Daffodil International University

SOHOJOG: A Collaborative Platform For All

Submitted By

Muhammad Shafayet Ullah

201-35-551

Supervised By

Md Rajib Mia

Senior Lecturer, Department of Software Engineering

This project report has been submitted in fulfillment of the requirements for the degree of **Bachelor of Science in Software Engineering**

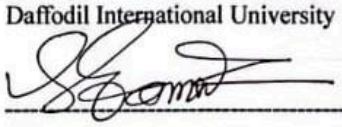
APPROVAL

This project titled on "**SOHOJOG: A Collaborative Platform For All**", submitted by **Muhammad Shafayet Ullah (ID: 201-35-551)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS

Chairman

Dr. Imran Mahmud
Associate Professor & Head
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University



Internal Examiner 1

Nuruzzaman Faruqi
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 2

Md. Rajib Mia
Lecturer (Senior Scale)
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

External Examiner

Md. Fazle Munim
Associate Director & Vice President
Government & Public Sector
Ernst & young (EY)

SUPERVISOR's DECLARATION

I/We* hereby declare that I/We* have checked this project and in my/our* opinion, this project is adequate in terms of scope and quality for the award of the degree of Bachelor of Science.

_____

(Supervisor's Signature)

Full Name : **Md Rajib Mia**

Position : Senior Lecturer, Department of Software Engineering

Date : 22-01-2025

STUDENT'S DECLARATION

I hereby declare that the work in this project is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Daffodil International University or any other institution.

_____

(Student's Signature)

Full Name : **Muhammad Shafayet Ullah**
ID Number : 201-35-551
Date : 31 December 2024

SOHOJOG

Muhammad Shafayet Ullah

The project submitted in fulfillment of the requirements
for the award of the degree of
Bachelor of Science/Master of Science

Department of Software Engineering

DAFFODIL INTERNATIONAL UNIVERSITY

DECEMBER 2024

ACKNOWLEDGEMENTS

All glory and thanks are owed to Allah Ta'ala, whose innumerable favors and direction have made it possible for us to successfully finish this project.

Our esteemed supervisor, Md. Rajib Mia, Senior Lecturer in the Department of Software Engineering, has our sincere gratitude. Throughout this process, his insight, support, and priceless advice have been crucial. His confidence in our ability inspired us to set higher goals and endure hardships.

We also want to express our sincere gratitude to everyone who helped and encouraged us throughout this effort. Your contributions, whether in the form of counsel, inspiration, or useful assistance, have streamlined and enhanced the significance of this undertaking.

We sincerely thank the Department of Software Engineering faculty members for their unwavering support and encouragement, which have been essential to our development and achievement.

Lastly, we would like to express our sincere gratitude to our parents. Everything we have accomplished is based on your efforts, prayers, and unwavering love. Throughout this journey, you have been our inspiration and our strength.

We sincerely thank everyone who supported us, no matter how small the contribution. This achievement is a result of your encouragement and faith in us

DEDICATION

I therefore declare that I have done this project under the oversight of *Md Rajib Mia, Senior Lecturer*, Department of Software Engineering, Daffodil International University. Also declare that neither entire record nor any portion of this record has been submitted somewhere else for my degree.

ABSTRACT

The SOHOJOG project is a task management system made to improve productivity and facilitate teamwork in projects. Offering resources to assign tasks, establish teams, and supervise daily operations, enables users to efficiently build and manage projects.

To assign tasks that are specific to their roles and responsibilities, a user can invite team members and start a project. The platform facilitates team creation, which makes it possible for participants to divide the labor effectively. Clarity, responsibility, and better coordination are guaranteed in the project environment with this methodical approach.

A notable characteristic of SOHOJOG is its integrated chat system, which makes communication easy. Within certain teams or throughout the project, participants can effectively communicate, share updates, and have real-time conversations. This centralizes all project-related communications and does away with the need for external tools.

With its emphasis on streamlining task distribution and encouraging collaboration, SOHOJOG hopes to become a vital resource for people and businesses seeking to improve project management and promote cooperation.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

1.1 Background

Context and Relevance

Efficient project management is essential for both individuals and organizations in today's connected and fast-paced environment. Digital tools are being used more and more by industries worldwide to boost teamwork, assign tasks more effectively, and guarantee smooth communication. Nevertheless, even though there are many task management solutions available, many of them don't take a unified approach to handling job allocation and real-time communication, which results in notable gaps in coordination and productivity.

Problem Identification

By providing a strong and intuitive framework for collaborative task and project management, the SOHOJOG project fills this gap. It attempts to address a frequent problem: keeping project participants productive and in sync while reducing communication overhead. Few of the current tools successfully combine these features; most of them concentrate on communication or task distribution. Because of this, users usually turn to juggling several apps at once, which can result in inefficiencies and poor management.

Purpose and Justification

The main objective of SOHOJOG is to offer a cohesive project management solution by combining task delegation, team building, and an integrated chat system. This improves project management's accessibility and efficiency by streamlining the workflow and doing away with the requirement for third-party applications. In one location, the platform guarantees that users can easily assign daily tasks, invite participants, develop projects, and interact.

Scope

Project development, team building, work distribution, participant administration, and real-time communication are all included in SOHOJOG's scope. Both individuals' and teams' demands are met by consolidating these features into a single platform, which facilitates easier collaboration and increases output. The built-in chat system also

promotes improved communication by allowing users to exchange updates and quickly address problems.

SOHOJOG is well-positioned to close the gap between communication and task management by providing a useful and pertinent solution in the current digital era. Its all-encompassing project management methodology makes it an effective tool for tackling the difficulties encountered by contemporary teams.

1.2 Project Planning and Initiation

To ensure effective teamwork, track team progress, and organize activities, a project management tool is necessary. Workflows are streamlined, communication is enhanced, and project deadlines are met. The success of a project in the fast-paced workplace of today depends on having a dependable platform to handle every facet of it. For this reason, SOHOJOG is a crucial tool for efficient project administration.

1.2.1 Preliminary Analysis & Project Scope Definition:

For SOHOJOG, the project scope involves creating a user-friendly, web-based platform designed to facilitate efficient project management. It addresses the need for task allocation, team collaboration, and progress tracking. The platform is tailored to users with varying technical expertise, ensuring ease of use while offering customizable features and providing insights into task performance.

Key features such as task management, real-time collaboration tools, and customizable dashboards are prioritized in this stage to meet the project's goals. By clearly defining the project's scope, the development process remains focused, ensuring the delivery of a valuable tool for users.

Project Scope

- Enable users to create and manage multiple projects.

- Allow project owners to assign daily tasks to individuals or teams and monitor progress.
- Include real-time collaboration features such as built-in chat for seamless communication among team members.
- Support file sharing for effective knowledge and resource exchange.
- Provide visual tools to track project progress and task statuses.
- Allow users to customize dashboards to suit individual or team preferences for better project insights.
- Ensure the platform is intuitive and easy to navigate for users with varying levels of technical expertise.

1.2.2 Market Feasibility Analysis:

- **Increasing Demand:** There is a growing demand for easy-to-use project management tools, particularly among small businesses, freelancers, and educational institutions who need cost-effective solutions.
- **Remote Work & Collaboration:** The rise of remote work and digital collaboration has intensified the need for platforms that streamline task management and foster effective communication within teams.
- **Target Market Fit:** SOHOJOG is designed to meet these demands, offering an intuitive and accessible solution tailored to its target market, making it a strong fit in a rapidly expanding market.

1.2.3 Technical Feasibility Analysis:

- Frontend: Next.js, paired with Redux Toolkit for efficient state management, and Tailwind CSS for responsive, consistent styling across platforms.
- Backend: NestJS framework with PostgreSQL for managing relational data, and Prisma ORM for streamlined database schema handling.
- Infrastructure: Hosted on DigitalOcean, utilizing Docker for consistent environments across development and production, and Jenkins for Continuous Integration/Continuous Deployment (CI/CD) automation.

Key Features

- Authentication: Secure, JWT-based authentication, integrated with OAuth for Google and Facebook login options.
- APIs: RESTful API design with built-in security features, including rate limiting and input validation to ensure system security.
- Data Management: A robust data schema managed via Prisma, covering core entities like users, tasks, and projects for efficient data handling.

1.2.4 Financial Feasibility Analysis

Project Budget Overview

The financial plan prioritizes cost-effective resources during development and testing, with moderate expenses for the initial launch to ensure efficient budget management.

Cost Breakdown

- Hosting: DigitalOcean (Free for testing, $10/month for launch)
- Image Storage: Cloudinary (Free for testing, up to $60 for launch)
- Database: PostgreSQL (Free for testing, $9/month for launch)
- Development Tools: Free open-source resources
- Domain: $10-$15 annually

Total Estimated Costs

- Development/Testing: $10-$75 annually
- Launch: $298-$303 annually

1.3 Target User Profile and Tentative Elicitation Process

1.3.1 Target User

- **Small Business Owners**: Professionals managing small teams, seeking affordable project management solutions.
- **Freelancers and Consultants**: Independent professionals needing flexibility for managing multiple client projects.

- **Non-Profit Organizations**: Budget-conscious organizations requiring accessible project coordination tools.
- **Educational Institutions**: Schools and universities needing simple tools for academic and administrative projects.
- **Creative Teams**: Design, media, and production teams needing tools to support creative workflows.
- **Event Organizers**: Individuals or teams managing event planning tasks, seeking flexible project tracking tools.

User Types:

- **Manager**: Manages the project, assigns tasks, and oversees progress.
- **Participant**: Joins the project and works on assigned tasks.

1.3.2 User profile

Table 1.1: User Profile for Manager

User Class	**Note on Characteristics**
Type of user	Project Managers
Age range	25–50 years
Frequency of use	Daily
Mandatory	Yes
Computer experience	Intermediate
Education	High school diploma or higher; experience-based users are included
goal	Efficient project management
Language skills	Basic to proficient in English
Number of users	One per project
Training	Minimal; designed for intuitive use
Others system use	Trello, ClickUp
Way of working	Leadership and decision-making oriented

Table 1.2: User Profile for Participants

User Class	**Note on Characteristics**
Type of user	Participant
Age range	18–50 years
Frequency of use	Daily or as per task deadlines
Mandatory	Yes
Computer experience	Basic to intermediate

Education	High school, diploma or relevant experience or higher
goal	Execute assigned tasks efficiently and contribute to project success
Language skills	Basic to proficient in English
Number of users	Larger than managers; varies with project size
Training	Minimal; intuitive features minimize the learning curve
Others system use	ClickUp, Trello
Way of working	Task-oriented and collaborative

1.3.3 Elicitation Process

Requirement Gathering

- **Conversations:** Talk to key stakeholders to understand their needs and expectations.
- **Surveys:** Use quick surveys to collect user preferences and insights.
- **Group Sessions:** Organize small discussions with users to brainstorm and refine ideas.

User Stories and Use Cases

- Write simple stories about how users might interact with the system to ensure their needs are addressed.

Prototyping and Feedback

- Create rough sketches and mockups to visualize ideas and gather feedback from stakeholders early.

Requirement Finalization

1. Document key features and ensure all must-haves are captured.
2. Review and confirm with stakeholders to align expectations.

2.4 Project Block Diagram

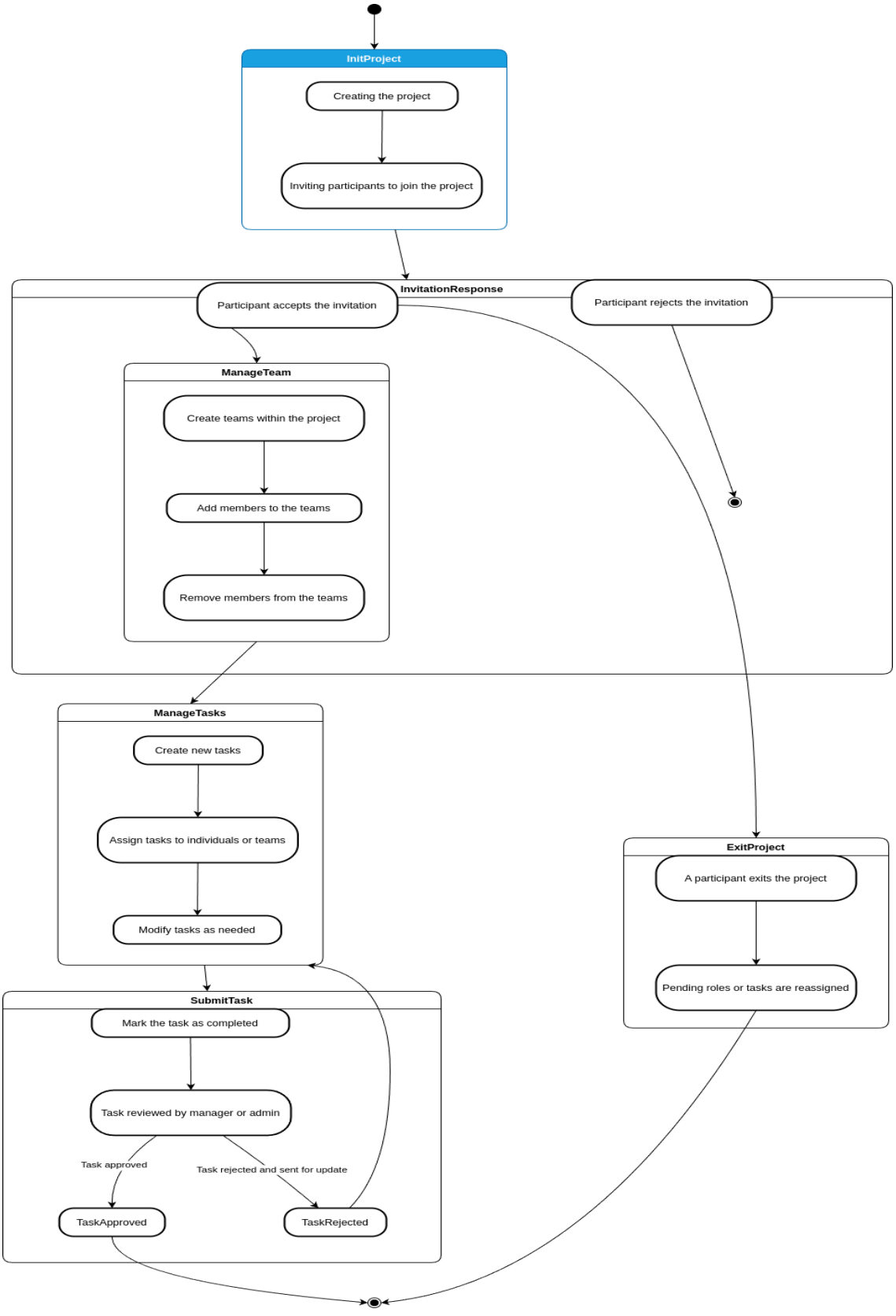


Figure 1: System Block Diagram

1.5 System Requirements

1.5.1 Hardware Requirements

Mobile Devices:

- RAM: 2 GB or more
- Storage: At least 100 MB of free space (additional space may be required for live messaging).

Web Browsers (Desktop):

- RAM: 4 GB or more
- Storage: At least 50 MB for caching (increased usage for live messaging).

1.5.2 Software Requirements

Web Browsers: The latest versions of Chrome, Firefox, Safari, or Edge.

Operating Systems: The platform is accessible on both desktop and mobile browsers, supporting:

- Desktop: Windows 10 or later, macOS 10.15 or later, or Linux.
- Mobile: iOS 11.0 or later, Android 7.0 or later.

1.5.3 Constraints and Dependencies

Internet Connection: A stable internet connection is required to use the platform, especially for live messaging and real-time updates.

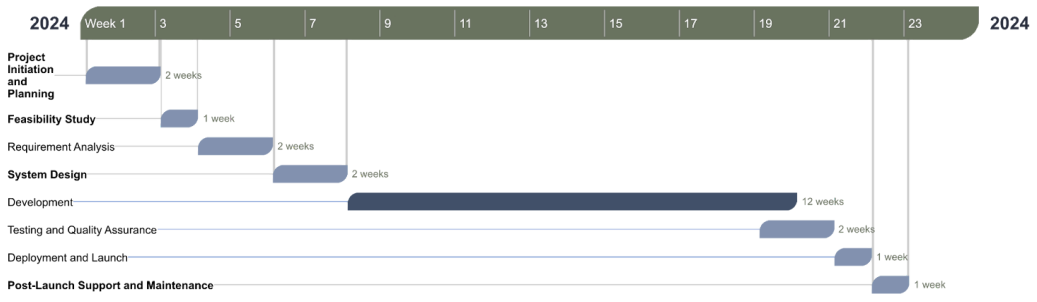
Mobile Device Storage: Users may need extra storage for live messaging and media files, depending on usage.

External Services: The platform depends on third-party services for authentication (Google, Facebook) and real-time messaging (WebSocket or similar technology).

1.6 Project Scheduling

Phase	Duration (Weeks)
Project Initiation and Planning	Week 1
Feasibility Study	Week 1
Requirement Analysis	Weeks 2-3
System Design	Weeks 4-5
Development	Weeks 6-20
Testing and Quality Assurance	Weeks 21-22
Deployment and Launch	Week 23
Post-Launch Support and Maintenance	Weeks 23-25

Gantt Chart



1.7 Summary

This project aims to provide a robust yet user-friendly task and project management platform to small enterprises, independent contractors, nonprofit organizations, and educational institutions. The platform will help boost productivity and efficiency by offering crucial capabilities for job management, real-time team collaboration, and customized dashboards. Hardware and software needs will be carefully considered to ensure seamless operation across various devices. The design and functionality of the platform will be heavily influenced by user input gathered through workshops, questionnaires, and interviews.

CHAPTER 2 DESIGN AND IMPLEMENTATION

2.1 Introduction

The design and execution of the project management platform are the main topics of this chapter. It describes the platform's technical development methods, architectural decisions, and UI design. With features that enable smooth task management, teamwork, and progress monitoring, the objective is to provide a solid, scalable, and user-focused solution.

2.2 Functional Requirements

This section lists the essential features that the project management platform has to have. Project creation and management tools, work delegation tools, real-time communication tools, and progress monitoring tools are among them. Every feature is made to be efficient, user-friendly, and in line with their demands.

FR01	**Registration**
Description	Enables users to create accounts through email/password or OAuth integrations (e.g., Google).
Stakeholder	Managers, participants

FR02	Login
Description	Allows users to log in securely using email/password or OAuth methods (e.g., Google).
Stakeholder	Managers, participants

FR03	Profile Management
Description	Allows users to manage their profiles by updating personal information, such as name, contact details, and profile settings.
Stakeholder	Managers, participants

FR04	Password Reset
Description	Provides users the ability to reset their password securely through an email-based OTP verification process.
Stakeholder	Managers, participants

FR05	Project Creation and Configuration
Description	Allows users to initiate new projects, configure essential details such as name, description, start and end dates
Stakeholder	Managers

FR06	Update Project Information
Description	Enables project managers to edit project details such as title, description, and deadlines to reflect changes or updates.
Stakeholder	Managers

FR07	Send Invitation to Join Project
Description	Allows project managers to send invitations to other users (team members or stakeholders) to join a project by providing their emai.
Stakeholder	Managers

FR08	Respond to Project Invitation
Description	Allows participants to accept or decline invitations to join a project sent by the project manager
Stakeholder	Participants

FR09	Create Team
Description	Enables project managers to create teams within a project, assign members to teams, and set team-specific roles and responsibilities.
Stakeholder	Manager

FR10	Add Member to Team
Description	Allows project managers to add new members to existing teams, assigning them specific roles and responsibilities within the team.
Stakeholder	Manager

FR11	Remove Member from Team
Description	Allows the project manager to remove a team member from the project team.
Stakeholder	Manager

FR12	Create Task and Assign
Description	Allows project managers to create tasks with relevant details (e.g., title, description, due date) and assign them to specific team members.
Stakeholder	Manager

FR13	Submit Task by Participant
Description	Allows participants to mark tasks as completed and submit them for review by the project manager or other responsible parties.

Stakeholder	Participant

FR14	**Chat**
Description	Enables real-time messaging between project team members for collaboration, discussions, and updates.
Stakeholder	Manager, Participant

2.3 Non-Functional Requirements

Category	Requirement	Possible Way
Performance	Handle up to 500 concurrent users and load within 3 seconds.	Optimize queries, implement caching, and use a scalable infrastructure like load balancers.
Security	Encrypt user data in transit and at rest.	Use HTTPS for data transmission and AES-256 for data storage encryption.
Reliability	Ensure 99.9% uptime with minimal downtime.	Use cloud hosting with redundancy and monitor uptime using services like Pingdom or UptimeRobot.
Scalability	Allow room for moderate user growth.	Use a flexible hosting plan that can handle small increases in traffic.
Usability	Make the system easy to navigate for non-technical users.	Use clear labels, and simple workflows, and gather user feedback for iterative improvements.

2.3.1 Performance

- **Requirement:** Handle up to 500 concurrent users and load within 3 seconds.
- **Description:** Optimize queries, implement caching, and use a scalable infrastructure like load balancers.

2.3.2 Reliability

- **Requirement:** Ensure 99.9% uptime with minimal downtime.
- **Description:** Use cloud hosting with redundancy and monitor uptime using services like Pingdom or UptimeRobot.

2.3.3 Portability

- **Requirement:** Multi-Device Accessibility
- **Description:** Ensure usability on both desktop and mobile devices.

2.3.3 Security

- **Requirement:** Encrypt user data in transit and at rest.
- **Description:** Use HTTPS for data transmission and AES-256 for data storage encryption.

2.3.3 Usability

- **Requirement:** Make the system easy to navigate for non-technical users.
- **Description:** Use clear labels, and simple workflows, and gather user feedback for iterative improvements.

2.4 Object-oriented System design using UML

2.4.1 Use Case Diagram

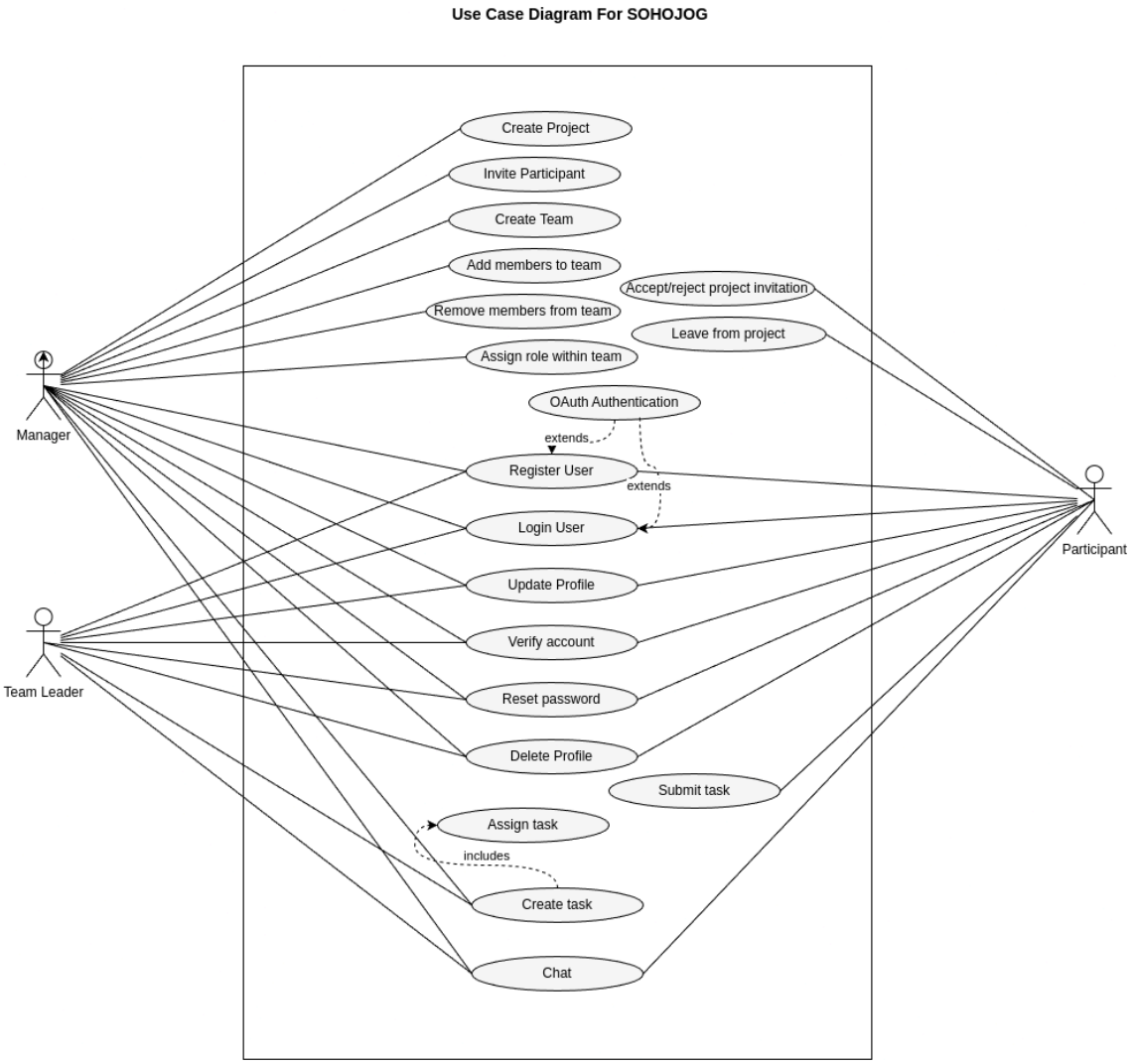


Figure 2: Use Case Diagram

2.4.2 Case Description

Case Description-01: Registration

Use Case	Registration
Goal	Users can register to sign in to the system.
Precondition	Users must access the system through the website.

Success End Condition	A new user account is created, and a confirmation email is sent to the user.
Failed End Condition	The user is informed of invalid input or a duplicate email, and no account is created.
Primary Actors: Secondary Actors:	All User
Trigger	User will request a registration form to fill up
Description / Main Success Scenario	1. User enters email/password. 2. System validates input. 3. System checks password criteria. 4. System checks email uniqueness. 5. (If valid/unique) System creates account. 6. System sends confirmation email.
Alternative Flows	● Invalid Input (from 2/3): Display error; return to 1. ● Duplicate Email (from 4): Prompt for different email; return to 1.
Quality Requirements	● Efficient input validation. ● Clear error messages. ● Prompt confirmation email. ● Handles concurrent registrations.

Case Description-02: Login

Use Case	Login
Goal	Authenticate a user using email and password.
Precondition	The user is registered and on the login page/interface.

Success End Condition	User has been given access to the system.
Failed End Condition	Error message displayed (invalid input, unregistered email, or incorrect password).
Primary Actors: Secondary Actors:	All User
Trigger	The user clicks the login button.
Description / Main Success Scenario	7. User inputs email/password. 8. System validates input. 9. System checks email in database. 10. System checks password. 11. (If email exists and password correct) System generates authentication tokens. 12. System sends tokens to user.
Alternative Flows	● Invalid Input (from 2): Display "Invalid input" error. ● Email Not Registered (from 3): Display "Email not registered" error. ● Incorrect Password (from 4): Display "Incorrect password" error.
Quality Requirements	● Secure authentication token generation. ● Fast login response time. ● Clear and informative error messages. ● Protection against brute-force attacks.

Case Description-03: Verify email

Use Case	Email Verification with OTP
Goal	Verify a user's email address using a One-Time Password (OTP).
Precondition	User has received a verification email containing an OTP.

Success End Condition	Email is marked as verified, and the account is activated if it was inactive.
Failed End Condition	Error message displayed (invalid OTP format, invalid OTP, or expired OTP).
Primary Actors: Secondary Actors:	All User
Trigger	User enters the OTP.
Description / Main Success Scenario	13. User enters OTP. 14. System validates OTP format. 15. System checks OTP validity (existence and match). 16. System checks OTP expiry. 17. (If format valid, OTP exists/matches, and not expired) System marks email as verified. 18. System checks if account is active. 19. (If account inactive) System activates account.
Alternative Flows	● Invalid OTP Format (from 2): Display "Invalid OTP Format" error. ● Invalid OTP (from 3): Display "Invalid OTP" error. ● OTP Expired (from 4): Display "OTP Expired" error.
Quality Requirements	● Secure OTP generation and delivery. ● Timely OTP expiry. ● Clear and informative error messages. ● Protection against brute-force attacks on OTP entry.

Case Description-04: Password Reset

Use Case	Password Reset
Goal	Allow users to reset their password via email and OTP.
Precondition	User is registered.

Success End Condition	User's password is updated.
Failed End Condition	Error message displayed (email not found, invalid OTP, or invalid new password).
Primary Actors: Secondary Actors:	All User
Trigger	User requests password reset.
Description / Main Success Scenario	20. User requests password reset. 21. User enters email address. 22. System checks email existence. 23. (If email exists) System sends OTP email. 24. User receives OTP email. 25. User enters OTP. 26. System validates OTP. 27. (If OTP valid) User enters new password. 28. System validates new password. 29. (If new password valid) System updates password. 30. User confirms password reset.
Alternative Flows	● Email Not Found (from 3): Show error message. ● Invalid OTP (from 7): Show "Invalid OTP" message; request new OTP; receive new OTP; enter new OTP; validate OTP. ● Invalid New Password (from 9): Show "Invalid Password" message; re-enter new password; validate new password.
Quality Requirements	● Secure OTP generation and delivery. ● Robust password validation rules. ● Clear and informative error messages. ● Protection against brute-force attacks.

Case Description-04: OAuth Login

Use Case	OAuth Login
Goal	Allow users to log in using an OAuth provider (e.g., Google).

Precondition	User is on the login page.
Success End Condition	User is logged in to the system.
Failed End Condition	Error throw
Primary Actors: Secondary Actors:	All User OAuth Provider
Trigger	User chooses an OAuth provider.
Description / Main Success Scenario	31. User chooses OAuth provider. 32. System redirects to OAuth provider. 33. OAuth provider authenticates user. 34. OAuth provider returns authorization code. 35. System exchanges code for access token. 36. System retrieves user information from OAuth provider. 37. System checks if user exists in the system. 38. (If user exists) System logs in existing user. 39. (If user does not exist) System creates new user account and logs in.
Alternative Flows	● Network Error: Show Error Page
Quality Requirements	● Secure handling of authorization codes and access tokens. ● Seamless redirection to and from the OAuth provider. ● Efficient retrieval of user information. ● Handles different OAuth providers.

Case Description-05: Update Account

Use Case	Update Account Information
Goal	Allow users to update their account information.

Precondition	User is logged in and has access to the account update page.
Success End Condition	Account information is updated, and confirmation is displayed.
Failed End Condition	Error message is displayed due to invalid input; user is prompted to correct the input.
Primary Actors: Secondary Actors:	All User
Trigger	User requests to update account information.
Description / Main Success Scenario	40. User requests update. 41. System displays update form. 42. User inputs updated information. 43. System validates input. 44. (If input valid) System updates account information. 45. System displays confirmation of update.
Alternative Flows	● Invalid Input: System displays error message; prompts user to correct input.
Quality Requirements	● Robust input validation. ● Clear and informative error messages. ● Secure handling of updated information.

Case Description-06: Account Deletion

Use Case	Account Deletion
Goal	Allow users to delete their accounts.

Precondition	User is logged in and has access to the account deletion page.
Success End Condition	Account is deleted.
Failed End Condition	Error message is displayed due to invalid input; user is prompted to correct the input.
Primary Actors: Secondary Actors:	All User
Trigger	User requests account deletion.
Description / Main Success Scenario	1. User requests account deletion. 2. System prompts user for authentication. 3. User provides authentication information. 4. System validates authentication information. 5. (If authentication valid) System confirms deletion request with the user. 6. (If user confirms deletion) System deletes the account.
Alternative Flows	● Invalid Authentication (from 4): System displays error message; prompts user to re-authenticate or cancel. ● Deletion Not Confirmed (from 5): System cancels the deletion process.
Quality Requirements	● Robust input validation. ● Clear and informative error messages. ● Secure handling of updated information.

Case Description-07: Create Project

Use Case	Create Project
Goal	Create a new project in the system.

Precondition	User is logged in.
Success End Condition	Project is created, and a confirmation message is displayed.
Failed End Condition	Error message is displayed due to invalid project details; user is prompted to correct the details.
Primary Actors: Secondary Actors:	All User
Trigger	Admin requests to create a project.
Description / Main Success Scenario	1. Admin requests project creation. 2. System displays project creation form. 3. Admin enters project details. 4. System validates project details. 5. (If details valid) System creates new project. 6. System creates project wallet. 7. System displays confirmation of project creation.
Alternative Flows	● Invalid Details (from 4): System displays error message; prompts admin to correct details.
Quality Requirements	● Robust validation of project details. ● Clear and informative error messages. ● Confirmation message to ensure successful creation.

Case Description-08: Invite User

Use Case	Create Project
Goal	Allow a project manager to invite a user to the project.

Precondition	User is logged in.
Success End Condition	Invitation is sent to the target user..
Failed End Condition	Error message is displayed.
Primary Actors: Secondary Actors:	All User(project manager)
Trigger	Project manager initiates the invitation process within a project.
Description / Main Success Scenario	1. Project manager initiates invitation process (within the project context). 2. Project manager enters the new user's email address. 3. System sends project invitation email to the provided email address. 4. System confirms invitation email sent.
Alternative Flows	● User Already in Project: System displays a message indicating the user is already a member of the project. ● Invalid Email Format: System displays an error message indicating the email format is invalid. ● Invitation Failure: System displays an error message indicating the invitation could not be sent.
Quality Requirements	● Clear confirmation and error messages. ● Appropriate handling of invalid email addresses. ● Preventing duplicate invitations to the same project.

Case Description-09: Accept/Decline Project Invitation

Use Case	Accept/Decline Project Invitation
Goal	Allow a user to accept or decline a project invitation.

Precondition	User has received a project invitation.
Success End Condition	**(Accept):** User's status is updated to "Accepted," user is added to the project, and confirmation is sent to the user and project admin. **(Decline):** User's status is updated to "Declined," and the project admin is notified.
Failed End Condition	Error message displayed due to an invalid or expired invitation.
Primary Actors: Secondary Actors:	User (Invited User)
Trigger	User receives a project invitation.
Description / Main Success Scenario	**Success Scenario (Accept):** 1. User receives invitation. 2. System displays invitation details. 3. User chooses to accept invitation. 4. System validates invitation acceptance. 5. (If invitation valid) System updates user's status to "Accepted." 6. System adds user to project. 7. System sends confirmation to user and project admin. **Success Scenario (Decline):** 1. User receives invitation. 2. System displays invitation details. 3. User chooses to decline invitation. 4. System validates invitation decline. 5. (If invitation valid) System updates user's status to "Declined." 6. System notifies project admin.
Alternative Flows	● Invalid/Expired Invitation (from 4 in both scenarios): System displays error message: "Invalid or Expired Invitation."
Quality Requirements	● Secure validation of invitations. ● Clear confirmation messages. ● Timely notification to the project admin. ● Preventing acceptance/decline of already processed invitations.

Case Description-10: Create Team

Use Case	Create Team
Goal	Allow an admin to create a new team.
Precondition	User has admin privileges.
Success End Condition	The new team is created, and confirmation is displayed.
Failed End Condition	Error message is displayed due to invalid team details; admin is prompted to correct the details.
Primary Actors: Secondary Actors:	User (Project manager)
Trigger	Click the create new team button.
Description / Main Success Scenario	1. Admin initiates team creation. 2. System displays create team form. 3. Admin enters team details. 4. System validates team details. 5. (If details valid) System creates new team. 6. System displays confirmation of team creation.
Alternative Flows	● Invalid Details (from 4): System displays error message; prompts admin to correct details.
Quality Requirements	● Robust validation of team details. ● Clear and informative error messages. ● Confirmation message to ensure successful creation.

Case Description-11: Assign Members to Team

Use Case	Assign Members to Team
Goal	Allow an admin to assign members to a team.

Precondition	User has admin privileges and the team exists.
Success End Condition	Selected members are assigned to the team, and confirmation is displayed.
Failed End Condition	Error message is displayed due to an invalid selection; admin is prompted to correct the selection.
Primary Actors: Secondary Actors:	User (Project manager)
Trigger	Admin initiates the assign members process.
Description / Main Success Scenario	1. Admin initiates assign members process. 2. System displays assign members form. 3. Admin selects team and members. 4. System validates selection. 5. (If selection valid) System assigns members to the team. 6. System displays confirmation of members assigned.
Alternative Flows	● Invalid Selection (from 4): System displays error message.
Quality Requirements	● Efficient selection of teams and members. ● Robust validation of the selection (e.g., checking if members already belong to the team, if the team exists). ● Clear and informative error messages. ● Confirmation message to ensure successful assignment.

Case Description-12: Remove Members from Team

Use Case	Remove Members from Team

Goal	Allow an admin to remove members from a team.
Precondition	User has admin privileges and the team exists.
Success End Condition	Selected members are removed from the team, and confirmation is displayed.
Failed End Condition	Error message is displayed if the selected members are not found in the team.
Primary Actors: Secondary Actors:	User (Project manager)
Trigger	Admin initiates the remove members process.
Description / Main Success Scenario	7. Admin initiates assign members process. 8. System displays assign members form. 9. Admin selects team and members. 10. System validates selection. 11. (If selection valid) System assigns members to the team. 12. System displays confirmation of members assigned.
Alternative Flows	● Invalid Selection (from 4): System displays error message.
Quality Requirements	● Efficient selection of teams and members. ● Robust validation of the selection (e.g., checking if members already belong to the team, if the team exists). ● Clear and informative error messages. ● Confirmation message to ensure successful assignment.

Case Description-13: Create and Assign Task

Use Case	Create and Assign Task
Goal	Allow a user to create a new task and optionally assign it to participants or team members.
Precondition	User is logged in and has the necessary permissions to create tasks within the project.
Success End Condition	Task is created, assigned to selected participants/team members, and notifications are sent.
Failed End Condition	Task is created, and confirmation is sent to the user.
Primary Actors: Secondary Actors:	User (Project manager)
Trigger	User requests task creation.
Description / Main Success Scenario	1. User requests task creation. 2. System displays task creation form. 3. User enters task details. 4. System validates task details. 5. (If task details valid) System creates new task in project. 6. System displays task assignment options. 7. User chooses to assign task. 8. User selects participants/team members. 9. System validates assignment. 10. (If assignments valid) System assigns task to selected participants/team members. 11. System notifies assigned users. 12. System sends confirmation to the user.
Alternative Flows	● Invalid Task Details (from 4): System displays error message: "Invalid Task Details." ● Invalid Assignment (from 9): System displays error message: "Invalid Assignment."
Quality Requirements	● Robust validation of task details and assignments. ● Clear and informative error messages. ● Timely notifications to assigned users. ● Confirmation message to the task creator.

Case Description-14: Submit Task

Use Case	Submit Task
Goal	Allow a user to submit a completed task, optionally providing proof.
Precondition	User is logged in and the task exists.
Success End Condition	Task is submitted, proof is stored, and confirmation is displayed.
Failed End Condition	Task is marked as complete, and confirmation is displayed.
Primary Actors: Secondary Actors:	User (Project participant)
Trigger	User selects a task for submission.
Description / Main Success Scenario	1. User selects task for submission. 2. User marks task as complete. 3. User optionally uploads file/provides text (proof). 4. (If proof provided) System validates submission. 5. (If submission valid) System submits task. 6. System stores files/text proof. 7. System marks task as completed. 8. System confirms task submission.
Alternative Flows	● Invalid Submission (from 4): System displays error message.
Quality Requirements	● Robust validation of submissions (if applicable). ● Secure storage of submitted files/text. ● Clear confirmation and error messages. ● Handling of different file types (if applicable).

2.4.3 Activity Diagram

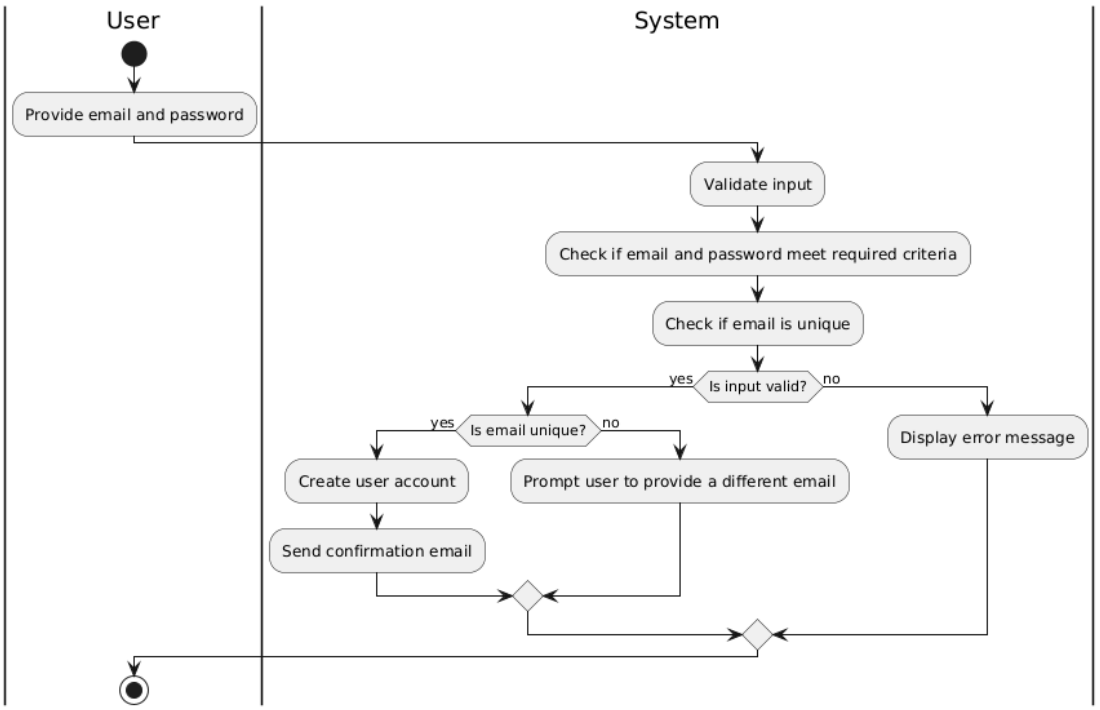


Figure 3.1: Registration activity diagram

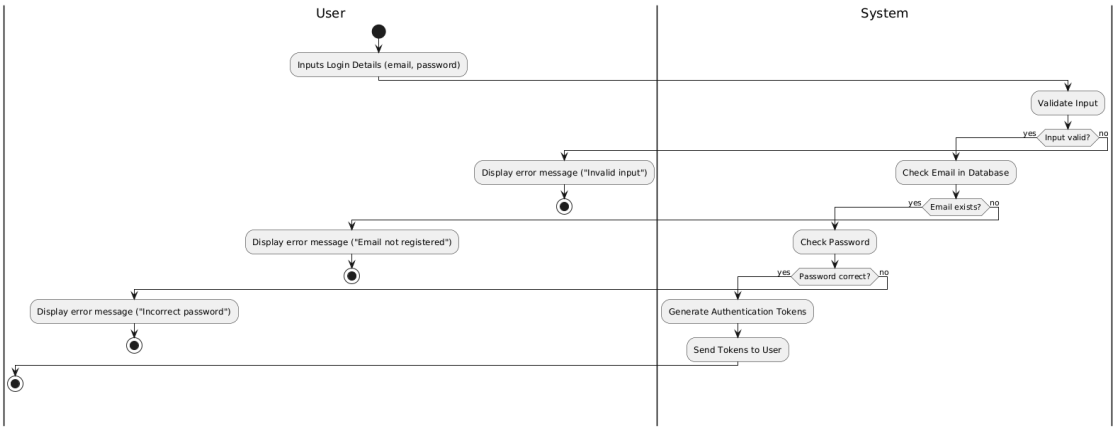


Figure 3.2: Login activity diagram

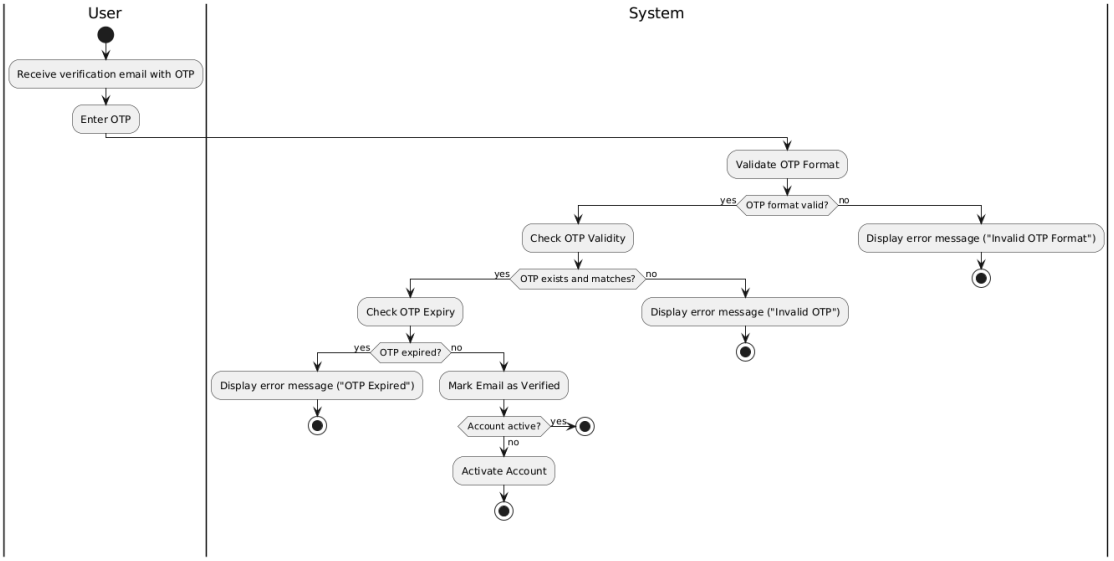


Figure 3.3: Verify email activity diagram

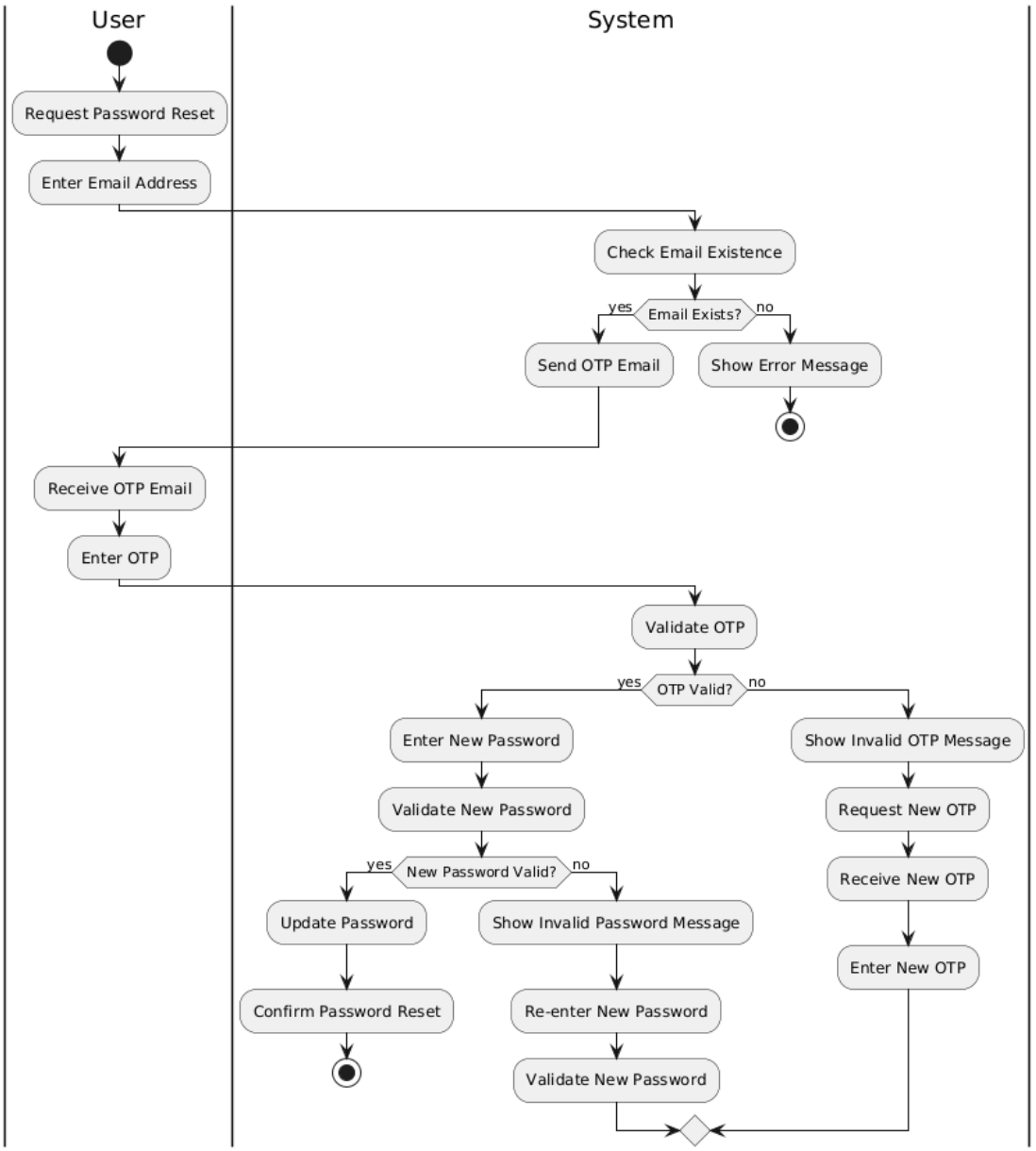


Figure 3.4: Reset Password Activity Diagram

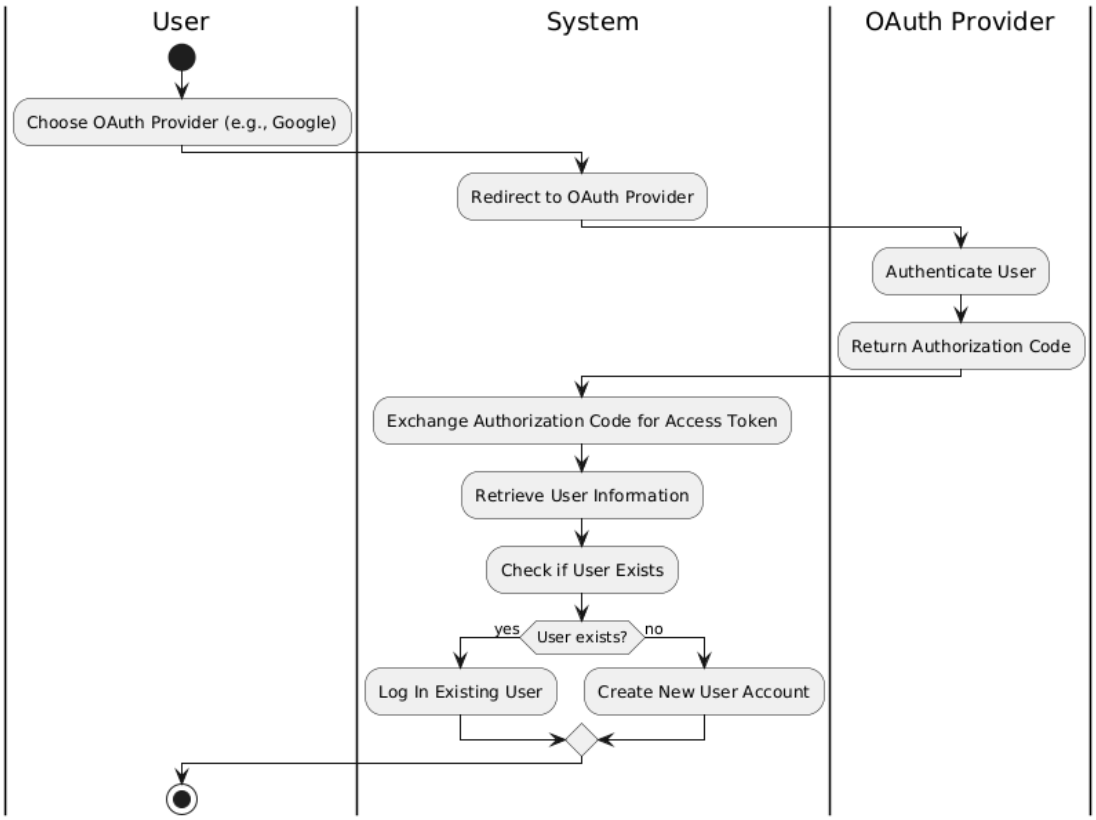


Figure 3.5: OAuth Login Activity Diagram

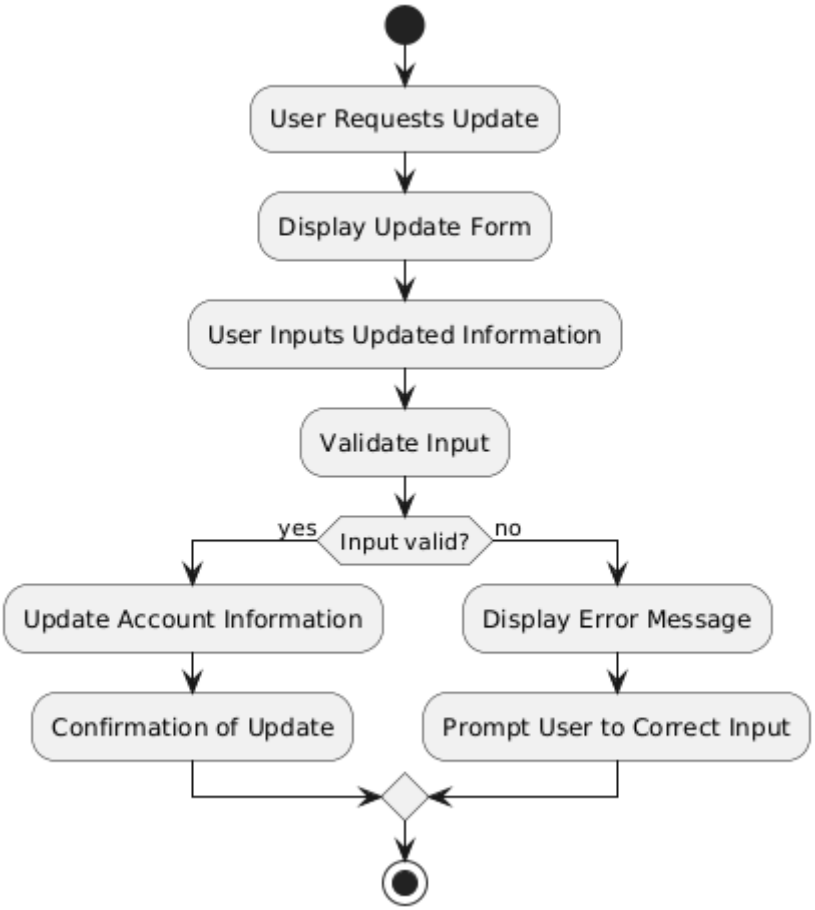


Figure 3.6: Update Profile Activity Diagram

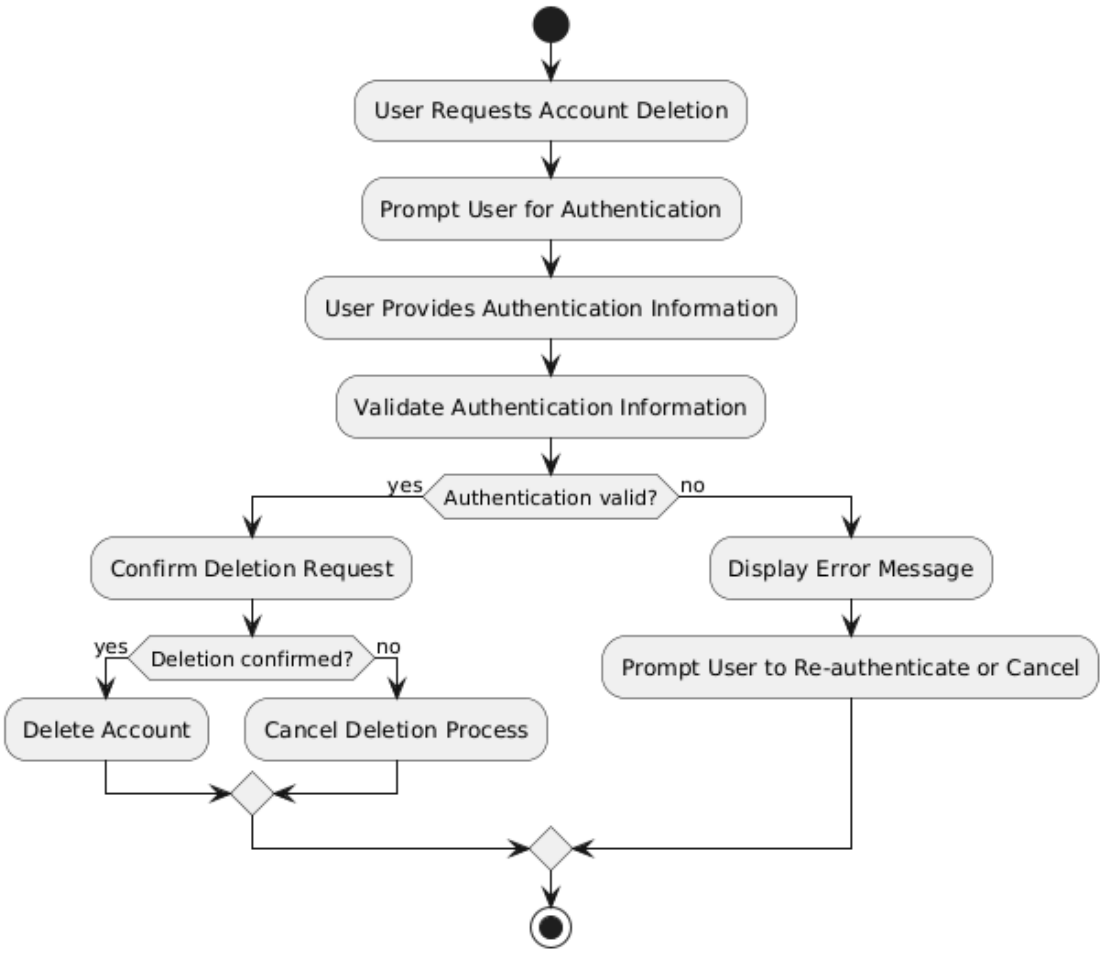


Figure 3.7: Delete User Account Activity Diagram

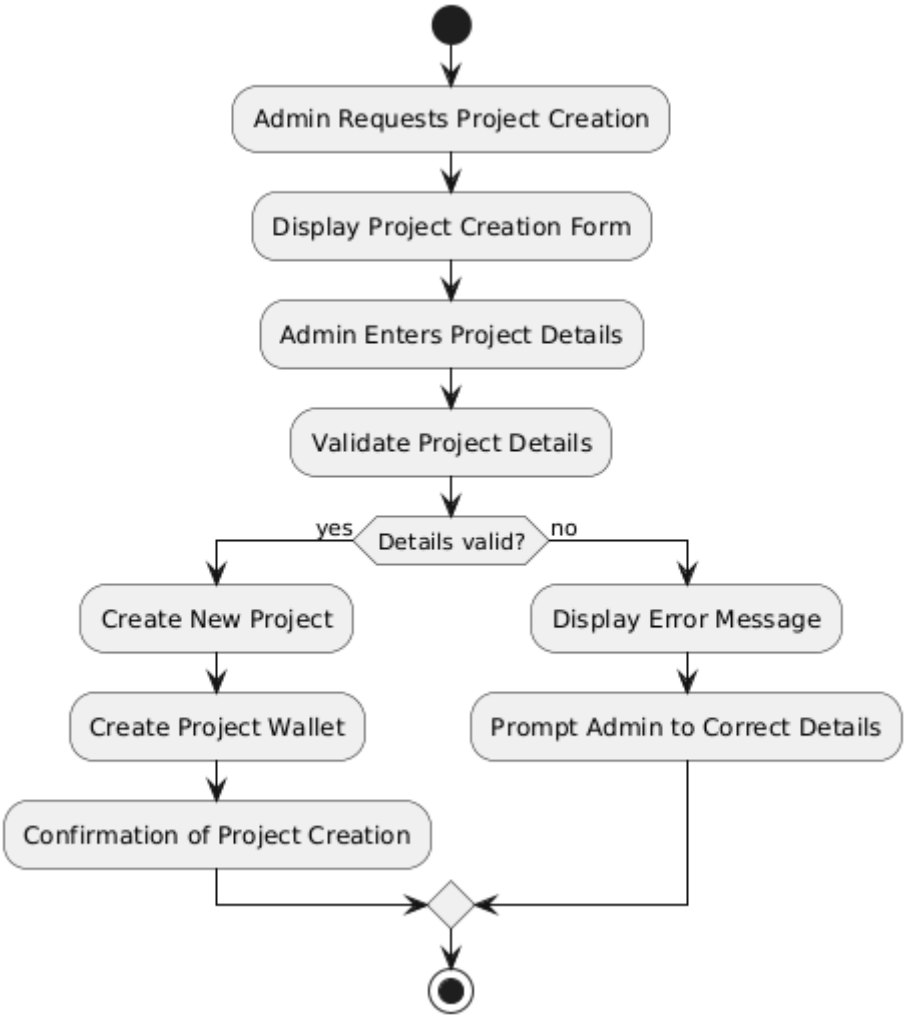


Figure 3.8: Invite Participant Activity Diagram

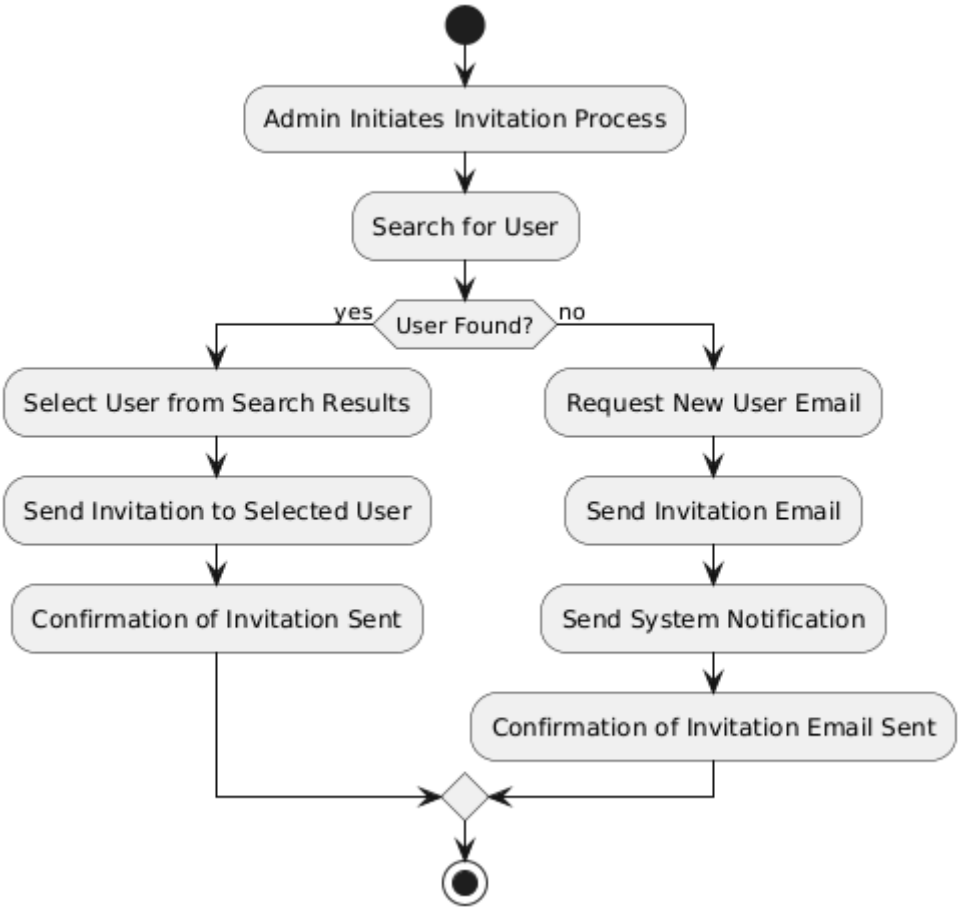


Figure 3.9: Respond Invitation Activity Diagram

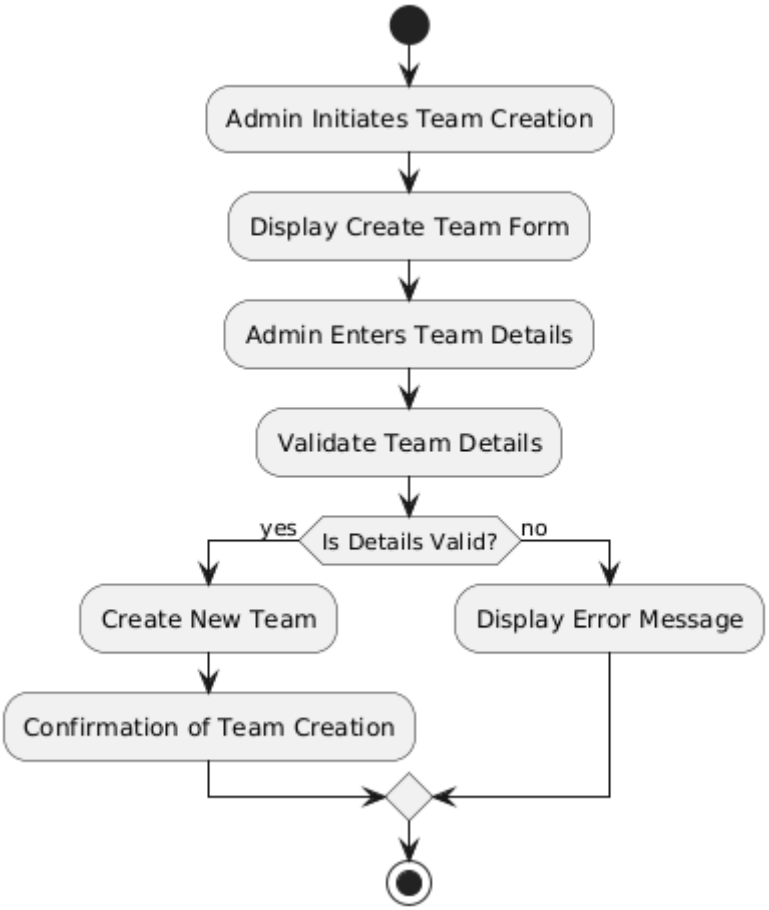


Figure 3.10: Create Team Activity Diagram

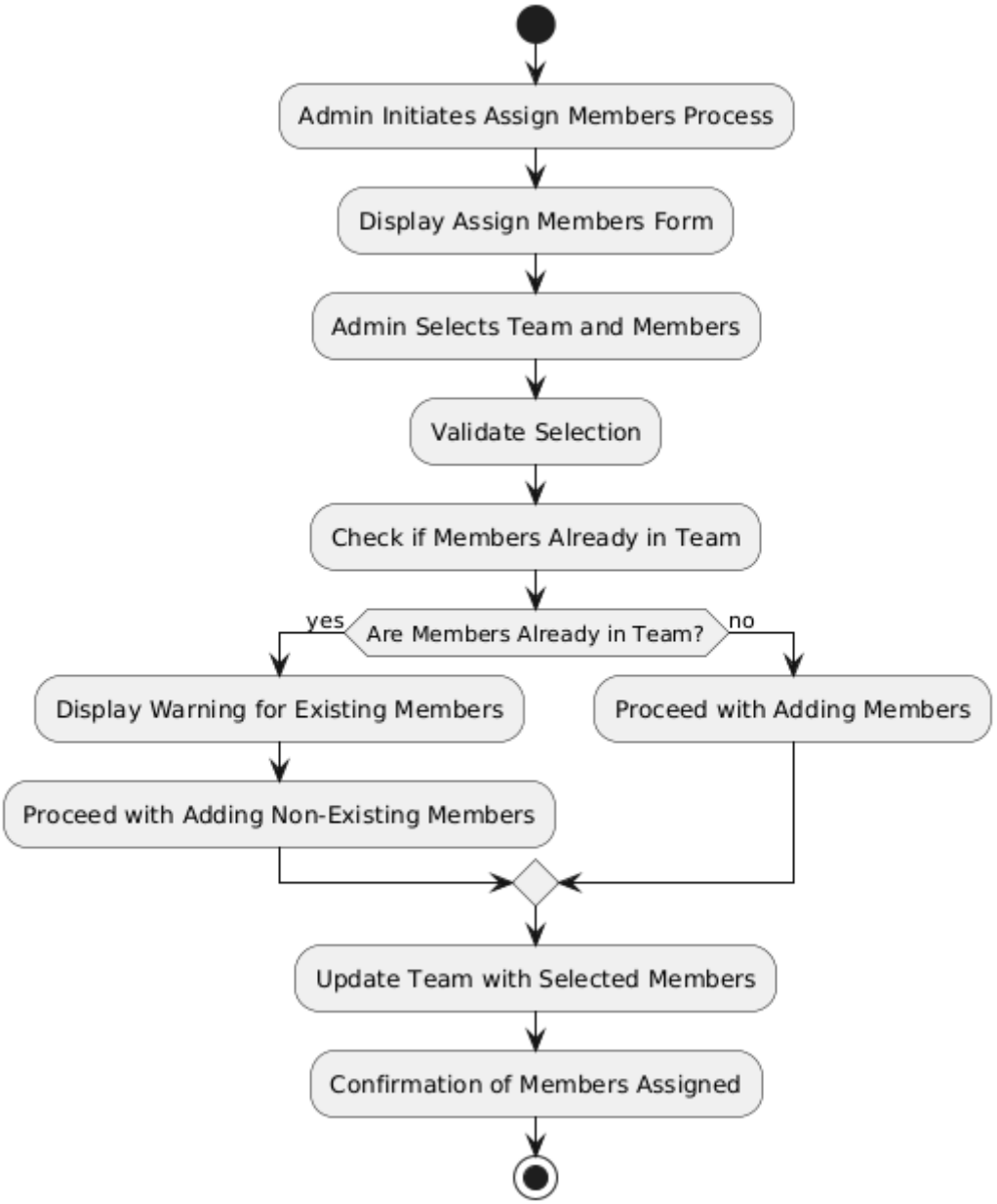


Figure 3.11: Add Team Member Activity Diagram

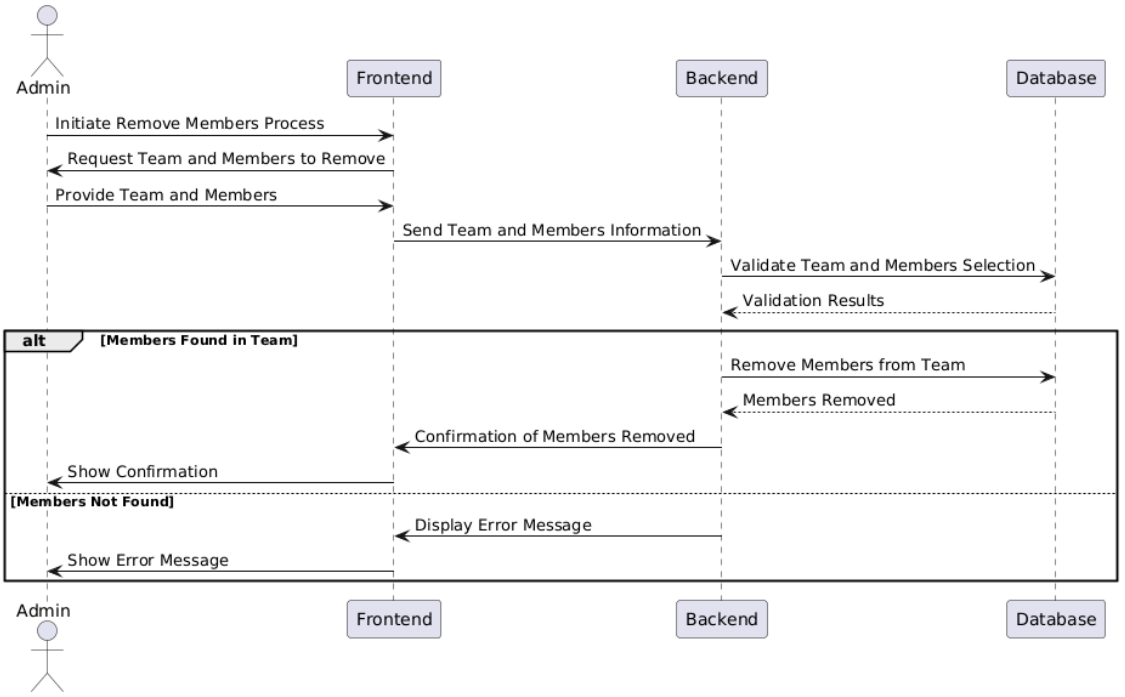


Figure 3.12: Remove Team Member Activity Diagram

Figure 3.13: Create Task Activity Diagram

Figure 3.14: Submit Task Activity Diagram

2.4.4 Sequence Diagram

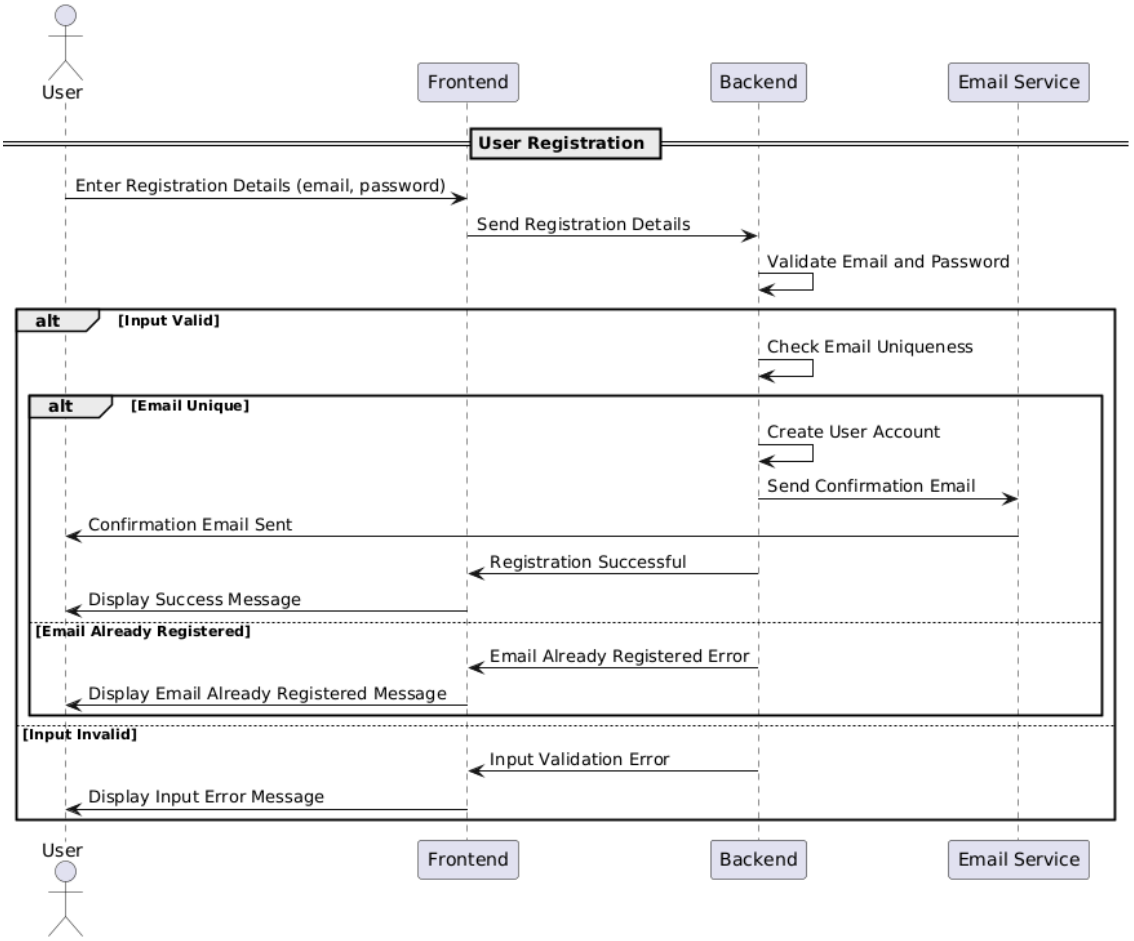


Figure 4.1: User Registration

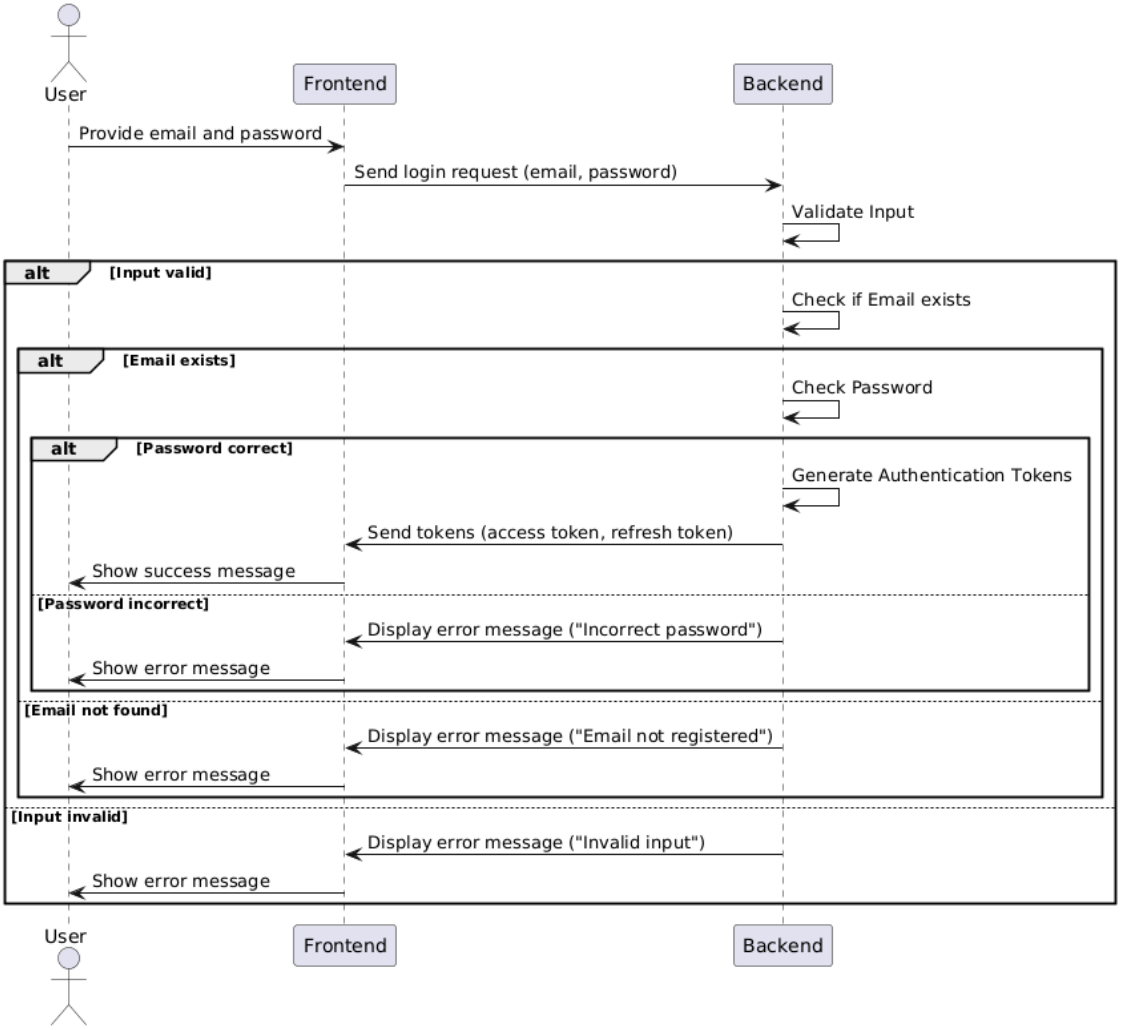


Figure 4.2:User Login

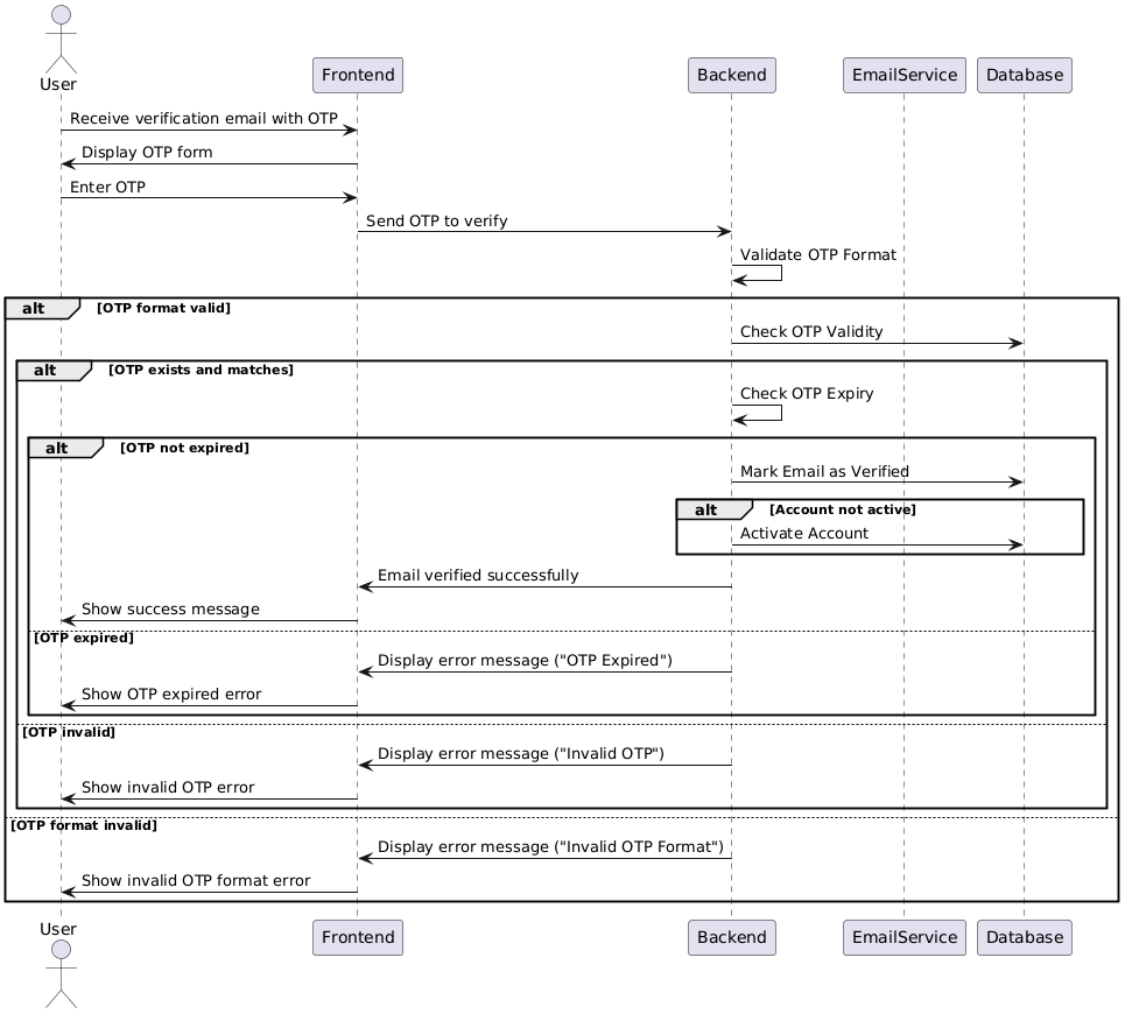


Figure 4.3: Verify Email

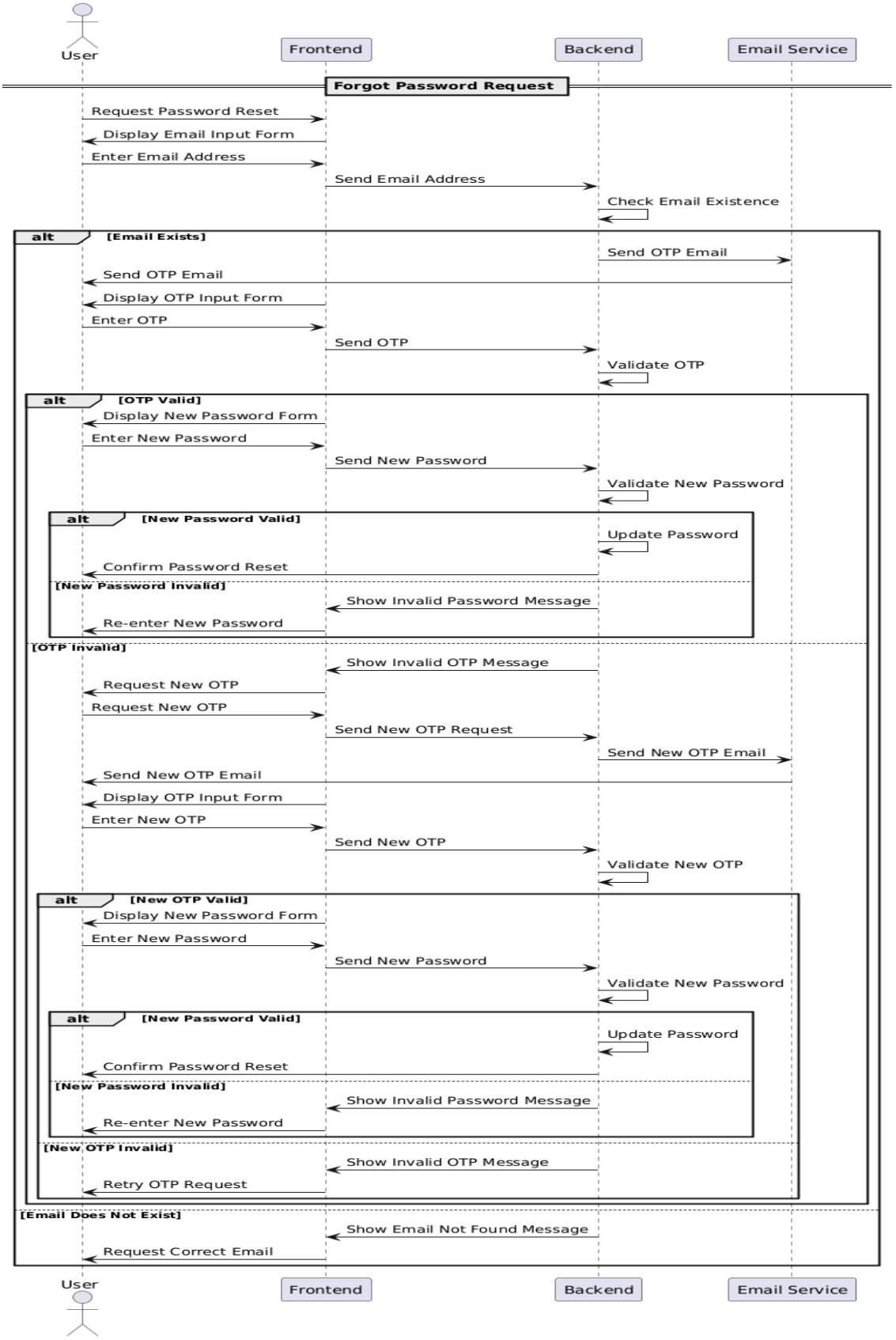


Figure 4.4: Reset Password

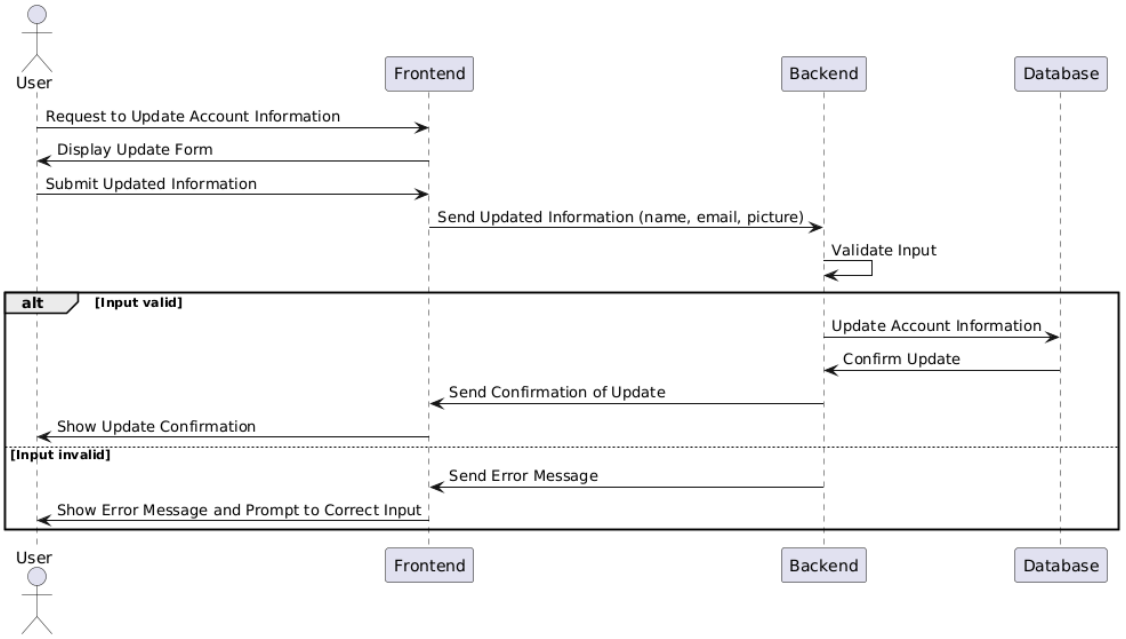


Figure 4.5: Update Profile

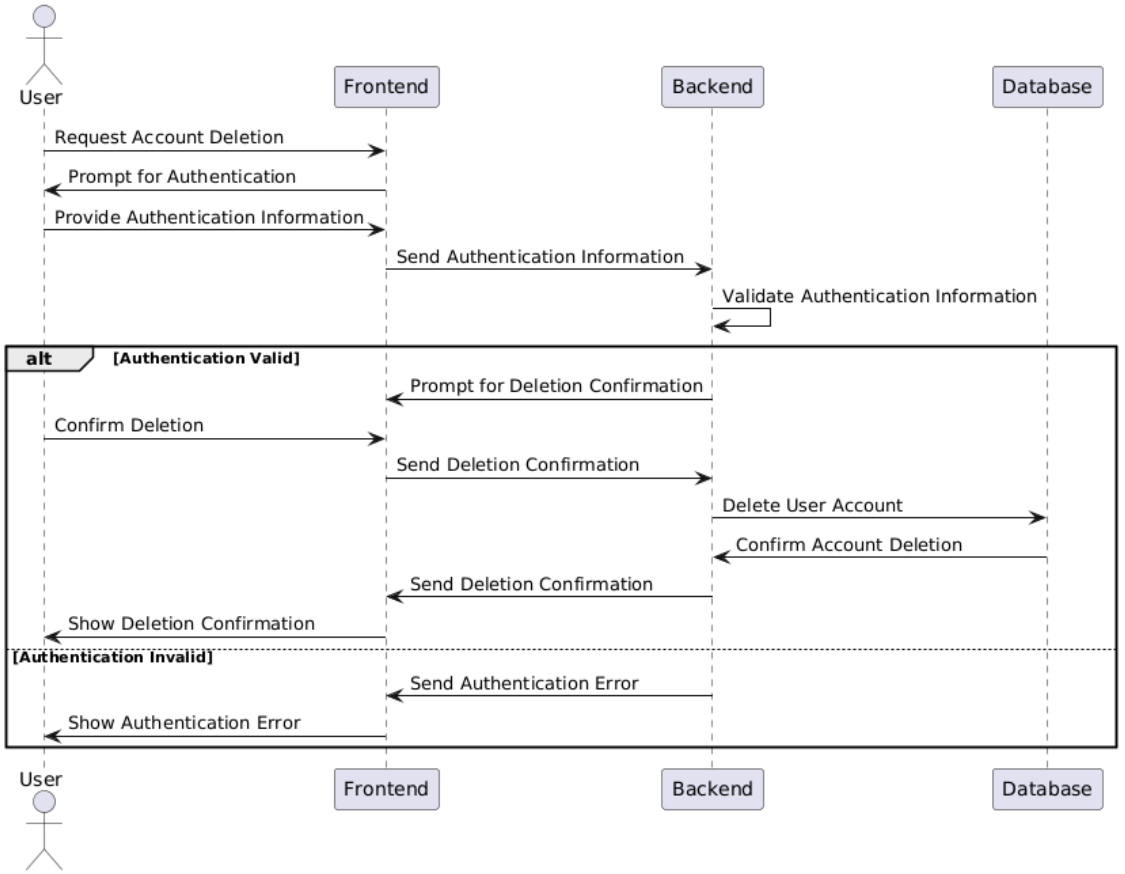


Figure 4.6: Delete Account

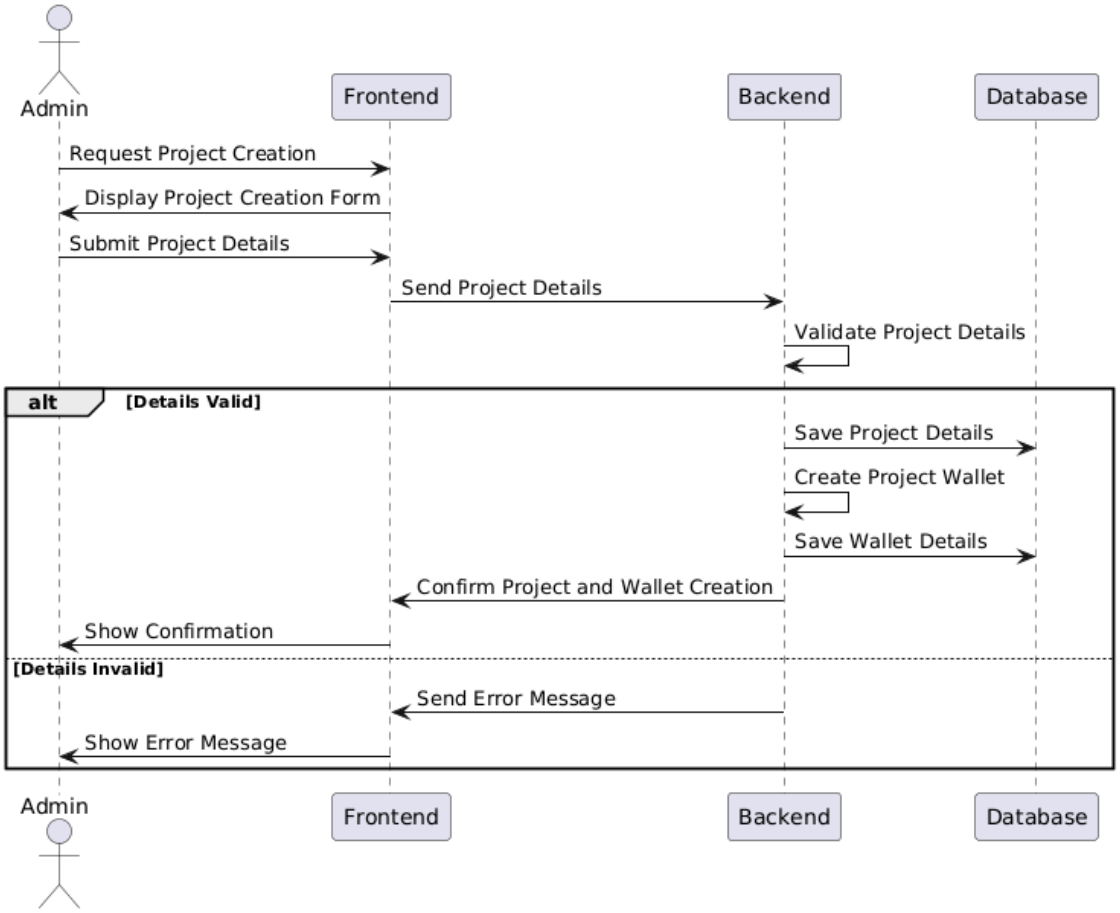


Figure 4.7: Create Project

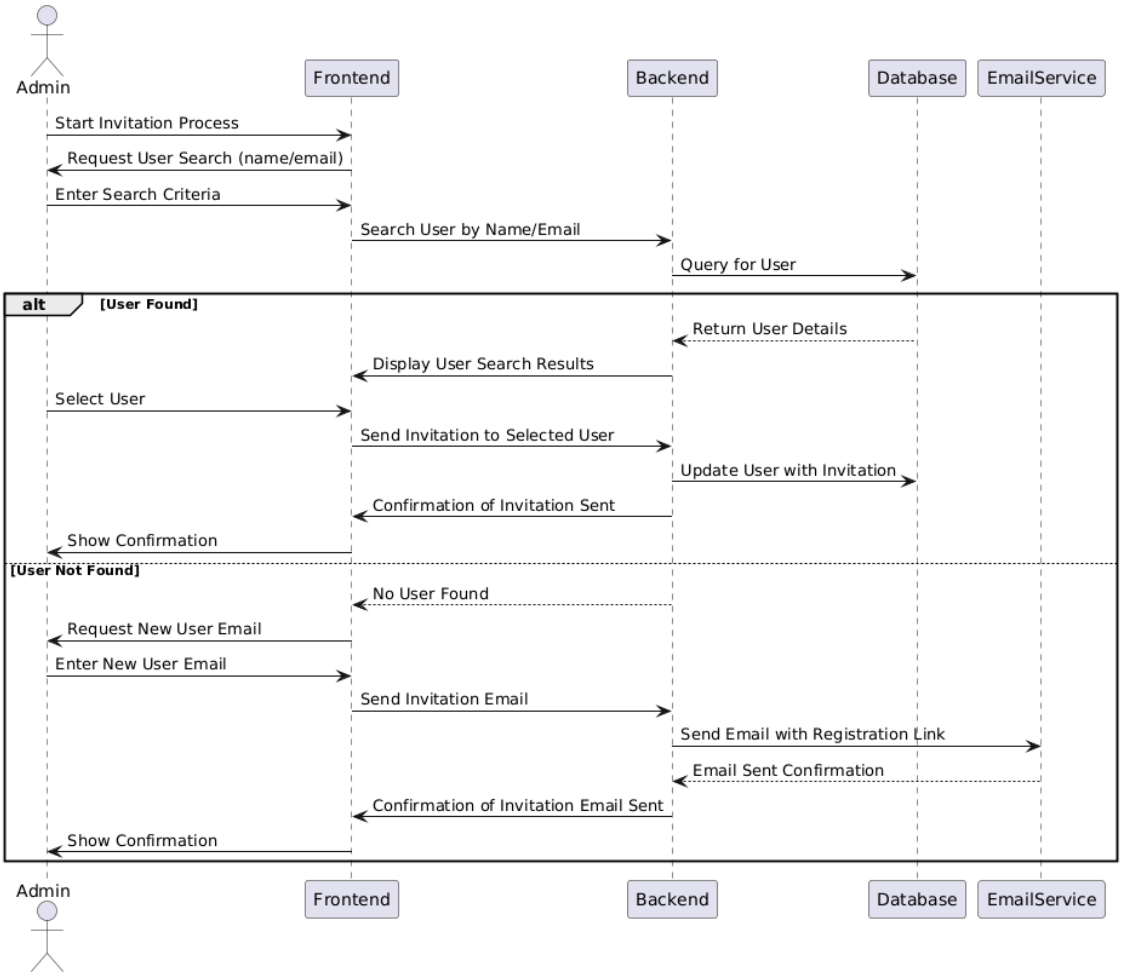


Figure 4.8: Invite Participant

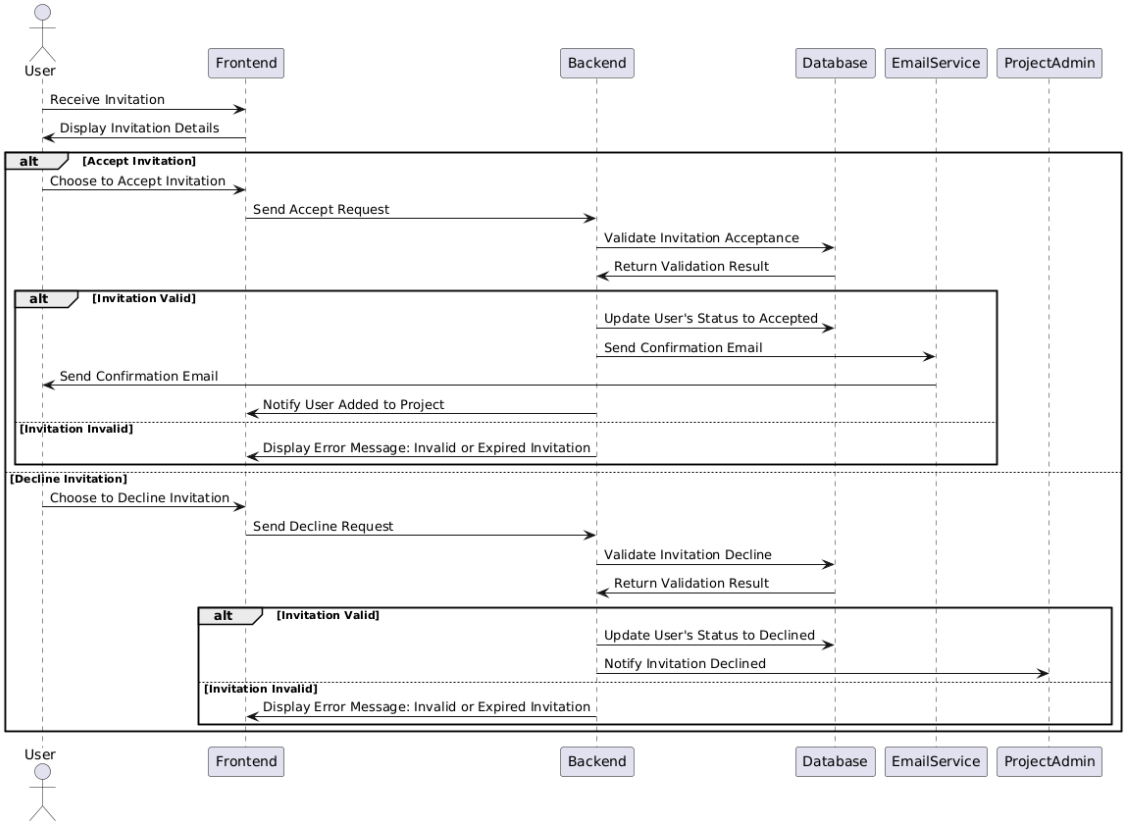


Figure 4.9: Respond Invitation

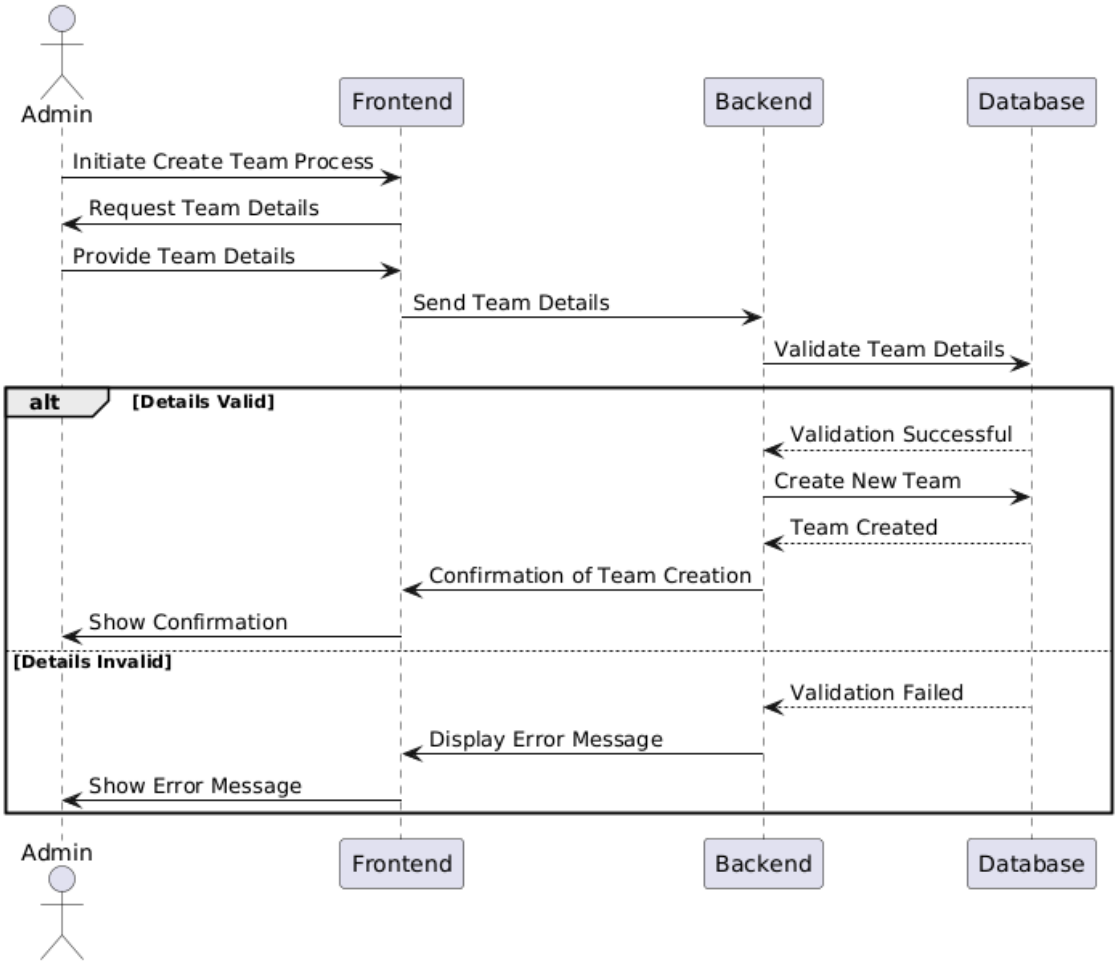


Figure 4.10: Create Team

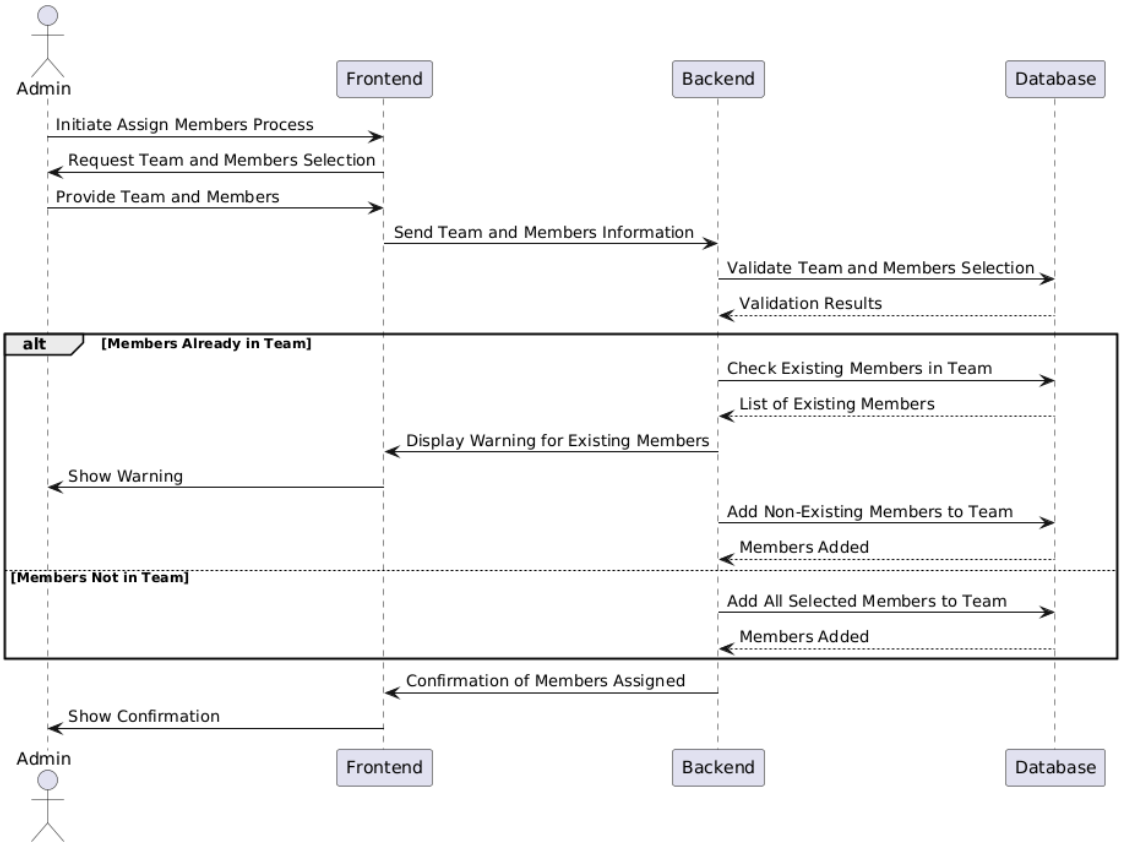


Figure 4.11: Add Team Member

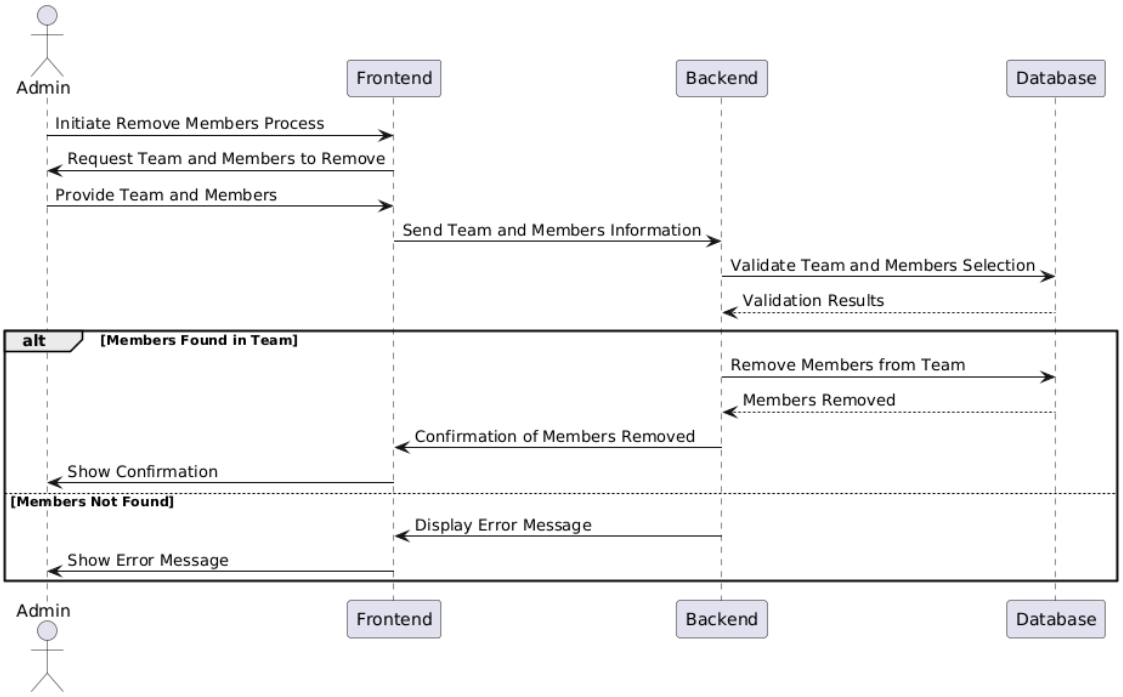


Figure 4.12: Remove Member From Team

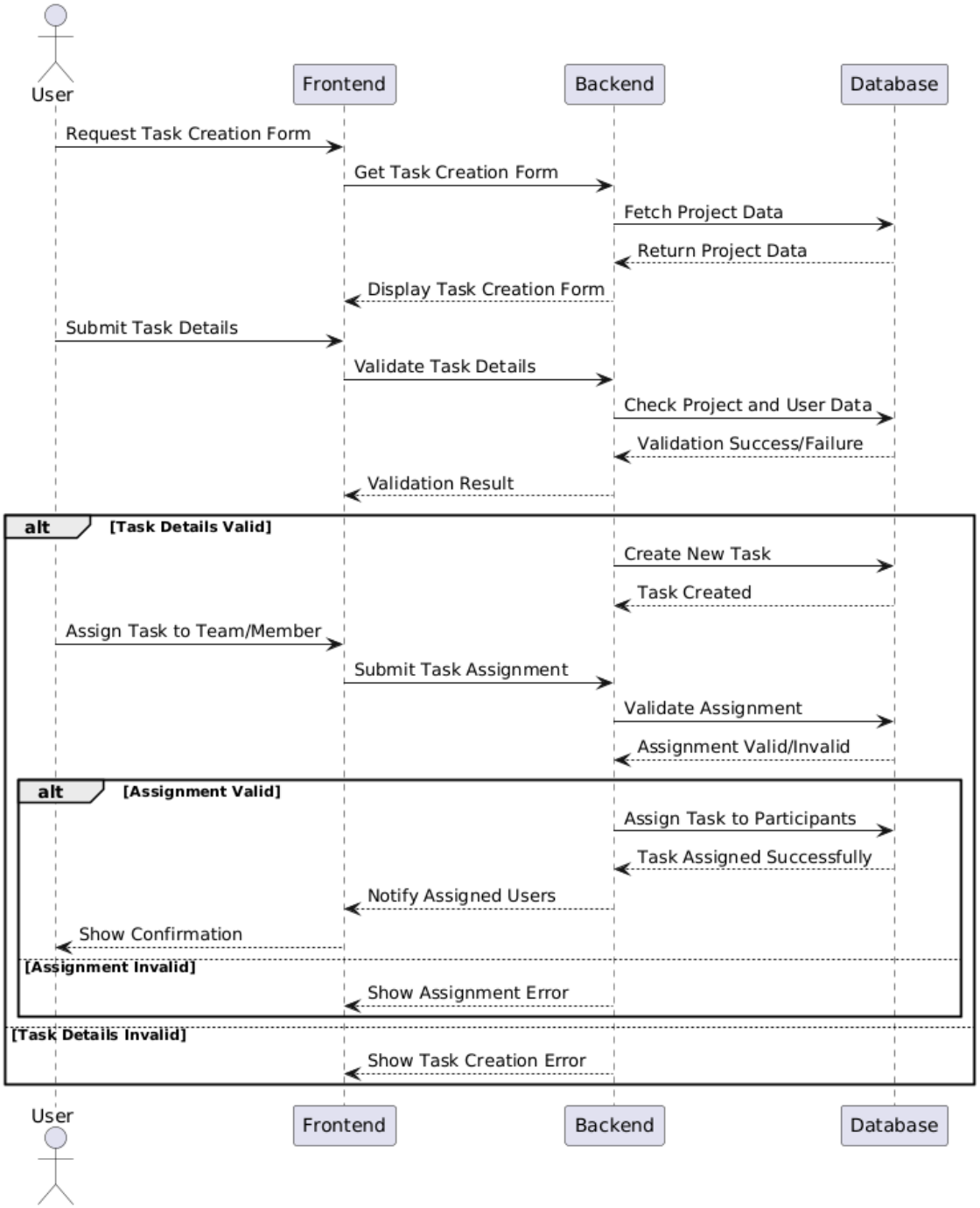


Figure 4.13: Task Assign

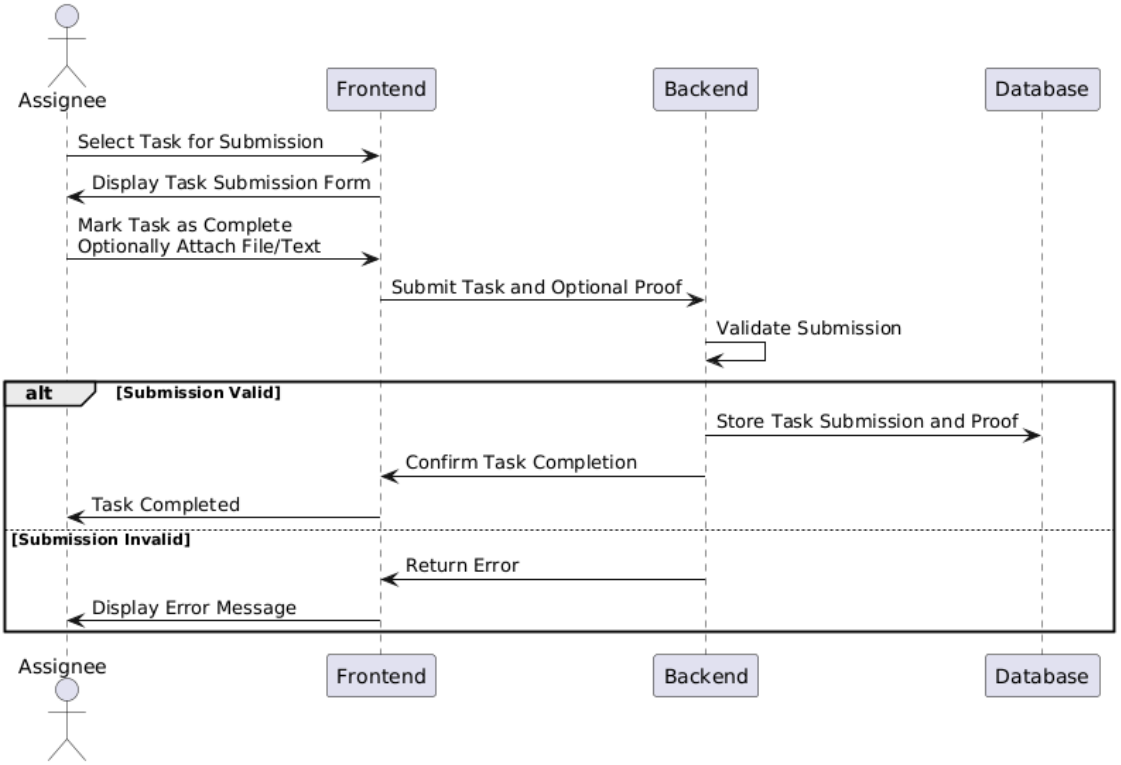


Figure 4.14: Submit Task

2.4.5 Class Diagram

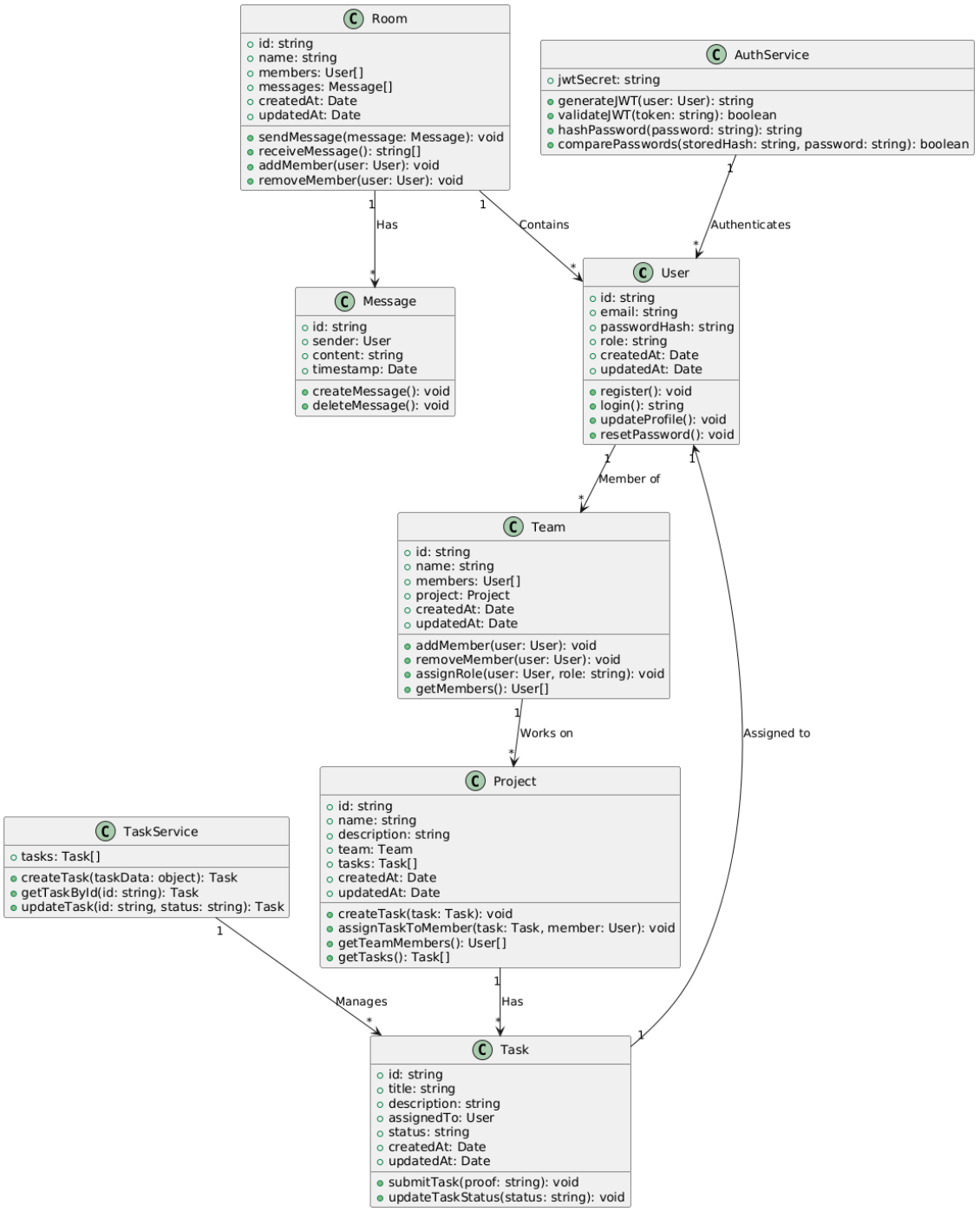


Figure 4: Class Diagram

2.4.6 ER Diagram

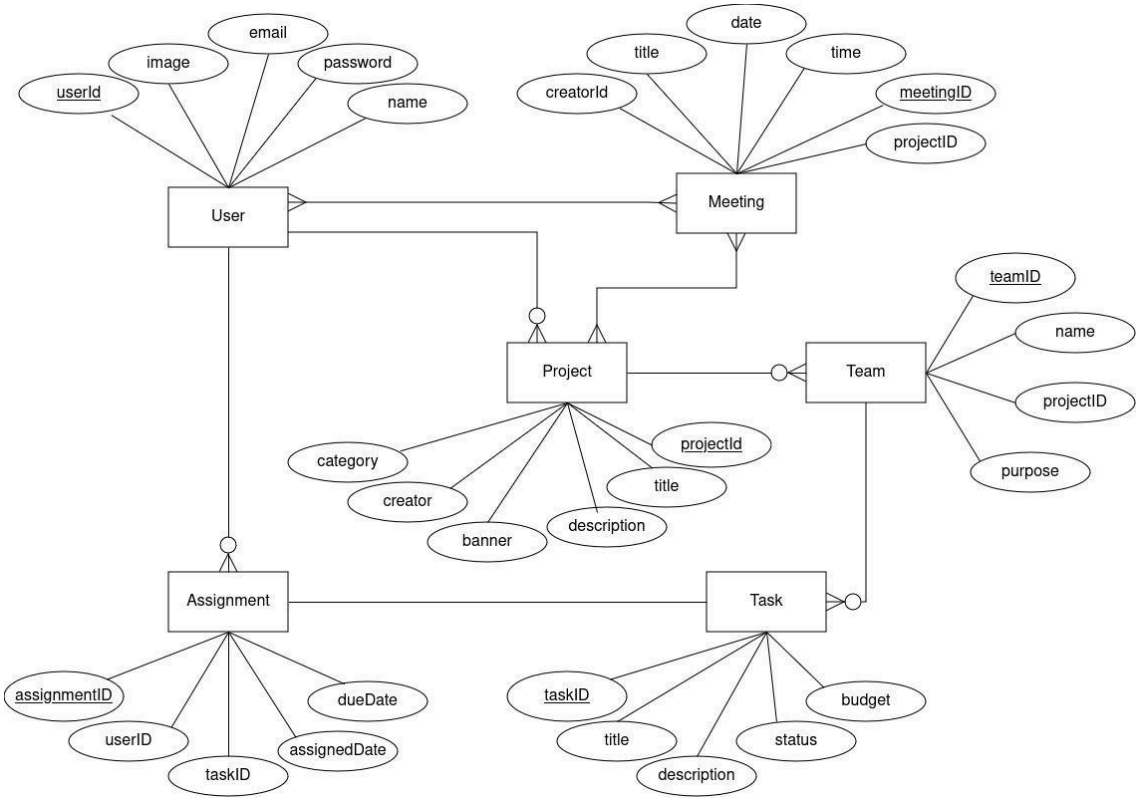


Figure 5: ER Diagram

2.5 Coding: Appendix A

```
model User {
 id                String   @id @default(uuid())
 email             String   @unique
 password          String?
 name              String
 createdAt         DateTime @default(now())
 updatedAt         DateTime @updatedAt
 profilePictureId    String?
}
```

2.6 Summary

This Entity-Relationship Diagram (ERD) models a system for managing projects, teams, tasks, and meetings, which also serve as chat rooms. The key entities and their relationships are:

- **User:** Stores user information (user ID, name, email, password, image).
- **Project:** Stores project details (project ID, title, category, creator, banner, description). A user can create multiple projects.
- **Team:** Stores team information (team ID, name, project ID, purpose). A project can have multiple teams.
- **Task:** Stores task details (task ID, title, description, budget, status). A project can have multiple tasks.
- **Meeting:** Stores meeting information *and serves as a chat room* (meeting ID, title, creator, project ID, date, time). A project can have multiple meetings/chat rooms.

The relationships between these entities are:

- A User can create multiple Projects (one-to-many).
- A Project can have multiple Teams (one-to-many).
- A Project can have multiple Tasks (one-to-many).
- A Project can have multiple Meetings/Chat Rooms (one-to-many).
- A User can be assigned to multiple Tasks through the Assignment entity (many-to-many).

Chapter 3 Software Testing

3.1 Introduction

This chapter details the testing strategy employed to ensure the quality, reliability, and functionality of the Restaurant Recommendation system. The system aims to provide users with personalized restaurant recommendations based on their preferences, location, and other criteria. It also includes an administrative interface for managing restaurant data and user accounts. Thorough testing is crucial to validate the system's core features, identify potential defects, and ultimately deliver a robust and user-friendly experience. This chapter will cover the specific features to be tested, the testing approaches used, and the criteria for determining test success or failure.

3.2 Testing Features

3.2.1 Feature to Be Tested

User Management:

- User Registration
- User Login and OAuth Login
- Password Reset
- Account Information Update
- User Logout

Project and Team Management:

- Project Creation and Selection
- Team Creation
- Assign Members to Team
- Remove Members from Team
- Invite Users to Project

Task Management:

- Create and Assign Tasks
- Submit Tasks
- View Assigned Tasks

3.3 Testing Strategies

3.3.1 Test Approach

Integration Testing:

- Check if different parts of the system, such as user management and task management, work together properly.

System Testing:

- Test the whole system to make sure everything works as expected when combined, like creating projects and assigning tasks.

3.3.2 Pass/Fail Criteria

Pass Criteria:

- The feature works as expected without errors.
- The system processes inputs and shows correct outputs.
- Validation and error messages appear when needed.

Fail Criteria:

- The feature produces errors or crashes.
- The system doesn't process inputs correctly.
- Missing or incorrect error messages.

3.4 System Testing (Test Cases with Report)

Test: Registration

Test ID	Test Name	Steps	Expected Outcome	Result
TC-01	Successful Registration	1. Fill valid email: testuser@example.com, password: P@ssw0rd123. 2. Submit.	Account created, confirmation email sent, success message: *"Registration successful!"*	Pass
TC-02	Invalid Email Format	1. Fill invalid email: invalidemail.com, valid password. 2. Submit.	Error message: *"Invalid email format."* User stays on form.	Pass
TC-03	Weak Password	1. Fill valid email: user@example.com, weak password: 123. 2. Submit.	Error message: *"Password does not meet criteria."* User stays on form.	Pass
TC-04	Duplicate Email	1. Fill already registered email: testuser@example.com, valid password. 2. Submit.	Error message: *"Email is already in use."* User stays on form.	Pass

Test: Login

Test ID	Test Name	Steps	Expected Outcome	Result
TC-01	Successful Login	1. Enter valid email/password. 2. Click Login.	User authenticated, tokens generated, access granted, success message: *"Login successful."*	Pass

TC -02	Invalid Input	1. Enter invalid email/empty password. 2. Click Login.	Error message: *"Invalid input."* User stays on login page.	Pass
TC -03	Unregistered Email	1. Enter unregistered email, valid password. 2. Click Login.	Error message: *"Email not registered."* User stays on login page.	Pass
TC -04	Incorrect Password	1. Enter valid email, incorrect password. 2. Click Login.	Error message: *"Incorrect password."* User stays on login page.	Pass

Test: Email Verification:

Test ID	Test Name	Steps	Expected Outcome	Result
TC -01	Successful Verification	1. Enter valid OTP received in email. 2. Submit.	Email marked as verified, account activated (if inactive), success message: *"Email verified."*	Pass
TC -02	Invalid OTP Format	1. Enter OTP in an invalid format (e.g., abc123). 2. Submit.	Error message: *"Invalid OTP format."* User stays on verification page.	Pass
TC -03	Invalid OTP	1. Enter incorrect OTP. 2. Submit.	Error message: *"Invalid OTP."* User stays on verification page.	Pass
TC -04	Expired OTP	1. Enter valid OTP after expiry time (e.g., 5 minutes). 2. Submit.	Error message: *"OTP expired."* User stays on verification page.	Pass

Test: Password Reset

Test ID	Test Name	Steps	Expected Outcome	Result
TC-01	Successful Password Reset	1. Request password reset. 2. Enter email address. 3. Receive OTP. 4. Enter OTP. 5. Enter valid new password.	Password updated successfully, success message: *"Password reset successful."*	Pass
TC-02	Email Not Found	1. Request password reset. 2. Enter unregistered email address.	Error message: *"Email not found."* User stays on reset page.	Pass
TC-03	Invalid OTP	1. Request password reset. 2. Enter email. 3. Enter incorrect OTP.	Error message: *"Invalid OTP."* User stays on reset page, option to request new OTP.	Pass
TC-04	Invalid New Password	1. Request password reset. 2. Enter email. 3. Enter OTP. 4. Enter invalid new password.	Error message: *"Invalid password."* User re-enters new password, password validation checked.	Pass

OAuth Login

Test ID	Test Name	Steps	Expected Outcome	Result
TC-01	Successful OAuth Login	1. Choose OAuth provider (e.g., Google). 2. System redirects to provider. 3. OAuth provider authenticates. 4. System exchanges authorization code for access token. 5. User information retrieved.	User logged in successfully, or new user account created, success message: *"Login successful."*	Pass
TC-02	Network Error	1. Choose OAuth provider. 2. System redirects to provider. 3. Network error occurs.	Error page displayed with message: *"Network error. Please try again later."*	Pass
TC-03	New User OAuth Login	1. Choose OAuth provider. 2. OAuth provider authenticates. 3. User does not exist in the system.	New account created and user logged in, success message: *"Account created and logged in successfully."*	Pass
TC-04	Existing User OAuth Login	1. Choose OAuth provider. 2. OAuth provider authenticates. 3. User exists in system.	Existing user logged in successfully, success message: *"Login successful."*	Pass

Test: Update Account

Test ID	Test Name	Steps	Expected Outcome	Result
TC-01	Successful Account Update	1. User requests account update. 2. System displays update form. 3. User inputs valid updated information. 4. Submit.	Account updated successfully, confirmation message: *"Account information updated."*	Pass
TC-02	Invalid Input	1. User requests account update. 2. System displays update form. 3. User inputs invalid information (e.g., invalid email). 4. Submit.	Error message: *"Invalid input. Please correct the information."* User stays on update page.	Pass
TC-03	Secure Handling of Updated Info	1. User requests account update. 2. User inputs updated information. 3. Submit.	Updated information securely processed, no unauthorized access, and confirmation displayed.	Pass

Test: Account Deletion

Test ID	Test Name	Steps	Expected Outcome	Result
TC-01	Successful Account Deletion	1. User requests account deletion. 2. System prompts for authentication. 3. User provides valid authentication. 4. User confirms deletion.	Account deleted successfully, confirmation message: *"Your account has been deleted."*	Pass
TC-02	Invalid Authentication	1. User requests account deletion. 2. System prompts for authentication. 3. User provides invalid authentication information.	Error message: *"Invalid authentication. Please re-enter your credentials."* User stays on deletion page.	Pass
TC-03	Deletion Not Confirmed	1. User requests account deletion. 2. System prompts for authentication. 3. User confirms authentication but cancels deletion.	Deletion process canceled, user stays on the account page with no changes.	Pass

Test: Create Project

Test ID	Test Name	Steps	Expected Outcome	Result
TC-01	Successful Project Creation	1. Admin requests project creation. 2. System displays project creation form. 3. Admin enters valid project details. 4. Submit.	Project created successfully, confirmation message: *"Project created successfully."*	Pass
TC-02	Invalid Project Details	1. Admin requests project creation. 2. System displays project creation form. 3. Admin enters invalid project details (e.g., missing required field). 4. Submit.	Error message: *"Invalid project details. Please correct the information."* User stays on form.	Pass
TC-03	Project Wallet Creation	1. Admin requests project creation. 2. Admin enters valid project details. 3. Submit.	System creates project wallet along with project, ensuring it's linked to the project.	Pass

Test: Invite User

Test ID	Test Name	Steps	Expected Outcome	Result
TC-01	Successful User Invitation	1. Project manager initiates invitation. 2. Enters valid email: user@example.com. 3. Submit.	Invitation email sent to user@example.com, confirmation message: *"Invitation sent successfully."*	Pass
TC-02	User Already in Project	1. Project manager initiates invitation. 2. Enters email: user@example.com (already a member). 3. Submit.	Error message: *"User is already a member of the project."* User stays on the invitation page.	Pass
TC-03	Invalid Email Format	1. Project manager initiates invitation. 2. Enters invalid email: invalidemail.com. 3. Submit.	Error message: *"Invalid email format. Please enter a valid email."* User stays on the invitation page.	Pass
TC-04	Invitation Failure	1. Project manager initiates invitation. 2. Enters valid email: user@example.com. 3. Submit.	Error message: *"Invitation could not be sent. Please try again later."* User stays on the invitation page.	Pass

Test: Accept/Decline Project Invitation

Test ID	Test Name	Steps	Expected Outcome	Result
TC-01	Accept Project Invitation	1. User receives invitation. 2. User chooses to accept. 3. Submit.	User's status updated to *"Accepted"*, added to the project, confirmation sent to user and project admin.	Pass
TC-02	Decline Project Invitation	1. User receives invitation. 2. User chooses to decline. 3. Submit.	User's status updated to *"Declined"*, project admin is notified.	Pass
TC-03	Invalid Invitation	1. User receives expired/inactive invitation. 2. User attempts to accept.	Error message: *"Invalid or Expired Invitation."* User stays on invitation page.	Pass
TC-04	Decline After Acceptance	1. User accepts invitation. 2. User attempts to decline after acceptance.	Error message: *"Invitation already processed."* User is unable to decline after accepting.	Pass

Test: Create Team

Test ID	Test Name	Steps	Expected Outcome	Result
TC-01	Successful Team Creation	1. Admin clicks "Create New Team" button. 2. Admin enters valid team details. 3. Submit.	New team is created, and confirmation message is displayed.	Pass
TC-02	Invalid Team Name	1. Admin clicks "Create New Team" button. 2. Admin enters invalid team name. 3. Submit.	Error message: *"Invalid team name."* Admin is prompted to correct the details.	Pass
TC-03	Missing Team Information	1. Admin clicks "Create New Team" button. 2. Admin leaves required fields empty. 3. Submit.	Error message: *"Please fill in all required fields."* Admin is prompted to correct the details.	Pass
TC-04	Invalid Team Details	1. Admin clicks "Create New Team" button. 2. Admin enters invalid team details (e.g., invalid characters). 3. Submit.	Error message: *"Invalid team details."* Admin is prompted to correct the details.	Pass

Test: Assign Members to Team

Test ID	Test Name	Steps	Expected Outcome	Result
TC-01	Successful Member Assignment	1. Admin initiates member assignment process. 2. Admin selects valid team and members. 3. Submit.	Selected members are successfully assigned to the team, and a confirmation message is displayed.	Pass
TC-02	Invalid Team Selection	1. Admin initiates member assignment process. 2. Admin selects a non-existent team. 3. Submit.	Error message: *"Invalid team selection."* Admin is prompted to correct the team selection.	Pass
TC-03	Member Already Assigned	1. Admin initiates member assignment process. 2. Admin selects members already assigned to the team. 3. Submit.	Error message: *"Selected members are already assigned to the team."* Admin is prompted to correct the selection.	Pass
TC-04	Invalid Member Selection	1. Admin initiates member assignment process. 2. Admin selects invalid members (e.g., non-existing members). 3. Submit.	Error message: *"Invalid member selection."* Admin is prompted to correct the member selection.	Pass

Test: Remove Members from Team

Test ID	Test Name	Steps	Expected Outcome	Result
TC-01	Successful Member Removal	1. Admin initiates remove member process. 2. Admin selects valid team and members to remove. 3. Submit.	Selected members are successfully removed from the team, and a confirmation message is displayed.	Pass
TC-02	Member Not Found in Team	1. Admin initiates remove member process. 2. Admin selects members who are not part of the team. 3. Submit.	Error message: *"Selected members not found in the team."* Admin is prompted to correct the selection.	Pass
TC-03	Invalid Team Selection	1. Admin initiates remove member process. 2. Admin selects a non-existent team. 3. Submit.	Error message: *"Invalid team selection."* Admin is prompted to correct the team selection.	Pass
TC-04	Invalid Member Selection	1. Admin initiates remove member process. 2. Admin selects invalid members (e.g., non-existing members). 3. Submit.	Error message: *"Invalid member selection."* Admin is prompted to correct the member selection.	Pass

Test: Create and Assign Task

Test ID	Test Name	Steps	Expected Outcome	Result
TC-01	Successful Task Creation and Assignment	1. User requests task creation. 2. User enters valid task details. 3. Task is created. 4. User assigns task to valid participants/team members. 5. Submit.	Task is created and assigned to selected participants/team members. Notifications are sent, and confirmation is displayed.	Pass
TC-02	Invalid Task Details	1. User requests task creation. 2. User enters invalid task details (e.g., missing required fields). 3. Submit.	Error message: *"Invalid Task Details."* Admin is prompted to correct the task details.	Pass
TC-03	Invalid Assignment	1. User requests task creation. 2. User enters valid task details. 3. User selects invalid participants/team members (e.g., non-existing members). 4. Submit.	Error message: *"Invalid Assignment."* Admin is prompted to correct the assignment selection.	Pass
TC-04	No Assignment Selected	1. User requests task creation. 2. User enters valid task details. 3. User does not select any participants/team members for assignment. 4. Submit.	Task is created, but no assignments are made. Confirmation is sent to the user.	Pass

Test: Submit Task

Test ID	Test Name	Steps	Expected Outcome	Result
TC-01	Successful Task Submission with Proof	1. User selects task for submission. 2. User marks task as complete. 3. User uploads file or provides text as proof. 4. Submit.	Task is marked as complete, proof is stored securely, task is submitted, and confirmation is displayed.	Pass
TC-02	Successful Task Submission without Proof	1. User selects task for submission. 2. User marks task as complete. 3. No proof is provided. 4. Submit.	Task is marked as complete, no proof is stored, task is submitted, and confirmation is displayed.	Pass
TC-03	Invalid Task Submission	1. User selects task for submission. 2. User marks task as complete. 3. User uploads invalid proof (e.g., unsupported file type). 4. Submit.	Error message: *"Invalid Submission."* User is prompted to correct the submission (e.g., upload valid proof).	Pass
TC-04	Task Submission Without Marking as Complete	1. User selects task for submission. 2. User does not mark task as complete. 3. Submit.	Task is not submitted. Confirmation is displayed stating that task must be marked as complete before submission.	Pass

3.5 Summary

To define the interactions between users and the system for various features, this chapter explored the system's numerous use cases. The precise objectives, preconditions, triggers, primary success scenarios, alternate processes, and quality standards for a reliable implementation were described in each use case.

The use cases provide thorough information about the system's expected behavior for various tasks, including account management activities like account deletion and invitation handling as well as project-related procedures like team formation, member assignment, and task management. Crucial situations such as task submission and proof validation highlighted the significance of safe data management, effective validation, and unambiguous user communication via error warnings and confirmations.

Chapter 4 Deployment and Maintenance

4.1 Introduction

Deploying the software and ensuring it runs smoothly after release are the main topics of this chapter. It goes over how to make the application accessible to users, handle updates, and keep it dependable over time. Proper implementation and maintenance are crucial to provide a flawless experience and maintain the system's functionality and security.

4.2 Try to follow the SRLC (software release life cycle)

Development Phase

- Focus on building and testing features.
- Fix any issues found during development.

Alpha Release

- Early version tested internally by the team.
- Fix major bugs and ensure basic functionality works.

Beta Release

- Shared with a small group of users for feedback.
- Identify issues from real-world use and improve features.

Release Candidate (RC)

- A near-final version is thoroughly tested.
- Ensure everything works well in the production environment.

General Availability (GA)

- The final version is released to all users.
- Set up monitoring to track performance and errors.

Maintenance

- Regular updates to fix bugs and improve the software.
- Add new features as needed while ensuring the system stays stable.

Chapter 5 User Manual

5.1 Introduction

This chapter offers a user manual to assist users in comprehending and efficiently utilizing the system. A seamless user experience is ensured by the inclusion of pictures, detailed instructions, and explanations of important features and functionalities. The goal of the handbook is to ensure that all users, regardless of technical proficiency, can use the system.

5.2 Project Functionalities

User Authentication

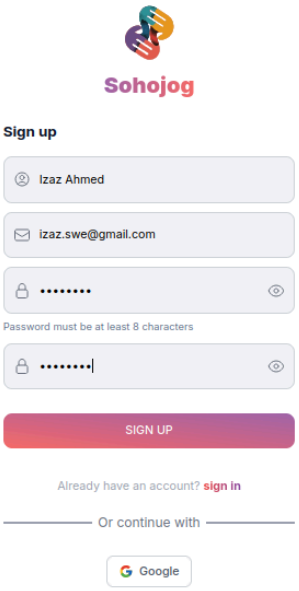


Figure: User Registration

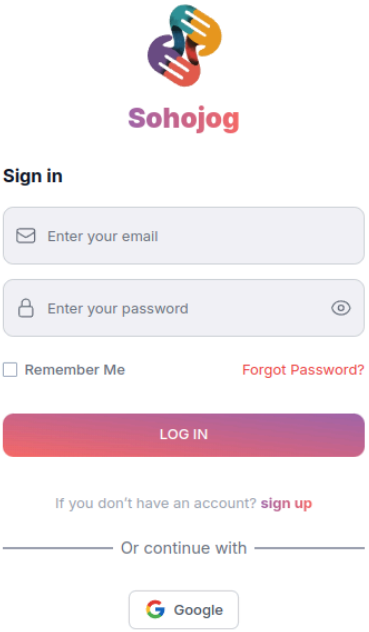


Figure: User Login

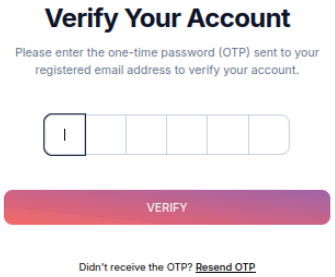


Figure: User Verification

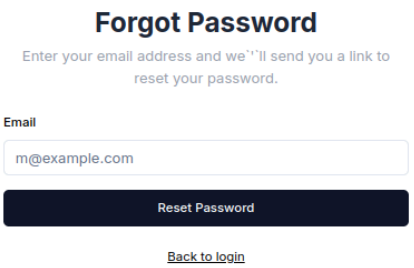


Figure: Reset Password

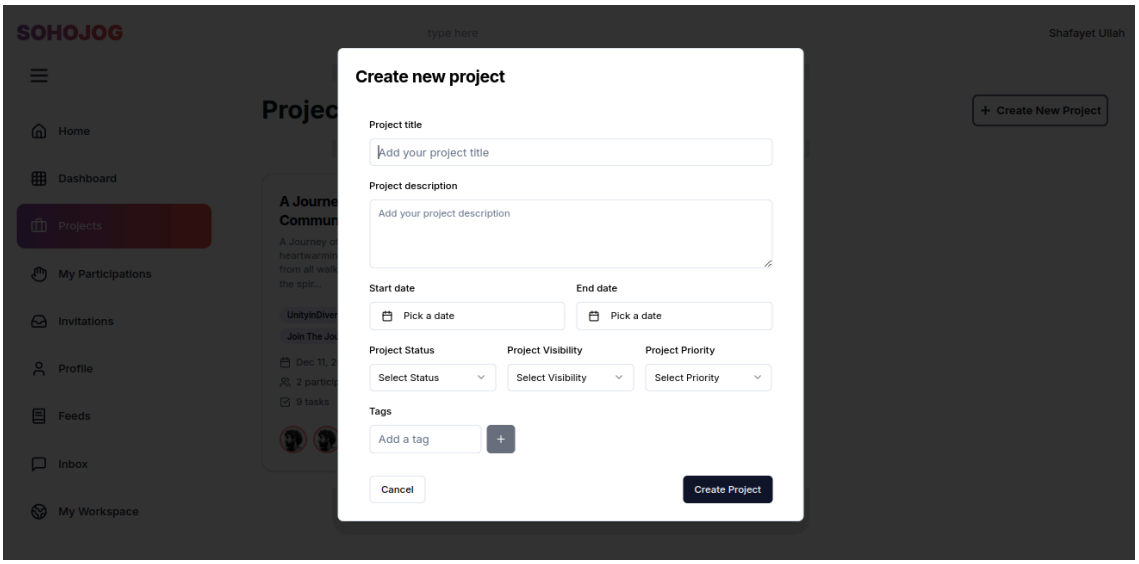


Figure: Create New Project

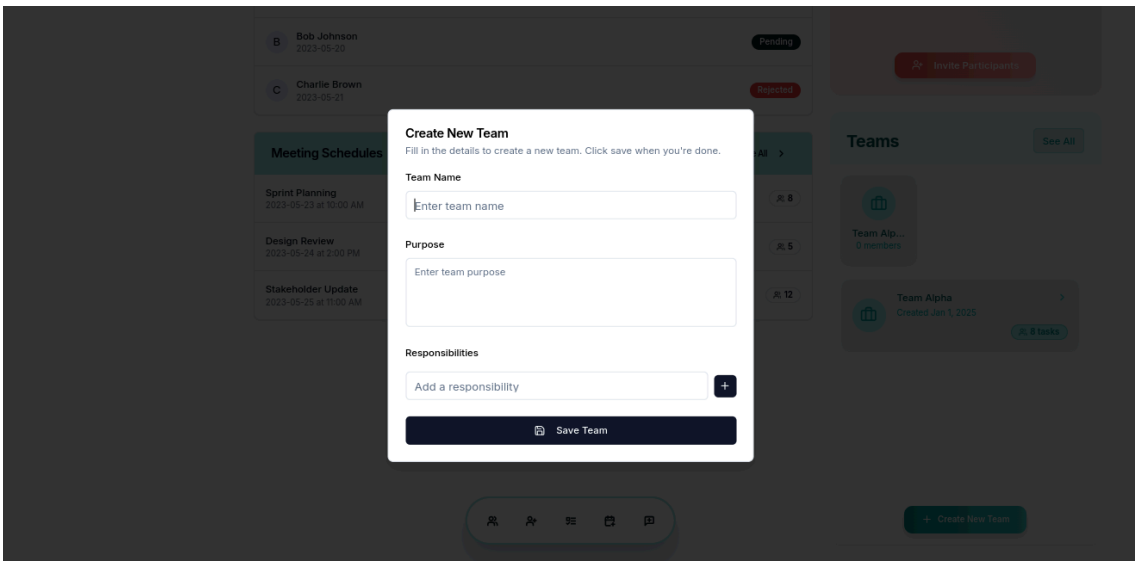


Figure: Create Team

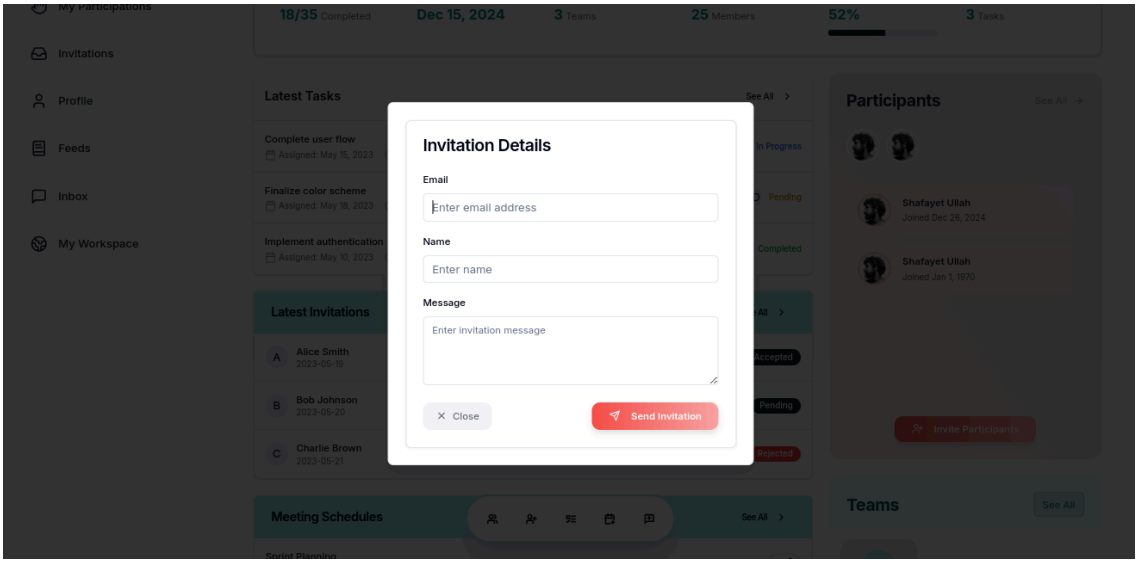


Figure: Send Invitaion

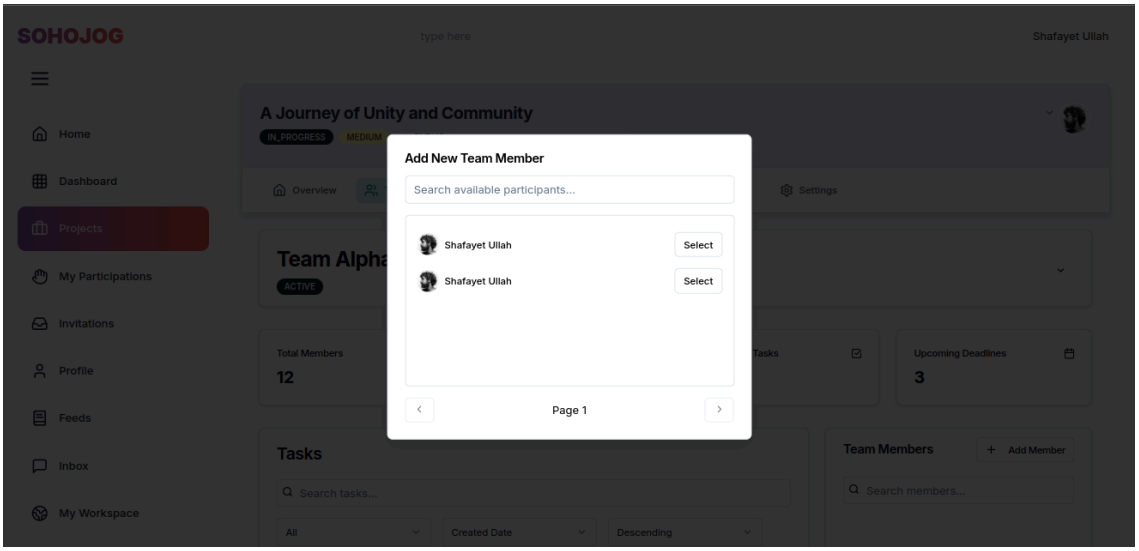


Figure: Add Team Member

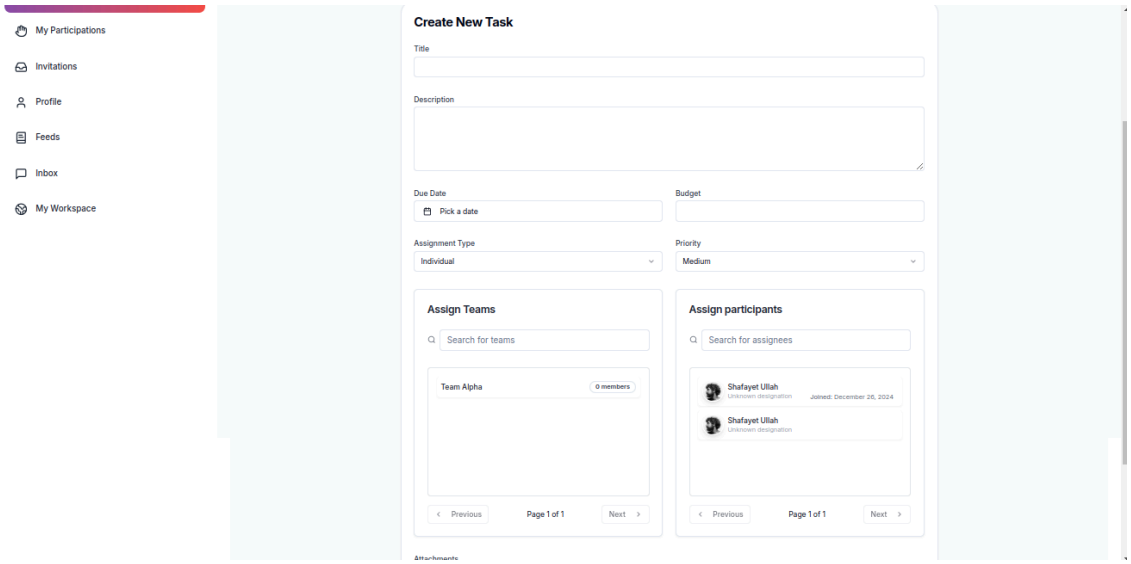


Figure: Create And Assign Task

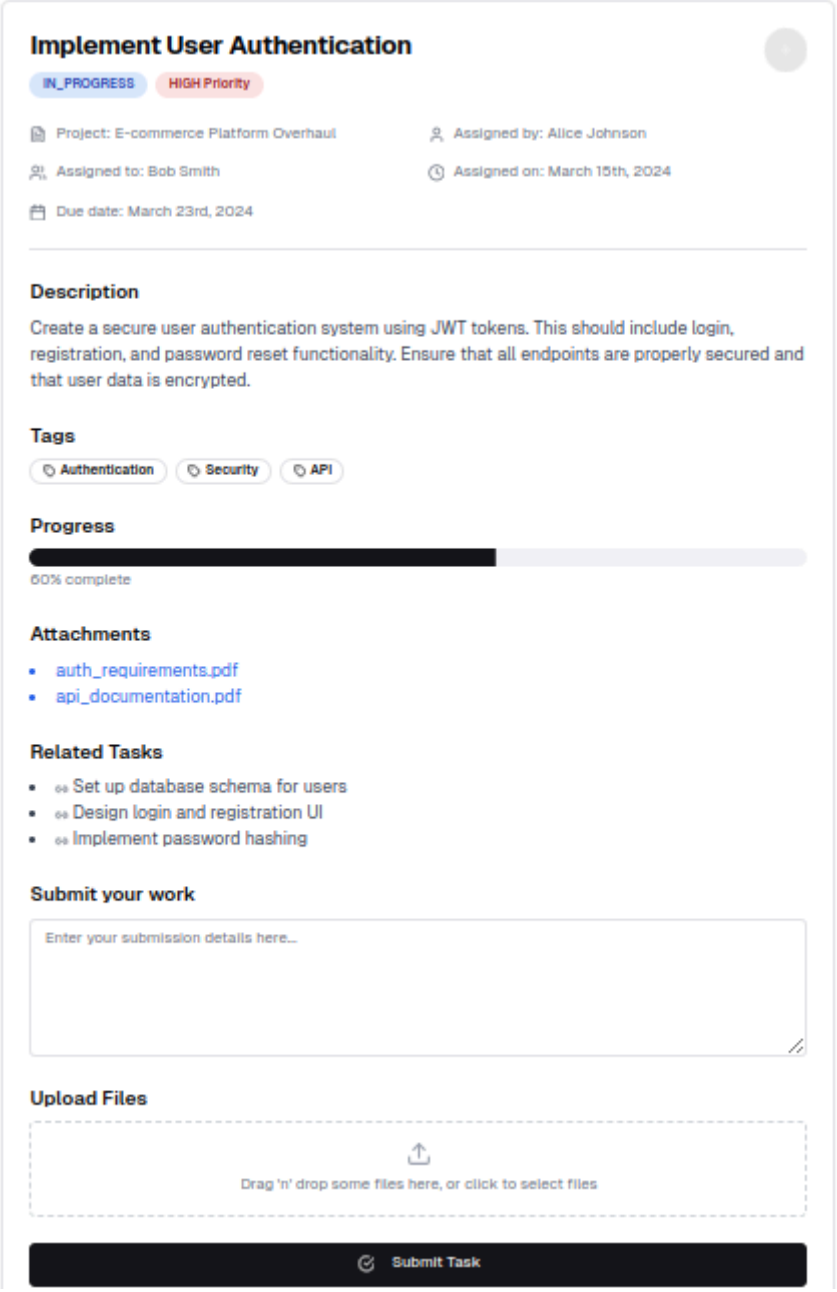


Figure : Submit Task

5.3 Summary

The workflow section provides a comprehensive overview of how users interact with the system, illustrated through screenshots. It begins with user access workflows, including registration, login, and OAuth-based authentication, showcasing how users enter the system securely and efficiently.

Once logged in, the dashboard serves as the central hub, allowing users to select a project and transition into the project context. Within this context, core functionalities like Team Management (creating, assigning, and removing team members) and Task

Management (creating, assigning, and submitting tasks) are performed seamlessly. The workflow also highlights the independence of Account Management, which is accessible system-wide for actions like updating user information or resetting passwords.

Each workflow is visually represented with clear and intuitive screenshots, emphasizing the logical flow of actions and ease of navigation. These visuals serve to reinforce the practical implementation of the system, ensuring users understand the sequence of operations and the system's functionality at every step.

Chapter 6 Project Summary

6.1 Introduction

The project is thoroughly reviewed in this chapter, which also offers insights into its broad scope, limitations, and strengths. It highlights the main features that have been put into place, admits any limitations or difficulties that have arisen during development, and points out areas that could use expansion. Through a retrospective of the project's trajectory, this chapter highlights its successes, assesses its influence, and lays the groundwork for upcoming improvements and innovations.

6.2 Project Limitation

The project faced the following constraints:

- **Time Constraints**: Limited time led to focusing on core features over advanced functionalities.
- **Technology Constraints**: Some technologies were not fully utilized due to compatibility or complexity.
- **Unaddressed Features**: Advanced analytics, real-time collaboration, and additional third-party integrations remain incomplete.
- **Scalability**: The current implementation may require optimization for large-scale usage.

6.3 Scope

The scope of this project defines the features and functionalities it covers, along with its boundaries.

Included Features:

- User registration, login, and OAuth integration for seamless access.
- Team and task management within project contexts to streamline collaboration.
- Account management functionalities, including profile updates and password reset.
- Basic chatting functionality to enhance communication among users.

Excluded Features:

- Advanced analytics and detailed reporting mechanisms.
- Real-time collaboration tools such as live editing or simultaneous updates.
- Video and audio calling features.
- Extensive third-party integrations beyond the essential ones.

This scope ensures the project meets core objectives while leaving room for future enhancements.

6.4 Future Work

Potential improvements and extensions include:

1. **Feature Enhancements**:
    - Real-time notifications and activity tracking.
    - Advanced analytics and reporting tools.
2. **Scalability**:
    - Optimize the system for handling larger teams and projects.
    - Implement distributed database solutions.
3. **Integrations**:
    - Add support for more OAuth providers.
    - Integrate with popular tools like Slack and Trello.
4. **User Experience**:
    - Refine the user interface for better usability.
    - Ensure the application is fully responsive and mobile-friendly.

6.5 Conclusion

The project successfully met its primary objective of creating a functional project management system. Core features such as user registration, login (including OAuth integration), team management, task assignment, account updates, and basic chat functionality were implemented effectively, providing a solid foundation for project organization and collaboration.

However, certain features were left out due to time constraints, such as advanced analytics, real-time collaboration tools like audio and video calls, and extensive third-party integrations. These omissions highlight areas for future development and improvement, ensuring the system can evolve to meet more complex user requirements.

Overall, the project demonstrates the importance of prioritizing essential requirements, delivering a usable and scalable solution while setting clear goals for future enhancements.

REFERENCES

Socket.IO Documentation
Socket.IO. (n.d.). Retrieved from https://socket.io/docs/

NestJS Documentation
NestJS. (n.d.). Retrieved from https://docs.nestjs.com/

Prisma Documentation
Prisma. (n.d.). Retrieved from https://www.prisma.io/docs/

Tailwind CSS Documentation
Tailwind CSS. (n.d.). Retrieved from https://tailwindcss.com/docs

Zod Documentation
Zod. (n.d.). Retrieved from https://zod.dev/

JWT Authentication in NestJS
Vojtech Licha. (2021). JWT Authentication in NestJS. Retrieved from https://medium.com/@vojtech.licha/jwt-authentication-in-nestjs-b4c6fe764491

Google OAuth 2.0 Documentation
Google. (n.d.). OAuth 2.0 for Web Server Applications. Retrieved from https://developers.google.com/identity/protocols/oauth2

Next.js Documentation
Next.js. (n.d.). Retrieved from https://nextjs.org/docs

APPENDICES

Appendix A: Socket.IO Implementation for Real-time Chat

What is Socket.IO?

Socket.IO is a library that enables real-time, bidirectional communication between the client and server. It allows instant messaging in web applications, ensuring that users can chat with each other without needing to refresh the page.

Why Socket.IO?

For this project, we used Socket.IO to implement the chat feature because it supports real-time communication, is easy to integrate, and works across different platforms. It ensures that messages are delivered instantly to users as soon as they are sent.

How it Works

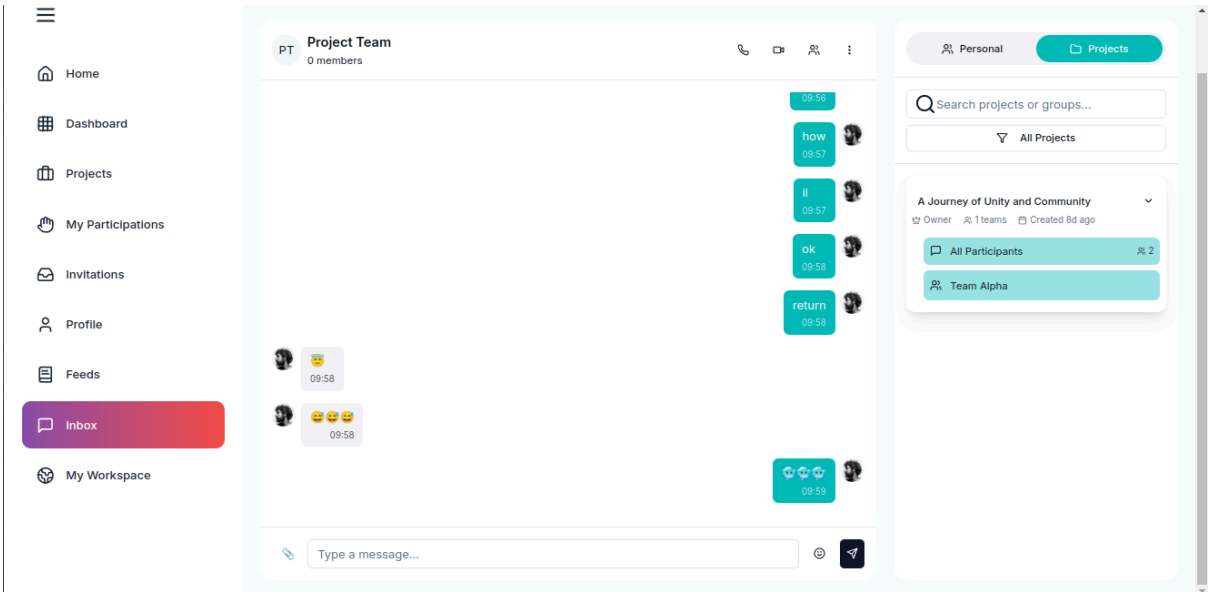
Socket.IO works by creating a persistent connection between the client (the user's browser) and the server. When a message is sent by one user, the server broadcasts it to all connected users in real-time. This setup ensures that all users can see the new messages instantly, without having to refresh their browsers.

Challenges

- **Real-time Updates**: Ensuring that messages were sent and received instantly without delays.
- **Connection Stability**: Ensuring the system remains stable even if a user disconnects and reconnects.

Test Cases

- **Test 1**: Messages should be displayed instantly for all connected users.
- **Test 2**: Messages should not be lost if a user disconnects and reconnects.
- **Test 3**: Messages should appear correctly and be received by the intended users.



SOHOJOG: A Collaborative Platform

ORIGINALITY REPORT

13%	7%	2%	11%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	4%
2	Submitted to HELP UNIVERSITY Student Paper	1%
3	Submitted to NCC Education Student Paper	1%
4	dspace.daffodilvarsity.edu.bd:8080 Internet Source	1%
5	Submitted to Kingston University Student Paper	1%
6	Submitted to INTI University College Student Paper	1%
7	Submitted to University of Greenwich Student Paper	<1%
8	Submitted to Kuwait University Student Paper	<1%
9	kahramanertan.com Internet Source	<1%

21	Submitted to Asia Pacific Instutute of Information Technology Student Paper	<1%
22	Submitted to Florida Institute of Technology Student Paper	<1%
23	Submitted to University of Technology, Sydney Student Paper	<1%
24	Submitted to Wawasan Open University Student Paper	<1%
25	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1%
26	Submitted to Macquarie University Student Paper	<1%
27	Ventura, André Fidalgo. "Sistema Colaborativo para a Identificação de Cultivares de Camélia", Universidade de Aveiro (Portugal), 2023 Publication	<1%
28	fastercapital.com Internet Source	<1%
29	Hsiang-Chuan Liu, Wen-Pei Sung, Wenli-Yao. "Information, Computer and Application Engineering", CRC Press, 2018 Publication	<1%