**Project Overview**

Develop a Customer Management System (CMS) that allows users to add, update, delete, and view customer profiles through a web interface. This application should have both frontend and backend components, with a focus on clean design, efficient data handling, and secure functionalities.

**Technical Requirements**

**Frontend:**

Technologies: ReactJS or VueJS

**Features:**

User Interface (UI) for listing all customers

Forms to add and edit customer details

Option to delete customers

Responsive design to ensure usability on desktop and mobile devices

Client-side routing

**Backend:**

**Technology:** ASP.NET MVC and Web API

**API Endpoints:**

GET /customers - Retrieve all customers

POST /customers - Add a new customer

PUT /customers/{id} - Update an existing customer

DELETE /customers/{id} - Delete a customer

**Authentication:** Implement JWT (JSON Web Tokens) for secure API access

**Database:**

**Database System:** SQL Server

**Schema:**

**Customers Table:**

CustomerId (Primary Key, Auto-Generated, Integer)

FirstName (String)

LastName (String)

Email (String, Unique)

Phone (String)

Address (String)

Additional Requirements:

**Documentation:** Provide a README file detailing setup instructions, API endpoints, and a brief overview of technologies used.

**Testing:** Include unit and integration tests for both frontend and backend components.

**Performance:** Basic performance optimization for database queries and frontend loading times.

**Development Guidelines**

**Code Quality:**

Follow best coding practices, such as using meaningful variable names, commenting on complex logic, and keeping functions short and focused.

Ensure code is modular and reusable with appropriate separation of concerns (SoC).

**Version Control:**

Use Git (Github) for version control.

Include a .gitignore file configured for Visual Studio and node\_modules.

Commit changes incrementally with meaningful commit messages.

**Security:**

Secure the API using JWT authentication.

Ensure that all data inputs are validated on the server side to prevent SQL Injection and XSS attacks.

**Deployment:**

Provide instructions for setting up the development environment and any configurations needed for deploying the application locally.

**Evaluation Criteria**

Functionality: The application must meet all the functional requirements specified in the technical requirements.

**Design:** The UI should be clean and user-friendly.

**Code Quality:** Code should be well-organized, properly commented, and follow best practices.

**Security:** Application should handle authentication and data validation securely.

**Documentation:** The project should include clear documentation on how to set up and run the application, along with an explanation of the API.

**Deliverables**

Source code for both frontend and backend.

Database script for creating and seeding the database.

Documentation and setup instructions.