Hi-res profile photo:

Preferred Name: Nima Motieifrd

Program Taken: iOS Development

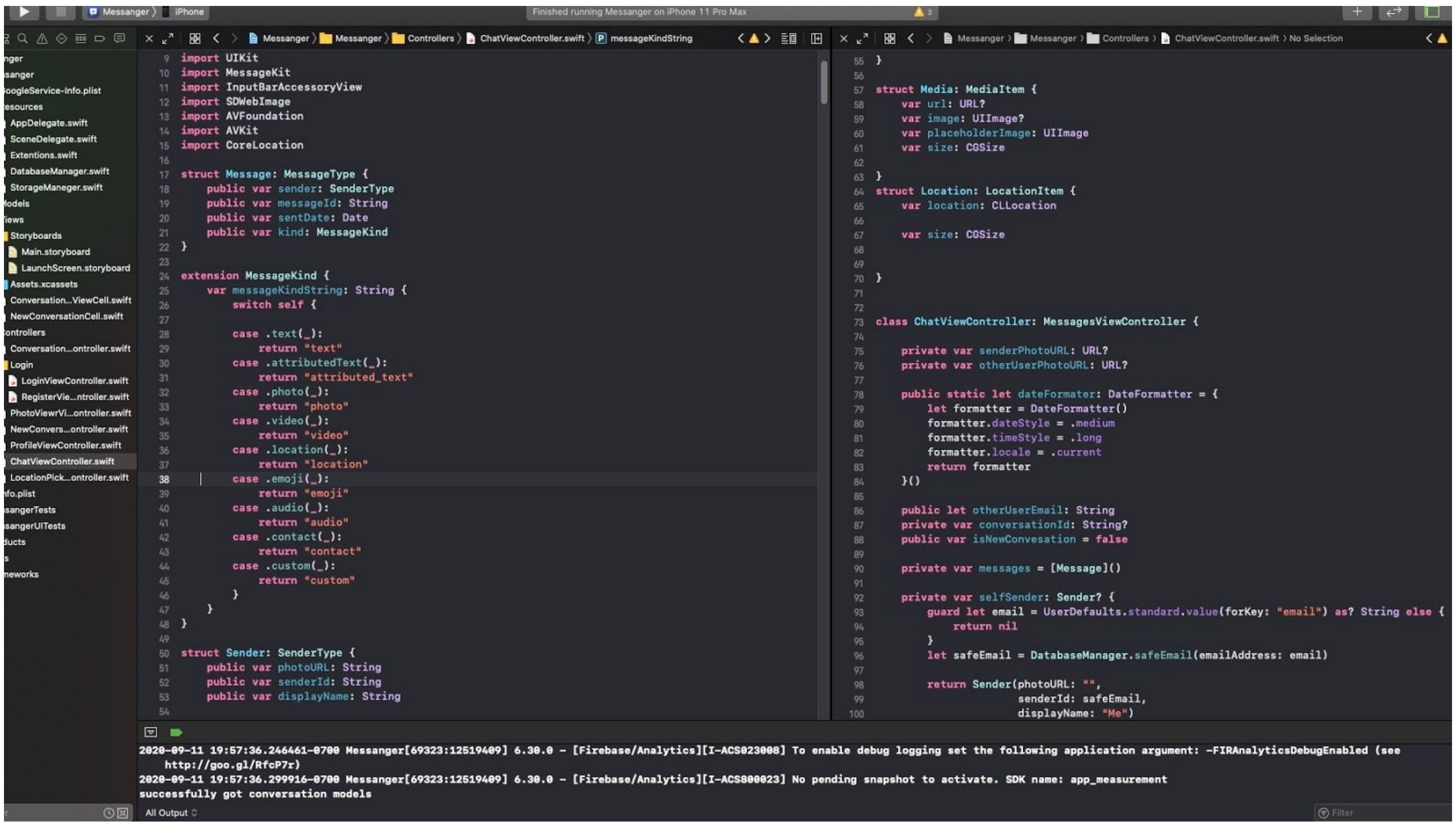
Project Name: Real Time Messenger

Brief summary:
The main goal for this project was to develop a real-time Messenger application using all the information provided by the instructor during the 5 week-long course. The project includes more than 10 different swift files which are each responsible for a specific section of the app. one of the most essential files is the “DatabaseManager.swift” which connects the app to the Firebase database where all the authentication is being handled. This application was designed and developed similar to many popular chat applications such as Facebook and WhatsApp.

Features:

- Text messages
- Picture and video messages
- Email/pass registration and login
- Deleting messages
- Search for other users

ChatViewController.swift


Messanger 〉 iPhone
Finished running Messanger on iPhone 11 Pro Max
Messanger 〉 Messanger 〉 Controllers 〉 ChatViewController.swift 〉 messageKindString
Messanger 〉 Messanger 〉 Controllers 〉 ChatViewController.swift 〉 No Selection
All Output
Filter

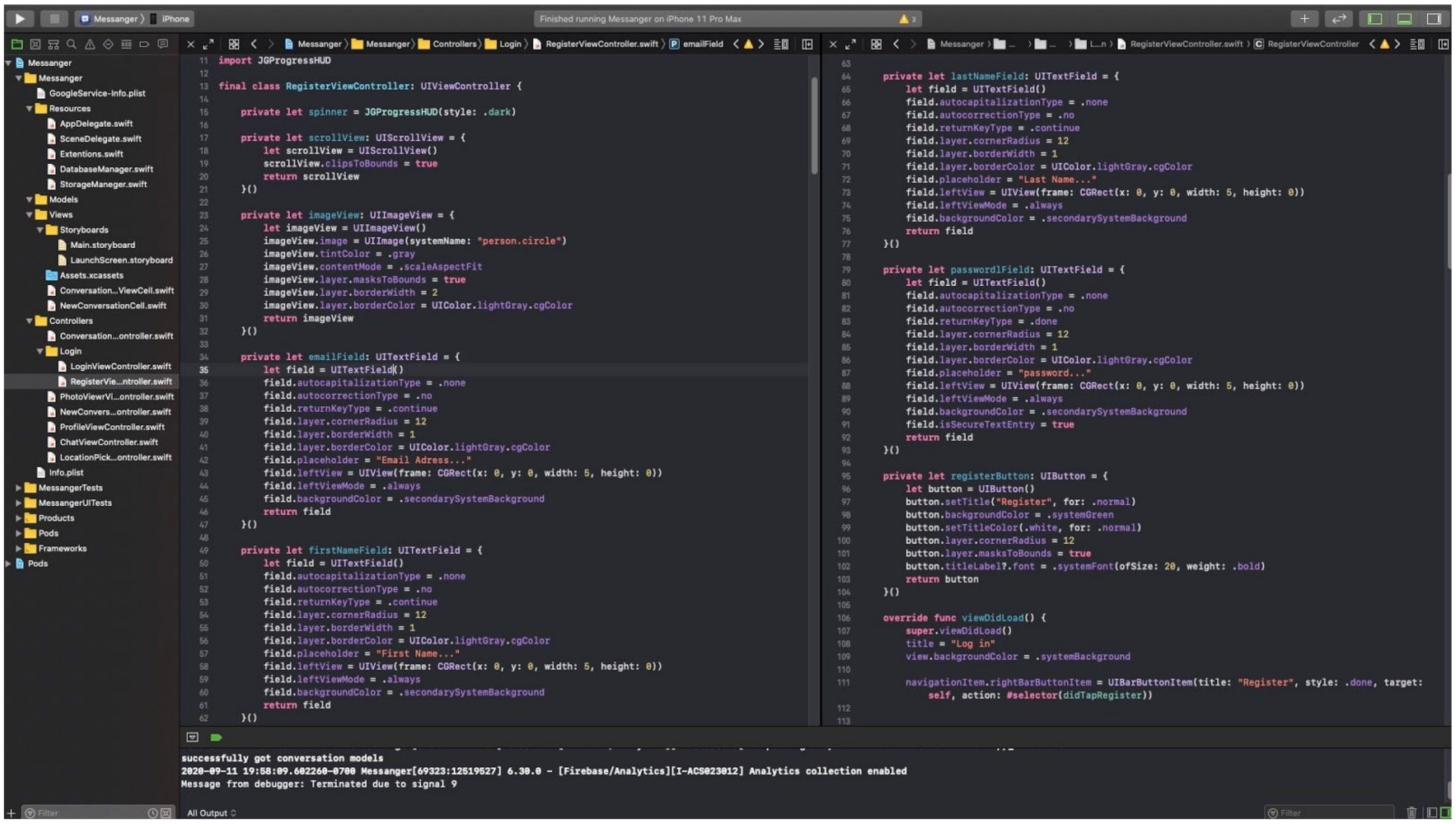
```swift
import UIKit
import MessageKit
import InputBarAccessoryView
import SDWebImage
import AVFoundation
import AVKit
import CoreLocation

struct Message: MessageType {
    public var sender: SenderType
    public var messageId: String
    public var sentDate: Date
    public var kind: MessageKind
}

extension MessageKind {
    var messageKindString: String {
        switch self {

        case .text(_):
            return "text"
        case .attributedText(_):
            return "attributed_text"
        case .photo(_):
            return "photo"
        case .video(_):
            return "video"
        case .location(_):
            return "location"
        case .emoji(_):
            return "emoji"
        case .audio(_):
            return "audio"
        case .contact(_):
            return "contact"
        case .custom(_):
            return "custom"
        }
    }
}

struct Sender: SenderType {
    public var photoURL: String
    public var senderId: String
    public var displayName: String

```

```swift
}

struct Media: MediaItem {
    var url: URL?
    var image: UIImage?
    var placeholderImage: UIImage
    var size: CGSize

}
struct Location: LocationItem {
    var location: CLLocation

    var size: CGSize


}


class ChatViewController: MessagesViewController {

    private var senderPhotoURL: URL?
    private var otherUserPhotoURL: URL?

    public static let dateFormater: DateFormatter = {
        let formatter = DateFormatter()
        formatter.dateStyle = .medium
        formatter.timeStyle = .long
        formatter.locale = .current
        return formatter
    }()

    public let otherUserEmail: String
    private var conversationId: String?
    public var isNewConvesation = false

    private var messages = [Message]()

    private var selfSender: Sender? {
        guard let email = UserDefaults.standard.value(forKey: "email") as? String else {
            return nil
        }
        let safeEmail = DatabaseManager.safeEmail(emailAddress: email)

        return Sender(photoURL: "",
                      senderId: safeEmail,
                      displayName: "Me")
```

```
2020-09-11 19:57:36.246461-0700 Messanger[69323:12519409] 6.30.0 - [Firebase/Analytics][I-ACS023008] To enable debug logging set the following application argument: -FIRAnalyticsDebugEnabled (see
    http://goo.gl/RfcP7r)
2020-09-11 19:57:36.299916-0700 Messanger[69323:12519409] 6.30.0 - [Firebase/Analytics][I-ACS800023] No pending snapshot to activate. SDK name: app_measurement
successfully got conversation models
```

This swift file is responsible for displaying all the messages, the senders and receivers name on the Chat View.

RegisterViewController.swift

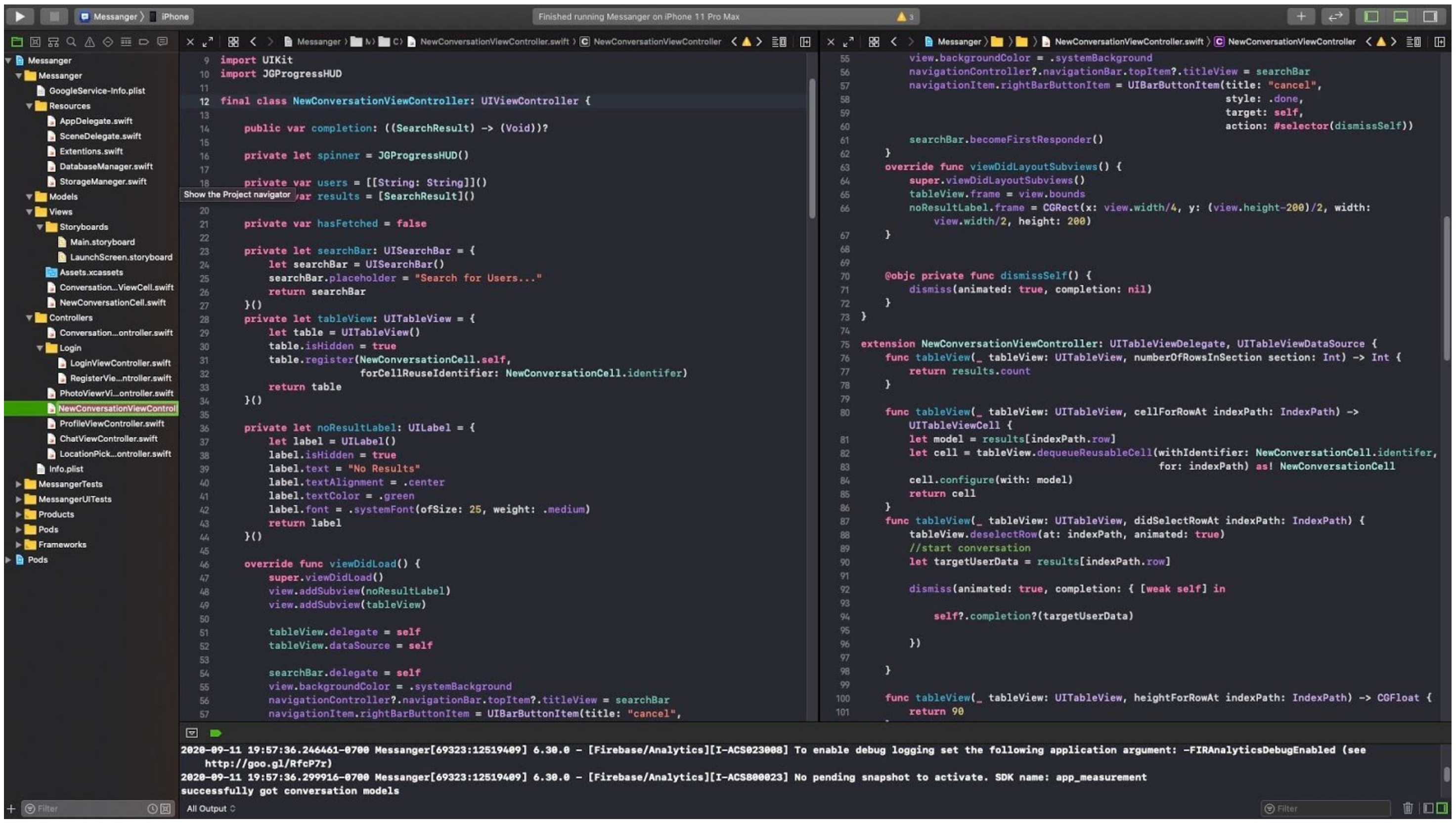


```swift
import JGProgressHUD

final class RegisterViewController: UIViewController {

    private let spinner = JGProgressHUD(style: .dark)

    private let scrollView: UIScrollView = {
        let scrollView = UIScrollView()
        scrollView.clipsToBounds = true
        return scrollView
    }()

    private let imageView: UIImageView = {
        let imageView = UIImageView()
        imageView.image = UIImage(systemName: "person.circle")
        imageView.tintColor = .gray
        imageView.contentMode = .scaleAspectFit
        imageView.layer.masksToBounds = true
        imageView.layer.borderWidth = 2
        imageView.layer.borderColor = UIColor.lightGray.cgColor
        return imageView
    }()

    private let emailField: UITextField = {
        let field = UITextField()
        field.autocapitalizationType = .none
        field.autocorrectionType = .no
        field.returnKeyType = .continue
        field.layer.cornerRadius = 12
        field.layer.borderWidth = 1
        field.layer.borderColor = UIColor.lightGray.cgColor
        field.placeholder = "Email Adress..."
        field.leftView = UIView(frame: CGRect(x: 0, y: 0, width: 5, height: 0))
        field.leftViewMode = .always
        field.backgroundColor = .secondarySystemBackground
        return field
    }()

    private let firstNameField: UITextField = {
        let field = UITextField()
        field.autocapitalizationType = .none
        field.autocorrectionType = .no
        field.returnKeyType = .continue
        field.layer.cornerRadius = 12
        field.layer.borderWidth = 1
        field.layer.borderColor = UIColor.lightGray.cgColor
        field.placeholder = "First Name..."
        field.leftView = UIView(frame: CGRect(x: 0, y: 0, width: 5, height: 0))
        field.leftViewMode = .always
        field.backgroundColor = .secondarySystemBackground
        return field
    }()

    private let lastNameField: UITextField = {
        let field = UITextField()
        field.autocapitalizationType = .none
        field.autocorrectionType = .no
        field.returnKeyType = .continue
        field.layer.cornerRadius = 12
        field.layer.borderWidth = 1
        field.layer.borderColor = UIColor.lightGray.cgColor
        field.placeholder = "Last Name..."
        field.leftView = UIView(frame: CGRect(x: 0, y: 0, width: 5, height: 0))
        field.leftViewMode = .always
        field.backgroundColor = .secondarySystemBackground
        return field
    }()

    private let passwordlField: UITextField = {
        let field = UITextField()
        field.autocapitalizationType = .none
        field.autocorrectionType = .no
        field.returnKeyType = .done
        field.layer.cornerRadius = 12
        field.layer.borderWidth = 1
        field.layer.borderColor = UIColor.lightGray.cgColor
        field.placeholder = "password..."
        field.leftView = UIView(frame: CGRect(x: 0, y: 0, width: 5, height: 0))
        field.leftViewMode = .always
        field.backgroundColor = .secondarySystemBackground
        field.isSecureTextEntry = true
        return field
    }()

    private let registerButton: UIButton = {
        let button = UIButton()
        button.setTitle("Register", for: .normal)
        button.backgroundColor = .systemGreen
        button.setTitleColor(.white, for: .normal)
        button.layer.cornerRadius = 12
        button.layer.masksToBounds = true
        button.titleLabel?.font = .systemFont(ofSize: 20, weight: .bold)
        return button
    }()

    override func viewDidLoad() {
        super.viewDidLoad()
        title = "Log in"
        view.backgroundColor = .systemBackground

        navigationItem.rightBarButtonItem = UIBarButtonItem(title: "Register", style: .done, target:
            self, action: #selector(didTapRegister))

```

```
successfully got conversation models
2020-09-11 19:58:09.602260-0700 Messanger[69323:12519527] 6.30.0 - [Firebase/Analytics][I-ACS023012] Analytics collection enabled
Message from debugger: Terminated due to signal 9
```

The Register View Controller subclasses the UIViewController and is responsible for registering a user into the database.

NewConversationViewController.swift

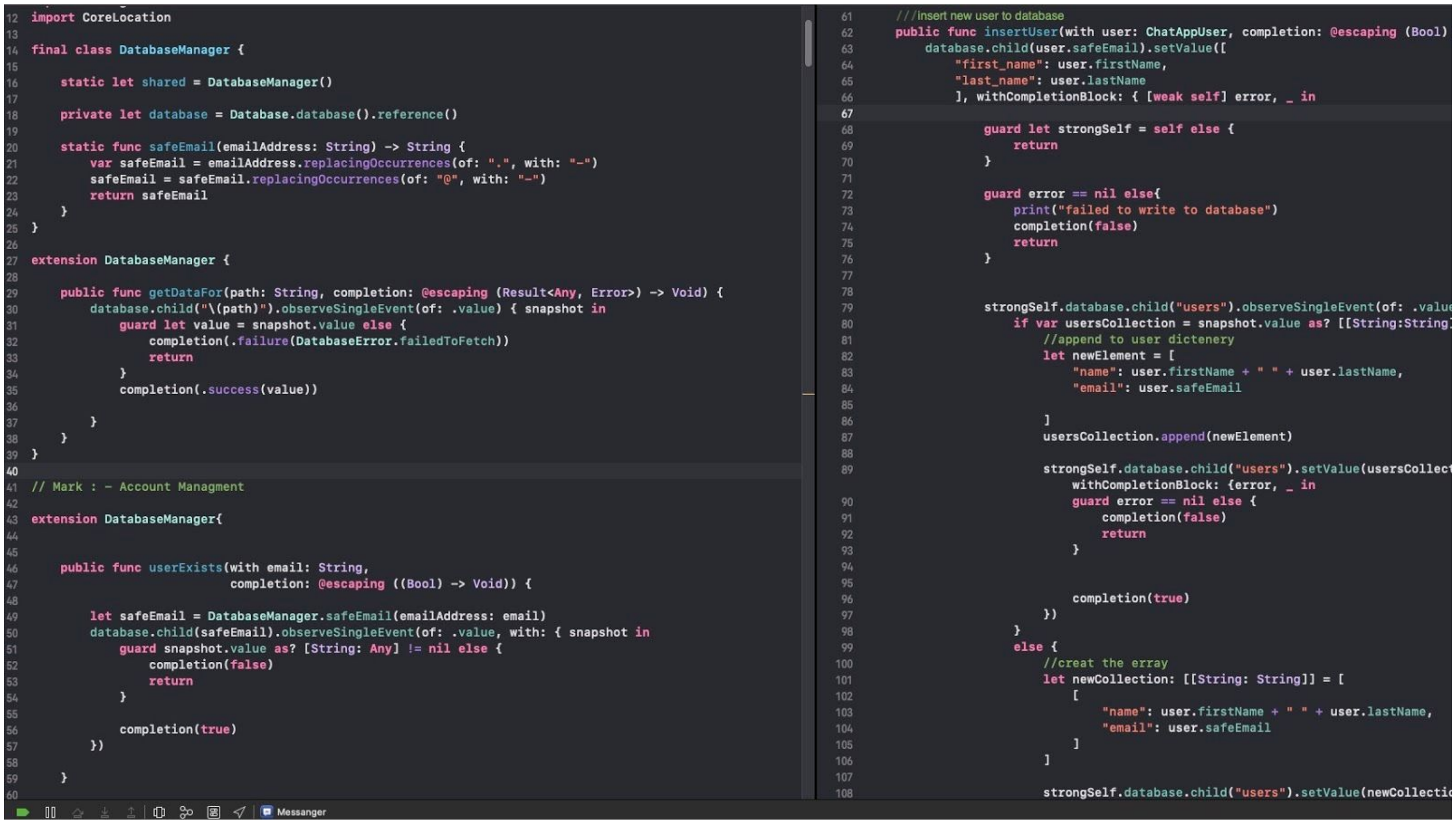

Messanger
iPhone
Finished running Messanger on iPhone 11 Pro Max
NewConversationViewController.swift
NewConversationViewController
Show the Project navigator
All Output
Filter

```swift
import UIKit
import JGProgressHUD

final class NewConversationViewController: UIViewController {

    public var completion: ((SearchResult) -> (Void))?

    private let spinner = JGProgressHUD()

    private var users = [[String: String]]()
           var results = [SearchResult]()

    private var hasFetched = false

    private let searchBar: UISearchBar = {
        let searchBar = UISearchBar()
        searchBar.placeholder = "Search for Users..."
        return searchBar
    }()
    private let tableView: UITableView = {
        let table = UITableView()
        table.isHidden = true
        table.register(NewConversationCell.self,
                       forCellReuseIdentifier: NewConversationCell.identifer)
        return table
    }()

    private let noResultLabel: UILabel = {
        let label = UILabel()
        label.isHidden = true
        label.text = "No Results"
        label.textAlignment = .center
        label.textColor = .green
        label.font = .systemFont(ofSize: 25, weight: .medium)
        return label
    }()

    override func viewDidLoad() {
        super.viewDidLoad()
        view.addSubview(noResultLabel)
        view.addSubview(tableView)

        tableView.delegate = self
        tableView.dataSource = self

        searchBar.delegate = self
        view.backgroundColor = .systemBackground
        navigationController?.navigationBar.topItem?.titleView = searchBar
        navigationItem.rightBarButtonItem = UIBarButtonItem(title: "cancel",
                                                            style: .done,
                                                            target: self,
                                                            action: #selector(dismissSelf))
        searchBar.becomeFirstResponder()
    }
    override func viewDidLayoutSubviews() {
        super.viewDidLayoutSubviews()
        tableView.frame = view.bounds
        noResultLabel.frame = CGRect(x: view.width/4, y: (view.height-200)/2, width:
            view.width/2, height: 200)
    }


    @objc private func dismissSelf() {
        dismiss(animated: true, completion: nil)
    }
}

extension NewConversationViewController: UITableViewDelegate, UITableViewDataSource {
    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return results.count
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
        UITableViewCell {
        let model = results[indexPath.row]
        let cell = tableView.dequeueReusableCell(withIdentifier: NewConversationCell.identifer,
                                                 for: indexPath) as! NewConversationCell
        cell.configure(with: model)
        return cell
    }
    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        tableView.deselectRow(at: indexPath, animated: true)
        //start conversation
        let targetUserData = results[indexPath.row]

        dismiss(animated: true, completion: { [weak self] in

            self?.completion?(targetUserData)

        })

    }

    func tableView(_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat {
        return 90
```

```
2020-09-11 19:57:36.246461-0700 Messanger[69323:12519409] 6.30.0 - [Firebase/Analytics][I-ACS023008] To enable debug logging set the following application argument: -FIRAnalyticsDebugEnabled (see
    http://goo.gl/RfcP7r)
2020-09-11 19:57:36.299916-0700 Messanger[69323:12519409] 6.30.0 - [Firebase/Analytics][I-ACS800023] No pending snapshot to activate. SDK name: app_measurement
successfully got conversation models
```

This section of the code is responsible for searching for users in the database by matching the given username to the existing one in the firebase.

DataBaseMannager.swift

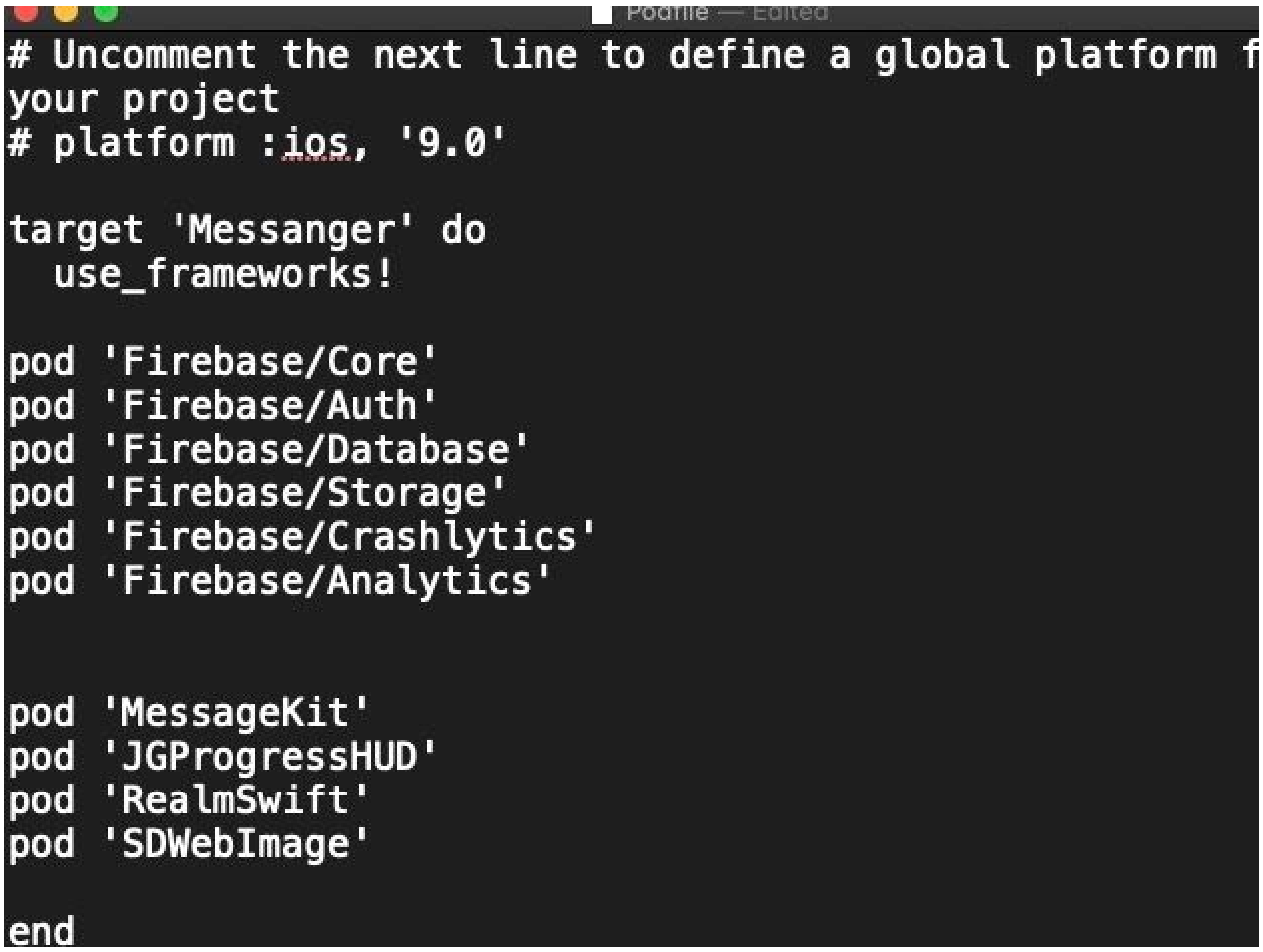


```swift
import CoreLocation

final class DatabaseManager {

    static let shared = DatabaseManager()

    private let database = Database.database().reference()

    static func safeEmail(emailAddress: String) -> String {
        var safeEmail = emailAddress.replacingOccurrences(of: ".", with: "-")
        safeEmail = safeEmail.replacingOccurrences(of: "@", with: "-")
        return safeEmail
    }
}

extension DatabaseManager {

    public func getDataFor(path: String, completion: @escaping (Result<Any, Error>) -> Void) {
        database.child("\(path)").observeSingleEvent(of: .value) { snapshot in
            guard let value = snapshot.value else {
                completion(.failure(DatabaseError.failedToFetch))
                return
            }
            completion(.success(value))

        }
    }
}

// Mark : - Account Managment

extension DatabaseManager{


    public func userExists(with email: String,
                           completion: @escaping ((Bool) -> Void)) {

        let safeEmail = DatabaseManager.safeEmail(emailAddress: email)
        database.child(safeEmail).observeSingleEvent(of: .value, with: { snapshot in
            guard snapshot.value as? [String: Any] != nil else {
                completion(false)
                return
            }

            completion(true)
        })

    }
```

```swift
    ///insert new user to database
    public func insertUser(with user: ChatAppUser, completion: @escaping (Bool)
        database.child(user.safeEmail).setValue([
            "first_name": user.firstName,
            "last_name": user.lastName
            ], withCompletionBlock: { [weak self] error, _ in

                guard let strongSelf = self else {
                    return
                }

                guard error == nil else{
                    print("failed to write to database")
                    completion(false)
                    return
                }


                strongSelf.database.child("users").observeSingleEvent(of: .value
                    if var usersCollection = snapshot.value as? [[String:String]
                        //append to user dictenery
                        let newElement = [
                            "name": user.firstName + " " + user.lastName,
                            "email": user.safeEmail

                        ]
                        usersCollection.append(newElement)

                        strongSelf.database.child("users").setValue(usersCollect
                            withCompletionBlock: {error, _ in
                            guard error == nil else {
                                completion(false)
                                return
                            }


                            completion(true)
                        })
                    }
                    else {
                        //creat the array
                        let newCollection: [[String: String]] = [
                            [
                                "name": user.firstName + " " + user.lastName,
                                "email": user.safeEmail
                            ]
                        ]

                        strongSelf.database.child("users").setValue(newCollecti
```

The DatabaseManager.swift is responsible for communicating with the database in order to do pretty much anything in the app such as creating/deleting an account, updating conversation entries and fetching the existing conversations for the users.

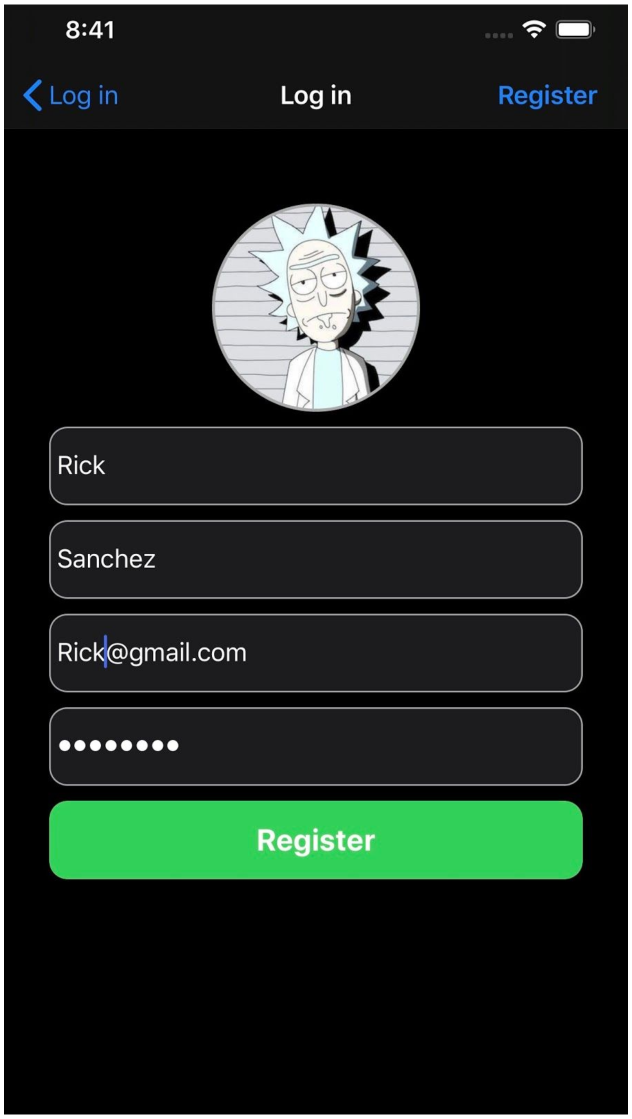
Pod File



```
# Uncomment the next line to define a global platform f
your project
# platform :ios, '9.0'

target 'Messanger' do
  use_frameworks!

pod 'Firebase/Core'
pod 'Firebase/Auth'
pod 'Firebase/Database'
pod 'Firebase/Storage'
pod 'Firebase/Crashlytics'
pod 'Firebase/Analytics'


pod 'MessageKit'
pod 'JGProgressHUD'
pod 'RealmSwift'
pod 'SDWebImage'

end
```

All the essential pods for this project are listed in the podFile.

Register View



Rick

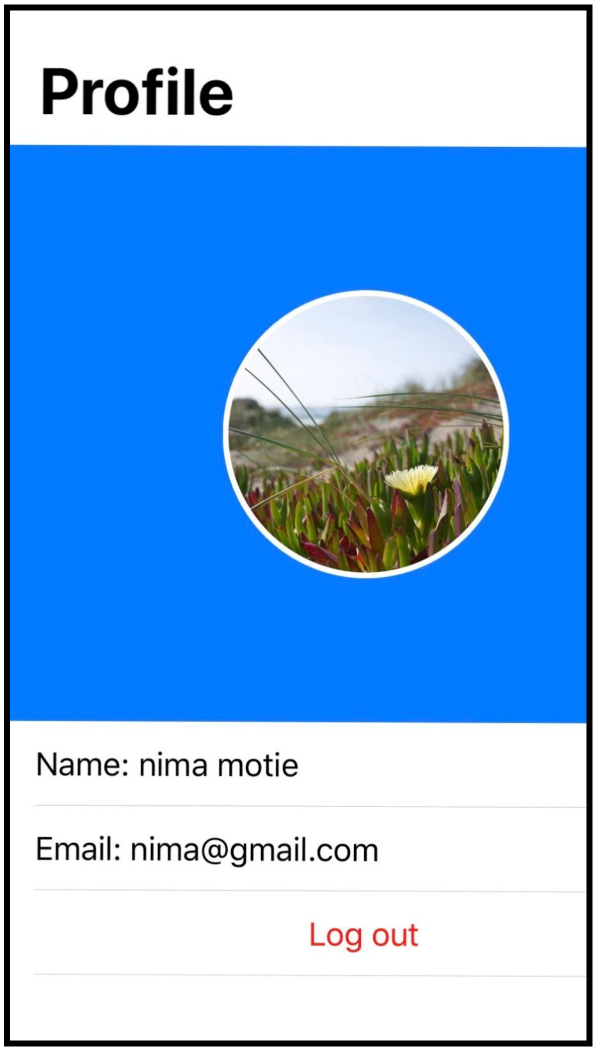
Sanchez

Rick@gmail.com

••••••••

Sample of the registration view in dark mod

Profile View



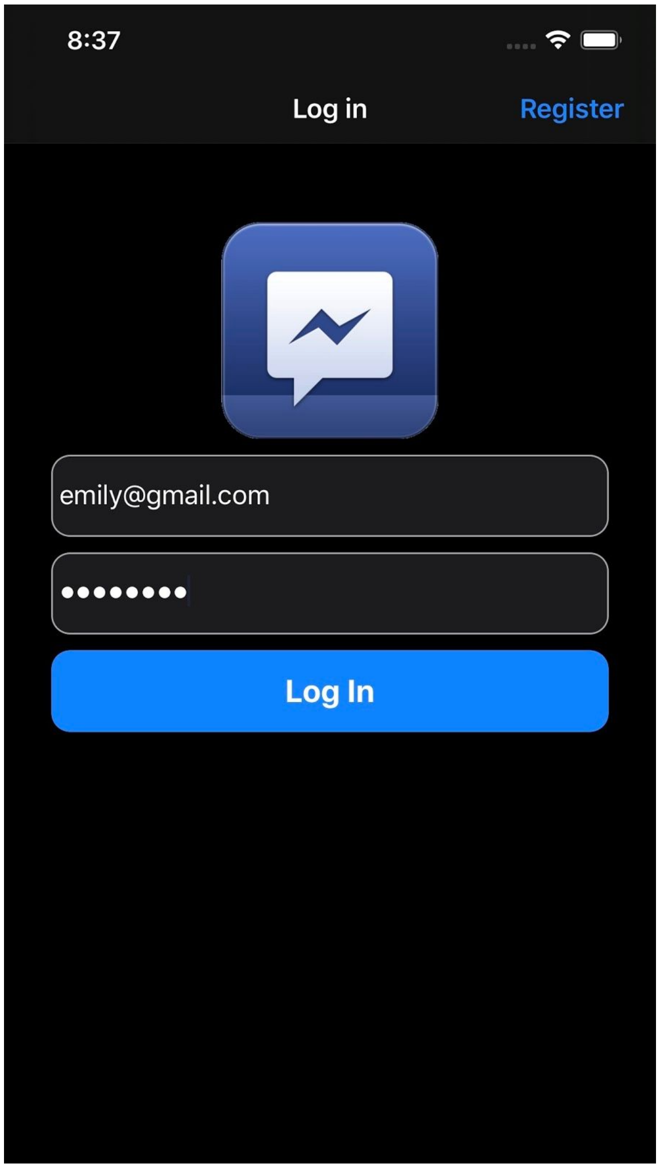
Profile

Name: nima motie

Email: nima@gmail.com

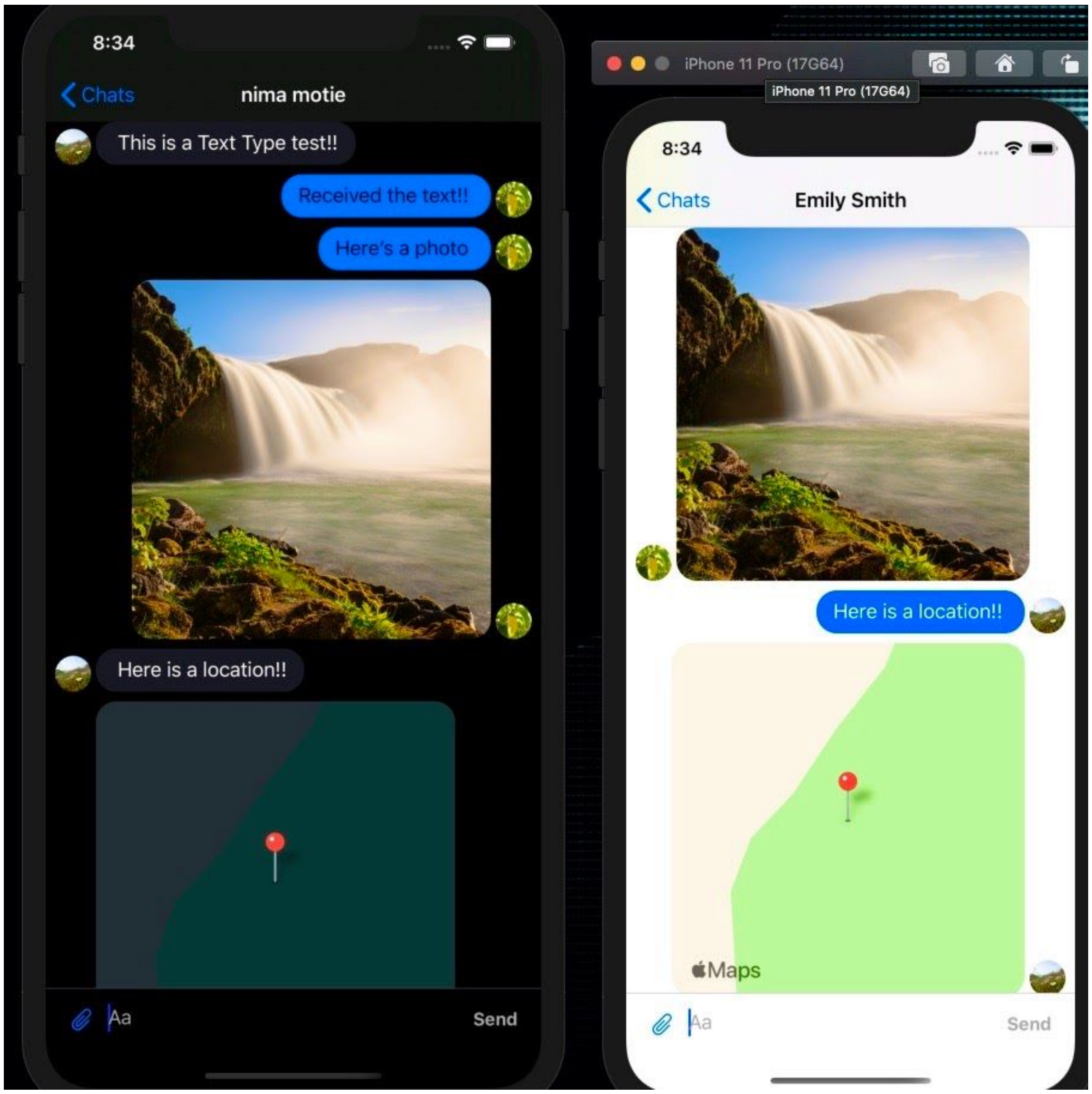
Sample of the profile TableView

Login View



Sample of the Login page

Conversation View



Sample of the conversation View