Final Year Project

UniVersa

Project Team:

Mr. Sandeep Kumar

2112127

Mr. Vivekanand

2112296

Project Supervisor:

Ms. Faria Jameel

July 1st , 2025

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science
in the

Faculty of Computing and Engineering Sciences

Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology University

(SZABIST University) Karachi Campus

.

Declaration of Authorship

We, the undersigned, hereby declare that the project titled *"**UniVersa**"* is our original work and has been completed in accordance with the academic standards of *SZABIST University* under the supervision of ***Faria Jameel***.

We confirm that all sources and references used during the research and development of this project have been properly cited and acknowledged. The content of this report is original and has not been copied or plagiarized from any external source.

We further declare that this project has not been previously submitted for the award of any degree, diploma, or other qualification at any other university or institution.

Name: Sandeep Kumar

Signature: _____

Date:_____

Name: Vivek Anand

Signature: _____

Date:_____

Project Description

UniVersa is a mobile application that is created to augment enhance communication and coordination within the educational ecosystem, specifically targeting universities and colleges. The app serves as a centralized platform to connect students, teachers, parents, and administrators, streamlining academic and administrative interactions. Built using Flutter for a cross-platform Android experience, UniVersa integrates a Node.js backend with MongoDB for secure and scalable data management. The application aims to improve engagement, accessibility, and efficiency in university operations through a comprehensive set of features.

Acknowledgement

In the name of Allah, the most beneficent and most merciful, who gave us the knowledge and courage to work on this project

We are grateful for the outcome and success of this project over the year. Our gratitude towards the people who have provided us with the guidance and assistance to be able to complete this project in such a difficult time.

We would like to thank our supervisor “Faria Jameel” of the computer science faculty at Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology. She was an integral part of the project as she was always there when we would get stuck at a point in a project or whenever we required any sort of suggestion. She constantly guided us, motivated us and cooperated with us throughout the duration of this project.

We would like to thank the teachers at Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology, who guided us and taught us throughout our time in the university. We would also like to express our gratitude to our parents and family members who helped and encouraged us during this time. Furthermore, we want to thank the staff at SZABIST for allowing us to use their labs and services to be able to complete the project.

Lastly, we would like to extend our gratitude to everyone at Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology for creating an environment for students to thrive in. The quality of education, the cooperative faculty members and the motivation provided by them.

Plagiarism-Free Certificate

This is to certify that the project titled **"UniVersa"** has been successfully completed and submitted by the undersigned students. The project was carried out under the supervision of **Faria Jameel** at **SZABIST University**

We hereby affirm that the content of this report is entirely original and has not been copied or reproduced from any external source. All references and materials used in the research and development of this project have been properly cited and acknowledged.

This project fully complies with the academic and ethical standards of **SZABIST University.** Furthermore, we declare that this work is our own and has not been submitted elsewhere for the award of any degree or qualification.

Name: _____ **Name:** _____

Registration #: _____ **Registration #:** _____

Signature: _____ **Signature:** _____

Supervisor: _____

Signature: _____

Table Of Content

Table of Figures

Table Of Tables

Proposal

1. Introduction

UniVersa is designed to address the needs of university and college communities by providing a platform for seamless communication and event management. Its Flutter-based development ensures a robust and user-friendly experience, integrating various features that support academic and social engagement.

2. Objective

A mobile application to be designed with Flutter, being able to scale up for the future to the largest extent and can be made extremely secure to the best of its ability, effectively serving as a one-stop platform for communication by university/college students, teachers & parents with ease of use in features such as event notifications/co-ordination, timetable, etc.

3. Problem Description

Against this backdrop, the growing use of technology for communication and coordination is pushing universities away from real-time interaction between students, teachers, and parents. This infrastructure is often fragmented; there rarely is secure messaging, real-time notifications, or a single place where all academic timetables and events are available. UniVersa to solve: a single-point solution for all university/college communication and scheduling requirements. It makes it easy to organize events, scheduling, and teacher notifications and also includes a job board for career opportunities. With a focus on security and scalability, the app can cater to the modern-day needs of academic institutions without compromising user experience.

4. Target Industry

Education sector: Universities, colleges, and other higher education institutions.

5. Methodology

UniVersa in Flutter for cross-platform apps This is managed by having a backend using Node JS server and MongoDB for user data, events, schedules, and messages storage. In this phase, we would also add API integrations for external data (ex: job listings, and event details). A user-centered design process will be used including rapid prototyping, informing initial spec development and testing phases to ensure user satisfaction as well as a stable and fully functional platform.

6. Project Scope

Universa will focus on:

- Secure login and authentication.
- Real-time communication (event notifications, timetables).

- Social interaction (private messaging and group discussions with teachers).
- Carpool Features (Students can communicate with each other if they are traveling the same route or live in the same area.)
- Parents can monitor their children's university schedules and activities. Parents will also have access to the app.

7. Feasibility Study

- **Risk Involved:** Possible delays due to complex backend integration and testing phases. We plan to mitigate these by thorough initial planning and breaking down tasks into manageable phases.
- **Resource Requirement:** Laptops/PCs for development, Flutter SDK, Node JS backend, design software (Figma for UI/UX), and MongoDB for database management.

8. Solution Application Areas

UniVersa will also assist schools/colleges/universities by giving them a secure way to communicate with students, faculty, and parents. This will enable them to improve their efficiency in event management, academic planning, and overall coordination. As well, the job board will provide a chance for academic growth to faculty and crew at large.

9. Tools/Technology

- Frontend: Flutter,Dart
- Backend: Node.js
- Database: MongoDB
- Design: Figma (UI/UX)
- Testing: Manual testing on android device/emulator
- Version Control: Git, GitHub

10. Expertise of the Team Members

Each team member has experience in the development of mobile applications using Flutter, as well as knowledge from courses on mobile application development and software engineering. We as a group have previously worked on Android Studio but not in depth. We have registered for the Flutter Development course which will help us in FYP. We as a group have mutual interest in the project.

11. Milestones

FYP-I (Fall Semester):

- Requirements gathering and planning (Week 1-2)
- UI/UX design and prototype development (Week 3-5)
- Backend setup and database integration (Week 6-8)
- API integration and event management module (Week 9-12)
- Initial testing and debugging (Week 13-15)

FYP-II (Spring Semester):

- Messaging system and notification module (Week 1-4)
- Timetable and teacher notifications implementation (Week 5-7)
- Job board and route-sharing features (Week 8-10)
- Final testing, feedback incorporation, and deployment (Week 11-14)
- Documentation and final presentation (Week 15-16)

12. Project Schedules

FYP SCHEDULE

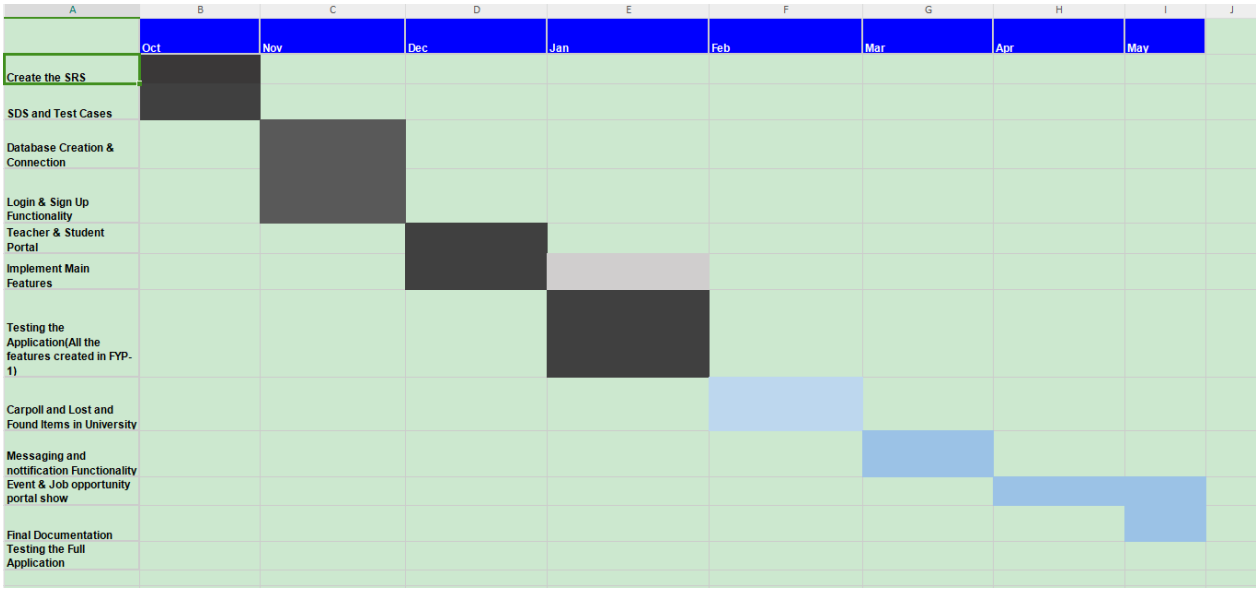

Figure 1 FYP Schedule

FYP 1 SCHEDULE

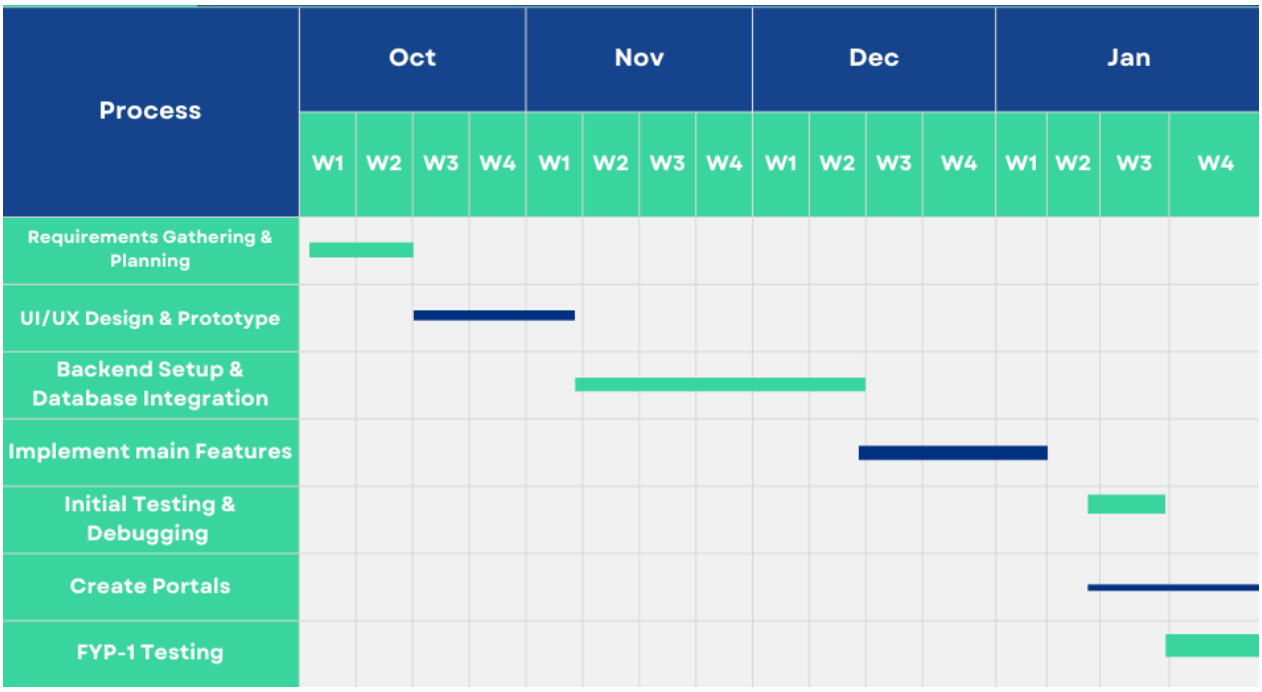

Figure 2 FYP I Schedule

FYP 2 SCHEDULE

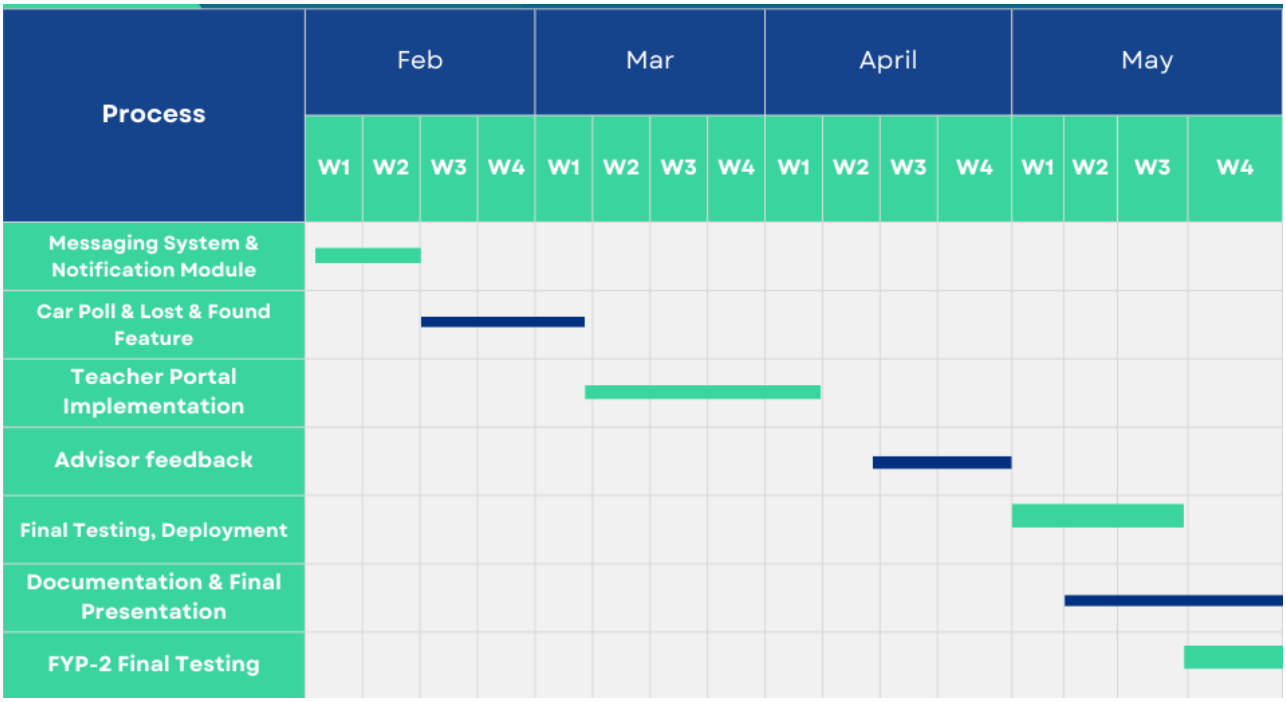

Figure 3 FYP Schedule II

13. Work Breakdown Structure

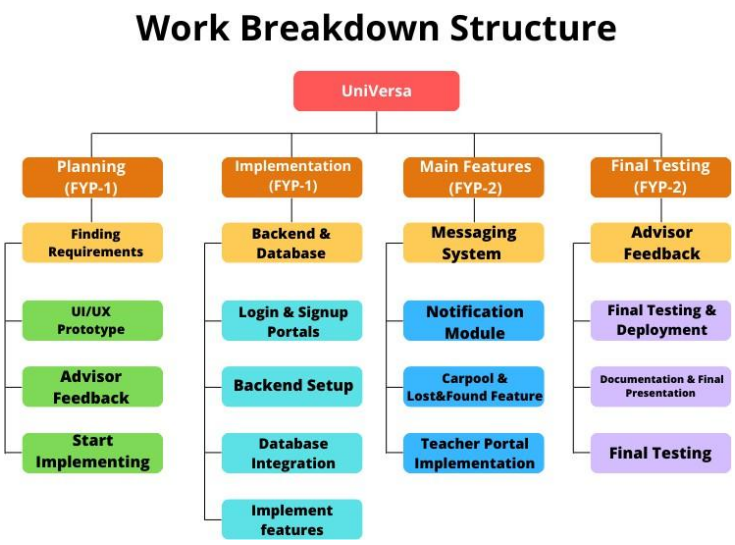

Figure 4 Work Breakdown Structure

Software Requirement Specifications

1. Introduction

1.1 Purpose

This SRS document is supposed to detail out the requirements and functionalities related to the **UniVersa** mobile application. The application shall act as a smooth communication channel between universities and colleges, serving their students, teachers, and parents alike. The SRS provides the basis for describing the system features along with design constraints and non-functional requirements necessary for developing the application by arriving at a comprehensive understanding over the scope and capabilities of the application.

1.2 Document Conventions

Heading	Font Size: 14pts Font Style: Times New Roman
Regular Text	Font Size: 12 pts Font Style: Times New Roman

Table 1 Document Conventions

1.3 Intended Audience and Reading Suggestions

The following are the major targets of this document:

- **Developers**: Involved in doing the code and technical implementation. They shall have a particular concentration on Section 4: System Features.
- **Testers**: Charged with the testing of functionality to ensure everything works as required. Shall be concentrating on Section 4: System Features and Section 5: Non-functional Requirements.
- **Project Managers**: Sad to say, it happens—the project is to follow a structured approach. They should read the entire document, with a focus on Section 2: Overall Description and Section 6: Other Requirements.
- **Stakeholders**: University administration and other decision-makers who need some knowledge of product scope and features. Sorting by high-level information, they may want to read Section 2 and Section 4.

1.4 Product Scope

UniVersa targets the education sector, universities, and colleges. It will promote better communication and coordination among students, teachers, and parents with its key feature set comprising event management, timetable synchronization, secure messaging, job boards, and carpooling features. It shall be cross-platform on Android using Flutter, and it will include scalability and security with a Node.js backend using MongoDB for data management.

2. Overall Description

2.1 Product Perspective

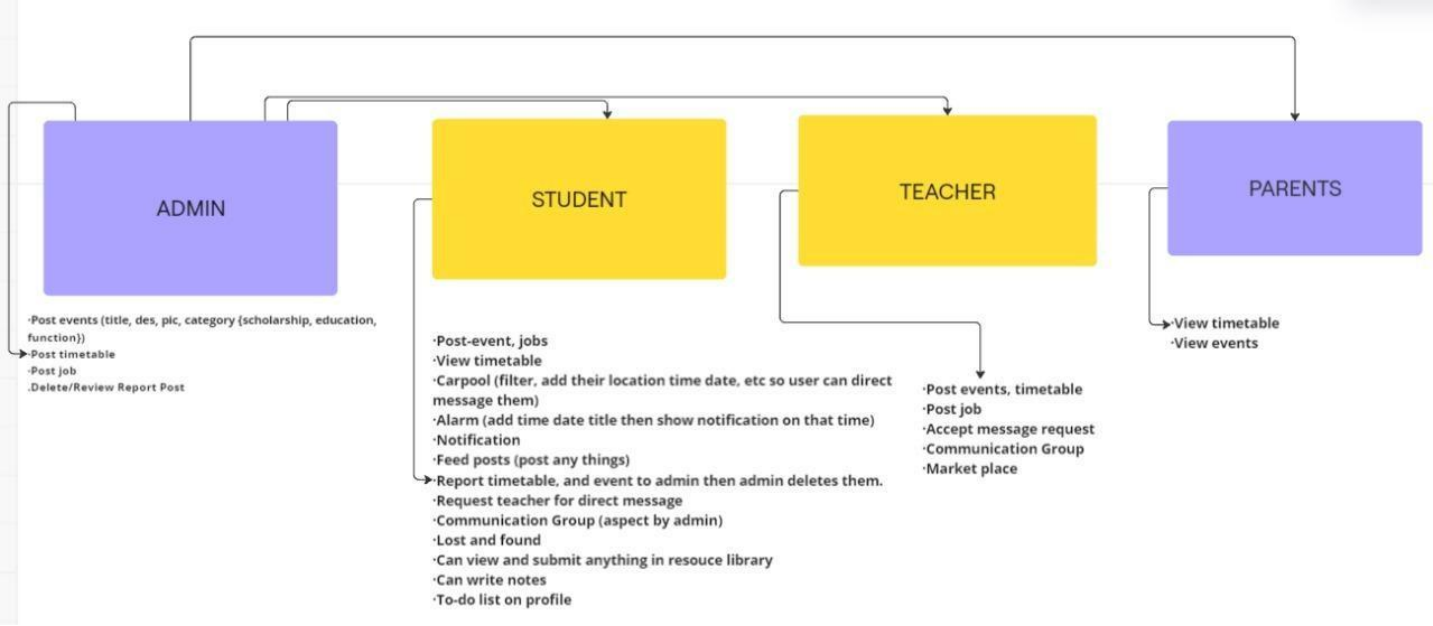

Figure 5 Product Perspective

2.2 Product Functions

2.2.1 Secure Login and Role-Based Authentication

- Functionality: Allow secure user account creation, login, and access to app features.
- Account Creation: Users can create accounts using a university-provided email address, with securely hashed and stored passwords.
- Login Instructions: Users log in with an email address and password.
- Role Management: Access is role-based, offering different features for students, teachers, and parents.

2.2.2 Real-time Notification

- Functionality: The real-time notification of the application will be very much engaging for the users regarding different updates so that they can be well-aware of any upcoming important events.
- Event Notifications: Events, class changes, and vital announcements will be informed to the users. Customizable notification settings provide the user with the ability to decide what notifications they want to receive—events reminders and teacher messages.

2.2.3 Event Management and Timetable Coordination

- Functionality: It allows the teacher to add and manage events and the student can view and respond to those events.
- Event Management: Instructors can create an event by filling in the name, date, time, venue, and description of the event; they can also select whether RSVPs are needed or not.
- Editing and Deleting Events: Events can be deleted and/or edited by the teachers, automatically notifying the modification to the users concerned.
- Student RSVP: This allows students to state whether they are attending events, maybe, or not. This would give the teacher some idea regarding participation.
- Timetable synchronization: Through the class timetable viewed in this application, users are able to manage their time effectively with event notifications.

2.2.4 Messaging System

- Functionality: A secure messaging system that students, teachers can use for communication along with parents.
- Direct Messaging: This allows users to send and receive messages to/from other users in private.
- Group Chats: Users can create group chats with regard to specific classes, projects, or discussions in which they are engaged. This fosters collaboration.
- Media Sharing: It enables the user to send images, documents, and links within messages, hence their communication is more effective.
- Message Encryption: The messages are encrypted end-to-end; thus, user privacy is completely ensured.

2.2.5 Job Board

- Core functionality: Centralized job board for students seeking jobs and internships.
- Vacancies: The job board will pinpoint vacancies within the university, as well as those available outside the institution, including details of the job title, description, requirements, and application process.
- Filtering and searching can be done on the basis of category—internships, part-time, full-time, location, and date for job postings to show relevant opportunities in the fastest time.
- Application Links: Each job posting contains links that direct the applicant directly to the application process or further information regarding the position.

2.2.6 Carpool Feature

- Functionality: This feature allows students to coordinate rides with each other, reducing transportation costs and environmental impact.

- Location Sharing: One can even share their location and intended routes so that others around are able to find carpool companions.
- Matchmaking: The system will be suggesting possible carpool matches based on proximity, route similarity, and schedules to make it easier for students to find suitable travel companions.

- In-app communication is possible wherein the user can communicate with the rider in the app, showing the details of ride information, timing, and pickup locations.

2.2.7 Parent Monitoring

- Functionality: The feature contributes to the establishment of parents' access to the children's activities. This is helpful in fostering the cause of transparency and engagement.
- Access Scheduling: The system makes it possible for parents to get a glimpse of their children's class and event schedule.
- Notifications: Events about attendance, going on at school, etc., are notified to the parents by the teachers to keep the parents involved in their children's education.

2.3 User Classes and Characteristics

2.3.1 Students

2.3.1.1 Characteristics:

- Primary users of the app.
- Experienced with mobile applications and technology.
- Are in need of easy access to academic information and communication facilities.

2.3.1.2 Functionalities:

- View class schedules and events.
- Receive notifications on important announcements and updates.
- Access the job board for career opportunities.
- Send and receive messages to/from teachers and peers.
- RSVP to events; manage personal schedules.

2.3.2 Teachers

2.3.2.1 Characteristics:

- Teachers in charge of organizing academic activities.
- Require tools to facilitate communication with students and parents.

2.3.2.2 Functionalities:

- Create, edit, and delete events.
- Posting of announcements and updates on important matters for students and parents.
- Manage class schedules and communicate directly with students.
- View RSVP statuses of events they organize.
- Use the messaging system to communicate with students and parents.

2.3.3 Parents

2.3.3.1 Characteristics:

- Guardians of students who want to be up-to-date with the school life of their child.
- Need an easy interface to reach the relevant information.

2.3.3.2 Functionalities:

- View their child's class schedules and events.
- Receive notifications on essential academic updates.

2.3.4 Admin

2.3.4.1 Characteristics:

- Responsible for managing the overall functionality and user management of the app.
- Typically affiliated with the university or college administration.
- Requires full access to all features and data of the application.

2.3.4.2 Functionalities:

- **User Administration:** Create, edit, and disable accounts of students, teachers, and parents within the app.
- **Event Management:** Review, approve, or edit events created by teachers to ensure consistency with institutional policies.
- **Manage Notifications:** Post important announcements intended for all users for timely dissemination.
- **Reporting:** Access analytics and reports on app usage, events, and user engagement to assess the app's impact and improve features.
- **System Settings:** Configure system settings, including notification preferences and adjusting access permissions for various user roles.

2.4 Operating Environment

Mobile Environment: Android (minimum version 5.0).

2.5 Design and Implementation Constraints

- Front End: Flutter
- Database: MongoDB
- Backend: NodeJS
- Android Studio

2.6 User Documentation

- There will be a user manual provided for the students and faculty articulating how to navigate and use the features of the app.

- Online help and tutorial videos will guide new users through account setup, communication tools, and event management.

- A section of FAQs in the app will solve small queries of the users.

2.7 Assumptions and Dependencies

- Users will have **internet access** for the app to function effectively.
- The app assumes **external APIs** (e.g., job boards) will remain accessible and stable for real-time integration.
- The system is **dependent on device compatibility** with the latest Flutter libraries.

3. External Interface Requirements

3.1 User Interfaces

Figure 6 Signup

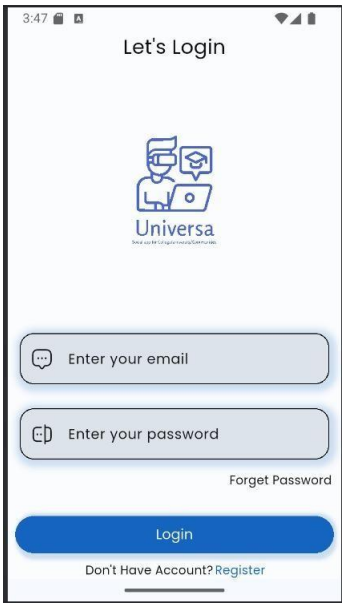

Figure 7 Login

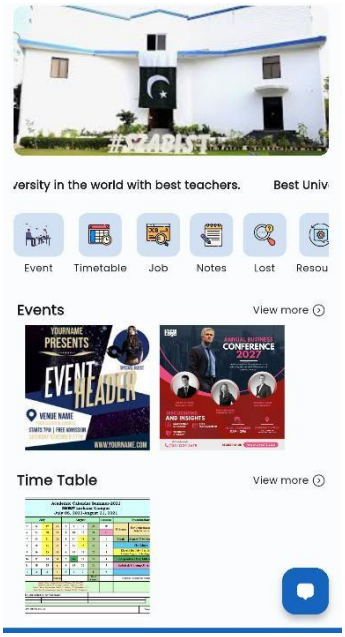

Figure 8 Dashboard

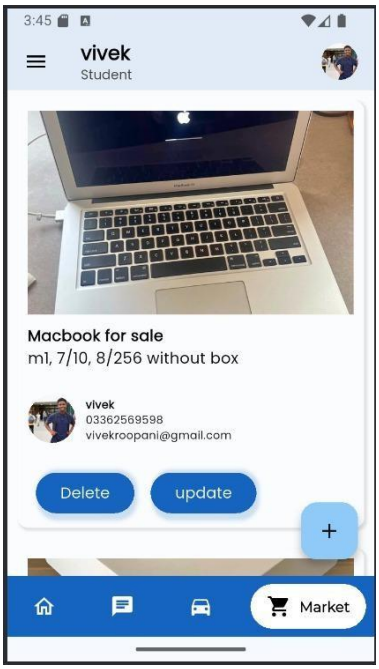

Figure 9 Marketplace

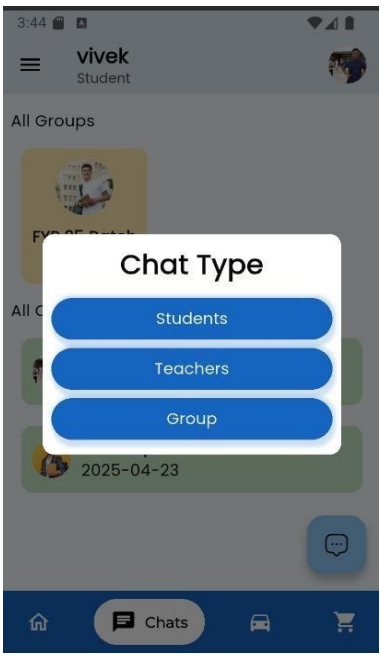

Figure 10 Chat Type

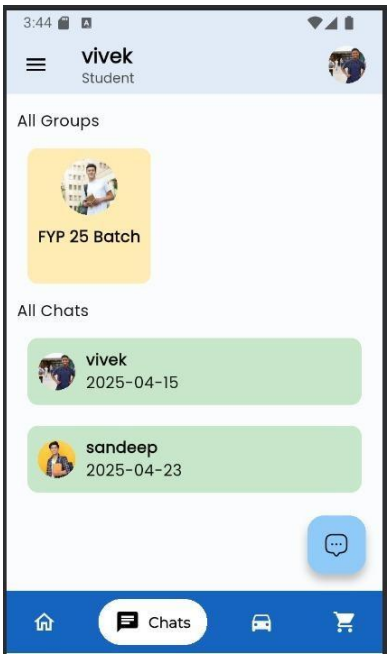

Figure 11 Chat Page

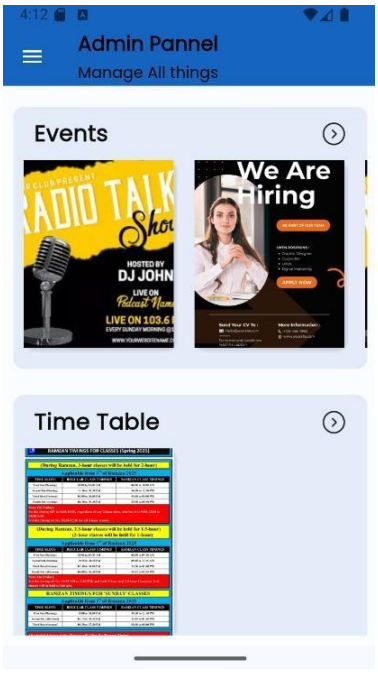

Figure 13 Admin Portal

Figure 12 Inbox Page

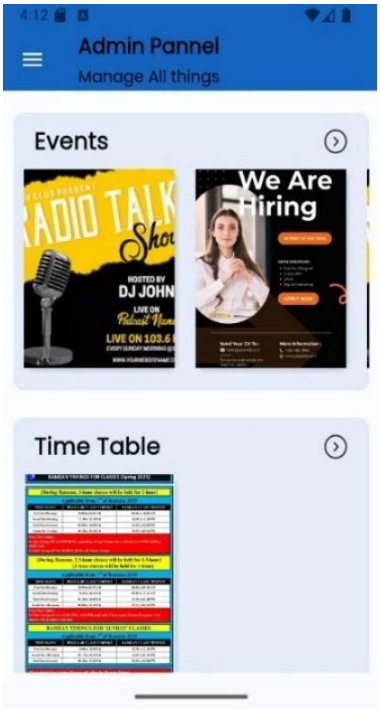

Figure 14 Teacher panel

3.2 Hardware Interfaces

There will be no hardware interface since the only way for the user to interact with the application would be through a web browser.

3.3 Software Interfaces

- It will employ Node.js on the backend to manage the server, user authentication, event manager, and message storage.
- MongoDB will securely store user information, events, and messages.
- Flutter will power the mobile user interface, ensuring compatibility across Android platform.

3.4 Communications Interfaces

The application will communicate on secure HTTPS protocols. Push notifications, like real-time updates to the users, will be done through Firebase Cloud Messaging or FCM. It ensures encrypted data transfer between a mobile application and a backend server to maintain sensitive information private.

4. System Features

4.1 User Authentication and Account Management

4.1.1 User Signup

4.1.1.1 Description

It provides a feature for logging into the UniVersa application securely using valid credentials like email and password. This assigns role-based access for students, teachers, parents, and admin users.

4.1.1.2 Stimulus/Response Sequences

Use Case Name: User Signup
Description: Allows users to create accounts on the platform.
Actors: All Users (Admin, Teacher, Student, Parent)
Goal: Enable users to securely register.
Pre-Conditions: User has not registered before.

Basic Course of Events: **Step 1:** User navigates to the signup page. **Response:** System displays the signup form. **Step 2:** User enters details and submits the form. **Response:** System validates and sends an activation email. **Step 3:** User clicks the activation link. **Response:** System activates the account.
Alternate Flow (Invalid Inputs): • **Step:** User enters invalid or incomplete information. • **Response:** System highlights errors and requests corrections.
Exception Flow (System Errors): • **Step**: System fails due to server issues. • **Response**: Displays "Signup unavailable."
Post-Conditions: User account is created and ready.

Table 2 Signup Response

4.1.1.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Provide a form to input user registration details.	High	Pending
REQ-2	Validate user information and send an activation link.	High	Pending
REQ-3	Activate the user account upon confirmation.	High	Pending
REQ-4	Display errors for invalid input during signup.	Medium	Pending

Table 3 Singup Requirements

4.1.2 User Login

4.1.2.1 Description

Administrators can schedule new events by entering details such as event title, date, time, location, and description. They can also assign events to specific categories or groups.

4.1.2.2 Stimulus/Response Sequences

Use Case Name: Add Event
Description: Allows registered users to securely log into the system.
All Users: (Admin, Teacher, Parent, Student)
Goal: Enable users to access their accounts securely.
Pre-Conditions: User has an active and verified account.
Basic Course of Events/Main Flow: 1. **User Action:** User navigates to the login page and enters credentials. • **System Response:** System validates the entered credentials. 2. **User Action:** User submits the credentials. • **System Response:** System grants access and redirects to the dashboard.

Alternate Flow (Invalid Inputs): 1. **User Action:** User enters incorrect or unrecognized credentials. • **Response**: System displays “Invalid username or password.”
Exception Flow (System Errors): • **User Action:** System fails to log in due to server issues.. • **System Response:** System displays “Login is temporarily unavailable. Please try again later."
Post-Conditions: User is logged in and redirected to their dashboard.

Table 4 Add Event Response

4.1.2.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Provide a form for entering login credentials.	High	Pending
REQ-2	Validate credentials against the user database.	High	Pending
REQ-3	Grant access to users upon successful validation.	High	Pending
REQ-4	Display error messages for incorrect credentials.	Medium	Pending

Table 5 Add Event Requirements

4.1.3 Forget Password

4.1.3.1 Description

This feature allows administrators to update or modify any aspect of an existing event, ensuring that information remains current and accurate.

4.1.3.2 Stimulus/Response Sequences

Use Case Name: Forget Password
Description: Allows users to reset forgotten passwords and regain account access.
Actors: All Users (Admin, Teacher, Parent, Student)
Goal: Enable users to securely reset their password..
Pre-Conditions: User has a registered and active account.
Basic Course of Events/Main Flow: • **Step 1:** User clicks on the “Forgot Password” option. • **Response:** System prompts for the registered email address**.** • **Step 2:** User enters their registered email and submits the request. • **Response:** System validates the email and sends a reset link. • **Step 3**: User clicks the reset link and enters a new password. • **Response:** System updates the password and confirms success.
Alternate Flow (Invalid Inputs): • **Step:** User enters an unregistered email address. • **Response:** System displays “Email not found. Please try again.”

Exception Flow (System Errors): • **Step:** Password reset fails due to technical issues.   • **Response:** System displays “Password reset unavailable. Please try again later.”
Post-Conditions: User successfully resets their password.

Table 6 Forger Password Responses

4.1.3.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Provide an option to request a password reset.	High	Pending
REQ-2	Validate the entered email against the database.	High	Pending
REQ-3	Send a secure password reset link to the user email.	High	Pending
REQ-4	Allow the user to set a new password securely.	Medium	Pending

Table 7 Add Event Requirements

4.1.4 Account Activation

4.1.4.1 Description

Administrators have the capability to delete events that are no longer relevant or have been canceled, helping to keep the event calendar up-to-date.

4.1.4.2 Stimulus/Response Sequences

Use Case Name: Account Activation
Description: Activates a user account after registration through email verification.
Actors: All Users (Admin, Teacher, Parent, Student)
Goal: Ensure the authenticity of user accounts.
Pre-Conditions: User has registered successfully but not activated their account.

Basic Course of Events/Main Flow:
• **Step 1:** User receives an email with an activation link.
  • **Response:** System provides a secure activation link.
• **Step 2:** User clicks on the activation link.
  • **Response**: System validates the link and activates the account.
• **Step 3:** System confirms successful activation to the user.

Alternate Flow: • **Step:** User clicks on an expired activation link.   • **Response**: System displays “Activation link expired. Request a new link.”
Exception Flow (System Errors): • **Step:** Account activation fails due to technical issues.   ○ **Response**: System displays “Account activation unavailable. Please try again later.
Post-Conditions: User account is activated and ready for use.

Table 8 Account Activation Response

4.1.4.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Send an activation email with a secure link.	High	Pending
REQ-2	Validate the activation link before confirming.	High	Pending
REQ-3	Display appropriate errors for invalid/expired links.	Medium	Pending
REQ-4	Confirm successful activation to the user.	Medium	Pending

Table 9 Account Activation Requirements

4.1.5 Event Category Management

4.1.5.1 Description

Administrators can manage (create, edit, and delete) categories that help organize events into groups such as sports, academics, and extracurricular, facilitating easier navigation and search ability.

4.1.5.2 Stimulus/Response Sequences

Use Case Name: Manage Event Categories
Description: Admin can create, edit, and delete categories for events to better organize and filter them.
Actors: Admin
Goal: Organize events into categories for efficient management and accessibility**.**
Pre-Conditions: Admin is logged in.

Basic Course of Events/Main Flow:
• **Step 1:** Admin navigates to the "Event Categories" section.
  • **Response:** System displays existing categories and options to add, edit, or delete.
• **Step 2:** Admin adds a new category or selects an existing category to edit or delete and submits changes.
  • **Response:** System validates and updates the categories in the database, confirms actions to the admin.

Alternate Flow (Invalid Inputs): • **Step:** Admin provides invalid category names or details.   • **Response:** System shows error messages and requests corrections.
Exception Flow (System Errors): • **Step:** Failures occur due to system issues during updates.   • **Response:** System displays "Unable to update categories" error message**.**
Post-Conditions: Event categories are updated in the system.

Table 10 Event Category Response

4.1.5.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Provide interface for managing event categories.	High	Pending
REQ-2	Validate category details for correctness and uniqueness	High	Pending
REQ-3	Update category details in the database.	High	Pending
REQ-4	Notify admin of successful update or failure.	Medium	Pending

Table 11 Event Category Requirements

4.1.6 Add Timetable

4.1.6.1 Description

This function allows administrators to create new timetables by defining class times, assigning instructors, and linking to specific courses or programs.

4.1.6.2 Stimulus/Response Sequences

Use Case Name: Add Timetable
Description: Admin can create timetables for classes or events, scheduling them in the system.
Actors: Admin
Goal: Efficiently schedule and manage timetables within the institution.
Pre-Conditions: Admin is logged in.

Basic Course of Events/Main Flow:
• **Step 1:** Admin accesses the "Timetable Management" interface.
  • **Response:** System displays a form to create a new timetable.
• **Step 2:** Admin fills out the timetable details and submits.
  • **Response:** System validates the timetable, saves it, and confirms creation to the admin.

Alternate Flow (Invalid Inputs): • **Step:** Admin submits incomplete or incorrect timetable details.   • **Response:** System displays error messages and requests corrections**.**
Exception Flow (System Errors): • **Step:** Submission fails due to technical issues.   • **Response:** System displays "Unable to create timetable" error message.
Post-Conditions: A new timetable is added to the system.

Table 12 Add Timetable Response

4.1.6.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Provide a form for inputting timetable details.	High	Pending

REQ-2	Validate timetable details against existing schedules.	High	Pending
REQ-3	Save timetable details to the database.	High	Pending
REQ-4	Confirm creation or display error messages	Medium	Pending

Table 13 Add TimeTable Requirements

4.1.7 Remove Timetable

4.1.7.1 Description

Administrators can delete timetables that are outdated or no longer in use.

4.1.7.2 Stimulus/Response Sequences

Use Case Name: Remove Timetable
Description: Allows the admin to delete existing timetables from the system.
Actors: Admin
Goal: Ensure the system's timetable is up-to-date by removing outdated or unnecessary schedules.
Pre-Conditions: Admin is logged in and has identified a timetable to delete**.**

Basic Course of Events/Main Flow:
• **Step 1:** Admin selects a timetable from the list in the "Timetable Management" section.
  • **Response:** System displays a confirmation prompt to verify the desire to delete.
• **Step 2:** Admin confirms the deletion.
  • **Response:** System deletes the timetable and updates the database, notifies admin of the deletion success.

Alternate Flow: • **Step:** Admin decides to cancel the deletion after the confirmation prompt.   • **Response:** System cancels the deletion process, and no changes are made
Exception Flow (System Errors): • **Step: Technical issues prevent the deletion (e.g., database errors).**   • **Response: System displays "Error deleting timetable. Please try again later."**
Post-Conditions: The timetable is removed from the system.

Table 14 Remove Timetable Response

4.1.7.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Provide a confirmation prompt before deleting a timetable.	High	Pending
REQ-2	Successfully delete the timetable from the database upon confirmation.	High	Pending
REQ-3	Notify the admin of the outcome of the deletion.	High	Pending

Table 15 Remove TimeTable Requirements

4.1.8 Update Timetable

4.1.8.1 Description

Enables administrators to make changes to existing timetables, such as altering class times, changing instructors, or adjusting the rooms.

4.1.8.2 Stimulus/Response Sequences

Use Case Name: Update Timetable
Description: Admin modifies details of existing timetables to adjust to new requirements or corrections.
Actors: Admin
Goal: Maintain accurate and functional timetables within the institution**.**
Pre-Conditions: Admin is logged in and selects a timetable to update.

Basic Course of Events/Main Flow:
• **Step 1:** Admin accesses the "Timetable Management" interface and selects a timetable to edit.
  • **Response:** System displays the timetable details in an editable form.
• **Step 2:** Admin makes necessary changes and submits the updated details.
  • **Response:** System validates the updates, saves them in the database, and confirms the update to the admin.

Alternate Flow (Invalid Inputs): • **Step:** Admin provides invalid or incomplete updated details.   • **Response**: System displays error messages requesting correct information**.**
Exception Flow (System Errors): • **Step:** Failures occur due to system issues during the update.   • **Response:** System displays "Unable to update timetable. Please try again."
Post-Conditions: Timetable details are updated in the system**.**

Table 16 Update TimeTable Response

4.1.8.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Provide an editable form with current timetable details.	High	Pending
REQ-2	Validate updated details for accuracy and completeness.	High	Pending
REQ-3	Update the timetable details in the database.	High	Pending
REQ-4	Notify admin of the update success or failure.	Medium	Pending

Table 17 Update TimeTable Requirements

4.1.9 Program Category Management

4.1.9.1 Description

Administrators manage categories for various educational programs, allowing them to organize and classify programs under academic, vocational, and extracurricular tags for better accessibility.

4.1.9.2 Stimulus/Response Sequences

Use Case Name: Manage Program Categories
Description: Admin can add, edit, or delete categories for academic programs to enhance organization and accessibility.
Actors: Admin
Goal: Efficiently manage and categorize academic programs**.**
Pre-Conditions: Admin is logged in.

Basic Course of Events/Main Flow:
• **Step 1:** Admin navigates to the "Program Categories" section.
  • **Response:** System displays existing categories with options to add, edit, or delete.
• **Step 2:** Admin performs desired actions on the categories and submits changes.
  • **Response:** System validates and updates the categories in the database, confirms the actions to the admin.

Alternate Flow (Invalid Inputs): • **Step:** Admin submits invalid or duplicate category names.   • **Response:** System displays error messages such as "Duplicate category name."
Exception Flow (System Errors): • **Step:** Failures occur due to database or network issues.   ○ **Response:** System displays "Error updating categories. Please try again**."**
Post-Conditions: Program categories are accurately reflected in the system**.**

Table 18 Program Category Response

4.1.9.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Provide an interface for adding, editing, and deleting program categories.	High	Pending
REQ-2	Validate category names for uniqueness and correctness.	High	Pending
REQ-3	Update category details in the database.	High	Pending
REQ-4	Notify admin of successful updates or failures	Medium	Pending

Table 19 Program Category Requiements

4.2 Event Management

4.2.1 Add Event

4.2.1.1 Description

Parents sign up by providing personal details and creating login credentials. Once registered, they can log in to access information pertinent to their child's education.

4.2.1.2 Stimulus/Response Sequences

Use Case Name: Add Event
Description: Allows users to add events to the system calendar.
Actors: Admin, Teacher, Student
Goal: Maintain an updated event calendar.
Pre-Conditions: User is logged in and authorized.

Basic Course of Events/Main Flow:
• **Step 1:** User navigates to the "Add Event" page.
  • **Response**: System displays the event creation form.
• **Step 2**: User enters event details and submits the form.
  • **Response**: System validates and saves the event.
• **Step 3**: System confirms successful creation.

Alternate Flow (Invalid Inputs): • **Step:** User enters incomplete or invalid details.   • **Response**: System highlights errors and requests corrections.
Exception Flow (System Errors): • **Step:** Event creation fails due to server issues.   • **Response**: System displays "Unable to add event. Please try again later."
Post-Conditions: Event is successfully added**.**

Table 20 Add Event Response

4.2.1.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Provide an event creation form.	High	Pending
REQ-2	Validate event details before saving.	High	Pending
REQ-3	Save the event to the calendar.	High	Pending
REQ-4	Confirm successful creation of the event.	Medium	Pending

Table 21 Add Event Requirements

4.2.2 Edit Event

4.2.2.1 Description

Parents sign up by providing personal details and creating login credentials. Once registered, they can log in to access information pertinent to their child's education.

4.2.2.2 Stimulus/Response Sequences

Use Case Name: Edit Event
Description: Allows users to edit details of existing events.
Actors: Admin, Teacher, Student
Goal: Update event details to ensure accuracy.
Pre-Conditions: User is logged in and authorized to edit events.

Basic Course of Events/Main Flow:
• **Step 1:** User selects an existing event to edit.
  • **Response**: System displays the current event details in an editable form.
• **Step 2**: User modifies the event details (title, date, time, location) and submits.
  • **Response**: System validates and updates the event details.
• **Step 3**: System confirms successful modification and displays the updated event.dashboard if correct.

Alternate Flow (Invalid Inputs): • **Step:** User enters invalid or incomplete updates.   • **Response**: System highlights errors and requests corrections.
Exception Flow (System Errors): • **Step:** Update fails due to technical issues.   • **Response**: System displays "Unable to update event. Please try again later."
Post-Conditions: Event details are successfully updated.

Table 22 Edit Event Response

4.2.2.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Display current event details in an editable form.	High	Pending
REQ-2	Validate updated event details.	High	Pending
REQ-3	Save the updated event details to the system.	High	Pending
REQ-4	Confirm successful update to the user.	Medium	Pending

Table 23 Edit Event Requirements

4.2.3 Remove Event

4.2.3.1 Description

Post-signup, parents receive an email link to activate their account, ensuring that only verified users can access the system.

4.2.3.2 Stimulus/Response Sequences

Use Case Name: Remove Event
Description: Allows users to delete events from the system calendar.
Actors: Admin, Teacher, Student
Goal: Remove outdated or incorrect events.
Pre-Conditions: User is logged in and authorized to delete events.

Basic Course of Events/Main Flow:
• **Step 1:** User selects an event to delete.
  • **Response**: System prompts for confirmation.
• **Step 2**: User confirms the deletion.
  • **Response**: System deletes the event and notifies the user of successful removal.

Alternate Flow: • **Step:** User cancels the deletion process.   • **Response**: Event remains unchanged.
Exception Flow (System Errors): • **Step:** Deletion fails due to a technical issue.   • **Response**: System displays "Unable to delete event. Please try again later."
Post-Conditions: Event is successfully removed from the system calendar.

Table 24 Remove Event Response

4.2.3.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Provide an option to delete events with confirmation.	Medium	Pending
REQ-2	Successfully remove the event from the system.	High	Pending
REQ-3	Notify the user of successful or failed deletion.	High	Pending

Table 25 Remove Event Requirements

4.2.4 View Events

4.2.4.1 Description

If a parent forgets their password, they can reset it using their registered email, ensuring they can regain access to their account without hassle.

4.2.4.2 Stimulus/Response Sequences

Use Case Name: View Events
Description: Allows users to view all scheduled events in the system.
Actors: All Users (Admin, Teacher, Student, Parent)
Goal: Provide access to the event calendar.
Pre-Conditions: User is logged in.
Basic Course of Events/Main Flow: • **Step 1:** User navigates to the event calendar page.   • **Response**: System displays a list of all upcoming events.
Alternate Flow (Invalid Inputs): • **Step:** No events are scheduled.   • **Response**: System displays "No events found."
Exception Flow (System Errors): • **Step:** System fails to load events due to server issues.   • **Response**: System displays "Unable to load events. Please try again later."
Post-Conditions: User successfully views the list of events.

Table 26 View Events Response

4.2.4.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Display all scheduled events in a list or calendar.	High	Pending
REQ-2	Update event list dynamically with new entries.	High	Pending
REQ-3	Provide appropriate messaging when no events exist.	High	Pending
REQ-4	Handle errors when events fail to load.	Medium	Pending

Table 27 View Events Requirements

4.3 Timetable Management

4.3.1 Add Timetable

4.3.1.1 Description

Teachers register and log into the system to manage their teaching schedules, student interactions, and administrative tasks.

4.3.1.2 Stimulus/Response Sequences

Use Case Name: Add Timetable
Description: Allows the admin to create a new timetable.
Actors: Admin
Goal: Manage and organize class or event schedules.
Pre-Conditions: Admin is logged in and authorized.

Basic Course of Events/Main Flow:
• **Step 1:** Admin navigates to the "Add Timetable" page.
  • **Response**: System displays a form to input timetable details.
• **Step 2**: Admin enters timetable details (class, time, date, location) and submits.
  • **Response**: System validates and saves the timetable.
• **Step 3**: System confirms successful creation and displays the new timetable.

Alternate Flow (Invalid Inputs): • **Step:** Admin submits incomplete or invalid data. **Response**: System highlights errors and requests corrections.
Exception Flow (System Errors): • **Step:** Timetable creation fails due to technical issues. **Response**: System displays "Unable to save timetable. Please try again later."
Post-Conditions: Timetable is successfully created and saved in the system.

Table 28 Add Timetable Response

4.3.1.3 Functional Requirement

Req. ID	Description	Priority	Status
REQ-1	Provide a form for entering timetable details.	High	Pending
REQ-2	Validate timetable details before saving.	High	Pending
REQ-3	Save the timetable to the system database.	High	Pending
REQ-4	Confirm successful creation and display the timetable.	Medium	Pending

Table 29 Add TimeTable Requirements

4.3.2 Edit Timetable

4.3.2.1 Description

Teachers register and log into the system to manage their teaching schedules, student interactions, and administrative tasks.

4.3.2.2 Stimulus/Response Sequences

Use Case Name: Edit timetable
Description: Allows the admin to modify an existing timetable
Actors: Admin
Goal: Update timetable details to ensure accuracy and relevance
Pre-Conditions: Admin is logged in and authorized to edit timetables.
Basic Course of Events/Main Flow: • **Step 1:** Admin selects an existing timetable to edit. **Response**: System displays the current timetable details in an editable form. • **Step 2**: Admin updates the timetable details (class, time, date, location) and submits. **Response**: System validates and updates the timetable details. • **Step 3**: System confirms successful modification and displays the updated timetable.
Alternate Flow (Invalid Inputs): • **Step:** Admin enters invalid or incomplete updates. **Response**: System highlights errors and requests corrections.
Exception Flow (System Errors): • **Step:** Update fails due to technical issues. **Response**: System displays "Unable to update timetable. Please try again later."
Post-Conditions: Timetable details are successfully updated in the system.

Table 30 Edit TimeTable Response

4.3.2.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Display the current timetable in an editable format.	High	Pending
REQ-2	Validate updated timetable details.	High	Pending
REQ-3	Save updated timetable details to the database.	High	Pending
REQ-4	Confirm successful update to the user.	Medium	Pending

Table 31 Edit TimeTable Requirements

4.3.3 Remove Timetable

4.3.3.1 Description

Teachers activate their accounts via an emailed link, confirming their credentials and granting them access to their professional dashboard.

4.3.3.2 Stimulus/Response Sequences

Use Case Name: Remove Timetable
Description: Allows the admin to delete an existing timetable.
Actors: Admin
Goal: Remove outdated or unnecessary timetables from the system.
Pre-Conditions: Admin is logged in and authorized to delete timetables.
Basic Course of Events/Main Flow: • **Step 1:** Admin selects a timetable to delete. **Response**: System prompts for confirmation. • **Step 2**: Admin confirms the deletion. **Response**: System deletes the timetable and notifies the admin.
Alternate Flow: • **Step:** Admin cancels the deletion process.   • **Response**: Timetable remains unchanged.
Exception Flow (System Errors): • **Step:** Deletion fails due to technical issues.   • **Response**: System displays "Unable to delete timetable. Please try again later."
Post-Conditions: Timetable is successfully removed from the system.

Table 32 Remove Timetable Response

4.3.3.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Provide an option to select and delete timetables.	High	Pending
REQ-2	Display confirmation before final deletion.	High	Pending
REQ-3	Successfully delete the timetable from the database.	High	Pending
REQ-4	Notify the admin of successful or failed deletion.	Medium	Pending

Table 33 Remove Timetable Requirements

4.3.4 View Timetable

4.3.4.1 Description

This provides teachers with the option to reset their passwords if forgotten, maintaining secure access to their accounts.

4.3.4.2 Stimulus/Response Sequences

Use Case Name: View Timetable
Description: Allows users to view the current timetable.
Actors: All Users (Admin, Teacher, Student, Parent)
Goal: Provide easy access to scheduled classes and events
Pre-Conditions: Timetable data is available in the system.
Basic Course of Events/Main Flow: • **Step 1:** User navigates to the "View Timetable" page.   • **Response**: System displays the current timetable.
Alternate Flow (Invalid Email): • **Step:** No timetable data is available.   • **Response**: System displays "No timetable available at this time."
Exception Flow (System Errors): • **Step:** System fails to load the timetable.   • **Response**: System displays "Unable to load timetable. Please try again later."
Post-Conditions: User successfully views the timetable.

Table 34 View TimeTable Response

4.3.4.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Display the current timetable in an organized format.	High	Pending
REQ-2	Ensure timetable data is dynamically updated.	High	Pending
REQ-3	Display appropriate messages when no data is found.	High	Pending
REQ-4	Handle errors when timetable fails to load.	Medium	Pending

Table 35 View TimeTable Requirements

4.4 Message Management

4.4.1 Send Messages

4.4.1.1 Description

Teachers register and log into the system to manage their teaching schedules, student interactions, and administrative tasks.

4.4.1.2 Stimulus/Response Sequences

Use Case Name: Send messages
Description: Allows users to send messages to other users on the platform.
Actors: Teacher, Student
Goal: Facilitate communication between users.
Pre-Conditions: User is logged in.

Basic Course of Events/Main Flow:
• **Step 1:** User navigates to the "Send Message" page.
  • **Response**: System displays a message form with recipient and content fields.
• **Step 2**: User enters the recipient and message content, then submits the form.
  • **Response**: System validates the message and sends it to the recipient.
• **Step 3**: System confirms the successful delivery of the message.

Alternate Flow (Invalid Inputs): • **Step:** User submits the message form with incomplete or invalid information.   • **Response**: System displays error messages and requests corrections.
Exception Flow (System Errors): • **Step:** Message sending fails due to technical or server issues. **Response**: System displays "Unable to send message. Please try again later."
Post-Conditions: Message is successfully delivered to the recipient.

Table 36 Send Message Response

4.4.1.3 Functional Requirement

Req. ID	Description	Priority	Status
REQ-1	Provide a form to input recipient and message content.	High	Pending
REQ-2	Validate message details before sending.	High	Pending
REQ-3	Deliver the message to the specified recipient.	High	Pending
REQ-4	Confirm successful message delivery to the sender.	Medium	Pending

Table 37 Send Message Requirements

4.4.2 Receive Messages

4.4.2.1 Description

Teachers register and log into the system to manage their teaching schedules, student interactions, and administrative tasks.

4.4.2.2 Stimulus/Response Sequences

Use Case Name: Receive messages
Description: Allows users to view and manage messages they have received.
Actors: Teacher, Student
Goal: Provide users with access to received messages.
Pre-Conditions: User is logged in and has messages in their inbox.

Basic Course of Events/Main Flow:
• **Step 1:** User navigates to the "Inbox" page.
  • **Response**: System displays a list of all received messages.
• **Step 2**: User selects a specific message to view its content.
  • **Response**: System displays the full content of the selected message.

Alternate Flow (Invalid Inputs): • **Step:** User's inbox is empty.   • **Response**: System displays "No new messages available."
Exception Flow (System Errors): • **Step:** System fails to load messages due to technical issues.   • **Response**: System displays "Unable to load messages. Please try again later."
Post-Conditions: User successfully views received messages.

Table 38 Recieve Message Response

4.4.2.3 Functional Requirement

Req. ID	Description	Priority	Status
REQ-1	Display all received messages in a user-friendly list.	High	Pending
REQ-2	Allow users to view the full content of a message.	High	Pending
REQ-3	Provide appropriate notifications for no new messages.	Medium	Pending
REQ-4	Handle errors when messages fail to load.	Medium	Pending

Table 39 Send Message Requirements

4.4.3 Approve/Reject Message Requests

4.4.3.1 Description

Teachers register and log into the system to manage their teaching schedules, student interactions, and administrative tasks.

4.4.3.2 Stimulus/Response Sequences

Use Case Name: Approve/Reject Messages Request
Description: Allows teachers to approve or reject incoming message requests from students.
Actors: Teacher
Goal: Provide control over communication permissions..
Pre-Conditions: Teacher is logged in and has pending message requests.

Basic Course of Events/Main Flow:
• **Step 1:** Teacher navigates to the "Message Requests" page.
  • **Response**: System displays a list of pending message requests.
• **Step 2**: Teacher selects a request and chooses to approve or reject it.
  • **Response**: System updates the status of the request based on the teacher's action.

Alternate Flow (Invalid Inputs): • **Step:** Teacher chooses not to act on any requests.   • **Response**: System leaves message requests unchanged.
Exception Flow (System Errors): • **Step:** Approval or rejection action fails due to technical issues.   • **Response**: System displays "Unable to process request. Please try again later."
Post-Conditions: Message request is approved or rejected successfully.

Table 40 Approve/Reject Message Request Response

4.4.3.3 Functional Requirement

Req. ID	Description	Priority	Status
REQ-1	Display all pending message requests to the teacher.	High	Pending
REQ-2	Allow teachers to approve or reject a request.	High	Pending
REQ-3	Update the request status based on the teacher's action.	High	Pending
REQ-4	Notify the requester of the approval or rejection status.	Medium	Pending

Table 41 Approve/Reject Message Request Requirements

4.5 Job Management

4.5.1 Post Job

4.5.1.1 Description

Teachers register and log into the system to manage their teaching schedules, student interactions, and administrative tasks.

4.5.1.2 Stimulus/Response Sequences

Use Case Name: Post Job
Description: Allows users to post job openings on the platform.
Actors: Admin, Teacher
Goal: Provide job opportunities for students or other users.
Pre-Conditions: User is logged in and authorized to post jobs.

Basic Course of Events/Main Flow:
• **Step 1:** User navigates to the "Post Job" page.
  • **Response**: System displays a job posting form with fields like title, description, category, and location.
• **Step 2**: User enters job details and submits the form.
  • **Response**: System validates the input and publishes the job posting.
• **Step 3**: System confirms successful job posting and displays the job in the job board.

Alternate Flow (Invalid Inputs): • **Step:** User submits incomplete or invalid details.   • **Response**: System highlights errors and requests corrections.
Exception Flow (System Errors): • **Step:** Job posting fails due to technical or server issues.   • **Response**: System displays "Unable to post job. Please try again later."
Post-Conditions: Job is successfully posted to the system and visible on the job board.

Table 42 Post Job Response

4.5.1.3 Functional Requirement

Req. ID	Description	Priority	Status
REQ-1	Provide a job posting with a relevant field.	High	Pending
REQ-2	Validate job details before publishing.	High	Pending
REQ-3	Publish the job posting to the job board.	High	Pending
REQ-4	Confirm successful job posting to the user.	Medium	Pending

Table 43 Post Job Requirements

4.5.2 Edit Job

4.5.2.1 Description

Teachers register and log into the system to manage their teaching schedules, student interactions, and administrative tasks.

4.5.2.2 Stimulus/Response Sequences

Use Case Name: Edit Job
Description: Allows users to modify details of existing job postings.
Actors: Admin, Teacher
Goal: Update job postings to ensure accuracy and relevance.
Pre-Conditions: User is logged in and authorized to edit jobs.

Basic Course of Events/Main Flow:
• **Step 1:** User selects an existing job posting to edit.
  • **Response**: System displays the current job details in an editable form.
• **Step 2**: User modifies the job details (title, description, location) and submits the updates.
  • **Response**: System validates and updates the job posting.
• **Step 3**: System confirms successful modification and updates the job on the job board.

Alternate Flow (Invalid Inputs): • **Step:** User submits invalid or incomplete updates. **Response**: System highlights errors and requests corrections.
Exception Flow (System Errors): • **Step:** Job update fails due to technical issues. **Response**: System displays "Unable to update job posting. Please try again later."
Post-Conditions: Job details are successfully updated and visible on the job board.

Table 44 Edit Job Response

4.5.2.3 Functional Requirement

Req. ID	Description	Priority	Status
REQ-1	Display existing job details in an editable form.	High	Pending
REQ-2	Validate updated job details before saving.	High	Pending
REQ-3	Save updated job details and display the changes.	High	Pending
REQ-4	Confirm successful job modification to the user.	Medium	Pending

Table 45 Edit Job Requirements

4.5.3 Remove Job

4.5.3.1 Description

Teachers register and log into the system to manage their teaching schedules, student interactions, and administrative tasks.

4.5.3.2 Stimulus/Response Sequences

Use Case Name: Remove Job
Description: Allows users to delete job postings from the system.
Actors: Admin, Teacher
Goal: Remove outdated or irrelevant job postings.
Pre-Conditions: User is logged in and authorized to delete jobs.

Basic Course of Events/Main Flow:
• **Step 1:** User selects a job posting to delete.
  • **Response**: System prompts for confirmation.
• **Step 2**: User confirms the deletion.
  • **Response**: System deletes the job posting from the system.
• **Step 3**: System confirms successful deletion and removes the job from the job board.

Alternate Flow (Invalid Inputs): • **Step:** User cancels the deletion process.   • **Response**: Job posting remains unchanged.
Exception Flow (System Errors): • **Step:** Deletion fails due to technical or server issues.   • **Response**: System displays "Unable to delete job posting. Please try again later."
Post-Conditions: Job posting is successfully removed from the system.

Table 46 Remove Job Response

4.5.3.3 Functional Requirement

Req. ID	Description	Priority	Status
REQ-1	Provide an option to select and delete job postings.	Medium	Pending
REQ-2	Display a confirmation message before deletion.	High	Pending
REQ-3	Successfully delete the job posting from the system.	High	Pending
REQ-4	Confirm successful deletion to the user.	Medium	Pending

Table 47 Remove Job Requirements

4.5.4 Manage Job Categories

4.5.4.1 Description

Teachers register and log into the system to manage their teaching schedules, student interactions, and administrative tasks.

4.5.4.2 Stimulus/Response Sequences

Use Case Name: Manage Job Categories
Description: Allows the admin to create, edit, or delete job categories for organizing job postings
Actors: Admin
Goal: Organize jobs effectively by categorizing postings.
Pre-Conditions: Admin is logged in and authorized to manage job categories.

Basic Course of Events/Main Flow:
• **Step 1:** Admin navigates to the "Manage Job Categories" page.
  • **Response**: System displays the list of current job categories.
• **Step 2**: Admin chooses to add, edit, or delete a category.
  • **Response**: System performs the respective operation and updates the list.
• **Step 3**: System confirms the successful operation.

Alternate Flow (Invalid Inputs): • **Step:** Admin enters invalid or duplicate category names.   • **Response**: System displays an error and requests corrections.
Exception Flow (System Errors): • **Step:** System fails to update categories due to technical issues.   • **Response**: System displays "Unable to update categories. Please try again later."
Post-Conditions: Job categories are updated successfully.

Table 48 Manage Job Categories Response

4.5.4.3 Functional Requirement

Req. ID	Description	Priority	Status
REQ-1	Display all current job categories.	High	Pending
REQ-2	Provide an option to add, edit, or delete categories.	High	Pending
REQ-3	Validate category names to prevent duplicates.	Medium	Pending
REQ-4	Confirm successful updates to job categories.	Medium	Pending

Table 49 Manage Job Categories Requirements

4.6 Marketplace Management

4.6.1 Add Marketplace Item

4.6.1.1 Description

Students log into the system to sell their items by listing them on the marketplace.

4.6.1.2 Stimulus/Response Sequences

Use Case Name: Add Marketplace Item
Description: Allows students to upload items for sale on the UniVersa marketplace.
Actors: Student
Goal: Enable students to sell personal items through the app's marketplace.
Pre-Conditions: User is logged in and authorized to post on the marketplace.

Basic Course of Events/Main Flow:
• **Step 1:** User navigates to the "Add Item" page.
  • **Response:** System displays a form with fields for title, description, price, and image upload.
• **Step 2:** User fills in item details and submits the form.
  • **Response:** System validates inputs and uploads the listing.
• **Step 3:** System confirms successful item listing.
  • **Response:** Item becomes visible on the marketplace.

Alternate Flow (Invalid Inputs): • **Step:** User provides missing or invalid details.   • **Response:** System displays appropriate error messages and requests corrections.
Exception Flow (System Errors): • **Step:** Upload fails due to technical issues.   • **Response:** System displays "Unable to list item. Please try again later."
Post-Conditions: Item is listed and available for other users to view and purchase.

Table 50 Add Item Response

4.6.1.3 Functional Requirement

Req. ID	Description	Priority	Status
REQ-1	Provide a form to input item details and upload images.	High	Pending
REQ-2	Validate user inputs and image formats.	High	Pending
REQ-3	Save item details to the database.	Medium	Pending
REQ-4	Confirm successful listing and notify the user.	Medium	Pending

Table 51 Add Item Requirements

4.6.2 Update Marketplace Item

4.6.2.1 Description

Students can manage and edit details of items listed on the marketplace.

4.6.2.2 Stimulus/Response Sequences

Use Case Name: Update Marketplace Item
Description**:** Allows users to modify existing marketplace listings.

Actors: Student
Goal**:** Ensure listings remain accurate and up to date.
Pre-**Conditions:** User is logged in and owns the item listing.

Basic Course of Events/Main Flow:
 	
• **Step 1:** User selects an item from their listings.
  • **Response:** System displays item details in editable format.
• **Step 2:** User edits details (e.g., price, description) and submits.
  • **Response:** System validates new inputs and updates listing.
• **Step 3:** System confirms successful update.
  • **Response:** Updated listing is now visible in the marketplace.

Alternate Flow (Invalid Inputs): • **Step:** User enters invalid data.   • **Response:** System highlights errors and prompts for correction.
Exception Flow (System Errors): • **Step:** Update fails due to system issues.   • **Response:** System displays "Unable to update item. Please try again later."
Post-Conditions: Item details are successfully updated on the marketplace.

Table 52 Update Item Response

4.6.2.3 Functional Requirement

Req. ID	Description	Priority	Status
REQ-1	Display editable item form.	High	Pending
REQ-2	Allow editing of all relevant fields.	High	Pending
REQ-3	Validate updated item data.	High	Pending
REQ-4	Save updated details and confirm success.	Medium	Pending

Table 53 Update Response Requirements

4.6.3 Delete Marketplace Item

4.6.3.1 Description

Students can remove items from the marketplace when no longer available.

4.6.3.2 Stimulus/Response Sequences

Use Case Name: Delete Marketplace Item
Description: Enables users to remove their posted items.
Actors: Student

Goal: Maintain an up-to-date and relevant marketplace.
Pre-Conditions: User is logged in and authorized to delete their own listings.

Basic Course of Events/Main Flow:
• **Step 1:** User selects an item to delete.
  • **Response:** System prompts for confirmation.
• **Step 2:** User confirms deletion.
  • **Response:** System deletes item and confirms removal.

Alternate Flow (User Cancels): • **Step:** User cancels the deletion.   • **Response:** Item remains unchanged.
Exception Flow (System Errors): • **Step:** Deletion fails due to technical issues.   • **Response:** System displays "Unable to delete item. Please try again later."
Post-Conditions: Item is removed from the marketplace.

Table 54 Delete Item Response

4.6.3.3 Functional Requirement

Req. ID	Description	Priority	Status
REQ-1	Allow users to select and delete items.	High	Pending
REQ-2	Display confirmation prompt.	High	Pending
REQ-3	Successfully delete the item from the database.	High	Pending
REQ-4	Notify the user of the outcome.	Medium	Pending

Table 55 Delete Item Requirements

4.7 Group Management

4.7.1 Joining Groups

4.7.1.1 Description

Teachers register and log into the system to manage their teaching schedules, student interactions, and administrative tasks.

4.7.1.2 Stimulus/Response Sequences

Use Case Name: Join Groups
Description: Allows users to join available groups within the system.

Actors: Student
Goal: Enable users to participate in relevant groups for collaboration and communication.
Pre-Conditions: User is logged in and authorized to join groups.

Basic Course of Events/Main Flow:
 	
• **Step 1:** User navigates to the "Groups" page.
  • **Response**: System displays a list of all available groups with their descriptions.
• **Step 2**: User selects a group they want to join and clicks the "Join" button.
  • **Response**: System validates the request and adds the user to the group.
• **Step 3**: System confirms successful group membership and updates the user's group list.

Alternate Flow (Invalid Inputs): • **Step:** User attempts to join a group they are already a member of.   • **Response**: System displays "You are already a member of this group."
Exception Flow (System Errors): • **Step:** Group joining fails due to technical issues.   • **Response**: System displays "Unable to join group. Please try again later."
Post-Conditions: User is successfully added to the selected group.

Table 56 Join Group Response

4.7.1.3 Functional Requirement

Req. ID	Description	Priority	Status
REQ-1	Display a list of all available groups to the user.	High	Pending
REQ-2	Display a list of all available groups to the user.	High	Pending
REQ-3	Prevent users from joining the same group multiple times.	Medium	Pending
REQ-4	Confirm successful group membership to the user.	Medium	Pending
REQ-5	Handle errors when group joining fails due to technical issues.	Medium	Pending

Table 57 Join Group Requirements

4.7.2 Leaving Groups

4.7.2.1 Description

Teachers register and log into the system to manage their teaching schedules, student interactions, and administrative tasks.

4.7.2.2 Stimulus/Response Sequences

Use Case Name: Leaving Groups
Description: Allows users to leave a group they are currently a member of.

Actors: Student
Goal: Enable users to manage their group memberships by leaving groups they no longer wish to participate in.
Pre-Conditions: User is logged in and a member of the group they want to leave.

Basic Course of Events/Main Flow:
 	
• **Step 1:** User navigates to the "My Groups" page.
  • **Response**: System displays a list of groups the user is currently a member of.
• **Step 2**: User selects a group they want to leave and clicks the "Leave" button.
  • **Response**: System prompts the user for confirmation.
• **Step 3**: User confirms the action.
  • **Response**: System removes the user from the group and updates the membership list.

Alternate Flow (Invalid Inputs): • **Step:** User cancels the leave group action.   • **Response**: User remains a member of the group.
Exception Flow (System Errors): • **Step:** Group leaving fails due to technical issues. • **Response**: System displays "Unable to leave group. Please try again later."
Post-Conditions: User is successfully removed from the selected group.

Table 58 Leaving Group Response

4.7.2.3 Functional Requirement

Req. ID	Description	Priority	Status
REQ-1	Display a list of groups the user is currently a member of.	High	Pending
REQ-2	Provide an option to leave a selected group.	High	Pending
REQ-3	Prompt the user for confirmation before leaving a group.	Medium	Pending
REQ-4	Successfully remove the user from the group's membership list.	Medium	Pending
REQ-5	Notify the user of successful or failed group leaving action.	Medium	Pending

Table 59 Leaving Group Requirements

4.8 Real-Time Notifications Management

4.8.1 Receive Real-Time Notifications

4.8.1.1 Description

Teachers register and log into the system to manage their teaching schedules, student interactions, and administrative tasks.

4.8.1.2 Stimulus/Response Sequences

Use Case Name: Receiving Real-Time Notifications
Description: Allows users to receive real-time notifications for system events, updates, or communications.
Actors: All Users (Admin, Teacher, Student, Parent)
Goal: Notify users promptly about relevant activities and updates.
Pre-Conditions: User is logged in and has notifications enabled.

Basic Course of Events/Main Flow:
• **Step 1:** A triggering event occurs (e.g., a new message, job posting, or event update).
  • **Response**: System generates a notification based on the event type.
• **Step 2**: System delivers the notification to the user in real-time.
  • **Response**: Notification appears in the user's notification center.
• **Step 3**: User clicks on the notification to view more details.
  • **Response**: System redirects the user to the relevant page or activity.

Alternate Flow (Invalid Inputs): • **Step:** User has disabled notifications or is offline.   • **Response**: System stores the notification and displays it when the user logs in or enables notifications.
Exception Flow (System Errors): • **Step:** Notification delivery fails due to technical issues.   • **Response**: System logs the error and attempts redelivery.
Post-Conditions: Notification is successfully delivered and accessed by the user.

Table 60 Notification Response

4.8.1.3 Functional Requirement

Req. ID	Description	Priority	Status
REQ-1	Generate notifications for all relevant system events.	High	Pending
REQ-2	Deliver notifications to users in real-time.	High	Pending
REQ-3	Store notifications for offline users and display them later.	Medium	Pending

REQ-4	Provide a user-friendly notification center for viewing updates.	High	Pending
REQ-5	Handle errors in notification delivery and log them for redelivery.	Medium	Pending

Table 61 Notification Requirements

4.9 Lost and Found Management

4.9.1 Add Lost Item

4.9.1.1 Description

Teachers register and log into the system to manage their teaching schedules, student interactions, and administrative tasks.

4.9.1.2 Stimulus/Response Sequences

Use Case Name: Add Lost Item
Description: Allows users to report a lost item to the system.
Actors: Student, Teacher Admin
Goal: Track and manage lost items effectively.
Pre-Conditions: User is logged in and authorized to report lost items.

Basic Course of Events/Main Flow:
• **Step 1:** User navigates to the "Add Lost Item" page.
  • **Response**: System displays a form to input details of the lost item, including description, date, location, and contact information.
• **Step 2**: User enters the required information and submits the form.
  • **Response**: System validates the input and saves the lost item details to the database.
• **Step 3**: System confirms successful addition of the lost item and displays it in the lost and found list.

Alternate Flow (Invalid Inputs): • **Step:** User submits incomplete or invalid details.   • **Response**: System highlights errors and requests corrections.
Exception Flow (System Errors): • **Step:** Addition of lost item fails due to technical issues.   • **Response**: System displays "Unable to add lost item. Please try again later."
Post-Conditions: Lost item is successfully added to the system and available in the lost and found list.

Table 62 Add Lost Item Response

4.9.1.3 Functional Requirement

Req. ID	Description	Priority	Status
REQ-1	Provide a form to input lost item details.	High	Pending
REQ-2	Validate user input for the lost item details.	High	Pending
REQ-3	Save the lost item details to the system database.	High	Pending
REQ-4	Confirm successful addition of the lost item to the user.	Medium	Pending

Table 63 Add Lost Item Requirements

4.9.2 Remove Lost Item

4.9.2.1 Description

Teachers register and log into the system to manage their teaching schedules, student interactions, and administrative tasks.

4.9.2.2 Stimulus/Response Sequences

Use Case Name: Remove Lost Item
Description: Allows users to remove a previously reported lost item.
Actors: Student, Teacher, Admin
Goal: Maintain an up-to-date list of lost items.
Pre-Conditions: User is logged in and authorized to manage lost items.

Basic Course of Events/Main Flow:
• **Step 1:** User selects a lost item to remove from the list.
  • **Response**: System prompts for confirmation before deletion.
• **Step 2**: User confirms the deletion of the lost item.
  • **Response**: System removes the lost item from the database and updates the lost and found list.
• **Step 3**: System confirms successful deletion.

Alternate Flow (Invalid Inputs): • **Step:** User cancels the deletion process.   • **Response**: Lost item remains unchanged in the system.
Exception Flow (System Errors): • **Step:** Removal of the lost item fails due to technical issues.   • **Response**: System displays “Unable to remove lost item. Please try again later.”
Post-Conditions: Lost item is successfully removed from the lost and found list.

Table 64 Remove Lost Item Response

4.9.2.3 Functional Requirement

Req. ID	Description	Priority	Status
REQ-1	Provide an option to select and delete a lost item.	High	Pending
REQ-2	Display a confirmation prompt before deletion.	High	Pending
REQ-3	Successfully remove the lost item from the database.	Medium	Pending
REQ-4	Notify the user of successful or failed removal.	Medium	Pending

Table 65 Remove Lost Item Requirements

4.10 Resource Management

4.10.1 Add Resource

4.10.1.1 Description

Students can add new digital resources (e.g., rooms, equipment, books) into the system for shared usage and management.

4.10.1.2 Stimulus/Response Sequences

Use Case Name: Add Resource
Description: Allows users to input a new resource into the system.
Actors: Student
Goal: Efficiently manage available institutional resources.
Pre-Conditions: User is logged in and authorized to add resources.

Basic Course of Events/Main Flow:
• **Step 1:** User navigates to the "Add Resource" page.
  • **Response:** System displays a form to enter resource name, type (room, book, etc.), description, quantity, and availability.
• **Step 2:** User fills out the form and submits.
  • **Response:** System validates the inputs and stores the resource in the database.
• **Step 3:** System confirms successful addition.
  • **Response:** The new resource appears in the resource management list.

Alternate Flow (Invalid Inputs): • **Step:** User enters missing or invalid fields.   • **Response**: System highlights errors and requests corrections.
Exception Flow (System Errors): • **Step:** Database connection or backend issue occurs.   • **Response**: "Unable to add resource. Please try again later."
Post-Conditions: The resource is added and available for allocation or usage tracking.

Table 66 Add Resource Response

4.10.1.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Provide a form input recourse details.	High	Pending
REQ-2	Validate the user input for the resourse form.	High	Pending
REQ-3	Save the resource data to the system database.	High	Pending
REQ-4	Confirm successful addition of the resource to the user.	Medium	Pending

Table 67 Add Resource Requirements

4.10.2 Remove Resource

4.10.2.1 Description

Students can remove outdated or unavailable resources from the system.

4.10.2.2 Stimulus/Response Sequences

Use Case Name: Remove Resource
Description: Enables deletion of existing resources.
Actors: Admin
Goal: Maintain an up-to-date and accurate resource list.
Pre-Conditions: User is logged in and authorized to delete.

Basic Course of Events/Main Flow:
• **Step 1:** User navigates to the resource list and selects a resource to delete.
  ○ **Response:** System prompts a confirmation dialog: "Are you sure?"
• **Step 2:** User confirms deletion.
  ○ **Response:** Resource is removed from the database and list is updated.

Alternate Flow (User Cancels): • **Step:** User chooses not to proceed.   ○ **Response:** No changes are made.
Exception Flow (System Errors): • **Step:** Deletion fails due to a backend or connectivity issue.   ○ **Response:** "Resource removal failed. Try again later."
Post-Conditions: The selected resource is permanently removed from the system.

Table 68 Remove Resource Response

4.10.2.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Display a list of existing resources.	High	Pending
REQ-2	Allow the user to select and delete a resource.	High	Pending
REQ-3	Prompt the user for confirmation before deletion.	High	Pending

REQ-4	Remove the resource and display a success message.	Medium	Pending

Table 69 Remove Resource Requirements

4.11 To-Do List

4.11.1 Create To-Do

4.11.1.1 Description

Students and teachers can create personal or academic tasks in their to-do list for better time and activity management.

4.11.1.2 Stimulus/Response Sequences

Use Case Name: Create To-do
Description: Allows users to create a new to-do item in their task list.
Actors: Student
Goal: Help users manage and track their tasks efficiently.
Pre-Conditions: User is logged in and has access to the to-do list feature.

Basic Course of Events/Main Flow:
• **Step 1:** User navigates to the "To-Do List" section and clicks the "Add Task" or "+" button.
  • **Response:** System displays a form with fields for task title, description, due date/time, and priority level.
• **Step 2:** User fills in the required fields and clicks "Save."
  • **Response:** System validates the inputs and saves the task to the database.
• **Step 3:** System confirms that the task was added successfully.
  • **Response:** New tasks are displayed in the user's to-do list.

Alternate Flow (Invalid Inputs): • **Step:** User enters incomplete or invalid data (e.g., missing title or invalid date).   • **Response:** System highlights missing or incorrect fields and prompts for corrections.
Exception Flow (System Errors): • **Step:** Task creation fails due to a technical issue (e.g., database error).   • **Response:** System displays a message: "Unable to create task. Please try again later."
Post-Conditions: The task is successfully saved and displayed in the user's active to-do list with appropriate details.

Table 70 Create To-Do Response

4.11.1.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Provide a form to input a new task.	High	Pending
REQ-2	Validate task title, description, and due date fields.	High	Pending
REQ-3	Store the task in the user to-do list.	High	Pending
REQ-4	Display confirmation after successful task creation.	Medium	Pending

Table 71 Create To-Do Requirements

4.11.2 Edit To-Do

4.11.2.1 Description

Users can edit existing tasks to update the title, description, deadline, or priority.

4.11.2.2 Stimulus/Response Sequences

Use Case Name: Edit To-do

Description: Allows users to modify an existing to-do item.

Actors: Student

Goal: Keep to-do tasks current and accurate.

Pre-Conditions: The to-do item exists and user is logged in.

Basic Course of Events/Main Flow:

- **Step 1:** User navigates to the to-do list and selects a task to edit.
  - **Response:** System displays the task details in editable form fields.
- **Step 2:** User modifies the desired fields (e.g., updates the deadline, changes priority) and clicks "Update."
  - **Response:** System validates and updates the record in the database.
- **Step 3:** System confirms that the task was successfully updated.
  - **Response:** Updated task appears in the list with the new details.

Alternate Flow (Invalid Inputs):

- **Step:** User provides an invalid format (e.g., invalid date).
  - **Response:** System prompts user to correct the data before submission.

Exception Flow (System Errors):

- **Step:** Task update fails due to technical issues.
  - **Response:** System shows "Unable to update task. Try again later."

Post-Conditions: Task is successfully updated and displayed with new information.

Table 72 Edit To-Do Response

4.11.2.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Allow users to select and edit an existing task.	High	Pending
REQ-2	Validate updated task fields.	High	Pending
REQ-3	Update task in the system database.	High	Pending
REQ-4	Confirm the successful update of the task.	Medium	Pending

Table 73 Edit To-Do Requirements

4.11.3 Delete To-Do

4.11.3.1 Description

Users can delete a task from their to-do list when it is no longer relevant.

4.11.3.2 Stimulus/Response Sequences

Use Case Name: Delete To-do
Description: Enable users to delete an existing task from their to-do list.
Actors: Student
Goal: Keep the task clear and organized.
Pre-Conditions: User is logged in and the to-do item exists.

Basic Course of Events/Main Flow:
• **Step 1:** User goes to the to-do list and clicks the "Delete" icon or button next to the task.
  • **Response:** System displays a confirmation dialog: "Are you sure you want to delete this task?"
• **Step 2:** User confirms deletion.
  • **Response:** Task is removed from the database and no longer shown in the list.

Alternate Flow (Cancelled Action): • **Step:** User cancels deletion.   • **Response:** System closes the dialog and no changes are made.
Exception Flow (System Errors): • **Step:** Task deletion fails due to technical issues.   • **Response:** System displays "Unable to delete task. Please try again later."
Post-Conditions: Task is permanently removed from the to-do list.

Table 74 Delete To-Do Response

4.11.3.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Display a list of tasks to the user .	High	Pending
REQ-2	Allow the user to delete a selected task.	High	Pending

REQ-3	Prompt the user for confirmation before deletion.	High	Pending
REQ-4	Remove task and confirm success to the user.	Medium	Pending

Table 75 Delete To-Do Requirements

4.12 Carpool

4.12.1 Add Carpool

4.12.1.1 Description

Users can add a carpool ride offer to help others join for shared transportation, promoting collaboration and saving costs.

4.12.1.2 Stimulus/Response Sequences

Use Case Name: Add Carpool
Description: Allows users to post a carpool ride by entering details such as route, time, vehicle info, and available seats.
Actors: Student
Goal: Promote transport-sharing and efficient commute planning.
Pre-Conditions: User is logged in and allowed to offer rides.

Basic Course of Events/Main Flow:
• **Step 1:** User navigates to the "Carpool" section and selects "Add Ride."
  • **Response:** System displays a form to input ride details (pickup point, drop-off, time, vehicle details, and number of seats).
• **Step 2:** User fills in all the ride information and submits the form.
  • **Response:** System validates input and adds the carpool offer to the database.
• **Step 3:** System confirms successful creation.
  • **Response:** New carpool ride is displayed in the list of available carpools.

Alternate Flow (Invalid Inputs): • **Step:** User submits incomplete or incorrect information.   • **Response:** System highlights issues and prompts for correction.
Exception Flow (System Errors): • **Step:** System fails to add carpool due to an internal error.   • **Response:** "Unable to create carpool ride. Please try again later."
Post-Conditions: The carpool ride is visible to others and open for join requests.

Table 76 Add Carpool Response

4.12.1.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Provide a form to input carpool ride details.	High	Pending

REQ-2	Validate carpool input fields such as route, time, and seats.	High	Pending
REQ-3	Store the carpool ride in the database.	High	Pending
REQ-4	Display confirmation of the ride creation	Medium	Pending

Table 77 Add Carpool Requirements

4.12.2 Edit Carpool

4.12.2.1 Description

Users can edit ride details such as time, route, or number of seats for an existing carpool offer.

4.12.2.2 Stimulus/Response Sequences

Use Case Name: Edit Carpool
Description: Allows users to update the information of a previously added carpool ride.
Actors: Student
Goal: Keep ride details accurate and up-to date.
Pre-Conditions: User is logged in and is the user of the ride.

Basic Course of Events/Main Flow:
• **Step 1:** User navigates to "My Rides" and selects "Edit" on a carpool ride.
  • **Response:** System shows an editable form populated with existing ride details.
• **Step 2:** User modifies the fields (e.g., pickup time or seats) and clicks "Update."
  • **Response:** System validates input and updates the record in the database.
• **Step 3:** Confirmation is displayed.
  • **Response:** Ride information is updated in the list.

Alternate Flow: • **Step:** User submits invalid or incomplete input.   • **Response:** System displays errors and requests corrections.
Exception Flow (System Errors): • **Step:** System error during update.   • **Response:** "Failed to update carpool ride. Try again later."
Post-Conditions: Updated ride details are saved and reflected in the carpool listing.

Table 78 Edit Carpool Response

4.12.2.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Allow users to access and edit carpool details.	High	Pending
REQ-2	Validate the updated ride fields.	High	Pending
REQ-3	Update carpool ride in the database.	High	Pending
REQ-4	Confirm the successful update of the table	Medium	Pending

Table 79 Edit Carpool Requirements

4.12.3 Delete Carpool

4.12.3.1 Description

Users can remove their previously created carpool ride offers if the ride is no longer available.

4.12.3.2 Stimulus/Response Sequences

Use Case Name: Delete Carpool
Description: Enable a user to delete a posted carpool ride.
Actors: Student
Goal: Allow ride owners to cancel rides and keep listing updated.
Pre-Conditions: User is logged in and owns the carpool ride.

Basic Course of Events/Main Flow:
• **Step 1:** User navigates to their created carpool rides and selects the "Delete" or "Remove" option.
  • **Response:** System prompts for confirmation: "Are you sure you want to remove this ride?"
• **Step 2:** User confirms the deletion.
  • **Response:** System deletes the ride from the database and updates the carpool list.

Alternate Flow: • **Step:** User cancels the deletion prompt.   • **Response:** No changes are made.
Exception Flow (System Errors): • **Step:** Deletion fails due to system/database error.   • **Response:** "Unable to remove carpool. Please try again later."
Post-Conditions: The ride is no longer visible or accessible in the carpool ride.

Table 80 Delete Carpool Response

4.12.3.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Show a list of posted carpool rides	High	Pending
REQ-2	Allow deletions of a selected ride.	High	Pending
REQ-3	Prompt user for confirmation.	High	Pending
REQ-4	Delete ride and show success message	Medium	Pending

Table 81 Delete Carpool Requiements

4.12.4 Send Location

4.12.4.1 Description

Users can send their current location to passengers in the carpool ride for coordination.

4.12.4.2 Stimulus/Response Sequences

Use Case Name: Send Location
Description: Allows users to send their GPS-based location to other carpool members.
Actors: Student
Goal: Facilitate accurate pickup coordination.
Pre-Conditions: Location services enabled, user is logged in, and has an active ride..

Basic Course of Events/Main Flow:
• **Step 1:** User opens their carpool ride and taps "Send Location."
  • **Response:** System fetches the current location via GPS.
• **Step 2:** User confirms the location to be sent.
  • **Response:** Location is shared with the ride participants via notification or message.

Alternate Flow (Location Not Enabled): • **Step:** GPS is turned off or permission is denied.   • **Response:** System prompts user to enable location services.
Exception Flow (Technical Failure): • **Step:** Location retrieval or sharing fails due to network or system issues.   • **Response:** "Unable to send location. Please check your connection or try again later."
Post-Conditions: Location is successfully sent to all selected participants.

Table 82 Send Location Response

4.12.4.3 Functional Requirements

Req. ID	Description	Priority	Status
REQ-1	Allow user to share live or current GPS location.	High	Pending
REQ-2	Access the user device's location services with permission.	High	Pending
REQ-3	Send location data to selected ride participants.	High	Pending
REQ-4	Confirm successful location sharing.	Medium	Pending

Table 83 Send Location Requirements

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- The system must ensure quick responsiveness, with real-time updates, and be scalable to accommodate increasing user demand without performance degradation.

5.2 Safety Requirements

- User data should always be encrypted and backed up daily to prevent losses in case of data loss.
- It shall ensure that no personal information, such as location information in the case of carpooling, is shared without consent**.**

5.3 Security Requirements

- The application will offer safe authentication for the users, using HTTPS to encrypt communication and password hashing. Correspondingly, other sensitive information will be treated in the backend in a security-aware way.

5.4 Software Quality Attributes

- In the app, maintainability should be assured with explicitly clarified modular design principles in order to expand the feature set in the future.
- Reliability: The application should be up 99.9% of the time and never down, especially during exam seasons when the load is at an all-time high.

5.5 Business Rules

- Events can be managed and posted by teachers, while students are able to view and RSVP; parents would have permissions only to view specific data, such as timetables and events.

6. Other Requirements

The **UniVersa app** will use MongoDB for secure and scalable data storage with encryption and backup. It will comply with local data protection laws and provide clear terms of use and privacy policies. Key components like authentication, notifications, and messaging will be designed for reuse in future projects. Security will be ensured through two-factor authentication and strict role-based access control, while the backend will support scalability to accommodate future feature additions.

Software Design Specifications

1. Introduction

1.1. Purpose of this document

This Software Design Specification (SDS) outlines the design decisions, architectural framework, and key components for UniVersa. It aims to guide development and implementation while ensuring a robust, scalable, and user-centric solution for seamless communication and event management across university communities.

1.2 Scope of the development project

UniVersa is a mobile application designed for universities and colleges to enhance communication and coordination among students, teachers, and parents. Key features include real-time event notifications, timetable synchronization, secure messaging, job boards, and a carpooling feature. The app aims to serve as a single-point solution for academic interaction and management.

1.3 Definitions, acronyms, and abbreviations

- SDS: Software Design Specification
- SRS: Software Requirements Specification
- API: Application Programming Interface
- UI/UX: User Interface/User Experience
- UniVersa: Project name of the mobile application
- Flutter: Android-platform UI toolkit
- Node.js: JavaScript runtime for server-side development
- MongoDB: NoSQL database for data storage

1.4 Overview of document

This document includes the following sections:

- Overview of the system architecture (Section 2)
- Detailed component descriptions (Section 3)
- User interface design principles (Section 4)
- Approach to reuse and relationships to other products (Section 5)
- Design decisions and tradeoffs (Section 6)
- Pseudocode and relevant diagrams (Sections 7 and 8)

2. System architecture description

2.1 Section overview

UniVersa follows a three-tier architecture to ensure scalability and maintainability:

- **Presentation Layer:** Flutter-based front-end.
- **Business Logic Layer:** Node.js server managing app logic.

- **Data Access Layer:** MongoDB for persistent storage.

2.2 General constraints

- **Platform**: Android and iOS using Flutter.
- **Backend**: Node.js for scalability and real-time processing.
- **Database**: MongoDB for dynamic and structured data storage.
- **Network requirements**: Internet connectivity for real-time updates.

2.3 Data design

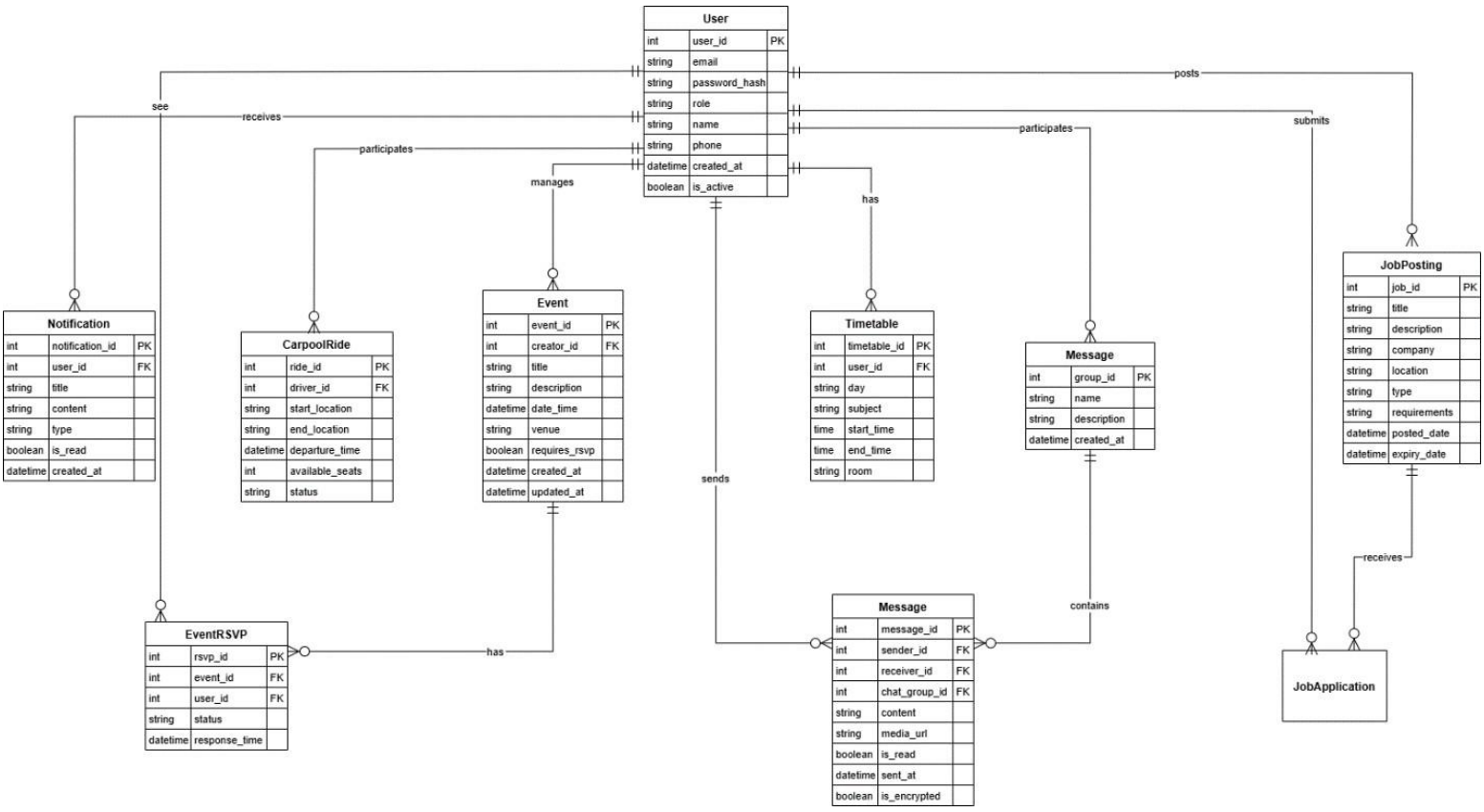

Figure 15 Data Design

2.4 Program structure

UniVersa adopts a modular architecture with the following layers:

- **Presentation Layer**: Handles user interactions.
- **Business Logic Layer**: Implements messaging, event management, and timetable synchronization.
- **Data Access Layer**: Interfaces with MongoDB for data storage and retrieval.

2.5 Alternatives considered

- Relational databases were considered but excluded for scalability.
- The firebase was considered but replaced by MongoDB for better control over data structures.

3. Detailed description of components

3.1 Section overview

This section describes the critical components of UniVersa and their purposes.

3.2 Component and Detail

3.2.1.1 User Authentication and Account Management

Description: Allow Users to Sign in to the App.

Data Members:

- Include Type: Views for Client and Freelancers for data entry.
- Visibility: All users

Methods:

Send request for verification to the back end.

Identification	**User Authentication**
Type	Module
Purpose	Manages secure user authentication, account creation, and role-based access.
Function	Handles login, signup, password recovery, and user role management.
Subordinates	Authentication handler, session management.
Dependencies	Database for user credentials, encryption library.
Interfaces	Login screen UI, backend API for validation.
Resources	Database, encryption algorithms.
Processing	Validates credentials, establishes session, logs login activity.
Data	User credentials, session tokens, login attempts.

Table 84 User Authentication component table

3.2.2 Event Management

Description: Allow management of events within the app.

Data Members:

- Event Data: Includes details such as date, time, location, and participants.
- Visibility: Admins and Users (read-only for users).

Methods:

- Create event records
- Update event records
- Delete event records

Identification	**Event Management**
Type	Subsystem
Purpose	Organizes and manages events within the app environment.
Function	Enables users to create, edit, view, and delete events.
Subordinates	Event creation module, notification handler.
Dependencies	Calendar module, notification service.
Interfaces	Event management UI, notification system.
Resources	Database, event scheduler.
Processing	Stores events, triggers notifications, syncs with user calendars.
Data	Event details, participant data.

Table 85 Event Management component table

3.2.3 Timetable Management

Description: Provides a feature for managing and accessing timetables.

Data Members:

- Timetable Entries: Details of schedules for classes or events.
- Visibility: Admins (full access), Users (view-only).

Methods:

- Add, modify, or delete timetable records.

Identification	**Timetable Management**
Type	Subsystem
Purpose	Manages scheduling and organizes activities.
Function	Allows admins to create and modify timetables, viewable by users.
Subordinates	Scheduler, notification handler.
Dependencies	Calendar services, database.
Interfaces	Timetable UI, calendar sync API.
Resources	Database, scheduling tools.
Processing	Updates timetables, resolves conflicts, syncs schedules.
Data	Schedule details, user preferences.

Table 86 Timetable Management component table

3.2.4 Message Management

Description: Provides real-time messaging capability between users.

Data Members:

- Message Content: Text, metadata (timestamps, sender/receiver IDs).
- Visibility: Sender and receiver only.

Methods:

- Send, receive, and archive messages.

Identification	**Message Management**
Type	Module
Purpose	Facilitates communication between users securely.
Function	Manages real-time and archived messages.
Subordinates	Message sender, receiver, archiver.
Dependencies	Database for message storage, push notification services.
Interfaces	Messaging UI, backend-messaging API.
Resources	Database, network bandwidth.
Processing	Stores and retrieves messages, sends delivery notifications.

Table 87 Message Management component table

3.2.5 Job Management

Description: Manages job postings and applications.

Data Members:

- Job Listings: Title, description, and application data.
- Visibility: Admins and job seekers.

Methods:

- Post job entries
- Edit job entries
- Delete job entries
- search job postings.

Identification	**Job Management**
Type	Subsystem
Purpose	Provides tools for job postings and applications.
Function	Handles posting, searching, and application tracking.
Subordinates	Job posting module, application handler.
Dependencies	Database, search index.
Interfaces	Job board UI, backend API.
Resources	Database storage, search tools.
Processing	Validates job data, updates database, fetches search results.
Data	Job details, applicant data.

Table 88 Job Management component table

3.2.6 Lost and Found Management

Description: Assists in reporting and managing lost and found items.

Data Members:

- Lost Items: Description, location, and reporting user.
- Found Items: Description, location, and finder.
- Visibility: All users.

Methods:

- Report, search, and claim items.

Identification	**Lost And Found Management**
Type	Subsystem
Purpose	Facilitates reporting and claiming lost and found items.
Function	Tracks lost and found items, allows claims.
Subordinates	Reporting module, claims handler.
Dependencies	Database, notification services.
Interfaces	Lost and found UI, backend API.
Resources	Database, notification services.
Processing	Stores reports, matches lost and found items, sends updates.
Data	Item descriptions, reporting timestamps.

Table 89 Lost and Found Management component table

3.2.7 Real-Time Notifications

Description: Sends real-time alerts to users for important updates.

Data Members:

- Notification Content: Message text, timestamp, priority level.
- Visibility: Targeted users.

Methods:

- Trigger, deliver, and manage notifications.

Identification	**Real-Time Notifications**
Type	Module
Purpose	Keeps users informed about critical updates and alerts.
Function	Sends and displays notifications in real time.
Subordinates	Notification handler, delivery tracker.
Dependencies	Push notification services, user database.
Interfaces	Notification UI, backend API.
Resources	Notification servers, network bandwidth.
Processing	Filters notifications by priority, delivers them to users.

Table 90 Notification component table

3.2.8 Group Management

Description: Allows users to create and manage groups for collaboration.

Data Members:

- Group Data: Group name, members, and permissions.
- Visibility: Admins and group members.

Methods:

- Create groups
- Join groups
- Delete groups.

Identification	**Group Management**
Type	Subsystem
Purpose	Facilitates group collaboration and communication.
Function	Handles group creation, member management, and permissions.
Subordinates	Group handler, permission manager.
Dependencies	Database, messaging service.
Interfaces	Group management UI, backend API.
Resources	Database, communication tools.
Processing	Validates group actions, updates member lists.
Data	Group details, member data, permissions.

Table 91 Group Management component table

3.2.9 Resource Library Management

Description: Provides a central repository for sharing resources.

Data Members:

- Resource Data: File name, type, and upload details.
- Visibility: Admins and users.

Methods:

- Upload, download, and manage resources.

Identification	**Resource Library Management**
Type	Subsystem
Purpose	Enables resource sharing and access.
Function	Manages uploaded resources, organizes library.
Subordinates	Upload handler, search module.
Dependencies	File storage system, search tools.
Interfaces	Library UI, backend API.
Resources	Storage servers, indexing services.
Processing	Handles file uploads/downloads, organizes resources.
Data	Resource metadata, user activity logs.

Table 92 Resource Library Management component table

3.2.10 Marketplace

Description: Provides a centralized platform for buying, selling, and trading items or services within the community.

Data Members:

- **Item Data:** Item name, category, price, description, and upload details.
- **Visibility:** Admins and users.

Methods:

- Upload, browse, purchase, and manage listings.

Identification	**Marketplace**
Type	Subsystem
Purpose	Enables item trading and transactions within the community.
Function	Manages uploaded listings, organizes marketplace, and facilitates purchases.
Subordinates	Upload handler, search module, payment processor.
Dependencies	Payment gateway, search tools.

Interfaces	Marketplace UI, backend API.
Resources	Storage servers, indexing services.
Processing	Handles item uploads/downloads, processes transactions.
Data	Item metadata, transaction logs, user activity logs.

Table 93 Marketplace component table

3.3.11 To-Do List

Description: Provides a tool for users to create, manage, and track personal or academic tasks and deadlines.

Data Members:

- **Task Data:** Task title, description, due date/time, and priority.
- **Visibility:** Admins and users.

Methods:

- Create, edit, and remove tasks.

Identification	**To-Do List**
Type	Subsystem
Purpose	Enables task management and tracking for users.
Function	Manages task creation, organizes to-do lists, and updates task status.
Subordinates	Task creation module, notification scheduler.
Dependencies	User profile system, calendar tools.
Interfaces	To-Do UI, backend API.
Resources	Storage servers, scheduling services.
Processing	Handles task creation/updates, sends reminders.
Data	Task metadata, user activity logs.

Table 94 To-Do List component table

3.3.12 Carpool

Description: Facilitates shared transportation by allowing users to create, join, and manage carpool rides.

Data Members:

- **Ride Data:** Pickup point, drop-off point, time, vehicle details, and available seats.
- **Visibility:** Admins and users.

Methods:

- Add, edit, remove, and share location for rides.

Identification	**Carpool**
Type	Subsystem
Purpose	Enables shared transportation and ride coordination.
Function	Manages postings, organizes carpool listings, and shares locations.
Subordinates	Ride creation module, location tracker.
Dependencies	GPS services, user authentication.
Interfaces	Carpool UI, backend API.
Resources	GPS servers, notification services.
Processing	Handles ride creation/updates, processes location sharing.
Data	Ride metadata, user participation logs.

Table 95 Carpool component table

4. User Interface Design

4.1 Section Overview

UniVersa's UI is designed for simplicity, accessibility, and efficiency.

4.2 Interface Design Rules

- Consistency in colors, typography, and navigation.
- Material design principles.

- Mobile responsiveness.

4.3 GUI components

- Navigation Menu: Role-based access to features
- Dashboard: Centralized display notifications and updates
- Interactive Calendar: View timetables and event details

4.4 Detailed Description

- **Student View**: Access to schedules, job boards, and messaging, Carpool, to-do-list, Lost & Found, resource Library.
- **Teacher View**: Access to schedules, job boards, and messaging, accept Requests.
- **Parent View**: Child's timetable, events, and messaging.
- **Admin View:** Tools for event creation, user management, and analytics.

5. Reuse and Relationships to other products

5.1 Reuse Strategy

- Adopt open-source libraries for UI components.
- Integrate third-party APIs for job board listings and carpool matching.
- Reuse design templates and Node.js modules

6. Design decisions and tradeoffs

Key decisions included prioritizing modular design and MongoDB for flexibility. Excluding real-time tracking reduced complexity and improved resource management.

7. Pseudocode for Components

7.1 Signup

```
void userSignup(Map<String, String> details) {
  showSignupForm();
  if (validateDetails(details)) {
    sendActivationEmail(details['email']!);
  } else {
    showError("Invalid details");
  }
}
```

7.2 Login

```
void userLogin(Map<String, String> credentials) {
  if (validateCredentials(credentials)) {
  grantAccess();
```

```
    } else {
      showError("Invalid username or password");
    }
  }
```

7.3 Forgot Password

```
  void forgotPassword(String email) {
    if (validateEmail(email)) {
    sendResetLink(email);
    } else {
      showError("Invalid email address");
    }
  }
```

7.4 Add Event

```
  void addEvent(Map<String, String> eventDetails) {
    if (validateEvent(eventDetails)) {
    saveEvent(eventDetails);
    } else {
      showError("Invalid event details");
    }
  }
```

7.5 Edit Event

```
void editEvent(String eventId, Map<String, String> newDetails) {
  if (validateEvent(newDetails)) {
    updateEvent(eventId, newDetails);
  } else {
    showError("Invalid updates");
  }
}
```

7.6 Delete Event

```
void deleteEvent(String eventId) {
  confirmDeletion();
  removeEvent(eventId);
```

```
}
```

7.7 Send Message

```
void sendMessage(Map<String, String> messageDetails) {
  if (validateMessage(messageDetails)) {
  deliverMessage(messageDetails);
  } else {
    showError("Message validation failed");
  }
}

List<String> receiveMessages(String userId) {
  return fetchMessages(userId);
}
```

7.8 Post Job

```
void postJob(Map<String, String> jobDetails) {
  if (validateJob(jobDetails)) {
  publishJob(jobDetails);
  } else {
    showError("Invalid job details");
  }
}
```

7.9 Edit Job

```
void editJob(String jobId, Map<String, String> updatedJob) {
  if (validateJob(updatedJob)) {
    updateJob(jobId, updatedJob);
  } else {
    showError("Invalid updates");
  }
```

```
    }
```

7.10 Remove Job

```
    void removeJob(String jobId) {
      confirmDeletion();
      deleteJob(jobId);
    }
```

7.11 Add Timetable

```
    void addTimetable(Map<String, String> timetableDetails) {
      if (validateTimetable(timetableDetails)) {
      saveTimetable(timetableDetails);
      } else {
        showError("Invalid timetable");
      }
    }
```

7.12 Edit Timetable

```
void editTimetable(String timetableId, Map<String, String> newDetails) {
  updateTimetable(timetableId, newDetails);
}
```

7.13 Remove Timetable

```
void removeTimetable(String timetableId) {
  confirmDeletion();
  deleteTimetable(timetableId);
}
```

7.14 Send Notifications

```
void sendNotification(Map<String, String> notificationDetails) {
  if (validateNotification(notificationDetails)) {
  pushNotification(notificationDetails);
```

```
  } else {
    showError("Invalid notification");
  }
}

List<String> receiveNotifications(String userId) {
  return fetchNotifications(userId);
}
```

7.15 Marketplace

```
void addProduct(Map<String, String> productDetails) {
  if (validateProduct(productDetails)) {
  saveProductToMarketplace(productDetails);
  showSuccess("Product listed successfully");
  } else {
    showError("Invalid product details");
  }
}
List<Map<String, String>> getAvailableProducts() {
  return fetchAllProductsFromMarketplace();
}
Map<String, String> viewProductDetails(String productId) {
  Map<String, String> product = fetchProductById(productId);
  if (product != null) {
    display(product["title"]);
    display(product["description"]);
    display(product["price"]);
    display(product["sellerPhoneNumber"]);
    display(product["sellerEmail"]);
    return product;
  } else {
    showError("Product not found");
```

```
    return {};
  }
}
```

7.16 Delete Marketplace

```
void deleteProduct(String productId, String sellerId) {
  Map<String, String> product = fetchProductById(productId);

  if (product != null) {
    if (product["sellerId"] == sellerId) {
      removeProductFromMarketplace(productId);
      showSuccess("Product deleted successfully");
    } else {
      showError("Unauthorized action: Only the seller can delete this product");
    }
  } else {
    showError("Product not found");
  }
}
```

7.17 Edit Marketplace

```
void editProduct(String productId, String sellerId, Map<String, String> updatedDetails) {
  Map<String, String> product = fetchProductById(productId);

  if (product != null) {
    if (product["sellerId"] == sellerId) {
      if (validateProduct(updatedDetails)) {
        updateProductInMarketplace(productId, updatedDetails);
        showSuccess("Product updated successfully");
      } else {
```

```
      showError("Invalid updated details");
     }
    } else {
     showError("Unauthorized action: Only the seller can edit this product");
    }
  } else {
   showError("Product not found");
  }
}
```

7.18 Carpool

```
void addCarpool(Map<String, String> carpoolDetails) {
  if (validateCarpoolDetails(carpoolDetails)) {
  saveCarpoolToDatabase(carpoolDetails);
  showSuccess("Carpool ride added successfully");
  } else {
   showError("Invalid carpool details");
  }
}
```

7.19 Edit Carpool

```
void editCarpool(String carpoolId, String userId, Map<String, String> updatedDetails) {
  if (isOwner(carpoolId, userId) && validateCarpoolDetails(updatedDetails)) {
  updateCarpool(carpoolId, updatedDetails);
   showSuccess("Carpool updated");
  } else {
   showError("Invalid details or unauthorized access");
  }
}
```

7.20 Delete Carpool

```
void deleteCarpool(String carpoolId, String userId) {
```

```
  if (isOwner(carpoolId, userId)) {
    removeCarpool(carpoolId);
    showSuccess("Carpool deleted");
  } else {
    showError("Unauthorized access");
  }
}
```

7.21 Carpool Using

```
void requestCarpool(String carpoolId, String userId, String message) {
  Map<String, String> carpool = fetchCarpoolById(carpoolId);

  if (carpool != null && userId != carpool["ownerId"]) {
    sendMessageToCarpoolOwner(carpool["ownerId"], userId, message);
    showSuccess("Request sent to carpool owner");
  } else {
    showError("Invalid request");
  }
}
```

7.22 Location Sharing

```
void shareLocation(String userId, String receiverId, Location currentLocation) {
  if (validateUser(userId) && validateUser(receiverId)) {
  sendLocationToReceiver(receiverId, currentLocation);
  showSuccess("Location shared successfully");
  } else {
    showError("User validation failed");
  }
}
```

7.23 Add To-Do List

```
void addToDoTask(String userId, String description, DateTime dateTime) {
  if (validateTask(description, dateTime)) {
    saveTask(userId, description, dateTime);
    scheduleNotification(userId, description, dateTime);
    showSuccess("To-Do task added and reminder set");
  } else {
    showError("Invalid task details");
  }
}
```

7.24 Edit To-Do List

```
void editToDoTask(String taskId, String userId, String newDescription, DateTime
newDateTime) {
  if (isTaskOwner(taskId, userId) && validateTask(newDescription, newDateTime)) {
    updateTask(taskId, newDescription, newDateTime);
    rescheduleNotification(taskId, newDateTime);
    showSuccess("To-Do task updated");
  } else {
    showError("Invalid update or unauthorized access");
  }
}
```

7.25 Delete To-Do List

```
void deleteToDoTask(String taskId, String userId) {
  if (isTaskOwner(taskId, userId)) {
  cancelScheduledNotification(taskId);
  removeTask(taskId);
    showSuccess("To-Do task deleted");
  } else {
    showError("Unauthorized access");
```

```
  }
}
```

7.26 Add Notes

```
void addNote(String userId, String title, String content) {
  if (validateNote(title, content)) {
    saveNote(userId, title, content);
    showSuccess("Note saved successfully");
  } else {
    showError("Invalid note details");
  }
}
```

7.27 Edit Notes

```
void editNote(String noteId, String userId, String newTitle, String newContent) {
  if (isNoteOwner(noteId, userId) && validateNote(newTitle, newContent)) {
  updateNote(noteId, newTitle, newContent);
    showSuccess("Note updated successfully");
  } else {
    showError("Unauthorized access or invalid data");
  }
}
```

7.28 Delete Notes

```
void deleteNote(String noteId, String userId) {
  if (isNoteOwner(noteId, userId)) {
  removeNote(noteId);
    showSuccess("Note deleted successfully");
  } else {
    showError("Unauthorized access");
  }
}
```

7.29 Resource Library

```
void uploadResource(String userId, String title, String description, File resourceFile) {
  if (validateResource(title, description, resourceFile)) {
  saveResourceToLibrary(userId, title, description, resourceFile);
  showSuccess("Resource uploaded successfully");
  } else {
    showError("Invalid resource details");
  }
}
```

7.30 Delete Resource Library

```
void deleteResource(String resourceId, String userId) {
  if (isResourceOwner(resourceId, userId)) {
  removeResourceFromLibrary(resourceId);
  showSuccess("Resource deleted successfully");
  } else {
    showError("Unauthorized access");
  }
}
```

7.31 Message to User

```
void sendMessage(String senderId, String receiverId, String message) {
  if (areUsersRegistered(senderId, receiverId)) {
  saveMessage(senderId, receiverId, message);
    showSuccess("Message sent");
  } else {
    showError("User not found");
  }
}
```

7.32 Message to Teacher

```
void requestTeacherChat(String studentId, String teacherId) {
```

```
  sendRequestToTeacher(studentId, teacherId);
  showSuccess("Message request sent to teacher");
}

void sendMessageToTeacher(String studentId, String teacherId, String message) {
  if (isRequestApproved(studentId, teacherId)) {
    saveMessage(studentId, teacherId, message);
    showSuccess("Message sent to teacher");
  } else {
    showError("Teacher has not approved your message request");
  }
}
```

7.33 Create Group

```
void createPublicGroup(String creatorId, String groupName, String description) {
  if (validateGroupDetails(groupName, description)) {
  saveGroupToDatabase(creatorId, groupName, description, isPublic = true);
  showSuccess("Public group created");
  } else {
    showError("Invalid group details");
  }
}
```

7.34 Send message in Group

```
void sendGroupMessage(String groupId, String senderId, String message) {
  if (isUserInGroup(groupId, senderId)) {
    saveGroupMessage(groupId, senderId, message);
    showSuccess("Message sent in group");
  } else {
    showError("Join the group to send messages");
```

```
  }
}
```

7.35 Join the group

```
void joinPublicGroup(String userId, String groupId) {
  if (!isUserInGroup(groupId, userId)) {
  addUserToGroup(groupId, userId);
  showSuccess("Joined public group");
  } else {
    showError("Already a member");
  }
}
```

7.36 Post Lost and Found

```
void postLostFoundItem(String userId, String title, String description, File image) {
  if (validateLostItem(title, description, image)) {
    saveLostItem(userId, title, description, image);
    showSuccess("Lost & Found item posted");
  } else {
    showError("Invalid item details");
  }
}
```

7.37 Delete Lost and Found

```
void deleteLostFoundItem(String itemId, String userId) {
  if (isItemOwner(itemId, userId)) {
  removeLostItem(itemId);
    showSuccess("Item deleted from Lost & Found");
  } else {
    showError("Unauthorized access");
  }
}
```

8. Appendices

8.1 Class Diagram

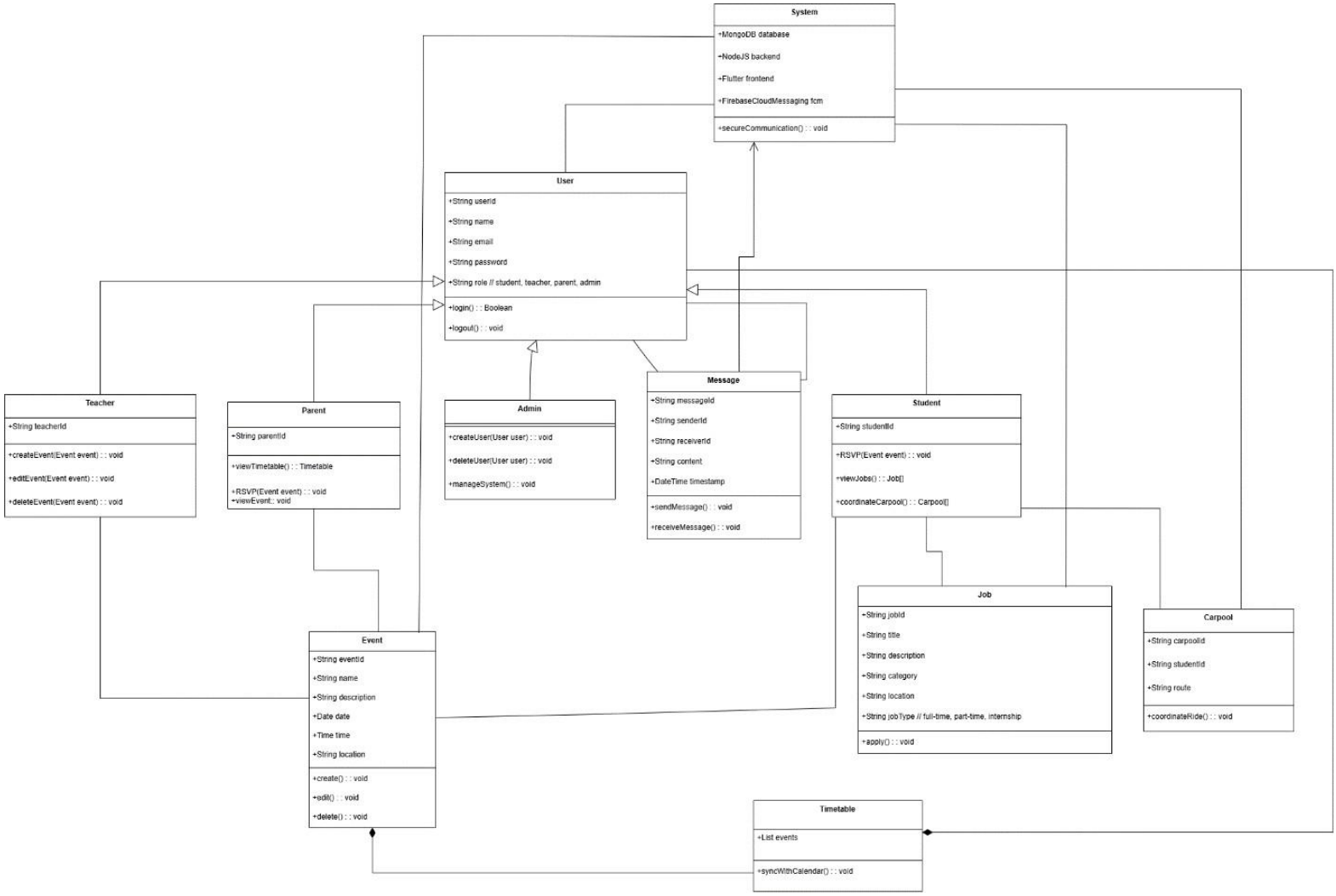

Figure 16 Class Diagram

8.2 Activity Diagram

8.2.1 Signup

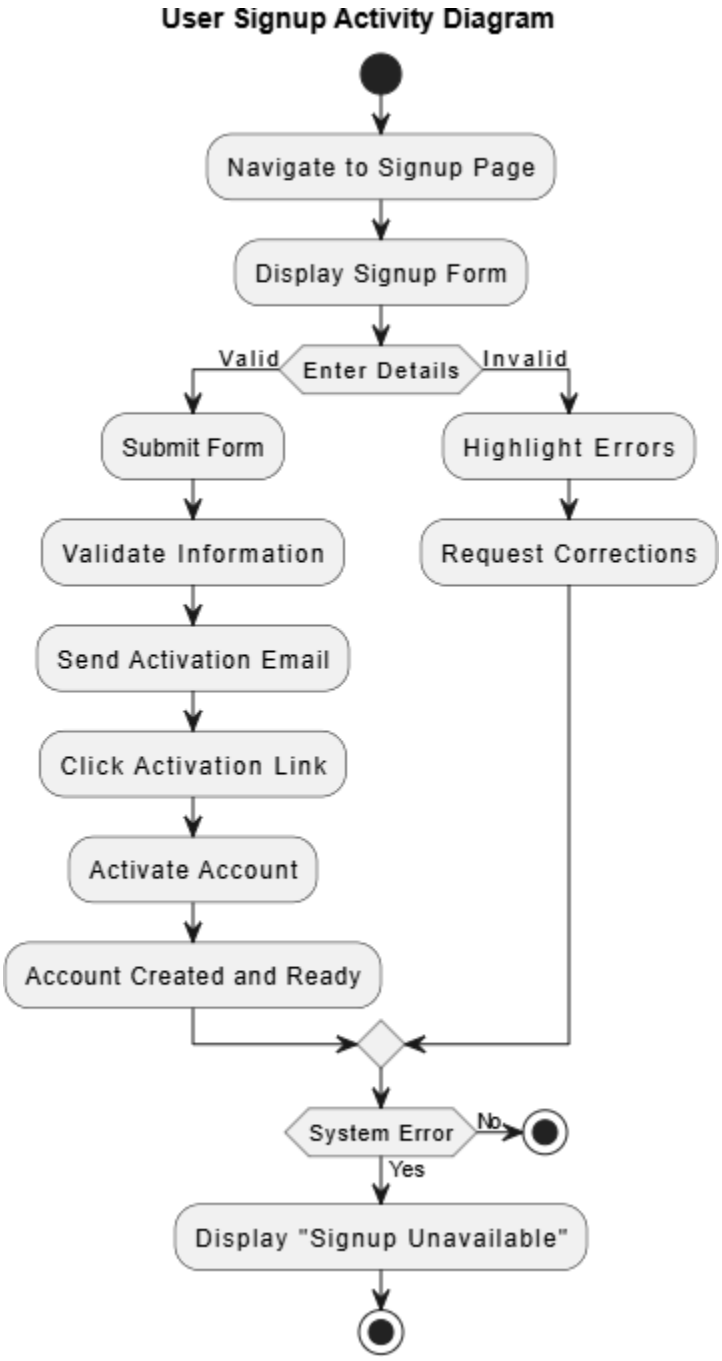

Figure 17 Sign-Up Activity Diagram

8.2.2 Login

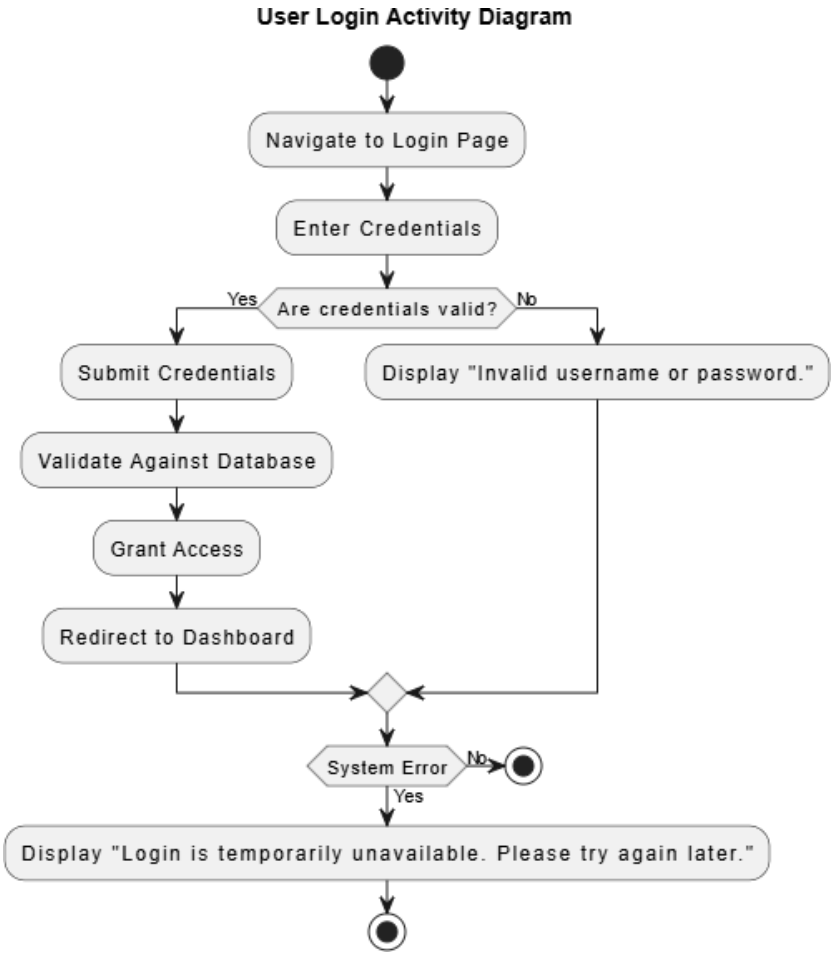

Figure 18 Login Activity Diagram

8.2.3 Forget Password

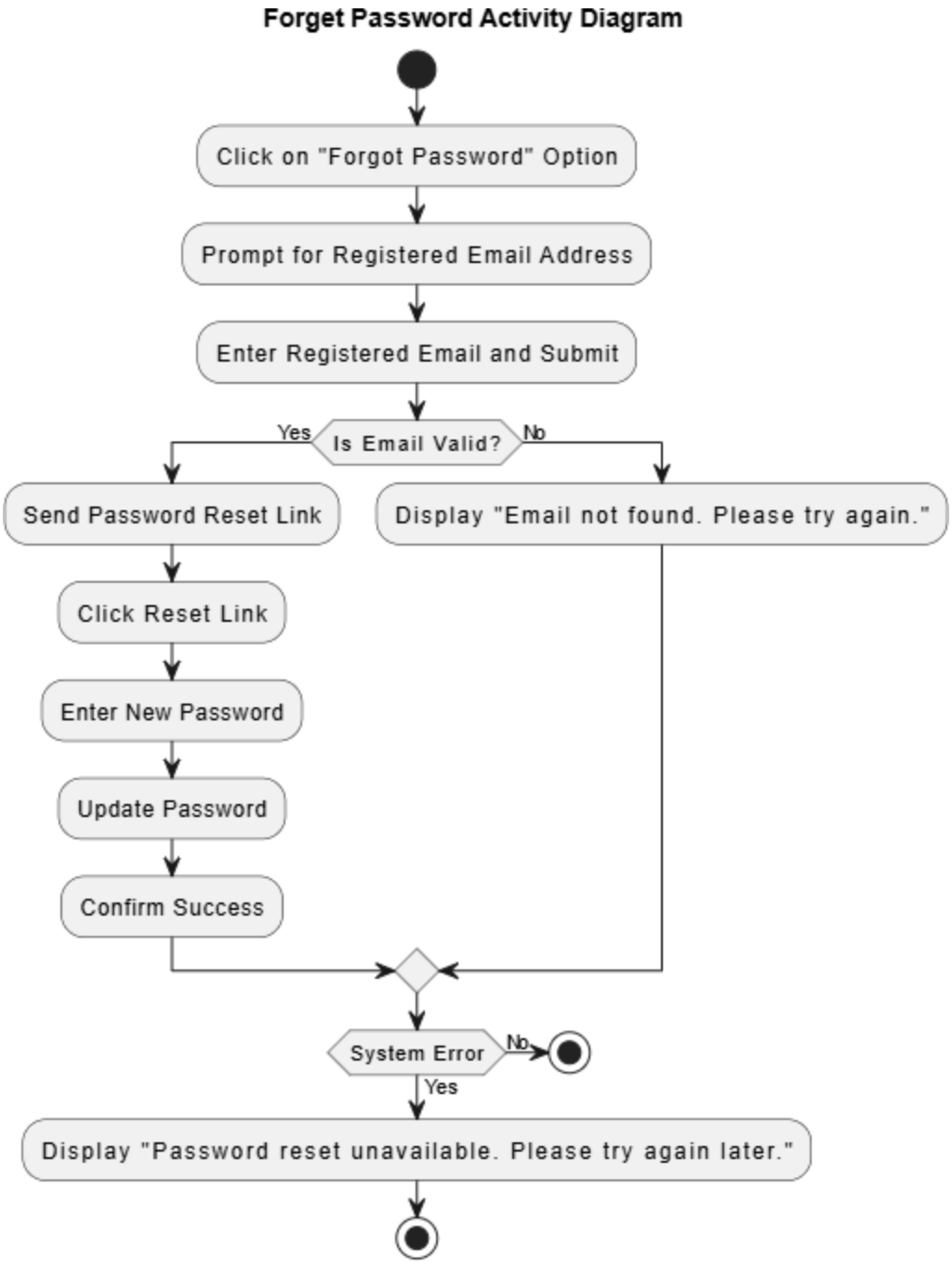

Figure 19 Forget Password Activity Diagram

8.2.4 Add Event

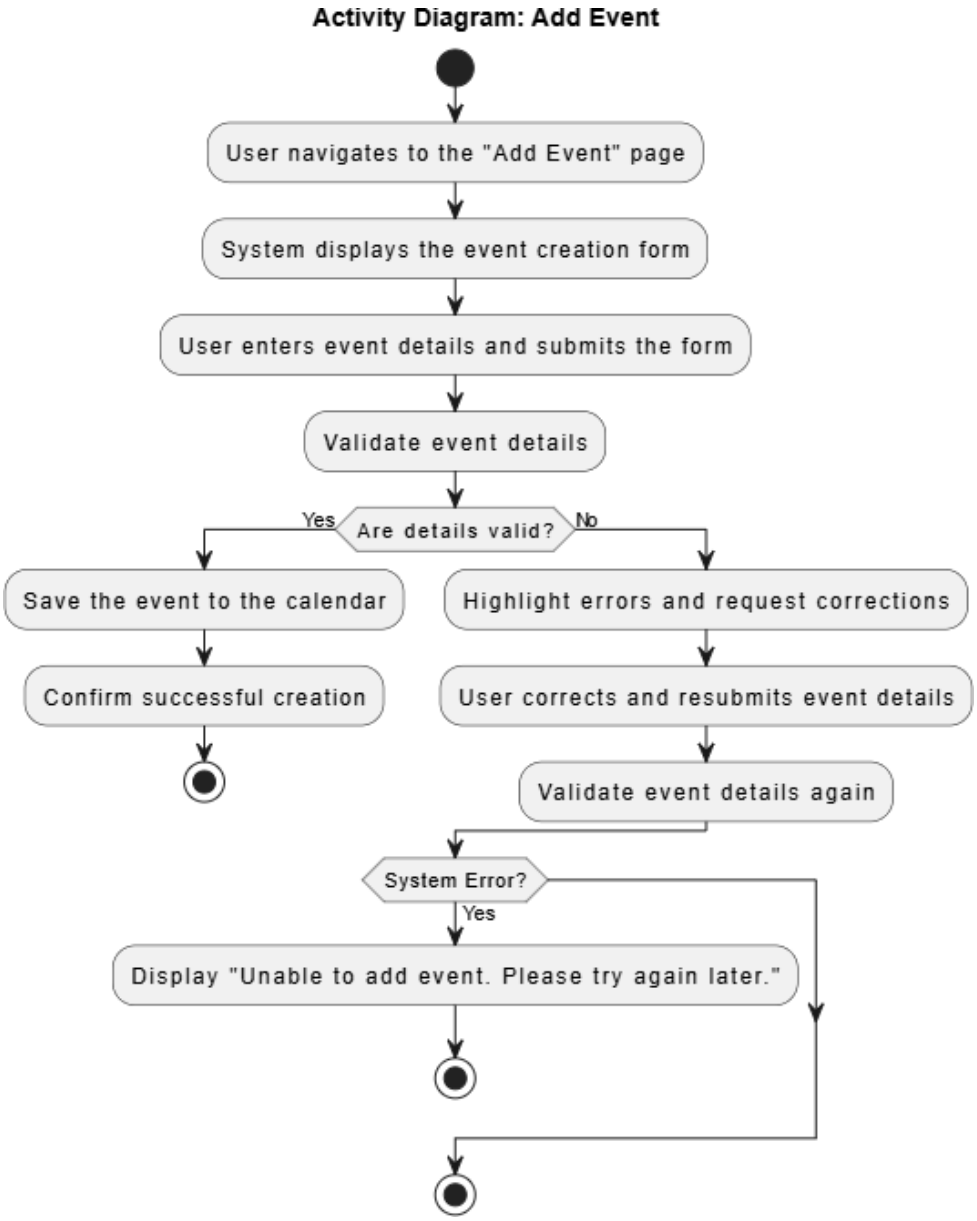

Figure 20 Add Event Activity Diagram

8.2.5 Remove Event

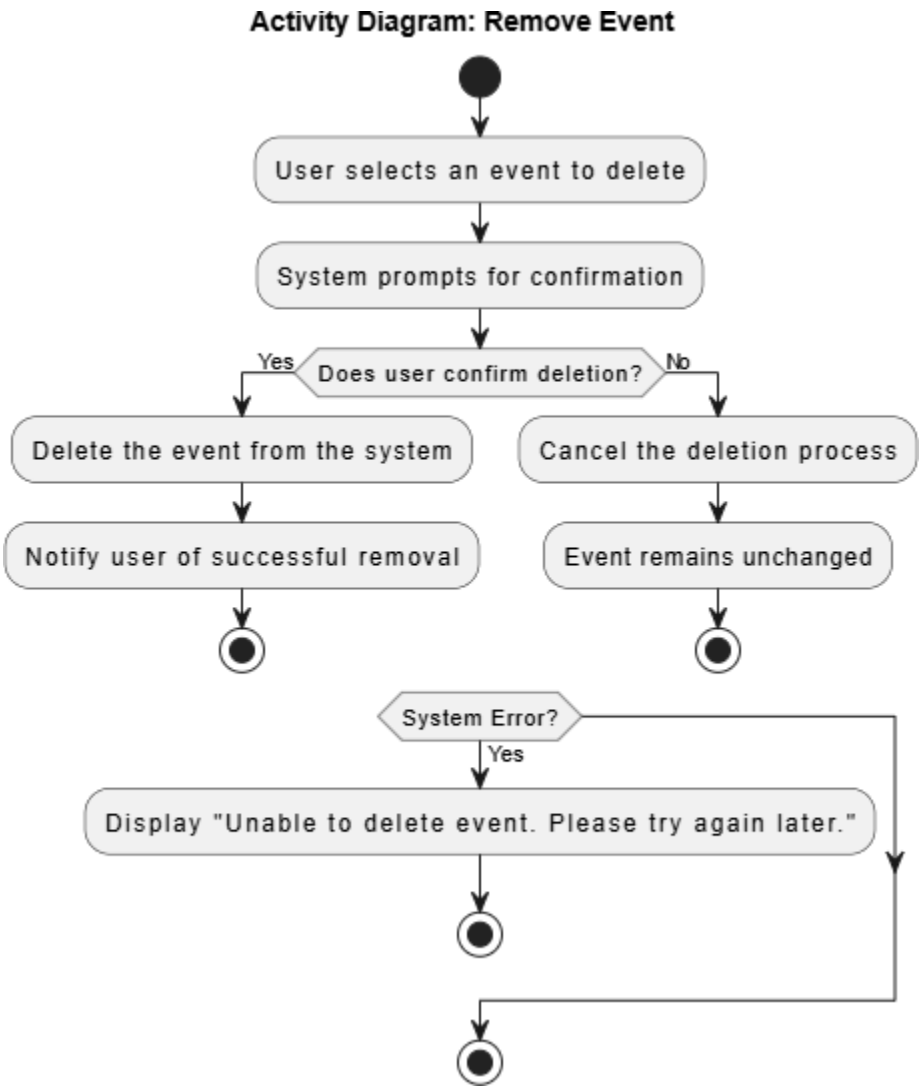

Figure 21 Remove Event Activity Diagram

8.2.6 Edit Event

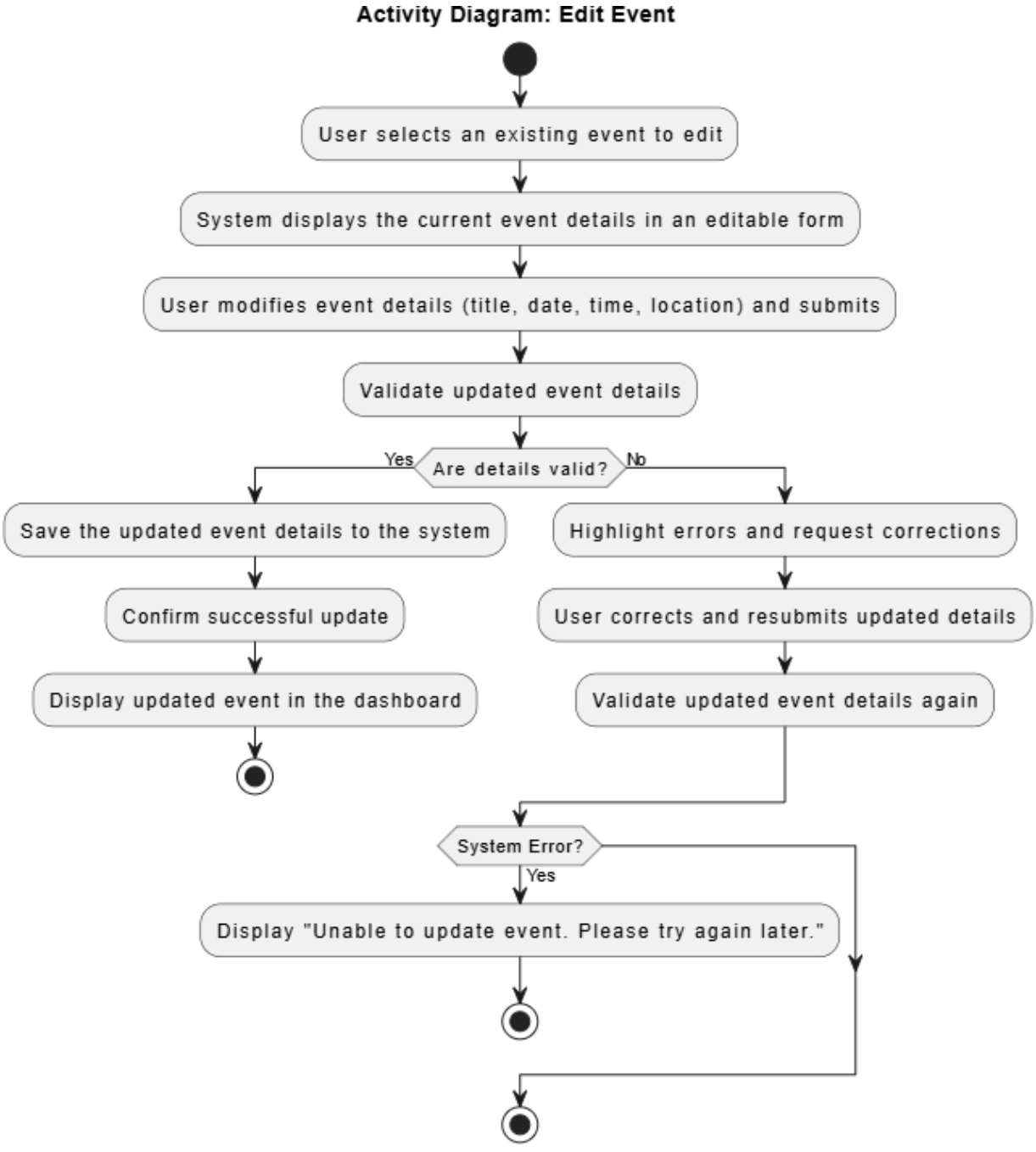

Figure 22 Edit Event Activity Diagram

8.2.7 View Event

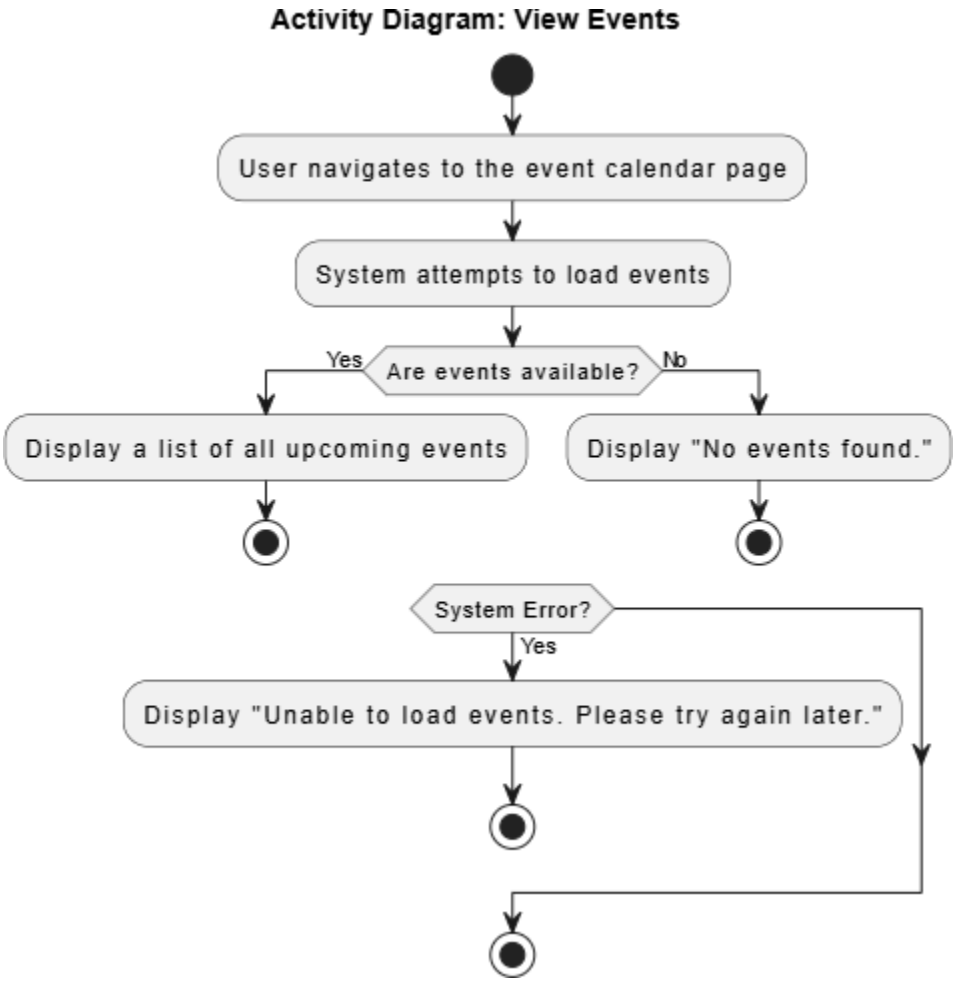

Figure 23 View Events Activity Diagram

8.2.8 Post Job

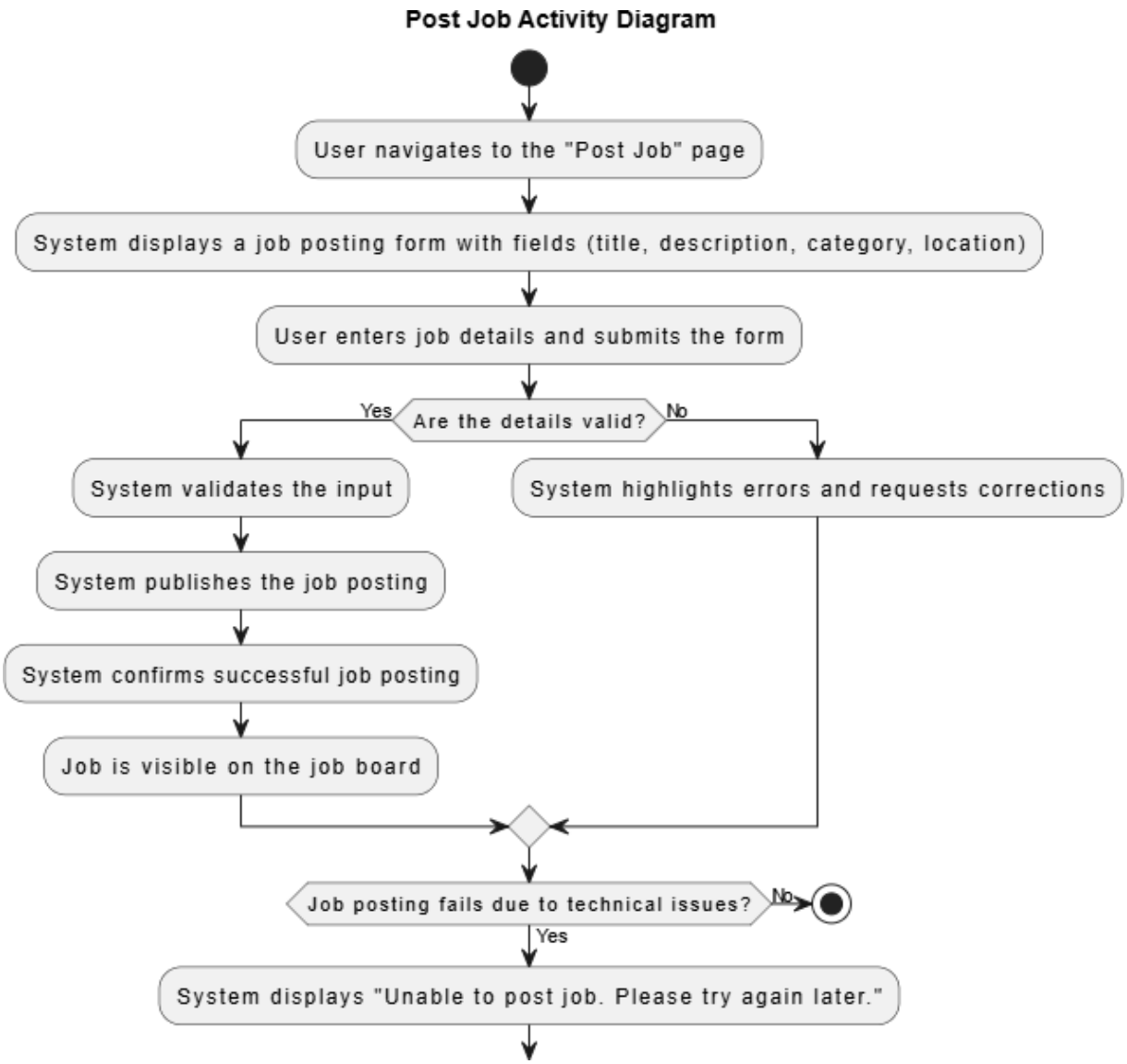

Figure 24 Post Job Activity Diagram

8.2.9 Remove Job

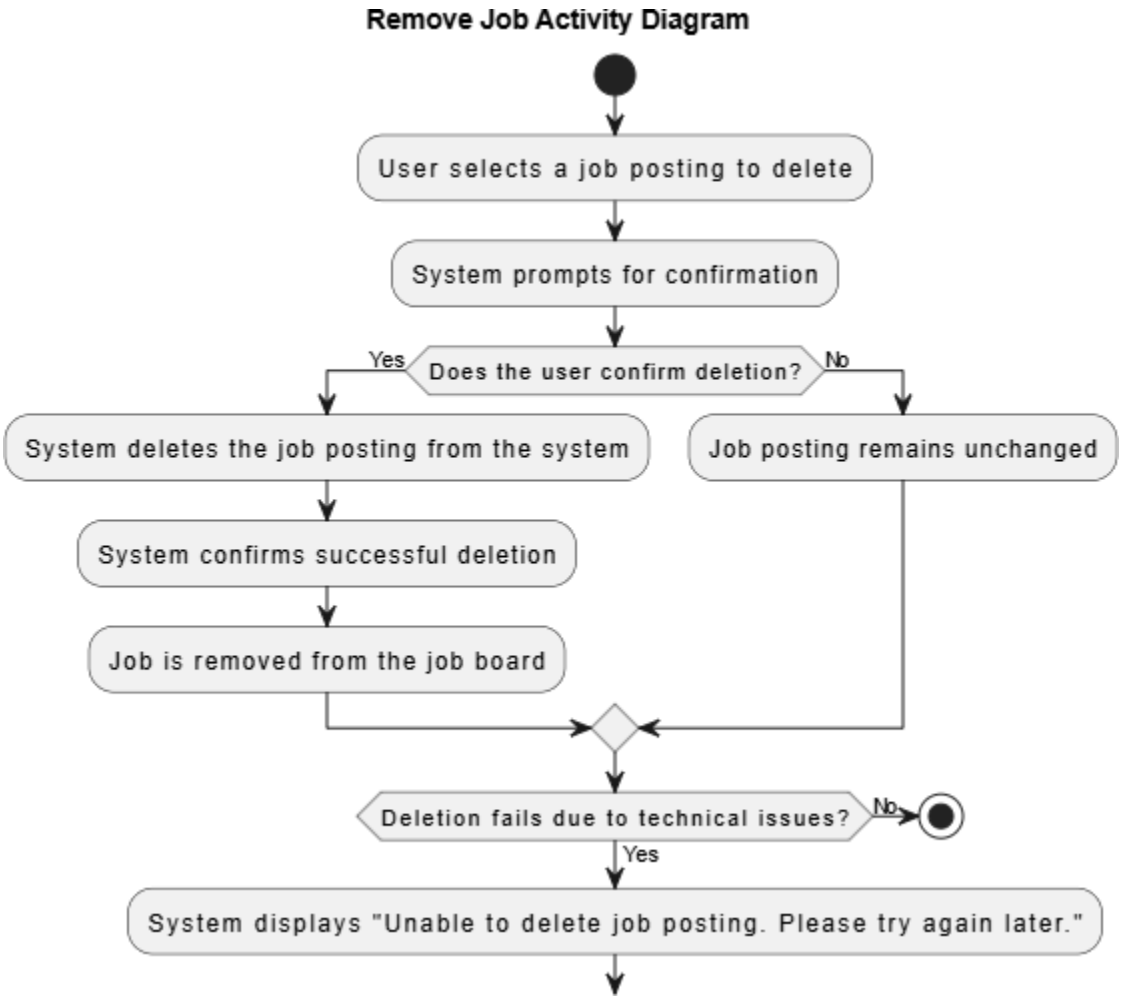

Figure 25 Remove Job Activity Diagram

8.2.10Edit Job

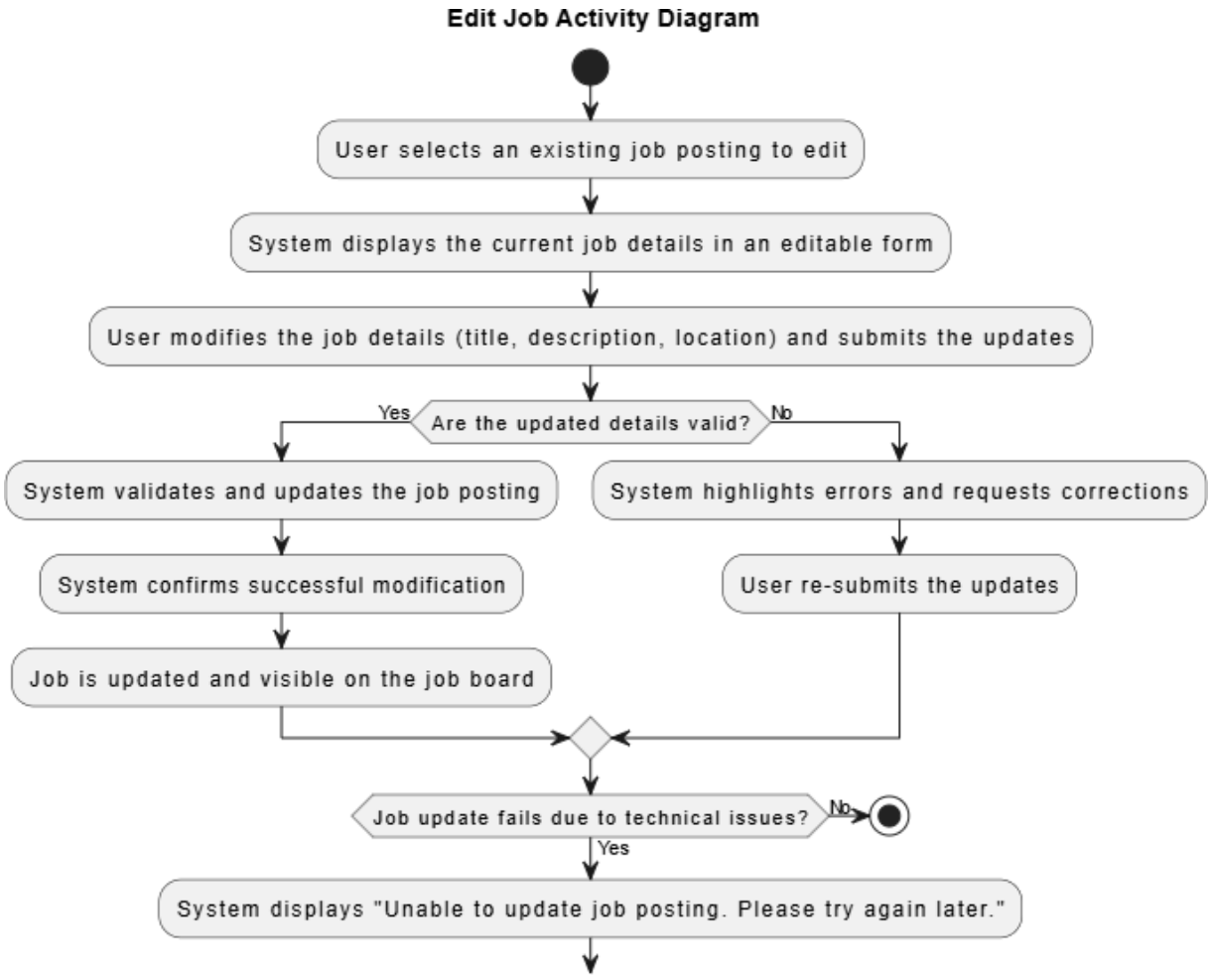

Figure 26 Edit Job Activity Diagram

8.2.11Manage Job

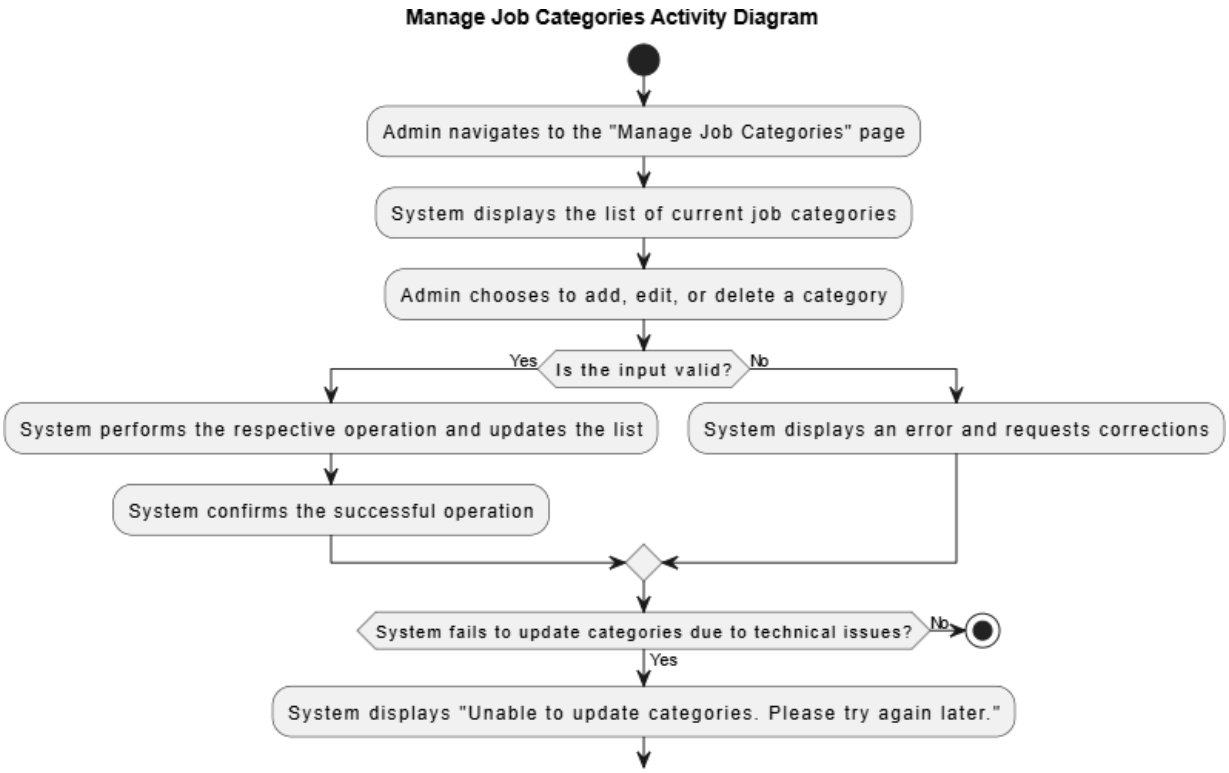

Figure 27 Manage Job Activity Diagram

8.2.12 Receive Message

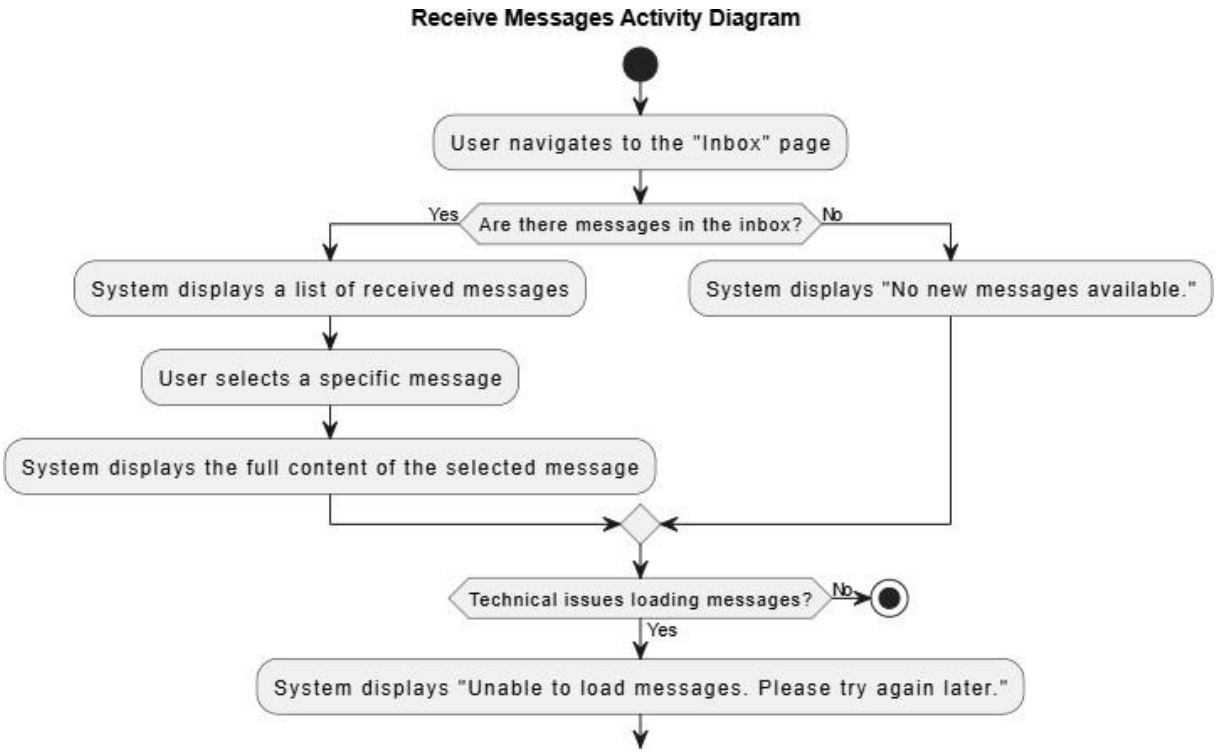

Figure 28 Receive Message Activity Diagram

8.2.13 Send Message

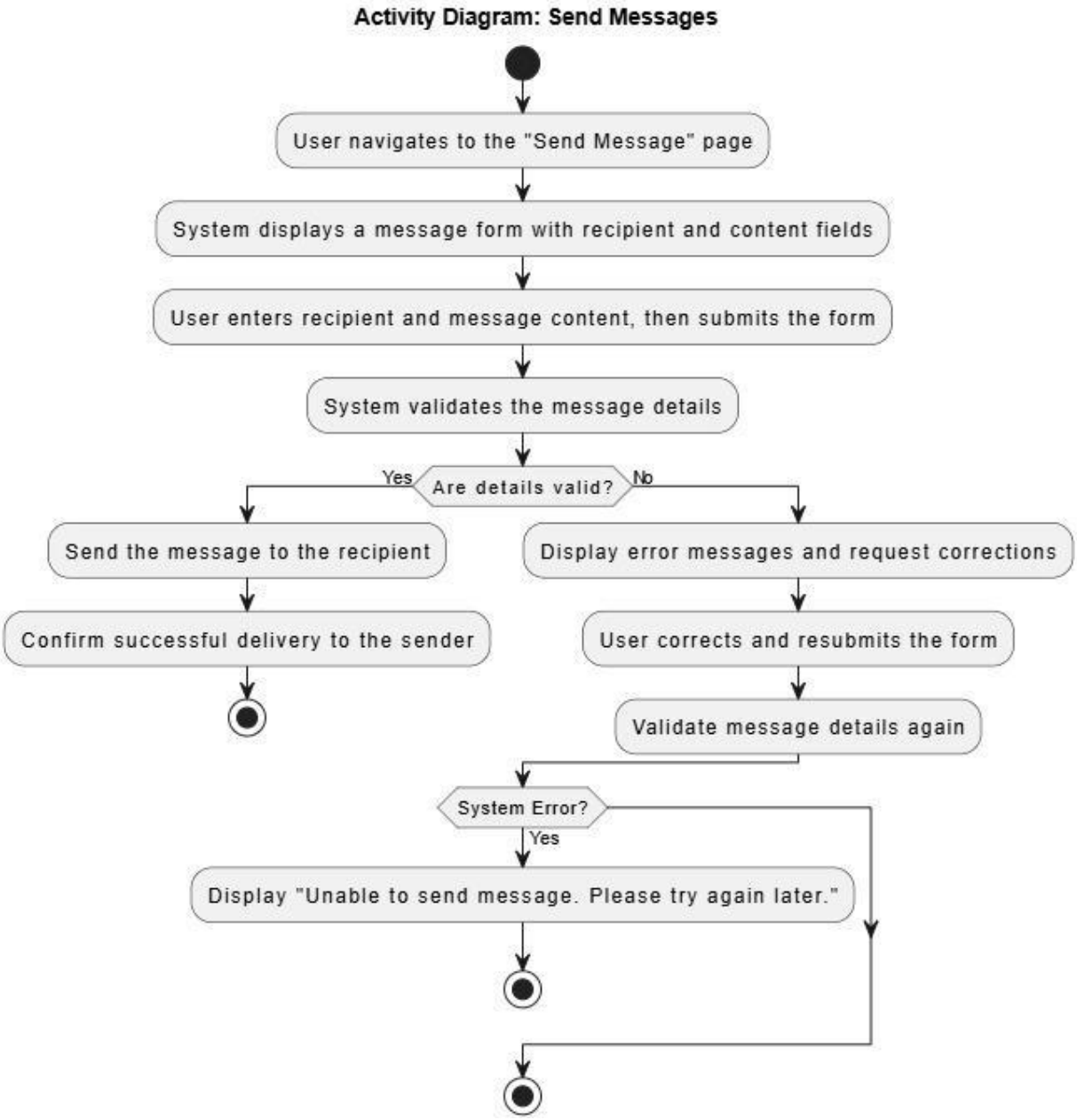

Figure 29 Send Message Activity Diagram

8.2.14 Join Group

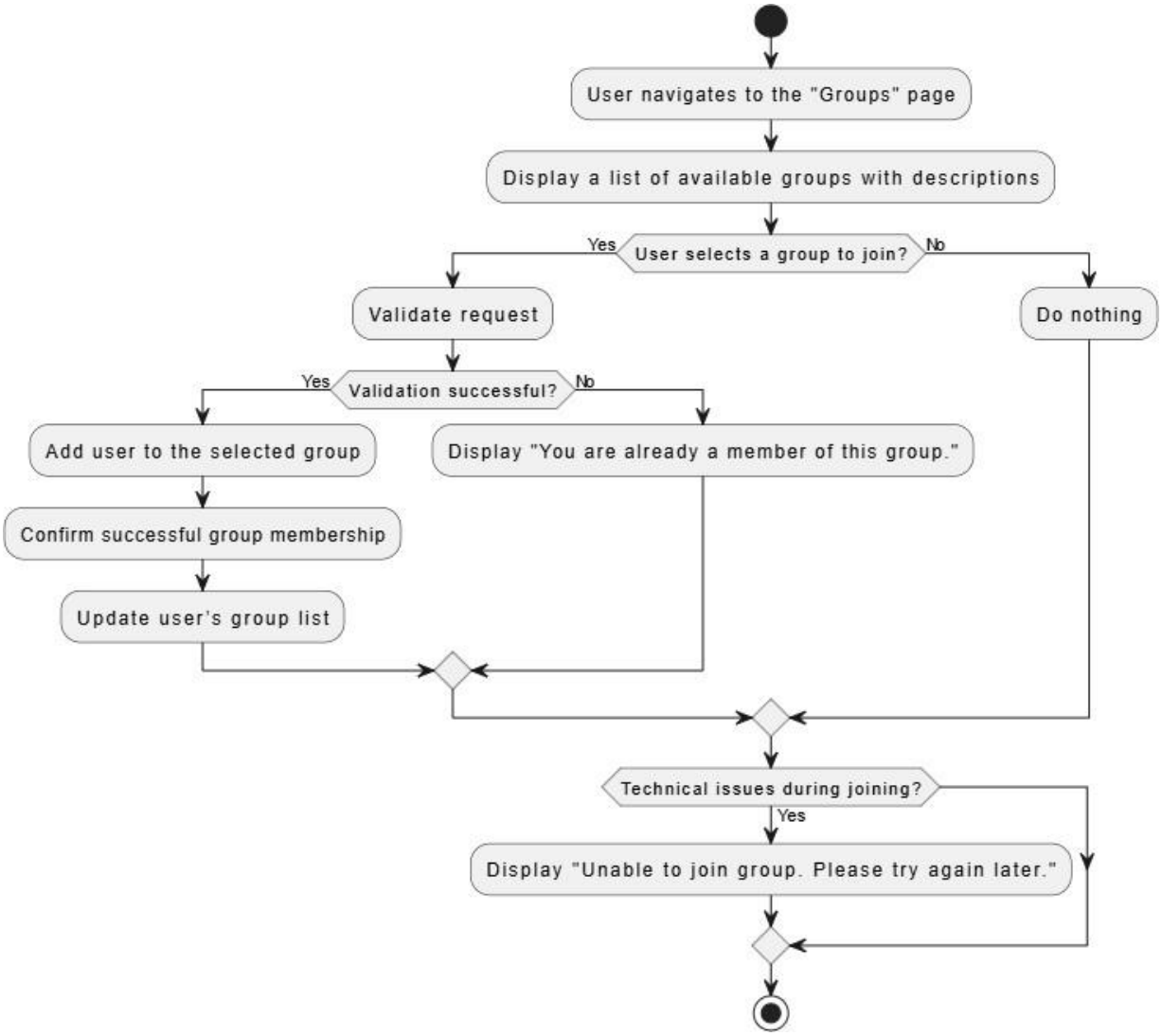

Figure 30 Join Group Activity Diagram

8.2.15Leave Group

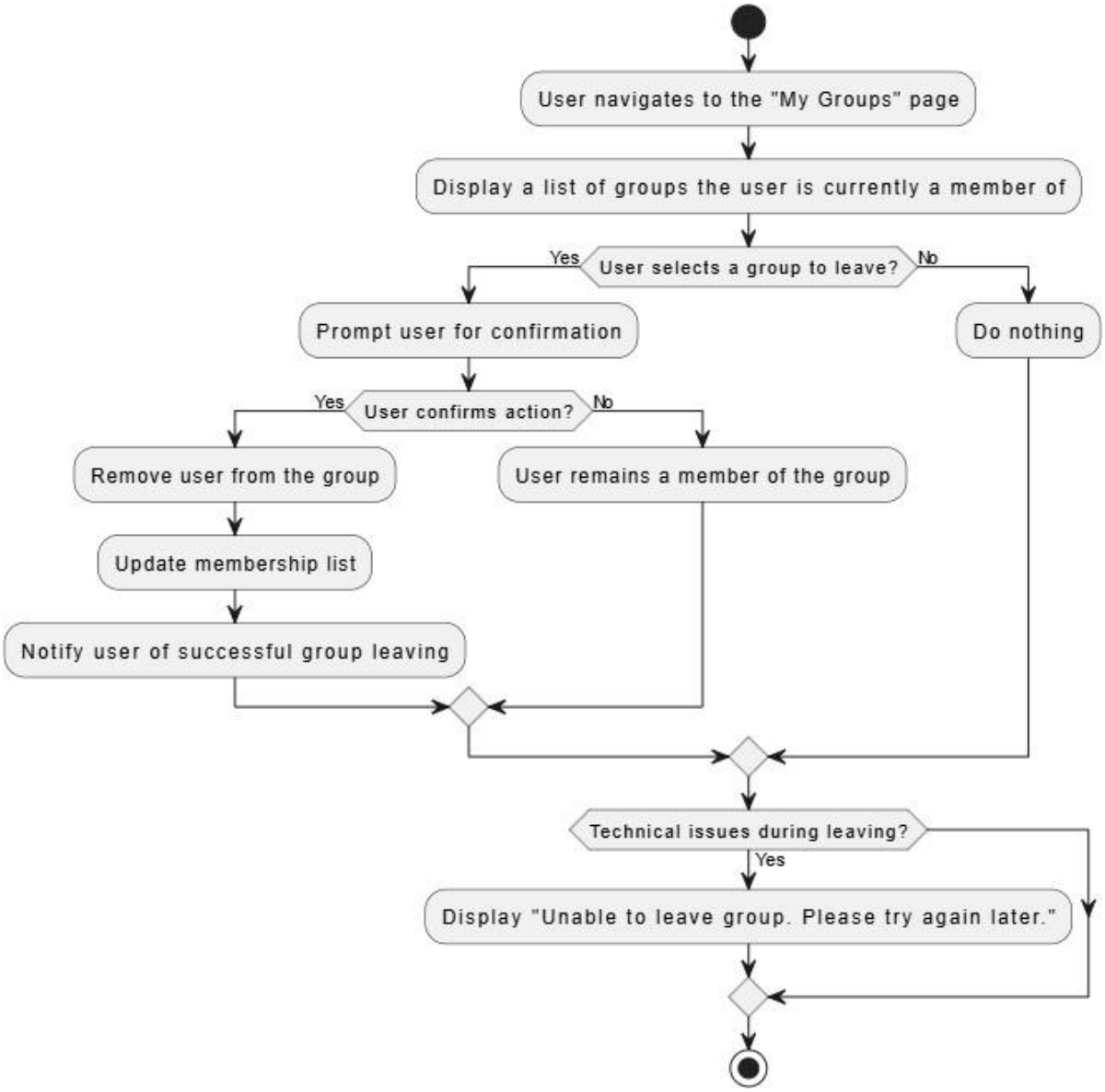

Figure 31 Leave Group Activity Diagram

8.2.16 Message Approval

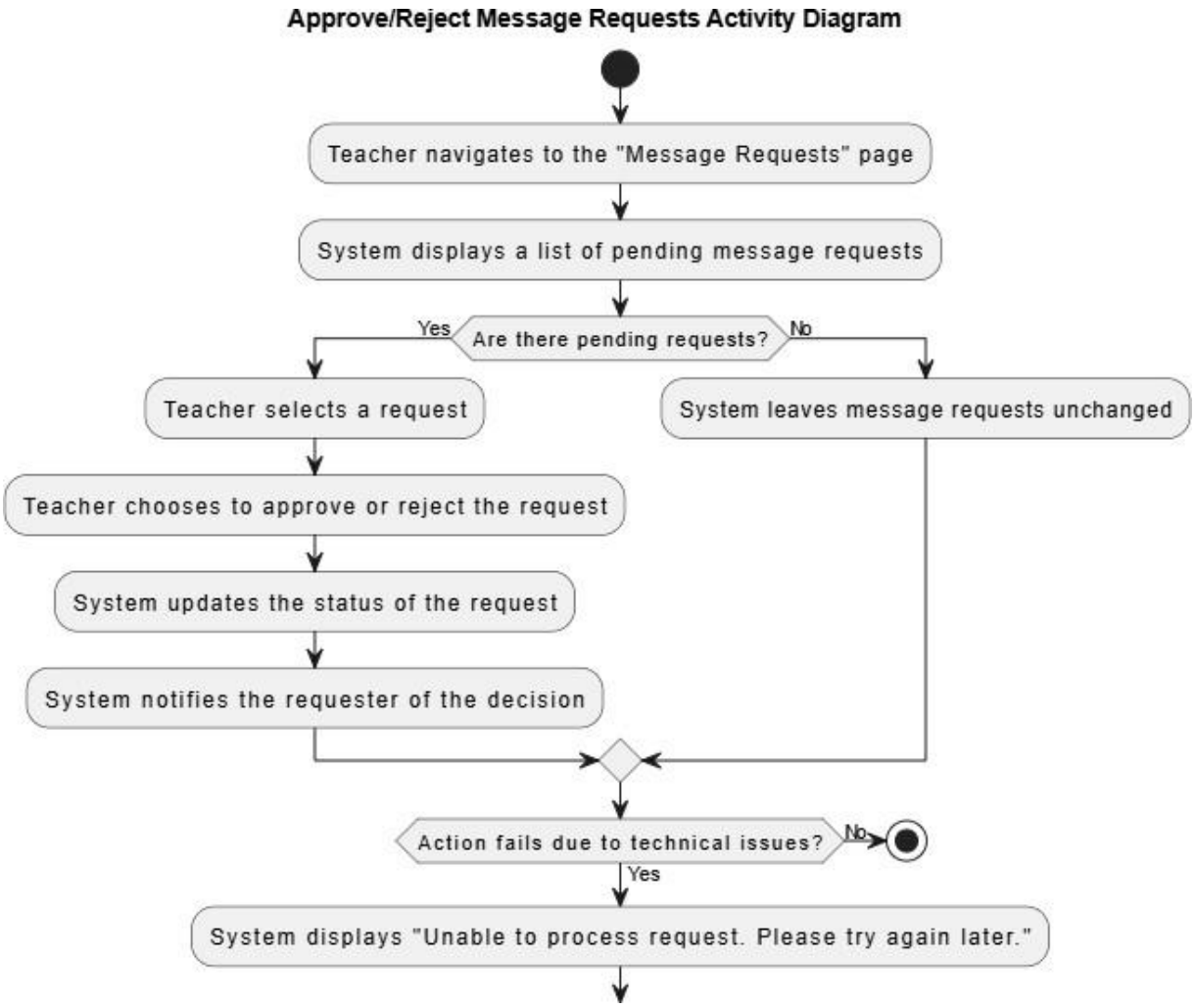

Figure 32 Message Approval Activity Diagram

8.2.17 Add Timetable

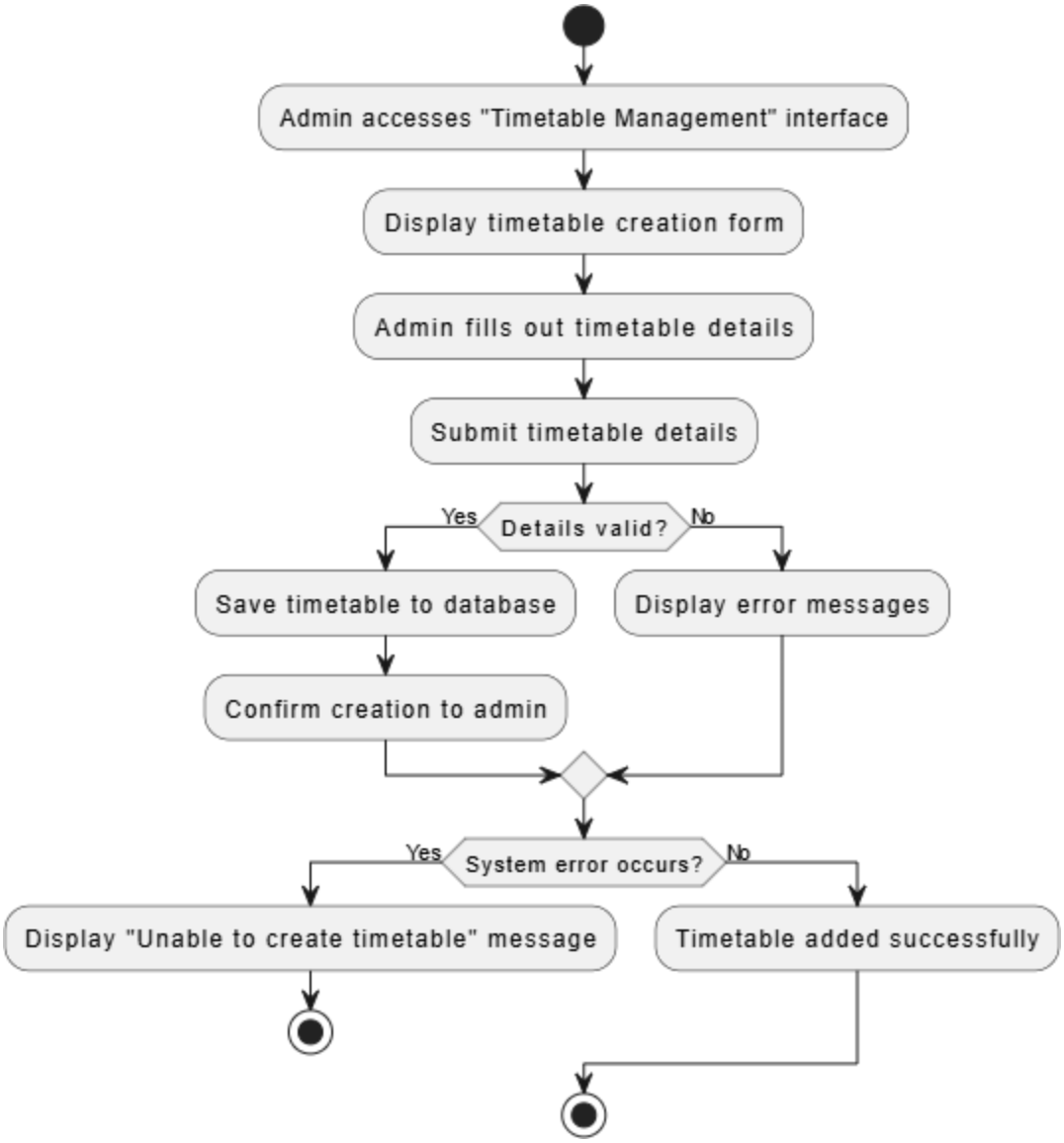

Figure 33 Add Timetable Activity Diagram

8.2.18 Update Timetable

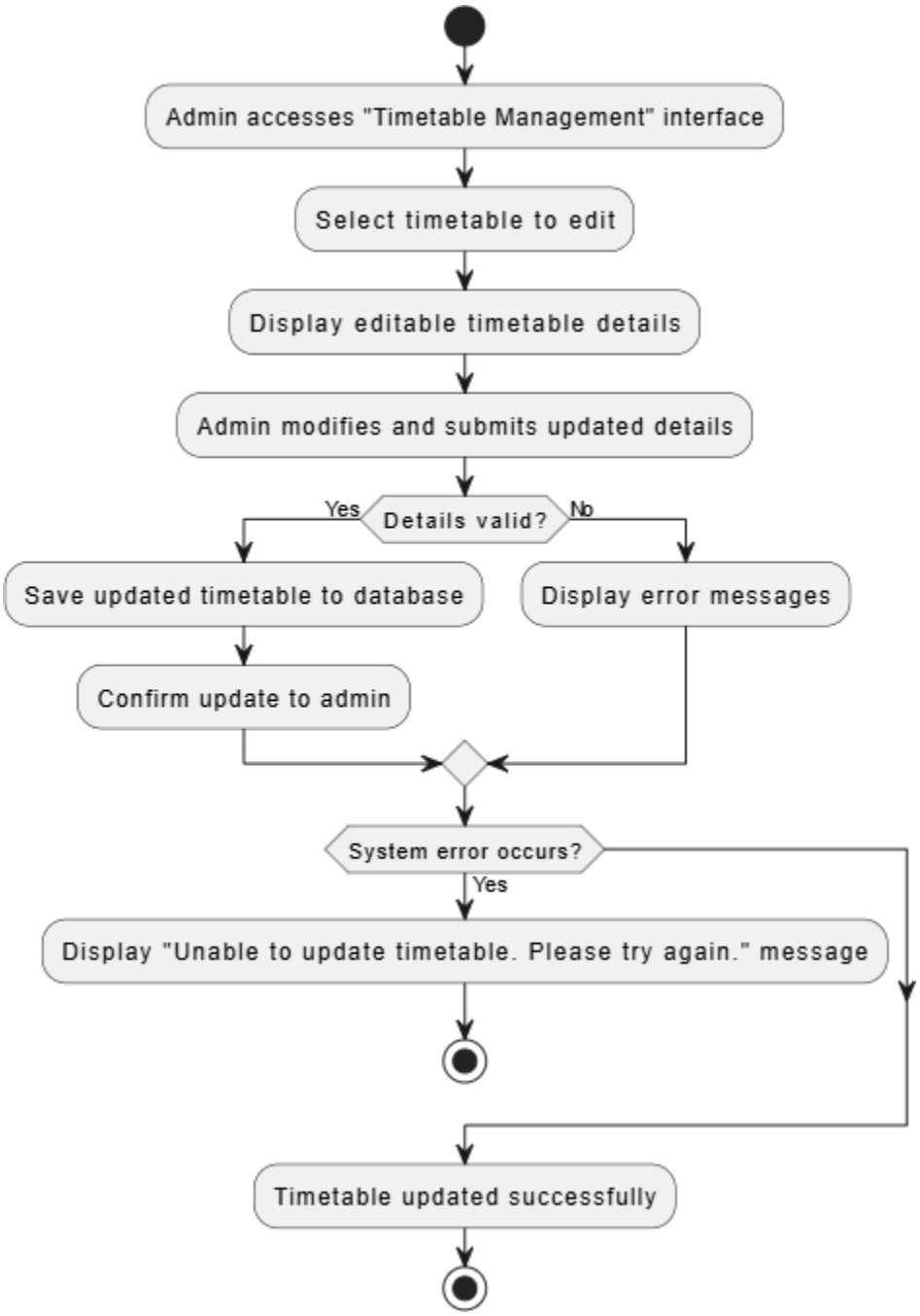

Figure 34 Update Timetable Activity Diagram

8.2.19 Delete Timetable

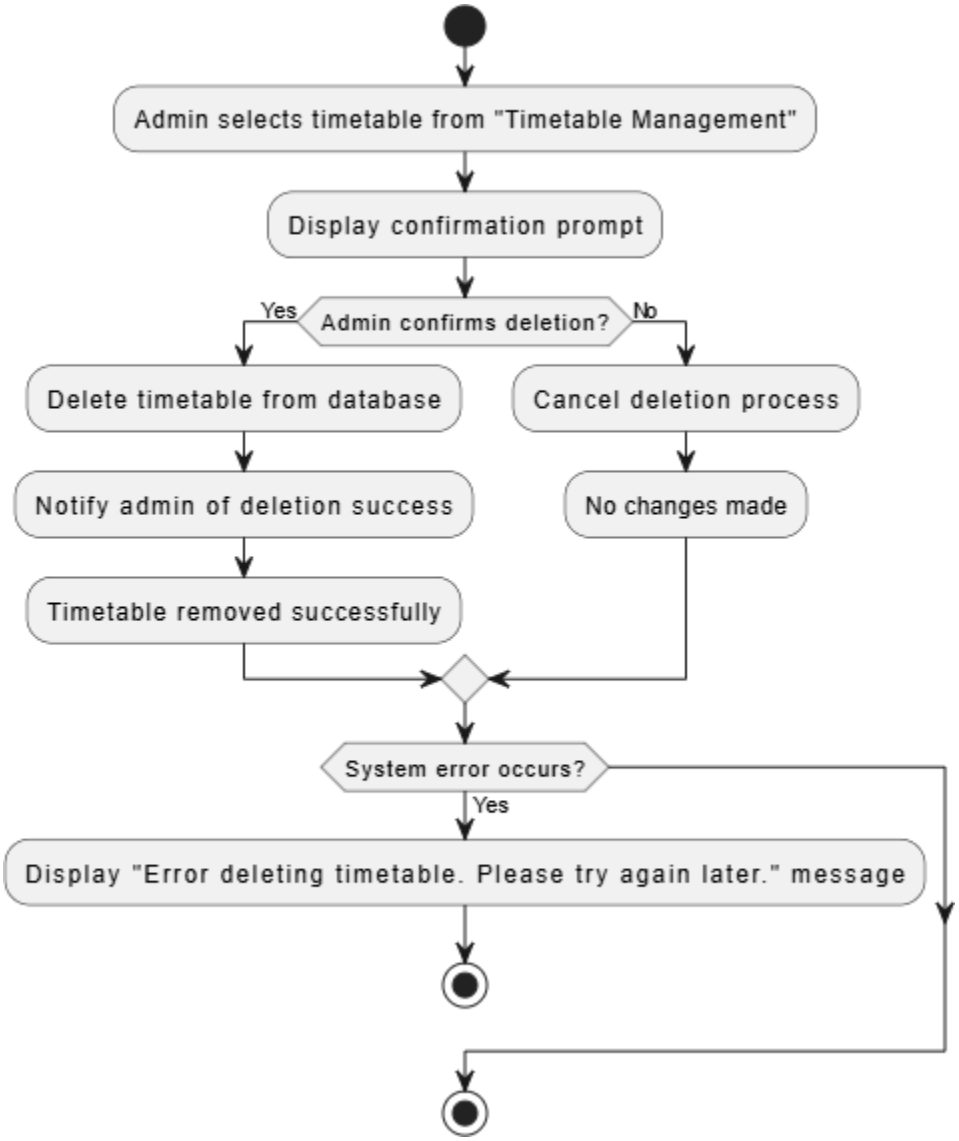

Figure 35 Delete Timetable Activity Diagram

8.2.20 Add Marketplace

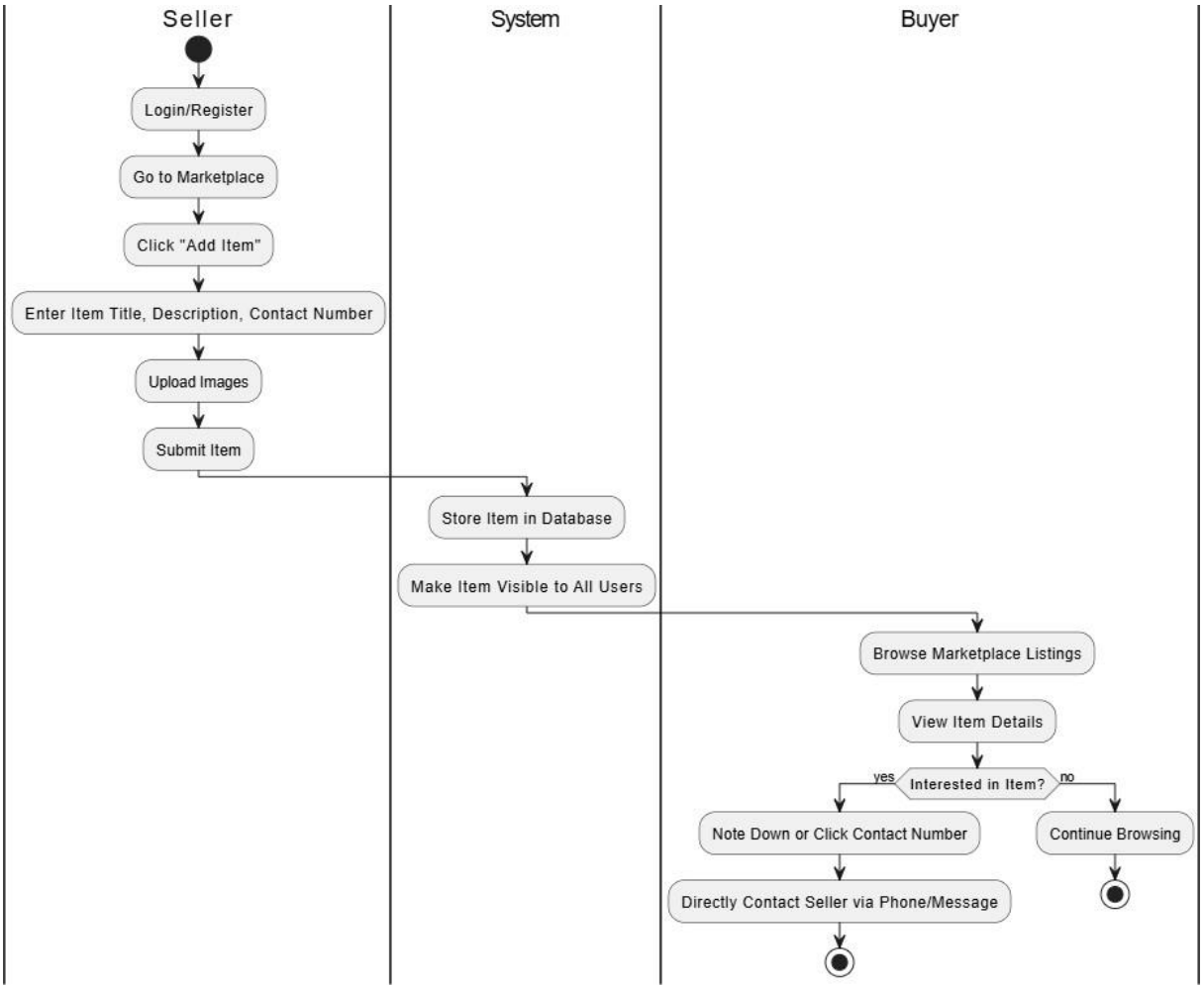

Figure 36 Add Marketplace Activity Diagram

8.2.21 Edit Marketplace

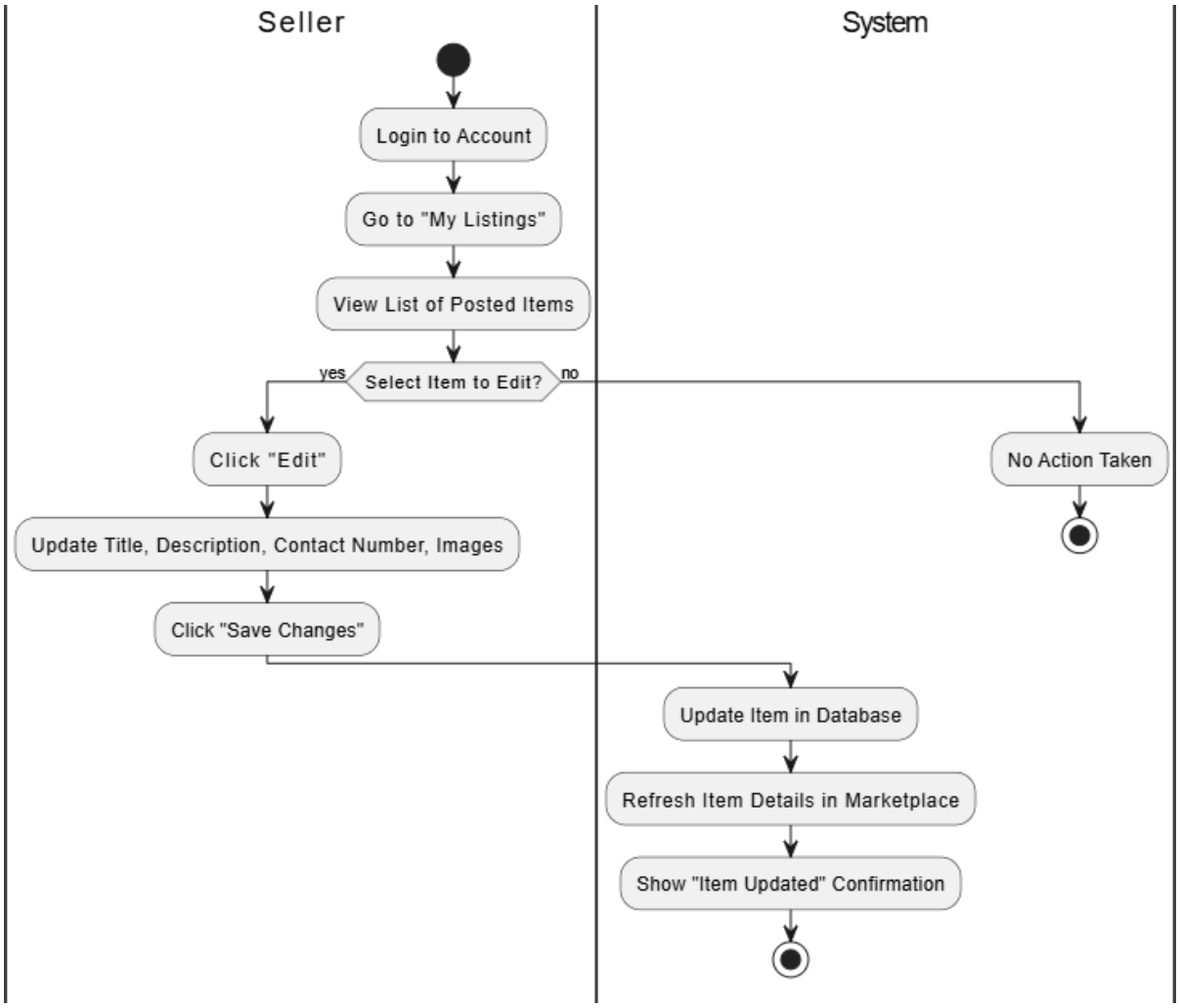

Figure 37 Edit Marketplace Activity Diagram

8.2.22 Delete Marketplace

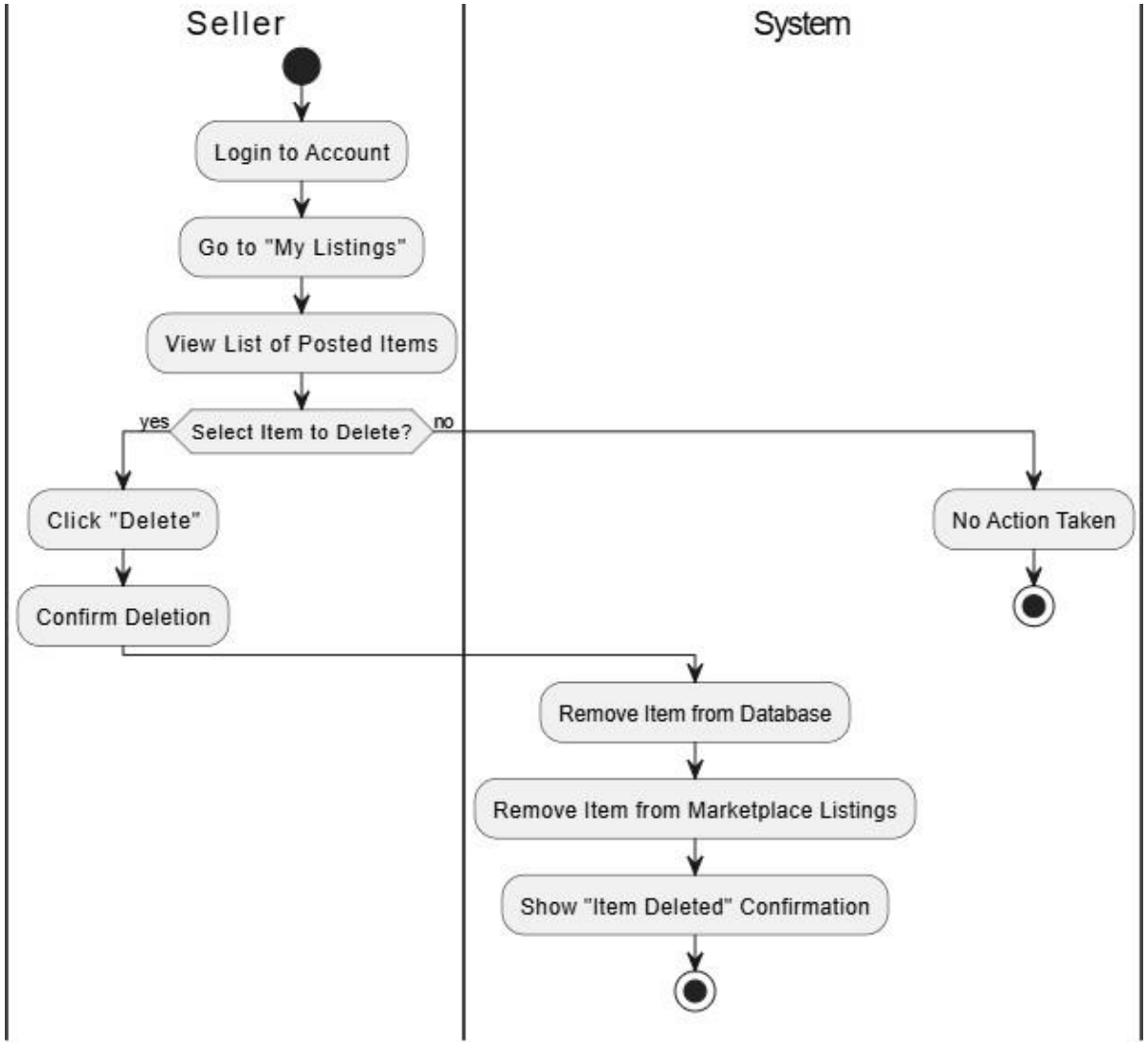

Figure 38 Delete Marketplace Activity Diagram

8.2.23 Add Carpool

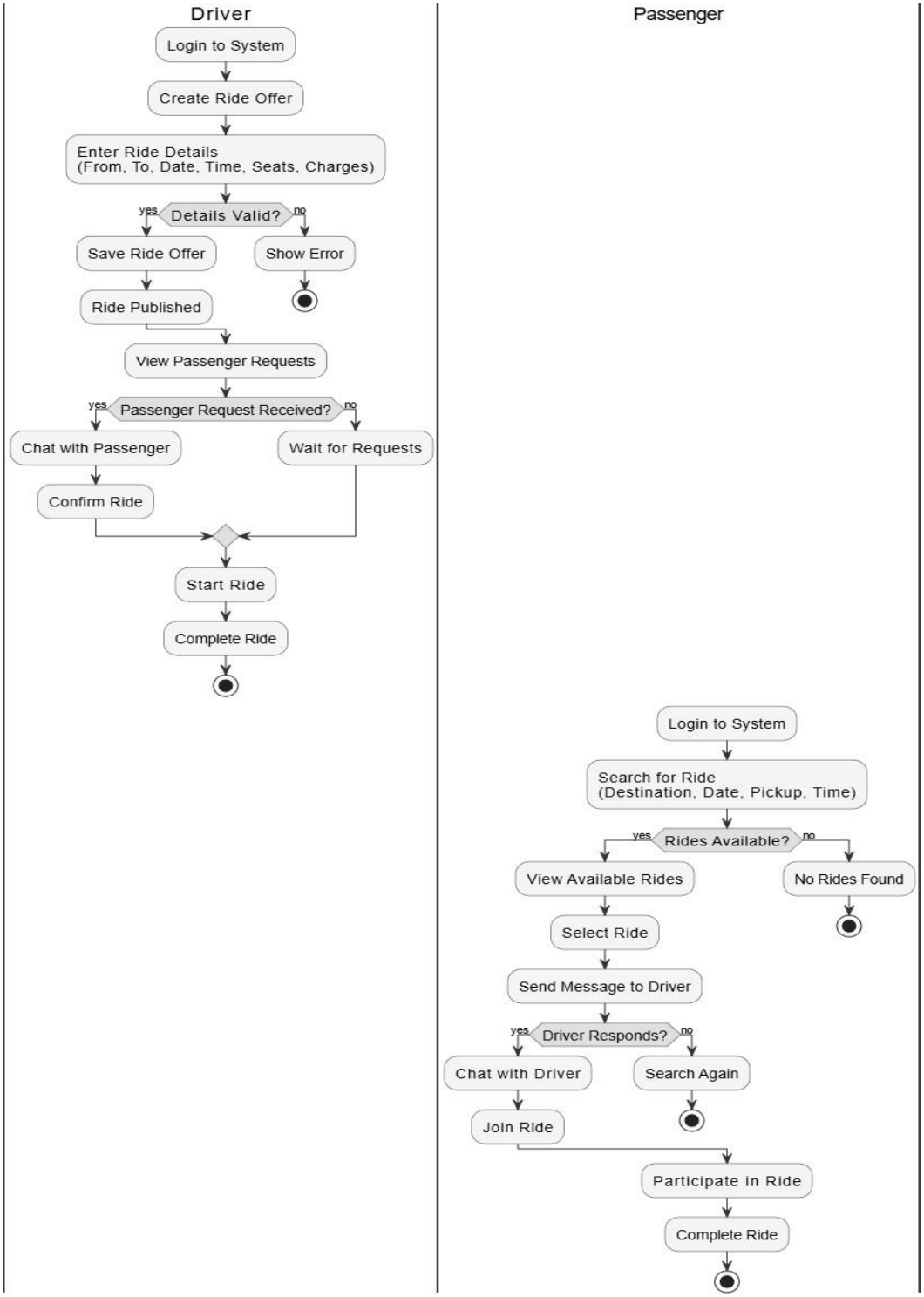

Figure 39 Add Carpool

8.2.24Delete Carpool

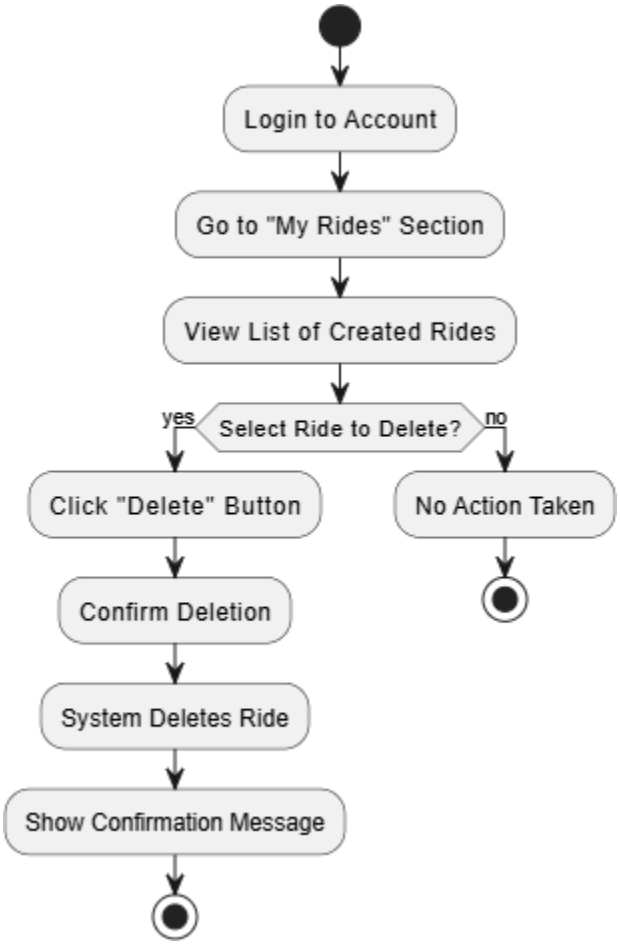

Figure 40 Delete Carpool Activity Diagram

8.2.25 Edit Carpool

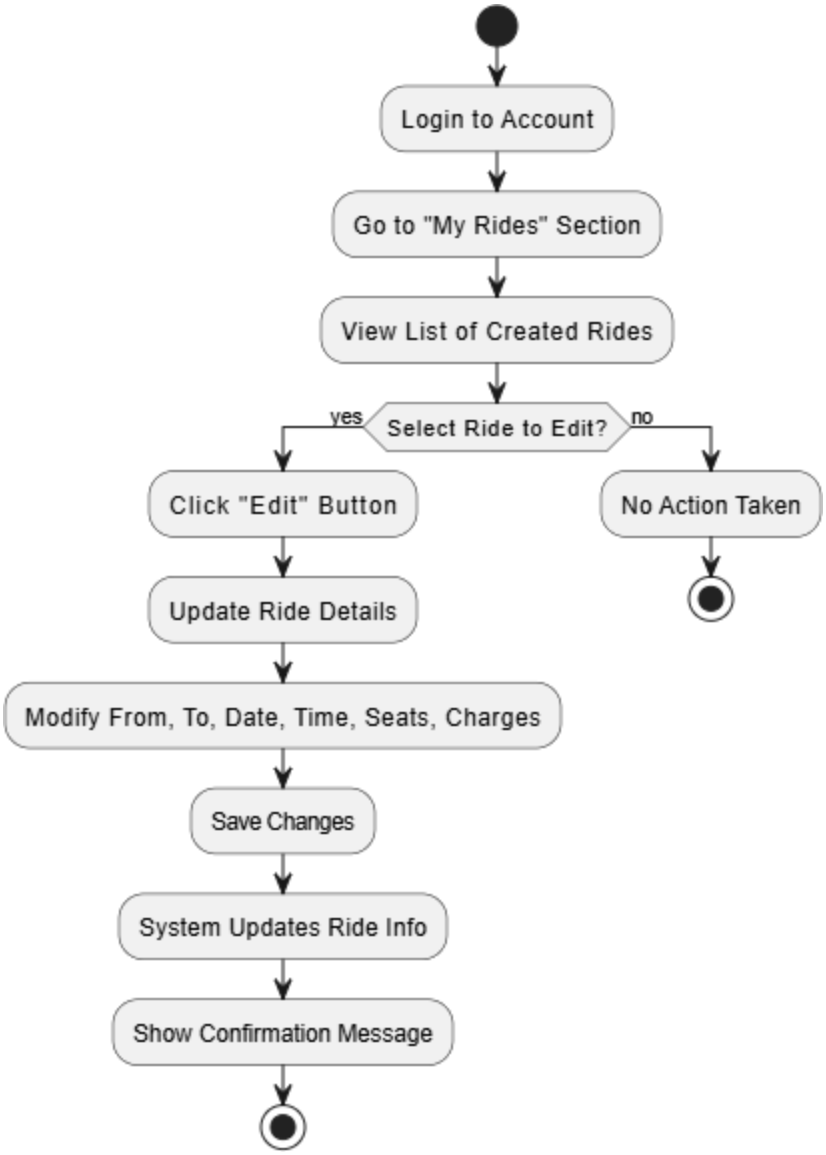

Figure 41 Edit Carpool Activity Diagram

8.2.26 Add To-Do List

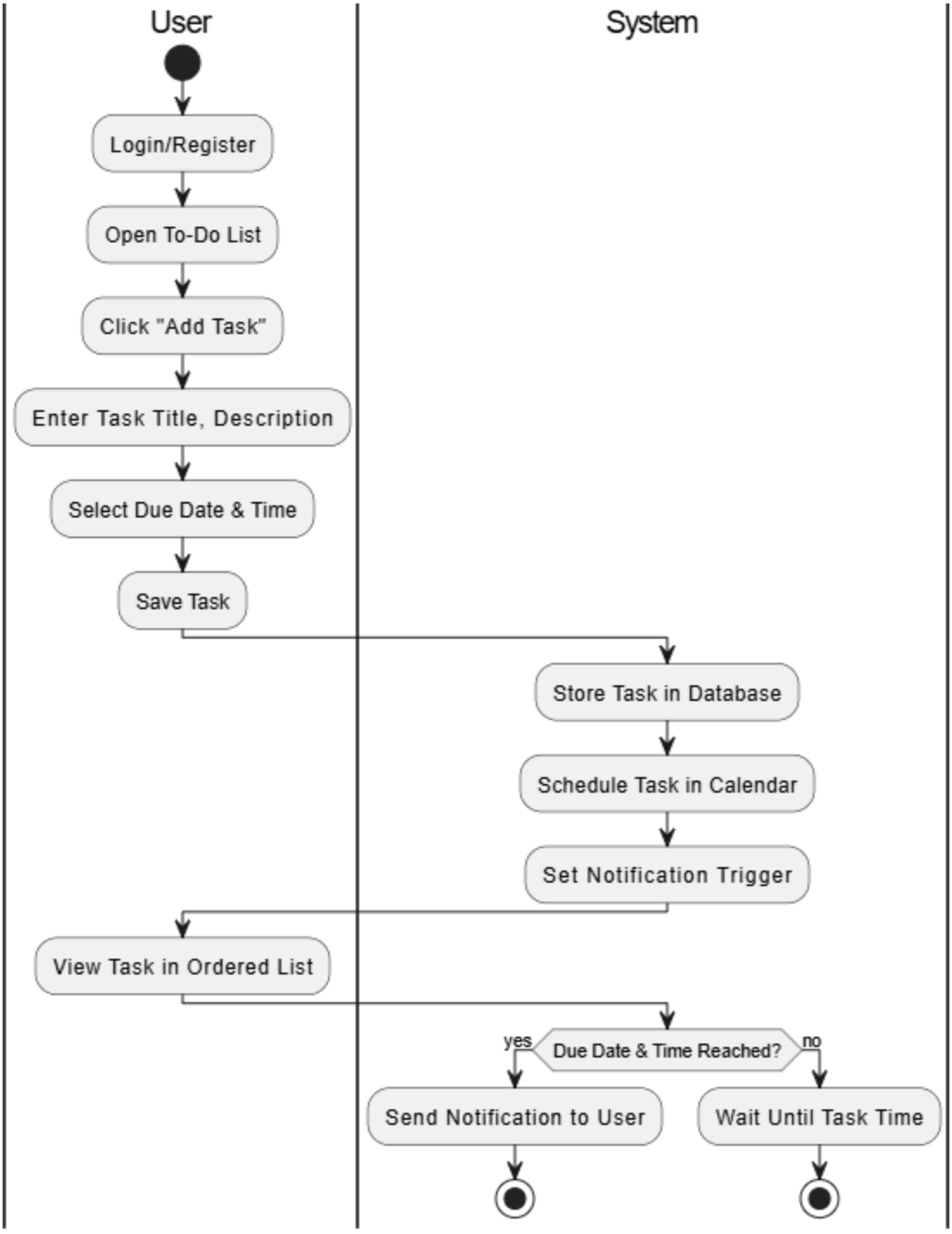

Figure 42 Add To-Do List Activity Diagram

8.2.27Edit To-Do List

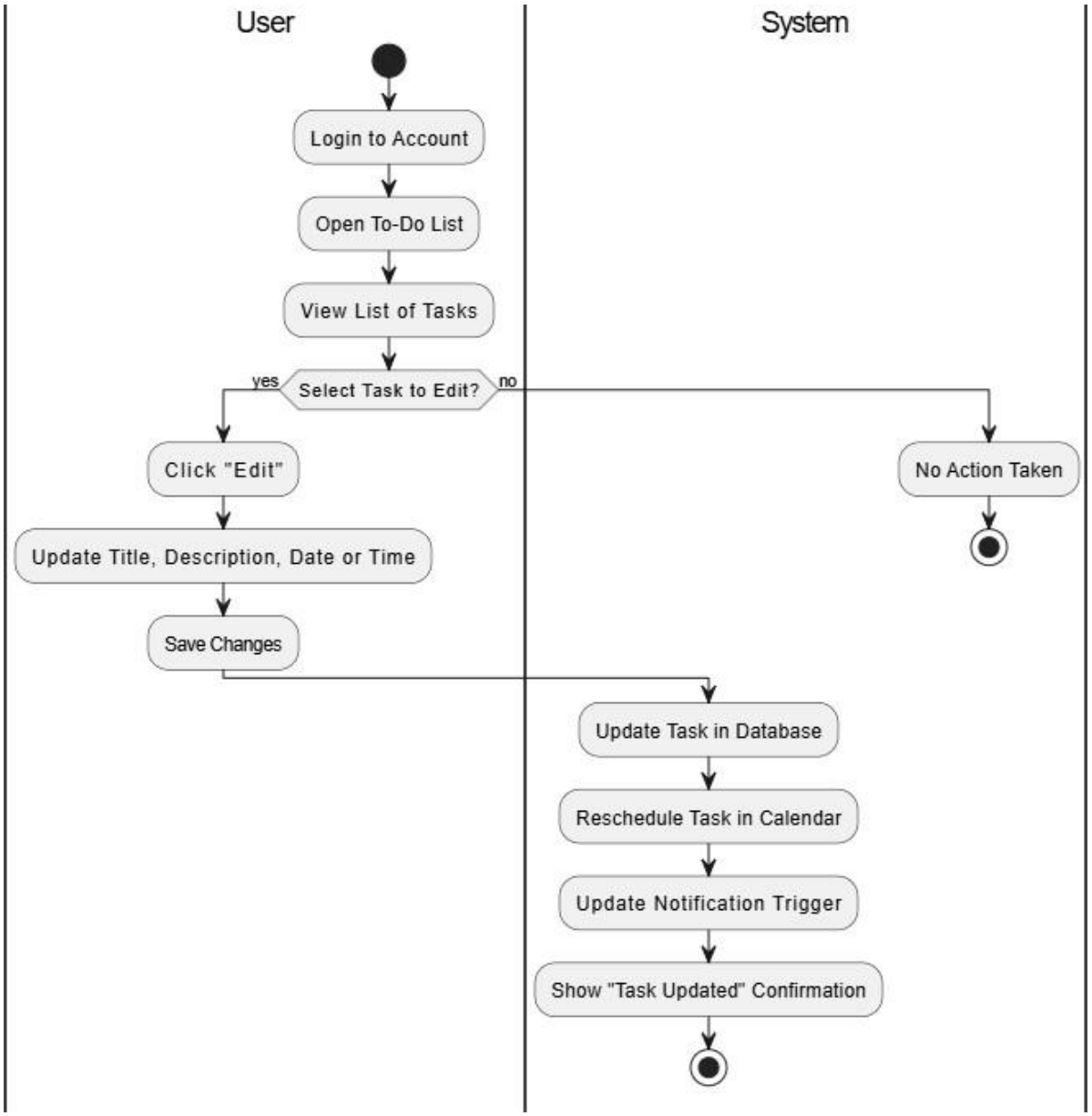

Figure 43 Edit TO DO-List Activity Diagram

8.2.28 Delete To-Do List

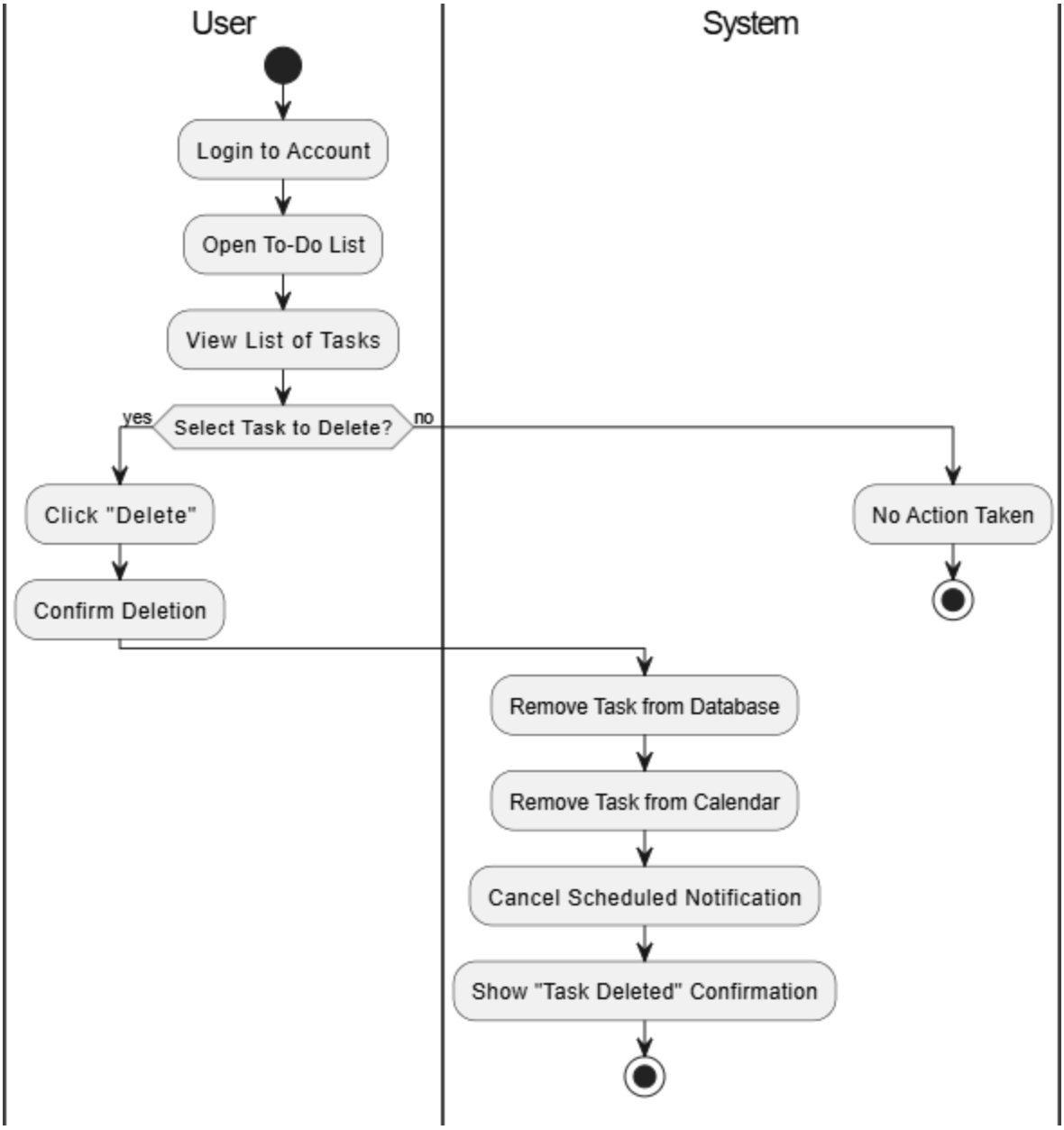

Figure 44 Delete To-Do List Activity Diagram

8.2.29Lost and Found

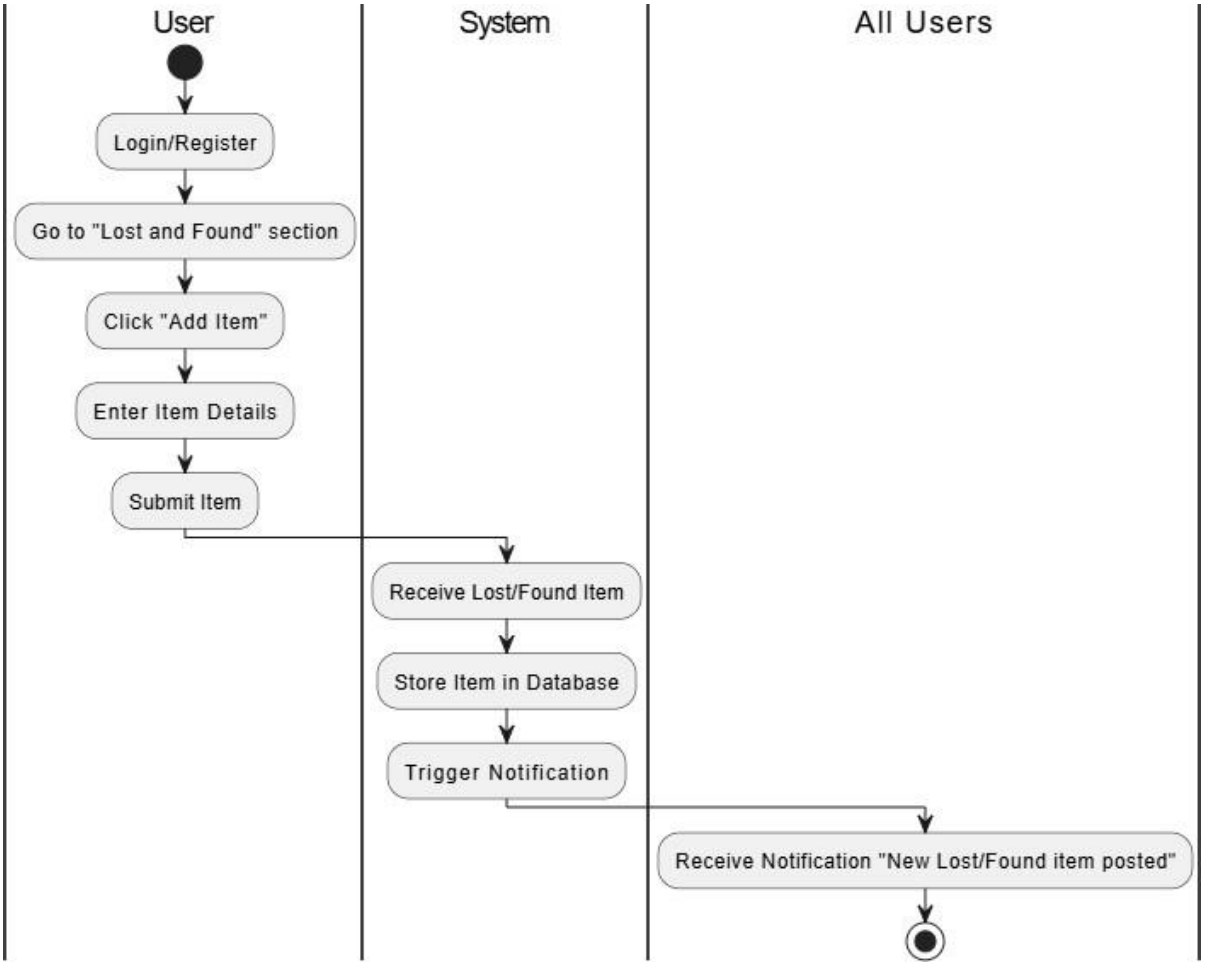

Figure 45 Lost and Found Activity Diagram

8.2.30 Resource Management

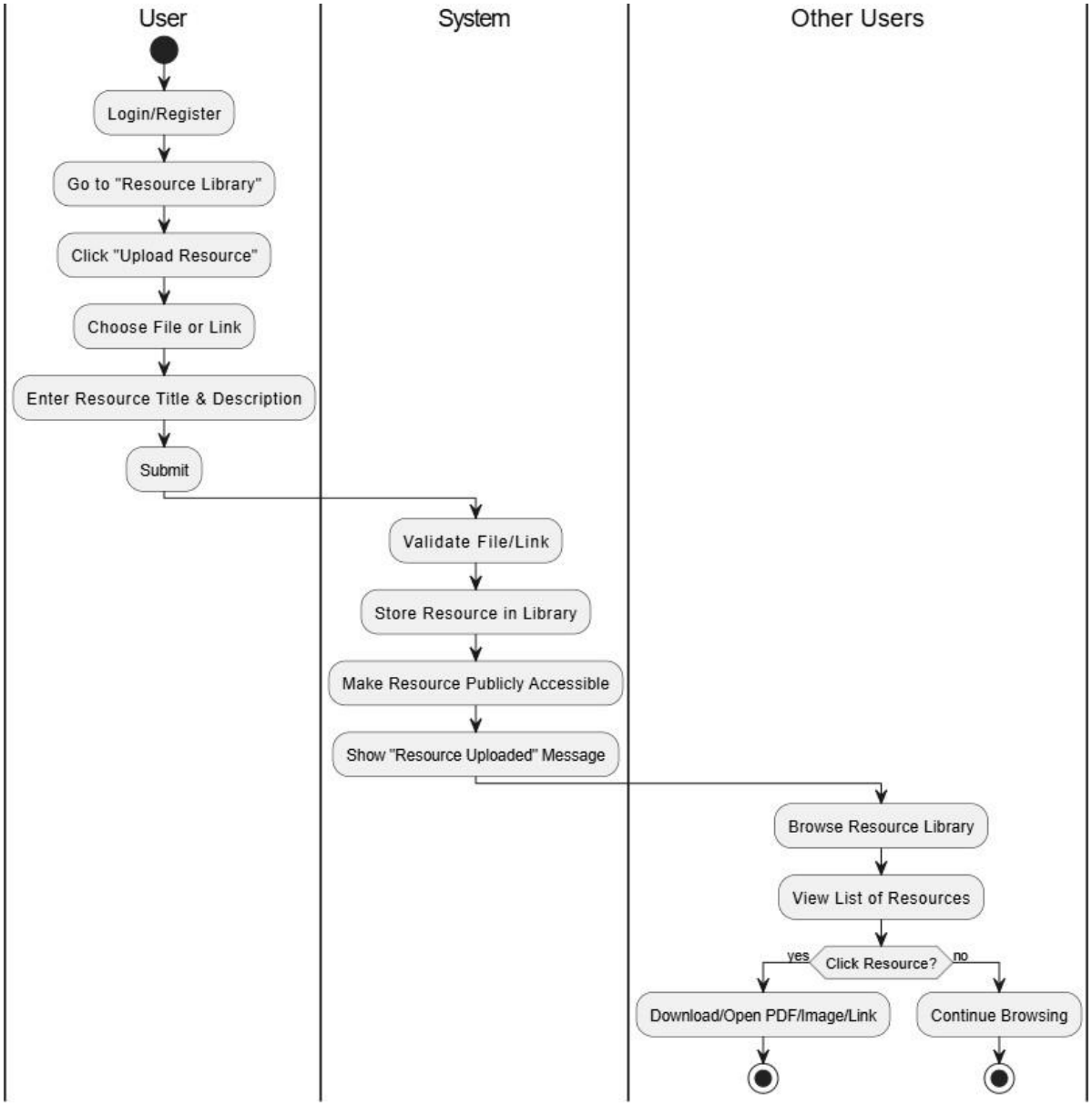

Figure 46 Resource Management Activity Diagram

8.3 Sequence Diagrams

8.3.1 Login

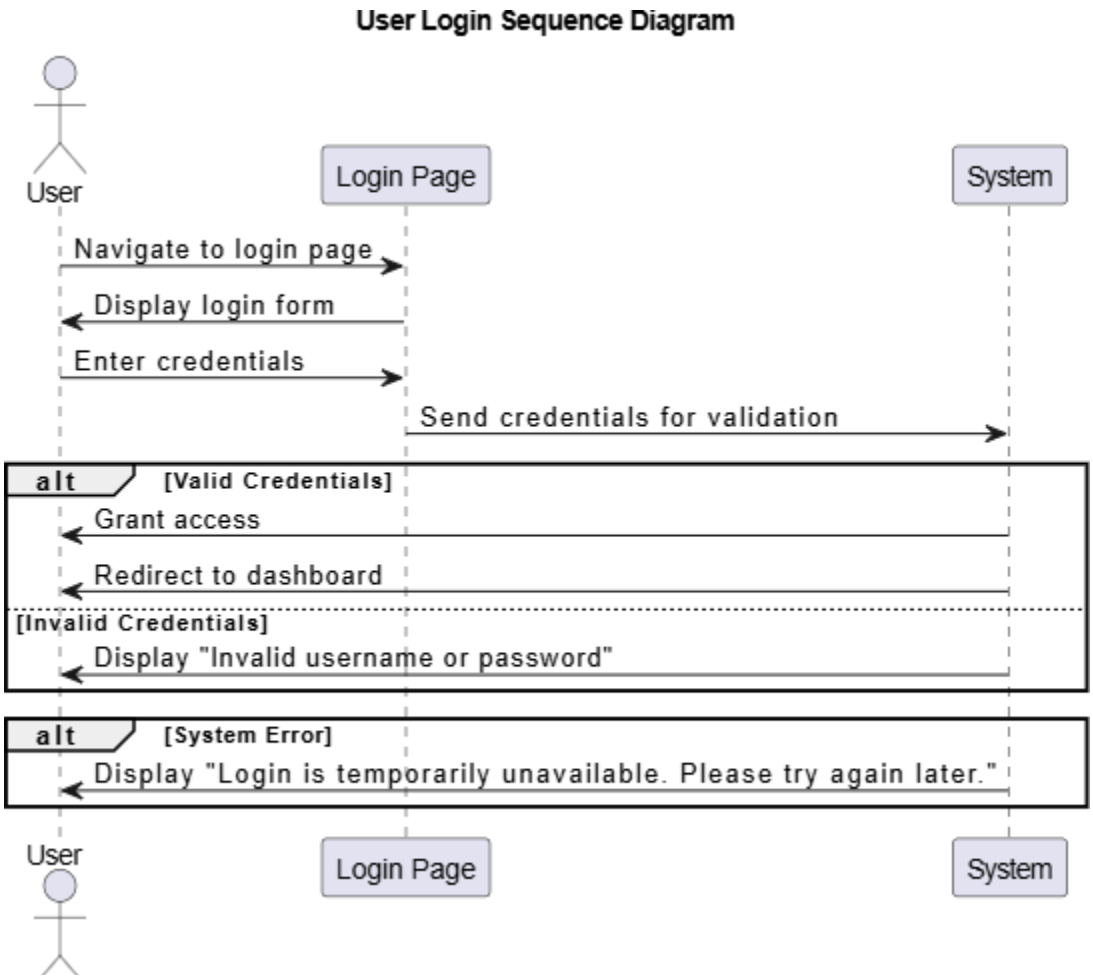

Figure 47 Login Sequence Diagram

8.3.2 Signup

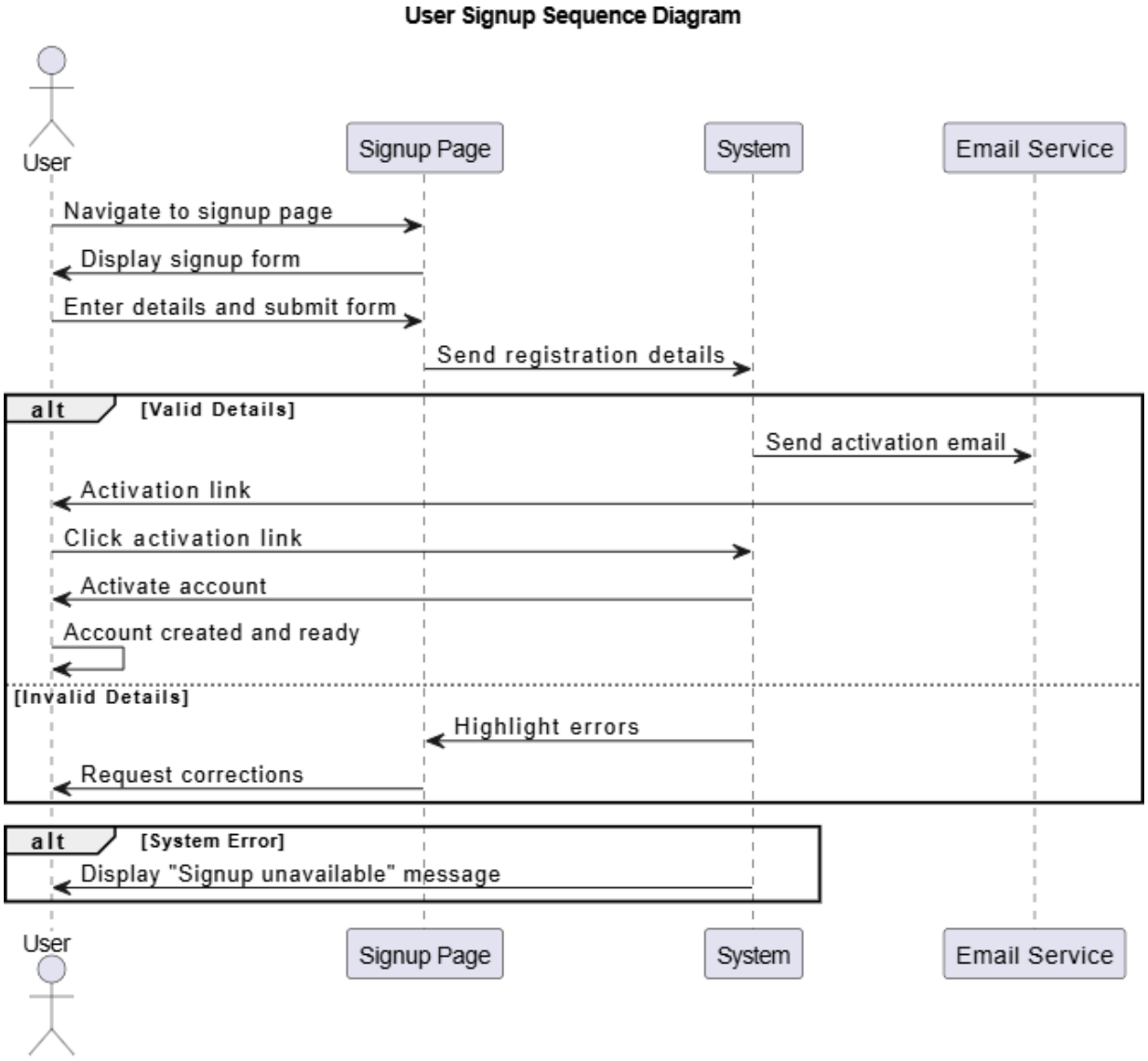

Figure 48 Sign-Up Sequence Diagram

8.3.3 Forget Password

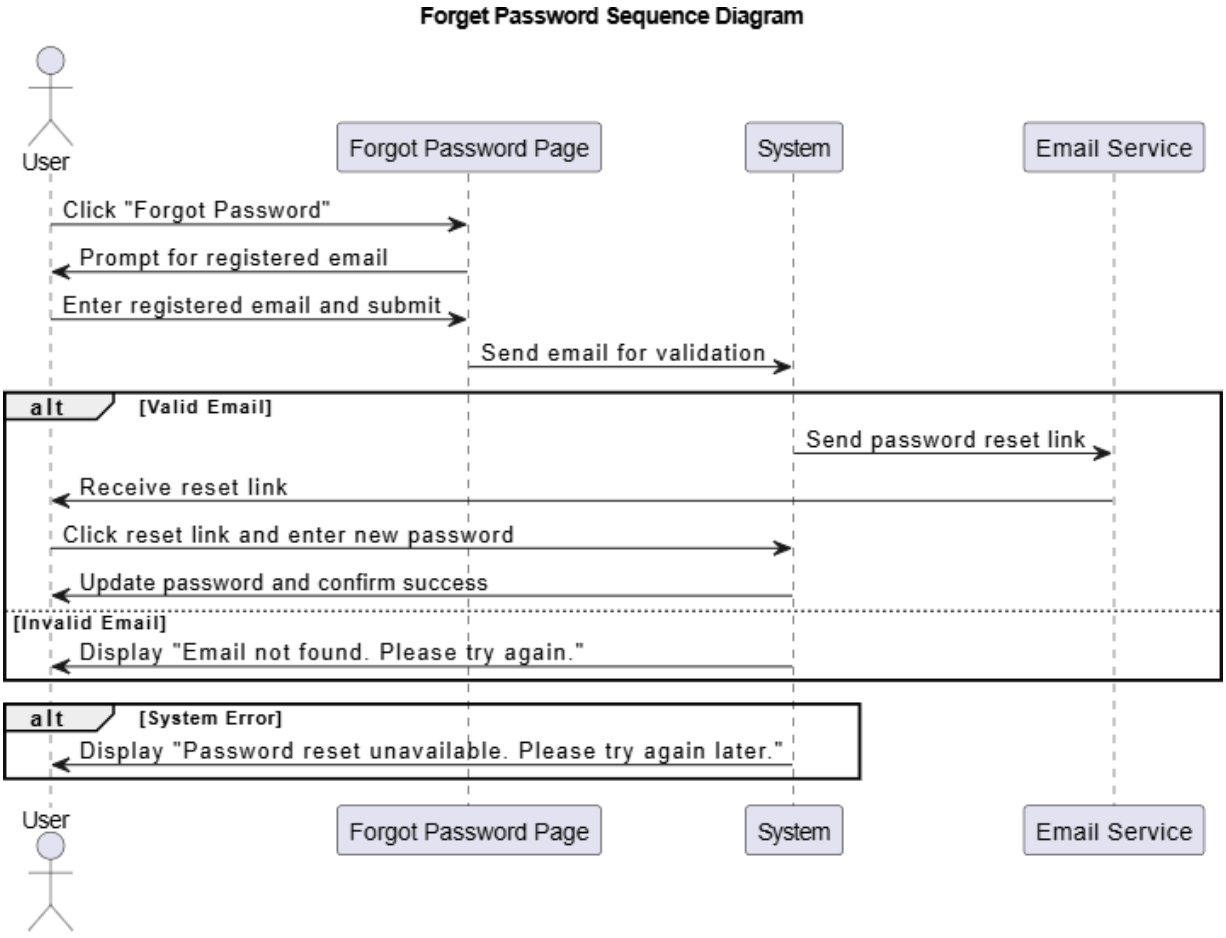

Figure 49 Forget Password Sequence Diagram

8.3.4 Add Event

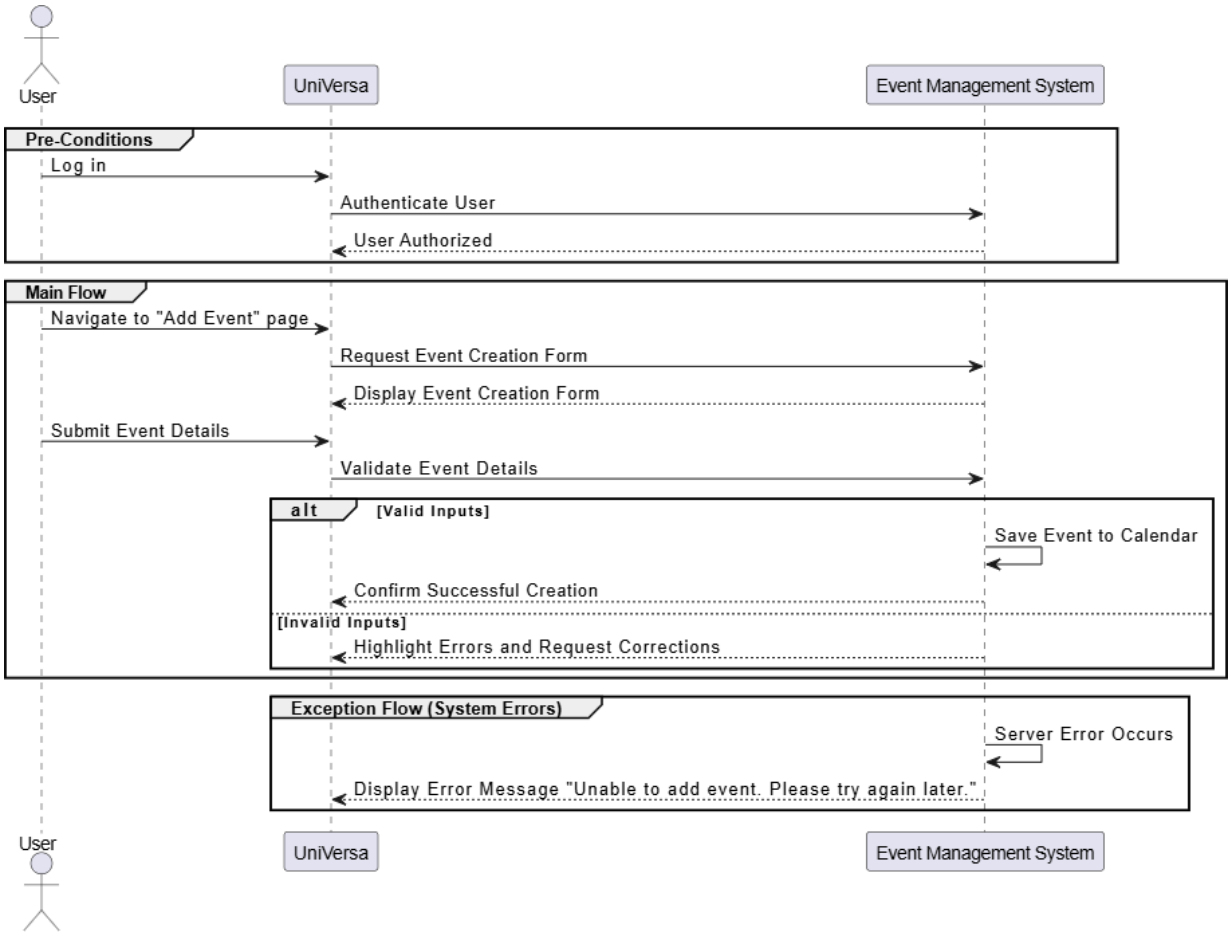

Figure 50 Add Event Sequence Diagram

8.3.5 Edit Event

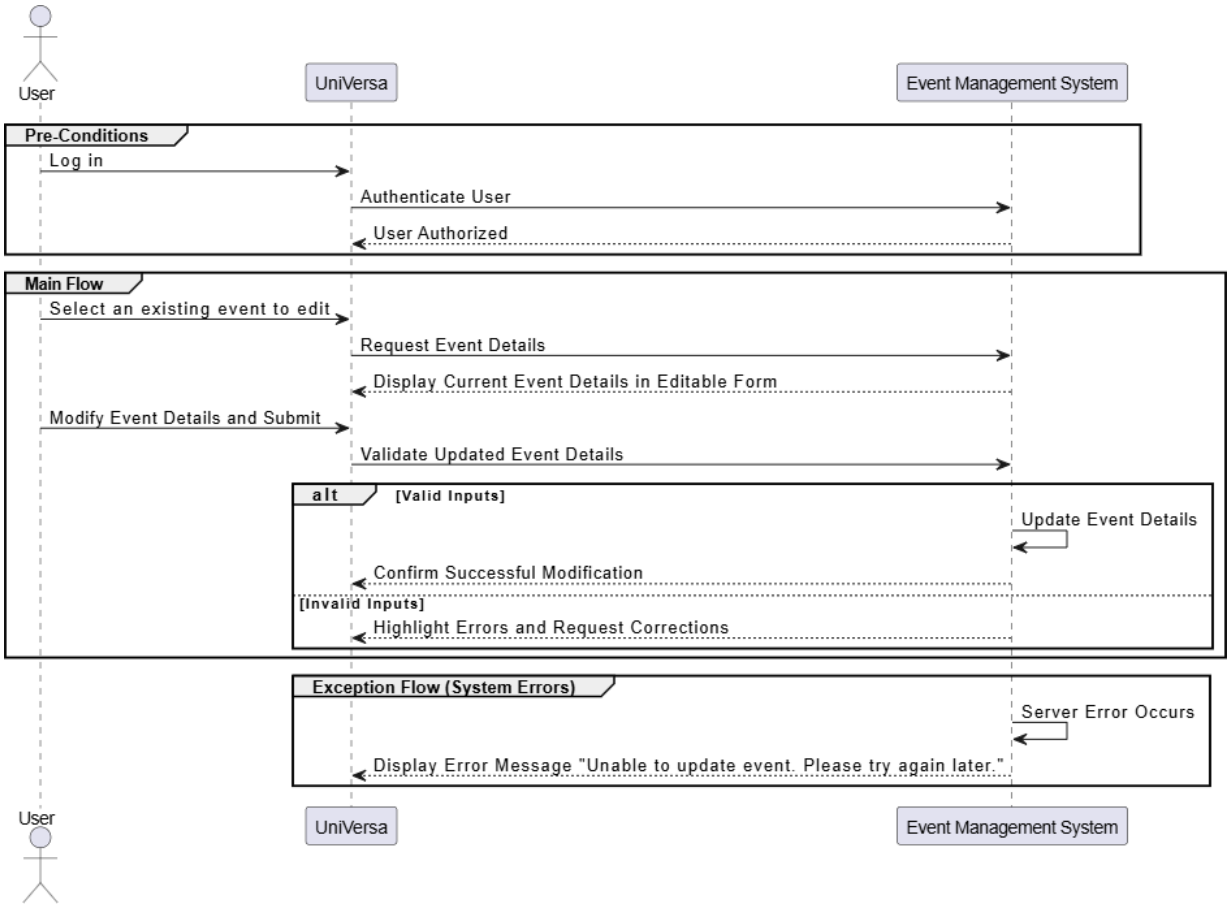

Figure 51 Edit Event Sequence Diagram

8.3.6 View Event

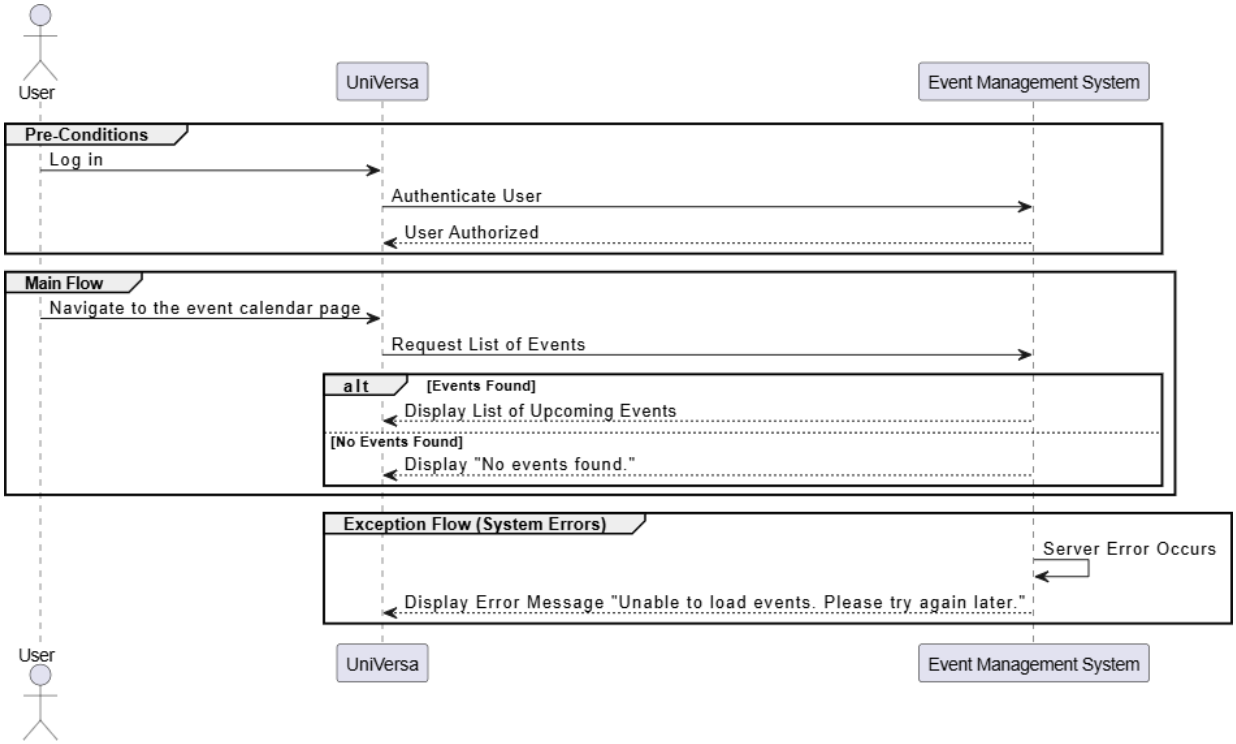

Figure 52 View Edit Sequence Diagram

8.3.7 Remove Event

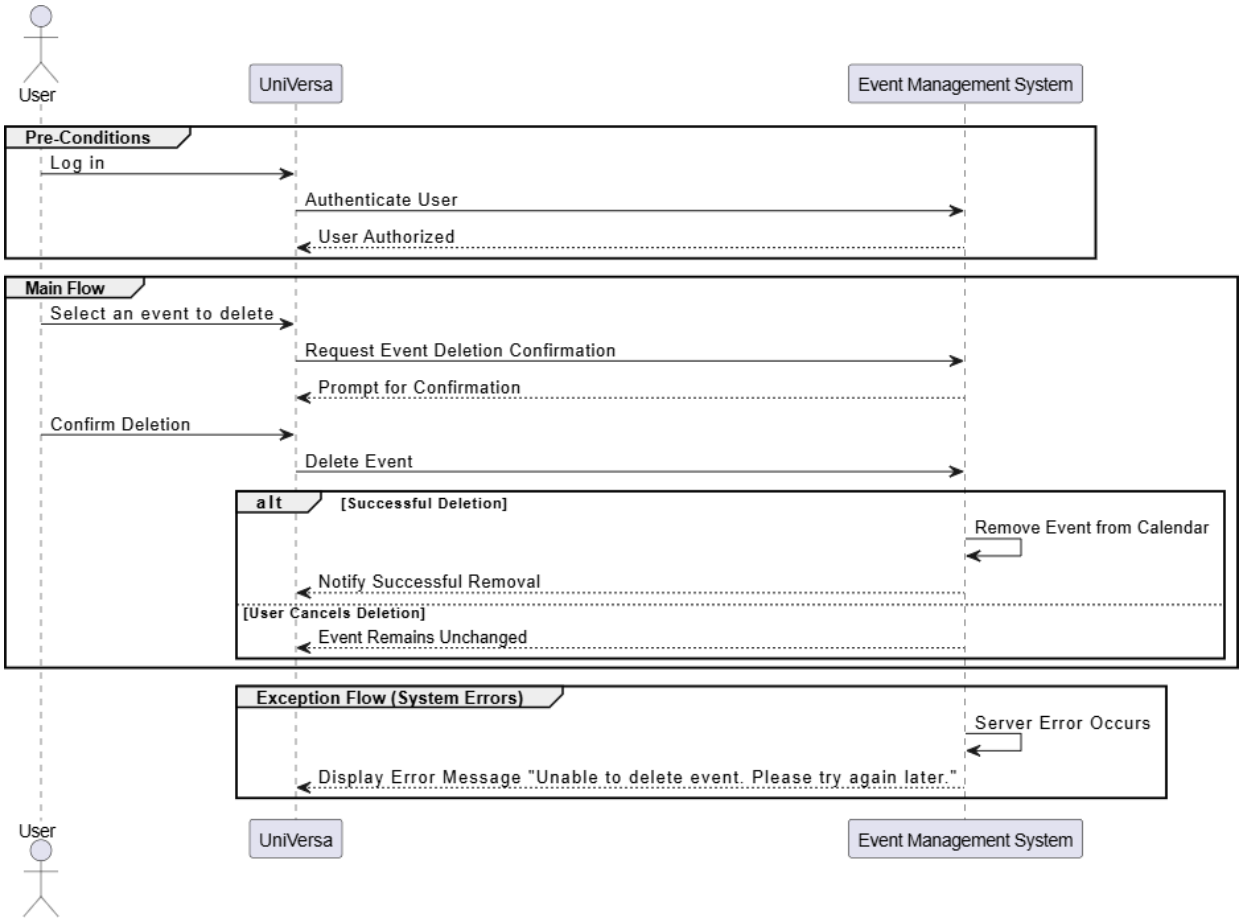

Figure 53 Remove Event Sequence Diagram

8.3.8 Post Job

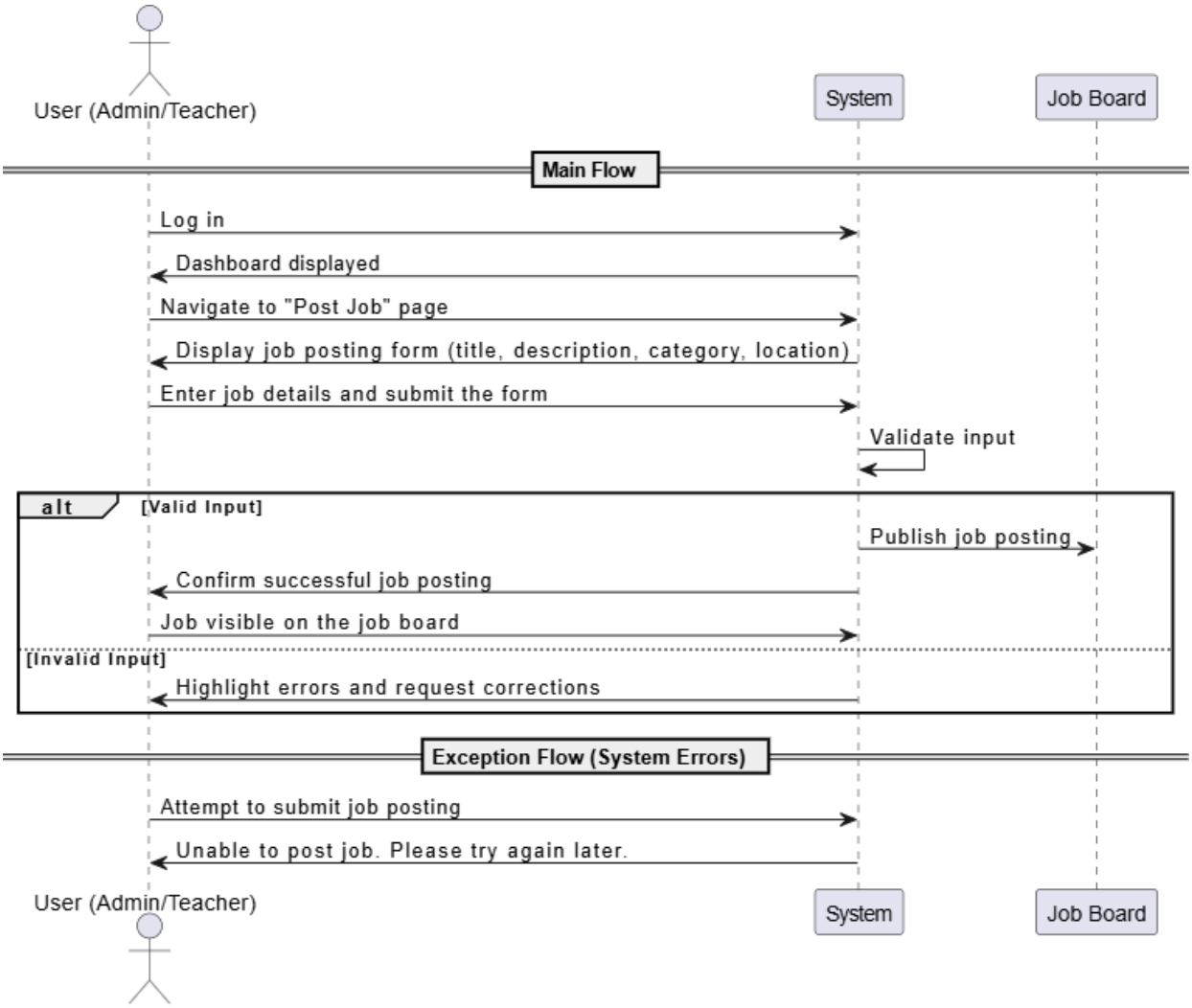

Figure 54 Post Job Sequence Diagram

8.3.9 Edit Job

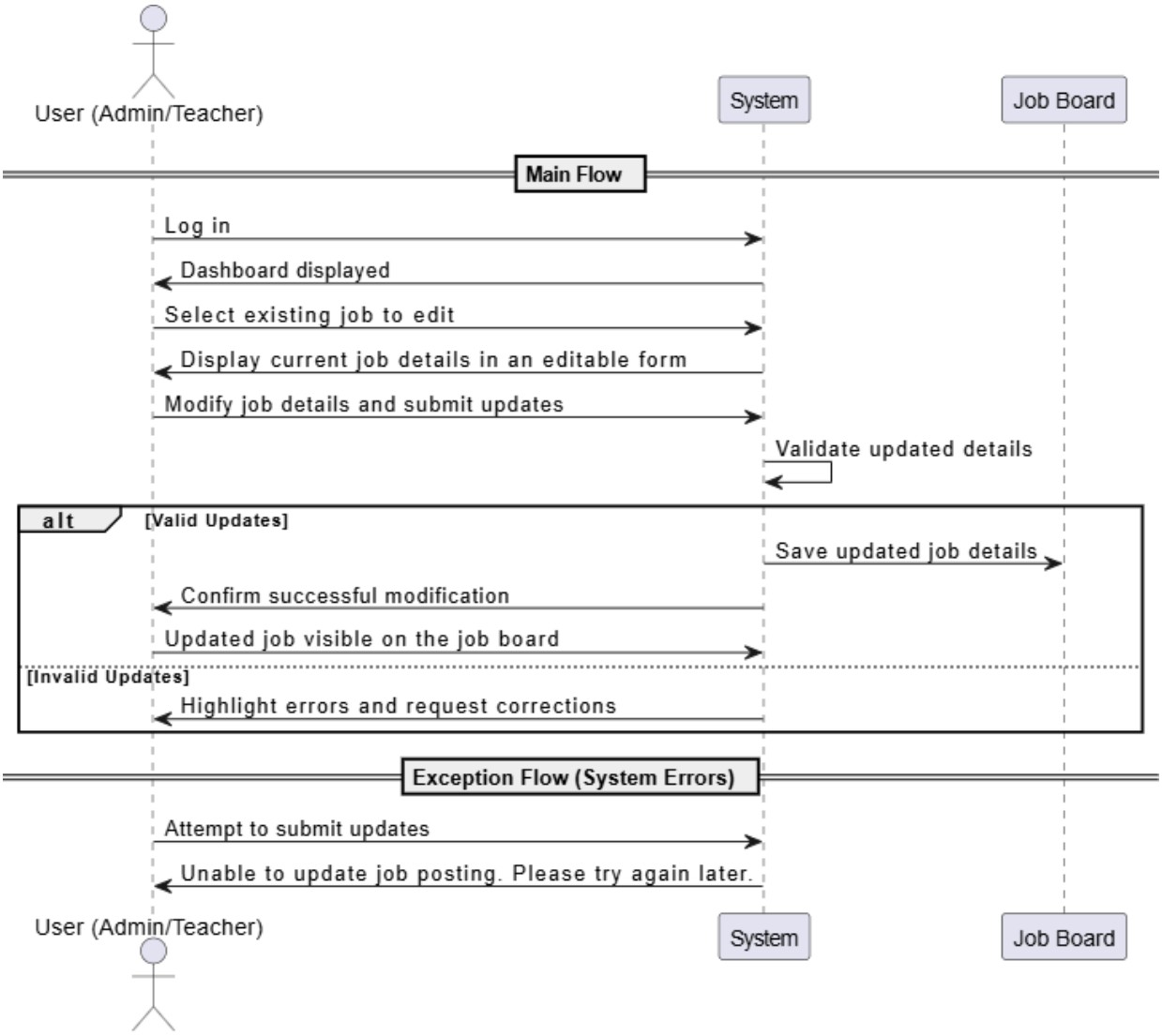

Figure 55 Edit Job Sequence Diagram

8.3.10 Remove Job

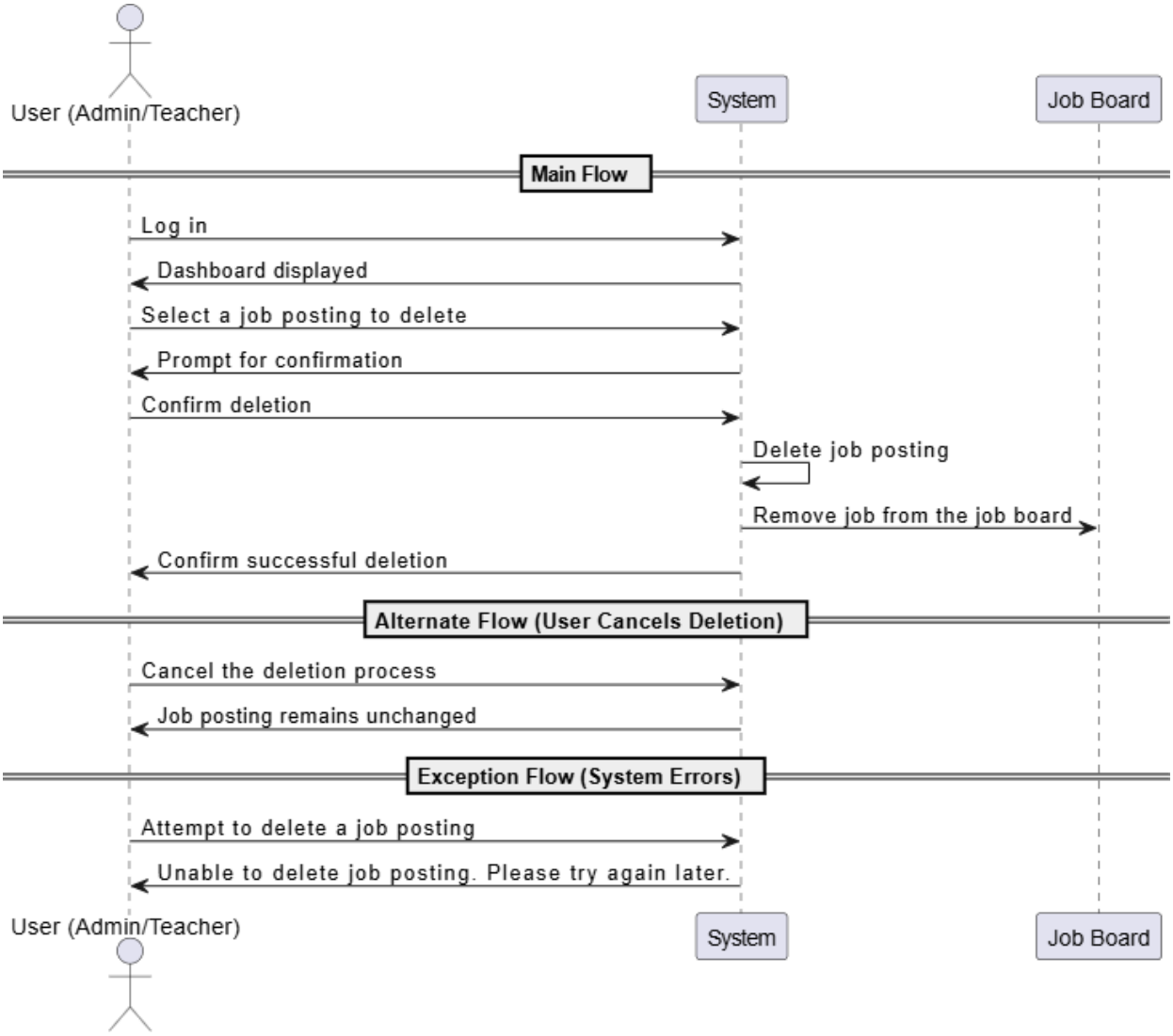

Figure 56 Remove Job Sequence Diagram

8.3.11 Manage Job

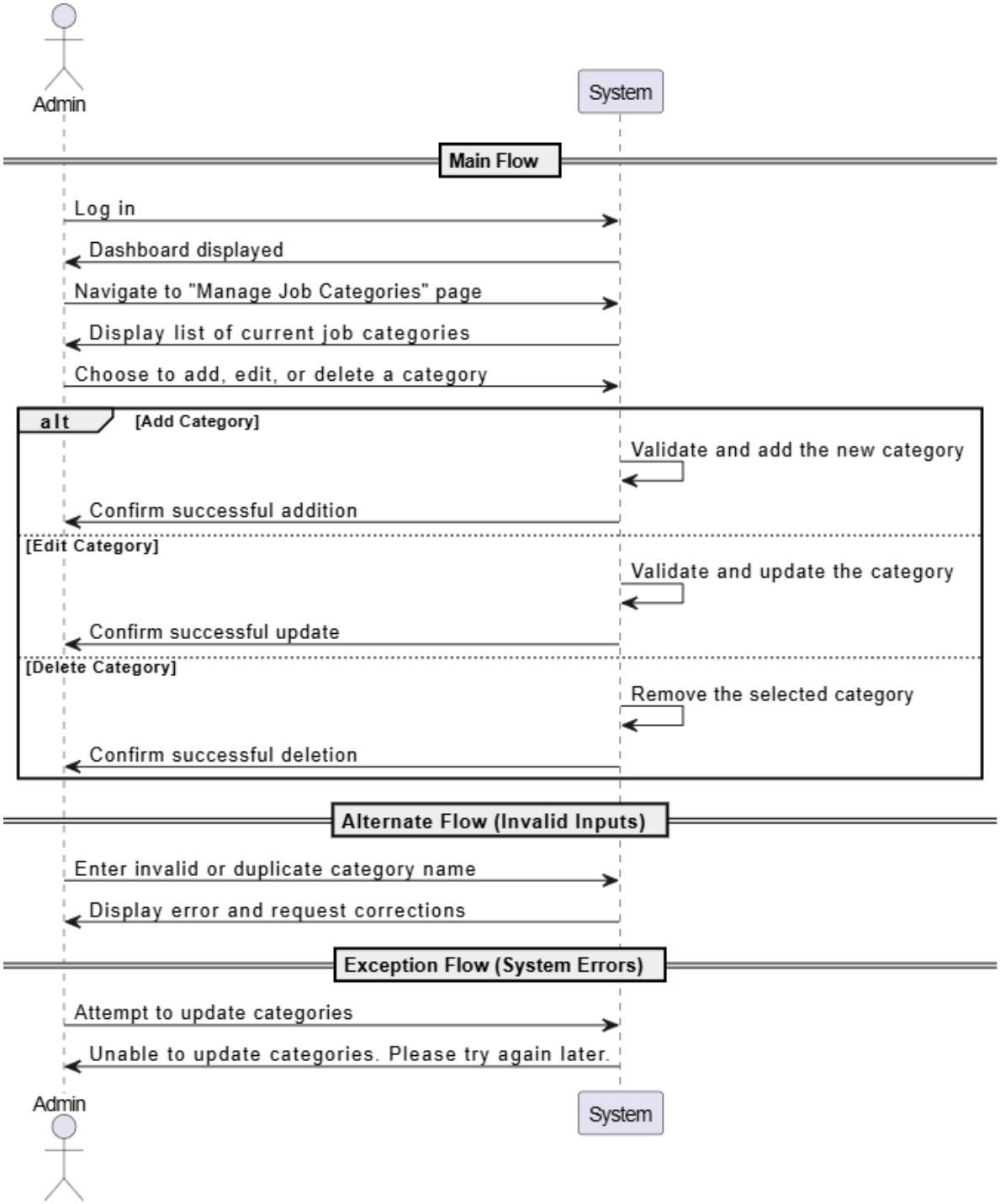

Figure 57 Manage Job Sequence Diagram

8.3.12 Add Timetable

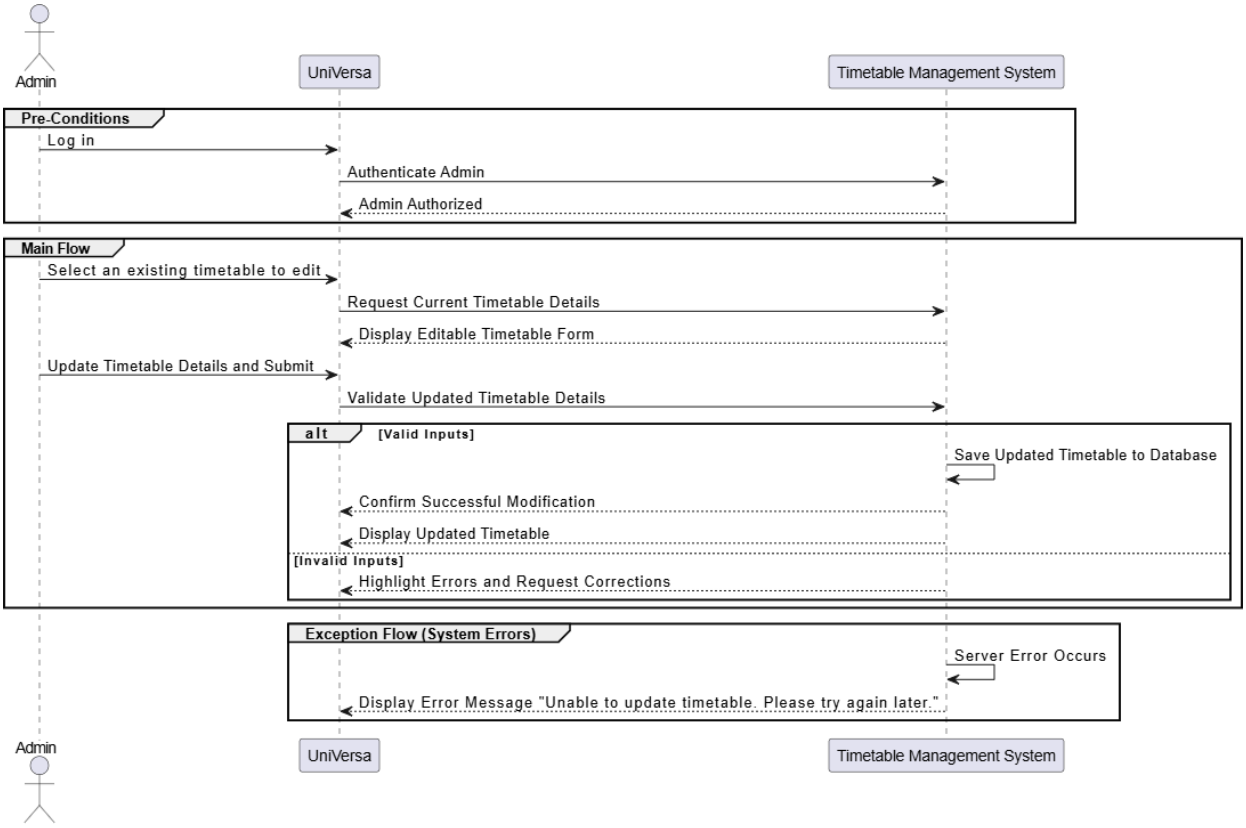

Figure 58 Add timetable Sequence Diagram

8.3.13 View Timetable

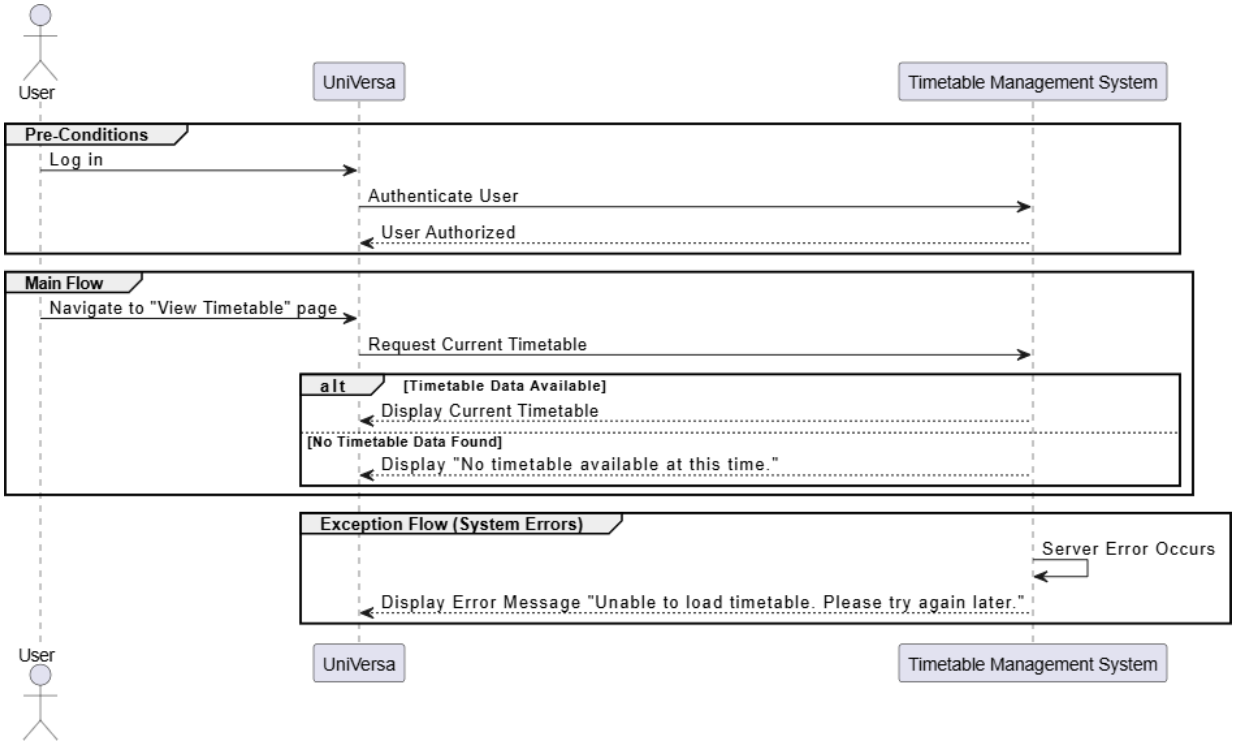

Figure 59 View Timetable Sequence Diagram

8.3.14Remove Timetable

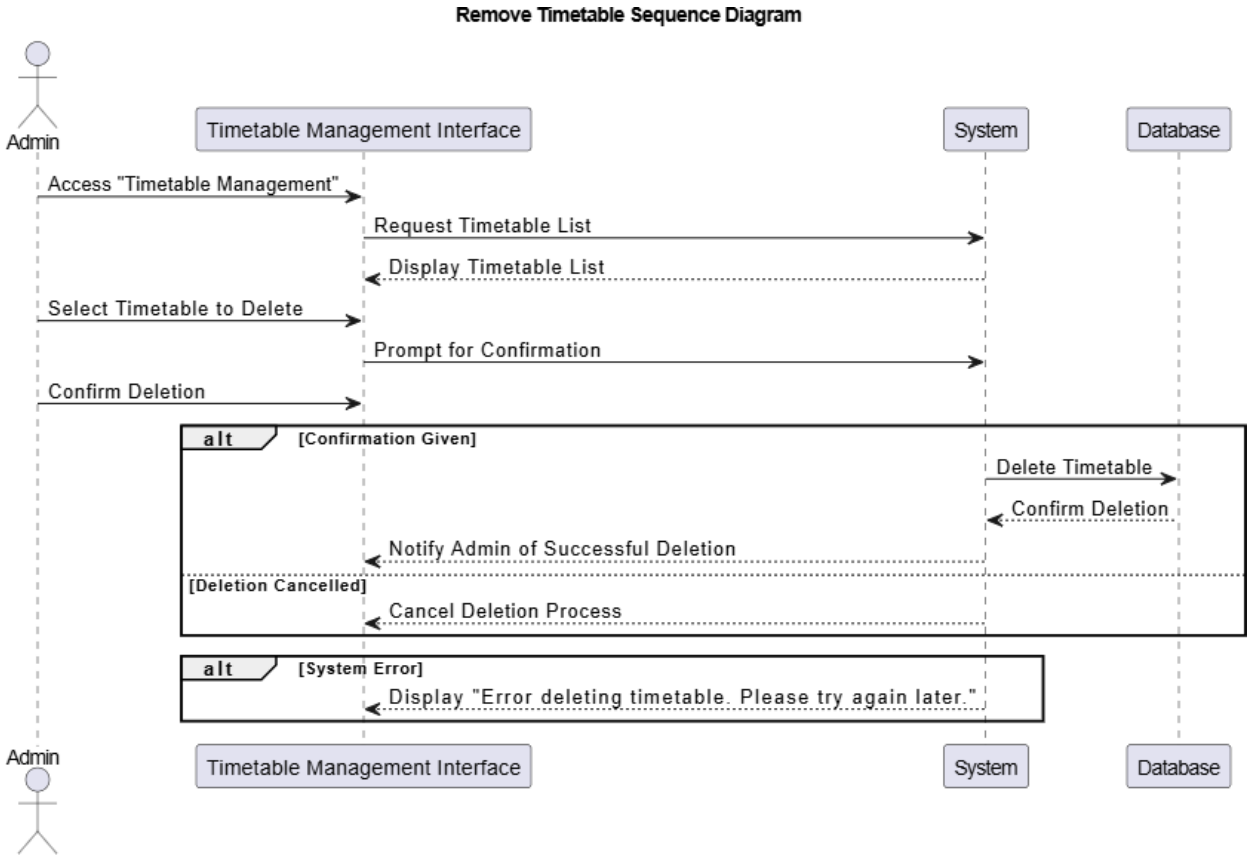

Figure 60 Remove Timetable Sequence Diagram

8.3.15 Send Message

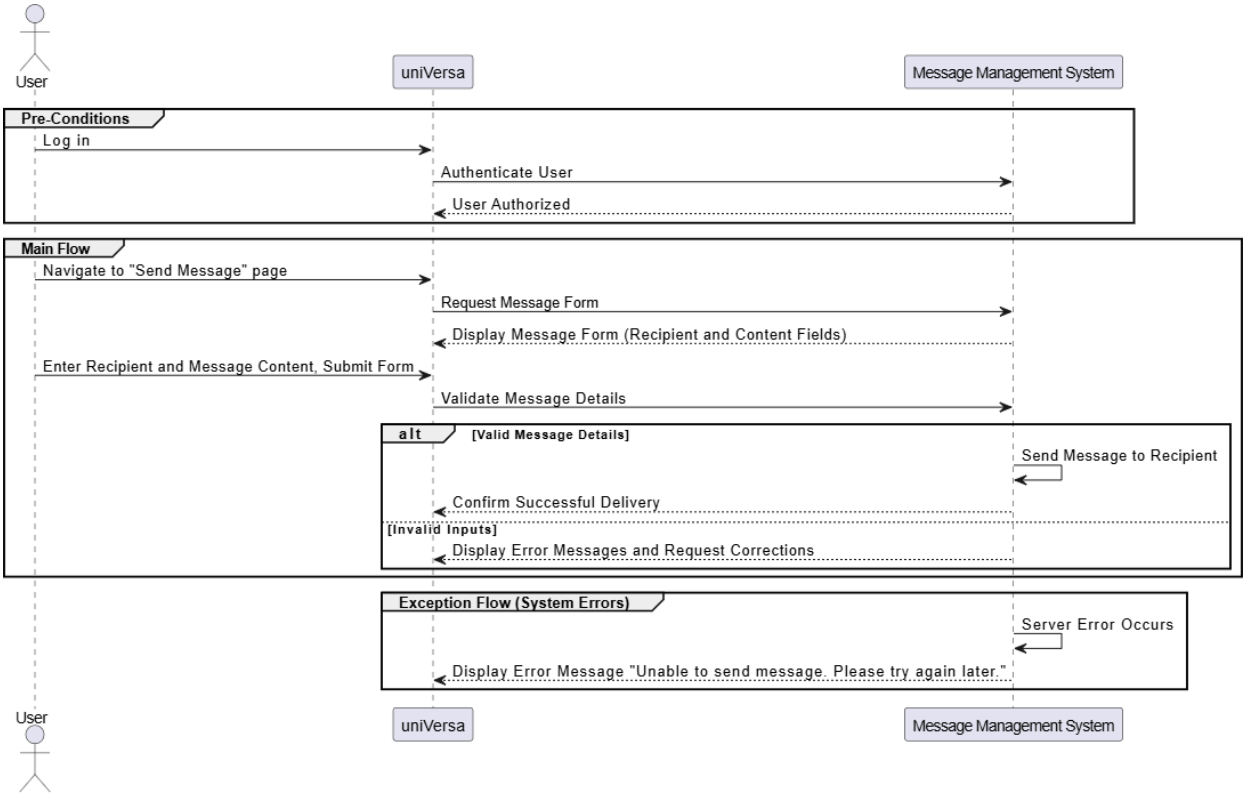

Figure 61 Send Message Sequence Diagram

8.3.16 Receive Message

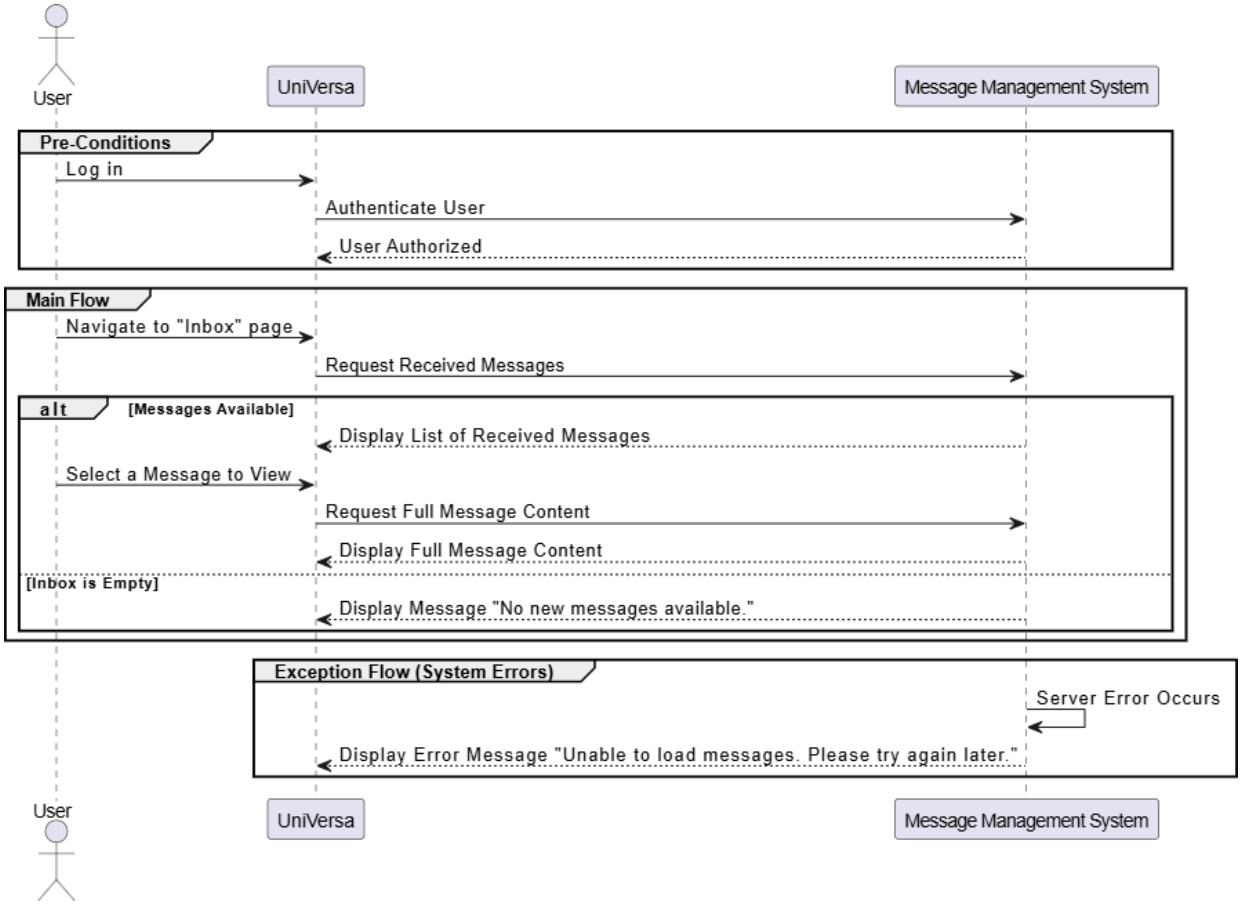

Figure 62 Receive Message Sequence Diagram

8.3.17Message Approval

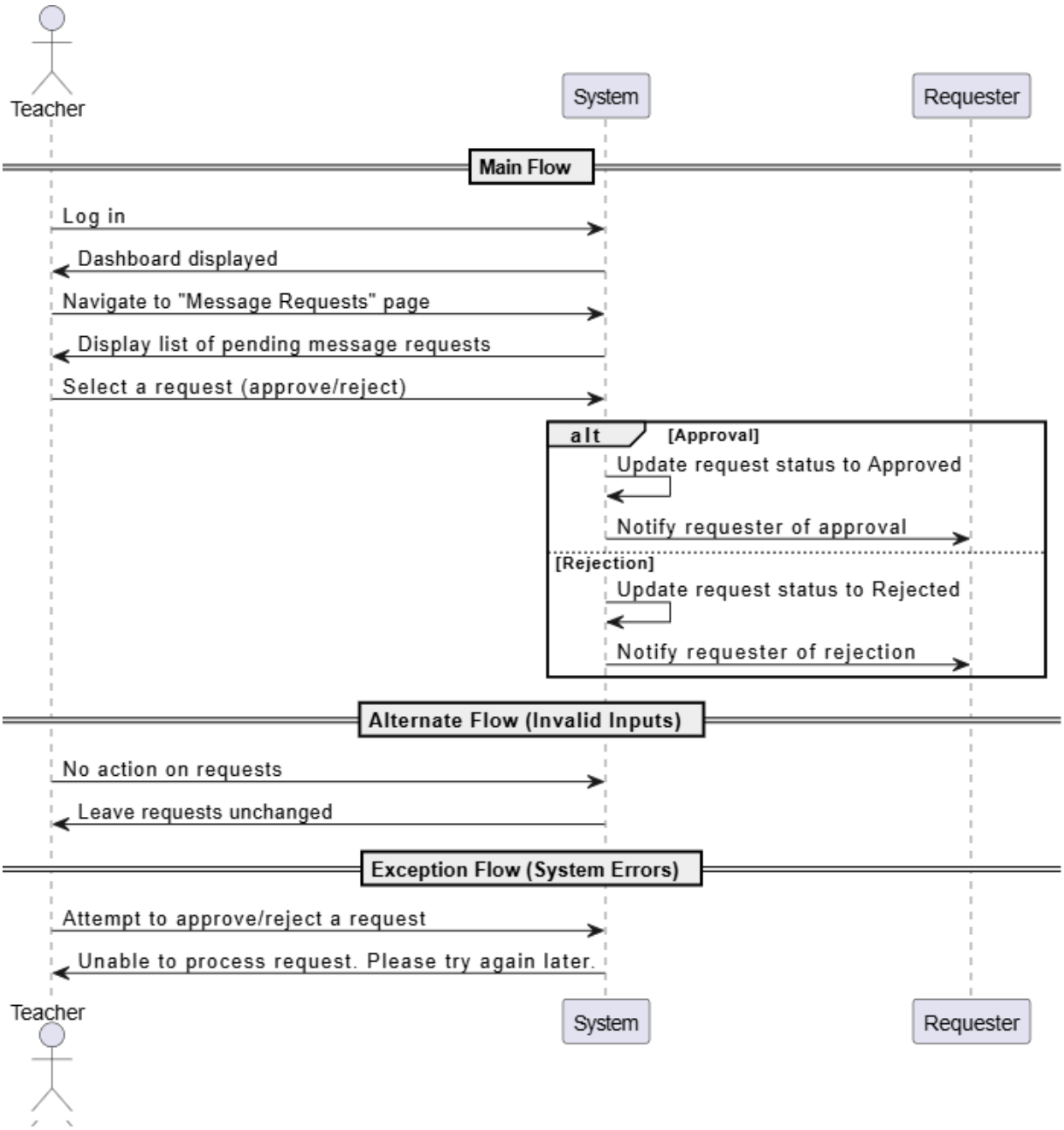

Figure 63 Message Approval Sequence Diagram

8.3.18 Receive Notification

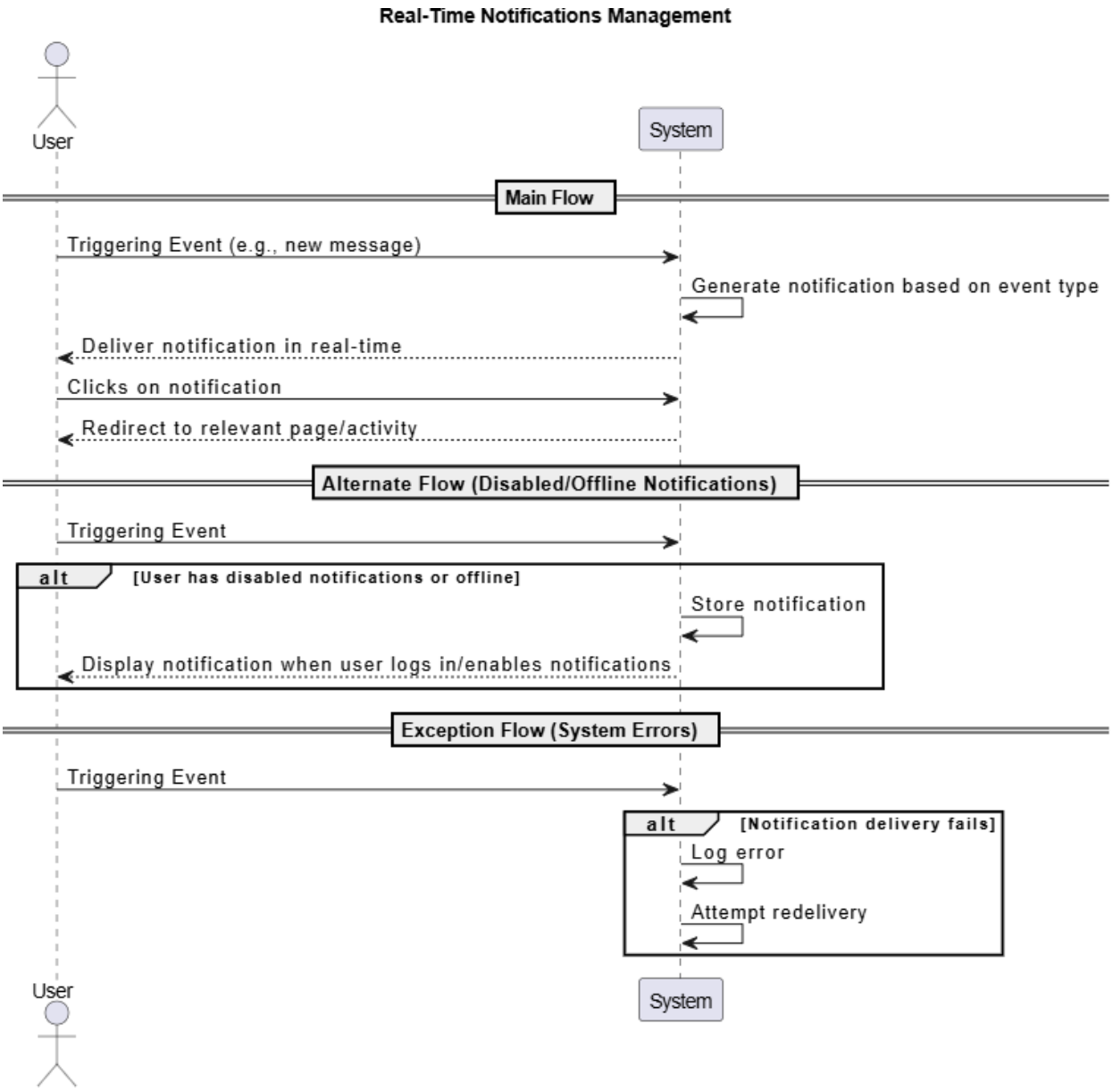

Figure 64 Receive Notification Sequence Diagram

8.3.19 Join Group

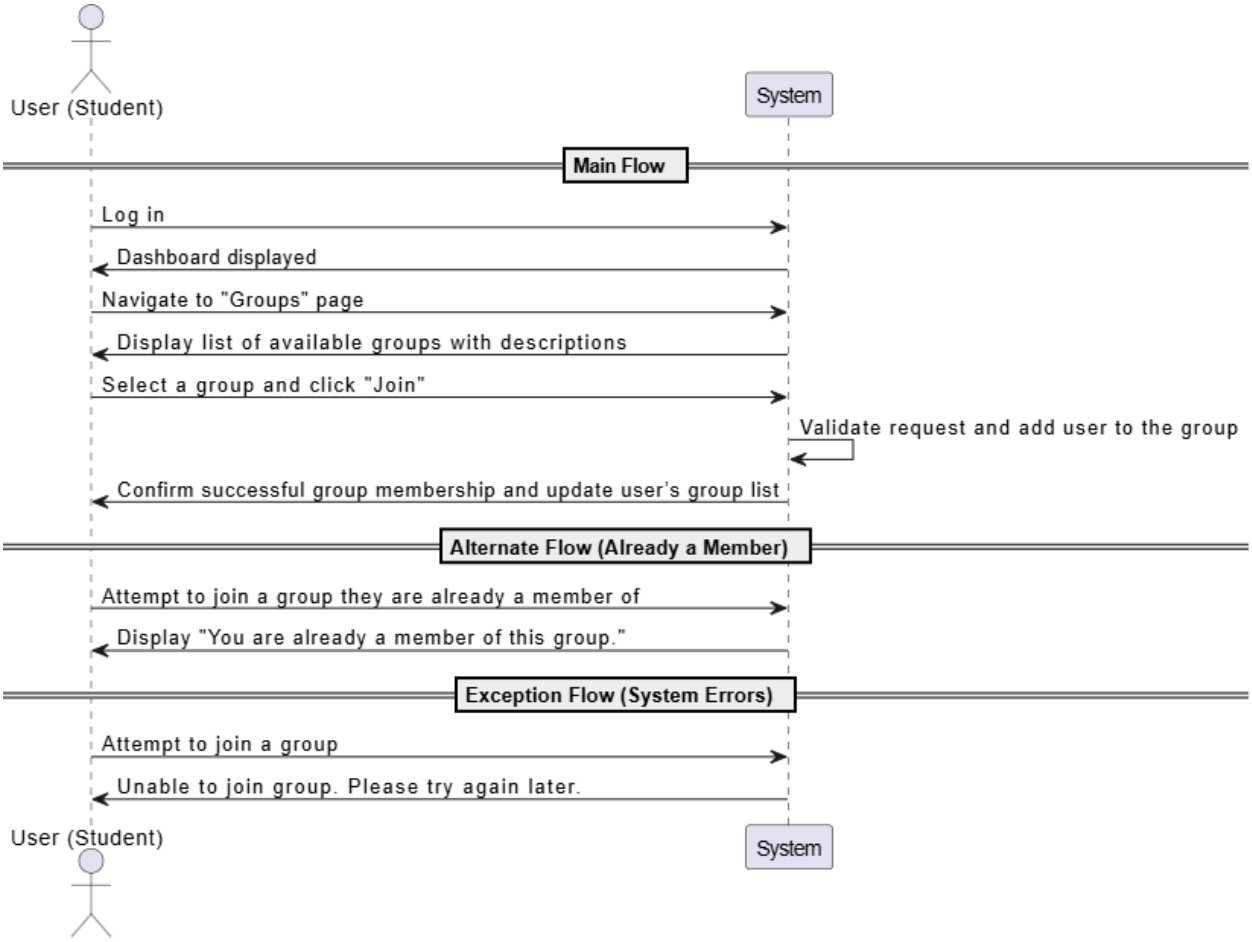

Figure 65 Join Group Sequence Diagram

8.3.20Leave Group

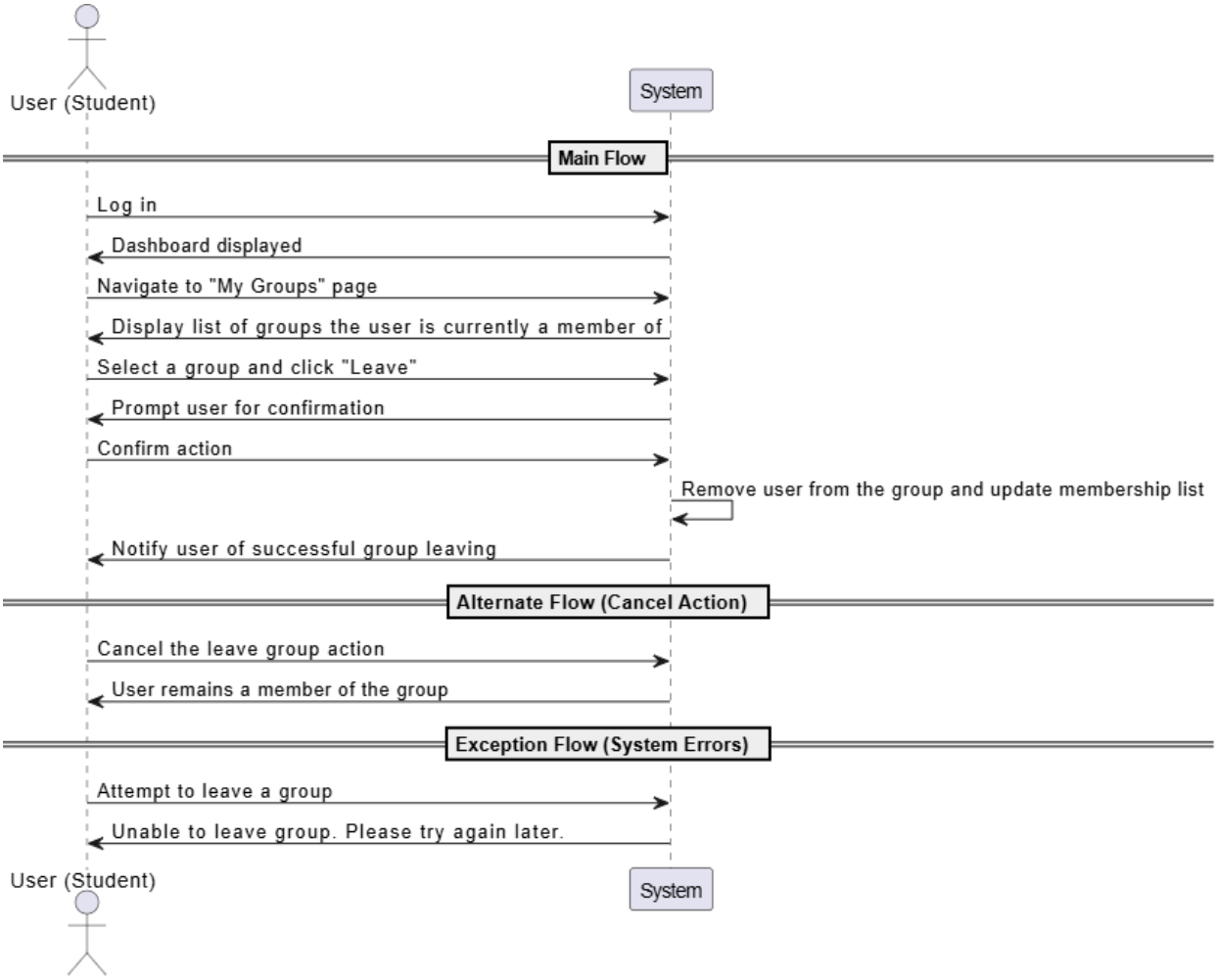

Figure 66 Leave Group Sequence Diagram

8.3.21 Marketplace

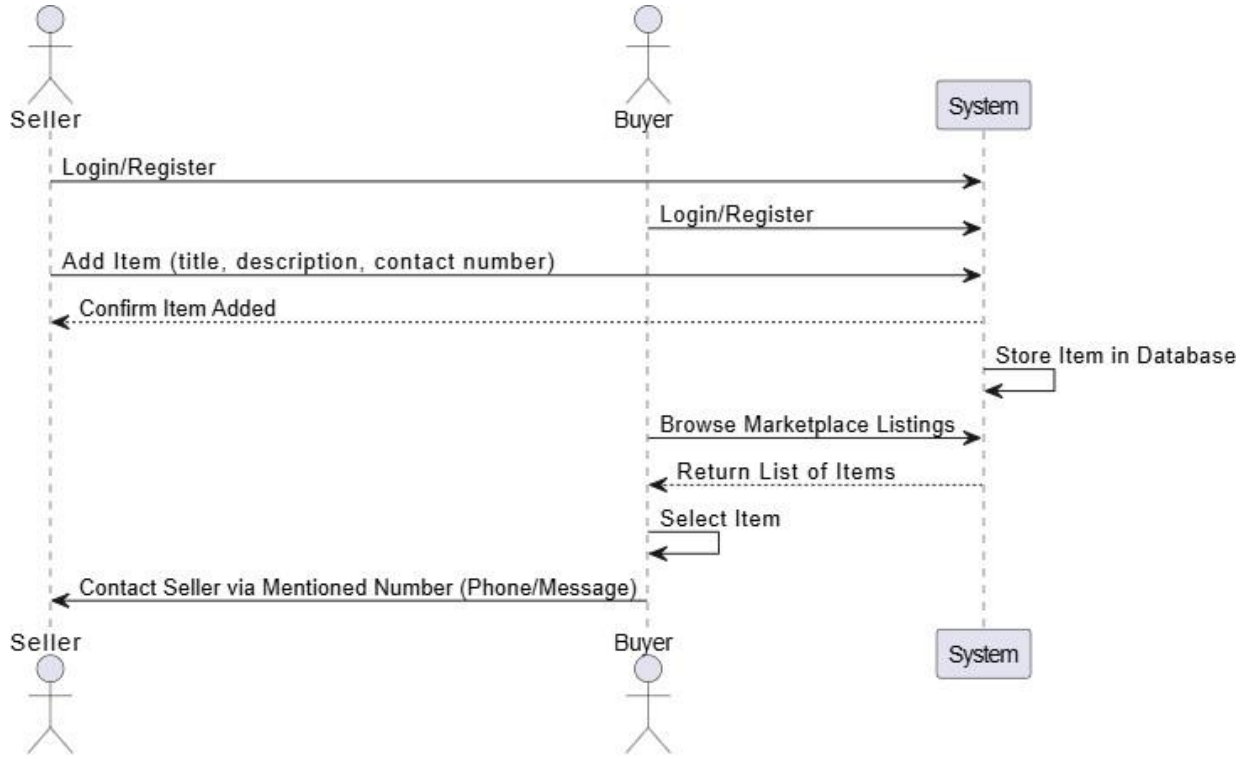

Figure 67 Marketplace Sequence Diagram

8.3.22 Edit Marketplace

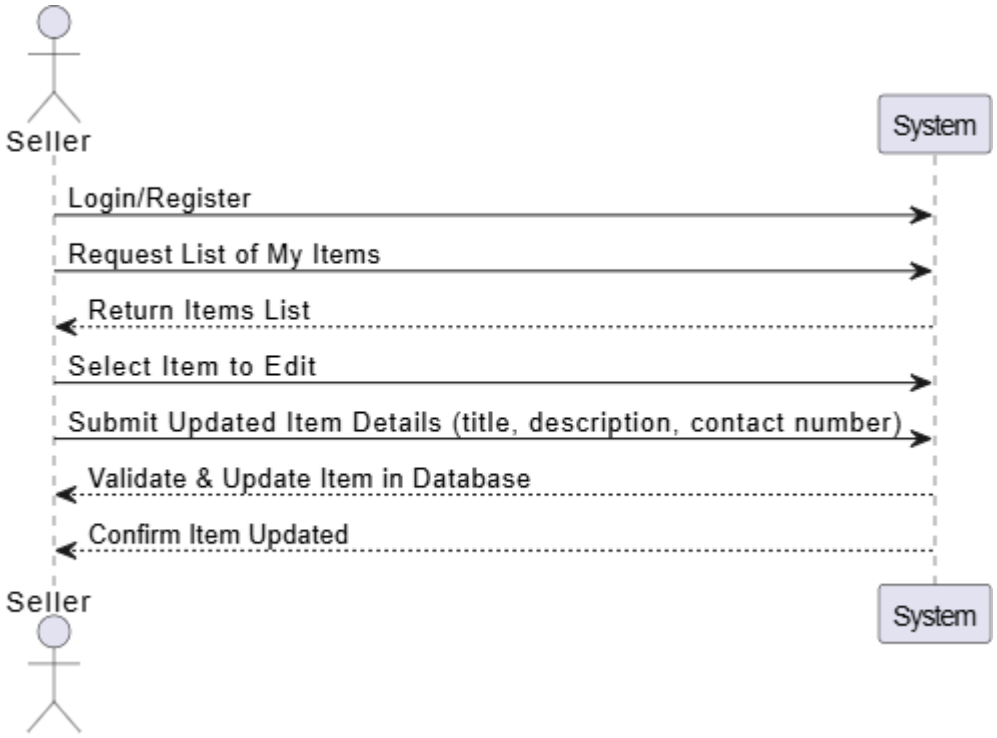

Figure 68 Edit Marketplace Sequence Diagram

8.3.23Delete Marketplace

Figure 69 Delete Marketplace Sequence Diagram

8.3.24 Carpool

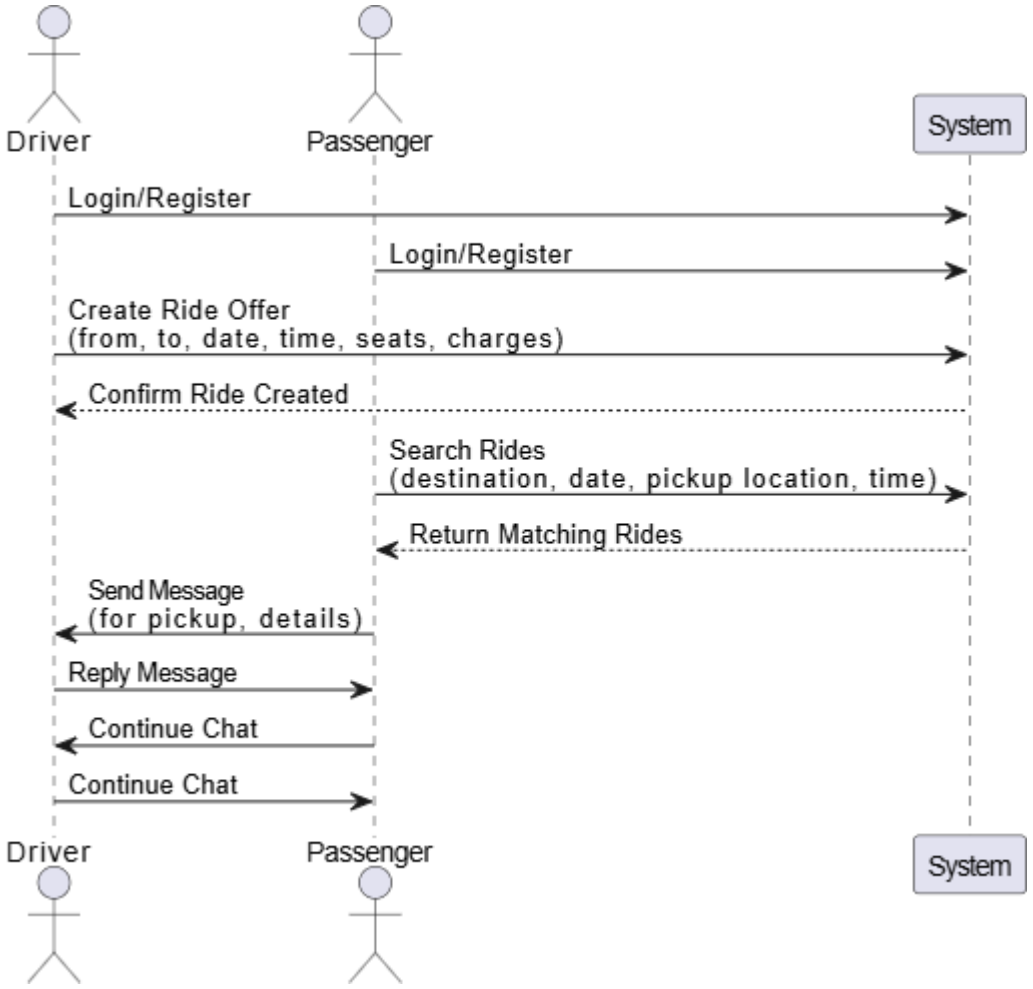

Figure 70 Carpool Sequence Diagram

8.3.25 Edit Carpool

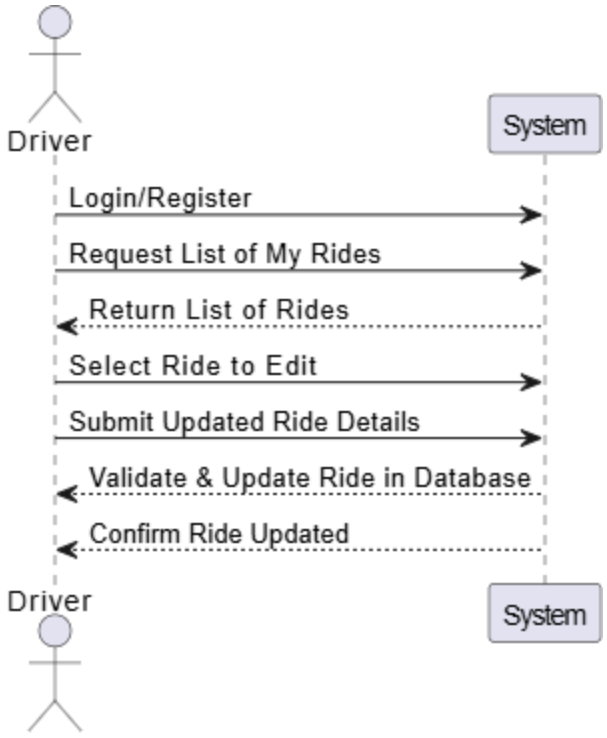

Figure 71 Edit Carpool Sequence Diagram

8.3.26 Delete Carpool

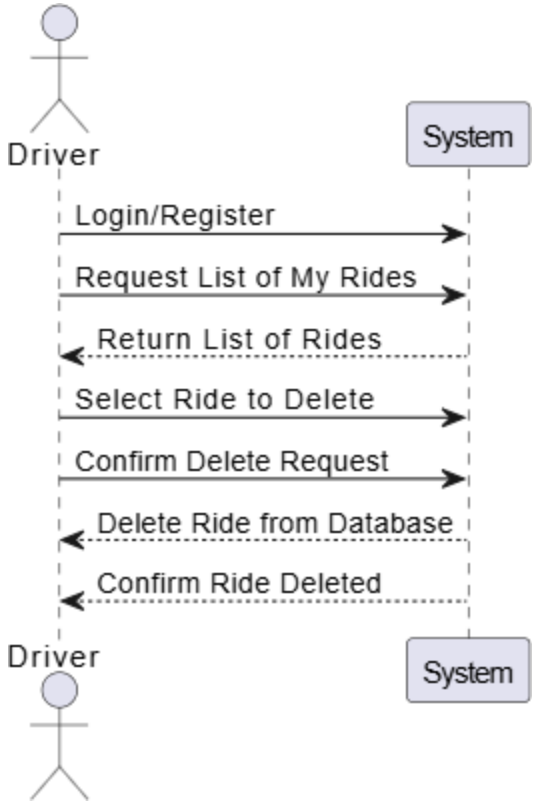

Figure 72 Delete Carpool Sequence Diagram

8.3.27 Add To-Do List

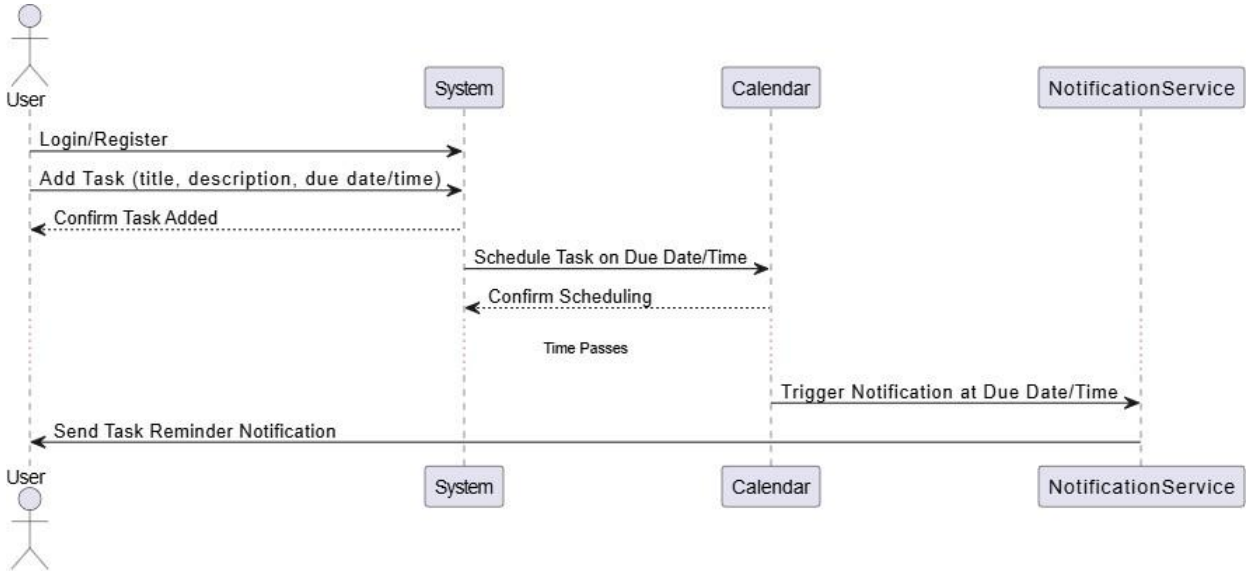

Figure 73 Add To-Do List Sequence Diagram

8.3.28 Edit To-Do List

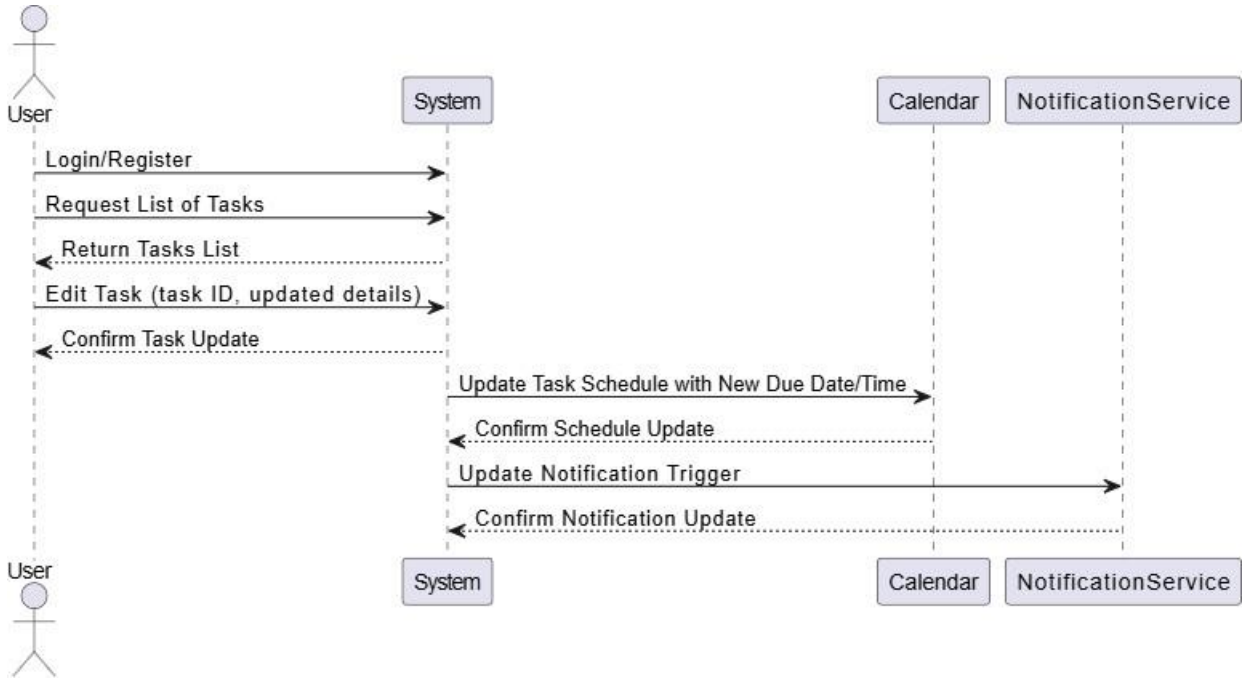

Figure 74 Edit To-Do List Sequence Diagram

8.3.29 Remove To-Do List

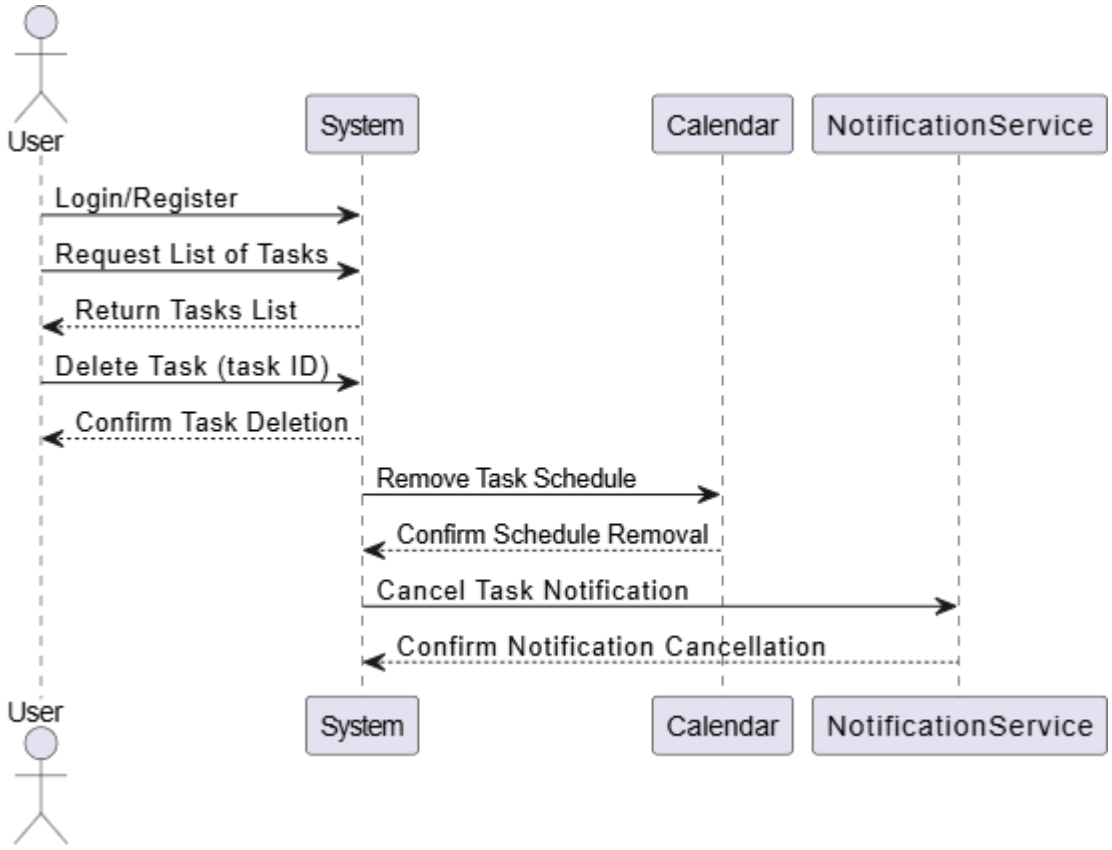

Figure 75 Remove TO-DO List Sequence Diagram

8.3.30 Resource Management

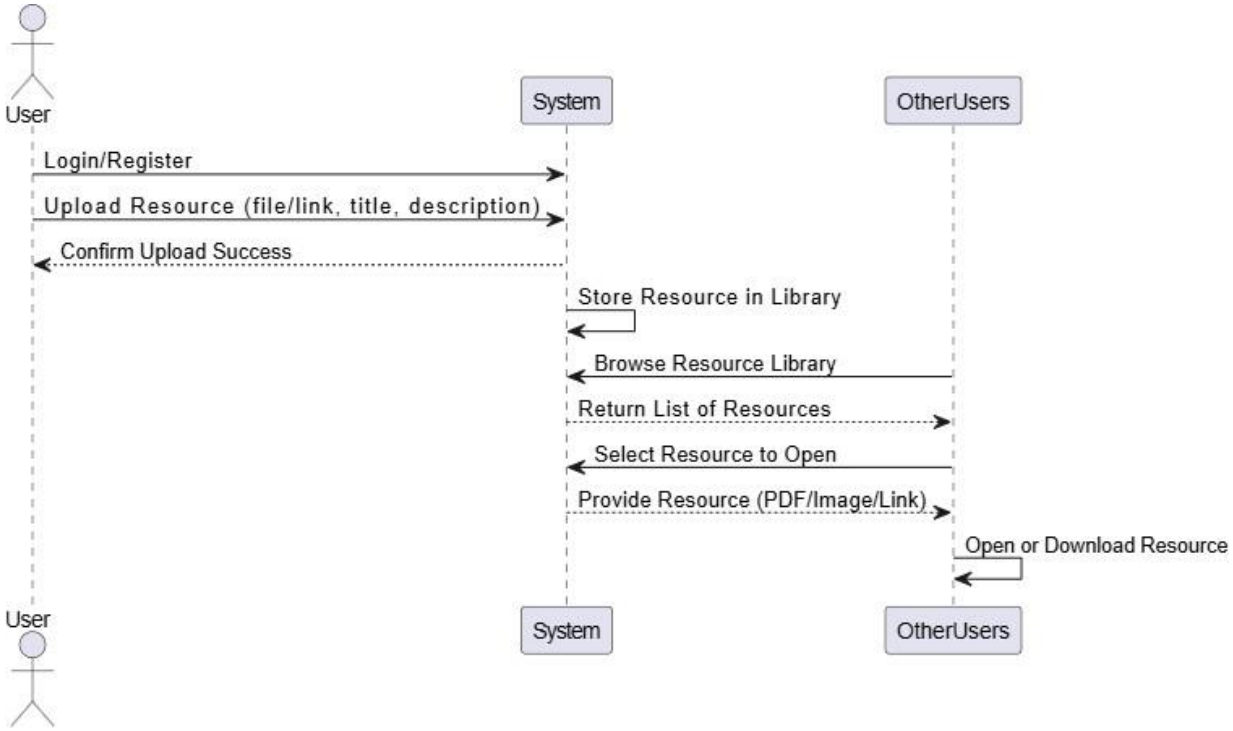

Figure 76 Resource Management Sequence Diagram

8.3.31 Lost and Found

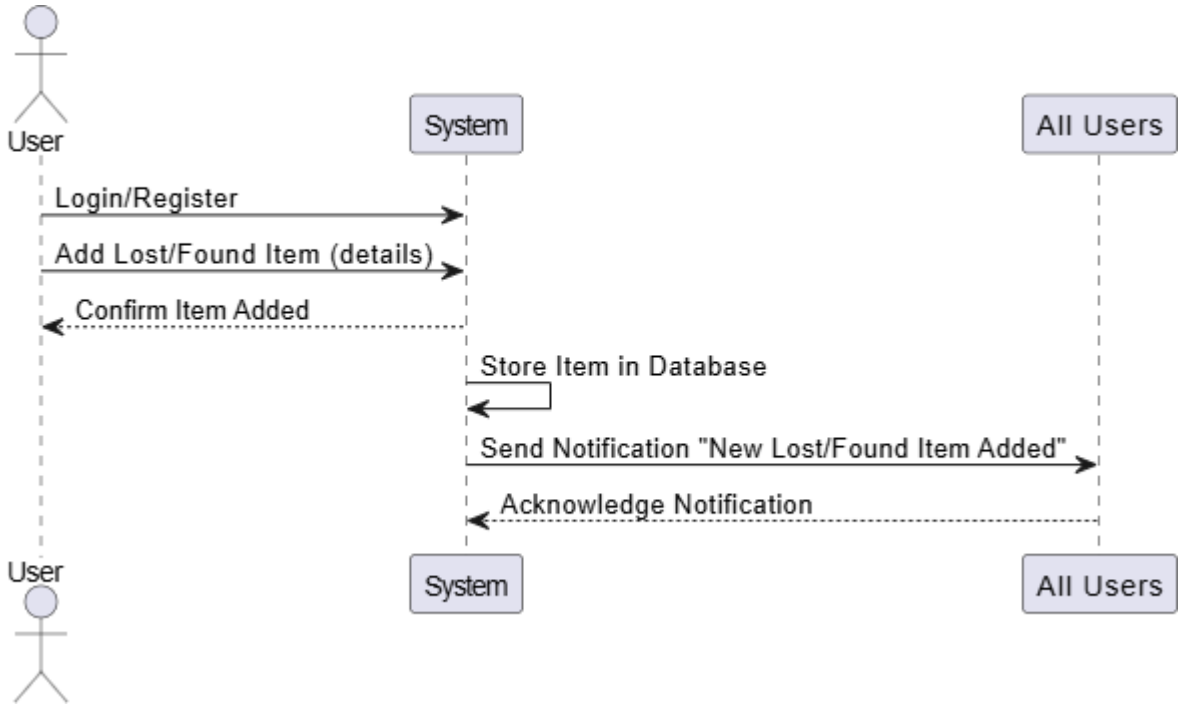

Figure 77 Lost and Found Sequence Diagram

8.4 Use-Case Diagrams

8.4.1 Student

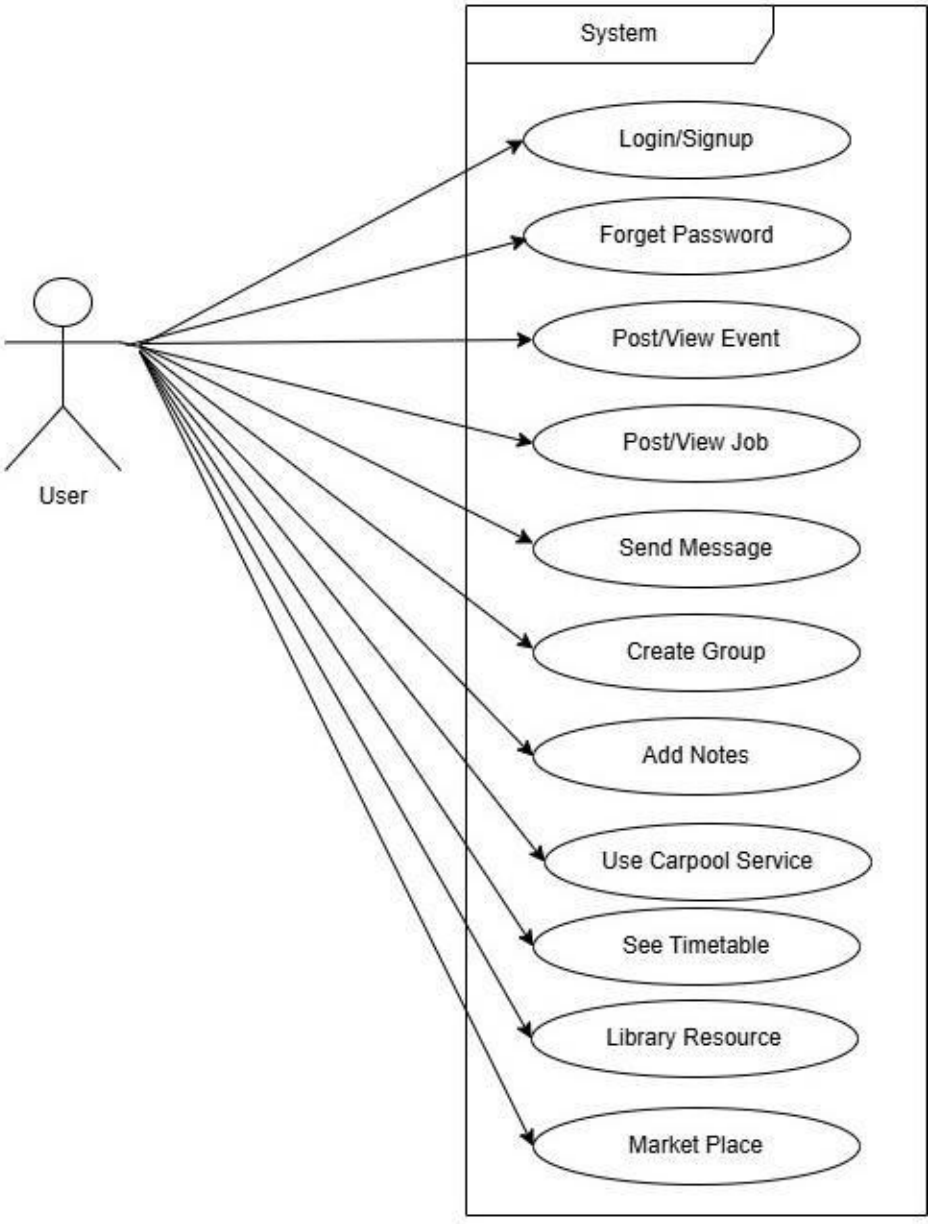

Figure 78 Student Use Case diagram

8.4.2 Teacher

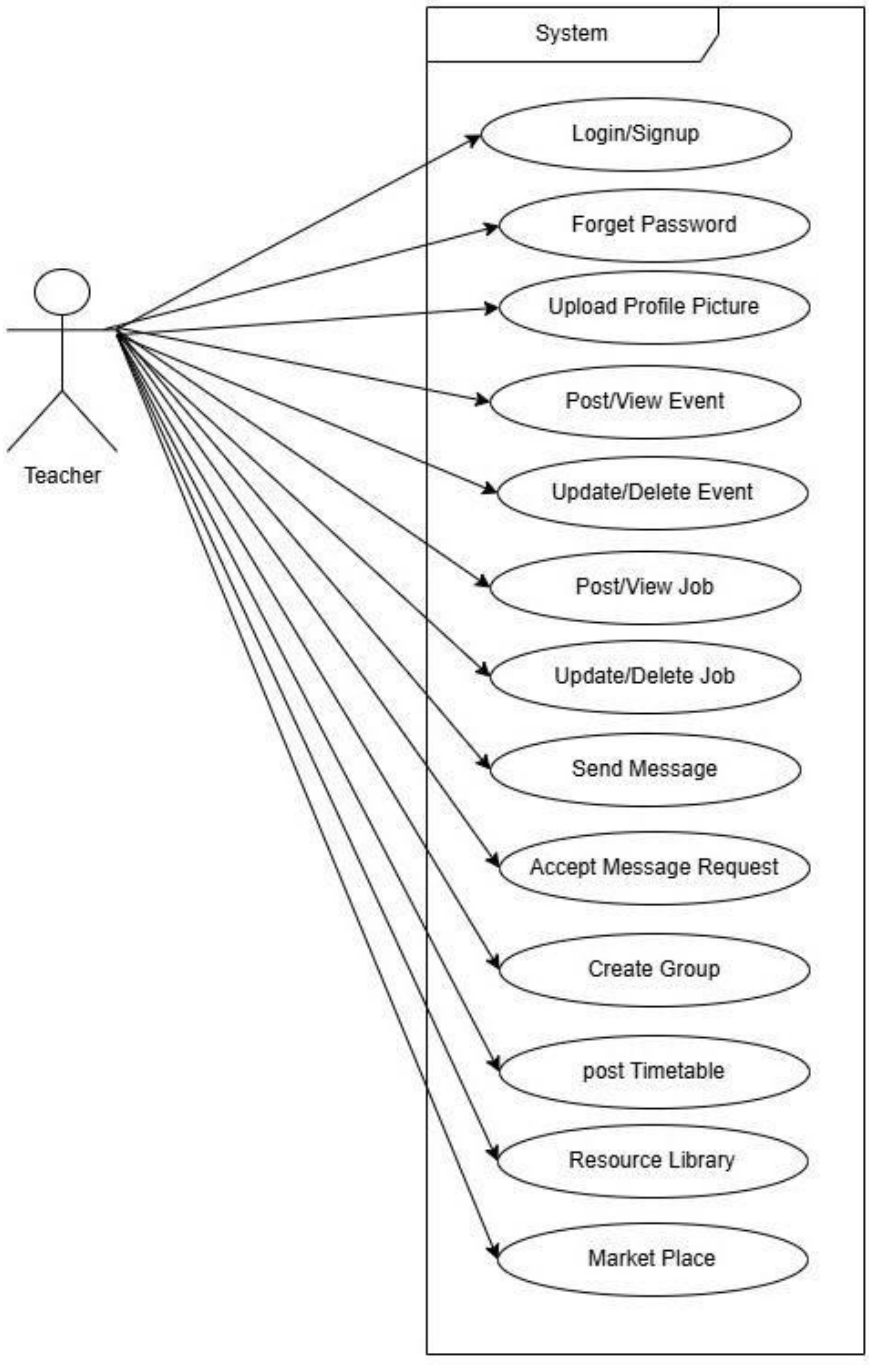

Figure 79 Teacher Use Case diagram

8.4.3 Parent

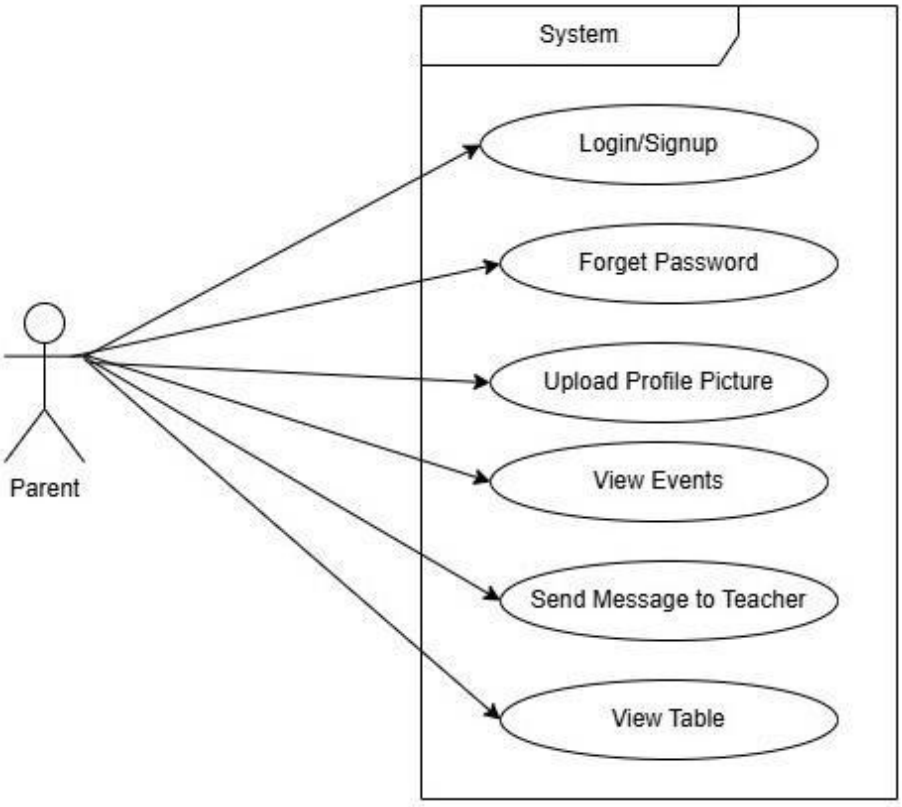

Figure 80 Parent Use Case diagram

8.4.4 Admin

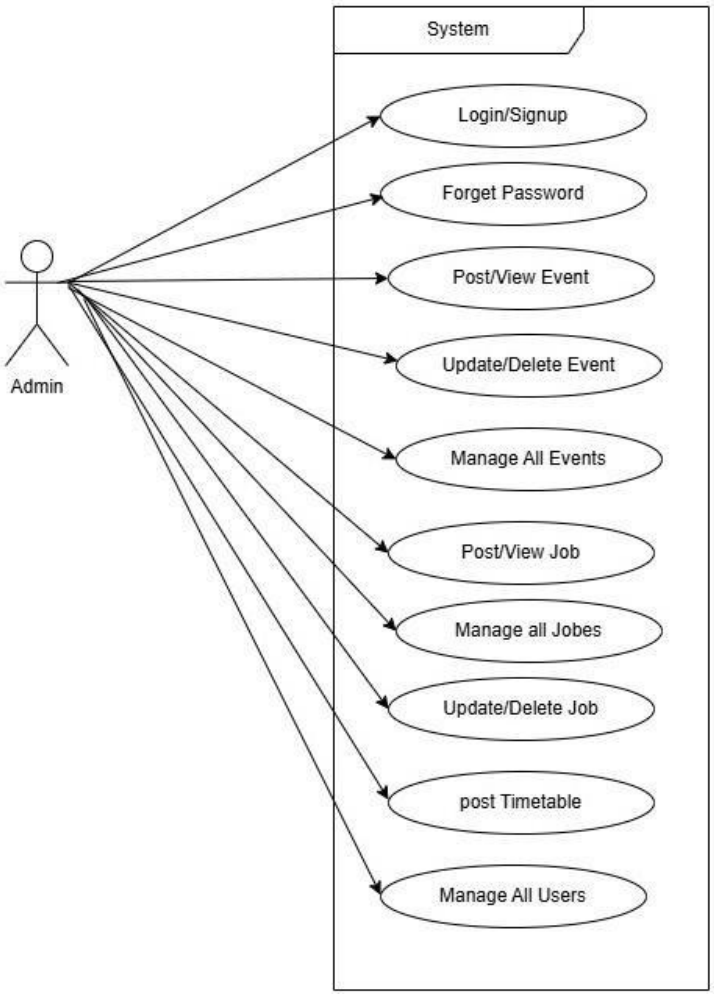

Figure 81 Admin Use Case diagram

8.5 Class Diagram

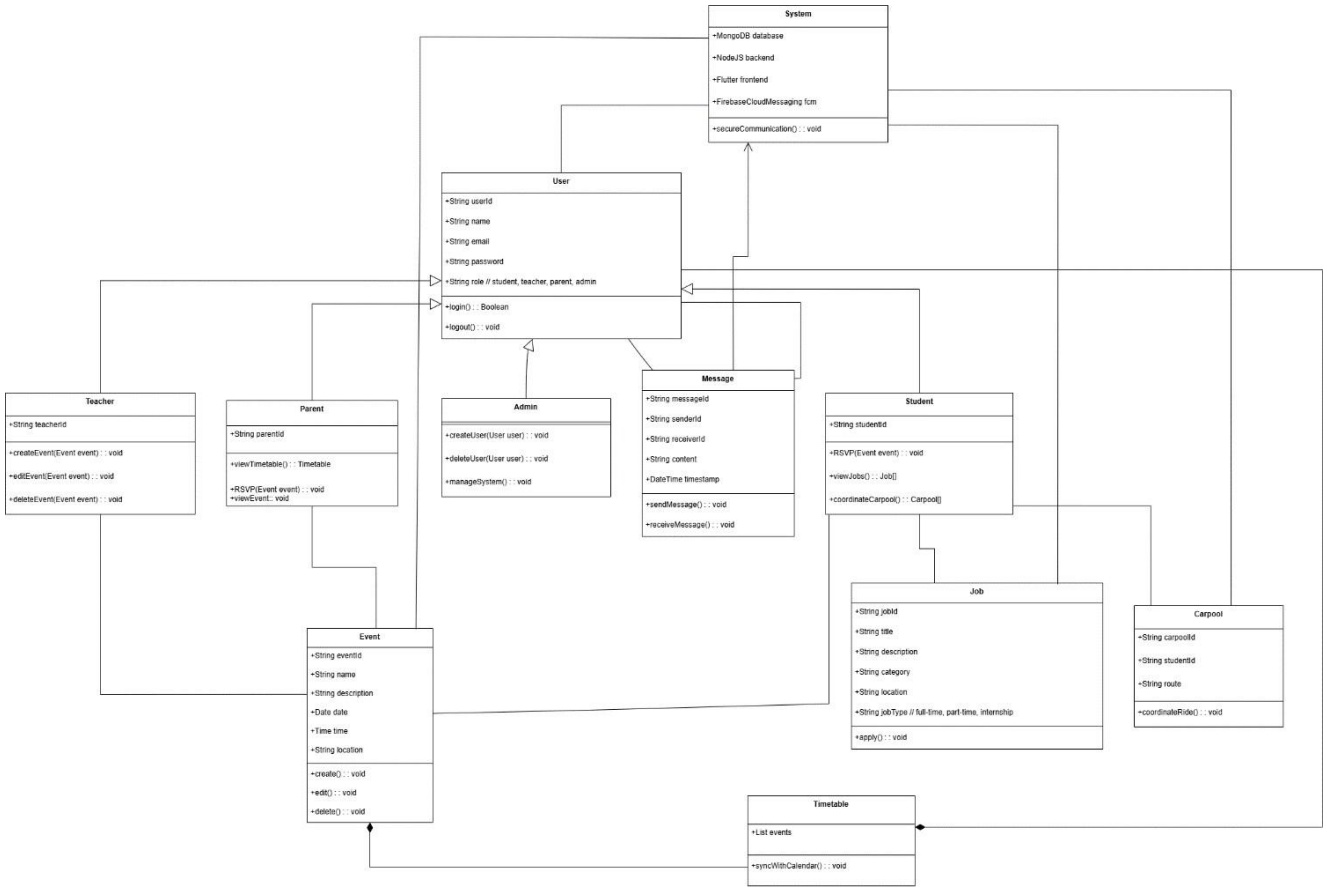

Figure 82 Class Diagram

8.6 Component Diagram

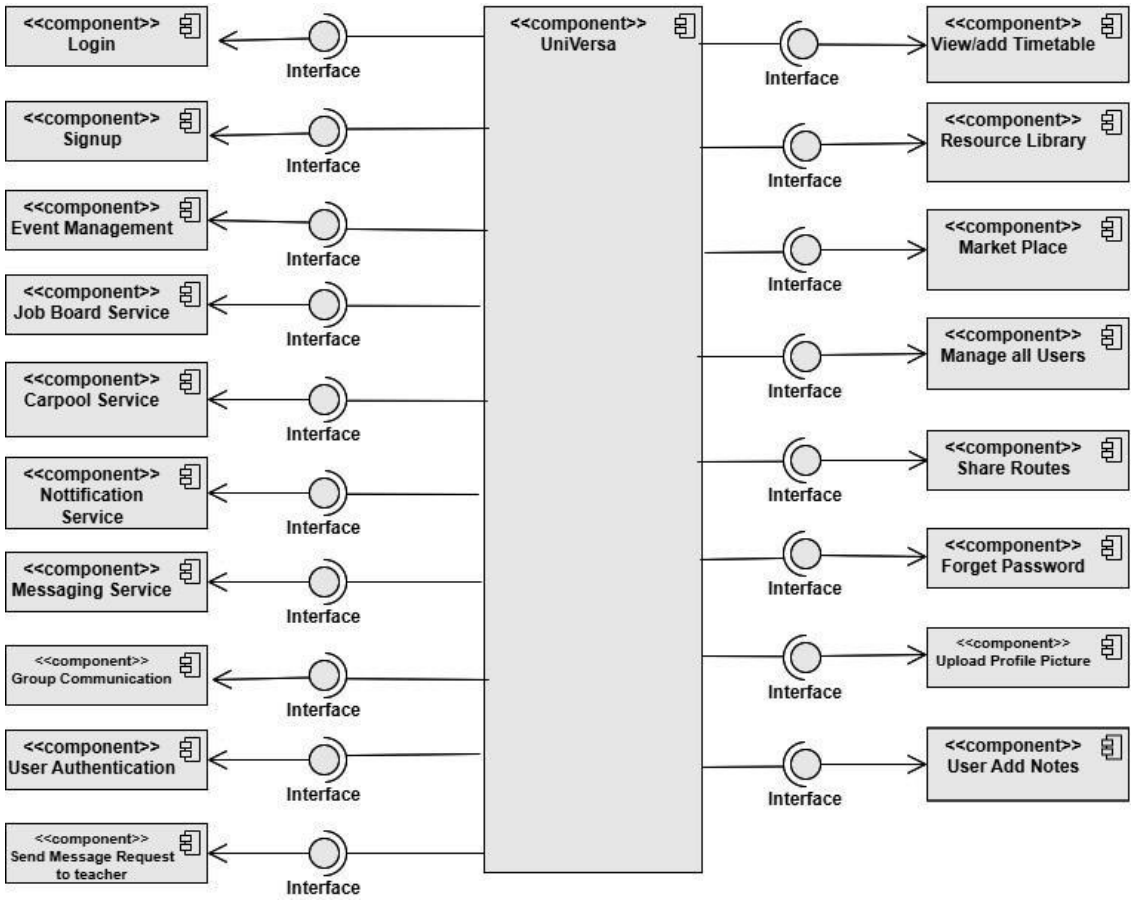

Figure 83 Component Diagram

8.7 State Chart Diagram

8.7.1 Admin

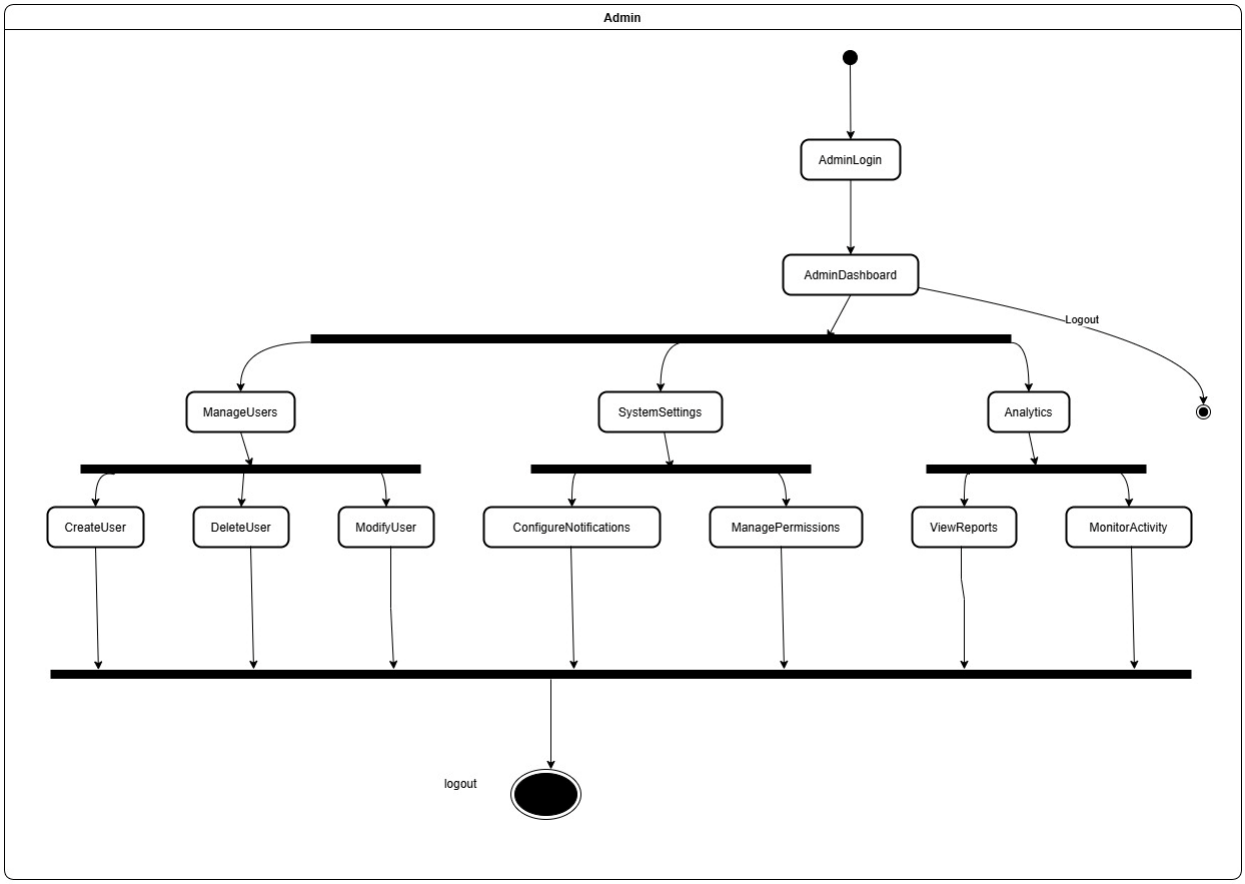

Figure 84 Admin State chart Diagram

8.7.2 Parent

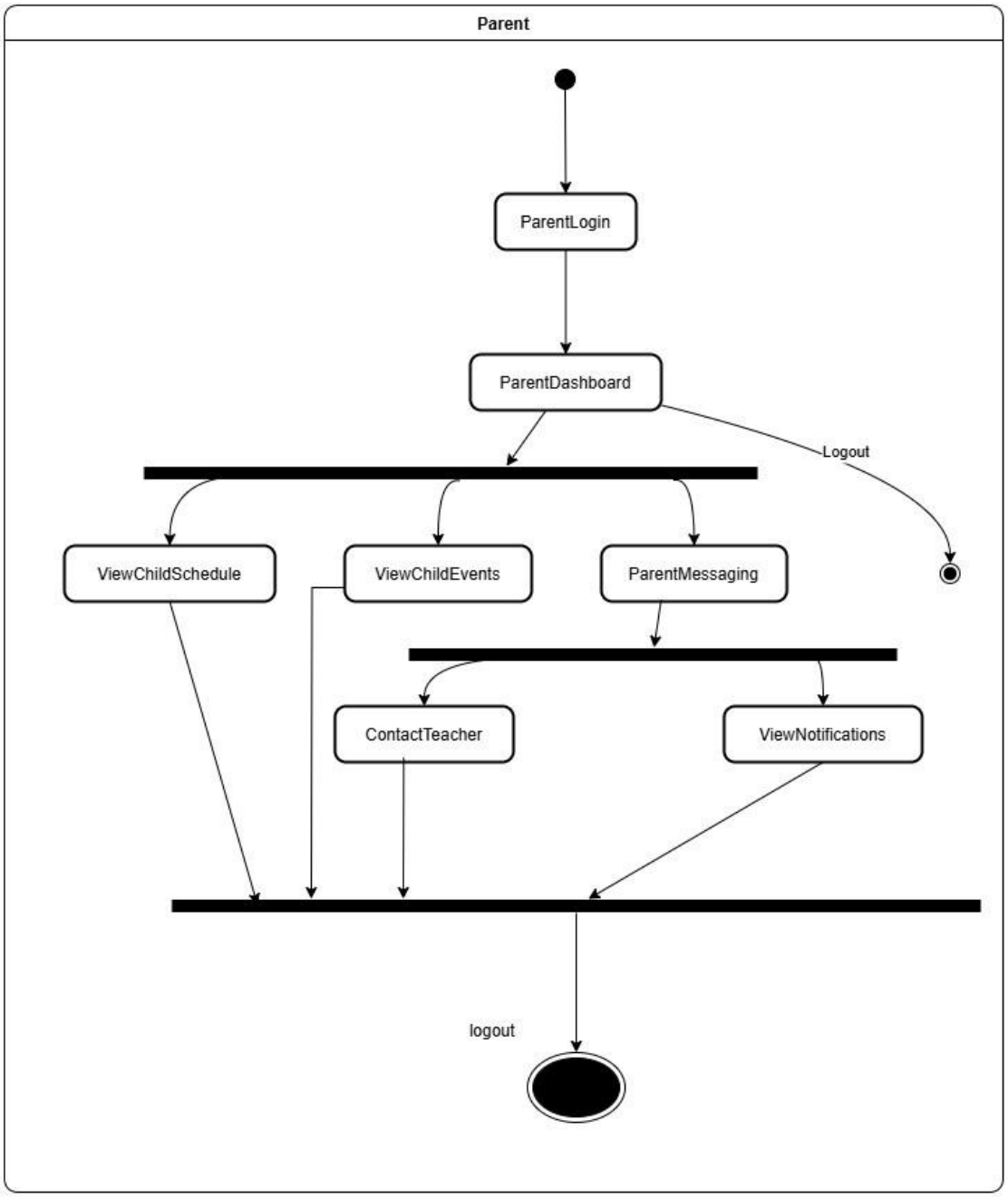

Figure 85 Parent State chart diagram

8.7.3 Teacher

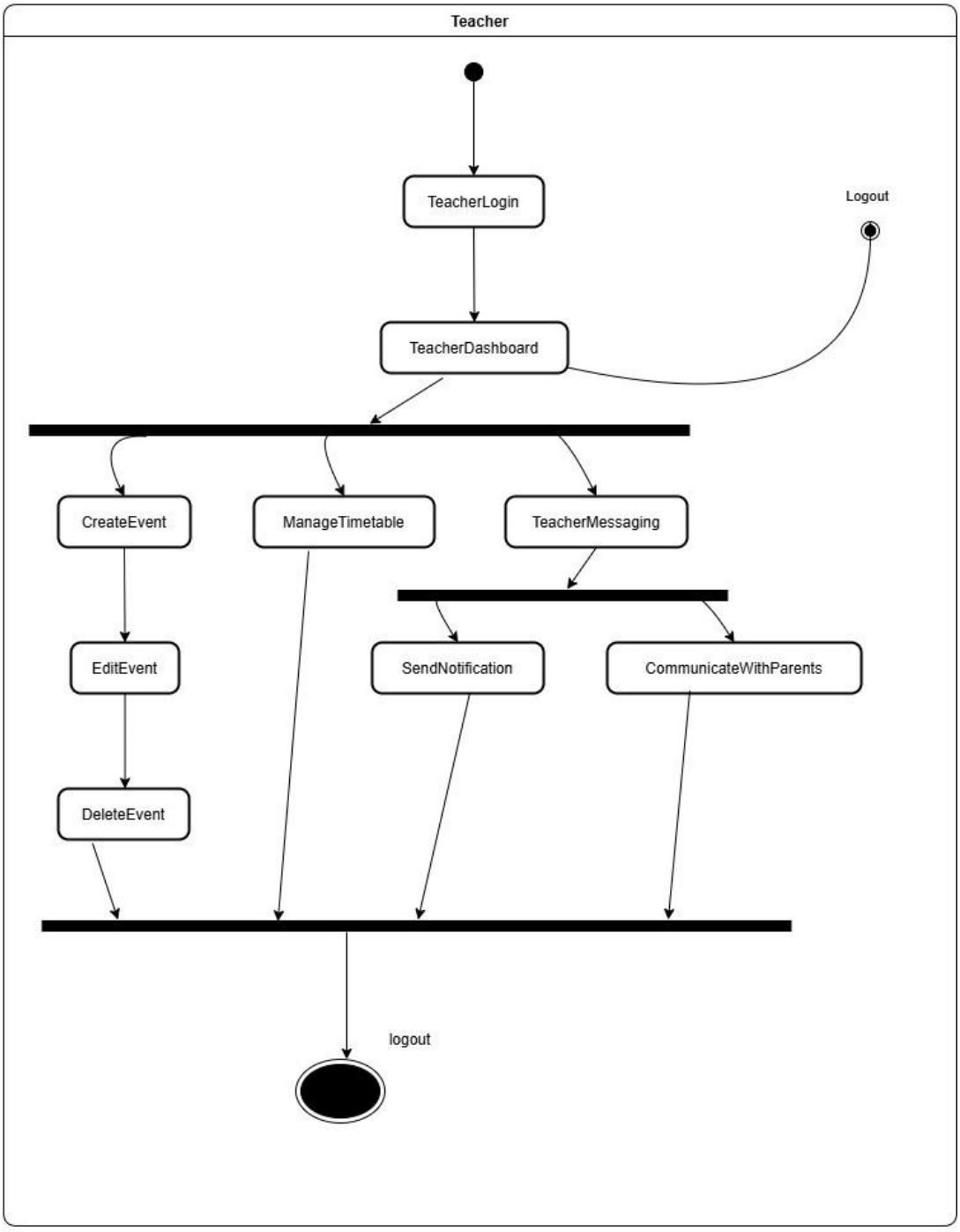

Figure 86 Teacher State chart Diagram

8.7.4 Student

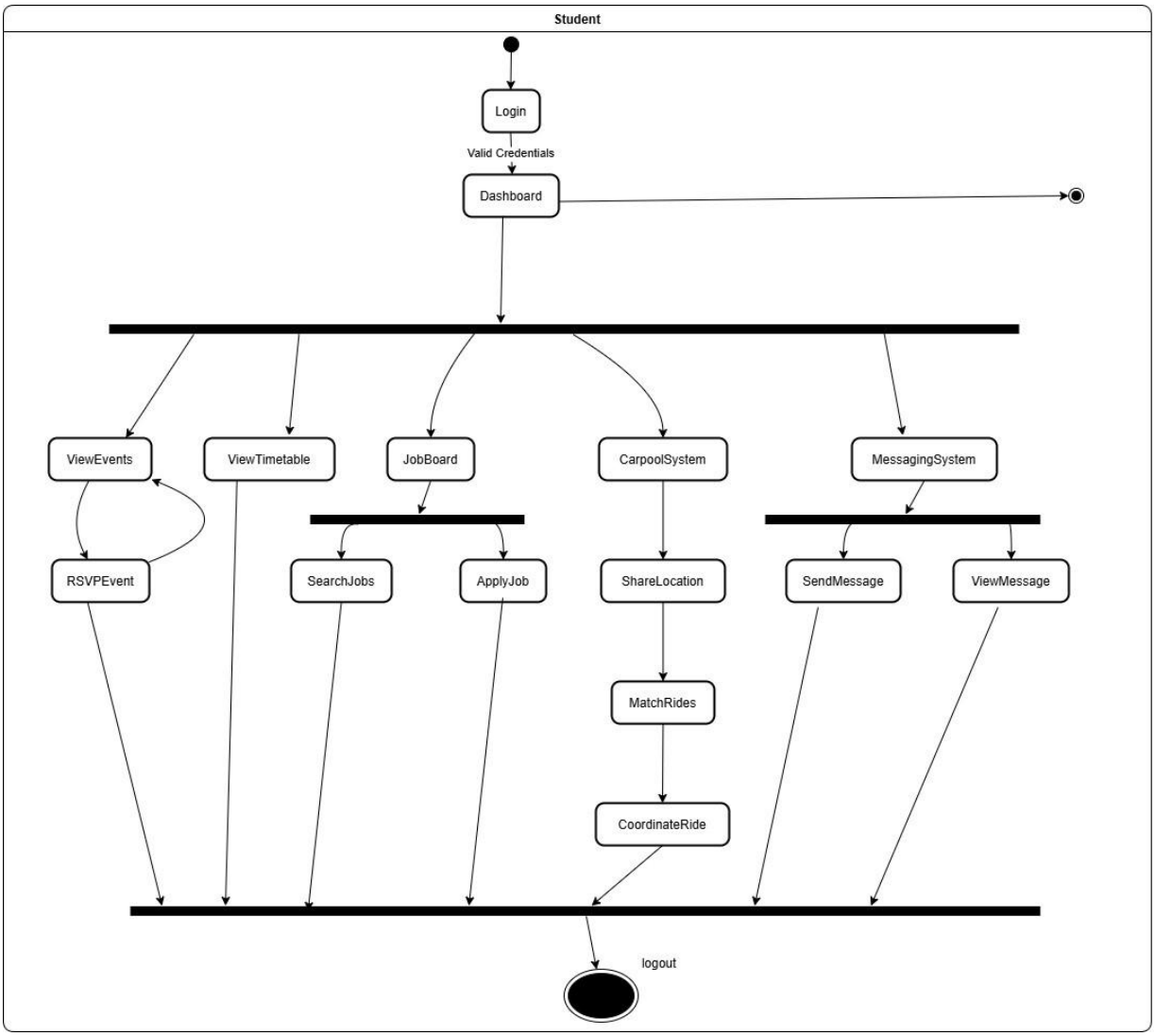

Figure 87 Student State chart Diagram

8.8 Deployment Diagram

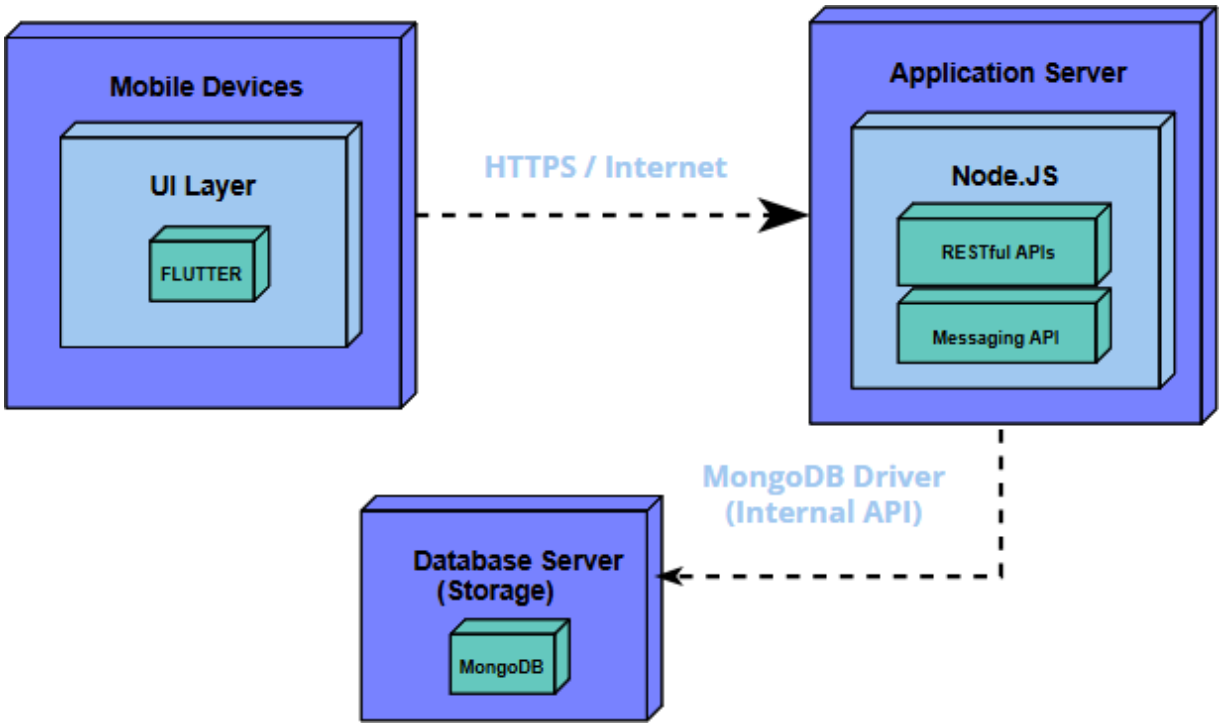

Figure 88 Deployment Diagram

8.9 System Block Diagram

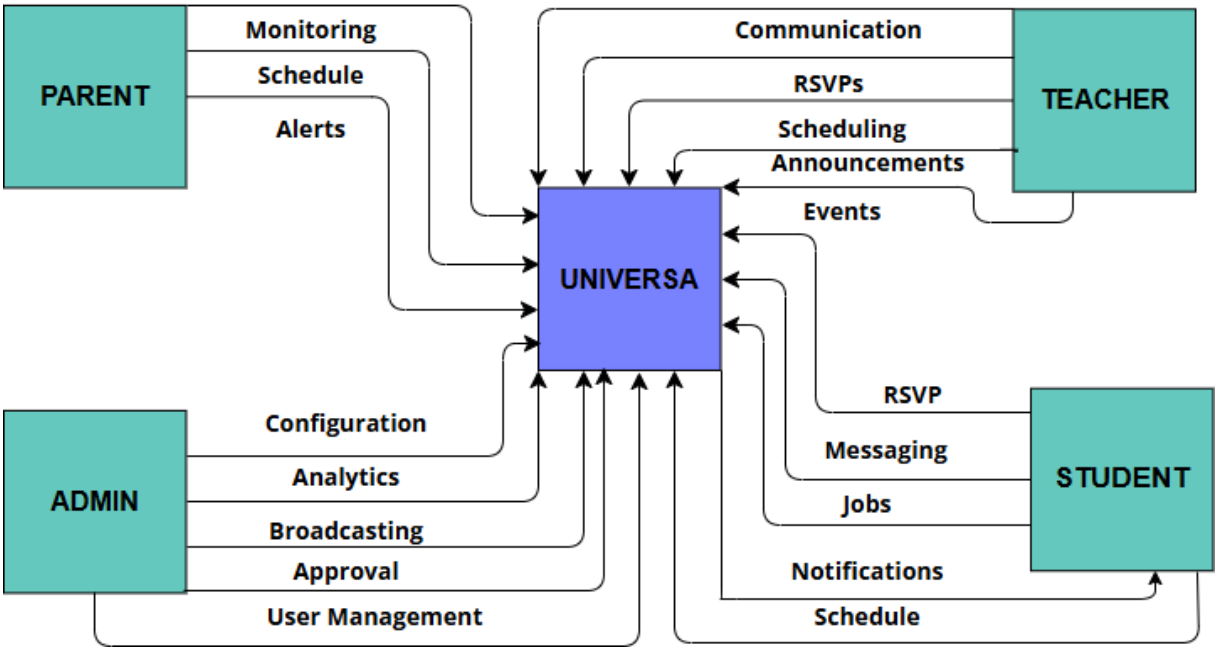

Figure 89 System Block Diagram

8.10 Collaboration Diagram

8.10.1 Registration

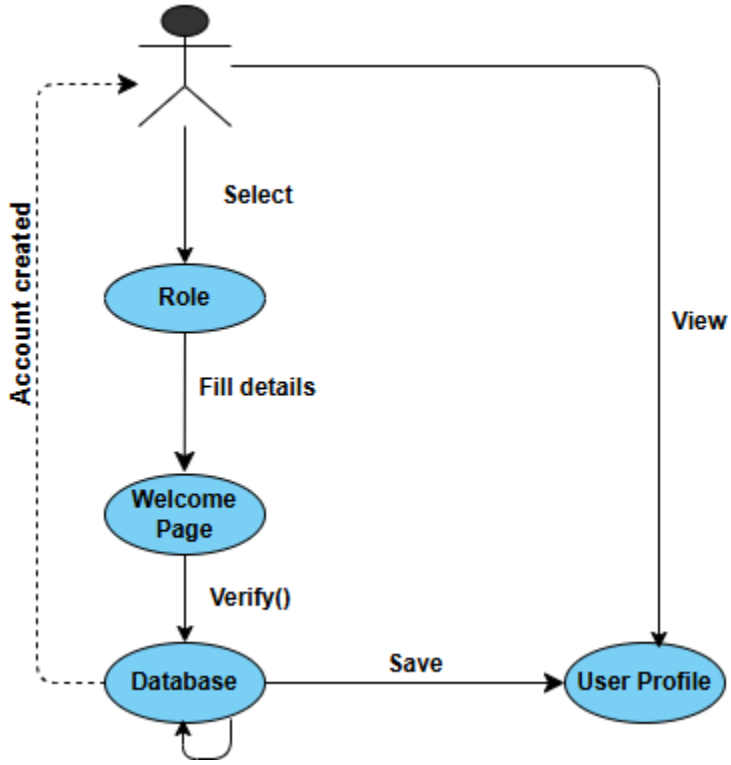

Figure 90 Registration Collaboration Diagram

8.10.2 Login

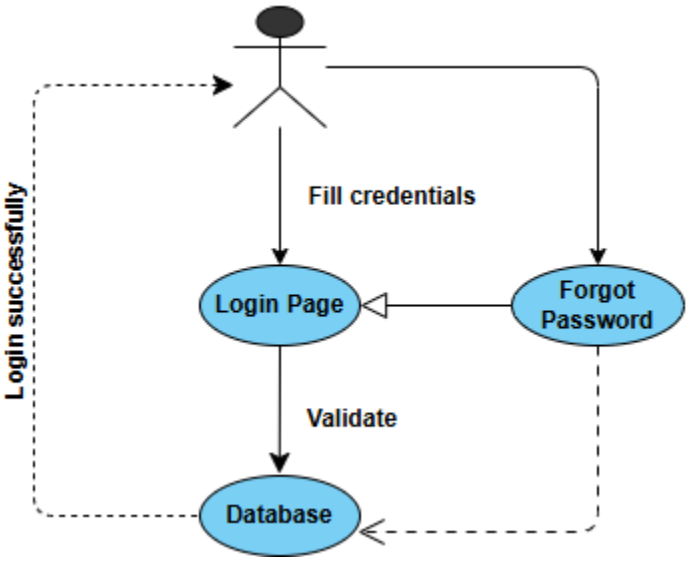

Figure 91 Login Collaboration Diagram

8.10.3 Message

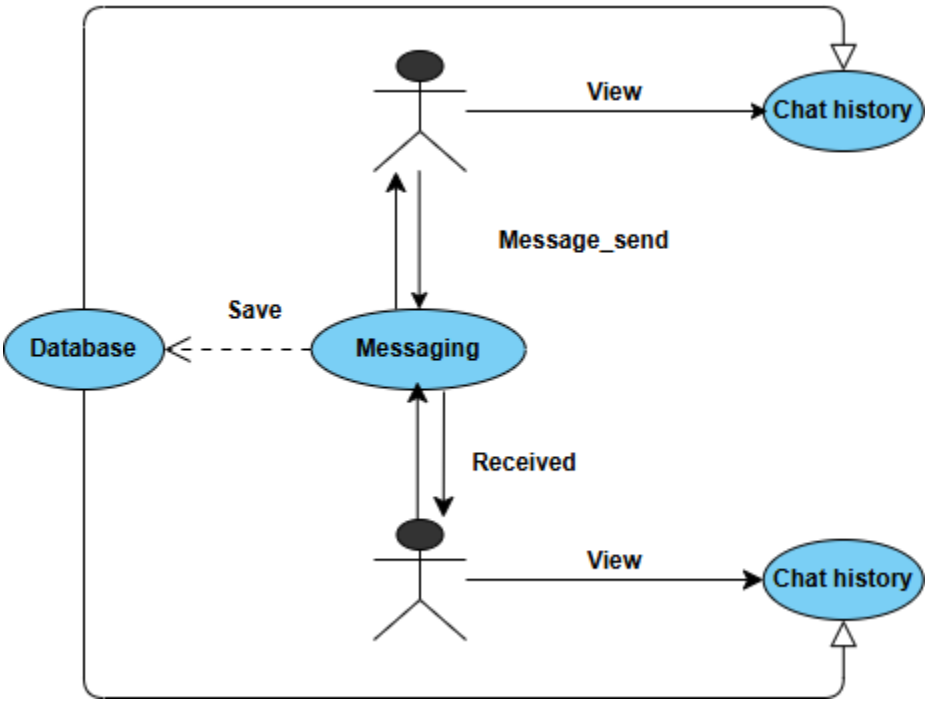

Figure 92 Message Collaboration Diagram

8.10.4 Event Management and Scheduling

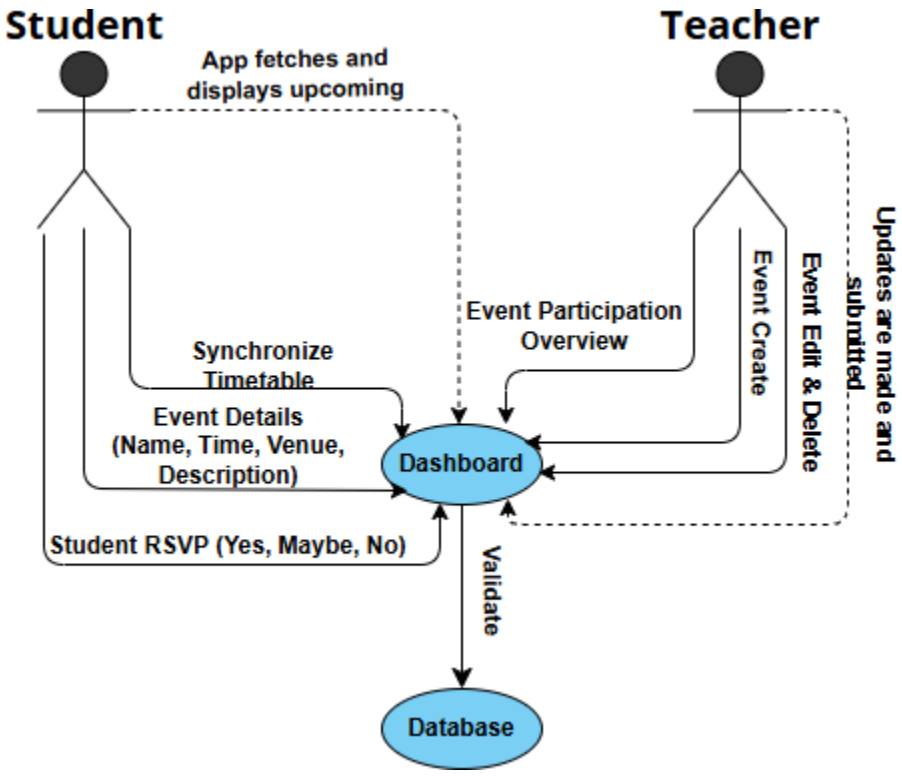

Figure 93 Event Management and Scheduling Collaboration Diagram

8.10.5 Job Board

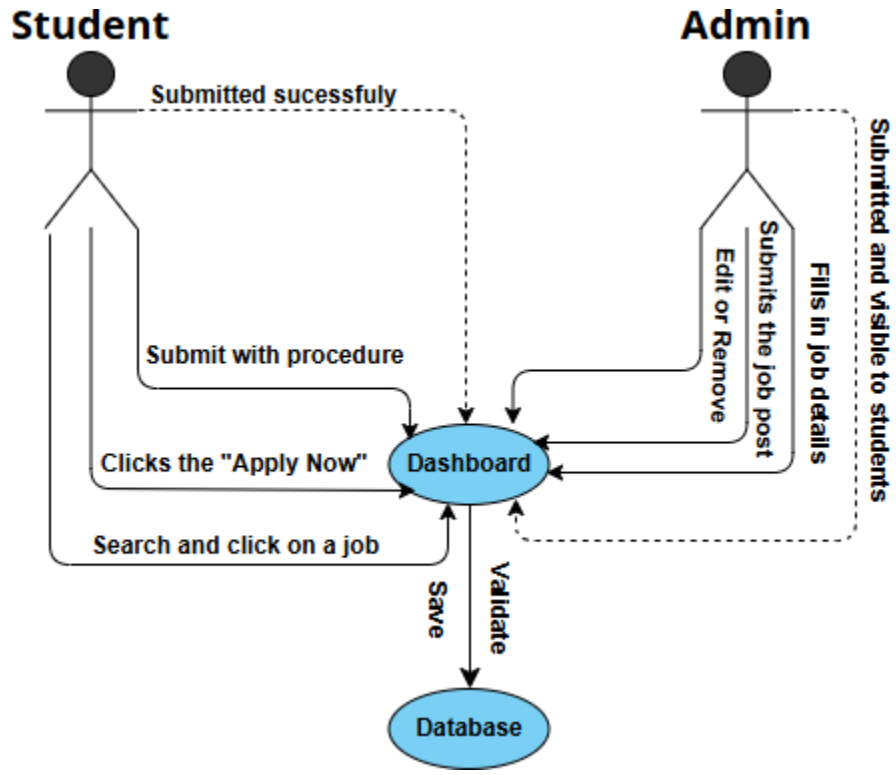

Figure 94 Job Board Collaboration Diagram

8.10.6 Carpool

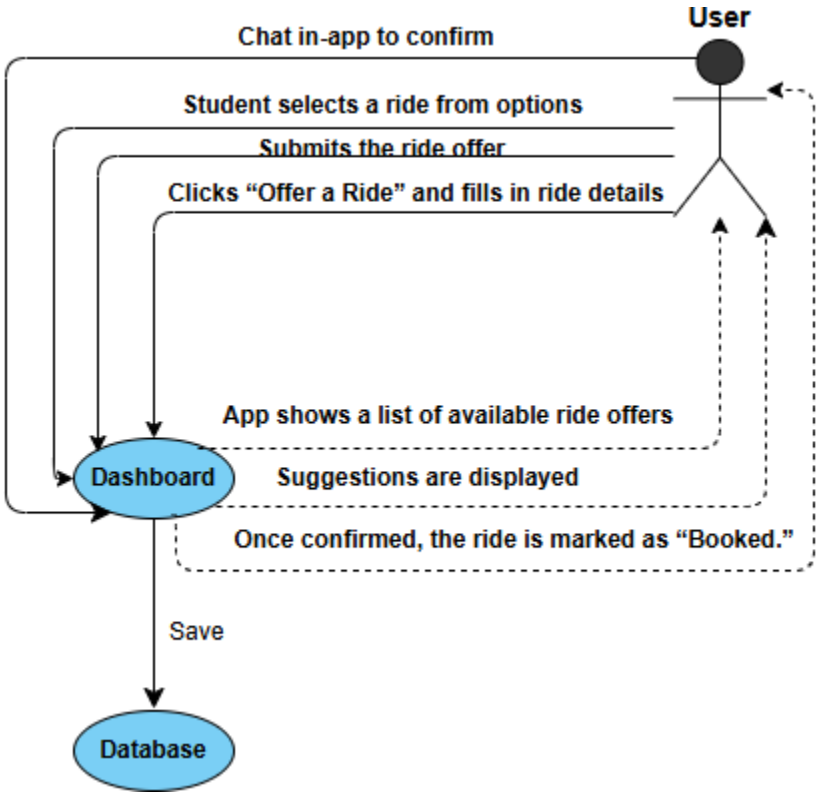

Figure 95 Carpool Collaboration Diagram

8.10.7 Parent Monitoring

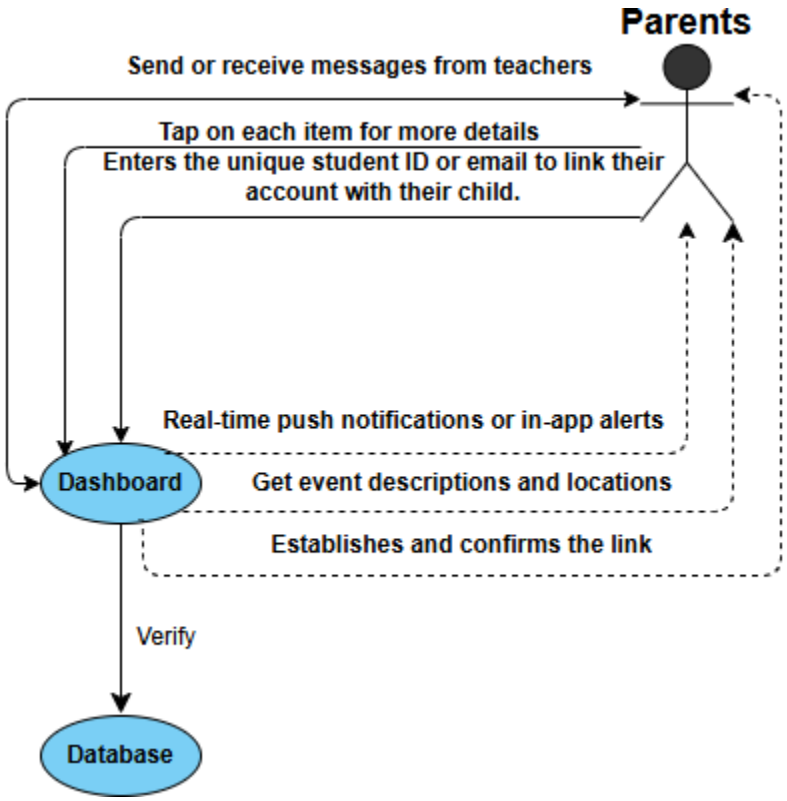

Figure 96 Parent Monitoring Collaboration Diagram

8.10.8 Admin

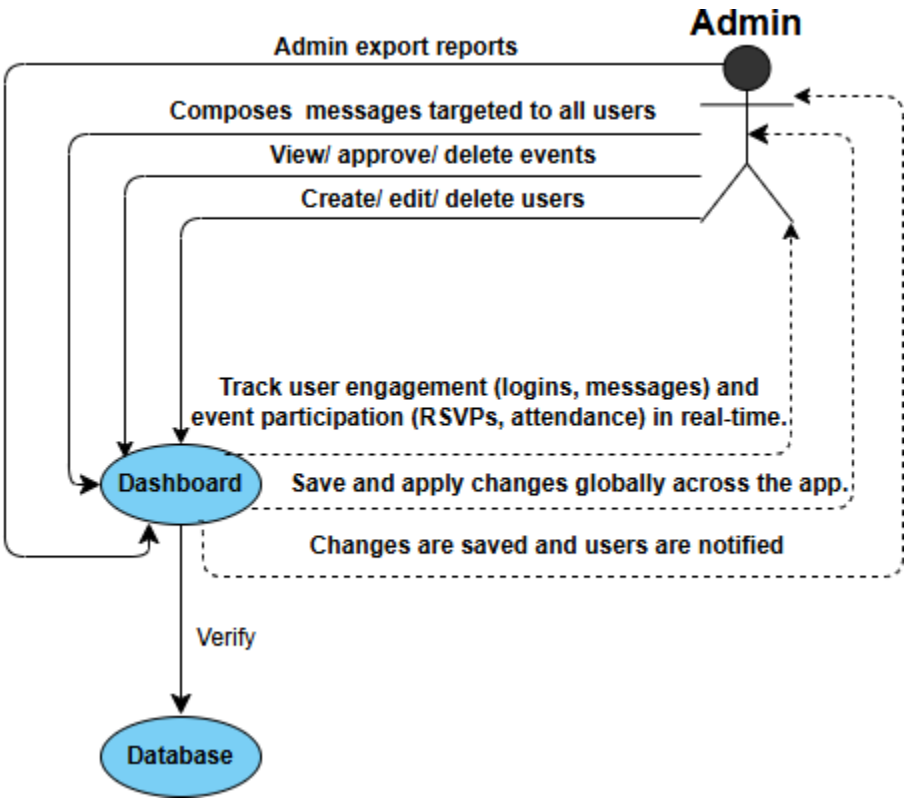

Figure 97 Admin Collaboration Diagram

8.11 Object Diagram

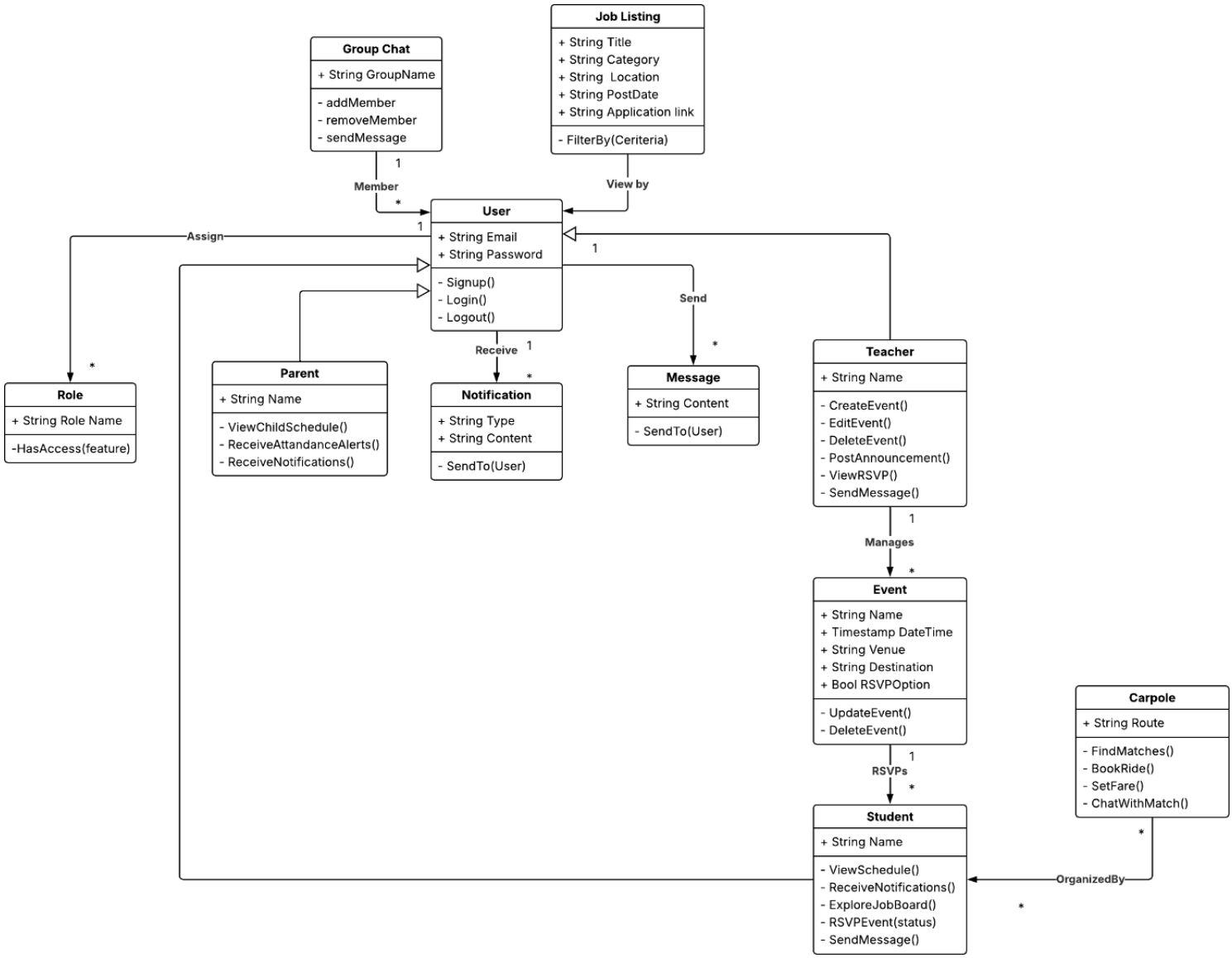

Figure 98 Object Diagram

Test Cases

9. Test cases

9.1 Login

Test Case Name	Test Admin Secure Login	Test Case Description	This test case is use to test the functionality of LogIn feature.		
Created By	Sandeep & Vivek	Verion	1	Date Tested	29 Dec 2024
ID:	TC_01				
S #	Prerequisites:			Test Scenario	Verify that the admin secure login functionality
1	Admin user account exists in the system.				
2	Admin is not logged into the system.				
3	Admin has valid credentials (email and password).				

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to the login page.	Login form is displayed.	Login form displayed successfully.	Pass
2	Enter valid credentials and submit.	Redirected to the admin dashboard.	Admin logged in successfully.	fail
3	Enter invalid credentials (e.g., incorrect password).	Error: "Invalid username or password."	Error message displayed correctly.	fail
4	Leave fields empty and attempt to log in.	Error: "Fields must not be empty."	Error message displayed correctly.	Pass

Table 96 Login TC01

Test Case Name	Test Admin Secure Login	Test Case Description	This test case is use to test the functionality of LogIn feature.		
Created By	Sandeep & Vivek	Verion	1.1	Date Tested	29 Dec 2024
ID:	TC_01				
S #	Prerequisites:			Test Scenario	Verify that the admin secure login functionality
1	Admin user account exists in the system.				
2	Admin is not logged into the system.				
3	Admin has valid credentials (email and password).				

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to the login page.	Login form is displayed.	Login form displayed successfully.	Pass
2	Enter valid credentials and submit.	Redirected to the admin dashboard.	Admin logged in successfully.	Pass
3	Enter invalid credentials (e.g., incorrect password).	Error: "Invalid username or password."	Error message displayed correctly.	Pass
4	Leave fields empty and attempt to log in.	Error: "Fields must not be empty."	Error message displayed correctly.	Pass

Table 97 Login TC 01 V1.1

9.2 Signup

Test Case Name	User Signup	Test Case Description	This test case verifies the functionality of the User's Signup feature.		
Created By	Sandeep & Vivek	Verion	1	Date Tested	29 Dec 2024
ID:	TC_02.0				
S #	Prerequisites:				
1	Teacher does not have an existing account.				
2	The system's email notification service is functional.				
			Test Scenario	Verify the user signup functionality for successful account creation.	

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to the signup page.	System displays a signup form requesting personal details and email address.	Signup form displayed successfully.	Pass
2	Fill out the form with valid information and submit.	System validates the information, creates the account, and sends a verification email.	Account created and email sent.	Pass
3	Click the verification link in the email.	System activates the account and confirms successful registration.	Account activated successfully.	fail
4	Submit incomplete or invalid information in the signup form.	System shows error messages and requests correct data.	Error message displayed correctly.	fail
5	Simulate a system error during signup.	System displays "Signup temporarily unavailable. Please try again later."	Error message displayed correctly.	fail

Table 98 Signup TC 02

Test Case Name	User Signup	Test Case Description	This test case verifies the functionality of the User's Signup feature.		
Created By	Sandeep & Vivek	Verion	1.1	Date Tested	29 Dec 2024
ID:	TC_02				
S #	Prerequisites:				
1	Teacher does not have an existing account.				
2	The system's email notification service is functional.				
			Test Scenario	Verify the user signup functionality for successful account creation.	

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Navigate to the signup page.	System displays a signup form requesting personal details and email address.	Signup form displayed successfully.	Pass
2	Fill out the form with valid information and submit.	System validates the information, creates the account, and sends a verification email.	Account created and email sent.	Pass
3	Click the verification link in the email.	System activates the account and confirms successful registration.	Account activated successfully.	Pass
4	Submit incomplete or invalid information in the signup form.	System shows error messages and requests correct data.	Error message displayed correctly.	Pass
5	Simulate a system error during signup.	System displays "Signup temporarily unavailable. Please try again later."	Error message displayed correctly.	Pass

Table 99 Login TC 01 V1.1

9.3 Forget Password

Test Case Name	Forget Password	Test Case Description	This test case verifies the functionality of the Forget Password feature.		
Created By	Sandeep & Vivek	Verion	1	Date Tested	29 Dec 2024
ID:	TC_04.0				
S #	Prerequisites:				
1	The student has an existing account.				
2	The student's email is registered in the system.			Test Scenario	Verify that the "Forgot Password" feature allows students to securely reset their password.
3	The system has the capability to send email notifications.				

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Click on the "Forgot Password" link on the login page.	System prompts the user to enter their registered email address.	Prompt displayed successfully.	Pass
2	Enter a registered email address and submit the request.	System verifies the email and sends a password reset link to the provided email.	Password reset link sent to the email.	fail
3	Enter an unregistered or invalid email address and submit the request.	System displays the message: "Email not found. Please try again."	Error message displayed correctly.	fail
4	Click on the password reset link sent to the registered email.	User is redirected to a secure page to create a new password.	Redirected to the secure password reset page.	fail
5	Enter and confirm a new password.	System updates the password securely in the database and confirms the reset.	Password updated successfully, confirmation shown.	fail

Table 100 Forgot Password TC 04

Test Case Name	Forget Password	Test Case Description	This test case verifies the functionality of the Forget Password feature.		
Created By	Sandeep & Vivek	Verion	1.1	Date Tested	29 Dec 2024
ID:	TC_04				
S #	Prerequisites:				
1	The student has an existing account.				
2	The student's email is registered in the system.			Test Scenario	Verify that the "Forgot Password" feature allows students to securely reset their password.
3	The system has the capability to send email notifications.				

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Click on the "Forgot Password" link on the login page.	System prompts the user to enter their registered email address.	Prompt displayed successfully.	Pass
2	Enter a registered email address and submit the request.	System verifies the email and sends a password reset link to the provided email.	Password reset link sent to the email.	Pass
3	Enter an unregistered or invalid email address and submit the request.	System displays the message: "Email not found. Please try again."	Error message displayed correctly.	Pass
4	Click on the password reset link sent to the registered email.	User is redirected to a secure page to create a new password.	Redirected to the secure password reset page.	Pass
5	Enter and confirm a new password.	System updates the password securely in the database and confirms the reset.	Password updated successfully, confirmation shown.	Pass

Table 101 Forgot Password TC 04 V1.1

9.4 Event Management

Test Case Name	Event & Timetable Manage	Test Case Description	This test case is use to test the functionality of Add events and Timetable.	
Created By	Sandeep & Vivek	Verion	1	Date Tested: 29 Dec 2024

ID:	TC_05.0
S #	Prerequisites:
1	The teacher's account exists in the system and is logged
2	Students are registered in the system and can access their accounts.
3	A class timetable exists and is synchronized in the system.

Test Scenario	To verify that the system allows teachers to create, edit, and manage events while students can view event & timetables.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Teacher accesses the "Create Event" page and enters event details.	Event is created successfully, and all registered students are notified of the event.	Event created successfully, and notifications sent.	Pass
2	Teacher edits an existing event (e.g., updates time or venue).	The event is updated successfully, and concerned users are notified of the changes.	Event updated successfully, and notifications sent.	fail
3	Student navigates to the event page and selects "Yes" for RSVP.	RSVP status is updated, and the teacher is notified of the student's response.	RSVP status updated successfully, and teacher notified.	fail
4	Student selects "Maybe" or "No" for RSVP.	RSVP status is updated, and the teacher is notified of the student's response.	RSVP status updated successfully, and teacher notified.	Pass
5	Student views the timetable and checks for event notifications.	Event notifications display on the timetable for schedule management.	Notifications displayed on timetable successfully.	Pass

Table 102 Event TC 05

Test Case Name	Event & Timetable Manage	Test Case Description	This test case is use to test the functionality of Add events and Timetable.	
Created By	Sandeep & Vivek	Verion	1.1	Date Tested: 29 Dec 2024

ID:	TC_05
S #	Prerequisites:
1	The teacher's account exists in the system and is logged
2	Students are registered in the system and can access their accounts.
3	A class timetable exists and is synchronized in the system.

Test Scenario	To verify that the system allows teachers to create, edit, and manage events while students can view event & timetables.

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Teacher accesses the "Create Event" page and enters event details.	Event is created successfully, and all registered students are notified of the event.	Event created successfully, and notifications sent.	Pass
2	Teacher edits an existing event (e.g., updates time or venue).	The event is updated successfully, and concerned users are notified of the changes.	Event updated successfully, and notifications sent.	Pass
3	Student navigates to the event page and selects "Yes" for RSVP.	RSVP status is updated, and the teacher is notified of the student's response.	RSVP status updated successfully, and teacher notified.	Pass
4	Student selects "Maybe" or "No" for RSVP.	RSVP status is updated, and the teacher is notified of the student's response.	RSVP status updated successfully, and teacher notified.	Pass
5	Student views the timetable and checks for event notifications.	Event notifications display on the timetable for schedule management.	Notifications displayed on timetable successfully.	Pass

Table 103 Event TC 05 V1.1

9.5 Job Management

Test Case Name	Job Board	Test Case Description	This test case verifies the functionality of the Job Board feature.		
Created By	Sandeep & Vivek	Verion	1	Date Tested	29 Dec 2024
ID:	TC_09.0				
S #	Prerequisites:				
1	Student accounts exist in the system.		Test Scenario	Validates Job Board access, job search, filtering, and application redirection.	
2	Job board has valid job and internship postings.				
3	Students have access to the job board.				

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Log in as a student and access the Job Board feature.	Job board page is displayed with all job and internship postings.	Job board displayed successfully with postings.	Pass
2	Search for job postings using filters (e.g., category, location).	Relevant job postings are displayed based on the selected filters.	Filtering and searching worked as expected.	fail
3	View details of a specific job posting.	Job details (title, description, requirements, application process) are displayed.	Job details displayed accurately.	fail
4	Click on the application link for a job posting.	User is redirected to the application process or relevant information page.	Redirected to the correct application link.	fail
5	Attempt to access the job board without logging in.	System displays an error message: "You must log in to access the Job Board."	Error message: "You must log in to access the Job Board" displayed.	Pass

Table 104 Job Board TC 09

9.6 Timetable Management

Test Case Name	Event & Timetable Manage	Test Case Description	This test case is use to test the functionality of Add events and Timetable.		
Created By	Sandeep & Vivek	Verion	1.1	Date Tested	29 Dec 2024
ID:	TC_05				
S #	Prerequisites:				
1	The teacher's account exists in the system and is logged				
2	Students are registered in the system and can access their accounts.		Test Scenario	To verify that the system allows teachers to create, edit, and manage events while students can view event & timetables.	
3	A class timetable exists and is synchronized in the system.				

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Teacher accesses the "Create Event" page and enters event details.	Event is created successfully, and all registered students are notified of the event.	Event created successfully, and notifications sent.	Pass
2	Teacher edits an existing event (e.g., updates time or venue).	The event is updated successfully, and concerned users are notified of the changes.	Event updated successfully, and notifications sent.	Pass
3	Student navigates to the event page and selects "Yes" for RSVP.	RSVP status is updated, and the teacher is notified of the student's response.	RSVP status updated successfully, and teacher notified.	Pass
4	Student selects "Maybe" or "No" for RSVP.	RSVP status is updated, and the teacher is notified of the student's response.	RSVP status updated successfully, and teacher notified.	Pass
5	Student views the timetable and checks for event notifications.	Event notifications display on the timetable for schedule management.	Notifications displayed on timetable successfully.	Pass

Table 105 Timetable TC 10

9.7 Job Board

Test Case Name	Job Board	Test Case Description	This test case verifies the functionality of the Job Board feature.		
Created By	Sandeep & Vivek	Verion	1.1	Date Tested	29 Dec 2024
ID:	TC_09				
S #	Prerequisites:				
1	Student accounts exist in the system.		Test Scenario	Validates Job Board access, job search, filtering, and application redirection.	
2	Job board has valid job and internship postings.				
3	Students have access to the job board.				

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Log in as a student and access the Job Board feature.	Job board page is displayed with all job and internship postings.	Job board displayed successfully with postings.	Pass
2	Search for job postings using filters (e.g., category, location).	Relevant job postings are displayed based on the selected filters.	Filtering and searching worked as expected.	Pass
3	View details of a specific job posting.	Job details (title, description, requirements, application process) are displayed.	Job details displayed accurately.	Pass
4	Click on the application link for a job posting.	User is redirected to the application process or relevant information page.	Redirected to the correct application link.	Pass
5	Attempt to access the job board without logging in.	System displays an error message: "You must log in to access the Job Board."	Error message: "You must log in to access the Job Board" displayed.	Pass

Table 106 Job Board TC 09

9.7 Admin

Test Case Name	Admin Management	Test Case Description	Verifies that admin can view and delete any content posted by students and teachers (e.g., posts, timetables, events).		
Created By	Sandeep & Vivek	Verion	1	Date Tested	20 dec 2024
ID:	TC_16				
S #	Prerequisites:				
1	User is logged into the UniVersa app.		Test Scenario	Admin can manage and remove content from the app.	
2	Internet connectivity is available.				
3	Admin has full access permissions.				

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Admin views a list of all content (posts, timetables, items).	All user-generated content is visible to admin.	Content list shown.	Pass
2	Admin deletes a student's timetable/any posted item.	content is removed and not visible to any user.	Deleted successfully.	fail
3	Admin deletes a teacher's Content.	Item removed instantly.	Removed as expected.	fail
4	Admin deletes a post with inappropriate content.	Post deleted and confirmation shown.	Post removed..	fail
5	Admin tries to delete during a simulated server issue.	System shows: "Unable to delete. Try again later."	Error handled properly.	Pass

Table 107 Admin Management TC 16

Test Case Name	Admin Management	Test Case Description	Verifies that admin can view and delete any content posted by students and teachers (e.g., posts, timetables, events).	
Created By	Sandeep & Vivek	Verion	1.1	Date Tested: 4 jan 2025
ID:	TC_16			

S #	Prerequisites:	Test Scenario	Admin can manage and remove content from the app.
1	User is logged into the UniVersa app.		
2	Internet connectivity is available.		
3	Admin has full access permissions.		

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Admin views a list of all content (posts, timetables, items).	All user-generated content is visible to admin.	Content list shown.	Pass
2	Admin deletes a student's timetable/any posted item.	content is removed and not visible to any user.	Deleted successfully.	Pass
3	Admin deletes a teacher's Content.	Item removed instantly.	Removed as expected.	Pass
4	Admin deletes a post with inappropriate content.	Post deleted and confirmation shown.	Post removed..	Pass
5	Admin tries to delete during a simulated server issue.	System shows: "Unable to delete. Try again later."	Error handled properly.	Pass

Table 108 Admin Management TC 16

9.8 Marketplace

Test Case Name	Marketplace Item Management	Test Case Description	This test case validates the core functionality of the UniVersa Marketplace, including adding, updating, and deleting items.	
Created By	Sandeep & Vivek	Verion	1	Date Tested: 20 Apr 2025
ID:	TC_11.0			

S #	Prerequisites:	Test Scenario	Validates listing, editing, and removal of items in the marketplace module.
1	Users (students) are registered and logged into the UniVersa app.		
2	Internet connectivity is available.		
3	User has access permissions for the marketplace.		

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	User navigates to "Add Item" page and enters valid details with an image.	Item listed successfully and visible in marketplace.	Item listed and visible.	Pass
2	User updates item price and description.	Item updated and reflects changes in marketplace.	Item updated successfully.	fail
3	User tries to add an item with missing price.	System shows validation error and blocks submission.	Error shown as expected.	fail
4	User deletes an existing item.	System confirms deletion and item disappears from marketplace.	Item deleted successfully.	fail
5	User attempts deletion during server error simulation.	System displays error message.	“Unable to delete item. Please try again later.” shown.	Pass

Table 109 Marketplace TC 11

Test Case Name	Marketplace Item Management	Test Case Description	This test case validates the core functionality of the UniVersa Marketplace, including adding, updating, and deleting items.		
Created By	Sandeep & Vivek	Verion	1.1	Date Tested	20 Apr 2025
ID:	TC_11				
S #	Prerequisites:				
1	Users (students) are registered and logged into the UniVersa app.		Test Scenario	Validates listing, editing, and removal of items in the marketplace module.	
2	Internet connectivity is available.				
3	User has access permissions for the marketplace.				

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	User navigates to "Add Item" page and enters valid details with an image.	Item listed successfully and visible in marketplace.	Item listed and visible.	Pass
2	User updates item price and description.	Item updated and reflects changes in marketplace.	Item updated successfully.	Pass
3	User tries to add an item with missing price.	System shows validation error and blocks submission.	Error shown as expected.	Pass
4	User deletes an existing item.	System confirms deletion and item disappears from marketplace.	Item deleted successfully.	Pass
5	User attempts deletion during server error simulation.	System displays error message.	"Unable to delete item. Please try again later." shown.	Pass

Table 110 Marketplace TC 11 V1.1

9.9 Carpool

Test Case Name	Carpoll	Test Case Description	This test case ensures the functionality of the carpool feature, which allows students to coordinate rides, share locations, and communicate in-app.		
Created By	Sandeep & Vivek	Verion	1	Date Tested	29 Dec 2024
ID:	TC_07.0				
S #	Prerequisites:				
1	User has a valid account and is logged into the system.		Test Scenario	Tests carpool request creation, matching, updates, and communication.	
2	Location permissions are enabled.				
3	Network connection is stable.				

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	User creates a carpool request with location, route, and schedule details.	Carpool request is created successfully and visible to other users.	Carpool request created and visible to others.	Pending
2	User views carpool matches based on proximity, route, and schedule.	System displays a list of potential matches sorted by relevance.	Matches displayed accurately based on route and timing.	fail
3	User selects a carpool match and initiates in-app communication with the rider.	In-app chat opens, and user is able to send and receive messages.	Chat functionality worked as expected.	Pass
4	Rider updates ride details such as timing or pickup location.	System notifies all participants of the updated ride details.	Notifications sent to all participants successfully.	fail
5	Rider or user cancels a carpool request.	System removes the carpool request and notifies affected users.	Request canceled and notifications sent successfully.	pass

Table 111 Carpool TC 07

Test Case Name	Carpoll	Test Case Description	This test case ensures the functionality of the carpool feature, which allows students to coordinate rides, share locations, and communicate in-app.	
Created By	Sandeep & Vivek	Verion	1.1	Date Tested: 29 Dec 2024
ID:	TC_07			
S #	Prerequisites:			
1	User has a valid account and is logged into the system.		Test Scenario	Tests carpool request creation, matching, updates, and communication.
2	Location permissions are enabled.			
3	Network connection is stable.			

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	User creates a carpool request with location, route, and schedule details.	Carpool request is created successfully and visible to other users.	Carpool request created and visible to others.	Pass
2	User views carpool matches based on proximity, route, and schedule.	System displays a list of potential matches sorted by relevance.	Matches displayed accurately based on route and timing.	Pass
3	User selects a carpool match and initiates in-app communication with the rider.	In-app chat opens, and user is able to send and receive messages.	Chat functionality worked as expected.	Pass
4	Rider updates ride details such as timing or pickup location.	System notifies all participants of the updated ride details.	Notifications sent to all participants successfully.	Pass
5	Rider or user cancels a carpool request.	System removes the carpool request and notifies affected users.	Request canceled and notifications sent successfully.	Pass

Table 112 Carpool TC 07 V1.1

9.10 Group Chat

Test Case Name	Messaging System and Group chat	Test Case Description	This test case validates the functionality of the messaging system, including direct messaging, group chats, media sharing, and message encryption.	
Created By	Sandeep & Vivek	Verion	1	Date Tested: 29 Dec 2024
ID:	TC_06.0			
S #	Prerequisites:			
1	Users (students, teachers, and parents) are registered and have access to their accounts.		Test Scenario	Validates messaging system features and highlights issues with broken URLs.
2	Internet connectivity is available.			
3	User permissions are configured for messaging (e.g., access to private messaging or group chats).			

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	User navigates to the "Messages" section and sends a direct message to another user.	Message sent and received successfully.	Message sent and received successfully.	Pass
2	User creates a group chat, adds multiple participants, and sends a message.	Group chat created, and message delivered to all participants.	Group chat created, and message received by all users.	Pass
3	User shares an image in a direct message.	Document uploaded and shared with all participants.	Document shared successfully.	Fail
4	User shares a document in a group chat.	RSVP updated, and teacher notified.	Link sent but failed to open due to broken URL.	Pass
5	User sends a link in a direct message.	Link sent successfully and accessible by the recipient.	Link sent but failed to open due to broken URL.	Fail

Table 113 Group Chat TC 06

Test Case Name	Messaging System and Group chat	Test Case Description	This test case validates the functionality of the messaging system, including direct messaging, group chats, media sharing, and message encryption.	
Created By	Sandeep & Vivek	Verion	1.1	Date Tested 29 Dec 2024
ID:	TC_06			
S #	Prerequisites:			
1	Users (students, teachers, and parents) are registered and have access to their accounts.		Test Scenario	Validates messaging system features and highlights issues with broken URLs.
2	Internet connectivity is available.			
3	User permissions are configured for messaging (e.g., access to private messaging or group chats).			

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	User navigates to the "Messages" section and sends a direct message to another user.	Message sent and received successfully.	Message sent and received successfully.	Pass
2	User creates a group chat, adds multiple participants, and sends a message.	Group chat created, and message delivered to all participants.	Group chat created, and message received by all users.	Pass
3	User shares an image in a direct message.	Document uploaded and shared with all participants.	Document shared successfully.	Pass
4	User shares a document in a group chat.	RSVP updated, and teacher notified.	Link sent but failed to open due to broken URL.	Pass
5	User sends a link in a direct message.	Link sent successfully and accessible by the recipient.	Link sent but failed to open due to broken URL.	Pass

Table 114 Group Chat TC 06 V1.1

9.11 Messaging Service

Test Case Name	Messaging System and Group chat	Test Case Description	This test case validates the functionality of the messaging system, including direct messaging, group chats, media sharing, and message encryption.	
Created By	Sandeep & Vivek	Verion	1	Date Tested 29 Dec 2024
ID:	TC_06.0			
S #	Prerequisites:			
1	Users (students, teachers, and parents) are registered and have access to their accounts.		Test Scenario	Validates messaging system features and highlights issues with broken URLs.
2	Internet connectivity is available.			
3	User permissions are configured for messaging (e.g., access to private messaging or group chats).			

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	User navigates to the "Messages" section and sends a direct message to another user.	Message sent and received successfully.	Message sent and received successfully.	Pass
2	User creates a group chat, adds multiple participants, and sends a message.	Group chat created, and message delivered to all participants.	Group chat created, and message received by all users.	Pass
3	User shares an image in a direct message.	Document uploaded and shared with all participants.	Document shared successfully.	Fail
4	User shares a document in a group chat.	RSVP updated, and teacher notified.	Link sent but failed to open due to broken URL.	Pass
5	User sends a link in a direct message.	Link sent successfully and accessible by the recipient.	Link sent but failed to open due to broken URL.	Fail

Table 115 Messaging Service TC 06

Test Case Name	Messaging System and Group chat	Test Case Description	This test case validates the functionality of the messaging system, including direct messaging, group chats, media sharing, and message encryption.	
Created By	Sandeep & Vivek	Verion	1.1	Date Tested 29 Dec 2024
ID:	TC_06			
S #	Prerequisites:		Test Scenario	Validates messaging system features and highlights issues with broken URLs.
1	Users (students, teachers, and parents) are registered and have access to their accounts.			
2	Internet connectivity is available.			
3	User permissions are configured for messaging (e.g., access to private messaging or group chats).			

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	User navigates to the "Messages" section and sends a direct message to another user.	Message sent and received successfully.	Message sent and received successfully.	Pass
2	User creates a group chat, adds multiple participants, and sends a message.	Group chat created, and message delivered to all participants.	Group chat created, and message received by all users.	Pass
3	User shares an image in a direct message.	Document uploaded and shared with all participants.	Document shared successfully.	Pass
4	User shares a document in a group chat.	RSVP updated, and teacher notified.	Link sent but failed to open due to broken URL.	Pass
5	User sends a link in a direct message.	Link sent successfully and accessible by the recipient.	Link sent but failed to open due to broken URL.	Pass

Table 116 Messaging Service TC 06

9.12 Message Request

Test Case Name	Message Request to Teacher	Test Case Description	This test case verifies the functionality of the Message Request to Teacher feature.	
Created By	Sandeep & Vivek	Verion	1	Date Tested 29 Dec 2024
ID:	TC_10.0			
S #	Prerequisites:		Test Scenario	Validates teacher messaging requests, duplicate prevention, and error handling.
1	The student is logged in to the system.			
2	The system has a list of teachers linked to the student's			
3	The messaging module is functional.			

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Access the messaging module.	System displays a list of teachers and a "Request Message" option for each teacher.	List of teachers and request option displayed.	Pass
2	Select a teacher and submit a message request.	System sends the request and displays a confirmation message to the student.	Request sent successfully, confirmation shown.	Pass
3	Attempt to send a duplicate message request to the same teacher.	System displays the message: "You have already requested to message this teacher.".	Error message displayed correctly.	fail
4	Simulate a system error during the message request submission.	System displays the message: "Unable to send message request at this time. Please try again later."	Error message displayed correctly.	fail

Table 117 Message Request TC 10

Test Case Name	Message Request to Teacher	Test Case Description	This test case verifies the functionality of the Message Request to Teacher feature.		
Created By	Sandeep & Vivek	Verion	1.1	Date Tested	29 Dec 2024
ID:	TC_10				
S #	Prerequisites:		Test Scenario	Validates teacher messaging requests, duplicate prevention, and error handling.	
1	The student is logged in to the system.				
2	The system has a list of teachers linked to the student's				
3	The messaging module is functional.				

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Access the messaging module.	System displays a list of teachers and a "Request Message" option for each teacher.	List of teachers and request option displayed.	Pass
2	Select a teacher and submit a message request.	System sends the request and displays a confirmation message to the student.	Request sent successfully, confirmation shown.	Pass
3	Attempt to send a duplicate message request to the same teacher.	System displays the message: "You have already requested to message this teacher.".	Error message displayed correctly.	Pass
4	Simulate a system error during the message request submission.	System displays the message: "Unable to send message request at this time. Please try again later."	Error message displayed correctly.	Pass

Table 118 Message Request TC 10 V1.1

9.13 Lost and Found

Test Case Name	Lost and Found Functionality	Test Case Description	This test case validates the posting and deletion of lost items by users in the UniVersa app.		
Created By	Sandeep & Vivek	Verion	1	Date Tested	15 june 2025
ID:	TC_13.0				
S #	Prerequisites:		Test Scenario	Validates the ability to post and delete lost items within the Lost and Found module.	
1	User is logged into the UniVersa app.				
2	Internet connectivity is available.				
3	Lost and Found feature is accessible from the main dashboard or menu.				

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	User navigates to the Lost and Found module and clicks "Add Lost Item".	Form is displayed with fields for title, description, and image upload.	Form displayed successfully.	Pass
2	User fills the form with valid details and uploads an image, then submits it.	Lost item is saved and shown in the Lost and Found feed.	Item successfully posted and visible to others.	fail
3	User opens their posted item and clicks "Delete".	Item is removed from the feed and deleted from the database.	Item deleted successfully and no longer visible in the feed.	fail

Table 119 Lost and Found TC 13

Test Case Name	Lost and Found Functionality	Test Case Description	This test case validates the posting and deletion of lost items by users in the UniVersa app.		
Created By	Sandeep & Vivek	Verion	1.1	Date Tested	15 june 2025
ID:	TC_13				
S #	Prerequisites:				
1	User is logged into the UniVersa app.		Test Scenario	Validates the ability to post and delete lost items within the Lost and Found module.	
2	Internet connectivity is available.				
3	Lost and Found feature is accessible from the main dashboard or menu.				

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	User navigates to the Lost and Found module and clicks "Add Lost Item".	Form is displayed with fields for title, description, and image upload.	Form displayed successfully.	Pass
2	User fills the form with valid details and uploads an image, then submits it.	Lost item is saved and shown in the Lost and Found feed.	Item successfully posted and visible to others.	Pass
3	User opens their posted item and clicks "Delete".	Item is removed from the feed and deleted from the database.	Item deleted successfully and no longer visible in the feed.	Pass

Table 120 Lost and Found TC 13 V1.1

9.14 Chatbot

Test Case Name	Chatbot Functionality	Test Case Description	This test case validates the chatbot's responses related to university academics, events, admissions, and general inquiries.		
Created By	Sandeep & Vivek	Verion	1	Date Tested	20 Apr 2025
ID:	TC_12.0				
S #	Prerequisites:				
1	User is logged into the UniVersa app.		Test Scenario	Validates prompt-based responses and interactions in the chatbot module.	
2	Internet connectivity is available.				
3	Chatbot feature is accessible from the main dashboard or menu.				

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	User initiates the chatbot from the app.	Chatbot opens and displays a Enter message with sample prompts.	Chatbot launched with prompts visible.	Pass
2	User asks about anything from saved prompts.	Chatbot returns accurate information from saved procedures.	message viewed successfully.	fail
3	User attempts to access chatbot without internet.	Error message is shown suggesting connectivity issue.	"Connection error. Please check your internet connection and try again." shown.	fail

Table 121 Chatbot TC 12

Test Case Name	Chatbot Functionality	Test Case Description	This test case validates the chatbot's responses related to university academics, events, admissions, and general inquiries.		
Created By	Sandeep & Vivek	Verion	1.1	Date Tested	20 Apr 2025
ID:	TC_12				
S #	Prerequisites:				
1	User is logged into the UniVersa app.		Test Scenario	Validates prompt-based responses and interactions in the chatbot module.	
2	Internet connectivity is available.				
3	Chatbot feature is accessible from the main dashboard or menu.				

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	User initiates the chatbot from the app.	Chatbot opens and displays a Enter message with sample prompts.	Chatbot launched with prompts visible.	Pass
2	User asks about anything from saved prompts.	Chatbot returns accurate information from saved procedures.	message viewed successfully.	Pass
3	User attempts to access chatbot without internet.	Error message is shown suggesting connectivity issue.	"Connection error. Please check your internet connection and try again." shown.	Pass

Table 122 Chatbot TC 12 V1.1

9.15 Resource Library

Test Case Name	Resource Library Functionality	Test Case Description	This test case validates the uploading and deletion of useful student resources in the UniVersa app.		
Created By	Sandeep & Vivek	Verion	1	Date Tested	15 june 2025
ID:	TC_14.0				
S #	Prerequisites:				
1	User is logged into the UniVersa app.		Test Scenario	Validates resource upload and deletion functionality in the Resource Library module.	
2	Internet connectivity is available.				
3	Resource Library feature is accessible from the main dashboard or menu..				

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	User navigates to the Resource Library and clicks "Upload Resource".	Upload form is displayed with fields for title, description, and file upload.	Form displayed successfully.	Pass
2	User uploads a valid PDF document with a title and description.	Resource is saved and appears in the Resource Library list.	Resource uploaded and visible to other users.	fail
3	User opens their uploaded resource and clicks "Delete".	Resource is removed from the list and deleted from the database.	Resource deleted successfully and removed from library.	Pass

Table 123 Resource Library TC 14

Test Case Name	Resource Library Functionality	Test Case Description	This test case validates the uploading and deletion of useful student resources in the UniVersa app.		
Created By	Sandeep & Vivek	Verion	1.1	Date Tested	15 june 2025
ID:	TC_14				
S #	Prerequisites:				
1	User is logged into the UniVersa app.		Test Scenario	Validates resource upload and deletion functionality in the Resource Library module.	
2	Internet connectivity is available.				
3	Resource Library feature is accessible from the main dashboard or menu..				

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	User navigates to the Resource Library and clicks "Upload Resource".	Upload form is displayed with fields for title, description, and file upload.	Form displayed successfully.	Pass
2	User uploads a valid PDF document with a title and description.	Resource is saved and appears in the Resource Library list.	Resource uploaded and visible to other users.	Pass
3	User opens their uploaded resource and clicks "Delete".	Resource is removed from the list and deleted from the database.	Resource deleted successfully and removed from library.	Pass

Table 124 Resource Library TC 14 V1.1

9.16 Parent Monitoring

Test Case Name	Parent Monitoring	Test Case Description	This test case verifies the functionality of the Parent Monitoring feature.		
Created By	Sandeep & Vivek	Verion	1	Date Tested	29 Dec 2024
ID:	TC_08.0				
S #	Prerequisites:				
1	Parent accounts exist in the system.		Test Scenario	Validates parent monitoring for access, notifications, and secure login.	
2	The system has child data linked to the parent accounts.				
3	Parents have valid credentials (email and password).				

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Log in with correct credentials.	Parent dashboard is displayed.	Parent dashboard displayed successfully.	Pass
2	View the child's class and event schedule.	Schedule is displayed with accurate data.	Schedule displayed correctly with all relevant details.	fail
3	Check notifications for attendance and school events.	Notifications are displayed accurately and on time.	Notifications displayed correctly and on time.	fail
4	Attempt to log in with incorrect credentials.	System displays a message: "Invalid email or password."	Error message: "Invalid email or password" displayed.	fail
5	Leave login fields empty and click Submit.	System displays a message: "Please enter email and password."	Error message: "Please enter email and password" shown.	Pass

Table 125 Parent Monitoring TC 08

Test Case Name	Parent Monitoring	Test Case Description	This test case verifies the functionality of the Parent Monitoring feature.		
Created By	Sandeep & Vivek	Verion	1.1	Date Tested	29 Dec 2024
ID:	TC_08				
S #	Prerequisites:				
1	Parent accounts exist in the system.		Test Scenario	Validates parent monitoring for access, notifications, and secure login.	
2	The system has child data linked to the parent accounts.				
3	Parents have valid credentials (email and password).				

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Log in with correct credentials.	Parent dashboard is displayed.	Parent dashboard displayed successfully.	Pass
2	View the child's class and event schedule.	Schedule is displayed with accurate data.	Schedule displayed correctly with all relevant details.	Pass
3	Check notifications for attendance and school events.	Notifications are displayed accurately and on time.	Notifications displayed correctly and on time.	Pass
4	Attempt to log in with incorrect credentials.	System displays a message: "Invalid email or password."	Error message: "Invalid email or password" displayed.	Pass
5	Leave login fields empty and click Submit.	System displays a message: "Please enter email and password."	Error message: "Please enter email and password" shown.	Pass

Table 126 Parent Monitoring TC 08 V1.1

9.17 Notification Service

Test Case Name	Notification Service	Test Case Description	Verifies that students can create, edit, and delete notes within the UniVersa app. This test case verifies the functionality of the notification system for different user roles (students, parents, teachers) when events or jobs are posted.	
Created By	Sandeep & Vivek	Verion	1	Date Tested: 2 june 2025

ID:	TC_19		
S #	Prerequisites:		
1	User is logged into the UniVersa app.	Test Scenario	Validates that students, parents, and teachers receive notifications for event/job postings.
2	The notification service is active and working.		
3	Events or job postings etc are available in the system.		

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Admin/Student/Teacher posts a new event.	System generates a notification for all users.	Notification received by all users.	Fail
2	User posts a new job opportunity.	System generates job notification for all users..	Notification received correctly.	Fail
3	Student checks notification panel.	List of new notifications is visible (event/job).	Notifications displayed as expected.	Fail
4	Parent receives a push notification for newly posted event.	Push notification arrives on the parent's device.	Error handled gracefully.	Fail
5	Attempt to simulate failure in notification delivery (e.g., network issue or service down).	System logs error and retries or informs user with message: "Failed to load notifications. Try again."	Notes visible correctly.	Fail
6	New user who joins after event/job post does not receive backdated notifications.	System does not show old event/job notification to new users unless reposted.	Correct behavior observed.	Fail

Table 127 Notification TC 19

Test Case Name	Notification Service	Test Case Description	Verifies that students can create, edit, and delete notes within the UniVersa app. This test case verifies the functionality of the notification system for different user roles (students, parents, teachers) when events or jobs are posted.	
Created By	Sandeep & Vivek	Verion	1.1	Date Tested: 7 june 2025

ID:	TC_19		
S #	Prerequisites:		
1	User is logged into the UniVersa app.	Test Scenario	Validates that students, parents, and teachers receive notifications for event/job postings.
2	The notification service is active and working.		
3	Events or job postings etc are available in the system.		

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Admin/Student/Teacher posts a new event.	System generates a notification for all users.	Notification received by all users.	Pass
2	User posts a new job opportunity.	System generates job notification for all users..	Notification received correctly.	Pass
3	Student checks notification panel.	List of new notifications is visible (event/job).	Notifications displayed as expected.	Pass
4	Parent receives a push notification for newly posted event.	Push notification arrives on the parent's device.	Error handled gracefully.	Pass
5	Attempt to simulate failure in notification delivery (e.g., network issue or service down).	System logs error and retries or informs user with message: "Failed to load notifications. Try again."	Notes visible correctly.	Pass
6	New user who joins after event/job post does not receive backdated notifications.	System does not show old event/job notification to new users unless reposted.	Correct behavior observed.	Pass

Table 128 Notification TC 19 V1.1

9.18 To-Do List

Test Case Name	To-Do List – Reminder Feature	Test Case Description	Verifies that student can create, edit, delete to-do items with reminders, and receive notifications on selected time/date.	
Created By	Sandeep & Vivek	Verion	1	Date Tested: 20 April 2025
ID:	TC_17			

S #	Prerequisites:		
1	User is logged into the UniVersa app.	Test Scenario	Student can manage reminders and get notifications.
2	Internet connectivity is available.		
3	Notification and reminder module is active.		

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Student adds a to-do with title, description, date & time.	To-do saved successfully and listed.	To-do added and shown.	Pass
2	Student edits the existing to-do (e.g., change time).	Changes saved and updated to-do shown.	Edited successfully.	fail
3	Student deletes a to-do item.	To-do removed from the list.	Deleted successfully.	fail
4	Time/date for reminder arrives.	Student receives a notification as reminder.	Notification received.	fail
5	Student tries to save without title or date/time.	System shows validation error.	Error displayed correctly.	fail

Table 129 To-Do List TC 17

Test Case Name	To-Do List – Reminder Feature	Test Case Description	Verifies that student can create, edit, delete to-do items with reminders, and receive notifications on selected time/date.	
Created By	Sandeep & Vivek	Verion	1.1	Date Tested: 2 May 2025
ID:	TC_17			

S #	Prerequisites:		
1	User is logged into the UniVersa app.	Test Scenario	Student can manage reminders and get notifications.
2	Internet connectivity is available.		
3	Notification and reminder module is active.		

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Student adds a to-do with title, description, date & time.	To-do saved successfully and listed.	To-do added and shown.	Pass
2	Student edits the existing to-do (e.g., change time).	Changes saved and updated to-do shown.	Edited successfully.	Pass
3	Student deletes a to-do item.	To-do removed from the list.	Deleted successfully.	Pass
4	Time/date for reminder arrives.	Student receives a notification as reminder.	Notification received.	Pass
5	Student tries to save without title or date/time.	System shows validation error.	Error displayed correctly.	Pass

Table 130 To-Do List TC 17 V1.1

9.19 Notes

Test Case Name	Notes Management – Student Notes	Test Case Description	Verifies that students can create, edit, and delete notes within the UniVersa app.		
Created By	Sandeep & Vivek	Verion	1	Date Tested	29 Dec 2024
ID:	TC_18				

S #	Prerequisites:		
1	User is logged into the UniVersa app.	Test Scenario	Student can manage personal notes inside the app.
2	Internet connectivity is available.		
3	Notes module is accessible and active.		

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Student adds a new note with title and content.	Note saved and shown in the notes list.	Note added successfully.	Pass
2	Student edits an existing note.	Note updated and changes reflected.	Note edited successfully.	Fail
3	Student deletes a note.	Note removed from the list.	Note deleted.	Fail
4	Student tries to save a note without a title/content.	System shows validation error.	Error message shown.	Fail
5	Student views saved notes.	All saved notes are displayed.	Notes visible correctly.	Fail

Table 131 Notes TC 18

Test Case Name	Notes Management – Student Notes	Test Case Description	Verifies that students can create, edit, and delete notes within the UniVersa app.		
Created By	Sandeep & Vivek	Verion	1.1	Date Tested	2 Jan 2025
ID:	TC_18				

S #	Prerequisites:		
1	User is logged into the UniVersa app.	Test Scenario	Student can manage personal notes inside the app.
2	Internet connectivity is available.		
3	Notes module is accessible and active.		

Step #	Step Details	Expected Results	Actual Results	Pass / Fail / Not executed / Suspended
1	Student adds a new note with title and content.	Note saved and shown in the notes list.	Note added successfully.	Pass
2	Student edits an existing note.	Note updated and changes reflected.	Note edited successfully.	Pass
3	Student deletes a note.	Note removed from the list.	Note deleted.	Pass
4	Student tries to save a note without a title/content.	System shows validation error.	Error message shown.	Pass
5	Student views saved notes.	All saved notes are displayed.	Notes visible correctly.	Pass

Table 132 Notes TC 18 V1.1

User Manual

UniVersa User Manual: Complete Tutorial

1. Introduction

UniVersa is a mobile application designed to streamline communication and coordination within universities for students, teachers, parents, and administrators. This comprehensive tutorial guides you through every step of using the app, from installation to mastering its features, ensuring a seamless experience. Access in-app tutorial videos in **Settings > Help Center** for visual guidance.

2. Getting Started

2.1 System Requirements

- **Device**: Android (version 5.0 or higher)
- **Internet**: Stable connection for real-time features (Wi-Fi or mobile data)
- **Storage**: Approximately 100 MB for installation

2.2 Installing UniVersa

1. Open the **Google Play Store** on your Android device.
2. Search for **UniVersa**.
3. Tap **Install** and wait for the download to complete.
4. Launch the app from your home screen or app drawer.

2.3 Account Creation

1. Open UniVersa and tap **Sign Up** on the welcome screen.
2. Enter your university-provided **email address**, a **secure password** (minimum 8 characters, including letters and numbers), and select your **role** (Student, Teacher, Parent, Admin).
3. Provide additional details: full name, phone number (optional), and university ID (if required).
4. Tap **Submit** to create your account.
5. Check your email for an **activation link** and click it to verify your account.
6. Return to the app, tap **Log In**, enter your email and password, and tap **Submit** to access your role-specific dashboard.

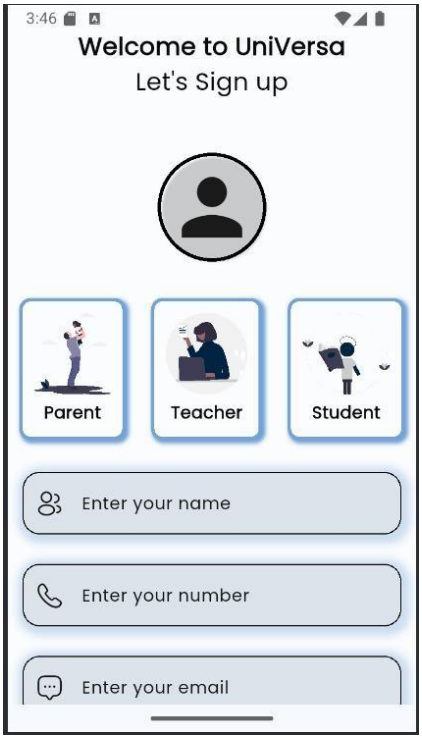

Figure 99 Account Creation – SignUp

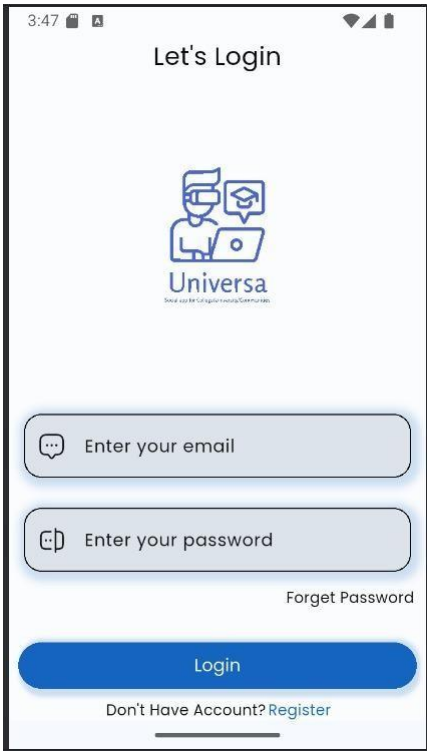

Figure 100 Account Creation – Login

3. Setting Up Your Profile

1. From the dashboard, tap the **Signup** icon in the bottom navigation bar.
2. Select **Profile > Add Profile**.
3. Upload a **profile picture** by tapping **Choose Image** and selecting a photo from your gallery (max size: 2 MB, formats: JPG/PNG).
4. Update details: full name, contact number, and email.
5. Set **notification preferences**: toggle options for events, messages, job board, and carpool alerts.
6. Tap **Save** to update your profile.

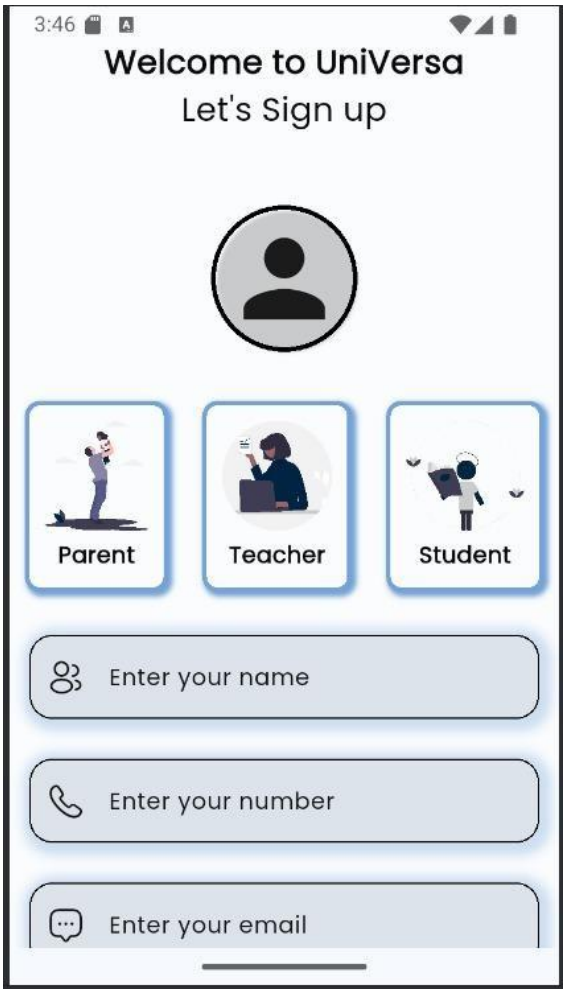

Figure 101 Setting Up Profile

4. Navigating the Dashboard

The UniVersa dashboard is your central hub, accessible after login. It features:

- **Quick Access Tiles**: View upcoming events, new messages, and tasks.
- **Bottom Navigation Bar**: Icons for Home, Events, Timetable, Messages, Job Board, Marketplace, Groups, Notifications, Lost and Found, Resources, To-Do List, Carpool, and Settings.
- **Role-Based Views**:
    - **Students**: Access all features.
    - **Teachers**: View Manage events, timetables, and job postings.
    - **Parents**: View schedules, events, and notifications.
    - **Admins**: Full access to manage users, events, timetables, and resources.
- Swipe left/right on the dashboard to view personalized updates or tap navigation icons to switch features.

Figure 103 Parent Dashboard

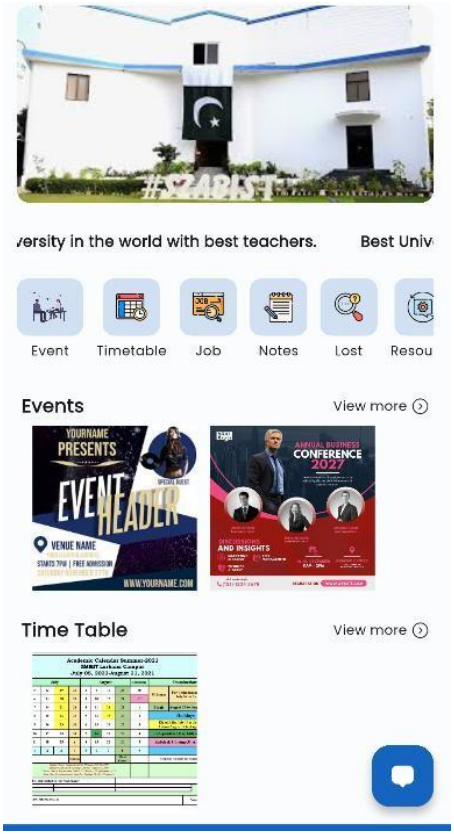

Figure 102 Student Dashboard

5. Complete Feature Tutorials

5.1 Events

For Teachers/Admins: Creating an Event

1. Tap **Events** in the navigation bar.
2. Select **Add Event**
3. Enter details:
    - **Title**: e.g., “CS101 Guest Lecture”
    - **Description**: Add event details.
    - **RSVP Option**: Toggle to enable/disable student RSVPs.
    - **Category**: Choose from Social, or Sports (admins can add categories in **Events > Manage Categories**).
4. Tap **Submit** to publish. Users receive notification.

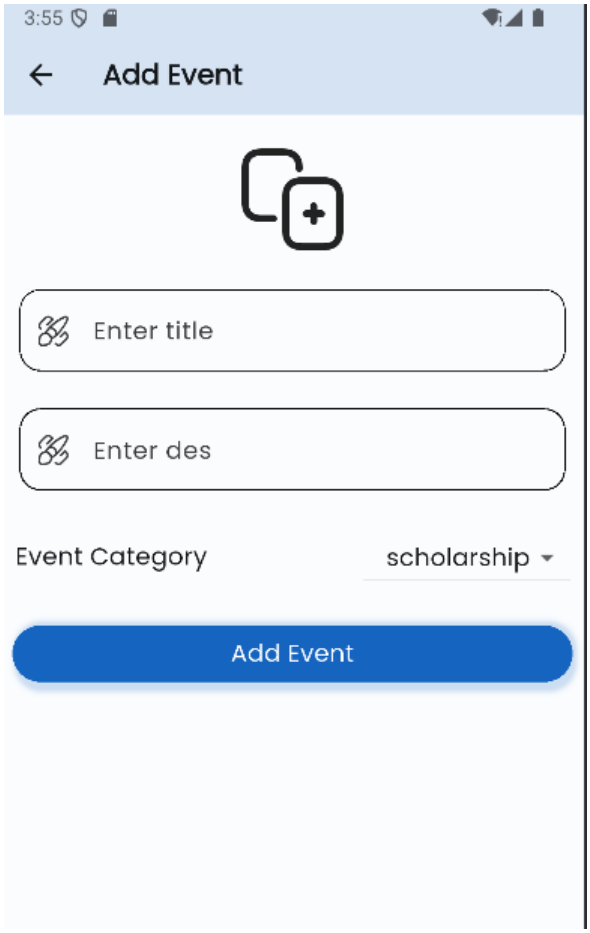

Figure 104 Add Event

Editing/Deleting an Event

1. Go to **Events > My Events**.
2. Select an event, tap **Edit**, update fields, and tap **Save**.
3. To delete, tap **Delete** and confirm. Users are notified of changes.

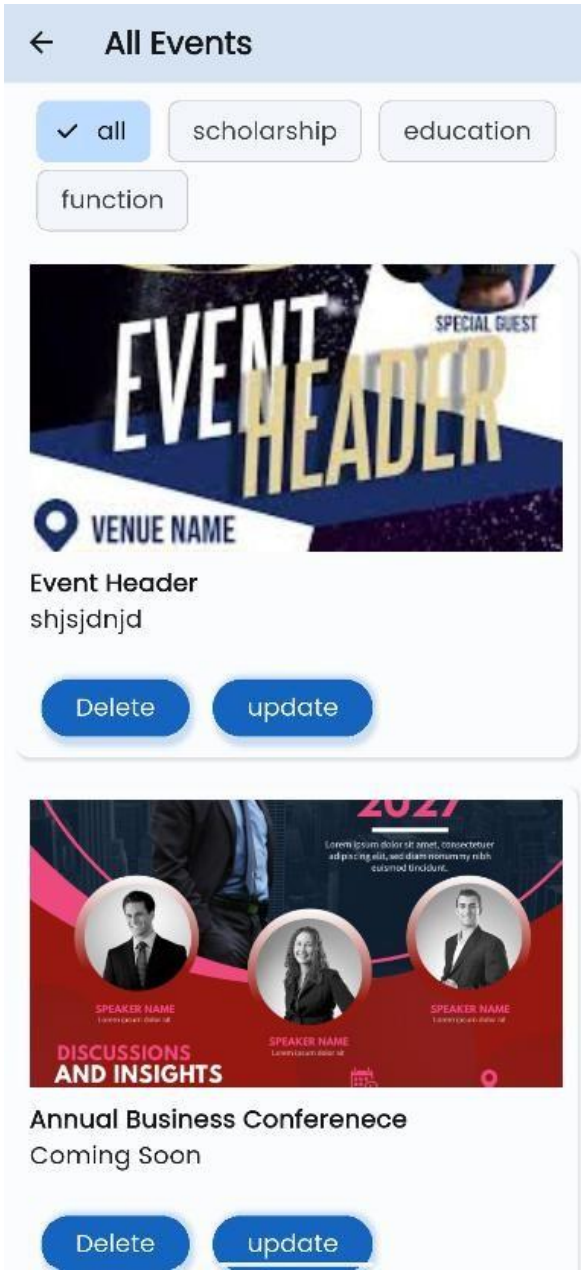

Figure 105 Edit Events

For Students: Viewing and RSVPing

1. Tap **Events** to view upcoming events.
2. Tap an event to see details (title, date, location, description).
3. Tap **Submit**.

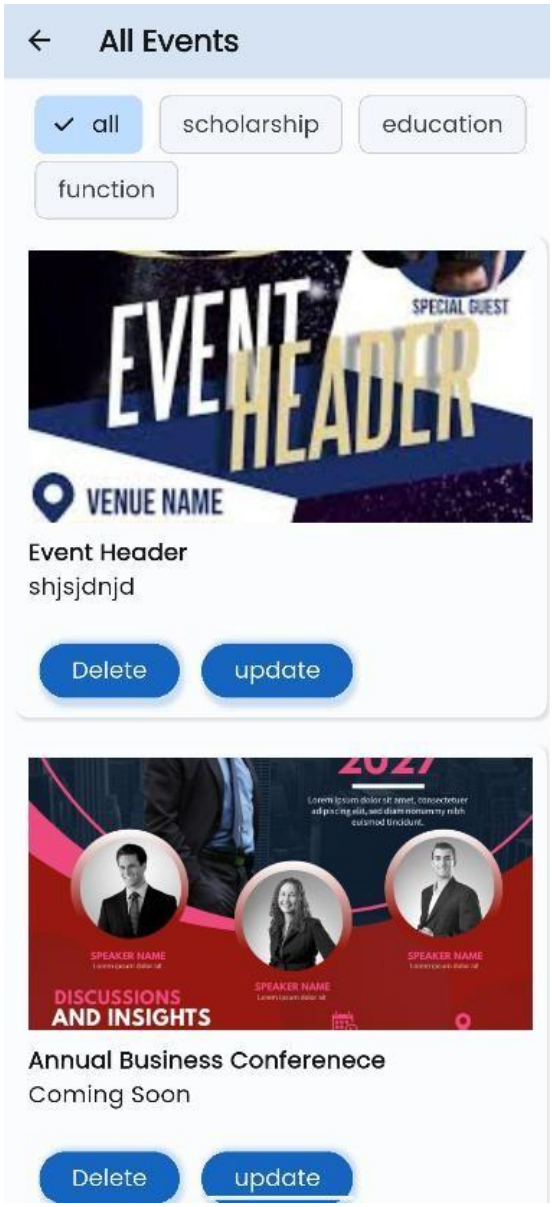

Figure 106 View Events

5.2 Timetable

For Admins: Managing Timetables

1. Tap **Timetable > Manage Timetables**.
2. Select **Add Timetable**.
3. Add Picture.
4. Tap **Submit** to sync with user calendars.
5. To edit/delete, go to **Timetable > Manage**, select a schedule, edit fields, or tap **Delete** and confirm.

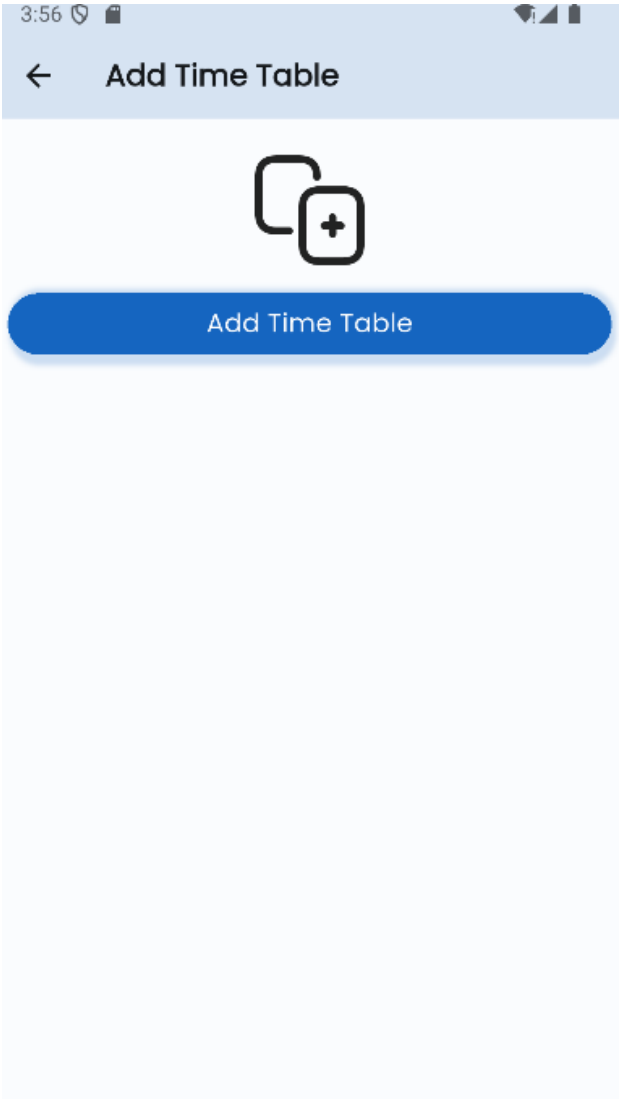

Figure 107 Add Timetable

For Students/Parents: Viewing Timetables

1. Tap **Timetable** to view your class schedule.
2. Tap a class to see details (course, time, location, instructor).

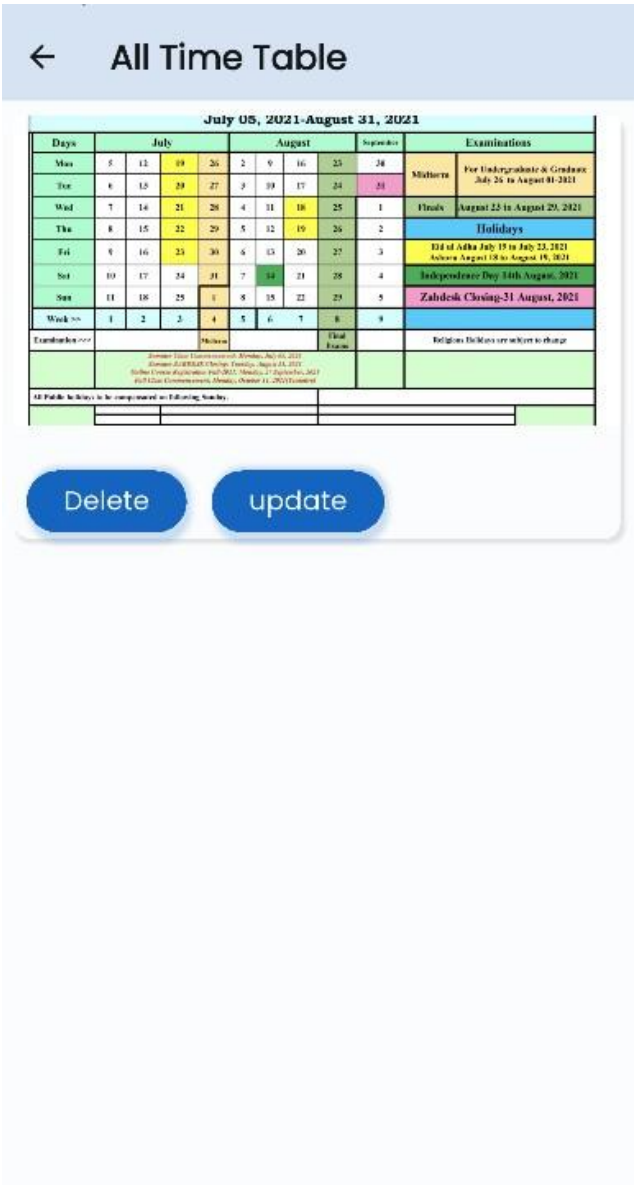

Figure 108 View Timetable

5.3 Messaging

Sending a Message

1. Tap **Messages > New Message**.
2. Select recipient(s): individual (search by name/email) or group (from existing groups).
3. Type your message (max 1000 characters).
4. Tap **Send**. Recipients receive notification.

Figure 109 Send Message

Viewing Messages

1. Go to **Messages > Inbox** to see new and past messages.
2. Tap a message to view its content and reply.

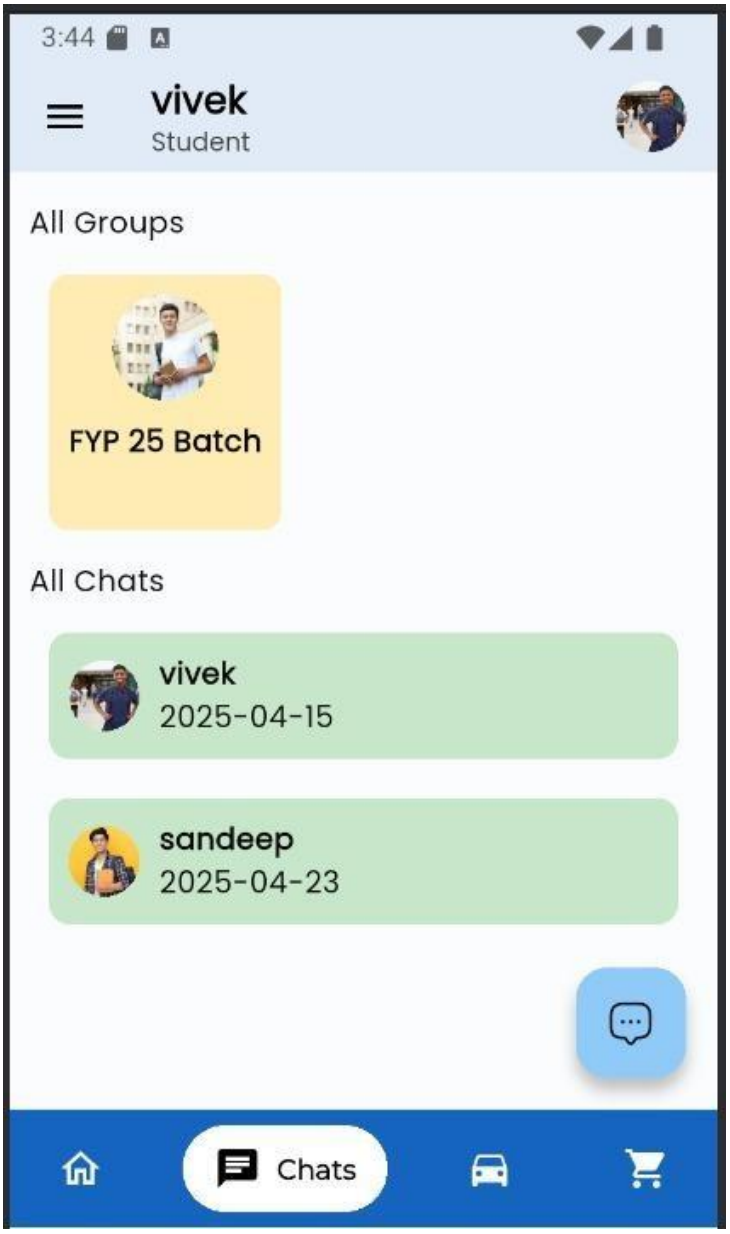

Figure 110 View Messages

For Teachers: Managing Requests

1. Go to **Messages > Requests**.
2. Review incoming message requests from students.
3. Tap **Approve** or **Reject** to control communication.

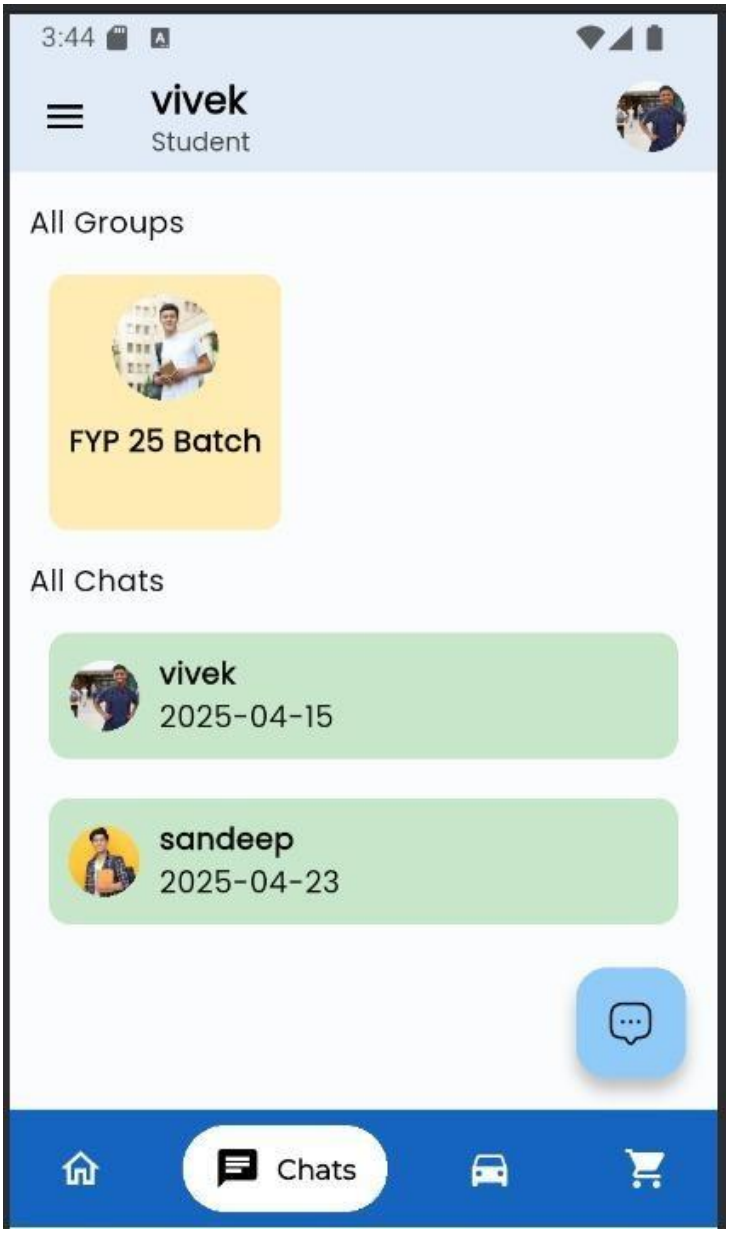

Figure 111 Manage Request

5.4 Job Board

For Students: Browsing Jobs

1. Tap **Job Board** in the navigation bar.
2. Filter jobs by **Category** (e.g., Internship, Part-Time), **Location**, or **Type**.
3. Tap a job to view details: title, description, requirements, application link.
4. Tap **Apply** to visit the external application URL or contact the poster.

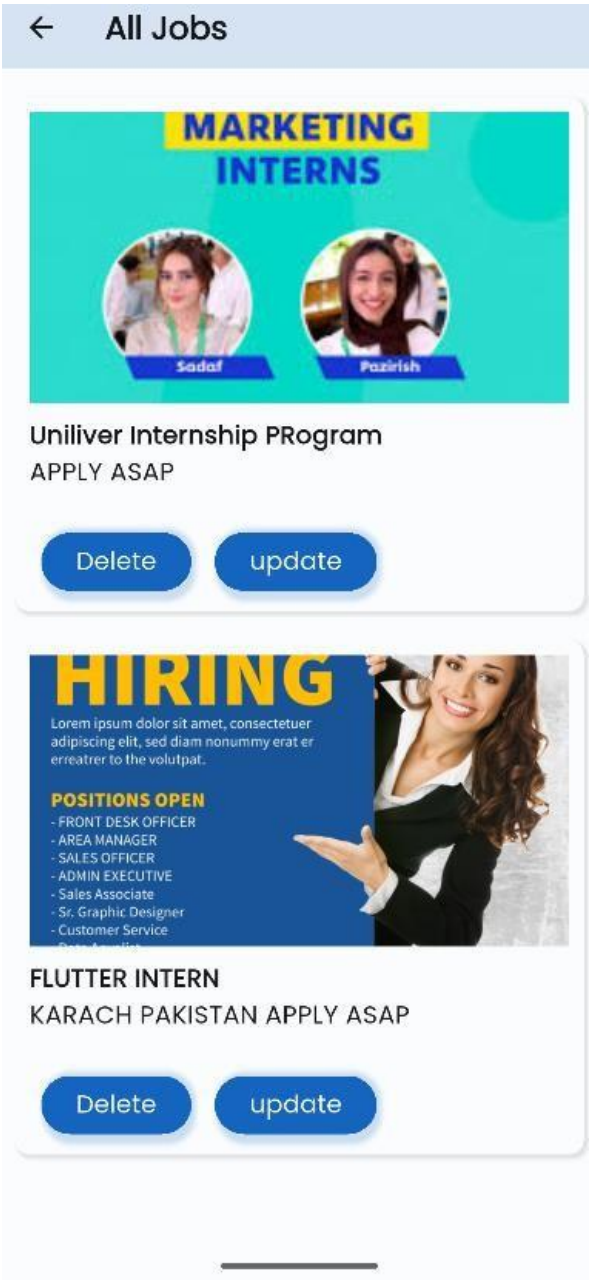

Figure 112 View Jobs

For Teachers/Admins: Posting Jobs

1. Tap **Job Board > Add Job**.
2. Enter details:
    - **Title**: e.g., “Software Engineering Intern”
    - **Description**: Job duties and requirements.
    - **Category**: Select Internship, Part-Time, or Full-Time.
    - **Application Link**: Provide a URL or contact email.
3. Tap **Submit** to publish.
4. To edit/delete, go to **Job Board > My Postings**, select a job, edit fields, or tap **Delete**.

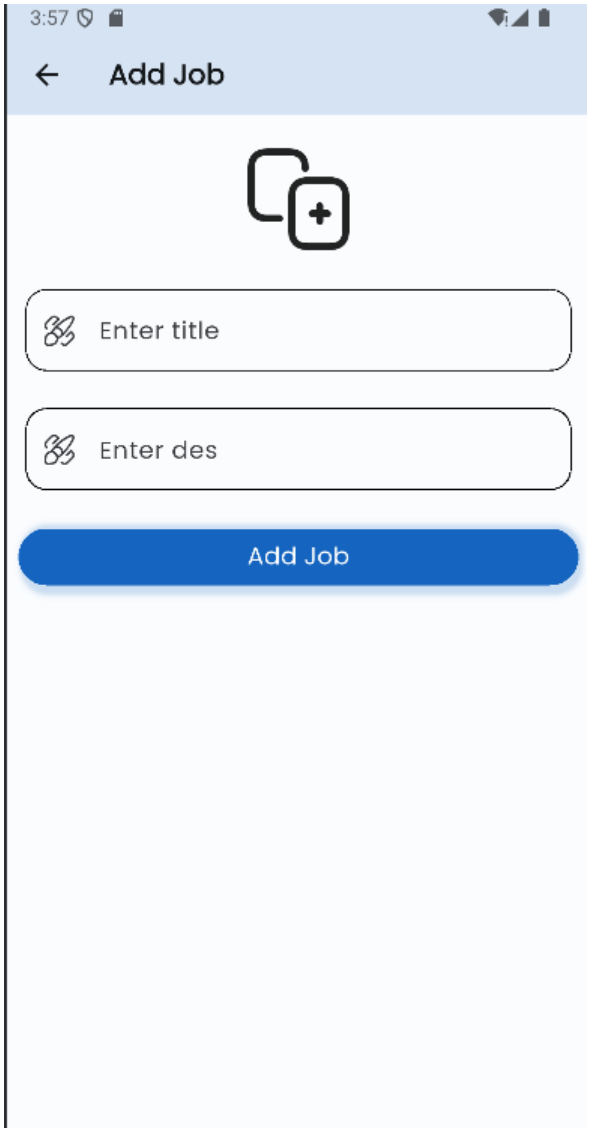

Figure 113 Add Job

5.5 Marketplace

For Students: Listing an Item

1. Tap **Marketplace > Add Item**.
2. Enter details:
    - **Title**: e.g., “Used CS Textbook”
    - **Description**: Item condition and details (max 500 characters).
    - **Price**: Enter amount (e.g., $20).
    - **Images**: Upload up to 3 images (JPG/PNG, max 5 MB each).
3. Tap **Submit** to list the item.

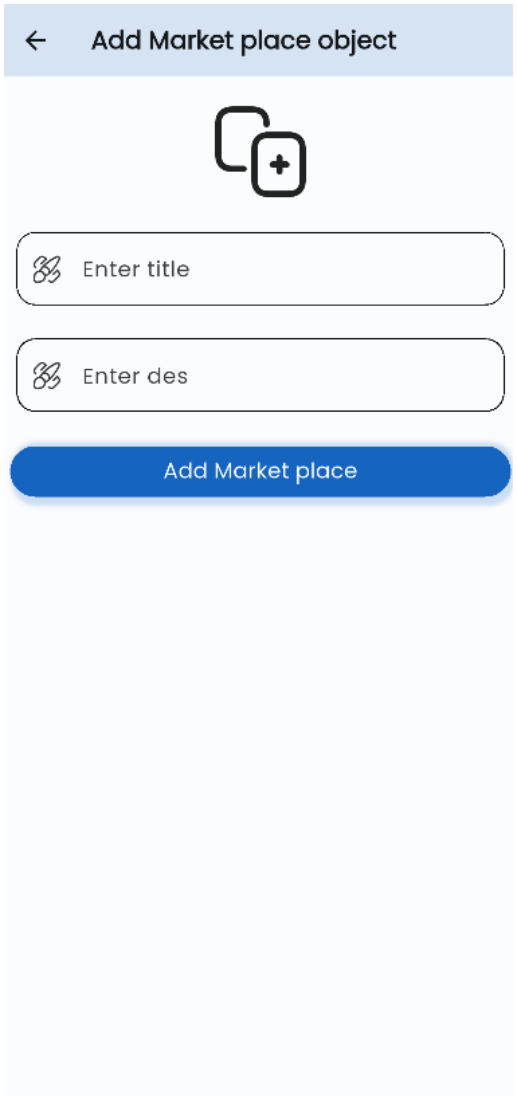

Figure 114 Add Item

Editing/Deleting an Item

1. Go to **Marketplace > My Listings**.
2. Select an item, tap **Edit**, update fields, and tap **Save**.
3. To delete, tap **Delete** and confirm.

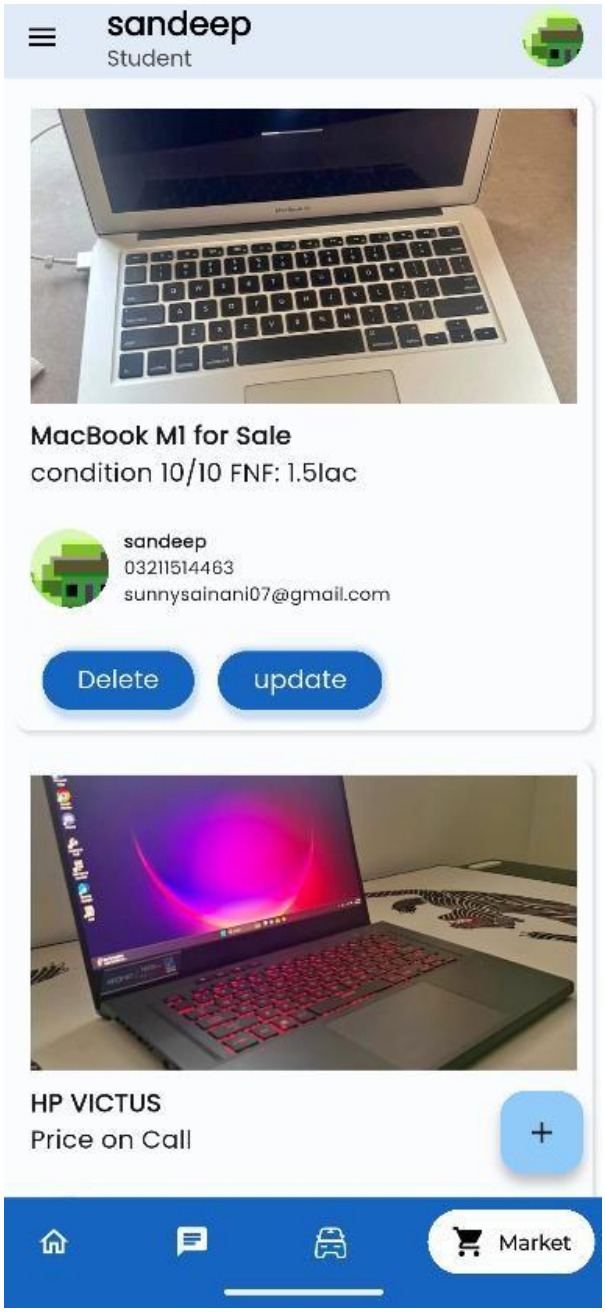

Figure 115 Delete Item

5.6 Groups

Joining a Group

1. Tap **Groups** to browse available groups (e.g., “CS Study Group”).
2. Tap a group to view its description and members.
3. Tap **Join Group** to become a member.

Figure 116 View Groups

Creating a Public Group (Students)

1. Tap **Groups** > **Create Group**.
2. Enter group name, description, and privacy settings (public/private).
3. Tap **Submit** to create. Invite members via **Messages**.

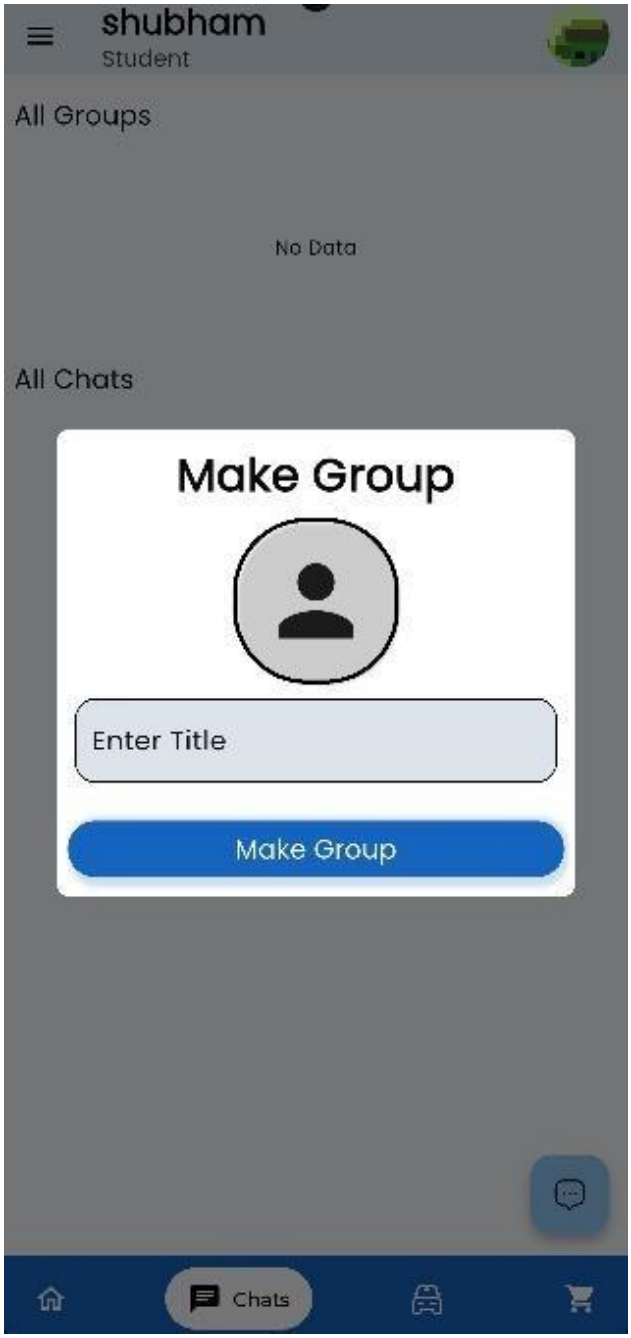

Figure 117 Create Public Group

5.7 Notifications

1. Tap **Notifications** to view alerts for events, messages, job postings, or carpool updates.
2. Tap a notification to go to the relevant feature (e.g., event details).
3. Customize alerts in **Settings > Notifications**:
    - Toggle options for Events, Messages, Job Board, etc.
    - Set notification sound and vibration preferences.
4. Clear notifications by tapping **Clear All** or swiping individual alerts.

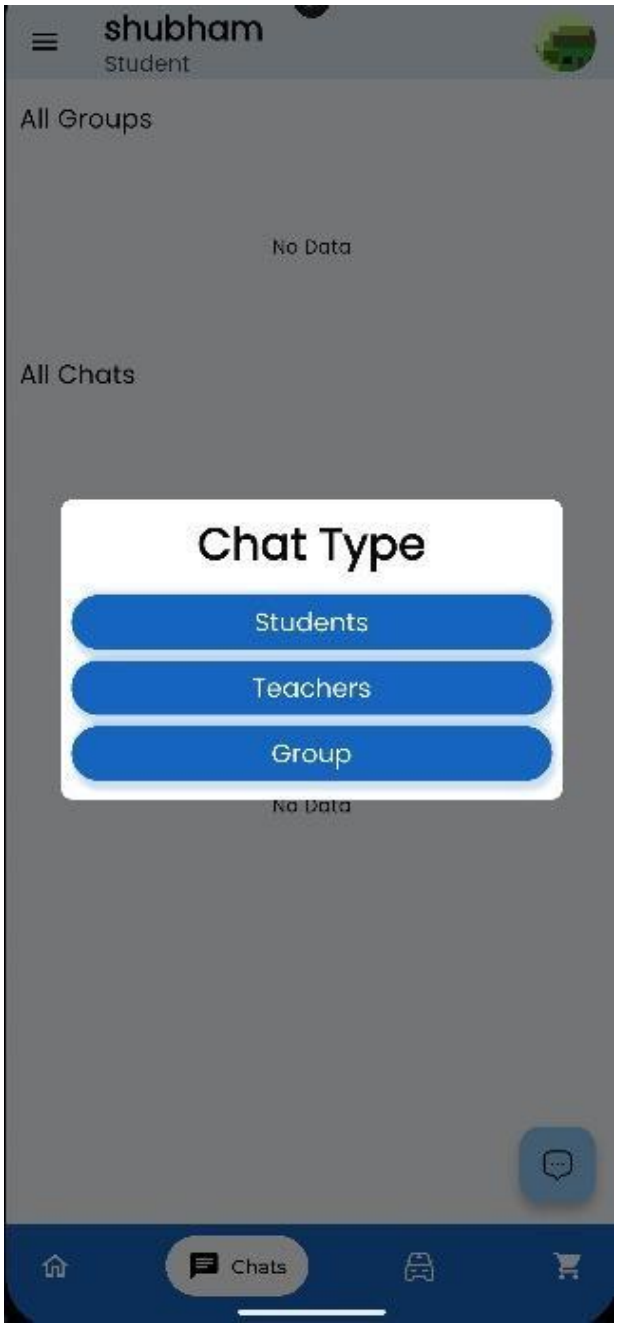

Figure 118 View Notifications

5.8 Lost and Found

Reporting a Lost Item

1. Tap **Lost and Found > Add Lost Item**.
2. Enter details:
    - **Description**: e.g., “Black backpack with laptop”
    - **Date Lost**: Select from calendar.
    - **Location**: e.g., “Library, 2nd floor”
    - **Contact Info**: Your email or phone (optional).
3. Tap **Submit** to list the item.

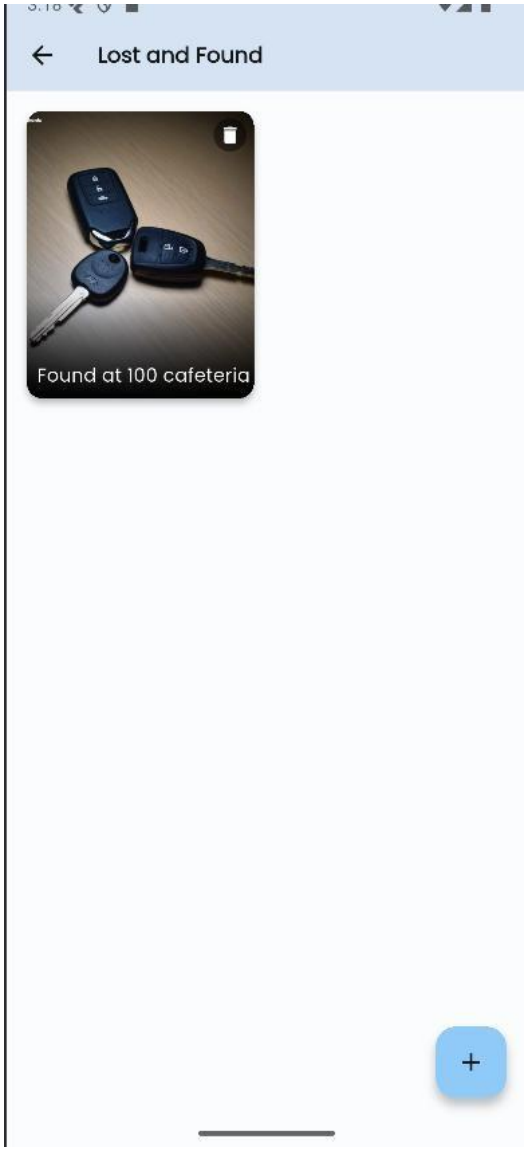

Figure 119 Add Item

5.9 Resources

Adding a Resource

1. Tap **Resources > Add Resource** (available to students/admins).
2. Enter details:
    - **Name**: e.g., “Lecture Room 101”
    - **Type**: Select Room, Book, or Equipment.
    - **Description**: e.g., “Projector available” (max 500 characters).
    - **Quantity**: e.g., “1 room”.
3. Tap **Submit** to add to the system.

Figure 120 Add Resource Item

5.10 To-Do List

Creating a Task

1. Tap **To-Do List > Add Task**.
2. Enter details:
    - **Title**: e.g., “Finish CS Assignment”
    - **Description**: Task details (max 500 characters).
    - **Due Date**: Select from calendar.
    - **Priority**: Low, Medium, High.
3. Tap **Submit** to add to your list.

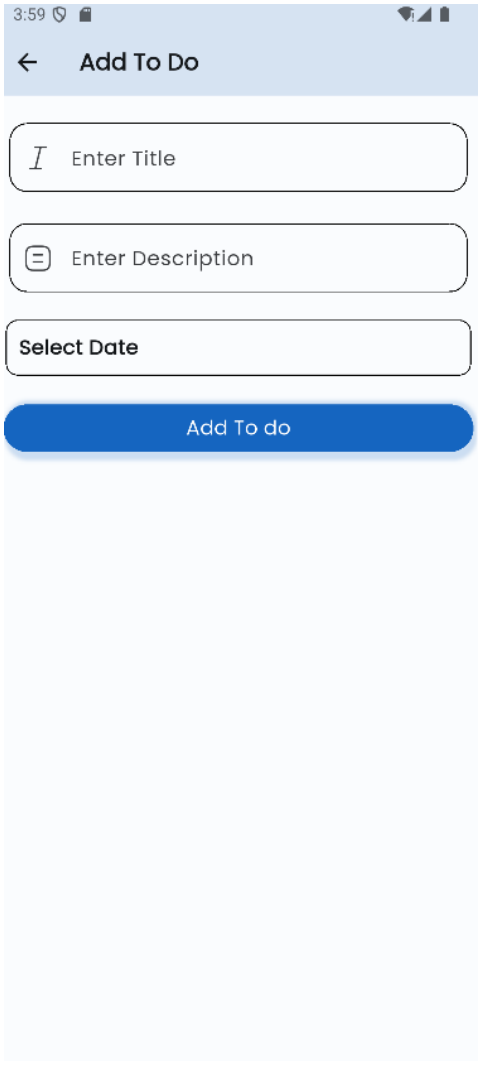

Figure 121 Add To-Do Item

Editing/Deleting a Task

1. Go to **To-Do List**.
2. Select a task, tap **Edit**, update fields, and tap **Save**.
3. To delete, tap **Delete** and confirm.

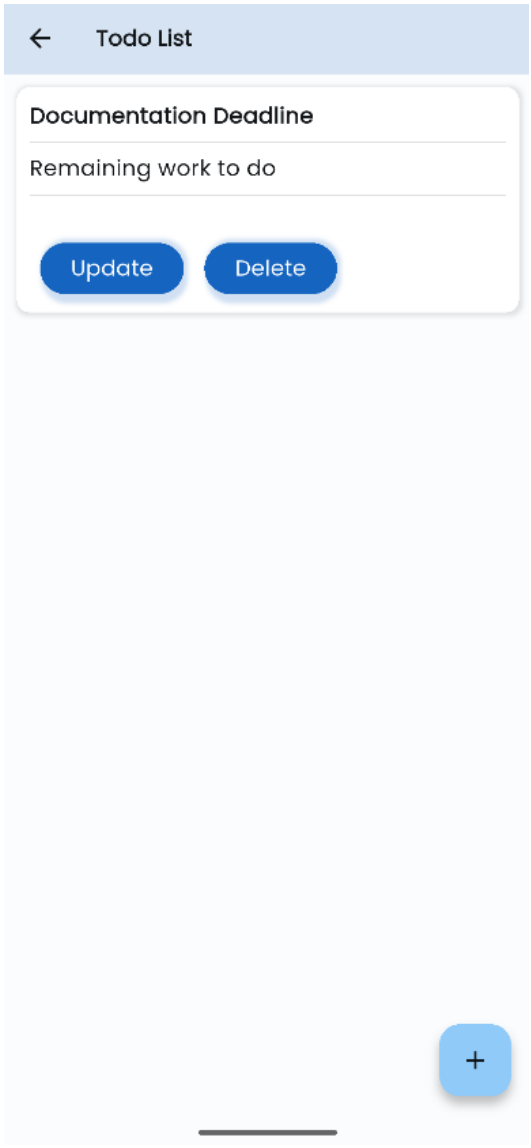

Figure 122 Delete & Edit To-Do Item

5.11 Carpool

Offering a Ride

1. Tap **Carpool > Add Ride**.
2. Enter details:
    - **Pickup Point**: e.g., “Main Campus Gate”
    - **Drop-Off**: e.g., “Downtown Library”
    - **Date and Time**: Select from calendar/time picker.
    - **Seats Available**: e.g., “3 seats”
    - **Vehicle Details**: e.g., “Toyota Corolla, Blue”
3. Tap **Submit** to list the ride.

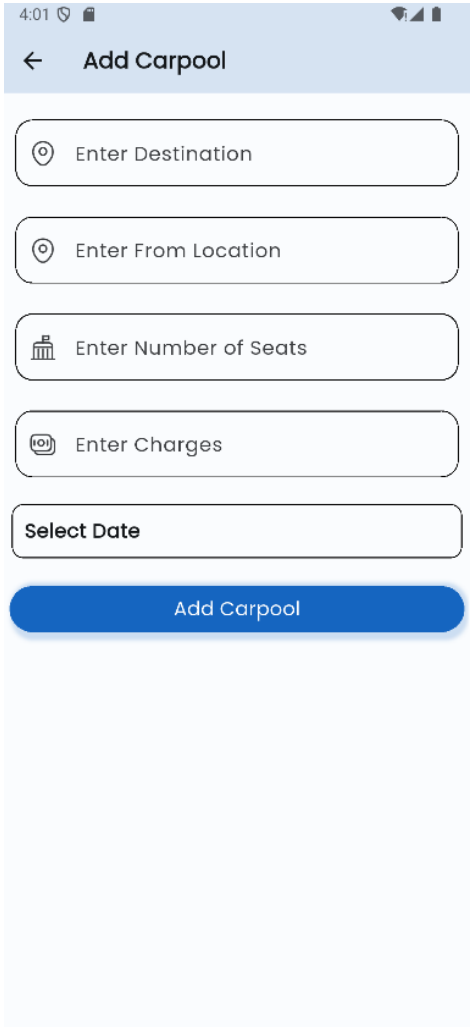

Figure 123 Add Ride

Sharing Location

1. In an active ride, tap **Send Location**.
2. Allow GPS access and tap **Share** to send your real-time location to participants.

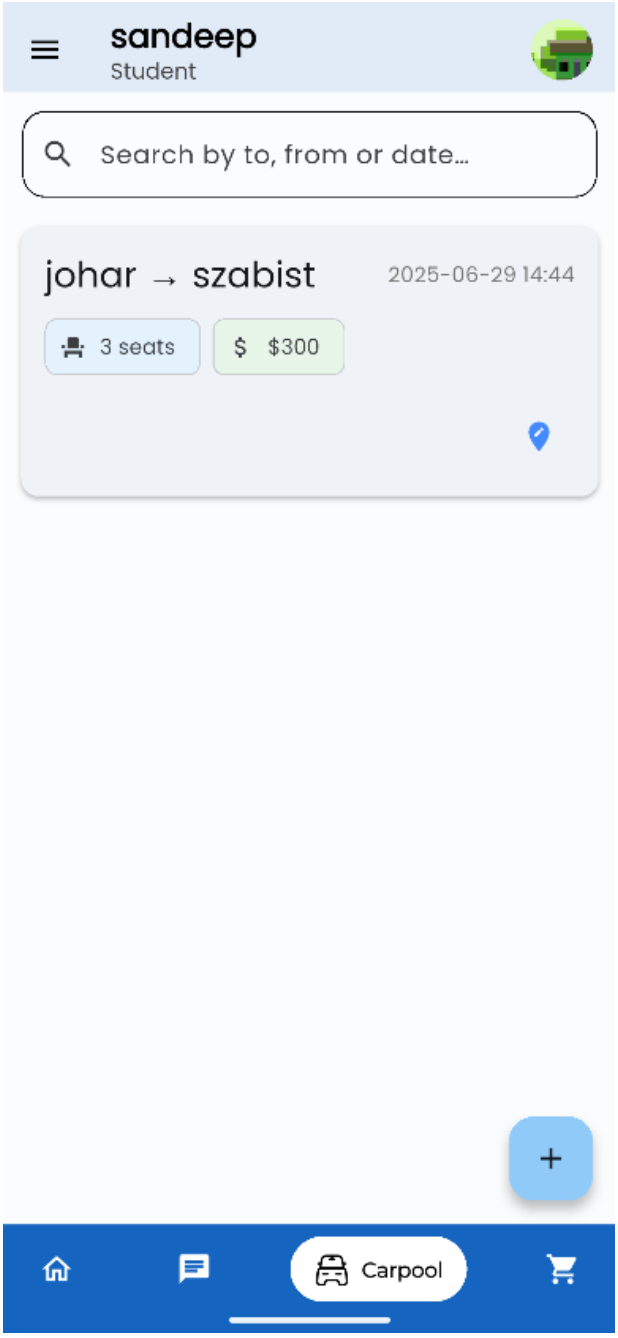

Figure 124 Share Ride Location

Editing/Deleting a Ride

1. Go to **Carpool > My Rides**.
2. Select a ride, tap **Edit**, update fields, or tap **Delete** and confirm.

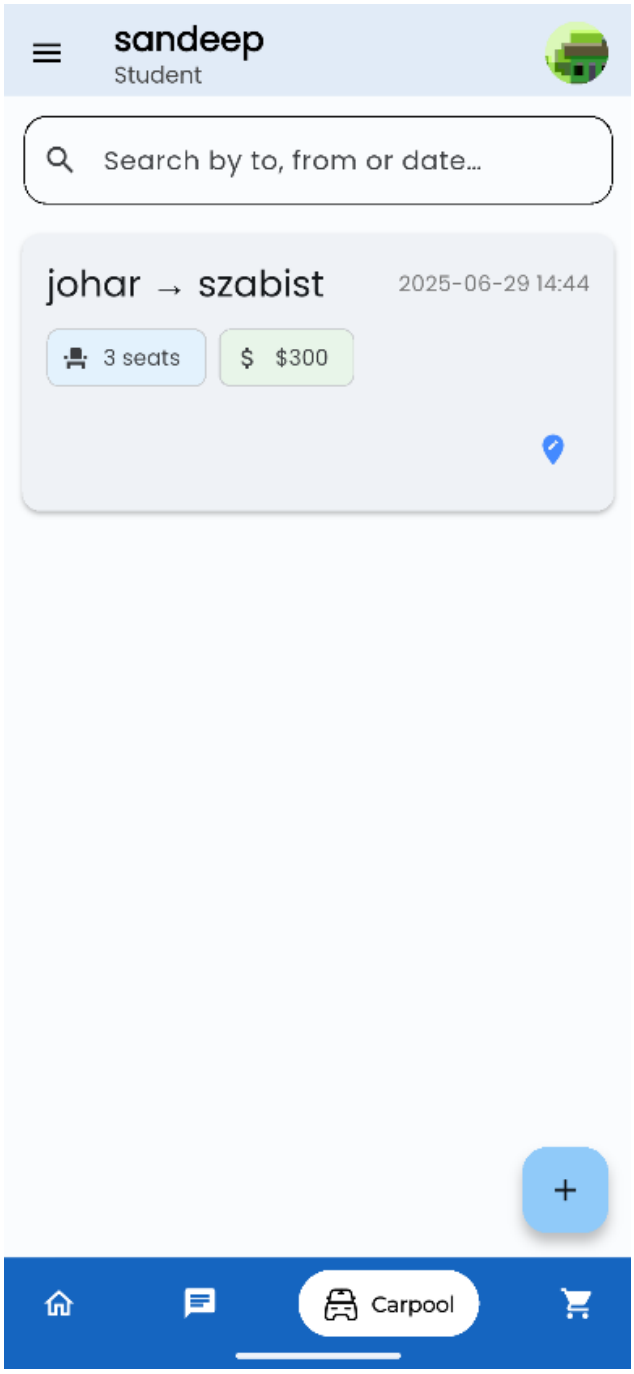

Figure 125 Edit & Remove Rides

6. Troubleshooting

- **Login Issues**: Verify email/password. Use **Forgot Password** if needed.
- **Notifications Not Received**: Check **Settings > Notifications** and ensure internet connectivity.
- **Feature Not Loading**: Restart the app or update via Google Play Store.
- **Sync Issues**: Ensure a stable internet connection and re-sync in **Settings > Sync Data**.
- **Error Messages**: Note the error code and contact support.

7. Tips for Best Use

- Regularly check **Notifications** for updates on events and messages.
- Sync your **Timetable** with your device calendar for easy planning.
- Use **Messages** to coordinate with group members or carpool participants.

Glossary

Sr.no	Terms	Description
1	API	Application Programming Interface – A set of protocols and tools for building software applications and facilitating communication between different systems.
2	MongoDB	A non-relational database management system used for storing and managing application data.
3	RESTful API	Representational State Transfer – A web service architecture that allows interaction with REST-based web services.
4	HTTPS	Hypertext Transfer Protocol Secure – A secure version of HTTP used for encrypted communication over a computer network.
5	UI	User Interface – Refers to the design and layout through which users interact with the app.
6	UX	User Experience – Refers to the overall experience of a user when interacting with the app.

References

1. Google. “Flutter Documentation.” 2023. Google. Accessed on September 1, 2023. https://flutter.dev/docs
2. Udemy.com (https://www.udemy.com/course/learn-flutter-dart-to-build-ios-android-apps/?couponCode=ST11MT91624A)
3. https://www.mongodb.com/resources/products/compatibilities/flutter-and-mongodb
4. Stackoverflow.com
5. Flutter Documentation: https://flutter.dev/docs
6. MongoDB Resources: https://www.mongodb.com
7. Udemy Course on Flutter & Dart: https://www.udemy.com/course/learn-flutter-dart
8. IEEE Standard 1016: Software Design Descriptions
9. Node.js Documentation: https://nodejs.org
10. MongoDB Documentation: https://mongodb.com
11. D. M. D. Oliveira, L. Pedro, and C. Santos, “The use of mobile applications in higher education classes: A comparative pilot study of the students’ perceptions and real usage,” *Smart Learn. Environ.*, vol. 8, no. 1, pp. 1–17, 2021, doi: 10.1186/s40561-021-00159-6.
12. B. G. Jayatilleke, G. R. Ranawaka, C. Wijesekera, and M. C. B. Kumarasinha, “Development of mobile application through design-based research,” *Asian Assoc. Open Univ. J.*, vol. 13, no. 2, pp. 145–168, 2018, doi: 10.1108/AAOUJ-03-2018-0013.
13. A. Hinze *et al.*, “A study of mobile app use for teaching and research in higher education,” *Technol. Knowl. Learn.*, vol. 27, no. 3, pp. 1007–1030, 2022, doi: 10.1007/s10758-022-09599-0.
14. G. Rangel-de Lazaro and J. M. Duart, “Moving learning: A systematic review of mobile learning applications for online higher education,” *J. New Approaches Educ. Res.*, vol. 12, no. 2, pp. 198–224, 2023, doi: 10.7821/naer.2023.7.1476.
15. zJ. S. Mtebe and M. M. Kissaka, “Mobile applications in university education: The case of Kenya,” in *Proc. IST-Africa Conf.*, Nairobi, Kenya, 2018, pp. 1–9, doi: 10.23919/ISTAFRICA.2018.8417320.

Plagiarism Report