
# Foodify

## Release Management Plan

**Version:** 1

**Last Modified:** October 24, 2022

# APPROVALS

**Foodify's Approving Authority:** Project Manager

| Signature | Printed Name | Date | Phone Number |
|---|---|---|---|
|  | **Anandarajan Sindini** | **25 October, 2022** | **+65 1200 5421** |

**Foodify's Approving Authority:** Release Engineer

| Signature | Printed Name | Date | Phone Number |
|---|---|---|---|
|  | **Kolady Anamika Martin** | **25 October, 2022** | **+65 1200 5421** |

# REVISION HISTORY

| Version | Date | Organization/Point of Contact | Description of Changes |
|---|---|---|---|
| 1.0 | 10/25/2022 | Verma Nandini<br>Kolady Anamika Martin | Initial Release Management Plan |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# TABLE OF CONTENTS

# LIST OF FIGURES



# LIST OF TABLES

# 1. INTRODUCTION

## Purpose

This document lays out the release plan for Foodify, a web application developed by Team Runtime Terror. The purpose of the release plan is to manage, plan, schedule and control the software build through different stages and environments, including testing and deploying software releases. This also helps create an actionable plan built on specific features, enhancements and fixes to build.

## Scope

This document describes the release strategy for Foodify and provides relevant details for all past, current and future releases of the platform, with the version 1.0.0 being the release version when the application is first rolled out for users.

It will be updated from time-to-time and utilized to keep a track of all the releases of the application. It will help all future team members understand the progress of the application over the years and make more informed decisions. It will also help developers resolve any issues the users of older releases may be facing. The document will also be used to capture and track features for upcoming releases.

## Expected Evolution

The expected evolution of Foodify's documentation will likely include issues such as fixing bugs and possibly new functionalities arising from user feedback. Additionally, maintenance must be done to ensure the system is up-to-date with new technology advances and industry trends.

## Target Audience and Security/Privacy Considerations

The main audience of the document will be the internal development and product team. It should not be made public as it may contain sensitive information about future releases. Some information from the document may be made public to provide

more information to the users once the Foodify web application is released. However, it must be done under the discretion of the project manager.

# 2. REFERENCED DOCUMENTS

**Table 1:** Referenced Documents

| Document Name | Issuance Date |
|---|---|
| Foodify Project Proposal | 09/14/2022 |
| Foodify Project Plan | 10/13/2022 |
| Foodify Quality Plan | 10/13/2022 |
| Foodify System Requirement Specification | 09/29/2022 |

# 3. OVERVIEW

## Background

Food waste is one of the biggest waste streams in Singapore and the amount of food waste generated has increased by around 20% in the last decade. In 2019, Singapore generated around 744 million kg of food waste. This is equivalent to two bowls of rice per person per day or around 51,000 double decker buses. An average Singapore household wastes around 1.5 kg of food each day, the majority of which can be prevented.

## Purpose

Avoidable food waste refers to food items that could have been consumed if better managed. Expired food items fall under the category of avoidable food waste. By large, users are not mindful of the expiry dates of their purchased food articles. This can be prevented if the user is reminded of their expiry date and also encouraged to consume them by recommending recipes that can be prepared using those food items. This is where Foodify wants to step in. Using our web application, we want to enable users to keep track of their foods, consume them before expiry and also sell/purchase food articles from other users. Our aim is to prevent food wastage due to expiry and contribute towards zero waste and planet sustainability.

## Scope

Foodify was developed by Team RuntimeTerror, a group of students in Nanyang Technological University. It was developed using the ReactJs Framework in the frontend along with Google firebase in the backend. The version 0.0.0 was developed internally for the purpose of testing and improvement. The version 1.0.0 is planned to be the first release which is rolled out to the users. The Software Development Life Cycle (SDLC) method was used for the development of the web application. The updated use case diagram for the system which provides a high-level context of the system is as follows:

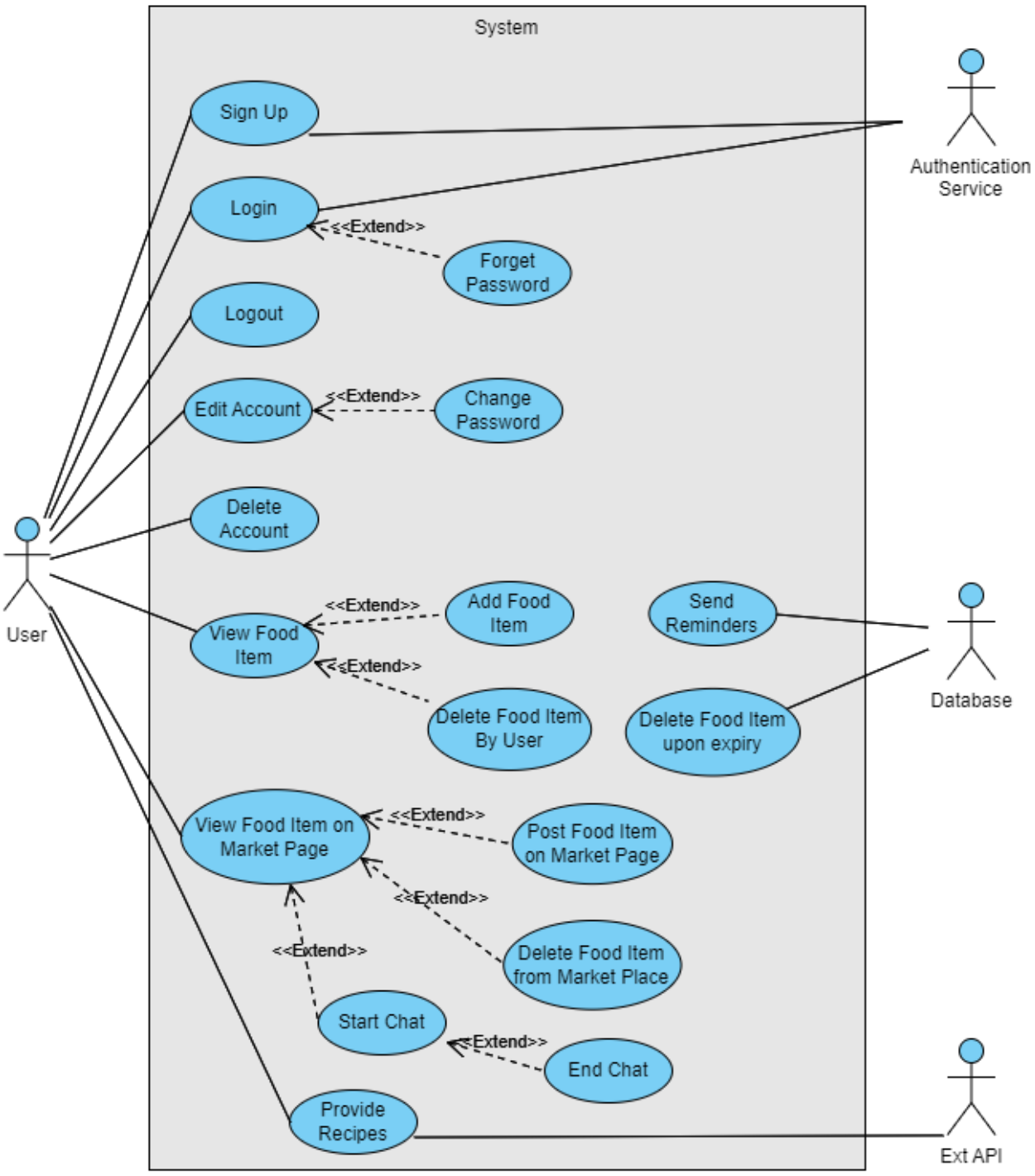


**Figure 1:** Foodify's Use Case Diagram

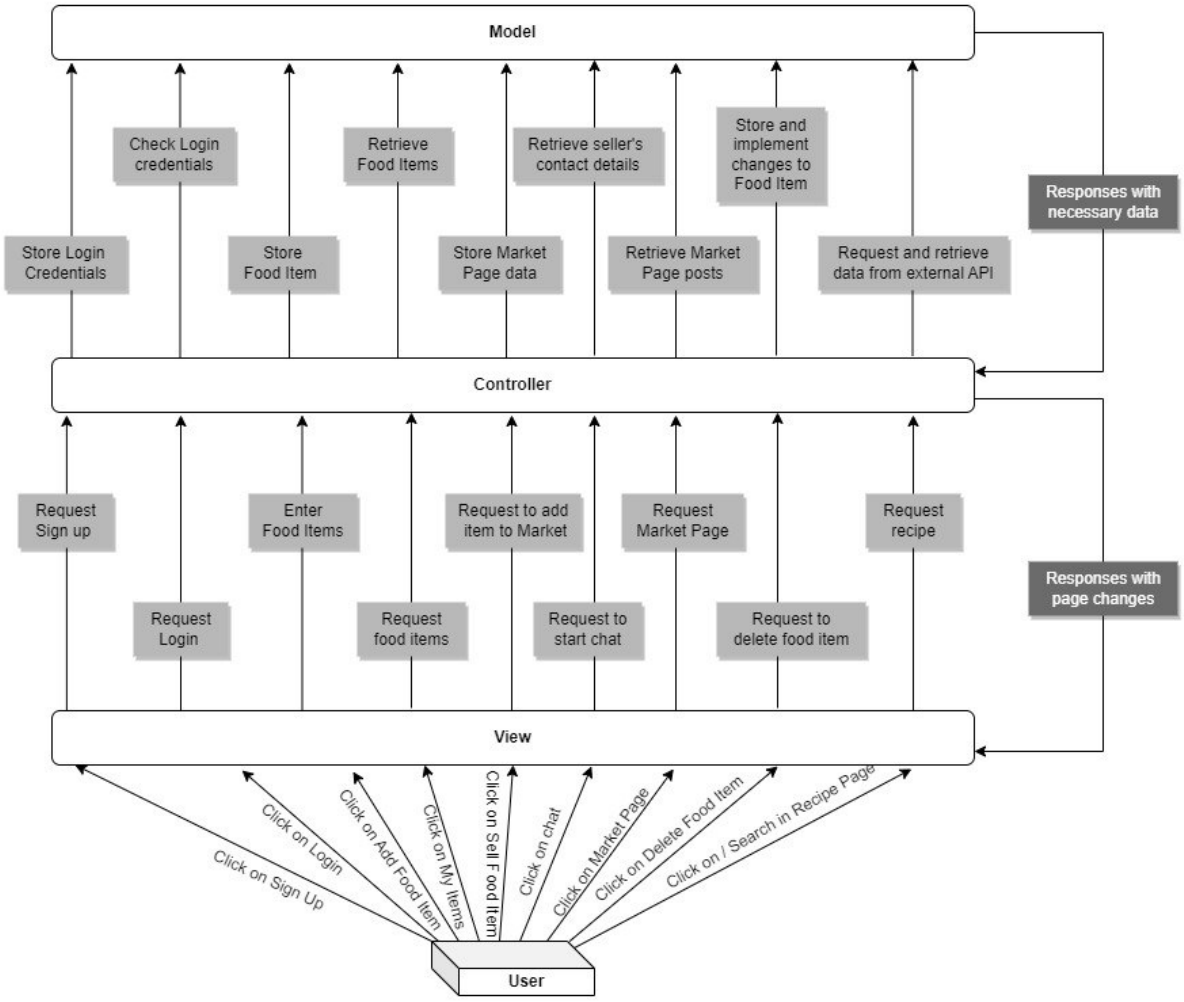


**Figure 2:** System Architecture for Foodify

# 4. ASSUMPTIONS, CONSTRAINTS, RISKS

## 4.1. Assumptions

**Schedule:**

1. The development and release timeline should not be altered considerably.
2. The team members are willing to commit the required amount of time for Foodify.
3. All tasks are completed as per the schedule.

**Budget and Financing:**

1. There are no sudden budget cuts or hidden costs.
2. The budget estimation and resources allocation for Foodify is precise.

**Development:**

1. The developers are skilled and able to meet expectations.
2. The developers and customers are able to collaborate on the development activities to get proper outcomes.

**Technology:**

1. The main framework of the system is to remain unchanged after release.
2. The required external dependencies, API and Firebase, are assumed to continue to be functional after the release of our application.

**Marketing:**

1. Support from Nanyang Technological University is required for our web application to be well-known and publicized.

## 4.2. Constraints

**Manpower:**

1. The team size is relatively small and consists only of 5 members. Careful work assignment and task planning must be carried out. The Project Manager must exercise a structured and flexible manpower allocation to tasks to easily move developers around the different sub teams to ensure the project tasks can be completed on time.

**Resources:**

1. Users' data on the web application are still owned by the users. Hence, all data can be used and stored in an external database only after obtaining rights (prior approval) by users.

**Time:**

1. The entire project from the planning to release stage is to be completed in a short time frame, between August to November 2022. To ensure no delays, project planning must be cautious and correctly estimated.

**Budget:**

1. With limited resources, impeccable planning must be carried out to ensure that the project is completed without a budget overrun.

## 4.3. Risks

| Serial Number | Risk | Probability | Effects |
|---|---|---|---|
| **1.** | **Technology** | | |
| 1.1 | Server crashes and essential data is lost | Low | High |
| 1.2 | Reused software components might have bugs and defects which may break the code | Medium | High |
| 1.3 | API used malfunctions after application release | Low | Medium |
| 1.4 | Inconsistent Programming Style. Code will be harder to read and understand and it will be difficult for another person to maintain the software | Low | Medium |
| 1.5 | Sensitive data being leaked/hacked | Low | High |
| 1.6 | Poor code design (e.g., high coupling). Modules cannot be separated. | Low | Medium |
| **2** | **People** | | |
| 2.1 | The developers are occupied with | Medium | Medium |

| | | | |
|---|---|---|---|
| | multiple projects at the same time and may be unavailable to attend possible meetings | | |
| 2.2 | Conflicts between the developers and project manager | Low | Medium |
| 2.3 | As the project duration is short, a team member falling ill and needing some time off would drastically delay project milestones | Medium | Medium |
| 2.4 | Undermined motivation within the team members and developers | Low | Low |
| 2.5 | Inadequate domain knowledge within developers | Low | Medium |
| 2.6 | Misunderstanding/ Miscommunication of requirements from stakeholders | Low | High |
| **3** | **Organizational** | | |
| 3.1 | The course project might be revamped with different components | Medium | Low |
| 3.2 | The project management might be restructured with different people responsible for the different parts | Low | Low |

| | | | |
|---|---|---|---|
| **4** | **Covid-19** | | |
| 4.1 | Unable to meet up due to restrictions, teammates contracting the virus. | Low | Medium |
| **5** | **Tools** | | |
| 5.1 | The use of Firebase for data storage might be insufficient for the user data | Medium | Medium |
| 5.2 | The deployment servers might not be scalable enough to meet the demands of peak user traffic | High | High |
| **6** | **Requirements** | | |
| 6.1 | Changes in requirements are proposed to improve the quality of the software at late stage of development | Medium | Medium |
| 6.2 | New use cases arise at late stage of development, giving rise to new requirements to be incorporated | Medium | Medium |
| 6.3 | Customers fail to understand the impact of requirement changes | Medium | Medium |
| **7** | **Estimation** | | |
| 7.1 | The time required to develop the various | Medium | Medium |

| | | | |
|---|---|---|---|
| | components of the software is underestimated | | |
| 7.2 | Abandoning of planning under pressure | Low | Medium |
| 7.3 | The rate of bugs and issues in the system might be underestimated | Medium | High |
| 7.4 | The size of the software application with be underestimated | Low | High |
| 7.5 | The user base of the application might be under or over-estimated, hence adversely impacting the design decisions | Low | Medium |

**Table 2:** Risk Analysis for Foodify

# 5. RELEASE APPROACH

## 5.1. Rationale

Team RuntimeTerror has used the Incremental Development Model as the ideal choice for the Software Development Lifecycle of Foodify. This model is a process of software development where requirements are divided into multiple modules or functions. Each module goes through the requirements, design, implementation and testing phases. Every subsequent release of the module adds function to the previous release.The main reason for using the Incremental model is because it allows for easier risk management. As the product is improved and reviewed with each build, risks can be easily identified early and fixed quickly to minimize the risks involved in the final product. Apart from this, it also generates working software earlier during the life cycle. This allows customers to respond to each build of the web application which will allow the developers to build the product as close to the desired outcome as possible thereby improving customer trust and satisfaction.

Since the release strategy adopted depends on the type of development lifecycle, the release approach specified here will be in accordance with the Incremental Model. Extensive testing will be carried out with every increment, to ensure that Foodify remains bug-free and meets our customer's needs. Such steps will help ensure that any release developed also meets appropriate quality requirements.

## 5.2 Release Strategy

The Release Strategy employed for Foodify will be **Phased Function Rollout**. This involves incremental development and implementation of separate modules or functions of the system, capable of existing independently, and combining all these in the end. This strategy sits very well with the Incremental development lifecycle, since this life cycle also divides its requirements into multiple modules, where each module goes through the requirements,

designing, implementation and testing phases. This ensures that each module is capable of existing independently. Furthermore, using Phased Rollout, the information gained from early implementation stages can be used to guide the rest of the process to reduce the number of bugs in the later stages.

To further facilitate the streamlining of the release process, **Continuous Integration** followed by a **Build-Package-Deploy** process will be employed. Continuous Integration, involves continuous merging of components and sub-systems along the way which will improve quality through enforcement of best practices, the benefits of which include anyone having access to the latest build and everyone being able to see what is happening, thereby decreasing time and costs for development. We will use **Git** as the default Version Control Software (VCS) to enable continuous integration.

Prior to a release, relevant members of Team Runtime Terror will have to ensure combining correct versions of all software configuration items, using the appropriate data, into an executable for delivery, to the users as well as to the test team. They must ensure that the entire process of building the software and then consequently releasing it, satisfies the following key requirements:

- Complete – self-sufficient
- Repeatable – automatic, consistent
- Informative – provides feedback
- Schedulable – auto-triggered

The Build-Package-Deploy process will include the following order of steps:

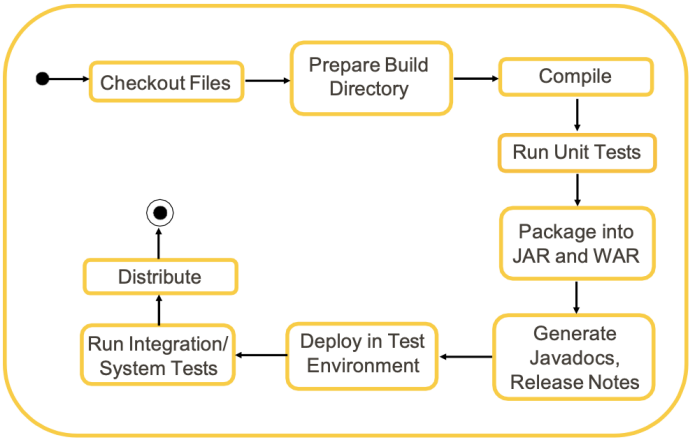


**Figure 3:** Build-Package-Deploy Process (Source - CZ3002 - Lecture Notes)

As the product matures and we start receiving feedback from users, Foodify will have many areas of improvement. Hence, it will surely involve several major and minor releases throughout the lifetime. The different types of releases involved for Foodify will be:

- **Major Release**: This includes the addition and updates of the most important functionalities of Foodify. These include features which form the basis of the application and without which, the application would not function, such as the reminder notification system, posting food items on the marketplace for sale, viewing all food items, etc. A major release will indicate a significant change and improvement in the application and will be an important upgrade for the user.
- **Minor Release**: This includes smaller feature changes and minor additions, which do not impact the existing features of the application at a large scale, but only serve to improve user experience and remove significant bugs that might have been introduced as part of a previous major or minor release.
- **Revision**: This includes the smallest changes as well as minor bug fixes and improvements. The importance of a revision is lesser than that of major and

  minor releases. Although recommended, Revision installations will be strictly optional.

- **Internal Release (Promotions)**: Throughout the development lifecycle of

Foodify, successive versions of the application would be made available within Team Runtime Teror. Such internal releases / promotions are absolutely essential before a major / minor release or a revision is made available to the users, to remove any bugs or errors discovered during testing by members not involved with the actual development.

### 5.2.1 Release Content

Based on NTUCollab's System Requirements Specification and Use Case Descriptions, following are the intended functionalities of the application:

1. **User Registration and Login (Sign Up and Login)**

   This use case intends to allow the user to login and register for a Foodify account. An independent profile allows the user to add in their food items, view them at one location, and sell excess food on the marketplace.

2. **Forget Password**

   This use case allows the user to reset their password in the event that they no longer remember it.

3. **Change Password**

   This use case allows the user to change their password as and when they wish to.

4. **Edit Account**

   This use case allows the user to edit his / her profile, specifically their name, phone number and password. This is incorporated as their personal information may change over time.

5. **Delete Account**

   This use case allows the user to delete their Foodify account if they no longer wish to use it.

6. **Logout**

   This use case allows the user to logout after using the Foodify web application.

7. **Add Food Item**

   This use case allows the user to add a food item to their account to get reminders via email notifications of the expiry of their foods. The food items inserted are stored in the database for use by the expiry reminder notification system.

8. **View Food Items**

   This use case includes displaying all the food items and their expiries to the user inserted by them.

9. **Delete Food Items by User**

   This use case allows the user to delete food items from their list in the event that they have consumed it before the expiry or if it has been sold on the marketplace.

10. **Send Expiry Reminder to User**

    This use case includes the system comparing the expiry date of food items and current date to notify the user regarding the expiry of his/her food items on the Foodify account.

11. **Post Food Item on the Marketplace**

    This use case allows the user to sell excess food on the marketplace at a discounted rate.

12. **View Food Items on Marketplace**

This use case allows buyers to view food items to purchase at discounted rates on the marketplace page.

**13. Provide Recipes**

This use case allows the user to view which can be prepared using their food items.

On the basis of the above, following are the releases and the content of each:

| Release Version | Release Type | Release Content |
|---|---|---|
| 0.0.0 | Internal (within Team Runtime Terror) | Use cases 1 to 6 (as mentioned above) to be ready for testing within Team Runtime Terror. These use cases are the bare essential for Foodify and hence need to be implemented and tested before a user-facing release is made. |
| 0.1.0 | Internal (within Team Runtime Terror) | This will include all the functionalities that were implemented and tested by internal release version 0.0.0 as well as use cases 7, 8, 10, 11 and 12.<br><br>These functionalities are critical for the usage and main purpose of the application. Hence, these use cases should be implemented and tested before the first major release is made available to the users. |
| 1.0.0 | Major (to the users) | This will include all the functionalities that were implemented and tested by internal release version 0.1.0.<br><br>With this release, Foodify will be made available as a web application. After a few weeks, Team Runtime Terror will begin collecting important information like adoption rate of the web application and user feedback. These would help to improve the application through future releases.<br><br>Concurrently, a forum where users can |

| | | |
|---|---|---|
| | | identify and report bugs in a release will be opened. This will help in fast and easy error-identification and increase test coverage that includes test cases. |
| 1.1.0 | Minor (to the users) | This will include tested implementations of use cases 9 and 13. Relevant notification mechanisms will be employed to let the users know of the availability of this important but optional feature.<br><br>Again, relevant statistics like user outreach about this release, etc. will be made so that Team Runtime Terror can make necessary amendments for future releases. |
| 1.1.X | Revisions (includes Patches) | As mentioned earlier, bugs and feedback collected earlier will be continuously worked on and maintained. The updated versions of software will be available through successive revisions, that users will be notified about as and when released. |

**Table 3:** Release Type and Content for Foodify

### 5.2.2 Release Schedule

The Release Schedule is an important aspect for Foodify as the frequency of releases can determine the user satisfaction. If the frequency of releases is too low users might be dissatisfied with the product as a whole as they might assume it's not maintained and does not have regular updates to add new features. However, too frequent releases can irritate users and they might choose to skip releases, thereby making them use older versions of the application.

The following is an approximate schedule for the different releases specified above:

| Release Version | Date of Release |
|---|---|
| 0.0.0 | 30th September, 2022 |
| 0.1.0 | 15th October, 2022 |
| 1.0.0 | 22st October, 2022 |
| 1.1.0 | 27th October, 2022 |
| 1.1.X | Recurring monthly revisions and patches |

**Table 4:** Timeline of Releases

The above timeline has been chosen to ensure timely completion of Foodify's first major release as well as at least a week for testing and bug identification. Well – spaced out minor releases and patches will help ensure users do not get irritated by frequent updates and at the same time are assured that continuous maintenance and improvement of the application is taking place.

### 5.2.3 Release Impacts

With every release, there are several components that are impacted by it. This includes the system, business processes, user interfaces and the overall release strategy. However, each release has a purpose and there are several goals and objectives that are met by it.

Following are the impacts of each release on each of the different domains:

| Release | Business Process Impact | System Impact | Goals and Objectives |
|---|---|---|---|
| 0.0.0 | Basic User Registration, Login and Profile features. | Updated source code in main branch. | Complement implementation and testing of fundamental features of the application. |
| 0.1.0 | Complete essential features of Foodify as a product. | Updated source code in main branch. | Complement implementation and testing of essential features of the |

| | | | |
|---|---|---|---|
| | | | application. |
| 1.0.0 | Complete Foodify application | Ready for deployment in Git VCS. | Foodify ready for deployment to actual users. |
| 1.1.0 | Improve business value through addition of new features | Minor release deployed to users. | First minor release to users with optional but helpful features of deleting items from the food items list and looking for recipes they can make with their food items. |
| 1.1.X | Improve user experience through bug squashes. | Parallel updates of main and release trunks in Git VCS. | Remove bugs, deploy patches, and introduce minor feature improvements continuously. |

**Table 5:** Release Impacts

Since Team Runtime Terror has employed the Build-Package-Deploy Process, the following machines are involved:

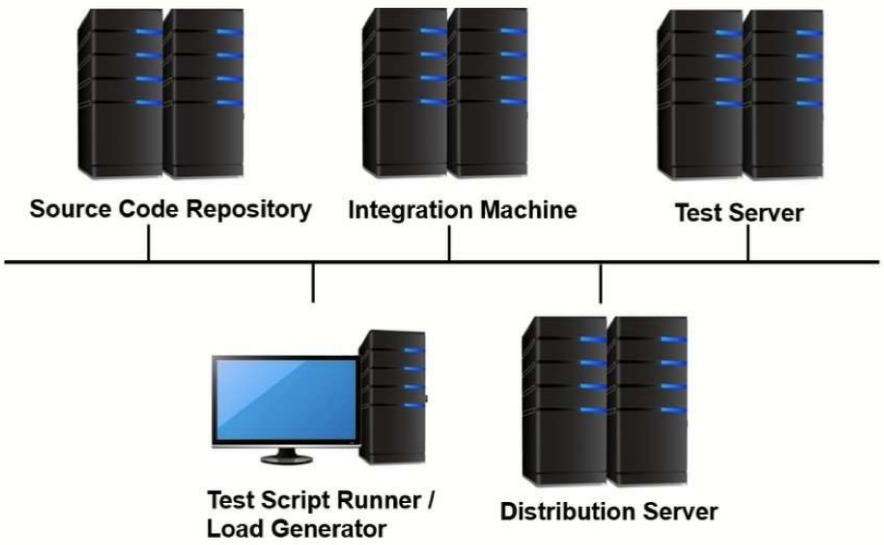


**Figure 4:** Build-Package-Deploy Configuration (Source - CZ3002 Lecture Notes)

Each of the above machines will be involved in all the releases. The source code repository will contain all the source code, the integration machine will be responsible for integrating different subsystems together, the test server and test script runner will carry out effective testing prior to any release and the distribution server will be actually involved in distributing both internal and external releases.

Appropriate identification of the impact of each release is essential as it helps to improve the quality of any future release and provides a good baseline. If any release encounters an issue during the process, it will be easier to avoid the same mistake in a future release. To support this process, Team Runtime Terror will ensure that all releases and related processes are properly documented and made available within the team, as a reference point for the future. Appropriate steps and guidelines shall also be specified for all issues encountered in the process of each release. These steps will help reduce the impact of errors and increase user satisfaction.

### 5.2.4 Release Notification

When a release version is generated, it is imperative to let the stakeholders and users of NTUCollab know about its availability. Notifying the users of releases ensures that the users know that the application is being upgraded and improved continuously and that every new release will ensure a better user experience.

The release notification mechanism will be employed through:

- Email reminders for users who have an account with Foodify.
- Detailed information upon user request through customer hotlines

Through future software releases, notifications would also be sent via text messages.

Such methods of notification will ensure that users stay up-to-date and do not miss out any release, thereby ensuring a better user experience.

Apart from the above methods that are intended for the users of NTUCollab, it is crucial to notify the entire team about the approval of a new release and the changes it involves. This is to ensure that all the members of the team are aware of the new version.

The primary mode of notification for team members will be official Email IDs, briefings and team meetings as well as text messages on the registered mobile phone numbers.

Notifications 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 | Features which were updated, new features added, bug fixes made, how the new changes will positively affect the user. | One day prior to the release of the version. |
| Foodify Team (notification of internal promotions) | All important information regarding architecture changes, availability of documentations, team members involved in the development of that release, user-side changes made in the application. | Immediately after the approval of the new release. |

**Table 6:** Notification Information and Timeframe

## 6. GLOSSARY

| Term | Definition |
|---|---|
| Build-Package-Deploy | A standard procedure which involves building the application, packaging the application into a software suitable for users to install and deploying it to the users. |
| Firebase | A backend platform created by Google for developing web applications. |
| Foodify | The name of the web application developed in this project. |
| Git | A free and open-source distributed version control system. |
| Release | A particular defined version of the application being made available for the users. |
| ReactJS | Free and open-source front-end JavaScript library for building user interfaces based on UI components. |
| Software Development Life Cycle | An incremental lifecycle method used for the development of software systems. |
| Use Case Diagram | Summarizes the details of system's users (also known as actors) and their interactions with the system |

# 7. ACRONYMS

| Acronym | Definition |
|---|---|
| API | Application Programming Interface |
| SDLC | Software Development Life Cycle |
| VCS | Version Control System |

# 8. APPENDICES

Foodify Project Proposal:

https://drive.google.com/file/d/1YKU1Z2Q3Bge3gv-LRE0SWCGbD4g9gT8a/view?usp=sharing

Foodify Project Plan:
https://drive.google.com/file/d/12HsSjnkHaIpRkSfJX_9foEPu21dU4xM2/view?usp=sharing

Foodify Software Quality Assurance Plan:

https://drive.google.com/file/d/1qHunISQCGAZhhSdLJXZxhUuRhaenZW7y/view?usp=sharing

Foodify System Requirement Specification:

https://drive.google.com/file/d/1Cg2jFAuaWHIAe_pdZJs2pdI-yOfPpJ-n/view?usp=sharing