Technical Architecture Document

1. Source Code Management
   * Version Control

We plan to use Git and Github as source code management in our project. We created an organization called “TrinityWayFinders” with all team members. A project repository called “WayFinder” was created to store and manage the source code of our project. Each user member should only have reading privilege to the Master branch, changes can only be made through creating new branch then merge into the master branch.

* + Branching Strategy

Git provides a completed branching strategy of source code management. Regarding of a member contributing to this project, a member should create a new branch and put the modification into the branch. The name of branch should be related to the feature that the member is working on.

* + Merging Strategy

Github provides the pull request feature to allow a member to merge the modification with the existing source code in the master branch in shared repository, even without writing privilege of the master branch. The pull request must be reviewed and approved by at least other two team members before the change can be merged into the master branch.

* + Conflict Resolution

Merge conflicts occur when competing changes are made to the same line of a file, or when one person edits a file and another person deletes the same file. When conflict happens, Git keeps two version of change and requests user to manually resolve the conflict. The code should be committed after the conflict resolved.

Git & Github <https://github.com/trinitywayfinders/WayFinder>

1. Document Management

We plan to utilise Github Wiki to manage our document so that users can easily access from a public network. On the other hand, Github Wiki supports Markdown tags and user can format their document.

* Github Wiki <https://github.com/trinitywayfinders/WayFinder/wiki>
* Markdown: <https://www.markdownguide.org/>

1. Plan, Track and Defect Management

As an agile team, we will use JIRA to track the project and teammates. Jira Software is built for every member of your software team to plan, track, and release great software.

Project Track: JIRA <http://34.247.159.151:8080/>

1. Build Strategy

Because our application consisting of few different components with various technologies, we have to choose different build management tools for build each single components.

* + NPM: <https://www.npmjs.com/>

For front end, we use a cross-platform JavaScript framework called “Ionic”, the NPM is default package manager provided by the framework, which can manage node.js package and dependencies automatically.

* + Gradle: <https://gradle.org/>

For the backend we will use spring boot. There are two options for project management, namely gradle and maven. Considering the popularity and accessibility of gradle, we have selected it to manage the Jar packages.

* + Jenkins: <https://jenkins.io/>

We will deploy Jenkins as our build engine on the cloud. It allows members to access and request to build at some points, eg. a new commit are pushed to Github repository. Jenkins also provide many useful plugins to support gradle and npm, which completely fits our requirement.

1. Deployment Pipeline
   * Docker: <https://www.docker.com/>

Freedom of choice, agile operations and integrated container security for legacy and cloud-native applications

* + Docker Compose: <https://docs.docker.com/compose/>

Compose is a tool for defining and running multi-container Docker applications

1. Communication Strategy

We use whatsapp group chat as an instant messaging platform.

1. Tool and Technologies
   * Frontend:
     + Language: Javascript
     + Required run-time environment: Node.js
     + Ionic Framework: <https://ionicframework.com/>
     + Cordova: <https://cordova.apache.org/>
     + IDE: Atom
   * Backend:
     + Language: Java:
     + Spring Boot Framework: <https://spring.io/projects/spring-boot>
     + IDE: Eclipse IDE for JEE: <https://www.eclipse.org/downloads/packages/release/kepler/sr2/eclipse-ide-java-ee-developers>
     + Build Manager: Gradle
   * APIs:
     + OpenAPI 3.0: <https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0.md>
     + Swagger: <https://swagger.io/>;

Swagger standard for OpenAPI 3.0: <https://swagger.io/specification/>

1. Peer Review Process
   * Github branch strategy ensures that every commit be reviewed and approved by at least two team members.
   * Daily Scrum meetings help to keep team on track and highlight any issues/blocks early.
   * Regular meetings ensure every member in the team up to date to the project and every decision is made by vote from the whole team.