Prepr iOS Development Challenge

By: Julian Boyko

Purpose:

In short, this application is an online Address Book. Once signed up and logged in, you are able to store as many addresses as you'd like in the app. From there, you can select an address in your "Address Book" to view its location on the map. You are also able to select the marker displayed on the map to see full address information.

Stack:

For the backend I used Firebase. This stores all the addresses and all the user information. The Application itself was programmed in Swift and uses various CocoaPods, such as Firebase, Codable Firebase and GoogleMaps.

Process:

This was my first time working with a Firebase backend, so for that reason I had to find online resources and tutorials to demonstrate how to properly set up Firebase as a NoSQL user database for iOS development.

Once Firebase was set up properly, I found some resources online allowing me to aesthetically create Buttons and TextFields, which can be found in the "Julian_Boyko_iOS_Developer_Utilities.swift" class. From there, on sign-up I stored users in the "users" collection under the user's unique "uid" value, allowing me to access their document with ease, later on. This can be seen here:

```
let firebaseDB = Firestore.firestore()
// Adds a user to the firestore database by their unique UID
firebaseDB.collection("users").document(result!.user.uid).setData(
[
        "first_name": self.firstNameTextField.text!.trimmingCharacters(in: .whitespacesAndNewlines),
        "last_name": self.lastNameTextField.text!.trimmingCharacters(in: .whitespacesAndNewlines),
        "uid": result!.user.uid
]) { (error) in
    if error != nil {
        self.displayError(error: "Firebase failed storing user first name and last name")
    }
}
```

Considering this was my first time using Firebase, I had to read a lot of their documentation to successfully implement their service. After fleshing out Sign-Up and Login, it was time to move on to the HomeViewController. For the HomeViewController, I added two buttons, one to Logout and one to add addresses. To add addresses, I needed to create an Address class which can be found in the "Julian_Boyko_iOS_Developer_Address.swift" file, which holds the Address structure.

Now that, that was complete, I created the View Controller that the user would enter their information into. In the "Julian_Boyko_iOS_Developer_AddAddressVC.swift" file, I created an

AddressAddDelegate protocol to handle the process of a user adding an address and to cancel their request.

```
import UIKit

protocol AddressAddDelegate: AnyObject {
    func addAddressDidCancel(_ controller: UIViewController)
    func addAddress(_ controller: UIViewController, didSave item: Address)
}

class Julian_Boyko_iOS_Developer_AddAddressVC: UIViewController {
```

When an Address object gets added, it is put into an array that stores Address objects in the HomeViewController and it is also encoded and pushed to Firestore, where it is saved in an array of address objects under the current user's document.

```
func addAddress(_ controller: UIViewController, didSave item: Address) {
    self.navigationController?.popViewController(animated: true)
    addresses.append(item)
    tableView.reloadData()

    let firebaseDB = Firestore.firestore()
    let docData = try! FirestoreEncoder().encode(item)

    let userDocument = firebaseDB.collection("users").document(Auth.auth().currentUser!.uid)
    userDocument.updateData([
        "addresses": FieldValue.arrayUnion([
            docData
        ])
    ])
}
```

In the HomeViewController class, I also had to decode address objects that were in the Firestore array and retrieve them on "viewDidLoad" to ensure that all objects stored in the Firestore database would be loaded on login. Also, notice how the highlighted portion demonstrates how I access the current users document. It is done by using their unique "uid" value.

Once that was completed, it was time to implement GoogleMaps. First, I installed GoogleMaps using it's CocoaPod. From there, I created the "Julian_Boyko_iOS_Developer_AddressDetailVC" which is where GoogleMaps is implemented. In this class, I have a UIView outlet called "viewForMap" which is where the GoogleMap will be loaded inside. I create a function called "getLonLatAndSetUpMap" which uses CoreLocation to turn the String of an inputted address into latitude and longitude coordinates. Once I was able to retrieve accurate latitude and longitude coordinates, I loaded them up into a function I created called "setUpGoogleMaps" which I figured out how to create following the Google Maps documentation. In that function, I simply load the coordinates passed into it, set the mapView to the view object, and now the map is loaded.

```
override func viewDidLoad() {
    super.viewDidLoad()

    setUpAttributes()
    getAddressesFromFirestore()
}

func setUpAttributes() {
    Utilities.styleFilledButton(addButton)
    Utilities.styleHollowButton(logoutButton)

    tableView.dataSource = self
}

func getAddressesFromFirestore() {
    let firestoreDB = Firestore.firestore()
    let userDocument = firestoreDB.collection("users").document(Auth.auth().currentUser!.uid)

    userDocument.getDocument { (document, error) in
        if let document = document, document.exists {
            if let addresses = document.get("addresses") as? NSArray {
                for address in addresses {
                    let retrievedAddress = try! FirestoreDecoder().decode(Address.self, from: address as! [String : Any])
                    self.addresses.append(retrievedAddress)
                }
                self.tableView.reloadData()
            }
        } else {
            print("Document does not exist")
        }
    }
}
```

Now notice the getAddressesFromFirestore() function. This was particularly tricky, because I had to retrieve the “addresses” portion of the user document and cast it to an NSArray. Once casted, I had to loop through the array, decoding each object and appending it to my addresses array. Once that was complete, I reload the Table View to display the objects.

It took quite a while to finish this challenge, but ultimately, I learned a lot and had a lot of fun.