

MENTOR MENTEE SYSTEM

A Project Report

Submitted for the partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY – INFORMATION TECHNOLOGY

Submitted by

Ayushi Mishra: 2216818

Himani: 2216830

Isha: 2216831

Swadha Sri: 2216913

Under the supervision of

Dr. Deepak Kumar



Department of Computer Science

Banasthali Vidyapith

Session: 2024-25

Certificate

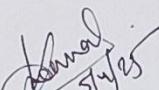
Certified that following students have carried out the project work titled “Mentor Mentee System” from 1st August 2024 to 31st March 2025 for the award of the Bachelor of Technology from Banasthali Vidyapith under my supervision. The report embodies result of original work and studies carried out by students themselves and the contents of the report do not form the basis for the award of any other degree to the candidates or to anybody else.

Member 1: 2216913 Swadha Sri (IT)

Member 2: 2216818 Ayushi Mishra (IT)

Member 3: 2216830 Himani (IT)

Member 4: 2216831 Isha (IT)


Dr. Deepak Kumar

Designation: Assistant Professor

Place: Banasthali Vidyapith

Date: 05/04/2025

ABSTRACT

Problem Statement:

In academic project management, fragmented communication, coordination issues, limited visibility into project progress, manual administrative tasks, and ineffective feedback mechanisms hinder efficient collaboration among mentees, mentors, and coordinators. These challenges result in missed deadlines, misunderstandings, and suboptimal project outcomes. A solution is needed to streamline communication, enhance coordination, automate administrative tasks, and facilitate effective feedback mechanisms to improve overall project management efficiency.

Problem Solution:

The Mentor Mentee System presented in this report is designed to facilitate efficient collaboration and communication among mentees, mentors, and coordinators within an academic environment. This, a web-based platform offers a user-friendly interface to access various features tailored to their roles.

For mentees, the system enables them to create teams, search for mentors based on their expertise, and add team members by entering their details. Once teams are formed, mentees can input project details and upload necessary documents, allowing them to monitor project progress and track deadlines.

Mentors have the ability to view teams under their supervision, and monitor their progress. They can also allocate, view, and update marks, providing valuable feedback to mentees. Additionally, mentors can communicate with their teams effectively through the platform.

Coordinators can log in to oversee the overall progress of teams and review marks allotted by mentors. They can also make announcements regarding schedules, venues, and deadlines, ensuring smooth coordination and communication across all stakeholders.

Overall, this project management system serves as a comprehensive platform for mentees, mentors, and coordinators to collaborate effectively, streamline project management processes, and enhance communication within academic settings.

ACKNOWLEDGEMENT

We would like to extend our heartfelt gratitude to our guide Dr. Deepak Kumar for giving us the opportunity to work under her guidance and for giving us valuable inputs and ideas right from the selection of topics for the project till its successful completion.

We thank Dr. Rajiv Singh (HOD Computer Department) for his ongoing support and encouragement in every aspect. Last but not the least, the entire staff of the department of computer engineering for guiding our thoughts and vision.

The successful completion of our project would not have been possible without the dedicated from all our mentors, family and friends.

AYUSHI MISHRA

HIMANI

ISHA

SWADHA SRI

TABLE OF CONTENTS

S. No.	Title	Page Number
1.	Introduction	06
2.	Requirement Analysis (SRS) 2.1 Requirement specification 2.2 H/w and S/w Requirements 2.3 Feasibility Study 2.4 Product Functions 2.5 Use-case Diagrams	07 07 07 08 09 11
3.	System Design (SDS) 3.1 High-level Design 3.2 ER Diagram/Class Diagrams 3.3 Database Design 3.4 Data flow diagrams/Activity Diagrams\ 3.5 Sequence Diagram/Flowcharts	14 14 15 15 16 21
4.	Coding	26
5.	Testing 5.1 Test Cases	48
6.	User Interfaces	51
7.	Appendices	61
8.	References	62

INTRODUCTION

With the evolving of technologies and trends, innovation and collaboration are vital for academic success, so to serve our coordinators to manage the projects, we propose the development of a “Mentor-Mentee System” that will revolutionize the way projects are planned, executed, and overseen within our college. This system will be designed to simplify the process of managing projects from the start of the processes involving the whole procedure that must be executed. It will help the college coordinators to incorporate easy management of the college’s in-house projects management, reduce their work functions and automate them. It is itself an In-House Project.

PURPOSE

- It provides a common canvas between mentors, mentees and coordinators where they can manage all the aspects of their projects.
- Aims to simplify project planning, task delegation, progress monitoring, and communication.
- It involves the creation of groups, allotment of mentors and verification without the involvement of the coordinator.
- Promotes a collaborative and effective environment for project execution and management.
- An invaluable asset to the college where it would be helpful to the coordinators in managing the project and required functions.

SCOPE

The software product to be produced is the "Mentor-Mentee System", a comprehensive online platform (a website) that is designed to facilitate and automate the processes required in project management for all the users (mentors, mentees and coordinators) of the college.

This system will provide ease of creating, assigning, verifying, and updating the desired requirements needed and will also automate the allocation process of mentor and panel as well as for the coordinator.

It will serve as a multifunctional solution that simplifies the process of project planning and helps the coordinator manage all perspectives of the projects. It will include the formation of groups,

generation of group IDs, allocation of mentors and tasks, progress monitoring, and communication among project group members and mentors.

It will provide regular updates regarding the In-House Projects. It will empower mentees and mentors to collaborate effectively to execute projects, enhancing project management capabilities and increasing communication.

REQUIREMENT ANALYSIS (SRS)

- Requirement Specifications

This software requirement specification is made with the purpose of outlining the software architecture and design of the OS Visual Studio software in detail. The document will provide the mentors and the developers an insight to the system design and a deep understanding of the system.

- Hardware Requirements

A. Server Side

1. Processor – 3.5 GHz Multi - core.
2. RAM – 16 GB
3. HDD - 10 TB

B. User Perspective

1. Processor – 3.0 GHz
2. RAM - 4 GB or more for smooth operation.
3. HDD - 1TB (SSD for faster performance).

C. Developers Perspective

1. Processor – 4.0 GHz Core i5 or i7
2. RAM - 32 GB.
3. HDD – 10 TB (SSD for quick file access and multitasking).

- Software Requirements

A. Server Side

1. Operating System – Linux/Windows/MacOS Big Sur
2. Web Server - Express.js, Nginx
3. Database Server – MongoDB

B. User Perspective

1. Operating System – WINDOWS/ Linux/ MacOS.
2. Browser - Any browser (E.g. Google Chrome/ Microsoft Edge/ Mozilla Firefox).

C. Developers Perspective

1. Operating System – Windows 10
2. Integrated Development Environment – Visual Studio Code, WebStorm, or Atom
3. FRONT END – HTML, CSS, JavaScript, React.js, JSX
4. BACK END – MongoDB, Node.JS, Express, JSON
5. BROWSER – Google Chrome, Microsoft Edge, Mozilla Firefox
6. SERVER – React.js with Express (or Apache Tomcat 8.0.27.0)
7. SCRIPTING LANGUAGE – JavaScript

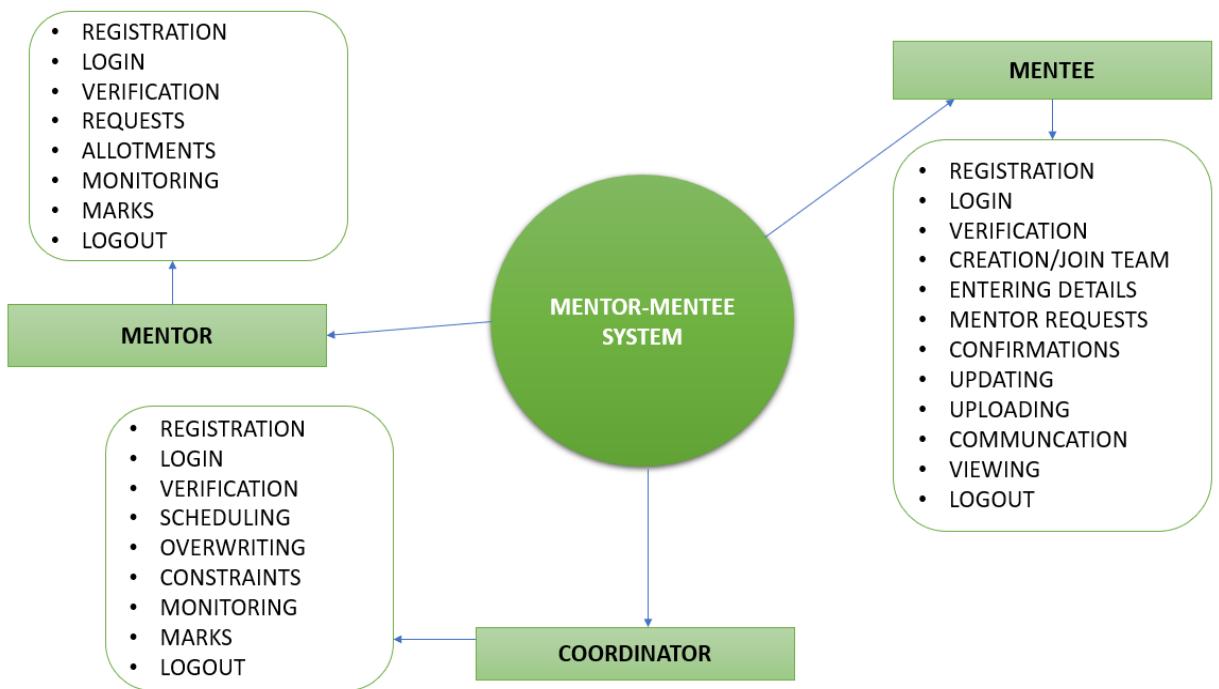
○ FEASIBILITY STUDY

The availability of this software is dependent upon the Internet connection of the user. The user should have a basic browser navigation skill in order to understand all the functionality provided by the system. The authorization mechanism of the application is that if the user enters wrong Id or/and password then user will not be able to login.

This system is reliable enough that it maintains the secure database of the record of the user. The password stored in the database has been encrypted. The unauthorized login is secured via password and Institution id.

The back-end of the application is developed using MONGODB, which again is a platform-independent language and works perfectly well with most platforms like Linux, Unix, Mac OS X, and Windows.

This application follows the modular structure so it will be easy to maintain.

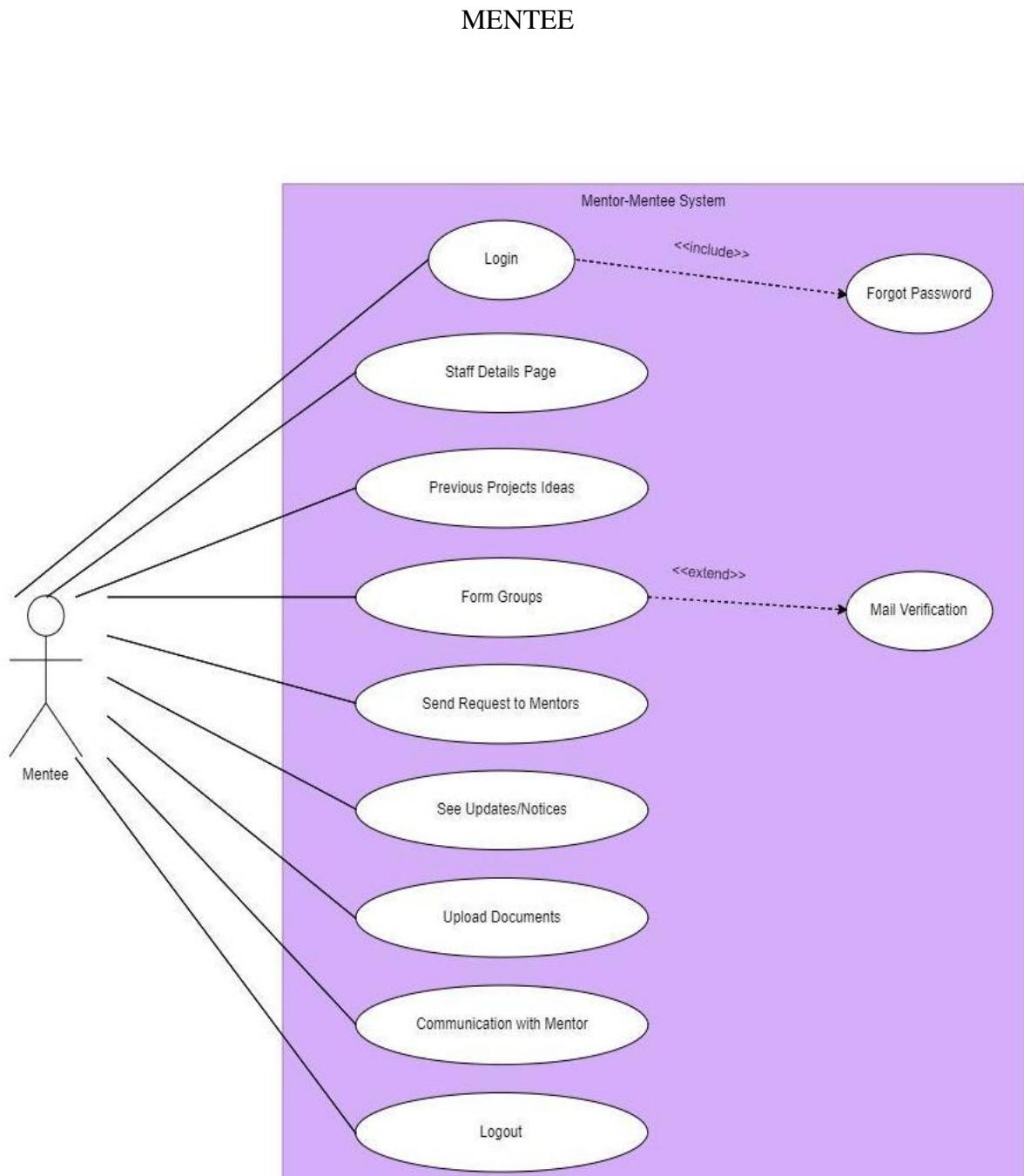


- PRODUCT – FUNCTIONS

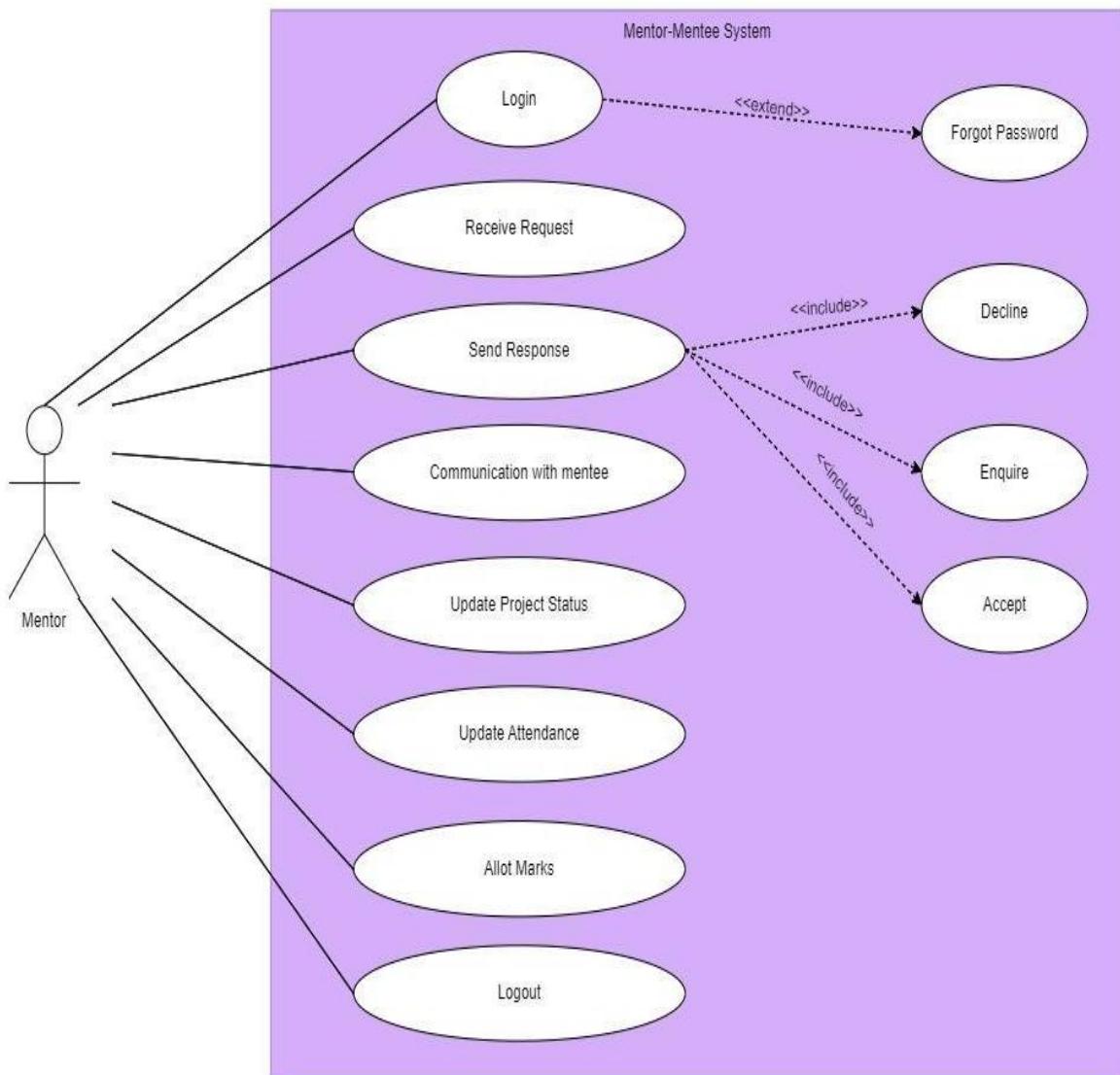
1. **Mentee Information Submission and Group Formation:** Mentees submit their details, and groups are automatically formed based on the data.
2. **Group ID Generation:** Each group is assigned a unique ID for identification and tracking.

3. **Mentor Profiles and Specialization Page:** A page displays mentor profiles, showing their expertise and availability.
4. **Approaching Mentors for Approval:** Mentees can request mentorship by sending approval requests to chosen mentors.
5. **Mentor Allocation Based on Specialization and Constraints:** Mentors are assigned to groups based on their skills and specific project requirements.
6. **Progress and Attendance Tracking:** The system tracks each group's project progress and attendance.
7. **Marks Allocation by Mentor:** Mentors can assign grades to mentees based on their performance.
8. **Data Storage for the Coordinator:** Coordinators can store and access all mentee and project data securely.
9. **Marks Allocation Based on Presentation:** Grades are assigned to mentees based on their project presentations.
10. **Announcement Dashboard:** A dashboard for posting and viewing important announcements and updates.

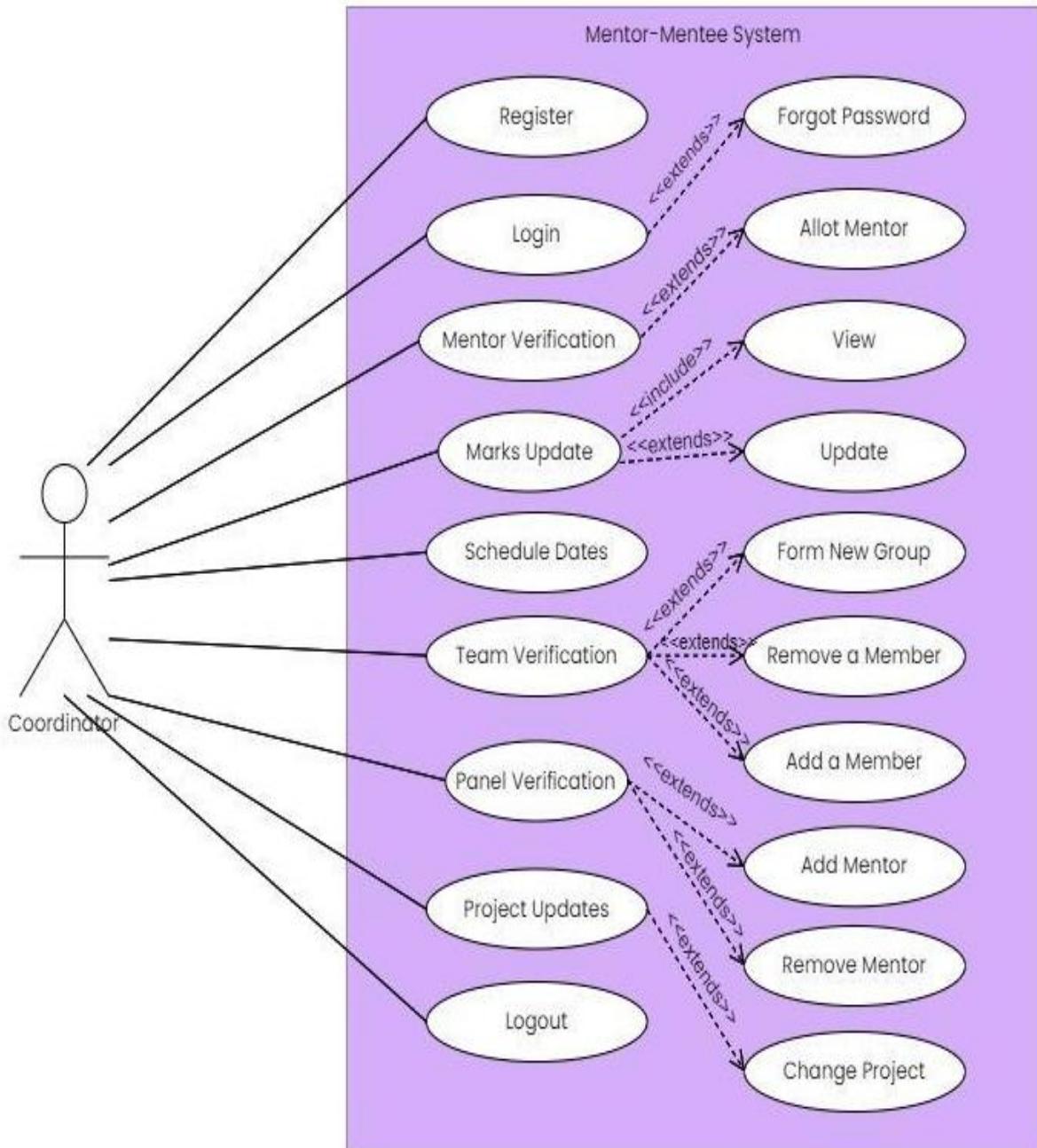
- USE CASE DIAGRAMS



MENTOR

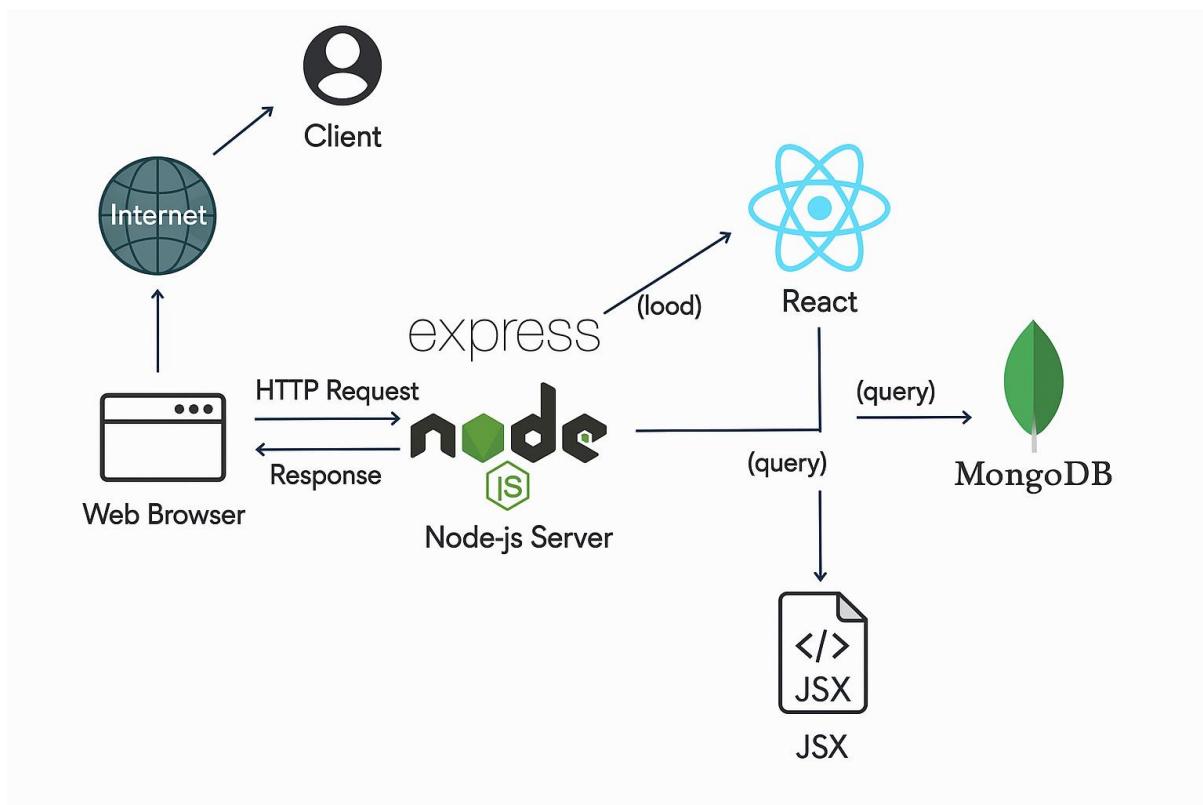


COORDINATOR

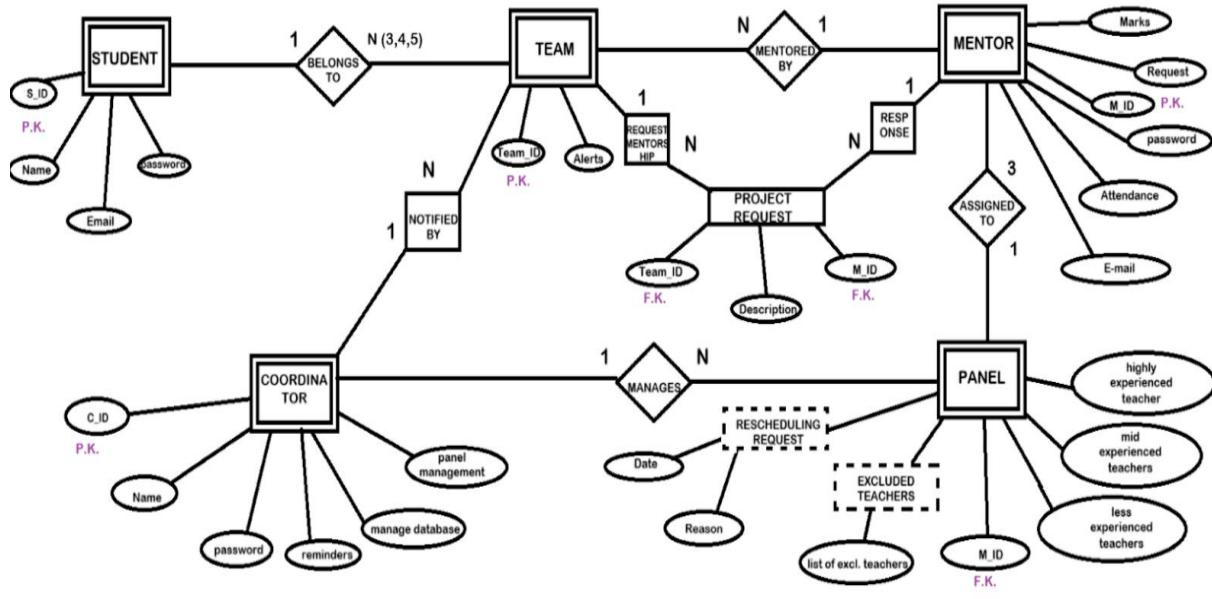


SYSTEM DESIGN (SDS)

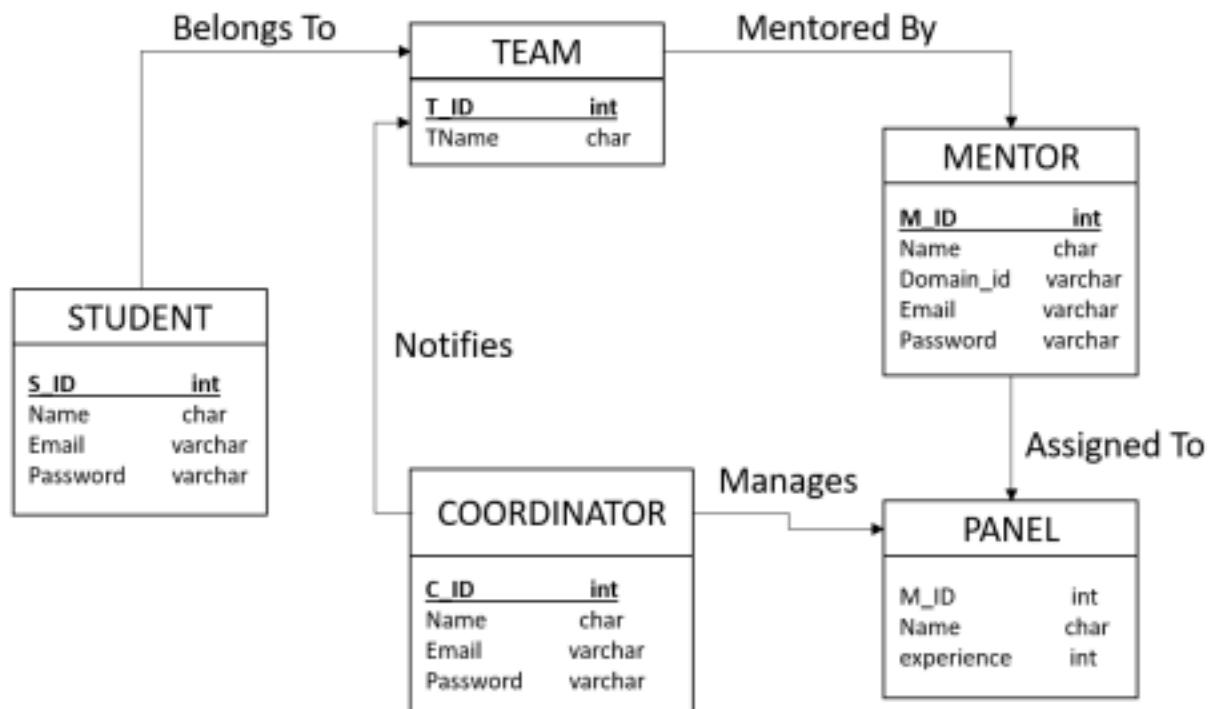
- High Level Design



○ ER Diagram

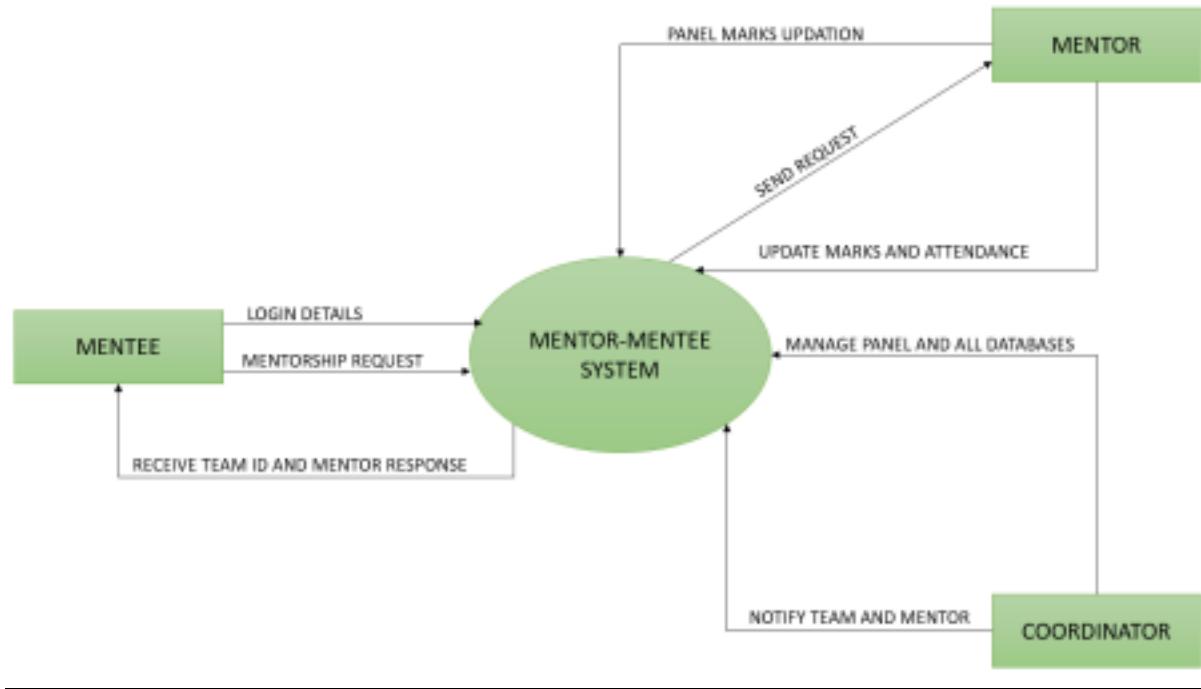


○ Database Design

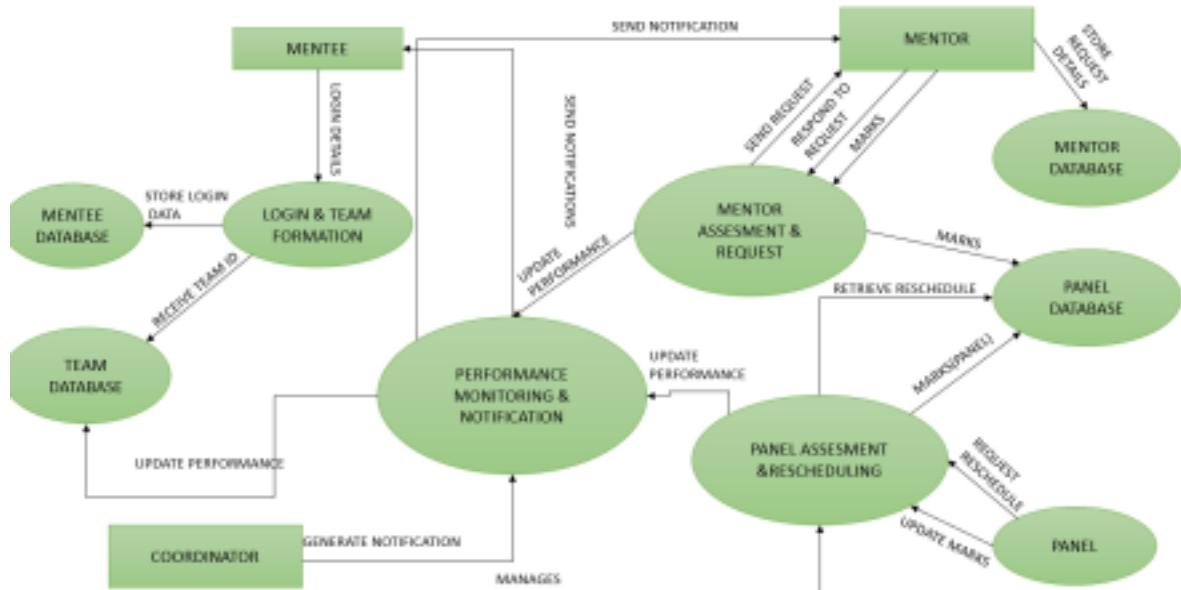


- DATA FLOW DIAGRAMS

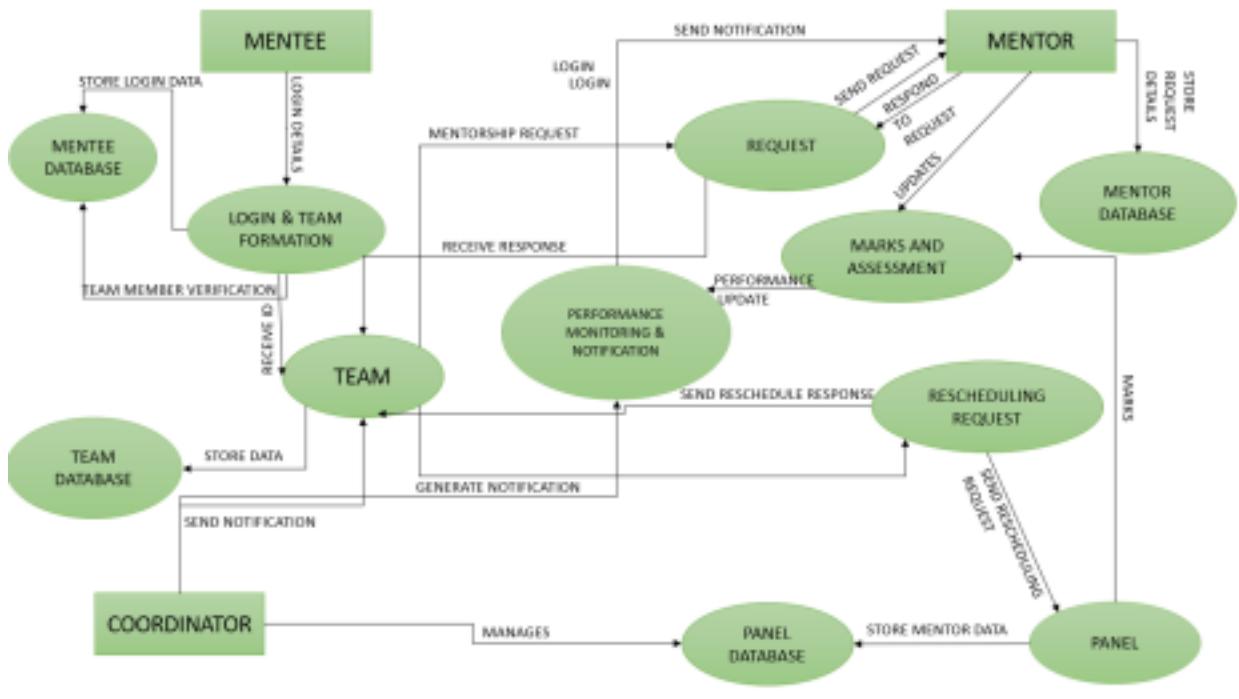
DFD LEVEL – 0



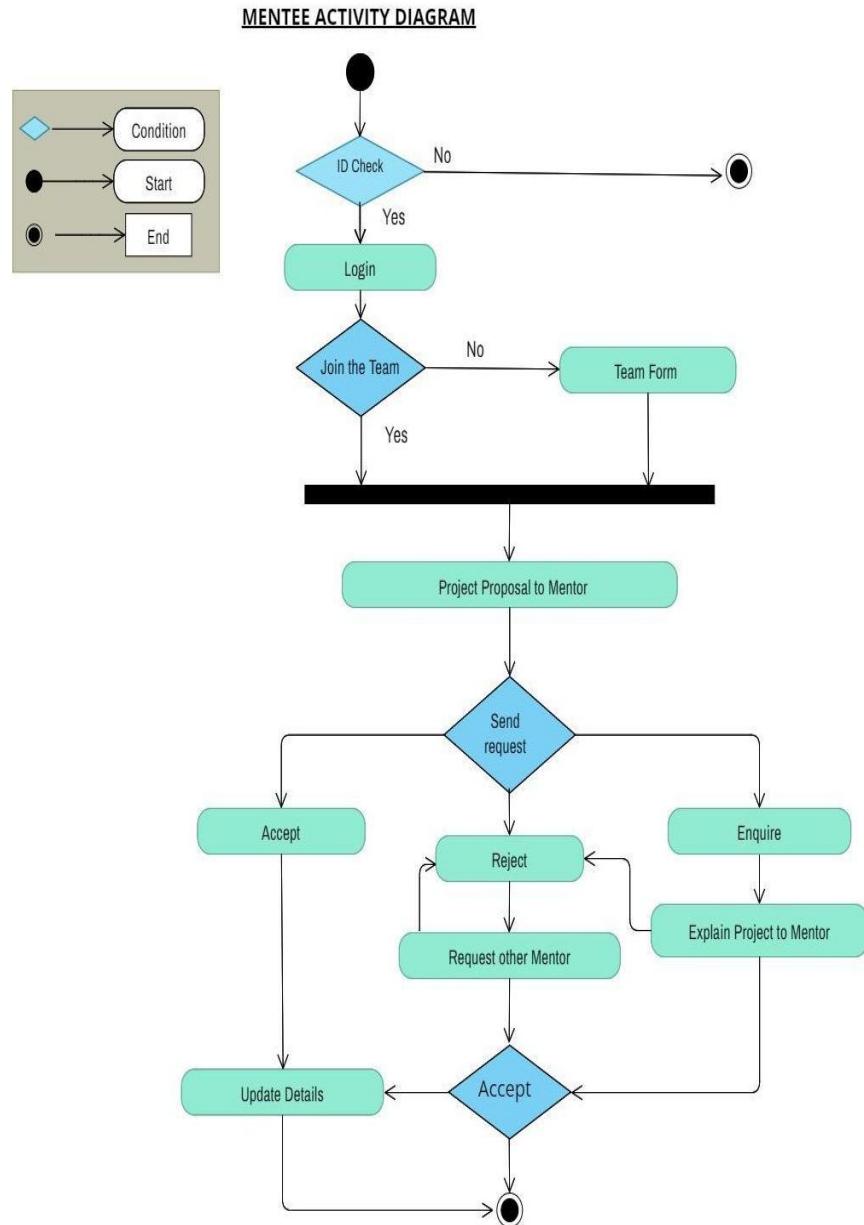
DFD LEVEL – 1

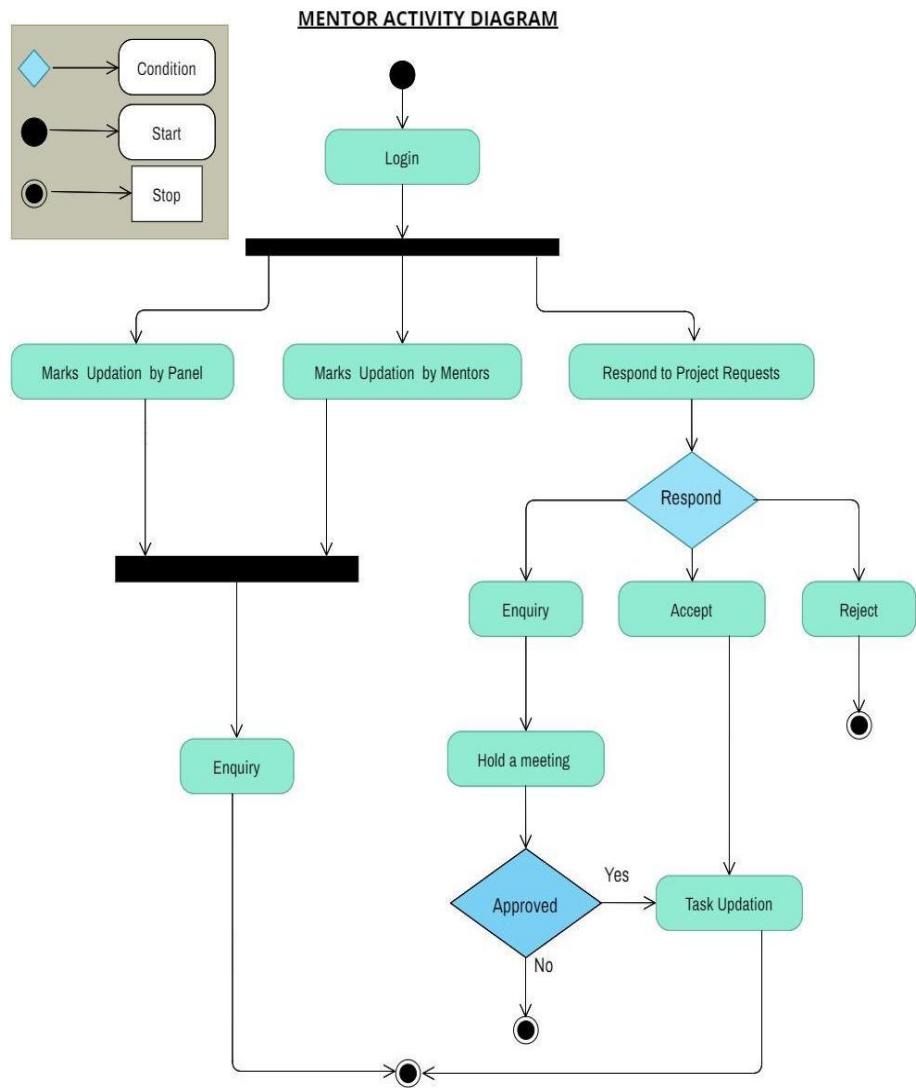


DFD LEVEL – 2

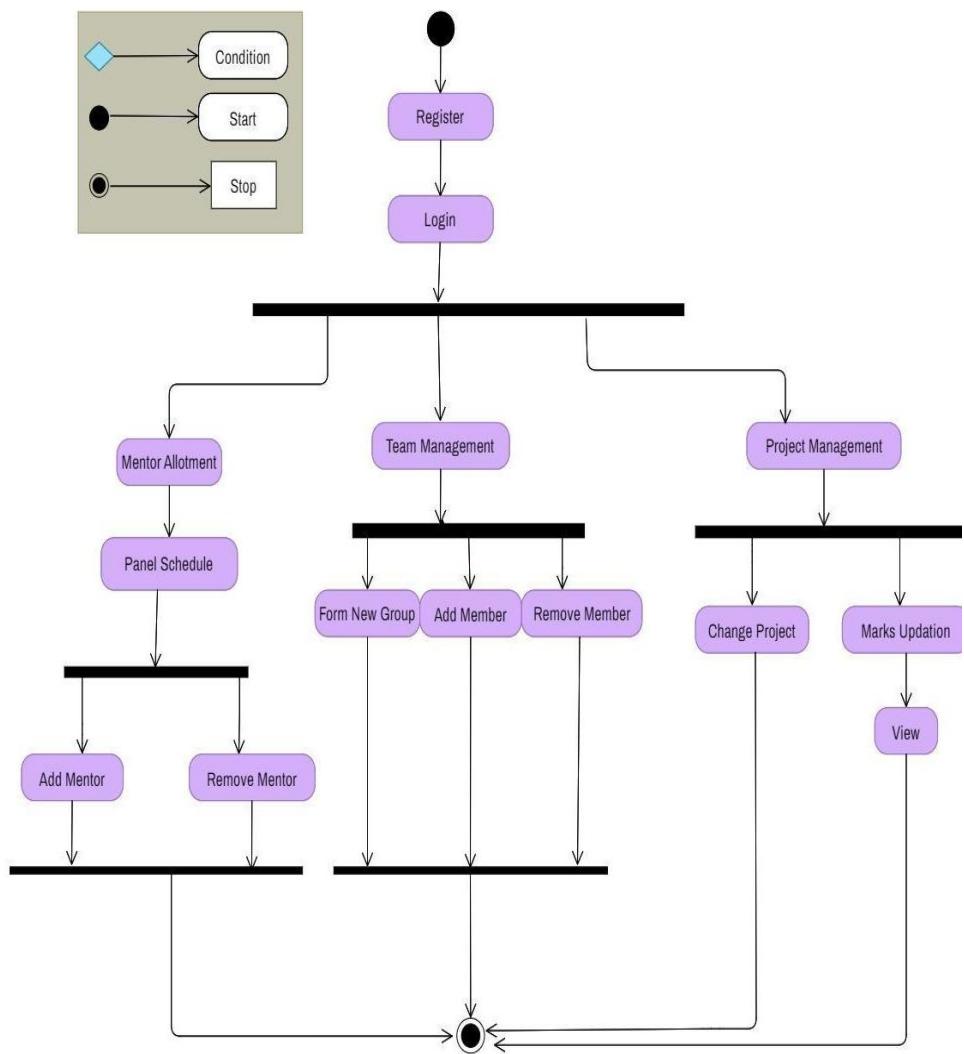


o ACTIVITY DIAGRAM

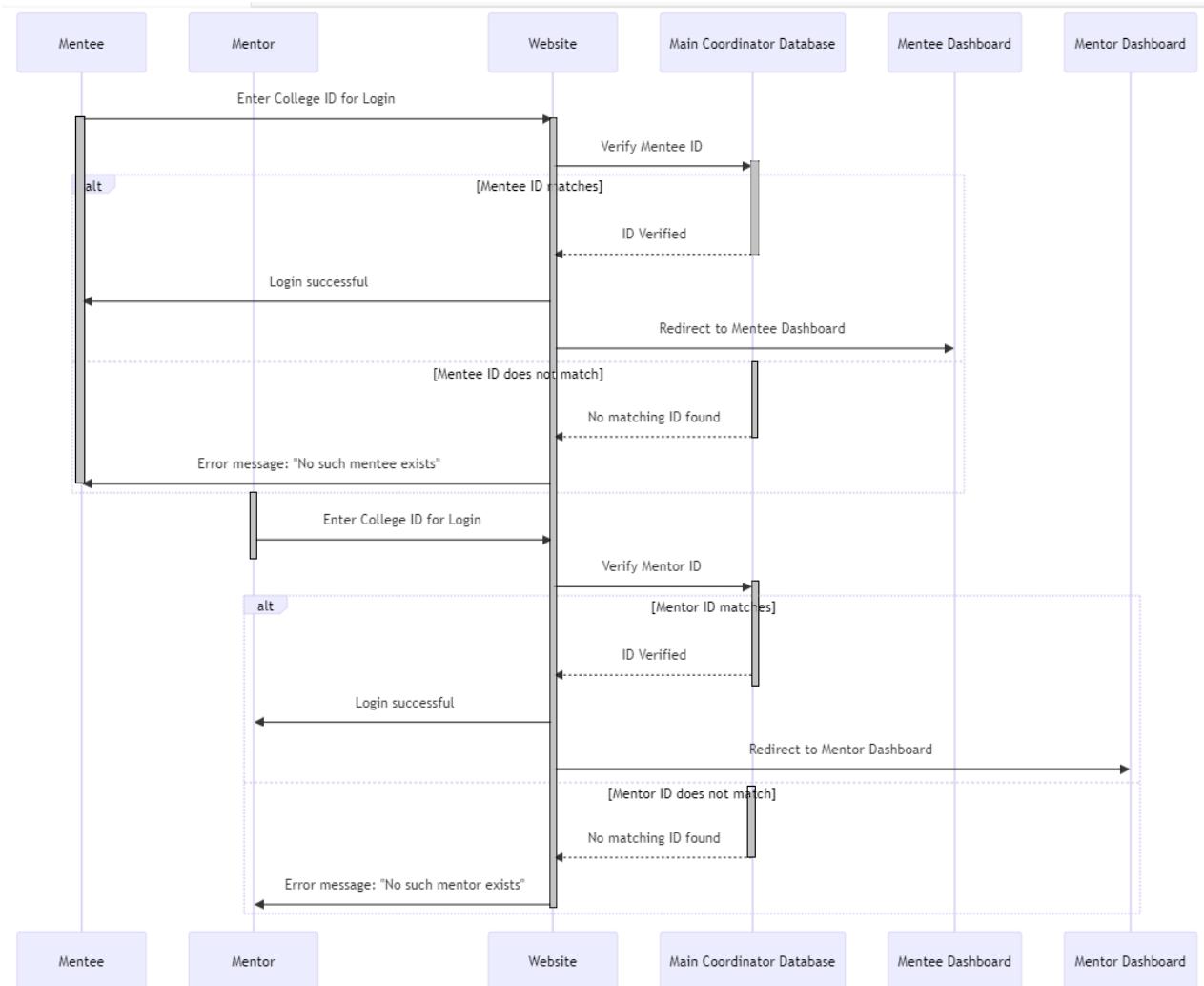




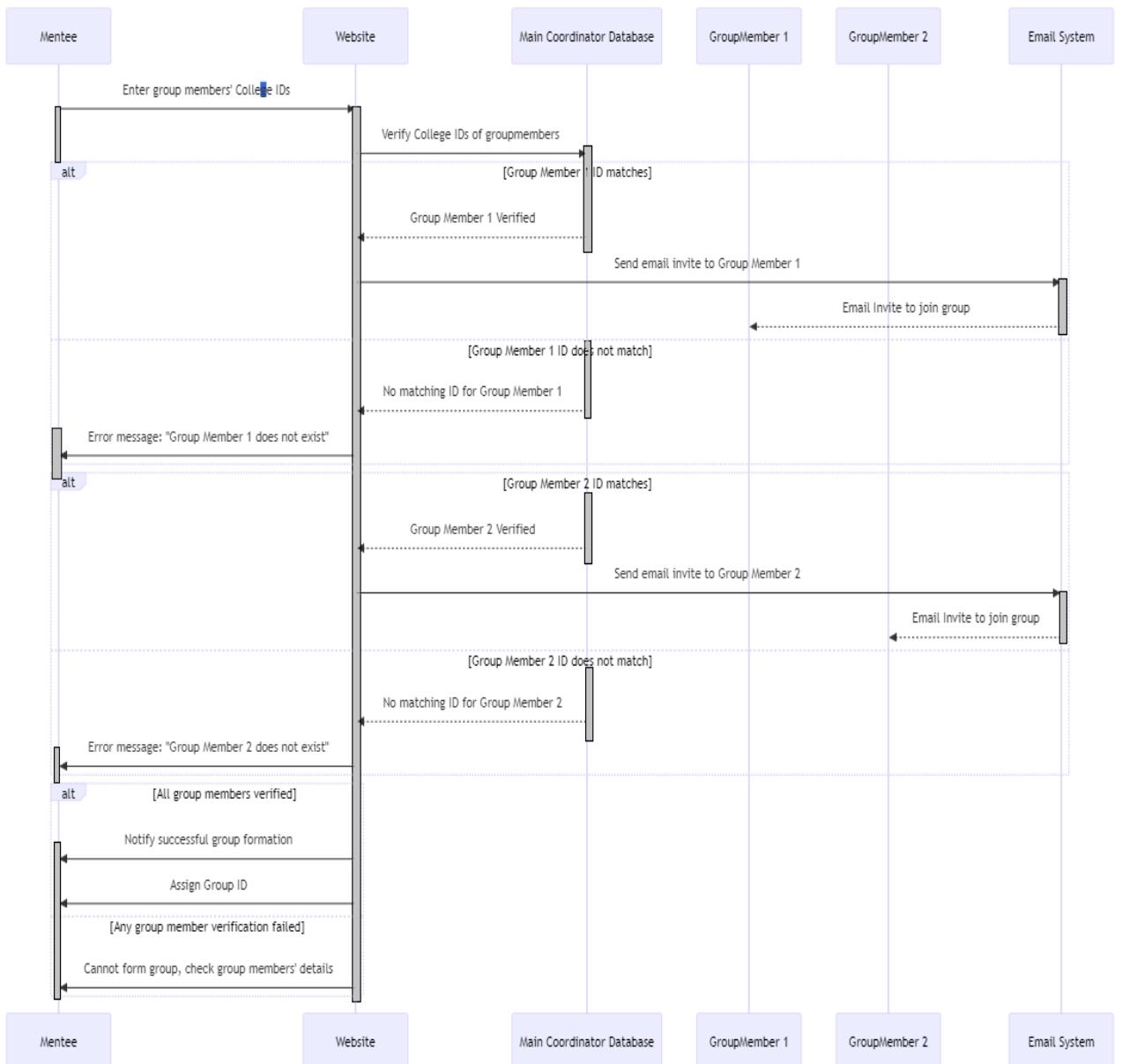
COORDINATOR ACTIVITY DIAGRAM



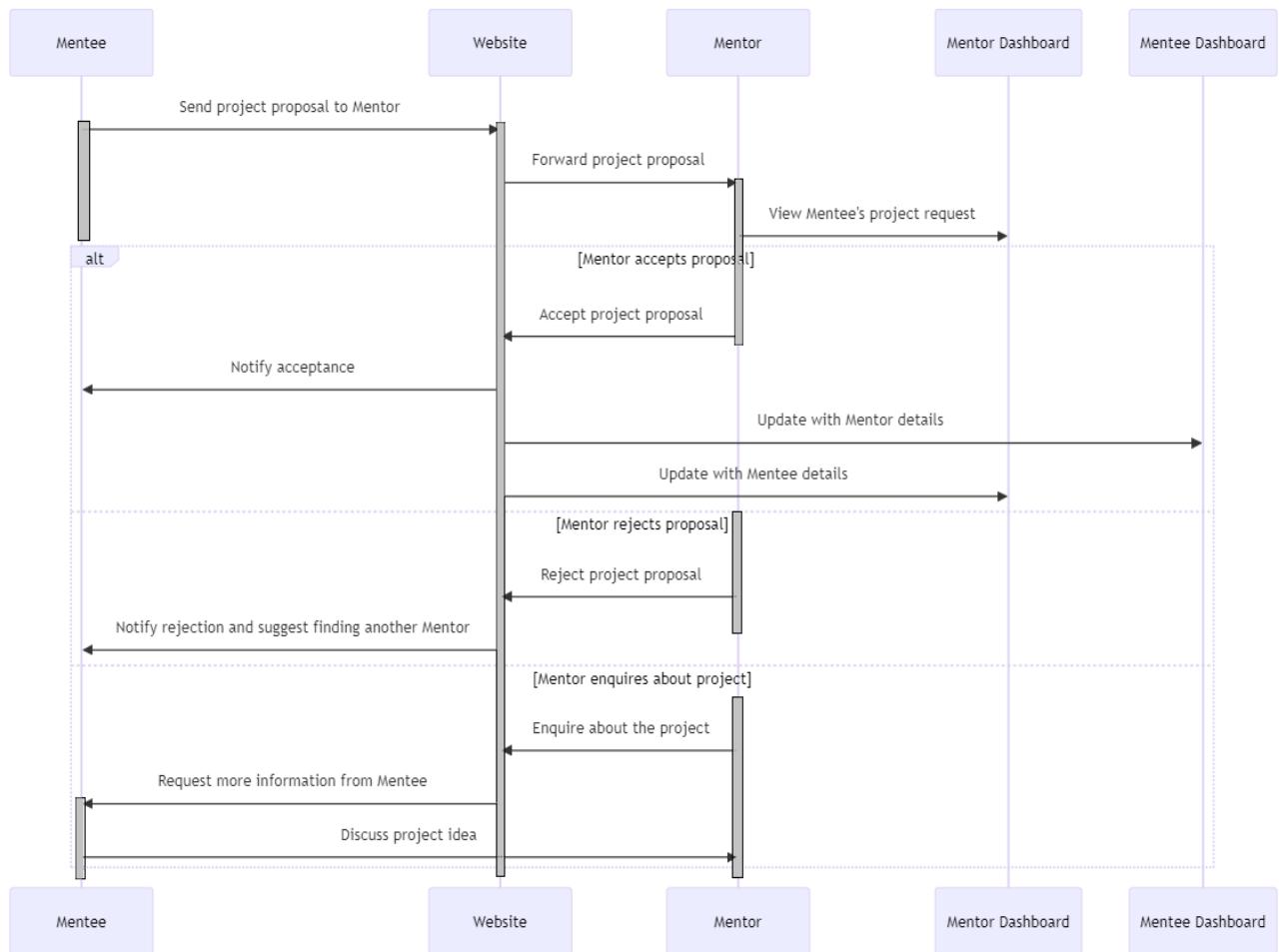
o SEQUENCE DIAGRAM



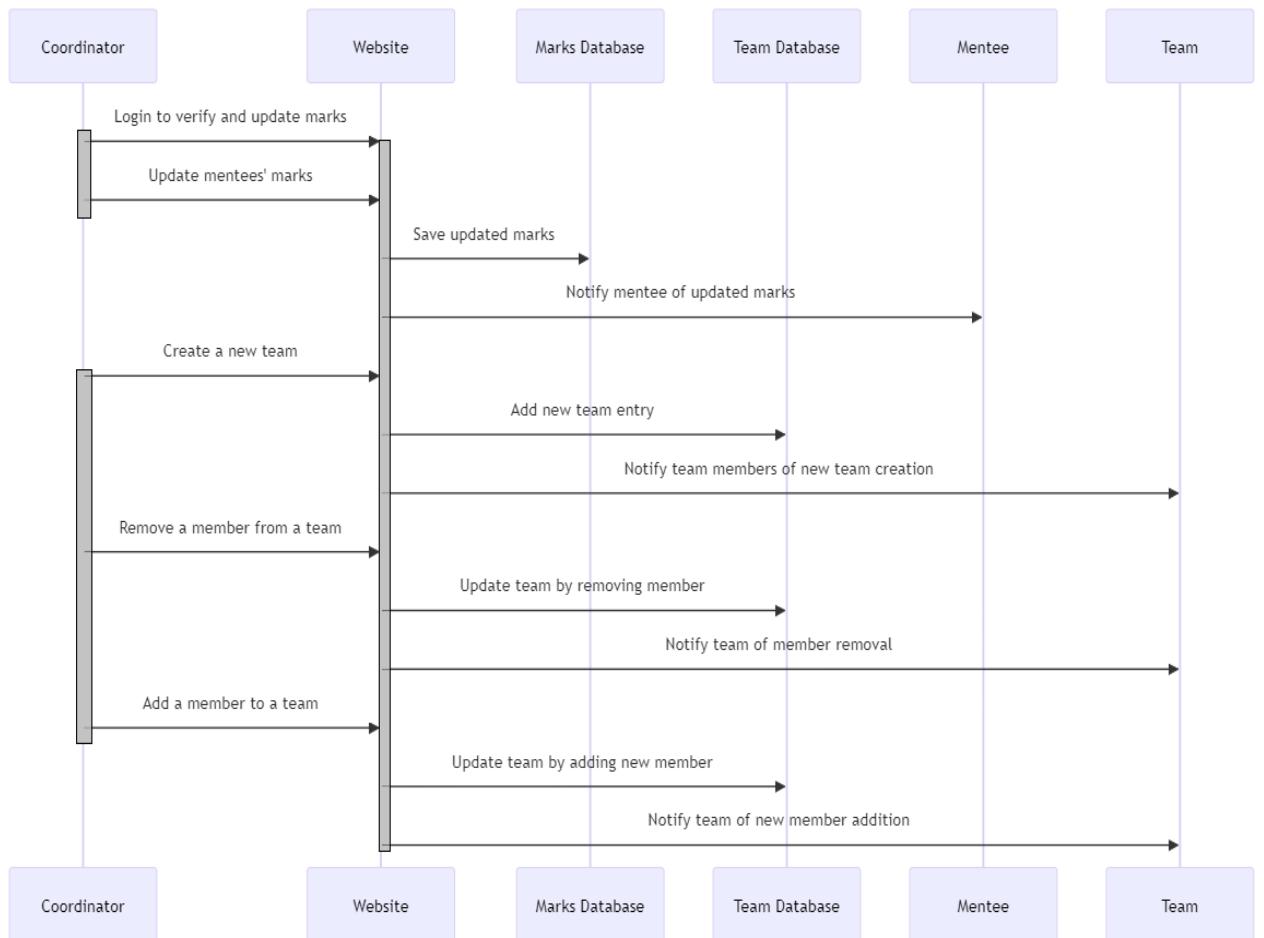
Login by both mentee and mentor



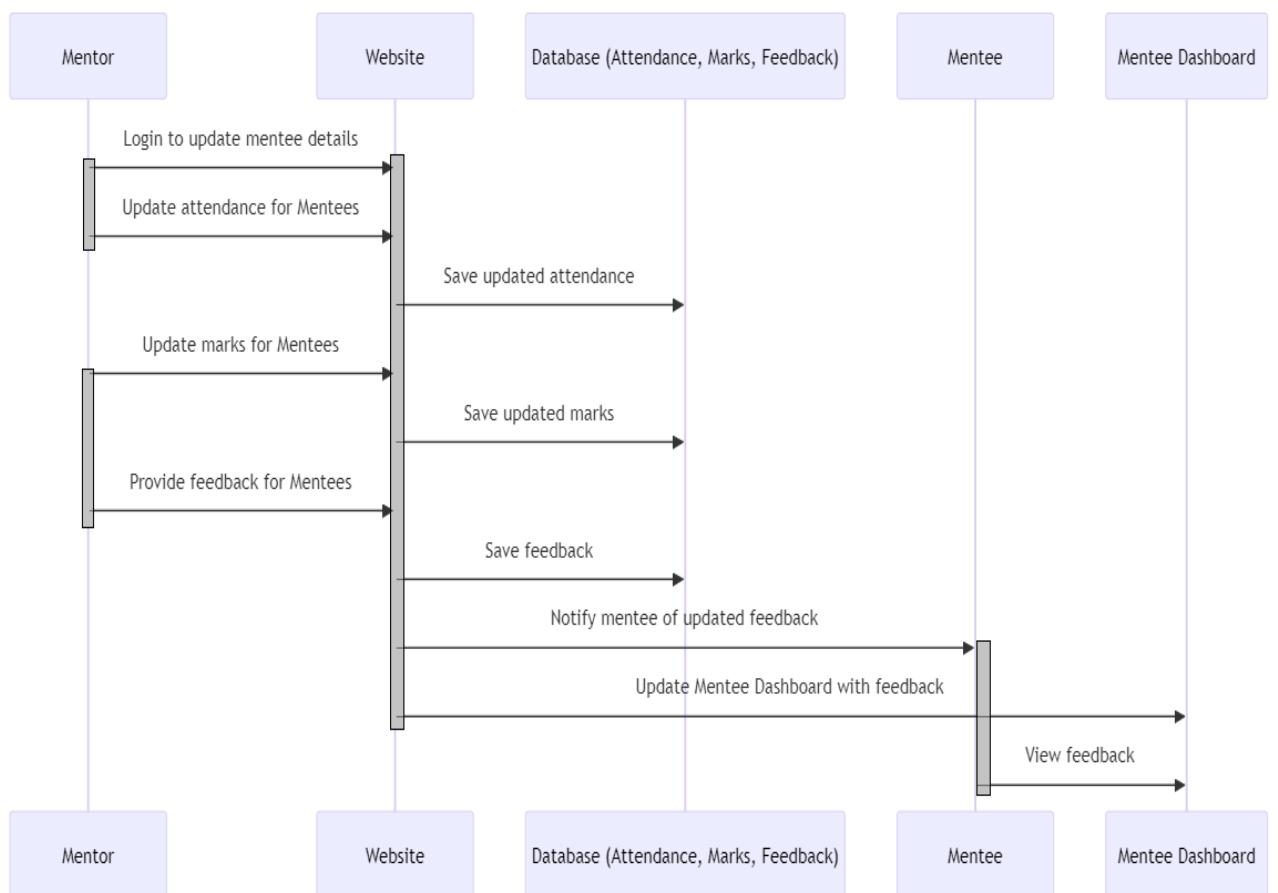
Team formation by mentee



Sending project proposal and receiving response from Mentor



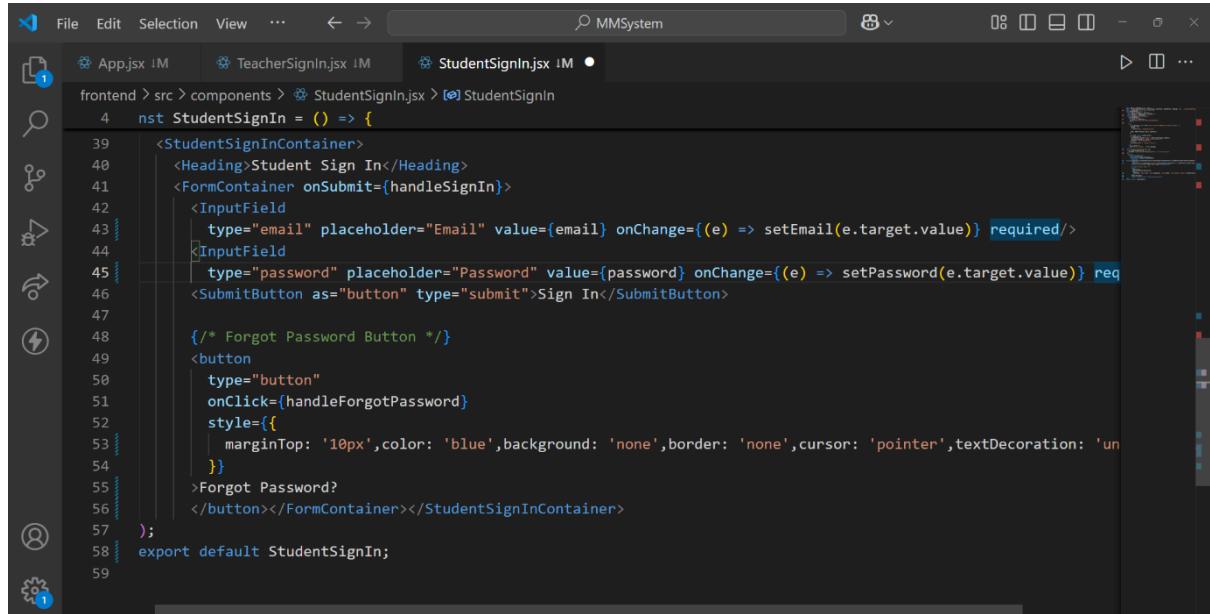
Coordinator accessing databases and performing updating



Mentor updating marks, attendance and providing feedback

CODING

LOGIN

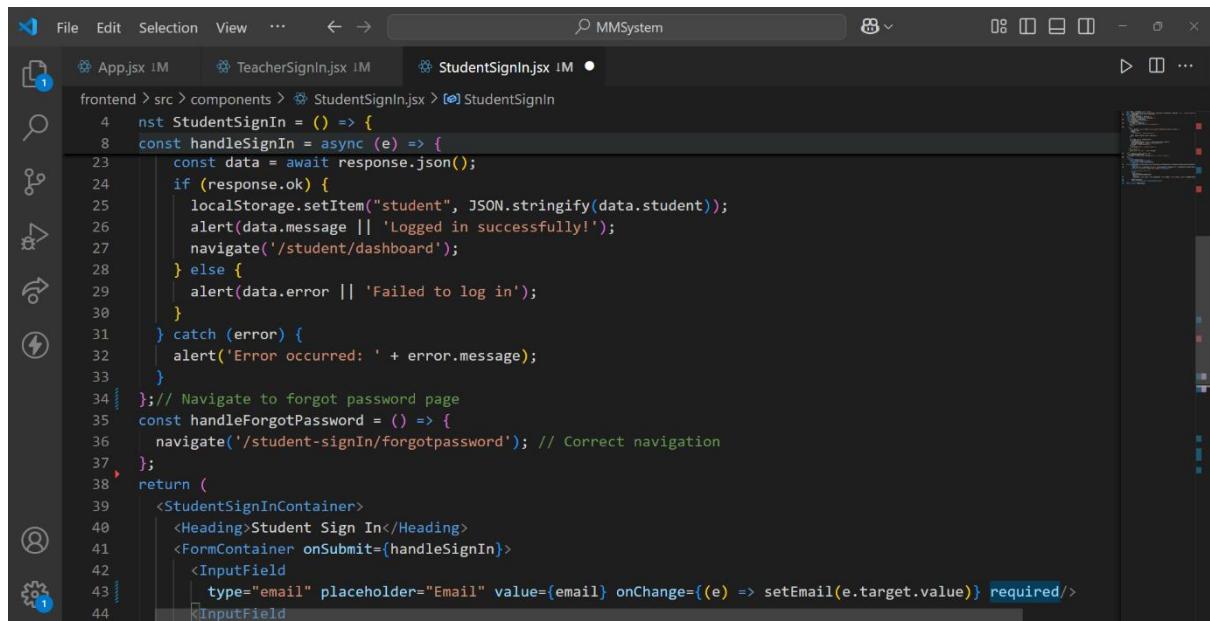


A screenshot of a code editor showing the `StudentSignIn` component. The component contains a form for logging in with email and password fields, and a forgot password button.

```
const StudentSignIn = () => {
  return (
    <StudentSignInContainer>
      <Heading>Student Sign In</Heading>
      <FormContainer onSubmit={handleSignIn}>
        <InputField
          type="email" placeholder="Email" value={email} onChange={(e) => setEmail(e.target.value)} required/>
        <InputField
          type="password" placeholder="Password" value={password} onChange={(e) => setPassword(e.target.value)} required/>
        <SubmitButton as="button" type="submit">Sign In</SubmitButton>

        {/* Forgot Password Button */}
        <button
          type="button"
          onClick={handleForgotPassword}
          style={{
            marginTop: '10px', color: 'blue', background: 'none', border: 'none', cursor: 'pointer', textDecoration: 'underline'
          }}
        >Forgot Password?
      </button>
    </FormContainer>
  );
}

export default StudentSignIn;
```

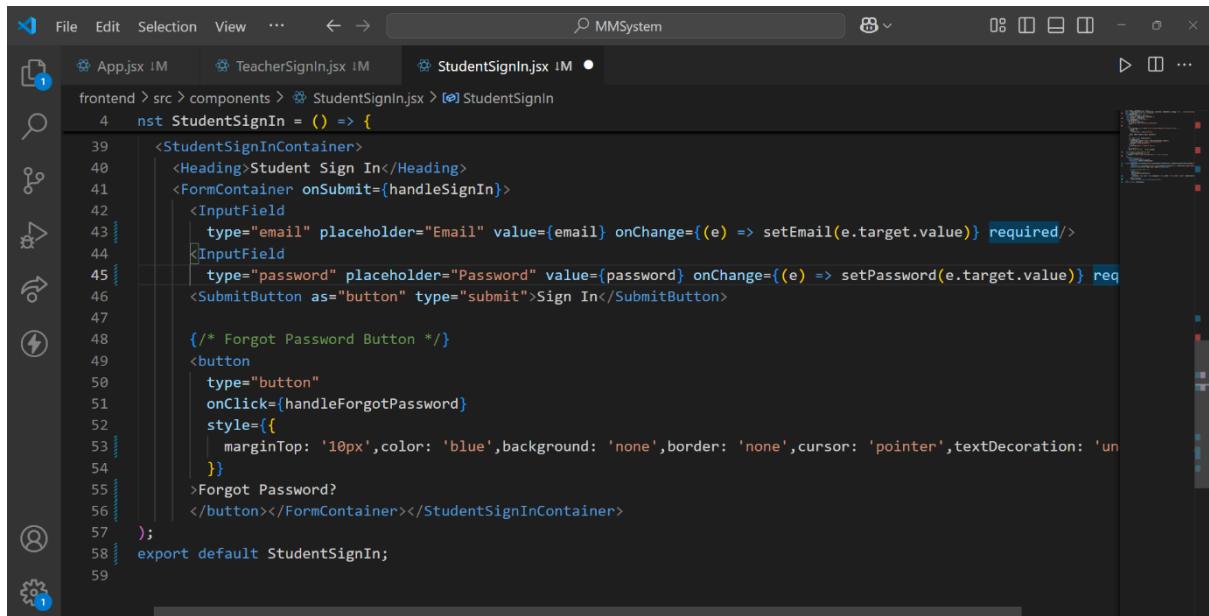


A screenshot of a code editor showing the implementation of the `handleSignIn` function. It uses an asynchronous function to handle the sign-in process, including handling errors and navigating to a dashboard.

```
const handleSignIn = async (e) => {
  const data = await response.json();
  if (response.ok) {
    localStorage.setItem("student", JSON.stringify(data.student));
    alert(data.message || 'Logged in successfully!');
    navigate('/student/dashboard');
  } else {
    alert(data.error || 'Failed to log in');
  }
} catch (error) {
  alert('Error occurred: ' + error.message);
}

// Navigate to forgot password page
const handleForgotPassword = () => {
  navigate('/student-signIn/forgotpassword'); // Correct navigation
};

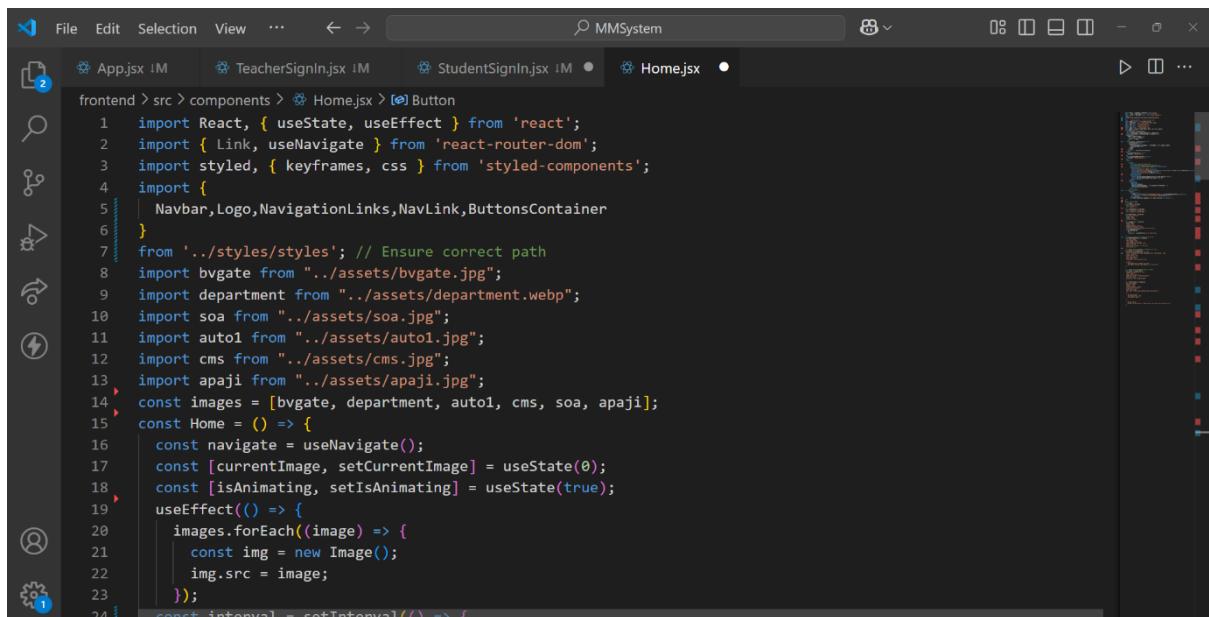
return (
  <StudentSignInContainer>
    <Heading>Student Sign In</Heading>
    <FormContainer onSubmit={handleSignIn}>
      <InputField
        type="email" placeholder="Email" value={email} onChange={(e) => setEmail(e.target.value)} required/>
      <InputField
```



A screenshot of a code editor window titled "MMSystem". The tabs at the top show "App.jsx", "TeacherSignIn.jsx", "StudentSignIn.jsx" (which is the active tab), and "Home.jsx". The code in the StudentSignIn.jsx file is as follows:

```
4 const StudentSignIn = () => {
39   <StudentSignInContainer>
40     <Heading>Student Sign In</Heading>
41     <FormContainer onSubmit={handleSignIn}>
42       <InputField type="email" placeholder="Email" value={email} onChange={(e) => setEmail(e.target.value)} required/>
43       <InputField type="password" placeholder="Password" value={password} onChange={(e) => setPassword(e.target.value)} required/>
44       <SubmitButton as="button" type="submit">Sign In</SubmitButton>
45
46       /* Forgot Password Button */
47       <button type="button" onClick={handleForgotPassword} style={{
48         marginTop: '10px', color: 'blue', background: 'none', border: 'none', cursor: 'pointer', textDecoration: 'underline'
49       }}>Forgot Password?</button>
50     </FormContainer></StudentSignInContainer>
51   );
52   export default StudentSignIn;
53 }
```

HOME PAGE



A screenshot of a code editor window titled "MMSystem". The tabs at the top show "App.jsx", "TeacherSignIn.jsx", "StudentSignIn.jsx", and "Home.jsx" (which is the active tab). The code in the Home.jsx file is as follows:

```
1 import React, { useState, useEffect } from 'react';
2 import { Link, useNavigate } from 'react-router-dom';
3 import styled, { keyframes, css } from 'styled-components';
4 import {
5   Navbar, Logo, NavigationLinks, NavLink, ButtonsContainer
6 }
7 from '../styles/styles'; // Ensure correct path
8 import bvgate from "../assets/bvgate.jpg";
9 import department from "../assets/department.webp";
10 import soa from "../assets/soa.jpg";
11 import auto1 from "../assets/auto1.jpg";
12 import cms from "../assets/cms.jpg";
13 import apoji from "../assets/apoji.jpg";
14 const images = [bvgate, department, auto1, cms, soa, apoji];
15 const Home = () => {
16   const navigate = useNavigate();
17   const [currentImage, setCurrentImage] = useState(0);
18   const [isAnimating, setIsAnimating] = useState(true);
19   useEffect(() => {
20     images.forEach((image) => {
21       const img = new Image();
22       img.src = image;
23     });
24     const interval = setInterval(() => {
```

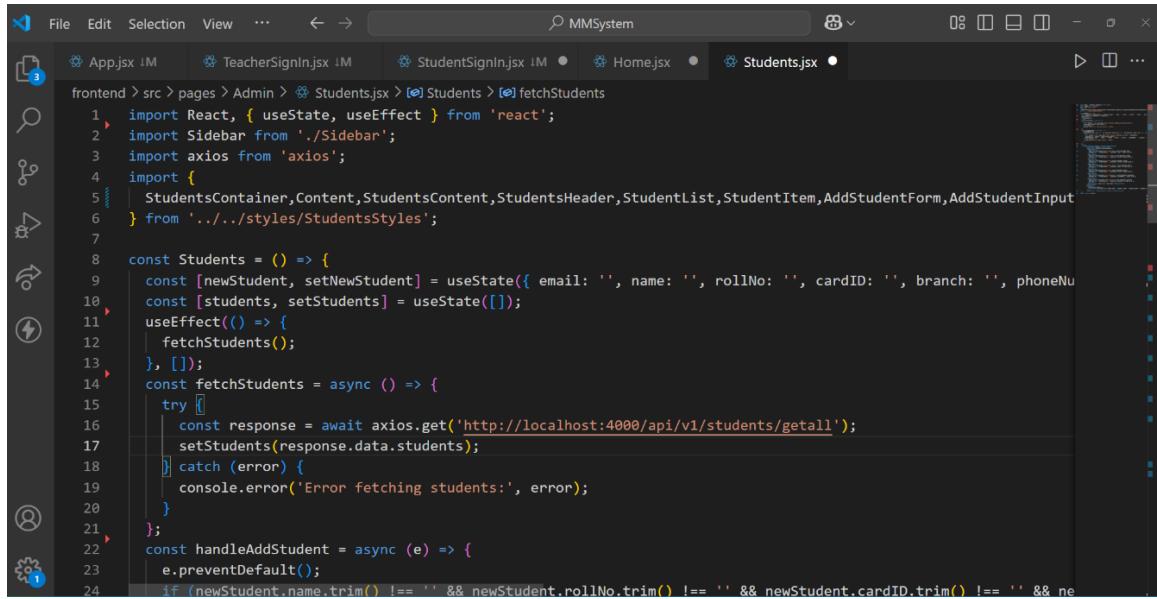
A screenshot of a code editor window titled "MMSystem". The current file is "Home.jsx". The code is as follows:

```
frontend > src > components > Home.jsx > [e] Home
15 const Home = () => {
16   useEffect(() => {
17     const interval = setInterval(() => {
18       setIsAnimating(false);
19       setTimeout(() => {
20         setCurrentImage((prevImage) => (prevImage + 1) % images.length);
21         setIsAnimating(true);
22       }, 300);
23     }, 3000);
24     return () => clearInterval(interval);
25   }, []);
26   const handleLoginClick = () => {
27     navigate('/choose-user');
28   };
29   const handleTeamRegisterClick = () => {
30     navigate('/teams/register');
31   };
32   return (
33     <Navbar>
34       <Logo src="/logo1.png" alt="Logo" />
35       {/* Mentor-Mentee System Title inside Navbar */}
36       <MentorMenteeTitle>Mentor-Mentee System</MentorMenteeTitle>
37       <NavigationLinks>
38         <NavLink to="/AboutUs">About Us</NavLink>
39         <NavLink to="http://www.banasthali.org/banasthali/wcms/en/home/" target="_blank">Banasthali</NavLink>
40         <NavLink to="/Coordinator">Message</NavLink>
41         <NavLink to="/teacher-details">Mentors</NavLink>
42     </NavigationLinks>
43     <ButtonsContainer>
44       <Button onClick={handleTeamRegisterClick}>Team Register</Button>
45       <Button onClick={handleLoginClick}>Sign In</Button>
46     </ButtonsContainer>
47   </Navbar>
48   <HeroSection>
49     <ImageSlider
50       key={currentImage}
51       style={({ backgroundImage: `url(${images[currentImage]})` })
52       $isAnimating={isAnimating}
53     />
54   </HeroSection>
55   /* Footer Section */
56   <Footer>
57     <FooterContent>
58       <p>Email: <a href="mailto:deanadmin@banasthali.ac.in">deanadmin@banasthali.ac.in</a></p>
```

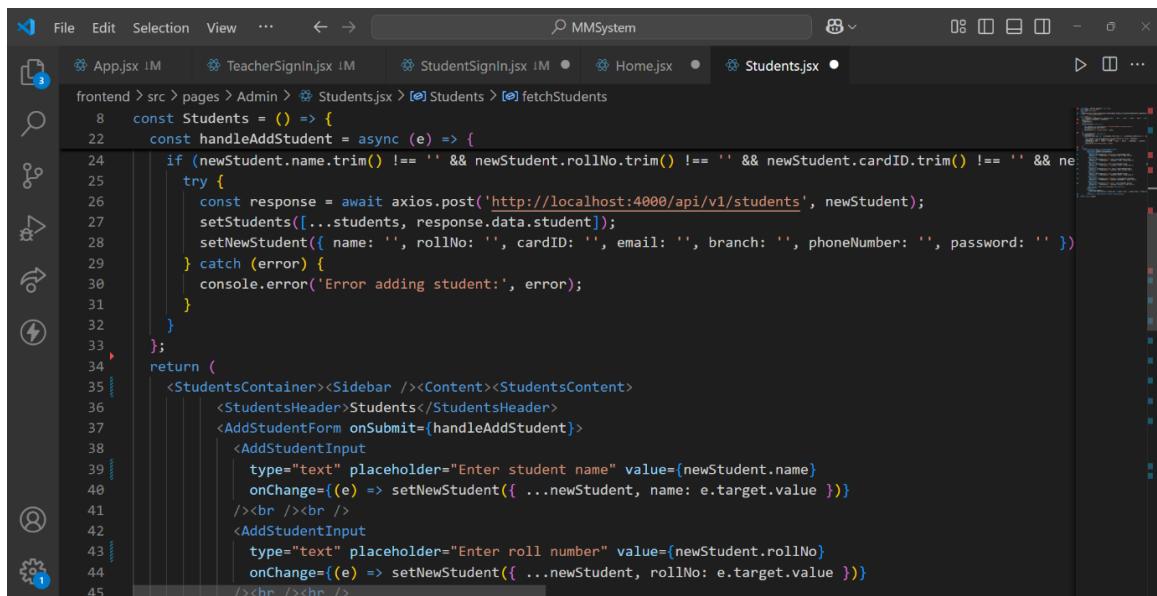
A screenshot of a code editor window titled "MMSystem". The current file is "Home.jsx". The code is as follows:

```
frontend > src > components > Home.jsx > [e] Home
15 const Home = () => {
16   useEffect(() => {
17     const interval = setInterval(() => {
18       setIsAnimating(false);
19       setTimeout(() => {
20         setCurrentImage((prevImage) => (prevImage + 1) % images.length);
21         setIsAnimating(true);
22       }, 300);
23     }, 3000);
24     return () => clearInterval(interval);
25   }, []);
26   const handleLoginClick = () => {
27     navigate('/choose-user');
28   };
29   const handleTeamRegisterClick = () => {
30     navigate('/teams/register');
31   };
32   return (
33     <Navbar>
34       <Logo src="/logo1.png" alt="Logo" />
35       {/* Mentor-Mentee System Title inside Navbar */}
36       <MentorMenteeTitle>Mentor-Mentee System</MentorMenteeTitle>
37       <NavigationLinks>
38         <NavLink to="/AboutUs">About Us</NavLink>
39         <NavLink to="http://www.banasthali.org/banasthali/wcms/en/home/" target="_blank">Banasthali</NavLink>
40         <NavLink to="/Coordinator">Message</NavLink>
41         <NavLink to="/teacher-details">Mentors</NavLink>
42     </NavigationLinks>
43     <ButtonsContainer>
44       <Button onClick={handleTeamRegisterClick}>Team Register</Button>
45       <Button onClick={handleLoginClick}>Sign In</Button>
46     </ButtonsContainer>
47   </Navbar>
48   <HeroSection>
49     <ImageSlider
50       key={currentImage}
51       style={({ backgroundImage: `url(${images[currentImage]})` })
52       $isAnimating={isAnimating}
53     />
54   </HeroSection>
55   /* Footer Section */
56   <Footer>
57     <FooterContent>
58       <p>Email: <a href="mailto:deanadmin@banasthali.ac.in">deanadmin@banasthali.ac.in</a></p>
```

STUDENT DASHBOARD



```
frontend > src > pages > Admin > Students.jsx > [e] fetchStudents
  1 import React, { useState, useEffect } from 'react';
  2 import Sidebar from './Sidebar';
  3 import axios from 'axios';
  4 import {
  5   StudentsContainer, Content, StudentsContent, StudentsHeader, StudentList, StudentItem, AddStudentForm, AddStudentInput
  6 } from '../../../../../styles/StudentsStyles';
  7
  8 const Students = () => {
  9   const [newStudent, setNewStudent] = useState({ email: '', name: '', rollNo: '', cardID: '', branch: '', phoneNu
10   const [students, setStudents] = useState([]);
11   useEffect(() => {
12     fetchStudents();
13   }, []);
14   const fetchStudents = async () => {
15     try {
16       const response = await axios.get('http://localhost:4000/api/v1/students/getall');
17       setStudents(response.data.students);
18     } catch (error) {
19       console.error('Error fetching students:', error);
20     }
21   };
22   const handleAddStudent = async (e) => {
23     e.preventDefault();
24     if (newStudent.name.trim() !== '' && newStudent.rollNo.trim() !== '' && newStudent.cardID.trim() !== '' && ne
```



```
frontend > src > pages > Admin > Students.jsx > [e] fetchStudents
  8 const Students = () => {
22   const handleAddStudent = async (e) => {
24     if (newStudent.name.trim() !== '' && newStudent.rollNo.trim() !== '' && newStudent.cardID.trim() !== '' && ne
25     try {
26       const response = await axios.post('http://localhost:4000/api/v1/students', newStudent);
27       setStudents([...students, response.data.student]);
28       setNewStudent({ name: '', rollNo: '', cardID: '', email: '', branch: '', phoneNumber: '', password: '' });
29     } catch (error) {
30       console.error('Error adding student:', error);
31     }
32   };
33 };
34 return (
35   <StudentsContainer><Sidebar /><Content><StudentsContent>
36     <StudentsHeader>Students</StudentsHeader>
37     <AddStudentForm onSubmit={handleAddStudent}>
38       <AddstudentInput
39         type="text" placeholder="Enter student name" value={newStudent.name}
40         onChange={(e) => setNewStudent({ ...newStudent, name: e.target.value })}>
41         /><br /><br />
42       <AddstudentInput
43         type="text" placeholder="Enter roll number" value={newStudent.rollNo}
44         onChange={(e) => setNewStudent({ ...newStudent, rollNo: e.target.value })}>
45         /><br /><br />
```

A screenshot of a code editor showing the `Students.jsx` component. The code handles adding new students, displaying a list of existing students, and rendering a form to add a new student. It uses functional components and hooks like `useState` and `useEffect`.

```
const Students = () => {
  const [newStudent, setNewStudent] = useState({ branch: "", phoneNumber: "", password: "" });
  const [students, setStudents] = useState([]);
  const [isFormVisible, setIsFormVisible] = useState(false);

  const handleAddStudent = (e) => {
    e.preventDefault();
    setStudents([...students, { ...newStudent, id: Date.now() }]);
    setNewStudent({ branch: "", phoneNumber: "", password: "" });
    setIsFormVisible(false);
  };

  const handleInputChange = (e) => {
    const { name, value } = e.target;
    setNewStudent({ ...newStudent, [name]: value });
  };

  const handleFormSubmit = (e) => {
    e.preventDefault();
    handleAddStudent(e);
  };

  return (
    <><br /><br />
    <AddStudentInput
      type="text"
      placeholder="Enter branch"
      value={newStudent.branch}
      onChange={(e) => setNewStudent({ ...newStudent, branch: e.target.value })}>
    </AddStudentInput>
    <br /><br />
    <AddStudentInput
      type="number"
      placeholder="Enter phoneNumber"
      value={newStudent.phoneNumber}
      onChange={(e) => setNewStudent({ ...newStudent, phoneNumber: e.target.value })}>
    </AddStudentInput>
    <br /><br />
    <AddStudentInput
      type="password"
      placeholder="Enter password"
      value={newStudent.password}
      onChange={(e) => setNewStudent({ ...newStudent, password: e.target.value })}>
    </AddStudentInput>
    <br /><br />
    <AddStudentButton type="submit">Add Student</AddStudentButton>
  );
};

export default Students;
```

EMAIL SERVICE

A screenshot of a code editor showing the `emailservice.js` file. It contains code for creating an email transporter using nodemailer, defining a template for emails, and setting up a port and secure connection.

```
import { createTransport } from "nodemailer";
// Email Transporter Configuration
const transporter = createTransport({
  service: "gmail",
  port: 587,
  secure: false,
  auth: {
    user: "mentor.mentee.banasthali@gmail.com",
    pass: "qsmk fceh oeof czjf"
  }
});

//Template
const getEmailTemplate = (title, message, role, link) => `
  <div style="`
```

The screenshot shows the VS Code interface with the title bar "MMSYSTEM". The left sidebar (EXPLORER) shows the project structure under "MMSSYSTEM": backend, routes (teacherRoutes.js, teamRoutes.js), services (emailservice.js, otpservice.js), uploads, utils, .env, .gitignore, app.js, fileupload.js, index.js, package-lock.json, package.json, frontend (node_modules, public). The right pane displays the code for "emailservice.js".

```
14  const getEmailTemplate = (title, message, role, link) => ` 15    <p style="font-size: 18px; line-height: 1.5; color: #f2f2f2;">${message}</p>
 16    <!-- CTA Button -->
 17    <div style="margin: 20px 0;">
 18      <a href="${link}" style="display: inline-block; background-color: white; color: #297011; font-size: 16px; font-weight: bold; text-decoration: none; border-radius: 6px; padding: 10px 20px; transition: 0.3s ease;">
 19        Go to Portal
 20      </a>
 21    </div>
 22    <p style="font-size: 14px; margin-top: 15px; color: white;">
 23      You are logged in as a <strong>${role}</strong>.
 24    </p>
 25    <hr style="border: 1px solid #fffff3; margin: 20px 0;">
 26    <!-- Footer -->
 27    <div style="text-align: center; font-size: 14px; color: #bbbbbb;">
 28      <p>Email: <a href="mailto:deanadmin@banasthali.ac.in" style="color: #f1c40f; text-decoration: none;">deanadmin@banasthali.ac.in</a>
 29      <p>Contact: <a href="tel:+911438228456" style="color: #f1c40f; text-decoration: none;">+911438228456</a>
 30      <p style="margin-top: 10px; font-size: 12px;">© 2025 Banasthali Vidyapith. All rights reserved.
 31    </div>
 32  
```

The screenshot shows the VS Code interface with the title bar "MMSYSTEM". The left sidebar (EXPLORER) shows the project structure under "MMSSYSTEM": backend, routes (teacherRoutes.js, teamRoutes.js), services (emailservice.js, otpservice.js), uploads, utils, .env, .gitignore, app.js, fileupload.js, index.js, package-lock.json, package.json, frontend (node_modules, public). The right pane displays the code for "emailservice.js".

```
50 // Send Student Email
51 export const sendStudentEmail = async (recipientEmail) => {
52   try {
53     const mailOptions = {
54       from: '"Mentor-Mentee Portal" <himanidobriyal8@gmail.com>',
55       to: recipientEmail,
56       subject: "Welcome to Mentor-Mentee Portal (Student)",
57       html: getEmailTemplate(
58         "Welcome, Student!",
59         "You have successfully logged in as a Student.",
60         "Student",
61         "http://localhost:5173/student/dashboard"
62       )
63     };
64
65     const info = await transporter.sendMail(mailOptions);
66     console.log("Student Email sent: " + info.response);
67     return { success: true, info };
68   } catch (error) {
69     console.error("Error sending student email:", error);
70     return { success: false, error };
71   }
72 }
73 
```

```

74 // Send Teacher Email
75 export const sendTeacherEmail = async (recipientEmail) => {
76   try {
77     const mailOptions = {
78       from: 'Mentor-Mentee Portal <himanidobriyal@gmail.com>',
79       to: recipientEmail,
80       subject: "Welcome to Mentor-Mentee Portal (Teacher)",
81       html: getEmailTemplate(
82         "Welcome, Teacher!",
83         "You have successfully logged in as a Teacher.",
84         "Teacher",
85         "http://localhost:5173/teacher/dashboard"
86       )
87     };
88
89     const info = await transporter.sendMail(mailOptions);
90     console.log("Teacher Email sent: " + info.response);
91     return { success: true, info };
92   } catch (error) {
93     console.error("Error sending teacher email:", error);
94     return { success: false, error };
95   }
96 };

```

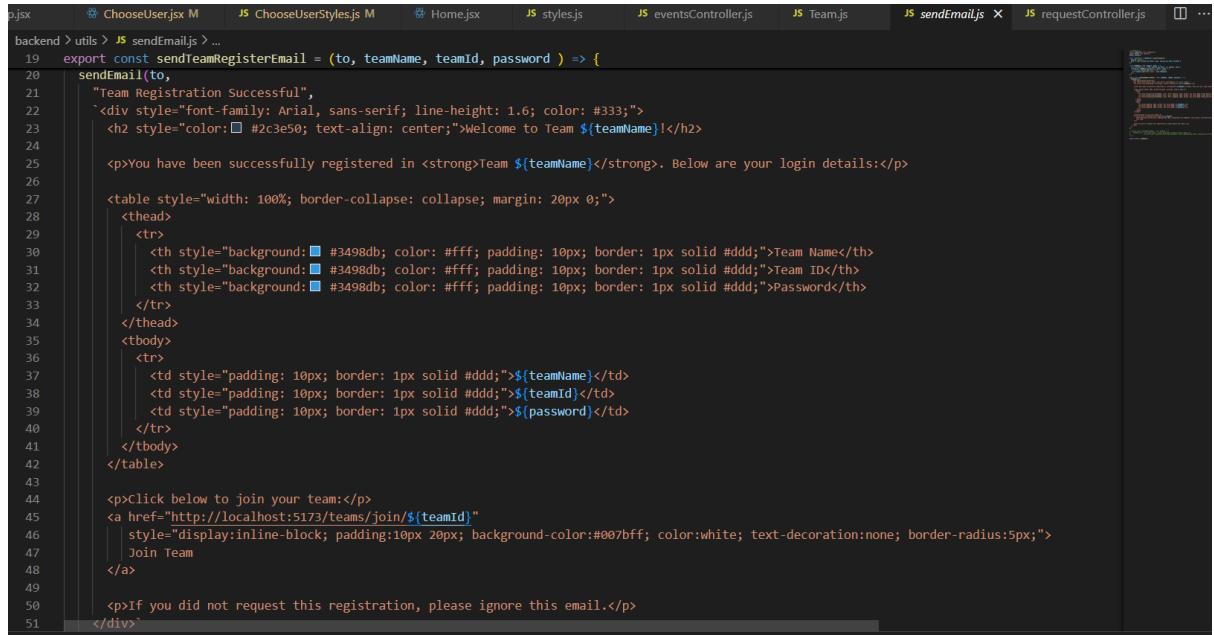
TEAM REGISTER

```

4
5 const TeamRegister = () => {
6   const [teamName, setTeamName] = useState("");
7   const [leaderEmail, setLeaderEmail] = useState("");
8   const [students, setStudents] = useState([{ name: "", email: "", course: "", studentId: "" }]);
9   const [members, setMembers] = useState([{ name: "", email: "", course: "" }]);
10  const navigate = useNavigate();
11
12  const generateTeamId = async (students) => {
13    if (!students || students.length === 0) return "T00";
14
15    const uniqueCourses = [...new Set(students.map((m) => m.course?.charAt(0).toUpperCase() || ""))];
16    const courseCode = uniqueCourses.sort().join("");
17
18    try {
19      const response = await axios.get("http://localhost:4000/api/v1/team/courses");
20      const totalTeams = response.data.count || 0;
21      const serialNumber = String(totalTeams + 1).padStart(2, "0");
22      return `${courseCode}${serialNumber}`;
23    } catch (error) {
24      console.error("Error fetching team count:", error);
25    }
26  };
27
28  const handleAddMember = () => {
29    setStudents([...students, { name: "", email: "", course: "", studentId: `S${students.length + 1}` }]);
30  };
31
32  const handleSubmit = async (e) => {
33    e.preventDefault();
34
35    if (!students || students.length === 0) {
36      alert("Please add at least one member to the team.");
37    }
38  };

```

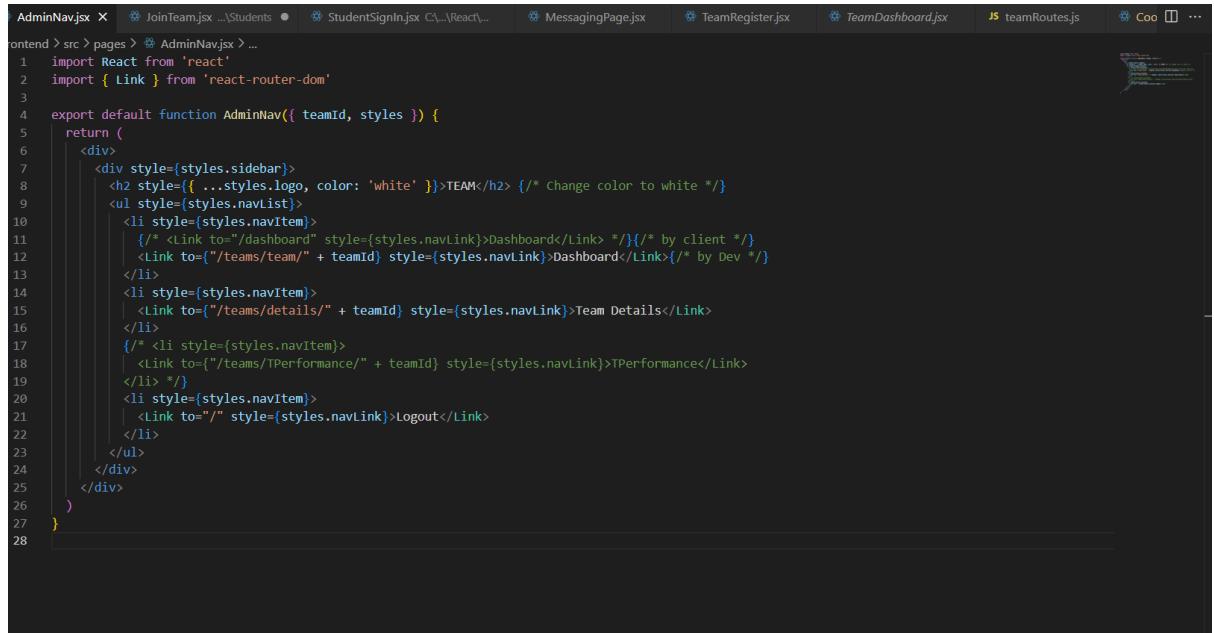
TEAM MAIL



A screenshot of a code editor showing a React component named `sendTeamRegisterEmail`. The component takes parameters `(to, teamName, teamId, password)` and sends an email with the following content:

```
19 export const sendTeamRegisterEmail = (to, teamName, teamId, password) => {
20   sendEmail(to,
21     "Team Registration Successful",
22     <div style="font-family: Arial, sans-serif; line-height: 1.6; color: #333;">
23       <h2 style="color: #2c3e50; text-align: center;">Welcome to Team ${teamName}!</h2>
24
25       <p>You have been successfully registered in <strong>Team ${teamName}</strong>. Below are your login details:</p>
26
27       <table style="width: 100%; border-collapse: collapse; margin: 20px 0;">
28         <thead>
29           <tr>
30             <th style="background: #3498db; color: #fff; padding: 10px; border: 1px solid #ddd;">Team Name</th>
31             <th style="background: #3498db; color: #fff; padding: 10px; border: 1px solid #ddd;">Team ID</th>
32             <th style="background: #3498db; color: #fff; padding: 10px; border: 1px solid #ddd;">Password</th>
33           </tr>
34         </thead>
35         <tbody>
36           <tr>
37             <td style="padding: 10px; border: 1px solid #ddd;">${teamName}</td>
38             <td style="padding: 10px; border: 1px solid #ddd;">${teamId}</td>
39             <td style="padding: 10px; border: 1px solid #ddd;">${password}</td>
40           </tr>
41         </tbody>
42       </table>
43
44       <p>Click below to join your team:</p>
45       <a href="http://localhost:5173/teams/join/${teamId}">
46         style="display:inline-block; padding:10px 20px; background-color:#007bff; color:white; text-decoration:none; border-radius:5px;">
47           Join Team
48         </a>
49
50       <p>If you did not request this registration, please ignore this email.</p>
51     </div>
```

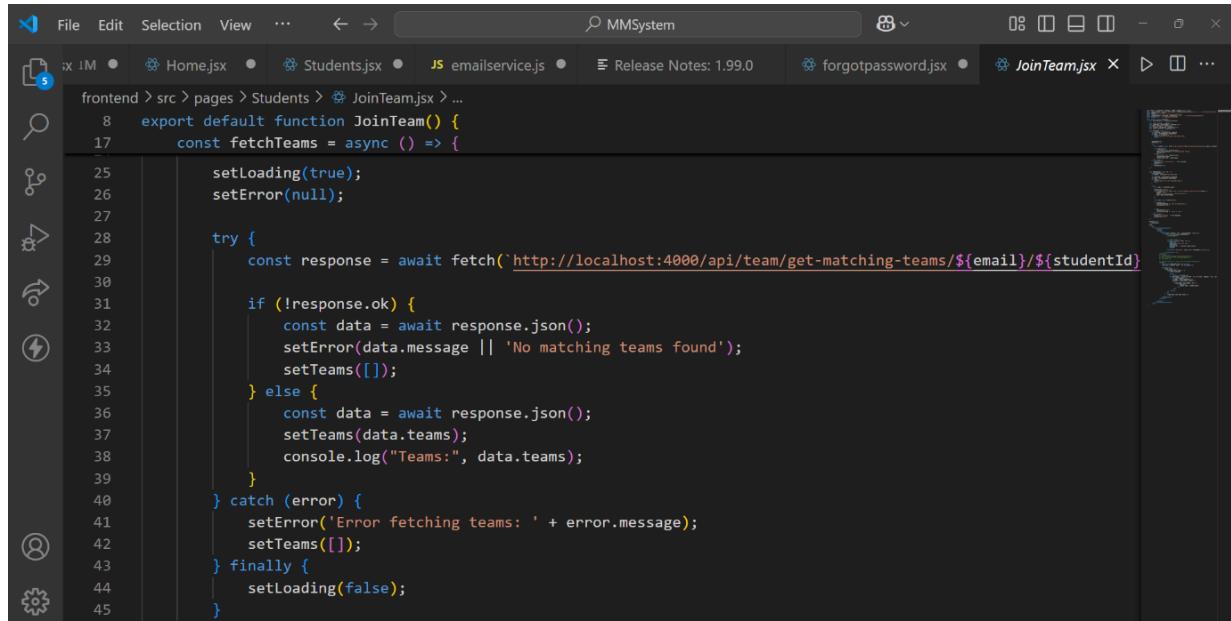
TEAM DASHBOARD



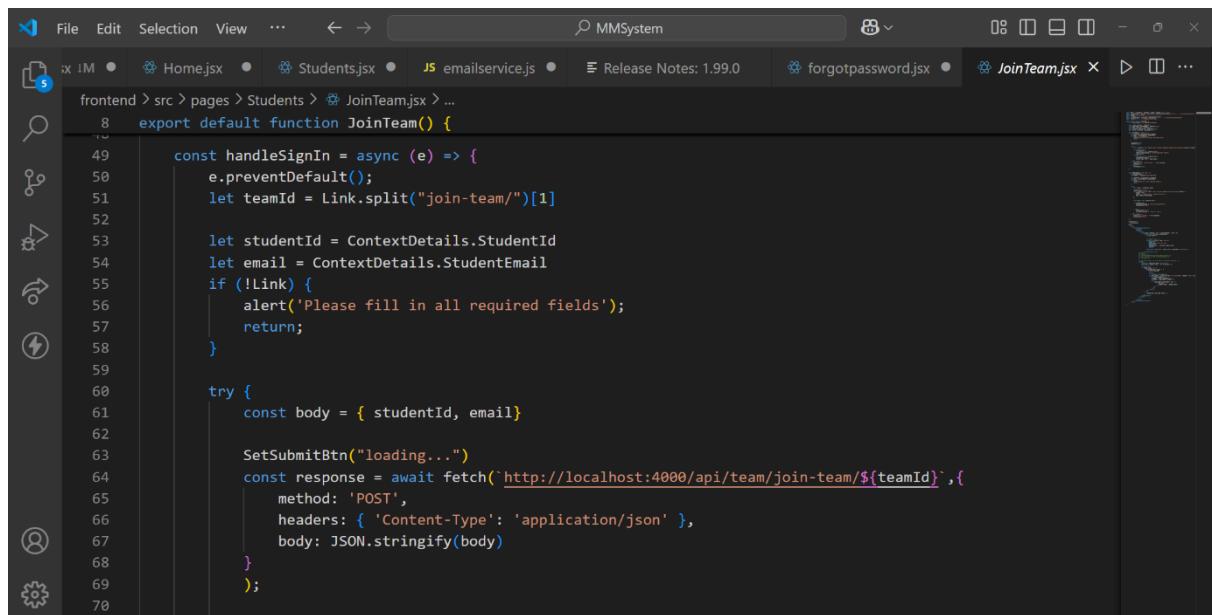
A screenshot of a code editor showing a React component named `AdminNav`. The component takes a parameter `{ teamId, styles }` and returns a sidebar navigation menu:

```
1 import React from 'react'
2 import { Link } from 'react-router-dom'
3
4 export default function AdminNav({ teamId, styles }) {
5   return (
6     <div>
7       <div style={styles.sidebar}>
8         <h2 style={{ ...styles.logo, color: 'white' }}>TEAM</h2> /* Change color to white */
9         <ul style={styles.navList}>
10           <li style={styles.navItem}>
11             /* <Link to="/dashboard" style={styles.navLink}>Dashboard</Link> */ /* by client */
12             <Link to={`/teams/team/" + teamId} style={styles.navLink}>Dashboard</Link> /* by dev */
13           </li>
14           <li style={styles.navItem}>
15             <Link to={`/teams/details/" + teamId} style={styles.navLink}>Team Details</Link>
16           </li>
17           /* <li style={styles.navItem}>
18             <Link to={`/teams/tPerformance/" + teamId} style={styles.navLink}>tPerformance</Link>
19           </li> */
20           <li style={styles.navItem}>
21             <Link to="/" style={styles.navLink}>Logout</Link>
22           </li>
23         </ul>
24       </div>
25     </div>
26   )
27 }
```

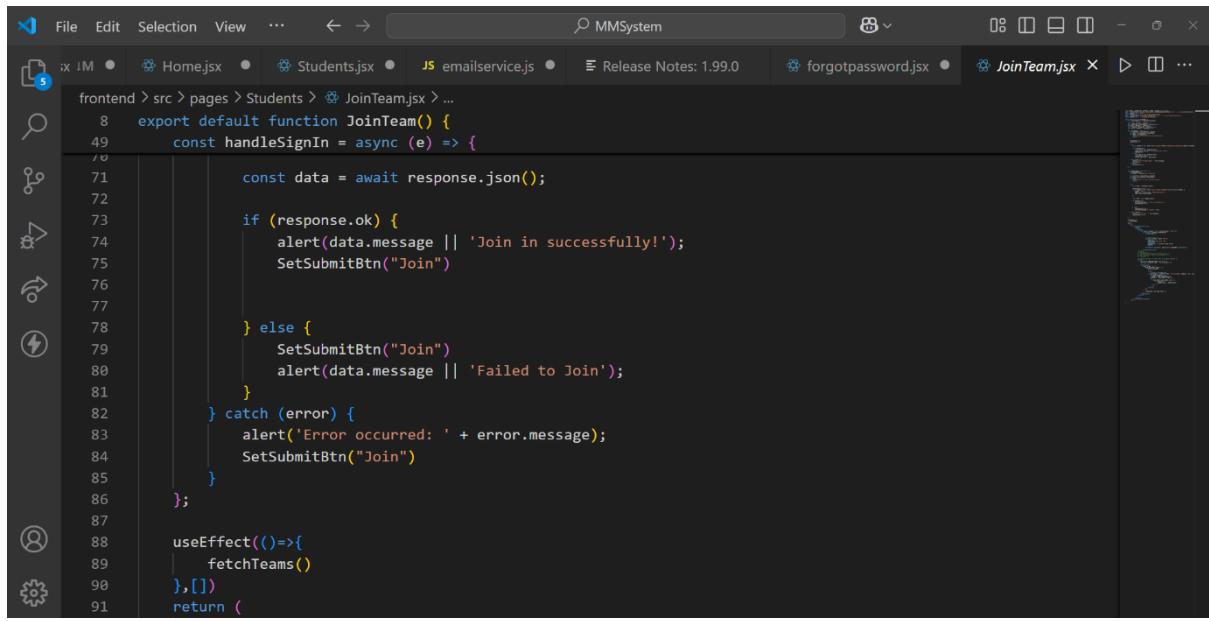
JOIN TEAM



```
frontend > src > pages > Students > JoinTeam.jsx > ...
8  export default function JoinTeam() {
17    const fetchTeams = async () => {
25      setLoading(true);
26      setError(null);
27
28      try {
29        const response = await fetch(`http://localhost:4000/api/team/get-matching-teams/${email}/${studentId}`);
30
31        if (!response.ok) {
32          const data = await response.json();
33          setError(data.message || 'No matching teams found');
34          setTeams([]);
35        } else {
36          const data = await response.json();
37          setTeams(data.teams);
38          console.log("Teams:", data.teams);
39        }
40      } catch (error) {
41        setError('Error fetching teams: ' + error.message);
42        setTeams([]);
43      } finally {
44        setLoading(false);
45      }
}
```



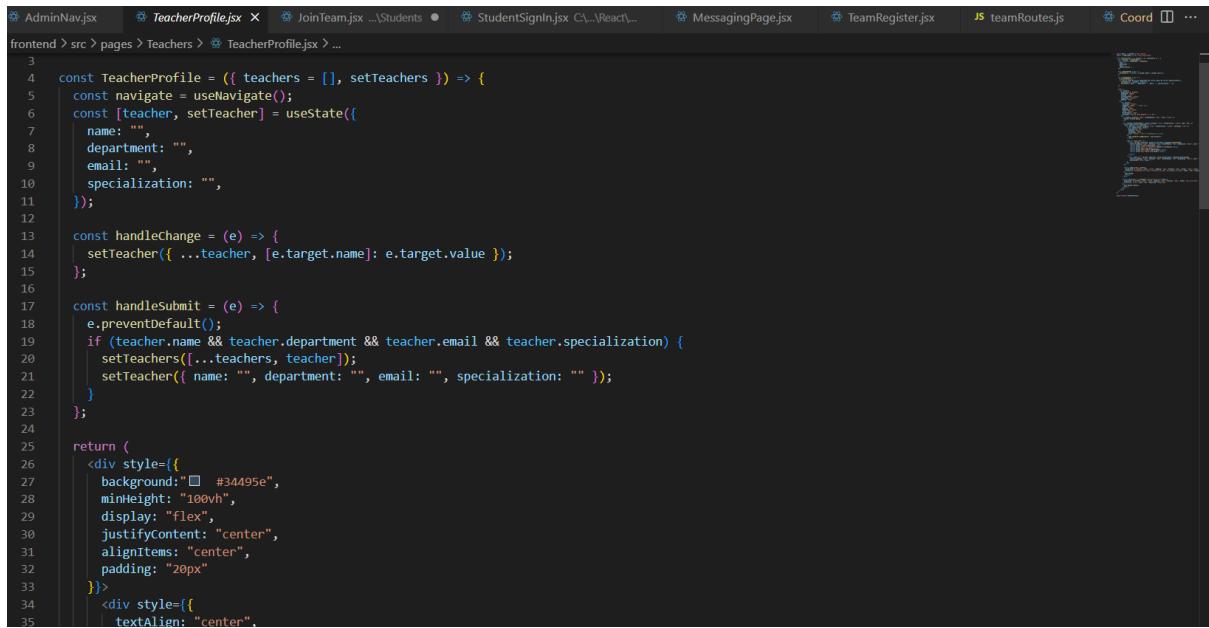
```
frontend > src > pages > Students > JoinTeam.jsx > ...
8  export default function JoinTeam() {
49    const handleSignIn = async (e) => {
50      e.preventDefault();
51      let teamId = Link.split("join-team/")[1];
52
53      let studentId = ContextDetails.StudentId;
54      let email = ContextDetails.StudentEmail;
55      if (!link) {
56        alert('Please fill in all required fields');
57        return;
58      }
59
60      try {
61        const body = { studentId, email };
62
63        SetSubmitBtn("loading...");
64        const response = await fetch(`http://localhost:4000/api/team/join-team/${teamId}`, {
65          method: 'POST',
66          headers: { 'Content-Type': 'application/json' },
67          body: JSON.stringify(body)
68        });
69      }
70    };
}
```



```
frontend > src > pages > Students > JoinTeam.jsx > ...
8  export default function JoinTeam() {
49    const handleSignIn = async (e) => {
70
71      const data = await response.json();
72
73      if (response.ok) {
74        alert(data.message || 'Join in successfully!');
75        SetSubmitBtn("Join")
76
77      } else {
78        SetSubmitBtn("Join")
79        alert(data.message || 'Failed to Join');
80      }
81    } catch (error) {
82      alert('Error occurred: ' + error.message);
83      SetSubmitBtn("Join")
84    }
85  };
86
88  useEffect(()=>{
89    fetchTeams()
90  },[])
91
92  return (

```

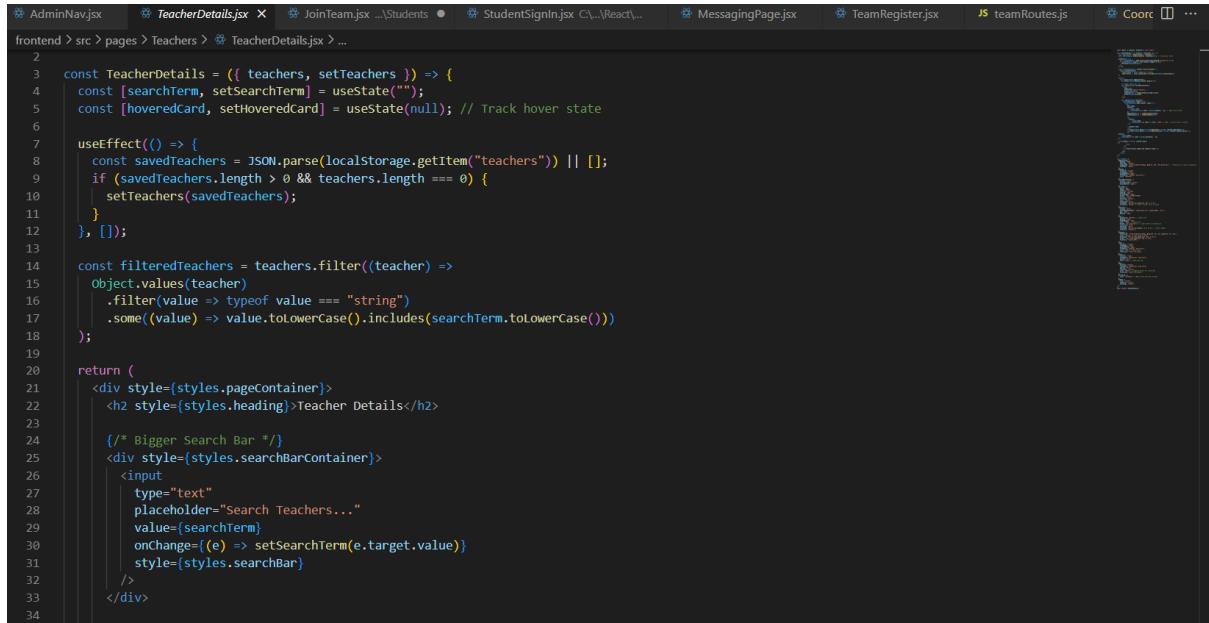
TEACHER PROFILE



```
frontend > src > pages > Teachers > TeacherProfile.jsx > ...
3
4  const TeacherProfile = ({ teachers = [], setTeachers }) => {
5    const navigate = useNavigate();
6    const [teacher, setTeacher] = useState({
7      name: "",
8      department: "",
9      email: "",
10     specialization: ""
11   });
12
13   const handleChange = (e) => {
14     setTeacher({ ...teacher, [e.target.name]: e.target.value });
15   };
16
17   const handleSubmit = (e) => {
18     e.preventDefault();
19     if (teacher.name && teacher.department && teacher.email && teacher.specialization) {
20       setTeachers([...teachers, teacher]);
21       setTeacher({ name: "", department: "", email: "", specialization: "" });
22     }
23   };
24
25   return (
26     <div style={{
27       background: "#34495e",
28       minHeight: "100vh",
29       display: "flex",
30       justifyContent: "center",
31       alignItems: "center",
32       padding: "20px"
33     }>
34       <div style={{
35         textAlign: "center",

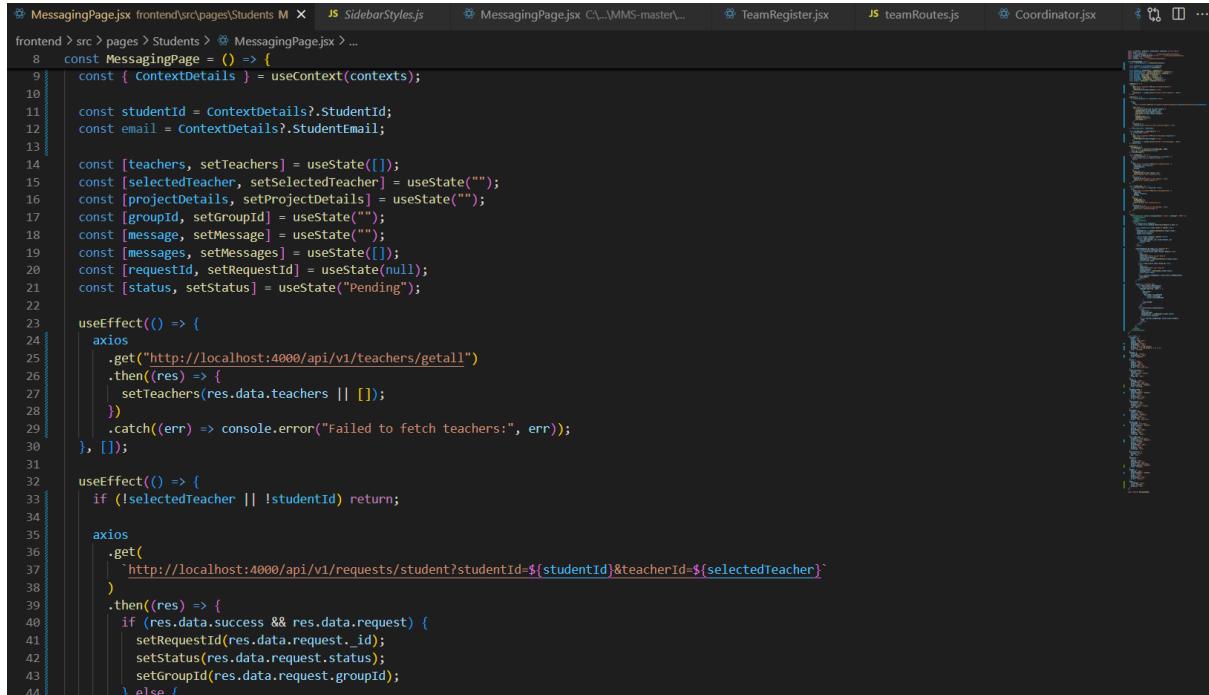
```

TEACHER DETAILS



```
AdminNav.jsx TeacherDetails.jsx X JoinTeam.jsx ...Students ... StudentSignIn.jsx C:\...\React\... MessagingPage.jsx TeamRegister.jsx JS teamRoutes.js Coore ...  
frontend > src > pages > Teachers > TeacherDetails.jsx > ...  
2  
3 const TeacherDetails = ({ teachers, setTeachers }) => {  
4   const [searchTerm, setSearchTerm] = useState("");  
5   const [hoveredCard, setHoveredCard] = useState(null); // Track hover state  
6  
7   useEffect(() => {  
8     const savedTeachers = JSON.parse(localStorage.getItem("teachers")) || [];  
9     if (savedTeachers.length > 0 && teachers.length === 0) {  
10       setTeachers(savedTeachers);  
11     }  
12   }, [ ]);  
13  
14   const filteredTeachers = teachers.filter((teacher) =>  
15     Object.values(teacher)  
16       .filter(value => typeof value === "string")  
17       .some((value) => value.toLowerCase().includes(searchTerm.toLowerCase()))  
18   );  
19  
20   return (  
21     <div style={styles.pageContainer}>  
22       <h2 style={styles.heading}>Teacher Details</h2>  
23  
24       /* Bigger Search Bar */  
25       <div style={styles.searchBarContainer}>  
26         <input  
27           type="text"  
28           placeholder="Search Teachers..."  
29           value={searchTerm}  
30           onChange={(e) => setSearchTerm(e.target.value)}  
31           style={styles.searchBar}  
32         />  
33       </div>  
34     </div>
```

MESSAGING REQUESTS



```
MessagingPage.jsx frontend\src\pages\Students M JS SidebarStyles.js MessagingPage.jsx C:\...\MMS-master\... TeamRegister.jsx JS teamRoutes.js Coordinator.jsx ...  
frontend > src > pages > Students > MessagingPage.jsx > ...  
8 const MessagingPage = () => {  
9   const { ContextDetails } = useContext(contexts);  
10  
11   const studentId = ContextDetails?.studentId;  
12   const email = ContextDetails?.studentEmail;  
13  
14   const [teachers, setTeachers] = useState([]);  
15   const [selectedTeacher, setSelectedTeacher] = useState("");  
16   const [projectDetails, setProjectDetails] = useState("");  
17   const [groupId, setGroupId] = useState("");  
18   const [message, setMessage] = useState("");  
19   const [messages, setMessages] = useState([]);  
20   const [requestId, setRequestId] = useState(null);  
21   const [status, setStatus] = useState("Pending");  
22  
23   useEffect(() => {  
24     axios  
25       .get("http://localhost:4000/api/v1/teachers/getall")  
26       .then((res) => {  
27         setTeachers(res.data.teachers || []);  
28       })  
29       .catch((err) => console.error("Failed to fetch teachers:", err));  
30   }, [ ]);  
31  
32   useEffect(() => {  
33     if (!selectedTeacher || !studentId) return;  
34  
35     axios  
36       .get(  
37         `http://localhost:4000/api/v1/requests/student?studentId=${studentId}&teacherId=${selectedTeacher}`  
38       )  
39       .then((res) => {  
40         if (res.data.success && res.data.request) {  
41           setRequestId(res.data.request._id);  
42           setStatus(res.data.request.status);  
43           setGroupId(res.data.request.groupId);  
44         } else {  
45       }  
46     }  
47   }, [selectedTeacher, studentId]);  
48  
49   return (  
50     <div style={styles.pageContainer}>  
51       <h2 style={styles.heading}>Messaging Requests</h2>  
52       <div style={styles.messageList}>  
53         {messages.map((message) => (  
54           <div style={styles.messageItem}>  
55             <div style={styles.messageText}>{message.text}</div>  
56             <div style={styles.messageActions}>  
57               <button style={styles.messageDelete}>Delete</button>  
58             </div>  
59           </div>  
60         ))}  
61       </div>  
62     </div>
```

```
mjs   JS requestController.js   JS app.js   JS index.js   .env   JS emailservice.js   JS usersController.js   TeacherMessagingPage.jsx   Requestjsx X ...  
frontend > src > pages > Teachers > Requestjsx > ...  
7  
8 const Requests = () => {  
9   const [requests, setRequests] = useState([ ]);  
10  
11  let { contextDetails } = useContext(contexts)  
12  console.log(ContextDetails)  
13 // const teacherId = localStorage.getItem("teacherId") || ""; // Get from local storage or auth context  
14  const teacherId = ContextDetails.TeacherId; // Get from local storage or auth context  
15  
16 useEffect(() => {  
17   if (teacherId) fetchRequests();  
18 // eslint-disable-next-line react-hooks/exhaustive-deps  
19 }, [teacherId]);  
20  
21 // ✅ Fetch requests assigned to the logged-in teacher  
22 const fetchRequests = async () => {  
23   try {  
24     const response = await axios.get(`http://localhost:4000/api/v1/requests/teacher/${teacherId}`);  
25     console.log(response)  
26     setRequests(response.data.requests);  
27   } catch (error) {  
28     console.error("Error fetching requests:", error);  
29   }  
30 };  
31  
32 // ✅ Handle request response (Accept/Reject/Query)  
33 const handleResponse = async (requestId, status) => {  
34   try {  
35     await axios.post(`http://localhost:4000/api/v1/requests/respond`, { requestId, status });  
36     fetchRequests(); // Refresh the request list  
37   } catch (error) {  
38     console.error("Error updating request:", error);  
39   }  
40 };  
41  
42 return (  
43   <div>
```

FRONTEND ROUTER

The screenshot shows a browser-based code editor with multiple tabs open. The active tab is 'App.jsx' under the 'js' folder. The code is a React component named 'App'. It imports various components and context providers from other files like 'TeamRegister', 'TeamDashboard', 'JoinTeamPage', etc. It uses the Context API to manage 'ContextDetails' and 'ContextData'. The component structure includes a Provider, Router, and several Route components for different pages such as Home, ChooseUser, AdminSignIn, StudentSignIn, TeacherSignIn, TeacherSignInForgotPassword, StudentSignInForgotPassword, AdminRegister, TeamSignIn, and TeamRegister. There are also sections for 'Information sections' and 'Announcement'.

```
File Edit Selection View Go Run Terminal Help < > MentorMenteeSystem App.jsx index.js
chemajs.js announcementRouter.js eventsRouter.js app.js fileupload.js Announcement.jsx EventCalendar.jsx SettingsProfile.jsx
frontend > src > App.jsx ...
61 // Import Team Pages
62 import TeamRegister from "./pages/Teams/TeamRegister";
63 import TeamDashboard from "./pages/teams/teamDashboard";
64 import JointeamPage from "./pages/teams/jointeamPage";
65 import TeamDetails from "./pages/Teams/teamDetails";
66 import TPerformance from "./pages/teams/TPerformance";
67 import TeamleaderLogin from "./components/TeamleaderLogin.jsx";
68 import JoinTeam from "./pages/students/JoinTeam.jsx";
69 import contexts from "./components/contextApi.jsx";
70
Tabnine | Edit | Explain
const App = () => {
  const [teachers, setTeachers] = useState([]);
  let [ContextDetails, setContextDetails] = useState({})

  return (
    <contexts.Provider value={{ ContextDetails, setContextDetails }}>
      <Router future={{ v7_startTransition: true, v7_relativesplatPath: true }}>
        <Routes>
          <Route path="/" element={<Home />} />
          <Route path="/choose-user" element={<ChooseUser />} />
          /* All the sign-in pages/routes */
          <Route exact path="/admin-signin" element={<AdminSignIn />} />
          <Route exact path="/student-signIn" element={<StudentSignIn />} />
          <Route exact path="/teacher-signIn" element={<TeacherSignIn />} />
          <Route exact path="/teacher-signIn/forgotpassword" element={<ForgotPassword />} />
          <Route exact path="/student-signIn/forgotpassword" element={<ForgotPassword />} />
          <Route exact path="/admin/register" element={<AdminRegister />} />
          <Route exact path="/Team-signIn" element={<TeamleaderLogin />} />
          /* Information sections */

```

LOGGED IN USER DETAILS

The screenshot shows a code editor with several tabs open, including `eventsSchema.js`, `announcementRouter.js`, `eventsRouter.js`, `app.js`, `fileupload.js`, `Announcement.jsx`, `EventCalendar.jsx`, `SettingsProfile.jsx`, and `index.js`. The active file is `SettingsProfile.jsx`. The code handles user login and profile fetching:

```

frontend > src > pages > Admin > SettingsProfile.jsx > SettingsProfile
16 const SettingsProfile = () => {
17   const handleLogin = async () => {
18     const response = await fetch("/admin/login", {
19       headers: { "Content-Type": "application/json" },
20       body: JSON.stringify({ email })
21     });
22
23     const data = await response.json();
24     if (data.admin) {
25       //localStorage.setItem("admin", JSON.stringify(data.admin));
26       //navigate("./admin/profile"); // Redirect to profile page
27     } else {
28       alert("Unauthorized access");
29     }
30   };
31
32   useEffect(() => {
33     if (!adminEmail) {
34       console.error("No email found in localStorage.");
35       return;
36     }
37
38     const fetchProfile = async () => {
39       try {
40         const res = await axios.get(`http://localhost:4000/api/v1/users/admin/profile?email=${adminEmail}`);
41         console.log("Profile Data from API:", res.data);
42         setProfile(res.data.admin);
43       } catch (error) {
44         console.error("Error fetching profile:", error);
45       }
46     };
47   });
48
49   const adminEvents = () => {
50     const handleAddEvent = async (e) => {
51       e.preventDefault();
52       if (!newEvent.trim()) return;
53
54       try {
55         const response = await axios.post('http://localhost:4000/api/v1/events', {
56           event: newEvent,
57           date: eventDate,
58         });
59
60         if (response.data.success) {
61           setEvents([...events, response.data.event]);
62           setNewEvent("");
63           setEventDate(new Date());
64         } else {
65           console.error("Failed to add event:", response.data.message);
66         }
67       } catch (error) {
68         console.error("Error adding event:", error);
69       }
70     };
71
72     const handleDeleteEvent = async (id) => {
73       try {
74         const response = await axios.delete(`http://localhost:4000/api/v1/events/${id}`);
75
76         if (response.data.success) {
77           setEvents(events.filter((event) => event._id !== id));
78         } else {
79           console.error("Failed to delete event:", response.data.message);
80         }
81       } catch (error) {
82     }
83     };
84   };
85
86   return (
87     <div>
88       <h1>Settings Profile</h1>
89       <form>
90         <input type="text" value={adminEmail} />
91         <button type="button" onClick={handleLogin}>Login</button>
92       </form>
93       <table border="1">
94         <thead>
95           <tr>
96             <th>Event Name</th>
97             <th>Event Date</th>
98           </tr>
99         </thead>
100        <tbody>
101          {events.map((event) => (
102            <tr>
103              <td>{event.event}</td>
104              <td>{event.date}</td>
105              <td><button type="button" onClick={()=>handleDeleteEvent(event._id)}>Delete</button></td>
106            </tr>
107          ))}
108        </tbody>
109      </table>
110    </div>
111  );
112}

```

EVENT FRONTEND

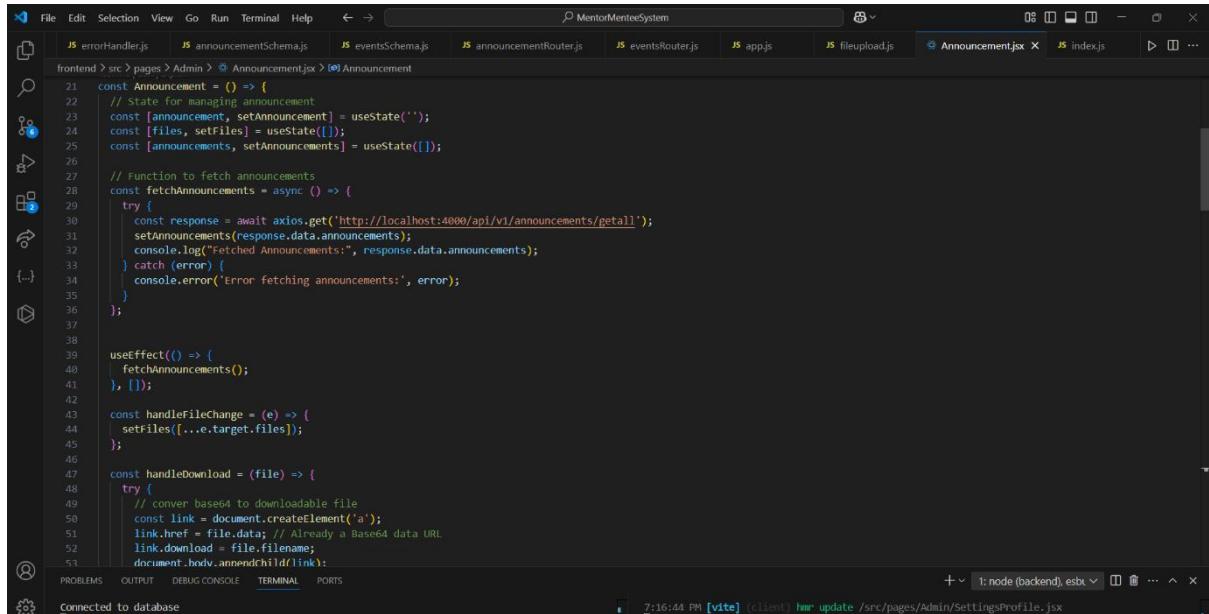
The screenshot shows a code editor with several tabs open, including `announcementSchema.js`, `eventsSchema.js`, `announcementRouter.js`, `eventsRouter.js`, `app.js`, `fileupload.js`, `Announcement.jsx`, `EventCalendar.jsx`, `index.js`, and `index.js`. The active file is `EventCalendar.jsx`. The code handles event creation and deletion:

```

frontend > src > pages > Admin > EventCalendar.jsx > AdminEvents > useEffect() callback > fetchData
16 const AdminEvents = () => {
17   const handleAddEvent = async (e) => {
18     e.preventDefault();
19     if (!newEvent.trim()) return;
20
21     try {
22       const response = await axios.post('http://localhost:4000/api/v1/events', {
23         event: newEvent,
24         date: eventDate,
25       });
26
27       if (response.data.success) {
28         setEvents([...events, response.data.event]);
29         setNewEvent("");
30         setEventDate(new Date());
31       } else {
32         console.error("Failed to add event:", response.data.message);
33       }
34     } catch (error) {
35       console.error("Error adding event:", error);
36     }
37   };
38
39   const handleDeleteEvent = async (id) => {
40     try {
41       const response = await axios.delete(`http://localhost:4000/api/v1/events/${id}`);
42
43         if (response.data.success) {
44           setEvents(events.filter((event) => event._id !== id));
45         } else {
46           console.error("Failed to delete event:", response.data.message);
47         }
48       } catch (error) {
49     }
50   };
51
52   const events = [
53     {
54       _id: "1",
55       event: "React Meetup",
56       date: "2023-10-15T18:00:00Z"
57     },
58     {
59       _id: "2",
60       event: "Node.js Workshop",
61       date: "2023-10-16T10:00:00Z"
62     }
63   ];
64
65   const newEvent = useState("");
66   const eventDate = useState(new Date());
67
68   const [events, setEvents] = useState(events);
69   const [newEvent, setNewEvent] = useState("");
70   const [eventDate, setEventDate] = useState(new Date());
71
72   const handleFormSubmit = (e) => {
73     e.preventDefault();
74     handleAddEvent();
75   };
76
77   return (
78     <div>
79       <h1>Event Calendar</h1>
80       <form onSubmit={handleFormSubmit}>
81         <input type="text" value={newEvent} />
82         <input type="date" value={eventDate} />
83         <button type="submit">Add Event</button>
84       </form>
85       <table border="1">
86         <thead>
87           <tr>
88             <th>Event Name</th>
89             <th>Event Date</th>
90           </tr>
91         </thead>
92         <tbody>
93           {events.map((event) => (
94             <tr>
95               <td>{event.event}</td>
96               <td>{event.date}</td>
97               <td><button type="button" onClick={()=>handleDeleteEvent(event._id)}>Delete</button></td>
98             </tr>
99           ))}
100         </tbody>
101       </table>
102     </div>
103   );
104 }

```

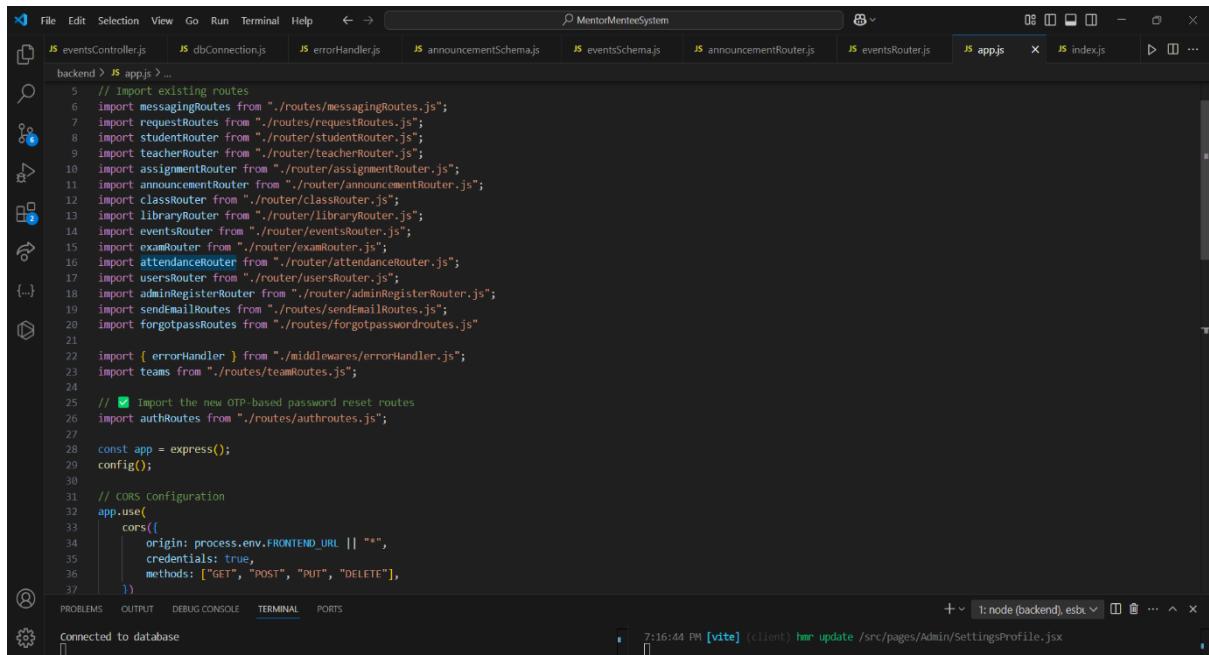
ANNOUNCEMENT FRONTEND



A screenshot of a code editor showing the `Announcement` component code. The code uses React hooks like `useState` and `useEffect` to manage state and fetch data from a backend API. It includes file upload logic and a download handler. The code editor interface shows tabs for other files like `errorHandler.js`, `announcementSchema.js`, etc.

```
frontend > src > pages > Admin > Announcement.js
1 const Announcement = () => {
2   // State for managing announcement
3   const [announcement, setAnnouncement] = useState('');
4   const [files, setFiles] = useState([]);
5   const [announcements, setAnnouncements] = useState([]);
6
7   // Function to fetch announcements
8   const fetchAnnouncements = async () => {
9     try {
10       const response = await axios.get('http://localhost:4000/api/v1/announcements/getall');
11       setAnnouncements(response.data.announcements);
12       console.log("Fetched Announcements:", response.data.announcements);
13     } catch (error) {
14       console.error('Error fetching announcements:', error);
15     }
16   };
17
18   useEffect(() => {
19     fetchAnnouncements();
20   }, []);
21
22   const handleFileChange = (e) => {
23     setFiles([...e.target.files]);
24   };
25
26   const handleDownload = (file) => {
27     try {
28       // Convert base64 to downloadable file
29       const link = document.createElement('a');
30       link.href = file.data; // Already a Base64 data URL
31       link.download = file.filename;
32       link.click();
33     } catch (error) {
34       console.error('Error downloading file:', error);
35     }
36   };
37
38   useEffect(() => {
39     handleDownload(files[0]);
40   }, [files]);
41
42   const handleDelete = (id) => {
43     fetch(`http://localhost:4000/api/v1/announcements/delete/${id}`, {
44       method: 'DELETE',
45     })
46     .then((response) => {
47       if (response.ok) {
48         setAnnouncements(announcements.filter((a) => a.id !== id));
49       }
50     })
51     .catch((error) => {
52       console.error('Error deleting announcement:', error);
53     });
54   };
55
56   return (
57     <div>
58       <h2>Announcements</h2>
59       <table>
60         <thead>
61           <tr>
62             <th>Title</th>
63             <th>Actions</th>
64           </tr>
65         </thead>
66         <tbody>
67           {announcements.map((a) => (
68             <tr key={a.id}>
69               <td>{a.title}</td>
70               <td>
71                 <button onClick={()=>handleDelete(a.id)}>Delete</button>
72               </td>
73             </tr>
74           ))}
75         </tbody>
76       </table>
77       <br/>
78       <input type="file" onChange={handleFileChange} />
79       <button onClick={()=>handleDownload(files[0])}>Download</button>
80     </div>
81   );
82 }
83
84 export default Announcement;
```

BACKEND ROUTER



A screenshot of a code editor showing the `app.js` file for the backend router. The code imports various route files and sets up the Express app with CORS configuration. It also includes a section for OTP-based password reset routes.

```
backend > JS app.js > ...
1 // Import existing routes
2 import messagingRoutes from './routes/messagingRoutes.js';
3 import requestRoutes from './routes/requestRoutes.js';
4 import studentRouter from './router/studentRouter.js';
5 import teacherRouter from './router/teacherRouter.js';
6 import assignmentRouter from './router/assignmentRouter.js';
7 import announcementRouter from './router/announcementRouter.js';
8 import classRouter from './router/classRouter.js';
9 import libraryRouter from './router/libraryRouter.js';
10 import eventsRouter from './router/eventsRouter.js';
11 import examRouter from './router/examRouter.js';
12 import attendanceRouter from './router/attendanceRouter.js';
13 import usersRouter from './router/usersRouter.js';
14 import adminRegisterRouter from './router/adminRegisterRouter.js';
15 import sendEmailRoutes from './routes/sendEmailRoutes.js';
16 import forgotPasswordRoutes from './routes/forgotPasswordRoutes.js';
17
18 import { errorHandler } from './middlewares/errorHandler.js';
19 import teams from './routes/teamRoutes.js';
20
21 // Import the new OTP-based password reset routes
22 import authRoutes from './routes/authRoutes.js';
23
24 const app = express();
25 config();
26
27 // CORS configuration
28 app.use(
29   cors({
30     origin: process.env.FRONTEND_URL || "*",
31     credentials: true,
32     methods: ["GET", "POST", "PUT", "DELETE"],
33   })
34 );
35
36 app.use('/api', [
37   announcementRouter,
38   assignmentRouter,
39   classRouter,
40   examRouter,
41   eventsRouter,
42   messagingRoutes,
43   messageRouter,
44   requestRoutes,
45   studentRouter,
46   teacherRouter,
47   usersRouter,
48   adminRegisterRouter,
49   sendEmailRoutes,
50   forgotPasswordRoutes,
51   authRoutes,
52   errorHandler,
53   teams,
54 ]);
55
56 module.exports = app;
```

EVENT ROUTER

The screenshot shows a browser-based code editor interface. The top navigation bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a search bar labeled "MentorMenteeSystem". Below the navigation bar is a toolbar with various icons: a file icon, a magnifying glass, a person icon, a star icon, a gear icon, a dropdown arrow, and a plus sign. The main workspace displays a file named "errorHandler.js" with the following code:

```
backend > middlewares > JS errorHandler.js > ...
1 // errorhandler.js
2
3 Tabnine | Edit | Explain
4 export const handleValidationErrors = (message, statusCode) => {
5   const error = new Error(message);
6   error.statusCode = statusCode;
7   throw error;
8 }
9 Tabnine | Edit | Explain
10 export const errorHandler = (err, req, res, next) => {
11   console.error(err.stack);
12
13   if (res.headersSent) {
14     return next(err); // ✅ Prevent sending headers again
15   }
16
17   const statusCode = err.statusCode || 500;
18   const message = err.message || 'Internal Server Error';
19   res.status(statusCode).json({ success: false, message });
20 }
```

At the bottom of the editor, there are tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is currently active, showing the command "node (backend).esb". The status bar at the bottom right shows the time as 7:16:44 PM and the message "[vite] (client) hmr update /src/pages/Admin/SettingsProfile.jsx".

DATABASE CONNECTION



The screenshot shows a code editor with the following file structure:

- File: announcementController.js
- File: eventsController.js
- File: dbConnection.js (active tab)

The dbConnection.js file contains the following code:

```
import mongoose from "mongoose";
export const dbConnection = () => {
  mongoose.connect(process.env.MONGO_URI, {
    dbName: "M5db",
  })
  .then(() => {
    console.log("Connected to database");
  })
  .catch((error) => {
    console.log("Error occurred while connecting to database", error.message);
  });
};
```

The code uses the Mongoose library to connect to a MongoDB database named "M5db". It logs a success message to the console if the connection is successful and logs an error message if it fails.

EVENT CONTROLLER

ANNOUNCEMENT CONTROLLER

CSV UPLOAD FILE

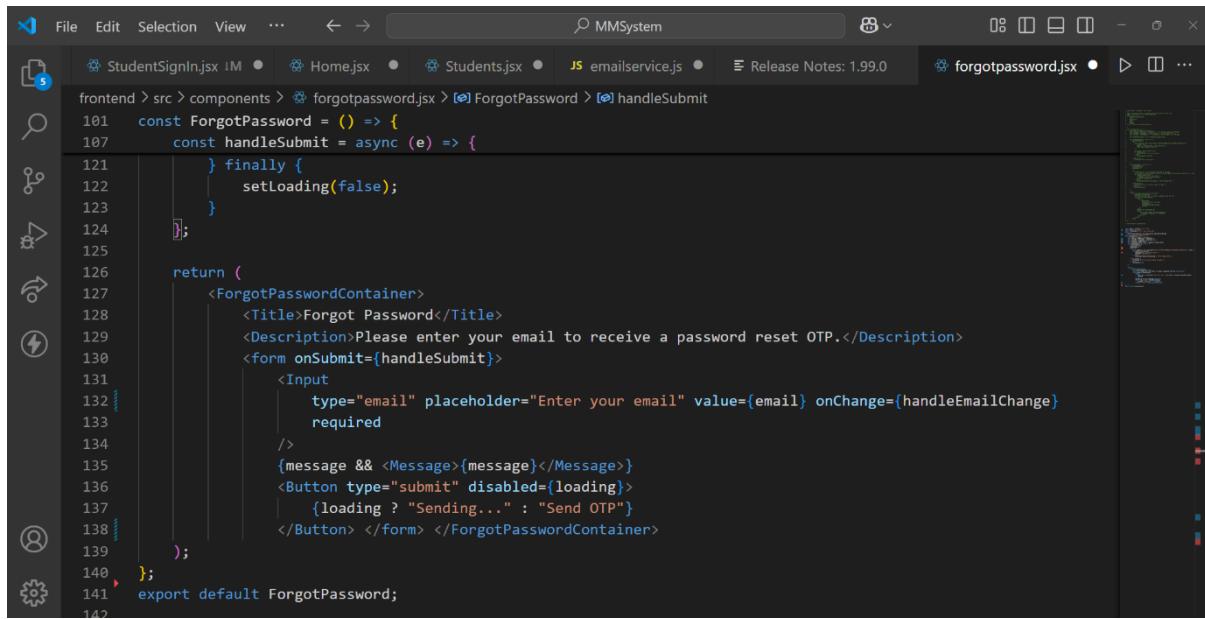
A screenshot of the Visual Studio Code interface. The title bar says "MentorMenteeSystem". The left sidebar shows a tree view of files under "MENTORMENTEEESYSTEM" and "backend". The main editor area has three tabs open: "studentController.js", "teacherController.js", and "studentRouter.js". The "studentRouter.js" tab is active, displaying code related to file uploads using Multer and disk storage. The code includes logic for handling CSV files and setting up file filters for PDF, PPT, and PPTX formats. The bottom status bar shows "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL", and "PORTS". A terminal tab is open with the command "1: node (backend).esbx".

```
studentRouter.js
backend > router > studentRouter.js > ...
13 // (Only Accept CSV)
14 const storage = multer.diskStorage({
15   destination: (req, file, cb) => {
16     cb(null, "uploads/");
17   },
18   filename: (req, file, cb) => {
19     cb(null, file.originalname);
20   }
21 });
22 //file upload
23 const store = multer.diskStorage({
24   destination: (req, file, cb) => {
25     const uploadDir = "uploads/students";
26     if (!fs.existsSync(uploadDir)) {
27       fs.mkdirSync(uploadDir, { recursive: true });
28     }
29     cb(null, uploadDir);
30   },
31   filename: (req, file, cb) => {
32     cb(null, file.originalname);
33   }
34 });
35 //file upload
36 const fileFilter = (req, file, cb) => {
37   const allowedTypes = [".pdf", ".ppt", ".pptx"];
38   const ext = path.extname(file.originalname).toLowerCase();
39   if (!allowedTypes.includes(ext)) {
40     return cb(new Error("Only PDF and PPT files are allowed"));
41   }
42   cb(null, true);
43 };
44 };
45 
```

FORGOT PASSWORD

A screenshot of the Visual Studio Code interface. The title bar says "MMSystem". The left sidebar shows a tree view of files under "frontend". The main editor area has five tabs open: "StudentSignin.jsx", "Home.jsx", "Students.jsx", "emailservice.js", and "forgotpassword.jsx". The "forgotpassword.jsx" tab is active, displaying code for a forgot password feature. It uses React, useState, and axios. The component handles user input for email, sends an OTP via axios post to the API, and navigates to the verification page. The code includes error handling for failed API requests.

```
forgotpassword.jsx
frontend > src > components > forgotpassword.jsx > ForgotPassword > handleSubmit
95 import React, { useState } from "react";
96 import axios from "axios";
97 import { useNavigate } from "react-router-dom";
98 import {
99   ForgotPasswordContainer, Title, Description, Input, Button, Message
100 } from "../styles/forgotpasswordstyles";
101 const ForgotPassword = () => {
102   const [email, setEmail] = useState("");
103   const [message, setMessage] = useState("");
104   const [loading, setLoading] = useState(false);
105   const navigate = useNavigate();
106   const handleEmailChange = (e) => setEmail(e.target.value);
107   const handleSubmit = async (e) => [
108     e.preventDefault();
109     setLoading(true);
110     setMessage("");
111     try {
112       const response = await axios.post('http://localhost:4000/api/v1/forgotpass/request-otp', { email });
113       if (response.data.success) {
114         setMessage("OTP sent to your email.");
115         navigate("/verify-otp");
116       } else {
117         setMessage(response.data.message || "Error sending OTP.");
118       }
119     } catch (error) {
120       console.error(error);
121       setMessage("An error occurred while sending the OTP.");
122     }
123   ];
124 }
```

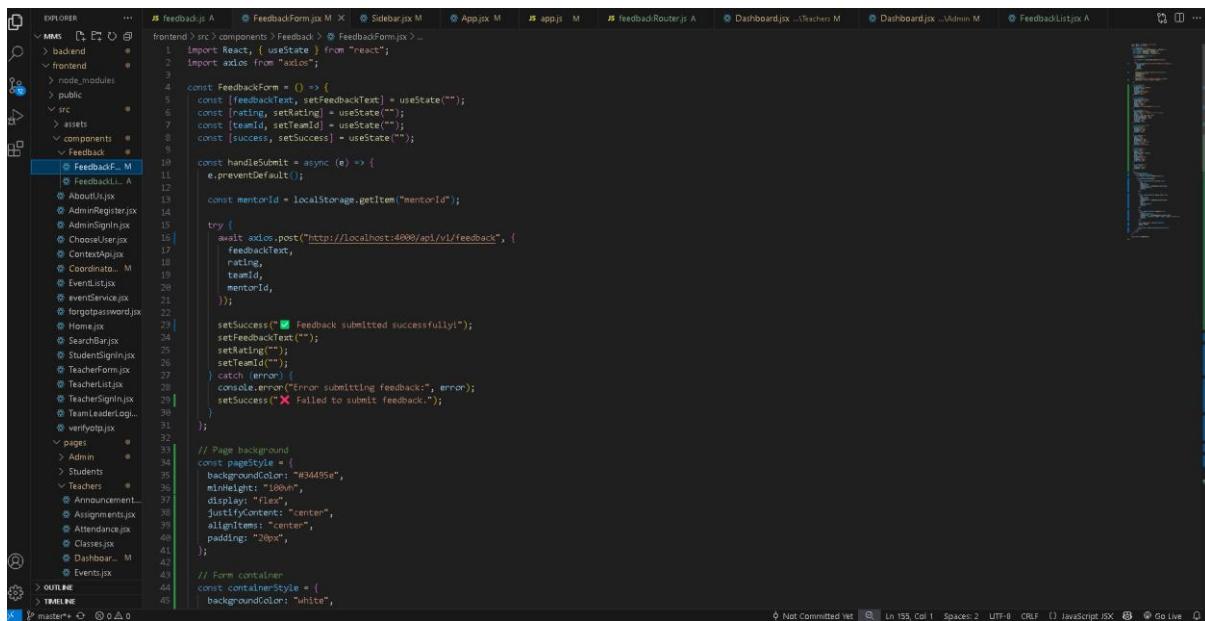


```

101  const ForgotPassword = () => {
107    const handleSubmit = async (e) => {
121      } finally {
122        setLoading(false);
123      }
124    };
125
126    return (
127      <ForgotPasswordContainer>
128        <Title>Forgot Password</Title>
129        <Description>Please enter your email to receive a password reset OTP.</Description>
130        <form onSubmit={handleSubmit}>
131          <Input
132            type="email" placeholder="Enter your email" value={email} onChange={handleEmailChange}
133            required
134          />
135          {message && <Message>{message}</Message>}
136          <Button type="submit" disabled={loading}>
137            {loading ? "Sending..." : "Send OTP"}
138          </Button> </form> </ForgotPasswordContainer>
139        );
140    };
141  export default ForgotPassword;
142

```

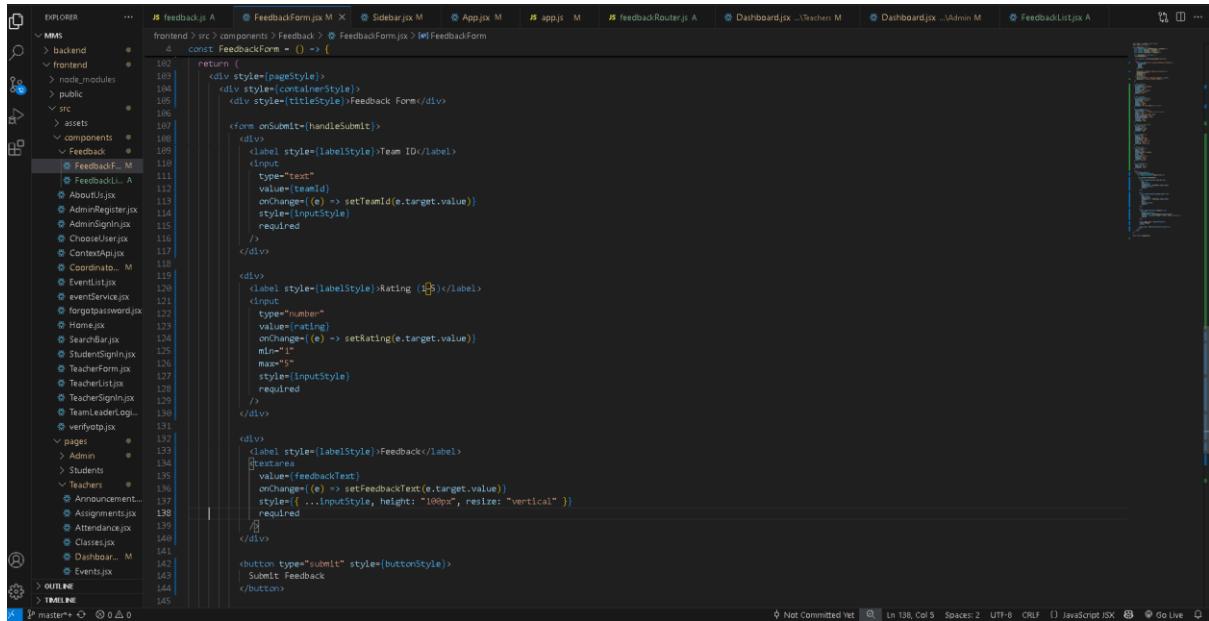
FEEDBACK FORM



```

1 import React, {useState} from "react";
2 import axios from "axios";
3
4 const FeedbackForm = () => {
5   const [feedbackText, setFeedbackText] = useState("");
6   const [rating, setRating] = useState("");
7   const [teamId, setTeamId] = useState("");
8   const [success, setSuccess] = useState("");
9
10  const handleSubmit = async (e) => {
11    e.preventDefault();
12
13    const mentorId = localStorage.getItem("mentorId");
14
15    try {
16      await axios.post("http://localhost:4000/api/v1/feedback", {
17        feedbackText,
18        rating,
19        teamId,
20        mentorId,
21      });
22
23      setSuccess("Feedback submitted successfully!");
24      setFeedbackText("");
25      setRating("");
26      setTeamId("");
27    } catch (error) {
28      console.error("Error submitting feedback:", error);
29      setSuccess("Failed to submit feedback.");
30    }
31  };
32
33  // Page background
34  const pageStyle = {
35    backgroundColor: "#34495e",
36    minHeight: "100vh",
37    display: "flex",
38    justifyContent: "center",
39    alignItems: "center",
40    padding: "20px",
41  };
42
43  // Form container
44  const containerStyle = {
45    backgroundColor: "white",
46    border: "1px solid #ccc",
47    padding: "10px",
48    width: "300px",
49    margin: "0 auto",
50  };
51
52  return (
53    <div style={pageStyle}>
54      <div style={containerStyle}>
55        <h3>Submit Feedback</h3>
56        <form>
57          <label>Feedback Text:</label>
58          <input type="text" value={feedbackText} onChange={setFeedbackText} />
59          <br/>
60          <label>Rating:</label>
61          <input type="text" value={rating} onChange={setRating} />
62          <br/>
63          <label>Team ID:</label>
64          <input type="text" value={teamId} onChange={setTeamId} />
65          <br/>
66          <button type="button" onClick={handleSubmit}>Submit</button>
67        </form>
68      </div>
69    </div>
70  );
71
72  export default FeedbackForm;
73

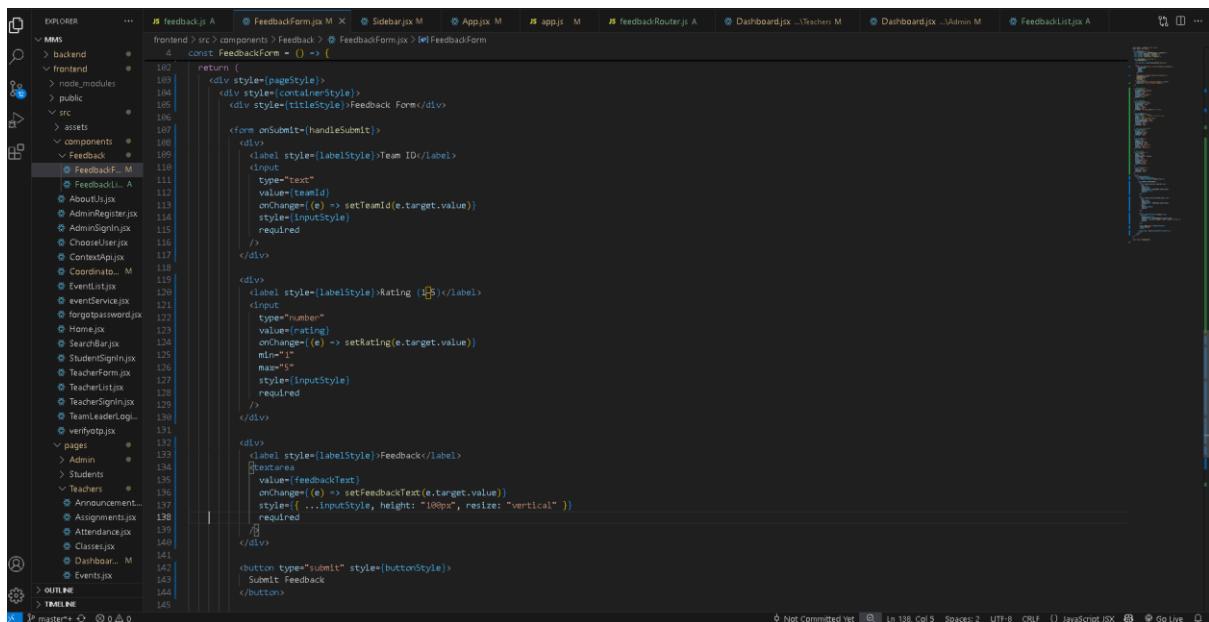
```



```
JS feedback.js A ① FeedbackForm.js M ② Sidebar.js M ③ App.js M ④ app.js M ⑤ feedbackRouter.js A ⑥ Dashboard.jsx -\Teachers M ⑦ Dashboard.jsx -\Admin M ⑧ FeedbackList.jsx A
```

```
frontend > src > components > Feedback > FeedbackForm.js > [FeedbackForm]
```

```
102 4 const FeedbackForm = () => {
103   return (
104     <div style={pageStyle}>
105       <div style={containerStyle}>
106         <div style={titleStyle}>Feedback Form</div>
107
108         <form onSubmit={HandleSubmit}>
109           <div>
110             <label style={labelStyle}>Team ID</label>
111             <input
112               type="text"
113               value={teamId}
114               onChange={(e) => setTeamId(e.target.value)}
115               style={inputStyle}
116               required
117             />
118           </div>
119
120           <div>
121             <label style={labelStyle}>Rating <input type="number" value={rating} onChange={(e) => setRating(e.target.value)} min="1" max="5" style={inputStyle} required/>
122           </div>
123
124           <div>
125             <label style={labelStyle}>Feedback</label>
126             <textarea
127               value={feedbackText}
128               onChange={(e) => setFeedbackText(e.target.value)}
129               style={{ ...inputStyle, height: "100px", resize: "vertical" }} required
130             />
131           </div>
132
133           <div>
134             <button type="submit" style={buttonStyle}>
135               Submit Feedback
136             </button>
137           </div>
138         </form>
139       </div>
140     </div>
141   );
142 }
143
144 <button type="submit" style={buttonStyle}>
145   Submit Feedback
146 </button>
```

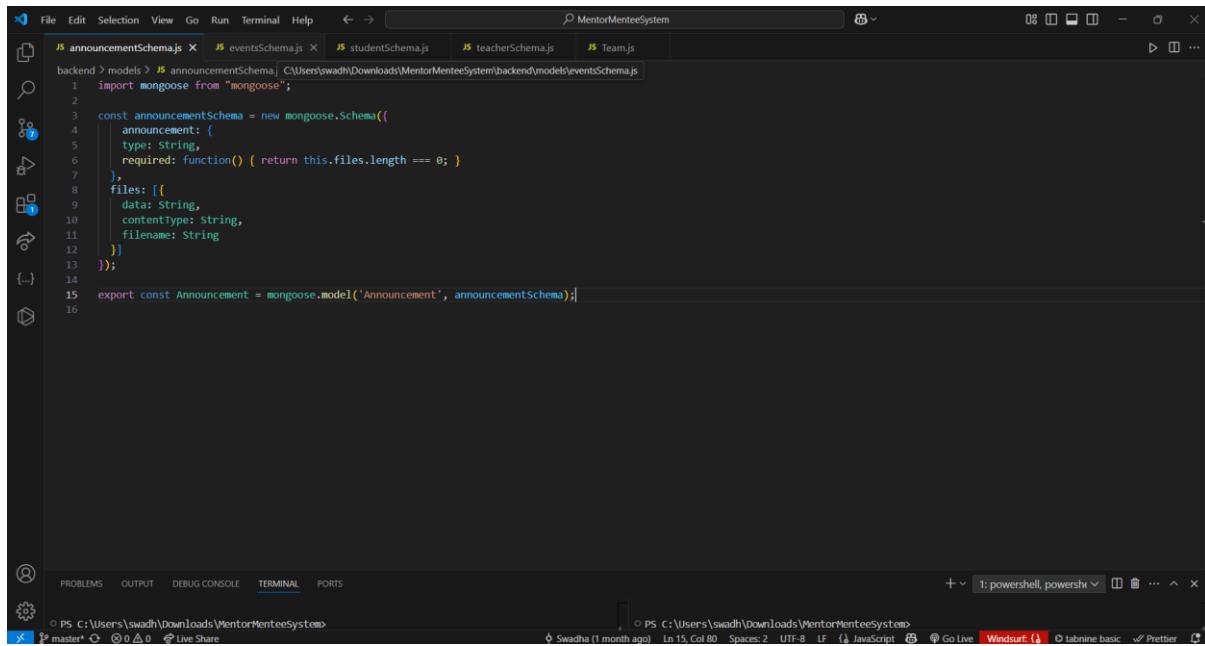


```
JS feedback.js A ① FeedbackForm.js M ② Sidebar.js M ③ App.js M ④ app.js M ⑤ feedbackRouter.js A ⑥ Dashboard.jsx -\Teachers M ⑦ Dashboard.jsx -\Admin M ⑧ FeedbackList.jsx A
```

```
frontend > src > components > Feedback > FeedbackForm.js > [FeedbackForm]
```

```
102 4 const FeedbackForm = () => {
103   return (
104     <div style={pageStyle}>
105       <div style={containerStyle}>
106         <div style={titleStyle}>Feedback Form</div>
107
108         <form onSubmit={HandleSubmit}>
109           <div>
110             <label style={labelStyle}>Team ID</label>
111             <input
112               type="text"
113               value={teamId}
114               onChange={(e) => setTeamId(e.target.value)}
115               style={inputStyle}
116               required
117             />
118           </div>
119
120           <div>
121             <label style={labelStyle}>Rating <input type="number" value={rating} onChange={(e) => setRating(e.target.value)} min="1" max="5" style={inputStyle} required/>
122           </div>
123
124           <div>
125             <label style={labelStyle}>Feedback</label>
126             <textarea
127               value={feedbackText}
128               onChange={(e) => setFeedbackText(e.target.value)}
129               style={{ ...inputStyle, height: "100px", resize: "vertical" }} required
130             />
131           </div>
132
133           <div>
134             <button type="submit" style={buttonStyle}>
135               Submit Feedback
136             </button>
137           </div>
138         </form>
139       </div>
140     </div>
141   );
142 }
143
144 <button type="submit" style={buttonStyle}>
145   Submit Feedback
146 </button>
```

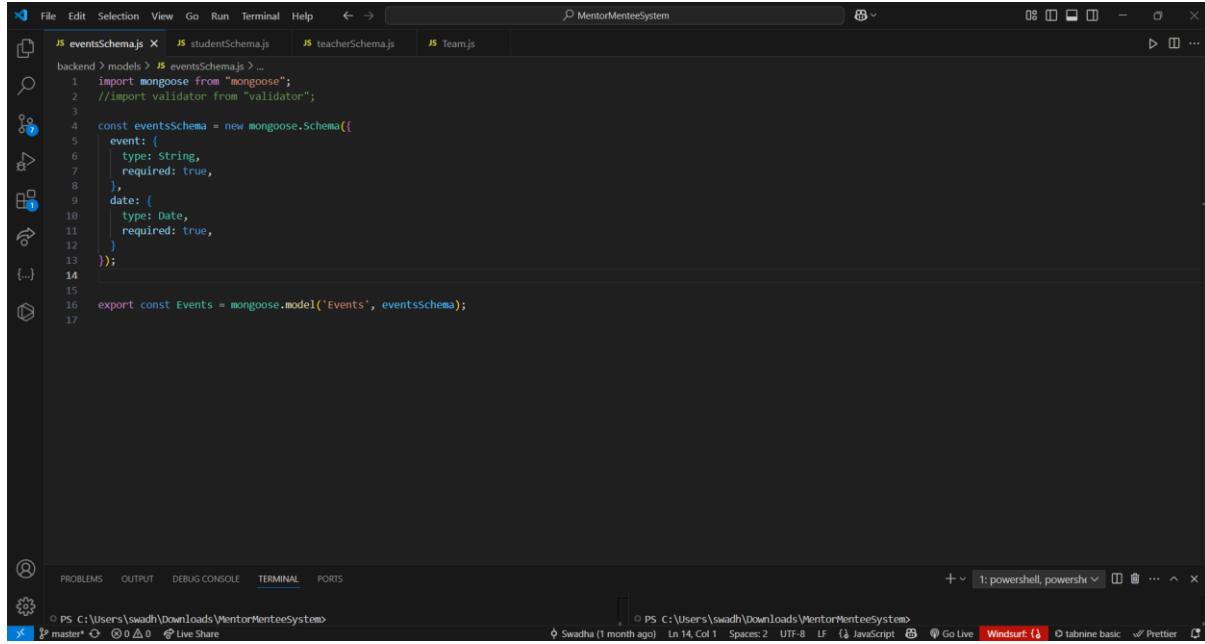
ANNOUNCEMENT SCHEMA



A screenshot of the Visual Studio Code interface. The title bar says "MentorMenteeSystem". The left sidebar shows files: announcementSchema.js, eventsSchema.js, studentSchema.js, teacherSchema.js, and Team.js. The announcementSchema.js tab is active. The code defines a mongoose schema for 'Announcement' with fields: announcement (type: string, required), files (array of objects with data, contentType, and filename), and a length check. It then exports the model. The bottom status bar shows a PowerShell terminal with the command "PS C:\Users\swad\Downloads\MentorMenteeSystem>".

```
1 import mongoose from "mongoose";
2
3 const announcementSchema = new mongoose.Schema({
4   announcement: {
5     type: String,
6     required: function() { return this.files.length === 0; }
7   },
8   files: [
9     {
10       data: String,
11       contentType: String,
12       filename: String
13     }
14   ],
15 });
16
17 export const Announcement = mongoose.model('Announcement', announcementSchema);
```

EVENT SCHEMA



A screenshot of the Visual Studio Code interface. The title bar says "MentorMenteeSystem". The left sidebar shows files: eventsSchema.js, studentSchema.js, teacherSchema.js, and Team.js. The eventsSchema.js tab is active. The code defines a mongoose schema for 'Events' with fields: event (type: string, required), and date (type: Date, required). It then exports the model. The bottom status bar shows a PowerShell terminal with the command "PS C:\Users\swad\Downloads\MentorMenteeSystem>".

```
1 import mongoose from "mongoose";
2 //import validator from "validator";
3
4 const eventsSchema = new mongoose.Schema({
5   event: {
6     type: String,
7     required: true,
8   },
9   date: {
10     type: Date,
11     required: true,
12   }
13 });
14
15
16 export const Events = mongoose.model('Events', eventsSchema);
```

TEAM SCHEMA

The screenshot shows the VS Code interface with the Team.js file open. The code defines a Team schema using Mongoose. It includes a password generation function, a studentId field, and a course field. The Team schema itself has fields for teamname, students, teamid, and password.

```
function generatePassword(length = 8) {
  const chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$%^&*";
  let password = '';
  for (let i = 0; i < length; i++) {
    password += chars.charAt(Math.floor(Math.random() * chars.length));
  }
  return password;
}

const StudentSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
    trim: true
  },
  email: {
    type: String,
    required: true,
    trim: true
  },
  course: {
    type: String,
    required: true,
    trim: true
  },
  studentId: {
    type: String,
    required: true,
    trim: true
  }
});

const TeamSchema = new mongoose.Schema({
  teamname: {
    type: String,
    required: [true, "Please provide a Unique Name"],
    trim: true
  },
  students: [StudentSchema],
  teamid: {
    type: String,
    required: true,
    unique: true
  },
  password: {
    type: String,
    required: true,
    minlength: [8, "Password must be at least 8 characters long"],
    default: () => generatePassword(10) // Generating password with 10 characters
  }
});
```

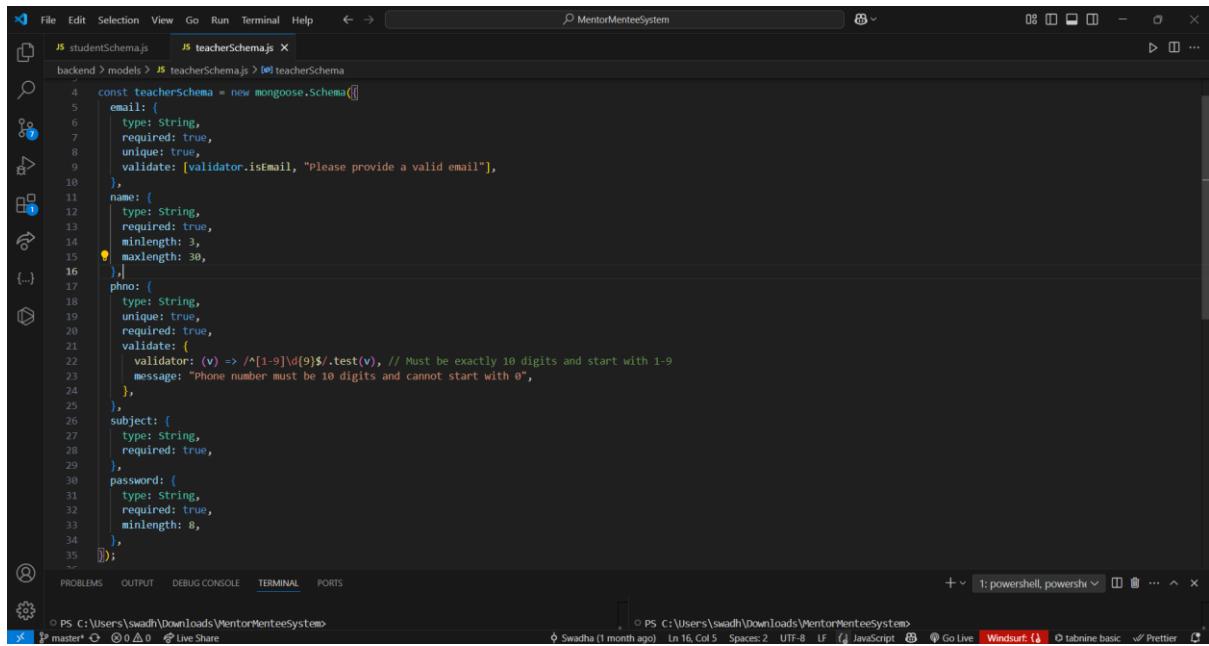
The screenshot shows the VS Code interface with the Team.js file open. The code defines a Team schema using Mongoose. It includes a password generation function, a studentId field, and a course field. The Team schema itself has fields for teamname, students, teamid, and password.

```
const StudentSchema = new mongoose.Schema({
  studentId: {
    type: String,
    required: true
  }
});

const TeamSchema = new mongoose.Schema({
  teamname: {
    type: String,
    required: [true, "Please provide a Unique Name"],
    trim: true
  },
  students: [StudentSchema],
  teamid: {
    type: String,
    required: true,
    unique: true
  },
  password: {
    type: String,
    required: true,
    minlength: [8, "Password must be at least 8 characters long"],
    default: () => generatePassword(10) // Generating password with 10 characters
  }
});

// Create model
const Team = mongoose.model("Team", TeamSchema);
export default Team;
```

TEACHER SCHEMA



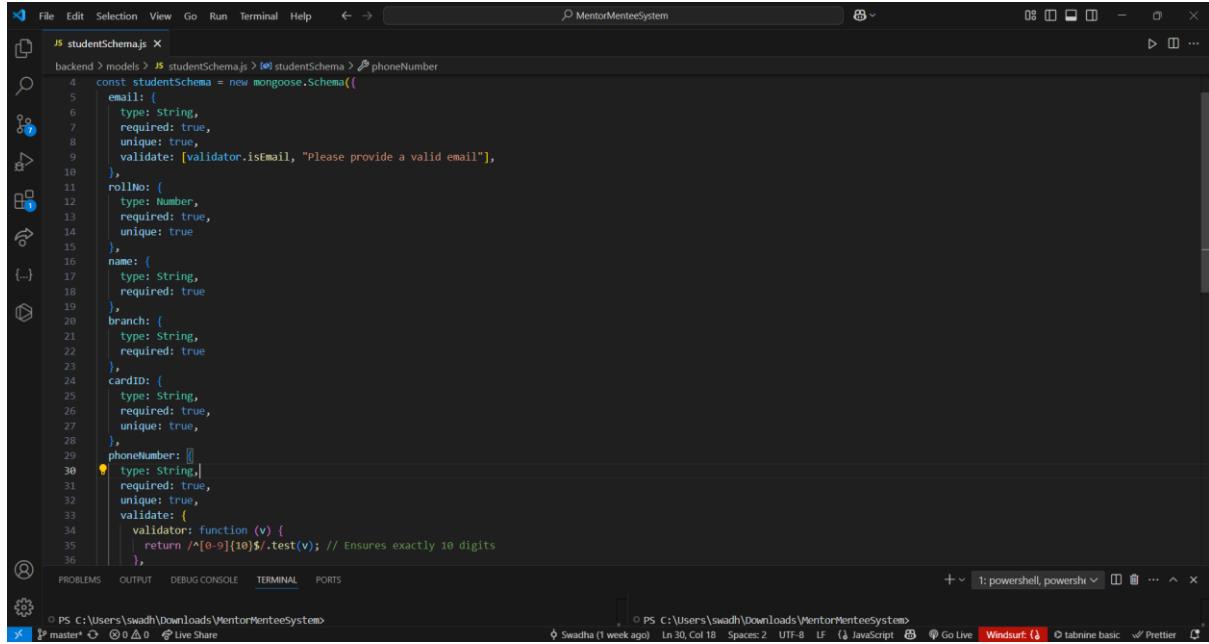
```
File Edit Selection View Go Run Terminal Help ⏪ ⏴ MentorMenteeSystem
studentSchema.js teacherSchema.js
backend > models > teacherSchema.js > teacherSchema.js
4 const teacherSchema = new mongoose.Schema({
5   email: {
6     type: String,
7     required: true,
8     unique: true,
9     validate: [validator.isEmail, "Please provide a valid email"],
10  },
11  name: {
12    type: String,
13    required: true,
14    minlength: 3,
15    maxlength: 30,
16  },
17  phone: {
18    type: String,
19    unique: true,
20    required: true,
21    validate: {
22      validator: (v) => /^[1-9]\d{9}$/.test(v), // Must be exactly 10 digits and start with 1-9
23      message: "Phone number must be 10 digits and cannot start with 0",
24    },
25  },
26  subject: {
27    type: String,
28    required: true,
29  },
30  password: {
31    type: String,
32    required: true,
33    minlength: 8,
34  },
35 });
~
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\swadhi\Downloads\MentorMenteeSystem

Swadhi (1 month ago) Ln 16, Col 5 Spaces: 2 UTF-8 LF JavaScript Go Live Windsurf tabnine basic Prettier

STUDENT SCHEMA



```
File Edit Selection View Go Run Terminal Help ⏪ ⏴ MentorMenteeSystem
studentSchema.js
backend > models > studentSchema.js > phoneNumber
4 const studentSchema = new mongoose.Schema({
5   email: {
6     type: String,
7     required: true,
8     unique: true,
9     validate: [validator.isEmail, "Please provide a valid email"],
10  },
11  rollNo: {
12    type: Number,
13    required: true,
14    unique: true
15  },
16  name: {
17    type: String,
18    required: true
19  },
20  branch: {
21    type: String,
22    required: true
23  },
24  cardID: {
25    type: String,
26    required: true,
27    unique: true,
28  },
29  phoneNumber: []
30  type: String,
31  required: true,
32  unique: true,
33  validate: {
34    validator: function (v) {
35      return /^[0-9]{10}$/.test(v); // Ensures exactly 10 digits
36    },
37  },
~
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\swadhi\Downloads\MentorMenteeSystem

Swadhi (1 week ago) Ln 30, Col 18 Spaces: 2 UTF-8 LF JavaScript Go Live Windsurf tabnine basic Prettier

TESTING

- WEBSITE

1. Unit Testing

This test is applied on each of the module to find whether or not each module is properly working or not.

2. Integration Testing

After module cleared the unit testing then modules are tested for their working all together in the integrated testing phase.

3. Acceptance Testing

This testing provides the final assurance that the web application needed all behavioural and performance requirements.

With the testing, the working databases are attached.

STUDENT DETAILS

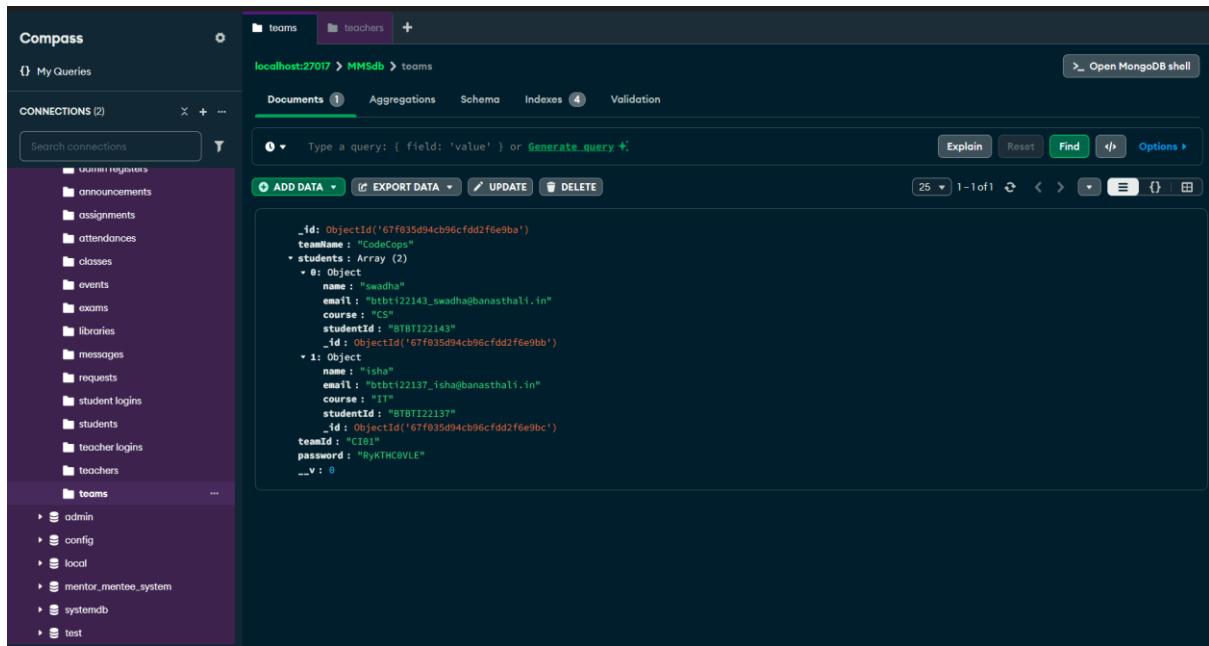
The screenshot shows the Compass MongoDB interface. On the left, there's a sidebar with a tree view of database connections and collections. Under 'My Queries', there are two connections: 'Compass' and 'MMSdb'. Under 'MMSdb', there are collections: 'announcements', 'assignments', 'classes', 'events', 'exams', 'libraries', 'messages', 'requests', 'student logins', 'students', 'teacher logins', 'teachers', and 'teams'. The 'students' collection is selected, and its details are shown on the right. The interface includes tabs for 'Documents', 'Aggregations', 'Schema', 'Indexes', and 'Validation'. A search bar at the top says 'localhost:27017 > MMSdb > students'. Below the search bar, there's a query builder with the placeholder 'Type a query: { field: 'value' } or Generate query'. There are buttons for 'EXPLAIN', 'Reset', 'Find', 'Options', and 'Open MongoDB shell'. The main area displays three documents from the 'students' collection:

```
_id: ObjectId('67e69ea265ae4b04f6415ade')
email: "btbit22143_swadha@banasthali.in"
rollNo: 2216913
name: "Swadha Sri"
branch: "IT"
cardID: "btbit22143"
phoneNumber: "9566115880"
password: "swagger123"
__v: 0

_id: ObjectId('67e69ea265ae4b04f6415ae0')
email: "btbit22141_himani@banasthali.in"
rollNo: 2216838
name: "Himani Dobreyal"
branch: "CS"
cardID: "btbit22141"
phoneNumber: "102345678"
password: "Himani@123"
__v: 0

_id: ObjectId('67e69ea265ae4b04f6415ae0')
email: "btbit22137_kisha@banasthali.in"
rollNo: 2216831
name: "Kisha"
branch: "S4"
cardID: "btbit22141"
phoneNumber: "7894560123"
password: "Kisha_0_0"
__v: 0
```

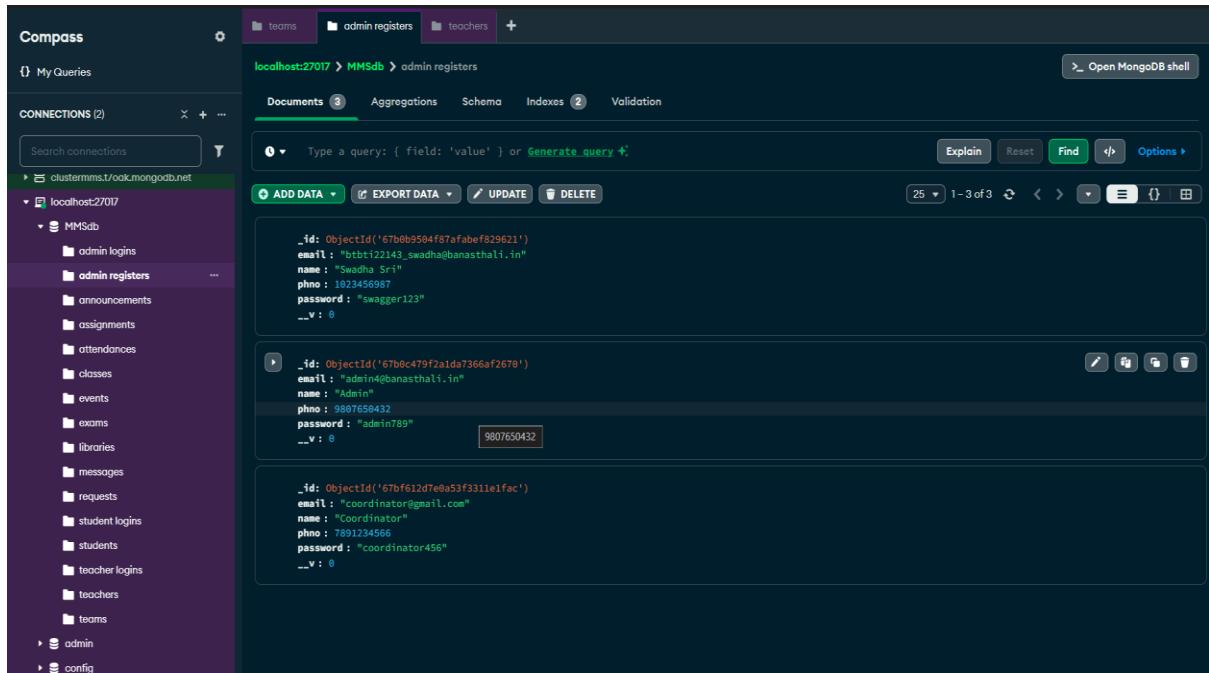
TEAM DETAILS



The screenshot shows the Compass MongoDB interface. The left sidebar lists connections, including 'MMSdb' which is currently selected. The main area displays the 'teams' collection with one document. The document structure is as follows:

```
_id: ObjectId('67f035d94cb96cfdd2f6e9ba')
teamName: "CodeCops"
students: [
  {
    _id: ObjectId('67f035d94cb96cfdd2f6e9bb')
    name: "swadha"
    email: "btbit122143_swadha@banasthali.in"
    course: "CS"
    studentId: "BTBTI22143"
  },
  {
    _id: ObjectId('67f035d94cb96cfdd2f6e9bc')
    name: "isha"
    email: "btbit122137_isha@banasthali.in"
    course: "IT"
    studentId: "BTBTI22137"
  }
]
teamId: "C101"
password: "ByKTHC0VLE"
__v: 0
```

ADMIN DETAILS



The screenshot shows the Compass MongoDB interface. The left sidebar lists connections, including 'localhost:27017' which is currently selected. The main area displays the 'admin registers' collection with three documents. The documents are as follows:

```
_id: ObjectId('67b0b9504f87afabef829621')
email: "btbit122143_swadha@banasthali.in"
name: "Swadha Sri"
phone: 1023456987
password: "swagger123"
__v: 0

_id: ObjectId('67b0c479f2a1da7366af2670')
email: "admin@banasthali.in"
name: "Admin"
phone: 98076568432
password: "admin789"
__v: 0

_id: ObjectId('67bf612d7e0a53f3311efac')
email: "coordinator@gmail.com"
name: "Coordinator"
phone: 7891234566
password: "coordinator456"
__v: 0
```

ANNOUNCEMENTS

The screenshot shows the Compass MongoDB interface. The left sidebar lists connections, including 'localhost:27017' and 'MMSSdb'. Under 'MMSSdb', the 'announcements' collection is selected. The main panel displays the 'announcements' collection with 2 documents. One document is shown in full:

```
_id: ObjectId('67e55d90f1da985c08491d8a')
announcement: "I want everyone to be present in uniform along with ID card"
__v: 0

_id: ObjectId('67eade54b0e05e745323c7c')
announcement: "So these are some files"
files: Array (3)
  0: Object
    data: "data:application/vnd.openxmlformats-officedocument.presentationml.presentation"
    contentType: "application/vnd.openxmlformats-officedocument.presentationml.presentation"
    filename: "ConversionToClauseForm.pptx"
    _id: ObjectId('67eade54b0e05e745323c7d')
  1: Object
    data: "data:application/pdf;base64,JVBERi0xLjcNCiW1tbW1DQoxIDAgb2JqDQo8PC9UeX_"
    contentType: "application/pdf"
    filename: "LabHandout_B_Tech_CSE_IT_VI_Sem_CS317L_AI&MLab.pdf"
    _id: ObjectId('67eade54b0e05e745323c7e')
  2: Object
    data: "data:application/pdf;base64,JVBERi0xLjcNCiW1tbW1DQoxIDAgb2JqDQo8PC9UeX_"
    contentType: "application/pdf"
    filename: "TheoryHandout_B_Tech_CSE_IT_VI_Sem_CS317_AI&ML.pdf"
    _id: ObjectId('67eade54b0e05e745323c7f')
__v: 0
```

EVENTS

The screenshot shows the Compass MongoDB interface. The left sidebar lists connections, including 'localhost:27017' and 'MMSSdb'. Under 'MMSSdb', the 'events' collection is selected. The main panel displays the 'events' collection with 4 documents. One document is shown in full:

```
_id: ObjectId('67e5ad4c4e1f76fbff4ada91')
event: "Khadi is Compulsory"
__v: 0

_id: ObjectId('67e5ad594e1f76fbff4ada94')
event: "Project Demonstration"
__v: 0

_id: ObjectId('67eb81baed50800abd50900884')
event: "Reload"
date: 2025-04-18T06:03:03.000+00:00
__v: 0

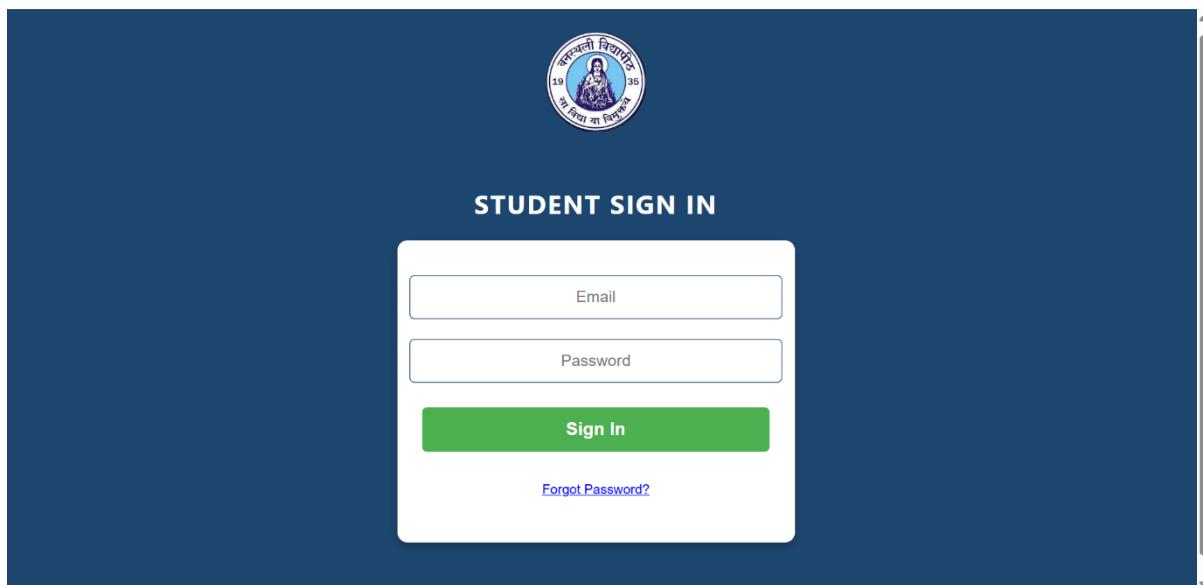
_id: ObjectId('67eb81b3d50800abd50900886')
event: "now"
date: 2025-03-03T06:03:26.000+00:00
__v: 0
```

USER INTERFACES

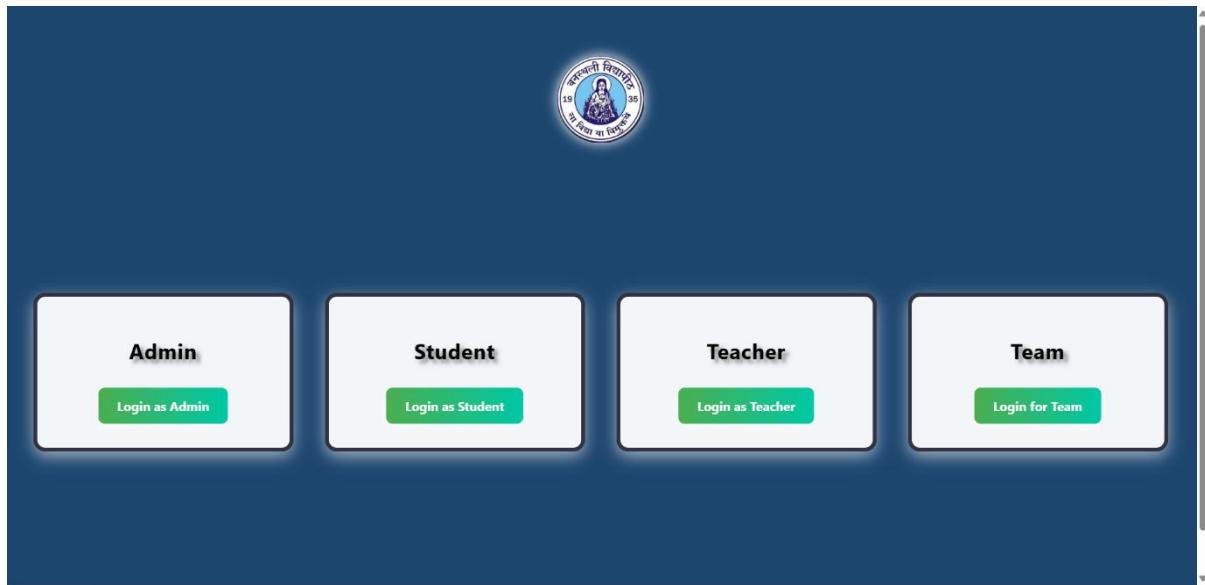
HOME PAGE –



LOGIN PAGE –



CHOOSING THE USER



SIGN IN MAIL

Welcome to Mentor-Mentee Portal (Student) [External](#) [Inbox x](#)

 Mentor-Mentee Portal <mentor.mentee.banasthali@gmail.com>
to me ▾ 8:27PM (6 minutes ago) [star](#) [left arrow](#)

BANASTHALI VIDYAPEETH

Welcome to MENTOR MENTEE SYSTEM

You have successfully logged in as a Student.

[Go to Portal](#)

You are logged in as a **Student**.

Email: deanadmin@banasthali.ac.in
Contact: [01438-228456](tel:01438-228456)

© 2025 Banasthali Vidyapith. All rights reserved.

ABOUT PAGE

About Us

Welcome to our platform! We are committed to providing an immersive experience where technology meets innovation. Our goal is to foster growth, creativity, and excellence in every step of the journey.

Who We Are

We are a team of dedicated professionals passionate about empowering learners and professionals through technology, mentorship, and collaboration.

Our Vision

To build a community where knowledge, creativity, and futuristic technology drive success and innovation.

ABOUT COORDINATORS

Coordinator's Message



Dr. Neelam Sharma
Associate Professor
Computer Science
sharmaneelam27@gmail.com

I am honored to guide students through this mentor-mentee system. Let's grow together!

Dr. Deepak Kumar
Assistant Professor
Computer Science
deepakkumar@banasthali.in

Mentorship is the key to success. I am excited to be a part of your journey!

TEAM REGISTER

The screenshot shows a registration form titled "Register a Team". The "Team Name" field contains the placeholder "Enter Team Name". The "Team Members" section contains four input fields: "Member 1 Name" (placeholder "Enter Name"), "Member 1 Email" (placeholder "Enter Email"), "Course" (placeholder "Enter Course"), and "Member 1 ID" (placeholder "Enter ID").

Team Members	
Member 1 Name:	Enter Name
Member 1 Email:	Enter Email
Course:	Enter Course
Member 1 ID:	Enter ID

TEAM DASHBOARD

The dashboard has a sidebar with "TEAM" at the top, followed by "Dashboard", "Team Details", and "Logout". The main area shows "Team Details" with Team ID: 67f295c6ff876ebcafc992c5, Team Name: Tech Queens, and Leader: isha@example.com. Below this is a "Team Members" section with four cards:

Member	Course	Status
Isha	IT	Accepted
Himani	CS	Accepted
Ayushi	IT	Pending
Swadha	AI	Pending

MENTOR DASHBOARD

The screenshot shows the Mentor Dashboard interface. On the left is a dark sidebar with a logo and navigation links: Dashboard, Messaging, Requests, Events & Calendar, Teacher Profile, Settings & Profile, and Logout.

Overview

Announcements	Students	Events
1	4	0

Announcement

Announcement:

Send Announcement

Announcements

hello	<input checked="" type="checkbox"/>	<input type="checkbox"/>
-------	-------------------------------------	--------------------------

MENTOR REQUEST

The screenshot shows the Student Mentorship Requests section. It features a header "Student Mentorship Requests" and two cards below it.

Isha

Email: isha@example.com
Project: Hospital Management (Java)
Group ID: CSD060

Ayushi

Email: ayushi@example.com
Project: Hospital Management (Java)
Group ID: CSD060

Each card has three buttons: Accept (green), Reject (red), and Query (yellow).

JOIN TEAM DASHBOARD

The dashboard features a dark blue sidebar on the left with a circular logo at the top. Below the logo, the word "Student" is displayed. A vertical list of navigation items includes: Dashboard (with a bar chart icon), Messaging (with a message icon), Events (with a calendar icon), Announcement (with a speech bubble icon), Team (with a team icon), Profile (with a user icon), and Logout (with a power-off icon). To the right of the sidebar, the main area is titled "Matching Teams" and displays the message "No teams found." Below this, a white rectangular box contains the "Join Team" form. The form has a title "Join Team" at the top, followed by a text input field labeled "Enter Join Link" and a large green "Join" button.

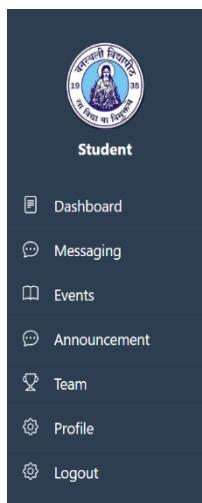
JOIN TEAM MAIL

An email inbox interface is shown. At the top, there's a header with the recipient's name "Join Team MentorMenteesystem" and an "External" link. Below the header, the email details are visible: the sender's name "ishakaliraman8@gmail.com", the recipient "to me", the timestamp "12:19 PM (7 hours ago)", and some standard email icons for printing, sharing, and more. The email body itself is titled "Join Our Team" and contains the following text:

Steps to Join:

1. Copy this link - (<http://localhost:4000/api/team/join-team/67f223c794c60b51550426de>)
2. Login to the student panel
3. Go to the 'Join Team' section
4. Insert the link
5. Click join

STUDENT ANNOUNCEMENT PAGE



The sidebar contains a logo at the top, followed by a title "Student". Below the title is a vertical list of navigation items:

- Dashboard
- Messaging
- Events
- Announcement
- Team
- Profile
- Logout

Announcements

I want everyone to be present in uniform along with ID card

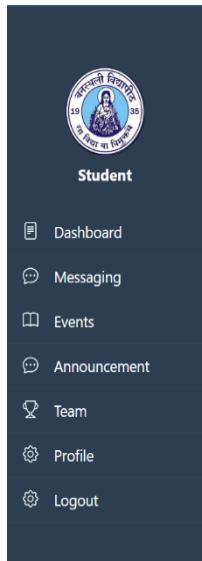
So these are some files

[ConversionToCausalForm.pptx \(application/vnd.openxmlformats-officedocument.presentationml.presentation\)](#)

[LabHandout_BTech_CSE_IT_VI Sem_CS317L_AI&MLab.pdf \(application/pdf\)](#)

[TheoryHandout_BTech_CSE_IT_VI Sem_CS317_AI&ML.pdf \(application/pdf\)](#)

STUDENT EVENT PAGE



The sidebar contains a logo at the top, followed by a title "Student". Below the title is a vertical list of navigation items:

- Dashboard
- Messaging
- Events
- Announcement
- Team
- Profile
- Logout

Student Event Calendar

Upcoming Events

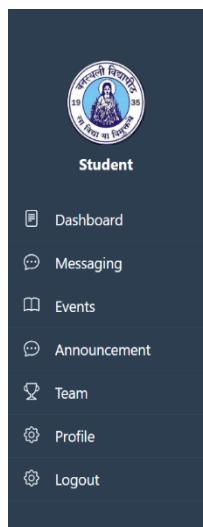
Khadi is Compulsory
Invalid Date

Project Demonstration
Invalid Date

Reload
4/18/2025

now
3/3/2025

STUDENT PROFILE PAGE



TEACHER FILE UPLOAD

Upload Teacher CSV

Choose File No file chosen

Upload

Teacher List

Name	Email	Phone	Subject
Swadha Sri	btbtii22143_swadha@banasthali.in	1020304055	dbms
Himani Dobriyal	btbtii22141_himani@banasthali.in	1020345678	html
Isha	btbtii22137_isha@banasthali.in	7894560123	css
Ayushi Mishra	btbtii22088_ayushi@banasthali.in	1234560789	js

ADMIN ANNOUNCEMENT PAGE

The screenshot shows the Admin Announcement Page. On the left is a sidebar with a logo at the top and a list of navigation links:

- Dashboard
- Announcement
- Events & Calendar
- Student Upload
- Teacher Upload
- Settings & Profile
- Logout

The main content area has a title "Announcement" and a form for sending an announcement:

Announcement:

Upload Files:

Choose Files No file chosen

Announcements Done:

I want everyone to be present in uniform along with ID card

So these are some files

ConversionToClausalfForm.pptx (application/vnd.openxmlformats-officedocument.presentationml.presentation)

LabHandout_BTech_CSE_IT_VI Sem_CS317L_AI&MLLab.pdf (application/pdf)

FEEDBACK FORM

The screenshot shows the Teacher Feedback Form page. On the left is a sidebar with a logo at the top and a list of navigation links:

- Dashboard
- Messaging
- Requests
- Events & Calendar
- Teacher Profile
- Feedback
- Settings & Profile
- Logout

The main content area contains a "Feedback Form" modal window:

Feedback Form

Team ID

Rating (1-5)

Feedback

FEEDBACK FORM

The screenshot shows a user interface for a feedback form. On the left, there is a vertical sidebar titled "Teacher" containing a logo and a list of navigation items: Dashboard, Messaging, Requests, Events & Calendar, Teacher Profile, Feedback, Settings & Profile, and Logout. The main area is titled "Feedback Form" and contains three input fields: "Team ID", "Rating (1-5)", and "Feedback". Below these fields is a "Submit Feedback" button.

Feedback Form

Team ID

Rating (1-5)

Feedback

Submit Feedback

APPENDICES

1. **Mentor (Teacher):** A mentor or professional who provides guidance and support to a mentee in a particular technical field.
2. **Mentee (Student):** A mentee seeking mentorship for their project or study.
3. **Coordinator (Admin):** An admin-like role responsible for managing system operations, including group assignments and database management.
4. **Dashboard:** A personalised interface that provides users with an overview of relevant data, such as notifications and project status.
5. **JSX:** It stands for JavaScript XML. It allows and makes it easier for us to write and add HTML in React.
6. **JSON:** It stands for JavaScript Object Notation, which is a standard text-based format for representing structured data based on JavaScript object syntax.
7. **RESTful API:** It is an interface that two computer systems use to exchange information securely over the internet.

REFERENCES

1. "IEEE Guide for Software Requirements Specifications," in *IEEE Std 830-1984*, vol., no., pp.1-26, 10 Feb. 1984, doi: 10.1109/IEEESTD.1984.119205.
2. "IEEE Recommended Practice for Software Design Descriptions," in *IEEE Std 1016-1987*, vol., no., pp.1-16, 13 July 1987, doi: 10.1109/IEEESTD.1987.122643.
3. 978-0134852157, Grady Booch, James Rumbaugh, Ivar Jacobson, Unified Modelling Language User Guide, Second Edition, Addison-Wesley, June 2005.
4. 0071168982, Raghu Ramakrishnan, Database Management Systems, International 2 Revised ed edition, McGraw-Hill Education (ISE Editions), December 1999.
5. S. Oaks, "Java Performance: The Definitive Guide," O'Reilly Media, 2014.
6. https://www.researchgate.net/publication/324472353_A_web_service_based_on_RESTful_API_and_JSON_SchemaJSON_Meta_Schema_to_construct_knowledge_graphs
7. https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1909&context=etd_projects
8. <https://dl.ebooksworld.ir/books/BEGINNINGReactJS.Foundations.Chris.Minnick.Wiley.9781119685548.EBooksWorld.i>