**The Background**

MicroCourses is a fictional educational platform offering a diverse range of online courses to learners worldwide. As the platform experiences growth, there's a pressing need for a modern, scalable, and responsive web application that effectively showcases their course offerings. This project is part of a micro-credential assessment in back-end web development, designed to evaluate students' skills in the MERN (MongoDB, Express.js, React, Node.js) stack, as well as their ability to implement advanced web development concepts such as caching, load balancing, and high availability.

The platform aims to provide an intuitive and efficient user experience for browsing courses, accessing detailed course information, and potentially enrolling in courses. As the user base expands, the application must be capable of handling increased traffic and data volume while maintaining performance and reliability.

**Role and Tasks**

As a full-stack web developer for MicroCourses, your role is to create a responsive, scalable, and robust web application that showcases the platform's course offerings and handles back-end operations efficiently. Your main tasks include:

1. Develop a responsive React-based front-end that displays a list of available courses on the home page and provides detailed views for individual courses, ensuring optimal viewing across various devices (desktop, tablet, mobile).
2. Create a Node.js and Express.js back-end that handles API requests, interacts with the MongoDB database, and serves data to the front-end efficiently.
3. Design and implement a MongoDB database schema to store course information, ensuring it can accommodate future growth and additional features.
4. Implement RESTful API endpoints for retrieving course listings and individual course details, as well as adding new courses to the database.
5. Integrate the React front-end with the Node.js back-end, ensuring smooth data flow and optimal performance.
6. Develop and implement a caching strategy to improve performance, especially when dealing with large numbers of courses.
7. Design and describe a load balancing solution to distribute incoming requests across multiple server instances, enhancing the application's ability to handle high traffic.
8. Plan and outline a high availability strategy for both the back-end application and the MongoDB database to ensure continuous operation, even in disaster scenarios.
9. Apply best practices in code organization, error handling, and security throughout the full-stack application.
10. Create comprehensive documentation, including a README file with project overview, setup instructions, and explanations of implemented advanced features (caching, load balancing, high availability).
11. Prepare a short video demonstration showcasing the application's functionality, performance improvements through caching, and the conceptual implementation of load balancing.
12. Utilise Git for version control and maintain a well-organised GitHub repository for the project.

This role requires a strong understanding of the MERN stack, as well as the ability to implement and describe advanced web development concepts to create a scalable, high-performance educational platform or follow any appropriate instructions that your educator has asked you to follow.

The project consists of following 4 components:

**Component 1:**Front-end Development (React) - Submit code files

**Component 2:**Back-end Development (Node.js & Express) - Submit code files

**Component 3:**Client-Side Integration - Submit code files

**Component 4:**Advanced Functionality - Submit Project Reflection Document

**Component 1**

**Submit Home Page & Detailed Page:                          Submit Code Files**

* **Create a home page that displays a list of courses with:**
  + An image representing the course.
  + The course title.
  + A brief description.
  + Hours to complete.
  + A button or link to view more details about the course.
* **Create a detailed view page for each course with**:
  + The course title.
  + An image representing the course.
  + A detailed description.
  + A list of modules or lessons included in the course.
  + An enrollment button.
* **Guidelines**:
  + Use Create React App to set up a new React project.
  + Ensure the page is fully responsive, with a mobile-first design approach.
  + Utilize Flexbox or CSS Grid for layout.
  + Use React components to make it reusable.
  + Setup routing via app.js.

**Component 3**

**Client-Side Integration:                                             Submit Code Files**

* **React Application**:
  + Set up a React application to interact with your backend.
  + Implement a course listing page where users can see all courses.
  + Implement a course detail page where users can view more information about a specific course.

**Component 2**

**Back-end Development (Node.js & Express)            Submit Code Files**

* **Project Initialisation**:
  + Initialise a new Node.js project.
  + Set up an Express server.
  + Setup and connect to a MongoDB database.
* **Data Models**:
  + Design a data model for courses with fields: title, description, instructor, duration, and category.
* **API Endpoints**:
  + GET /courses: Retrieve a list of all courses.
  + GET /courses/:id: Retrieve detailed information about a specific course.
  + POST /courses: Add a new course.

**Component 4**

**Advanced Functionality (Report)                             Submit Code Files**

* **Caching**:
  + Report on how would you Implement caching strategies to handle a large number of courses efficiently.
  + Plan for cache invalidation.
* **Load Balancing**:
  + Report on how would you distribute incoming requests across multiple instances of your Express server.
* **High Availability**:
  + Report on how would you ensure high availability for your backend application, especially in disaster scenarios.
  + Report on how would you implement high availability for the MongoDB database in a production environment.

**Code Submission**

Along with your reflective document, submit the code files that were integral to your project. These scripts serve as tangible evidence of your practical skills.

**Code Overview:**Provide a brief explanation of each code file's purpose and how it fits into the project.

**Technical Details:**Describe the functionality of each code file, including any parameters, input data, and the expected output. This section should give insight into how each code file contributes to the project's objectives.

**Problem-solving:**Discuss any challenges faced during code development and deployment, detailing the solutions you implemented.

**Achievements:**Highlight the effectiveness of the code files in achieving project goals, focusing on any optimizations or efficiencies gained.

**Submission Guidelines**

* **Code Repository or any other submission details that your educator has provided:**
  + Create a GitHub repository for your project.
  + Ensure your code is well-organized and follows best practices.
* **Documentation:**
  + **Provide a README file that includes:**
    - An overview of your project.
    - Instructions on how to set up and run the project locally.
    - A brief explanation of how you plan to implement caching, load balancing, and high availability.
* **Demonstration:**
  + Record a short video (5-10 minutes) demonstrating your application. Highlight the following:
    - How users can search for and view courses.
    - How caching improves performance.
    - How load balancing works in your setup