Appwrite Account Recovery Implementation

This document provides step-by-step instructions for setting up and handling the account recovery process in your web application using Appwrite's `createRecovery` function. It covers both the development phase (local environment) and the production phase (deployed application).

Overview

Appwrite's `createRecovery` function allows users to recover their accounts by sending them a recovery email. The email contains a link that redirects the user to a recovery page where they can reset their password.

The process involves two key steps:

1. **Initiating the Recovery:** Using the `createRecovery` method to send a recovery email.
2. **Handling the Recovery:** Redirecting the user to the recovery page and completing the recovery process using the `userId` and `secret` parameters from the query string.

Development Stage

When your application is in the development phase and running locally, follow these steps:

1. Setup the Recovery Function

In your application, use the following code to initiate the recovery process:

```
import { Client, Account } from "appwrite";
import { v4 as uuidv4 } from "uuid";

const client = new Client()
    .setProject('<PROJECT_ID>'); // Replace <PROJECT_ID> with your Appwrite project ID

const account = new Account(client);

const createRecoveryEmail = async () => {
    try {
        const promise = await account.createRecovery(
            'email@example.com', // Replace with the user's email address
            'http://localhost:3000/recover' // Local recovery URL
        );

        console.log("Recovery email sent successfully:", promise);
    } catch (error) {
        console.error("Error sending recovery email:", error.message);
    }
};
```

```
createRecoveryEmail();
```

2. Handle the `/recover` Route

Create a route in your web app to handle the recovery process. For example, if you are using React:

React Example:

```
import { useSearchParams } from "react-router-dom";
import { Client, Account } from "appwrite";

const client = new Client().setProject('<PROJECT_ID>');
const account = new Account(client);

const RecoveryPage = () => {
    const [searchParams] = useSearchParams();
    const userId = searchParams.get("userId");
    const secret = searchParams.get("secret");

    const handleRecovery = async () => {
        try {
            await account.updateRecovery(userId, secret, '<NEW_PASSWORD>',
'<CONFIRM_PASSWORD>');
            console.log("Password successfully updated!");
        } catch (error) {
            console.error("Error updating password:", error.message);
        }
    };

    return (
        <div>
            <h1>Account Recovery</h1>
            <button onClick={handleRecovery}>Complete Recovery</button>
        </div>
    );
};

export default RecoveryPage;
```

3. Run the Application Locally

- Ensure your development server is running on `http://localhost:3000`.
- Use the local recovery URL (`http://localhost:3000/recover`) in the `createRecovery` function.

4. Test the Recovery Flow

- Trigger the recovery function (e.g., via a button or API call).
- Check your email for the recovery link.

- Click the link, which should redirect you to the `/recover` route in your app.
- Verify that the recovery process completes successfully.

Production Stage

When deploying your web application, update the recovery URL and ensure your hosting provider supports HTTPS.

1. Update the Recovery URL

Replace the local URL with your production URL in the `createRecovery` function:

```
const promise = account.createRecovery(
    'email@example.com', // User's email address
    'https://yourdomain.com/recover' // Production recovery URL
);
```

2. Deploy Your App

Deploy your application to a hosting provider (e.g., Vercel, Netlify, AWS). Ensure the following:

- Your app is served over HTTPS.
- You have a route or page at `/recover` that handles the recovery process.

3. Verify the Recovery Flow in Production

- Trigger the recovery process in the production environment.
- Check that the email contains the correct recovery link (e.g., `https://yourdomain.com/recover?userId=<USER_ID>&secret=<SECRET>`).
- Confirm that users can reset their passwords successfully.

Notes

1. **HTTPS Requirement:**
    - Appwrite may reject non-HTTPS URLs in production for security reasons. Ensure your deployed app uses HTTPS.
2. **Environment-Specific Configuration:**
    - Use environment variables for URLs to easily switch between development and production environments.

    ```
    const recoveryUrl = process.env.NODE_ENV === 'production'
        ? 'https://yourdomain.com/recover'
        : 'http://localhost:3000/recover';
    ```

3. **Error Handling:**
    - Implement proper error handling to inform users of issues during the recovery process.

Example Recovery Email Flow

1. User triggers the recovery process by entering their email.
2. Appwrite sends a recovery email containing a link like:

    ```
    https://yourdomain.com/recover?userId=<USER_ID>&secret=<SECRET>
    ```

3. User clicks the link and is redirected to `/recover`.
4. The `/recover` route parses the `userId` and `secret` parameters from the query string.
5. The app completes the recovery process and allows the user to set a new password.

Conclusion

This guide provides a complete solution for implementing account recovery using Appwrite in both development and production environments. Ensure you handle sensitive information securely and test the entire recovery flow thoroughly in both environments before deploying to production.