

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE


Release Plan for “Go Where GaiGai” Web Application

Version: 1.0

Last Modified: 10/10/2022

Approvals

Submitting Organization's Approving Authority:

Signature 

Printed Name: Eugene Lim Zhi Jie

Date: 10/10/22

Phone Number: 86830294

Go Where GaiGai Project Manager

Revision History

Version	Implemented By	Revision Date	Approved By	Approval Date	Description of Changes
1.0	Eugene Lim	10/10/22	Eugene Lim	16/10/22	Initial Release Management Plan

Table of Contents

1 Introduction	5
2 Referenced Documents	5
3 Overview	6
4 Assumptions, Constraints, Risks	9
4.1 Assumptions	9
4.2 Constraints	9
4.3 Risks	10
5 Release Approach	11
5.1 Rationale	11
5.2 Release Strategy	11
5.2.1 Release Content	14
5.2.2 Release Schedule	16
5.2.3 Release Impacts	17
5.2.4 Release Notification	18
6 Glossary	20
7 Acronyms	20

List of Figures

Figure 1: Use Case Diagram	7
Figure 2: System Architecture Diagram	8

List of Tables

Table 1: Referenced Documents	5
Table 2: Prototype Implementation Changelog	6
Table 3: Risk Management Summary	10
Table 4: Content Release Plan	15
Table 5: Release Schedule	16
Table 6: Release Impacts	17
Table 7: Notification Mechanism Structure	19

1 Introduction

This document lays out the release plan for the software project “Go Where GaiGai”, developed by Team ONE. The aim of this document is to describe the release strategy adopted by the project team and provides details regarding the past, current, and future releases of “Go Where GaiGai”. The first major revision of this document to be released is Version 1.0.

The scope of this document includes any assumptions (if any), constraints and risks in the deployment of “Go Where GaiGai”. The document will also describe the release strategy chosen and explain the reasons for its selection. The Release Plan will be updated at each iteration of the software development lifecycle (SDLC) to keep track of all releases published to allow readers to understand the legacy issues faced during the development of “Go Where GaiGai” and capture new and upcoming features planned for the project. This document should not be made public as it contains sensitive information regarding the system’s architecture and future releases of the product. Any information from this document that is made public should be done so at the discretion of the Project Manager and stakeholders.

2 Referenced Documents

Document Name	Document Version	Issuance Date
Project Proposal	1.0	4/9/2022
Project Plan	1.0	29/9/2022
Quality Plan	1.8	18/9/2022
Risk Management Plan	1.0	19/9/2022
System Requirement Specification	1.0	18/9/2022

Table 1: Referenced Documents

3 Overview

Go where GaiGai is a website created to help busy friends and families plan outings with the help of an algorithm designed to dynamically suggest dining and leisure locations with the click of a button. This is made possible with an algorithm written to select locations based on user ratings and review counts extracted from Yelp via Yelp's public API. The website also accepts inputs from the user as user preferences. User preferences influence the behavior of the algorithm by allowing the user to choose between certain categories of locations, as well as asking the algorithm to select destinations within a close proximity to the user's current location. Once the algorithm suggests appropriate locations, the user can then add the locations into a planner for sharing and viewing. The planner will be exported in a plain text file format and can be imported in a similar fashion.

The website was developed using ReactJS for the front-end, and Python for the back-end database. The first build of the system, version 0.0, contained the application skeleton after the Lead Developer designed the system architecture for the project. From this version, the development team consisting of front-end and back-end developers implemented the features of the system into the prototype of Go Where GaiGai. Version 1.3 of Go Where GaiGai was the first build that was released publicly to stakeholders and clients to demonstrate use of the software and collect feedback for further development. The addition of major features and other quality of life features are as follows:

Software Version	Additions and/or Changes
0.0	Application skeleton and dependencies added
1.0	Back-end Database was implemented
1.1	Planner was implemented
1.2	Map view was implemented
1.2.1	Map view changed from leaflet API to Google Maps API
1.3	Map pins added

Table 2: Prototype Implementation Changelog

The latest versions of the use case diagram and system architecture diagram will be shown in this document to provide context to the high-level technical design concepts adopted in this software project. For the full use case model, the use case descriptions can be found in the *Project Proposal* document. The following is the latest **use case diagram** of Go Where GaiGai:

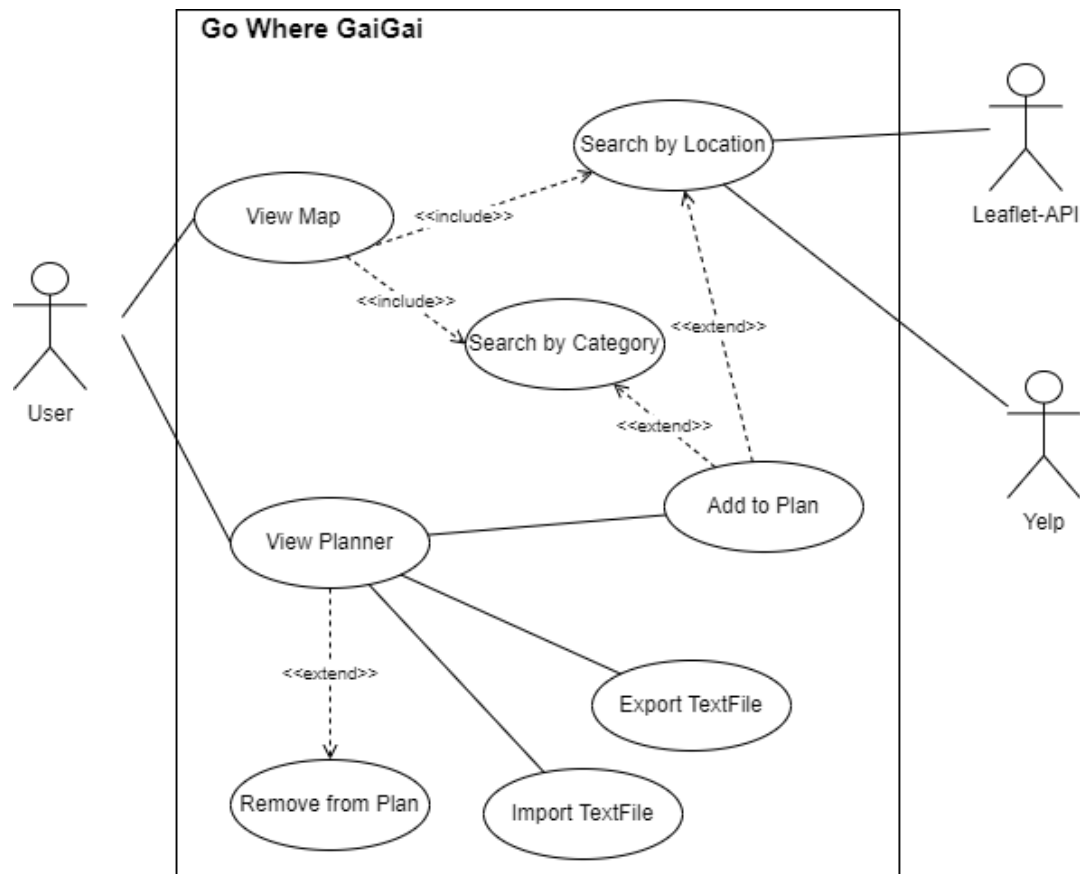


Figure 1: Use Case Diagram

The following is the latest **system architecture diagram** of Go Where GaiGai:

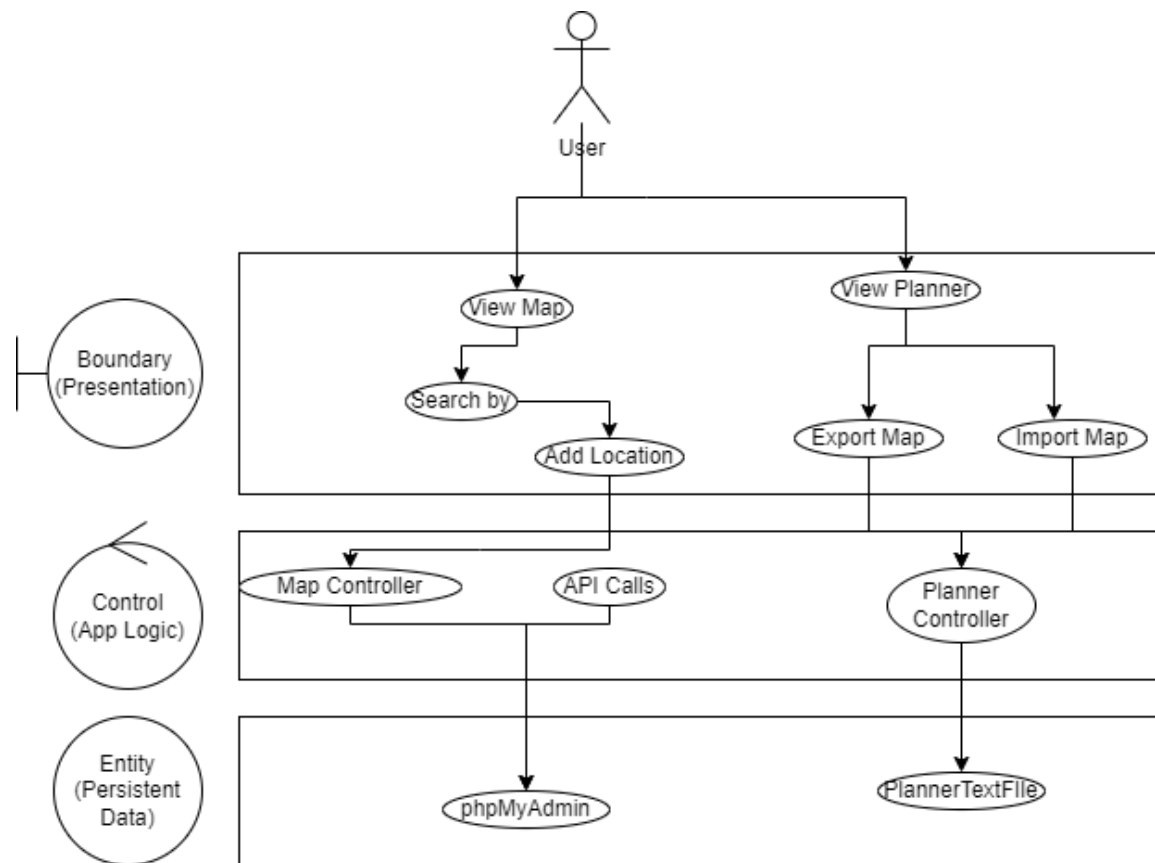


Figure 2: System Architecture Diagram

4 Assumptions, Constraints, Risks

4.1 Assumptions

The development of Go where GaiGai was made based on the following assumptions:

Schedule:

1. The Development and Release timeline should not be altered considerably without the consultation and approval of stakeholders, the Project Manager and relevant team leaders.
2. Project members are able to commit the amount of time required for the completion of Go Where GaiGai.
3. All tasks and sub-tasks are completed as per the Project Schedule.

Budget and Financing:

1. There are sudden changes to the budget calculated for the project.
2. The budget allocated for the project in the *Project Proposal* document is accurate.

Development:

1. The developers and QA team are skilled and able to meet expectations
2. The project team and clients are able to collaborate during development activities to support the development of Go Where GaiGai.

4.2 Constraints

Manpower:

1. The size of the project team is small and consists only of six members.

Functionality:

1. The data stored in the back-end database used for the search algorithm is kept updated provided by Yelp's public API.

Budget:

1. The budget allocated for the project is limited and fixed. Hence, the product must be delivered within the allocated budget and resources.

4.3 Risks

A complete analysis and description of the risks identified in this project can be found in the *Risk Management Plan* document. A summary is provided below:

Risk	Impact	Probability	Level of Control
Web Server crashes	High	Low	Medium
Database crashes	High	Low	Medium
Unable to handle server traffic	Medium	Low	Medium
API malfunctions	Medium	Low	Low
Inadequate domain expertise	Low	Low	Low
Change of data structure of API	High	Medium	Medium
Software bugs	High	Medium	Medium
Other project commitments	Medium	Medium	Low
Man hours underestimation	Medium	Medium	Medium
Conflict among team members	Low	Medium	Low

Table 3: Risk Management Summary

5 Release Approach

5.1 Rationale

Team ONE has adopted a plan-driven Incremental Development Model for the development of Go Where GaiGai. The Incremental Development Model allows the project team to gain and act on feedback from testers quickly within the short timeline of this project. This means that bug fixes and changes to the code can be released more frequently. The initial cycles of the incremental model will be used to ensure major features of the website are implemented, before any minor features and quality of live updates are introduced to meet the full system requirement specifications. Hence the Incremental Development Model helps the project team implement features faster, and saves costs involved in the more structured models such as the Waterfall Model.

Since the release strategy adopted will depend on the type of SDLC model chosen for Go Where GaiGai, a suitable approach will be in accordance with the Incremental Development Model. Extensive testing coupled with thorough documentation between any two phases of the development process will be carried out, to ensure that the project 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 the appropriate quality requirements.

5.2 Release Strategy

The release strategy Team ONE will employ for the SDLC of Go Where GaiGai will be a Phased Function Rollout. Similar to how the Incremental Development Model splits up the development process into multiple iterations, the Phased Function Rollout involves building and refining the software product over multiple iterations. This methodology allows each member of the project team to reevaluate their workflow at every iteration. This ensures that the deliverables produced at each stage is at its best quality. Coupling the Phased Function Rollout with the Incremental Development Model will result in the smooth delivery of multiple iterations, each with implementation, and testing of the implemented code.

In this strategy the release process used is Continuous Integration. Every release cycle will follow a Build - Package - Deploy phases. Continuous Integration follows a DevOps best practice whereby developers of a software team frequently merge code changes into a single central repository where builds are tested. The tests are automated with tools to assert the correctness and accuracy of new code with each integration. By breaking down the workload into smaller iterations, the small development team is able to ensure that the quality of implementations are tested with completeness therefore increasing the quality of deliverables. The Continuous Integration release process also gives all project members to the latest build via the repository. In the development of Go Where GaiGai, Git has been chosen as the Version Control Software, and Github has been chosen for the central repository before the source code is pushed into NTU's Tortoise SVN for the final build.

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 dependencies. These builds must be executable for delivery and demonstration of the product at all times and should satisfy the CRISP requirements:

- Complete - self-sufficient
- Repeatable - automatic, consistent
- Informative - closed feedback loop
- Schedulable - auto-triggered
- Portable - independent (of IDE)

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

1. Checkout files
2. Prepare build directory
3. Compile
4. Run unit tests
5. Package into JAR and WAR
6. Generate Javadocs, release notes
7. Deploy in test environment
8. Run integration/system tests
9. Distribute

Throughout the development life cycle of Go Where GaiGai, the system requirements proposed in the *System Requirements Specification* (SRS) document will be fulfilled by shipping out releases of different scales. The various types of release will imply different levels of impact to existing users, and understanding these release types will help Team ONE upgrade, update and maintain Go Where GaiGai. The types of release are as follows:

- **Major Release:** These releases carry major features that the web application cannot function without. In the context of Go Where GaiGai, the major features include the interactive map, browse locations page, activity planner and search algorithm. Major releases indicate a significant change/improvement or additions in the software product and will be an important upgrade for users of the application.
- **Minor Release:** A minor release introduces small improvements to the software's functionality. However, changes or modifications are made to existing features instead of the introduction of new features. These releases include the improvement of user experience via quality of life updates, refactoring of software interfaces, performance upgrades or the patching of significant bugs.
- **Revision:** Revisions include small changes to the application such as minor bug fixes, commonly known errors with low priority or changes to documentation such as user manuals. Revisions improve the product's performance in the specified operational environment.
- **Emergency Fixes:** Emergency fixes are reserved for urgent issues that may impact user experience and require immediate attention. Errors with very high priority and high impact especially to user's data and security are fixed and shipped in emergency fixes.

The use of Git as the Version Control System (VCS) will be through branches and will involve a structured process like tagging and release and distribution of files to project members. The full order will remain consistent throughout the use of Continuous Integration as the project's release strategy. The detailed steps are shown below:

1. **Create** release branch
2. **Checkout** release branch
3. **Build and test** release branch
4. **Create release** distribution file
5. **Test** distribution file contents
6. **Tag** release
7. **Hand-off** distribution file to QA

5.2.1 Release Content

The content of each release should correspond to the use cases from the Use Case Model, and system requirement specifications from the SRS document. The following is a list of content to be released:

1. Search Attractions

The user is able to search for dining/leisure destinations using the search function on the website. The website will utilize an algorithm which selects the best locations based on the user ratings and review counts obtained from Yelp's public API. The information in the database is polled continuously to ensure updated information is shown to the user and used by the algorithm.

2. Interactive Map

The website will display the results of the search on an interactive map of Singapore. The map can be viewed in a topographical or street view, and the user can zoom in and out of the map.

3. Activity Planner

From the Interactive Map or the Browse Locations page, the user can choose to add locations of their choice to an activity planner. The planner is displayed in the form of a schedule.

4. Search Attractions by user preferences

The user can also choose to input preferences before they search for a location. User preferences include the type of location such as dining, gardens, bars, etc, or different types of cuisines in restaurants. The user can also choose to have the algorithm search within a close proximity to the user's current location.

5. Display destination information

From the interactive map the user can click on the drop pins of each individual location, and the website will display basic information of the location such as address, postal code, its average user ratings and user review counts.

6. Import/Export Activity

The user can export their completed plan into a plain text file for sharing with other users via email or messaging app.

7. Browse Locations

The user can also search for locations from the Browse Locations page which displays information of each location in a card on a list. Each location card displays the basic information of each destination.

The following is a list of content planned for future releases:

8. Algorithm update

The algorithm will be updated to be able to record past user search choices, and use it as an input to suggest better locations for food/leisure experiences.

9. Linking View Map page and Browse Locations page

When the user clicks on a specific location card on the Browse Locations page or on a drop pin on the View Map page, the user will be able to seamlessly swap between the two pages.

10. User registration and login

Create a registration and login service to allow users to be able to share plans between their user's profiles or linking their email to the service. The users will be able to add each other to a friends list.

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

Release Version	Release Type	Content
0.0.0	Internal (Team ONE)	Application skeleton, no implementation of functional requirements are present in this version.
0.1.0	Internal (Team ONE)	Contents 1, 2 and 3 form the basis of Go Where GaiGai, hence they are to be added in this internal version for testing.
1.0.0	Major (public)	<p>Contents 4, 5 and 6 are extensions of content 1, 2 and 3 respectively. They add great value to the earlier use cases, and are shipped as major releases after testing is complete. This is the first version available to the general public.</p> <p>From this version, Team ONE will begin to collect feedback from testers, and users. The information collected will be used to improve the application through future releases.</p>

1.1.0	Minor (public)	<p>Content 7 is a minor update that adds a new page to improve user experience. Content 7 adds upon the features currently present in content 2 (interactive map).</p> <p>The relevant statistics like user downloads and bug reports will be collected to increase test coverage and for the project team to make necessary changes for future releases.</p>
2.0.0	Major (public)	Contents 8, 9 and 10 are new features planned for release in the future once code is implemented and the features pass QA requirements.
2.0.X	Revision (includes patches)	<p>As mentioned earlier, bugs and feedback collected will be continuously worked on and maintained. The updated versions of software will be available through successive revisions, that users can download as per convenience.</p>

Table 4: Content Release Plan

5.2.2 Release Schedule

A release schedule is required to act as a guideline for the project team to adhere to. A release acts as a project's map, providing context direction on product goals, and the project team's vision and expectations. By scheduling a project into Agile releases, the Project Manager of Team ONE can better manage project constraints and adapt to the evolving requirements and challenges that arise throughout the development while producing quality deliverables for the end user. A schedule is planned out below:

Release Version	Release Date	Functionality Released
0.0.0	11 September 2022	Application skeleton and dependencies
0.1.0	18 September 2022	Search Attractions, Interactive Map, Activity Planner
1.0.0	2 October 2022	Search by user preferences, Display destination information, Import/Export Plan
1.1.0	9 October 2022	Browse Locations
2.0.0	23 October 2022	Algorithm 2.0, View Map and Browse Locations link, User registration and login

2.0.X	Recurring monthly revisions and patches	Patches and bug fixes
-------	---	-----------------------

Table 5: Release Schedule

The above timeline has been created to ensure a timely completion of Go Where GaiGai first major releases and future content updates. The dates shown above will take into account the time needed for the files to be distributed to the QA Manager and Engineer for testing of code. The releases are spaced out after discussion with the Lead Developer to ensure a smooth but quick delivery of system requirements at each iteration of the development cycle. The releases are also not rolled out too quickly to prevent the need for users to update the software too frequently while improvements and maintenance is performed on Go Where GaiGai.

5.2.3 Release Impacts

Each time the software is changed or upgraded the software enters a different stage of operational readiness. It also impacts processes such as the release strategy and the goals and objectives of the project team moving forward. The table below describes the impacts of each release in greater detail:

Release Version	Business Impact	System Impact	Goals and Objectives
0.0.0	NIL	NIL	Complete application skeleton to begin implementation.
0.1.0	Users can begin using the website's algorithm to search for food/leisure locations. The user can begin using the interactive map to view food/leisure locations.	The database must be populated with data from Yelp's public API.	Launch basic functionalities of the website. Begin creating market awareness and open sign ups for beta testing for the first public release in version 1.0.0.
1.0.0	The first release available to the public. Introduction of user preference will allow the algorithm to generate more well-informed suggestions to the user. User engagement is expected to increase.	The server must be able to begin handling real user traffic.	Launch Go Where GiaGia. Begin collecting bug reports to be studied by the QA Team.

1.1.0	Browse Location page is introduced.	System must ensure both View Map and Browse Location pages are using the same and updated information from the back-end database.	First minor release to users.
2.0.0	New major features are implemented. By this release, users can create accounts on Go Where GaiGai and register or login to the website.	System must be able to handle login sessions for each user on top of guest user traffic.	Project team to continue to hold engagement activities with users to gather ideas for new features.
2.0.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 VCS.	Decrease latency and improve system performance and reliability. Continue to remove bugs and introduce feature improvements based on user feedback.

Table 6: Release Impacts

5.2.4 Release Notification

Whenever there is a new upgrade or improvement of Go Where GaiGai, it is imperative that stakeholders and users of the website are notified. Notifications are sent to allow users to keep track of releases, release phases and release activities. There are three main types of notifications; Information, Warning and Error notifications.

This can be achieved through a detailed release notification mechanism. The release notification mechanism involves the following media components:

- **Push notifications** on the phone for mobile versions of Go Where GaiGai. These are applicable to versions available on the Appstore for iOS and PlayStore for Android devices.
- **In-app notifications** that will remind users when they are accessing the website. In-app notifications will be displayed to users 48 hours before a new release is rolled out, and will also be displayed information about the change in a window the next time a user opens the website after the update.

- **Email and text messages** for users who have subscribed for email and text notifications. This mechanism will only be enabled in future releases where users can create accounts on Go Where GaiGai (software version 2.0.0).

In addition to the end users of the website, members of Team ONE have to be kept in the loop of any approval of upcoming releases and the changes it curtails, especially to the development team. This allows every team member to be aware of new versions, even if they are not directly involved with the implementation of code. The primary mode of communication would be a Kanban Board hosted on Github Project Board. Notifications will be posted on the board in full view of members and team leaders. The full details of the Kanban Board selected can be found in the *Quality Plan* document.

Further information regarding the information provided, such as time frames of notifications will be a shown:

Stakeholder	Content of notification	Timeframe for receipt of notification
Users	Changes made, which include: 1. New features 2. Updating of existing features 3. Warnings for Warning and Error type notifications 4. Information about how new changes will affect the user	Based on the changes: 1. 48 hours before release is rolled out 2. 48 hours before release is rolled out 3. Immediately after the error is detected. Notification will be displayed until a fix is deployed. 4. Notification will be displayed the next time the user opens Go Where GaiGai
Team ONE	Information about system architecture changes, availability of documentation, team members involved in the development of releases. User notifications will also be displayed for team members as above.	Immediately after the approval of new releases, and after the rollout of the new release.

Table 7: Notification Mechanism Structure

6 Glossary

Term	Definition
Agile	An iterative approach to software development
Build-Package-Deploy	A procedure involving building the application, packaging it into a unit and deploying for operational use.
Incremental Development Model	An iterative lifecycle method use in software development
Kanban Board	Agile project management tool
Python	High-level programming language
ReactJS	Front-end Javascript library for building user interfaces
System Architecture Diagram	High level system concept design
Use Case Descriptions	Written description on modules found in use case diagram
Use Case Diagram	Object-oriented graphical representation of a system
Use Case Model	Includes a use case diagram and use case descriptions

7 Acronyms

Acronym	Expansion
API	Application Programming Interface
IDE	Integrated Development Environment
JAR	Java Archive
QA	Quality Assurance
SDLC	Software Development Life Cycle
SRS	System Requirement Specification
SVN	Subversion (Apache Subversion Client)
VCS	Version Control System
WAR	Web Application Resource