Project Description: Extensive Overview

The project involves developing a fully functional Todo website that leverages modern web technologies including Firebase, Next.js, and Git. This website will allow users to manage their tasks efficiently by providing a comprehensive set of features for creating, reading, updating, and deleting (CRUD) todo items. Additionally, the application will support user authentication, session management, and real-time data handling, ensuring a seamless and dynamic user experience.

Key Technologies:

1. **Firebase:**
    - **Authentication:** Firebase Authentication will be used to manage user sign-ups, logins, and session persistence.
    - **Firestore Database:** Firebase Firestore, a flexible and scalable NoSQL cloud database, will store the todo items and user data.
    - **Cloud Functions (optional):** For handling server-side operations such as sorting and searching if needed.
2. **Next.js:**
    - **React Framework:** Next.js will serve as the primary framework for building the front-end, enabling server-side rendering and static site generation for optimal performance.
    - **API Routes:** Next.js API routes will be utilized to create RESTful endpoints for CRUD operations, ensuring efficient communication between the front-end and the Firebase backend.
    - **TypeScript (preferred):** TypeScript will be used to enhance code quality and maintainability through static type checking.
3. **Git:**
    - **Version Control:** Git will be used for version control, allowing for efficient project management and collaboration through platforms like GitHub or Bitbucket.

Core Functionalities:

1. **User Authentication and Session Management:**
    - **Registration:** New users can register by providing their name, email, and a password (minimum 8 characters, with email validation).
    - **Login:** Registered users can log in using their email and password. Sessions should persist so users don’t need to log in every time they visit the site.
    - **Logout:** Users can terminate their session and log out of the application.
2. **Todo Management:**
    - **Create:** Users can add new todos with relevant details such as title, description, and due date.
    - **Read:** Users can view a list of their todos, categorized into "Completed" and "Incomplete" sections.
        - **Sorting:** Todos in both sections will be sorted by their creation date (newest first), managed server-side.
    - **Update:** Users can update existing todos, modifying details such as the title, description, or completion status.
    - **Delete:** Users can remove todos from their list.
    - **Search:** Users can search for specific todos based on their content, with the search functionality applied to both "Completed" and "Incomplete" lists.
3. **User Interface and Experience:**
    - **Responsive Design:** The website will be responsive, ensuring usability across various devices and screen sizes.

- **Atomic Design Pattern:** The front-end will be organized according to the Atomic Design Pattern, promoting reusable components and consistent UI elements.
- **Bootstrap Integration:** Bootstrap will be used to style the application, ensuring a modern and responsive design.

4. **Caching and Performance:**
   - **Redis Cache:** Todos will be cached using Redis to enhance performance. The cache will be updated whenever CRUD operations are performed, ensuring that the user sees the most up-to-date information without unnecessary database queries.
5. **Additional Features:**
   - **Form Validations:** All forms will include validation to ensure data integrity and provide feedback to users.
   - **Browser Tab Customization:** The browser tab will display a custom image and title for a polished user experience.

Development Process and Best Practices:

1. **Version Control with Git:**
   - Regular commits with meaningful messages to document progress.
   - Use of branches for features and pull requests to merge into the main branch, facilitating code reviews and collaborative development.
2. **Clean Code and Documentation:**
   - Adherence to clean coding principles, including meaningful variable names, modular functions, and thorough commenting.
   - Documentation of code and project structure to aid future development and maintenance.
3. **Testing and Error Handling:**
   - Although unit tests are optional, exploring and understanding testing tools and methodologies is encouraged.
   - Implement robust error handling and user feedback mechanisms to improve reliability and user experience.
4. **Deployment:**
   - Deploy the application using platforms like Vercel, ensuring it is accessible and performant in a live environment.
