LEARNHUB – YOUR CENTER FOR SKILL ENHANCEMENT

Online Learning & Skill Development Management System

Project Description:

In today's rapidly evolving digital world, the education and skill development industry is undergoing significant transformation. Traditional learning methods often require physical classroom attendance, fixed schedules, geographical limitations, and limited access to expert mentors. These conventional approaches may lack flexibility, affordability, and accessibility, creating challenges for both learners and educators.

With the advancement of web technologies and increased internet accessibility, there is a strong need for a centralized digital platform that simplifies and enhances the learning experience. LearnHub – Your Center for Skill Enhancement is developed to address these challenges by providing a secure, scalable, and user-friendly web-based solution for online course creation, management, and learning.

The primary objective of the LearnHub system is to bridge the gap between students and mentors by offering a digital ecosystem where skill development can take place efficiently and interactively. The platform enables mentors to create and upload detailed course information, including course title, description, category, pricing, skill level, and video lectures. On the other hand, students can browse available courses, apply filters, enroll in programs, and access learning materials without geographical constraints.

Unlike traditional learning platforms that provide limited features, LearnHub offers:

- Secure user authentication and role-based access
- Real-time course browsing and filtering
- Structured database-driven course management
- Video upload functionality for mentors
- Student enrollment and progress tracking
- Admin-level monitoring and management
- Modular and scalable backend architecture

The system follows a modern full-stack development approach using React for the frontend interface, Node.js and Express for backend API handling, and MongoDB for database management. This architecture ensures separation of concerns, improved performance, and easier maintainability.

One of the key strengths of the LearnHub system is its role-based access mechanism. The application supports three primary user roles:

1. Admin – Responsible for managing users and overseeing platform operations.
2. Mentor/Teacher – Can create, update, and manage courses and upload learning materials.
3. Student – Can browse courses, enroll in programs, and access course content.

By implementing secure authentication using JSON Web Tokens (JWT) and password encryption using bcrypt, the system ensures user data confidentiality and protection against unauthorized access.

The system is also designed with scalability in mind. As the number of users and courses increases, the backend structure and database organization allow smooth handling of larger datasets without performance degradation.

Furthermore, the filtering functionality enhances user experience by allowing course searches based on:

- Course category (Programming, Design, Marketing, etc.)
- Skill level (Beginner, Intermediate, Advanced)

• Price (Free or Paid)
• Instructor name
• Ratings

This ensures that students can quickly find courses that match their learning goals.

In addition to functional efficiency, the system emphasizes usability and interface design. The frontend is built using a component-based architecture, enabling responsive layouts and reusable UI components. The design ensures smooth navigation, intuitive dashboards, and minimal complexity.

Overall, LearnHub – Your Center for Skill Enhancement demonstrates a practical implementation of full-stack web development concepts, including:

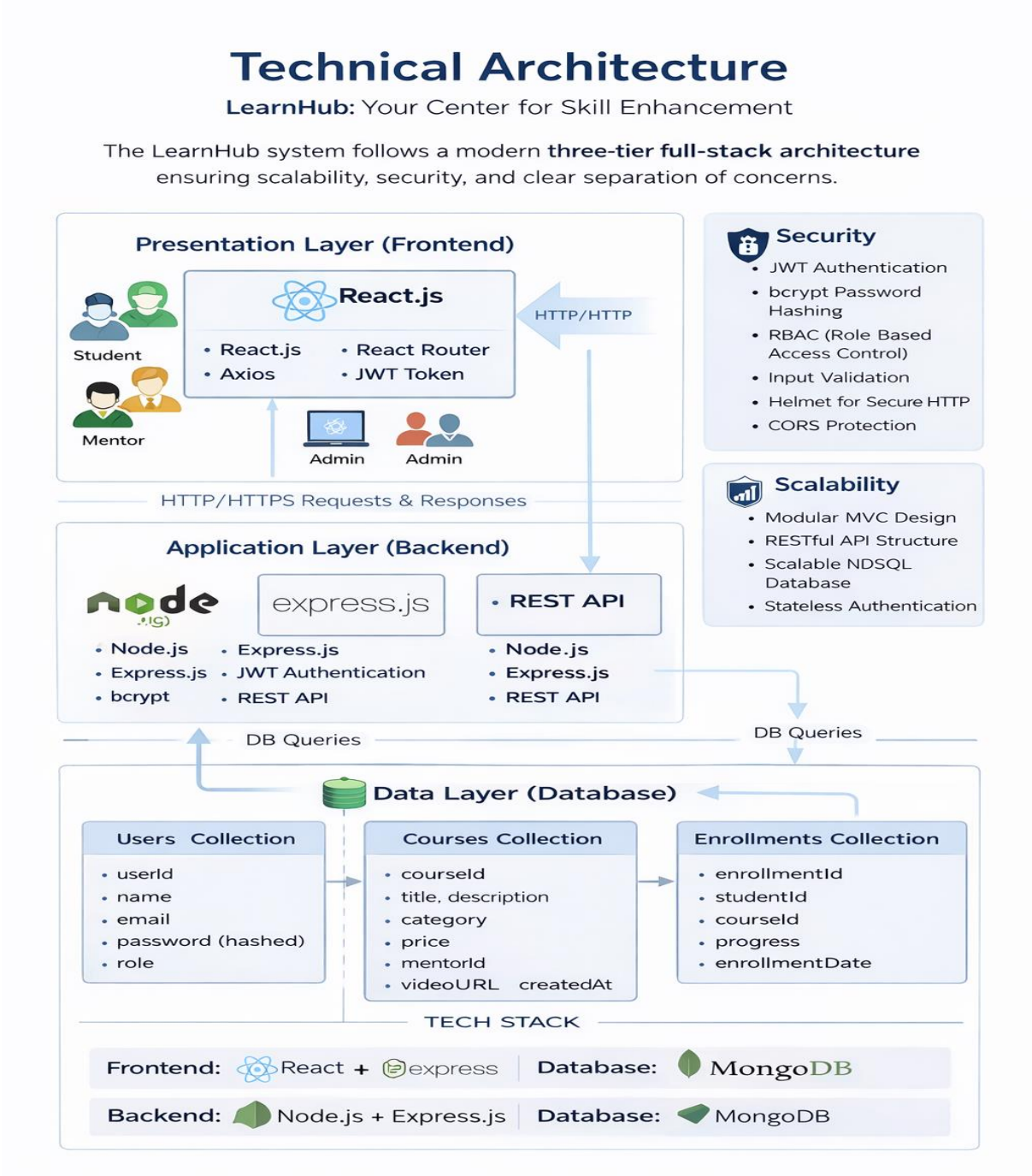
• RESTful API design
• MVC architecture
• Database modeling
• Authentication and authorization
• File handling (video upload)
• Client-server communication
• Modular project structuring

The project showcases how modern web technologies can be leveraged to create a reliable, scalable, and secure real-world application that enhances online education and skill development.

In addition, LearnHub promotes continuous learning and professional growth by providing an interactive and accessible environment for both students and mentors. The system encourages knowledge sharing, skill enhancement, and collaborative learning through structured course delivery and digital accessibility. It also supports future enhancements such as certification generation, live classes integration, discussion forums, payment gateway integration, and performance analytics.

With its scalable architecture and modular design, LearnHub can be extended to support thousands of users simultaneously, making it suitable for real-world deployment. The platform not only demonstrates technical implementation skills but also reflects practical problem-solving by addressing modern educational challenges through technology-driven solutions.

Technical Architecture:



Pre requisites:

successfully develop, test, and deploy the LearnHub – Your Center for Skill Enhancement system, certain software tools, programming knowledge, and packages are required. These prerequisites ensure smooth development, proper backend–frontend integration, database connectivity, secure authentication, and efficient course management implementation.

Software Required

1 Node.js

- Node.js is a JavaScript runtime environment used to execute JavaScript code outside the browser. It is used to build the backend server for handling API requests, authentication, and database communication.

• Enables server-side scripting

• Supports asynchronous programming

• Handles concurrent requests efficiently

• Required for running Express server

-

2 MongoDB

- MongoDB is a NoSQL database used to store application data in JSON-like BSON format.
- Stores user, course, and enrollment details
- Supports flexible schema design
- Provides high performance and scalability
- Integrates easily with Node.js using Mongoose

3 Visual Studio Code (VS Code)

- VS Code is the primary code editor used for writing frontend and backend code.
- Recommended extensions:
- ES7+ React Snippets
- Prettier (Code Formatter)

- MongoDB for VS Code
- ESLint
- REST Client

4️⃣ Postman

- Postman is used for testing backend APIs.
- Sends GET, POST, PUT, DELETE requests
- Validates API responses
- Helps debug server-side issues
- Tests authentication tokens

5️⃣ Git

- Git is used for version control.
- Tracks code changes
- Enables collaboration
- Supports branching and merging
- Maintains project history

6️⃣ Web Browser (Chrome Recommended)

- Used for frontend testing and debugging.
- Developer Tools for inspecting API calls
- Console for debugging JavaScript
- Network tab for monitoring requests

📦 **Backend Dependencies**

1️⃣ express

- Express.js is a lightweight web framework for Node.js.
- Handles routing
- Manages middleware
- Simplifies REST API development

2. mongoose

- Mongoose is an ODM (Object Data Modeling) library for MongoDB.
- Defines schemas
- Enforces validation rules
- Manages database relationships
- Simplifies CRUD operations

3. jsonwebtoken (JWT)

- Used for secure authentication.
- Generates authentication tokens
- Verifies user sessions
- Protects private routes

4. bcrypt

- Used for password hashing.
- Encrypts passwords before storing in database
- Prevents plain-text password storage
- Enhances data security

5. dotenv

- Used to manage environment variables.

- Stores sensitive data like:
- Database URI
- JWT Secret Key
- Server Port
- Keeps credentials secure

6. multer

- Middleware for handling file uploads.
- Uploads course videos or thumbnails
- Stores files in uploads folder
- Manages file size and format validation

7. cors

- Enables Cross-Origin Resource Sharing.
- Allows frontend and backend communication
- Prevents CORS errors

8. nodemon (Development Dependency)

- Automatically restarts server when code changes.
- Improves development speed
- Avoids manual server restart

9. express-validator

- Used for input validation.

- Validates user inputs
- Prevents invalid data submission
- Enhances API reliability

🔟 helmet

- Adds security headers to HTTP responses.
- Protects against common vulnerabilities
- Improves API security

📦 **Frontend Dependencies**

1️⃣ react

- Core library for building user interfaces.
- Component-based architecture
- Virtual DOM for fast rendering
- Reusable UI components

2️⃣ react-router-dom

- Used for navigation and routing.
- Manages multiple pages
- Enables protected routes
- Improves SPA (Single Page Application) experience

3️⃣ axios

- Used for making HTTP requests to backend APIs.
- Sends GET, POST, PUT, DELETE requests

- Handles JSON data
- Supports token-based authentication headers
-

4. bootstrap / CSS

- Used for styling and responsive design.
- Grid system
- Predefined UI components
- Mobile-friendly layouts

5. react-icons

- Provides scalable icons.
- Enhances UI design
- Improves user interaction

6. react-toastify

- Displays notifications and alerts.
- Success messages
- Error alerts
- Login confirmation messages

7. formik / react-hook-form (Optional)

- Used for form handling and validation.
- Simplifies form state management
- Reduces boilerplate code

8. redux / context API (Optional for advanced version)

- Used for global state management.
- Manages authentication state
- Stores user data globally
- Improves scalability

Prior Knowledge:

You must have prior knowledge of the following topics to successfully complete the LearnHub – Your Center for Skill Enhancement project.

● Full Stack Development Concepts

o Frontend Development (React Basics):

https://react.dev/learn

o Components, Props & State Management:

https://react.dev/learn/passing-props-to-a-component

o React Hooks (useState, useEffect):

https://react.dev/reference/react

o React Router:

https://reactrouter.com/en/main/start/tutorial

o Axios (API Integration):

https://axios-http.com/docs/intro

● **Backend Development (Node.js & Express)**

o Node.js Basics:

https://nodejs.org/en/docs

o Express.js Framework:

https://expressjs.com/en/starter/installing.html

o REST API Concepts:

https://restfulapi.net/

o Middleware in Express:

https://expressjs.com/en/guide/using-middleware.html

o MVC Architecture:

https://www.geeksforgeeks.org/mvc-design-pattern/

● **Database Concepts (MongoDB)**

o MongoDB Basics:

https://www.mongodb.com/docs/manual/introduction/

o Mongoose (ODM):

https://mongoosejs.com/docs/

o CRUD Operations:

https://www.mongodb.com/docs/manual/crud/

o Schema Design & Validation:

https://mongoosejs.com/docs/guide.html

● Authentication & Security

o JWT (JSON Web Token):

https://jwt.io/introduction

o Password Hashing using bcrypt:

https://www.npmjs.com/package/bcrypt

o Role-Based Access Control (RBAC):

https://auth0.com/docs/manage-users/access-control/rbac

● **Additional Concepts**

o HTTP Methods & Status Codes:

https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

o JSON Data Format:

https://www.json.org/json-en.html

o File Upload using Multer:

https://www.npmjs.com/package/multer

o Git & Version Control:

https://git-scm.com/docs

Project Objectives:

By the end of this project, you will:

• Understand the fundamentals of full-stack web application development.
• Gain practical knowledge of frontend development using React.js.
• Learn how to design and implement RESTful APIs using Node.js and Express.
• Develop the ability to design and manage a MongoDB database using Mongoose.
• Understand how to implement secure authentication using JWT and password hashing.
• Learn how to build role-based access control (Admin, Mentor, Student).
• Implement CRUD (Create, Read, Update, Delete) operations for course management.
• Develop dynamic course browsing and filtering functionality.
• Learn how to upload and manage course videos or thumbnails using Multer middleware.
• Implement student enrollment and progress tracking features.
• Understand the integration between frontend and backend using Axios.

• Gain experience in modular project structuring using MVC architecture.
• Learn how to test APIs using Postman.
• Improve debugging and error-handling skills.
• Understand how to deploy and maintain a scalable web application.
• Develop real-world problem-solving skills through practical implementation in an online learning platform.

Project Flow:

• The user interacts with the React-based User Interface (UI).

• The frontend collects user input such as registration details, login credentials, course creation data, or enrollment requests.

• The frontend sends HTTP requests to the backend APIs using Axios.

• The backend (Node.js + Express) receives the request and validates the data.

• Authentication is verified using JWT tokens.

• Business logic is executed in the controller layer.

• The backend communicates with MongoDB database using Mongoose.

• The database processes the query and returns the required data.

• The backend sends a JSON response back to the frontend.

• The frontend dynamically updates the UI and displays the result to the user.

To accomplish this, the following activities are performed:

• **Requirement Analysis**

  * Identify user roles (Admin, Mentor, Student)

  * Define system features and functionalities

• **Backend Development**

  * Setup Node.js and Express server

  * Configure MongoDB database connection

  * Create Mongoose schemas (User, Course, Enrollment)

  * Implement authentication using JWT and bcrypt

  * Develop RESTful APIs

    * Implement role-based access control
    * Configure Multer for video or file uploads

- **Frontend Development**
    * Create React application
    * Design UI components and pages
    * Implement routing using React Router
    * Integrate APIs using Axios
    * Implement protected routes

- **Course Management**
    * Mentor creates and manages courses
    * Upload videos or course materials
    * Students browse and enroll in courses
    * Track learning progress

- **Testing & Debugging**
    * Test APIs using Postman
    * Validate frontend forms
    * Perform integration testing
    * Fix bugs and optimize performance

- **Deployment**
    * Configure environment variables
    * Build production-ready frontend
    * Deploy backend and database
    * Perform final testing

Project Structure:

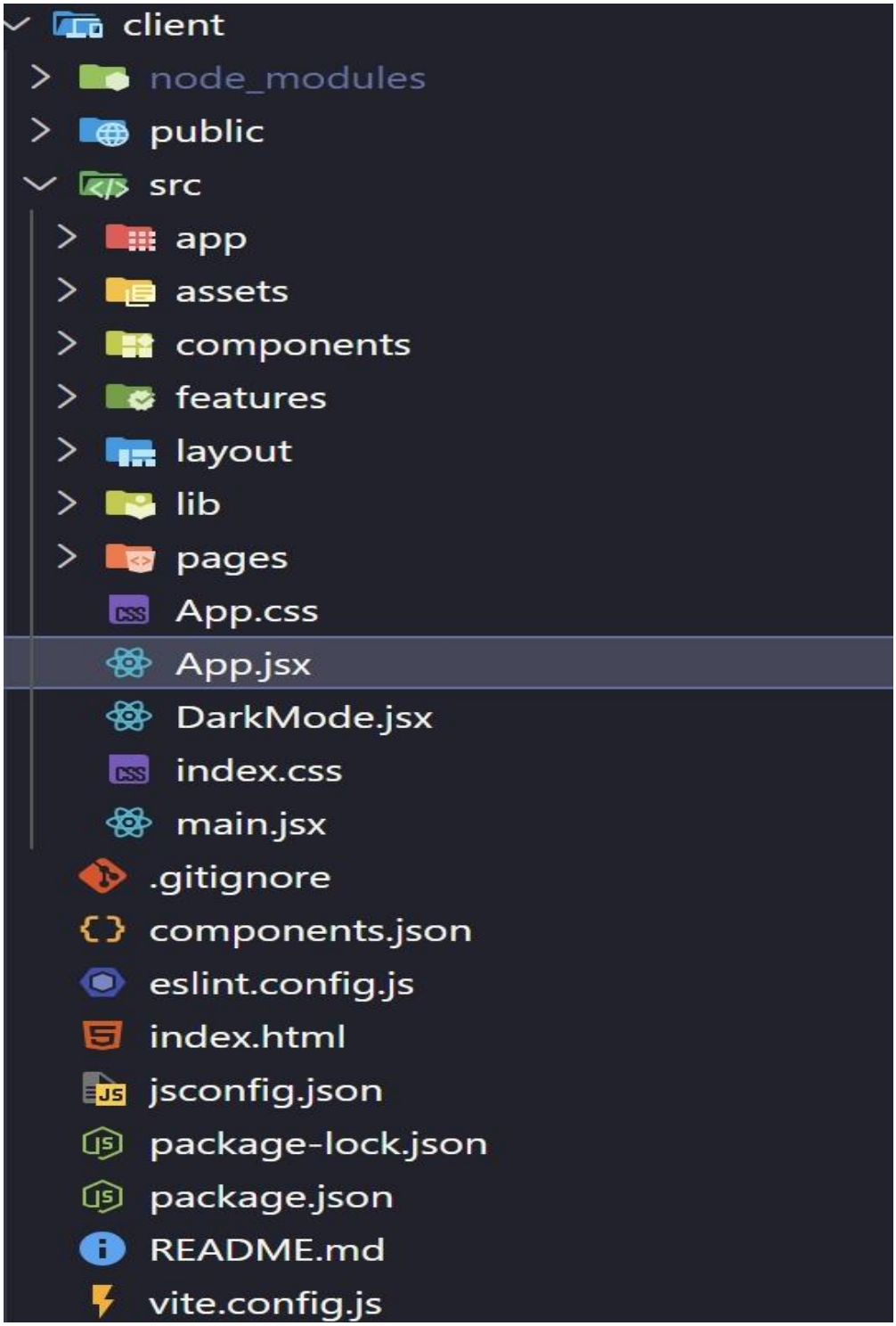
Create the Project folder which contains files as shown below:

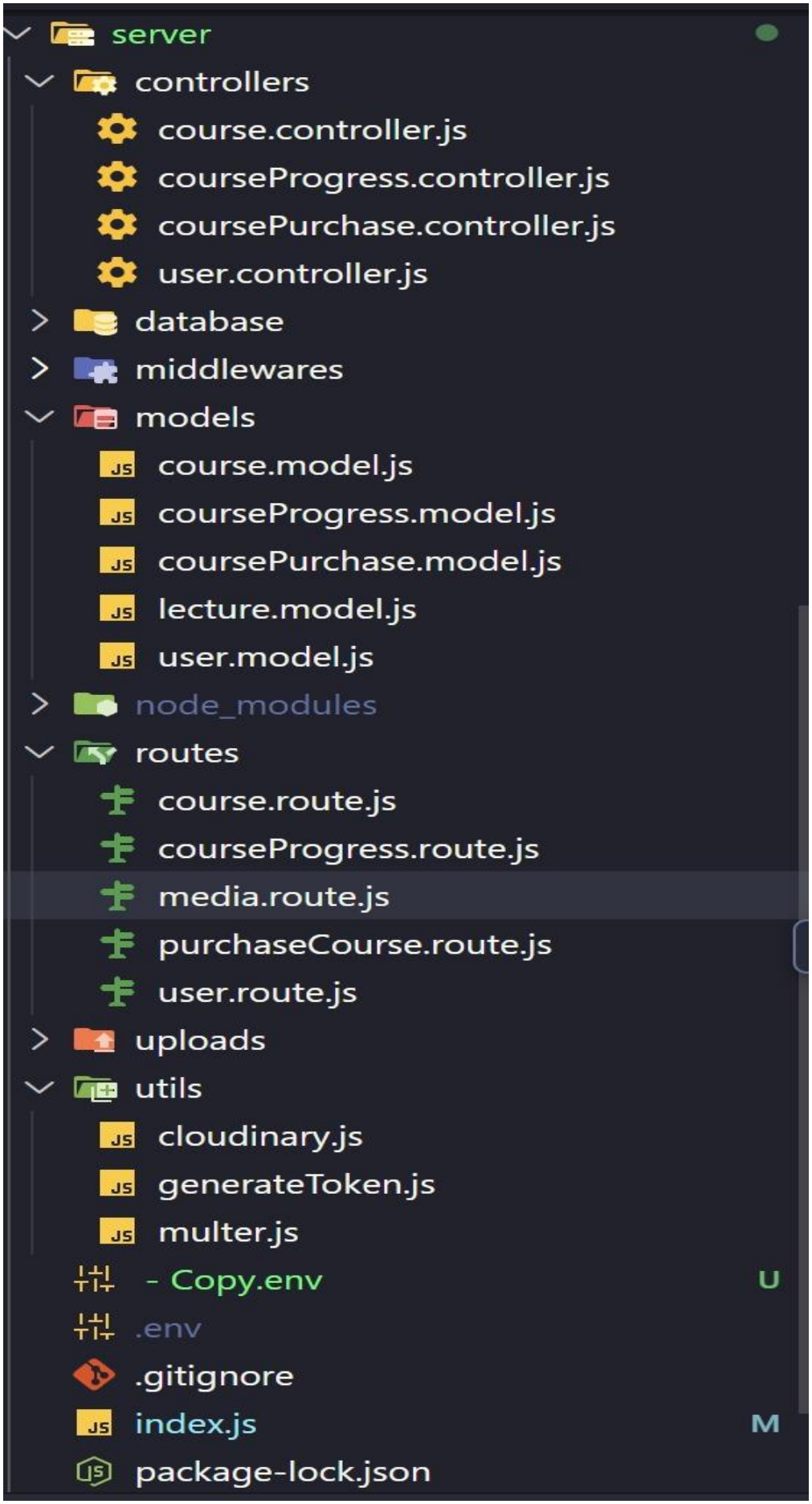
LEARNHUB-MAIN

```
|
 ├— client
|   ├— node_modules
|   ├— public
|   ├— src
|   |   ├— app
|   |   ├— assets
|   |   ├— components
|   |   ├— features
|   |   ├— layout
|   |   ├— lib
|   |   ├— pages
|   |   ├— App.css
|   |   ├— App.jsx
|   |   ├— DarkMode.jsx
|   |   ├— index.css
|   |   ├— main.jsx
|   ├— .gitignore
|   ├— components.json
|   ├— eslint.config.js
|   ├— index.html
|   ├— jsconfig.json
|   ├— package-lock.json
|   ├— package.json
|   ├— vite.config.js
```

```
|   └── README.md
|
├── server
|   ├── controllers
|   |   ├── course.controller.js
|   |   ├── courseProgress.controller.js
|   |   ├── coursePurchase.controller.js
|   |   └── user.controller.js
|   |
|   ├── database
|   ├── middlewares
|   ├── models
|   |   ├── course.model.js
|   |   ├── courseProgress.model.js
|   |   ├── coursePurchase.model.js
|   |   ├── lecture.model.js
|   |   └── user.model.js
|   |
|   ├── routes
|   |   ├── course.route.js
|   |   ├── courseProgress.route.js
|   |   ├── media.route.js
|   |   ├── purchaseCourse.route.js
|   |   └── user.route.js
|   |
|   ├── uploads
|   ├── utils
|   |   ├── cloudinary.js
|   |   ├── generateToken.js
|   |   └── multer.js
```

```
|  |
|  ├── .env
|  ├── index.js
|  ├── package-lock.json
|  └── package.json
```




server
controllers
course.controller.js
courseProgress.controller.js
coursePurchase.controller.js
user.controller.js
database
middlewares
models
course.model.js
courseProgress.model.js
coursePurchase.model.js
lecture.model.js
user.model.js
node_modules
routes
course.route.js
courseProgress.route.js
media.route.js
purchaseCourse.route.js
user.route.js
uploads
utils
cloudinary.js
generateToken.js
multer.js
- Copy.env
U
.env
.gitignore
index.js
M
package-lock.json

Backend Folder:

The backend folder (server) contains all server-side logic and API implementations.

● **database folder**

Contains MongoDB database connection configuration files.

● **controllers folder**

Contains business logic files such as:

• user.controller.js

• course.controller.js

• courseProgress.controller.js

• coursePurchase.controller.js

● **middlewares folder**

Contains authentication and authorization middleware such as:

• JWT verification

• Role-based access control (Admin / Mentor / Student)

● **routes folder**

Defines REST API endpoints such as:

• /api/users

• /api/courses

• /api/progress

• /api/purchase

• /api/media

● **models folder**

Contains Mongoose models such as:

• User Model

• Course Model

• Lecture Model

• Course Progress Model

• Course Purchase Model

● **uploads folder**

Stores uploaded course files or media temporarily.

● **utils folder**

Contains helper utilities such as:

• Token generation

• Multer configuration

• Cloudinary integration

● **.env file**

Stores environment variables such as:

• MongoDB URI

• JWT Secret

• Cloudinary Keys

• Server Port

● **index.js**

Main entry file of backend server. It connects to the database and starts the Express server.

Frontend Folder:

The frontend folder (client) contains React application files.

● **public folder**

Contains static files and index.html template.

● **src folder**

Contains React components, features, and pages such as:

• Login Page

• Registration Page

• Course Listing Page

• Course Detail Page

• Mentor Dashboard

• Student Dashboard

• Admin Dashboard

● **App.jsx**

Main React application component.

● **main.jsx**

Entry point of the React application.

● **package.json**

Contains frontend dependencies and scripts.

● **vite.config.js**

Configuration file for Vite build tool.

Milestone 1: Requirement Analysis

Software development begins with proper requirement analysis. In this phase, the system requirements of LearnHub – Your Center for Skill Enhancement are identified, analyzed, and documented clearly.

Activity 1: Identifying System Requirements

The LearnHub E-Learning Platform must support the following features:

• User Registration and Login

• Role-Based Access (Admin / Mentor / Student)

• Course Creation and Management

• Course Search and Filtering

• Course Thumbnail & Media Upload

• Secure Authentication
• Admin Dashboard Monitoring
• Course Purchase and Progress Tracking

Activity 2: Functional Requirements

• Users must be able to register and log in securely.
• Mentors must be able to create, update, publish, and delete courses.
• Students must be able to browse, search, and enroll in courses.
• Students must be able to track course progress.
• Admin must be able to manage users and monitor revenue.
• Course thumbnails and media files must be uploaded and stored properly.

Activity 3: Non-Functional Requirements

• System must be secure.
• System must be scalable.
• Response time must be fast.
• UI must be user-friendly and responsive.
• System must handle multiple users simultaneously.
• Data must be protected using encryption and secure authentication mechanisms.

Milestone 2: Backend Development

After requirement analysis, backend development is initiated.

Activity 1: Project Setup

• Install Node.js
• Initialize npm project
• Install required dependencies

• Configure environment variables
• Setup Express server

Activity 2: Database Connection

• Install MongoDB
• Configure Mongoose
• Connect database to backend server
• Handle connection errors

Activity 3: Creating Schemas

Create Mongoose schemas for:

• User Schema (Admin / Mentor / Student roles)
• Course Schema
• Lecture Schema
• Course Purchase Schema
• Course Progress Schema

Each schema defines fields, validation rules, and data types.

Activity 4: Implementing Authentication

• Hash passwords using bcrypt
• Generate JWT tokens
• Create login and registration APIs
• Implement role-based access middleware
• Protect private routes using middleware

Activity 5: Building REST APIs

Implement CRUD operations:

- POST → Create new course
- GET → Fetch all courses
- GET → Fetch single course details
- PUT → Update course details
- DELETE → Remove course
- POST → Purchase course
- GET → Fetch course progress

Activity 6: Media Upload

- Configure Multer middleware
- Upload course thumbnails and lecture videos
- Integrate Cloudinary for cloud storage
- Validate file type and size

Milestone 3: Frontend Development

After backend APIs are ready, frontend development begins.

Activity 1: Create React Application

- Setup React project using Vite
- Install required dependencies
- Configure routing using React Router
- Setup global state management

Activity 2: Design User Interface

Develop the following pages:

- Home Page (Course Listing & Search)
- Login Page
- Signup Page
- Student Dashboard
- Mentor Dashboard
- Course Management Page
- Add / Edit Course Page
- Admin Dashboard

Activity 3: API Integration

- Use Axios to connect backend APIs
- Send authentication tokens in headers
- Display dynamic course data
- Handle error and success responses

Activity 4: Implement Filtering & Search

- Create course search bar
- Implement category-based filtering
- Implement level-based filtering (Beginner / Intermediate / Advanced)
- Display filtered results dynamically

Milestone 4: Testing & Debugging

Testing ensures the system works correctly.

Activity 1: Backend Testing

- Test APIs using Postman
- Verify authentication and authorization
- Validate error handling
- Test CRUD operations

Activity 2: Frontend Testing

- Test form validation
- Verify UI responsiveness
- Check routing functionality
- Test protected routes

Activity 3: Integration Testing

- Ensure frontend and backend communication
- Validate token-based authentication
- Confirm database operations
- Test course creation and purchase workflow

Milestone 5: Deployment & Finalization

Activity 1: Code Optimization

- Remove unnecessary code
- Optimize database queries
- Improve UI responsiveness
- Improve API performance

Activity 2: Deployment Preparation

• Configure environment variables
• Prepare production build
• Deploy backend server
• Deploy frontend application
• Test final deployed application

Activity 3: Documentation

• Prepare project report
• Add screenshots
• Create architecture diagrams
• Prepare database schema diagram
• Final review and submissionApplication Screens & Output Explanation

1️⃣User Registration Page (Sign Up Interface)

The first image represents the User Registration (Sign Up) Page of the LearnHub E-Learning system.

This page allows new users to create an account by entering:

• Full Name
• Email Address
• Password

Functional Explanation:

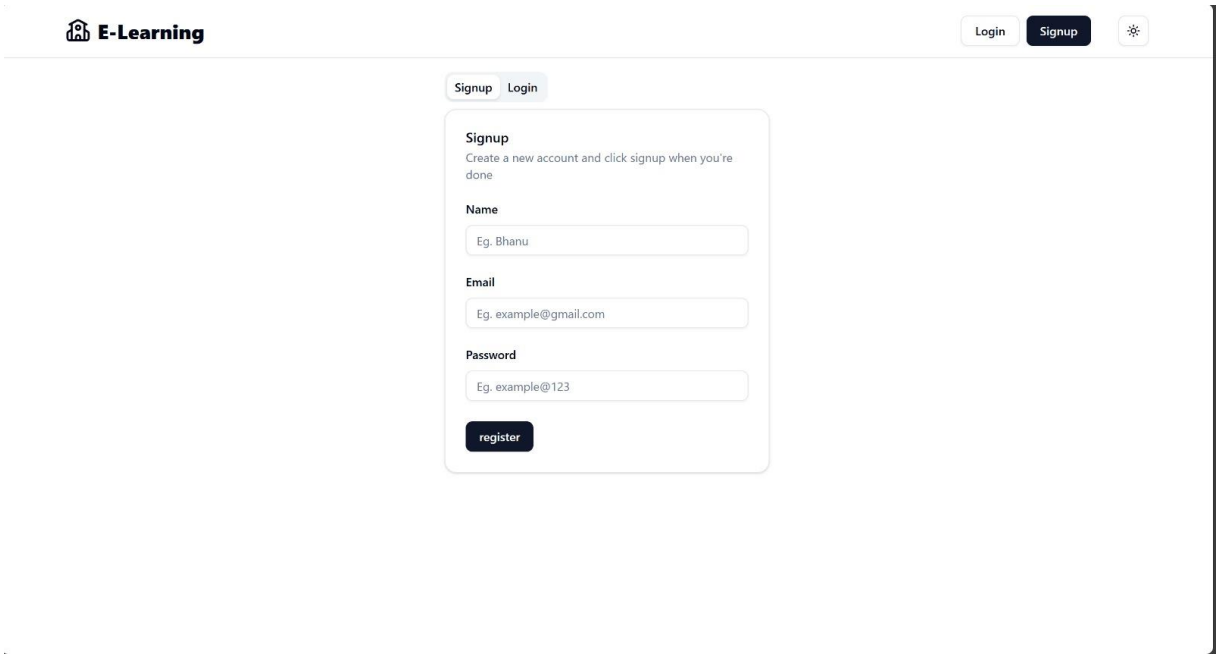
• The user enters personal details.
• The system validates the input fields.
• Email format validation is performed.
• Password is securely hashed using bcrypt before storing.
• Upon successful registration, user data is stored in MongoDB.
• A JWT token is generated for authentication.

• User is redirected to dashboard or login page.

Security Features Implemented:

• Password hashing using bcrypt
• Duplicate email validation
• JWT-based authentication
• Server-side validation
• Protected routes using middleware

This page ensures that only authenticated users can access platform functionalities.



2️⃣ Home Page – Course Listing & Search Interface

The second image represents the LearnHub Home Page where users can explore available courses.

Features Available:

• Search courses using search bar
• Explore featured courses
• View course details such as:

- Course Title
- Instructor Name
- Difficulty Level (Beginner / Medium / Advanced)
- Course Price

• Navigate to Login or Signup

• Dark mode toggle option

Functional Flow:

• When the page loads, the frontend sends a GET request to fetch courses.

• Backend retrieves course data from MongoDB.

• Data is returned in JSON format.

• React dynamically renders course cards.

Search Mechanism:

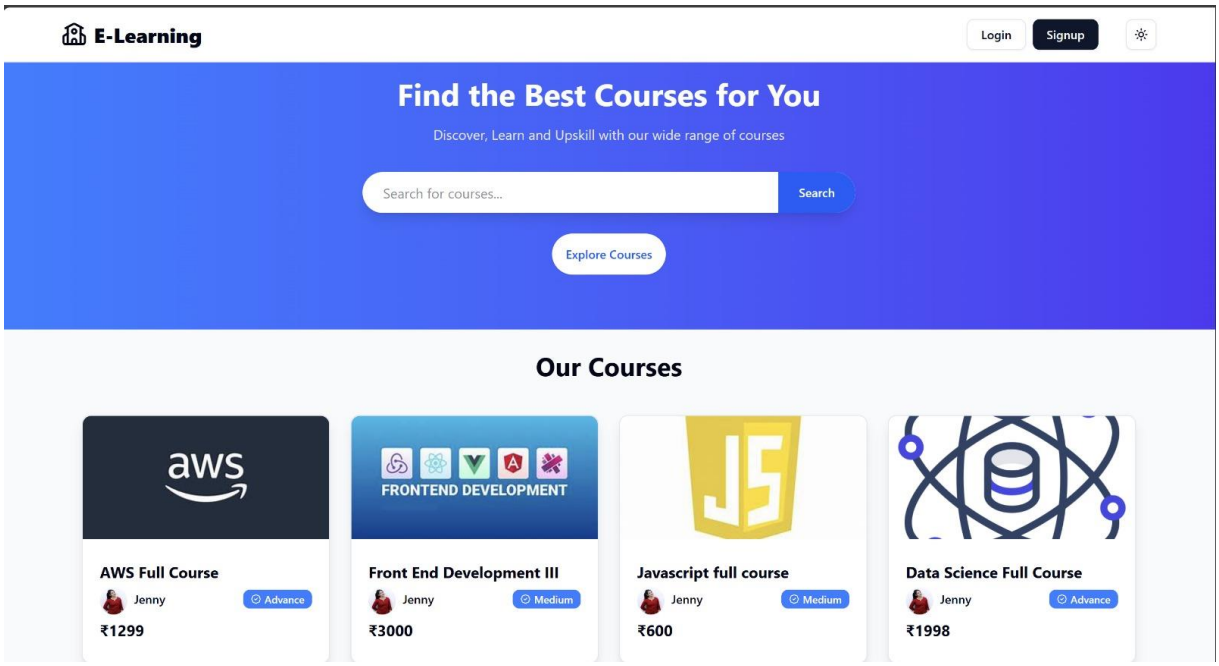
• User enters keywords in search bar.

• Search query is sent to backend.

• Backend filters courses based on title/category.

• Filtered results are returned and displayed instantly.

This improves user experience by allowing quick and efficient course discovery.



3️⃣ Admin Dashboard – Analytics Page

The third image represents the Admin Dashboard of LearnHub.

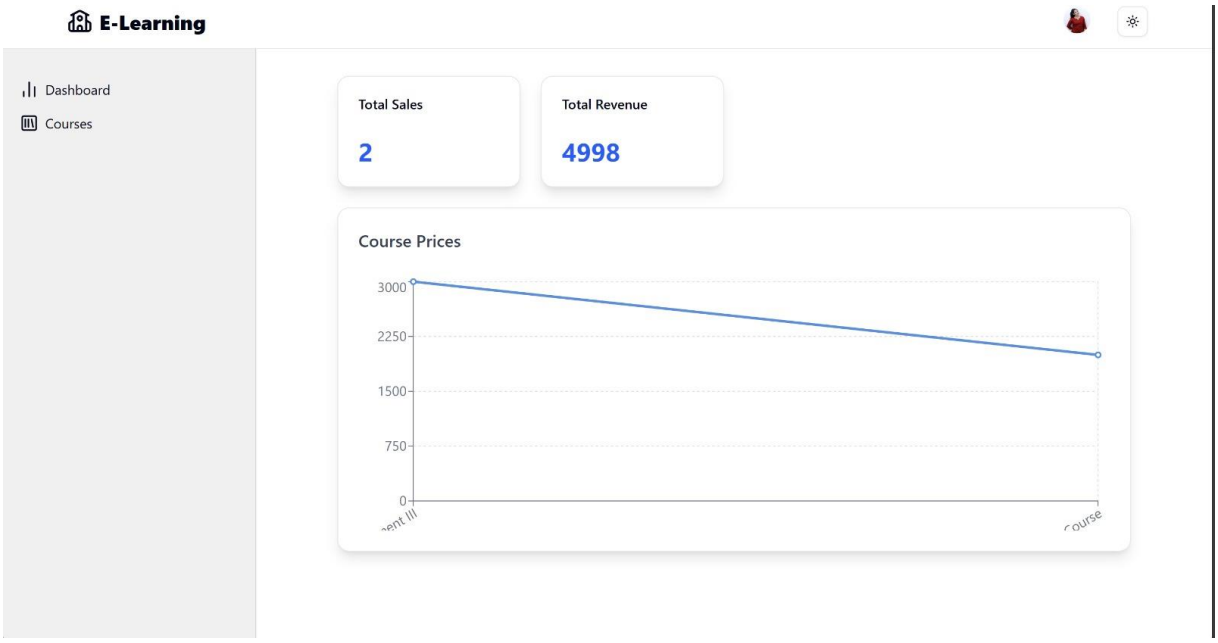
Features Available:

- Total Sales count
- Total Revenue generated
- Course price analytics (graph visualization)
- Sidebar navigation (Dashboard, Courses)

Functional Flow:

- Backend aggregates purchase data.
- Total revenue and sales are calculated.
- Data is sent to frontend.
- Chart component renders graphical representation.

This dashboard enables administrators to monitor platform performance and revenue statistics.



4️⃣ Course Management Page (Admin / Mentor Panel)

This page displays the list of all created courses.

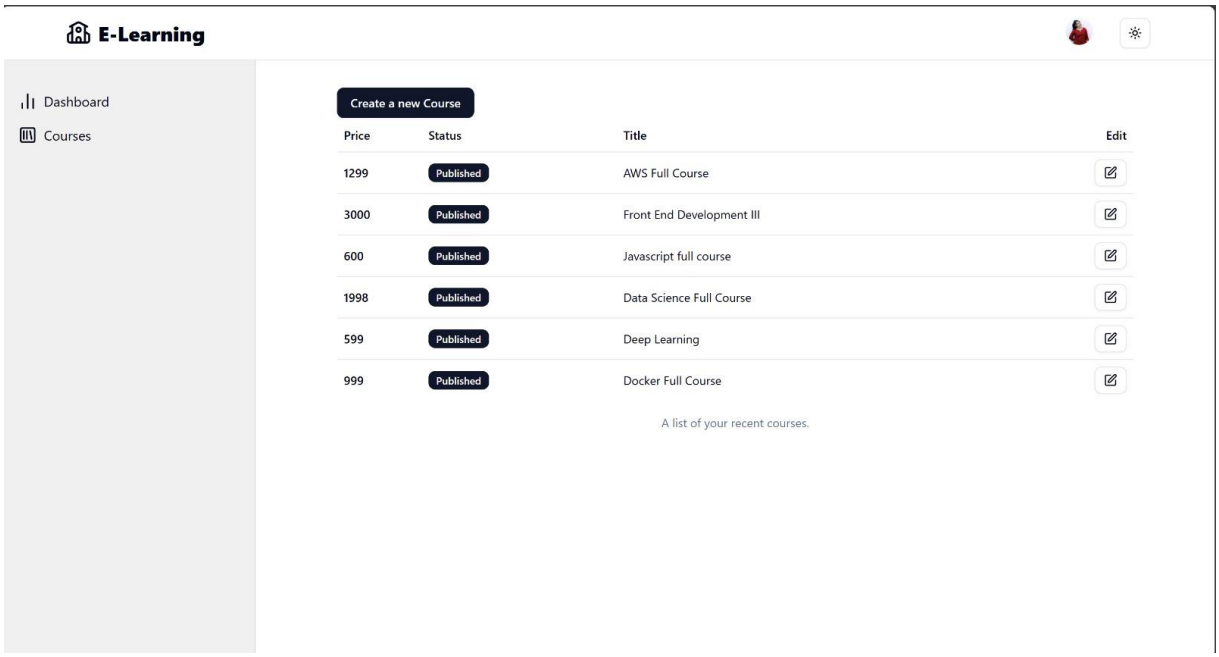
Features Available:

- View all courses
- View course price
- View publishing status
- Edit course
- Create new course

Functional Flow:

- Backend fetches all courses from database.
- Courses are displayed in tabular format.
- Edit button redirects to course detail page.
- Publish/Unpublish option controls course visibility.

This module ensures structured and efficient course management.



5️⃣ Add / Edit Course Page

This page allows mentors or admin to add detailed information about a course.

Course Details Collected:

- Course Title
- Subtitle
- Description
- Category
- Course Level
- Price (INR)
- Course Thumbnail Image

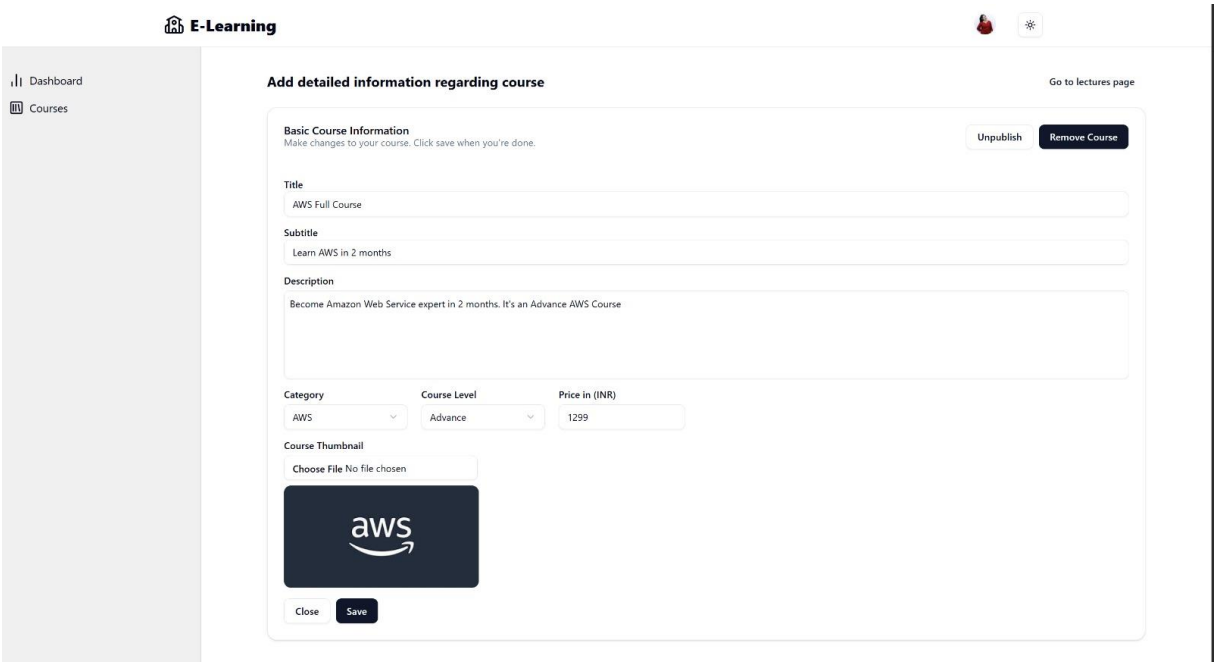
Functional Flow:

1. Mentor fills course form.
2. Thumbnail is uploaded using Multer middleware.
3. Data is sent to backend via POST request.
4. Backend validates input fields.
5. Course details are stored in MongoDB.
6. Course can be Published or Unpublished.

Validation Implemented:

- Required field validation
- Numeric validation for price
- Authentication check before submission
- File type validation for thumbnail

This module ensures structured and secure course management.



UI Design & User Experience

The interface follows a modern and minimal design:

- Clean navigation bar (Home, Login, Signup)
- Gradient hero section
- Responsive layout
- Card-based course display
- Sidebar-based dashboard
- Dark mode feature

The design ensures:

- Ease of use
- Smooth navigation
- Reduced user confusion
- Fast loading performance
- Better user engagement

System Output Summary

The LearnHub system successfully demonstrates:

- Secure user authentication
- Real-time course listing
- Dynamic search functionality
- Role-based dashboard access
- Course creation and editing
- Revenue analytics visualization
- Image upload and storage
- Database-driven data rendering