Software Engineering Project
Group 20

Panic Attack Monitoring System

Beep Beep

Joseph Chotard - jpc844@student.bham.ac.uk - 1935844
Changhyun Kim- cxk858@student.bham.ac.uk - 1940758
Yueyi Wang - yxw835@student.bham.ac.uk - 1940014
Selma Kander - sxk1093@student.bham.ac.uk - 1935950
Jonathan Caines - jec845@student.bham.ac.uk - 1937445
Tudor Catalin - cxt879@student.bham.ac.uk - 1937979
Joe Feehily - jmf812@student.bham.ac.uk - 1936411

-

Introduction

The aim of this project is to conceive an accurate and accessible application targeting panic attacks. The application tracks the user's health data and alerts them of potential panic attacks. In the eventuality of a panic attack, the app will perform specific actions predefined by the user. It will also provide a comprehensive analysis on the frequency and location of panic attacks upon the user's request. We will call the app Beep Beep. Below are some assumptions made about subsystems the application uses as well as the scope of the project.

Here are some assumptions we made regarding our app:

- The user has a phone running Android or iOS that can run the app
- The user has wearable tech connected to the device with the app that can, at least, monitor the wearer's heart rate. Breathing and trembling will also be considered if they can be monitored
- Beep Beep has access to servers hosted on [hosting provider] that securely store the user's login information as well as their panic attack history on a database.

The app constantly runs in the background of the device and monitors input from the wearable tech. This input can be the user's heart rate, their breathing rate as well as their trembling. Should a spike in all of those be detected, an alert will be displayed to the user asking them to confirm if they're having a panic attack. In the case of a confirmed panic attack, actions predefined by the user will be executed. When having a panic attack, the app will log the time and location for future reference. The app will make all time and location info available to the user for them to track where and when their panic attacks are happening to see if there are any patterns.

According to *American Psychiatric Association (2013), Diagnostic and Statistical Manual of Mental Disorders (5th ed.),* around 3% of the European population has a panic attack each year. Furthermore, this study also showed that the occurred mainly in people between 20 and 65. According to pew research centre, the demographic with the highest smartphone ownerships is the 18-65 demographic which goes from 96% between 18 and 29 to 79% between 50 and 64. We clearly see that the demographic most affected by panic attacks is also the demographic that has the most access to smartphones. This makes a smartphone app a clear choice for targeting panic attacks.

According to our estimates, there are approximately 22 000 000 people suffering from panic attacks in Europe. If 1% of people suffering from Panic Attacks download the app, our infrastructure needs to be able to handle 220 000 users without affecting performance.

Requirement Analysis

Functional Requirements

User Authentication

- The system must allow the user access to the application after their login details have been verified
- The system must allow the user to change their password, through email, if they've forgotten it/failed to login three times.

- The system must allow the use of two factor authentication (2FA)

Account Information

- The system must allow the user to review their saved account information
- The system must allow the user to update their account information including
    - Their pre-set contact messages
    - Usage preferences
        - Enabling the application to turn on mobile data in the event they don't have a WIFI connection
        - Enabling automatic contact function
        - Enabling location to be logged
        - Enabling time to be logged
        - Language preference
        - The means of communication

• The system must allow the user to delete their account and all information stored with their account

Dashboard

- The system must allow the user to see the readings the application is monitoring
    - The system must allow the user to see their heartbeat reading
    - The system must allow the user to see their movement reading
    - The system must allow the user to see their breathing reading
- The system must allow the user to see past panic attack locations
- The system must allow the user to see past panic attack times
- The system must allow the user to log out of the application

Contact

- The system must allow the contact of the user's predefined contacts through whichever means is preferred by the user
    - The system must be able to contact the user's trusted contacts through text message
    - The system must be able to contact the user's trusted contacts through a phone call

Non-Functional Requirements

Monitoring and Analysis

- The system must be able to analyse the user's heartbeat constantly
    - If the user's heart rate exceeds 120bpm, the system should respond
- The system must be able to analyse the user's movement if available within 2 seconds
    - The system must be able to detect if the user is shaking and respond within 3 sec
- The system must be able to analyse the user's breathing if available within 2 seconds
    - The system must be able to respond if the user's respiration rate is above 27 breaths per minute

Data Storage/Data governance

- The system must only store information required for functionality

    - Login details
    - Location history of panic attacks
    - Time history of panic attacks
    - The system must abide by all local legislation

The system must store information in accordance with that region’s data protection laws

Safety

- The system must have a back-up database
- The system must sync the database and back-up database when a new user signs up or every day

Security

- The system must only allow users to access their account information securely if they’re logged in
- The system must ensure the safe storage of user data using Argon2id hash and a symmetric encryption algorithm like AES
    - The system must ensure users’ data can’t be lost or manipulated
- The system must ensure a secure connection between the user and the system through SSL
    - The system must ensure a secure connection between the app and account information database
- The system must lock the account for 2 hours and retain its information if the login has been failed 3 times

Scalability

- The servers must be able to handle 220 000 users at the same time without affecting performance
    - Users should be able to use the system concurrently without affecting performance
- The system should be able to handle an increased number of users
    - The system should have multiple identical servers with the same functionality
    - The system should control server traffic and prevent server overload by splitting the user load between identical servers

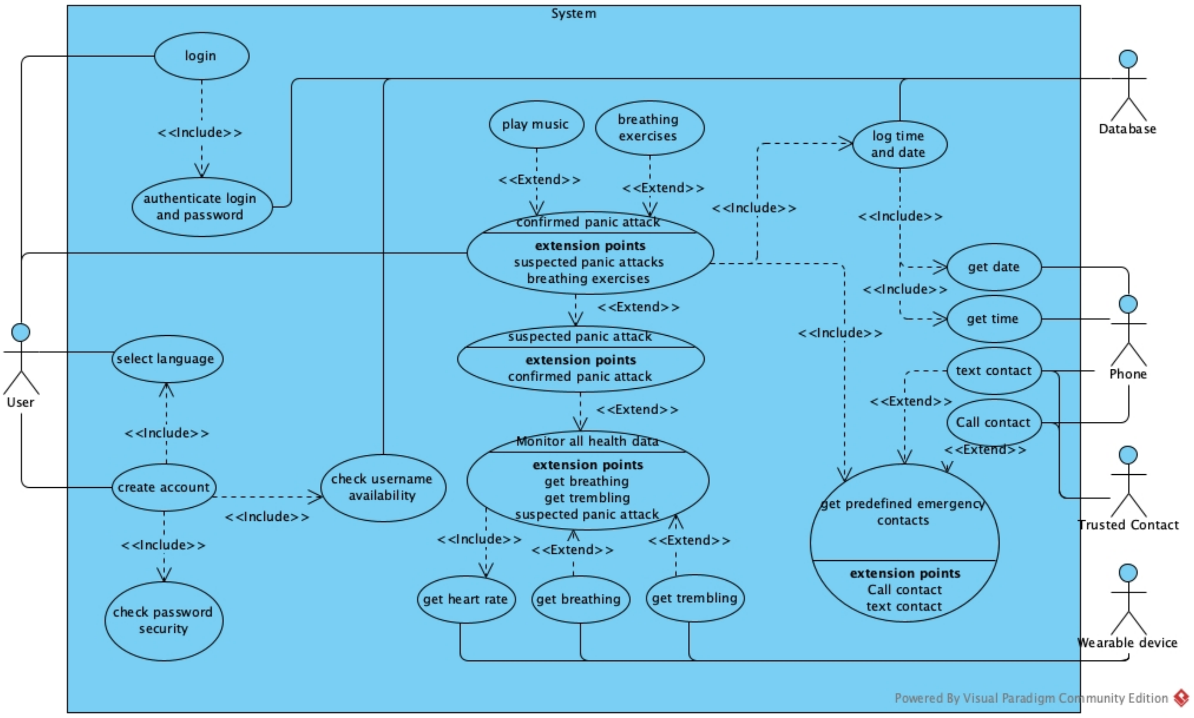
Process Efficiency

- The system should minimise analysis time
    - The analysis time should not exceed 500ms

Usability

- The system must have an on-screen help option
- The system should notify the user when they’re offline

Use Case Diagram



Non trivial Use Case documentation

Case: The app notices that the user has a confirmed panic attack

Pre-conditions

- The user has an account already and the account stays in logged in
- The user already selected preferred language.
- The user’s phone is connected to the user’s wearable device.
- The user’s phone is connected to a network
- The app is giving breathing exercise and sending emergency messages when the user has a panic attack
- The wearable device can detect heart rate, breathing, trembling etc.
- The app has access to user location
- The app has access to wireless data (WLAN and Cellular)
- Allow Notifications for this app

Flow of events

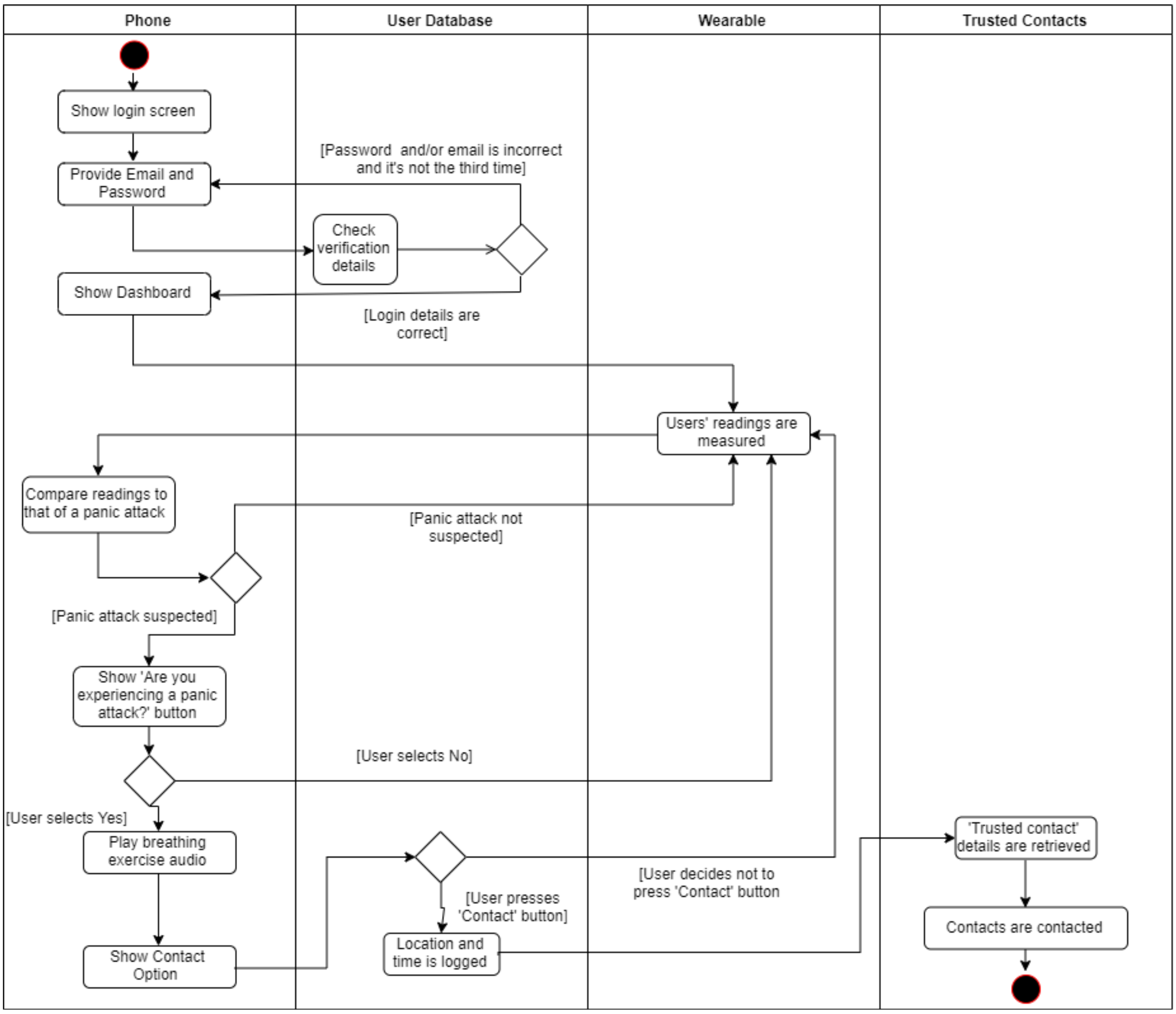
1. The app is running in the background of the phone
2. The user has a panic attack
3. The app suspects that the user has a panic attack according to heart rate data a, shaking breathing data provided by the wearable device.
4. The app asks whether the user is having a panic attack
   - If the user answers no, the system continues to monitor health data
   - If the user answers yes, the system moves on to the next step
5. The system contacts pre-defined contacts
   - The system can call the contacts or text them through sms
6. The app displays breathing exercises along with audio instructions

7. The app stores the location and time data of which the panic attack happens and stores the information on the server
8. The app asks the user if they still require breathing exercises
    - If the user answers yes, the app executes the breathing exercise once more. In this case, making emergency contact will not be executed.
    - If the user answers no, the app stops the breathing exercises
9. The app recommends the user go get medical assistance

Post-condition

- The database should update the user’s panic attack log.
- The user is no longer having a panic attack.
- The system has notified the predefined people by messages or phone calls.
- The app should update the dashboard to show past panic attacks log including the current one.
- The app should not be terminated until the user logs out.

Activity Diagram



Phone
User Database
Wearable
Trusted Contacts
Show login screen
Provide Email and Password
[Password and/or email is incorrect and it's not the third time]
Check verification details
Show Dashboard
[Login details are correct]
Users' readings are measured
Compare readings to that of a panic attack
[Panic attack not suspected]
[Panic attack suspected]
Show 'Are you experiencing a panic attack?' button
[User selects No]
[User selects Yes]
Play breathing exercise audio
Show Contact Option
[User presses 'Contact' button]
Location and time is logged
[User decides not to press 'Contact' button
'Trusted contact' details are retrieved
Contacts are contacted

Class Analysis

To identify the potential classes and operations, we will perform the noun/verb analysis over our specification. Some possibilities were discarded for being redundant or beyond the scope of our system. We used the tables below to illustrate this analysis and describe each noun or verb. We used component keywords for components that are not in our primary interest.

Noun Verb Analysis

Candidate class(Nouns)	Use
Panic attacks	Sub-Class(of Confirmed panic attack)
User	Class
Health data	Class
Frequency and location	Attribute(Time and Location)
Subsystems	Component
Wearable tech	Class
Heart Rate	Sub-Class(of Wearable tech)
Breathing and Trembling	Sub-Class(of Wearable tech)
Servers	Component
Login information	Sub-Class(of User)
Panic attack history	Sub-Class(of Health data)
Database	Sub-Class(of User)
Alert	Attribute(of Panic attacks)
Device	Sub-Class(of Wearable tech)
Actions	Attribute(of Alert)
Time and Location	Attribute(of Database)
Patterns	Sub-Class(of Confirmed panic attacks)
Phone	Class
Spike	Attribute(of Alert)
Comprehensive Analysis	Attribute(Time and Location)
Confirmed panic attack	Class
Application	Sub-Class(of Phone)

Candidate Operations(Verbs)	Possible class Method
Tracks	Health data class
alerts	Outside scope
Perform	User class
Provide	User class
Assumptions made	Outside scope
Run the app	Phone class
Taken into account	Wearable tech class
Hosted	N/A
Store	N/A
Runs	Phone class
Monitors	Confirmed panic attack class
Detected	Too broad
Displayed	Wearable tech class
Asking	Wearable tech class
Confirm	Wearable tech class
Executed	Outside scope
Having a panic attack	Outside scope
Log the time	User class
Make available	User class
Happen	User class
To see	Outside scope
Download	N/A
Connected	Phone class

Using noun/verb analysis, CRC cards were created to model each potential class and sub-class. The Responsibilities tab describes how each card is going to implement its functionality in regards to the system. Then carefully choose how they will interact with the rest of the app in the collaboration tab.

User

Responsibilities - Performs user login, it requires user’s password, creates a database of date and locations of previous panic attacks.

Collaborators - Email, Phone, Preferences.

Health data

Responsibilities - Performs a close analysis and creates a log history of the user’s previous panic attacks.

Collaborators - Wearable tech, Sensors, Heart Rate.

Confirmed panic attacks

Responsibilities - Performs an alert as soon as a panic attack is confirmed by the sensors and the user.

Collaborators - Sensors, Wearable tech, Heart Rate, Patterns.

Phone

Responsibilities - Keeps track of the current time and location, inputs the existing contacts and the language set as default.

Collaborators - Preferences, Contacting Emergency Contact, Time and Location.

Wearable tech

Responsibilities - Used to perform and monitor heart rate, breathing and trembling, output messages and notifications, alerts and other features.

Collaborators -Panic attacks, Heart Rate, Breathing and Trembling, Device, Phone.

Heart Rate(sub-class of Wearable tech)

Responsibilities - Outputs current heart beat on the wearable device screen.

Collaborators - Wearable tech device.

Breathing and Trembling(sub-class of Wearable tech)

Responsibilities - Records current breathing and trembling and sends useful information to the wearable device in order to recognise a panic attack.

Collaborators - Wearable tech, Device.

Login information(sub-class of user)
Responsibilities - Performs user login, it requires user’s password and email.
Collaborators - Email, Password encryption, Database, User.

Database(sub-class of User)
Responsibilities - Stores multiple users location and date of previous panic attacks.
Collaborators - Server, User, Device.

Device(sub-class of Wearable tech)
Responsibilities - Used to perform and monitor heart rate, breathing and trembling.
Collaborators - Wearable tech,Phone.

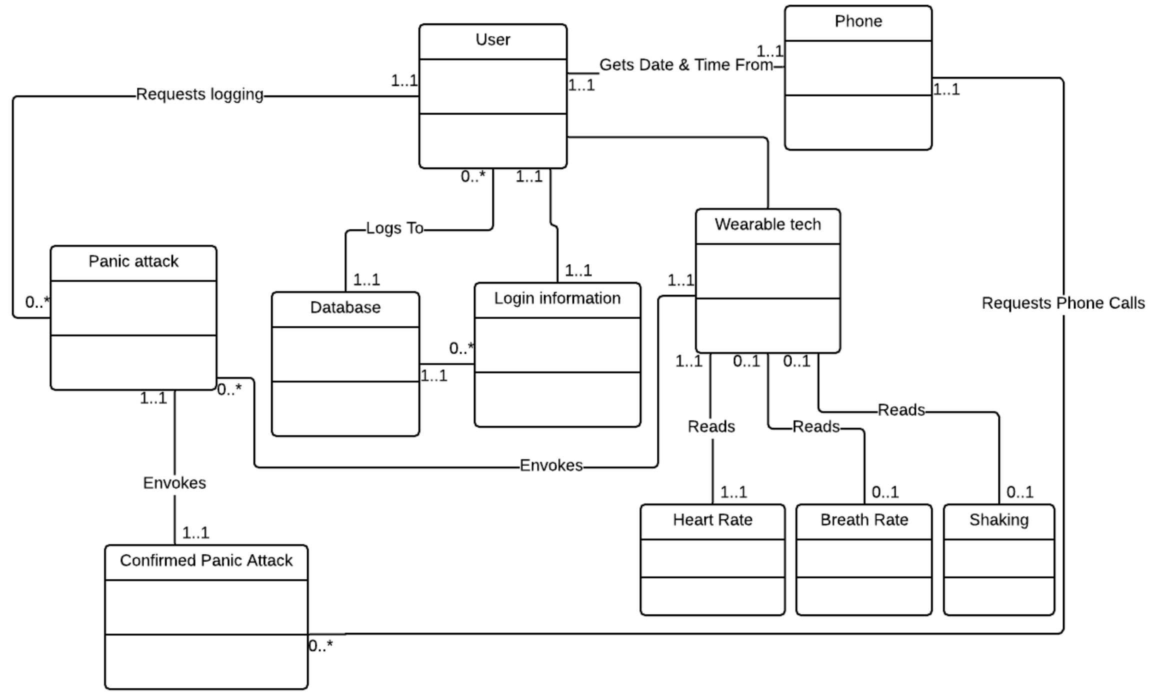
Patterns(sub-class of Confirmed panic attack)
Responsibilities - Track where and when their panic attacks are happening to see if there are any patters. Also important for the alert sub-class.
Collaborators - Alert, Panic attacks, Wearable tech, Device, Time and Location.

Panic attack(sub-class of Confirmed panic attack)
Responsibilities - Monitors user’s activity and using wearable’s patterns and sensors detects when the user has a panic attack.
Collaborators - User, Panic attacks, Wearable tech.

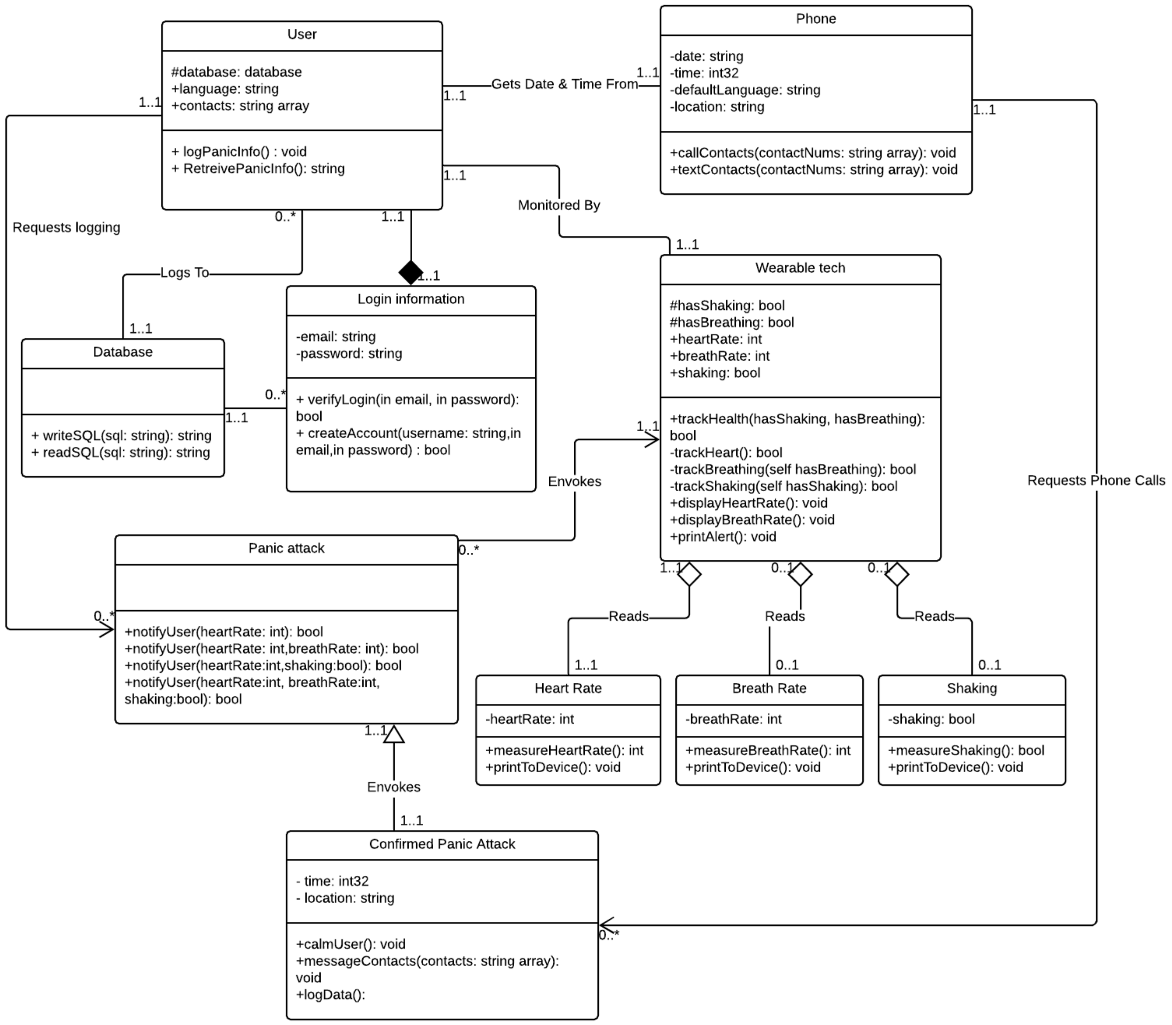
Panic attack history(sub-class of Health data)
Responsibilities - Keeps track of user’s previous panic attacks.
Collaborators - User, Panic attacks, Wearable tech, Confirmed panic attacks.

Application(sub-class of Phone)
Responsibilities - Tracks the user’s health data and alerts them of potential panic attacks.
Collaborators - User, Phone, Wearable tech, Confirmed panic attacks.

First-Cut Class Diagram



Class Diagram

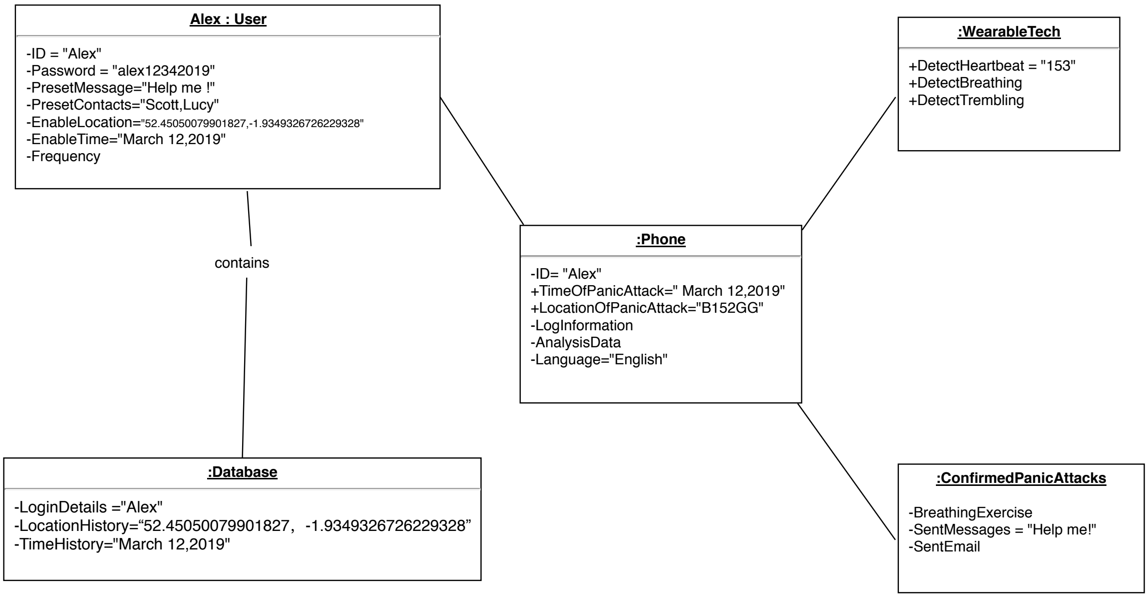


Object Diagram

The following, is an object diagram for the system, outlining the main objects used in the scenario where the user requires assistance as they are having a panic attack.

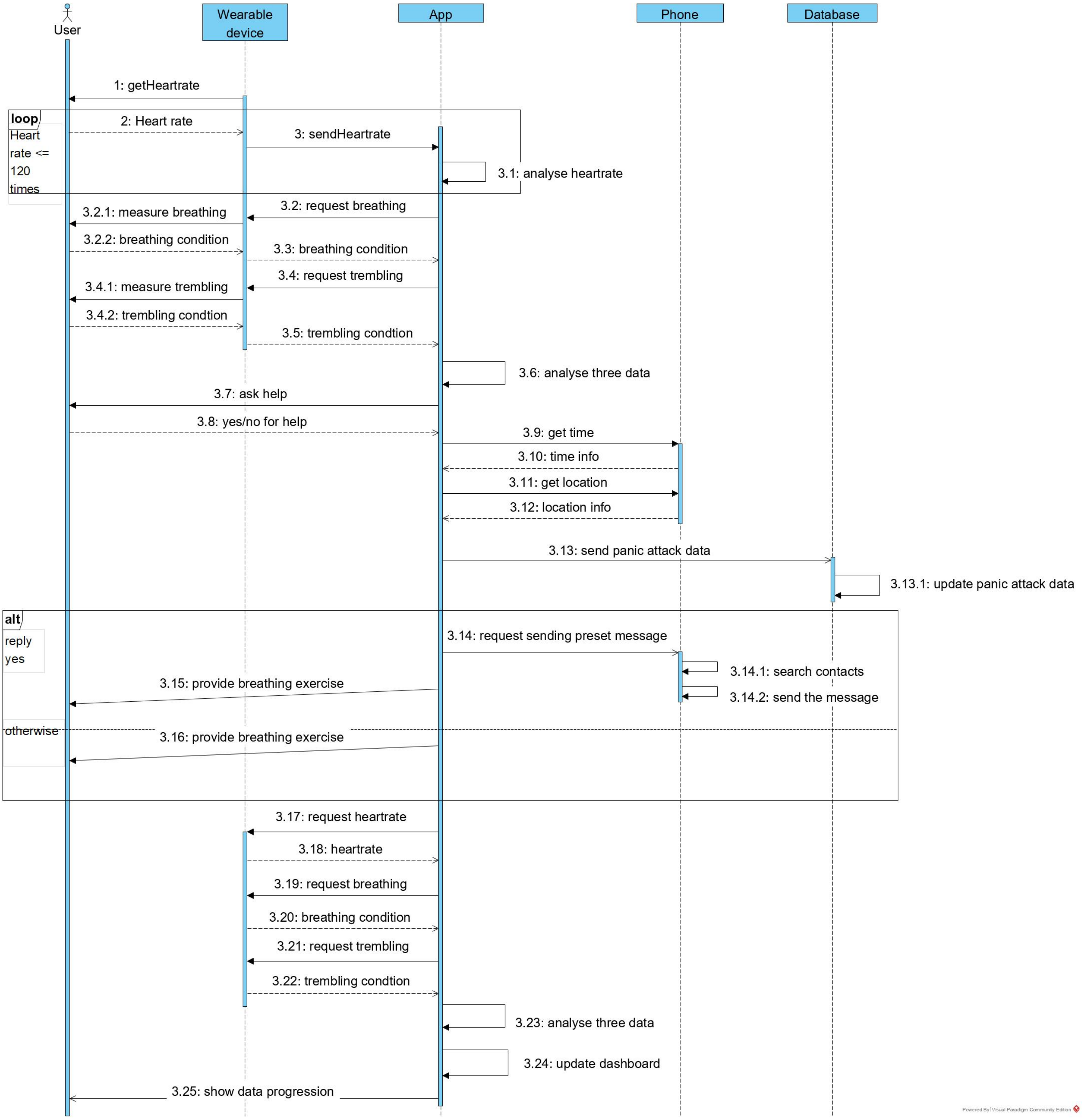
Scenario:

This object diagram describes a scenario where the user requires assistance as they are having a panic attack. The user, Alex, has used their password and username ID to log on to the app. Alex has set the app language to English and chose “Help me!” as an emergency message and has set Scott and Lucy as emergency contacts. The app is automatically running in the background of the mobile phone and is getting health data from the wearable device. The app detects that Alex’s heart rate is above 120 bpm and decides to check the user’s breath rate and if they’re shaking. After analysing all this information, the app suspects that Alex is having a panic attack. Therefore, the app displays a notification on the user’s phone asking if they require assistance. The user answers yes, so the app logs the user’s coordinates in Latitude and Longitude (52.45050079901827, -1.9349326726229328) and the Epoch time (1576071566) to the database. Meanwhile, it sends Alex’s chosen message to his emergency contacts while displaying breathing exercises for Alex


Alex : User
-ID = "Alex"
-Password = "alex12342019"
-PresetMessage="Help me !"
-PresetContacts="Scott,Lucy"
-EnableLocation="52.45050079901827,-1.9349326726229328"
-EnableTime="March 12,2019"
-Frequency
contains
:WearableTech
+DetectHeartbeat = "153"
+DetectBreathing
+DetectTrembling
:Phone
-ID= "Alex"
+TimeOfPanicAttack=" March 12,2019"
+LocationOfPanicAttack="B152GG"
-LogInformation
-AnalysisData
-Language="English"
:Database
-LoginDetails ="Alex"
-LocationHistory=“52.45050079901827, -1.9349326726229328”
-TimeHistory="March 12,2019"
:ConfirmedPanicAttacks
-BreathingExercise
-SentMessages = "Help me!"
-SentEmail

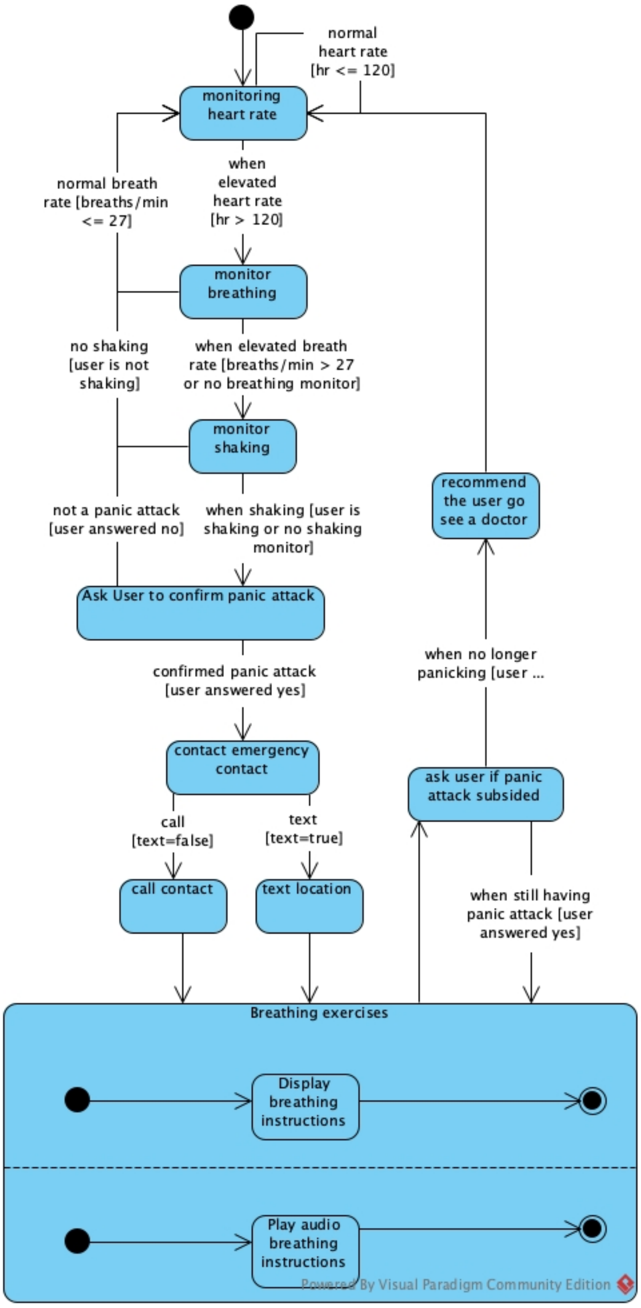
Sequence Diagram

This diagram describes the scenario when the user is having a panic attack. We assume that the app is running in the background of the user’s phone, the app monitors the user’s heart rate data until the user an abnormally high heart rate. The app then requests the user’s breathing and shaking data from the wearable device to confirm that the user is having a panic attack. To confirm that the user is having a panic attack, the app asks the user if they need help. The app provides breathing exercises to the user and can call or text the user’s trusted contacts. The app also sends the time and location data to the database to log when and where the user had a panic attack. The app, finally, allows the user to see their past panic attack history on the app’s dashboard.



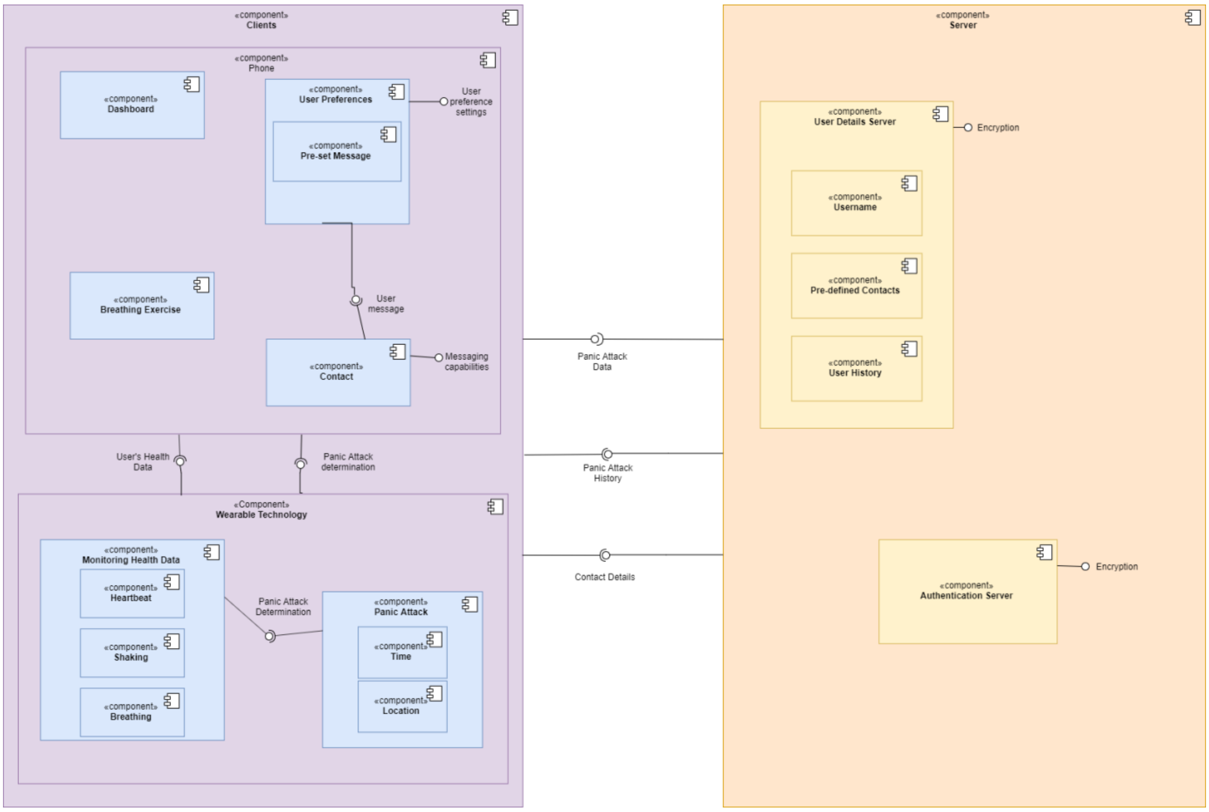
State Diagram

The following state diagram, details the way the app monitors the user's health data and how the app reacts in the case of a panic attack.

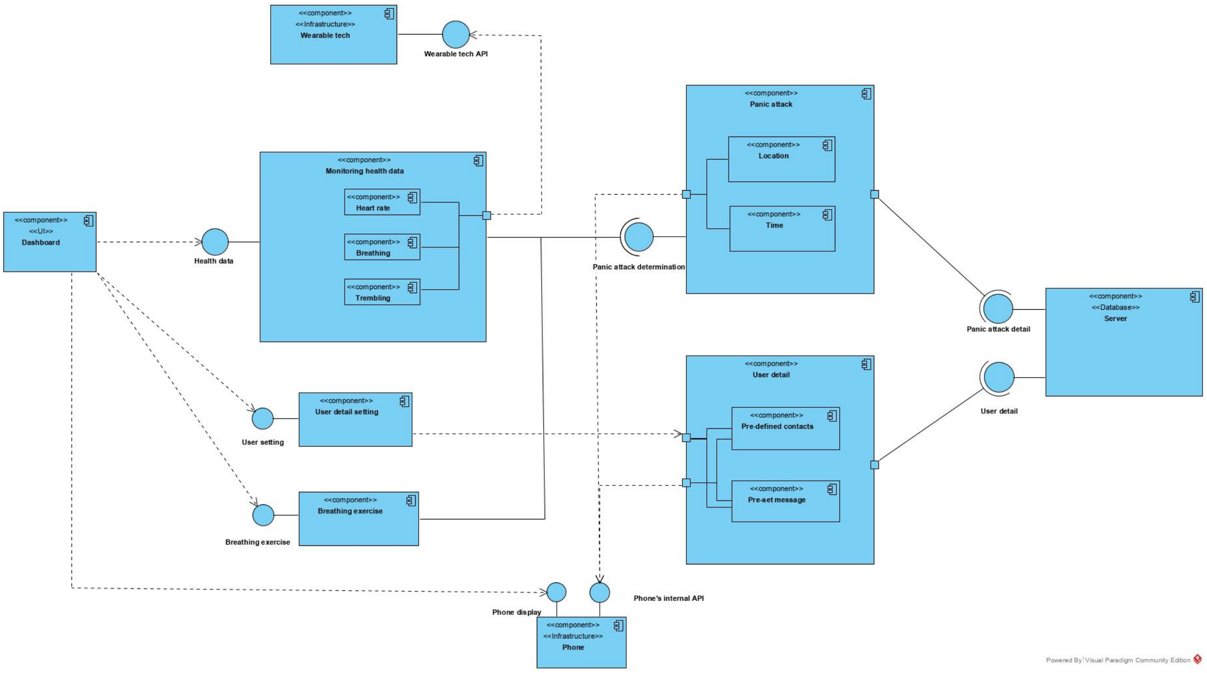

normal heart rate [hr <= 120]
monitoring heart rate
normal breath rate [breaths/min <= 27]
when elevated heart rate [hr > 120]
monitor breathing
no shaking [user is not shaking]
when elevated breath rate [breaths/min > 27 or no breathing monitor]
monitor shaking
recommend the user go see a doctor
not a panic attack [user answered no]
when shaking [user is shaking or no shaking monitor]
Ask User to confirm panic attack
when no longer panicking [user ...
confirmed panic attack [user answered yes]
contact emergency contact
ask user if panic attack subsided
call [text=false]
text [text=true]
call contact
text location
when still having panic attack [user answered yes]
Breathing exercises
Display breathing instructions
Play audio breathing instructions
Powered By Visual Paradigm Community Edition

Component Diagrams

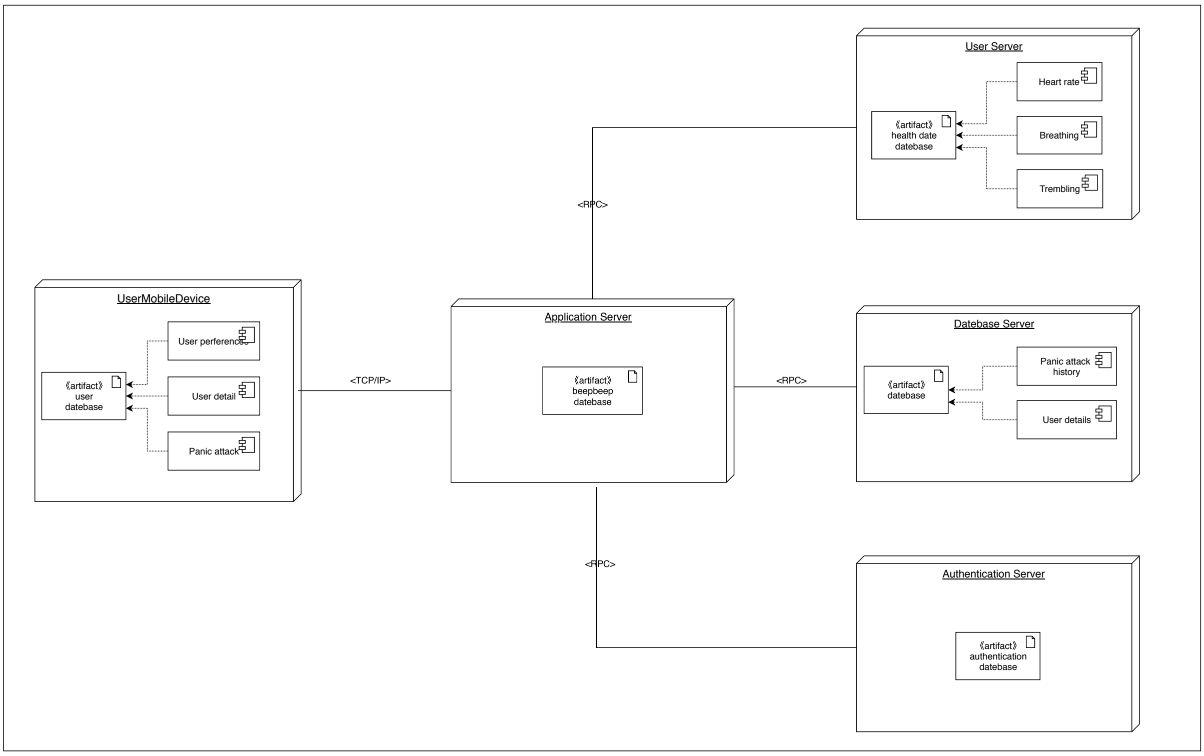
Client-Server architecture



Layered Architecture

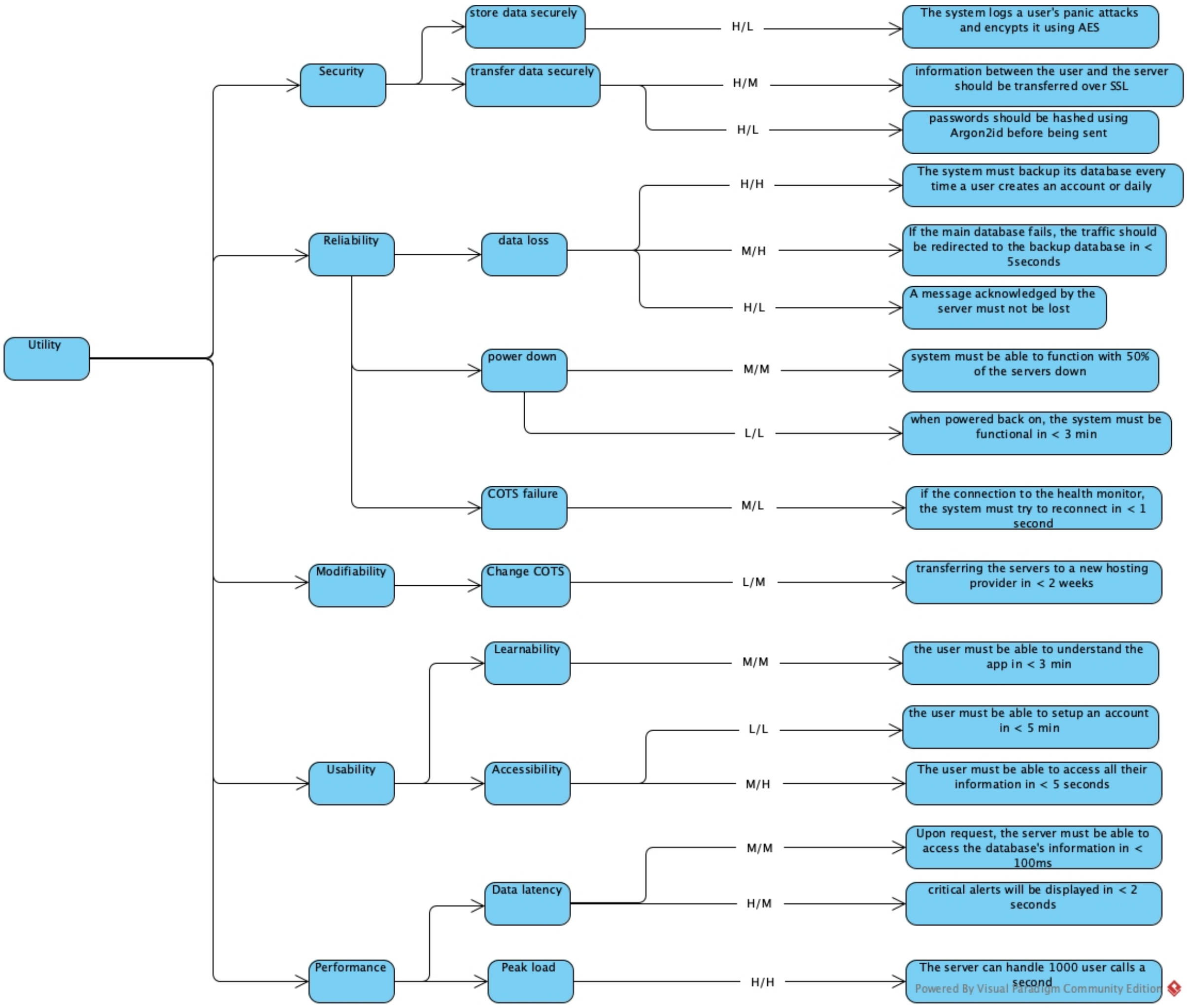


Deployment Diagram



Architectural Trade-off Analysis

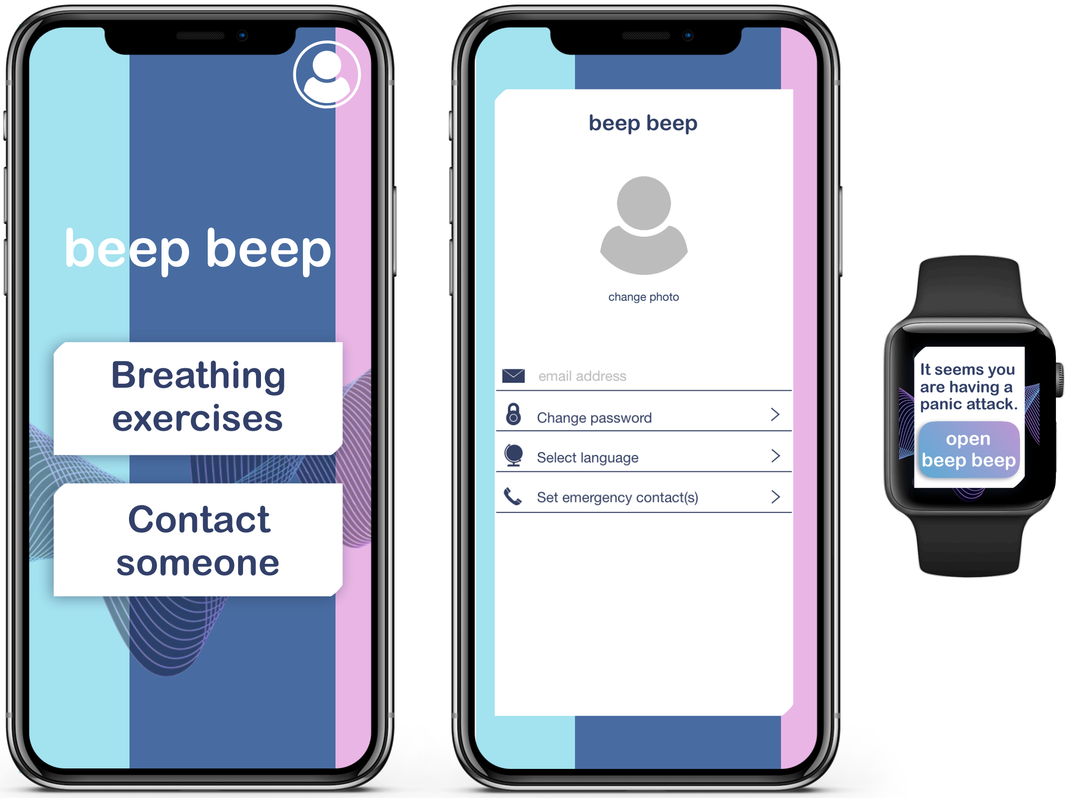
Utility Tree


Utility
Security
store data securely
H/L
The system logs a user's panic attacks and encypts it using AES
transfer data securely
H/M
information between the user and the server should be transferred over SSL
H/L
passwords should be hashed using Argon2id before being sent
Reliability
data loss
H/H
The system must backup its database every time a user creates an account or daily
M/H
If the main database fails, the traffic should be redirected to the backup database in < 5seconds
H/L
A message acknowledged by the server must not be lost
power down
M/M
system must be able to function with 50% of the servers down
L/L
when powered back on, the system must be functional in < 3 min
COTS failure
M/L
if the connection to the health monitor, the system must try to reconnect in < 1 second
Modifiability
Change COTS
L/M
transferring the servers to a new hosting provider in < 2 weeks
Usability
Learnability
M/M
the user must be able to understand the app in < 3 min
Accessibility
L/L
the user must be able to setup an account in < 5 min
M/H
The user must be able to access all their information in < 5 seconds
Performance
Data latency
M/M
Upon request, the server must be able to access the database's information in < 100ms
H/M
critical alerts will be displayed in < 2 seconds
Peak load
H/H
The server can handle 1000 user calls a second
Powered By Visual Paradigm Community Edition

With this utility tree, we are going to compare the client-server architecture with the layer-based architecture. First, let’s look at Performance vs Security, we choose to look at “Upon request, the server must be able to access the database’s information in < 100ms”. In this case, the client server is probably a better choice than the layer-based architecture. However, if we consider “ information between the user and the server should be transferred over SSL”, then it’s clear that the layer based architecture is a better fit for this because of the simplicity in the connections, every layer only interacts with at most two others, making it easier to check that all data is SSL encrypted”. In our system, security is the most important factor above performance as we are handling sensitive user data. Therefore, in this case, the layer-based architecture is the better choice.
Let’s now consider reliability and more specifically “if the main database fails, traffic should be redirected to the backup database in < 5 seconds”. Once again, the architecture of choice for this scenario, seems to be the layer-based architecture because of the simplicity of the system: to redirect traffic to the backup database, the server needs only to redirect its traffic to a different address. However, in a client server architecture, this change would be internal making it potentially more challenging.

Finally, let's look at modifiability with "transferring the servers to a new hosting provider in < 2 weeks". With a client-server approach, we see that it would be harder to transfer the whole server. Whereas, through a layer-based architecture, it would be possible to transfer each layer one-by-one relatively easily without really affecting reliability.

User Interface



It seems you are having a panic attack.