**Backend Development Plan for Released Email App**

**Phase 1: Core Setup & User Management**

**✅ Task 1.1 – Project Initialization**

* Create project structure: app/, models/, routes/, utils/, config/
* Install Flask, Flask-CORS, Flask-SQLAlchemy, Flask-Migrate, python-dotenv
* Create a config.py file for environment settings

**✅ Task 1.2 – Database Setup**

* Define User model with fields: id, name, email, auth\_provider, created\_at
* Set up SQLite or PostgreSQL with SQLAlchemy
* Apply migrations with Flask-Migrate

**✅ Task 1.3 – Authentication**

* Implement Google OAuth2 token verification route (using google-auth)
* Store users after successful token verification
* Return JWT or session token for frontend use

**Phase 2: Subscription Handling**

**✅ Task 2.1 – Store Subscription Scan Results**

* Define Subscription model: id, user\_id (FK), from\_name, from\_email, subject, category, unsub\_link, created\_at
* Create route to accept and store scanned results (bulk insert)
* Link each subscription to its user

**Phase 3: Unsubscribe Workflow**

**✅ Task 3.1 – Logging Unsubscribe Actions**

* Define UnsubscribeAction model: id, subscription\_id (FK), status, method, timestamp
* Create route to log when an unsubscribe is attempted or succeeded

**✅ Task 3.2 – Deleting Subscriptions**

* Make subscription database match up with unsub actions
* Remove duplicate subscriptions

**🔄 Task 3.2 – Process Gmail Unsubscribe Requests (Optional)**

* Store tokens with refresh support (encrypted)
* Implement server-side Gmail API interaction (for bulk unsubscribe filters)

**Phase 4: Categorization & Dashboard Handling**

**🔄 Task 4.1 – Categorization Logic (Optional)**

* Allow re-categorization or enhancement of classification using backend rules

**✅ Task 4.2 – Retrieve Dashboard Data**

* Create route to return total subscriptions, grouped by category
* Return category summary and individual subscription previews

**Phase 5: User Dashboard & API Polish**

**✅ Task 5.1 - User-specific API**

* Secure all routes to check current user (JWT/session)
* Add user profile route (get/update basic info)

**🔄 Task 5.2 - Analytics (Optional)**

* Return stats: unsub rate, most common senders, etc.
* Prepare routes to support future frontend charts

**Phase 6: Frontend Checks and Fixes**

**✅ Task 6.1 - Dashboard Page**

* Fix the subscription numbers on the categories

**🔄 Task 6.2 - Navigation**

* Have logged in/saved users go straight to dashboard (as home page)

**🔄 Task 6.3 - Design**

* Style the frontend and incorporate dark mode

**Phase 7: Mobile-Backend**

**✅ 1. Deploy your Flask backend**

If you haven’t already:

* Use platforms like **Render**, **Railway**, **Fly.io**, or **DigitalOcean App Platform** (easy Flask hosting).
* Make sure your Flask app is accessible via HTTPS (e.g. https://api.releasedapp.com).

**✅ 2. Update API URLs in Expo**

In your RN-Expo project:

js

CopyEdit

// src/api/axiosInstance.js

import axios from 'axios';

export const api = axios.create({

baseURL: 'https://api.releasedapp.com', // replace with your backend URL

});

**✅ 3. Use Secure Storage for JWTs**

Instead of localStorage, use Expo's SecureStore:

bash

CopyEdit

npx expo install expo-secure-store

js

CopyEdit

// src/utils/token.js

import \* as SecureStore from 'expo-secure-store';

export async function saveToken(token) {

await SecureStore.setItemAsync('jwt', token);

}

export async function getToken() {

return await SecureStore.getItemAsync('jwt');

}

export async function deleteToken() {

await SecureStore.deleteItemAsync('jwt');

}

**✅ 4. Connect login/register flows**

Make sure your Expo app can:

* Log in → save JWT → make authenticated requests.
* Register → same as above.
* Logout → clear token.

**✅ 5. Gmail OAuth for mobile**

Use Expo’s AuthSession to start the Google login:

bash

CopyEdit

npx expo install expo-auth-session

Example:

js

CopyEdit

import \* as AuthSession from 'expo-auth-session';

// use AuthSession.startAsync to begin the Google login flow

Then send the code or access\_token to your backend's /auth/google route.

**✅ 6. Test key flows**

From your mobile device/emulator:

* ✅ Fetch subscriptions
* ✅ Unsubscribe
* ✅ View stats (placeholder or real)
* ✅ Access premium/settings