WORKFLOW

PROJECT ID – 20

Online Course Portal – Mern Stack

Problem statement Idea:

Develop a web application that allows users to browse, search, and enroll in online courses on task based topics. The platform should have the following features User authentication, Course creation and management, User reviews and ratings, Output visualization, Progress Tracking Task Submission along with Video Submission Instructors Chat Board The application should be responsive, user-friendly, and scalable to accommodate a large number of users and courses.

Project idea:

The project idea is to create an online portal for students and instructors to interact, create task-based courses, enroll in courses, and have a great hands-on learning experience with standard validation techniques.

Workflow:

1. **User Registration:** Students and instructors can sign up using their BIT mail id credentials.
2. **Profile Creation:**
    a. **For Students:** Username, Area of interest, department, Linkedin Id.
    b. **For instructors:** Username, department, professor category, domain.
3. **Home page:** In landing page, there will floating images about XR studios creations, projects, hackathons with small description about them.
4. Upon further scrolling About XR studios should get displayed and their vision and mission should get displayed / these things can be displayed in the about page in navbar upon clicking.
5. **Admin portal:** For this whole online portal admin management should be created. He should manage all the instructors. Upon successful registration as an instructor admin should approve their access to upload videos for the task-based courses. He should be able to see any students progress, and their submitted tasks.
6. **Course enrolment:** Students can enroll in their desired courses upon filling basic details. After successful enrolment they will be given with a deadline for the course completion and every individual videos and tasks. They are supposed to complete that within the give time limit (by instructor or admin) for that particular tasks and course.
7. **Student instructor interaction:** An online messaging portal should be created for encouraging an healthy interaction between student and instructor and the message history should be saved and should not be able to get deleted and admin should have the access to read all the messages.
8. **Feedback and Improvements:** Regular feedback collection from users to enhance the platform's features and user experience.

Implementation:

1. **Technical Development:** Design and develop the online course portal using web development technologies like HTML, CSS, JavaScript, and backend frameworks like Node.js.
2. **Database Management:** Set up a database to store user profiles, job postings, event information, messages, and other relevant data using tools like MongoDB.
3. **User Authentication**: Implement a secure user authentication system using encryption techniques and frameworks like OAuth to ensure user data protection.
4. **Networking Features:** Develop algorithms to suggest connections based on user profiles, implement search filters for users to find connections, and create a system for sending and accepting connection requests.
5. **Task and deadline intimation:** Alert system for upcoming tasks and dealine and notification system for the new courses added and remarks given for the completed tasks.
6. **Messaging System:** Develop a real-time messaging system for users to communicate with instructors/admin.
7. **Feedback Mechanism:** Include feedback forms, surveys, or ratings to gather user input for continuous improvement and to tailor features according to user needs and preferences.
8. **Testing and Deployment:** Conduct thorough testing to ensure the platform's functionality and user experience meet expectations before deploying it for use by the BIT community.

Design and Requirements:

Design responsive and user-friendly layouts for the following features: (Student Login)

- User authentication (login, registration, and profile management)
- Course browsing and search functionality
- Course enrollment
- User reviews and ratings
- Output visualization (charts, graphs, etc.)
- Progress tracking (course completion status)
- Task submission (uploading assignments or project work)
- Video submission (if applicable)
- Student chat board for student-instructor interaction

Implement the following backend functionalities:(Instructor Login)

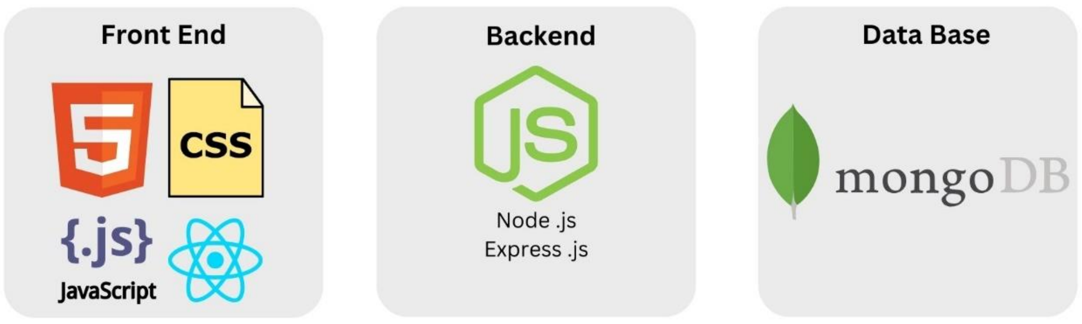
a. User authentication
b. Course creation and management
c. User reviews and ratings (store and retrieve user feedback)
d. Task validation (store uploaded files or links)
e. Video submission (if applicable)
f. Instructor chat board (real-time communication)

Technical Workflow:

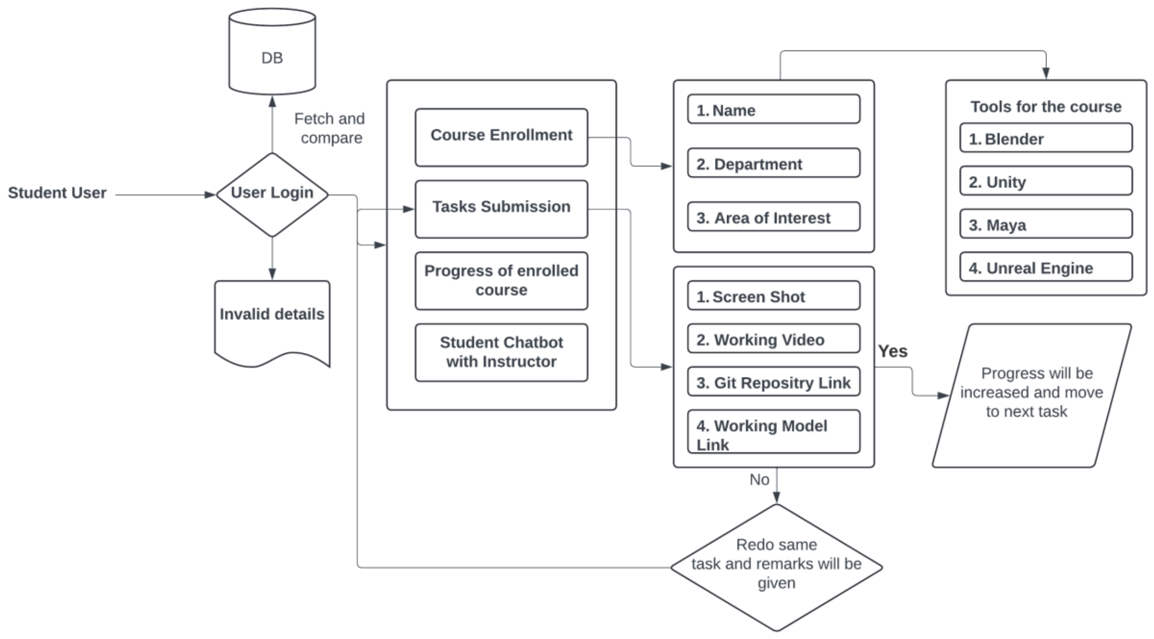
1. **Frontend Development**:
   - Use HTML, CSS, and JavaScript along with React (a frontend framework) to create the user interface.
2. **Backend Development with Express.js and Node.js:**

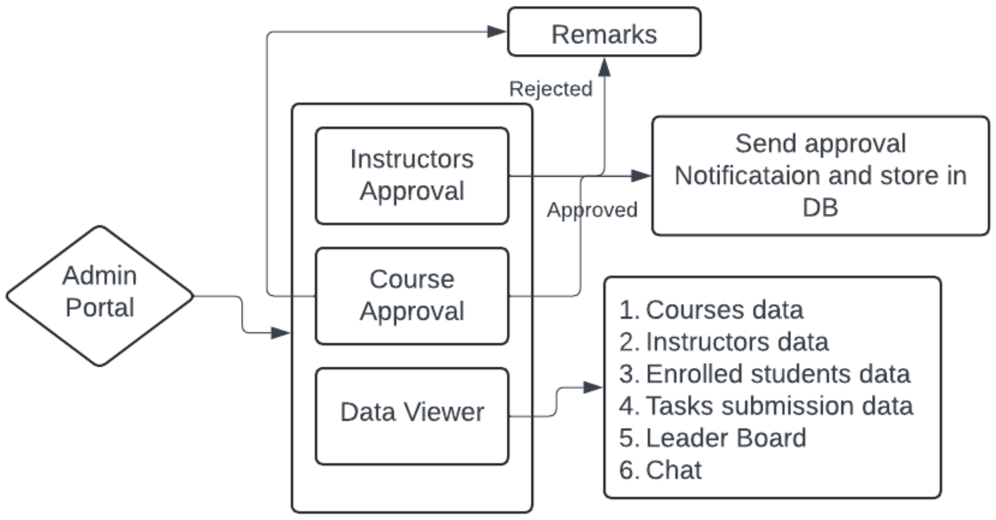
        - Set up a React project to handle backend logic, data processing, and API creation.
        - Use Node.js (with Express.js) as the backend server.

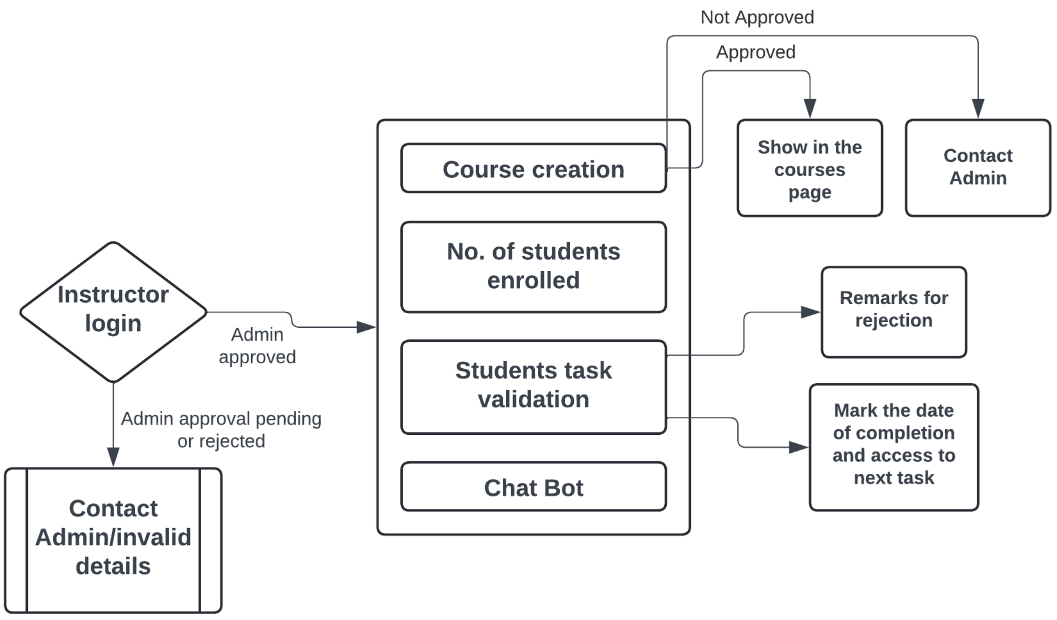
3. **Database Integration:**
    a. Configure React to connect to a MongoDB database using Mongoose (an ODM library).
    b. Design the database schema to store information related to users, courses, reviews, tasks, and chat messages.
4. **User Authentication and Authorization:**
    a. Implement authentication using O Auth to secure routes.
    b. Define user roles (student, instructor, admin) and manage access based on roles.
5. **RESTful API Development:**
    a. Create RESTful APIs in Express.js to handle CRUD (create, read, update and delete) operations for user profiles, courses, reviews, tasks, and chat messages.
    b. Ensure proper validation and error handling.
6. **Frontend-Backend Integration:**
    a. Establish communication between the frontend and backend using HTTP requests to fetch and update data.
    b. Implement API endpoints for user registration, login, course management, reviews, and chat messages.
7. **Testing/Debug Phase:**
    a. Conduct unit tests for backend logic using tools like Jest or Mocha.
    b. Perform integration tests to validate the interaction between frontend and backend components.
8. **Deployment:**
    a. Deploy the application on a cloud platform (e.g., Heroku, AWS, or Azure) to make it accessible to users.
    b. Monitor performance, handle scalability, and gather user feedback for improvements.



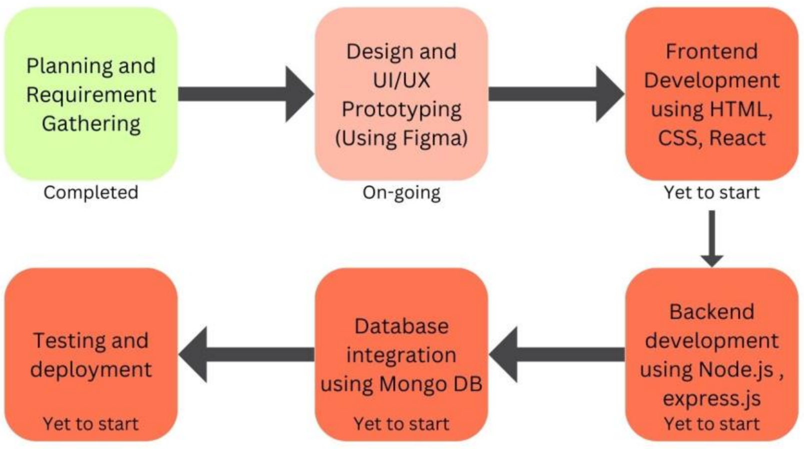
Workflow Diagram:







Development work flow



Planning and Requirement Gathering
Completed
Design and UI/UX Prototyping (Using Figma)
On-going
Frontend Development using HTML, CSS, React
Yet to start
Backend development using Node.js , express.js
Yet to start
Database integration using Mongo DB
Yet to start
Testing and deployment
Yet to start