Green List: Software Design Document

CS-4850 Section I

Spring 2026

Professor Perry

February 1st 2026

SP-5 Green: Tate Swidorski, Kevin Gomes, Chris Heyward

Table of Contents

1. Introduction and Overview

1.1 Document Outline

This document describes the software design for the Green List (grocery list) application. It outlines the architecture, constraints, and tools that shall be used in order to implement this system. It shall also go over some of the design principles and strategies used during development.

1.2 System Overview

The Green List system consists of a Flutter-based mobile frontend, Firebase Authentication for user login and account management, and Firebase Database for storing grocery lists and list items. The system also includes a Spring Boot backend application used to support system functionality and manage backend processing. Users interact through the mobile interface while grocery list data is stored and synchronized through Firebase in real time.

2. Design Considerations

2.1. Assumptions and Dependencies

- Users must have internet access for real-time synchronization.
- Users must have a valid email account for sign-up/login.
- Firebase services must remain available.
- Flutter SDK must be installed for development.
- Spring Boot may be required if backend APIs are implemented.

2.2. General Constraints

- The app must run on iOS devices (Android support is TBD).
- The system must support real-time updates.
- The system must restrict access to lists only to authorized users.
- Firebase will be used as the primary database and authentication provider.
- UI design may follow Figma prototypes.

2.3. Development Methods

The Green List application will be developed using an incremental and iterative development process. The team will begin by implementing and testing the basic features first such as the CRUD functionality for grocery list items and grocery lists. Afterwards the Firebase authentication system shall be integrated and implemented. Collaboration features will be added after the authentication system is implemented, allowing users to now invite others to collaborate on their grocery lists. The collaboration features will be tested and reviewed as this may be the trickiest thing about this project. Once the main system is complete, the team will focus on making improvements to the UI and usability.

If time permits it, the team can also begin working on phase 2 features. These features may include autocomplete item searching, pricing estimate features, and possible

location-based functionality. Phase II implementation is dependent on available development time and complexity of the features.

3. Architectural Strategies

3.1 Technology Stack Strategy

The Green List system will be built using Flutter for the mobile frontend, Firebase Authentication for user login, and Firebase Database for storing grocery list information. These tools were chosen because they integrate well together and reduce the amount of custom backend infrastructure required.

3.2 Authentication Strategy

Firebase Authentication will be used to handle account creation, login, and logout. This design decision prevents the need for the team to manually store passwords or implement session handling, improving system security and reducing development complexity.

3.3 Data Storage Strategy

Firebase Database will be used to store grocery lists, list items, and collaborator permissions. Firebase was selected because it supports real-time data synchronization, which is required for the collaboration features of the system.

3.4 Real-Time Collaboration Strategy

The system will rely on Firebase's real-time update functionality to synchronize grocery list changes between multiple users. When an item is added, edited, or removed, the update will automatically appear for all collaborators who have access to the list.

3.5 System Expansion Strategy

The system will be designed in a modular way so that future Phase II features may be added if time permits. These features may include autocomplete searching, pricing estimates, and location-based features.

4. System Architecture

The Green List system is divided into multiple subsystems based on responsibility. The system consists of a Flutter mobile frontend, Firebase Authentication for login and account management, Firebase Database for storing grocery list information, and a Spring Boot backend application that supports backend processing.

The Flutter frontend is responsible for user interaction and displaying application screens such as login, list selection, and list editing. Firebase Authentication ensures only valid users can access the system. Firebase Database stores grocery list data and provides real-time synchronization so multiple users can collaborate on the same list. The Spring Boot backend application provides additional backend support and allows the system to expand in functionality if needed.

These subsystems work together to provide a secure, real-time grocery list application that supports multiple lists and collaboration between users.

5. Detailed System Design

5.1. Authentication Component

Classification: Frontend component (Flutter) using Firebase Authentication.
Role: Handles user signup, login, and logout functionality.
Responsibilities:

- Allow users to create an account using email and password
- Allow users to log in and log out securely
- Prevent unauthorized access to grocery lists

Constraints:

- Requires internet access
- Requires valid email and password credentials

5.2. Lists Page Component

Classification: Frontend component (Flutter) with access to Firebase Database.
Role: Displays all grocery lists the user owns or collaborates on and allows list management.
Responsibilities:

- Display all grocery lists available to the user
- Allow users to create new grocery lists
- Allow users to delete grocery lists they own
- Allow users to select a list to open

Constraints:

- User must be logged in to access this page
- Users may only delete lists they own

5.3. Grocery List Detail Component

Classification: Frontend component (Flutter) with access to Firebase Database
Role: Displays grocery list items and allows users to add, edit, and delete items.
Responsibilities:

- Display all items in a selected grocery list
- Allow users to add grocery items
- Allow users to edit grocery items
- Allow users to delete grocery items
- Sync item updates in real time for collaborators

Constraints:

- User must have access to the grocery list
- Item updates require internet access for real-time syncing

5.4. Collaboration Component

Classification: Frontend component (Flutter) with access to Firebase Database
Role: Allows grocery list owners to manage collaborators on a grocery list.
Responsibilities:

- Allow list owners to invite collaborators
- Allow list owners to remove collaborators
- Store collaborator access permissions in the database

Constraints:

- Only list owners can invite or remove collaborators
- Collaborators must have an existing account

5.5. Firebase Database Component

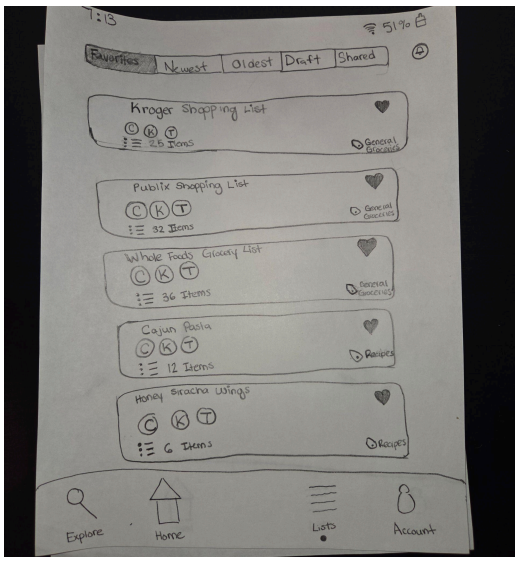
Classification: External database service (Firebase Database)
Role: Stores grocery lists, grocery list items, and collaborator permissions while supporting real-time synchronization.
Responsibilities:

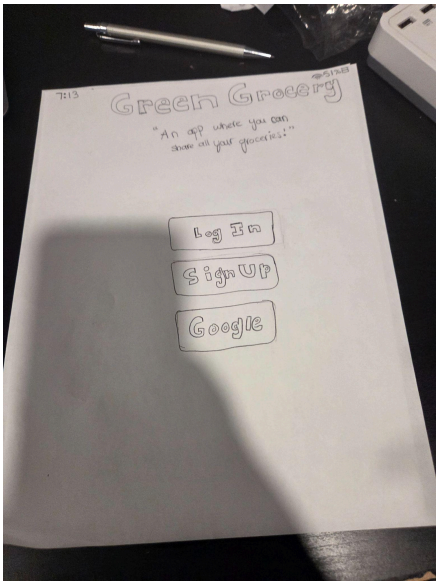
- Store grocery list data and item data
- Store collaborator access information
- Support real-time updates across all connected users

Constraints:

- Database security rules must restrict access to authorized users only
- Internet access is required for real-time collaboration features

6. User Interface Design





7. Glossary

CRUD: Create, Read, Update, Delete operations used for managing lists and items.
Flutter: A mobile development framework used to build the Green List frontend.
Firebase Authentication: A Firebase service used to manage user login, signup, and logout.
Firebase Database: A Firebase service used to store grocery lists and items, supporting real-time updates.
Collaborator: A user who has been invited to view and edit a grocery list.

Real-Time Synchronization: Automatic updating of grocery list changes across all users without refreshing.
Spring Boot: A Java backend framework used for the backend application.
Frontend: The part of the system the user interacts with (Flutter mobile app).
Backend: The part of the system responsible for processing and support services (Spring Boot and Firebase).

8. Bibliography

Flutter Documentation: https://docs.flutter.dev/
Firebase Documentation: https://firebase.google.com/docs/flutter
Spring Boot Documentation:https://docs.spring.io/spring-boot/docs/3.2.5/reference/htmlsingle/