NANYANG TECHNOLOGICAL UNIVERSITY

SINGAPORE

School of Computer Science and Engineering

CZ3002 Advanced Software Engineering

Test Plan (IEEE 829 Format)

Project Name: Hangout

Group Name: Mac & Cheese
Lab Group: TDDP1

Group Members
Jethro Phuah An Ping (Team Leader)
Lam Zhi Fah (Deputy Team Leader)
Alicia Chua Jieying
Ernest Ang Cheng Han
Ong Jing Heng Shaun
Ong Sim Hao
Shannon Tan Xinyi

Table of Contents

1. Test Plan Identifier

Master Test plan for HangOut App 1.0.

2. Introduction

This Master Test plan describes how the testing is planned and managed across different test levels. It provides a bird's eye view of the key decisions taken, strategies to be implemented and the testing effort required for our project. Lastly, it ensures software testing is done in a defined process which is organised and easy to follow.

Through testing, we hope to deliver a complete and excellent user experience. This document will mainly contain:

1. Identified features of our App to be tested
2. Features not to be tested, along with their reasons why
3. Tools used to test features
4. Criteria to check if tests have passed
5. Breakdown of testing activities and responsibilities

More details on our approach, constraints and deliverables will also be provided.

3. Test Items

We will be testing the following:

- User Mobile Application, HangOut, Version 1.0
- Admin Web Application, HangOut Admin, Version 1.0.

4. Features to be Tested

The following is a list of the areas to be focused on during testing of the **mobile application**:

No.	Feature	Level of Risk
1	Interface to view food locations, places of interests, upcoming events and meetups according to selected location	High
2	Creating and joining of meetups	High
3	Favorite food locations and places of interests	Medium

4	Creation of account and login	Medium
5	Interface to edit profile details	Low

The following is a list of the areas to be focused on during testing of the **admin web application**:

No.	Feature	Level of Risk
1	Interface to add/delete/edit food locations, places of interests, upcoming events and meetups	High
2	Creation of account and login	Medium

5. Features not to be Tested

The following is a list of the areas that will not be specifically addressed during testing of the application:

No.	Feature	Level of Risk
1	Authentication of account during login	Low

The above feature will not be tested as it has relatively low risk. We will be using pre-existing libraries for this feature, which is considered stable and does not require additional testing.

We will be using Firebase Authentication, which is a pre-defined class by Firebase. It provides backend services and ready-made UI libraries to authenticate users. It supports a wide range of authentication, including using passwords, phone numbers, Google and Facebook.

6. Approach (Strategy)

The primary model used in this project is the Agile model. The characteristics of the model will be adhered to by this test plan.

6.1. Types of Testing

Our teams' testing utilizes two types of testing for our user mobile app and admin web application:

1. Manual Testing
2. Automated Flutter Testing

6.2. Levels of Testing

Our team applied four levels of testing namely, unit testing, integration testing, system testing and user acceptance testing for Hangout.

6.2.1. Unit Testing

Unit Testing will be performed using the White Box and Black Box techniques. For White Box Testing, test cases will be used to test each individual unit of HangOut. For Black Box Testing, participants who do not have any prior knowledge about our application will be brought in to test the individual unit. It is crucial that every individual component in the source code is performing well before we proceed to the next testing phase of Integration testing.

6.2.2. Integration Testing

As both our applications are built around multiple components, it is essential that each component remains functional after they are integrated into one another. This test focuses on combining tested units into groups and exposes faults between integrated units. The purpose of integration testing is to expose defects in the interactions between different software modules. Integration testing will be done automatically using the Flutter test package. Integration testing will be written and conducted by Quality Assurance (QA) manager and QA engineer.

6.2.3. System Testing

A diverse group of users (Namely our project group members and friends) will be brought in to do a final test of the app. As our application serves a wide range of users, it is essential that people from different environments are involved in this final stage of testing. This is to check if existing features are working well and any change does not introduce new bugs. The feedback obtained will then be documented for future application improvement, and bugs found will be resolved before release.

6.2.4. User Acceptance Testing

User acceptance testing (UAT) is the last phase of software testing before release of the application. The purpose of UAT is to verify the entire business flow regarding usage scenarios and identify problems that testers might have missed in the previous phases of testing. We need end users to perform UAT for preparing the software for the production environment. In our case, end users who perform UAT will be the whole team and volunteers including members of the public and GRC members. The feedback obtained will then be documented for future application improvement, and bugs found will be resolved before release.

6.3. Tools

The table below shows the tools used in testing:

Tool Name	Description	Special Training Required
Microsoft Word	For writing problem reports and other necessary documents	No
Microsoft Excel	For documenting data in test metrics	No
An IDE (preferably VSCode for web application and VSCode/Android Studio for phone application)	An source code editor is required to write code	No
Dart	The Development language to write and run unit tests and integration tests for our Hangout mobile application	No
HTML/Javascript	The Development language to write and run unit tests and integration tests for our Hangout web application	No
Firebase	Database used for the application	No

6.4. Metrics in Testing

The table below shows the testing metrics required to be collected:

Metrics
Number of white box testing done per functional requirements
Number of black box testing done per functional requirements

Number of test cases written per functional requirements
Total number of test cases written for all requirements
Total number of test cases executed
Total number of test cases passed
Total number of test cases failed
Total number of defects identified
Total number of defects fixed
Total number of defects accepted

6.5. Software & Hardware

The software to be tested is the HangOut phone application and our HangOut Admin Web application. There is no hardware to be tested.

6.6. Action for Untestable Elements

Elements that are untestable will be recorded in the report by the QA engineer. The items will be further discussed and handled in the next day's meeting with the QA team, development team and project manager.

7. Item Pass/Fail Criteria

At the unit test level, item would be considered a pass if:

1. All test cases completed
2. 90% of cases completed with less than 80% containing some number of minor defects
3. Code coverage tool indicates all code covered

At the master test plan level, item would be considered a pass if::

1. All lower-level plans completed
2. 90% of plans completed without errors and less than 80% with minor defects

7.1. Mobile Application HangOut:

Feature Tested	Pass Criteria	Fail Criteria
Register	User successfully signs up and an account is created	User signs up fail, no account created
Login	User can successfully login to HangOut	User cannot login to HangOut
Create Meetup	User can successfully create a meetup in the database	User cannot create a meetup
Join meetup	User successfully join meetup	User cannot join meetup OR User joins the wrong meetup
View / Update Profile	User can successfully view / edit account	User cannot view / edit account
Select desired location	User can successfully select their desired location from a dropdown list of location/GPS	User is unable to select the desired location
View Food/Events/Places of Interest	User can successfully view all Food/Events/Places of Interest from the database	User is stuck at home page
Favourite Food/Events/Places of Interest	User can successfully favourite desired food/events/places of interest and view them under the favourite tab	User is unable to favourite and unable to view them at favorited tab

7.2. Admin Web Application:

Feature Tested	Pass Criteria	Fail Criteria

Register	Admin successfully signs up and an account is created	Admin sign up fails, no account created
Login	Admin can successfully login	Admin cannot login
Add/Edit/Delete Food/Event/Places of interest into the database	Admin can successfully add/delete/edit food/events/places of interest in the database	User cannot add/delete/edit food/events/places of interest

8. Suspension Criteria and Resumption Requirements

The team has agreed on 2 scenarios where testing may be temporarily halted and resumed after rework:

1. Number of failed tests exceeds more than 80% of total test cases
2. Any test done on functional requirements fails

In these 2 scenarios, testing will stop temporarily. The App will be handed over to the development team for them to perform corrective actions. Only after the development team is done will testing resume.

While testing is important, the team recognises that if the number of defects reaches a certain threshold, there will be no value in continuing testing. Ultimately, with a high defect rate, the App will still have to be handed over to the development team for rework. Resources, such as time and budget, will be wasted if testing is continued.

9. Test Deliverables

The test deliverables include:

1. Master test plan document
2. Test cases
3. Tools and results/outputs
4. Test design specifications
5. Problem reports and corrective actions

The App itself is not a test deliverable. It is listed under “Test Items” in section 3 of this document and will be delivered by the development team.

10. Remaining Test Tasks

Task	Assigned To
Generate test plan	QA engineer
Prepare functional requirements for the QA team	Development team
Generate test data required	QA engineer
Set up test environment	Development team
Program test scripts	QA manager, QA engineer
Document test results and reports	QA engineer
Document problem report	QA engineer
Create UAT plan	Project manager, QA manager

11. Environmental Needs

11.1. White Box Test Stage

The White Box Test Stage will be conducted by the development team. They will make use of the 8 personal computers allocated to them to conduct the White Box Testing.

11.2. Black Box Test Stage

The Black Box Test Stage will be conducted on 20 participants who have no prior experience with the web and phone application. They will be tasked to evaluate it based on product quality, user friendliness and bugs that occur during the testing process. 7 of the participants will be using the personal computers allocated to the development team to reduce the need for additional computers and allow for more personalized feedback.

11.3. Automated Testing

Test data is created automatically by our test dart script. No outside user activity is required.

11.4. Final Test Stage

The Final Test Stage will be conducted by the development team. Based on the feedback provided by the 20 participants from the Black Box Test stage, the development team will perform the necessary changes and testing required and contact the participants if the need arises.

12. Staffing and Training needs

The development team is required to be very familiar with every single feature within the application. This is to ensure that the development team can review and validate the application fairly.

During the Black Box Test Stage, the development team is required to respond to any queries that the participants might have. 20 separate user accounts will be prepared before the testing.

Automated Testing should be done by a QA manager and QA engineer. The QA manager should provide training to the QA engineer if the engineer does not have strong technical skills to write test cases or documentation skills to write reports.

13. Responsibilities

Role	Responsibilities	Person in charge
Project manager	● Selecting features to be tested ● Review and approve Test Plan, Schedule and approaches ● Review and validate test results ● Ensure deadline is met for each task assigned ● Ensure project development is on track and arrange for discussion ● Provide for resolution of scheduling conflicts ● Make the critical go/no go decisions for items not covered in the test plan with	Jethro Phuah

	QA manager	
Development team	● Fix bugs raised by QA team ● Conduct regular review of product requirements regularly with client ● Update progress of application development and existing problems ● Provide support to QA engineer regarding test scripts writing ● Review and validate test results ● Review test approaches ● Review test progres with Quality Assurance Team	Ernest Ang Cheng Han, Ong Jing Heng Shaun, Lam Zhi Fah
QA manager	● Facilitate the testing activities ● Give approval for the next level of testing ● Communicate any risk in the testing to the rest of the team ● Set overall strategy for this level of plan ● Provide required training to QA engineer for writing test and documenting test metrics ● Make the critical go/no go decisions for items not covered in the test plan with project manager	Ong Sim Hao

QA engineer	● Implement test cases ● Ensure all required elements are in place for testing. ● Document test metrics	Shannon Tan Xinyi, Alicia Chua

14. Schedule

In the project plan, time has been allocated to conduct testing activities. The allocated time for testing can be found in the Project Proposal. In total, 10 working days have been allocated. Testing will commence on 1 October 2021.

The people in charge of the testing process can be found in the Responsibilities section directly before this section. Coordination of the different personnel required for this task will be managed by the project manager.

The testing schedule can be broken down into a few key milestones listed below. The recommended amount of time the Quality Assurance Manager should spend on each milestone has also been included in square brackets:

1. Review of Use Cases document by Quality Assurance manager and SCRUM Master [0.5 day]
2. Development of Master test plan by Quality Assurance manager [1.5 day]
3. Review of the System Requirement Specifications with Quality Assurance manager with Project Manager and SCRUM Master. This will provide a clearer understanding of the App structure [0.5 day]
4. Review and amendment of Master test plan by Quality Assurance Manager
5. Review and approval of Master test plan by Project Manager and SCRUM Master [0.5 day]
6. Unit, system and integration testings performed by Quality Assurance Manager [3 days]
7. Correction of issues found performed by backend and frontend developers [3 days]
8. Final testing performed by Quality Assurance Manager [1 day]

In the event that the App is unable to be completed in time for the full round of testing, 2 options will be provided:

1. If the delay is no more than 2 days, the Quality Manager will stick to the original testing timeline. He will be more tolerant as it is in the team's interest to get a better tested application
2. If the delay is more than 2 days, he will discuss with the Project

Manager and Project Sponsor to agree on a few allowable defects for the launch. These defects will be for smaller and unimportant features and will be rectified in future versions/releases of the App

The Quality Assurance Manager will be keeping a close eye on the progress of the App and ensures any changes to the testing schedule and scope be discussed as early and as advanced as possible.

15. Planning Risks and Contingencies

The table below states the risks as well as mitigation plan for each of them:

Risk	Mitigation Plan
Lack of availability of required test data	Developers and QA engineers should keep the latest copy of the test data on GitHub
Late delivery of the application from development team due to incomplete features	Project manager to see if completed parts of the app can be tested first. Else, re-planning of the testing schedule may be required
Delays in training on the application	Developers to provide necessary software testing documentations for testers to start
Changes to the original requirements or designs	Development team will have to make the necessary changes before handing the app ober. Number of participants for the Black Box Test Stage will be reduced

16. Approvals

Name	Role	Signature & Date
Jethro Phuah	Project Manager	JPHUAH / 17 September 2021
Ong Sim Hao	Quality Assurance Manager	Hao / 17 September 2021
Alicia Chua	SCRUM Master	Aliciacjy / 17 September 2021

Shannon Tan	UI/UX Designer	ShannTan / 17 September 2021

17. Glossary

Term	Definition
Authentication	The process of determining whether someone or something is, in fact, who or what it is declared to be.
Functional Requirements	Functional requirements specify Business Product features and what the Business Product must do. They are directly derived from the objectives defined in the Project Management Plan. A functional requirement is a tangible service, or function, that the Business Product must provide and is a non-technical requirement. how the Business Product should behave. See also Non-functional Requirements.
Constraint	A constraint is an applicable restriction or limitation, either internal or external to the project that will affect the performance of the project.
Defect	An imperfection or deficiency in a project component where that component does not meet its requirements or specifications and needs to be either repaired or replaced.
Quality Assurance	The process of evaluating overall project performance on a regular basis to provide confidence that the project will satisfy the relevant quality standards.
Threshold	The level at which an event or change occurs.
Use Case	A description of system behavior, in terms of sequences of actions. A use case should yield an observable result of value. A use case should contain all alternative flows of events related to producing the intended observable value.
User	The person or organization that will use the project's product.

18. References

Name of Document	Link to Document
Project Proposal	https://docs.google.com/document/d/1EoYewf-0lJAJCXPDDcKRO3dBOc9HzpBd/edit?usp=sharing&ouid=106342154695843875731&rtpof=true&sd=true
Use Case Diagram	https://docs.google.com/document/d/1GQjZ6esG6CPVBHDPkkg9vFesW4zFAspg/edit?usp=sharing&ouid=106342154695843875731&rtpof=true&sd=true
System Requirement Specifications	https://docs.google.com/document/d/1ljNHc51czofe8xC9qBpVJftVS5RNvcCW/edit?usp=sharing&ouid=106342154695843875731&rtpof=true&sd=true