PLACEMENT MANAGEMENT SYSTEM

SPRINT REVIEW 1

TEAM DETAILS :

1.) Tanushka Agrawal
RA2211003010137

2.) Varsha Singh
RA2211003010157

3.) Abhinav Kumar
RA2211003010163

4.) Mohd. Farhan Alam
RA2211003010165

SCRUM ROLES :

- **PRODUCT OWNER** - Mohd. Farhan Alam
- **SCRUM MASTER** - Tanushka Agrawal
- **DEVELOPERS** - Varsha Singh , Abhinav Kumar

USER STORIES COMPLETED IN SPRINT 1

1. Student Registration

As a student, I want to register on the system so that I can create my profile and apply for placements.

2. Student Login

As a student, I want to log into the system securely so that I can access my account and placement opportunities.

3. Student Home View

As a student, I want to view a dashboard of available jobs and application status so that I can easily navigate and track my placement activities.

DOCUMENTATIONS

Architectural Document

Introduction

This document outlines the high-level architecture of the **Student Placement Management System**. The system is designed to help students register, log in, and view available job opportunities and their application statuses. The architecture follows a **client-server model** with a **Flask backend** and a **static frontend** built using HTML, CSS, and JavaScript.

System Architecture

Frontend: Built using HTML, CSS, and JavaScript and Bootstrap JS for a responsive and interactive user interface.

Backend: Developed using Python Flask to handle API requests and business logic.

Database: PostgreSQL is used to store student profiles, job details, and application statuses.

Authentication: **Session-based authentication** for secure student login.

Components

Frontend

Student Registration Page: HTML form for students to register.

Student Login Page: HTML form for secure login.

Student Dashboard: Displays available jobs and application status using JavaScript to fetch data from the backend.

Backend

Registration API: Handles student registration and stores data in the database.

Login API: Validates student credentials and manages session-based authentication.

Dashboard API: Fetches job details and application status for the student.

Database

Students Table: Stores student details (id, name, email, password, etc.).

Jobs Table: Stores job postings (id, title, description, company, etc.).

Applications Table: Tracks job applications (id, student_id, job_id, status).

Data Flow

Student registers → Data sent to Flask backend → Stored in the database.

Student logs in → Credentials validated → Session created.

Student accesses dashboard → JavaScript fetches jobs and application status from Flask API → Data displayed on the frontend.

Non-Functional Requirements

Security: Passwords are hashed using bcrypt. Session-based authentication ensures secure access.

Scalability: The system is designed to handle up to 1,000 concurrent users.

Performance: API response time should be under 500ms.

Functional Document

Introduction

This document describes the functional requirements of the **Student Placement Management System**. It outlines the features implemented in **Sprint 1**, including student registration, login, and dashboard view.

User Stories and Features

Student Registration

Description: Allows students to create an account by providing their details.

Input: Name, email, password.

Output: Success message or error (e.g., "Email already exists”).

Student Login

Description: Allows students to log in securely using their credentials.

Input: Email and password.

Output: Access to the dashboard or error message (e.g., "Invalid credentials”).

Student Home View

Description: Displays a dashboard with available jobs and application status.

Input: None (fetches data from the backend using JavaScript).

Output: List of jobs and application status (e.g., "Applied", “Pending").

Workflow Diagrams

Registration Workflow:
Student → Fills registration form → Submits → Flask backend validates → Data stored in DB → Success message.

Login Workflow:
Student → Enters credentials → Flask backend validates → Session created → Access granted to dashboard.

Dashboard Workflow:
Student → Accesses dashboard → JavaScript fetches jobs and status from Flask API → Data displayed.

Dependencies

Registration and login features depend on the Authentication API.

Dashboard depends on the Jobs API and Applications API.

Test Case Report

Introduction

This document outlines the test cases for the features implemented in **Sprint 1**. The testing approach includes **unit testing** for backend APIs and **manual testing** for frontend components.

Test Cases

Student Registration

Test Case 1: Valid registration.

Input: Name = "John Doe", Email = "john@example.com", Password = “Password123".

Expected Output: Success message and profile created in DB.

Test Case 2: Duplicate email.

Input: Email = "john@example.com" (already exists).

Expected Output: Error message "Email already exists”.

Student Login

Test Case 1: Valid login.

Input: Email = "john@example.com", Password = “Password123”.

Expected Output: Access to dashboard.

Test Case 2: Invalid login.

Input: Email = "john@example.com", Password = “WrongPassword”.

Expected Output: Error message "Invalid credentials”.

Student Home View

Test Case 1: Dashboard loads.

Input: None.

Expected Output: List of jobs and application status displayed.

Test Results

All test cases for Student Registration and Student Login passed.

Student Home View test case passed, but loading time was slightly higher than expected.

Conclusion

The features implemented in Sprint 1 are functioning as expected.

Retrospective Document

Sprint Overview

Sprint Goal: Complete student registration, login, and dashboard features.

Team Members:

Mohd. Farhan Alam (**Product Owner**).

Tanushka Agrawal (**Scrum Master**).

Varsha Singh and Abhinav Kumar (**Developers**).

What Went Well

All user stories were completed on time.

Effective collaboration between frontend and backend developers.

Daily stand-ups helped in tracking progress and resolving blockers quickly.

Challenges Faced

Integration between frontend and backend took longer than expected.

Initial delays in setting up the database schema.

Improvements for Next Sprint

Break down tasks into smaller sub-tasks for better tracking.

Allocate more time for integration testing.

Improve documentation for API endpoints to avoid confusion.

Action Items

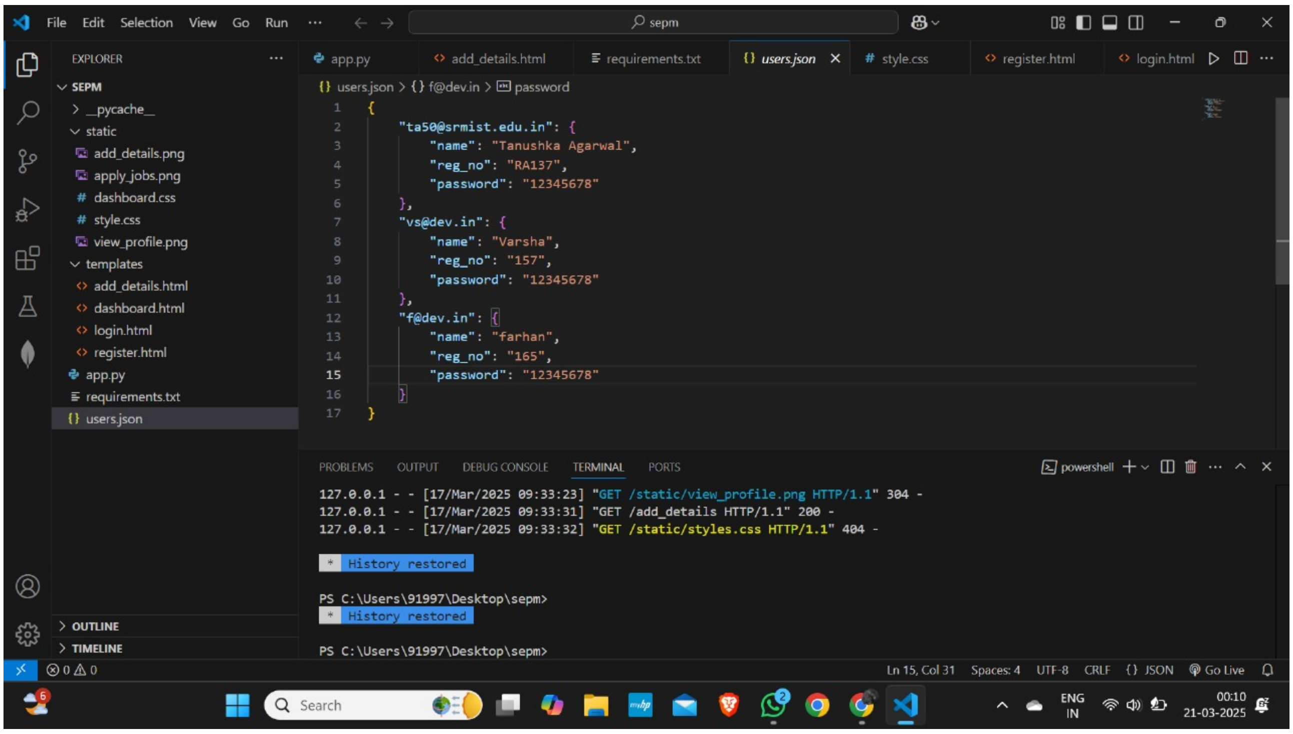
Assign Abhinav to optimize API response time for the dashboard.

Varsha to create detailed API documentation.

Tanushka to ensure better time management in the next sprint.

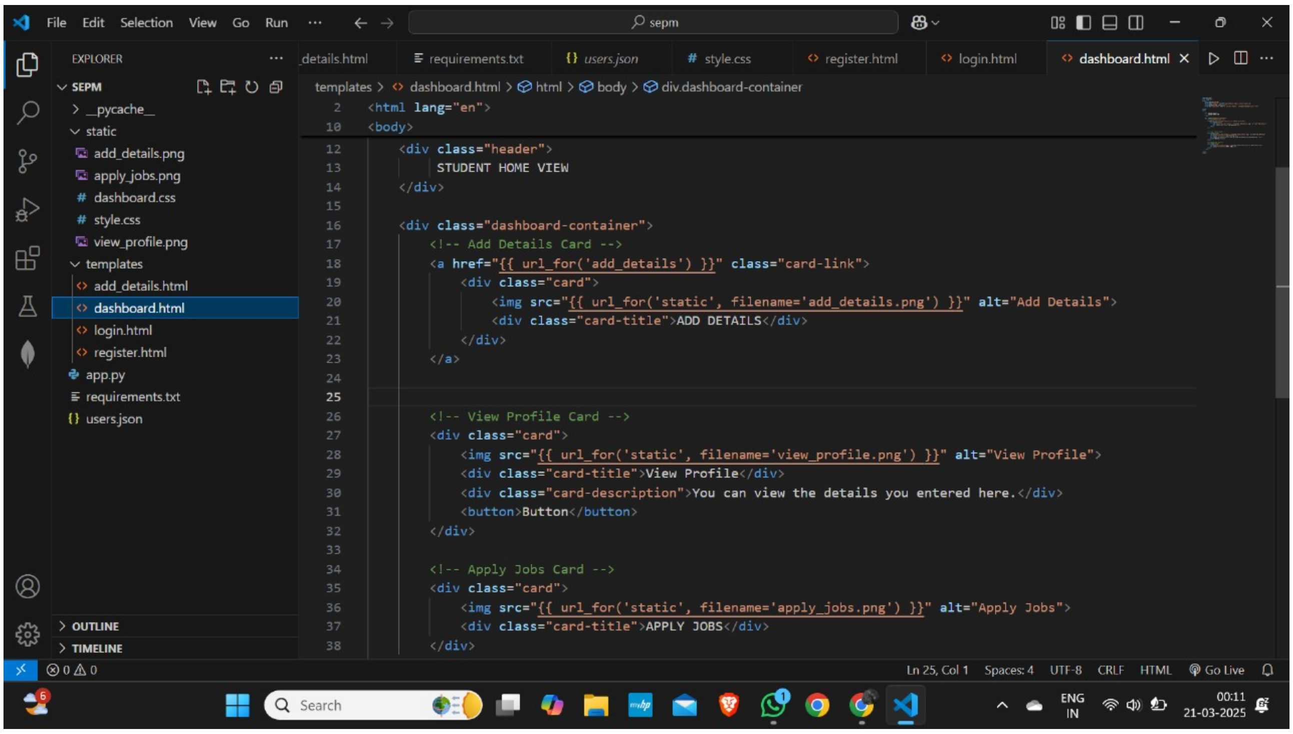
IMPLEMENTATION

CODE SNIPPETS

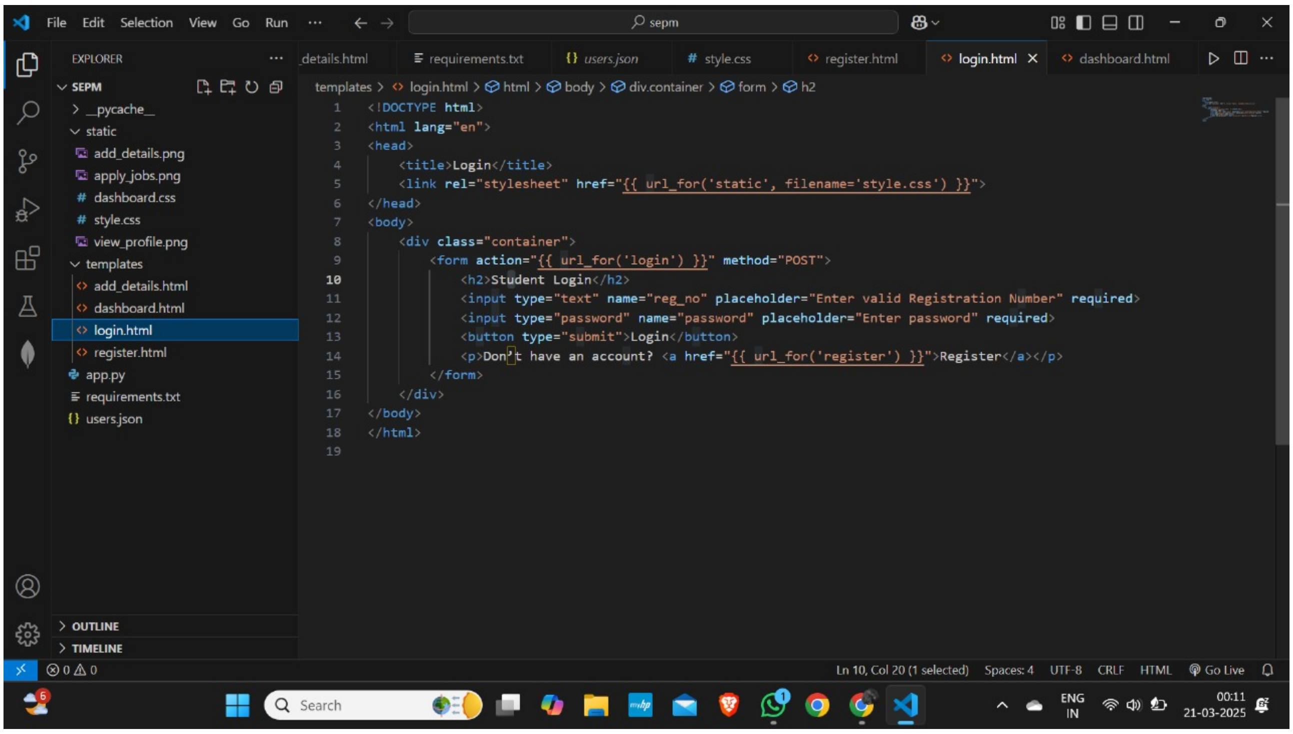


```json
{
    "ta50@srmist.edu.in": {
        "name": "Tanushka Agarwal",
        "reg_no": "RA137",
        "password": "12345678"
    },
    "vs@dev.in": {
        "name": "Varsha",
        "reg_no": "157",
        "password": "12345678"
    },
    "f@dev.in": {
        "name": "farhan",
        "reg_no": "165",
        "password": "12345678"
    }
}
```

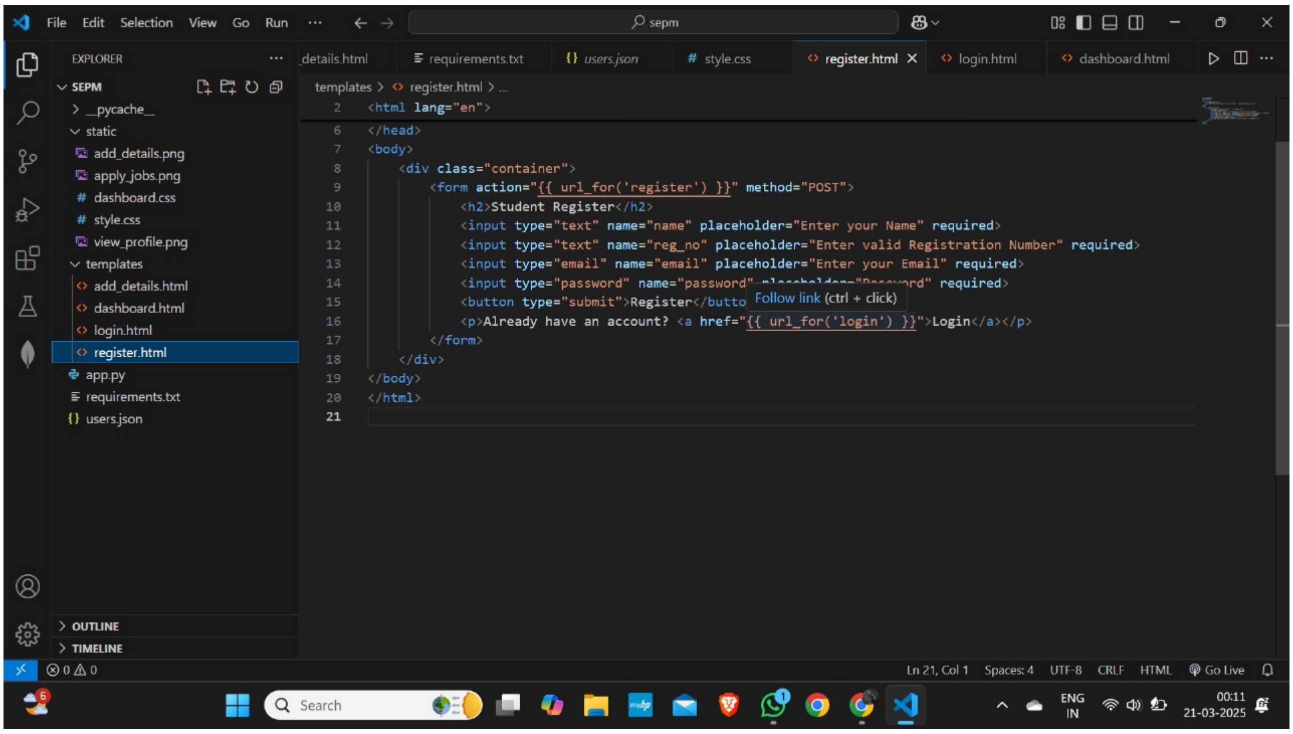
```
127.0.0.1 - - [17/Mar/2025 09:33:23] "GET /static/view_profile.png HTTP/1.1" 304 -
127.0.0.1 - - [17/Mar/2025 09:33:31] "GET /add_details HTTP/1.1" 200 -
127.0.0.1 - - [17/Mar/2025 09:33:32] "GET /static/styles.css HTTP/1.1" 404 -

 * History restored

PS C:\Users\91997\Desktop\sepm>
 * History restored

PS C:\Users\91997\Desktop\sepm>
```



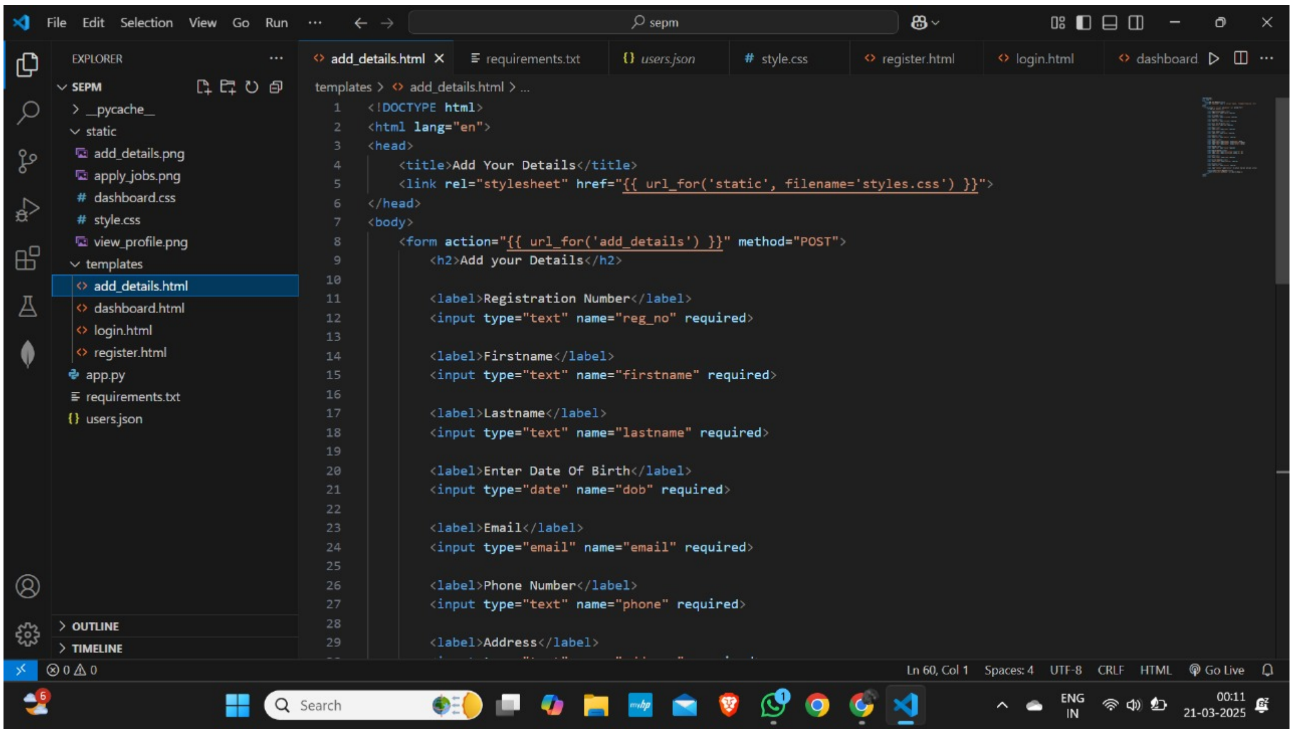
```html
<html lang="en">
<body>
    <div class="header">
        STUDENT HOME VIEW
    </div>

    <div class="dashboard-container">
        <!-- Add Details Card -->
        <a href="{{ url_for('add_details') }}" class="card-link">
            <div class="card">
                <img src="{{ url_for('static', filename='add_details.png') }}" alt="Add Details">
                <div class="card-title">ADD DETAILS</div>
            </div>
        </a>


        <!-- View Profile Card -->
        <div class="card">
            <img src="{{ url_for('static', filename='view_profile.png') }}" alt="View Profile">
            <div class="card-title">View Profile</div>
            <div class="card-description">You can view the details you entered here.</div>
            <button>Button</button>
        </div>

        <!-- Apply Jobs Card -->
        <div class="card">
            <img src="{{ url_for('static', filename='apply_jobs.png') }}" alt="Apply Jobs">
            <div class="card-title">APPLY JOBS</div>
        </div>
```



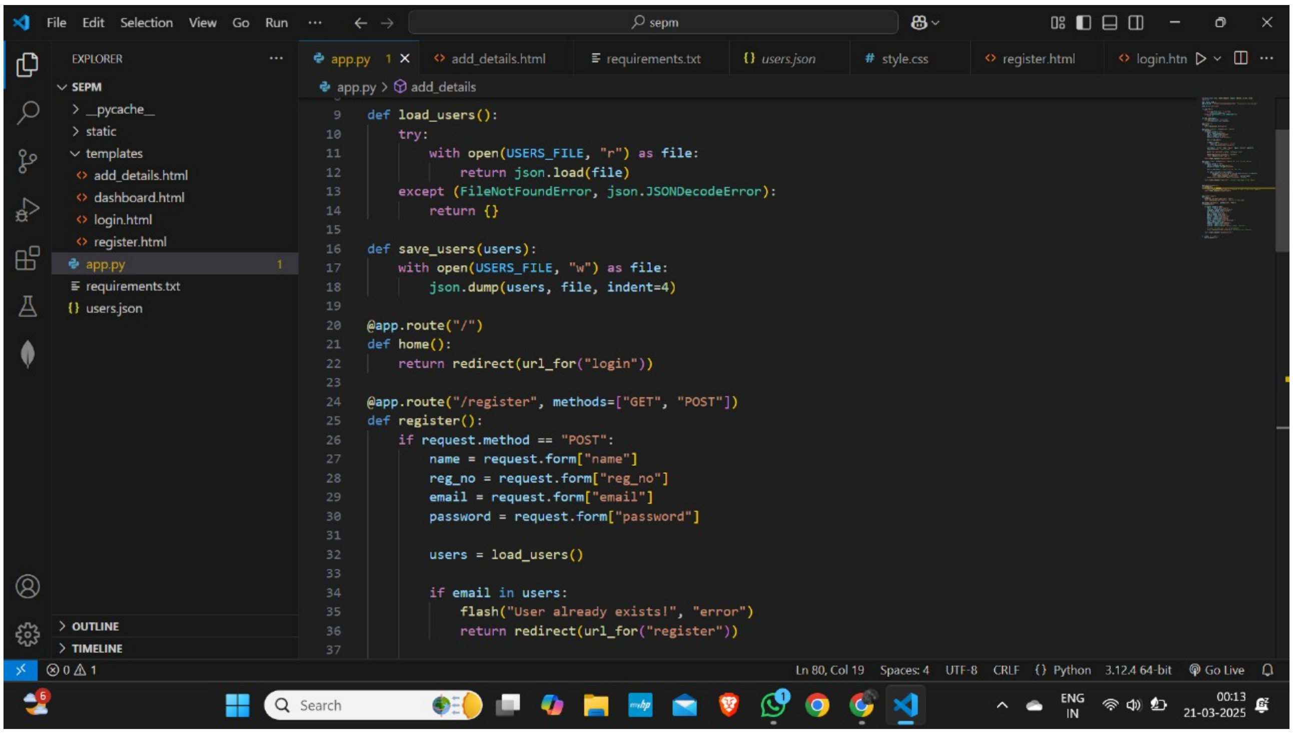
```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Login</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <div class="container">
        <form action="{{ url_for('login') }}" method="POST">
            <h2>Student Login</h2>
            <input type="text" name="reg_no" placeholder="Enter valid Registration Number" required>
            <input type="password" name="password" placeholder="Enter password" required>
            <button type="submit">Login</button>
            <p>Don't have an account? <a href="{{ url_for('register') }}">Register</a></p>
        </form>
    </div>
</body>
</html>
```



```
<html lang="en">
</head>
<body>
    <div class="container">
        <form action="{{ url_for('register') }}" method="POST">
            <h2>Student Register</h2>
            <input type="text" name="name" placeholder="Enter your Name" required>
            <input type="text" name="reg_no" placeholder="Enter valid Registration Number" required>
            <input type="email" name="email" placeholder="Enter your Email" required>
            <input type="password" name="password" placeholder="Password" required>
            <button type="submit">Register</butto
            <p>Already have an account? <a href="{{ url_for('login') }}">Login</a></p>
        </form>
    </div>
</body>
</html>
```

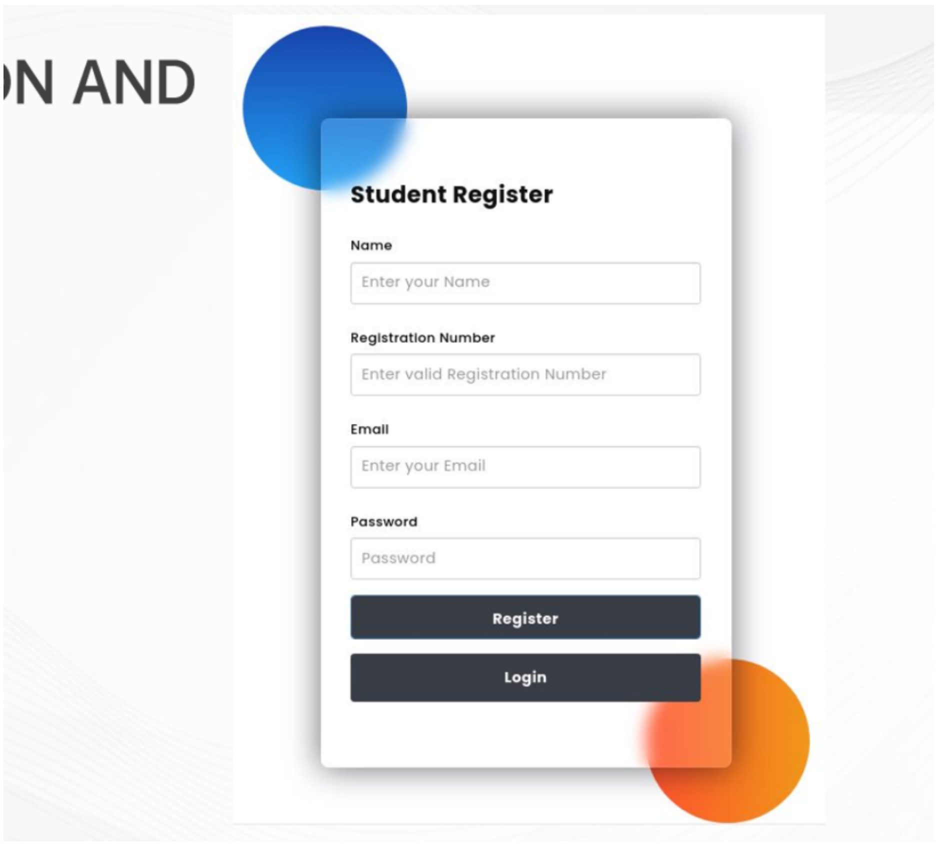


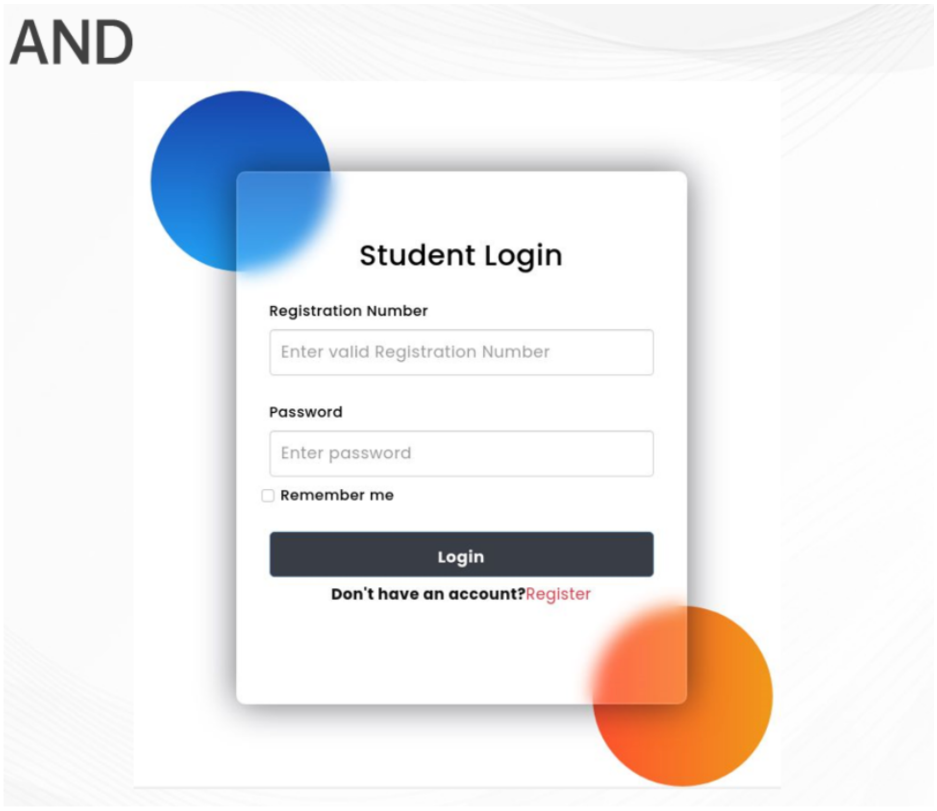
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Add Your Details</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <form action="{{ url_for('add_details') }}" method="POST">
        <h2>Add your Details</h2>

        <label>Registration Number</label>
        <input type="text" name="reg_no" required>

        <label>Firstname</label>
        <input type="text" name="firstname" required>

        <label>Lastname</label>
        <input type="text" name="lastname" required>

        <label>Enter Date Of Birth</label>
        <input type="date" name="dob" required>

        <label>Email</label>
        <input type="email" name="email" required>

        <label>Phone Number</label>
        <input type="text" name="phone" required>

        <label>Address</label>
```

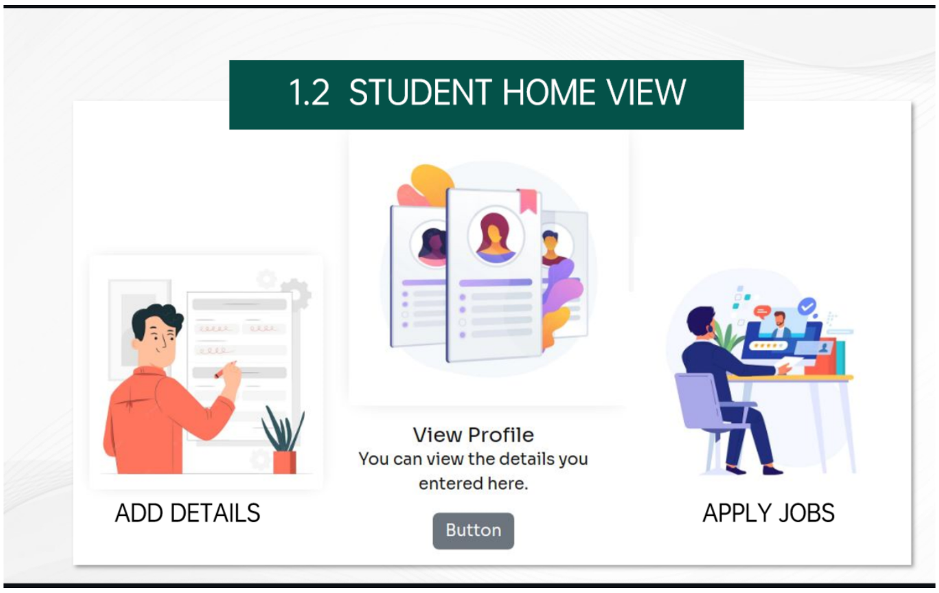


```python
def load_users():
    try:
        with open(USERS_FILE, "r") as file:
            return json.load(file)
    except (FileNotFoundError, json.JSONDecodeError):
        return {}

def save_users(users):
    with open(USERS_FILE, "w") as file:
        json.dump(users, file, indent=4)

@app.route("/")
def home():
    return redirect(url_for("login"))

@app.route("/register", methods=["GET", "POST"])
def register():
    if request.method == "POST":
        name = request.form["name"]
        reg_no = request.form["reg_no"]
        email = request.form["email"]
        password = request.form["password"]

        users = load_users()

        if email in users:
            flash("User already exists!", "error")
            return redirect(url_for("register"))
```

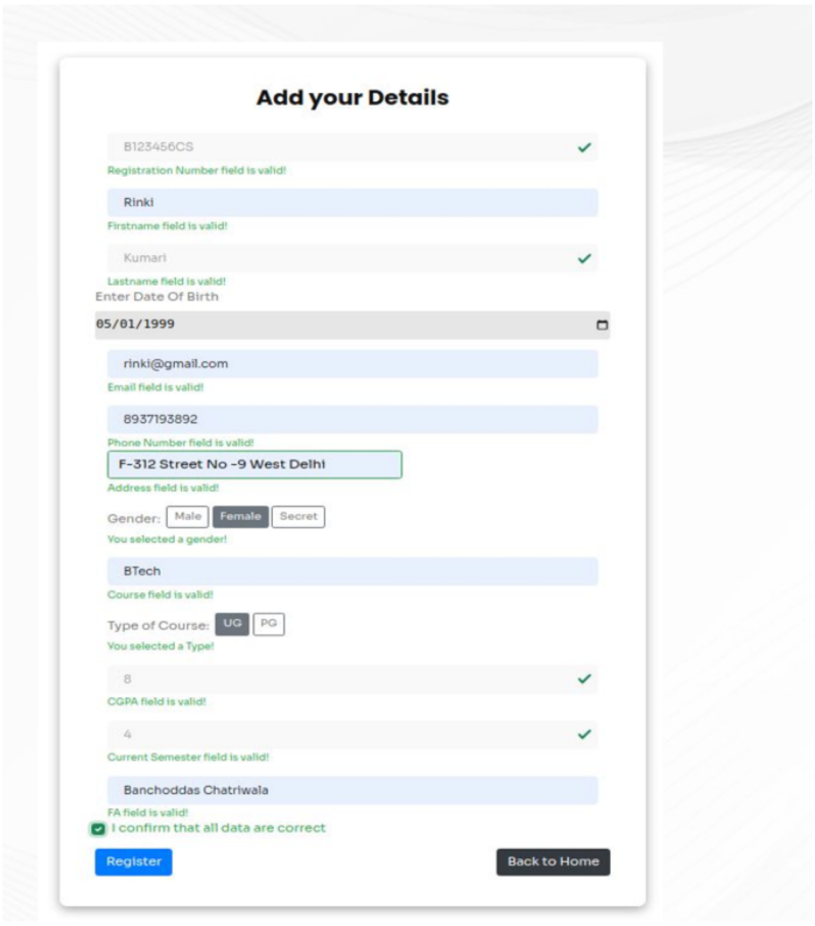
FRONT END IMPLEMENTATION







1.2 STUDENT HOME VIEW



Add your Details

B123456CS ✓
Registration Number field is valid!

Rinki
Firstname field is valid!

Kumari ✓
Lastname field is valid!

Enter Date Of Birth
05/01/1999

rinki@gmail.com
Email field is valid!

8937193892
Phone Number field is valid!

F-312 Street No -9 West Delhi
Address field is valid!

Gender: Male [✓ Female] Secret
You selected a gender!

BTech
Course field is valid!

Type of Course: [✓ UG] PG
You selected a Type!

8 ✓
CGPA field is valid!

4 ✓
Current Semester field is valid!

Banchoddas Chatriwala
FA field is valid!

[✓] I confirm that all data are correct