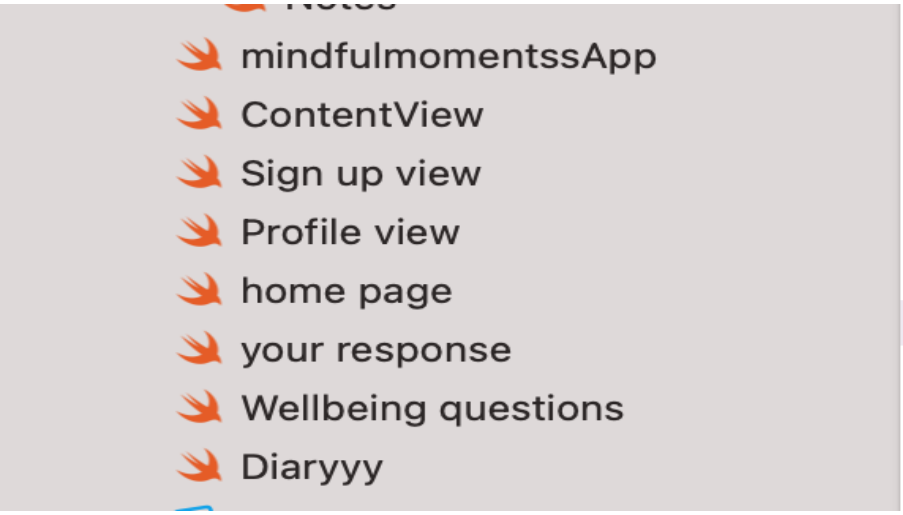
Criterion C: Development

Programming the user interfaces:

I am developing a program that would help to reduce stress and increase motivation for my client, Mr. K. After doing some research, I concluded that the best solution for doing the user interface would be Xcode. In the IDE, I created different files to develop each screen view for the application. By separating them into different files, it makes me easier to go through each view and know which screen has errors to fix.



Setting up variables in different screen view:

For each of the screen views, I created variables to do their roles, this helped make it easier to code and know which variable to use as a condition. These variables are used for handling login input, error tracking, displaying text and view navigation. I set them as private variables to restrict their access within the struct. This ensures that there will be no misuse by other parts of the app.

ContentView:

```
struct ContentView: View {
    @State private var username = ""
    @State private var password = ""
    @State private var wrongUsername = 0
    @State private var wrongPassword = 0
    @State private var showingProfileView = false
    @State private var showingSignUpView = false
```

SignUpView:

```
struct SignUpView: View {
    @State private var username1 = ""
    @State private var password1 = ""
    @State private var wrongUsername1 = 0
    @State private var wrongPassword1 = 0
    @State private var showingLoginScreen1 = false
    @State private var showingSignUpView1 = false
    @Environment(\.dismiss) var dismiss
```

ProfileView:

```
struct ProfileView: View {
    @State private var showingHomePage = false
    @State private var showingContenttview = false
```

HomePageView:

```
struct HomePageView: View {
    @State private var showingYourResponseScreen = false
    @State private var showingWellbeingQuestionScreen = false
    @State private var showingDiaryScreen = false
    @State private var showingProfileView = false
    @State private var showingSleeproutineView = false
    @StateObject var viewModel = SleepViewModel()
```

sleep_routine:

```
class SleepViewModel: ObservableObject {
    @Published var sleepHour: Int = 0
    @Published var sleepMinute: Int = 0
    @Published var wakeHour: Int = 0
    @Published var wakeMinute: Int = 0
}

struct sleep_routine: View {
    @ObservedObject var viewModel: SleepViewModel
    @State private var sleepHour = 0
    @State private var sleepMinute = 0
    @State private var wakeHour = 0
    @State private var wakeMinute = 0
    @State private var isEditingSleepTime = false
    @State private var isEditingWakeTime = false
```

Wellbeing_questions:

```
   @State private var currentQuestionIndex = 0
   @State private var answer = ""
```

Diaryyy:

```
struct Diaryyy: View {
    @State private var diary = ""
```

Buttons and navigation link/destination:

The captures of the code below show the code that I used to make buttons and their destination/ navigation link for each button in each screen view. For the button, I used the Image design that I uploaded in criterion B, so I do not need to code the it colour and shape. These buttons will enable the user to click and change the screen view.

ContentView:

Sign up button:

```
ZStack {
    Button(action: { showingSignUpView = true }) {
        Text("Sign up")
            .bold()
            .foregroundColor(.blue)
            .frame(width: 100, height: 30)
    }

Button(action: {
    authenticateUser(username: username, password:
        password)
}) {
    Text("Sign in
                    ")
}
.buttonStyle(.borderedProminent)
.tint(.blue)
.buttonBorderShape(.roundedRectangle(radius: 10))
.controlSize(.large)
```

this button will send the user from the sign-in view to the sign-up screen view

Sign In button:

This button will submit the sign-in and send the user into the home page after they input their username and password.

Sign up View:

For the Sign-up view I have not successfully connected to the database yet, so I have built the navigation link for this button

```
49                      Text("Sign up
                                    ")
50                  }
51                  .buttonStyle(.borderedProminent)
52                  .tint(.blue)
53                  .buttonBorderShape(.roundedRectangle(radius: 10))
54                  .controlSize(.large)
55                  .position(x: 195, y: 390)
56                  Button(action: {
57                      authenticateUser(username: username1, password:
                            password1)
58                      dismiss.callAsFunction()
59                  }) {
60                      Text("Sign in")
61
62                          .bold()
63                  }
64                  .position(x: 195, y: 550)
```

Profile View:

Home page button:

```
VStack {
    NavigationLink(destination: HomePageView(),
        isActive: $showingHomePage) {
        EmptyView()
    }
    Button(action: { showingHomePage = true }) {
        Text("Home page")
            .bold()
```

This button will send the user to the home page.

Log out button:

```
        NavigationLink(destination: ContentView(),
            isActive: $showingContenttview) {
            EmptyView()
        }
        Button(action: { showingContenttview = true }) {
            Text("Log out")
                .bold()
                .foregroundColor(.blue)
                .position(x: 50, y: 10)
```

This Logout button will send the user back to the ContentView(Login view)

HomePageView:

```
ZStack {
    Image("6")
        .resizable()
        .frame(width: 420, height: 920)
        .position(x: 200, y: 356)
        .navigationBarHidden(true)
        .navigationBarBackButtonHidden(true)
```

Here is the image for the colour and shape of the button

Profile button:

```
Button(action: { showingProfileView = true }) {
    Text("                    ")
        .bold()
        .foregroundColor(.blue)
        .frame(width: 300, height: 300) // Adjust the
            frame size as needed
        .position(x: 300, y: 195) // Adjust the
            position to center the button
```

This button will send the user back to the profile view

Wellbeing question button:

```
Button(action: { showingWellbeingQuestionScreen = true
    }) {
    Text("                  ")
        .bold()
        .foregroundColor(.blue)
        .frame(width: 300, height: 300) // Adjust the
            frame size as needed
        .position(x: 130, y: 455) // Adjust the
            position to center the button
        .buttonStyle(PlainButtonStyle())
        .controlSize(.large)
        .buttonStyle(.borderedProminent)
        .tint(.blue)
        .buttonBorderShape(.roundedRectangle(radius:
            10))
}
.buttonStyle(PlainButtonStyle())

NavigationLink(destination:
    Wellbeing_questions(), isActive:
    $showingWellbeingQuestionScreen) {
    EmptyView()
}
```

This button will send the user to the wellbeing question view

Diary button

```
    Button(action: { showingDiaryScreen = true }) {
        Text("          ")
            .bold()
            .foregroundColor(.blue)
            .frame(width: 300, height: 300) // Adjust the
                frame size as needed
            .position(x: 310, y: 455) // Adjust the
                position to center the button
```

This button will send the user to the diary view

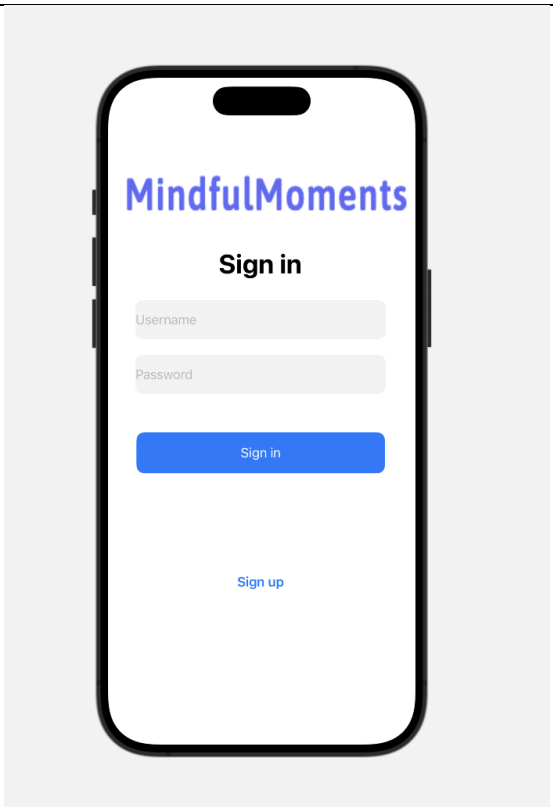
Sleep view button:

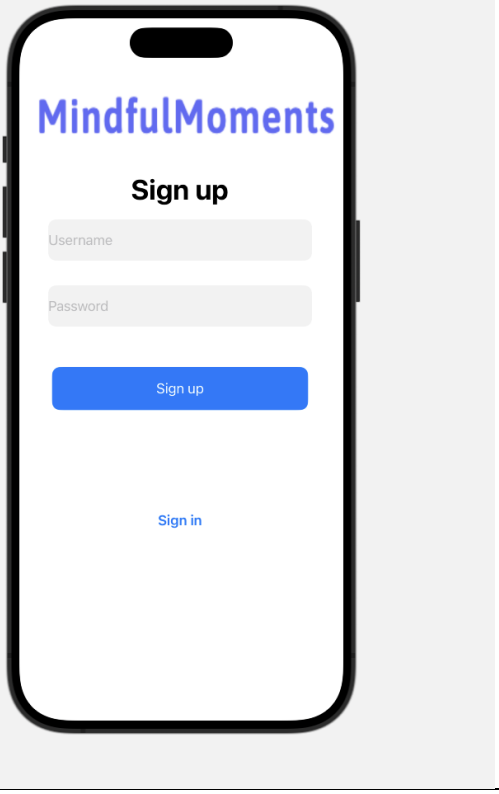
```
Button(action: { showingSleeproutineView = true }) {
    Text("    edit       ")
        .bold()
        .foregroundColor(.blue)
        .frame(width: 300, height: 300) // Adjust the
            frame size as needed
        .position(x: 90, y: 285) // Adjust the
            position to center the button
        .buttonStyle(PlainButtonStyle())
        .controlSize(.large)
        .buttonStyle(.borderedProminent)
        .tint(.blue)
        .buttonBorderShape(.roundedRectangle(radius:
            10))
}
.buttonStyle(PlainButtonStyle())
```

This button will send the user to the sleeping routine view

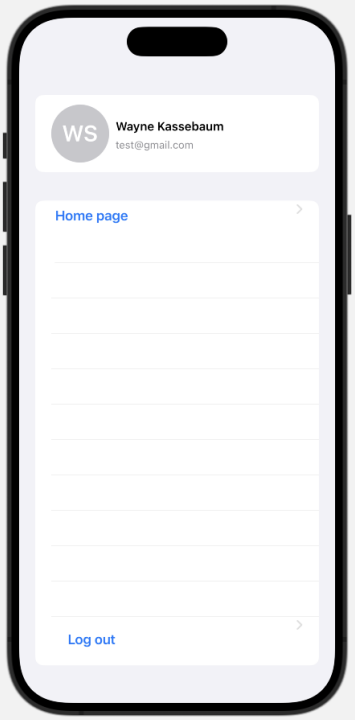
```
NavigationLink(destination:
    sleep_routine(viewModel: SleepViewModel()),
    isActive: $showingSleeproutineView) {
    EmptyView()
}
```

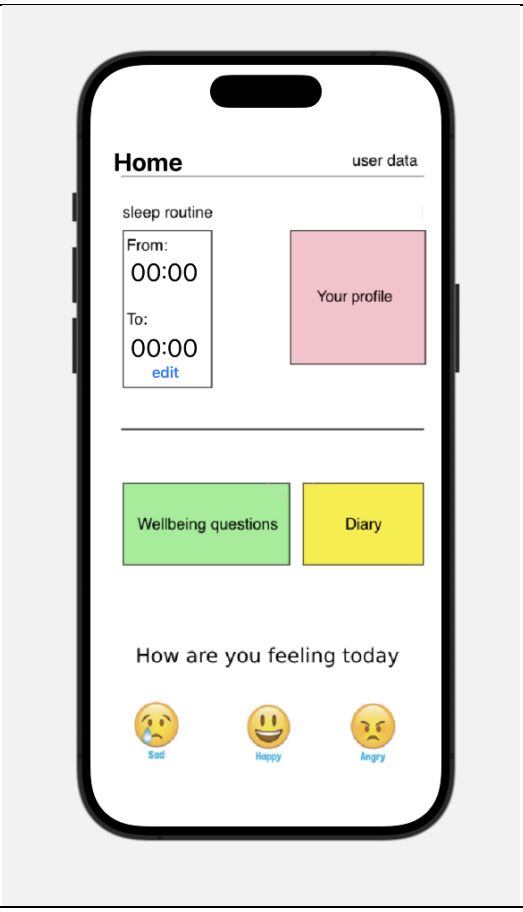
Overall view of each screen view in the app:

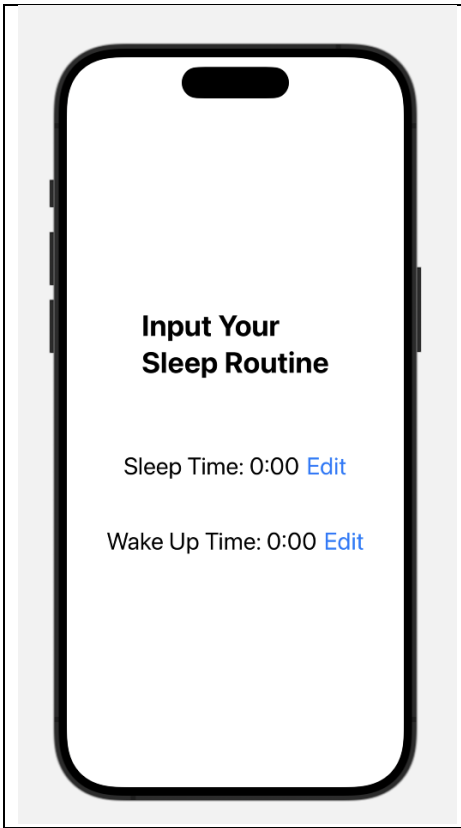
ContentView

Sign up view

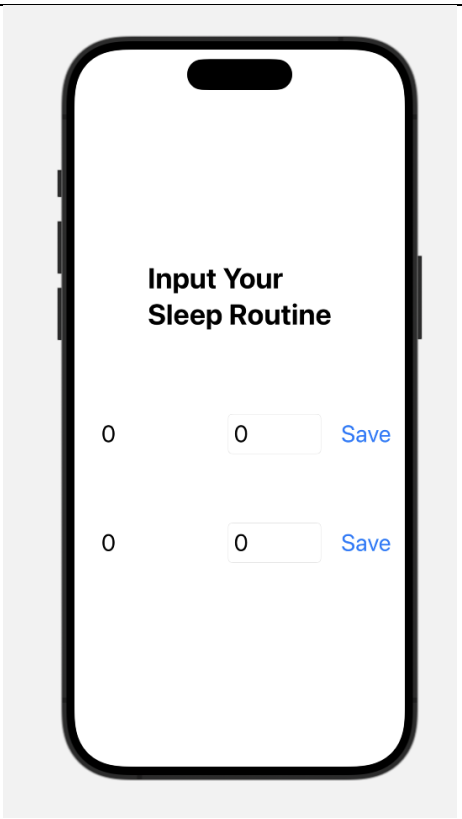


Profile View



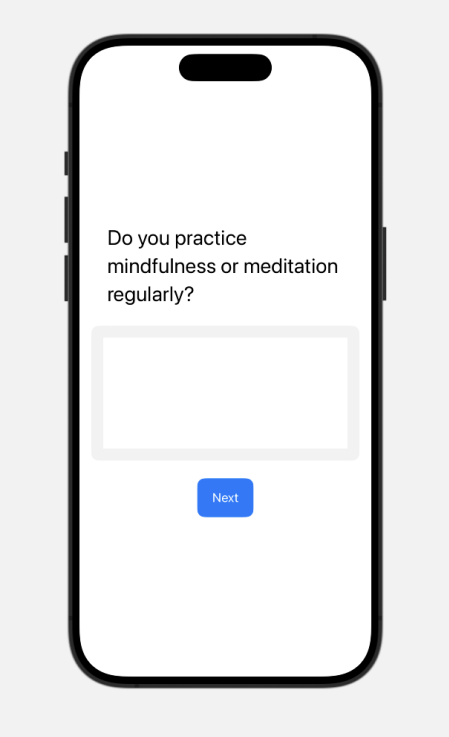
home page view

sleep routine view





.

wellbeing question view

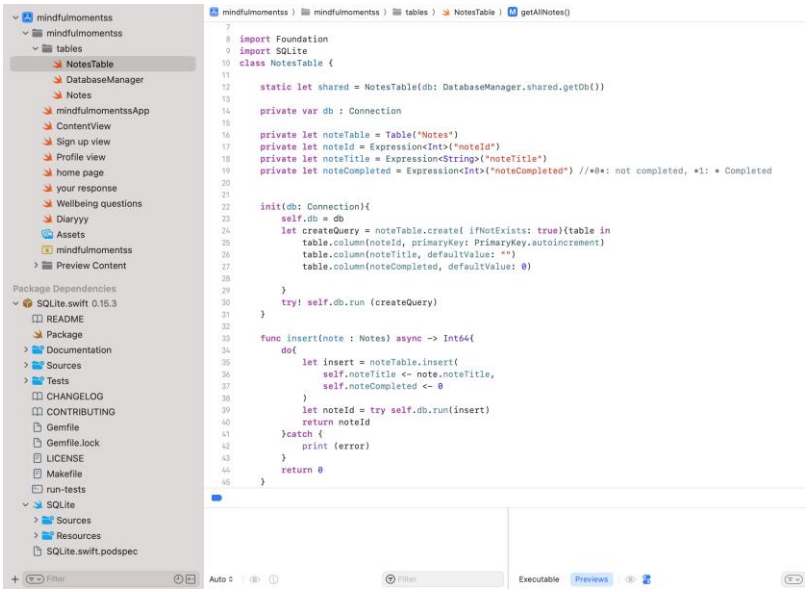


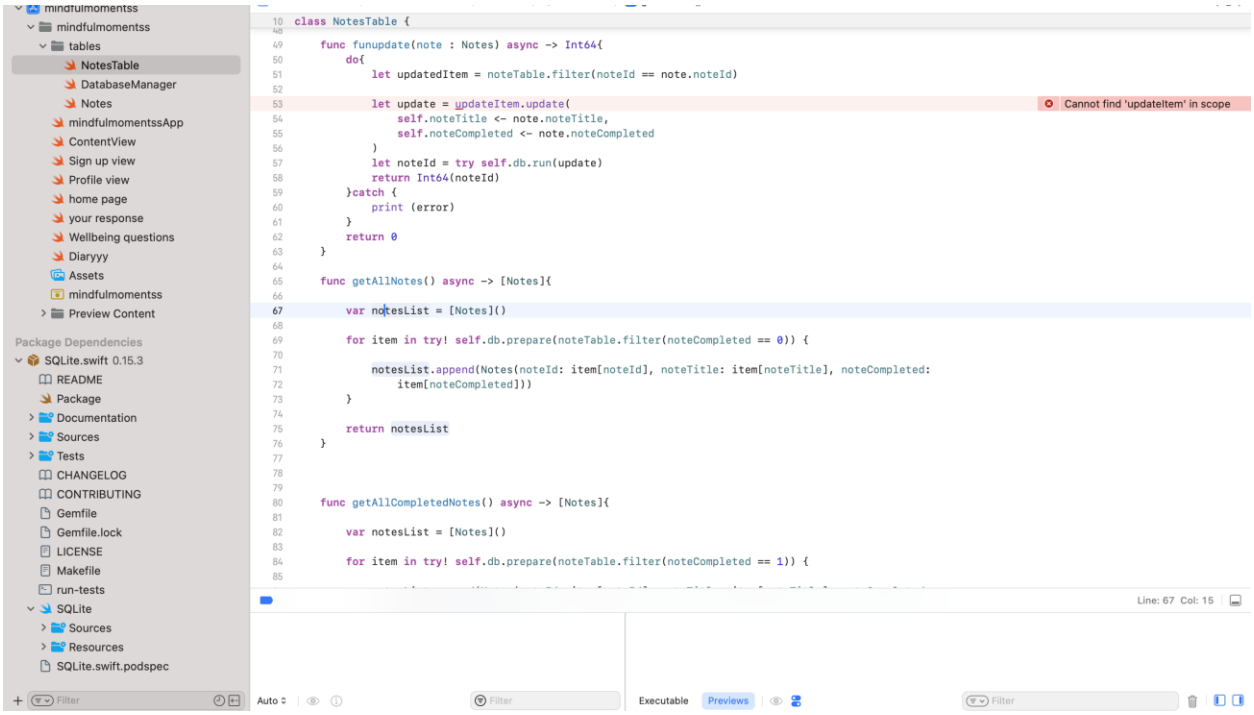
Diary view

Backend development:

In order to store the data, I have chosen SQLite as a database for my backend. I am now following tutorials and researching of how to connect my database properly to Xcode and store my program's data.

Here is the progress of trying to connect with SQLite:





For now, I am following the tutorial. My plan is to make these code work with no errors and change it back so it can store my application's data.