Department of Computer Science & Engineering
The University of Texas at Arlington

# Detailed Design Specification
# CSE 4317: Senior Design II
# Spring 2020

Team Infrared
Infrared Arena

Edgar Acevedo
Justine Batongmalaki
Linda Phanvilay
Danny Vu
Jean-Marcel Yacho

## Revision History

| Revision | Date | Author(s) | Description |
|---|---|---|---|
| 0.1 | 11.08.2019 | LP | document creation |
| 0.2 | 03.02.2020 | LP, JB, JMY, DV, EA | complete draft |
| 0.3 | 05.13.2020 | LP, JB, JMY, DV, EA | final draft |

## CONTENTS

**1 Introduction** **5**

**2 System Overview** **5**

**3 Tagging Layer Subsystems** **6**
3.1 Layer Hardware . . . . . 6
3.2 Layer Software Dependencies . . . . . 6
3.3 Emitter Subsystem . . . . . 6
3.4 Sensor Subsystem . . . . . 6

**4 Server Layer Subsystems** **8**
4.1 Layer Operating System . . . . . 8
4.2 Layer Software Dependencies . . . . . 8
4.3 Cloud Subsystem . . . . . 8
4.4 Database . . . . . 8

**5 Client Layer Subsystems** **10**
5.1 Layer Hardware . . . . . 10
5.2 Layer Operating System . . . . . 10
5.3 Layer Software Dependencies . . . . . 10
5.4 User Interface . . . . . 10

**6 Network Layer Subsystems** **12**
6.1 Layer Hardware . . . . . 12
6.2 Layer Operating System . . . . . 12
6.3 Layer Software Dependencies . . . . . 12
6.4 Bluetooth Subsystem . . . . . 12
6.5 Internet Connection Subsystem . . . . . 12
6.6 GPS Subsystem . . . . . 13

## List of Figures

1 System architecture . . . . . 5
2 Emitter data flow diagram . . . . . 6
3 Sensor data flow diagram . . . . . 7
4 Cloud data flow diagram . . . . . 8
5 Database data flow diagram . . . . . 9
6 User interface subsystem description diagram . . . . . 10
7 Bluetooth subsystem description diagram . . . . . 12
8 Internet subsystem description diagram . . . . . 13
9 GPS subsystem description diagram . . . . . 13

## List of Tables

## 1 Introduction

Infrared Arena will allow users to host their own private laser tag matches with their friends. The system will consist of a laser tag gun which contains an emitter, and an app user will download on their smartphones. The laser tag gun contains a phone mount for the user to attach during game play. The app is created using Unity The app will allow users to create and join multiplayer matches which will be hosted on a Firebase cloud services. The location of player's will be obtained using Google Map API. The intended audience for this system is users 12 and up. The app and hardware will be available commercially to the public. A list of all requirements is contained in the System Requirement Specification documentation.

Infrared Arena's architecture has consists of a network, server, tagging, and client layer. Client Layer includes the user interface subsystem, which is the way the user communicates with the app. Network Layer contains the Bluetooth, GPS, and Internet Connection subsystems which is all the connections . Tagging Layer contains the Emitter and Sensor subsystems which is how the laser gun detects actions from another laser gun. The Server Layer contains the Cloud and Database subsystems which is how the game server will be implemented. Further explanation of the architecture is specified in the Architectural Design Specification documentation.

## 2 System Overview

Infrared Arena will consist of 4 layers; the client layer, server layer, network layer, and tagging layer. A simple diagram depicts the system below. In the following paragraphs, each layer's purpose shall be described in summary.

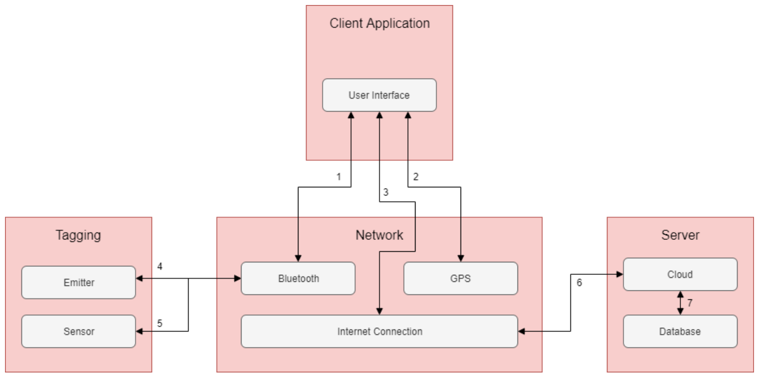

Figure 1: System architecture

The Client layer represents the smartphone being used to play Infrared Arena. Its only subsystem is the UI of the mobile application which will allow users to play laser tag matches with one another. The Server layer contains 2 subsystems, being the Cloud and the Database subsystems which manage the system's computing power, resources, and data that's stored and transmitted between layers. The Tagging layer consists of the Emitter and Sensor subsystem. The Emitter subsystem is used to target and tag players, while the Sensor subsystem is used to receive the tags. The Network layer manages the connection between all 4 layers, so that data can be transmitted between them all.

## 3 Tagging Layer Subsystems

The tagging subsystem is responsible for targeting/sensing hits from players/enemies. The hardware component is the infrared laser and sensors on the blasters. The software used is the bluetooth low energy which will allow connection between the blasters to the client.

### 3.1 Layer Hardware

The hardware used is the Recoil Starter Kit which contains the RK-45 Spitfire Blaster and the SR-12 Rogue Blaster.

### 3.2 Layer Software Dependencies

Software dependency is the bluetooth low energy.

### 3.3 Emitter Subsystem

The purpose of this subsystem is to target and hit/score the enemy players from the infrared blasters.

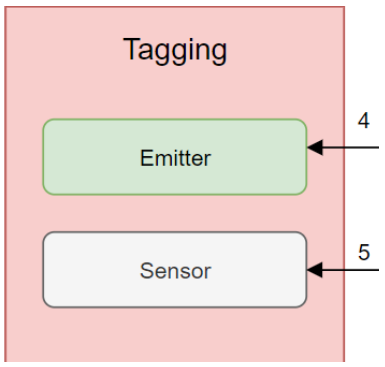

Figure 2: Emitter data flow diagram

#### 3.3.1 Subsystem Hardware

This uses the blasters infrared laser.

#### 3.3.2 Subsystem Software Dependencies

Software dependency is the bluetooth low energy.

#### 3.3.3 Subsystem Programming Languages

Java for bluetooth low energy.

#### 3.3.4 Subsystem Data Structures

In this subsystem, the blaster sends information to the client.

### 3.4 Sensor Subsystem

The purpose of this subsystem is the receive damage taken from getting hit by the infrared blasters.

#### 3.4.1 Subsystem Hardware

This uses the sensors located on the blasters.

#### 3.4.2 Subsystem Software Dependencies

Software dependency is the bluetooth low energy.

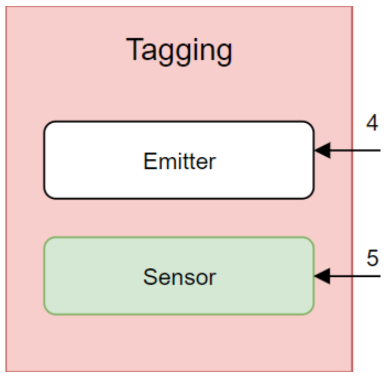

Figure 3: Sensor data flow diagram

### 3.4.3 Subsystem Programming Languages

Java for bluetooth low energy.

### 3.4.4 Subsystem Data Structures

In this subsystem, the blaster sends information to the client.

## 4 Server Layer Subsystems

### 4.1 Layer Operating System

The layer should be operating on a Android OS. There is on specific operating system which the cloud server operates on.

### 4.2 Layer Software Dependencies

Firebase SDK for Realtime Database API and Firestore API REST API from Unity asset store Google Play Services JSON

### 4.3 Cloud Subsystem

The cloud subsystem will utilize Firebase services to build a real-time server for the game to be based off of.

Additional research was done into Azure's PlayFab and DigitalOcean servers, but due to time constraints and issues regarding COVID-19, the team switched to using Firebase services in the mid Spring semester.

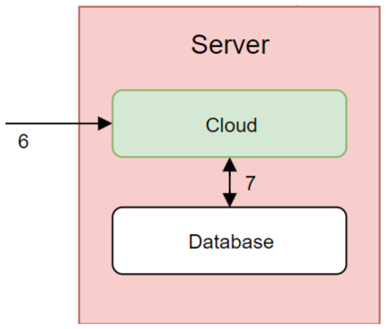

Figure 4: Cloud data flow diagram

#### 4.3.1 Cloud Operating System

No operating system

#### 4.3.2 Cloud Programming Languages

C# NoSQL

#### 4.3.3 Cloud Data Processing

Firebase services use a cloud-hosted NoSQL database that syncs between users. Data is stored as JSON and synchronized in realtime to every connected client. Clients will share one Realtime database and should be able to receive updates automatically. The steps to processing are to integrate the required Firebase SDK's, create the references to the JSON data, set data and listen for changes and then get data. [3] Firebase Authentication will allow a way to secure information.

With DigitalOcean API, DigitalOcean resources programmatically using conventional HTTP requests. Data is collected in the cloud, received through transmissions from the Network layer. After processing, the data is stored in the Database for future use.

### 4.4 Database

#### 4.4.1 Database Operating System

The database does not run on a specific operating system but will run on Android OS for this project.

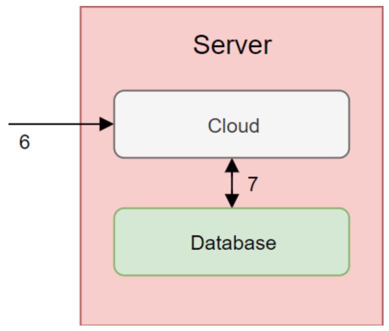

Figure 5: Database data flow diagram

#### 4.4.2 Database Software Dependencies

We are using Firebase Realtime Database and Cloud Firestore services to manage our database and storage.

#### 4.4.3 Database Programming Languages

NoSQL Expression-based rules language

#### 4.4.4 Database Data Processing

Firebase uses data synchronization so the information in stored while offline and automatically synced when connection is regained. The rules set in the Firebase console define how data is read or written to. Data is transferred to the cloud, received through transmission from the Network layer. After processing, the data is stored in Database for future use. [2]

# 5 Client Layer Subsystems

## 5.1 Layer Hardware

The only hardware for this layer is the user's smartphone.

## 5.2 Layer Operating System

Infrared Arena will run on Android OS.

## 5.3 Layer Software Dependencies

This layer was created using Unity along with the Mapbox SDK.

## 5.4 User Interface

The user interface for Infrared Arena is the mobile application. The app will be used to display the current state of the game, as well as take input from the user and the laser tag gun.

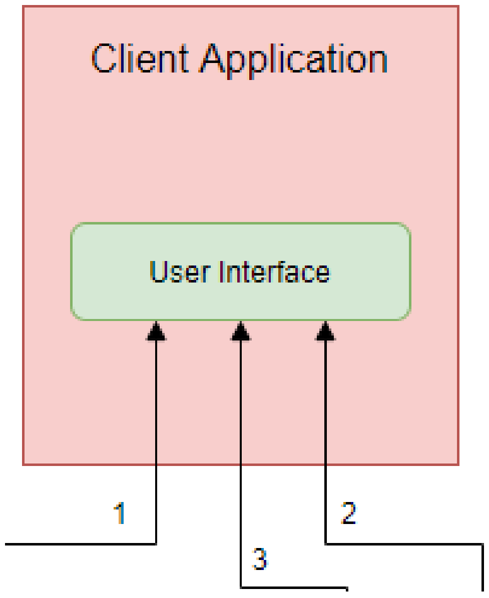

Figure 6: User interface subsystem description diagram

### 5.4.1 Subsystem Hardware

The only hardware that is required for this subsystem is the customer's smartphone which should come with GPS, internet connection, Bluetooth, and WiFi capabilities.

### 5.4.2 Subsystem Operating System

The application is currently being developed for Android OS so the customer will be expected to have an Android smartphone. Specifically, the customer will be required to be running a smartphone with Android 8.0 or newer.

### 5.4.3 Subsystem Software Dependencies

We are building our app using Unity so any packages or software dependencies we have would be built in Unity.

### 5.4.4 Subsystem Programming Languages

The application is currently being developed on the Unity platform which uses C# as its object-oriented scripting programming language.

## 6 Network Layer Subsystems

The Network Layer subsystem is in charge of data connection between the Client, Tagging and Server subsystems. Input into the Network layer is through Bluetooth connectivity from the Tagging subsystem to the Client subsystem. The Tagging subsystem determines if user is tagged. The Client subsystem sends data through a internet connection to the Network subsystem which goes to the Server subsystem.

### 6.1 Layer Hardware

Android Mobile Phone

### 6.2 Layer Operating System

Android

### 6.3 Layer Software Dependencies

Unity Libraries Firebase SDKs

### 6.4 Bluetooth Subsystem

The Bluetooth subsystem will serve as the connection between the phone and hardware system to indicate the changes from the hardware system into the application.

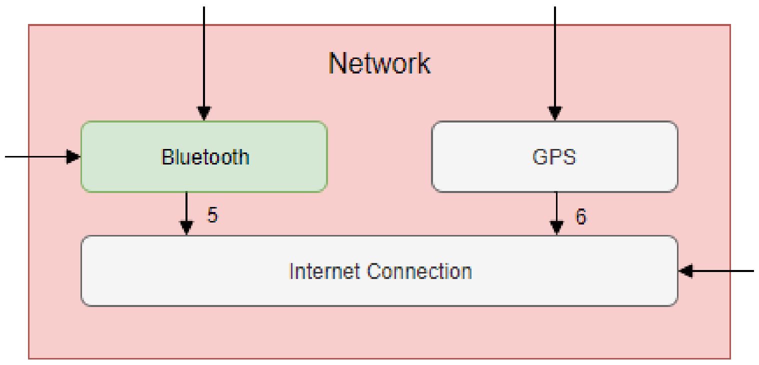

Figure 7: Bluetooth subsystem description diagram

#### 6.4.1 Subsystem Hardware

Bluetooth adapter and receiver which is built into the phone and Recoil gun.

#### 6.4.2 Subsystem Programming Languages

C# Java

#### 6.4.3 Subsystem Data Structures

Bluetooth data is transmitted in packets that put into divided slots from the wireless link between the devices. [1]

### 6.5 Internet Connection Subsystem

The Internet Connection subsystem of the Network Layer's purpose to send the data from the Client Layer to the Network Layer which then goes to the Server Layer from which the client will use to play the application.

#### 6.5.1 Subsystem Hardware

Wireless

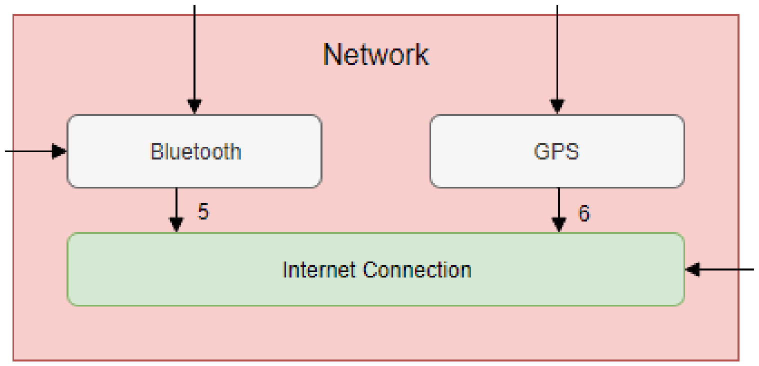

Figure 8: Internet subsystem description diagram

#### 6.5.2 Subsystem Software Dependencies

Depends on the Android phone system software to connect to Internet Global System for Mobile Communications (GSM) Code Division Multiple Access (CDMA)

#### 6.5.3 Subsystem Programming Languages

C#

#### 6.5.4 Subsystem Data Structures

N/A

#### 6.5.5 Subsystem Data Processing

This is done externally by the cellular service providers Internet connection systems through radio waves

### 6.6 GPS Subsystem

The GPS subsystem of the Network Layer's purpose is to update the location, direction, and position of the players in the game.

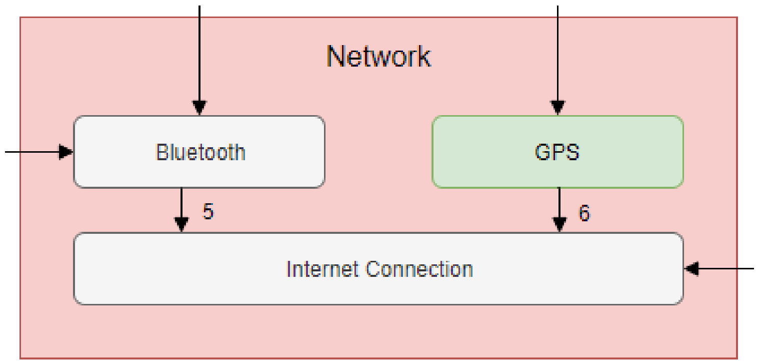

Figure 9: GPS subsystem description diagram

#### 6.6.1 Subsystem Hardware

GPS receiver, or capability to use triangulate location from signals.

#### 6.6.2 Subsystem Software Dependencies

Mapbox (https://www.mapbox.com/)

#### 6.6.3 Subsystem Programming Languages

Mapbox uses C#

#### 6.6.4 Subsystem Data Processing

Location wil be sent in the server and shared between users

## References

[1] Scientific American. *How does Bluetooth work?* https://www.scientificamerican.com/article/experts-how-does-bluetooth-work/.

[2] Google Developers. *Cloud Firestore*. https://firebase.google.com/docs/firestore/how_does_it_-work.

[3] Google Developers. *Cloud Realtime Database*. https://firebase.google.com/docs/database/.