Assignment 3: Building a CRUD API with Node.js and MongoDB

Objective:

Create a fully functional CRUD (Create, Read, Update, Delete) API for a simple blogging platform using **Node.js** and **MongoDB**. This is an Individual Project.

Requirements:

1. Setup and Initialization

- Initialize a new **Node.js** project.
- Set up a **MongoDB** database for the application.

2. API Endpoints

Develop the following RESTful API endpoints:

1. **POST /blogs**: Create a new blog post.
2. **GET /blogs**: Retrieve all blog posts.
3. **GET /blogs/:id**: Retrieve a single blog post by its ID.
4. **PUT /blogs/:id**: Update a blog post by its ID.
5. **DELETE /blogs/:id**: Delete a blog post by its ID.

3. Database Operations

- Use **MongoDB methods**
- Handle errors and return appropriate **HTTP status codes** and responses.
- Each blog post should include:
    - Title (string, required)
    - Body (string, required)
    - Author (string, optional, default: "Anonymous")
    - Timestamps (createdAt, updatedAt)

4. Data Validation

- Ensure all blog post submissions include:
    - A **title**
    - A **body**

5. Error Handling

- Implement comprehensive error handling for:

    - Database issues
    - Invalid requests

6. Testing

- Test the API manually using tools like **Postman**.

7. Building a Simple Interface

- Create a basic front-end interface to interact with the CRUD API.
- This task provides an opportunity to practice full-stack development skills.

Submission:

- Submit the following:
    - A .zip file or a GitHub repository link containing your project.

Grading Criteria:

Criteria	Description	Weight
1. Core Functionality	The CRUD API is fully functional with all endpoints implemented: POST, GET, PUT, DELETE.	30%
2. Data Validation and Error Handling	Proper data validation is in place for required fields (e.g., title, body). Errors are handled effectively, with appropriate HTTP status codes and clear responses.	20%
3. Project Organization and Design	The project demonstrates clean code practices, follows an organized structure, and implements a responsive and visually appealing user interface.	10%
4. Defense	The student clearly explains the functionality, structure, and decisions behind the code during the defense. They should also answer theoretical questions related to the project and the concepts used.	40%