Glass Map Application Design Document

May14-22 Senior Design Group

Project Definition

The Google Glass Map Application will allow the user (the person wearing the Google Glass) to view their position on a global map in real time. Our application will be interfacing with an Android device in order to supply the Glass with GPS data as well as the map visual. Through the course of developing the Google Glass application we will work to push the envelope on what the Glass can do. We hope that by the end of the project, we will have a viable mapping application with a wide array of uses. These could include, but are not limited to, GPS tracking and the display of other people/things, a step by step GPS guide, the overlaying of pertinent information to go along with what is being viewed.

Goals

1. The map display is completely independent of the source of both the incoming GPS and map data.
2. The user is able to capture and calibrate a map image to be displayed on the map.
3. The map displays the user's location in the map with a delay no greater than 500ms.
4. 2-way audio communication, at user's prompt. (or walkie-talkie style?)
5. Video streaming application displays on server side with delay of less than 500ms.
6. These settings are configurable on the client side: Video Streaming On/Off, Scale of map display, ???.

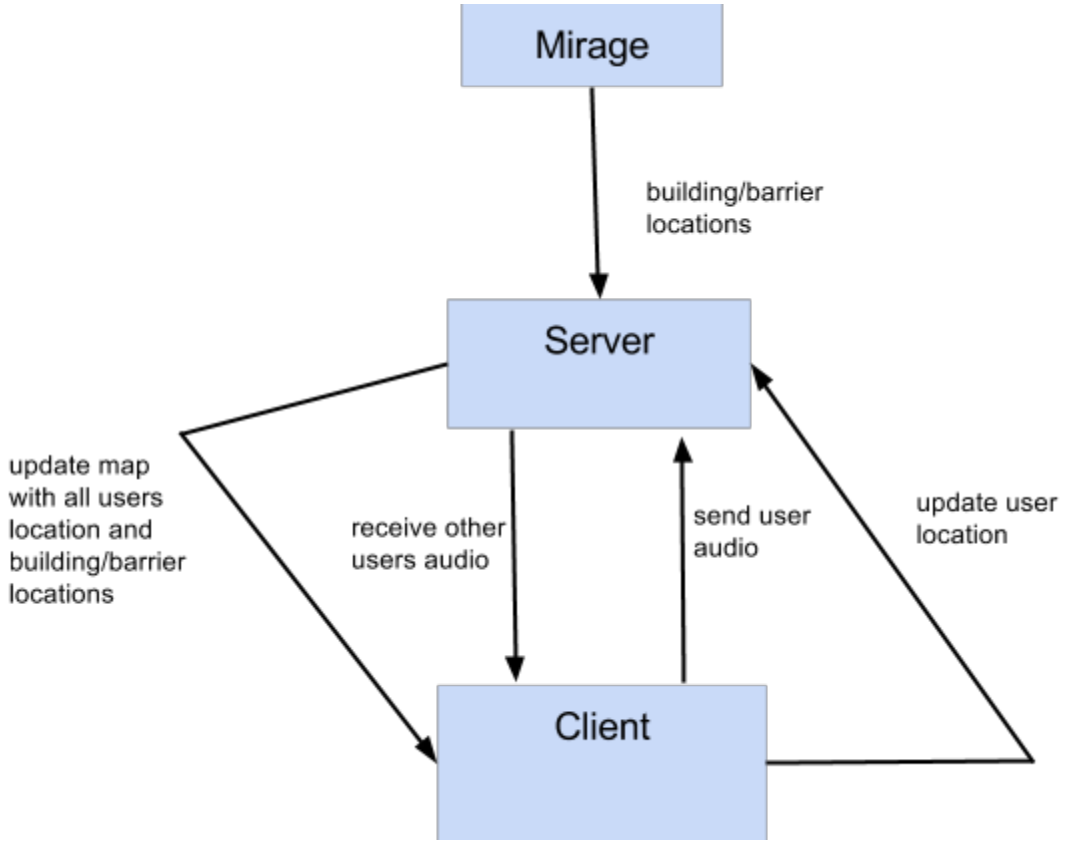
Deliverables

1. Map Display Module
    1.1. Global map display with user location
    1.2. Local map display with user location
        1.2.1. Completely independent of the source of mapping data
        1.2.2. MIRAGE source and Google FloorPlan overlay source implementations complete.
        1.2.3. Track and display up to 8 registered users on the map.
        1.2.4. Server only pushes data down to device upon device request.
    1.3. Automatically calibrate the local map (stretch goal)
2. 2-way walkie-talkie communication module
    2.1. Audio streaming as close to real time as possible
    2.2. Can't receive audio data if sending and vice versa
3. Video Streaming Module
    3.1. Constant video streaming to server in as close to real time as possible
    3.2. Server displays video stream
4. Normalized map/location data specification document
5. Data normalization plug-in specification document

System Level Design

1. Server Side Responsibilities
    1.1. determine map data normalization method
    1.2. maintain/update normalized data
2. Client Side Responsibilities
    2.1. Receive request specifications upon registration
    2.2. Send update requests at specified intervals

2.3. Draw screen and pass gps location/orientation when necessary

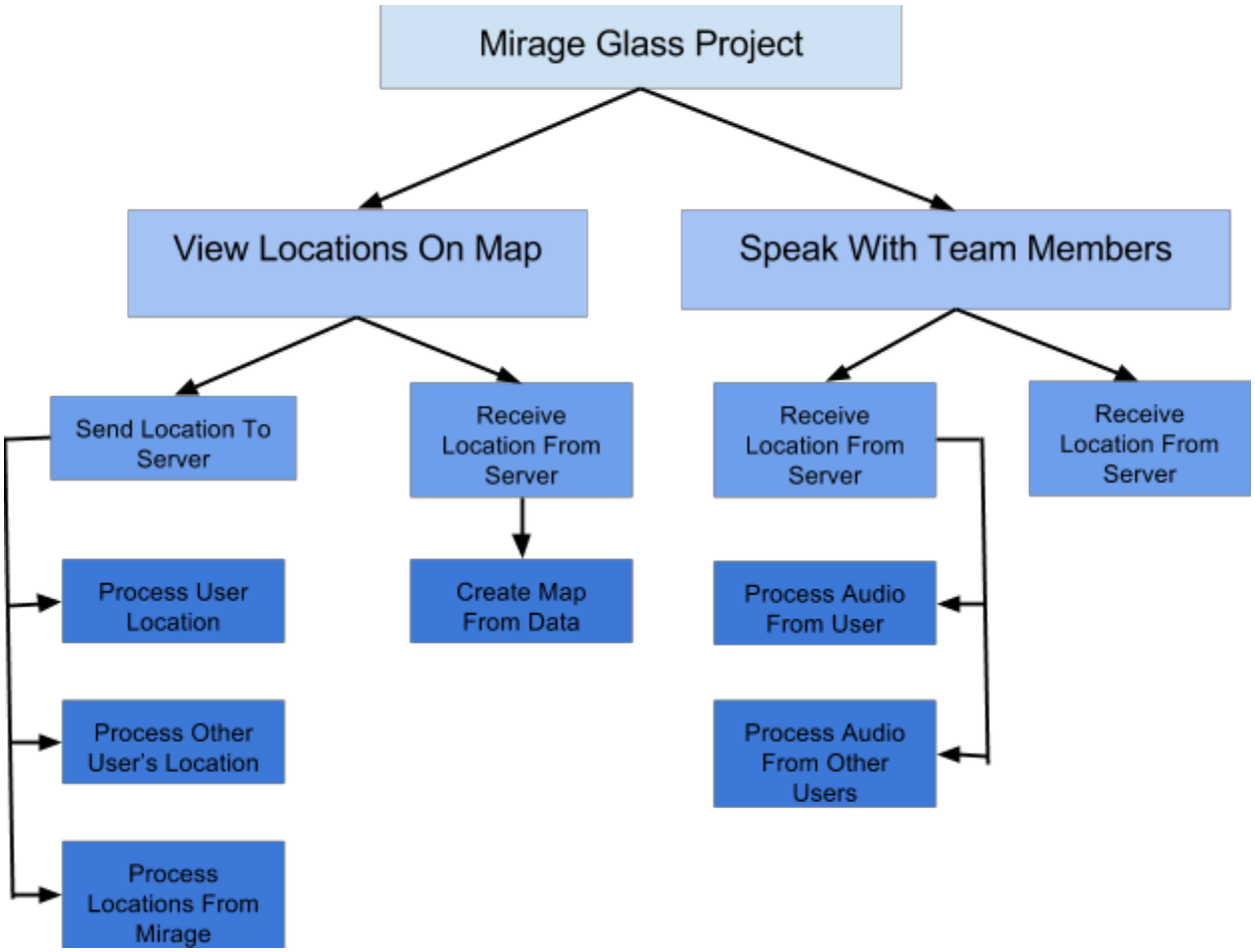


System Requirements

1. Android Device
    - 1.1. Android 4.0.4
    - 1.2. Bluetooth Capabilities
    - 1.3. Wi-Fi/3G Capabilities
2. Google Glass
    - 2.1. Android 4.0.4
    - 2.2. 640x360 display
    - 2.3. Bluetooth Capabilities
    - 2.4. Wi-Fi Capabilities
    - 2.5. 16 GB Storage
    - 2.6. 682 MB RAM
    - 2.7. 3 axis magnetometer
    - 2.8. Bone Conduction Transducer

3. Server
    3.1. Java
    3.2. Android SDK 4.0.4

Functional Decomposition



System analysis

Feasibility Study

Technical

The Glass Map Application is a challenging project. There are not a lot of resources available for developing Google Glass applications. Currently, Google recommends developing applications for Google Glass with the Android SDK. In the near future Google will release the GDK (Glass Development Kit). The GDK will be very similar to the SDK, but a few Android features that don't

currently work on Google Glass will not be available. After testing development in the Android SDK and running the apps on Glass we feel confident we can make the application in the Android SDK and it will function properly on the Glass.

Building maps with features in the Android application is feasible thanks to the recent release of Android Maps API v2 Android Maps API v2 allows us to do two very important tasks that will make a huge difference with the development of our application. First, it allows us to add shapes to a Google Map object. This means we can render new rooms and have real-time accurate GPS locations of the objects in the room. It also allows us to apply ground overlays to a real map. These two features will allow us to make customizable maps with real GPS coordinates.

The resources needed to complete this application are all currently available. There is currently not a lot of documentation or testing that has been done on the Android Maps API v2 because many of the main features we will be using were released very recently.
Overall, the project will be feasible from a technical standpoint, but will be very challenging.

Investigation of the current environment

Currently there are no apps written for Google Glass that display a detailed map of a building and track a user as they move through the building. Our app will be the first of its kind and will allow for a user to navigate flawlessly through a building.

There are currently many GPS applications that track a user's location on a satelitte map view or by road view, but no GPS applications that allow the user to build a new map setup and track their movement on it.

System Options

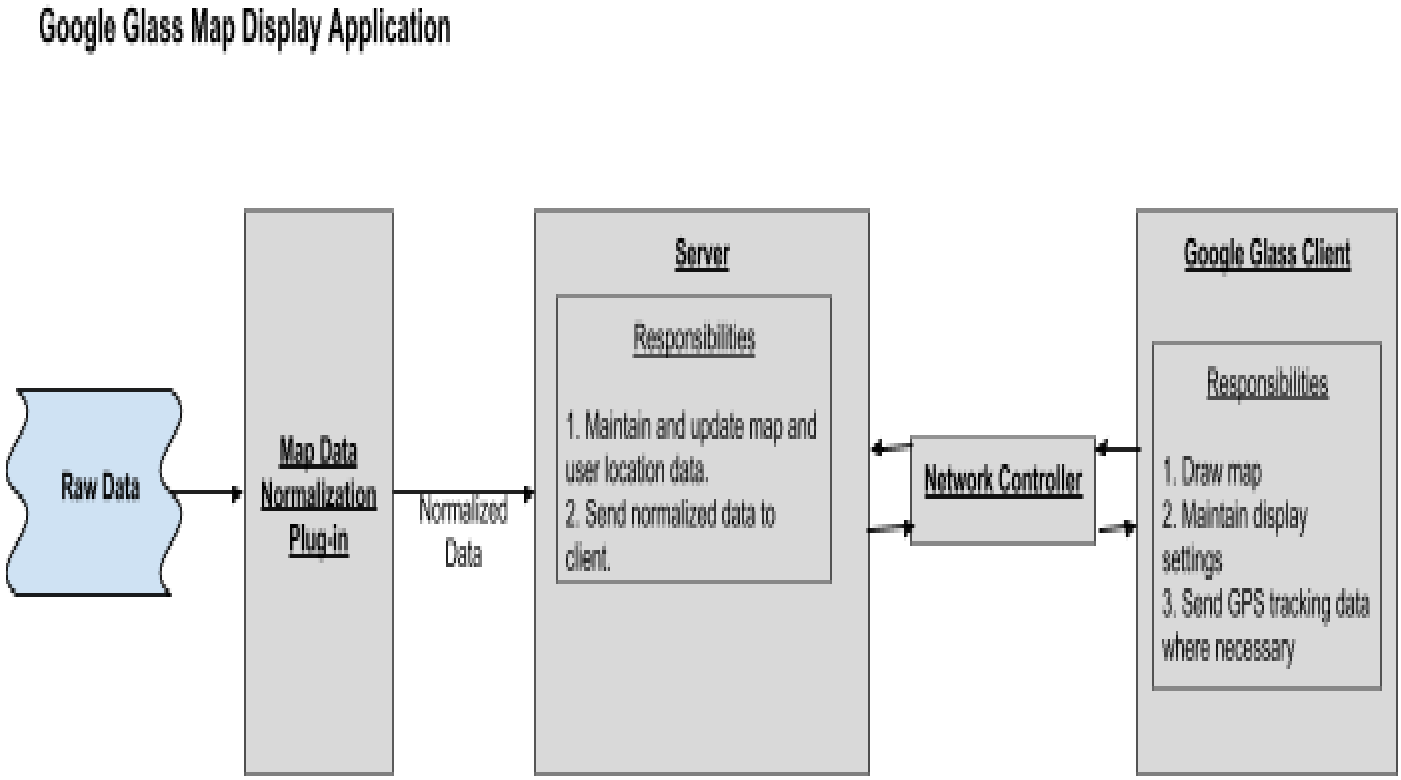
Our system is going to be different from other applications because there is going to be a server that builds the map objects before the map is rendered on the device. The server will receive GPS data from a variety of resources and build the map data from those resources. It will then send via TCP the map data to the Glass to be rendered.

Logical Design

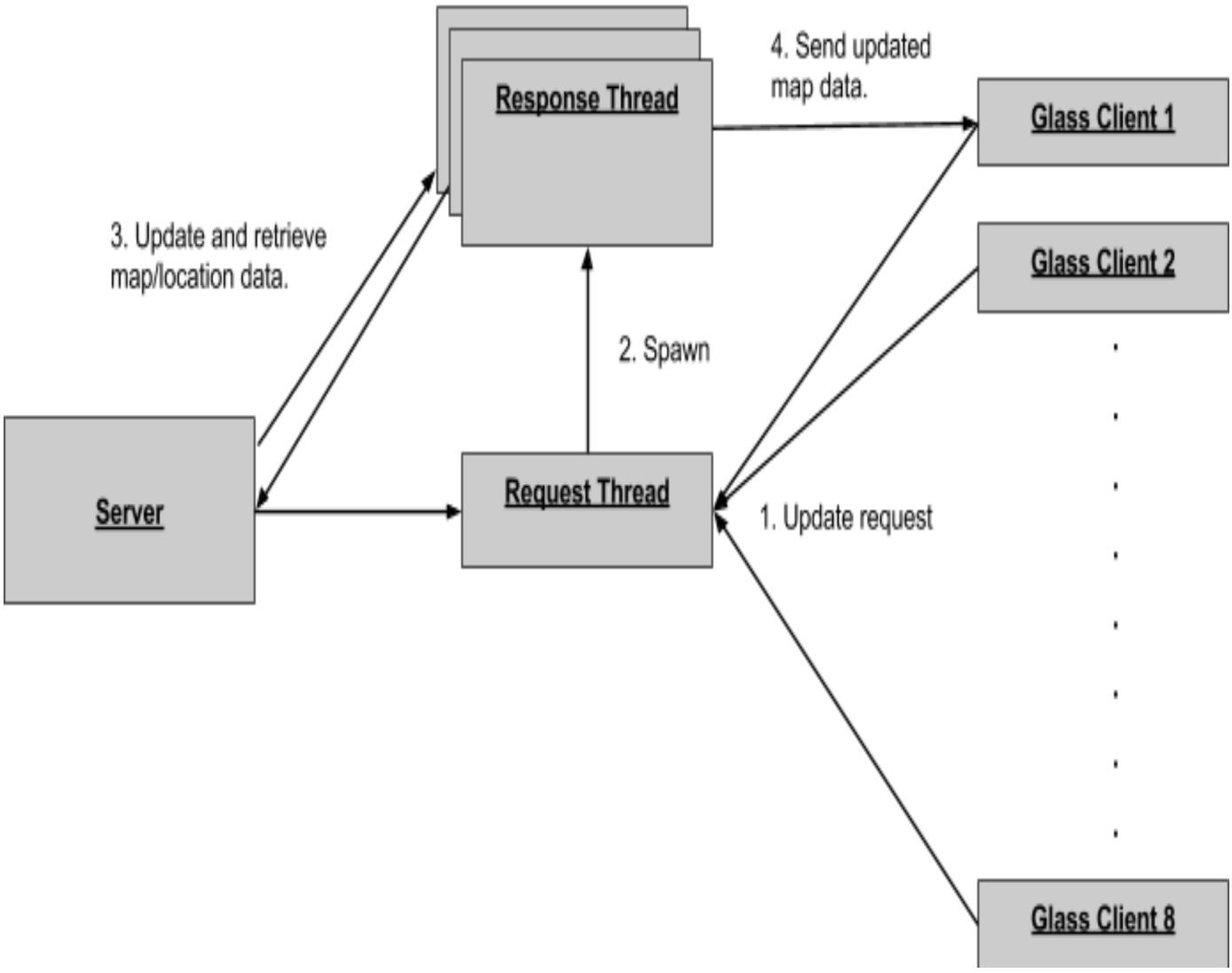
The Google Glass application needs to be very easy to use. Our users will most likely need to have their hands free and will not be able to control the application other than turning it on and possibly sending a new message.

The application will be designed so that the only interaction with the user is if they want to record a new message to send to base. This action will be started by tapping any part of the glasses with any object. The rest of the application will work without any user interaction.

Block diagrams of the concept



Network Controller



Detailed description

Input/Output Specifications

1. Input
    1.1. To users
        1.1.1. audio data
        1.1.2. gps location
        1.1.3. orientation
        1.1.4. mapping data
    1.2. To server
        1.2.1. user location
        1.2.2. user orientation
        1.2.3. user audio
        1.2.4. building/barrier location
2. Output
    2.1. From users
        2.1.1. audio data
        2.1.2. gps location
        2.1.3. orientation
    2.2. From server
        2.2.1. other users gps location
        2.2.2. other users orientation
        2.2.3. building/barrier location
        2.2.4. mapping data
        2.2.5. other users audio data
    2.3. From mirage
        2.3.1. users location
        2.3.2. building/barrier location
    2.4. From GPS satellites
        2.4.1. user location
        2.4.2. user orientation

Interface specifications

Server Specifications

Action	**Interface**	**Methods Used**
Receive user location	Server Module TCP Thread	listenForLocation()

from Android device		exception: No connection to device
Receive audio from Android device	Server Module TCP Thread	listenForAudio() exception: No connection to device
Receive location data from Mirage	Server Module TCP Thread	listenForLocation() exception: No connection to device
Send map build to Glass	Server Module TCP Thread	sendMap()
Send audio to glass	Server Module TCP Thread	sendAudio()

Client Specifications

Action	**Interface**	**Methods Used**
Open Application	Android Launcher Screen (Click on application)	onCreate()
Send message	Main Activity of Audio app (Tap on Glass)	sendServerMessage()

Hardware/software specifications

1. Google Glass Client
    1.2 Android Compatible Software
2. Server
    2.1 Java Runtime Environment
    2.2 Java-based GUI
3. Location Provider
    3.1 GPS Satellites via Glass or other sensors
    3.2 Mirage Camera System
        3.2.1 Motion Analysis Tracking
        3.2.2 AR-Tracking Cameras

Simulations and modeling

- Audio Communication Prototype

Implementation Issues and Challenges

Issue: Maintaining near real-time response times
Issue: Getting specific enough GPS info in real-world implementation

Testing, procedures and specifications

Our testing procedure will be based on agile development workflow. We will make incremental changes to the project over time, adding features and fixing bugs. This process will therefore be repeated continually as we build our application. The general approach will be:

1. Design Component
2. Test Component
    2.1 Usability Testing
    2.2 Software Mock Tests
3. Repeat steps 1 and 2 until satisfactory
4. Implement Component

The the ‘component’ will usually be a piece or module of the project. There are times during the process where it will be a more broad aspect of the Application, such as the architecture, or communication. In step one we will refactor existing code or add new code to introduce new functions or fix bugs.
The testing process, step two, will consist of usability or software testing, depending on which would be more appropriate. Usability testing will consist of running the module on a device and ensuring that it works from a user’s point of view. We may also make use of software mocks to provide sample data and scenarios to specific components, using either custom Java classes or Unit testing software. We will want to do usability and software tests for anything involving sensor data in order to verify the accuracy of the sensor and our implementation.
The ‘satisfactory’ level does not have concrete specifications due to the nature of our project. The level will be determined based on the capabilities of the Glass unit. We will maintain close contact with our client to ensure that our results from testing are in line with the project goals.
In step 4, we will take the new code

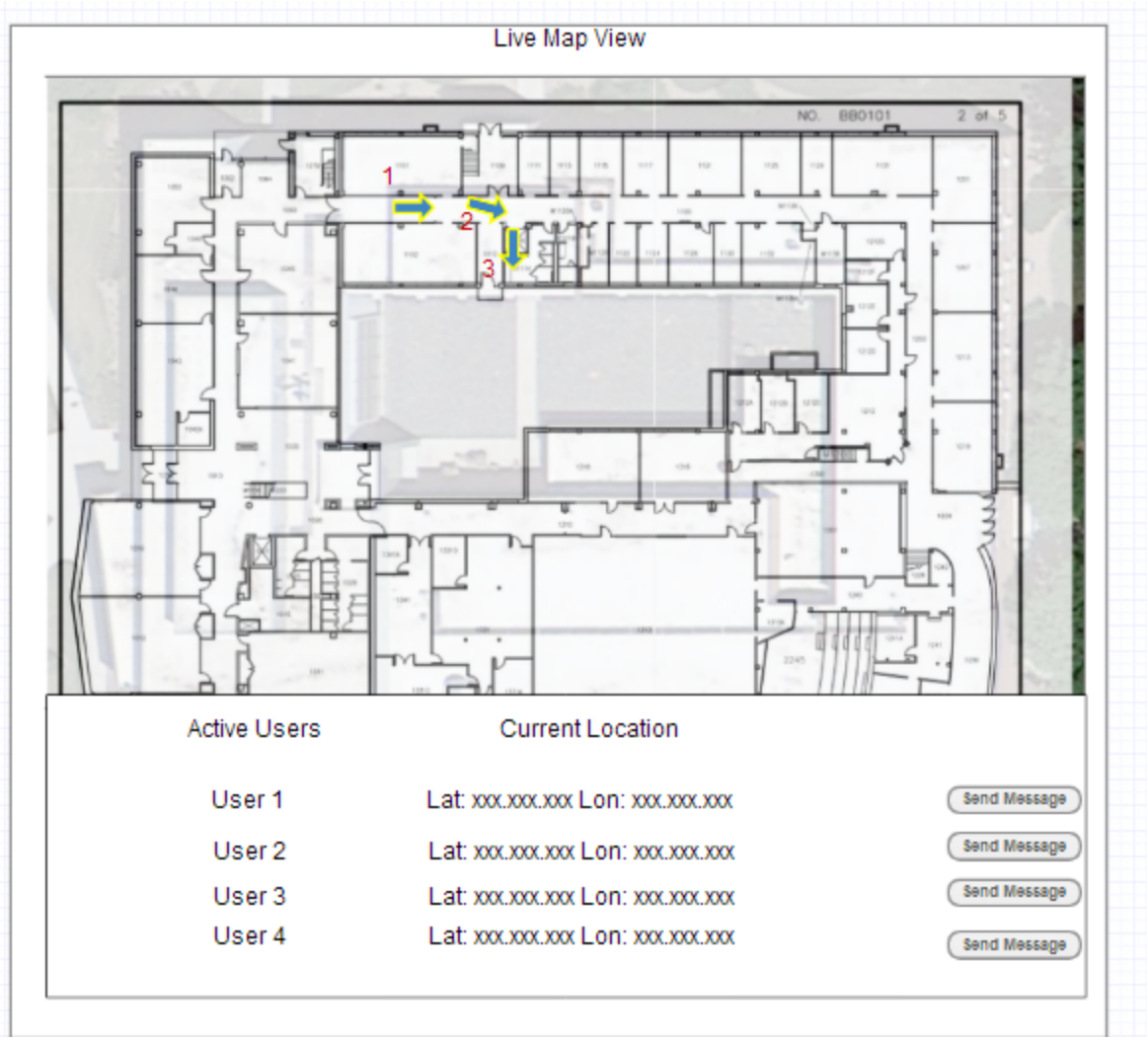
Use Case Diagrams, Use Cases, Module Diagrams

Glass UI Screen

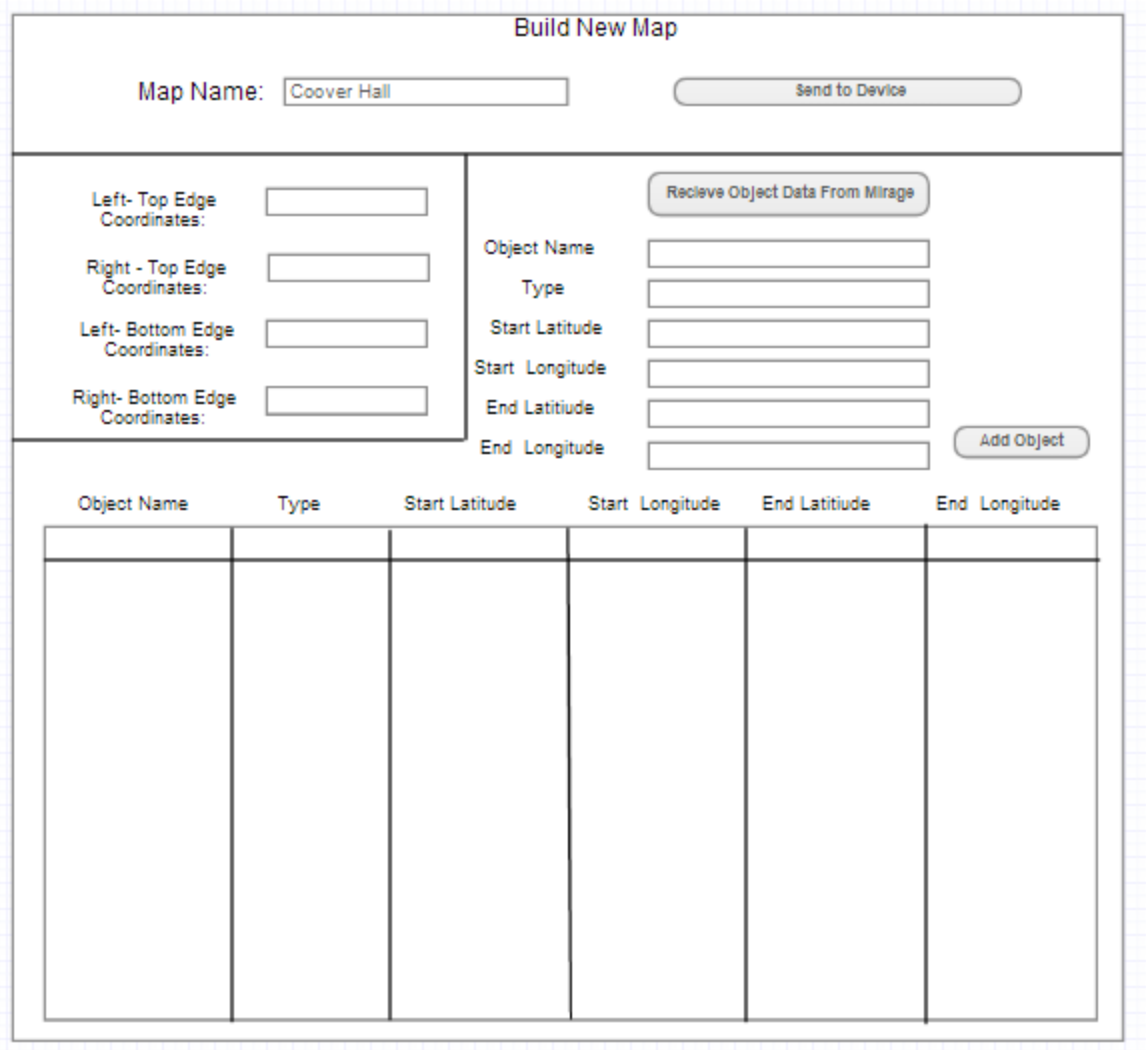
The UI screen of the glass will display the current building floor plan and the current users location. It will also display all other team members locations in the building. The user can tap the glass to record and send a voice message back to the server.

Java Module UI Screens

View users:



Build New Map:



Edit Existing Map: