St.Lukes HRV Monitoring

Team Name: EWU St.Lukes BSCS Team

Team Members:

Tyler Woody- Developer
Austin Harrison- Developer
Brady Dullanty - Developer
Will Reese - Developer

October 2024

Contents

1 Overview 1

2 Project Description 2

3 User Stories and Scenarios 3

4 Functional Requirements 6

5 Use Case Diagrams and Use Cases 7

5.1 Use Case Diagram . . . 7

5.2 Use Cases . . . 8

6 Non-functional Requirements 9

7 Traceability Matrix (10 Points) 10

8 Evaluation of Existing Solution (10 Points) 11

9 Individual Contributions Summary 12

10 System Architecture Design 13

11 UI Design 17

12 Data Design 18

13 Environment Setup 20

List of Figures

1 System Architecture Diagram . . . 13
2 Activity Diagram for HRV Mobile App . . . 14
3 Activity Diagram for HRV Watch App . . . 15
4 Class Diagram for System Structure . . . 16
5 UI Prototype for HRV Project . . . 17
6 Data Entity Relationship Diagram . . . 18
7 Database Schema . . . 18
8 Sample User Data Table . . . 19
9 Time Section Data Table Example . . . 19
10 Medication Data Table Example . . . 19
11 GAD7 Table Example . . . 19
12 Heart Rate Data Table Example . . . 19

1 Overview

We are aiming to leverage the capabilities that the Apple Watch offers in order to monitor patients that require their heart rate variability (HRV) addressed. In particular, patients that are suffering from PTSD and other related disorders such as panic attacks. By developing a dedicated watch app accompanied by a smartphone app, we are planning to provide real time monitoring of the HRV, which is vital in understanding the physiological impact of stress and emotional triggers. The watch app will continuously track HRV data and communicate with the companion app on their smartphone.

The data will not only inform the user of their own physiological responses, but will also support potential medical research by providing anonymous data on HRV patterns associated with PTSD related disorders. To ensure data integrity, the data will be securely stored on a backend server. The data from this project can also be useful to healthcare providers when users provide explicit consent.

With the combination of real-time monitoring, securing data, and having a user-focused functionality; we are aiming to make this solution a valuable tool for individuals who will want their HRV monitored.

2 Project Description

The Heart Rate Variability Monitoring App is a cutting-edge mobile application designed to detect early signs of panic attacks and provide immediate support to users. By leveraging advanced heart rate monitoring technology, the app identifies unusual heart rate patterns that may indicate an impending medical event, alerting users to take preventive action. The app also offers tools and resources to help users navigate through panic episodes, promoting mental well-being and resilience.

Panic attacks and other serious medical events can occur without warning, often leaving individuals feeling helpless and unprepared. Unusual fluctuations in heart rate variability (HRV) can serve as critical indicators of these episodes, but current solutions often lack timely data and insights. There is a pressing need for a reliable method to monitor HRV leading up to these events, enabling users to respond proactively and safely.

The app is specifically designed for individuals who experience panic attacks, post-traumatic stress disorder (PTSD), and other anxiety-related conditions. Additionally, it caters to anyone interested in monitoring their heart rate for health and wellness purposes, making it a versatile tool for various users.

Using our app, users can easily view their heart rate data, gaining insights into their physiological state throughout the day. The application also provides timely notifications when it detects abnormal heart rate patterns, empowering users to take action before a panic attack escalates. In moments of distress, the app acts as a supportive companion, offering calming techniques, breathing exercises, and cognitive strategies to help users manage their anxiety and regain control.

By combining heart rate monitoring with immediate support resources, the Heart Rate Variability Monitoring App aims to enhance the quality of life for individuals facing panic attacks and related challenges, fostering a sense of safety and well-being.

3 User Stories and Scenarios

User Story 1

U1. As someone with PTSD or a related condition, I want the Apple Watch app to monitor my heart rate variability (HRV) in real-time, so I can track potential triggers or episodes.

Feature: Monitor HRV

Scenario: Real-time Monitoring of HRV

- Given I have opened the app on my watch
- When the app starts monitoring my HRV in real-time
- Then I can see my HRV values displayed on my phone
- And I receive notifications if abnormal readings are detected

Scenario: Notification of HRV Event

- Given the app is already open and monitoring my HRV
- When my HRV data falls into a range associated with a medical ”episode”
- Then the app should send a notification to my watch/phone
- And I should receive different recommendations based on the episode

User Story 2

U2. As a healthcare provider, I want to access an individual’s HRV data on the user’s device with their consent.

Feature: Data For Doctors

Scenario: No portal for doctors exists

- Given the user has their HRV data on their phone
- When I request consent from the patient to view the data
- And the user opens the HRV app on their device
- Then the user displays the data for the doctor

User Story 3

U3. As a user, I want to make sure the data that is collected from me is properly made anonymous and stored securely on the back end.

Feature: Secure Data Storage

Scenario: Secure storage of HRV data

- Given I am a user of the HRV app, and it is collecting data from my watch
- When my HRV data is transferred to the back end
- Then the data should be encrypted and anonymized before being stored
- And I should receive a confirmation that my data is stored securely

Scenario: Data access is restricted to authorized users only

- Given my HRV data is stored securely on the back end
- When an unauthorized entity tries to access the data
- Then access should be denied
- And an alert should be triggered for the unauthorized access attempt

User Story 4

U4. As a person who suffers from panic attacks, I would like to see what happened to my heart rate and HRV after the event.

Feature: Toggle Live Notifications Until After Event

Scenario: User toggles event suppression

- Given the user has HRV issues
- When the user normalizes
- Then the user is prompted about the potential attack

Scenario: User does not toggle event suppression

- Given the user has HRV issues
- When a potential or current attack is detected
- Then the user is informed of the attack
- And the user is prompted for confirmation

User Story 5

U5. As someone inexperienced with panic attacks, like a child, I want the app to handle possible events without needing my confirmation.

Feature: Toggle Event Confirmation

Scenario: Event confirmation is turned off

- Given I might be having a panic attack
- When event confirmation is turned off
- Then the app records the event without asking if I am having a panic attack

Scenario: Event confirmation is turned on

- Given I might be having a panic attack
- When event confirmation is turned on
- Then the app asks if I am having a panic attack
- And I can choose to confirm or ignore the prompt

User Story 6

U6. As someone who values my privacy, I never want any of my data off my devices, anonymous or not.

Feature: Export Anonymized Data Only with Consent

Scenario: I toggle my data being sent

- Given I value my privacy
- When data transmission is toggled off
- Then my data never leaves my device

Scenario: I want my data sent anonymously to help others

- Given I agree to anonymous data sharing
- When data is sent at intervals
- Then my device sends new data to a back end storage system I cannot access

User Story 7

U7. As someone who is giving my watch as a gift, I want to remove my old data.

Feature: Delete All On-Device Data Permanently

Scenario: I hit the delete button

- Given I am preparing to gift my watch
- When I press the delete button
- Then I am warned about the ramifications and permanence
- And the data is permanently removed from this device

User Story 8

U8. As a patient who suffers from attacks, I want to receive soothing messages when I'm having an event so that I can calm down.

Feature: Soothing Assistance

Scenario: Receiving Soothing Messages During an Attack

- Given I have received a notification of an attack
- When I acknowledge the notification
- Then the app displays soothing messages and guidance on my watch

4 Functional Requirements

Watch App

Functional Requirement	**Description/Specification**
[FR-1(U1-U8)] Monitor Heart Rate	Continuously collect heart rate data using the Apple Watch's built-in sensors Convert heart rate data into interbeat intervals (IBI).
Priority	Level 0 (Essential)
[FR-2(U1,U2,U4,U5,U8)] Real-Time Alerts	The system must detect and display events based on HRV deviations from baseline
Priority	Level 0 (Essential)
[FR-3(U1,U2,U4,U5,U8)] HRV Metrics Calculation	Compute standard HRV metrics (e.g., SDNN, RMSSD, PNN50) from IBI data.
Priority	Level 0 (Essential)

Companion Mobile App

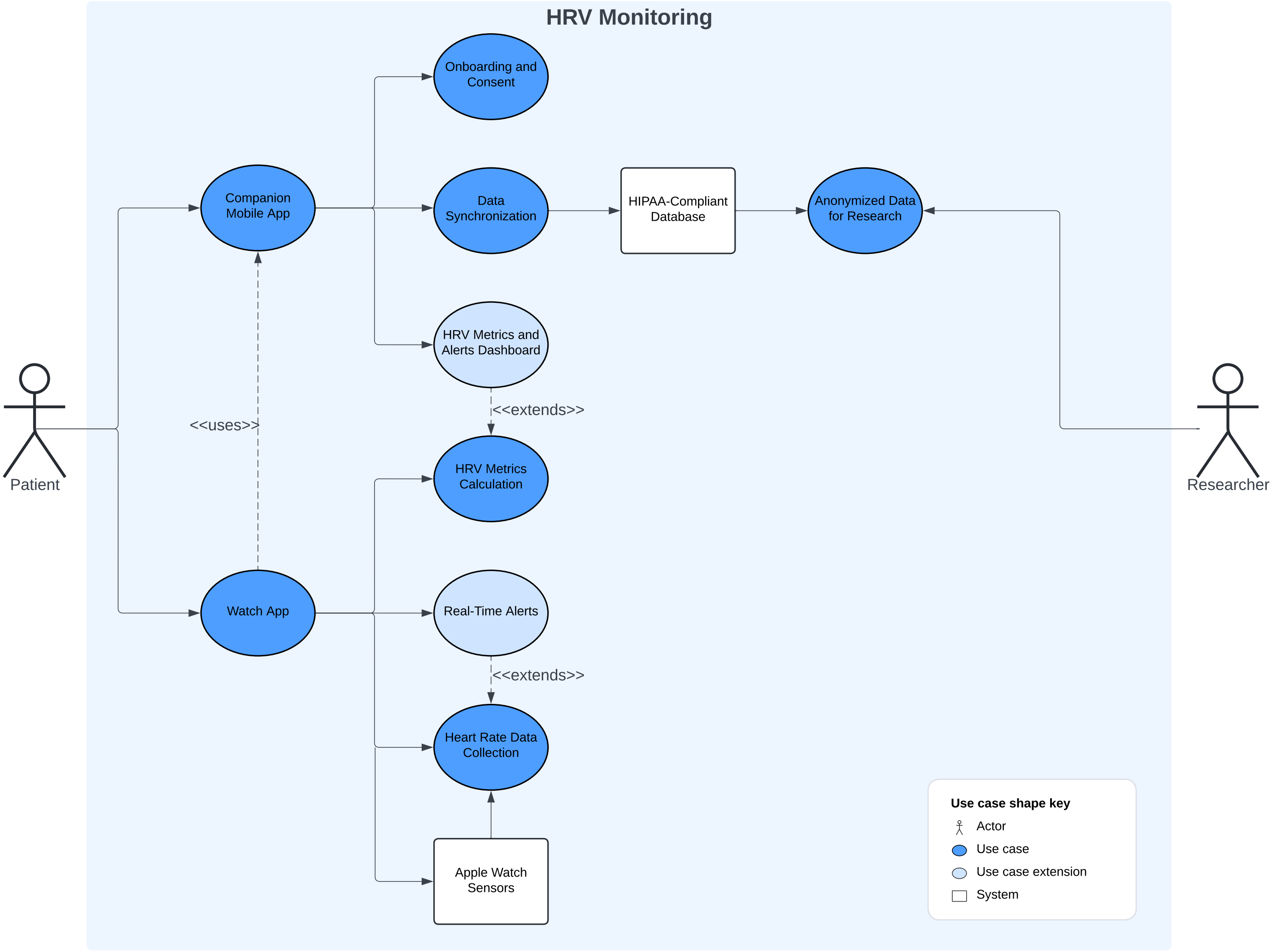
Functional Requirement	**Description/Specification**
[FR-4(U1,U2,U4,U5,U8)] User Interface	Customers must be able to view all metrics, events, and history on the phone.
Priority	Level 0 (Essential)
[FR-5 (U1-U6,U8)] Data Synchronization:	Sync data between the Apple Watch and the mobile app securely.
Priority	Level 0 (Essential)
[FR-6 (U1-U6,U8)] Consent and Onboarding:	Sync data between the Apple Watch and the mobile app securely.
Priority	Level 0 (Essential)

Data Storage and Management

Functional Requirement	**Description/Specification**
[FR-7(U3)] Secure Database	Aggregate all collected data anonymously into a centralized database with HIPPA compliance.
Priority	Level 0 (Essential)
[FR-8(U3)] Data Processing	Preprocess and clean incoming data for future machine learning purposes.
Priority	Level 0 (Essential)
[FR-9(U7)] Data Deletion	Data on device should be able to be cleared easily.
Priority	Level 0 (Essential)

5 Use Case Diagrams and Use Cases

5.1 Use Case Diagram



5.2 Use Cases

Use Case 1: Monitor and Display HRV

- **Actors**: Patient
- **Preconditions**: User has the app open and running.
- **Postconditions**: HRV is being monitored and displayed.
- **Main Flow**:
  - Watch starts collecting HRV data from the patient.
  - Watch sends data to the companion app.
  - Companion app uses data to detect any possible events.
  - Companion app displays HRV data to the user.
- **Alternative Flows**:
  - **If the app detects abnormal HRV levels**:
    - The app sends a notification to the user's watch and phone, alerting them to a potential medical event.
    - The app provides calming messages or breathing exercises to help manage symptoms.
  - **If the user has allowed for anonmyzed data transfer**:
    - The app automatically sends all possible anonymized data.
  - **If the user has turned off event confirmation prompts**:
    - The app automatically records the event without asking if the user is experiencing a panic attack, suitable for inexperienced users such as children.
  - **If the user has turned off live event notifications** :
    - The app automatically records the event without informing the user, prompting for feedback after the event.
  - **If the patient's watch is out of connectivity range with the companion app**:
    - HRV data is cached locally on the watch until connectivity is restored.
    - Data synchronization resumes automatically when connectivity is reestablished.
- **Related Requirements**: Real-time alerts, notification handling, support resources display.

Use Case 2: Analyze Anonymous Data

- **Actors**: Researcher
- **Preconditions**: Authorized user is logged into the database.
- **Postconditions**: Authorized user can view and export data for research purposes.
- **Main Flow**:
  - Authorized user logs into the database.
  - User views anonymized HRV data collected from all consenting users.
  - User can export data for further research.
- **Alternative Flows**:
  - **N/A**:
- **Related Requirements**: Secure data access, patient consent management.

6 Non-functional Requirements

Non-Functional Requirement	**Description**
[NFR-1] Security	The app should securely encrypt and send HRV data to our secure back-end storage container.
[NFR-2] Security	The app should securely send data from the watch to the companion app.
[NFR-3] Performance	The companion app on the phone should display the data being collected from the watch with minimal delay. (undetermined how long delay will actually be)
[NFR-4] Scalability	The users phone and watch are doing all of the collecting and processing, so we only need to think about data storage. The storage at first should be able to hold a years worth of data for 500 users.
[NFR-5] Usability	The app should be easy to navigate for all users, regardless of technical skill level.
[NFR-6] Availability	The system shall have an uptime of 99.95%.

7 Traceability Matrix (10 Points)

The traceability matrix maps functional requirements to corresponding use cases and user stories, ensuring complete coverage. The table below maps functional requirements to their respective use cases and user stories.

Function Requirement	**Use Case**	**User Story**	**Priority**
FR-1: Monitor Heart Rate	UC1: Monitor and Display HRV	US1-US8	Priority 0
FR-2: Real-Time Alerts	UC1: Monitor and Display HRV	US1, US2, US4, US5, US8	Priority 0
FR-3: HRV-Metrics Calculation	UC1: Monitor and Display HRV	US1, US2, US4, US5, US8	Priority 0
FR-4: User Interface	UC1: Monitor and Display HRV	US1 ,US2, US4, US5, US8	Priority 0
FR-5: Data Synchronization	UC1: Monitor and Display HRV	US1-US6, US8	Priority 0
FR-6: Consent and Onboarding	UC1: Monitor and Display HRV	US1-US6, US8	Priority 0
FR-7: Secure Database	UC2: Analyze Anonymous Data	US3	Priority 0
FR-8: Data Processing	UC2: Analyze Anonymous Data	US3	Priority 0
FR-10: Data Deletion	UC1: Monitor and Display HRV	US7	Priority 0

8 Evaluation of Existing Solution (10 Points)

The existing solution for monitoring Heart Rate Variability (HRV) through an Apple Watch faced significant challenges, primarily in the consistent and accurate retrieval of heart rate data. The initial attempts may not have fully leveraged Apple's HealthKit API, which is crucial for accessing physiological data streams from user devices. This shortfall likely contributed to the project's inability to reliably monitor HRV, a key indicator for preemptively identifying panic attacks and other anxiety-related episodes.

For enhancements, adopting a more robust approach to data integration with HealthKit is recommended to ensure a steady and reliable flow of HRV data. Improving the app's architecture to support efficient data handling and real-time processing will also be critical. Additionally, enhancing the user interface to make it more intuitive and engaging could significantly improve user satisfaction and retention. Implementing a comprehensive testing strategy, particularly focusing on automated tests, will help in maintaining high standards of quality and reliability. Finally, integrating advanced notification systems that can adapt based on user response and history will make the app not only a tool for monitoring but also a proactive component in managing user health. These improvements aim to transform the Heart Rate Variability Monitoring App into a dependable resource for individuals facing significant health challenges, thus enhancing their quality of life and well-being.

9 Individual Contributions Summary

For this task, please include a brief description of contributions made by each team member.

Example

- Austin Harrison: Created Non-functional Requirements, use cases, attended all meetings, wrote user stories, helped create and maintain documents.
- Tyler Woody: Created use case diagram, attended all meetings, wrote user stories, helped create and maintain documents.
- Brady Dullanty: Deep research and implementation of backend, attended all meetings, created, contributed, and maintained all project documents.
- William Reese: Designed slides for presentation, collaborated on details for back-end and system design, contributed to documents, and project planning.

10 System Architecture Design

- **System Architecture Diagram**:

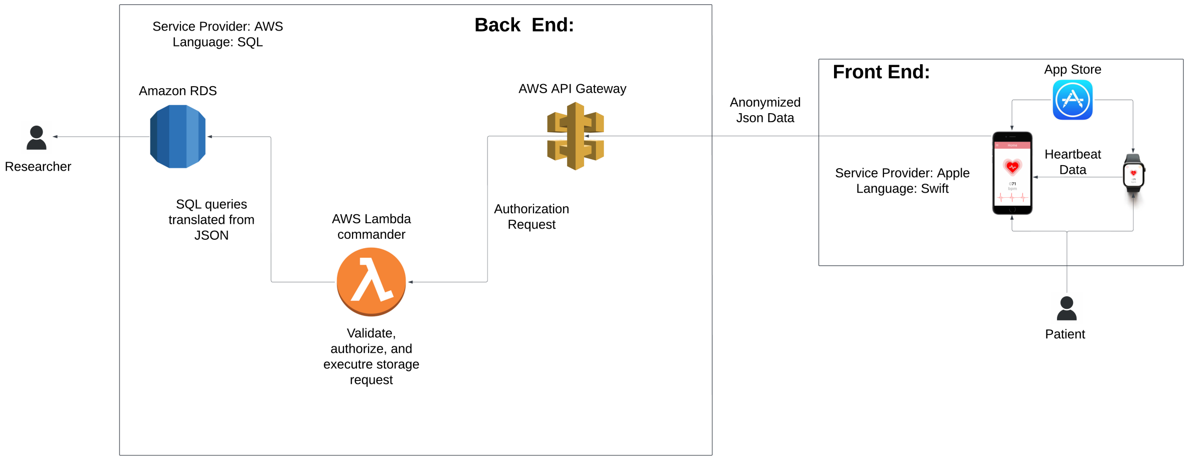


Figure 1: You can get the app on the watch and phone from the app store. The watch uses healthkit ai to record heart rate info, and detect events. Information is sent to the users phone, which then does more intense analysis and translation, as well as sending anonymized information to the API Gateway. The Backend filters and authorizes requests through a front facing AWS API gateway, with a lambda authorization function that checks several factors to authenticate. If authenticated, the api gateway translates the request in queries, which parse the expected json, storing data on the RDS database.

- **UML Diagrams**:
  - **Behavioral Diagrams**:

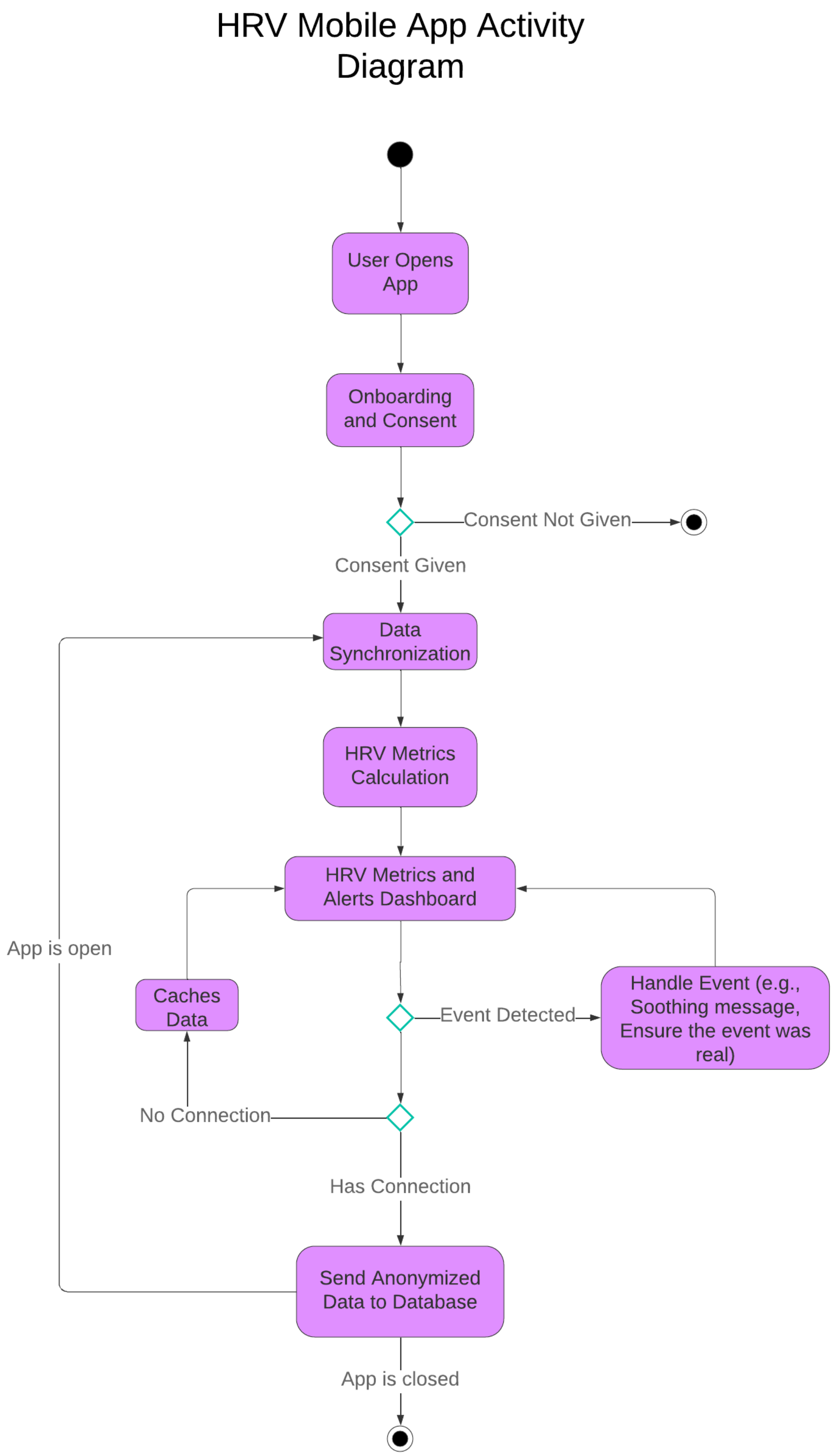


Figure 2: Activity Diagram for HRV Mobile App

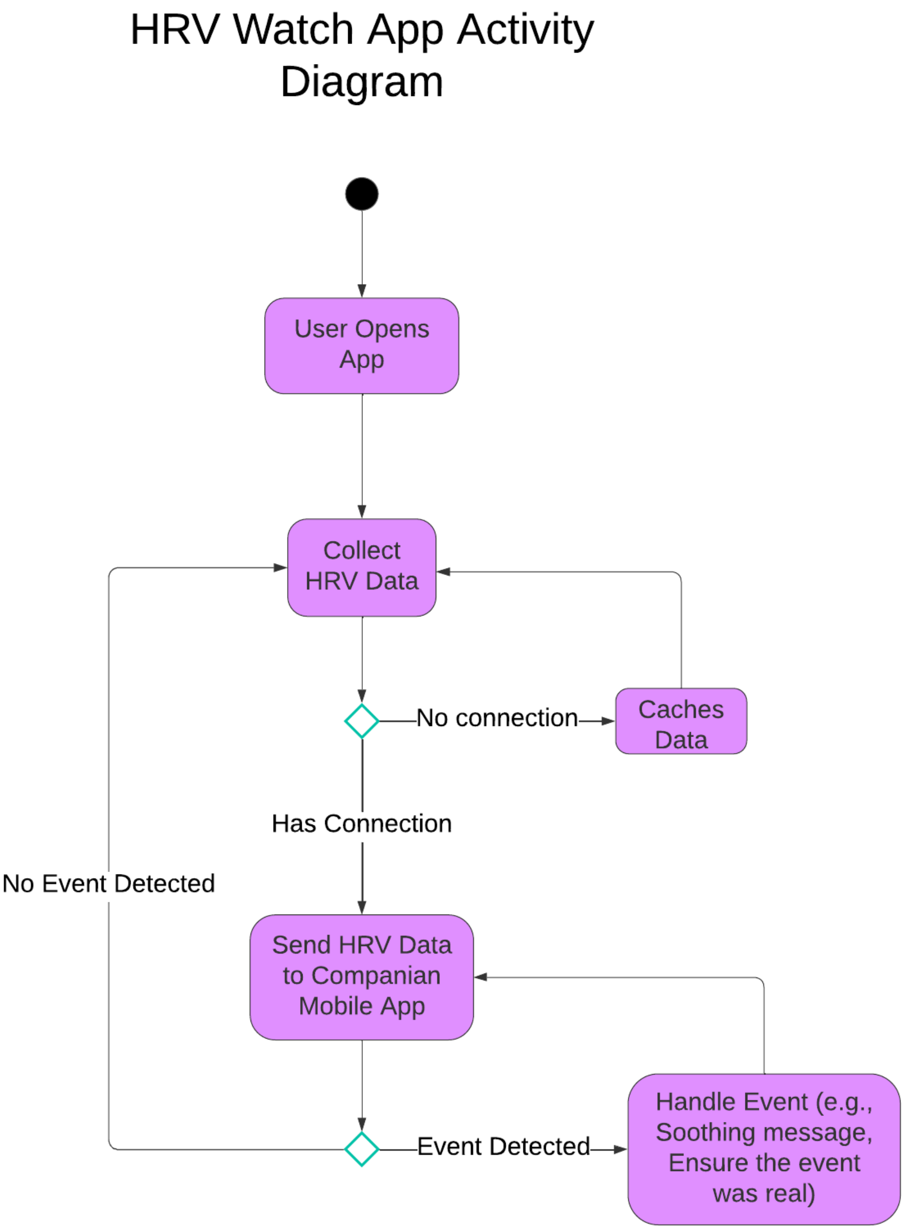
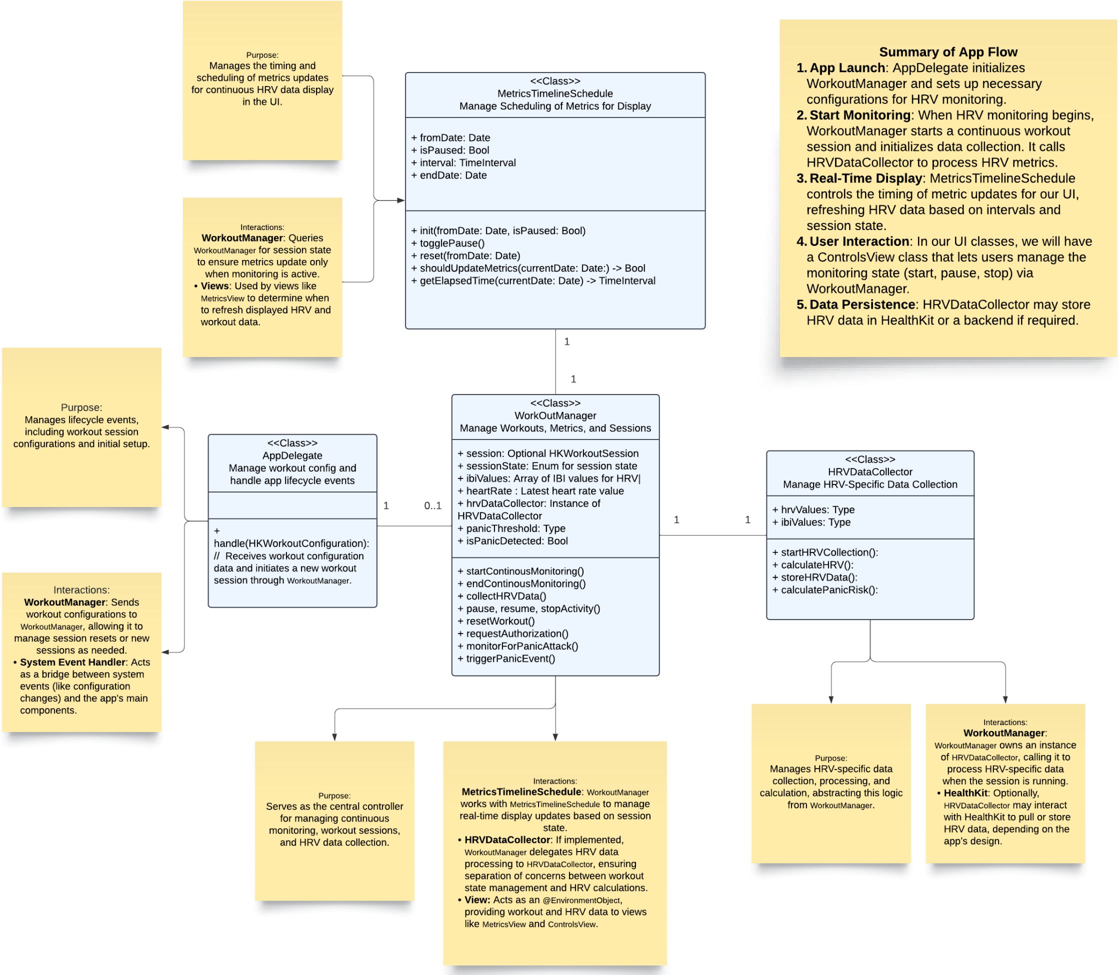

HRV Watch App Activity Diagram
User Opens App
Collect HRV Data
No connection
Caches Data
Has Connection
No Event Detected
Send HRV Data to Companian Mobile App
Event Detected
Handle Event (e.g., Soothing message, Ensure the event was real)

Figure 3: Activity Diagram for HRV Watch App

– **Structural Diagram**:



<<Class>>
MetricsTimelineSchedule
Manage Scheduling of Metrics for Display
+ fromDate: Date
+ isPaused: Bool
+ interval: TimeInterval
+ endDate: Date
+ init(fromDate: Date, isPaused: Bool)
+ togglePause()
+ reset(fromDate: Date)
+ shouldUpdateMetrics(currentDate: Date:) -> Bool
+ getElapsedTime(currentDate: Date) -> TimeInterval
1
1
<<Class>>
WorkOutManager
Manage Workouts, Metrics, and Sessions
+ session: Optional HKWorkoutSession
+ sessionState: Enum for session state
+ ibiValues: Array of IBI values for HRV|
+ heartRate : Latest heart rate value
+ hrvDataCollector: Instance of HRVDataCollector
+ panicThreshold: Type
+ isPanicDetected: Bool
+ startContinousMonitoring()
+ endContinousMonitoring()
+ collectHRVData()
+ pause, resume, stopActivity()
+ resetWorkout()
+ requestAuthorization()
+ monitorForPanicAttack()
+ triggerPanicEvent()
<<Class>>
AppDelegate
Manage workout config and handle app lifecycle events
+ handle(HKWorkoutConfiguration): // Receives workout configuration data and initiates a new workout session through WorkoutManager.
1
0..1
1
1
<<Class>>
HRVDataCollector
Manage HRV-Specific Data Collection
+ hrvValues: Type
+ ibiValues: Type
+ startHRVCollection():
+ calculateHRV():
+ storeHRVData():
+ calculatePanicRisk():

Purpose:
Manages the timing and scheduling of metrics updates for continuous HRV data display in the UI.

Interactions:
WorkoutManager: Queries WorkoutManager for session state to ensure metrics update only when monitoring is active.
- **Views**: Used by views like MetricsView to determine when to refresh displayed HRV and workout data.

Summary of App Flow

1. **App Launch**: AppDelegate initializes WorkoutManager and sets up necessary configurations for HRV monitoring.
2. **Start Monitoring**: When HRV monitoring begins, WorkoutManager starts a continuous workout session and initializes data collection. It calls HRVDataCollector to process HRV metrics.
3. **Real-Time Display**: MetricsTimelineSchedule controls the timing of metric updates for our UI, refreshing HRV data based on intervals and session state.
4. **User Interaction**: In our UI classes, we will have a ControlsView class that lets users manage the monitoring state (start, pause, stop) via WorkoutManager.
5. **Data Persistence**: HRVDataCollector may store HRV data in HealthKit or a backend if required.

Purpose:
Manages lifecycle events, including workout session configurations and initial setup.

Interactions:
WorkoutManager: Sends workout configurations to WorkoutManager, allowing it to manage session resets or new sessions as needed.
- **System Event Handler**: Acts as a bridge between system events (like configuration changes) and the app's main components.

Purpose:
Serves as the central controller for managing continuous monitoring, workout sessions, and HRV data collection.

Interactions:
MetricsTimelineSchedule: WorkoutManager works with MetricsTimelineSchedule to manage real-time display updates based on session state.
- **HRVDataCollector**: If implemented, WorkoutManager delegates HRV data processing to HRVDataCollector, ensuring separation of concerns between workout state management and HRV calculations.
- **View:** Acts as an @EnvironmentObject, providing workout and HRV data to views like MetricsView and ControlsView.

Purpose:
Manages HRV-specific data collection, processing, and calculation, abstracting this logic from WorkoutManager.

Interactions:
WorkoutManager: WorkoutManager owns an instance of HRVDataCollector, calling it to process HRV-specific data when the session is running.
- **HealthKit**: Optionally, HRVDataCollector may interact with HealthKit to pull or store HRV data, depending on the app's design.

Figure 4: Class Diagram for System Structure

11 UI Design

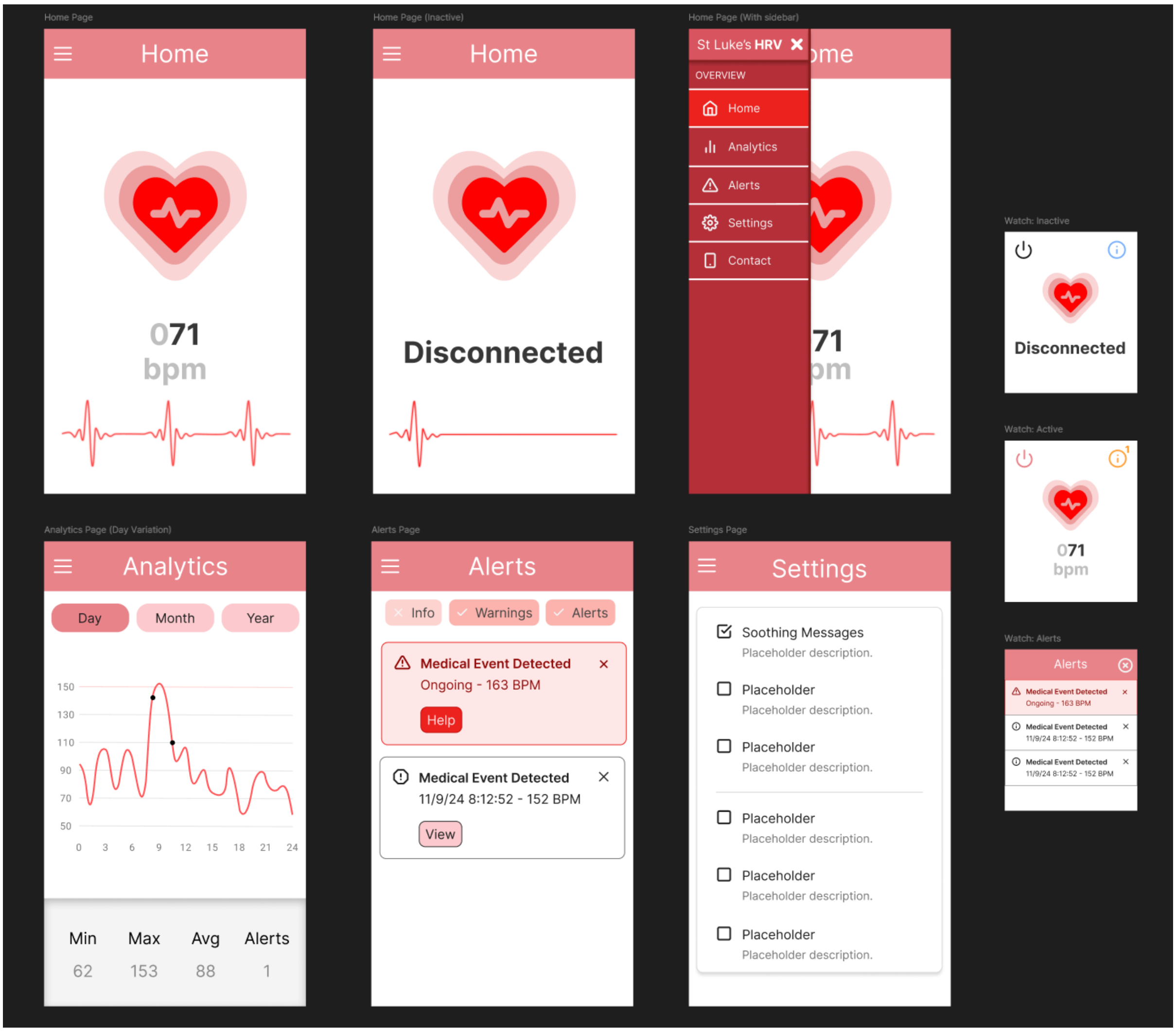


Figure 5: UI Prototype for HRV Project

12 Data Design

- **Data Flow**:

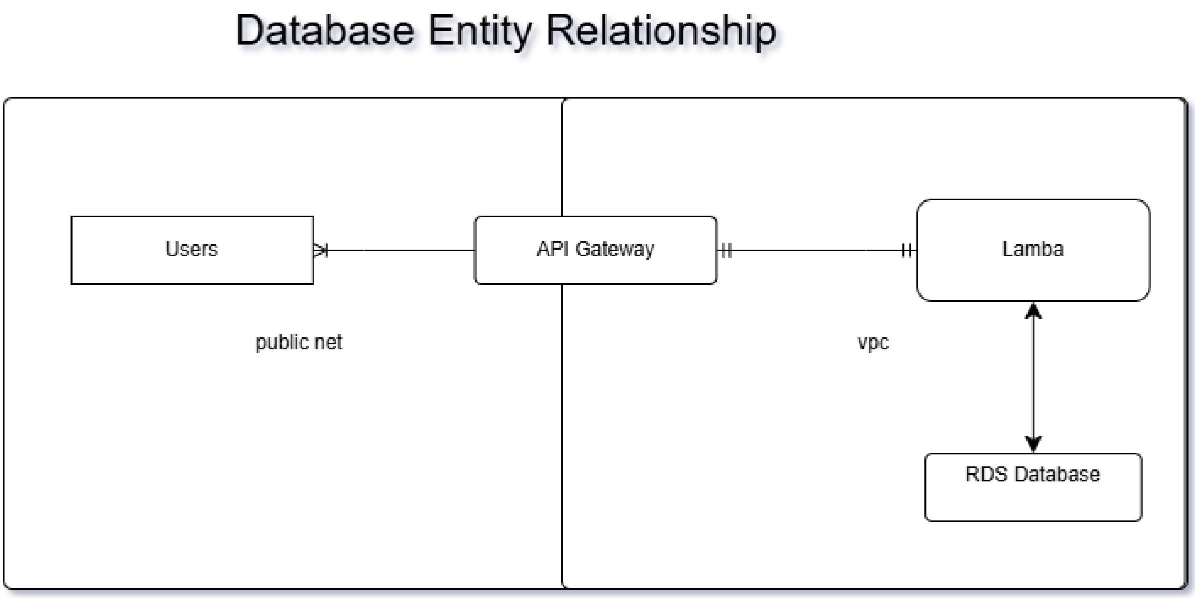
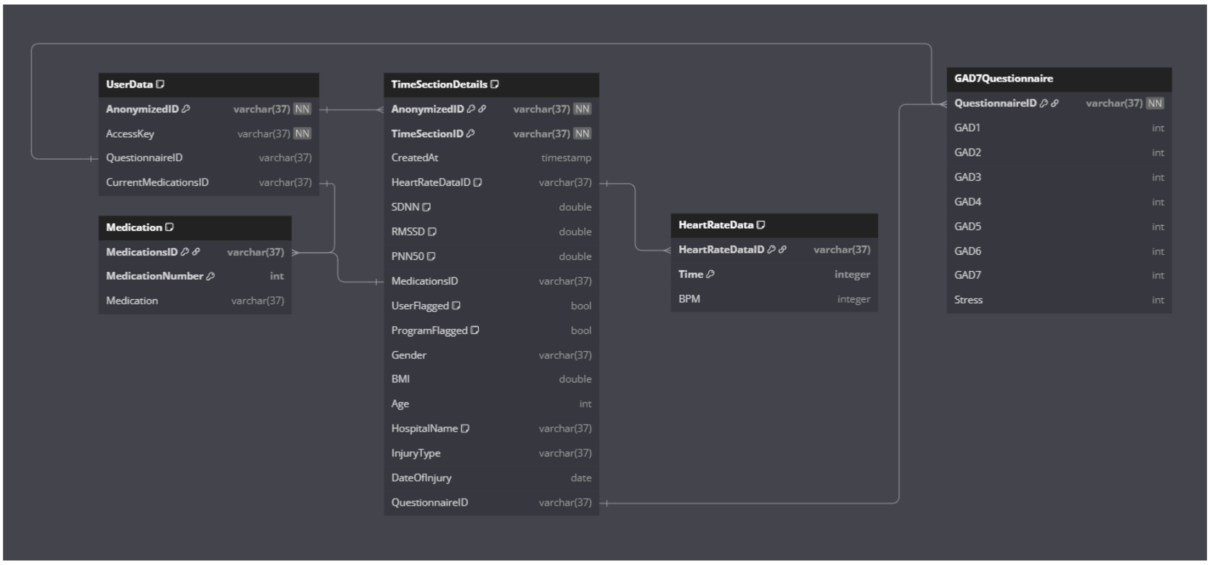


Figure 6: Users use the front end application on their mobile device to send storage requests, if they gave consent, through the API gateway and into our backend area, where a lambda and RDS database comunicate and store data.

- **Database Schema**:



UserData		
AnonymizedID	varchar(37)	NN
AccessKey	varchar(37)	NN
QuestionnaireID	varchar(37)	
CurrentMedicationsID	varchar(37)	

Medication		
MedicationsID	varchar(37)	
MedicationNumber	int	
Medication	varchar(37)	

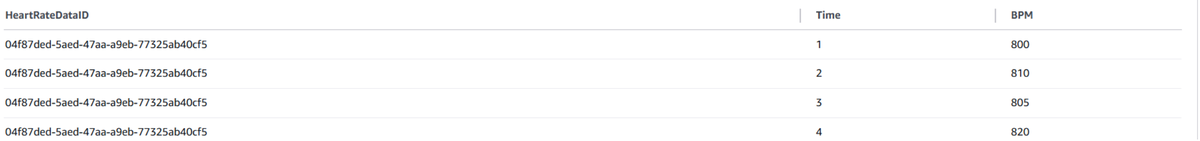
TimeSectionDetails		
AnonymizedID	varchar(37)	NN
TimeSectionID	varchar(37)	NN
CreatedAt	timestamp	
HeartRateDataID	varchar(37)	
SDNN	double	
RMSSD	double	
PNN50	double	
MedicationsID	varchar(37)	
UserFlagged	bool	
ProgramFlagged	bool	
Gender	varchar(37)	
BMI	double	
Age	int	
HospitalName	varchar(37)	
InjuryType	varchar(37)	
DateOfInjury	date	
QuestionnaireID	varchar(37)	

HeartRateData		
HeartRateDataID	varchar(37)	
Time	integer	
BPM	integer	

GAD7Questionnaire		
QuestionnaireID	varchar(37)	NN
GAD1	int	
GAD2	int	
GAD3	int	
GAD4	int	
GAD5	int	
GAD6	int	
GAD7	int	
Stress	int	

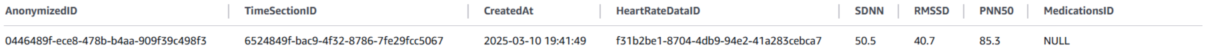
Figure 7: The userdata table helps by using an anonymized ID and access key to ensure authentication and authorization. It also holds a dynamic MedicationsID, that changes when medications are changed, as well as being copied over when creating a TimeSectionDetails entry, as a snapshot of their medications at that time. The Time Section Details table is our main table, and has lots of references to foreign indexes and keys in the other tables. The other tables, (Medication, IBIData, and heart-rate data) all store an snapshot in time for the TimeSectionsDetails to refer to.

– Table Example Data:



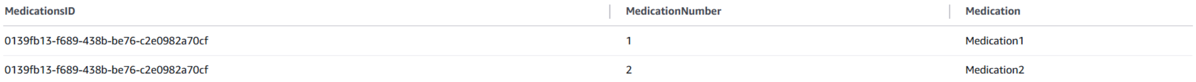
HeartRateDataID	Time	BPM
04f87ded-5aed-47aa-a9eb-77325ab40cf5	1	800
04f87ded-5aed-47aa-a9eb-77325ab40cf5	2	810
04f87ded-5aed-47aa-a9eb-77325ab40cf5	3	805
04f87ded-5aed-47aa-a9eb-77325ab40cf5	4	820

Figure 8: Sample User Data Table



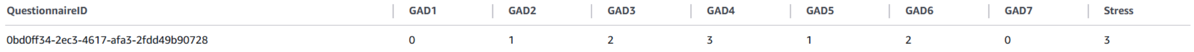
AnonymizedID	TimeSectionID	CreatedAt	HeartRateDataID	SDNN	RMSSD	PNN50	MedicationsID
0446489f-ece8-478b-b4aa-909f39c498f3	6524849f-bac9-4f32-8786-7fe29fcc5067	2025-03-10 19:41:49	f31b2be1-8704-4db9-94e2-41a283cebca7	50.5	40.7	85.3	NULL

Figure 9: Time Section Data Table Example



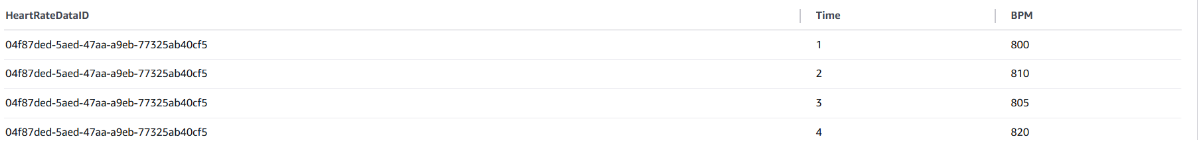
MedicationsID	MedicationNumber	Medication
0139fb13-f689-438b-be76-c2e0982a70cf	1	Medication1
0139fb13-f689-438b-be76-c2e0982a70cf	2	Medication2

Figure 10: Medication Data Table Example



QuestionnaireID	GAD1	GAD2	GAD3	GAD4	GAD5	GAD6	GAD7	Stress
0bd0ff34-2ec3-4617-afa3-2fdd49b90728	0	1	2	3	1	2	0	3

Figure 11: GAD7 Table Example



HeartRateDataID	Time	BPM
04f87ded-5aed-47aa-a9eb-77325ab40cf5	1	800
04f87ded-5aed-47aa-a9eb-77325ab40cf5	2	810
04f87ded-5aed-47aa-a9eb-77325ab40cf5	3	805
04f87ded-5aed-47aa-a9eb-77325ab40cf5	4	820

Figure 12: Heart Rate Data Table Example

13 Environment Setup

- **Local Development Environment**:
  - Developing with a Mac is necessary, so We acquired an Apple Developers License, and we also secured an AWS account from our sponsor.
- **Source Control**:
  - We used Github as our source Control.
- **Testing and Debugging Tools**:
  - XCodeprovides simulation as well as programming capabilities.
  - c# allowed us to use testing libraries
  - AWS has built in testing for the lambda and aurora services.
- **Documentation**:
  - Github has several readme's, and the code is commented.