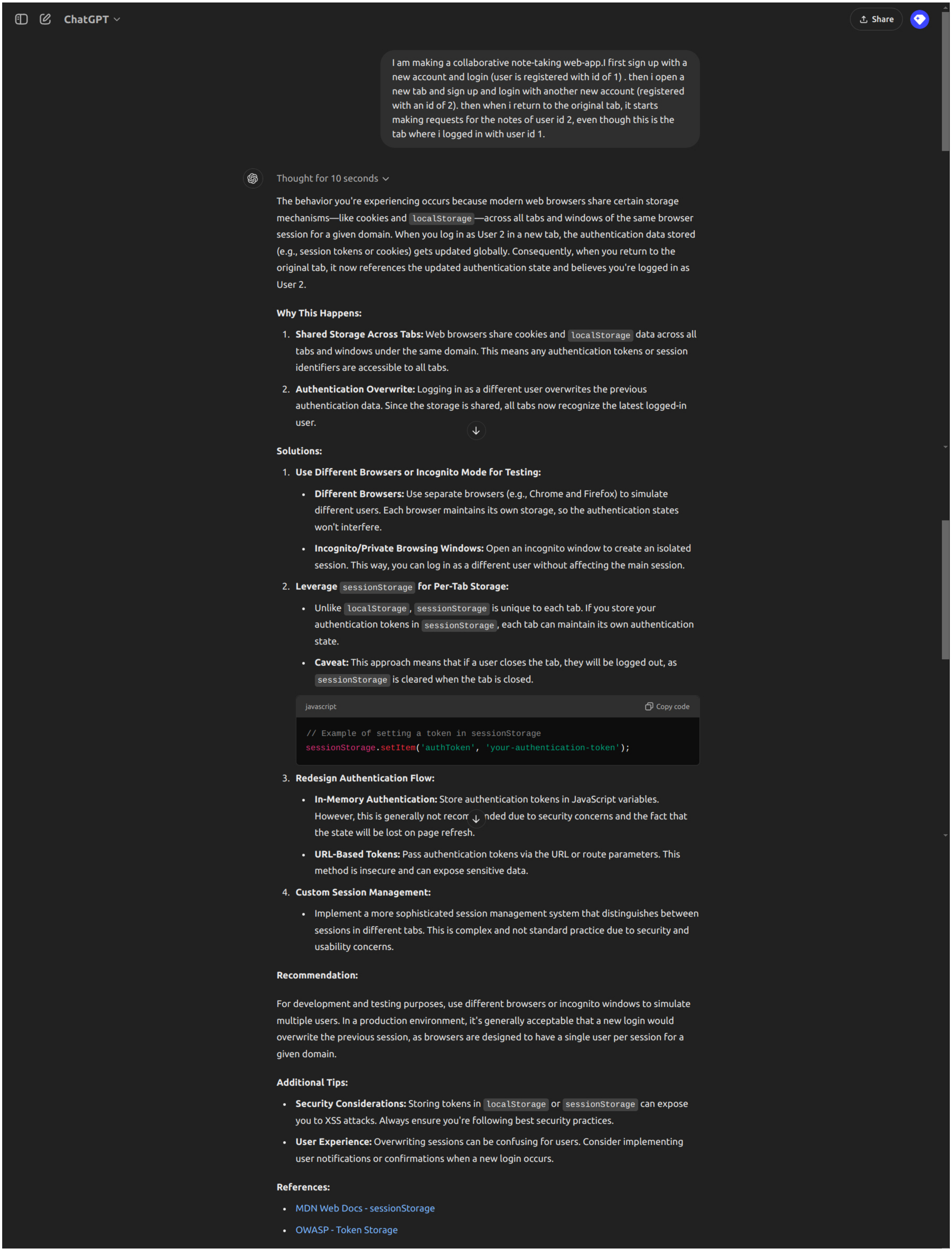


I am making a collaborative note-taking web-app.I first sign up with a new account and login (user is registered with id of 1) . then i open a new tab and sign up and login with another new account (registered with an id of 2). then when i return to the original tab, it starts making requests for the notes of user id 2, even though this is the tab where i logged in with user id 1.

The behavior you're experiencing occurs because modern web browsers share certain storage mechanisms—like cookies and `localStorage`—across all tabs and windows of the same browser session for a given domain. When you log in as User 2 in a new tab, the authentication data stored (e.g., session tokens or cookies) gets updated globally. Consequently, when you return to the original tab, it now references the updated authentication state and believes you're logged in as User 2.

Why This Happens:

1. **Shared Storage Across Tabs:** Web browsers share cookies and `localStorage` data across all tabs and windows under the same domain. This means any authentication tokens or session identifiers are accessible to all tabs.
2. **Authentication Overwrite:** Logging in as a different user overwrites the previous authentication data. Since the storage is shared, all tabs now recognize the latest logged-in user.

Solutions:

1. **Use Different Browsers or Incognito Mode for Testing:**
    - **Different Browsers:** Use separate browsers (e.g., Chrome and Firefox) to simulate different users. Each browser maintains its own storage, so the authentication states won't interfere.
    - **Incognito/Private Browsing Windows:** Open an incognito window to create an isolated session. This way, you can log in as a different user without affecting the main session.
2. **Leverage `sessionStorage` for Per-Tab Storage:**
    - Unlike `localStorage`, `sessionStorage` is unique to each tab. If you store your authentication tokens in `sessionStorage`, each tab can maintain its own authentication state.
    - **Caveat:** This approach means that if a user closes the tab, they will be logged out, as `sessionStorage` is cleared when the tab is closed.

```javascript
// Example of setting a token in sessionStorage
sessionStorage.setItem('authToken', 'your-authentication-token');
```

3. **Redesign Authentication Flow:**
    - **In-Memory Authentication:** Store authentication tokens in JavaScript variables. However, this is generally not recommended due to security concerns and the fact that the state will be lost on page refresh.
    - **URL-Based Tokens:** Pass authentication tokens via the URL or route parameters. This method is insecure and can expose sensitive data.
4. **Custom Session Management:**
    - Implement a more sophisticated session management system that distinguishes between sessions in different tabs. This is complex and not standard practice due to security and usability concerns.

Recommendation:

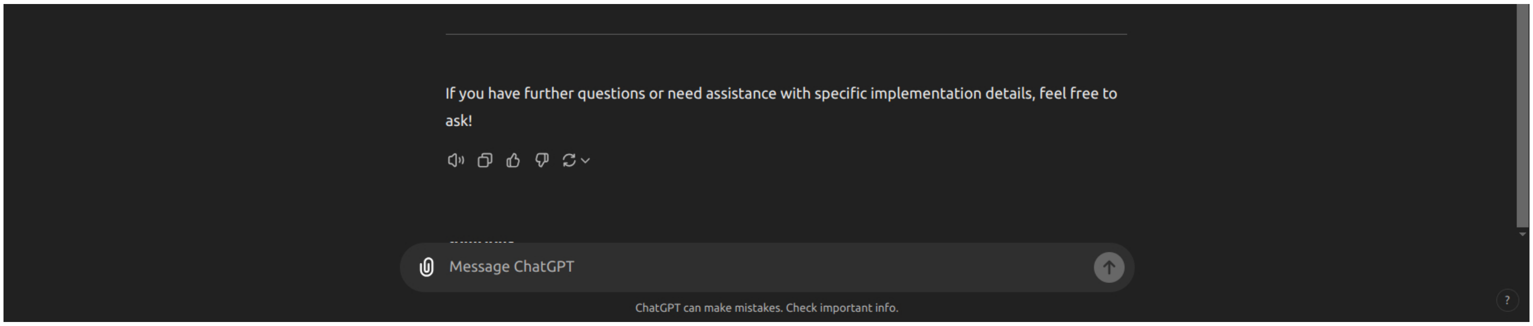
For development and testing purposes, use different browsers or incognito windows to simulate multiple users. In a production environment, it's generally acceptable that a new login would overwrite the previous session, as browsers are designed to have a single user per session for a given domain.

Additional Tips:

- **Security Considerations:** Storing tokens in `localStorage` or `sessionStorage` can expose you to XSS attacks. Always ensure you're following best security practices.
- **User Experience:** Overwriting sessions can be confusing for users. Consider implementing user notifications or confirmations when a new login occurs.

References:

- MDN Web Docs - sessionStorage
- OWASP - Token Storage



If you have further questions or need assistance with specific implementation details, feel free to ask!