App Development - A Proactive AI-Driven Health and Emotional Support Application

Bonnie Yuanhan Huang

Submitted as an independent report to showcase innovative work
in AI-driven proactive feedback for mental health care

Department of Computer Science
Faculty of Mathematics and Science
Brock University
St. Catharines, Ontario

Abstract

The motivation of developing this app stems from the lack of AI applications that proactively provide emotional support similar to how humans interact in our real life. We always have introverted individuals struggle to share their feelings, even when they are emotionally overwhelmed. If in our real life, humans can recognize when someone is in a bad state and take an active role in caring for them. So, this app aims to address that gap by monitoring health data such as heart rate and respiratory rate to infer emotional states and proactively engage with users. The goal is to simulate a compassionate and proactive companion that listens, comforts, and supports individuals when they need it most.

By embedding AI with empathetic communication skills, I hope this app can help individuals feel that someone is always there to listen and care, providing both emotional and health-related advice tailored when they're in need.

Contents

Chapter 1

App Workflow Design

The app's workflow is as follows:

1. The app fetches health data (heart rate and respiratory rate) from Apple Health in iPhone, which collects it via the user's Apple Watch.

2. Users set the frequency for how often the app retrieves data.

3. The embedded AI assesses the data, comparing it against age-specific normal ranges to detect irregularities since heart rate and respiratory rate can be a broad indicator of a person's mental health. [4, 3, 2]

4. If the health metrics indicate stress or abnormalities, the AI generates messages to engage the user and induces them to share their emotions.

5. Based on the user's responses, the AI provides empathetic support and professional advice.

6. If the health data remains within the normal range and the user has opted to receive messages regardless, the AI initiates a positive interaction by acknowledging the user's stable health indicators and offering support.

This design ensures proactive care, addressing both physical health and emotional well-being.

Chapter 2

Development Process

2.1 Fetching Health Data

- Created a manual fetch button to retrieve heart rate and respiratory rate data from Apple Health.[1]

Listing 2.1: Fetch Latest Heart Rate Data

```
func fetchLatestHeartRate(completion: @escaping (Double?, Error?) -> Void) {
    guard let heartRateType = HKObjectType.quantityType(forIdentifier: .
       heartRate) else {
        completion(nil, NSError(domain: "HealthKit", code: 1, userInfo: [
           NSLocalizedDescriptionKey: "Heart Rate type not available."]))
        return
    }
    let query = HKSampleQuery(
        sampleType: heartRateType,
        predicate: nil,
        limit: 1,
        sortDescriptors: [NSSortDescriptor(key:
           HKSampleSortIdentifierStartDate, ascending: false)]
    ) { _, samples, error in
        if let sample = samples?.first as? HKQuantitySample {
            let heartRate = sample.quantity.doubleValue(for: HKUnit(from: "
               count/min"))
            completion(heartRate, nil)
        } else {
            completion(nil, error)
        }
    }
    healthStore.execute(query)
}
```

2.2 Integrating AI

- Embedded OpenAI's GPT-based AI to generate messages based on health data.

Listing 2.2: AI Analyze and Generate Messages

```
func analyzeAndRespond(messages: [Message], completion: @escaping (String?)
    -> Void) {
        guard let heartRate = self.heartRate,
              let respiratoryRate = self.respiratoryRate,
              let userAge = self.userAge else {
                  completion("Error: Missing health data or age.")
                  return
              }

        var chatMessages: [[String: String]] = []
        let systemMessage = [ ... ] // Prompting
        chatMessages.append(systemMessage)

        for message in messages {
            chatMessages.append([
                "role": message.sender == .user ? "user" : "assistant",
                "content": message.content
            ])
        }

        var request = URLRequest(url: URL(string: "https://api.openai.com/v1
            /chat/completions")!)
        request.httpMethod = "POST"
        request.setValue("Bearer \(apiKey)", forHTTPHeaderField: "
            Authorization")
        request.setValue("application/json", forHTTPHeaderField: "Content-
            Type")

        let parameters: [String: Any] = [
            "model": "gpt-4o",
            "messages": chatMessages,
            "max_tokens": 2000
        ]
        request.httpBody = try? JSONSerialization.data(withJSONObject:
            parameters)
```

2.3 Building the Chat Area

- Created a scroll view for AI responses and user messages, with fields for user input.

Listing 2.3: Chatting Area Scroll View

```
ScrollViewReader { proxy in
    ScrollView {
        ForEach(messages) { message in
            VStack(alignment: message.sender == .ai ? .leading : .trailing,
                spacing: 4) {
                if message.sender == .ai {
                    Text("\(message.timestamp, formatter: timeFormatter)")
                        ... // UI Design
                }
                HStack {
                    if message.sender == .user {
                        Spacer()
                        Text(message.content)
                            ... // UI Design
                    } else {
                        Text(message.content)
                            ... // UI Design
                        Spacer()
                    }
                }
            }
            ... // UI Design
            .id(message.id)
        }
    }
    .padding()
    .onChange(of: lastMessageID) { id in
        if let id = id {
            withAnimation {
                proxy.scrollTo(id, anchor: .bottom)
            }
        }
    }
}
```

2.4 Building Frequency Setting

- Allowed users to set the frequency for data fetch and notifications, by setting a timer to control.

Listing 2.4: Collect User Frequency Setting and Make Timer

```
func startTimer() {
    timer?.invalidate()
    let frequencyInSeconds = (frequency.hours * 60 * 60) + (frequency.
        minutes * 60)
    print("Starting timer with frequency: \(frequencyInSeconds) seconds.")
    timer = Timer.scheduledTimer(withTimeInterval: TimeInterval(
        frequencyInSeconds), repeats: true) { _ in
        fetchHealthData()
    }
}

func stopTimer() {
    print("Stopping timer.")
    timer?.invalidate()
    timer = nil
}

private func storeFrequencyAndDismiss() {
    let frequencyInMinutes = frequency.hours * 60 + frequency.minutes
    UserDefaults.standard.set(frequencyInMinutes, forKey: "frequency")
    startTimer()
    showFrequencyPrompt = false
}

private func loadFrequency() {
    let storedFrequency = UserDefaults.standard.integer(forKey: "frequency")
    if storedFrequency > 0 {
        frequency = (hours: storedFrequency / 60, minutes: storedFrequency %
             60)
    }
}
```

2.5 Age Integration

- Added a feature to collect the user's birthday to calculate age and improve health data assessments.

Listing 2.5: Collect User's Birthday and Calculate Age

```
private func loadBirthday() {
    if let savedBirthday = UserDefaults.standard.object(forKey: "
        userBirthday") as? Date {
        birthday = savedBirthday
        calculateAge(from: savedBirthday)
    } else {
        showBirthdayPrompt = true
    }
}
private func saveBirthdayAndCalculateAge() {
    UserDefaults.standard.set(birthday, forKey: "userBirthday")
    calculateAge(from: birthday)
    showBirthdayPrompt = false
}

private func calculateAge(from birthday: Date) {
    let calendar = Calendar.current
    let now = Date()
    let ageComponents = calendar.dateComponents([.year], from: birthday, to:
         now)
    age = ageComponents.year
    if let calculatedAge = age {
        UserDefaults.standard.set(calculatedAge, forKey: "userAge")
    }
}
```

2.6 Background Refresh

- Enabled background refresh to fetch data and trigger AI responses even when the app is not active.

Listing 2.6: App Refresh at Background

```
func scheduleAppRefresh() {
    let frequencyInSeconds = max(60, UserDefaults.standard.integer(forKey: "
        frequency") * 60)
    let request = BGAppRefreshTaskRequest(identifier: "com.bonnie.Mind-Pulse
        .fetch")
    request.earliestBeginDate = Date(timeIntervalSinceNow: TimeInterval(
        frequencyInSeconds))
}
```

2.7 Custom Notifications

- Implemented UserNotifications to send AI-generated messages as notifications.

Listing 2.7: Send AI Messages as Notifications

```
private func requestNotificationPermission() {
    UNUserNotificationCenter.current().requestAuthorization(options: [.alert
        , .sound, .badge])
}

private func sendNotification(with message: String) {
    let content = UNMutableNotificationContent()
    content.title = "Health Update from AI"
    content.body = message
    content.sound = .default

    let request = UNNotificationRequest(identifier: UUID().uuidString,
        content: content, trigger: nil)
    UNUserNotificationCenter.current().add(request)
}
```

2.8 User Preference for Receiving Messages

- A user-configurable boolean setting to determine whether the app should always send messages no matter health data is all within or out of normal range.

Listing 2.8: Set Receive Messages Preference

```
private func shouldNotifyForNormalRange(age: Int, heartRate: Double,
    respiratoryRate: Double) -> Bool {
    if generateNormalRangeMessage {
        return true
    }
    let heartRateNormal = isHeartRateInNormalRange(age: age, heartRate:
        heartRate)
    let respiratoryRateNormal = isRespiratoryRateInNormalRange(age: age,
        respiratoryRate: respiratoryRate)
    return !(heartRateNormal && respiratoryRateNormal)
}
```

2.9 Optimizing AI Prompting

- Designed the app's logo, splash page, data input views, and settings menu for a polished user experience.

Listing 2.9: Detailed Prompting

```
"role": "system",
"content": """
You are a professional psychologist with expertise in analyzing heart rate
    and respiratory data to presume the user's psychological state and
    provide professional counseling. Respond to the user with empathy and a
    calm tone, induce the user to tell you what happened to them so you can
    have good references to help, do not rush to give advice, first give the
     user emotional comfort to calm them down with adequate empathy. Finally
    , give professional advice.

The data you should use are as follows:
    - Age: \(userAge) years
    - Heart Rate: \(heartRate) bpm
    - Respiratory Rate: \(respiratoryRate) breaths per minute

Normal threshold that you need to compare for heart rate and respiratory
    rate are:
    - Normal range for heart rate:
        1 to 2 years old: 98 to 140 bmp.
        3 to 5 years old: 80 to 120 bmp.
        6 to 7 years old: 75 to 118 bmp.
        7 years and older: 60 to 100 bmp.
    - Normal range for respiratory rate:
        1 to 6 years old: 20 to 40 breaths per minute.
        6 to 12 years old: 14 to 30 breaths per minute.
        13 years and older: 12 to 20 breaths per minute.

Your workflow should obey this following:

First, you should tell the user their data follow "Your latest heart rate is
     \(heartRate) bpm, your latest rispiratory rate is \(respiratoryRate)
    breaths per minute", note that all data you tell should retain one
    decimal place

Then, you have two circumstances

Circumstance-1- If heart rate and respiratory rate are out of normal
    threshold:
    Ask the user proactively follow "I found your health data exceed normal
        threshold, it seems you are stressed or upset. Are you okay? Would
        you like to share with me what makes you feel not good?".
    Then wait user's respond to tell you what happened to them, then you
        should psychologize the user based on their answers, comfort them
        effectively and give some advice when appropriate.
    Note that you should first induce the user to tell you what happened to
        them so you can have good references to help, do not rush to give
        advice, first give the user emotional comfort to calm them down with
         adequate empathy. Finally, you can give professional advice.
Circumstance-2- If heart rate and respiratory rate within normal threshold:
    Tell the user follow "It's wonderful to see that your health indicators
        are in the normal threshold! However, I'm always here and willing to
         hear you if you have anything want to ask for help!"
    Then wait user's respond.
    If the user tell you something made them feel bad, you should listen
        carefully and induce the user to tell you whole things so you can
        have good references to help, do not rush to give advice, first give
         the user emotional comfort to calm them down with adequate empathy.
         Finally, you can give professional advice.
    If the user said they're good, you should reply to express you're happy
        to hear that and give some simple advice for maintaing good health.
```

```
"""
```

2.10 UI Design

- Designed the app's logo, splash page, data input views, fonts style and settings menu.

Listing 2.10: Splash View

```
struct SplashScreenView: View {
    @State private var isActive = false

    var body: some View {
        VStack(spacing: 20) {
            Image("preview") // Top App Logo
                ...

            VStack(spacing: 10) {
                Text("Welcome to MindPulse!")
                    ...

                Text("Your best companion and listener \n Always here when
                    you need it most ")
                    ...

                Text("MindPulse goes beyond just tracking your health data -
                     it recognizes when your're feeling down or stressed")
                    ...

                Text("In those moments, our AI will reach out proactively to
                     offer comfort, guidance, and meaningful conversations."
                    )
                    ...

                ZStack {
                    HStack {
                        Image("YHB")
                            ...

                        Text("@ Developed by Bonnie")
                            ...
                    }
                    .padding(.top, 80)
                }
            }
            Spacer()
        }
        .onAppear {
            DispatchQueue.main.asyncAfter(deadline: .now() + 5) {
                    isActive = true
                }
        }
        .fullScreenCover(isPresented: $isActive) {
            ContentView()
        }
    }
}
```

Chapter 3

Challenges Faced

3.1 Data Reset Issues

- Data set by users such as frequency and birthday, reset when the app was reopened.

3.2 AI Timer Failures

- AI sometimes failed to generate responses after the frequency timer stopped.

3.3 Background Inactivity

- The app didn't send notifications unless it was reopened.

3.4 HealthKit Permissions

- Encountered errors when requesting HealthKit authorization or accessing health data.

Chapter 4

Solutions

4.1 Challenge 1 - Data Reset Issues

- Resolved by saving user preferences (e.g., frequency, birthday) in UserDefaults

Listing 4.1: Solve Data Reset Issues

```
UserDefaults.standard.set(frequency, forKey: "fetchFrequency")
UserDefaults.standard.set(birthday, forKey: "userBirthday")
```

4.2 Challenge 2 - AI Timer Failures

- Implemented robust error handling and retries for API requests

Listing 4.2: Solve AI Timer Failures

```
private func handleAppRefresh(task: BGAppRefreshTask) {
    scheduleAppRefresh()
    let healthKitManager = HealthKitManager()

    task.expirationHandler = {
        print("Background task expired.")
        task.setTaskCompleted(success: false)
    }

    healthKitManager.fetchAndAnalyzeData { response in
        if let heartRate = healthKitManager.heartRate,
           let respiratoryRate = healthKitManager.respiratoryRate,
           let userAge = UserDefaults.standard.integer(forKey: "userAge") as
                Int? {

            let shouldNotify = !self.isHealthDataInNormalRange(age: userAge,
                 heartRate: heartRate, respiratoryRate: respiratoryRate) ||
                self.generateNormalRangeMessage
            if shouldNotify, let responseText = response {
                let content = UNMutableNotificationContent()
                content.title = "Health Update from AI"
                content.body = responseText
                content.sound = .default

                let request = UNNotificationRequest(identifier: UUID().
                    uuidString, content: content, trigger: nil)
                UNUserNotificationCenter.current().add(request) { error in
                    if let error = error {
                        print("Error adding notification: \(error.
                            localizedDescription)")
                    }
                }
                task.setTaskCompleted(success: true)
            } else {
                print("No notification required.")
                task.setTaskCompleted(success: true)
            }
        } else {
            print("Error fetching health data or missing user's age.")
            task.setTaskCompleted(success: false)
        }
    }
}
```

4.3 Challenge 3 - Background Inactivity

- Enabled required background modes (Fetch, Background processing) in Info.plist and ensured background refresh was correctly scheduled

Listing 4.3: Solve Background Inactivity

```
<key>UIBackgroundModes</key>
<array>
    <string>remote-notification</string>
    <string>fetch</string>
    <string>external-accessory</string>
    <string>processing</string>
</array>

private func registerBackgroundTasks() {
    BGTaskScheduler.shared.register(forTaskWithIdentifier: "com.bonnie.Mind-
       Pulse.fetch", using: nil) { task in
        self.handleAppRefresh(task: task as! BGAppRefreshTask)
    }
}
```

4.4 Challenge 4 - HealthKit Permissions

- Added necessary keys to Info.plist[1]

Listing 4.4: Solve HealthKit Permissions

```
<key>NSHealthShareUsageDescription</key>
<string>This app needs access to your health data to analyze and provide
    insights on your heart rate and respiratory rate.</string>

<key>NSHealthUpdateUsageDescription</key>
<string>This app needs permission to update your health data for improved
    insights.</string>
```

Chapter 5

Conclusions

This App demonstrates the potential of AI-driven proactive care for emotional and physical well-being. Through this design, it addresses a gap in health monitoring apps by combining data-driven insights with empathetic communication. The challenges faced during development were valuable learning experiences that improved the app's robustness and functionality.

The next steps could involve expanding the app's capabilities to monitor sleep data, enhancing its support for overall well-being.

In the future, the app could add one more feature allowing users to upload a portrait, which the AI would use to generate a personalized figure, giving the AI a customizable appearance that aligns with the user's preferences, creating a more engaging and relatable experience.

Bibliography

[1] Apple Developer Documentation. Healthkit, n.d. Accessed on: November 20, 2024.

[2] R. Jerath and C. Beveridge. Respiratory rhythm, autonomic modulation, and the spectrum of emotions: the future of emotion recognition and modulation. *Frontiers in Psychology*, 11, 2020.

[3] M. Mather and J. F. Thayer. How heart rate variability affects emotion regulation brain networks. *Current Opinion in Behavioral Sciences*, 19:98–104, 2018.

[4] A. Thakkar, A. Gupta, and A. De Sousa. Artificial intelligence in positive mental health: a narrative review. *Frontiers in Digital Health*, 6, 2024.