System Requirements Specifications document (SRS)

for

Enhancing Indoor IOT Network Design to support energy Saving

Abdelrahman Samir, Fady Khaled, Rana Muhammed, Samar Saeid.
Supervised by: Dr. Mohamed Elgazzar, Eng. Silvia Soliman

October 2018

1 Introduction

1.1 Purpose of this document

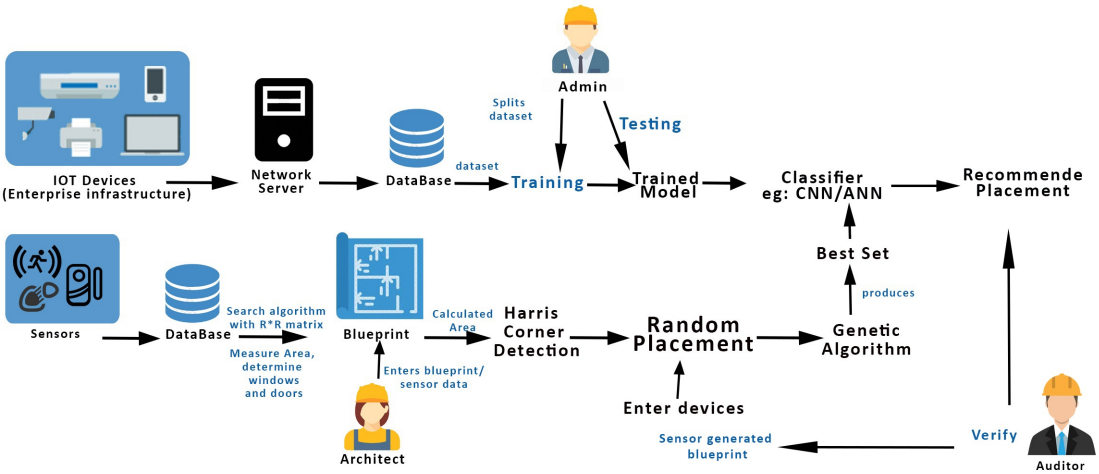
This is an SRS (software requirement specification) document to determine the set of functional and nonfunctional requirements for graduation project regarding saving power through optimizing the positioning IOT devices.

1.2 Scope of this document

This document is meant for apartments or buildings owners who use multiple IOT devices , m. it would also be beneficial to other developers whom might work on expanding this system later, as it will form a good base, ,the development environment for this system includes a simulation environment in addition to the wifi based part of the system, the estimated cost of developing the system will involve nothing but the IOT devices itself.

1.3 Overview

This project is meant to help in placement of IOT devices inside buildings, it will be used by an apartment or building owners who aim at reducing power consumption, while getting the most optimal throughput from the devices
Figure 1: System Overview



1.4 Business Context

Although we don't have a client, they system could be used by many companies.if we take Vodafone for example, they are a telecommunications conglomerate. where they hire people to manually -and through the use of other systems- position Wifi in the optimum placement to get the strongest coverage.

2 General Description

2.1 Product Functions

2.2 IoT Device Module

Add device.
Remove device.
Edit Device Data.

2.3 Network Server Module

Receives readings from devices.
Store Reading in Database.

2.4 User Module

Add user.
Delete user.
Modify user data.

2.5 Blueprint Module

Create blueprint.
Upload blueprint.
Modify blueprint.

2.6 IoT device Verification Module

Verify data sent from IoT devices.
Create data set from verified data.

2.7 Classifier Module

Models classified according to their average power consumption.

2.8 Sensors Module

Add sensor.
Remove sensor.
Get Sensor readings.

Delete Reading.

2.9 Sensor Reading Verification Module

Verify readings received from sensors.
Create floorplan from readings.

2.10 Placement Recommendation Module

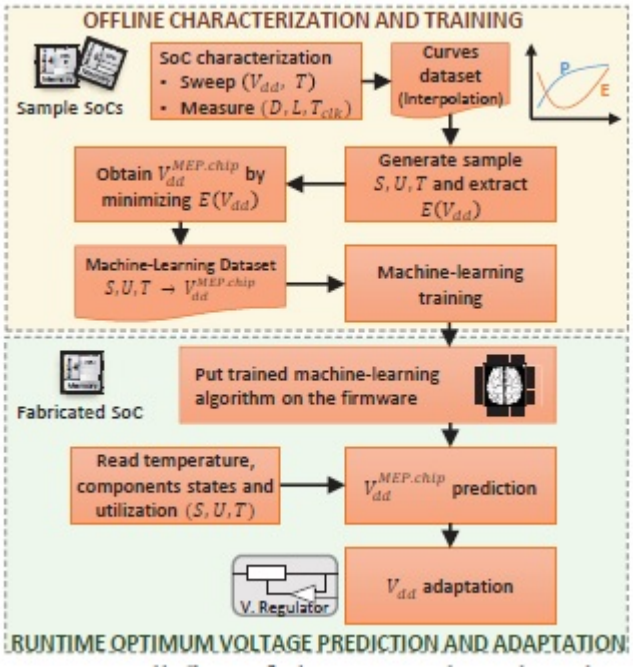
Randomize devices on blueprint.
Get Best Set of placements.
Input best set to classifier.
Generate Recommended Placement.

2.11 Similar System Information

2.11.1 Runtime Adjustment of IoT System-on-Chips for Minimum Energy Operation

The system proposes a run time machine learning technique for achieving the best voltage tuning for the IoT SoCs.The system takes into consideration the runtime parameters and their effect, such as: operating temperature, powergating states of the SoC internal components, their utilization, as well as performance requirements of the running application.
The system goes as follow, 1st: characterization of samples of the SoC from the same process corner , training the dataset based on characterized data, ML algorithm is trained.2nd: The trained ML algorithm is put on the "rmware of the fabricated chips. The optimum supply voltage is calculated by the ML algorithm during runtime, according to the characteristics. Afterwards, the chip is adapted to that voltage by tuning the voltage regulator.

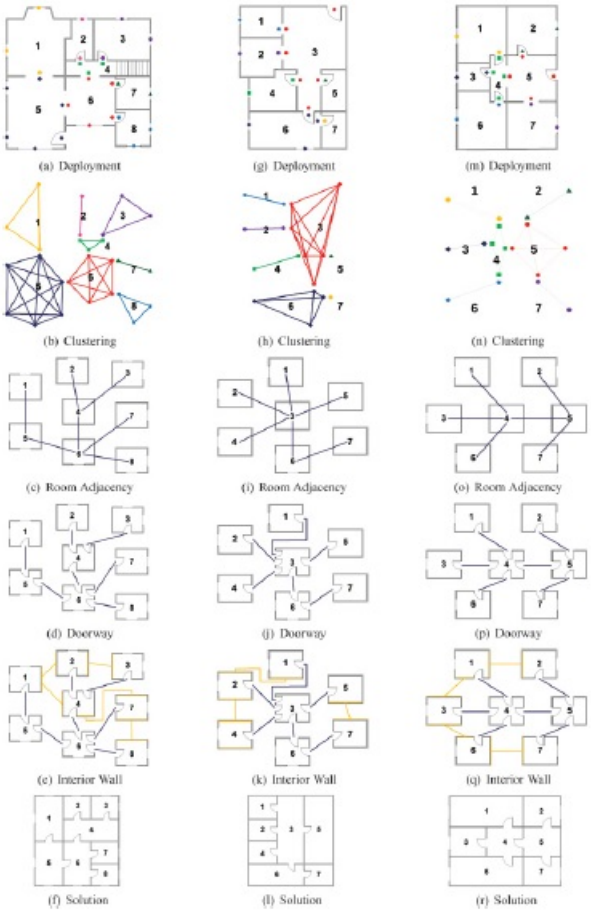
Figure 2: Flow of the system



2.11.2 Smart Blueprints

This system uses already existing sensors in smart homes to accurately give a blueprint of the apartment. It uses light sensors to measure the angle and intensity of sun light at a given time to detect the orientation of the room. They are also used on walls to determine windows, with the help of magnetometers placed on doorways. Magnetometers also give orientation of doors. Motion sensors are used to measure the room, any two motion sensors within the same enclosure are guaranteed to be monitoring different rooms so they won't cross room boundaries.
The algorithm works as follow: (1) sensors are clustered together based on similar activity patterns, (2) these clusters are converted into rooms and adjacency between rooms, (3) doorways are assigned to specific walls (N, S, E, W), (4) interior walls for room pairs are defined to be adjacent, (5) spatial layout of the floor plan.

Figure 3: Flow of algorithm



2.12 User Characteristics

The user is expected to be one of the engineers or workers implementing the IOT devices. the users shouldn't need much training on how to use the system.

2.13 User Problem Statement

The proposed System Will help the user use IOT devices more efficiently while saving unnecessary power consumed through bad positioning of IOT devices. It will help them use the devices to their most optimum throughput.

2.14 General Constraints

Password and user name will be needed to log in.

3 Functional Requirements

3.0.1 Function Requirement 1

Name: Login
Critically: High
Input: Username, Password
Output: Redirect to the system
Description: The user input his/her User name and password to check them in the database
Priority: 10/10
Expected risk: Database size, bugs, wrong User name or password
Preconditions: User is not logged in yet
Post-condition: User is logged in and redirect to the System
Dependencies: none

3.0.2 Function Requirement 2

Name: Add new Readings
Critically: High
Input: Readings from devices
Output: CSV File
Description: Adding new Readings from Iot devices and pass it to ANN for making analysis
Priority: 10/10
Expected risk: Data corrupted, null values
Preconditions: sensors to generate data
Post-condition: Fill session with the data
Dependencies: F.01

3.0.3 Function Requirement 3

Name: Add User
Critically: High
Input: User name ,password
Output: New user is recorded in the Database
Description: Admin registers to add user
Priority: 10/10
Expected risk: Database bugs, username duplicate
Preconditions: Data of user should be available

Post-condition: User data is added to Database
Dependencies: none

3.0.4 Function Requirement 4

Name: Edit User
Critically: Medium
Input: ID of user ,username
Output: User information can be Edited or not
Description: Admin or user itself Edit User
Priority: 8/10
Expected risk: corrupted data, wrong username, wrong ID
Preconditions: Data of user should be available
Post-condition: Updated Data and recorded in Database
Dependencies: F.01

3.0.5 Function Requirement 5

Name: Delete User
Critically: Medium
Input: ID of user,username
Output: User can be deleted or not
Description: Admin or user itself Delete user
Priority: 7/10
Expected risk: corrupted data, wrong username
Preconditions: Data of user should be available
Post-condition: User Data deleted from Database
Dependencies: F.01

3.0.6 Function Requirement 6

Name: Verify Results
Critically: High
Input: Trained Module
Output: Data is tested if it is Valid or not
Description: Auditor verify the result
Priority: 10/10
Expected risk: Corrupted results, data loss
Preconditions: Module is trained and expected good results
Post-condition: Module is verified on the system or not
Dependencies: F.01, F,07

3.0.7 Function Requirement 7

Name: Upload Data set
Critically: High
Input: CSV File
Output: Data can be uploaded to a web server
Description: Architect uploads the dataset to the web server
Priority: 10/10
Expected risk: Network error , big dataset size
Preconditions: Sensors are available and data generated
Post-condition: Data set is available on web server
Dependencies: F.01, F.02

3.0.8 Function Requirement 8

Name: Upload Blueprint
Critically: High
Input: Blueprint image (jpg)
Output: Blueprint can be uploaded to the server
Description: Architect uploads the Blueprint to the server
Priority: 10/10
Expected risk: Network error
Preconditions: Sensors are available and data generated
Post-condition: Blueprint is available on web server
Dependencies: F.01, F.02

3.0.9 Function Requirement 9

Name: Upload Data from sensors

Critically: High
Input: CSV File from Light sensors, Motion sensors, Magnetometer
Output: Data can be uploaded to a web server
Description: Architect uploads the dataset
Priority: 10/10
Expected risk: Wrong configuration
Preconditions: Magnetometer, Motion and light sensors are available and generate data
Post-condition: Blueprint is available with specifications
Dependencies: F.01, F.02

3.0.10 Function Requirement 10

Name: Edit data from sensors
Critically: Medium
Input: CSV File from Light sensors, Motion sensors, Magnetometer
Output: Data can be updated to a web server
Description: Architect edits the data from sensors
Priority: 9/10
Expected risk: Wrong configuration
Preconditions: Magnetometer, Motion and light sensors are available and generate data
Post-condition: Data is updated and available on the server with blue print
Dependencies: F.01 ,F.02

3.0.11 Function Requirement 11

Name: Delete data from sensors
Critically: Medium
Input: CSV File from Light sensors, Motion sensors, Magnetometer
Output: Data can be deleted or not from server
Description: Architect deletes the data from sensor
Priority: 7/10
Expected risk: Wrong configuration
Preconditions: Magnetometer, Motion and light sensors are available and generate data
Post-condition: Data is deleted from server,sensor
Dependencies: F.01,F.09

3.0.12 Function Requirement 12

Name: Analyze Features
Critically: High
Input: CSV File
Output: Dataset of device data with their clustered classes
Description: Analyze data from trained model and output data set
Priority: 10/10
Expected risk: Database size, bugs, wrong calculations
Preconditions: Data set without analyze or cluster
Post-condition: Data set is analyzed and identified into classes
Dependencies: F.01, F.02, F.07,F.09

3.0.13 Function Requirement 13

Name: Verify Blueprints
Critically: High
Input: Blueprint specification, blueprint image
Output: Data is valid or not
Description: Auditor verifies blueprint and check the specifications
Priority: 10/10
Expected risk: Database size, bugs, wrong calculations, weak tests
Preconditions: Blueprint
Post-condition: verified blue print and record the Architecture in Database
Dependencies: F.01, F.02,F.14

3.0.14 Function Requirement 14

Name: Retrieve blueprint's specifications
Critically: High
Input: Blueprint
Output: Length, width, height, room, floor
Description: the system collects the specifications of such a blueprint like (width, height, length) to pass them to a drawing function
Priority: 10/10
Expected risk: Database size, bugs, wrong calculations, weak tests
Preconditions:
Post-condition: Blueprint Specifications
Dependencies: F.01,F.02,F.08,F.09

3.0.15 Function Requirement 15

Name: upload filtering module
Critically: Low
Input: filtering module
Output: Data can be updated and sent to web server
Description: the Architect setups the amount of training and testing data that he can also edit
Priority: 7/10
Expected risk: Database size, bugs, wrong calculations, weak tests
Preconditions: old data with a predefined amount of training and testing
Post-condition: Data is updated in the filtering module
Dependencies: F.01,F.02,F.03,F09,F.14

3.0.16 Function Requirement 16

Name: Get Readings from file
Critically: High
Input: Simulation model with IOT devices
Output: File containing device's features passed to ANN
Description: Retrieving Features from device like (Average Power Consumption) to ANN
Priority: 10/10
Expected risk: The values of the data can be NULL
Preconditions: Simulate IOT devices
Post-condition: Fill the file with the data
Dependencies: none

3.0.17 Function Requirement 17

Name: build (draw) Blueprint from sensors
Critically: High
Input: CSV file from sensors
Output: Blueprint
Description: Identify positions of door, windows and walls in blueprint using light, motion sensors and Magnetometer
Priority: 10/10
Expected risk: wrong configuration, wrong calculations
Preconditions: Sensors' readings
Post-condition: approximated image of blueprint
Dependencies: F.01, F.09, F.14

3.0.18 Function Requirement 18

Name: logout
Critically: Medium
Input: user interaction (button)
Output: Redirect to the system
Description: user clicks logout so the system is no longer available with its login features and getting out of session
Priority: 7/10
Expected risk: Database size, bugs, wrong calculations
Preconditions: User is logged in on the system with system's features
Post-condition: User is logged out and no longer feature existence
Dependencies: F.01, F.03
subsubsectionFunction Requirement 19 Name: addObserver
Critically: Medium
Input: observer or role to be updates and notified for Results
Output: Redirect to the system

Description: this functions works to add observer in the cycle of observer design pattern
Priority: 7/10
Expected risk: Database size, bugs
Preconditions: User is logged in on the system with system's features
Post-condition: User is updated for result models
Dependencies: F.01, F.02,F.03,F.06,F.08
subsubsectionFunction Requirement 20 Name: RemoveObserver
Critically: Medium
Input: observer or role is deleted from the observers array and may not be notified for Results
Output: Redirect to the system
Description: this functions works to remove observer from the cycle of observer design pattern
Priority: 7/10

Expected risk: Database size, bugs
Preconditions: User is logged in on the system with system's featuresand updated or notified
Post-condition: User is logged in and not updated for result models
Dependencies: F.01, F.02,F.03,F.06,F.08
subsubsectionFunction Requirement 21 Name: notifyObservers
Critically: Medium
Input: observer or role is notified for Results
Output: Redirect to the system
Description: this functions works to notify observer of the cycle of observer design pattern
Priority: 7/10
Expected risk: Database size, bugs
Preconditions: User is logged in on the system with system's features and updated or notified
Post-condition: User gets notified
Dependencies: F.01, F.02,F.03,F.06,F.08

4 Other non-functional attributes

4.1 Security

The Username and Password should be encrypted and the data transmitted to database should be saved securely.

4.2 Maintainability

The system can be improved by different developers so its maintainability should be easy by just changing small changes not the whole Architecture. And also by implementing different design patters such as MVC.

4.3 Portability

The system has the ability to run on a variety of operating systems (windows, Linux, IOS) and it does not depend on a particular type of hardware. So it can be deployed on multiple platforms.

4.4 Reliability

The reliability that when the system goes down and the data is lost, there's a backup storage stored in a Database then we can restore it back from other device. Also the web application run in multiple of servers and if the data is lost from specific server, we can restore it back from other server.

5 Interface Requirement

The system interface is a web application accessed by any web browser. through a log in panel users will then proceed to choosing the number of training sets and testing sets.User will also view building architecture provided by Wifi signals in the building.

5.1 User Interface

5.1.1 GUI

Figure 4: Login GUI

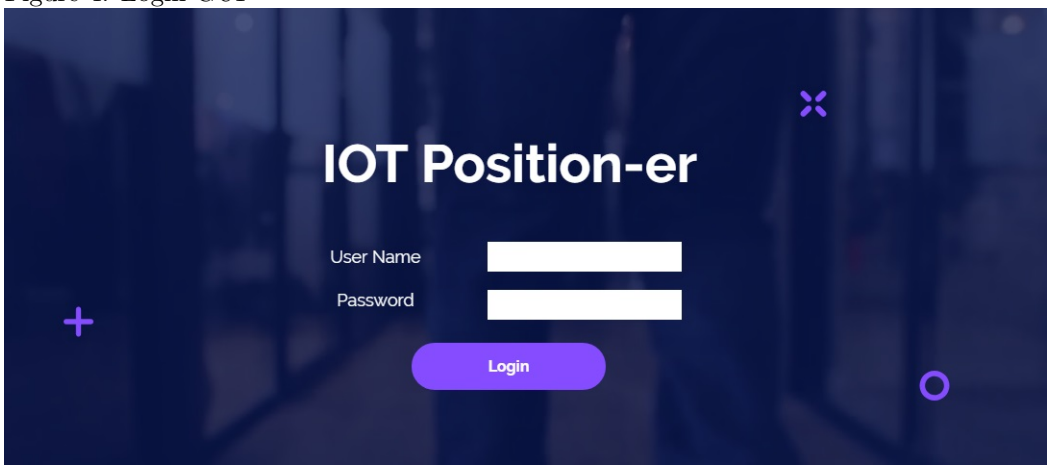


Figure 5: Testing GUI

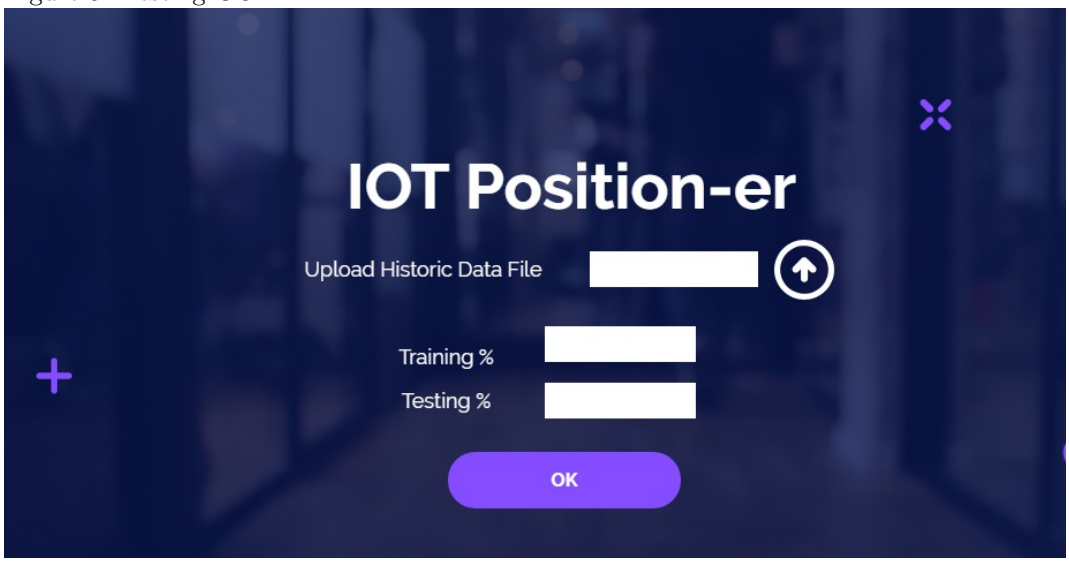


Figure 6: Uploading for Result

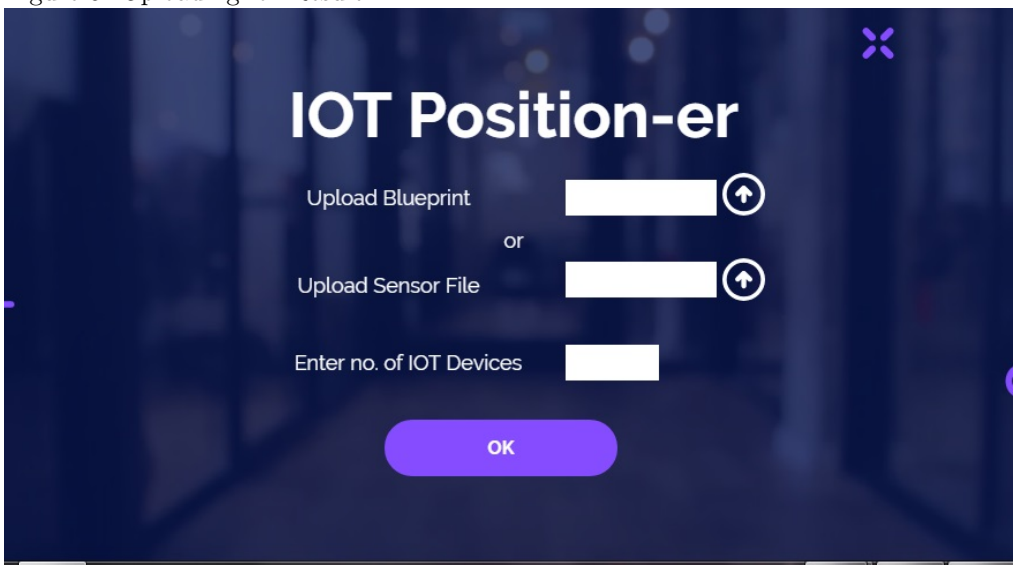
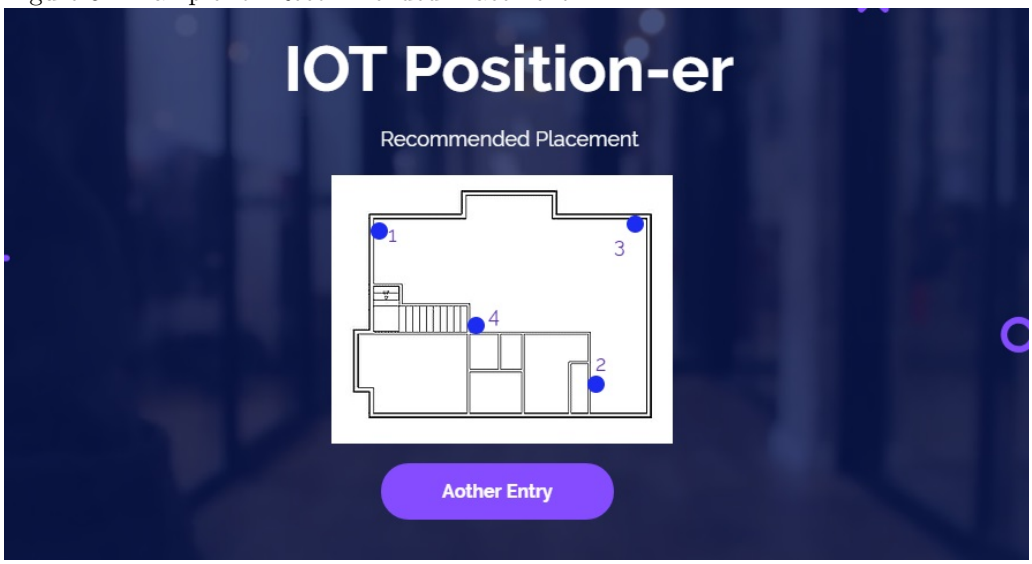


Figure 6: Example for Recommended Placement



5.1.2 CLI

N/A

5.1.3 API

N/A

5.1.4 Diagnostics or ROM

N/A

5.2 Hardware Interfaces

In case of using simulator there will be no hardware. if using real devices and sensors. Then Multiple IOT devices will be used along with Light, Motion, Magnetometer sensors.

5.3 Communications Interfaces

N/A

5.4 Software Interfaces

6 Performance Requirements

7 Design Constraints

7.1 Standards Compliance

The web application part of the system is compatible with all browsers. It's also compatible with 64 bit processors.

7.1.1 Hardware Limitations

N/A

8 Other Non-Functional Attributes

8.1 Security

The Username and Password should be Encrypted and The data transmitted to database should be saved securely.

8.2 Maintainability

The system can be improved by different developers so its maintainability should be easy by documenting the code and the design and by implementing different design patters as Pipe and filter.

8.3 Portability

The system has the ability to run on a variety of operating systems (windows, Linux, IOS) and it does not depend on a particular type of hardware. So It can be deployed on multiple platforms.

8.4 Reliability

The reliability that when the system goes down and the data is lost, There's a backup storage stored in a Database then we can restore it back from other device.

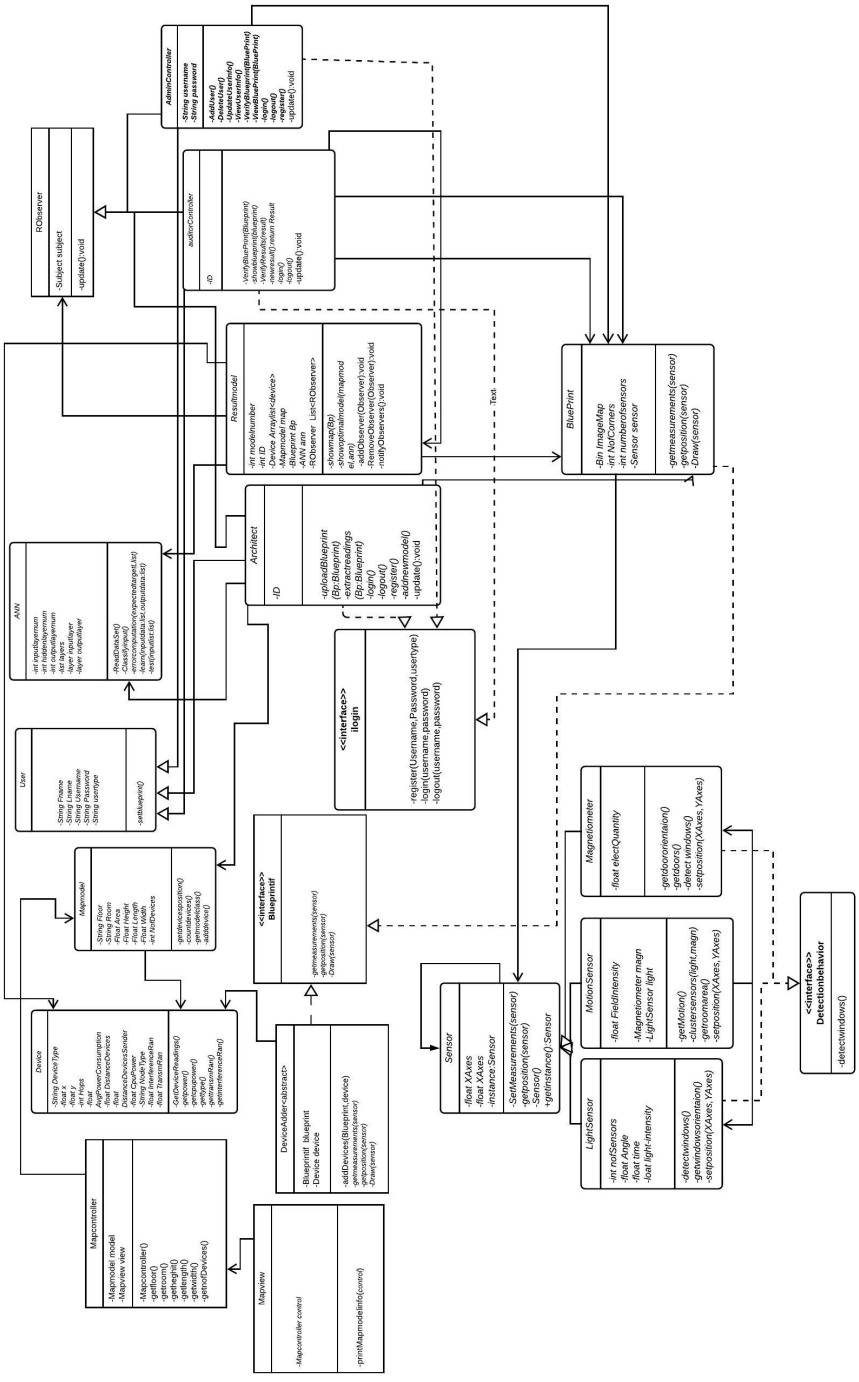
9 Preliminary Object-Oriented Domain Analysis

9.1 Inheritance Relationships

Class Architect inherits from class user Class auditorController inherits from class user Class AdminController inherits from class user Class MotionSensor inherits from class Sensor Class LightSensor inherits from class Sensor Class Magnetiometer inherits from class Sensor Class Resultmodel inherits from class RObserver Class auditorController inherits from class RObserver Class AdminController inherits from class RObserver

9.2 Class description

Figure 6: Class diagram



Class Diagram.jpeg

9.2.1

- **Class name:**User, Concrete.
- **List of super classes:** None.
- **List of sub classes:** Architect, AdminController amp; auditorController.
- **Purpose :** To define the basic attributes and methods of user such as his username and password and the login method and register method.
- **Collaboration:** To achieve its purpose User class should perform login process using username and password to ensure security. It will need to have the classes Architect, AdminController and AuditorController to inherit from it.
- **Attributes:** Fname,Lname,username,usertype and password will be string.
- **Operation:**
  setblueprint() it's argument: username(string),password(string),blueprint(jpg)

9.2.2

- **Class name:**ANN , Concrete.
- **List of super classes:** None.
- **List of sub classes:** None.
- **Purpose :** To define the basic attributes and methods of the artificial neural network such as number of input layers and number of output layers and and number of hidden layers and a list of layers and readdataset,classifyinput,error computation,learn and test methods.
- **Collaboration:** To achieve its purpose Device's class readings are passed by the Architect class to Ann for further analysis as the mentioned methods. It will need to have inheritance relationship with Architect class.

- **Operation:**
  ReadDataset() it's argument: Dataset(csv), ClassifyInput() it's argument: model(Resultmodel), errorComputation() it's argument: expectedTarget(List), Learn() it's argument: inputdata(list),outputdata(list) Test() it's argument: inputlist(list)

9.2.3

- **Class name:**Device , Concrete.
- **List of super classes:** None.
- **List of sub classes:** None.
- **Purpose :** To define the basic attributes and methods of the positioned devices such as Device Type and Devices' positions and number of hops and the average power consumption of all devices and Distance between devices and distance between every device and the sender device (router) and cpu power consumption and node type and interference range and transmission range and getDeviceReadings, getpower, getcpupower,gettype,gettransRan and getinterferenceRan methods.
- **Collaboration:** To achieve its purpose Device's class readings are retrieved by the Resultmodel class to be verified by the auditorController for further analysis as the mentioned methods. It will need to be aggregated by Resultmodel class and inherited from Mapmodel class.
- **Attributes:** devicetype will be string, and position,hops and avgpowerconcumption and distancebetweendevices and cpupower and nodetype and interference range and transmission range would be integers.
- **Operation:**
  GetDeviceReadings () it's argument:simulationreadings(csv),

  getpower () it's argument: simulationreadings(csv), getcpupower () it's argument: simulationreadings(csv), gettype () it's argument simulationreadings(csv), gettransmRan () it's argument simulationreadings(csv), getinterferenceRan () it's argument simulationreadings(csv)

9.2.4

- **Class name:**Mapmodel, Concrete.
- **List of super classes:** None.
- **List of sub classes:** None.
- **Purpose :** To define the basic attributes and methods of map model such as which floor and which room and area of the room and height of the room and length of room and width of the room and number of attached devices and getdevicesposition, countdevices, getmodelclass and adddevice methods.

- **Collaboration:** To achieve its purpose Map model class should specify all the attached devices in the model and their readings to make a model with a blueprint and attached devices. It will need to have inheritance relationship to Device class and needs to be inherited from Architect class and it also needs to be aggregated from Resultmodel class.
- **Attributes:** floor,room,area,height,length,width and numberofdevices will be integers.
- **Operation:**
  getdevicesposition () it's argument: Device(device), countdevices () it's argument: Device(device), getmodelclass () it's argument: simpulationreadings(int), adddevice () it's argument:None

9.2.5

- **Class name:** Architect, Concrete.
- **List of super classes:** User.
- **List of sub classes:** None.
- **Purpose :** To define the basic attributes and methods of the architect such username and password and UploadBluePrint, UploadDevData, UploadDataset, EditDataSet, DeleteDataSet,login, logout, register and addnewmodelmethods.
- **Collaboration:** To achieve its purpose Architect class it should login first to be responsible of managing the blueprints and the databases. It will need to have and association to class User and an inheritance relationship to Mapmodel class and needs to depend on the login interface class and it also needs to inherit from ANN class and an inheritance relationshop from Blueprint class.
- **Attributes:** username,password will be strings.
- **Operation:**
  buildBluePrint () it's argument:None, UploadDevData () it's argument: username(string),password(string), UploadDataset () it's argument: simpulationreadings(int), EditDataSet () it's argument: simpulationreadings(int), login () it's argument: : username(string),password(string) logout () it's argument: : username(string),password(string) register () it's argument: : username(string),password(string) addnewmodel () it's argument: None

9.2.6

- **Class name:** AdminController, Concrete.
- **List of super classes:** User.

- **List of sub classes:** None.
- **Purpose :** To define the basic attributes and methods of the Admin controller such username and password and AddUser, DeleteUser, UpdateUserInfo, ViewUserInfo, VerifyBlueprint, ViewBluePrint,login, logout and register.
- **Collaboration:** To achieve its purpose AdminController class it should login first to be responsible of managing the users and the blueprint and needs to depend on the login interface class.
- **Attributes:** username,password will be strings.
- **Operation:**
  AddUser () it's argument: username(string),password(string), DeleteUser () it's argument: username(string),password(string), UpdateUserInfo () it's argument: username(string),password(string), ViewUserInfo () it's argument: username(string),password(string), login () it's argument: : username(string),password(string) logout () it's argument: : username(string),password(string) register () it's argument: : username(string),password(string) VerifyBlueprint () it's argument: Blueprint(blueprint), ViewBluePrint () it's argument: Blueprint(blueprint)

9.2.7

- **Class name:** auditorController, Concrete.
- **List of super classes:** User.
- **List of sub classes:** None.
- **Purpose :** To define the basic attributes and methods of the auditor Controller such as username and password and showresult, VerifyBlueprint, VerifyResult,login, logout and register methods.
- **Collaboration:** To achieve its purpose auditorController class it should login first to be responsible of managing the result and the blueprint. It will need to have and association to class User and an inheritance relationship to Mapmodel class and needs to depend on the login interface class and resultmodel class and ojf it also needs to inherit from ANN class and an inheritance relationshop from Blueprint class.
- **Attributes:** username,password will be strings.
- **Operation:**
  login () it's argument: : username(string),password(string) logout () it's argument: : username(string),password(string) register () it's argument: : username(string),password(string) VerifyBlueprint () it's argument: Blueprint(blueprint),

showBluePrint () it's argument: Blueprint(blueprint), VerifyResult () it's argument: Resultmodel(resultModel),

showResult () it's argument: Resultmodel(resultModel)

9.2.8

- **Class name:** Blueprint, Concrete.
- **List of super classes:** None.
- **List of sub classes:** None.
- **Purpose :** To define the basic attributes and methods of the Blueprint such as ImageMap and number of Corners and getuserinfoand getmeasurements methods.
- **Collaboration:** To achieve its purpose Blueprint class should get an image and a number of corners . It will need to have and inheritence from class Admincontroller,Architect and auditorController.
- **Attributes:** ImageMap as a BIN, NofCornerswill be integer.
- **Operation:**
  getmeasurements () it's argument:None, getuserinfo () it's argument:classtype

9.2.9

- **Class name:** appartement, Concrete.
- **List of super classes:** User.
- **List of sub classes:** None.
- **Purpose :** To define the basic attributes and methods of the appartement such as numberofsensors ord and getmeasurements and getpositionmethods.
- **Collaboration:** To achieve its purpose appartement will need to be aggregated with sensor class to let the sensor perform their operations to scan the plase.
- **Attributes:** numberofsensors will be integer.
- **Operation:**
  getmeasurements () it's argument: Blueprint(blueprint), getposition () it's argument: Blueprint(blueprint), VerifyResult () it's argument: Sensor(Sensor)

9.2.10

- **Class name:** Sensor, Concrete.
- **List of super classes:** None.
- **List of sub classes:** MotionSensor, Magnetiometer, LightSensor.
- **Purpose :** To define the basic attributes and methods of the Sensor of the sensors such as xAxes of every position and YAxes of every position and SetMeasurements and getposition methods.
- **Collaboration:** To achieve its purpose Sensor class It will need to associated from class MotionSensor, Magnetiometer, LightSensor.
- **Attributes:** XAxes, XAxes will be floats.
- **Operation:**
  SetMeasurements () it's argument: : Sensor(sensor), getposition () it's argument:Sensor(sensor)

9.2.11

- **Class name:** LightSensor, Concrete.
- **List of super classes:** Sensor.
- **List of sub classes:** None.
- **Purpose :** To define the basic attributes and methods of the LightSensor such as number of sensors and angle and time and light intensity and X axes and Y axes and detectwindows, getwindowsorientaion, and setposition methods.
- **Collaboration:** To achieve its purpose LightSensor class It will need to have and association to class Sensor
- **Attributes:** nofSensors will be integer, Angle,time,light-intensity xAxes and yAxes will be floats.
- **Operation:**
  detectwindows () it's argument:None, getwindowsorientaion () it's argument:None, setposition () it's argument: : xAxes(float),yAxes(float)

9.2.12

- **Class name:** Magnetiometer, Concrete.
- **List of super classes:** Sensor.
- **List of sub classes:** None.

- **Purpose :** To define the basic attributes and methods of the Magnetiometer such electric quantity and X axes and Y axes and getdoororientaion, getdoors, and setposition methods.
- **Collaboration:** To achieve its purpose Magnetiometer class It will need to have and association to class Sensor
- **Attributes:** electQuantity,xAxes and yAxes will be floats.
- **Operation:**
  getdoororientaion () it's argument:None, getdoors () it's argument:None, setposition () it's argument: : xAxes(float),yAxes(float)

9.2.13

- **Class name:** MotionSensor, Concrete.
- **List of super classes:** Sensor.
- **List of sub classes:** None.
- **Purpose :** To define the basic attributes and methods of the MotionSensor such as field intensity and Magnetiometer and Light sensor and X axes and Y axes and getMotion, clustersensors, getroomarea, and setposition methods.
- **Collaboration:** To achieve its purpose MotionSensor class It will need to have and association to class Sensor also needs to aggregate both classes Lightsensor and Magnetiometer.
- **Attributes:** fieldIntensity, xAxes and yAxes will be floats and mag would be an object from class Magnetiometer and light would be an object from class LightSensor.
- **Operation:**
  getMotion () it's argument:light(lightSensor),mag(Magnetiometer), clustersensors () it's argument:Sensors, getroomarea () it's argument: None, setposition () it's argument: : xAxes(float),yAxes(float)

9.2.14

- **Class name:** Resultmodel, Concrete.
- **List of super classes:** None.
- **List of sub classes:** None.
- **Purpose :** To define the basic attributes and methods of the Resultmodel such as model number and an array of devices and a map model from map and showmap, showoptimalmodel, showdevices, and showsimread methods.

- **Collaboration:** To achieve its purpose Resultmodel class It will need to aggregate both Mapmodel and Device class.
- **Attributes:** modelnumber will be an integer, devices array would be an array and mapmodel which is an object from mapModel.
- **Operation:**
  showmap () it's argument:mapmodel(Mapmodel), showoptimalmodel () it's argument: mapmodel(Mapmodel, showdevices () it's argument: device(Device)

9.2.15

- **Class name:** ilogin, interface.
- **List of super classes:** None.
- **List of sub classes:** None.
- **Purpose :** To define the basic attributes and methods of the ilogin such as Login, Logout and register methods.
- **Collaboration:** To achieve its purpose ilogin class It will need to be depended from Architect,auditorController and AdminController classes.
- **Attributes:** None.
- **Operation:**
  Login () it's argument:username(String),password(String), Logout () it's argument: username(String),password(String), register () it's argument: username(String),password(String),usertype(int)

9.2.16

- **Class name:** Blueprintif, interface.
- **List of super classes:** None.
- **List of sub classes:** None.
- **Purpose :** To define the basic methods of the Blueprintif such as getmeasurements, getposition and Draw methods.
- **Collaboration:** To achieve its purpose Blueprintif class It will need to be depended from DeviceAdder and BluePrint classes.
- **Attributes:** None.

- **Operation:**
  getmeasurements () it's argument:sensor(Sensor), getposition () it's argument: sensor(Sensor), Draw () it's argument: sensor(Sensor)

9.2.17

- **Class name:** Detectionbehavior, interface.
- **List of super classes:** None.
- **List of sub classes:** None.
- **Purpose :** To define the basic method of the Detectionbehaviorsuch as detectwindows method.
- **Collaboration:** To achieve its purpose Detectionbehaviorclass It will need to be depended from LightSensor and Magnetiometerclasses.
- **Attributes:** None.
- **Operation:**
  detectwindows () it's argument: None

9.2.18

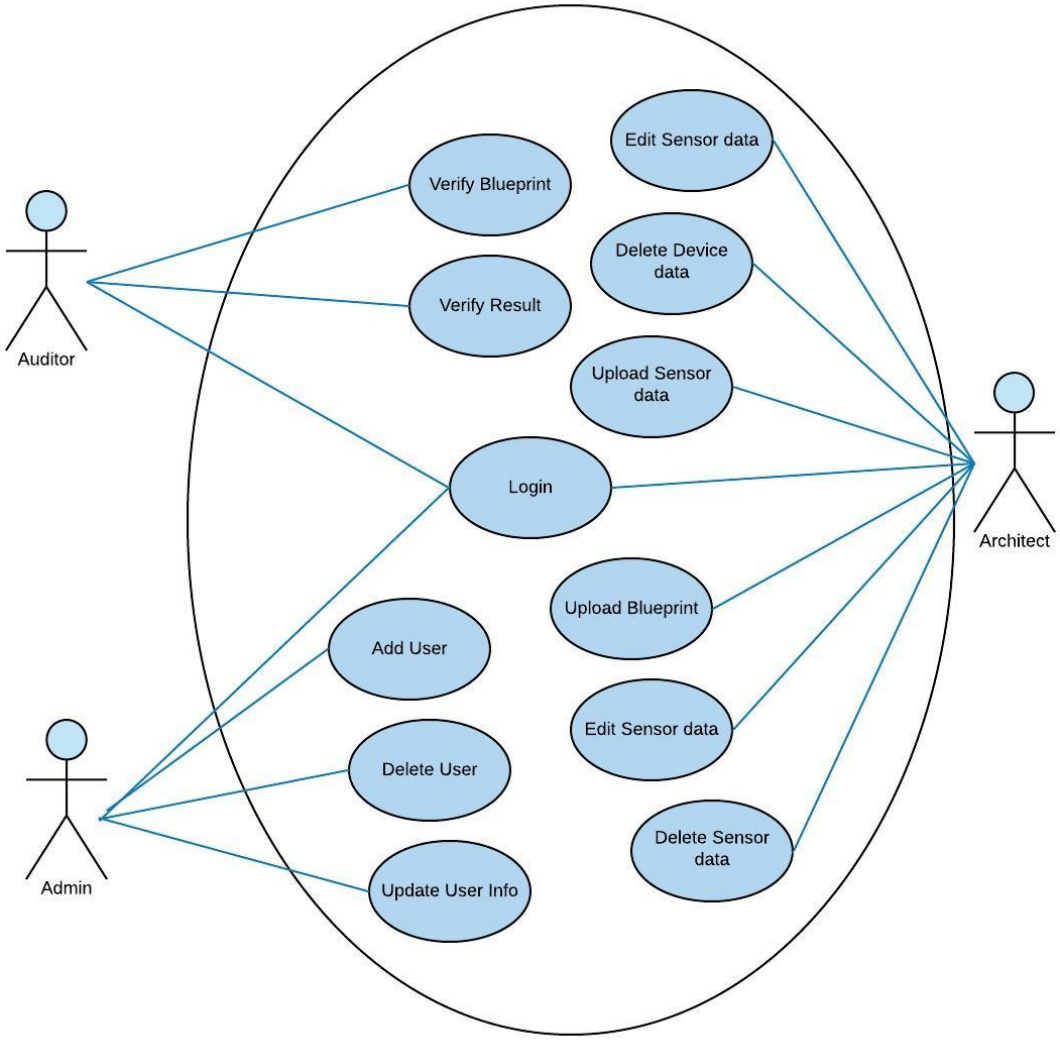
- **Class name:** DeviceAdder, Abstract.
- **List of super classes:** None.
- **List of sub classes:** None.
- **Purpose :** To define the basic attributes methods of the DeviceAdder such as blueprint, device and addDevices, getmeasurements, getposition, Draw methods.
- **Collaboration:** To achieve its purpose DeviceAdder class It will need to depended from Blueprintif interface and associated with Device classe.
- **Attributes:** None.
- **Operation:**
  getmeasurements () it's argument:sensor(Sensor), addDevices () it's argument: Blueprint(blueprint),Device(device), getposition () it's argument: sensor(Sensor), Draw () it's argument: sensor(Sensor)

9.2.19

- **Class name:** RObserver, Abstract.
- **List of super classes:** None.
- **List of sub classes:** None.
- **Purpose :** To define the basic attribute and method of the RObserver such as subject and update method.
- **Collaboration:** To achieve its purpose RObserver class It will need to be depended from auditorController, AdminController and Architectclasses.
- **Attributes:** Resultmodel will be an object from Resultmodel.
- **Operation:**
  update () it's argument: None

10 Operational Scenario

Figure 7: Use case diagram



11 Preliminary Schedule Adjusted

Figure 8: Schedule

Project Phase	Starting	Ending
Receiving proposal and ideas from Dr's and student	02-Jul-18	15-Jul-18
Announce Proposals for students	16-Jul-18	22-Jul-18
Register Students to Projects	26-Jul-18	27-Jul-18
1 Proposal Evaluation	26-Sep-18	27-Sep-18
Submitting Survery Paper	End Of Nov.	
2 SRS Evaluation	27-Nov-18	29-Nov-18
External Examiner	03-Dec-18	
SDD Evaluation	2 week of feb	
Evaluation Implementation	after spring break	
Delevring 8 Pages Paper	3 days after spring vacation	
Technical Evaluation	1st week of may	
Final Thesis	25-Jun-19	
Final Presentation	26-Jun-19	

12 Preliminary Budget Adjusted

Budget will be the money needed for The laptop LE 7000. In case of using actual IOT devices, more money will be needed, same goes for paying for web host. Light sensor LE 135, Motion sensor LE 175, Magnetometer 194

References

[1] J. Lu, Y. T. Shams, and K. Whitehouse, "Smart blueprints: how simple sensors can collaboratively map out their own locations in the home," *ACM Transactions on Sensor Networks (TOSN)*, vol. 11, no. 1, p. 19, 2014.

[2] M. S. Golanbari and M. B. Tahoori, "Runtime adjustment of iot system-on-chips for minimum energy operation," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2018.

(1) (2)