AN IDP PROJECT REPORT

on

“Edutimely: Optimize Time, Master Tasks and Organize Schedule Planner”

Submitted

By

221FA04402
SK. Ruhi

221FA04436
G.Likhitha

221FA04538
K. Pallavi

221FA04737
Ankit Gupta

Under the guidance of

Mr.R. Prathap Kumar

Assistant Professor

SCHOOL OF COMPUTING & INFORMATICS

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

VIGNAN'S FOUNDATION FOR SCIENCE, TECHNOLOGY AND RESEARCH

Deemed to be UNIVERSITY

Vadlamudi, Guntur.

ANDHRA PRADESH, INDIA, PIN-522213.

VIGNAN'S
FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

CERTIFICATE

This is to certify that the Field Project entitled **"Edutimely: optimize Time, Master Tasks and organize schedule planner"** that is being submitted by 221FA04402 (SK. Ruhi), 221FA04436(G. Likhitha), 221FA04538(K. Pallavi) and 221FA04737(Ankit Gupta) for partial fulfilment of Field Project is a bonafide work carried out under the supervision of Mr.R. Prathap Kumar , Assistant Professor, Department of CSE.

[illegible]

Guide name & Signature
Mr.R. Prathap Kumar
Assistant Professor,
Department of CSE

[illegible]
Dr. S. V. Phani Kumar
HOD,CSE

VIGNAN'S
FOUNDATION FOR SCIENCE, TECHNOLOGY & RESEARCH
(Deemed to be University) - Estd. u/s 3 of UGC Act 1956

DECLARATION

We hereby declare that the Field Project entitled "**Edutimely: optimize Time, Master Tasks and organize schedule planner"** that is being submitted by 221FA04402(SK. Ruhi), 221FA04436(G. Likhitha), 221FA04538(K. Pallavi) and 221FA04737(Ankit Gupta) in partial fulfilment of Field Project course work. This is our original work, and this project has not formed the basis for the award of any degree. We have worked under the supervision of Mr.R. Prathap Kumar, Assistant Professor, Department of CSE.

By

221FA04402 (Ruhi),

221FA04436(Likhitha),

221FA04538(Pallavi),

221FA04737(Ankit

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREIVATIONS

• ADS	: ANDROID DATA SYNCHRONIZATION
• SDK	: SOFTWARE DEVELOPMENT KIT
• ADT	: ANDROID DEVELOPMENT KIT
• API	: APPLICATION PROGRAMMING INTERFACE
• AOSP	: ANDROID OPEN SOURCE PROJECT
• AVD	: ANDROID VIRTUAL DEVICE
• CRM	: CUSTOMER RELATIONSHIP MANAGEMENT
• UI	: USER INTERFACE
• JSON	: JAVA SCRIPT OBJECT NOTATION
• XAMPP	: WINDOWS/LINUX APACHE MYSQL PERL PHP
• PHP	: HYPERTEXT PREPROCESSOR
• IDE	: INTEGRATED DEVELOPMENT ENVIRONMENT
• HTML	: HYPER TEXT MARKUP LANGUAGE
• JDK	: JAVA DEVELOPMENT KIT
• XML	: EXTENSIBLE MARKUP LANGUAGE
• WIFI	: WIRELESS FIDELITY
• AVD	: ANDROID VIRTUAL DEVICES
• ADB	: ANDROID DEBUG BRIDGE

CHAPTER - 1

INTRODUCTION

1. INTRODUCTION

1.1 Introduction

The current project involves the development of a web application that helps students efficiently manage their academic and personal responsibilities. The main aim of this project is to provide a centralized platform where students can organize their schedules, track academic progress, and optimize their study time. Traditional methods of time management and academic tracking often lead to inefficiencies, missed deadlines, and stress. This project offers a digital solution to streamline academic planning and improve productivity.

This web application comes in handy by allowing students to balance their academic and personal commitments effectively. With features such as task management, reminders, and a Pomodoro timer, students can stay on top of their studies while reducing procrastination. The grade-tracking module, which includes a CGPA progress visualizer, helps students monitor their academic performance over multiple semesters, making it easier to set goals and stay motivated. Faculty members can access specific modules to assist students, ensuring a collaborative academic environment.
The platform will provide role-based access to three types of users: **Administrators, and Students**.

1. **Administrators**:

Administrators oversee the system's functionality and manage user accounts, user details, and stored data. They have the ability to add, update, or remove users, ensuring a secure and well-maintained platform. Administrators also handle system settings and maintain the integrity of stored academic records.

2. **Students:**

Students can utilize the platform to manage their schedules, set reminders, track their academic performance, and enhance their study efficiency. The grade-tracking module includes a graph visualizer where students can input their CGPA for each semester and view their progress over time. Additionally, students can use tools such as the

Pomodoro timer and exam preparation modules to improve focus and time management.

1.2 Literature Survey

Edu-Timely: A **Student Productivity and Time Management System** is designed to help students efficiently organize their academic and personal responsibilities. Various studies and existing digital solutions have contributed to the development of such platforms, providing insights into their effectiveness, challenges, and potential improvements.

Historically, students relied on manual methods such as paper planners, handwritten timetables, and basic calendar applications to manage their schedules. According to **Jones et al. (2016)**, traditional time management techniques often resulted in missed deadlines, poor task prioritization, and academic stress. The transition to digital productivity tools has significantly improved accessibility, scheduling flexibility, and task-tracking efficiency (**Clark & Peterson, 2018**).

Studies indicate that administrators play a crucial role in maintaining digital academic platforms. Research by **Roberts (2019)** highlights that centralized management systems help administrators oversee user roles, maintain user records, and ensure data security. A well-structured system enhances user experience and prevents unauthorized access to sensitive information.

Faculty members contribute to academic planning and student guidance. According to **Singh & Verma (2020)**, digital systems that allow faculty to monitor student progress help create a more supportive learning environment. By accessing select modules such as grade tracking and scheduling, faculty members can assist students in managing their workload more effectively.

Student engagement is a key factor in academic success. A study by **Taylor (2021)** suggests that productivity-enhancing tools such as Pomodoro timers, reminders, and visualized academic progress charts motivate students to stay on track. The use of **graph-based performance tracking**, where students can visualize their CGPA progress over multiple semesters, has been shown to encourage goal setting and self-improvement.

With the growing demand for efficient student productivity systems, integrating scheduling tools, academic tracking, and time management techniques into a single platform provides a comprehensive solution for students to enhance their academic performance.

1.2.1 Project Background

In the modern educational landscape, effective time management and academic tracking are essential for student success. Traditionally, students relied on manual methods such as paper planners, handwritten notes, and basic digital calendars to organize their academic schedules and track their progress. However, these approaches often led to inefficiencies, missed deadlines, and difficulties in monitoring long-term academic performance. As academic demands grow and students juggle multiple responsibilities, the need for a comprehensive, automated system becomes evident.

The Student Productivity and Time Management System aims to provide a centralized, digital platform that helps students efficiently plan their schedules, track academic progress, and optimize their study habits. Administrators play a key role in managing user accounts, overseeing system functionality, and ensuring secure data handling. Faculty members have access to select modules that enable them to monitor student progress and assist with academic planning. Students can manage their tasks, set reminders, utilize the Pomodoro technique for focused study sessions, and track their CGPA progress through a visualized performance graph.

By integrating these features into a single platform, the system enhances student productivity, reduces academic stress, and fosters a structured approach to learning. The inclusion of faculty oversight and administrative management ensures a well-regulated and user-friendly environment, ultimately contributing to improved academic outcomes.

1.3 Objective

The objective of the Student Productivity and Time Management System is to develop a centralized digital platform that helps students effectively manage their academic and personal responsibilities. The system aims to provide administrators with the ability to oversee user accounts, manage user details, and ensure data security, ensuring smooth and efficient platform operations.

Additionally, the platform empowers students by offering tools for task management, scheduling, reminders, and study optimization. The grade-tracking module, which includes a CGPA progress visualizer, allows students to monitor their academic performance over multiple semesters, helping them set goals and stay motivated. The Pomodoro timer and exam preparation tools further aid students in improving focus and productivity.

Ultimately, this project seeks to enhance time management, academic tracking, and student engagement by providing an all-in-one solution for academic success. By integrating these features into a structured and user-friendly system, the platform fosters productivity, reduces stress, and promotes better academic performance.

1.4 Project Description

In this section all features in application are explained in brief.

1) Administrator

- i. Sign Up
- ii. Login
- iii. Features
    - a. Manage Users
    - b. Manage System Data
    - c. Monitor Activity
- iv. Logout

i. **Sign Up**:

Administrators can create an account by providing their name, email, phone number, and a secure password. Once registered, they gain full control over system management.

ii. **Login:**

Registered administrators can log in using their credentials. If they forget their password, they can use the **Forgot Password** option to reset it via email**.**

iii. <u>**Features:**</u>

a. **Manage Users:**

Administrators can add, edit, or delete user accounts, ensuring proper role assignments and secure access for faculty and students.

b. **Manage System Data:**

Administrators oversee the platform's database, ensuring data integrity and smooth functionality.

c. **Monitor Activity:**

They can track user engagement and maintain platform security.

iv. <u>**Logout:**</u>

Administrators can securely log out after managing the system.

2) Student

i. Sign Up

ii. Login

iii. Features

a. Upload Certificates

b. View Approved certificates

c. Track verification Status

iv. Logout

i. <u>**Sign in**</u>**:**

Students can register by providing their details, including name, email, and password, to create an account.

ii. <u>**Login:**</u>

Students log in with their credentials to upload and track their achievement certificates. The **Forgot Password** feature allows them to recover their account if needed.

iii. **<u>Features:</u>**

a. Task & Schedule Management:

Students can organize their schedules, set reminders, and plan study sessions efficiently.

b. Grade Tracking:

A CGPA progress visualizer allows students to track their academic performance over multiple semesters.

c. Pomodoro Timer:

Helps students stay focused by breaking study sessions into intervals.

d. Exam Preparation Tools:

Provides study aids to help students prepare for exams effectively.

iv. **<u>Logout:</u>**

Students can securely log out once they have completed their tasks.

CHAPTER - 2

SOFTWARE REQUIREMENT SPECIFICATION

2. SOFTWARE REQUIREMENTS SPECIFICATION

2.1 Requirement Analysis

For easy access, scalability, and portability, we propose to develop a **web- based system** that runs on **Windows or Ubuntu**, as these are widely used operating systems. The system will enable students to upload their achievement certificates while allowing teachers to verify and manage them efficiently. Using **MongoDB** as the database ensures **flexibility, scalability, and efficient handling of unstructured data**, making it ideal for storing student certificates in various formats. The required documents for the development process include:

1. Problem statement
2. Data flow diagrams
3. Use case diagram
4. Other UML diagrams.

The above mentioned documents gives us diagrammatical view of the system what we are going to develop.

2.2 Problem Statement

Students often face challenges in managing their time, organizing academic tasks, and tracking progress, leading to missed deadlines, inefficient study habits, and lack of focus. Traditional planning methods, such as paper planners or scattered digital tools, fail to provide a centralized and automated solution. There is a need for a comprehensive digital planner that integrates task management, scheduling, productivity tracking, and AI-driven insights to help students stay organized, improve focus, and achieve their academic goals efficiently.

2.3 Functional Requirements

- **Student Registration and Login**

  2.1. The website provides **Sign In** and **Login** options. If a student is not registered, they must sign up.
  2.2. Our website takes first name.
  2.3. Our website takes last name.
  2.4. Our website takes phone number.
  2.5. Our website takes email address.

2.6. Our website takes password.

2.7. Our website asks again to retype the password for confirmation of the password

2.8. Our website takes institution name

The Software Requirement Specification (SRS) **for** Edutimely **outlines key** functional requirements **essential for efficient academic task management and student progress tracking.**

- **Task Management** enables students to create, prioritize, and track their tasks, ensuring timely completion of academic activities. It supports task categorization, setting deadlines, and monitoring progress.
- **Exam Preparation Calendar** facilitates daily, weekly, and semester-wise planning, allowing students to structure their study schedules efficiently and avoid last-minute preparation.
- **Grade Management** helps track academic performance, generate reports, and provide insights into progress over time, enabling students to assess their strengths and areas of improvement.
- **Pomodoro Timer** enhances focus by breaking study sessions into structured intervals with breaks, improving concentration and productivity.
- **Reminder System** ensures students receive timely alerts for deadlines, assignments, and exams, reducing the risk of missing important academic activities.
- **Performance Tracking** identifies students' academic progress, highlights areas for improvement, and provides personalized feedback to enhance learning outcomes.

These **functionalities** collectively contribute to an **efficient learning system**, helping students manage their schedules, improve academic performance, and stay organized throughout their academic journey.

2.4 Software Requirement Specification

The **Edutimely** application is designed to optimize student productivity through efficient task management, progress tracking, and academic planning. The software stack includes modern web technologies ensuring **scalability, security, and performance**.

2.4.1 Purpose

1. Frontend

- Built using **React.js** with **Context API** or **Redux** for state management.
- Ensures a responsive UI with **Tailwind CSS** for styling.
- Uses **Recharts** for visualizing student progress and performance analytics.

2. Backend

- Developed using **Node.js** and **Express.js** for handling RESTful API requests.
- Implements **MVC architecture** for clean and modular code structure.
- Supports **CRUD operations** for tasks, schedules, and progress tracking.

3. Database

- Uses **MongoDB** as a NoSQL database for storing user tasks, events, and academic data.
- Implements **Mongoose** for object data modeling (ODM).
- Ensures **data consistency and retrieval efficiency**.

4. Security

- **bcrypt** for password hashing to enhance user security.
- **JWT (JSON Web Token)** for authentication and authorization.
- Implements **CORS and Helmet** for additional security measures.

5. Hosting & Deployment

- **Frontend** deployed on **Netlify/Vercel** for fast and reliable hosting.
- **Backend** hosted on **Heroku/AWS** for seamless API performance.
- **MongoDB Atlas** for cloud-based database hosting.

This technology stack ensures that Edutimely is **secure, scalable, and efficient**, meeting the needs of students for better academic organization and productivity.

2.4.2 Scope of the project

- The **AI-Based Personalized Recommendations:** Enhance task scheduling and study plans using AI-driven insights based on user performance and habits.
- **Cross-Platform Integration:** Develop mobile and desktop applications for seamless accessibility and offline functionality.

1. Users (Students)

Users are the primary stakeholders of Edutimely, utilizing the platform to manage their academic and personal tasks efficiently.

- **Task Management**: Create, prioritize, and track daily tasks.
- **Exam Preparation**: Plan study schedules, set goals, and track preparation progress.
- **Grade Management**: View academic performance, receive feedback, and reports.
- **Pomodoro Timer**: Use time-blocking techniques for enhanced focus.
- **Reminder System**: Get timely alerts for deadlines and important events.
- **Performance Tracking**: Monitor progress through analytics and personalized reports.

User Features:

- Access to a **Dynamic Calendar** for event planning.
- Set **Study Reminders** and receive notifications.
- View **CGPA and semester-wise grades** through the **Grade Tracking System**.
- Customize study strategies based on **performance insights**.

2. Admins

Admins manage the platform to ensure a **smooth user experience**, system integrity, and security.

Admin Responsibilities:

- **User Account Management**: Handle user registrations and access permissions.
- **System Configuration**: Manage application settings and security measures.
- **Performance Monitoring**: Oversee the efficiency and reliability of the system.
- **Report Generation**: Analyze student progress and system performance through data reports.

Admin Features:

- **Manage User Accounts**: Modify, delete, or reset user credentials.
- **Generate System Reports**: Track overall student engagement and feature usage.
- **Set Global Notifications & Reminders**: Configure important academic or system-wide alerts.
- **Monitor Security & Authentication**: Oversee **bcrypt encryption**, **JWT-based authentication**, and **system logs**.

2.4.3 Technologies Used

The Lesson-Plan Creator project is built using modern web development technologies to ensure efficiency, scalability, and a seamless user experience. The following technologies are utilized:

1. **Frontend:**

- **Vite + React (JSX):** A fast build tool and framework for creating application
- **Context API/Redux:** For efficient state management across components.
- **Tailwind CSS & Custom CSS:** For a responsive and visually appealing UI/UX.

2. **Backend:**

- **Node.js & Express.js:** To build a RESTful API for handling requests.
- **Google Calendar API:** To integrate calendar functionality for scheduling and reminders.

3. **Database:**

- **MongoDB:** A NoSQL database used for storing tasks, events, notes, grades, and user data.

4. **Security:**

- **bcrypt:** For hashing passwords.
- **JWT (JSON Web Token):** For authentication and session management.

5. **Hosting & Deployment:**

- **Netlify/Vercel:** For deploying the frontend.
- **Heroku/AWS:** For backend hosting.

6. **Additional Libraries & Tools:**

- **Chart.js or Recharts:** For visualizing progress graphs in the agenda & grade tracker.
- **Framer Motion:** For smooth UI animations.
- **Cron Jobs & Nodemailer:** For automated reminders and email notifications.

2.4.4 Overview

The **Lesson-Plan Creator** is a comprehensive productivity tool designed to help students efficiently manage their academic schedules, tasks, and progress. It integrates multiple features to optimize time management and enhance study efficiency. The application includes a **Calendar System** with Google Calendar API integration, allowing users to add, delete, and set reminders for events based on their preferred date and time. The **Agenda Tracker** monitors daily task completion and visualizes progress through interactive graphs. A **Pomodoro Timer** enables users to select focused study intervals (55, 45, 35, 25, 15, or 5 minutes) with an alarm and a scheduled break to maintain productivity. The **Notes Picker** allows users to store important notes, upload files, and save exam timetables. The **Grade Tracker** records semester-wise grades, calculates the overall CGPA, and uses AI-driven analytics to display progress graphs with target improvement indicators. Lastly, the **Task Manager** lets users create, track, and mark tasks as completed or delayed, ensuring effective day-to-day planning. Built with **Vite + React (JSX) for the frontend** and **Node.js & Express.js for the backend**, the application prioritizes seamless UI/UX, efficient state management, and secure data storage using MongoDB.

2.5 Software Requirements

The software interface is the operating system, and application programming interface used for the development of the software.

• Operating System	-	Windows 10/11, Linux (Ubuntu), macOS
• Frontend	-	React.js with Context API/Redux for state management
• Stack	-	MERN Stack (MongoDB, Express.js, React.js, Node.js)
• Hosting	-	Netlify/Vercel for frontend, Heroku/AWS for backend

HTTP 1.1

• Back End	-	Node.js & Express.js for RESTful API development
• Cloud Platform	-	Amazon Web Services (AWS) for cloud deployment
• Technologies	-	JavaScript, JSX, Redux, REST API, CSS, Tailwind CSS

2.6 Hardware Requirements

CLIENT			
OPERATING SYSTEM	SOFTWARE	DISK SPACE	RAM
Any operating system	Any Web Browser	Minimum 10GB for local development	Minimum 4GB (Recommended: 8GB+)

Table 2.1 Client Requirements

SERVER				
OPERATING SYSTEM	SOFTWARE	PROCESSOR	RAM	DISK SPACE
Windows, macOS, or Linux	Node.js, Express.js, MongoDB, Google Calendar API	Intel i3 or above / AMD Ryzen equivalent	Minimum 4GB (Recommended: 8GB+)	Minimum 10GB

Table 2.2 Server Requirements

2.7 Functional Requirements (Modules)

The **Edutimely** is a comprehensive academic management tool designed to help students track their progress, manage tasks, and improve time management skills effectively. The platform integrates multiple features aimed at enhancing productivity and academic performance. Below are the detailed **functional requirements** of the system:

1. Task Management

- Users can **create, prioritize, and track tasks**.
- Tasks can be marked as **completed** or **delayed**, ensuring clear visibility of progress.
- Stores **day-by-day** task data for future reference.
- Allows filtering tasks based on **priority levels, due dates, and categories**.

2. Exam Preparation Calendar

- Uses **Google Calendar API** to integrate a **personalized event system**.

- Users can **add, delete, and modify events** related to academic schedules.
- Provides **daily, weekly, and semester-wise planning**.
- Includes **reminders and notifications** based on user preferences.
- Supports **color-coded categorization** of events for better organization.

3. Grade Management

- Allows users to **input semester-wise grades** and **track academic progress**.
- Calculates **overall CGPA** based on user-inputted grades.
- Uses **AI-based analytics** to visualize academic trends and suggest areas for improvement.
- Generates **detailed reports** on academic performance, which can be downloaded.

4. Pomodoro Timer

- Implements a **customized Pomodoro timer** with **55, 45, 35, 25, 15, and 5-minute intervals**.
- Displays time in a **circular UI** where users can pick their preferred duration.
- Rings an **alarm notification** after time completion.
- Includes **5-minute break intervals** to encourage structured study habits.

5. Reminder System

- Sends **timely alerts for deadlines, events, and exam schedules**.
- Users can **set custom reminders** with date and time selection.
- Supports **push notifications and email alerts**.

6. Performance Tracking

- Tracks **daily task completion and improvement trends**.
- Provides **graphical insights** to visualize academic progress over time.
- Uses **AI-driven analytics** to predict **target achievement** and suggest improvement strategies.
- Helps users **compare past performance with current progress** for better self-evaluation.

2.8 Non-Functional Requirements

2.8.1 Performance requirements:

The **Edutimely** is designed to be a highly secure, scalable, and performance-optimized system that enhances the user experience while ensuring data integrity and availability. Below are the **detailed non- functional requirements** that define the system's quality attributes:

1. Security

- Implements **JWT (JSON Web Token)** for **secure authentication** and **session management**.
- Uses **bcrypt hashing** for encrypting user passwords.
- Supports **role-based access control (RBAC)** to differentiate access between **students, administrators, and faculty**.
- Implements **automatic logout after inactivity** to prevent unauthorized access.
- Ensures **secure API endpoints** using **HTTPS** and **CORS** policies.

2. Performance Optimization

- Implements **caching mechanisms (e.g., Redis)** to **reduce server load** and **optimize API responses**.

- Uses **lazy loading** for efficiently rendering UI components, ensuring fast page loads.
- Optimizes **database queries** to reduce **response time** and **enhance speed**.
- Minimizes **API calls** by utilizing **data batching techniques**.
- Uses **code-splitting** and **minification** for front-end performance improvements.

3. Scalability

- The system is designed to handle a **large number of users** simultaneously without degrading performance.
- Uses **MongoDB (NoSQL database)**, which allows **horizontal scaling** and efficient data storage.
- Implements **load balancing** strategies to distribute traffic across multiple servers.
- Supports **serverless architecture** for seamless deployment and maintenance.

4. Usability

- Provides a **user-friendly interface** with an intuitive **dashboard and navigation**.
- Features a **responsive design** to ensure a **seamless experience on desktops, tablets, and mobile devices**.
- Includes **tooltips, user guides, and tutorials** to assist users in understanding the features.
- Supports **dark mode** and **accessibility features** for enhanced usability.

5. Reliability

- Uses **cloud-based hosting (AWS/Heroku/Netlify/Vercel)** for **high availability**.
- Ensures **continuous integration (CI/CD) pipelines** for seamless updates and deployments.
- Implements **automated database backups** to prevent data loss.
- Monitors **server uptime** and provides **failover mechanisms** to ensure uninterrupted service.

2.8.2 Security

The **Edutimely** system shall be a **secure** and **authenticated** platform designed for managing student schedules, tasks, and progress tracking. The system shall be developed using **Vite + React (JSX) for the frontend** and **Node.js with Express.js for the backend**, ensuring secure API communication.

2.8.3 Standards Compliance:

The **Edutimely** shall adhere to **consistent coding standards** and **best practices** to ensure maintainability and scalability.

2.8.4 Availability:.

The **Edutimely** shall be available **24/7** for users, ensuring

seamless access to schedules, tasks, and progress tracking.

2.8.5 Portability:.

The **Edutimely** shall be accessible from **any device with an internet connection**, ensuring smooth usability for students and teachers.

2.8.6 Reliability:

The **Edutimely** is designed to be a **highly reliable** and **efficient** platform for students and educators.

2.9 External Interface Requirements

2.9.1 User Interface

A critical aspect of the **Edutimely** is designing a **user-friendly and intuitive interface** to enhance usability. The web application layout is implemented using **React.js with JSX**, ensuring a **modern and interactive user experience**. The **frontend UI** is styled with **Tailwind CSS / Material UI**, providing a clean, professional, and responsive design.

2.9.2 Event log/database

User table:

1. **User ID** (Unique identifier)Phone number
2. Full Name
3. Email address
4. Role (students/Admin)
5. Password

Task Table:

1. Task ID (Unique identifier)
2. Task Name
3. Task Description
4. Priority Level (High, Medium, Low)
5. Due Date
6. Completion Status (Completed/Pending)
7. User ID (Foreign key linking to Users Table)

Calendar events Table:

1. Event ID
2. Event Name
3. Date & Time

4. Event Type (Exam, Assignment, Meeting, etc.)
5. Reminder Status
6. User ID

Grade Tracking Table:

1. Grade ID
2. Course Name
3. Semester
4. Grade Obtained
5. CGPA Calculation
6. User ID

Notes Table:

1. Note ID
2. Title
3. Content
4. File Attachments (if any)
5. User ID

Pomodoro Timer Log Table:

1. Session ID
2. User ID
3. Timer Duration
4. Break Duration
5. Completion Status

This structured database design ensures **efficient data retrieval, management, and security** while maintaining a **smooth user experience**.

2.10 Feasibility study

The feasibility study is a critical step in evaluating whether the **Edutimely** system is viable in terms of **cost, time, technical feasibility, and operational efficiency**. The study examines the following factors:

2.10.1 Organisational Feasibility

The application is designed to enhance student productivity by providing a structured and efficient way to plan lessons, track tasks, and monitor progress. The system integrates Google Calendar API, automated reminders, and progress-tracking graphs**,** ensuring a seamless and interactive user experience**.** The platform is easy to use with an intuitive interface**,** requiring minimal effort for students to navigate and utilize its features effectively**.**

2.10.2 Economic Feasibility

The project is highly cost-effective as it leverages **free and open-source technologies** such as **React.js, Node.js, and MongoDB**, eliminating licensing costs. By utilizing **Netlify/Vercel for frontend hosting** and **Heroku/AWS for backend hosting**, infrastructure expenses are minimized, making the system financially viable. Additionally, the **cloud-based architecture** reduces the necessity for high-end hardware, ensuring accessibility for students without significant investments in computing resources. This affordability makes the system an ideal choice for students looking to optimize their academic planning without financial constraints.

2.10.3 Technical Feasibility

The project follows an **agile development methodology**, ensuring efficient planning, continuous testing, and timely deployment. The **modular architecture** enables **incremental feature updates**, reducing overall development time while maintaining system stability. Deployment is further streamlined using **CI/CD pipelines**, allowing for rapid and reliable updates without disrupting the user experience. This structured development approach ensures that new features and improvements can be introduced seamlessly while keeping the application functional and efficient.

2.10.4 Behavioural Feasibility

The application is behaviourally feasible since it requires no technical guidance, all the modules are user friendly and execute in a manner they were designed to.

CHAPTER - 3

ANALYSIS & DESIGN

3 ANALYSIS & DESIGN

3.9 Introduction

3.9.1 Purpose

In this section the purpose of the document and the project is described.

3.9.1.1 Document Purpose

This Software Design Document (SDD) represents the architecture and design of the **Edu-timely** web application. It serves as a communication medium between developers and stakeholders to ensure the system is built according to the requirements.

3.9.1.2 Project Purpose

The prime purpose of this "Edutimely" application is to create a comprehensive web-based platform that helps students and teachers manage their academic schedules effectively. This application communicates with a remote server to store and retrieve data as required. It operates seamlessly with internet connectivity, ensuring accessibility for users anytime. Students can organize their calendar schedules, plan daily activities, track CGPA with graphical insights, and utilize a Pomodoro timer for better focus. This system enhances academic productivity by providing a structured and interactive platform for students and teachers.

3.9.2 Scope

In this section the scope of the document and the project is explained in brief.

3.9.2.1 Document Scope

This document presents a comprehensive high-level design overview of Edutimely, outlining the key architectural components and structural elements that define the system.It serves as a blueprint for the platform's development, ensuring seamless integration of various functional aspects. The document provides insights into the system architecture, detailing how different layers interact, including the front-end, back-end, and database. It also explores the UI/UX components, emphasizing user friendly design principles and accessibility features to enhance the learning experience.

Additionally, the document covers database design, defining the structure for efficiently storing and retrieving data while ensuring scalability and security. The functional modules section breaks down the core features, such as user authentication, course management, progress tracking, and notifications. Lastly, the document outlines deployment strategies, discussing the hosting environment, cloud services, and DevOps practices to ensure smooth deployment, maintenance, and updatesforEdutimely.

3.9.2.2 Project Scope

The **Edutimely** application consists of multiple modules, each serving a specific functionality:

1. **Calendar & Event Management**
   - ✓ Integration with **Google Calendar API**
   - ✓ Add, delete, and manage events
   - ✓ Set reminders for events
2. **Agenda Tracker**
   - ✓ Displays task progress using graphs
   - ✓ Shows improvement/decrement in work over time
3. **Pomodoro Timer**
   - ✓ Circular UI with time options: 55, 45, 35, 25, 15, and 5 minutes
   - ✓ Alerts user upon task completion and suggests a 5-minute break
4. **Notes Picker**
   - ✓ Store and manage notes
   - ✓ Upload and organize files (e.g., exam timetables, lecture notes)
5. **Grade Tracker**
   - ✓ Allows users to input semester grades
   - ✓ Calculates CGPA
   - ✓ Displays academic progress graphically
   - ✓ AI-driven growth analysis with **target achievement graph**
6. **Task Manager**
   - ✓ Create, edit, and delete tasks
   - ✓ Mark tasks as completed or delayed
   - ✓ Day-by-day task tracking

3.10 System Overview

The Edutimely application is built using a modern MERN (MongoDB, Express.js, React.js, Node.js) stack, ensuring a scalable, efficient, and user-friendly experience. The system leverages various development tools and technologies to create a seamless, full-stack productivity platform with a focus on real-time updates, responsiveness, and AI-driven analytics.

3.10.1 Development Tools

To achieve high performance, maintainability, and flexibility, Edutimely uses the following development tools and technologies:

3.10.2 Visual Studio Code

A lightweight yet powerful code editor used for both frontend and backend development. Supports extensions for debugging, code formatting, and auto-completions, improving development efficiency. Features built-in Git integration, enabling version control and team collaboration. Supports Live Server, allowing real-time previews of frontend changes.

3.10.3 Node.js & Express.js

Node.js is a **JavaScript runtime environment** that enables the execution of JavaScript outside the browser, making it ideal for backend development. **Express.js** is a **lightweight and flexible framework** for handling HTTP requests and responses efficiently. Used to create **RESTful APIs** that manage user authentication, tasks, notes, events, and academic tracking. Features **middleware** for logging, error handling, and security improvements. Supports **asynchronous operations**, ensuring faster request handling and real-time updates.

3.10.4 MongoDB

A **NoSQL database** used to store **structured and unstructured** data such as:

- **User profiles**
- **Tasks and deadlines**
- **Notes and uploaded files**
- **Event schedules**
- **Grade records**

Provides **scalability**, allowing efficient storage and retrieval of large datasets. Uses

Mongoose ORM for defining schema-based models, ensuring data integrity.

Supports **real-time database updates**, enabling instant synchronization of changes across devices.

.

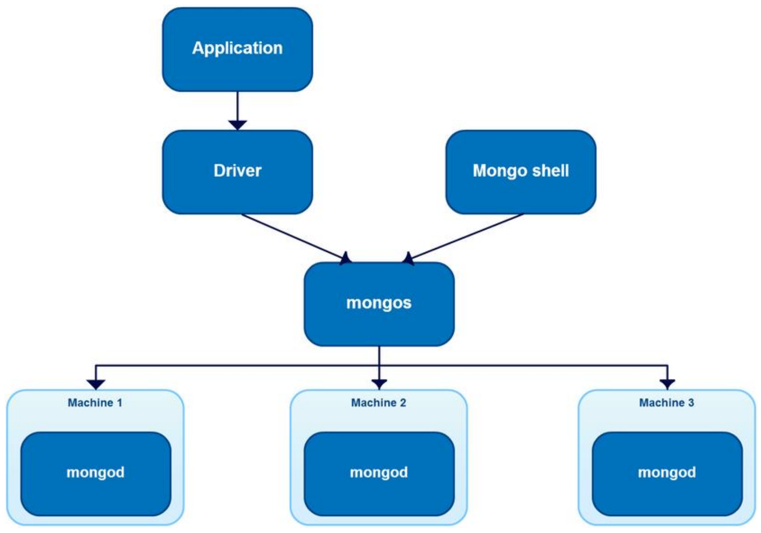


Figure 3.1: *MongoDB Replica Set Architecture*

3.11 System Architecture

3.11.1 Architectural Design

Web application architecture is a framework connecting different elements to enable a web experience. It is the backbone of our daily internet browsing: typing in a URL and viewing and interacting with the website while the browser communicates with the server is one of the ways to describe what is web application architecture.

Attributes of a well-built web application architecture:

Components of Web Application Architecture

Web application architectures consist of application components, middleware systems, and databases. They can be divided into two groups:

- UI/UX components
- Structural components

UI/UX components include dashboards, statistical data, notification elements, layouts, activity tracking, and other elements. These components create the visuals of a web page and lay the foundation for user experience.

Meanwhile, **structural components** include the web application server and the database server. knowledge of HTML, JavaScript, and CSS, as well as Python, PHP, Java, Ruby, .NET and Node.js are required to create them.

When it comes to building the components, there are several models to choose from:

- 1 web server and 1 database
- 2 web servers and 2 databases
- More than 2 web servers and databases

One web server with one database is the simplest model. With this web server architecture, the successful operation of an application depends on server stability. In other words, if there is a problem with the server, the app will not work. Still, the model is sufficient for testing and private sessions.

Using **one database for two web servers** is a more reliable model, as there is a backup server. On the other hand, ensuring the database is secure and always running is important.

Having **more than two databases and web servers** is the most dependable option. Due to its ability to manage and process large amounts of data, this model is a solid the basis for an enterprise web application architecture.

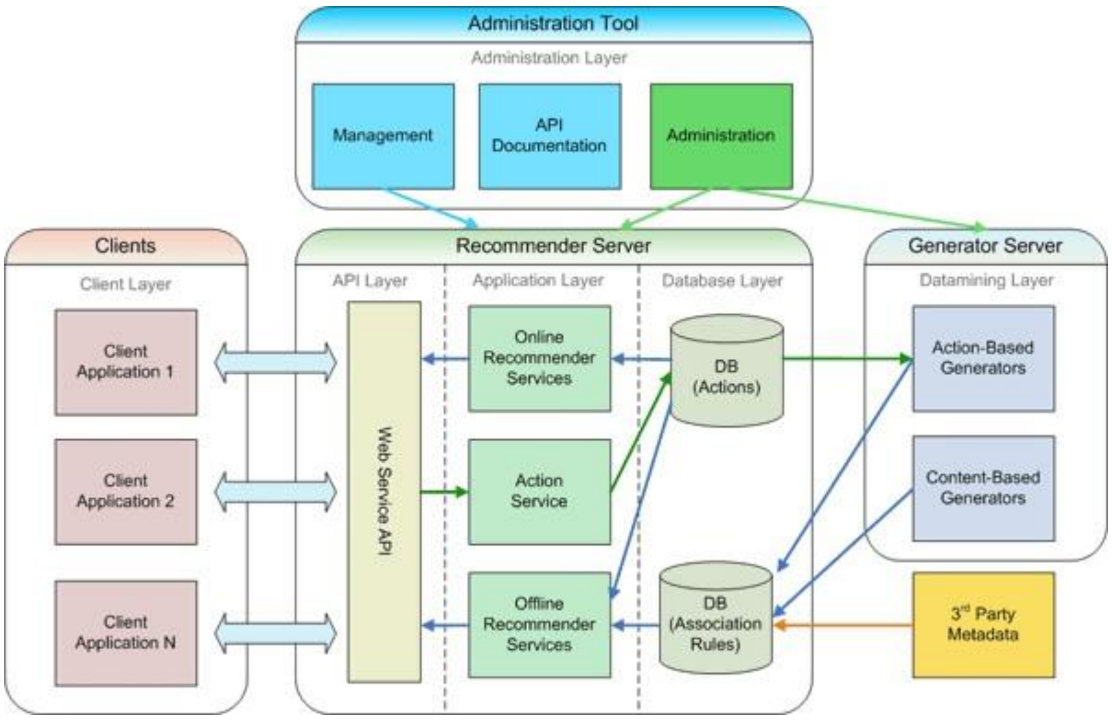


Figure 3.2 **Web Application Architecture Diagram**

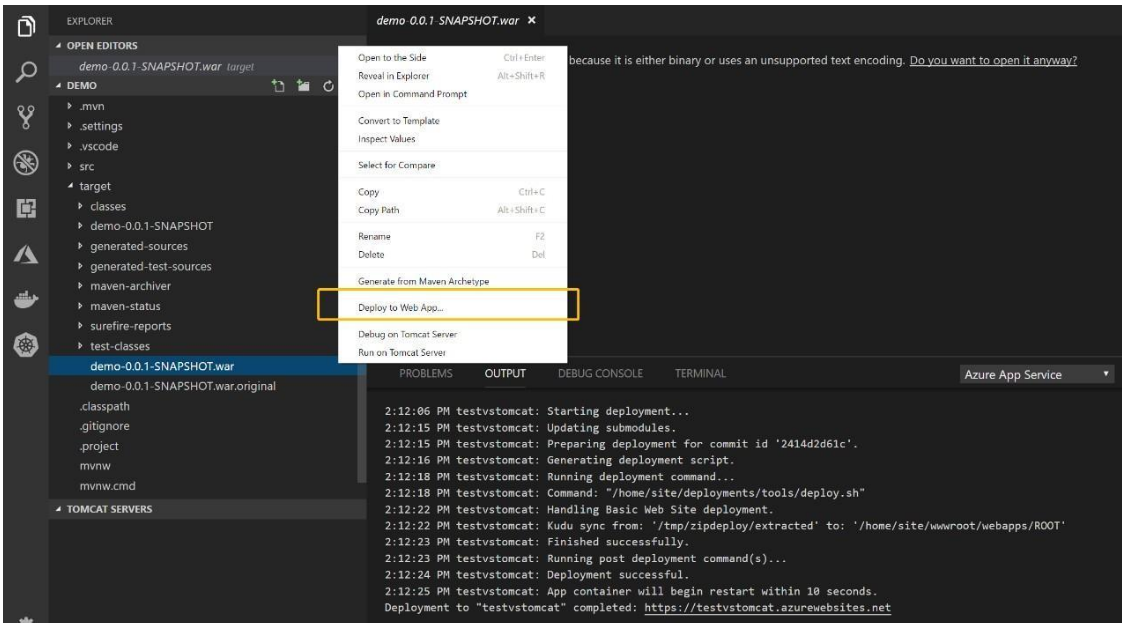
3.11.2 Build and deploy to a Web App in cloud

The deploy process leverages the Azure Account extension (installed along with the Azure Functions extension as a dependency) and you need to sign in with your Azure subscription. Or in the Amazon web services through the aws account extension and you need to sign in with your Aws subscription.

Once you have signed in, you can open the command prompt or terminal window and build the project using Maven commands. This will generate a new `name.war` or `name.jar` file in the `target` directory.

```
mvn clean package
```

After building the project, open the target directory in VS Code Explorer. Right-click on the name.war or name.jar file and choose **Deploy to Web App**, and follow the prompts to choose the Web App for your deployment.



because it is either binary or uses an unsupported text encoding. Do you want to open it anyway?

```
2:12:06 PM testvstomcat: Starting deployment...
2:12:15 PM testvstomcat: Updating submodules.
2:12:15 PM testvstomcat: Preparing deployment for commit id '2414d2d61c'.
2:12:16 PM testvstomcat: Generating deployment script.
2:12:18 PM testvstomcat: Running deployment command...
2:12:18 PM testvstomcat: Command: "/home/site/deployments/tools/deploy.sh"
2:12:22 PM testvstomcat: Handling Basic Web Site deployment.
2:12:22 PM testvstomcat: Kudu sync from: '/tmp/zipdeploy/extracted' to: '/home/site/wwwroot/webapps/ROOT'
2:12:23 PM testvstomcat: Finished successfully.
2:12:23 PM testvstomcat: Running post deployment command(s)...
2:12:24 PM testvstomcat: Deployment successful.
2:12:25 PM testvstomcat: App container will begin restart within 10 seconds.
Deployment to "testvstomcat" completed: https://testvstomcat.azurewebsites.net
```

Figure:3.3 Deploy to web App

Open the **Output** window in VS Code to view the deployment logs. Once the deployment is completed, it will print out the URL for your Web App. Click the link to open it in a browser, you can see the web app running on Azure or Aws.

3.11.3 Application Components

Any web application, big or small, contains four major components:

View Layer

When you consider an MVC application, the View layer component gives an interface to the application. Regardless if it is for users with a browser or for another application using Web services. View layer is the bridge for getting the data in and out of the application.

It does not have business logic, like calculating interest for a banking application or storing items in a shopping cart for an online catalogue. It also does not contain any code for existing data to or retrieving data from a data source. Business logic is managed by the Model layer. View layer is more focused on the interface.

Business Layer

It is also known as Business Logic or Domain Logic or Application Layer. The function of the business layer is to accept user requests from the browser, processes them, and determine the routes through which the data will be accessed. The workflows by which the data and requests travel through the back end lay encoded in a business layer.

Data Access Layer

This layer is built to keep the code you use to pull data from your data store like database, flat files, or web services separate from business logic and presentation code. So even if you have to change data stores, you don't end up rewriting the whole thing. There are many ORM frameworks that are blending the DAL with other layers which makes development easy during web application development services.

Error handling, security, logging

When you build a web application, people generally tend to focus on the end-goal, building and testing only for situations when things go right. Alas! things rarely go right all the time in the real world.

This is where error handling is a vital part of any application's user experience. And, if it is done well, it can leave your users feeling informed and properly considered.

3.11.4 Overall Software Architecture

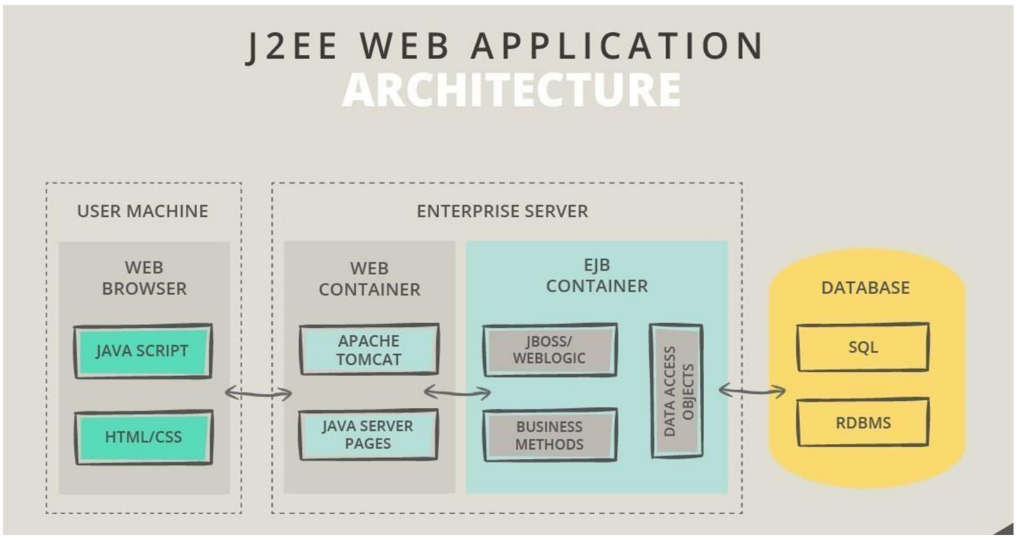


Figure 3.4 Java-based web application architecture

The architecture shown in above figure is used in the sync operation where the data from web pages goes to web server (Apache Tomcat) to database server (MySQL). JSP is used here because of the interaction it can offer with the databases and it is easy to deploy on the XAMPP web server and here it sits in middle as shown in figure. SQL

CHAPTER - 4

MODELING

4 MODELING

4.1 Design

The design of the **Lesson-Plan Creator** project follows a structured Object-Oriented Analysis and Design (OOAD) approach, ensuring efficient implementation of its features. Unified Modeling Language (UML) diagrams are utilized to visually represent the system, aiding in the specification, modification, and documentation of various components. The project integrates several functionalities, including a **Calendar** using Google Calendar API for event creation, deletion, and reminders; an **Agenda Tracker** for monitoring task completion with graphical progress visualization; a **Pomodoro Timer** offering time intervals in a circular selector with alerts and breaks; a **Notes Picker** for storing notes, file uploads, and exam schedules; a **Grade Tracker** to manage semester-wise grades and calculate CGPA with AI- driven growth analysis; and a **Task Manager** for creating, tracking, and managing tasks with delay alerts. The frontend is developed using **Vite + React (JSX)** with a focus on an intuitive **UI/UX**, ensuring a seamless user experience.

4.1.1. Use Case Diagram

In the Unified Modeling Language (UML), the use case diagram is a type of behavioral diagram defined by and created from a use-case analysis. It represents a graphical over view of the functionality of the system in terms of actors, which are persons, organizations or external system that plays a role in one or more interaction with the system. These are drawn as stick figures. The goals of these actors are represented as use cases, which describe a sequence of actions that provide something of measurable value to an actor and any dependencies between those use cases.

In this application there is only actor – soldier and below is the use case diagram of this application.

For the **Lesson-Plan Creator** project, the **Student** and **Admin** are the primary actors. The **Student** interacts with multiple modules

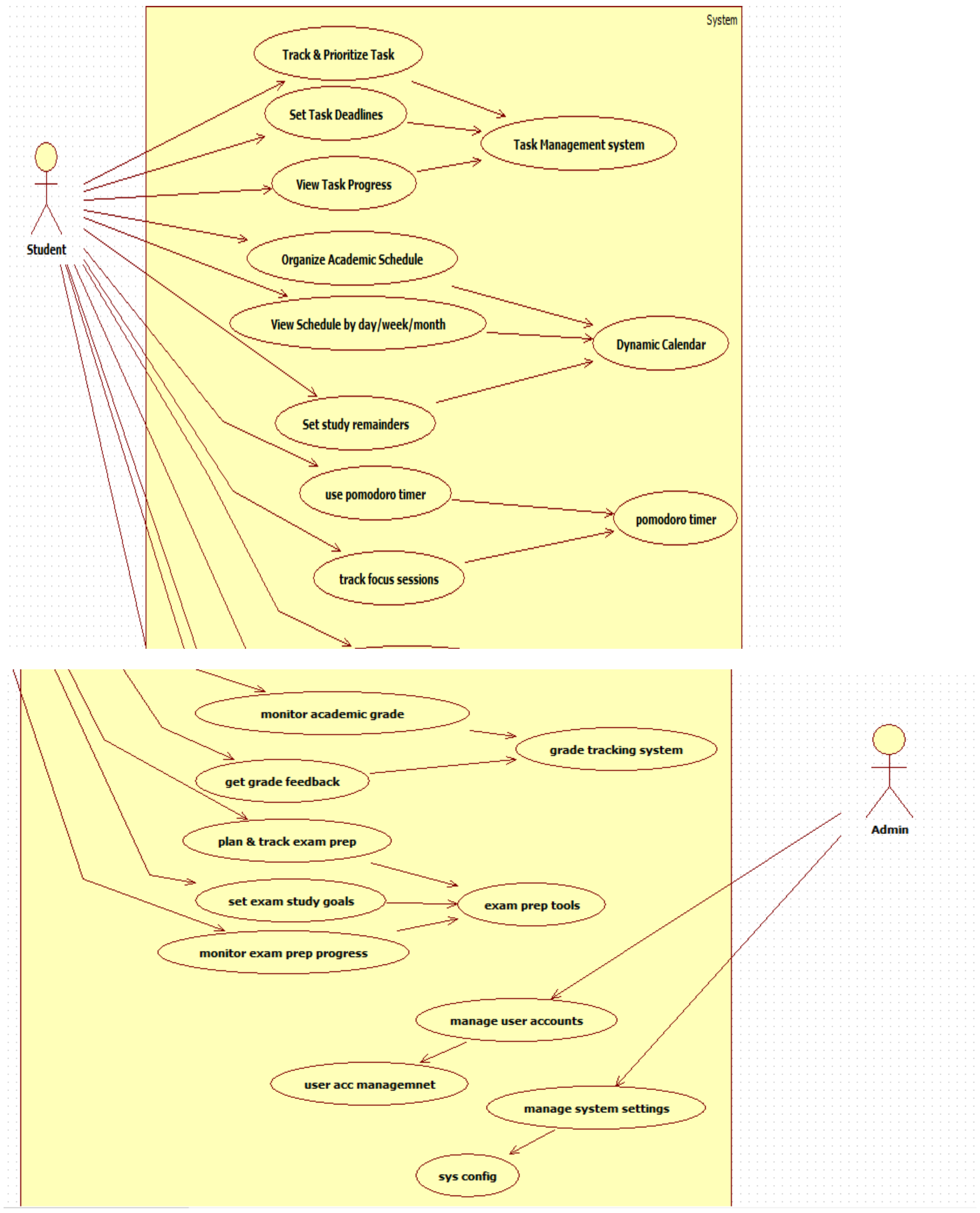


Figure 4.1 Use Case Diagram for Edutimely

4.1.2 Sequence Diagram

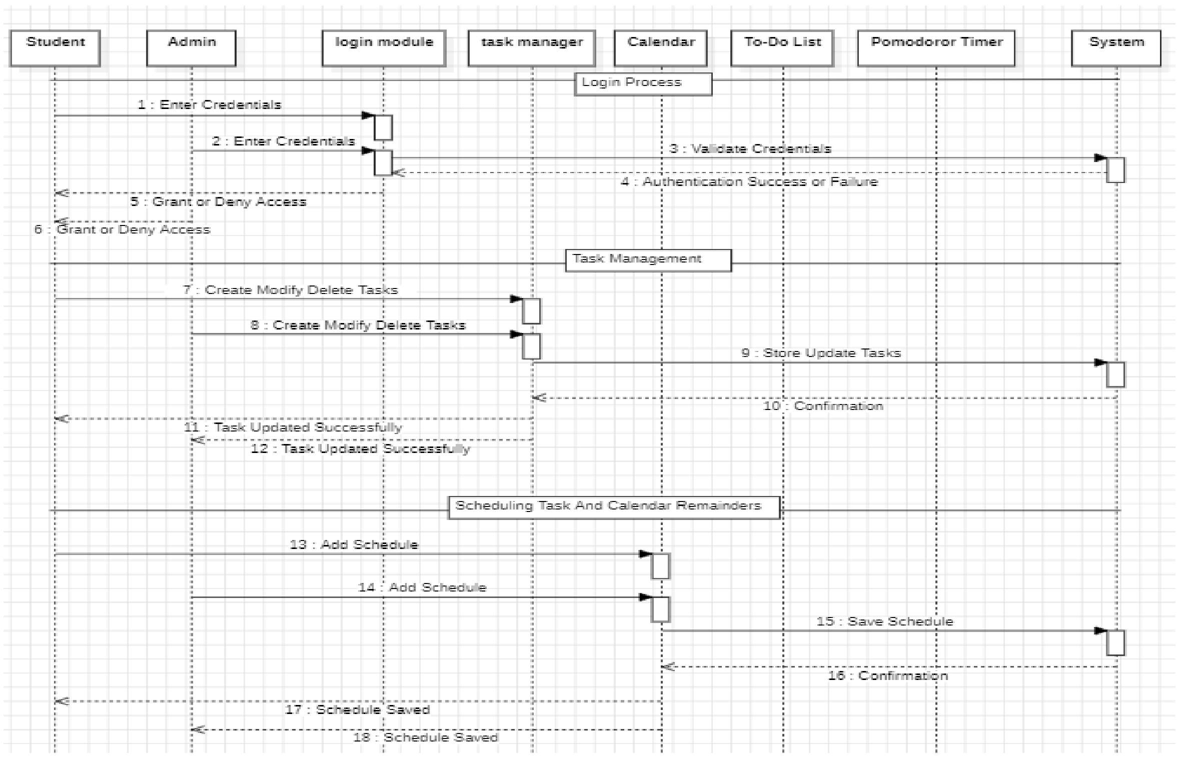
A **UML Sequence Diagram** represents the flow of interactions between objects in a system over time. It visually maps the sequence of messages exchanged among various components, ensuring smooth communication within the **Lesson-Plan Creator** project.

Sequence diagrams help illustrate **object interactions** in specific use cases, such as **task management, calendar scheduling, grade tracking, and exam preparation**. The interactions are depicted vertically, where **time progresses from top to bottom**, showing how events are triggered sequentially.

In the **Lesson-Plan Creator**, some key interactions include:

1. **Task Management System** : The student initiates task creation, sets deadlines, and views progress. The system responds by updating the task list and tracking completion.
2. **Dynamic Calendar** : The student interacts with the calendar to add or remove events, update schedules, and set reminders, ensuring organized study planning.
3. **Pomodoro Timer** : The student selects a preferred time interval for focus sessions. The system starts the timer, tracks progress, and alerts the user when the session ends.
4. **Grade Tracking System** : The student requests grade data, views academic reports, and receives AI-driven feedback. The system processes inputs and updates the grade graph dynamically.
5. **Exam Preparation Tools** : The student sets study goals, tracks preparation, and updates progress. The system analyzes and reflects improvements using a growth graph.
6. **Administrator Operations** : The admin manages user accounts, updates system configurations, and generates reports, ensuring smooth platform functionality.

By leveraging **UML Sequence Diagrams**, the **Lesson-Plan Creator** effectively demonstrates real-time interactions, making the system's behavior and data flow easier to understand.



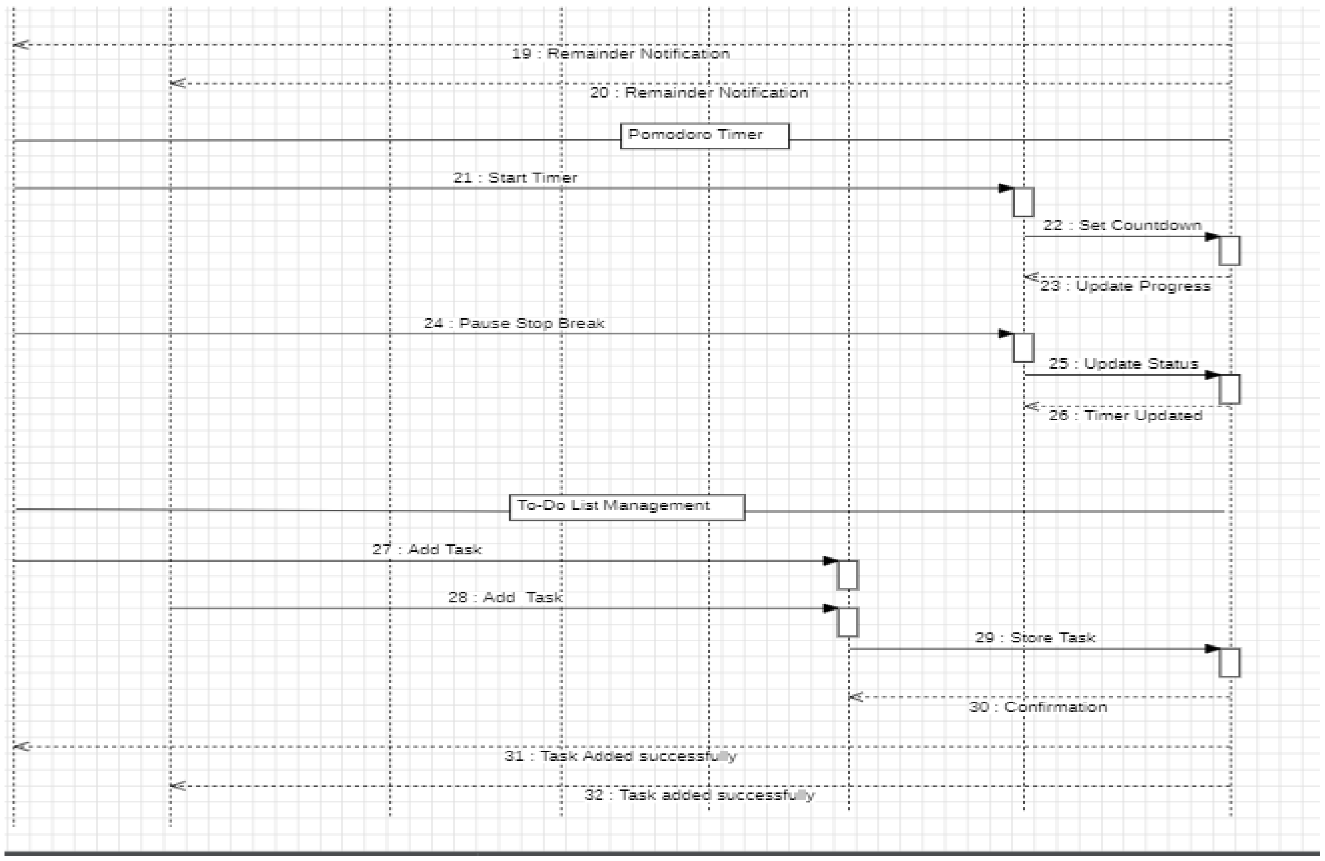


Figure 4.2 Sequence Diagram

A **Sequence Diagram** in the Unified Modeling Language (UML) visually represents the interaction between various components of the **Lesson-Plan Creator** system. It illustrates how objects communicate with each other over time, ensuring the seamless execution of core functionalities. The **sequence diagram** for this project demonstrates the flow of interactions between the **user, system, and external services**, such as the **Google Calendar API** for scheduling events and reminders.

The diagram follows a **top-to-bottom flow**, where user actions initiate requests, and the system processes these actions step by step. For instance, when a student adds a new task, the system records it in the **task manager**, updates the **progress-tracking graph**, and schedules necessary reminders. Similarly, in the **Pomodoro Timer**, user selections trigger the timer countdown, followed by break reminders.

By organizing interactions into lifelines and messages, the **sequence diagram** provides a structured view of the application's workflow. This ensures an **efficient, real-time response mechanism**, reducing delays and improving user experience. The design follows an **object-oriented approach**, making it **scalable and maintainable** for future enhancements.

4.1.3 Statechart Diagram

An **Activity Diagram** is a crucial UML diagram that illustrates the **workflow** of a system, representing how different components interact dynamically. It functions like a **flowchart**, showing the **sequence of operations** in the **Edutimely** project. Each activity signifies a specific operation, and control flows **sequentially**, in **branches**, or in **parallel** depending on the system's logic.

In the **Edutimely** application, the **Activity Diagram** maps out the interactions for both **Students** and **Administrators**:

- **Students** can log in, access the **Task Management System**, set deadlines, track progress, utilize the **Dynamic Calendar**, use the **Pomodoro Timer**, plan for exams, receive **notifications**, and synchronize data.
- **Administrators** oversee **user accounts**, configure system settings, **view reports**, and trigger **reminders** for students.
- **Unauthorized users** attempting access are denied.

By incorporating elements like **decision points, parallel operations, and system synchronization**, the **Activity Diagram** ensures smooth workflow management in **Edutimely**, making academic planning **efficient and organized**.

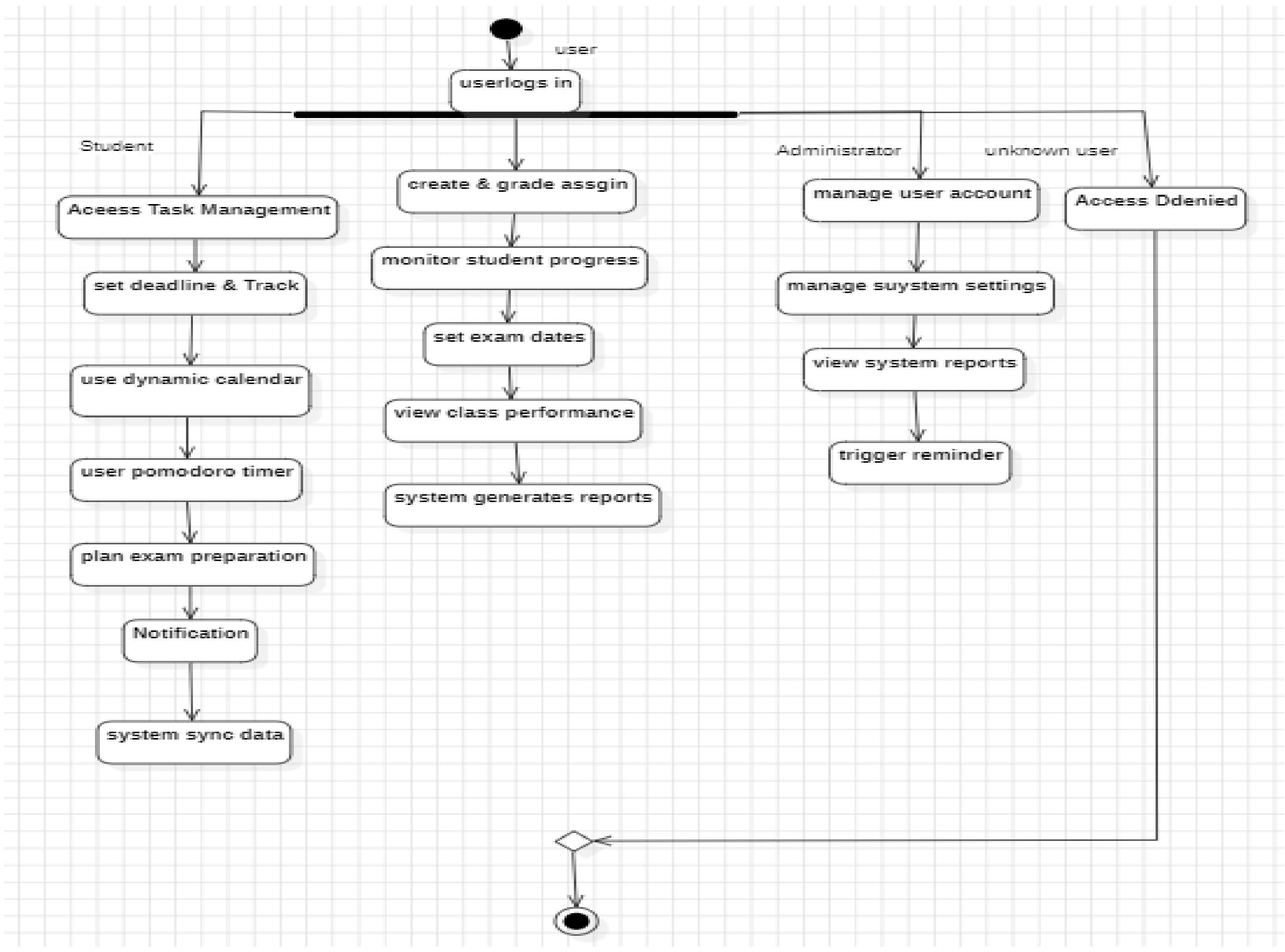
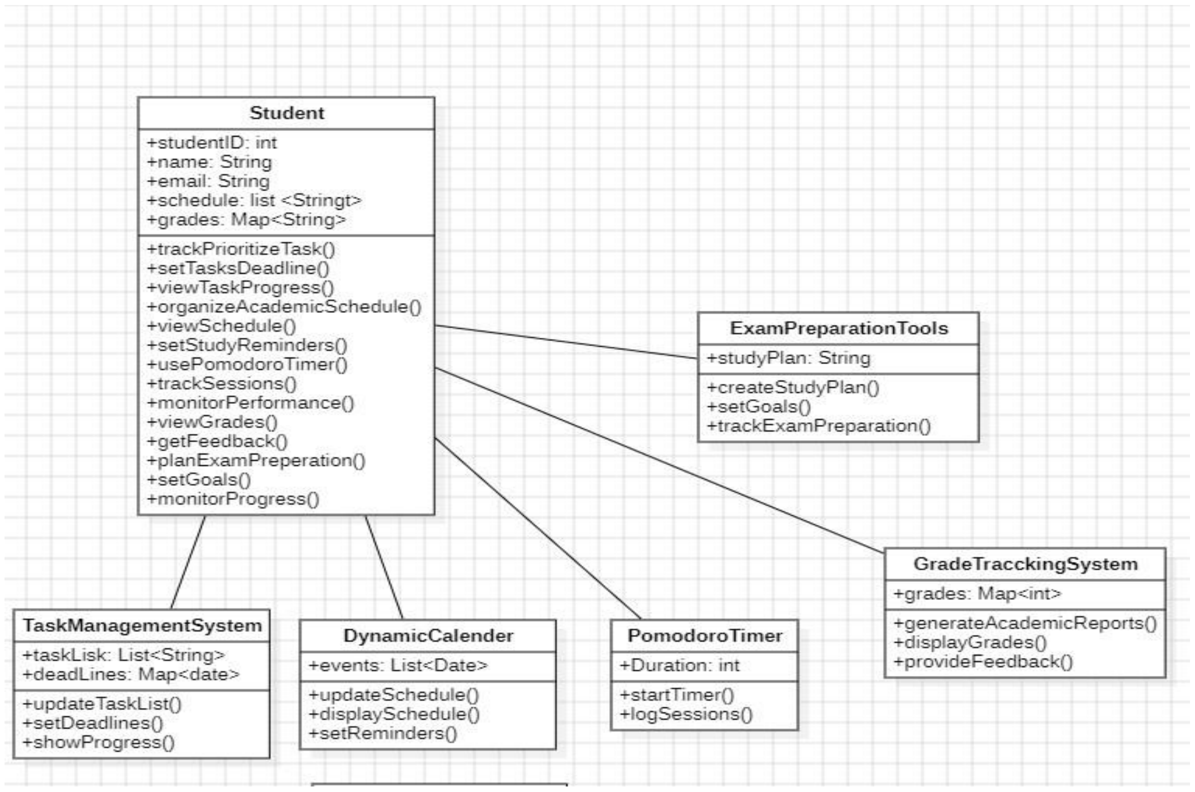


Figure 4.3 **Activity Diagram for Edutimely**

4.1.4 Class Diagram

Edutimely In software engineering, a class diagram in the Unified Modeling Language (UML) is type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. The class diagram is the main building block of object-oriented Modelling. It is used both for general conceptual modelling of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modelling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed.



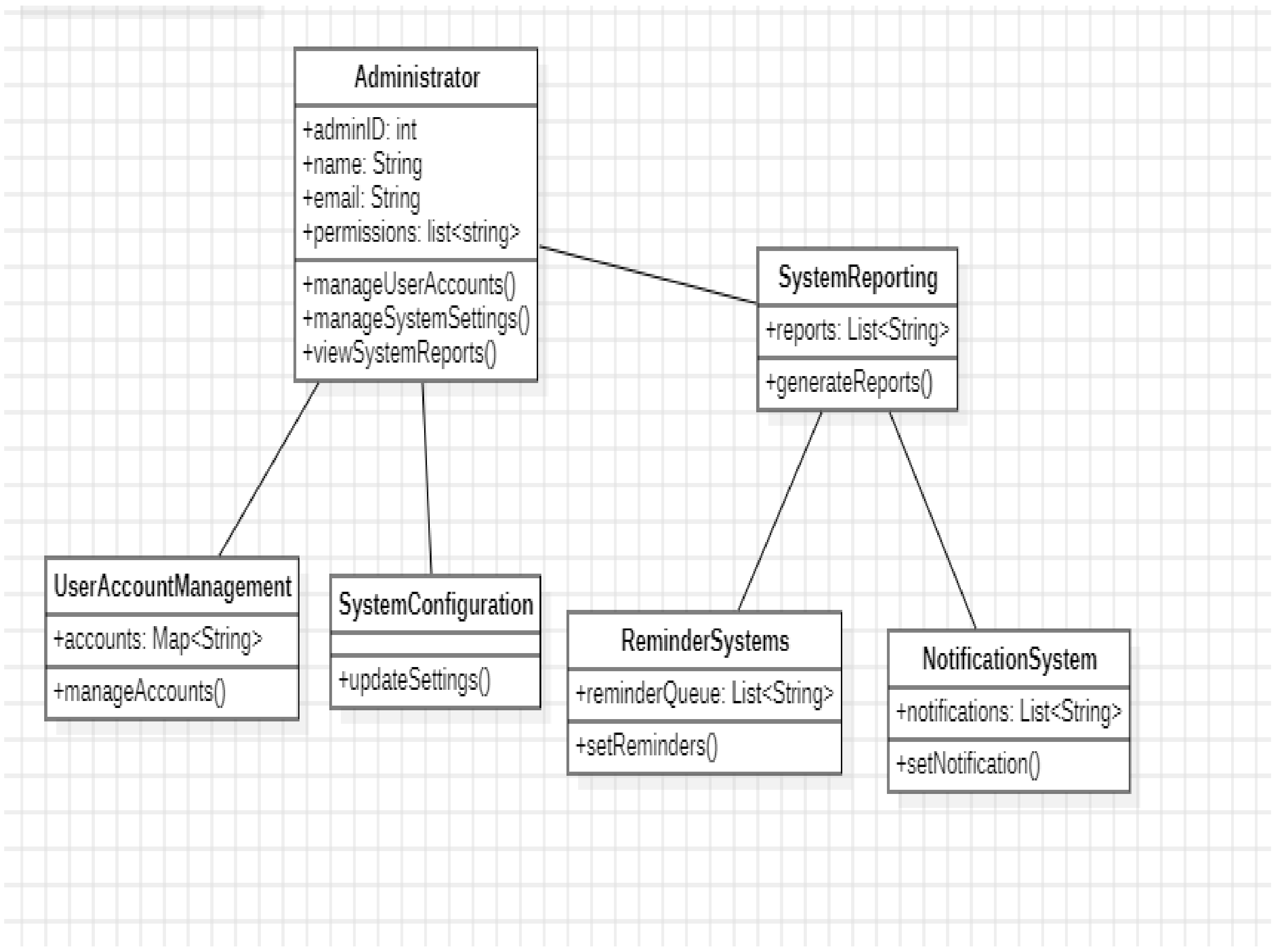


Figure 4.4 Class Diagram for Edutimely

The **Class Diagram** for the **Lesson-Plan Creator** project represents the **structural foundation** of the system by defining its key components, attributes, and relationships. It illustrates the core classes such as **User, TaskManager, Calendar, PomodoroTimer, NotesPicker, GradeTracker, and Progress Tracker**, each encapsulating specific functionalities. The **User class** connects with multiple modules, ensuring seamless interaction between features like **task creation, schedule management, and progress tracking**. Associations between classes highlight how data flows within the system, such as the **TaskManager class storing tasks and updating progress in real-time**, or the **Calendar class integrating with Google Calendar API for event scheduling and reminders**. This **object-oriented design** ensures modularity, scalability, and maintainability, making it easier to extend and improve functionalities as needed.

4.1.5 Deployment Diagram

The **Deployment Diagram** of **Edutimely** represents the execution architecture by mapping software components onto hardware and software environments. It illustrates how the **frontend (Vite + React)** interacts with the **backend server** to manage student schedules, task tracking, and academic progress. The **database server** stores essential data, while APIs facilitate smooth communication between system components. **User devices**, such as laptops and mobile phones, access the platform through a web-based interface. **Administrators** manage accounts, generate reports, and configure system settings on secure **server nodes**. The system also integrates with external services like **Google Calendar API** for scheduling and reminders. This deployment setup ensures a scalable, efficient, and well-structured execution of the **Edutimely** platform.

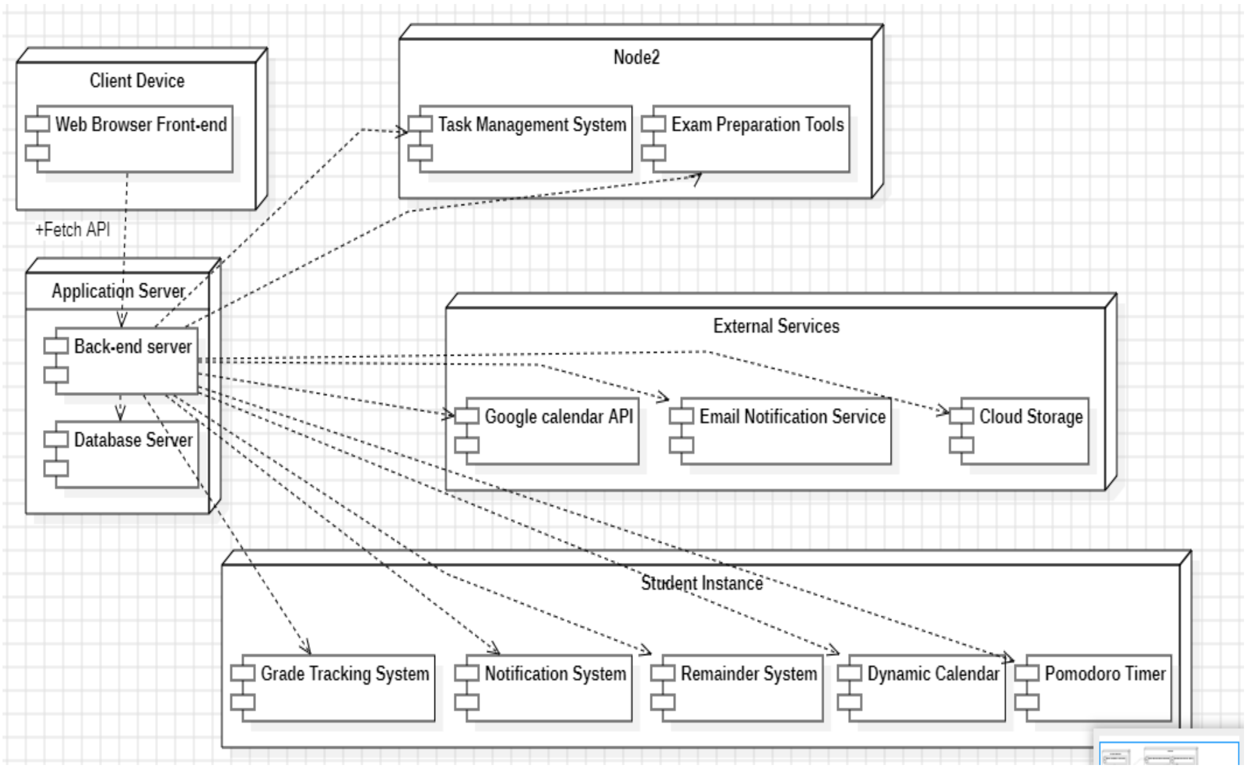


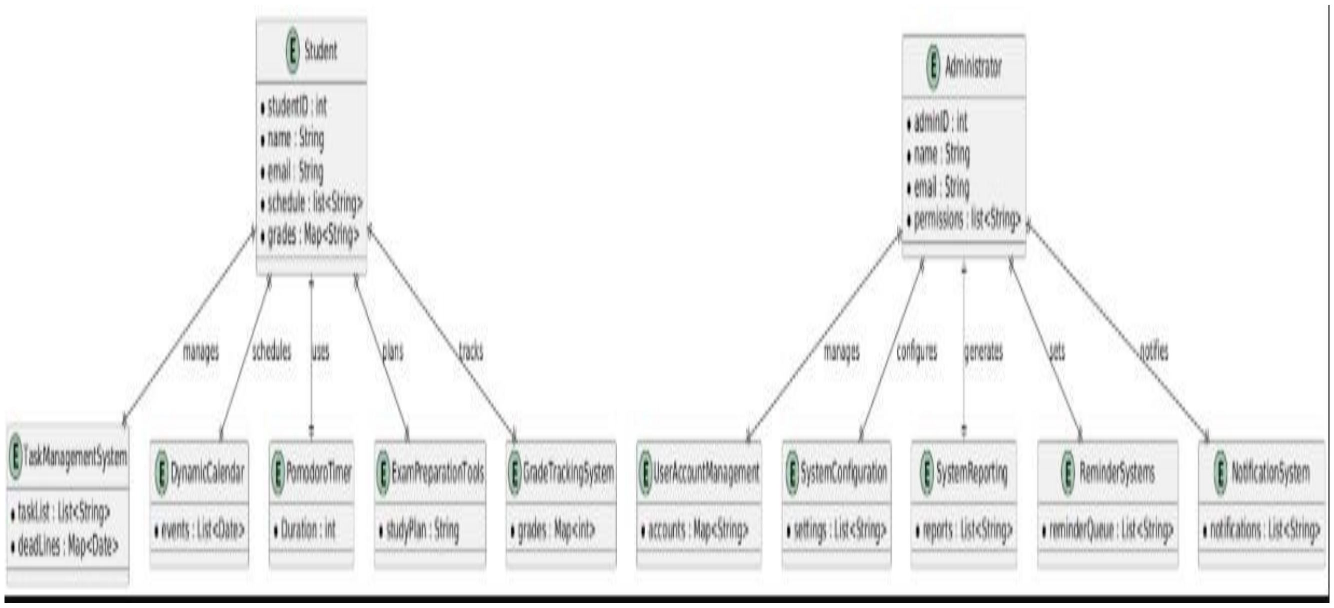
Figure 4.5 Deployment Diagram of the system

4.1.5 ER Diagram

In the **Edutimely** project, the **ER Diagram** represents the structured design of the database by defining entities, their attributes, and relationships. The **Student** entity plays a central role, storing information like student ID, name, email, schedule, and grades. It maintains relationships with key systems such as **Task Management System**, **Dynamic Calendar**, **Pomodoro Timer**, **Exam Preparation Tools**, and

Grade Tracking System, enabling students to efficiently track tasks, set deadlines, plan exams, and monitor academic progress. The **Administrator** entity is responsible for managing user accounts, configuring system settings, and generating reports. It interacts with **User Account Management**, **System Reporting**, **Reminder Systems**, and **Notification System**, ensuring smooth operations and timely updates. The **ER Diagram** helps visualize these relationships, optimizing database design for seamless data retrieval and management in **Edutimely**.

Figure:4.6 ER-diagram



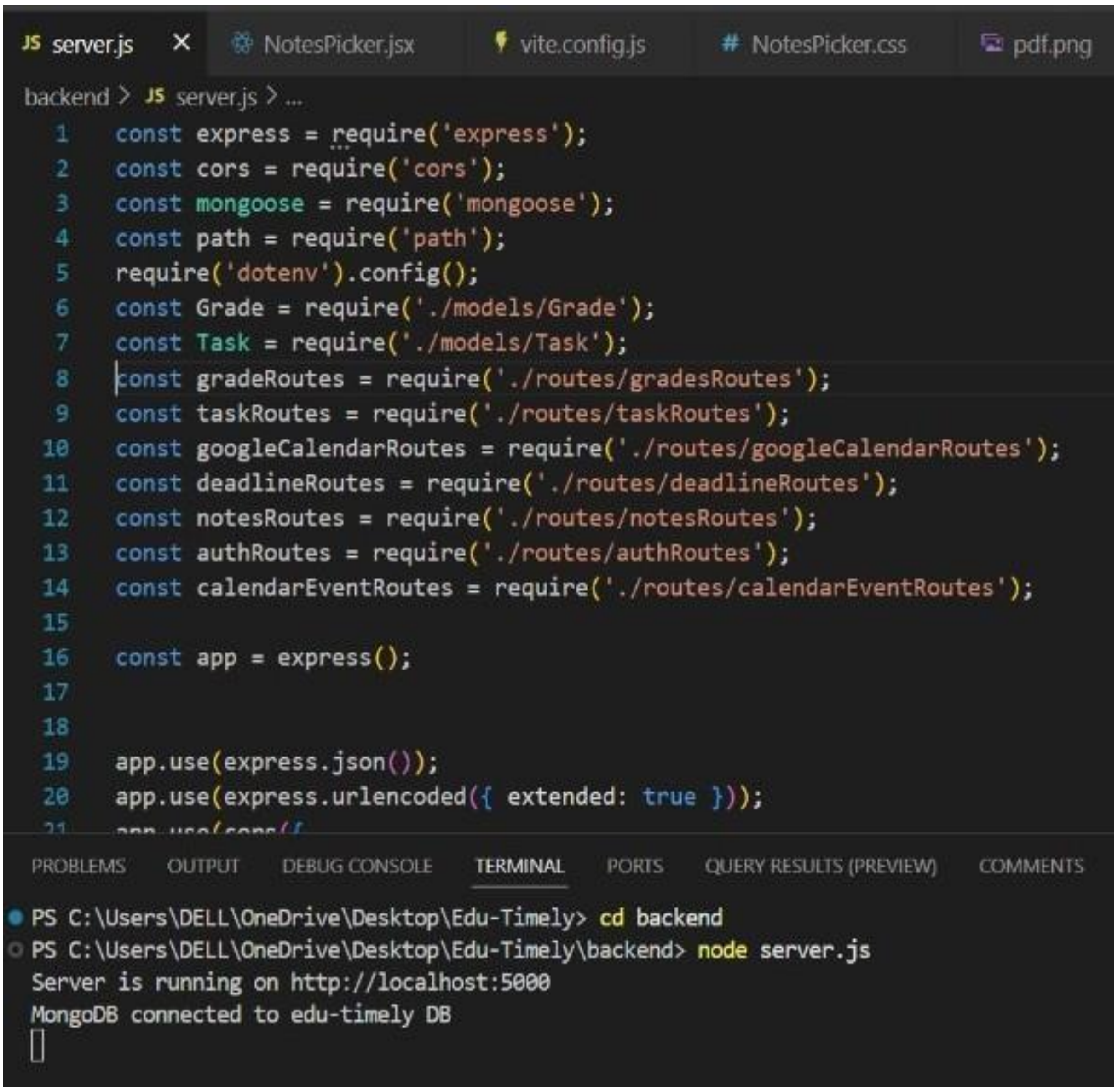
CHAPTER - 5

IMPLEMENTATION

5 IMPLEMENTATION

5.1 Sample Code

5.1.1 Code for server.js

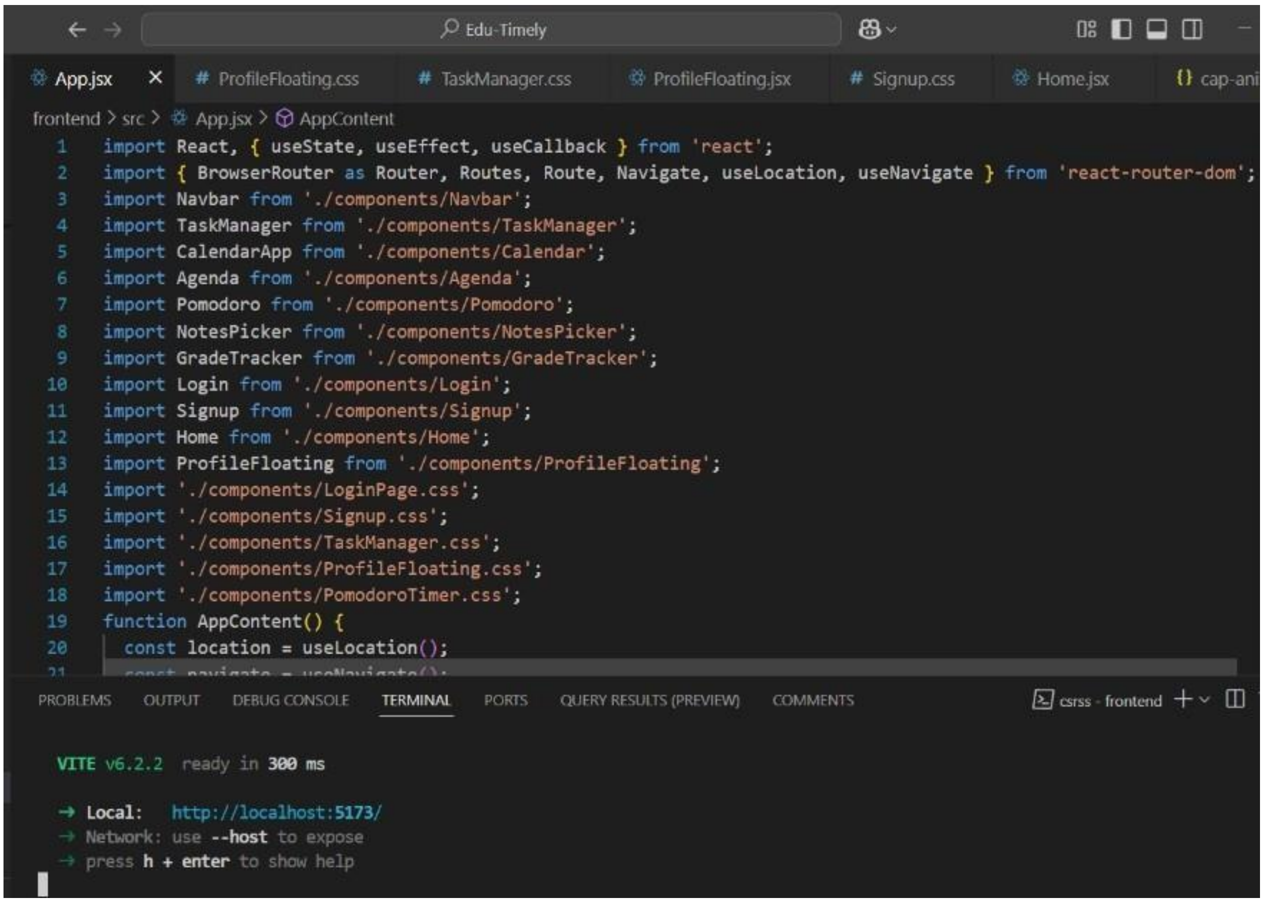


```javascript
const express = require('express');
const cors = require('cors');
const mongoose = require('mongoose');
const path = require('path');
require('dotenv').config();
const Grade = require('./models/Grade');
const Task = require('./models/Task');
const gradeRoutes = require('./routes/gradesRoutes');
const taskRoutes = require('./routes/taskRoutes');
const googleCalendarRoutes = require('./routes/googleCalendarRoutes');
const deadlineRoutes = require('./routes/deadlineRoutes');
const notesRoutes = require('./routes/notesRoutes');
const authRoutes = require('./routes/authRoutes');
const calendarEventRoutes = require('./routes/calendarEventRoutes');

const app = express();


app.use(express.json());
app.use(express.urlencoded({ extended: true }));
```

```
PS C:\Users\DELL\OneDrive\Desktop\Edu-Timely> cd backend
PS C:\Users\DELL\OneDrive\Desktop\Edu-Timely\backend> node server.js
Server is running on http://localhost:5000
MongoDB connected to edu-timely DB
```

Figure:5.1 code for server.js

5.1.2 Code for App.jsx

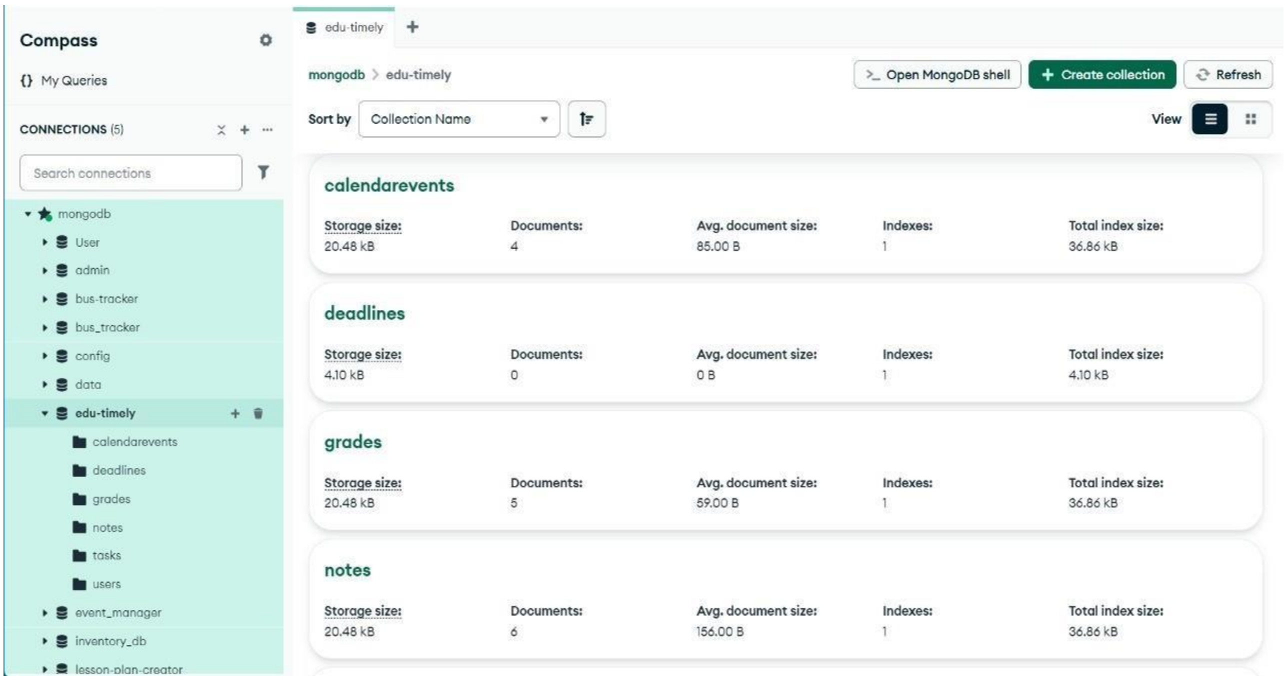


```
import React, { useState, useEffect, useCallback } from 'react';
import { BrowserRouter as Router, Routes, Route, Navigate, useLocation, useNavigate } from 'react-router-dom';
import Navbar from './components/Navbar';
import TaskManager from './components/TaskManager';
import CalendarApp from './components/Calendar';
import Agenda from './components/Agenda';
import Pomodoro from './components/Pomodoro';
import NotesPicker from './components/NotesPicker';
import GradeTracker from './components/GradeTracker';
import Login from './components/Login';
import Signup from './components/Signup';
import Home from './components/Home';
import ProfileFloating from './components/ProfileFloating';
import './components/LoginPage.css';
import './components/Signup.css';
import './components/TaskManager.css';
import './components/ProfileFloating.css';
import './components/PomodoroTimer.css';
function AppContent() {
  const location = useLocation();
```

```
VITE v6.2.2  ready in 300 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```

Figure:5.2 code for app.jsx

5.1.3 MongoDb Schema Setup

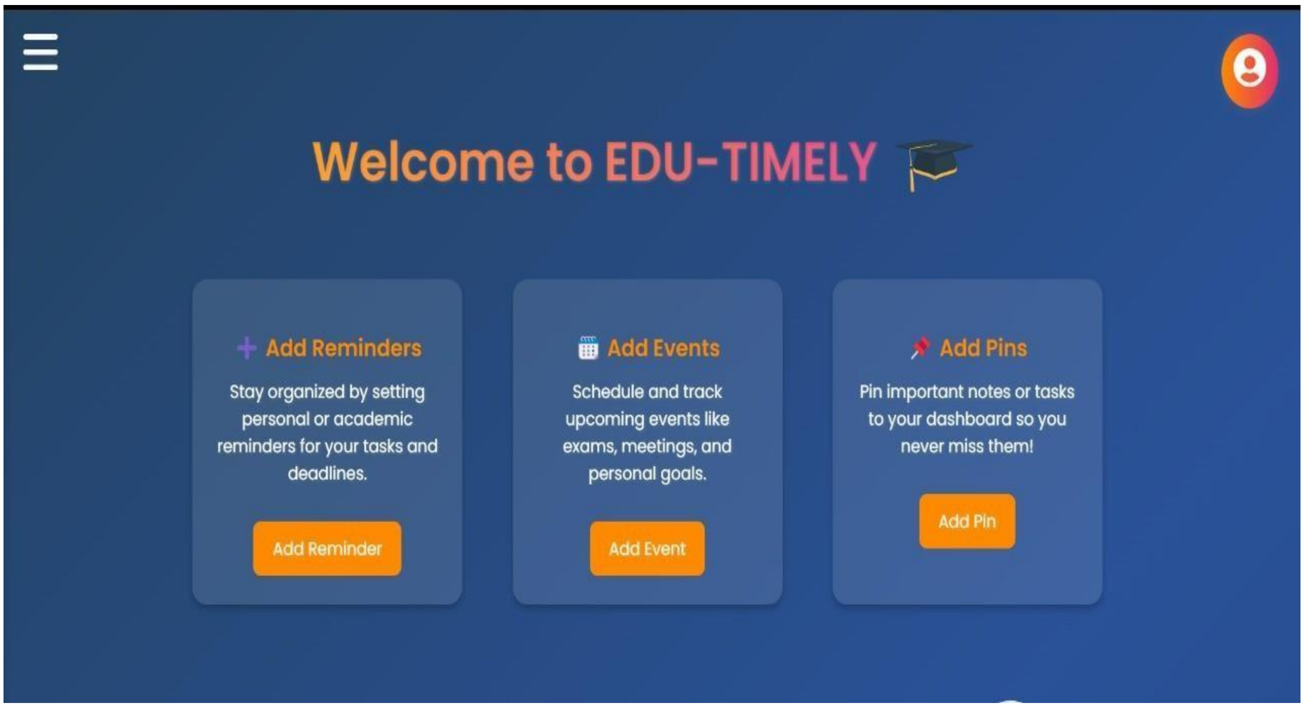


```
calendarevents
Storage size:   Documents:   Avg. document size:   Indexes:   Total index size:
20.48 kB        4            85.00 B               1          36.86 kB

deadlines
Storage size:   Documents:   Avg. document size:   Indexes:   Total index size:
4.10 kB         0            0 B                   1          4.10 kB

grades
Storage size:   Documents:   Avg. document size:   Indexes:   Total index size:
20.48 kB        5            59.00 B               1          36.86 kB

notes
Storage size:   Documents:   Avg. document size:   Indexes:   Total index size:
20.48 kB        6            156.00 B              1          36.86 kB
```

Figure:5.3 DataBase Schema

5.2 Screen Output(Frontend)

5.2.1 Home Screen


Welcome to EDU-TIMELY

Add Reminders

Stay organized by setting personal or academic reminders for your tasks and deadlines.

Add Events

Schedule and track upcoming events like exams, meetings, and personal goals.

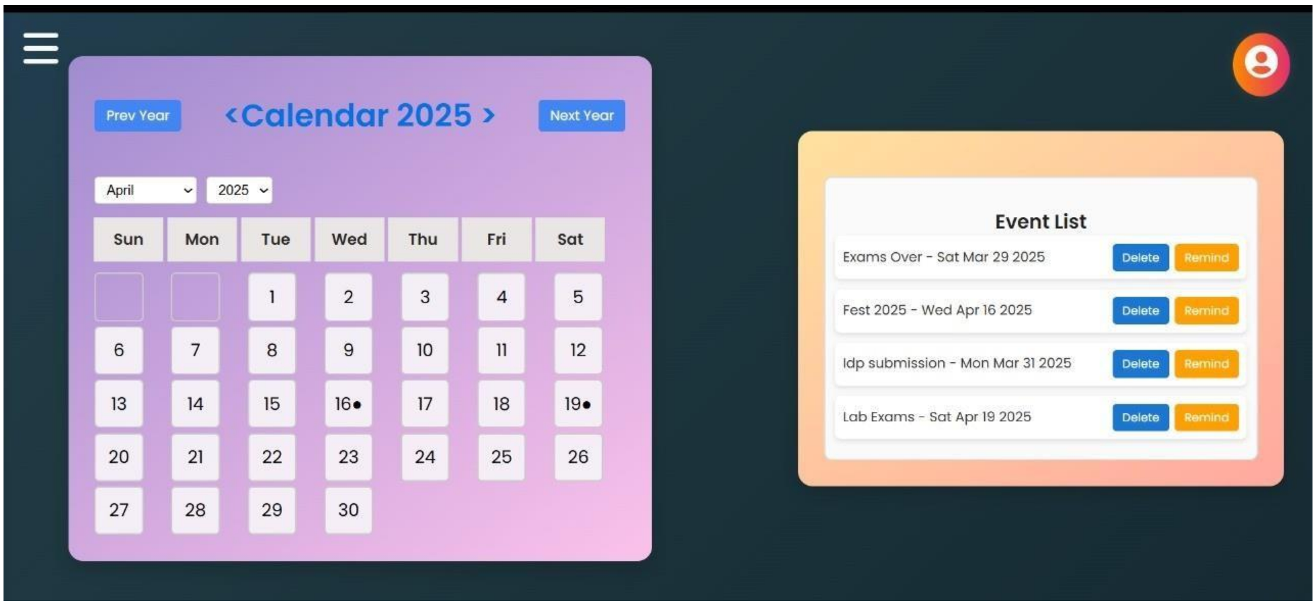
Add Pins

Pin important notes or tasks to your dashboard so you never miss them!

Figure 5.4: Home Screen Of Project.

Description: After Lunching of the project this will be visible on our screen.

5.2.2 Calendar Page



Sun	Mon	Tue	Wed	Thu	Fri	Sat
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16●	17	18	19●
20	21	22	23	24	25	26
27	28	29	30			

Event List

Exams Over - Sat Mar 29 2025

Fest 2025 - Wed Apr 16 2025

Idp submission - Mon Mar 31 2025

Lab Exams - Sat Apr 19 2025

Figure: 5.5 Calendar Route

Description: A Calendar page where user can marks their task for specific date.

5.2.3 Agenda Page

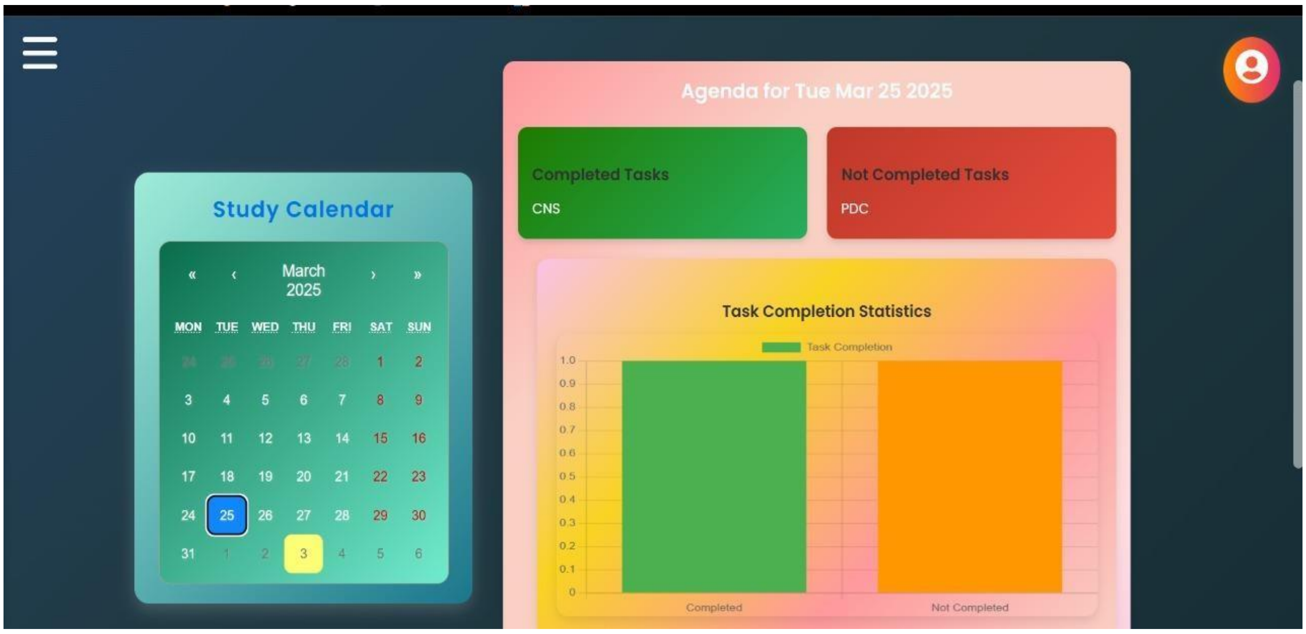


Figure:5.6 Agenda Page

Description: A Agenda page where user can mark their Agenda for particular date.

5.2.4 Pomodoro Timer

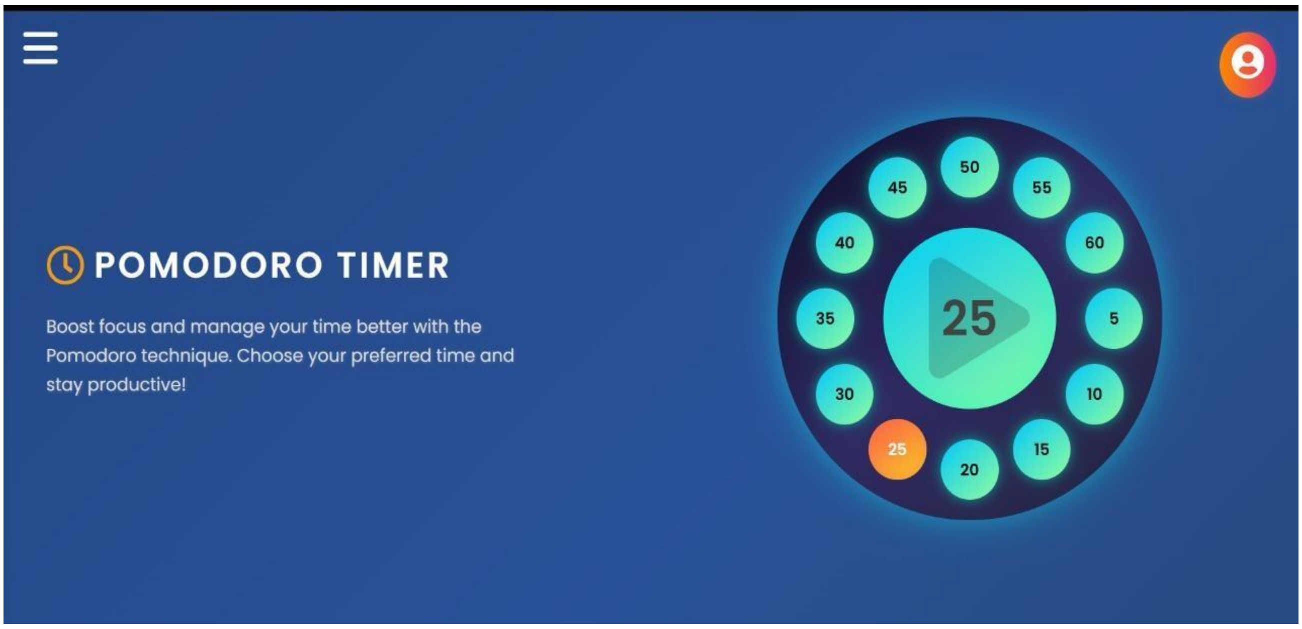

POMODORO TIMER
Boost focus and manage your time better with the Pomodoro technique. Choose your preferred time and stay productive!
25

Figure:5.7 Pomodoro Timer

Description: A pomodoro timer where user can set timer for their specific tasks.

5.2.5 Notes Picker

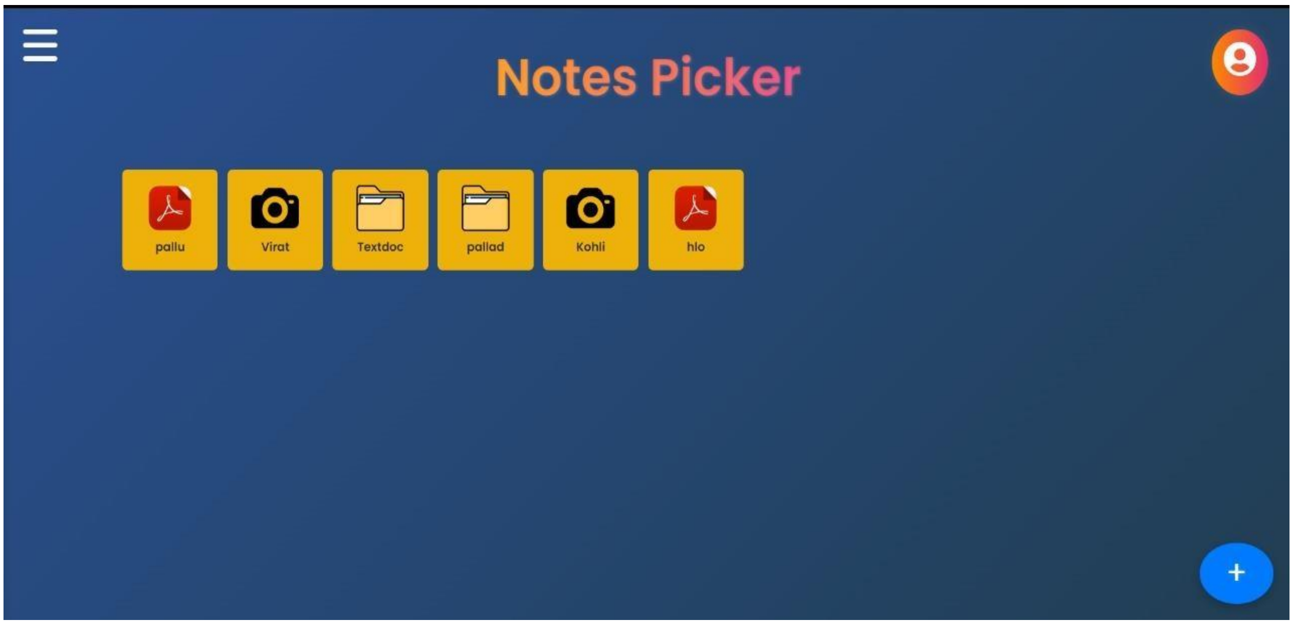


Figure:5.8 Notes Picker

Description: A Notes Picker Page where user can pick note for them.

5.2.6 Task Manager

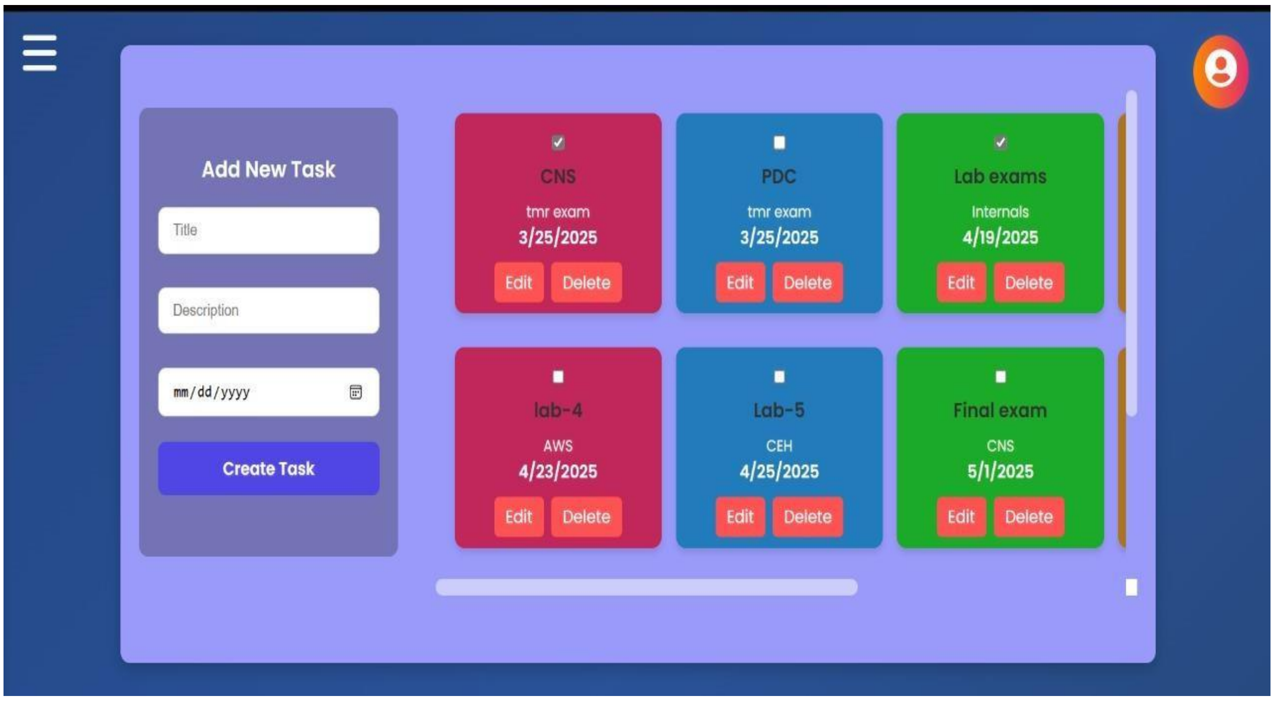


Figure:5.9 Task Manger

Description: A place where user can add their task and update them as well.

5.2.7 Sign Up Page

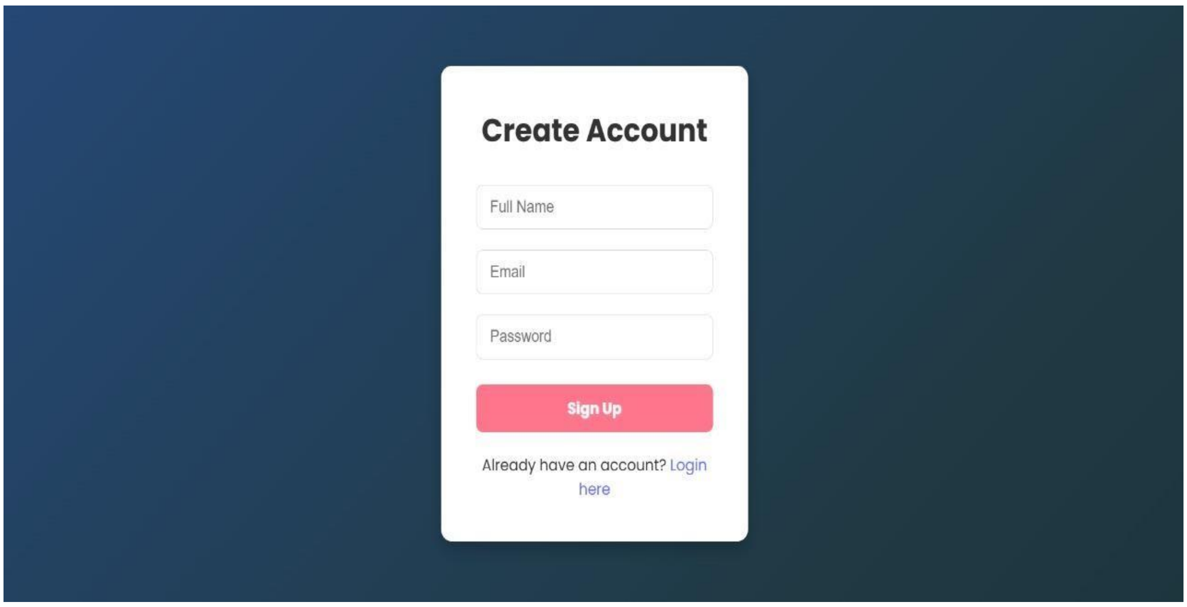


Figure:5.10 Sign up Page

Description: User can fill the form and can create a account for the Application.

5.2.8 Login Page

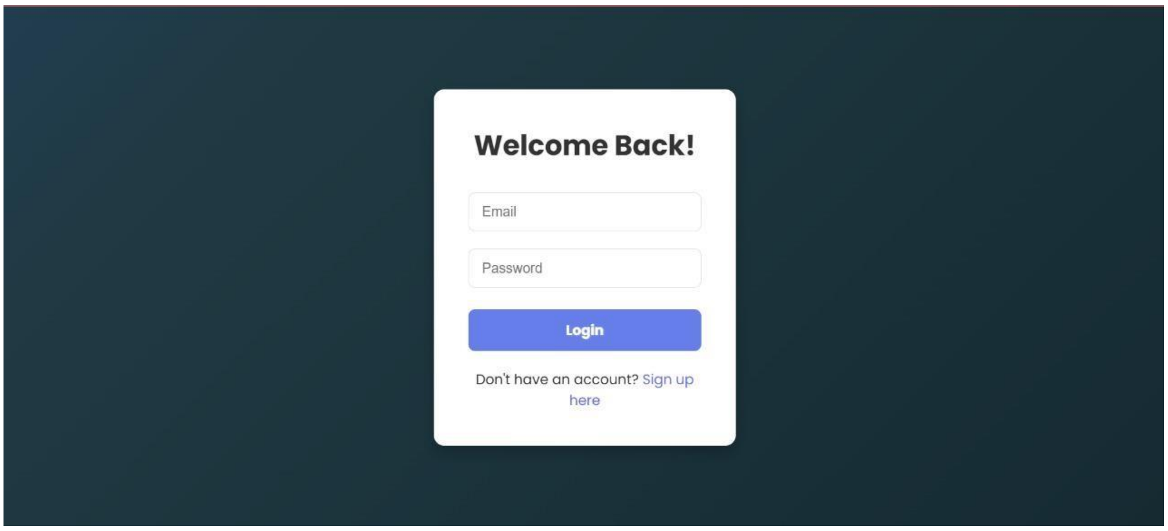


Figure:5.11 Login page.

Description: User can fill the asked field to log into the application.

CHAPTER - 6

TESTING

6 TESTING

6.1 Software Testing

Software testing is the process of validating and verifying that a software application meets the technical requirements which are involved in its design and development. It is also used to uncover any defects/bugs that exist in the application. It assures the quality of the software. There are many types of testing software viz., manual testing, unit testing, black box testing, performance testing, stress testing, regression testing, white box testing etc. Among these performance testing and load testing are the most important one for an android application and next sections deal with some of these types.

6.2 Black box Testing

Black box testing treats the software as a "black box"—without any knowledge of internal implementation. Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.

6.3 White box Testing

White box testing is when the tester has access to the internal data structures and algorithms including the code that implement these.

6.4 Performance Testing

Performance testing is executed to determine how fast a system or sub-system performs under a particular workload. It can also serve to validate and verify other quality attributes of the system such as scalability, reliability and resource usage.

6.5 Load Testing

Load testing is primarily concerned with testing that can continue to operate under specific load, whether that is large quantities of data or a large number of users.

6.6 Manual Testing

Manual Testing is the process of manually testing software for defects. Functionality of this application is manually tested to ensure the correctness. Few examples of test case for Manual Testing are discussed later in this chapter.

Test Case 1	
Test Case Name	Inserting New Task
Description	If all fields given.
Output	New task added

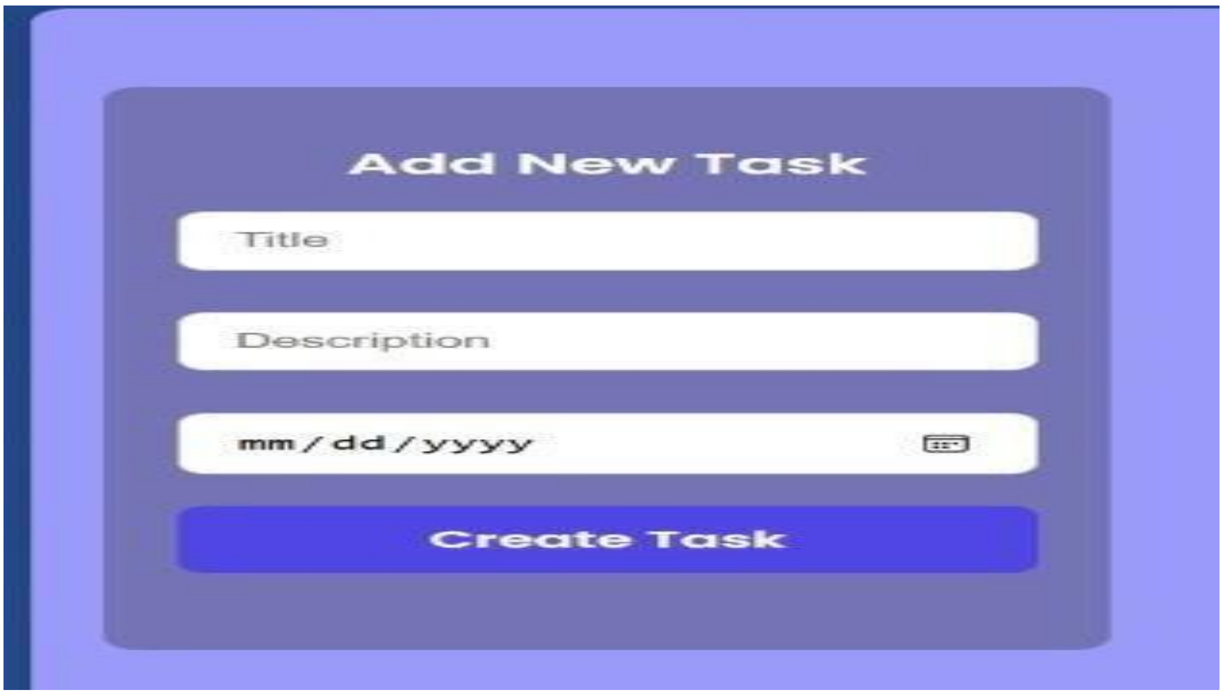


Figure:6.1 New task adding field.

Test Case 2	
Test Case Name	Updating the task status
Description	If all fields given.
Output	task status updated

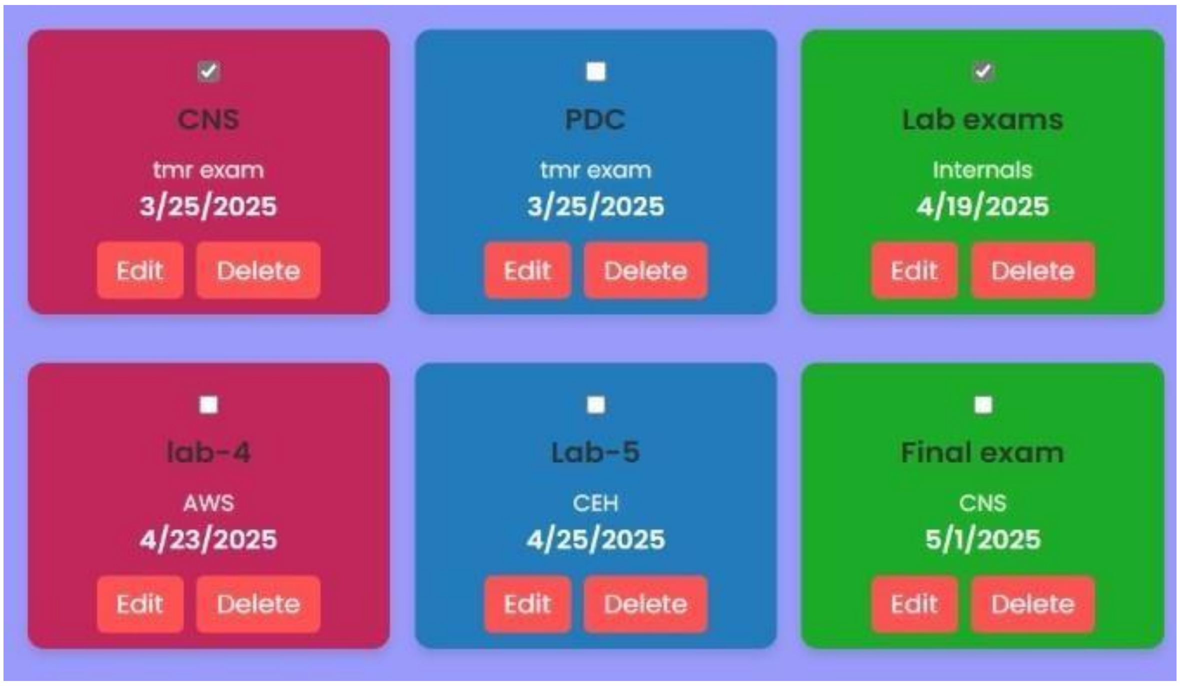


Figure:6.2 Task Updated.

CHAPTER - 7

RESULTS &CHALLENGES

7 RESULTS AND CHALLENGES

7.1 Results

The **Edutimely** platform successfully enhances the academic experience for students and teachers by integrating essential productivity tools. The system enables users to efficiently **schedule events, plan their day, track CGPA with graphical analysis, and improve focus using a Pomodoro timer**.

For students, **Edutimely** provides an intuitive **calendar-based scheduling system**, allowing them to **plan tasks, set reminders, and monitor their academic progress**. The **CGPA tracking feature** offers insightful visual analytics, helping students assess their performance effectively.

Teachers benefit from a structured approach to **managing schedules and coordinating with students**. The platform ensures seamless user interaction with a **modern UI and real-time updates**.

By leveraging the **MERN stack**, **Edutimely** ensures a **scalable, efficient, and user-friendly experience**, helping students and teachers **stay organized, productive, and academically motivated**.

7.2 Challenges

Developing **Edutimely** came with several challenges that were addressed throughout the project:

- **Understanding user requirements**: Designing a system that meets the needs of both students and teachers required extensive research and feedback collection.
- **Creating an intuitive UI/UX**: Ensuring a user-friendly experience while integrating multiple features like scheduling, CGPA tracking, and task management was complex.
- **Data synchronization and real-time updates**: Managing seamless synchronization of event schedules, reminders, and performance analytics across users posed technical difficulties.
- **Integrating multiple functionalities**: Merging **calendar-based scheduling, Pomodoro timers, and graphical CGPA tracking** into a single cohesive platform required careful system architecture planning.

- **Learning new technologies**: Working with the **MERN stack**, implementing **graphical data visualization**, and optimizing **database performance** involved continuous learning and problem-solving.

Despite these challenges, **Edutimely** was successfully built as a **student and teacher-friendly** academic management platform, improving productivity and organization.

CHAPTER - 8

CONCLUSIONS & FUTURE WORK

8 CONCLUSION

8.1 Conclusions

The application has been designed successfully to meet all the user requirements. I found this project to be far more difficult than I ever anticipated. Without doubt, this has been the most challenging and at the same time rewarding programming project I have undertaken since I started college.

8.2 Scope for future work

The application can further be modified in the following ways:

- Finish the implementation of syncing the customer visit details to server.
- Additional features like tracking the order can be implemented.
- Introduce a feature to call contacts directly from within the app.
- Integrate with built-in Google Maps API, SMS, and Email.
- Fix all existing bugs

8.3 Limitations

Edutimely currently requires manual input for tasks and grades, and lacks real-time collaboration or full academic platform integration. Advanced AI suggestions and automated scheduling features are planned for future updates.

BIBLIOGRAPHY

[1] Oracle corporation and its affiliates “MySQL open source database”

with no author

[online] available at: https://www.mysql.com/

[2] Sonoo Jaiswal “An Overview on Java”

[online] available at: https://www.javatpoint.com/

[3] JMockit “An automated testing toolkit for java” with no author

[online] available at: http://jmockit.github.io/tutorial/Introduction.html

[4] M. Vaqqas, September 23, 2014 “RESTful web Services: A Tutorial”

[online] available at http://www.drdobbs.com/web-development/tutorial

[5] The MIT License “Angular one framework” with no author

[online] available at: https://angularjs.org/

[6] DaolrevoLtd. Asim “Jasmine and Karma”

[online] available at: https://codecraft.tv/courses/angular/unit-testing/jasmine-and-karma/

[7] Software Freedom Conservancy “git local branching on the cheap” with no author

[online] available at: https://git-scm.com/