**ONLINE LEARNING PLATFORM**

**INTRODUCTION :**

The Online Learning Platform is a web-based application designed to provide students and instructors with an efficient environment to engage in educational activities. The platform allows students to access courses, view tutorials, attend live sessions, and track their learning progress, while instructors can create and manage courses, interact with students, and evaluate performance. The system leverages modern technologies such as MERN stack to create a robust and scalable web application.

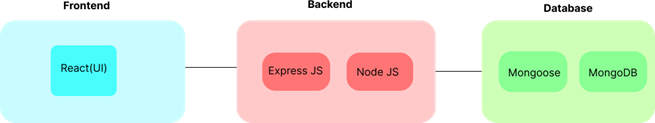
## **SCENARIO:**

The platform aims to serve both students and instructors in a highly interactive and personalized learning environment. Students can browse and enroll in courses, interact with instructors through discussions and messaging, and track their progress. Instructors can upload course materials, create assignments, grade submissions, and host live sessions. The system also supports payment integrations for course purchases using Razorpay.

**KEY FEATURES:**

* **User Registration and Authentication**: Both students and instructors can sign up, log in, and maintain secure sessions using JWT tokens.
* **Course Management**: Instructors can create, update, and manage courses.
* **Learning Dashboard**: Students can track progress, view enrolled courses, and access content.
* **Live Sessions**: Integration with video conferencing tools for live lectures and student-instructor interaction.
* **Payment Gateway**: Razorpay integration for course purchases.
* **Email Notifications**: Automated email system for course-related updates.
* **Security**: User authentication and authorization through JWT tokens, password hashing, and helmet for security headers.

## **TECHNICAL ARCHITECTURE:**



The platform uses the MERN stack to ensure efficient management of both the frontend and backend components.

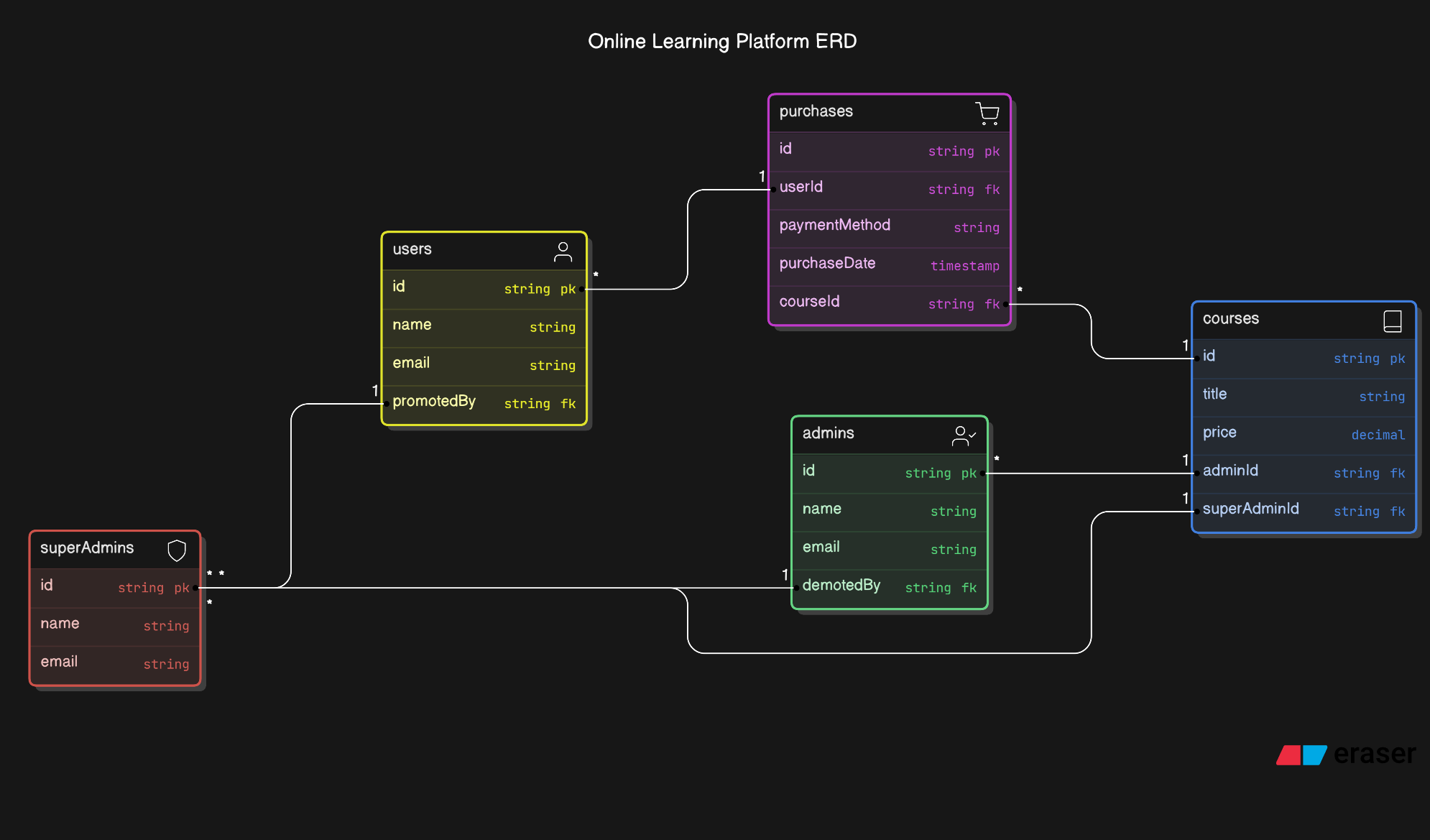
**Frontend:**

* **React**: Core library used for building the user interface. React provides a responsive and interactive experience, where components are reusable.
* **React-Router-Dom**: Used for handling routing in a single-page application (SPA).
* **Axios**: A promise-based HTTP client for making API requests to the backend server.
* **React-Icons**: Provides icons (such as Md and Io icons) for various UI components.
* **React-Hot-Toast**: Used for displaying toast notifications to users, providing real-time feedback on actions such as course enrollment or payment success.
* **React-Google-Recaptcha**: Integrated to add CAPTCHA verification for user registration and login to prevent bot activities.

**Backend:**

* **Express**: A minimal and flexible Node.js web application framework that is used to build the RESTful API.
* **Mongoose**: An ODM (Object Data Modeling) library for MongoDB, allowing for easy interaction with the MongoDB database.
* **Dotenv**: A module that loads environment variables from a .env file, helping keep sensitive data like API keys and database credentials secure.
* **Bcrypt**: A library used for hashing passwords and ensuring user credentials are stored securely.
* **Jsonwebtoken**: Used to generate JSON Web Tokens (JWT) for secure authentication and authorization of users.
* **Nodemailer**: A library for sending emails, used to notify users about registration, course updates, and other events.
* **Razorpay**: Integration for handling online payments for course purchases.
* **UUID**: Generates unique identifiers for users, orders, and other entities in the system.
* **Helmet**: A middleware used to secure the application by setting various HTTP headers to prevent attacks such as XSS, clickjacking, and others.
* **Morgan**: A HTTP request logger middleware used for logging request details, useful for debugging and monitoring.
* **Custom Middlewares**:
* **isAuth**: Middleware to check if a user is authenticated before allowing access to certain routes.
* **TryCatch**: A custom middleware to handle errors more efficiently by wrapping API logic in try-catch blocks.
* **Custom Modules**:
* **sendMail**: A custom module that uses Nodemailer to send emails for various notifications (e.g., course registration, password reset).
* **connectDb**: A custom module for establishing and managing the database connection.

**ER DIAGRAM:**



## **FEATURES:**

* **Authentication & Authorization**: Secure login and registration system with JWT tokens for user authentication. Middleware ensures that only authorized users can access certain routes.
* **Course Management**: Instructors can create, update, and delete courses; students can browse, enroll, and view courses.
* **Interactive Learning**: Students can participate in live sessions, view multimedia content, and complete assignments.
* **Payment Integration**: The platform integrates Razorpay to facilitate course payments securely.
* **Notifications**: Automated email notifications for course updates, password reset requests, and other platform-related activities.
* **Security**: The platform uses helmet for secure HTTP headers, bcrypt for password hashing, and JWT for secure session management.
* **Real-time Feedback**: React-Hot-Toast notifications provide users with instant feedback on actions like course enrollment, payment status, etc.

**PREREQUISITES:**

To develop a Online Learning Platform Using React JS, Node.js,Express.js and MongoDB, there are several prerequisites you should consider. Here are the key prerequisites for developing such an application:

**Node.js and npm:** Install Node.js, which includes npm (Node Package Manager), on your development machine. Node.js is required to run JavaScript on the server side. • Download: https://nodejs.org/en/download/

• Installation instructions: https://nodejs.org/en/download/package-manager/

**MongoDB:** Setup a MongoDB To Username,Password,Payment Details,Courses ..etc. Install MongoDB locally or use a cloud-based MongoDB service.

• Download: <https://www.mongodb.com/try/download/community>

• Installation instructions: <https://docs.mongodb.com/manual/installation/>

**Express.js:** Express.js is a web application framework for Node.js. Install Express.js to handle server-side routing,middleware, and API development.

• Installation: Open your command prompt or terminal and run the following command: **npm install express**

**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. To 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.

**Front-end Framework:** Utilize Angular to build the user-facing part of the application, including product listings, booking forms, and user interfaces for the admin dashboard.

**Version Control**: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

**• Git:** Download and installation instructions can be found at: <https://git> scm.com/downloads

**Development Environment:** Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

• **Visual Studio Code:** Download from <https://code.visualstudio.com/download>

**• Sublime Text:** Download from <https://www.sublimetext.com/download>

**To Connect the Database with Node JS go through the below provided link:**

**Link:** <https://www.section.io/engineering-education/nodejs-> mongoosejs-mongodb/

**To run the existing OLP project downloaded from github:**

Follow below steps:

**Clone the repository:**

• Open your terminal or command prompt.

• Navigate to the directory where you want to store the e-commerce app.

• Execute the following command to clone the repository:

**Git clone:** <https://github.com/ganeshamoorthy246/Online-Learning-Platform-MERN.git>

**Install Dependencies:**

• Navigate into the cloned repository directory:

**Cd Online-Learning-Platform-MERN**

• Navigate to frontend directory:

**Cd frontend**

• Install the required dependencies by running the following command:

**npm install**

• Navigate to server directory:

**Cd server**

• Install the required dependencies by running the following command:

**npm install**

**Start the Development Server:**

• Locate to the server Directory

• To start the development server, execute the following command:

**npm run start**

**Start the Frontend Server :**

• Locate to the frontend Directory

• To start the development , execute the following command:

**npm run dev**

• The Online Learning app will be accessible at http://localhost:5000 by default. You can change the port configuration in the .env file if needed.

**USER & ADMIN & SUPERADMIN FLOW :**

**User Flow :**

* **Course Purchase:** Users can browse courses available on the platform and purchase them using Razorpay as the payment gateway.
* **Dashboard Access:** After purchasing a course, the user can access their personal dashboard where they can:
  + View a list of all the courses they’ve purchased.
  + Track progress, see course completion status, and access course content.

**Admin Flow :**

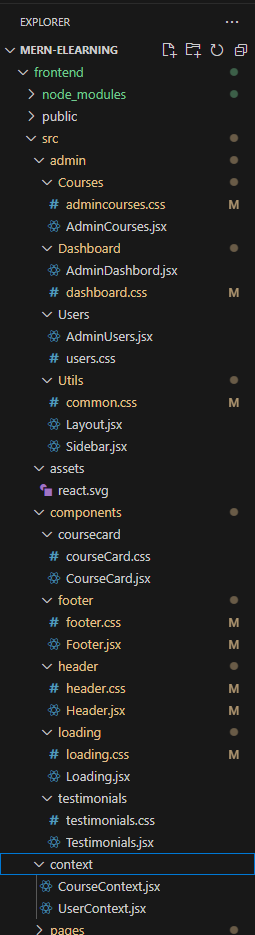
* **Course Management:** Admins can upload courses to the platform, including:
  + Setting prices for the courses.
  + Adding descriptions, lessons, and any related materials.
* **Course Content Control:** Admins have control over the course content, which can include editing, updating, or removing courses as needed.
* **Limited User Management:** Admins may have the ability to manage a limited set of users, such as:
  + Viewing user activity or course purchase history.

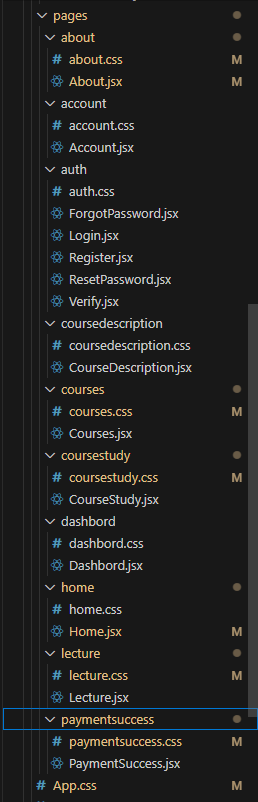
**SuperAdmin Flow:**

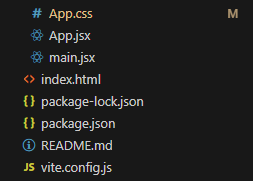
* **Full Control Over Courses:** Super Admins have all the rights of an Admin, including uploading courses and setting prices.
* **User and Admin Monitoring:** Super Admins can monitor all the users and admins on the platform, including:
  + Viewing user activities (course purchases, learning progress, etc.).
  + Reviewing admin actions and ensuring compliance with platform rules.
* **User and Admin Promotion/Demotion:**
  + **Promote User to Admin:** The Super Admin can promote a regular user to an Admin role, granting them course management privileges.
  + **Demote Admin to User:** The Super Admin can demote an Admin back to a regular user, revoking their course management privileges.

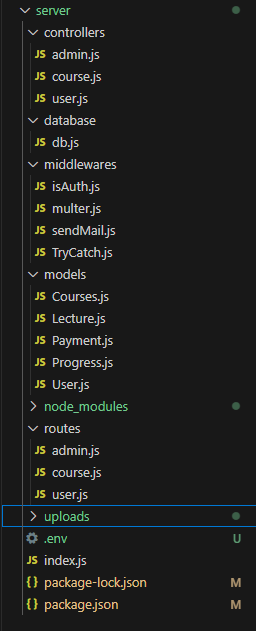
This system structure ensures that there is a clear division of roles and responsibilities, with the Super Admin having the highest level of control and monitoring capabilities.

## **PROJECT STRUCTURE:**









This structure assumes a React app and follows a modular approach. Here's a brief explanation of the main directories and files:

• src/components: Contains components related to the application such as, register, login, home, etc.,

• src/pages has the files for all the pages in the application.

## **PROJECT SETUP AND CONFIGURATION:**

**Install required tools and software:**

• Node.js.

Reference Article: <https://www.geeksforgeeks.org/installation-of-node-js-on-windows/>

• Git.

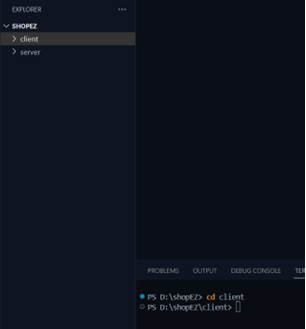
Reference Article: <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

**Create project folders and files:**

• Client folders.

• Server folders

Referral Image:

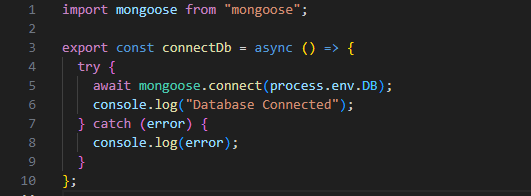


**Server :**

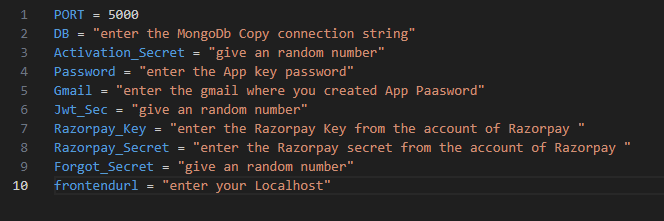
**Create index.js :**



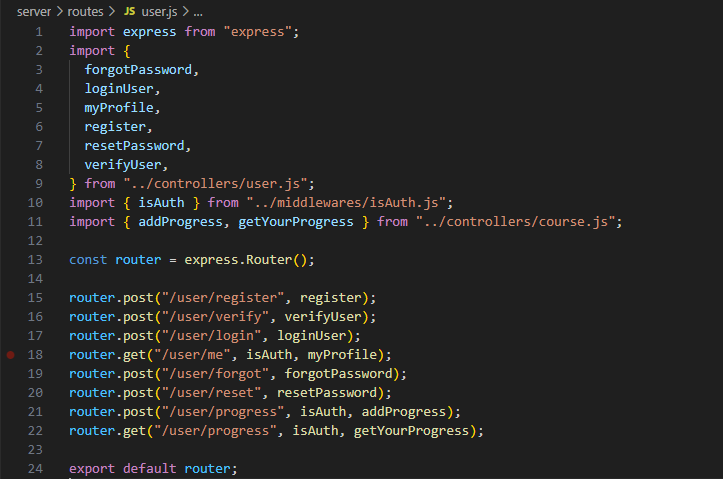
**Create Database db.js:**



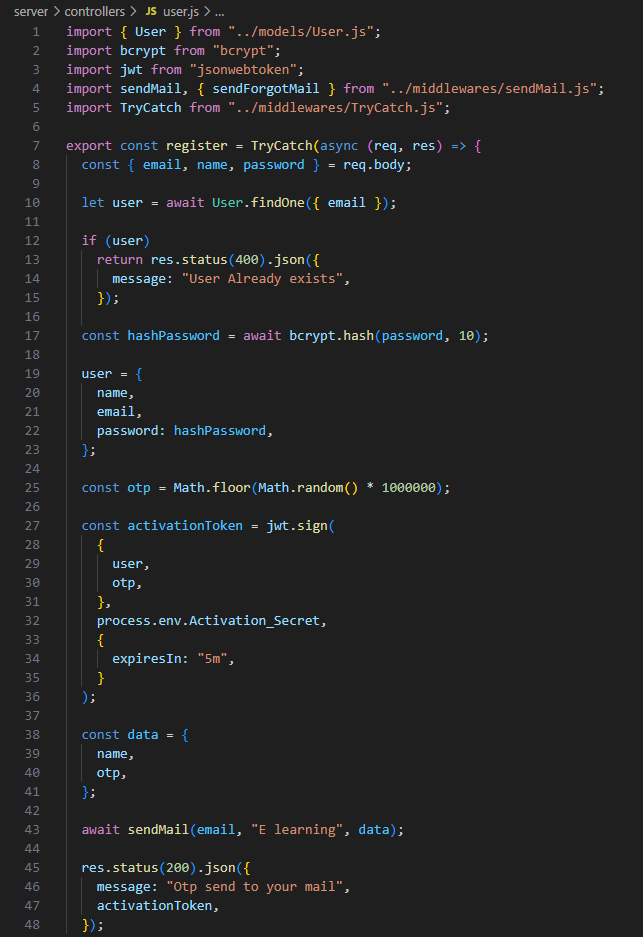
**Enter All the Details in .env file :**



**Create user.js inside routes :**



**Create user.js inside controllers :**

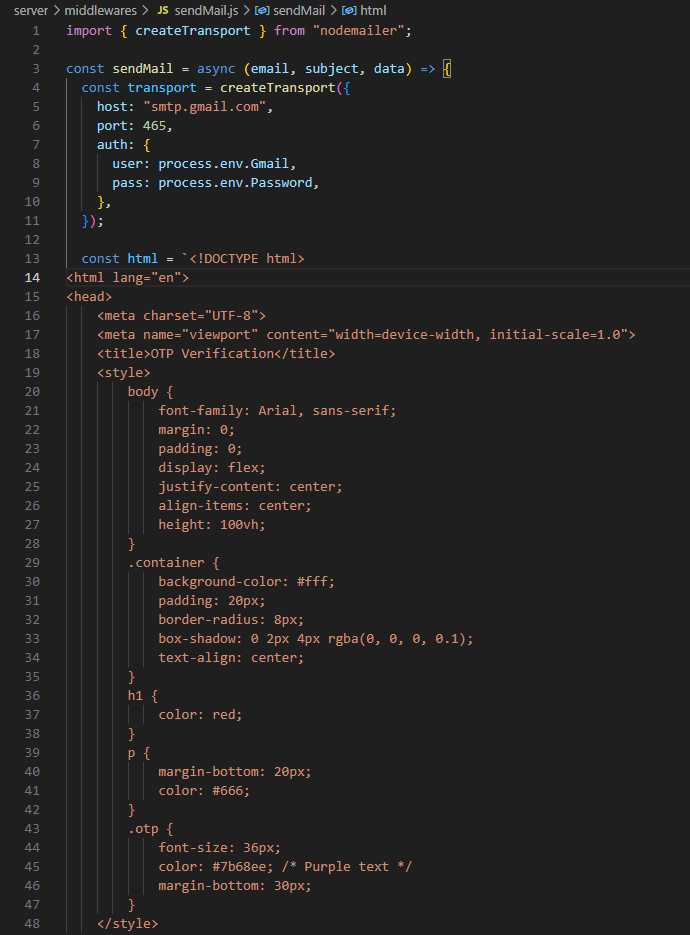


**Create user.js inside models:**



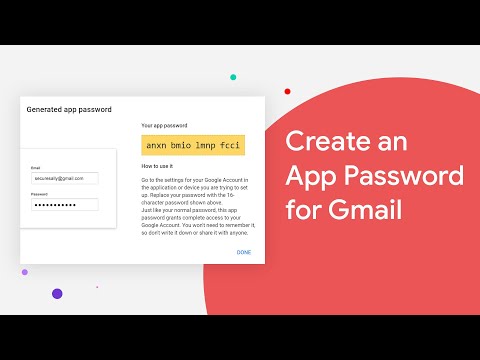
**Create sendMail inside middlewares :**

**This is for getting OTP Password in Mail**



**Create an App Password :**

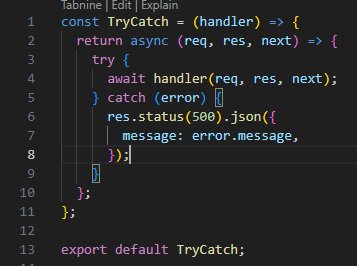
**LINK**: <https://youtu.be/hXiPshHn9Pw?si=CJ9C-OeTleIJG-o0>

[](https://youtu.be/hXiPshHn9Pw?si=CJ9C-OeTleIJG-o0)

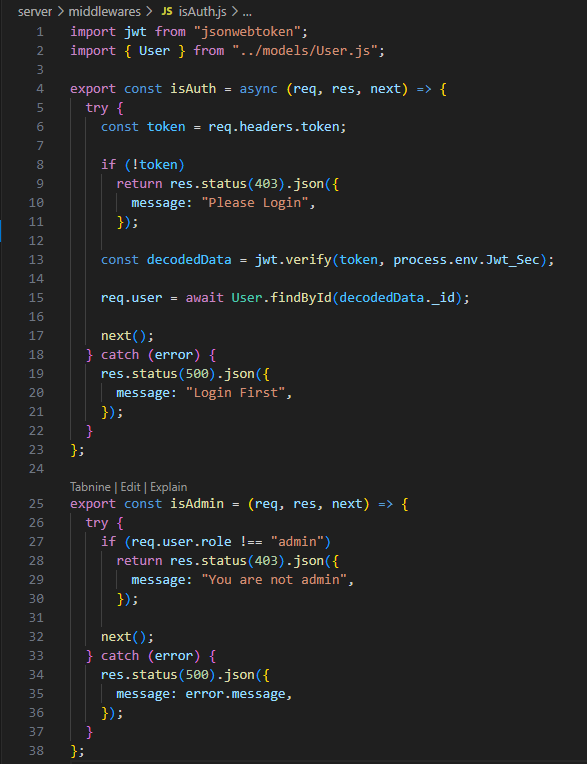
Get the App Password Key From Your Mail And Save in Password in the .env File and Enter the Gmail Inside the .env file

+

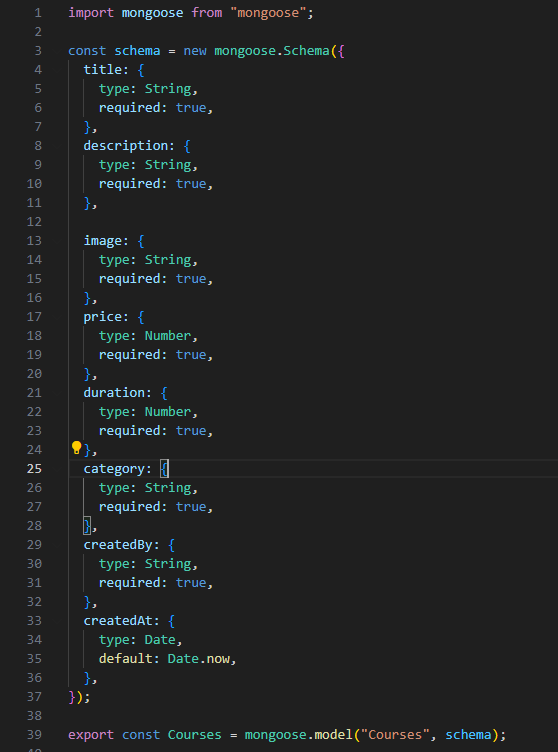
**Create TryCatch.js inside middleware:**



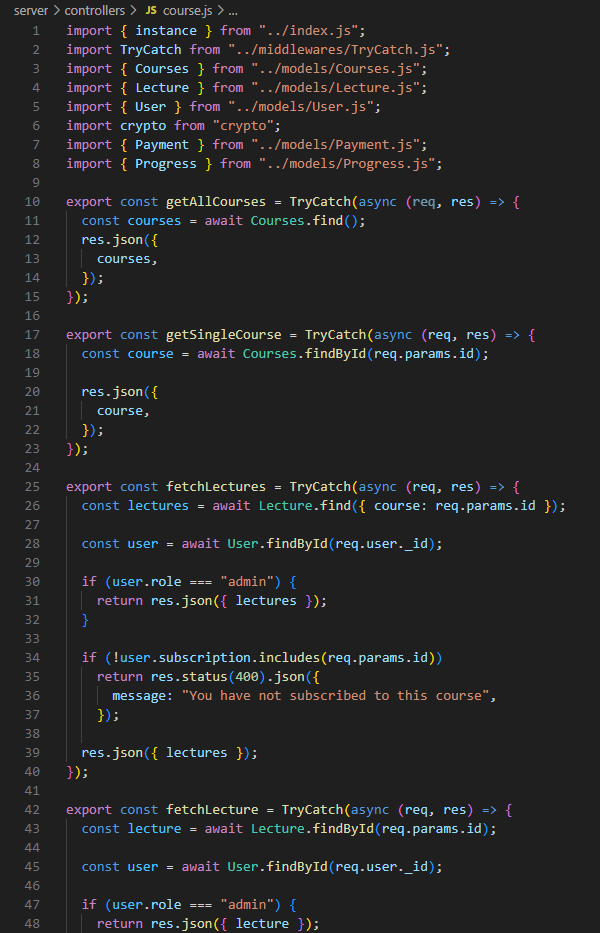
**Create isAuth.js inside middlewares:**



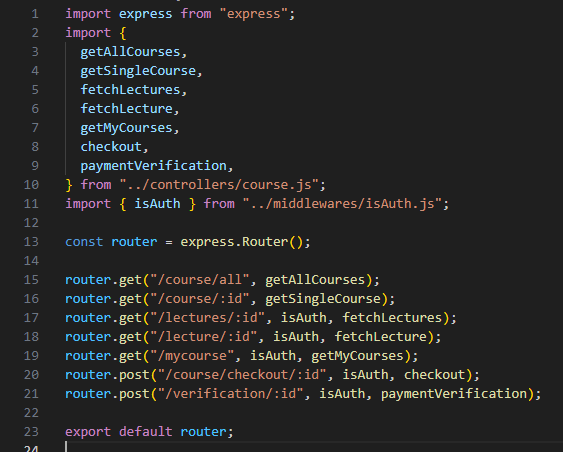
**Create courses.js inside models :**



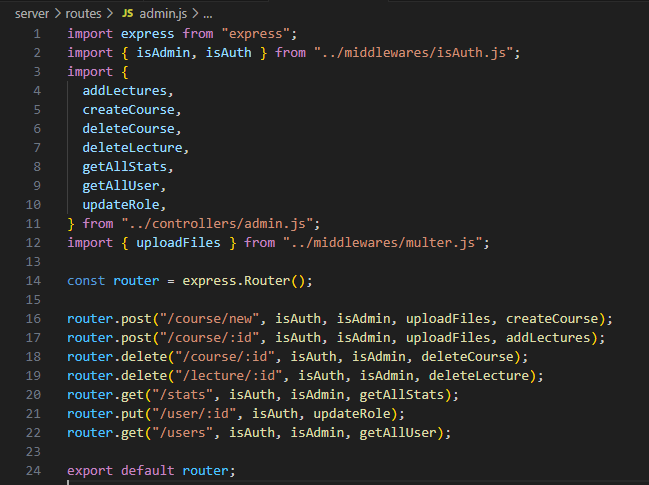
**Create courses.js inside controllers :**



**Create courses.js inside routes:**



**Create admin.js inside routes :**



**Create admin.js inside controllers :**



**Create multer.js inside middlewares :**



**Create an upload folder :**

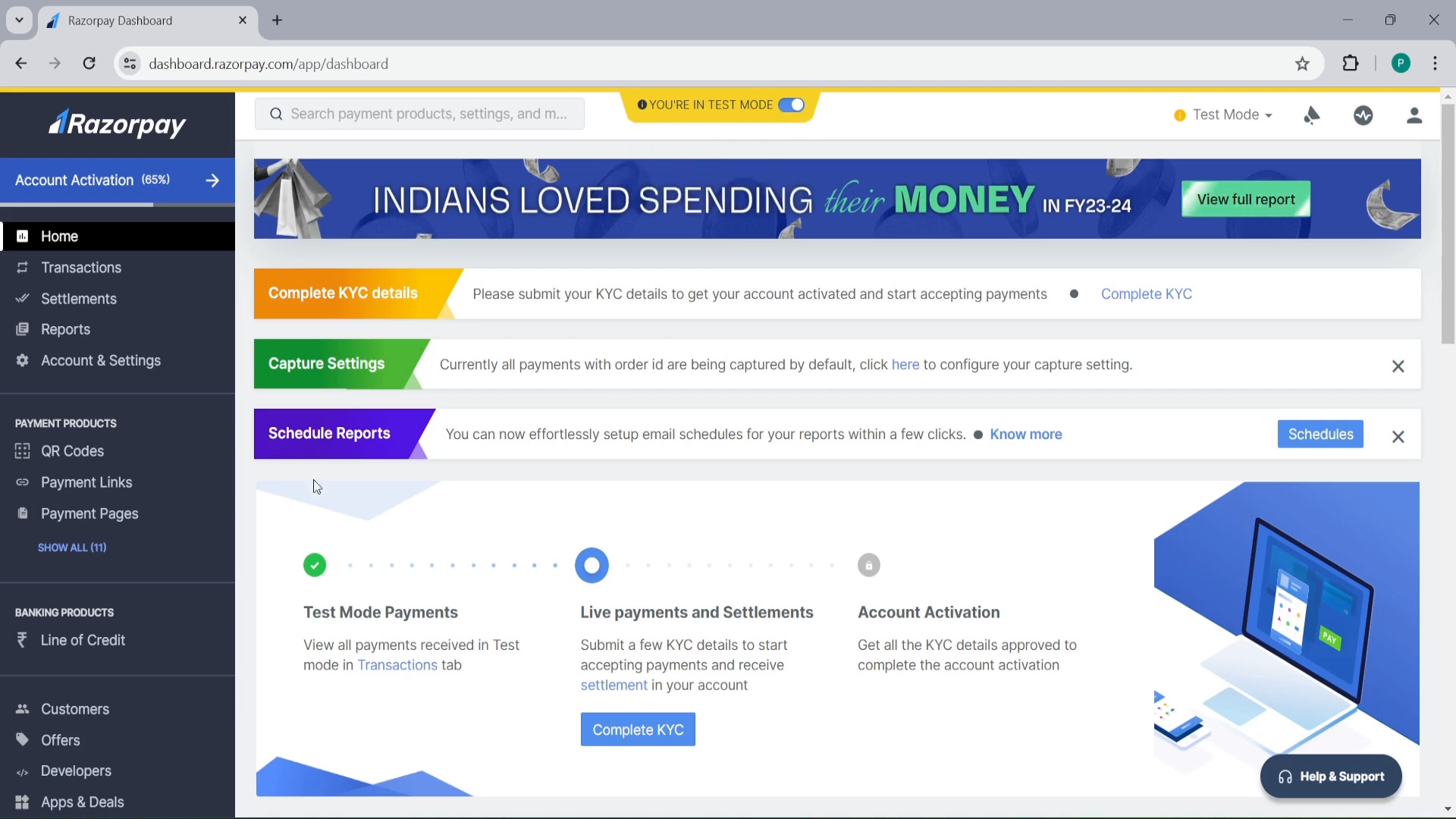
It is used to save the course videos ,documents ..etc.

**Create Lecture.js inside models :**

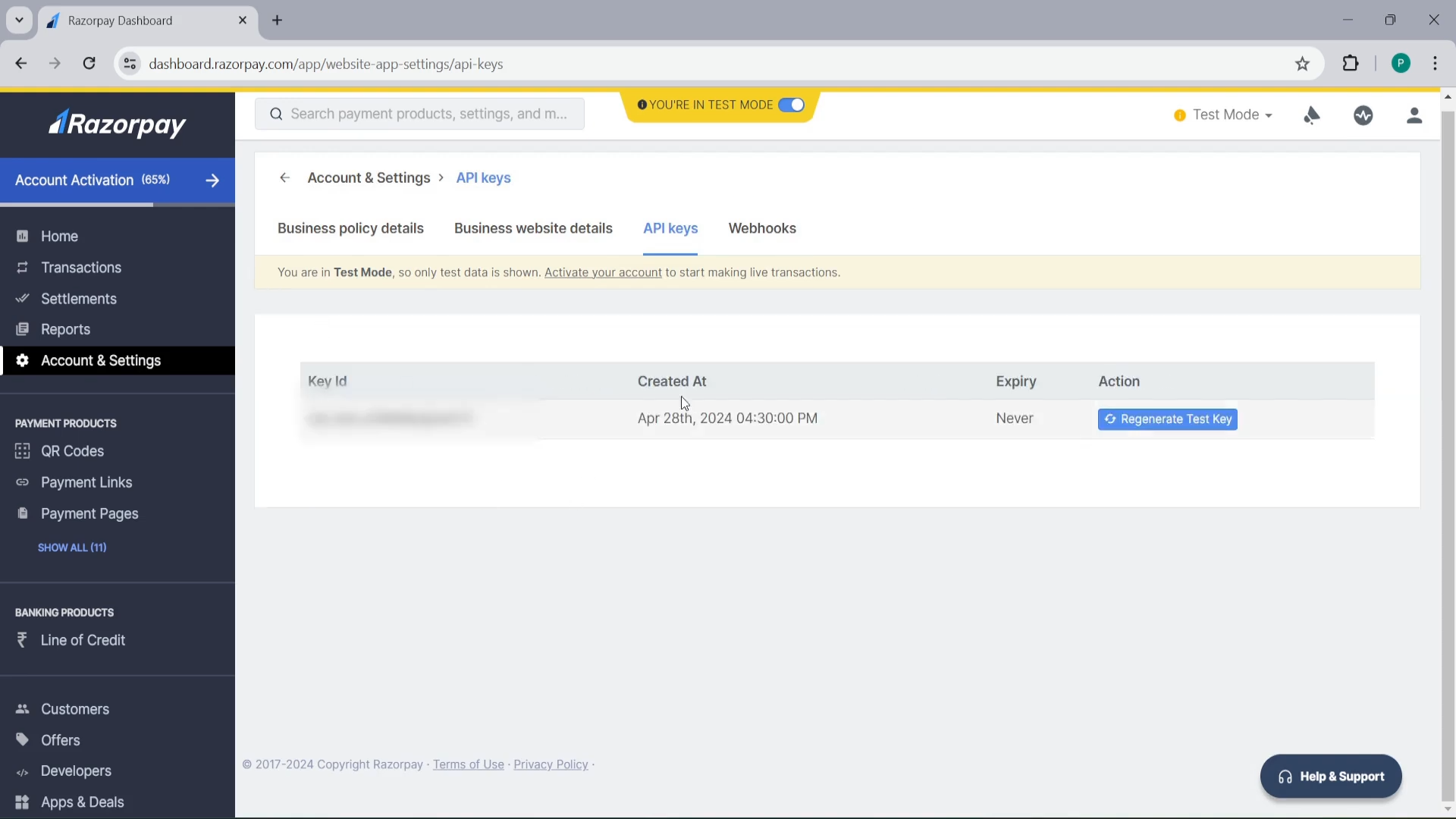


**RazorPay Payments :**

• Go to the razorpay website create an account

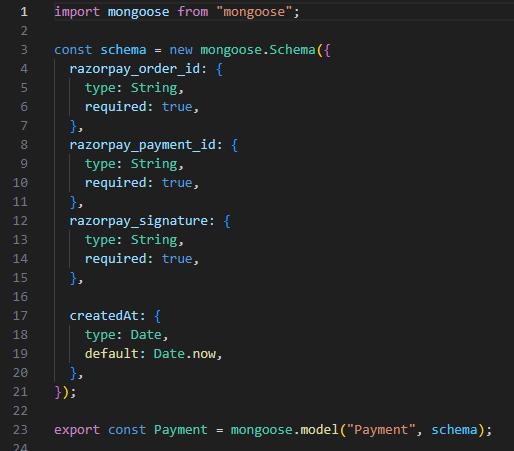


• Once you go to the Account&Settings Page :

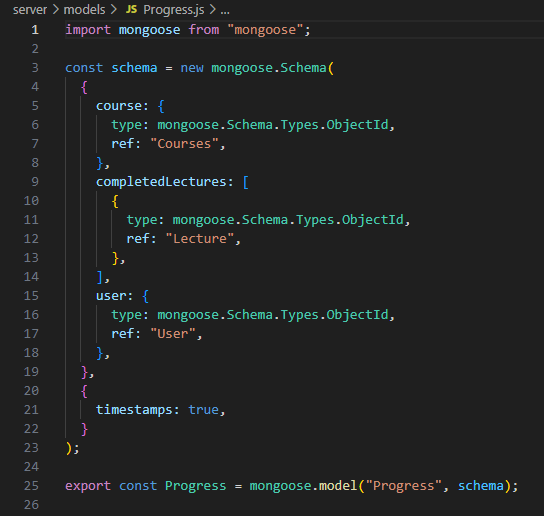


• Create an new Test Key Copy Key and Secret key and paste it in the .env file

**Create Payment.js inside models :**



**Create Progress.js inside models :**

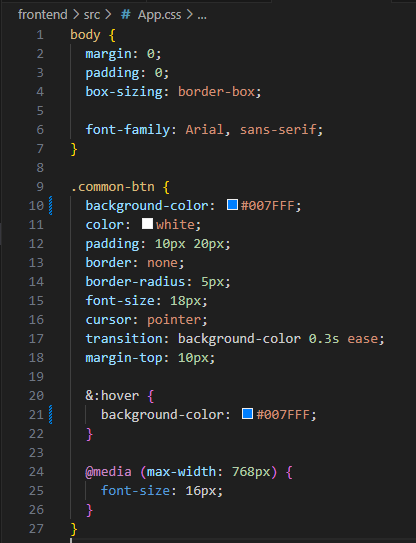


**FRONTEND :**

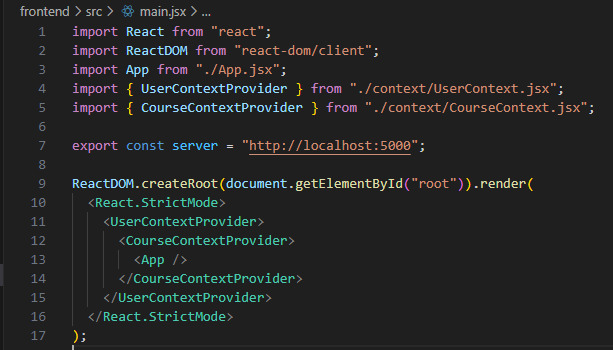
**Create App.jsx inside src folder :**



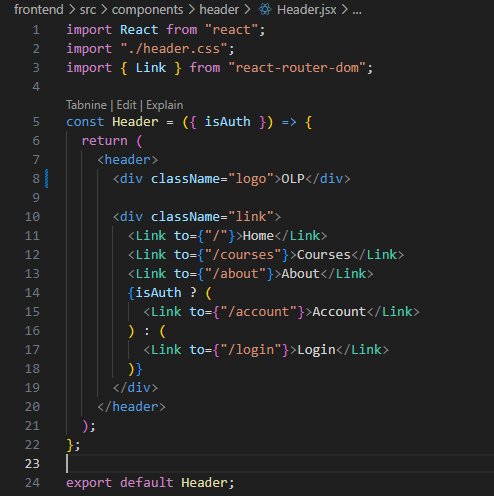
**Create App.css inside src folder :**



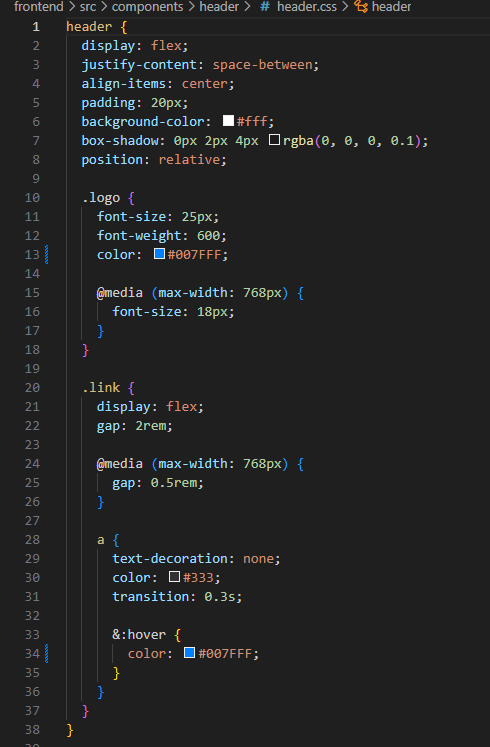
**Create main.jsx inside src folder :**



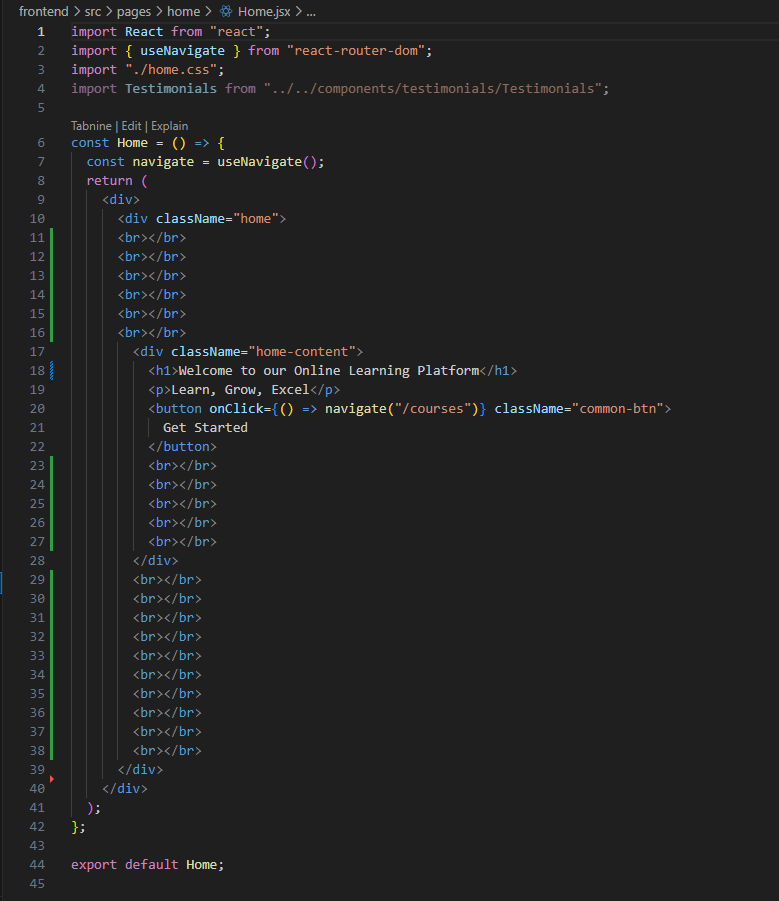
**Create Header.jsx inside components folder :**



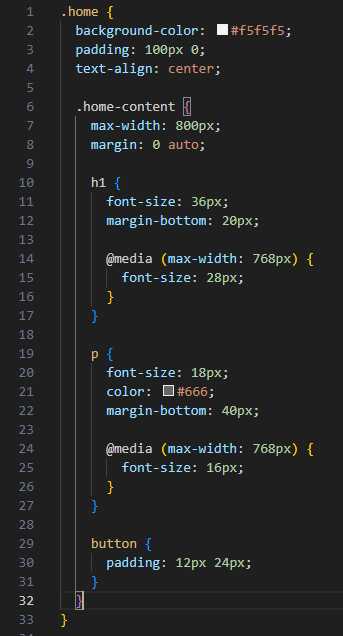
**Create Header.css inside components folder :**



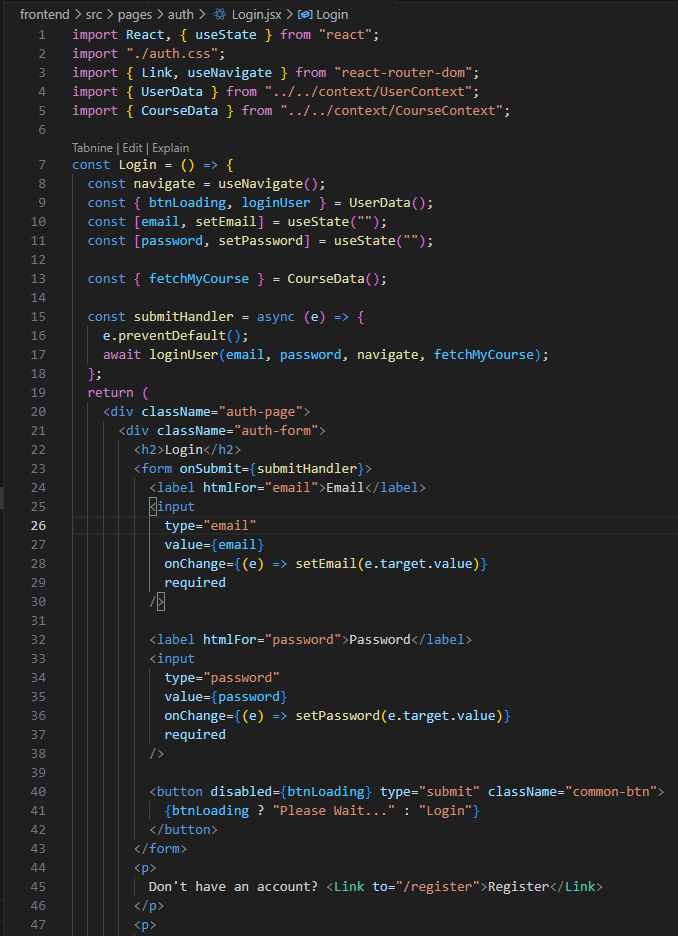
**Create Home.jsx inside pages folder :**



**Create Home.css inside pages folder:**



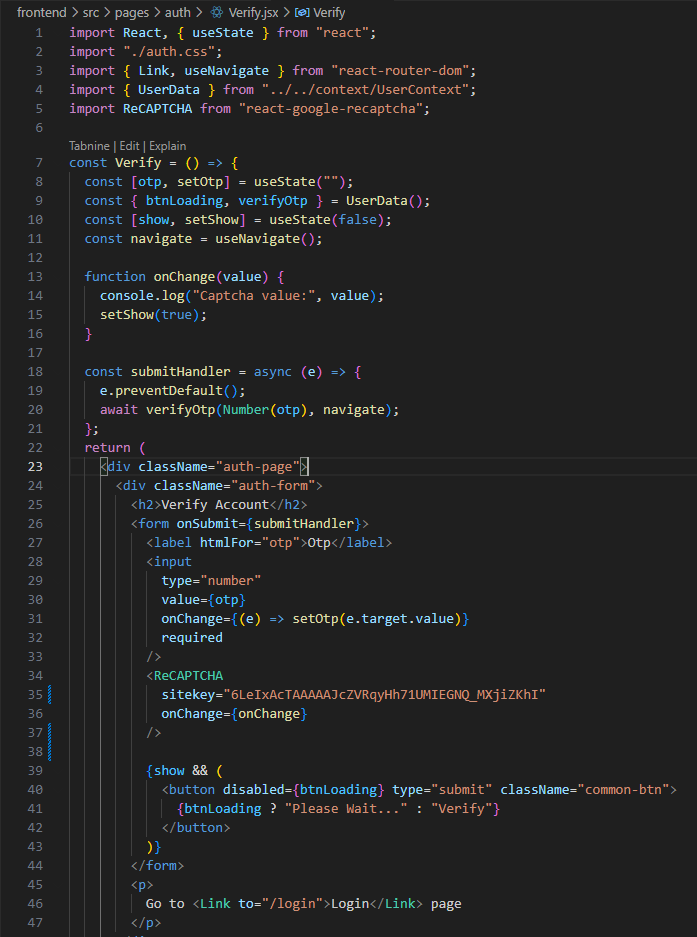
**Create Login.jsx :**



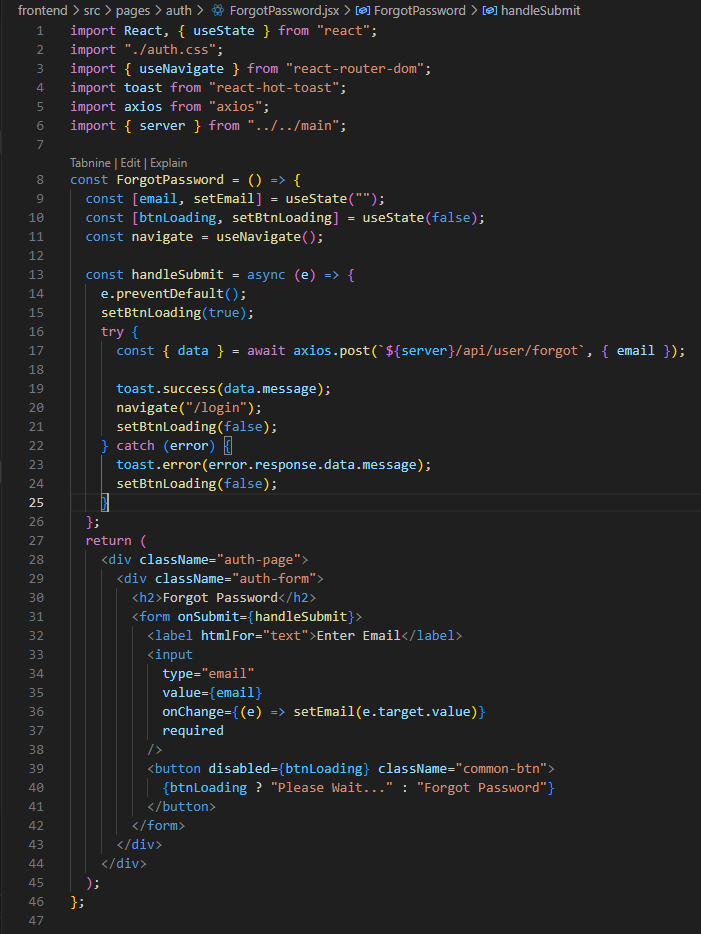
**Create Register.jsx :**



**Create Verify.jsx :**



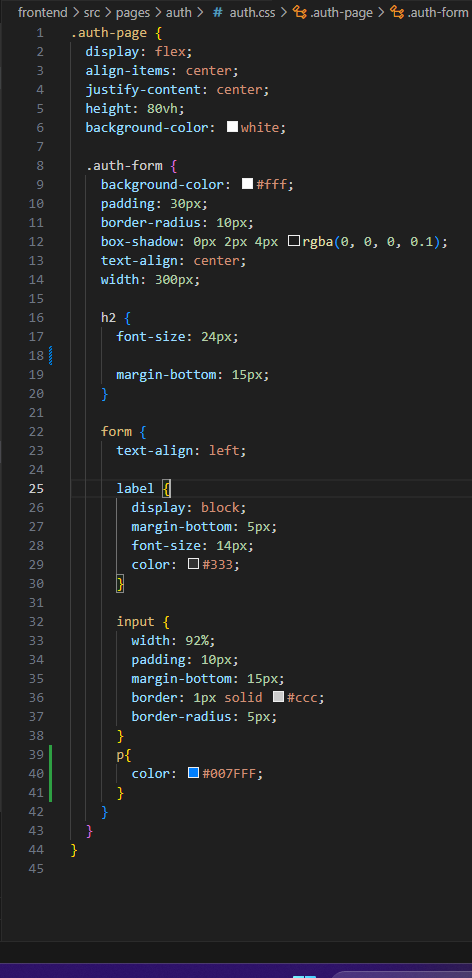
**Create ForgotPassword.jsx :**



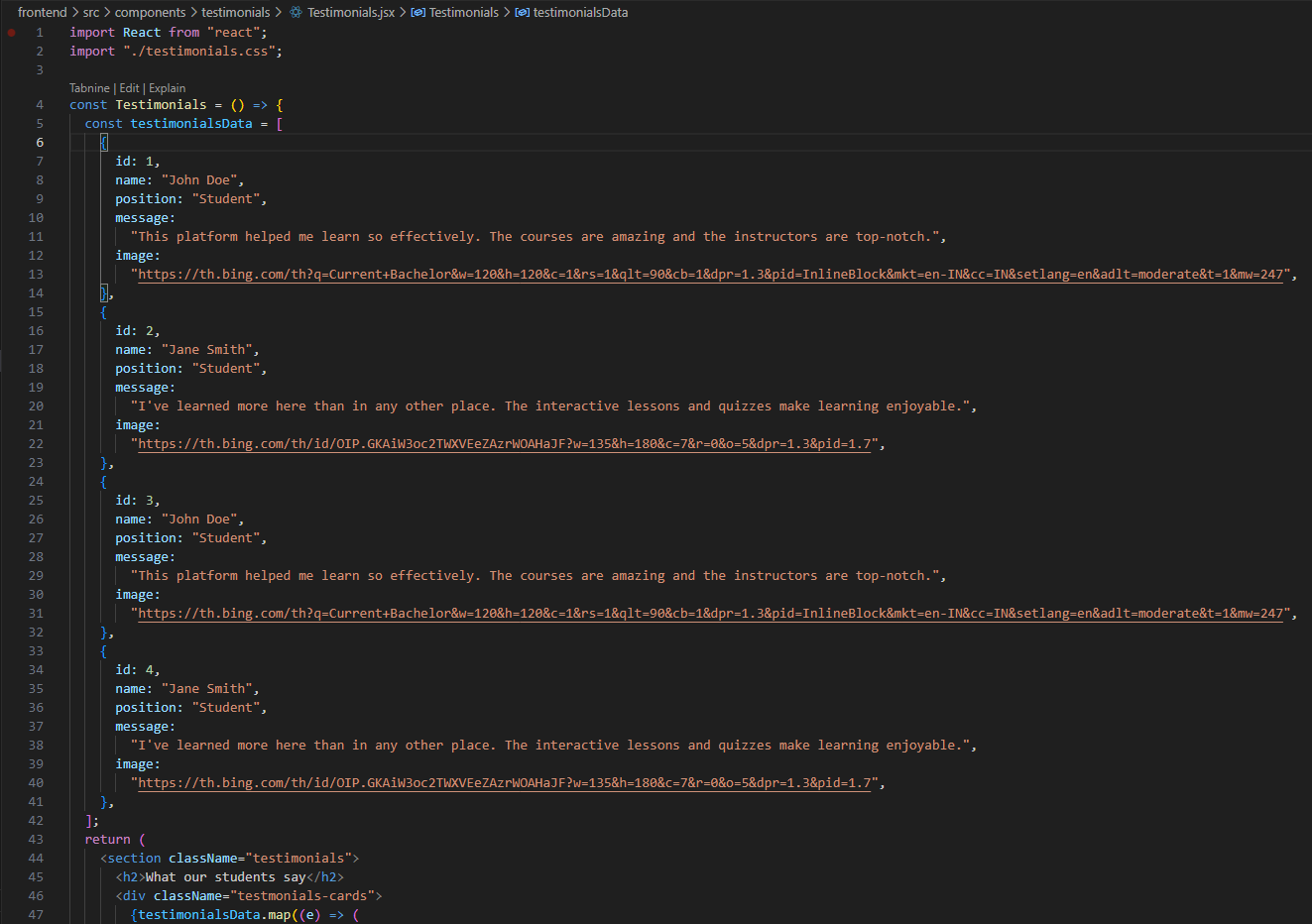
**Create ResetPassword.jsx :**



**Create auth.css :**



**Create Testimonials.jsx :**



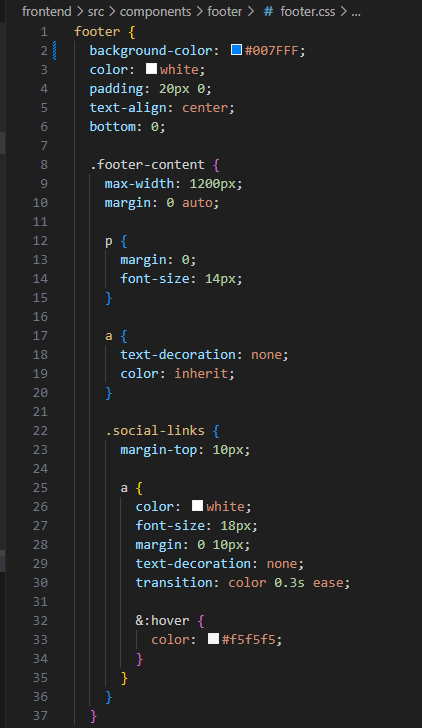
**Create Testimonials.css**



**Create Footer.jsx:**



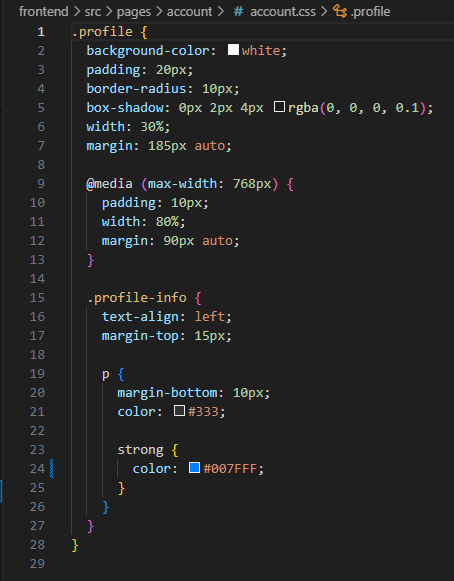
**Create Footer.css :**



**Create Account.jsx :**



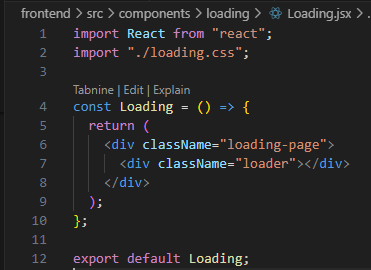
**Create Account.css:**



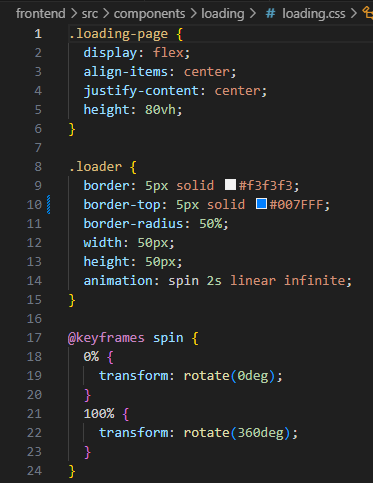
**Create Usercontext.jsx :**



**Create Loading.jsx :**



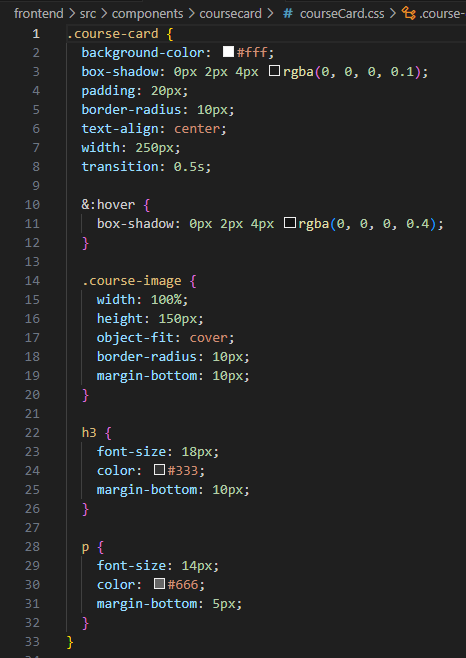
**Create Loading.css :**



**Create Coursecard.jsx :**



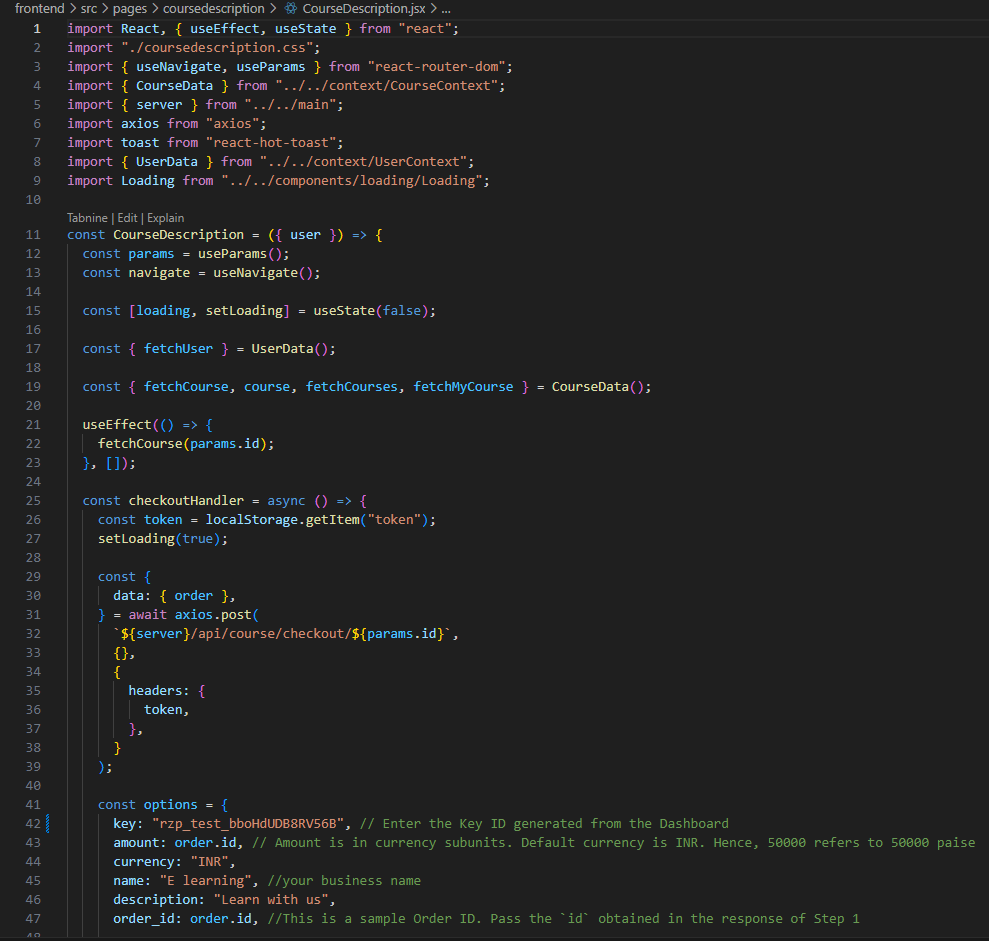
**Create CourseCard.css :**



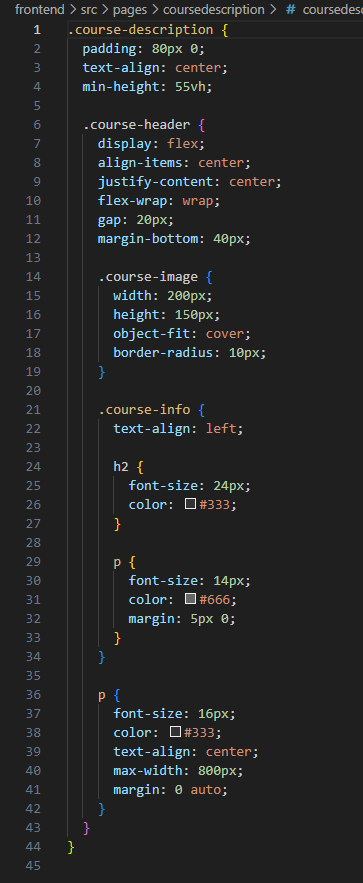
Create CourseContext.jsx :



**Create CourseDescription.jsx :**



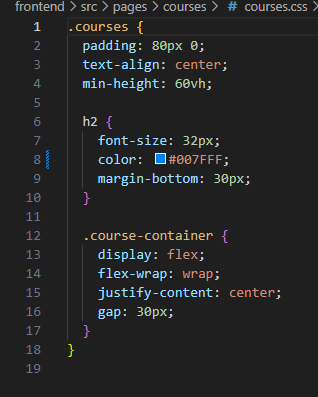
**Create** **CourseDescription.css :**



**Create Courses.jsx :**



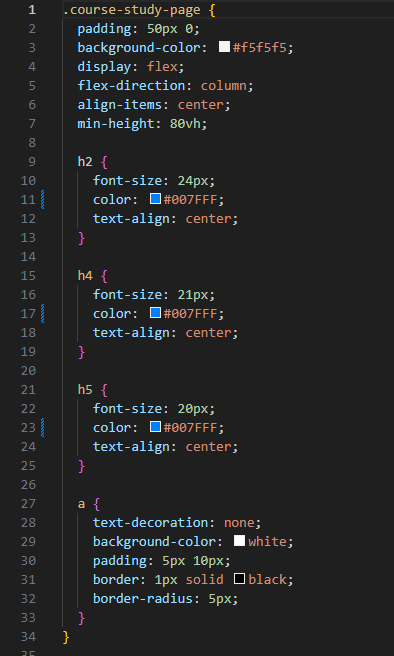
**Create Courses.css :**



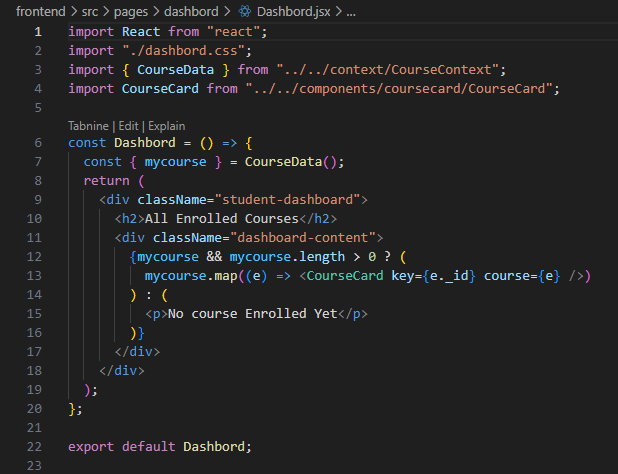
**Create CourseStudy.jsx:**



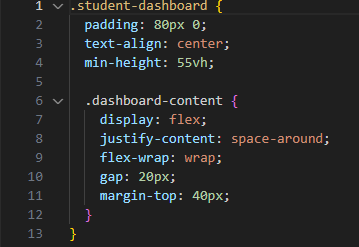
**Create CourseStudy.css :**



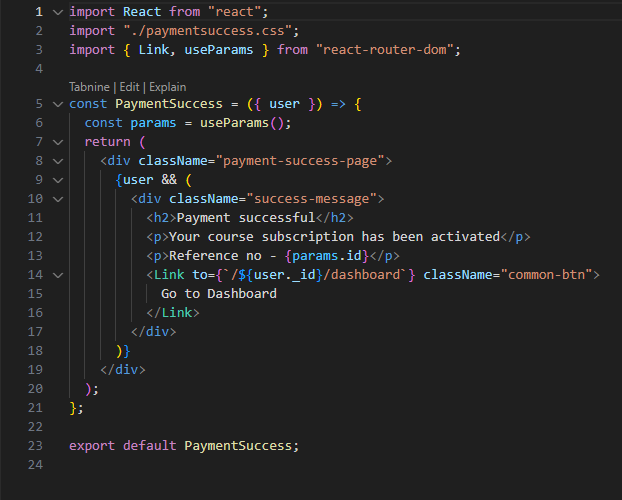
**Create Dashbord.jsx :**



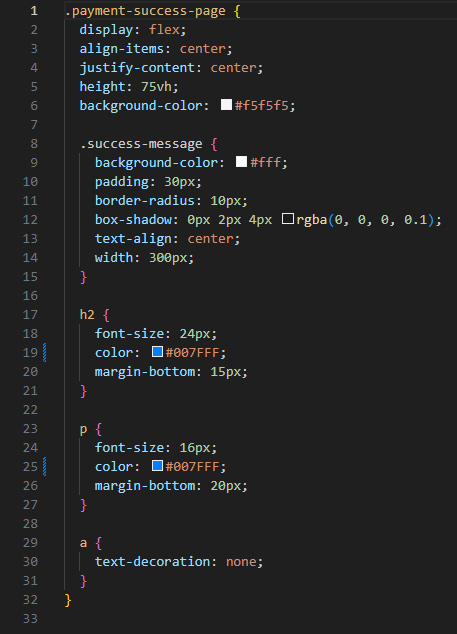
**Create Dashbord.css :**



**Create PaymentSucces.jsx :**



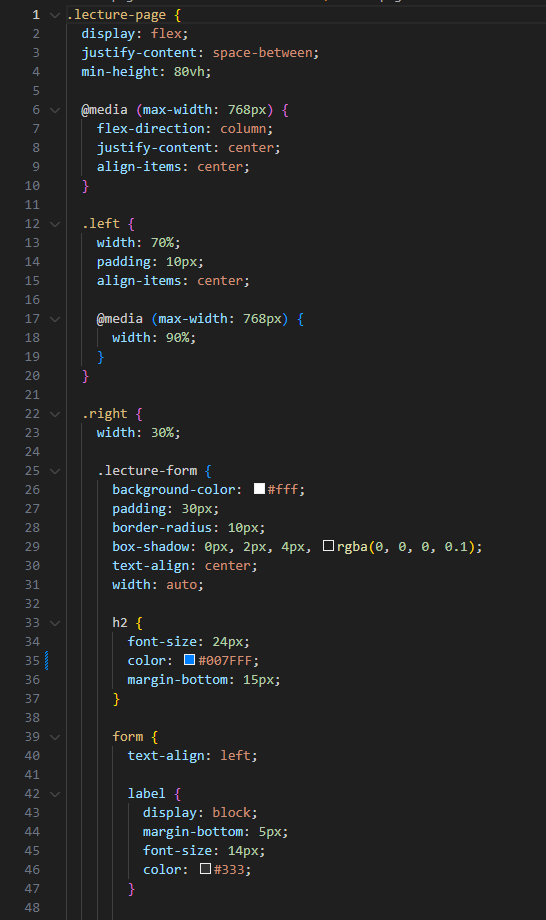
**Create PaymentSucces.css:**



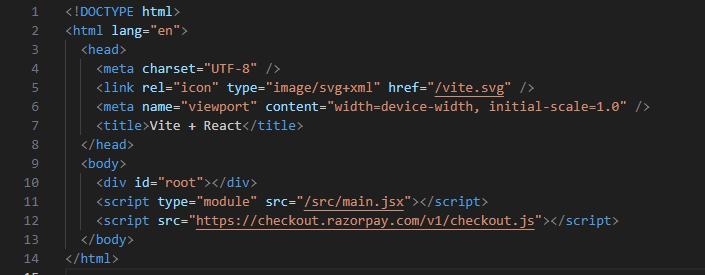
**Create Lecture.jsx :**



**Create Lecture.css :**



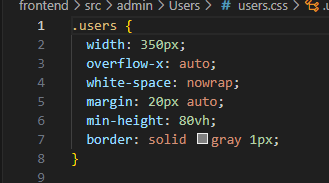
**Create index.html :**

**.**

**Create Adminuser.jsx :**



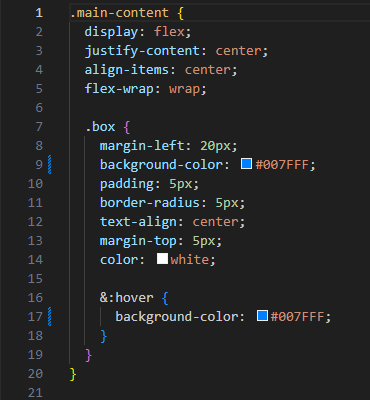
**Create user.css:**



**Create AdimDashbord.jsx :**



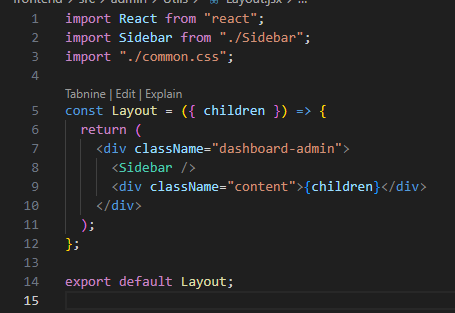
**Create Dashbord.css :**



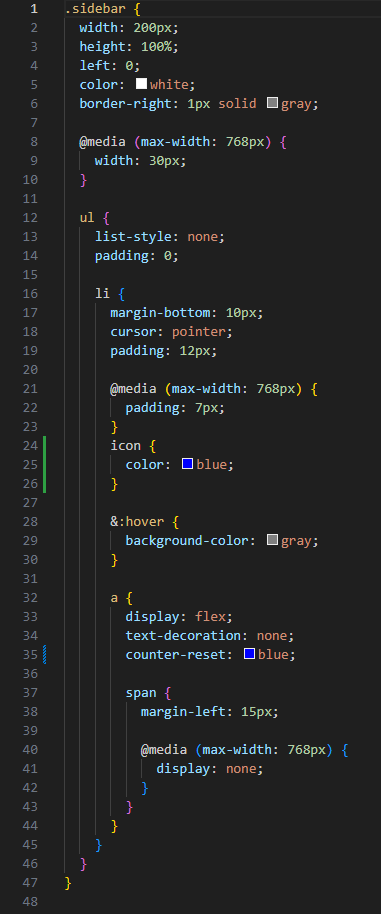
**Create Sidebar.jsx :**



**Create Layout.jsx ;**



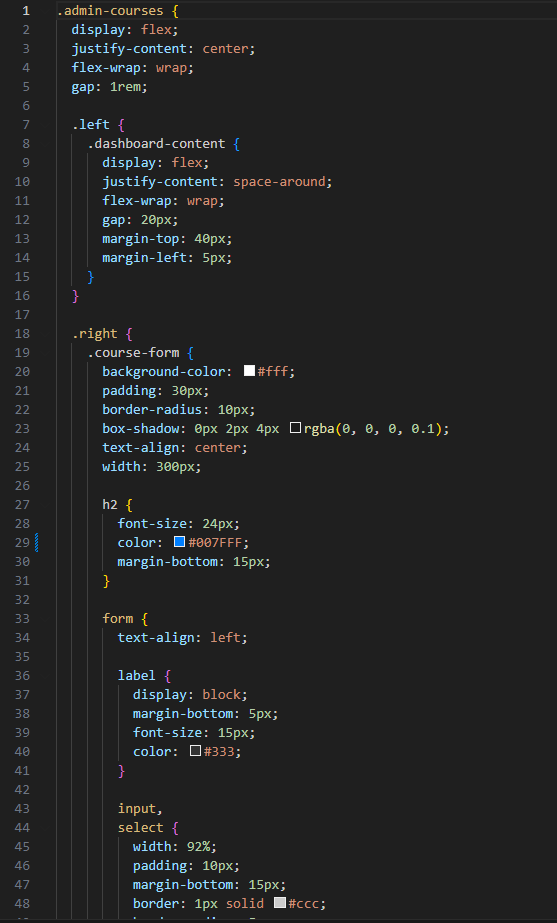
**Create common.css:**



**Create AdminCourses.jsx :**

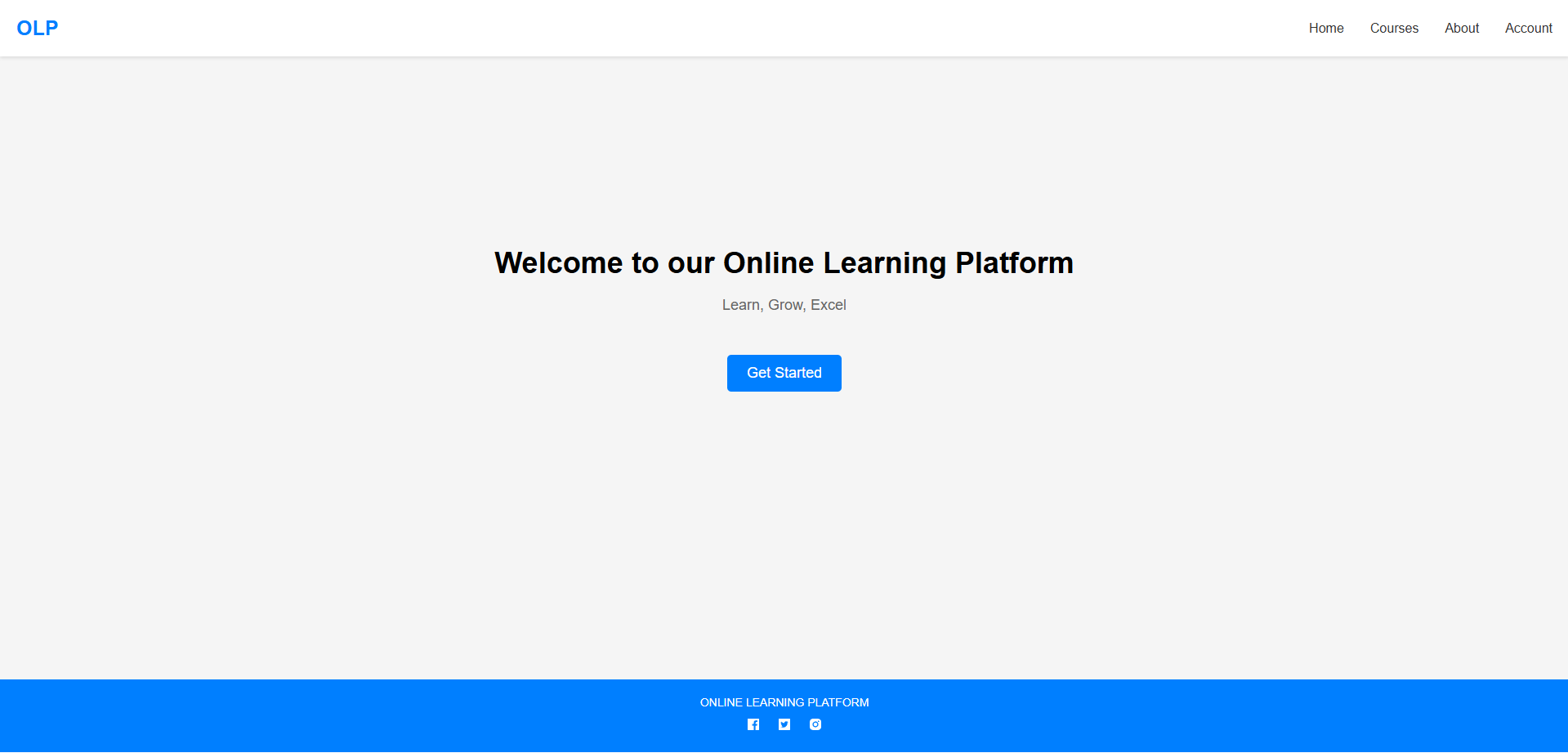


**Create AdminDashbord.css :**

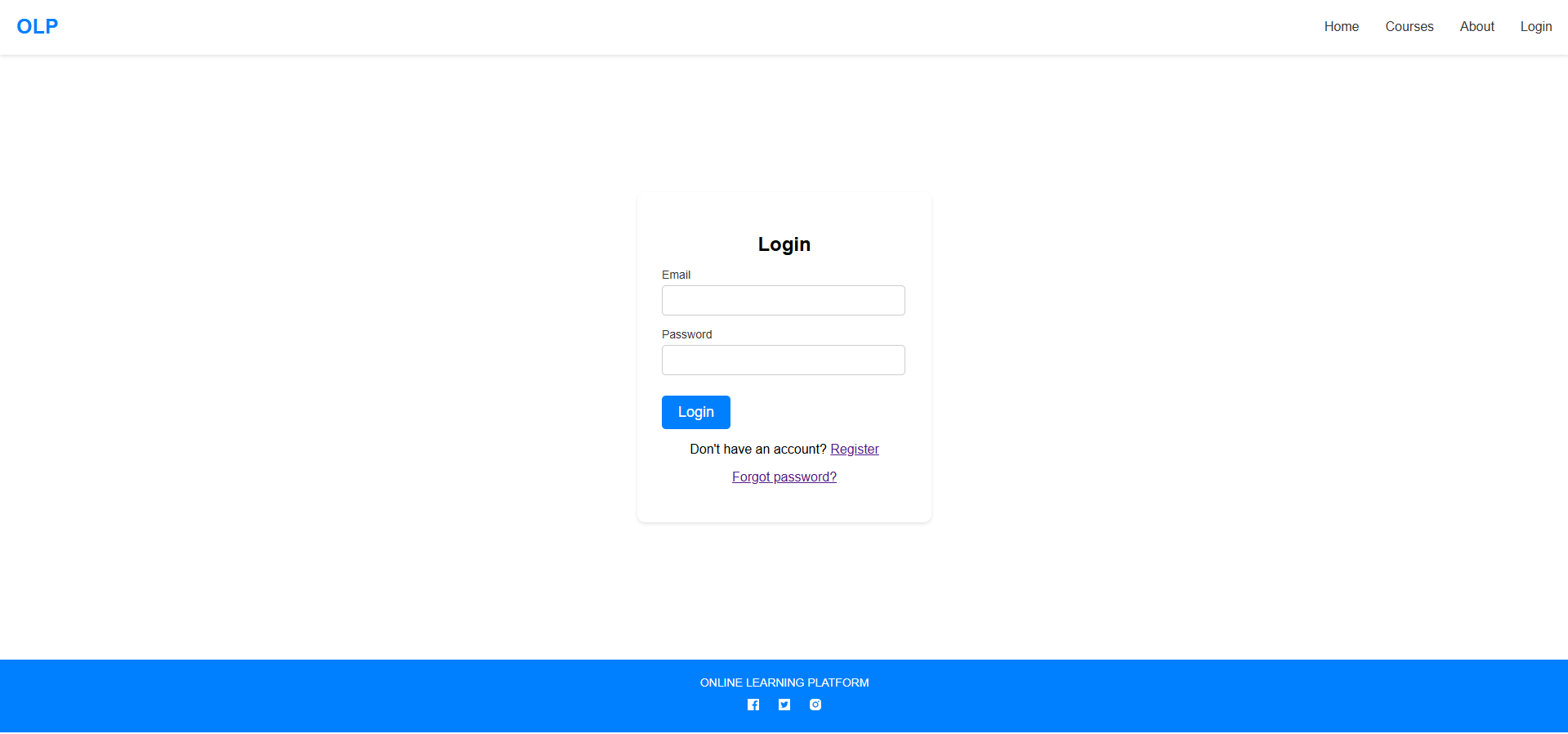


**Demo UI images:**

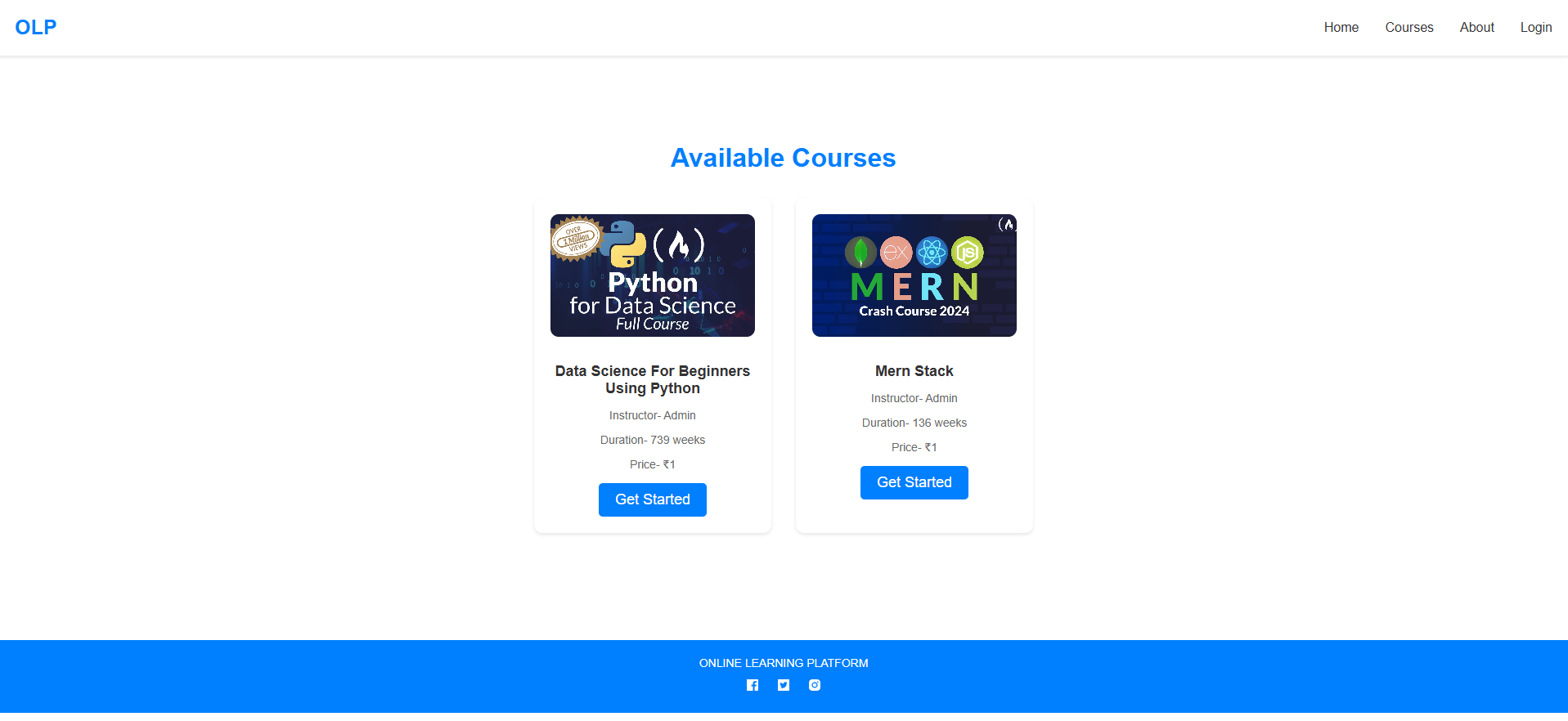
· **Landing page**



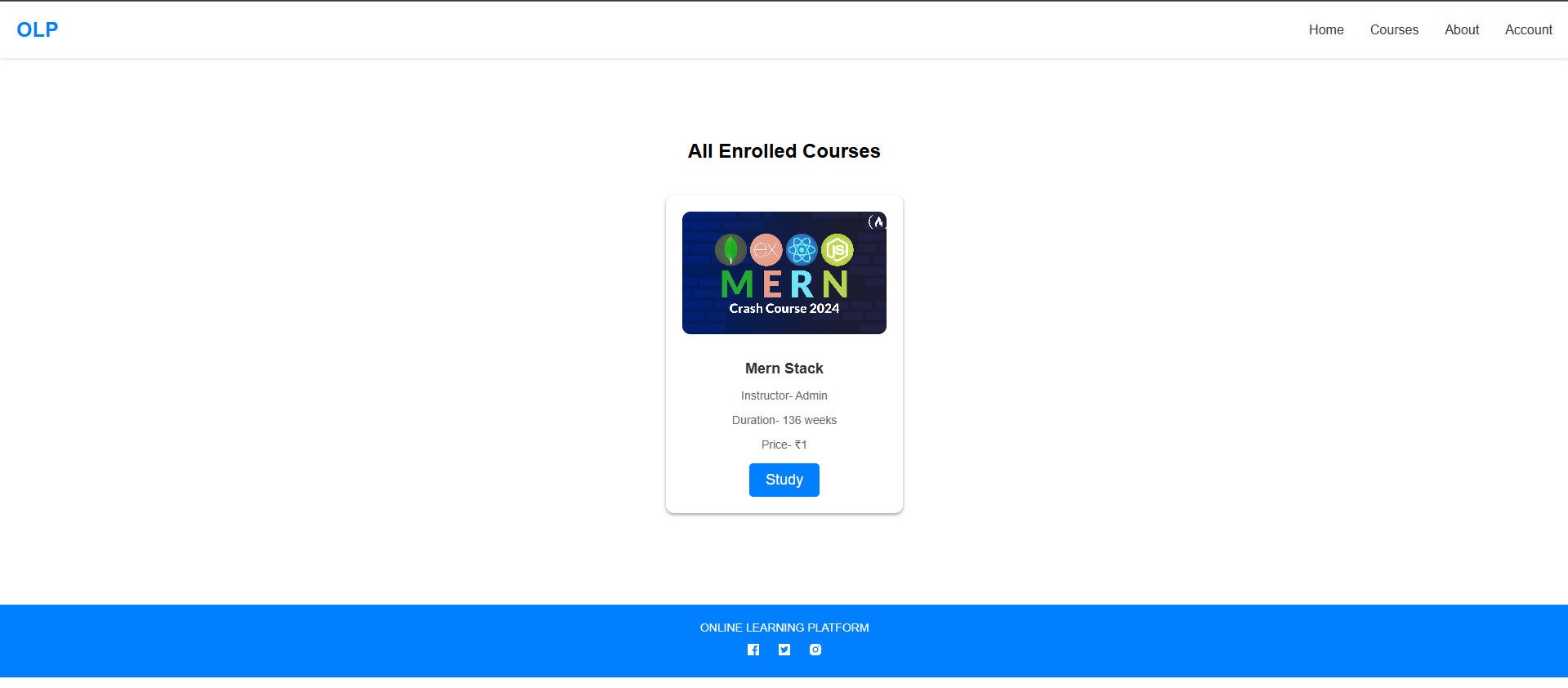
· **Authentication:**



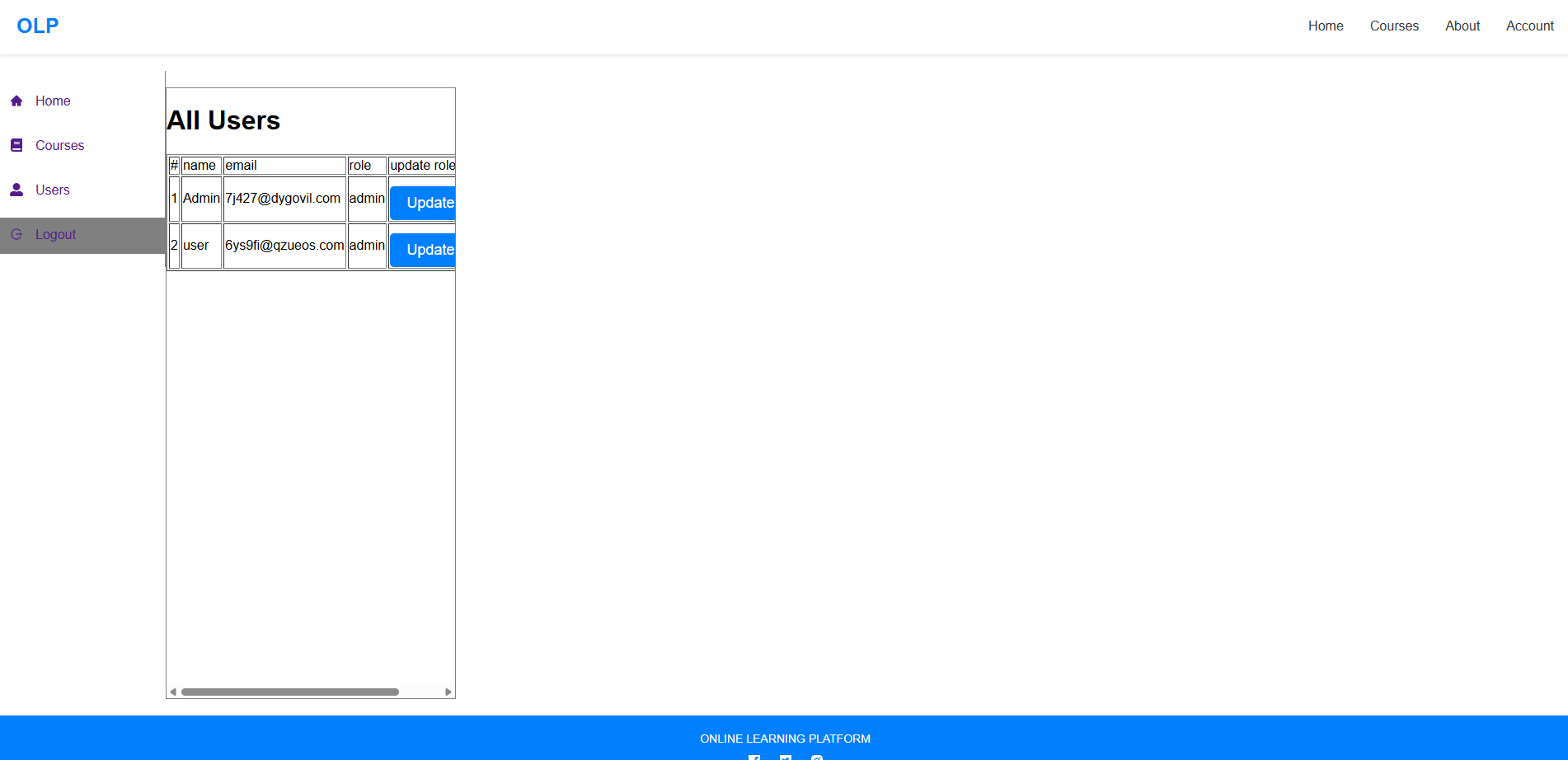
· **Courses :**



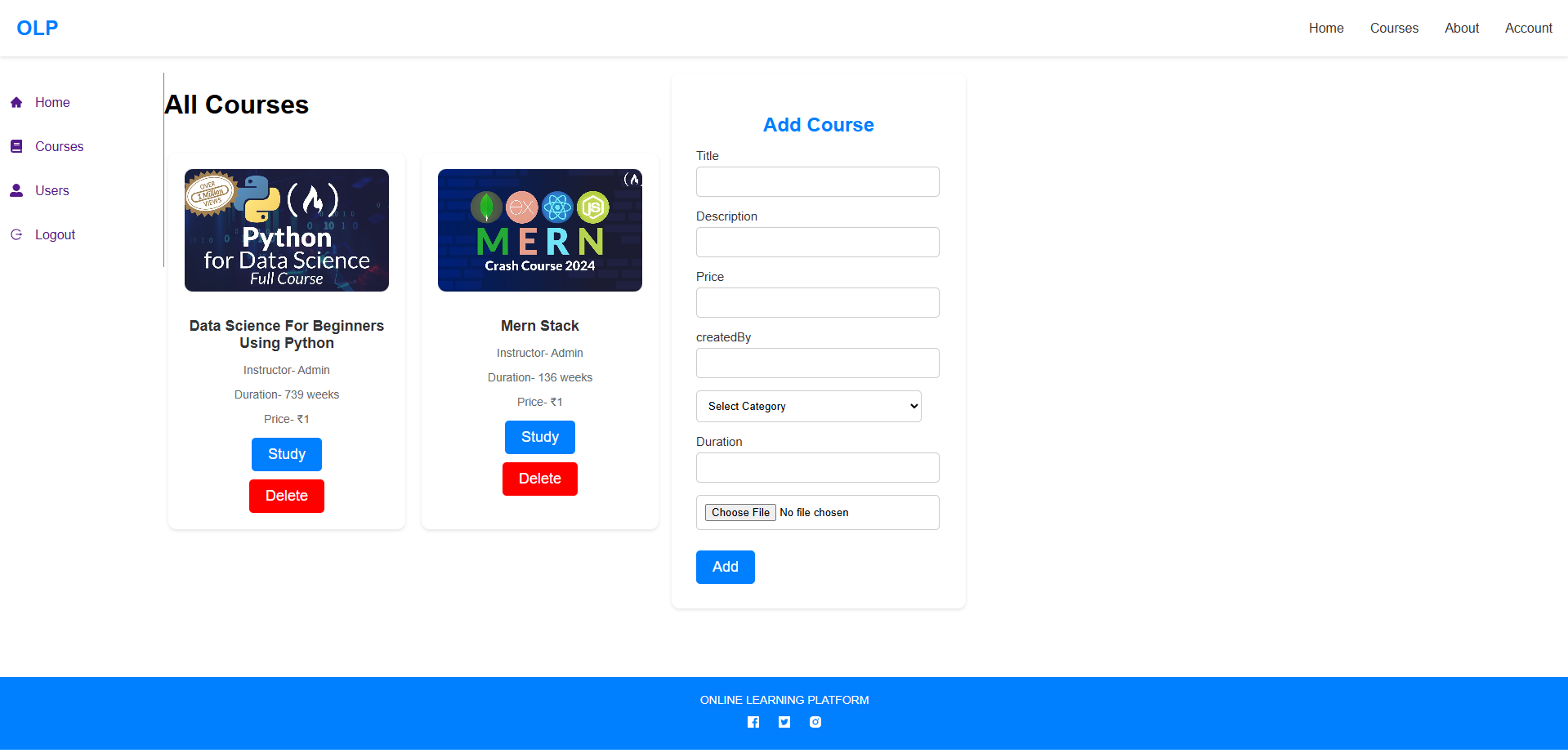
·  **UserDashBoard:**



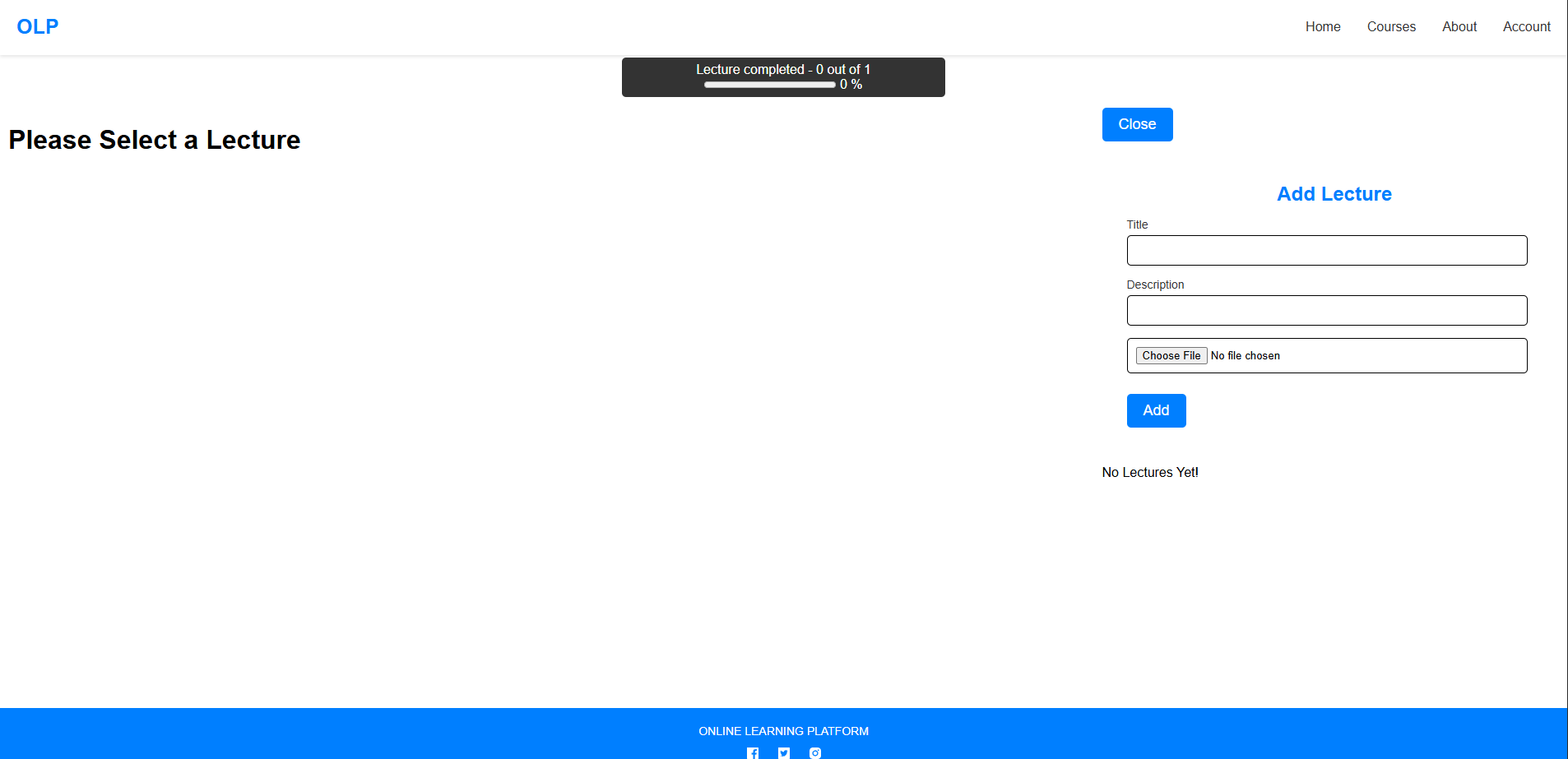
·  **AdminDashBoard:**



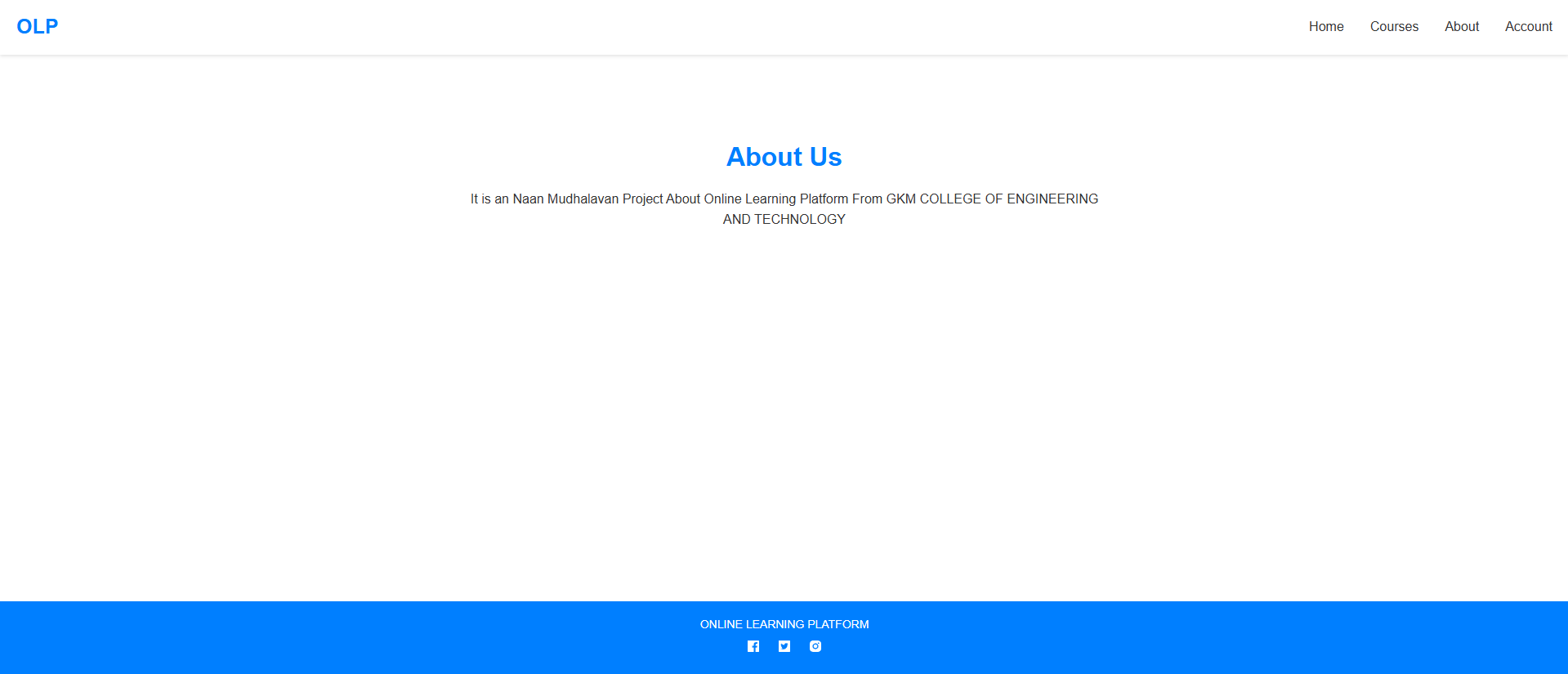
·  **AdminCourses**



·  **AdminLecture :**



·  **Aboutpage :**



**Conclusion:**

The Online Learning Platform, built using the MERN stack, provides a comprehensive solution for digital education, empowering both students and instructors with essential tools for effective learning and teaching. With features like secure user authentication, real-time notifications, payment integration, and an interactive learning environment, the platform is designed to deliver a seamless and engaging experience.

By leveraging modern technologies like React for a responsive frontend, Express and MongoDB for a robust backend, and integrations with services like Razorpay and Nodemailer, this platform is scalable, secure, and flexible enough to adapt to future requirements. The architecture and modular design make it easy to add new features and improve existing ones, ensuring it can evolve alongside user needs and the education landscape.