**Full Stack Development with MERN**

**1. Introduction**

* **Project Title**: Learn Hub – Online Learning Platform
* **Team ID**: LTVIP2025TMID45942
* **Team Members**:
  + **Devendra Bethireddy** – Team Leader
  + **Boppadapu Karthik** – Developer
  + **Botu Sreya** – Frontend & Documentation Lead
  + **Battula Likhitha** – Testing & Deployment

**2. Project Overview**

* **Purpose**:  
  To create an interactive, user-friendly online platform where teachers can upload educational content, students can enroll and track their progress, and admins can oversee the overall functionality.
* **Features**:
  + Role-based login (Teacher, Student, Admin)
  + Course creation with video/image uploads
  + Student enrollment & mock payment system
  + Course progress tracking
  + Admin dashboard for user/course management

**3. Architecture**

* **Frontend (React.js)**:
  + Component-based design
  + React Router for navigation
  + Axios for API calls
  + Tailwind CSS for styling
* **Backend (Node.js + Express.js)**:
  + RESTful API structure
  + Controllers for handling business logic
  + Multer + Cloudinary for file uploads
* **Database (MongoDB)**:
  + Collections: Users, Courses, Enrollments, Progress
  + Mongoose for schema and validation

**4. Setup Instructions**

* **Prerequisites**:
  + Node.js v18+
  + MongoDB (local or Atlas)
  + Cloudinary Account for media upload
  + Vercel/Render for deployment
* **Installation**:
* git clone https://github.com/botusreya/learn-hub.git
* cd learn-hub
* cd frontend && npm install
* cd ../backend && npm install
* **Environment Variables**:
  + Create .env in backend folder with:
  + MONGO\_URI=your\_mongodb\_uri
  + CLOUDINARY\_NAME=your\_cloud\_name
  + CLOUDINARY\_KEY=your\_api\_key
  + CLOUDINARY\_SECRET=your\_api\_secret
  + JWT\_SECRET=your\_jwt

**5. Folder Structure**

* **Client (Frontend)**:
* /src
* /components
* /user
* /admin
* /teacher
* /pages
* /services
* /App.js
* /index.js
* **Server (Backend)**:
* /controllers
* /routes
* /models
* /middleware
* /config
* /server.js

**6. Running the Application**

* **Frontend**:

cd frontend

npm start

* **Backend**:

cd backend

npm start

**7. API Documentation**

| **Endpoint** | **Method** | **Description** |
| --- | --- | --- |
| /api/user/register | POST | User Registration |
| /api/user/login | POST | User Login |
| /api/user/addcourse | POST | Teacher adds a course |
| /api/user/getallcoursesteacher | GET | Get teacher’s courses |
| /api/user/enrollcourse/:id | POST | Student enrolls in a course |
| /api/admin/getallusers | GET | Admin views users |

*(Add more with request/response formats if needed)*

**8. Authentication**

* Implemented using **JWT (JSON Web Token)**
* Tokens stored in localStorage on login
* Protected routes using middleware (authMiddleware.js)

**9. User Interface**

* Clean UI built with **Tailwind CSS**
* Role-based rendering
* Responsive layouts for dashboard, course cards, progress tracker

**10. Testing**

* APIs tested using **Postman**
* Functional testing for course upload, progress tracking, enrollment
* Manual UAT (User Acceptance Testing) done

**12. Known Issues**

* Refresh required after course creation to reflect immediately
* Mock payment UI needs polish
* Section progress may not persist under slow connections

**13. Future Enhancements**

* Real payment integration (Stripe/Razorpay)
* Mobile App (React Native)
* AI-driven course recommendations
* Certificate generation after course completion
* Live classes integration (Zoom/Google Meet API)