**Architecture Documentation**

**Team 1: Precision Beacon Navigation**

**Tahir Aziz**

**Adeel Khan**

**Sabur Khan**

**Alejandro Guzman**

**Casey Boyle**

**2/24/17**

**ABSTRACT**

This document serves as an outline to the architectural documentation that will be used in this project. Overall, this document will introduce this document, outline the basis of our planned architecture and how it works with our application. Additionally, we will outline the architectural model and subsystems along with the technology, software, and hardware used. We will then justify our architectural decision.

**TABLE OF CONTENTS**

* **INTRODUCTION**
* **ARCHITECTURAL STYLE(S) USED**
* **ARCHITECTURAL MODEL**
* **TECHNOLOGY, SOFTWARE, AND HARDWARE USED**
* **RATIONALE FOR YOUR ARCHITECTURAL STYLE AND MODEL**
* **EVIDENCE OF DOCUMENT HAVING CONFIGURATION MANAGEMENT**

**INTRODUCTION**

Our project is being built to solve a specific problem. As such, while coding the solution through the software development lifecycle, we must also consider how we are going to develop the software. This in of itself is another problem to solve. The purpose of this document will be to outline our specific choice of architecture to be used to build our application and to explain why this architectural choice will fit our needs. The scope includes outlining the architecture and why it supports our needs, outlining the architectural model via stereotyping subsystems, describing the technology, hardware, and software used, and the justification for our architectural choice. The scope does not include detailed design such as defining interfaces and classes. The scope of this document covers the high level architecture of the application under development.

**ARCHITECTURAL STYLE USED**

We chose the MVC framework (Model, View, Controller) for our project. The application is modularized into three main components: model, view, and controller. The model is where the data is stored, the view is where the user sees and provides inputs to the interface, and the controller handles the logic coming from user inputs in the view to any changes in the model that occur as a result of those inputs. The main feature of our application will be navigating a user along a path and checking if they are on track by trilaterating their position via bluetooth beacons. Thus, the main functionality involves some sort of logic updating the view and the data when new signals are read in. As such, MVC supports the main functionalities of our application. Other features involved providing feedback based on the user being on or off the path, which also falls in with the MVC framework since the view can be updated with the feedback based on the user input (the trilaterated user location in this case). The main category of functionality involves displaying information to the user based on what the user is doing, so the MVC makes sense because we can have the view, a controller to handle the input, and the model for the data.

**ARCHITECTURAL MODEL**

System: Android application navigating a person between 2 points in a building where the navigation is checked with bluetooth beacons to locate the user.

Subsystems:

* Beacons
* Navigation
* User View
* Database
* Controller

**TECHNOLOGY, SOFTWARE, AND HARDWARE USED**

Technology: Beacon navigation, wayfinding

Software: Android Studio, Git/Github, Java, SQLite

Hardware: Aruba Bluetooth Beacons

Communication between application server and database server: Since this is an Android application, the app will be run on the user’s device via the OS, and the database will be stored within the application. Since we are doing the MVC pattern, the controller/logic will handle interactions with the database, and those updates will be reflected in the application view.

**RATIONALE FOR YOUR ARCHITECTURAL STYLE AND MODEL**

There are many different types of architectures, we had the opportunity to look through the different aspects of our application and choose an appropriate architecture, we first looked at client/server architectures and understood our app doesn't truly require a lot of security as there won't be a login system, or too much sensitive data being shared, so we crossed that one out, then we then looked at component based architectural styles this model is better for games that will constantly need reusable features, and are not context specific. Our application will not need to reuse a lot of features as each map will have a different path for example. So in the end we ended up choosing kind of a mixed model using the MVC model gives us the range that we need to show how the data is stored, a view to help indicate how the software will be viewed and the controller which will be the main connection of how the database interacts with the user view.

**EVIDENCE OF DOCUMENT HAVING CONFIGURATION MANAGEMENT**

This document was modified from its template version by copy/pasting it to Google Documents where the team could edit it together to fill in the necessary information for the architecture documentation.