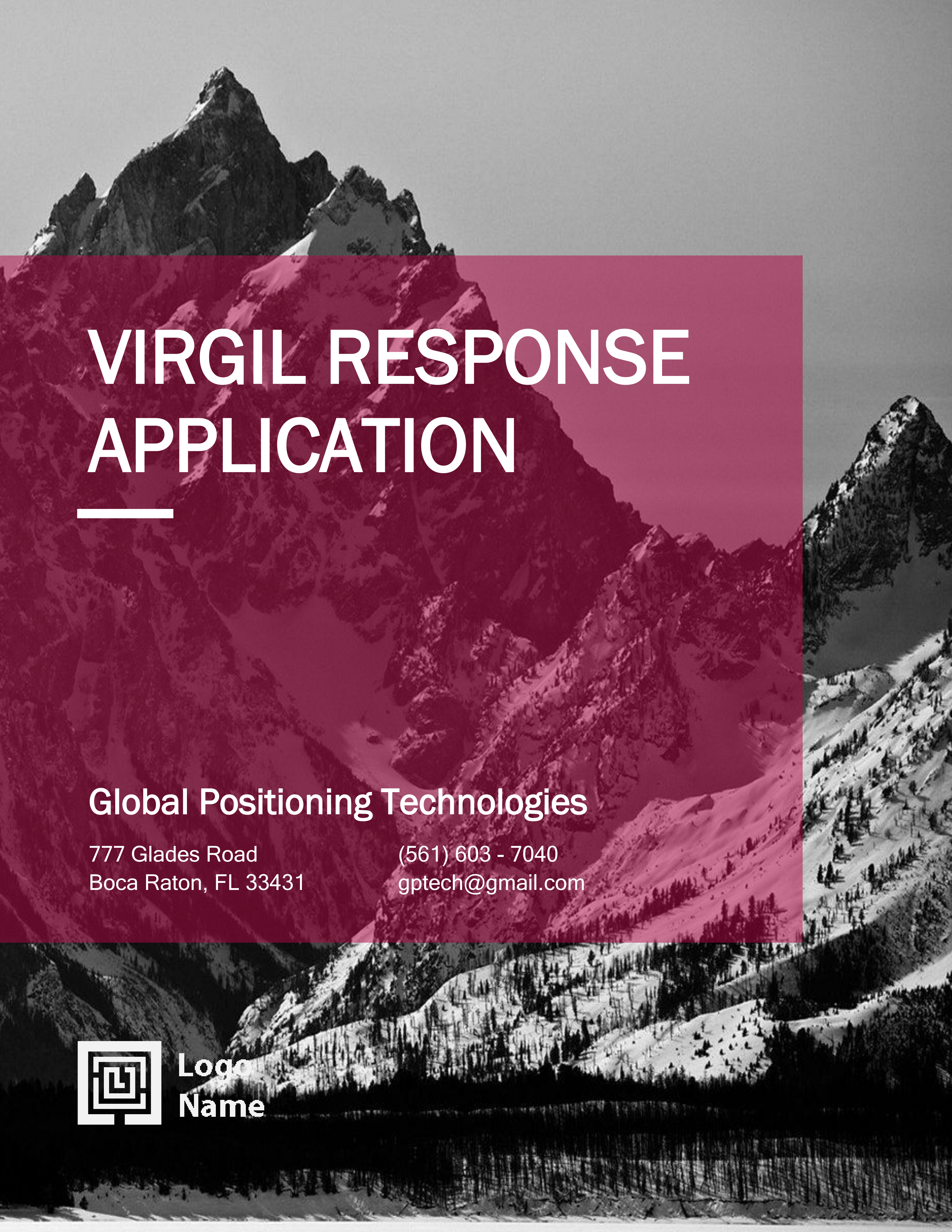


VIRGIL RESPONSE APPLICATION

Global Positioning Technologies

777 Glades Road
Boca Raton, FL 33431

(561) 603 - 7040
gptech@gmail.com

Logo
Name

Table of Contents

Vision Statement

Main Goal: Create an application that will use GPS satellite tracking to ping people's phones when they are in dire need of emergency services. The need is for those who are lost, trapped after a natural disaster, or in situations where human contact is needed with emergency first responders like the sheriff department and/or fire & rescue.

The Problem: In today's modern world, almost every person on the planet has a cell phone or a smart phone. From children to senior citizens, cell phones are becoming the main use of interaction between person to person. But what happens, when an emergency service is needed? A user may call emergency services, but what happens when they are not in range of a cell tower, or their calls are unable to be sent out? What happens after a user tries to relocate or is trapped and cannot get to an exit, or a way for someone to reach them?

The Solution: To create an application where a user may have their cellphones pinged via the global positioning system that a cellphone is connected to. A phone with the location on, will have their locations tracked via satellite in orbit of Earth. A user would not have to be in range of a cell tower for this signal to be tracked and would be helpful for those who are in dire need of emergency services. The user will also be able to send a profile about themselves, which includes information about themselves, as well as a picture that helps emergency services to find them.

Concept: The concept comes from the technology that low-wing and high-wing aircraft use, like the black-box, where when an impact is felt, a beacon is sent where satellites pick up on the distress signal. The distress signal allows for emergency services, the FAA, and the NTSB to come out and search for the lost aircraft. This concept would be applied to a user's cellphone, as well as creating a mapping system that allows other users in the area to pick up on the signal. In a way, a safety web will be created so that others can help for the cause.

Group Members

Stakeholder Definitions

A. Mobile Cellphone Users
   a. These users are separated into two groups:
      i. There are users who shall need the application when they become lost.
      ii. There are users who can use the application to help locate users who might be lost in the area around them.

B. Emergency Response Units
   a. The emergency response units will be that of anyone that is part of a first-response emergency crews. This includes, but is not limited to:
      i. Police/Sheriff Departments
      ii. Medical Response
      iii. Firefighters
      iv. Search & Rescue
      v. Maritime Operations*
         1. Not included in base application concept design but could be included.

C. IT Staff
   a. Anyone who is related to the technical and hardware development of the application must have access to the design. They are responsible for maintaining all aspects of the application.

Incremental Developmental Process Model Statement

At first, as a team we thought that to have a steady representation of our product, we could go with the Waterfall Process model as we had ideas in mind. The core process allowed us to think ahead of our design and have certain aspects of the product finished. We could start a development process, to a backend design, and then to the front-end, and finishing off with a testing environment that could help with finish prototyping. This all sounds handy when it came to a checklist, but in fact we soon realized that the Incremental Development process would be better for our application.

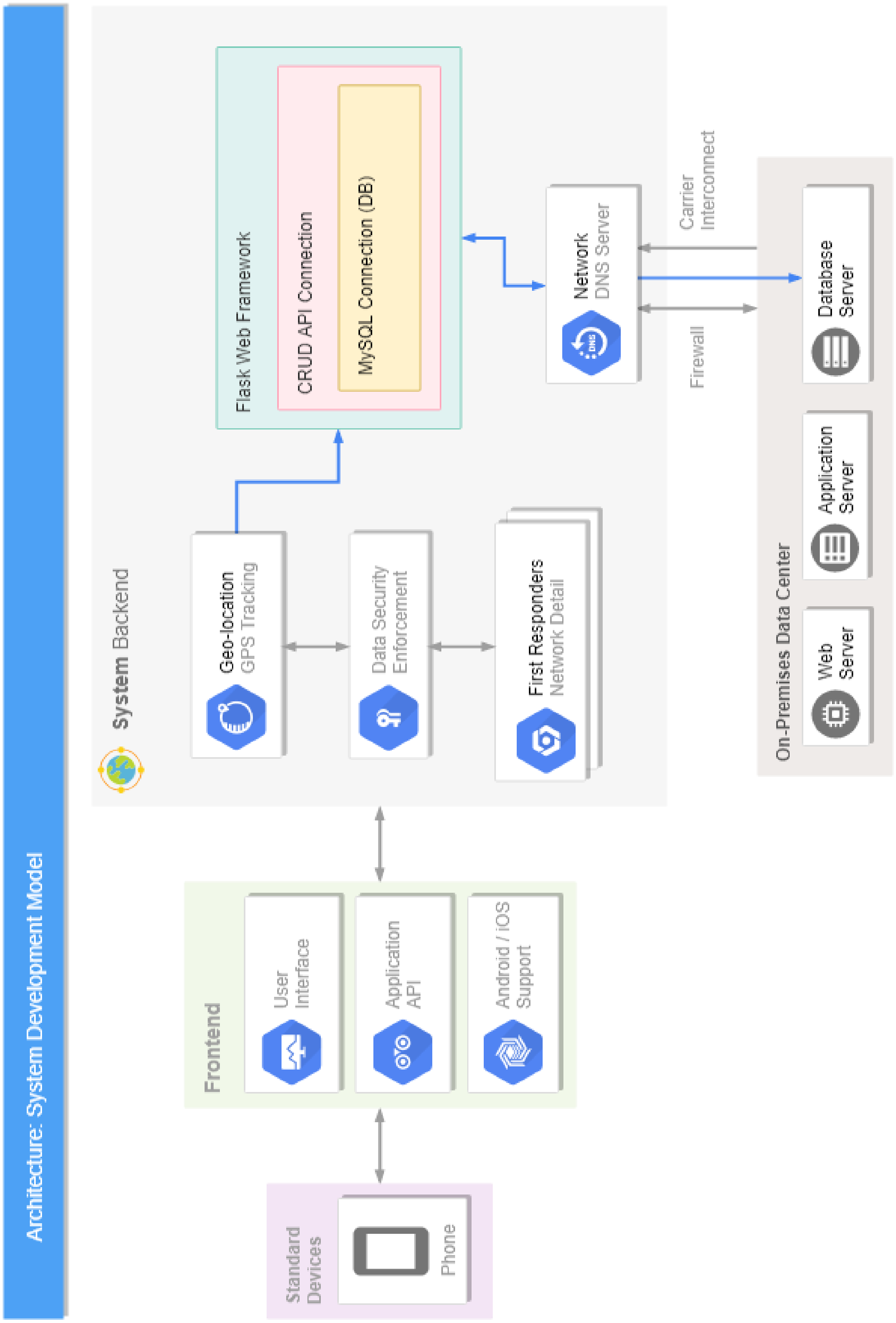
The Waterfall Process model allows for us as a team to have an early onset of what we want in mind. We could finish a step and then move onto the next step, but it would be a hassle if we ended a step and wanted to go back. We want to create an application where we could add parts in over time; the Waterfall Process would not be ideal for this type of setup. We looked over at the Integration and Configuration Process model, but we saw it as an application process model where a completed system would be already created, and we would later modify it. There would have to be a system already at play, thus allowing us to configure it for the consumer, but our idea was more based on the creation of an application from scratch. We could integrate parts here and there, and then configure it all together, allowing us to pace ourselves, but a managerial position, I felt like this was out of our scope. We could add features that we wanted, but in the end, it might cause us to future feature creep that could land us in hot water as the deadline circled in.

In the end, we decided to embed ourselves in with the use of the Incremental Development Process model because of the ability to be able to slowly add portions of features into a unit as we went along. Currently, our application has begun from scratch, from the backend, to even the front-end of the application. The concept is born from a system that exists elsewhere for different types of consumers. Since the application does not exist for a ground level consumer, someone who is out of the scope of the original concept design, these consumers would benefit from our design more-or-less from the original concept. So, we wanted to go out and see how we can redesign a model for our new consumer. The benefits of this type of process model is that we can steadily add new features to the application and the overall design. Instead of overwhelming ourselves with feature creeping, we can add in features that could benefit the consumer at a very basic level. From there, once we have our first prototype, and we feel that the prototype is lacking, we can add more to it. It the process allows us to add and remove the same features, if we make sure that the consumer can have the right application in hand for the scenario that the application would be used for.

The scenario of creating the application is that each team member has a part in building the application, but that so in a way that the team can work in a group conjunction to design the application at a set time. For this, there are two to three members who work on the back-end and another two to three members who work on the front-end. Once both groups have accomplished a satisfactory condition, both groups unite and work together to form the middle-ware of the application. The Incremental Development Process allows for this type of connection to work because we add to it as we steadily come together to form the final product. In the end, everyone works together, and if there are questions that need to be answered, we work in conjunction with each other to make sure that both groups don't have any misunderstandings as to how the application is supposed to work. There must be a connection in between each other, just like the application from the back-end to the front-end, because without it, it can fall flat.

Overall, even though there are positives for the chosen process model, there are side effects like that of time and being able to create more than one prototype before the deadline. We would want to have more than one prototype created, but as time moves on, we must steadily do everything right so that there aren't any misplaced steps in the development. Time is a key factor, and even though we might want to add every single idea into our application, we will have to decide to not add in features because of time constraints and programming efficiencies. Testing, another important key factor will probably run around at the same time as that of final prototyping. We don't want to feature creep, but at the same time, we wouldn't want the user to feel that what they get is also sloppy in design. In the end, if we dedicate to a great user experience and application design, we can get the most out of our application development.

System Architecture Model



User Requirements with System Requirements

User Requirements:

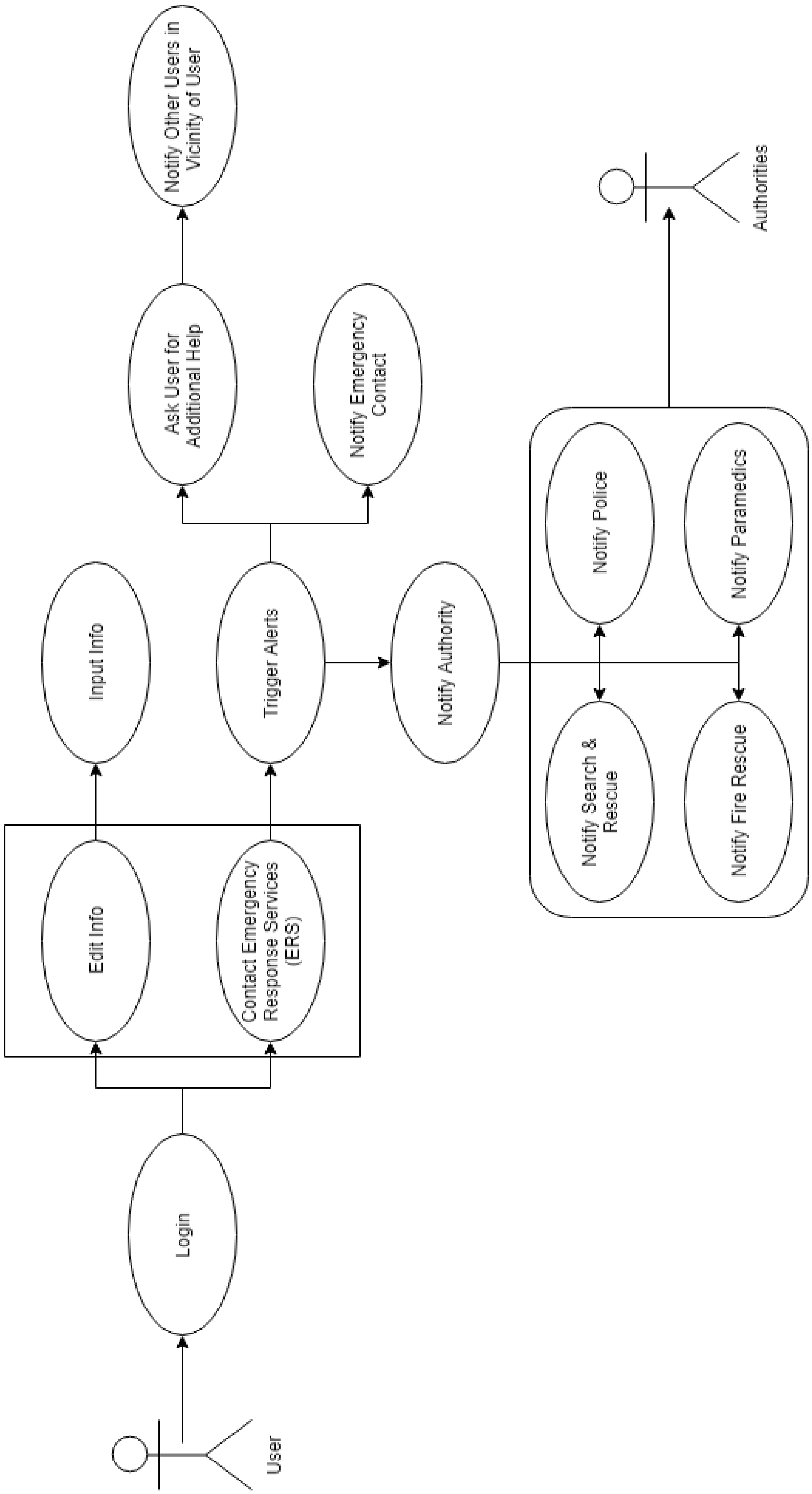
Writing this as if we were to persuade the consumer to download the application.

- If a user gets lost anywhere in the United States and needs to contact emergency services, where a user does not know where they are, the application shall relay GPS information to emergency services.
- User can add in additional information about what they look like, who they are, medical conditions, and a picture so that emergency response units know who they are looking for.
- At the press of a button, application shall send information about their location as longitude and latitude coordinates that emergency response units can use to track a user down.
- A user will have the priority if they want their information shared to emergency response units and any other people that might be in the area.
  - If someone else with the application is in the area, they will be able to volunteer themselves to help locate the missing user.

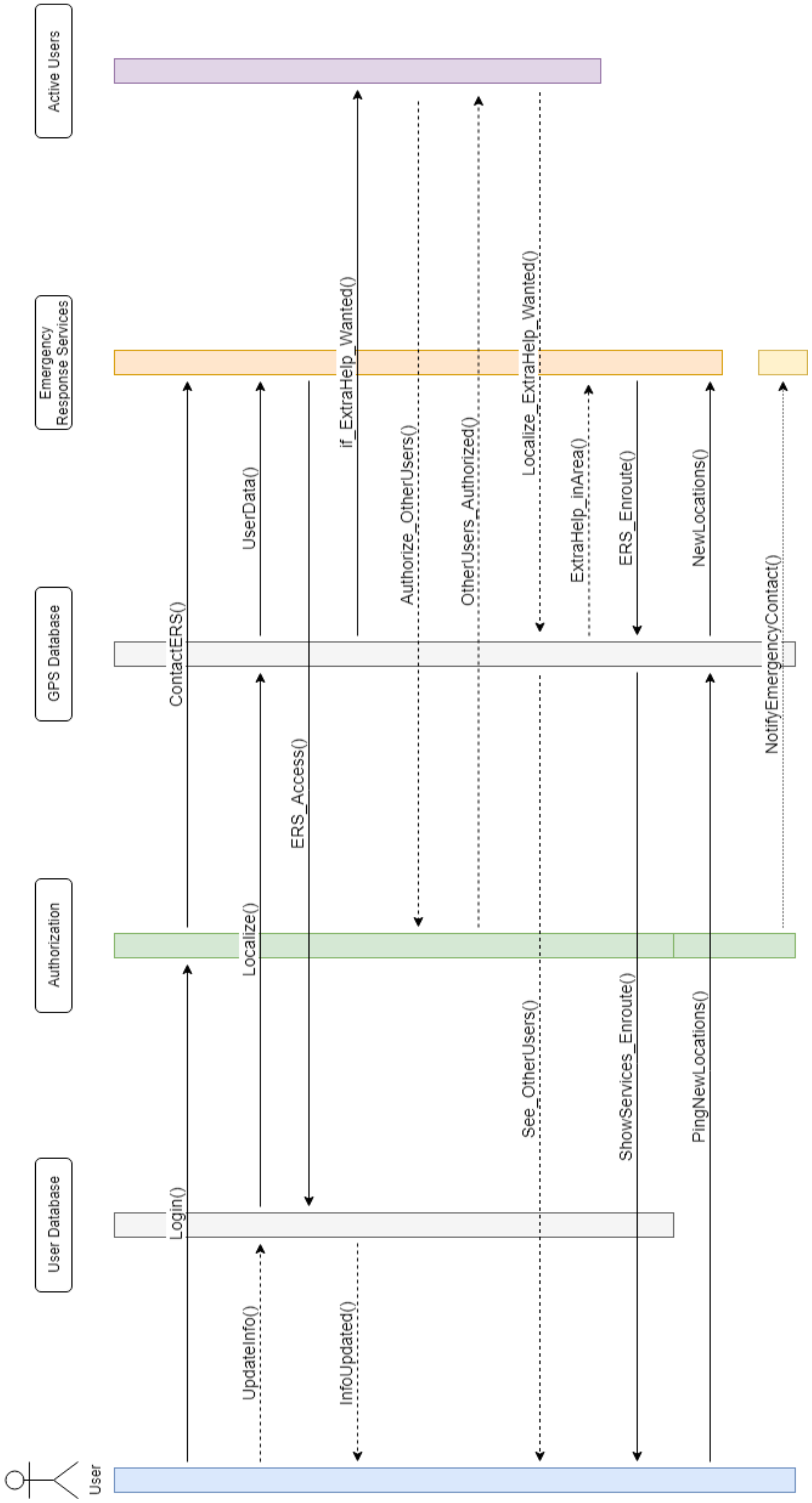
System Requirements:

- System must be able to store login information from the user so that a user may be able to login with their personal information.
- Personal information shall be securely hashed so that no one without the right clearance can get access to the information.
- The application shall know the longitude and latitude coordinates of any user that is seeking emergency services.
- Application security is a must and it must be secured carefully so that no malicious intent can cause harm to either the application system, the user using the application, emergency response units, and the IT staff.
- Application shall be able to connect to emergency services as soon as possible. No hang-ups should occur while the system is dialing between the lost user and the emergency services.
- Users connected to the system to volunteer must appear near the user.
- User must know when emergency services are arriving or are around the lost user.
- User must have an option as to what information can be shared, and what cannot be.

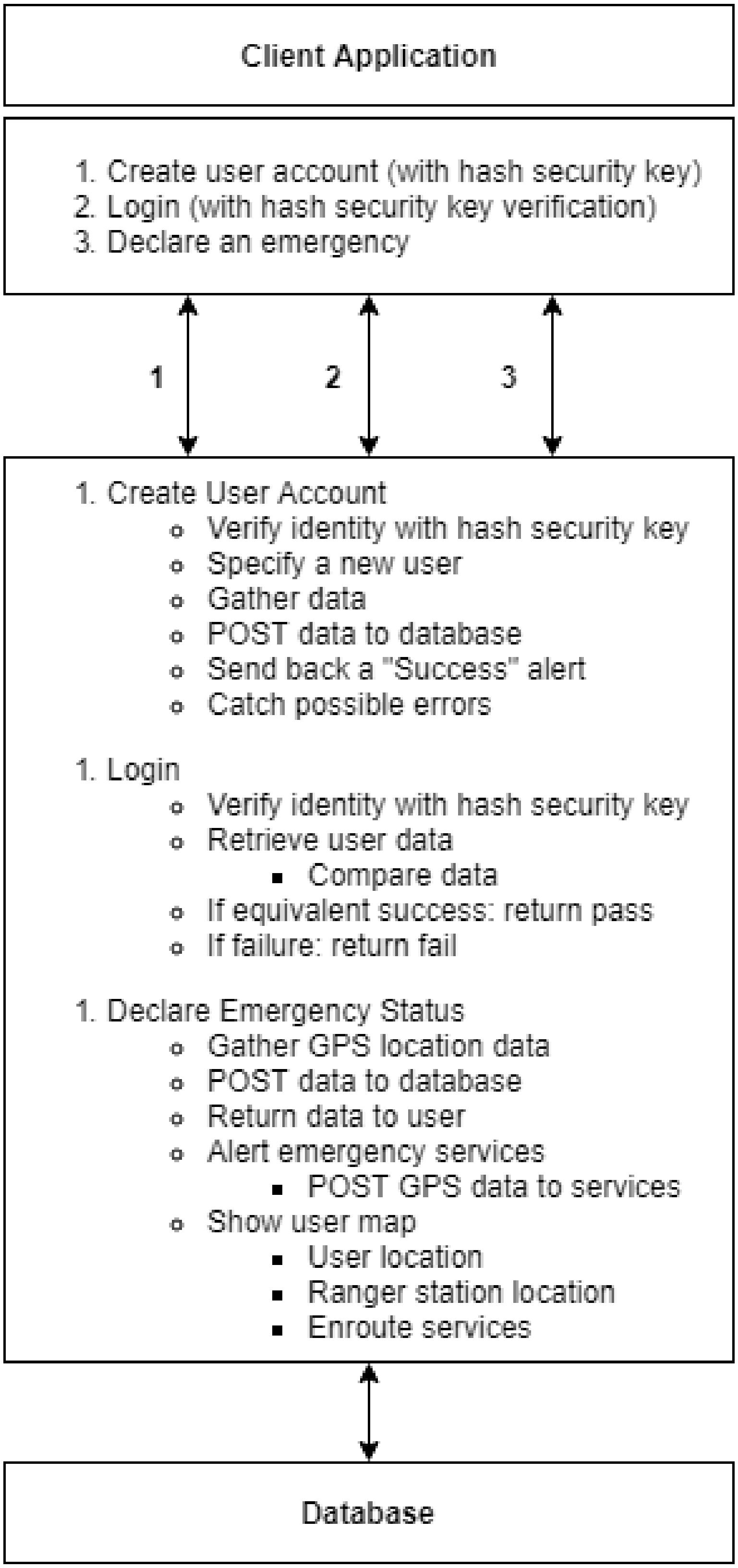
Use Case Diagrams and Descriptions



Sequence Diagrams



Class Diagrams



Client Application

1. Create user account (with hash security key)
2. Login (with hash security key verification)
3. Declare an emergency

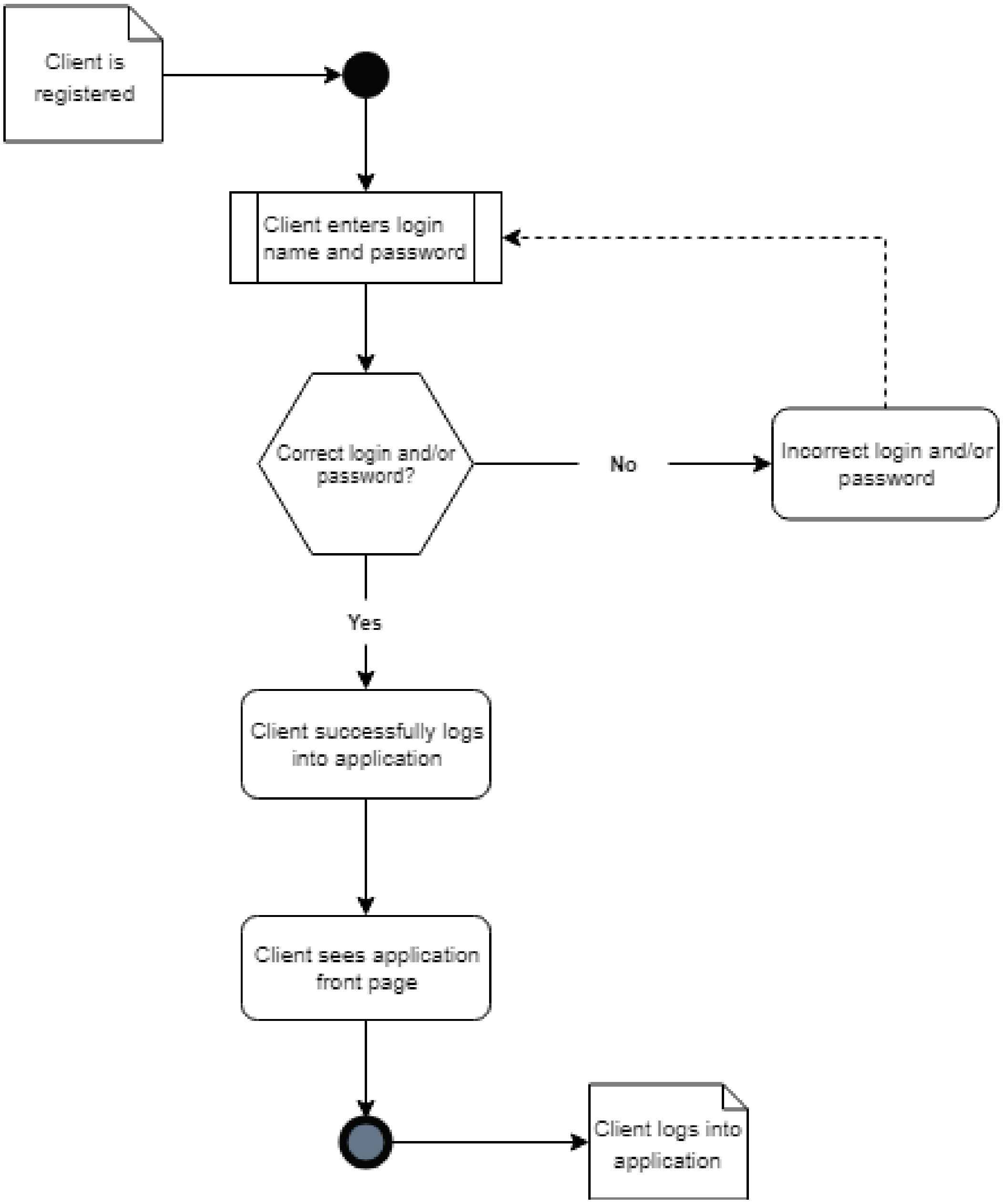
1. Create User Account
   - Verify identity with hash security key
   - Specify a new user
   - Gather data
   - POST data to database
   - Send back a "Success" alert
   - Catch possible errors

1. Login
   - Verify identity with hash security key
   - Retrieve user data
     - Compare data
   - If equivalent success: return pass
   - If failure: return fail

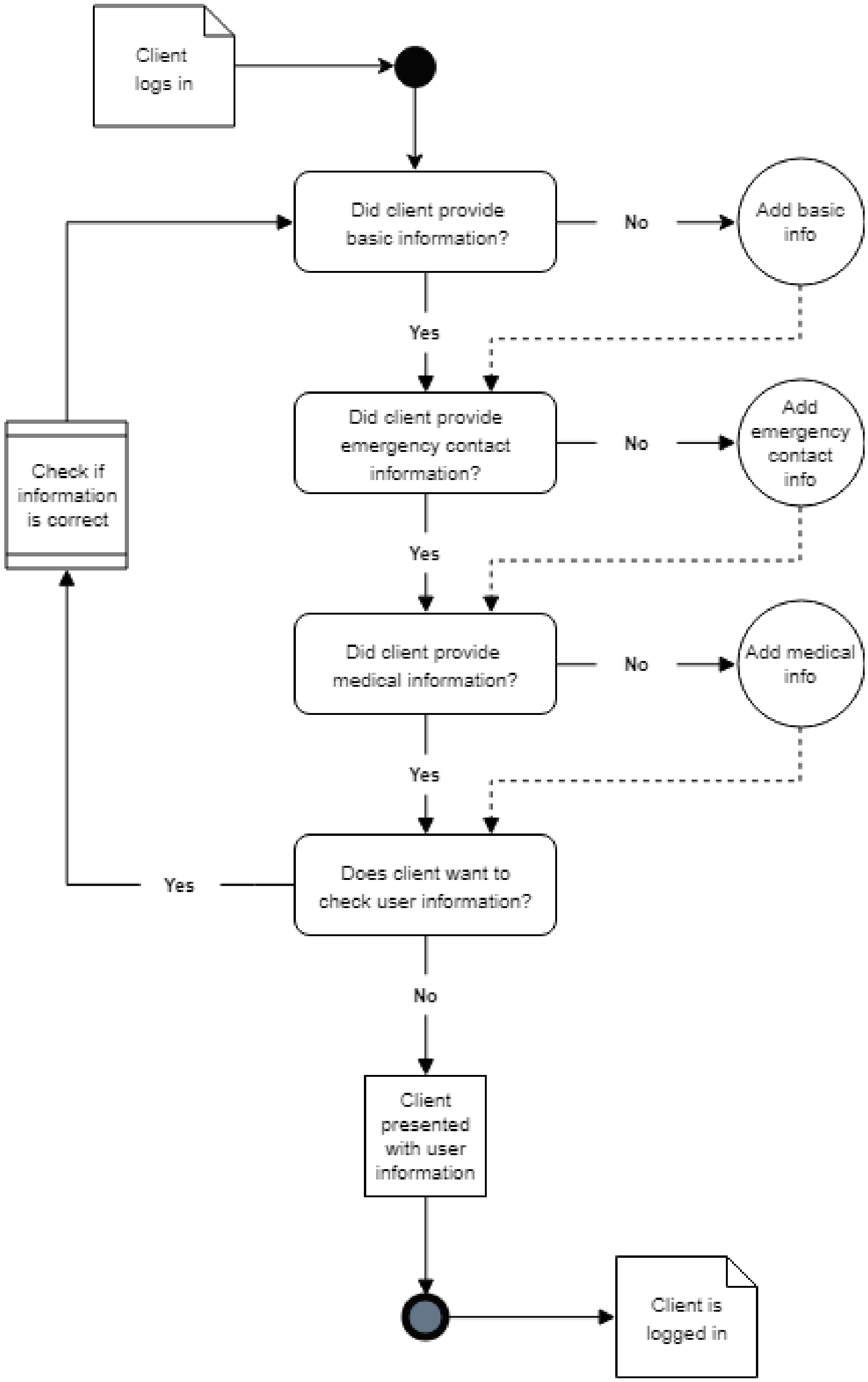
1. Declare Emergency Status
   - Gather GPS location data
   - POST data to database
   - Return data to user
   - Alert emergency services
     - POST GPS data to services
   - Show user map
     - User location
     - Ranger station location
     - Enroute services

Database

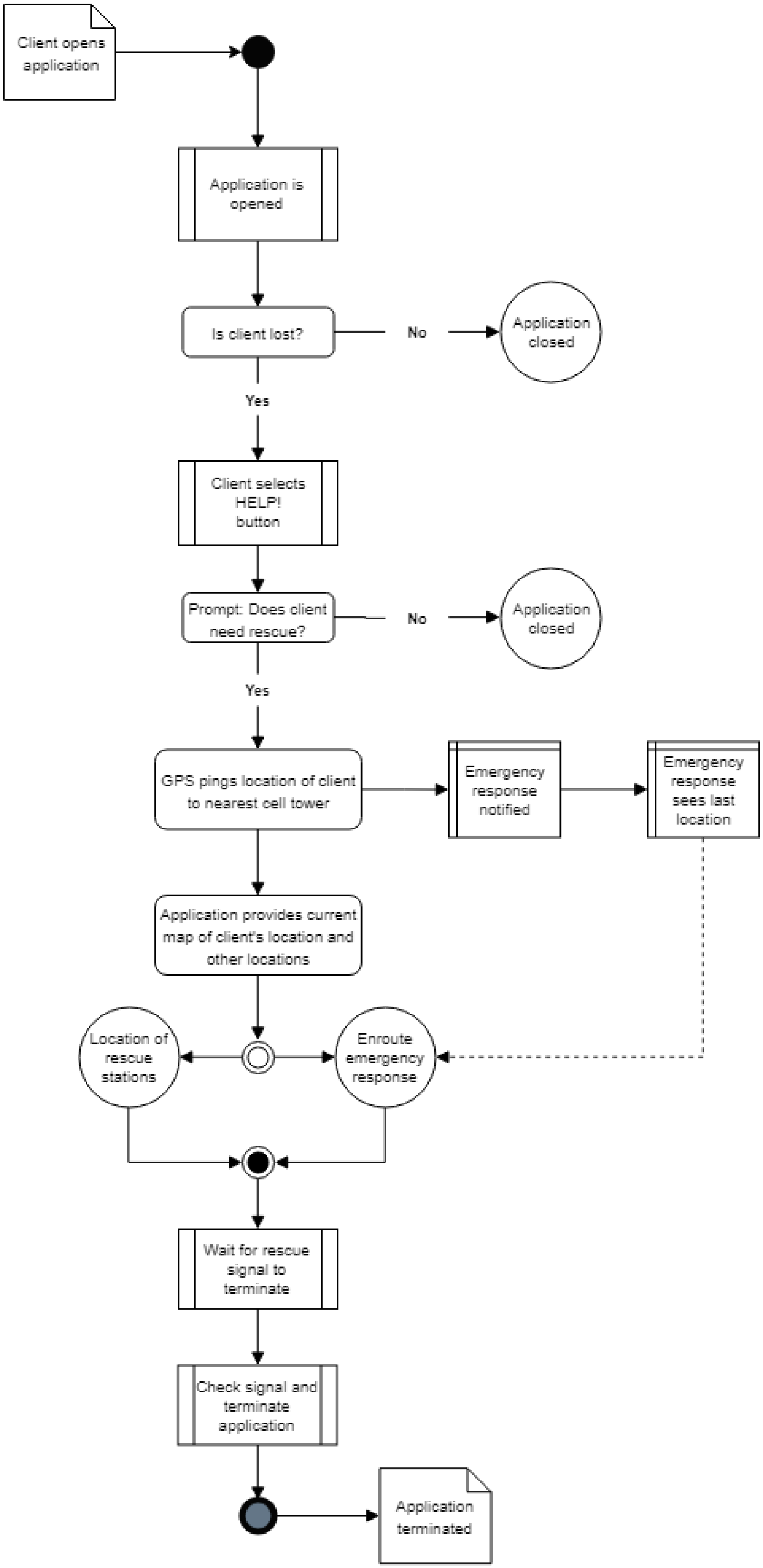
Activity Diagram


Client is registered
Client enters login name and password
Correct login and/or password?
No
Incorrect login and/or password
Yes
Client successfully logs into application
Client sees application front page
Client logs into application

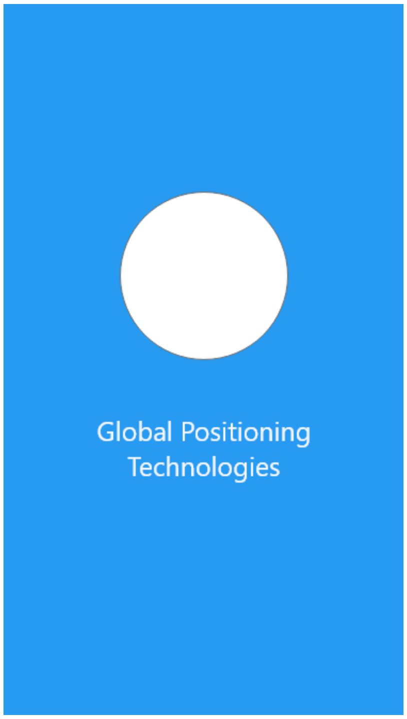
Activity Diagram

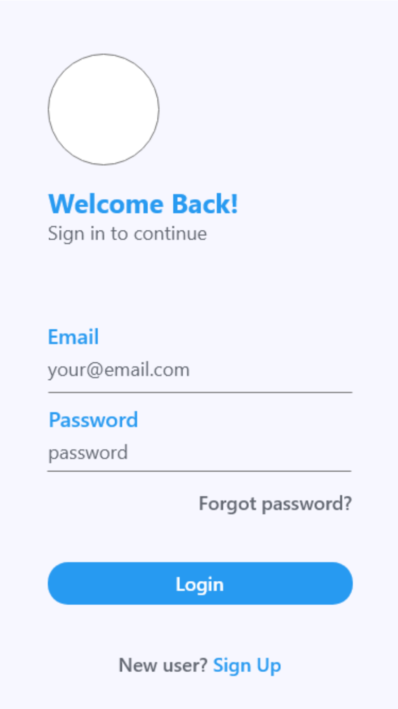

Client logs in
Did client provide basic information?
No
Add basic info
Yes
Did client provide emergency contact information?
No
Add emergency contact info
Yes
Did client provide medical information?
No
Add medical info
Yes
Does client want to check user information?
Yes
Check if information is correct
No
Client presented with user information
Client is logged in

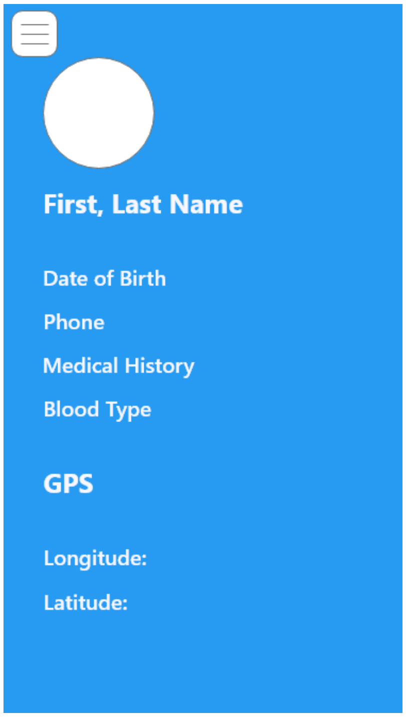
Activity Diagram

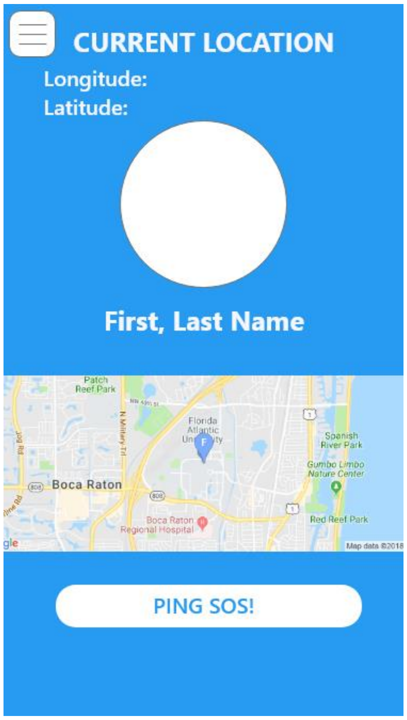

Client opens application
Application is opened
Is client lost?
No
Application closed
Yes
Client selects HELP! button
Prompt: Does client need rescue?
No
Application closed
Yes
GPS pings location of client to nearest cell tower
Emergency response notified
Emergency response sees last location
Application provides current map of client's location and other locations
Location of rescue stations
Enroute emergency response
Wait for rescue signal to terminate
Check signal and terminate application
Application terminated

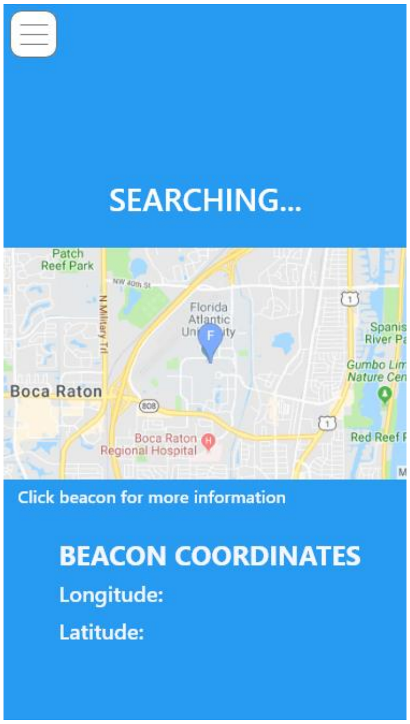
User Interface Diagrams



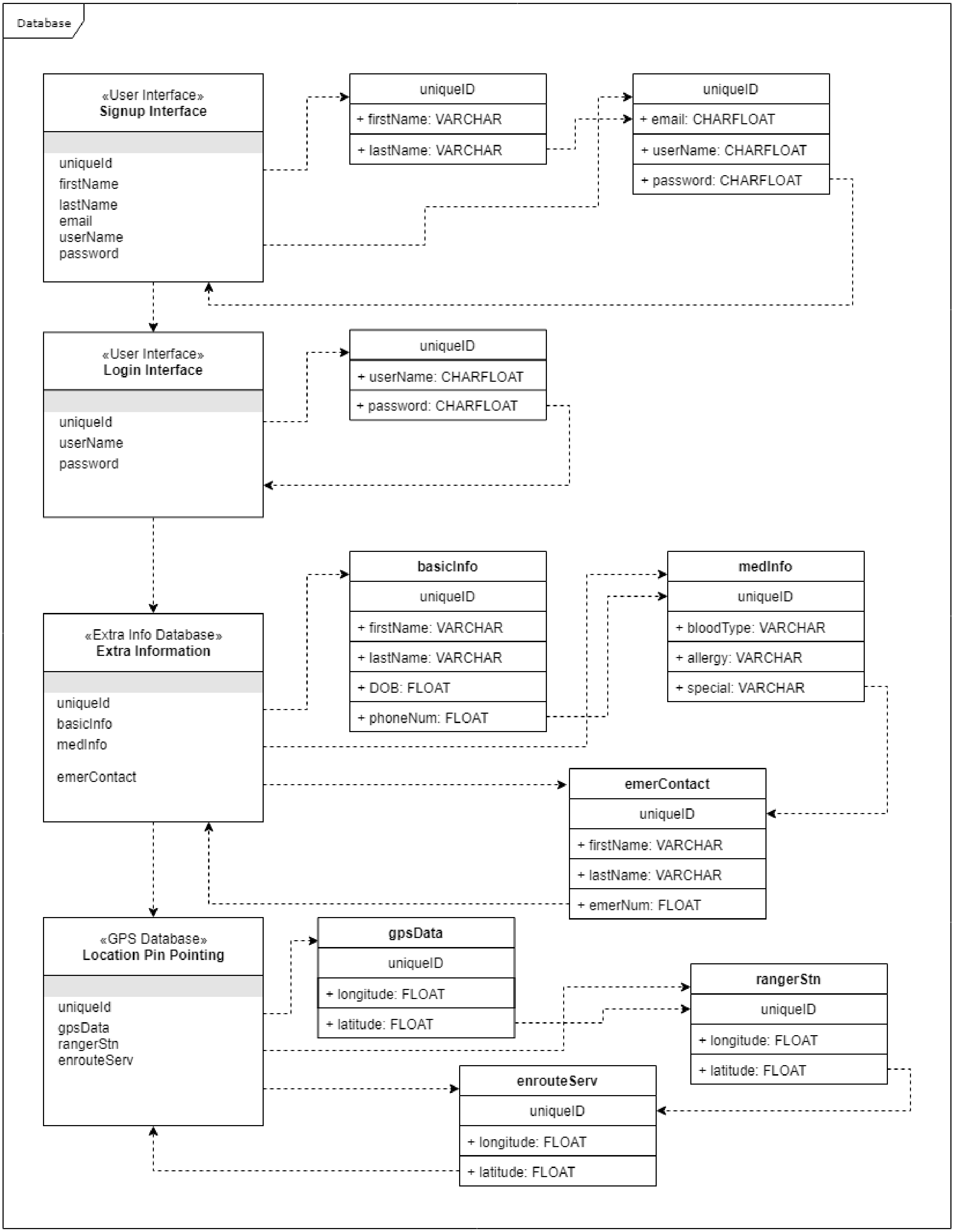








Database Design Diagram



Distributed Model Statement

Security Statement

Glossary

Poster