Assignment Title: Full-Stack Developer Challenge

Introduction: You are tasked with building a simple task management application. The application will allow users to create, update, and delete tasks. Tasks should have a title, description, and a status (e.g., "To Do," "In Progress," "Done"). Users should also be able to view a list of tasks and filter them by status.

If you are attempting this as Frontend(mobile/web) Assignment, feel free to use firebase for db and authentication.

Front-End Requirements:

1. **User Interface:** Create a user-friendly interface for the task management application. It should have atleast the following components:
    - A form to create a new task with fields for title, description, and status.
    - A list of tasks with the ability to update the status or delete a task.
    - A filter or dropdown to filter tasks by status (e.g., "All," "To Do," "In Progress," "Done").

You can be creative in adding additional features here.

2. **User Experience:** Implement smooth and responsive user interactions, including form validation to ensure that tasks cannot be created without a title. Use modern front-end technologies such as React, Angular, or Vue.js.

3. **Styling:** Style the application using CSS or a CSS preprocessor (e.g., SASS/SCSS). You can also use a CSS framework if preferred.

4. **Responsive Design:** Ensure that the application is responsive and works well on both desktop and mobile devices.

Back-End Requirements:

1. **API Development:** Create a RESTful API to handle the CRUD (Create, Read, Update, Delete) operations for tasks. The API should be built using a back-end technology of your choice (e.g., Node.js with Express, Ruby on Rails, Django, etc.).

2. **Data Storage:** Implement a database to store task data. You can use any database system (e.g., PostgreSQL, MySQL, MongoDB) and set up the necessary data models to represent tasks.

3. **Validation:** Implement server-side validation to ensure that task data is valid before saving it to the database. Tasks must have a title and a valid status.

4. **Error Handling:** Properly handle errors, including sending appropriate error messages and status codes in response.

General Requirements:

1. **Code Quality:** Write clean, well-documented, and maintainable code. Use coding best practices and conventions for the chosen programming language and framework.
2. **Version Control:** Use a version control system (e.g., Git) to track changes in your code and provide a Git repository for the assessment.
3. **Testing:** Write unit tests for critical parts of your application, such as API endpoints and data validation.
4. **Security:** Implement basic security measures to protect the application from common vulnerabilities.

Bonus Features (Optional):

You can implement additional features to make your project stand out:

1. User authentication and authorization to restrict access to tasks.
2. Task due dates and reminders.
3. Task sorting and searching capabilities.
4. User profiles with avatars.

Submission:

1. Provide a link to your version-controlled repository (e.g., GitHub, GitLab).
2. Include clear instructions on how to set up and run your application.
3. Share any additional documentation or notes that might help reviewers understand your project.

Assessment Criteria:

Your assignment will be evaluated based on:

1. Functionality: Does the application meet the specified requirements and work as expected?
2. Code Quality: Is the code clean, organized, and well-documented?
3. User Experience: Is the user interface intuitive and responsive?

4. Security: Are there basic security measures in place?
5. Testing: Are there unit tests for critical components?
6. Bonus Features: If implemented, do they enhance the application's usability?

This assignment is designed to assess your full-stack development skills, so feel free to showcase your capabilities and creativity. Good luck!

P.S. Any assumptions taken while design / implementation should be documented in README file