Health Monitoring
Group #1

Report #3

Team Members:

- **Ai, Ziqi**
  - **Email: ziqi.ai@rutgers.edu**
  - **Background: Experienced with C++, Familiar with Python**
- **Guo, Kai**
  - **Email: kai.guo@rutgers.edu**
  - **Background: Experienced with C, Familiar with Java**
- **Pielacki, Kevin**
  - **Email: kpielack@scarletmail.rutgers.edu**
  - **Background: Experienced with Python, C++, and MySQL**
- **Shen, Hongyuan**
  - **Email: hongyuan.shen@rutgers.edu**
  - **Background: Experienced with C, Familiar with Java**
- **Xing, Xiang**
  - **Email: xx52@scarletmail.rutgers.edu**
  - **Background: Experienced with C++ and JDE, Familiar with Java and MySQL**
- **Yang, Guanjiang**
  - **Email: gy101@rutgers.edu**
  - **Background: Experienced with C and Matlab**

Project Repository: https://github.com/kpielacki/ece-567

Table of Content:

Summary of Changes

1. Customer Statement of Requirements
2. Glossary of Terms
3. System Requirements
4. Functional Requirements Specification
    - Refinement of all the items and descriptions in the part of the Functional Requirements Specification
    - Removing the unused part of User Case 5 Sharing Health Conditions in the first project and modification of the later part of actor and actor's goals
5. Effort Estimation
6. Domain Analysis
7. Interaction Diagrams
8. Class Diagram and Interface Specification
    - Removing, refinement of all interaction between separate classes that may cause ambiguous understanding
    - Changing and modifying the inconsistent field names between classes to make sure them could match each other
    - Removing and refinement of all redundancy in variables between different classes, eliminating data inconsistency in the software implementation of whole system
    - For better connection between the diagrams and modules, modify the whole structure of those parts
    - Removing the original UC5 related contents in previous reports
    - Refinement of the data types, variables and methods included in the system
    - Adding related descriptions and labels to these associating links
    - For the description for the traceability matrix, adding more information based on the real-used application activities to the Domain Concept
    - Adding all the location and displaying map function related items, contents with description and diagrams
9. System Architecture and System Design
    - included the reason of choosing this architecture
    - identify the usage of local database and cloud database
    - Map the subsystem to the hardware
    - explanation about the classes that are connected with each model type
10. Algorithms and Data Structures
    - Add more description about health advice section, this can make this part become understanding easily
    - Add more information about machine learning for rating

- Add corresponding classes in the class diagram for some concepts from the domain model

11. User Interface Design and Implementation
    - Delete the section “Getting Advice”
    - Add a new section “Get My Detailed Location”
    - Add a subset “Body Mass Index” in section “Health Rating”
12. Design of Tests
    - Echo Reply From Server
    - Login Using Mobile Device
    - Upload Data with Mobile Device to Remote Database
13. History of Work, Current Status, and Future Work
    - Add the history work which is completed step by step
14. OCL Contract Specification

1. Customer Statement and Requirements (CSR)

Problem Statement

It's general knowledge that a person's chosen living environment will have some impact on a their health. Occasionally a study will appear claiming that a person exposed to some condition is more or less likely to suffer from some physical ailment. When these claims are definitively proven drastic steps are taken to improve the state of being. Some intuitive examples of this would be exposure to pollution or hazardous build materials. Cities and homes are now designed in such a way to avoid health hazards due to these findings however these are now obvious hazards. Perhaps a more worrying concern is what has not been found or proven to be hazardous yet for conditions that are not as obvious.

So it's possible to come up with a whole new set of questions but collecting data on a person's daily living situation and fitting their habits to a generic category of well known issues does not really give insight on a better living style. Instead an effort should be made to try to find new ways of proving what may be abnormal in a person's health. For example it may be interesting to know the rate cancer appears in people that live in a close proximity to a power plant against the normal rate but there is no reason to limit this to power plants. Analysis can be done on long term exposure to high pollution. Living in areas with high or low atmospheric pressure can be found to lead to irregular heartbeat. Not only does this allow for long term health recording but it also informs the user how their environment could potentially harm them.

Thanks to the development of technology, it's now possible to use a mount of sensors and applications in smartphones to monitor the data of our health. For example, the PPG sensor monitors a person's heart rate. This can be considered one of the most basic index of health. The temperature transducer can be used to detect skin temperature. Another emerging trend is to use a phone's inbuilt gyroscope sensor to track how many steps the user has taken today. Once all this data is logged in a single database a number of methods can be applied to manipulate data in such a way that an simple mobile application will be able to tell a user more about their health on demand. To keep up with a user's standards and emerging market trends there's a dire need to have smarter feedback that moves most of the work away from a user. Many mobile users are not willing to dive deep into a topic and instead want a simplified summary of what they're looking for.

In general, feedback regarding our health is given through three major source. The first and most reliable option would be a visit to the doctor's office. Since a doctor's office is properly equipped to perform a patient checkout this is without a doubt the preferred method but these visits are infrequent. It's also impossible for your doctor to know about your lifestyle habits from a simple checkup. The next major source would be a word of mouth advice. Many people seem to want to cross check what their doctor might of told them with a friend or family member. This is a fair response however everyone will need to validate these responses moving to the final source of webpage documents. There's a few trusted sites that can help validate this information however it's alway left to the user's discretion and not always easy to find. The new proposed method is to instead collect basic vitality signs and health indicators from a user's mobile device and return a simple feedback on how well their health is.The idea is to not only collect information on how healthy the user is but also information that would help determine health risks with the user. The value of this information includes a large user base for ad revenue, keeping a consistent record of vitality factors for research and development, and a very useful indicator for health insurance companies to possibly provide discounts to lower risk users.This indicates that a domain of possibilities may be overlooked from a simple doctor's visit. In the future it may even be possible to tell more about a person's health than a visit to a doctor provided the platform evolves with emerging technology.

The first major requirement is to establish a way to collect user health information from their mobile device. The application will need to be simple to use so information can be obtained on people of all ages which consequently extends our user base. It should also keep user information private. It's very easy for someone to feel hesitant that they're health is being monitored so it's important that they feel the application can be trusted. No one will want to download an application that does not give them something in return so it's important that they get active and fun feedback about their health. It's very import that the average person feels like the application is helping them make better health decisions and not just collecting information on them. It's also very important to get this done as soon as possible. The earlier data is collected the easier it will be to strike a balance between the software functionality and time to develop. Overall it's important to find a use for all the information that's being collected on the user and in return be able to give better insight on their health.

The next step is to to figure out the data can be related to user's health condition. External data sources can be used to map back to health data. For example a lookup of locations nearby that can negatively impact their health or the pollution level of the region might reveal interesting facts about the user. By knowing how often a user is exposed to these conditions it's possible yield more logical conclusions. It will be very helpful to notify a user that they are say for example spending 60% of their time near a long term health hazard. It would also be very helpful for researchers to possibly verify or debunk some of the general beliefs about these so called hazards. Overtime this data can be anonymously shared for medical research to validate or come up with new conclusions. It's also important to come up with simple and generalized conclusions based on the data being collected alone. By collecting some initial information like the user's age and gender it's easier to categorize the user base. Some other initial user information that may be helpful is if the user has a pre-existing health condition however it may be possible to cross check this information from a medical system in the future as the application evolves. Over time it should be possible to find what may be normal conditions for a person's health for that particular group.

Finally, it's important to come to a conclusion regarding the user's health through some generalized health index and relaying that information in a simplified form back to the user. The main challenge of this part is finding a index that is easy to measure but also a reliable judge of a user's health. One of the simplest measures that can be a good indicator of user health is possibly heart rate. It may be possible as more data is collected to figure out a trend between a user with a healthy heart versus a user with an unhealthy heart. This builds off the first major requirement in the sense that the earlier data is collected the better. Basically a smarter decision can be made by getting a head start on finding health irregularities from these measurements. By building up reliable algorithms predictions will be made, tested and verified. Only then can a reliable conclusion be made.

Contemporary, some existing personal health monitoring application collect user data via a smartwatch or other external devices that are not available to the majority. For most cases, those apps cannot work without a Bluetooth connection to external devices. Since the smartphone consists of many different kinds of sensors with the networking functionality, it could be used in monitoring user's health conditions. It will definitely be of great convenience to the user. Additionally, it will be quite easy to link the internet using smartphone rather than wearable devices for collecting data. Especially for uploading and analyzing those data. This makes the data a lot more shareable and casts a larger net on the possible number of users the application covers.

In order to generate more conclusive results without intuitive data, connecting different types of data together is a great approach. For example, location information from mapping service providers will play an important role in this project. By mapping the location info to different databases, such as income, pollution and altitude, additional data will be found and analyzed. So overall the data we're collecting needs to be able to easily link back to pre-existing information possibly collected by another company.

Some additional social network functions can be a good way for applying the sense of competition into this app. For example, users can share their exercising status on facebook. An intuitive UI design may courage people to download the same application and achieving a great health condition with friends can be a strong motivator for customers to improve their own health. What's more, the system will apply different strategies to analysis data collected. For instance, the application could count the time which user marks as "going to the gym" for “strenuous exercise” and other activity time for “mild exercise” such as walking or doing housework. Then the total activity condition will be generated based on those two main kind of activity status.

After adopting machine learning method, the application can give users some advice. At this time, using some external links, actual examples and pictures can help users understand better. Then users can easily combine application’s advice and information on the Internet to make a plan which is suitable to the users themselves. And this way can help users relieve pressure, they do not need to follow the application.

2.Glossary of Terms

Accelerometer: An accelerometer is a device that measures proper acceleration. In the system it can be used in detecting the vibration and user’s abrupt actions. Function: measure acceleration, tilt and vibration

PPG sensor: Measure the bio-potential generated by electrical signals that control the expansion and contraction of heart chambers.

Gyroscope: Used to detect orientation. Typically used to detect motion.

Temperature sensor：Detect the temperature change, and pass into data.

Light sensor: A light sensor is a device that detects light or other electromagnetic energy. In this system it detect light conditions to show the sleeping status.

User Interface (UI):The goal of user interface design is to produce a user interface which makes it easy (self-explanatory), efficient, and enjoyable (user-friendly) to operate a machine in the way which produces the desired result.

Compass:An instrument containing a magnetized pointer that shows the direction of magnetic north and bearings from it.

Displacement sensor: An instrument to measure travel range between where an object is and a reference position.

Atmospheric pressure sensor: A a scientific instrument used in meteorology to measure atmospheric pressure. Pressure tendency can forecast short term changes in the weather.

Tile Server: A program running on a remote machine that returns mapped location images for a set of coordinates.

Android (operating system): Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets.

GIS: Geographic Information System, a system designed to capture, store, manipulate, analyze, manage, and present spatial or geographic data.

User Base: Established group of users for a particular computer program.

3.System Requirements

Functional Requirements

Identifier	Priority Weight	Requirement
Req-1	5	The system will insert sensor measurements to a remote database.
Req-2	1	Hold sensor data on mobile phone when no network connection if present.
Req-3	1	Upload sensor data upon connection reestablishment.
Req-4	2	History log will be presented when there is a requirement
Req-5	2	Send periodic user health feedback based on the environment.
Req-6	3	Send user general health feedback based on heart rate.
Req-7	4	Send user energy consumption feedback based on quantity of motion.
Req-8	2	System should give out advice about user’s daily activity.
Req-9	1	Allow user to give out feedbacks to developer.

Req-10	3	Use GPS sensors to get the data about location, velocity and moving distance.
Req-11	3	Use map information and location data from GPS to analyze which places users stay in.
Req-12	2	Provide pollution level based on map location.
Req-13	3	Use PGG sensors to get user's current heart rate.
Req-14	3	Use temperature transducer to get the skin temperature.
Req-15	2	The data analysis should be processed in the cloud system.
Req-16	3	Analyze the places users stay in and the data from acceleration sensor and speculate the action of users.
Req-17	3	Use gyroscope to get data of level, orientation, velocity and accelerated velocity.
Req-18	4	Using vibration sensor to detect user's spontaneous physical actions during sleep.
Req-19	3	Use heart rate to estimate the mental health condition of the user.
Req-20	2	The application should require minimum number of user's inputs.
Req-21	2	The system should be easy to use. For example, feedback should be got within 3 clicks.
Req-22	1	The administrator should be able to access user's data and make changes.

Req-23	2	The system should be robust. Malfunction of several sensors will not break down the whole system.
Req-24	4	All user data should be encrypted for privacy security.
Req-25	3	The application should keep working in the background.
Req-26	1	Every icon must be clickable, otherwise it should become a grey one.
Req-27	2	Allow users share their location.
Req-28	1	Landscape mode should be available.

Non-Functional Requirements

Identifier	Priority Weight	Requirement
Req-1	3	Allow user to select which sensors to report on to allow additional privacy and power efficiency.
Req-2	2	Option to only upload data when on WiFi to avoid extra mobile charges.
Req-3	2	Allow user to share health condition with friends.
Req-4	2	The system should require little maintenance.
Req-5	2	The system should provide with a straightforward UI showing the conditions of user’s parts of body.
Req-6	1	The feature of UI can change based on different seasons and festivals.
Req-7	2	The system should provide link about professional nouns.

Req-8	1	Some tips related to weather condition are posted to users.
Req-9	4	Users’ data should be saved in their cell phone when transmission is termination.
Req-10	1	Apply larger icons and limit the number of items in each menu for better visual experience.

4.Functional Requirements Specification

Stakeholders

- Customer
    - Using the application and web apps to using the system to get advice
- Developers and Maintenance Team
    - Developing the application and maintain the whole system
    - Fix bugs and update the system modules
- Business Companies
    - Getting the health information in some aspects of a person(User)

Actors and Goals

- User
    - Primary Initiating Actor
    - Attempting to monitor and improve health condition
- Administer
    - Supporting Actor for running the system
    - Keeps track of user accounts and system maintenance
- Database: Cloud
    - Offstage Actor
    - Keeps record of user data for analysis and summary results
- Sub-database: Smart-Phone
    - Offstage Actor
    - Keeps temporary record of user data until ready to upload to cloud
- System
    - Offstage Actor
    - Inner working of web server, database, and analyzer

- Health Insurance Agent
  - Initiating Actor
  - Attempting to better categorize users

Actor	Actor’s Goal	Use Case Name
User	Begin tracking their personal health.	Registration (UC-1)
User	View current account.	Login (UC-1)
Administrator	Track and modify user accounts.	AdminPortal (UC-1)
Database	Log sensor data to remote server.	DataCollection (UC-2, UC-3)
Sub-Database	Temporarily hold sensor data until ready to upload with steady network connection.	TempData (UC-3)
User	View logged health status and history.	HealthDashboard (UC-2)
User	View a quick summary of overall health.	HealthScore (UC-3)
GPS Data	Provide list of nearby environmental hazards.	GISAnalysis (UC-3)
Heart Rate Data	Provide prediction of better or declining health.	HeartRateAnalysis (UC-3)
User	Get suggestions on how to improve health.	HealthFeedback (UC-4)
Insurance Agent	View a new applicant’s health condition.	UserPublicSummary (UC-3, UC-6)

5. Effort Estimation using Use Case Points

a. Use Case Points

Projects with many complicated requirements take more effort to design and implement than projects with few simple requirements. In addition, the effort depends not only on inherent

difficulty or complexity of the problem, but also on what tools the developers employ and how skilled the developers are. The formula for calculating UCP is composed of three variables:
1. Unadjusted Use Case Points (UUCP), which measures the complexity of the functional requirements.
2. The Technical Complexity Factor (TCF), which measures the complexity of the nonfunctional requirements
3. The Environment Complexity Factor (ECF), which assesses the development team's experience and their development environment.
UCP = UUCP × TCF × ECF

Actor Type	Description of how to recognize the actor type	Weight
simple	The actor is another system which interacts with our system through a defined application programming interface (API)	1
average	The actor is a person interacting through a text-based user interface.	2
complex	The actor is a person interacting via a graphical user interface.	3

i. Functional Requirements
Unadjusted Use Case Points (UUCPs) are computed as a sum of these two components:
1. The Unadjusted Actor Weight (UAW), based on the combined complexity of all the actors in all the use cases.
2. The Unadjusted Use Case Weight (UUCW), based on the total number of activities (or steps) contained in all the use case scenarios.

Actor	Description of actor	Complexity	Weight
database	The actor another system which interacts with our system through a defined API.	simple	1
user	User is interacting with the system via a graphical user interface (when managing user on the central computer).	complex	3

UAW=3 × Complex +1 × Simple= 3 × 1 +1 × 1 =4.

dimension	Description of the dimension
complexity	The numbers of icons and objects that interact with the user

fréquence	How often the user view the page.

weight	complex(>10 objects)	simple(<10 objects)
frequent (<1 times one day)	15 Type: A	10 Type: B
seldom(>1 times one day)	10 Type: B	5 Type: C

The UUCW is calculated by tallying the use cases in each category, multiplying each count by its specific weighting factor, and then adding the products.

User case	Description	Type
UC-1	Registration and Sign In:User A creates an account through the login in page. User A provides his/her personal information on the login page. After server confirmation, User A can login with email and password.	B
UC-2	User profile:User can change and view their his basic information in this page.	C
UC-3	Health rating:User can get ratings based on the data set selected.User can also get an overall score of their health condition.	C
UC-4	Hazard nearby: In this page, users can locate the nearby hazard points in the map based on Google map, additionally the hazard points including: fire, nuclear pollution and air pollution.	B
UC-5	Get detailed location: Locate themselves and get the accurate longitude and altitude of the User.	A

UUCW=2×A+2×B+2×C=2×5+2×10+2×15=60

UUCP=UAW+UUCW=64

Ii Nonfunctional requirements

Identifier	Priority Weight	Requirement	Complexity	Factor
Req-1	3	Allow user to select which sensors to report on to allow additional privacy and power efficiency.	2	6
Req-2	2	Option to only upload data when on WiFi to avoid extra mobile charges.	1	

Req-3	2	Allow user to share health condition with friends.	2	4
Req-4	2	The system should require little maintenance.	1	2
Req-5	2	The system should provide with a straightforward UI showing the conditions of user’s parts of body.	1	2
Req-6	1	The feature of UI can change based on different seasons and festivals.	1	1
Req-7	2	The system should provide link about professional nouns.	2	4
Req-8	1	Some tips related to weather condition are posted to users.	1	1
Req-9	4	Users’ data should be saved in their cell phone when transmission is termination.	1	4
Req-10	1	Apply larger icons and limit the number of items in each menu for better visual experience.	1	1

TFT=27
Constant-1 (C1)=0.6, Constant-2 (C2)=0.01.
TFC=0.87
As a result, this will cause in a reduction of the UCP by 13%.

Iii Environmental Factors

Identifier	Description	weight	impact	factor
1	Beginner familiarity with the UML- based development	1	1	1
2	Some familiarity with application problem	1	2	2
3	Some knowledge of object-oriented approach	1	2	2
4	Beginner lead analyst	1	2	2
5	Highly motivated, but some team members occasionally slacking	2	3	6

6	Stable requirements expected	2	3	6

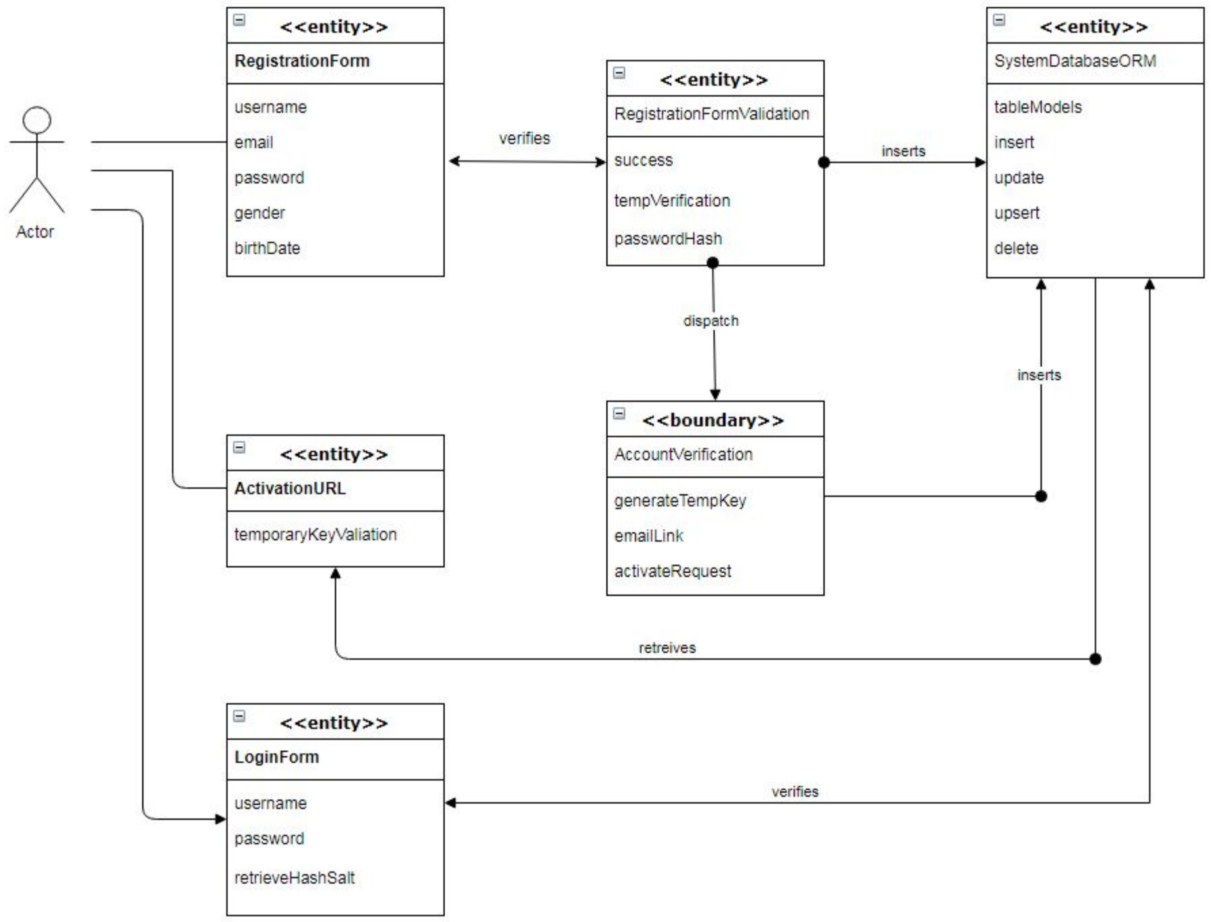
EFT=19
Constant-1 (C1)=1.4, Constant-2 (C2)=-0.03
ECF=1.4-0.57=0.83

UCP=UUCP×TCF×ECF=46.2

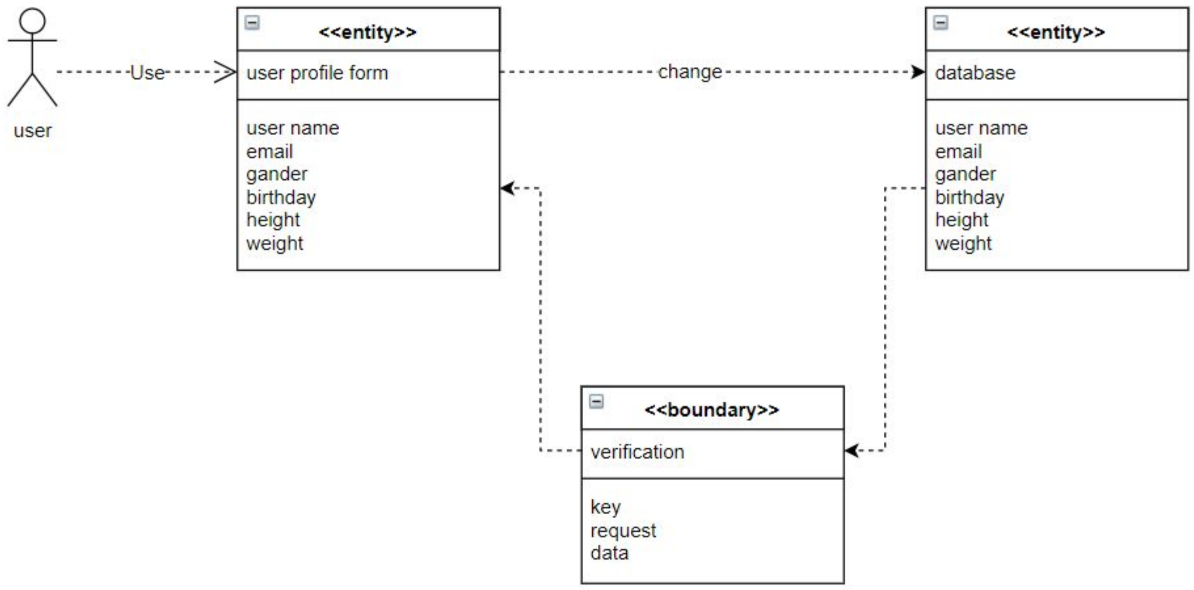
Use case points (UCP) are a measure of software size. When we set Productivity Factor PF=28 peruse case point, we can get our project Duration=UCP×PF, where PF=28 hours per use case point.
Duration=UCP×PF=46.2×28=1294 hours

6. Domain model

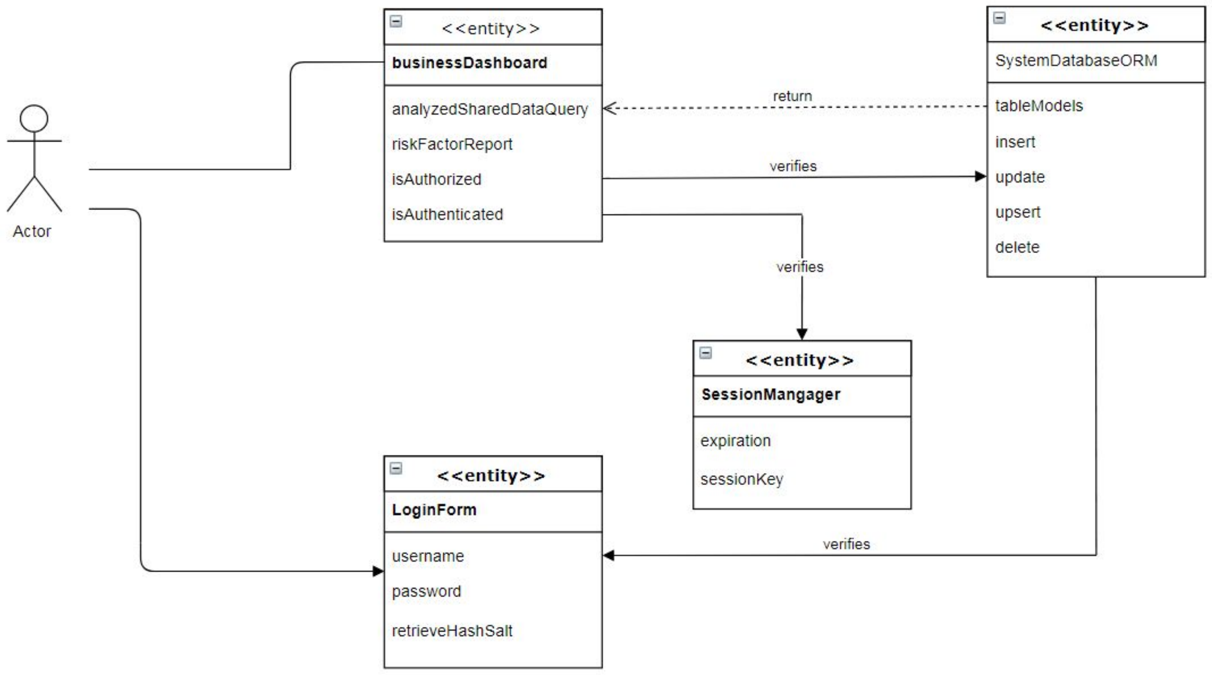
UC-1 Registration and Sign in



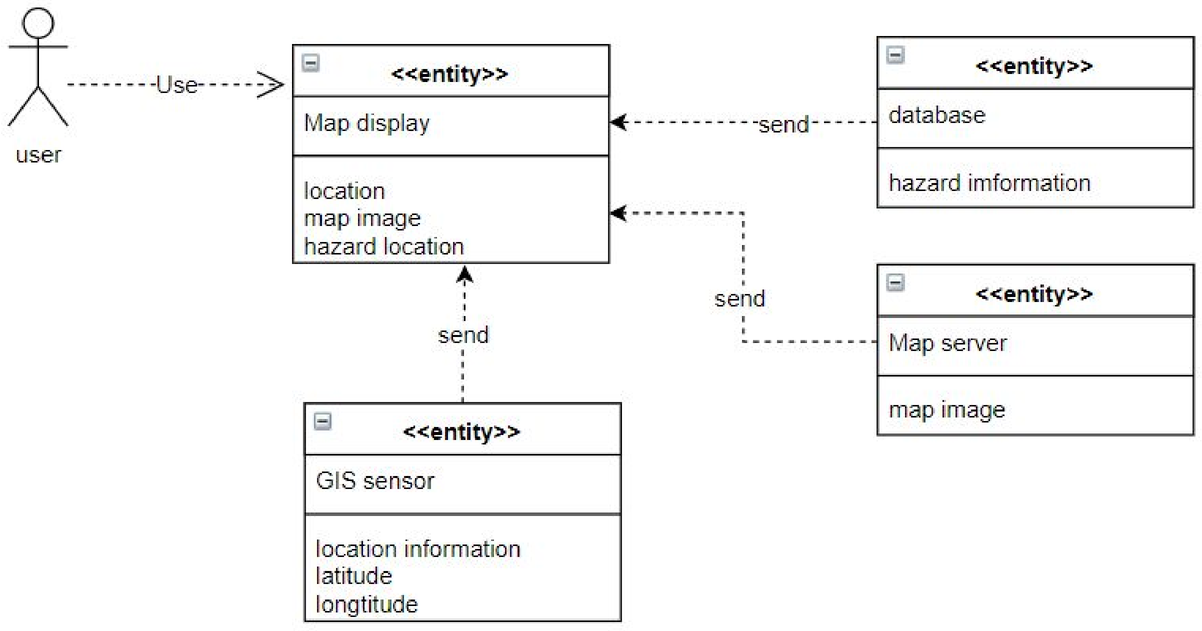
UC-2 User profile:
User can change and view their his basic information in this page.



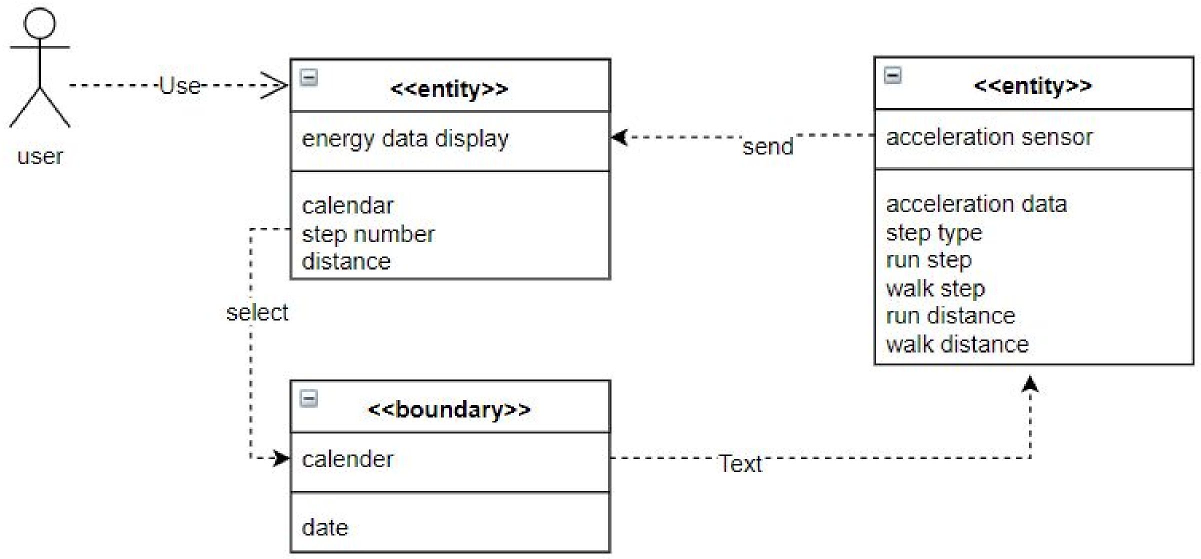
UC-3 Health rating::User can get ratings based on the data set selected.User can also get an overall score of their health condition.



UC-4 Hazard nearby:
In this case, users can locate the nearby hazard points in the map based on Google map, additionally the hazard points including: fire, nuclear pollution and air pollution.



UC-6 Calories and exercise



Concept Definitions
(D=Doing; K=Knowing; N=Neither)

Responsibility	Type	Concept
User creates account and inputs all of required personal information on the login page	D	User Interface
User can login the app using his/her account and password	K	User Interface
User can retrieve their password	K	User Interface
Transmits the data from web server to user's smartphone	D	System
Displays the data to user in the form of timetable or chart	D	System
Analyzes user's data and give useful advice to the user	D	System
Stores the data from user's	D	Database

smartphone		
User can download the previous data from the database	D	Database
Manages all users' health information	D	Manager Interface
Allows the App to use the location information and search around location information from the map server	D	Internet Connection
Links to the database	D	Internet Connection
Senses user's changes all the time.	D	Sensor
All part of this App should connect with each part.	D	Controller

Associative Definitions

Registration and Sign in And databases	User passes information through an interface like App. The information ranges from account info to making a reservation	Pass information to local database and upload to cloud database
User profile database	Database and Sub-database pass data to User Interface	Database Updated and pass Data to UI
Health rating database	Database pass user's data to analysis and estimation system and system pass the conclusion back to database	Data analysis and storage
Hazard nearby and Database	Database pass user's conclusion to system and system show the result to user	Conveys Information

Attribute Definitions

Concept	Attributes	Definition

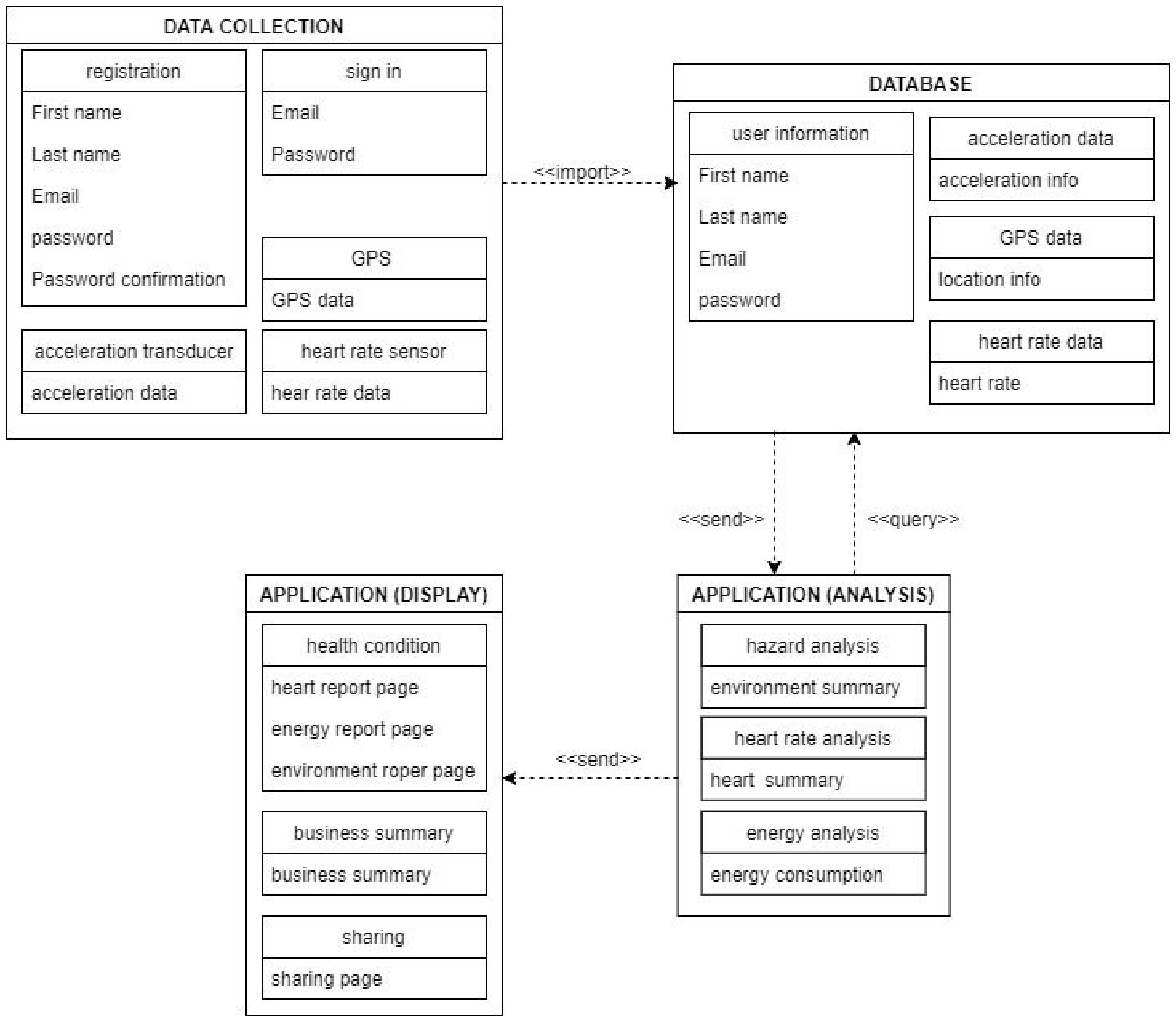
User Interface	Registration form	User fills in the form and passes the user’s info to the database
	Loginform	User fills in the form and submits for verification
	Analyze dashboard	Page to show the analyzed data such as energy consumption and average heart rate of the user
	Heart rate trend	A page shows the heart rate report based on the data analysis
	Environmental summary	A page shown the nearby environmental report based on the recent pollution conditions
	Suggested sources	A page shown the advice given by the health summary.
Database	Registration and validation	Check out the database and add user’s info into the database. Create validation status to be confirmed
	Analyzing and estimation processes	Store the conclusion and link them to the corresponding user’s information
	Updating and edit	Synchronize user’s local database to System cloud database. Allow admin to edit user’s info and data
	Userinfo	Contains the user information such as email, gender, date of birth and personal preferences.
	HistroyList	Contain user’s health history, both in local database and system database
Analyzing System	Step model	A series of math models applied to estimate the number of step through the collected data.
	Intersection analyze	Combine two or more different kinds of data together to analyze health condition.
Hazard nearby	Hazard information	A authorized information of the public hazard and the corresponding location on

		map
	Locarion	User can choose to locate themselves and show the latitude and longitude

		Domain Concepts			
		User Interface	Application	Map	Database
Use Cases	UC-1 (Registration and Sign In)	X	X		X
	UC-2 (calories and exercise)	X		X	X
	UC-3 (Hazard nearby)	X	X	X	X
	UC-4 (Health rating)	X	X	X	
	UC-5 (Locating)		X		X
	UC-6 (User profile)	X	X		X

7.Interaction Diagrams

Structural patterns adapter:
Adapter means the adapter pattern is a software design pattern (also known as Wrapper, an alternative naming shared with the Decorator pattern) that allows the interface of an existing class to be used as another interface.



The System Architecture mainly composed of 4 parts.
The first part is Data collection. The subsystem of the part are:
(1) Registration
(2) Sign In
(3) GPS
(4) Acceleration transducer
For this part, we want to make a relationship between each part, we just need to use the adapter. And the adapter can translate some unique part to unique, we translate the User ID, name and other identified information. So it can be efficient.

Concurrency patterns reactor pattern

The reactor design pattern is an event handling pattern for handling service requests delivered concurrently to a service handler by one or more inputs. The service handler then demultiplexer the incoming requests and dispatches them synchronously to the associated request handlers. Architectural Styles: The architectural style of this health monitor system include principle from a variety of styles. Our system incorporate the Client-server model in business summary part. Business company can access the data through the internet via a website then dashboard. Service-oriented architecture also plays an important part in our app, as the system provides a number of self-contained functions including registering an account , providing user's health condition and history, providing health rating report to business company. What's more, this app is base on an Database-centric architecture, each sub-system fetch and analysis the data from the database and pass the result back to the database. We may also have a local database that store the contemporary user's info and then update to the cloud database.

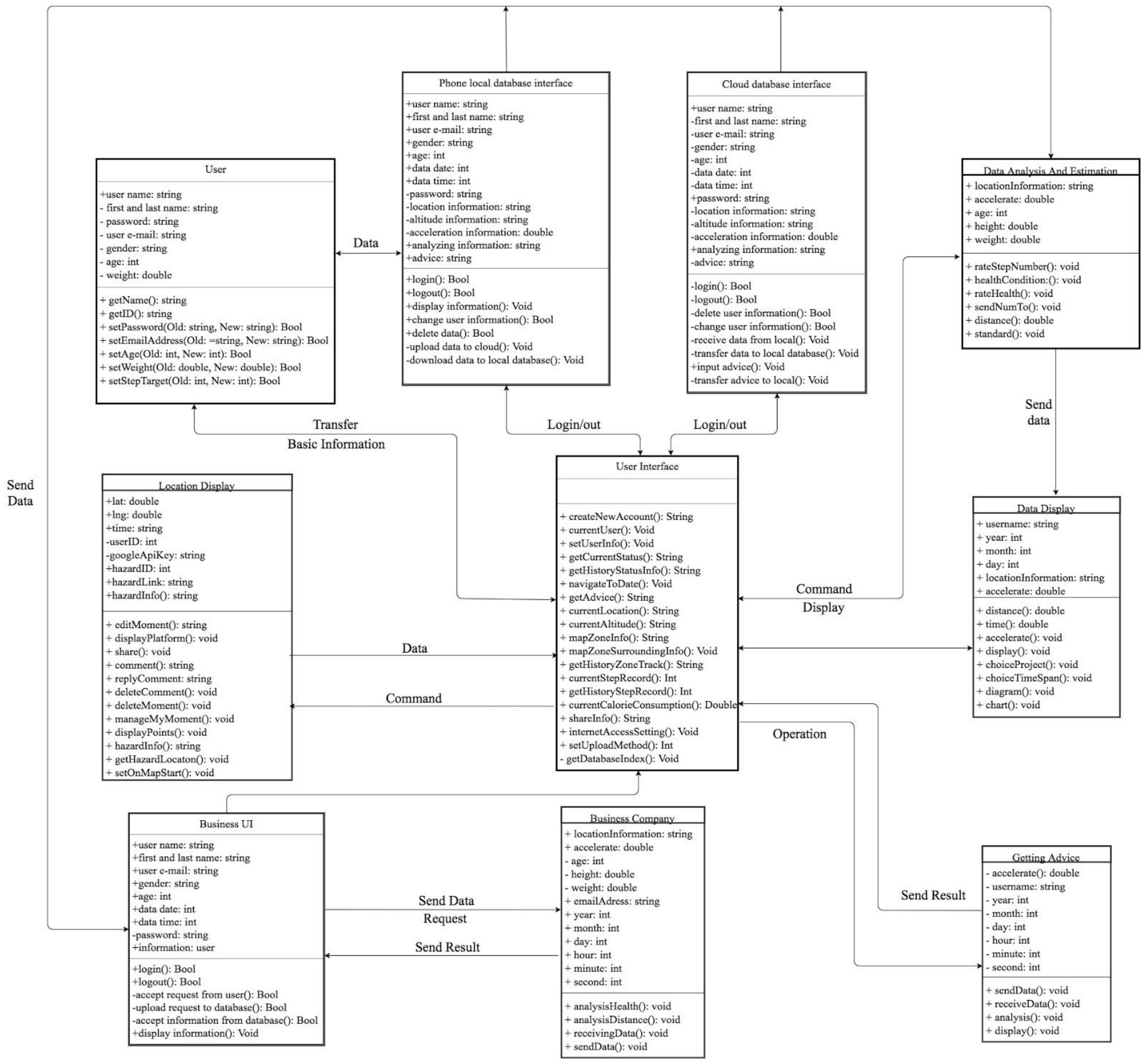
For this part, when people want to give an instruction to the system, the system need to give an feedback at once, and we use the reactor pattern to help us to make the reactor immediately

Mapping Subsystems to Hardware

The subsystem can be mapped into the following hardware component easily. The Web Browser package are allocated in client's PC, and the user would use this to access the application user interface. The Web Server as well as Server Logic package is allocated in the server. They would use this to appropriately analyze and process the user requests. The Server Resource package is allocated in the central server.

8.1 Class Diagram and Interface Specification

Class Diagram



Data Types and Operation Signatures

Sharing Health Condition

- +editMoment(): string

Compile moments that users want to share with others in the platform. Such as words, images or videos.

- +displayPlatform(): void

A platform where users share moments with each other.

- +share(): void

A function that sending moments to the platform.

- +comment(): string

Every user can freely comment others' moments.

- +replyComment: string

Reply comments of moments.

- +manageMyMoment(): void

A function that can help users manage their own moments, such as reply others' comment, delete comment and delete their own moments.

Data Analysis And Estimation

- +standard(): void

Defining the standard that what health condition is better.

- +rateStepNumber(): void

According to the data of all users' step number, comparing the number of the user and his/her friend. Showing user the result.

- +healthCondition:(): void

According to user's everyday data and health standard defined, showing users their health condition.

- +rateHealth(): void

The app has already know everyone's health condition, then rate it and showing the result to users.

- +sendNumTo(): void

Sending health data to business company. After analyzing, the company will give results back to users.

- +distance(): double

According to location information, distance is easy to figure out.

Data Display

- +display(): void

If user A wants to check his/her health data in a diagram or chart. Just selecting year/month/week then health data and the corresponding chart will display on screen.

- +chooseProject(): void

Choose what form user want, such as chart or diagram.

- +chooseTimeSpan(): void

Choose time span. for example, data of 5 days or a month that user want to know.

- +diagram(): void

Showing diagram.

- +chart(): void

Showing chart.

Getting Advice

- +sendData(): void

Sending data to business company.

- +receiveData(): void

Receiving data from business company.

- +display(): void

Displaying data on the screen.

Business Company

- analysisHealth(): void

According to users' everyday data, analyzing their health condition

- analysisDistance(): void

According to users' everyday data, analyzing movement condition.

- receivingData(): void

Receiving data from users.

- sendData(): void

Sending data to users.

Phone local database interface

- -Upload data to cloud(): system

Upload all collection data to the cloud system.

- -Download data to local database(): system

Download all collection data to the local system.

Cloud database interface

- -Transfer data to database(): system

Transfer selection data to the local database.

- -Receive data from local(): user

Accept all data from local database.

User

- + getName(): String

A function that users can get usernames with IDs.

- + getID():

StringA function that users can get IDs with usernames.

- + setPassword(Old: String, New: String): Bool

A function that users can set and change their passwords.

- + setEmailAddress(Old: String, New: String): Bool

A function that users can set and change their Email addresses.

- + setAge(Old: Int, New: Int): Bool

A function that users can set and change their age information.

- + setWeight(Old: Double, New: Double): Bool

A function that users can set and change their weight information.

- + setStepTarget(Old: Int, New: Int): Bool

A function that users can set and change their step record targets (goals).

User Interface

- + createNewAccount(): String

To create a new account for new users.

- + currentUser(): Void

A function that let users to login with their account.

- + getCurrentStatus(): String

Providing the user with his/her current health condition information.
(Excellent/fine/ordinary/weak/dangerous)

- + navigateToDate(): Void

Navigation to a specific date in user’s status history list.

- + getAdvice(): String

Providing advice for the user, based on his/her health condition.

- + currentLocation(): String

Showing user’s current location.

- + mapZoneInfo(): String

Showing the name of location.

- + mapZoneSurroundingInfo(): Void

Providing the geography information around the current location. (Weather/humidity etc.)

- + getHistoryZoneTrack(): String

Tracking user’s location history.

- + currentStepRecord(): Int

Showing current step record of the user.

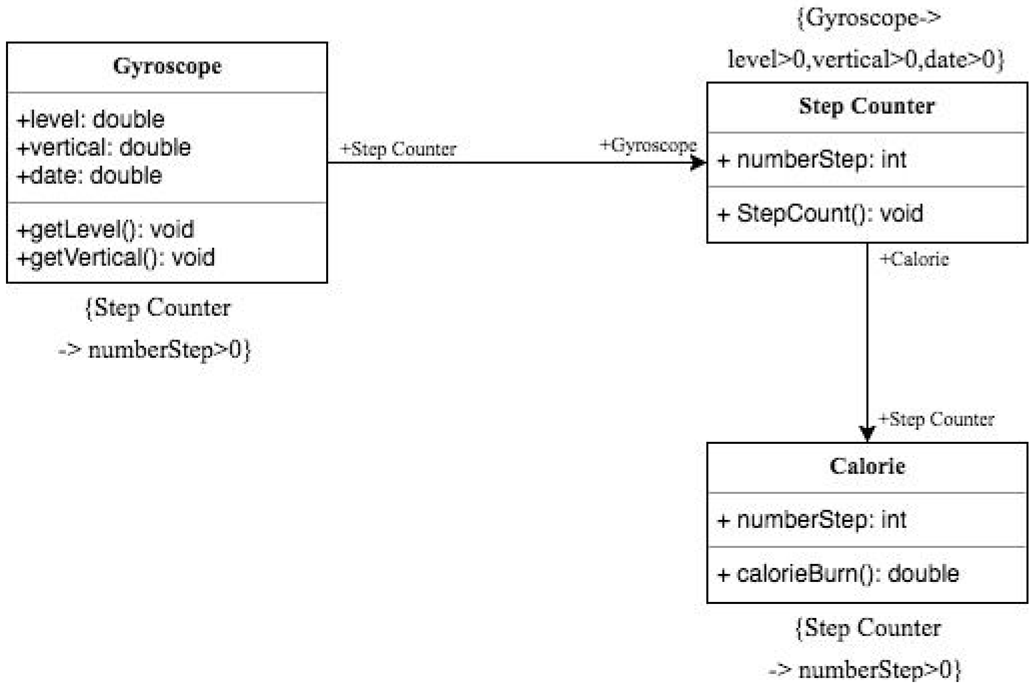
- + currentCalorieConsumption(): Double

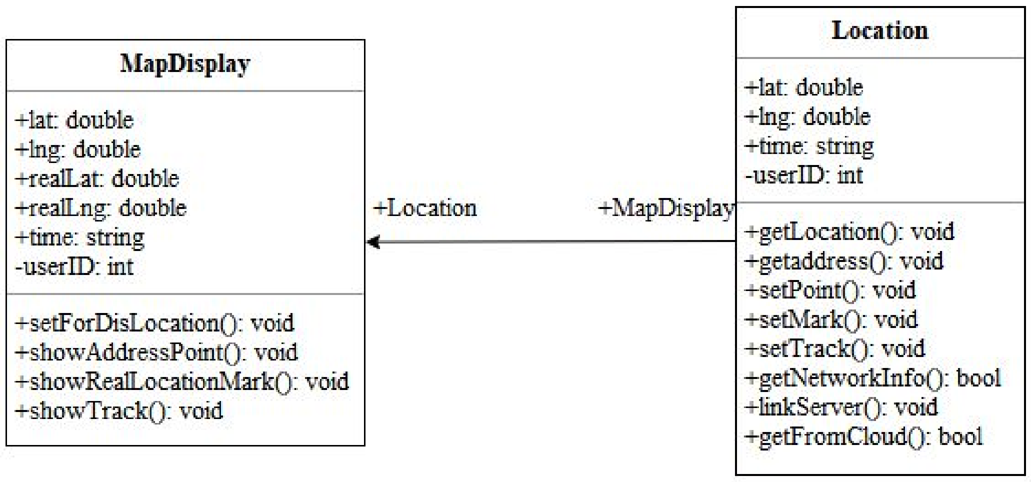
Showing current Calorie Consumption of the user.

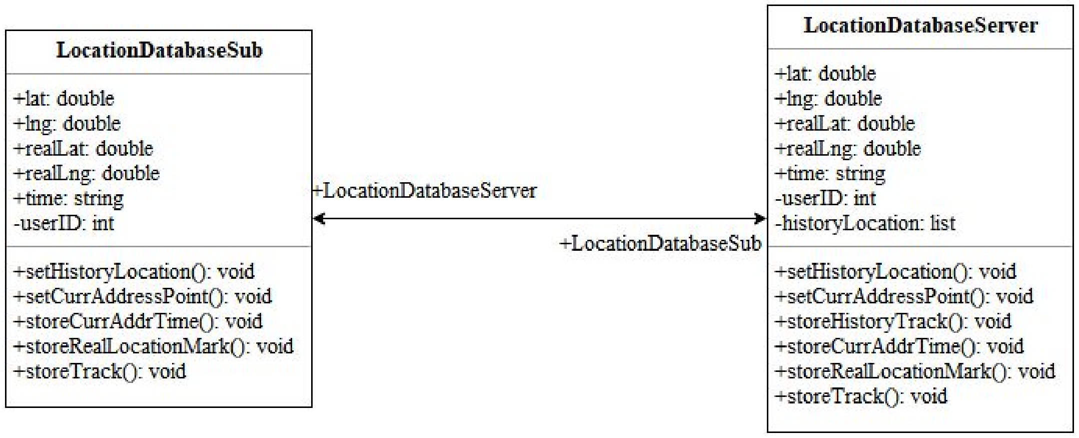
- + setUploadMethod(): Int

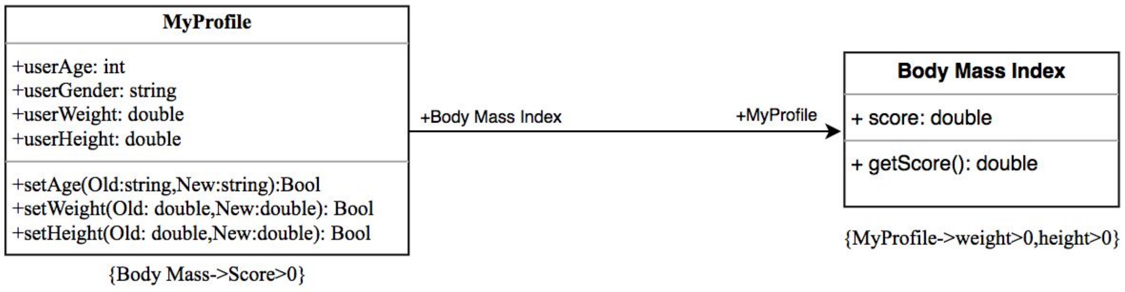
Setting whether to upload user info via a data or wifi connection.

8.2 OCL Contract Specification









8.3 Traceability Matrix

User: The actor that operate on the app
User Interface: The space where system interacts with Users
Data Display: Portal view of user information and advice
Data Analysis and Estimation: Studying health condition of users by their health data
Phone local database interface: Runs checks for local data upload in temporary local database
Cloud database interface:The place for Administrators or Users to interact with cloud database
Getting Advice: Providing the build in Advice tag to User Interface
Business UI: The space with interacts with Business Company
Business Company: A company that give professional advice to users.

Domain concept	User Interface	Saystem	Business Company	Database
User	X			
User Interface	X	X		
Data Display	X	X		X
Data Analysis and Estimation		X		X
Cloud database Interface				X
Phone Local Database Interface				X
Getting Advice		X	X	
Business UI			X	
Business Company			X	

Important Description of the Traceability Matrix:
The above four Domain Concepts are implemented through many detailed activities in our cell phone application, which are shown as follows:
The User Interface:

- MapsActivity
- MainActivity
- MY_PROFILE
- MyLocation
- Caloryandexercies
- BodyMass
- Calorie
- stepcounter
- BeforeOrAfterCalendarView
- RecordsCalenderItemView

Database:

- MobilePost

- PostData
- DBOpenHelper
- StepDataDao

The system (including the cloud server service):

- Rating
- Hazard
- Findlocation
- Details
- AlxLocationService
- AlxLocationManager
- AlxAsynTask
- Advice
- StepEntity
- Constant
- ServerConfig
- StepService
- StepCountCheckUtil
- TimeUtil

9. System Architecture and System Design

Architectural Styles

The architectural style of this health monitor system includes principles from a variety of styles. Our system incorporate the client-server model. This is mainly to host user data on a shared server. By hosting all user data on the same server it's possible to perform more complex data processing and return the results. An example of this would be to classify gyroscope and accelerometer data to human activity like walking or running. This also grants the ability to construct a dashboard for a partnered company. This partnered company can access a user health summary through selecting their user ID. Service-oriented architecture also plays an important part in our app, as the system provides a number of basic self-contained functions including registering an account, providing user's health condition and history, providing health rating report to business company. What's more, this app is based on an database-centric architecture, each sub-system fetches and analyzes data to store the results for later access back into the database. We also decided on a local database stored on each user's mobile device. This will contain remote access keys and also provide a temporary place to store sensor data until it's ready to be uploaded to the remote database.

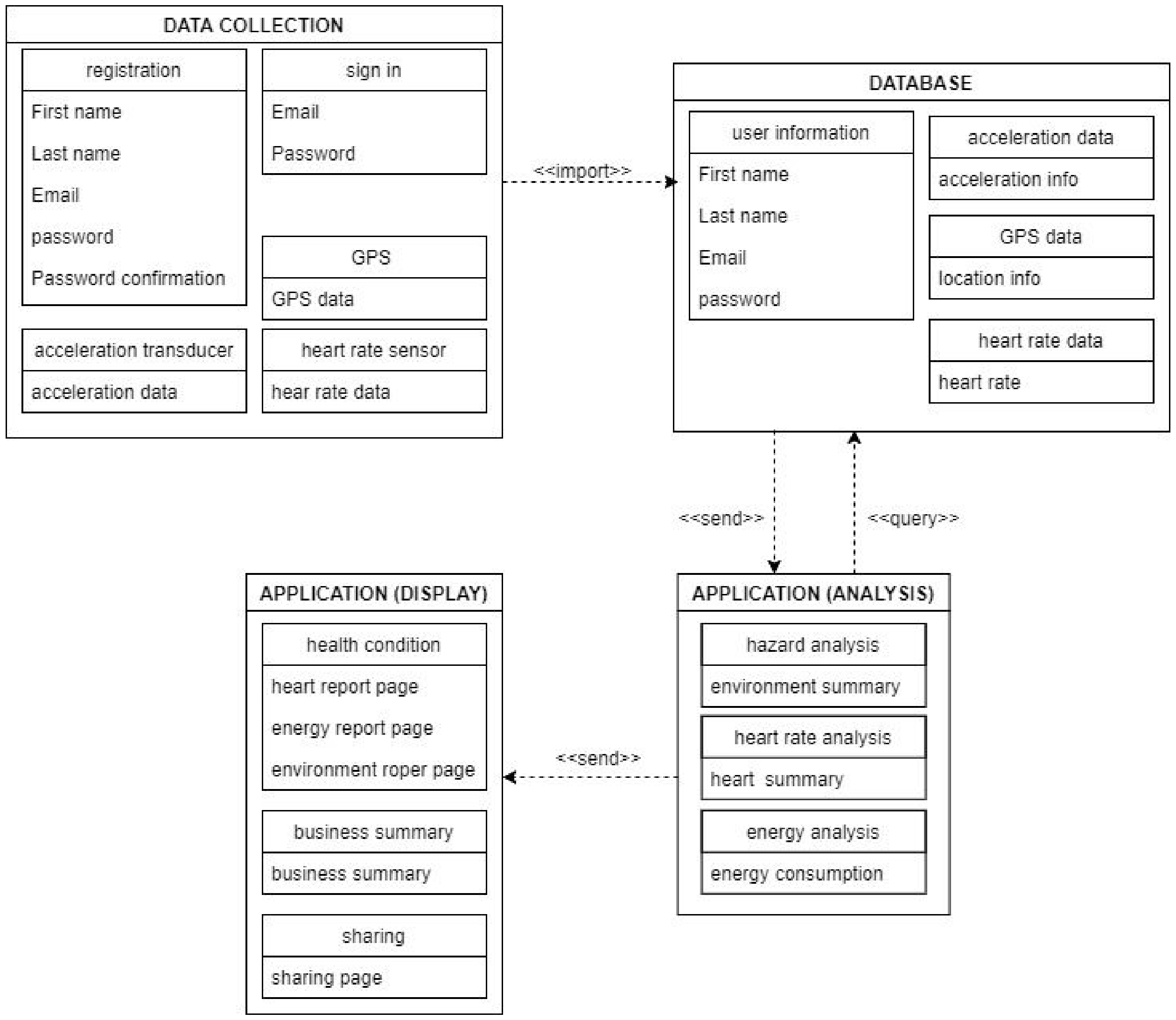
Identifying Subsystems

Our app is designed to be consist of 4 subsystems.

The data collection subsystem, which is responsible for collection of all data needed and uploads them to the remote database. Android built in sensor data is stored into an SQLite database temporarily. Once a day a batch upload is performed to our online MySQL database. Furthermore, since we have to minimize the risk of losing data while pushing and pulling data from our online server and database so all communication will be performed with a TCP connection to provide reliability. To further simplify communication we will be using a pre existing application protocol HTTP which will contain JSON data to fetch and upload data.

The database subsystem can be divided into two parts. The online cloud database MySQL and the SQLite local database. The local database will periodically upload user's information to the cloud database and then to the user profile. The purpose of not directly uploading all information immediately is to avoid overusing a mobile phones battery life. Since the information being used is mainly for a long term analysis for possibly a month to many years it makes more sense to send the information over during a time frame that is not cumbersome to the user such as during the night while the phone is charging or when connected to a WiFi access point.

The application display and analysis part is obvious. After fetching data, we analysis the data and display the corresponding content on our UI



The System Architecture mainly composed of 4 parts.
The first part is Data Collection, which can be composed into following sections:
(1) Registration and Sign in :

User's will be asked to login in or register for a new account, during this process, the app will collect the user's based info to the local database and then set the email as the primary key for the user's profile. Then it will be uploaded to the cloud database

(2) GPS :

The map info collector will constantly collect user's GPS info and save it to local database and then upload to the remote server

(3) Acceleration transducer

We applied this sensor to collect user's steps and walking distance for further use, such as analysing user's calorie consumption

Users enter their personal information in Registration to create a new account.
When the app activate the three sensor collects data and send them to database.

The second part is Database
(1) User information
Saved as a primary key for our database to category user's and also for our business portal
(2) Heart rate data
Collecting heart rate data, can be only used as a factor to display to user's, we actually do not have any further usage on this parameter.
(3) Acceleration data
This is the one of the main factors that we are about to apply in our system, which is used to perform the user's energy consumption for our linear regression procedure. So we will classify user's
(4) GPS data
User information save the data the user enter, and the other three part save the data collected from sensors.
After receiving the requirement from the application, it sends the corresponding data to the next part.

The third part is Application(Analysis)
(1) Hazard analysis
We combine the user's map location history to our hazard point and give out a threaten level based on the hazard level and the distance between the user's habitat, then we can come up with an overall influence on user's health.
(2) Heart rate analysis
Just a basic factor to display for users, we will just collect the data and merge it into our database.
(3) Energy analysis
The basic analysing use steps count and calorie consumption and connect the data with the BMI function.
The fourth part is Application(Display)

This part can display the result of the analysis in several page. The sharing and business summary function is also in this part.

Mapping Subsystem to Hardware

Data Collection:
The data collection part is connected with the following hardware components:

Android Mobile device:
The accelerate sensor and the GPS sensor. This two hardware component
Local database and then Cloud database:
First collect the data collected to the local database and the local database will upload user's info to the remote cloud database periodically.

Database:
The database is our subsystem as well as our hardware components, which is consist of two main part:
Local database:
Store user's info for analysis and display purpose, also a buffer to the cloud database,
The raw data from users will be stored here and passed to the analyzing system for data reduction, get the factor for user's health condition and also store the data factors into this local database.
Remote Cloud Database:
The main database that we are applying to store all user's information and health condition factor, including user's personal info and health condition, business portal data and etc. Each of the available data for this app will be pass to this Cloud database.
This database is mainly connected with 2 parts of our app, one is the local database that will pass data to this main database once a day to minimize the usage of our battery. Also this database will pass the required data to our mobile app when there is a post requirement. We are applying http connection that pass JSON format packages to achieve successful connections.

Data analysis and Data Display:
These two components are only connected with our Android mobile devices. All the method will be build-in methods dealing with the current data sets and the result will be passed to the user's local database.

Persistent Data Storage

Persistent Data Storage is an important facet of the health monitor program. The main reason is that the data collected from the sensor will be consistently updated to the database to establish a statistical model to analysis user's health condition. The persistent Data will be stored in 2 kind of databases, user's local database and the cloud database. The cloud database will storage all metadata for this app. And the local database is actually a buffer that stores the user's data and prepare for uploading.

The local database format: The database will contain these set of fields.:
User's ID, password(if already exist), email address, accelerate sensor data,
GPS data.

The cloud database format:
User ID, password, email address, accelerate sensor data, GPS info,

Network Protocol

Due to the system uploading data to a remote server the primary network protocol will be HTTP. This protocol provides a few benefits that can simplify the data upload tremendously. The main benefit is the existing website can be used to listen to requests directed to a specific mobile URL to handle the mobile user's requests. For example, if a user wants to login to their mobile app the form would be submitted using a POST request to a mobile URL for handling login credentials. The simplest way to format the POST request would be a JSON text containing the form's ID as a dictionary key and string value assigned to the key as the form's value. Once the user has been signed in they can retrieve the remote data required for each mobile view through an HTTP GET request. The server should provide a content response with JSON as well containing the needed data to render to a mobile view. An example of this would be if a user wants to view their heart rate trend the HTTP response will have a list of values pertaining to the user's logged heart rate. From there the mobile application will create a graph showing the user's heart rate trend.

The other benefit of using HTTP is the fact that it's mainly used over TCP. TCP is a transport layer protocol that provides reliable transfer or data as well as congestion control. Overall that means the data integrity of the content will be handled mostly by the underlying network protocols. This also means should the server be overloaded with a high amount of requests the user's accessing the content will be throttled to better serve their content. Many higher level programming languages will already have an HTTP implementation library the generate proper request headers to achieve the project's desired result. This will ultimately make implementation a lot simpler and reliable. The idea is to utilize a flexible, well tested, and proven dependable implementation on an existing protocol (HTTP) and build our network needs on top of it. This way if the project demand shifts it will not require a large effort to change direction in the future with a customer's demand.

The final benefit of using HTTP is we can easily provide a bit of network security. This would be done through HTTPS which provides application layer security. This is necessary since anyone sniffing the network while the user is running the POST request with their login details mentioned earlier could potentially learn the login credentials of that user. It's important to not solely depend on a single encryption protocol since they can be cracked later in the future exposing all or your user's data. Most mobile devices will access the internet using WiFi to save on mobile data. Recently as of October 2017 it has been shown that the primary WiFi encryption protocol WPA2 has a fundamental security flaw that allows a hacker to easily decrypt WiFi packets. This attack has been shown to be especially dangerous for Android devices. HTTPS solves this by utilizing a key exchange so a potential hacker listening into the key negotiation will not learn the user's login details. It should be a noted to obtain a valid HTTPS certificate we would need a registered domain name for the remote site which can be done upon production deployment.

Global Control Flow

Execution Orderliness:

Our app is mostly procedure-driven and each customer may go through the same steps. Generally, most user's will go through the same registration and login procedure. For instance, once customer has created the account and try to check his health status, he/she have to follow the build-in procedure to get the final result and advice.

Time Dependency:

Our system use several timer to control the data collection procedure and automatic logoff procedure. For example, we collect user's info such as position or energy consumption index once a hour, thus the time will be set to 60 minute and then the corresponding data will be transferred to the local database and then the cloud database.
Furthermore, we may apply a timer for log-off. If the customer do not make any action in several hours, the program will automatically log off. During logoff time, the user's info will be collected into the local database but user cannot check the history log. This is one way of protecting user's privacy.

Concurrency:
Our system may use multiple threads. There are some sensors, which requires multiple thread in order for them to work all at the same time. For the database part, we are applying http and there may not me multiple thread method to be applied.

Hardware Requirements

- User Requirements
    - Android Mobile Device
        - Version 4 or Higher
        - 200 MHz Processor
        - 2 GB of RAM
        - 1080x1920 Resolution
        - 50 MB Available Storage

- Mobile Sensors (Optional per Feature)
    - Heart Rate Sensor
        - Heart Rate Predictor Feature
    - GPS Sensor
        - Hazard Area Notification
- Mobile Network or WiFi Access

- Service System Requirements (Demo Purpose Only)
    - Remote Application Server
        - Ubuntu 16.0.4.1 x64
        - 2.40 GHz Intel CPU
        - 1 GB RAM
        - 30 GB Storage Space
        - MySQL 5.7
- Service System Requirements (Deployment)
    - Remote Application Server
        - Ubuntu 16.0.4.1 x64
        - 2.40 GHz Intel CPU
        - 128 GB RAM
        - 1 TB Storage Space
        - MySQL 5.7

10. Algorithms and Data Structures

GPS

Based on the observation of the distance (or distance difference) between GPS satellites and the antenna of the user receiver, the position corresponding to the receiver antenna, is the position of the observing station, is determined on the basis of known instantaneous satellite coordinates. The essence of GPS absolute positioning method is the space intersection in surveying. In principle, the observing station is located at the intersection of the sphere with the three satellites as the center of the sphere and the radius corresponding to the intersection with the plane where the observing station is located. Because GPS uses the principle of one-way ranging, the actual observed distance from the station to the satellite contains the influence of the satellite clock and the receiver's clock synchronization difference. The satellite clock error can be corrected according to the clock error parameter given in the navigation message The clock error of the receiver is generally unpredictable. It is usually treated as an unknown parameter in the data processing and coordinates with the observatory.A station solves four unknowns in real time and requires at least four simultaneous pseudo-range observations. Absolute positioning can be based on the state of the antenna is divided into dynamic absolute positioning and static absolute positioning. Whether dynamic or static, the observations are based on the measured satellite pseudoranges.

Rating

In rating part, since the Machine Learning algorithm was not used at this time, there are mainly two kinds of approximate mathematical algorithms used so far. One is the algorithms of Counting Steps and the other is the algorithms of Calculating the Calorie Consumption. Many products provide their own scoring mechanism. Power of Vitality. Static Score. Provides rewards for good biometric screening results. Used to monitor employee health. Vitality is an interactive and personalized wellness program that makes it easy for you to live your healthiest life. Our program is trusted by millions of members throughout the world. HealthyEnviron: Healthiest cities in America. Air Quality, "The goal of HealthyEnviron is to increase public awareness of how environmental hazards vary across the United States and how these factors contribute to health and chronic illness."

Step counting part:

For the step counting part, the x, y, z value extracted from the accelerator meter will appear as regularly vibrating graphs while the user is walking--no matter what kind of position the smartphone is, which could be taken into considering via different threshold values for one count. For example, the previous four counts beyond the threshold could be taken as the start signal, and the continue six counts within the threshold could be taken as the ending of counting. Then the rest is relatively easier part to count the coming steps.

Calorie Consumption calculating part:

For the calculating of the Calorie Consumption part, because of cannot use wearable devices other than mobile phones, the data obtained in this project are mainly calculated for showing calorie consumption while walking or running. The consumption rates of the different status of the user are not the same. The system requires users to input their weight, height, gender, and age for further calculating. For example, the Calorie consumption of running can be put as a factor multiplies the weight and the distance of running; the consumption could also be calculated as two factors multiplies the weight and the time spent on running/walking (the factors can be calculated via the speed during this time from map track information). Additionally, the factors are also influenced by age and gender of one person.

One of the more standard and most accurate ways to calculate the equation is to use the calorie expenditure formula below. It comes from the Journal of Sports Sciences and provides a formula for each gender.

Men use the following formula:

Calories Burned = [(Age x 0.2017) — (Weight x 0.09036) + (Heart Rate x 0.6309) — 55.0969] x

Time / 4.184.
Women use the following formula:
Calories Burned = [(Age x 0.074) — (Weight x 0.05741) + (Heart Rate x 0.4472) — 20.4022] x Time / 4.184.

Health Advice

Getting advice is a part to remind users about their bad behavior, helping them notice the problem and then doing better. It give general feedback to a user about their health
Collect information about daily activity. Some actions like: Walking, Sitting, Standing and Laying. Some locations like: Report nearby locations negatively impacting their health. Cross check with medical health records. Living near a power plant causes increase in cancer risk

The US Centers for Disease Control says "300,000 deaths each year in the U.S. likely are the results of physical inactivity and poor eating habits." Actually, physical inactivity raises the risk of death from heart disease, stroke, colon cancer and diabetes. In order not to be inactive and to reduce health risks, health authorities such as the American Heart Association recommends moderate intensity exercise for either 30 minutes a day for 5 days a week or a total of 2 hours and 30 minutes per week.

In part Energy Consumption and Rating, the system has already analysis motion data of users and got conclusions such as step number and calories burned. Our application will alerts and remind users to get up and move when they have been inactive for a period of time. When users click the icon of Professional Advice, they will get a precise advice that can help them being health.

For users who exercise acute than it should be, our application will also reminds users risks. It is useful for people who spend long hours at sedentary jobs or sedentary recreational activities and people who can not endure the exercise intensity. Helping them moving health may help reduce the health risks.

Hazard Detection

Algorithm

In the hazard detection part of our health monitor app, the system will apply two linear process:

1. Fetch the hazard that is close to the user’s living area from the database. The system will require nearby hazard location and the hazard level. Base on the hazard location and level, the corresponding “ Health threaten zone” will be give out and stored in the database

2. Find out if user’s location is close to the hazard location and give out alert. The system will compare user’s location with the hazard zone in the database.

The alert will be give if catch the within bound data.

Data Structures

The hash table will be applied to link each user’s data to the corresponding hazard zone. This will be a great way of facilitating our speed of searching process. One the other hand, this will also save a lot of space for data storage and leave more space for the local database for analysing purpose.

User Interface Implementation

<u>My Profile</u>
The My Profile part will come up and show user’s personal information. When user wants to see his personal information, he needs to click the “My Profile” button, and the App will download user’s information data from database and display with some instruction in the screen.

<u>Hazard Detection:</u>
The hazard detection page is designed for already login user. Within 1 click, a user is able to jump to this page and check nearby hazards in a map view. The corresponding distanced toward the hazard will be display and marked. Furthermore, there will be a text alert displayed for detailed explanation for the hazard type and threat level. The hazards are categorized a group where each category is defined in the “hazard_summary” table. The table provides a brief write up of the hazard’s impact, a bad distance in miles to be within the hazard site, and a cited source for why it’s declared a hazard site. By importing GIS data for facilities that contribute towards the hazard category a intersection calculation can be performed. This is done by drawing a circle within the dangerous distance for each found location and finding all points the user intersects this circles vs the points that don’t intersect. It should be noted that it’s possible to exist within two or more sources of the same hazard. For this case they should be tallied up as a double intersection which gives the possibility of the greater than 100% time spent within that zone. The choice behind this is to avoid giving a misleading result where say one area may have a lot hazards vs another area may have a single source. It’s not a fair comparison to view long term health effects between the two without at least weighing them differently.

<u>Health Rating:</u>
The page displays the user's rating score with a range of 0 to 10 and will add the ranking among user's friends who use the same application in the future. The variable result will just display in

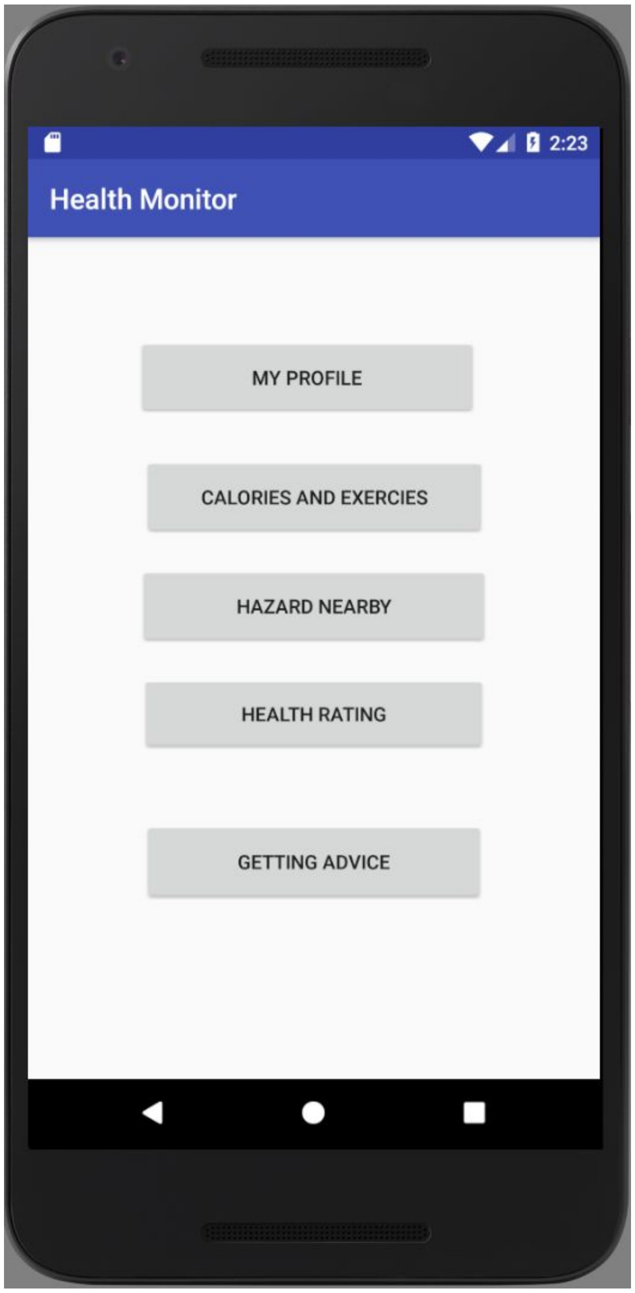
the center of the page. What is more, on this page user can using the bottom button to jump to another page--Get Advice. This Rating page is reached by using main page's navigation and so is the Get Advice page.

Energy Consumption:
This screen is a data display of the data about the user's energy consumption, and it is designed to show two types of data: recent data and long term statistics. The recent data includes "Today's Steps" and "Energy Consumption", and separately shows the row step data collected by acceleration sensor and energy consumption calculated by step data. Then, the long term statistics including "Total consumption this week" and "Daily average consumption this month" are the direct feedback on the user's exercise saturation.

Advice:
When the icon of Professional Advice clicked, screen will jump to another interface and display the content of advice which based on their motion data.
When the icon of Diagnose clicked, screen should be able to jump to an interface and display many kinds of specific disease. Users should be able to select the disease that suit for their symptom and know these diseases better and getting advice simultaneously.

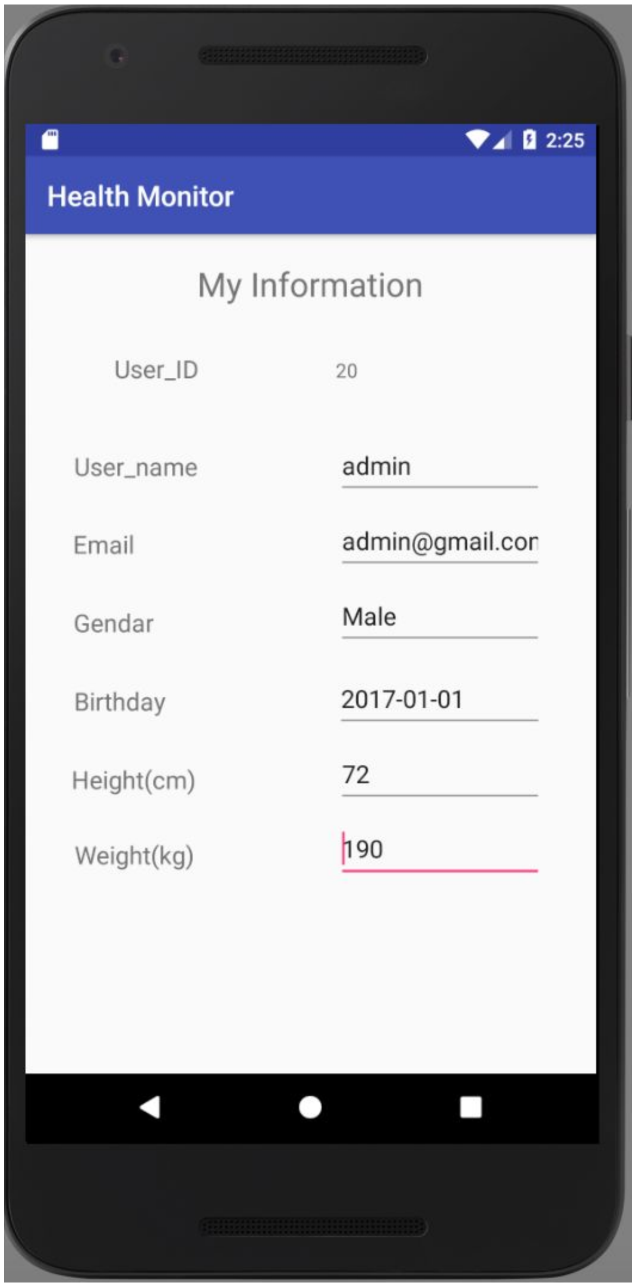
11. User Interface Design and Implementation

Main Interface



My Profile

The My Profile part show information of users. When user wants to see his personal information, he/she just needs to click the "My Profile" button, and the App will download user's information data from database and display with some instruction in the screen.

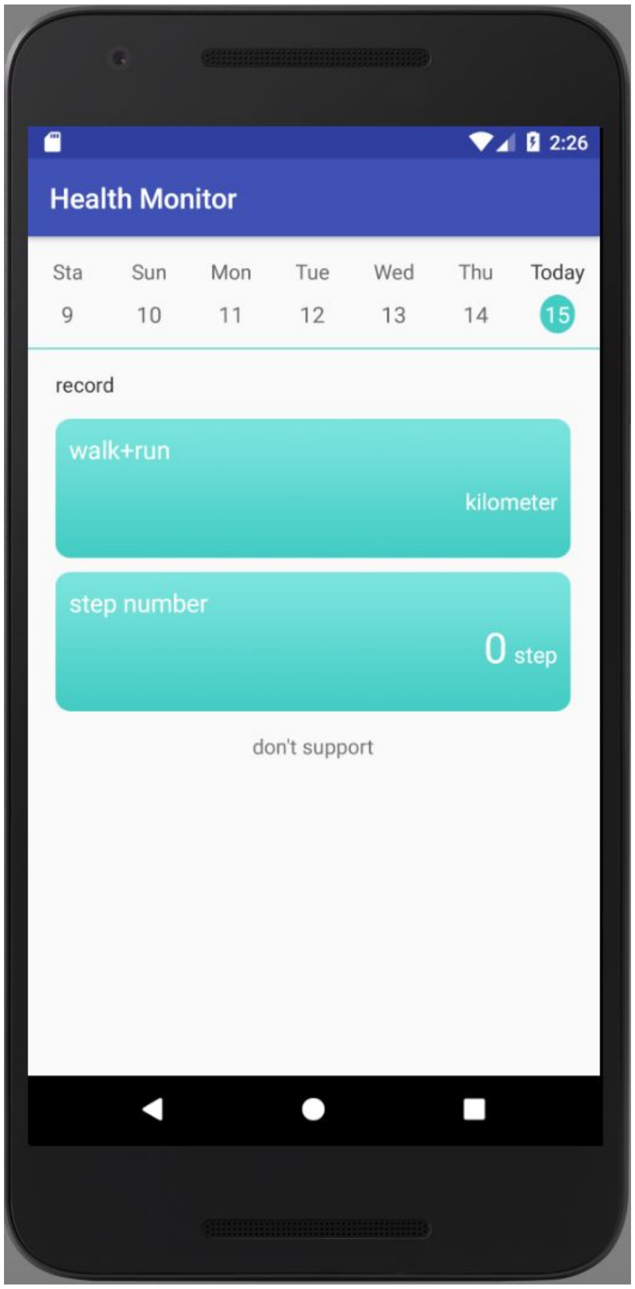


My Information

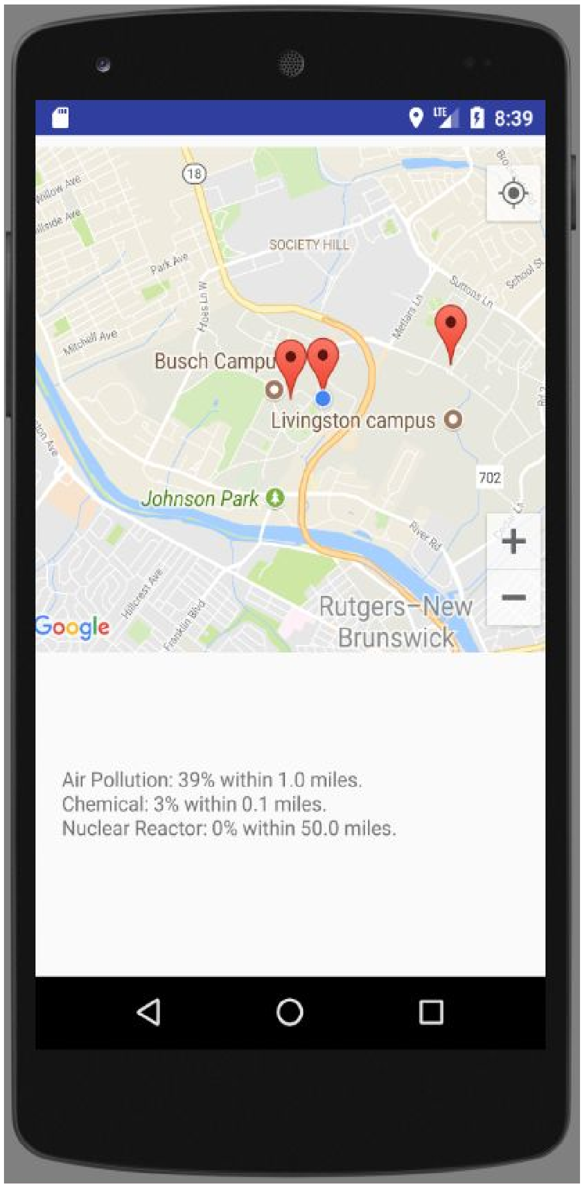
User_ID	20
User_name	admin
Email	admin@gmail.cor
Gendar	Male
Birthday	2017-01-01
Height(cm)	72
Weight(kg)	190

Calories and Exercise

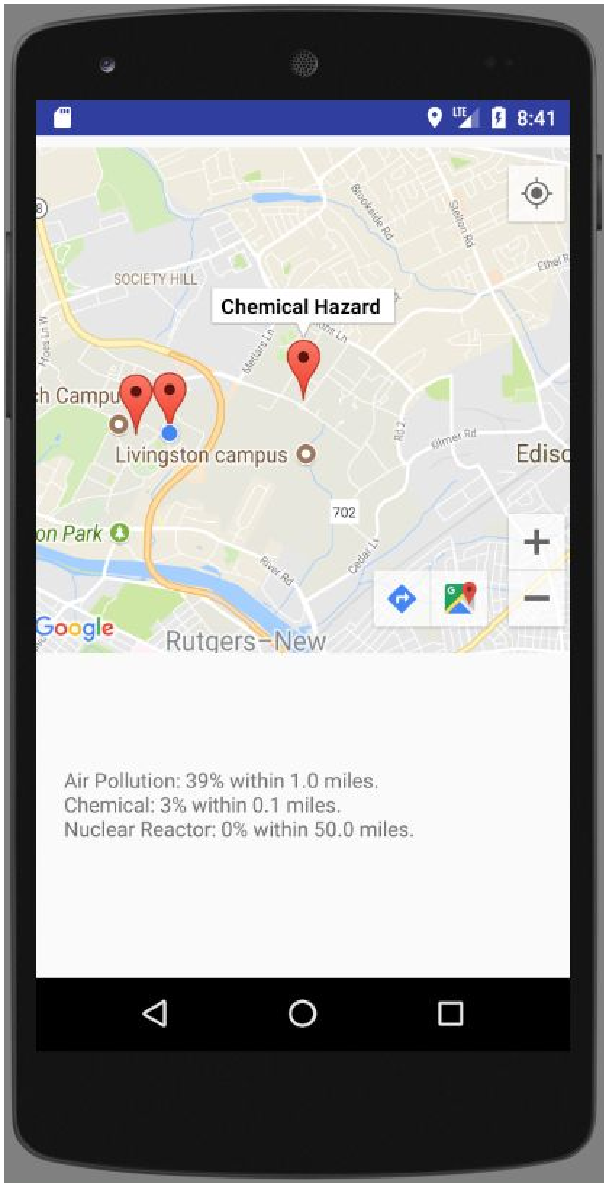
It is designed to show two types of data: recent data and long term statistics. The recent data includes “Steps numbers” and “walk+run (kilometer)”, and separately shows the step data collected by acceleration sensor and distance calculated by GPS information. Then, the long term statistics are able to show users about their history step information. It is a direct feedback on the user’s exercise saturation.



Hazard Detection:
The hazard detection page is designed for already login user. Within 1 click, a user is able to jump to this page and check nearby hazards in a map view. The corresponding distanced toward the hazard will be display and marked. Furthermore, there will be a text alert displayed for detailed explanation for the hazard type and thread level.

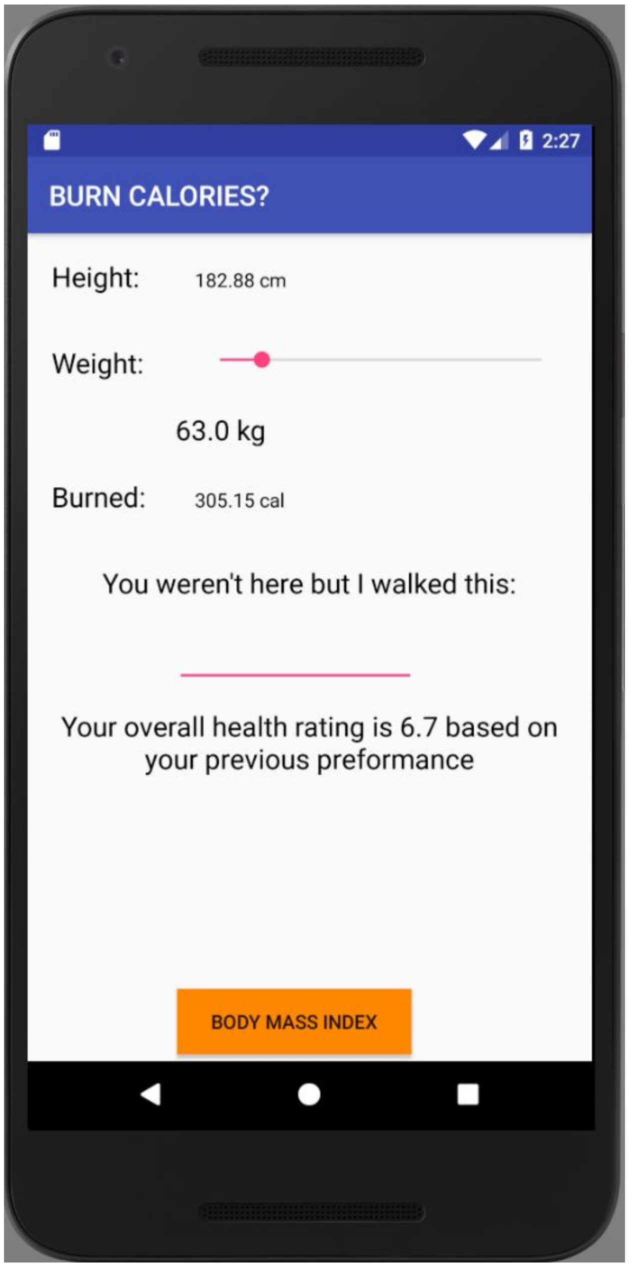


Air Pollution: 39% within 1.0 miles.
Chemical: 3% within 0.1 miles.
Nuclear Reactor: 0% within 50.0 miles.



Air Pollution: 39% within 1.0 miles.
Chemical: 3% within 0.1 miles.
Nuclear Reactor: 0% within 50.0 miles.

Healthy Rating
This screen displays data about the user's basic information and calories burned. Besides, user can input their exercise information since they do not always exercise with the phone.
What's more, the app will calculate health score about the user by BMI formula.


BURN CALORIES?

Height: 182.88 cm

Weight:

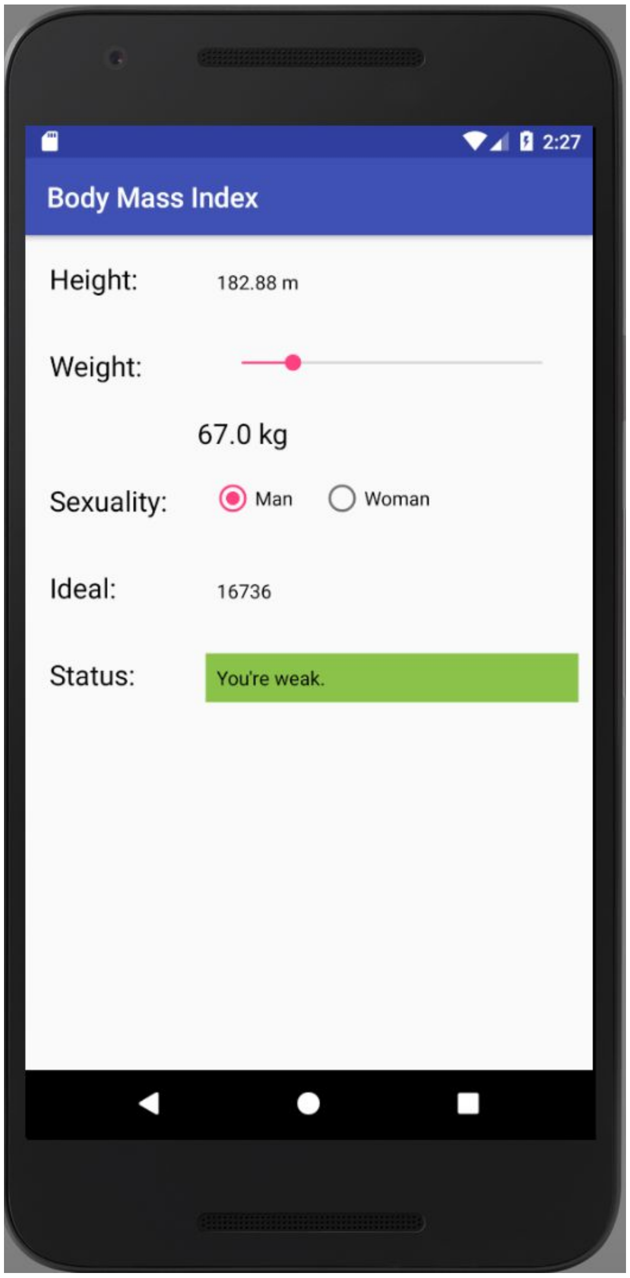
63.0 kg

Burned: 305.15 cal

You weren't here but I walked this:

Your overall health rating is 6.7 based on your previous preformance

Body Mass Index
This page display users' body information and giving result about their health condition. Besides, based on the weight that user input, it will give the ideal height so that user knows their health condition better.



2:27

Body Mass Index

Height: 182.88 m

Weight:

67.0 kg

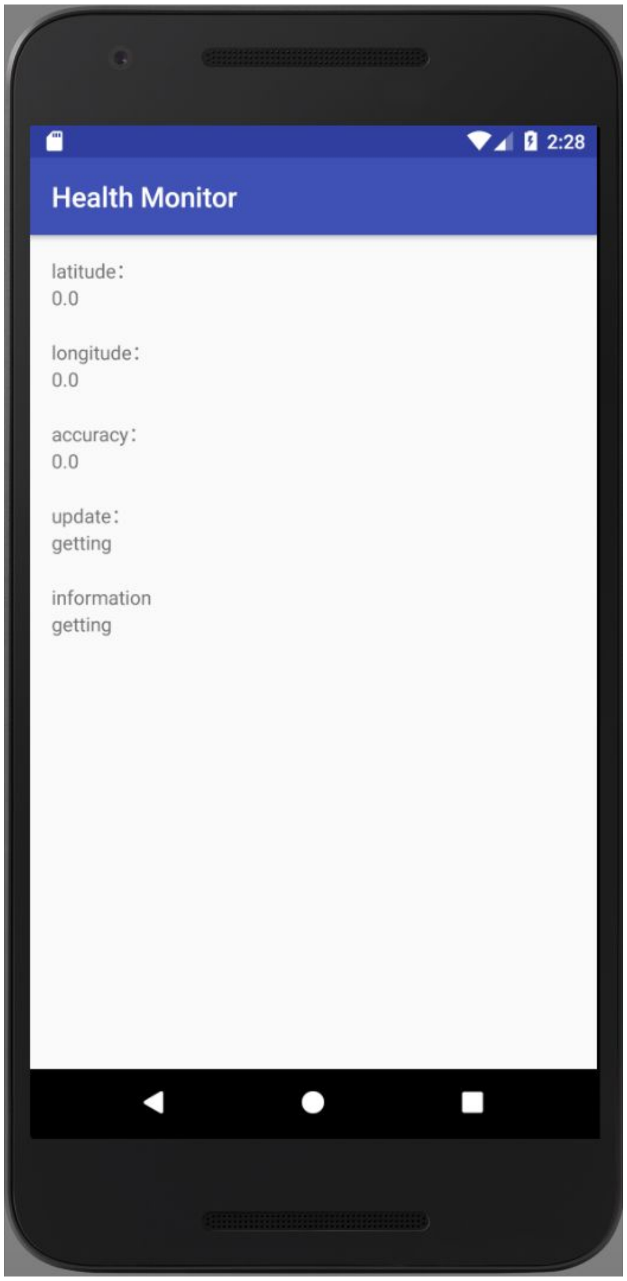
Sexuality: [✓ Man] Woman

Ideal: 16736

Status: You're weak.

Get My Detailed Location

User is able to get their real-time and accurate location from this interface.



latitude：
0.0

longitude：
0.0

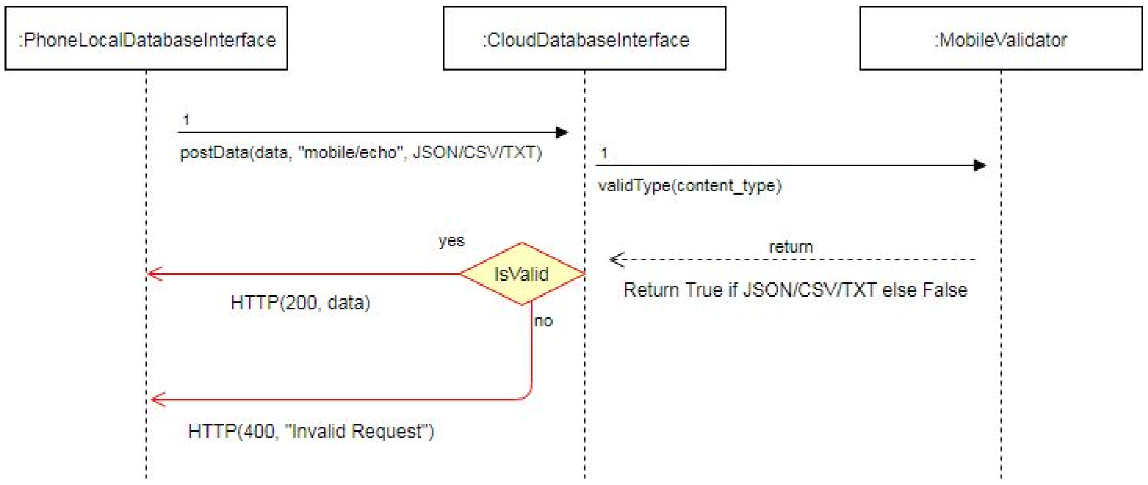
accuracy：
0.0

update：
getting

information
getting

12. Design of Tests

Echo Reply

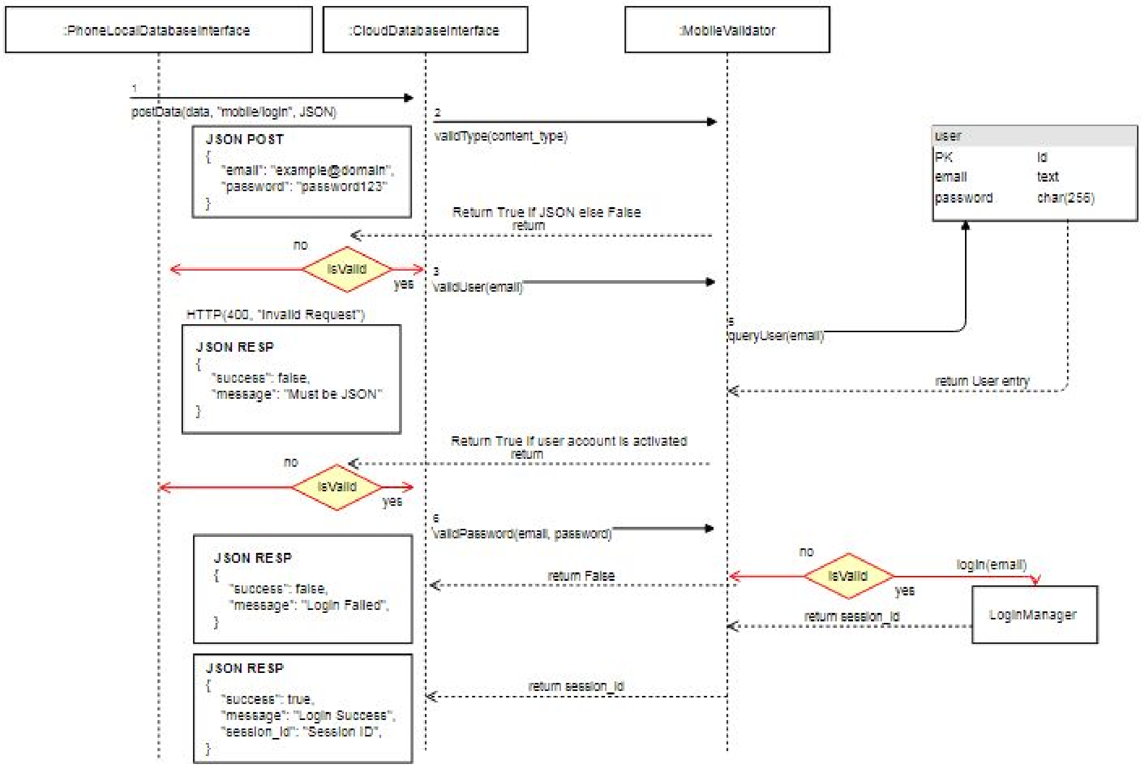

:PhoneLocalDatabaseInterface
:CloudDatabaseInterface
:MobileValidator
1
postData(data, "mobile/echo", JSON/CSV/TXT)
1
validType(content_type)
return
Return True if JSON/CSV/TXT else False
yes
IsValid
no
HTTP(200, data)
HTTP(400, "Invalid Request")

Echo response test between mobile application and remote server. Only accepts content types JSON, CSV, and plain/text. Should return HTTP "200" success code with posted data when valid content type otherwise return HTTP "400" client error code with "Invalid Request" message.

- Content Type not JSON/CSV/TXT
    - **Pass:** 400 JSON RSP
        - Message: Only JSON/CSV/TXT Allowed
    - Use HTTP header content type to declare
    - JSON
        - **Content Type:** "application/json"
    - CSV
        - **Content Type:** "text/csv"
    - TXT
        - **Content Type:** "text/plain"
- Content Type is JSON/CSV/TXT
    - **Pass:** HTTP 200 Original Data Posted to Server (Echo)
    - Use HTTP header content type to declare
    - JSON
        - **Content Type:** "application/json"
    - CSV

        - **Content Type:** “text/csv”
    - TXT
        - **Content Type:** “text/plain”

Mobile Login



```
JSON POST
{
    "email": "example@domain",
    "password": "password123"
}
```

```
JSON RESP
{
    "success": false,
    "message": "Must be JSON"
}
```

```
JSON RESP
{
    "success": false,
    "message": "Login Failed",
}
```

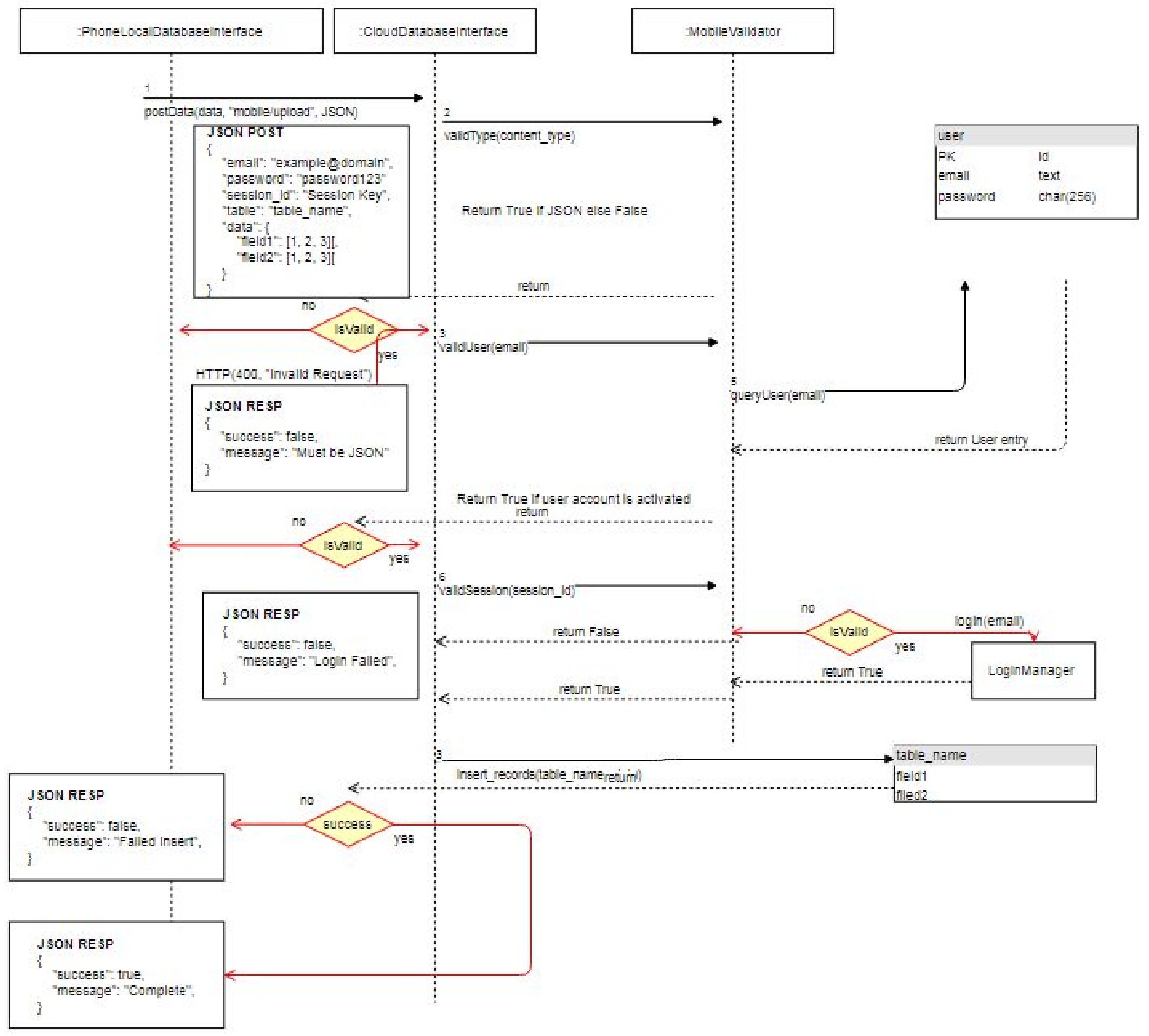
```
JSON RESP
{
    "success": true,
    "message": "Login Success",
    "session_id": "Session ID",
}
```

Login requires a JSON with the email and password. The Validator will check for a valid JSON, check if the user is an existing and activated account, check the password with the hashed password, and return either a success with a session ID or a login failed. Note the session ID is a temporary key embedded in future requests to avoid logging into the application upon every request. Normal expiration is set to one day. If invalid JSON or password provided return JSON with success entry failed. Messages are used for debugging and the mobile app should handle user messages.

- Content Type not JSON
    - **Pass:** JSON RSP
        - Success: False
        - Message: Only JSON Allowed
- User has no account
    - **Pass:** JSON RSP
        - Success: False
        - Message: No Account Found

- User account is not active
    - **Pass:** JSON RSP
        - Success: False
        - Message: Account is Not Active
- Email and password don’t match
    - **Pass:** JSON RSP
        - Success: False
        - Message: Invalid Email or Password
- Login Successful
    - **Pass:** JSON RSP
        - Success: True
        - Session_id”: Login Manager Assigned Session Key

Mobile Upload



```
JSON POST
{
    "email": "example@domain",
    "password": "password123"
    "session_id": "Session Key",
    "table": "table_name",
    "data": {
        "field1": [1, 2, 3][,
        "field2": [1, 2, 3][
    }
}
```

```
JSON RESP
{
    "success": false,
    "message": "Must be JSON"
}
```

```
JSON RESP
{
    "success": false,
    "message": "Login Failed",
}
```

```
JSON RESP
{
    "success": false,
    "message": "Failed Insert",
}
```

```
JSON RESP
{
    "success": true,
    "message": "Complete",
}
```

The URL /mobile/upload is used to handle record insert attempts by the mobile device. Messages are used for debugging and the mobile application handles unsuccessful responses returned to the user.

- Invalid Table Name (Doesn't Exist)
    - **Pass:** JSON RSP
        - Success: False
        - Message: Invalid Table
    - **Valid Table Names:** ["user_steps", "user_location"]
- Invalid session ID (Not Logged In)
    - **Pass:** JSON RSP
        - Success: False

    - Must Login
  - **Invalid Session ID:** Flask Login tracks user sessions. If a session is not found with the provided ID should return False
- Invalid Data (Wrong data type)
  - **Pass:** JSON RSP
    - Success: False
    - Message: Invalid Data Insert
  - **Invalid Data**
    - insert_record method handled by a MVC Flask SQL Alchemy when bad data types
      - Provided char value to datetime SQL Field
        - Example: Providing 0 to `date` in user_steps or user_location
      - Provided duplicate entry for Unique Field
        - Example: date_user_id_unq on user_steps or user_location
    - Length of each fields provided is different
      - Each list item per field in the JSON data section represents a table row so all fields should have the same number of row entries
    - Mandatory fields are not present in the data section
      - Depends on per table
      - “user_steps” table
        - **Required Fields:** [“step_count”, “date”]
        - All other user data pulled from session_id
      - “user_location” table
        - **Required Fields:** [“latitude”, “longitude”, “date”]
        - All other user data pulled from session_id

- Successful Record Insert
  - **Pass:** JSON RSP
    - Success: True
    - Message: Data Insert Successful
  - When all above failed cases are passed

13. History of Work, Current Status, and Future Work

In demo#1, we analyze the problem: No centralized system to track general public health: Long term health studies are generally not conducted, New studies have difficulty finding pre-existing data to build from, New health policies are difficult to monitor. Health feedback is limited:

Doctor visits can be infrequent, User habits are not tracked, Online research can be tedious, Online information may not apply to a specific person.

And our solution to this: Mobile phone app to passively track a user's health to a web server and provide feedback on the results.

And our outpoints are: fits most of android mobile phones, combining Map information with real time Hazards to show Health threatens to User, calorie consumption will be calculated based on user's activation record of a certain period, and a well designed Website to display our data and analysis

Our Technical Challenges are Designing the database to work with future data sources (Medical Records, Family History, Future Features), Moving mobile measurements to a remote server (Secure and Reliable Data Transfer, Easy to Adjust), Giving meaningful advice (Updating with New Studies, Reliable User Prediction, Summarized Findings).

In demo#2, the general description is Give general feedback to a user about their health: collect information about daily activity (Actions: walking, sitting, standing, lying and locations), report nearby locations negatively impacting their health, cross check with medical health records, Living near a power plant causes increase in cancer risk.

Working influence is Many products provide their own scoring mechanism, (Power of Vitality: Static Score, Provides rewards for good biometric screening results, Used to monitor employee health, vitality is an interactive and personalized wellness program that makes it easy for you to live your healthiest life. Our program is trusted by millions of members throughout the world. (HealthyEnviron: Healthiest cities in America, Air Quality) "The goal of HealthyEnviron is to increase public awareness of how environmental hazards vary across the United States and how these factors contribute to health and chronic illness."

Environment Solution: Build GIS Database, create and source environmental hazards (http://138.197.80.193/hazardsummary/). Import GIS data of classified places (http://www.arcgis.com/), Nuclear Reactors, Chemical Plants, High Pollution Areas, (http://138.197.80.193/hazardlocation/), Collect mobile GPS data periodically, Upload the data to a remote server, Find intersecting hazards (http://138.197.80.193/userdash/).

And now our group will collect all report and finish all part which should be submitted.

14. References

1.https://content.sakai.rutgers.edu/access/content/group/19cbbd6e-6ece-44c3-aef2-d89159f2f1fd/JavaScript/JavaScript_WebDevelopment.pdf
2.https://content.sakai.rutgers.edu/access/content/group/19cbbd6e-6ece-44c3-aef2-d89159f2f1fd/JavaScript/JavaScript_Fundamentals.pdf
3.https://content.sakai.rutgers.edu/access/content/group/19cbbd6e-6ece-44c3-aef2-d89159f2f1fd/JavaScript/JavaScript_DataMining.pdf
4.http://www.ece.rutgers.edu/~marsic/books/SE/book-SE_marsic.pdf
5.http://www.ece.rutgers.edu/~marsic/books/SE/instructor/slides/
6.https://sakai.rutgers.edu/portal/site/19cbbd6e-6ece-44c3-aef2-d89159f2f1fd/tool/54b324f5-8914-492f-bf8c-cc0196d1f92f?panel=Main
7.http://www.ece.rutgers.edu/~marsic/books/SE/projects/HealthMonitor/2014-g5-report3.pdf
8.http://www.ece.rutgers.edu/~marsic/books/SE/projects/HealthMonitor/2014-g12-report3.pdf
9.http://www.ece.rutgers.edu/~marsic/books/SE/projects/HealthMonitor/2013-g7-report3.pdf
10.http://www.ece.rutgers.edu/~marsic/books/SE/projects/HealthMonitor/2013-g8-report3.pdf
11.https://github.com/bfetler/human_activity
12.http://www.arcgis.com/
13.http://www.ece.rutgers.edu/~marsic/
14.https://view.officeapps.live.com/op/view.aspx?src=http://www.ece.rutgers.edu/~marsic/books/SE/instructor/slides/lec-14%20Metrics-Intro.ppt
15.http://www.ece.rutgers.edu/~marsic/books/SE/projects/HealthMonitor/
16.http://hukai.me/android-training-course-in-chinese/location/index.html
17.https://developers.google.com/maps/documentation/android-api/
18.https://en.wikipedia.org/wiki/Google_APIs#Google_Apps_Script
19.https://developer.android.com/reference/android/location/Geocoder.html
20.https://google-developers.appspot.com/maps/documentation/utils/geocoder/
21.https://blog.mapbox.com/introducing-the-geocoder-library-for-android-b19c191c9c92
22.https://github.com/egemenzeytinci/stepandcalorie
23.https://en.wikipedia.org/wiki/Acceleration