A SMARTPHONE APP FOR SUBSTANCE USE DISORDERS

by

Haibi Hu

A thesis submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Master of Science in Bioinformatics and Computational Biology

Summer 2019

© 2019 Haibi Hu
All Rights Reserved
A SMARTPHONE APP FOR SUBSTANCE USE DISORDERS

by

Haibi Hu

Approved: 
Hui Fang, Ph.D.
Professor in charge of thesis on behalf of the Advisory Committee

Approved: 
Cathy Wu, Ph.D.
Director of the Center for Bioinformatics and Computational Biology

Approved: 
Levi T. Thompson, Ph.D.
Dean of the College of Engineering

Approved: 
Douglas J. Doren, Ph.D.
Interim Vice Provost for Graduate and Professional Education and Dean of the Graduate College
ACKNOWLEDGMENTS

I would like to express the appreciation to my advisor Dr. Hui Fang for her continuous advice, patience and academic support during the past year. Her guidance helped me finish this project and inspired my interest in app development.

I wish to thank my committee members—Dr. Tammy L. Anderson improved my knowledge on substance use disorder and Gabe Humphreys helped with the technical issue for app development.

In addition, my sincere thanks goes to our HeNN team, Dr. Cathy Wu, Julie Cowart, Sachin Gavali, Logan Neitzke-Spruill, Joshua H. Stout, for their efforts in constructing backend database and collecting SUD-related resources.

Finally, I wish to express my gratitude to my parents and friends for their unconditional love and support.
# TABLE OF CONTENTS

LIST OF TABLES ......................................................... vi  
LIST OF FIGURES ..................................................... vii  
ABSTRACT ................................................................. ix  

Chapter  

1 INTRODUCTION ......................................................... 1  
1.1 Motivation ......................................................... 1  
1.2 Challenges ......................................................... 3  
1.3 Project Impact ..................................................... 6  
1.4 Overview ......................................................... 7  

2 RELATED WORK ......................................................... 9  

3 FUNCTIONALITIES AND DESIGN ............................... 12  
3.1 HeNN Functionalities ............................................. 12  
3.2 HeNN Design ..................................................... 13  
3.2.1 Architecture ................................................... 13  
3.2.2 Development Tool ............................................. 14  
3.2.3 Plugin Used .................................................... 15  
3.2.4 Data Structure ............................................... 15  

4 IMPLEMENTATION ..................................................... 17  
4.1 HeNN User Interface ............................................. 17  
4.1.1 Home Screen .................................................. 19  
4.1.2 Service List Screen ........................................... 20  
4.1.3 Service Screen ............................................... 25  
4.1.4 Search Screen ................................................ 29  

iv
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Mobile Applications for SUDs</td>
<td>10</td>
</tr>
<tr>
<td>4.1</td>
<td>Correspondence from File to Screen</td>
<td>17</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Substance Use Disorder in 2016 and 2017 among Individuals Aged 12 or Older, by State: Percentages, Annual Averages Based on 2016 and 2017 NSDUHs[5]</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>Reasons for Not Receiving Substance Use Treatment in the Past Year among People Aged 12 or Older Who Felt They Needed Treatment in 2017[17]</td>
<td>3</td>
</tr>
<tr>
<td>1.3</td>
<td>The Screenshot for the First Page of Google Search Results</td>
<td>4</td>
</tr>
<tr>
<td>1.4</td>
<td>The Screenshots for App Qytt</td>
<td>6</td>
</tr>
<tr>
<td>3.1</td>
<td>Relationship between Service Provider, User and HeNN App</td>
<td>13</td>
</tr>
<tr>
<td>3.2</td>
<td>Sitemap for HeNN App</td>
<td>14</td>
</tr>
<tr>
<td>4.1</td>
<td>Screenshot for the GIS-based Notification</td>
<td>18</td>
</tr>
<tr>
<td>4.2</td>
<td>Screenshot for the Home Screen of HeNN</td>
<td>19</td>
</tr>
<tr>
<td>4.3</td>
<td>Screenshot for the Service List Screen of HeNN</td>
<td>21</td>
</tr>
<tr>
<td>4.4</td>
<td>The Screenshot of Services Filter</td>
<td>23</td>
</tr>
<tr>
<td>4.5</td>
<td>Screenshot for the Service Screen of HeNN</td>
<td>25</td>
</tr>
<tr>
<td>4.6</td>
<td>Screenshots for the Service Screen of HeNN (Directions Tab)</td>
<td>27</td>
</tr>
<tr>
<td>4.7</td>
<td>Screenshot for the Full Screen Map</td>
<td>27</td>
</tr>
<tr>
<td>4.8</td>
<td>Screenshots for the Service Screen of HeNN (Comments Tab)</td>
<td>28</td>
</tr>
<tr>
<td>4.9</td>
<td>Screenshot for the New Comment Screen</td>
<td>28</td>
</tr>
<tr>
<td>4.10</td>
<td>Screenshot for the Search Screen</td>
<td>29</td>
</tr>
<tr>
<td>Section Title</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>4.11 Screenshots for the Event Screen of HeNN</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>4.12 Screenshots for the Event Filter Screen</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>4.13 Screenshots for the Settings Screen</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>4.14 Screenshot for the Location Service Section</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>4.15 Screenshots for the Login and Signup Screen</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>4.16 Screenshot for the Help Screen</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>5.1 Term Counting for All Positive Reviews</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>5.2 Term Counting for All Negative Reviews</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>5.3 Term Counting for Positive Reviews from Christiana Care</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>5.4 Term Counting for Negative Reviews from Christiana Care</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>5.5 The Report of Events Analytics from Firebase Console</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>5.6 The Number of Click for Buttons in Home Screen</td>
<td>48</td>
<td></td>
</tr>
</tbody>
</table>
ABSTRACT

As the number of people suffering from Substance Use Disorder (SUD) increases yearly, the government and related organizations established various resources for the prevention and treatment of SUD. Nevertheless, they have encountered the challenge that most people who needed medical help did not receive it due to the difficulties in finding appropriate resources. With the convenience brought by smartphones and handheld devices, the mobile application can construct a platform which connects patients to the SUD services.

To assist in solving this problem, we developed a GIS-based smartphone app, HeNN, which provides an interactive platform for SUD patients, their families and friends, resource providers in Delaware as well as surrounding areas. HeNN’s user interface was built based on Flutter, and its backend services leveraged Flutter plugins for Google Map Platform, Google Calendar API, Firebase API.

The project is expected to make impacts. For SUD patients and their families, HeNN categorizes local services as well as upcoming events and provides all detailed information with distances and directions according to user’s preference. For providers and policy-makers, HeNN establishes a peer network by allowing users to leave comments on resources. Through the potential analysis for reviews and the usage of app components, we present the feedback for the utilization and effectiveness of services.
Chapter 1
INTRODUCTION

Substance use disorder (SUD), known as drug use disorder, is a medical condition under the recurrent use of one or more substances (e.g., alcohol, illicit drugs or pain relievers)\cite{1}. Nowadays, SUD, as a mental health problem, affects individuals from various careers and all age groups\cite{2}. Even though government or SUD-related organizations provided large amount of resources, people suffering from SUD still need to spend much time to find the treatment they want. With the convenience brought by smartphones, the mobile application can establish a platform which connects patients to the service.

1.1 Motivation

The situation for SUDs in Delaware is more serious than most states. A report\cite{3} from National Center for Health Statistics shows, in 2017, the drug overdose death rate in Delaware’s population (37.0 per 100,000) is much higher than the national rate (21.7 per 100,000), which is the sixth highest among all states.

The severe situation is not only reflected in overdose death rate, but also in the prevalence of SUD. A national survey\cite{4} on drug use and health estimated that 69,000 people aged 12 or older are suffering from SUD based on the annual average in 2016 and 2017, which takes 8.53 percent of the state population. Comparing to other states (see Figure 1.1), Delaware, colored in pink, had the tenth highest percentage in state population.

Since a large number of people in Delaware are suffering from SUDs, the local government and SUD-related organizations established facilities in order to help people get rid of substances addiction\cite{6}. The facilities established can be divided into four
categories, prevention, treatment, recovery, and other health resources, such as drug drop box or syringe exchange.

Prevention is the first step that stop the numbers of SUD from growing. The prevention process focuses on educating people, especially teenagers, to develop their knowledge on substances and keep them away from the overdose of alcohol or illicit drugs.

Treatments are the methods, which help people quit using alcohol or drugs. Each treatment program from facilities offers comprehensive services. For example, the Medication-Assisted Treatment (MAT) uses medications combined with counseling and therapy. There are more than 30 licensed treatment facilities in Delaware listed in the report for National Survey of Substance Abuse Treatment Services.

Recovery is a long process after the treatment for SUD which may take years. The resources for recovery mainly provide a drug-free place for people to live, like Oxford House. With similar addiction experience, people live together in the community.
surrounded by understanding, support and hope, which benefits their recovery[15, 16].

1.2 Challenges

Even though plenty of SUD services are intended to address addiction problems, a survey[4] indicates that an average of 62,000 people in Delaware needed but didn’t receive the expected treatment based on the statistics in 2016 and 2017.

To explore why people didn’t seek specialty treatment, a national survey on drug use and health[17] reported six main reasons(see Figure 1.2). Top four factors are stress coming from patients themselves, their community and career, which make them not willing to accept treatment. Two reasons left indicate that there is a group of people who want specialty treatment but can’t find the appropriate one. It’s hard to change people’s mind if they refuse treatment, but we can help those 20 percent people who are seeking for treatment.

![Figure 1.2: Reasons for Not Receiving Substance Use Treatment in the Past Year among People Aged 12 or Older Who Felt They Needed Treatment in 2017[17]](image)

Two existing solutions, which help people look for required resources, have their limitations. Providers in Delaware have published detailed information for the treatment programs and therapies on their websites, which allow the web search engine to find them. However, not all results from the search engine meet people’s expectation
and it’s time consuming to filter out the non-relevant links manually. Here we use an example to explain the limitations (see Figure 1.3).

Figure 1.3: The Screenshot for the First Page of Google Search Results

Suppose we are looking for the SUD treatment facilities in Delaware and take
"drug addiction treatment in Delaware" as the query for Google, a total of 34 documents are listed in top two result pages. We look through all 34 links to judge whether each result is related or not.

Looking into 17 results in the first page, we find that most results are relevant; while six of them are non-relevant. The ninth result, Delaware Substance Abuse Treatment Centers, is a helpful web page, which lists SUD treatment programs in Delaware and allows people to filter by county. Most links here are official websites of treatment facilities in Delaware, like the Addiction Campus or Gateway in Figure 1.3, which exactly match the search query. The fourth advertisement at the top, Ranch Caring Drug Rehab in Philadelphia is acceptable as the resource in surrounding area. The non-relevant results in this screenshot are the first three advertisements, because they are either treatment centers far away from Delaware or facilities for recovery.

Among 37 results, only 20 of them correspond to the query. The 17 documents left are either news, events, papers for SUD or facilities located far away from the Delaware, which are considered as irrelevant.

Thus, the process of seeking appropriate services by web search engine is time-consuming. It took hours to simply check if the results are treatment resources and their locations. People are concerned about more factors when selecting treatment facilities, such as the qualification, distance, insurance type, age, gender etc. Looking into each website or consulting the staffs for more detailed information will take much more time.

In addition to using a search engine, another option for finding treatment is the existing SUD-related mobile application[18, 20]. The smartphone apps recommend facilities based on people’s GPS location. It’s more straightforward to display surrounding services with maps or distances. However, most apps have the same limitations and we’ll take the app named Qytt as the example.

First, when the GPS or network is not available, no locations can be displayed on the map. Figure 1.4b is the screenshot for Qytt when GPS is disabled. This means if people want to use Qytt, both GPS and network must be available.
(a) Location permission enabled

(b) Location permission disabled

Figure 1.4: The Screenshots for App Qytt

Besides, Qytt only provides name, address, phone number and the hotlink to the direction page (see Figure 1.4a), which are insufficient for people to judge whether the place is helpful or not. As we mentioned before, treatment is one of the four categorizes for SUD resources. For patients who want to find services for prevention or recovery, Qytt is not going to help.

No feedback is also a limitation, because comments are critical resources for patients and providers[24]. Patients can share their experience for the treatment programs or consider others’ evaluations when selecting facilities; while for providers, comments reflect the pros and cons of their services. For now, there is no mobile apps developed for addressing SUD problems in Delaware and surrounding areas.

1.3 Project Impact

To overcome the previous challenges and combat SUDs in Delaware, this project is expected to make the following impact:
Reduce the risk of SUDs. The first step to prevent SUD is early intervention for people who haven’t yet become addicted or in the early stage. By increasing people’s knowledge of SUD and educating them to understand the harm brought by SUD, they might keep away from alcohol and illicit drugs.

Increase the awareness and utilization of SUD services. By categorizing resources and listing GIS-base treatment services, we make recommendations to the 10 percent people who are seeking treatment but do not know about SUD facilities in Delaware or surrounding areas. In other words, one of the aims of this project is to build a peer network for those patients and their families, helping them find the resources they need in time.

Provide feedback to providers. As we discussed previously, there was 9 percent of people who didn’t find the treatment program they wanted, which indicates that providers should consider patients’ requirements when they build the programs or make some adjustments. In order to grasp people’s demands, their comments for the program are essential feedback for providers. In this project, we offer information as well as gather feedback like reviews for potential analysis to assist service providers.

1.4 Overview

In this project, we developed a smartphone app, HeNN (Help Near & Now), which supports both Android and iOS, and provides one-stop high-quality services to people in Delaware or surrounding areas.

For SUD patients and their families, experts on SUD research collected and categorized the local resources. For each service, HeNN not only provides general information like phone number, address, website but also offers the number of available beds as well as real-time distance based on the user’s GPS location. Besides, HeNN shares upcoming events and 12-step meetings, which is also GIS-based by showing the distances. To recommend resources and events that meet people’s requirement, HeNN allows users to customize their profile and preferences.
For providers and policymakers, HeNN establishes a peer network by allowing users to provide comments on each resource. Based on the analysis for reviews and the usage of app components, we help providers to understand people’s requirement and preference and offer feedback on the effectiveness of services.

This thesis consists of six chapters to demonstrate the project in detail.

Chapter 2 discusses current mobile applications available in app store. By analyzing their functions and user interface, the pros and cons of apps become the reference for this project. Chapter 3 summarizes features and design of HeNN, developed in this project, to display the project’s framework. A brief introduction for the development tools — Flutter, APIs — and the data structure of this app is also mentioned in this section. Chapter 4 addresses the implementation of HeNN’s user interface and back-end services, including the constructing process of each page and the communication between the backend services with the API. This section is a detailed instruction for HeNN’s development. Chapter 5 outlines the potential analysis behind this project, involving the analysis for users’ comments and the analysis for the usage of app’s components, which indicates the possible feedback from patients and their families to service providers. Chapter 6 concludes the work we’ve done and demonstrates the prospect of the project by listing some future work, like new functionalities.
Chapter 2
RELATED WORK

This section presents a brief review for current SUD applications in Google Play and App Store, followed by their analysis.

More and more mobile apps are developed to help people suffering from SUD\cite{22, 23}. To compare and find the common features for the existing mobile apps, we presented a total of 11 mobile apps\cite{21} here from Google Play or App Store and classified them by their primary functionality.

For the area, among 11 applications, five of them are regional apps, and four are national apps. The national or international app intends to serve more people and receive more feedback; meanwhile, it takes much effort to collect and manage data, especially for real-time meetings.

For the operating system, almost all the apps support both Android and iOS, because they both take about half of the market share for operating systems in the US\cite{25}. Therefore a better option provided by Flutter is to develop the app in both operating systems at the same time.

For the user interface, like all medical applications, they are concise and easy to use, which means users should find what they want immediately. Maps with brief information or lists of meetings with time and distance highlighted are the strategies to make the app as simple as possible.

There are mainly two types of information provided by 11 mobile apps, meetings and services. Except for offering general contact information like phone numbers, addresses, websites and emails in text format for treatment facilities and meetings, most of the apps are GIS-based, which means they rely on user’s GPS location to make recommendations of the nearby services or meetings.
<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Area</th>
<th>Platform &amp; Rate</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meetings</td>
<td>Prevent Addiction Relapse</td>
<td>Regional</td>
<td>Android(3.7/3) iOS(1.0/1)</td>
<td>Meetings, Videos and Call Help for John Giordanos’s Recovery Center</td>
</tr>
<tr>
<td></td>
<td>Meeting Guide</td>
<td>International</td>
<td>Android(4.5/270) iOS(4.3/163)</td>
<td>GIS-based AA Meetings, in US and Canada</td>
</tr>
<tr>
<td></td>
<td>Pink Could: AA Meeting Finder</td>
<td>International</td>
<td>iOS(4.6/315)</td>
<td>GIS-based AA, NA, CMA Meetings, Worldwide</td>
</tr>
<tr>
<td>Services</td>
<td>Finding Help for Substance Use Disorders</td>
<td>Regional</td>
<td>Android(1.0/1) iOS</td>
<td>Phone Numbers and Addresses for Prevention, Intervention, Treatment, Recovery Services</td>
</tr>
<tr>
<td></td>
<td>You Are Linked to Resources Lite</td>
<td>National</td>
<td>iOS(5.0/1)</td>
<td>Phone Numbers, Addresses, News, Professional Magazines for SUDs</td>
</tr>
<tr>
<td></td>
<td>CRARC</td>
<td>Regional</td>
<td>Android/iOS</td>
<td>Phone Numbers, Addresses for Prevention, Treatment and Recovery Support Services</td>
</tr>
<tr>
<td></td>
<td>Routes to Recovery</td>
<td>Regional</td>
<td>Android</td>
<td>GIS-based Recommendation, Share Comments</td>
</tr>
<tr>
<td></td>
<td>Stay Alive L.I.</td>
<td>Regional</td>
<td>Android(3.0/1) iOS</td>
<td>GIS-based Services Recommendation</td>
</tr>
<tr>
<td></td>
<td>Fall Catcher</td>
<td>National</td>
<td>Android iOS(1.0/1)</td>
<td>GIS-based Services Recommendation</td>
</tr>
<tr>
<td></td>
<td>Thought Spot</td>
<td>Canada</td>
<td>Android(4.0/1) iOS</td>
<td>GIS-based Services Recommendation</td>
</tr>
<tr>
<td></td>
<td>Qytt</td>
<td>National</td>
<td>iOS</td>
<td>GIS-based Recommendation, Private Journal, Daily Notification</td>
</tr>
</tbody>
</table>

* Platform (Rating/Number of Reviews).

Data Source: Android - Google Play, iOS - App Store
However, these mobile apps have some limitations.

First, for location-based apps, people who don’t want to enable GPS are not going to see services on the live map as well as the detailed information. When GPS is disabled, no marker is displayed on the map and there is no hint explaining why the map is empty. One of the methods to solve this problem is to allow users searching by address or ZIP code. Moreover, none of the apps allow managing the GPS permission inside the app.

Second, most apps depend on the network and fetch the data from the backend database only when they need the data. It’s a wise method for real-time data, meetings and comments, or a big data set like national services data. However, when the mobile network is not available, markers can’t be displayed on the map as well. Thus, for regional services dataset, creating a local database would be a better choice.

Third, apps provide insufficient information. Although markers on the map with brief information visualize the locations, it is not enough for people to judge if the facility is helpful. Distance is one of the factors people consider when selecting treatment facilities, so apps should offer more detailed information. Meanwhile, two various resources, meetings and services, are both crucial for the treatment and recovery process of people suffering from SUDs. But, all apps are focusing on either various meetings or categorized services. To display regional meetings and services inside a single app is more convenient for local SUD patients and their families.

Finally, none of the apps offers a public place for users to make their comments, which means there is no feedback for the providers. Since comments are critical resources for developers, providers, as well as policymakers, it’s worthwhile to generate a review system. However, it’s not a simple system just for storage and displaying comments because it may get spamming and inappropriate language from the user, which shouldn’t be allowed when populating comments. Therefore, an automatic filter is also essential.
Chapter 3

FUNCTIONALITIES AND DESIGN

This section introduces the functionalities and design of HeNN app, including its architecture, plugin, and data structure.

3.1 HeNN Functionalities

HeNN (Help Near & Now) is a smartphone app in Android and iOS that provides a location-based, interactive platform to connect people suffering from SUDs, and their families, to services where and when they need them. For now, HeNN is a regional mobile app, which focuses on offering help to SUDs around Delaware and its surrounding area.

To serve SUD patients and their families better, HeNN categorizes local services according to different stages: Prevention & Education, Health Resources, Treatment, and Recovery Support. Every category contains a list of facilities with their distance or online resources with their websites. For each service, HeNN not only provides text information like phone number, address, link to the official website, but also offers the number of available beds as well as real-time distance based on the user’s GPS location. Users are allowed to search a specific service by its name. Besides, HeNN shares upcoming events, Alcoholic Anonymous meetings, Narcotics Anonymous meetings from different resources, which is also location-based by showing the distance. To recommend resources and events that meet user’s requirement, HeNN allows users to set their preferences and saves settings to the local database.

For providers and policymakers, HeNN establishes a peer network by allowing users to provide comments on resources. The backend analysis for the comments and component usage informs them about the utilization and effectiveness of services.
3.2 HeNN Design

One of the crucial parts for app design is to transfer expected functionalities to the original architecture, which guides programmers to implement.

3.2.1 Architecture

In order to fulfill the functionalities and requirements, HeNN plays the role of building bridges between service providers and users (see Figure 3.1). First, HeNN collects resources from service providers according to the requests from the patients and their families. After cleaning and classifying the data, HeNN provides location-based service recommendation corresponding to the user’s preference. By analyzing the comments and tracking on the usage of each app component, HeNN can provide better feedback to the service provider.

![Figure 3.1: Relationship between Service Provider, User and HeNN App](image)

The design framework that HeNN applies is the sitemap (see Figure 3.2). HeNN consists of two types of helpful information—services and events.

For SUD services, users are allowed to browse classified resources or search a specific one by its name. The Service List page is for displaying links to online resources or navigating users to a detailed page for facilities, providing text information like address, phone number, website and number of available beds in the overview tab. The direction tab presents the distance to the destination by driving and walking, with links to the live map respectively; while the comment tab displays the reviews and allows users to leave their comments after they log in.

The Event page lists upcoming events with their distances by driving and the links to 12-Step Meetings. The event filter allows users to sort events by distance, time,
or type. Besides, the Settings page is for saving user’s preference of event time and distance locally. There is a help screen for clarifying definitions and contact information, presenting the mission statement and privacy policy.

### 3.2.2 Development Tool

Flutter, created by Google and first launched on Dec 4th, 2018, is a free and open-source mobile application development framework[26]. Even though Flutter is a new platform and its plugins are not completed, Flutter is utilized in this project for several reasons.

Most importantly, Flutter can develop mobile applications for both Android and iOS at the same time. In the US, both operating systems take half of the market share of operating systems[25]. Flutter saves time and reduces the complexity of app development.

Another advantage of Flutter is the hot reload feature. It can quickly deal with the changed code, like the size or layout of widgets, and reload the page immediately instead of restart the whole app. This feature brings convenience to the development.
3.2.3 Plugin Used

In computer science, application programming interface (API) builds the connections among various components by communication protocols [27]. APIs in this project are utilized as the approaches for transferring data. HeNN app mainly leverages Flutter plugins for Google Map Platform, Google Calendar API, and Firebase API.

A group of APIs in Google Maps Platform are utilized. For example, Google Maps API is for building static and dynamic maps which display the real world to the users. Google Routes API gives the user choices of high-quality direction and helps to calculate travel times and distances for multiple destinations. Google Places API locates user and converts addresses to geographic coordinates. Most of these APIs create the JSON-formatted response based on URL parameters sent through a standard HTTP request.

The Google Calendar API integrates mobile apps with Google Calendar. We manage our events from different organization through various calendars in our Google Calendar Service Account. This API assists in pulling recent events and meetings in specified date or time period.

The role of Firebase API is to retrieve data stored in Cloud Firestore and provides statistics on app usage and user engagement [28]. The Cloud Firestore is temporary database for saving services data. By sending an push request, HeNN receives the JSON-formatted response and uploads the data to the local database. Google Analytics for Firebase tracks the interactions between users and HeNN, such as the number of clicks on buttons, the services users look through and times for launching the app.

3.2.4 Data Structure

HeNN, as a smartphone app, utilizes two kinds of database—local database and backend database.

SQLLite, the SQLite plugin for Flutter, is used to construct the local database. The general information for resources, such as service name, address, phone number,
website, is stored in the local database for two reasons. On the one hand, basic information on facilities and online resources rarely change, so there is no need to fetch the list from backend every time the app is launched. On the other hand, users are still able to browse the services offline when the network is not available. Users’ preference and settings are also located in SQFLite database for safety purpose.

The backend database is for saving real-time data like comments or the number of available beds, which updates frequently. The services data is stored in Firebase that allows HeNN to download when it is launched for the first time or services data is modified.

In the future, as we spread to more areas, our dataset is growing. We may consider to move our local service database to backend, if it takes too much storage from the mobile device. To improve the quality of growing services data, we may leverage an algorithm to regularly filter out the candidate invalid facilities and check them manually, in case the service is shut down but we are not informed.
Chapter 4

IMPLEMENTATION

This chapter is divided into two sections. The first section details the HeNN user interface, which provides interactions to the public via the screen of smartphone. The second section is basically app’s backend service which helps HeNN talk to database or API.

4.1 HeNN User Interface

The user interface follows the sitemap in Chapter 3, which consists of a root file and 8 main screens with 16 files in screens folder:

Table 4.1: Correspondence from File to Screen

<table>
<thead>
<tr>
<th>Screen</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home Screen</td>
<td>home_page.dart</td>
</tr>
<tr>
<td>Services List Screen</td>
<td>category_page.dart</td>
</tr>
<tr>
<td></td>
<td>category_service_item.dart</td>
</tr>
<tr>
<td></td>
<td>category_filter_dialog.dart</td>
</tr>
<tr>
<td>Service Screen</td>
<td>service_page.dart</td>
</tr>
<tr>
<td></td>
<td>service_overview.dart</td>
</tr>
<tr>
<td></td>
<td>service_direction.dart</td>
</tr>
<tr>
<td></td>
<td>service_comment.dart</td>
</tr>
<tr>
<td></td>
<td>new_comment_page.dart</td>
</tr>
<tr>
<td>Search Screen</td>
<td>search_page.dart</td>
</tr>
<tr>
<td>Events Screen</td>
<td>event_page.dart</td>
</tr>
<tr>
<td></td>
<td>event_page_item.dart</td>
</tr>
<tr>
<td></td>
<td>event_filter_dialog.dart</td>
</tr>
<tr>
<td>Settings Screen</td>
<td>settings_page.dart</td>
</tr>
<tr>
<td>Login Screen</td>
<td>login_page.dart</td>
</tr>
<tr>
<td>Help Screen</td>
<td>help_page.dart</td>
</tr>
</tbody>
</table>
Main.dart represents the root for HeNN, which builds a MaterialApp widget and sets the theme of the app. The builder attribute here is for control the font size of the whole app. Without this part, the font size of app would be different as the system font size changes. The home parameter of MaterialApp is built with the _buildBody() method and the _buildBottomNavigationBar() method with four tabs — Home, Search, Events and Settings, in order to navigate users to different screens. The pageRouteFactories map indicates the four pairs of names and page routes. The Home Page is the first screen when the app is launched, so it’s represented by a slash here. The objects, analytics and observer in previous code, are utilized to track user's usage of each button and send the result back to Firebase server, which is also applied in other pages.

"/": () => MaterialPageRoute(
    builder: (context) =>
        HomePage(analytics: analytics, observer: observer))
Another function that implemented in the main.dart file is the recommendation on services based on user’s location by popping up a notification (see Figure 4.1) when HeNN is launched. In the initState(), the function initializes the location notification settings for Android and iOS with the app icon, and then loads initData() for finding the closest services. The initData() function first verifies the GPS and notification permissions which indicates whether the user wants the recommendation service. If allowed, HeNN downloads all the information of services and calculates their distance to find the closest one to recommend. Finally, it calls the _showNotificationWithDefaultSound() to push the notification with a simple text revealing the distance and service name. The function onSelectNotification navigates user to a new page (see Section 4.1.3) with more detailed information.

4.1.1 Home Screen

The home screen (see Figure 4.2) contains two section, the upper logo part and bottom main function part.

Figure 4.2: Screenshot for the Home Screen of HeNN
The following two parameters in build() method obtain the height and width of device screen, which are used to calculate the size of widgets:

```dart
final double _deviceHeight = MediaQuery.of(context).size.height;
final double _deviceWidth = MediaQuery.of(context).size.width;
```

The top one third part is a Stack widget consists of an app logo, a slogan with an alert sign in up-left corner and a question mark in up-right corner. The alert sign is a button which pops up a dialog (_emergencyCall() function) for displaying phone numbers for emergency call. The _launchURL() function from url_launcher package navigates user to the dialer and makes a phone call.

Below the slogan, there are four buttons corresponding to four kinds of services, Prevention & Education, Health Resources, Treatment and Recovery Support, sorted by the stages of SUD. The map _buttonParameter stores the categories, subcategories and the color for each category. The _button() customizes the button with appropriate size and color, and navigates user to the list of services. To track on user’s click of these buttons, the _buttonTrack() function is called by the method in the onPress attribute of the button. The name of the button clicked, such as Prevention & Educatin or Treatment, is passed to the logEvent() method as a parameter of logEvent. The following lines of code shows the _buttonTrack() function:

```dart
Future<Null> _buttonTrack(String type) async {
  await analytics.logEvent(
    name: 'home_screen_button',
    parameters: <String, dynamic>{'name': type});
}
```

Then in the Events tab under Analytics of Firebase console, there is an event named home_screen_button with its times of click. More details will be discussed in Section 5.2.

### 4.1.2 Service List Screen

The service list screen (see Figure 4.3) mainly displays a list of online resources and facilities with their general information by three files:
Figure 4.3: Screenshot for the Service List Screen of HeNN

Category_page.dart is the root file of the screen which can be divided into two parts, data managing part and widget displaying part. The data managing part has three main functions, initData(), _filterDataList() and _sortDistance().

InitData() function is for data preparation:

```dart
await _sqfliteDbProvider.initServices();
bool _hasData = await _sqfliteDbProvider.hasData("services");
if (!_hasData) {
    List<Map<String, dynamic>> serviceData;
    serviceData = await _databaseApiProvider.fetchData();
    _sqfliteDbProvider.uploadServicesData(serviceData);
    _rawData = await _sqfliteDbProvider.fetchCategoryList(category);
}
```

First, the initServices() method initializes local database, services.db and checks if the database is empty by hasData() method. If it’s empty, the checkConnectivity()
in connectivity package helps to confirm the status of network connection. Then, `fetchData()` and `uploadServicesData()` methods download data from Firebase and upload it to local database. `fetchCategoryList()` picks services which belong to a chosen category, like Prevention & Education here.

Similarly, `initUser()` initializes `user.db` and checks if it’s empty. If the database has been initialized, `fetchUserOption()` will fetch user’s preference on distance, GPS permission and location permission. If the HeNN is not allowed to access user’s location, `Service.fromMap(map)` directly converts the list of maps to a list of Service objects, `_servicesData`. Otherwise, the following code extracts the addresses and then calculates the distances of driving from user to the facilities by `distance()` method:

```dart
for (Map<String, dynamic> map in _rawData) {
  if (map['Address'].length > 10) destinations.add(map['Address']);
}
_await _distanceUtil.init(_gpsAllowed, _lat, _lng);
_serviceDistance = _await _distanceUtil.distance(destinations);
```

Then, the list of service objects can be created with their actual distance. At the end, the `_sortDistance()` function filters the list of services according to user’s preference and sort it by distance. In the function, the services are classified into two groups, online resources and services with a real address. The facilities are ordered in the front followed by the online resources.

The `_filterDataList()` function is utilized to filter and sort service list by different conditions. The input argument should be a list with four values, which represent user’s choices for sort, gender restriction, subcategory and distance respectively. Checking the length of the argument is to make sure the list is not empty. After filtering and ranking by user’s choices, `setState()` function reloads the list for the new data set, `_filterServicesData`.

For the widget displaying part, the `build()` function is the root of the widgets, which creates a Scaffold containing an AppBar and a body. The AppBar has a title and a filter button for the following full-screen filter dialog:
Navigator.of(context).push(MaterialPageRoute<List<String>>(
  builder: (BuildContext context) {
    return CategoryFilterDialog(
      _subCategory, _locationAllowed),
    fullscreenDialog: true
  }
).then((List<String> result) {
  if (result != null) _filterDataList(result);
});

The dialog takes two parameters, the subcategory list and user’s location permission, and returns a list which stands for user’s requirement. The _filterDataList() is called to deal with the result.

The category_filter_dialog.dart file creates a full-screen filter dialog (see Figure 4.4).

![Figure 4.4: The Screenshot of Services Filter](image)

(a) Location permission enabled  
(b) Location permission disabled

The radio buttons in this page are customized widgets built in filter_radio.dart file. There are four groups of radio buttons in this page which are initialized in
initState() function. The build() function builds the scaffold of the dialog containing radio buttons with clear and submit buttons at the bottom. Each group of the radio button has a corresponding function to set the state of the group in setState():

```dart
sortRadio.forEach((element) => element.isSelected = false);
radio.isSelected = true;
result[0] = radio.text;
```

The clear button provides a shortcut to reset all user’s options to default; while the submit button navigates user back and passes the options to the list screen. Whether user enables location permission or not makes the the dialog different in the sort option and the distance radio buttons.

The first line of the Service List Screen, built by the tags() function, explains the definition of each abbreviation, which is also clickable as a simple filter for the subcategory. Then, the _listView() function displays the list for services. Here, the _firstLoad parameter checks if the page is loaded for the first time. If yes, HeNN should wait for initializing services data by FutureBuilder widget which displays the context after initData() returns the value. While waiting, the screen shows a centered circular progress indicator. Alternatively, if the page is reset by the filter function, services should be listed immediately. The interface() function visualizes the list of services by a ListView. In onTap attribute of each item, it’s checked if the service is an online resources. If yes, HeNN launches the its website; otherwise, HeNN navigates user to the detailed information page.

Each item of the list is built by the category_service_item.dart file including the name, distance, rate, treatment type, address and the abbreviation of subcategory. Since the ways of displaying online resources and facilities are slightly different, most widgets of the service are built by a function with a verification of the distance or address attribute. For example, if the distance is zero, the _nameDistanceText() function will hide the distance text on the right side of the service name. Besides, the
images of the abbreviations are saved in the images folder and utilized by calling its path.

Same as the tracking process of the four buttons in home screen, the functions `_serviceTrack()` and `_linkTrack()` generates a report containing the name of the services, which are clicked and browsed by the users.

4.1.3 Service Screen

The service screen (see Figure 4.5) reveals the detailed information of the services with the its direction and comments. This page involves five files:

- service_page.dart
- service_overview.dart
- service_direction.dart
- service_comments.dart
- new_comment_page.dart

![Figure 4.5: Screenshot for the Service Screen of HeNN](image)

Service_page.dart is the root of Service screen which holds a static map, the general information, and a tab bar. To present the static map, the `_geocode()` function is called in the FutureBuilder widget. In order to make the map more functional, the
_showMap() function displays a full-screen map when user clicks on static map. The following code applies the geocode package to convert the address to a pair of latitude and longitude in _geocode() function:

```dart
var addresses = await Geocoder.local
    .findAddressesFromQuery(_service.address.toString());
var first = addresses.first;
```

Next, the getStaticUriWithMarkers() method from map_view package helps to build the url of the static map with required parameters, like the position of the marker and size of the image.

```dart
_STATIC_MAP_URI = staticMapProvider.getStaticUriWithMarkers(
    [Marker('position', _service.name, _latitude, _longitude)],
    center: Location(_latitude, _longitude),
    width: width.toInt(),
    height: height.toInt(),
    maptype: StaticMapViewType.roadmap);
```

The main content of the Service Screen is the tab bar, containing three tabs, overview, directions and comments, which correspond to different views of the tab bar.

The overview tab (see Figure 4.5) matches the service_overview.dart file which is composed of four lines, address, number of beds, phone number and website. Except for number of beds, each line in this section is clickable, which leads user to the browser or dialer.

The directions tab (see Figure 4.6) matches the service_directions.dart file. Likewise, the direction tab has an initData() function to get user’s setting on location permission from the user database and utilizes the distanceDuration() method from distance util. Then _distanceDuration() function displays the tab under different conditions. If the location permission is disabled (see Figure 4.6b), it presents a hint. If enabled (see Figure 4.6a), it shows the distance and duration by driving and walking as well as another option to launch Google Map. When click the driving direction button,
Figure 4.6: Screenshots for the Service Screen of HeNN (Directions Tab)

(a) Location permission enabled
(b) Location permission disabled

A full-screen map (see Figure 4.7) pops up with the direction and markers. The green marker is user’s location while the red one is the destination.

Figure 4.7: Screenshot for the Full Screen Map

The comment tab (see Figure 4.8) matches the `service_comments.dart` file and lists the comments from other users for a chosen service. If the users want to leave their own comments, they have to sign up a new account or log in.

The `initData()` function is here to visit `admin.db` and check user’s account and
login status. If user has already logged in with the account, a token would be fetched as the requirement for creating new comment. The `getComments()` method obtains all comments from backend database and verifies if it's an empty list. The `listView()` function constructs the comment list while each comment item is designed by the `_commentItem()` function.

After user has logged in, the new comment icon turns blue and navigates user
to the new comment page (see Figure 4.9) which allows users to leave their rates and comments. The submit button navigates user back to the service page and upload rate and comment to the background real-time database.

4.1.4 Search Screen

The Search Screen (see Figure 4.10), which involves search_pages.dart file and the Search tab at the bottom, allows user to search for services by its name.

![Search Screen Screenshot](image)

**Figure 4.10: Screenshot for the Search Screen**

The `initData()` function here fetches all services data from local services database and extracts all service names to form a list(_nameList) for searching.

In the `build()` function, The `WillPopScope()` is for controlling the back button for Android. By setting the `onWillPop` attribute, the back button navigates user to the home page rather than exit the app. The AppBar contains a TextField as the search bar and a close icon to delete all the search queries.

The function _buildSearchList() generates the content of this page. The `suggestionList` holds the candidate choices according to the query. If the query is
empty, instead of returning nothing, the screen shows user’s recent search results. The ListView builds a list of ListTile showing searching result and navigates user to the detailed information page for the services or launch the website of online resources.

```dart
final suggestionList = _searchText.isEmpty
  ? recentSearch : _names.where(
    (p) => p.startsWith(RegExp(_searchText, caseSensitive: false)))
  .toList();
```

The _SearchPageState() adds a listener to the search query in TextField and reloads the whole page when the query is changed. That’s the reason why the searching result keep changing as users continue typing.

### 4.1.5 Events Screen

The Events Screen (see Figure 4.11) displays the events from different resources and consists of following three files:

- `event_page.dart`
- `event_page_item.dart`
- `event_filter_dialog.dart`

The Event Screen has two tabs, respectively for the events list (see Figure 4.11a) and 12-step meetings (see Figure 4.11b). The 12-step meeting tab shows the links to Alcoholic Anonymous (AA) Meetings and Narcotics Anonymous (NA) Meetings by `_stepMeetings()` function. The events tab is built as a FutureBuilder in `_listView()` function, which waits for the result from `initData()`.

The `initData()` function can be divided into three part. First, get user’s permission on location, preference on the time and distance for the events, from user database. Then, check if user has selected a specific date. If yes, call `getDayEvent()` method which directly returns events in that date; if no, gather all events in a period of time and pick events corresponding to user’s preference. Finally, calculate the distances if user enables location permission.

After required data has been initialized, the `interface()` function generates a date picker and the ListView. The date picker provides a short cut for users to browse
the data in a specific date. Once the new date is picked, the screen is reloaded with a new list of events. The clear button is for going back to the events in future period of time.

The layout of each event is managed by `event_page_item.dart` file, with the date on the left, detailed information in the center and a share button on the right. If the location permission is enabled, the distance will be displayed below the share icon. The center part is a GestureDetector which is clickable and launches the Google Map with the event’s location. The share icon utilizes the share package that allows to share the events in text format through email, text message, or copy to the clipboard.

The filter dialog of Events screen (see Figure 4.12) is much similar to the one in Service List screen, with various status for different location permission. The filter allows setting preferred time period, type of the event and distance. The preferred time period contains the amount of days and the time in a day, which is set by two time...
pickers representing the starting time and end time. The filter also returns a result list and _filterData() function sorts the events according to the result.

Two variables should be noted here, _firstLoad and _selectDate. _firstLoad checks if the page is loaded for the first time; while _selectDate means if user has selected a date. Only when the page is loaded for the first time or user picks a date needs HeNN to connect the Google Calendar and takes some time to gather the events which builds the list inside FutureBuilder. Otherwise, when the page is reloaded by the filter, _firstLoad and _selectDate are set to false, which means we don’t need the outer layer, the FutureBuilder to spend much time on filtering existing event list.

### 4.1.6 Setting Screen

The settings_page.dart constructs the layout of Settings Screen(see Figure 4.13) which helps users to set their preference and profile.
The `initData()` function is responsible for checking user’s account and preference from the local database, admin and user. Subsequently, the `FutureBuilder` in the `_body()` function exhibits the interface in `interface()`. In user profile part, the `SwitchListTile` controls user’s login status, and displays the username after logging in successfully (see Figure 4.13a). If user launches HeNN for the first time or has signed out the account (see 4.13b), the click of switch button navigates user to the login page which is another full-screen dialog. In user preference part, the distance is a group of radio buttons if the location permission is enabled (see Figure 4.13a). The row of setting time for events consists of two time pickers in `_timePicker()` function.

The location service section (see Figure 4.14) contains two `SwitchListTiles` for location permission, GPS permission and the `TextFormField` for entering the ZIP code. The location permission performs whether user wants to use the location service like the distance service. Then HeNN provide two ways for sharing user’s location if they
want the distance — one is allow GPS and the other is input a ZIP code. There is a validator for the text field to simply check whether the input ZIP code is valid. Otherwise, if user enables location, they have to either allow GPS or enter a ZIP code, that is another usage for validator.

![Location Service](image)

**Figure 4.14: Screenshot for the Location Service Section**

The clear button empties all user’s selections in settings page as well as the local database. The save button first validates the input ZIP code and transfers it to a pair of coordinates by `_geocode()` function. Then it generates a map and uploads it to user database. Therefore, the save button must be clicked after user has made any changes.

### 4.1.7 Login Screen

Both settings screen and comment tab in service screen can navigate user to the login page (see Figure 4.15) built by `login_page.dart`.

The FlatButton with the text, Switch to Signup/Login, in the center directly controls the mode to the page by the `setState()` function. The login mode (see Figure 4.15a) contains a username TextField and password TextField; while the sign up mode (see Figure 4.15b) has two more TextFields for confirming password and the email. The function `_signUpMode()` returns those two more TextFields and whether show these two widgets depends on the value of `_authMode`. The logo size and text of the button work as the same way.

The Form widget is for validating the text in all TextFields when the button is clicked. Each TextField applies a validator attribute that is utilized to check the input value, for example the length of username, the format of email and if the input
passwords is the same as the confirm password. Then login or signup button return from `_submitButton()` validates the value by following code:

```
if (!_formKey.currentState.validate()) return;
_formKey.currentState.save();
```

The `save()` method calls the functions in the `onSave` attribute of all the TextField to save the values in an `admin` object. If any of the value is invalid, the hint appears below the TextField.

For the signup mode, after checking the availability of the network, `signUp()` method of `CommentsDbApiProvider` creates the new account in the backend database and returns the status code. If success, the returned token is saved and the admin status is updated in the local database through `_submitLocalDb()` function, same as the login mode.
4.1.8 Help Screen

The help screen (see Figure 4.16) is launched by the question mark icon in upright corner of the home screen. This is a place for displaying HeNN’s mission statement, contact information and the definitions of the abbreviations, and the FAQs in the future.

![Help Screen Screenshot](image)

Figure 4.16: Screenshot for the Help Screen

Each section here is placed inside a ExpansionTile within a Card, which is easy to use by clicking and expanding. The definition() function sets up a list of Text widgets automatically according to the definitions map.

4.1.9 Customized Radio Button

The filter_radio.dart creates three classes which builds customized radio buttons for the app. The RadioModel controls the status and title of each radio button, while two FilterRadio classes generate the appearance of radio button in different size. As the isSelected attribute changes, the color of the radio button is also modified.
4.2 HeNN Backend Service

The back-end service of HeNN consists of three packages — database, map and utils. The database package communicates HeNN with Firebase, local SQLite database and Google Calendar. The map package is for calculating distance, locating user and displaying the map. The utils package contains two classes, which are used to create service object and event object.

4.2.1 Database Package

The database package consists of three source files, database_api_provider.dart, sqflite_db_provider.dart and calendar_provider.dart.

4.2.1.1 The DatabaseApiProvider Class

The DatabaseApiProvider class in database_api_provider.dart is utilized to fetch services data from Firebase. The fetchData() method first sends an HTTP request by http.get() and uses json.decode() to convert the response, a json file, to a map by following code:

```dart
final http.Response response = await http.get(_url);
final Map parsedJson = json.decode(response.body);
```

Then, the forEach() method converts the map to a list of map order by its ID and returns the list.

4.2.1.2 The SqliteDbProvider Class

The SqliteDbProvider class builds local database on device with the help of sqlite package. There are three groups of functions for three databases, services.db, user.db and admin.db.

The services.db stores the detailed information of the services. The initServices() method initializes the database by openDatabase() function. Services.db is created in the path getting by following code with its version number:
Directory documentsDirectory = await getApplicationDocumentsDirectory();
final path = join(documentsDirectory.path, 'services.db');

The function in `onCreate` parameter creates the database, while the `onUpgrade` checks the version of database and rebuilds it if it’s smaller than the latest version number. The `uploadServicesData()` method takes a list of maps as an argument and import the maps to database through `insert()` method. `FetchCategoryList()` returns the services belonging to the input category and ordered by their names. Likewise, `fetchAll()` is the function returning all services as a list of maps.

The `user.db` is built for saving user’s setting and preference. Similar to `services.db`, the `initUser()`, `uploadUserData()`, `fetchUserOption()` are three basic methods for initializing database, sending data and fetching data. It should be noted that the boolean type is not supported by SQFLite Database, so integer one stands for true here and zero means false, which is used in the column like GPS. When `uploadUserData()` is called, a record will be built in user table. Since the settings are changeable, `updateUserData()` provides the way to update the values in the only record of user table.

The `admin.db` saves user’s username, password and login status. Also, there is a group of methods like `initAdmin()`, `uploadAdminData()`, `fetchAdminOption()` and `updateAdminData()`. In addition to above four methods, `logout()` is created for changing user’s login status.

4.2.1.3 The CalendarProvider Class

The main functionality of `CalendarProvider` class is to gather events from HeNN’s Google Calendar service account through Google Calendar API. The `calendarIds` provides the ids of the calendar which are the sources of the events.

The `getRecentEvent()` method collects events in a period of time from now on. `DateTime.now()` refers to the starting time and `timeMax` is calculated based on
the timeMin by adding the number of days. The following code is for authentication in order to get access to Google Calendar API.

```csharp
AutoRefreshingAuthClient client =
    await clientViaServiceAccount(_credentials, _SCOPES);
CalendarApi calendar = new CalendarApi(client);
```

Then, to get the events, calendarId, timeMin and timeMax are passing to `calendar.events.list()` that returns an `Events` object. Since most of the events are recurring events, `singleEvents` should be set to true which treats recurring events individually. The next step is to filter the events so that the time and location of the events shouldn’t be empty, and then converts the Event object from Google Calendar to the list of EventItem objects (see Section 4.2.3). The `toLocal()` method transfers the event time to Eastern Standard Time.

The `getDayEvent()` method is quite similar to `getRecentEvent()`. The only difference is that the argument of `getDayEvent()` is a `DateTime` object and it returns events in a specific date.

The function `sortEvents()` method sorts the events by time, since the initial outputs of two previous methods are ordered by the calendar ID.

4.2.1.4 The CommentsDbApiProvider Class

The CommentsDbApiProvider class is mainly built for user’s signup, login and functions related to comments and ratings by talking to the backend database.

The `login()` and `signup()` method send the admin object in json format as a post to the database API and get the response with a token as a certification. The `newComment()` method publishes new comments by sending the token together with location ID, rating and comment. The `getComments()` method submits the location ID as the parameter and gather all the comments from the service.
4.2.2 Map Package

The map package is responsible for assignments related to maps and places, which consists of four files:

- distance_util.dart
- gps_util.dart
- map_util.dart
- network_util.dart

4.2.2.1 The GpsUtils Class

To build GpsUtils class, the location package is imported to track user’s location. There is an abstract class named GpsUtilListener with an abstract method onLocationChange(), which is build for tracking if the location has changed.

The enableGpsPermission() method checks user’s GPS permission for HeNN. If it’s the first time HeNN is launched, a dialog pops up asking for GPS permission. Then enableGpsPermission() returns a boolean value on permission is denied or not. The initPlatformState() is method that actually returns user’s GPS location by getLocation() method in location package.

4.2.2.2 The NetworkUtil Class

NetworkUtil class is the one sends the request to Google Map API and decodes the response. The Direction API from Google Map is utilized to create the route from starting point to the destination; while the Distance Matrix API calculates the distance and duration for multiple destinations. The JsonDecoder helps to convert responding json files to maps.

The getDistanceDuration() method calculates the distance and duration between two places, therefore it gathers two parameters from the response to form a simple list by following code:

distanceDuration = [
  distanceDecoder.convert(res)["rows"][0]["elements"][0]["distance"]["text"].toString(),
  distanceDecoder.convert(res)["rows"][0]["elements"][0]["duration"]["text"].toString()];
The response in `getDistance()` method contains the distances from one position to multiple destinations, thus the `distanceList` here is a list of map:

```plaintext
distanceList = distanceDecoder.convert(res)['rows'][0]['elements'];
```

Then the loop is used to extract the distances from the map and generate a distance list.

In `getRoute()` method, after checking the returning status code, the response from Direction API is converted to a list of encoded polyline codes. The reason why not gather coordinates is that the polyline is more precise than just link the coordinate when displaying in the map.

The `decodePolyline()` method is build for the map, since the polyline code is a single string that stores a series of coordinates. (For a detailed description of Encoded Polyline Algorithm Format, visit https://developers.google.com/maps/documentation/utilities/polylinealgorithm).

### 4.2.2.3 The MapUtil Class

With the help of `map_view` package, main functionality of MapUtil class is to display the live map. The `init()` method checks whether the GPS is allowed by the user and initializes user’s location.

The `getDirectionSteps()` method shows a map with the direction from user’s location to the destination. After `getRoute()` method returns a list of polylines decoded by `decodePolyline()` , following code gets the map ready by adding markers and routes:

```dart
mapView.onMapReady.listen((_) {
    mapView.clearAnnotations();
    mapView.clearPolylines();
    mapView.setMarkers(getMarker(location.latitude, location.longitude, destinationLat, destinationLng));
    mapView.addPolyline(
        new Polyline("1", polylines, width: 5.0, color: Colors.blue));
});
```
The `showMap()` method sets some options for the live map, including titles and type for the full-screen map, buttons on the map to display the compass. `getCamera()` function is for camera position and zoom level.

```javascript
mapView.show(
    new MapOptions(
        mapViewType: MapViewType.normal,
        initialCameraPosition: getCamera(),
        showUserLocation: true,
        showMyLocationButton: true,
        showCompassButton: true,
        title: "Route"),
    toolbarActions: [new ToolbarAction("Close", 1)]);
mapView.onToolbarAction.listen((id) {
    if (id == 1) mapView.dismiss();});
```

Another method `launchGoogleMap()` provides a convenient way to launch Google Map with the route on the map. If Google Map is not installed on the device, it opens the browser showing its web page.

### 4.2.2.4 The DistanceUtil Class

Similar to `MapUtil`, `DistanceUtil` class has an `init()` method as well to check if GPS is allowed.

The `distance()` method takes a list of addressed as an input and outputs a list of distance corresponding to each address. It should be noticed that Google Map Distance Matrix API takes up to 25 addresses as destination at a time, therefore there is a loop to controls the number of addresses. Every address is checked by the regular expression below to replace the characters:

```javascript
url += destinations[i].replaceAll(new RegExp(r"[@&/]"), "")
    .replaceAll(new RegExp(r"\s+"), "+");
```

The `distanceDuration()` takes two argument, the latitude and longitude of the destination. According to the requirement, HeNN should display both distance
and duration by driving and walking, thus \texttt{network.getDistanceDuration()} is called twice to calculate under two different travel mode.

4.2.3 Utils Package

The utils package contains three classes \texttt{Admin}, \texttt{Service} and \texttt{EventItem}.

The \texttt{Admin} represents user account information including username, password, token, email and the log in status. The \texttt{fromMap()} constructor and \texttt{toMap()} methods converts between the map and \texttt{Admin} object. The \texttt{loginMap()} and \texttt{signupMap()} methods gathers required information for either login or signup process to form a map.

The \texttt{Service} class is built for all online resources and treatment locations displaying in category page. The named constructor \texttt{Service.fromMap()} provides a shortcut that converts the map to a service object.

The \texttt{EventItem} stands for the events from Calendar and the variable \texttt{organizer} is recognized from different calendar ID. The method, \texttt{getDay()}, \texttt{getMonth()}, \texttt{getWeek()} and \texttt{getTime()} transfer the integer value like the month or weekdays to a string.
Another vital feature of HeNN is offering feedback to resource providers by analyzing backend data from users. After HeNN has been officially released in the app store, we can collect service reviews and usage of the app from users for preliminary analysis. Since HeNN is in the testing stage, this section will introduce some possible potential analysis based on the app.

5.1 Comment Analysis

Reviews are crucial because they share customer’s evaluation on places or services. To understand patients’ requirements and serve them better, instead of taking much time to read a large amount of reviews, analyzing the comments by algorithm and extracting the keywords save much time. On account of unfinished reviewing system, we haven’t got any reviews from the user, so that I crawled the comments on listed services which have reviews in Facebook and Google Review, in order to display the method for comment analysis.

The result for crawling the reviews of 32 treatment services is a total of 330 positive reviews($\text{Rate } \geq 3$) and 109 negative reviews($\text{Rate } < 3$). To understand user’s attitudes towards the treatment program and services, the adjectives and surrounding nouns are more meaningful than others. Thus, all the adjectives were picked separately from two kinds of reviews and picked nouns in surrounding five words of each adjective. The nouns represent the topics people talked about, while the paired adjective expressed their attitudes. For visualizing the result, word clouds were created...
respectively for adjectives and nouns from positive reviews (see Figure 5.1) and negative reviews (see Figure 5.2). The larger the size of words, the more frequent the word occurs in reviews.

By comparing terms and analyzing adjective-noun pairs from two kinds of reviews, it’s evident that although the adjectives are different, the nouns are similar. What SUD patients and their families care about most was the qualification or professionality of the staff in treatment centers, and staff’s attitude. This is indicated by words like "people", "staff", "counselor". They also shared their impression on the services by mentioning "service", "place" or "experience" with different adjectives. However, there are still some differences between positive and negative reviews in topics they talked about. In positive reviews, patients also recommended the treatment program, which they thought it’s helpful and expressed gratefulness for saving their lives. On the other hand, patients left negative reviews and complained about much time or money they spent without getting effective therapy as expected.
Except for the statistics of all reviews, it also allows analyzing reviews from a single treatment center to see the pros and cons of their services. Figure 5.3 and Figure 5.4 display the word clouds for Christiana Care, which has 15 positive reviews and 22 negative reviews. Combining the adjective-noun pairs, the main reason that people recommended Christiana Care is because of physician’s great kudos and caring nurses; meanwhile, other people complained about spending many hours in the waiting room.

![Word Clouds for Christiana Care](image)

Figure 5.3: Term Counting for Positive Reviews from Christiana Care

Figure 5.4: Term Counting for Negative Reviews from Christiana Care

By analyzing the comments, we can conclude that the two factors people considering most are the professionality and attitude of the staff who provides SUD counseling or treatment. They also care about the effect of treatment programs and the time or money they would spend on the program. The feedback from patients and their friends, families to the service in the treatment center becomes a meaningful reference for providers to improve their SUD programs according to people’s requirements.
5.2 Component Usage Analysis

For the developers and providers, another useful information comes from the smartphone app itself. By tracking on the usage of HeNN, like the number of click for each button, we are able to learn more about user’s preference in a specific category of services or events. In Section 4.1.1 for introducing the implementation of Home Screen, the Firebase Analytics package is used for tracking the usage of each button. Similar trackers are also added to the list of services and events so that the name of resources can be recorded and sent to Firebase.

![Figure 5.5: The Report of Events Analytics from Firebase Console]

Figure 5.5 displays the testing result from the Events tab under Firebase Analytics section. The Event name column lists the name of each setting tracker with some default trackers, while the Count and users column shows the numbers. For example, the home_screen_button indicates that the four main buttons in the home page are clicked for 37 times by six different users. Besides, two opposing arrows here denote the column for growth rate. The event_track, online_resources_track, and service_track are recording the names of services or events user visited inside the app, which measures the popularity of every resource. The effect of local notification
function is also tracked by the `notification_open` parameter, which means how many times the notification has been noticed and clicked by users.

![Table showing button click statistics](image)

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
<th>Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>29</td>
<td>4</td>
</tr>
<tr>
<td>Prevention &amp; Education</td>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td>Treatment</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>Recovery Support</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Health Resources</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 5.6: The Number of Click for Buttons in Home Screen

Figure 5.6, the detailed page for `home_screen_button` event, demonstrates the actual count for each button in the home screen and visualizes it with a line chart. In this case, comparing the number of click for buttons, we are able to see that the resources for Prevention & Education are more frequently visited, which may indicate the user’s interest.

Even though the tracker only records simple data like the statistics of click, it’s still powerful because the number of trackers is unlimited and could be added to all interested components of HeNN app. Also, the result is straightforward, which describes the interaction between users and the app. For developers, the tracking data reflects if the interface is easily operated for the user and if the functionalities meet user’s requirement. For example, if the local notification for service recommendation is displayed, but hasn’t been clicked by the user. Then this indicates that the users may not be satisfied with the content of the recommendation. For providers, the analysis for components may help to understand people’s preference in choosing services and events through the trackers for the filter or settings.
Chapter 6

CONCLUSIONS AND FUTURE WORK

This thesis summarizes our design, functionalities, the implementation for the mobile application, HeNN, and potential analysis for backend data. We developed HeNN based on Flutter that supports Android and iOS. We also leveraged APIs from Google Map Platform to calculate distances and display directions, Google Calendar API for collecting and managing events, and Firebase API to track the usage of app components. The app provides a location-based, interactive platform to connect SUD patients and their families with the resources in Delaware and surrounding areas.

The impact of the project can be seen in two aspects. For SUD patients and their families, HeNN categorizes local services as well as upcoming events and provides all detailed information with distances and directions based on the user’s GPS location. The resources recommended are customized according to people’s preference in settings. HeNN aims at providing high-qualities resources and saves people time in finding what they want.

For providers and policymakers, HeNN establishes a peer network by allowing users to provide comments on resources. The potential analysis for reviews and the usage of components may explain people’s concern in selecting SUD programs. Meanwhile, it provides feedback about the utilization and effectiveness of services, by counting various words mentioned in people’s comments or the statistic for click on service detailed information page.

For the future work, there are still some improvements for HeNN and its backend analysis.

First, the service recommendation system could be customized to provide more accurate recommendations. The aim of push notification is to select one treatment
center out of all the resources that may best fit the patient’s need. Therefore, more personal information like people’s gender, age, preference for distance and time, type of insurance, and stage for treatment is helpful for the process of customizing.

Second, as HeNN used by people from other regions, the data for services and events will increase significantly. The place for storing backend database and the method of retrieving data should be modified considering efficiency and sustainability. For now, the local database for treatment resources doesn’t take up much storage in mobile devices. When the size of database grows, the backend database would be a better place for saving services data. Besides, to improve the quality of growing dataset, we need to leverage an algorithm to regularly filter out the candidate invalid facilities and check them manually, in case the service is shut down and we are not informed.

Third, for potential analysis, more complex text mining strategies could be applied for analyzing comments instead of term counting. For example, we may summarize the topics users frequently talked about by clustering or pick the keywords as the tags for services. Alternatively, we can sum up the pros and cons from the reviews of each service.

Finally, with more and more reviews from users, the comment system should be optimized. After reviews are submitted, the one that is not informative or contains inappropriate language should be detected by a filtering system. Then, we need to avoid analyzing these comments or displaying them in the app.
BIBLIOGRAPHY


