**Code Inspection for account.js**

**Description of Functionality:**

The account.js script manages user authentication, preferences, and password updates for the account page. Key features include:

* **Session Validation:** Redirects unauthenticated users to the login page.
* **Dynamic User Data:** Populates account details from sessionStorage.
* **Dark Mode Toggling:** Syncs theme preferences with the server.
* **Password Update:** Validates and submits new passwords with real-time feedback.
* **Logout:** Clears session data and redirects to the login page.

**Key Features and Implementation Details:**

1. **Session Management:**
   * **Redirect Unauthenticated Users:**

javascript

Copy

const user = JSON.parse(sessionStorage.getItem('user'));

if (!user) window.location.href = '/pages/login.html';

* + **Logout Function:** Clears sessionStorage and redirects:

javascript

Copy

function logout() {

sessionStorage.clear();

window.location.href = '/pages/login.html';

}

1. **User Data Population:**
   * Renders account details (e.g., operator\_id, email) from sessionStorage:

javascript

Copy

document.getElementById('accountDetails').innerHTML = `...`;

* + Handles legacy full\_name field for backward compatibility.

1. **Dark Mode Toggle:**
   * **Initialization:**

javascript

Copy

htmlElement.setAttribute('data-bs-theme', user.dark\_mode ? 'dark' : 'light');

darkModeToggle.checked = user.dark\_mode;

* + **Preference Sync:** Updates sessionStorage and sends POST request:

javascript

Copy

updatePreference('dark\_mode', isDarkMode ? 1 : 0);

1. **Password Update Workflow:**
   * **Validation:** Checks password match before submission.
   * **API Call:** Sends new password to http://localhost:5000/api/update-password.
   * **User Feedback:** Displays success/error messages with color-coded alerts.
2. **API Communication:**
   * **Endpoints:**
     + Preferences: http://localhost:5000/api/update-preference
     + Password: http://localhost:5000/api/update-password
   * **Headers:** Uses application/json for all requests.

**Error Handling and Feedback:**

* **Password Mismatch:** Displays red alert: Passwords do not match.
* **Network Errors:** Logs to console and shows Network error - Please try again.
* **Server Responses:** Parses success flag to display success/failure messages.

**Security Considerations:**

* **Session Storage:** Uses sessionStorage for short-lived user data (cleared on logout).
* **Password Validation:** Relies on backend for complexity checks (no client-side regex).
* **Risk:** Logout only clears client-side data; server-side session invalidation not implemented.

**Areas for Improvement:**

1. **Loading States:**
   * Add spinner visibility toggling during API calls (e.g., .spinner-border.d-none).
2. **Enhanced Validation:**
   * Client-side password complexity checks (e.g., regex for special characters).
3. **Server-Side Logout:**
   * Call an API endpoint to invalidate the server session.
4. **Environment Configuration:**
   * Replace hardcoded localhost:5000 with dynamic API base URLs.
5. **Error Messaging:**
   * Display server-side errors (e.g., result.error) more prominently.

**Code Snippets:**

**Password Update Handler:**

javascript

Copy

async function updatePassword(event) {

event.preventDefault();

const newPass = document.getElementById('newPassword').value;

const confirmPass = document.getElementById('confirmPassword').value;

// ... validation and API call

}

**Preference Update Function:**

javascript

Copy

function updatePreference(key, value) {

// Updates sessionStorage and sends POST request

}

**Final Notes:**

* The script effectively integrates frontend logic with backend APIs but lacks robust error recovery.
* User experience can be enhanced with loading indicators and detailed input validation.
* Security could be strengthened by implementing server-side session termination on logout.