LearnHub: Your Center for Skill Enhancement

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 decides 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.

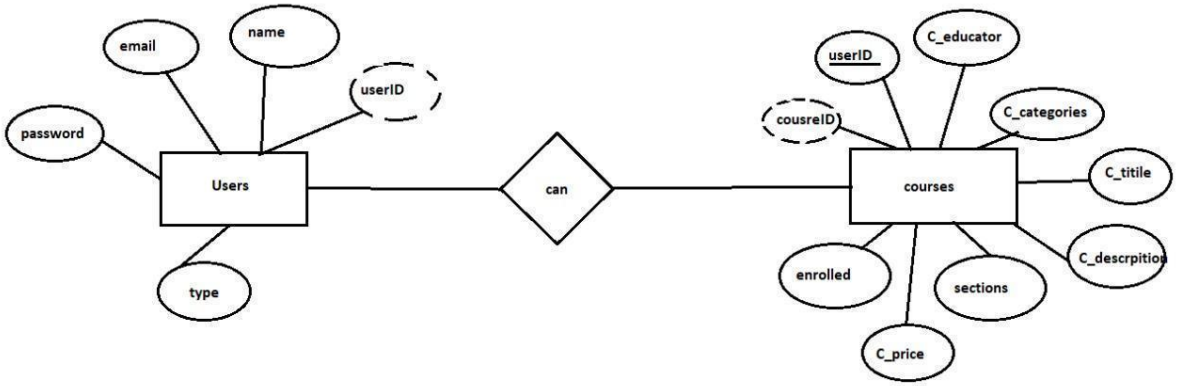
The frontend utilizes the bootstrap and material UI library to establish real-time and better UI experience for any user.

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 is 2 collections namely users, courses which 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, 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/

?**Front-end Framework**: Utilize Reactjs to build the user-facing part of the application, including entering booking room, status of the booking, and user interfaces for the admin dashboard.
For making better UI we have also used some libraries like material UI and boostrap.

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:

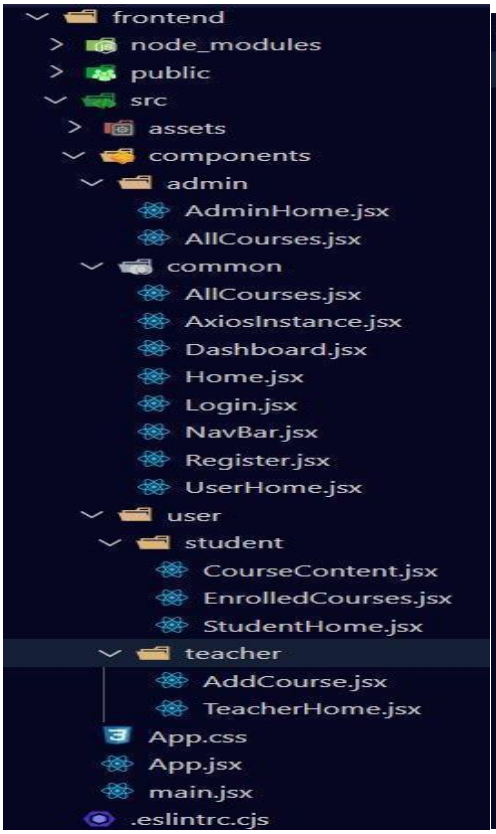
- To start the development server, execute the following command:
  npm start
- The OLP app will be accessible at http://localhost:5172

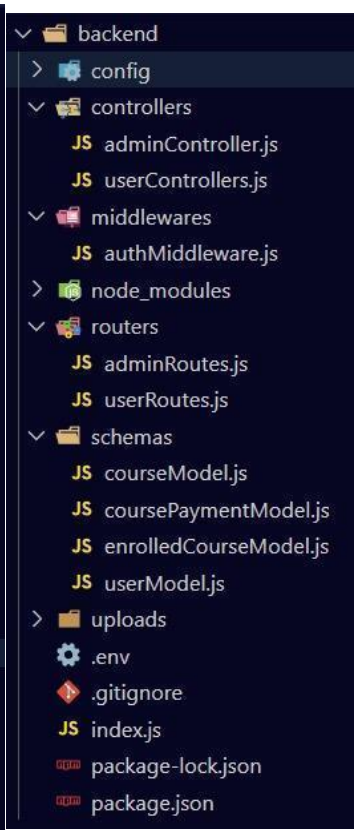
You have successfully installed and set up the Online leraning app on your local machine. You can now proceed with further customization, development, and testing as needed.

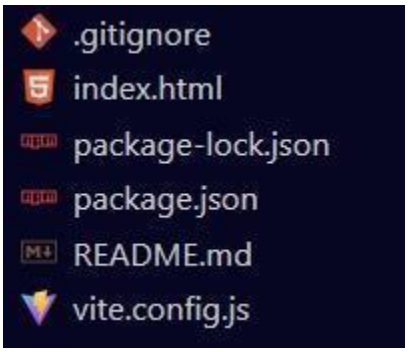
Project Structure

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







-
    - Project Flow

Application Flow

The project has a user called– teacher and student and other will be Admin which takes care of all the user. 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 any other reasons.
3. Also add sections to courses.

Student:

4. Can enroll in an individual or multiple course.
5. Can start the course where it has stopped.
6. Once the course is completed, they can download their certificate of completeion of the course.
7. For 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 course that are present in the app.
10. Watch out all kind of users in app.
11. Record all the enrolled all the student that are enrolled in course.

Project Setup And Configuration

-
    - **Folder setup:**

2. Create frontend and
3. Backend folders

. Open the backend folder to install necessary tools

For backend, we use:

- cors
- bcryptjs
- express
- dotenv
- mongoose
- Multer
- Nodemon
- jsonwebtoken

Backend Development

-
    - **Setup express server**

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

    - **Add authentication:** for this,

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

Database Development

-
    - **Configure MongoDB**

2. Import mongoose.
3. Add database connection from config.js file present in config folder.
4. Create a model folder to store all the DB schemas.

Frontend development

-
    - **Installation of required tools:**
    - For frontend, we use:

3. React
4. Bootstrap
5. Material UI
6. Axios
7. Antd
8. mdb-react-ui-kit
9. react-bootstrap

Project Implementation & Execution

On completing the development part, we then run 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