LEARNHUB-Your Center for Skill Enhancement

Team ID: LTVIP2025TMID53104

Team Size: 4

Team Leader: Vyshnavi Kuncha

Team member: Veerla Shanmuka Yadav

Team member: Vemireddy Venkatathrivendra Reddy

Team member: Vemula Purna Vijya Sai Phani Kumar

INTRODUCTION

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.

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.

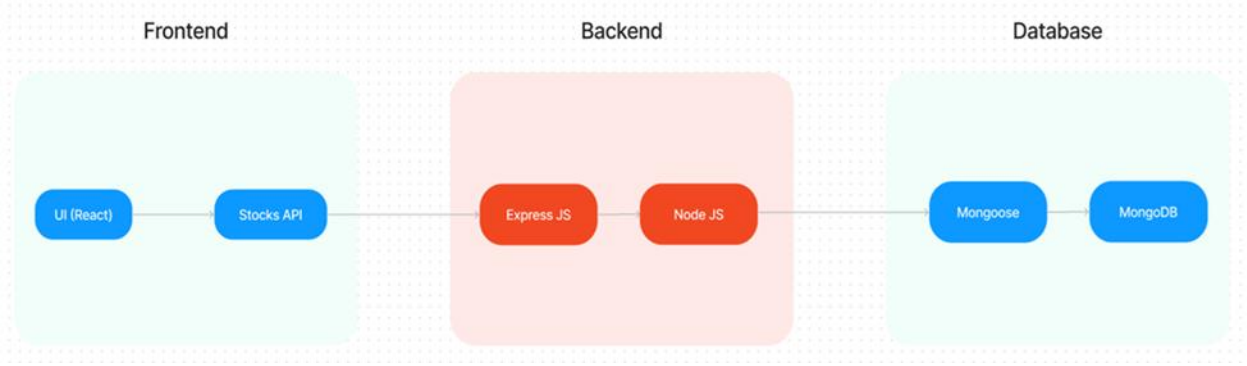
Course Completion and Certification: After completing all the modules and assignments, Sarah takes the final exam. Upon passing, she receives a digital certificate of completion, which she can download and add to her portfolio.

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.

TECHNICAL ARCHITECTURE



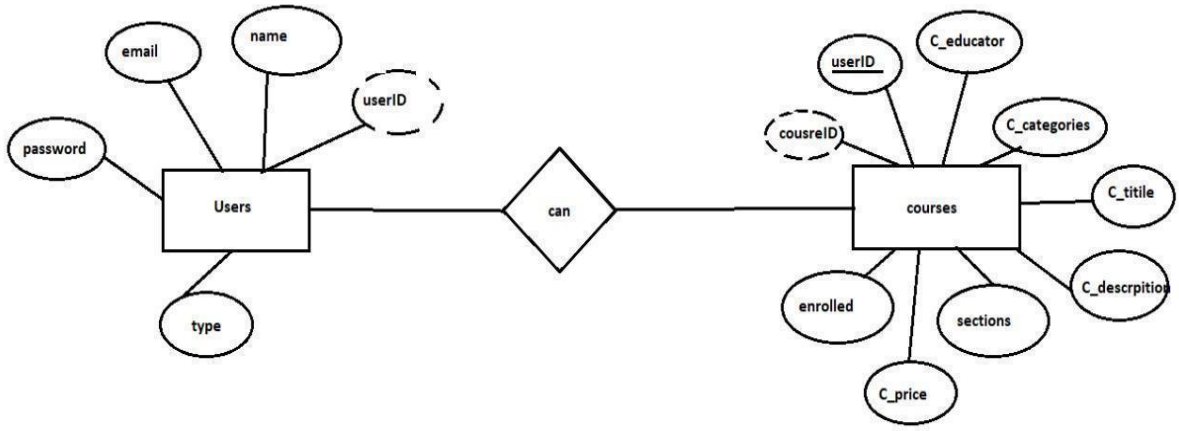
The technical architecture of OLP app follows a client-server model, where the frontend serves as the client and the backend acts as the server. The frontend encompasses not only the user interface and presentation but also incorporates the axios library to connect with backend easily by using RESTful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

PRE-REQUISITES:

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
```

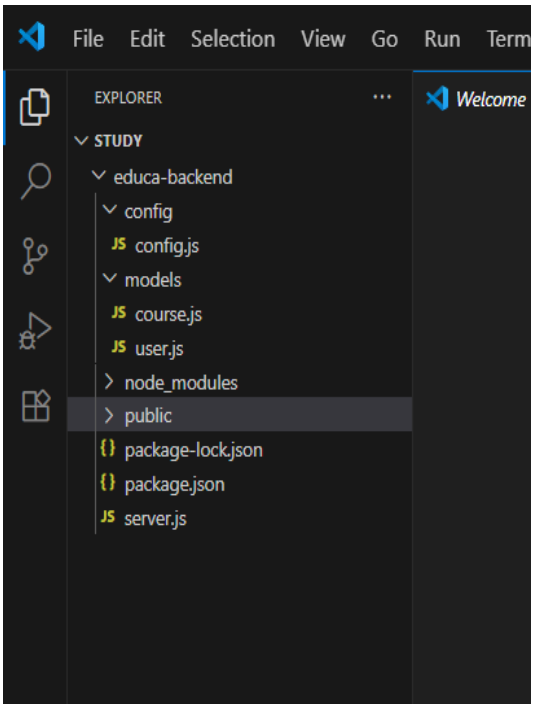
• The OLP app will be accessible at http://localhost:5172

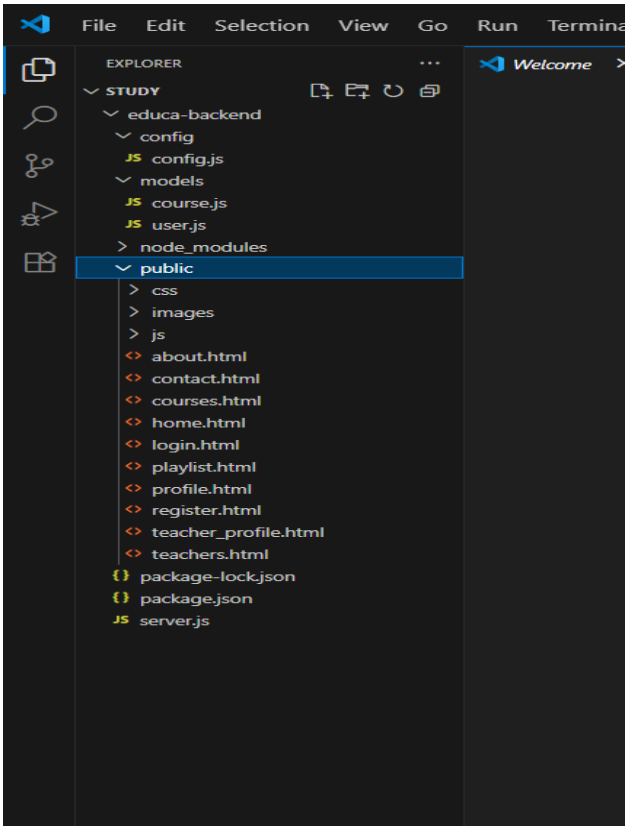
You have successfully installed and set up the Online learning app on your local machine. You can now proceed with further customization, development, and testing as needed.

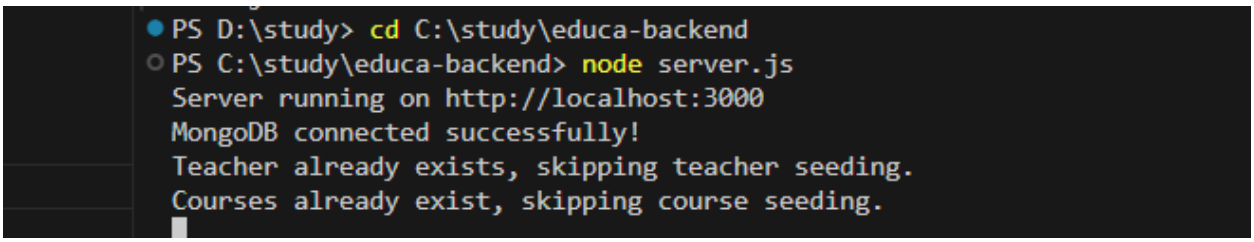
PROJECT STRUCTURE:

The first image is of the front part which shows all the files and folders that have been used in UI development

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







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:

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

Student:

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

Admin:

9. They can alter all the courses that are present in the app.
10. Watch out for all kinds of users in the app.
11. Record all the enrolled students that are enrolled in the course.
    https://drive.google.com/file/d/10HbOaLjKqgFYqZrRUlsMh81GD0iifb2X/view?usp=drivesdk

Milestone 1- Setup & configuration

1. Install required tools and software:

- Node.js.
- MongoDB.
- Create-react-app.

2. Create project folders and files:

- Client folders.
- Server folders.

3. Install Packages:

Frontend npm Packages

- Axios.
- React-Router –dom.
- Bootstrap.
- React-Bootstrap.
- React-icons.

Backend npm Packages

- Express.

- Mongoose.
- Cors.

Milestone 2- Backend Development

- **Setup express server**

1. Create index.js file in the server (backend folder).
2. define the port number, MongoDB connection string, and JWT key in the env file to access it.
3. Configure the server by adding cors, and body-parser.

- **Add authentication:** for this,

1. You need to make a middleware folder and in that make authMiddleware.js file for the authentication of the projects and can use in.

Ref: **backend.mp4**

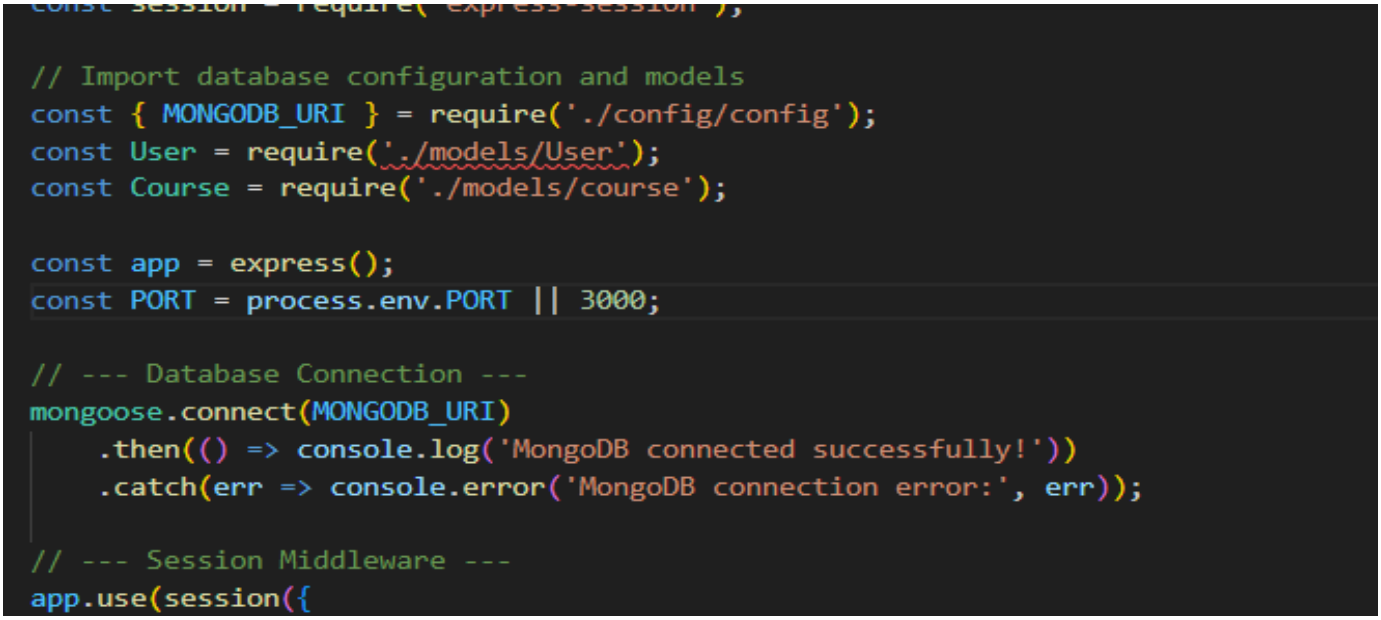
Milestone 3- Database

1. Configure MongoDB:

- Install Mongoose.
- Create database connection.
  Create Schemas & Models.

2. Connect database to backend:

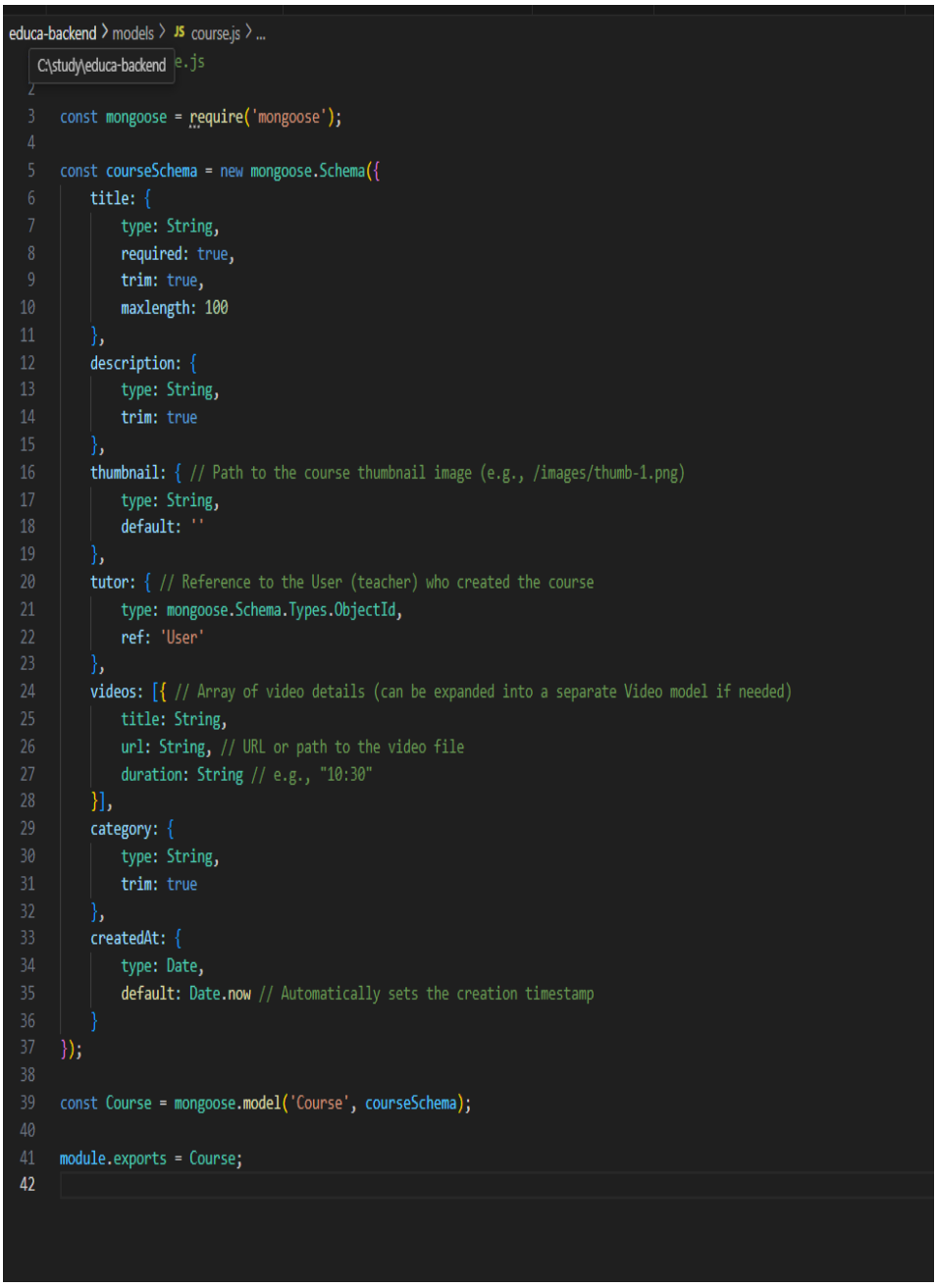
Now, make sure the database is connected before performing any of the actions through the backend. The connection code looks similar to the one provided below.



```
const session = require('express-session');

// Import database configuration and models
const { MONGODB_URI } = require('./config/config');
const User = require('./models/User');
const Course = require('./models/course');

const app = express();
const PORT = process.env.PORT || 3000;

// --- Database Connection ---
mongoose.connect(MONGODB_URI)
    .then(() => console.log('MongoDB connected successfully!'))
    .catch(err => console.error('MongoDB connection error:', err));

// --- Session Middleware ---
app.use(session({
```

3. Configure Schema: Firstly, configure the Schemas for MongoDB database, to store the data in such pattern. Use the data from the ER diagrams to create the schemas.

The schemas are looks like for the Application.


educa-backend > models > course.js > ...
C:\study\educa-backend
e.js
const mongoose = require('mongoose');
const courseSchema = new mongoose.Schema({
title: {
type: String,
required: true,
trim: true,
maxlength: 100
},
description: {
type: String,
trim: true
},
thumbnail: { // Path to the course thumbnail image (e.g., /images/thumb-1.png)
type: String,
default: ''
},
tutor: { // Reference to the User (teacher) who created the course
type: mongoose.Schema.Types.ObjectId,
ref: 'User'
},
videos: [{ // Array of video details (can be expanded into a separate Video model if needed)
title: String,
url: String, // URL or path to the video file
duration: String // e.g., "10:30"
}],
category: {
type: String,
trim: true
},
createdAt: {
type: Date,
default: Date.now // Automatically sets the creation timestamp
}
});
const Course = mongoose.model('Course', courseSchema);
module.exports = Course;

duca-backend > models > user.js > userSchema > enrolledCourses

```
// models/User.js

const mongoose = require('mongoose');

// Define the User Schema
const userSchema = new mongoose.Schema({
    name: {
        type: String,
        required: true,
        trim: true,
        maxlength: 50
    },
    email: {
        type: String,
        required: true,
        unique: true, // Ensures email addresses are unique
        trim: true,
        lowercase: true, // Stores emails in lowercase for consistency
        maxlength: 50
    },
    password: {
        type: String,
        required: true,
        minlength: 6 // Example: Minimum password length
    },
    profileImagePath: {
        type: String,
        default: '' // Store the path to the uploaded profile image
    },
    role: {
        type: String,
        enum: ['student', 'teacher', 'admin'], // Example roles
        default: 'student'
    },
    enrolledCourses: [{ // Array of ObjectIds referencing Course model for enrollment
        type: mongoose.Schema.Types.ObjectId,
        ref: 'Course' // This tells Mongoose to link to the 'Course' model
    }],
    createdAt: {
        type: Date,
        default: Date.now // Automatically sets the creation timestamp
    }
});

// Create the User Model from the schema
```

Milestone 4- Frontend Development

1. Setup React Application:

• Create React application.

• Configure Routing.

• Install required libraries.

2. Design UI components:

• Create Components.

• Implement layout and styling.

• Add navigation.

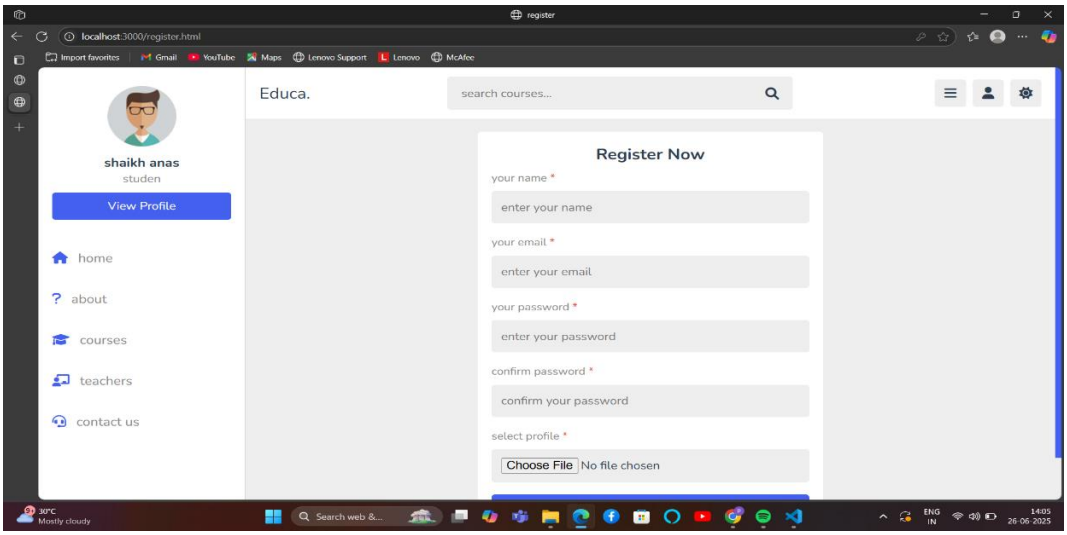
3. Implement frontend logic:

• Integration with API endpoints.

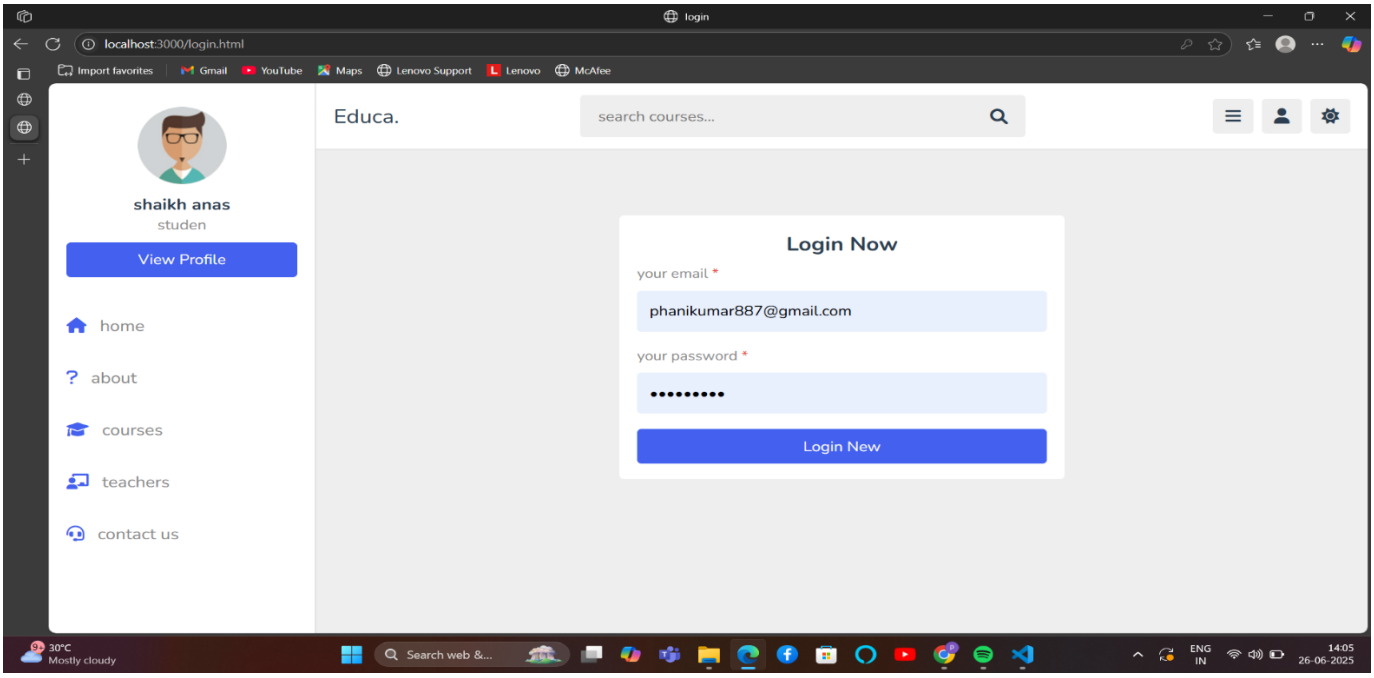
• Implement data binding

Milestone 5: 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.

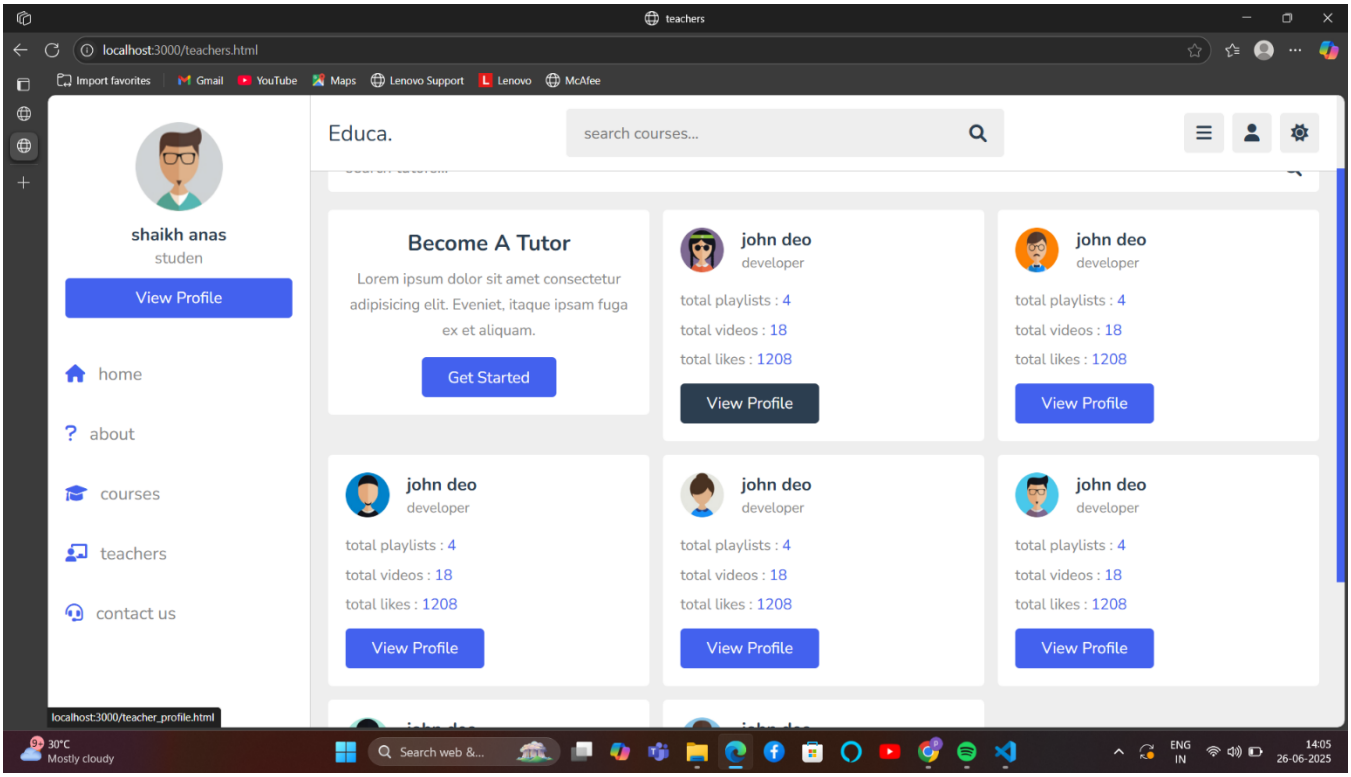
Register page:



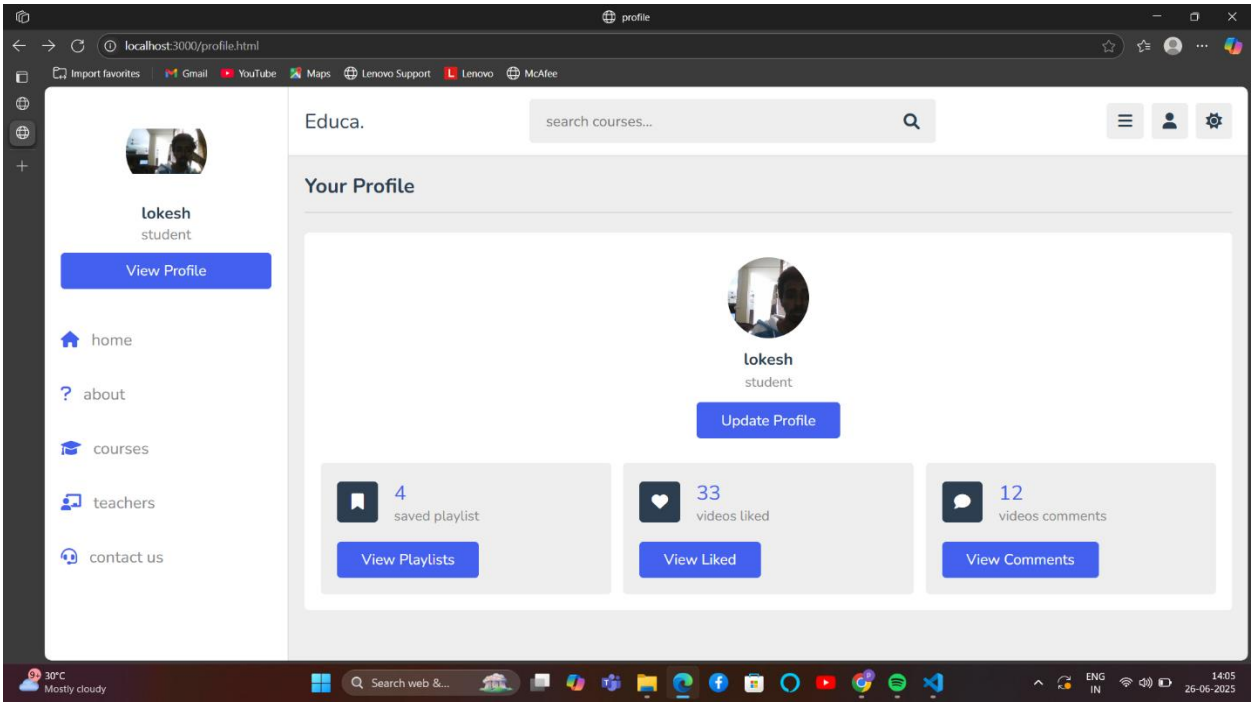
Login page:



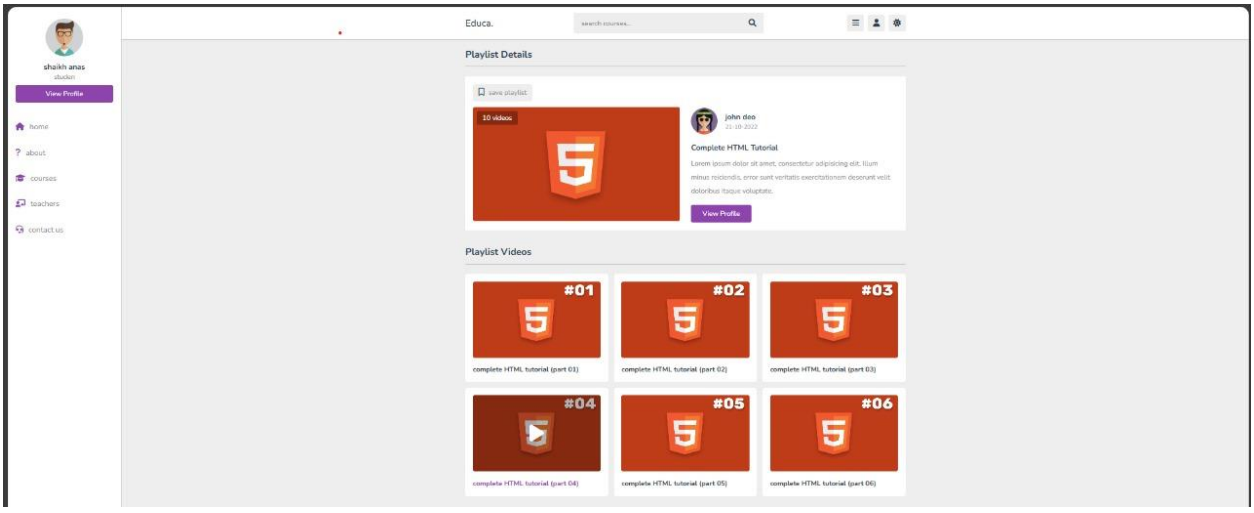
Teacher Dashboard:

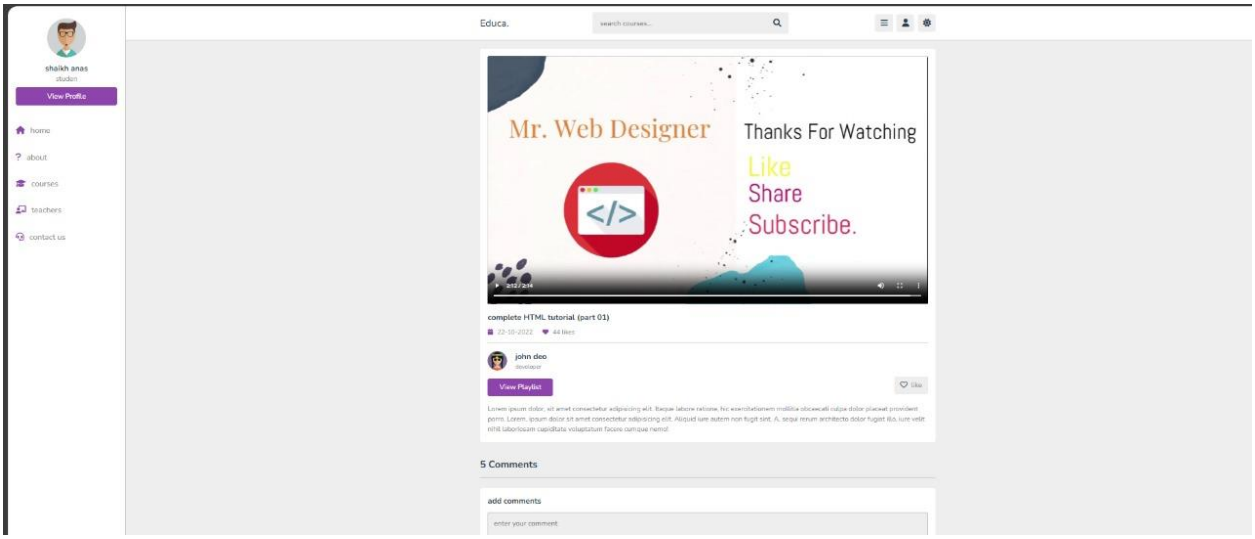


Student Profile:



Course playlist:





Note: https://vemulaphani162.github.io/travel/

DEMO LINK:
https://drive.google.com/file/d/10HbOaLjKqgFYqZrRUIsMh81GD0iifb2X/view?usp=drivesdk