CERTIFICATE

Project Title: "**LearnHub: Your Center For Skill Enhancement" is** a bona fide work carried out by the following students:

- **TEAM ID:** LTVIP2025TMID54806

- **Vijayasri Peeka (228B1A0517)**
- **Medam Venkata Seena Reddy (Team Member)**
- **Nukathoti Udhay Babu (Team Member)**
- **Shaik Assen(228B1A0554) (Team Member)**

Date Of Submission: 30-06-2025

TABLE OF CONTENTS

1. INTRODUCTION

Online Learning Platform using MERN

An online learning platform(OLP) is a digital platform that provides a variety of tools and resources to facilitate learning and education over the internet. These platforms have become increasingly popular, especially in recent years, as they offer flexibility and accessibility for learners of all ages and backgrounds. Here are some key features and a description of an online learning platform:

.

- → **User-Friendly Interface:** Online learning platforms typically have an intuitive and user-friendly interface that makes it easy for learners, regardless of their technical proficiency, to navigate and access the content.

- → **Course Management:** Instructors or course creators can upload, organize, and manage course materials. Learners can enroll in courses and track their progress.

- → **Interactivity:** Many platforms include interactive elements like discussion forums, chat rooms, and live webinars, which foster communication and collaboration among learners and instructors.

- → **Certification:** Learners can earn certificates or badges upon completing courses or meeting certain criteria, which can be valuable for employment or further education.

- → **Accessibility:** Content is often accessible on various devices, including computers, tablets, and smartphones, making learning possible from anywhere with an internet connection.

- → **Self-Paced Learning:** Learners can typically access course materials at their own pace. This flexibility allows for learning that fits into individual schedules and preferences.

- → **Payment and Subscription Options**: There may be free courses, but some content may require payment or a subscription. Platforms often offer multiple pricing models.

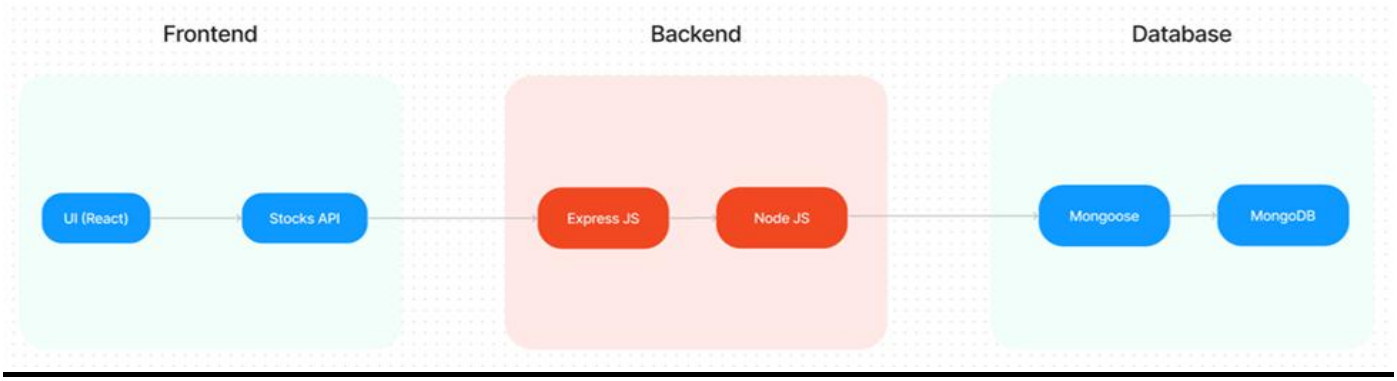
2.PROJECT OVERVIEW

→ Scenario-based Case Study:

- ✓ Scenario: Learning a New Skill
- ✓ **User Registration:** Sarah, a student interested in learning web development, visits the Online Learning Platform and creates an account. She provides her email and chooses a password.
- ✓ **Browsing Courses:** Upon logging in, Sarah is greeted with a user-friendly interface displaying various courses categorized by topic, difficulty level, and popularity.
- ✓ She navigates through the course catalog, filtering courses by name and category until she finds a "Web Development Fundamentals" course that interests her.
- ✓ **Enrolling in a Course:** Sarah clicks on the course and reads the course description, instructor details, and syllabus. Impressed, she decided to enroll in the course.
- ✓ After enrolling, Sarah can access the course materials, including video lectures, reading materials, and assignments.
- ✓ **Learning Progress:** Sarah starts the course and proceeds through the modules at her own pace. The platform remembers her progress, allowing her to pick up where she left off if she needs to take a break.
- ✓ **Interaction and Support:** Throughout the course, Sarah engages with interactive elements such as discussion forums and live webinars where she can ask questions and interact with the instructor and other learners.
- ✓ **Course Completion and Certification:** After completing all the modules and assignments, Sarah takes the final exam. Upon passing, she receives a digital certificate of completion, which she can download and add to her portfolio.
- ✓ **Paid Courses:** Sarah discovers an advanced web development course that requires payment. She purchases the course using the platform's payment system and gains access to premium content.
- ✓ **Teacher's Role:** Meanwhile, John, an experienced web developer, serves as a teacher on the platform. He creates and uploads new courses on advanced web development topics, adds sections to existing courses, and monitors course enrollments.
- ✓ **Admin Oversight:** The admin oversees the entire platform, monitoring user activity, managing course listings, and ensuring smooth operation. They keep track of enrolled students, handle any issues that arise, and maintain the integrity of the platform.

3.ARCHITECTURE

TECHNICAL ARCHITECTURE:



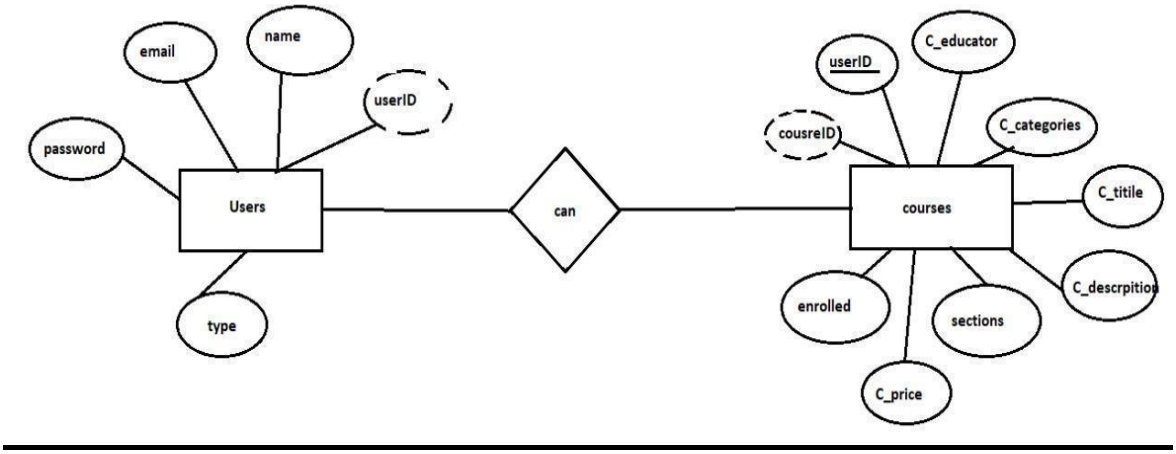
The technical architecture of OLP app follows a client-server model, where the frontend serves as the client and the backend acts as the server. The frontend encompasses not only the user interface and presentation but also incorporates the axios library to connect with backend easily by using RESTful Apis.

The front end utilizes the bootstrap and material UI library to establish a real-time and better UI experience for any user.

On the backend side, we employ Express.js frameworks to handle the server-side logic and communication. For data storage and retrieval, our backend relies on MongoDB. MongoDB allows for efficient and scalable storage of user data and necessary information about the place.

Together, the frontend and backend components, along with Express.js, and MongoDB, form a comprehensive technical architecture for our OLP app. This architecture enables real-time communication, efficient data exchange, and seamless integration, ensuring a smooth and immersive blogging experience for all users.

ER DIAGRAM:



Here there are 2 collections namely users, courses that have their own fields in

Users:

1. _id: (MongoDB creates by unique default)
2. name
3. email
4. password
5. type

Courses:

1. userID: (can act as a foreign key)
2. _id: (MongoDB creates by unique default)
3. C_educator
4. C_categories
5. C_title
6. C_description
7. sections
8. C_price
9. enrolled

4. SET INSTRUCTIONS

PRE-REQUISITES/INSTALLATION:

Here are the key prerequisites for developing a full-stack application using Node.js, Express.js, MongoDB, and React.js:

Vite:

Vite is a new frontend build tool that aims to improve the developer experience for development with the local machine, and for the build of optimized assets for production (go live). Vite (or ViteJS) includes a development server with ES _native_ support and Hot Module Replacement; a build command based on rollup.

```
npm create vite@latest
```

Node.js and npm:

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the server side. It provides a scalable and efficient platform for building network applications.
Install Node.js and npm on your development machine, as they are required to run JavaScript on the server side.
Download: https://nodejs.org/en/download/
Installation instructions: https://nodejs.org/en/download/package-manager/

```
npm init
```

Express.js:

Express.js is a fast and minimalist web application framework for Node.js. It simplifies the process of creating robust APIs and web applications, offering features like routing, middleware support, and modular architecture.
Install Express.js, a web application framework for Node.js, which handles server-side routing, middleware, and API development.
Installation: Open your command prompt or terminal and run the following command:

```
npm install express
```

MongoDB:

MongoDB is a flexible and scalable NoSQL database that stores data in a JSON-like format. It provides high performance, horizontal scalability, and seamless integration with Node.js, making it ideal for handling large amounts of structured and unstructured data.

Set up a MongoDB database to store your application's data.
Download: https://www.mongodb.com/try/download/community
Installation instructions: https://docs.mongodb.com/manual/installation/

React.js:

React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications. Install React.js, a JavaScript library for building user interfaces.
Follow the installation guide: https://reactjs.org/docs/create-a-new-react-app.html

HTML, CSS, and JavaScript: Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

Database Connectivity: Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations. To Connect the Database with Node JS go through the below provided link: https://www.section.io/engineering-education/nodejs- mongoosejs-mongodb/

Install Dependencies:

- Navigate into the cloned repository directory:
  cd containment-zone
- Install the required dependencies by running the following commands:
  cd frontend
  npm install
  cd ../backend
  npm install

Start the Development Server:

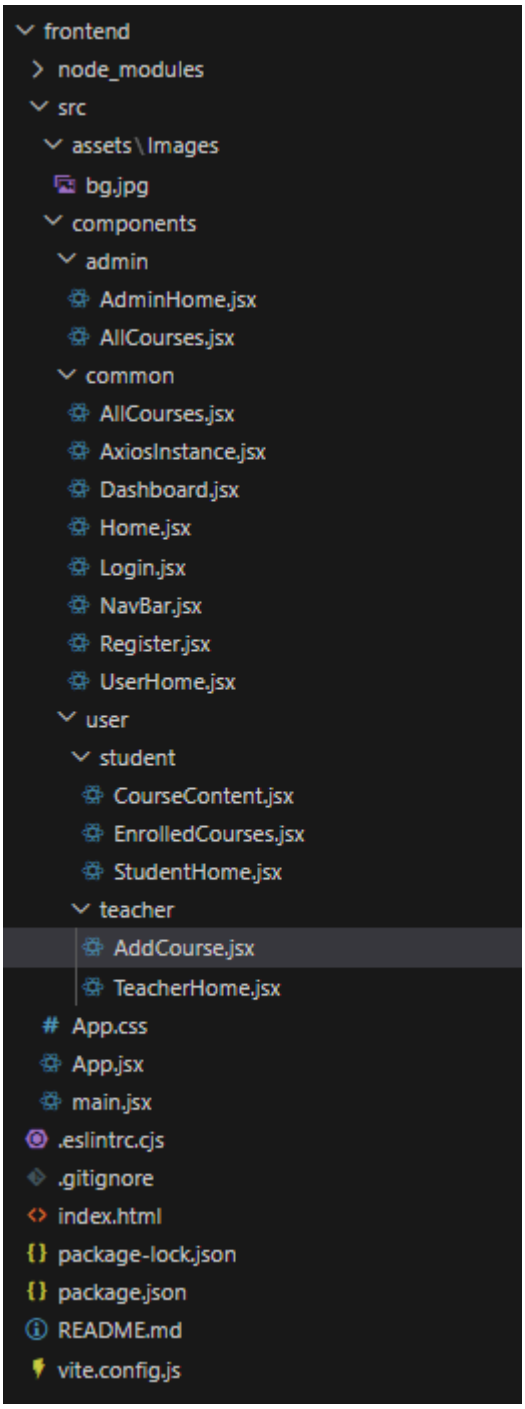
- To start the development server, execute the following command:
  npm start
- The OLP app will be accessible at http://localhost:5172

You have successfully installed and set up the Online learning app on your local machine. You can now proceed with further customization, development, and testing as needed.

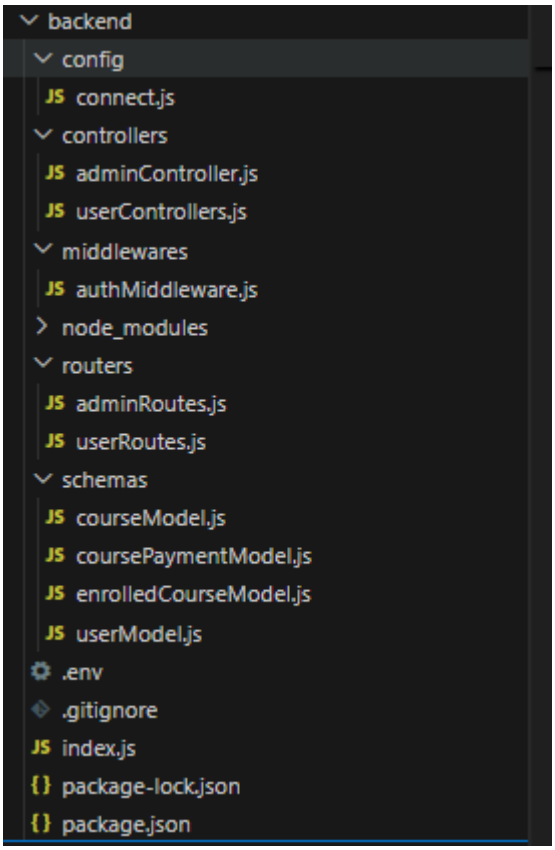
5.PROJECT STRUCTURE

Create a Project folder that contains files as shown below:

1. Frontend:



2. Backend:



- The first image is of frontend part which is showing all the files and folders that have been used in UI development

- The second image is of Backend part which is showing all the files and folders that have been used in backend development

6. API DOCUMENTATION

Application Flow:

The project has a user called– teacher and student and the other will be Admin which takes care of all the users. The roles and responsibilities of these users can be inferred from the API endpoints defined in the code. Here is a summary:

Teacher:

- Can add courses for the student.
- Also, delete the course if no student enrolled in it or for any other reasons.
- Also, add sections to courses.

Student:

- Can enroll in an individual or multiple courses.
- Can start the course where it has stopped.
- Once the course is completed, they can download their certificate of completion of the course.
- For a paid course, they need to purchase it and then they can start the course.
- They can filter out the course by searching by name, category, etc

Admin:

- hey can alter all the courses that are present in the app.
- Watch out for all kinds of users in the app.
- Record all the enrolled students that are enrolled in the course.

Setup & configuration:

→ **Folder setup:**

- ✓ Create frontend and
- ✓ Backend folders

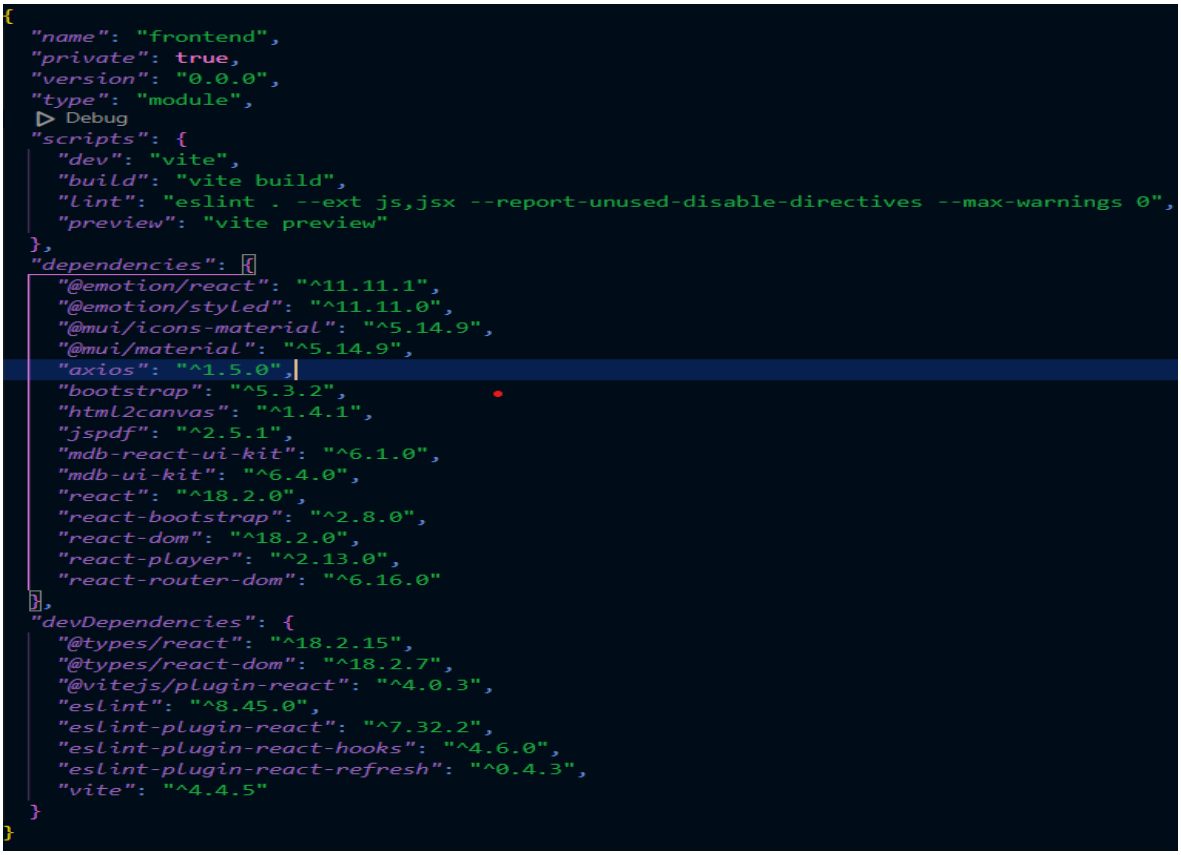
Open the backend folder to install the necessary tools

For backend, we use:

- ✓ cors
- ✓ bcryptjs
- ✓ express

- ✓ dotenv
- ✓ mongoose
- ✓ Multer
- ✓ Nodemon
- ✓ jsonwebtoken

→ After the installation of all the libraries, the package.json files for the frontend looks like the one mentioned below:



```
{
  "name": "frontend",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  ▷ Debug
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "lint": "eslint . --ext js,jsx --report-unused-disable-directives --max-warnings 0",
    "preview": "vite preview"
  },
  "dependencies": {
    "@emotion/react": "^11.11.1",
    "@emotion/styled": "^11.11.0",
    "@mui/icons-material": "^5.14.9",
    "@mui/material": "^5.14.9",
    "axios": "^1.5.0",
    "bootstrap": "^5.3.2",
    "html2canvas": "^1.4.1",
    "jspdf": "^2.5.1",
    "mdb-react-ui-kit": "^6.1.0",
    "mdb-ui-kit": "^6.4.0",
    "react": "^18.2.0",
    "react-bootstrap": "^2.8.0",
    "react-dom": "^18.2.0",
    "react-player": "^2.13.0",
    "react-router-dom": "^6.16.0"
  },
  "devDependencies": {
    "@types/react": "^18.2.15",
    "@types/react-dom": "^18.2.7",
    "@vitejs/plugin-react": "^4.0.3",
    "eslint": "^8.45.0",
    "eslint-plugin-react": "^7.32.2",
    "eslint-plugin-react-hooks": "^4.6.0",
    "eslint-plugin-react-refresh": "^0.4.3",
    "vite": "^4.4.5"
  }
}
```

→ After the installation of all the libraries, the package.json files for the backend looks like the one mentioned below.

Backend Development:

- **Setup express server**
  - ✓ Create index.js file in the server (backend folder).
  - ✓ Define the port number, MongoDB connection string, and JWT key in the env file to access it.
  - ✓ Configure the server by adding cors, and body-parser.

- **Add authentication:** f
  - ✓ You need to make a middleware folder and in that make authMiddleware.js file for the authentication of the projects and can use in.

Database Development:

- **Configure MongoDB**
  - ✓ Import mongoose.
  - ✓ Add database connection from config.js file present in the config folder

- ✓ Create a model folder to store all the DB schemas.

Frontend Development:

→ **Installation of required tools:**

- For frontend, we use:
  - ✓ React
  - ✓ Bootstrap
  - ✓ Material UI
  - ✓ Axios
  - ✓ Antd
  - ✓ mdb-react-ui-kit
  - ✓ react-bootstrap

7. AUTHENTICATION

➢ OVERVIEW

LearnHub uses **JWT (JSON Web Tokens)** for secure, stateless authentication and implements **role-based access control** to ensure users only access features relevant to their roles.

➢ USER ROLES:

Student – Can register, browse courses, enroll, and interact via course d discussion/chat.
Instructor – Can create and manage courses, upload content, and respond to student queries.
Admin – Has full privileges to manage users, courses, and system settings.

➢ REGISTRATION:

Users sign up via the /register endpoint.
Required fields: **name**, **email**, **password**, and **role**.
Passwords are **hashed using bcrypt** before being stored in the database to ensure secure credential storage.

➢ LOGIN:

Users log in using /login endpoint.
If credentials are valid, a **JWT token** is generated and returned.
This token is required for accessing protected routes.

➢ TOKEN-BASED AUTHENTICATION:

JWT tokens are stored client-side (typically in localStorage or sessionStorage).
For secure API access, the token is attached to the **Authorization header** as:

```
Authorization: Bearer <token>
```

➢ MIDDLE WARE PROTECTION:

Backend routes are secured using **authentication middleware**.
Middleware verifies the token and extracts user details.
If invalid or missing, a **401 Unauthorized** response is returned.

- **ROLE-BASED ACCESS CONTROL:**

  - ✓ After authentication, user **roles are verified** to control access.
    - For example:
      - Only **Admins** can access the Admin Dashboard.
      - Only **Instructors** can create/edit courses.
  - ✓ Prevents unauthorized access to privileged endpoints and features.

- **LOGOUT:**

  Logout is handled on the client side by **removing the JWT token** from storage. Since sessions are stateless, no server-side cleanup is required.

- **PASSWORD SECURITY:**

→ LearnHub ensures strong password practices:

  - ✓ Passwords are **hashed using bcrypt**.
  - ✓ **Salting** is used to prevent rainbow table and brute-force attacks.
  - ✓ The original password is **never stored** in the database.

- **SESSION SECURITY AND TOKEN EXPIRY:**

  - ✓ JWT tokens include an **expiry time** (e.g., 1 hour).
  - ✓ Once expired, users must **log in again** to receive a new token.
  - ✓ This reduces risk of long-term token misuse and improves overall system security.

➢ FRONTEND TOKEN HANDLING:

- Tokens are stored in **localStorage** or **sessionStorage** on the client side.
- During logout, the token is cleared from storage to prevent unauthorized access.
- The frontend includes checks to detect expired or missing tokens and redirect users to the login page.

➢ ERROR HANDLING:

→ Learn-Hub provides clear and meaningful error messages in case of:

- Invalid login credentials (wrong email/password)
- Expired or missing token
- Unauthorized access to restricted routes
- Improper user roles

This helps both users and developers quickly identify and resolve issues.

➢ SECURITY BEST PRACTICES:

To enhance the overall security of the authentication system:

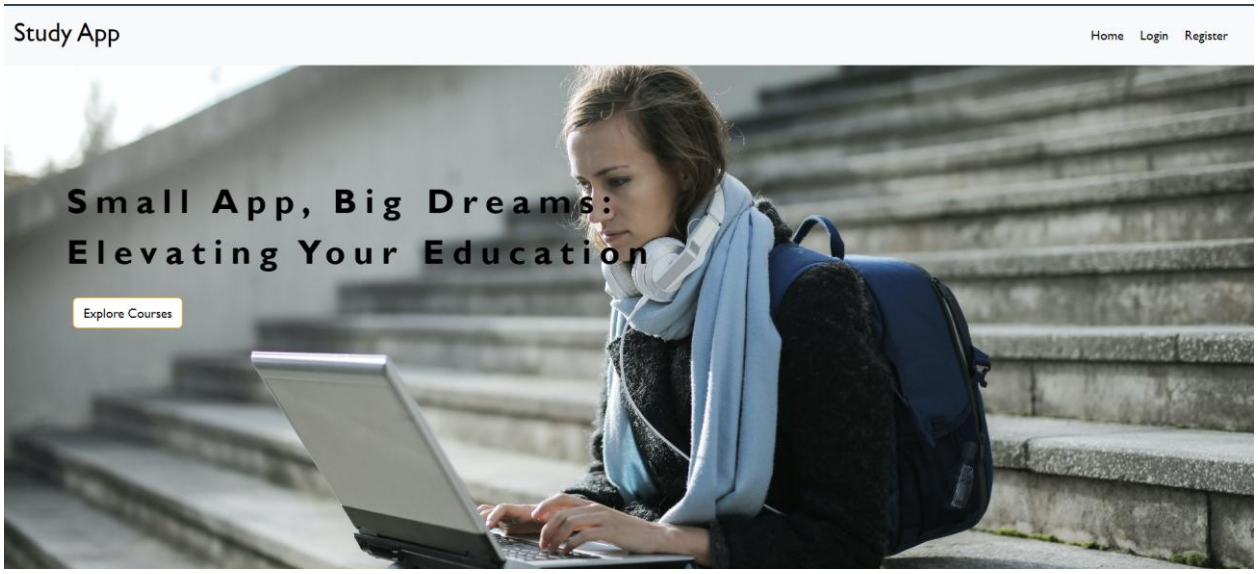
- HTTPS is used to encrypt all client-server communication.
- Input fields are validated on both client and server to prevent injection attacks.
- Rate limiting and brute-force protection mechanisms can be added to login APIs.

8. USER INTERFACE

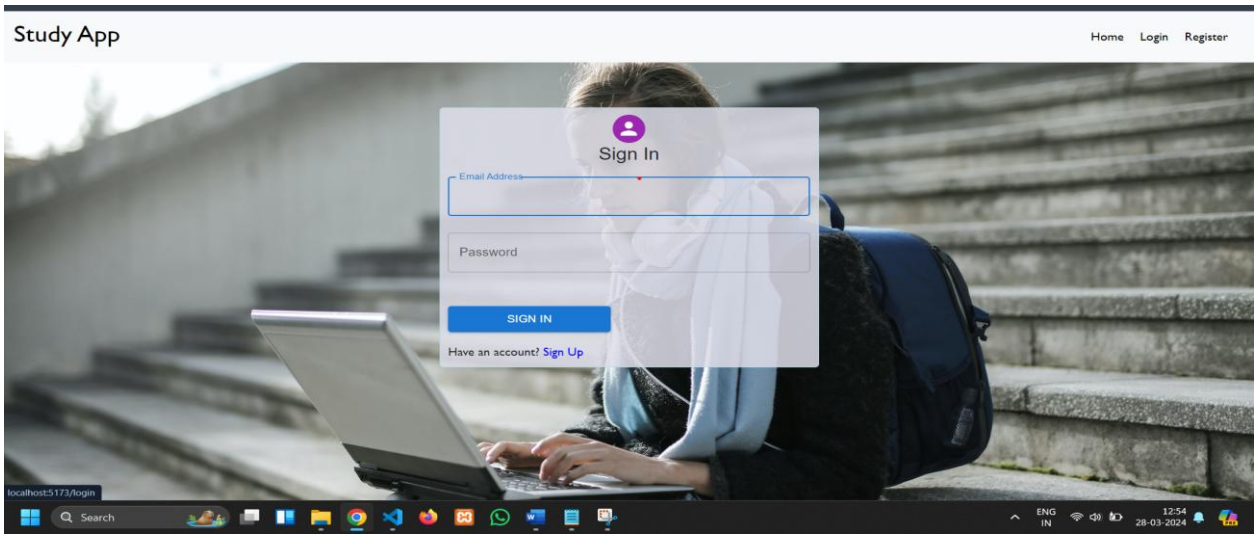
Project Implementation:

On completing the development part, we then ran the application one last time to verify all the functionalities and look for any bugs in it. The user interface of the application looks a bit like the one's provided below.

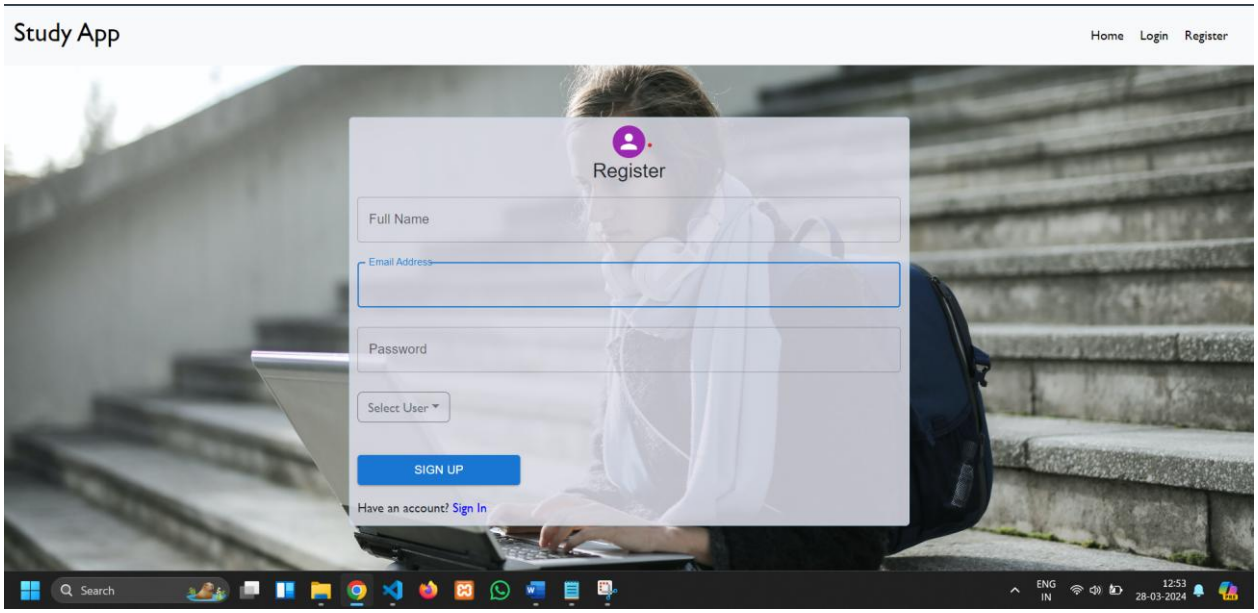
- **LANDING PAGE:**



- **LOGIN PAGE:**

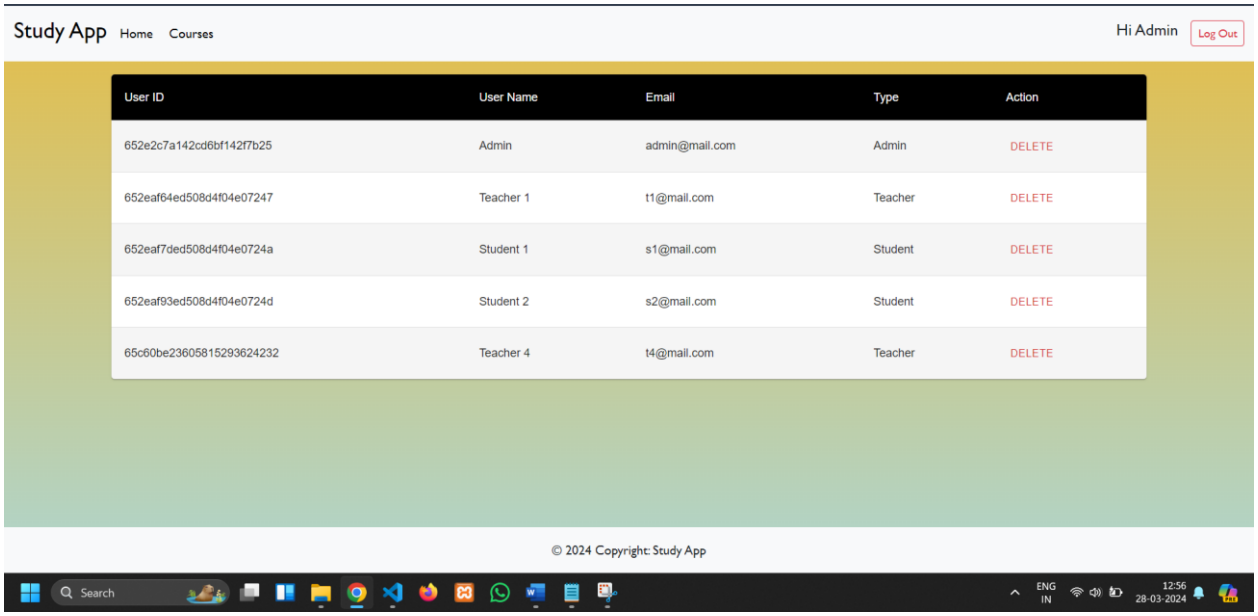


- **REGISTRATION PAGE:**

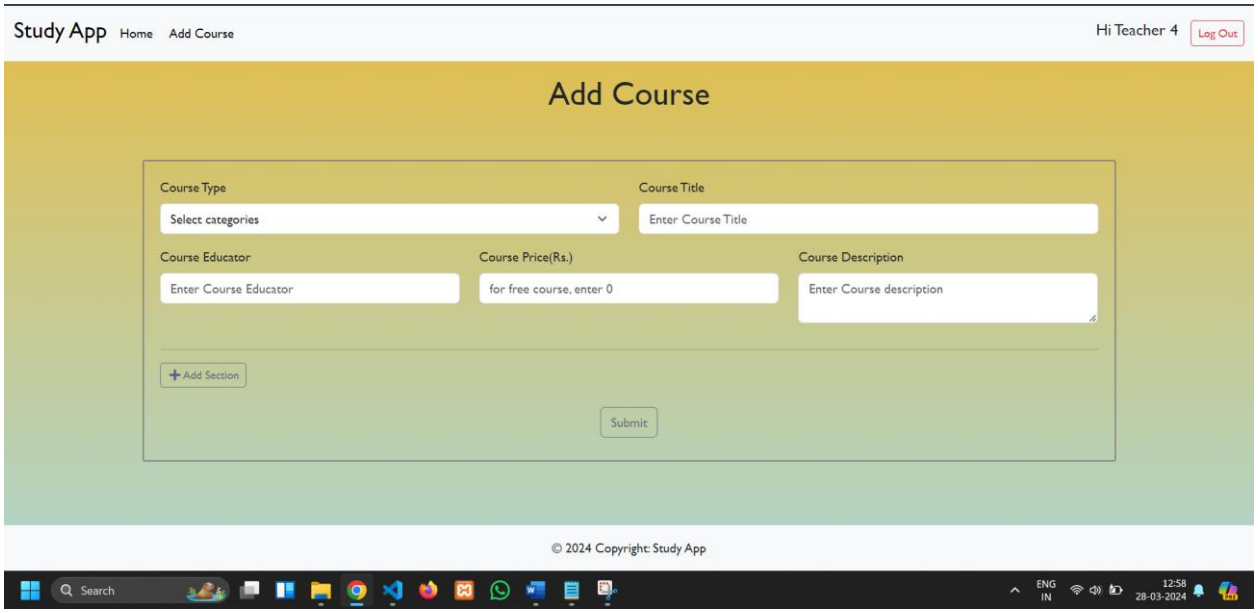


- **COURSES PAGE:**

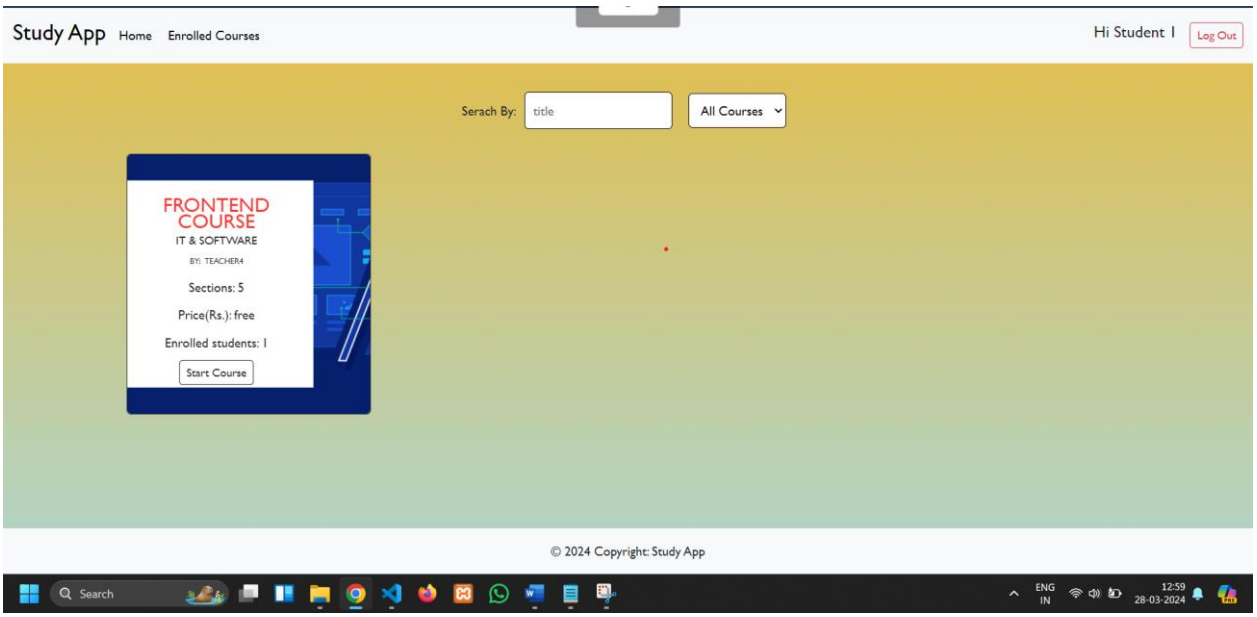
- **ADMIN DASHBOARD:**

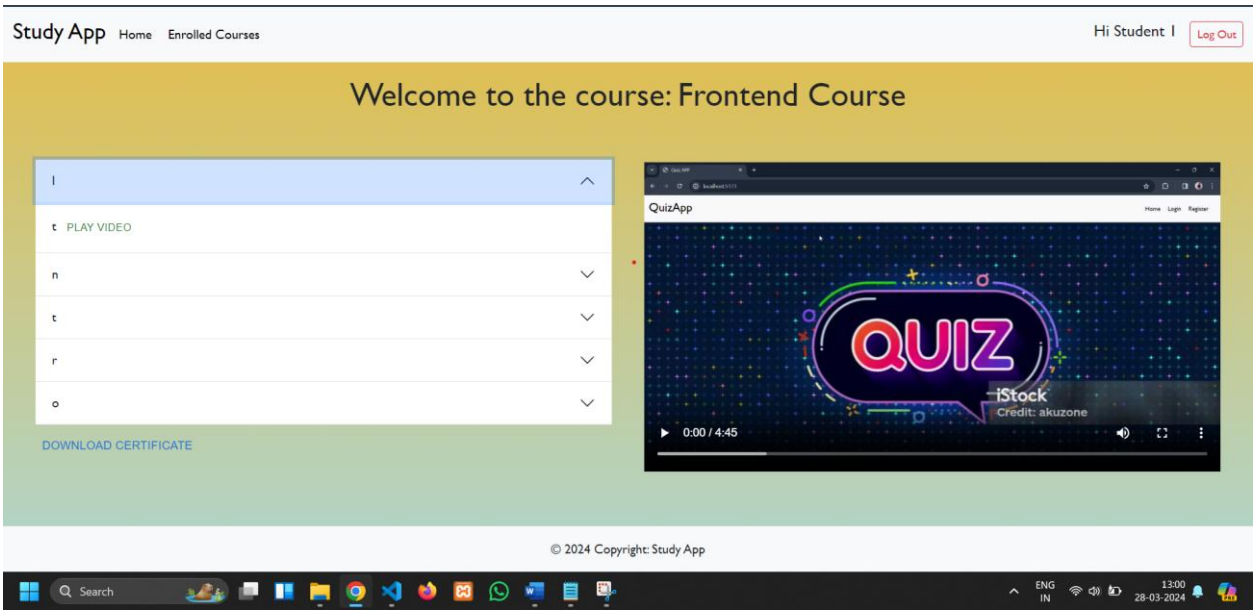


- **TEACHER DASHBOARD:**



- **STUDENT DASHBOARD:**





9. TESTING

- Manual Testing
  - ✓ All major modules were manually tested using real user scenarios.
  - ✓ Core features like user registration, login, course creation, enrollment, video playback, and discussion/chat were verified for correctness.
- API Testing with Postman
  - ✓ Backend APIs were tested independently using Postman.
  - ✓ Endpoints such as:
    - POST /register
    - POST /login
    - POST /courses/add
    - GET /courses/:id
    - POST /enroll
    - POST /messages
- UI Testing
  - ✓ All user interface components like buttons, forms, course cards, and chat inputs were tested for responsiveness and proper behavior.
  - ✓ UI was tested across various screen sizes including mobile, tablet, and laptop, to ensure responsive design and cross-device compatibility.
  - ✓ Form validations (e.g., required fields, email format) were verified.

10. KNOWN ISSUES

- ✓ **No email verification** is implemented after user registration, which may affect account authenticity.
- ✓ **Course discussion or doubt-solving chat does not auto-refresh**; users need to manually reload the page to see new messages.
- ✓ **File or assignment uploads** are not currently supported in course content or submissions.
- ✓ **Chat feature uses polling instead of real-time sockets**, which may cause delays in message delivery.
- ✓ **Role assignment (student/teacher/admin)** is manually handled; there's no user interface for admin role management.
- ✓ **Some forms lack client-side validation**, leading to poor UX when users forget to fill required fields like email or password.

11. FUTURE ENHANCEMENTS

- ✓ **Real-time Doubt Solving via Chat (Socket.IO):**
  Enable instant messaging between students and instructors for live query resolution.

- ✓ **Email & SMS Notifications:**
  Notify learners about course updates, new content, assignment deadlines, or feedback.

- ✓ **Learning History & Progress Analytics:**
  Allow admins and educators to view detailed analytics on course completion rates, student performance, and engagement trends.

- ✓ **File Upload & Assignment Submission:**
  Support for students to upload homework, project files, or essays directly through the platform.

- ✓ **Multilingual Support:**
  Offer regional language options to improve accessibility and inclusiveness for diverse learners.

- ✓ **Push Notifications:**
  Provide instant browser/mobile alerts about new course releases, instructor announcements, or reminders.

- ✓ **Instructor & Admin Role Management Panel:**
  Simplified interface for managing user roles, permissions, and responsibilities across courses.

- ✓ **Mobile App Version:**
  Develop a cross-platform **React Native** app to enable learning on-the-go for Android and iOS users.

Running the Application:

A. FRONTEND:

- To run the **React frontend**-
  - ✓ Open terminal and navigate to the frontend folder:

```
cd frontend
```

  - ✓ Install dependencies:

```
npm install
```

  - ✓ Start the frontend:

```
npm run dev
```

  - ✓ Open browser and visit:

    http://localhost:5173

B. BACKEND:

- To run the **Node.js + Express backend**:
  - ✓ Open another terminal and navigate to the backend folder:

```
cd backend
```

  - ✓ Install dependencies:

```
npm install
```

  - ✓ Start the backend server:

```
npm start
```

  - ✓ Server runs at:

    http://localhost:8000

DEMO LINKS

- Video Demo Link:

  https://drive.google.com/drive/folders/1W7iWq-6_-Kuc_1hwYQDEBCAve10TrBAO

- Drive Link:

  https://drive.google.com/drive/folders/15pjLvgXIpNMN9EdLx1FSAx4V3seb_QYC