IOS Development Assignment 1 Report

Name: Hanjun Lee
Student Number: s3732878
Project Name: YumMap

Lecture: Tom Huynh

Table of Contents

a. Introduction

YumMap is an application that introduces my favorite restaurant in Ho Chi Minh City. It allows me to store and remember information about the restaurants I have been. I can check the location of the restaurant on the map and search for the restaurants by the name of them. Also, they are shown in alignment under certain conditions such as sorted by rate and sorted by alphabet.
The reason why I made this is that my parents made food for me in Korea before I came to Vietnam, and I didn't think much about what to eat today. However, as I lived alone in Vietnam, I thought a lot about what to eat every day, and naturally, while looking for delicious restaurants in Ho Chi Minh City, I have remembered them one by one. Fortunately, the topic of this IOS development assessment 1 is to introduce my favorite things, so I wanted to introduce my favorite restaurants that I have been while staying in Ho Chi Minh City.

b. Project Description

Short video: Entire flow of the program.mp4 in GIF Folder

When I first run the application, there's a logo, a slogan of application, and an arrow button at the bottom of the page that goes to the main menu. In addition, there is a button at the top left of the page that allows you to select a color theme for the application. It supports light mode and dark mode of this application. Finally, at the top right, there's a little I-shaped button, and user press it, it shows my information in the alert window.

If user come to the main menu, search field is located at the top, and if user search for the name of the restaurant, the matching results are displayed as a list below. User can tap the search results to go to the page where user can find short information about the restaurant. In the middle of the main menu, there are six buttons to organize restaurants into six types: Breakfast, Lunch, Dinner, Coffee, Pub, and All. If user tap the button, user can check the restaurants that fit each type in a list format.

When user presses a button of each type in the main menu, it shows the restaurants in a list format, where user can see the main image, name, and short description of the restaurants. User can only see three items at first, but if user want to see more items, user can scroll down to see more restaurants. Each element of the list is a button that goes to a page that provides detailed information about the restaurant. Also, at the top of the page is a sorted by button that lets user sorts the list of restaurants, where user can choose either Sorted by rate or Sorted by Alphabet to rearrange the list of restaurants in descending of the rate or alphabetical order according to their choose.

Finally, if user go to the restaurant detail page, user can see the location of the restaurant on the top with a red pin on the map. Below that, there's a window like popover window where user can check all the detail information about the restaurants. From the top, there are image, names, and short description of the restaurant and four buttons below them. Tap each button to view the restaurant's address, phone number,

and detailed description, and finally some pictures of the restaurant in a slide format. In addition, when you click on the empty space in this area, a window that shows information about the restaurant goes down and you can see the map with full screen. And if user press the Open Detail button underneath the map, user can see the map and restaurant information at the same time as initial state.

You can return to the previous section by pressing the blue back button at the top left of the entire flow.

c. Implementation Details

In order to store the information of each item in this program, I defined a structure called Item, and created an Item type array called items that collected all the information of each item.

```
// Declaring struct Item to store the information about the restaurant
struct Item: Identifiable, Codable{
    var id: UUID {
        UUID()
    }
    let name: String // name of restaurant
    let address: String // address of restaurant
    let type: String // type of address of restaurant(Breakfast, Lunch, Dinner, Coffee, Pub)
    let phone: String // Phone number of the restaurant
    let rate: Double // rating of the restaurant
    let imageName: String // the name of main image of the restaurant
    let information: String // Short description for the restaurant
    let subInfomation: String // additional type of restaurant (depending on restaurant)
    let imageArray: [String]
    let coordinates: Coordinates
    // Method
    var image: Image{ // method to inserting restaurant image
        Image(imageName)
    }

    var locationCoordinate : CLLocationCoordinate2D{
        CLLocationCoordinate2D(latitude: coordinates.latitude, longitude: coordinates.longitude)
    }
}
```

Figure 1: Item struct


```
{
    {
        "name": "Ganesh",
        "address": "54 Đường số 6, Tân Phong, Quận 7, Thành phố Hồ Chí Minh, Vietnam",
        "type": "Dinner",
        "phone": "+84 28 5410 1627",
        "rate": 4.2,
        "imageName": "Ganesh",
        "information": "In this restaurant, you can enjoy a variety of Indian curry with naan. The price is a little high, but you can have a
            decent dinner.",
        "subInfomation": "Indian Restaurant",
        "imageArray": ["Ganesh1", "Ganesh2", "Ganesh3"],

        "coordinates": {
            "latitude": 10.730389980231449,
            "longitude": 106.70781280341541
        }
    },
```


Figure 2: Part of items

<Main feature requirements>

1. The button to navigate to the main menu view
Short video: Navigation Button.mp4 in GIF Folder

NavigationLink was used to create the Navigation button for the first page of the application. NavigationLink goes to the View where the parameter destination was received.

```
// Navigation button to the main menu
NavigationLink(destination: NavigationListView(isDarkMode: $isDarkMode)){
    Circle() // Creating the Rounded Rectangle
        .fill(Color(isDarkMode ? "Color_Dark" : "SubColor"))
        .frame(width: 70, height: 70)
        .shadow(color: isDarkMode ? Color.white.opacity(0.8) : Color.black.opacity(0), radius: 5)
        .overlay(
            // Arrow-shape icon inside the Circle
            Image(systemName: "arrow.right")
                .font(.system(size: 40))
                .foregroundColor(Color(isDarkMode ? "SubColor_Dark" : "Color"))
            )// overlay

}// NavigationLink
```

Figure 3: Navigation button UI

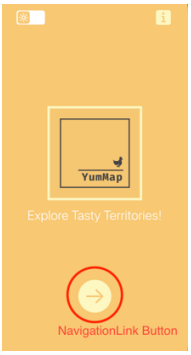

Figure 4: Navigation button code

Declare NavigationLink and assign NavigationListView() to the parameter destination, press this button to go to NavigationListView(), the main menu View. In addition, the

shape of the button was rounded using Circle(), and the arrow-shape was inserted inside the button using overlay() to suggest that the page will be moved.

2. The small information icon button to pop-up a small view displaying the app author
Short video: Information Icon Button.mp4 in GIF Folder

To create Information icon Button, Button() is used. When the Button() user taps, it performs the specified action.

```
// Information icon Button
Button(action: {
    isPopoverInfo.toggle()
})
{
    // i-shape icon
    Image(systemName: "info.square.fill")
        .font(.system(size: 35))
        .foregroundColor(Color(isDarkMode ? "Color_Dark" : "SubColor"))
        .shadow(color: isDarkMode ? Color.white.opacity(0.6) : Color.black.opacity(0.0), radius: 3)
}

//Alert to show the information of me
.alert(isPresented: $isPopoverInfo) {
    Alert(
        title: Text("My information!"),
        message: Text("Name: Hanjun Lee \nStudent ID: S3732878 \nMy program: YumMap(Save great
            restaurants)"),
        dismissButton: .default(Text("Back"))
    )//Alert
}//alert
```

Figure 5: Info button code

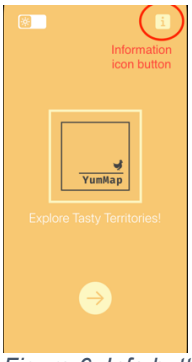

Figure 6: Info button UI

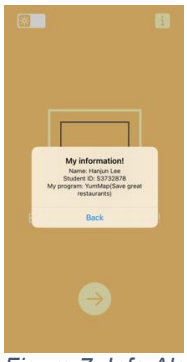

Figure 7: Info Alert

As can be seen in Figure 3, the operation performed by this button is isPopover Info.toggle(). This action reverses the value of the Boolean type variable, which reverses the value to true if isPopoverInfo is false and false if true.
When isPopoverInfo is true, due to .alert(isPresented: $isPopoverInfo) below, the title is "My information" and the message is "Name: Hanjun Lee \nStudent ID: s3732878…" is displayed in alert window. By pressing the dismissButton, the value of isPopoverInfo is changed to false. Then the alert window will be closed.

3. The main menu buttons

```
var body: some View {
    // RoundedRectangle shape
    RoundedRectangle(cornerRadius: 10)
        .fill(Color(isDarkMode ?  "Color_Dark" : "Color"))
        .frame(width: 150, height: 150)
        .shadow(color: isDarkMode ? Color.white.opacity(0.3) : Color.black.opacity(0.5),
            radius: 5)
    //content inside the RoundedRectangle(icon & icon name)
        .overlay(
            ZStack{
                VStack{
                    // Inserting icon
                    Image(buttonName)
                        .resizable()
                        .aspectRatio(contentMode: .fit)
                        .frame(width: 80)
                        .foregroundColor(Color.white)
                    Spacer().frame(height: 7)
                    // Inserting icon name
                    Text(buttonName)
                        .font(.system(size: 25))
                        .fontWeight(.bold)
                        .foregroundColor(Color.white)
                }
```

Figure 6: MenuButton() code

Figure 7: MenuButton() UI

To make a list by categorizing items according to their type, we made MenuButton() to make buttons according to each type. The background of the button is made with RoundedRectangle(), and the icon and item type is put inside the button by using overlay(). Figure 7 is a button for Lunch type items.

```
// For ranging the Navigation Buttons(Breakfast, Lunch, Dinner...)
VStack{
    // For ranging two Navigatikon Buttons in horizon
    HStack{
        // Button to move to the Breakfast Item list
        NavigationLink(destination: BreakfastList(isDarkMode: $isDarkMode)){
            MenuButton(buttonName: "Breakfast", isDarkMode: $isDarkMode)
        }// NavigationLink to Breakfast list
        Spacer().frame(width: 40)
        // Button to move to the Lunch Item list
        NavigationLink(destination: LunchList(isDarkMode: $isDarkMode)){
            MenuButton(buttonName: "Lunch", isDarkMode: $isDarkMode)
        }// NavigationLink to Lunch list
    }// HStack
    Spacer().frame(height: 40)
    HStack{
        // Button to move to the Dinner Item list
        NavigationLink(destination: DinnerList(isDarkMode: $isDarkMode)){
            MenuButton(buttonName: "Dinner", isDarkMode: $isDarkMode)
        }// NavigationLink to Dinner list
        Spacer().frame(width: 40)
        // Button to move to the Coffee Item list
        NavigationLink(destination: CoffeeList(isDarkMode: $isDarkMode)){
            MenuButton(buttonName: "Coffee", isDarkMode: $isDarkMode)
        }// NavigationLink to Coffee list
    }// HStack
    Spacer().frame(height: 40)
    HStack{
        // Button to move to the Pub Item list
        NavigationLink(destination: PubList(isDarkMode: $isDarkMode)){
            MenuButton(buttonName: "Pub", isDarkMode: $isDarkMode)
        }// NavigationLink to Pub list
        Spacer().frame(width: 40)
        // Button to move to the All Item list
        NavigationLink(destination: AllItemList(isDarkMode: $isDarkMode)){
            MenuButton(buttonName: "All", isDarkMode: $isDarkMode)
```

Figure 8: Part of MainMenu() code

Figure 9: MainMenu() UI

In MainMenu(), I created 6 MenuBotton() to create a list for each item type(Breakfast, Lunch, Dinner, etc.). When each button is pressed due to NavigationLink, it moves to BreakfastList(), LunchList(), DinnerList(), etc., which shows a list suitable for each type of button.

4. The navigation lists of items

Short video: Item Lists.mp4 in GIF Folder

I used ForEach() to create an item list. The reason why ForEach() was used instead of List() was to make the UI suitable for the design of the application more. First, an ItemRow() view was created to represent each element of the list ForEach().

```
@Binding var isDarkMode: Bool
var body: some View {
    //ZStack contains item image, name, subname and rate with star image
    ZStack(alignment: .leading){
        Color(isDarkMode ?  "Color_Dark" : "Color")
        //HStack contains item name and subname
        HStack(){
            Spacer().frame(width: 15)
            // item image
            rowItem.image
                .resizable()
                .frame(width: 110, height: 110)
                .cornerRadius(10)
            Spacer().frame(width: 20)
            VStack(alignment: .center){
                // item name
                Text(rowItem.name)
                    .font(.system(size: 20))
                    .fontWeight(.bold)
                    .foregroundColor(isDarkMode ? Color.white : Color.black)
                Spacer().frame(height: 10)
                // item subname
                Text(rowItem.subInfomation)
                    .foregroundColor(Color.gray)
                    .offset(y: -5)
            }//VStack
        }//HStack
        // VStack contains rate with star-shape icon
        VStack{
            // star-shape icon
            Image(systemName: "star.fill")
                .font(.system(size: 20))
                .foregroundColor(Color("SubColor"))
            // item rate
            Text(String(rowItem.rate))
                .font(.system(size: 15))
                .foregroundColor(isDarkMode ? Color.white : Color.black)
        }//VStack
```

Figure 8: ItemRow() code

Figure 9: ItemRow() UI

ItemRow() received an Item type instance with the parameter rowItem: Item, and used its properties to put an image, name, and short description and rate of the item into ItemRow().

```
// Making list by using ForEach
ForEach(itemArr) {
    item in
    // when the user choose "All" in NavigationListView, showing all items
    if listName == "All"{
        NavigationLink(destination: ItemView(eachItem: item, isDarkMode:
            $isDarkMode)){
            ItemRow(rowItem: item, isDarkMode: $isDarkMode)
        }
        .frame(width: 390, height: 140)
        Spacer().frame(height: 32)

    }
    //When the user choose other type of item, showing items depending on
        type
    else{
        if(item.type == listName){
            NavigationLink(destination: ItemView(eachItem: item, isDarkMode:
                $isDarkMode)){
                ItemRow(rowItem: item, isDarkMode: $isDarkMode)
            }
            .frame(width: 390, height: 140)
            Spacer().frame(height: 32)
        }// if statement
    }//else statement
}// ForEach
```

Figure 10: ForEach() code in MakingList()

Figure 11: MakingList() UI

I got an Item type array with the variable itemArr and created an item list by giving the Item type parameter to ItemRow() using ForEach().

5. The detailed view of each item (Details: image, name, address, description and other Extra information)
Short video: Detail information.mp4 in GIF Folder

First, to show the location of the item on the map, ItemMap() view that displays the location of the item on the map, was created by using MapKit library. In addition, CoreLocation library was used to store the value of latitude and longitude of the item, and Coordinates including latitude and longitude values was defined. Also, latitude and longitude values were delivered to the CLLocationcoordinate2D type property called by locationCoordinate.

```
// Declaring struct Item to store the information about the restaurant
struct Item: Identifiable, Codable{
    var id: UUID {
        UUID()
    }
    let name: String // name of restaurant
    let address: String // address of restaurant
    let type: String // type of address of restaurant(Breakfast, Lunch, Dinner, Coffee, Pub)
    let phone: String // Phone number of the restaurant
    let rate: Double // rating of the restaurant
    let imageName: String // the name of main image of the restaurant
    let information: String // Short description for the restaurant
    let subInfomation: String // additional type of restaurant (depending on restaurant)
    let imageArray: [String]
    let coordinates: Coordinates
    // Method
    var image: Image{ // method to inserting restaurant image
        Image(imageName)
    }

    var locationCoordinate : CLLocationCoordinate2D{
        CLLocationCoordinate2D(latitude: coordinates.latitude, longitude: coordinates.longitude)
    }
}

struct Coordinates: Codable{
    var latitude: Double
    var longitude: Double
}
```

Figure 12: properties of Item and Coordinates struct

```
struct ItemMap: View {
    // To get the location of the Item
    let ItemCoordinate: Item
    // Set ItemRegion as MKCoordinateRegion to store the center of map and Zoom Ratio
    @State private var ItemRegion = MKCoordinateRegion()

    var body: some View {
        //Show the Map
        // Set the area to present ItemRegion
        // Store the MapMaker data by using annotationItems
        // Creating Map Marker based on the data of annotationItems by using ItemCoordinate
        Map(coordinateRegion: $ItemRegion, annotationItems: [ItemCoordinate])
        {
            ItemCoordinate in MapMarker(coordinate: ItemCoordinate.locationCoordinate, tint:
                Color.red)

        }
            // Set the initial state of the map
            .onAppear{
                ItemRegion = MKCoordinateRegion(center: ItemCoordinate.locationCoordinate, span:
                    MKCoordinateSpan(latitudeDelta: 0.004, longitudeDelta: 0.004))
```

Figure 13: ItemMap() code

In the figure 13, the value of the latitude and longitude are delivered by the locationCoordinate that the property of the Item instance ItemCoordinate.
I saved the MKCoordinateRegion type variable that stores the center and zoom ratio of the map in ItemRegion. In addition, the initial center of the map was set to the location of the item using OnAppear(), and the zoom ratio was set to 0.004.

Second, InfoButton() was created to show images, addresses, phone numbers, and descriptions of items as users press the buttons.

```
// Creating buttons to show the information
ForEach(0...3, id: \.self){
    index in
    VStack{
        Circle() // Creating the Rounded Rectangle
            .fill(Color(isDarkMode ? "Color_Dark" : "Color")) // Making it always keep the color
                white
            .frame(height: 70)
            .shadow(color: isDarkMode ? Color.white.opacity(0.3) : Color.black.opacity(0.0),
                radius: 5)
            .overlay(
                ZStack{
                    Image(systemName: isInfoItem[index].icon)
                        .font(.system(size: 25))
                        .foregroundColor(Color.white)

                }//ZStack
            )//overlay
            .onTapGesture {
                withAnimation(
                    //if-else statement to check whitch button that user tap
                    if index == 0{
                        infoTitle = isInfoItem[index].iconName
                        infoText = buttonItem.address
                        isSlide = false
                        if !isInfo{
                            isInfo.toggle()
                        }

                    }
                    else if index == 1{
                        infoTitle = isInfoItem[index].iconName
                        infoText = buttonItem.phone
                        isSlide = false
                        if !isInfo{
                            isInfo.toggle()
                        }
                    }
                    else if index == 2{
                        infoTitle = isInfoItem[index].iconName
                        infoText = buttonItem.information
                        isSlide = true
                        if !isInfo{
                            isInfo.toggle()
                        }
                    }
                    else{
                        isSlide = true
                        isInfo = false
                    }
```

Figure 14: ForEach() to creating four information buttons

Figure 15: Four information buttons UI

Figure 16: Four information buttons UI(Operated)

Using ForEach(), take a string-type value from the isInfoItem[] array containing the names and image names of buttons and obtain names such as Address, Phone, etc. under the button and the name of the image located in the center of the button. In addition, through the if-else statement below, information is displayed under the button according to the button pressed by the user by manipulating the Boolean type isInfo and isSlide that determine whether the button operates.

Finally, I created an ItemSlide() view to show additional pictures of items in a slide format.

```
HStack{
    Spacer().frame(width: 5)
    // Button to slide the picture to the left side
    Circle() // Creating the Rounded Rectangle
        .fill(Color(isDarkMode ?  "Coler_Dark" : "Color"))
        .frame(width: 30, height: 30)
        .overlay(
            ZStack{
                Image(systemName: "arrow.left")
                    .foregroundColor(Color.white)
            }//ZStack
            )//overlay
    // When user tap the button, show the next picture
        .onTapGesture {
            // if statement to make a cycle of the image
            if imageIndex > 0 {
                imageIndex -= 1
            }
            else{
                imageIndex = slideItem.imageArray.count - 1
            }
        }// ontapGesture
    // Inserting image between the buttons
    Image(slideItem.imageArray[imageIndex])
        .resizable()
        .aspectRatio(contentMode: .fill)
        .frame(width: 300, height: 200 )
        .clipped()
        .clipShape(RoundedRectangle(cornerRadius: 15))
    //Button to slide picture to right side
    Circle() // Creating the Rounded Rectangle
        .fill(Color(isDarkMode ?  "Coler_Dark" : "Color"))
        .frame(width: 30, height: 30)
        .overlay(
            ZStack{
                Image(systemName: "arrow.right")
                    .foregroundColor(Color.white)
            }//ZStack
            )// overlay

        .onTapGesture {
            // if statement to make a cycle of the image
            if imageIndex < slideItem.imageArray.count - 1 {
                imageIndex += 1
            }
            else{
                imageIndex = 0
            }
        }
```

Figure 17: ItemSlide() code

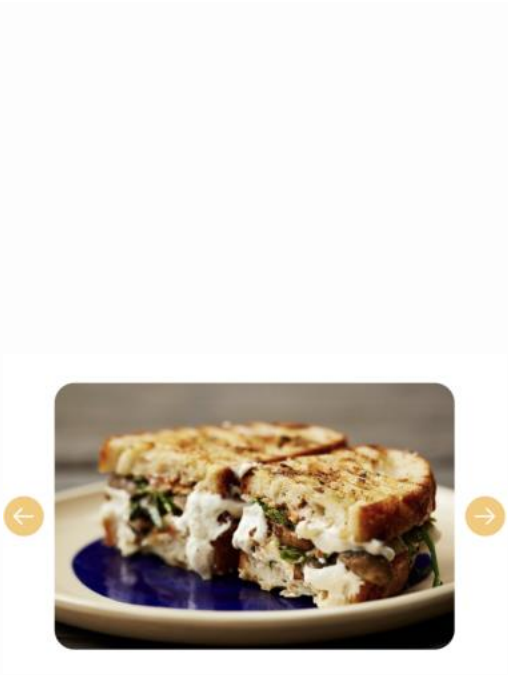

Figure 18: ItemSlide() UI

As you can see above, the structure of the ItemSlide() shows one arrow-shaped button on each side of the image. The if-else statement in each button's code manipulates the index of imageArray, the property of the Item type instance, so that the pictures change in order when the buttons are pressed.

```
ScrollView{
    // ZStack contains ItemMap, HalfPopover, Infomation Buttons
    ZStack{
        // Presenting the location of the Item on the map
        ItemMap(ItemCoordinate: eachItem)
            .frame(height: 900)
            .edgesIgnoringSafeArea(.all)
            .offset(y: isInfoPopover ? 0 : -280)
        // If user tap the empty space Popover-like window, it will be colsed
        if !isInfoPopover {
            InfoPopover(infoItem: eachItem, isInfo: $isInfo, isSlide: $isSlide, isDarkMode: $isDarkMode)
                .onTapGesture {
                    withAnimation{
                        isInfoPopover.toggle()
                    }
                }

        }// if statement
        // When the users whatching the map only, provied them a button to show more detail information of the
            Item
        if isInfoPopover{
            Button("Open Detail"){
                withAnimation{
                    isInfoPopover.toggle()
                }
            }
            .foregroundColor(.white)
            .frame(width: 200, height: 30)
            .background(Color(isDarkMode ? "Color_Dark" : "Color"))
            .cornerRadius(10)
            .font(.system(size: 25))
            .bold()
            .offset(y: 350)
```

Figure 19: ItemView() code

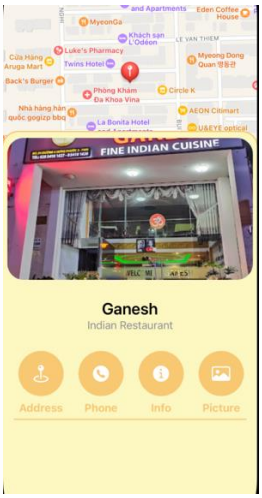

Figure20: ItemView() UI

I created ItemView() by integrating the above-mentioned views such as ItemMap(), InfoButton(). InfoButton() was run on InfoPopover(), as shown in Figure 19.

<Advanced feature requirements>

1. The searching field

Short video: Searching Bar.mp4 in GIF Folder

To configure the Searching field, I used TextField(), which allows me to receive text input from users.

```
TextField("Searching by the name!", text: $searchText)
    .padding(.leading, 10)
    .padding(.vertical, 8)
    .background(Color.white.opacity(0.8))
    .cornerRadius(10)
    //overlay to contain "x.circle" icon to reset the rearching bar content
    .overlay(
        Image(systemName: "x.circle")
            .foregroundColor(.gray)
            .font(.system(size: 20))
            .padding(.leading, 270)
        //onTapGesture (resetting searchText, closing the SearchingResult list )
            .onTapGesture {
                searchText = ""
                isSearch.toggle()
            }
    )
```

Figure 21: SearchingBar() code

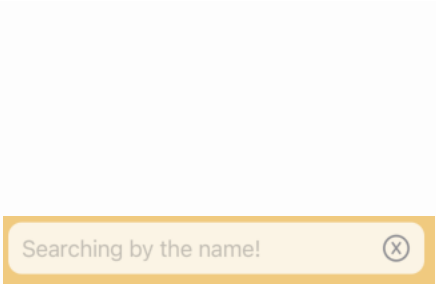

Figure 22: SearchingBar() UI

As a parameter for TextField(), I gave the string type "Searching by the name!" and instructed the user to search for the item name. I also created an x-shape button on the right to allow users to reset their search values.

In addition, I created a view called SearchingResult() to show the item names under SearchBar() that matches the user's input value. All of the components that appear in a list format in the SearchingResult(). If users tap the components, they can view information about the item.

```
List(items){
    item in
    // if user input is matched to item name, show the ItemView link
    if item.name.contains(searchText) {
        //NavigationLink to matched items
        NavigationLink{
            ItemView(eachItem: item, isDarkMode: $isDarkMode)
        } label: {
            Text("\(item.name)")
        }
    }// if statement
}// List
```

Figure 23: SearchingResult() code

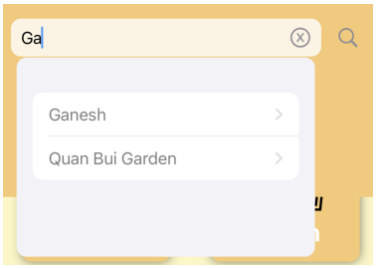

Figure 24: SearchingResult() UI

2. Some filtering options
Short video: Sort By.mp4 in GIF Folder

To allow users to view the list of items in several sort ways, a SortBy() view has been created. This feature supports two functions: "Sorted By Rate" and "Sorted By Alphabet". The array of item types for each function is declared as follows.

```
// Item array for using when the user choose Sorted by rate option
let sortedByRate: [Item] = items.sorted(by: { $0.rate > $1.rate })
// Check wheather or not user top Sorted by rate option
@State var isSortedByRate: Bool = false
// Item array for using when the user choose Sorted by Alphabet option
let sortedByAlphabet: [Item] = items.sorted(by: { $0.name < $1.name })
// Check wheather or not user top Sorted by allphabet option
@State var isSortedByAlphabet: Bool = false
```

Figure 25: Variables for SortBy()

These variables are passed to the MakingList() view by if-else statement according to the value of the option selected by the user in SortBy(). And the MakingList() rearranges the item list according to the Sorted by option selected by the user.

```
ZStack{
    //Making RoundedRectangle-spape button
    RoundedRectangle(cornerRadius: 5)
        .fill(Color(isDarkMode ? "SubColor_Dark" : "SubColor"))
        .frame(width: 230, height: 20)
        .overlay(
            //VStack contains Text(Sorted by) and arrow-shape icon
            HStack{
                Text("Sorted by")
                    .foregroundColor(isDarkMode ? Color.white : Color.black)
                Spacer().frame(width: 120)
                // Arrow Shape icon
                Image(systemName: isDropDown ? "arrowtriangle.up" : "arrowtriangle.down")
                    .foregroundColor(isDarkMode ? Color.white : Color.black)
            }
        )
    //onTapgesture (if user tap show the drow down list)
        .onTapGesture {
            withAnimation{
                isDropDown.toggle()
            }
        }
    // if statement isDropDown
    if isDropDown {
        // RoundedRectangle-Shape Drop Down frame
        RoundedRectangle(cornerRadius: 5)
            .fill(Color(isDarkMode ? "Color_Dark" : "Color"))
            .frame(width: 227, height: 80)
        //overlay to contain text and arrow-shape icon
            .overlay(
                //ZStack contains border of the Drop Down frame, texts
                ZStack{
                    //Border of Drop Down frame
                    RoundedRectangle(cornerRadius: 5)
                        .stroke(isDarkMode ? Color.white : Color("SubColor"), lineWidth: 3)
                    // VStack contains texts
                    VStack{
                        Spacer().frame(height: 10)
                        // Decide user tap Sorted By Rate
                        Text("Sorted By Rate")
                            .foregroundColor(isDarkMode ? Color.white : Color.black)
                            .onTapGesture {
                                isSortedByRate.toggle()
                                isSortedByAlphabet = false
                            }
                        Spacer().frame(height: 8)
                        // Decide user tap Sorted By Alphabet
                        Text("Sorted By Alphabet")
                            .foregroundColor(isDarkMode ? Color.white : Color.black)
                            .onTapGesture {
                                isSortedByAlphabet.toggle()
                                isSortedByRate = false
                            }
```

Figure 26: SortBy() code

```
if isSortedByRate{
    MakingList(listName: $listName, itemArr: sortedByRate, isDarkMode: $isDarkMode)
}
else if isSortedByAlphabet {
    MakingList(listName: $listName, itemArr: sortedByAlphabet, isDarkMode: $isDarkMode)
}
else{
    MakingList(listName: $listName, itemArr: normalList, isDarkMode: $isDarkMode)
}
```

Figure 27: if-else statement to pass the Item array to MakingList() in ItemList()

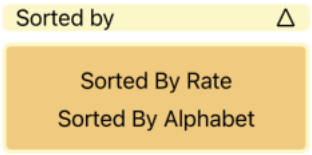

Figure 28: SortBy() UI

3. The pin displaying on the map
Short video: MapMarker.mp4 in GIF Folder

In order to accurately display the location of the item on the map with pin, [ItemCoordinate] was passed to the parameter annotationItems of Map() to provide the location information to be displayed on the MapMarker.

```
let ItemCoordinate: Item
// Set ItemRegion as MKCoordinateRegion to store the center of map and Zoom Ratio
@State private var ItemRegion = MKCoordinateRegion()

var body: some View {
    //Show the Map
    // Set the area to present ItemRegion
    // Store the MapMaker data by using annotationItems
    // Creating Map Marker based on the data of annotationItems by using ItemCoordinate
    Map(coordinateRegion: $ItemRegion, annotationItems: [ItemCoordinate])
    {
        ItemCoordinate in MapMarker(coordinate: ItemCoordinate.locationCoordinate, tint: Color.red)

    }
        // Set the initial state of the map
        .onAppear{
            ItemRegion = MKCoordinateRegion(center: ItemCoordinate.locationCoordinate, span:
                MKCoordinateSpan(latitudeDelta: 0.004, longitudeDelta: 0.004))
        }
```

Figure 29: ItemMap() code

Figure 30: ItemMap() MapMarker UI

4. The button to switch between light mode and dark mode
Short video: Dark Mode.mp4 in GIF Folder

For light mode and dark mode switching, I added 2 color sets for dark mode. I also declared Boolean type isDarkMode as a @State property for transition of light mode and dark mode at the top level View (Intro()).

```
@State var isDarkMode: Bool = false

var body: some View {
    // Most top NavigationView
    NavigationView{
        //Background
        ZStack{
            //Background color depending on the light/Dark Mode
            Color(isDarkMode ? "SubColor_Dark" : "Color")
                .edgesIgnoringSafeArea(.all)
            //Elements (Dark Mode button, Logo, Slogon, Navigation button..)
            VStack{
                // Dark Mode Button, Information icon Button
                HStack{
                    // DarkMode Button
                    RoundedRectangle(cornerRadius: 5)
                        .fill(Color.white)
                        .frame(width: 60, height: 30)
                    // overlay to put the RoundedRectangle inside the RoundedRectangle
                        .overlay(
                            RoundedRectangle(cornerRadius: 5)
                                .fill(Color(isDarkMode ? "SubColor_Dark" : "Color"))
                                .frame(width: 25, height: 26)
                                .overlay{
                                    Image(systemName: isDarkMode ? "moon.stars" : "sun.max")
                                        .bold()
                                        .foregroundColor(Color.white)
                                }
                                .offset(x: isDarkMode ? 15 : -15)
```

Figure 31: Code for dark mode button

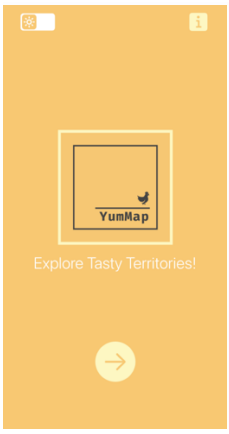

Figure 32: Example of light mode

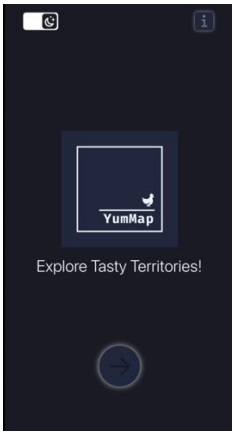

Figure33: Example of dark mode

The shape of the button is that a small rounded rectangle is placed inside the rounded rectangle using overlay(), and when pressed by the user, the small rounded rectangle moves from side to side to act like a switch. In addition, when a user presses a button, the isDarkMode value is changed by toggle(), and color elements can be changed as the value is changed by ternary condition operator like Color(isDarkMode? "Color_Dark": "Color"). Also, if you press this button, the color elements of all Views of the application must be changed at the same time, so @Binding property was used to deliver the value of isDarkMode to all Views.

< Known Bugs/Problems>

Error message: Cannot assign to property: 'self' is immutable

The difficulty that was not solved during this project was that when creating SortBy(), an item-type arrangement containing information about all items in the program called ItemList was declared. And I wanted to rearrange this array by using the sorted() function within the view according to the option selected by the user, and then pass the rearranged array back to the MakingList() view to create a list. However, after the view was executed, it was not possible to relocate the declared array or add values by using append(). In order to solve this, I've tried many things, but I cannot solve this problem and using another method to provide sort by function. The method was not to rearrange one array according to the condition, but to declare two additional arrays that met the condition from the beginning.

d. Conclusion

I made a simple application that introduces my favorite restaurants using basic Swift and SwiftUI elements that I have learned so far. While satisfying the Main feature requirements, I was able to study what I have learned so far again, and while creating Advanced feature requirements, I was able to further study and understand what I have not learned, such as MapMarker. If I study more in the future, I will be able to add features to the app that allow users to leave reviews or add their own favorite restaurants to others. In addition, it seems that I can add a function to save user’s restaurant list by creating a user's personal ID.

e. Reference

[1] “Banh Mi 362”, Google Maps, https://www.google.com/maps/place/Banh+Mi+362/@10.7302559,106.7051781,14z/data=!4m10!1m2!2m1!1zYsOhbmggbcOsIDM2Mg!3m6!1s0x31752f912522253d:0xb04123e128680524!8m2!3d10.7268849!4d106.7087971!15sCg1iw6FuaCBtw6wgMzYyIgOIAQFaDyINYsOhbmggbcOsIDM2MpIBDXNhbmR3aWNoX3Nob3DgAQA!16s%2Fg%2F11ddykt9gp?hl=en&entry=ttu (accessed Aug. 6, 2023).

[2] “Banh Mi 3621”, Google Maps, https://www.google.com/maps/place/Banh+Mi+362/@10.7302559,106.7051781,14z/data=!4m10!1m2!2m1!1zYsOhbmggbcOsIDM2Mg!3m6!1s0x31752f912522253d:0xb04123e128680524!8m2!3d10.7268849!4d106.7087971!15sCg1iw6FuaCBtw6wgMzYyIgOIAQFaDyINYsOhbmggbcOsIDM2MpIBDXNhbmR3aWNoX3Nob3DgAQA!16s%2Fg%2F11ddykt9gp?hl=en&entry=ttu (accessed Aug. 6, 2023).

[3] “Banh Mi 3622”, Google Maps, https://www.google.com/maps/place/Banh+Mi+362/@10.7302559,106.7051781,14z/data=!4m10!1m2!2m1!1zYsOhbmggbcOsIDM2Mg!3m6!1s0x31752f912522253d:0xb04123e128680524!8m2!3d10.7268849!4d106.7087971!15sCg1iw6FuaCBtw6wgMzYyIgOIAQFaDyINYsOhbmggbcOsIDM2MpIBDXNhbmR3aWNoX3Nob3DgAQA!16s%2Fg%2F11ddykt9gp?hl=en&entry=ttu (accessed Aug. 6, 2023).

[4] “Banh Mi 3623”, Google Maps, https://www.google.com/maps/place/Banh+Mi+362/@10.7302559,106.7051781,14z/data=!4m10!1m2!2m1!1zYsOhbmggbcOsIDM2Mg!3m6!1s0x31752f912522253d:0xb04123e128680524!8m2!3d10.7268849!4d106.7087971!15sCg1iw6FuaCBtw6wgMzYyIgOIAQFaDyINYsOhbmggbcOsIDM2MpIBDXNhbmR3aWNoX3Nob3DgAQA!16s%2Fg%2F11ddykt9gp?hl=en&entry=ttu (accessed Aug. 6, 2023).

[5] “Boomerang Bistro Saigon”, https://www.google.com/maps/place/Boomerang+Bistro+Saigon/@10.7262831,106.7204036,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f8bb258914f:0x121e666b3bc665c4!8m2!3d10.7262831!4d106.7204036!16s%2Fg%2F113j6v1f0?hl=en&entry=ttu (accessed Aug. 6, 2023).

[6] “Boomerang Bistro Saigon1”, https://www.google.com/maps/place/Boomerang+Bistro+Saigon/@10.7262831,106.7204036,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f8bb258914f:0x121e666b3bc665c4!8m2!3d10.7262831!4d106.7204036!16s%2Fg%2F113j6v1f0?hl=en&entry=ttu (accessed Aug. 6, 2023).

[7] “Boomerang Bistro Saigon2”, https://www.google.com/maps/place/Boomerang+Bistro+Saigon/@10.7262831,106.7204036,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f8bb258914f:0x121e666b3bc665c4!8m2!3d10.7262831!4d106.7204036!16s%2Fg%2F113j6v1f0?hl=en&entry=ttu (accessed Aug. 6, 2023).

[8] “Boomerang Bistro Saigon3”, https://www.google.com/maps/place/Boomerang+Bistro+Saigon/@10.7262831,106.7204036,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f8bb258914f:0x121e666b3bc665c4!8m2!3d10.7262831!4d106.7204036!16s%2Fg%2F113j6v1f0?hl=en&entry=ttu (accessed Aug. 6, 2023).

[9] “Cafe Mongdang”, Google Maps, https://www.google.com/maps/place/Cafe+Mongdang/@10.7306635,106.7069428,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f913c18d6c3:0xdb806162ee8f795e!8m2!3d10.7306635!4d106.7069428!16s%2Fg%2F11f0_ghvwd?hl=en&entry=ttu (accessed Aug. 6, 2023).

[10] “Cafe Mongdang1”, Google Maps, https://www.google.com/maps/place/Cafe+Mongdang/@10.7306635,106.7069428,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f913c18d6c3:0xdb806162ee8f795e!8m2!3d10.7306635!4d106.7069428!16s%2Fg%2F11f0_ghvwd?hl=en&entry=ttu (accessed Aug. 6, 2023).

[11] “Cafe Mongdang2”, Google Maps, https://www.google.com/maps/place/Cafe+Mongdang/@10.7306635,106.7069428,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f913c18d6c3:0xdb806162ee8f795e!8m2!3d10.7306635!4d106.7069428!16s%2Fg%2F11f0_ghvwd?hl=en&entry=ttu (accessed Aug. 6, 2023).

[12] “Cafe Mongdang3”, Google Maps, https://www.google.com/maps/place/Cafe+Mongdang/@10.7306635,106.7069428,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f913c18d6c3:0xdb806162ee8f795e!8m2!3d10.7306635!4d106.7069428!16s%2Fg%2F11f0_ghvwd?hl=en&entry=ttu (accessed Aug. 6, 2023).

[13] “Cheesecake Ngon Cake Shop”, Google Maps, https://www.google.com/maps/place/Cheesecake+Ngon+Cake+Shop+-+Thao+Dien/@10.8024973,106.7326149,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f330a0d7c89:0xf0659735889eefd6!8m2!3d10.8024973!4d106.7326149!16s%2Fg%2F1hjgxxf9z?hl=en&entry=ttu (accessed Aug. 6, 2023).

[14] “Cheesecake Ngon Cake Shop1”, Google Maps, https://www.google.com/maps/place/Cheesecake+Ngon+Cake+Shop+-+Thao+Dien/@10.8024973,106.7326149,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f330a0d7c89:0xf0659735889eefd6!8m2!3d10.8024973!4d106.7326149!16s%2Fg%2F1hjgxxf9z?hl=en&entry=ttu (accessed Aug. 6, 2023).

[15] “Cheesecake Ngon Cake Shop2”, Google Maps, https://www.google.com/maps/place/Cheesecake+Ngon+Cake+Shop+-+Thao+Dien/@10.8024973,106.7326149,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f330a0d7c89

:0xf0659735889eefd6!8m2!3d10.8024973!4d106.7326149!16s%2Fg%2F1hjgxxf9z?hl=en&entry=ttu (accessed Aug. 6, 2023).

[16] “Cheesecake Ngon Cake Shop3”, Google Maps, https://www.google.com/maps/place/Cheesecake+Ngon+Cake+Shop+-+Thao+Dien/@10.8024973,106.7326149,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f330a0d7c89:0xf0659735889eefd6!8m2!3d10.8024973!4d106.7326149!16s%2Fg%2F1hjgxxf9z?hl=en&entry=ttu (accessed Aug. 6, 2023).

[17] “Ganesh”, Google Maps, https://www.google.com/maps/place/Ganesh+Indian+Restaurant/@10.7302559,106.707753,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f916cca6167:0x4eb2614232a10b90!8m2!3d10.7302559!4d106.707753!16s%2Fg%2F11b6_r7q5q?hl=en&entry=ttu (accessed Aug. 6, 2023).

[18] “Ganesh1”, Google Maps, https://www.google.com/maps/place/Ganesh+Indian+Restaurant/@10.7302559,106.707753,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f916cca6167:0x4eb2614232a10b90!8m2!3d10.7302559!4d106.707753!16s%2Fg%2F11b6_r7q5q?hl=en&entry=ttu (accessed Aug. 6, 2023).

[19] “Ganesh2”, Google Maps, https://www.google.com/maps/place/Ganesh+Indian+Restaurant/@10.7302559,106.707753,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f916cca6167:0x4eb2614232a10b90!8m2!3d10.7302559!4d106.707753!16s%2Fg%2F11b6_r7q5q?hl=en&entry=ttu (accessed Aug. 6, 2023).

[20] “Ganesh3”, Google Maps, https://www.google.com/maps/place/Ganesh+Indian+Restaurant/@10.7302559,106.707753,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f916cca6167:0x4eb2614232a10b90!8m2!3d10.7302559!4d106.707753!16s%2Fg%2F11b6_r7q5q?hl=en&entry=ttu (accessed Aug. 6, 2023).

[21] “Kashew Cheese Deli”, Google Maps, https://www.google.com/maps/place/Kashew+Cheese+Deli/@10.8033925,106.7391788,17z/data=!3m1!4b1!4m6!3m5!1s0x3175274ee9f5fb59:0x6c9e640fb0838eee!8m2!3d10.8033925!4d106.7391788!16s%2Fg%2F11h40yrh_w?hl=en&entry=ttu (accessed Aug. 6, 2023).

[22] “Kashew Cheese Deli1”, Google Maps, https://www.google.com/maps/place/Kashew+Cheese+Deli/@10.8033925,106.7391788,17z/data=!3m1!4b1!4m6!3m5!1s0x3175274ee9f5fb59:0x6c9e640fb0838eee!8m2!3d10.8033925!4d106.7391788!16s%2Fg%2F11h40yrh_w?hl=en&entry=ttu (accessed Aug. 6, 2023).

[23] “Kashew Cheese Deli2”, Google Maps, https://www.google.com/maps/place/Kashew+Cheese+Deli/@10.8033925,106.7391788,17z/data=!3m1!4b1!4m6!3m5!1s0x3175274ee9f5fb59:0x6c9e640fb0838eee!8m2!3d10.8033925!4d106.7391788!16s%2Fg%2F11h40yrh_w?hl=en&entry=ttu (accessed Aug. 6, 2023).

[24] “Kashew Cheese Deli3”, Google Maps, https://www.google.com/maps/place/Kashew+Cheese+Deli/@10.8033925,106.7391788,17z/data

=!3m1!4b1!4m6!3m5!1s0x3175274ee9f5fb59:0x6c9e640fb0838eee!8m2!3d10.8033925!4d106.7391788!16s%2Fg%2F11h40yrh_w?hl=en&entry=ttu (accessed Aug. 6, 2023).

[25] “kimgane kimbap”, Google Maps, https://www.google.com/maps/place/김가네김밥+kimgane+kimbap+Q7/@10.731783,106.6380619,12z/data=!4m10!1m2!2m1!1skimgane+kimbap!3m6!1s0x31752f0333da3e53:0x4de26cac51329df3!8m2!3d10.731783!4d106.7080997!15sCg5raW1nYW5lIGtpbWJhcFoQIg5raW1nYW5lIGtpbWJhcJIBEWtvcmVhbl9yZXN0YXVyYW504AEA!16s%2Fg%2F11txfpp45d?hl=en&entry=ttu (accessed Aug. 6, 2023).

[26] “kimgane kimbap1”, Google Maps, https://www.google.com/maps/place/김가네김밥+kimgane+kimbap+Q7/@10.731783,106.6380619,12z/data=!4m10!1m2!2m1!1skimgane+kimbap!3m6!1s0x31752f0333da3e53:0x4de26cac51329df3!8m2!3d10.731783!4d106.7080997!15sCg5raW1nYW5lIGtpbWJhcFoQIg5raW1nYW5lIGtpbWJhcJIBEWtvcmVhbl9yZXN0YXVyYW504AEA!16s%2Fg%2F11txfpp45d?hl=en&entry=ttu (accessed Aug. 6, 2023).

[27] “kimgane kimbap2”, Google Maps, https://www.google.com/maps/place/김가네김밥+kimgane+kimbap+Q7/@10.731783,106.6380619,12z/data=!4m10!1m2!2m1!1skimgane+kimbap!3m6!1s0x31752f0333da3e53:0x4de26cac51329df3!8m2!3d10.731783!4d106.7080997!15sCg5raW1nYW5lIGtpbWJhcFoQIg5raW1nYW5lIGtpbWJhcJIBEWtvcmVhbl9yZXN0YXVyYW504AEA!16s%2Fg%2F11txfpp45d?hl=en&entry=ttu (accessed Aug. 6, 2023).

[28] “kimgane kimbap3”, Google Maps, https://www.google.com/maps/place/김가네김밥+kimgane+kimbap+Q7/@10.731783,106.6380619,12z/data=!4m10!1m2!2m1!1skimgane+kimbap!3m6!1s0x31752f0333da3e53:0x4de26cac51329df3!8m2!3d10.731783!4d106.7080997!15sCg5raW1nYW5lIGtpbWJhcFoQIg5raW1nYW5lIGtpbWJhcJIBEWtvcmVhbl9yZXN0YXVyYW504AEA!16s%2Fg%2F11txfpp45d?hl=en&entry=ttu (accessed Aug. 6, 2023).

[29] “Nha hang Lang man Oc”, Google Maps, https://www.google.com/maps/place/Nhà+hàng+Lãng+mạn+Ốc/@10.8026659,106.7362663,17z/data=!3m1!4b1!4m6!3m5!1s0x317527704b91fe03:0x4f280db284c38c69!8m2!3d10.8026659!4d106.7362663!16s%2Fg%2F11jvc_txlm?hl=en&entry=ttu (accessed Aug. 6, 2023).

[30] “Nha hang Lang man Oc1”, Google Maps, https://www.google.com/maps/place/Nhà+hàng+Lãng+mạn+Ốc/@10.8026659,106.7362663,17z/data=!3m1!4b1!4m6!3m5!1s0x317527704b91fe03:0x4f280db284c38c69!8m2!3d10.8026659!4d106.7362663!16s%2Fg%2F11jvc_txlm?hl=en&entry=ttu (accessed Aug. 6, 2023).

[31] “Nha hang Lang man Oc2”, Google Maps, https://www.google.com/maps/place/Nhà+hàng+Lãng+mạn+Ốc/@10.8026659,106.7362663,17z/data=!3m1!4b1!4m6!3m5!1s0x317527704b91fe03:0x4f280db284c38c69!8m2!3d10.8026659!4d106.7362663!16s%2Fg%2F11jvc_txlm?hl=en&entry=ttu (accessed Aug. 6, 2023).

[32] “Nha hang Lang man Oc3”, Google Maps, https://www.google.com/maps/place/Nhà+hàng+Lãng+mạn+Ốc/@10.8026659,106.7362663,17z/data=!3m1!4b1!4m6!3m5!1s0x317527704b91fe03:0x4f280db284c38c69!8m2!3d10.8026659!4d106.7362663!16s%2Fg%2F11jvc_txlm?hl=en&entry=ttu (accessed Aug. 6, 2023).

[33] “Østerberg Ice Cream”, Google Maps, https://www.google.com/maps/place/Østerberg+Ice+Cream/@10.8037163,106.7349952,17z/data=!3m1!4b1!4m6!3m5!1s0x31752619fec37adb:0xad30b23e4730be0b!8m2!3d10.8037163!4d106.7349952!16s%2Fg%2F11c57dl40b?hl=en&entry=ttu (accessed Aug. 6, 2023).

[34] “Østerberg Ice Cream1”, Google Maps, https://www.google.com/maps/place/Østerberg+Ice+Cream/@10.8037163,106.7349952,17z/data=!3m1!4b1!4m6!3m5!1s0x31752619fec37adb:0xad30b23e4730be0b!8m2!3d10.8037163!4d106.7349952!16s%2Fg%2F11c57dl40b?hl=en&entry=ttu (accessed Aug. 6, 2023).

[35] “Østerberg Ice Cream2”, Google Maps, https://www.google.com/maps/place/Østerberg+Ice+Cream/@10.8037163,106.7349952,17z/data=!3m1!4b1!4m6!3m5!1s0x31752619fec37adb:0xad30b23e4730be0b!8m2!3d10.8037163!4d106.7349952!16s%2Fg%2F11c57dl40b?hl=en&entry=ttu (accessed Aug. 6, 2023).

[36] “Østerberg Ice Cream3”, Google Maps, https://www.google.com/maps/place/Østerberg+Ice+Cream/@10.8037163,106.7349952,17z/data=!3m1!4b1!4m6!3m5!1s0x31752619fec37adb:0xad30b23e4730be0b!8m2!3d10.8037163!4d106.7349952!16s%2Fg%2F11c57dl40b?hl=en&entry=ttu (accessed Aug. 6, 2023).

[37] “Pho Trang”, Google Maps, https://www.google.com/maps/place/Phở+Trang+-+Phạm+Thái+Bường/@10.7274926,106.6900695,14z/data=!4m10!1m2!2m1!1spho+trang!3m6!1s0x31752f1bed1a44cf:0x1449743f4b0f70ec!8m2!3d10.7274926!4d106.707579!15sCglwaG8gdHJhbmdaCyIJcGhvIHRyYW5nkgEOcGhvX3Jlc3RhdXJhbnTgAQA!16s%2Fg%2F11jgkht05f?hl=en&entry=ttu (accessed Aug. 6, 2023).

[38] “Pho Trang1”, Google Maps, https://www.google.com/maps/place/Phở+Trang+-+Phạm+Thái+Bường/@10.7274926,106.6900695,14z/data=!4m10!1m2!2m1!1spho+trang!3m6!1s0x31752f1bed1a44cf:0x1449743f4b0f70ec!8m2!3d10.7274926!4d106.707579!15sCglwaG8gdHJhbmdaCyIJcGhvIHRyYW5nkgEOcGhvX3Jlc3RhdXJhbnTgAQA!16s%2Fg%2F11jgkht05f?hl=en&entry=ttu (accessed Aug. 6, 2023).

[39] “Pho Trang2”, Google Maps, https://www.google.com/maps/place/Phở+Trang+-+Phạm+Thái+Bường/@10.7274926,106.6900695,14z/data=!4m10!1m2!2m1!1spho+trang!3m6!1s0x31752f1bed1a44cf:0x1449743f4b0f70ec!8m2!3d10.7274926!4d106.707579!15sCglwaG8gdHJhbmdaCyIJcGhvIHRyYW5nkgEOcGhvX3Jlc3RhdXJhbnTgAQA!16s%2Fg%2F11jgkht05f?hl=en&entry=ttu (accessed Aug. 6, 2023).

[40] “Pho Trang3”, Google Maps, https://www.google.com/maps/place/Phở+Trang+-+Phạm+Thái+Bường/@10.7274926,106.6900695,14z/data=!4m10!1m2!2m1!1spho+trang!3m6!1s0x31752f1bed1a44cf:0x1449743f4b0f70ec!8m2!3d10.7274926!4d106.707579!15sCglwaG8g

dHJhbmdaCyIJcGhvIHRyYW5nkgEOcGhvX3Jlc3RhdXJhbnTgAQA!16s%2Fg%2F11jgkht05f?hl=en&entry=ttu (accessed Aug. 6, 2023).

[41] “Quan Bui Garden”, Google Maps, https://www.google.com/maps/place/Quan+Bui+Garden/@10.805121,106.735759,17z/data=!3m1!4b1!4m6!3m5!1s0x317526176671636b:0x471be660ffc59db!8m2!3d10.805121!4d106.735759!16s%2Fg%2F11c0rh7rdx?hl=en&entry=ttu (accessed Aug. 6, 2023).

[42] “Quan Bui Garden1”, Google Maps, https://www.google.com/maps/place/Quan+Bui+Garden/@10.805121,106.735759,17z/data=!3m1!4b1!4m6!3m5!1s0x317526176671636b:0x471be660ffc59db!8m2!3d10.805121!4d106.735759!16s%2Fg%2F11c0rh7rdx?hl=en&entry=ttu (accessed Aug. 6, 2023).

[43] “Quan Bui Garden2”, Google Maps, https://www.google.com/maps/place/Quan+Bui+Garden/@10.805121,106.735759,17z/data=!3m1!4b1!4m6!3m5!1s0x317526176671636b:0x471be660ffc59db!8m2!3d10.805121!4d106.735759!16s%2Fg%2F11c0rh7rdx?hl=en&entry=ttu (accessed Aug. 6, 2023).

[44] “Quan Bui Garden3”, Google Maps, https://www.google.com/maps/place/Quan+Bui+Garden/@10.805121,106.735759,17z/data=!3m1!4b1!4m6!3m5!1s0x317526176671636b:0x471be660ffc59db!8m2!3d10.805121!4d106.735759!16s%2Fg%2F11c0rh7rdx?hl=en&entry=ttu (accessed Aug. 6, 2023).

[45] “Quan Ut Ut Xuan Thuy”, Google Maps, https://www.google.com/maps/place/Quán+Ụt+Ụt+Xuân+Thủy+%7C+Q.2/@10.80344,106.73237,17z/data=!3m1!4b1!4m6!3m5!1s0x3175272afe5a22fb:0x6ace004025d5e769!8m2!3d10.80344!4d106.73237!16s%2Fg%2F11f8h9v4bh?hl=en&entry=ttu (accessed Aug. 6, 2023).

[46] “Quan Ut Ut Xuan Thuy1”, Google Maps, https://www.google.com/maps/place/Quán+Ụt+Ụt+Xuân+Thủy+%7C+Q.2/@10.80344,106.73237,17z/data=!3m1!4b1!4m6!3m5!1s0x3175272afe5a22fb:0x6ace004025d5e769!8m2!3d10.80344!4d106.73237!16s%2Fg%2F11f8h9v4bh?hl=en&entry=ttu (accessed Aug. 6, 2023).

[47] “Quan Ut Ut Xuan Thuy2”, Google Maps, https://www.google.com/maps/place/Quán+Ụt+Ụt+Xuân+Thủy+%7C+Q.2/@10.80344,106.73237,17z/data=!3m1!4b1!4m6!3m5!1s0x3175272afe5a22fb:0x6ace004025d5e769!8m2!3d10.80344!4d106.73237!16s%2Fg%2F11f8h9v4bh?hl=en&entry=ttu (accessed Aug. 6, 2023).

[48] “Quan Ut Ut Xuan Thuy3”, Google Maps, https://www.google.com/maps/place/Quán+Ụt+Ụt+Xuân+Thủy+%7C+Q.2/@10.80344,106.73237,17z/data=!3m1!4b1!4m6!3m5!1s0x3175272afe5a22fb:0x6ace004025d5e769!8m2!3d10.80344!4d106.73237!16s%2Fg%2F11f8h9v4bh?hl=en&entry=ttu (accessed Aug. 6, 2023).

[49] “Ruby Soho”, Google Maps, https://www.google.com/maps/place/Ruby+Soho/@10.7311186,106.7065695,17z/data=!3m1!4b

1!4m6!3m5!1s0x31752f91244fbdd1:0xc9872d030e9572b3!8m2!3d10.7311186!4d106.7065695!16s%2Fg%2F11zjdy0pr?hl=en&entry=ttu (accessed Aug. 6, 2023).

[50] “Ruby Soho1”, Google Maps, https://www.google.com/maps/place/Ruby+Soho/@10.7311186,106.7065695,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f91244fbdd1:0xc9872d030e9572b3!8m2!3d10.7311186!4d106.7065695!16s%2Fg%2F11zjdy0pr?hl=en&entry=ttu (accessed Aug. 6, 2023).

[51] “Ruby Soho2”, Google Maps, https://www.google.com/maps/place/Ruby+Soho/@10.7311186,106.7065695,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f91244fbdd1:0xc9872d030e9572b3!8m2!3d10.7311186!4d106.7065695!16s%2Fg%2F11zjdy0pr?hl=en&entry=ttu (accessed Aug. 6, 2023).

[52] “Ruby Soho3”, Google Maps, https://www.google.com/maps/place/Ruby+Soho/@10.7311186,106.7065695,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f91244fbdd1:0xc9872d030e9572b3!8m2!3d10.7311186!4d106.7065695!16s%2Fg%2F11zjdy0pr?hl=en&entry=ttu (accessed Aug. 6, 2023).

[53] “Speakeasy”, Google Maps, https://www.google.com/maps/place/Speakeasy/@10.7309532,106.7088204,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f910786b665:0x7c1178ecd6efa743!8m2!3d10.7309532!4d106.7088204!16s%2Fg%2F11d_c0xg31?hl=en&entry=ttu (accessed Aug. 6, 2023).

[54] “Speakeasy1”, Google Maps, https://www.google.com/maps/place/Speakeasy/@10.7309532,106.7088204,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f910786b665:0x7c1178ecd6efa743!8m2!3d10.7309532!4d106.7088204!16s%2Fg%2F11d_c0xg31?hl=en&entry=ttu (accessed Aug. 6, 2023).

[55] “Speakeasy2”, Google Maps, https://www.google.com/maps/place/Speakeasy/@10.7309532,106.7088204,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f910786b665:0x7c1178ecd6efa743!8m2!3d10.7309532!4d106.7088204!16s%2Fg%2F11d_c0xg31?hl=en&entry=ttu (accessed Aug. 6, 2023).

[56] “Speakeasy3”, Google Maps, https://www.google.com/maps/place/Speakeasy/@10.7309532,106.7088204,17z/data=!3m1!4b1!4m6!3m5!1s0x31752f910786b665:0x7c1178ecd6efa743!8m2!3d10.7309532!4d106.7088204!16s%2Fg%2F11d_c0xg31?hl=en&entry=ttu (accessed Aug. 6, 2023).

[57] “All”, Flaticon, https://www.flaticon.com/free-icon/restaurant_1980788?term=restaurant+menu&page=1&position=4&origin=search&related_id=1980788 (accessed Aug. 6, 2023).

[58] “Breakfast”, Flaticon, https://www.flaticon.com/free-icon/english-breakfast_3480666?term=breakfast&page=1&position=1&origin=search&related_id=3480666 (accessed Aug. 6, 2023).

[59] “Lunch”, Flaticon, https://www.flaticon.com/free-icon/fast-food_738079?term=fastfood&page=1&position=2&origin=search&related_id=738079 (accessed Aug. 6, 2023).

[60] “Dinner”, Flaticon, https://www.flaticon.com/free-icon/steak_3480577?term=steak&page=1&position=31&origin=search&related_id=3480577 (accessed Aug. 6, 2023).

[61] “Coffee”, Flaticon, https://www.flaticon.com/free-icon/coffee-cup_751621?term=coffee&page=1&position=1&origin=search&related_id=751621 (accessed Aug. 6, 2023).

[62] “Pub”, Flaticon, https://www.flaticon.com/free-icon/beer_931898?term=pub&page=1&position=5&origin=search&related_id=931898 (accessed Aug. 6, 2023).