GreenList

Final Report

CS-4850 Section I, Spring 2026

Professor Perry

April 2026

Tate Swidorski, Kevin Gomes, Chris Heyward

Website Link: https://greenlist-portfolio-website.vercel.app/

GitHub Link: https://github.com/GreenCTK/greenlist-app

Metric	Value
NDA	Not Applicable
Lines of Code	5,400 (TypeScript / TSX)
Project Components / Tools	18 components across 32 source files
Total Man Hours	135 hours
Status	95% complete — working as designed

Project Team

Roles	Name	Role
Team leader	Chris Heyward	Developer / Team Leader
Team members	Kevin Gomes	Developer
	Tate Swidorski	Documentation / Testing

Table of Contents

1. Introduction

1.1 Overview

Green List is a mobile grocery list application built using the flutter framework. The app will allow users to create and manage multiple grocery lists, collaborate on said lists with friends, and the lists will update automatically in real time.

1.2 Project Goals

- Provide a simple grocery list application for iOS (Android is TBD).
- Allow users to create an account and log in securely.
- Allow users to create, edit, and delete multiple grocery lists.
- Allow users to add, edit, and delete items within lists.
- Provide real-time synchronization of grocery lists across multiple users.
- Allow users to invite other users to join their grocery list (allow removal as well).
- Provide logout functionality.
- Provide a clean, user-friendly interface.

1.3 Assumptions

- The user must have internet access to use the application.
- The user must have an email in order to sign up.

1.4 Phase Overview

PHASE 1 — The overall project is to make a mobile grocery list application. This app will allow users to login, set up their own grocery lists, and collaborate on other people's lists. The lists will be able to update in real time when a user adds, deletes, or edits an item.

PHASE 2 — The remainder of the project will be adding features such as: autocomplete item names when searching, a way to calculate the potential cost of items, or even some location features. Phase 2 may be subject to change depending on the difficulty to implement them or the cost to afford them.

The project will be created using the Flutter framework, Spring Boot for backend development, and Firebase for the authentication and database systems.

1.5 Project Website

Project Website: https://greenlist.com

Disclaimer: this is subject to change

1.6 Deliverables

We will deliver a Final Report which will include the Software Requirements Specification and Software Development Design documentation. Include the source code, the link to the GitHub repository, and a video demonstration/presentation of the grocery list application and its features.

1.7 Collaboration and Communication Plan

We will be meeting weekly on Wednesdays at 9 pm on Discord. Where we will assign one another tasks, discuss progress, and manage overall workflow.

1.8 Version Control Plan

We will be setting up a group GitHub account where we can collaborate, utilizing different branches to organize and keep track of our tasks.

2. Requirements

2.1 Design Constraints

2.1.1 Environment

- The application will be developed as a mobile application using Flutter.
- The application shall support iOS devices (Android is TBD).
- Firebase shall be used for authentication and database services.
- Spring Boot will be used for backend API services.
- Figma may be used in order to help design UI.

2.1.2 User Characteristics

- Intended for general users.
- Roommates, family, friends are collaborators.

2.1.3 System

- Flutter mobile application.
- Firebase authentication.
- Firebase database.
- Spring Boot for backend.

2.2 Functional Requirements

2.2.1 Login / Authentication

- Create account.
- Login with username and password.
- Should be redirected to this page if the user is not logged in.

2.2.2 Your Lists Page

- Display all of your lists / lists you are a collaborator on.
- Add or remove lists from this page.

2.2.3 Inside a List

- Display all of the items on the list.
- Allowed to add, remove, and edit items on the list.
- Can invite collaborators on this page.
- List updates in real time.

2.3 Non-Functional Requirements

2.3.1 Security

- User authentication will be handled using Firebase Authentication.
- Users will only be able to manage and view lists they have access to.

2.3.2 Capacity

- The system shall support multiple grocery lists per user.
- The system shall support multiple items per grocery list.
- The system shall support multiple collaborators per grocery list.

2.3.3 Usability

- The system shall provide a simple and user-friendly interface.
- The system shall allow users to add, edit, or delete items with minimal steps.
- The system shall be usable without requiring training or technical knowledge.

2.3.4 Other

- Grocery list updates shall appear in real time across all users in a list.

2.4 External Interface Requirements

2.4.1 User Interface Requirements

- The system shall provide a login/account creation screen.
- The system shall provide a "Your Lists" page to display and manage grocery lists.
- The system shall provide a grocery list page showing all list items.
- The system shall provide buttons/options for adding, editing, and deleting grocery list items.
- The system shall provide an interface for inviting and removing collaborators.
- The system shall have a button to logout.

2.4.2 Software Interface Requirements

- The system shall interface with Firebase Authentication.
- The system shall interface with Firebase Database services.
- The system may interface with Spring Boot backend API services if needed.

2.4.3 Communication Interface Requirements

- The system shall require internet access for real-time updates.

3. Technology Pivot

3.1 Overview

During the early stages of development, the team made a significant decision to pivot away from the originally planned technology stack. The initial project plan called for Flutter as the mobile framework and Spring Boot as the backend API layer. After beginning development, the team decided to switch to React Native with Expo for the frontend and to rely entirely on Firebase services as the backend, eliminating the need for Spring Boot altogether.

3.2 From Flutter to React Native

The original plan specified Flutter as the mobile development framework. However, after beginning UI prototyping with Figma and v0, the team found that the generated component structures translated far more naturally into React Native than into Flutter. React Native's component model aligned more closely with what Figma and v0 produced, which allowed the team to move from mockup to working code much faster.

Switching to React Native with Expo also simplified the development environment setup and allowed the team to run the app quickly on both the iOS Simulator and physical devices via Expo Go without the additional configuration overhead that Flutter would have required.

3.3 Eliminating Spring Boot

The original design included Spring Boot as a backend API layer to sit between the mobile app and Firebase. After evaluating the project requirements, the team determined that Spring Boot was unnecessary. Firebase Authentication and Cloud Firestore together provided everything needed: user authentication, real-time database reads and writes, and security rules that enforce access control at the data layer.

By connecting the React Native frontend directly to Firebase services, the team was able to keep the architecture simple and avoid the overhead of building, deploying, and maintaining a separate backend server.

3.4 Impact on the Project

While the pivot did disrupt the initial flow of the project, it ultimately led to a more productive development process. The team was able to implement features faster and with fewer integration issues than the original stack would have required. All core project goals were still met, and the final application delivers on everything outlined in the original requirements.

4. Analysis / Design

4.1 Architecture Overview

GreenList follows a simple mobile architecture where the React Native frontend communicates directly with Firebase backend services. There is no intermediary application server.

The architecture consists of primarily three layers:

- Presentation Layer: React Native (TypeScript/TSX) screens and components, styled with React Native's built in styling system and react navigation, essentially the UI in general.
- Business Logic Layer: Custom React hooks and modules that manage authentication state, Firebase read/writes, and event listeners.
- Data Layer: Firebase Authentication for identity management and Cloud Firestore for list and item information.

4.2 System Components

Component	Technology	Purpose
Mobile Frontend	React Native	All UI screens, navigation, and user interaction
Language	TypeScript / TSX	Programming Language used for React Native
Authentication	Firebase Authentication	Email/password login, Google authentication, session management
Database	Cloud Firestore	Real-time database storage for lists, items, and users
Food Search	USDA FoodData API	Nutritional lookup and item autocomplete
UI Prototyping	Figma / v0	Wireframes and screen mockup design
Package Manager	Node.js / npm	Dependency management
Version Control	GitHub	Code management and team collaboration

4.3 UI Prototyping

Before React Native code was written, the team used Figma and v0 to generate mockups of all major screens. This included the login/signup screens, the lists overview page, the grocery list detail screen. The mockups served as a direct reference during frontend development.

5. Development

5.1 Development Process

5.1.1 UI and Prototyping

The first stage of development for Greenlist has focused primarily on designing the User Interface and planning the user experience. The team has used Figma and v0 to create UI mockups for the applications. These designs include screens for login, signup, viewing all of the users grocery lists, and inside of the grocery lists themselves.

Using Figma and v0 allows the team to define the structure and layout of the application before implementing it in code. The mockups are used as references when building the React Native interface. Changes to layout, navigation flow, and screen organization are first tested in the prototypes and then translated into React Native components during development.

5.1.2 Frontend Development Tools

The mobile application is being implemented using React Native. The user interface is built using React Native components such as View, Text, TextInput, Button, and FlatList. Each screen of the application, such as the login screen, list overview screen, and grocery list detail screen, is implemented as a separate component, typically managed through a navigation library such as React Navigation.

To help create the frontend interface, v0 is used to generate UI layout ideas and component structures. These generated layouts are then translated into React Native components and integrated into the application code. Components are organized into screens that allow users to navigate between viewing their grocery lists and editing list items.

5.1.3 Authentication System Implementation

The project implements user authentication using Firebase Authentication. This system allows users to create accounts and log in using their email and a password OR using google.

The authentication workflow includes the following development steps:

- Creating login and registration screens with help of Figma and v0.
- Connecting the application to Firebase Authentication.
- Validating user login credentials through firebase.
- Maintaining user session state within the application.

Once authentication is fully implemented, each Greenlist user account will be associated with the grocery lists that the user owns or has been invited to collaborate on.

5.1.4 Grocery List Management

The project implements grocery list management using the React Native frontend and Firebase Realtime Database. This functionality allows users to create grocery lists and manage the items within those lists.

The grocery list workflow includes the following development steps:

- Creating screens that display the user's grocery lists with help of React Native and v0.
- Connecting the application to the Firebase Database system.
- Allowing users to create new grocery lists through input fields in the application.
- Allowing users to add, edit, and delete grocery lists and list items.
- Updating the Firebase Database whenever a list or items is changed.

- Show the change in real-time.

Once grocery list management is fully implemented, users will be able to organize multiple grocery lists and manage the items within each list directly through the Greenlist application.

5.2 Database Connection

The Greenlist application connects to the Firebase Database to store and retrieve grocery lists and grocery list items. The database connection process includes the following development steps:

- Creating a database on firebase.
- Adding Firebase to the React Native project using the Firebase configuration files.
- Connecting the React Native application to Firebase Realtime Database.
- Reading grocery list and item data from the database.
- Writing new lists and item updates to the database when changes are made.

Once the database connection is fully implemented, grocery lists and items will be stored in Firebase and retrieved by the application when users access their lists.

5.3 Part 1 — UI Design and Prototyping

The first part focused entirely on designing the UI before writing any functional code. The team used Figma and v0 to produce screen-level mockups which covered:

- Login and signup screens.
- The main lists overview page.
- Grocery list screens.

Using Figma allowed the team to generate React components that were then translated into React Native components. The mockups allowed the team to think about the feel of the application, the navigation flow, and screen organization that carried through the entire project.

5.4 Part 2 — Frontend Development

Once the UI designs were finalized, the team began implementing the frontend using React Native with Expo and TypeScript. Each screen in the application was developed as a separate component, with navigation handled using a bottom tab navigation system. Core UI components such as lists, buttons, and input fields were built using React Native's built-in components. The frontend was structured to allow users to easily navigate between viewing their grocery lists, managing items, and accessing account features.

State management and data handling were implemented to ensure that user actions, such as adding or editing items, were immediately reflected in the interface.

5.5 Part 3 — Backend Integration

Firebase Authentication was used to handle user login and account management. The application allows two sign-in methods:

- Email and password: Users can create an account and log in using standard credentials.
- Google sign-in: Users can authenticate using their Google account through Expo's Google authentication integration.

User session state was managed in react using Firebase libraries, which can tell whether a user is logged in or not. Based on this, the application routes users to either the main application or to the login screen.

Cloud Firestore was used as the main database to store grocery lists, items, and user data. The frontend React Native application connects directly to Firestore using Firebase libraries.

Data is organized into collections for lists and items, and the app reads and writes this data whenever users create lists, add items, or make changes. These updates are handled directly in the frontend, allowing the app to stay simple without needing a separate backend server.

Real-time updates were implemented so that when a user makes a change (such as adding or editing an item), it is immediately reflected for all other users viewing the same list. This allows multiple users to collaborate on a list at the same time without needing to refresh the app.

Access to data is controlled using Firebase security rules, ensuring that only logged-in users and authorized collaborators can view or modify specific lists.

5.6 Part 4 — Collaboration, Favorites, and USDA Integration

The final development phase added the advanced features that differentiated GreenList from a basic list app:

- Collaboration: Users could add friends and then collaborate on lists together.
- Favorites: A heart icon on the list overview allows users to toggle a list as a favorite.
- USDA FoodData API: The Nutrition screen provides a search interface backed by the USDA FoodData Central AI. Searching results return nutritional information which are associated with grocery items.

5.7 Project Setup

In the event that a developer wants to download this project, the following steps are required to run it:

- Install Node.js and a package manager (npm or yarn).
- Install React Native Expo.
- Clone the repository from GitHub.
- Set up a Firebase project through the Firebase console.
- Enable Firebase Authentication and Firebase Database.
- Add the Firebase configuration files to the React Native project.
- Run the application on an emulator or connected mobile device.

To test this if the application is working, log in on two different accounts and try creating lists, editing items, collaborating, etc.

5.8 Development Summary

At the current stage of development, the Greenlist project has focused primarily on establishing the foundation of the application. The user interface layout and screen structure have been designed using Figma and v0, and the React Native project structure has been set up for frontend development. Authentication and Firebase integration have now been implemented and integrated into the application.

The development process has prioritized building the core components required for the application, including the login and signup interfaces, grocery list viewing screens, and the structure for managing grocery list items. These components are now connected to Firebase services to support user authentication and data storage.

Some features are still in progress, including full implementation of grocery list collaboration and final database structure. Additional testing will be required as authentication and database connections continue to be refined.

At this stage, the project has established the core structure of the system and is continuing development toward completing the remaining functionality.

6. Testing

6.1 Purpose

The purpose of this document is to define the test scope, focus areas, and objectives for the GreenList grocery list mobile application. Specifically, this document establishes:

- The test scope and focus areas relevant to GreenList features.
- The test responsibilities across the development team.
- The test strategy for all levels and types of testing.
- Entry and exit criteria for each testing phase.
- The test schedule and major milestones.
- Assumptions, risks, and dependencies that affect testing.

6.2 Scope

This document details the testing to be performed by the SP-5 Green project team for the GreenList application — a React Native mobile app backed by Firebase Authentication and Cloud Firestore, allowing users to create, manage, and collaboratively share grocery lists in real time. The document defines:

- What will be tested (authentication, list CRUD, real-time sync, collaboration, UI).
- How testing will be performed (manual, unit, integration, and connectivity testing).
- What resources are needed and when.

6.3 Testing Summary

6.3.1 In Scope

The following areas are within scope for this project's testing efforts:

- User authentication: sign-up, login, logout, delete account, and Google Authentication via Firebase authentication.
- Grocery list management: create, read, update, and delete (CRUD) lists.
- Grocery item management: add, edit, delete, and mark items complete within a list.
- Real-time synchronization: verifying changes propagate instantly to all collaborators.
- Collaboration features: inviting friends, managing permissions, removing collaborators.
- Favorites and template features.
- USDA food database integration for food search and nutritional information.
- UI responsiveness and navigation across major application screens.
- Firebase security rules enforcement (access control).

6.3.2 Out of Scope

- Android platform compatibility (team only testing on iOS).
- Performance/load testing under high concurrent user counts.
- Dark mode or Desktop web version (planned improvements, not yet implemented).

6.4 Test Cases

ID	Description	Expected Result
TC-01	Sign up with a new email and password	Account Created, user lands on lists page
TC-02	Log in with valid credentials	User authenticated, lists page displayed
TC-03	Attempt Login with invalid credentials	User shown error logging in, no authentication gained
TC-04	Log in with Google	Google account linked, user authenticated
TC-05	Log out of the app	Session cleared, user returned to login screen
TC-06	Delete account	Account deleted, not able to log into it anymore
TC-07	Create a new grocery list	List appears on lists page and saved to firestore
TC-08	Delete a grocery list	List removed from UI and database
TC-09	Add an item to a grocery list	Item appears in list and database in real-time
TC-10	Edit an existing grocery item	Updated name/details reflected in the list
TC-11	Delete a grocery item	Item removed from list and database in real-time
TC-12	Mark an item as complete	Item shows as checked; completion count updates
TC-13	Two users edit the same shared list simultaneously	Both users see each other's changes with minimal delay
TC-14	Invite a friend as a collaborator	Friend can see and edit the shared list
TC-15	Remove a collaborator from a list	Removed user can no longer access the list
TC-16	Attempt to access a list without permission	Access denied; list does not appear for unauthorized user
TC-17	Mark a list as a favorite	List appears under the favorites tab
TC-18	Navigation between all major screens	No crashes or broken routes

6.5 Test Procedures

TC-01: Sign Up

- Open the app on the iOS simulator or device.
- Tap Sign Up and enter a new email address and password.
- Expected: Account is created and the Lists page loads.

TC-02: Login

- Open the app and tap Log In.
- Enter valid credentials and submit.
- Expected: User is authenticated and the Lists page is shown.

TC-03: Invalid Credentials

- Open the app and tap Log In.
- Enter invalid credentials and submit.
- Expected: User is shown an error message and user is not authenticated.

TC-04: Google Authentication

- Tap Continue with Google on the login screen.
- Select a Google account and authorize.
- Expected: User is logged in and redirected to the Lists page.

TC-05: Logout

- Navigate to the Account screen.
- Tap Log Out.
- Expected: Session ends and the login screen is displayed.

TC-06: Delete Account

- Go to the Account screen.
- Click delete account.
- Expected: The account should be deleted and logged out to the login page.

TC-07: Create List

- From the Lists page, tap the plus button.
- Enter a list name and confirm.
- Expected: New list appears in the list view.

TC-08: Delete List

- Go into the list, tap the three dots in the top right.
- Select Delete and confirm.
- Expected: List is removed from the screen and Firestore.

TC-09: Add Item

- Open a grocery list.
- Tap Add Item, enter a name, and confirm.
- Expected: Item appears at the bottom of the list.

TC-10: Edit Item

- Tap an existing item to open edit mode.
- Change the name and save.
- Expected: Updated name is shown in the list.

TC-11: Delete Item

- Open a grocery list.
- Tap the trash icon on the right side next to the item.
- Expected: Item is removed from the list.

TC-12: Complete Item

- Tap the checkbox next to a grocery item.
- Expected: Item is marked complete; count updates on the Lists page.

TC-13: Real-Time Sync

- Open the same list on two separate devices/accounts.
- Add or edit an item on one device.
- Expected: The change appears on the other device within ~2 seconds without refreshing.

TC-14: Invite Collaborator

- Open a list and navigate to the share/collaborator settings.
- Search for a friend's account and send an invite.
- Expected: Friend can now see and edit the list on their device.

TC-15: Remove Collaborator

- In the list's collaborator settings, select a collaborator and remove them.
- Expected: Removed user can no longer see or access the list.

TC-16: Access Control

- Log in with a test account that has not been invited to a specific list.
- Attempt to navigate to or search for that list.
- Expected: List is not visible; Firestore security rules block access.

TC-17: Favorites

- Tap the heart icon next to the grocery list.
- Navigate to the Favorites tab.
- Expected: The list appears under Favorites.

TC-18: Navigation

- Tap through all bottom navigation tabs: Home, Lists, Nutrition, Friends, Account.
- Open and close several lists.
- Expected: All screens load without crashes or blank pages.

6.6 Test Environment

Component	Details
Frontend Framework	React Native with Expo (TypeScript/TSX)
Backend / Database	Firebase Authentication + Cloud Firestore

Component	Details
Development Machine	macOS with Xcode installed
Mobile Runtime	iOS Simulator (Xcode) and physical iPhone via Expo Go
Package Manager	Node.js / npm
Version Control	GitHub
Network	Standard Wi-Fi

6.7 Test Data

Testing required three accounts: An owner account used for all list creation and CRUD tests, a collaborator account added to shared lists for real-time sync and permission tests, and an unrelated account intentionally kept off all lists to verify access control. A separate Google account was also used for Google Authentication login testing. The following data was set-up to start testing:

- A "Weekly Groceries" list owned by the owner account, pre-filled with items (Milk, Eggs, Bread, Butter, Apples) used for item add, edit, delete, and complete tests.
- A "Shared Shopping" list owned by the owner account and shared with the collaborator account, used for real-time sync and collaborator management tests.
- A "Delete Me" list with placeholder items, used solely for the list deletion test so that other lists remain intact across test runs.

6.8 Software Test Report (STR)

The table below records the result of each test case executed against the GreenList prototype.

ID	Description	Pass	Fail	Severity
TC-01	Sign up with a new email and password	✓		
TC-02	Log in with valid credentials	✓		
TC-03	Attempt Login with invalid credentials	✓		
TC-04	Log in with Google	✓		
TC-05	Log out of the app	✓		
TC-06	Delete account	✓		
TC-07	Create a new grocery list	✓		
TC-08	Delete a grocery list	✓		
TC-09	Add an item to a grocery list	✓		
TC-10	Edit an existing grocery item	✓		
TC-11	Delete a grocery item	✓		

ID	Description	Pass	Fail	Severity
TC-12	Mark an item as complete	✓		
TC-13	Two users edit the same shared list simultaneously	✓		
TC-14	Invite a friend as a collaborator	✓		
TC-15	Remove a collaborator from a list	✓		
TC-16	Attempt to access a list without permission	✓		
TC-17	Mark a list as a favorite	✓		
TC-18	Navigation between all major screens	✓		

7. Version Control

7.1 Repository Structure

Repository: https://github.com/GreenCTK/greenlist-app

Branch Strategy

- master — Production-ready code
- Nutritional-Data — Feature branch for the nutritional database integration
- Chris-Dev — Active development branch for feature iteration and polish

Directory Organization

```
greenlist-app/
|-- App.tsx                    # Root component with context providers
|-- index.js                   # Expo entry point
|-- app.json                   # Expo configuration
|-- package.json               # Node dependencies & scripts
|-- tsconfig.json              # TypeScript configuration
|-- firebase.json              # Firebase emulator config
|-- firestore.rules            # Firestore security rules
|-- firestore.indexes.json     # Firestore index definitions
|-- README.md                  # Project documentation
|-- .gitignore                 # Git ignore rules
|-- scripts/
|   `-- seedNutrition.js       # Seeds the nutritional database
`-- src/
    |-- config/
    |   `-- firebase.ts        # Firebase SDK initialization
    |-- context/               # Global state providers
    |   |-- AuthContext.tsx    # Auth state, login, signup
    |   |-- ListsContext.tsx   # Lists & items CRUD + real-time sync
    |   |-- NutritionContext.tsx # Nutritional database
    |   |-- PreferencesContext.tsx # User preferences & haptics
    |   `-- ThemeContext.tsx   # Light/dark mode
    |-- navigation/
    |   `-- AppNavigator.tsx   # Auth stack + bottom tabs
    |-- components/            # Reusable UI components
    |   |-- CreateListModal.tsx
    |   `-- NutritionLabel.tsx
    |-- screens/
    |   |-- auth/              # Welcome, Login, Signup, Google
    |   `-- app/               # Home, Lists, InsideList, Nutrition, etc.
    |-- theme/
    |   `-- colors.ts          # Centralized color palette
    |-- types/
    |   `-- nutrition.ts       # TypeScript type definitions
    `-- data/
        `-- nutritionSeedData.ts # Seed data for 50+ grocery items
```

7.2 Commit History

The project includes a comprehensive commit history tracking the major development milestones:

- Initial project setup and Firebase integration (Auth + Firestore + Security Rules)
- Core list management (create, delete, add/remove items, mark complete)
- Real-time synchronization across devices using Firestore onSnapshot listeners
- Authentication flow with email/password and Google OAuth
- Friends system and collaborative list sharing
- UI navigation with bottom tabs and nested stack navigators
- Nutritional database feature with seed data, search, and per-list summaries (PR #1)
- Explore page restoration after navigation refactor
- Dark mode support with centralized theme context
- Edit Item feature with nutrition database search integration (PR #2)
- Firestore security rule updates for the nutritionalData collection
- Bug fixes, UI polish, and code cleanup

Commit Log

From `git log --all --oneline`:

```
f9c5372 Merge pull request #2 from GreenCTK/Chris-Dev
806d443 Add edit item feature with nutrition database search and pencil icon trigger
7d97c99 Add dark mode support
b064b0d Fixed Explore Page
2d0d70c Merge pull request #1 from GreenCTK/Nutritional-Data
45905f1 Add nutritional data feature with Firestore integration
d7f5ddc Initial commit with Firebase integration
```

8. Summary

8.1 Summary

GreenList is a fully functional iOS grocery list mobile application built with React Native and Firebase. The application delivers on all original project goals: user authentication, full CRUD grocery list management., real-time collaboration and synchronization via Firestore, permission management, USDA food database integration, and a clean navigable User Interface.

8.2 Lessons Learned

- Pivoting from Flutter to React Native in the project, while it disrupted the flow of the project, it led to faster development because it was easier to translate the components Figma gave.
- Using Figma and v0 prototyping before writing any code helped reduce the number of UI revisions and provided a good idea of what the UI would look like.
- Firebase was a very useful tool during development and potentially could be great in other projects or in the industry.

Appendix

Project Setup

- Install Node.js and a package manager (npm or yarn).
- Install React Native Expo.
- Clone the repository from GitHub.
- Set up a Firebase project through the Firebase console.
- Enable Firebase Authentication and Firebase Database.
- Add the Firebase configuration files to the React Native project.
- Run the application on an emulator or connected mobile device.

References and Resources

- React Native Documentation: https://reactnative.dev/docs/getting-started
- Firebase Documentation: https://rnfirebase.io/
- React Native + Firebase Tutorial: https://www.youtube.com/watch?v=6DN8mzxineI&list=PLKWMD009Q4qTGuqXxRq51f8OoDaIoJ1yo&index=8

-