**Profile Page**  
**Description:** This page allows users to view and edit their personal information including avatar, username, email, first name, and last name. Users can upload a new avatar, edit information, set a new password, log out, and delete their account.

**User Story (if applicable):**  
*"As a user, I want to view and update my profile information, so that I can manage my account."*

**1. Acceptance Criteria**

Define what needs to be met for this item to be considered **complete**.

✅ Automatically fetches and displays current user’s avatar and basic info on page load.

✅ Users can click icons to edit their username, email, first name, and last name.

✅ Clicking the icon to confirm after editing sends updates to the server.

✅ Users can upload a new avatar image and see it updated.

✅ Clicking “Set New Password” redirects to the password change page.

✅ Clicking “Log out” prompts a confirmation dialog, and logs the user out if confirmed.

✅ Clicking “Account Cancellation” prompts a confirmation dialog and deletes the user account upon confirmation.

✅ All actions provide feedback via pop-up messages.

**2. Requirements & Specifications**

Provide clear **functional** and **non-functional** details.

* **Functional:**

Fetch and display user’s avatar, username, email, first name, and last name.

Allow field-by-field toggling between view and edit mode.

Avatar upload must accept image files and update both in UI and server.

“Set New Password” redirects to change password page.

“Log out” sends a POST request to /api/logout, clears cookies and session, and redirects to the login page.

“Account Cancellation” sends a DELETE request to /api/user/delete, deletes the account, and logs out the user.

All interactions provide feedback through success/error popups or confirmation modals.

* **Non-functional:**

Page must be accessible only by logged-in users.

All requests must be protected by CSRF tokens.

**Optional:** Include wireframes or process flow diagrams.

**3. Dependencies & Constraints**

* Users must be logged in to access the page.

• The system must have models like CustomUser and Profile.

• Email addresses must be unique across users. Users need to log in again after changing their email.

**4. GWT**

Provide Given-Then-When for main cases and edge cases.

**Scene1:** Update user information

**Given:** User is on the profile page with editable fields.

**When:** User edits values and clicks confirm.

**Then:** System updates the database and shows “Profile updated successfully.”.

**Scene2:** Change avatar

**Given:** User clicks the upload avatar button and selects an image.

**When:** The image is uploaded to /api/user/avatar and the database.

**Then:** System shows the new avatar and displays “Avatar updated successfully.”.

**Scene3:** Delete account

**Given:** User clicks “Account Cancellation” button.

**When:** User confirms in the dialog.

**Then:** System calls DELETE /api/user/delete, deletes the account, and logs out.

**Scene4:** Logout

**Given:** User clicks “Log out” button.

**When:** User confirms in the dialog.

**Then:** System calls POST /api/logout, logs the user out, and redirects to login page.