LearnHub: Your Center for Skill Enhancement

INTRODUCTION:

An **Online Learning Platform** built with the **MERN stack** offers a dynamic, efficient, and scalable solution for delivering educational content 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:

The MERN stack consists of:

- **MongoDB**: A NoSQL database used to store course content, user information, progress, quizzes, and more.
- **Express.js**: A lightweight backend web framework for Node.js that manages API endpoints and handles server-side logic.
- **React.js**: A powerful frontend library that enables dynamic and responsive user interfaces for students and instructors.
- **Node.js**: A runtime environment that executes JavaScript code on the server side, allowing for full-stack JavaScript development.

Purpose of the Platform:

The goal is to create a seamless digital environment where:

- **Students** can register, browse courses, watch video lectures, take quizzes, and track their progress.
- **Teachers** can create and manage courses, upload materials, and interact with learners.
- **Admins** can manage users, content, and platform analytics.

Core Features:

1. **User Authentication & Authorization**
   - Sign up, login, role-based access (student/instructor/admin)
2. **Course Management**
   - Create, edit, delete courses (title, description, video content, PDFs, etc.)
3. **Content Delivery**
   - Video streaming, downloadable materials, structured modules
4. **Progress Tracking**
   - Quiz results, course completion tracking, certificates

5. **Interactive UI**
    - Responsive design using React for a smooth user experience
6. **API Integration**
    - RESTful APIs with Express and Node.js for frontend-backend communication
7. **Database Operations**
    - MongoDB stores user data, course content, and analytics

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.

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:

Srimukhi, 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, Srimukhi 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:

Srimukhi clicks on the course and reads the course description, instructor details, and syllabus. Impressed, she decided to enroll in the course.

After enrolling, Srimukhi can access the course materials, including video lectures, reading materials, and assignments.

Interaction and Support:

Throughout the course, Srimukhi engages with interactive elements such as discussion forums and live webinars where she can ask questions and interact with the instructor and other learners.

Paid Courses:

Srimukhi 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, Mike , 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 LEARN HUB platform is designed with a modern, scalable, and secure architecture to ensure seamless user experience and easy maintenance.

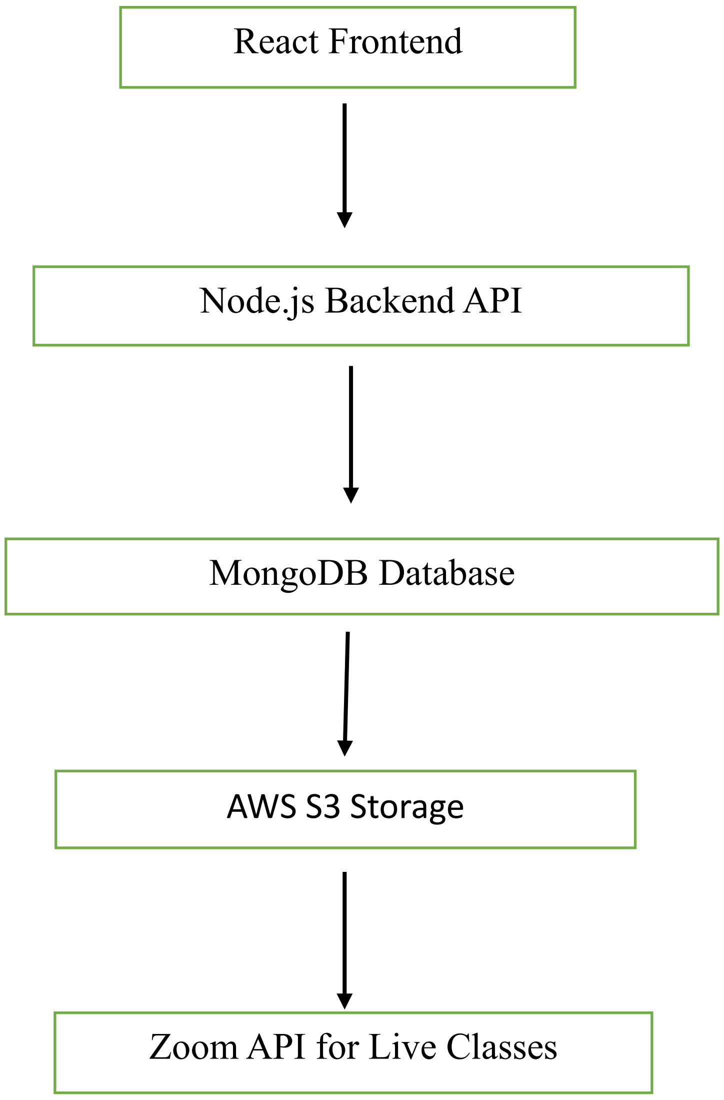
Components:

- **Frontend:**
  Built using **React.js**, providing a responsive and dynamic user interface accessible across devices.

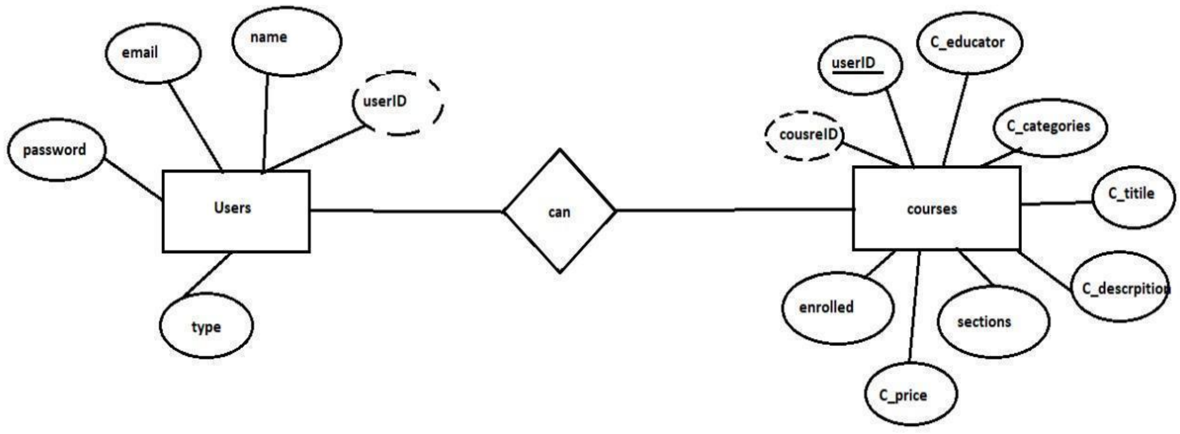
- **Backend:**
  Developed with **Node.js** and **Express.js** to handle API requests, business logic, and integrations.

- **Database:**
  MongoDB stores user data, course content, progress tracking, and mentorship schedules.
- **Authentication:**
  Firebase Authentication secures user login, supports email/password and social logins.
- **File Storage:**
  Course videos, assignments, and user-uploaded files are stored securely on **AWS S3**.

FLOW CHART FOR TECHNICAL ARCHITECTURE



ER Diagram:



Here there are 2 entities named users, courses that have their own fields and shows how **users** interact with **courses.**

Users:

1. userID: acts as primary key
2. name
3. email
4. password
5. type

   This represents the users of the system like students, tecahers, admins.

Courses:

1. userID: act as foreign key
2. courseID: acts as primary key
3. C_educator
4. C_categories
5. C_title
6. C_description
7. sections
8. C_price
9. Enrolled

   This contains all course related data like who teaches the course, content, how many users are enrolled, and its price.

Relationship:

- **can**
    - This connects **Users** with **Courses**, showing that a user *can* interact with one or more courses.

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 tool that helps 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.

Install Vite using the following command:

npm create vite@latest

✓Node.js and npm:

Node.js is a powerful JavaScript runtime environment. It allows you to run JavaScript code on the server side. npm (Node Package Manager) comes with it, can be used to install backend packages.

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/

To initialize the project and create the package. json file use the following command:

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, and server- side logic (middleware support, and modular architecture).

Install Express.js using 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, which makes 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
Install by following the instructions: https://docs.mongodb.com/manual/installation/

✓React.js:

React.js is one of JavaScript library for building responsive user interfaces. It enables developers to create interactive and reusable UI components which makes it easier to build dynamic and responsive web applications.

Install React.js following the installation guide: https://reactjs.org/docs/create-a-new-react-app.html

✓HTML, CSS, and JavaScript:

HTML for creating the frame or structure of the app, CSS for styling the app and JavaScript for the logic and functionality like client-side interactivity.

✓**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:

• Open Command Prompt

• Navigate to project folder and run the following command:

```
cd containment-zone
```

• Install the required dependencies by running the following commands:

```
cd frontend
npm install
cd ../backend
npm install
```

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

```
npm start
```

• The OLP app will be run at: http://localhost:5173

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 project can be divided into **two main parts**:

- **Frontend** – React.js
- **Backend** – Node.js, Express.js, MongoDB

How Each Part Works

Backend:

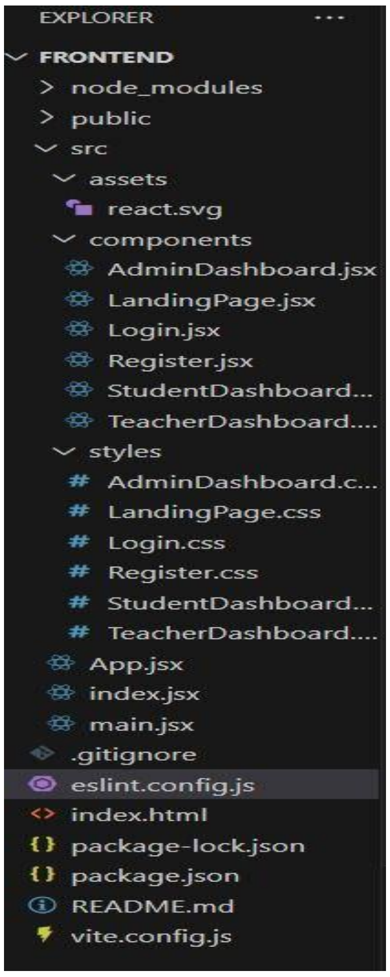
- **server.js** starts the Express server.
- **Routes** handle API endpoints like /api/courses, /api/auth, etc.
- **Controllers** contain the logic for those routes.
- **Models** define your MongoDB data structure.
- **Middleware** is used for JWT validation, protecting routes for students or instructors.

Frontend:

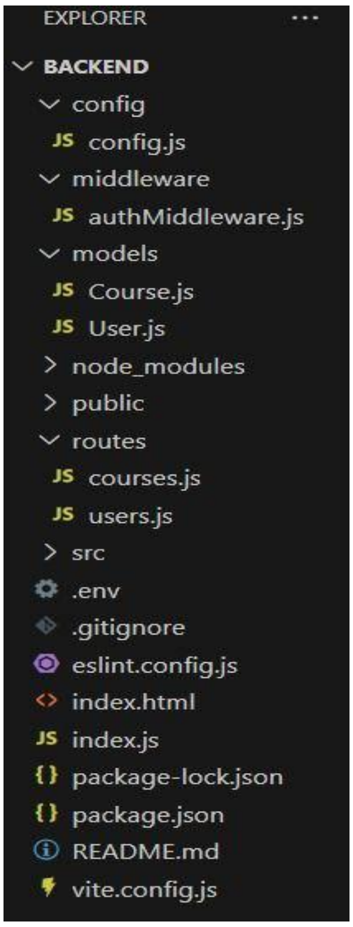
- **React components** build your UI: dashboards, video players, course lists.
- **Pages** define routes using react-router-dom.
- **Context API** can be used for global state (e.g., logged-in user info).
- **API folder** centralizes Axios API requests to the backend.

Communication Flow

1. React frontend sends HTTP requests to backend (Express APIs).
2. Backend handles logic, accesses MongoDB, and sends responses.
3. Authentication via JWT ensures secure login and protected routes.



```
EXPLORER ...
v FRONTEND
  > node_modules
  > public
  v src
    v assets
      react.svg
    v components
      AdminDashboard.jsx
      LandingPage.jsx
      Login.jsx
      Register.jsx
      StudentDashboard...
      TeacherDashboard....
    v styles
      AdminDashboard.c...
      LandingPage.css
      Login.css
      Register.css
      StudentDashboard...
      TeacherDashboard....
    App.jsx
    index.jsx
    main.jsx
  .gitignore
  eslint.config.js
  index.html
  package-lock.json
  package.json
  README.md
  vite.config.js
```



```
EXPLORER ...
v BACKEND
  v config
    config.js
  v middleware
    authMiddleware.js
  v models
    Course.js
    User.js
  > node_modules
  > public
  v routes
    courses.js
    users.js
  > src
  .env
  .gitignore
  eslint.config.js
  index.html
  index.js
  package-lock.json
  package.json
  README.md
  vite.config.js
```

Application Flow:

The project has many users like 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 students to enroll.
2. Also, delete the course if no student enrolled or for any other reasons.
3. Also, add sections to courses if needed.

Student:

4. Can enroll in an individual or multiple courses.
5. Can continue the course from where they stopped.
6. Once the course is completed, they can download the 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. Takes care of all kinds of users in the app.
11. Record all the students that are enrolled in the course.
12. Resolve any issues within the app.

Milestone 1- Setup & configuration:

- **Folder setup:**

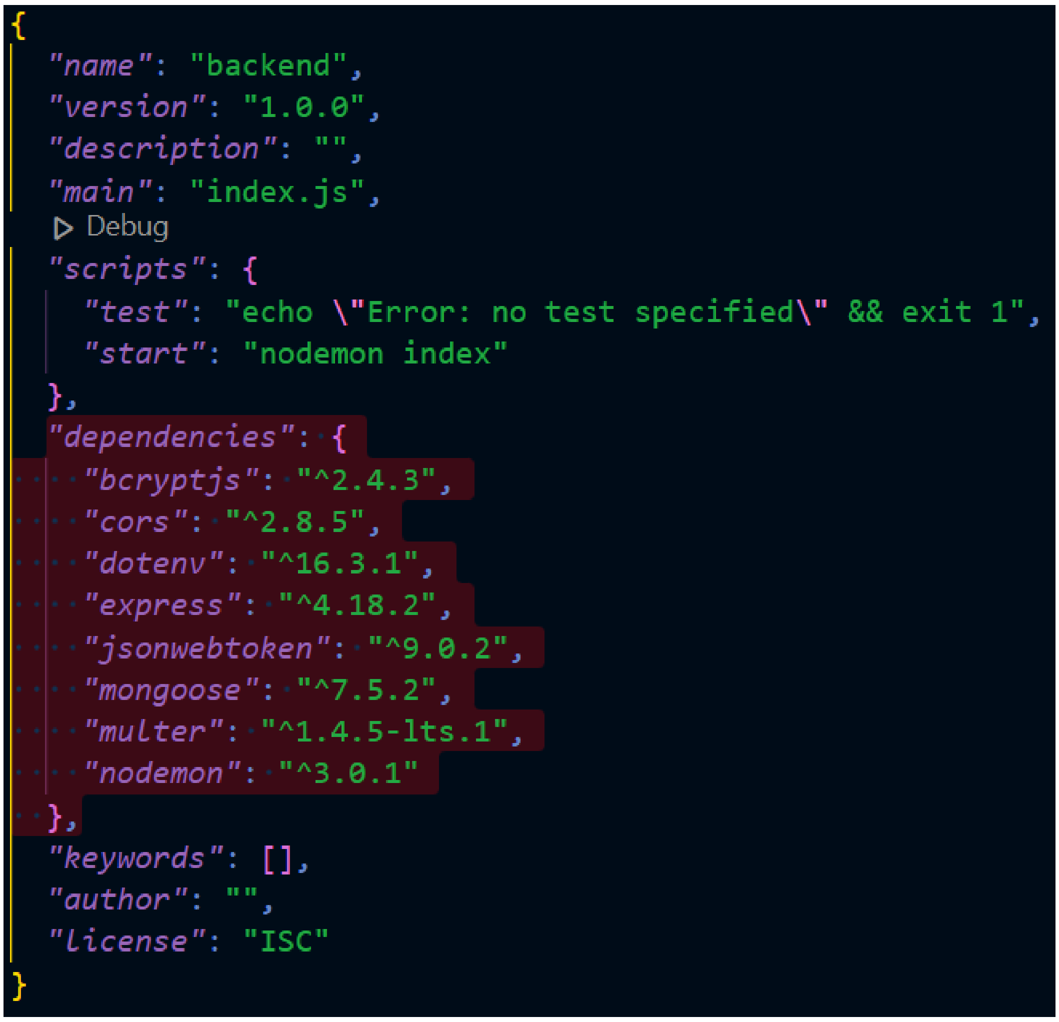
1. Create frontend and
2. Backend folders

Open the backend folder to install the necessary tools

For backend, we use:

- cors
- bcryptjs
- express
- dotenv
- mongoose
- Multer

- Nodemon
- Jsonwebtoken



```json
{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  ▷ Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "nodemon index"
  },
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "cors": "^2.8.5",
    "dotenv": "^16.3.1",
    "express": "^4.18.2",
    "jsonwebtoken": "^9.0.2",
    "mongoose": "^7.5.2",
    "multer": "^1.4.5-lts.1",
    "nodemon": "^3.0.1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

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:

- **Configure MongoDB**

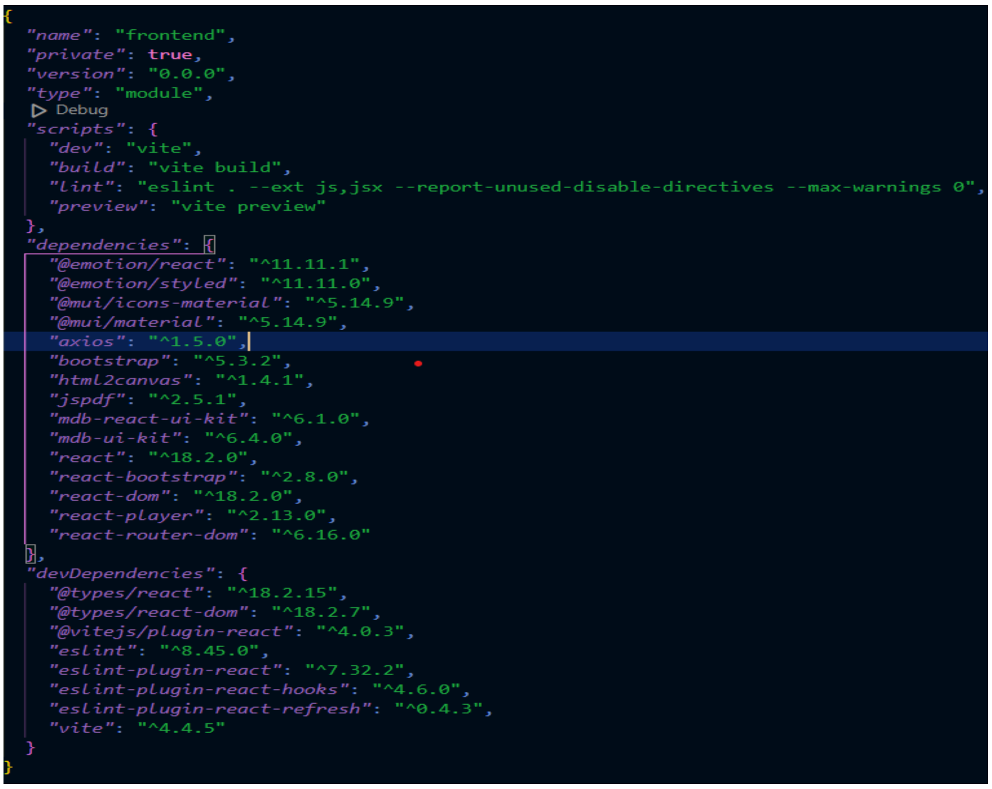
1. Import mongoose.
2. Add database connection from config.js file present in the config folder
3. Create a model folder to store all the DB schemas.

ref: database.mp4

Milestone 4- Frontend Development:

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

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



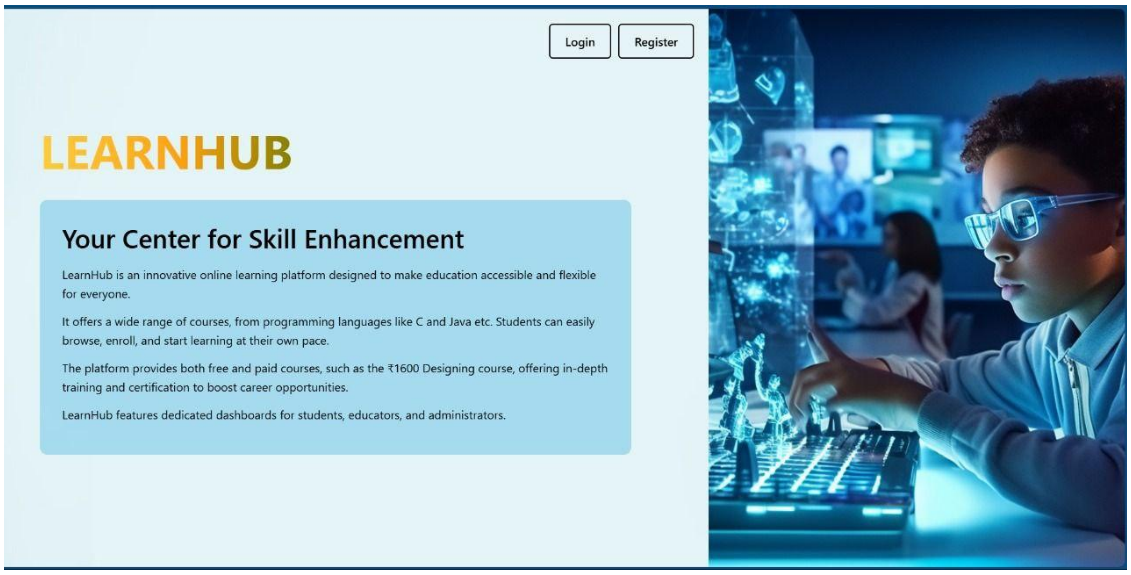
```
{
  "name": "frontend",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  ▷ Debug
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "lint": "eslint . --ext js,jsx --report-unused-disable-directives --max-warnings 0",
    "preview": "vite preview"
  },
  "dependencies": {
    "@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"
  }
}
```

ref: frontend.mp4

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.

Landing page:



Your Center for Skill Enhancement

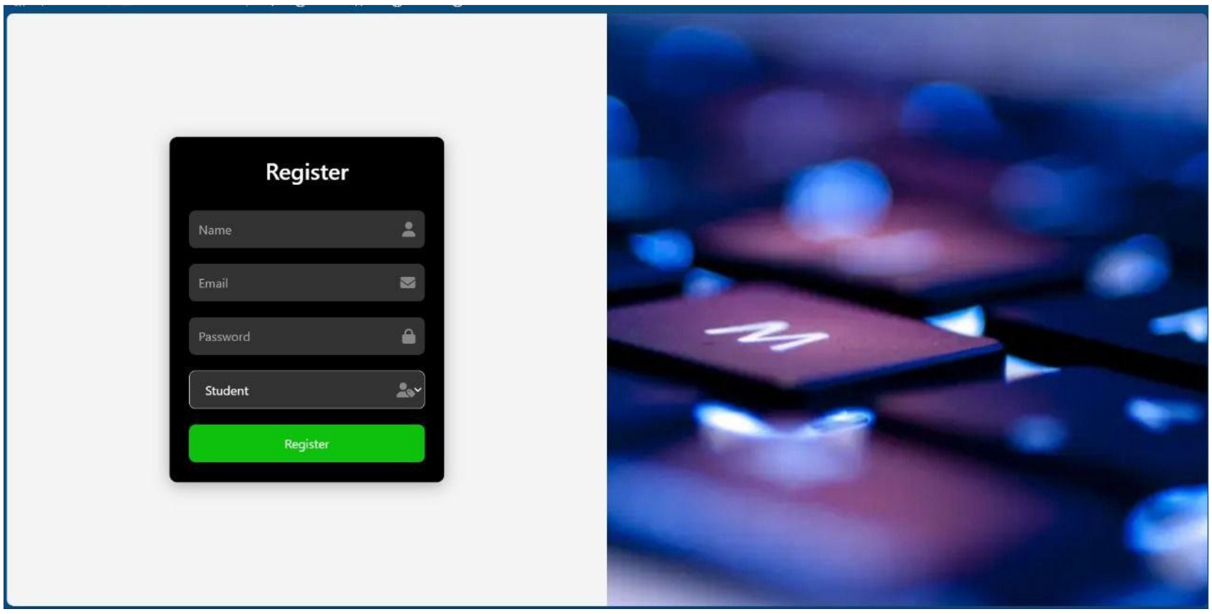
LearnHub is an innovative online learning platform designed to make education accessible and flexible for everyone.

It offers a wide range of courses, from programming languages like C and Java etc. Students can easily browse, enroll, and start learning at their own pace.

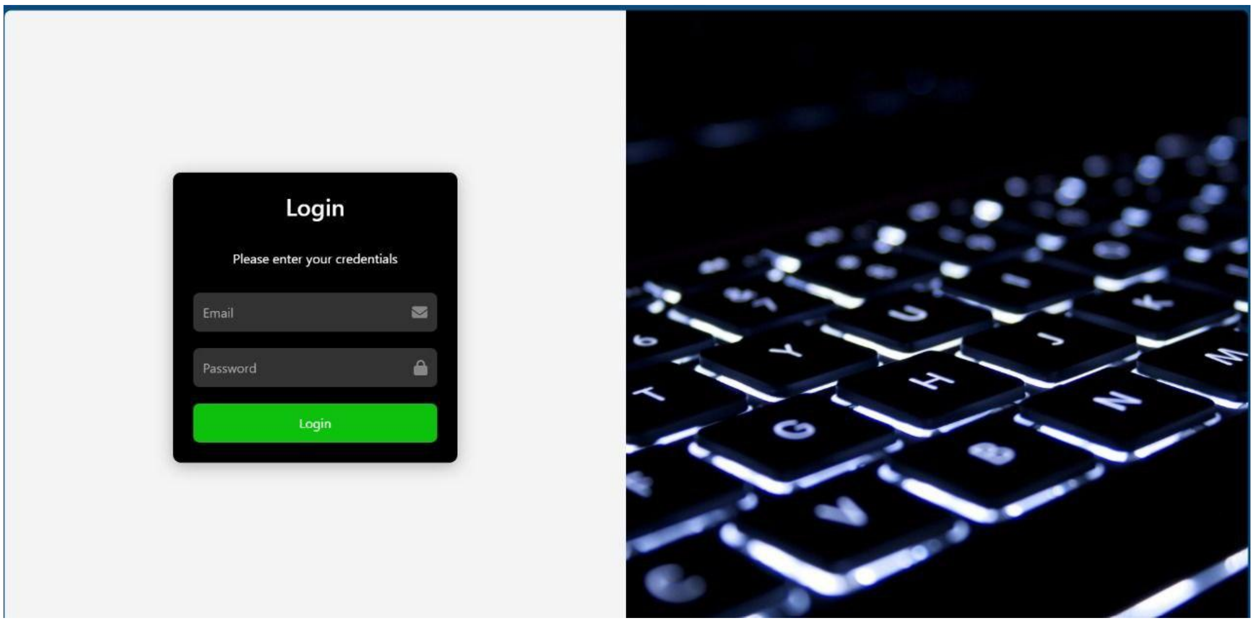
The platform provides both free and paid courses, such as the ₹1600 Designing course, offering in-depth training and certification to boost career opportunities.

LearnHub features dedicated dashboards for students, educators, and administrators.

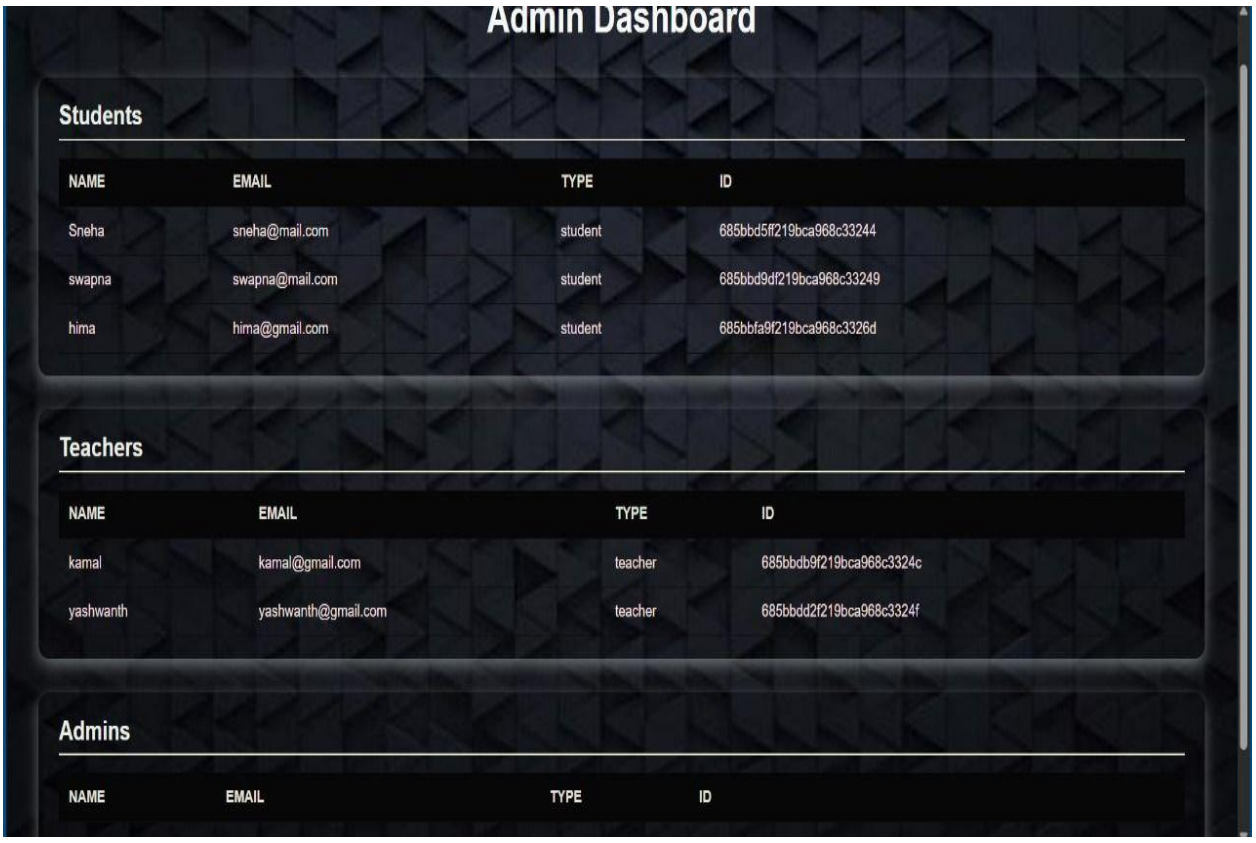
Register page:



Login page:



Admin Dashboard:



Admin Dashboard

Students

NAME	EMAIL	TYPE	ID
Sneha	sneha@mail.com	student	685bbd5ff219bca968c33244
swapna	swapna@mail.com	student	685bbd9df219bca968c33249
hima	hima@gmail.com	student	685bbfa9f219bca968c3326d

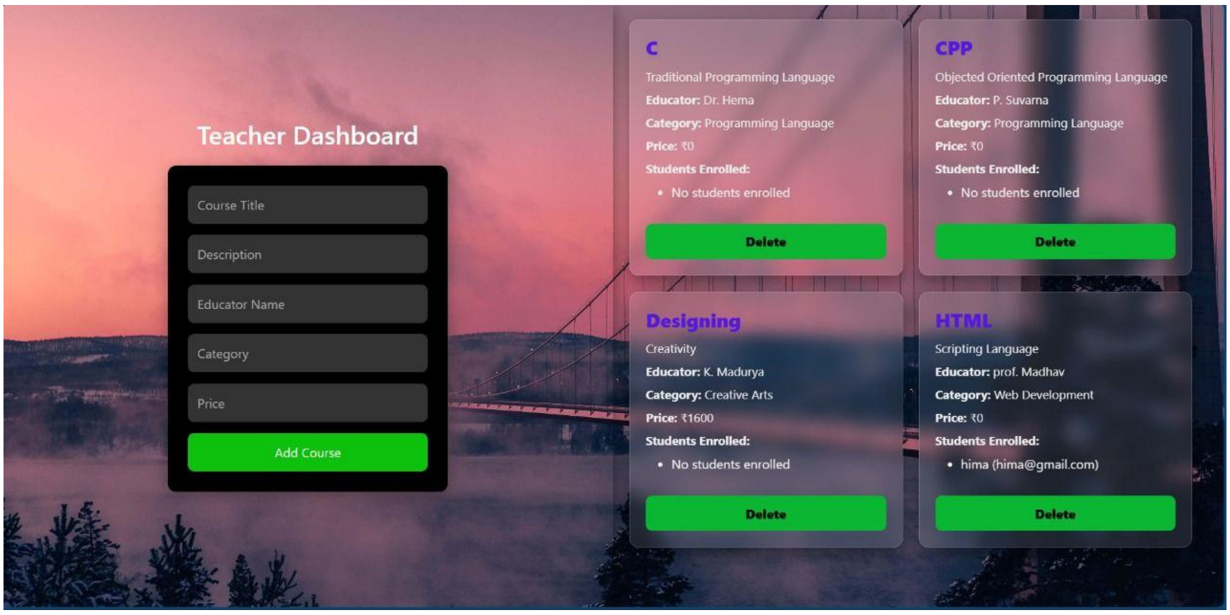
Teachers

NAME	EMAIL	TYPE	ID
kamal	kamal@gmail.com	teacher	685bbdb9f219bca968c3324c
yashwanth	yashwanth@gmail.com	teacher	685bbdd2f219bca968c3324f

Admins

NAME	EMAIL	TYPE	ID

Teacher Dashboard:


Teacher Dashboard

Course Title
Description
Educator Name
Category
Price
Add Course

C
Traditional Programming Language
Educator: Dr. Hema
Category: Programming Language
Price: ₹0
Students Enrolled:
- No students enrolled

Delete

CPP
Objected Oriented Programming Language
Educator: P. Suvarna
Category: Programming Language
Price: ₹0
Students Enrolled:
- No students enrolled

Delete

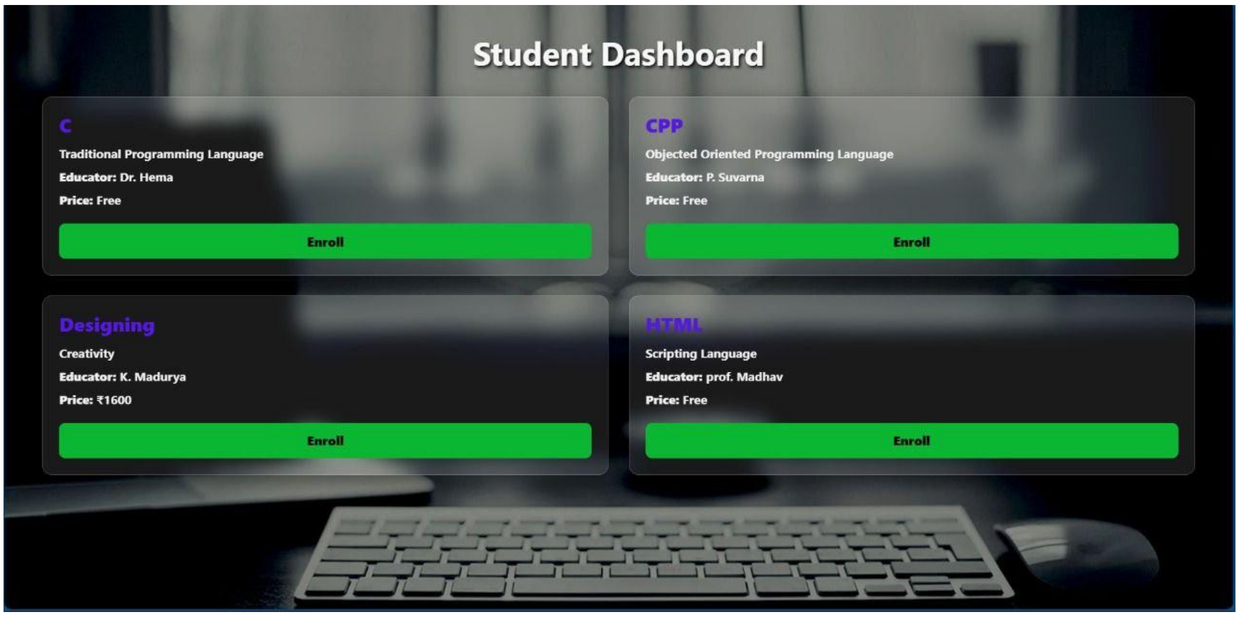
Designing
Creativity
Educator: K. Madurya
Category: Creative Arts
Price: ₹1600
Students Enrolled:
- No students enrolled

Delete

HTML
Scripting Language
Educator: prof. Madhav
Category: Web Development
Price: ₹0
Students Enrolled:
- hima (hima@gmail.com)

Delete

Student Dashboard:


Student Dashboard

C
Traditional Programming Language
Educator: Dr. Hema
Price: Free

Enroll

CPP
Objected Oriented Programming Language
Educator: P. Suvarna
Price: Free

Enroll

Designing
Creativity
Educator: K. Madurya
Price: ₹1600

Enroll

HTML
Scripting Language
Educator: prof. Madhav
Price: Free

Enroll

Note: For the code drive, click on link, and for the demo link, click on project-implementation.mp4