

**F *for***  
**Final Report**  
**CIS 651**  
**Mobile Application Programming**

**Instructor:**  
**Dr. Swastik Brahma**

**Designed By,**  
**Nishant Agarwal**  
**Triveni Ashok Naik**  
**(Group 20)**

# INTRODUCTION

## Introduction of App

The F-for App is an interactive application that provides the interface for the user to check the restaurants nearby, view the menu of each restaurant and then order food online by putting them in the cart.

## Why useful

The F *for* App would be a treasure for all the users because it would help them in finding the nearby restaurants. The nearby restaurants that are present would be displayed depending on their specific location, so the users can know what all restaurants are present within that area and can choose upon their interest.

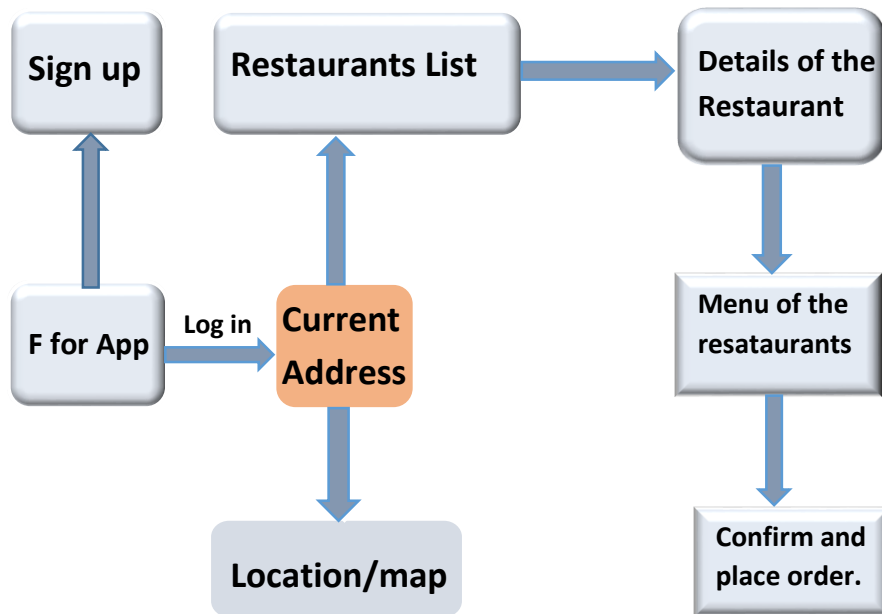
This F *for* App is user friendly and can be used by all age groups. The app will also give the location of all the restaurants so that the users would know how far are the restaurants from each other. This would be useful when the user changes their mind at the last moment and wishes to go to a different restaurant.

This will also provide with the current hours of operation. This will also provide restaurant phone numbers.

## Application Scope

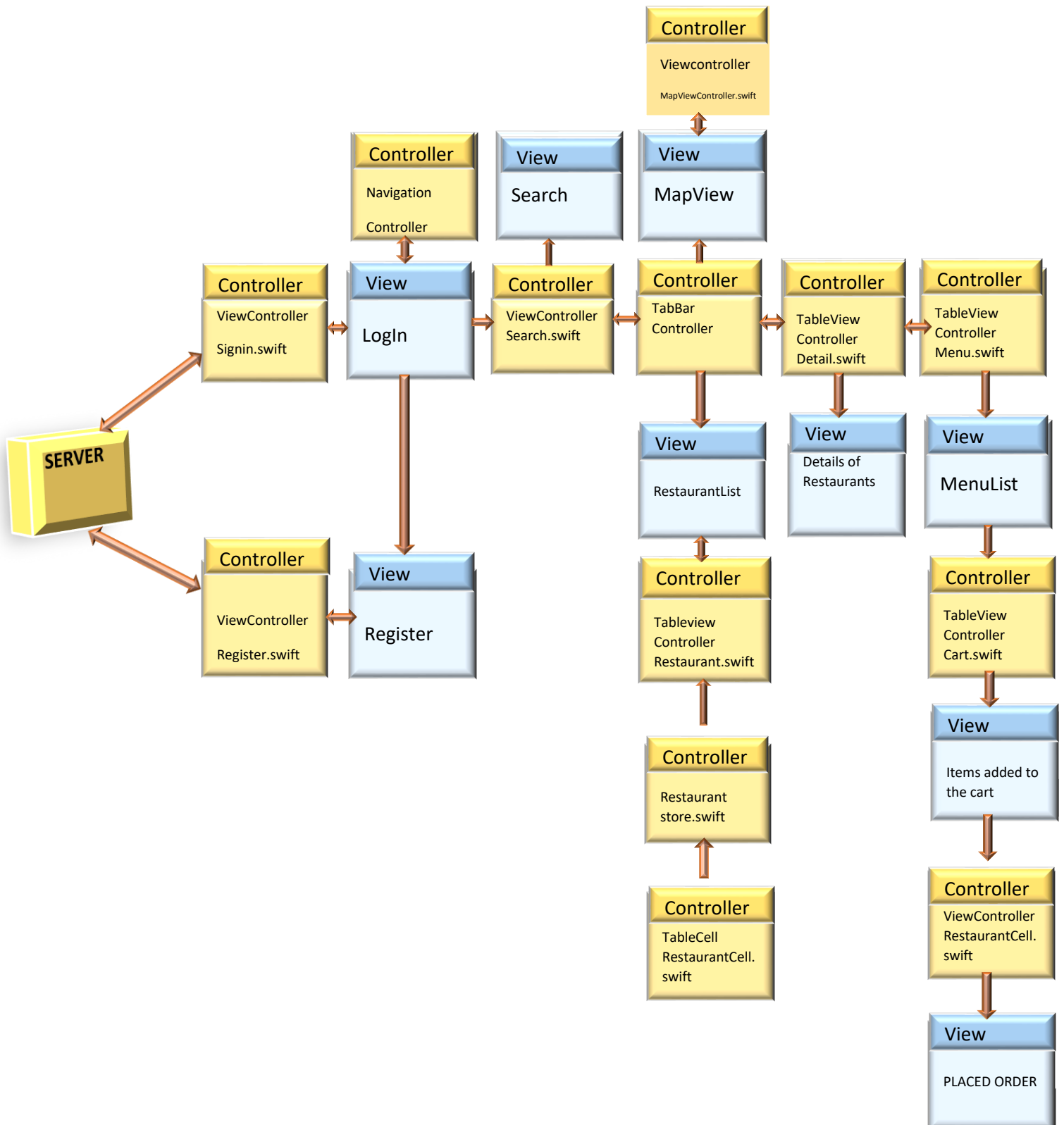
- 1) Providing an interface that allows the user to sign up using his/her own information (name, email, password) to save the information to the server and use the information to log in the App.
- 2) Alternating login using Facebook also implemented.
- 3) Providing an initial view to get the user's current location and display it.
- 4) Providing a list of restaurants that are nearby the user's location.
- 5) Providing directions to reach to the specific restaurant from their current location.
- 6) Providing the details of each restaurant like the menu, address, telephone numbers, working hours.
- 7) Providing an interface that allows users to add food items into the cart and then order them online.

# BASIC DESIGN OF *F for* APP



The above interface is just brief overview of different modules. Each module have different functionalities and attributes.

# MVC ARCHITECTURE



# TECHNICAL DETAILS

## *Web Services*

### **Zomato API**

We are using the Zomato API which gives the details of all the restaurants that are present nearby. First we need to generate the key and had to obtain the key from the Zomato Developers website through email. Once the Zomato key was generated, we used it to get all the details of the nearby restaurants. A small snippet of the code is given below.

```
let zomatoKey = "49264a3f83cada47160a024d9de4d9de45b1b"  
let urlString =  
https://developers.zomato.com/api/v2.1/search?&lat=\\\(centerLatitude\)&lon=\\\(centerLongitude\);  
let url = NSURL(string: urlString)
```

We use struct, to store all the details of the restaurant in the Dictionary

### **HTTP request and JSON**

To retrieve the information from the API, the app sends the HTTP request to the server. The data is retrieved through the API Key and is in the form of JSON. In app we use 'NSJSONSerialization' class to parse JSON to get the value we need.

```
let request = NSMutableURLRequest(URL: url!)  
request.HTTPMethod = "GET"  
request.addValue("application/json", forHTTPHeaderField: "Accept")  
request.addValue(zomatoKey, forHTTPHeaderField: "user_key")
```

To link with the server, our app sends 'GET' HTTP request to the server. When sign up, it sends all these attributes to the server. In app we use 'NSJSONSerialization' class to parse JSON to get the value we need.

Notice that the HTTP request is sent on different thread. So if we want to let main thread waiting for response, we must use 'dispatch\_sync' to block waiting thread.

# *Types Of Controllers & Delegates Used*

## **Controllers**

### **NavigationController**

The `UINavigationController` class implements a specialized view controller that manages the navigation of hierarchical content. This **navigation interface** makes it possible to present your data efficiently and makes it easier for the user to navigate that content.

We have implemented Navigationcontroller at the beginning. But since we do not want it appeared all the time, we hide it the main page controlled by TabBarController. Not only we need enough space for list of restaurants and map, but also we not want use to navigate back after log in.

### **TabBarController**

The `UITabBarController` class implements a specialized view controller that manages a radio-style selection interface. This *tab bar interface* displays tabs at the bottom of the window for selecting between the different modes and for displaying the views for that mode.

We have used TabBarController for a list view and a MapView. The list view shows all the restaurants that are present nearby and the MapView shows the locations of the nearby restaurants.

### **AlertViewController**

A `UIAlertController` object displays an alert message to the user. This class replaces the `UIActionSheet` and `UIAlertView` classes for displaying alerts. After configuring the alert controller with the actions and style you want, we present it using the `present(_:animated:completion:)` method. We have used this feature in the Cart View to confirm the items added into the cart and placed the order.

### **ActivityIndicator View**

An activity indicator is a spinning wheel that indicates a task is in the midst of being processed. If an action takes a noticeable and indeterminate amount of time to process—such as a CPU-

intensive task or connecting to a network. We have used The activity controller in Search Address Bar

## **TableviewController**

A controller object that manages a table view. We have used TableViewController for many of our views like to display the Restaurant list, Details of the Restaurants, Menu List.

## **SegueViews**

The `UIStoryboardSegue` class supports the standard visual transitions available in UIKit. You can also subclass to define custom transitions between the view controllers in your storyboard file.

Segue objects contain information about the view controllers involved in a transition. We have used this feature implement the transition from one view to another.

## Delegates

### UITableViewDelegateProtocol

The delegate of a `UITableView` object adopt the `UITableViewDelegate` protocol. Optional methods of the protocol allow the delegate to manage selections, configure section headings and footers, help to delete and reorder cells, and perform other actions.

### CLLocationManagerDelegate

Use instances of this class to establish the parameters that determine when location and heading events should be delivered, to start the delivery of those events, and stop delivery when you no longer need the location data. We have used this in our Application to get the current location/address from the user.

### MapViewDelegate

A protocol that defines a set of optional methods that you can use to receive map-related update messages. For managing the annotation views we use the `ViewForAnnotation` method to return the views associated with the specified annotation object.

### NavigationBarDelegate

The `UINavigationControllerDelegate` protocol defines optional methods that a `UINavigationController` delegate should implement to update its views when items are pushed and popped from the stack. The navigation bar represents only the bar at the top of the screen, not the view below.

### MailComposeViewControllerDelegate (FutureWork)

The `MFMailComposeViewController` class provides a standard interface for managing, editing, and sending an email message. We use this view controller to display a standard email interface inside our app. Before presenting the interface, we populate the fields with initial values for the subject, email recipients, body text, and attachments of the email. After presenting the interface, the user can edit the initial values before sending the email.



# MapView

Displays map or satellite imagery from the windows and views of your custom apps. Annotate your maps with points of interest, and determine placemark information for map coordinates. We used the MapKitFramework to provide the location details of the restaurants that are present nearby.

The MapKit framework provides an interface for embedding maps directly into your own windows and views. This framework also provides support for annotating the map, adding overlays, and performing reverse-geocoding lookups to determine placemark information for a given map coordinate.

```
extension ViewController: MKMapViewDelegate {  
  
    // 1  
    func mapView(MKMapView!, viewForAnnotation annotation: MKAnnotation!) -> MKAnnotationView! {  
        if let annotation = annotation as? Artwork {  
            let identifier = "pin"  
            var view: MKPinAnnotationView  
            if let dequeuedView = mapView.dequeueReusableAnnotationViewWithIdentifier(identifier)  
                as? MKPinAnnotationView { // 2  
                dequeuedView.annotation = annotation  
                view = dequeuedView  
            } else {  
                // 3  
                view = MKPinAnnotationView(annotation: annotation, reuseIdentifier: identifier)  
                view.canShowCallout = true  
                view.calloutOffset = CGPoint(x: -5, y: 5)  
                view.rightCalloutAccessoryView = UIButton.buttonWithType(.DetailDisclosure) as! UIView  
            }  
            return view  
        }  
        return nil  
    }  
}
```

Classes Used:

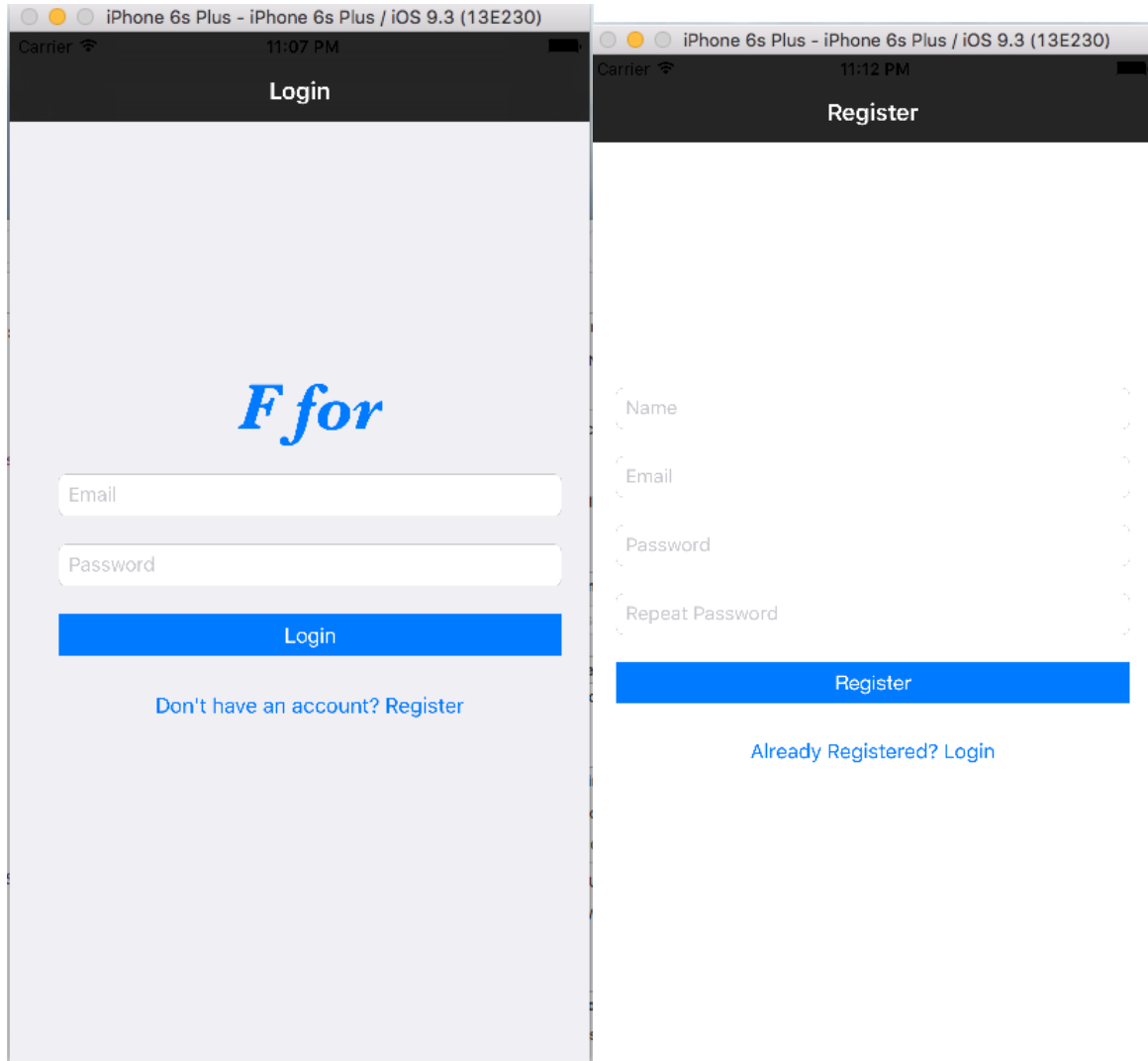
## MKAnnotationView

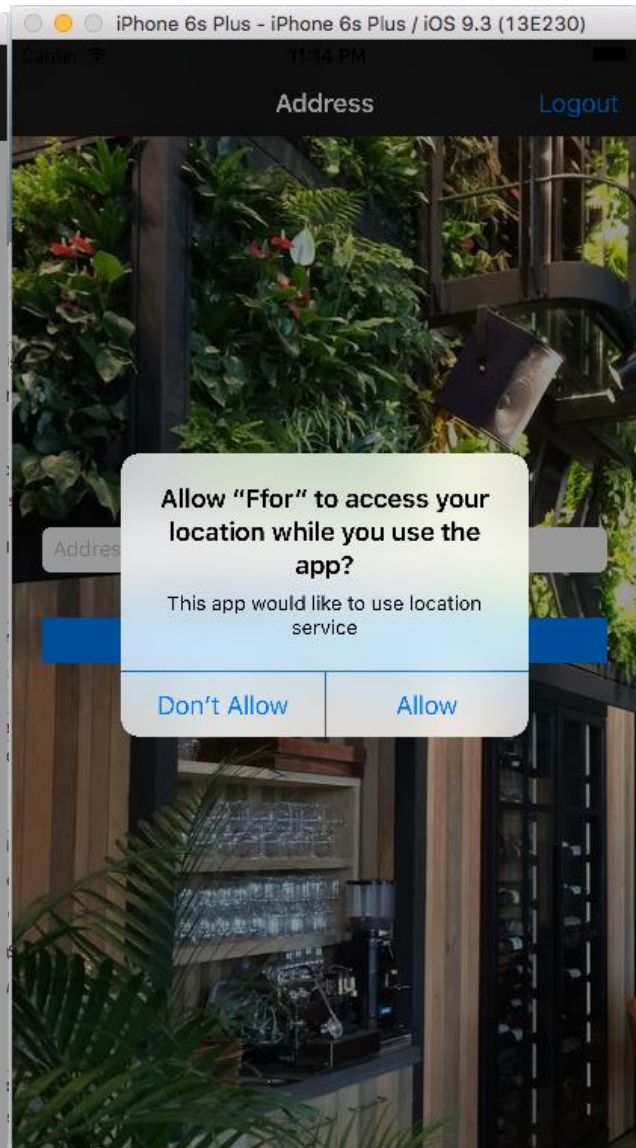
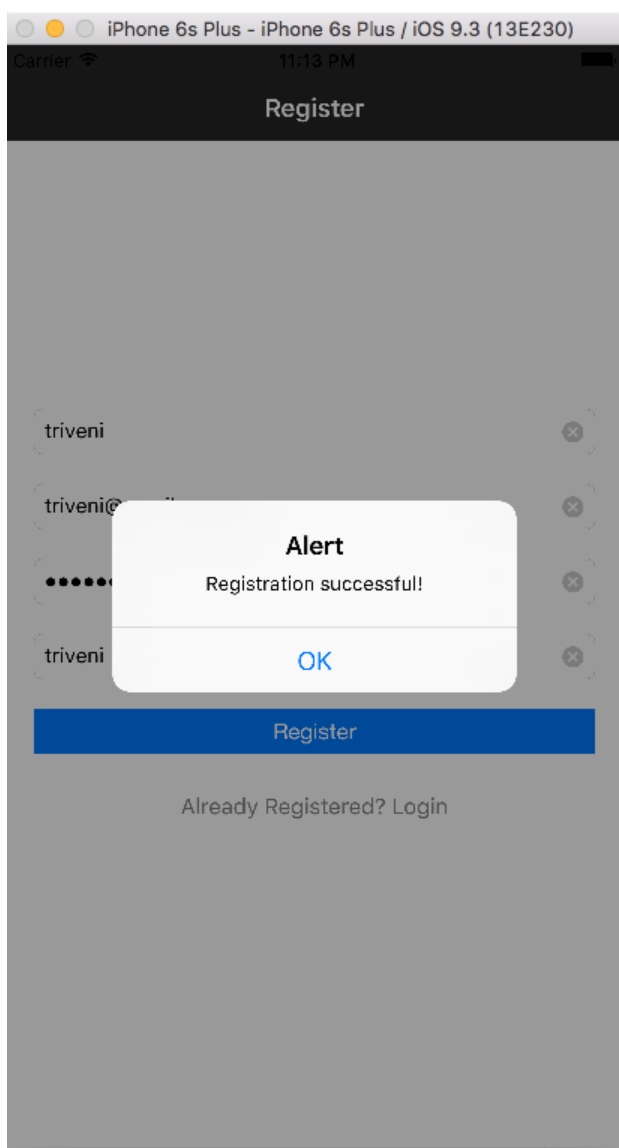
The MKAnnotationView class is responsible for presenting annotations visually in a map view. Annotation views are loosely coupled to a corresponding annotation object, which is an object that corresponds to the [MKAnnotation](#) protocol. When an annotation's coordinate point is in the visible region, the map view asks its delegate to provide a corresponding annotation view.

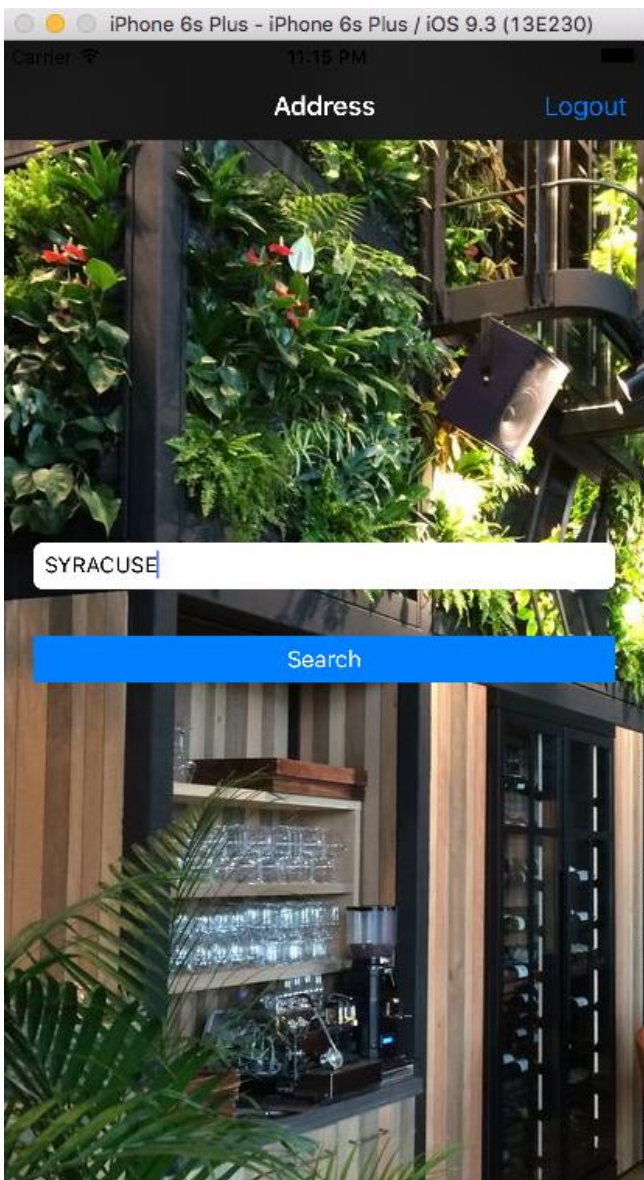
## **MKPinAnnotationView**

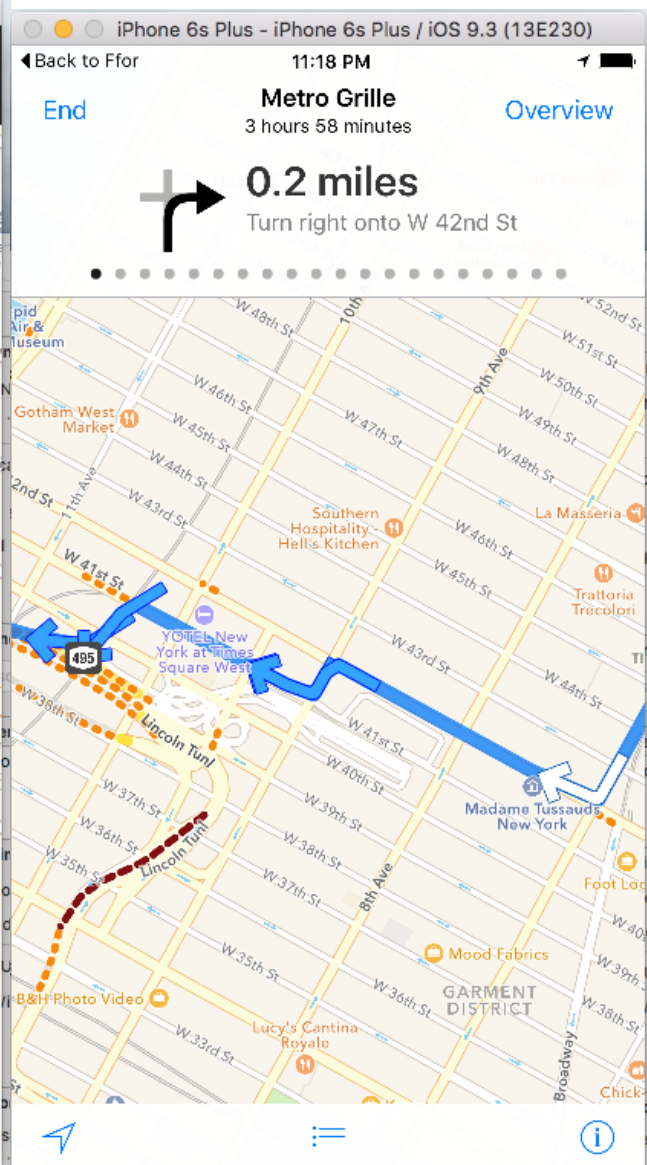
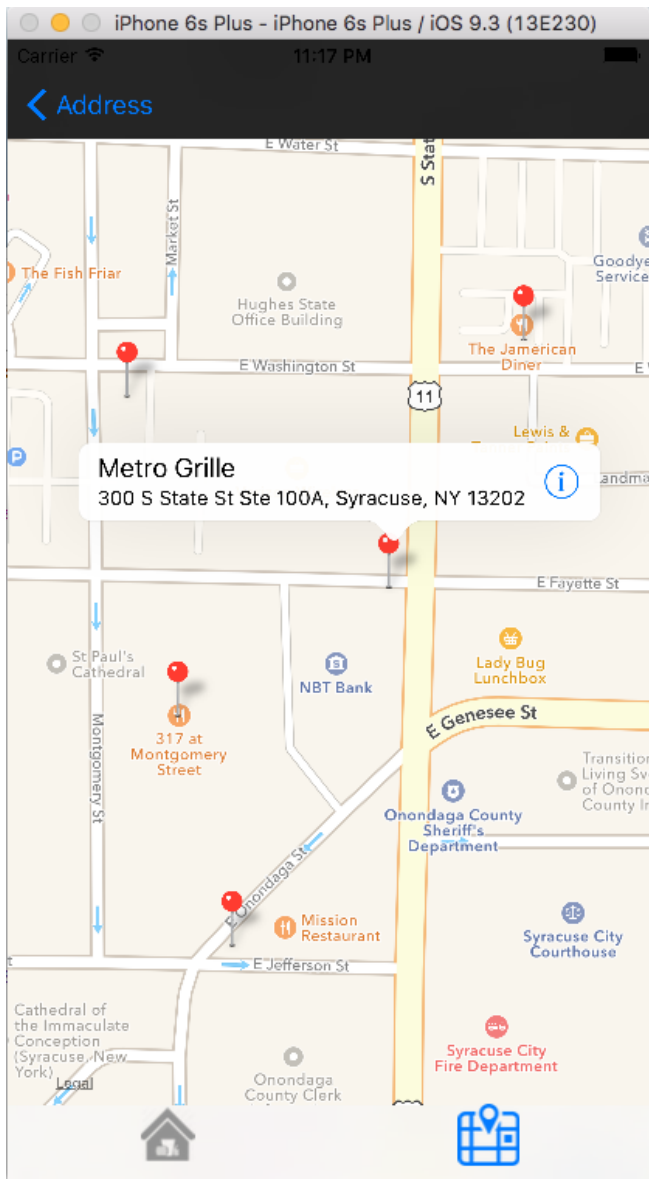
The `MKPinAnnotationView` class provides a concrete annotation view that displays a pin icon like the ones found in the Maps application. Using this class, you can configure the type of pin to drop and whether you want the pin to be animated into place

# SCREENSHOTS

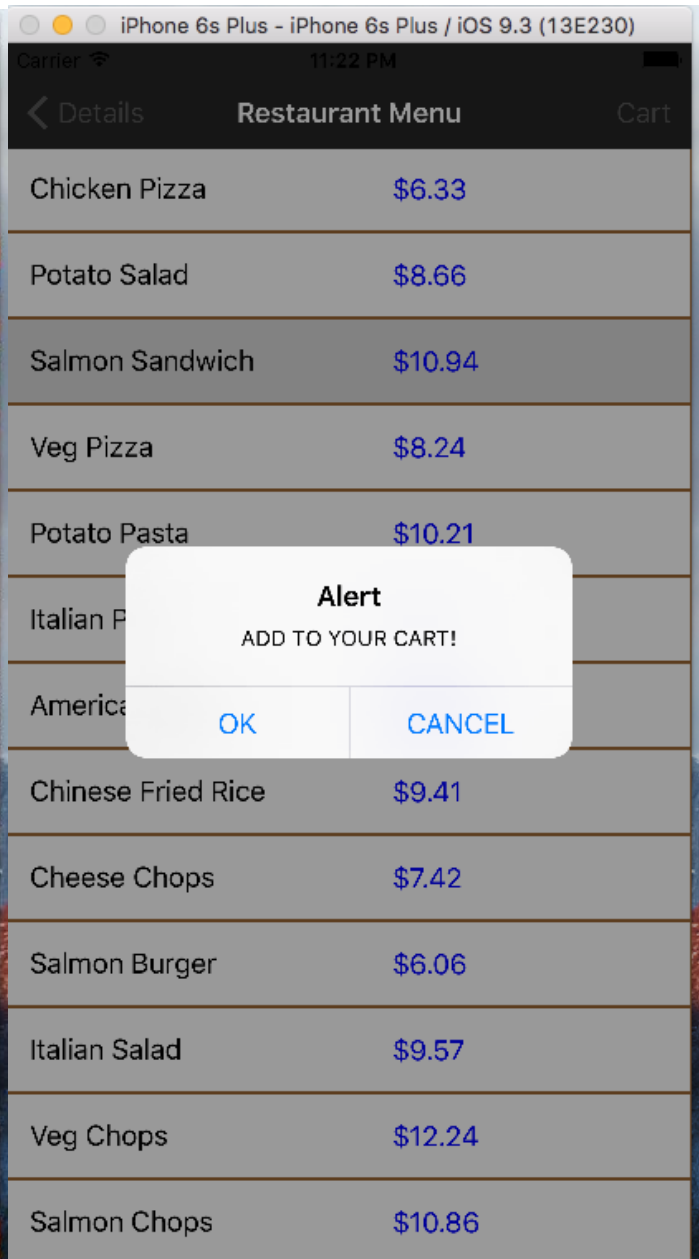
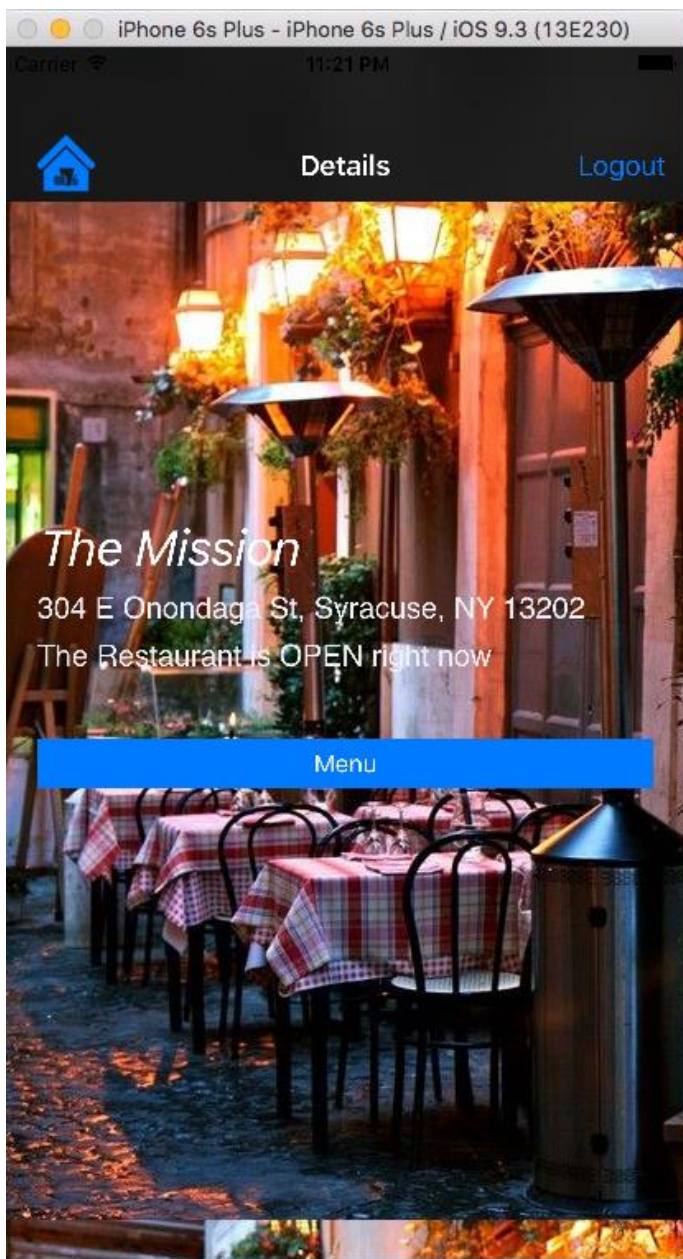












iPhone 6s Plus - iPhone 6s Plus / iOS 9.3 (13E230)

Carrier 11:22 PM

[Details](#) **Restaurant Menu** [Cart](#)

Chicken Pizza	\$6.33
Potato Salad	\$8.66
Salmon Sandwich	\$10.94
Veg Pizza	\$8.24
Potato Pasta	\$10.21
Italian Pizza	\$6.72
American Pasta	\$4.35
Chinese Fried Rice	\$9.41
Cheese Chops	\$7.42
Salmon Burger	\$6.06
Italian Salad	\$9.57
Veg Chops	\$12.24
Salmon Chops	\$10.86


iPhone 6s Plus - iPhone 6s Plus / iOS 9.3 (13E230)

Carrier 11:24 PM

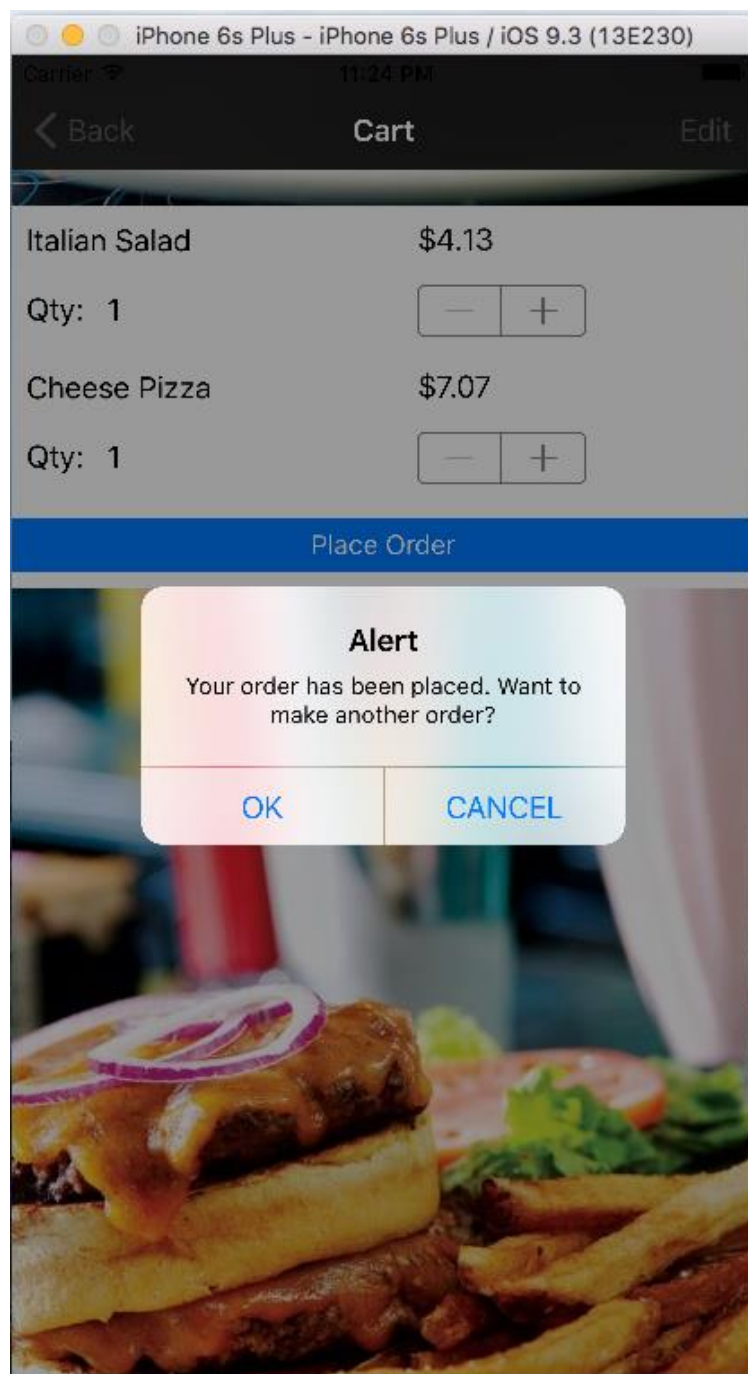
[Back](#) **Cart** [Edit](#)

Italian Salad	\$4.13
Qty: 1	<input type="button" value="-"/> <input type="button" value="+"/>
Cheese Pizza	\$7.07
Qty: 1	<input type="button" value="-"/> <input type="button" value="+"/>

[Place Order](#)







# Summary & Lessons Learned

First we should emphasize that the F for App is an interactive application which is designed mainly for users who are new to a particular area. The main function of the F for App is to let the users to find all the nearby restaurants and would allow them to have a look at the menu of each restaurant. This would make their life more easier as they can choose from F for App as in to which restaurant to go depending on their interests.

Second we say that the F for App is almost a mix of every part knowledge we learned this semester. It is really a good chance to review what we have learn before as well as a good chance to learn something new. For homework we may just need to review the class content and read the source code uploaded by the instructor on the blackboard and finish it. Because the architecture is given. But for the F for App, we started by ourselves from the beginning.

We built the architecture, search the internet every time for the remaining and new problems, ran the App every time to correct the logic and try to make it better through upgrading the code. While this is not the real industry software engineering. But we have learned a lot from the whole procedure and acquired a lot of precious experience for our future life.

But F for App is not perfect application, it still has a lot of part to supplement to make it even better. We can add more functions to the App but one idea is important from the beginning that F for App is a App to make some certain part better. We still have to and we will focus on the part where the users can also select the food they want, put them into the cart and then order them online.

## Future Work

- Currently the map does not provide the exact directions from one restaurant to another, we would try to update this feature in the upcoming days.
- We want to make the App useful for all the other users by adding an option of sending email to the restaurant owner with all the customer's credentials and the list of the ordered food in the mail.
- We thought of implementing it using the `MFMailComposeViewController` protocol. We could not implement this due to security reasons. We are planning to look into an alternative method to implement this feature.

## Challenges Faced

- We had an idea of sending the user's details and the ordered food list to the restaurant owner through email-id. We thought of implementing it using the `MFMailComposeViewController`. But there was some issues faced with Apple's Security because the login credentials could not be hardcoded in the code. So we had to think of an alternative for this.
- Zomato API did not return thumb images of the restaurants for some locations. Hence we downloaded the images from the internet and used them for the display.

