SE 3XA3: Test Plan
CamRuler

Team 10,
Meet Patel: patelm16
Prince Kowser: kowserm
Kshitij Mehta: mehtak1

December 6, 2017

Contents

List of Tables

List of Figures

1 Revision History

Table 1: **Revision History**

Date	**Version**	**Notes**
October 27, 2017	Kshitij Mehta, Meet Patel, Prince Kowser	Rev 0
December 6, 2017	Kshitij Mehta, Meet Patel, Prince Kowser	Rev 1

This document is the Test Plan for the application CamRuler.

2 General Information

2.1 Purpose

The purpose of CamRuler is to simplify the process of measuring an object by allowing the user to use their phones camera to measure any object. Having this app on mobile devices will enable users to conveniently measure objects without the struggle of nding a ruler and doing tedious measurements by hand.
The purpose for testing this project is to ensure that the application works properly and that the functionality of this project can be achieved. Having adequate test cases for this project will increase the quality of user experience and lead to a satisfactory use of the application.

2.2 Scope

This document provides a meaningful method to ensure that CamRuler meets all the technical, functional and non-functional requirements. This test plan will describe the strategy for testing each of the requirements as well as describe the testing framework that will be used for all testing related to this application.

2.3 Acronyms, Abbreviations, and Symbols

Table 2: **Table of Abbreviations**

Abbreviation	**Definition**
GUI	Graphical User Interface. An interface that humans can use to interact with computers
OS	Operating System
SRS	Software Requirements Specifications
API	Application Programming Interface. A set of functions and the purpose of those functions that a client can call in their software
PoC	Proof of Concept
cm	centimeters
m	meters
mm	millimeters
app	Application

Table 3: **Table of Definitions**

Term	**Definition**
Compiler	A software utility that converts code written in any programming language into machine language that can be understood by the computer
CamRuler	The name of the application
Product	The mobile application that is being developed
Project	The overall development of the product
Java	A programming language

2.4 Overview of Document

This document outlines the testing plan for CamRuler and is structured as follows:

- Section 2: Describes the testing schedule, testing approach and testing tools to be used in this project.

- Section 3: Detailed descriptions of each test plan for functional and non-functional requirements.
- Section 4: Defines the plan to test the PoC.
- Section 5: Provides any tests that will compare CamRuler to the original implementation of the product.
- Section 6: Describes all unit testing plans.

The requirements listed in this document is derived from CamRuler's SRS document.

3 Plan

3.1 Software Description

The software being tested is CamRuler and the product will be implemented in Java. Here is a detailed description of the product:

Table 4: Software Description

Component of the Product	Description
Picture Taking function	This function will allow the user to take a picture of the object and the application prepares the picture to be analyzed for length measurements.
User Drawing function	This function allows the user to draw two points covering the length of the objects (both reference and the object to-be-measured) and the application draws a line between the two points.
Calculation function	This function allows the user to input the reference object's length in cm, m, or mm, and the application calculates the actual object's length using a ratios algorithm.
Display function	This function will allow the application to display the measurements to the user.

3.2 Test Team

The testing team comprises of the following developers:

- Prince Kowser
- Meet Patel
- Kshitij Mehta

This group of individuals will be responsible for writing and executing all tests.

3.3 Automated Testing Approach

This project will undergo automated testing through JUnit. Methods which can have an input value such as the mathematical ratios used to calculate measurements of objects can be tested through an automated means. Stubs of edge cases, normal cases, and negative cases will be created in order to be run and ensure that the program methods run properly. Stubs will also be created to test error handling in case the user enters something wrong. We also plan on having an efficient time taken to do the calculations so we will also run time tests to make sure that the calculations are being conducted within a reasonable amount of time. Because we do not have all our implementation done, we have just planned these aforementioned tests. Upon completion of the implementation, we will think of and conduct more tests that could be tested automatically. Automated tests tend to have an input and output value which could be compared to other values. Tests such as checking if a picture is set as the background of the application would have to be done manually.

3.4 Testing Tools

This application will use JUnit as it's main unit testing framework. Stubs will be generated for the calculation module and results will be compared to our coverage metrics. Our metrics include that we receive a 95%+ accuracy rate to the original measurements of objects through manual testing and 100% for calculation testing as they are just simple ratios.If metrics not met, then we will have to figure out a better implementation.With our project being an Android application, a lot of the testing would be done manually and

through user feedback. Due to this reason, we would only be using JUnit to test the calculations taking place in our application, whereas the other tests would be done manually.

3.5 Testing Schedule

See Gantt Chart at the following location.

4 System Test Description

4.1 Tests for Functional Requirements

4.1.1 Picture taking

1. FR-PT-1

   Type: Functional, Dynamic, Manual

   Initial State: Android application with a "take photo" button for the user to press.

   Input: Press button

   Output: Open phones camera application

   How test will be performed: The "take photo" button on the app will be pressed once and the tester will check if the app opens up the user's phone camera application for the user to take a picture or if the app crashes.

2. FR-PT-2

   Type: Functional, Dynamic, Manual

   Initial State: Phone camera

   Input: The user takes a picture

   Output: Sets the picture to the specified image area on application

   How test will be performed: The tester will take a picture with the phone's camera application and check if the picture is set properly and clearly on the app screen background or if the app crashes.

3. FR-PT-3

   Type: Structural, Dynamic, Manual

   Initial State: The application is launched but the camera is not working.

   Input: The user selects the option to select a picture from the gallery.

   Output: The application successfully opens the gallery application on the user's phone, allowing the user to select a picture and uses that picture as the application's background.

   How will test be performed: The tester will check that an option to select from gallery appears and by clicking the button, the gallery application opens. The tester will check that a picture can be selected and that after selecting the picture, the gallery application closes and goes back to CamRuler automatically.

4.1.2 User Drawing

4. FR-UD-3

   Type: Functional, Dynamic, Manual

   Initial State: A picture is taken and set on to the specified image area on application.

   Input: Draw two dots.

   Output: A line is drawn between the two dots.

   How test will be performed: Draw two dots on the picture and check if a line is drawn between the two dots only and if a dot is able to be drawn

5. FR-UD-4

   Type: Functional, Dynamic, Manual

   Initial State: A line is drawn between the two dots on the picture.

   Input: Draw another pair of dots.

Output: A line drawn between the new pair of dots only.

How test will be performed: Draw two new dots and check if a line is drawn between the new two dots and no more dots is able to be drawn.

6. FR-UD-5

   Type: Functional, Dynamic, Manual

   Initial State: All lines and dots are drawn.

   Input: Press the redraw button.

   Output: All lines and dots are cleared and prompts the user to redraw.

   How test will be performed: Press the redraw button and check if all lines are cleared and user is able to redraw.

4.1.3 Calculation

7. FR-C-5

   Type: Functional, Dynamic, Manual

   Initial State: Waiting for user to confirm their drawn dots.

   Input: Press confirm button.

   Output: A window pops up with measurement information.

   How test will be performed: The confirm button is pressed and the tester will check if a window pops up asking the user for reference length with unit option of cm, mm and m, and what units they would like their actual length to be in with option of cm, mm and m.

8. FR-C-6

   Type: Functional, Dynamic, Manual

   Initial State: Pop up window requiring measurement information.

   Input: Enter reference object in cm and select output unit as cm

   Output: A new window pops up with the length of the object

How test will be performed: Check for a scroll down menu that allows user to select from cm, m, or mm. Then select cm and input the reference object's length while selecting cm as output length. Then check if output length is in cm.

9. FR-C-7

   Type: Functional, Dynamic, Manual

   Initial State: Pop up window requiring measurement information.

   Input: Enter reference object in cm and select output unit as mm.

   Output: A new window pops up with the length of the object.

   How test will be performed: Check for a scroll down menu that allows user to select from cm, m, or mm. Then select cm and input the reference object's length while selecting mm as output length. Then check if output length is in mm.

10. FR-C-8

    Type: Functional, Dynamic, Manual

    Initial State: Pop up window requiring measurement information.

    Input: Enter reference object in cm and select output unit as m.

    Output: A new window pops up with the length of the object.

    How test will be performed: Check for a scroll down menu that allows user to select from cm, m, or mm. Then select cm and input the reference object's length while selecting m as output length. Then check if output length is in m.

11. FR-C-9

    Type: Functional, Dynamic, Manual

    Initial State: Pop up window requiring measurement information.

    Input: Enter reference object in mm and select output unit as cm.

    Output: A new window pops up with the length of the object.

How test will be performed: Check for a scroll down menu that allows user to select from cm, m, or mm. Then select mm and input the reference object's length while selecting cm as output length. Then check if output length is in cm.

12. FR-C-10

    Type: Functional, Dynamic, Manual

    Initial State: Pop up window requiring measurement information

    Input: Enter reference object in mm and select output unit as mm.

    Output: A new window pops up with the length of the object.

    How test will be performed: Check for a scroll down menu that allows user to select from cm, m, or mm. Then select mm and input the reference object's length while selecting mm as output length. Then check if output length is in mm.

13. FR-C-11

    Type: Functional, Dynamic, Manual

    Initial State: Pop up window requiring measurement information.

    Input: Enter reference object in mm and select output unit as mm.

    Output: A new window pops up with the length of the object.

    How test will be performed: Check for a scroll down menu that allows user to select from cm, m, or mm. Then select mm and input the reference object's length while selecting m as output length. Then check if output length is in m.

14. FR-C-12

    Type: Functional, Dynamic, Manual

    Initial State: Pop up window requiring measurement information.

    Input: Enter reference object in m and select output unit as cm.

    Output: A new window pops up with the length of the object.

How test will be performed: Check for a scroll down menu that allows user to select from cm, m, or mm. Then select mm and input the reference object's length while selecting cm as output length. Then check if output length is in cm.

15. FR-C-13

    Type: Functional, Dynamic, Manual

    Initial State: Pop up window requiring measurement information.

    Input: Enter reference object in m and select output unit as mm.

    Output: A new window pops up with the length of the object.

    How test will be performed: Check for a scroll down menu that allows user to select from cm, m, or mm. Then select mm and input the reference object's length while selecting mm as output length. Then check if output length is in mm.

16. FR-C-14

    Type: Functional, Dynamic, Manual

    Initial State: Pop up window requiring measurement information.

    Input: Enter reference object in m and select output unit as m.

    Output: A new window pops up with the length of the object.

    How test will be performed: Check for a scroll down menu that allows user to select from cm, m, or mm. Then select mm and input the reference object's length while selecting m as output length. Then check if output length is in m.

4.2 Tests for Nonfunctional Requirements

4.2.1 Look and Feel

1. SS-1

   Type: Structural, Static, Manual

   Initial State: The application is installed on the phone.

Input: Several users are asked to launch the application and give feedback about the application's layout and theme.

Output: All users provide their feedback

How test will be performed: A test group of people from various ages and occupation will be asked to install the application on their device and give feedback regarding the layout, the theme and how they like the "look and feel" of the application. This feedback will be used to further enhance the application.

4.2.2 Usability

2. SS-2

   Type: Structural, Static, Manual

   Initial State: The application is not running.

   Input: The user launches the application.

   Output: The application provides an option for the user to see instructions on how to use the application and an option to take a picture.

   How test will be performed: The tester will check to see if the program displays a set of instructions upon launching the application. The effectiveness of the those instructions will be tested by having a small test group of people from various ages and occupations launch the application and read the instructions without any assistance. The feedback received from this group of people will be used to enhance tests.

3. SS-3

   Type: Structural, Dynamic, Manual

   Initial State: The dialog box to enter the reference object's measurements is open with the measurements filled in.

   Input: Several users are asked to click on "units" to specify a unit of measurement.

   Output: The users all understand the metrics being used and select the appropriate unit.

How test will be performed: A test group of people from various ages and occupation will be given a set of metrics and judge how familiar they are with those metrics. Upon entering the required measurements, the tester will ensure that none of the calculation process is displayed to the user and only the final result is displayed.

4. SS-4

   Type: Structural, Static, Manual

   Initial State: The application is installed on the phone.

   Input: Several users with different Android phones are asked to launch the application.

   Output: The application is launched successfully on all Android phones.

   How test will be performed: A test group of people from various ages and occupation who all have various models of Android phones (e.g. Samsung, HTC, LG, etc) will be asked to install and use the application on their phone. This test will ensure that the application works regardless of the screen size, camera quality or the phone's system properties (i.e. memory, speed, Android version, etc.).

4.2.3 Performance

5. SS-5

   Type: Structural, Dynamic, Automatic

   Initial State: The dialog box for the measurements of the reference object is open and filled out.

   Input: The user taps the "Done" button.

   Output: The application calculates the actual object's measurements within 3 seconds after the user taps "Done".

   How test will be performed: The program will run a timer and check to see if the output is displayed within a maximum of 2 seconds.

6. SS-6

Type: Structural, Dynamic, Automatic

Initial State: The dialog box for the measurements of the reference object is open.

Input: User is asked to input varying measurements for the reference object(smaller - larger) and using various units.

Output: The application calculates the actual object's measurements within 3 seconds after the user taps "Done"

How test will be performed: The program will run a timer and check to see if the output is displayed within a maximum of 2 seconds.

7. SS-7

   Type: Structural, Dynamic, Automatic

   Initial State: The dialog box for the measurements of the reference object is open and filled out.

   Input: The user taps the "Done" button.

   Output: The application calculates the actual object's measurements and has a precision of 3 decimal places.

   How test will be performed: The program will check that the number of decimal places of the calculated length is not more than 3.

8. SS-8

   Type: Structural, Dynamic, Manual

   Initial State: The picture is the background of the screen and the reference object, as well as the actual object are both selected.

   Input: The user taps the screen to draw another dot.

   Output: The application display's a message "no more objects can be measured".

   How will test be performed: The tester will check that after selecting two objects, attempting to draw another dot by tapping on the screen will result in the application displaying the appropriate message..

4.2.4 Stress Testing

9. SS-9

   Type: Structural, Dynamic, Manual

   Initial State: The picture is the background of the application.

   Input: The user is asked to retake the picture several times.

   Output: The system does not crash and it allows to user the take as many pictures as they want.

   How will test be performed: The tester will check to see how to system responds when re-taking a picture several times. In all conditions, the system should not crash. If the system does crash, the application should be able to re-open without any problems.

4.2.5 Stress Testing

10. SS-10

    Type: Structural, Dynamic, Manual

    Initial State: The picture is the background of the application.

    Input: The user is asked to retake the picture several times.

    Output: The system does not crash and it allows to user the take as many pictures as they want.

    How will test be performed: This will be a pass or fail test.

4.3 Traceability Between Test Cases and Requirements

Requirement	Test case
FR3	FR-PT-1, FR-PT-3
FR4	FR-PT-2, FR-UD-3
FR5	FR-UD-4
FR6	FR-UD-5
FR7	FR-C-5
FR8	FR-C-(6-14)
NF-AP	SS-1
NF-EU	SS-2
NF-UPR	SS-3
NF-AR	SS-4
NF-SLR	SS-5,SS-6
NF-PAR	SS-7
NF-CR	SS-8

5 Tests for Proof of Concept

The Proof of Concept testing will be used to verify the current implementation. This will be done through manual as the implementation currently finished is all interface based. This testing will also show that the project is feasible. The proof of concept testing will include testing the interface capture button as well as setting the image captured to be the background of the application.

5.1 Capture Picture Button Captures Image

1. PC-1

   Type: Functional, Dynamic, Manual

   Initial State: No input is set

   Input: Clicking the capture button on the application screen

   Output: User is enabled to take a picture of object to be measured

How test will be performed: This test will be conducted by simply clicking on the capture button on the user interface to make sure that our capture button correctly works. Clicking the button will allow user to take a picture.

5.2 Picture Captured As Application Background

1. PC-2

   Type: Functional, Dynamic, Manual

   Initial State: No input is set, but location of image taken is set

   Input: A picture which is taken by the user upon clicking capture button

   Output: Picture taken by user should appear as the background of the application from where user can select options from

   How test will be performed: This test will be conducted by comparing the output to the existing implementation where once the user takes a picture, it is set as the background image from which the features of the application can be used.

5.3 Application Does Not Crash on Open

1. PC-3

   Type: Functional, Dynamic, Manual

   Initial State: No input is set

   Input: App is opened on device by user

   Output: Screen with capture button appears

   How test will be performed: This test will be conducted by opening the application on the mobile device and ensuring that it does not crash and the functions to run the application are presented to the user.

6 Comparison to Existing Implementation

Since the existing implementation was successful in terms of its functionality, we could use it to compare our implementation to in order to see if we are going down the correct path. The following tests compare the program to the existing implementation called CameraRuler:

- PC-1
- PC-2
- FR-PT-1
- FR-PT-2
- FR-UD-3
- FR-C-6
- FR-C-9

7 Unit Testing Plan

As Java is being used to implement the project, the JUnit Framework will be used in order to perform unit testing.

7.1 Unit testing of internal functions

To test internal functions of the application, methods can be tested individually through an automated means. Methods which return values would be simplest to test using JUnit. Such methods which could be testing for this application would be the ratio calculations and metric conversions. This

would require an input value and an expected output value which would then be verified by JUnit if the formulas are correct. Because unit testing is quick using JUnit, every method with an input and output can be tested for normal values as well as edge case values. Inputs generating exceptions will also be tested to make sure of correct error handling. Stubs may have to be created to test all possible cases as we do not want program to crash on any kind of input. Coverage metrics will be used in order to maintain that the application works effectively and as we know that 100% coverage is usually not possible, we will aim for perfection while testing the internal functions as they are very important to our applications success.

7.2 Unit testing of output files

There are no output files to be tested with this program as the final implementation simply requires a picture taken by the user, and a selection of a few points and metrics.