Test Plan Document
NotifyMe

Group 10
Deepanker Mishra - 2013CS50282,
Garvit Jain - 2013CS50284,
Shivanshu Gupta - 2013CS50298

April 26, 2017

1. Introduction

1.1 Purpose

The purpose of this document is to describe the plan for testing the android application NotifyMe. This test plan document supports the following objectives.

- Identify existing project information and the software that should be tested.
- Recommend and describe the testing strategies to be employed.
- Approach for executing the test plan.
- Resources required for testing.

1.2 Scope

This test plan describes the unit and system tests that will be conducted on the NotifyMe android application. It will describe the test items, the features to be tested, the pass/fail criteria and the physical requirements for executing the test cases.

1.3 References

- IEEE. IEEE Std 829-1983 IEEE Standard for Software Test Documentation
- NotifyMe Project Design Document

1.4 Definitions, acronyms and abbreviations

- **User:** Someone who interacts with the mobile phone application
- **App:** is abbreviation for Application and may occur in this document at many places.
- **SMS:** Short Message Service, which is the medium most companies use these days to update their customers on recent coupons, discounts and offers.
- **DB:** stands for Database of the application.
- **Promo:** is any promotional offer received via SMS or as a notification from an e-commerce application.
- **Vendor:** is the source of the promotional offer. In most cases it will be an e-commerce application.

2. Test Items (Functions)

This section lists the various functions that will be tested.

2.1 Notification Listener

The Notification Listener should be able to tell when a SMS/notification has been received on the smart-phone, all the time.

2.2 Text Processing

- **Check if Promo** - Correctly check whether the message is a promo or not.
- **Category of Promo** - The promo should be placed in the correct category according to the context of the promo.
- **Discount Value** - The discount amount/percentage returned after processing text should be the same as given in the promo
- **Coupon Code** - The exact coupon code should be returned as present in the promo
- **Validity** - The validity of the promo, if returned, should atleast be before the actual expiry of the promo, if not exactly the same. (Conservative approach)
- **Max Uses available** - The maximum allowed uses of the promo, if returned, should atleast be less than the actual allowed uses of the promo.

2.3 Promo Database

- **Add Promos** - The database should insert promo into the database after the Text Processing Module parses a message with the isPromo() function returning true
- **Filter Query** - The different queries applied on the database corresponding to the filters should return the relevant promos.
- **Order query** - Query used to order the promos according to some field should output the ordered promos.
- **Delete Promo** - Query to delete a promo should remove the promo from the database.

2.4 User Interface

- **Settings page** - The settings page should be displayed on tapping the 'Settings' option.
- **Promo Description** - Any promo can be selected to display the full promo message in a dialog box.
- **Filter Selection** - User can select the filters he wishes to apply.

2.5 System tests

- **Starting Page Selection** - On selecting a category on the landing page, the promos relevant to that category should be displayed on the screen.
- **Category Switch** - User can tap on any category to view promos of that category.
- **Settings change** - Any change in settings should be reflected by the application.
- **Filter application** - Any filter applied should show only those promos satisfying the filter criteria.
- **Filter time** - Any filter applied should display the results within one second.
- **Max Promos** - Application should work properly with atleast 100 promos.

3. Features to be Tested

- **Ordering of Promos** - Promos ordered by score, validity, etc. in ascending or descending order.
- **Filters** - The different filters that can be applied to each filter, like filter by vendor, filter by score, filter by validity, etc.
- **Landing Page** - The starting page of the application with the categories showing the correct promos.
- **Delete Promos** - Any promo can be selected and deleted by the user.
- **Settings** - User can select which apps' promos should be displayed, and whether notifications should be shown or hidden.

4. Features Not to be Tested

Since all the components of the application have been developed from scratch, and no 3rd party components have been used, all the features have to be tested.

5. Pass/Fail Criteria

5.1 Module Pass/Fail Criteria

Tests executed on individual modules pass only when they satisfy the module definitions, objectives and constraints as specified in the Project Design Document. This includes positive tests, negative tests and boundary tests. If a test exhibits a product failure to meet the objectives, it will fail and a defect issue will be reported.

5.2 System Pass/Fail Criteria

Tests executed against the system use the functional requirements, non-functional requirements, and use cases as the oracle to determine pass or fail. If a test exhibits a product failure to meet the objectives of any of the functional requirements, non-functional requirements, or the use cases, it will fail and a defect/issue will be reported.

6. Test Strategy

This section describes the testing strategy to be employed to test the application. Different strategies are often needed to test different parts of the system. First unit testing and component testing will be performed on the components as they are developed. Then system testing will be performed once the application is integrated and developed.

6.1 Database Testing

The databases and the database processes should be tested as separate systems. These systems should be tested without the applications (as the interface to the data).

6.1.1 Test Objective

Ensure Database access methods and processes function properly and without data corruption.

6.1.2 Technique

- Invoke each database access method and process, seeding each with valid and invalid data (or requests for data).
- Inspect the database to ensure the data has been populated as intended, all database events occurred properly, or review the returned data to ensure that the correct data was retrieved (for the correct reasons)

6.1.3 Completion Criteria

All database access methods and processes function as designed and without any data corruption.

6.2 Function Testing

Testing of the application should focus on any target requirements that can be traced directly to use cases. The goals of these tests are to verify proper data acceptance, processing, and retrieval.

6.2.1 Test Objective

Ensure proper application navigation, data entry, processing, and retrieval.

6.2.2 Technique

Execute each use case, use case flow, or function, using valid and invalid data, to verify the following:

- The expected results occur when valid data is used.
- The appropriate error / warning messages are displayed when invalid data is used.

6.2.3 Completion Criteria

All planned tests have been executed and all identified defects have been addressed.

6.3 User Interface Testing

User Interface testing verifies a user's interaction with the software. The goal of UI Testing is to ensure that the User Interface provides the user with the appropriate access and navigation through the functions of the applications.

6.3.1 Test Objective

Verify the following:

- Navigation through the application properly reflects functions and requirements, including window to window, field to field, and use of different gestures (tap, double tap, swipe)
- Window objects and characteristics, such as menus, size, position, state, and focus conform to standards.

6.3.2 Technique

Create / modify tests for each window to verify proper navigation and object states for each application window and objects.

6.3.3 Completion Criteria

Each window successfully verified to remain consistent with benchmark version or within acceptable standard

6.4 Performance Testing

Performance testing measures response times, transaction rates, and other time sensitive requirements. The goal of Performance testing is to verify and validate the performance requirements have been achieved.

6.4.1 Test Objective

Validate System Response time for designated transactions.

6.4.2 Technique

- Use test scripts developed for system testing.
- Modify data files (to increase the number of transactions) or modify scripts to increase the number of iterations each transaction occurs.
- Script should be run on one machine.

6.4.3 Completion Criteria

Successful completion of the test scripts without any failures and within the expected / required time allocation (per transaction).

6.5 Load Testing

Load testing measures subjects the system-under-test to varying workloads to evaluate the system's ability to continue to function properly under these different workloads. The goal of load testing is to determine and ensure that the system functions properly beyond the expected maximum workload.

6.5.1 Test Objective

Verify System Response time for designated transactions under varying workload conditions.

6.5.2 Technique

- Use tests developed for system testing.
- Modify data files (to increase the number of transactions) or the tests to increase the number of times each transaction occurs.

6.5.3 Completion Criteria

Successful completion of the tests without any failures and within acceptable time allocation.

7. Test Deliverables

- Test Plan
- Test DataSets

- Test Scripts
- Test Results

8. Environmental needs

8.1 Hardware Requirements

Android smart-phone with Operating System Lollipop (v5.0) or above and having cellular network.

8.2 Software Requirements

We need Android Studio to develop and test the application and SQLite to maintain the database.