

# Group 4 - Final Report

28th July 2021

**Course Name:** PBL 3: Creative Design - Software Engineering

<b>Name</b>	<b>Student ID</b>
Yasith Dhamsara Bandaranayake	2600200486-0
Noraishah Mohd Rapi	2600200518-1
Rifqi Ihsan Nabil	2600200527-0
Stanley Jusuf	2600200533-5
Rikuto Momoi	2600200416-9
Ian Suzuki	2600200547-5

# 1 Introduction

*Responsible Member: Noraishah Mohd Rapi*

The idea of our app came from the problem of textile waste accumulation across the globe. For now, we are specifically focusing on one of the major cities of fashion, the New York City, where on average of 200 million pounds of textiles ended up in landfill every year. It was estimated that 95 percent of this landfilled clothing still has market value and could be recycled.

In correspondence with raising awareness regarding the issue, it is important for the public to be able to play their role as well. Therefore, the goal of our app is to make it convenient for people to locate nearby textile drop-off centers around the city of New York to have their clothes recycled or donated instead of going into dumpsters.

## 2 Literature Review

*Responsible Member: Yasith Dhamsara Bandaranayake*

The textile recycling business is considered as big as the clothing industry. The recycling business itself can have an impact on the economy of a country. The secondhand market is also big as the clothing market. The secondhand market around the world has created a number of opportunities for people around the globe to come up with new businesses and the number of entrepreneurs has risen as a result as well. [1]Not only has it given opportunities to many people, but the secondhand market has also provided the clothing needs to the poor countries throughout the world. However, even with the secondhand market, it is impossible to reduce textile wastage, therefore, a technological improvement is needed in textile recycling. How does this textile wastage impact the Environment? Previous literature states that most of the textile waste is either mixed into household waste or [16]ends up in landfills. According to the 2017 pulse of the Fashion industry report, Natural Fibers such as silk and [14]cotton have the highest impact on the environment. Furthermore, Synthetic fibers can discharge microfibers which can release toxins and may end up in our food chain. In addition, textile wastewater can have harmful effects on aquatic and terrestrial life. Various greenhouse gases are being released during the production of textiles and [13][15]the use of chemicals and the high use of energy affects the environment enormously. Considering the damage caused to the environment, recycling and [2]reusing have surfaced as solutions for this issue. Taking a deeper look shows how beneficial it can be. According to previous literature, recycling textiles could be more worthwhile than reusing. However, since [11]the current technologies cannot completely recycle textile wastage, reusing comes into the equation as well. As textiles can be easily reused and cheaper, it has given the chance to people who don't have much money to buy clothes cheaply. Furthermore, it reduces the number of greenhouse gases emitted and [12]72 million cubic meters of water use. These benefits arise because of processing of used clothes is cheaper than the brand-new production of cloths. However, the attitude of the consumer plays a major role

in sustainable consumption. According to several studies, [5]consumers are not aware of the effects caused by textile waste. Therefore, if the necessary information is provided on the consequences, consumers would be buying textiles with an environmentally friendly attitude. Moreover,[17] Previous literature suggests that there should be better communication methods to convince the customer to sustainable consumption. Information on how the consumers' behavior affected the process. For example, how the chemicals were used, what happened to the waste etc. Furthermore, [18]the fashion index of a consumer also plays a role in sustainable consumption. The higher the index the lower the chance of disposing of clothes. Therefore, the knowledge of consequences and information on a product can be important when it comes to convincing customers towards sustainable behavior. Multiple pieces of research show people are used to giving away their clothes to charities. However, most of them find difficulties when finding a place to give away their clothes. This is where technology can be used. Coming up with applications that show collection points of textile. But it is important to come up with an appealing app. [8]Design aesthetics, image appealing, and [4]information quality have positive impacts on the usage of an application. In addition, [9]usefulness and practicality play a part when it comes to using the app. Furthermore, [3]price does not affect the user of the application. Therefore, to make it appeal it should be user-friendly and should have [7]sufficient information for the customer.

## 3 Content

### 3.1 State Diagram and other diagrams

*Responsible Member: Stanley Jusuf*

Because we have finished our storyboard, we had a clear idea on how we want our application to function. As such, we set our sights into how our app would operate by making system flowchart and state diagram. Our final draft of the system flowchart can be broken down into two parts. The first part, which is the setting and tips display, are created to complement the user with customizability and useful guides for recycling clothing materials as well as a platform to post a question or suggestion. The second part, which are the map functions, are the focus of our project. As represented in the system flowchart, with our application, users have the choice of finding recycling facilities in three ways, through a search bar, filter, or user GPS. Each of these search functions which will run through an online database through a REST API that returns location and other information of facilities in the form of JSON format. And that JSON format will be used to pinpoint the location of recycling facilities as well as display information regarding said facilities.

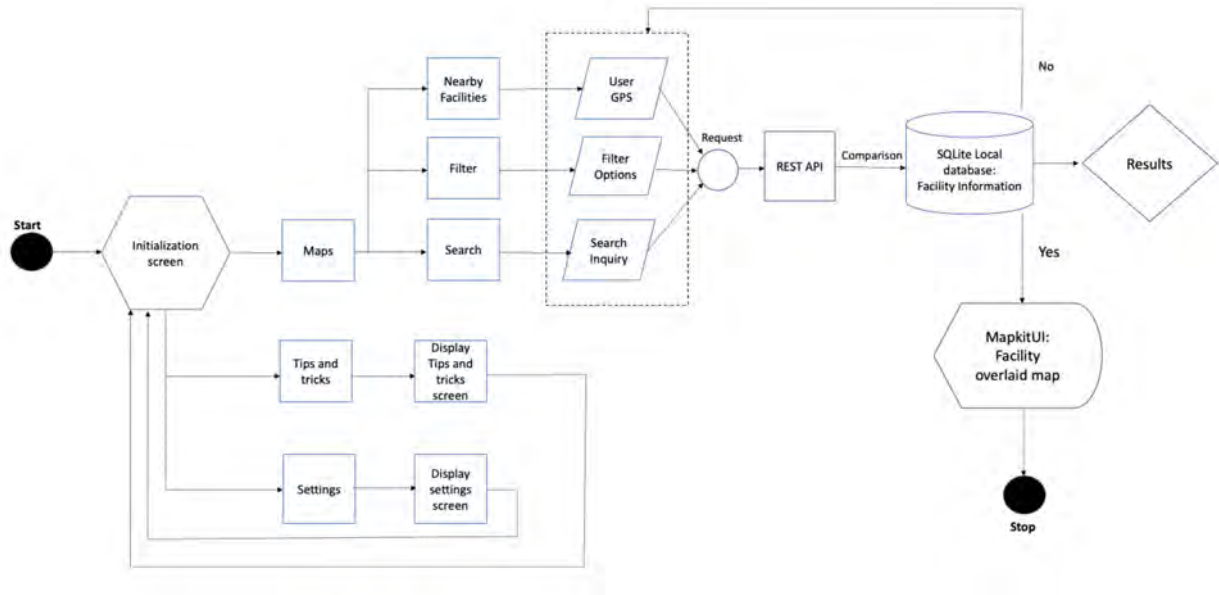


Figure 1: Figure of the final flowchart.

Next, we made our state diagram. We made our state diagram based on our storyboard. We made our state diagram to better understand how our potential users can navigate through our application. Our state diagram can be divided into four parts. The first part is the initial state, which represents the initial screen users can see when entering our application. The second part is the “Tips and settings” state which represents the screen when users tap the tips and settings button. The third part is the state called “Map Display” which represents the state when users start searching for recycling facilities. Finally, the “Facility Informations” state represents the condition if users tap on one of the facilities.

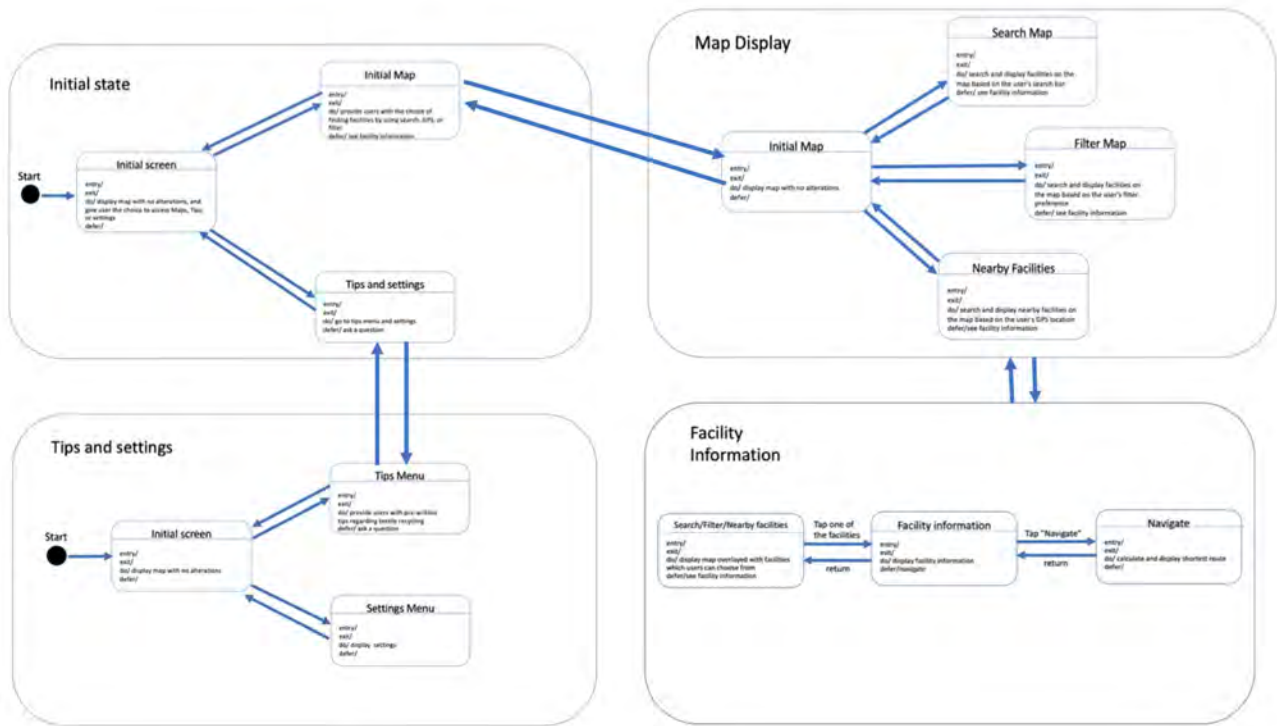


Figure 2: Figure of the Overall State Diagram.

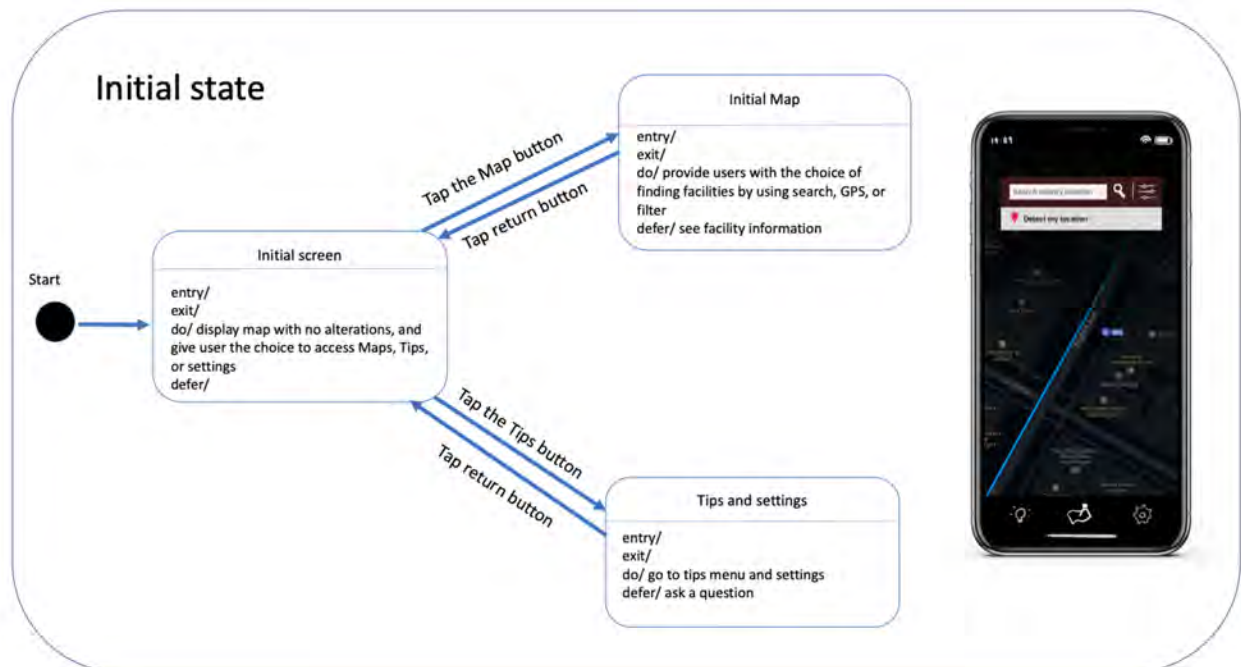


Figure 3: Figure of the Initial State.

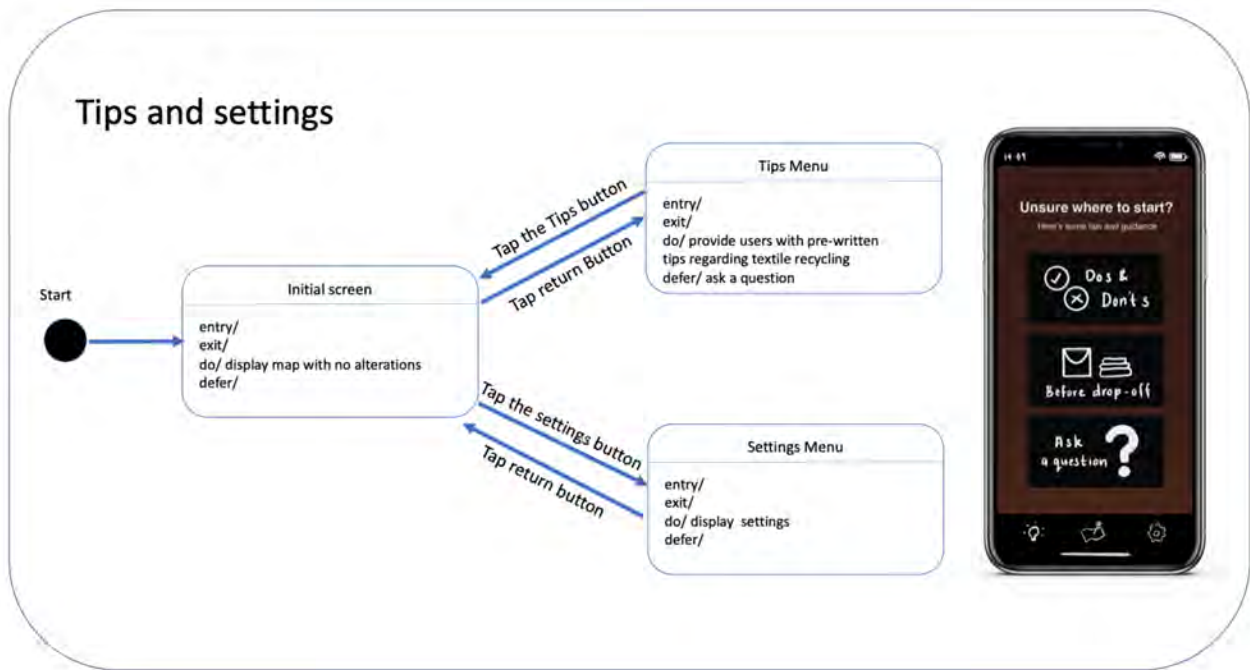


Figure 4: Figure of the " Tips and Settings" State.

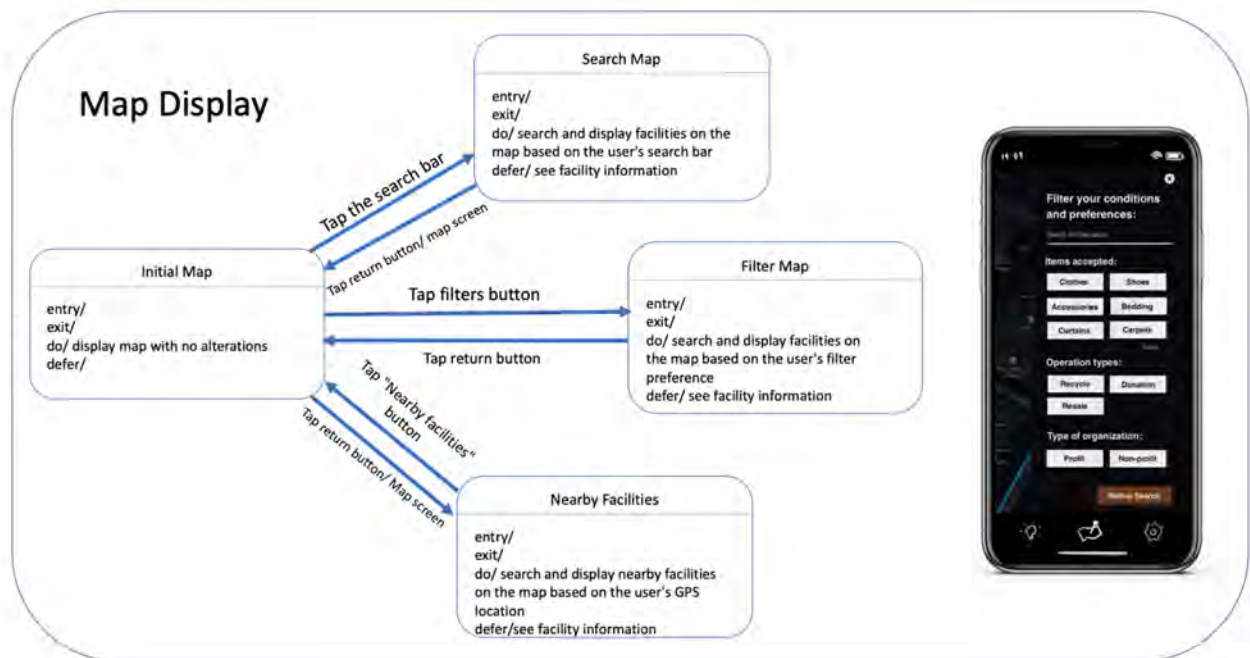


Figure 5: Figure of the "Map Display" State.

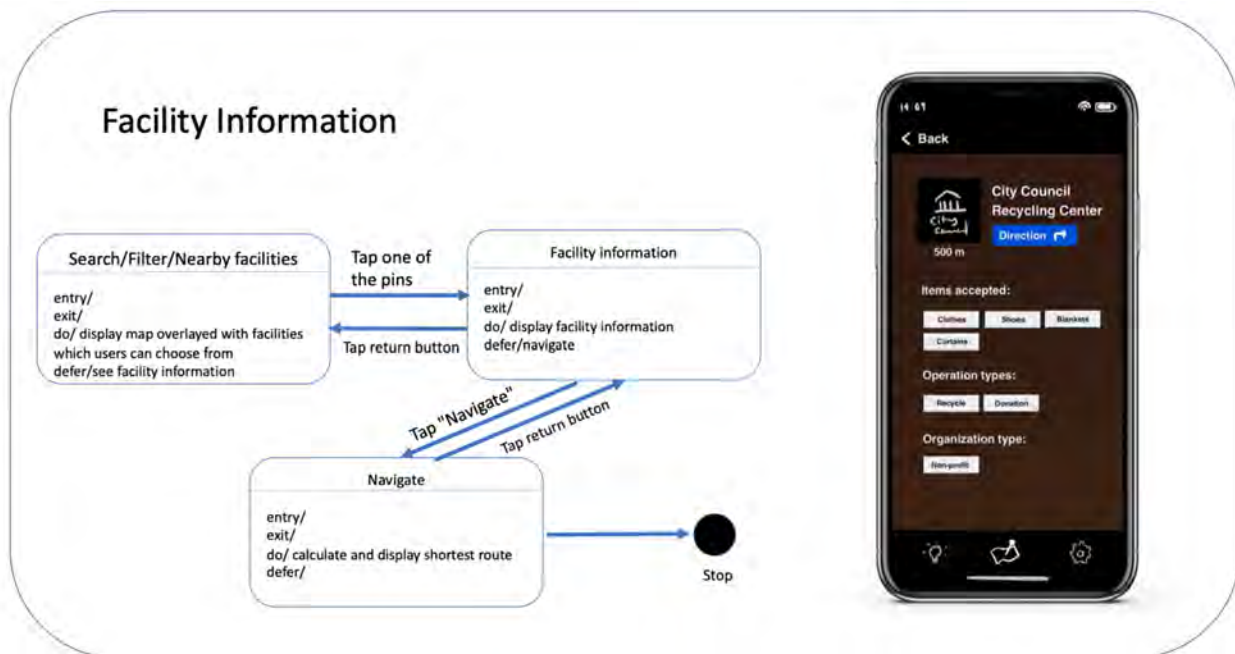


Figure 6: Figure of the "Facility Information" State.

### 3.2 Use Case Diagram

*Responsible Member: Ian Suzuki*

This is our Use Case Diagram for our application. This diagram shows what the user will do in the application, as well as show what the facilities will provide for the application. In this diagram, the user can search for any nearby facility that allows for textile drop-offs, as well as search for that facility's location on the map that is given by the app. Next, they will be able to filter out any facilities that satisfy the user's needs. For example, they will be able to filter if they want a facility that allows the recycling of shoes. They can use the filter and the app will display facilities that allow that kind of service. Furthermore, they will be able to view any information that is given from the facility through the app. Opening and closing times, types of textiles, and addresses will all be displayed through this option. Lastly, the user can also open the tips and tricks on how to recycle or get rid of unwanted textiles if they need any help in doing so. Over on the facility side, as mentioned previously, they can display any information for the user that they are willing to give out through the application. They are responsible for providing the data for our application so that we can display that on our app.



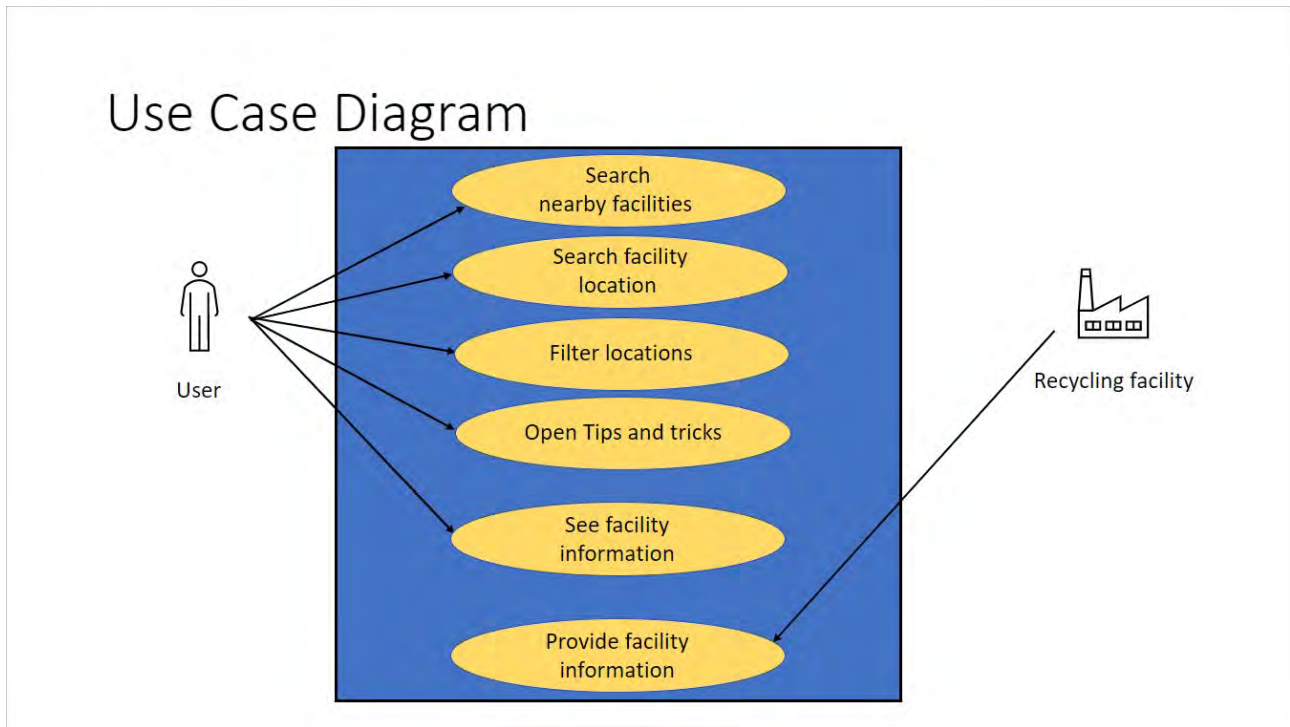


Figure 7: Figure of the Website.

### 3.3 Requirements

*Responsible Member: Ian Suzuki*

When planning on our “DrawerDrop” application, we wanted to include several requirements for our prototype. The requirements were to show what our application hopes to achieve in the final product and to set up a baseline for our application. First, we wanted to make the app run on iOS specifically. Instead of trying to make our prototype run both on Android and iOS, we wanted to help minimize that workload by focusing on one operating system. This is to allow us to work on the other requirements without having to waste time on trying to get the app to work on Android as well. Second, we wanted to locate textile drop-off points in New York only. Instead of gathering data all over the world, or in a domestic area where said data is not easily found, we decided to go for a location that has the data we need and is easy to implement. Third, we implemented a way to gather information on facilities that were displayed on the map. This is to help users know what facility they want to go to and see any other information on the facility that they might want to know. Lastly, we also wanted to implement a user review system. Unfortunately, because of time restraints, we were not able to implement such a feature. If we had more time available, it would have been possible to implement; however, due to the lack of experience our members had, it was difficult to implement some of the features listed here making it difficult to implement our user review system.

### 3.4 Weekly Reports

*Responsible Member: Stanley Jusuf*

For the first two weeks, the team discussed ideas for a smart city app as well as distribute roles for each of team member. Several ideas were brought up, however, the team decided on a textile drop-off application that will eventually be called “DrawerDrop” due to its unique



functionality as a smart city app, as well as seemingly doable given the team's experience app development. For the third week, the team prepares requirement elicitation to be sent to potential customers. Response from potential customers was taken into consideration when making the application. Next, the team did literature review for topic relating to the application (such as environmental impact of used clothing, second-hand clothing market, economics of used clothes, etc). For the fifth week, the team started working on the technical side of the project. To get started, the team created the storyboard, use-case diagram, system flowchart. The following week, the team is split into two groups, one in charge of the frontend, and the other for the backend. During that week, the team also created the state diagram. After that, the following weeks, the team worked on the application. For the last two weeks, the team finalized the application as well as made preparation for the final presentation.

### **3.5 Kanban**

*Responsible Member: Noraishah Mohd Rapi*

We utilized the Kanban board feature provided by WordPress as our agile project management tool. The board had allowed us to keep track of each of the tasks and the member in charge in detailed. By setting the estimated duration and deadline, we were able to increase our efficiency and managed the unprecedented problems we encountered systematically. As a result, we had successfully implemented about 75 percent of our minimum requirement by the sixth week of working on the project, even though there were several technical problems that took longer to fix.

### **3.6 Wordpress Website**

*Responsible Member: Noraishah Mohd Rapi*

We had established a WordPress website as a tool to communicate our goal, plan, and progress of our project. On the Homepage, we provided a future download link of our app and information about the background problem we are trying to solve, as well as the plan of our app, such as its functions and features. To know more about our project, one can access our Documentation page where we lay out the detailed process of our product, from requirements elicitation to technical diagrams. Next, we also added a Literature Review section where one can look into the key topics that we had researched for the development of the app. Lastly, information about the crews and our contact details can be found on Our Team page.

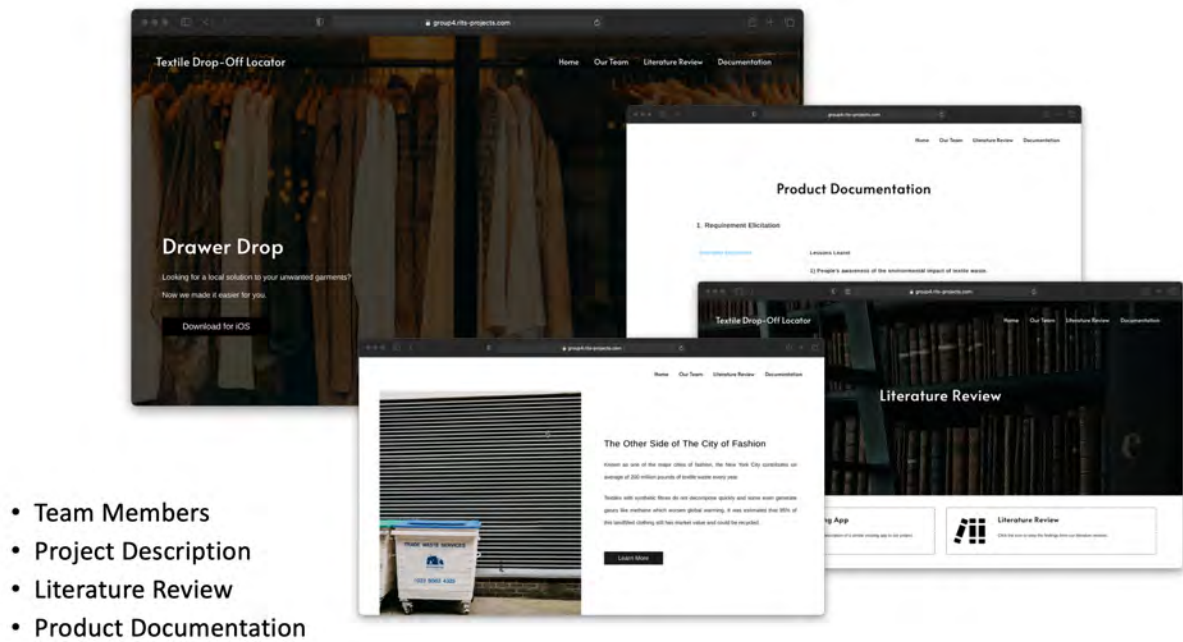


Figure 8: Figure of the Website.

### 3.7 Storyboard

*Responsible Member: Noraishah Mohd Rapi*

Aligned with the state diagram, we then made a storyboard to visualize the look of our app and how it navigates. I first roughly sketched the storyboard on Procreate and after presenting my idea to the team, I made a proper UI design that is presentable to everyone. Here are the six main screens from the storyboard with its respective description.

<b>Launch screen</b>	<b>Homepage: search nearby locations</b>	<b>Homepage: filter conditions and preferences</b>
		
<p>Welcome to Drawer Drop: This is what user see when they start the app.</p>	<p>This is the homepage, where user can proceed to search for nearby locations.</p>	<p>The homepage also provides options for user to filter their conditions and preferences.</p>
<b>Homepage: search and filtered results</b>	<b>Homepage: details of drop-off point selected</b>	<b>Tips and guidance page</b>
		
<p>The textile drop-off points will be shown based on nearest location and the preferences as customized.</p>	<p>User can click the facility's icon to view the details including its navigation.</p>	<p>User can also check out the tips page for some tips, guidance or ask questions before dropping-off their textiles.</p>

Figure 9: Figure of the Storyboard.

## 4 Analysis and Discussion (Technical Aspects)

### 4.1 Frontend

*Responsible Members: Rikuto Momoi and Yasith Dhamsara Bandaranayake*

### 4.2 The structure of our application

The initial Screen of Drawer Drop will be running for 2 seconds. This is the screen that will pop when the user opens up the application. After the 2 seconds, it will navigate into the next display. This is the Home Screen of our application. This screen is used to show the user's current location. Even though we weren't able to implement the User GPS, with more work it can be implemented. This Display provides the user with several options. Home, Filter, and Settings are the options that are available to the User. Filter option provides the user with filtering options which will help the user to narrow down the options and to find a drop-off point. When the user selects the filter button, it will display the pins of the locations near the user. Furthermore, A button is displayed on the top right-hand corner that can be used to obtain information on the facility. This will provide the vendor name, address, types of Items collected to the user. Settings is another option that a user can select. It will provide the user with the information that they need to consider when they are preparing their items for donation. We also included a Q and A section where the user can inquire about the problems that they have. The user can always come back to the main screen by pressing the home button.

### 4.3 How we display Map and location pins

To give our application a sense of place with map and locations information, we used Mapkit. MapKit is a neat API, comes with the iOS SDK, that allows us to display maps, navigation through maps, add annotations for specific locations. Hence firstly, we define latitude and longitude of New York City by using CLLocationCoordinate2D struct. The span value is made relatively small, so a big portion of New York is visible. The MKCoordinateRegion method defines the visible region. With the Map view the set region is displayed. Lastly, to display the annotations (map pins), we had to read locations data (latitude and longitude) from local Joson file given by backend team. The code for these executions can be seen below.

```

struct Model: Codable, Identifiable{
    var id: Int
    var Address: String
    var Items_Accepted: String
    var Vendor_Name: String
}

class ContentViewModel: ObservableObject {
    @Published var items = [Model]()
    func fetchData() {
        let api = "http://18.176.162.126/datainfo/shoes_list"
        guard let url = URL(string: api) else { return }
        URLSession.shared.dataTask(with: url) { (data, response, error) in
            do {
                if let data = data {
                    let result = try JSONDecoder().decode([Model].self, from: data)
                    DispatchQueue.main.async {
                        self.items = result
                    }
                } else {

```

Figure 10: Figure of the code to display the Map.

```

import SwiftUI
import MapKit
import Foundation
import SwiftyJSON

struct MapView1: View{
    @State var selected = ""
    @State var show = false
    @State var text: String = ""
    @State private var locations1: [Location1] = []
    @State private var coordinateRegion = MKCoordinateRegion(
        center: CLLocationCoordinate2D(latitude: 40.58115, longitude: -73.96376),
        span: MKCoordinateSpan(latitudeDelta: 70, longitudeDelta: 70)
    )
    var body : some View{
        NavigationView{
            ZStack{
                VStack{

                    Map(
                        coordinateRegion: $coordinateRegion,
                        annotationItems: locations1
                    ) { locations1 in
                        MapAnnotation(
                            coordinate: CLLocationCoordinate2D(
                                latitude: locations1.latitude,
                                longitude: locations1.longitude
                            )
                        ) {
                            VStack {
                                Text(locations1.Items_Accepted) .font(.caption2).bold()

```

Figure 11: Figure of the code to display the location pins.

## 4.4 How we implement the Fast Api

To fetch the list of date given by backend team, I created a simple struct Model that conforms to the Codable and Identifiable protocols. Codable is used for decoding and encoding the JSON data we get from our API call. Identifiable is used to help us make a unique identifier for our Model object so our app can keep track of it. We added date types Id, Address, Items Accepted, Vendor Name, which we are going to get from our Fast Api. We create ContentViewModel class that conforms to ObservableObject protocol. It has an array of Model which is empty. Inside of the class, we have fetchData function. which reads the url and gets data. Then we generate a URL class from a URL string and use it to make a URL request. Since the URL class returns null if the string specified in the argument is not a valid URL, by using the guard statement, we coded for the handling in case of error. The URLSession.shared.dataTask is the iOS class responsible for managing network requests. Next, it checks if the data could be retrieved from the server. "if let" is judging null of the target property and executing the process at the same time DispatchQueue is an object that manages the execution of tasks on our application. It has a stored property called result, which is initialized with a new, empty Model array. This is how I fetch data from FastApi. There is the code for these shown below.

```
HStack{
    Button(action: { withAnimation{ coordinateRegion.span = MKCoordinateSpan(latitudeDelta: 0.3, longitudeDelta: 0.3)},
        label:{ Text(" Your Location").bold().frame(width: 250, height: 50, alignment:
        .center).background(Color.blue).cornerRadius(8).foregroundColor(.white)
        } }
    ) {
        self.show.toggle()
    } {
        Text("Open").padding(.vertical).padding(.horizontal,25).foregroundColor(.white)
    }
    .background(LinearGradient(gradient: .init(colors: [Color("Color"),Color("Color1")]), startPoint: .leading, endPoint:
    .trailing))
    .clipShape(Capsule())
}
}
}
}
VStack{
    Spacer()
    RadioButtons1().offset(y: self.show ? (UIApplication.shared.windows.last?.safeAreaInsets.bottom)! + 15 : UIScreen.main.bounds.height)
    }.background(Color(UIColor.label.withAlphaComponent(self.show ? 0.2 : 0)).edgesIgnoringSafeArea(.all))
    }.background(Color("Color2").edgesIgnoringSafeArea(.all))
    .animation(.default)
}
}
}
private func readFile() {
    if let url = Bundle.main.url(forResource: "nonprofit_list", withExtension: "json"),
    let data = try? Data(contentsOf: url) {
        let decoder = JSONDecoder()
        if let jsonData1 = try? decoder.decode(JSONData1.self, from: data) {
            self.locations1 = jsonData1.locations1
        }
    }
}
```

Figure 12: Figure of the code to implement the Fast Api.

## 4.5 Backend

*Responsible Members: Rifqi Ihsan Nabil*

It is common for applications to have a backend system to connect the app to other things such as the internet or to store the data that are required to show inside the app. Same just like any other app, Drawer drop application also has a back-end that has a function to store the data of the pinpoint such as the id, longitude and latitude of the pinpoint, name of the venue and other information and relay it to the front-end using a webserver so that the data can

be shown to the user. This section will be divided into how the back-end works with another subsection of which API and tools are used for the app.

## 4.6 How it works

First, Drawer Drop use excel data as the raw data that are provided online on the internet. This data contains around 1100 pinpoint locations in New York that consist of many useful information. Then this excel data are going to be converted to .CSV format so that the data can be imported to SQLite database. After the data has been imported to the SQLite database for our app, then we have a python script to run the database which makes the database can be convertible from human readable format to computer readable format (for our app we, we convert it to JSON format). The python script are also used to make a filter or make it as a query search so that the user can find the location data based on the order that they want. All the SQLite database and the python script are put in cloud. For the cloud, we use Amazon Web service (AWS) using elastic compute cloud (EC2) tools. This cloud provides the space for the database and the python script to be able to run online and be send to the front-end based on the information that the user request. To display the result of the python script to the URL, we use an API called the FAST API so that the program that are running through the AWS cloud can display the JSON The picture of the steps of how the back-end of our app work can be looked at the figure below.

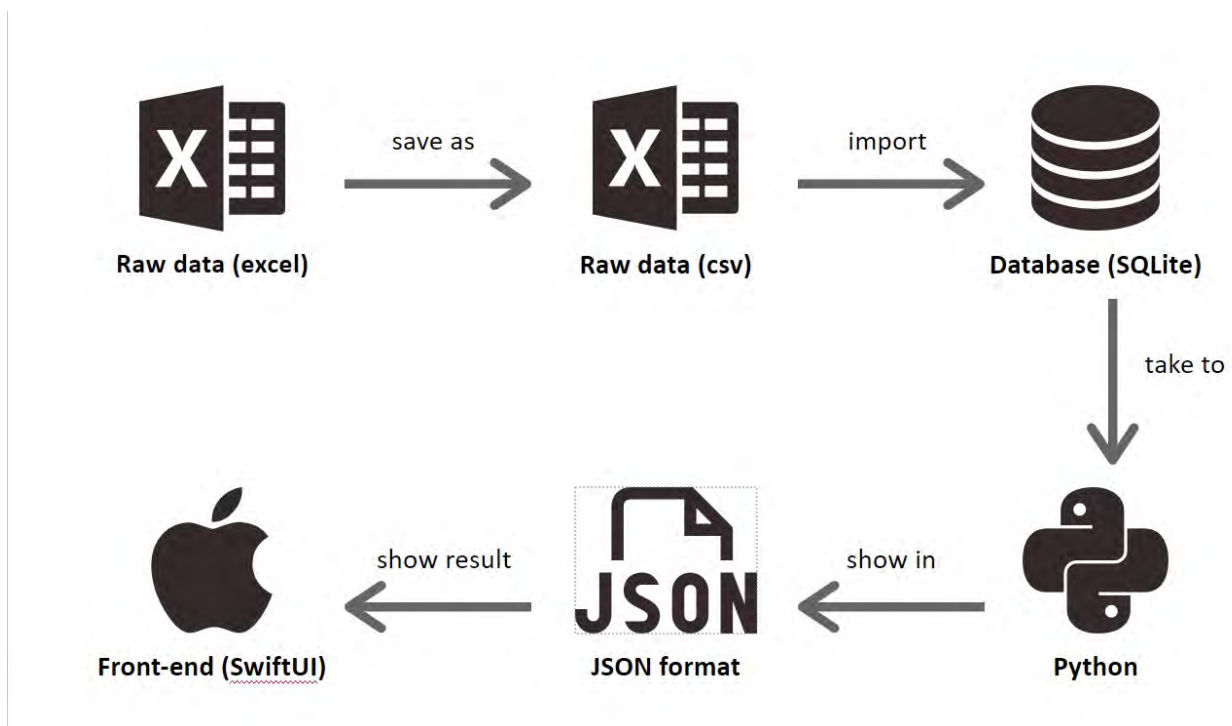


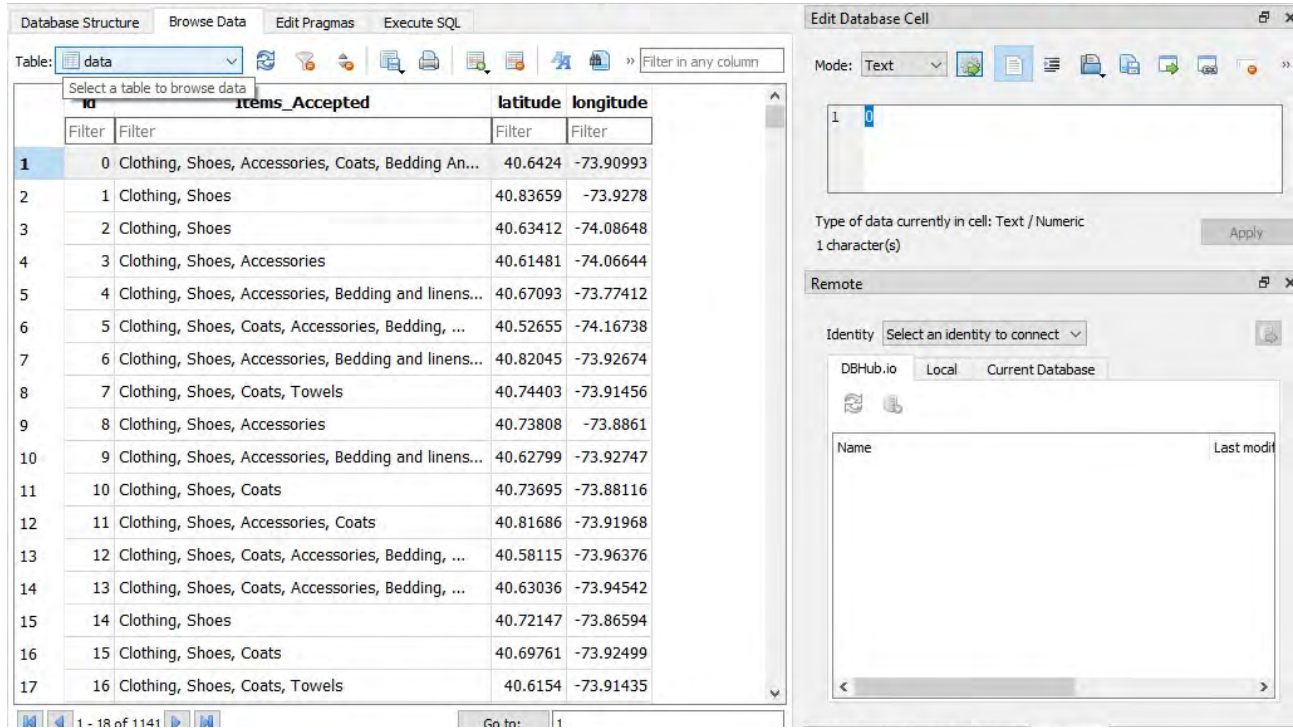
Figure 13: How it Works.

## 4.7 SQLite database

Our application use SQLite database as this database is not heavy. This database is also easy to learn as it is beginner friendly and can controlled using python script. For importing and



managing the data in SQLite database, we use an external software called DB Browser(SQLite). This software helps us importing csv data to the SQLite, renaming column titles, look the insides of the database and several function that makes the database readable to us. The data that can be read by the DB Browser to the SQLite can be seen on the figure below.



id	items_Accepted	latitude	longitude
1	0 Clothing, Shoes, Accessories, Coats, Bedding An...	40.6424	-73.90993
2	1 Clothing, Shoes	40.83659	-73.9278
3	2 Clothing, Shoes	40.63412	-74.08648
4	3 Clothing, Shoes, Accessories	40.61481	-74.06644
5	4 Clothing, Shoes, Accessories, Bedding and linens...	40.67093	-73.77412
6	5 Clothing, Shoes, Coats, Accessories, Bedding, ...	40.52655	-74.16738
7	6 Clothing, Shoes, Accessories, Bedding and linens...	40.82045	-73.92674
8	7 Clothing, Shoes, Coats, Towels	40.74403	-73.91456
9	8 Clothing, Shoes, Accessories	40.73808	-73.8861
10	9 Clothing, Shoes, Accessories, Bedding and linens...	40.62799	-73.92747
11	10 Clothing, Shoes, Coats	40.73695	-73.88116
12	11 Clothing, Shoes, Accessories, Coats	40.81686	-73.91968
13	12 Clothing, Shoes, Coats, Accessories, Bedding, ...	40.58115	-73.96376
14	13 Clothing, Shoes, Coats, Accessories, Bedding, ...	40.63036	-73.94542
15	14 Clothing, Shoes	40.72147	-73.86594
16	15 Clothing, Shoes, Coats	40.69761	-73.92499
17	16 Clothing, Shoes, Coats, Towels	40.6154	-73.91435

Figure 14: Figure of the SQLite database.

## 4.8 Python Script

In terms of the code, we need our app to be able to fetch the data automatically based on what the user needs based on the filter and the location that is shown on the front-end of the app. From that purpose, this is where the python script comes in handy. On the coding part, there are 2 parts of scripts that are written for the backend, which is the main.py code and the requirements.txt. The requirements.txt is used to run the maincode.py in the AWS EC2 cloud. The maincode.py import some of external resource using the pip such as sqlite3, FAST API, json, Uvicorn(for the server) and other that can be seen on the requirements.txt figure below.

```
1  asgiref==3.4.1
2  click==8.0.1
3  colorama==0.4.4
4  fastapi==0.66.0
5  gunicorn==20.1.0
6  h11==0.12.0
7  pydantic==1.8.2
8  starlette==0.14.2
9  typing-extensions==3.10.0.0
10 uvicorn==0.14.0
```

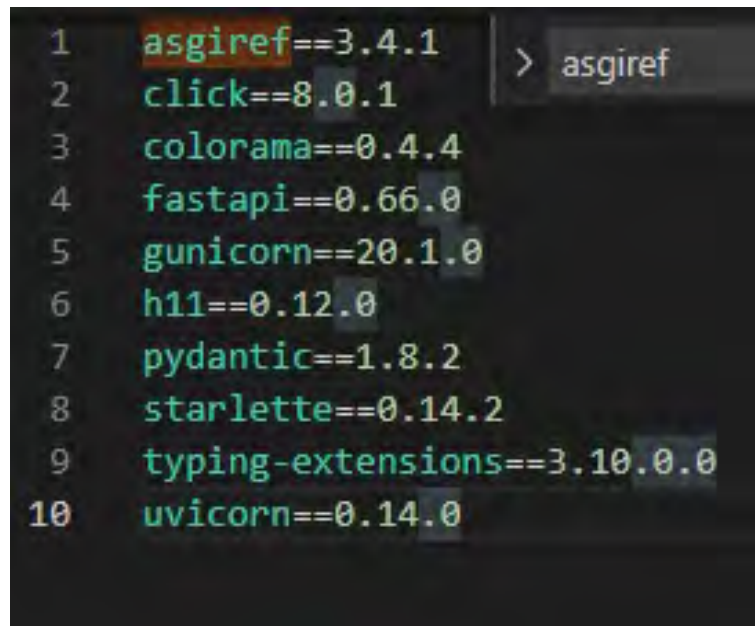


Figure 15: Figure of the imported resources.

The maincode.py is a script that has a function to convert the data from the SQLite database to a JSON web format so that the front-end part can use the data based on their needs. The code for the main.py can be seen on the figure below.

```

@app.get("/search/{query}")
def searchitems( query: str ,json_str = False ):
    conn = sqlite3.connect( DB )
    conn.row_factory = sqlite3.Row # This enables column access by name: row['column_name']
    db = conn.cursor()
    sqlquery = "SELECT * FROM data WHERE Items_Accepted LIKE '{0}'".format(query)
    #sqlquery = "SELECT * FROM data WHERE Items_Accepted LIKE '%"+ query +"%"
    rows = db.execute(sqlquery).fetchmany(10)

    conn.commit()
    conn.close()

    if json_str:
        #return json.dumps( [dict(ix) for ix in rows] ) #CREATE JSON
        return json.dumps( {'variable' : query} ) #CREATE JSON
    return rows

```

Figure 16: Figure of the code for main.py.

From figure 15, it shows that the script will connect to the database and find based on the column(in this case we use the Items accepted column) and translate it to JSON string. The result for the script will be shown to the web-URL with the help of FAST API and AWS EC2. The full code for the main.py can be seen on the figure below.

```

1  import sqlite3
2  import json
3  from typing import Optional
4  from fastapi import FastAPI
5
6  DB = "pbl_projectdata.db"
7  app = FastAPI()
8
9  @app.get("/")
10 def getName():
11     return {"Name": "Rifqiiii"}
12
13 @app.get("/clothing_list")
14 def Clothing_list( json_str = False ):
15     conn = sqlite3.connect( DB )
16     conn.row_factory = sqlite3.Row # This enables column access by name: row['column_name']
17     db = conn.cursor()
18
19     rows = db.execute('''
20     SELECT * FROM data WHERE Items_Accepted LIKE '%Clothing%'
21     ''').fetchmany(10)
22
23     conn.commit()
24     conn.close()
25
26     if json_str:
27         return json.dumps( [dict(ix) for ix in rows] ) #CREATE JSON
28     return rows
29
30 @app.get("/shoes_list")
31 def Shoes_list( json_str = False ):
32     conn = sqlite3.connect( DB )
33     conn.row_factory = sqlite3.Row # This enables column access by name: row['column_name']
34     db = conn.cursor()
35
36     rows = db.execute('''
37     SELECT * FROM data WHERE Items_Accepted LIKE '%Shoes%'
38     ''').fetchmany(10)
39

```

Figure 17: Figure of the code- part 1.

```

40     conn.commit()
41     conn.close()
42
43     if json_str:
44         return json.dumps( [dict(ix) for ix in rows] ) #CREATE JSON
45     return rows
46
47
48 @app.get("/coat_list")
49 def Coats_list( json_str = False ):
50     conn = sqlite3.connect( DB )
51     conn.row_factory = sqlite3.Row # This enables column access by name: row['column_name']
52     db = conn.cursor()
53
54     rows = db.execute('''
55     SELECT * FROM data WHERE Items_Accepted LIKE '%Coats%'
56     ''').fetchmany(10)
57
58     conn.commit()
59     conn.close()
60     (parameter) json_str: Any
61     if json_str:
62         return json.dumps( [dict(ix) for ix in rows] ) #CREATE JSON
63     return rows
64
65 @app.get("/towel_list")
66 def Towel_list( json_str = False ):
67     conn = sqlite3.connect( DB )
68     conn.row_factory = sqlite3.Row # This enables column access by name: row['column_name']
69     db = conn.cursor()
70
71     rows = db.execute('''
72     SELECT * FROM data WHERE Items_Accepted LIKE '%Towels%'
73     ''').fetchmany(10)
74
75     conn.commit()
76     conn.close()

```

Figure 18: Figure of the code- part 2.

```

76     conn.close()
77
78     if json_str:
79         return json.dumps( [dict(ix) for ix in rows] ) #CREATE JSON
80     return rows
81
82 @app.get("/nonprofit_list")
83 def NonProfit_list( json_str = False ):
84     conn = sqlite3.connect( DB )
85     conn.row_factory = sqlite3.Row # This enables column access by name: row['column_name']
86     db = conn.cursor()
87
88     rows = db.execute('''
89     SELECT * FROM profitcsv WHERE Nonprofit_Organization ='True'
90     ''').fetchmany(10)
91
92     conn.commit()
93     conn.close()
94
95     if json_str:
96         return json.dumps( [dict(ix) for ix in rows] ) #CREATE JSON
97     return rows
98
99 @app.get("/profits_list")
100 def profits_list( json_str = False ):
101     conn = sqlite3.connect( DB )
102     conn.row_factory = sqlite3.Row # This enables column access by name: row['column_name']
103     db = conn.cursor()
104
105     rows = db.execute('''
106     SELECT * FROM profitcsv WHERE Nonprofit_Organization ='False'
107     ''').fetchmany(10)
108
109     conn.commit()
110     conn.close()
111
112     if json_str:
113         return json.dumps( [dict(ix) for ix in rows] ) #CREATE JSON

```

Figure 19: Figure of the code- part 3.

```

113         return json.dumps( [dict(ix) for ix in rows] ) #CREATE JSON
114     return rows
115
116 @app.get("/search/{query}")
117 def searchitems( query: str ,json_str = False ):
118     conn = sqlite3.connect( DB )
119     conn.row_factory = sqlite3.Row # This enables column access by name: row['column_name']
120     db = conn.cursor()
121     sqlquery = "SELECT * FROM data WHERE Items_Accepted LIKE '%{0}%'.format(query)
122     #sqlquery = "SELECT * FROM data WHERE Items_Accepted LIKE '%" + query +"%"
123
124     rows = db.execute(sqlquery).fetchmany(10)
125
126     conn.commit()
127     conn.close()
128
129     if json_str:
130         #return json.dumps( [dict(ix) for ix in rows] ) #CREATE JSON
131         return json.dumps( {'variable' : query} ) #CREATE JSON
132     return rows
133
134
135 @app.get("/datainfo/clothing_list")
136 def Clothing_list( json_str = False ):
137     conn = sqlite3.connect( DB )
138     conn.row_factory = sqlite3.Row # This enables column access by name: row['column_name']
139     db = conn.cursor()
140
141     rows = db.execute('''
142     SELECT * FROM datainfo WHERE Items_Accepted LIKE '%Clothing%'
143     ''').fetchmany(10)

```

Figure 20: Figure of the code- part 4.



```

149         return json.dumps( [dict(ix) for ix in rows] ) #CREATE JSON
150     return rows
151
152 @app.get("/datainfo/clothing_list")
153 def Clothinfo_list( json_str = False ):
154     conn = sqlite3.connect( DB )
155     conn.row_factory = sqlite3.Row # This enables column access by name: row['column_name']
156     db = conn.cursor()
157
158     rows = db.execute('''
159     SELECT * FROM datainfo WHERE Items_Accepted LIKE '%Clothing%'
160     ''').fetchmany(10)
161
162     conn.commit()
163     conn.close()
164
165     if json_str:
166         return json.dumps( [dict(ix) for ix in rows] ) #CREATE JSON
167     return rows
168
169 @app.get("/datainfo/shoes_list")
170 def shoesinfo_list( json_str = False ):
171     conn = sqlite3.connect( DB )
172     conn.row_factory = sqlite3.Row # This enables column access by name: row['column_name']
173     db = conn.cursor()
174
175     rows = db.execute('''
176     SELECT * FROM datainfo WHERE Items_Accepted LIKE '%Shoes%'
177     ''').fetchmany(10)
178
179     conn.commit()
180     conn.close()
181
182     if json_str:
183         return json.dumps( [dict(ix) for ix in rows] ) #CREATE JSON
184     return rows

```

Figure 21: Figure of the code- part 5.

```

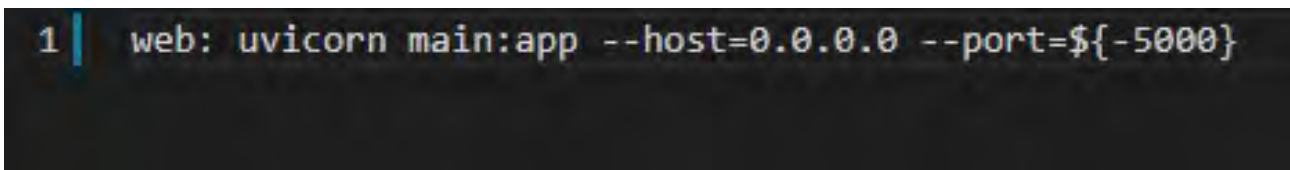
184     return rows
185
186 @app.get("/datainfo/coat_list")
187 def Coatsinfo_list( json_str = False ):
188     conn = sqlite3.connect( DB )
189     conn.row_factory = sqlite3.Row # This enables column access by name: row['column_name']
190     db = conn.cursor()
191
192     rows = db.execute('''
193     SELECT * FROM datainfo WHERE Items_Accepted LIKE '%Coats%'
194     ''').fetchmany(10)
195
196     conn.commit()
197     conn.close()
198
199     if json_str:
200         return json.dumps( [dict(ix) for ix in rows] ) #CREATE JSON
201     return rows
202
203 @app.get("/datainfo/towel_list")
204 def Towelinfo_list( json_str = False ):
205     conn = sqlite3.connect( DB )
206     conn.row_factory = sqlite3.Row # This enables column access by name: row['column_name']
207     db = conn.cursor()
208
209     rows = db.execute('''
210     SELECT * FROM datainfo WHERE Items_Accepted LIKE '%Towels%'
211     ''').fetchmany(10)
212
213     conn.commit()
214     conn.close()
215

```

Figure 22: Figure of the code- part 6.

## 4.9 FAST API

Fast API holds an important role for our application. The purpose of having the FAST API is to show the result of the python script to the website. In this case we want to show the result in json format so that the front end can fetch using the URL API as well. To run the fast API, we used the Uvicorn command which are shown below.

A terminal window with a dark background. A blue vertical bar highlights the first character '1' of the command. The command is: `web: uvicorn main:app --host=0.0.0.0 --port=${-5000}`

```
1 | web: uvicorn main:app --host=0.0.0.0 --port=${-5000}
```

Figure 23: Figure of the SQLite database.

Running the Uvicorn command will make a new port that are going to be shown in the URL.

## 4.10 AWS EC2

AWS EC2 is used to put the python script and database online so it can be accessed by the front-end anywhere. EC2 can also be called as an online computer as we connect the computer remotely using the command line interface(CLI) SSH. The reason we are using the EC2 is because SQLite database is a local database that actually can't be accessed online like postgresql or mysql. If the python script run in a normal webserver(for example heroku) then it will produce an error as SQLite is not supported to be an online database. For this purpose we use EC2 so that it can be accessed and the reason that it is possible is because EC2 is an online computer. Which the result of the computer can be accessed online through web browser. Instruction ofm how to deploy the EC2 can be seen on the figure below.



1. Copy the fastapi.pem
2. Open cmd in same folder
3. Enter this command to ssh into the ec2 instance: `ssh -i "fastapi.pem" ubuntu@ec2-18-176-162-126.ap-northeast-1.compute.amazonaws.com`
4. Go into fast api folder using this command: `cd fastapi`
5. Now git pull the new changes
6. Run this command: `gunicorn main:app -w 4 -k uvicorn.workers.UvicornWorker -b 0.0.0.0:8000`
7. After running this your app will be live on the url

Figure 24: Figure of How to deploy the EC2.

## 4.11 Other Tools

There are several tools and API's that we planned to use such for this app, but the reason it's not part of the procedure is because some of them cause an error that are difficult to be fixed or the tools or API's are not compatible with the database. Several tools and API's that we're try to use are deta API, postgresql and Heroku

## 5 Conclusion

*Responsible Member: Stanley Jusuf and Rifqi Ihsan Nabil*

With this application, our main goal is to encourage people to recycle clothing materials instead of directly disposing as well as introducing people to cloth recycling facilities. We had hoped that our application would allow our audience to find clothing recycling facilities in a convenient manner. However, even creating a simple app to find clothing recycling facility has been proven to be challenging. Even in our final build, we have much to improve. For instance, our application is still unable to calculate the distance between the origin and the destination, the filter and the search function only work conditionally, the user GPS search is not functional, the GPS pins are still unclickable, still mostly using local database, and did not have the chance to improve the settings and tips display due to time constraint. Regardless, this project has been an important learning experience for every member of our team. We get to experience numerous first learning experiences as well as form a social bond. Our only regret is not being able to finish the application in a way that would satisfy all of us. Given the chance, surely, we would be delighted to continue improving DrawerDrop and maybe add new features as well.

## **6 Individual Reports**

### **6.1 Report: Yasith Dhamsara Bandaranayake**

First and foremost I would like to thank all the team members as it was a pleasure to work with them all. This project was the first project that I was able to use Swift. There were several times where I wasn't able to come up with what I wanted to do. However, working in the frontend and helping with the coding was a new experience and I was able to learn a lot. Furthermore, I worked as the Literature reviewer of the team. It was certainly a new role. I went through a number of research papers and the hardest part was to summarize every single one of them. Nonetheless, I enjoyed every single bit of it and I also believe it will be important in my future activities. I might not have perfected it but it was certainly good practice for me. Furthermore, as a result of this project, I'm interested in learning more about designing applications and also more interested in coding. This project was certainly very challenging. With the amount of time that we had for this project, As a group, we were able to achieve many of our targets and I'm grateful that I was a part of it.

### **6.2 Report: Noraishah Mohd Rapi**

Throughout this project, I was able to learn a lot about team management and work first-hand on the intricate process of app development. One of the most prominent skills I harnessed was website-making. When I was assigned as the team's webmaster, I did not have any experience of using WordPress prior to this project. After spending quite some time on tutorials and trials and errors, I could finally get a good grasp to utilize the WordPress website. With that, I was able to assemble the visual hierarchy with my design idea to make the website convenient to navigate throughout the content. In addition, as an assistant front-end coder, I had the chance to work on Swift for the first time when implementing the storyboard design on the prototype. Of course, I was able to familiarize myself with this new programming language and this experience had also sparked my interest to develop IOS applications.

### **6.3 Report: Rifqi Ihsan Nabil**

I am grateful that I have the chance to work to make an IOS web app for 7 weeks with this group(group4). There are many experience that I receive from this project, especially the ability to code as a backend. During my first two weeks of the project, being assigned the senior back-end role give me the anxiety due to my lack of experience about the field and people opinions that coding as a back-end tends to be difficult. Since everyone believed that I was able to do it, I started to take the responsibility and take a leap to this new field. It is true that there are many errors occurred during the process especially when setting the environment as it took 2 or 3 days to solve 1 problem. As time goes by I am starting to become more interested to the role and it becomes natural to search the solutions individually on the internet and be creative with it. If the problems really that difficult and I really do not know the direction I tried to ask the TA to give a clue of guidance to solve it. Once again I am blessed to be given the opportunity to work this project with team 4.

### **6.4 Report: Stanley Jusuf**

Being a project manager has been a humbling experience. Realizing the amount of work that needs to be done with the hand you have been dealt with has been an eye-opening experience for me. As project manager, I have learned several leadership and team management skills. I realized that, to have a project, a grounded vision of an idea is essential to not only meet

customer demand, but also to manage expectations. Regarding expectations, managing expectations is also essential when planning. It also goes without saying that I had learned a great deal about project flow, task delegation, time management, and more. In addition, I've also learned some technical skills such as coding, creating charts and diagrams, implementing an API, and more. Moving forward, I appreciate the experience this project has given me.

## 6.5 Report: Rikuto Momoi

It was a great opportunity to make an IOS application for several weeks. The most grateful things I archived was creating the font-end codes. I was in charge of senior font-end coder, so I have been coding for 7 weeks. There were many things I leaned and improved for the 7 weeks, because it was my first time to do coding. Looking back a year ago, I didn't even know how to use computer, but I could accomplish my tasks this time. My codes were a little bit messy, so I will learn how to code cleaner. Also, I leaned team work and professional speaking through software engineering class. Although my presentation skill is still not good, I believe that I will be able to use this software engineer presentation skills in the future. I'd like to appreciate to my team member and professors who gave me this experience.

## 6.6 Report: Ian Suzuki

This project was a great experience and allowed me to work with a team in a working-esque environment. I was able to learn many great things about developing an application, as well as coding programs such as SwiftUI and XCode. Furthermore, I was able to learn a lot about team dynamics and how to work with others when developing a project. As I was part of the full stack team, I was able to experience bits of each side of the development. I was able to grasp a little bit on the UI and front-end portion of our app and was able to see the back-end also. I was also responsible in making charts and diagrams for our project with the help of other team members. It was a unique experience developing and creating these charts and see what goes through the minds of actual developers when creating such applications. I learned that there is so much more in creating an application and with the help of others, it can be a very fun and humbling experience.

## References

- [1] L. Norris, "Trade and Transformations of Secondhand Clothing: Introduction" *TEXTILE*, Vol. 10, No. 2, pp. 128–143, 2012.
- [2] L. Farrant, S. Olsen, and A. Wangel, "Environmental benefits from reusing clothes" *The International Journal of Life Cycle Assessment*, Vol. 15, No. 7, pp. 726–736, 2010.
- [3] L.Hoon,R.Vasa, J.G Schneider, and J. Grundy, "An analysis of the mobile app review landscape trends and implications" *Faculty of Information and Communication Technologies, Swinburne University of Technology, Tech. Rep*, 2013.
- [4] B.Fu., J.Lin, L.Li.,C. Faloutsos, J.Hong and N.Sadeh, "Why people hate your app: Making sense of user feedback in a mobile app store" pp. 1276–1284 2013.
- [5] M. Grebosz-Krawczyk and D. Siuda, "Attitudes of Young European Consumers Toward Recycling Campaigns of Textile Companies" *Autex Research Journal*, Vol. 19 No. 4, pp. 394-399, 2019.

- [6] R.Pal, "EPR-SYSTEMS AND NEW BUSINESS MODELS FOR SUSTAINED VALUE CREATION: A STUDY OF SECOND-HAND CLOTHING NETWORKS IN SWEDEN" 2015.
- [7] H. van Heerde, I. Dinner and S. Neslin, "Engaging the unengaged customer: The value of a retailer mobile app" *International Journal of Research in Marketing*, Vol. 36, No. 3, pp. 420–438, 2019.
- [8] C. Chaitanya and D. Gupta, *2017 2nd IEEE International Conference on Recent Trends in Electronics*, "Factors influencing customer satisfaction with usage of shopping apps in India,"
- [9] C. Xu, D. Peak and V. Prybutok, "A customer value, satisfaction, and loyalty perspective of mobile application recommendations", *Decision Support Systems*, Vol. 79, pp. 171-183, 2015.
- [10] E. Paulicelli and H. Clark *The fabric of cultures*. London: Routledge, 2009.
- [11] I. Sandvik and W. Stubbs, "Circular fashion supply chain through textile-to-textile recycling", *Journal of Fashion Marketing and Management: An International Journal*, Vol. 23, No. 3, pp. 366–381, 2019.
- [12] *Exports of Nordic Used Textiles: Fate, benefits and impacts*. Nordic Council of Ministers, 2016.
- [13] S. Cuc and M. Vidovic, "Environmental Sustainability through Clothing Recycling", *Operations and Supply Chain Management: An International Journal*, pp. 108-115, 2014.
- [14] F. Esteve-Turrillas and M. de la Guardia, "Environmental impact of Recover cotton in textile industry", *Resources, Conservation and Recycling*, Vol. 116, pp. 107–115, 2017.
- [15] T. Toprak and P. Anis, "Textile Industry's Environmental Effects and Approaching Cleaner Production and Sustainability: an Overview", *Journal of Textile Engineering and Fashion Technology*, Vol. 2, No. 4, 2017.
- [16] Researchgate. <https://www.researchgate.net/publication/342625358-Harmful-effects-of-textile-waste>. Last accessed : July 2021.
- [17] K. Vehmas, A. Raudaskoski, P. Heikkilä, A. Harlin and A. Mensonen, "Consumer attitudes and communication in circular fashion", *Journal of Fashion Marketing and Management: An International Journal*, Vol. 22, No. 3, pp. 286-300, 2018.
- [18] S. Weber, J. Lynes and S. Young, "Fashion interest as a driver for consumer textile waste management: reuse, recycle or disposal", *International Journal of Consumer Studies*, Vol. 41, No. 2, pp. 207-215, 2016.