D.O.M.E

GO-TO GROCERY APP

TEST PLAN

Date: OCT 17, 2025.

Table of Contents

Introduction

The Test Plan outlines how the Go-to Grocery App will be tested throughout each sprint. It outlines our objectives, scope, risks, and testing methodology. The key goal is to ensure that every aspect of the app works as intended and is simple for people to use. The document informs all team members on what has to be tested and what is not part of the current sprint.

1. Objectives

The Go-to Grocery App allows users to compare grocery prices across retailers, create shopping lists, identify cost-effective solutions, and practice healthy diet habits. Since the development has not been started yet, this document only defines the testing approach for upcoming sprints. The primary purpose of testing is to ensure that these functionalities run smoothly and without errors.

In Sprint 1, we are focused on creating all necessary documentation such as requirements, design, and test planning so that future testing can run smoothly once the first prototype is ready.

2. Team Members

Resource Name	**Role *(examples are given below)***
Subol Dhital	Project manager/developer
Elsa Joy	Tester/ AI developer
Mikal Debsay	Front-end developer lead
Saubhagya Bhandari	Back-end developer
D'yanna Grey	
Olajumoke Kupoluyi	

2. Scope

Because the team is still in the process of documentation phase, no actual testing has been performed to this date. The scope for sprint 1 is to define which features will be tested later and to prepare basic test cases and data sets.

At the end of sprint 1, the team will have:

1. A clear list of functional requirements and user stories.
2. Draft designs for user interface and database structure.
3. A written test plan explaining how testing will be done later.

4. Identified tools and devices that will be used for testing.
5. Defined roles and responsibility for each member.

These steps will help ensure the team is ready for coding and testing in Sprint 2, where actual test cases and system functionality will be executed.

Assumptions / Risks

2.1. Assumptions

1. The app's initial prototype will be available during the next sprint.
2. When development begins, each team member will gain access to testing tools and devices.
3. Before genuine testing can begin, test data and mock APIs will be ready.
4. Team meetings will be held weekly to review progress.
5. Changes in requirements will be considered and documented formally.
6. R1 has mockups (prototypes/stubs) screens open for clicking around before conducting tests.
7. Mock data set (≈10 items with a UPC; several are on sale) is ready.
8. One device running an Android or emulator (for camera + notifications to work).
9. One network to be available during auth testing; offline tests can't happen until later.

2.2. Risks

The following risks have been identified and the appropriate action identified to mitigate their impact on the project. The impact (or severity) of the risk is based on how the project would be affected if the risk was triggered. The trigger is what milestone, or event would cause the risk to become an issue to be dealt with.

#	Risk	Impact	Trigger	Mitigation Plan
1	Late Submission of Documents – team delays completing documentation	High	Missing project deadlines	Set internal deadlines a few days before the official due date and review work early.
2	Miscommunication in Team Tasks – members not clear on roles	Medium	Overlapping or missing sections	Hold short team meetings each week to review task progress.

3	Incomplete Requirements – unclear features or user	High	Confusion when coding	Discuss unclear requirements with the instructor before
4	stories Technical Setup Delays – issues installing or configuring tool	Medium	starts Developme nt or testing cannot start on time	Development begins. Assign one person to handle setup early and verify everyone's environment.
5	Loss of Files or Version Conflicts	Low	Team members overwrite work accidentally	Use GitHub or Teams shared folder for version control and backups

3) Test Approach

- Method: Black-box testing against prototypes/stubs; limited performance/accessibility spot checks subsequently.
- Order of coverage:

1. Registration/Login → Onboarding (allergies/budget)
2. Search + Filters (mock data)
3. Shopping List (add/remove-check)
4. Scan after Purchase → create/update PantryItem
5. Reminder creation (delivery checks next sprint)

- Exit (Sprint 1): Above flows run against mock/stubs, major issues documented in Trello/Test Plan Tracker.

(Optional test names in your tracker: TC-Auth-01 Register, TC-Auth-02 Login, TC-Search-01 Search, TC-Filter-01 Filters, TC-List-01 Add to List, TC-Scan-01 Scan at Home, TC-Rem-01 Create Reminder).

4) Test Setting

- Devices/OS: Android 12+ (1 recommended physical device) or Android emulator
- Data: Mocked provider; ~10 sample products with UPCs.
- Permissions: Camera (to scan); Notifications (to be used later); Location note: optional.

- Builds/Tools: prototype/stub builds; Trello for issues; TestPlanTracker.xlsx for results; printable barcode sheet.

5) Test Schedule & Roles (brief)

Schedule (Sprint 1)

Week 1: Install mock data, printable UPCs; craft rough test titles; set up TestPlanTracker.
Week 2: Execute tests on Auth/Onboarding, Search + Filters (mock).
Week 3: Test Lists, Scan-after-purchase, Reminder creation (entry only); log defects; results summary.

Roles

Test coordination: Mikal (assign cases, review defects).
Functional testing: Elsa (Auth/Onboarding), D'yanna (Search/Filters), Saubhagya (Lists), Olajumoke (Scan/Reminders), Subol (cross-checks).
Data/fixtures: Elsa + Saubhagya (mock products, UPCs).
Reporting: Mikal (end-of-sprint test summary in the tracker).