



university of
 groningen

faculty of mathematics
and natural sciences

Human-assisted Smart Environments

Bachelor's thesis

July, 2020

Author: Sabina Boranbayeva

Primary supervisor: Dr. Viktoriya Degeler

Secondary supervisor: Michel Medema

Abstract

The importance of smart environments becomes very significant nowadays. The reason is that they improve life productivity, extend independent living, and reduce caregivers' necessary time and health care costs [9]. However, the more substantial they become, the more problems arise while working with them. More developed Smart Systems require more expensive devices to use; more data to process. Consequently, it is relatively impossible to acquire all the required equipment for Smart Buildings' owners. Therefore, the problem arises that Smart Environments often lack full automation, and this leads to non-optimal states of the buildings, where these Smart Systems work. The only reasonable solution to this is to assign some people to assist the Smart Systems to help the systems to collect missing data and perform some actions they cannot do themselves. Therefore, the research question of this paper is to find an efficient approach to handle this problem and create a convenient way for people to communicate with Smart Environments. The well-developed solution to this question can encourage more people to assist Smart Systems, which will lead to a more satisfactory work of Smart buildings.

There are already multiple studies that talk about how to ease human-participation in Smart Environments and how to reduce its costs. For instance, the Runner Updation Optimization Algorithm (RUOA) that maximizes the energy-saving in Smart Buildings [13] or the decision tree that minimizes the human participation in Smart Buildings [2]. Also, there are some apps available for everyday use, such as Pilot, eNET SMART HOME, xComfort, etc., that allow people to operate the Smart System in their house. Therefore, in order to solve the research question, I propose the development of the app that will let people assist the Smart Environments in a fast and convenient way. However, this paper is not about how to find the most efficient energy state or the minimum possible human-assistance cost. My solution is more about letting people choose between those two things or find a balance between them. After users make a choice, the system asks all necessary questions to collect data it cannot get from sensors, and asks the users to perform some actions based on their preference if required. Due to this, for the app, it has to be possible to automatically request help from several people and prevent delays in assistance if someone is not available to do what smart environments require at the moment.

Contents

1	Introduction	3
1.1	Smart Environments and users' involvement	3
1.2	Motivation	4
1.3	Structure	4
2	Related work	6
2.1	Human-assistance in smart environments	6
2.2	CSPs as an approach for energy saving optimization	6
2.3	Available applications for human-assistance in Smart Environments	7
3	Problem Statement	9
3.1	Research Questions	9
3.2	Expected results	9
4	Approach	11
4.1	Requirements	11
4.1.1	Functional Requirements	11
4.1.2	Non-functional Requirements	12
4.2	App structure	13
4.3	Methods	14
5	Implementation	15
5.1	"Add a new simulated Environment" page	15
5.2	HomePage	17
5.3	"See Environments' data" page	19
5.4	Algorithm	20
5.4.1	Set Temperature Indoors	21
5.4.2	Set Lux Level Indoors	21
5.4.3	Set CO2 Level Indoors	22
6	Evaluation	23
6.1	Methodology evaluation	23
6.2	Testing algorithm	23
6.3	App Interface	28
7	Conclusion	29
7.1	Results	29
7.2	Future Work	29

Bibliography	31
A Algorithm (code)	33

Chapter 1

Introduction

1.1 Smart Environments and users' involvement

The smart environment is a concept that was coined at the end of the 20th century and has become prominent nowadays. It encompasses concepts such as smart home, city, office, etc., [10] and aims to maximize energy performance by both comforting the lives of its occupants and improving their productivity [3]. Overall, a smart environment operates as an intelligent agent that receives all necessary data using sensors such as connected light bulbs, thermostats, security cameras, etc., to achieve its goal [12]. Nevertheless, smart buildings are not able to perform every action and collect all necessary information on their own due to the lack of fully automated actuation capabilities [3]. This problem arises since the total cost of all sensors and actuators required for the most efficient work of Smart Systems can be very high. Therefore, it follows that they require human-assistance for their full operation.

There are already multiple researchers who aim to find the most-efficient energy state of Smart Buildings or to achieve the minimum human-assistance cost required. For example, according to Degeler and Curry [2], it is possible to construct the optimal decision tree with the minimum human-participation cost required. Moreover, Shuja and Javaid [13] introduce the Runner Updation Optimization Algorithm (RUOA)-which schedules the consumption pattern of residential appliances-to reduce energy use. Furthermore, even though there are some apps responsible for human-assistance in Smart Environments, such as eNET SMART HOME [4], xComfort [15], Pilot [5], etc., they work mostly as a controller, that does not find the optimal solution. The goal of these applications is to let users control their devices themselves through the app. Their inconvenience is that they do not consider actuators that do not connect to the System. Thus, people may not know what actions to perform at the moment to find the state of the environment they want.

Therefore, in this paper, I do not discuss ways to minimize the human-assistance cost or reduce energy conservation in Smart Systems. My goal is to develop an app for smartphones that considers everything that can impact energy usage and users' comfort in the building. It has to help users to make their own

decisions and, based on them, assist the Smart Environments by answering required questions and performing specific actions. These actions are determined by using certain predefined behavior rules that are established based on specific conditions that I introduce later throughout this paper. By the end of this research, and after I built my application, I have to answer the following question: Is the smartphone app an easy and convenient way for people to interact with Smart Systems?

1.2 Motivation

There is a good quote from NABERS: "A sensor can tell you what the temperature is, but only a person can tell you whether that is the right temperature for them." These words closely describe the problem I aim to solve. Since the existing solutions are mostly about the data collected from sensors the System gets, it does not always count as the real users' opinion. For example, the rules can state that 22 degrees in the room is a perfect temperature for the user, while in reality the user might find it a bit too hot for them. In contrast, for some people, 15 degrees is considered cold, but the System can ignore it because there is no constraint stating that. Therefore, it is important to consider users' opinion, especially if they need a comfortable environment, and not an energy-efficient state. Therefore, if there is a possibility to develop an app that considers the user's opinion and consent to important decisions made by Smart Systems, it will lead to more accurate solutions on what actions are required to perform.

Moreover, the app for human-assistance, which is the aim of this project, is beneficial for owners of smart buildings, when it comes to a financial matter. Because the app is for public access, everyone will be able to assist the Smart System. Therefore, there is no need to hire new employees to work with it.

1.3 Structure

In addition to the Introduction, this paper consists of 8 more chapters: Related work, Problem Statement, Approach, Implementation, Evaluation, Conclusion, and Future Work.

In the **Related work** Chapter I describe the field in general, and how other researchers have tried to solve my research problem.

The **Problem Statement** Chapter gives a detailed explanation of the problem I am trying to solve.

The **Approach** Chapter describes my approach to solve the research problem.

In the **Implementation** Chapter I demonstrate how I implemented my approach.

In the **Evaluation** Chapter I evaluate the approach I chose and answer the question of whether it was successful or not. Moreover, I provide the reasoning behind my conclusion.

Furthermore, I provide a small summary of what was discussed in the paper to conclude the results I got in the **Conclusion** Chapter.

Finally, I discuss what future possible extensions to my app I can add to improve it in the **Future Work** Chapter.

Chapter 2

Related work

2.1 Human-assistance in smart environments

The **Introduction** Chapter presented Smart Environments and their aim. Therefore, it is clear that their System usually works in two ways: first, it finds the most energy-efficient state of the building, and, second, it comforts the lives of its occupants and increases their productivity. However, it is hard for Smart Environments to process the changes made by users during their interaction with the environment. Moreover, newly installed sensors and devices can disturb the productivity of the dynamic environment itself [3]. Therefore, even though Smart Systems can implement many tasks automatically, it still heavily depends on the human. There are two reasons behind it. First, smart environments cannot perform every action on their own. Second, sometimes there are no sensors that can collect necessary data. Thus, Smart Systems require human-assistance to obtain lacking information and perform actions that they cannot do automatically [2]. Therefore, the problem is to find a convenient way for people to communicate with Smart Environments and assist them; also, to make these systems as much independent from the user as possible, but at the same time keeping them energy-saving if required. Consequently, to understand how to find a solution to this problem, I have to comprehend how smart environments work first.

2.2 CSPs as an approach for energy saving optimization

In this chapter, I explain more about why Smart Environments need human-assistance. When people discuss how Smart environments work, the first thing that comes to mind is logic. The representation of decision logic in these systems is hard-coded if-then rules [3]. However, in [2, 3] authors argue that there are some disadvantages to the hard-coded rules approach. These disadvantages appear in two cases: when there is a need to maintain the integrity of the rules with a large number of sensors and devices, or when it is necessary to update the system. Thus, in [2] and [3], the authors conclude that it is better and more efficient to use constraints instead of hard-coded if-then rules. This approach

allows the Smart System to select one of the possible actions to perform based on the given constraints [2, 3]. More studies advise to use the same method. For example, Pecora and Cesta [11] describe one of the similar approaches in their work. Their research is about multi-agent coordination in Smart Homes. The simulation of this kind of coordination is a distributed constraint optimization problem (COP) meaning that intelligent agents depend only on communication with other agents [3, 11].

Therefore, after the System models a problem as a constraint satisfaction problem (CSP), it turns all given rules and constraints into a Disjunctive Normal Form (DNF). In this form, every conjunction has at least one possible alternative to lead to the satisfaction of all given environment's rules. Then, the System sends the resulting DNF to the Situation Manager Module that collects all necessary data and creates questions for users to fulfill the gaps. After the Situation Manager finishes its work, it sends everything it has obtained to the Decision Tree Manager, which then creates a decision tree with all questions and actions it receives. In this way, the Tree Manager follows the decision tree, asks relevant questions to users, and makes sure that all given environment's rules are satisfied [2, 3]. Therefore, with this approach, it is possible to create a decision tree that requires minimum user's efforts. However, is there another way to minimize human participation in Smart Environments, or is it possible to let people choose what they can do at the moment? Consequently, the problem that remains unsolved is to find another way to reduce human-assistance cost. My approach to solving this is not to reduce every human assistance cost but rather let users decide whether they want to assist the Smart Environment at the moment or not. Thus, it will not make people feel uncomfortable doing what the System pushes them to. Also, because I want to let multiple people assist one Smart Environment, the total human participation cost for one person should be divided into a number of all users who assisted this Environment. Therefore, this approach can minimize the one human-participation cost, and at the same time, not make users feel tired.

2.3 Available applications for human-assistance in Smart Environments

As I stated in the **Introduction** Chapter, there are multiple applications for human-assistance available. However, in this section, I discuss only 3 of them: Pilot [5], eNet SMART HOME [4], and xComfort [15]. All three apps work as a controller of all, connected to them, devices. Users can choose suitable for them the brightness or the warmth of the room. Also, they can control the window's shades, and the HVAC system with it. As I understand, these apps do not consider devices that do not connect to them. Thus, for example, if users do not have a thermostat connected to the System in their house, they cannot control it. As a result, there will not be any possibility to change the room temperature or get the advice for the optimal temperature at the moment. These applications provide the opportunity to users to choose everything themselves. Moreover, for energy-efficiency, there is functionality, that shows how much energy every device connected to the system consumes. Thus, people

can turn off the device if it requires too much energy. What differentiates my app from the apps mentioned above is that people can choose in what state of the environment they want it to be. Then based on their choice, the System provides the list of actions recommended to perform to obtain the desired result. Also, it considers all devices, even the ones not connected to the app.

Chapter 3

Problem Statement

3.1 Research Questions

In the **Introduction**, there was a statement that Smart Environments is not a very novel idea but prominent. For centuries people always consider various ways to ease their lives, and Smart System is one of their ways to comfort their everyday routine. But how to make it even more convenient to use if not all possibilities for that are optimized? As a result of constant technological progress and the evolved role of smartphones as an inseparable part of our lives, developers think about the ways they can connect Smart Systems of ways to connect Smart Systems to them. Even though it appears to be a useful tool, there are still issues that arise, despite many solutions to avoid them. Therefore, the questions I try to answer in this project are:

1. There are multiple solutions to reduce energy consumption and minimize human-assistance costs. But can people find a balance between them?
2. Since sensors' data is not the best basis for the Smart System's rules, then what is a better substitute?
3. How to provide a way for the user to communicate their opinion about the environment's condition to the system?
4. How to minimize human participation without considering the cost for one action?
5. Is it possible to involve several people to assist the smart environment? How to indicate who is available to help at the moment?
6. Is a mobile-application an efficient way to assist Smart Environments? Should developers consider other devices?

3.2 Expected results

As I mentioned above, the approach, I assumed to be the most sufficient to answer these questions, was to create an app for human-assistance in smart environments. In this application, I wanted to apply an algorithm that goes

through all relevant constraints, rules, and questions. If the application's system is not able to answer these questions itself, then it has to ask them to users. I did not want to build the algorithm in a way to find the most efficient solution for energy conservation or minimum human-assistance costs only. It is supposed to consider the users' choice of what is more important to them. Therefore, this app has to collect available data and ask people questions if it can not answer them automatically. Furthermore, it should ask users to perform actions based on the collected information. So, after answering the research questions stated in the previous section and successfully building the application, I expected to get to the following contributions:

1. A solution that leads to the balance between saving energy and minimization of human assistance required.
2. A functionality to ask users their opinions about different states of the environment they are in.
3. A real optimal solution based on human interests and the most energy-efficient state possible.
4. An application that will work with several people at the same time.
5. An optimization system that can find and recommend a set of actions that have to be performed by people to achieve the goal state of the environment they want.

Overall, the goal of this specific project is to build a communication bridge between Smart Systems and people. It must provide a set of actions that users have to perform to achieve the desired room's condition. If the result of this project succeeds, the society will benefit in ways: people will have less work, they will be able to operate the Smart System from anywhere, and people will not waste energy, i.e., there will be a reduction of energy consumption.

Chapter 4

Approach

4.1 Requirements

As I stated in the previous chapter, the goal of the project is to build the app that provides an easy way to assist a Smart Building System to the user. With this app, users can fill in blanks with data, answer questions, and get instructions with actions to perform to help the System. The app is supposed to minimize a human-participation cost by asking different people whether they are available to assist or not at the moment. If they are free, they have a choice to assist the System or skip their turn and ask another user. It also has to be able to collect data from sensors and apply specific rules, which I predefined based on the information I collected. I explain it later in the **Implementation** Chapter.

4.1.1 Functional Requirements

There have to be **3** pages in the app:

- Homepage
- "Add a new simulated Environment"
- "See environments' data"

Homepage

- The **System** must be able to:
 1. Ask specific questions to users to fill in empty data cell in a table
 2. Store new data
 3. Run the algorithm to find what actions users have to perform
 4. Ask users to perform actions based on the result of running the algorithm
- The **Users** must be able to:
 1. Choose the environment they want to work with
 2. Add their names to the chosen environment

3. Answer questions asked by the **System**
4. Skip their turn to answer
5. Get instructions of what to do

"Add a new simulated Environment" page

This page should consist of empty blanks such as name, address (country, city), the temperature inside/outside, humidity indoor/outdoor, etc. It is required to create different simulated environments to test the algorithm with different situations.

- The **System** must be able to:
 1. Store data for a simulated Environment
 2. Create a new simulated Environment
- The **Users** must be able to:
 1. Fill in blanks with new information
 2. Choose to what extent comfort is more prioritized than energy conservation.

"See environments' data" page

- The **System** must be able to:
 1. Retrieve data of simulated environments
- The **Users** must be able to:
 1. See data of simulated environments

4.1.2 Non-functional Requirements

1. Should perform well with different **operating systems**
2. **User friendly design.** It should be easy to navigate through the application for everyone
3. The application should provide **ease of access to information.** Accessing and navigating through the page should be easy and intuitive.
4. The website should provide a **stable connection.** It must be available throughout the whole day during normal circumstances; this means the system can sometimes be down for maintenance.

4.2 App structure

This section demonstrates the representation of all components of the app and their interactions with each other.

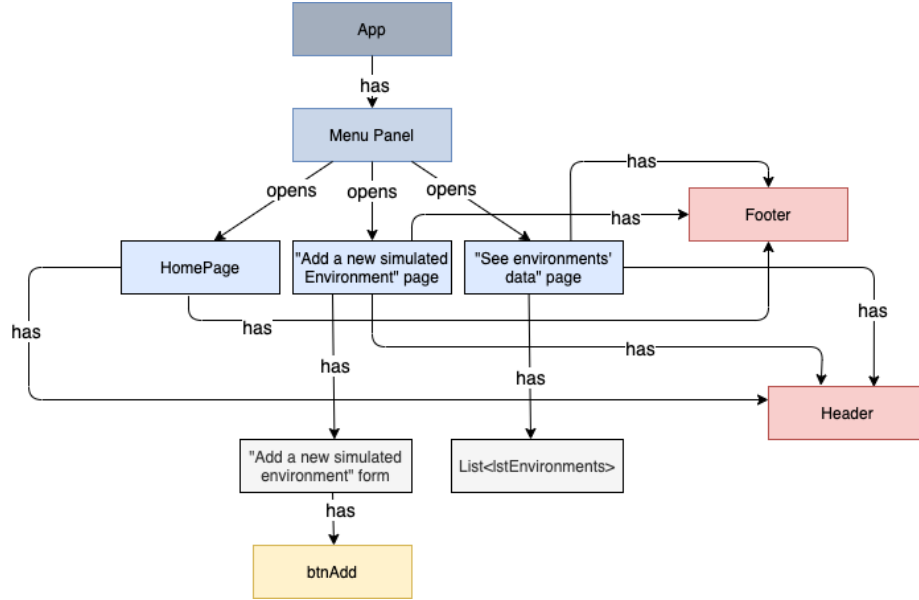


Figure 1. The diagram to represent the interactions of different parts of the app

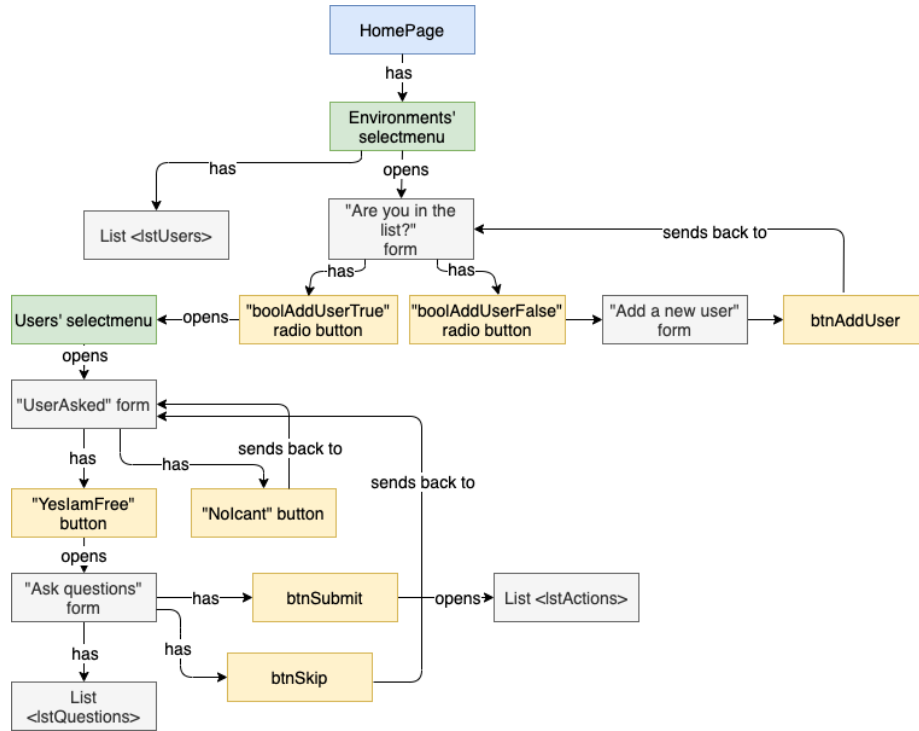


Figure 2. The diagram to represent the interactions of different parts of the app

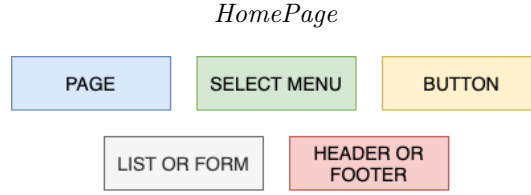


Figure 3. Colors decision

Figure 1 shows the general structure of the app, while figure 2 extends it and provides the diagram that consists of all components of the HomePage. Figure 3 indicates what colors are for what components.

4.3 Methods

To reach the goal stated above, I had to consider technologies, such as Algorithm Design, Software Architecture, Operating Systems, etc. The programming language that I used was JavaScript, particularly the Apache Cordova (PhoneGap) framework. PhoneGap is a mobile application development framework. Therefore, because the application is supposed to run on mobile, I find Apache Cordova convenient to use. Its advantage is being intuitive and straightforward. Also, it did not require a big budget and any additional knowledge except what I have already studied. For the front-end, I used languages HTML and CSS together with JavaScript. For API I decided to work with Node.js and SQLite. Node.js is another JavaScript framework, while SQLite is good to work with data. I made these choices based on the existing libraries that would make work easier to implement. To test the app every time I added new functionality, I used Android OS. I chose that because it was the best at working with Apache Cordova.

I studied most of the previous research papers and their results discussed in this paper. Also, I have tested all created sets of rules, actions, and questions several times. Moreover, I chose Stackoverflow and GitHub to get answers if I did not understand something. However, I also asked help to supervisors who I met during the arranged video call meetings. For my project, I did not have a lot of sensors and actuators, so I wrote the algorithm using hard-coded if-then statements. The reason was that it was faster to implement, and I had to finish the application as soon as possible. From the research papers made by my supervisor [2, 3], I got the attributes' names for my database. Furthermore, because I did not have enough time, I had to use simulated data of smart environments. Thus, the app is more for testing different simulated smart environments rather than using it in real Smart Systems for now.

Chapter 5

Implementation

5.1 "Add a new simulated Environment" page

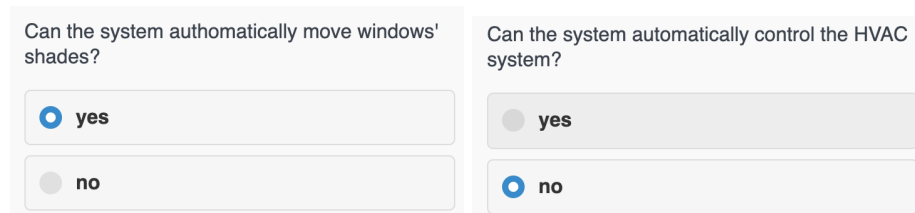


Tempurature indoor:

Tempurature outdoor:

Figure 4. Temperature indoor (filled in) and temperature outdoor (empty)

In the **Approach** Chapter, I stated that the "Add a new Simulated Environment" page is responsible for the creation of different environments to test. The page consists of many empty blanks to fill in. Their idea is that if I fill in the blank, it means that my Smart Building "has" the specific sensor from which the System can receive this data. However, in contrast, if I leave the input empty, then it means that the Smart Environment lacks the device to collect this information. Therefore, the System has to ask users a question on the HomePage to get this data. Figure 4 shows an example: the temperature indoor is 25. Thus, the System has a thermostat inside the room that can give this value. But, at the same time, the temperature outdoor is left empty, which means that the Smart Environment can not get this data by itself. So, it has to ask users to obtain a lacking value.



Can the system authomatically move windows' shades?	Can the system automatically control the HVAC system?
<input checked="" type="radio"/> yes	<input type="radio"/> yes
<input type="radio"/> no	<input checked="" type="radio"/> no

Figure 5. Move Windows' Shades (True) and control the HVAC System (false)

The same holds for actuators in the building. I added multiple radio input types to specify whether there are actuators or not. Figure 5 shows how it works: I

indicate that the System-I want to create—can move windows' blinds automatically, but at the same time, it does not control the HVAC System. Thus, when the algorithm wants to obtain the actions that actuators do automatically or users should perform, it will follow these rules. If there is a need to open windows' blinds, then it will be done by the System. On the contrary, if there is a need to turn on the fan, air conditioner, or heater, then the System will ask users to perform it by themselves. The representation of the page itself is in Figure 6.

09:47 192.168.1.247:3000 5

Add new simulated e...

How much is comfort more important than saving energy?

50

Name:

Country:

City:

Tempurature indoor:

Tempurature outdoor:

Humidity indoor:

Humidity outdoor:

Wind speed:

Contact information

Figure 6. "Add a new simulated Environment" page

There are two main rules when a user wants to create a new simulated Smart Environment to test. First, it is required to write the name of the Environment. Second, a user must specify whether true or false for every actuator.

5.2 HomePage

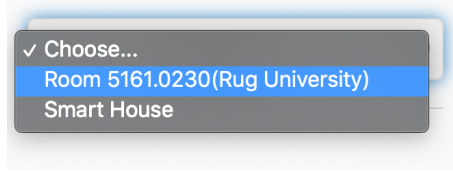


Figure 7. Choose the simulated Environment

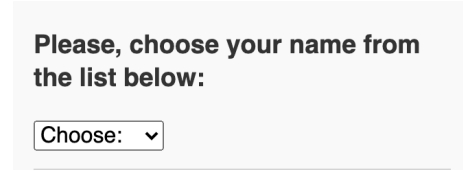


Figure 8. Choose the user's name

Figure 9. Add a new user

The HomePage is responsible for working with the algorithm itself. It starts with the list of all available simulated Environments (Figure 7). After users choose the environment they want to work with, the system asks them to indicate whether they have already assigned themselves to this environment or not, i.e., whether their names are in the database (Figure 9). If users see their names in the list provided by the System, they have to click "yes" in the form represented in Figure 9. Otherwise, they have to indicate that they do not see themselves in the list and, therefore, add their names to the environment's database (Figure 8).

Figure 10. Asks whether a user is free at the moment

Finally, users can start communication with the System. Before asking the required questions, the System asks users if they are free at the moment (Figure 10). Here users have two options: start working with the environment or skip their turn. If they choose to skip their turn, then the System will just ask

another user in the users' table in the Smart environment's database. Otherwise, the System will begin the collection of lacking data from users.

Name: Smart House
 Id: 2
 Country: France
 City: Paris
Temperature Inside: null
 Temperature Outside: 10
Humidity Inside: null
 Humidity Outside: null

Figure 11. Data of the simulated Smart Environment

It seems that the system does not know what temperature inside is

 Could you please help us to get some necessary data?

 Temperature inside:

+ Submit

+ Ask another user

Figure 12. Ask to fill in temperature inside value

It seems that the system does not know what humidity outside is

 Could you please help us to get some necessary data?

 Humidity outside:

+ Submit

+ Ask another user

Figure 13. Ask to fill in humidity outside value

Actions performed automatically:

 3 lamps are turned on

Actions you need to perform:

 Open window

Figure 14. List of Actions to perform

Thus, the "Smart House" simulated environment does not have information about the temperature inside and the humidity level outside. So, it has to ask a user to fill in their values (Figure 12, 13). After the System stores new data, it runs the algorithm to get the list of actions to perform (Figure 14). Therefore, for example, in this case, there is one action, which the System does automatically, and another - users have to perform.

5.3 "See Environments' data" page

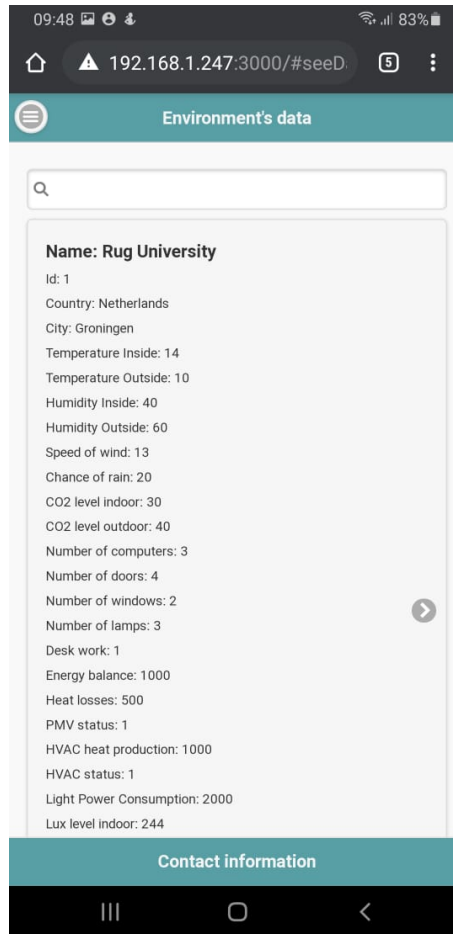


Figure 15. "See Environments' data" page

Figure 15 shows the implemented "See Environments' data" page. As stated in the **Approach** chapter, this page contains all data of the simulated Smart environments, so users can see what values lead to what results. The System updates this page every time it stores new values.

5.4 Algorithm

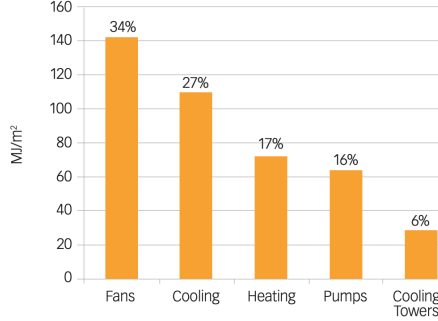


Figure 16. Typical HVAC end use breakdown 1 ^[7]

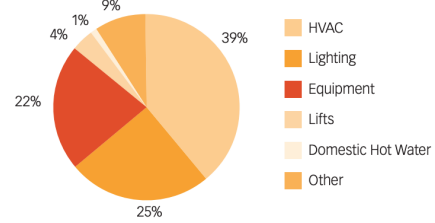


Figure 17. Typical energy consumption breakdown in an office building ^[7]

Before starting to write the algorithm, I had to decide what cost every device in the system will have. To obtain these values, I used the diagrams in figure 16, 17. According to the diagram in figure 17, it is obvious that the HVAC system is more expensive than electricity. Also, based on the histogram in figure 16, the ventilation requires more energy than cooling and heating. Thus, I obtained these cost values of devices of a simulated Smart Environment:

	FAN	AIR CON.	HEATER	ONE LAMP	ONE COMPUTER
COST	34	27	24	5	3

After I derived the necessary values, I was searching for the optimal values for temperature, levels of lux and CO₂ indoors. According to Carrier [1], Kane [8], and The greenage [6], the optimal values for the temperature, Lux Level, and CO₂ level indoors are 20 degrees, between 500 and 1000, and between 400 and 1000, respectively. So, based on these numbers and devices' energy cost values, I built my algorithm.

The main idea of my algorithm relies on the ratio between comfort and energy saving. On the "Add a new simulated Environment" page, there is a slider where users can choose how much comfortable environment is more important than energy saving. That is where the system gets the required ratio. Because the range of the slider is between 0 to 100, I have decided to create four different paths of the algorithm. First, when the relation of comfort and energy saving is between 0 and 0.33, this means that the reduction in energy consumption is more important. Second, when the ratio is between 0.33 and 1, this means that saving energy is slightly more important than a comfortable environment. Third, when the ratio is between 1 and 3, this means that the comfort of users is slightly more significant than the minimization of energy usage. Finally, when a comfortable environment is the most important, i.e., the ratio is greater than 3. Thus, that is when the devices' cost values are necessary. For the first ratio, the total devices' cost must not extend 30, for the second ratio - 50, for the third - 80, and the last one can have all 100. Therefore, when the app runs the algorithm, and it goes through all existed rules, it counts the total current cost. So, when the algorithm finishes running and obtains the final total cost value,

it verifies whether it is less than the relevant maximum possible total cost value assigned to the ratio of the environment. For example, if my ratio is 0.2, then the maximum possible devices' cost value is 50. So, a total devices' cost value 61 (34 (fan) + 27 (air conditioner)) is inappropriate for my environment. In this case, the System has to ask what is less important for me: turning on a fan or an air conditioner. After I choose one, the System will subtract its value from 61 and send me the list with only one action required (the one I did not choose as less important).

I attached the full version of the algorithm in Appendix A.

5.4.1 Set Temperature Indoors

In this document, I use logic statements to represent the rules I wrote to set the temperature inside the room:

$$\begin{aligned} R_1: & \text{ratio_comfort_economy} < 1 \\ Tin_{20}: & \text{the temperature inside} < 20 \\ Tout_{20}: & \text{the temperature outside} \geq 20 \\ R_{20}: & \text{the chance of rain} \leq 20 \\ W_{30}: & \text{wind speed} \leq 30 \end{aligned}$$

$$R_1 \wedge Tin_{20} \wedge Tout_{20} \wedge R_{20} \wedge W_{30} \rightarrow \text{open window}$$

$$R_1 \wedge Tin_{20} \wedge Tout_{20} \wedge \neg(R_{20} \wedge W_{30}) \rightarrow \text{turn on the HVAC system}$$

$$(R_1 \wedge Tin_{20} \wedge \neg Tout_{20}) \vee (R_1 \wedge \neg Tin_{20}) \rightarrow \text{consider PMV}$$

"Consider PMV" means to check the PMV status of the environment. There are seven possible values of PMV status, which are in the range from -3 ("too cold") to 3 ("too hot"). Because for some people, the optimal temperature value may be different, it is necessary to know their opinion before deciding what actions to perform. Some people can be hot when there are less than 20 degrees in the room, and some people can be cold when it is more than 20. Thus, this method helps to find the most optimal temperature value for users, which is only possible to obtain.

5.4.2 Set Lux Level Indoors

I, also, wrote logic statements to demonstrate rules for lux level indoors:

$$\begin{aligned} L_1: & \text{lux level indoor} < 500 \\ L_2: & \text{lux level indoor} > 1000 \\ T: & \text{time is between 9 am and 5 pm} \end{aligned}$$

$$L_1 \wedge T \rightarrow \text{open windows' blinds}$$

$$L_1 \wedge \neg T \rightarrow \text{turn on n number of lamps}$$

$$\neg(L_1 \vee L_2) \rightarrow \text{done}$$

$$L_2 \rightarrow \text{turn off l number of lamps, close windows' blinds if needed}$$

5.4.3 Set CO2 Level Indoors

Again, I derived logic statements to represent rules for CO2 level indoors:

C_1 : CO2_level indoor ≥ 400
 C_2 : CO2_level indoor ≤ 1000
 $R3$: ratio_comfort_economy < 3

$C_1 \wedge C_2 \rightarrow \text{done}$

$\neg(C_1 \wedge C_2) \wedge R3 \rightarrow \text{open window if possible}$

$\neg(C_1 \wedge C_2) \wedge \neg R3 \rightarrow \text{Turn on the ventilation}$

Chapter 6

Evaluation

6.1 Methodology evaluation

The choice of methods I used in this project was successful. The PhoneGap framework was convenient to use. Because it is open-source, it was completely free. Everything I had to do was to install its desktop app and run the server. After that, I could open and review my app on my smartphone. Therefore, it was easy to test the project every time I implemented a new functionality. Furthermore, because I was already familiar with JavaScript, HTML, and CSS, I did not have to spend time to learn it. To set the SQLite database was slightly harder due to the reason that I did not have a lot of work experience with the Back-end and implementing API. However, because SQLite is often used together with PhoneGap, there are many tutorials, which helped me to understand the topic better. Overall, I did not have any difficulties in setting up the project environment.

6.2 Testing algorithm

Env Name	Temperature Inside	Temperature Outside	CO2 level Inside	CO2 level Outside	Lux level Inside	Lux level Outside
First Attempt	25	20	500	500	100	400
Second Attempt	18	15	300	1000	100	400

Env Name	PMV Status	Number of Lamps	Wind Speed (km/hr)	Chance Of Rain (%)
First Attempt	2	9	6	10
Second Attempt	-2	9	6	10

Figure 18. Testing Simulated Environments Data

To test whether the algorithm gives the correct results, I tested it eight times. First, I created a simulated environment based on the data I got from my house.

I took temperature inside value from the thermometer. Other data, such as temperature outdoor, wind speed, the chance of rain, and humidity, I obtained from the weather app on my smartphone. Because I could not get the CO2 level inside or outside, I chose it to be 500. Also, because it was sunrise with no clouds, the lux level outdoor was 400 [14], and the lux level indoor was 100. Other values were not necessary for the algorithm at that moment. Therefore, I gave them random values. Moreover, because it was hot in the room, I let the PMV status be equal to 2. Figure 18 represents all required data for testing. The first row consists of data values about my house. The second row is data that I created myself to check all written rules in the algorithm and show that the algorithm works correctly.

Home	Ratio
1	0.17
2	0.67
3	1.86
4	99

Figure 19. Ratio of Testing Simulated Environments

Your environment is Home (1)	index.js:1163
It seems that saving energy is more important in this environment.	index.js:1183
The room is neither too cold nor too hot. No need to perform any actions.	index.js:1350
current cost is 0	index.js:1166
No actions are needed. CO2 level is in its optimal value.	index.js:1210
current cost is 0	index.js:1171
It is not bright enough in the room.	index.js:1249
It is dark outside, you have to turn on lamps	index.js:1255
Please, turn on 5 lamps. Current lux level is: 500.	index.js:1327
current cost is 25	index.js:1173
COST IS 25	index.js:1084

Figure 20. Testing the algorithm with data from my house

As I stated in section 5.4, there are four different possible results of the algorithm, and they depend on the ratio between the comfort of people and energy saving. Thus, I tested the same environment's data with four different ratio values, which figure 19 demonstrates. However, since except for lux level, other data values were optimal, the results were the same: final total devices' cost was 25, and the only action was to turn on five lamps (Figure 20). Therefore, I decided to create another simulated environment with data values shown in the second row of the table in figure 18. The reason I used those values was that they were not optimal according to my algorithm. Thus, it required more actions to perform. Home (1) is the simulated Smart Environment with the ratio between comfort and energy saving equal to 0.17, i.e., minimizing energy consumption is the more important in this case, which means that the best maximum possible devices' cost can not extend 30.

Your environment is Home (1)	index.js:1163
It seems that saving energy is more important in this environment.	index.js:1183
Your temperature indoor = 18. It is not enough for an optimal value of temperature.	index.js:1185
It is cold outside. Outdoor temperature is 15	index.js:1190
It is too cold in the room.	index.js:1344
Turn on actuator	index.js:1390
it is not okay. hvacOn is false	index.js:1409
working?	index.js:1417
Turn on heater.	index.js:1420
current cost is 26	index.js:1166
It is impossible to open the window right now. Turn on fan	index.js:1219
it is not okay. hvacOn is false	index.js:1409
working?	index.js:1417
Turn on fan.	index.js:1420
current cost is 60	index.js:1171
It is not bright enough in the room.	index.js:1249
It is dark outside, you have to turn on lamps	index.js:1255
Please, turn on 5 lamps. Current lux level is: 500.	index.js:1327
current cost is 85	index.js:1173
COST IS 85	index.js:1084

Figure 21. Testing the algorithm with the environment Home 1

What is LESS important for you?

✓ Choose

Fresh air (using energy)

Temperature in the room (warmer)

Light (lamps)

Figure 21(1). Choose what is less important

Подтвердите действие на странице localhost:8000

You chose Fresh air (using energy). Its cost is 34

Отмена

OK

Figure 21(2). Submit request

COST IS 51	index.js:1084
------------	-------------------------------

Figure 21(3). New total current cost (1)

COST IS 27	index.js:1084
------------	-------------------------------

Figure 21(4). New total current cost (2)

Actions you need to perform:

Turn on 5 lamps

Figure 21(5). List of actions to perform in Home 1

Thus, according to figure 21, I got the total devices' cost equal to 85, which was much larger than 30. Therefore, the System asked me to choose what I need less at the moment (Figure 21(1)). The first choice was Fresh Air, i.e., fan. According to the table of devices' cost given in section 5.4, the fan has a cost

value 34. Therefore, after I decided that I did not need to turn on the fan, the System subtracted its cost from 85 (Figure 21(2)), which made a new total cost value equal to 51 (Figure 21(3)). However, it was still over 30 and required for me to choose one more action I did not need to perform at the moment. Thus, I decided that I did not need the temperature in the room to be warmer. So, now the total cost value became 27 (Figure 21(4)). Therefore, the actions list, I obtained for Home (1), was to turn on five lamps (Figure 21(5)).

Your environment is Home (2)	e.g. /event\d/ -cdn url:a.com	index.js:1163
It seems that saving energy is more important in this environment.		index.js:1183
Your temperature indoor = 18. It is not enough for an optimal value of temperature.		index.js:1185
It is cold outside. Outdoor temperature is 15		index.js:1190
It is too cold in the room.		index.js:1344
Turn on actuator		index.js:1390
Turn on heater.		index.js:1418
current cost is 26		index.js:1166
It is impossible to open the window right now. Turn on fan		index.js:1219
Turn on fan.		index.js:1418
current cost is 60		index.js:1171
It is not bright enough in the room.		index.js:1249
It is dark outside, you have to turn on lamps		index.js:1255
Please, turn on 5 lamps. Current lux level is: 500.		index.js:1327
current cost is 85		index.js:1173
COST IS 85		index.js:1084
COST IS 51		index.js:1084
COST IS 26		index.js:1084

Figure 22. Testing the algorithm with the environment Home 2

Actions you need to perform:

Turn on heater

Figure 22(1). List of actions to perform in Home 2

Home (2) had a ratio value equal to 0.67 meaning that the comfort of the environment was also necessary. However, energy-saving in this place was still more important. Thus, the best maximum possible devices' cost could be anything less than 50. The result was similar to the one in Home (1) (Figure 22). But, the only difference was that instead of turning off the heater, I chose that I did not need my room to be brighter. Thus, the actions list (Figure 22(1)) differs from the one in Figure 21(5).

Your environment is Home (3)	index.js:1163
It is too cold in the room.	index.js:1344
Turn on actuator	index.js:1390
Turn on heater.	index.js:1418
current cost is 26	index.js:1166
It is impossible to open the window right now. Turn on fan	index.js:1219
Turn on fan.	index.js:1418
current cost is 60	index.js:1171
It is not bright enough in the room.	index.js:1249
It is dark outside, you have to turn on lamps	index.js:1255
Please, turn on 5 lamps. Current lux level is: 500.	index.js:1327
current cost is 85	index.js:1173
COST IS 85	index.js:1084
COST IS 51	index.js:1084

Figure 23. Testing the algorithm with the environment Home 3

Actions you need to perform:
Turn on heater
Turn on 5 lamps

Figure 23(1). List of actions to perform in Home 3

Home (3) had a ratio value of 1.86 meaning that even though energy conservation was still substantial, the comfort of users was more important. Thus, the best maximum devices' cost value was 80, and because the total devices' cost value was also 85 in Home (3), the System had to subtract only one value from 85 to obtain the desired number. Thus, I removed only one action: turn on a fan from the actions' list. Figure 23(1) shows the result I obtained.

Your environment is Home (4)	index.js:1163
It is too cold in the room.	index.js:1344
Turn on actuator	index.js:1390
Turn on heater.	index.js:1418
current cost is 26	index.js:1166
Turn on fan.	index.js:1418
current cost is 60	index.js:1171
It is not bright enough in the room.	index.js:1249
It is dark outside, you have to turn on lamps	index.js:1255
Please, turn on 5 lamps. Current lux level is: 500.	index.js:1327
current cost is 85	index.js:1173
COST IS 85	index.js:1084

Figure 24. Testing the algorithm with the environment Home 4

Actions you need to perform:
Turn on fan
Turn on heater
Turn on 5 lamps

Figure 24(1). List of actions to perform in Home 4

For Home (4), the comfort of users was the only important condition. Thus, there was no reason to remove any actions from the actions' list. Therefore, the System did not ask me any questions, and the result I have got is shown in Figure 24(1).

So, according to tests and their results, the algorithm I built is successful, i.e., everything works as I expected to.

6.3 App Interface

To check whether the app's interface is user-friendly enough, I asked my family members to try to use it. First, I created a simulated environment with no data except its name. Also, I added all my family members to the users' table attached to the new environment. Thus, the total number of users equaled to seven people. Then, I asked my brother to start answering questions on the HomePage. I also told him that if he could not find the required information, or he did not want to continue anymore, he could just skip his turn. So, after he answered three questions, my sister took his turn. This process continued until we answered all questions. After this experiment, I asked those who participated in it, whether it was convenient to use, and whether everything was clear for them. All of them answered that it was clear what the System expected from them. However, they did not know how to obtain some data such as CO₂ level inside because there was no CO₂meter. Therefore, I realized the goal to make the interface clear and easy to use. However, I have to consider to change questions in a way that all people can answer or add hints to some questions.

Chapter 7

Conclusion

7.1 Results

Smart Environments are the prominent topic of the 21st century. Multiple studies already discussed them and provided different solutions to improve their work. The main goal of the project was to develop the app that would make a solid communication bridge between users and the Smart System. After I reached what I aimed to, I obtained the following results. First, my app gives different solutions based on what users want. It can be an energy-efficient state of the environment, or it can focus on the comfort of users more. As I stated in the **Implementation** Chapter, the reason why it is possible is that I take into consideration the opinion of users on to what extent their comfort is more important to them than energy conservation in the environment. Second, when I derived the rules for the System, I wrote them in a way that it does not consider only numeric values, but also users' opinion. For example, in the `setTemperature` function of the algorithm, the PMV status is as important as temperature values. Thus, if the temperature value is not optimal, but the PMV status states that the condition is satisfied, then the System does not ask to perform additional actions. Also, the application asks users what is less substantial for them if it is required. One more useful functionality of the app is that a user does not have to answer all questions alone. If he/she is not in a condition to do that at the moment, they can always skip their turn. Thus, the System asks multiple users, and, consequently, minimizes one human participation cost. It allows people who assist the Smart Environment to stay comfortable and do not get exhausted. In conclusion, if I implement the possible future improvements discussed in section 7.2, I consider my app efficient to participate in the assistance of Smart Systems.

7.2 Future Work

There are two potential weaknesses in the approach discussed in Chapter 4. First, because the IOS operating system created some restrictions rules, the Apache Cordova is not working with it at the moment. Therefore, I was not able to test my application with Apple smartphones. If the situation does not change in the future, then I have to change the platform I was working on so that

people could use my app with every operating system. Second, the algorithm I wrote works well. But it is only due to the reason that I have a few data, sensors, and actuators to process. If I have more, then I have to add new if-else statements for each new data. It is time-consuming, and, therefore, the most important thing for future work is to change the algorithm to the Constraint Satisfaction problems (CSPs) solver. So, it will be possible to add more sensors, actuators, and data to process. Moreover, it would be preferable to implement the shared server, so that if one user updates some information, then others can also see it. Also, if the app gets the location of users, it does not need to ask to choose the environment on the HomePage. Furthermore, the notification system will make the app more efficient to use. Thus, if it is well-developed later, it will be sufficient to connect the app with real Smart buildings.

Bibliography

- [1] Carrier. Stay comfy blog. <https://www.staycomfy.com/blog/ideal-indoor-temperature>.
- [2] V. Degeler and E. Curry. Human-Assisted Rule Satisfaction in Partially Observable Environments. In *11th IEEE International Conference on Ubiquitous Intelligence and Computing (UIC 2014)*, IEEE, 2014.
- [3] V. Degeler and A. Lazovik. Dynamic Constraint Satisfaction with Space Reduction in Smart Environments. *International Journal on Artificial Intelligence Tools*, 23(06), 2014.
- [4] eNet SMART HOME. <https://www.enet-smarthome.com/en/>.
- [5] Patrick Ferreira. <https://apps.apple.com/pl/app/pilot-domoticz-jeedom/id902546368>.
- [6] THE greenage. What are lux and how much light do you need? <https://www.thegreenage.co.uk/lux-much-light-need/:text=500>
- [7] HVAC HESS. Hvac energy breakdown. <https://www.environment.gov.au/system/files/energy/files/hvac-factsheet-energy-breakdown.pdf>.
- [8] Kane. What are safe levels of co and co2 in rooms? <https://www.kane.co.uk/knowledge-centre/what-are-safe-levels-of-co-and-co2-in-rooms>.
- [9] C. Martini F. Francesca Odone M. Chessa, N. Noceti. Designing assistive tools for the market. <https://www.sciencedirect.com/topics/engineering/smart-environment>.
- [10] M. Obaidat and P. Nicopolitidis. *Smart cities and homes*. Morgan Kaufmann, pp. 1–16, 2016.
- [11] F. Pecora and A. Cesta. Dcop for smart homes: A case study. *Computational Intelligence*, 23(04):395–419, 2007.
- [12] P. Rashidi, N. Krishnan, and D. J. Cook. *Handbook on Securing Cyber-Physical Critical Infrastructure, Discovering and Tracking Patterns of Interest in Security Sensor Streams*. p. 481, 2012.
- [13] S. M. Shuja and N. Javaid. Efficient Scheduling of Smart HomeAppliances for Energy Managementby Cost and PAR OptimizationAlgorithm in Smart Grid. 2019.

- [14] Wikipedia. Daylight. <https://en.wikipedia.org/wiki/Daylight>.
- [15] xComfort. <http://www.xcomfort.com/>.

Appendix A

Algorithm (code)

```
1 function setMinMaxCosts(results, currentCost, selectedOption) {
2     var ratioBestCost = results.rows.item(0).bestMaxCost;
3     var description = results.rows.item(0).description;
4     environmentHandler.retrieveActions(getActions, currentCost,
5         selectedOption, ratioBestCost);
6 }
7
8 function getActions(results, currentCost, ratioBestCost,
9     selectedOption) {
10     var lstActions = $("#lstActions");
11     lstActions.empty();
12     var item = results.rows.item(0);
13     var costValue = currentCost;
14     var attributes = [item.turnOnFan, item.turnOnAirCon, item.
15         turnOnHeater,
16         item.turnOnLamp, item.turnOnComp];
17     var attrNames = ['Fresh air (using energy)',
18         'Temperature in the room (colder)', 'Temperature in the room (
19         warmer)',
20         'Light (lamps)', 'computer'];
21     var actuator = ['fan', 'air conditioner', 'heater', 'light', '
22         computer'];
23     var length = attributes.length;
24     if (currentCost > ratioBestCost) {
25         var h3 = $("

### 


```

```

37         aSelectAction.attr("id", "btnSelectAction");
38         aSelectAction.text("Submit");
39         $("#select-native-4").on("change", aSelectAction.on("tap",
function () {
40             var newValue =
41                 $("#select-native-4").children("option:selected").
val();
42             var r;
43             var actuatorCost = calculateActuatorCost(actuator[
newValue]);
44             if (actuator[newValue] === 'light') {
45                 actuatorCost *= item.num_LampsON;
46             }
47             r = confirm("You chose " + attrNames[newValue] +
48                 ". Its cost is " + actuatorCost);
49             if (r === true) {
50                 environmentHandler.updateActions(actuator[
newValue], 0,
51                 selectedOption);
52                 costValue -= actuatorCost;
53                 environmentHandler.updateActions('cost',
costValue,
54                 selectedOption);
55                 environmentHandler.retrieveActions(getActions,
costValue,
56                 selectedOption, ratioBestCost);
57             }
58         })
59     );
60     (lstActions.append(selectActions)).append(aSelectAction);
61 } else {
62     var count = 0;
63     item.turnOnFan + " " + item.turnOnAirCon + " " + item.
turnOnHeater +
64     " " + item.turnOnLamp + " " + item.turnOnComp);
65     var allAttributes = [item.openWindow, item.moveShades, item
.turnOnFan,
66     item.turnOnAirCon, item.turnOnHeater, item.turnOnLamp, 0];
67     var actionsIfOne = ['', 'Windows blinds are opened',
68     'Fan is turned on', 'Air Conditioner is turned on',
69     'Heater is turned on', item.num_LampsON +
70     ' lamps are turned on', ''];
71     var actionsIfTwo = ['Open window', 'Open windows blinds',
72     'Turn on fan', 'Turn on Air Conditioner', 'Turn on heater',
73     'Turn on ' + item.num_LampsON + ' lamps', 'Turn on ' +
74     item.num_CompsON + ' computers'];
75     var listIfOne = $("#listIfOne");
76     var listIfTwo = $("#listIfTwo");
77     var h4IfOne = $("

#### 


```

```

89         count = 0;
90         var h4IfTwo = $("<h4 />").text("Actions you need to perform
:");
91         for (var k = 0; k < allAttributes.length; k++) {
92             if (allAttributes[k] === 2) {
93                 listIfTwo.append($("<li />").text(actionsIfTwo[k]))
;
94                 count++;
95             }
96         }
97         if (count > 0) {
98             lstActions.append(h4IfTwo);
99         }
100         lstActions.append(listIfTwo);
101         listIfTwo.listview("refresh");
102     }
103 }
104
105 function setRules(results, selectedOption) {
106     var currentCost = 0;
107     var windowCanBeOpen = false;
108     var item = results.rows.item(0);
109     //start with a temperature.
110     currentCost += setTemperature(item.ratio_comfort_economy, item.
pmv_status,
111     item.temp_inside, item.temp_outside, item.rain_perc, item.
wind_kmhr,
112     item.isHVACsystemAuthoControlled, currentCost, selectedOption);
113     if (currentCost < 5) {
114         windowCanBeOpen = true;
115     }
116     currentCost += setCO2Level(item.ratio_comfort_economy, item.
CO2_level,
117     windowCanBeOpen, item.isHVACsystemAuthoControlled,
selectedOption);
118     currentCost += setLuxLevel(item.ratio_comfort_economy,
item.number_of_lamps, item.lux_level, item.
areWindowShadesAuthoMoved,
120     item.isLightAuthoSwitched, selectedOption);
121     environmentHandler.setCostsForRatio(setMinMaxCosts,
item.ratio_comfort_economy, currentCost, selectedOption);
122 }
123
124
125
126 function setTemperature(ratio, pmv, temp_inside, temp_outside, rain
,
127 wind, hvacOn, cost, selectedOption) {
128     var currentCost = 0;
129     switch (true) {
130         case (ratio < 1):
131             if (temp_inside < 20) {
132                 if (temp_outside >= 20) {
133                     currentCost += rainAndWind(rain, wind, 'heater
',
134                     calculateActuatorCost('heater'), false, hvacOn,
true,
135                     selectedOption);
136                 } else {
137                     currentCost += getPMVStatus(pmv, rain, wind,
hvacOn,
138                     false, selectedOption);
139                 }

```

```

140         } else {
141             currentCost += getPMVStatus(pmv, rain, wind, hvacOn
,
142             true, selectedOption);
143         }
144         break;
145         case (ratio >= 1):
146             currentCost += getPMVStatus(pmv, rain, wind, hvacOn,
false,
147             selectedOption);
148             break;
149             default:
150                 break;
151         }
152         return currentCost;
153     }
154
155     function setCO2Level(ratio, CO2_level, windowOpen, hvacOn,
selectedOption) {
156         var currentCost = 0;
157         switch (true) {
158             case (CO2_level >= 400 && CO2_level <= 1000):
159                 break;
160             case (ratio < 3):
161                 switch (windowOpen) {
162                     case true:
163                         environmentHandler.updateActions('window', 2,
selectedOption);
164                         break;
165                     default:
166                         currentCost += hvacONOFF('fan', hvacOn, false,
calculateActuatorCost('fan'), selectedOption);
167                 }
168                 break;
169             case (ratio >= 3):
170                 currentCost += hvacONOFF('fan', hvacOn, false,
calculateActuatorCost('fan'), selectedOption);
171                 break;
172             }
173         }
174         return currentCost;
175     }
176 }
177
178 //500-800 assuming led lamp is 8-12 = 800 lux = 800 per m^2 = 80
per 10 m^2.
179
180 function setLuxLevel(ratio, lampNum, lux_in, shadesMove,
lightSwitch,
181 selectedOption) {
182     var currentCost = 0;
183     var time = new Date();
184     var startTime = new Date();
185     startTime.setHours(9, 0, 0); // 9.00 am
186     var endTime = new Date();
187     endTime.setHours(17, 0, 0);
188     var count = 0;
189     var lux_inside = lux_in;
190     var lux_per_lamp = 800 / 10;
191     var lamp_num = lampNum;
192     switch (true) {
193         case (lux_in >= 500 && lux_in <= 1000):
194             break;
195         // case (ratio < 1):
196         //     switch (true) {
197

```

```

198         case(lux_in < 500):
199             switch (true) {
200                 case(time >= startTime && time <= endTime):
201                     currentCost += windowBlindsCloseOpen('open', '
closed',
202                         shadesMove, selectedOption);
203                     break;
204                 case(time < startTime || time > endTime):
205                     while (lamp_num > 0 && lux_inside < 500) {
206                         currentCost += calculateActuatorCost('light
');
207                         lux_inside += lux_per_lamp;
208                         lamp_num--;
209                         count++;
210                     }
211                     environmentHandler.updateActions('num_lamps',
count,
212                         selectedOption);
213                     environmentHandler.updateValue('lux_inside',
lux_inside,
214                         selectedOption);
215                     currentCost += lampsOnOff(count, lightSwitch, '
on',
216                         'off', lux_inside, selectedOption);
217                     break;
218                     default:
219                         break;
220             }
221             break;
222         case(lux_in > 1000):
223             while (lux_inside > 1000 || lamp_num > 0) {
224                 lux_inside -= lux_per_lamp;
225                 lamp_num--;
226             }
227             var lampsToTurnOff = lampNum - lamp_num;
228             environmentHandler.updateActions('num_lamps',
lampsToTurnOff,
229                 selectedOption);
230             environmentHandler.updateValue('lux_inside', lux_inside
,
231                 selectedOption);
232             var costLamps = lampsToTurnOff * calculateActuatorCost
('light');
233             currentCost -= costLamps;
234             currentCost += lampsOnOff(lampsToTurnOff, lightSwitch,
'off',
235                 'on', lux_inside, selectedOption);
236             if (lux_inside > 1000) {
237                 currentCost += windowBlindsCloseOpen('clos', 'open
',
238                     shadesMove, selectedOption);
239             }
240             break;
241             default:
242                 break;
243         }
244         return currentCost;
245     }
246
247     function windowBlindsCloseOpen(todo, state, shadesMove,
selectedOption) {
248         var currentCost = 0;

```

```

249     switch (shadesMove) {
250         case 'true':
251             currentCost += 2;
252             if (todo === 'open') {
253                 environmentHandler.updateActions('shades', 1,
254                     selectedOption);
255             }
256             break;
257         default:
258             if (todo === 'open') {
259                 environmentHandler.updateActions('shades', 2,
260                     selectedOption);
261             }
262
263             break;
264     }
265     return currentCost;
266 }
267
268 function lampsOnOff(number, lightSwitch, on_off, state, lux_inside,
269 selectedOption) {
270     var currentCost = 0;
271     switch (lightSwitch) {
272         case 'true':
273             currentCost += 2;
274             if (on_off === 'on') {
275                 environmentHandler.updateActions('light', 1,
276                     selectedOption);
277             }
278             break;
279         default:
280             if (on_off === 'on') {
281                 environmentHandler.updateActions('light', 2,
282                     selectedOption);
283             }
284             break;
285     }
286     if (on_off === 'on') {
287         environmentHandler.updateSetActuators('light',
288             selectedOption);
289     }
290     return currentCost;
291 }
292
293 function getPMVStatus(pmv, rain, wind, hvacON, openWindow,
294 selectedOption) {
295     var currentCost = 0;
296     var actuator = '';
297     var actuatorCost = 0;
298     var isOkay = false;
299     switch (true) {
300         case (pmv < -1):
301             actuator = 'heater';
302             actuatorCost = calculateActuatorCost(actuator) + 2;
303             currentCost += rainAndWind(rain, wind, actuator,
304                 actuatorCost,
305                 isOkay, hvacON, openWindow, selectedOption);
306             break;
307         case (pmv >= -1 && pmv <= 2):
308             isOkay = true;
309             break;
310         case (pmv > 2) :

```



```

309         actuator = 'air conditioner';
310         actuatorCost = calculateActuatorCost(actuator) + 2;
311         currentCost += rainAndWind(rain, wind, actuator,
    actuatorCost,
312         isOkay, hvacOn, openWindow, selectedOption);
313         break;
314     default:
315         break;
316 }
317 return currentCost;
318 }
319
320 function rainAndWind(rain, wind, actuator, actuatorCost, isOkay,
    hvacOn,
321 openWindow, selectedOption) {
322     var currentCost = 0;
323     switch (openWindow) {
324         case true:
325             switch (true) {
326                 case (rain >= 80):
327                     currentCost += hvacONOFF(actuator, hvacOn,
    isOkay,
328                     actuatorCost, selectedOption);
329                     break;
330                 case (rain <= 20):
331                     switch (true) {
332                         case (wind > 30):
333                             currentCost += hvacONOFF(actuator,
    hvacOn,
334                             isOkay, actuatorCost, selectedOption);
335                             break;
336                         case (wind <= 30):
337                             environmentHandler.updateActions('
    window', 2,
338                             selectedOption);
339                             break;
340                         default:
341                             break;
342                     }
343                 }
344             break;
345         case false:
346             currentCost += hvacONOFF(actuator, hvacOn, isOkay,
    actuatorCost, selectedOption);
347             break;
348         default:
349             break;
350     }
351     return currentCost;
352 }
353 }
354
355 //isCold true means too cold, isCold false mean too hot
356 //isOkay true means no need to turn on heater or conditioner
357 //Add actuator name and its actuatorCost
358 function hvacONOFF(actuator, hvacOn, isOkay, actuatorCost,
    selectedOption) {
359     var currentCost = 0;
360     var heaterTurnedOn = false;
361     var airConditionerTurnedOn = false;
362     var fanTurnedOn = false;
363     switch (isOkay) {
364         case false:

```

```

365         switch (hvacOn) {
366             case 'true':
367                 currentCost += actuatorCost;
368                 environmentHandler.updateActions(actuator, 1,
369                     selectedOption);
370                 break;
371             case 'false':
372                 currentCost += actuatorCost;
373                 environmentHandler.updateActions(actuator, 2,
374                     selectedOption);
375                 break;
376             default:
377                 break;
378         }
379         switch (actuator) {
380             case 'heater':
381                 heaterTurnedOn = true;
382                 environmentHandler.updateSetActuators(actuator,
383                     selectedOption);
384                 break;
385             case 'air conditioner':
386                 airConditionerTurnedOn = true;
387                 environmentHandler.updateSetActuators(actuator,
388                     selectedOption);
389                 break;
390             case 'fan':
391                 fanTurnedOn = true;
392                 environmentHandler.updateSetActuators(actuator,
393                     selectedOption);
394                 break;
395             default:
396                 break;
397         }
398         break;
399     case true:
400         break;
401     default:
402         break;
403 }
404 return currentCost;
405 }
406
407
408 function calculateActuatorCost(actuator) {
409     var actuatorCost = 0;
410     switch (actuator) {
411         case 'heater':
412             actuatorCost = 24;
413             break;
414         case 'air conditioner':
415             actuatorCost = 27;
416             break;
417         case 'fan':
418             actuatorCost = 34;
419             break;
420         case 'light':
421             actuatorCost = 5;
422             break;
423         case 'computer':
424             actuatorCost = 3;
425             break;
426         default:

```

```
427         break;
428     }
429     return actuatorCost;
430 }
```