<u>To-Do App - [Afshan Khan]</u>

<u>Challenges and Solutions</u>

1. **<u>Fetch Task Details</u>**: I encountered a challenge when attempting to retrieve task details belonging to a user from the Task Listing Page and pass them to the Update Task Page in order to pre-fill certain values
   <u>Solution</u>: After conducting a Google search, I utilized the **useLocation** hook in React to **access the current object** and passed the state object to the Update Task page using the **useNavigation** hook, which facilitates routing navigation**.**

2. **<u>Past Date:</u>** I encountered an issue while implementing the logic to prevent the selection of past due dates
   <u>Solution:</u> After extensive research, I managed to understand the logic and successfully implemented it..

3. **<u>Json Web Token (JWT) Implementation:</u>** I encountered an issue during the implementation of **JWT (JSON Web Token)** integration for controlling access to both **frontend and backend routes**, aiming to restrict access only to p**rotected routes** and **prevent unauthorized users** from accessing certain resources.
4. **<u>Responsive Design</u>** : I utilized media queries to ensure responsiveness across all devices.

<u>Security Concerns</u>

1. **<u>Authentication and Authorization:</u>** I implemented **JWT (JSON Web Tokens)** for secure authentication mechanisms to verify user identities and authorization checks to control access to different parts of the application, adding an extra layer of protection against unauthorized access.

2. **Bcrypt:** Hash password using **Bcrypt** before saving it in the database during registration to keep user password secure from hackers.
3. **Data Validation:** Ensure that user inputs are properly validated on both the client and server sides to prevent potential attacks like SQL injection or Cross-Site Scripting (XSS).
4. **Secure Error Handling**: Be careful not to expose **sensitive** information in error messages that could be exploited by attackers.
5. **Protection against Cross-Site Request Forgery (CSRF)**: Implement measures to protect against CSRF attacks by using **tokens** or other methods to validate requests originating from your application.
6. **Response from Server:** The server must send responses to the browser in a secure manner to prevent security concerns.

Optimization Concerns

1. **Performance Optimization:** Minimize load times by optimizing code, reducing unnecessary API calls, and employing techniques like code splitting in React to ensure faster rendering of components.
2. **Database Indexing**: Optimize database queries by appropriately indexing fields that are frequently used for searching or filtering tasks.
3. **Code and Resource Minification**: Minify JavaScript, CSS, and HTML files to reduce their file sizes, thereby improving load times.
4. **Avoid Unnecessary Rendering:** Optimize rendering performance by avoiding unnecessary re-renders, and using PureComponent or React.memo where applicable.
5. **Avoid Repetition of Code:** Optimize application by avoiding repetition of code.
6. **Monitoring**: Continuously monitor the application's performance using tools to identify bottlenecks and areas that need optimization.
7. **Caching:** Implement caching strategies (client-side and server-side) for frequently accessed data to reduce server load and improve response times.

EXTRA : I have also created a Register and Login page. Please note that the responsiveness of these pages might not fully reflect my focus, as I primarily concentrated on meeting the given project requirements.