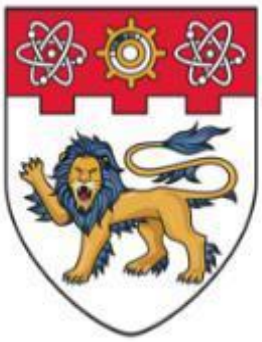

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Test Plan for Planit

Lab group : TS4

Team Members	Role
Ryan Tan Jinn En	Project Manager
Neo Yong Tai	Development Lead
Lee Yu Sheng Daniel	Back-End Developer
Mamuduri Paulani	Front-End Developer
Frankie Ye Htet Myat	Release Engineer / Manager
Chen Xueyao	QA Manager
Kundu Koushani	QA Engineer

Name of Design Team: **Team Syan**
School of Computer Science and Engineering, Nanyang Technological University

Version History

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	Kundu Koushani, Chen Xueyao	30/10/2021	Ryan Tan Jinn En	30/10/2021	Initial Plan and template
1.1	Kundu Koushani, Chen Xueyao	30/10/2021	Ryan Tan Jinn En	30/10/2021	Final proof-read and formatting

Table of Contents

1. Test Plan Identifier – TP1.1

The test plan for Planit contains the scope, approach, and schedule of the testing activities to be undertaken for Planit throughout its lifecycle as well as during the maintenance phase. This is the first major version of the document with 1 minor revision. The test plan is at level 2, as all the fundamental testing criteria 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 Planit, and how the testing process is controlled, and configuration managed throughout the product life cycle.

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 Planit. 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 Planit 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. CRUD Operations for Plants - CRUD operations are available for every plant created by the user. The users should be able to carry out the respective create, read, update (modify) and delete functions to the plants.

3.1.2. Tasks - Users can create daily tasks that can be checked, modified and deleted. Users should be able to carry out these operations on the tasks.

3.2. Integration Testing

3.2.1. Authentication - The user must be able to login into the system using their credentials upon verification against the data stored in MongoDB.

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

3.3.3. Scalability Testing using Database - Database should be able to process queries from multiple users (e.g. concurrent login by 3 different users) without crashing.

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

3.3.5. Security Testing - Any user must not be able to login in an unauthorized manner or manipulate Planit'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. Modifications to the product will be made in response to the feedback given to optimize Planit prior to public release.

4. Software Risk Issues

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

4.1. Ability to use and understand a new package/tool, etc.

4.2. Maintenance of certain complex functions

4.3. Modifications to components with a past history of failure

4.4. Poorly documented modules or change requests

Unit testing will help identify potential areas within the software that are risky or prone to errors. Through unit testing, defects or tendency towards defects in components of the software can be detected quickly. Brainstorming and software review sessions can be conducted afterwards to identify the cause and location of the risks.

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
CRUD operations on Plants	High
CRUD operations on tasks	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

Test approaches can be classified under the following types:

Reactive - An approach in which design and coding are completed prior to 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:

- Methodological approach based on failures
- Dynamic and heuristic approaches
- Consultative approaches
- Model-based approach that uses statistical information about failure rates
- Approaches based on risk-based testing where the entire development takes place based on the risk
- Standard-compliant approach specified by industry-specific standards

The following are some factors that affect a team's choice in testing approaches:

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

Planit has chosen to adopt a **reactive** and **methodological** approach, whereby testing comes after design and coding. The 6-stage testing approach is as follows:

1	Identify Test Areas	Through analysing the Planit Software Requirements Specification document, identify the functional and non-functional requirements that require testing.
2	Plan Tests	The QA team prepares a test plan documentation. Factors such as scope of testing, risks involved in testing, time and effort spent for testing, testing timeline and testing environment are finalized. Tasks are then assigned to individuals.
3	Design and Develop Test Cases	QA engineers will design and develop test cases according to the following guidelines: 1. Test cases should cover at least 90% of all functionalities. 2. All high and medium priority functionalities must be fully tested.

		3. All applicable permutations and combinations should be gathered. 4. Tests should be conducted under various predefined test conditions, with predefined inputs and expected outcomes. 5. The outcomes of all tests should be well-documented. 6. All specified requirements in the documentation stage should be verified and validated. The test cases are documented in the Test Summary Report.
4	Setup Test Environment & Simulate Test Conditions	The QA team creates various test environments in preparation for testing. As a web application, some test environments applicable to Planit are as follows: 1. Testing on various browsers - Google Chrome, Microsoft Edge, Safari etc. 2. Testing on devices with varying WiFi signal strength
5	Test Execution	Testers will run the different test cases. The outcomes of each test will be documented. In the case that there is deviation from the expected outcome or no outcome is achieved, the test is marked as "Failed". The team will proceed to conduct debugging and software review. After bug fixes, the respective components will be tested again to ensure that the expected outcome is achieved.
6	Test Closure	The QA team verifies that all tests are completed. Factors such as quality, test coverage, timeline and cost are evaluated and documented together with the conclusion of the test. The team will come together to reflect on whether the testing process can be further improved for future tests.

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 input and expected outcome have been predefined. 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 there is any deviation in the output, 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 in case 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
- 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 testing process in full or only in part. 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.

Below are some criteria we have identified that may lead to a suspension in Planit's 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 addition, if the team members report that there are 40% or higher test cases failed, the testing is suspended until the development team fixes all the failed cases before testing continues. 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 Planit, following specifications and requirements have to be met:

- Linux, Mac OS X, or Windows.
- git (used for source version control).
- An IDE. Any appropriate IDE can be used.

13. Staffing and Training Needs

Training will be required to set up the application on the testers' machines as well as to get acquainted with the 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 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 Planit will last for up to 4 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

15.1.1. Number of days assigned to each test case will depend on their risk level. High risk will get 5 days, medium risk will get 4 days and low risk will get 3 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