

Criterion C: Development

Techniques Used	1
The Structure of the Application	2
Launch Screen	3
Main Menu Screen	4
New Medicine Entry Screen	5
Notification Editing Screen	6
Medicine Data Screen	9
Statistics Screen	10
Calendar Screen	12
Map Screen	13
Settings Screen	15
Works Cited	16

1. Techniques Used

Multi-user environment	File reading	Connection with Apple Map	Database recording
Notification system	Mail integration	Nested loops	Two-dimensional arrays
Hierarchical data structure	UserNotifications, UIKit, PDFKit, MapKit (API).	Loops	

2. The Structure of the Application

MeDaily Reminder is a medicine reminder application for elderly people who often forget to take medicines. The name of the application is the combination of “Me”, “Daily” and “MeD” which is the abbreviation for medicine. The background colour of the application is chosen carefully as navy blue (#12185A) for a better user experience. According to research¹, it is the favourite blue tone for the elderly that creates a calming and confident atmosphere that effectively fits the purpose of the application.

Declaration types:

- **@IBOutlet² var;** declares that the Interface Kit can identify the property and synchronize with the display.
- **override func viewDidLoad;** is used for loading the view of these variables.
- **@IBAction func;** declares actions.
- **let³;** declares constant variables.
- **guard let⁴;** is used for exiting the function/condition.

When the user scrolls down the screen, she can go back to the previous screen.

¹ Eldertech, “*Designing Technology for Seniors - Color in User Interfaces for Elderly People.*”, 22 Apr. 2017.

² Apple Inc., “*Concepts in Objective-C Programming.*” *Outlets*, 9 Jan. 2012.

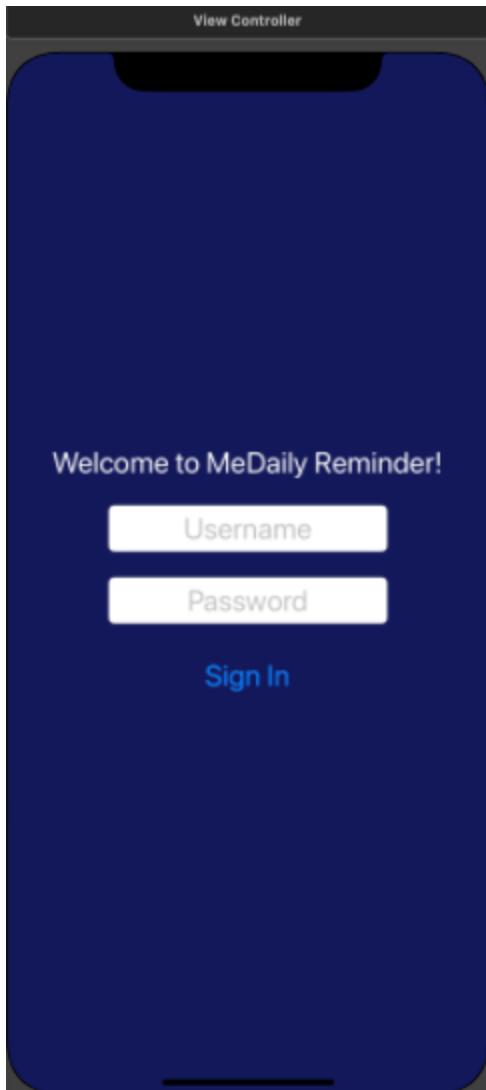
³ Agoi Abel, “*The ‘Let’ Keyword in Swift.*”, Medium, 8 Jan. 2018.

⁴ Paul Hudson, “*When to Use Guard Let Rather than If Let.*” Hacking with Swift, 5 June 2020.

2.1. Launch Screen

The user enters the Username and Password. It creates a multi-user environment. As the client is just Ms G for now, no authentication is required to keep data safe, but in the next versions of the application, Google Firebase Authentication will be used.

Screenshot 1: Launch Screen



Screenshot 2: Launch Screen Code (ViewController⁵)

```

8 import UIKit
9
10 class ViewController: UIViewController {
11
12     @IBOutlet var welcomeLabel: UILabel!
13     @IBOutlet var usernameField: UITextField!
14     @IBOutlet var passwordField: UITextField!
15     @IBOutlet var signinButton: UIButton!
16
17
18     override func viewDidLoad() {
19         super.viewDidLoad()
20
21     }
22
23     @IBAction func signinGo(_ sender: UIButton) {
24
25     }
26 }
```

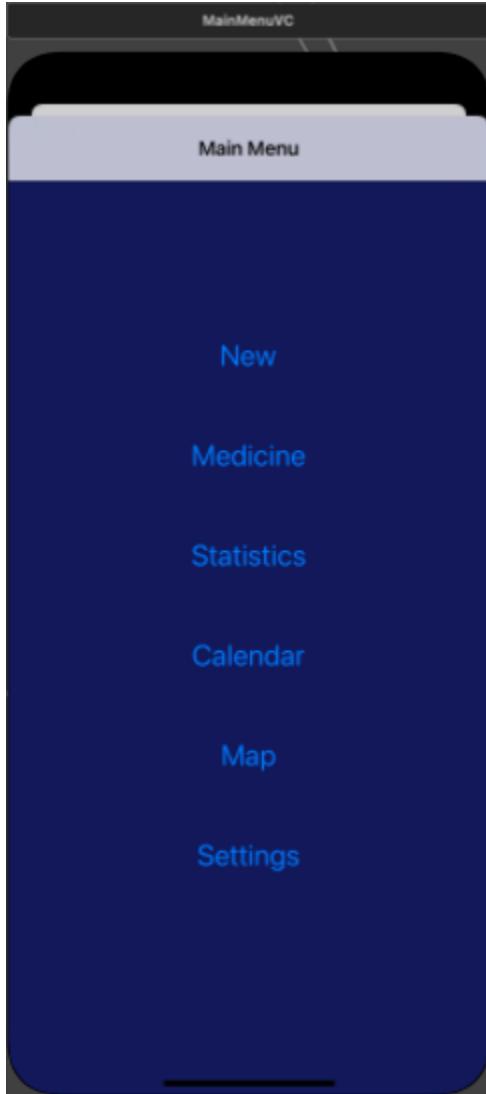
In order to bind the UI to the code, UIKit as an API frame is imported. Between 12-15, variables are identified to show the launch screen. Line 23 represents the sign-in button as a function: When the user presses Sign In, the function works as showing the Main Menu screen.

⁵ See Appendix “Variable Tables” [Table 1].

2.2. Main Menu Screen

It consists of 6 submenus: New, Medicine, Statistics, Calendar, Map and Settings.

Screenshot 3: Main Menu Screen



Screenshot 4: Main Menu Code (MainMenuVC)⁶

```

27
28 class MainMenuVC: UIViewController {
29
30     ...
31     ...
32     ...
33     ...
34     ...
35     ...
36     ...
37     ...
38     override func viewDidLoad() {
39         super.viewDidLoad()
40     }
41 }
42 }
43 }
44 }
45 }
```

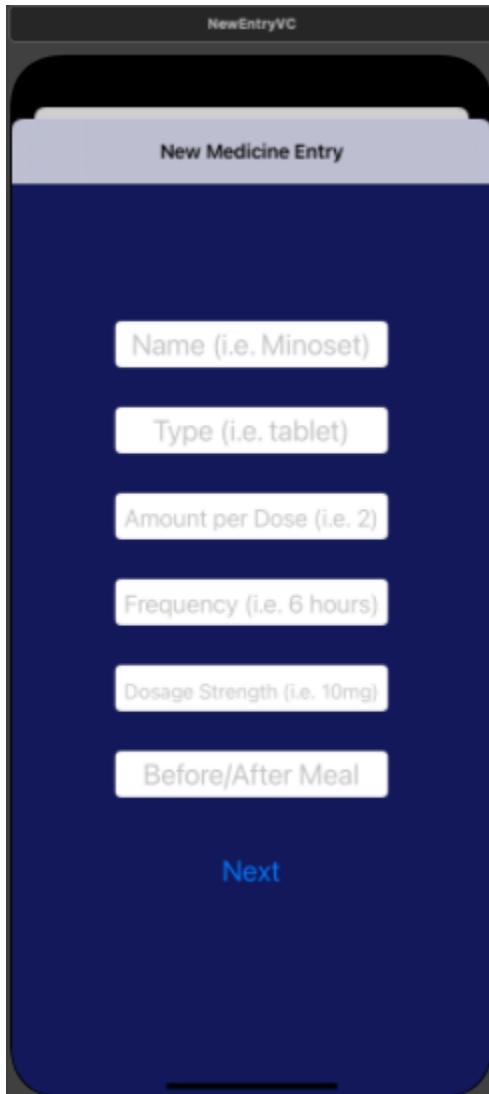
Between 31-35, the variables are identified. The user can switch between the other view controllers, so this screen has the most connections to the other screens.

⁶ See Appendix “Variable Tables” [Table 2].

2.3. New Medicine Entry Screen

When the user presses the “New” button on the main screen, some specific parameters are shown.

Screenshot 5: New Medicine Entry Screen



```

107
108 class NewEntryVC: UIViewController {
109
110     @IBOutlet var nameField: UITextField!
111     @IBOutlet var typeField: UITextField!
112     @IBOutlet var amountPerDoseField: UITextField!
113     @IBOutlet var frequencyField: UITextField!
114     @IBOutlet var strengthField: UITextField!
115     @IBOutlet var mealField: UITextField!
116     @IBOutlet var morningNextButton: UIButton!
117
118     override func viewDidLoad() {
119         super.viewDidLoad()
120
121     }
122
123 }
124

```

Screenshot 6: New Medicine EntryCode (NewEntryVC⁷)

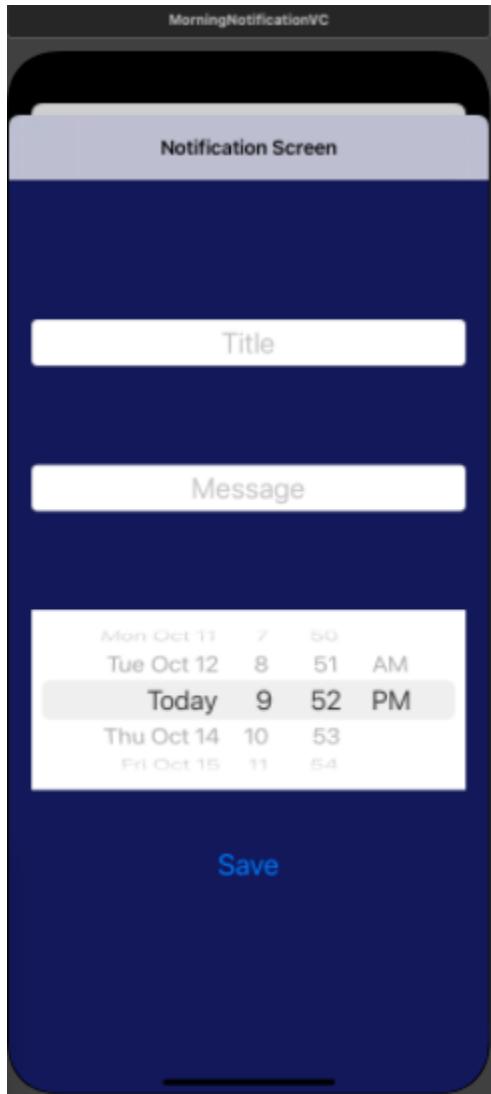
Between 110-116, all the variables for this screen are identified. After filling in the parameters, the client presses the Next button, and the information she has filled in is saved to the medicine database⁸.

⁷ See Appendix “Variable Tables” [Table 3].

⁸ See Appendix “Medicine Database JSON File”.

2.4. Notification Editing Screen

Screenshot 7: Notification Editing Screen (morningNotificationsVC⁹)



The client freely changes the title and the message of the notification and chooses the time for medicine.

To access the notification centre, UserNotifications as an API is imported to the code. The names of the variables start with morning because at the beginning, I planned to create three notification pages for different times of day, but I thought that this feature wouldn't be useful for the elderly. Between 137-145, there is a connection to Apple device settings that the application asks the user to receive notifications. If the user doesn't allow notifications, the permission will be denied, so the user should allow the notifications. Between 148-205, morningSaveAction as a function is declared. When the time the user chooses to get a reminder notification comes, the application triggers the notification center to send it to the user's screen. In this situation, when the user presses on the reminder notification, she opens the application, so she confirms that she takes her medication. In the first situation, if she doesn't press the reminder notification, the algorithm understands that Ms G didn't take her medicine. So, the

application decides to send another reminder notification. Likewise, in the first situation, if the user doesn't press the reminder notification, this time, the application won't trigger the notification center. So, the application sends a reminder email to the companion. Between 207-214, a function is declared for indicating the format of the banner when the user saves the notification.

⁹ See Appendix “Variable Tables” [Table 4].

Screenshot 8: Notification Editing Code

```

125
126 import UserNotifications
127
128 class morningNotificationsVC: UIViewController{
129     ...
130     @IBOutlet var morningNotificationTitleField: UITextField!
131     @IBOutlet var morningNotificationMessageField: UITextField!
132     @IBOutlet var morningNotificationDatePicker: UIDatePicker!
133
134     let notificationCenter = UNUserNotificationCenter.current()
135
136
137     override func viewDidLoad() {
138         super.viewDidLoad()
139         notificationCenter.requestAuthorization(options: [.alert, .sound]) { (permissionGranted, error) in
140             if(!permissionGranted){
141                 print("Permission denied.")
142             }
143         }
144     }
145
146
147
148     @IBAction func morningSaveAction(_ sender: Any) {
149         notificationCenter.getNotificationSettings { (settings) in
150             ...
151             DispatchQueue.main.async {
152                 ...
153                 let title = self.morningNotificationTitleField.text!
154                 let message = self.morningNotificationMessageField.text!
155                 let date = self.morningNotificationDatePicker.date
156                 if(settings.authorizationStatus == .authorized) {
157                     let content = UNMutableNotificationContent()
158                     content.title = title
159                     content.body = message
160
161                     let dateComp =
162                         Calendar.current.dateComponents([.year,.month,.day,.hour,.minute], from: date)
163                     let trigger = UNCalendarNotificationTrigger(dateMatching: dateComp, repeats: false)
164                     let request = UNNotificationRequest(identifier: UUID().uuidString, content: content, trigger: trigger)
165
166                     self.notificationCenter.add(request) { (error) in
167                         if(error != nil) {
168                             print("Error " + error.debugDescription)
169                         return
170                     }
171                 }
172             }
173         }
174     }

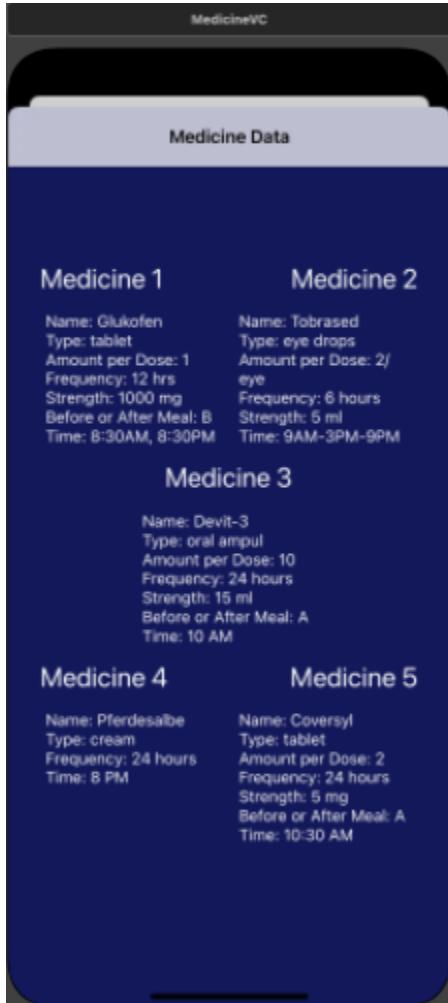
```

```
172     let ac = UIAlertController(title: "Notification Scheduled", message: "At " + self.formattedDate(date: date), preferredStyle: .alert)
173     ac.addAction(UIAlertAction(title: "OK", style: .default, handler: {(_ ) in}))
174     self.present(ac, animated: true)
175
176
177 }
178
179 {
180     let ac = UIAlertController(title: "Enable Notifications?", message: "To use this feature you must enable notifications in settings ", preferredStyle: .alert)
181     let goToSettings = UIAlertAction(title: "Settings", style: .default) {(_ ) in
182         guard let settingsURL = URL(string: UIApplication.openSettingsURLString)
183         else {
184             return
185         }
186
187         if(UIApplication.shared.canOpenURL(settingsURL)) {
188             UIApplication.shared.open(settingsURL) {(_ ) in
189             }
190
191             }
192         }
193
194         ac.addAction(goToSettings)
195         ac.addAction(UIAlertAction(title: "Cancel", style: .default, handler: {(_ ) in}))
196         self.present(ac, animated: true)
197
198
199 }
200
201 }
202
203
204
205
206
207 func formattedDate(date: Date) -> String {
208 {
209     let formatter = DateFormatter()
210     formatter.dateFormat = "d MMM y HH:mm"
211     return formatter.string(from: date)
212 }
213
214 }
215
```

2.5. Medicine Data Screen

It shows the medicines that the client has filled before. In the application, Ms G uses only 5 different medicines, so there are only 5 areas for her to see their information.

Screenshot 9: Medicine Data Screen

Screenshot 10: Medicine Data Code(MedicineVC¹⁰)

```

89 ...
90 class MedicineVC: UIViewController {
91     ...
92     @IBOutlet var med1Label: UILabel!
93     ...
94     @IBOutlet var med2Label: UILabel!
95     ...
96     @IBOutlet var med3Label: UILabel!
97     ...
98     @IBOutlet var med4Label: UILabel!
99     ...
100    @IBOutlet var med5Label: UILabel!
101   ...
102   ...
103   ...
104   ...
105   ...
106 }

```

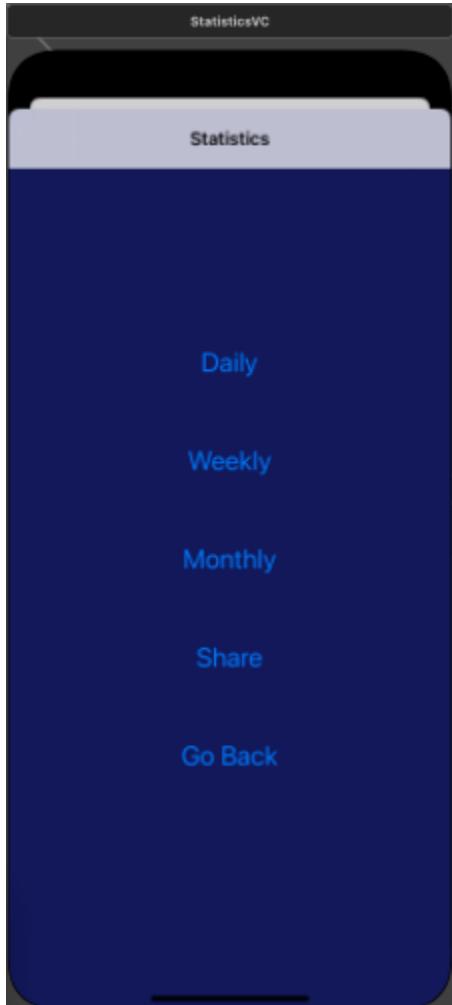
The client's previously filled medicines are in the database. Between 91-100, the variables are identified. The medicine information filled in New screen is saved to the database. So, Medicine Data screen is the visual appearance of the medicines.

¹⁰ See Appendix “Variable Tables” [Table 5].

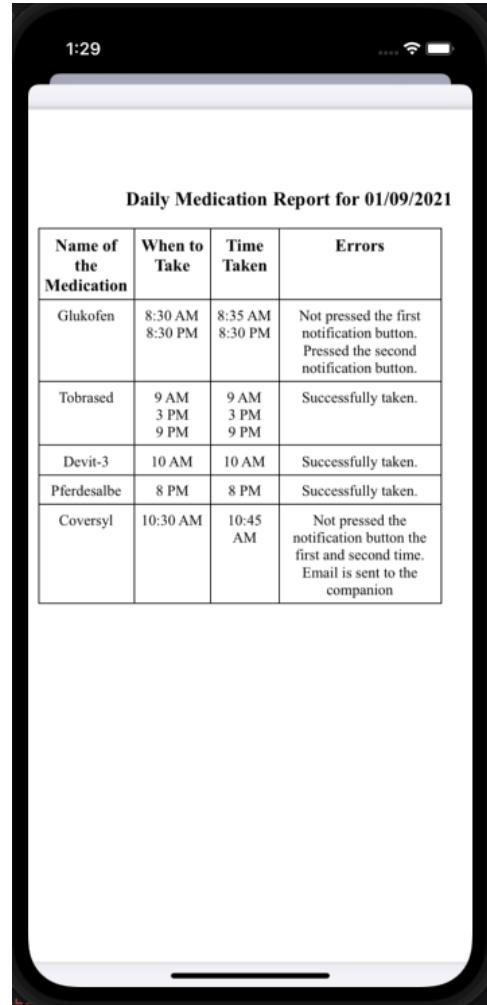
2.6. Statistics Screen

It is used to looking at daily, weekly and monthly reports for the medicine.

Screenshot 11a: Statistics Screen



Screenshot 11b: Daily Report Screen



```

74 class StatisticsVC: UIViewController {
75     ...
76     @IBOutlet var dailyButton: UIButton!
77     @IBOutlet var weeklyButton: UIButton!
78     @IBOutlet var monthlyButton: UIButton!
79     @IBOutlet var shareButton: UIButton!
80     @IBOutlet var goBackStatisticsButton: UIButton!
81     ...
82     ...
83     override func viewDidLoad() {
84         super.viewDidLoad()
85     }
86 }
87
88 }
```

Screenshot 12a: Statistics Code

(StatisticsVC¹¹)

The variables are identified between lines 76-80.

¹¹ See Appendix “Variable Tables” [Table 6].

```

218 import PDFKit
219
220 class DailyReportVC: UIViewController {
221     ...
222     let pdfView = PDFView()
223
224     override func viewDidLoad() {
225         super.viewDidLoad()
226         view.addSubview(pdfView)
227     }
228
229     ...
230
231     guard let url = Bundle.main.url(forResource: "DMR",
232                                     withExtension: "pdf") else {
233         return
234     }
235
236     guard let document = PDFDocument(url: url) else {
237         return
238     }
239
240     pdfView.document = document
241
242 }
243
244 override func viewDidLayoutSubviews() {
245     super.viewDidLayoutSubviews()
246     pdfView.frame = view.bounds
247 }
248
249 ...
250 }
251
252 }
253 }
```

Screenshot 12b: Daily Report Code (DailyReportVC¹²)

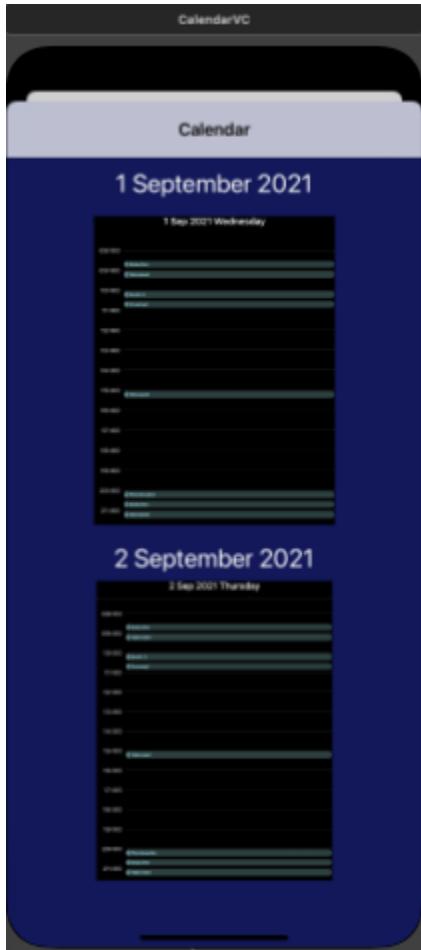
PDFKit as an API is imported to create a PDF view. Between 231-232, the source is identified as the PDF file's name and its type. If the user presses “Daily”, the pop-up comes out which shows the daily report.

¹² See Appendix “Variable Tables” [Table 10].

2.7. Calendar Screen

It shows a 2-days schedule for the medicines. The 2-days schedule contains today's and tomorrow's medicines. It is Ms G's idea because of her illnesses she couldn't go out to the pharmacy every day. So, 2 consecutive days are important to realize if the medicine runs out and buy a new one.

Screenshot 13: Calendar Screen



Screenshot 14: Calendar Code (CalendarVS¹³)

```

61   ↴
62   class CalendarVC: UIViewController {
63       ... @IBOutlet var cal1Label: UILabel!
64       ... @IBOutlet var cal1View: UIImageView!
65       ... @IBOutlet var cal2Label: UILabel!
66       ... @IBOutlet var cal2View: UIImageView!
67       override func viewDidLoad() {
68           ... super.viewDidLoad()
69           ...
70       }
71   }
72 }
73

```

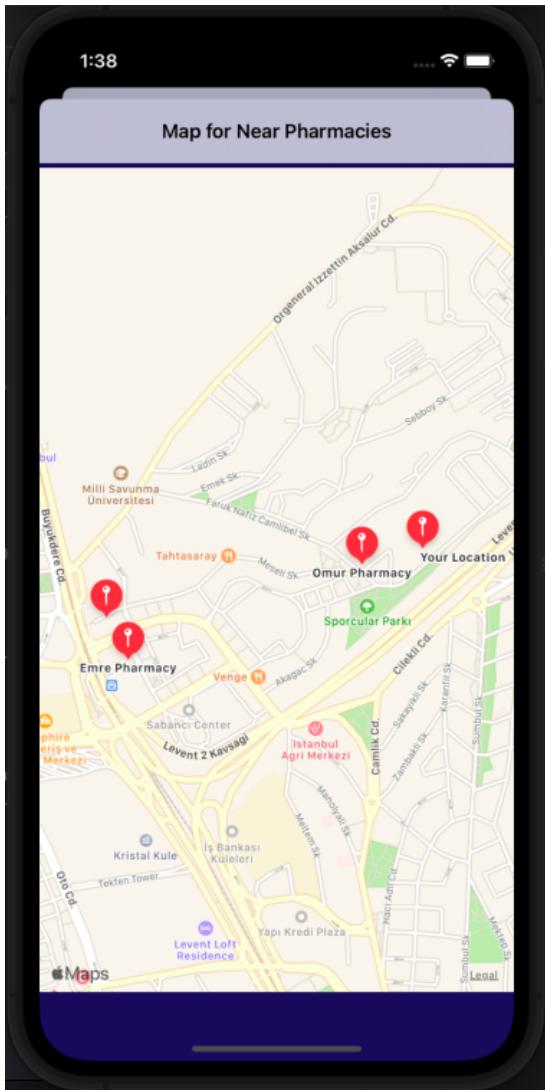
Between 63-66, the variables are identified. When the user saves the medicine information in the New Screen, the data is saved to the database. In calendar screen, the data from the database are shown for the medicines and their times.

¹³ See Appendix "Variable Tables" [Table 7].

2.8. Map Screen

It shows the client's location and the pharmacies within 500 meters. If the client runs out of medicine, she can see nearby pharmacies within walking distance. However, the problem is the client's location: If Ms G goes to another place, she can't see the pharmacies because her location is not live.

Screenshot 15: Map Screen



Screenshot 16: Map Code (MapVC¹⁴)

```

5 import UIKit
6 import MapKit
7
8 class MapVC: UIViewController {
9
10    @IBOutlet var mapView: MKMapView!
11
12    override func viewDidLoad() {
13        super.viewDidLoad()
14        let pharmacy1 = MKPointAnnotation()
15        pharmacy1.coordinate = CLLocationCoordinate2D(latitude: 41.087853, longitude:
16            29.014096)
17        pharmacy1.title = "Omur Pharmacy"
18        pharmacy1.subtitle = "100 meters away"
19        mapView.addAnnotation(pharmacy1)
20
21        let pharmacy2 = MKPointAnnotation()
22        pharmacy2.coordinate = CLLocationCoordinate2D(latitude: 41.086059, longitude:
23            29.008232)
24        pharmacy2.title = "Emre Pharmacy"
25        pharmacy2.subtitle = "400 meters away"
26        mapView.addAnnotation(pharmacy2)
27
28        let pharmacy3 = MKPointAnnotation()
29        pharmacy3.coordinate = CLLocationCoordinate2D(latitude: 41.086854, longitude:
30            29.007683)
31        pharmacy3.title = "Arzu Pharmacy"
32        pharmacy3.subtitle = "450 meters away"
33        mapView.addAnnotation(pharmacy3)
34
35        let randomLocation = MKPointAnnotation()
36        randomLocation.coordinate = CLLocationCoordinate2D(latitude: 41.087338, longitude:
37            29.010724)
38
39        let yourLocation = MKPointAnnotation()
40        yourLocation.coordinate = CLLocationCoordinate2D(latitude: 41.088154, longitude:
41            29.015615)
42        yourLocation.title = "Your Location"
43        mapView.addAnnotation(yourLocation)
44
45        let region = MKCoordinateRegion(center: randomLocation.coordinate,
46            latitudinalMeters: 1000, longitudinalMeters: 1000)
47        mapView.setRegion(region, animated: true)
48    }
49 }

```

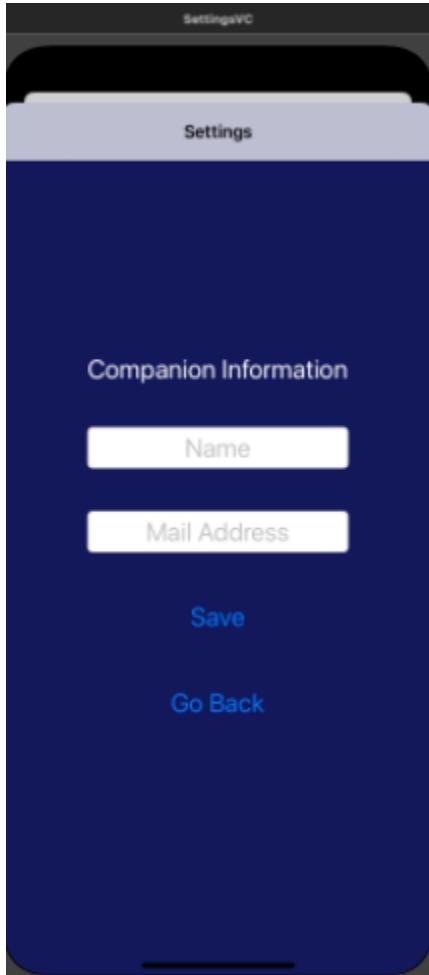
In order to access the current map view, MapKit as an API is imported to the code. As I know Ms G's location, I found her latitude and longitude. This process is also done for the pharmacies. To show a specific area, the diameter is arranged to 1000 meters. In this way, the user can see the pharmacies within a radius of 500 meters.

¹⁴ See Appendix “Variable Tables” [Table 8].

2.9. Settings Screen

It shows Companion Information where the client fills in Name and Mail Address.

Screenshot 17: Settings Screen



Screenshot 18: Settings Code (SettingsVC¹⁵)

```

45  ...
46  class SettingsVC: UIViewController {
47  ...
48  ...
49  ...
50  ...
51  ...
52  ...
53  ...
54  ...
55  ...
56  ...
57  ...
58  ...
59  ...
60  }

```

The code shows the implementation of the `SettingsVC` class. It defines several outlets for UI elements: `compInfoLabel`, `compUsernameField`, `compPasswordField`, `compSaveButton`, and `goBackSettingsButton`. It also overrides the `viewDidLoad()` method to call `super.viewDidLoad()`.

Between 48-52, the variables are identified. When the user fills in Companion Information, the data is saved to the Google Database. It is important for the user to save companion information because it is connected to the notification algorithm to break and send an email to the companion.

Word count: 980

¹⁵ See Appendix “Variable Tables” [Table 9].

Works Cited

- Abel, Agoi. “The ‘Let’ Keyword in Swift.” *Medium*, Medium, 8 Jan. 2018, <https://medium.com/@agoiabeladeyemi/the-let-keyword-in-swift-86b9e311da64#:~:text=In%20swift%2C%20we%20use%20the%20let%20keyword%20to%20declare%20a,value%20can%20not%20be%20changed>.
- Apple Inc., “Concepts in Objective-C Programming.” *Outlets*, 9 Jan. 2012, <https://developer.apple.com/library/archive/documentation/General/Conceptual/CocoaEncyclopedia/Outlets/Outlets.html>.
- Eldertech. “Designing Technology for Seniors - Color in User Interfaces for Elderly People.” *User Interfaces for Seniors*, 22 Apr. 2017, eldertech.org/color-in-designing-technology-for-seniors/#:~:text=Navy%20blue%2C%20sky%20blue%2C%20and,the%20spiritual%20or%20reflective%20mood.
- Hudson, Paul. “When to Use Guard Let Rather than If Let.” *Hacking with Swift*, Hacking with Swift, 5 June 2020, www.hackingwithswift.com/quick-start/understanding-swift/when-to-use-guard-let-rather-than-if-let#:~:text=Swift%20gives%20us%20an%20alternative,stay%20around%20after%20the%20check.