<Company Name>

Contact Details Manager
<Iteration/ Master> Test Plan

Version <1.0>

Revision History

Date	Version	Description	Author
25/03/2018	1.0	This is a test plan for version 1.0	Tharsanan

Table of Contents

Test Plan

1. Evaluation Mission and Test Motivation

1.1 Background

This project involves implementing the contact details manager(CDM) which is an android mobile application based on the requirements and design documents which were submitted in the previous iteration. The CDM is developed by the implementation using Java.

A comprehensive test plan has been developed to ensure that the system conforms to the specifications, design and to perform quality assurance on the final product. This will enable to release as much as possible bug free CDM and minimize the risk of software failure. The test plan will allow verifying if the final product successfully meets the specifications which were specified in the previous documents (requirement and architecture document) with a variety of testing techniques. The plan will also help in fault detection with the test cases that have been designed.

1.2 Evaluation Mission

The main objectives of the testing plan are Ensuring that the specifications of the requirements document have been achieved, Ensuring that the system specifications of the design document have been achieved, Ensuring that the risk of software failure is reduced to a minimum and finding defects and correct them before go-live. Meeting these objectives will enable to release a stable version. There is no approach or method to guarantee a system completely free of defects.

2. Target Test Items

Category	Target
Unit testing	All the main functions, that process arithmetic operations and database fetch operations.
Integration testing	Add contact details Add profile info Change contact info View contact details View contact by filtering Add dynamic detail tag
Function testing	Login with correct details Attempt login with false details Extract details from text

	Combine extracted details with tags Logout Share contacts
User interface testing	Check screen validation on below activities • Login page • Home page • Contact View page • Edit view page Verify navigations
Performance testing	Time taken to display home screen Time taken to extract letters from image Time take to login a user Time taken to add a contact to database Time taken to receive a image from gallery
Security & access control	Login Online backup

3. Test Approach

3.1 Testing Techniques and Types

3.1.1 Data and Database Integrity Testing

Technique Objective:	Ensure that data accessing and storing functions works properly Ensure that data is stored and retrieved without any data corruption (consider the data type of the data) in the database Data stored in the shared preference are stored and retrieved correctly.
Technique:	Give wrong username and password for the login and see the response from database. Try to add same contact twice. Try to add different contacts with same phone number or details. Check whether all the data are stored on add function.

Oracles:	Can be achieved by debugging and unit testing. This provide output by comparing of output from the method with expected outcomes. Some of the methods used for it are: 'AssertEqual(Expected,Actual)'. , 'AssertTrue(Expected,Actual)
Required Tools:	Junit4 SqliteBrowser Android studio
Success Criteria:	No duplicates saved on database. Alerted when try to add a contact with same details that exist on the database. All the data are stored correctly.
Special Considerations:	Not organized sql queries will lead databases to inconsistent state when unit testing.

3.1.2 Function Testing

3.1.2.1 Create a user account and change details

Technique Objective:	At the installation time application prompt user for create an account with username and password. At any time, these stored data (password and user name) should able to be checked. Testing these functionality is to verify account details are stored correctly and retrieved correctly.
Technique:	1. Unit testing Process of storing encrypted password and compare them to verify authentication are tested. When changing password, it should be verified that current password is correct and newly entered password and re-entered password is correct. Method that ensure those process is being carried out properly when the changing account details are tested. 2. Debugged in Mobile device to emulate hardware and software operations.

Oracles:	In unit testing this can be verified with ‘AssertEqual(Expected,Actual)’, ‘AssertTrue(Expected,Actual). The created unit testing method will determine the whether application authenticate a user only with correct info other wise it will log about the problem.
Required Tools:	• Unit testing tool available in android studio • Android Mobile device for manual testing.
Success Criteria:	If authentication worked correctly unit test method will log “authentication ok” else it will log “wrong authentication” If password stored encrypted and correctly it will log “encrypted and success”. Else it will log “failed to store”.
Special Considerations:	To increase accuracy unit testing done several times.

3.1.2.2 Add contact detail

Technique Objective:	Related functions Create contact instance Add to the pool Add to the database
Technique:	Unit testing Some of the text manually created and tested with Junit that how extracted text changes into contacts.
Oracles:	In unit testing this can be verified with ‘AssertEqual(Expected,Actual)’, ‘AssertTrue(Expected,Actual). The created unit testing method will determine the whether application authenticate a user only with correct info other wise it will log about the problem.
Required Tools:	• Unit testing tool available in android studio • Android Mobile device for manual testing.
Success Criteria:	If authentication worked correctly unit test method will log “authentication ok” else it will log “wrong authentication” If password stored encrypted and correctly it will log “encrypted and success”. Else it will log “failed to store”.
Special Considerations:	To increase accuracy unit testing done several times.

3.1.3 User Interface Testing

Technique Objective:	The goal of UI testing is to ensure that the UI provides the user with the appropriate access and navigation through the functions of the target-of-test. This software involves user interaction via User Interfaces. To verify all the UIs works properly in worst cases as they works in best cases, User Interface Testing is carried out. Testing the functionality of UI widgets on each interface is carried out, Placement and orientation of widgets on the user interface grid, Loading data into particular widgets and Event handling functions of the widgets are tested.
Technique:	Verify all navigations Check screen validations Check buttons presses Check whether the user can go back and forward from any particular screen.
Oracles:	3 types of UI tests are created: By entering valid data By entering invalid data By entering missing data
Required Tools:	espresso
Success Criteria:	All window objects can be exercised, proper navigation through test target, and test target acts as expected. All button presses are working correctly. User can go easily back and forward
Special Considerations:	Navigation through hardware button also works perfectly.

3.1.4 Performance Profiling

Technique Objective:	Identify how expensive tasks work and messure the performance with the varying code will give us the flexibility to create faster applications. Some of the time expensive functions: 1. Extract letters from image 2. Create tags on created text 3. Sqlite queries.
Technique:	Monitor performance variance with code variance and improve some algorithms or query.
Oracles:	Once running the application if it is pushing or exceeding hardware constraints causes the app to be slow, have bad display performance, or exhaust the battery.
Required Tools:	Trace view and android ide with emulator.
Success Criteria:	If all the functions doesn't affect much of the screen visibility and battery usage application has no problem with performance
Special Considerations:	Performance testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement. The databases used for Performance Testing should be either actual size or scaled equally.

3.1.5 Load Testing

Technique Objective:	There is no need for parallel load test because this is a offline application. But there is possibility of lack of performance when invalid inputs given.
Technique:	Measure the time for a function to process a valid input and valid input after an invalid input. Monitor UI crashes on invalid inputs.
Oracles:	Unit testing tools and UI testing tools are used.
Required Tools:	Trace view, android ide and android emulator.
Success Criteria:	No crashes or no lack in performance while invalid inputs

Special Considerations:	• Load testing should be performed on a dedicated machine or at a dedicated time. This permits full control and accurate measurement. • The databases used for load testing should be either actual size or scaled equally

3.1.6 Security and Access Control Testing

Technique Objective:	This is an application developed for any Android Phone users to use freely. Once the product is purchased from the App Store, the service can be received. There are no limitations for access control. Security and access control testing is to protect the users of the platform any corruptions. To verify the application response properly for valid and invalid password
Technique:	User authentication Encryption of password
Oracles:	Encrypted password will be saved in shared preference file and also database. Authentication procedure will increase the data confidentiality and integrity of the application.
Required Tools:	Junit Android ide
Success Criteria:	If correct password is entered “Authentication is successful” message will be displayed and transactions will be continued.
Special Considerations:	Application considered as a stand alone application.

3.1.7 Failover and Recovery Testing

Technique Objective:	Failover and recovery testing ensures that the target-of-test can successfully failover and recover from a variety of hardware, software or network malfunctions with undue loss of data or data integrity Objective of running failover testing is to ensure that, when a failover condition occurs, the alternate or backup systems properly “take over” for the failed system without any loss of data or transactions.

Technique:	- Power interruption to the application. - Test the application at instance of low battery of the device. - Inserting invalid input and output
Oracles:	Application doesn't deal with any transactions, business logics so no need to worry about the power failure when accessing data
Required Tools:	Mobile device which run application.
Success Criteria:	When invalid data are input by user, It will be clearly notified to the user by error message without crashing the application.

3.1.8 Configuration Testing

Technique Objective:	To verifies the operation of the target-of-test on different software and hardware configurations.
Technique:	- Testing on different emulators - Testing on deferent mobile phones which runs android OS
Oracles:	When application is tested on different devices, the proper layout of the User interface should be tested to guarantee if their positioning and sizing are up to the standard.
Required Tools:	Different android mobile phones and different type of emulators
Success Criteria:	The application should be compatible to be released as a universal app if the configuration tests are passed.
Special Considerations:	This can be integrated with user acceptance test for feedback from user.

4. Deliverables

4.1 Test Evaluation Summaries

Testing is carried out throughout the project form the initial stage to the end. Testing is carried out after adding new functionality and changing the already implemented functionality. This was helpful to identify deviation of the system functionality and to ensure system is work as it is intended. Testing is carried out under several categories as mentioned above.

Functional Testing :
Techniques that used for functional testing are Unit testing and UI testing. Some functions are evaluated by the returned messages which indicate failure or success of the functionality. Other functional testing was based on the return value. The return value should be equal to the expected value, to consider a test is successful.

UI Testing :
Expectations of the UI testing was to verify all the UIs function properly, UIs do not get corrupted with invalid inputs and application provides proper messages when the user do unacceptable operation

Performance profiling :
Performance profiling is carried out to Identify expensive tasks that affect to performance and proceed them with different approach and Check how the application execution affect to the device performance. These testing result may helpful to make decision of improving the application performance.

Load Testing : This application is a stand-alone application which has a simple, light weight SQLite database. So there are no identifiable issues which drops the performance drastically when work load is increasing. But this test was carried out to observe the behavior of the system when there are many transactions happens

Security and access control testing : Since this application is a stand-alone application and the back end database is a simple, light weight and locally located SQLite database there is no major issues with security.

Failover and recovery testing : Application is tested at the instance of device running low battery. Device switched off after certain time of working with application.

Configuration testing : By doing configuration testing on several devices, compatibility of the application to run on any android device will be ensured.

4.2 Reporting on Test Coverage

Every iteration releases a report on testing. It will also include which bugs are found out and how they have been fixed out. The report contains testing process, technology and test cases used for each testing strategy. In recording test cases, details for each test, the exact steps necessary to perform the test, the expected results and observed results are included. That comprises both the plan and the results of executing the plan. A tabular test recording is used and the tables are filled as the testing process proceeds.

5. Risks, Dependencies, Assumptions, and Constraints

Risk	Mitigation Strategy	Contingency (Risk is realized)
Text extract performance can be vary regarding to mobile.	<Tester> Should identify minimum level of hardware feature for better performance	Need to state minimum hardware features for better user experience.
Some of the UI feature can vary regarding to tablet or different devices. Tested only on some mobiles.	<Tester> Need to determine which are the devices that supported by current version.	Add some support tools for all type of devices.
In long term usage performance can be degraded because of large data of the database.	<customer> Should occasionally delete some information if large data doesn't support on that mobile device <Tester>. Should store massive data to database to check this case	Need to state minimum storage for better performance

6. References

Espresso : https://developer.android.com/training/testing/ui-testing/espresso-testing.html
Trace View : https://developer.android.com/studio/profile/traceview.html