Software Design Document

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. Introduction and Overview

This document will describe the software design of our cross-platform grocery list app which lets families create a shared list for items they would like to purchase from a grocery store.

The core features of the app include:
- Account creation and authentication.
- Group (family) creation with invitations.
- Real-time shared grocery lists (add/edit/delete items).
- Each member can add, edit, delete items; changes appear instantly to all members.
- Integration with third-party grocery store data (products, prices, availability).

Project scope:
This SDD covers architecture, detailed component design, data models, interfaces, constraints, security, testing & validation, and deployment guidance and suggested roadmap for enhancements.

2. Design Considerations

Goals & Priorities:
- Real-time collaboration: Minimal latency, consistent state across devices.
- Simplicity and UX: Easy onboarding, intuitive list interactions.
- Reliability and offline: App usable offline, syncs when online; conflict handling.
- Extensibility & modularity: Easy to add store integrations, analytics, monetization channels.
- Cross-platform: Single codebase for iOS & Android using Flutter.
- Scalability: The architecture should support multiple families, each with multiple users. Security: Authentication and row-level security (RLS) policies in Supabase will ensure that only authorized family members can view/edit their lists.

Key Scenarios:
- User creates an account, creates a family group, and invites members.
- A user adds an item to the list, and all members see the change instantly.
- A user edits or marks an item purchased, list updates in realtime.
- Offline changes are queued and synced when connection returns; conflicts resolved deterministically.
- Users browse nearby grocery stores via third-party data; optionally add items to list or place affiliate orders.

2.1. Assumptions and Dependencies

Assumptions:

- The primary client is Flutter (Dart). Backend uses managed cloud services
- Users have smartphones with intermittent connectivity.
- Group sizes are modest per list (a typical group consists of less than 10 members).

Dependencies:

- Supabase services: Authentication, Postgres SQL- real-time database.
- Third-party grocery APIs (store product catalog/price/availability): vendor APIs or affiliate partners.
- Push notification provider (done with the help of supabase).
- Hardware: iOS and Android (mobile) devices.
- OS: iOS 14+ and Android 10+.

2.2. General Constraints

- Hardware/Software Environment: Runs on IOS/Android devices with limited CPU/GPU, battery, and memory
- End-User Environment: Users may be on the go or have limited interactivity (limited attention). Screen sizes may affect experience.
- Availability/Volatility of Resources: Mobile OS may kill the app (through crashes). Network and battery fluctuations.
- Standards Compliance: OAuth 2.0 / OIDC flows for social auth; platform store policies (e.g., “Sign in with Apple” on iOS when using third-party logins); secure transport (TLS).
- Interface/protocol requirements: Supabase Realtime uses WebSockets. DB access via HTTPS (REST/RPC) and PostgREST. Push notifications use APNs/FCM (future)
- Memory/Capacity Limits: Mobile memory pressure. Supabase plan quotas.
- Verification & Validation: Must validate RLS and concurrency; mobile UI across OS versions.
- User Management: Household roles (admin vs standard).
- Database: Supabase Postgres hosting.
- Performance: Real-time updates must propagate within <1s latency.
- Security: Use Supabase Row-Level Security (RLS) to ensure data isolation per group.
- Network: Requires active internet connection; offline mode may be a later enhancement.
- Interoperability: Must handle multiple OAuth providers (Google, Apple, etc.) via Supabase Auth.

2.3. Development Methods

Our approach is to use the Agile methodology with iterative sprints for a faster output. The team members work on their own branch and are later integrated into the main branch as needed. The tools that will be utilized are Flutter (for frontend), Supabase (for backend), and Github (for version control). We also think a code review would be beneficial by having at least one team member look at what the other has developed. Moreover, we will also refer to Flutter documentation to ensure using best practices.

3. Architectural Strategies

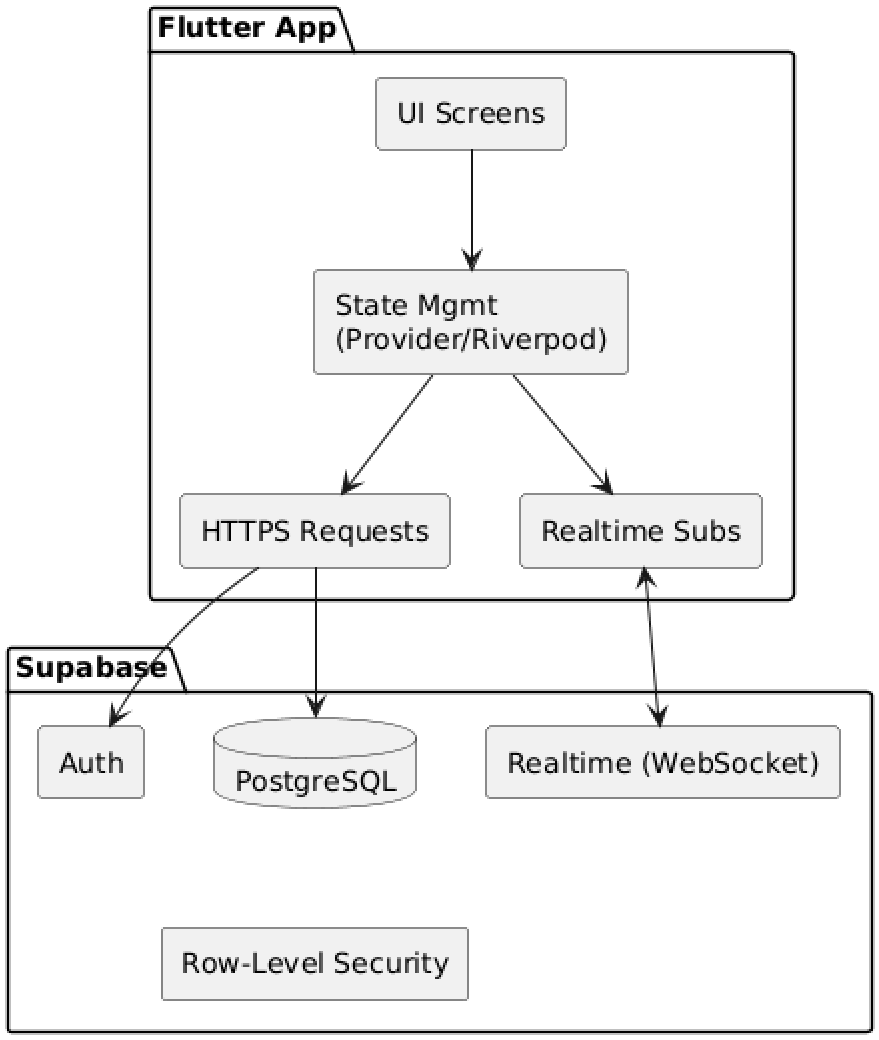
- Programming language and framework(s): Flutter (Dart) for cross-platform development with a single codebase, consistent UI, and near-native performance.
- Database & Backend: Supabase for authentication, PostgreSQL storage, and real-time synchronization. Considered Firebase but rejected for limited SQL flexibility.
    - System leverages Supabase for user authentication, PostgreSQL data storage, and real-time synchronization, thereby eliminating the need to develop a custom backend.
    - Flutter’s Material and Cupertino widget libraries are utilized to provide a consistent, native-like user interface while minimizing redundant UI development.
    - Flutter’s testing framework and GitHub Actions are integrated to support automated validation and continuous integration
- Future plans focus on expanding functionality and improving scalability:
    - Refine Third-Party API Integration: Redefine connections with grocery store APIs to display real-time item prices and availability.
    - Performance Improvements: Optimize database queries and network synchronization to support larger user groups with minimal latency.
    - Advanced Features: Add predictive item suggestions and analytics to further streamline the shopping experience.
- The system follows a mobile-first design using Flutter’s Material Design (Android) and Cupertino (iOS) widgets to ensure a consistent, native-like experience across platforms.
    - Input Model: Users interact primarily through touch gestures, including taps, swipes, and long presses. Forms and dialogs are used for item creation, editing, and authentication.
    - Output Model: The application provides real-time visual feedback, automatically updating list views when items are added, edited, or marked as purchased. Status indicators ensure clarity during network operations.
    - Navigation: A bottom navigation bar or drawer offers quick access to core screens (Home, List Detail, Profile/Settings).
- Software Interfaces: The client application communicates with Supabase services over HTTPS for authentication and database operations, and uses WebSockets for real-time synchronization. The app runs on Android and iOS, leveraging their native SDKs through the Flutter framework to ensure consistent performance and UI rendering.

- Error detection and recovery:
  - o Input Validation: User input is validated client-side to prevent malformed data from reaching the backend.
  - o Network Error Handling: Connectivity issues are detected and communicated to the user through alerts and retry prompts.
  - o Authentication Errors: Failed login or registration attempts return descriptive error messages while protecting sensitive information.
  - o Data Consistency: Supabase's real-time synchronization automatically resolves conflicts by applying the most recent valid update.
- External databases and/or data storage management and persistence: Supabase (PostgreSQL) as the central data store for users, households, lists, and items. All data access is secured with Row-Level Security and transmitted over HTTPS. Real-time updates are handled through WebSocket subscriptions, ensuring sub-second synchronization across devices.
- Concurrency & Synchronization: System relies on Supabase's real-time subscriptions to synchronize list and item changes across all connected clients.
  - o Updates are applied using a last-writer-wins strategy based on server timestamps to prevent conflicts.
  - o On the client side, optimistic UI updates are used so users see changes immediately while the server confirms them in the background. Stream subscriptions are paused when a screen is backgrounded to reduce resource usage and resubscribed when resumed.

4. System Architecture

High-level components:

1. Frontend (Flutter):
   a. Authentication screen (Sign in, Register, Google Auth)
   b. Family or group management screens
   c. Grocery List screen
   d. Third party integration with other grocery stores screen.
2. Backend:
   a. Auth: Handles user registration, login, and session management.
   b. Database:
      i. Users Table
      ii. Families Table
      iii. GroceryItems Table
   c. Storage: For images and files.
3. External Services:
   a. Grocery store(s) APIs.



5. Detailed System Design

5.1. Classification

- Subsystem: Subsystems included in the design of the app include a sign-in prompt at the beginning of the app.
- Module:
  - “Start a new family,” “Group 1,” “Family XYZ,” “My own list,” are all functions within the subsystem. These are functions where you can interact and start adding items to the specific list chosen.
- Function:
  - addItem()- Functions within the module include the addItem() where the program will add desired wants from the user.
  - removeItem()- In the case that the user will want to delete an item from a list they are able to do so by sliding left on the unwanted item.

    - updateItem() - The grocery list will need to be updated in real time there for this method, it will be used to update
- Package/ File - Application is implemented using Flutter (framework) with Dart (language). The codebase is organized into separate Dart files for UI screens, business logic, and service integrations (Supabase API calls).

5.2. Definition

- The grocery list app is responsible for organizing items the user has added to a list. Furthermore, keeping a list of multiple users in a family or group. The app will be responsible for creating new grocery list, adding, updating, and deleting items
- Authentication System - The user will be prompted to sign in with email and password.
- Sharing Feature - When working on a shared list, multiple users are able to add items to the list. This feature allows this to be possible

5.3. Constraints

- Authentication Module: Requires internet; Supabase session tokens expire and must be refreshed.
- Supabase free plan:
    - Limited to 50,000 active users monthly (beyond needed, but limits future scalability)
    - Storage: 500 MB of storage is available, limiting how many grocery items and user accounts can be stored before needing an upgrade.
    - Row Level Security (RLS): needs to be enabled properly to prevent unauthorized access.
- API: Limited API for third party integration calls using a free version.

5.4. Resources

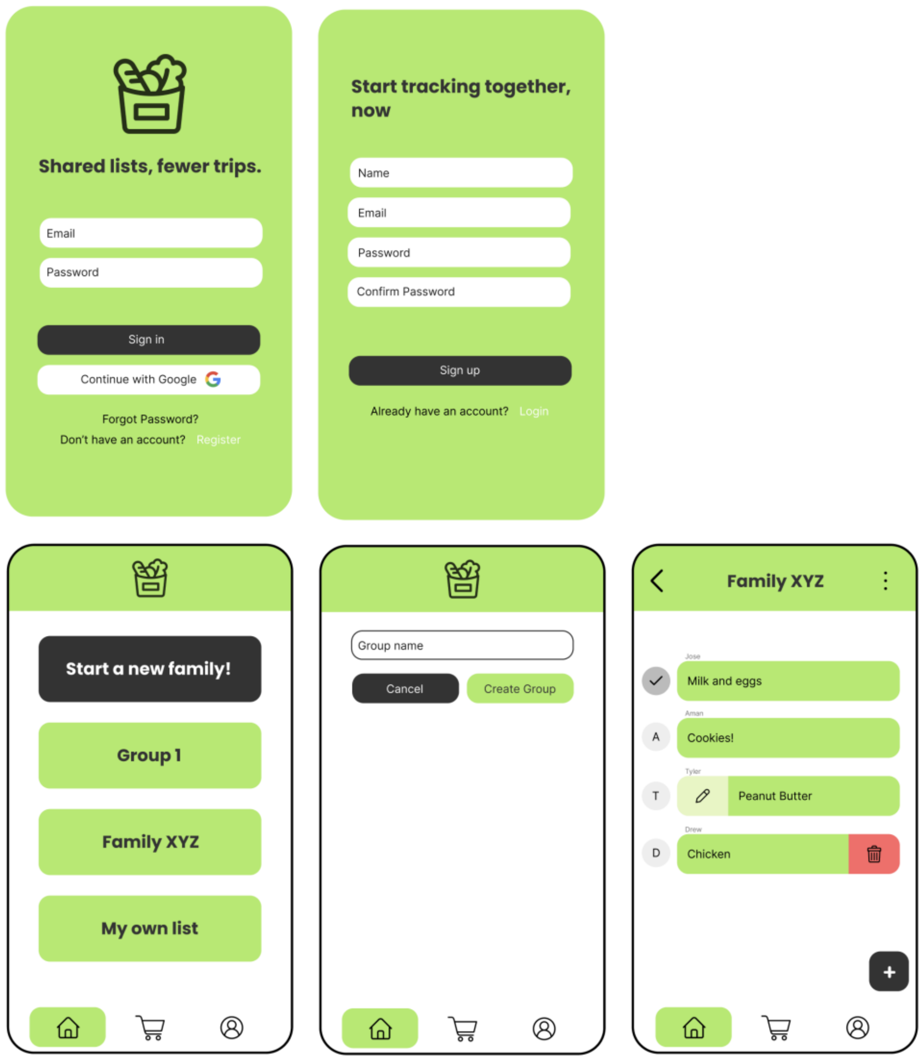
Authentication model/database: Supabase Auth and Postgres
Database Schema – Supabase PostgreSQL, RLS policies, triggers.
Third-Party Integration Module: Grocery store APIs.
Grocery list module: Flutter ListView and Supabase tables.

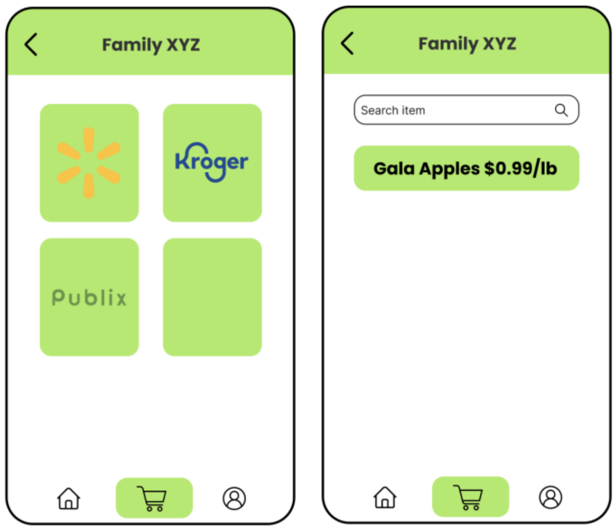
5.5. Interface/Exports

Services:

- createList() -This function is used to create a new list
- deleteList()- This function is used to delete a list
- sharedList()- This is used to access shared list where multiple users can add or delete items

- Items - objects that identify what items are which
- Int - integers are used to specify how much of an item is needed





6. Glossary

- Edge Functions: Serverless functions on Supabase for backend logic.
- RLS (Row-Level Security): Database-level security policies ensuring user-level data isolation.
- Supabase: Open-source Firebase alternative using Postgres.

7. Bibliography

"Flutter Documentation." *Docs*, docs.flutter.dev/.

Supabase. "Supabase Docs." *Supabase Docs*, supabase.com/docs.