1. Introduction

- Project title :

Learn Hub: Your Center for Skill Enhancement

- Team members**:**

Sanapala Karunasri

Sai Sandeep Gajula

Sai Teja Sri

S Sai Bhargav

2. Project Overview

- Purpose:

The purpose of the LearnHub project is to create an easy-to-use online platform that helps individuals enhance their skills through structured learning. The platform provides:

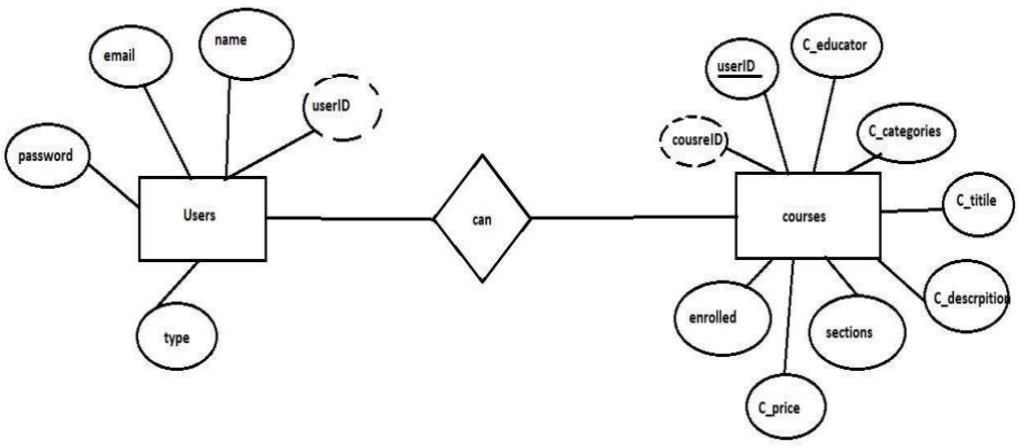
- **Skill-based courses** across various fields such as IT, Finance, and Personal Development.
- **Interactive learning** with videos and quizzes.
- **Progress tracking** and **certificates** to motivate learners and validate their knowledge.
- This project aims to make **quality education accessible to everyone**—students, job seekers, and professionals—so they can grow personally and professionally at their own pace.

- Features & Functionalities:
  - User Authentication
  - Secure Sign Up and Login functionality for students.
  - Uses JWT (JSON Web Tokens) to ensure secure session handling.
  - Protects user data with proper access control
  - Course Enrolment & Management
  - Users can browse all available courses.

- Ability to enrol in multiple courses.
- Enrolled courses are shown in a personalized student dashboard
- Video-Based Course Content
- Each course contains embedded video lessons.
- Learners can watch videos at their own pace.
- Courses are divided into sections/modules for structured learning
- Progress Tracking System
- Automatically tracks which videos/modules a user has completed.
- Displays progress bars or badges like “In Progress” or “Completed” on enrolled courses.
- Backend tracks user progress via MongoDB.
- Dynamic Quiz System
- Each course includes course-specific quizzes.
- Multiple-choice questions (MCQs) are dynamically loaded based on course ID.
- Users receive instant feedback and results.
- Certificate Generation
- After completing all modules and the quiz, users receive a downloadable certificate.
- Certificate includes user name, course name, and completion date.
- Generated using html2canvas and downloadable as an image.
- Course Filtering by Category
- Courses are categorized under:
- IT & Software
- Finance & Accounting
- Personal Development
- Users can easily filter and explore based on interest.
- Admin Controls (Optional)
- Admins can add new courses, videos, and quiz questions.

- Backend endpoints to manage course data efficiently.
- Tech Features
- Frontend**:** Built using React.js, React-Bootstrap, and MUI for responsive and attractive UI.
- Backend: Developed with Node.js + Express.js.
- Database**:** Uses MongoDB for storing users, course data, and quiz results.
- Video Player**:** Integrated using ReactPlayer for smooth video streaming.
- Certificate Generator**:** Uses html2canvas to capture and generate certificates.

3. Architecture



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

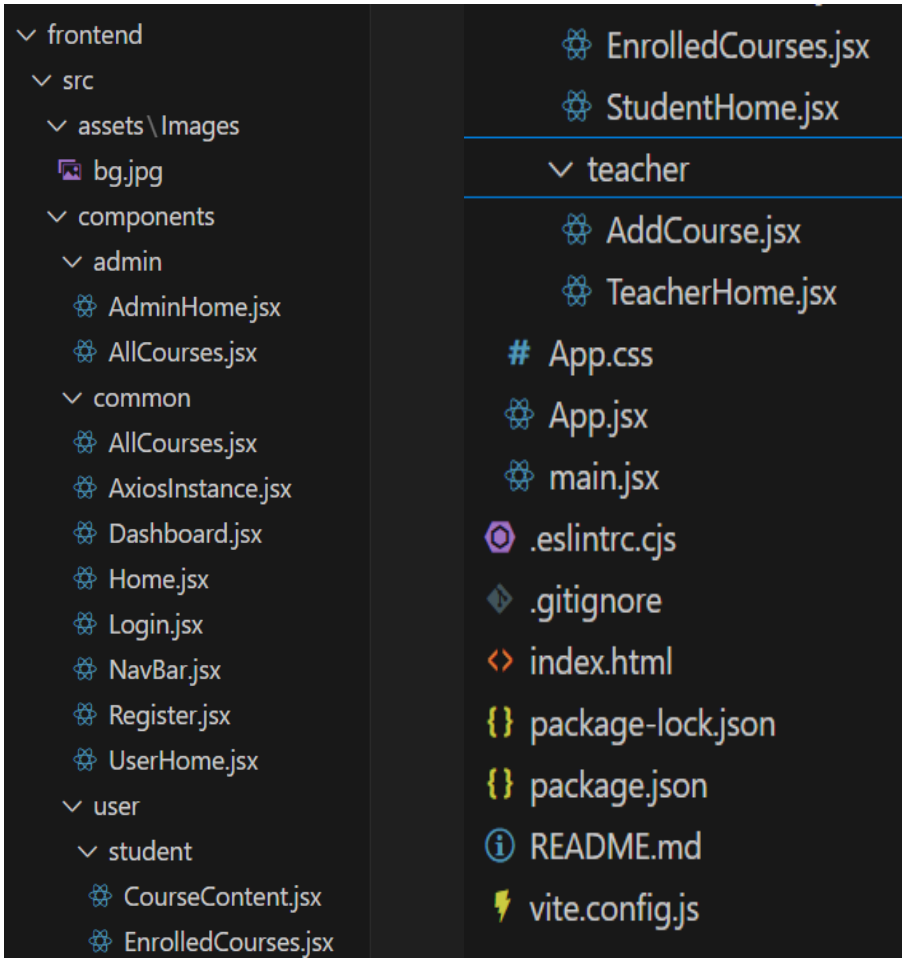
Frontend:

The frontend of LearnHub is built using React.js, a powerful JavaScript library for building interactive user interfaces. The architecture follows a component-based design pattern, promoting reusability, scalability, and maintainability.

🔧 Key Technologies Used

- **React.js** – Core library for UI development.
- **React Router DOM** – For routing/navigation between pages.
- **Axios** – For making HTTP requests to the backend.
- **React-Bootstrap & MUI** – For UI styling and responsive components.
- **ReactPlayer** – For embedding and playing course videos.
- **html2canvas** – For generating downloadable certificates.

📁 Frontend Structure :



❖Backend:

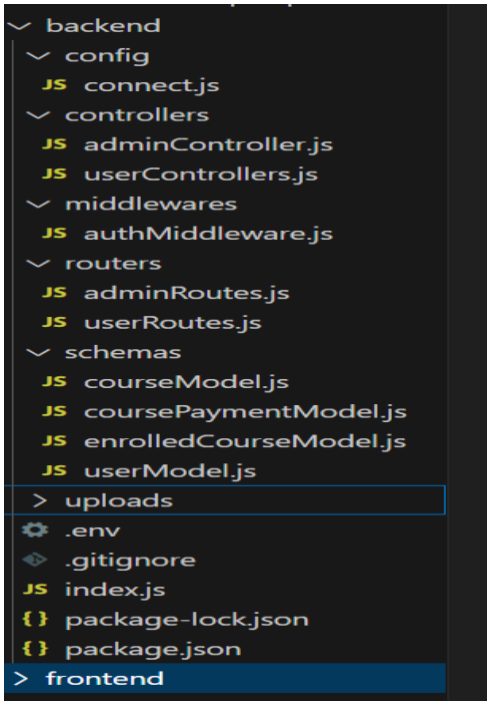
The backend of LearnHub is built using Node.js and Express.js, providing a robust and scalable RESTful API that handles user management, course handling, quiz logic, progress tracking, and certificate generation.

🔧 Key Technologies Used

- Node.js – JavaScript runtime for server-side development.
- Express.js – Web application framework for building APIs.

- MongoDB – NoSQL database for storing data (connected using Mongoose).
- JWT (jsonwebtoken) – For secure user authentication.
- Mongoose – ODM to interact with MongoDB in an object-oriented way.
- html2canvas (on frontend) – Used in coordination with backend logic to generate certificates.

📁 Backend Folder Structure



❖Database:

The LearnHub platform uses MongoDB as its primary database to store structured data for users, courses, quizzes, and progress tracking. All data interactions are performed using Mongoose, an Object Data Modeling (ODM) library for MongoDB and Node.js.

Technologies Used

- **MongoDB** – NoSQL database (stored in JSON-like documents).
- **Mongoose** – ODM to model and interact with MongoDB.

- **MongoDB Compass** – For local database visualization and testing.

4. Setup Instructions

- **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.

5. Running the Application

To run your full-stack application locally, follow these steps:

Frontend (React + Vite)

If you are using Vite (which you mentioned before), the correct command is:

npm run dev

Steps:

1. Open terminal.

2. Navigate to the frontend directory:
cd client
3. Start the frontend:
npm run dev

Backend (Node.js + Express)

npm start

Steps:

1. Open another terminal window or tab.

2. Navigate to the backend directory:

cd server

3. Start the backend:
npm start

If you're using nodemon for auto-restart, you can use:

npm run dev(based on this package.json script):

json

```
"scripts": {
  "start": "node index.js",
  "dev": "nodemon index.js"
}
```

Access

*Frontend:http://localhost:5173 (default Vite port)
*Backend API: http://localhost:5000 (or whatever you configured)

6. API Documentation

Here's a basic *API documentation* for your Node.js + Express backend based on a typical course/quiz learning platform:

API Documentation

Base URL

http://localhost:5000/api

Auth Routes (/api/user)

1. POST /register

Registers a new user.

Request Body:

json

```
{
  "name": "John Doe",
  "email": "john@example.com",
  "password": "123456"
}
```

Response:

json

```
{
  "success": true,
  "message": "User registered successfully"
}
```

2. POST /login*

Logs in a user and returns a token.

Request Body:

json

```
{
  "email": "john@example.com",
  "password": "123456"
}
```

Response:

json

```
{
  "success": true,
  "message": "Login successful",
  "token": "JWT_TOKEN",
  "userData": {
    "_id": "user_id",
    "name": "John Doe",
    "email": "john@example.com"
  }
}
```

Course Routes (/api/user)

3. GET /getAllCourses

Returns a list of all courses.

Response:

json

```
[
  {
    "_id": "courseId1",
   "title": "React Basics",
    "description": "Learn React from scratch",
    "thumbnail": "url/to/image",
    "price": 299
  }
]
```

4. GET /getCourseContent/:courseId

Returns full content, video links, progress, and quiz for a course.

Requires Authorization header:

Authorization: Bearer <token>

Response:

json

```
{
  "success": true,
  "courseContent": {
    "title": "React Basics",
    "videos": [
      { "title": "Intro", "url": "/uploads/videos/intro.mp4" }
    ],
    "quiz": [
      {
        "question": "What is JSX?",
        "options": ["HTML", "JavaScript", "Java", "XML"],
        "answer": "JavaScript"
      }
    ],
    "progress": 50,
    "certificateAvailable": true
  }
```

}

Enrollment & Payment

5. POST /enroll

Enrolls a user in a course (after payment).

Request Body:

json

```
{
  "courseId": "courseId1"
}
```

Response:

json

```
{
  "success": true,
  "message": "Enrolled successfully"
}
```

6. GET /getEnrolledCourses

Returns list of user's enrolled courses.

Requires Auth

Response:

json

```
[
  {
    "courseId": "courseId1",
    "title": "React Basics",
    "progress": 60
  }
]
```

📝 Quiz Submission

7. POST /submitQuiz

Submits quiz answers and returns results.

Request Body:

json

```
{
  "courseId": "courseId1",
  "answers": [
    { "questionId": "q1", "selected": "A" },
    { "questionId": "q2", "selected": "C" }
  ]
}
```

Response:

json

```
{
  "success": true,
  "score": 8,
  "total": 10,
  "message": "Quiz submitted successfully"
}
```

📜 Certificate

8. GET /getCertificate/\:courseId

Returns certificate link (if eligible).

Response:

json

```
{
```

"success": true,
"certificateUrl":
"http://localhost:5000/certificates/cert_123.pdf"
}

7. Authentication

JWT (JSON Web Token) is used for authentication.

- When users log in, the server sends a token.
- The token is stored in localStorage on the frontend.
- All protected API requests include the token in the Authorization header.
- A middleware on the backend checks the token before allowing access.
- Passwords are securely hashed using bcryptjs.
- No sessions are stored — it's a stateless system.
- To log out, simply remove the token from localStorage.

8. UserInterface

Built using React + Vite for fast, modular frontend development. Responsive design for mobile and desktop (can use Bootstrap, Tailwind, etc.).Clean, user-friendly layout with navigation and protected routes.

9. Testing

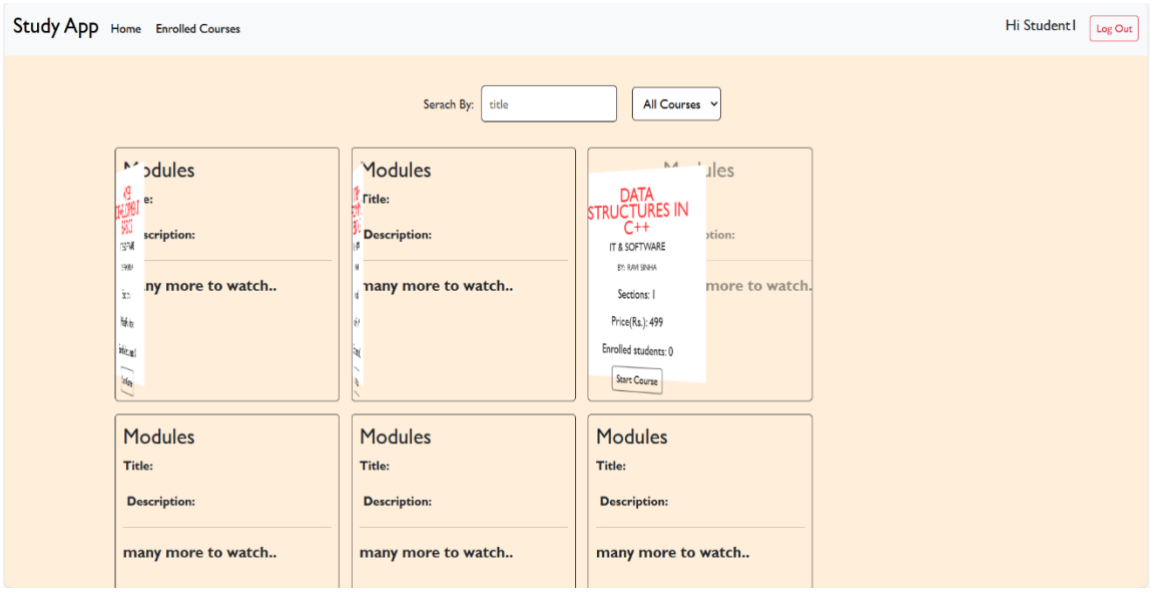
- Manual testing was used to verify core flows like:
- User registration and login
- Course enrollment and playback
- Quiz completion and certificate generation

Tools used:

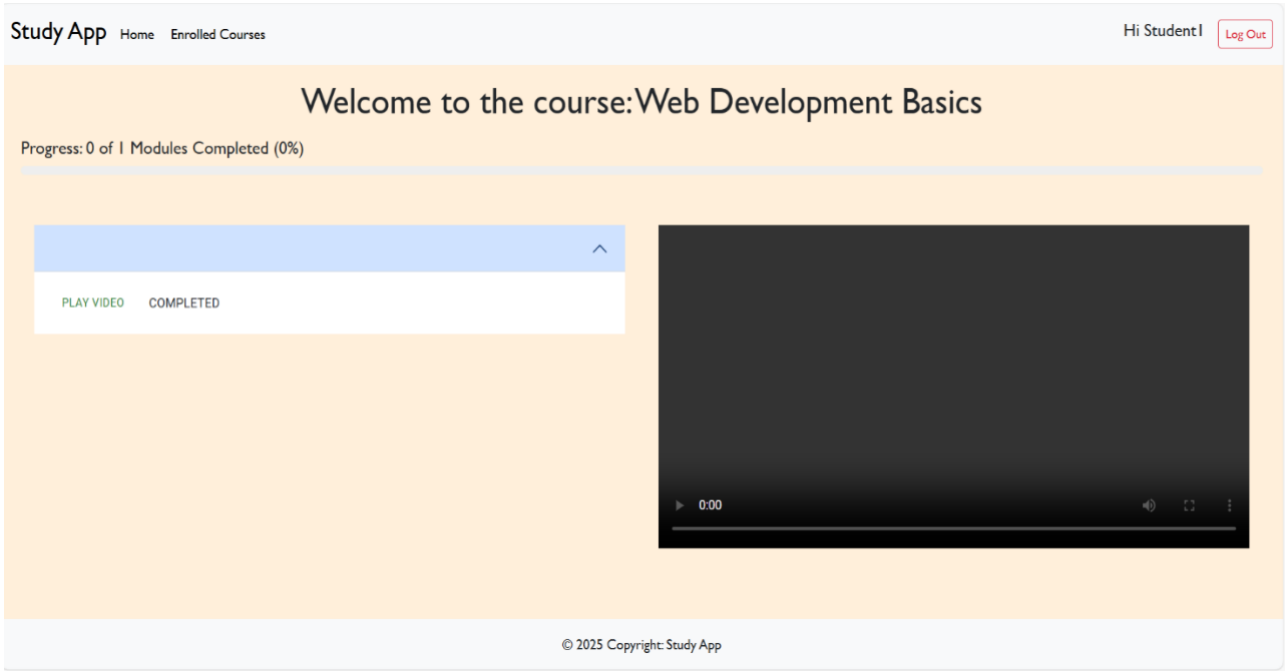
- Postman – For testing backend APIs (GET, POST, token auth).
- Browser DevTools – For inspecting network requests and debugging React components.
- Console logs – Used on both backend and frontend for tracking state and errors.

10. Demo application

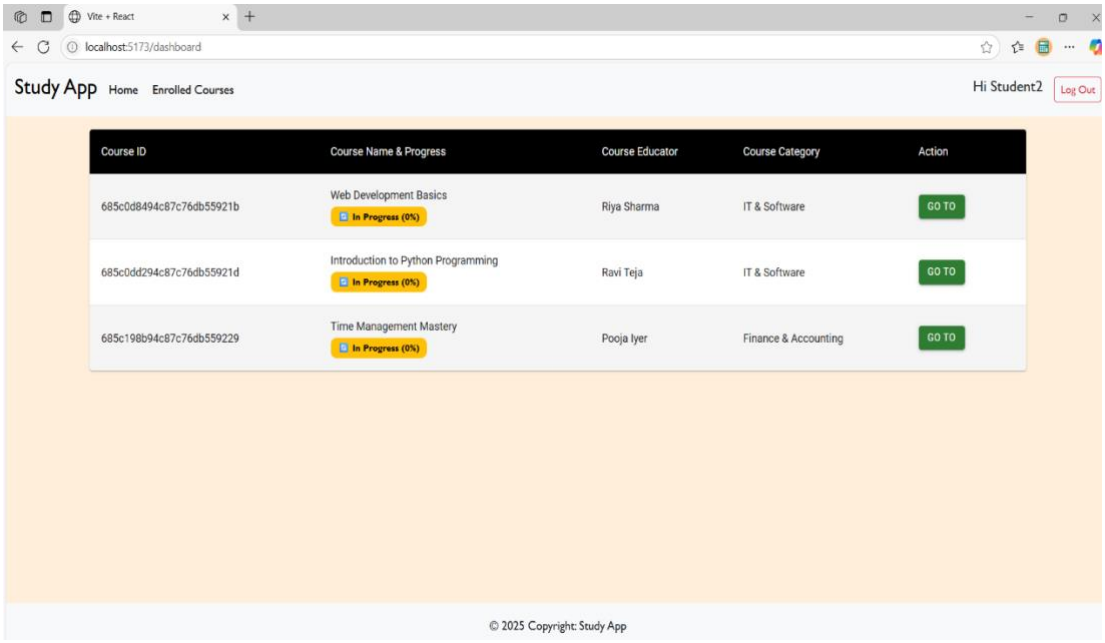
- List of courses in home page



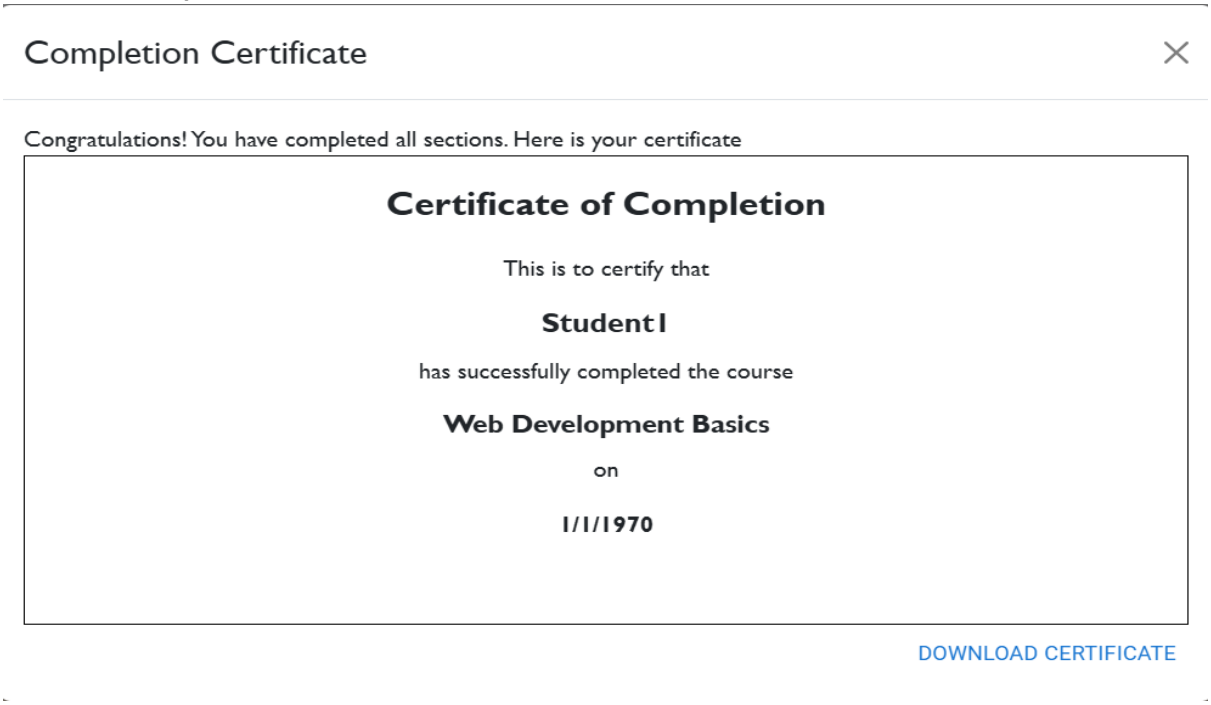
- Course content



- Enrolled courses



- Completion certificate



11. Known issues

Blank screen after login -May occur if React routing isn't properly set after login.
Quiz data not showing - Sometimes caused by missing quiz data in MongoDB or incorrect API call.
Certificate not generating -Happens if course completion check fails (e.g., progress not updated).
Video not loading - Could be due to wrong file path or missing videos in uploads/videos

12. Future Issues

- Live Classes Integration

Add real-time class support using Zoom, Jitsi, or WebRTC.

Enable live chat, raise-hand feature, and screen sharing.

- Mobile App Version

Build a React Native or Flutter version of Learn Hub.
Sync data with the web version using the same backend.

- Discussion Forums / Community

Add course-specific discussion forums. Use threaded replies, likes, and pinned posts.

- AI-Powered Quiz Generator

Use GPT API to generate quizzes from video transcripts or PDFs.Adaptive quizzes based on user's past performance.

- Gamification Features

Add badges, XP points, leaderboards, and streaks. Weekly goals to encourage consistent learning.

- Advanced Analytics,Admin dashboard showing course engagement, dropout rates, quiz success, etc.User dashboard with weekly learning summary and performance.
- Interactive Learning Tools

Code playgrounds for coding courses.

Code:link
Demo:link

THE END