K. J. Somaiya College of Engineering, Mumbai-77

(Autonomous College Affiliated to University of Mumbai)

Batch: A1	**Roll No.: 16010120015**
	16010120013
	16010120020
Experiment / assignment / tutorial No.____7___	

Title: Designing test plan document for Mini Project

Aim: To learn and understand the way of developing the software by classical methods of software engineering. Planning and monitoring, testing, validating of the project using tools and prepares a document for the same by using the concept of software engineering

---

CO: Test the given software for different test cases with proper test planning.

Books/ Journals/ Websites referred:

1. Roger Pressman, Software Engineering: A practitioners Approach, McGraq Hill, 2010 ,6th edition
2. Ian Somerville , Software Engineering , Addison Wesley,2011,9th edition
3 http://en.wikipedia.org/wiki/Software_requirements_specification

---

Test Plan Template:

Spotify , Music Player App

Prepared by:

16010120013 - Rohan Dsouza

16010120015 - Yash Gavade

16010120020 -Khushi Kenia

Date 25/11/2022

TABLE OF CONTENTS

1.0 INTRODUCTION

Spotify is an application that allows users to listen to music online. This product comes to rescue in case of weak internet connection. It is an easy to use and beautifully designed music player to enhance the user's experience.

Product Functions:

- Play/Pause
- Favorite/Like button – all liked songs will be showed in a separShuffle – will
- play random songs
- Next track
- Previous track
- Albums
  Songs – will show the list of all songs
- Artists – will show songs based on artists
- Top tracks – will show the list of top tracks
- Recently played – will show the list of recently played songs

2.0 OBJECTIVES AND TASKS

2.1 Objectives

The document's major goal is to create a master test plan that thoroughly evaluates all of the software's essential aspects, such as the database, user interface for all operations, correctness of the output, processing efficiency, and so on. To guarantee

that all components of our project run smoothly, the team will use a variety of testing

approaches and create a variety of test cases.

The following are some the features that will be tested:

● Proper loading of songs

● Playing/Pausing of songs

● Creation of favourites playlist

● Display and accuracy of different albums

● Recommendation of songs based on artists

2.2 Tasks

Pre-Testing: Designing the test cases corresponding to each unit.

Unit Testing: Testing all the test cases for the pool of possible inputs.

Problem reporting: Reporting the test cases which do not provide desired results

System Testing: Once all test cases in a Unit are successful, specify the test cases which have to be tested once the unit has been integrated with the system.

3.0 SCOPE

General

Features to be tested:

- Proper loading of songs
- Playing/Pausing of songs
- Creation of favourites playlist
- Display and accuracy of different albums
- Recommendation of songs based on artists

Hardware and Software Interfaces: Saviour, being a music player app, requires an android device with android version 7.0 or above and or an iOS device running iOS 9.0 or above. Saviour is simply an offline music player app hence an active internet connection is not required. Moreover, if recommendations are required, a weak internet connection may be required to fetch the recommendations from the lastFM API. A working set of inbuilt speakers or external audio devices are required.

Communication interfaces: Since the application is offline there is little to no communication transfer. The recommendation system which uses the lastFM API sends a http request to fetch and send the user data and retrieves the recommended album

Tactics

A proper schedule will be made and after each unit is tested by the testing team the feedback will be given to the development team. After all the units are modified according to the feedback received, they will be integrated by the development team and will be sent to the testing team so that the system can be tested as a whole. After the whole system is tested and modified, feedback from the user will be taken and the system will be enhanced accordingly.

The test results and subsequent action taken to fix errors and bugs will be well documented to avoid confusion between the development and the testing teams and to have a proper and well-managed track of each results and actions taken.

4.0 TESTING STRATEGY

The strategy for testing is to break down the application into small units and test each unit to ensure proper functioning of each unit. The team also plans to integrate the small units and perform testing on the integrated system to ensure that there is smooth functioning of the application as a whole. Testing strategies planned to be used:

- Unit Testing
- System and Integration Testing
- Performance and Stress testing
- User Acceptance Testing
- Batch testing
- Beta Testing

4.1 Unit Testing

Definition:
Specify the minimum degree of comprehensiveness desired. Identify the techniques Individual units of source code ie. sets of one or more computer programme modules, as well as accompanying control data, usage processes, and operating procedures are examined to see whether they are fit for use in unit testing. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software.

Participants:

- Rohan Dsouza
- Yash Gavade
- Khushi Kenia

Methodology:
The units will be made according to the different functionalities. For each unit a test plan will be designed containing various test cases. Each team member will test different units to ensure that the units are being tested simultaneously and the schedule can be followed.

4.2 System and Integration Testing

Definition:
System Integration Testing (SIT) is the overall testing of the whole system which is composed of many sub-systems. The main objective of SIT is to ensure that all software module dependencies are functioning properly and the data integrity is preserved between distinct modules of the whole system.

Participants:

- Rohan Dsouza
- Yash Gavade
- Khushi Kenia

Methodology:
After each individual unit is tested for all it's test cases, the separate units will be integrated and each test case will be tested again. The features and functionality to be kept in mind include:

● Each and every feature of individual units must provide correct and desired results.

● The data consistency must be maintained in all the units while updating/deleting data in a particular feature.

● The user interfaces must appear uniform throughout.

● There must be a smooth transition and navigation to use features of two different units.

● One unit must not affect the working of another unit.

Each of these requirements will be looked for in each test case during the integration testing. The schedule will be planned in a way that the tasks of a similar group will be allotted to a particular participant and will be carried out in an asynchronous manner.

4.3 Performance and Stress Testing

Definition:
Stress testing is defined as a type of software testing that verifies the stability and reliability of the system. This test particularly determines the system on its robustness and error handling under extremely heavy load conditions. It even tests beyond the normal operating point and analyses how the system works under the extreme conditions. Stress testing is performed to ensure that the system would not crash under crunch situations. Stress testing is also known as Endurance Testing or Torture Testing.

Participants:
● Rohan Dsouza

● Yash Gavade

● Khushi Kenia

Methodology:
The system behaviour will be monitored when the memory of the phone is entirely filled with songs and the database is loaded under a very heavy intake of new data. Also, a low end device will be used in this process so as to ensure the proper functioning in the worst case scenario.

4.4 User Acceptance Testing

Definition:
The purpose of the acceptance test is to confirm that the system is ready for operational use. During the acceptance test, end-users (customers) of the system compare the system to its initial requirements. UAT is done in the final phase of testing after functional, integration and system testing are done.

Participants:
- Rohan Dsouza
- Yash Gavade
- Khushi Kenia

Methodology:
Acceptance criteria are defined on the basis of the following attributes:

- Functional Correctness and Completeness
- Data Integrity
- Confidentiality and Availability
- Install-ability and Upgrade-ability
- Performance
- Scalability
- Documentation

● Usability

Hence, we will test for all the above mentioned attributes and check whether the user's requirements are satisfied and verify it accordingly.

4.5 Batch Testing

A batch test occurs when you run multiple scripts. It is typically done with automation. You program a batch test by placing the scripts in the order you wish to have them run and employing a tool that will execute the scripts in that specified order.

Participants:

● Rohan Dsouza

● Yash Gavade

● Khushi Kenia

Methodology:

According to the definition of batch testing, we will arrange the scripts of different modules in a queue. An example queue of scripts to be tested: loading songs section, playing a song section, favourites section and so on. These sections will consist of multiple dependent test cases and at the end of batch testing all the interconnected test

4.6 Automated Regression Testing

Definition:
Regression tests ensure that code that was previously functioning correctly does not regress when changes are made. Having a comprehensive suite of unit level regression tests provides a safety net, making sure that code changes do not break existing functionality

Participants:

- Rohan Dsouza
- Yash Gavade
- Khushi Kenia

Methodology:

Regression testing will be done periodically after each and every significant update is delivered to the application. It will be done to ensure that after each update none of the previous updates have been bricked to ensure proper functioning of the music player application.
For Example,
Complete app will be tested when the favourites button is added since the database will now label favourite songs separately.

4.7 Beta Testing

Definition:

Beta testing is also known as user testing takes place at the end users site by the end-users to validate the usability, functionality, compatibility, and reliability testing. It will allow the "real" customer an opportunity to provide inputs into the design, functionality, and usability of a product. Beta Testing is performed by real users of the software application in a real environment

Participants:

- Rohan Dsouza
- Yash Gavade
- Khushi Kenia

Methodology:
We plan on taking feedback from a few of our fellow classmates with the help of Google forms. The form shall consist of various questions that will help us understand the usability and reliability of the application from the user's point of view. Any constructive suggestions will be worked upon in the future scope of the application

5.0 HARDWARE REQUIREMENTS

Computers

64-bit Microsoft® Windows® 8/10

● x86_64 CPU architecture; 2nd generation Intel Core or newer, or AMD CPU with support for a Windows Hypervisor

● 8 GB RAM or more

● 10 GB of available disk space minimum (IDE + Android SDK + Android Emulator)

● 1280 x 800 minimum screen resolution

● 1.6 GHz or faster processor Smartphone

● 1 GB Ram, 16 GB ROM, minimum Android version 7

● Any iPhone running iOS 9 or later

Smartphone

● 1 GB Ram, 16 GB ROM, minimum Android version 7

● Any iPhone running iOS 9 or later

6.0 ENVIRONMENT REQUIREMENTS

6.1 Main Frame

● Virtual Storage
● Android Studio

- BatchProcessing
- Time Sharing
- Android and iOS Smartphone

6.2 Workstation

A physical device (Android or iOS) connected to your computer and set

- to developer mode.

- Android Studio with the iOS simulator (Requires installing Xcode tools) and the

Android emulator (Requires setup in Android Studio.)

- Dart

- Flutter

- Visual Studio Code

- JDK

Date	Milestones	Output
8th June 2021 to 10th June 2021	Unit Testing	Documentation and feedback.
10th June 2021 to 12th June 2021	System integration testing	Developer's comments, automated tools report
12th June 2021 to 14th June 2021	Performance testing, Stress testing	System performance report
14th June 2021 to 16th June 2021	User Acceptance testing	User's feedback
16th June 2021 to 19th June 2021	Batch testing	Combined module report
19th June 2021 to 21th June 2021	Beta testing	User's feedback

7.0 TEST SCHEDULE

Include test milestones identified in the Software Project Schedule as well as all item transmittal events.

Define any additional test milestones needed. Estimate the time required to do each testing task. Specify the schedule for each testing task and test milestone. For each testing resource (that is, facilities, tools, and staff), specify its periods of use.

8.0 CONTROL PROCEDURES

Problem Reporting

While testing a particular unit or the system, if there is any problem encountered by the testing team, it will be documented and added to the test analysis report which will be

handed over to the development team. Also the results of all the test cases tested will be added to the report. Based on the analysis report, the development team will resolve the issues faced and send the system back to the testing team for further testing if required.

Change Requests
The changes to be made will be identified by a unique id and documented by the testing team. This document with changes to be done/errors to be corrected will be accessed by the development team who will document the actions taken and further attach sub-ids to each module and their features affected by the change. This will then be tested by the testing team and flagged as resolved when desired results will be obtained.

9.0 FEATURES TO BE TESTED

Sr.no	Test case	Description	Intended result	Actual result
1	**Slider**	The slider of the song is used to move through the song ie. Go backward/forward in the song	On sliding the slider, you should be able to move through the song	The slider works perfectly and one can move through the song using it
2	**Favourites**	To check if on pressing the favourite icon in a song, it should be added to the favourites list	Once you click on the favourites button of a song, the song name should appear in the favourites playlist	The favourites list incorporates all the songs that have been clicked favourite by the user
3	**Home Screen**	To check if the home screen is scrollable so that all the various components like recent songs/top albums/ you may like sections are visible	When one scrolls through the screen, it shouldn't freeze but smoothly scroll through up and down	The screen is scrollable and hence all the options are seen one after the other when one scrolls down
4	**Bottom Navigation Bar**	This bar is used to navigate through various pages of the app	By clicking on different options of the bottom navigation bar, you should be able to navigate through the 5 different screens of the app	This functionality works and hence the user can visit different pages of the app using the bottom navigation bar

5	**Shuffle**	This functionality allows the user to play random songs from the list of all the songs in his/her phone	Once you click on the shuffle button, the app will randomly select songs from your playlist and play them for you	This button works and once pressed randomly plays the songs
6	**Download a new song**	When the user downloads a new song, it should appear in the all-songs list	Once a new song is downloaded and user refreshes the app, it should be added in the database	This functionality works and once a new song is downloaded, app is refreshed, the song is added to the database and also appears in the list of songs on the app

10.0 FEATURES NOT TO BE TESTED

- The format of the music files (.mp3) will not be checked or tested.
- The user manual for the app is not to be tested.

11.0 RESOURCES/ROLES & RESPONSIBILITIES

Developers and testers:

- **Rohan Dsouza** - preparing and designing the test case activities and providing the environment for testing.
- **Yash Gavade** - preparing and designing the test case activities and providing the environment for testing.
- **Khushi Kenia** - Executing the testing and providing the test analysis report for the results.

12.0 SCHEDULES

Major Deliverables

Test Plan document: Specifies the testing methods, communication method between development and testing teams and the schedules.

- Test Cases: Specifies individual test cases against which units and modules will

be tested.
Test Incident Reports: Contains a log of test cases, actions taken and results.
● Requirements manual

13.0 SIGNIFICANTLY IMPACTED DEPARTMENTS (SIDs)

● Testing team

● Development team

14.0 DEPENDENCIES

The software units/modules must be available and completed for the test schedule to be followed as planned

15.0 RISKS/ASSUMPTIONS

The testing approach and schedule is planned with an assumption that there will be no major changes in the system architecture during the debugging phase. There is a riskthat the system might crash while testing which may lead to delay in the schedule. It is assumed that the environment and resources required for testing will be available thereby not causing any hurdle in the test schedule.

16.0 TOOLS
The application will be tested manually with the help of test documents

17.0 APPROVALS

Name: PROF. POONAM BHOGALE
Signature:
Date:

Conclusion:
Hence we understood about the various testing methods and the need for a testing plan document. A test plan document for the system was created successfully.

Post Lab Descriptive Questions:

1. **Distinguish between Black Box and White Box Testing**

Black Box Testing	White Box Testing
It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it.	It is a way of testing the software in which the tester has knowledge about the internal structure or the code or the program of the software.
It is mostly done by software testers.	It is mostly done by software developers.
No knowledge of implementation is needed.	Knowledge of implementation is required.
It can be referred to as outer or external software testing.	It is the inner or the internal software testing.
It is a functional test of the software.	It is a structural test of the software.
This testing can be initiated on the basis of requirement specifications document.	This type of testing of software is started after a detailed design document.
No knowledge of programming is required.	It is mandatory to have knowledge of programming.
It is the behavior testing of the software.	It is the logic testing of the software.
It is applicable to the higher levels of testing of software.	It is generally applicable to the lower levels of software testing.
It is also called closed testing.	It is also called clear box testing.

2. Consider following scenario: An institute is interested in developing a Library Information System (LIS) for the benefit of students and employees of the institute. LIS will enable the members to borrow a book (or return it) with ease while sitting at his desk/chamber. The system also enables a member to extend the date of his borrowing if no other booking for that particular book has been made. For the library staff, this system aids them to easily handle day-to-day book transactions. The librarian, who has administrative privileges and complete control over the system, can enter a new record into the system when a new book has been purchased, or remove a record in case any book is taken off the shelf. Any non-member is free to use this system to browse/search books online. However, issuing or returning books is restricted to valid users (members) of LIS only.

   The final deliverable would a web application (using the recent HTML 5), which should run only within the institute LAN. Although this reduces security risk of the software to a large extent, care should be taken no confidential information (e.g. passwords) is stored in plain text.

Assume following test suite is used

A test suite to verify the "User Login" feature						
#	**TS1**					
Title	**Verify "User Login" functionality**					
Description	**To test the different scenarios that might arise while an user is trying to login**					
#	**Summary**	**Dependency**	**Pre-condition**	**Post-condition**	**Execution Steps**	**Expected Output**
TC1	Verify that user already registered with the LIS is able to login with correct user ID and password		Employee ID *149405* is a registered user of LIS; user's password is *this_is_password*	User is logged in	1. Type in user ID as *149405* 2. Type in password *this_is_password* 3. Click on the 'Login' button	"Home" page for the user is displayed
TC2	Verify that an unregistered user of LIS is unable to login		Employee ID *149405xx* is not a registered user of LIS	User is not logged in	1. Type in employee ID as *149405xx* 2. Type in password *whatever* 3. Click on the 'Login' button	The "Login" dialog is shown with a *"Login failed! Check your user ID and password"* message

A test suite to verify the "User Login" feature	
#	**TS1**
Title	**Verify "User Login" functionality**
Description	**To test the different scenarios that might arise while an user is trying to login**

#	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected Output
TC3	Verify that user already registered with the LIS is unable to login with incorrect password		Employee ID *149405* is a registered user of LIS; user's password is *this_is_password*	User is not logged in	1. Type in employee ID as *149405* 2. Type in password *whatever* 3. Click on the 'Login' button	The "Login" dialog is shown with a *"Login failed! Check your user ID and password"* message
TC4	Verify that user already registered with the LIS is unable to login with incorrect password	TC3	This test case is executed after execution of TC3 before executing any other test case	User is not logged in	1. Type in employee ID as *149405* 2. Type in password *whatever2* 3. Click on the 'Login' button	The "Login" dialog is shown with a *"Login failed! Check your user ID and password"* message

A test suite to verify the "User Login" feature	
#	**TS1**
Title	**Verify "User Login" functionality**
Description	**To test the different scenarios that might arise while an user is trying to login**

#	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected Output
	given twice consecutively					
TC5	Verify that user already registered with the LIS is unable to login with incorrect password given thrice consecutively	TC4	This test case is executed after execution of TC4 before executing any other test case	User is not logged in	1. Type in employee ID as *149405* 2. Type in password *whatever3* 3. Click on the 'Login' button	The "Login" dialog is shown with a *"Login failed! Check your user ID and password"* message; the security question and input box for the answer are displayed
TC6	Verify that a registered user can login after three	TC5	This test case is executed after execution of TC6 before executing any	Email sent containing new password.	1. Type in the answer as *my_answer* 2. Click on the 'Email Password' button	Login dialog is displayed; an email containing the new password is received

A test suite to verify the "User Login" feature	
#	**TS1**
Title	**Verify "User Login" functionality**
Description	**To test the different scenarios that might arise while an user is trying to login**

#	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected Output
	consecutive failures by correctly answering the security question		other test case. Answer to the security question is *my_answer*.	The email is expected to be received within 2 minute.		
TC7	Verify that a registered user's account is blocked after three consecutive failures and answering the security		Execute the test cases TC3, TC4, and TC5 once again (in order) before executing this test case	User account has been blocked	1. Type in the answer as *not_my_answer* 2. Click on the 'Email Password' button	The message *"Your account has been blocked! Please contact the administrator."* appears

A test suite to verify the "User Login" feature	
#	**TS1**
Title	**Verify "User Login" functionality**
Description	**To test the different scenarios that might arise while an user is trying to login**

#	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected Output
	question incorrectly					

create a Requirements Traceability Matrix (RTM) showing a mapping from individual requirement to test case(s).

Table 1: A simplified mapping from requirements to test cases	
Requirement #	**Test Case #**
R1	
R2	
R3	
R4	

Consider requirements are summarized in the table below

#	Requirement
R1	New user registration
R2	User Login
R3	Search book
R4	Issue book
R5	Return book
R6	Reissue book

Ans :

Table 1: A simplified mapping from requirements to test cases	
Requirement #	**Test Case #**
R1	TC2
R2	TC1, TC3, TC4, TC5, TC6
R3	TC8 , TC9
R4	TC10 , TC12
R5	TC13, TC14
R6	TC11

A test suite to verify the "Search Book" feature	
#	**TS2**
Title	**Verify "Search Book" functionality**
Description	**To test the different scenarios that might arise while an user is trying to search book**

#	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected Output
TC 8	Searched Book exists	TC1	Employee ID *149405* is a registered user of LIS; searched book xyz exists	Book is found	1. Type in book name in SearchBox.	"Book" page for the user is displayed
TC 9	Searched Book doesn't exist	TC1	Employee ID *149405* is a registered user of LIS, searched book *xyz* does not exist	Book not found	1. Typing book name 2. Click on the 'Login' button	The " Search" page shows no book found

A test suite to verify the "Issue Book" feature	
#	**TS3**
Title	**Verify "Issue Book" functionality**
Description	**To test the different scenarios that might arise while an user is trying to issue book**

#	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected Output
TC 10	Issue available Book	TC8	Employee ID *149405* is a registered user of LIS; searched book xyz exists; xyz book is available for issuing	Book is issued	1. Click on the issue button.	"My issues" page for the user is displayed
TC 11	Reissue Book	TC1	Employee ID *149405* has already issued the book	Book re-issued	1. Click on the Re-issue button.	due date gets extended and books appear in "my issue" page

TC 12	Book is unavailable for issuing	TC8	Employee ID *149405* is a registered user of LIS; searched book xyz exists; xyz book is unavailable for issuing	Book is not issued	1. Click on the issue button.	The message "*Book not available.*"

A test suite to verify the "Return Book" feature	
#	**TS4**
Title	**Verify "Return Book" functionality**
Description	**To test the different scenarios that might arise while an user is trying to return book**

#	Summary	Dependency	Pre-condition	Post-condition	Execution Steps	Expected Output
TC 13	Return book on Time	TC10 , TC11	Employee ID *149405* is a registered user of LIS; Book issued and return date not completed	Book returned	1. Click on the "return" Button.	"Book is returned"
TC 14	Return book Late	TC10 , TC11	Employee ID *149405* is a registered user of LIS, book is returned late	Book is returned late 7 penalty is charged	1. Click on the return Button. 2. Late fees is paid	"Book is returned & Late fees are paid"