# BATCH-ID LTVIP2025TMID59565

**1.Introduction**

**• Project Title:**

* LearnHub: Your Center for Skill Enhancement

**• Team Members:**

* **Team Leader :** Vemulapalli Sreshta
* **Team member :** Upputuri Harini
* **Team member :** Uppiretla Venkata Vaishnavi
* **Team member :** Varikallu Devi Supraja

**2. Application Flow**

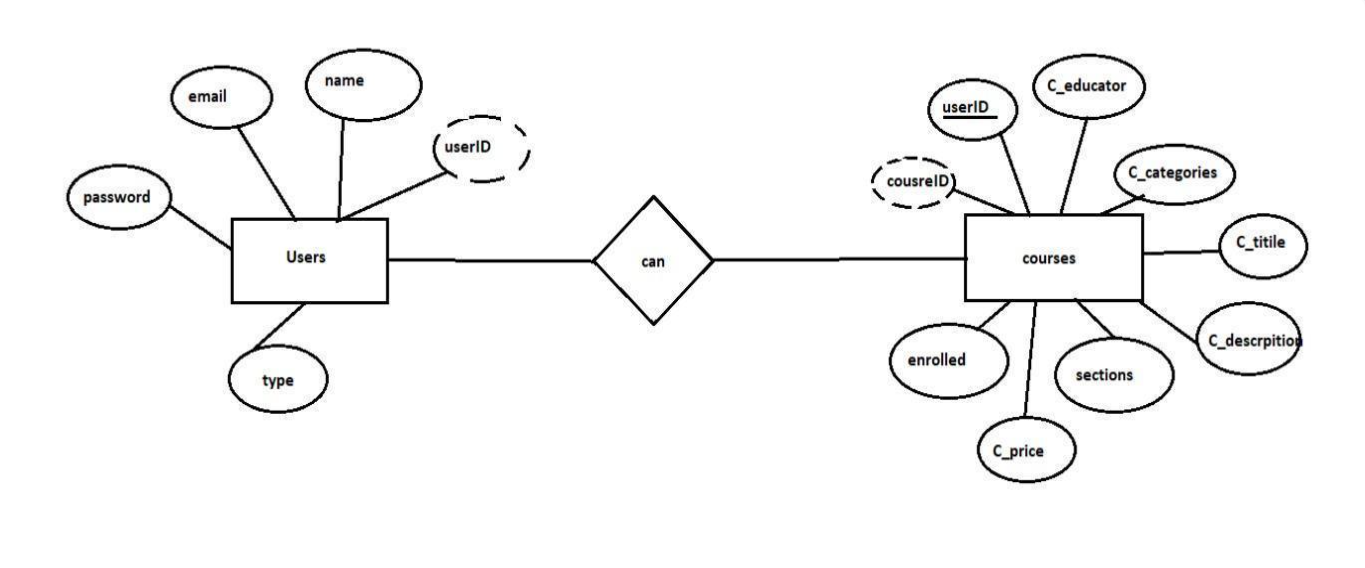
**• Purpose:**

LearnHub is designed to serve as a comprehensive platform for skill development and lifelong learning. Its primary purpose is to bridge the gap between theoretical knowledge and practical application by offering learners hands-on experience through interactive courses and real-world projects. In a rapidly evolving job market, staying relevant requires continuous learning, and LearnHub empowers individuals to upskill, reskill, and stay competitive. Whether you are a student laying the foundation for your career, a working professional looking to advance in your field, or someone exploring new areas of interest, LearnHub provides personalized learning paths tailored to your goals. The platform promotes flexibility and self-paced learning, allowing users to access high-quality content anytime, anywhere. Beyond individual growth, LearnHub fosters an inclusive learning community where users can collaborate, share knowledge, and support each other. By offering expert-led sessions, progress tracking, certification, and career guidance, LearnHub goes beyond traditional education—it becomes a trusted companion on your journey of continuous growth and skill enhancement.

**Features :**

1. **Personalized Learning Paths:**  
   Tailored course recommendations based on users' goals, skill level, and progress.
2. **Diverse Course Catalog:**  
   Wide range of subjects including programming, data science, soft skills, design, and more.
3. **Interactive Content:**  
   Hands-on exercises, quizzes, and real-world projects to reinforce learning.
4. **Progress Tracking:**  
   Visual dashboards that display course completion, skill level advancement, and learning streaks.
5. **Expert-Led Sessions:**  
   Courses and webinars led by industry professionals and experienced educators.
6. **Certification:**  
   Earn shareable certificates upon course or module completion to showcase your skills.
7. **Community Support:**  
   Discussion forums, peer interactions, and study groups for collaborative learning.
8. **Career Assistance:**  
   Resume-building tips, interview prep sessions, and job matching based on learned skills.
9. **Mobile-Friendly:**  
   Access content anytime, anywhere via a responsive web interface or mobile app.
10. **Gamification:**  
    Earn badges, rewards, and ranks to make the learning experience more engaging and fun.

**3. Architecture**

****

**Fig: - Architecture**

**Frontend (React.js)**

The frontend is responsible for the visual representation and user interaction with the LearnHub platform. It includes a user-friendly and responsive interface built using React.js, ensuring seamless navigation and dynamic updates. Key modules include:

* **User Authentication:**Forms for login and registration with role-based access control (learner, instructor, admin). Includes email verification, password reset, and session management.
* **Dashboard:**Personalized dashboards tailored to each role: learners see course progress, enrolled modules, and certificates; instructors manage uploaded courses and student performance; admins monitor platform analytics and user activity.
* **Course Browsing:**Interface for exploring available courses with filters such as category, difficulty level, duration, instructor, and ratings. Includes a search bar and dynamic sorting options.
* **Learning Module Interface:**A dedicated interface to access course content, including video lectures, PDFs, quizzes, and assignments. Tracks progress and allows learners to resume from their last activity.
* **Quiz and Assessment:**Interactive quizzes embedded within each module with automatic grading, answer explanations, and retry options.
* **Certificate Generator:**After completing courses, learners can view and download auto-generated certificates, which include user name, course name, and completion date.
* **User Profile Management:**  
  Users can edit their personal information, upload a profile picture, view achievements, manage enrolled courses, and download certificates.
* **Notifications and Alerts:**Real-time updates using toast messages and alerts for new courses, quiz deadlines, certificate generation, and course updates.
* **Community Forum Access:**A section for learners to participate in discussions, ask questions, and engage with peers and instructors.
* **Responsive Design:**Mobile-first design using Tailwind CSS or Material UI for a consistent experience across smartphones, tablets, and desktops.

**Backend (Node.js + Express.js)**

The backend handles all the server-side operations, providing robust APIs, secure data handling, and business logic management for the LearnHub platform. It is built using **Node.js** and **Express.js**, with MongoDB as the primary database.

* **User Authentication**:  
  Handles user registration and login using JWT (JSON Web Token) for secure authentication. Passwords are hashed using bcrypt before being stored in the database.
* **Role-Based Authorization**:  
  Ensures that learners, instructors, and admins have different access levels. Middleware is used to check tokens and validate permissions for each route.
* **Course Management**:  
  APIs for instructors to create, update, and delete course content, including titles, descriptions, modules, and media. Admins have control over all courses for moderation purposes.
* **Enrollment and Progress Tracking**:  
  Logic to manage course enrollments, track user progress, mark module completion, and update learning history in the database.
* **Assessment & Certificate Management**:  
  Manages quiz submission, result evaluation, and auto-generates certificates upon successful course completion.
* **Learning Analytics & Dashboards**:  
  Provides APIs to deliver insights on user progress, popular courses, completion rates, and activity tracking, viewable by both users and admins.
* **Data Validation & Security**:  
  Uses middleware and validation libraries (like express-validator) to sanitize input. JWT ensures secure API access. Passwords are encrypted, and sensitive operations are protected with authentication checks.
* **Server Configuration**:  
  The backend is initialized with an index.js file, and configured with essential middleware such as **CORS**, **body-parser**, and **dotenv** to manage environment variables (like MongoDB URI, PORT, JWT\_SECRET).
* **Middleware Integration**:  
  A custom authMiddleware.js is created inside a middleware folder to verify tokens and protect restricted routes.
* **Error Handling & Logging**:  
  Global error handling and request logging mechanisms ensure smooth debugging and error reporting across the backend.

**Database (MongoDB)**

The database layer is responsible for storing and managing all persistent data of the LearnHub platform. MongoDB, a flexible NoSQL database, is used for its schema-less nature and scalability, allowing efficient handling of structured and semi-structured educational data.

**Database Collections**

* **Users:**  
  Stores user details such as full name, email, encrypted password, role (learner, instructor, admin), enrollment history, and certificate data.
* **Courses:**  
  Contains course metadata including title, description, instructor ID, category, level, duration, modules, rating, price (if any), and publishing status.
* **Modules:**Stores individual course module content such as titles, video URLs, PDF links, quiz references, completion status, and timestamps.
* **Quizzes:**  
  Holds quiz questions, options, correct answers, explanations, and course/module linkage for each test.
* **Certificates:**Maintains data on issued certificates including user ID, course ID, completion date, and certificate number.
* **Enrollments:**Tracks users enrolled in each course, along with progress percentage, date of enrollment, and completion status.
* **Feedback &Ratings:**Collection to store user feedback, star ratings, comments, and review timestamps for each course or module.
* **Notifications:**Saves alerts, reminders, and system announcements for each user based on their activity or role.

**• Installation:**

* React
* React DOM
* React Router DOM
* Axios
* Bootstrap
* Material UI (MUI)
* React Bootstrap
* Moment.js
* Express
* Mongoose
* Dotenv
* Bcryptjs
* Jsonwebtoken (JWT)
* CORS
* Body-Parser
* Nodemon

**5. Folder Structure**

This project follows a standard **MERN stack** (MongoDB, Express.js, React.js, Node.js) architecture, organized into **Client** and **Server** folders to separate the frontend and backend concerns.

* **Client:**

This contains the **ReactJS frontend** application.

**• public/**

Contains static files and assets used in the React app.

* index.html: The root HTML file where the React app mounts.
* favicon.ico: The icon shown in the browser tab.
* logo192.png, logo512.png: Icons used in Progressive Web App (PWA) mode.
* manifest.json: Metadata about the app (name, theme, icons, etc.) for PWA support.
* robots.txt: SEO file used to guide search engine indexing.

**• src/**

This is the source code for the React application.

* **App.jsx / main.jsx**: Entry point of the application; includes routing and main rendering logic.
* **App.css**: Styling for global components.
* **components/**:  
  Contains reusable and role-based UI components:
  + admin/: Components used by the Admin (e.g., AdminHome.jsx, AllCourses.jsx).
  + common/: Shared components like Login, Register, Home, NavBar, Dashboard, etc.
  + user/: User-specific components subdivided by role:
    - student/: Contains StudentHome.jsx, CourseContent.jsx, EnrolledCourses.jsx.
    - teacher/: Contains TeacherHome.jsx, AddCourse.jsx.
* **assets/**: Placeholder for images, icons, or other static files (if applicable).
* **AxiosInstance.jsx**: Axios setup for making API calls with common configuration.

**• Project Metadata:**

* package.json: Lists dependencies, scripts (start, build, etc.), and basic project metadata.
* package-lock.json: Provides an exact version tree of installed npm packages to ensure consistent installs.

**Server:**

This contains the **Node.js + Express backend** application.

**• config/**

Configuration files for environment settings, database connections, etc.

**• controllers/**

Contains logic for handling HTTP requests and sending responses:

* adminController.js, userControllers.js: Define API functionality for different roles.

**• middlewares/**

Custom middleware functions:

* authMiddleware.js: Middleware for authentication/authorization.

**• routers/**

Defines API routes and links them to controllers:

* adminRoutes.js, userRoutes.js: Handle HTTP requests to specific endpoints.

**• schemas/**

Mongoose schemas for MongoDB collections:

* userModel.js: User schema.
* courseModel.js, coursePaymentModel.js, enrolledCourseModel.js: Related to courses and enrollments.

**• uploads/**

Folder for storing user-uploaded files (if any).

**• Root files:**

* .env: Environment variables.
* .gitignore: Files/folders to be ignored by Git.
* index.js: Entry point of the Node.js backend server.
* package.json: Lists backend dependencies and server scripts.
* package-lock.json: Locks specific package versions.

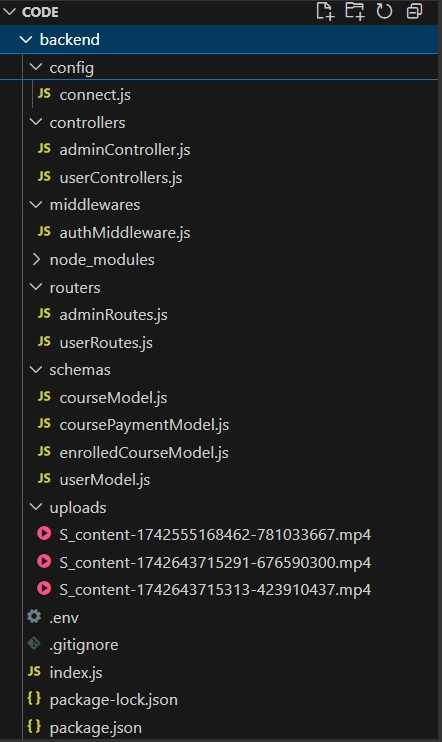
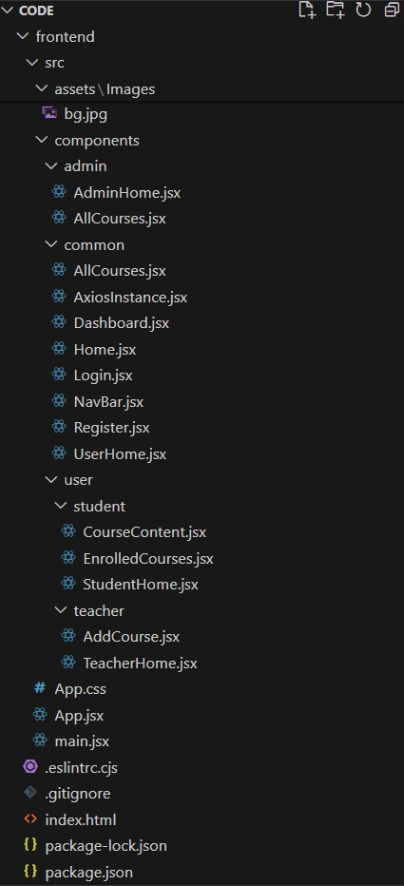


Fig: - Backend Fig: - Frontend

**6. Running the Application**

**Frontend:** npm start -in the client directory  **Backend:** npm run debug- in the server directory.

**7. 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:**

* Can add courses for the student.
* Also delete the course if no student enrolled in it or any other reasons.
* Also add sections to courses.

**Student:**

* Can enroll in an individual or multiple course.
* Can start the course where it has stopped.
* Once the course is completed, they can download their certificate of completeion of the course.
* For paid course, they need to purchase it and then they can start the course.
* They can filter out the course by searching by name, category, etc

**Admin:**

* They can alter all the course that are present in the app.
* Watch out all kind of users in app.
* Record all the enrolled all the student that are enrolled in course.

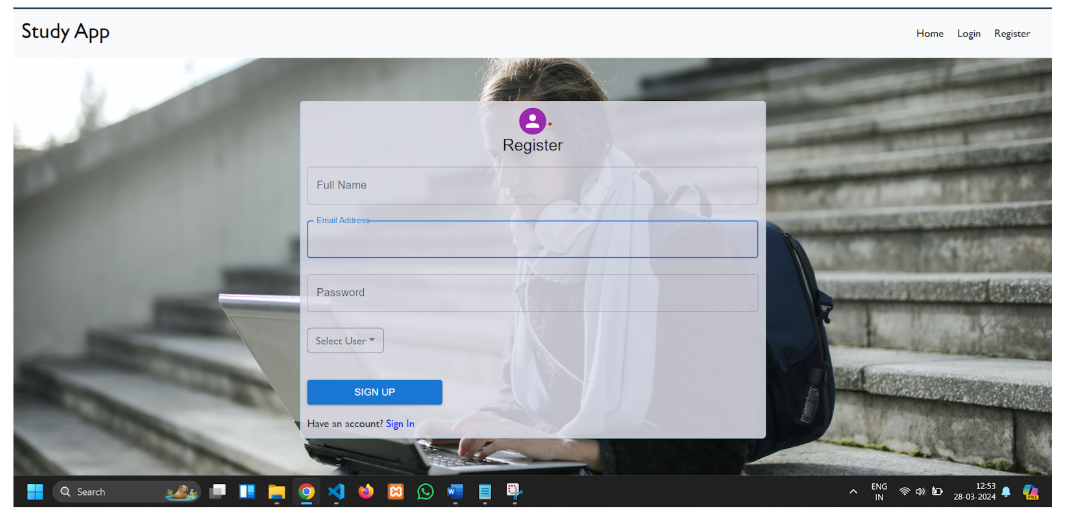
**8. Authentication:**

LearnHub ensures a secure and role-specific experience for all users—whether you're an admin, instructor, or learner—by implementing robust authentication and authorization mechanisms. Here's how security is handled in the platform:

* **Authentication**
* **User Registration:**  
  New users sign up by providing a **username, email, password**, and selecting a **role** (Admin, Instructor, or Learner).
* **Secure Password Storage:**  
  Passwords are **hashed** before being stored in **MongoDB**, ensuring they are never saved in plain text.
* **User Login:**  
  Upon submitting valid credentials, a **JWT (JSON Web Token)** is generated on the server and sent to the client.
* **Token Storage:**  
  The JWT is securely stored on the client-side, typically in **localStorage**, enabling seamless access to protected resources in future sessions.
* **Authorization**
* **Role-Based Access Control:**  
  Every user is assigned a role upon registration (e.g., **Admin, Instructor, Learner**). Each role has specific access rights within LearnHub.
* **Protected Routes & Features:**  
  Access to certain routes and actions (like **course creation**, **user management**, or **dashboard access**) is restricted based on user role. For instance:
  + **Admins** manage platform settings and users.
  + **Instructors** can create and manage course content.
  + **Learners** can access enrolled courses and track progress.
* **Middleware Validation:**  
  All protected actions pass through middleware that verifies user identity and role before granting access.
* **Token Management**
* **Token Generation:**  
  JWTs are created using the user's **ID, assigned role**, and an **expiration time**, ensuring secure and temporary access.
* **Client-Side Token Storage:**  
  Tokens are stored on the frontend (typically **localStorage**) and attached to each request for authentication.
* **Token Validation:**  
  Each incoming request is validated on the backend via middleware to ensure the token is authentic and not expired.

**9. User Interface**

• Provide screenshots showcasing login page and registration form.



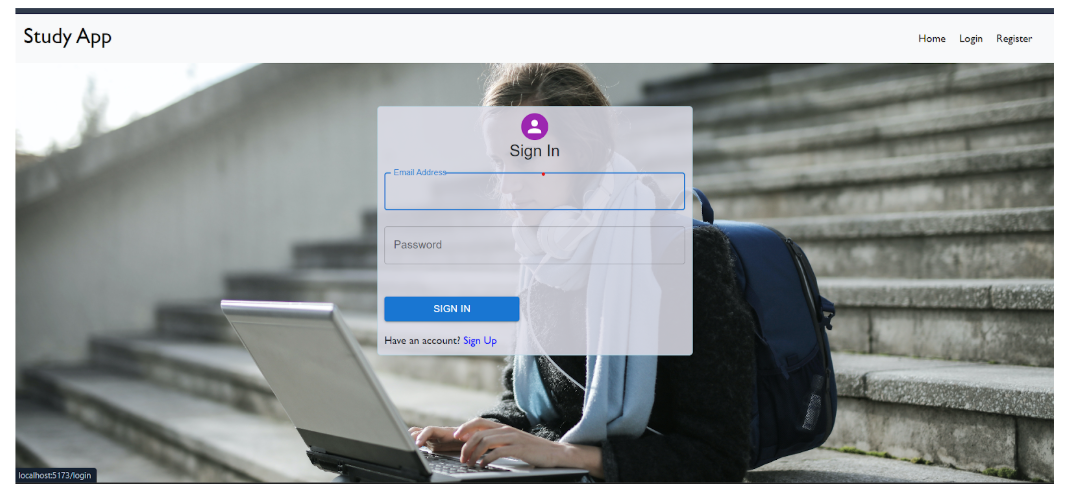
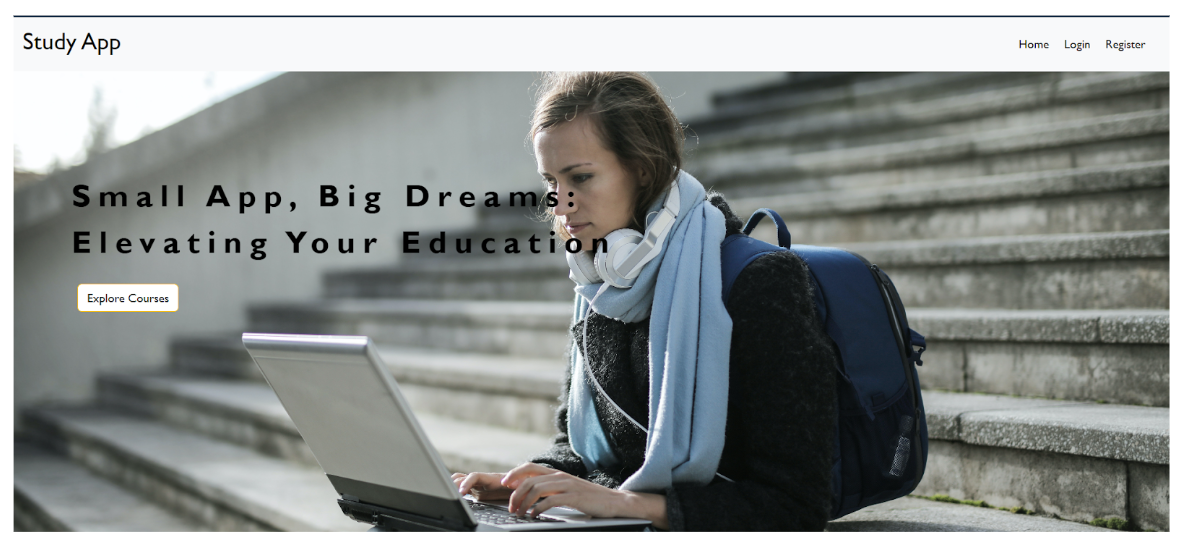
 Fig: - Register Form

Fig: - Login Page

**10.Testing:**

* **API Testing:**  
  Use tools like **Postman** or **Insomnia** to test authentication endpoints (/register, /login) and protected routes with and without **JWT tokens**.
* **Unit Testing:**  
  Use Jest or Mocha to write test cases for individual functions like password hashing, token generation, and role checks.
* **Integration Testing:**  
  Test complete flows (e.g., user registration to course enrollment) using Supertest with Express to ensure modules work together correctly.
* **Frontend Testing:**  
  Use React Testing Library or Jest to test form validations, component rendering, and user interactions.

**11. Screenshots or Demo**

 **Project Structure**

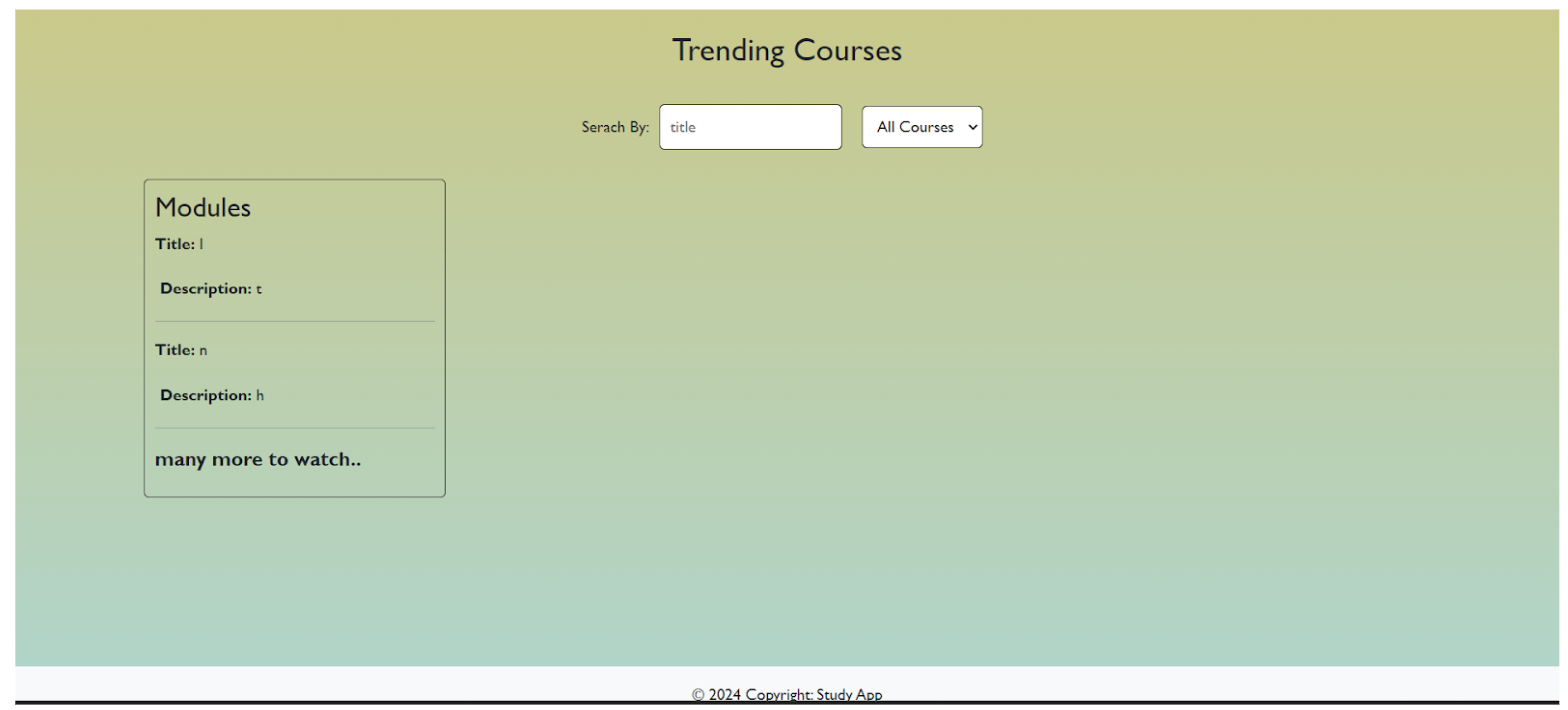


Fig: -Landing page

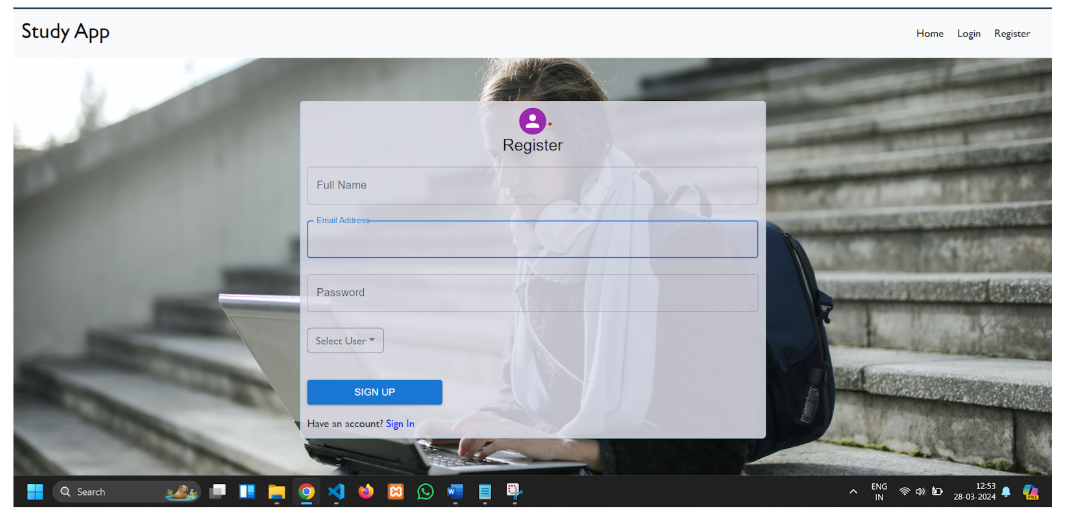


Fig: - Registration Form

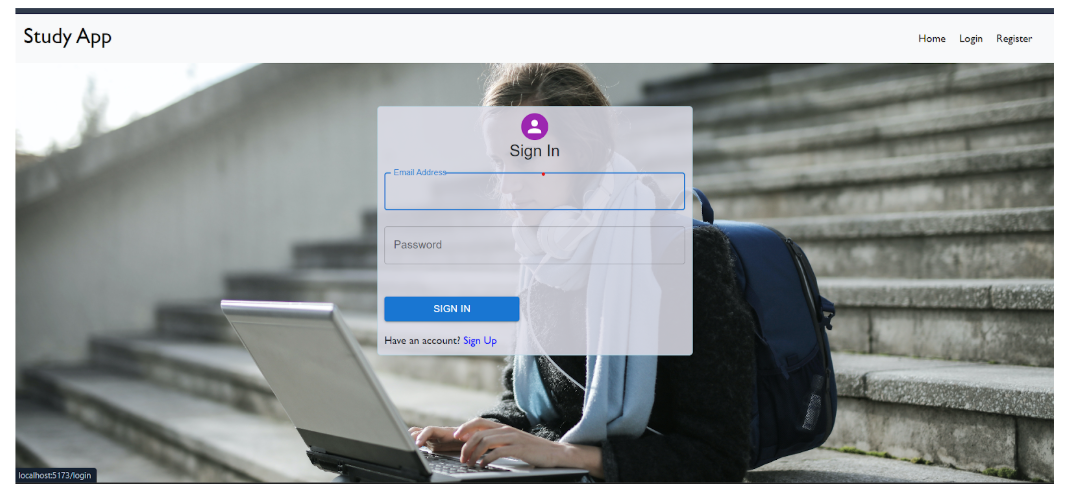


Fig: - Login Page

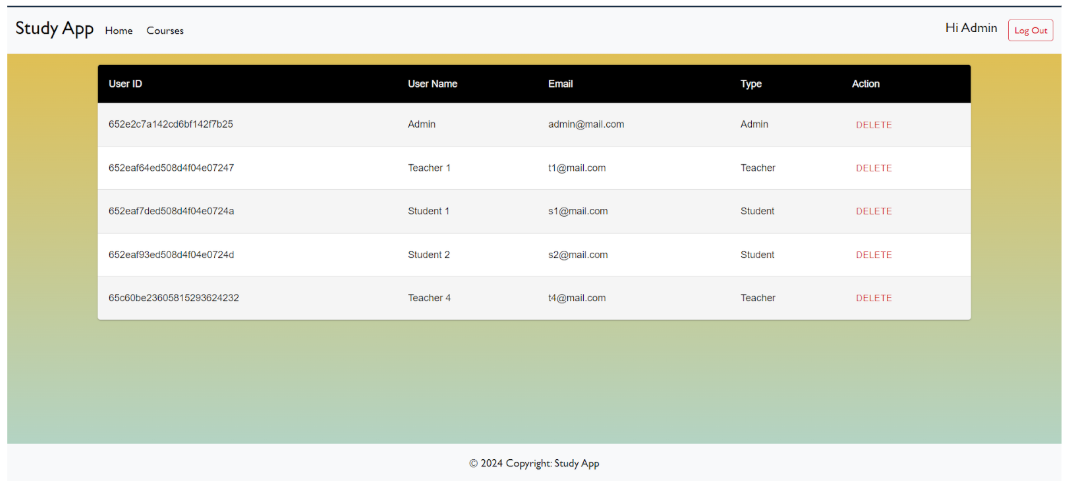


Fig: - Admin Dashboard

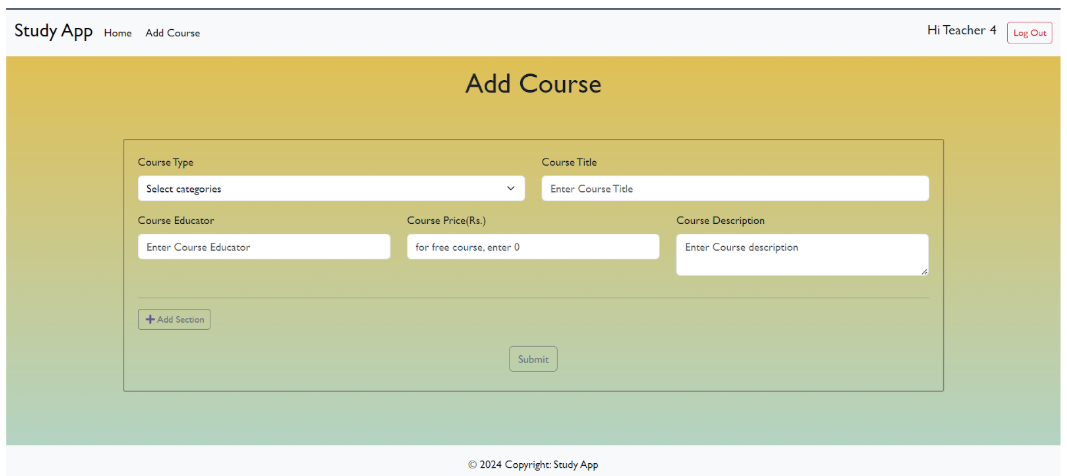
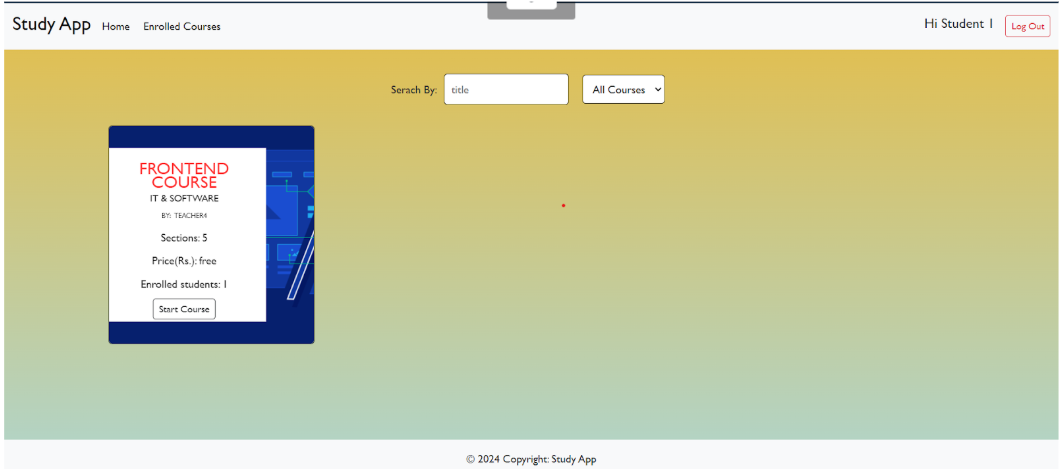


Fig: - Teacher Dashboard



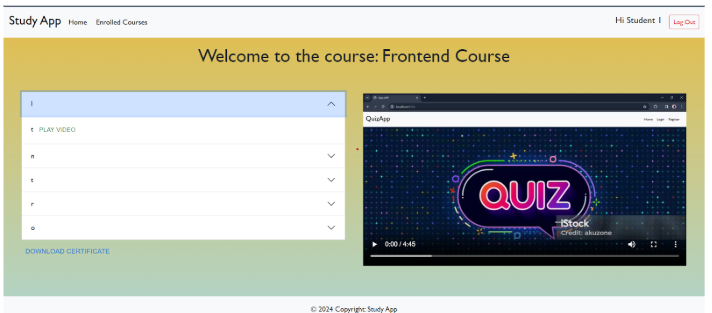


Fig: - Student Dashboard

**12. Known Issues**

The following are known issues identified during development and testing. These may affect users or developers and are being tracked for future resolution:

**• Backend (Node.js):**

* **Error Handling:**  
  Uncaught exceptions may cause server crashes in edge cases.
* **MongoDB Connection Issues:**  
  Intermittent disconnections may require a manual server restart.
* **API Rate Limiting:**  
  Currently not implemented, making the backend vulnerable to excessive requests or abuse.

**• Frontend (React):**

* **Browser Compatibility:**  
  Display issues may occur in older browsers like Internet Explorer and certain versions of Safari.
* **Mobile Responsiveness:**  
  Layout inconsistencies on smaller screen sizes, especially in complex views.
* **State Management:**  
  Complex or asynchronous state updates may occasionally cause rendering glitches.

**• Database (MongoDB):**

* **Data Consistency:**  
  Absence of transaction handling can lead to inconsistent data in multi-step operations.
* **Query Optimization:**  
  Some database queries are not optimized, leading to slower performance.
* **Indexing:**  
  Missing indexes on large collections can result in longer query times.

**• Security:**

* **Authentication:**Current use of BCrypt is adequate but could be enhanced with stronger salting and iteration rounds.
* **Authorization:**Role-based access control is implemented but lacks full coverage in some routes.
* **XSS Protection:**  
  Limited input sanitization may leave the frontend vulnerable to cross-site scripting (XSS) attacks.

**13. Future Enhancements**

* **Extend schema:**  
  Include models for Course Progress, Enrollments, and Advanced User Roles (Admin, Instructor, Learner).
* **Implement pagination and lazy loading:**  
  Improve performance for both frontend and backend when handling large datasets (e.g., courses, users).
* **Use React Query or SWR:**  
  Enable efficient API calls, data caching, and automatic refetching in the frontend.
* **Secure inputs and hash passwords:**  
  Strengthen security using bcrypt for password hashing and apply input sanitization to prevent injection attacks.
* **Modularize codebase and improve deployment:**  
  Refactor for maintainability, integrate Docker, and set up CI/CD pipelines for automated testing and deployment.