Smartphone Tracking App for Microsoft HoloLens

Design Document

Team Number
27
Client
Optical Operations
Adviser
Dr. Qiao

Team Members/Roles
Travis Harbaugh/HoloLens Development
Ben Holmes/Android Development
Anthony House/Website Development/Security
Cory Johannes/Report Management
Jose Lopez/Website Development
Ryan Quigley/Android Development

Team Email
sdmay19-27@iastate.edu
Team Website
http://sdmay19-27.sd.eve.iastate.edu/team.html
Revised
October 30th, Version 1

List of Figures

List of Tables

List of Symbols

Not Applicable

List of Definitions

Not Applicable

1 Introductory

1.1 Acknowledgement

Sdmay19-27 would like to thank Andrew Guillemette and his engineers from Optical Operations for their contribution to our project. Andrew contributed significant assistance by providing our team with technical advice and resources.

1.2 Problem Statement

General problem

Work surveillance is difficult in large scale construction sites. These sites have many different zones with no way to manage workers in 60 story buildings. These sites can be very dangerous and there is no real time tracking solution that can manage the logistics. GPS accuracy is within 3 meters and doesn't always work indoors because there is a lot of interference with steel, concrete, and large objects (skyscrapers). Many large tech corporations have tried to improve indoor navigation but have been unable to find a scalable solution. If a feasible method for tracking workers is realized, it would provide a way for managers to monitor their employees and be able to better update their clients on process.

General Solution Approach

The purpose of our project is to make an application that can track an individual indoors within 1 meter as a proof of concept so our client can create a prototype that can be used as a construction site tracking solution.

To create our solution, we will be developing mobile/HoloLens software application that will complement each other. The mobile applications primary responsibility will be to track a user within a 1-meter indoor/outdoor environment. To achieve this requirement, we are going to create a smartphone application which will unitize the phone's accelerometer, magnetometer, and gyroscope to update changes on the phone's movement, and relay the information to a server.

The accelerometer sensor is used to determine the change in acceleration which will be used to find the users location. The accelerometer will also be used for step detection and will estimate the length of user's footstep in real-time. The magnetometer, and gyroscope will be used to determine the change in the device orientation which will give you the user's direction and

heading. With these three sensors we will be able to calculate the user's location and relay the information back to the database that will be used by the HoloLens application.

The HoloLens' primary responsibly is to retrieve the user's location stored in the database recorded by the mobile device and then render the new position of the user's avatar in the augmented environment on the HoloLens. To achieve this requirement, we are going to create a Unity application that maps an environment to a 3D model of a particular jobsite. We will then get 3D avatar models on the unit that will simulate workers on the website and simulate the user's movement throughout the work day.

Project Motivation

What is driving this project is that there is no real solution that provides any surveillance onsite. There is a lot of construction that goes into creating huge skyscrapers using steel, concrete, etc. There are many different zones, machinery, and infrastructure that need to be monitored to provide safety for workers. According to United States Department of Labor, 4,190 workers were killed in 2016.[1] They stated that 1 out of 5 workers deaths last year were in construction. We need a solution that can provide information on what goes on at these construction sites to provide better safety at the work site.

Project Overview

The Project is a mobile indoor tracking solution utilized on a construction job site that will allow you to track an individual's movement either indoors or outdoors. The construction workers will be tracked by using the phone's accelerometer, magnetometer, and gyroscope sensor that will update the user's position and relay it to the server. From the office we will use the HoloLens to display a 3D model of the jobsite. Within the model there will be 3D avatars that represent construction workers that allow the office employees to monitor their position at the job site.

Project Outcome

Our team would like to complete an Android application which will unitize phone sensors to predict an individual's location to within 2-meter accuracy and store it on our database. Next, we would like to develop the HoloLens application that will take the new position of the user and use augmented reality to display their position in a 3D environment in real time. The outcome of this project would be update the workers position and display it in real-time on the HoloLens.

1.3 Operating Environment

Our operating environment is going to be for a construction site. Our end product will be using a mobile application for gathering a user's location without the use of GPS. We will be using a

combination of the phone's sensors and bluetooth beacons in order to accomplish this. Due to extreme weather and dangerous conditions our device must stay secured in a specific location and orientation on the user's body. A suitable position will not impede the workers ability to perform on the job site.

The user must wear this device at all times while in the designated work zone. While operating, our application will collect data from the phone's sensors, and use heading/distance prediction estimates to map the user's trajectory into a 3D representation of the work zone. In addition, this application will be able to store current location estimates on the Android phone's file system when an established internet connection is not available. As a final requirement, the application will limit battery usage when possible, to achieve battery life durations of 8 hours or more.

1.4 Intended Users and Intended Uses

Our intended user base is construction workers. Our smartphone application will track its users using a combination of the Android phone sensors, bluetooth, and sound. Doing so will provide a new level of supervision over construction projects. It is our hope that this additional level of supervision can be useful in solving logistics issues. It will also provide a record of daily work, which will detail the productivity and current progress of the construction project.

1.5 Assumptions and Limitations

We assume that all equipped phones have the following hardware capabilities:

1. Magnetometer
2. Gyroscope
3. Accelerometer
4. Bluetooth 4.0
5. Wifi chip
6. Speaker & Microphone

We also assume that the audience will be either walking or running when wearing the device. Our proposed solution is not meant to track people riding in vehicles.

To begin with, in order to ensure the accuracy of our tracking algorithm, we will require that users hold their phone vertically in front of their bodies so that the top of the phone points in the direction of motion. Eventually, as our methods improve, we will require that users wear the phone on an armband strapped to their right arm, again oriented so that the top of the phone

points in the direction of motion. We are also limiting the environment to Durham. Aside from this, the application is expected to work regardless of a user's trajectory, or type of movement.

1.6 Expected End Product and Other Deliverables

The end product will be an Android application with the following functional requirements:

1. Will track a user within an accuracy of 1 meter
2. Will run on all Android phones which have API 21 (Lollipop) or higher
3. Will use the phone's Magnetometer and Gyroscope to determine the direction of motion
4. Will use phone's accelerometer to measure movement
5. Will use the Bluetooth sensor to detect re-calibration points and reset the users location
6. Will use RSSI values obtained from WiFi access points to infer location
7. Will include a web interface which maps the phone's movement into a 2D and 3D representation of Durham

2 Specification and Analysis

2.1 Proposed Design

2.1.1 Concept Diagram

Figure 1. Concept Diagram

Office Monitoring

From the client's office a supervisor will be sitting in the conference room with the Microsoft HoloLens monitoring the jobsite. The HoloLens renders the construction site, buildings, vehicles, and employee avatars in augmented reality through a connection with the Rethink database. Once the Rethink database receives a location update, Rethink will automatically

notify all change listeners attached to the given data, and transmit the information through the corresponding sockets (to the HoloLens). The HoloLens will use an access point from the office to receive this transmission.

Construction Site

Before the construction workers enter the work site, the Android application will calibrate to the user's specific orientation and step size. The phone will be attached to the user's right arm using an armband and must be worn at all times during construction hours. As the construction workers walk around the site, the mobile application will record data from the phone's sensors. The data collected from these sensors will then be used to estimate the user's new position in real-time. Finally, these position estimations will be sent to the server in order to be saved in the database.

2.1.2 Architecture Diagram

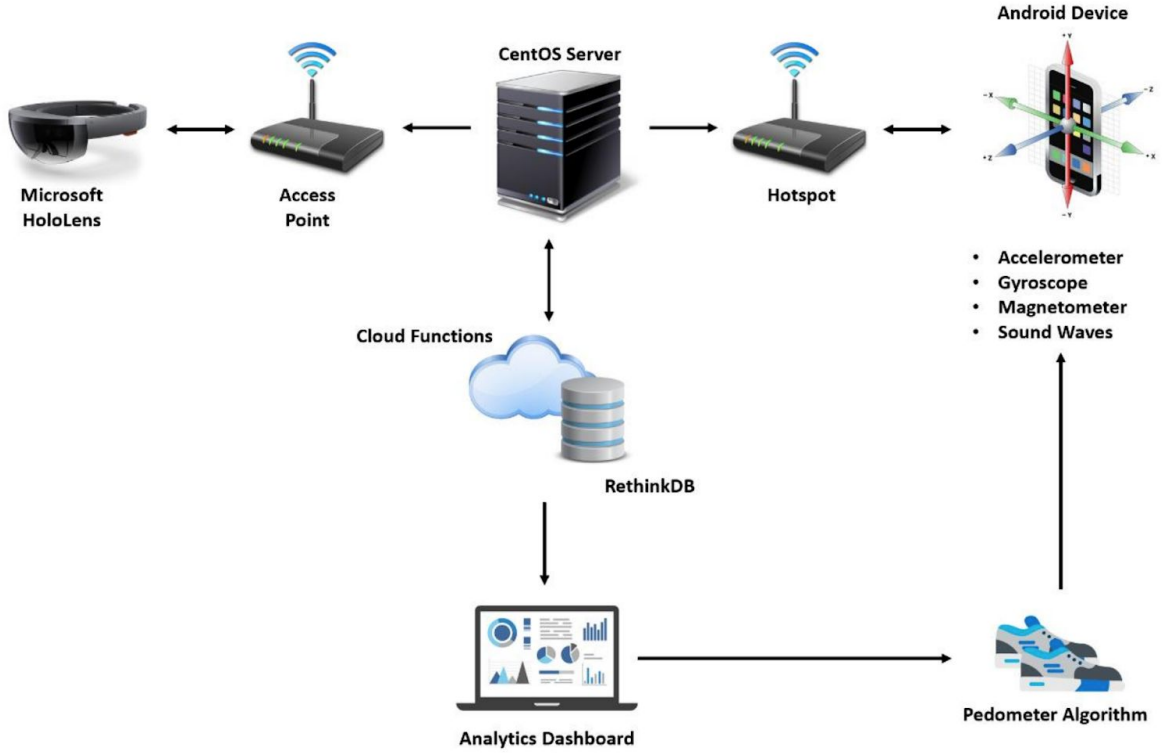


Figure 2. Architecture Diagram

Website

The website receives accelerometer, magnetometer, and gyroscope data from the Android phone, saves the data in the database, and updates corresponding user markers representing the devices being tracked. The website will maintain a display which moves these markers through a 2D representation of the worksite (Durham) and trace the paths as they develop.

HoloLens

The HoloLens connects to the office access point in order to receive updates from the server. The server sends a notification when new location data is received from the mobile application. The HoloLens then takes this location data from the database and renders 3D avatars representing the current positions of all devices being tracked.

Android Application

The Android application detects user movement via the phone's sensors. The application determines if the user has moved from his/her current position, and if so, makes a new position estimation based on both the previous known location, and the collected data. The application then connects to an access point within the construction site (Durham) and sends this new position estimate to the server in order to be saved in the database.

Database

The database we are using is RethinkDB. Our schema will store 3D accelerometer data and position coordinate data. When the database receives a request from the server to store a new user position or piece of data, the database updates its storage and broadcasts the new data to the Microsoft HoloLens.

2.1.3 Process Flow

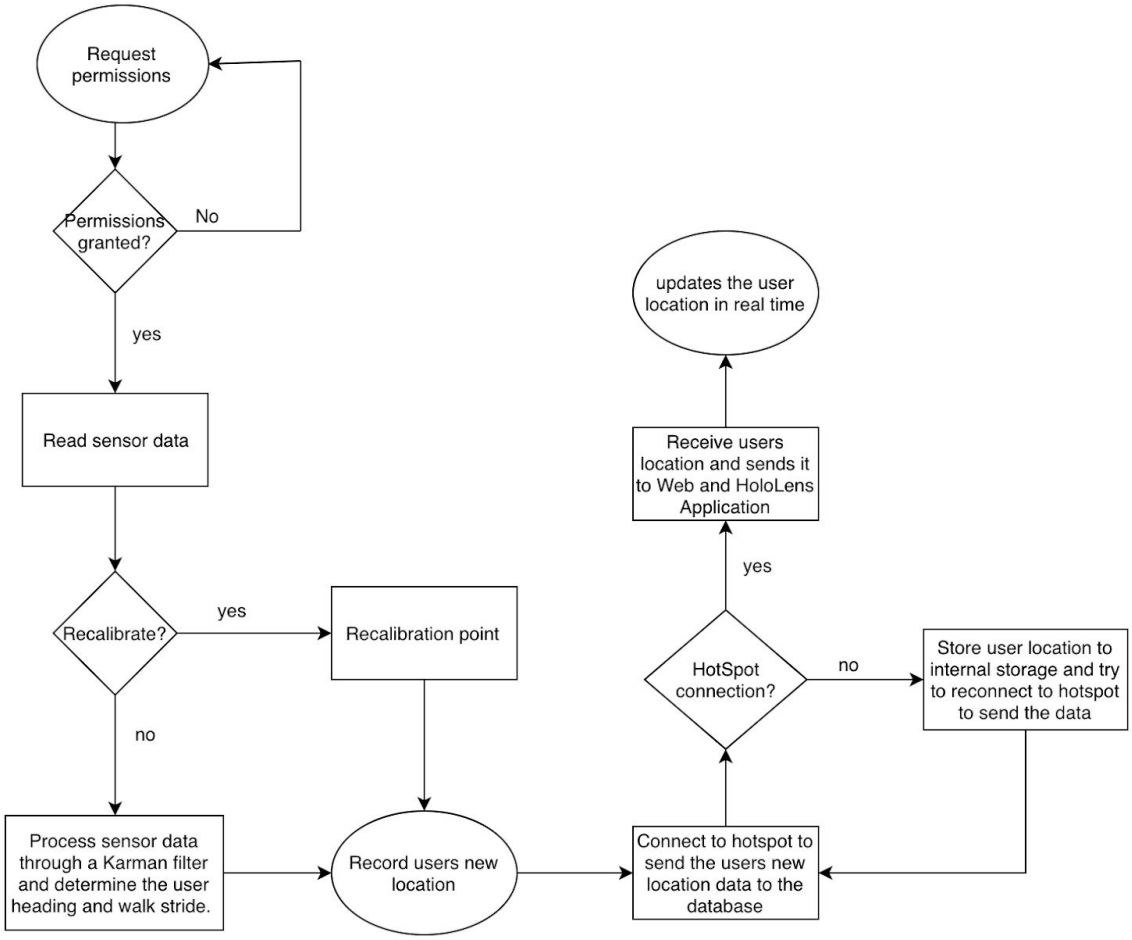


Figure 3. Process Flow Diagram

When the android device application is executed the user will be asked to accept permissions that will allow the software to track the user's location. If the user doesn't grant the software permission, then the application will request the user to grant the application permission until either they accept or kill the application. If the user grants the application permission it will then start to read the hardware sensors on the device. It will then scan the area for a recalibration point. If it finds a recalibration point within 1 meter of the device, then it will receive the recalibration location from the beacon and then record the user new location. But if there is no beacon nearby then it will process the data gathered by the hardware and then calculate the user's new location and record it. After the application has gathered a new user location it will try to connect to a WiFi hotspot. If the application successfully connects, it will send the location data to the database. If the application doesn't successfully connect to the hotspot it will then store the new user location to the hardware internal storage and try to connect to the hotspot again. After

the database receives a new user location it will send it to the web and HoloLens application. After the web and HoloLens receive an update from the database it will update the user's location in their mapping environment.

2.1.4 Mobile Application

Java

Java is the primary object-oriented language used to develop Android applications. Our solution will utilize all 3 of the above concepts to create an Android application for tracking user movement. Our app will be written with Java and support devices operating on android API 21 or above. We will use Java to communicate with the Android API which in turn will communicate with the Hardware Abstraction Layer (HAL) and send us sensor data.

GPS

Primarily this sensor will not be used except for an initial position determination. Before entering a work site, GPS would be used to collect the latitude/longitude coordinates of the current user. This latitude longitude value will then be sent to the website, and mapped to a specific location in our 2D visual representation of the work site. Note, the website will display a 2D representation of the work site, while the HoloLens will provide a 3D representation.

Accelerometer

The Accelerometer sensor is used to measure acceleration. Android phones primarily use piezoelectric accelerometers. These types of accelerometers are built using crystal materials (usually quartz) which generate an electric charge when squeezed. This property makes it possible to infer the relative acceleration by measuring the corresponding changes in the electrostatic field. We will use the Accelerometer for step detection in our Pedestrian Dead Reckoning (PDR) system.

Gyroscope and Magnetometer

Typical phone gyroscope sensors are implemented with microelectronic mechanical systems (MEMS). The point being, they are hardware based, and not software based. The gyroscope is used to determine the cellphones orientation. Changes in a phone's orientation will be measured as angular velocities. We can then integrate these velocities to find displacement.

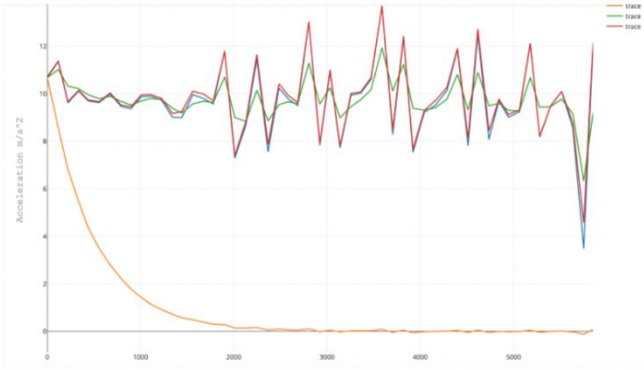
Bluetooth Beacon

The bluetooth beacon will emit an advertising packet to transmit data to the android device. The android device will scan for beacons within range of the device. It will then establish Radio

Frequency Communications (RFC) and receive the advertising packet from the Bluetooth beacon if it is within 1 meter. We can calculate the distance by using the TX power level that is transmitted from the beacon.

Step Tracking Algorithm

To determine a user's location, we have to create an algorithm that can determine the distance a user travels based on their stride. We have two thresholds that will determine if the (x, y, z) accelerometer data is a peak or valley. The peak is the maximum value that passed the max threshold. After the peak is detected and the accelerometer data is decreasing, and it goes below the minimum threshold, then we know that the user has walked a step. We have three vectors. We have a start point, peak, and end point for our algorithm. The distance (foot stride) can be calculated by computing the difference between the start and end point.



Magnitude values for 3D accelerometer data collected while walking 10 steps

Basic Algorithm

```
max = 11
min = 9
peakDetected = false
currentPeak = -1
stepCount = 0;

for item in data:
        if (item >= max)
                peakDetected = true
                currentPeak = item
        if(peakDetected)
                if(item > currentPeak)
                        currentPeak = item
                else if (item <= min)
                        stepCount++
                        peakDetected = false
                        currentPeak = -1
```

Figure 4. Step Tracking Algorithm

Heading Algorithm

The heading determination algorithm serves an important part of our Pedestrian Dead Reckoning system. We are currently using the Android based Rotation Vector Sensor, which creates quaternion estimations of a phone's orientation based on a sensor fusion between the Gyroscope and Magnetometer. To improve our accuracy we can eliminate the Rotation Vector Sensor, and instead make our own quaternion estimations by reading the output from the Gyroscope. Essentially we will start with a current orientation quaternion, which we may either hard code as a default starting orientation, or estimate based on the magnetometer and accelerometer readings.

Next we will use the Gyroscope outputs to create a rotation quaternion, which we can add to our current orientation quaternion in order to get a new orientation.

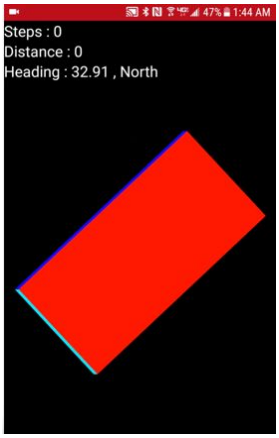


Figure 5 on the left shows a representation of a phone in 3D space. The phone is a rectangle with color coded sides (red is front). We created this simple OpenGL representation to aid in our understanding of quaternion rotations.

Figure 5. OpenGL Visualization

Bluetooth Beacon Algorithm

To re-calibrate a user's position our software will determine if the android device within 1-meter of the Bluetooth beacon. The txPower is the strength of the broadcasting power level constant that is determined by the advertisement protocol. The RSSI value is determined by the Strength of the RSSI value which transmits from the Bluetooth beacon. This algorithm is used to determine when the android device is within 1 meter from the Bluetooth beacon. This algorithm is needed to recalibrate the user's position due to error associated with the sensor readings. Figure 6 (below)

```
calculateDistance(int txPower, double rssi)
    if(rssi == 0)
        return -1
    ratio = rssi * 1.0/txpower
    if(ratio < 1.0)
        return ratio^10
    else
        accuracy = (0.89976)*(ratio^7.0795) + 0.111
        return accuracy
```

Figure 6. Bluetooth Beacon Algorithm

2.1.5 Database

RethinkDB is a scalable, open sourced database for Real-Time applications solutions. This database schema offers a unique solution over traditional database schemas such as MySQL.

RethinkDB allows objects to register on change listeners to a specific data set. It is a classic Observer pattern, in which database changes will automatically trigger pre-defined updates. We will utilize this functionality to automatically update HoloLens data whenever a new location coordinate is saved.

RethinkDB is different from other real-time databases such as Google's Firebase or Amazon's Relational Database service because it is open sourced, and does not restrict on storage size or daily limits. Some of the key functionalities to RethinkDB are outlined below.

1. An advanced query language that supports table joins, subqueries, and massively parallelized distributed computation.
2. An elegant and powerful operations and monitoring API that integrates with the query language and makes scaling RethinkDB dramatically easier.
3. A simple and beautiful administration UI that lets you share and replicate data.

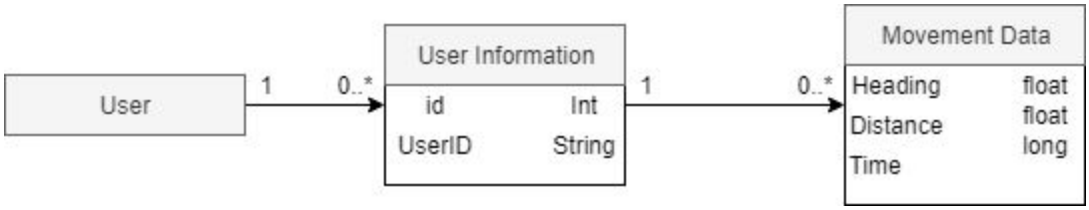


Figure 7. Database Schema Diagram

User Information

The user's id and UserID are stored for unique identification. id is stored as an integer for simple comparison when dealing with multiple users. The UserID however, is a string for readable identification of the user.

Movement Data

Within the Movement Data, three values are stored. Heading is stored as a float that represents the direction the user is facing on a 360 degree scale. Distance is the estimated step size of the user that is also stored as a float. Time represents what time, in milliseconds, that the data was collected on the app; stored as a long, this data can be interpreted to obtain the date and time.

2.1.6 HoloLens

Camera Scripts

The building zone camera script will allow office personal to monitor different zones of the jobsite by using an xbox controller, hand gestures, or voice commands. The 1st person/3rd camera script will allow the office personal to monitor a worker from the worker's perspective by

switching to the avatar representing the worker and toggling between 1st or 3rd person camera angles. Reference the Camera controller script for more information.

Worker Movement Scripts

The player control script will allow the avatar to move from the current position to its new position received from the mobile application and then translate their movement in the HoloLens augmented relativity environment where it can be monitor by the client. When the database script receives a response from rethink database it will send new location information to the Software controller interface Script and determine what avatar needs to be retranslated and send the movement script the new coordinates. The movement script will translate the current GameObject to be moved from its current position to its new position.

Particle System Script

The particle system script will render some special effects when an avatar hits an object (wall, vehicle) to notify the supervisor that a collision has occurred. When the avatar collides with game object the game engine will render a particle that will be displayed in the augmented reality. The collider detection script will detect when the avatar is collided with its surrounds (vehicle, Building) and send the information to the Software controller interface Script which will relay the message to the particle system script which will then render the particle effect in augmented reality. This could be useful after an accident to review the events leading up to it.

Animation Script

The animation script will animate (idle state, walking state, running state, etc.) objects in the HoloLens depending on their perceived state. We will create animation clips for the avatar objects. Each state will have a transition to the next animation state. Each avatar instance will have an animation script that is an instance to the animator controller component that controls which state the avatar is currently in. The script will pass in the animation instance variables that controls which state is active. This script will be attached to the avatar game object and the state will be controlled by the data received from the mobile application.

Collider Detection Script

The collider detection script will detect when a worker or vehicle has collided with an object on the job site. This script will inform the mobile application if the worker avatar has hit a wall or some other structure. The avatar will have a rigid body component attach to its game object that has collision detection by adding a collider component to the avatar game object. By adding a collider component with the rigid body, we call detect if a collision has occurred. The colliders will be attached to the Durham 3D model and when the avatar collides with the building it can be

detected by using Raycast. A Raycast is a ray that send out in space that detects if a GameObject is collided with any other GameObject . This script will be attached to the avatar game object.

UI Menu Input Scripts

UI menu scripts will receive finger gestures and voice commands. These commands will be used for switching between zones and view perspectives. The UI Menu script will receive input from user wearing the HoloLens. The Software controller interface Script will receive the input and send it to the UI interface controller. The UI interface controller will have an array of UI components that implements the Menu UI interface and will call the interface method. Each component of the UI menu will perform the operation that is requested from the user.

Database Script

The Database script will be used to send and receive data from the database. If the database sends data to the HoloLens, this script will parse the data received, and send it to the player or vehicle script in order to render the appropriate translation. If the Database Script receives date from the rethinkDB it will parse the data and send it to the Software controller interface script. The software interface script will send that data to either the avatar or vehicle script and it will translate the correct game object to its new location.

2.1.7 Modeling

3D Modeling

The HoloLens renders 3D models of the worksite in augmented reality using the Mapbox SDK, a mapping platform for Unity3D. Our prototype involves tracking user locations inside of Iowa state campus buildings as a proof of concept. To be able to monitor individuals on the HoloLens we need to have 3D model of the campus building so that it can be rendered in augmented reality.

For location information about the campus building about the campus building our team visit the Iowa state website www.fpm.iastate.edu. The site helped us locate a link to 3D SketchUP models from Trimble Warehouse that had 3D models of the campus buildings. The model of each building is in a .skp file format along with a texture pack for each campus building.

Unity requires that the 3D model be in the format of .obj or .aed to import it into its IDE. SketchUP Pro offers a converter that can convert .skp files to .aed and .obj model. Once a model file is converted and loaded into Unity3D, it can take the texture files and apply them to the polygons of the 3D model. The model can then be geolocated to the exact location on the map platform to be rendered in augmented reality.

To be able to monitor individual workers and vehicles on the job site we need to create some vehicle and character models. Unity offers these models on the asset store and can be purchased and imported into our project.

Figure 8. 3D Model of Durham

2.1.8 Web

We will be using a web platform to show a 2D rendering of Durham, and the recorded trajectories of our 3 devices. The technologies we will be using include, but are not limited to, HTML, CSS, Canvas, JavaScript, and JQuery.

Front-End Architecture

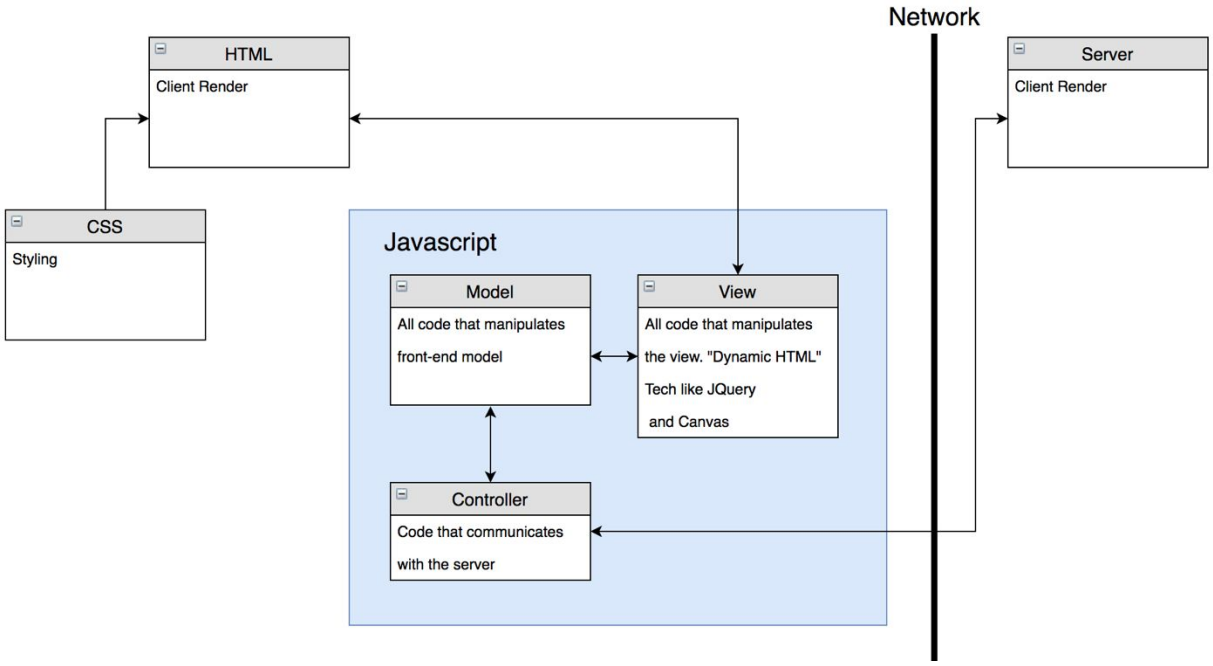


Figure 9. Front-End Architecture

The frontend architecture will be using HTML5 with CSS to render a Durham floor model canvas on the webpage. When the client makes a URL request to this website it will send a request to the server. The model will render the Durham floor plan that the user will interact with. Any requests the user makes will be sent through the controller. On the view, there will be a dropdown that will allow you to switch between users. When the client has selected a user, and hits the submit button, the controller will make an API call to the server and collect all of the location data from the mobile application. This will be rendered on the front-end for the client to observe.

Server Architecture

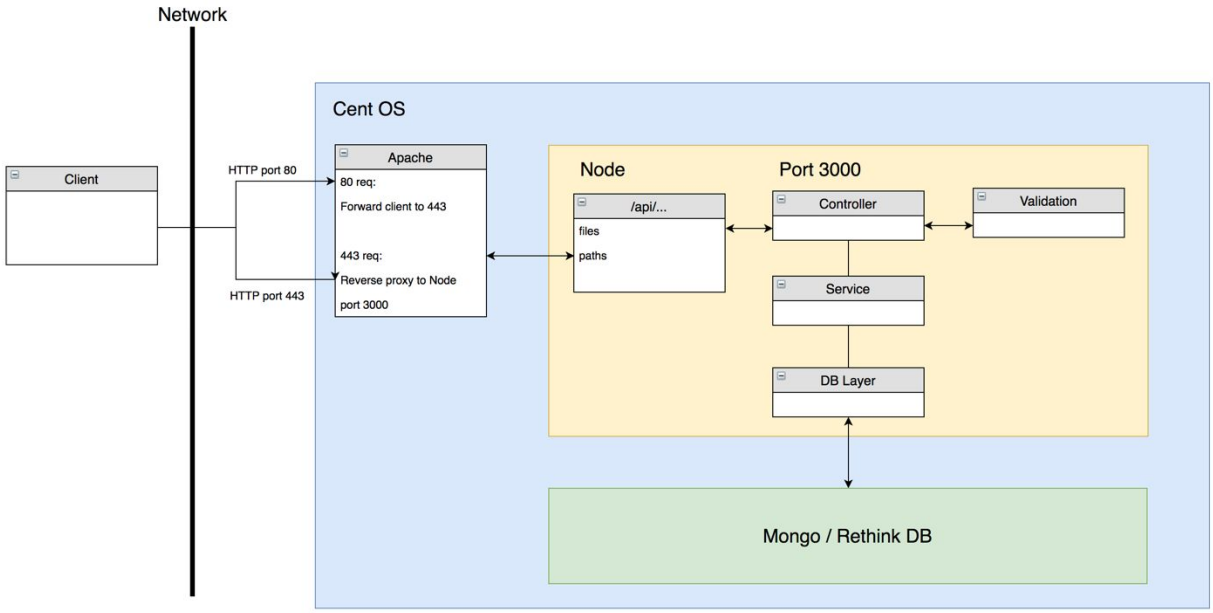


Figure 10. Server Architecture

The server architecture is fairly simple. The client will communicate with the server either through port 80 or port 443. If the client connects through port 80, they will be redirected to port 443 (HTTPS). Since Apache handles SSL very well, we are using that as our point of contact. Once the request is decrypted, it is sent to Node through a reverse proxy to Node running on a closed port, 3000. Node runs through the request described in the back-end architecture. When a database call is initiated, Node communicates with either MongoDB or RethinkDB, which are running on their own closed ports.

Backend Architecture

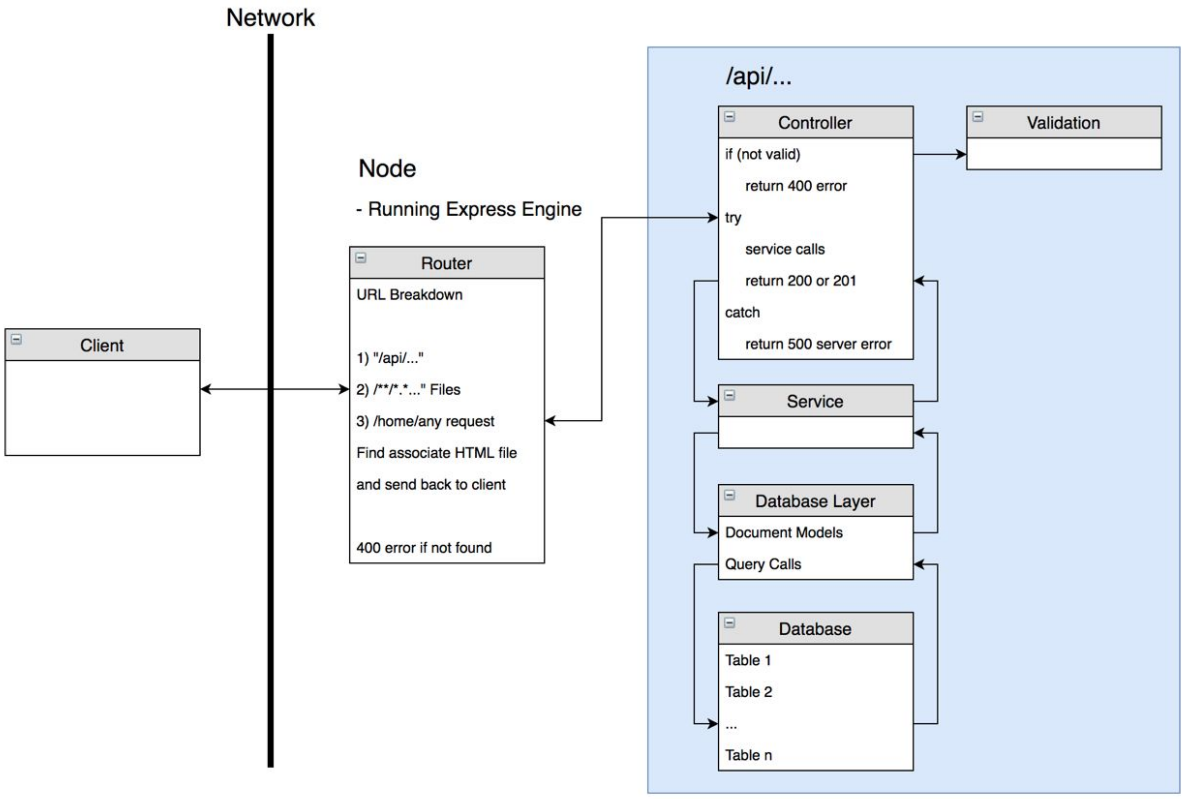


Figure 11. Backend Architecture

When the client sends requests to the API, it will go through a layered architecture. The way our layered architecture will work is exactly as the name might suggest. The request will go to our router, where it will be directed to send back webpages, or handle API requests. If it's an API request, it will go to our controller, where it will be validated and then sent off to the service layer. If the validation layer fails, it will send back a 400 response to the client. The service layer will manipulate the data as needed and communicate to the database layer. The database layer will create the database calls and do any CRUD operations. It will pull from the database whether that is MongoDB, or RethinkDB. Once the data has been collected, it will go back up the layer chain. The service will have the data at this point and manipulate as needed. It will send it back to the controller where it will send a 200 response if successful, or a 500 response if there was any internal error along the way. The validation layer will defend against injection attacks by making sure everything meets the requirements. This is a strict format, and at no point should layers cross communicate, i.e. the controller should not talk to the database layer.

2.2 Design Analysis

Cross Platform vs Native

One of the problems surrounding mobile development is that the two main platforms (Android and iOS) run their applications on different languages. It would be great if we could code an application in one language, and then just automatically transform it for any platform we want to use. This is what Xamarin and React Native do. Xamarin is an open source software that uses the .Net framework. Xamarin uses the C# programming language within the .Net framework. React Native is another open source software which uses JavaScript.

Pros of React Native

1. Developers who develop in reactive native can create apps for the web, iOS, and Android platforms.
2. Does not require multiple teams for multiple platforms.
3. Allows for rapid prototyping
4. Saves development time
5. Quickly solves cross-platform bugs

Cons of React Native

1. It is a fairly new software, so it has less support and documentation.
2. It is impossible to write a truly native feeling app in react native without also having knowledge of the native operating system.
3. Limited access to platform specific APIs.
4. Steep learning curve.
5. High security vulnerability.
6. Not suited for computationally intensive tasks.

Pros of Xamarin

1. Xamarin performance is very close to native performance.
2. The platform has high scalability.
3. Xamarin is open sourced.
4. It provides its own IDE.

Cons of Xamarin

1. Delayed support for new iOS/Android software updates.
2. Knowledge of the native platform language (Android, iOS) is required.
3. Xamarin applications are larger than native applications.

4. Compatibility issues with third party libraries.
5. Has limited open source libraries.

Advantages of Native Android

1. Better user experience.
2. Optimized for Android.
3. Compiles faster.
4. Unrestricted access to Android API's.
5. Third party compatibility.
6. More control.
7. Update support through the Google Play Store.

Disadvantages of Native Android

1. Applications written for Android, only run on Android phones.

Disadvantages of Cross-Platform

1. Lack of documentation.
2. Restricted access to Android frameworks, libraries, and hardware sensors.

iOS vs Android

iOS mobile devices have Accelerometer, Gyroscope, Pedometer, Magnetometer, GPS, Barometric, Proximity, Facial Recognition, Radio, Bluetooth, and Fingerprint sensors. Android devices have Accelerometer, Gyroscope, Magnetometer, Heart Rate, Light, Fingerprint, Proximity, Temperature and Bluetooth sensors. The devices available for our project are all Android. Our client has given our team two Android devices (HTC One, Galaxy Note).

Pros of Using an Android Device

1. Team experience.
2. Android is open source.
3. Android is programmed in Java.
4. Highly customizable screens
5. Expandable storage.

Cons of Using an Android Device

1. Complex layout schemes.
2. High fragmentation due to many different Android devices running on many different API's.

Pros of Using an iOS Device

1. Fingerprint /Face recognition increase security.
2. iOS using swift programming language is easy to read.
3. Swift is open-source.

Cons of Using an iOS device

1. iOS is not open source.
2. Integration issues with projects written in Objective-C
3. Internal storage thus you cannot upgrade the hard drive unless you buy a new device.
4. iOS framework documentation isn't as detailed.
5. XCode IDE has a steep learning curve.

Our team chose Android because as a whole, we have more experience working with Android than iOS. Also, Android has more open source resources, better documentation, and typically more affordable phones than devices running iOS.

Backend

Our solution revolves around storing and monitoring worker coordinate data. Since our solution requires having an access point to transfer the coordinate data to our database, our application must be able to store data internally when a network connection is not available. We will use Realm for this internal storage process.
For our external database we have considered MySQL, RethinkDB, MongoDB, and MariaDB. The advantages and disadvantages are as follows:

Internal Storage Advantages

1. Designed for small lightweight operations.
2. Reads and writes without no extra configuration.
3. Stored in a single file.
4. Increased performance on reading operations.

Internal Storage Disadvantages

1. Designed for small datasets.
2. Writing and reading operations can't run concurrently.

MYSQL Advantages

1. Host-based verification. [3]
2. Flexible privilege and password system. [3]

3. Security encryption for all password traffic. [3]

MySQL Disadvantages

1. No built-in support for XML or OLAP.
2. Speed deficiency on big tables (not good for agile).

MongoDB Advantages

1. Database engine supports JSON. [4]
2. It can store any structure of data. [4]
3. The data Schema can be written without any downtime. [4]
4. Supports JSON. [4]
5. Encrypted storage engine.

MongoDB Disadvantages

1. It was not designed to handle relational data models. [3]
2. Setting up MongoDB is a long process. [4]
3. Default settings are not secure. [3]

MariaDB Advantages

1. Allows high scalability with easy integration. [3]
2. Real-Time Access. [3]
3. Fast query performance and processing. [3]
4. Has the core functionality of MySQL. [3]
5. Encryption for network, server and application levels. [3]
6. Fast and stable. [3]
7. Variety of plugins. [3]

MariaDB Disadvantages

1. MariaDB is poorly supported on some operating systems (works best for Linux). [3]
2. Migrating from MariaDB to MYSQL is difficult. [3]
3. Not supported by hosting environment. [3]

RethinkDB Advantages

1. Can listen for changes to data. [2]
2. Supports sharing, parallel queries and MVCC. [4]
3. Powerful query language, (Node driver for JavaScript developers).

4. Compatible with JavaScript (Node.js), Python, PHP, Ruby, C, C#, C++, Objective-C, Java.
5. Designed to be accessed from an application server.
6. Atomic updates. [2]
7. Easy to setup and learn.
8. Open source.

RethinkDB Disadvantages

1. It's not ACID-compliant, doesn't have schema, it stores the field name of each document individually which can impact compression.
2. Mongo is 3x faster then Rethink for querying.
3. You need to setup your own auth and user accounts.

We chose RethinkDB because of the change feeds the architecture provides.[2] The reason why this is important is because the HoloLens and website need to constantly update the worker's positions in real-time. If we used a different database, then both the HoloLens and website would have to constantly make pull requests to check for changes in data. By using RethinkDB our website/HoloLens team can subscribe to the change feed and will automatically receive an event when a location coordinate has been updated. This saves us from making unnecessary requests to the database.

We didn't go with mongoDB because our solution doesn't require a lot of pull requests unless the HoloLens and web operator want to render an early event. MongoDB has great performance when pulling documents of data from their database but since we don't need that functionality for our solution it is not feasible for our project. MYSQL database is very nice for storing data in a structure format but it doesn't offer change feeds meaning which it receives an update from the application it will send to the subscribed applications.

HoloLens

Possible AR technologies currently available are the Microsoft HoloLens, Google Glass, and Magic Leap. The Microsoft HoloLens operates by rendering 3D images into the visible world around you.

Advantages of HoloLens

1. Comfortable to wear.
2. Gesture recognition.
3. Voice command recognition.
4. Eye-detection based selection.

5. Hands free experience.
6. Unity development.

Disadvantages of HoloLens

1. Gesture recognition is very tricky and only a few gestures are available.
2. Costs $3000 for the developer edition.
3. Field of view gets distorted once you start moving.
4. Requires Windows OS for development.

Disadvantages of Google Glass

1. In 2017, Google only launched the Google Glass Enterprise Edition to companies like Boeing.
2. Google Glass Developer Edition was discontinued in 2015.

Advantages of Magic Leap

1. Much wider field of view than the HoloLens.
2. Two different size glasses which depend on your interpupillary distance.
3. Lighter hardware on your head.
4. Supports Mac development.

Disadvantages of Magic Leap

1. Costs $2,295 for developer edition.
2. Light pack computing device needs to be attached to your belt and has a USB-C cord attached to the headset.
3. If the computing device breaks it will cost an extra $495 to replace.
4. The leap glasses level needs to be aligned almost perfectly.
5. Leap technology tracking and sense recognition is more jittery than HoloLens. Space mating doesn't work properly.
6. Problem when glasses have to scan a room. Menus can go through walls.
7. Excessive heat on your head from wearing the device.

Our client is working on several projects on the Microsoft HoloLens to increase visualization solutions for his business. Our client product line involves around optical solutions. Therefore, our team will be using the Microsoft HoloLens because it provides visible solution that stills allows the user to interact with the world around them.

3 Testing and Implementation

For the scope of our project, there will be no safety issues so long as the users behave normally. Even in the case of the project being implemented on a construction site, there would be no "new" safety consideration. While a construction site may have dangers, this project won't introduce any new risks.

3.1 Functional Decomposition

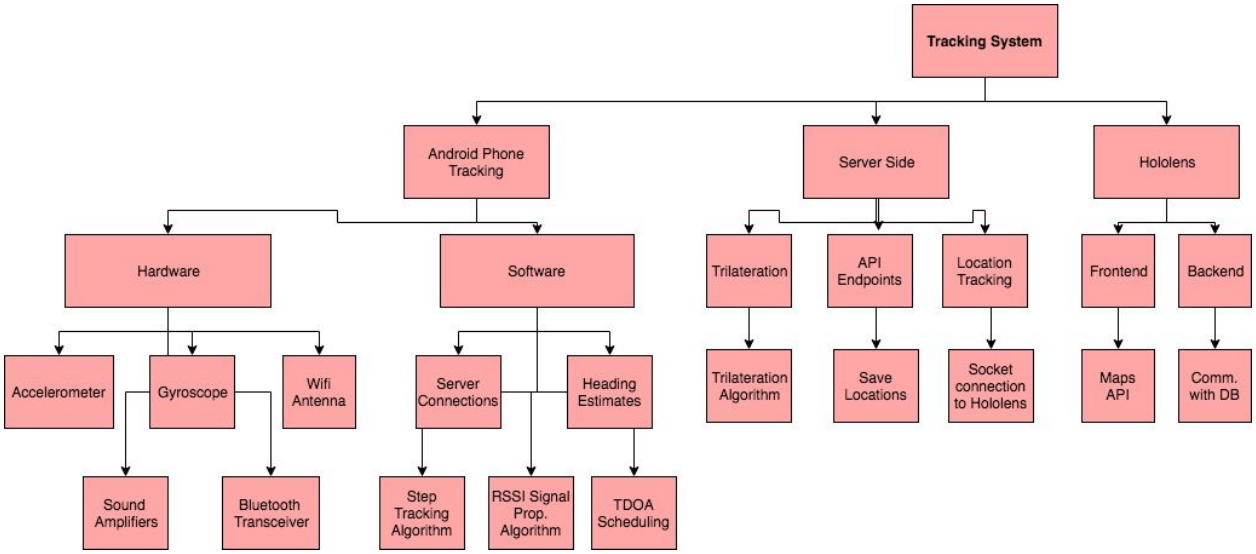


Figure 12. Functional Decomposition

Our Tracking solution involves 3 main components, the Android phone, server, and Hololens. The Android phone application has both hardware and software. Some of the hardware is specific to the Android phone, such as the Accelerometer and Gyroscope, while other hardware exists in both our Android phone and our Bluetooth Beacons, Access Points, or sound amplifiers.

The software primarily represents the methods for communicating with the different hardware components, while also including functionality for communicating with the server, and implementing our different algorithms.

The server is responsible for three things:

1. Implement trilateration algorithm
2. Save locations to database
3. Send information to Hololens via websockets.

We chose to implement the trilateration algorithm on the server as opposed to the phone in order to save valuable Android resources, and also because there would be no additional functionality gained from implementing trilateration on the phone as opposed to the server.

The final component, the Hololens serves to display our location data, and also communicate with the database. The HoloLens application will use Mapbox to render the jobsite into the augmented reality. A websocket will connect the Hololens to our database in such a way, that upon updating an entry in our Database, the updated information is automatically sent via the websocket to the Hololens. This is done by simply attaching a listener to the dataset.

3.2 Hardware and Software

Mobile Testing

For the Mobile testing we will be using Espresso unit test framework. This framework will allow us to test input with the mobile interface of our application. With this framework we can use annotations (@BeforeClass, @ rule, @BeforeClass, @after, @rule) that can be applied to class methods which will allow our team to do a lot of different things. With this framework we can write different behavior tests that will check different action on View interaction like pressing a button.

The Espresso unit test framework is useful because it allows you to test views which are actions that the user interacts with on the mobile application. The view is could be a button in which performs an action. This testing framework allows you to perform checks on these actions which can help you define coverage for your application. This allows you to test your UI interface which is really helpful for mobile development.

HoloLens Testing

For the HoloLens tests we will be using unity test runner tool that allows our team to test our implement by using both edit and play mode. This framework is an open source unit testing library that allows you to create test scripts and load it up into a scene. The edit mode is used to check if certain GameObject is within a scene. Edit mode is like any other unit test framework but the play mode allows you to create tests while you are playing is running within a scene in real time. Play mode is what we will be using for our testing environment.

The rest runner framework is very useful because it offers two different modes for testing. The play mode is very useful for game/VR development because you can create unit test scripts, and have it run in the background. This means that we can detect more bugs because the general unit

test are static meaning you might not necessary cause a fault and have it propagate down to where it causes your software to crash.

3.3 Functional Testing

Functional Requirement:	Test Plan:	Verification:	Validation:
Indoor/outdoor user tracking	We will setup 3 android devices at certain starting locations for tracking. We will provide the user with a specific path to walk which includes a start point and an end point. We will record their path and display it on Mapbox and compare their path with the test path.	Integration Testing: The AR team will design unit tests: 3 avatar game object instances get created when 3 individuals are using the mobile application. Compare the avatar path with the test path and report the results. The Web team will design Unit tests: Will store the user's path from the start point to their end point and compare their path with the test path and report their results. The backend team will create unit tests: Verifies that the schema tables get the correct user path from the mobile application. The test from all 3 teams will have code walkthroughs and reviews to verify that the unit tests are tracking 3 individuals according to their path trajectories.	AR validation: The tester shall verify that there are 3 avatars that are rendered in the 3D worksite. They will verify that their path matches the test path specified by the test plan and it is rendered in the correct location on the map. Web Validation: The tester will verify visually that the users path taken has the same longitude and latitude coordinates as the test plan. Database Validation: It will verify that the table schemas recorded all the location data from the 3 users' path from the mobile device. After the tests are completed, the tester will do a report analysis of the errors that they found and will be reviewed by the development team.

Movement Sensitivity	We will turn on the mobile application and then have the user perform normal walking and running speeds.	Usability Mobile Test: The user will launch the application and perform a walking movement and running movement and report the results. The mobile team will have a walkthrough to verify how this test is going to meet moving sensitivity requirement.	Tester: The user will turn on the mobile application and will perform running and walking movement and look to see if the mobile device detects their walking and running speeds. After the tests are completed, the tester will do a report analysis of the errors that they found and will be reviewed by the development team.

Store tracking to Rethink DB	We will connect the mobile device to a Wi-Fi and will then perform a step and make a http post request to the database.	Mobile Unit Test: Verifies that the mobile application does an HTTP request when the device is connected to a WI-FI access point. Backend Unit Test: Verifies that the data received from mobile application matches with the recorded data from the mobile application. The backend and mobile team will have code walkthroughs and reviews to verify that they the unit tests are fulfilling sending data through an access point according to the specifications of the storing tracking information requirement.	Mobile Tester: Verifies that the application does an HTTP request to the server and the connection was successful. Backend Tester: The tester will verify that the data recorded on the phone matches with the data stored on the database table scheme. After the tests are completed the tester will do a report analysis of the errors that they found and will be reviewed by the development team.

Distance Accuracy	We will give a user a new starting location to start the application at. We will instruct him/her to move 1 meter and see if the new current location is within meter.	Mobile Unit Test: Create a unit test that starts at Location A and moves to location B and then verifies the user new location is within our 1-meter accuracy requirement. The mobile test team will perform a code walkthrough that explains how their test meets the specification of the distance accuracy requirement. The review team will check documents and files to ensure that the code meets our coding standards.	Mobile Tester: Verifies that the movement from Location A to Location B is within 1-meter accuracy. After the tests are completed the tester will do a report analysis of the errors that they found and will be reviewed by the development team.
Delay Accuracy	We will use a timestamp when the mobile application has gathered all the sensor data and starts to perform the smoothing/predicti on algorithm while the software creates a 5 second delay. After the delay is completed the software will create another time stamp and compute the time difference.	Mobile Unit Test: Create a unit test that takes the time stamp after the data has been collected and after the x second delay has been completed. Then compute the time difference and compare it with delay requirement. The mobile test team will perform a code walkthrough that explains how their test meets the specification of the delay accuracy requirement. The review team will look over the documents and files and confirm it meets our coding standards.	Mobile Tester: Verifies that the delay time of the application is within the 5 second threshold by running the test script and report the results. After the test is completed, the tester will complete a report analysis of the errors that were found. The development team will review these documents.

Drift Accuracy	We will start a user at a starting point by Durham. We will then have the user move 2 meters in any direction and record the position and then do a computation of the possible positions and see if the user is within the 1 meter of the computed location.	Mobile Unit Test: Create a unit test that takes the starting position of the test plan and then computes all the possible new position paths the user could had taken and then compare those position with the user's current position and report the results. The mobile test team will perform a code walkthrough that explains how their new positions will meets the specification for the drift accuracy requirement. The reviewers will look at the code to see if it meets our coding standards.	Mobile Testers: Verifies the users new position is within 1-meter of the possible computed positions that was computed. After the test is completed, the tester will complete a report analysis of the errors that were found. The development team will review these documents.

HoloLens Monitoring	The HoloLens will receive the new recorded location data from the Rethink database. It will then render the user's new position in the augmented reality map. The user that is wearing the device should see the avatar move from the user's current position to the new position that was received.	AR Unit Test: Create a unit test that takes the newly recorded data from RethinkDB and calculate the new position that the avatar transform should read after unity renders the new position. Compare the avatar's new position with the computed position and report the results. The AR team will perform a code walkthrough that explains how their new computed transform position meets the specification of the HoloLens monitoring requirement. The reviewers will look at the code and see if it meets the unity coding standards.	AR Testers: Verifies the new transform position the avatar moves to in the AR map is the same as the newly computed position. After the test is completed, the tester will complete a report analysis of the errors that were found. The development team will review these documents and address the software errors that were reported.

Monitor Accuracy	We will create a timestamp and send a location packet by using a http request to the database. When the database has received the packet, it will create another timestamp. The database will then send the location packet to the website and HoloLens software. The HoloLens and website will then create a new timestamp and calculate the difference and see if it is within 10 second threshold.	AR Unit Test: Create a unit test that take the recorded timestamp from the mobile application and create a new timestamp. Then compute the difference and compare it with the 10 second requirement and report the results. Web Unit Test: Create a unit test that takes the recoded timestamp from the mobile application and creates a new timestamp. Then compute the difference and compare it with the 10 second requirement and reports the results. Backend Unit Test: Create a unit test that takes the recoded timestamp from the mobile application and create a new timestamp. Then compute the difference and compare it with the 10 second requirement and reports the results. The AR, Web, and Backend will perform a code walkthrough that explains how their time calculation meets the monitor accuracy requirement. The reviews will look at the code and see if it meets the coding standards.	AR Testers: Verifies the that recorded location is being displayed on the map in AR within the 10 second threshold from the time it was recorded on the mobile device. Web Testers: Verifies the that recorded location is being displayed on the website map within the 10 second threshold from the time it was recorded on the mobile device. Backend Testers: Verifies that the recorded location is being displayed on the website map within the 10 second threshold from the time it was recorded on the mobile device. After the test is completed, the tester will complete a report analysis of the errors that were found. The development team will review these documents and address the software errors that were reported.

Bluetooth Sensor	The android device will be paired with the beacon so that they our connected to each other. We will set the beacon at a location x inside the jobsite. When the android device receives a signal from the beacon, it will send a location update to the android device.	Mobile Unit Test: Create a unit test that takes the updated the location from the beacon and then compares it with the current location that is stored on the device. It should compare these two and report the result. The Mobile team will perform a code walkthrough that explains how their comparisons meet the Bluetooth sensor requirement. The reviews will look at the code to make sure that the developers are meeting the coding standards.	Mobile Tests: The user will stand within 1 meter of the beacon to trigger the location update. They will then verify that the mobile devices current location is getting updated. After the test is completed, the tester will complete a report analysis of the errors that were found. The development team will review these documents and address the software errors that were reported.
Android Low Battery Notification	The android device battery must be drained to 10% of its total capacity. Then we will run the application on the android device.	Usability Mobile Test: Let the device get below 10 percent of battery and display a low battery notification. The Mobile team will perform a code walkthrough that explains how their notification meets the Android low battery notification requirement. The reviews will look at the code to make sure that the developers are meeting the coding standards.	Mobile Tester: Tester should see if the low battery notification displays when the devices battery is below 10 percent. After the test is completed, the tester will complete a report analysis of the errors that were found. The development team will review these documents and address the software errors that were reported

HoloLens Low Battery Notification	The HoloLens battery must be drained to 10% of its total capacity and then run the application on the device.	Usability AR Test: Let the device get below 10 percent of battery and display a low battery notification. The AR team will perform a code walkthrough that explains how their notification meets the HoloLens low battery notification requirement. The reviews will look at the code to make sure that the developers are meeting the coding standards.	AR Tester: Tester should see if the low battery notification displays when the devices battery is below 10 percent. After the test is completed, the tester will complete a report analysis of the errors that were found. The development team will review these documents and address the software errors that were reported

Table 1. Function Requirements Test Plan

3.4 Non-Functional Testing

Non-Functional Requirements	Test Plan	Verification	Validation

Battery Life Cycle	A user will turn on the mobile device and launch the application as a background process to monitor the battery and CPU usage throughout an 8-hour work day.	Usability Mobile Test: The user will launch the app and monitor the mobile application every hour to report the results. The Mobile team will explain why the monitoring test meets the Battery Life Cycle requirement.	Mobile Tester: The user will launch the mobile application and monitor the Android profiler every hour for 8 hours. After the test is completed, the tester will complete a report analysis of the CPU usage. The development team will review these documents and address the software battery issues, if any.
GPS Sensor	We will turn off the GPS location service on the mobile device and create a pre-planned walking path around Durham. We will record the path walked, compare it with the pre-planned walking path and show the differences.	Mobile Unit Test Create a unit test which takes the pre-planned route, compares it with the user's route and reports the results. The mobile team will perform a code walkthrough that explains how their unit test meets the GPS sensor requirement. The reviewers will look at the code to ensure the developers are meeting the coding standards.	Unit test where you create a walking path, perform the walk, compare the expected path with the actual path and return the result.

Look and Feel	We will place the mobile device inside of the armband and wear the device for 8 hours.	Usability Test: We will create a test that requires the tester to wear the mobile device for a normal work day and report the results.	Mobile Tester: The tester will wear the device around his/her arm for 8 hours and then report the comfortability to the development team.
Environment	We will launch the HoloLens application at the client's facility inside of their conference room.	Usability Test: We will create a test that requires the tester to wear the HoloLens inside of the client's facility.	AR Tester The tester will go to the client's facility and see if their facility has the proper environment for running the HoloLens application. The tester will complete a report analysis of their findings to the HoloLens development team.

Table 2. Non-Functional Requirements Test Plan

3.5 Process

Mobile Test Plan

Functional Requirement Test: Movement Sensitivity (Mobile)

For the Movement User Tracking Test, we will create an Espresso unit test to test the step detection algorithm by using mock data to mimic what would be expected of a person walking or running. The test coverage will confirm if the user is in a running or walking state based on the acceleration values for normal running and walking speeds. If the acceleration is between 2m/s^2 and 6m/s^2 then we know that the user is walking, but if the speed is above 6m/s^2 then the user must be running. The results will be recorded into a file that will be analyzed.

Functional Requirement Test: Store Tracking to RethinkDB

For the Store Tracking to Rethink DB Test, we will use the Espresso framework to create a unit test which will connect to an access point and send an HTTP request to the server. The unit test

will verify the request code is received from the server. If it receives the success request code the unit test will output, it was successful. If the success code is not received, it means that we were unable to connect to WIFI.

Functional Requirement Test: Distance Accuracy

For the Distance Accuracy Test, we will use the Espresso framework to create a unit test that will store the user's starting point. We will call this location A. Next the user will move three steps to a new location. We will call this location B. We will then use the AutoCAD to calculate the distance the user should have traveled and compare it with the distance the user actually traveled. If it is within our 1-meter requirement, then the test has passed.

Functional Requirement Test: Delay Accuracy

For the Delay Accuracy Test, we will use the Espresso framework to create a unit test that will store a timestamp of when the new location was recorded. The way the test will work is we will have the user walk to a step to record a new location. The test will then record that timestamp and send the data to rethinkDB. Next, when the test receives a success code it will create another timestamp. If the timestamp is within the 5 seconds requirement the test will pass.

Functional Requirement Test: Drift Accuracy

For the Drift Accuracy Test, we will use the Espresso framework to create a unit test that will store the user's starting point, and the possible end points that the user could have walked. When the new position is recorded, the software will check all the possible positions which fall within a tolerance. Afterwards, the unit test will output the results to a file where the results can be analyzed.

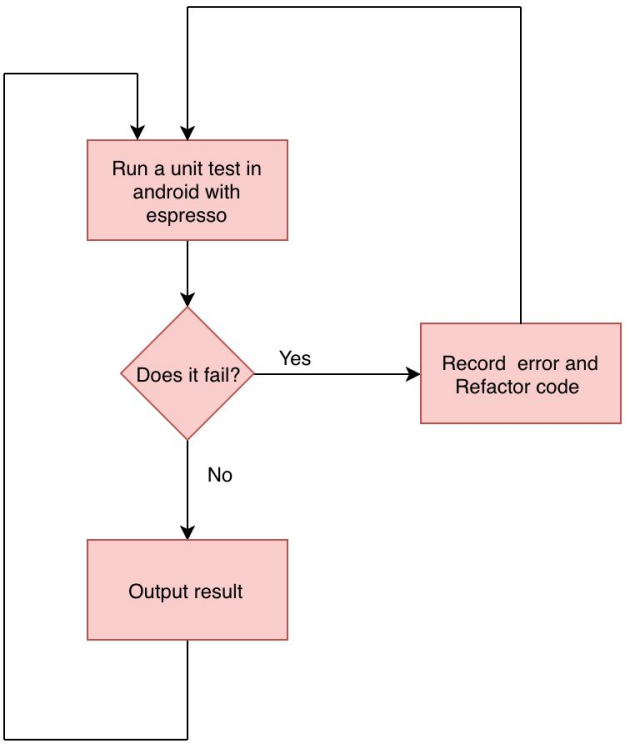


Figure 13. Android Testing Process Diagram

HoloLens Test Plan

Functional Requirement Test: Indoor/Outdoor User Tracking (HoloLens)

For the Indoor/Outdoor User Tracking Test, we will use the test runner framework that will load up a test scene by having 3 fake users' login (hard code) and it will verify that 3 avatar instances are displayed in virtual reality environment. This information will be retrieved by the database script when HoloLens application first executes.

Functional Requirement Test: HoloLens Monitoring

For the HoloLens Monitoring Requirement, we will create a unit test that will be loaded up into the scene. We will then hard code the users' new position and send it to the RethinkDB. The database script will receive the user's new location and translate the avatar to the new position. The unit test will then check to see if the avatar's transform position (x,y,z) has the same values as the new position received.

Functional Requirement Test: Monitor Accuracy

For the Monitor Accuracy Requirement, we will create a unit test that will be loaded up into the scene and will receive an update from the database script. The update will contain a user's new location (x,y), the timestamp of when the user location was read, and a timestamp of when the database script received it. We will then calculate the difference and see if it meets the 10 second requirement.

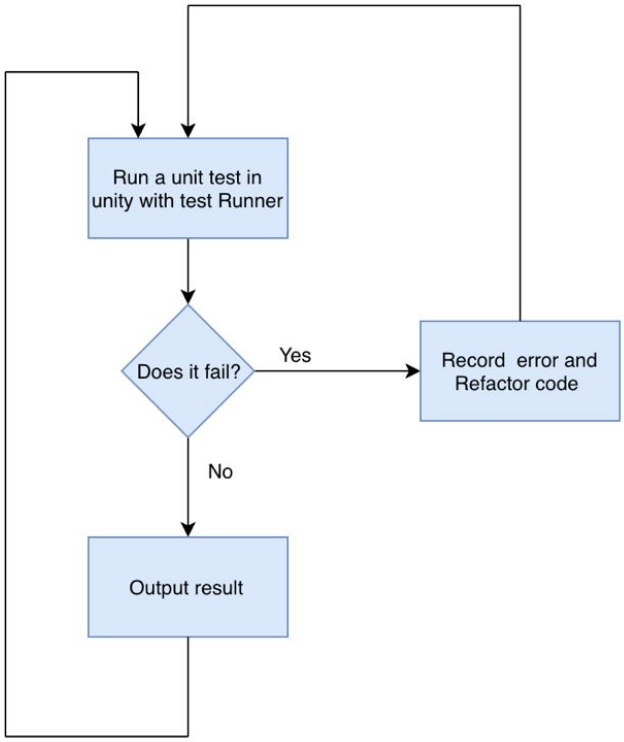


Figure 14. HoloLens Testing Process Diagram

3.6 Results

We have created a step tracking algorithm to simultaneously calculate the distance traveled by a user stride and the direction of motion. We successfully walked the marked path in Durham from a starting point to an end point. To test our prototype, we first outlined the path, and then selected team members to walk it. The data would be stored into the database, and visualized on the front end by using HTML Canvas. Figure 12 (below) shows a screenshot of our front end. The floor plan of Durham is overlaid on the Canvas.

Modeling and Simulation

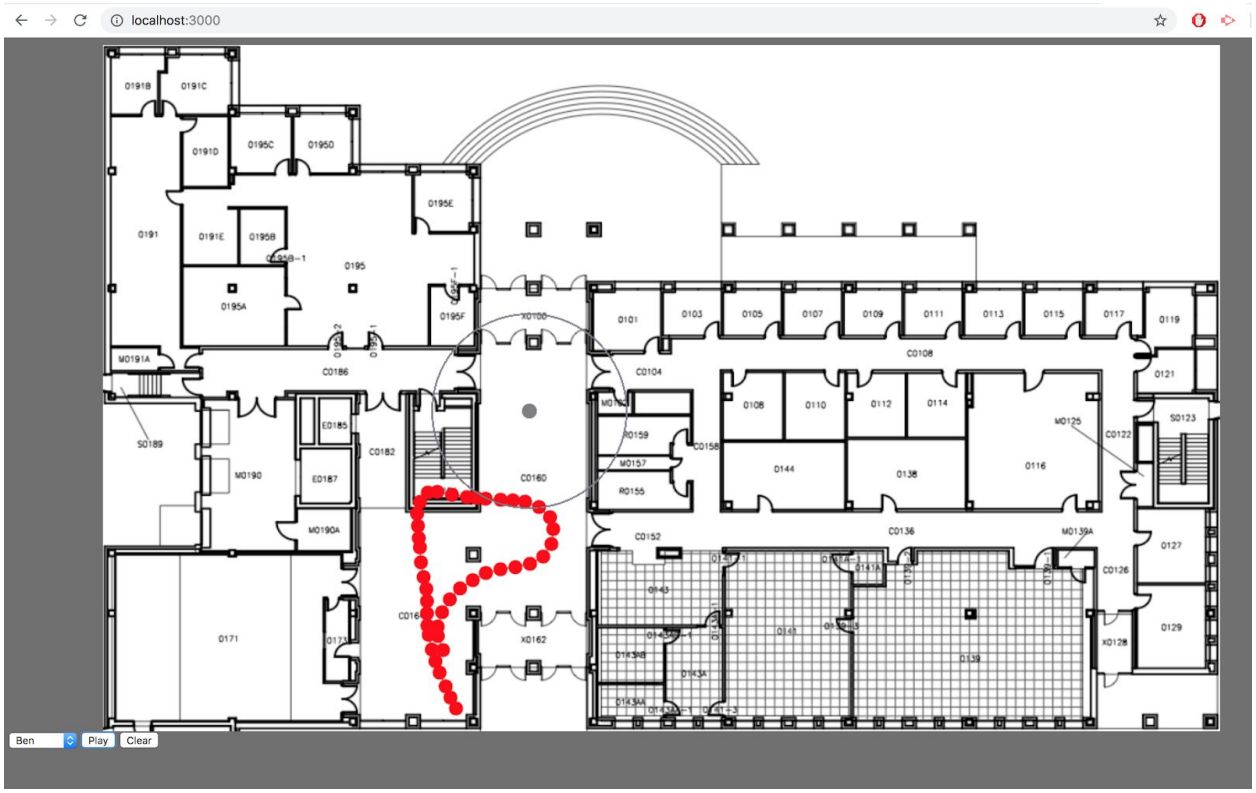


Figure 15. Model of Durham Floorplan

Implementation Issues

With this model, we use a red dot to represent the user. The path is visualized by connecting each dot with a straight line. In order to overlay our path correctly to the Canvas, we had to rotate our orientation measurements to fit our orientation on Canvas. This proved to be slightly more difficult than a simple translation of data, because electromagnetic radiation in the building actually disrupted our heading estimations. We hope to fix this next semester by switching our Android Rotation Vector Sensor implementation to a Gyroscope implementation. Basically we will start with a current orientation quaternion. We will then rotate the quaternion based on the axial displacement given by the Gyroscope. Each displacement will be added to the current orientation quaternion to get the new orientation.

Another issue our team faced involved using a Bluetooth beacon. We first tried to use a mobile device to advertise Bluetooth packets to our application, but due to inconsistencies in measuring the txPower (RSSI value read at distance of 1m from advertising source) of the advertising phone, we received very inconsistent results. We wanted to compare our results to what would be recorded from a commercially available Bluetooth Beacon. This led us to order a BLE

Bluetooth beacon from Amazon. We followed instructions to implement the Eddystone advertising protocol, and compared its results to our initial tests. The BLE Beacon was much more consistent than our mobile phone beacons. This leads us to believe there is a way to increase the accuracy of our mobile phone beacons by controlling its Bluetooth advertising settings. We hope to develop more on this idea in the coming semester.

4 Closing Material

4.1 Conclusion

So far, our team has researched localization techniques utilizing light sensors, radio frequencies, sound, and magnetism. We have created a Dead Reckoning system which estimates distance and heading orientation, and have plans to incorporate Bluetooth and Wifi technologies. Ultimately we would like to implement a solution which is robust enough to track a user to within 1 meter of accuracy. Our testing site will include both outdoor and indoor localization. The outdoor portion will include a small area extending from the west entrance of Durham. The indoor portion will be the entire first floor of Durham.

Next semester we plan to add to our tracking system by utilizing two types of trilateration. The first is RSSI based, interpreted from radio frequency measurements of access points. The second, using Time Dilation of Arrival is expected to be slightly more accurate than the RSSI based trilateration method, and should bring us closer to reaching our goal of 1 meter accuracy.

4.2 References

1. “UNITED STATES DEPARTMENT OF LABOR,” Occupational Safety and Health Administration. [Online]. Available: https://www.osha.gov/oshstats/commonstats.html. [Accessed: 04-Dec-2018].
2. “Getting started,” Frequently asked questions - RethinkDB. [Online]. Available: https://www.rethinkdb.com/faq/. [Accessed: 04-Dec-2018].
3. “The Top 7 Free and Open Source Database Software Solutions,” Capterra Blog The 11 Most Popular Hotel Management Software Solutions for Small Hotels Compared Comments. [Online]. Available: https://blog.capterra.com/free-database-software/. [Accessed: 04-Dec-2018].
4. “The Pros and Cons of 8 Popular Databases,” KeyCDN Blog. [Online]. Available: http://www.keycdn.com/blog/popular-databases. [Accessed: 04-Dec-2018].
5. “Getting started,” RethinkDB 2.1.5 performance & scaling report - RethinkDB. [Online]. Available: https://rethinkdb.com/docs/2-1-5-performance-report/. [Accessed: 04-Dec-2018].

4.3 Appendices