Web Development
Group Project
Team 4

Online Learning Platform- Learny

Sunera Samuditha - PM, testing, documentation
Dilum Piyumantha - Frontend
Achintha Dilhara - Backend
Nethmi Senarathna - Backend
Chenuli Ranasinghe - Frontend

What is an Online Learning Platform?

An **online learning platform** is a digital space where students and educators interact for educational purposes. These platforms typically include the following features:

- **User Authentication**: Login/signup functionality for both students and teachers.
- **Course Creation & Management**: Allows instructors to create, upload, and manage course materials like videos, text, and assignments.
- **Content Delivery**: Students can access lectures, assignments, quizzes, and other materials.
- **Tracking Progress**: Students can track their learning progress, and teachers can evaluate their performance.
- **Interaction & Engagement**: Features like discussion forums, quizzes, messaging, and feedback.

Technologies, Resources, Platforms, and Software

Building an online learning platform involves using various **front-end**, **back-end**, and **database** technologies, along with additional tools for **deployment**, **collaboration**, and **testing**. Below is a list of common technologies and platforms used for each aspect.

1. Frontend Technologies (User Interface)

- **HTML5/CSS3**: For structuring and styling web pages.
- **JavaScript**: For dynamic content and interactivity.
- **Frameworks**:
    - **React.js**: A popular framework for building fast and dynamic single-page applications (SPAs).
    - **Angular** (alternatives if the team prefers).
- **Bootstrap**: For rapid styling and responsiveness.

2. Backend Technologies (Server Logic)

- **Node.js** (with Express): A popular JavaScript runtime environment for building scalable backend applications.
- **RESTful APIs**: For communication between frontend and backend.
- **Axios or Fetch API**: For making HTTP requests to backend APIs.

3. Database Technologies (Data Storage)

- **MongoDB** (NoSQL): Great for flexible and scalable document-based storage.
- **MySQL** or **PostgreSQL** (SQL): Relational databases for structured data (tables, relations).

4. Authentication and Authorization

- **JWT (JSON Web Tokens)**: For user authentication.
- **OAuth**: For third-party login integrations (e.g., Google, Facebook).

5. Cloud Storage and File Handling

- **AWS S3** or **Google Cloud Storage**: For storing large files (course videos, images, PDFs).
- **Cloudinary**: A media management service for image and video handling.

6. Payment Integration (Optional)

- **Stripe** or **PayPal APIs**: For integrating payment systems if courses are paid.

7. Development & Collaboration Tools

- **GitHub** : For version control and collaboration.
- **Trello** : For task management and tracking.
- **Whatsapp**: For team communication.
- **Postman**: For testing backend APIs.

8. Deployment Platforms

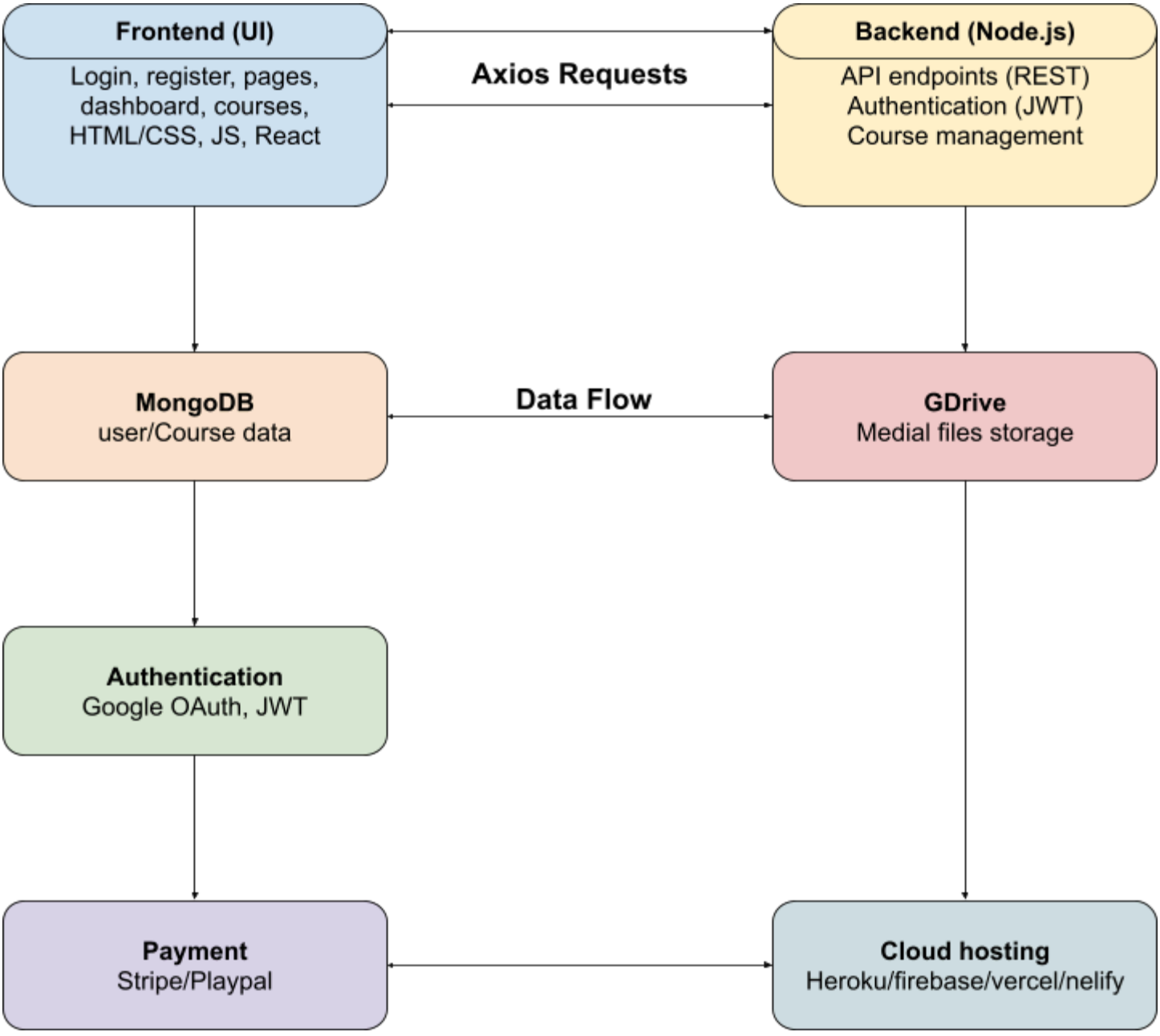
- **Heroku**: For deploying backend applications.
- **Netlify** or **Vercel**: For frontend deployment.
- **Docker**: For containerizing the application and simplifying the deployment process.

9. Testing Tools

- **Jest**: For JavaScript unit testing.
- **Mocha**: For backend testing.
- **Selenium** or **Cypress**: For automated end-to-end testing.

10. Additional Tools

- **Google Analytics**: For tracking user activity and metrics.
- **Firebase**: As an alternative for database, hosting, and authentication.


Frontend (UI)
Login, register, pages, dashboard, courses, HTML/CSS, JS, React
Axios Requests
Backend (Node.js)
API endpoints (REST)
Authentication (JWT)
Course management
MongoDB
user/Course data
Data Flow
GDrive
Medial files storage
Authentication
Google OAuth, JWT
Payment
Stripe/Playpal
Cloud hosting
Heroku/firebase/vercel/nelify

How the System Works:

1. **Frontend** (User Interaction):
    - **User Interface (UI)**: Built using **React.js**, **HTML/CSS**, and **Bootstrap**, the frontend serves the user with pages such as login, course catalog, dashboard, etc.
    - **Axios** is used to send HTTP requests to the backend to retrieve or send data.

2. **Backend** (Business Logic):
    - The backend is powered by **Node.js** (with **Express**) or **Django** to handle logic such as user registration, login, and course management.
    - **RESTful APIs** allow the frontend to communicate with the backend. This ensures a modular and scalable approach.
    - **JWT** is used to handle user authentication securely, with token-based verification.

3. **Database** (Data Storage):
    - **MongoDB** or **MySQL** stores user data (such as profiles, courses, and progress).
    - The backend queries this database and sends the data to the frontend as needed.

4. **Cloud Storage** (For Media):
    - Course materials like video lectures, PDFs, or images are uploaded and stored using cloud services like **AWS S3** or **Google Cloud Storage**.
    - **Cloudinary** can be used to manage and optimize media files.

5. **Authentication and Authorization**:
    - Users can register and log in via a custom system using **JWT**, or you can integrate **OAuth** to allow login through Google or Facebook.

6. **Payment System (Optional)**:
    - If the platform supports paid courses, APIs from **Stripe** or **PayPal** can be integrated to handle payments.

7. **Deployment and Hosting**:
    - Once the frontend and backend are developed, they are deployed using platforms like **Heroku** (for backend) and **Netlify** or **Vercel** (for frontend).
    - These platforms automatically handle hosting, server management, and scaling.