass

T Sen ID		TS-021	**T C ID**		TC-034
T C Desc		Client updates his profile negative test case	**T Prior**		medium
Pre		Client logged in	**Post**		Profile not Updated
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Change Input field you want to update. Press Update button.	Before: Email: zoha@gmail.com After: Email: Zoha	Error Shown	Error Shown	pass

6.5.23 Client Contacts Admin

T Sen ID		TS-022	**T C ID**		TC-035
T C Desc		Client Contacts Admin positive test case	**T Prior**		medium
Pre		Client logged in	**Post**		Message sent
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Enter Name, Email, Select Project, write message, select file and press submit button.	Name: Saqib Email: saqib@gmail.com Select Project: Project1 write message: Hi select file: srs.docx	Message Sent	Message Sent	pass

T Sen ID		TS-021	**T C ID**		TC-036
T C Desc		Client Contacts Admin negative	**T Prior**		medium
Pre		Client logged in	**Post**		Message not sent
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Enter Name, Email, Select Project, write message and press submit button. (Not selecting file)	Name: Saqib Email: saqib@gmail.com Select Project: Project1 write message: Hi	Error Shown	Error Shown	pass

6.5.24 Client Downloads Submitted Report

T Sen ID		TS-023	**T C ID**		TC-037
T C Desc		Client downloads submitted report	**T Prior**		High
Pre		Client logged in and test report is submitted	**Post**		Report Downloaded
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Select Project and Press download Test Report Button.	Select Project: ortho website testing	Report downloaded Successfully	Report downloaded Successfully	Pass

6.5.25 Admin Accepts Client Request

T Sen ID		TS-024	**T C ID**		TC-038
T C Desc		Admin Accepts Client Request	**T Prior**		medium
Pre		Admin logged in.	**Post**		Request Accepted
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Click Accept Button	NA	Request Accepted	Request Accepted	pass

6.5.26 Admin rejects Client Request

T Sen ID		TS-025	**T C ID**		TC-039
T C Desc		Admin Rejects Client Request	**T Prior**		medium
Pre		Admin logged in.	**Post**		Request Rejected
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Click Reject Button	NA	Request Reject	Request Reject	pass

6.5.27 Admin Accepts Tester Request

T Sen ID		TS-026	**T C ID**		TC-040
T C Desc		Admin Accepts Request	**T Prior**		medium
Pre		Admin logged in.	**Post**		Request Accepted
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Click Accept Button	NA	Request Accepted	Request Accepted	pass

6.5.28 Admin Rejects Tester Request

T Sen ID	TS-027	**T C ID**	TC-041
T C Desc	Admin Rejects Tester Request	**T Prior**	medium
Pre	Admin logged in.	**Post**	Request Rejected

Test Execution Steps:

Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Click Reject Button	NA	Request Reject	Request Reject	pass

6.5.29 Admin deletes Client Account

T Sen ID	TS-028	**T C ID**	TC-042
T C Desc	Admin deletes client account	**T Prior**	medium
Pre	Admin logged in and client account is there to be deleted.	**Post**	Account deleted

Test Execution Steps:

Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Click delete icon from Clients table.	NA	Client Deleted	Client Deleted	pass

6.5.30 Admin Deletes Tester Account

T Sen ID	TS-029	**T C ID**	TC-043
T C Desc	Admin deletes Tester account	**T Prior**	medium
Pre	Admin logged in and tester account is there to be deleted.	**Post**	Account deleted

Test Execution Steps:

Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Click delete icon from tester table.	NA	Tester Deleted	Tester Deleted	pass

6.5.31 Admin Downloads Submitted Report

T Sen ID	TS-030	**T C ID**	TC-044
T C Desc	Admin downloads submitted report	**T Prior**	Medium
Pre	Admin logged in and test report is submitted	**Post**	Report Downloaded

Test Execution Steps:

Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Select Project and Press download Test Report Button.	Select Project: ortho website testing	Report downloaded Successfully	Report downloaded Successfully	Pass

6.5.32 Admin Downloads Project Artifact

T Sen ID		TS-031	**T C ID**		TC-045
T C Desc		Admin Downloads Project Artifact	**T Prior**		Medium
Pre		Admin logged in	**Post**		Artifact downloaded
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Click Project Name. Click file icon.	Project Name: ortho website testing	File Downloaded Successfully	File Downloaded Successfully	pass

6.5.33 Admin Profile Update

T Sen ID		TS-032	**T C ID**		TC-046
T C Desc		Admin updates his profile positive test case	**T Prior**		medium
Pre		Admin logged in	**Post**		Profile Updated
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Change Input field you want to update. Press Update button.	Before: Name: Admin After: Name: Zoha	Profile Update Successfully	Profile Update Successfully	pass

T Sen ID		TS-032	**T C ID**		TC-047
T C Desc		Admin updates his profile negative test case	**T Prior**		medium
Pre		Admin logged in	**Post**		Profile not Updated
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Change Input field you want to update. Press Update button.	Before: Email: admin@tester-online.live After: Email: Admin	Error Shown	Error Shown	pass

6.5.34 Admin Deletes Test Case

T Sen ID		TS-033	**T C ID**		TC-048
T C Desc		Admin deletes test case from Progress	**T Prior**		Medium
Pre		Admin logged in and test case is available in progress	**Post**		Test Case deleted
Test Execution Steps:					
Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Select Project, Select Requirement and Press delete Button in front of test case.	Select Project: ortho website testing requirement: buttons	Test Case deleted successfully	Test Case deleted successfully	Pass

6.5.35 Admin Deletes Bug

T Sen ID	TS-034	**T C ID**	TC-049
T C Desc	Client deletes bug from traceability	**T Prior**	Medium
Pre	Client logged in and bug is available in progress.	**Post**	Bug deleted

Test Execution Steps:

Sno.	Action	Inputs	Expected Outcome	Actual Outcome	Test Result
1	Select Project, Select Requirement And Press delete Button in front of bug.	Select Project: ortho website testing requirement: buttons	Bug deleted successfully	Bug deleted successfully	Pass

CHAPTER - 7

CONCLUSION AND OUTLOOK

7 Conclusion

We present conclusion and outlook in this chapter.

7.1 Conclusions

Our web application marketplace represents a significant step towards addressing the challenges facing the software testing industry. Through the platform, software testers can register and access remote testing job opportunities, while companies can find suitable testers for their projects. This approach will improve the efficiency and quality of testing services while providing a centralized marketplace for both testers and companies.

Additionally, the platform offers several unique features such as input test data generation, test case management, and traceability metrics, which will enhance the testing process and enable effective testing. The platform will play an instrumental role in reducing the gap between software companies and testers, making it easier for companies to access quality testing services while offering job opportunities to testers.

As the software testing industry continues to grow, our web application marketplace will play a vital role in connecting software testers with companies in need of their services. We look forward to seeing how this platform will contribute to the future of software testing and the broader software industry.

7.2 Outlook

As a future direction, automated test case generation using AI methods such as use of reinforcement learning, and a mechanism that allows test case prioritization. [3]–[5]

References

[1] H. Tahbildar and B. Kalita, "Automated Software Test Data Generation: Direction of Research," *International Journal of Computer Science & Engineering Survey*, vol. 2, no. 1, pp. 99–120, Feb. 2011, doi: 10.5121/ijcses.2011.2108.

[2] A. Anand and A. Uddin, "Importance of Software Testing in the Process of Software Development Load Balancing In Cloud Computing View project Deployment of DataBase-as-a-Service and connecting it with the Local Server View project Importance of Software Testing in the Process of Software Development," 2019. [Online]. Available: www.ijsrd.com

[3] E. Ashraf, K. Mahmood, T. A. Khan, and S. Ahmed, "Value based PSO Test Case Prioritization Algorithm," 2017. [Online]. Available: www.ijacsa.thesai.org

[4] F. S. Ahmed, A. Majeed, and T. A. Khan, "Value-Based Test Case Prioritization for Regression Testing Using Genetic Algorithms," *Computers, Materials and Continua*, vol. 74, no. 1, pp. 2211–2238, 2023, doi: 10.32604/cmc.2023.032664.

[5] F. S. Ahmed, A. Majeed, T. A. Khan, and S. N. Bhatti, "Value-based cost-cognizant test case prioritization for regression testing," *PLoS One*, vol. 17, no. 5 May, May 2022, doi: 10.1371/journal.pone.0264972.

Appendix A

Ethical Analysis

When we talk about ethics it means all the ethical concerns to be considered. To define it in more precisely we categorize it into two parts.

- FYP Process Level
- FYP Product Level

FYP Process Level

By Process Level we mean the ethical concerns that we face or to be considered while developing product. It includes the inside management and inside problems.

Honesty

It is micro ethics based on individualism we both partners remain honest and loyal to each other in developing this project.

Integrity

We keep all the engineering ethics in our mind while developing this product and we remain loyal to engineering code of ethics. We remain honest to society as our product does not provide any harm to public.

Fairness

We make all the things clean and clear to ourselves and to the supervisor. There is no ambiguity between any of the instance of this project.

Conflict Resolution

We have clearly defined the process and terms and conditions for resolving the conflict that may arise during product development.

Morality

Some of the basic ethics are set as standard to be follow like the communication process, weekly meetings standards etc. and ethics like Don't Cheat, Don't Lie.

Right Ethics

All the right ethics are considered to be follow and adopt during the product development.

This includes Privacy, Faithfulness, Respect, Honesty etc.

FYP Product Level

When we say product level, we mean the concerns that will be developed after the product is in market, Launched or on launching pad. We have prepared ourselves also for that and all the ethical concerns that we considered will be discussed here.

As the project is market viable product so we have to be more careful about its launching and operability.

Following are the ethical concerns about our project that we considered important to be discussed.

1. **Fair treatment of testers**

- Avoiding discrimination in job opportunities and pay rates.
- Ensuring fair treatment of all testers, regardless of gender, race, or other factors.
- Providing equal opportunities for all testers, regardless of their circumstances or disabilities.

2. **Privacy and security of user information**

- Safeguarding personal and sensitive information from unauthorized access or breaches.
- Transparency in data collection practices.
- Obtaining user consent before collecting any personal information.

3. **Quality of testing process and results**

- Ensuring that testers are appropriately qualified for assigned projects.

- Promoting ethical standards and industry best practices.
- Avoiding rushed or incomplete testing due to underpayment or overwork.

4. Fair and competitive marketplace

- Ensuring that companies do not use the platform to undercut market rates or exploit testers.
- Preventing collusion among testers to artificially inflate rates.
- Promoting a fair and competitive marketplace for both testers and companies.

5. Accessibility for all users

- Designing the platform with accessibility in mind.
- Ensuring that testers with disabilities can access and complete testing projects without barriers.
- Providing equal opportunities for all testers, regardless of disabilities or accessibility needs.

WBS (Work Break Down Structure)

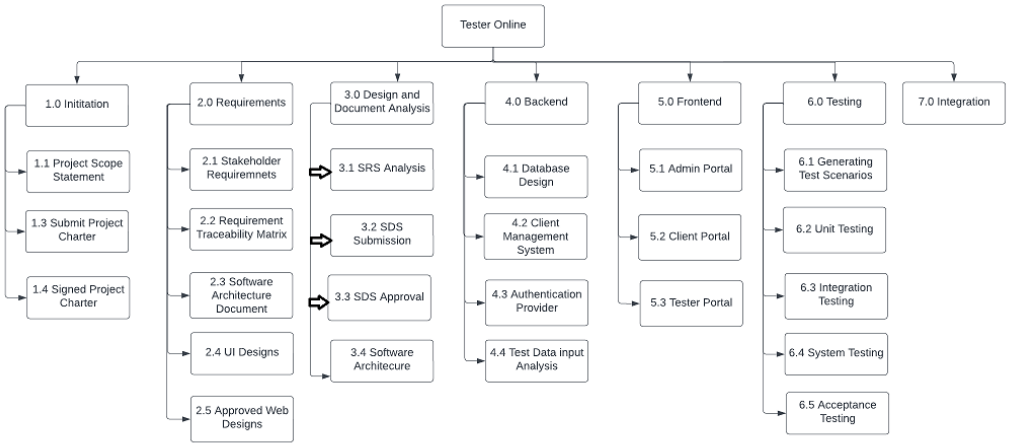

Figure 0-1 Work Break Down Structure