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)

5/4/25

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

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

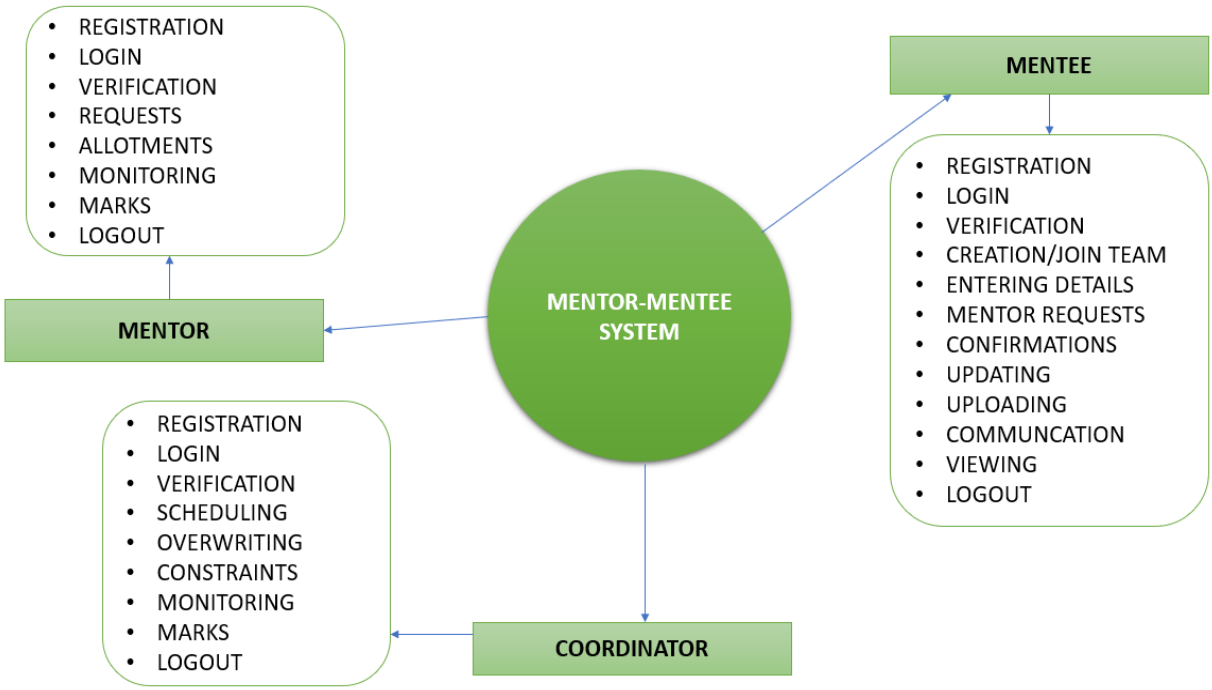
- FEASIBILITY STUDY

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.

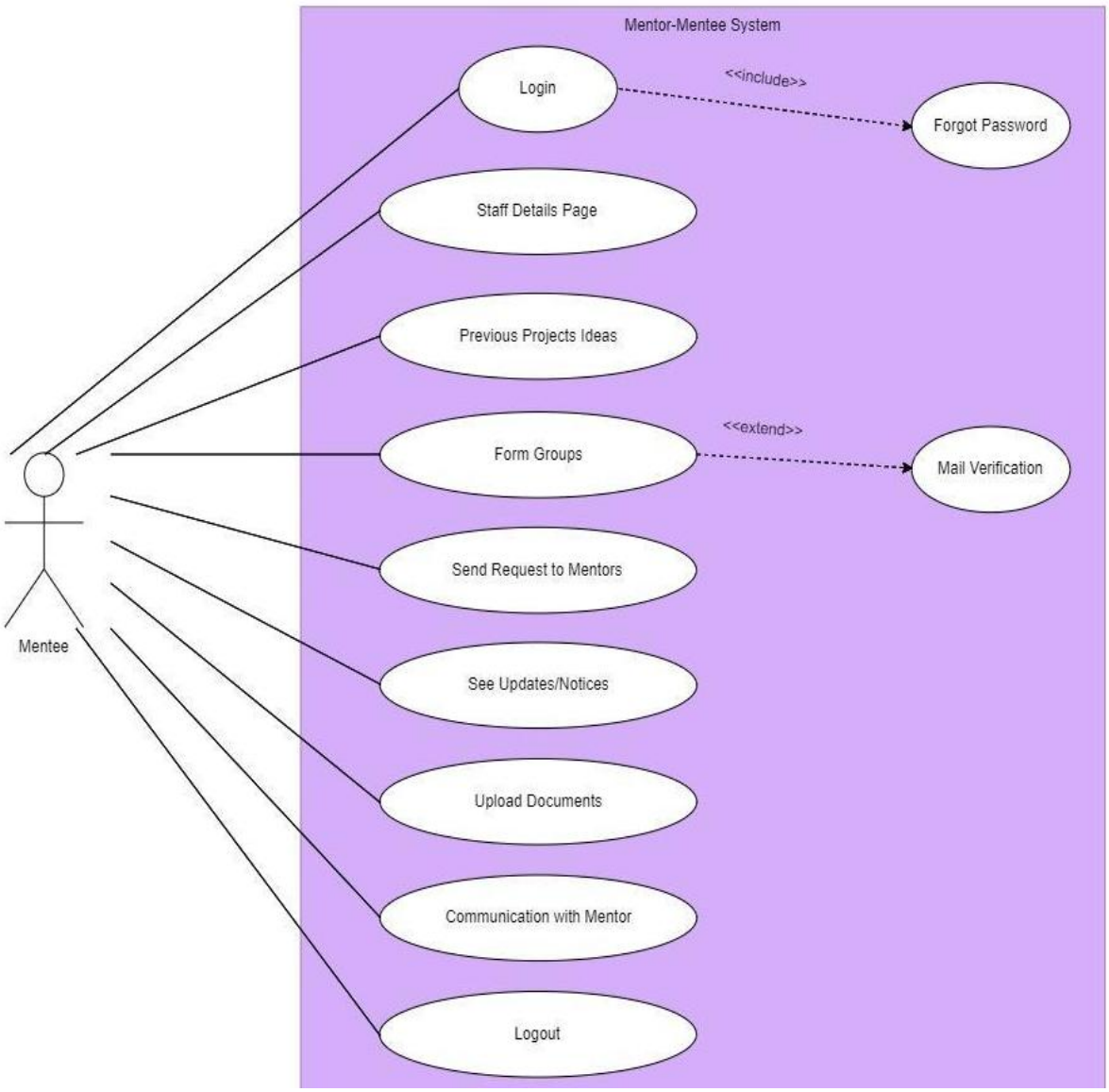
8. **Data Storage for the Coordinator:** Coordinators can store and access all mentee and project data securely.

9. **Marks Allocation Based on Presentation:** Grades are assigned to mentees based on their project presentations.

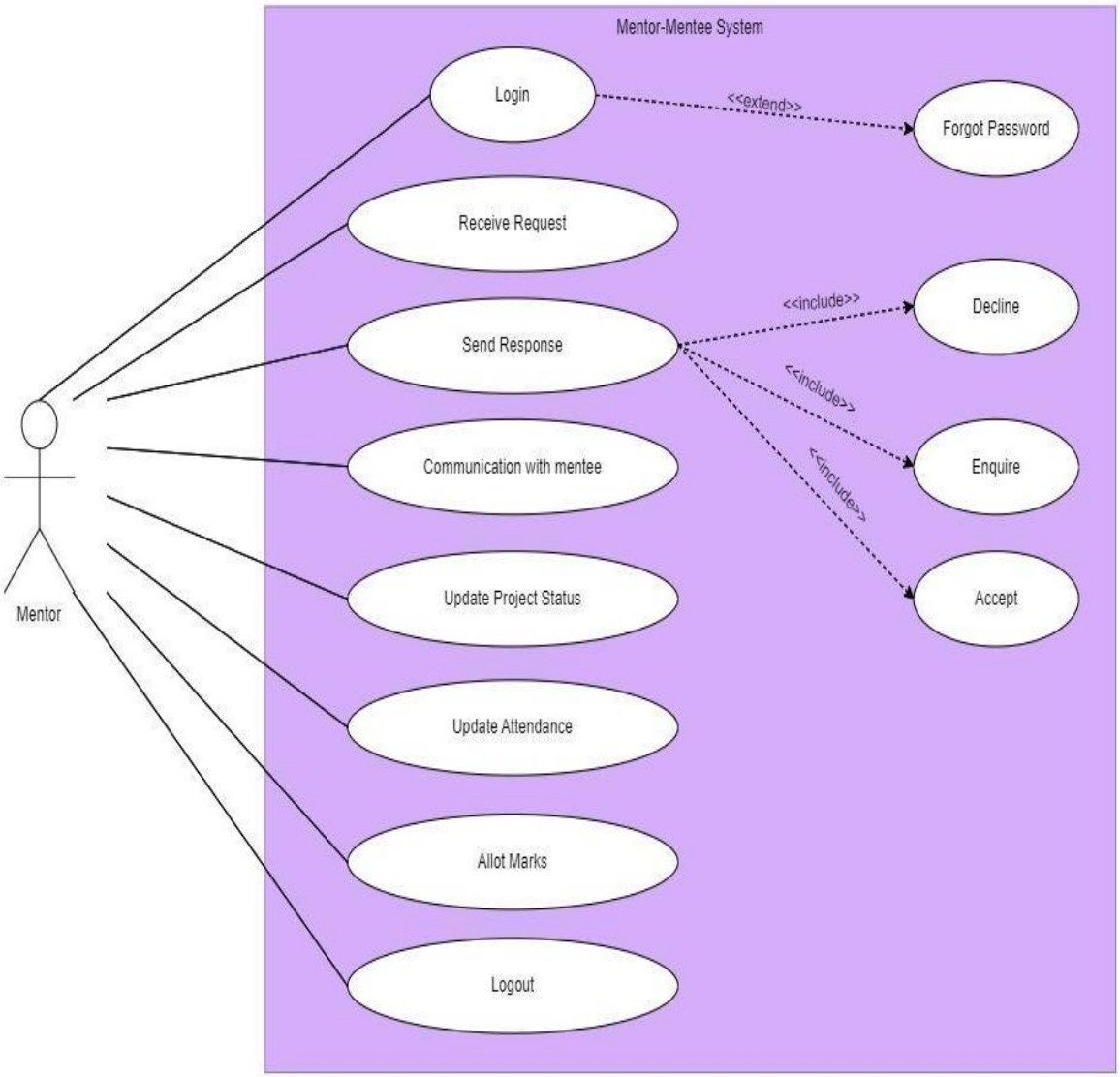
10. **Announcement Dashboard:** A dashboard for posting and viewing important announcements and updates.

- USE CASE DIAGRAMS

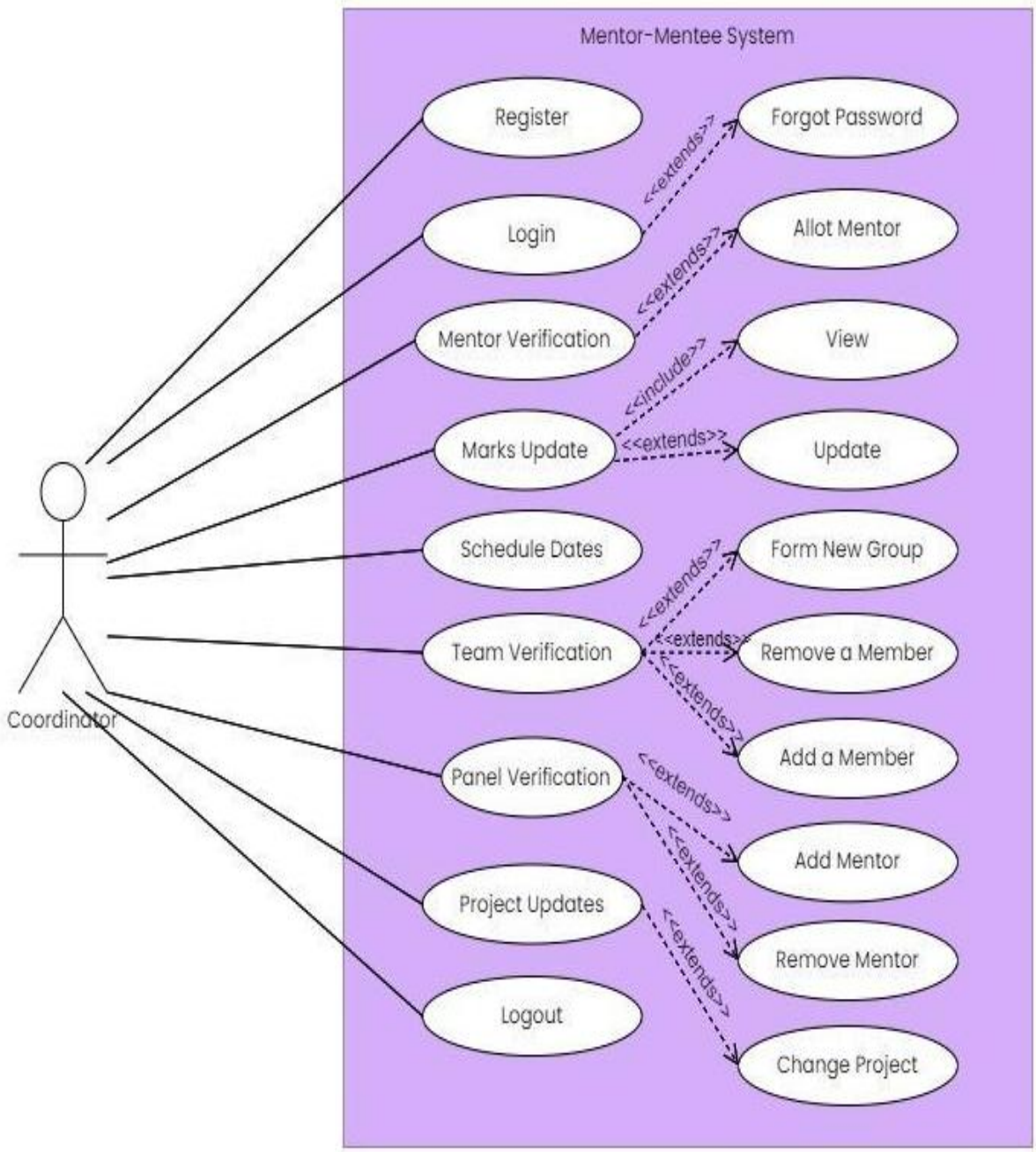
MENTEE



MENTOR

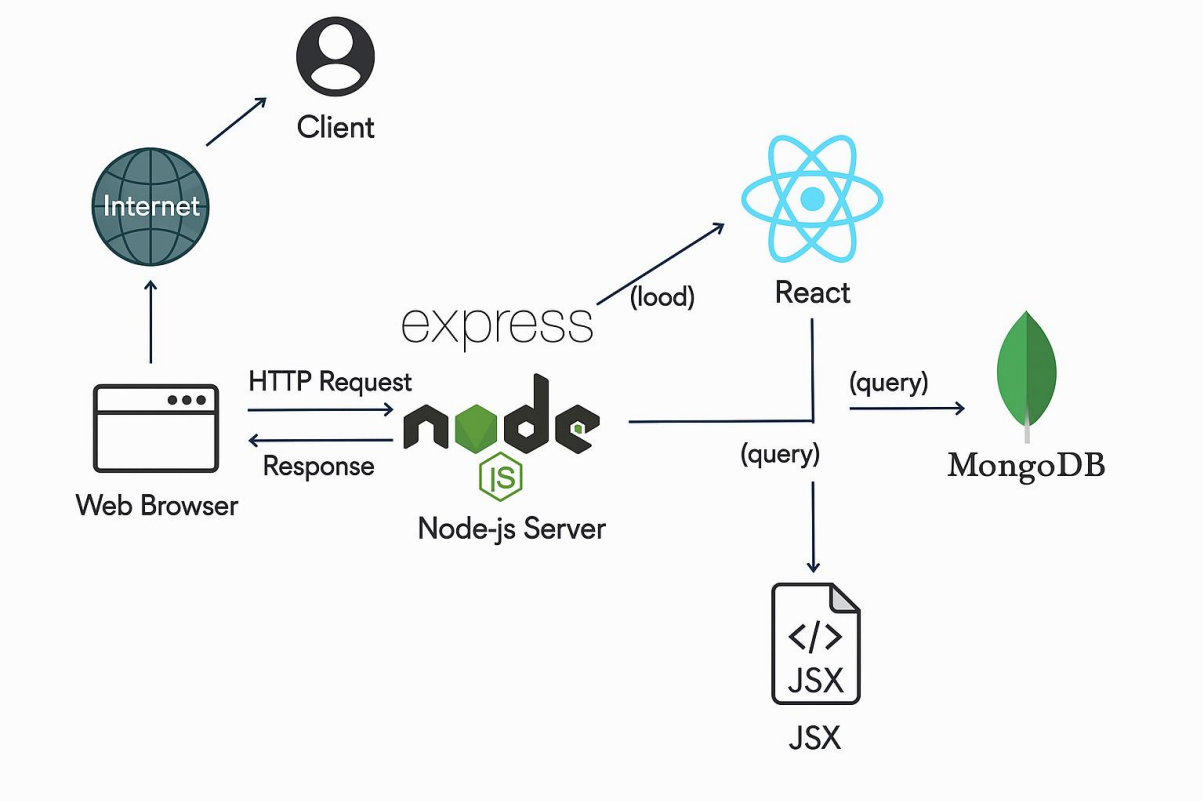

Mentor-Mentee System
Login
Forgot Password
<<extend>>
Receive Request
Send Response
Decline
<<include>>
<<include>>
<<include>>
Enquire
Accept
Communication with mentee
Update Project Status
Update Attendance
Allot Marks
Logout
Mentor

COORDINATOR

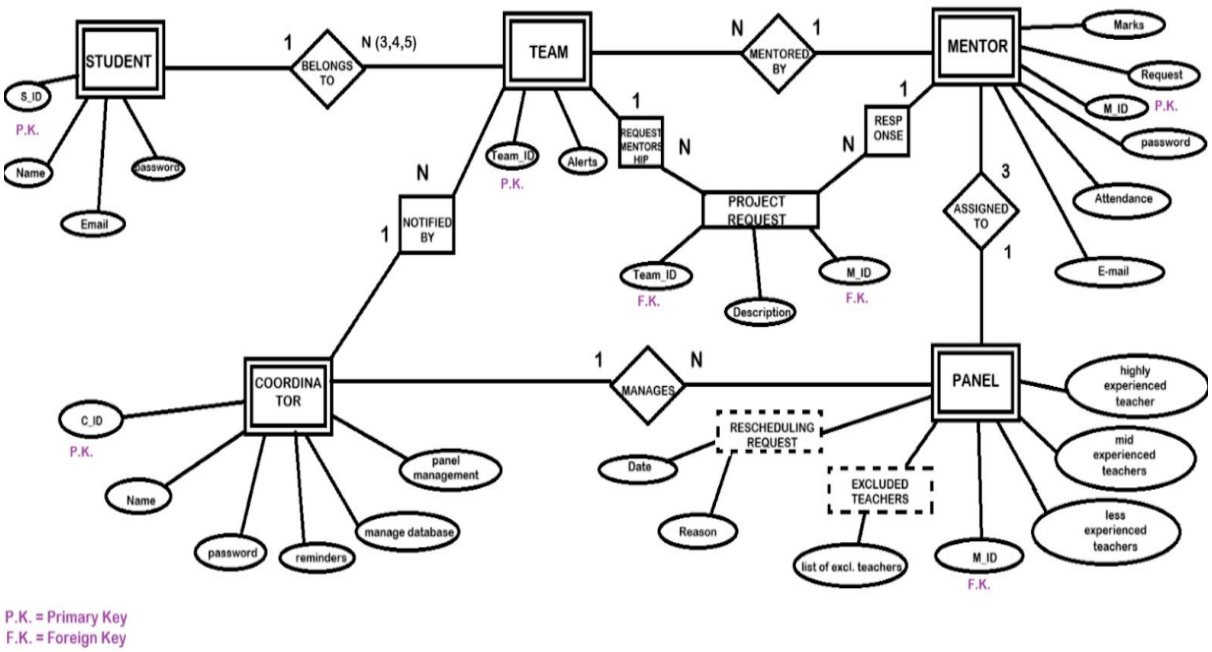


SYSTEM DESIGN (SDS)

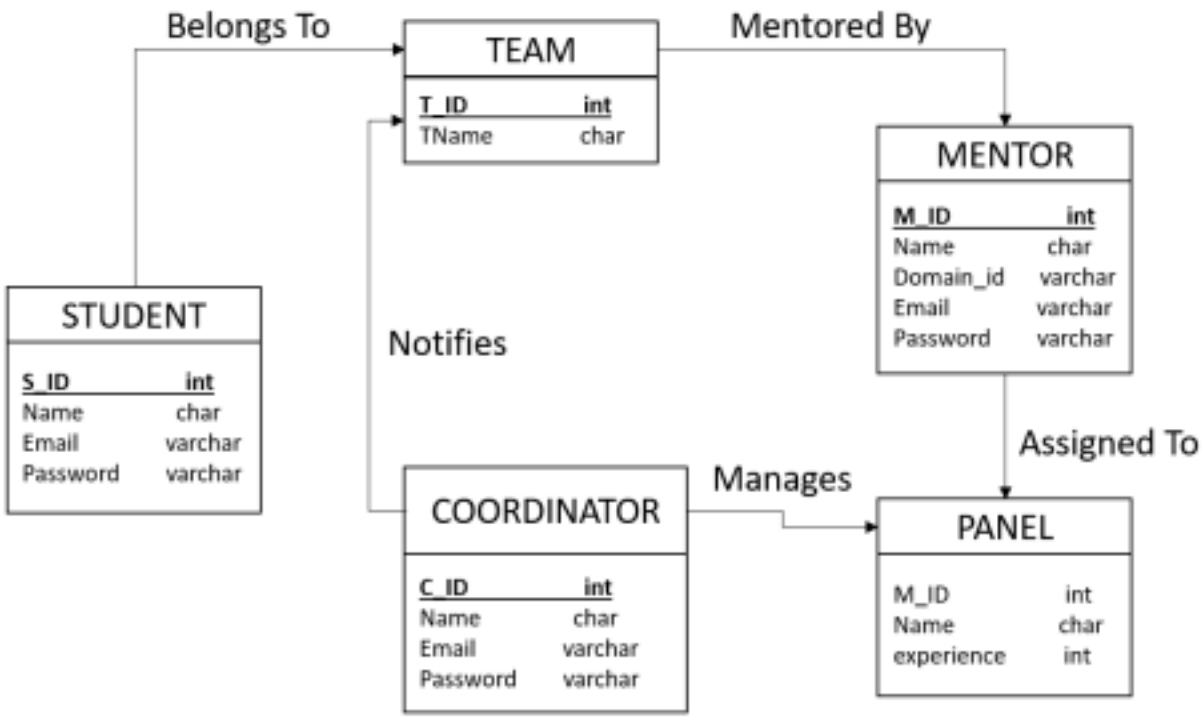
- High Level Design



- ER Diagram

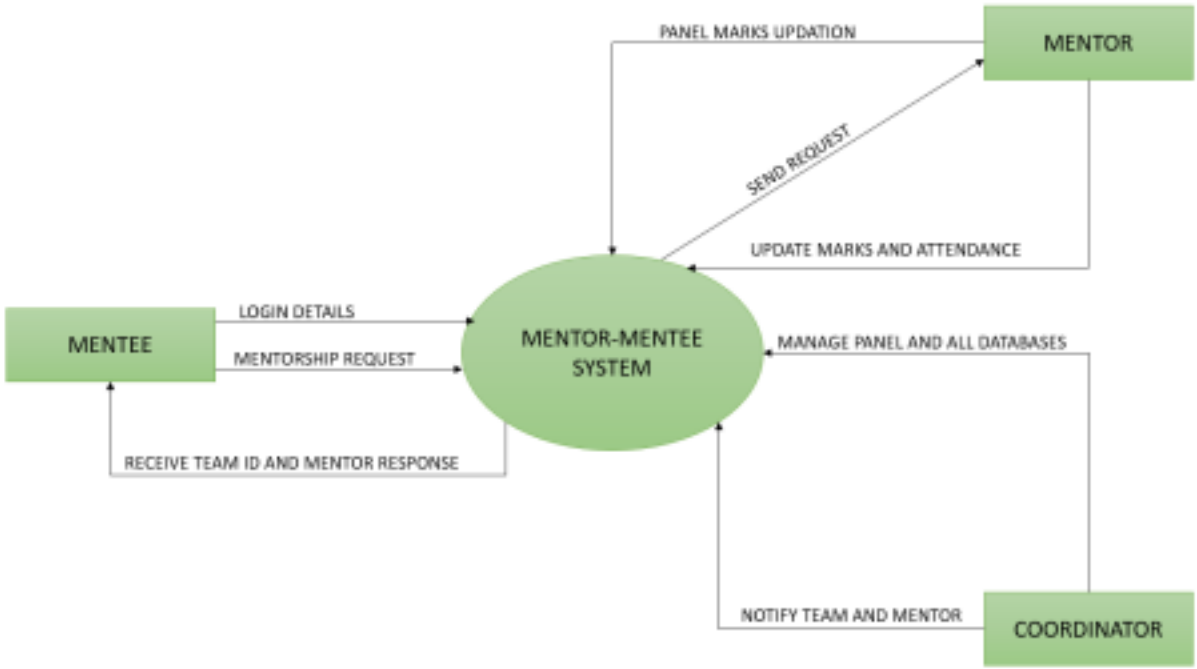


- Database Design

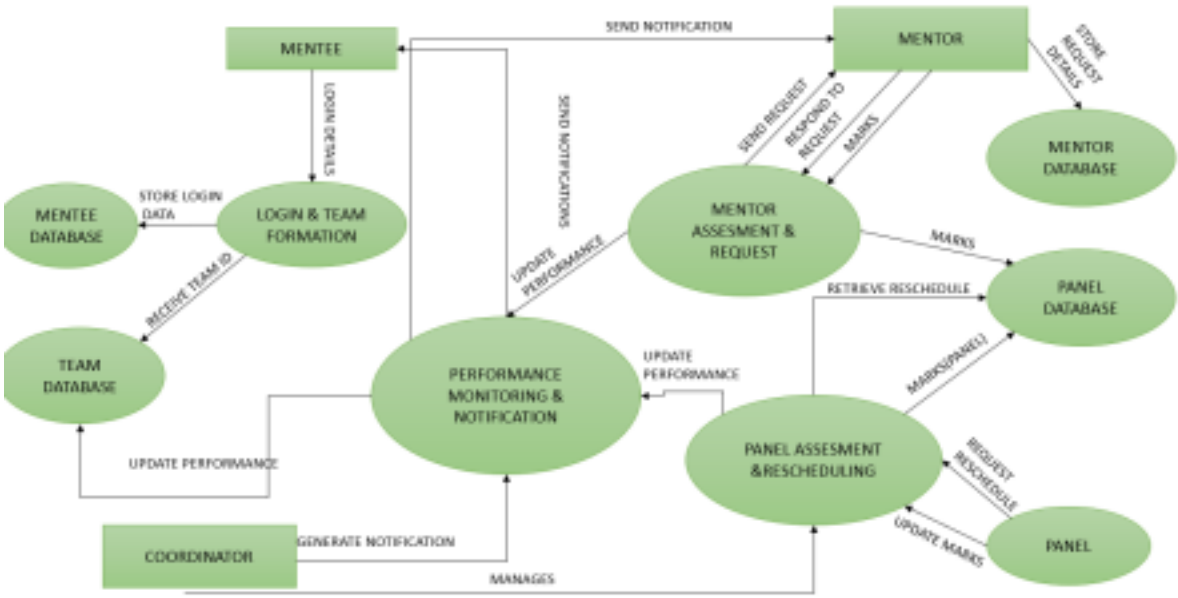


- DATA FLOW DIAGRAMS

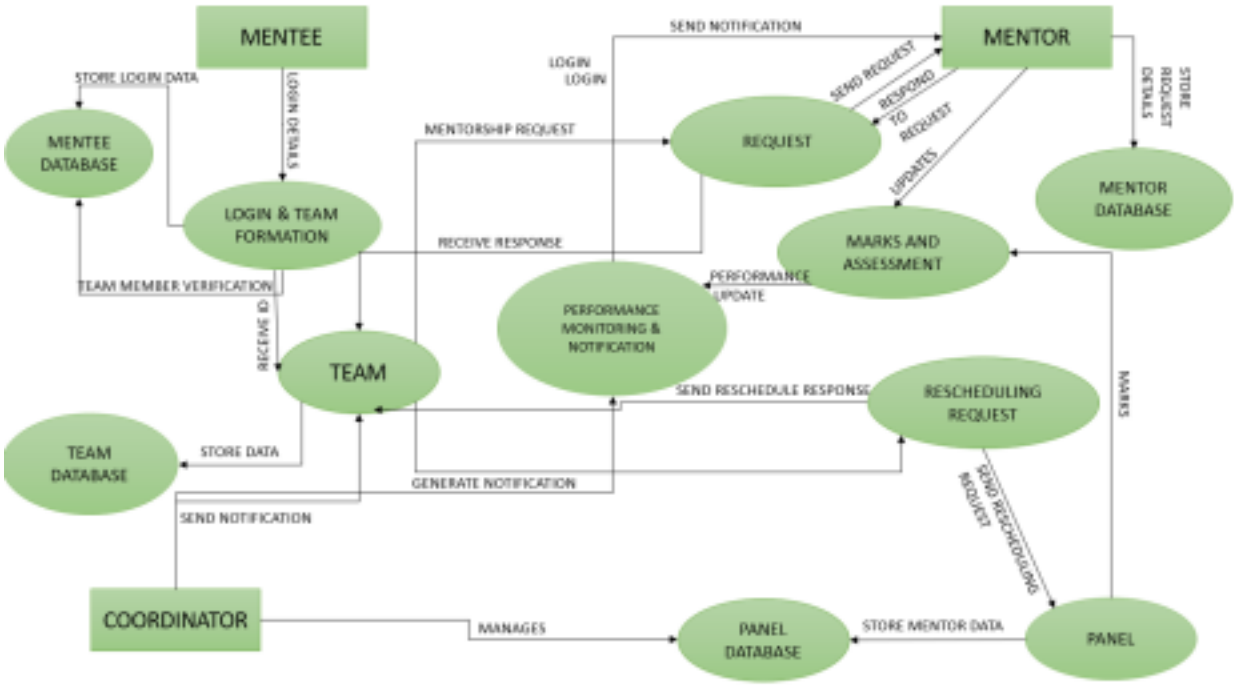
DFD LEVEL – 0



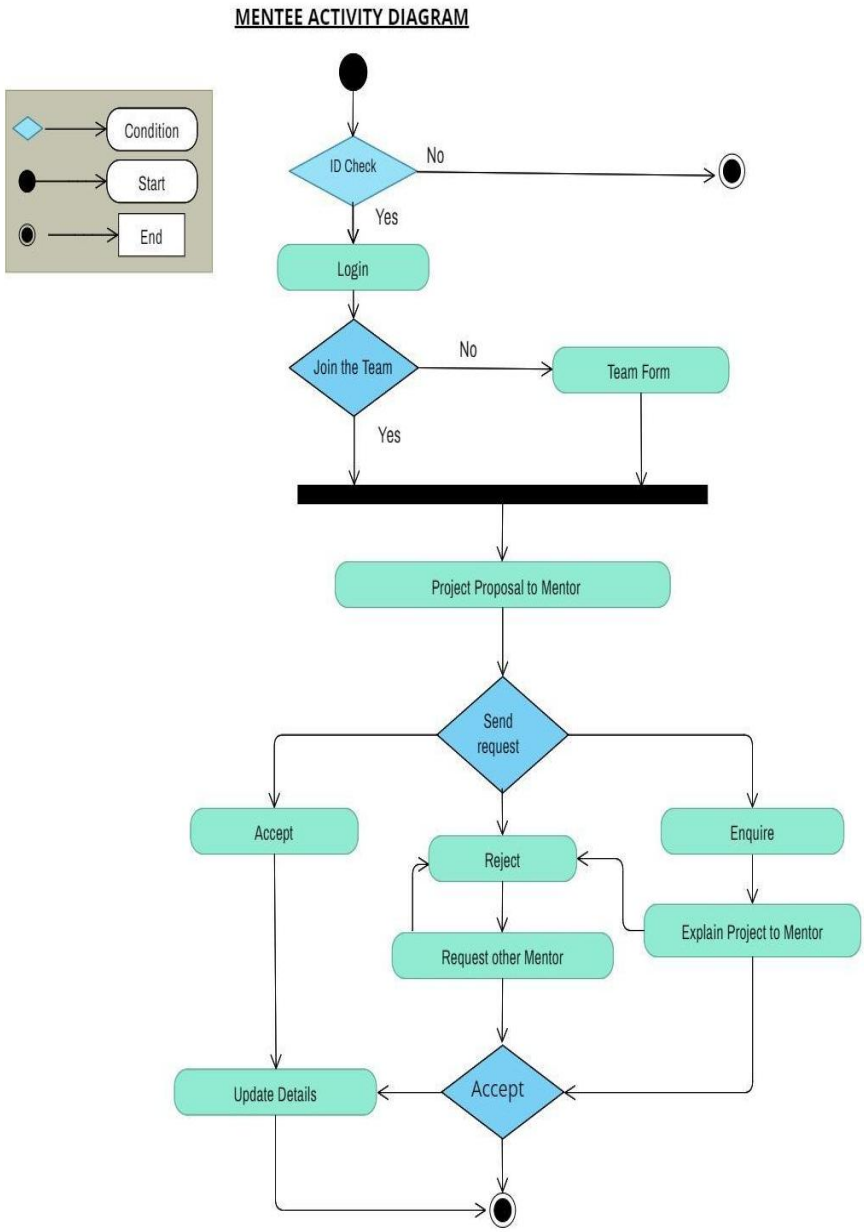
DFD LEVEL – 1



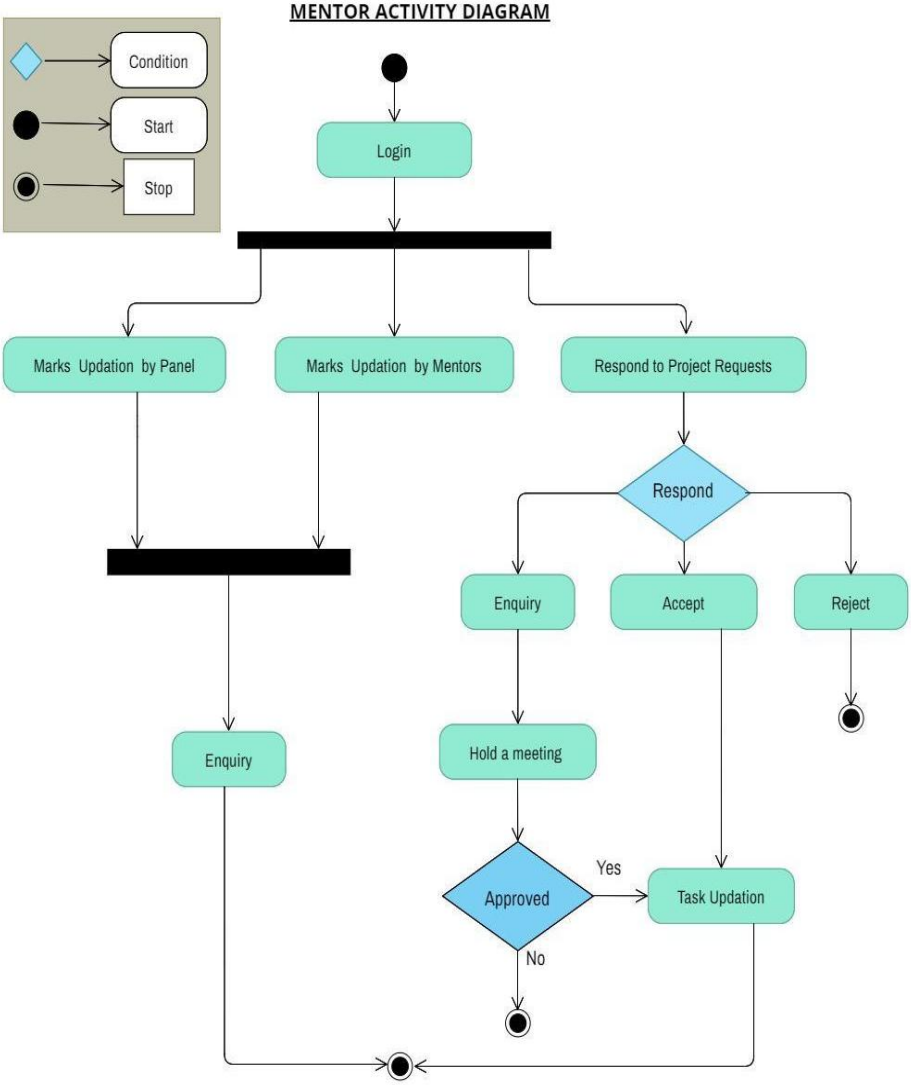
DFD LEVEL – 2


MENTEE
STORE LOGIN DATA
LOGIN DETAILS
MENTEE DATABASE
LOGIN & TEAM FORMATION
TEAM MEMBER VERIFICATION
RECEIVE ID
TEAM
TEAM DATABASE
STORE DATA
SEND NOTIFICATION
COORDINATOR
MANAGES
PANEL DATABASE
STORE MENTOR DATA
PANEL
SEND NOTIFICATION
LOGIN
LOGIN
MENTORSHIP REQUEST
REQUEST
RECEIVE RESPONSE
PERFORMANCE MONITORING & NOTIFICATION
PERFORMANCE UPDATE
GENERATE NOTIFICATION
SEND RESCHEDULE RESPONSE
RESCHEDULING REQUEST
SEND RESCHEDULING REQUEST
MARKS
MENTOR
SEND REQUEST
RESPOND TO REQUEST
UPDATES
STORE REQUEST DETAILS
MENTOR DATABASE
MARKS AND ASSESSMENT

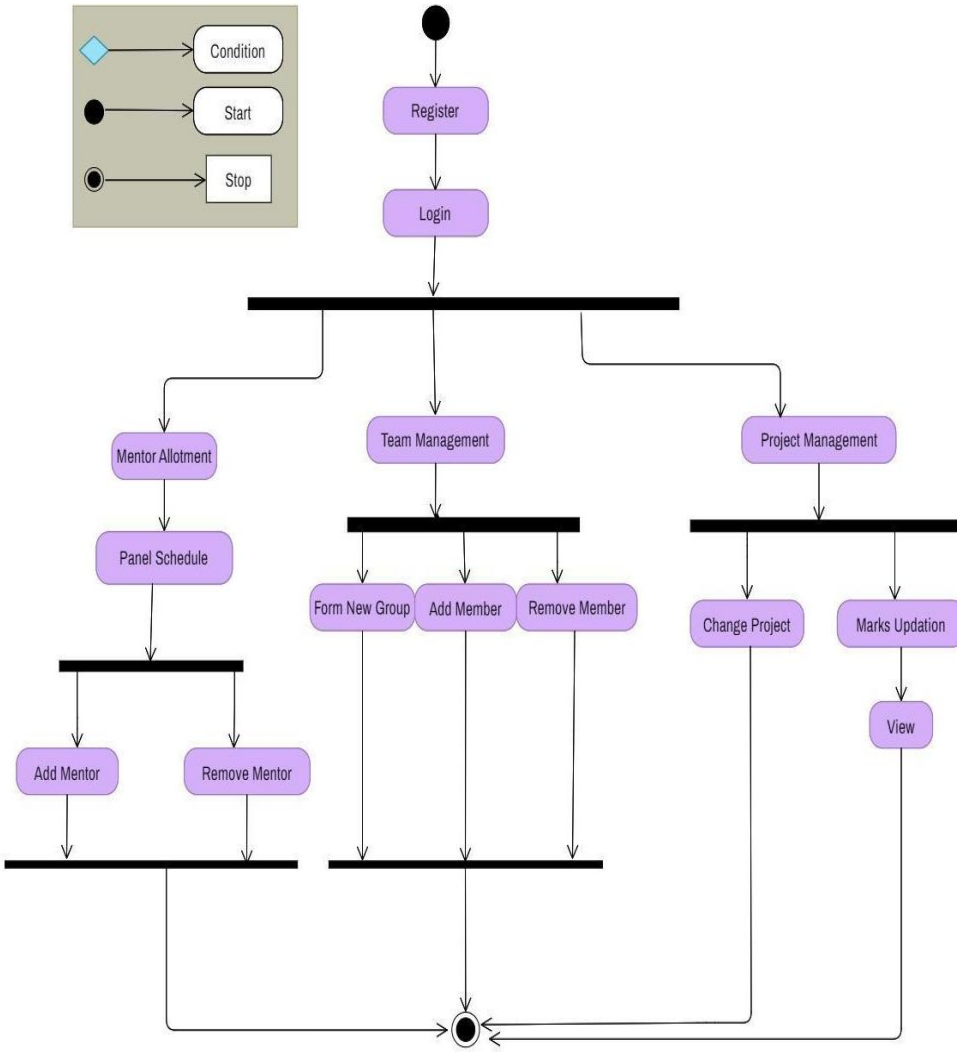
- ACTIVITY DIAGRAM



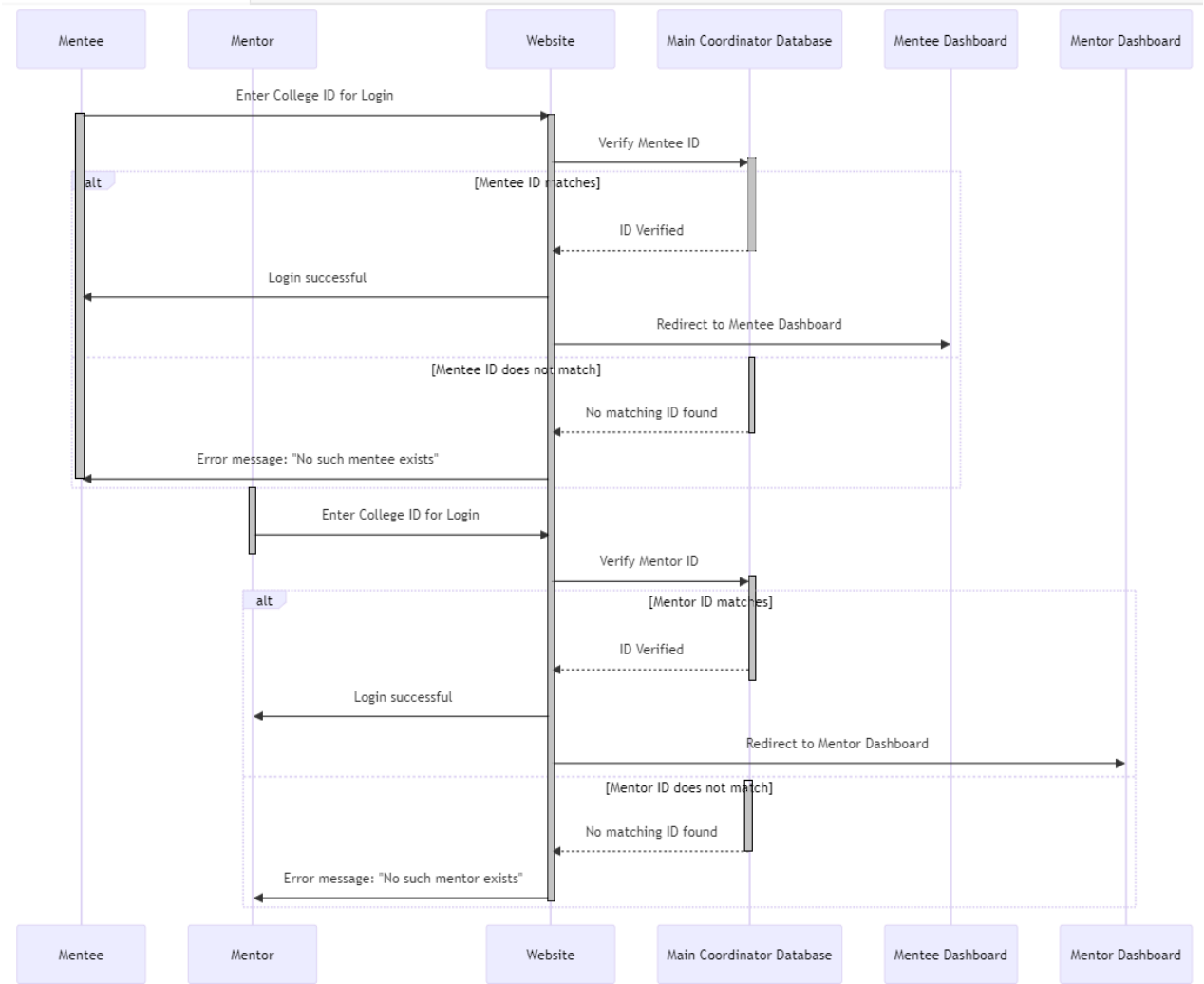
MENTOR ACTIVITY DIAGRAM



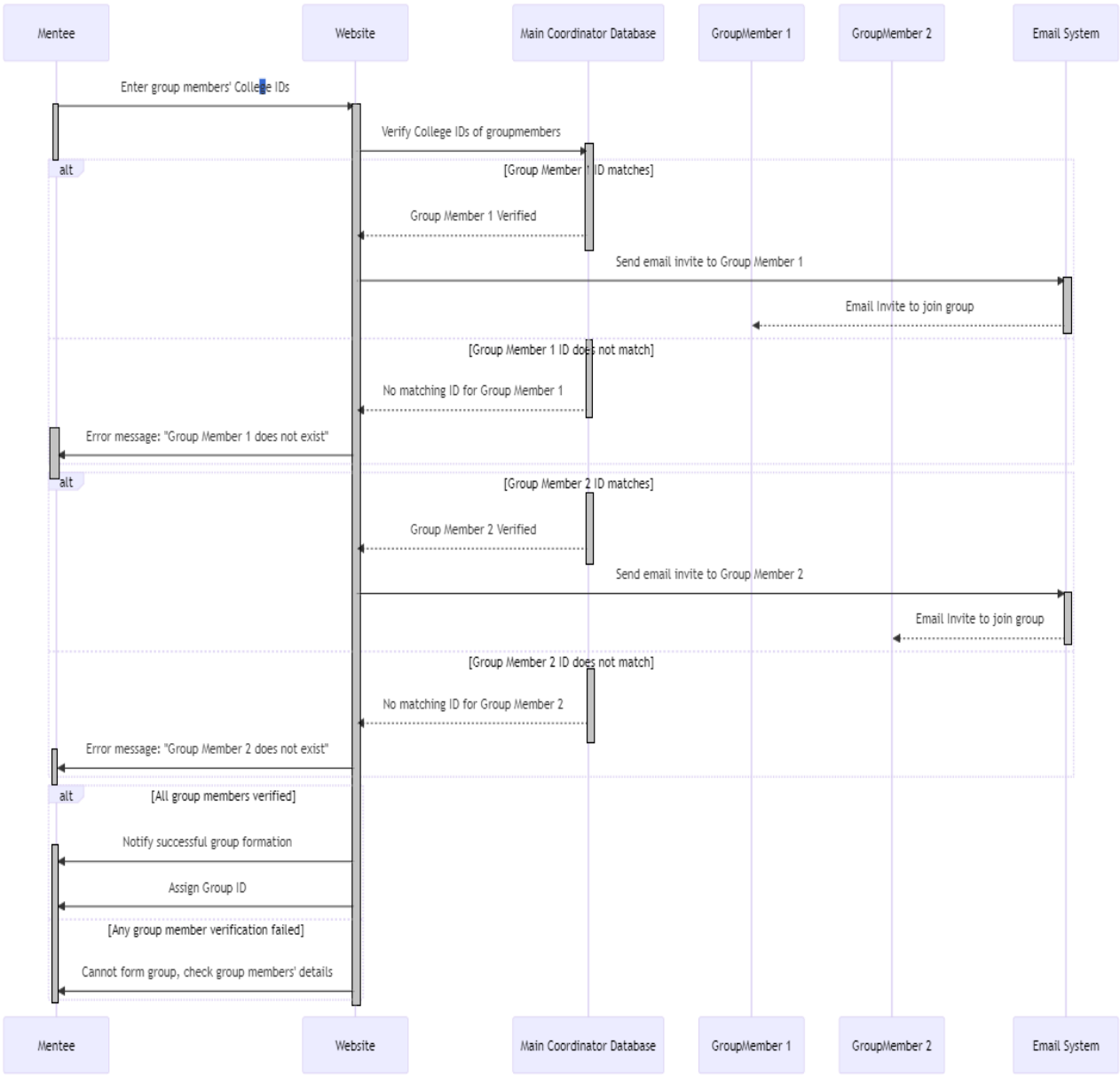
COORDINATOR ACTIVITY DIAGRAM



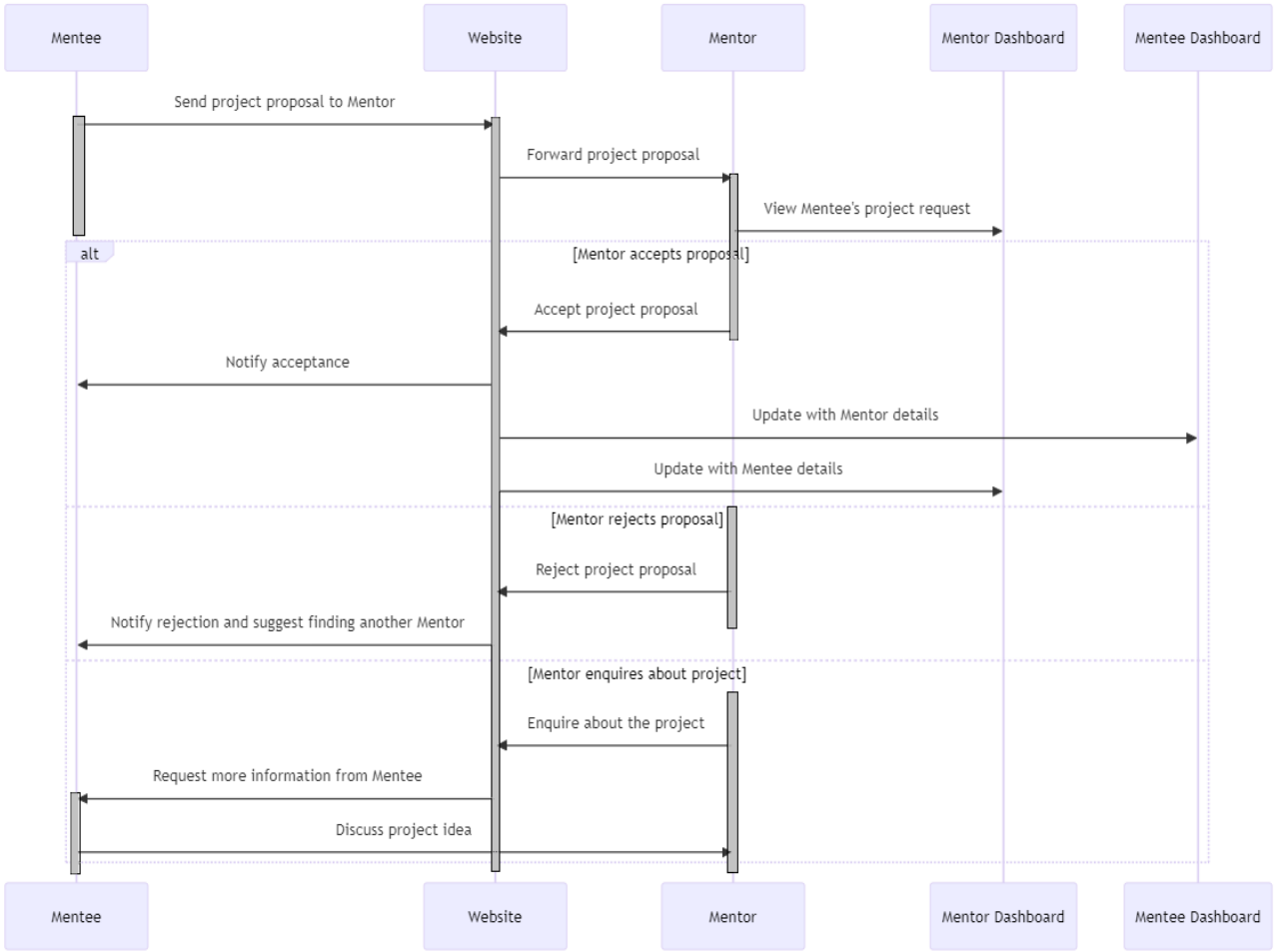
- SEQUENCE DIAGRAM



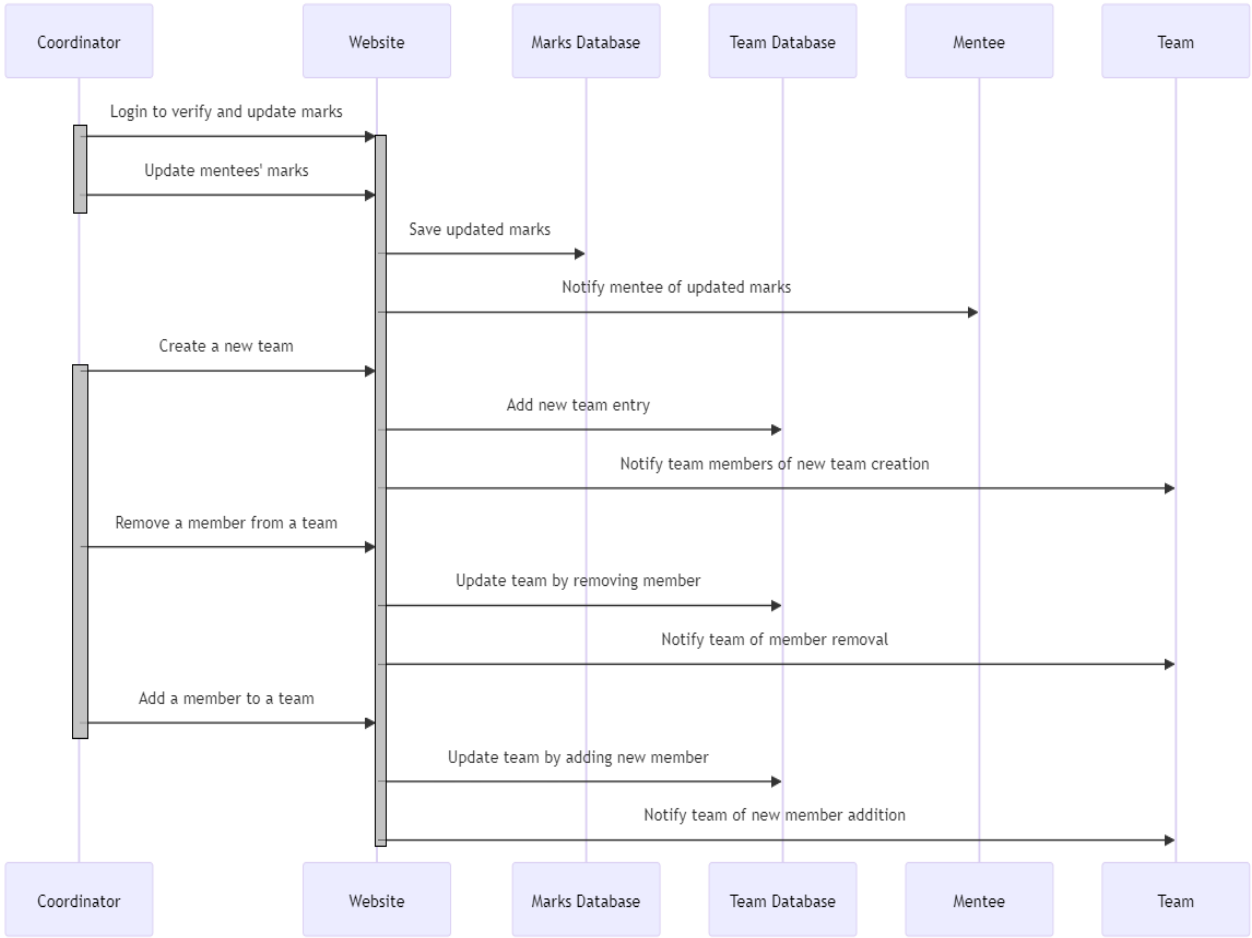
Login by both mentee and mentor



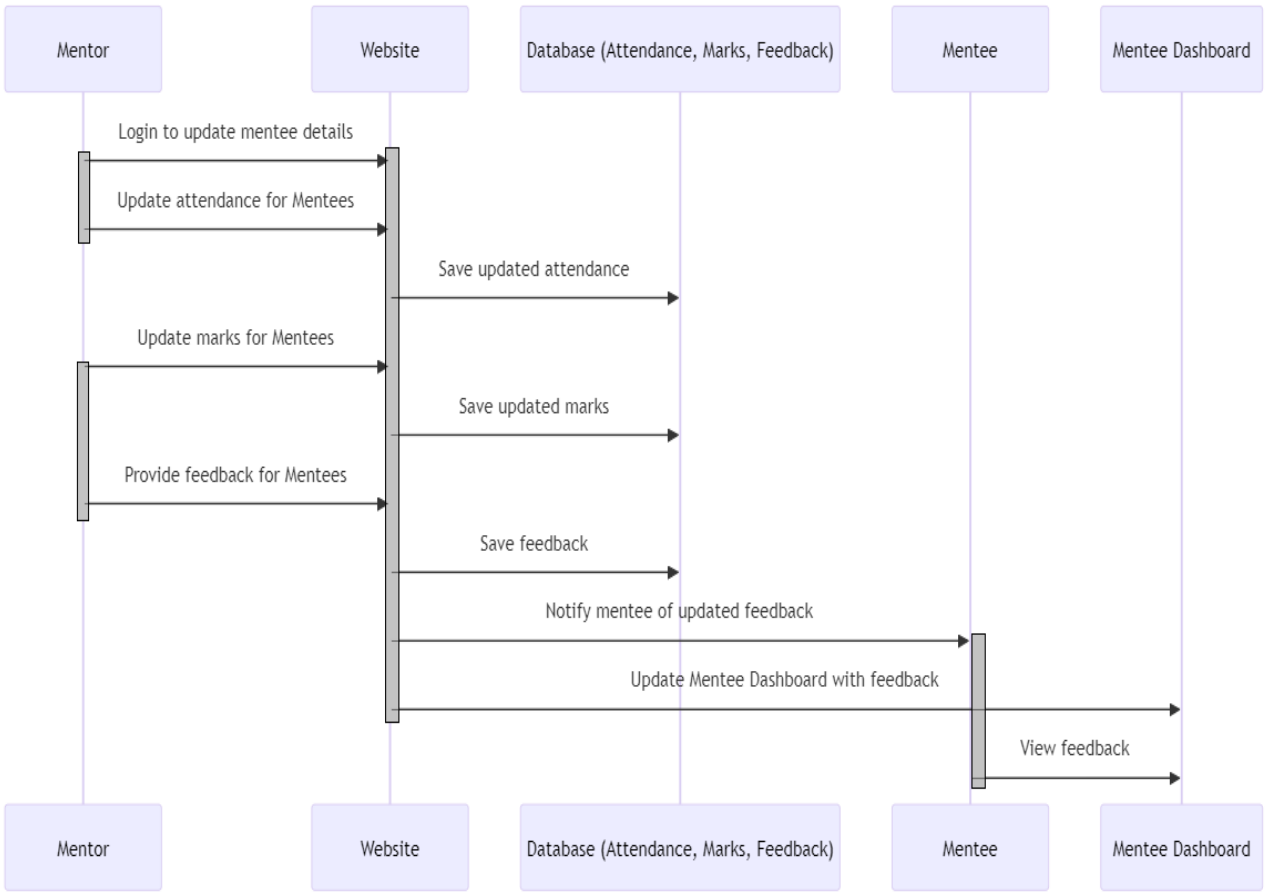
<u>Team formation by mentee</u>



Sending project proposal and receiving response from Mentor



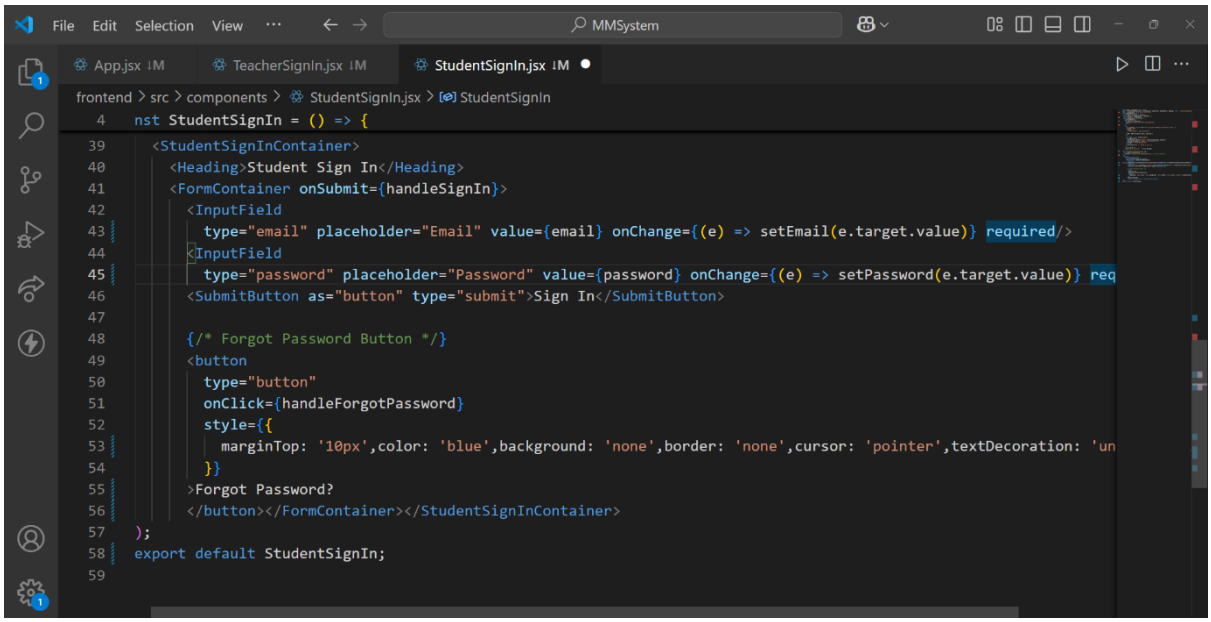
Coordinator accessing databases and performing updating

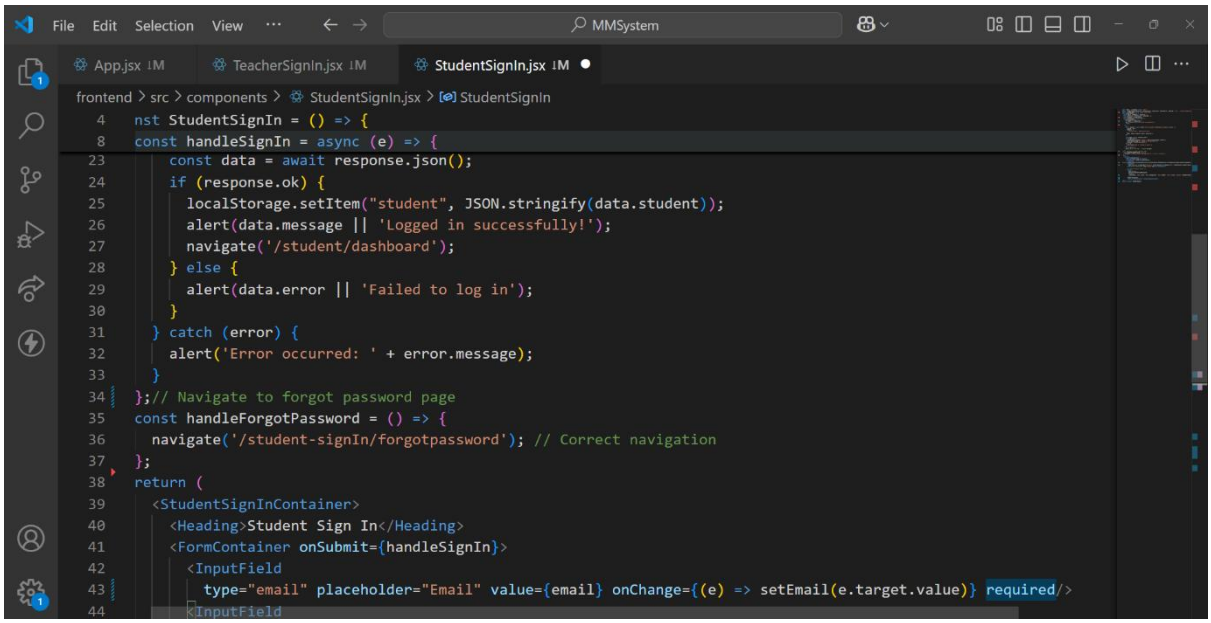


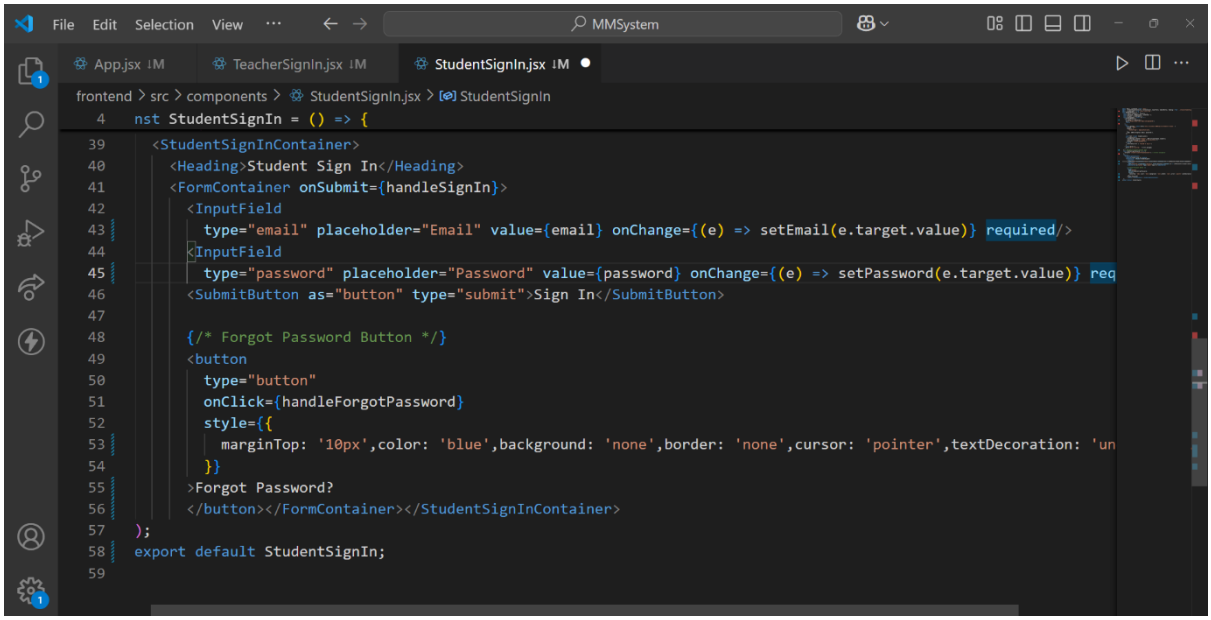
Mentor updating marks, attendance and providing feedback

CODING

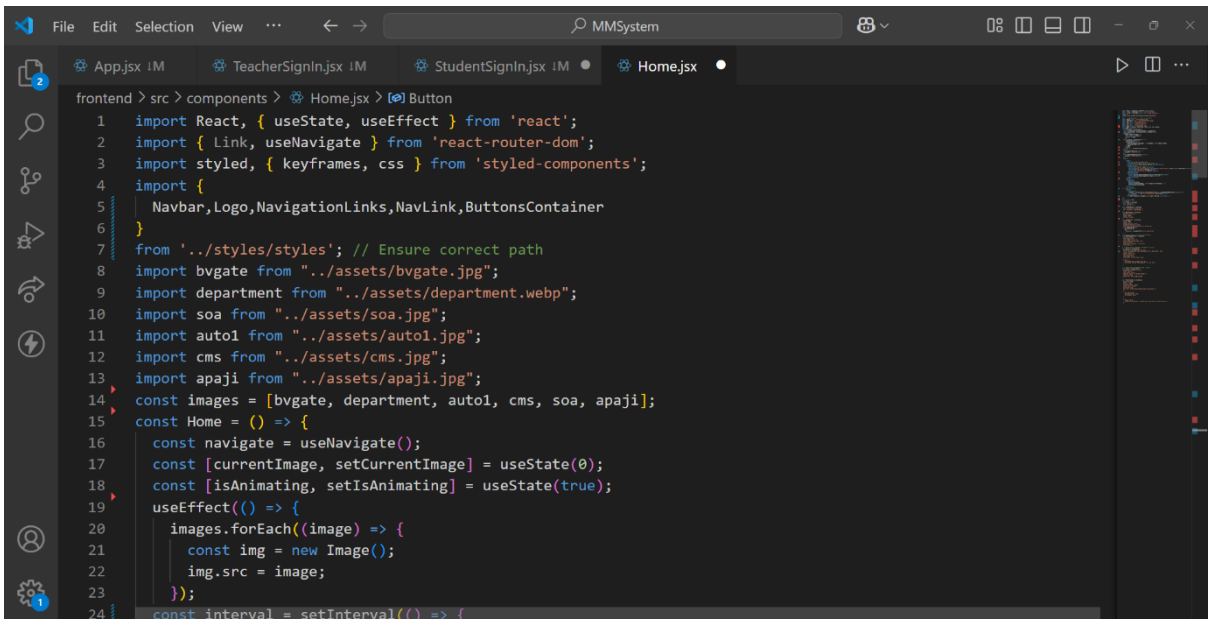
LOGIN

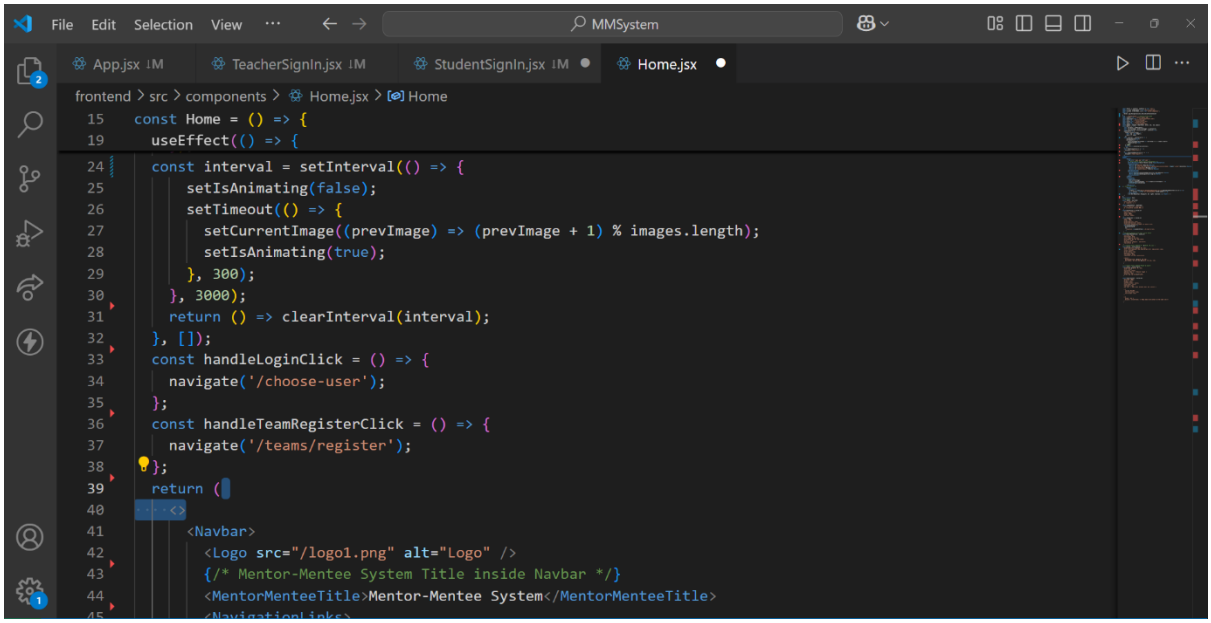


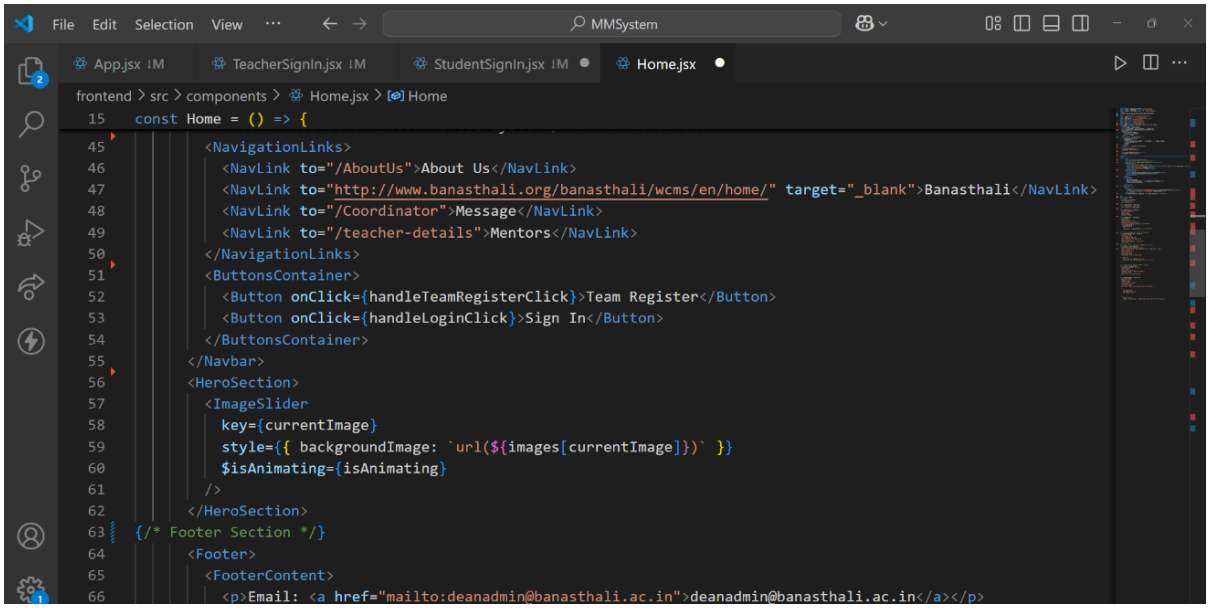



File Edit Selection View
MMSystem
App.jsx
TeacherSignIn.jsx
StudentSignIn.jsx
frontend > src > components > StudentSignIn.jsx > StudentSignIn
nst StudentSignIn = () => {
<StudentSignInContainer>
<Heading>Student Sign In</Heading>
<FormContainer onSubmit={handleSignIn}>
<InputField
type="email" placeholder="Email" value={email} onChange={(e) => setEmail(e.target.value)} required/>
<InputField
type="password" placeholder="Password" value={password} onChange={(e) => setPassword(e.target.value)} req
<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: 'un
}}
>Forgot Password?
</button></FormContainer></StudentSignInContainer>
);
export default StudentSignIn;

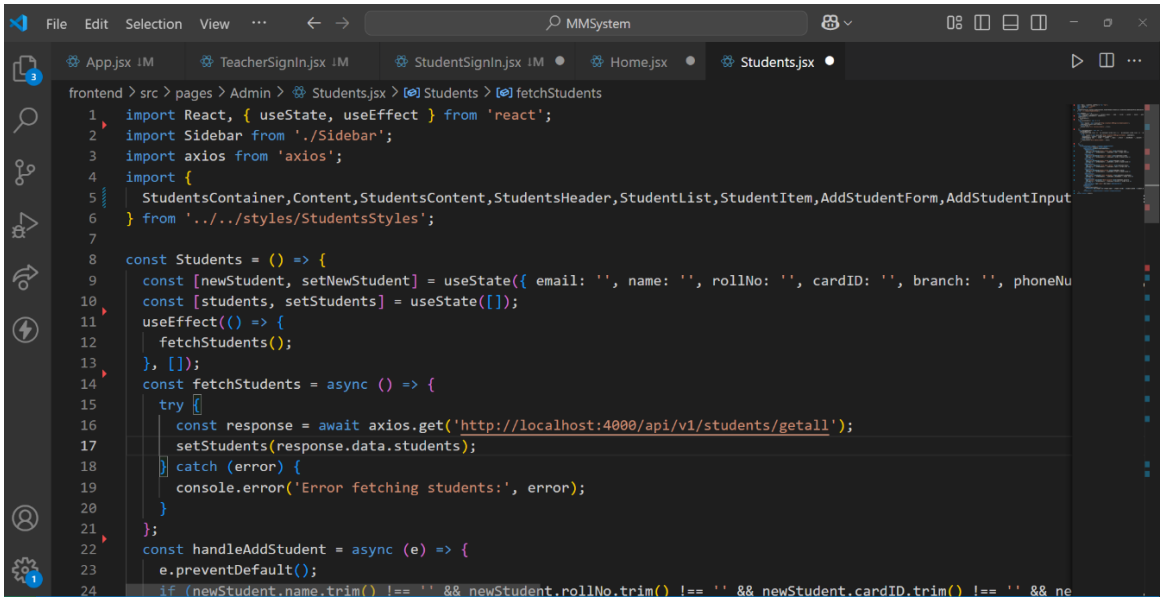
HOME PAGE

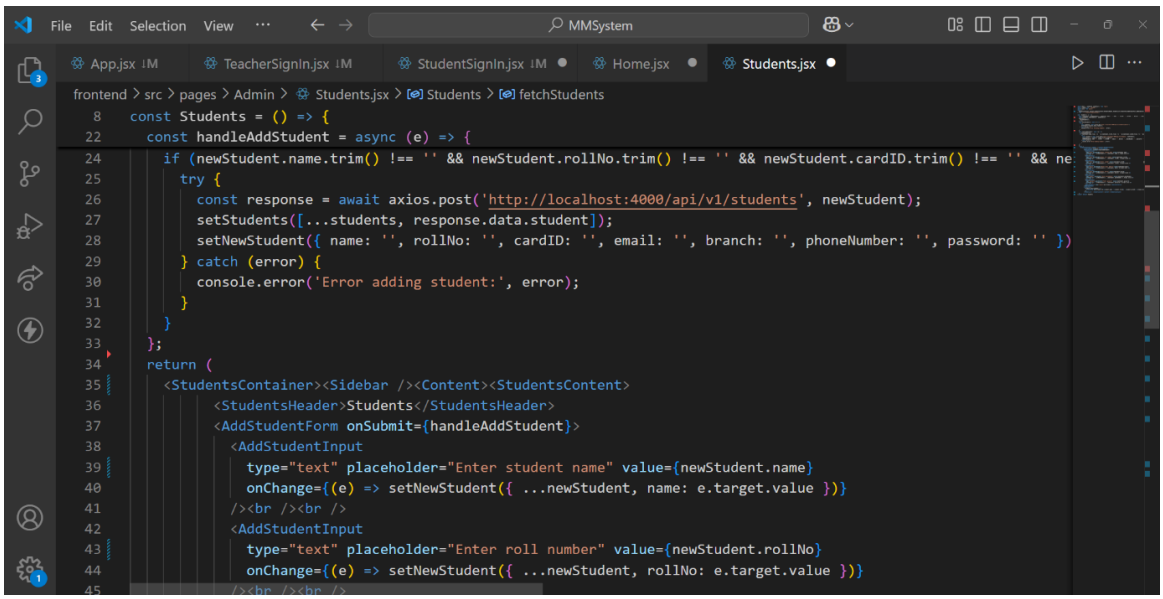

File Edit Selection View
MMSystem
App.jsx
TeacherSignIn.jsx
StudentSignIn.jsx
Home.jsx
frontend > src > components > Home.jsx > Button
import React, { useState, useEffect } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import styled, { keyframes, css } from 'styled-components';
import {
Navbar,Logo,NavigationLinks,NavLink,ButtonsContainer
}
from '../styles/styles'; // Ensure correct path
import bvgate from "../assets/bvgate.jpg";
import department from "../assets/department.webp";
import soa from "../assets/soa.jpg";
import auto1 from "../assets/auto1.jpg";
import cms from "../assets/cms.jpg";
import apaji from "../assets/apaji.jpg";
const images = [bvgate, department, auto1, cms, soa, apaji];
const Home = () => {
const navigate = useNavigate();
const [currentImage, setCurrentImage] = useState(0);
const [isAnimating, setIsAnimating] = useState(true);
useEffect(() => {
images.forEach((image) => {
const img = new Image();
img.src = image;
});

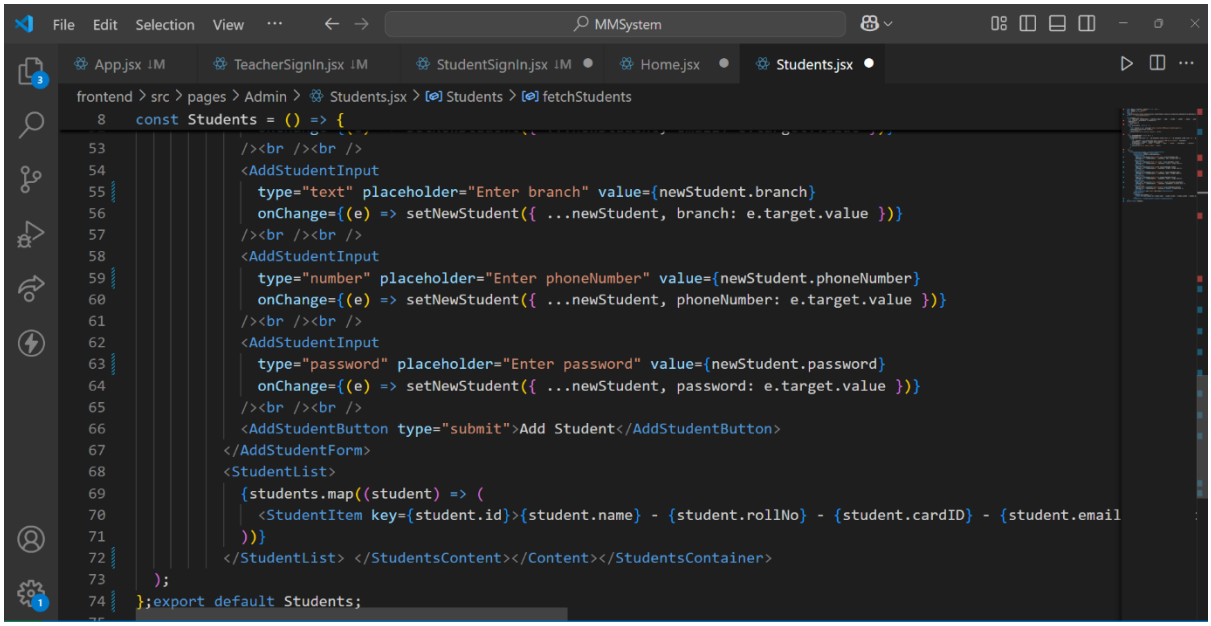

File Edit Selection View
MMSystem
App.jsx M
TeacherSignIn.jsx M
StudentSignIn.jsx M
Home.jsx
frontend > src > components > Home.jsx > Home
const Home = () => {
useEffect(() => {
const interval = setInterval(() => {
setIsAnimating(false);
setTimeout(() => {
setCurrentImage((prevImage) => (prevImage + 1) % images.length);
setIsAnimating(true);
}, 300);
}, 3000);
return () => clearInterval(interval);
}, []);
const handleLoginClick = () => {
navigate('/choose-user');
};
const handleTeamRegisterClick = () => {
navigate('/teams/register');
};
return (
<>
<Navbar>
<Logo src="/logo1.png" alt="Logo" />
{/* Mentor-Mentee System Title inside Navbar */}
<MentorMenteeTitle>Mentor-Mentee System</MentorMenteeTitle>


File Edit Selection View
MMSystem
App.jsx M
TeacherSignIn.jsx M
StudentSignIn.jsx M
Home.jsx
frontend > src > components > Home.jsx > Home
const Home = () => {
<NavigationLinks>
<NavLink to="/AboutUs">About Us</NavLink>
<NavLink to="http://www.banasthali.org/banasthali/wcms/en/home/" target="_blank">Banasthali</NavLink>
<NavLink to="/Coordinator">Message</NavLink>
<NavLink to="/teacher-details">Mentors</NavLink>
</NavigationLinks>
<ButtonsContainer>
<Button onClick={handleTeamRegisterClick}>Team Register</Button>
<Button onClick={handleLoginClick}>Sign In</Button>
</ButtonsContainer>
</Navbar>
<HeroSection>
<ImageSlider
key={currentImage}
style={{ backgroundImage: `url(${images[currentImage]})` }}
$isAnimating={isAnimating}
/>
</HeroSection>
{/* Footer Section */}
<Footer>
<FooterContent>
Email: deanadmin@banasthali.ac.in

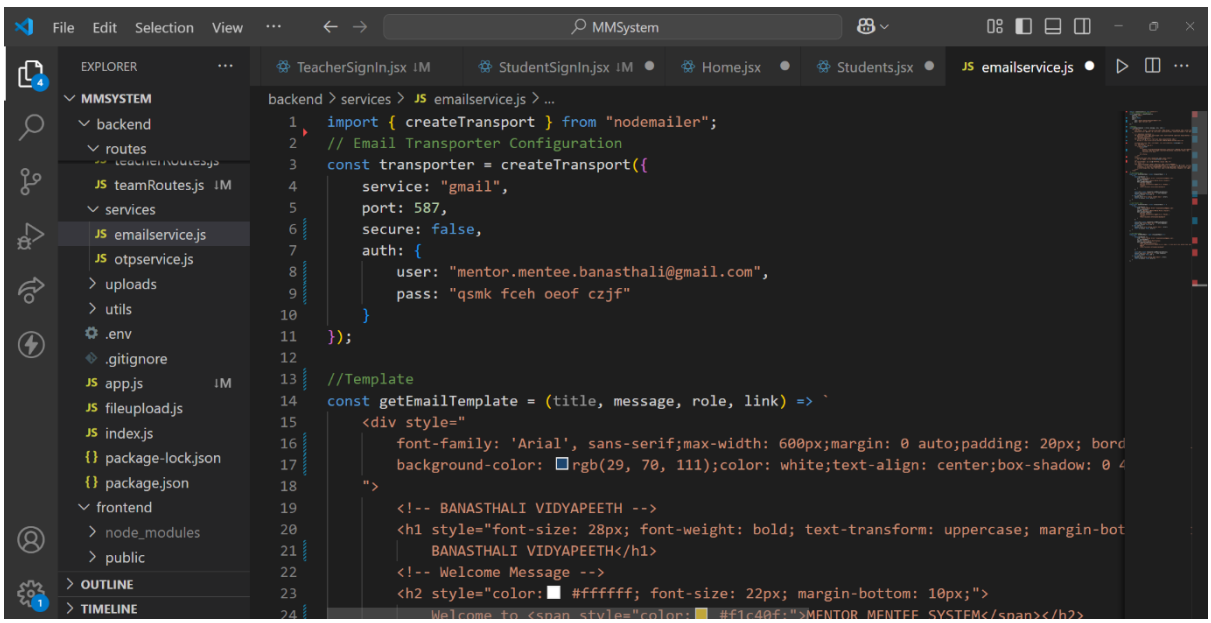
STUDENT DASHBOARD

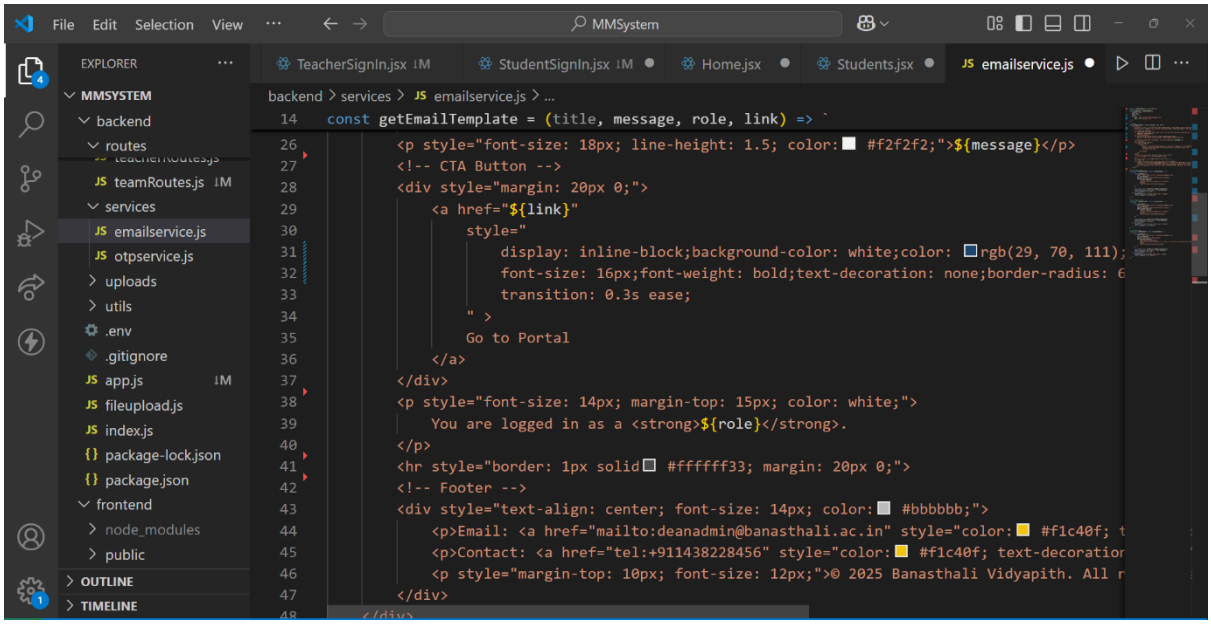






EMAIL SERVICE




File Edit Selection View
MMSystem
EXPLORER
MMSYSTEM
backend
routes
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
OUTLINE
TIMELINE
TeacherSignIn.jsx
StudentSignIn.jsx
Home.jsx
Students.jsx
emailservice.js
backend > services > emailservice.js > ...
const getEmailTemplate = (title, message, role, link) => `
${message}
<!-- CTA Button -->

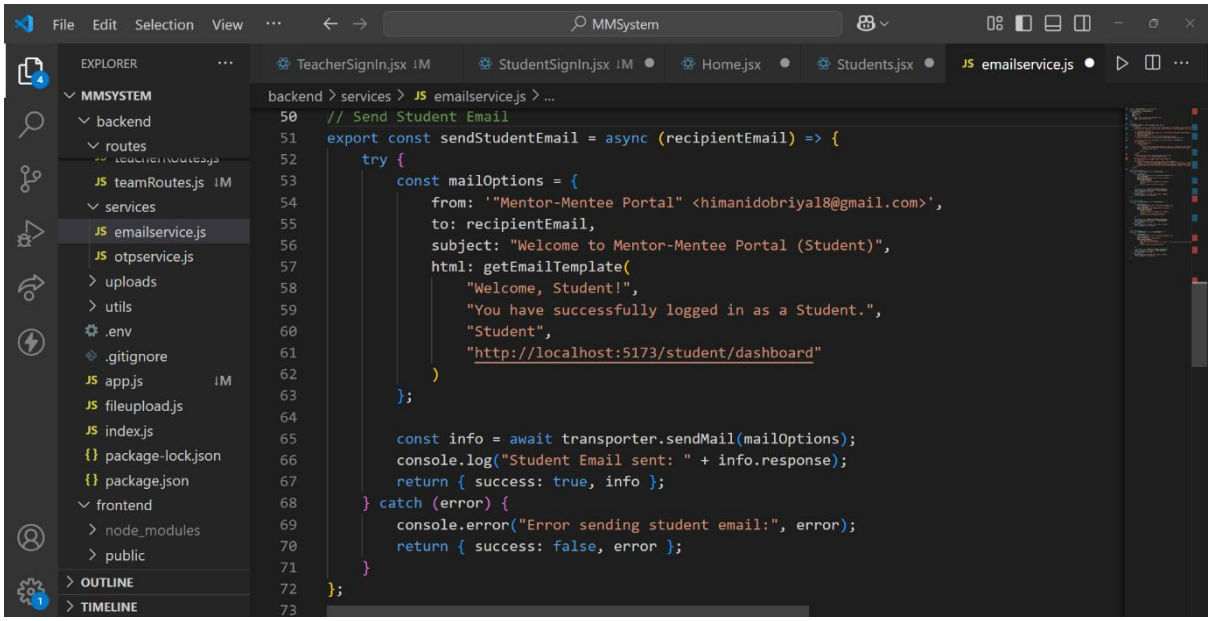
<a href="${link}"
style="
display: inline-block;background-color: white;color: rgb(29, 70, 111);
font-size: 16px;font-weight: bold;text-decoration: none;border-radius: 6
transition: 0.3s ease;
" >
Go to Portal

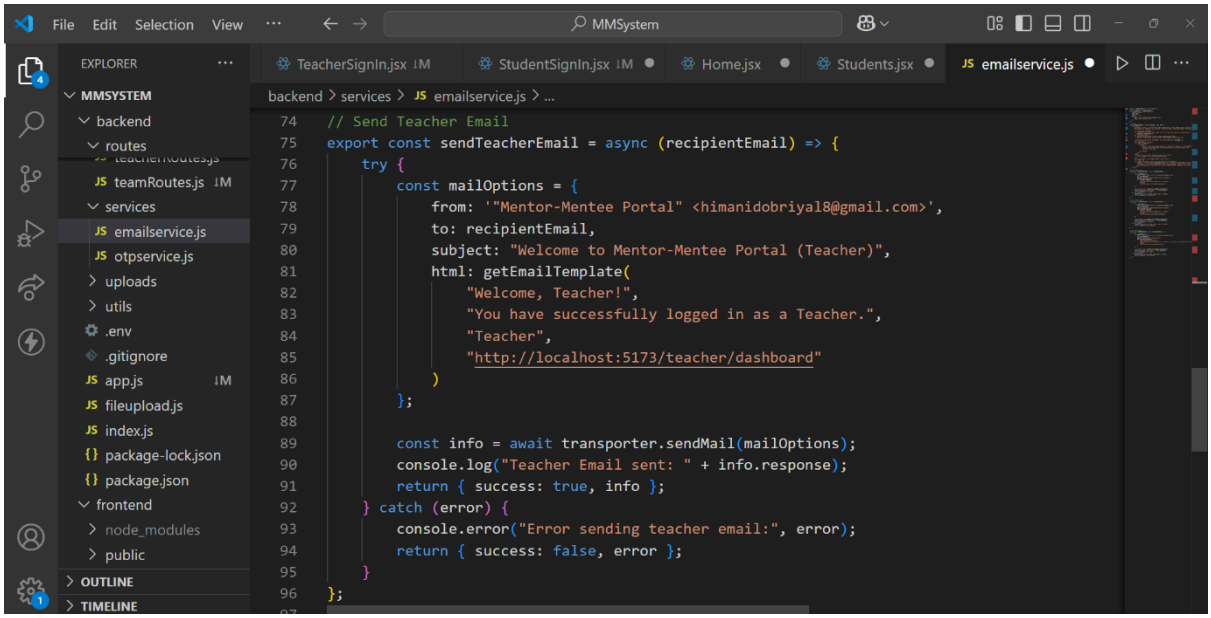
You are logged in as a ${role}.

<hr style="border: 1px solid #ffffff33; margin: 20px 0;">
<!-- Footer -->

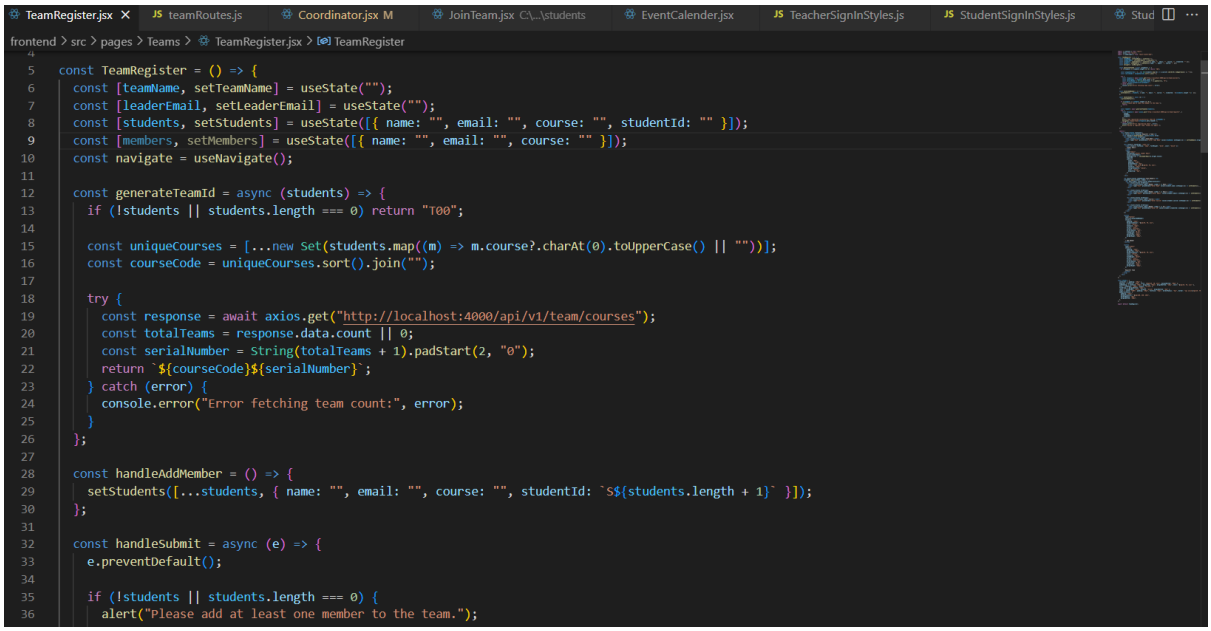
Email: <a href="mailto:deanadmin@banasthali.ac.in" style="color: #f1c40f; t
Contact: <a href="tel:+911438228456" style="color: #f1c40f; text-decoratior
© 2025 Banasthali Vidyapith. All r

`

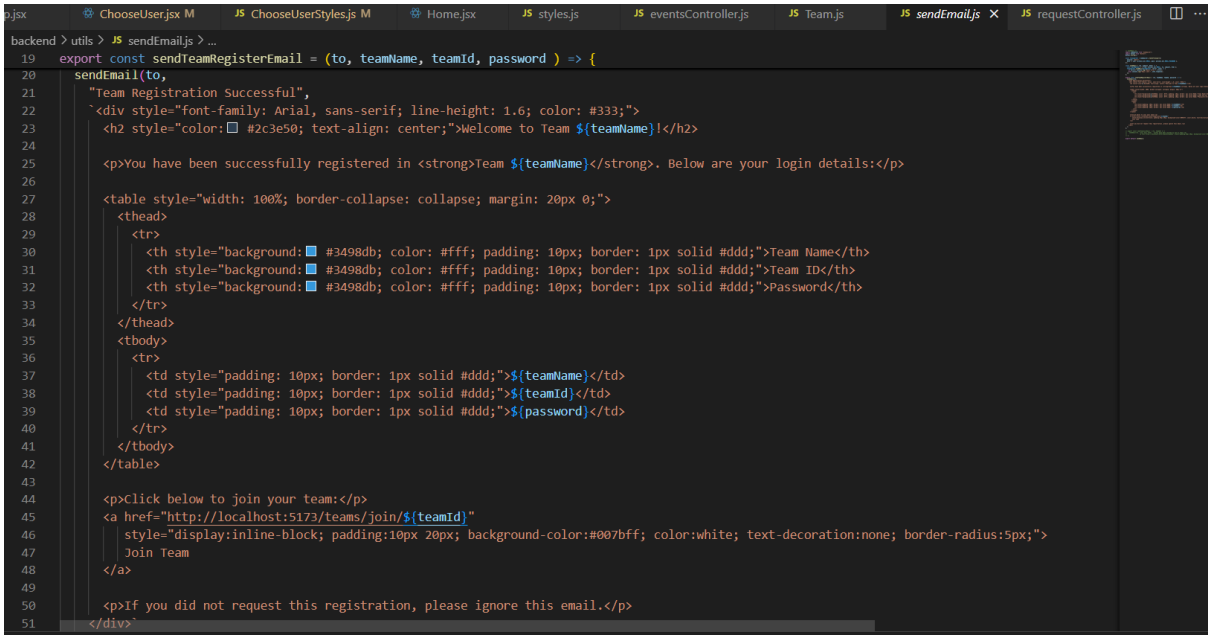

File Edit Selection View
MMSystem
EXPLORER
MMSYSTEM
backend
routes
teamRoutes.js
services
emailservice.js
otpservice.js
uploads
utils
.env
.gitignore
app.js
fileupload.js
index.js
package-lock.json
package.json
frontend
node_modules
public
OUTLINE
TIMELINE
TeacherSignIn.jsx
StudentSignIn.jsx
Home.jsx
Students.jsx
emailservice.js
backend > services > emailservice.js > ...
// Send Student Email
export const sendStudentEmail = async (recipientEmail) => {
try {
const mailOptions = {
from: '"Mentor-Mentee Portal" <himanidobriyal8@gmail.com>',
to: recipientEmail,
subject: "Welcome to Mentor-Mentee Portal (Student)",
html: getEmailTemplate(
"Welcome, Student!",
"You have successfully logged in as a Student.",
"Student",
"http://localhost:5173/student/dashboard"
)
};
const info = await transporter.sendMail(mailOptions);
console.log("Student Email sent: " + info.response);
return { success: true, info };
} catch (error) {
console.error("Error sending student email:", error);
return { success: false, error };
}
};



TEAM REGISTER

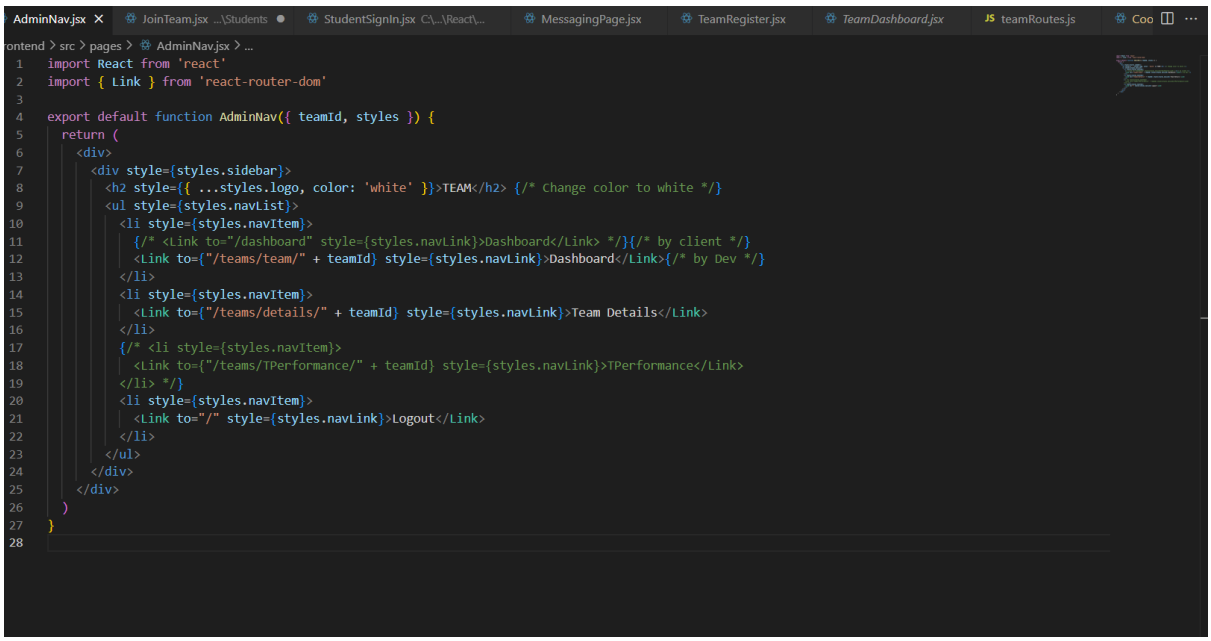


TEAM MAIL



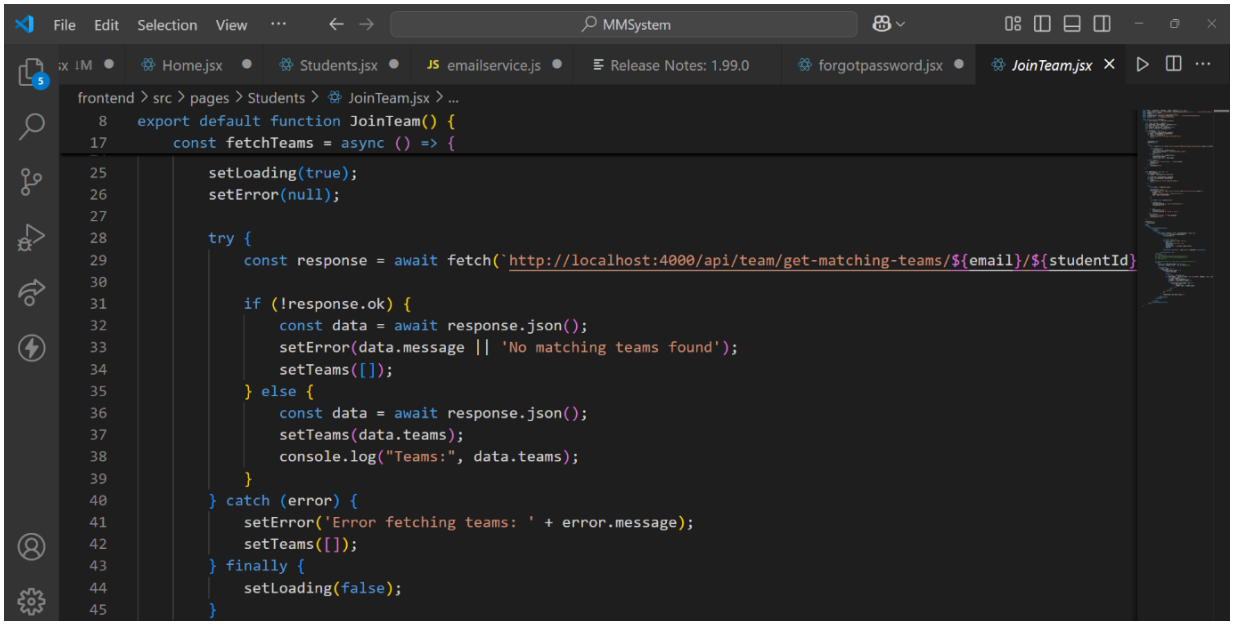
```
export const sendTeamRegisterEmail = (to, teamName, teamId, password ) => {
  sendEmail(to,
    "Team Registration Successful",
    `<div style="font-family: Arial, sans-serif; line-height: 1.6; color: #333;">
      <h2 style="color: #2c3e50; text-align: center;">Welcome to Team ${teamName}!</h2>

      <p>You have been successfully registered in <strong>Team ${teamName}</strong>. Below are your login details:</p>

      <table style="width: 100%; border-collapse: collapse; margin: 20px 0;">
        <thead>
          <tr>
            <th style="background: #3498db; color: #fff; padding: 10px; border: 1px solid #ddd;">Team Name</th>
            <th style="background: #3498db; color: #fff; padding: 10px; border: 1px solid #ddd;">Team ID</th>
            <th style="background: #3498db; color: #fff; padding: 10px; border: 1px solid #ddd;">Password</th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td style="padding: 10px; border: 1px solid #ddd;">${teamName}</td>
            <td style="padding: 10px; border: 1px solid #ddd;">${teamId}</td>
            <td style="padding: 10px; border: 1px solid #ddd;">${password}</td>
          </tr>
        </tbody>
      </table>

      <p>Click below to join your team:</p>
      <a href="http://localhost:5173/teams/join/${teamId}"
        style="display:inline-block; padding:10px 20px; background-color:#007bff; color:white; text-decoration:none; border-radius:5px;">
        Join Team
      </a>

      <p>If you did not request this registration, please ignore this email.</p>
    </div>`
```

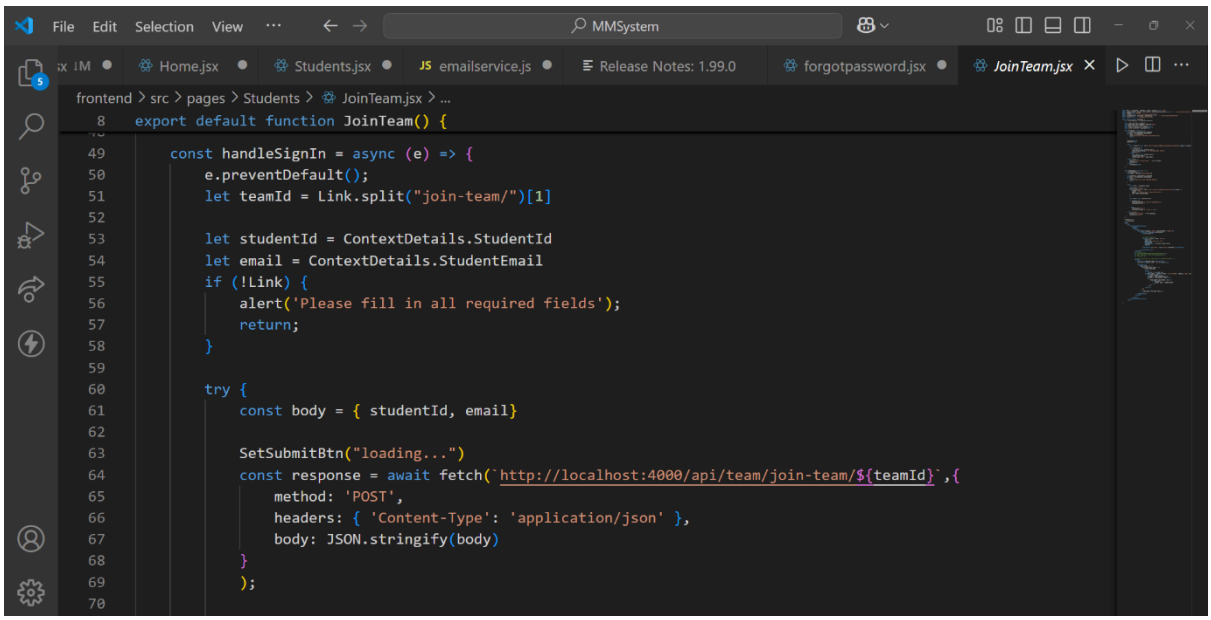
TEAM DASHBOARD

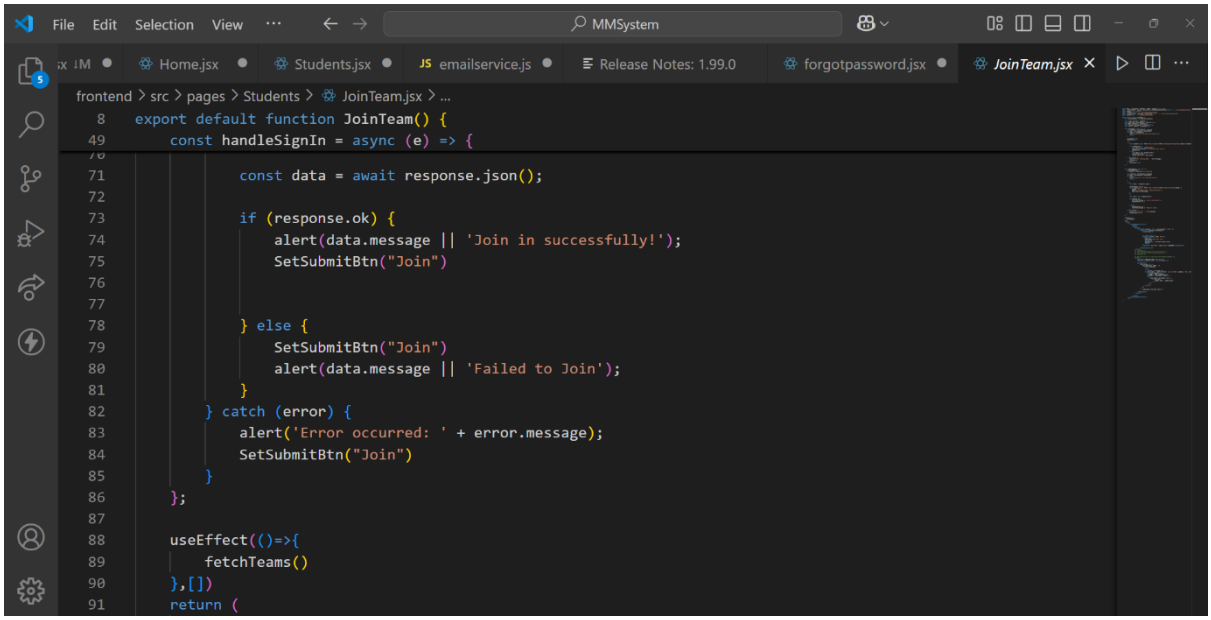


```
import React from 'react'
import { Link } from 'react-router-dom'

export default function AdminNav({ teamId, styles }) {
  return (
    <div>
      <div style={styles.sidebar}>
        <h2 style={{ ...styles.logo, color: 'white' }}>TEAM</h2> {/* Change color to white */}
        <ul style={styles.navList}>
          <li style={styles.navItem}>
            {/* <Link to="/dashboard" style={styles.navLink}>Dashboard</Link> */}{/* by client */}
            <Link to={"/teams/team/" + teamId} style={styles.navLink}>Dashboard</Link>{/* by Dev */}
          </li>
          <li style={styles.navItem}>
            <Link to={"/teams/details/" + teamId} style={styles.navLink}>Team Details</Link>
          </li>
          {/* <li style={styles.navItem}>
            <Link to={"/teams/TPerformance/" + teamId} style={styles.navLink}>TPerformance</Link>
          </li> */}
          <li style={styles.navItem}>
            <Link to="/" style={styles.navLink}>Logout</Link>
          </li>
        </ul>
      </div>
    </div>
  )
}
```

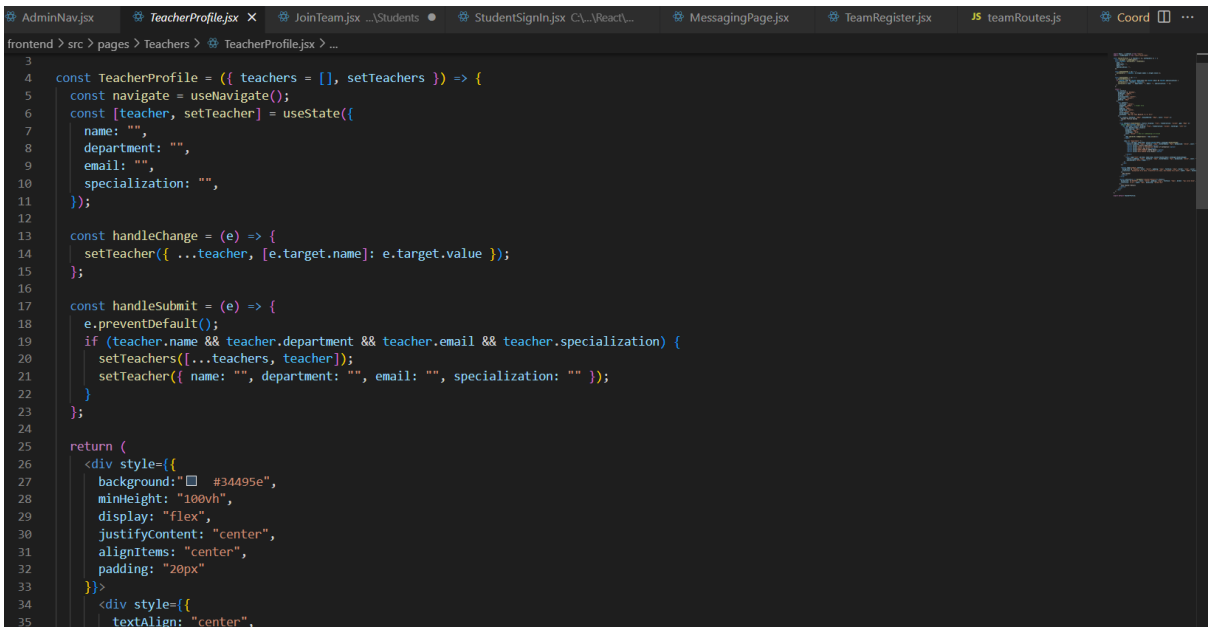
JOIN TEAM


File Edit Selection View
MMSystem
Home.jsx
Students.jsx
emailservice.js
Release Notes: 1.99.0
forgotpassword.jsx
JoinTeam.jsx
frontend > src > pages > Students > JoinTeam.jsx > ...
export default function JoinTeam() {
const fetchTeams = async () => {
setLoading(true);
setError(null);
try {
const response = await fetch(`http://localhost:4000/api/team/get-matching-teams/${email}/${studentId}
if (!response.ok) {
const data = await response.json();
setError(data.message || 'No matching teams found');
setTeams([]);
} else {
const data = await response.json();
setTeams(data.teams);
console.log("Teams:", data.teams);
}
} catch (error) {
setError('Error fetching teams: ' + error.message);
setTeams([]);
} finally {
setLoading(false);
}

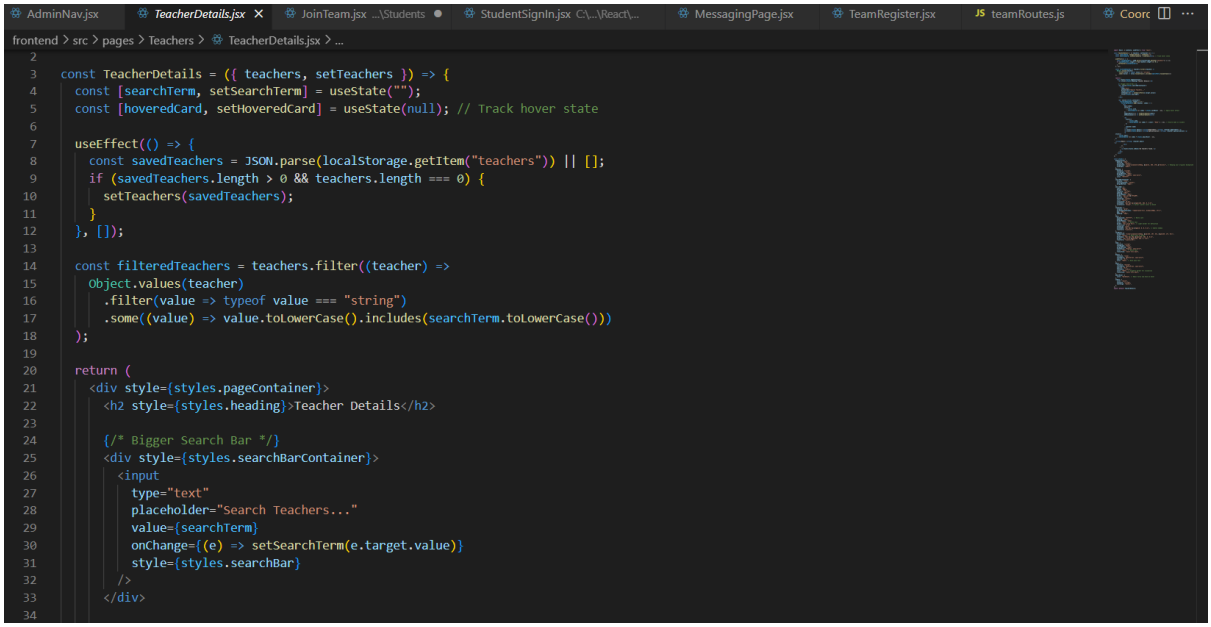

File Edit Selection View
MMSystem
Home.jsx
Students.jsx
emailservice.js
Release Notes: 1.99.0
forgotpassword.jsx
JoinTeam.jsx
frontend > src > pages > Students > JoinTeam.jsx > ...
export default function JoinTeam() {
const handleSignIn = async (e) => {
e.preventDefault();
let teamId = Link.split("join-team/")[1]
let studentId = ContextDetails.StudentId
let email = ContextDetails.StudentEmail
if (!Link) {
alert('Please fill in all required fields');
return;
}
try {
const body = { studentId, email}
SetSubmitBtn("loading...")
const response = await fetch(`http://localhost:4000/api/team/join-team/${teamId}`,{
method: 'POST',
headers: { 'Content-Type': 'application/json' },
body: JSON.stringify(body)
}
);



TEACHER PROFILE

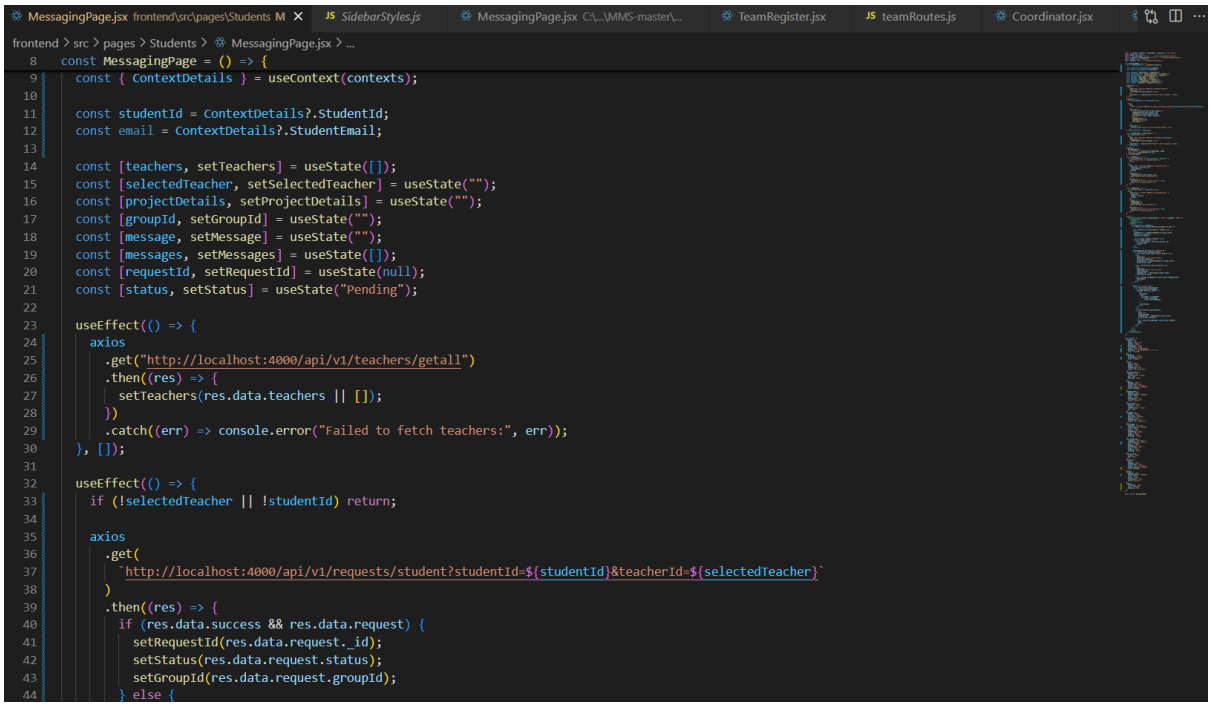


TEACHER DETAILS

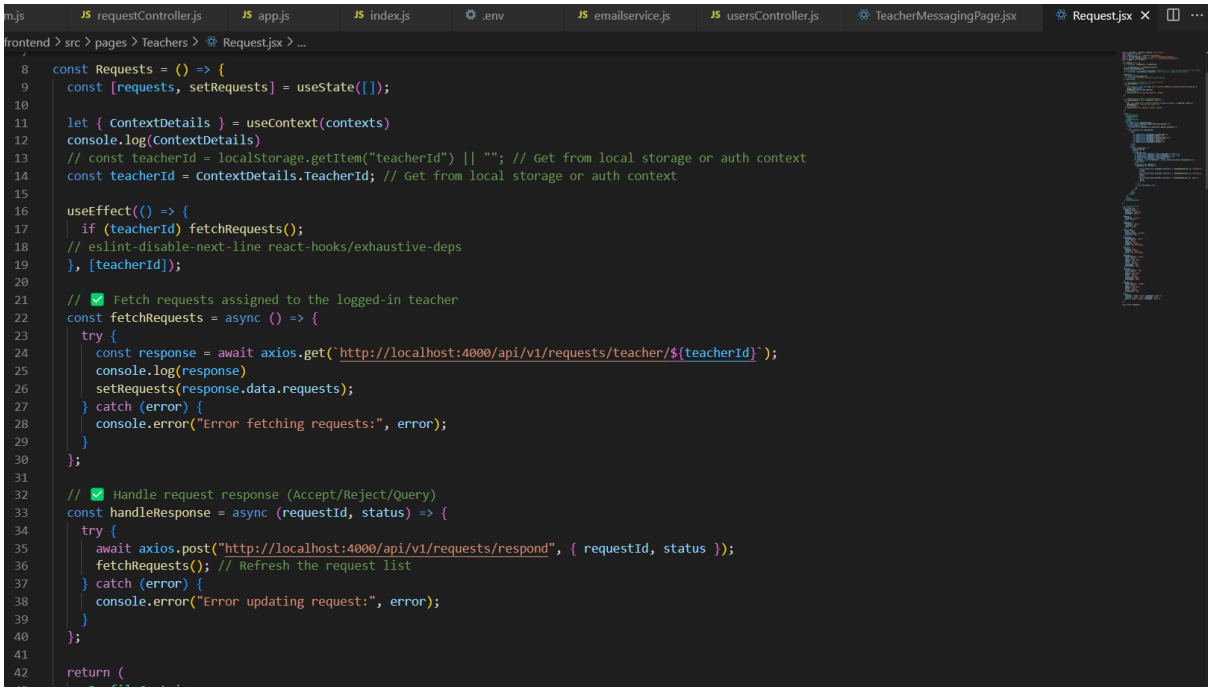


```
const TeacherDetails = ({ teachers, setTeachers }) => {
  const [searchTerm, setSearchTerm] = useState("");
  const [hoveredCard, setHoveredCard] = useState(null); // Track hover state

  useEffect(() => {
    const savedTeachers = JSON.parse(localStorage.getItem("teachers")) || [];
    if (savedTeachers.length > 0 && teachers.length === 0) {
      setTeachers(savedTeachers);
    }
  }, []);

  const filteredTeachers = teachers.filter((teacher) =>
    Object.values(teacher)
      .filter(value => typeof value === "string")
      .some((value) => value.toLowerCase().includes(searchTerm.toLowerCase()))
  );

  return (
    <div style={styles.pageContainer}>
      <h2 style={styles.heading}>Teacher Details</h2>

      {/* Bigger Search Bar */}
      <div style={styles.searchBarContainer}>
        <input
          type="text"
          placeholder="Search Teachers..."
          value={searchTerm}
          onChange={(e) => setSearchTerm(e.target.value)}
          style={styles.searchBar}
        />
      </div>
```

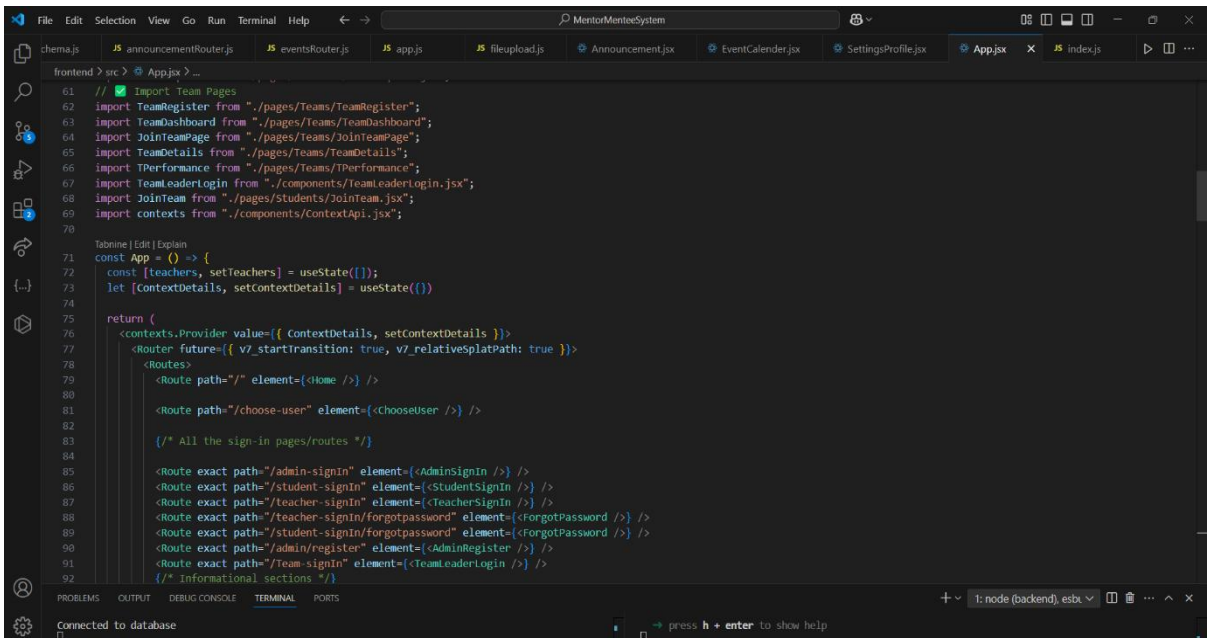
MESSAGING REQUESTS



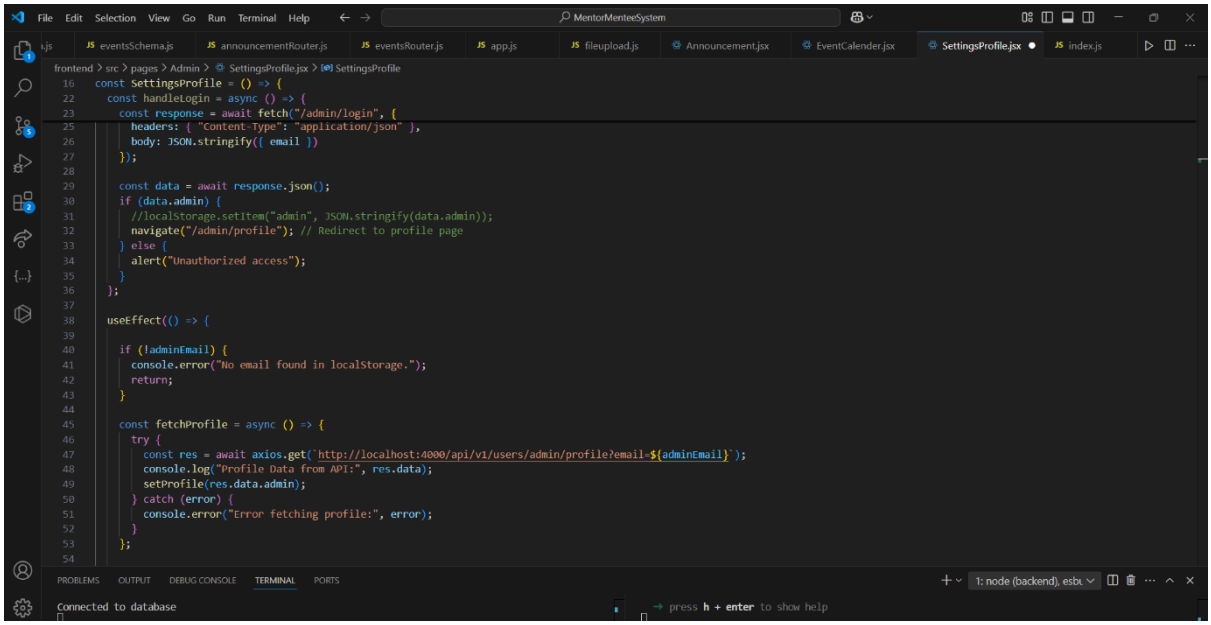
```
const MessagingPage = () => {
  const { ContextDetails } = useContext(contexts);

  const studentId = ContextDetails?.StudentId;
  const email = ContextDetails?.StudentEmail;

  const [teachers, setTeachers] = useState([]);
  const [selectedTeacher, setSelectedTeacher] = useState("");
  const [projectDetails, setProjectDetails] = useState("");
  const [groupId, setGroupId] = useState("");
  const [message, setMessage] = useState("");
  const [messages, setMessages] = useState([]);
  const [requestId, setRequestId] = useState(null);
  const [status, setStatus] = useState("Pending");

  useEffect(() => {
    axios
      .get("http://localhost:4000/api/v1/teachers/getall")
      .then((res) => {
        setTeachers(res.data.teachers || []);
      })
      .catch((err) => console.error("Failed to fetch teachers:", err));
  }, []);

  useEffect(() => {
    if (!selectedTeacher || !studentId) return;

    axios
      .get(
        `http://localhost:4000/api/v1/requests/student?studentId=${studentId}&teacherId=${selectedTeacher}`
      )
      .then((res) => {
        if (res.data.success && res.data.request) {
          setRequestId(res.data.request._id);
          setStatus(res.data.request.status);
          setGroupId(res.data.request.groupId);
        } else {
```



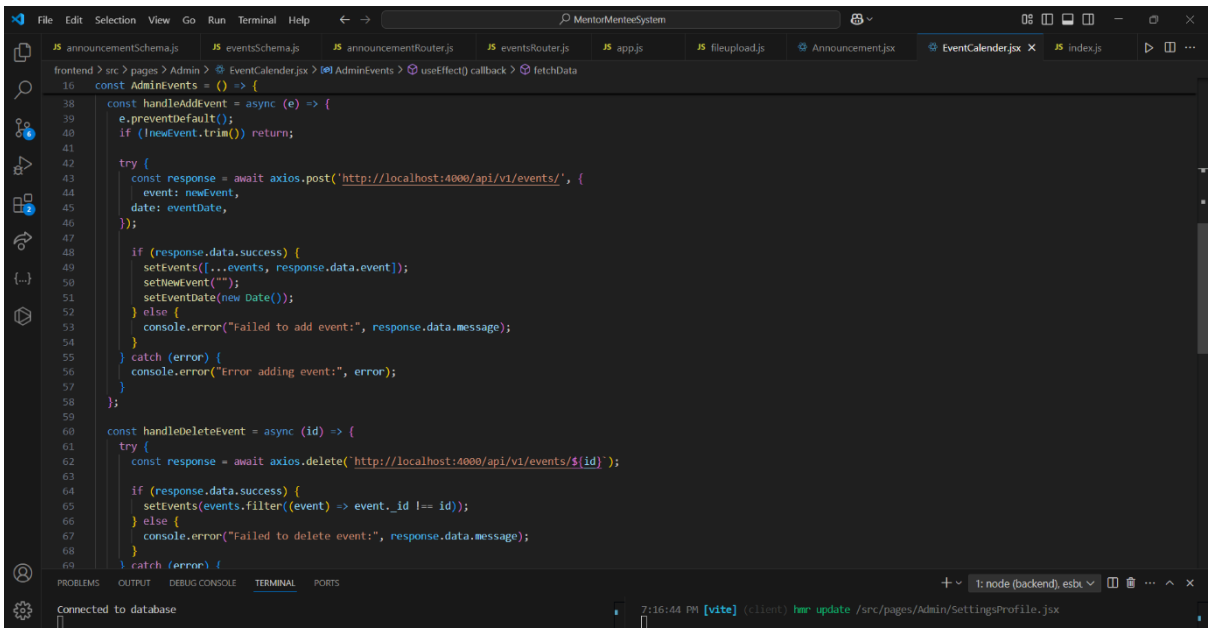
FRONTEND ROUTER



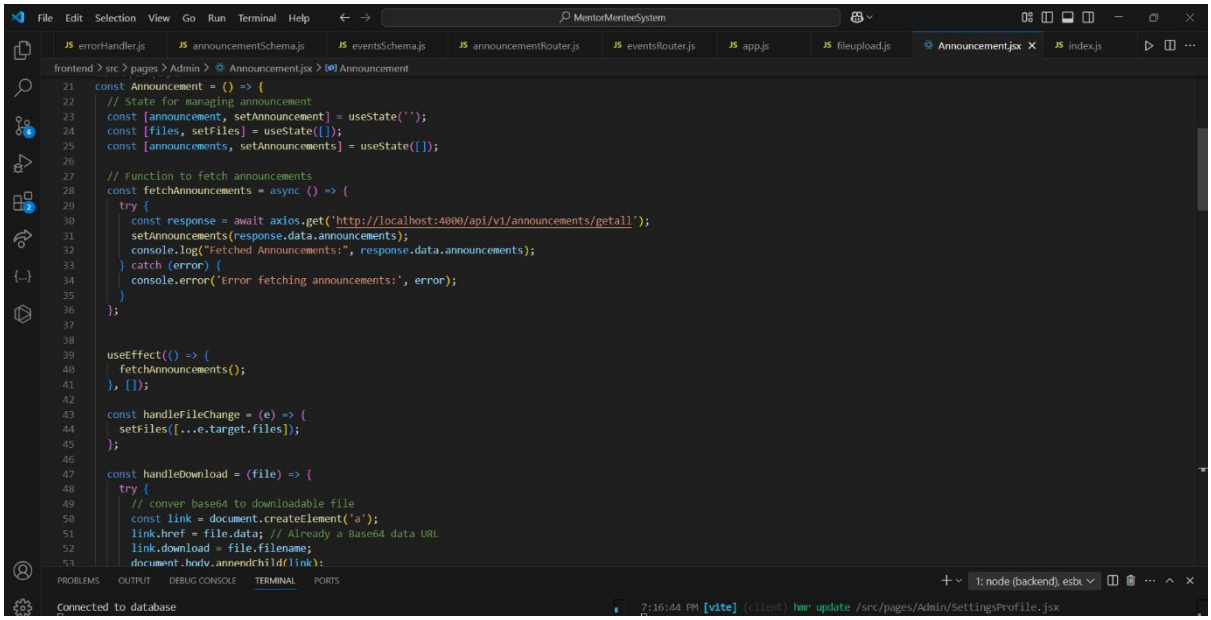
LOGGED IN USER DETAILS



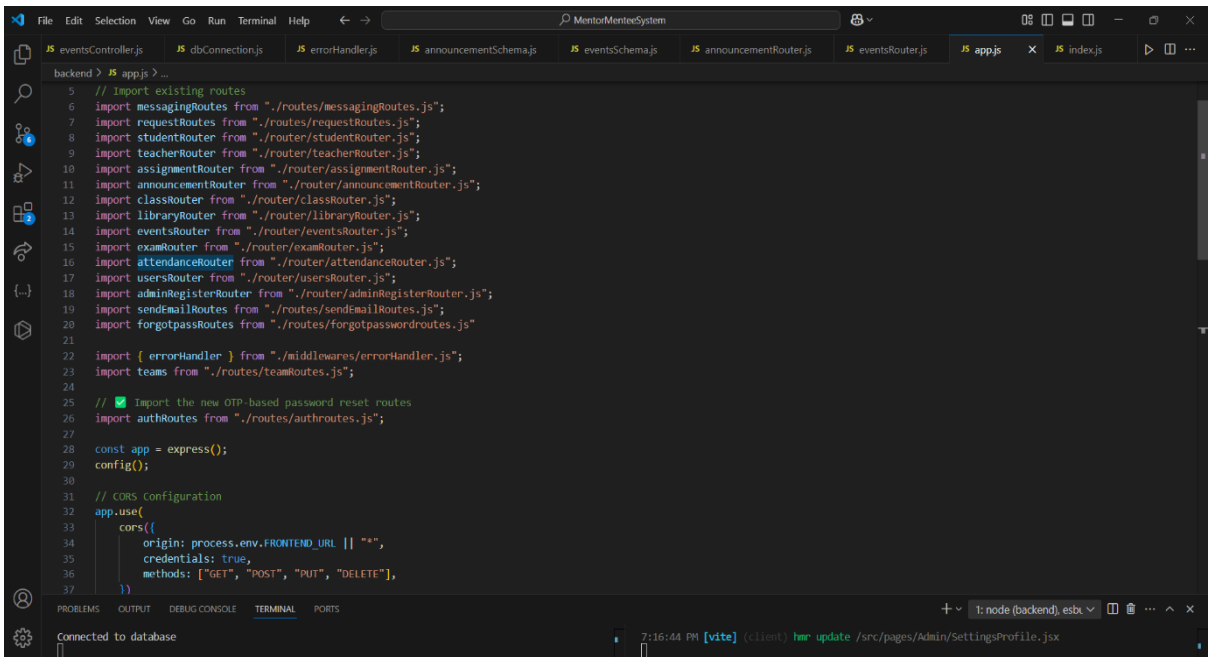
EVENT FRONTEND



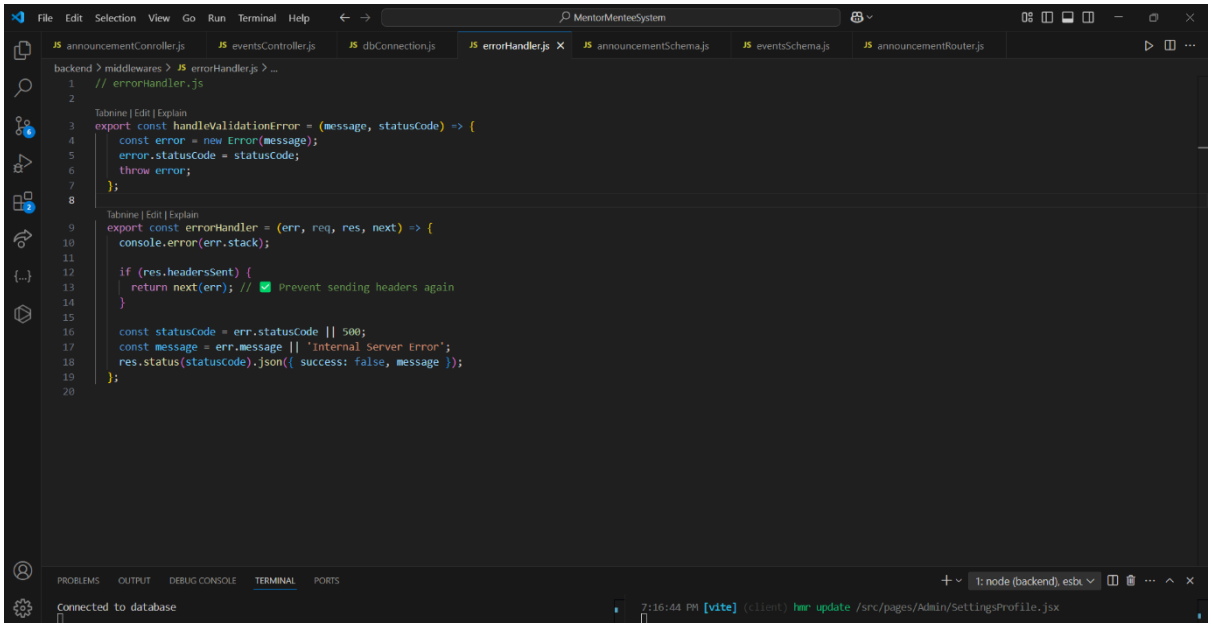
ANNOUNCEMENT FRONTEND



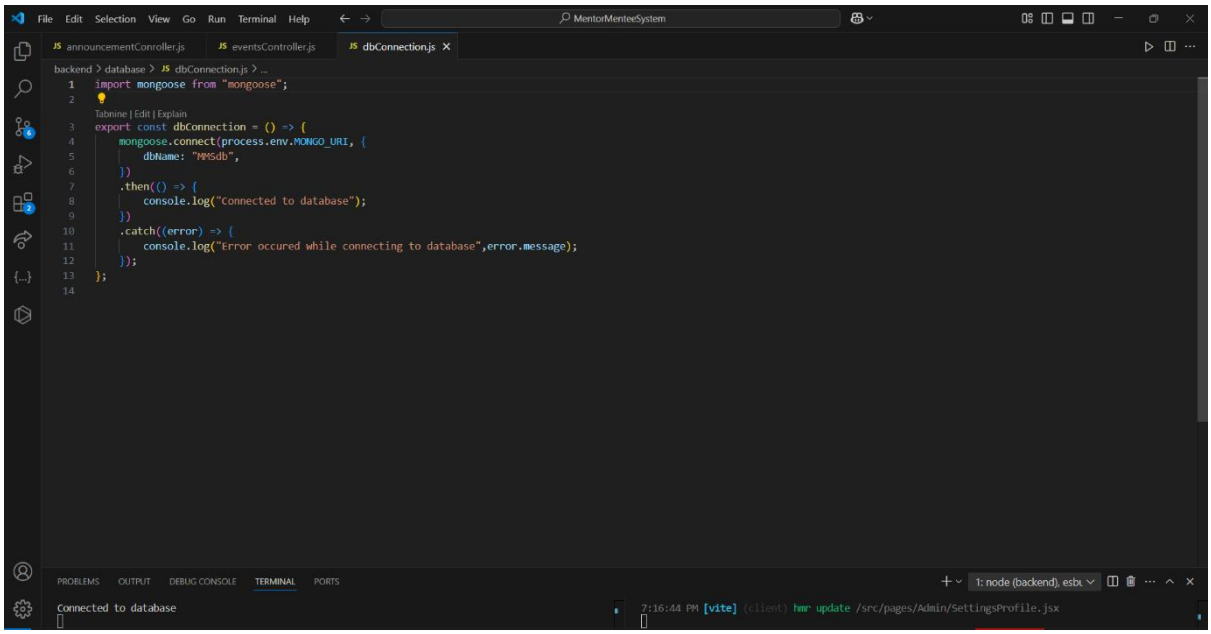
BACKEND ROUTER



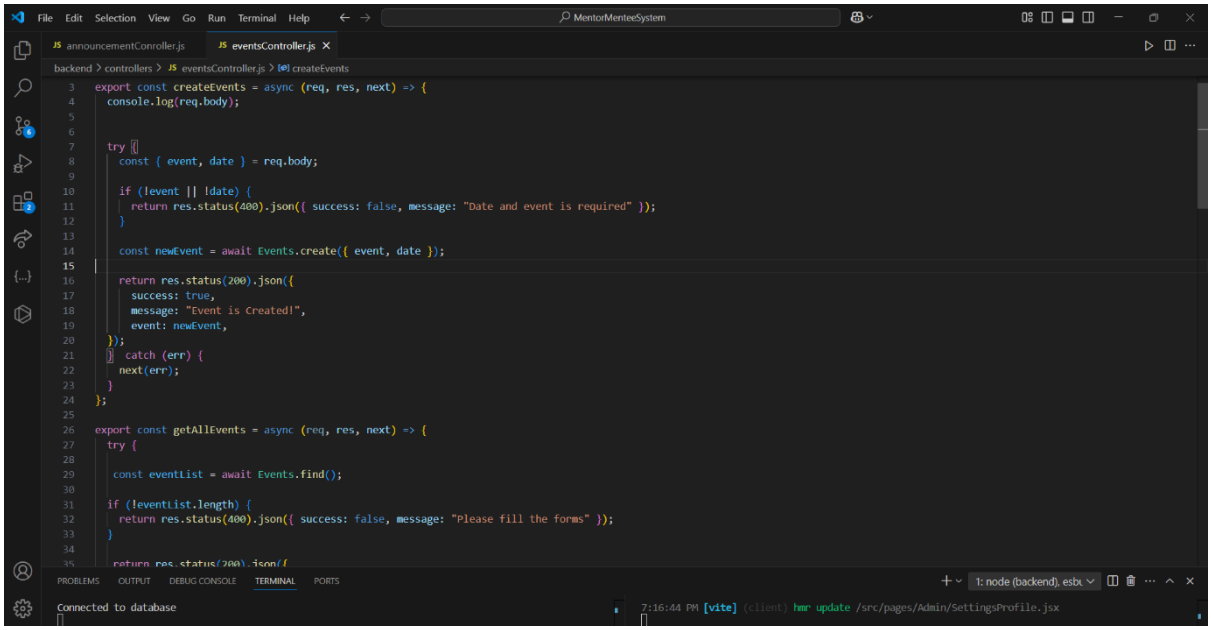
EVENT ROUTER



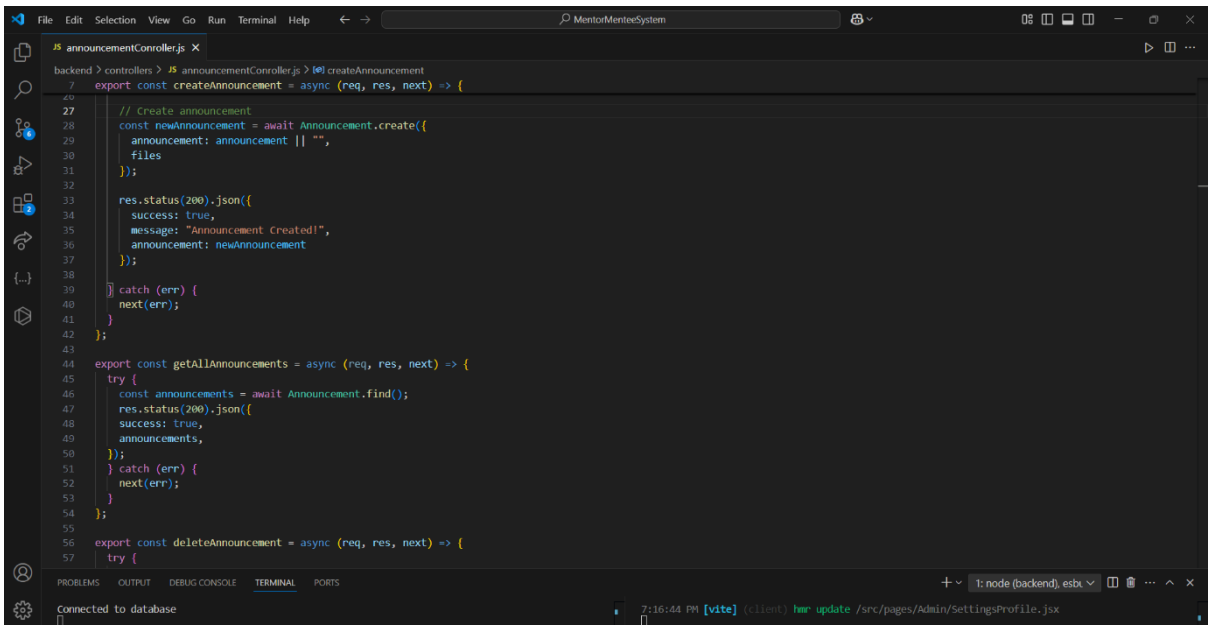
DATABASE CONNECTION



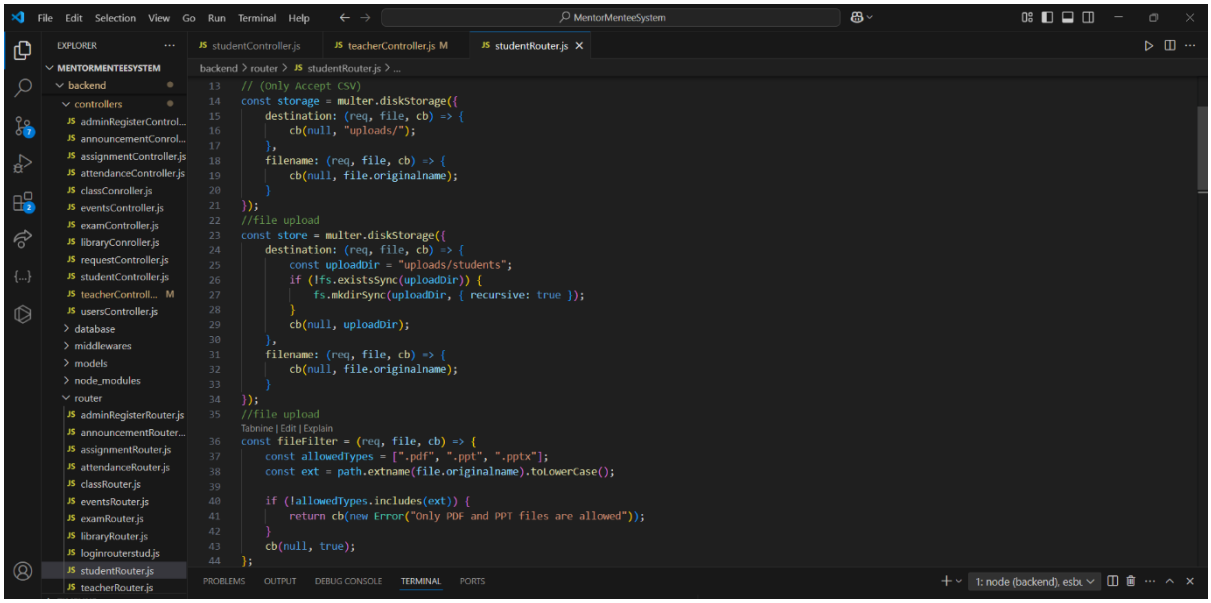
EVENT CONTROLLER



ANNOUNCEMENT CONTROLLER

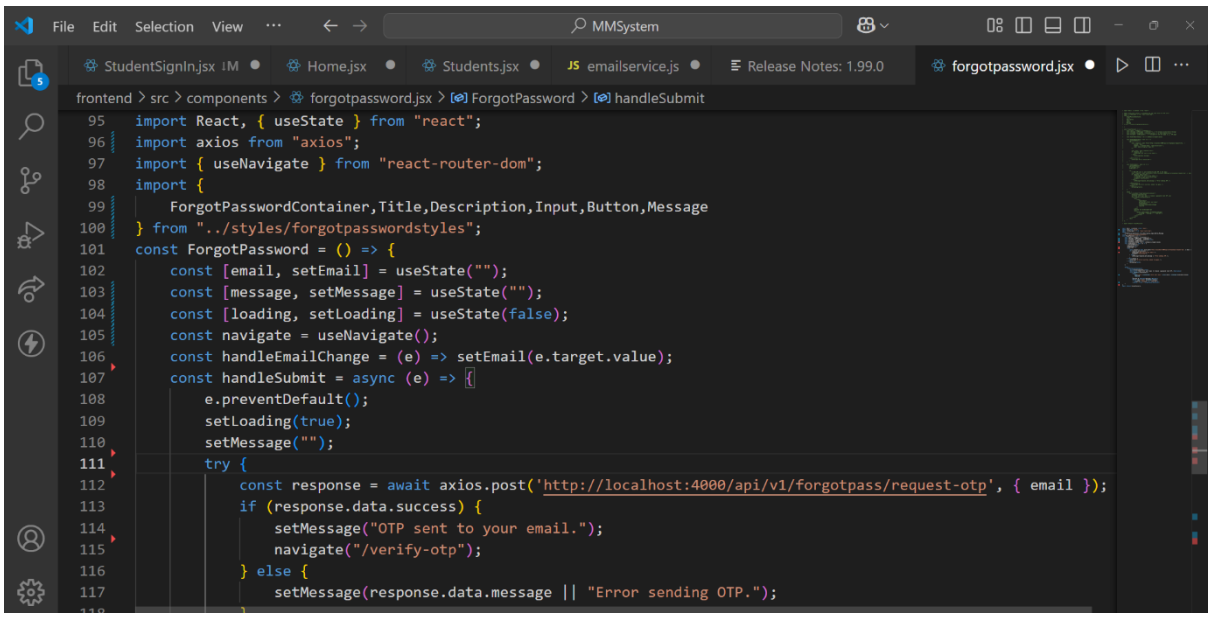


CSV UPLOAD FILE

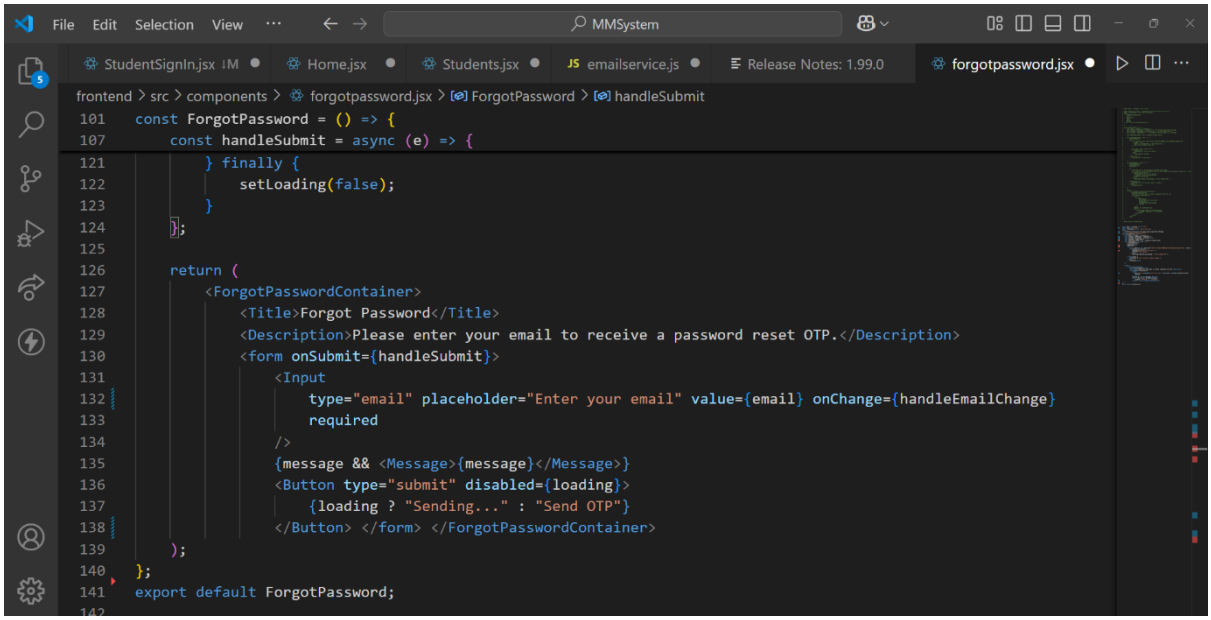


```
// (Only Accept CSV)
const storage = multer.diskStorage({
    destination: (req, file, cb) => {
        cb(null, "uploads/");
    },
    filename: (req, file, cb) => {
        cb(null, file.originalname);
    }
});
//file upload
const store = multer.diskStorage({
    destination: (req, file, cb) => {
        const uploadDir = "uploads/students";
        if (!fs.existsSync(uploadDir)) {
            fs.mkdirSync(uploadDir, { recursive: true });
        }
        cb(null, uploadDir);
    },
    filename: (req, file, cb) => {
        cb(null, file.originalname);
    }
});
//file upload
Tabnine | Edit | Explain
const fileFilter = (req, file, cb) => {
    const allowedTypes = [".pdf", ".ppt", ".pptx"];
    const ext = path.extname(file.originalname).toLowerCase();

    if (!allowedTypes.includes(ext)) {
        return cb(new Error("Only PDF and PPT files are allowed"));
    }
    cb(null, true);
};
```

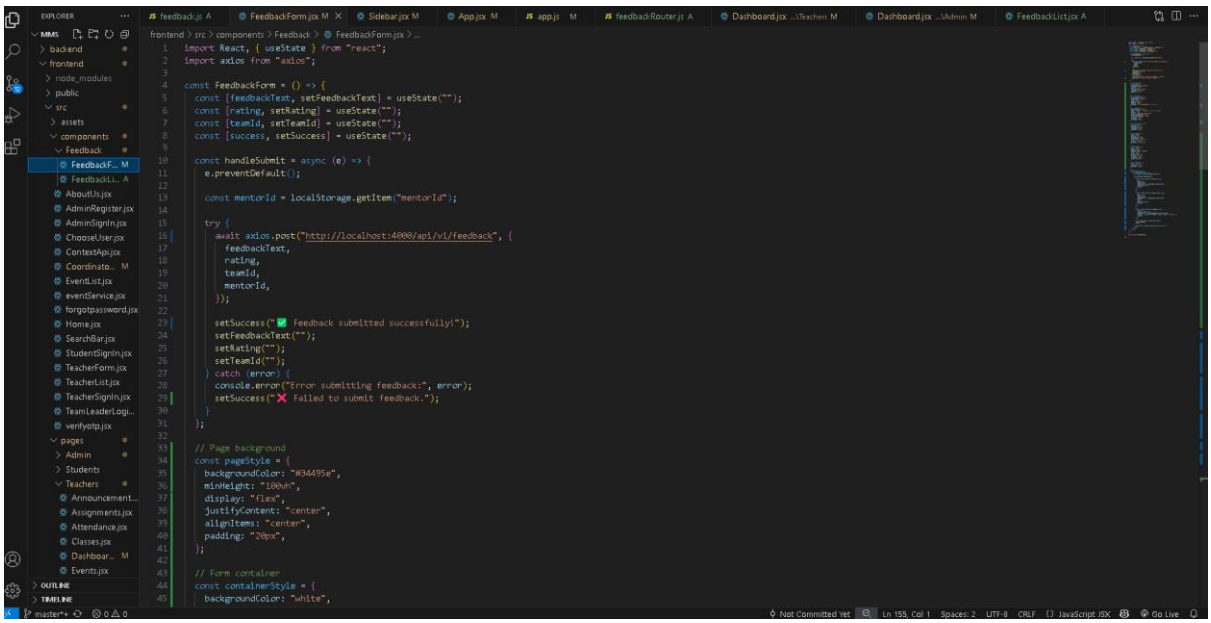
FORGOT PASSWORD

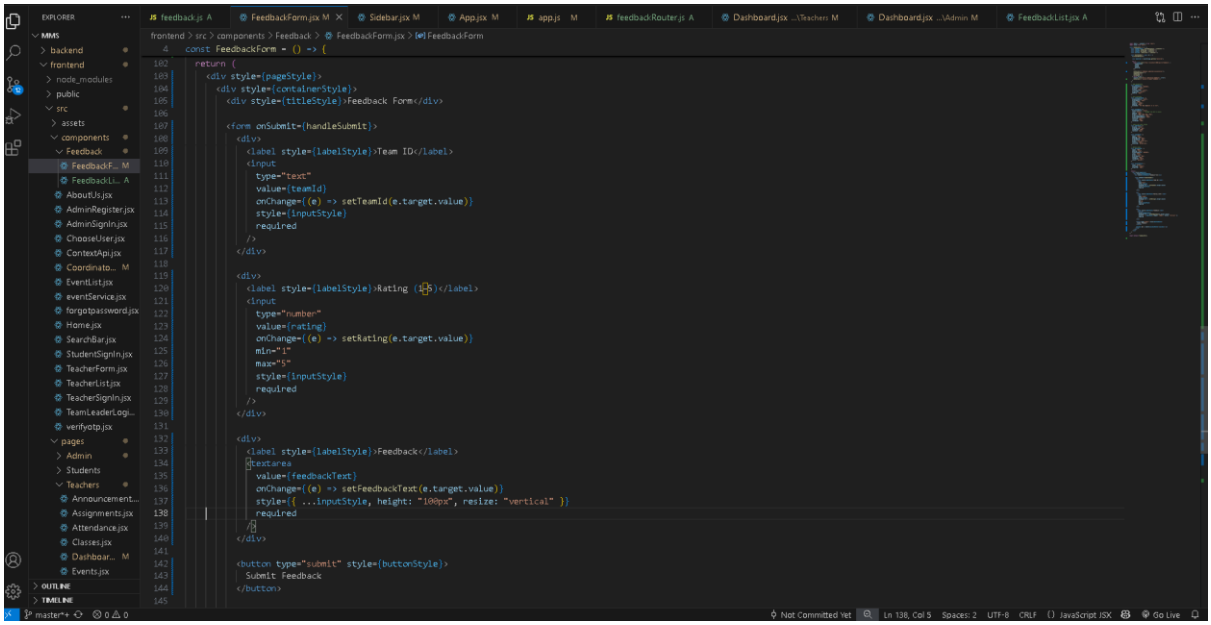


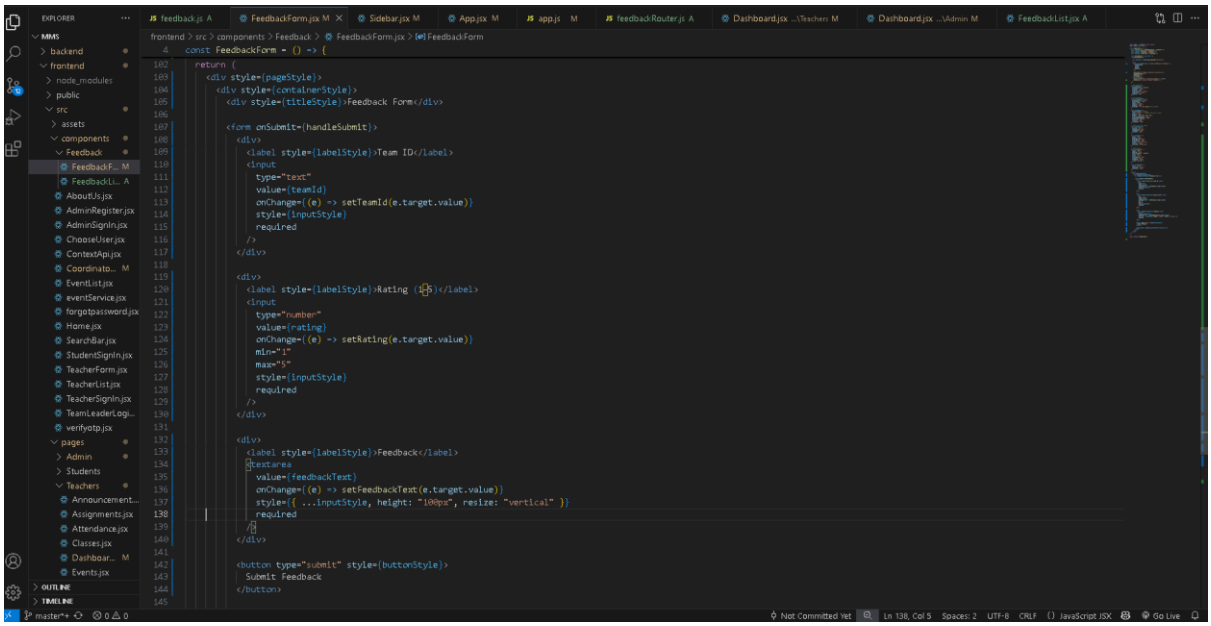
```
import React, { useState } from "react";
import axios from "axios";
import { useNavigate } from "react-router-dom";
import {
    ForgotPasswordContainer,Title,Description,Input,Button,Message
} from "../styles/forgotpasswordstyles";
const ForgotPassword = () => {
    const [email, setEmail] = useState("");
    const [message, setMessage] = useState("");
    const [loading, setLoading] = useState(false);
    const navigate = useNavigate();
    const handleEmailChange = (e) => setEmail(e.target.value);
    const handleSubmit = async (e) => {
        e.preventDefault();
        setLoading(true);
        setMessage("");
        try {
            const response = await axios.post('http://localhost:4000/api/v1/forgotpass/request-otp', { email });
            if (response.data.success) {
                setMessage("OTP sent to your email.");
                navigate("/verify-otp");
            } else {
                setMessage(response.data.message || "Error sending OTP.");
```



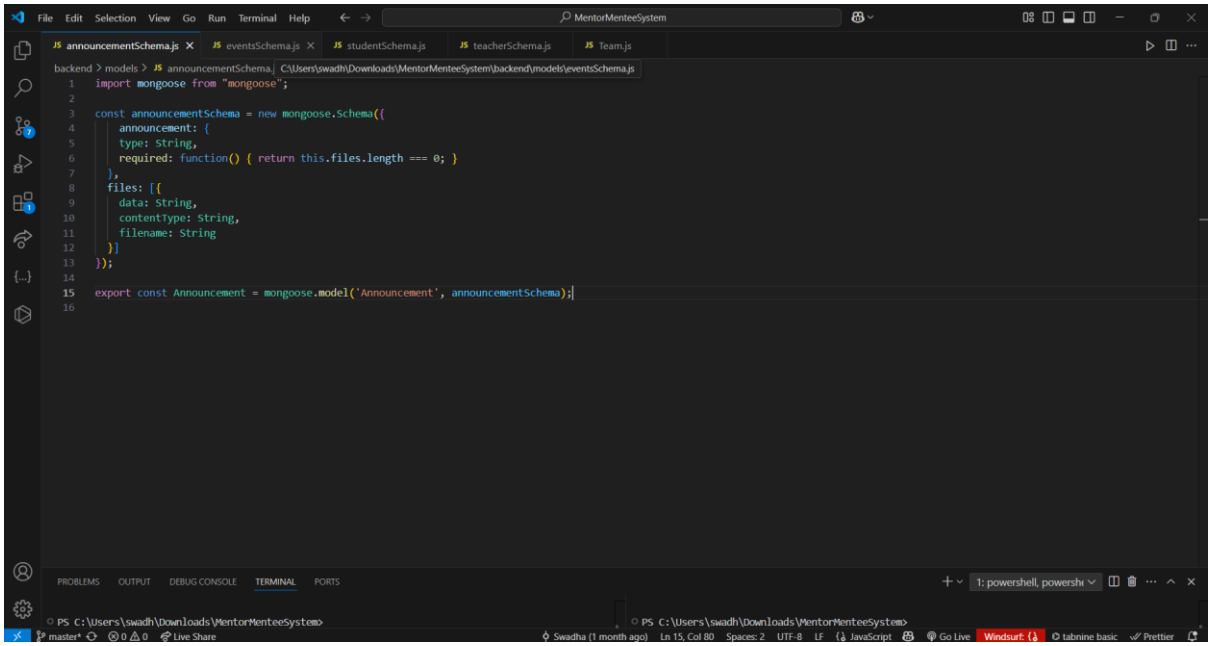
FEEDBACK FORM





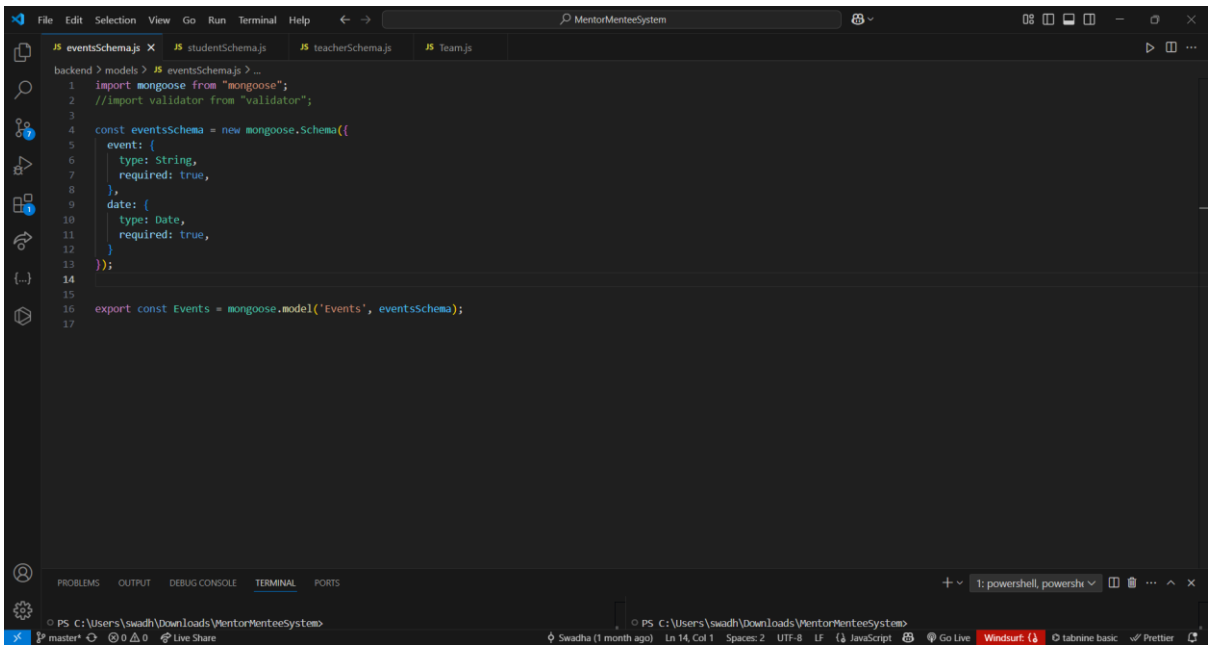


ANNOUNCEMENT SCHEMA



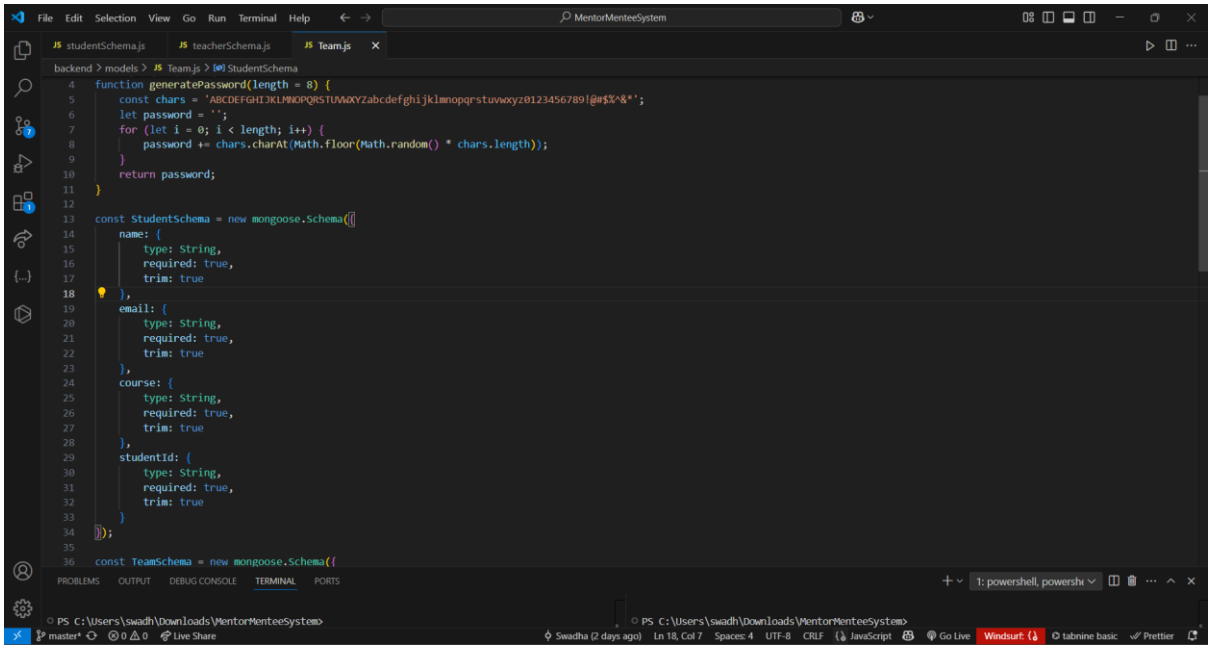
```
import mongoose from "mongoose";

const announcementSchema = new mongoose.Schema({
    announcement: {
    type: String,
    required: function() { return this.files.length === 0; }
  },
  files: [{
    data: String,
    contentType: String,
    filename: String
  }]
});

export const Announcement = mongoose.model('Announcement', announcementSchema);
```

EVENT SCHEMA

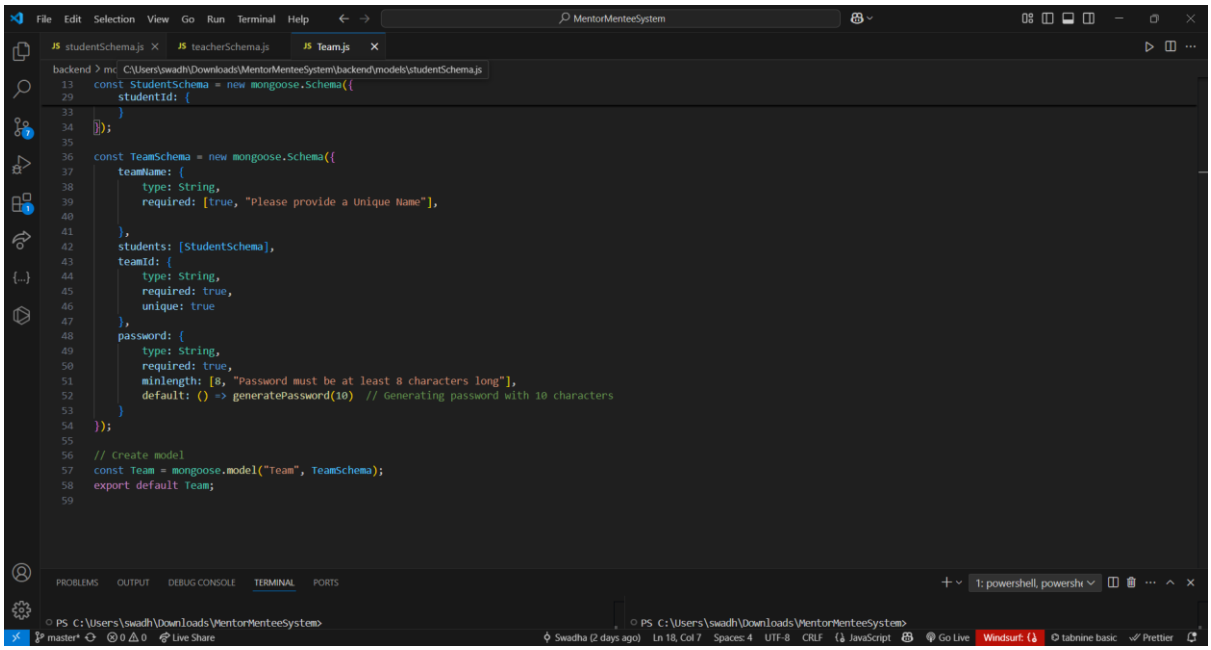


```
import mongoose from "mongoose";
//import validator from "validator";

const eventsSchema = new mongoose.Schema({
  event: {
    type: String,
    required: true,
  },
  date: {
    type: Date,
    required: true,
  }
});

export const Events = mongoose.model('Events', eventsSchema);
```

TEAM SCHEMA

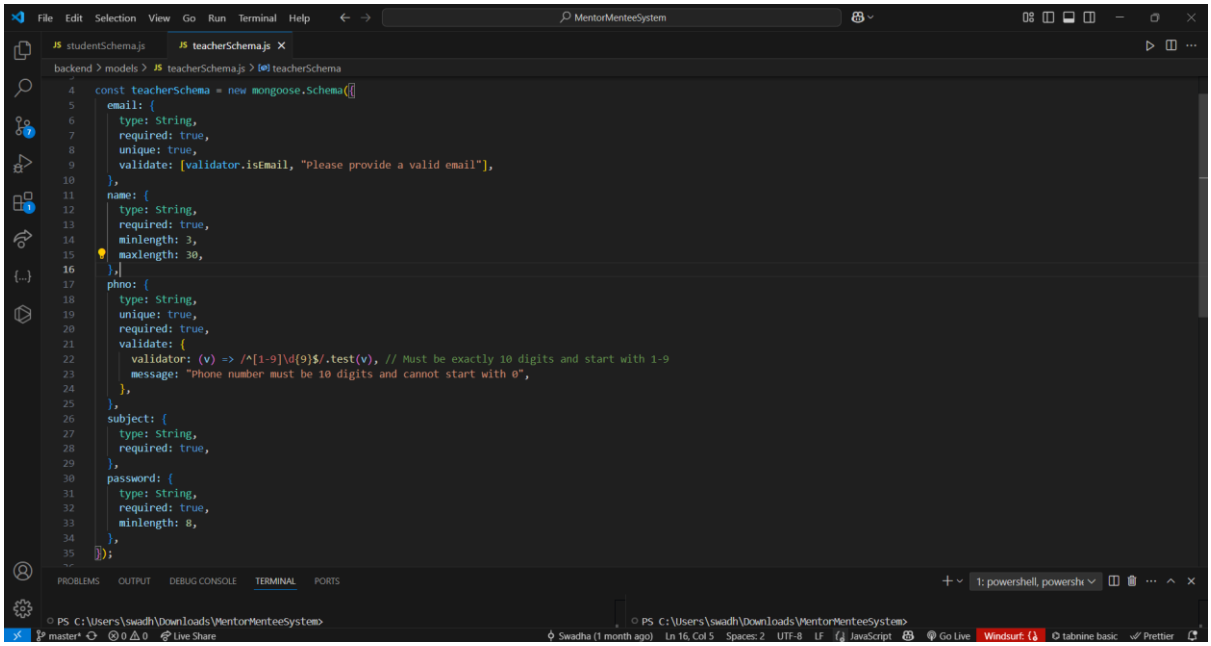


```
function generatePassword(length = 8) {
    const chars = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$%^&*';
    let password = '';
    for (let i = 0; i < length; i++) {
        password += chars.charAt(Math.floor(Math.random() * chars.length));
    }
    return password;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

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({
```



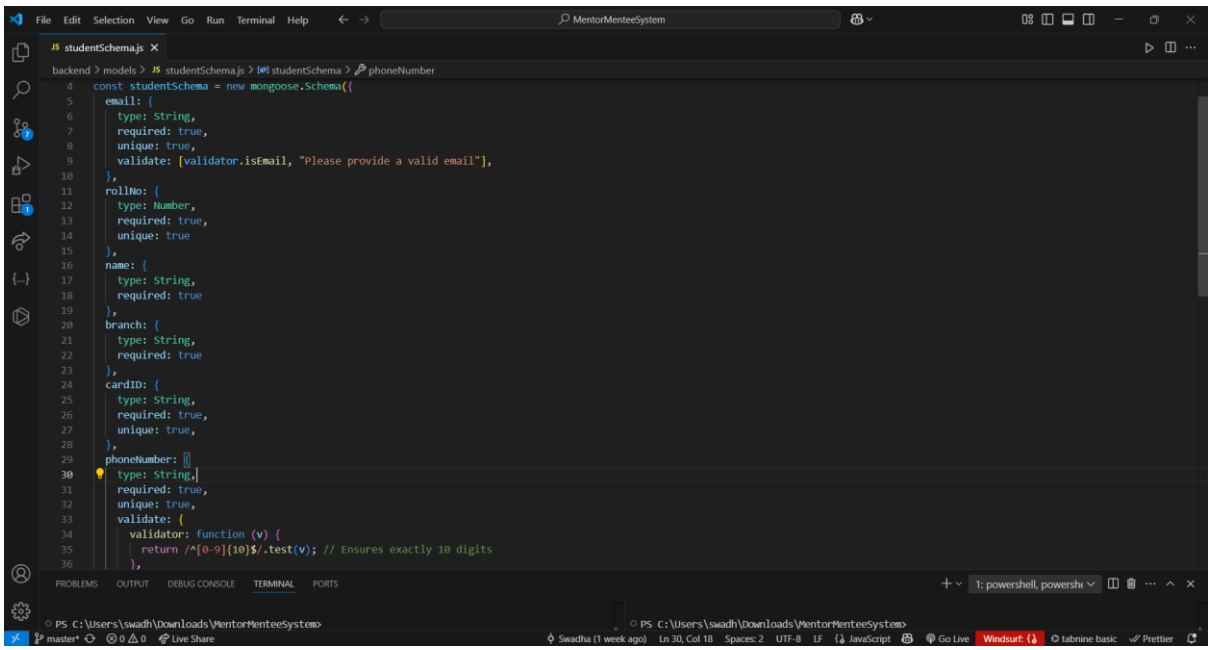
```
const StudentSchema = new mongoose.Schema({
    studentId: {
    }
});

const TeamSchema = new mongoose.Schema({
    teamName: {
        type: String,
        required: [true, "Please provide a Unique Name"],

    },
    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



```
const teacherSchema = new mongoose.Schema({
  email: {
    type: String,
    required: true,
    unique: true,
    validate: [validator.isEmail, "Please provide a valid email"],
  },
  name: {
    type: String,
    required: true,
    minlength: 3,
    maxlength: 30,
  },
  phno: {
    type: String,
    unique: true,
    required: true,
    validate: {
      validator: (v) => /^[1-9]\d{9}$/.test(v), // Must be exactly 10 digits and start with 1-9
      message: "Phone number must be 10 digits and cannot start with 0",
    },
  },
  subject: {
    type: String,
    required: true,
  },
  password: {
    type: String,
    required: true,
    minlength: 8,
  },
});
```

STUDENT SCHEMA



```
const studentSchema = new mongoose.Schema({
  email: {
    type: String,
    required: true,
    unique: true,
    validate: [validator.isEmail, "Please provide a valid email"],
  },
  rollNo: {
    type: Number,
    required: true,
    unique: true
  },
  name: {
    type: String,
    required: true
  },
  branch: {
    type: String,
    required: true
  },
  cardID: {
    type: String,
    required: true,
    unique: true,
  },
  phoneNumber: {
    type: String,
    required: true,
    unique: true,
    validate: {
      validator: function (v) {
        return /^[0-9]{10}$/.test(v); // Ensures exactly 10 digits
      },
```

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**

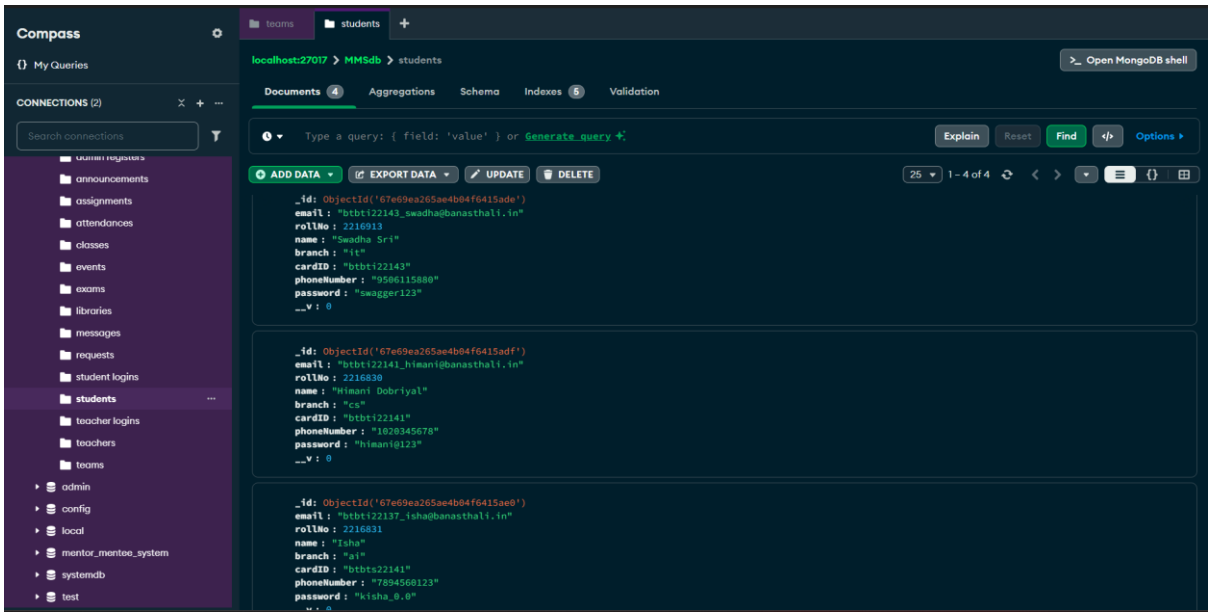
After module cleared the unit testing then modules are tested for their working all together in the integrated testing phase.

3. **Acceptance Testing**

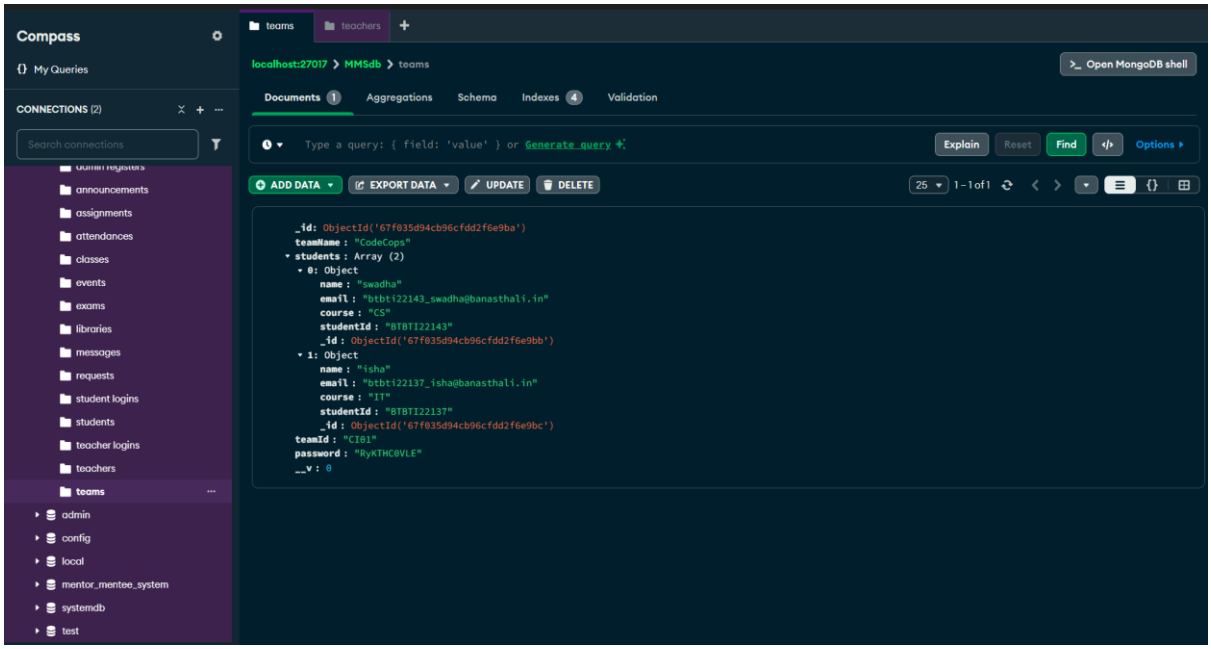
This testing provides the final assurance that the web application needed all behavioural and performance requirements.

With the testing, the working databases are attached.

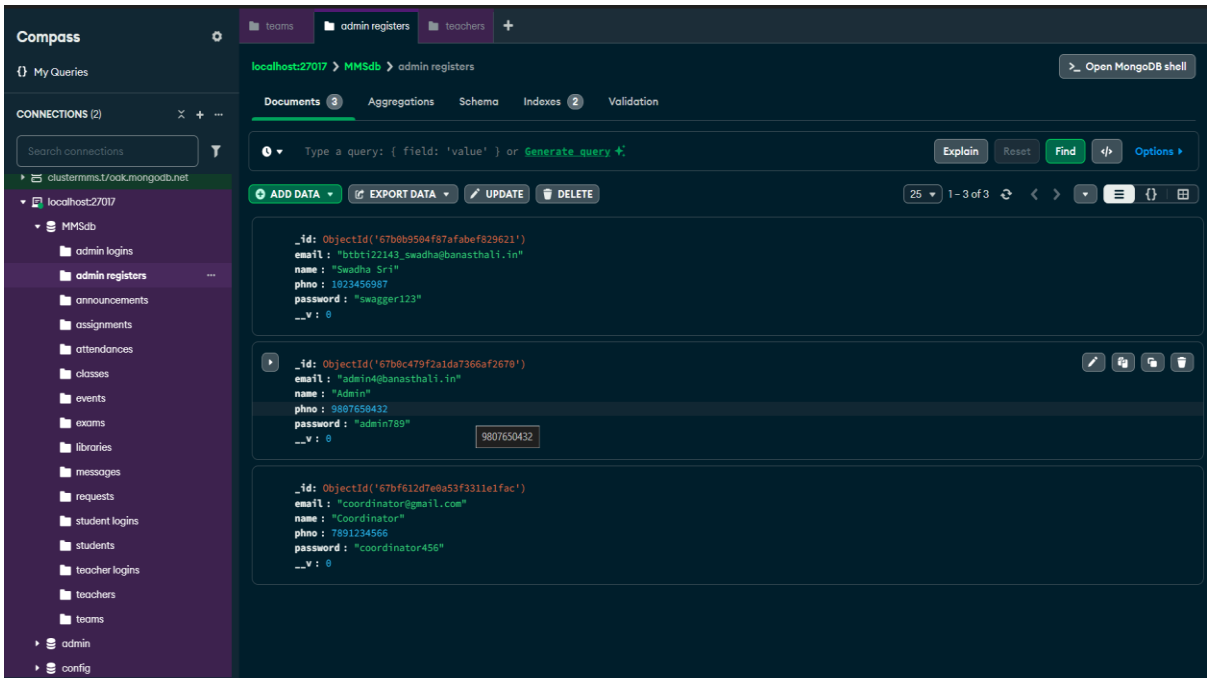
STUDENT DETAILS



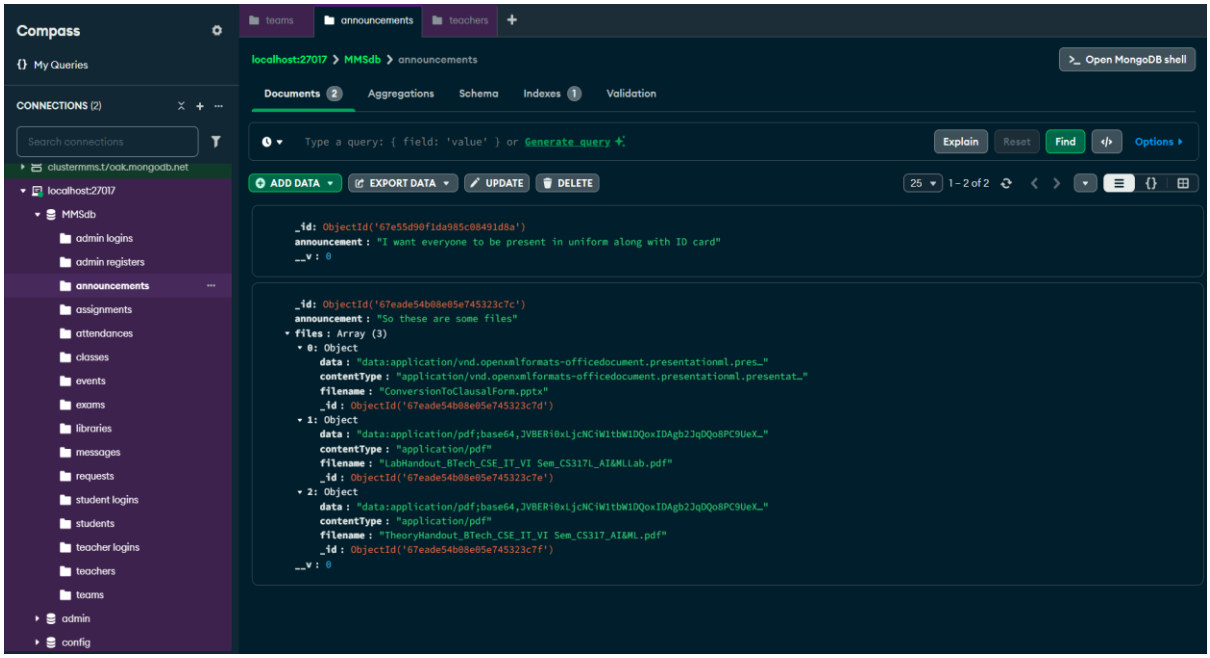
TEAM DETAILS



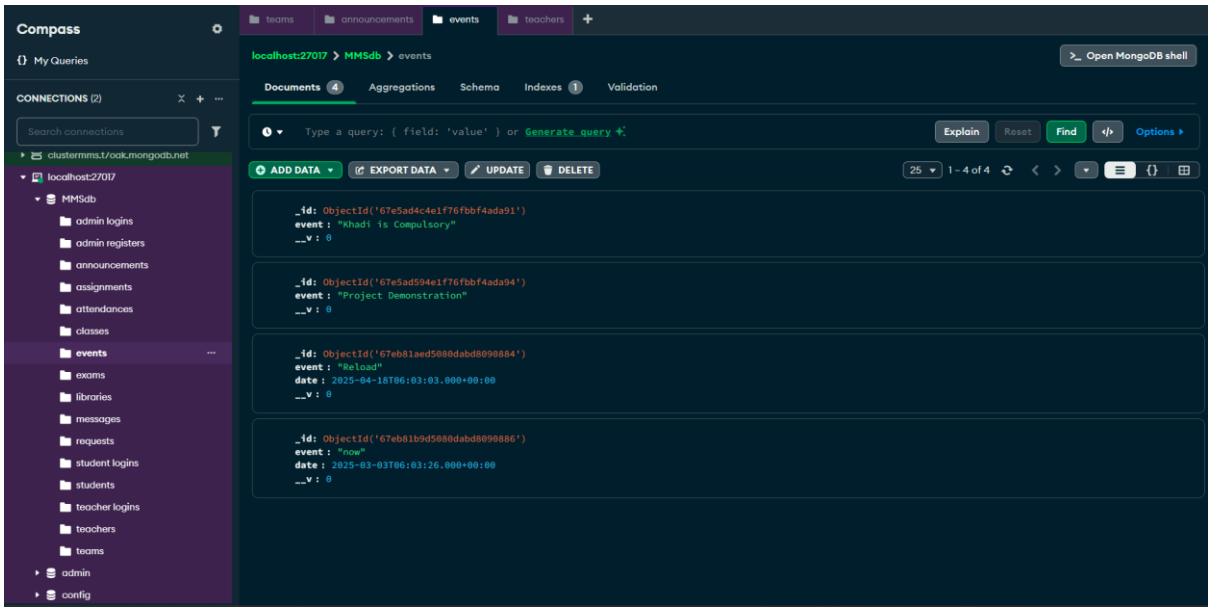
ADMIN DETAILS



ANNOUNCEMENTS

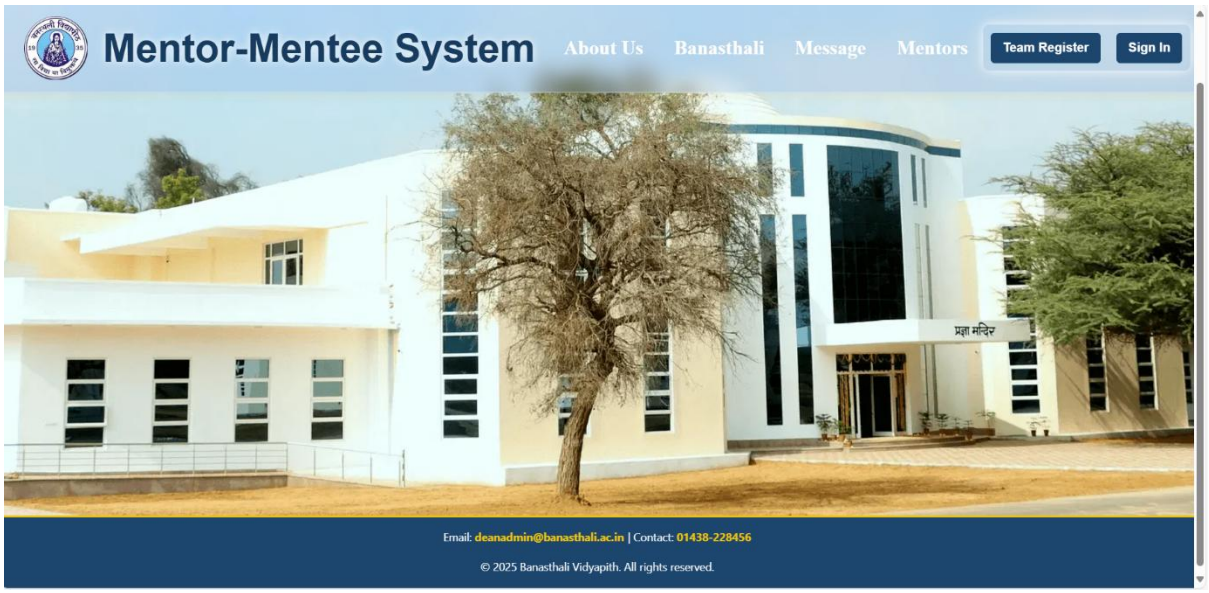


EVENTS

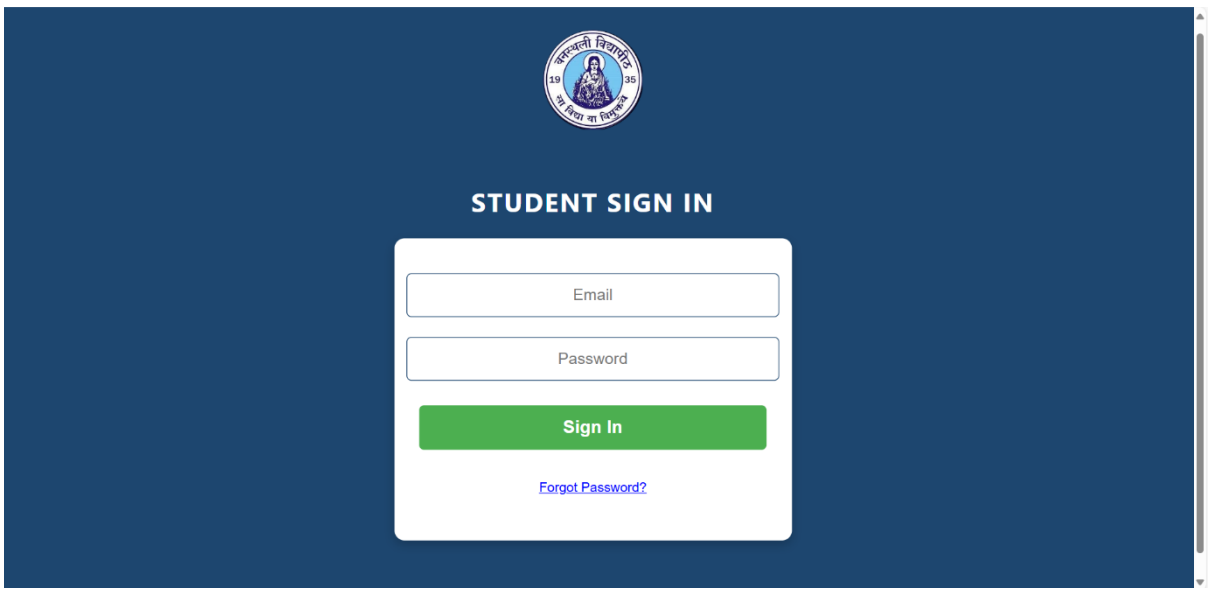


USER INTERFACES

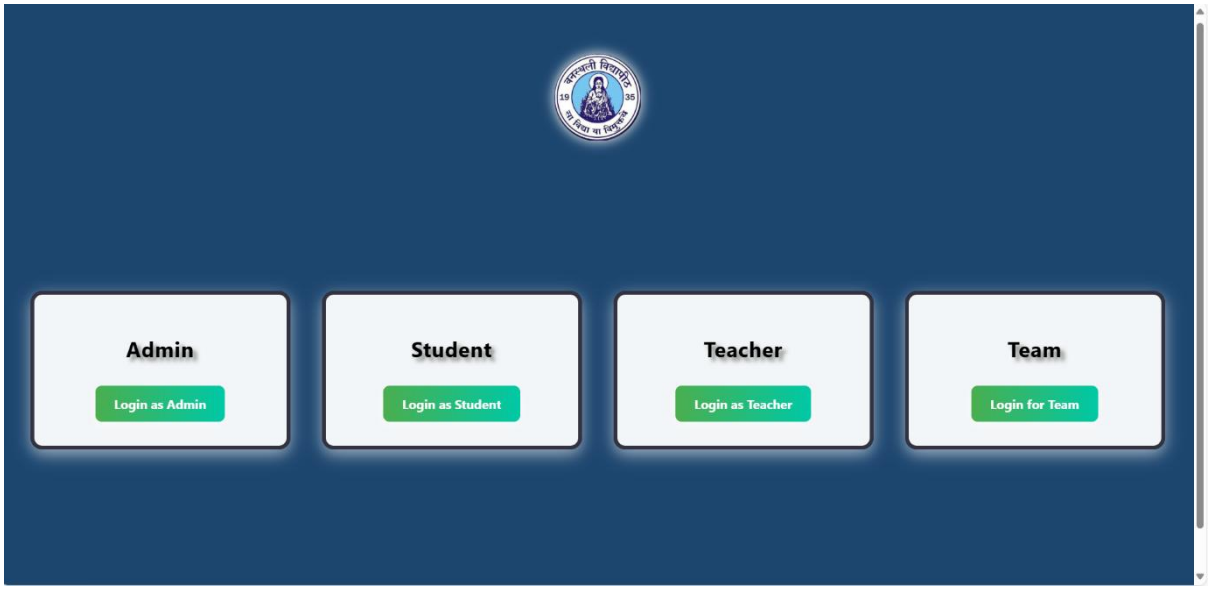
HOME PAGE –



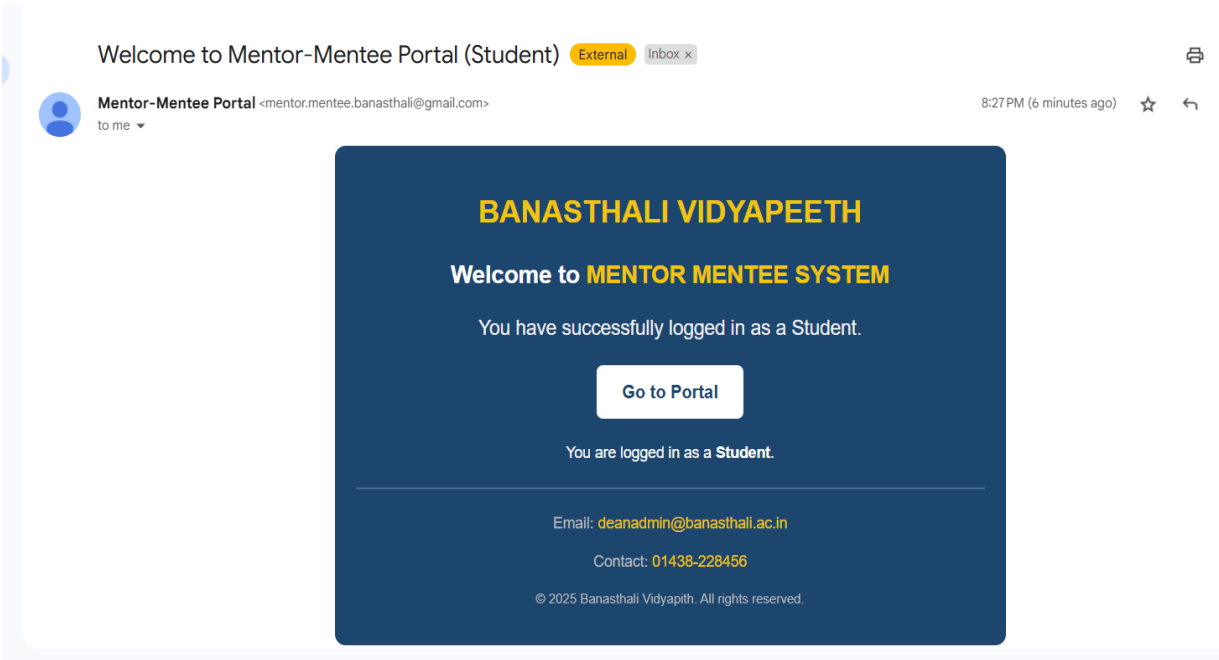
LOGIN PAGE –



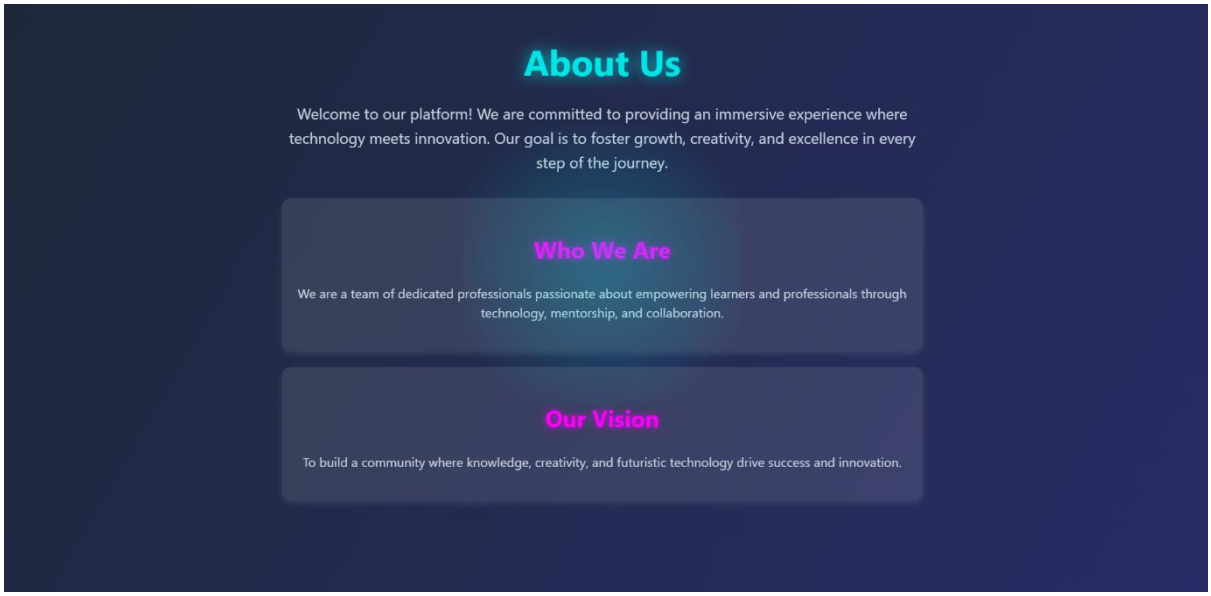
CHOOSING THE USER



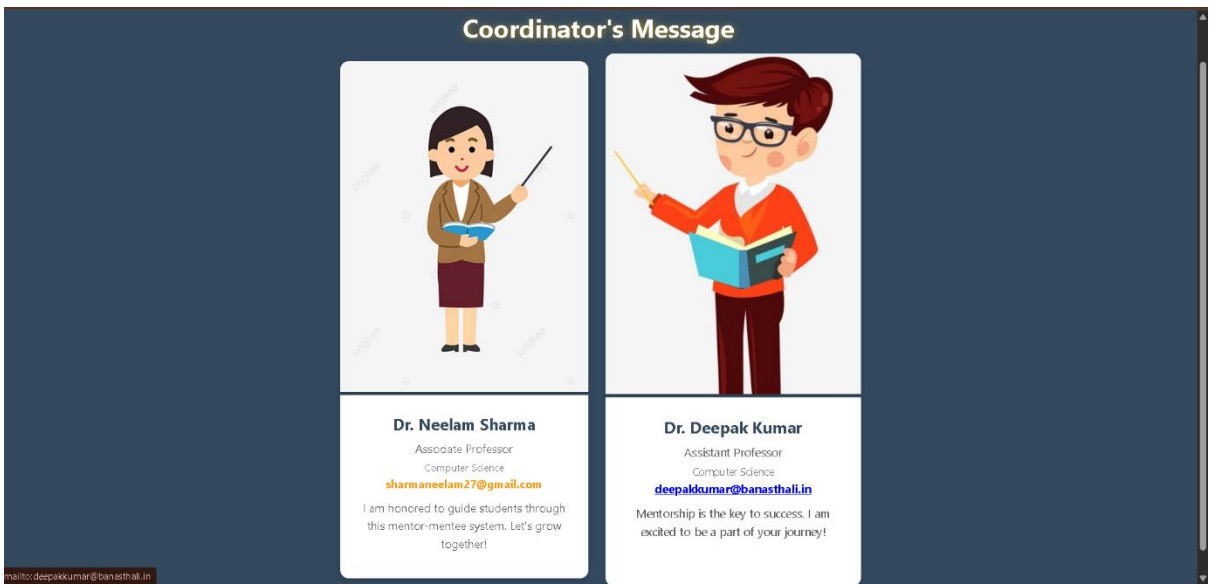
SIGN IN MAIL



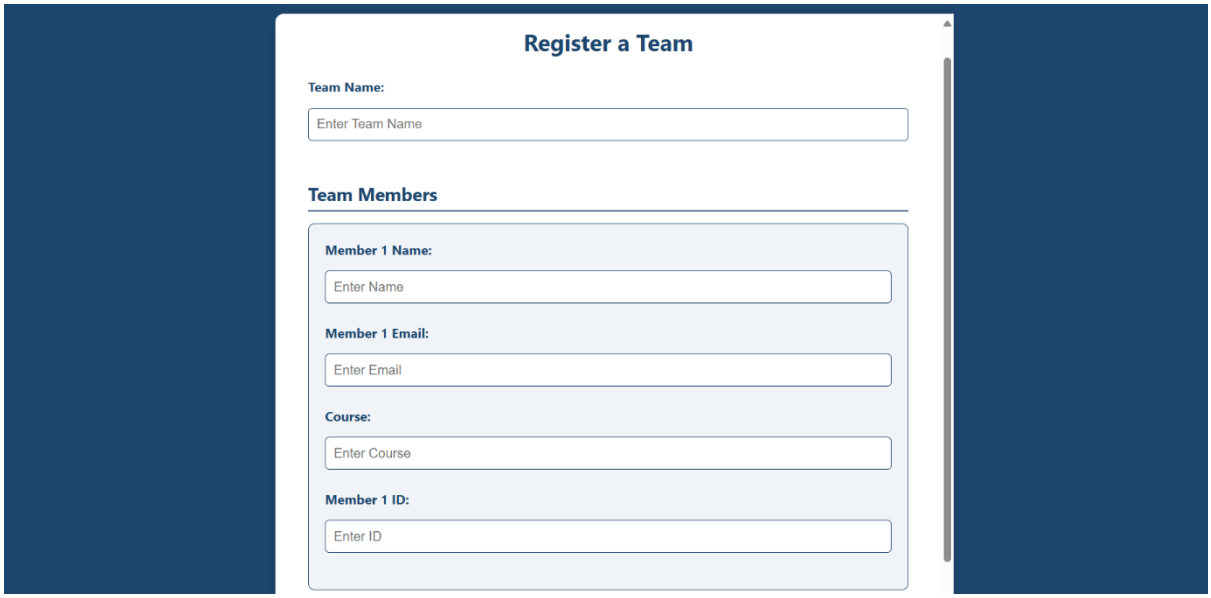
ABOUT PAGE



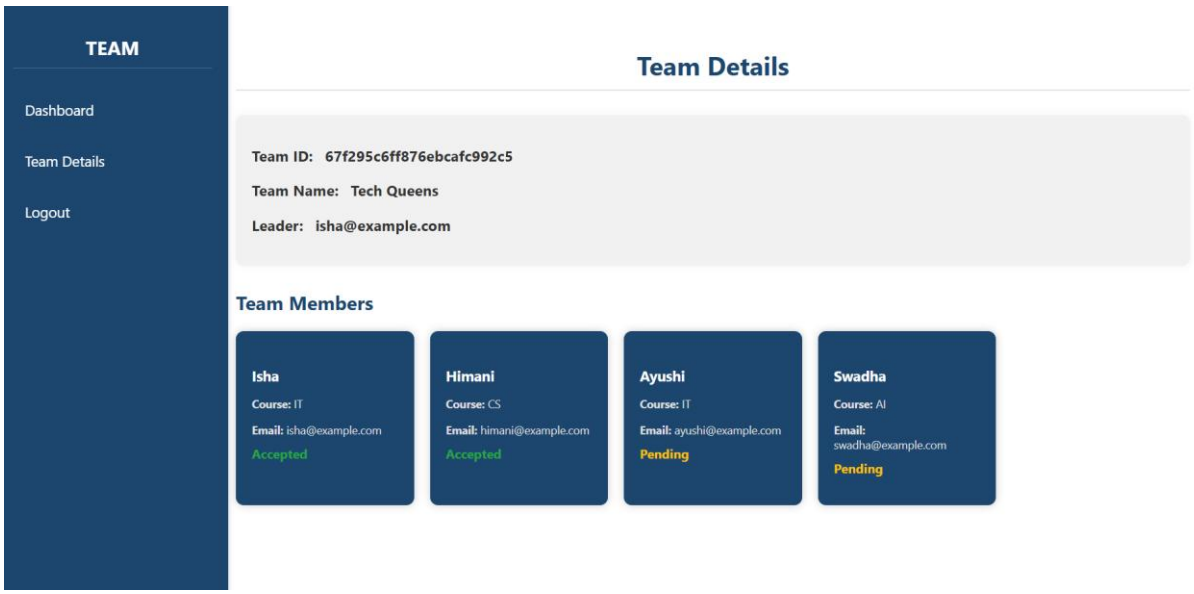
ABOUT COORDINATORS



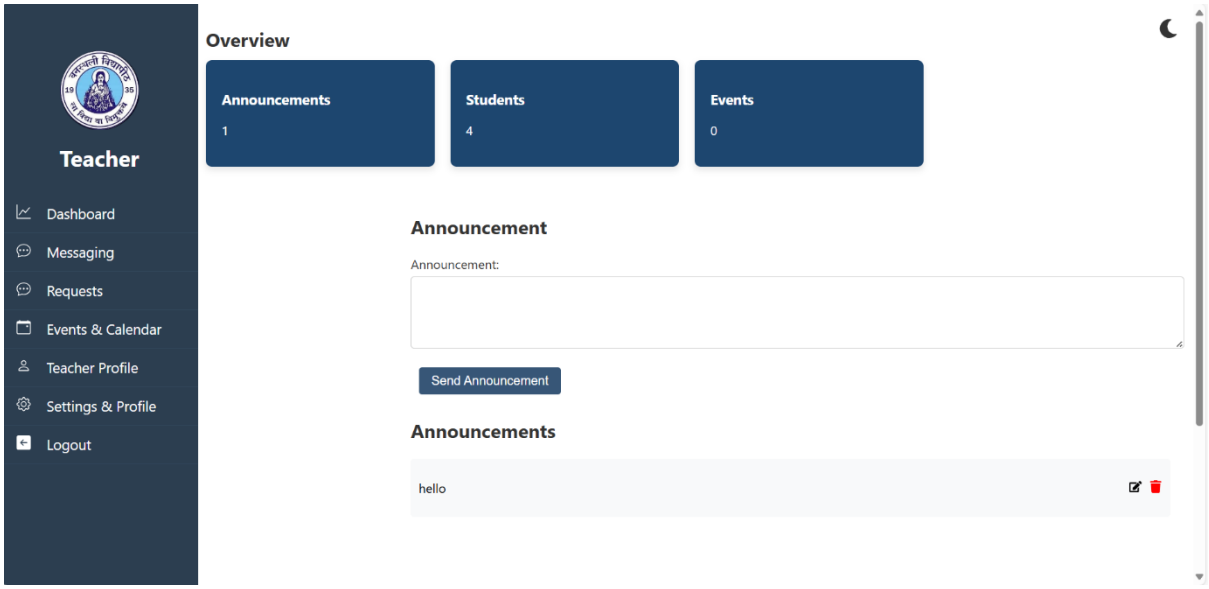
TEAM REGISTER



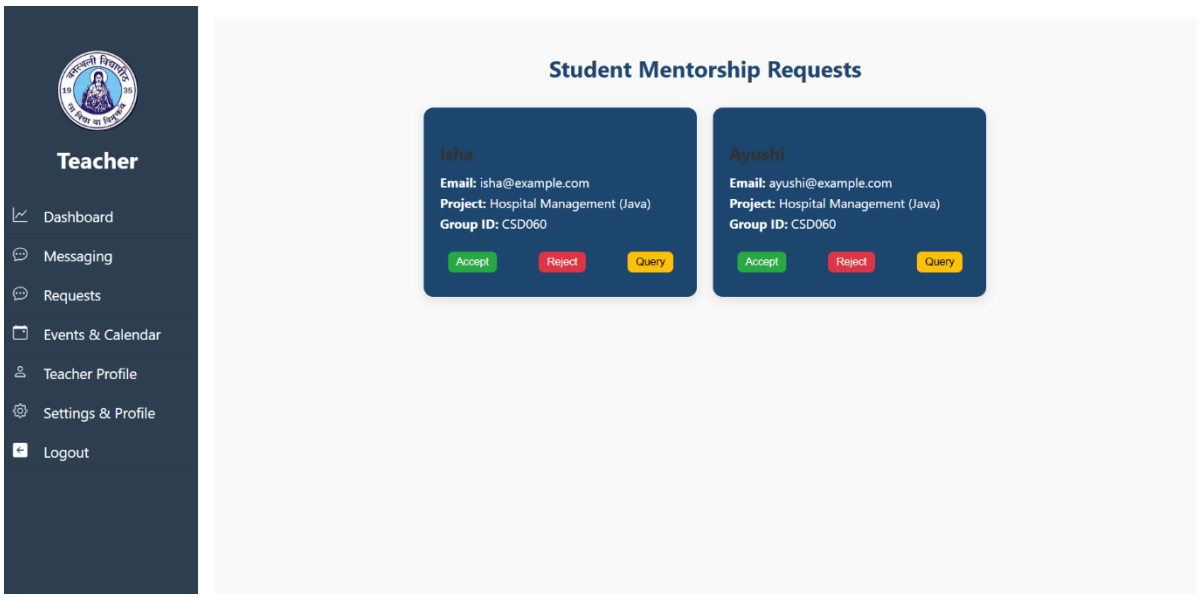
TEAM DASHBOARD



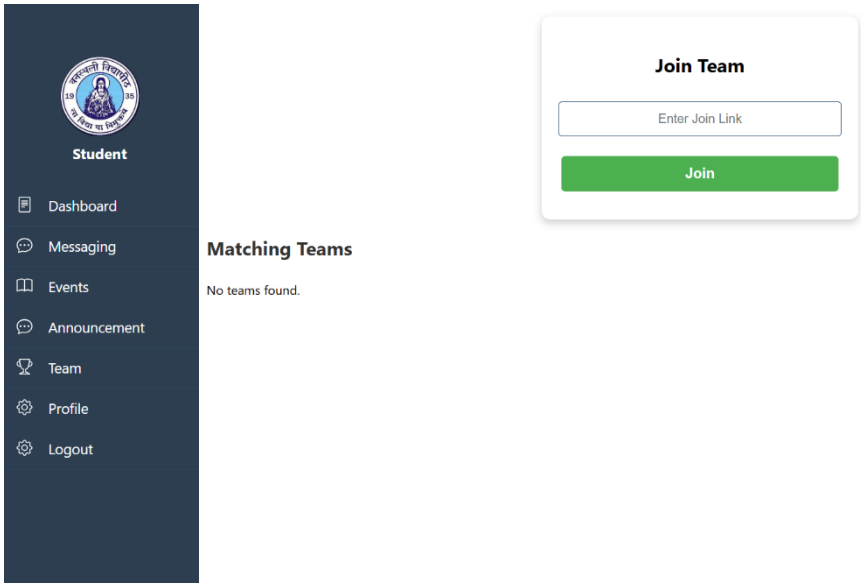
MENTOR DASHBOARD



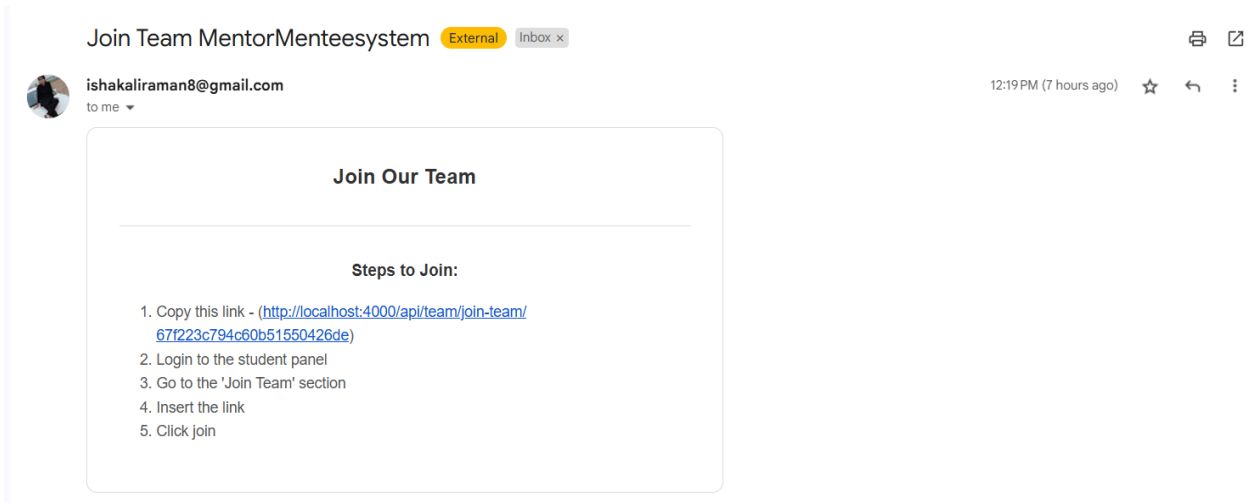
MENTOR REQUEST



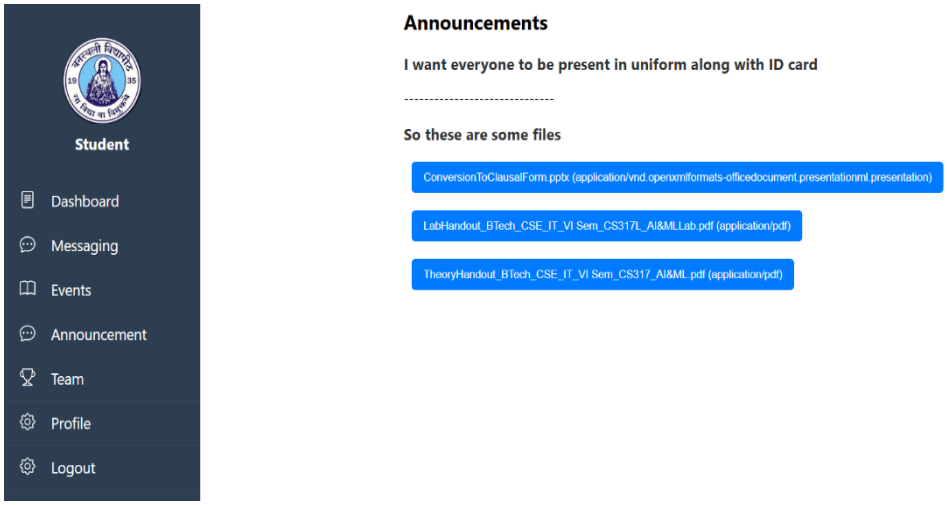
JOIN TEAM DASHBOARD



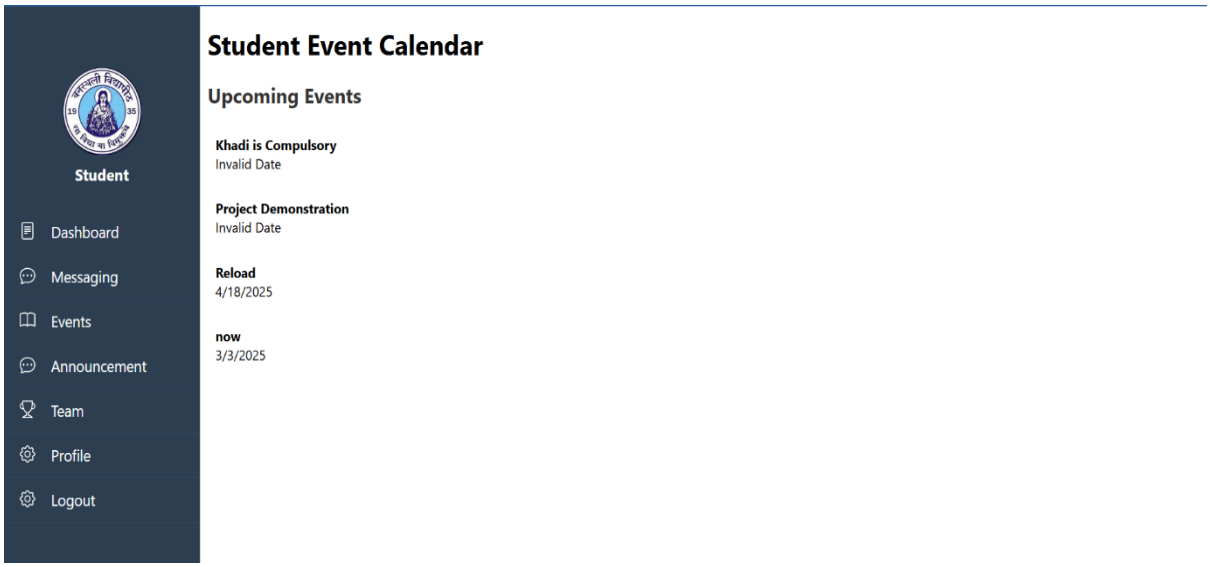
JOIN TEAM MAIL



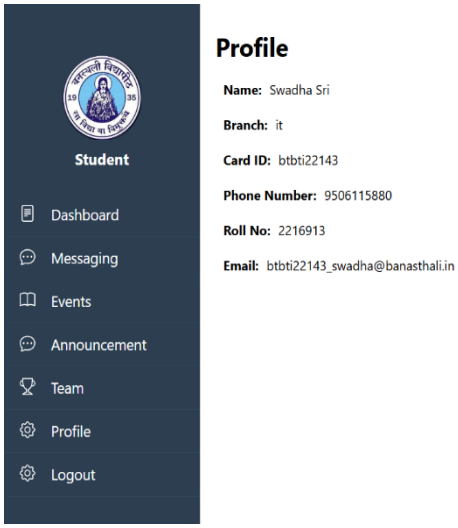
STUDENT ANNOUNCEMENT PAGE



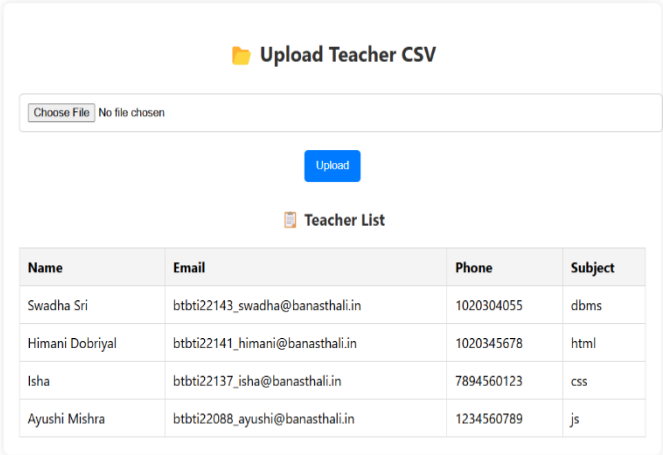
STUDENT EVENT PAGE



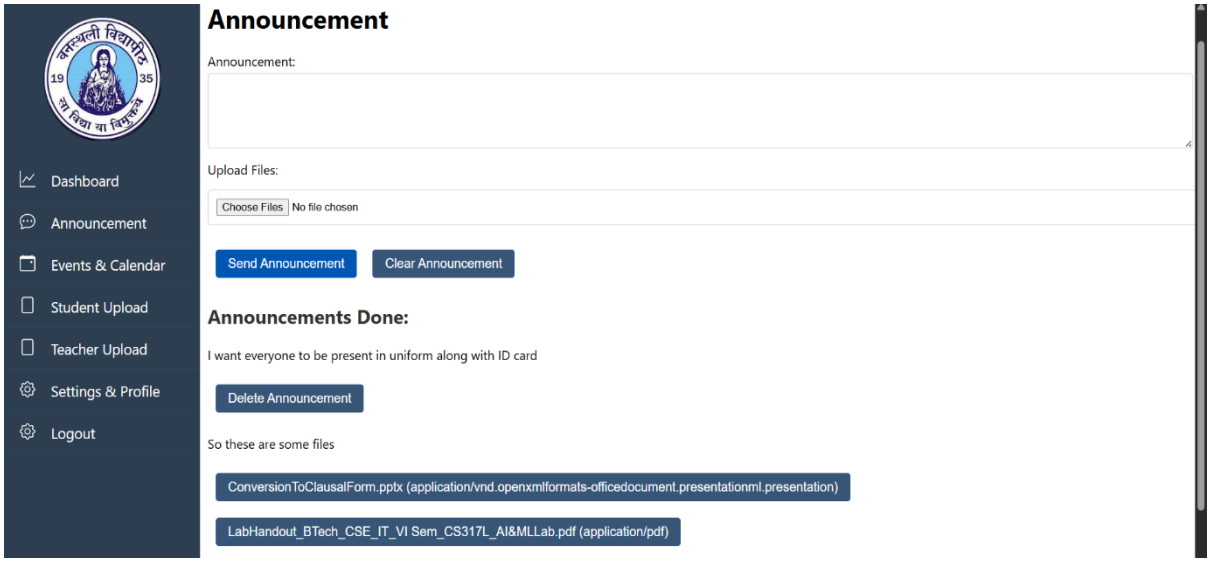
STUDENT PROFILE PAGE



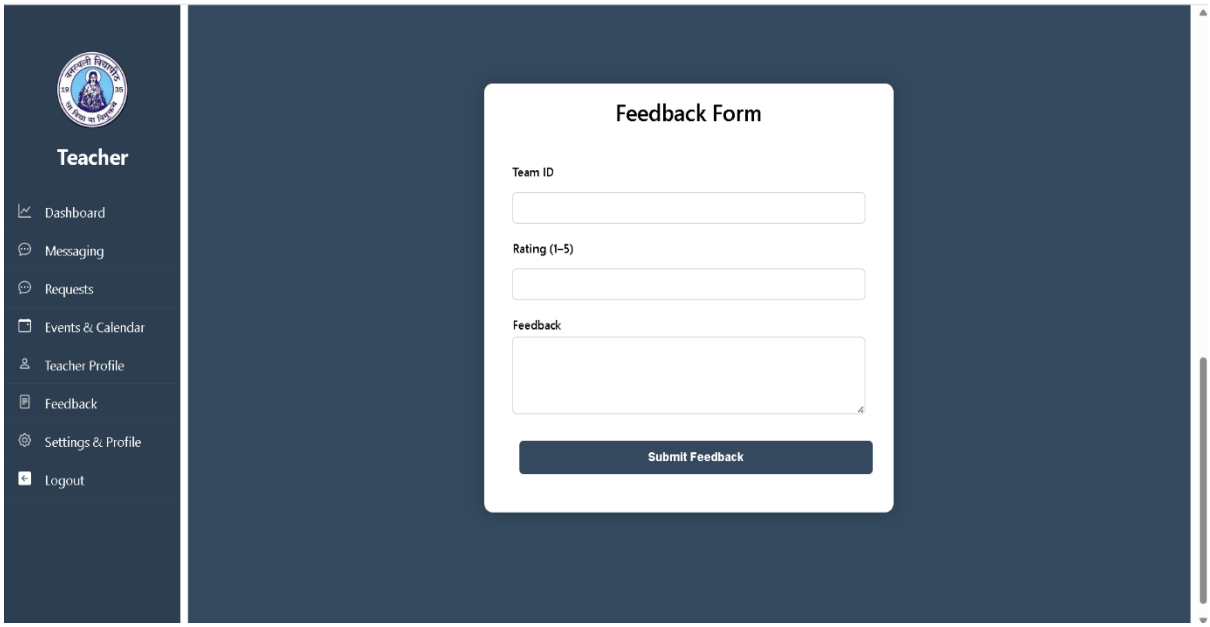
TEACHER FILE UPLOAD



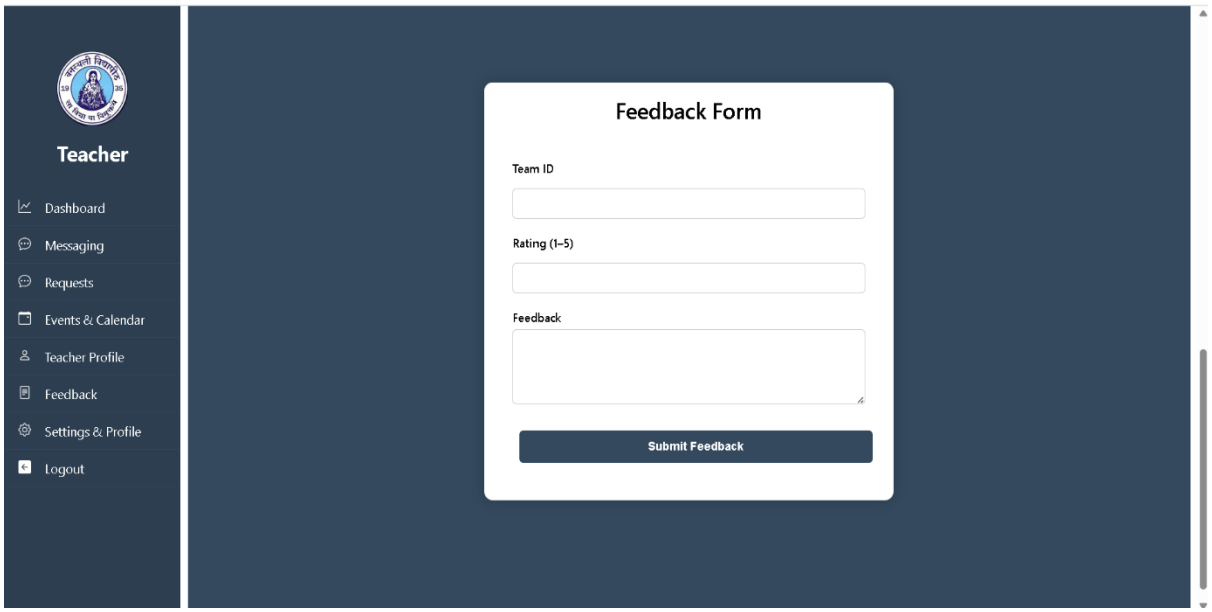
ADMIN ANNOUNCEMENT PAGE



FEEDBACK FORM



FEEDBACK FORM



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/Beginning.ReactJS.Foundations.Chris.Minnick.Wiley.9781119685548.EBooksWorld.i