**Online Learning Platform using MERN stack**

Khushi k

**Table of Contents**

[Introduction 3](#_Toc174401959)

[Functional and Non-Functional Requirements 3](#_Toc174401960)

[Technical Requirements 6](#_Toc174401961)

[Setup Instructions 8](#_Toc174401962)

[Backend Setup, 8](#_Toc174401963)

[Frontend Setup 9](#_Toc174401964)

[API Documentation 10](#_Toc174401965)

[System Design 17](#_Toc174401966)

[UML Diagram 17](#_Toc174401967)

[Sequence Diagram 18](#_Toc174401968)

[Class Diagram 19](#_Toc174401969)

[Git graph Diagram 20](#_Toc174401970)

[Database Diagram 21](#_Toc174401971)

[Architecture Diagram 22](#_Toc174401972)

[SWOT Analysis of the Developer 23](#_Toc174401973)

# Introduction

In the digital age, online learning platforms have revolutionized education by making quality content accessible to a global audience. The development of such platforms involves a combination of innovative technology and user-centric design. This project aims to create a robust online learning platform using the MERN stack, which encompasses MongoDB, Express.js, React.js, and Node.js, to provide a comprehensive and interactive learning experience.

The core functionality of the platform will include user registration and authentication, enabling users to create accounts, log in securely, and manage their profiles. Leveraging JWT (JSON Web Tokens) for authentication ensures that users’ data remains secure and access to sensitive features is protected. The backend will support CRUD (Create, Read, Update, Delete) operations for course management, allowing instructors to design and update courses, and students to browse and enroll in available offerings.

On the frontend, a user-friendly interface will be developed using React.js (or PHP for those preferring a different approach), incorporating essential pages for user registration, login, and course listings. The design will prioritize ease of navigation and responsiveness to enhance user engagement and satisfaction. Students will have access to features for course enrollment, status tracking, and viewing their enrolled courses.

An exciting aspect of this project is the integration of GPT-3, an advanced language model developed by OpenAI. This integration will offer personalized course recommendations based on user interests and career goals, enhancing the learning experience by providing tailored guidance.

The platform will be hosted on a cloud service, ensuring scalability and accessibility. Detailed documentation will accompany the project, covering setup instructions, API documentation, and system design, to facilitate smooth deployment and future maintenance. This project represents a significant step towards creating a modern, scalable, and intelligent online learning environment.

## Functional and Non-Functional Requirements

**Functional Requirements**

1. **User Authentication**
   * **Registration:** Users must be able to create an account by providing necessary details (e.g., username, password, email).
   * **Login:** Users must be able to log in using their registered credentials.
   * **JWT Authentication:** Implement JWT-based authentication to manage user sessions securely.
2. **Course Management**
   * **Create Course:** Instructors should be able to create new courses by providing details such as title, description, and content.
   * **Read Course:** Users should be able to view a list of available courses and details of individual courses.
   * **Update Course:** Instructors should be able to modify course details.
   * **Delete Course:** Instructors should be able to remove courses from the platform.
3. **Course Enrollment**
   * **Enroll in Course:** Students should be able to enroll in available courses.
   * **View Enrollments:** Students should be able to view their enrolled courses and their enrollment status.
   * **Success Notification:** Display a success message upon successful enrollment.
4. **Role-Based Access Control (RBAC)**
   * **Different Roles:** Implement roles such as student and instructor.
   * **Access Control:** Restrict access to certain features based on user roles (e.g., only instructors can create or manage courses).
5. **GPT-3 Integration**
   * **Personalized Recommendations:** Provide course recommendations based on user input, such as career goals or interests.
   * **Interactive Assistance:** Allow students to ask for guidance and receive tailored course suggestions.
6. **User Interface**
   * **Registration/Login Pages:** Provide user-friendly interfaces for account creation and login.
   * **Course Listing:** Display a searchable and filterable list of available courses.
   * **Enrollment Pages:** Provide pages for students to view and manage their course enrollments.

**Non-Functional Requirements**

1. **Performance**
   * **Response Time:** The system should respond to user actions (e.g., API calls) within an acceptable time frame (typically under 2 seconds).
   * **Scalability:** The platform should handle a growing number of users and courses without significant performance degradation.
2. **Reliability**
   * **Availability:** The system should be available 24/7, with minimal downtime.
   * **Error Handling:** The system should gracefully handle errors and provide meaningful error messages to users.
3. **Security**
   * **Data Protection:** User data, including passwords and personal information, should be stored securely using encryption.
   * **Secure Authentication:** Implement secure authentication practices, including password hashing and JWT protection.
   * **Access Control:** Ensure that users can only access features and data appropriate to their role.
4. **Usability**
   * **User Experience:** The interface should be intuitive and easy to navigate, with clear instructions and feedback.
   * **Accessibility:** The platform should be accessible to users with disabilities, adhering to web accessibility standards.
5. **Maintainability**
   * **Code Quality:** The codebase should be well-organized, documented, and adhere to best practices to facilitate future updates and maintenance.
   * **Documentation:** Provide comprehensive documentation for setup, API usage, and system design.
6. **Compatibility**
   * **Browser Support:** The platform should be compatible with major web browsers (e.g., Chrome, Firefox, Safari).
   * **Device Compatibility:** The platform should be responsive and function properly on various devices, including desktops, tablets, and smartphones.

## Technical Requirements

**1. Backend**

1. **Node.js**
   * **Version:** Node.js v14.x or later
   * **Dependencies:** Express.js for handling HTTP requests, Mongoose for MongoDB object modeling
2. **Express.js**
   * **Version:** Express.js v4.x or later
   * **Middleware:** Use middleware for handling authentication, error handling, and parsing request bodies (e.g., body-parser)
3. **MongoDB**
   * **Version:** MongoDB v4.x or later
   * **Database:** Ensure MongoDB is installed locally or use a cloud-based solution (e.g., MongoDB Compass)
   * **Schema Design:** Use Mongoose for defining schemas and interacting with the database
4. **Authentication**
   * **JWT (JSON Web Token):** Use the jsonwebtoken package for secure authentication and authorization
   * **Password Hashing:** Use bcrypt for hashing and verifying passwords
5. **API Documentation**
   * **Tool:** Consider using tools like Swagger or Postman for documenting and testing APIs
6. **GPT-3 Integration**
   * **API:** Use the openai package to integrate GPT-3 for personalized course recommendations
   * **API Key:** Store the OpenAI API key securely, and use it for making API requests

**2. Frontend**

1. **React.js**
   * **Version:** React v17.x or later
   * **Dependencies:** react-router-dom for routing, axios or fetch for making API requests
2. **Frontend Libraries**
   * **UI Framework:** Consider using a UI framework like Bootstrap or Material-UI for consistent styling and responsive design
   * **State Management:** Use React's built-in state management or consider libraries like Redux for complex state management
3. **Development Tools**
   * **Build Tool:** Use webpack or Create React App for building and bundling the application
   * **Testing:** Implement testing with tools like Jest and React Testing Library
4. **Environment Variables**
   * **File:** Use a .env file to manage environment variables for the React application, such as API URLs

## Setup Instructions

### Backend Setup,

1. **Install Node.js**
   * Download and install Node.js from [nodejs.org](https://nodejs.org/). Ensure you install the LTS version.
2. **Clone the Repository**

git clone git@github.com:Shahana-rifkhan/Online-Learning-Platform.git

cd Online-Learning-Platform

1. **Navigate to the Backend Directory**
   * Assuming the backend code is in a directory named backend:

cd backend

1. **Install Dependencies**
   * Install the required npm packages:

npm install

1. **Setup Environment Variables**
   * Create a .env file in the backend directory with the following content (adjust values as needed):

MONGO\_URI=mongodb://localhost:27017/online-learning-platform

JWT\_SECRET=your\_jwt\_secret\_key

GPT\_API\_KEY=your\_openai\_api\_key

1. **Run the Server**
   * Start the server:

npm start

* + The server should now be running on http://localhost:5000 (or whatever port you have configured).

### Frontend Setup

1. **Navigate to the Frontend Directory**
   * Assuming the frontend code is in a directory named frontend:

cd frontend

1. **Install Dependencies**
   * Install the required npm packages:

npm install

1. **Setup Environment Variables**
   * Create a .env file in the frontend directory with the following content:

REACT\_APP\_API\_URL=http://localhost:5000/api

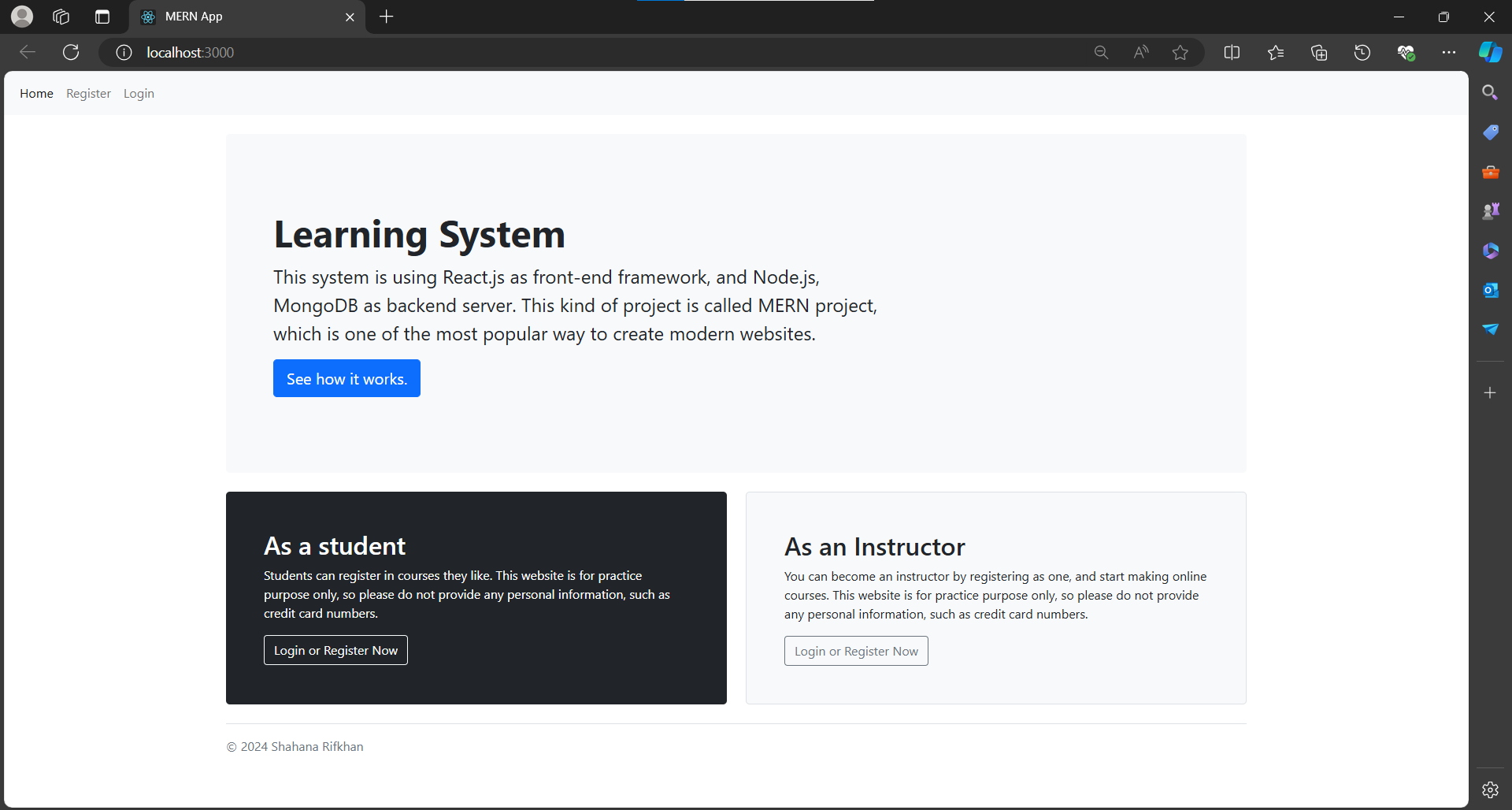
* + Adjust the REACT\_APP\_API\_URL to match your backend server URL.

1. **Run the React Application**
   * Start the development server:

npm start

* + The React application should now be running on <http://localhost:3000>.

**Output:**



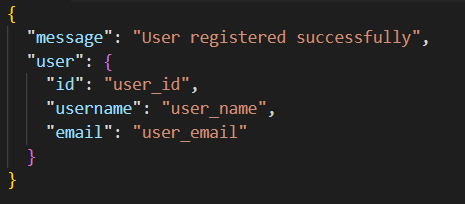
You can access the application through online 🡪 <http://192.168.167.53:3000/>

## API Documentation

**1. Endpoints**

**1.1 User Registration**

* **Method:** POST
* **URL:** /api/register
* **Request Parameters:**
  + body: JSON object containing user details
    - username: string, required
    - email: string, required
    - password: string, required
* **Response Format:**
  + **Success (201 Created):**



* + **Error (400 Bad Request):**

A black background with white text

Description automatically generated

**User Login**

* **Method:** POST
* **URL:** /api/login
* **Request Parameters:**
  + body: JSON object containing login details
    - username: string, required
    - password: string, required
* **Response Format:**
  + **Success (200 OK):**

A screen shot of a computer screen

Description automatically generated

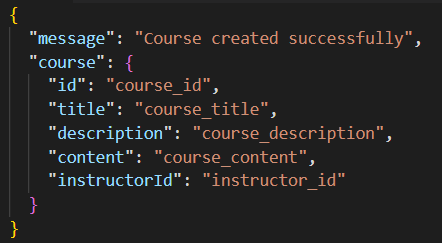
* + **Error (401 Unauthorized):**

A black background with white text

Description automatically generated

**1.3 Create Course**

* **Method:** POST
* **URL:** /api/courses
* **Request Parameters:**
  + body: JSON object containing course details
    - title: string, required
    - description: string, required
    - content: string, required
    - instructorId: string, required
* **Response Format:**
  + **Success (201 Created):**



* + **Error (400 Bad Request):**

A black background with white text

Description automatically generated

**1.4 Get Course List**

* **Method:** GET
* **URL:** /api/courses
* **Request Parameters:**
  + None
* **Response Format:**
  + **Success (200 OK):**

A computer screen shot of text

Description automatically generated

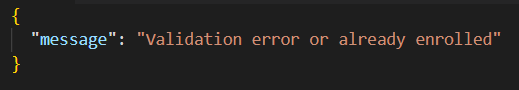
**1.5 Enroll in Course**

* **Method:** POST
* **URL:** /api/enrollments
* **Request Parameters:**
  + body: JSON object containing enrollment details
    - courseId: string, required
    - studentId: string, required
* **Response Format:**
  + **Success (201 Created):**

A screen shot of a computer code

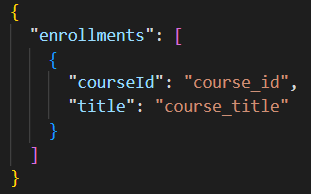
Description automatically generated

* + **Error (400 Bad Request):**



**1.6 Get Enrolled Courses**

* **Method:** GET
* **URL:** /api/enrollments/:studentId
* **Request Parameters:**
  + path: studentId: string, required
* **Response Format:**
  + **Success (200 OK):**



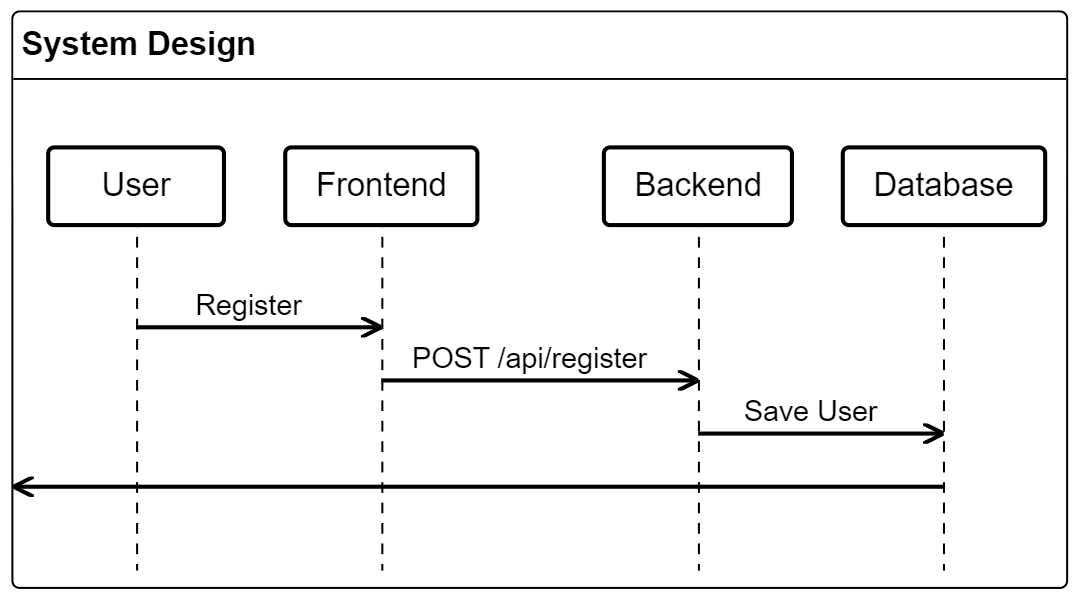
**2. Authentication**

**JWT Authentication Overview:**

* **Generating JWT:**
  + Upon successful login, a JWT is generated and returned to the client.
  + The JWT is signed using a secret key (JWT\_SECRET).
* **Using JWT:**
  + **Include Token in Requests:** The token should be included in the Authorization header of requests that require authentication.
  + **Middleware:** Implement middleware on the backend to validate the JWT. This middleware should:
    - Check the presence of the token in the Authorization header.
    - Verify the token using the secret key.
    - Decode the token to extract user information and attach it to the request object.
* **Error Handling:**
  + **Invalid Token:** If the token is invalid or expired, respond with a 401 Unauthorized status and an appropriate error message.
  + **Missing Token:** If the token is not provided, respond with a 401 Unauthorized status and indicate that authentication is required.

## System Design

### UML Diagram



### Sequence Diagram

A diagram of a company

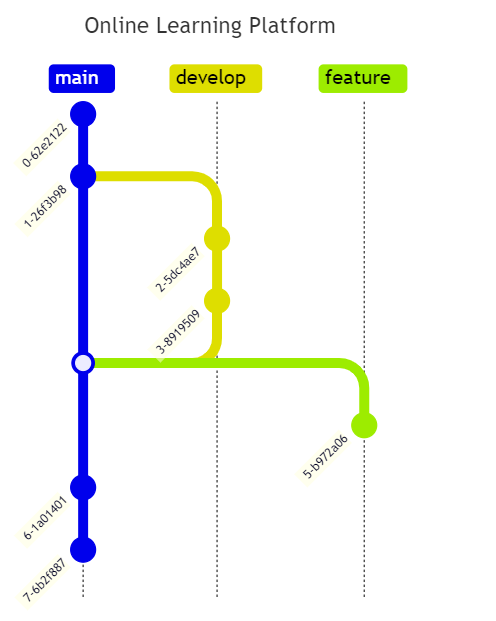
Description automatically generated

### Class Diagram

A diagram of a course

Description automatically generated

### Git graph Diagram

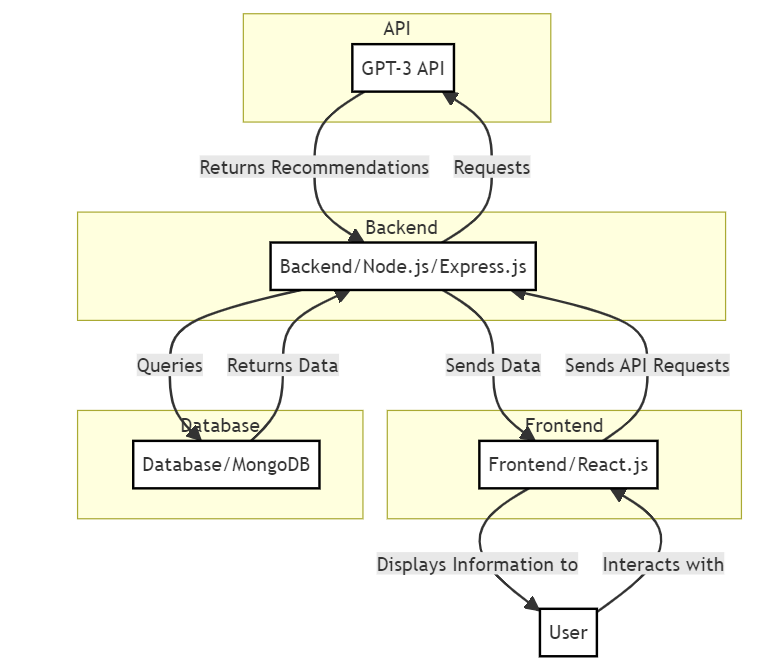


### Database Diagram

A diagram of a course

Description automatically generated

### Architecture Diagram



# SWOT Analysis of the Developer

|  |  |
| --- | --- |
| **Strengths** | **Weaknesses** |
| Proficient in the MERN stack and PHP. | May face challenges in effectively integrating GPT-3 due to limited experience. |
| Strong understanding of backend and frontend development. | Requires time and learning to master advanced API integrations. |
| Ability to tackle complex technical issues. | Balancing multiple tasks and deadlines can be challenging. |
| Proven track record in implementing robust solutions. | Potential for delays if project scope and deadlines are not well managed. |
| Knowledge in deploying applications on cloud platforms like Azure. | Reliance on GPT-3 API and other external services can lead to potential issues if these services experience downtime or changes. |
| Familiarity with cloud-based scaling and management. | Requires consistent monitoring and management of third-party integrations. |
| **Opportunities** | **Threats** |
| Opportunity to gain experience with cutting-edge technologies like GPT-3. | Rapid evolution of technology can require continuous learning and adaptation. |
| Potential to enhance skills in cloud services and advanced API integrations. | Risk of falling behind if not continuously updated with the latest trends and tools. |
| Chance to collaborate with other professionals and experts in the field. | Ensuring the security of user data and privacy can be challenging and requires rigorous implementation. |
| Potential to build a strong professional network through project involvement. | Threat of security vulnerabilities if not properly addressed. |
| Adding a significant project to the portfolio which can lead to new opportunities. | Ensuring the application performs well under high traffic and scales appropriately. |
| Potential to showcase expertise in a high-impact project. | Risk of performance issues if not carefully optimized and tested. |