**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**

Here’s a beginner‑friendly, step‑by‑step guide to wrapping your existing React app in Capacitor so you can run it on iOS & Android devices without rewriting it in React Native.

**1. Prerequisites**

1. **Node.js & npm** installed
2. **Android Studio** (for Android)
3. **Xcode** on macOS (for iOS)
4. A **physical device** or emulator for each platform

**2. Install Capacitor**

In your React project root:

bash

Copy code

npm install @capacitor/core @capacitor/cli --save

npx cap init

* **App name**: e.g. Released
* **App ID**: e.g. com.yourcompany.released

This creates a capacitor.config.json.

**3. Build Your Web App**

Every time you update your React code, you need to rebuild your production bundle:

bash

Copy code

npm run build

This outputs into build/ (Create‑React‑App default).

**4. Add Native Platforms**

From your project root:

bash

Copy code

npx cap add android

npx cap add ios

You’ll now have two new folders:

* android/ → Android Studio project
* ios/ → Xcode project

**5. Copy Web Assets into Native Projects**

After each npm run build:

bash

Copy code

npx cap copy

This syncs your latest React output into both native wrappers.

**6. Open & Run on Android**

1. **Connect** your Android device (or start an emulator).
2. Run:

bash

Copy code

npx cap open android

1. Android Studio will launch.
2. Click the ▶️ “Run” button to install & launch on your device.

**7. Open & Run on iOS**

**macOS only**

1. Connect your iPhone (or start the Simulator)
2. Run:

bash

Copy code

npx cap open ios

1. Xcode will launch.
2. Select your device and click ▶️ “Run”.

**8. Live‑Reload During Development**

To see changes without rebuilding each time:

1. In one terminal, start React:

bash

Copy code

npm start

1. In another, run Capacitor’s live‑reload plugin:

bash

Copy code

npm install @capacitor-community/live-reload

npx cap sync

1. Configure capacitor.config.json:

json

Copy code

{

"server": {

"url": "http://YOUR\_LOCAL\_IP:3000",

"cleartext": true

}

}

1. Rebuild the native project once (npx cap copy), then open & run it. It’ll pull from your dev server.

**9. Testing & Debugging**

* **Android**: Logcat in Android Studio; or adb logcat.
* **iOS**: Xcode console; Safari’s Web Inspector (Develop → Your Device).

Verify:

* Google OAuth redirects
* API calls (CORS must allow your device IP)
* Navigations and UI

**10. Next Steps**

* Tweak **layout/CSS** for small screens
* Add **native plugins** if you need camera, storage, etc.
* Prepare **release builds** when ready to publish