W3villa Technologies - Assignment - Task Manager Web Application

Assignment Title: Task Manager Web Application

Description:
In this assignment, you will develop a Task Manager web application that allows users to create, view, update, and delete tasks. The application will consist of frontend, backend, database, and API components. Users should be able to register, log in, and manage their tasks effectively.

Tasks:

Backend Development:
1. Set up a backend server using Node.js and Express.js.
2. Implement user authentication and authorization using JWT (JSON Web Tokens).
3. Create API endpoints to handle user registration, login, and task management operations (CRUD - Create, Read, Update, Delete).
4. Ensure proper validation and error handling for API requests.

Database Setup:
1. Choose a database system (e.g., MongoDB, PostgreSQL) and set it up.
2. Design and create the necessary database schema to store user information and tasks.

Frontend Development:
1. Develop a responsive and user-friendly frontend interface using HTML, CSS, and JavaScript (you may use frameworks like React.js, Vue.js, or Angular).
2. Implement user registration and login forms.
3. Create pages for displaying and managing tasks, including options to add, edit, and delete tasks.

Integration:
1. Integrate the frontend with the backend APIs to enable data retrieval and manipulation.
2. Implement user authentication mechanisms to secure access to task management functionalities.

Acceptance Criteria:
Functionality: The web application should allow users to register, log in, and perform CRUD operations on tasks.
User Interface: The frontend should have an intuitive and visually appealing interface, with responsive design for different screen sizes.
Backend Implementation: The backend should handle user authentication, authorization, and task management operations efficiently.

Database Integration: The database should be properly integrated with the backend, and data should be stored securely.
Code Quality: Write clean, well-structured code following best practices and coding standards of the chosen language and frameworks.
Security: Implement proper security measures, including input validation, password hashing, and protection against common security vulnerabilities.
Error Handling: Implement robust error handling mechanisms to provide meaningful error messages to users.
Documentation: Provide clear documentation for setting up and running the application, including any dependencies or configurations required.
Testing: Conduct thorough testing to ensure all functionalities work as expected and handle edge cases appropriately.
Additional Criteria:

- Code pushed and hosted on public GitHub repository
- Clear instructions in README to setup and run the project
- Live application hosted on a server/cloud platform
- Short 2-3 minute demo video showcasing the functionality
- Present main workflows and code structure