

CRC Cards

2	CRC Cards
7	CRC Model
9	System Interaction and Environment
9	System Architecture
10	System Decomposition
11	Design Mockup

1. AndriodApp

Parent Class: None

Subclasses: None

Responsibilities:

- Uses navigator to navigate through the app

Collaborators:

- Navigator

2. Navigator

Parent Class: none

Subclasses: None

Responsibilities:

- Controls the user's access
- Controllers the view displayed for the user
- Transitions to different pages of the app

Collaborators:

- User
- Map
- Login
- UpdateInfo
- Register

3. Login

Parent Class: None

Subclasses: None

Responsibilities:

- Authenticates user, upon filling in the sign in form
- Gets a user given the correct username and password

Collaborators:

- User
- Navigator

4. Register

Parent Class: None

Subclasses: None

Responsibilities:

- Create a user account upon filling in user information
- Create a professional account upon filling in user and professional information

Collaborators:

- User
- Professional
- Navigator

5. User

Parent Class: None

Subclasses: None

Responsibilities:

- Knows its username and password
- Knows its location
- Knows its medical history
- Knows its messages
- Can send an SOS signal to professionals
- Can send messages to other users

Collaborators:

- SendSOSDAO
- MessageDAO
- Message
- Navigator

7. Message

Parent Class: None

Subclasses: None

Responsibilities:

- Knows its timestamp
- Knows its message
- Knows the user who sent the message

Collaborators:

- User

8. UpdateInfo

Parent Class: None

Subclasses: None

Responsibilities:

- Contains data of user settings
- Updates user authentication
- Updates user's medical history

Collaborators:

- User
- Navigator

9. Map

Parent Class: None

Subclasses: None

Responsibilities:

- Display location of both user and professional once an SOS signal is accepted
- Provides directions from professional to user once an SOS signal is accepted
- Displays all nearby user locations for the professional

Collaborators:

- User
- SendSOSDAO
- ReceiveSOSDAO

10. SendSOSDao <<Interface>>

Parent Class: None

Subclasses: None

Responsibilities:

- Finds all professionals nearby given a user's location
- Sends an SOS signal to all nearby professionals given a user's location

Collaborators:

- User

11. ReceiveSOSDAO <<Interface>>

Parent Class: None

Subclasses: None

Responsibilities:

- Gets all SOS signals given a professional's location

Collaborators:

- User

12. MessageDAO <<interface>>

Parent Class: None

Subclasses: None

Responsibilities:

- Sends a message to a user given their username
- Receives a message from a user

Collaborators:

- User
- Message

CRC Model

AndriodApp	
<ul style="list-style-type: none"> - Uses navigator to navigate through the app 	Navigator

Login	
<ul style="list-style-type: none"> - Authenticates user, upon filling in the sign in form - Gets a user given the correct username and password 	User Navigator

User	
<ul style="list-style-type: none"> - Knows its username and password - Knows its location - Knows its medical history - Knows its messages - Can send an SOS signal to professionals - Can send messages to other users 	SendSOSDAO MessageDAO Message Navigator

Message	
<ul style="list-style-type: none"> - Knows its timestamp - Knows its message - Knows the user who sent the message 	User

Map	
<ul style="list-style-type: none"> - Display location of both the user and professional once an SOS signal is accepted - Provides directions from the professional to the user once an SOS is accepted - Displays all nearby user locations for the professional 	User Navigator ReceiveSOSDAO SendSOSDAO

<<Interface>> ReceiveSOSDAO	
<ul style="list-style-type: none"> - Gets all SOS signals given a professional's location 	User

Navigator	
<ul style="list-style-type: none"> - Controls the user's access - Controls the view displayed for the user - Transitions to different pages of the app 	User Map Login Update Info Register Andriod App

Register	
<ul style="list-style-type: none"> - Create a user account upon filling in user information - Create a professional account upon filling in user and professional information 	Navigator User

UpdateInfo	
<ul style="list-style-type: none"> - Updates user authentication - Updates user's medical history - Contains data of user settings 	User Navigator

<<Interface>> SendSOSDAO	
<ul style="list-style-type: none"> - Finds all professionals nearby given a user's location - Sends an SOS signal to all nearby professionals given a user's location 	User

<<Interface>> MessageDAO	
<ul style="list-style-type: none"> -Sends a message to a user given their username -Receives a message from a user 	User Message

System Interaction and Environment

The technologies QuickHealth will be using will include:

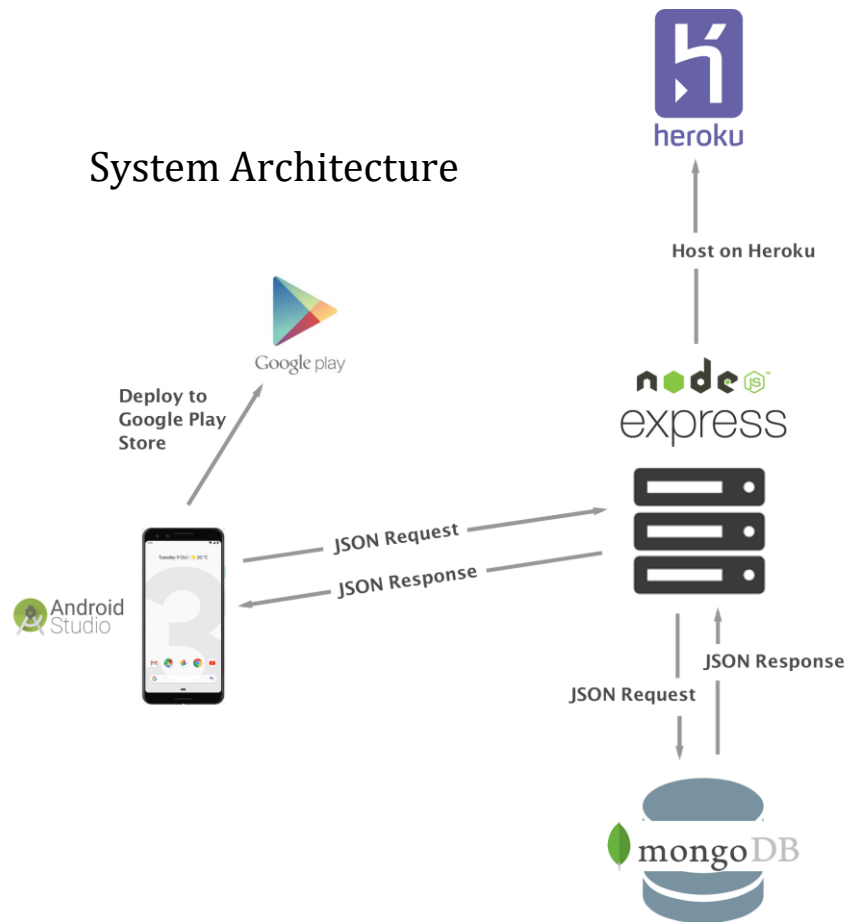
- Server: Node.JS, Express
- Database: MongoDB
- Frontend-end: Java (Android Studio)

Android mobile operating system is currently taking the lead in the market share worldwide. Big companies such as Samsung, Google, and Huawei have mobile devices operating on Android. Taking this into consideration, deploying an android application will allow Quick Help to reach a wider range of users.

We will be using Android Studio as our IDE to build our front-end. Android Studio encompasses many advantages over other technologies, most evidently its built-in emulator. The built-in emulator will make it easier for the team to run and test the application. We decided to use mLab for our MongoDB database. Heroku will be used to host our Node.js server which will contain our Express application that will handle API requests to our server. Finally, the final product will be deployed to Google play console so the application is available for download worldwide. To

accommodate users without mobile data service, we are considering to use Twilio API for sending help requests and receive text based direction in our next release.

System Architecture



Design Mockup

System Decomposition

There are three main components in the Quick Help system architecture: client, server, and database.

The classes associated with the server are Register, SendSOSDAO, RecieveSOSDAO, UpdateInfo, Login, and MessageDAO. This is because the client will send REST API calls to our Node.js server in which one of these classes will handle the request and send out the appropriate response. To be more precise, the SendSOSDAO and ReciveSOSDAO in Quick Help will use a Node.js middleware which will find, send and notify the professional of the SOS signal that a user has sent out. Similar to Register, UpdateInfo, Login, and MessageDAO, the Node.js middleware will search through the database and send the necessary information between the users and professionals.

The client component classes are AndroidApp, Map, and Navigator. These are classes that the client uses to interact with the Android application. The classes send JSON formatted requests to the server in hopes to receive back the required responses.

The database component consists of the class User. These classes represent schemas in our MongoDB database.

The system decomposition is provided on the system architecture diagram. It shows each component and its functionality on a higher-level architectural view.

We anticipate that there might be connectivity issues that will interfere with a user's ability to submit a registration or send SOS signals. In order to fix any issues with network connectivity, during a SOS signal, we intend to implement a quick restart feature which will be prompted when the app receives two consecutive 500 error from the server. This can ensure that a client-server connection is intact.

Another problem to consider is view scalability issues. Views may vary depending on the size on the client's android device. The strategy will be to create multiple views of the app and to make sure that the design we use is easily resizable.

Other failures such as incorrect user input from our client will be dealt with in the server where we will validate the data and send appropriate errors when necessary. This will be handled using a Node.js library called validator.js.



QuickHelp

Login

Register

Request For Help



Additional Details



Blocked airflow



Heart Disease



Broken Bone



Skin Burn



Bleeding



Unconsciousness

Specify injury details here

Update Request

Call 911





**We found a skilled
professional! Help is on
the way!**

Dismiss Request

Help Arrived



Procedures While Waiting



Bleeding

- (1) Keep the patient calm
- (2) Find a clean piece of cloth
- (3) Use the cloth to put pressure on the wound
- (4) Help the patient lie down
- (5) Remove debris from wound (do not remove any objects embedded in the body)



Broken Bone

- (1) Don't try to realign the bone or push a bone that's sticking out back in
- (2) Apply ice packs wrapped in a cloth to limit swelling and help relieve pain





**We are still searching
for a skilled
professional to assist
you**

Dismiss Request

Contact Emergency Services



Profile



Betty Smith

Member since 2018



Member

Personal Information

Offline/Online



Medical History

Emergency Contact

Update Member Status





Rate the Skill Professional



Personal Information



First Name

Veronica

Last Name

Sullivan

Address

56 Castle Drive

Occupation

Pediatrician

Blood Type

O

Medical History

- ☒ High Blood Pressure
- ☐ High Cholesterol
- ☒ Kidney Disease
- ☒ Thyroid Problems
- ☐ Joint Replacement
- ☐ Lung Disease
- ☐ Stroke
- ☐ Asthma
- ☐ Heart Problem
- ☐ HIV
- ☐ Kidney Problems
- ☐ Nervous Disorder


Other

No SIM card — Emergency calls only



4:18

Tue, Aug 7 ☀ 17°C

 Messenger • 2h ▾

Julian Chokkattu
Hi there



 QuickHealth • 1m

Car Accident on Bloor Street



Notifications

- Car Accident on Bloor St.
- Asthma Attack on St.George St





Emergency Detail

[View Patient Profile](#)

Injury Details

Head injury, unconscious

Accept Request

Dismiss Request



Profile



Veronica Sullivan 

Member since 2018



Doctor



Personal Information

Offline/Online

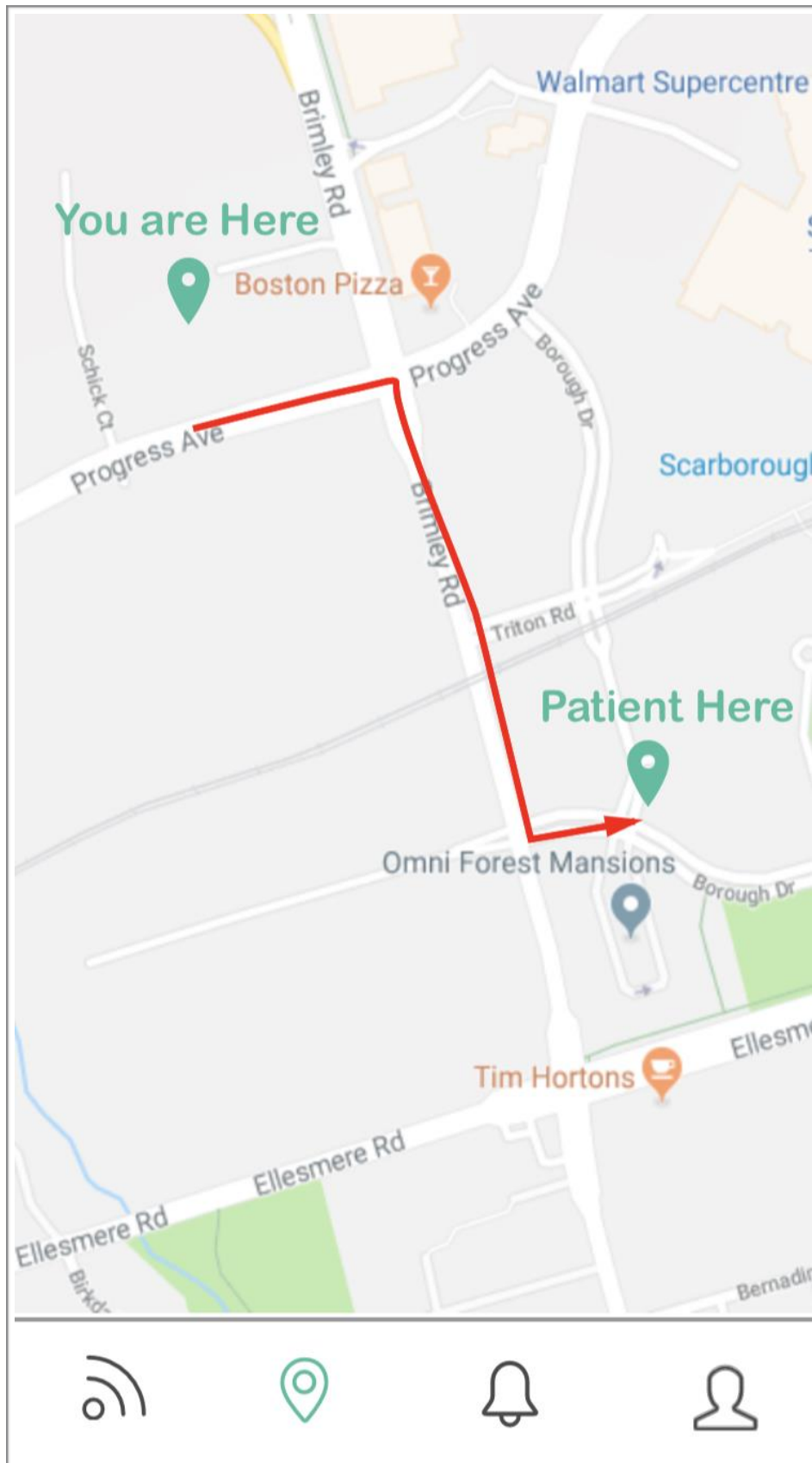


Medical History

Emergency Contact

Update Member Status





Update Status

To be able to access this application as a medical professional, it is required that some valid credentials are submitted to ensure all doctors operating this app

If you wish to become certified on this app, please use the upload button below to upload pictures of your certifications.

If you meet the requirements, you will receive an email containing the certification key.

Upload Credentials

Upload images of your medical credentials using the upload button below.

Upload

Documents 0

Send

Enter Certification Key:

Verify