PROJECT REPORT

on

iOS App

ChatterBox

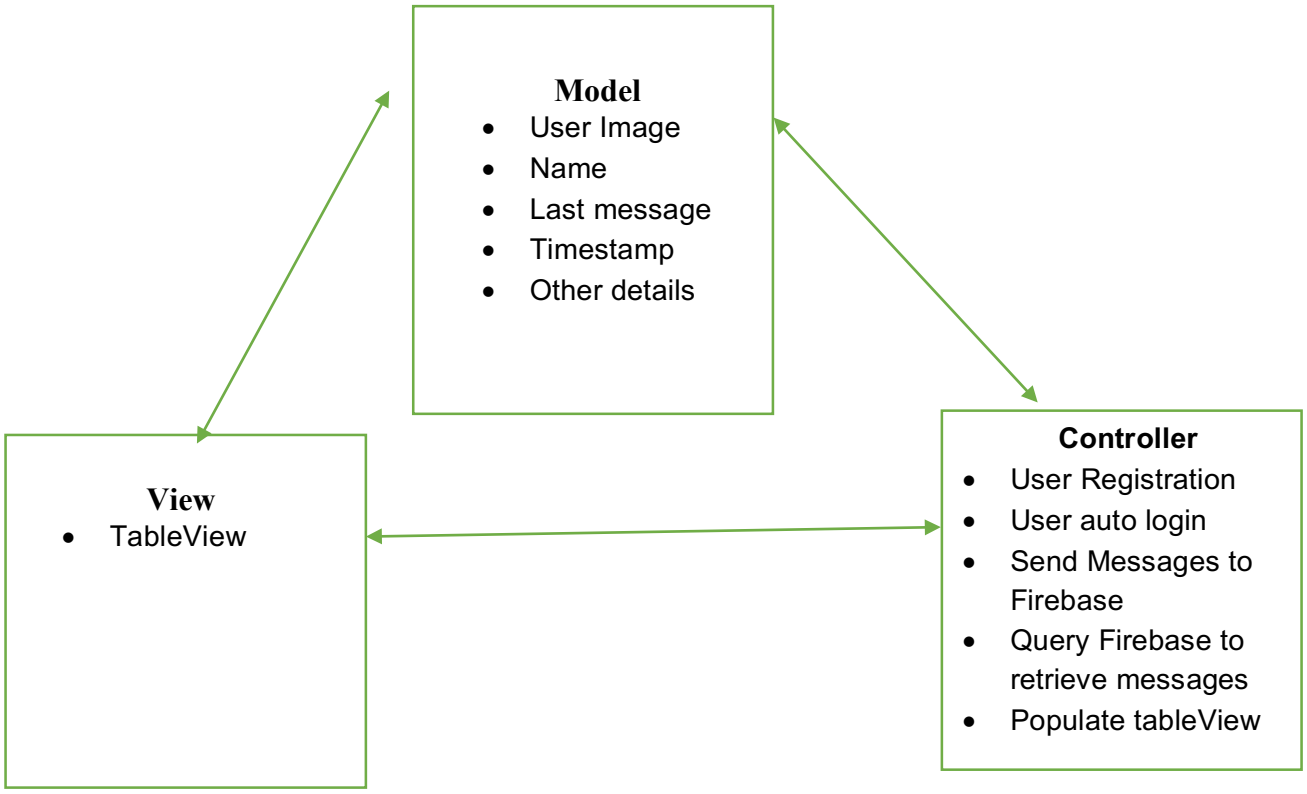
Introduction

- ChatterBox is an iOS app that lets uses chat with other users of the application via text messages, picture messages and location messages.
- The app uses Backendless a MBaaS provider to store user credentials that allows the user to register for the first time and login the user.
- A cloud engine called Firebase is be used to store the user messages a user send to other users.

Usefulness of Chatterbox

- The popularity of messaging apps changes depending on the country. WhatsApp's pitch is simple: it looks like any old text messaging app, but you can create group chats, send photos, videos, voice.
- Chatterbox is a Real-time iOS chat application similar to WhatsApp. It enables user to communicate with others in his contact list through text messages, pictures, videos and their current location. This will also have offline application functionality.

Model View Controller (MVC)



Technical Details

For this project : I've used the following :

- Backendless MBaaS
- Firebase

- JSQMessgeViewController
- MapKit
- UITableViewController
- UITabBarController
- UINavigationController

Backendless MBaaS :

- Mobile backend as a service (MBaaS), also known as "backend as a service" (BaaS), is a model for providing web app and mobile app developers with a way to link their applications to backend cloud storage and APIs exposed by back end applications while also providing features such as user management, push notifications, and integration with social networking services.[4] These services are provided via the use of custom software development kits (SDKs) and application programming interfaces (APIs).

JSQMessageViewController :

- JSQMessageViewController is a messaging UI developed by Jesse Squires and team. It mimics the iOS messaging app by apple.

FIREBASE

Firebase is a web and mobile application platform used for developing apps and storing data.In our app we are using the backendless service to authenticate the users when they login and can also create an account from this service. We are using the backendless database to store and retrieve data to our app. The Firebase database provides real time database and backend service.

Adding Firebase, Backendless, JSQMessageViewController to our app:

1. Create a Firebase project in the firebase console.Click Add firebase to your ios app.
2. Enter the app's bundle ID when prompted.
3. Download the GoogleService-Info.plist file and place it in the project folder.
4. Create an account in backendless website and create a project with your app name.
5. Note down the app ID and secret key.
6. Create a podfile in the project directory.
   $ cd your-project directory

   $ touch Podfile

7. Add the required pod files and install the podfile.
   source "https://github.com/CocoaPods/Old-Specs"
   pod 'JSQMessagesViewController','~>7.3.3'
   pod 'IDMPhotoBrowser','~>1.8.4'
   pod 'Firebase/Core'
   pod 'Firebase/Database'
   pod 'Backendless', '3.0.29'
   pod 'MBProgressHUD', '0.9.2'
   Install the pods and open the .xcworkspace file.
8. The final step is to add initialization code to the application.
   code:
   Import Firebase
   In AppDelegate,
   add : FIRApp.configure()

   Import backendless and call the function

   backendless.initApp(APP_ID, secret: SECRET_KEY, version: VERSION_NUM)

Authenticating with backendless

1. After importing and setting up backendless, we can register the user for the first time as follows :

```
 backendless.userService.registering(newUser, response: { (registeredUser :
BackendlessUser!) in

        //User login
        ProgressHUD.dismiss()
        self.loginUser(email, username: username, password: password)


    }) { (fault: Fault!) in       print("Server error! \(fault)" }}
```

2. Similarly we can use the login method to login the user.

MapKit

We are using Mapkit in our project to display the location of the user when the user sends s location message.A pin lands on the mapkit and displays the address of the user .
Adding Mapkit to app in Xcode

1. In Main.StoryBoard, drag a Mapkit to the scene.
2. Set all the constraints and resolve all the auto layout issues.
3. Import the Mapkit framework to the project.
4. Create an outlet for the mapview in the viewcontroller.
5. In our project we display the view in the location of the user with the position of the pin pointing to user location.

UITableViewController

This creates a controller that manages a table view.In our project we use the table view to display the search results for the restaurants that are available from the api and also to display the favorite restaurants stored and also to store the receipes that the user adds.

UITabBarController

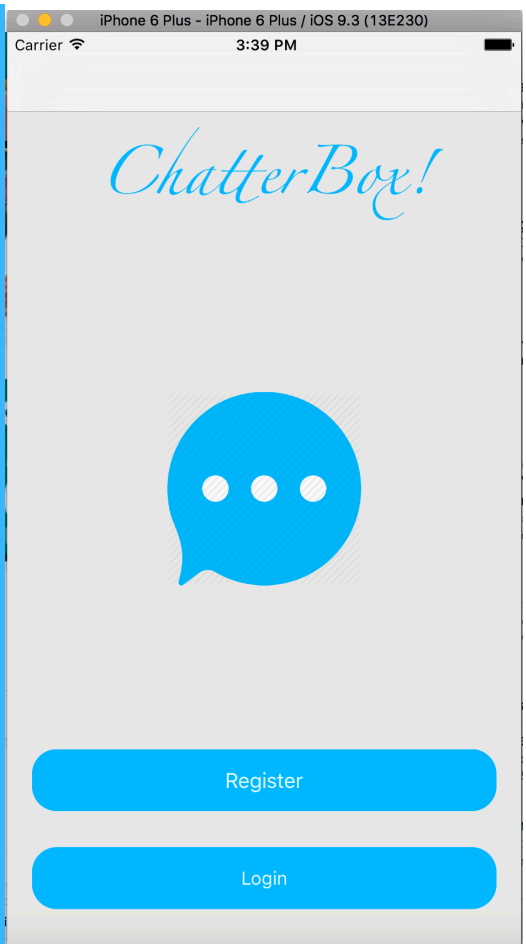
This is a controller that manages a radio style interface.The interface displays the tabs at the bottom of the screen.Each tab has a different view controller .
In our project we use the UITabBarController to display different tabs such as
Recent chats and settings.

UINavigationController:

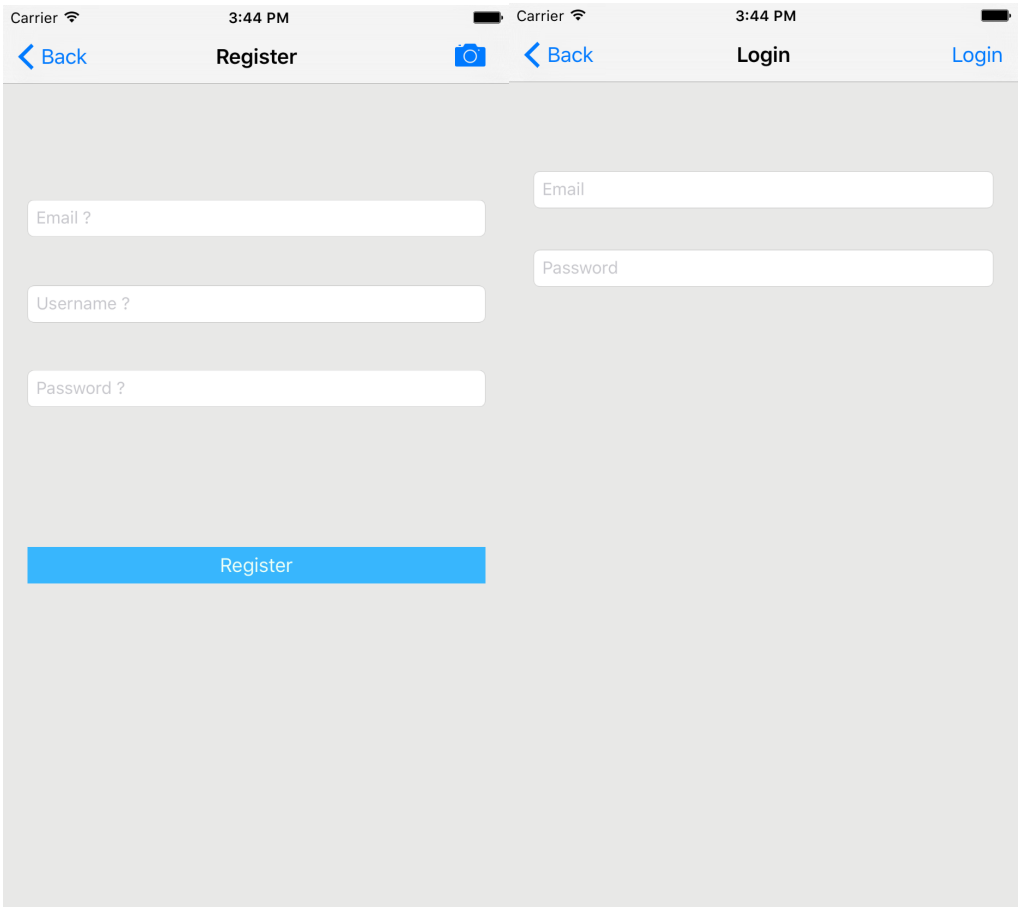
The UINavigationController implements a specialized view controller that manages the navigation of hierarchical content

Screenshots

Splash Screen

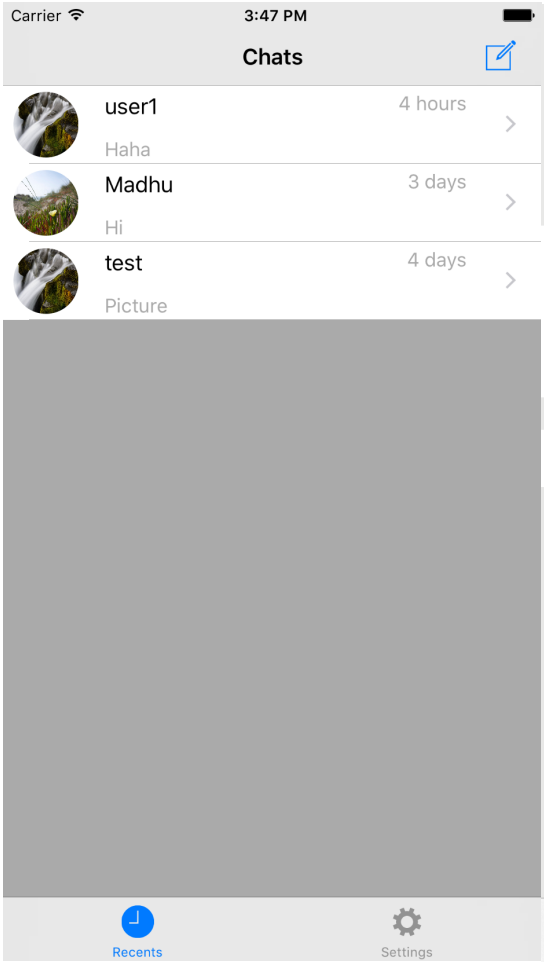


Welcome Screen

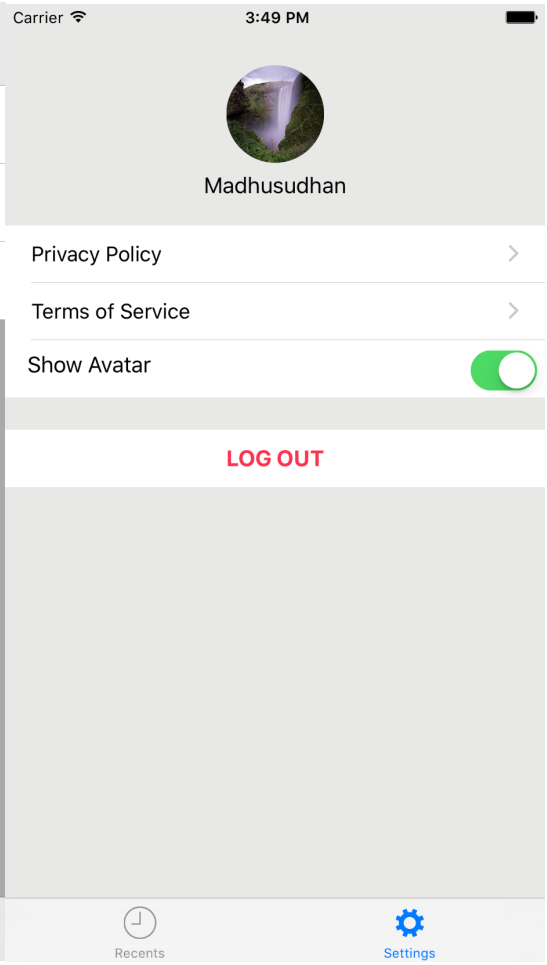


Register Screen

Login Screen

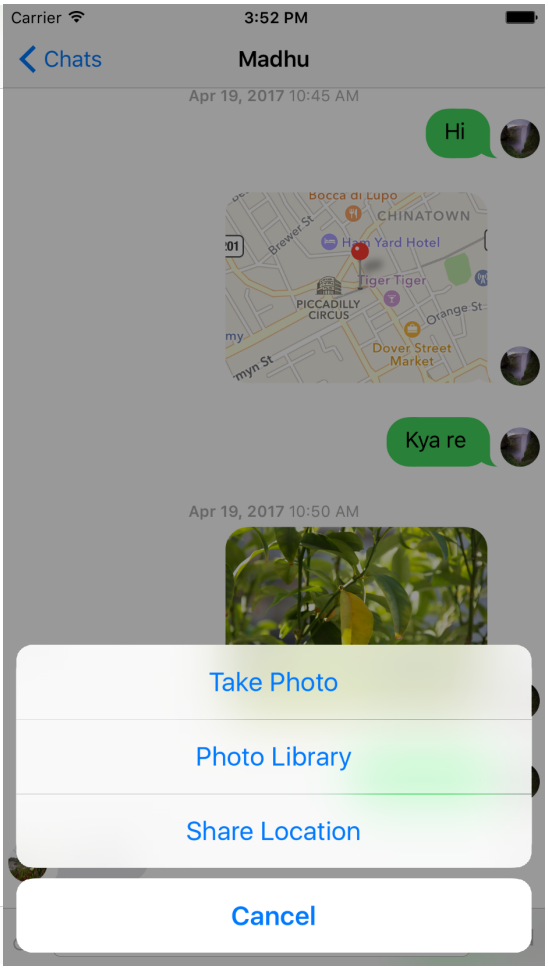


Chats screen

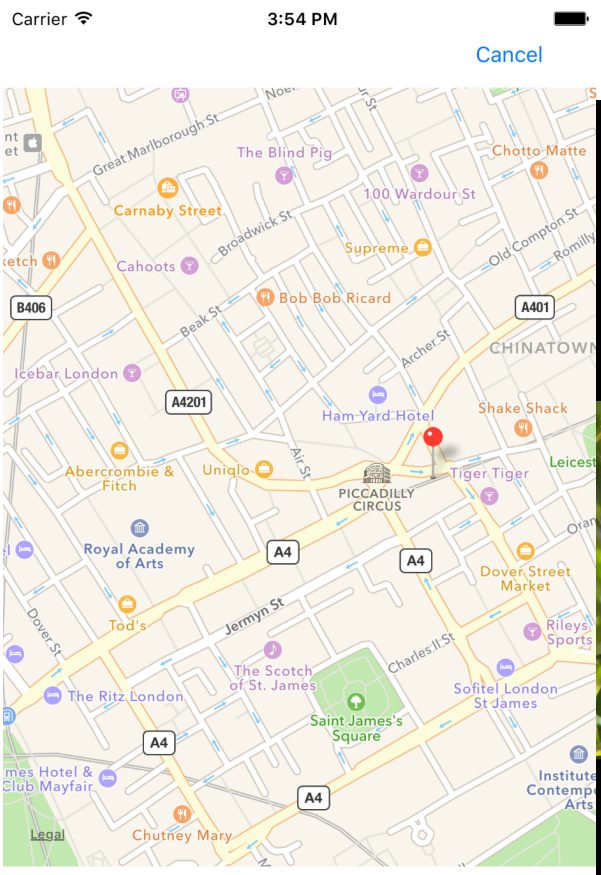


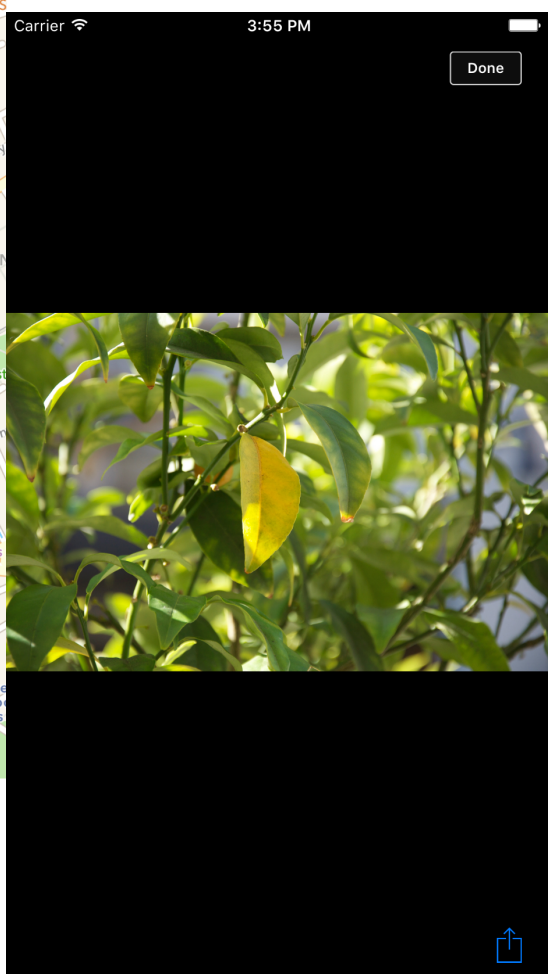
Settings screen

Individual Chat Screen



Utilities inside Chats





User location

Picture Messages

Future Work

- To update the app to iOS 10 and swift 3.
- To add social media login such as google and facebook
- Add the ability to send push notifications.
- Add more features such as group conversations, voice calling.

References

https://developer.apple.com/reference
https://firebase.google.com/docs/
https://backendless.com/products/documentation/