RoomieLah
RELEASE PLAN

Version 1.1

03/29/2022

`

VERSION HISTORY

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	Agarwal Gopal	22/03/22	Pandey Pratyush	22/03/22	Initial Change Management Plan
1.1	Agarwal Gopal	26/03/22	Pandey Pratyush	29/03/22	Final Revision and added Appendices

`

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

`

1. INTRODUCTION

This document elucidates the release plan for our application *Roomie Lah!* along with other relevant implementation details for the stakeholders. It describes the release strategy for the application and provides details about the past, current and possible future releases. Version 1.1 will be the major release of the application to users (NTU students).

The scope of activities includes assumption (if any), dependencies, constraints, potential risks, and the release strategy. The intended audience for the document may include Lab supervisor, Internal development, product and infrastructure teams. The document will be revised and updated to keep track of all releases of the applications. This will also help solve problems of users of older versions of the application. The document should not be released to the public as it may contain sensitive information like system architecture, context diagram and information about past and future release.

2. REFERENCED DOCUMENTS

***Table 1**: Referenced Documents*

Document Name	Document Number	Issuance Date
Project Proposal	1.0	09/02/2022
Project Plan	1.0	23/02/2022
Quality Plan	1.0	15/03/2022
Risk Management Plan	1.0	15/03/2022
System Requirement Specification	1.1	23/02/2022

3. OVERVIEW

Roomie Lah! is a cross platform mobile application wherein university students could create their account and fill in their roommate preferences and the traits they would be comfortable with, after which the application would dynamically suggest to them other students who they might be compatible with.

In case a match occurs, they would then have an option to chat with the matched user and if needed, connect with them on social media. The larger aim is to create a one-stop solution for university students to find roommates they are compatible with and thereby improving overall living experience.

The mobile application was developed using Flutter for frontend and Google Firebase for the database. The recommendation algorithm was developed using Python and deployed using Flask on a Heroku server. The version 0.0 was used for testing and improvement purposes and in the subsequent version, major features like algorithm and chat server communication were added. The version 1.1 is planned to be the first beta release which is rolled out to

university students at NTU. The waterfall lifecycle method was used for developing this application. The following is the updated ***use case diagram*** which provides a high level context of the system:

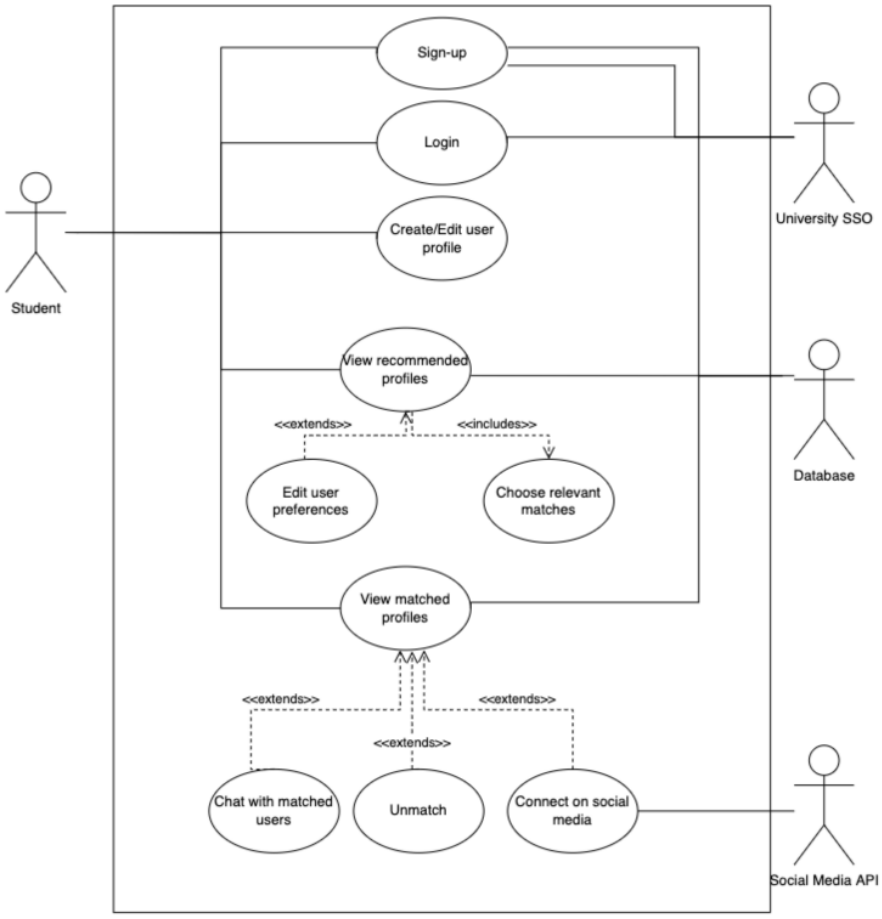

***Figure 1**: Use Case Diagram*

The following is the updated ***system architecture diagram***:

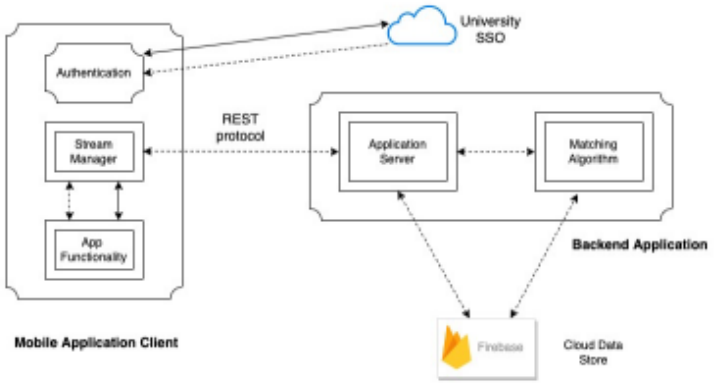

***Figure 2**: System Architecture Diagram*

4. ASSUMPTIONS, CONSTRAINTS, RISKS

4.1. Assumptions

The product development was made based on the following assumptions:

- The team members of the project will be able to complete all the tasks assigned to them.
- The developers and the customers can communicate effectively to ensure that the product meets the requirements of the customer.
- The project timeline is at least loosely adhered to.
- There is no significant increase from the budget calculated for the project.
- There is no major change in the requirements of the customer.

4.2. Constraints

The product development was made based on the following constraints:

- The size of the team is small and consists of only 7 members.
- The budget allocated for the project is limited and fixed. So, the product must be delivered within the allocated budget and resources.

4.3. Risks

The detailed description of the risks can be found in the Risk Management Plan. A summary is given below:

__Table 2__: Risk Analysis

S. No	Risk	Probability	Impact
1.	**TECHNOLOGY**		
1.1	The system might not work as intended on the end devices due to device requirements.	**Low**	**High**

1.2	The server potentially crashes, and essential data is lost.	**Low**	**High**
1.3	Firebase API malfunctions	**Low**	**Medium**
1.4	The Version Control System in use might experience fallacies.	**Medium**	**High**
2.	**PEOPLE**		
2.1.	The key people working on the project might leave the project.	**Medium**	**Medium**
2.2.	Conflicts between the developers and project manager	**Low**	**Medium**
2.3.	The people working on the project might fall ill and would not be able to work for a period of time.	**Medium**	**Medium**

2.4.	The team lacks motivation to complete deliverables.	**Low**	**Low**

2.5.	The developer team experiences domain knowledge inadequacies.	**Low**	**Medium**
3.	**ORGANIZATIONAL**		
3.1.	The organization might be restructured causing a change in the people responsible for different components that might lead to confusion.	**Low**	**Low**
3.2.	The project might introduce contract-based members, who can affect team communication and morale.	**Low**	**Low**
4.	**TOOLS**		
4.1.	The final code generated by the SDK might not meet the required performance requirements.	**Low**	**High**
4.2.	Possibility of a single point of failure like loss of all the data that might severely affect the timeline of the project.	**Medium**	**Medium**

4.3.	The deployment servers might not be scalable enough to meet the demands of peak user traffic	**Low**	**High**
5.	**REQUIREMENTS ELICITATION**		
5.1.	There might be changes requested by the stakeholders at a later stage of the software development cycle.	**Medium**	**Medium**
5.2.	There might be a difference in the functional requirements documented and functional requirements developed.	**Medium**	**Medium**
6.	**BUDGET**		
6.1	Budget allocated for the project might not be sufficient to cover additional expenses.	**Medium**	**Medium**

7.	**ESTIMATION**		
7.1.	The time taken to finish a deliverable is underestimated.	**Medium**	**Medium**
7.2.	The rate of bugs and issues in the system might be underestimated	**Medium**	**High**
7.3.	The amount of resources required for the completion of a component is underestimated.	**Low**	**High**
7.4.	The number of users in the system might be wrongly estimated. This might impact the design decisions badly.	**Low**	**Medium**

Table <Number>: Risk Analysis

5. RELEASE APPROACH

5.1. Rationale

Team Strawhats has used the Waterfall model as the ideal choice of lifecycle for development of RoomieLah software. The main reason for using the Waterfall model is because it is much more structured than other traditional methods wherein each phase must be completed before the next phase of development. Further, it is relatively easy to accommodate any changes made to the software during the development process. Given the cost constraints, as mentioned in the Project Plan, the Waterfall model will help to decrease costs associated with the development process. Since the release strategy adopted depends on the type of development lifecycle, the release approach specified here will be in accordance with the Waterfall methodology. Extensive testing

coupled with thorough documentation between any two phases of the development process will be carried out, to ensure that RoomieLah remains bug-free, errors (if any) are reproducible, and successive phases of development can rely on documentation to speed up the development process. Such steps will help ensure that any release developed also meets appropriate quality requirements.

5.2. Release Strategy

The Phased Function Rollout release strategy will be used for RoomieLah. Phased rollouts are the process of building and refining your product over multiple iterations. The core methodology helps you gain more control over various aspects of your development pipeline, from inception to launch day. It includes incremental development and implementation of modules which are later combined. It is to be noted that these are capable of existing independently too. It is known to be well suited to the Waterfall software development cycle chosen wherein development only proceeds to the next step if the current phase is tested completely.

We will employ the release process - Continuous Integration followed by Build - Package - Deploy. Continuous Integration is a software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run. It is better than one big integration move as it allows regular updates of the integrated code which helps in increasing the quality of the product overall. Added benefits are - everyone having access to the latest build and everyone being able to see what is happening. For RoomieLah, Git has been chosen as the default Version Control Software for Continuous Integration.

An important aspect of the Release Strategy would be for the Release Engineer to ensure the correct combination of versions of all software components and that they are using the correct data. This must build an executable for delivery and should satisfy the CRISP requirements:

- Complete
- Repeatable
- Informative
- Schedulable
- Portable

The Build - Package - Deploy process will involve the following sequence of steps to be carefully executed.

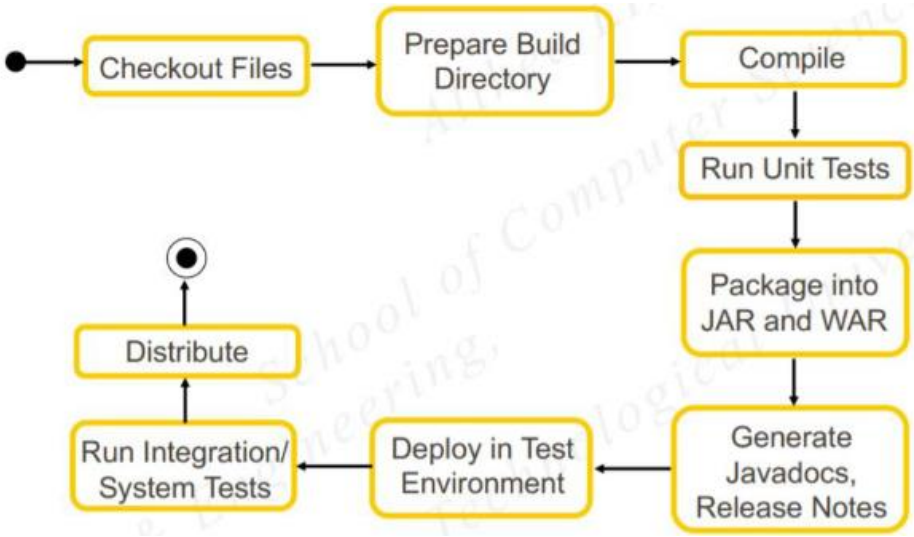

Figure 3: *Deploy Process*

Since the core idea behind the product RoomieLah is extremely unique and very much needed in the current Singapore scenario, it will be widely accepted and used by university students. This will help us receive feedback from a large audience very soon which will result in more areas of development. It will cause the team to use one of the following types of releases depending on the development/upgrade made.

- **Major Release** - In case there is any addition or upgrade to the core principle/idea of RoomieLah a major release will be made. This includes but is not limited to essential parts of the application without which it would never function as expected such as the match-making algorithm or database migration.
- **Minor Release** - In case there are any small changes or addition of features which do not change the entire app but only affects a part of it will be a part of the Minor Releases. This includes but is not limited to improving user experience, upgrading user interface or removing significant bugs
- **Revision** - This will include the smallest of changes which have been made to be application and have to be released. This includes but is not limited to minor bug fixes.

For preparation of any of the releases, our chosen Version Control System - Git will be used. Useful features such as branches will be used and steps such as tagging releases, distribution files, etc. will be done.

5.2.1. Release Content

Based on the Roomie Lah!'s Requirement specifications and use case descriptions, the following are the functionalities of the application:

1) User Registration and Login

2) Create and Edit User Profile
3) View Recommended Profiles
4) Match and Unmatch profiles
5) Chat with Matched profiles

Release Version	Release Type	Release Content
0.0 (Beta)	Internal	Use Case 1 and 2. These are the basic functionalities where a user can register and create his/her new account and input his/her profile information as well as preferences. This must be implemented and tested before the other functionalities can be added.
0.1 (Beta)	Internal	Use Case 4. Now the user will be able to view other users and their profile and preference. If the user believes that the other user is a potential roommate candidate, based on his/her profile, then can match with the other user. This is the core functionality of the application and must be implemented and tested before the first major release is made to the users.
1.0	Major Release	Use Case 3. An algorithm is implemented that will consider the user profile and preference details to recommend potential roommates candidates that the user would prefer. This makes it easier for the user. With this release, the application will be available on both Google Playstore and Apple AppStore. The team will also start collecting feedback from the end users in order to improve the application.

1.1	Major Release	This will contain the tested implementation of Use Case 5. The users will be prompted of the new update and will recommend the user to update.
1.x	Revisions (including patches)	The bugs and the feedback obtained from the users will be used to further improve the application. The updated versions of the application

5.2.2. Release Schedule

The foundation of a successful project is a well-laid plan. A release plan acts as a project's map, providing context and direction on product goals, vision, and expectations. By scheduling a project into Agile releases, the product managers of RoomieLah can better manage project constraints and adapt to evolving needs or challenges that arise through the development stage while regularly producing product deliverables for the end user.

Release Version	Release Date	Functionality Released	Estimated Story Points
0.0 (Beta)	1st March 2022	Signup, login, logout	3
0.1 (Beta)	15th March 2022	View profiles, match with users	5
1.0	20th March 2022	Recommendation System	5
1.1	29th March 2022	Chat with other users	7
1.x	Recurring monthly revisions and patches	Patch the bugs (if any)	2

The above timeline has been chosen to ensure timely completion of RoomieLah. Well spaced out minor releases and patches will ensure that the users do not get frustrated by constant updates.

5.2.3. Release Impacts

Release version	Business Impact	System Impact	Goals and Objectives

0.0	The users can create an account	The database will be populated with the user sign ups	Create market awareness and open sign ups
0.1	The beta version of the app is launched. The users can view potential roommates and match.	The server will be up and running. Several clients will connect with the server and exchange data.	Launch the basic functionalities of the application. Record bugs if any
1.0	The first official launch of the product. Introduction of the recommendation system will provide better roommate options to choose from. Thus, making the users hooked to the application and thereby increasing the user engagement.	The recommendation algorithm will be added to the server. Using users' data Machine learning models will be trained in the backend to improve existing algorithms.	Launch the product. Increase the user engagement rates by 15%.
1.1	Introduction of the chat feature will enable the users to interact with each other, thus impacting the user retention rates and increasing the number of impressions, clicks and matches.	The chat server will be up and running. Database will be updated to store chat histories.	Launch the chat functionality. Increase number of matches by 10% and retention rates by 25%.
1.x	Improve user experience by fixing any minor issues faced by users or potential threats to users' data.	Parallel updates of main and release branches in Git VCS	Decrease latency and improve system availability.

5.2.4. Release Notification

When RoomieLah has a new release, it is critical that the stakeholders and daily active users of the app are made aware about the availability of an update. This is achievable via a detailed notification-release mechanism. Notifying the users of a release ensures they know the application is being improved continuously and that this new release would further try to make their user experience better.

The release notification mechanism involves the following methods:

1. **In-app notifications** while users are using the application so that they are reminded every time they use the application.
2. **Push notifications** when the app is not in the foreground, from the application store native to the environment. For example, RoomieLah's update would be available to download on the AppStore for

iOS and Google PlayStore for Android.

3. Email and text message reminders to update applications.
4. Firebase Cloud Messaging SDK for notifications over the cloud to relevant users. This uses the following architecture to achieve a high throughput and maximum reach:

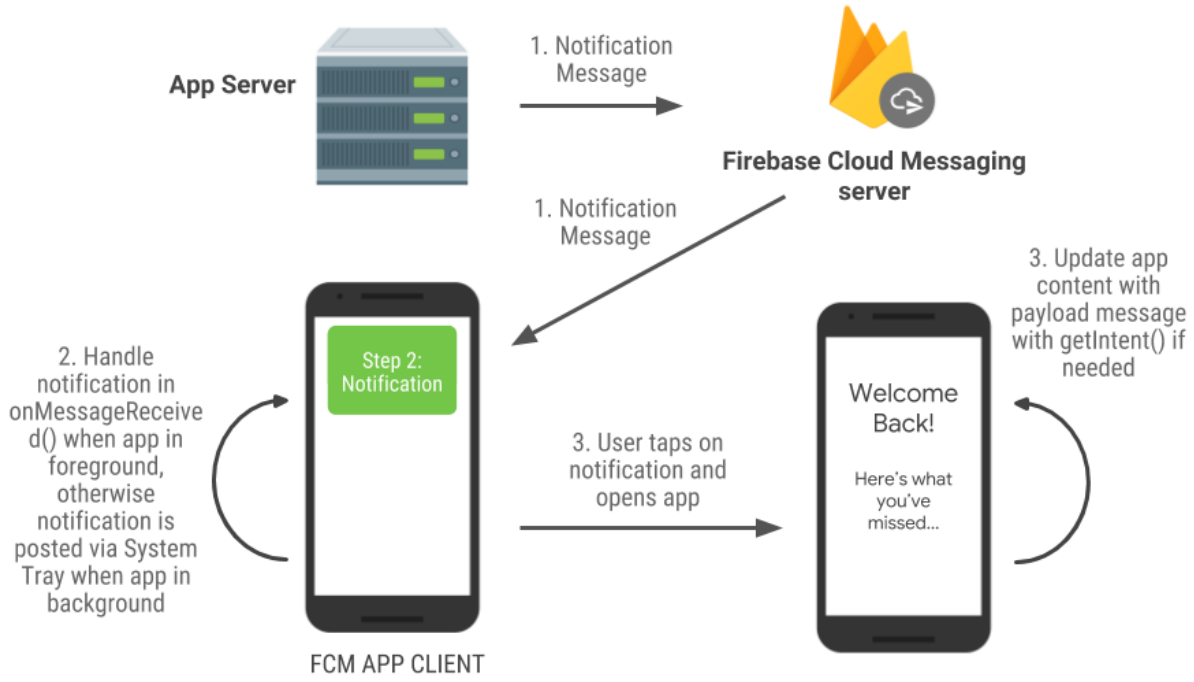

Figure 4: *Firebase Cloud Messaging Architecture*

Along with updating stakeholders, it is also critical that the development team of Team StrawHats be informed of the approval of the update and the changes it curtails. This ensures that each team member is aware of the new version, even if they were not directly involved in its update. The primary mode of communication would be the communication platform used by the team viz. Microsoft Teams, Slack.

Further information regarding the information provided, the time frame etc. will be provided as below:

Stakeholder	Information included in the notification	Timeframe for receipt of notification
Users	Changes made, which includes features updated, features added, bug fixes made, how the new One day prior to the release of version, followed by recurring reminders in the application changes will positively affect the user.	One day prior to the release of version, followed by recurring reminders in the application changes will positively affect the user (if appropriate permissions are provided by the user)
RoomieLah Team	All the information to bring the entire team up to speed, with important information about	

	architecture changes, availability of documentations, team members involved in the development of that release as well as user-side changes made in the application	Immediately after the approval of the new release

6. GLOSSARY

Term	Definition
Use Case Diagram	Simple representation of user functionality in the system
Waterfall Lifecycle Method	Linear lifecycle method for software development
Release	A particular version of the application
Flutter	UI toolkit developed by Google
Firebase Cloud Messaging	Backend Notification service offered by Firebase

7. ACRONYMS

Acronym	Definition
SDK	Software Development Kit
VCS	Version Control System
NTU	Nanyang Technological University

8. APPENDICES

Project Proposal:
https://github.com/tayalaks2001/roomie_lah/blob/main/Docs/Lab%201/Project%20Proposal.pdf

Project Plan:
https://github.com/tayalaks2001/roomie_lah/blob/main/Docs/Lab%203/Project%20Plan.pdf

Quality Plan:
https://github.com/tayalaks2001/roomie_lah/blob/main/Docs/Lab%202/Quality_Plan.pdf

Risk Management Plan:
https://github.com/tayalaks2001/roomie_lah/blob/main/Docs/Lab%203/Risk_Management_Plan.pdf

System Requirement Specifications:
https://github.com/tayalaks2001/roomie_lah/blob/main/Docs/Lab%202/System%20Requirement%20Specifications.pdf