Look It
Test Plan Report
For Classify the Face Emotions

Version 1.0

Revision History

Date	Version	Description	Author
05/Apr/19	1.0	Document created with initial plans	Sabesan S.

Table of Contents

Test Plan

1. Evaluation Mission and Test Motivation

1.1 Background

The project is about handling the results data of the Face API and handing them over to the users when their systems are online. When an analysis is done by an Face API using a certain dataset, the user will be notified through an Android Application informing that the analysis processing has been finished. Thereafter the user has to share the results of the analysis using the link provided in the Android Application. But when the user fails to collect the results within a certain period, the input dataset as well as the results data will be erased from the Face API. Thus, a platform has to be developed so as to receive the results data from Face API, hold them and send them to the user when the user's system is online. The Look It will be providing the solution for this. It will make sure that the user will receive the results data without loss.

A test plan has been created so as to make sure that the system conforms to the specifications, design and to perform quality assurance on the final product. It will help to remove bugs in the system as much as possible and reduce the risk of software failure. The test plan will help to verify whether the final product delivers the expected outcomes and fulfills the requirements successfully with a variety of testing techniques. Moreover, it will also help to detect faults by using the designed test cases.

1.2 Evaluation Mission

The main objectives of the testing plan is to make sure that the specified requirements of the SRS have been achieved successfully, making sure that the specifications of the architecture document have been achieved successfully, ensuring that the risk of software failure is minimized as much as possible and detecting the defects and correct them before go-live. Fulfilling these objectives will help to produce a stable version which obtains the expected outcomes successfully. Although there are no approaches or methods to guarantee that the system is completely free of defects, the following test plan will help to reduce faults of the system and provide an improved product.

1.3 Test Motivation

The targeted test items listed below will be the motivation for testing in this phase.

- **Unit testing:** It mainly focuses on the verification effort on the smallest unit of the software design. In unit testing the pass and fail criteria for each sub-system depends on the functionality of those sub-system. For this system, modules will be implemented independently as a stand-alone unit. Unit testing ensures that each module works correctly in isolation. Set of independently tested software modules is the output of this Unit testing. These modules are directly connected with the database.
- **Data and database integrity testing:** Units will be integrated with other units and checked whether they work together successfully. This is done by giving the input details programmatically and checking the activities.
- **Function testing:** Function testing would help to find whether all the use cases have been met successfully
- **User Interface testing:** This would help to identify whether the requirements of the user interfaces have been implemented as specified.
- **Configuration testing:** It ensures that the system works successfully under different environment configurations.
- **Performance profiling:** It makes sure that the performance of the system is up to an acceptable level.
- **Load testing:** This checks the performance of the system when it is used at its limits.

- **Security and access control testing:** It ensures to protect data and maintain functionality as intended and check whether the requests to access a particular resource should be granted or denied
- **Failover and recovery testing:** It verifies product in terms of ability to confront and successfully recover from possible failures, arising from software bugs, hardware failure or communication problems

2. Target Test Items

The targets for the testing are listed below.

- **Integration testing:** Integration testing is related to the sub-systems which are formed by integrating the small units of the system together one by one. It will check whether each sub systems work successfully by giving the input details programmatically and checking the activities. Capture Image, view received results, share results, suggest Videos, receive results from Emotion API , sent image to Emotion API and such are focused here.
- **Function testing:** Function testing deals with all the requirements and specifications which are expected from the system. The list of functions to test will correspond to the list of use cases and requirements in the requirements document. Registration, updating details, obtaining results from analytical platform, showing results details, sending results to customers and such are focused here.
- **User interface testing:** User interface testing related with checking whether each of the functionalities concerning the user interface works successfully according to the requirements. Moreover, it should check whether the user can go back and forward from any particular activity through button presses. Labels, buttons, text boxes, icons, links and such things will be tested. Home page, login page, registration page, change details page, view results page and such others will be focused here.
- **Performance profiling:** Performance profiling is related to the run time behavior of the system. Testing the different response times of the software and concurrent usage of the system by several customers are included in this type of testing. Registration of customers, requesting to download files, changing the details, login and such actions are focused in this regard.
- **Security and access control testing:** Security and access control deals with the testing for unauthorized control. Login, request to download files and such actions are mainly focused with regard here.

3. Test Approach

Tests will be automated by using the Espresso testing framework wherever possible. Before any part of the code is checked, it must successfully pass the appropriate unit test. If any faults are found, they should be fixed. Thereafter re-testing of the code will be done to make sure that defect has been fixed and there no faults produced due to change in code.

3.1 Testing Techniques and Types

3.1.1 Data and Database Integrity Testing

This testing verifies whether the data in the database is accurate and whether it functions as expected in the given application. Technique Objective:	Ensure Database access methods and processes function properly and without data corruption and Verify the data integrity of the imported schema independent of the UI.
Technique:	Give wrong user name and passwords and see the relevant outputs. Try to create more than one database with the same name Verify that user can create, modify, and delete any data in tables. Verify that, when a particular set of data is saved to the database, each value gets saved fully, and the truncation of strings and rounding of numeric values do not occur.
Oracles:	
Required Tools:	MySQL will be used for data examination MySQL Tools for manual backups Flapdoodle Embedded MySQL for testing
Success Criteria:	Users authenticated correctly. All database access methods and processes function as designed and without any data corruption, All schema elements can be stored and retried with no data loss
Special Considerations:	

3.1.2 Function Testing

Function testing would help to find whether all the use cases have been met successfully Technique Objective:	Ensure proper application navigation, data entry, processing, and retrieval and Verify system functional requirements.
Technique:	Ensure every function produces its expected outcome. Check whether the expected results occur when valid data is used. Ensure every line of code executes properly. Ensure new changes did not adversely affect other parts of the system. heck whether the appropriate error or warning messages are displayed when invalid data is used. Ensure all functions combine to deliver the desired result.
Oracles:	
Required Tools:	Espresso
Success Criteria:	Planned tests have been executed successfully. Identified defects have been addressed correctly.
Special Considerations:	

3.1.3 User Interface Testing

This would help to identify whether the requirements of the user interfaces have been implemented as specified. Technique Objective:	Navigation through the system interface by window to window, page to page and use of access methods (tab keys and mouse movements). The main objective is to evaluate the GUI of the system. Check whether the window objects and characteristics (menus, buttons, position, state and focus) are accurate
Technique:	Verify All Navigations Check Screen Validations Check whether the button presses are working correct Check whether the clickable buttons are showed correctly Check whether the user can go back and forward from any particular screen
Oracles:	

Required Tools:	Zombie
Success Criteria:	All window objects can be exercised, proper navigation through test target, and test target acts as expected. Each window successfully verified to remain consistent with benchmark version or within acceptable standard. All button presses are working correctly. It's Easy to identify clickable buttons. User can go easily back and forward
Special Considerations:	

3.1.4 Performance Profiling

It makes sure that the performance of the system is up to an acceptable level. Technique Objective:	Determine whether the system will be able to sustain the workload. Measures the quality attributes of the system, such as scalability, reliability and resource usage. Discover the system's performance under sustained use. Measure which parts of the system or workload cause the system to perform badly.
Technique:	Check the response time of the system is in acceptable level.
Oracles:	
Required Tools:	Android studio built in profiler
Success Criteria:	Single user: Successful emulation without any failures due to test implementation problems. Multiple users: Successful emulation of the workload without any failures due to test implementation problems.
Special Considerations:	

3.1.5 Load Testing

This checks the performance of the system when it is used at its limits. Technique Objective:	Verify System Response time, throughput, resource utilization, and maximum user load for designated transactions or business cases under varying workload conditions. Determine how the application behaves when multiple users hits it simultaneously.
Technique:	Check when there are more users try to access response time is acceptable.
Oracles:	
Required Tools:	nodeload
Success Criteria:	multiple users: Successful completion of the tests without any failures and within acceptable time allocation
Special Considerations:	

3.1.6 Security and Access Control Testing

It ensures to protect data and maintain functionality as intended and check whether the requests to access a particular resource should be granted or denied Technique Objective:	Data Security: Verify that user can access only those data for which their user type is provided permissions. System Security: Verify that only those users with access to the system and application(s) are permitted to access them.
Technique:	Test with different user name and password, and check whether the system allowed or give error messages.
Oracles:	
Required Tools:	Espresso
Success Criteria:	For each known user type the appropriate function / data are available and all function as expected and run in prior Application Function tests.
Special Considerations:	

3.1.7 Failover and Recovery Testing

It verifies product in terms of ability to confront and successfully recover from possible failures, arising from software bugs, hardware failure or communication problems Technique Objective:	Failover testing is a testing technique that validates a system's ability to be able to allocate extra resource and to move operations to back-up systems during the failure due to one or the other reasons. Discover the specific point at which failure occurs. Recovery testing is basically done in order to check how fast and better the application can recover against any type of crash or hardware failure
Technique:	While the application is running, suddenly restart the mobile phone or computer, and afterwards check the validness of the application's data integrity.
Oracles:	
Required Tools:	Espresso
Success Criteria:	One or more simulated recoveries involving one or more combinations of the application, database, and system to a known desired state.
Special Considerations:	Resources from the Systems (or Computer Operations), Database are required. These tests should be run after hours or on an isolated machine

3.1.8 Configuration Testing

It ensures that the system works successfully under different environment configurations. Technique Objective:	Verify the test-target functions correctly on different platforms and under different configurations.
Technique:	Normal test functions will be used to test the system with different configurations. Unit Tests as well as integration test and data-base access tests will be executed in different configurations. When configuring the system, the test configuration will be created to cover the maximum possible scope.
Oracles:	
Required Tools:	Espresso
Success Criteria:	The test-target behaves as expected and the non test-target software also behaves as expected.
Special Considerations:	

4. Deliverables

4.1 Test Evaluation Summaries

The Test Evaluation Summary collects, organizes, and presents the Test Results and key measures of test to enable objective quality evaluation and assessment. The Test Evaluation Summary also presents an interim evaluation from the test, indicating the assessment of the software. The test evaluation summaries are realized in form of an internet survey.
Following documents will be submitted after finishing the testing phase

- Espresso
- Test plan
- Develop Test cases
- Test Case Review
- Complete Defect Reports

4.2 Reporting on Test Coverage

Incident log entries will be made for all bugs found during testing. The log will be used to track the status of the bugs.

5. Risks, Dependencies, Assumptions, and Constraints

Risks, Dependencies, Assumptions, and Constraints Risk	Mitigation Strategy	Contingency (Risk is realized)
All the functionalities may not be able to be checked by the automated testing.		Refractor and re-architect the design in order to enable easy mocking of components where possible in order to automate testing.
It is a web application and is tested for some of the platforms only, thus there is a possibility for the system not working on some of them.		Check application on many platforms as much as possible. Review Test Plan and modify components
Database requires refresh.	System Admin will ensure the Database is regularly refreshed as required.	Restore data and restart Clear Database

6. References

[1] Espresso, https://developer.android.com/training/testing/espresso