Software Requirements Specification

Prepared by Newline

for the project Hypnos

METU - Department of Computer Engineering

CENG 491 Senior Design Project I

Fall 2015-2016

Table of Contents

Tables

Figures

1. Introduction

1.1 Problem Definition

Even if people spend nearly one third of their lives at their sleep, they don’t track it. Sleep data is highly valuable and since people do not track it, it is lost. Current solutions related to these are problem are either not easy to use or they gather very shallow data and do not analyze sleep deeply. People need a wearable device to track their sleep which is at reasonable prices and is portable so that anyone can use it in their home. This project aims to meet these needs. It aims to collect, analyze, visualize sleep data and by making inferences, optimize sleep by informing user. Our product will be guessing the right time to wake up the user and act accordingly. Our goals include understanding some possible health problems by analyzing data continuously.

1.2 System Overview

Infrastructure of Hypnos consists of five parts: Wearable, fixed devices and cloud server, mobile application and website.

Wearable device consists of various sensors that constantly gather data from the user. The device will join all the data coming from sensors and push them to fixed device for further computation.

- Fixed device is responsible for processing the data in real time. It is the main computation unit of the system. Fixed device also maintains the communication with the cloud server, and sends user data in a daily basis.
- Cloud server is the main storage unit for past user data. It acts as a bridge between the wearable device and mobile and web applications. It will handle users and will be answering requests coming from mobile application and website.
- Mobile application will be used to visualize and show daily user data. The app will be an easy to use platform suited for end users.
- Website will be used by users to see more detailed data than the mobile application. The main purpose of the website will be analysis of this data. It will be more suited for data analysts.

1.3 Definitions, acronyms, and abbreviations

Term	**Explanation**
API	Application Programming Interface
SRS	Software Requirement Specification
GUI	Graphical user interface
HTTP	Hypertext Transfer Protocol
HTML	HyperText Markup Language
I2C	Inter-Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
JSON	JavaScript Object Notation
OS	Operating System
UML	Unified Modeling Language
Wi-Fi	Wireless Fidelity
XBee	Radio Module Using ZigBee Protocol
ZigBee	Low-power Radio Communication Protocol
SMS	Short Message Service

Table 1: Terms and Explanations

1.4 Assumptions and dependencies

- We assume that sensors will be able to give data frequently.
- We assume that sensors’ accuracy will be enough to meet our needs.
- Computing power of the fixed device will be adequate.
- Internet connection will be available for our system to work.
- Users will have a mobile phone compatible with the app.
- Users will have WiFi connection in their homes.

2. Overall description

2.1 Product functions

In this project, sensors placed on wearable device will collect information from person with a configured frequency and transmit this data to fixed device. This fixed device will also have sensors collecting data and will process all sensor data in real time. Unless this device detects an emergency situation, it will send processed data to cloud component in a daily basis. If it detects an emergency such as heart attack, predetermined people will be alerted via an automatic phone call or SMS etc.. The cloud component will take data from fixed device and apply machine learning algorithms to add a meaning to data. This component will understand light stages of sleep in order to wake up the user in the right time, will figure out personalized sleep patterns and hopefully will diagnose some health problems like epilepsy crisis or sleep apnea. Users will be informed about their sleep based on time and sensor choice through web application and mobile application.

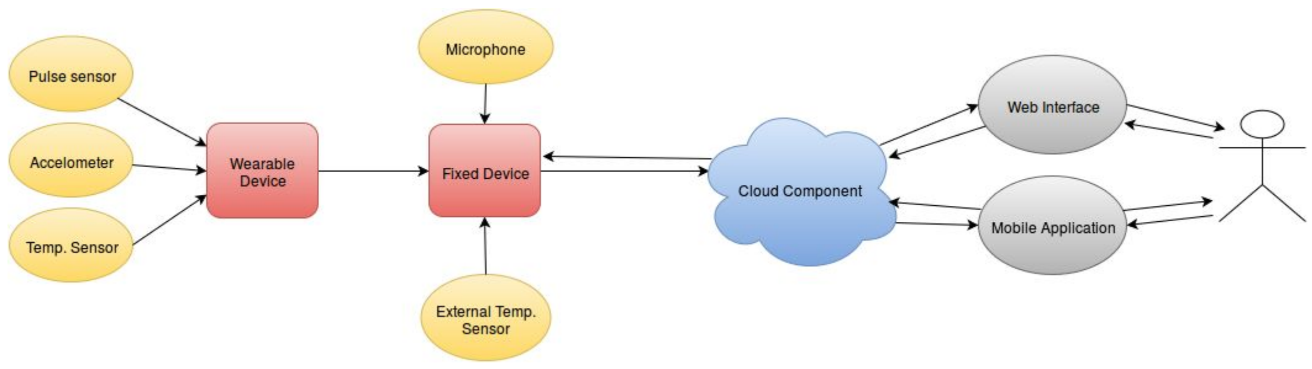

Figure 1: Overall system design

2.1.1 Use-case model survey

2.1.1.1 User Use Cases

Use Case	Description
Login	Users need to login to start using system.
Register	Users need to register themselves and their devices to system.
Set Alarm	Users should set an alarm to be woken up by Hypnos.
View Profile	Users can see their profile which includes health information.
Edit Profile	Users can change their personal & health information.
View Emergency Records	Users can see history of emergency situations.
View Disease Predictions	Users can see possible diseases predicted by Hypnos.
Filter	Users can see their information filtered by time and sensor choice.

View Dashboard	Users can see general information about their sleep.
View Analytics	Users can see the results of the applied machine learning algorithms.
Edit emergency contacts	Users can edit contacts to be informed in emergency situations

Primary actors: User - Secondary actors: None
Active actors = User - Passive actors = Emergency Contact

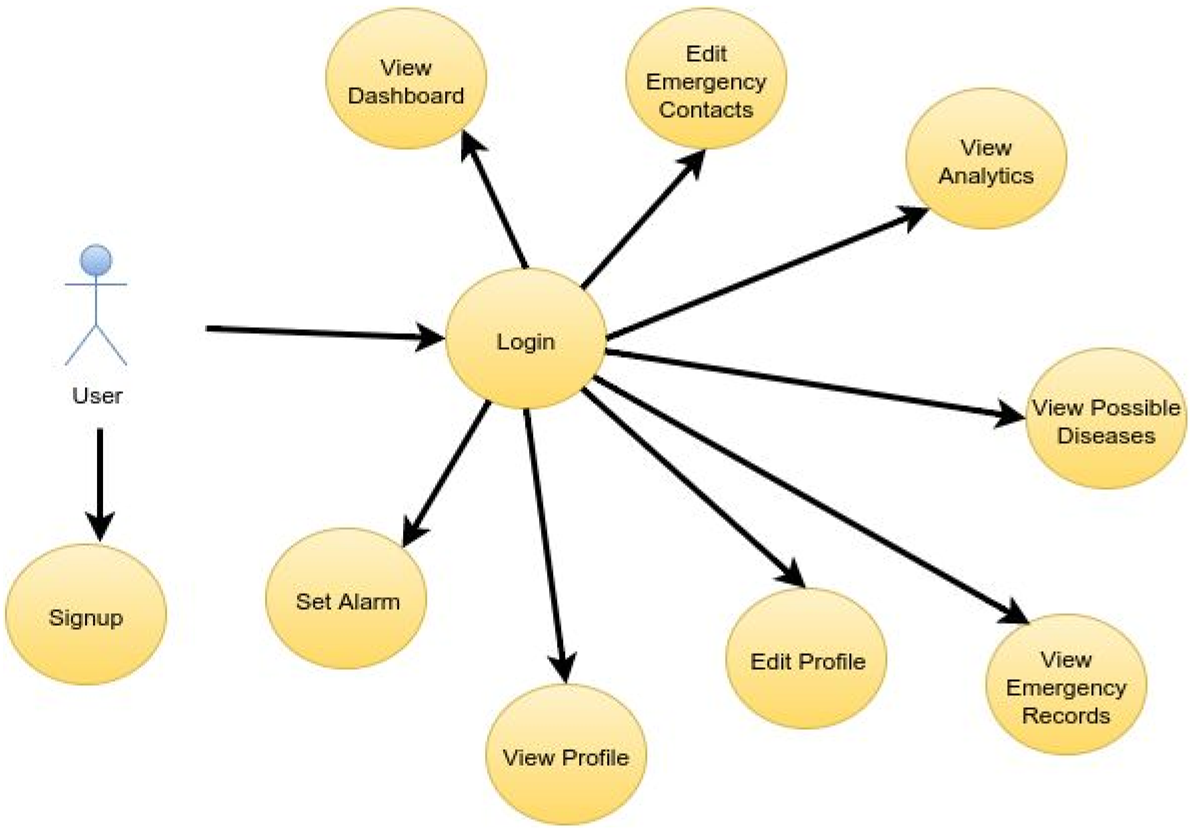

Figure 2: User use cases

2.1.2 Actor survey

User

This actor refers to the people who own a device. A user will be able to view his/her personal information based on time and sensor choice. Users also see health record, analysis and configuration of devices.

Emergency Contact

This actor refers to the people to be informed by system when there is an extreme situation.

2.2 Interfaces

2.2.1 User Interfaces

User interface of web application will be comprehensible and easy to use. There will be 4 basic pages; dashboard, analytics, event history, disease prediction. The characteristics of web application's user interface will be as follows: Interface will be shown by web browsers. There will be dashboard page. In dashboard page, user will be able to see charts showing each sensor data separately and an initial chart that includes all sensor data. In analytics page, user will be able to see his or her results from application of machine learning algorithms. On event history page, user will be able to view list of past events detected by the device. On disease predictions page, user will be informed about the diseases that s/he might have.

System will also have an android application. Application will provide same functionality as the web application, but it will have a simpler interface. In addition, mobile application will show the user's own sleep information in a daily basis.

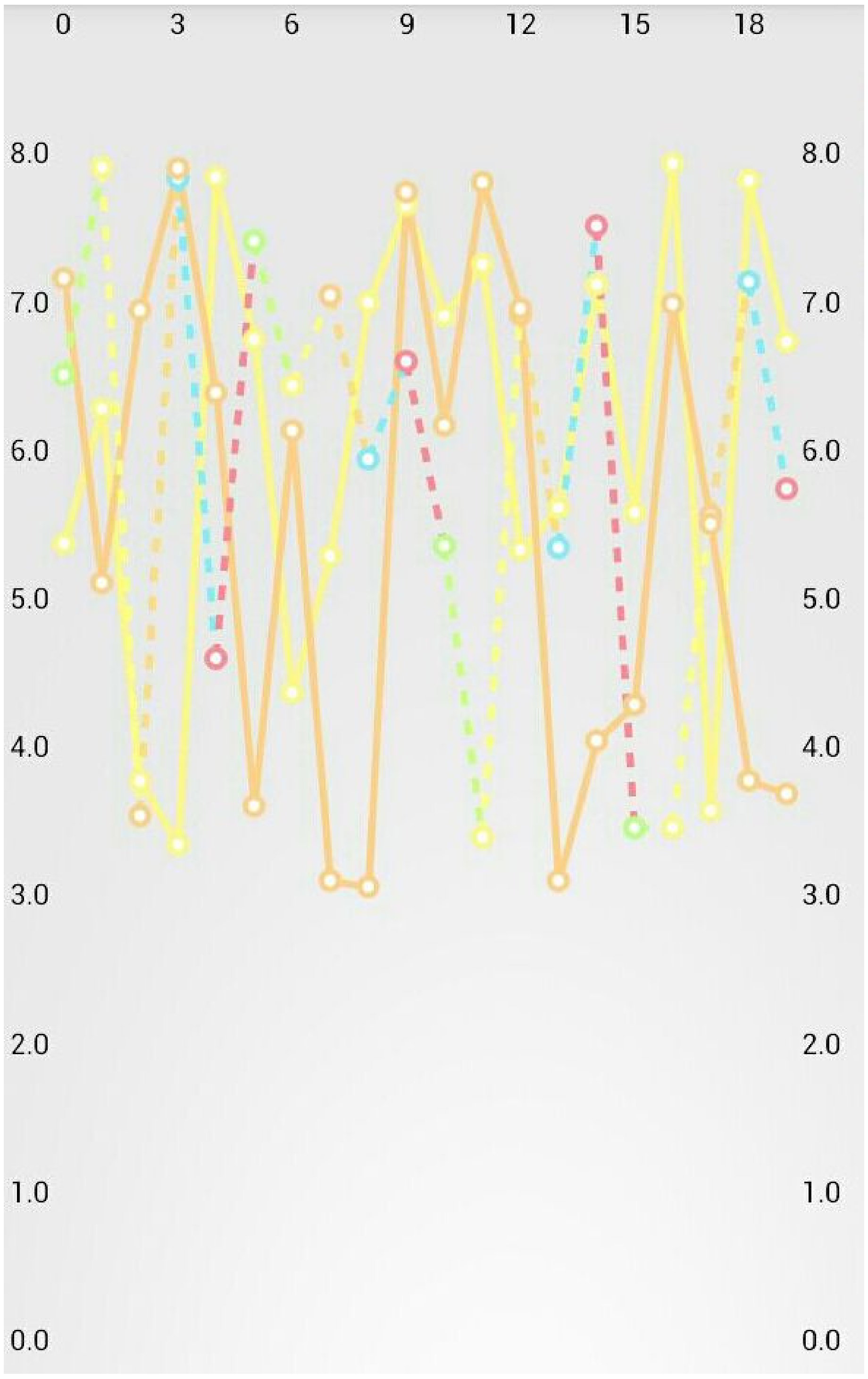

Figure 3: Mobile - Sleep stats

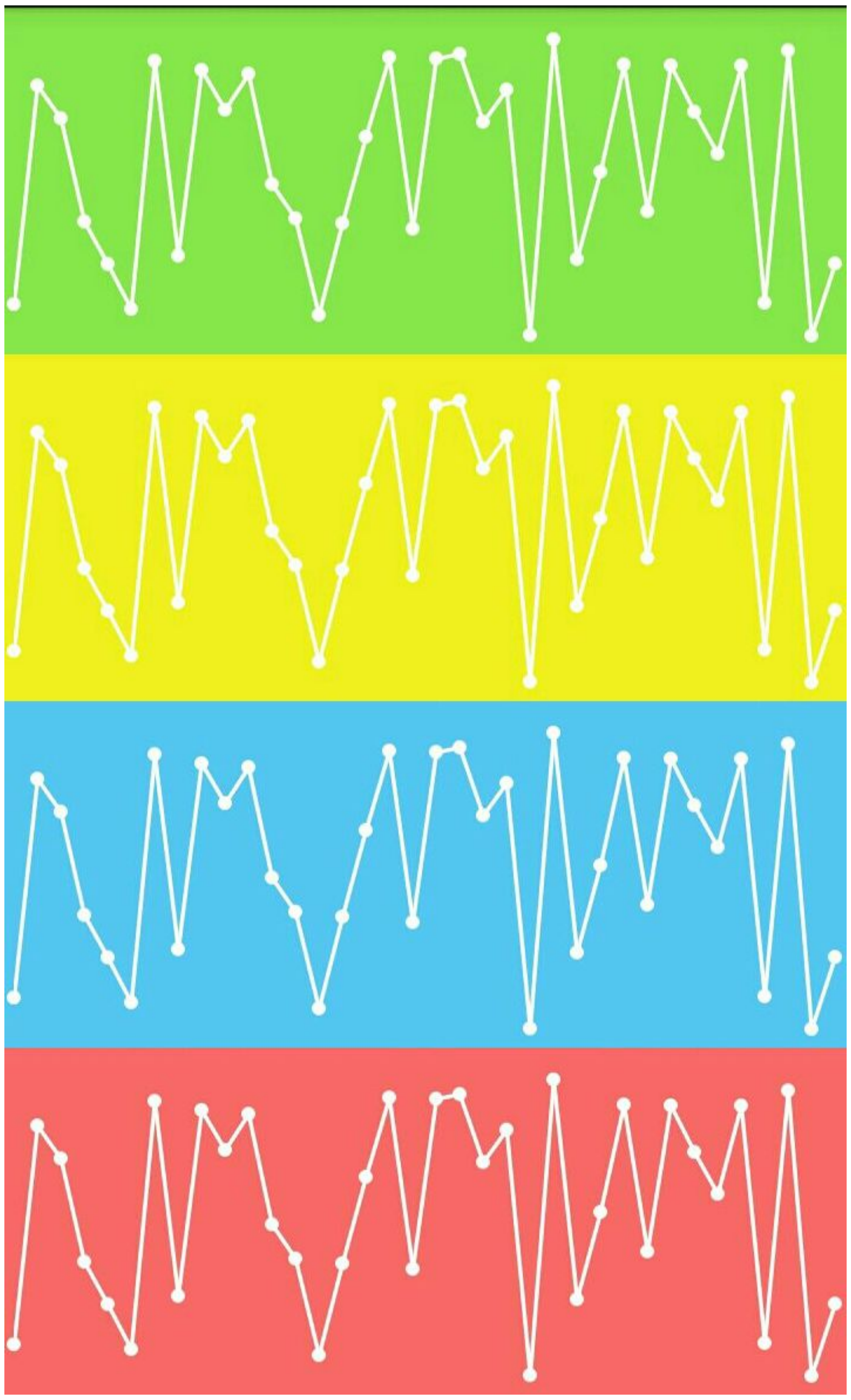

Figure 4: Mobile - Sleep graph

Sleep activities of the user will be visualised in various graphs in the mobile application.

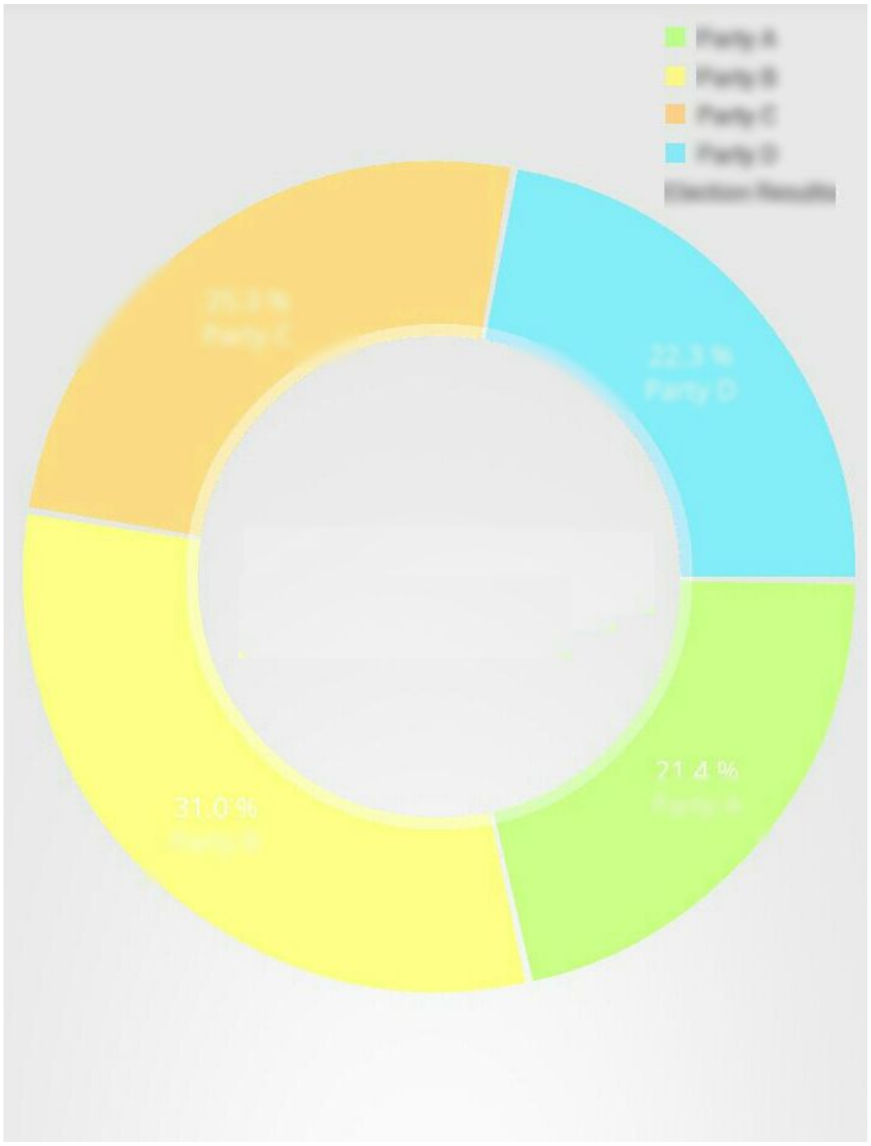

Figure 5: Mobile - Sleep State Distribution

Distribution of sleep states the user experienced will be shown as a pie chart.

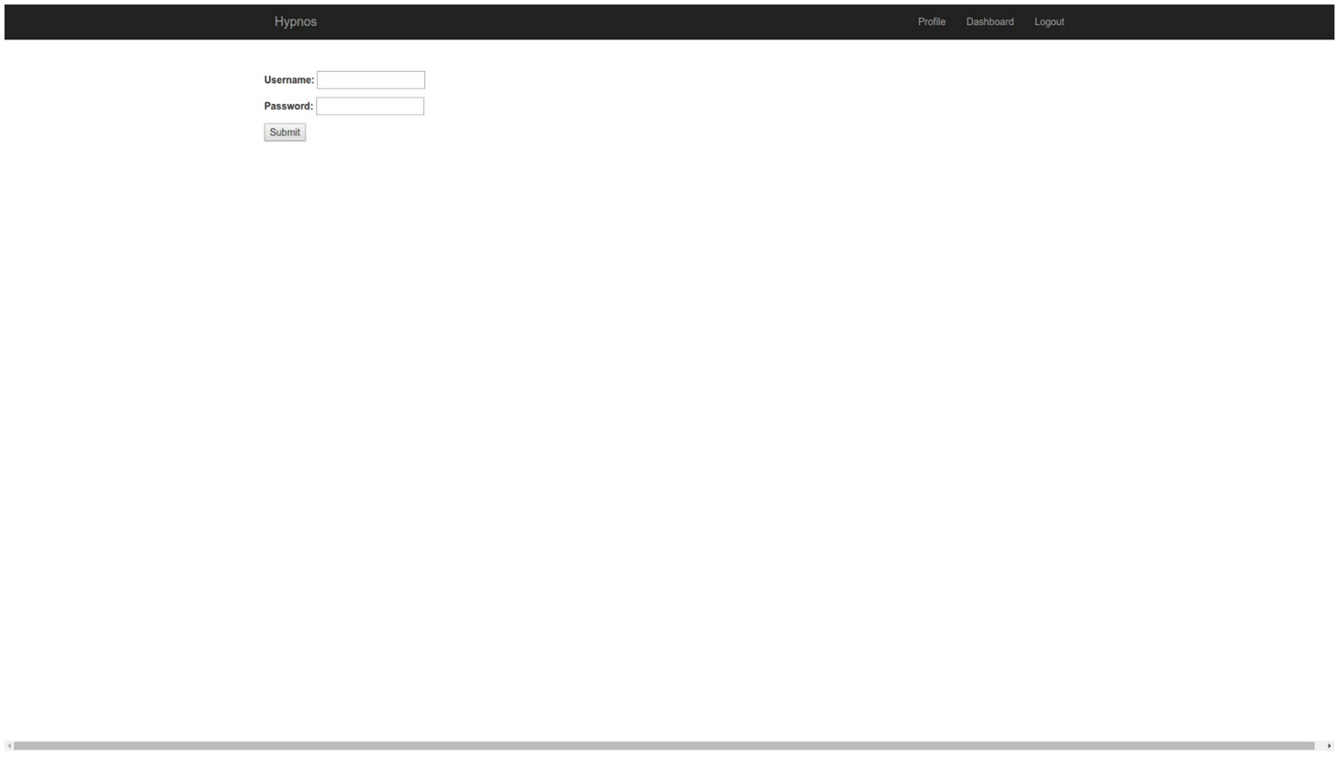

Figure 6: Web - Sign in

Users have to sign in to the system to access their data

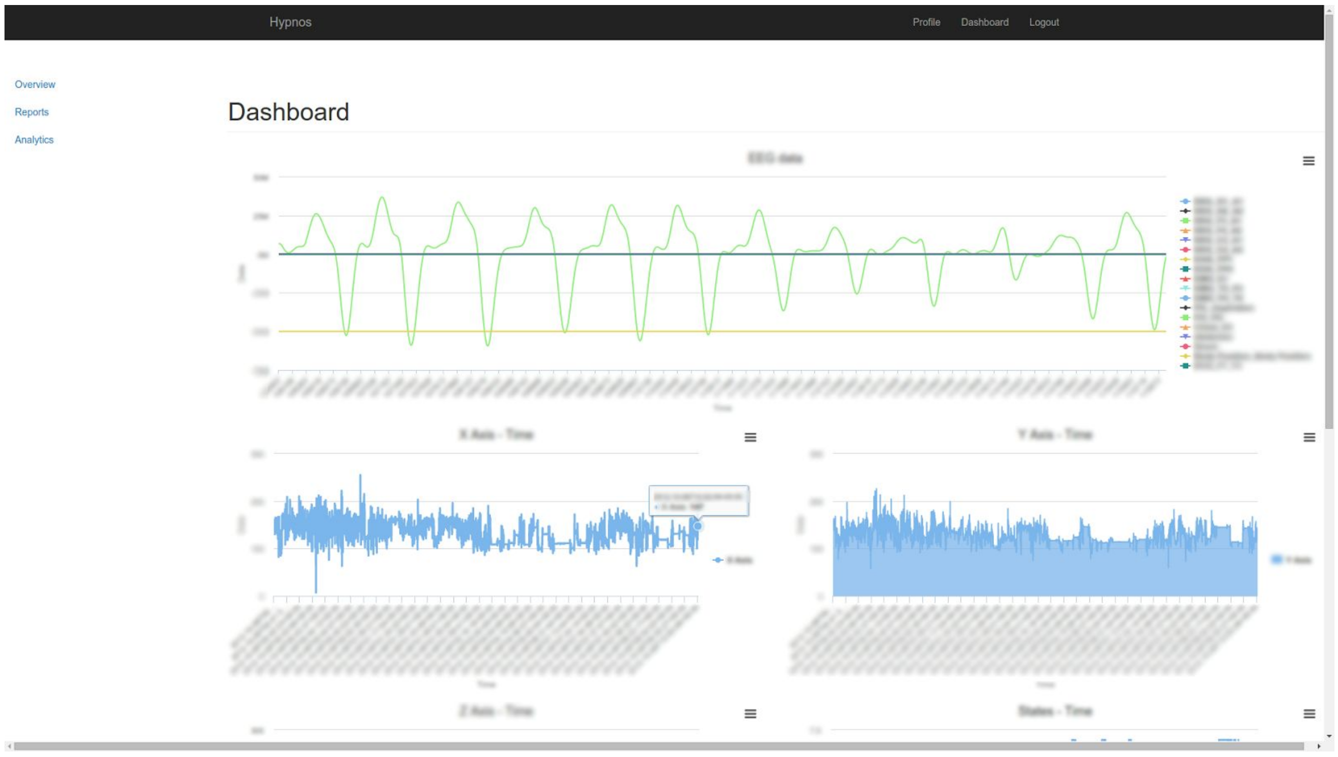

Figure 7: Web - Dashboard

When user logins to the system, dashboard is shown by default. In dashboard, general information and recent data of the user will be displayed.

2.2.2 Hardware Interfaces

Our hardware interfaces are wearable and fixed devices, which carry different sensors.

Wearable component consists of an accelerometer, infrared thermometer, pulse sensor, XBee radio module, battery and a microcontroller unit. Battery is needed for supplying voltage for sensors to be able to work.

Fixed device consists of a microphone, thermometer, XBee radio module and an Intel Galileo on it. Data of sensors on wearable is transferred through XBee radio modules between wearable and fixed components. Intel Galileo is used to store, process all the data and communicate with the cloud. Sensors are connected to Galileo via I2C and analog pins.

Wearable device sensors:

- XYZ-axis accelerometer is a low-g sensor with analog voltage outputs and adjustable sensitivity (±1.5g or ±6g).
- Pulse sensor works with 5V and its output should be connected to an analog input pin.
- Infrared thermometer is used to measure body temperature. Its communication protocol is I2C which is digital I/O.

Fixed device sensors:

- Microphone is used to get ambient sound. It gives analog output.
- Thermistor is used to get ambient temperature. Its output should be connected to an analog input pin. 1 °C changes 10 mV.
-

2.2.3 Software Interfaces

	Software Product	**Source**
Client on internet	Web Browser	
Cloud OS	Ubuntu 14.04	http://www.ubuntu.com/
Web Server	Django 1.9	https://www.djangoproject.com/
Database Server	PostgreSQL 9.4	http://www.postgresql.org/
Development End	HTML5, JS, Bootstrap, Android, MPAndroidChart, Highcharts	Latest & Stable Version
Machine Learning	Python, Scikit, Numpy, Scipy, Matplot	Latest & Stable Version

Table 2 - Software Interfaces

2.2.4 Communications Interfaces

Various communication protocols will be used for different parts of the system. Communication between fixed and wearable device will be done with Zigbee protocol.

All sensor data in wearable device will be concatenated and sent as a string A + ”-” + T + “-” P + “-” (A: Accelerometer, T: Temp, P: Pulse) to the fixed device via Xbee. Fixed device will parse the string and get the sensor values as integer values.

Communication between fixed device and cloud server will be done with JSON. Fixed device will send the data through Wi-Fi. TCP/IP standards will be used for packet transmission as personal data should be kept securely.

JSON data will consist of sensor values retrieved from wearable device, sensor values of the temperature and sound sensors on the fixed device and timestamp of when the sensor values were retrieved. Cloud will parse this JSON upon retrieving, and save it to the database with corresponding user ID.

Mobile applications and website will be sending requests to cloud. Firstly, requested data will be converted to JSON. Then,

- JSON data will be fed to dynamic HTML in backend(Django) and it will be parsed by Highcharts, the HTML page including the Highcharts graph will be sent to the user as an HTTP GET response.(Web app)
- JSON data will be sent as an HTTP GET response. After getting the response, mobile app will format this JSON and feed it to MPAndroidChart which will create the graph of the data.(Mobile app)

Emergency contacts will be notified by email, phone call and SMS.

2.3 Constraints

2.3.1 Constraint 1

Since sensor data will be sent to fixed device and cloud server, due to transmission delays, it will not be possible to consistently keep the delay under 1-2 seconds.

2.3.2 Constraint 2

Sensors used in the project will limit accuracy. For example, temperature precision will be 0.5 degrees.

2.3.3 Constraint 3

Availability of all components must not be under 95%.

2.3.4 Constraint 4

All software languages and libraries must be open source software.

2.3.5 Constraint 5

User should not see any irrelevant data on the mobile component.

2.3.6 Constraint 6

User must be able to see all the data in real time.

3. Specific requirements

3.1 Functional Requirements

3.1.1 Tracking Functionality Requirements

3.1.1.1 Functional Requirement 1

Pulse,body temperature and body movement data of user will be taken by wearable device and sent to fixed device.

3.1.1.2 Functional Requirement 2

Room temperature and sound will be taken by fixed device.

3.1.1.3 Functional Requirement 3

Every morning user’s feedback on how well they have slept will be taken by mobile device and sent to cloud.

3.1.1.4 Functional Requirement 4

User should be able to enable/disable wearable device.

3.1.1.5 Functional Requirement 5

Wearable device should run on batteries and energy consumption should be at minimum.

3.1.1.6 Functional Requirement 6

User will be woken up at their lightest sleep in an interval that they specified.

3.1.2 Analyzing Functionality Requirements

3.1.1.1 Functional Requirement 1

All sensor data coming to fixed device will be processed in real time.

3.1.1.2 Functional Requirement 2

Extracted information from the sensor data will be sent to cloud via internet from time to time.

3.1.1.3 Functional Requirement 3

Cloud will store the user sleep data in the database.

3.1.1.4 Functional Requirement 4

User should be able to enable/disable fixed device.

3.1.1.5 Functional Requirement 5

When there is no internet connection available, fixed device should continue processing the data offline and sent them to cloud as internet becomes available.

3.1.3 Optimizing Functionality Requirements

3.1.1.1 Functional Requirement 1

User will be able to see their daily sleep in charts every morning on mobile device.

3.1.1.2 Functional Requirement 2

User will be able to see their recommended optimal sleeping environment on mobile device.

3.1.1.3 Functional Requirement 3

User will be able to see overall sleeping experience on mobile device.

3.1.1.4 Functional Requirement 4

On website, user will be able to see all data more explicitly.

3.2 Non-functional Requirements

3.2.1 Usability

Since user interfaces of the project does not have complex tasks on them, user should have confidence using those in no more than an hour.
Usability requirements of the system are:

- Wearable device should be comfortable to sleep with.
- Necessary configurations of fixed device should be automatic not to confuse user.
- Charts on the mobile should be simplistic and clear.
- Questions on the user feedback form should be little, short and clear. (Not more than 3-5 questions, questions should be at most two lines etc.)
- Website and mobile application interfaces should provide a uniform look and feel between all of the pages.

3.2.2 Reliability

- Cloud component in our system should be available at least 99.5% of the time since it should always answer requests from website and mobile phones.
- Mean time to repair the cloud server should be less than a day. System should have a failover mechanism to make the server working. (Load should be balanced between more than one servers.)
- Since we are tracking extreme situations and they can be observable in just a few seconds, wearable component should be working 99.90 % of the time. Fixed device should also be available working 99.90% of the sleep duration for the same reason.

- Wearable and fixed devices should be working all along the sleep. In the circumstances that the fixed device cannot send data to server, fixed device should cache it and send when available.

3.2.3 Performance

- Wearable device should be able to send all sensor data binded to it in less than a second to the fixed device. Wearable device should be keeping up with the sensor data flow speed.
- Fixed device should be able to receive data, save to database and send periodically to cloud server.
- In the extreme situations, fixed device response should be less than 3 seconds. If there is no extreme situation, data could be stored on fixed device and can be sent to cloud later.
- Number of transactions per second can be up to 100.
- The number of customers system can hold will be 10000.
- Fixed device should have a processor capable of applying exhaustive operations.
- Fixed device will not need great memory as only one users' data will be saved and analyzed. Cloud server will need memory linearly depending on user number.
- On the fixed device, there is not a large disk space need since data will be sent to cloud with chunks and there will be one user using it. On the cloud, however there could be vast storage need. As user number gets larger, disk storage need will linearly increase.

3.2.4 Supportability

- Codes will be written according to GNU coding standards.

- Naming convention will be as follows:
  - Python codes in our project will have UpperCamelCase for class names, CAPITALIZED_WITH_UNDERSCORES for constants, and lowercase_separated_by_underscores for other names.
  - Java and C codes will have CamelCase naming convention.

- Class libraries that will be used contain but is not limited to are machine learning libraries for python, visualization libraries for python and android. Libraries can be extended for future needs.

4 Data Model and Description

Class diagrams are as follows:

4.1 Fixed Component Class Diagram

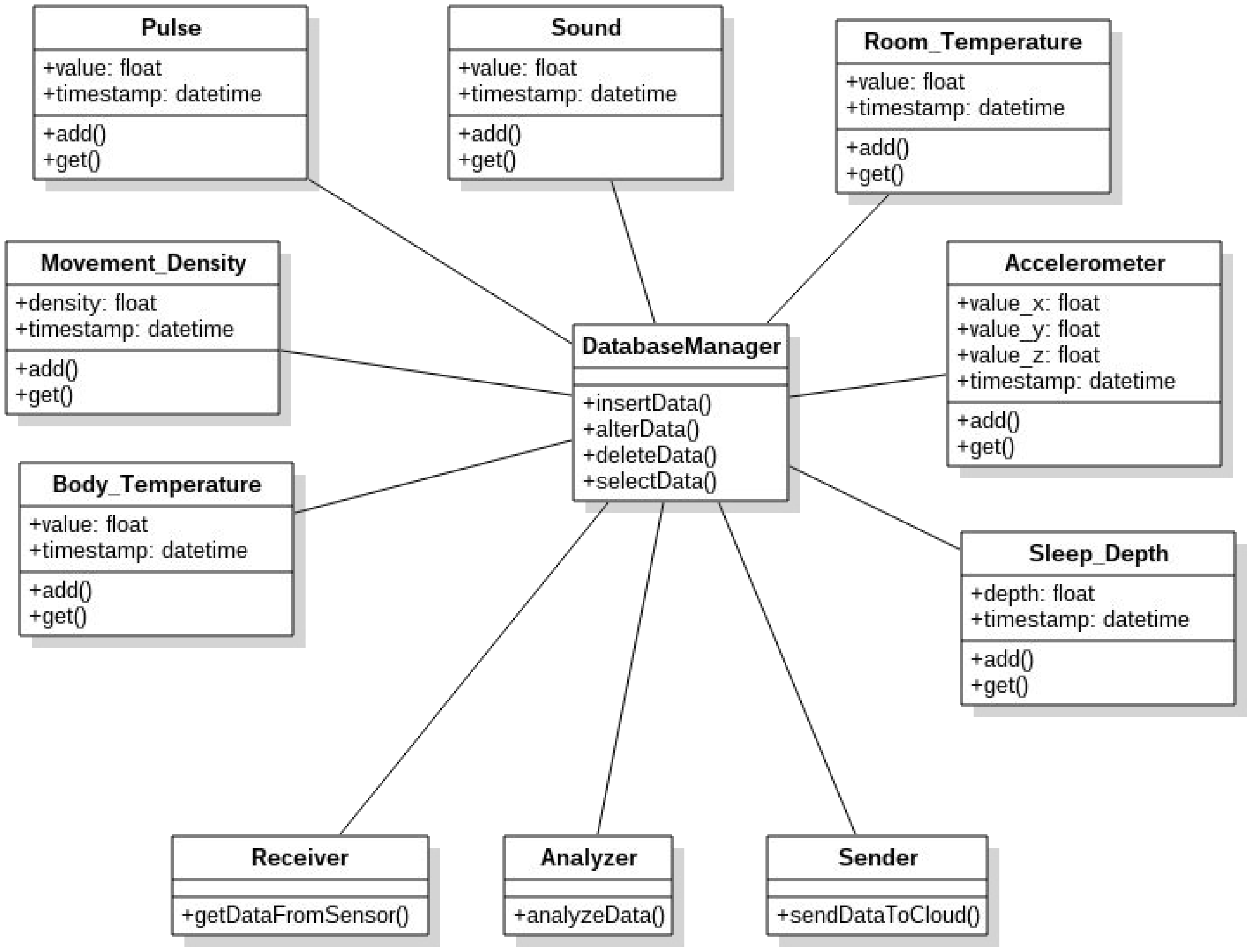

Figure 8 - Fixed Component Class Diagram

Class	Description
DatabaseManager	It is responsible for data transactions **Methods:** ● **insertData:** inserts new sensor data into database ● **alterData:** alters a sensor data from database ● **deleteData:** deletes a sensor data from database ● **selectData:** returns a sensor data from database
Receiver	It is responsible for getting the sensor data **Methods:** ● **getDataFromSensor:** gets data from a sensor
Analyzer	It is responsible for analyzing the sensor data **Methods:** ● **analyzeData:** analyzes the sensor data
Sender	It is responsible for sending sensor data to cloud **Methods:** ● **sendDataToCloud:** sends the sensor data to cloud
Pulse	It represents the pulse values coming from pulse sensor **Attributes:** ● **value:** pulse value ● **timestamp:** the time of pulse value **Methods:** ● **add:** adds a new pulse value with timestamp ● **get:** returns a pulse value
Sound	It represents the digitalized sound values coming from microphone **Attributes:** ● **value:** sound value ● **timestamp:** the time of sound value **Methods:** ● **add:** adds a new sound value with timestamp ● **get:** returns a sound value

Room_Temperature	It represents the room temperature values coming from thermometer **Attributes:** ● **value:** room temperature value ● **timestamp:** the time of temperature value **Methods:** ● **add:** adds a new temperature value with timestamp ● **get:** returns a temperature value
Body_Temperature	It represents the body temperature values coming from thermometer **Attributes:** ● **value:** body temperature value ● **timestamp:** the time of temperature value **Methods:** ● **add:** adds a new temperature value with timestamp ● **get:** returns a temperature value
Accelerometer	It represents the x,y,z acceleration values coming from accelerometer sensor **Attributes:** ● **value_x:** x axis acceleration value ● **value_y:** y axis acceleration value ● **value_z:** z axis acceleration value ● **timestamp:** the time of acceleration values **Methods:** ● **add:** adds a new accelerometer value with timestamp ● **get:** returns a accelerometer value
Movement_Density	It represents the movement density values calculated using accelerometer values of each timestamp **Attributes:** ● **density:** density of movement at timestamp ● **timestamp:** the time **Methods:** ● **add:** adds a new density value ● **get:** returns a density value

Sleep_Depth	It represents the sleep depth values calculated using all sensor values of each timestamp **Attributes:** ● **density:** density of movement at timestamp ● **timestamp:** the time **Methods:** ● **add:** adds a new density value ● **get:** returns a density value

4.2 Cloud Component Class Diagram

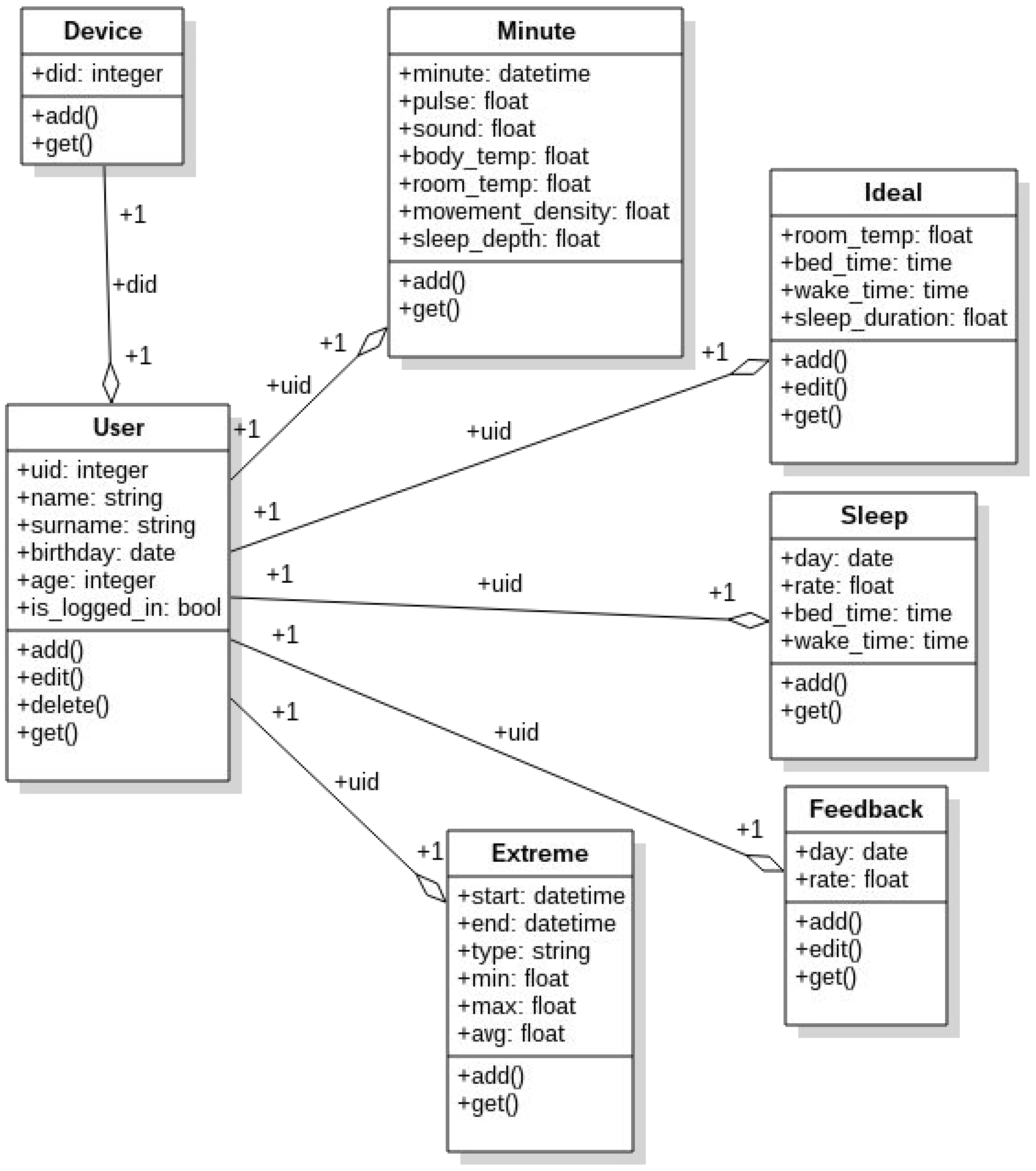

Figure 9 - Cloud Component Class Diagram

Class	Description
Device	It represents the fixed device ids which are in use **Attributes:** ● **did:** device id **Methods:** ● **add:** adds a new device id
Minute	It represents the average values of all sensors at 1 minute intervals. **Attributes:** ● **uid:** id of the user ● **minute:** time of average sensor values ● **pulse:** pulse sensor value ● **sound:** sensor value ● **body_temp:** body temperature value ● **room_temp:** room temperature value ● **movement_density:** density of acceleration values ● **sleep_depth:** sleep depth score of user **Methods:** ● **add:** adds new minute values ● **get:** gets minute values
Ideal	It represents the ideal sleeping environment for a user **Attributes:** ● **uid:** id of the user ● **room_temp:** room temperature value ● **bed_time:** time to go to bed ● **wake_time:** time to wake up ● **sleep_duration:** sleep duration **Methods:** ● **add:** adds new ideal values ● **edit:** edits the ideal values ● **get:** gets ideal values

User	It represents the user **Attributes:** ● **uid:** id of the user ● **name:** name of the user ● **surname:** surname of the user ● **birthday:** birthday of the user ● **age:** age of the user ● **is_logged_in:** value states whether the user is logged in **Methods:** ● **add:** adds user ● **edit:** edits the user ● **delete:** deletes the user ● **get:** gets the user
Sleep	It represents a sleep of a user **Attributes:** ● **uid:** id of the user ● **day:** day of sleep ● **rate:** calculated sleep rate ● **bed_time:** the time user went to bed ● **wake_time:** the time user woke up **Methods:** ● **add:** adds new sleep ● **get:** gets the sleep
Extreme	It represents the extreme cases that sensors values got **Attributes:** ● **uid:** id of the user ● **start:** start time of the extreme case ● **end:** end time of the extreme case ● **type:** sensor name ● **min:** minimum value the sensor got ● **max:** maximum value the sensor got ● **avg:** average value the sensor got **Methods:** ● **add:** adds an extreme case ● **get:** get an extreme case

Feedback	It represents the feedback of the user about their sleep **Attributes:** ● **uid:** id of the user ● **day:** day of sleep ● **rate:** rate of user to their sleep **Methods:** ● **add:** adds a feedback ● **edit:** edits a feedback ● **get:** gets a feedback

4.3 Mobile Component Class Diagram

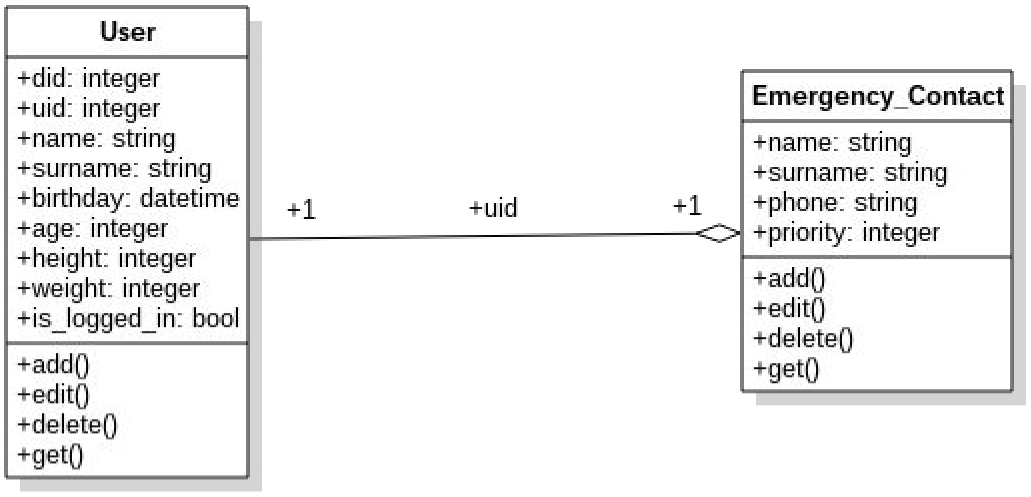

Figure 10 - Mobile Component Class Diagram

Class	**Description**
User	It represents the fixed device ids which are in use **Attributes:** ● **did:** device id ● **uid:** user id ● **name:** name of the user ● **surname:** surname of the usee ● **birthday:** birthday of the user ● **age:** age of the user ● **height:** height if the user ● **weight:** weight of the user ● **is_logged_in:** value states whether the user is logged in **Methods:** ● **add:** adds a new user ● **edit:** edits the user ● **delete:** deletes the user ● **get:** gets the user
Emergency_Contact	It represents the emergency contact person **Attributes:** ● **uid:** id of the user ● **name:** name of the contact person ● **surname:** surname of the contact person ● **phone:** emergency contact number ● **priority:** priority value of the contact person **Methods:** ● **add:** adds a new emergency contact ● **edit:** edits the emergency contact ● **delete:** deletes the emergency contact ● **get:** gets the emergency contact

About this template

This template was adapted by Emre Akbas from two sources: the IEEE 830 [1] and the ``Modern SRS package” [2].

5 References

[1] IEEE Guide for Software Requirements Specifications," in IEEE Std 830-1984 , vol., no., pp.1-26, Feb. 10 1984, doi: 10.1109/IEEESTD.1984.119205,
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=278253&isnumber=6883

[2] Appendix C of Don Widrig, Dean Leffingwell, “Managing Software Requirements: A Unified Approach,” Addison-Wesley Professional, Release Date: October 1999, ISBN: 0201615932.

[3] Marko Borazio, Eugen Berlin, Nagihan Kücükyildiz, Philipp M Scholl and Kristof Van Laerhoven, "Towards Benchmarked Sleep Detection with Inertial Wrist-worn Sensing Units", ICHI 2014, Verona, Italy, IEEE Press, 09/2014.

[4] Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), URL: http://standards.ieee.org/getieee802/download/802.15.4d-2009.pdf