Testing Plan

Unit Converter Application

Maria Harrson
CS 6387: Topics in Software Engineering
Vanderbilt University
Nashville, TN
maria.harrison@vanderbilt.edu

Table of Contents

1. Introduction

a. Purpose

This document outlines a test plan for a Unit Converter application's main components: the Landing Page, Base Measurement Selection Toggle, the Inputs Page, and the Results Page. The primary objective is to ensure the functionality, usability, and reliability of these components.

b. Project Overview

The Unit Converter is an Android mobile application designed to offer a solution to users who need to convert liquid and dry units of measurement. By addressing the need for accurate and efficient measurement conversions while in the kitchen, this app may serve both casual users and professionals who require precise data for cooking, baking, and various scientific applications. The app supports conversions between metric and imperial units, ensuring flexibility for users across different regions and industries.

The unit converter system accepts numeric data representing the current measurement, to a single predefined unit of measurement. Additionally the system accepts encoded data from user selection input describing the unit of measurement for conversion. All input data provided by the user must align with the list of predefined units of measurements provided by the stakeholder. Immediately after the user inputs the data, it is processed through a conversion algorithm and converts the value for all units from the list of predefined units of measurements. Only one value is returned to the user, based on their initially requested unit of measurement. The CupChameleon system supports both metric and imperial unit systems.

2. Scope

a. In-scope

Landing Page: Verify the users can select and update their preferred measurement system.

Inputs Page: Ensure users can enter base measurement unit and value, select target unit, and initiate conversion algorithm

Results Page: Confirm users can convert measurements and view converted results.

b. Out of Scope

User tutorial/help guide: Added in future release.

User Analytics: Added in future release.

3. Testing Strategy

a. Test Objectives

Functional Testing: Verify all in-scope features work as expected under various conditions.

Performance Testing: Test the application's performance under various conditions (e.g., different device types, varying internet speeds).

Usability Testing: Assess the application's user interface and user experience, ensuring users can navigate between pages including: the landing page, measurement system selection, input page with input fields, and results page.

Security Testing: Verify the secure storage of user base system preferences and data.

Compatibility Testing: Ensure the application works correctly on different Android mobile devices, versions and device screen sizes.

b. Data Approach

Valid Data: Create sets of mock user data like user profiles, consisting of numerous different dry and liquid measurements in a range of units and target output units, to reflect typical use cases.

Invalid Data: Create data to test edge cases and error handling, including incorrect measurement values, string literals instead of numerical values, and SQL injection code.

c. Level of Testing

Manual Testing: Perform tests manually to evaluate user interface and design components.

Automated Testing: Use tools and scripts to perform repetitive and complex test cases.

d. Usability Testing

User Interviews and Surveys: Perform multiple rounds of user surveys to collect feedback from a sample of users to understand their experience with the platform.

e. Unit Testing

Landing Page: Ensure the Landing Page displays correctly and transitions smoothly to the Inputs Page.

Measurement System Selection Toggle: Ensure the Measurement System Selection Toggle functions correctly and saves the user's preference.

Inputs Page: Ensure the Inputs Page accepts user data and validates input, enables the conversion button, and transitions correctly to the Results Page.

Results Page: Ensure the Results Page displays the converted value and shows a loading indicator while the algorithm is running.

f. Functional Testing

Manual Testing: Testers will manually execute test cases for main functionalities like base measurement selection, entering base measurement value, selecting base unit and target unit, and converting values.

Automation Testing: Automation scripts will be created using Espresso UI testing, Mockito Unit testing to handle repetitive tasks like base measurement system selection, entering base value and unit data, selecting target unit, starting conversion process, and performing conversion operations.

g. Security Testing

Vulnerability Scanning: Use SonarQube tools to scan for common vulnerabilities like SQL injection, cross-site scripting (XSS), and insecure configurations.

Penetration Testing: Perform manual tests to identify potential security threats that automated tools might miss.

h. User Acceptance Testing

Google BetaStore: Application will be released to BetaStore 10 days prior to final production release date to ensure behavior is as expected before releasing to higher production environments. BetaStore users will have the ability to report defects during the 10 days prior to final production.

Google PlayStore: 10 days after the app is released to BetaStore, the app will automatically be published to Google PlayStore through PlayStore's release process.

4. Test Cases

a. Landing Page

TC-1: Verify Landing Page Display

- Preconditions: None
- Steps:
- Launch the application.
- Expected Result: The Landing Page is displayed with the application title, measurement system selection toggle, and a button to proceed to the Inputs Page.

TC-2: Verify Title Display

- Preconditions: None
- Steps:
- Launch the application.
- Expected Result: The application title is displayed prominently on the Landing Page.

b. Measurement System Selection

TC-3: Verify Measurement System Selection

- Preconditions: None
- Steps:
    1. Launch the application.
    2. Select "Metric" or "Imperial" from the toggle.
- Expected Result: The selection is highlighted and saved.

TC-4: Verify Measurement System Persistence

- Preconditions: A measurement system selection has been made previously.
- Steps:
    1. Launch the application.
- Expected Result: The previously selected measurement system is automatically set.

TC-5: Verify Changing Measurement System

- Preconditions: A measurement system selection has been made previously.
- Steps:
    1. Launch the application.
    2. Change the selection to the other measurement system.
- Expected Result: The new selection is saved and applied.

c. Inputs Page

TC-6: Verify Inputs Page Display

- Preconditions: User has navigated from the Landing Page.
- Steps:
    1. Navigate to the Inputs Page.
- Expected Result: The Inputs Page is displayed with input value text field, input unit dropdown, target unit dropdown, and conversion button.

TC-7: Verify Input Value Text Field

- Preconditions: None
- Steps:
    1. Enter a decimal value up to the hundredths in the input value text field.
- Expected Result: The value is accepted and displayed correctly.

TC-8: Verify Input Unit Dropdown Selection

- Preconditions: None
- Steps:
    1. Select an input unit from the dropdown.
- Expected Result: The selection is accepted and displayed correctly.

TC-9: Verify Target Unit Dropdown Selection

- Preconditions: None
- Steps:
    1. Select a target unit from the dropdown.
    2. Expected Result: The selection is accepted and displayed correctly.

TC-10: Verify Conversion Button Disabled State

- Preconditions: At least one input field (value, input unit, or target unit) is null or blank.
- Steps:
    1. Ensure one of the input fields is not filled.
- Expected Result: The conversion button remains in a disabled state.

TC-11: Verify Conversion Button Enabled State

- Preconditions: All input fields are filled out.
- Steps:
    1. Fill out all input fields (value, input unit, and target unit).
- Expected Result: The conversion button becomes enabled.

TC-12: Verify Conversion Button Functionality

- Preconditions: All input fields are filled out, and the conversion button is enabled.
- Steps:
    1. Click the conversion button.
- Expected Result: The input data is stored, the conversion algorithm is triggered, and the user is transitioned to the Results Page.

d. Results Page

TC-13: Verify Results Page Display

- Preconditions: User has navigated from the Inputs Page after clicking the conversion button.
- Steps:
    1. Click the conversion button on the Inputs Page.
- Expected Result: The Results Page is displayed.

TC-14: Verify Loading Indicator

- Preconditions: Conversion algorithm is running.
- Steps:
    1. Click the conversion button on the Inputs Page.
- Expected Result: A loading indicator is displayed while the conversion algorithm runs.

TC-15: Verify Converted Value Display

- Preconditions: Conversion algorithm has completed.
- Steps:
    - Wait for the conversion algorithm to complete.
- Expected Result: The converted value is displayed on the Results Page.

5. Execution Strategy

a. Entry Criteria

- Code has merged in successfully
- Manual testing/regression test scripts are completed and reviewed
- App's release build runs CI/CD pipeline
- Automated Unit and UI test scripts are completed and reviewed

b. Exit Criteria

- 1% of manual test scripts are completed
- 1% of automated test scripts are completed
- 95% pass rate of test scripts
- Non-critical defects found are documented as a change request in future release
- Critical defects found are immediately addressed and regression testing occurs
- All expected and actual results of test scripts are captured and documented with
- All defects are logged and tracked in a spreadsheet

c. Validation and Defect Management

Defects found during testing must be categorized as follows

Severity	**Impact**
4- Critical	Functionality is blocked. Application or feature is unusable in current state. **Testing cannot continue.**
3 - High	Functionality is not usable and no workaround exists. **Testing can continue.**
2 - Medium	Functionality is usable with issues. Workaround exists to achieve expected behavior. **Testing can continue.**
1- Low	Functionality is usable with unclear error or cosmetic error. Minimal impact on product usage. **Testing can continue**