Software Requirement Specification

SP-05 - Grocery List App
CS 4850 - Section 03: Fall 2025
Sep 14, 2025

			
Drew Claerbout Documentation	Jose Garcia Documentation	Tyler Chetty Developer	Aman Bhayani Developer

Team Members:

Roles	Name	Role	Cell Phone / Alt Email
Documentation	Drew	Responsible for project reports, risk assessment, and SDLC documentation.	770-283-0379 prowlersdc@gmail.com
Documentation	Jose	Creates design diagrams, screen mockups, and compiles final report package.	678-761-4883 gjose8120@gmail.com
Developer	Tyler	Focus on feature integration (grocery list) and testing.	470.363.0302 tyler.chetty7@gmail.com
Developer	Aman	Develop basic app structure and UI. Set up user authentication.	404.884.4149 amanbhayani608@gmail.com
Project Owner or Advisor	Sharon Perry	Facilitate project progress; advise on project planning and management.	770.329.3895 Sperry46@kennesaw.edu

Table of Contents

1.0 Introduction

1.1 Overview

The Grocery List app is created to help users plan and manage their shopping necessities. This app will allow users to both create and customize grocery lists. The app goes into such detail that quantity, and notes are featured. Pricing goes as far as to be updated as you add items to your list. The app keeps all shopping information in one place and will save time while shopping.

1.2 Project Goals

The goal of this project is to keep organization in grocery shopping as simple as possible for the user. Reducing time in the grocery store is a part of being as efficient as possible, this is a factor the user will surely appreciate.

Technical goals include adding a user authentication system. This allows all other users within the family or party to know who added what. Another goal includes developing an interface that supports quick item entry, real time updates.

1.3 Definitions and Acronyms

- SRS: Software Requirements Specification
- UI: User Interface
- UX: User Experience
- FAB: Floating Action Button
- Supabase: A backend as a service platform commonly used for authentication, databases, and cloud functions
- Widget: Fundamental building block of a flutter application's UI

1.4 Assumptions

- Users will have access to an Android or iOS device
- Users will have an internet connection for account creation, sharing lists, and synchronizing changes
- Users will have a valid email address for account registration and authentication
- The application will rely on a third-party backend service (Supabase) for real-time functionality

2.0 Design Constraints

2.1 Environment

This app works on Android and iOS. Internet access will be needed to update changes made by the user when dealing with a shared list. Otherwise accessing the list will not require internet access. Flutter is the framework in which this app is developed. This app is also developed using Dart.

2.2 User Characteristics

Target users include those who are looking for convenient ways to organize their shopping experience. These users will not have to be experts in navigating through their phone, the app is user friendly and minimal. These users will most likely be multitasking when shopping therefore needing a simple minimal app.

2.3 System

The system will be a client-server application. On the client-side, a mobile application built with the Flutter framework, running on Android and iOS systems. The server side will use Google Supabase, handling data storage, user authentication, and real-time synchronization.

3.0 Functional Requirements

3.1 User Authentication

- 3.1.1 The system shall allow a user to create a new account using an email address and a password.
- 3.1.2 The system shall allow a user to log in using their registered email and password.
- 3.1.3 The system shall provide a "Forgot Password" feature to allow users to reset their password via email.
- 3.1.4 The system shall allow a user to log out of their account.

3.2 List Management

- 3.2.1 The system shall allow an authenticated user to create a new grocery list.
- 3.2.2 The system shall allow a user to view a list of their owned and shared grocery lists on a home screen.
- 3.2.3 The system shall allow a user to edit the title of a list they own.
- 3.2.4 The system shall allow a user to delete a list they own.
- 3.2.5 The system shall allow a user to mark a list as "complete" or "in progress".

3.3 List Sharing & Collaboration

- 3.3.1 The system shall allow the owner of a list to share it with other users via their email address.
- 3.3.2 The system shall provide real-time synchronization of all changes (add, edit, delete items) for all users viewing a shared list.
- 3.3.3 The system shall display the name of the user who added or last modified an item.
- 3.3.4 The system shall allow the list owner to remove users from a shared list.

3.4 Item Management

- 3.4.1 The system shall allow a user to add a new item to a list.

- 3.4.2 The system shall allow a user to edit an item's properties: name, quantity (e.g., "2", "500g"), and notes (e.g., "organic", "for birthday cake").
- 3.4.3 The system shall allow a user to delete an item from a list.
- 3.4.4 The system shall allow a user to mark an item as "purchased" or "not purchased" (e.g., with a checkbox).
- 3.4.5 The system shall allow for quick item entry, suggesting recently added or frequently used items.

3.5 Pricing Feature

- 3.5.1 The system shall allow a user to manually enter a price for an item.
- 3.5.2 The system shall automatically calculate and display a running total cost for all items on the list.
- 3.5.3 The running total shall update in real-time as items are added, removed, or their prices/quantities are changed.

3.6 Navigation

- 3.6.1 The system shall provide a clear and consistent navigation structure (e.g., a bottom navigation bar or drawer) to access the Home (lists) screen, and user Profile/Settings.
- 3.6.2 The system shall allow users to tap on a list from the home screen to navigate to the detailed item view for that list.

4.0 Non-Functional Requirements

4.1 Security

- User passwords shall be hashed and salted before storage in the database (handled automatically by Supabase Auth).
- Data access shall be governed by security rules on the backend to ensure users can only read and write data for lists they own or are shared on.
- All communication between the client and server shall be encrypted using HTTPS/SSL.

4.2 Capacity

- The system shall be designed to handle a minimum of 10,000 concurrent users on shared lists. The scalability will be managed by the Supabase platform.
- The application should perform efficiently on devices with limited storage and memory.

4.3 Usability

- The User Interface shall be intuitive and require minimal instruction to use.
- The app shall follow modern Material Design guidelines on Android and Cupertino (iOS-style) guidelines where appropriate for a native feel on both platforms.
- Task completion times for core functions (e.g., adding an item) shall be under 3 seconds, including network latency.

4.4 Other

- Performance: The application should launch in under 2 seconds on a mid-range device. List scrolling should be smooth at 60fps.
- Reliability: The application should have a crash-free rate of over 99.5%. Data loss due to sync conflicts or poor network conditions should be prevented.
- Portability: The single Flutter codebase shall run correctly on all supported Android (API 21+) and iOS (iOS 11+) versions without major UI inconsistencies.

5.0 External Interface Requirements

5.1 User Interface Requirements

- Authentication Screen: A simple form for login and registration.
- Lists Home Screen: A scrollable list of the user's grocery lists, with a floating action button (FAB) to create a new list.
- List Detail Screen: The main shopping view, displaying items, checkboxes, quantities, prices, and a total. Features a FAB to add a new item.
- Add/Edit Item Dialog/Sheet: A modal form for entering item details.
- Sharing Settings Screen: A screen to add or remove users from a list via email.

5.2 Hardware Interface Requirements

- The application shall utilize the device's touch screen for all user input.
- The application shall require access to an internet connection (Wi-Fi or Cellular data) for synchronization and authentication features.

5.3 Software Interface Requirements

- Backend Services: The application shall interface with Supabase:
- Supabase Authentication: For user login and account management.
- Database: Primary SQL database for storing lists, items, and user relationships.
- Operating System: The application shall interface with the Android and iOS SDKs via the Flutter framework.

5.4 Communication Interface Requirements

Communication with Supabase services shall be accomplished using standard HTTPS REST APIs and WebSocket connections for real-time listeners, as provided by the official supabase_core and supabase_auth Flutter plugins.