Test Plan

NTUCollab: A Collaboratory Platform for NTU Students

Dandapath Soham – Project Manager, Back-end Developer

Gupta Jay – Lead Developer, Release Engineer

Kanodia Ritwik – Front-end Developer, Release Engineer

Mundhra Divyesh – Front-end Developer

Bhatia Ritik – Back-end Developer

Somani Palak – QA Manager, QA Engineer

Bansal Aditya – QA Manager, QA Engineer

Team Eagles

School of Computer Science & Engineering

Nanyang Technological University, Singapore

Submitted to—

Dr Shen Zhiqi

School of Computer Science & Engineering

Nanyang Technological University, Singapore

Version History

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	Somani Palak	02/04/2021	Bansal Aditya	03/04/2021	Initial Plan and template
1.1	Bansal Aditya	04/04/2021	Dandapath Soham	06/04/2021	Add Test Items, Software Risk Issues & features to be tested
1.2	Somani Palak	05/04/2021	Bansal Aditya	06/04/2021	Add approach, responsibilities & contingencies
1.3	Bansal Aditya	15/04/2021	Dandapath Soham	16/04/2021	Final proof-read and formatting

Table of Contents

1. Test Plan Identifier – TP1.3

The test plan for NTUCollab contains the scope, approach, and schedule of the testing activities to be undertaken for NTUCollab throughout its lifecycle as well as during the maintenance phase. This is first major version of the document with 3 minor revisions. The test plan is at level 2, as all the fundamental testing criterion and cases have been documented and the basic functionality has been implemented and tested in the software application. The plan will gradually transition to the master plan as the testing becomes more robust, and comprehensive.

The plan contains information about the resources and procedures of testing of NTUCollab, and how the testing process is controlled, and configuration managed throughout the product lifecycle.

2. Introduction

The Test Plan provides a comprehensive overview of the scope, approach, resources, and schedule of all testing processes and activities to be performed for NTUCollab. The plan identifies the items to be tested, the features to be tested, the types of testing to be performed, the personnel responsible for testing, the resources and schedule required to complete testing, and the risks associated with the plan.

Four types of tests will be carried out, namely, Unit Testing, Integration Testing, System Testing and User Acceptance Testing. More specifically, we will be using a black-box approach.

i. **Unit Testing:** Unit testing will be performed to test the individual units or components of NTUCollab in isolation. This will validate each component of the application. Unit testing will be performed during the development phase.

ii. **Integration Testing:** The individual tests will now be tested together to test the combined functionality of the application.

iii. **System Test:** Quality Assurance will perform independent testing to test the entire system as a whole and provide feedback to the development team.

iv. **User Acceptance Test:** Selected end users will be asked to use the application and provide their feedback which forms the user acceptance test results.

3. Test Items (Functions)

The following functional test items are identified for various testing techniques:

3.1. Unit Testing

3.1.1. Bottom Navigation - There are three buttons at the bottom navigation of NTUCollab: Modules, Clubs, and Groups. The user should be redirected to the respective pages when the buttons are tapped.

3.1.2. Recommend Slide - For the three main pages of NTUCollab: Recommended Modules/Clubs/Groups, a slider is shown for the recommendations. The user must be able to swipe left and right in the slider to see all the recommended options.

3.1.3. Selection of Tags - When the user logs in to the application for the first time, NTUCollab must prompt them to select the suitable tags for modules, clubs, and interest groups.

3.1.4. Rating Buttons - When the user taps on any type of group card, a detail page is displayed where they can see more information about the group. In the detail page, they application must allow to rate the group out of 5 in all three categories which differ by the type of group selected.

3.1.5. Data Mapping - When the user taps on any type of group card, a detail page is displayed where they can see more information about the group. The detail page is populated by the information supplied from the home page.

3.2. Integration Testing

3.2.1. Authentication - The user must be able to login into the system using their credentials upon verification by Firebase Auth.

3.2.2. Tag Selection - The user must be able to login into the system and select the tags for the three main components.

3.2.3. Recommendations - The user must be able to login into the system, select tags, and see the recommendations for the three respective components.

3.2.4. Groups - The user must be able to login into the system, select the tags for the three main components, get recommendations and see description, rate, and make posts on

the recommended groups. The user must also be able to able to see other groups which were not recommended.

3.2.5. Create interest groups - The user must be able to login into the system, select the tags for the three main components, get recommendations. The user must be able to create interest groups

3.3. System Testing

3.3.1. Smoke testing - The user must be able to login to the application and be able to navigate across and use all the features designed.

3.3.2. Installation Testing - The user must be able to install application on their mobile device.

3.3.3. Stress Testing - Multiple users must be able to use the application simultaneously

3.3.4. Scalability Testing using Database - Database should be able to process queries from multiple application without crashing.

3.3.5. Recoverability Testing - The data stored for any user must be stored in the database if the app crashes or the device loses internet connectivity

3.3.6. Security Testing - Any user must not be able to login in an unauthorized manner or manipulate NTUCollab' s database

3.4. User Acceptance Testing

For the User Acceptance Testing phase, a beta version of the app will be released to a selected beta tester and their feedback will be used for launched and the users will be asked to give their feedback.

4. Software Risk Issues

Some of the potential software risks which could be faced are:

4.1. Integration with STARS system and potential changes might impact certain functionality of NTUCollab
4.2. Ability to use and understand a new package/tool, etc.
4.3. Maintenance of certain complex functions
4.4. Modifications to components with a past history of failure
4.5. Poorly documented modules or change requests

Unit testing will help identify potential areas within the software that are risky. If the unit testing discovered a large number of defects or a tendency towards defects in a particular area of the software, which is an indication of potential future problems. An approach that will be taken in to resolve such issues will be to define where the risks are by having several brainstorming sessions.

5. Features to be Tested

The features to be tested will be given a risk rating, which is one of three levels: High, Medium, Low. High level risks have the highest importance and must be tested and debugged as soon as possible. Medium level features are to be tested only after high level bugs. Low level features are not as important and will only be tested once all other features are working properly

Item	Risk
Authentication	High
Tag Selection	Medium
Recommendations	High
Groups Formation	Medium
Interest groups	Medium

6. Features not to be Tested

All the features implemented in the application will be tested, since all the features are high priority necessary to run the application.

7. Approach

A test approach specifies how testing would be performed i.e. It is the test strategy implementation of a project. A Test approach has two techniques:

Reactive - An approach in which design and coding are completed first before testing.

Proactive - An approach in which the test design process is initiated as early as possible with an aim to find and fix the defects before the build is created.

7.1. Testing Approaches

Apart from the above stated two techniques, a test approach based on different context and perspective may be categorized into following types:

7.1.1. Methodological Approach

This approach consists of all those methods which are pre-determined and pre-defined, to carry out the testing activity. The methods covered under this approach are used to test a software product from each different perspective and requirements, ranging from static analysis of the programming code to dynamic testing of the application.

7.1.2. Analytical Approach

Analytical approaches involve, selecting and defining the approaches based on the analysis of some factors or conditions associated with the software product, which may produce some significant changes to the testing environment, such as on requirement basis, risk based, defect severity basis, defect priority basis, etc. For example, when defining and preparing the test approach based on risk, it may include approach/method to consider the area of higher risks, for the execution under the testing phase, whereas that of lower risk may be taken up at a later stage.

7.1.3. Regression Averse Approach

It includes preparation of approach based on the usage of existing test material and automation of regression test suites.

7.1.4. Model-based approach

Approach based on some specific model, build up on some sort of statistical or mathematical value associated with the software product entity or its functionality, such as rate of failures, etc. Generally, it may be seen as a preventive approach, which should

be initiated as soon as the model used in the SDLC is identified and selected, in the early stage of development life cycle.

7.1.5. Dynamic or Heuristic Approach

The approach involves heuristic testing types such as exploratory testing, where tests are designed, created, and executed as per the current scenario and is not pre-planned like in other approaches. Basically, it is a reactive test approach which carries out the simultaneous execution and evaluation of the test cases.

7.1.6. Standard Compliant Approach

As the name suggests, the approaches are based on some specific regulation, guidelines, or industry standards such as IEEE 829 standard. The testing activities covered under this approach like designing and creation of test cases follows and adheres to the given specified standards.

7.1.7. Consultative Approach

This approach is based on the recommendation and suggestion given by the experts or other professional from the business or the technical domain outside the boundary of the organization, which may include end users also.

7.2. Steps in Testing

A test plant approach can be summarized with six steps as illustrated in Fig – 1.

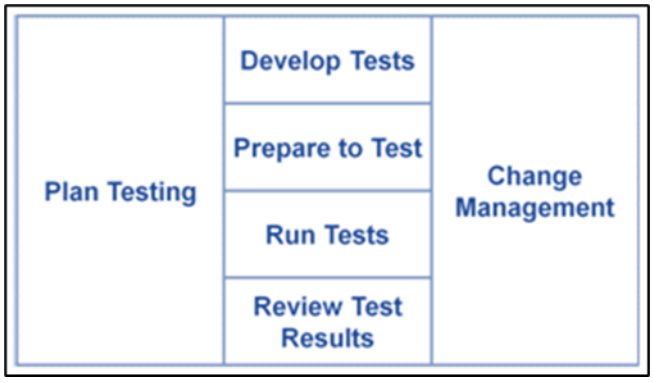

Fig – 1: Steps of the test plan

Below we briefly summarize the activities performed in each of these steps:

7.2.1. Plan Testing

This is where the timescale, scope, quality, risk, and resources are decided in advance, and kept up to date throughout the UAT, including Entry Criteria and Exit Criteria.

7.2.2. Develop Tests

This involves numerous activities shown Fig-2 in order to develop the formal tests required to run any form of testing:

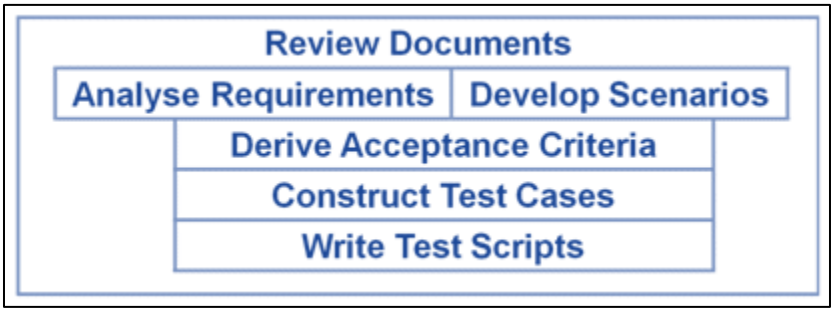

Fig – 2: Activities in the Develop Tests Step

- **Review Documents**

  This is a key quality process of checking all documentation produced during the development of the system.

- **Develop Scenarios**

  This involves the preparation of scenarios that use the system, using techniques such as Use Case.

- **Analyse Requirements**

  This involves understanding the formal tests and looking for what is inconsistent and missing from what is actually required.

- **Derive Acceptance Criteria**

  Once the previous two activities are underway or completed then the set of questions to be asked about the system to see if it matches the capability needed are prepared.

- **Construct Test Cases**

  Test Cases are the set of specific inputs and expected results which enable one or more Acceptance Criteria to be proved.

- **Write Test Scripts**

  Test scripts are the operational instructions for running Test Cases. It includes what has to be done to the system, what has to be measured, and how to do the measurement.

7.2.3. Prepare to Test

These are the activities other than developing the tests that are required to allow testing to take place:

7.2.3.1. **Preparing Test Data** - Building the data files that are required to run the test cases.

7.2.3.2. **Preparing the environment to run the tests** - Making sure that the people, processes, hardware, software etc. are all in place to enable the testing to take place.

7.2.3.3. **Creating three key documents:**

- **Test Procedure** - The instructions about how to run the tests on the test system including the Test Scripts.
- **Entry Criteria** - What must be done before testing starts.
- **Test Item Transmittal Report** - The instructions from the developers about the system version released for testing.

7.2.4. Run Tests

Run Tests comprises of running the tests and recording the results:

- Running the tests involves using the input and expected results from the Test Cases and applying the Test Scripts and other elements of the Test Procedure to run the tests.
- Recording the results involves recording inside the Test Log the order in which the activities were performed, and the events that occurred when the test was run. If any test has actual results that differ from the expected results have the information recorded in an Incident Report. The Incident Severity is also decided at this point.

7.2.5. Review the Results

The acceptability of the system is assessed once the tests have been performed. A simple method involves determining how many outstanding incidents were there and their corresponding severity. However, this is not sufficient as a mere count of incidents does not give much detail into their impact on what the organisation aims to achieve with the

system. A flawed system which delivers capability to an organisation is much better than a perfect system that does not. Therefore, the test results need to be checked and traced to see what effect they have on:

- Business or System Impact,
- Requirements and
- Scenarios

This analysis ensures that a balanced decision can be made as to whether the system passes these particular tests and also allows to make effective recommendations about its use. The results of all this activity are then recorded in a Test Summary Report.

7.2.6. Change Management

This process involves making an impact analysis for any changes for their effect on the system. As the stage of starting testing, or even during it gets closer, changes can have a major influence on how effective the testing will be.

7.3. Approach for NTUCollab

For our project we follow a methodological approach. Tests will be carried out as per the documented test cases stored in the Test Summary Report. The Quality Assurance engineers will be responsible to create test runs. The tester will execute the tests in the Test Summary Report and mark each case as Pass/Fail. The tester should record down the actual results on the Test Summary Report. When tests are marked as Fail, developers that are responsible of the corresponding feature should look into the errors and correct the system.

The Quality Assurance Engineers will review the test runs in the Test Summary Report and review alongside the Quality Assurance Manager accordingly. The test cases will be presented to the Quality Assurance Manager before they are finally marked as passed. When testing is deemed to be complete by the QA Manager, a test report will be submitted to the project manager who will mark the testing as complete if the application works as, it should.

Following are some of the factors that were considered when selecting the test approach:

- The nature of the product and the domain.
- Risks of product or risk of failure of the environment and the team.
- Regulatory and legal aspects, such as external and internal regulations of the development process.
- Experience and expertise and of the people in the proposed tools and techniques.

8. Item Pass/Fail Criteria

This section deals with defining when an item has passed or failed a test in this project.

In our project, for each test case, the tester is equipped with the proper input and the right output. The test case will only be deemed to have passed if the result matches exactly with the ideal output defined in the test case. If the output is not exactly the same, the test case will be deemed to have failed. In case it is not clear whether the output matches the ideal result, the Quality Assurance Manager will be contacted who might consult the Project Manager incase necessary.

The completion criteria for this plan are:

- 100% of unit testing cases are complete
- 90% of integration testing cases are complete and 10% cases or lesser have minor defects
- A minimum of 250 users have given their feedback for User Acceptance Testing
- System Testing is done with at least 10 external testers9.

9. Suspension Criteria and Resumption Requirements

9.1. Suspension Criteria

Suspension in the software testing process in which the testing team will suspend the testing activities based on some criteria. The test team decides whether to suspend the complete or the part of the software testing process. Suspension can occur when the external components are not readily available or when a serious defect is detected. Suspension is also known as Test-Stop criteria for the testing process.

Furthermore, testing processes are suspended mainly when the actual result is not the same as that of the expected result. Technically speaking, Suspension occurs when the output of the parallel program is not identical to the output of the sequential program. Fig-3 depicts when the testing process might be suspended. The testing team needs to stop the testing if a defect is detected at a point after which the testing is not valid for the test-case.

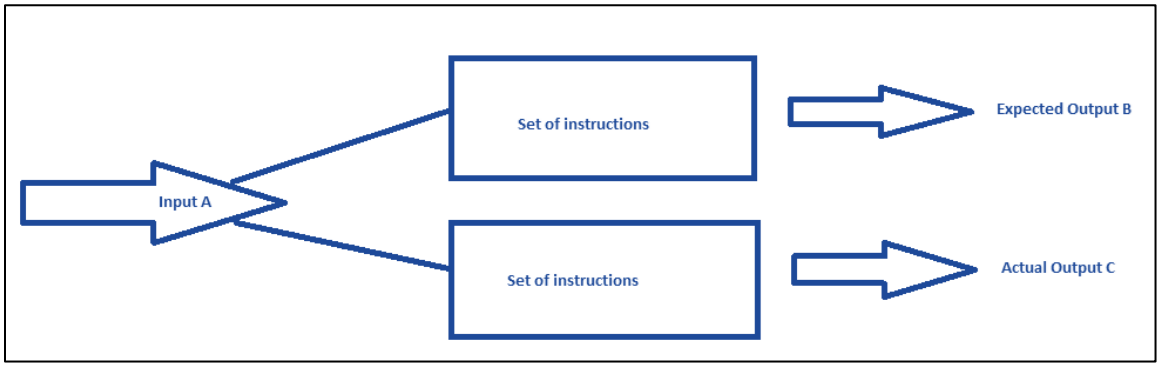

Fig – 3: Suspension of the testing process

In our project in case the testing team is suspending any testing process a proper reason and the documentation which defines the acceptable level of the defect with is provided. There may be various scenarios to suspend the testing process. For e.g., network, hardware, or the defect in the code.

Below are a few common criteria that lead to the suspension in the software testing process:

- Hardware or software not available.
- Any serious defect which highly impacts the testing progress.
- The testing process can be limited by the defects in the build.
- Any problem related to connectivity.
- Test resources are not available when needed.
- Schedule of the project (Giving priority to deliverables).
- The output is not the same as expected.
- Functionality does not work as specified in SRS (Software Requirement Specification).

In our project, If the team members report that there are 40% of test cases failed, suspend testing until the development team fixes all the failed case. The Lead Developer will be informed about the test cases which have failed.

9.2.**Resumption**

Resumption is restarting or resuming the process which is invoked after the suspension criteria are met. As the name suggests resumption is the contrary process of suspension. It involves verification of the defect by which suspension was invoked during the testing process.

Below are a few common criteria to resume the testing process:

- Hardware or software resources are available as per the requirements.
- Issue due to which suspension occurs gets resolved.
- No further defect has been found in the resumption technique.
- Output meets what is being expected.

To summarize, the suspension criteria specify the criteria to be used to suspend all or a portion of the testing activities while the resumption criteria specifies when testing can resume after it has been suspended.

10. Test Deliverables

Test Deliverables are the test artifacts which are given to the stakeholders of a software project during the SDLC (Software Development Life Cycle). Some of the deliverables are provided before the testing phase commences and some are provided during the testing phase and rest after the testing phase is completed.

The deliverables included in this Test Plan are:

10.1. Test Cases

Test cases are the set of positive and negative executable steps of a test scenario which has a set of pre-conditions, test data, expected result, post-conditions, and actual results.

10.2. Test report

This contains the test results and the summary of test execution activities.

10.3. Test Plan Document

Test plan document is a document which contains the plan for all the testing activities to be done to deliver a quality product. The test Plan document is derived from the Product Description, SRS, or Use Case documents for all future activities of the project. It is usually prepared by the Test Lead or Test Manager.

10.4. Revision Logs

The revision history of all the documents will be duly documented and stored.

10.5. Defect logs with solutions implemented

All the defects identified will be documented and logged in the appropriate documents as well as the solution implemented for the respective defect.

11. Test Tasks

The following activities must be completed:

i. Test plan prepared
ii. Items to be tested are identified
iii. Identify method of conducting tests
iv. Personnel assigned to each test case
v. Conduct testing
vi. Fix bugs and errors which are discovered
vii. Create defect logs
viii. Create test report

12. Environmental Needs

For the testing of NTUCollab, following specifications and requirements have to be met:

- Linux, Mac OS X, or Windows.
- git (used for source version control).
- An IDE. Android Studio with the Flutter plugin is our flagship IDE. Any appropriate IDE can be used.
- An ssh client (used to authenticate with GitHub).
- Python (used by some of our tools).
- The Android platform tools.

Further, the testers need to have access to NTU STARS to test the STARS related tests of the application.

13. Staffing and Training Needs

Training will be required to setup the application on the testers' machines as well as to get acquainted with the Flutter testing frameworks.

The Lead Developer will be responsible for providing training on how to run the application locally and the structure of the code. The QA Engineers can consult the front-end and back-end developers if they run into any problems. The QA Manager will be responsible for finding methods to train the testers on the Flutter testing frameworks.

14. Responsibilities

Role	Responsibilities
DEV Lead	Providing required training in code details. Providing solutions for running the code Conducting white-box testing on the entire system
Project Manager	Deciding which features should be tested Monitoring the errors found Collecting and analyzing final testing report
Release Manager	Collecting details of test runs to make sure software meets requirements before deployment
DEV Front-End	Conducting white-box testing on the front-end components
DEV Back-End	Conducting white-box testing on the back-end components
QA Engineer	Conducting black-box testing Reporting test logs to the QA manager Generating additional test cases as needed
QA Manager	Stating the risk and contingency plan for the different phases of the test. Monitoring testing activities and ensuring all testing resources are available Setting overall testing strategy

15. Schedule

The testing phase for NTUCollab will last for up to 5 weeks, the details of which are given in the Project Plan. Our planned activities will be as follows:

15.1.1. A daily backlog will be maintained to keep track of items which are yet to be completed. The Quality Assurance Manager will continuously monitor this backlog and will help the Quality Assurance Engineers if the backlog gets too large.

15.1.2. A one-week buffer is put in the plan in case there are not enough users for User Acceptance Testing

15.1.3. Number of days assigned to each test case will depend on their risk level. High risk will get 6 days, medium risk will get 5 days and low risk will get 4 days. This is to ensure that too much time is not wasted on testing low risk items.

15.1.4. For any bugs that are discovered, the result will be immediately updated on the GitHub Projects Board and the Lead Developer will be notified of the same immediately.

16. Risks and Contingencies

The risks and methods to mitigate them are as follows:

Risk	Contingency
Shortage of testers	Members of the development team as well as management will pitch in to help with the testing if the QA team gets overwhelmed
Improper training for testers	The QA Manager and Lead Developer will be responsible to provide all the required training
Improper communication amongst testers and developers	The daily testing backlog will ensure testing is on track. Testers will be required to report any bugs they discover immediately to the Lead Developer

17. Approvals

Role	Test Type	Approval Criteria
Release Manager	Black Box	User Acceptance Testing meets acceptance criteria
Project Manager	Overall	The application works as required and user acceptance testing meets client needs
Lead Developer	White Box	System performs in accordance with functional requirements
QA Manager	Black Box	All test cases are covered

18. References

Document	Link
Project Plan	https://172.21.149.196/svn/3002/B2/Eagles/Lab-3/Documents/Project_Plan.pdf
Requirements specifications	https://172.21.149.196/svn/3002/B2/Eagles/Lab-2/System_Requirement_Specifications.pdf
High Level design document	https://172.21.149.196/svn/3002/B2/Eagles/Lab-1/System_Architecture_Diagram.png
IEEE Test Plan Standard	https://ntulearn.ntu.edu.sg/bbcswebdav/pid-2348857-dt-content-rid-16753586_1/courses/20S2-CZ3002-LEC/Content%281%29/Labs%20IEEE%20829%20Test%20Plan%20Template/ieee829Handout.pdf
Flutter Environment Setup Guide	https://github.com/flutter/flutter/wiki/Setting-up-the-Framework-development-environment