Bhumi Vyas

Student ID: 202412123

Faculty : shruti Bhilare

Discuz

Where Learning Meets Simplicity

Document Outline

Educational Platform for Course & Assignment Management

In the modern academic environment, the need for streamlined digital platforms that support effective course, assignment, and student management is greater than ever. This project is a comprehensive Educational Web Application designed to meet the core administrative and academic needs of institutions, teachers, and students by providing a centralized system to manage courses, assignments, communications, and reminders efficiently.

The platform introduces a role-based access system with three types of users: Students, Admins, and Super Admins. Each role is assigned specific privileges to ensure secure and structured operations. Super Admins manage institutional authority by assigning or revoking admin privileges. Admins are responsible for course creation, assignment uploads, class management, and feedback. Students can join courses using a code, submit assignments, and communicate with instructors.

1.1 Purpose

The purpose of this project is to build an educational web platform that helps admins manage courses and assignments, allows students to submit their work and set reminders, and gives super admins control over user roles — all in one centralized system.

1.2 Key features

- User Authentication (Login & Registration)
- Course Management with secure code-based enrollment
- Assignment Upload and Submission with public and private commenting features
- Personalized Dashboard displaying course, reminder, and activity information
- Reminder Calendar for setting academic alerts
- Student Directory with course-specific visibility
- Role-Based Controls ensuring each user accesses features as per their authority

The frontend is developed using React, providing a fast and responsive user experience, while the backend is powered by Node.js and MongoDB, ensuring scalable and flexible data handling. The system supports dynamic CRUD operations, ensuring seamless management of assignments, courses, and user activities.

This platform is particularly beneficial for educators seeking a structured workflow, and for students needing clarity and organization in their academic progress. It also promotes transparent communication, timely submissions, and efficient supervision.

2.1 Functional Requirements

These define the core features and operations the system must perform.

User Management

- The system must allow users to register and login securely.
- The system must support three user roles: Student, Admin, and Super Admin.
- Super Admin must be able to promote users to Admin role or revoke it.

Course Management

- Admin must be able to create and delete courses.
- Admin must generate and share a course access code.
- Students must be able to join courses using a valid code.

Assignment Management

- Admin must be able to upload assignments for a course.
- Students must be able to view and submit assignments for enrolled courses.
- Students must be able to unsubmit assignments before the deadline.
- Students can post public or private comments along with their submissions.

- Admin must be able to view submissions and comment privately.

Calendar & Reminder System

- Students and Admins must be able to add personal or course-related calendar reminders.

- The system must display upcoming events/reminders on the dashboard.

Student Directory

- Students must be able to view a directory of other students in the same course.

Dashboard

- Each user must have a personalized dashboard showing assignments, reminders

- Admin dashboard must show stats like submissions and student activity.

2.2 Non-Functional Requirements

These define the quality attributes, performance, and behavior of the system.

Performance & Scalability

- The system should support multiple concurrent users without significant performance drop.

Security

- All user data must be protected using JWT-based authentication.
- Role-based access must be strictly enforced (e.g., students can't access admin pages).

Usability

- The user interface should be clean, intuitive, and responsive across devices.
- System should provide feedback for actions (e.g., submission success, errors).

Maintainability & Extensibility

- The codebase should follow modular structure for easy updates and bug fixes.

Methodology and Development Process

3.1 Development Methodology – Agile Approach

The project follows a lightweight **Agile (Iterative Development)** methodology, which emphasizes incremental delivery, continuous feedback, and modular coding. Given the nature of the platform—where features like assignment submission, role management, reminders, and notifications evolve based on real user requirements—Agile proved to be the most appropriate model.

The core principles applied include:

- **Incremental Development**: The system was built feature-by-feature, starting with authentication, followed by course management, assignment handling, calendar integration, and notification systems.

- **Iterative Enhancements**: Feedback and testing led to iterative improvements such as adding an "unsubmit" feature, implementing private/public commenting, and dashboard statistics.

- **Modular Structure**: The backend (controllers, routes, models) and frontend (React components) were developed in an isolated, modular fashion to simplify debugging and extension.

- **Role-based Workflow**: All functionalities are governed by roles (Student, Admin, Super Admin), ensuring secure, structured access and user flows.

3.2 Process Modeling Steps

The overall development process was modeled into the following stages:

1. **Requirement Gathering**
   Identification of platform goals, target users, and key features such as assignment submission, reminder calendar, and role-based access.

2. **System Design**
   Creation of backend models (`User`, `Course`, `Assignment`, `Submission`, `Comment`, `Reminder`) and UI wireframes for login, dashboard, class details, and calendar views.

3. **Module-wise Development**
   Each core feature was implemented independently:

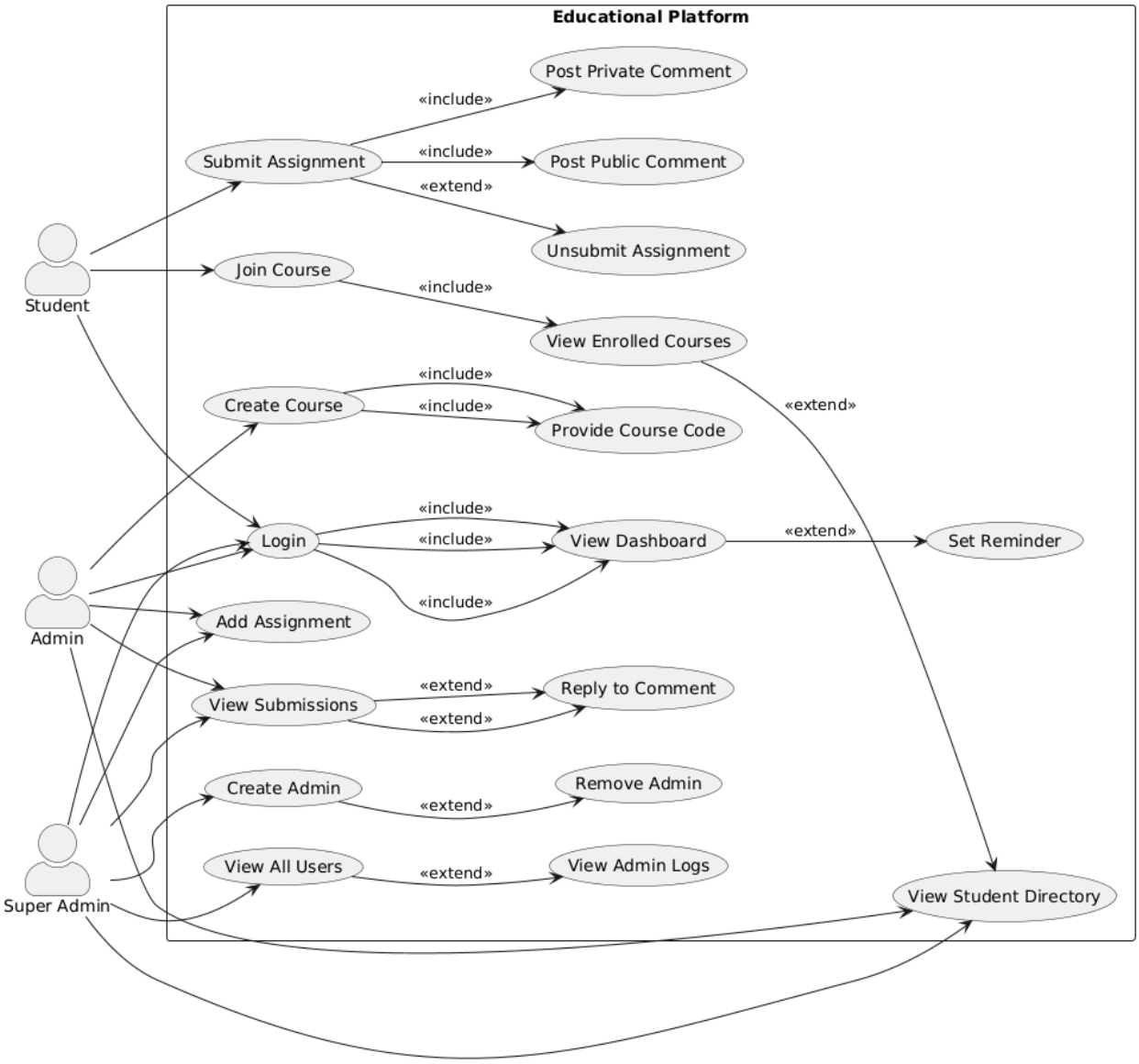
   - Authentication (Login/Registration)
   - Course Management
   - Assignment Upload & Submission
   - Commenting System
   - Reminder Calendar
   - Notifications and Student Directory

4. **Testing & Validation**
   Manual validation and debugging were conducted on each module using realistic test cases and workflows.

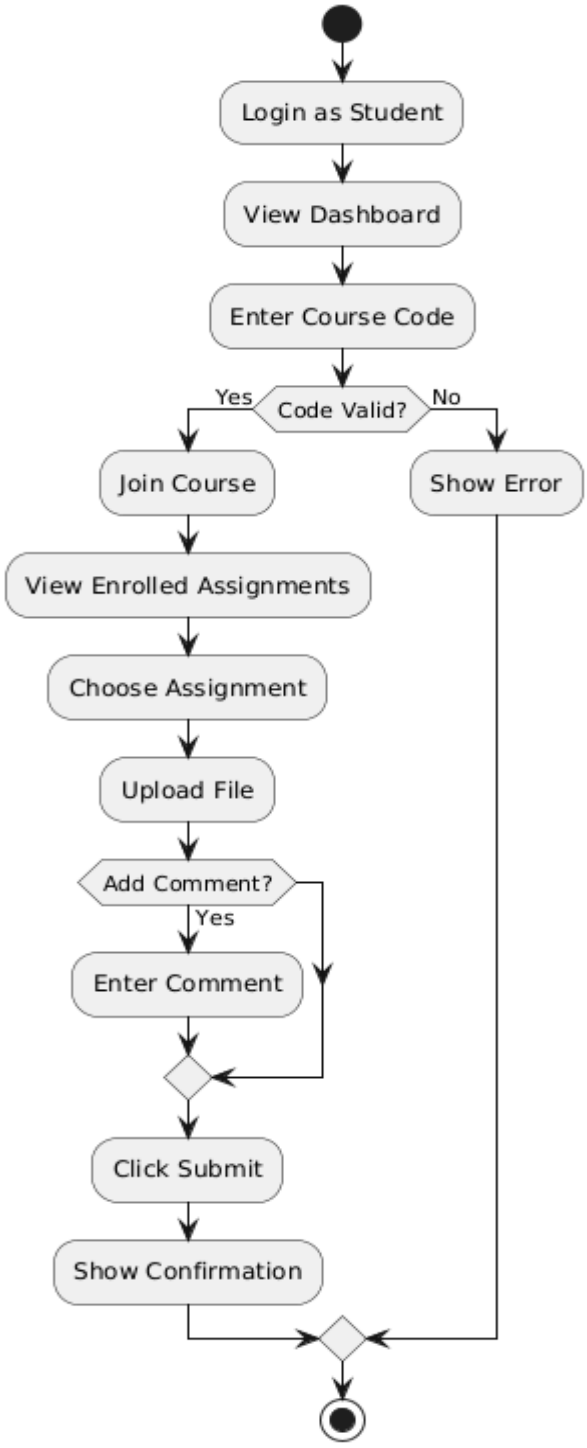
5. **Refactoring & Enhancements**
   Additional features such as assignment unsubmission, reply notifications, and download options for student lists were introduced post-initial testing.

6. **Documentation and UML Diagrams**
   Finalized use case flows, functional models, and UML diagrams were created for formal documentation.

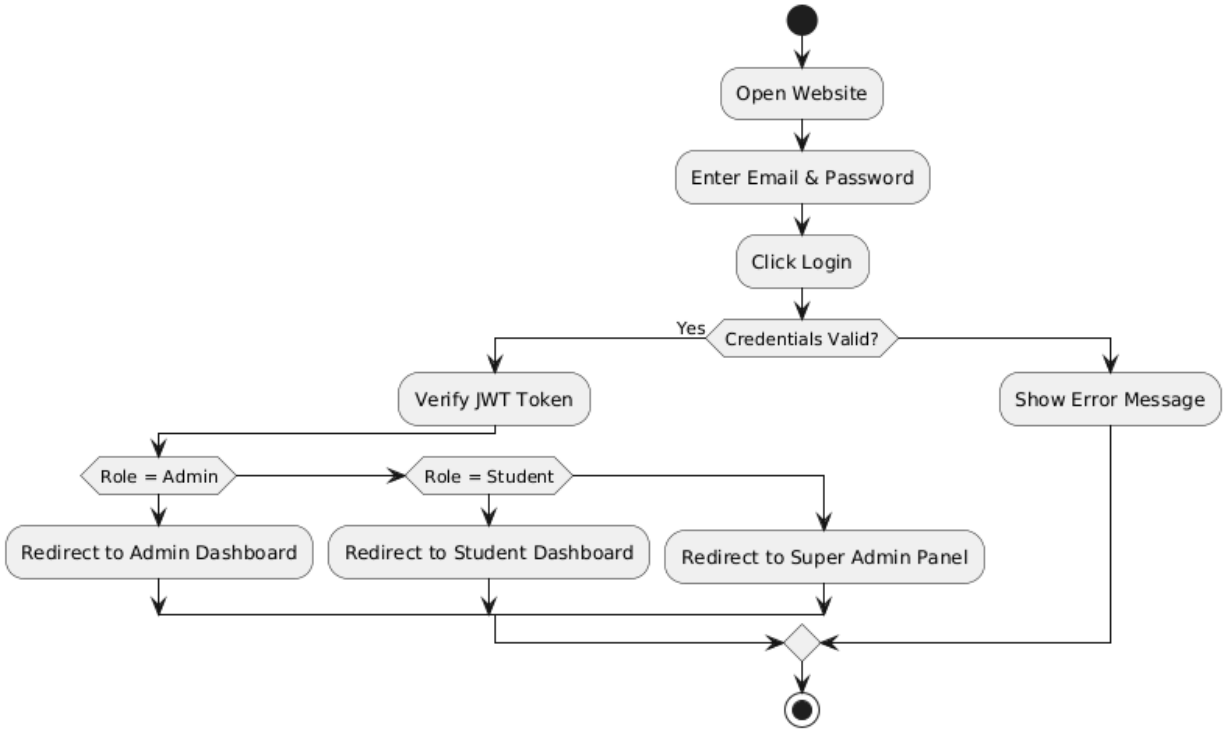
4.1 Use Case Diagram



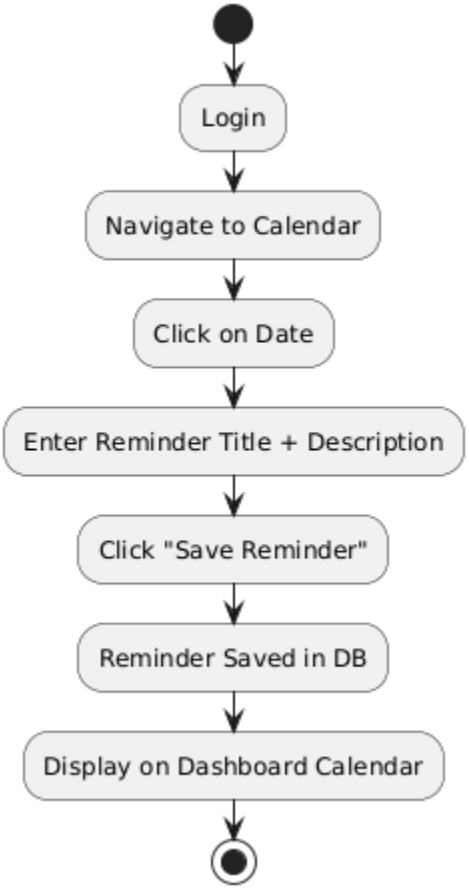
4.2 Activity Diagram(course enrollment and assignment submission)



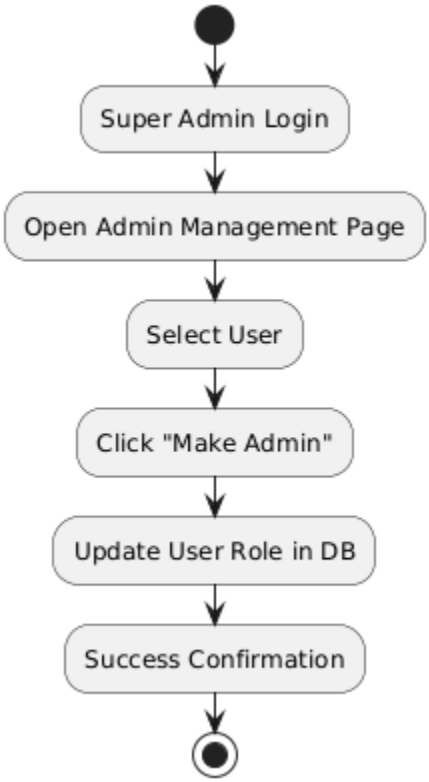
Login & Role-Based Routing



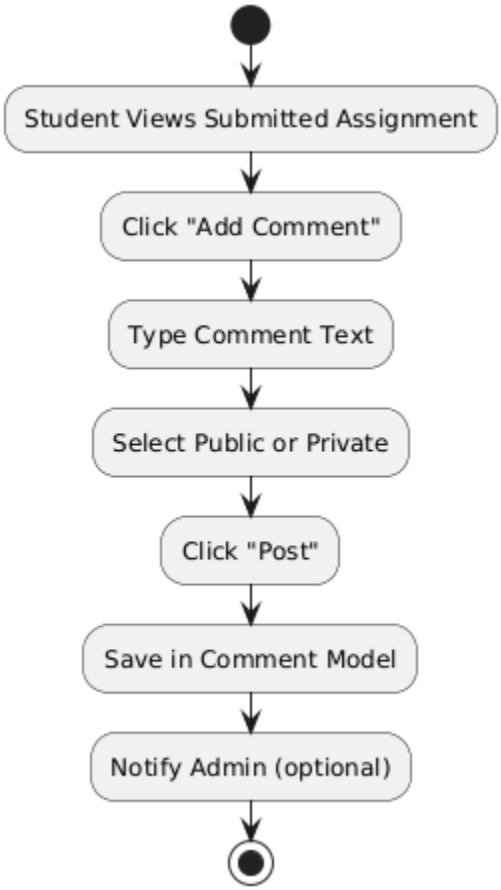
Set Reminder



Make User an Admin

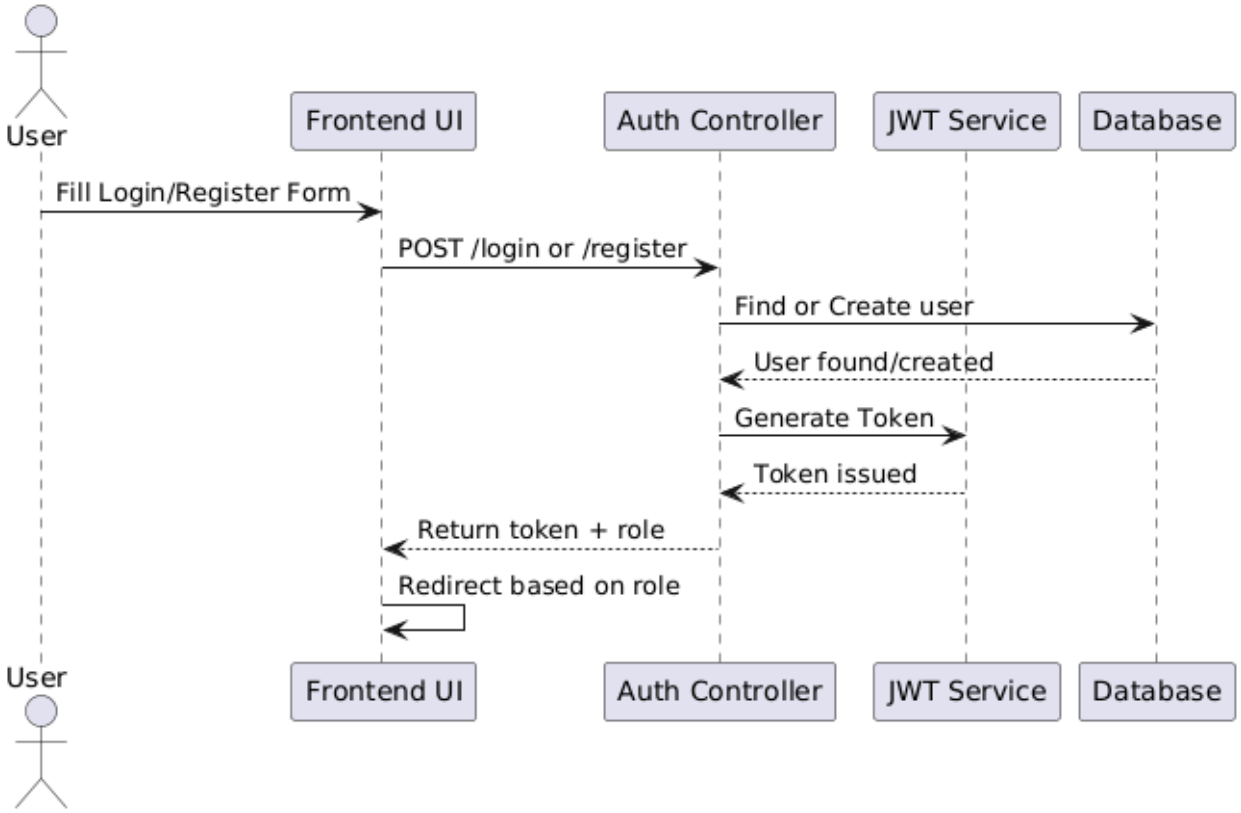


Post Public or Private Comment

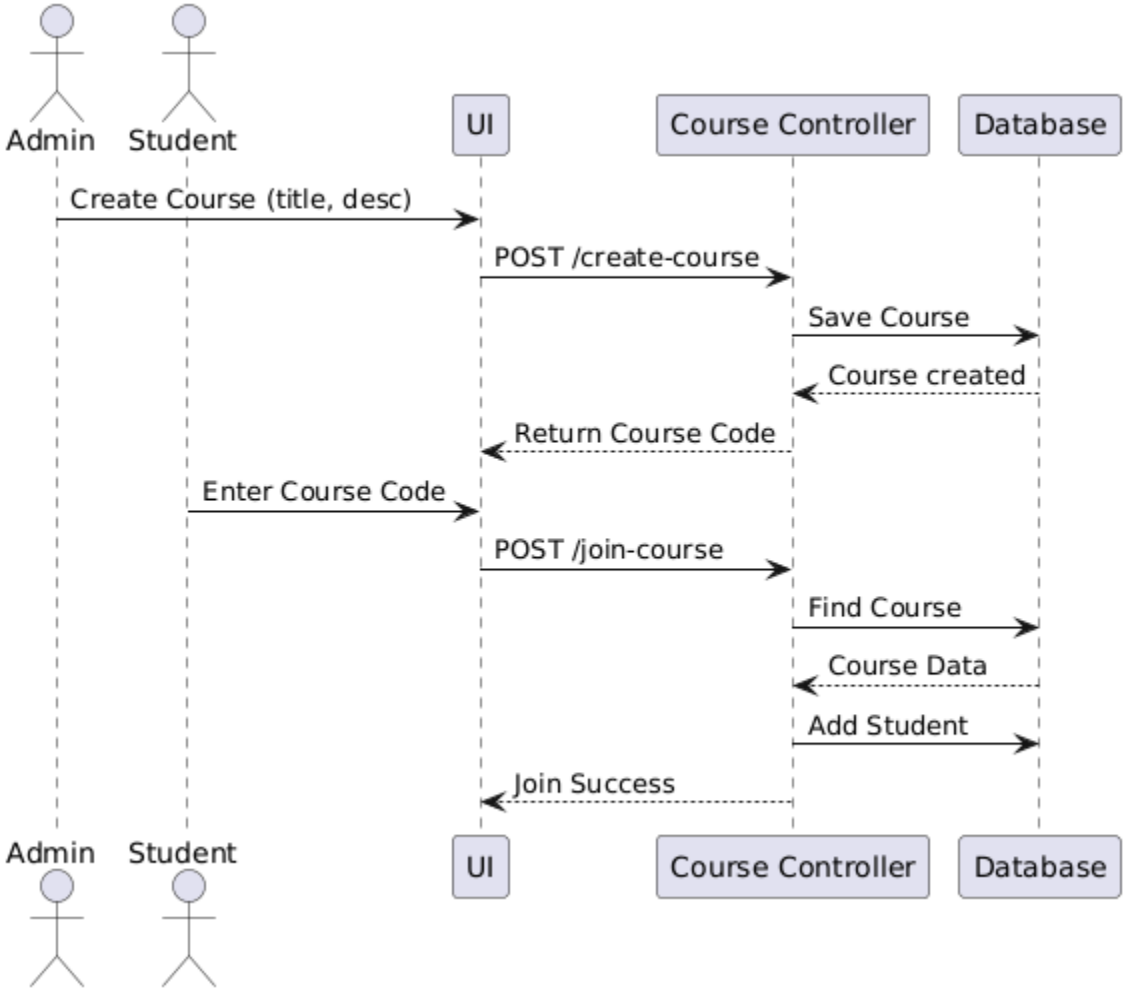


4.3 Sequence Diagram

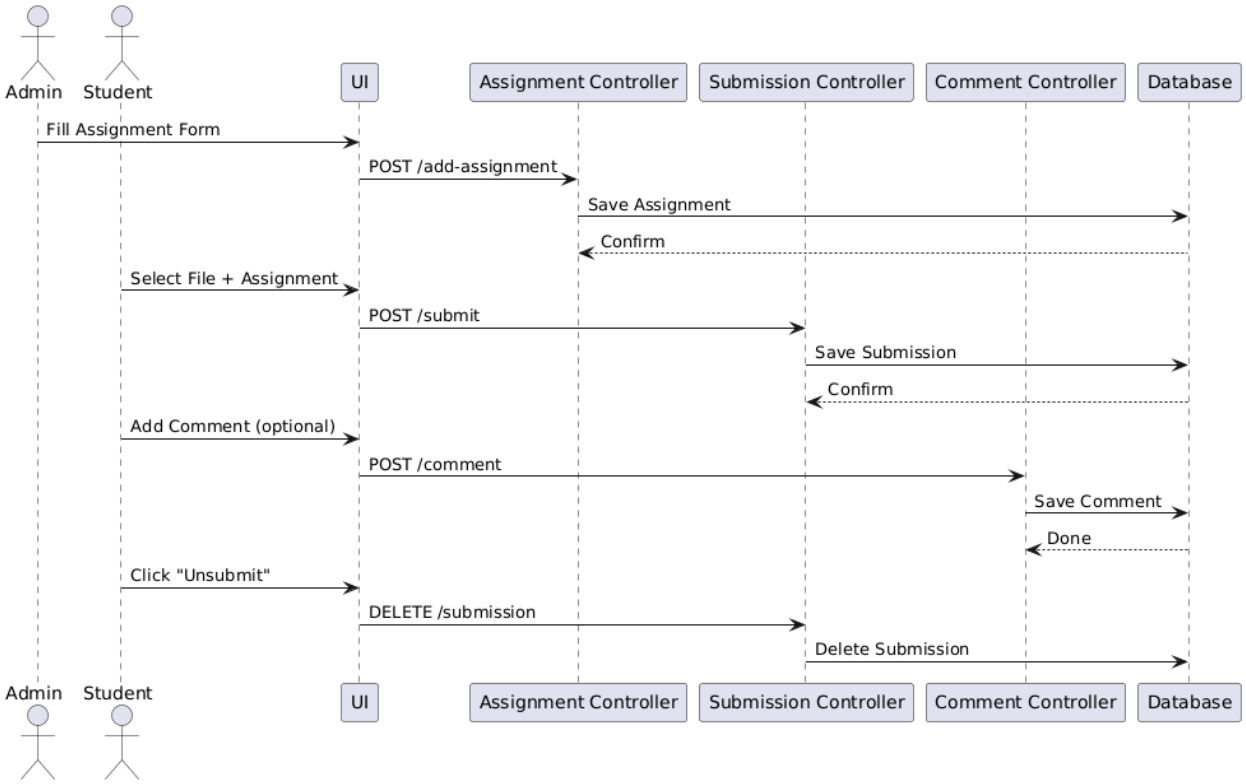
Authentication & Role Routing



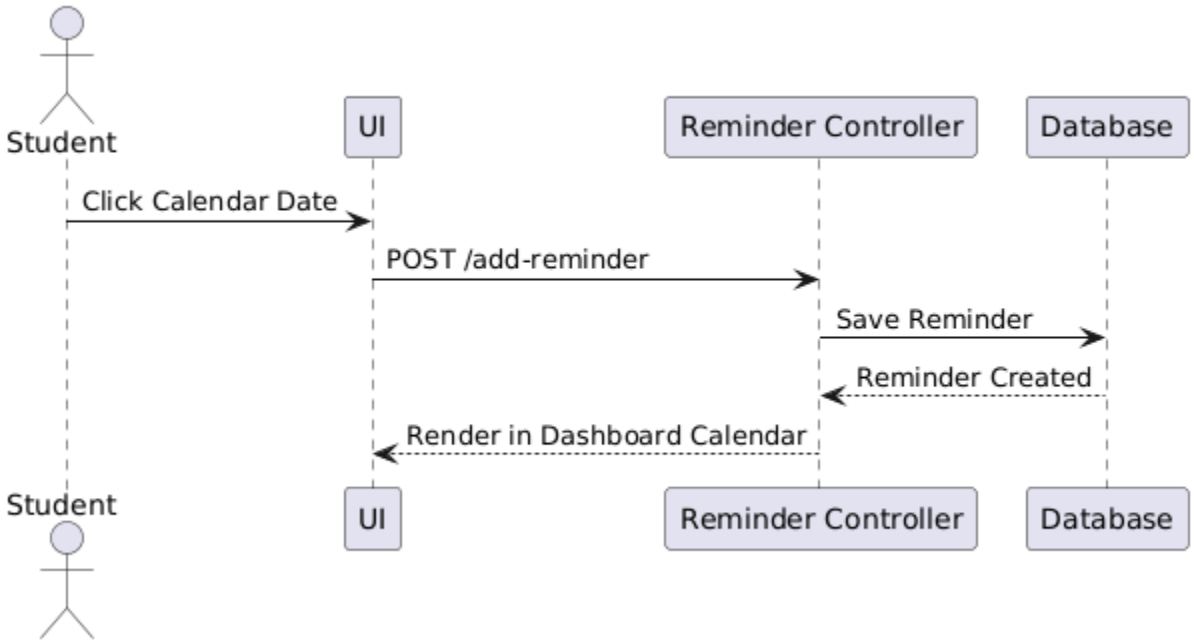
Course Lifecycle



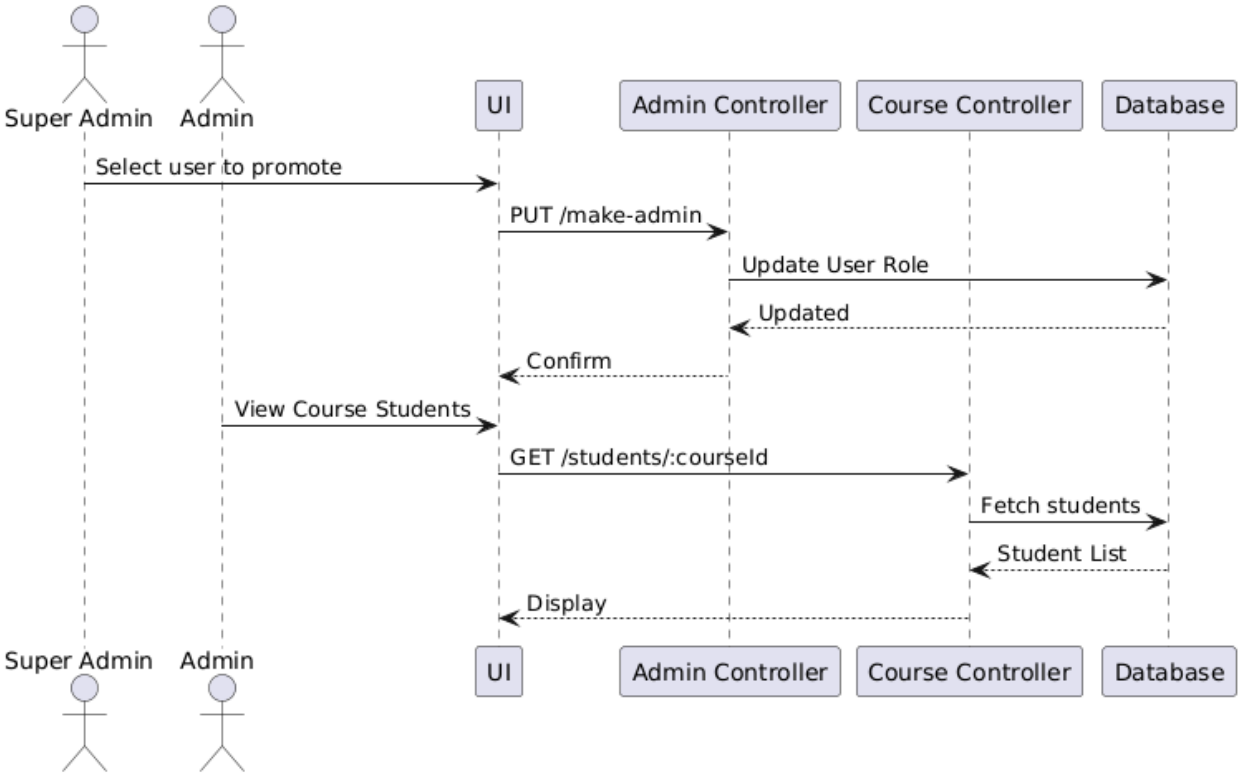
Assignment Management (Upload + Submit + Comments)



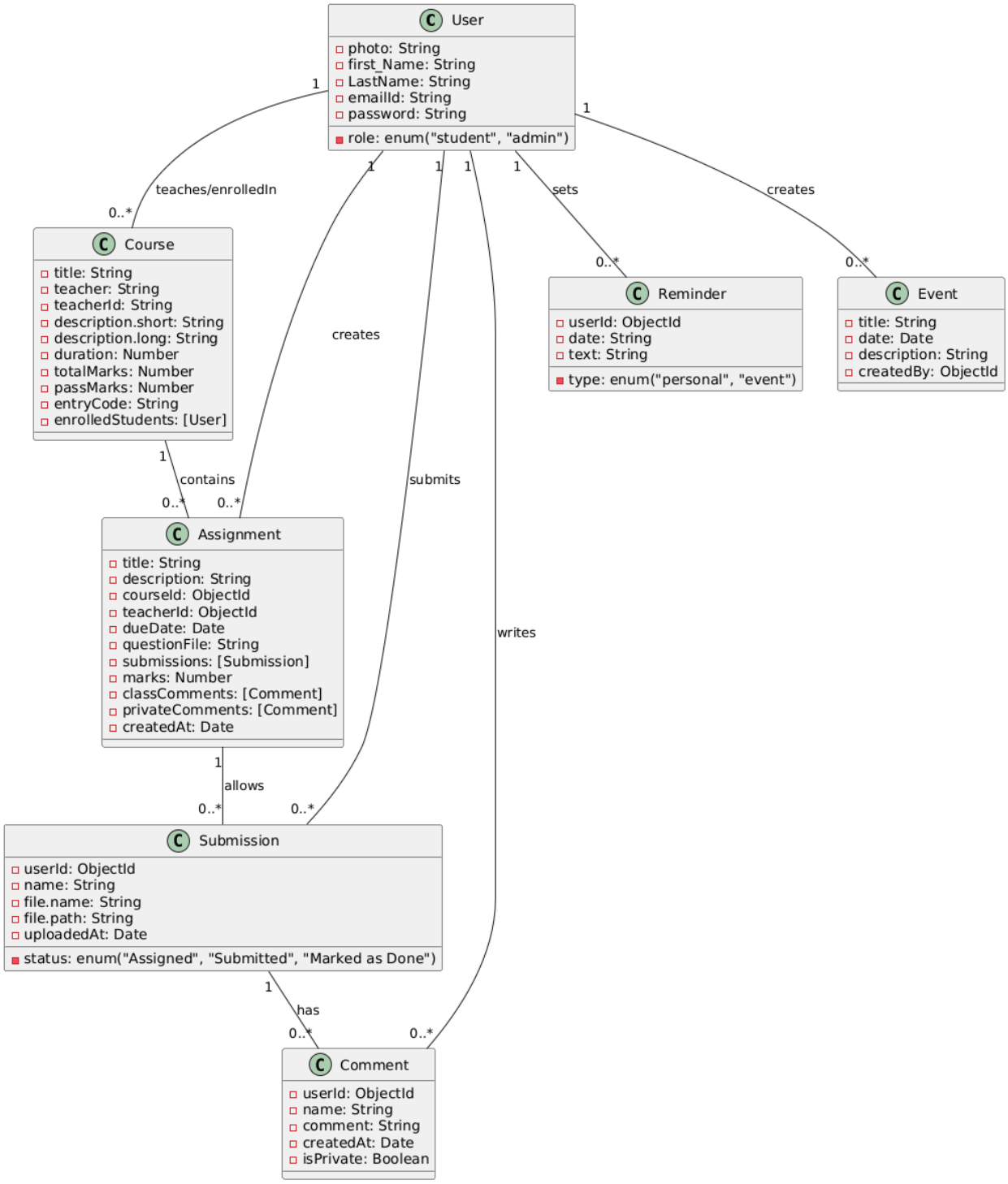
Reminder & Dashboard Calendar



Admin Controls & Directory Access



4.4 Class Diagram



Coding (APIs & Frameworks)

This project was built using a full-stack JavaScript-based technology stack, combining powerful backend APIs with a responsive frontend.

5.1 Frontend

- Framework: React.js
- Tools: Axios (for API calls), React Router, Tailwind CSS
- Features:
    - Role-based UI rendering
    - Assignment submission forms
    - Calendar with reminders
    - Dashboard with real-time data

5.2 Backend

- Runtime & Framework: Node.js with Express.js
- Database: MongoDB (Mongoose ODM)
- APIs:
    - POST /register, POST /login – User authentication
    - POST /courses, GET /courses/:id – Course management

- POST /assignments, GET /assignments/:courseId – Assignment handling
- PUT /make-admin – Super Admin user control
- POST /reminders – Reminder and calendar events
- GET /students/export – Export student data as CSV

5.3 Authentication

- Method: JWT (JSON Web Tokens)
- Storage: Tokens stored in `localStorage`

Testing

Unit Testing:

Individual components like user login, assignment upload, and submission logic were tested in isolation to ensure correctness.

Integration Testing:

APIs were tested with real frontend requests to verify end-to-end data flow (e.g., course enrollment, assignment submission, comment posting).

Manual Testing:

All user roles (student, admin, super admin) were manually tested to ensure that role-based access and UI behavior functioned correctly.

Github Snapshot

Github - BhumiVyas

Project Summary

This project is a full-stack Educational Management Web Platform designed to streamline learning, communication, and administrative workflows for students, admins, and super admins. The system supports course creation, assignment distribution, student submissions, reminders, and admin management, offering an intuitive interface and role-based access control.

The application enables:

- Students to register, join courses via code, view and submit assignments, post public/private comments, and set personal or event-based reminders.

- Admins to create and manage courses, upload assignments, monitor student submissions, and view student directories with exportable data.

- Super Admins to manage users, promote other users to admin roles, and oversee the administrative structure of the platform.

Additional modules include calendar-based reminders, assignment status tracking, and CSV exporting of student directories. All user actions are restricted and rendered dynamically based on their roles, ensuring a secure and tailored experience.

This platform is built with MongoDB (for schema-rich data), Express.js, and a React-based frontend, ensuring scalability, interactivity, and maintainability for future educational features.

Lessons Learnt

- Learned how modular design (separate models for users, courses, assignments, etc.) improves code clarity and scalability.
- Understood the importance of role-based access control for managing permissions securely across students, admins, and super admins.
- Gained experience in MongoDB schema design, especially for nested fields like submissions and comments.
- Realized that proper frontend-backend synchronization is essential for smooth and bug-free feature implementation.
- Developed a better understanding of user-centric design, focusing on practical features like reminders, unsubmission, and CSV exports.

References

- **MongoDB Official Documentation**
  https://www.mongodb.com/docs/manual/
- **Mongoose ODM**
  https://mongoosejs.com/docs/guide.html
- **Express.js Web Framework**
  https://expressjs.com/
- **React.js Official Docs**
  https://reactjs.org/docs/getting-started.html
- **Node.js Official Docs**
  https://nodejs.org/en/docs