**Areas for Improvement:**

1. **User Experience**:
   * **Better Feedback**: After deleting or editing data (like a QR code or profile info), showing a success message (via toast notifications) would enhance the experience. Right now, users may not be immediately sure if their action was successful unless they refresh the page.
   * **Enhanced Validation**: Consider adding more validation for user input, especially for profile edits and QR code inputs. For instance, ensuring email formatting or providing clear messages when passwords don’t meet certain criteria could be helpful.
2. **Performance**:
   * **Loading States**: While you have basic loading states, adding a spinner or more visual cues during data fetching (e.g., when generating QR codes or fetching user data) would make the app feel more responsive.
   * **Debouncing Inputs**: If the app handles large datasets (e.g., users generating many QR codes), adding techniques like **input debouncing** for search fields or optimizing the rendering of QR codes could improve performance.
3. **Error Handling**:
   * You're displaying errors (e.g., if fetching QR codes fails), which is great. However, **user-friendly error pages** (like 404 or server errors) might be useful, especially if the app grows.
   * Make sure all backend errors are properly handled and that users aren't left stuck in an unresponsive state in case of a server issue.
4. **User Engagement**:
   * **QR Code History**: You might want to allow users to add labels or tags to their generated QR codes to make it easier to categorize or search for older codes.
   * **QR Code Customization**: Offering customization options for QR codes (like changing colors, adding logos, or generating different formats) could make the feature more engaging for users.
   * **Profile Personalization**: Allow users to add a profile picture or short bio, especially if the app evolves into a more social or professional platform.
5. **Security Enhancements**:
   * **Rate Limiting**: If the app scales up, especially with public-facing endpoints (like QR code generation), implementing rate limiting could prevent abuse.
   * **JWT (JSON Web Tokens)**: While sessions are good, modern applications often use JWT for stateless authentication. If you anticipate scaling, it may be worth exploring a switch to JWT, which can simplify session management.
6. **Testing**:
   * **Unit and Integration Testing**: If you haven't already, introducing automated tests for both the frontend and backend will help ensure that as the app grows, new features don’t break existing functionality. Tools like **Jest** for React and **PyTest** for Python could be helpful.
   * **Cross-Browser Testing**: Make sure the app looks and behaves as expected across different browsers and screen sizes.

**Potential Future Features:**

* **API Integration**: You could allow users to generate QR codes for dynamic content, such as linking them to specific databases or generating codes that expire after a set time.
* **QR Code Analytics**: If your user base grows, adding basic analytics (like how many times a QR code has been scanned) could be a valuable feature.
* Check if the python script isn’t too long. If it is, it might be helpful to divide it and add a helpers file or something. Ask ChatGPT what are the industry standards.
* When sessions end, send a notification to the user that their session expired and redirect them to the main page again with a cleared session.
* When users log out, send a notification saying they were successfully logged out.