Test Plan

RoomieLah: A platform to find your ideal college roommate

Arora Srishti – Project Manager, Back-end Developer

Pandey Pratyush Kumar – Lead Developer, Release Engineer

Iyer Rajagopal – Front-end Developer, Release Engineer

Surawar Sanath Sachin– Front-end Developer

Agarwal Gopal – Back-end Developer

Acharya Atul – QA Manager, QA Engineer

Tayal Aks – QA Manager, QA Engineer

Team Strawhats

School of Computer Science & Engineering

Nanyang Technological University, Singapore

Submitted to:

Shi Hanyu

School of Computer Science and Engineering

Nanyang Technological University

Version History

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	Tayal Aks	31/03/2022	Acharya Atul	01/04/2022	Created the template and added the Initial Plan
1.1	Tayal Aks	04/04/2022	Arora Srishti	05/04/2022	Added Test Items, Software Risk Issues and the features to be and not to be tested
1.2	Tayal Aks	08/04/2022	Acharya Atul	08/04/2022	Finished the Report and sent it for proof-reading
1.3	Acharya Atul	11/04/2022	Arora Srishti	11/04/2022	Final amended report

Table of Contents

1. Test Plan Identifier – TP1.3

The test plan for RoomieLah contains the scope, approach, and schedule of the testing activities to be undertaken for RoomieLah throughout its lifecycle as well as during the maintenance phase. This is the first major version of the document. 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 to test RoomieLah, and how the testing process is controlled and how the configuration is 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 RoomieLah. 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.

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

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

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

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

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 on the bottom navigation bar of RoomieLah: View/Edit Profile, View Recommendations and Chat. The user should be redirected to the respective pages when the buttons are tapped.

3.1.2. Recommendations Swipe - For the main page of RoomieLah: Recommendations, cards are shown for the recommendations. The user must be able to swipe left and right on the cards to indicate interest to see all the recommended options.

3.1.3. Setting up profile- When the user logs in to the application for the first time, RoomieLah must prompt them to set their personal details along with their preferences for their roommate.

3.1.4. Viewing others' profiles - When the user taps on any recommended user's card on the recommendations screen, a details page is displayed where they can see more information about the user, along with social media links (if the user provides any).

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. Preferences - The user must be able to login into the system and edit their roommate preferences.

3.2.3. Recommendations - The user must be able to login into the system, edit their roommate preferences, and see the recommendations for roommates.

3.2.4. Matches - The user must be able to login into the system, edit their roommate preferences, get recommendations and match with another user.

3.2.5. Chat - The user must be able to login into the system, edit their roommate preferences, get recommendations, match with other users and chat with all the users that they match with.

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 the 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 applications 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 RoomieLah's database

4. Software Risk Issues

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

4.1. Potential changes to university hall allocation systems might impact certain functionality of RoomieLah
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
Profile setup	Medium
Recommendation	High
Viewing profile details	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 an 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 an 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 professionals 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.

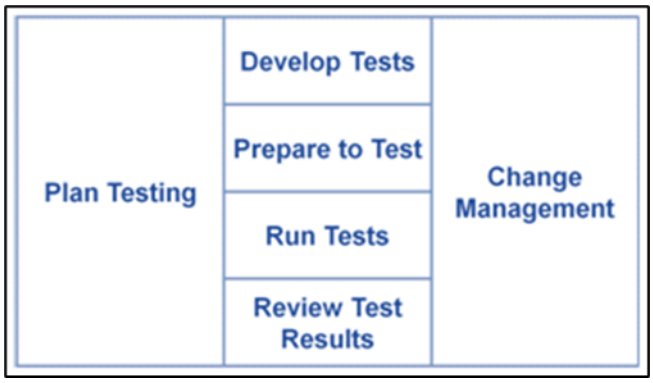

Fig – 1: Steps of the test plan

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:

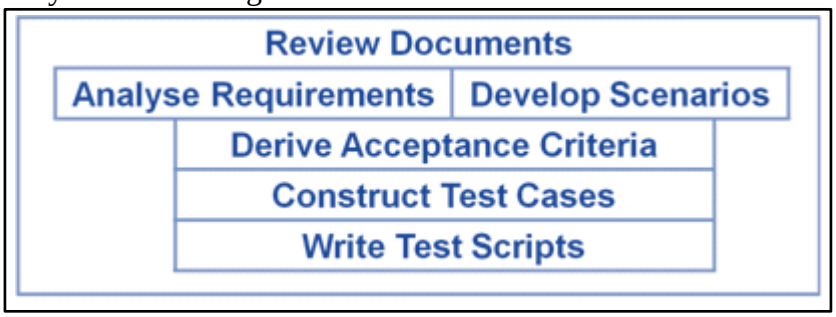

Fig – 2: Activities in the Develop Tests Step

- **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.

- **Analyze 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 apart from developing the tests that are needed for successful execution of the testing process:

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, tools 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 executing the tests and documenting the results:

- Executing the tests entails taking the input and the expected result from the Test Cases and applying the Test Scripts to execute the tests.
- Documenting the results entails noting down the order in which the tests were performed, and the outcome inside the Test Log. In case the obtained result is different from the expected result, it is immediately recorded in the incidence report. The impact and the probability of this defect is also calculated.

7.2.5. Review the Results

Once all the tests are performed and the defects are identified, the system is assessed on its acceptability. Based on the impact as well as probability of the defect manifesting, the system is deemed to be acceptable or not. However, it is very important to realize that there will always be flaws found. So, it is important identify whether it is necessary to fix it since a system with flaws that is delivered is better than a flawless system that is never delivered. 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 it 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

There is always a possibility of new changes being introduced at any stage of the testing. So, it is very important to accommodate that during the testing process to ensure that the testing is effective in identifying the defects.

7.3. Approach for RoomieLah

There is a well-defined procedure for testing. The QA Manager and Engineer will design the test runs. The testers will carefully carry out each individual test and document the result if it passed or failed. If a test has failed, then the developers who made that part are responsible for identifying where the defect is and correcting it.

The QA Manager will review the document containing the test results. When the QA Manager decides that the testing is complete, a test report is created and sent to the Project Manager who will review it and if satisfactory, will marking the testing process as complete.

This approach was designing keeping in mind the following aspects:

- 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 of the people in the proposed tools and techniques.

8. Item Pass/Fail Criteria

It is very important to define the pass and the failure criteria for each test case. There are two aspects you need to consider when trying to classify a test case as failed. They are:

a) Severity - The impact that the defect can have on the application.
   i) Critical
   ii) Major
   iii) Moderate
   iv) Minor
   v) Cosmetic
b) Probability - The probability that the defect will manifest itself.
   i) High
   ii) Medium
   iii) Low

Any test case that has Moderate and higher severity and Medium and higher probability fails.

Each test case will contain the input and the correct expected output in order to make it easier for testing. If the obtained output exactly matches the expected output, then the test case is deemed to have passed without any issues. If the obtained output is not exactly the same, then further analysis must be done to identify the severity and the probability of it before it is deemed to have passed or failed. In case of any ambiguity, the tester must consult the QA Manager who will make the decision.

The completion criteria for this plan are:

- 100% of unit testing cases are complete and have passed
- 80% of integration testing cases are complete and 10% cases or lesser have minor defects
- A minimum of 100 users have given their feedback for UAT

9. Suspension Criteria and Resumption Requirements

9.1. Suspension Criteria

Suspension Criteria defines the criteria under which the software testing will be suspended. The test team can decide to suspend the software testing process partially or completely. Suspension is also known as Test-Stop criteria for the testing process.

Below are a set of criteria that lead to the suspension in the software testing process:

- The scheduled time for testing is over.
- The budget allocated for testing is over.
- When a high impact defect is identified that other parts of the testing depend on.
- The test environment for conducting the test is unavailable.

9.2. Resumption

Resumption specifies the conditions due to which a suspended testing process might be resumed.

Below are a set of criteria that lead to the resumption in the software testing process:

- When there is negotiation with the client to extend the time required for testing.
- When there is negotiation with the client to the budget allocated for testing
- When the high impact defect is fixed, and the other parts can be tested.
- The test environment for conducting the test is available

10. Test Deliverables

The deliverables included in this Test Plan are:

1) Test Cases
2) Test Report
3) Test Plan Document
4) Revision Logs
5) Defect logs with solutions implemented

11. Test Tasks

The following activities must be completed:

- Preparation of the Test Plan.
- Identification of the items to be tested.
- Identification of the method of conducting tests
- Assignment of Tester for each test case
- Testing
- Fix bugs and errors that are discovered
- Creation of the defect logs
- Creation of the test report

12. Environmental Needs

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

- Mac OS device.
- git (used for source version control).
- Visual Studio Code - IDE used
- The Android platform tools.

13. Staffing and Training Needs

All the testers need to be trained to set up the environment to run the application and perform the tests correctly. The QA Manager will be responsible for finding methods to train the testers on the Flutter testing frameworks. The Front-end and the Back-end developers will assist the QA Engineer in case they encounter any issue.

14. Responsibilities

Role	Responsibilities
Project Manager	Deciding on the features that must be tested. Being in contact with the QA Manager and the Lead Developer to ensure that the testing process is following the intended timeline.
QA Manager	Designing the overall testing process. Identifying the risks as well as establishing the contingency plan in case the risk manifests itself. Being the point of contact between the QA Team and the other teams.
Lead Developer	Providing required training in code details. Providing solutions for running the code Conducting white box testing on the entire system
Front-End Developer	Conduct white box testing on the front-end components and carefully document each outcome and send it to the Lead Developer.
Back-End Developer	Conduct white-box testing on the back-end components and carefully document each outcome and send it to the Lead Developer.
QA Engineer	Conducting black box testing and carefully document each outcome and send it to the QA Manager.

15. Schedule

The testing phase for RoomieLah will last for up to 4 weeks, the details of which are given in the Project Plan. Our planned activities will be as follows:

Task	Estimated Time Taken (days)	Start Date	End Date
Produce Test Plan	5	9th March	16th March
Unit Test	10	14th March	25th March
Integration Test	10	21st March	1st April
Load Test	10	21st March	1st April
Produce Requirement Test Coverage Report	8	28th March	6th April

16. Risks and Contingencies

The risks and methods to mitigate them are as follows:

Risk	Contingency
Testers are unavailable	Other team members, specifically from the development team will help with the testing.
Delay in finishing the test items due to the delay in the implementation phase.	Closely monitor the delay and amend the release scope so there is sufficient time for testing.
Testers are not trained properly	The QA Manager is responsible for overseeing the training of both the QA Engineer as well as the Developers.
Testing tools and software doesn't work as intended or crashes	Immediately try to fix the tool or find other alternatives and reschedule the testing tasks.

17. Approvals

The QA Manager designs the tests and sends it to the Project Manager for approval.
After receiving the approval from the Project Manager, the QA Manager supervises the implementation of these tests, which is done by the QA Engineer and the Developer. The QA Engineer and the Developer prepare an in-depth documentation of the tests and send it to the QA Manager. The QA Manager ensures everything is being done correctly and reports to the Project Manager.

18. References

Document	Link
Project Plan	https://github.com/tayalaks2001/roomie_lah/blob/main/Docs/Lab%203/Project%20Plan.pdf
Requirements specifications	https://github.com/tayalaks2001/roomie_lah/blob/main/Docs/Lab%202/System%20Requirement%20Specifications.pdf
Project Proposal	https://github.com/tayalaks2001/roomie_lah/blob/main/Docs/Lab%201/Project%20Proposal.pdf