Table of Contents

GitHub Username: geocohn

Nutriscope

Description

This app helps you discover detailed information about packaged foods.

Intended User

It is a useful tool for consumers with nutritional or dietary concerns.

Features

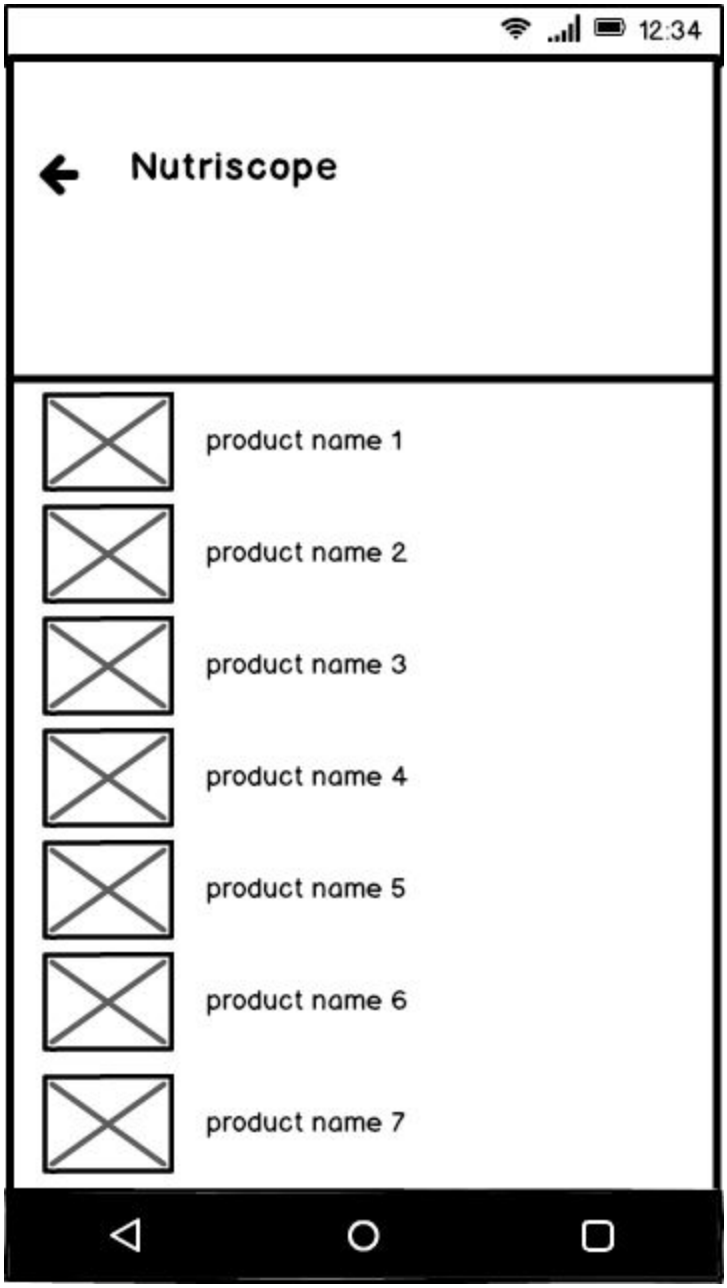
- Search the Open Food Facts database and receive a list of matching items
- Searches may include UPCs
- UPCs may contain wildcards
- The app includes an optical scanner for acquiring UPCs
- View the list via a widget
- Open an extensive detail view for any item

User Interface Mocks

Search screen

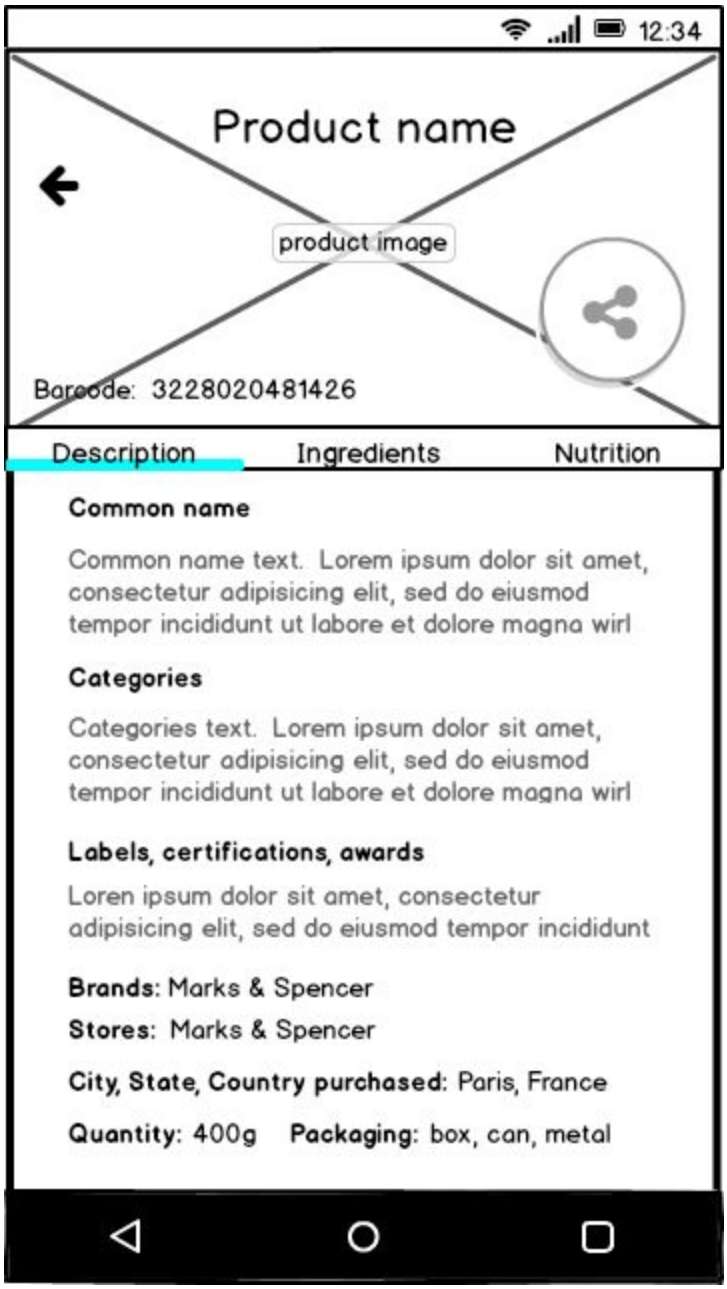
The main activity and starting point for the app.

Search Results screen



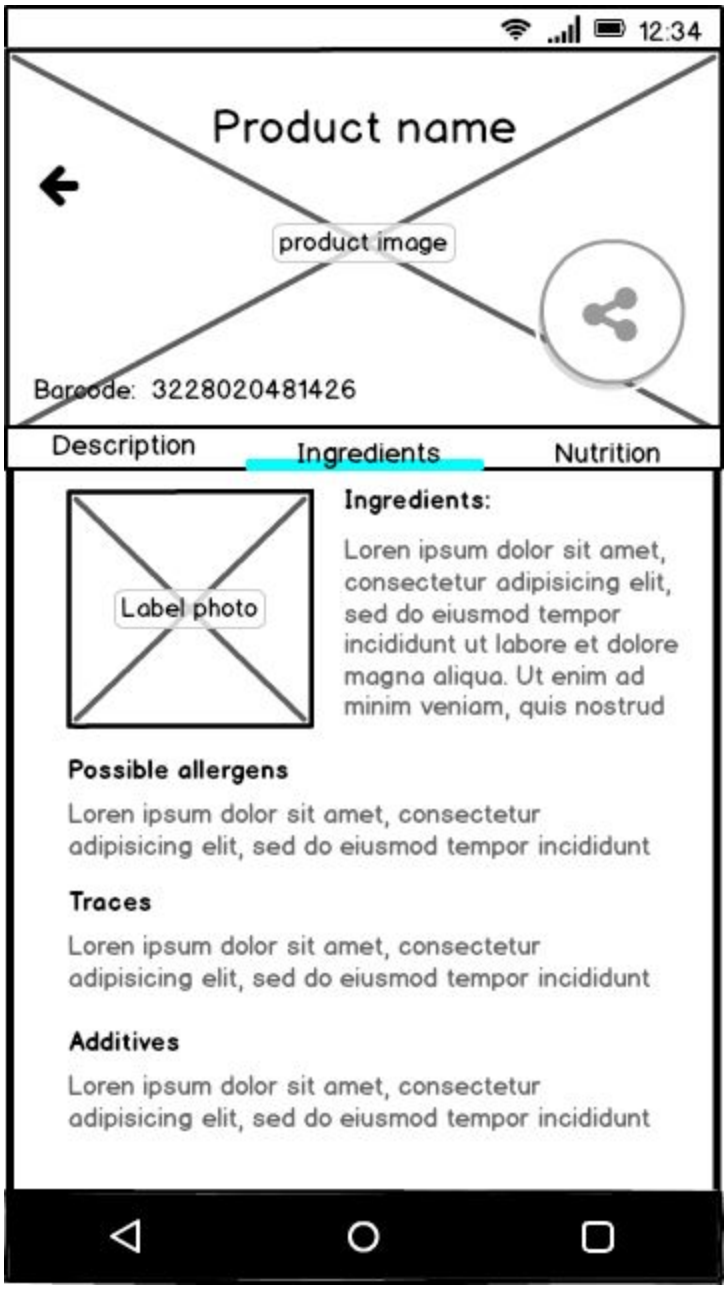
A RecyclerView list of the search results.

Detail screen, Description tab



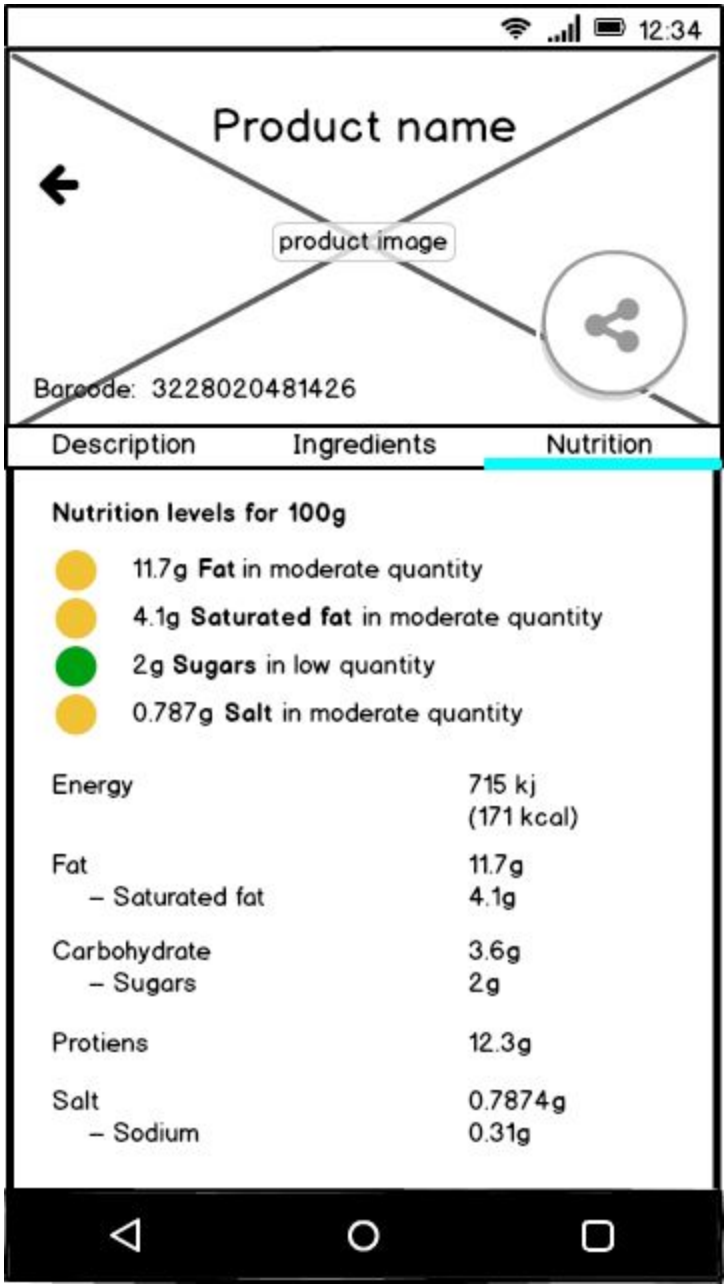
Miscellaneous description of the product.

Detail screen, Ingredients tab



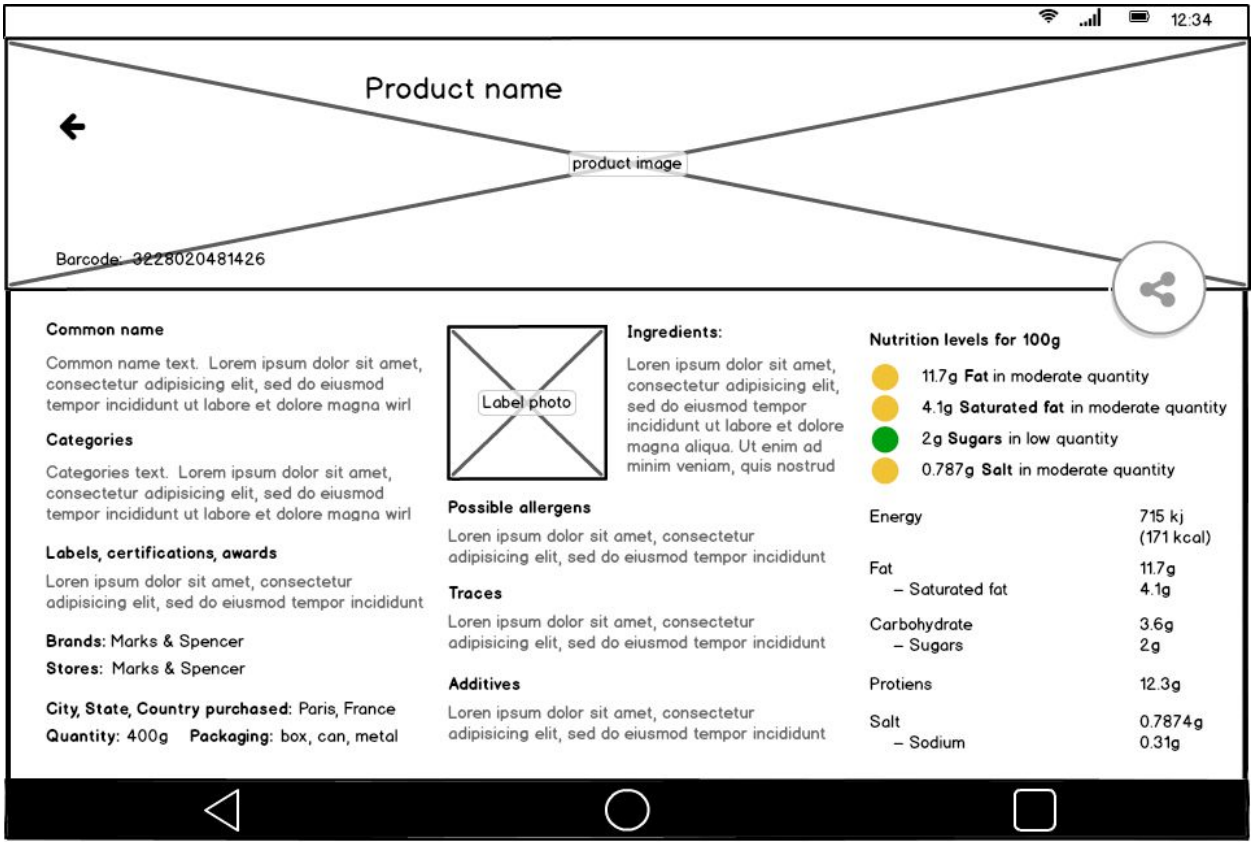
A complete list of the product's ingredients, with advisories for allergens, additives, and trace ingredients.

Detail screen, Nutrition tab



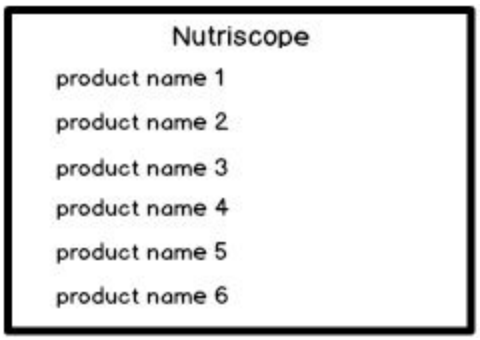
Nutritional analysis of the product.

Detail screen, wide devices



All panels detail screen for large devices in horizontal orientation.

Widget



Simple widget layout showing the most recent search results. Clicking the top of the widget takes the user to the Search Results screen. Clicking a single item in the widget takes the user to the Detail screen.

Key Considerations

How will your app handle data persistence?

The app will use Open Food Facts API as the primary data source. It will maintain the information internally via a content provider implemented as a relational database in sqlite database.

Describe any corner cases in the UX.

Special care will have to be taken to ensure the selected list item stays on the screen when returning from the detail screen after the device is rotated.

Describe any libraries you'll be using and share your reasoning for including them.

- Picasso will be used to fetch, cache, and display images.
- Retrofit will be used to retrieve HTTP data.
- AdMob will be used to display an ad on the search screen.
- Google Play Services Analytics will be used to view how the user is using the app.
- Junit will be used for unit testing involving JVM-only (no Android) modules.
- Espresso will be used to integration and Android feature testing.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and decompose them into tangible technical tasks that you can complete incrementally until you have a finished app.

Task 1: Project Setup

The app's backbone will be the sqlite database that the app populates using the Open Food Facts API. The API calls and database will be implemented and tested first.

The next step will be to implement and test a Content Provider that uses the sqlite database. This will be the app's single point of communication with the data.

Task 2: Implement Main Activity UI

The main screen is the search UI. Here the user can choose between two types of searches:

1. Optically scan a UPC code or enter it by hand, and receive a detail view for that particular product.
2. Specify for the name of a product or ingredient name, and receive a list of products that match.

Task 3: Implement UI for Product List and Detail

For wide devices, the detail views will share the screen, otherwise they will be implemented as tabs using a ViewPager.

Each master view will implement a RecyclerView, and both master and detail views will be filled using the Content Provider via a cursor loader.

Task 4: Communicate with the Outside World

Implement a FAB in the detail view that allows the user to share the data as text via email, social media, etc.

Task 5: Implement a Widget

The widget will show an abbreviated view of the list from the most recent search, even if it was a singleton result.

Task 6: Instrument the App with Google Analytics

This will allow us to understand where the user spends time in the app, and how much time.

Task 7: Create Free and Paid Flavors

Implement ads at the bottom of the main screen using AdMob for the free flavor but not for the paid flavor.

Task 7: Create and Test a Release Version

Implement a signed Release version with debug logging disabled, and run all test cases.

Task 8: Beta Test

Share the app with interested parties and incorporate important or low-risk feedback.

Task 9: Submit the App for Review

Implement a signed Release version with debug logging disabled, and run all test cases.