EMS Mobile Application

Detailed Design

Gautham Arun
Anna Dingler
Allison Nakazawa
Willem Taylor
Ryan Tobin

April 18th, 2021

Table of Contents

List of Figures

Introduction

Background

In medical emergencies, it is imperative that Emergency Medical Technicians (EMTs) can quickly locate the closest available hospital with appropriate resources. The current web solution created by the Georgia Coordinating Center (also referred to as Georgia Regional Coordinating Center or Georgia RCC) details the availability and location of hospitals, but fails to translate well to mobile devices, making it more difficult for EMTs to use on the road. In addition, the website does not take advantage of GPS services to locate the nearest hospital, instead forcing EMTs to scroll through a table and waste valuable time on logistics rather than focusing on the patient. Having real-time information about the closest hospitals with the correct resources would bring EMTs one step closer to saving a patient's life. Our client, Harold Solomon from Georgia Tech Venture Lab, first recognized the need to improve the hospital locating process. He began reaching out to key players in the industry to further develop his idea for a widespread solution and enlisted our assistance to make it a reality. Our project aims to cut down the time it takes for EMTs to select an available hospital by providing them easy access up-to-date hospital information on their mobile devices.

While our application will be available to the general public on the app store, our target audience is EMTs on the job. They will use our application to locate the nearest hospital that can accommodate the patient in their care. In order to make the applications easy to use for EMTs, we will provide an intuitive interface that minimizes the clicks needed to find the closest hospital. Usage of this application requires only the ability to operate a mobile device. With all this in mind, we will not make special accommodations to the app for the general public. This excerpt from an EMT job description details how EMTs need to decide which hospital to bring their patient to while caring for them:

"The EMT-B uses the knowledge of the condition of the patient and the extent of injuries and the relative locations and staffing of emergency hospital facilities to determine the most appropriate facility to which the patient will be transported, unless otherwise directed by medical direction."

This application would reduce the potential time that it takes for an EMT to determine the most appropriate facility to transport their patient to by giving them easy access to much of the information that must be considered, including NEDOCS score (a measure of the severity of overcrowding in the hospital's emergency department), diversion status (whether the hospital is redirecting certain types of ambulance traffic to other hospitals), and more. The EMTs can expect to find information about the relative locations and occupancy of nearby emergency hospital facilities in the app. We must also consider that the EMTs using the app are likely to be focused on helping the patient, so finding the nearest hospital through our app should not have to be the focus of their attention.

Our solution addresses the problem by providing intuitive iOS and Android versions of a mobile application. In the applications, users will be able to view information such as NEDOCS score, diversion status, and time of last data update for all hospitals in Georgia. In order to make the information as accessible to users as possible, the application will provide capability to filter and sort the hospital list by a variety of different factors so that EMTs can prioritize hospitals with a better NEDOCS score, a particular specialty center, closer proximity, and more. While viewing the list of hospitals, a user can

click on a particular hospital to view detailed information about its specialty centers, its diversion status, and its address and phone number. All of this aims to provide EMTs with the most visibility possible into a hospital's status while facilitating communication and navigation to the hospitals to help EMTs make the best decision when selecting a hospital for a patient.

Document Summary

In this document, we will detail the design of our application. In section 2, we provide our system architecture, which explains how the components of our system communicate with each other. In section 3, we explain the design of our data storage using an ER diagram as well as how data is exchanged across the different components of our application. In section 4, we give a structural diagram that shows the low-level components of our application and details their interactions. Finally, in section 5, we provide images of the UI design of major screens in our application. Our application is defined by 4 main components: the database used to store hospital data, the web scraper to pull updated hospital data in real time, a mobile application to allow the EMTs to interact with the application, and a series of backend end-points to allow the mobile application to communicate with the database.

System Architecture

Introduction

To illustrate the major components of our system and the interactions between them we have provided designs for the static and dynamic architecture of our system in Figure 2.1 and Figure 2.2, respectively. Both iOS and Android versions of our application share a nearly identical architecture, and the diagrams have been generalized to represent both versions and have been divided into three layers:

User interface, which the user interacts with to initiate app functionality.

Application logic layer, which contains the models used to represent real-world constructs in our application.

Data layer, which retrieves and stores the data in a database and provides the data to the application logic layer.

The static and dynamic architecture of our system is described in more detail below.

Static Architecture

The static architectural design shown in Figure 2.1 provides a functional view of our application. We display how each of the application's components are related and how they interact with each other in the diagram below.

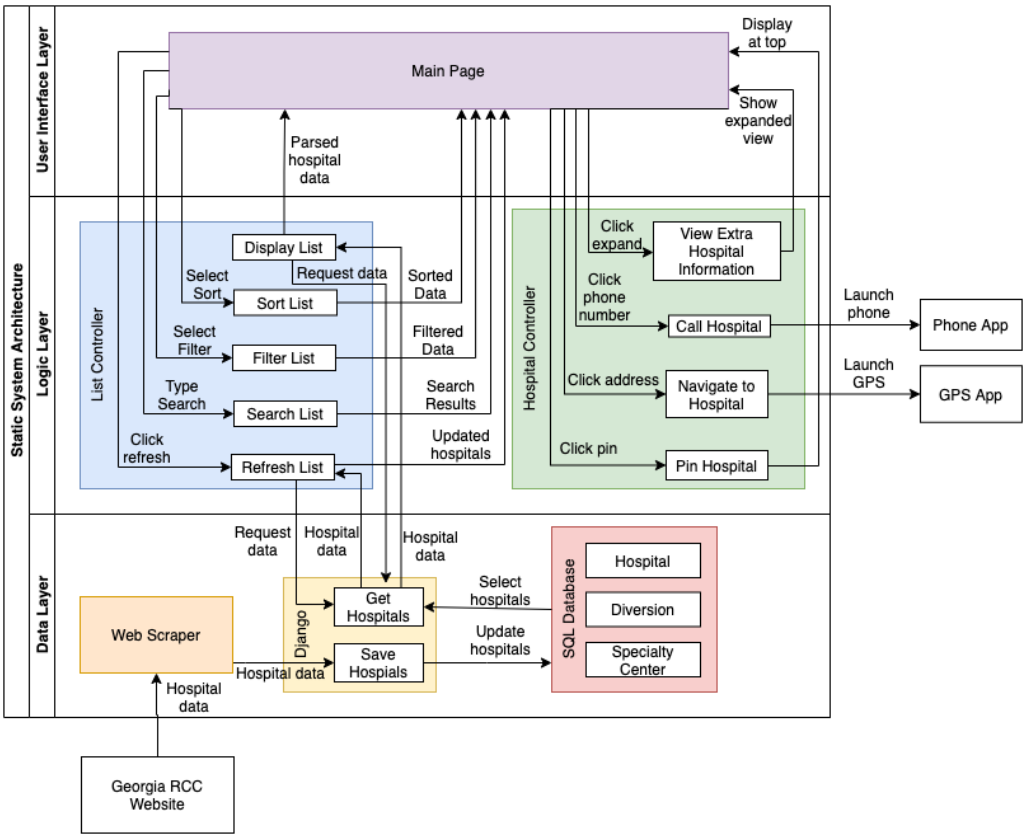

Figure 2.1 *Static System Architecture Design for EMS Mobile App*

Dividing the structure into three parts helps with the organization of the application and ensures that all procedures flow logically. The structure outlined in the diagram applies to both the Android and the iOS versions of the application. All three of the architecture layers are described below.

For our mobile application, we decided to follow a Model View Controller (MVC) architectural approach in line with what many use in industry. The user interface layer as seen in the diagram is the view component of an MVC architecture and consists of the actual user interface and its components. It is completely separated from the logic within the application. Most of our application's functionality gets initiated by interacting with the user interface. In Android, these are the xml and drawable files. In iOS, these are the storyboards and XIB files. Our user interface is one page, called the "Main Page" in the diagram, which consists of a list view that shows each hospital and an expand button to see that hospital's details. Our user interface also consists of a filter, sort, search, refresh, call, navigate, and pin hospital option which all call the appropriate methods from the appropriate controllers in the logic layer.

To control the application, we have the logic layer as seen in the diagram. This layer is broken into two main parts: the list controller and the hospital controller. We chose to have two separate controllers in the application to increase the readability of the backend. The list controller will handle all logic that adjusts the list of hospitals on the main page. These functionalities are displaying, filtering, sorting, searching, and refreshing the hospital list. When the application loads or the user refreshes the list, the list controller will make a call to the Django get hospitals endpoint, and then it will send the new hospital list to the user interface layer. For the filtering, sorting, and searching, the list controller will make the adjustments to the list of hospitals and send the updated hospital list to the user interface layer. The hospital controller is meant to perform all functionality for an individual hospital. When the user selects the call or navigate buttons for a specific hospital, the hospital adapter will launch the phone and navigation apps, respectively. The ability to pin a hospital and view additional hospital information will also be dictated by the hospital controller. When the user selects one of these options in the user interface layer, the hospital adapter will notify the correct interface elements to make adjustments.

The data layer consists of 3 main components: the web scraper, a Django web app, and an SQL database. The Django web app serves as the intermediary between the web scraper, database, and mobile application. Each of those components must go through the Django web app to communicate with each other. The Django web app provides a series of endpoints to allow for this communication. The web scraper pulls data from the Georgia RCC website and delivers it to the save_hospitals endpoint, which then saves the data in the database. Upon application open and refresh list, the mobile application calls the get_hospitals endpoint, which retrieves the hospital data from the database and sends it back to the mobile application in a json format to be parsed and displayed. This architecture allows for an easy and reliable flow of data. Using an SQL database, it would be very difficult and bad practice for the mobile application to directly touch the database. Having the mobile application directly receive information from the web scraper would also be unreliable. Both of these facts make an intermediary like a Django web app very necessary and important.

Dynamic Architecture

The dynamic architecture shows the flow of information in the system and between components within it. The figure below depicts a user's interactions with the mobile application. These include filtering, sorting, searching, pinning a hospital, calling, and mapping to a hospital.

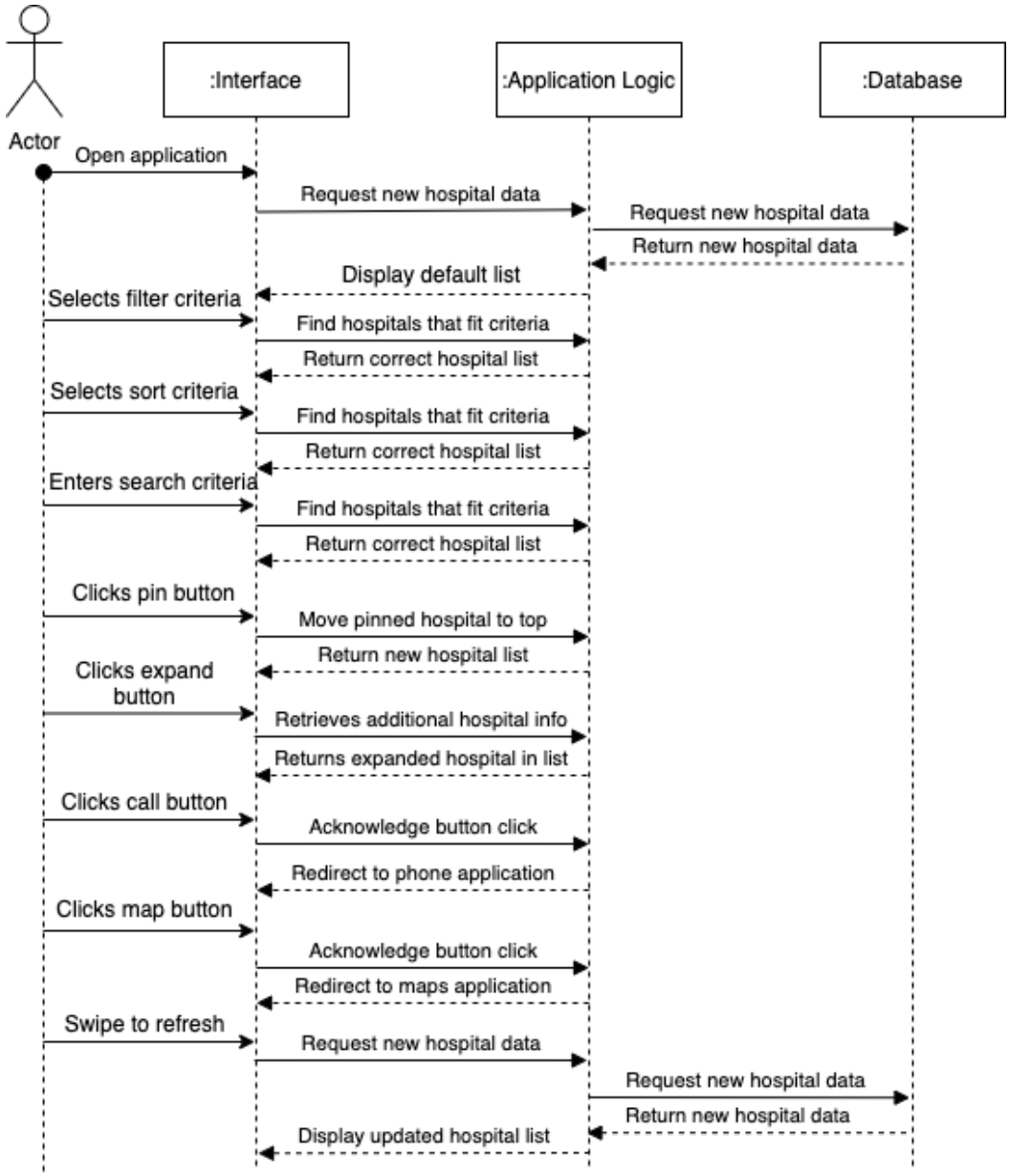

Figure 2.2 *System Sequence Diagram for EMS Mobile App*

When a user opens the application, a request is sent to the backend database to retrieve the most recent hospital data, which is then displayed on the main screen of the application. Additionally, a user can swipe to refresh the data anytime in which a request will be sent to the backend database to retrieve the most recent hospital data, which is then displayed on the main screen of the application. Once the data is displayed, users can use the interface to specify a particular filter, sort, or search

criteria to modify the data display. When these requests are selected in the interface, they are then sent to the application logic layer where the data is filtered and sorted accordingly. The resulting data is then sent back to the interface to be displayed on the main screen. When a user pins a particular hospital in the interface, the request is sent to the application layer, where the pinned hospital is moved to the top of the list and displayed in the interface. When the user selects to expand the view of a particular hospital, no request to the backend or application layer is necessary because the information is already in the interface. When a user clicks the call button in the interface, the request is sent to the application layer, which then causes the phone application to be opened in the interface. When the user clicks the map button, this request is also sent to the application layer, which then causes the mapping application to be opened in the interface.

Data Storage Design

Introduction

This section details our application's data storage design. Our application consists of 3 main data storage points: a relational database, data exchange from our web scraper to our database, and data exchange from our database to our mobile application. Our relational database is used to store hospital information that will be pulled and displayed on the mobile application. The data exchange from the web scraper is used to scrape data from the Georgia RCC website and save it in the relational database. The data exchange to the mobile application is used to pass data stored in the database to be displayed and interacted with on the mobile application.

Relational Database Design

Figure 3.1 diagrams our relational database design as an Entity Relationship Diagram (ERD). The relational database is used to store hospital information that will be pulled and displayed on the mobile application. Our relational database consists of 3 entities: hospital, specialty center, and diversion. Hospital represents an actual hospital. A hospital has a name (primary key), phone number, EMS region, Regional Coordinating Hospital (RCH), address which is a compound attribute consisting of street, city, state, zip, county, latitude, and longitude, an attribute denoting the last time in which the hospital's information has been updated, and the hospital's current NEDOCS score. The diversion table represents medical diversions that can occur at a hospital. It only has a type attribute (primary key) denoting the type of medical diversion. The specialty center table represents the type of hospital and only has a type attribute (primary key). A hospital has a many-to-many relationship with the specialty center entity as a hospital can have many specialty centers and a specialty center type can belong to many hospitals. A hospital also has a many-to-many relationship with the diversion table as a hospital can have many diversions and a diversion type can belong to many hospitals. Each of these relationships has partial participation on each side, as a hospital does not have to have a specialty center or a diversion, and vice versa.

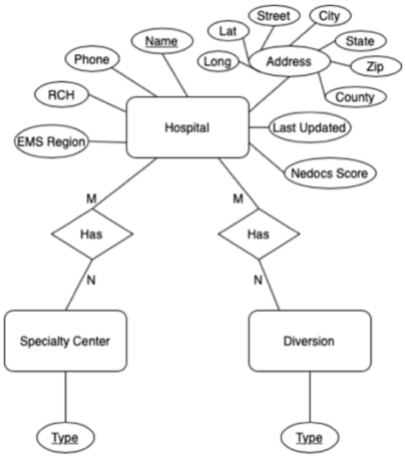

Figure 3.1 *Entity Relationship Diagram for EMS Mobile App Database*

Data Exchange from Web Scraper

Upon completion of the web scraper's execution, the data retrieved from the GCC website is stored in a JSON file inside the backend directory of the application. Each hospital is stored as its own line in the JSON file. Below is an example of a JSON object for a specific hospital.


```
{
    "name": "Emory - Decatur Hospital",
    "street": "2701 N Decatur Rd",
    "city": "Decatur",
    "county": "DeKalb",
    "state": "GA",
    "zip": "30033",
    "lat": "33.7917584000",
    "long": "-84.2981244000",
    "phone": "4045011000",
    "rch": "D",
    "ems_region": "3",
    "specialty_centers": [],
    "last_updated": "2021-04-15T22:59:57",
    "nedocs_score": "Severe",
    "diversions": ["ER Saturation", "Medical Saturation"]
}
```


When the Django server is running, the web scraper runs every 60 seconds. The reason it executes every 60 seconds is because the Georgia RCC website, the website from which we scrape our data, refreshes its data every 60 seconds. When the web scraper runs, it retrieves the most recent data from the Georgia RCC website. The resulting data is then dumped into the JSON file as described above, which is located in the same directory as the Django server code. The server then reads the JSON file one line at a time and creates a Hospital object populated with all the data found in the fields shown above for that particular hospital. All of the current entries in the Hospital table in the database are then replaced by the Hospital objects created from parsing the JSON. This ensures the database always has the most up to date data to be displayed on our mobile applications.

Data Exchange to Mobile Application

The data from our database is sent to the mobile application as a JSON string whenever a get_hospitals request is made. To execute a get_hospitals request, the mobile application makes a url request to https://our-domain-name/hospitals. Each entry in the Hospitals table is then written into a JSON string containing all the same fields outlined in the section above. Once all the hospitals have been added to this JSON, the string is returned back to the mobile application as the result of the get_hospitals request. The data is then displayed on the main screen of the application. Though the format of the JSON string returned to the app is the same as the JSON string created by the web scraper, the database exists to ensure that the users always have access to data that is as up to date as possible. It is possible for the web scraper's execution to fail if the GCC website is down or the user has poor internet connection, but the existence of the backend database ensures that the user always has access to the most recent hospital data possible.

Component Detailed Design

Introduction

The different components of our application that we consider for the following diagrams are the major classes and core functionalities. Main Page, Hospital List Adapter, and Hospital View are the focal elements of the diagrams. These components refer to the main screen, the container for a list of hospitals, and the view for an individual hospital respectfully. The other components are all intermediate actions that facilitate the interactions between these three parts. Note, that the components correlate to both the Android and iOS versions of our application, however some classes might have different names. Below you will see the static component design which follows the baseline interaction of the components and the dynamic design which follows specific runtime interactions of the components.

Static Component Design

The diagram below highlights important pieces of the application and displays their interactions without considering specific runtime scenarios. For this diagram we follow these conventions: a line signifies a connection between two components, an arrow signifies a directional connection between two components, a circle signifies the provided interface, and the semicircle signifies the required interface. Additionally, relationships are labeled with their specific action.

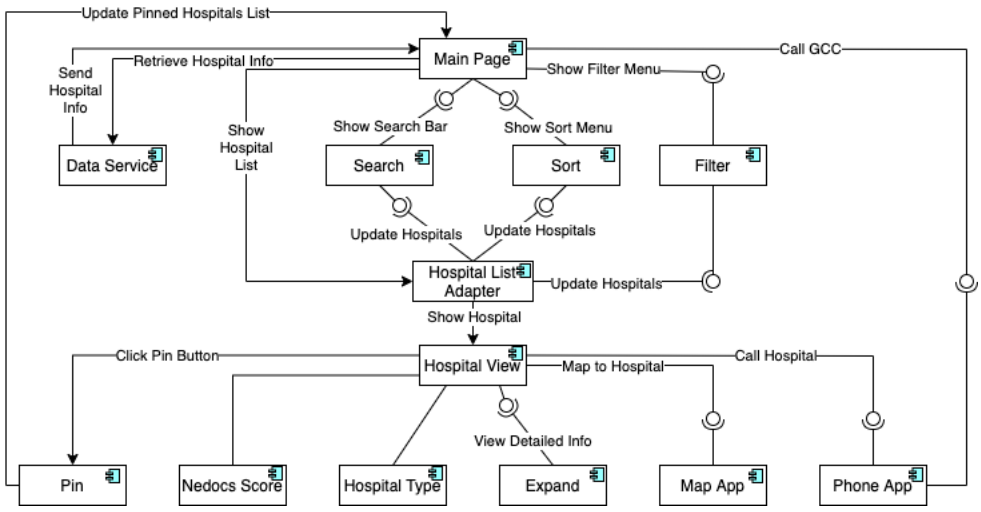

Figure 4.1 *Static Component Diagram for EMS Mobile App*

The Main Page includes a list of hospitals and a top menu bar which allows the user to search, sort, or filter the hospital list as well as call the Georgia Coordinating Center (GCC). The Main Page communicates with the database through a Data Service to retrieve information about each hospital continuously every sixty seconds and the Data Service returns the most recently updated information about each hospital. The user can select the Search button to display a search bar across the top that allows the user to search for a specific hospital by name. The user can also select the Sort or Filter button to display a menu that allows them to select the attributes they wish to sort or filter by.

Additionally, the user can select the Call GCC button to be redirected to their device's Phone Application to complete the call.

The Hospital List Adapter receives information about the hospitals from the Main Page and displays them in a list. The list may be sorted and/or filtered based on the user's selections and its display will be updated accordingly. The Hospital List Adapter manages the information about each hospital in individual Hospital Views.

Each Hospital View contains information about the hospital's Pin status, NEDOCS Score, Hospital Type(s), and, when expanded, location and contact information. The user can click on the Pin button to either favorite or un-favorite a hospital, depending on its previous status. If a hospital is pinned, the Main Page will handle adding it to a Pinned Hospitals List which will always be displayed at the top of the page. The NEDOCS Score and Hospital Type values are validated and displayed in the Hospital View along with corresponding icons and descriptive text. The user can click on the Expand button to either expand or collapse more detailed information about a specific hospital. If a Hospital View is expanded, the user can select the hospital's address to be redirected to their device's Mapping Application to navigate to the hospital or the hospital's phone number to be redirected to their device's Phone Application to complete the call.

Dynamic Component Design

This diagram (Figure 4.2) includes details about a user's interaction with the system, and more specifically considers control flow. It is similar to the static diagram (Figure 4.1) but shows the actions a user could take in order to filter, search, or sort the hospital list, call a hospital or the GCC, expand a hospital in the list, map to a hospital, and pin a hospital to the top of the list.

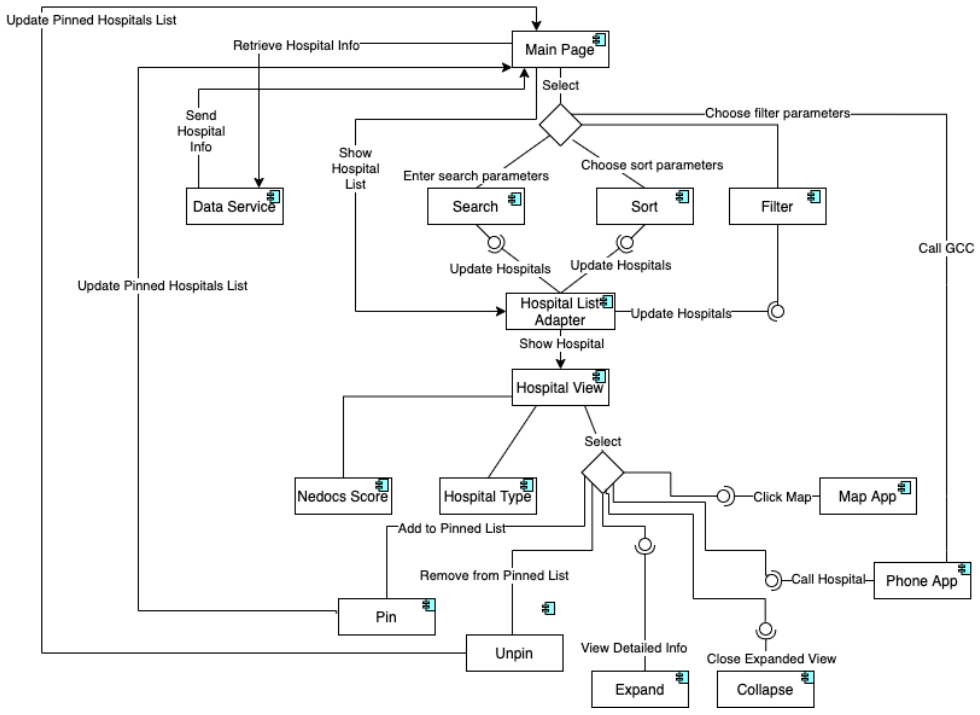

Figure 4.2 *Dynamic Component Diagram for EMS Mobile App*

A user begins their usage of the application with the Main Page. From there, they are able to take many different paths. One possible scenario would be pressing the sort button to open a sort menu that allows the user to sort the hospital list by different attributes, selecting one of those attributes, and then viewing the newly sorted hospital list by the chosen attribute. Another possible path is to press the filter button to open a filter menu that allows the user to filter the hospital list by different values for different attributes, selecting one or more of those values, and then viewing the newly filtered hospital list by the chosen values. The user could also press the search button to reveal a search bar, type a search term into the search bar, and view a list of hospitals that contain the search term in the name. Other paths the user could take include pinning or unpinning a hospital from the top of the list and expanding or collapsing a singular hospital. After expanding a hospital, the user may choose to contact the GCC or the hospital directly, which redirects the user to their phone application. Finally, the user may choose to map to that hospital, which redirects them to their mapping application of choice.

UI Design

Introduction

In this section we present the User Interface (UI) of our mobile application, which EMTs will use when they are deciding where to transport a patient. We will provide screenshots of all major pages within our app as well as the rationale behind our UI decisions for each screen.

Major Screens

Upon opening the application, the user is presented with a list of all the hospitals in Georgia. Each row in the table presents the hospital's name, how far the hospital is away from the EMTs current location, if the hospital is currently under medical diversion, the hospital's specialty centers if it has any, the hospital's NEDOCS score, and the ability to pin a hospital to the top of the list. Each of these components were selected to initially be displayed because they provide the most important and necessary information that an EMT needs when considering a particular hospital to take a patient to. The UI was chosen to be displayed in this manner to fall in line with the conventions of the Georgia RCC website (www.georgiarcc.org), the current solution that EMTs use.

If the user would like to see more information about a specific hospital, they can click on the expand arrow button in the bottom right-hand corner of that hospital in the list. Upon clicking this button, the section for the selected hospital will expand to display the hospital's phone number, address, diversions, specialty centers, county, region, regional coordinating hospital, and time of last data update. These pieces of information are initially hidden because they were not deemed as vital until the EMT is more closely considering taking their patient to that specific hospital. Additionally, this expanded view provides a more verbose explanation of the hospital's current diversion and specialty center icons displayed in the unexpanded view. From this view, the user can click the hospital's phone number to initiate a call to the hospital or click the hospital's address to launch their phone's mapping service with directions to the hospital. To collapse the information about a hospital the user can click the collapse button in the bottom right-hand corner or simply expand a different hospital's information.

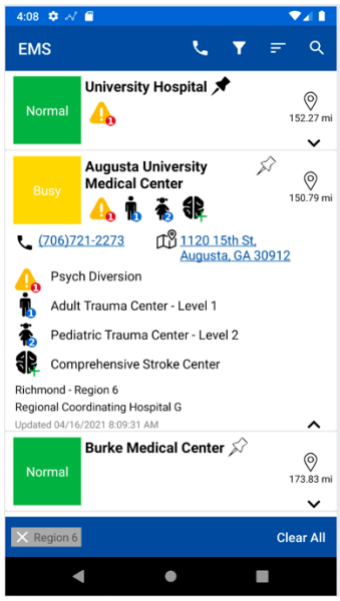

Figure 5.1 *Main Page of Application*

If the user would like to search for a specific hospital by name, they click on the search icon located in the top right-hand corner of the screen. Upon clicking this button, the search bar edit text will automatically be selected and the keyboard will appear so that the user has a smooth transition into the search page. As the user types in the search bar, the hospital list will be updated below the search bar so that the user has immediate visibility of the search results. Figure 5.2 shows what the search page would look like while the user is typing "wellstar" into the search bar. When the user selects the clear button located on the right-hand side of the search bar, the current search parameter will be removed, and all of the hospitals will be displayed. To close the search bar, the user clicks on the search icon again or clicks the clear button while the search bar has no text.

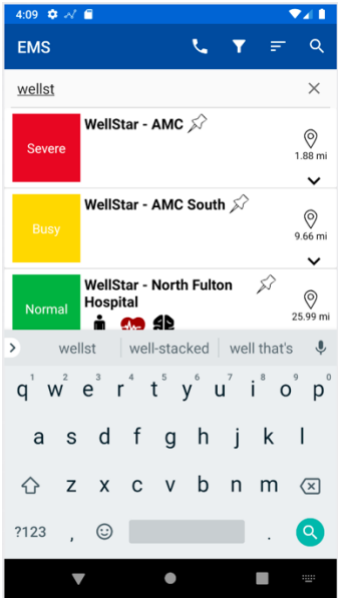

Figure 5.2 *Search Page of Application*

If the user wishes to filter the list of hospitals displayed on the main screen of the app, they can click on the filter icon located in the top bar to filter the results. Upon clicking this button, the filter selection menu is opened displaying the four hospital attributes that can be filtered on: county, regional coordinating hospital, EMS region, and specialty centers. Upon clicking one of these categories, the category's box is expanded to show the user a scrollable list of all potential values of that attribute (for example, if the user clicks the "county" category they will be presented with a list of all the possible counties they could filter on). Users can apply a particular filter by clicking the checkbox next to it, and they can apply as many filters as they wish at once. Users can remove all applied filters by clicking the "Clear All" button at the top of the filter menu. Once they have completed their filter selection, users click the "Done" button at the bottom of the filter menu to return to the main screen of the app with the selected filters applied.

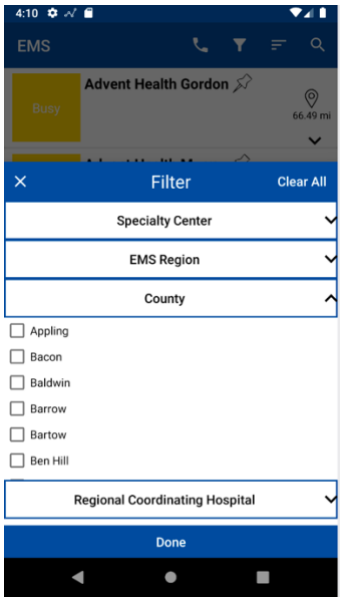

Figure 5.3 *Filter Menu*

If a user wants to sort the list of hospitals displayed on the screen, they are able to click the sort icon located next to the search icon in the top right of the screen. This displays a menu where the user can choose between 3 different sort options, distance, alphabetical, or NEDOCS score. After selecting one of these options, the hospital list is updated with the new sort being shown. The hospital list is automatically sorted alphabetically when the application is opened by a user as shown below, but they can change the ordering at any time using this functionality.

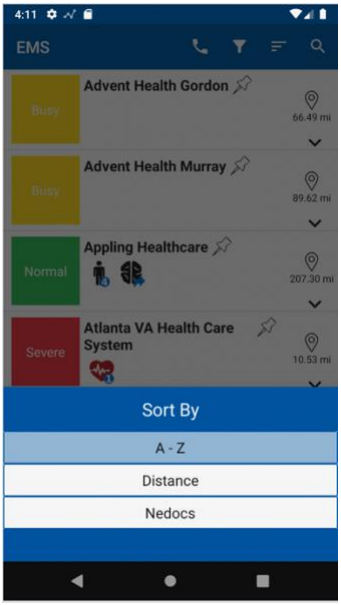

Figure 5.4 *Sort Menu*