



autoHome Report #3

Group #1

Paul Kania

Elie Rosen

Calvin Chiu

Rohith Dronadula

Elvison Dominguez

Wayne Chang

<http://autohome.mylifeiscomputers.net:3000>

<https://github.com/autohomeproject>

1. Contribution Breakdown

Responsibilities	Calvin Chiu	Elie Rosen	Elvison Dominguez	Paul Kania	Rohith Dronadula	Wayne Chang	Totals	Possible Points
Sec. 3 Changes				100%			100%	5
Sec. 4 CSR		25%		75%			100%	6
Sec. 5 Term Glossary		75%		25%			100%	4
Sec. 6 FRS	50%	15%	15%	20%			100%	37
Sec. 7 NFRS		2%	17%	1%		80%	100%	6
Sec. 8 DA					50%	50%	100%	25
Sec. 9 Interaction Diag	4%	4%	45%	4%	34%	9%	100%	40
Sec. 10 Class Diag	50%	25%			20%	5%	100%	20
Sec. 11 SA & SD	21%	28%		26%	1%	26%	100%	22
Sec. 12 Algo & Data		97%				3%	100%	4
Sec. 13 UID & UII				50%	50%		100%	8
Sec. 14 History & Cur						100%	100%	5
Sec. 15 Future Work		50%		50%			100%	5
Sec. 16 References	9%	21%	15%	10%	20%	25%	100%	3
Project Management		55%		30%		15%	100%	10
Totals:	34.99	35	25.02	35.08	34.92	34.99	100%	200

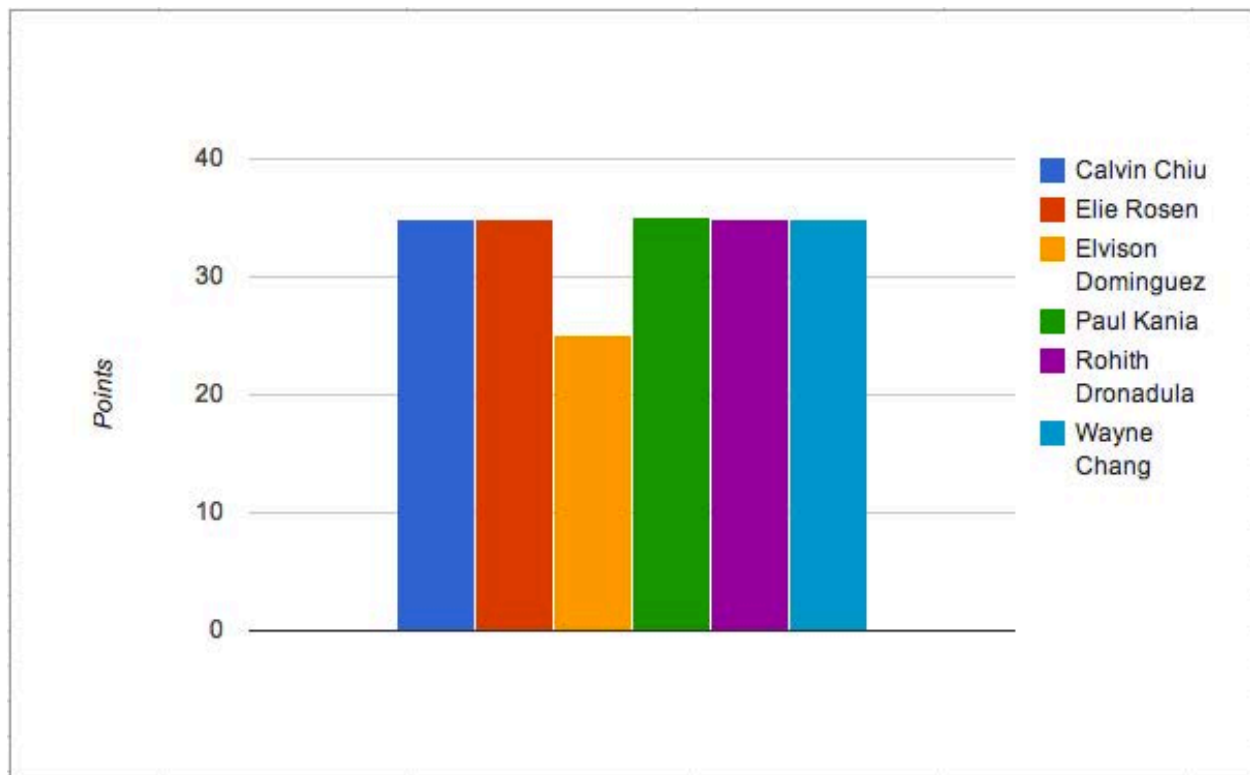


Table of Contents

1.	Contributions Breakdown.....	0
2.	Summary of Changes.....	0
3.	Customer Statement of Requirements.....	1
4.	Glossary of Terms.....	5
5.	Requirements Specification.....	9
	a. Enumerated Functional Requirements.....	9
	b. Enumerated Nonfunctional Requirements.....	10
	c. Stakeholders.....	10
	d. Actors and Goals.....	11
	e. Use Cases.....	11
	I. Casual Descriptions.....	11
	II. Use Case Diagrams.....	14
	III. Fully-Dressed Descriptions.....	21
	f. Traceability Matrix.....	35
	g. System Sequence Diagrams.....	36
	h. Nonfunctional Requirements.....	37
6.	Effort Estimation.....	39
7.	Domain Analysis.....	43
	a. Domain Model.....	43
	b. System Operation Contracts.....	47
	c. Mathematical Model.....	48
8.	Interaction Diagrams.....	49
9.	Class Diagram and Interface Specification.....	82
	a. Class Diagram.....	83
	b. Data Types and Operation Signatures.....	84
	c. Traceability Matrix.....	85
	d. Design Patterns.....	89
	e. Object Constraint Language (OCL).....	92
10.	System Architecture and System Design.....	95
	a. Architectural Styles.....	95
	b. Identifying Subsystems.....	96
	c. Mapping Subsystems to Hardware.....	97
	d. Persistent Data Storage.....	97
	e. Network Protocol.....	104
	f. Global Control Flow.....	104
	g. Hardware Requirements.....	105
11.	Algorithms and Data Structures.....	106
12.	User Interface Design and Implementation.....	107
	a. Preliminary Design and Implementation.....	107
	b. Current Design and Implementation.....	121

13. Design of Tests.....	140
a. Casual Test Cases.....	140
b. Descriptive Test Cases.....	141
14. Past, Present and Future Work.....	148
a. History of Work.....	148
b. Key Accomplishments.....	148
c. Current Status of Implementation.....	148
d. Future Work.....	149
15. References.....	150

2. Summary of Changes

As you read through this report, it will become very obvious that we have changed various aspects of our project since we created our first report. The changes made between the first two reports and this third, final report, are as follows:

- We've changed our Customer Statement of Requirements to better reflect what we have implemented and what we have stated through our use cases. Additionally, we have added an example of a day in the life without autoHome and the complications that can be caused.
- We've changed the design of the Web Interface to better suit the needs of potential customers with increased effectiveness while still being simplistic
- Our Use Cases have been updated to better reflect the objective of our project
- Since Report 1, we have decided to mainly focus on what we can do in the span of the class. With that, we decided against monitoring power and water consumption in the house as that is not essential to autoHome. We are focusing on the Web Interface and the functionality that comes with controlling a house over the web.
- Requirements have been changed to put in more detail for easier understanding. We also changed the priority weight of a few requirements that we deemed weighted unrealistically.
- The traceability matrix was updated to reflect the priority weight changes made in the functional requirements table.
- The preliminary design section has been updated to show what the preliminary design idea was, compared to what the actual implementation looks like in its current form.
- Requirements were split into both tabular Functional and Nonfunctional requirement tables.

The changes listed above only scratch the surface of the number of changes that were made between our first report, and this third, last report. As you read through this report, you will notice how far autoHome has truly come. It is no longer a concept, or a drawing on paper... autoHome has become a reality, and something that we, as a team, can prove is possible.

3. Customer Statement of Requirements

To our loyal followers,

Everyone has made the same mistake. You leave your house in the morning, rushing to get to work on time, and you forget to turn off the toaster oven in the house before you leave. You get to work, remember that you forgot to turn it off, and start to panic because this could possibly start a fire in your house when left unattended. Now, you really start freaking out and don't know what to do.

You end up clocking out of work, and rush home only to find that you actually turned the toaster oven off. You have now wasted gas by driving home, you are not making money at work because you clocked out, and you have to make up the time you missed; added stress that you don't need.

All of this stress will now be alleviated. The day has finally come, it is time to prepare yourselves for a home automation revolution.



After hearing about all of the pit falls that come with various home automation solutions available on the market, we knew it was time for us to step in.

Welcome to Advanced Utility to Operate Home, also known as autoHome.

autoHome plans to tackle common everyday household problems by implementing a unique set of sensors and controls to fully maintain and secure your home along with adding more modern conveniences that would further enable you to become more relaxed with your ever-growing stressful lifestyle.

Our product aims to provide a more simplified lifestyle for the average homeowner in the confines of their home. We understand the frustration users have when attempting to use our competitors products and we are here to revolutionize and simplify the realm of home automation.

After extensive testing of our competitor's products, we came to find that the basis of most users troubles come from the very complicated user interfaces which make it extremely hard to achieve completing tasks as simple as changing the temperature in your home. Instead, autoHome corrects this problem by implementing a centralized online dashboard, the user will be able to control their entire home from their computer or

mobile device even from thousands of miles away. Each home that uses autoHome will be configured to communicate with our dashboard by the use of sensors and automated appliances through the web interface. This would be useful for people who want to have the most control possible over their home as possible and certainly helpful for people with young children, or simply anyone who wants to have an easier, more enjoyable, living environment.

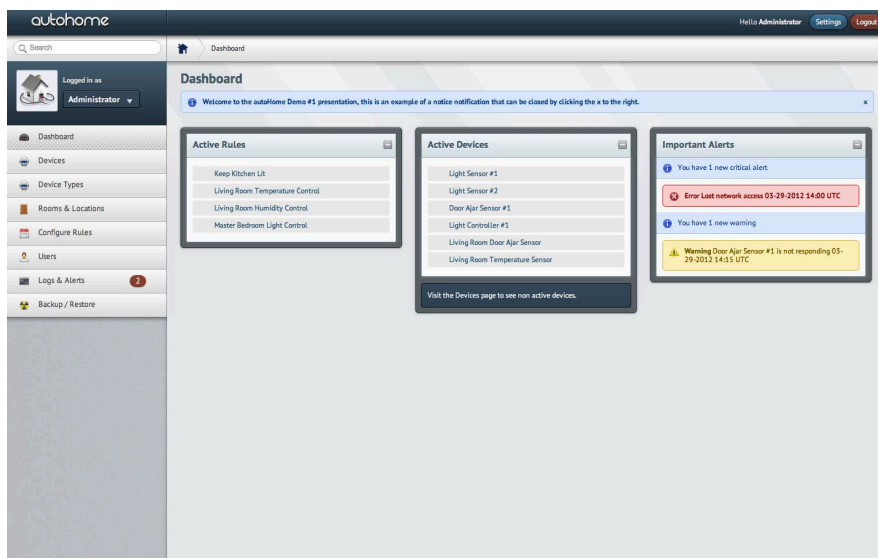


Figure 3.1 - autoHome's dashboard provides information in a clean, streamlined manner

The autoHome web interface is many things. Right at the login screen, you will notice that it is aesthetically pleasing. This is the first thing you see when you start using autoHome, so we wanted to reassure the user that they are using a top-notch product through its appearance. Upon logging in, you are greeted with a slew of options. The first screen the user is greeted with is the Dashboard. A number of different pieces of information are displayed on the Dashboard pane including active devices, active rulesets, and any alerts that should be brought to the user's attention. Overall, the autoHome dashboard will have an aesthetically pleasing look to it, while also providing key information to the user about their home. Additional features of the autoHome web interface will be detailed in the following paragraphs.

With autoHome, different sensors will be installed throughout the home to collect various data on the status of different devices in your home. The data is then sent to the central server and is analyzed to determine actions and triggers. These triggers are extremely customizable by the user, but autoHome also comes with a default set of rules that are very easy to implement by the user. All events are logged in the "Logs and Alerts" portion of the web interface, which the user can use to audit their home and make adjustments to further enhance their comfort and luxury. autoHome will also maintain a default profile that is not publicly available on the site, but will be used to recommend the user the most comfort available, while at the same time, saving the customer money by using energy wisely and effectively. This is yet another feature not found in any of our competitors' home automation solutions.

In order to help you save energy, autoHome will be able to control the window blinds in your home. By having control of the blinds, autoHome will be able to save energy by allowing sunlight to come through the

windows to serve as natural lighting. autoHome will keep track of sunrise and sunset data and adjust the blinds accordingly to provide optimal lighting in the home throughout the day. Of course, this can all be modified to your liking, such as when to shut the blinds if you decide you want them to shut at a specified time, or if you do not have work on a certain day and do not want to be potentially woken up by sunlight at sunrise that morning. Overall, this is another one of many features that will help you save money on your electricity bill each month using autoHome.

Climate control is a feature that is found in nearly every competitors' home automation solution. Obviously, we've included the feature in our autoHome system, but we've also implemented a clever feature to help control the temperature that the user sets in the home. We plan to implement sliding doors wherever possible in the home to make sure that the door is immediately closed right after a person enters or leaves through it. In ordinary homes, people turn on their air conditioning to cool the home, but then leave the doors open accidentally and make the home warmer. autoHome aims to eliminate this problem with our sliding doors.

One of the biggest features that we saw demand for were the use of motion sensors. Installing motion sensors in the home provides for a number of different features to function properly. First is the home security portion of functionality. autoHome can be armed and disarmed according to if a person is home or not. If our sensors detect motion when nobody is home, you will be notified by means of text messaging, email, or even a phone call. This, alone, makes our system more accessible than anything on the current market. In conjunction with motion, sensors will be wired to monitor the status of fire detectors and carbon monoxide detectors you use in your home. For instance, if autoHome detects that any of these alarms are going off while you are not home, namely the fire alarm, not only will you be notified by text message, email, or phone call, but local authorities will be notified at the same time to prevent possible excessive damage to your home. Similarly, if the motion sensors detect motion in your home when nobody is home, you will be notified as well as local authorities to thwart any criminal activity in your home.

In addition to monitoring electricity and our intelligent use of motion sensors, we will be doing some very interesting things with regards to water in your home. First off, autoHome will also monitor the water consumption in the home. Water usage will be an estimation, but will give the user a very good idea of how much water is used, what uses the most water, and what may even be wasting water in your home. Ultimately the user will have an overview of all the resource consumption in their home displayed conveniently on the web panel. Just like electricity monitoring, autoHome can give advice on how to adjust your water consumption to save money on your water and sewer bills. More importantly with regards to water is how autoHome reacts to potential flooding in your home. autoHome's sensors will also be able to determine if water is coming into your home. When it detects water coming into your home, autoHome will intelligently shut off all electrical devices that can be damaged by the incoming water.

Another interesting feature that we're implementing is the option to initialize devices. By this we mean that say you're in a situation where: you're at work until 5 PM, your wife calls and the two of you decide to watch a movie (streaming from Netflix) at 7 PM after dinner. You can simply log on to the autoHome dashboard from work, and set the TV, as well as your Xbox 360, to turn on at 6:55 PM, so that when you and your wife go to watch the movie, everything is set up, ready to go. Instead of waiting for devices to turn on and boot

up, devices will now be waiting for YOU! Additionally, all devices and appliances in the home can be scheduled to turn on or off at certain times, as you adjust them to. This proves to be especially useful when you need to turn an appliance on or off at a certain time when you are not home. On top of the scheduled controls feature, we plan to integrate inductive charging “spaces” around the home in order to easily charge your mobile devices anywhere around the home. Adapters will be made for a wide range of devices so that you can easily make sure that your device is compatible with the inductive charging spaces found around the home.

If it is not obvious enough yet, our system is unlike anything available on the market. We looked at what is available on the market, as well as your feedback that you made available online, and we designed autoHome. autoHome morphs a home automation system with a home security system as well as an energy-saving monitor. This is a very unique product entering an already flooded market and we hope you are a part of it!

We truly hope that you are as excited for autoHome as we are. We enjoyed developing it just as much as you'll enjoy using it on a daily basis. On behalf of autoHome, we would like to welcome you to the home automation revolution!

We are excited for you to try out our product and would like to again, welcome you to the automation revolution.

Sincerely,

Your friends at autoHome

4. Glossary of Terms

Action	An arbitrary setting of a single sensor value.
Action Set	A group of actions to be executed at the same time.
Analog Controller	A controller setting fluctuating data values. (Examined in the Domain Model)
Analog Sensor	A sensor providing fluctuating data values. (Examined in the Domain Model)
autoHome	The name of our system that proves to be the next big step in the home automation revolution.
autoHome Device	Any component that can be interfaced with the autoHome system through a Device Interface Array and scaled with a Device Interface Module.
Condition	A sensor value comparison that can be held true or false.
Condition Set	A group of conditions that must all be met to evaluate as true.
Controller Conductor	A conductor (telephony-grade wire) with voltages controlled on the system side.
Dashboard	A graphical web interface that allows users to view device information and set up rule sets to control devices.
Default Device Value	A value provided by the Device Interface Module to set Controlling Conductors to in the event of no active Action on an autoHome Device.

Device Interface Array

A physical device that provides many Controlling, Reporting, and Ground Conductors to link with autoHome devices. It aggregates the voltage values of the controlling and reporting conductors into a single connector to the autoHome System.

The Device Interface Array can aggregate data streams into a physical or virtual communications port. At the time of this writing, only USB ports are supported. There are prototypes to allow virtual ports over Ethernet, and these will be implemented in the future.

All Device Interface Arrays use a hardware driver specific to the type of connection used to interface with the system.

Device Interface Module

A driver-like piece of software providing reasonable scaling and unit conversions from voltages. A Device Interface Module might convert the voltage from a Reporting Conductor for a refrigerator from voltages of 2.0V to 5.0V to temperatures of 0C to 100C.

It might also convert a logical off or on for a lightbulb to a functional 0.0V or 3.0V actually sent through the Controlling Conductor.

The Device Interface Module requires close interaction with a Device Interface Array through a hardware driver.

Digital Controller

A controller setting binary data values. (Examined in the Domain Model)

Digital Sensor

A sensor providing binary data values. (Examined in the Domain Model)

FURPS+

Acronym that stands for Functionality, Usability, Reliability, and Performance. "+" stands for various requirements such as design, implementation, interface and physical requirements. This is a standard method that is used to describe and classify the non-functional requirements.

Ground Conductor

A conductor (telephony-grade wire) to ensure a common ground.

Home Automation	Automation of the home, housework or household activity. Home automation may include centralized control of lighting, HVAC (heating, ventilation and air conditioning), appliances, and other systems, to provide improved convenience, comfort, energy efficiency and security.
MySQL	World's most used relational database management system that runs as a server providing multi-user access to a number of databases.
PHP	A hypertext pre-processor. It's an open source, server-side, cross-platform, scripting language used to create dynamic web pages.
Plug & Play	Refers to the ability of a computer system to automatically configure expansion boards and other devices. You should be able to plug in a device and play with it, without worrying about setting DIP switches, jumpers, and other configuration elements.
Reporting Conductor	A conductor (telephony-grade wire) with voltages controlled on the device side.
Rule Set	A relationship between a Condition Set and an Action Set that may be enabled or disabled.

System Interface Module

A driver-like piece of software acting almost identically to a Device Interface Module, with a readings and controls linked to the computer system and Internet rather than physical devices.

A System Interface Module might have sensor report the value of the weather in a certain zip code.

A different System Interface Module might be controlled to send a text message selecting a text messaging profile (phone number and message) with the controller value.

System Interface Modules will be configured from within the web interface. Each System Interface Module will provide its own configuration web page. An example weather-reporting System Interface Module might have a web configuration page containing a zip code field and a units drop-down box containing Fahrenheit and Celsius. The autoHome System can then call the sensor to receive an analog value of the temperature in the specified units from the Internet.

Use Case

A usage scenario for our system which shows how a user communicates with the system to get their desired response.

XSS

A group of interrelated web development techniques used on the client-side to create asynchronous web applications.

5. Requirements Specification

a. Enumerated Functional Requirements

Req-X	PW	Description
REQ-1	2	The system shall have a robust authentication and permissions system as to prevent attackers from entering the system, and to prevent normal users from accessing unauthorized devices.
REQ-2	5	The system shall record values from its light, temperature, motion or any other sensors in a database at the specific rates (i.e. Volts, Lux, Celsius) pertaining to each sensor, and have the ability to clearly display these values to users.
REQ-3	10	The system shall control devices according a set of user-specified rules. The system controls devices such as the stove/oven, air conditioner, heater, television, radio, gaming consoles, alarm systems, lights, faucets, thermostats and other devices according to a set of user-specified rules. All system values will be monitored and controlled via the autoHome website or the system located at home.
REQ-6	6	The system shall include sufficient hardware to be able to effectively communicate via text messaging, phone calls, or emails.
REQ-8	3	The system shall have sufficient hardware to be able to query external data from the Internet to use in user-defined Rule Sets.

Figure 5.1 - Table of Enumerated Functional Requirements

b. Enumerated Nonfunctional Requirements

Req-X	PW	Description
REQ-2	5	The system shall record values from its light, temperature, motion or any other sensors in a database at the specific rates (i.e. Volts, Lux, Celsius) pertaining to each sensor, and have the ability to clearly display these values to users.
REQ-4	4	The system shall be redundant and have many precautions in place, such as repetitively backing up the system in the case that the system crashes. This would allow an easy and stream-lined recovery process in the case of a crisis such as a power outage.
REQ-5	7	The system shall have a very effective, simplistic, and stream-lined graphical user interface which allows for complete control over the system. The stream-lined GUI will allow people of any experience level to become completely familiar with the system in a very short amount of time.
REQ-7	5	The system shall allow administrators to add new devices and remove existing devices with ease.

Figure 5.2 - Table of Enumerated Nonfunctional Requirements

c. Stakeholders [2]

1. Internal Stakeholders
 - a. Developers (Internal): A person or group of people who codes and programs for autoHome and has rights to access all parts of it. Internal developers' interests are advanced functionality, ease of use, and affordability.
2. External Stakeholders
 - a. Developers (External): A person or group of people who want to utilize modules of auto Home for their personal use and/or business. External developers' interests are increasing profitability of consumer products and better relations with consumers.
 - b. Homeowners (Users): People who have our system installed in their home. Homeowners' interests are quality, value, service and reliability.
 - c. Competitors: Companies that provide very similar products to sell on the market to the general public. Competitors' interests are demand, profit, and customers.

d. Actors and Goals

1. User
 - Initiating type
 - Goals: To control their home based on their specifications, allowing for a safer and more user-controlled home environment.
2. Device Sensors
 - Initiating type
 - Goals: To constantly report meaningful binary or fluctuating values to the central server.
3. autoHome System
 - Participating/Initiating Type
 - Goals: To collect various data from sensors, analyze data and issue instructions to devices.
4. User Web Interface
 - Participating Type
5. autoHome Database
 - Participating Type
6. Device Controllers
 - Participating type
7. System Modules
 - Participating type

e. Use Cases

I. Casual Descriptions

UC#1 Authenticate User

A user wants to log into the system to modify system preferences, add or remove devices, configure schedules or request information. The user must provide the credentials of a valid user account in the autoHome system. The user is prompted to create an administrator-level account upon first boot of the system. An invalid entry shall result in an error notification and logging by the system, and user shall be prompted again for password.

UC#2 Create User Account

An authenticated administrator or initiating user wishes to create additional user accounts. User accounts may only be created without authentication on the first-time system start-up (hence initiating user), in which the first user account may be created with administrator-level privileges. The account creation process will prompt for valid first and last names, a username, a password, and privilege level (user or administrator).

UC#3 Change User Account Settings

An authenticated administrator wishes to change user account settings. The ability to change user account settings is only granted to administrators as to prevent abuse (falsification of names or data). The administrator may change any fields populated at account creation, or delete the user account.

UC#4 Configure User Device Permissions

An authenticated administrator may change the device permissions of a user. Device permissions give or restrict access to individual sensors and controllers to devices. Permission to a device allows a user to use that device in rule sets. Administrators have access to all devices.

UC#5 Configure Device Rule Sets

An authenticated user wants to configure condition sets and related action sets. Users can schedule certain times to enable devices, set up sensor value-dependent actions, or utilize the abilities of systems modules. Examples include locking doors during the night, turning on lights as light sensors detect low luminance, or sending a text message to a phone number if a motion sensor is activated on a certain date and time.

UC#6 Display Device Information

An authenticated user wants to geographically view authorized devices on a floor map and their data values as either text or graphs. The system will poll device map databases for locations, and logs for data values. The values are then pretty-printed or used to generate graphs. Additional calculations will be optionally displayed as well to show total usages, relationships, and trends.

UC#7 Export Device Log Data

An authenticated user wishes to export data values for authorized devices from the system logs. The data is outputted to the user's browser in a MIME-typed common database format.

UC#8 Install Device Interface Modules

An authenticated administrator wishes to update the system to support new devices. The administrator can upload a vendor-provided module to normalize raw sensor and controller data into sensible values and units.

UC#9 Uninstall Device Interface Modules

An authenticated administrator wishes to remove an interface module for a device. The module will be removed and the system will no longer be able to interface with devices using the module and will automatically remove all such devices.

UC#10 Add Interface Array

An authenticated administrator wishes to add an additional physical or virtual set of device slots for controlling and reading device values. The system will attempt to auto-detect the newly inserted physical or virtual interface arrays and request more information upon failure.

UC#11 Remove Interface Array

An authenticated administrator may remove a device interface array in case of malfunction or disuse of the array. All devices on the array are automatically removed during this use case.

UC#12 Add Device

An authenticated administrator may add a device to the system. The administrator is responsible for physical installation of the device and a proper connection to a valid interface array. The administrator is also responsible for installing the correct vendor-provided interfacing modules. The administrator shall specify the correct port of the device interface array and device type from drop-down style lists. The correct database entries to utilize the device will be made.

UC#13 Remove Device

An authenticated administrator may remove a device from the system. All rule sets, conditions, actions, condition sets, and action sets directly or indirectly referencing the device will be removed. This is to prevent security issues with rule sets not meeting full conditions before executing actions. Administrators are advised to refactor all condition and action sets prior to device removal. The device database entries are removed and the device can no longer be interfaced with.

UC#14 Send Device Signal

The system specifies values to set the connected devices to. The device interfacing modules convert these values to meaningful voltages to set the controlling conductors to. When there are no device settings, the interfacing modules provide default values to hold the controllers at.

UC#15 Receive Device Signal

The system receives values from the connected devices. The device interfacing modules convert the voltages from the devices into meaningful values to the system. Values are polled at rates specified by the device interfacing modules. All values are evaluated against relevant rule sets upon change.

UC#16 Add System Interface Module

An authorized administrator may add a system interface module to the autoHome system. These modules provide ways to interact with the computer system and the Internet as though as they were devices.

UC#17 Remove System Interface Module

An authorized administrator may remove a system interface module. All devices using the Interface Module will be removed.

UC#18 Send System Interface Module Signal

Set an active value to the system to use as a controller value in rule sets. Such values may include text message profiles IDs to text messaging modules, email profile IDs to emailing modules, and shell-script profile IDs to computer-control modules. Values and interfacing are handled by the device interfacing modules.

UC#19 Receive System Interface Module Signal

Retrieve a value from the system to use as a sensor value in rule sets. Such values may include system time, system date, and information (such as weather) from over the Internet. Polling rates and interfacing are handled by the device interfacing modules. All values are evaluated against relevant rule sets upon change.

UC#20 Evaluate Data Against Rule Sets

When new data is received, it is checked against the relevant active rule sets that contain the data value as a condition in its condition set. If the condition set is met (all conditions inside the set are fulfilled), then the associated action set will be executed.

UC#21 Execute Action Set

Executing an action set will actively set all devices to specified values until the associated condition set is no longer met (one or more conditions are unfulfilled). The device controllers are then set at their default values.

UC#22 Backup Database and Device Interface Modules

The autoHome Database and Device Interface Modules will be periodically dumped and stored either on server, on removable storage, or through the network at an off-site location. The database and Device Interface Modules store the entirety of the configuration data for autoHome, and are all that are required for complete system recovery. An administrator may also manually start a backup to a specified location.

UC#23 Restore Database and Device Interface Modules

An authorized administrator may restore the autoHome system to a previous state. The autoHome Database contents will be dropped and loaded with a verified backup. The Device Interface Modules will be erased and replaced with the ones in the backup.

II. Use Case Diagrams

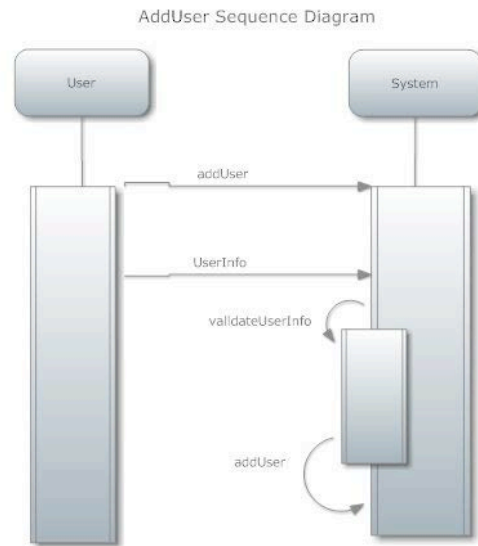


Figure 5.3 - AddUser Sequence Diagram

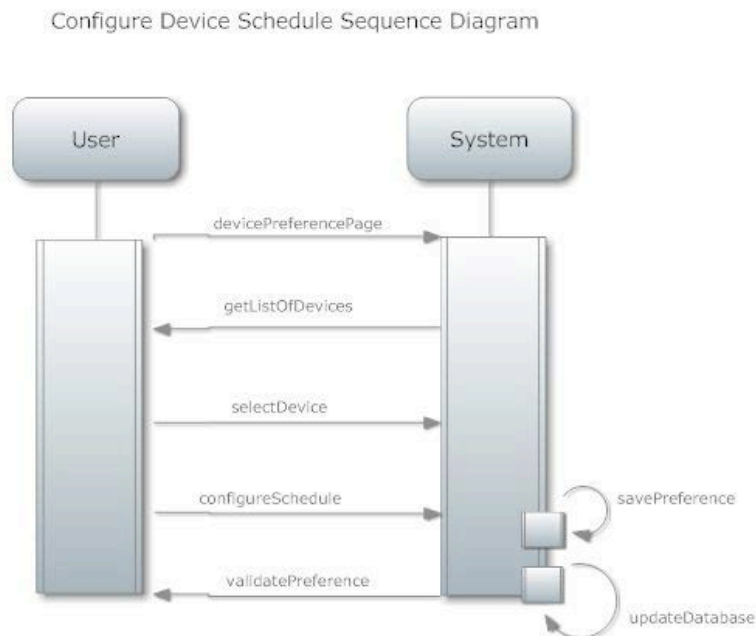


Figure 5.4 - Configure Device Schedule Sequence Diagram

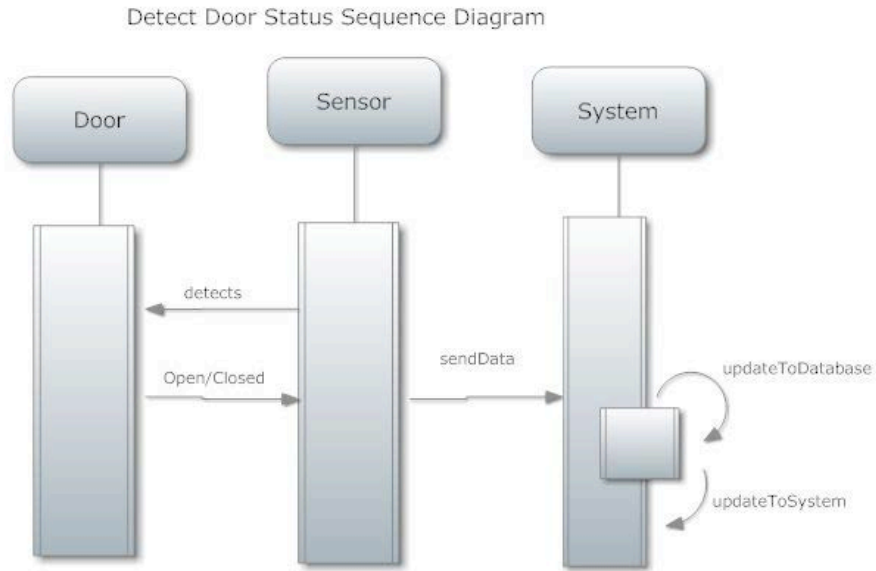


Figure 5.5 - Detect Door Status Sequence Diagram

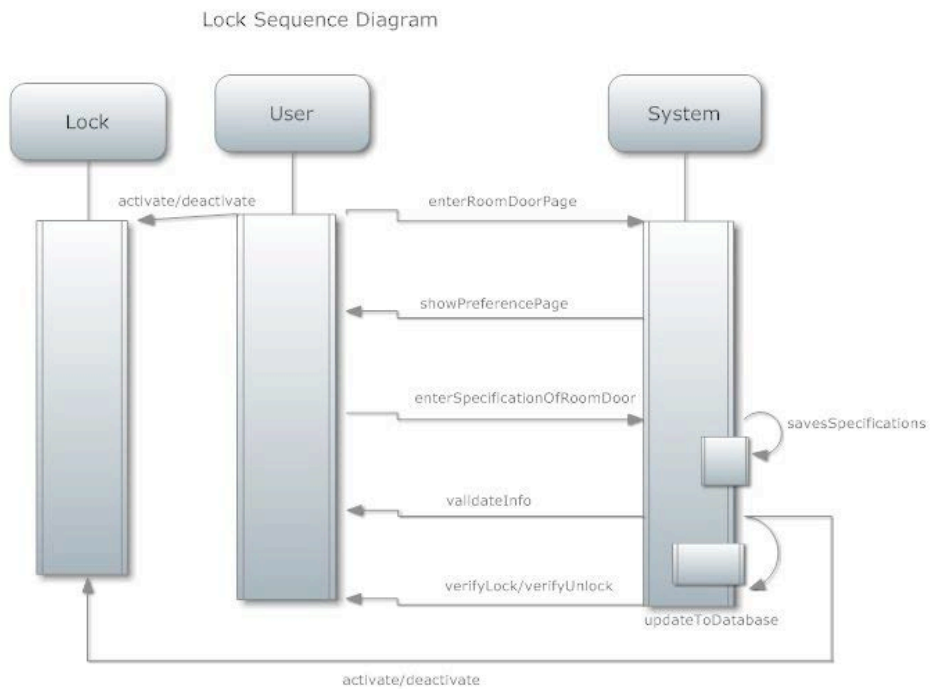


Figure 5.6 - Lock Sequence Diagram

Login Sequence Diagram

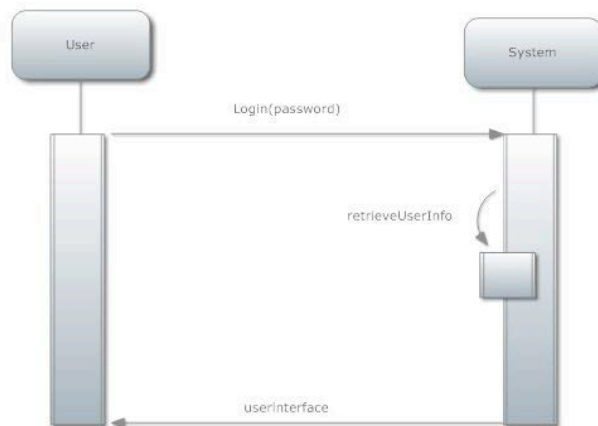


Figure 5.7 - Login Sequence Diagram

Measure Electricity Usage Sequence Diagram

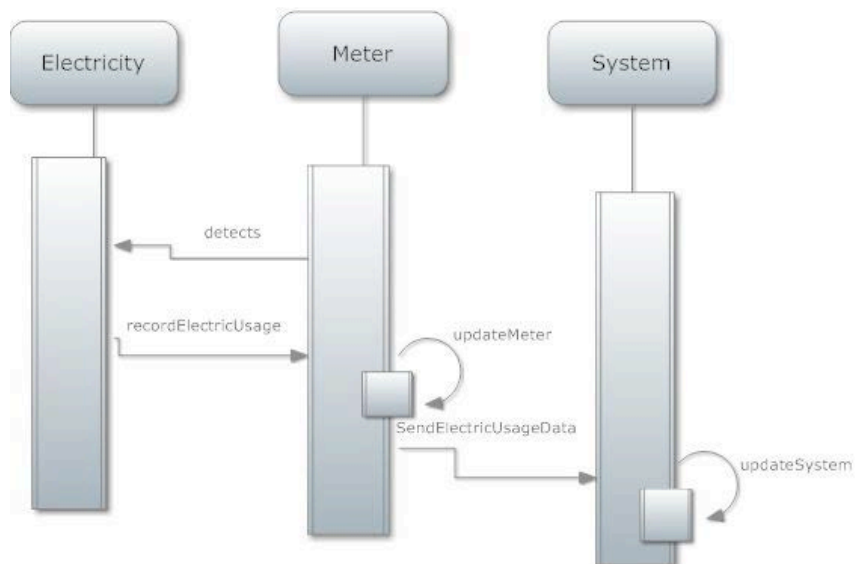


Figure 5.8 - Measure Electricity Usage Sequence Diagram

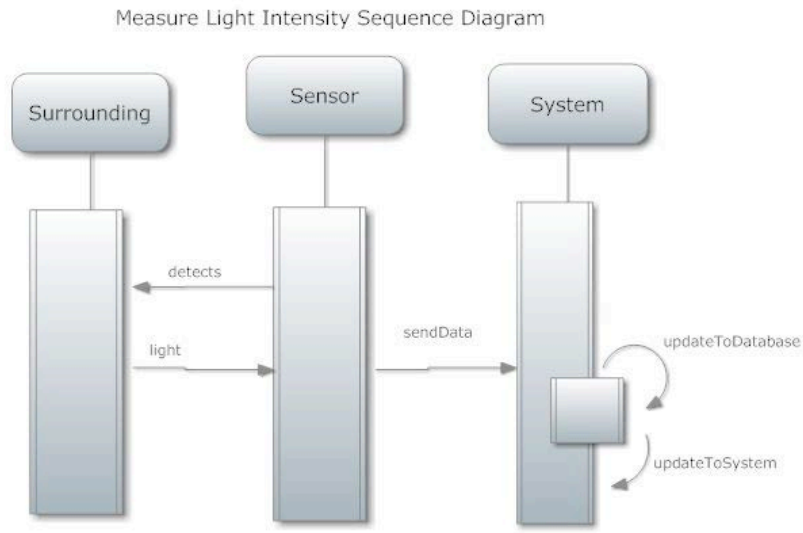


Figure 5.9 - Measure Light Intensity Sequence Diagram

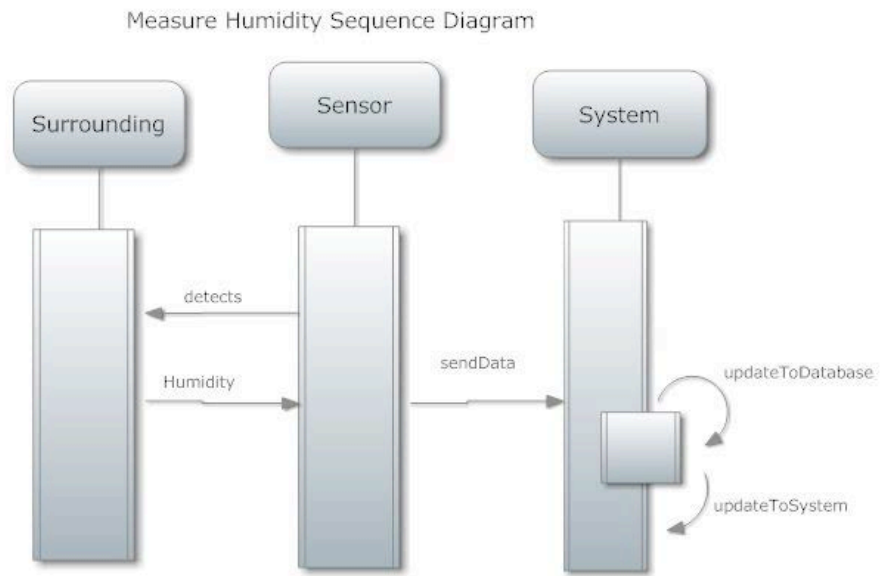


Figure 5.10 - Measure Humidity Sequence Diagram

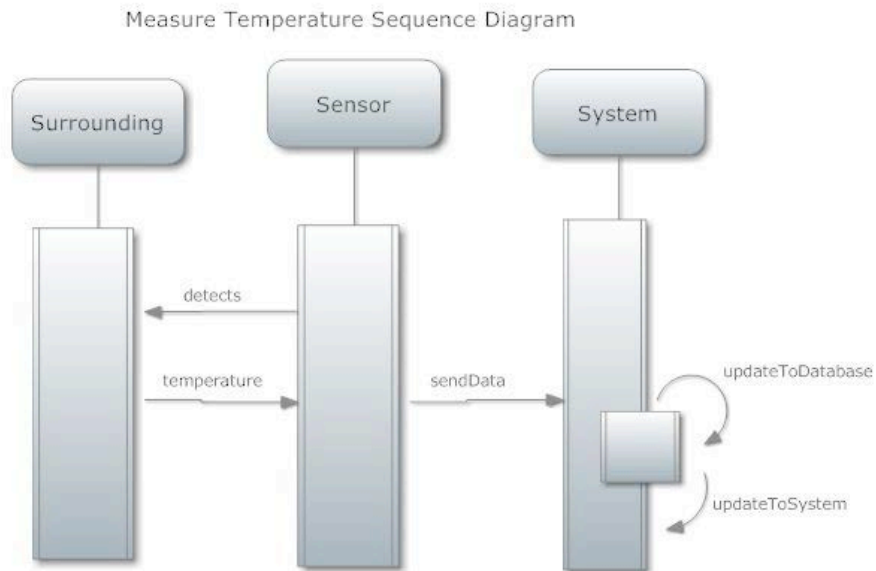


Figure 5.11 - Measure Temperature Sequence Diagram

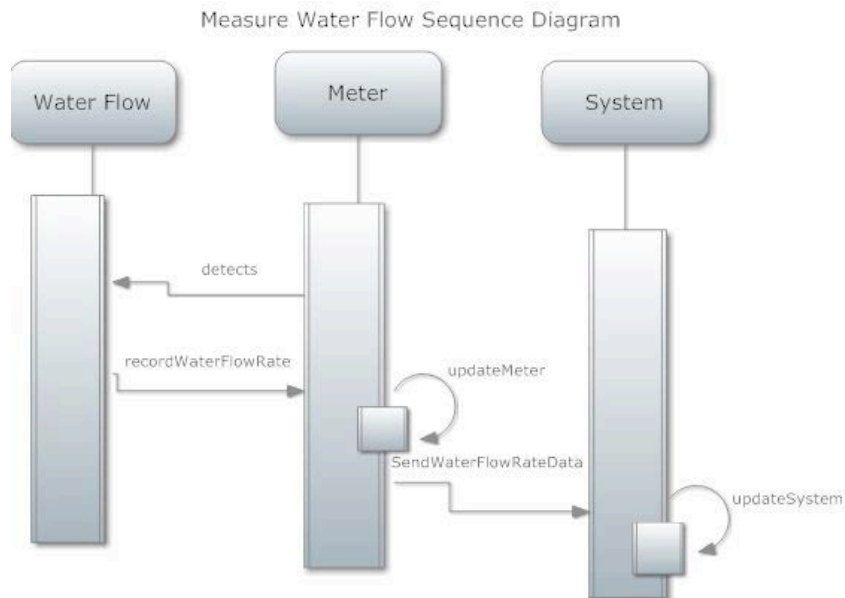


Figure 5.12 - Measure Water Flow Sequence Diagram

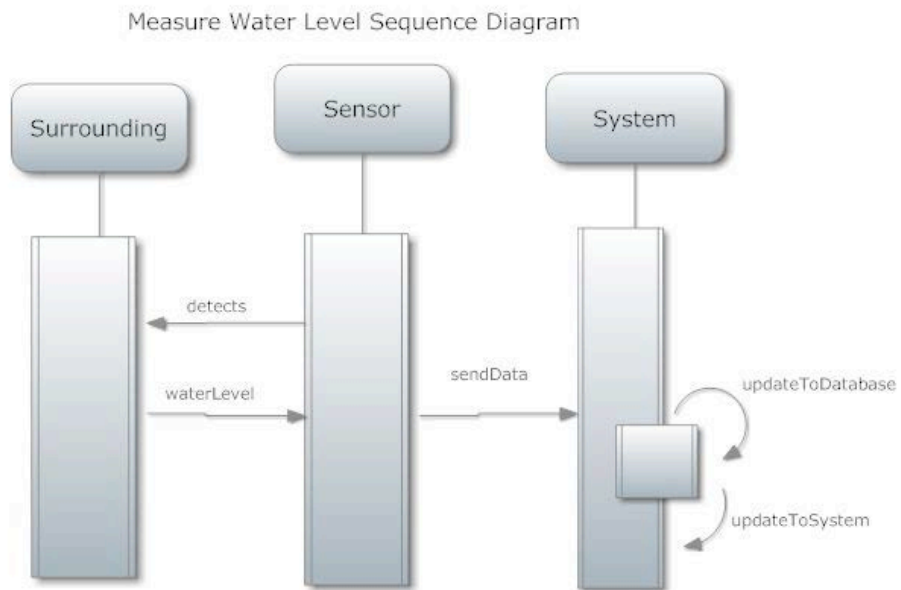


Figure 5.13 - Measure Water Level Sequence Diagram

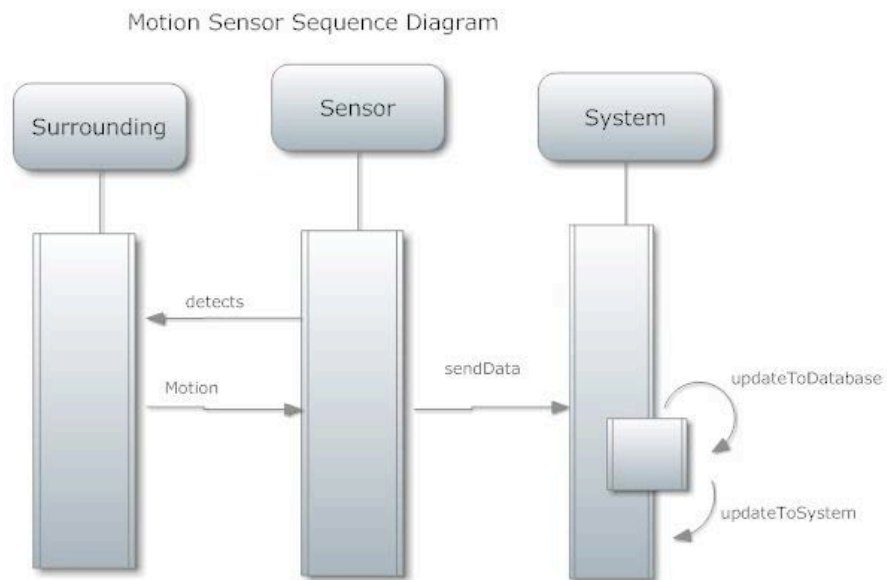


Figure 5.14 - Motion Sensor Sequence Diagram

Remove Device Sequence Diagram

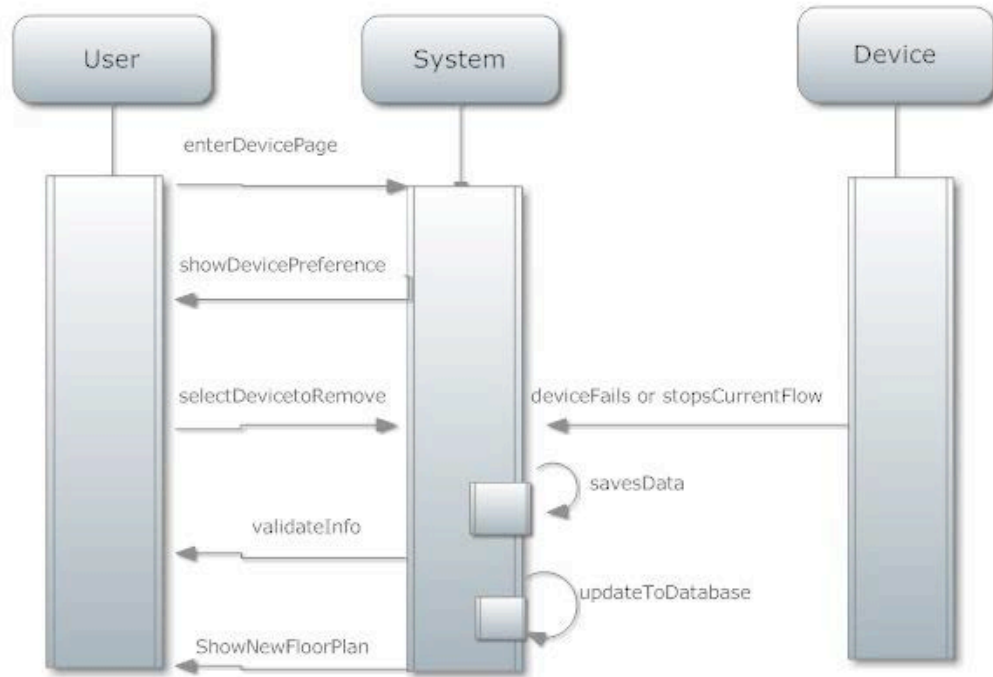


Figure 5.15 - Remove Device Sequence Diagram

III. Fully-Dressed Descriptions

UC-#1: Authenticate User		
Related Requirements	REQ-1, REQ5	
Goal In Context	Administrator attempts to log into the central control system and configure the preference or requests information	
Preconditions	System is activated and working	
Successful End Condition	Administrator inputs correct password using the factory default password and the interface is provided to the user	
Failed End Condition	Administrator inputs incorrect password and the login interface is provided to the user	
Actors	Administrator, Server	
Trigger	Administrator enters password number on the login screen	
Main Flow	Step	Action
	1	Administrator enters password and attempts to login
	2	System reads the password entered by the administrator authenticates with the password. Then the system provides the administrator with an interface that allows administrator to perform system configuration or request information.
Extensions	Step	Action
	2.1	Administrator enters invalid password. System notifies the administrator and returns to log-in screen. Invalid log in attempts trigger system to send a message notification to administrator. By default user can only try 5 times before the system is locked for 5 minutes.
	2.2	Administrator verifies his password and enters the password again.

Figure 5.16 - UC1: Authenticate User Use Case Table

UC-#2: Create User Account		
Related Requirements	REQ-1, REQ-5	
Goal In Context	Administrator creates user account for the system after first time initiation or a new user is created by administrator.	
Preconditions	First-time system initiation has not been done and administrator has logged into the system using factory default password or user wishes to create an account	
Successful End Condition	Administrator creates a new account and a new password is set up	
Failed End Condition	Administrator does not create a new account nor a new password	
Actors	Administrator, System, User	
Trigger	Administrator starts up the system on the first time and logs in using factory default password or administrator enters create user account page to create a new user account	
Main Flow	Step	Action
	1a	Administrator starts up the system on first time and logs in using factory default password
	1b	Administrator enters create user account page

	2	System provides the administrator an account set up interface
	3	Administrator enters preferences and password and saves the configuration
	4	System validates the information and saves the new account to the database
	5	System provides the administrator an interface that allows him to perform system configuration or request information
Extensions	Step	Action
	4.1	System detects incorrect information entered by the administrator.
	4.2	System prompts the administrator about the incorrect information and stays on the new user account configuration screen with errors highlighted.
	4.3	Administrator corrects the error and saves it.

Figure 5.17 - UC2: Create User Account Use Case Table

UC-#3 : Change User Account Settings		
Related Requirements	REQ-1, REQ-5	
Goal In Context	Administrator updates the account preferences	
Preconditions	A user account has been set up, the system is logged in	
Successful End Condition	Administrator successfully updates the account preferences	
Failed End Condition	Administrator does not update the account preferences	
Actors	Administrator, System	
Trigger	Administrator enters the account settings page	
Main Flow	Step	Action
	1	Administrator enters the account settings page
	2	Administrator enters new information and saves
	3	System validates the information (password length and combination, username) and saves the new information to the database
	4	System returns the administrator to home screen
Extensions	Step	Action
	3.1	System detects incorrect information entered by the administrator.
	3.2	System prompts the administrator about the incorrect information and stays on the new user account configuration screen with errors highlighted.
	3.3	Administrator corrects the error and saves it.

Figure 5.18 - UC3: Change User Account Settings Use Case Table

UC-#4 : Configure User Device Permissions		
Related Requirements	REQ-3, REQ-5	
Goal In Context	Administrator changes device permission for a user	
Preconditions	System is logged in and devices are connected to the system	
Successful End Condition	Administrator changes device permission for a user	
Failed End Condition	Administrator does not change device permission for a user	
Actors	Administrator, System, Device	
Trigger	Administrator enters Device Preference page	
Main Flow	Step	Action
	1	Administrator enters Device Preference page and select a specific device
	2	System provides an interface that allows administrator to edit the device preference
	3	Administrator changes device permission and saves
	4	System validates the integrity of the configuration (admin and user permissions level) and updates the database.
	5	System returns the administrator the Device Preference page.
Extensions	Step	Action
	1.1	Administrator cannot find a specific device and proceed to UC#12
	3.1	System detects incorrectness in the configuration
	3.2	System prompts the administrator about the incorrectness and stays on the same screen with errors highlighted
	3.3	Administrator corrects the errors and saves it.

Figure 5.19 - UC4: Configure User Device Permissions Use Case Table

UC-#5 : Configure Device Rule Sets		
Related Requirements	REQ-3, REQ-5	
Goal In Context	Administrator configures the device condition sets and related action sets	
Preconditions	System is logged in and devices are connected to the system	
Successful End Condition	Administrator configures the device rule sets	
Failed End Condition	Administrator does not configure the device rule sets	
Actors	Administrator, System, Device	
Trigger	Administrator enters Device Preference page	
Main Flow	Step	Action
	1	Administrator enters Device Preference page
	2	System provides an interface that allows administrator to edit the device preference
	3	Administrator configures the device condition sets and related action sets
	4	System verifies the integrity of the configuration with respect to the specification of the device and updates the database

Extensions	Step	Action
	4.1	System detects incorrect configuration entered by administrator
	4.2	System prompts the administrator and stays on the same screen with errors highlighted
	4.3	Administrator corrects the configuration and saves it

Figure 5.20 - UC5: Configure Device Rule Sets Use Case Table

UC-#6 : Display Device Information		
Related Requirements	REQ-5	
Goal In Context	System displays various device information to the administrator	
Preconditions	System is logged in and devices are connected to the system	
Successful End Condition	System displays device information with respect to administrator's request. Information including device map, logs, usages, trends, relationships, in text form and graphs.	
Failed End Condition	System does not display device information to administrator nor the information is correct	
Actors	Administrator, System, Device	
Trigger	Administrator enters Device Information screen and request information	
Main Flow	Step	Action
	1	Administrator enters Device Information screen and request information
	2	System polls device map from database for locations and logs, as well as analyzed data from UC#18 and displays information to user in text form and graphs
Extensions	Step	Action
	2.1	Some device information is not shown
	2.2a	Administrator manually force-refreshes the system to detect newly connected device.
	2.2b	Administrator manually adds device to the system database
	2.3	System re-polls device map from updated database and display information

Figure 5.21 - UC6: Display Device Information Use Case Table

UC-#7 : Export Device Log Data		
Related Requirements	REQ-5, REQ-6	
Goal In Context	Administrator exports data values for specific device from the system log	
Preconditions	System is logged in and devices are connected	
Successful End Condition	Data for specific device is exported in MIME-type format	
Failed End Condition	Data is not exported	
Actors	Administrator, System, Device	
Trigger	Administrator requests system to export device data	

Main Flow	Step	Action
	1	Administrator requests system to export device data
	2	System polls data of the specific device and saves it in a MIME-typed common database format. Then the data is accessible on administrator's mobile device or PC's browser
Extensions	Step	
	2.1	System cannot find any data for the specific device
	2.2	System acknowledges the administrator that no data for that device is found.

Figure 5.22 - UC7: Export Device Log Data Use Case Table

UC-#8: Install Device Interface Modules		
Related Requirements	REQ-5, REQ-7	
Goal In Context	Administrator updates the system to support new devices	
Preconditions	System is logged in and some new devices are not recognized by the system	
Successful End Condition	Administrator updates the system to support new devices and system detects them	
Failed End Condition	Administrator does not update the system to support new device and thus the system cannot recognize the new devices	
Actors	Administrator, System, Device	
Trigger	Administrator enters Add/Remove Device Interface Modules page	
Main Flow	Step	Action
	1	Administrator enters Add/Remove Device Interface Modules page
	2	System provides an interface that allows administrator to choose to update the modules from internet, driver disc or USB
	3	Administrator chooses the media and updates the system database
	4	System displays a list of newly updated devices to the administrator
Extensions	Step	Action
	4.1	Some devices are still missing after the update is completed
	4.2	Administrator requests system to reinstall the package
	4.3	System removes the package in UC#9 and reinstalls it back

Figure 5.23 - UC8: Install Device Interface Modules Use Case Table

UC-#9: Uninstall Device Interface Modules		
Related Requirements	REQ-5, REQ-7	
Goal In Context	Administrator removes a device interface module	
Preconditions	System is logged in and devices are connected	
Successful End Condition	Administrator removes a device interface module from the database	
Failed End Condition	Administrator does not remove a device interface module from the database	
Actors	Administrator, System, Device	
Trigger	Administrator enters Add/Remove Device Interface Modules page	
Main Flow	Step	Action
	1	Administrator enters Add/Remove Device Interface Modules page
	2	System provides a list of devices for administrator to choose
	3	Administrator chooses a device and removes the interface module
	4	System refreshes the database and display the updated list
Extensions	Step	Action
	4.1	After the system refreshes the database the device is still shown on the list. System prompts administrator about unsuccessful removal
	4.2	Administrator checks if the corresponding device is still in use or not and stops it. Then he uninstalls the module again.

Figure 5.24 - UC9: Uninstall Device Interface Modules Use Case Table

UC-#10 : Add Interface Array		
Related Requirements	REQ-5, REQ-7	
Goal In Context	Administrator adds a physical or virtual set of device slots for controlling and reading device values	
Preconditions	System is logged in	
Successful End Condition	Administrator adds a physical or virtual set of device slots	
Failed End Condition	Administrator does not add a physical or virtual set of device slots	
Actors	Administrator, System, Device	
Trigger	Administrator plugs in the physical device to the system or installs a virtual device like a software probe	
Main Flow	Step	Action
	1	Administrator plugs in the physical device to the system or installs a virtual device like a software probe
	2	System automatically detects the device and updates the database
	3	Administrator is notified upon successful installation
Extensions	Step	Action
	3.1	Upon failure, system requests the administrator to provide additional information for the interface array. System provides a interface configuration page for the administrator to edit the details
	3.2	Administrator enters the configuration and saves it

	3.3	System refreshes the database and re-detects the interface array. Then it updates the database.
--	-----	---

Figure 5.25 - UC10: Add Interface Array Use Case Table

UC-#11 : Remove Interface Array		
Related Requirements	REQ-5, REQ-7	
Goal In Context	Administrator removes a physical or virtual set of device slots	
Preconditions	Interface array is installed	
Successful End Condition	Administrator removes a physical or virtual set of device slots	
Failed End Condition	Administrator does not remove a physical or virtual set of device slots	
Actors	Administrator, System, Device	
Trigger	Array malfunction or misuse of the array is detected by the system	
Main Flow	Step	Action
	1	Array malfunction or misuse of the array is detected by the system
	2	System automatically stops the service and removes the interface array. Then it notifies the administrator about the removal
Extensions	Step	Action
	2.1	System does not stop the service nor remove the interface array. Then it notifies the administrator about this error and requests manual removal. It provides an interface for administrator to remove the array manually (software) or requests the administrator to physically remove the device.

Figure 5.26 - UC11: Remove Interface Array Use Case Table

UC-#12 : Add Device		
Related Requirements	REQ-5, REQ-7	
Goal In Context	Administrator adds a new device to the system and the system updates the device database	
Preconditions	A new device has just connected to the system and configured	
Successful End Condition	Administrator adds a new device to the system and the system updates the device database	
Failed End Condition	Administrator does not add a new device and the system does not update the device database	
Actors	User, Server, Device	
Trigger	Administrator connects a new device to a valid interface array and enters Add/Remove Device page	
Main Flow	Step	Action
	1	Administrator connects a new device to a valid interface array and enters Add/Remove Device page
	2	System checks the database for a corresponding device interface module
	3	System provides an interface for administrator to specify the correct port of the device interface array and device type

	4	Administrator enters the required information and saves it
	5	System updates the database and notifies administrator upon success
Extensions	Step	
	2.1	System cannot find the corresponding device interface module. Then it prompts the administrator to install the interface module in UC#8
	5.1	System cannot find the specified array port. Then it prompts the administrator to install a new array in UC#10

Figure 5.27 - UC12: Add Device Use Case Table

UC-#13 : Remove Device		
Related Requirements	REQ-5, REQ-7	
Goal In Context	Administrator removes a device from the system and its corresponding configurations	
Preconditions	The device is connected and configured	
Successful End Condition	Administrator removes a device from the system and its corresponding configurations such as rule sets, conditions, actions, condition sets, and action sets directly or indirectly referencing the device.	
Failed End Condition	Administrator does not remove a device nor the corresponding configurations are removed	
Actors	Administrator, System, Device	
Trigger	Administrator enters Add/Remove Device page	
Main Flow	Step	Action
	1	Administrator enters Add/Remove Device page
	2	System display a list of configured and connected device to administrator
	3	Administrator remove a device from the system
	4	System prompts the user to edit the configurations in UC#4 and UC#5 to prevent security issues with the residue rule sets and conditions
	5	User shall remove the device configuration in UC#4 and UC#5
	6	System updates the database after all the configurations are made
Extensions	Step	
	6.1	System prompts the administrator upon unsuccessful removal in UC#18 and requests the administrator to check if the device is still running
	6.2	Administrator checks the device and redo the process again

Figure 5.28 - UC13: Remove Device Use Case Table

UC-#14 : Send Device Signal		
Related Requirements	REQ-2	
Goal In Context	System sends instruction via specific voltage to control devices	
Preconditions	Device is connected to the system	
Successful End Condition	System sends instruction to the device and the device responds accordingly	
Failed End Condition	System does not send instruction to the device nor the device responds accordingly	
Actors	System, Device, Administrator	
Trigger	UC#18 is successful	
Main Flow	Step	Action
	1	UC#18 is successful and system detects a need to alter the device status
	2	System sends instruction via a specific value and a device interfacing module convert the value into meaningful voltages to set the controlling conductors.
	3	The corresponding device is thus initiated with respect to the instruction and responds to the system in UC#15
Extensions	Step	
	3.1	The device is not initiated by the instruction and abnormality in UC#15 is detected by the system
	3.2a	The system attempts to send instruction again
	3.2b	Upon failure, administrator is notified in UC#18 and is requested to check the connection of the device
	3.3b	Administrator checks the device connection and acknowledges the system

Figure 5.29 - UC14: Send Device Signal Use Case Table

UC-#15 : Receive Device Signal		
Related Requirements	REQ-2	
Goal In Context	System receives values from the connected devices and the values are converted into meaningful values by device interfacing modules	
Preconditions	Sensors are connected to the system	
Successful End Condition	System receives meaningful values and can be used in UC#20	
Failed End Condition	System does not receive values nor the device interfacing gives meaningful values to the system	
Actors	System, Device, Administrator	
Trigger	System receives signals from device	
Main Flow	Step	Action
	1	System receives signals from device
	2	Device interfacing module converts the value into system readable value
	3	System saves the value into a log file and prepares the value to be used in UC#14

Extensions	Step	
	2.1	System cannot read the value from device interfacing module
	2.2a	System attempts to send an instruction to request device status via UC#14
	2.2b	Upon failure, administrator is notified in UC#18 and requested to check the status of the device interfacing module configuration and the device connection
	2.3b	Administrator corrects the problem and system attempts 2.2a

Figure 5.30 - UC15: Receive Device Signal Use Case Table

UC-#16 : Add System Interface Module		
Related Requirements	REQ-5, REQ-7	
Goal In Context	Administrator adds a system interface module to the system to fetch Internet data	
Preconditions	System is logged on	
Successful End Condition	Administrator adds a system interface module to the system	
Failed End Condition	Administrator does not add a system interface module	
Actors	Administrator, System	
Trigger	Administrator enters Add/Remove External Data Source page	
Main Flow	Step	Action
	1	Administrator enters Add/Remove External Data Source page
	2	System provides a page for administrator to configure the source of the data
	3	Administrator choose to fetch data from internet or from a software that is installed on the system
	4	System updates the database and starts fetching data from the new source in UC#19
Extensions	Step	
	4.1	System cannot fetch any data from the specified source and prompts the administrator about the error
	4.2	Administrator corrects the error and system updates the database and starts fetching data from the source in UC#19

Figure 5.31 - UC16: Add System Interface Module Use Case Table

UC-#17 : Remove System Interface Module		
Related Requirements	REQ-5, REQ-7	
Goal In Context	Administrator removes a system interface module	
Preconditions	A system interface module is installed	
Successful End Condition	Administrator removes a system interface module and system no longer receives data from UC#19	
Failed End Condition	Administrator does not remove a system interface module	
Actors	Administrator, System	
Trigger	Administrator enters Add/Remove External Data Source page	
Main Flow	Step	Action
	1	Administrator enters Add/Remove External Data Source page
	2	System provides a page for administrator to configure the source of the data
	3	Administrator removes a source from the preference
	4	System updates the database and the source is no longer involved in UC#19
Extensions	Step	
	N/A	N/A

Figure 5.32 - UC17: Remove System Interface Module Use Case Table

UC-#18 : Send System Interface Module Signal		
Related Requirements	REQ-6, REQ-7	
Goal In Context	System sets an active value to control modules such as text message, email and computer-control modules to initiate them.	
Preconditions	System is connected to the internet and devices are connected to system	
Successful End Condition	System sets an active value to control specific modules	
Failed End Condition	System does not set an active value to control specific modules	
Actors	System, Device	
Trigger	UC#20 is successful and system detects the need of UC#18	
Main Flow	Step	Action
	1	UC#20 is successful and system detects the need of UC#18
	2	System sets an active value with respect to the module
	3	A device interface module converts the value into meaningful commands and initiates the corresponding modules
Extensions	Step	
	3.1	The corresponding module does not respond to the commands
	3.2a	System attempts to resend the command to the module
	3.2b	Upon failure, system checks if the internet connection is established. An error message on screen is generated.

Figure 5.33 - UC18: Send System Interface Module Signal Use Case Table

UC-#19 : Receive System Interface Module Signal		
Related Requirements	REQ-6, REQ-7	
Goal In Context	System receives data from internet	
Preconditions	Internet connection is established	
Successful End Condition	System receives data from internet	
Failed End Condition	System does not receive any data or incorrect data from internet	
Actors	System, Internet	
Trigger	UC#20 is successful and system detects the need to fetch data from internet for analysis. System requests data from internet in UC#18.	
Main Flow	Step	Action
	1	UC#20 is successful and system detects the need to fetch data from internet for analysis. System requests data from internet in UC#18.
	2	System receives data from internet and the data are saved to the database. Device interface module controls the poll rate and interfaces
Extensions	Step	
	2.1	System does not receive any data from a specific source.
	2.2	Administrator is prompted about this error and requested to check the data source in UC#16

Figure 5.34 - UC19: Receive System Interface Module Signal Use Case Table

UC-#20 : Evaluate Data Against Rule Sets		
Related Requirements	REQ-2, REQ-8	
Goal In Context	System analyzes data collected and data stored in condition set and rule sets	
Preconditions	Data is collected in UC#19 and UC#5 are successful	
Successful End Condition	System analyzes the data and stores it in the database	
Failed End Condition	System does not analyze the data nor the analyzed data is correct	
Actors	System	
Trigger	System received data from UC#15 and followed by UC#19	
Main Flow	Step	Action
	1	System received data from UC#15 and followed by UC#19
	2	System analyzes data base on the mathematical models and saves it into the database
Extensions	Step	
	N/A	N/A

Figure 5.35 - UC20: Evaluate Data Against Rule Sets Use Case Table

UC-#21 : Execute Action Set		
Related Requirements	REQ-3, REQ-6	
Goal In Context	System activates a set of devices at the same time	
Preconditions	UC#20 is successful and the need of UC#21 is detected	
Successful End Condition	System activates a set of devices at the same time	
Failed End Condition	System does not activate a set of device at the same time	
Actors	System, Device	
Trigger	UC#20 is successful and the need of UC#21 is detected	
Main Flow	Step	Action
	1	UC#20 is successful and the need of UC#21 is detected
	2	System sends instructions to the set of corresponding devices in UC#14
Extensions	Step	
	2.1	One or more devices are not initiated at the same time
	2.2	System prompts the administrator to check the condition sets and rule sets in UC#5 and the connections of the devices
	2.3	Administrator corrects the problem in UC#5 and the connection of the devices

Figure 5.36 - UC21: Execute Action Set Use Case Table

UC-#22 : Backup Database and Device Interface Modules		
Related Requirements	REQ-4, REQ-5	
Goal In Context	System performs backup and saves it to a specified media	
Preconditions	N/A	
Successful End Condition	System backup is saved to a specified media	
Failed End Condition	System backup fails	
Actors	Administrator, System	
Trigger	Administrator manually initiates backup or initiated by schedule	
Main Flow	Step	Action
	1	Administrator manually initiates backup or initiated by schedule
	2	System performs a full backup and saves the data to a specified media and generates a MD5 hash for it.
Extensions	Step	
	2.1	System fails to perform backup due to either the media is full or no media is connected
	2.2	Administrator is notified and requested to check the media
	2.3	Administrator corrects the error and proceed with the backup procedure

Figure 5.37 - UC22: Backup Database and Device Interface Modules Use Case Table

UC-#23 : Restore Database and Device Interface Modules		
Related Requirements	REQ-4, REQ-5	
Goal In Context	Administrator performs a full restore on the system	
Preconditions	System is malfunctioning	
Successful End Condition	System is restored	
Failed End Condition	System is not restored	
Actors	Administrator, System	
Trigger	Administrator enters Backup/Restore page and initiates restore procedure	
Main Flow	Step	Action
	1	Administrator enters Backup/Restore page and initiates restore procedure
	2	System verifies the backup image's MD5 hash and performs restore
Extensions	Step	
	2.1a	The backup image is non-functioning, corrupted or MD5 hash is different from that of the previous recovery record
	2.2a	System prompts the administrator about the error and requests for another backup image
	2.1b	Restore process fails
	2.2b	System performs roll-back to the previous state and prompts administrator about the error

Figure 5.38 - UC23: Restore Database and Device Interface Modules Use Case Table

f. Traceability Matrix

Use Cases	REQ1	REQ2	REQ3	REQ4	REQ5	REQ6	REQ7	REQ8	Max PW	Total PW
PW	2	5	10	4	7	6	5	3		
UC#1 AuthenticateUser	x				x				7	9
UC#2 CreateUser	x				x				7	9
UC#3 ChangeUserSetting	x				x				7	9
UC#4 ConfigPermissions			x		x				10	17
UC#5 ConfigRuleSets			x		x				10	17
UC#6 DisplayDevInfo					x				7	7
UC#7 ExportLog					x	x			7	13
UC#8 InstallDevMod					x		x		7	12
UC#9 UninstallDevMod					x		x		7	12
UC#10 AddIntArray					x		x		7	12
UC#11 RemoveIntArray					x		x		7	12
UC#12 AddDevice					x		x		7	12
UC#13 RemoveDevice					x		x		7	12
UC#14 SendDevSignal		x							5	5
UC#15 ReceiveDevSignal		x							5	5
UC#16 AddSysMod					x		x		7	12
UC#17 RemoveSysMod					x		x		7	12
UC#18 SendSysSig						x	x		6	11
UC#19 ReceiveSysSig						x	x		6	11
UC#20 EvalData		x						x	5	8
UC#21 ExecuteAction			x			x			10	16
UC#22 BackupDBDev				x	x				7	11
UC#23 RestoreDBDev				x	x				7	11

Figure 5.39 - Traceability Matrix Table

g. System Sequence Diagram

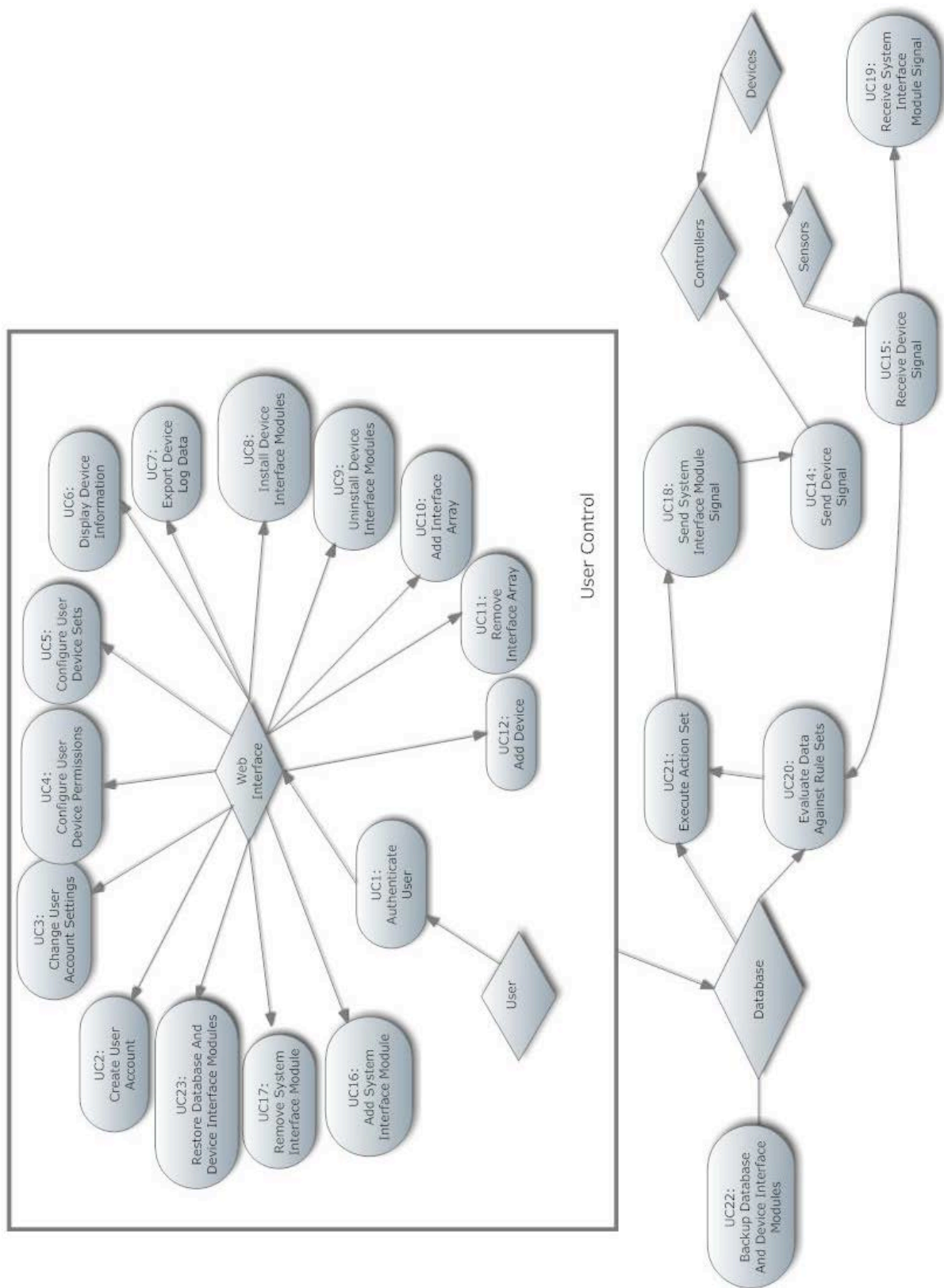


Figure 5.40 - System Sequence Table

h. Nonfunctional Requirements

With autoHome, we use the FURPS+ model [1] to describe the nonfunctional requirements as follows:

Functionality

See the Functional Requirements section for details on the functionality of autoHome.

Usability

The autoHome dashboard should be simplified and easy to maneuver, while also being easy on the eyes. This means that the dashboard should have an aesthetically pleasing look to it. This would allow it to be accessible to a wide range of consumers. The dashboard would have a panel with a list of all the rooms in the main menu, with a sub-menu of all the appliances and functions to each specific room. autoHome will support a simple “plug and play” system with all appliances in the home so that they will be easily added to the dashboard when they are installed in the home. The rest of the system should also be designed as intelligently by implementing other features that make the experience the easiest possible for the user.

Reliability

Data Safety

autoHome will create a routine backup in the case of system failure. The backups will be database dumps compatible with many other home automation systems after data manipulation. Backups will be made on a user-specified basis. All user settings, recorded data, and stored rulesets will be backed up. autoHome data and configuration only lives in the database, so a database dump is all that is required. An identical autoHome central server may be installed in the case of another autoHome central server's demise. The reconfiguration would be as simple as loading a database file. The autoHome central server will be installed on an enterprise-grade Uninterrupted Power Supply. This will provide the system long enough time to realize the power loss and back up all data before a potential power failure. The backup system will also be stored to a cloud based storage system. [3]



Device Resilience

Complicated devices are considered as a bundle of lesser devices to the central system. An automated refrigerator would have multiple sensors such as refrigerator temperature, freezer temperature, refrigerator door-open, and freezer door-open sensors. These are all considered different devices by the system as to provide a redundant set of controls such that if one sensor malfunctions, the other sensors may behave normally as they are treated as entirely separate devices.



It was considered to add individual devices (with multiple sensor values) to the system as wholes, but then decided against with the realization that it would greatly decrease the modularity and reliability of the system. Instead, individual sensors or controllers may be grouped logically to form the logical association of a whole complex device.

Performance

autoHome should be a very streamlined experience. autoHome system hardware will be tested to work proficiently with its software in

order to achieve an optimal experience. Thus, autoHome should perform smoothly and on time with the consumers' inputs. autoHome should be able to recognize all of the appliances in the home by means of the dashboard.

Supportability

Technical support should be available on our website via live chat, email, or phone call. autoHome should have commands that can be processed by either a smartphone, the autoHome website, or the system itself at home. autoHome should have a reboot system with a security code that can be accessed through technical support, in case the system malfunctions. autoHome should receive updates that should be "pushed" to the system through a wireless network to ensure that potential bugs are fixed.

± (Other criteria)

Security

autoHome should have passwords for each individual that uses the system. After keying in multiple invalid password entries, autoHome should lock up until the system generates an "override password" that will be sent to the user by an email or text notification.

Implementation

autoHome's Dashboard interface will be coded in PHP, MySQL, AJAX and jQuery [4]. autoHome Embedded Devices will use an Arduino [5] coded in C.

Operation

autoHome will be able to detect which appliance is attached to the system with the "plug and play" system. autoHome will have Climate control, inductive charging, ambient lighting and window blind control, bathroom automation, automatic sliding doors, scheduled controls, energy and water usage monitor, and fire/CO alarms. autoHome will also be able to detect home intrusion and alert the police with an alarm system, and it will alert the family members with an alarm and via text or email. For a detailed outline of operations, please refer to the "Statement of Requirements" section.

Packaging

Ideally, autoHome should be installed in new homes. However, autoHome can be installed into previously existing homes, but the same functionality is not guaranteed. In previously existing homes, autoHome will require many holes in the walls for the wiring of the system. Since some previously existing homes may be older than others, the overall conditions may not be optimal for the installation of autoHome. Connections and wiring would be handled by the professional installer of autoHome. The consumer would simply deal with the autoHome dashboard, which is detailed in the "Usability" section above.

Legal

The programming, design and ideas of this product will be provided by our own team. A license of this product will be obtained when autoHome is completed and ready to sell as a market product.

6. Effort Estimation

- 1) Log In
Respective Usecase: UC1
Data Entry: 2 keystrokes
Navigation: 3 clicks
 - Click in "username" field and enters user ID
 - Click in "password" field and enter password
 - Click the arrow button to commence login
- 2) Create a user account
Respective Usecase: UC2
Prerequisite: Log in (2 keystrokes and 3 clicks)
Data Entry: 3 keystrokes
Navigation: 6 - 7 clicks
 - Click on "Users"
 - Click on "New User"
 - Click in "username" field and enters user ID
 - Click in "password" field and enter password
 - Click on the check box to set user permission (Optional)
 - Click in "phone number" field and enter phone number
 - Click the arrow button to commence user creation
- 3) Change user settings
Respective Usecase: UC3
Prerequisite: Log in (2 keystrokes and 3 clicks)
Data Entry: 1 - 3 keystrokes
Navigation: 3 – 7 clicks
 - Click on "Users"
 - Click on "Edit"
 - Click in "username" field and enters user ID (Optional)
 - Click in "password" field and enter password (Optional)
 - Click on the check box to set user permission (Optional)
 - Click in "phone number" field and enter phone number (Optional)
 - Click the arrow button to commence user creation
 - Click on "Cancel" to cancel operation (Optional)
- 4) Delete user
Prerequisite: Log in (2 keystrokes and 3 clicks)
Navigation: 2 clicks
 - Click on "Users"
 - Click on "Delete"

- 5) Configure device permission
Respective Usecase: UC4
Prerequisite: Log in (2 keystrokes and 3 clicks)
Data Entry: 0 – 2 keystrokes
Navigation: 3 – 10 clicks
- Click on “Devices”
 - Click on “Edit”
 - Click on “Name” field and enter name (Optional)
 - Click on “Device type” drop-down list and click on respective device type of the device (Optional)
 - Click on “Location” drop-down list and click on respective location of the device (Optional)
 - Click on “Active” check box (Optional)
 - Click on “Default value” field and enter value (Optional)
 - Click on “Update Device” to update the new settings
- 5) Set up rule sets
Respective Usecase: UC5
Prerequisite: Log in (2 keystrokes and 3 clicks)
Data Entry: 2 keystrokes
Navigation: 10 – 12 clicks
- Click on “Configure Rules”
 - Click on “New Rule Set”
 - Click on “Name” field and enter rule name
 - Click on “Condition set” drop-down list and click on a condition
 - Click on “Action set” drop-down list and click on an action set
 - Click on “User” drop-down list and click on user name
 - Click on “Description” field and enters description (Optional)
 - Click on “Active” check box to set the rule to be active (Optional)
 - Click on “Create Rule Set” to commence creation
- 6) Display device information
Respective Usecase: UC6
Prerequisite: Log in (2 keystrokes and 3 clicks)
Navigation: 2 clicks
- Click on “Devices”
 - Click on “Show” or the device’s name
- 7) Install device interface module
Respective Usecase: UC8
Prerequisite: Log in (2 keystrokes and 3 clicks), device driver installation
Data Entry: 3 keystrokes
Navigation: 10 clicks
- Click on “Device Types”
 - Click on “New device type”
 - Click on “Name” field and enters device type name
 - Click on “Module name” field and enters module name
 - Click on “Data type” drop-down list and click on respective type
 - Click on “Data flow” drop-down list and click on respective data
 - Click on “Unit” field and enters unit
 - “Click on “Create Device type” to commence creation

- 8) Uninstall device interface module
Respective Usecase: UC9
Prerequisite: Log in (2 keystrokes and 3 clicks), device driver uninstallation
Navigation: 2 clicks
 - Click on "Device Types"
 - Click on "Remove"
- 9) Add device
Respective Usecase: UC12
Prerequisite: Log in (2 keystrokes and 3 clicks), physical device connection
Data Entry: 2 keystrokes
Navigation: 9 clicks
 - Click on "Devices"
 - Click on "Name" field and enter device name
 - Click on "Device type" drop-down list and click on respective type
 - Click on "Room" drop-down list and click on respective location
 - Click on "Active" check box to set the device to active
 - Click on "Default Value" field to set default value of the device
 - Click on "Add device" to commence creation
- 10) Remove device
Respective Usecase: UC13
Prerequisite: Log in (2 keystrokes and 3 clicks), physical device disconnection
Navigation: 2 clicks
 - Click on "Devices"
 - Click on "Remove"
- 11) Add a new system interface module from internet source
Respective Usecase: UC16
Prerequisite: Log in (2 keystrokes and 3 clicks)
Data Entry: 1 keystroke
Navigation: 4 clicks
 - Click on "Update"
 - Click on "Add source"
 - Click on "Source" field and enters a link
 - Click on "Enter" to create a new source
- 12) Remove a system interface module
Respective Usecase: UC17
Prerequisite: Log in (2 keystrokes and 3 clicks)
Navigation: 2 clicks
 - Click on "Update"
 - Click on "Remove source"
- 13) Back up database
Respective Usecase: UC22
Prerequisite: Log in (2 keystrokes and 3 clicks), source connected
Navigation: 3 clicks
 - Click on "Backup / Restore"
 - Click on one of the radio buttons to choose a source
 - Click on "Next"

- 14) Restore database
Respective Usecase: UC23
Prerequisite: Log in (2 keystrokes and 3 clicks), source connected
Navigation: 4 clicks
- Click on "Backup / Restore"
 - Click on one of the radio buttons to choose a source
 - Click on "Next"
 - Click on "Confirm" or "Cancel"
- 15) Check Logs and Alerts
Respective Usecase: UC7
Prerequisite: Log in (2 keystrokes and 3 clicks)
Navigation: 1 – 2 clicks
- Click on "Logs and Alerts"
 - Click on "Log or "Alert" tab to view different messages (Optional)
- 16) Add Room
Respective Usecase: N/A
Prerequisite: Log in (2 keystrokes and 3 clicks)
Data Entry: 3 keystrokes
Navigation: 6 clicks
- Click on "Rooms & Locations"
 - Click on "New room"
 - Click on "Name" field to enter name of the room
 - Click on "Floor" field to enter floor number
 - Click on "Description" field to enter description of the room
 - Click on "Create Room" to commence room creation
- 17) Remove Room
Respective Usecase: N/A
Prerequisite: Log in (2 keystrokes and 3 clicks)
Navigation:
- Click on "Rooms & Locations"
 - Click on "Remove"
- 18) Add Location
Respective Usecase: N/A
Prerequisite: Log in (2 keystrokes and 3 clicks)
Data Entry: 2 keystrokes
Navigation: 7 clicks
- Click on "Rooms & Locations"
 - Click on "new location"
 - Click on "Name" field to enter name of the location
 - Click on "Room" drop-down list to choose respective room
 - Click on "Description" field to enter description of the location
 - Click on "Create Location" to commence location creation
- 19) Remove Location
Respective Usecase: N/A
Prerequisite: Log in (2 keystrokes and 3 clicks)
Navigation:
- Click on "Rooms & Locations"
 - Click on "Remove"

7. Domain Analysis

a. Domain Model

Standardization

Home automation has many moving parts and one of the main challenges is to standardize all the different equipment being used to monitor and control household appliances. Devices controlled or monitored may include electronic ovens, fire alarms, ajar door sensors, individual power meters, motion sensors, and luminescence detectors. Most, if not all, of the useful sensors that this project plans to utilize will send or process a single data value. This makes it very convenient to lay out a generic way to process the inputs from, or outputs to these devices. Each device can then be reduced to an object bearing a single data value to report to the system, or an object receiving a single data value from the system to govern the device's actions, or both. These standardized objects will be the device nodes as shown in the diagram. The data values obtained from these devices can then be categorized into analog, digital, or none (if the device hasn't a value to report, or cannot be controlled).



Sensors

Alarm-type (Digital) Sensors

These devices have two states. We will denote them in our application as logical 0 (off) or logical 1 (on). They can be either active or inactive. A fire alarm will be either blaring due to hazardous conditions, or not.

A window-mounted burglar alarm will show if the window is open or not. A simple light detector will be active if it receives light above a certain intensity threshold. A carbon-monoxide alarm will be active only if the carbon-monoxide levels are above a certain threshold. The data collected from these devices can be put into a meaningful graph that will look like a digital signal. Users would be able to determine exactly when a fire alarm went off or what time windows were open at and for how long.

Reading-type (Analog) Sensors

If required to track values over a period of time, reading-type sensors with analog-reporting capabilities are required. Power sensors would report a varying signal value correlated to the use of electricity. Complex light intensity sensors would give numbers correlating to how intense light is shining on them. Water usage sensors might give values indicating the electromagnetic disturbance varying volumes of flowing water might have on them. Thermometer devices give the current temperature. The values obtained are then logged according to specific device standards to produce meaningful and useful graphs and data sets.

Temperature Sensor



Controllers

On/off-type (Digital) Controllers

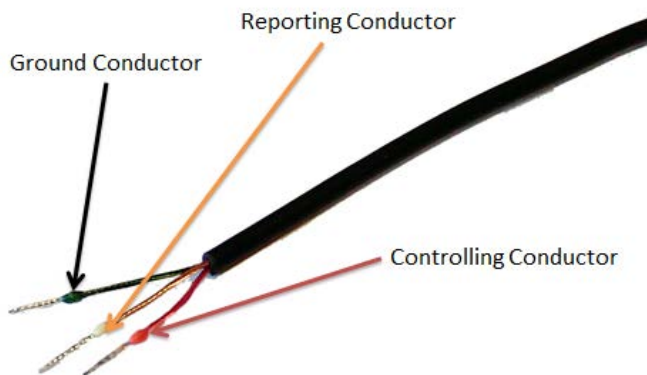
Digital controllers will send commands to devices with only two states. Again, these states will be represented as logical 0 (off) and logical 1 (on). Remotely controlled light-switches would only have an on or off state. Controlling whether a power outlet receives power only involves two states--off or on. Unlocking or locking a door would also deal with only two states.

Adjustable-type (Analog) Controllers

Analog controllers send commands to devices and govern their behavior by means of a meaningful value. A temperature control system would receive an analog value specifying what temperature to keep a certain room at. A speaker system could receive volume controls to adjust the sound output. Dimmer-light switches might receive values indicating how much to dim the lighting by. With these fine-tuned controls, a user could, for example, set up rules to control the intensity of the built-in interior lights according to the levels of light already in the room from the sun or other sources.

Physical (Layer 1) Device Communication

Although the home automation system is geared towards new homes being built (as retrofitting wiring is quite laborious), some methods are introduced to reduce the wiring requirements of new homes, and to ease the transition of older homes into automation.



The physical interaction of these devices will consist of three different types of conductors (wires). The first (required) conductor is the relative ground of the system and the device. The second type of conductor acts as a data channel that the device will use to report either digital or analog values to the central system. The third type of conductor acts as a central system-controlled channel to relay commands (in either analog or digital encoding) to the device. All devices will have a ground. They may have one or many of any of the combinations of the other conductors.

Device Interface Array and Device Interface Modules

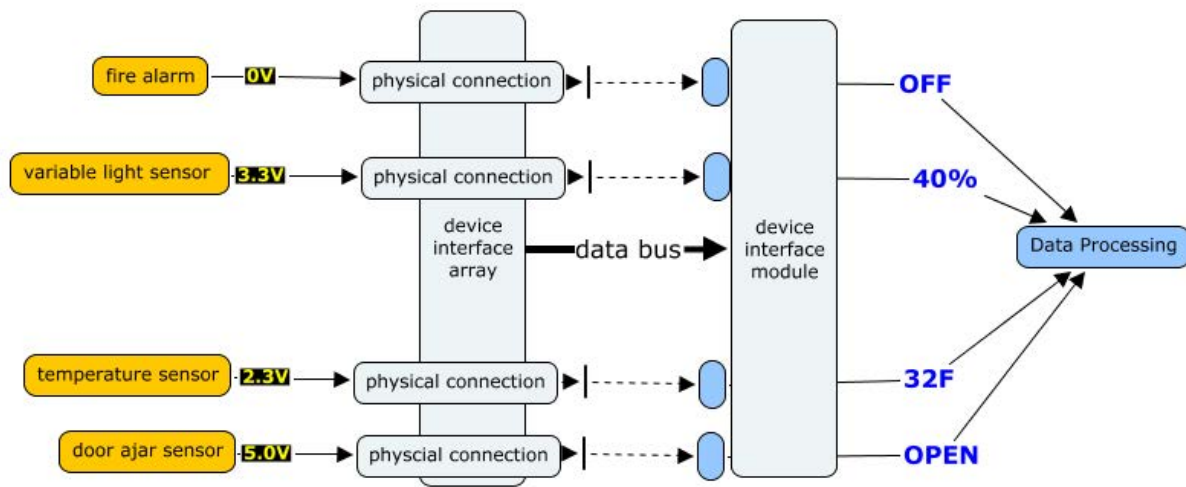


Figure 7.1 - Device Interface Modules and Array Diagram

System Devices

- The autoHome system comes with software-based sensor devices that report useful information such as time and date.
- There are also software-based controller devices that can use the Internet to send text messages, make phone calls, check server status, and many other useful things.

Ruleset

A home is hardly automated if it does not intelligently follow a set of rules configurable by the home owner. We introduce two rulesets used to govern the collecting of data, and the execution of commands to devices. By having a generic way to add conditions that will result in actions, the user is granted extreme flexibility in deciding exactly what they want their home to do.

Actions are simple commands that relay commands to control individual values. Each action has a respective controller and value defined by the user.

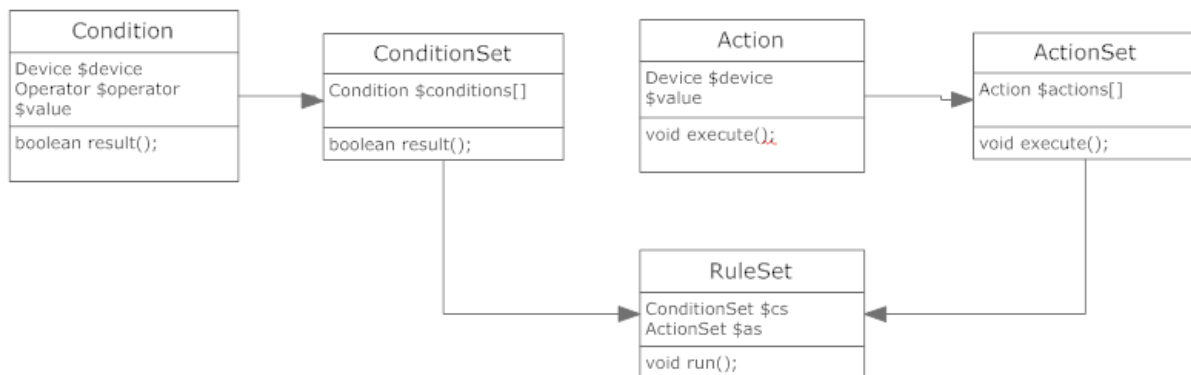


Figure 7.2 - Ruleset Diagram

Examples of Actions:

1. Dim "Living Room Light" to 50%
2. Lock "Basement-Outside Door"
3. Set "A/C System" to 50.00 Degrees Fahrenheit
4. Log "Temperature Sensor" Value

Actions are chained together to form action sets--sets of events that execute quickly enough in sequence as to be considered instantaneous..

Example Action-Set:

{Action #1, Action #2} => Dim "Living Room Light" to 50%, Lock "Basement Door"

The execution of action sets are governed by condition sets. Condition sets are made of conditions, which are simple variable comparisons to sensor values.

Examples of Conditions:

1. "Bathroom Light Sensor" > 60%
2. "Front Door Status" == Open
3. "Fire Alarm Status" == Active
4. "Time of Day" == 08:23
5. "Time of Day" > 06:30
6. "Time of Day" < 23:30
7. "Date" == 2012-02-17

Conditions must have all events evaluated as true to execute an associated action set.

For example, in the following condition set and action set relation:

{Condition #5, Condition #6} => {Action #1, Action #2}

As long as conditions 5 and 6 both remain satisfied, actions 1 and 2 will be continuously held.

I.E. as long as it is between 6:30AM and 11:30PM, the living room light will be dimmed to 50% and the basement-outside door will be locked.

Allowing users to define their own rules gives many possibilities for complex and useful rulesets. A user could, for example, keep porch lights on during the evening, only on weekends.

b. System Operation Contracts

Operation	Poll Sensor Data
Preconditions	<ul style="list-style-type: none"> • Device Interface Module thread for reading sensor device running • Device-specific wait-time (specified in Device Interface Module) elapsed
Post-conditions	<ul style="list-style-type: none"> • Push updated data to device objects • Drive all RuleSets

Operation	Set Controller Value
Preconditions	<ul style="list-style-type: none"> • Device Interface Module thread for setting controller device running • Device object set function called
Post-conditions	<ul style="list-style-type: none"> • Lock Device Interface Module thread to value.

Operation	Execute ActionSet
Preconditions	<ul style="list-style-type: none"> • New data updated in device • Relevant ConditionSet successful
Post-conditions	<ul style="list-style-type: none"> • Actions in ActionSet executed.

Operation	Database Backup
Preconditions	<ul style="list-style-type: none"> • User-specified backup time has elapsed
Post-conditions	<ul style="list-style-type: none"> • System dumps databases and module code into compressed tarball. • Tarball is backed up at a different location.

Figure 7.3 - System Operation Contracts Table

c. Mathematical Model

The autoHome software contains a few mathematical models as listed below:

- Electricity Usage [11]
 - A database containing common power usage of most common home appliances will be used for an estimate
 - **Power(Watts) = Volts(V) * Amperes(A)**
 - The amount of time that a device is turned on will be measured in seconds
 - **Work(Joules) = Power(Watts) * Time(s)**
 - Cost of energy will be calculated
 - **Total Cost(\$) = Work * (\$ per joules)**
- Water Usage [12]
 - Flow meters will be installed on showers, sinks, outdoor hoses, and toilets
 - **Flow = Gallons(gal) * Time(min)**
 - Cost of water will be calculated
 - **Total Cost(\$) = Flow * (\$ per gallon)**
- Inductive Charging [13]
 - The voltage transferred to the device we want to charge is

$$V_s = \frac{N_s}{N_p} V_p$$

V_s = voltage to the device being charged

$N_1 = N_2$ = the number of copper coils

V_p = the voltage from the source

- The current is calculated as

$$I_s = \frac{N_p}{N_s} I_p$$

I_s = current to the device being charged

$N_1 = N_2$ = the number of copper coils

I_p = the current from the source

- The device power can then be calculated as described from “Electricity Usage”

8. Interaction Diagrams

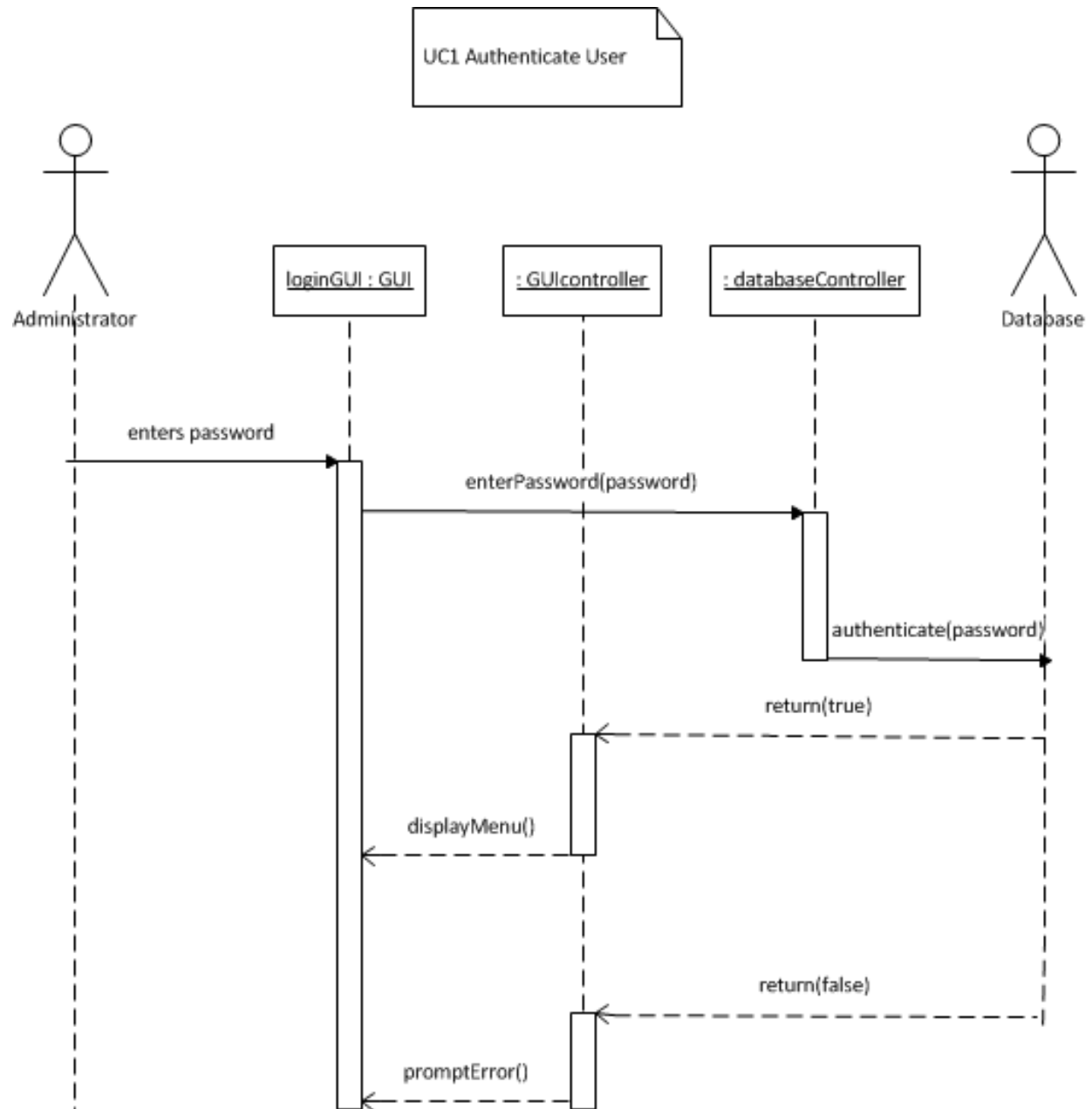


Figure 8.1 - UC-1 - Authenticate User

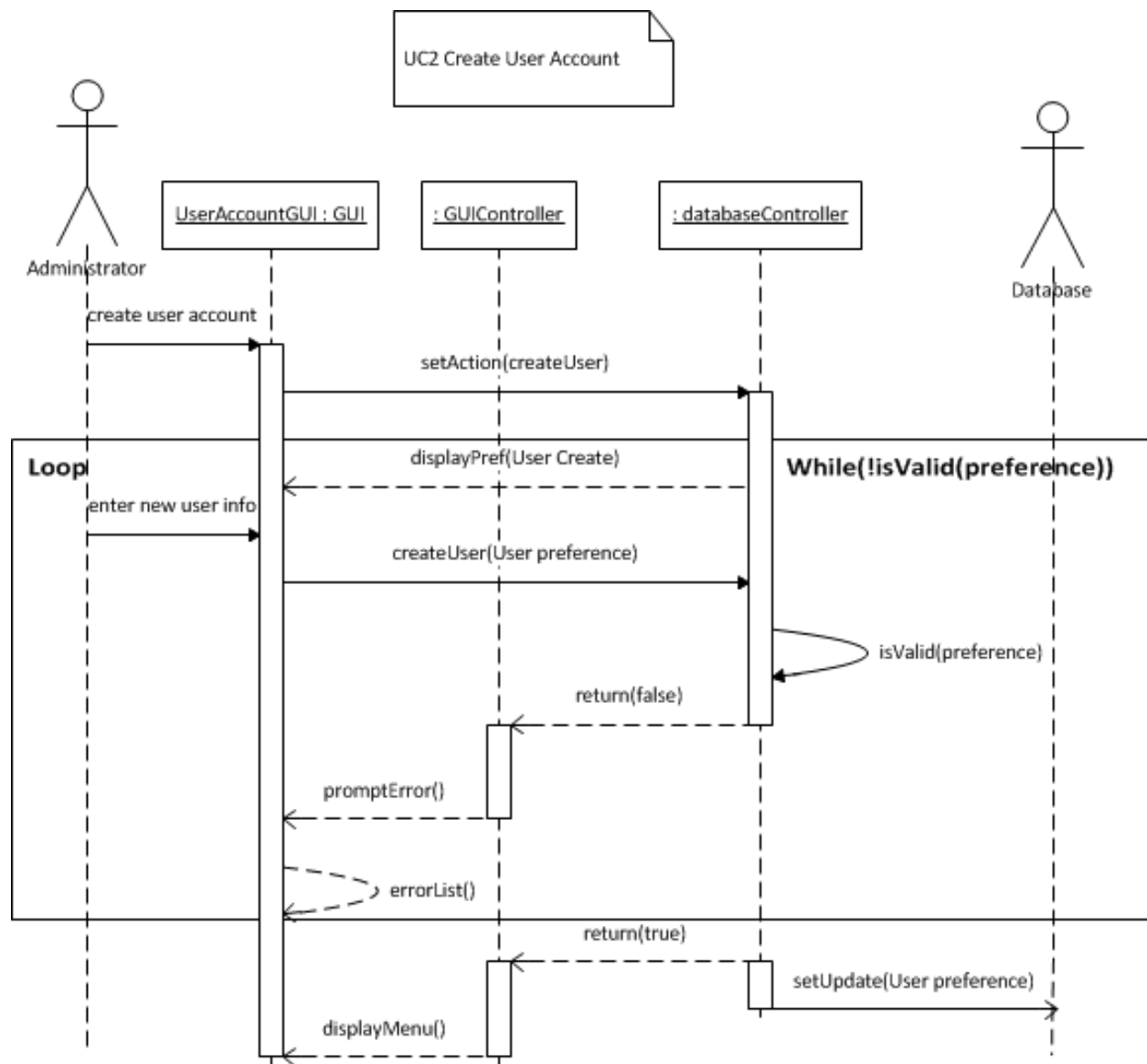


Figure 8.2 - UC-2 - Create User Account

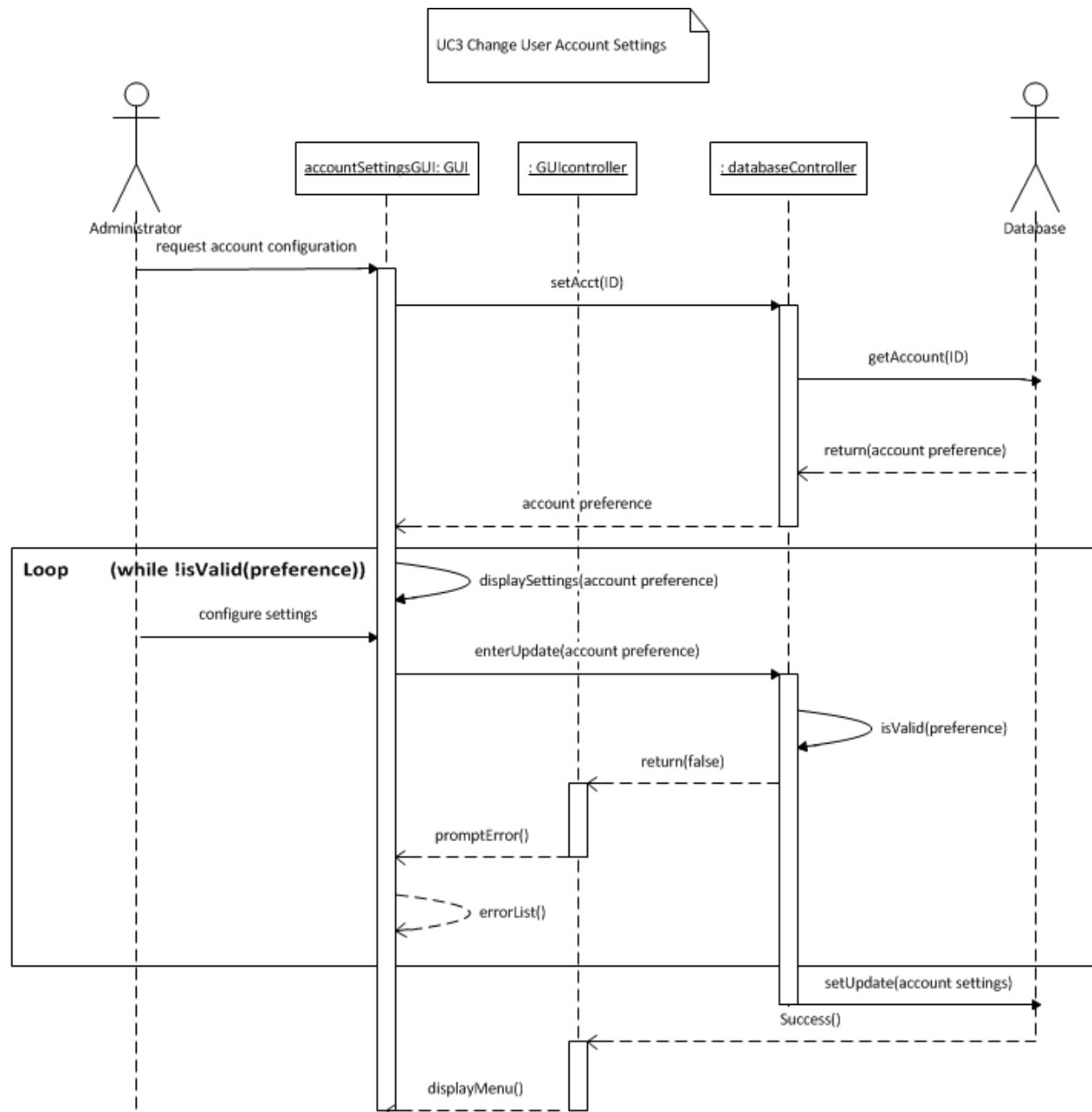


Figure 8.3 - UC-3 - Change User Account Settings

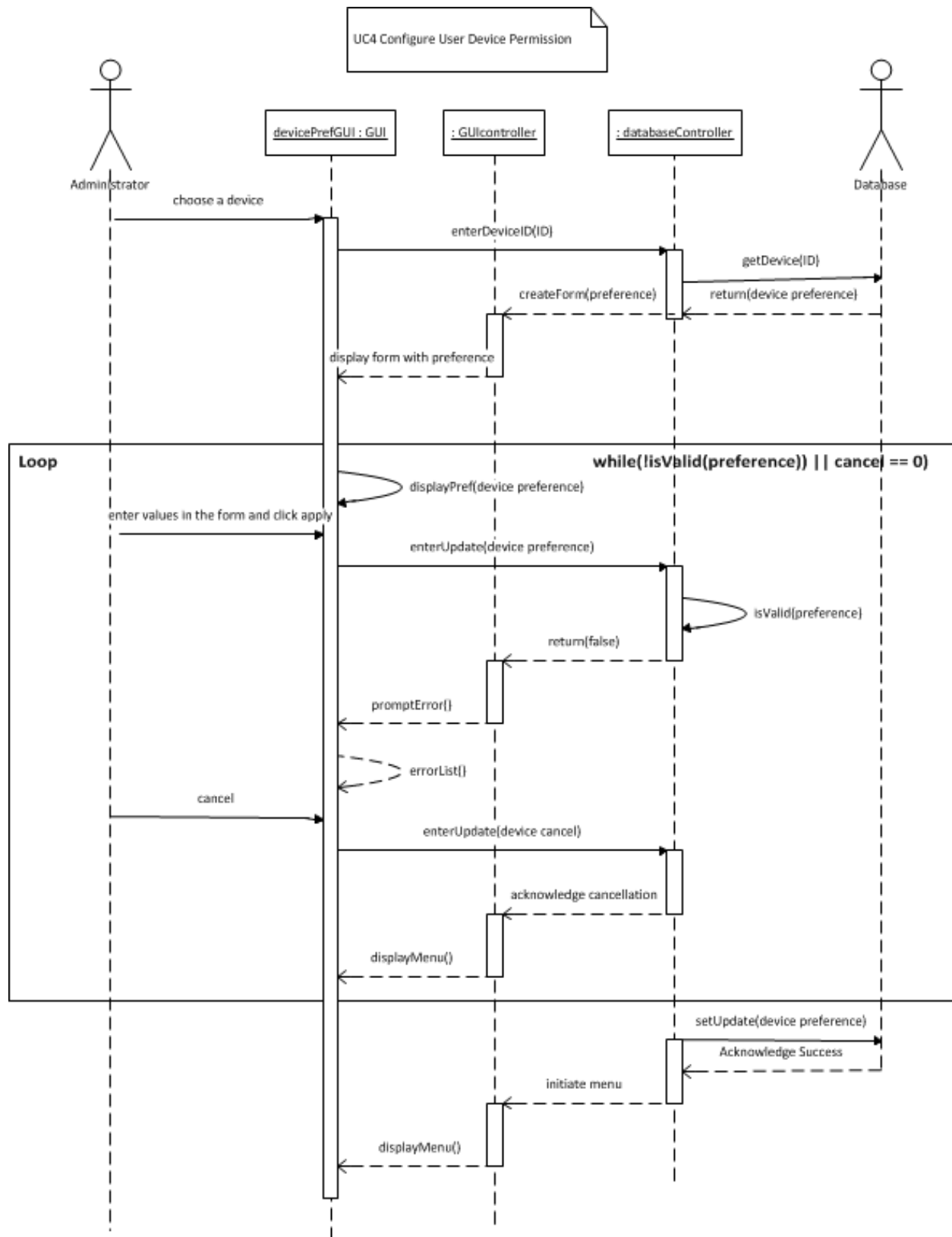


Figure 8.4 - UC-4 - Configure User Device Permission

Use Case 4: Configure User Device Permissions

Use Case 4 is designed with the intention of allowing the Administrator (in a typical setting, this would be a parent or owner of the system) to allow or deny access to devices connected to the system by users registered to the system (such as a child). This can be important for example if a parent wants to restrict children from turning on the oven while the parent is out of the house. The interaction diagram displays the general process that an administrator uses to make a change to available devices. It is assumed that the administrator is already on the device preference page. A module is loaded on the device information page that shows the current status and features of a particular device. The administrator is able through use of a checkbox list to select which user roles can and cannot access the device. The setting is then saved into the database when the form is submitted. Error controls are built into the system such as not allowing an administrator to get locked out from using a device if it has control over all devices. Additionally, any user that enters a device setting page with a role that has been declared by the administrator to not have access to the device, cannot change these permissions as well.

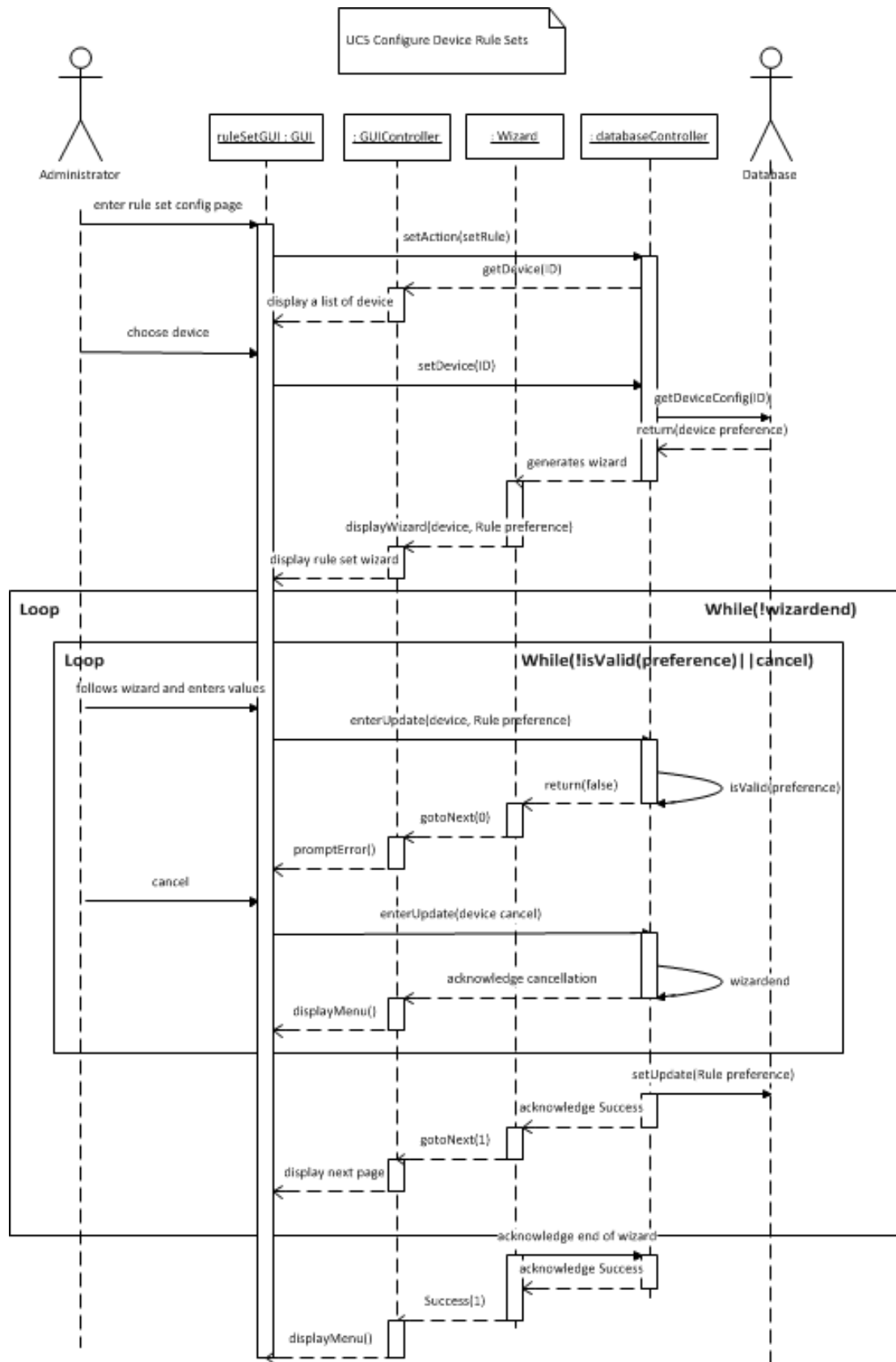


Figure 8.5 - UC-5 - Configure Device Rule Sets

Use Case 5: Configure Device Rule Sets

Use Case 5 allows users and administrators to configure RuleSets, which are the core-functionality of the autoHome system. The image below is an example of what the user will see when configuring a RuleSet. RuleSets consist of a set of Conditions (Condition Sets) and a set of corresponding Actions (Action Sets) to be executed when all conditions in the Condition Set are met. The list of available Conditions to be added into a ConditionSet can be seen on left, and the list of available Actions for an ActionSet can be seen on right. Both lists have their members selectable and editable.

The screenshot shows the 'autohome' web interface. The top navigation bar includes a search bar, a home icon, and the text 'Configure Rules'. The user is logged in as 'Administrator'. The sidebar on the left contains links to Dashboard, Devices, Device Types, Rooms & Locations, Configure Rules (highlighted), Users, Logs & Alerts (with a red badge showing '2'), and Backup / Restore.

The main content area is titled 'Configure Rules' and features a 'List of Rule Sets' table. The table has columns for Name, Condition set, Action set, User, Description, and Active. It lists four rule sets: 'Keep Kitchen Lit', 'Living Room Humidity Control', 'Living Room Temperature Control', and 'Master Bedroom Light Control'. Each row includes links for Show, Edit, and Destroy.

Below the table, there are two sections: 'Listing condition_sets' and 'Listing action_sets'. Each section has a table with columns for Name, Description, and User, listing various conditions and actions available for configuration.

Figure 8.5.1 - Configure Rules Page on Web Interface

As seen above, Conditions are simple comparisons of the status of sensors. A user may select a condition as “the bathroom light is off” or “the back door is open.” When these two Conditions are put into a ConditionSet together, both of the Conditions must be met for the ConditionSet to be true. The system provides internal Conditions that need not compare any physical devices. These are received as Virtual Device Signals instead of Device Signals and may include information such as Time, weather in another city, CPU Load of another computer, or even user triggers from mobile applications.

Actions are similar to Conditions, in that they usually use a value to command something. Examples of actions include “lock the back door,” “turn off the bathroom light,” “set the porch light to 80%.” Actions are chained together in Action Sets, in which all actions are executed together when called. The system may provide Actions that do not correspond to any physical device. These are sent through Virtual Device Signals, which actually run system code instead, sending text messages, making phone calls, or updating websites. Actions can occur from 100ms (locking a door) to infinite time (keeping a light on).

To edit either Actions or Conditions, the user selects “Edit” for either screen and they are then brought to another screen with an available device list displaying all the relevant devices that the user has permissions for.

On another screen, the Conditions may be formed into a ConditionSet and Actions may be formed into an ActionSet. A ConditionSet is then linked to a RuleSet to provide a list of actions to execute when a list of conditions are met. This provides extreme flexibility, allowing users a simple RuleSet as turning the lights off when it is after 6AM and before 8PM, or a complex RuleSet as sending a text message to the owner only when there is no motion detected in the kitchen and the oven has reached 400 degrees F. The possibilities are endless.

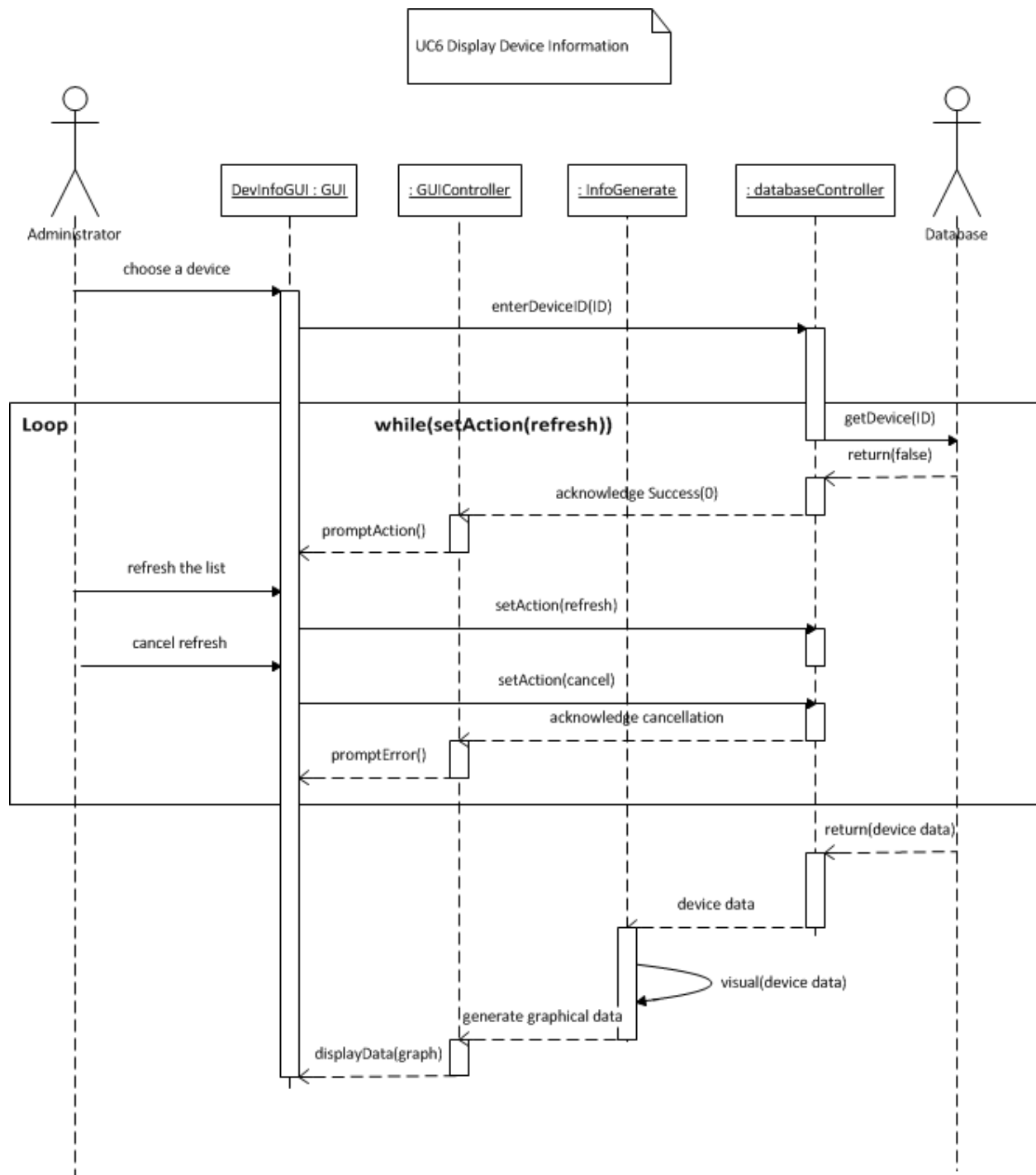


Figure 8.6 - UC-6 - Display Device Information

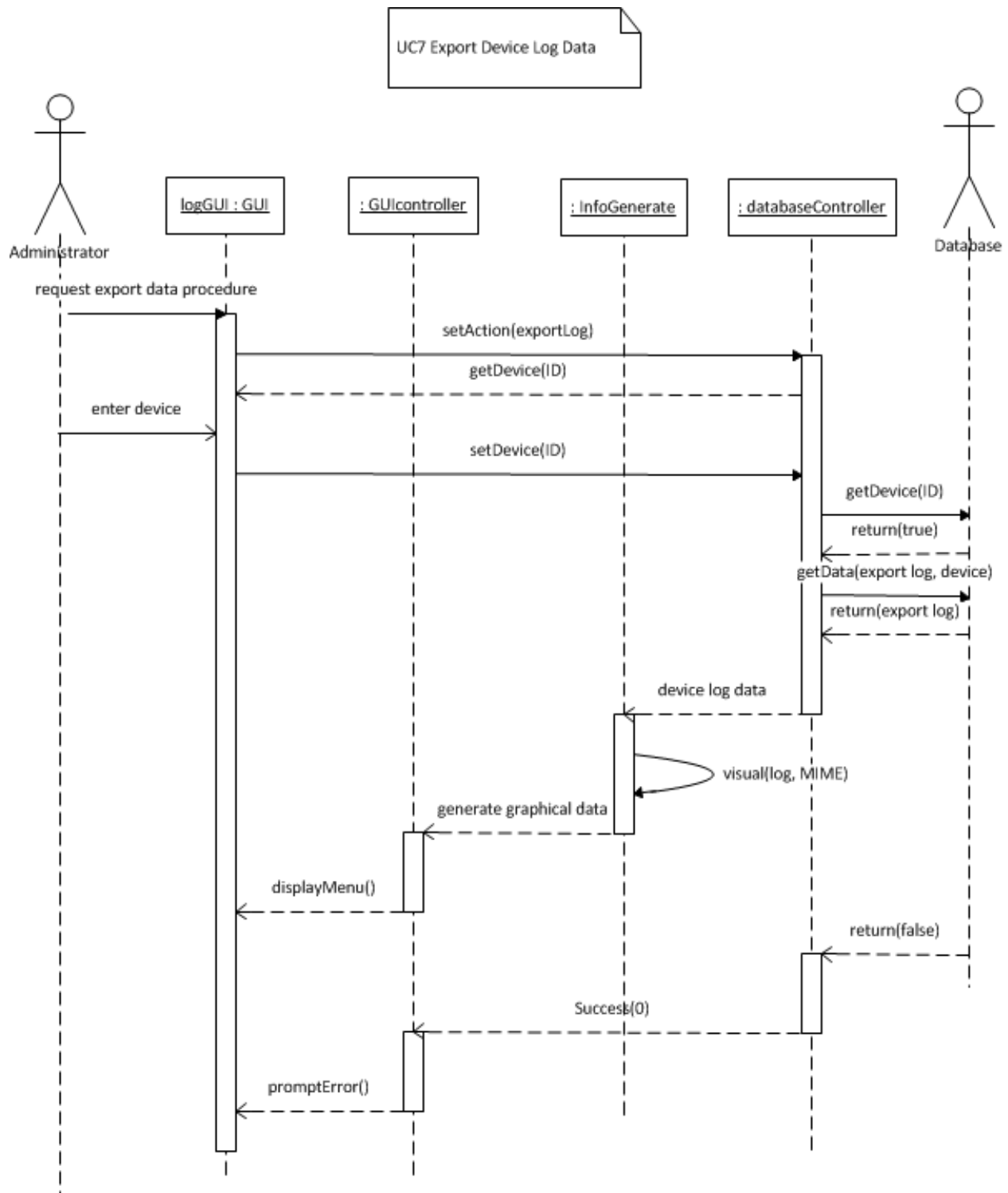


Figure 8.7 - UC-7 - Export Device Log Data

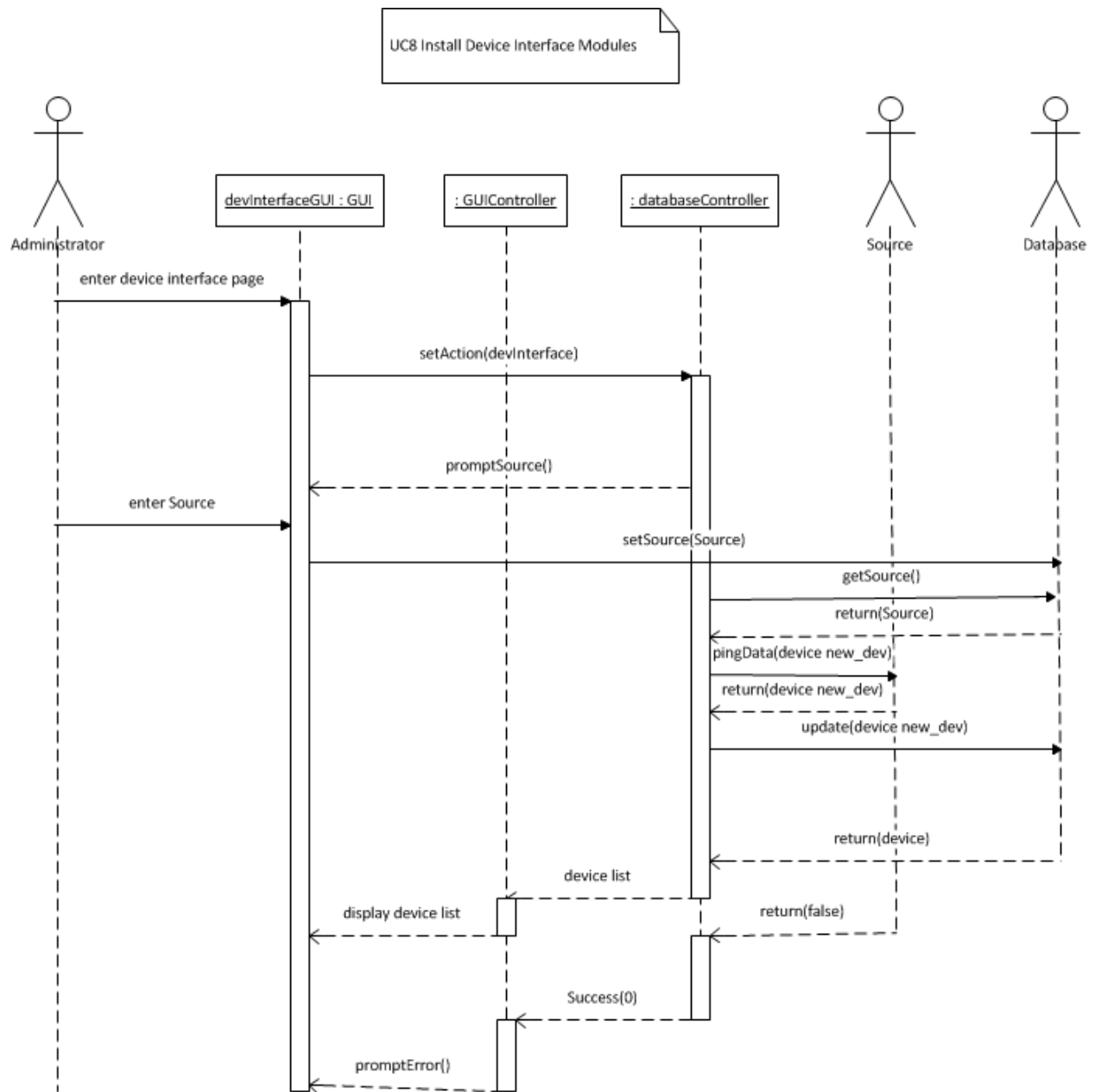


Figure 8.8 - UC-8 - Install Device Interface Module

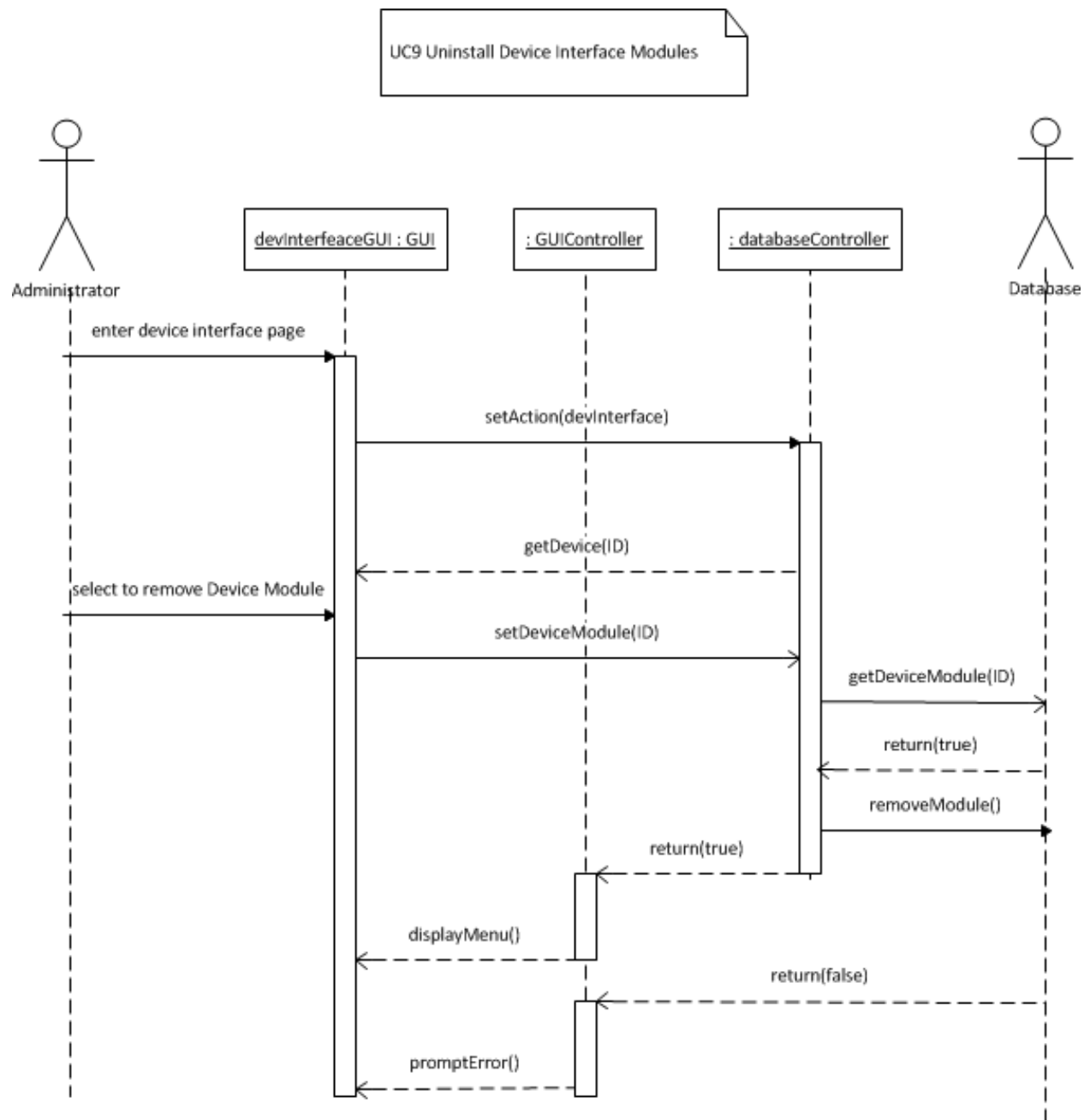


Figure 8.9 - UC-9 - Uninstall Device Interface Modules

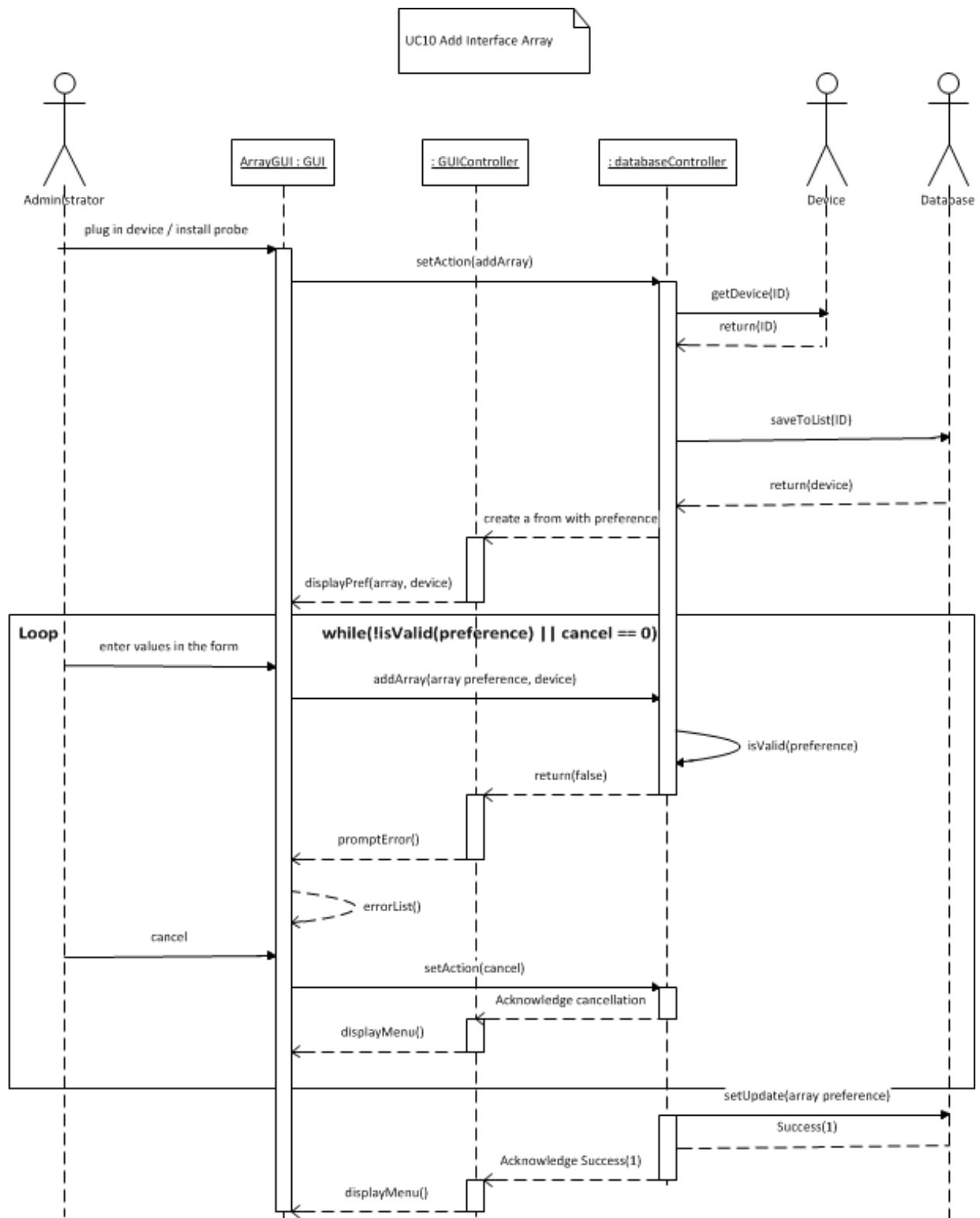


Figure 8.10 - UC-10 - Add Interface Array

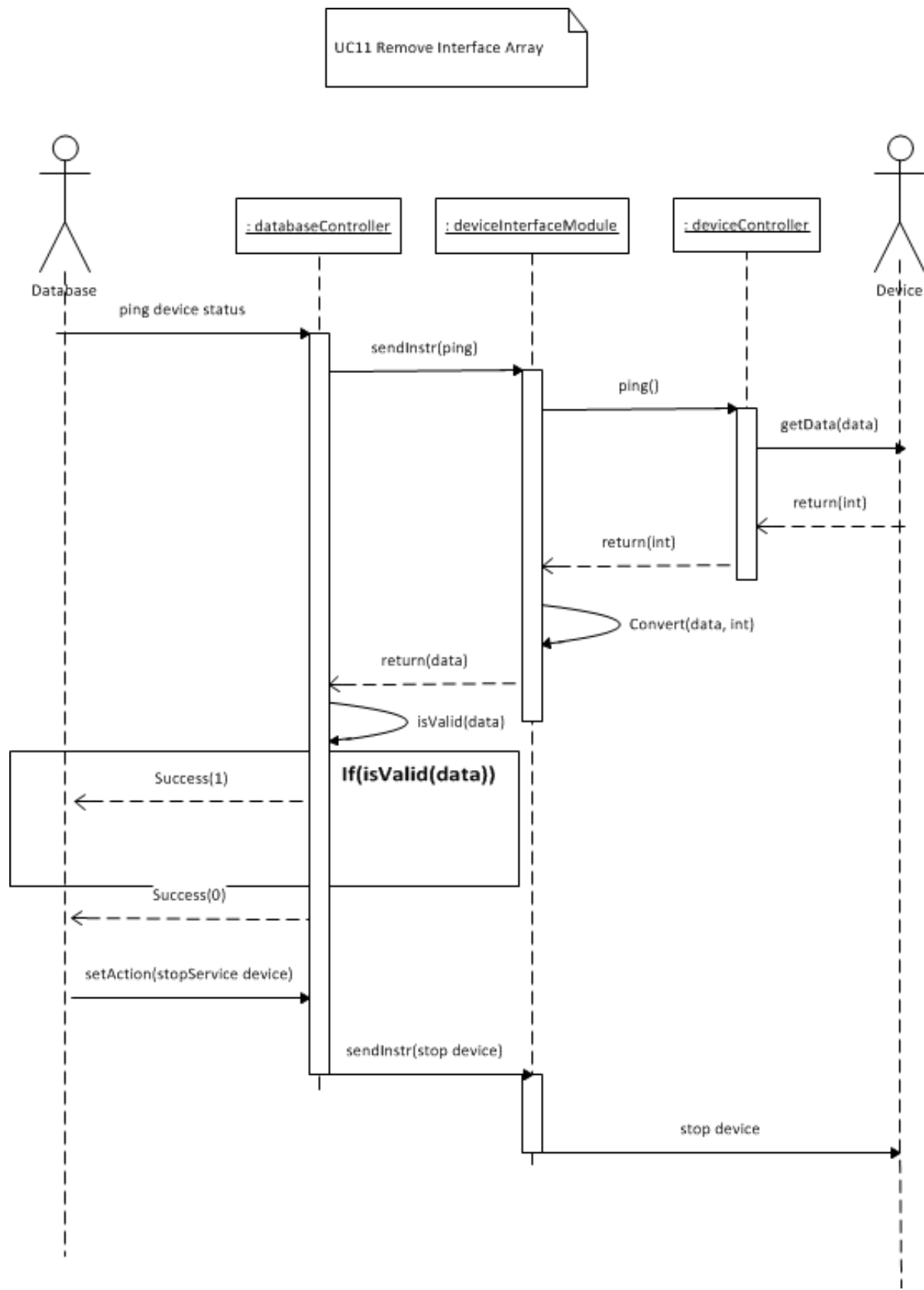


Figure 8.11 - UC-11 - Remove Interface Array

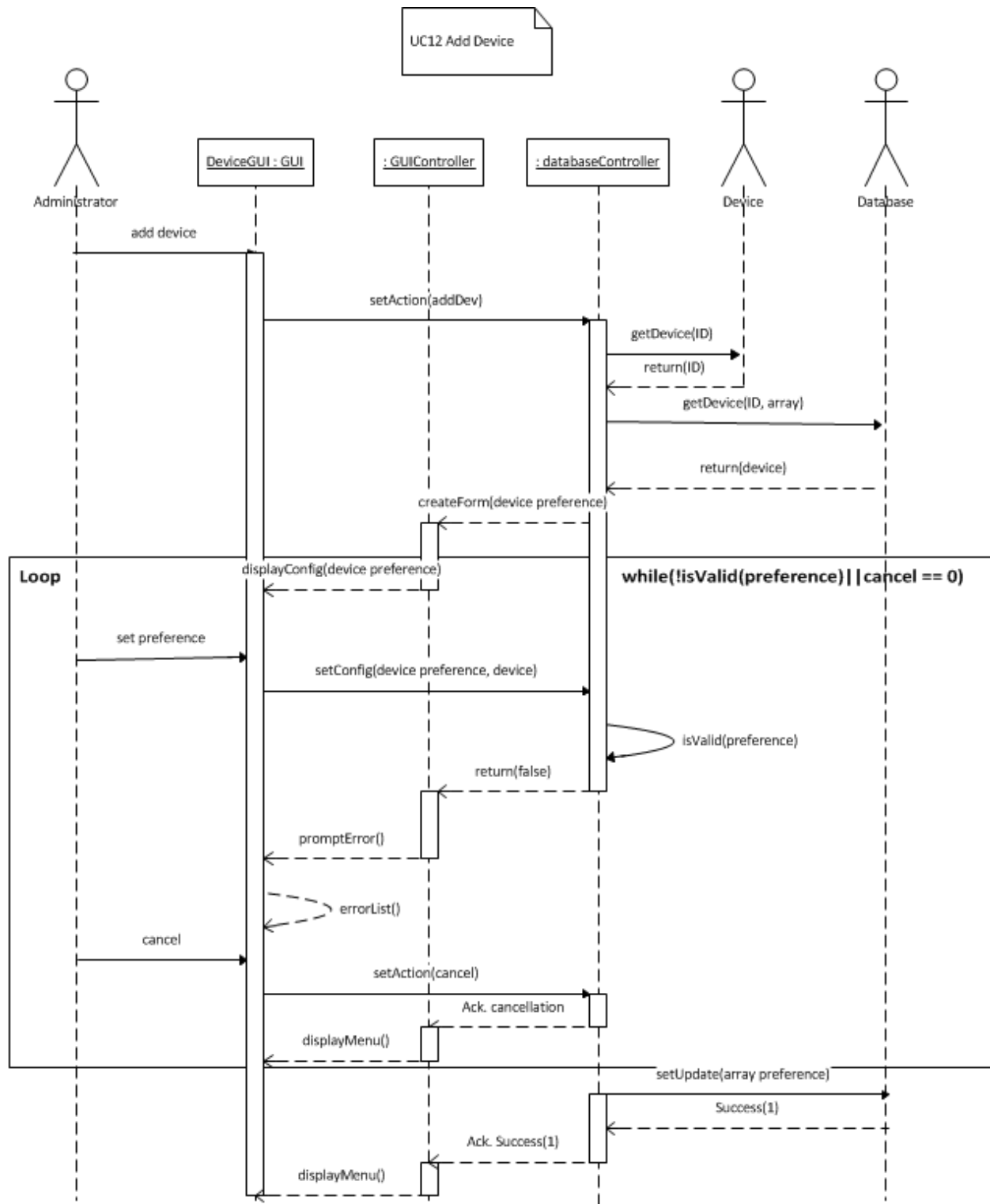


Figure 8.12 - UC-12 - Add Device

Use Case 12: Add Device

Use Case 12 is designed with the intention of allowing an administrator to add a new device into the auto-Home system, thus allowing it to becoming accessible to other devices and the system as a whole. In order to facilitate this, the user must connect the device to the system and install the device physically in its appropriate position in the home such as “a lamp in the master bedroom.” Once this step is completed the administrator will proceed to go to the add device module on the web based control panel and tell the system what type of device it is (electrical, security, temperature, etc...) and the location of where the device is placed as well. Once completed and submitted, the information is saved to the database for the system to keep track of. Additionally, the add device panel will then become the device preference panel where regular settings for specific devices can be set such as on and off and more as well as device permissions as discussed in UC 4. If a preference selection is invalid the administrator will be prompted to correct the issue.

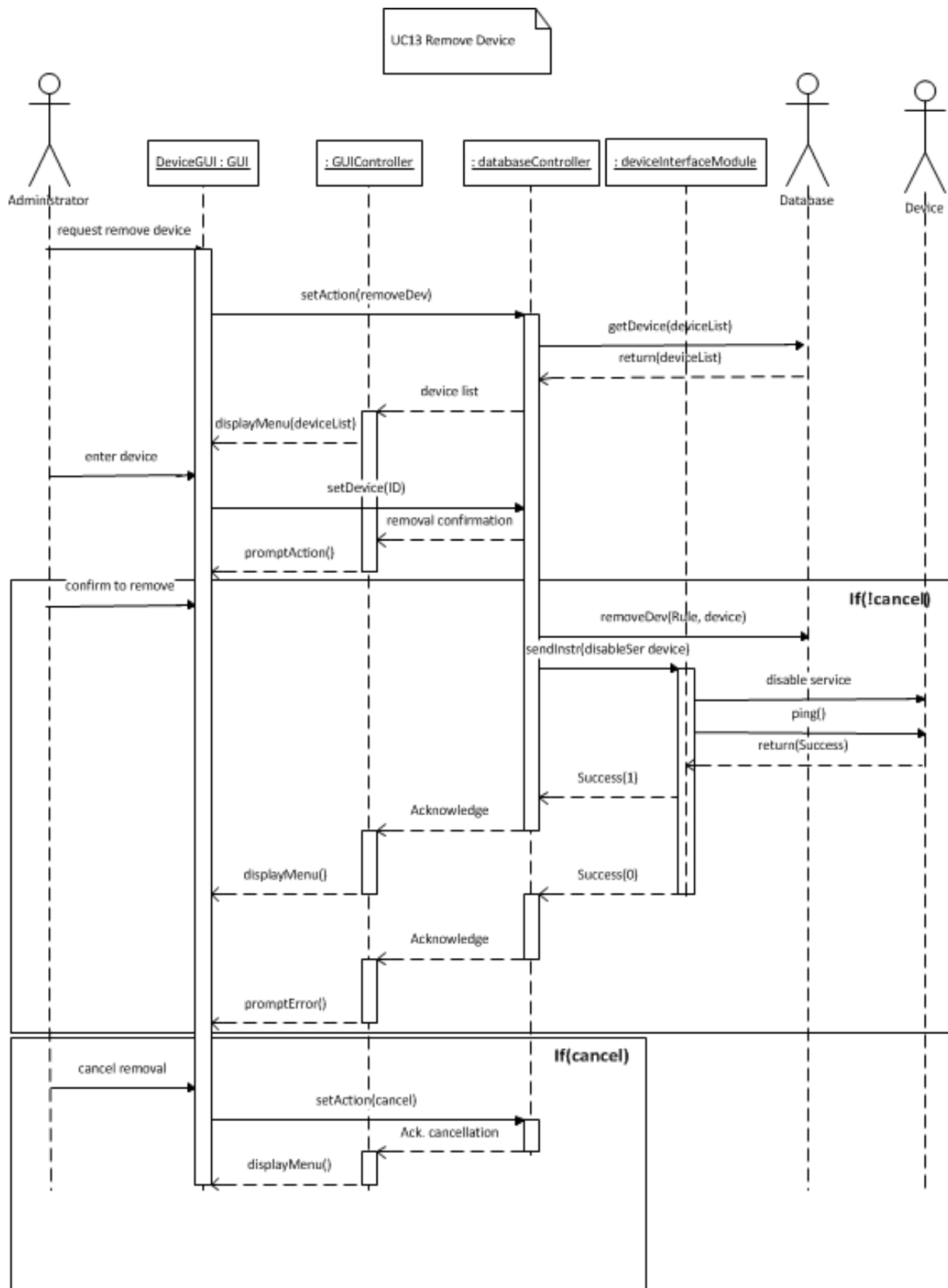


Figure 8.13 - UC-13 - Remove Device

Use Case: 13 Remove Device

Use Case 13 is designed with the intention of safely removing a device from the system without corrupting data and protecting the integrity of the device by assuming that all devices may be fragile as to the point that a sudden disconnect may damage a component within the device and or should also be shut down appropriately before removal to ensure that all data communication between the device and system has terminated. In order to remove a device, an administration will go to the device preferences page of the device the administrator wishes to remove, and then clicks on the button for permanently removing device located at the bottom. When this is done, the system will respond with a confirmation to ensure that the administrator is certain of the decision. After the final confirmation, the system will set the device to an off state or initiate a shutdown protocol that is handled by the individual device and is beyond the scope of autoHomes offering. The system will then ping the device to see if it is still alive and once the system can confirm that the device is indeed removed, it will remove all device presences from the database.

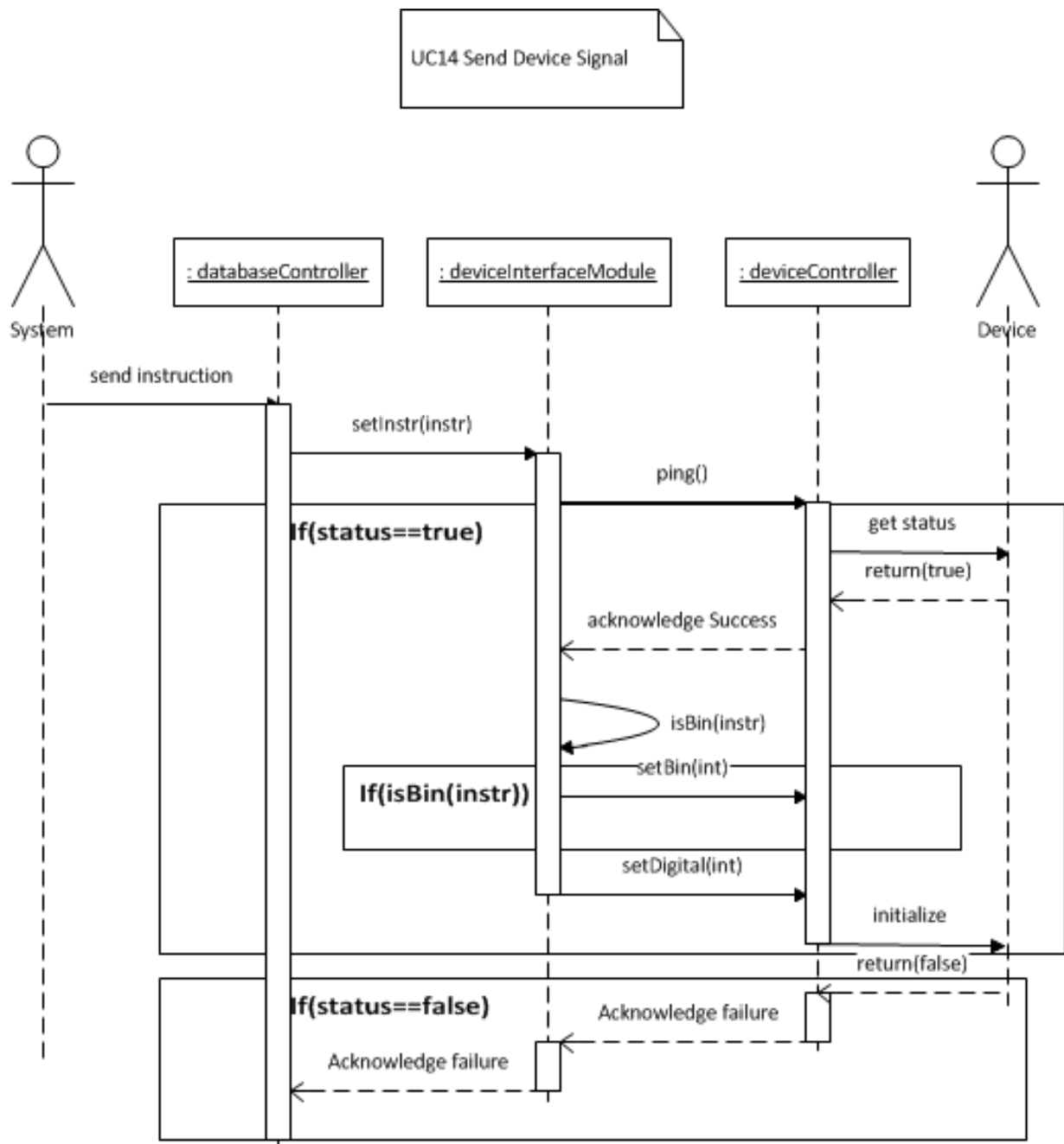


Figure 8.14 - UC-14 - Send Device Signal

Use Case 14: Send Device Signal

Use Case 14 is designed with the intentions to take the settings previously set in UC 4, 5, 12, and 13 and then implements them onto the actual physical selected device. This task requires no user interaction and is handled entirely by the system. When the system wants to send a signal to a device it first sends the instruction; once that has been handled the deviceInterfaceModule checks to see if the device is active and willing to accept a command. When ping responds by indicating that the device is on and ready, the deviceInterfaceModule checks to see what kind of data is being sent, if it is an analog value the data is encoded into a binary value which is then sent to the device. If the data is not analog the data must be digital and a Boolean value is used to indicate on or off command. As long as everything is smooth sailing the process completes. When an error occurs such as the device disappearing by being pulled out or so while the process is happening, an error is sent to the log and the process completes in failure.

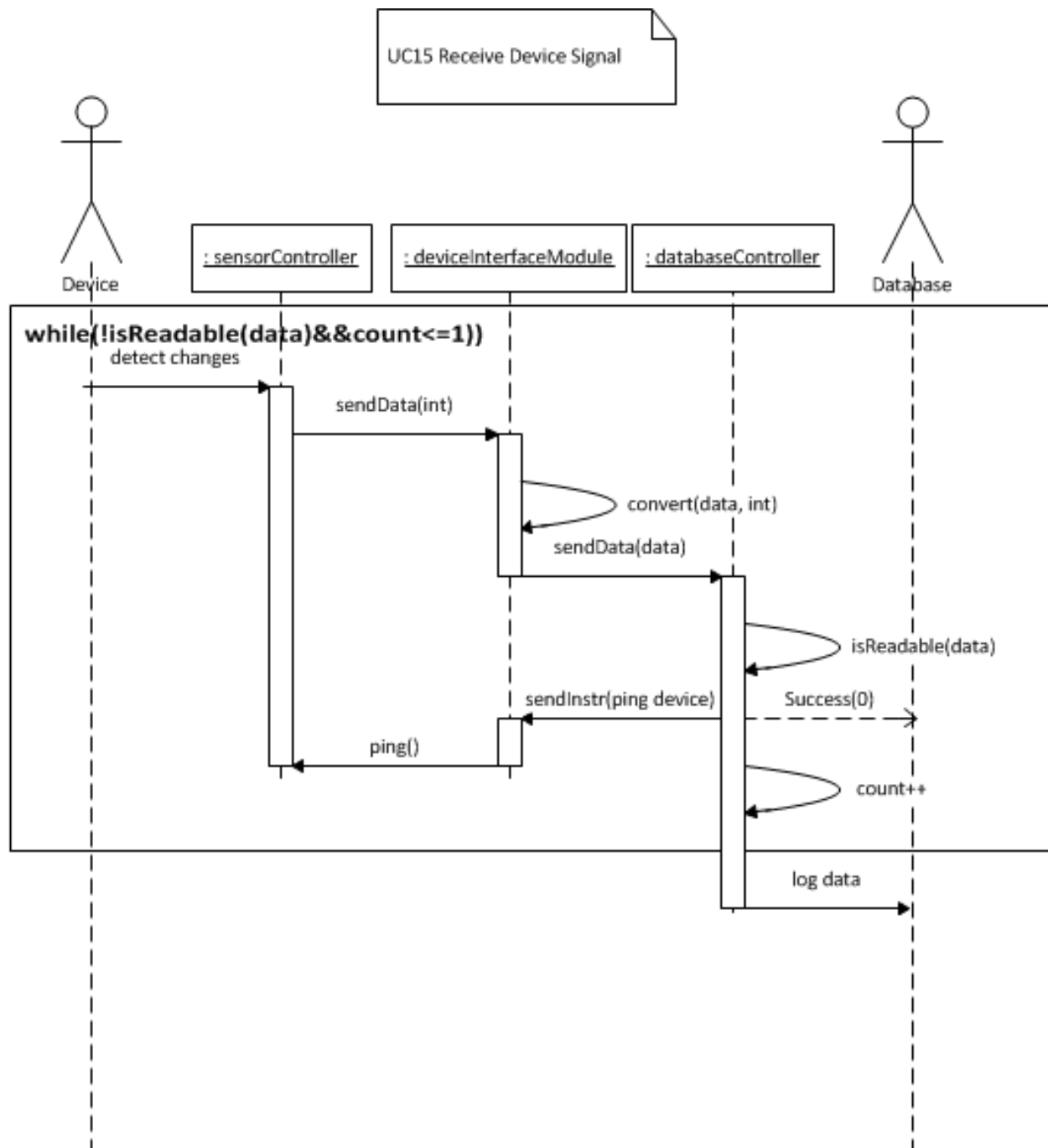


Figure 8.15 - UC-15 - Receive Device Signal

Use Case: 15 Receive Device Signal

Use Case 15 is designed with the intention of receiving incoming data and responses from individual devices. An incoming result can come from two different methods; either by a trigger that is hosted on the device, such as a manual change to a device (ie turning off a light) or by responding to a system ping query where ping returns to the system with device information such as if the device is alive and running, status of it as well. The basic principal to this use case is that 'something' in the system has changed as in a value that needs to be recorded and this use case takes that changed data, stores it in the database, and adds it to the log. This use case is performed by measuring to see if a change has been made, if this is the case, the deviceInterfaceModule converts the data acquired into formats better suited for logging such as translating an analog value to an equivalent temperature value. Once the data is converted and correct, the data is placed into the database, the log count for the device history is bumped up by one and then this allows a history of past changes to be kept. All of this happens as long as the device is connected.

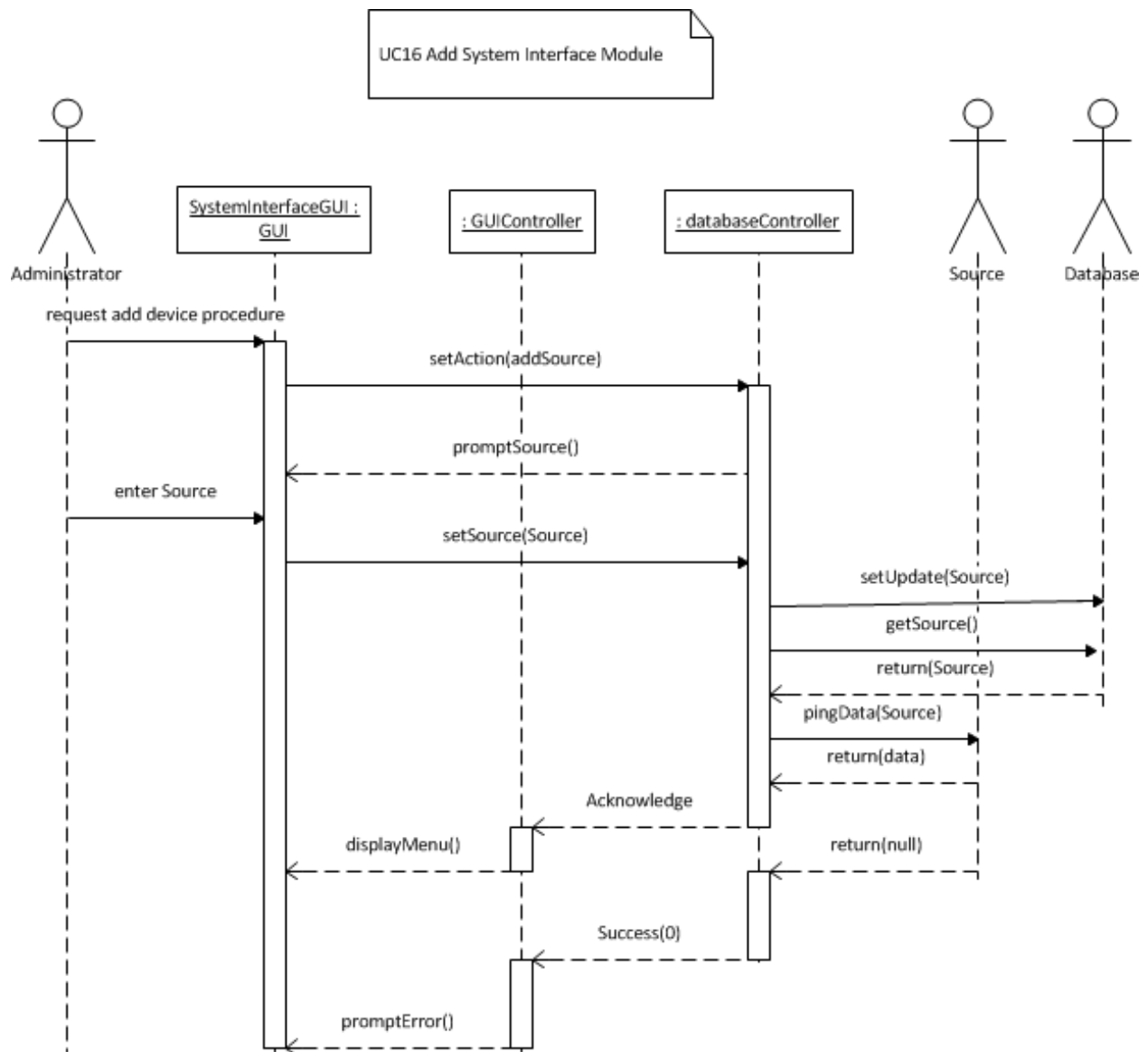


Figure 8.16 - UC-16 - Add System Interface Module

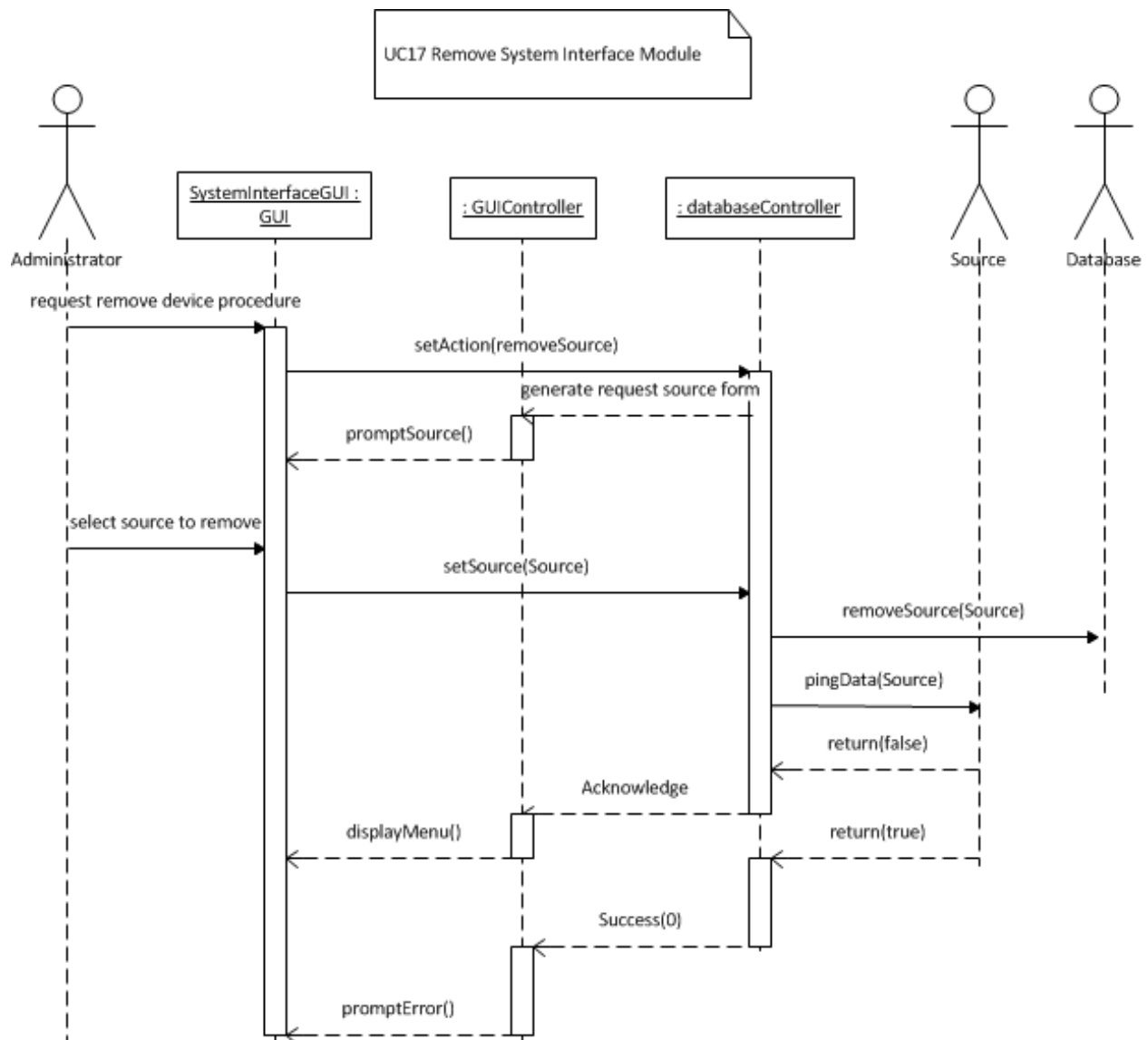


Figure 8.17 - UC-17 - Remove System Interface Module

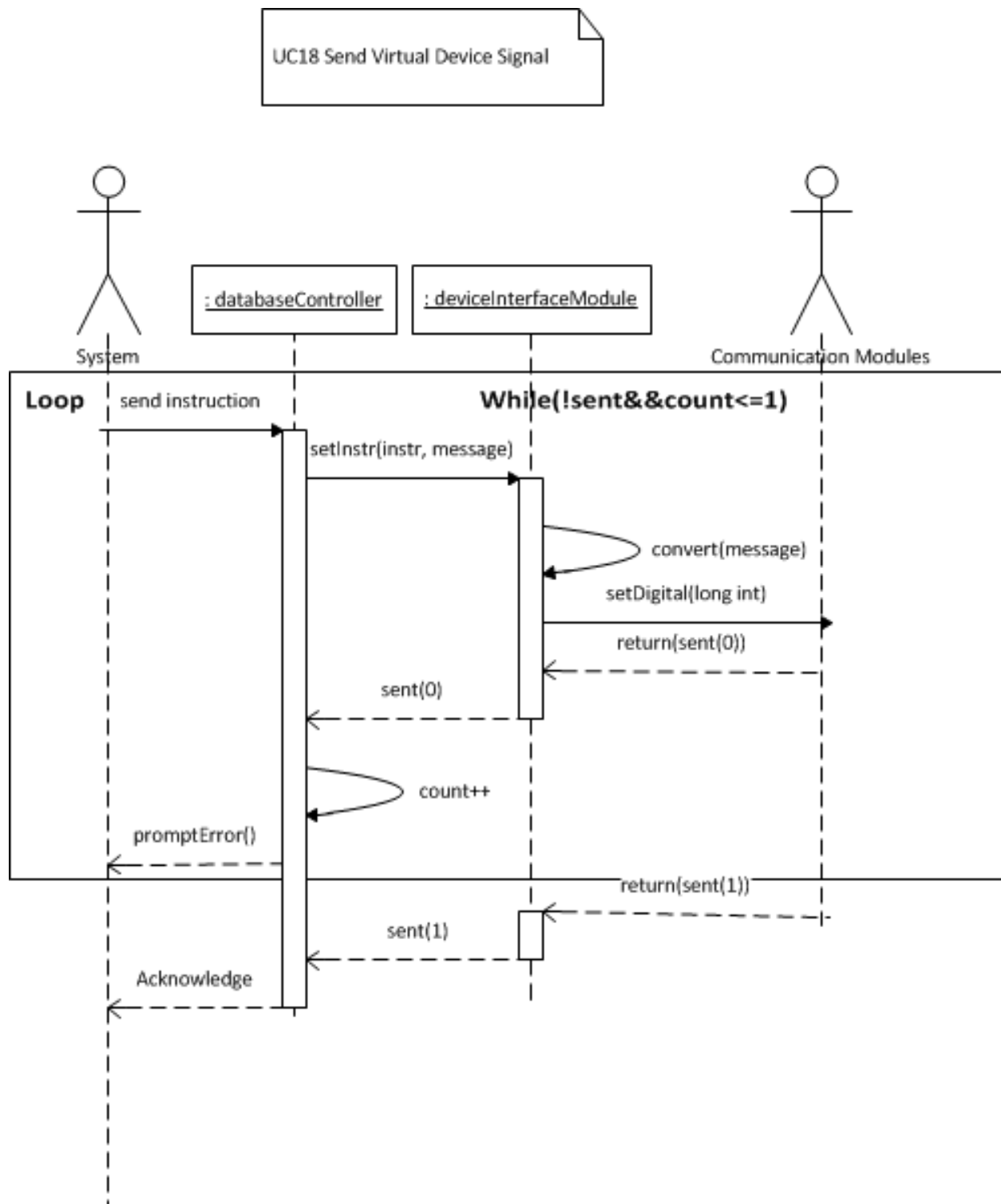


Figure 8.18 - UC-18 - Send Virtual Device Signal

Use Case 18: Send Virtual Device Signal

Use Case 18 was designed with the intention of sending a command to an external api such as the control for sending out text messages. A typical scenario involving this use case would include a fire alarm being activated and alerting the administrator through means other than the web panel such as a text messaging service. Another example may be a notification to the administrator to warn about a garage door being left open over a specified threshold where it is possible that the administrator forgot to close the door and the presence of the open door exposes a security risk to the home which should be addressed in a more active manor. When the system sends a command such as 'initiate a text message' the deviceInterfaceModule checks to see that the command is valid, if so the command is forwarded to the appropriate communication modules which include the text messaging service and email client. Once this occurs to completion, the action is logged. If an error occurs throughout the process it will be logged, and an attempt will be made to still process the request since high priority tasks such as a fire may be occurring and action will need to be prompt.

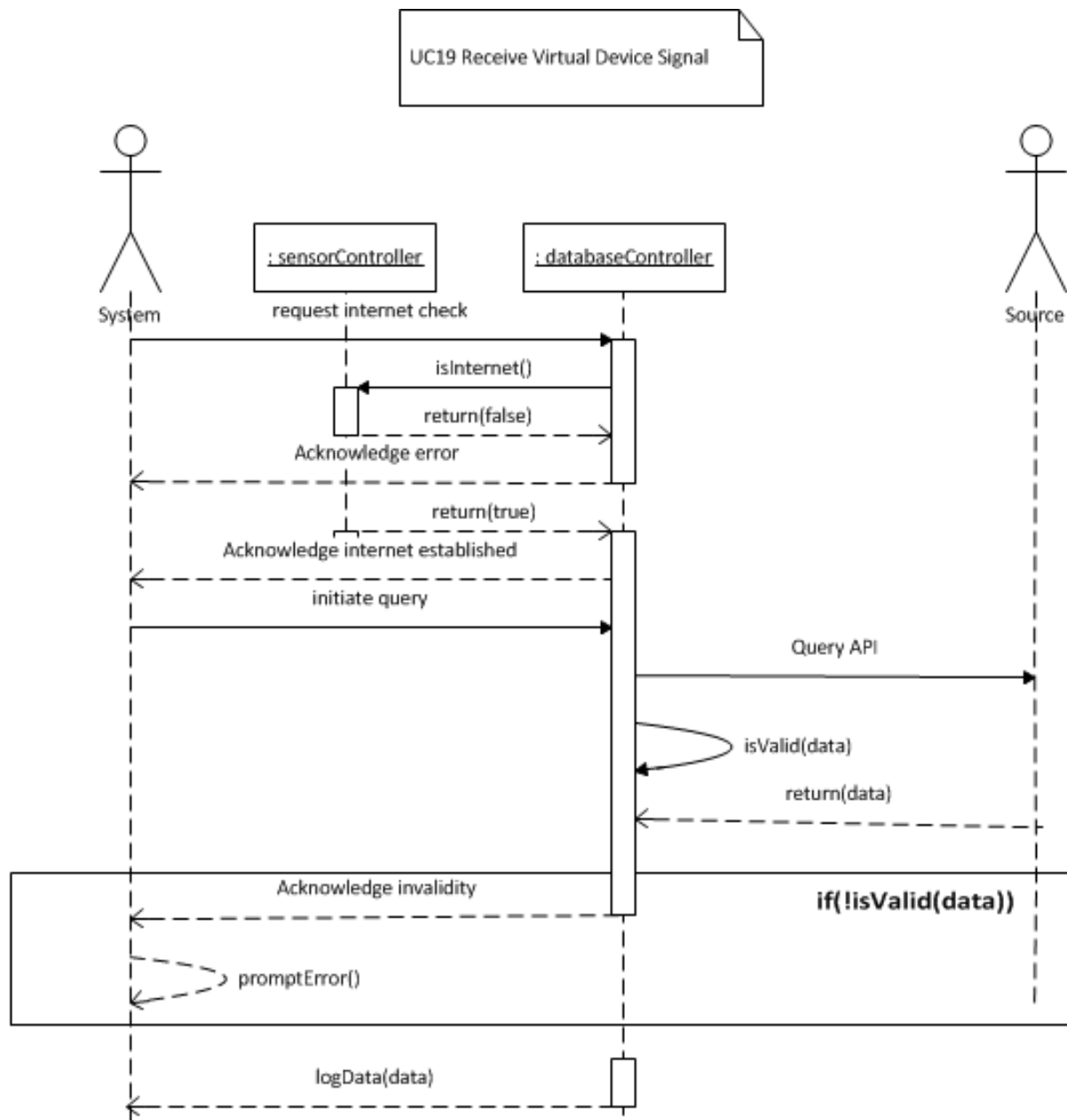


Figure 8.19 - UC-19 - Receive Virtual Device Signal

Use Case 19: Receive Virtual Device Signal

Use Case 19 was designed to perform two tasks; it will receive confirmation that UC 18 sent out for example a text message properly. But more importantly it will handle the result of querying external data sources such as local weather and almanac data, this includes sunrise/sunset times and also the clock synchronization that is set to occur about every week, this is discussed in the Network section. The general flow for this use case is that the system is already aware of UC18 being completed and as such it now will check connection is still available to these external communication API's. Then it will receive the data, verify that the data acquired is valid and then store this new data into the database. If the data is invalid the receive request will fail and an error will be logged as well as a notification will be presented to the administrator.

UC20 Evaluate Data Against Rule Sets

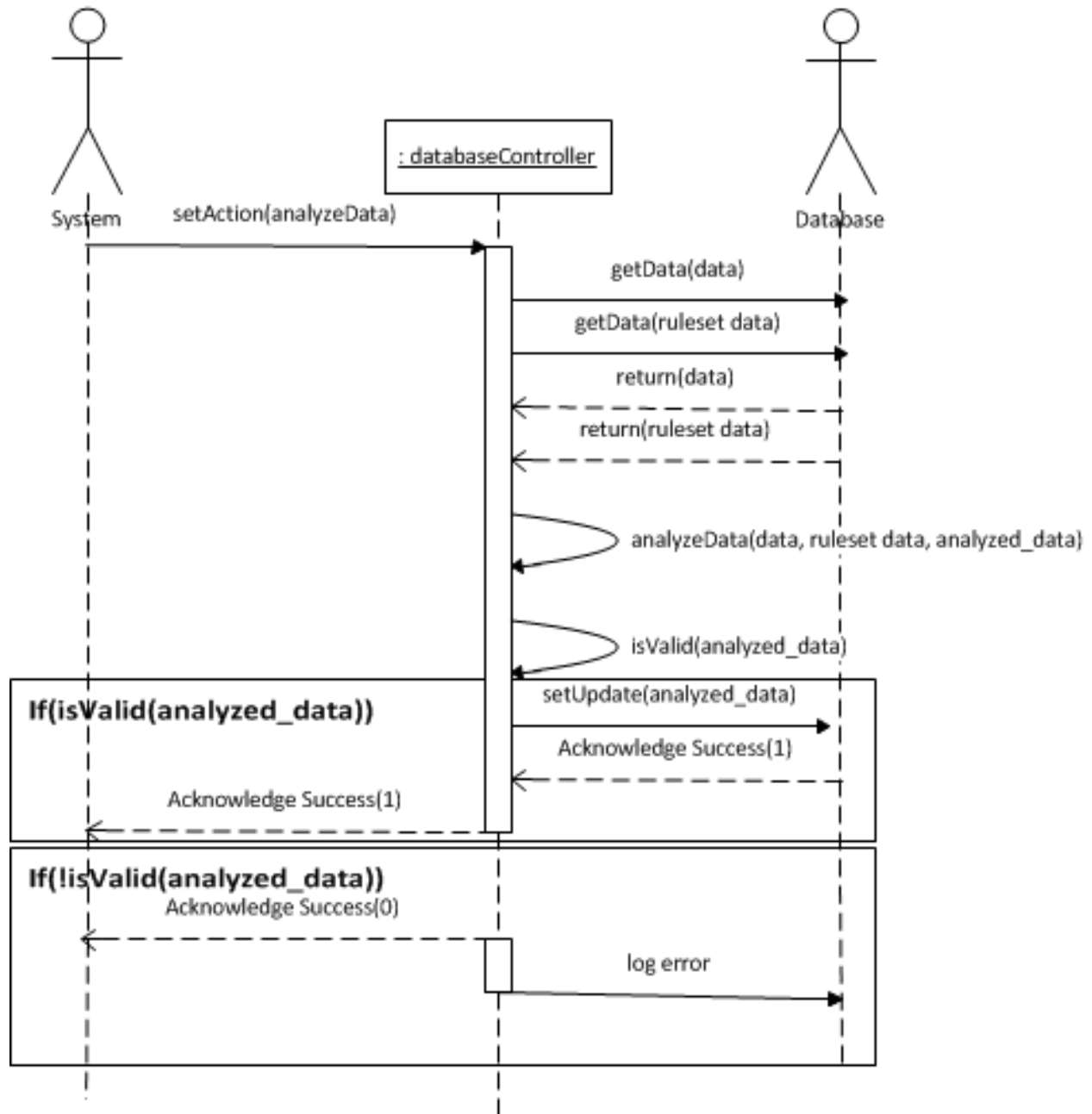


Figure 8.20 - UC-20 - Evaluate Data Against Rule Sets

Use Case 20: Evaluate Data Against Rule Sets

Use Case 20 is triggered when new data is fed into the system. The devices have pre-configured polling rates in their drivers, and when this period has elapsed, new data is read from the sensors. There is always data flowing through the reporting conductors, but data is only polled so often due to data logging and processing constraints of computers. This use-case is not directly-accessible by the user, but instead is activated as soon as a user successfully adds a device to the system. The physical voltage values from the devices are converted to real units (Fahrenheit, Percentage, True/False, Lumens) by their respective drivers. This use case takes those converted values and evaluates them against all active RuleSets (and therefore, the ConditionSets).

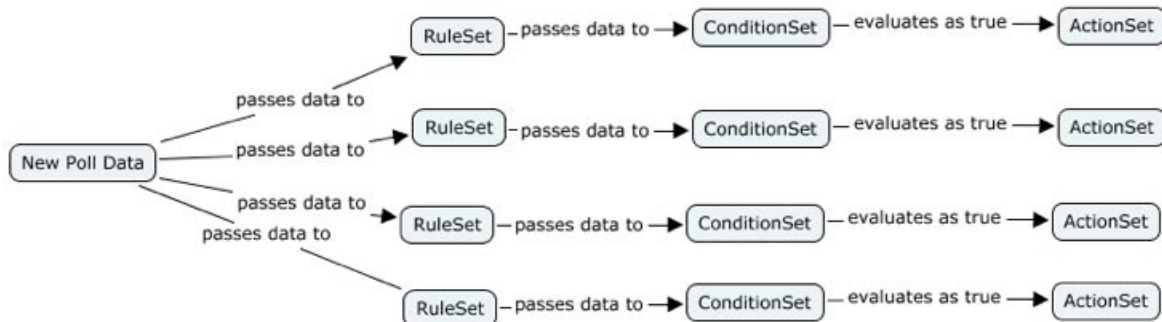


Figure 8.20.1 - UC-20 - Evaluate Data Against Rule Sets Diagram

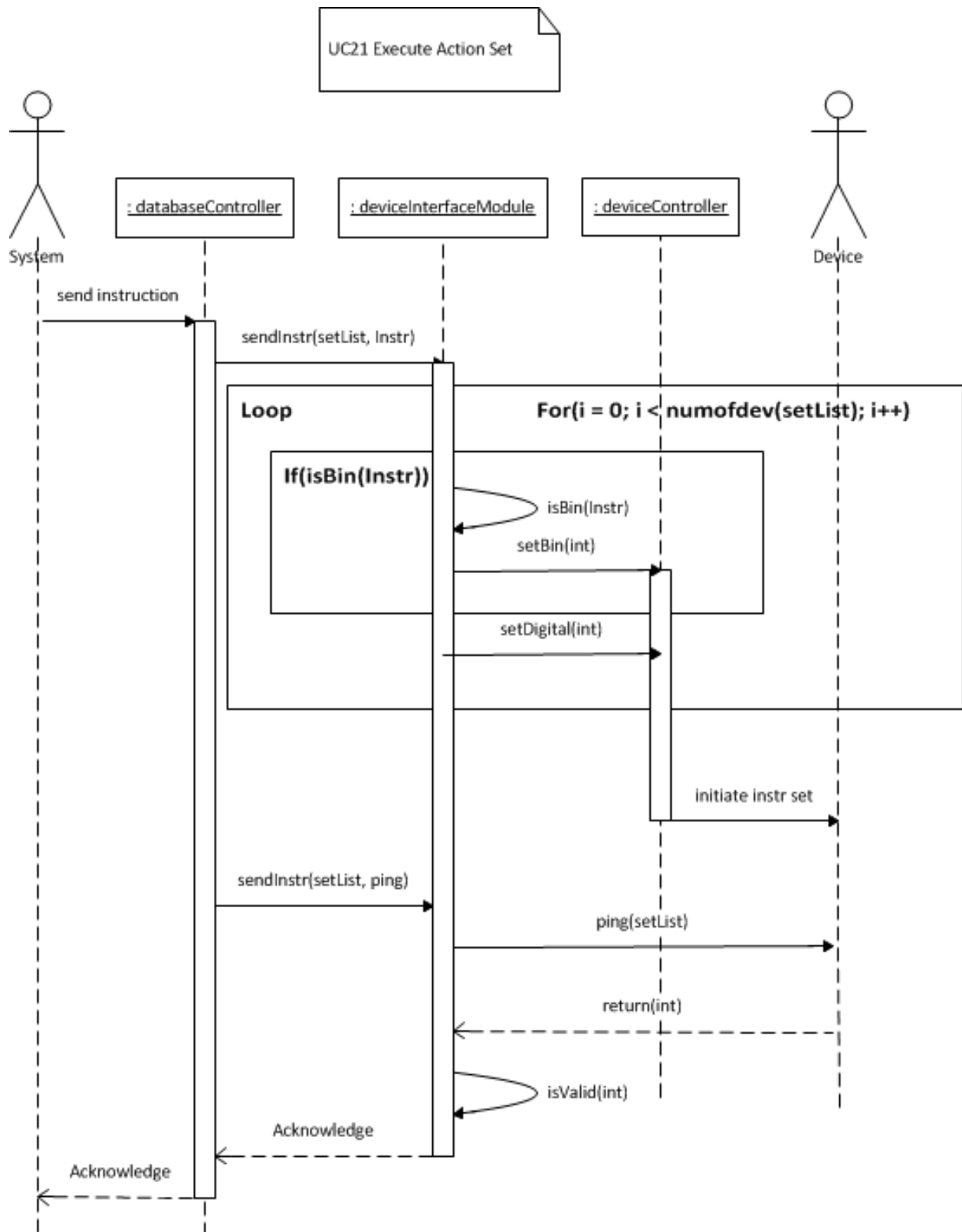


Figure 8.21 - UC-21 - Execute Action Set

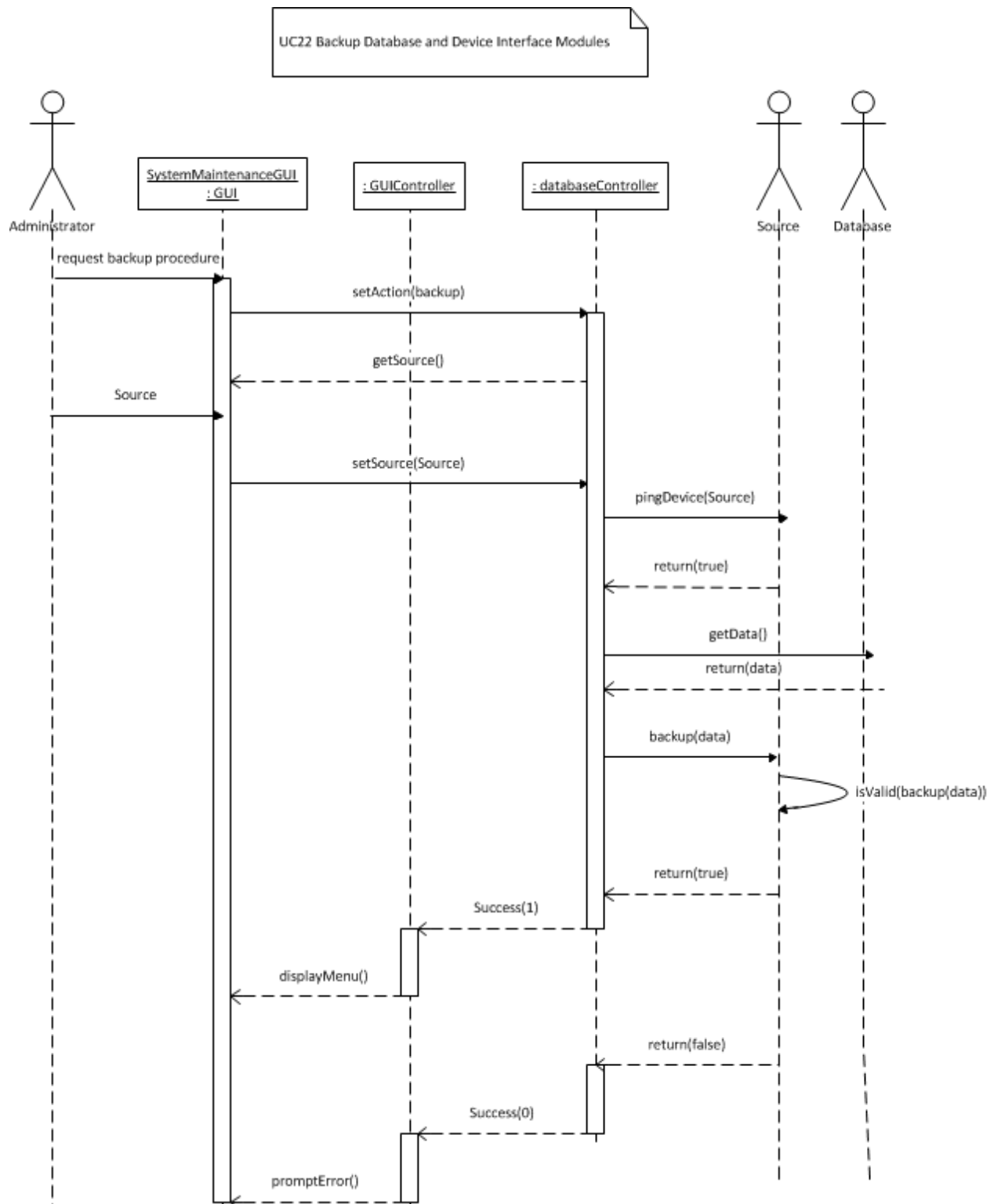


Figure 8.22 - UC-22 - Backup Database and Device Interface Modules

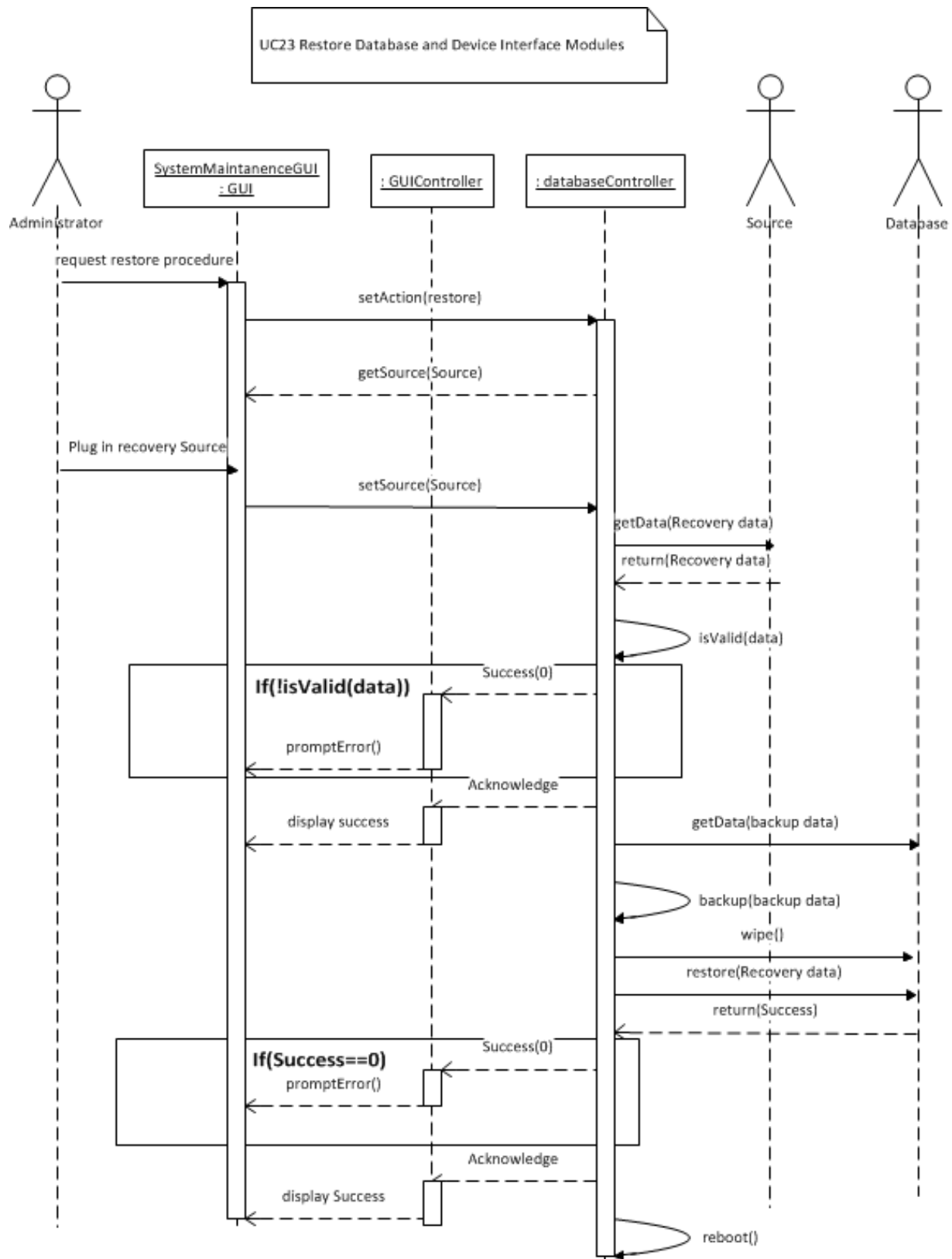


Figure 8.23 - UC-23 - Restore Database and Device Interface Modules

9. Class Diagram and Interface Specification

The following pages will show our class diagrams and traceability matrix.

a. Class Diagram

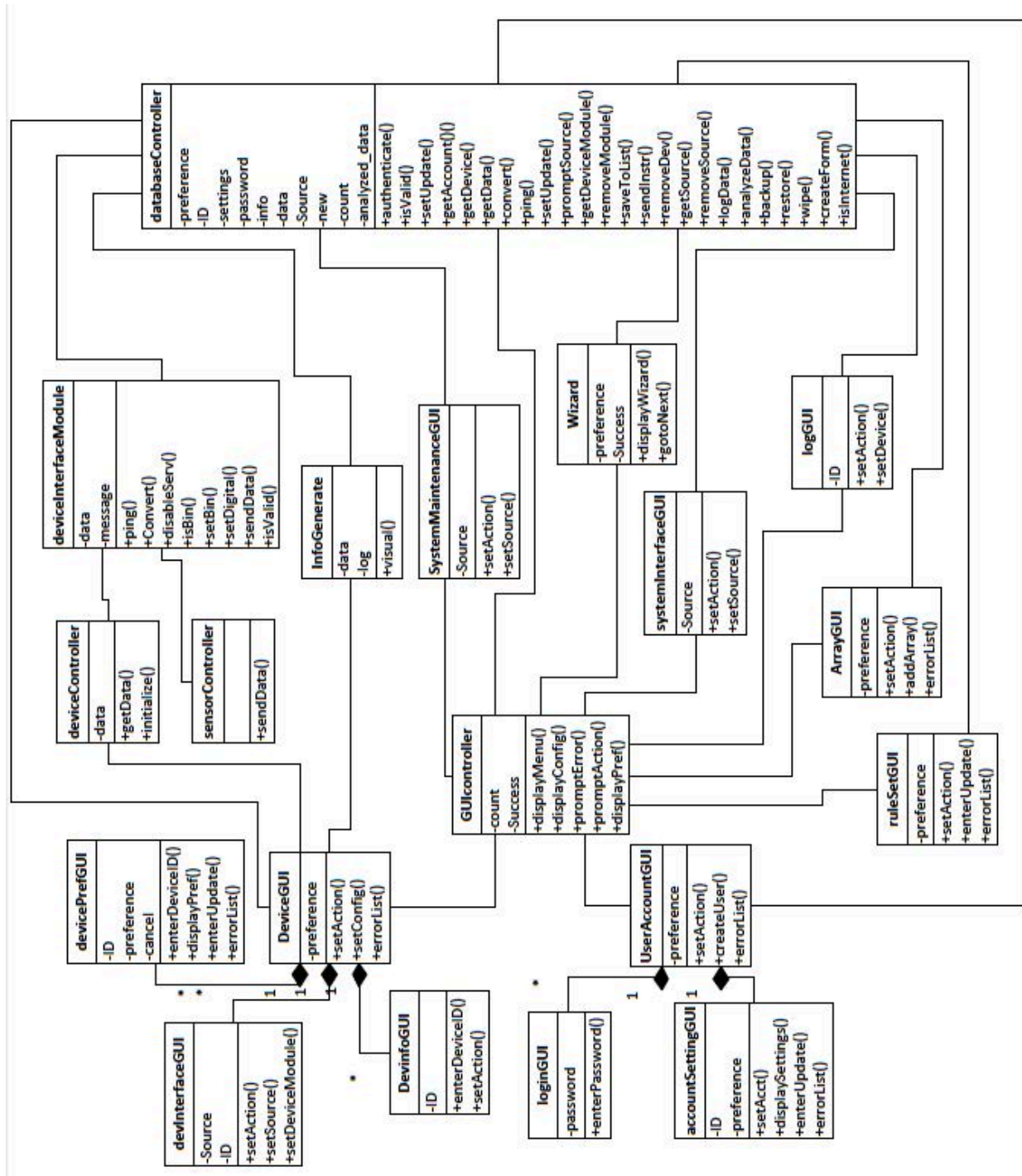


Figure 9.1 - autoHome Class Diagram

b. Data Types and Operation Signatures

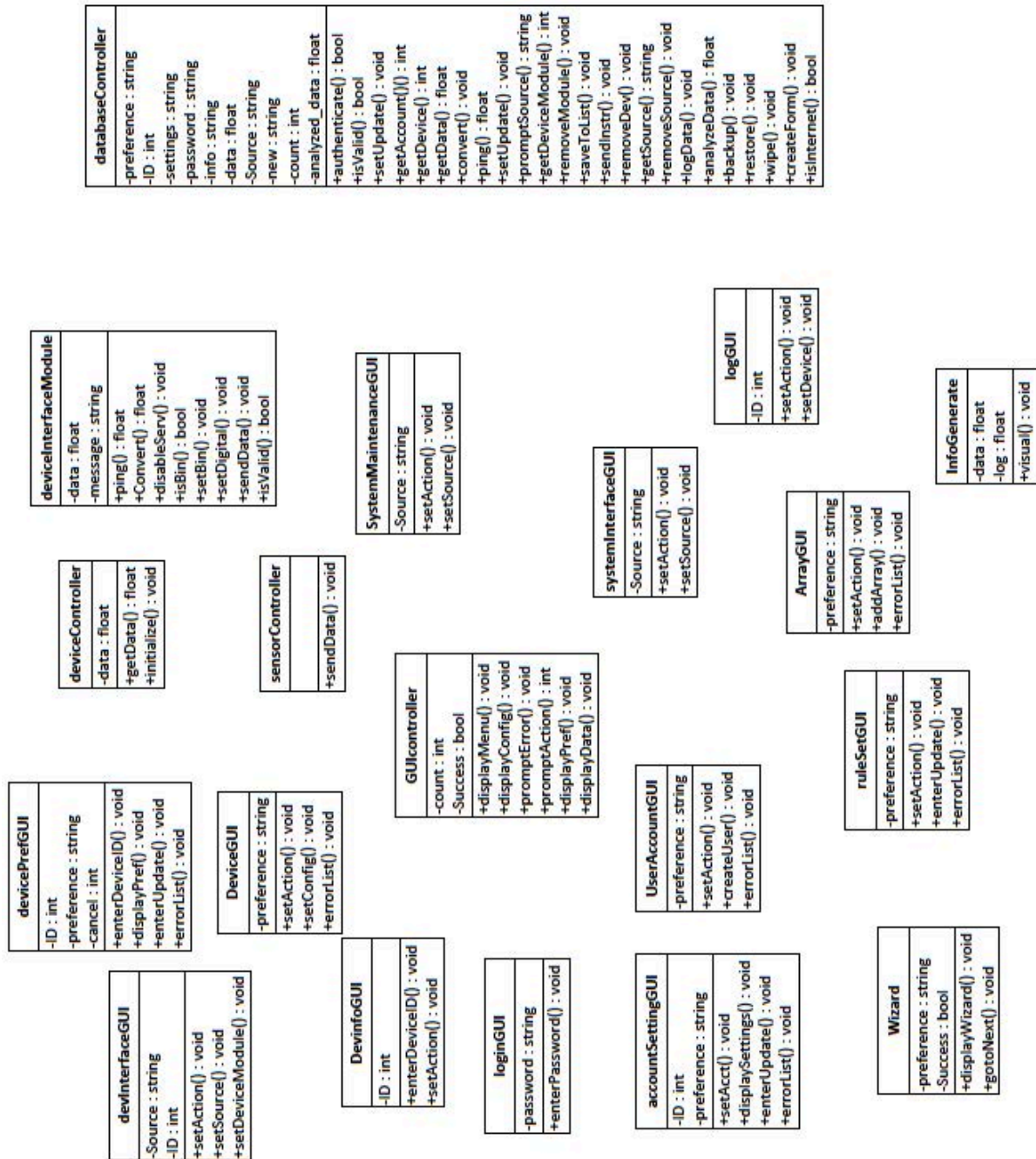


Figure 9.2 - autoHome Operation Signatures Diagram

c. Traceability Matrix

DatabaseManager

ArrayGUI is derived from *DatabaseManager* because the array of devices is shown to the user. The array of devices is logged in the Database since the Database has information on what devices are in the system.

databaseController is derived from *DatabaseManager* based on the principle that *databaseController* has over 20 different functions that the database uses. It is crucial that the functions in *databaseController* are only used by the Database and that they are not tampered with. If a user were to use these functions, they could possibly mess up the internal functioning of autoHome and brick their system.

GUIcontroller is derived from *DatabaseManager* because the Database will have to push GUI's to the user and *GUIcontroller* is used to regulate the GUI's and when they are pushed out. This is important because the system could malfunction if the database didn't have a class to regulate the GUI's actions.

InfoGenerate is derived from *DatabaseManager* based on the principle that *InfoGenerate* displays information from within the Database. It is important to have information within the Database displayed with classes that are embedded in the Database as well.

logGUI is derived from *DatabaseManager* because it will display a log in the form of a GUI to the user based on the current status of the system and it will also provide a list of events for the system in a certain period of time.

RuleSetGUI is derived from *DatabaseManager* because ruleset preferences need to be shown and the GUI needs to allow the user to change preferences as desired.

SystemInterfaceGUI is derived from *DatabaseManager* based on the fact that *DatabaseManager* has a number of different pieces of information that *SystemInterfaceGUI* needs to display. This is important because the system should print out correct and accurate information that the Database holds. If the system were to print out incorrect information, autoHome would lose credibility on all counts which would lead to negative publicity, among other things.

SystemMaintenanceGUI is derived from *DatabaseManager* based on the principle that *SystemMaintenanceGUI* requires access to parts of the Database that shouldn't be accessible to the user. Since *SystemMaintenanceGUI* checks for updates, it needs to use the "getUpdate()" function found within the Database.

Wizard is derived from *DatabaseManager* because *Wizard* provides the user with a step-by-step guide on how to use autoHome. This is beneficial because it will eliminate any possible confusion on how to use the system. This will also make autoHome more user-friendly for people that are "afraid" of learning how to use newer pieces of technology.

ControlManager

deviceController is derived from *ControlManager* due to the principle that *deviceController* receives data for the system. This is important because if our system does not take in any input, it wouldn't function as it should.

sensorController is derived from *ControlManager* due to the principle that *sensorController* sends data through the system. This is very important since our system will be taking in data, and users will expect that since they provided data, data should also be sent back to them.

DeviceManager

DeviceGUI is derived from *DeviceManager* due to the fact that a GUI needs to be displayed, which allows the user to choose what they want to see and change. This means that this GUI will display the option to pick device info, current device information, and device preferences.

devicePrefGUI is derived from *DeviceManager* due to the principle that the preferences of a device are held within *DeviceManager* and need to be available for the user to change as desired.

DevinfoGUI is derived from *DeviceManager* due to the principle that it will display information about the device which is kept in *DeviceManager*. This is important because the device's information will almost constantly be changing and *DevinfoGUI* needs the most up-to-date status reports on the device.

devInterfaceGUI is derived from *DeviceManager* due to the principle that it will give the interface of the desired device which will provide different preferences for the device that the user can change.

deviceInterfaceModule is derived from *DeviceManager* due to the principle that *deviceInterfaceModule* sends and receives data depending on what the user inputs for the preference they want to change.

GUIManager

accountSettingGUI is derived from *GUIManager* because if a user wants to change the settings on their account, such as changing their password, there needs to be a GUI that has options for the user to change. The user would need these options for a number of reasons like changing their password, a security question, or even their name on the account. There are a number of settings that should be provided for the user.

ArrayGUI is derived from *GUIManager* because the system has to produce a GUI that shows the array of devices to the user. Without this, the user would not be able to know what devices are in the system. Without knowing that at the very least, the system would crash entirely.

DeviceGUI is derived from *GUIManager* due to the fact that a GUI needs to be displayed, which allows the user to choose what they want to see and change. This means that this GUI will display the option to pick device info, current device information, and device preferences.

devicePrefGUI is derived from *GUIManager* because a GUI must be displayed that allows the user to change the preferences pertaining to the device as desired. This is beneficial because the user is allowed to have complete control over the device.

DevinfoGUI is derived from *GUIManager* due to the principle that it will display information about the specific device. This is important because the device's information will almost constantly be changing and *DevinfoGUI* needs the most up-to-date status reports on the device to provide to the user..

devInterfaceGUI is derived from *GUIManager* because a GUI needs to be shown that provides the user with preferences.

GUIcontroller is derived from *GUIManager* because there needs to be a class to regulate when a GUI is sent to the user and when it is necessary or not. This is beneficial because, without it, an unwanted GUI could pop up on the user's screen and lead to a number of confusions. For instance, a prompt could come up that wants the user's input on changing the temperature in the house. The user could put in a new number that overrides a previously inputted value, without realizing what they have done.

logGUI is derived from *GUIManager* because it will display a log in the form of a GUI to the user based on the current status of the system and it will also provide a list of events for the system in a certain period of time.

loginGUI is derived from *GUIManager* because when a user logs into autoHome, they must be prompted to enter their password. If they were to not be prompted, their account would be open for anyone to access and someone could change their settings without the user realizing it.

ruleSetGUI is derived from *GUIManager* because ruleset preferences need to be displayed for the user so that the user can choose what preferences they can change. Once again, if the GUI is not displayed, the user wouldn't be able to change them as they should.

systemInterfaceGUI is derived from *GUIManager* because a GUI needs to be displayed to show the overall system and provide options pertaining to the system such as backup, restore, wipe, check for updates, and a number of other options solely pertaining to the system as a whole.

SystemMaintenanceGUI is derived from *GUIManager* because when the system needs to check for updates, or is possibly being updated, the user needs to be prompted to update a certain part of the system, and the system needs to receive either an "update now" or "save for later" option. If the user was not prompted on when updates happen, their interface may become sluggish due to the unknown update in the background, and the system might even restart to apply the new updates to the system, without the user being aware.

UserAccountGUI is derived from *GUIManager* because when the user goes to login, upon successful login, they will be prompted with a screen showing their account in brief. They will have a brief rundown of their account and an option to change settings pertaining to their account. Without this GUI, the user would not know if they entered their password correctly, since this GUI would appear after a successful login is made, but they also would not know if their account exists since there would not be a prompt for them to see.

UserAccountManager

accountSettingGUI is derived from *UserAccountManager* due to the principle that the user should be able to change their account settings if they want, and the *UserAccountManager* should have all pertinent information stored, available for the user to change upon pulling up their "account settings."

loginGUI is derived from *UserAccountManager* due to the principle that the system should allow different users to log in to the system. *UserAccountManager* will be able to retrieve the respective account upon being authenticated.

UserAccountGUI is derived from *UserAccountManager* due to the principle that the system should be able to retrieve a certain user account and display that certain account in a GUI for the user to see. This is important because the user may want to verify any important information on their account, i.e. changing their password.

Authenticator

loginGUI is derived from *Authenticator* based on the principle that when a user enters in their user name and password, the *Authenticator* will ensure that the user name and password match. This is essential because users shouldn't have access to accounts and information that doesn't belong to them.

	Database Manger	Control Manager	Device Manager	GUI Manager	User Account Manager	Authenticator
AccountSettingsGUI				x	x	
ArrayGUI	x			x		
DatabaseController	x					
DatabaseManager	x					
DeviceController		x				
DeviceGUI			x	x		
DeviceInterfaceModule			x			
DevicePrefGUI			x	x		
DevInfoGUI			x	x		
DevInterfaceGUI				x		
GUIController	x			x		
InfoGenerate	x					
LogGUI	x			x		
LoginGUI				x	x	x
RuleSetGUI	x			x		
SensorController		x				
SystemInterfaceGUI	x			x		
SystemMaintenanceGUI	x			x		
UserAccountGUI				x	x	
Wizard	x					

Figure 9.3 - Traceability Matrix for Class Diagram

d. Design Patterns

The interaction diagrams have already been optimized to design pattern specification. Design patterns that were incorporated into our project are as follows:

Creational Patterns:

- Prototype:
 - Purpose of Pattern: The pattern is used to specify the type of objects to create using a prototypical instance. This pattern is primarily used in the instance of adding a new device or creating a new rule set in the system.
 - Use Cases utilizing pattern:
 - UC12 – Add a Device
 - In the case of adding a device which corresponds to UC12, the createForm(device preference) class located in the database controller would be the prototype from which the objects (new devices being added) would be generated.
 - UC5 – Configure Device Rule Sets
 - Similarly this type of process is followed during the creation of a new rule set or configuring of on in UC5, the prototype of the a rule would be stored in the database controller and then the wizard would access those prototypes in order to make a new rule set for a given device.

Behavioral Patterns:

- Command:
 - Purpose of Pattern: Knows the receiver of an action request and executes an action. It may also know if the action is reversible and if so then can undo the action. This pattern is mainly utilized in the removal of devices or objects from the system as the system has knowledge of the receiver of the action and executes an action based on the request. However, this pattern is not only limited to that of removal of objects, it is also utilized when the user polls for information on a device or configures a device.
 - Use Cases utilizing pattern:
 - UC3 – Change User Account Settings
 - In this case the receiver of the action request is the user themselves, and the database/ database controller executes the action (adjusting the preference of the user) based on what the user selects. In this case the database already has knowledge of the user initiating the commands (invoker). Basically the Invoker is also the receiver of the action, except when the invoker is the Administrator of the system which in that case the receivers would be the users registered into the system and the administrator themselves.
 - UC4 – Configure User Device Permission
 - Similar to UC3, except in this case the action executed determines the access permissions for the user to a set or specific device(s).
 - UC5 – Configure Device Rule Sets
 - The database controller along with the database already know the receiver of the action request, the device being configured, and executes an action which is essentially changing a specification of the device in question.

- ✧ UC6 – Display Device Information
 - In this case the invoker requests the information of a device, the ID of the device is stored in the database which the InfoGenerate has knowledge of (basically stating that the InfoGenerate knows the receiver of the action, which is the device that is located the database) . And executes the display action provided by the GUIController.
- ✧ UC13 – Remove Device
 - The device in question (one that is being removed) is known by the database and the database Controller executes the action to remove the device, this is one of the reversible objects where the user is prompted with a removal confirmation box. If the user was to cancel the removal of the device then the action is reversed/undone.
- ✧ UC17 – Remove the system interface module.
 - In this case as well the database controller has the previous knowledge of the source being removed from the SystemInterfaceGUI and the database contains the information of the source. Once the database controller gets the request to remove a source it executes the action up on the source receiving the request.
- ✧ UC20 – Evaluate Data Against Rule Sets
 - System invokes an action to the database controller to analyze data, the data is then acquired from the database and the controller performs the analysis (action) and returns the analyzed data.
- ✧ Undo/Redo:
 - Purpose of the Pattern: Allows the rollback of the user's actions in an elegant fashion. This is primarily used in the back-up and restoring of the database.
 - Use Cases utilizing pattern
 - ✧ UC23 – Restore Database and Device Interface Modules
 - This use case displays the pattern of undo/redo, since it undo the entire database from a backup. As in by restoring the database the system is essentially undoing the current changes and reverting to an older iteration.
- ✧ Publish/Subscribe:
 - Purpose of the Pattern: Defines a dependency between objects where the publisher knows the event source and the interested object (subscriber) and registers/unregisters the subscribers and notifies the subscribers of events. The subscriber knows event types of interest and knows the publisher; is responsible for registering/unregistering with the publishers and process any event notifications received. This pattern is essentially what the entire autoHome system is based on. Almost every use case utilizes the publisher/subscriber pattern. Shown below are the few that are the more obvious use cases, where in most if not all the use cases the publisher's role is played by the database and the database controller.
 - Use Cases utilizing pattern
 - ✧ UC1 – Authenticate User
 - In this case the publisher's role is played by the database and the database controller and the subscriber is the user themselves. The user authenticates themselves with the database by logging in.
 - ✧ UC2 – Create User Account
 - Again the database and the database controller play the role of the publisher, however in this situation it's easy to notice the process of the publisher's job, as the database controller registers a new user. Similarly the user is also registering with the publisher as is required by them, in order to gain access to the database.
 - ✧ UC4 – Configure User Device Permission

- Undo/Redo:
 - Purpose of the Pattern: Allows the rollback of the user's actions in an elegant fashion. This is primarily used in the back-up and restoring of the database.
 - Use Cases utilizing pattern
 - UC23 – Restore Database and Device Interface Modules
 - This use case displays the pattern of undo/redo, since it undo the entire database from a backup. As in by restoring the database the system is essentially undoing the current changes and reverting to an older iteration.

- Publish/Subscribe:
 - Purpose of the Pattern: Defines a dependency between objects where the publisher knows the event source and the interested object (subscriber) and registers/unregisters the subscribers and notifies the subscribers of events. The subscriber knows event types of interest and knows the publisher; is responsible for registering/unregistering with the publishers and process any event notifications received. This pattern is essentially what the entire autoHome system is based on. Almost every use case utilizes the publisher/subscriber pattern. Shown below are the few that are the more obvious use cases, where in most if not all the use cases the publisher's role is played by the database and the database controller.
 - Use Cases utilizing pattern
 - UC1 – Authenticate User
 - In this case the publisher's role is played by the database and the database controller and the subscriber is the user themselves. The user authenticates themselves with the database by logging in.
 - UC2 – Create User Account
 - Again the database and the database controller play the role of the publisher, however in this situation it's easy to notice the process of the publisher's job, as the database controller registers a new user. Similarly the user is also registering with the publisher as is required by them, in order to gain access to the database.
 - UC4 – Configure User Device Permission

- The database “registers” the user under the device which the user is being given access to. Then the database notifies the user of this event.
- ⤴ UC5 – Create Rule Sets
 - The database performs it's duty by sending the user a notification of the event (the event being the user making a rule for a device) and as would the user checks the notification and takes action depending on the type of notification.
- ⤴ UC22/23 – Backup/Restore Database and Device Interface Modules
 - These two use cases the database controller notifies the user after a backup or restore has been completed. This notification could be due to a failure in the backup/restore process or a successful process.
- ⤴ State
 - Purpose of the pattern: The pattern is used when an object's behavior depends on it's state in a complex way. The pattern is not as common in the autoHome system however it is utilized in situations such as polling data and executing an action based on the validity of the data received.
 - Use Cases utilizing pattern:
 - ⤴ UC19 – Receive Virtual Device Signal:
 - Here the system and source display how the state pattern is utilized, the system polls for data from the source (which is a device/sensor on a device). Once the system retrieves the data it checks for the validity and consistency of the data, after doing so it either logs the data (if the data is valid) or it displays an error (if invalid).

e. Object Constraint Language (OCL)

Contract CO1:	Configure user device permission
Operation:	setUpdate(deviceID; string, Permission: bool)
Cross Reference:	UC4 Configure user device permission
Precondition:	The device is connected to the database and is active The device is already configured
Postcondition:	The administrator updates the permissions for users to configure the selected device

Figure 9.4 - Contract CO1: Configure User Device Permission

Contract CO2:	Create Rule Sets
Operation:	setUpdate(deviceID; string, RulePreference: string)
Cross Reference:	UC5 Create Rule Sets
Precondition:	The device has to be registered in the database and active. The user must have access to the device's permissions. A wizard is provided to set up the rule sets.
Postcondition:	The device is updated with the new permissions and rules. A notification of the event is sent to the user. Exception: The device permission is not configured if the device is not active or in the database. An error message is sent to the user and is also displayed on the screen.

Figure 9.5 - Contract CO1: Create Rule Sets

Contract CO3:	Install Device interface modules
Operation:	setUpdate(new_dev:string)
Cross Reference:	UC8 Install Device interface modules
Precondition:	<p>The device being added must be connected to the module.</p> <p>The device is on.</p> <p>The database connection is online.</p> <p>The device drivers are available to install.</p>
Postcondition:	<p>The device interface modules are installed from the drivers to the database.</p> <p>The database has successfully installed and assimilated the device into its ecosystem.</p> <p>The device is refreshed and displayed to the user.</p> <p>Exception: The drivers are not properly installed.</p> <p>An error is reported to the administrator.</p>

Figure 9.6 - Contract CO3: Install Device Interface Modules

Contract CO4:	Add interface array
Operation:	setUpdate(deviceType:string)
Cross Reference:	UC10 Add Interface Array
Precondition:	The device is connected to a module.
Postcondition:	The system successfully detected and an additional interface array is created.

Figure 9.7 - Contract CO4: Add Interface Array

Contract CO5:	Poll Sensor Data
Operation:	sendData(DataValue:float, DeviceID:string)
Cross Reference:	UC15 Receive Device Signal
Precondition:	The sensor is connected to the database system and is active.
Postcondition:	<p>The data is collected and converted to more user-friendly terminology.</p> <p>Exception: The sensor does not respond to the ping by the device interface module.</p>

Figure 9.8 - Contract CO5: Poll Sensor Data

Contract CO6:	Execute Action Set
Operation:	sendInstr(setList:array, ping:integer)
Cross Reference:	UC21 Execute Action Set
Precondition:	Device is configured and is pinging with the system.
Postcondition:	<p>The action set is executed successfully and multiple devices are activated.</p> <p>Exception: Some of the related devices are not activated.</p> <p>System retries to get connection with devices and sends notifications to the administrator regarding this error.</p>

Figure 9.9 - Contract CO6: Execute Action Set

10. System Architecture and System Design

a. Architectural Styles

Software architecture refers directly to the composition of our system. The architecture can be shown through a conceptual model, or models, that show the system's behavior and other views of our system. Our system uses a number of different software architectures, which are defined as follows:

- The client-server model allows for a client to use their computer to access a server through the Internet. [15]
- Plug-in computing is the idea that smaller software can be used and added to a system to provide additional functionality to a larger piece of software. [15]
- An event-driven architecture is where the system focuses primarily on events. An event is defined as the action that changes the state of something in the system. [15]

It may seem peculiar that our system uses a few different software architectures, but it also makes a lot of sense. Since autoHome is considered to be a complete system, there are a lot of different parts that need to work together in order to accomplish their assigned tasks through software.

autoHome works by allowing a consumer to access a web-based interface, allowing them to access their entire home and the home's respective utilities/components. At this level, it sounds like the software architecture would fall under the client-server model, since the consumer can access the system from any computer with an Internet connection. However, on the interface, there will also be a list of the current states of all devices connected to autoHome. These states can be changed at any given time by the consumer in just a few clicks. This would now make our system seem to fall under an event-driven software architecture. Despite our system being very large, it will have number of different functionalities when it is put in consumers' hands. Nonetheless, we plan to have additional modules that will provide added functionality to the physical system as well as the software interface. These additional modules will continue to be developed for autoHome for added functionality. This makes our architecture now seem to fall under a plug-in architecture.

As you can see, once you understand how our system works and how the interface reacts to the consumer, we have to consider it under three different software architectures.

b. Identifying Subsystems

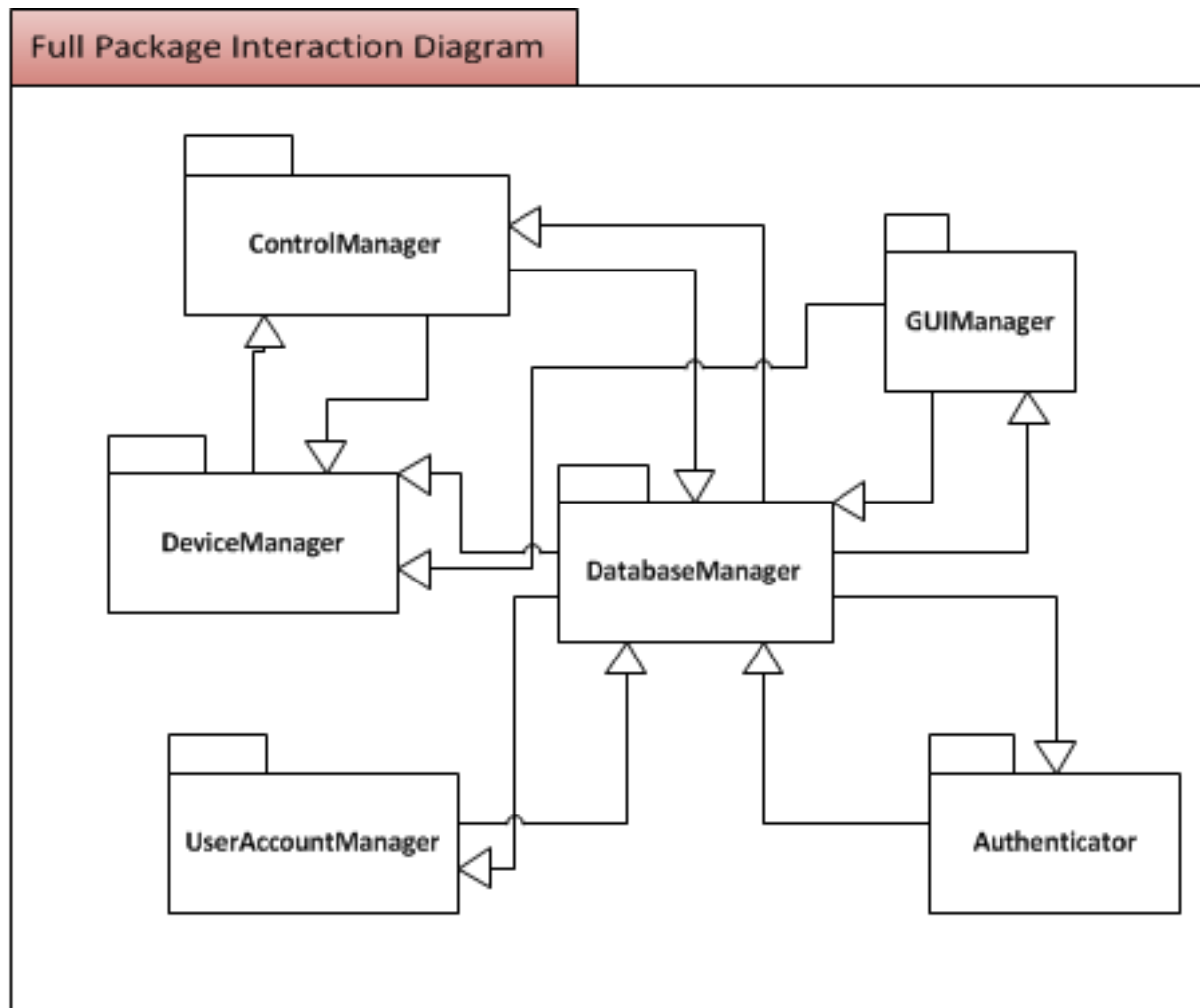


Figure 10.1 - Packaging Diagram

From the traceability matrix above, it can be seen that 6 distinct packages exist in the autoHome system. These subsystems are as follows; DatabaseManager, DeviceManager, GUIManager, UserAccountManager, ControlManager, and Authenticator. As a result of these subsystems and contained classes, a relationship can be seen between the subsystems which match the classes.

c. Mapping Subsystems to Hardware

autoHome exists on a physical in home server as well as embedded hardware, tailored to work with devices in the home.

The server in the users home will be used to connect all of the devices and record system status and properties such as whether or not a light is on or if an instruction is sent to close the garage door. The server will store all of this data into a database as well as a web server to display the content stored in the database. The server will also run a daemon to query all of the devices plugged in.

Embedded systems will be used to run hardware based code, the embedded devices will transmit data between the devices and the system which in a sense give the devices the ability to talk.

d. Persistent Data Storage

Below are select table properties along with completed database schematics.

Device
integer id integer location_id integer driver_id integer device_type_id boolean alive boolean active string name string description
integer getDeviceID() integer getDeviceLocationID() setDeviceLocationID(integer) boolean isAlive() setAlive(boolean) boolean isActive() setActive(boolean) string getName() setName(string) string getDescription() setDescription()
extends ActiveRecord

Figure 10.2 - Device Database Properties

RuleSet System (using ActiveRecord)

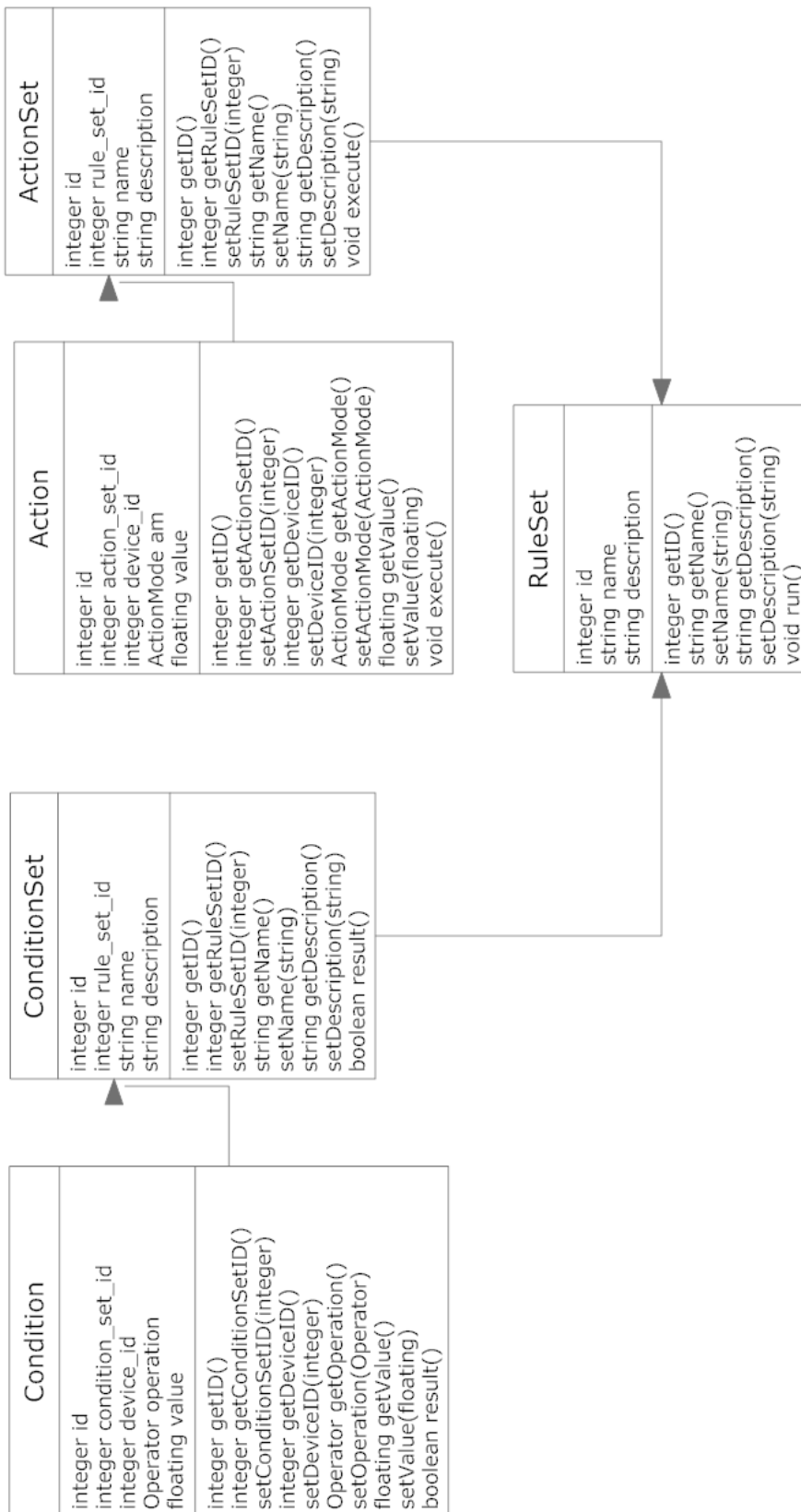


Figure 10.3 - Rule Set System Diagram

Account Management System (using ActiveRecord)

User	UserPermission
integer id string user_name string hashed_password boolean administrator string full_name date creation_date date last_login_date	integer id integer user_id integer device_id integer creator_id date creation_date
static string hashPassword(string) integer getID() string getUserName() setUserName(string) boolean isAdministrator() setAdministrator() string getHashedPassword() setHashedPassword(string) string getFullName() setFullName(string) date getCreationDate() date getLastLoginDate()	integer getID() integer getUserID() integer getDeviceID() integer getCreatorID() date getCreationDate()
extends ActiveRecord	extends ActiveRecord

Figure 10.4 - User and User Permissions Database Properties

actions

Field	Type	Null	Default
id	int(11)	No	
date_created	datetime	Yes	NULL
device_id	int(11)	Yes	NULL
value	decimal(5,3)	Yes	NULL
duration	int(11)	Yes	NULL
user_id	int(11)	Yes	NULL

Figure 10.5 - “actions” Database Table

action_relation

Field	Type	Null	Default
id	int(11)	No	
action_id	int(11)	Yes	NULL
action_set_id	int(11)	Yes	NULL

Figure 10.6 - “action_relation” Database Table

action_sets

Field	Type	Null	Default
id	int(11)	No	
date_created	datetime	Yes	<i>NULL</i>
description	varchar(1024)	Yes	<i>NULL</i>
user_id	int(11)	Yes	<i>NULL</i>

Figure 10.7 - “action_sets” Database Table

conditions

Field	Type	Null	Default
id	int(11)	No	
date_created	datetime	Yes	<i>NULL</i>
device_id	int(11)	Yes	<i>NULL</i>
value	decimal(5,3)	Yes	<i>NULL</i>
comparison	int(11)	Yes	<i>NULL</i>
user_id	int(11)	Yes	<i>NULL</i>

Figure 10.8 - “conditions” Database Table

condition_relation

Field	Type	Null	Default
id	int(11)	No	
condition_id	int(11)	Yes	<i>NULL</i>
condition_set_id	int(11)	Yes	<i>NULL</i>

Figure 10.9 - “condition_relation” Database Table

condition_sets

Field	Type	Null	Default
id	int(11)	No	
date_created	datetime	Yes	<i>NULL</i>
description	varchar(1024)	Yes	<i>NULL</i>
user_id	int(11)	Yes	<i>NULL</i>

Figure 10.10 - “condition_sets” Database Table

data_log

Field	Type	Null	Default
id	int(11)	No	
date	datetime	Yes	<i>NULL</i>
device_id	int(11)	Yes	<i>NULL</i>
value	decimal(5,3)	Yes	<i>NULL</i>

Figure 10.11 - “data_log” Database Table

data_log_rules

Field	Type	Null	Default
id	int(11)	No	
device_id	int(11)	Yes	<i>NULL</i>
delta	decimal(5,3)	Yes	<i>NULL</i>
period	int(11)	Yes	<i>NULL</i>

Figure 10.12 - “data_log_rules” Database Table

devices

Field	Type	Null	Default
id	int(11)	No	
name	varchar(32)	Yes	<i>NULL</i>
device_type_id	int(11)	Yes	<i>NULL</i>
location_id	int(11)	Yes	<i>NULL</i>
active	tinyint(1)	Yes	<i>NULL</i>
default_value	decimal(5,3)	Yes	<i>NULL</i>

Figure 10.13 - “devices” Database Table

device_permission

Field	Type	Null	Default
id	int(11)	No	
user_id	int(11)	Yes	<i>NULL</i>
device_id	int(11)	Yes	<i>NULL</i>

Figure 10.14 - “device_permission” Database Table

device_types

Field	Type	Null	Default
id	int(11)	No	
name	varchar(32)	Yes	<i>NULL</i>
module_name	varchar(32)	Yes	<i>NULL</i>
data_type	varchar(32)	Yes	<i>NULL</i>

Figure 10.15 - “device_types” Database Table

locations

Field	Type	Null	Default
id	int(11)	No	
name	varchar(32)	Yes	<i>NULL</i>
room_id	int(11)	Yes	<i>NULL</i>
description	varchar(1024)	Yes	<i>NULL</i>

Figure 10.16 - “locations” Database Table

rooms

Field	Type	Null	Default
id	int(11)	No	
name	varchar(32)	Yes	<i>NULL</i>
floor	int(11)	Yes	<i>NULL</i>
description	varchar(1024)	Yes	<i>NULL</i>

Figure 10.17 - “rooms” Database Table

rule_sets

Field	Type	Null	Default
id	int(11)	No	
action_set_id	int(11)	Yes	<i>NULL</i>
conditon_set_id	int(11)	Yes	<i>NULL</i>
description	varchar(1024)	Yes	<i>NULL</i>
date_created	datetime	Yes	<i>NULL</i>
active	tinyint(1)	Yes	<i>NULL</i>

Figure 10.18 - “rule_sets” Database Table

users

Field	Type	Null	Default
id	int(11)	No	
name	varchar(256)	Yes	NULL
date_created	datetime	Yes	NULL
last_login	datetime	Yes	NULL
administrator	tinyint(1)	Yes	NULL
password_hash	varchar(256)	Yes	NULL
last_ip	varchar(64)	Yes	NULL
phone_number	varchar(32)	Yes	NULL

Figure 10.19 - “users” Database Table

e. Network Protocol

autoHome uses serial communication to connect the embedded hardware to the server. Serial or RS-232 communication is an older platform and as such a FTDI chip [18] will be used to translate the serial communication to Universal Serial Bus or USB. USB will serve two purposes for autoHome; It will power low level devices in the range of 0 - 5v as supplied from the USB standard, while the up and down streams will be used to transmit messages that can be read in to the embedded hardware. Messages can easily be read on the embedded devices through built in serial libraries, while the server can read incoming messages with some simple C code.

f. Global Control Flow

The autoHome system is an event driven system where actions are based from cause and effect type actions such as “turn on a light in this room” then the light in the room turns on. Also an event doesn't just have to come as a result of a user based action, an event can come from a timer such as an alarm to wake up or a sensor being triggered such as a fire alarm.

System time will be maintained by the computer clock which is also synchronized with the national atomic government time [16] once a week. Accuracy of time is not crucial because the timing of events that rely on time such as turning on a sprinkler really do not need to be more accurate than to the nearest minute. Time is also backed up through use of a simple clock battery that can last around fifteen years.

autoHome is a real-time system however multiple actions can occur at any given time and the system will be able to prioritize by importance such as a safety control like a fire alarm over tuning on a light but it can also process tasks in a queue like system.

g. Hardware Requirements

Our server hardware does not need to be too sophisticated given that it will only be processing PHP, MySQL, AJAX, jQuery, and C. To enhance system performance, the following specification is recommended:

Processor: Intel Pentium 4 2.0 GHz or better
Memory: 512MB or more
Storage: 50GB or larger
Broadband Internet Connection \geq DSL 1.2 MBits/s or equivalent

Client (Administrator) platform requires the following hardware specification to run our software:

Processor: Intel Pentium III 2.0GHz or better
Memory: 256MB or more
Storage: 2GB or more
Display: 1024 x 768 16-bit color or better

Client can also access the system interface via Android devices [17]. Our Android software requires the following hardware specification:

OS: Google Android 2.1 or later
Processor: Qualcomm MSM8250 768MHz or better
Memory: 512MB or more
Storage 1GB or more
Display: HVGA 480 X 320 16-bit color or better

11. Algorithms and Data Structures

The autoHome system has been designed in such a way that it requires no algorithms to be implemented in order for decisions to occur from within the system. This is due to the fact that many of the processes occurring in the system occur on a schedule such as “query the weather, every hour” this requires no algorithm to make this happen.

Because of the structure of the database, all information is stored within the database tables. As a result when the user wants to for example see a list of all devices on the system, all the system has to do is spit out the table that contains all of the devices. It is with this simplicity and convenience in using MySQL with PHP that this can be done all without implementing any data structures.

12. User Interface Design and Implementation

User Interface Design

Preliminary Design

For the design of the web interface it was essential that the user use the least amount of effort to navigate the interface. To make things very clean and simple to work with a minimalist approach was taken to the design of the interface.

a. Preliminary Design and Implementations

The following pages display our interface design with brief explanations.



Figure 12.1 - Preliminary Login Screen Design

The user is initially greeted with the login page for the web interface of autoHome. Here, they will login with their given ID and PIN to access the interactive settings section of the web interface.



Figure 12.2 - Preliminary User Settings Screen Design

After logging in successfully, the user is presented with a very simple selection of options. This is an effort to make the system as user friendly as possible. The selections range from Status, Settings, Alerts, and Exit. Status button redirects the user to the status page, which displays the status of the devices connected to the system. Settings redirects the user to the settings page where the user can configure the devices. Alerts will redirect to the alerts page where all of the alerts are displayed in a simple format for the user to check. The Exit button simply exits from the web interface. This screen is made larger to augment with the entire minimalist and user friendly theme.

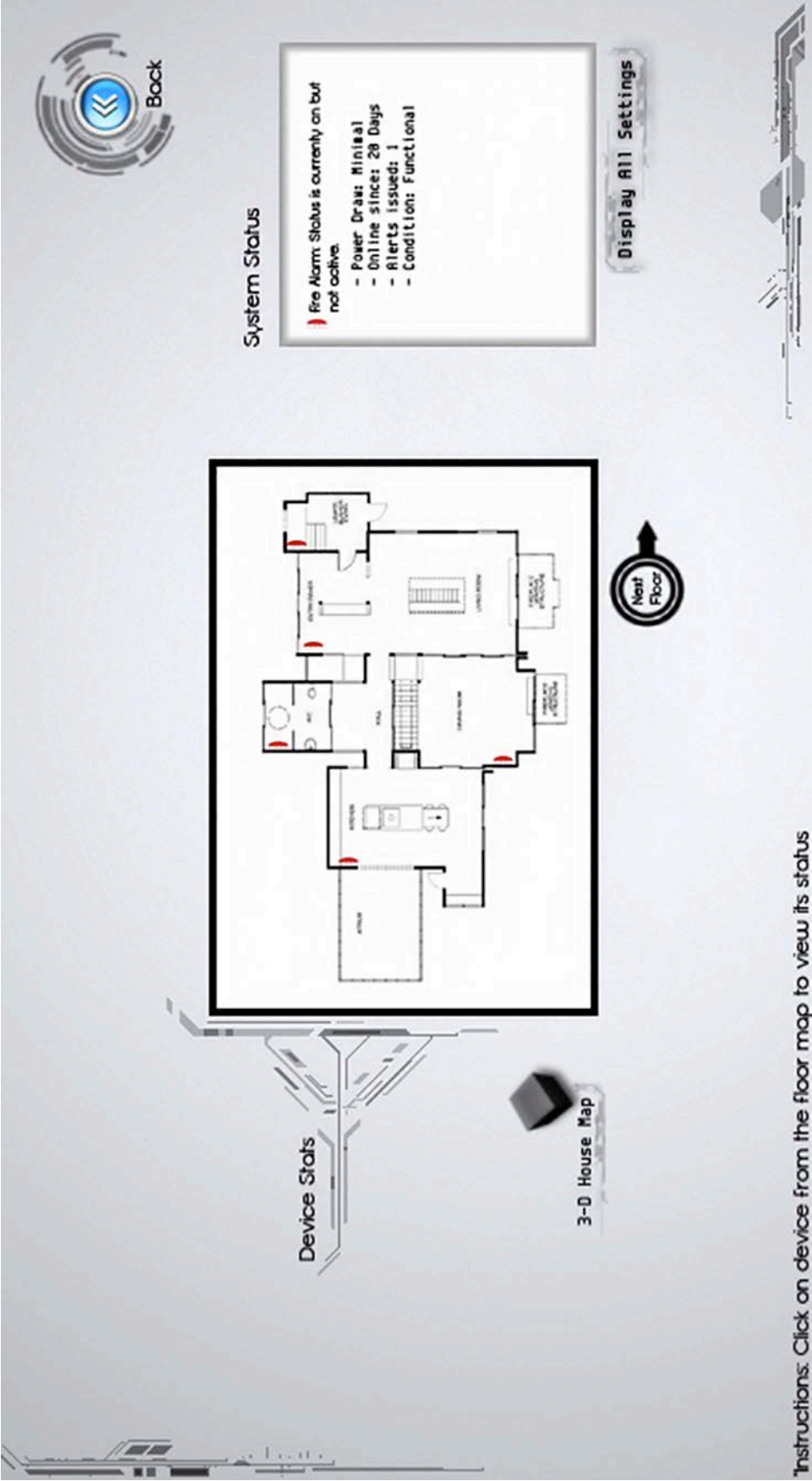


Figure 12.3 - Preliminary Rooms Page Design

Selecting the Status option will bring the user to this page where the user can check the status of the devices connected to the system. To eliminate any complexity a graphical interface is used where all the connected devices are shown in their respective location on the floor map, clicking the “next floor” button will change the floor, there is also a “previous floor” button as well but it only appears with the user is on a higher floor. The 3-D house map will show the entire house in a three dimensional view for asthetic purposes. The “Display All Settings” will show the basic working status of all the devices connected to the system. This section’s requirements stem from requirements 5 and 6, it would be trivial to include a completely different requirement since the status page basically utilizes the information from these requirements and display’s them in a much more presentable way. The 3D view is basically an asthetic option, which is there to show a more grandiose prespective of the house.

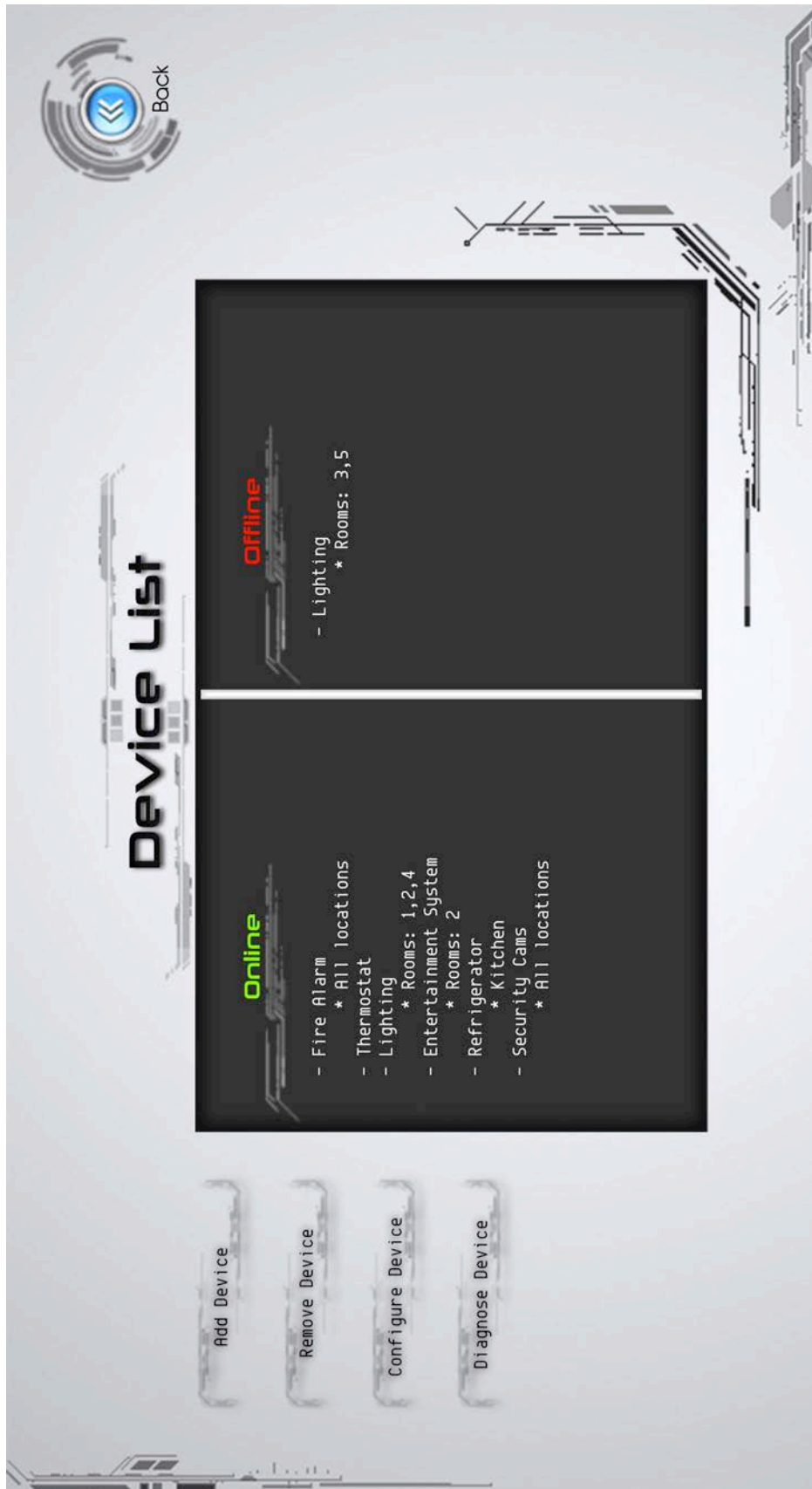


Figure 12.4 - Preliminary Devices Page Design

The Settings page gives the user the options to add, remove, configure or diagnose a device. The add and remove functions open a dialogue box where the user can simply select from a preconfigured list of common devices or manually set up a device. Remove device is very simple, clicking remove device will prompt the user to select a device from the center panel device list and the selected device disconnects from the system and no longer appears on the list. Diagnose device prompts the user to select a device from the list and automatically begins to check if there are any connection issues with the device, and reports the issue to the user. The device list has an online and offline section, this basically shows which devices are being used and which devices are not being used along with the locations of those devices.

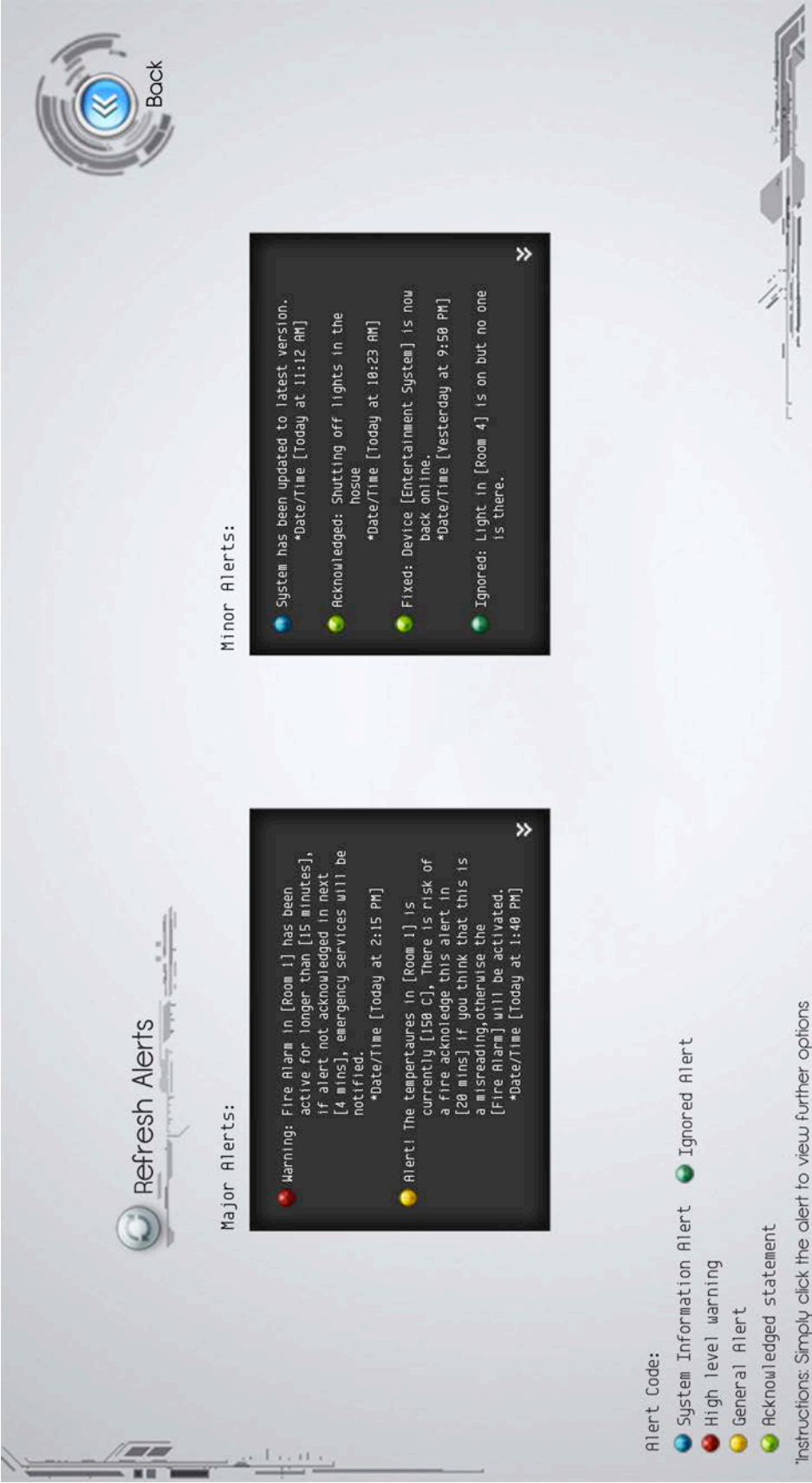


Figure 12.5 - Preliminary Logs & Alerts Page Design

The Alerts page displayed is separated into two columns major alerts and minor alerts, this allows the user quick access to the any messages generated by the system. The system itself can distinguish major and minor alerts quite easily, major alerts are generated if the alert is based on a data anomaly such as a major inconsistency in the data pool ie. If there is large change in room temperature that is not consistent with the previous data, constituting to a fire there; Minor alerts however, are basically just alerts that the user has acknowledged or any system based message such as updates and the daily activity from users. The major alerts section is separated so that the severe alerts are not buried within alerts that do not require as much attention. To act on the alert itself the user has to simply select the alert by clicking on it, this brings up a window with the following options: "Details", "Ignore", and "Close", the user would then have to click one of those options to proceed. Clicking on Details gives a much more elaborate description of the alert, such as where the alert is originating from, what caused the alert and a log of previous alerts of the same kind and the choice to acknowledge the alert, which would then be marked as acknowledged. Clicking details from any alert in the major alert's section gives the user the option to contact emergency services (which the system would automatically do if the user does not acknowledge the alert in a given time-span) if need be. Clicking ignore will simply ignore the alert, if this alert was in the major alert's section then it would just be moved into the minor alerts section; however, if the alert is in the the minor alert's section then the alert is just deleted. Close is quite self-explanatory it just closes the window without doing anything. The Alerts system is based on requirement 6, since it's a means of communicating with the user by reporting any issues or status-updates to them.



Figure 12.6 - Preliminary Rule Sets Page Design

The Device rules page allows the user to set rules to certain devices. This includes setting the lighting to turn on at a certain time. This allows flexibility to completely customize rules to how the user wishes to “run” the house. Rules are set and actions can be enabled to augment the rules.

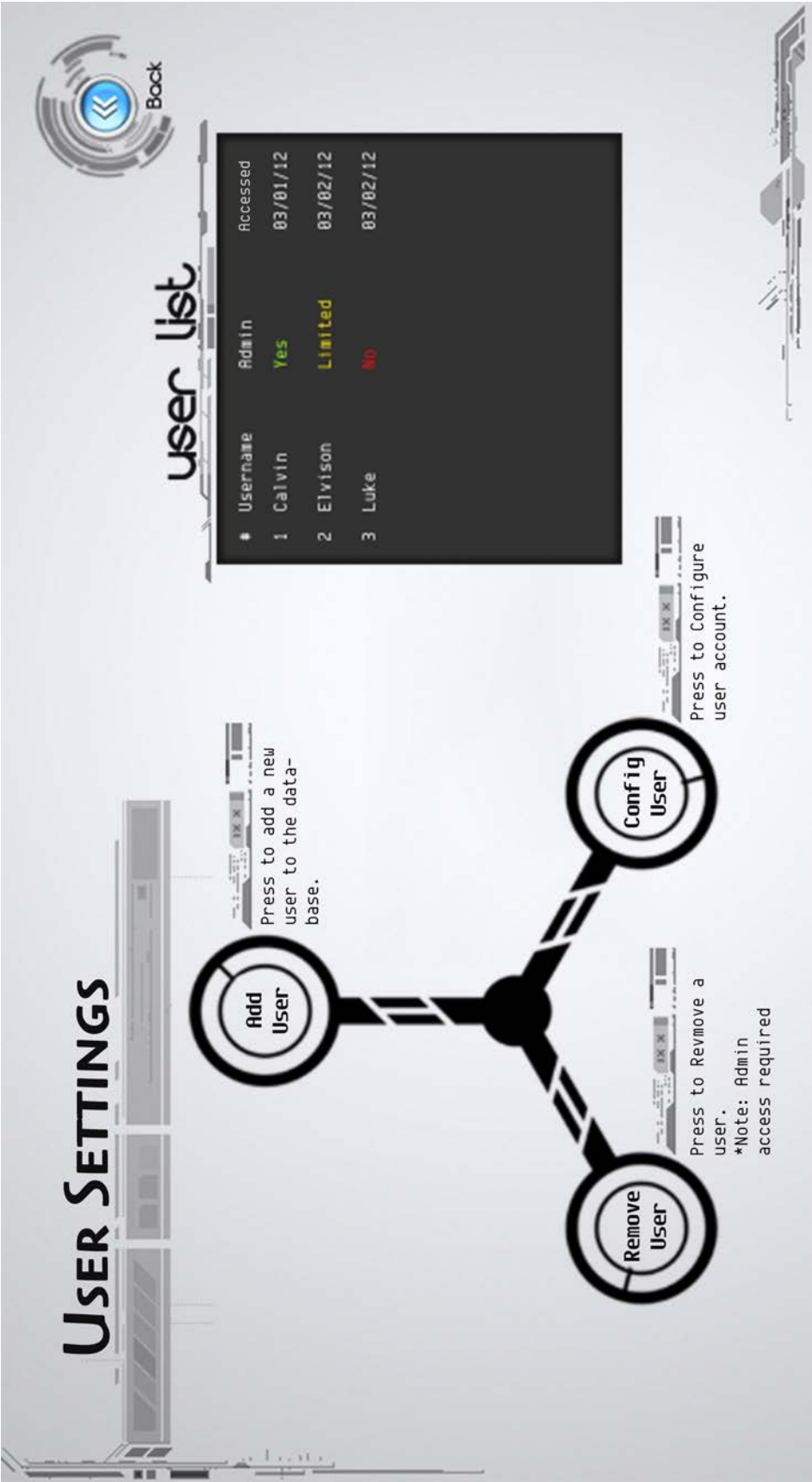


Figure 12.7 - Preliminary User Settings Page Design

The User settings page is basically a backend user management system. It allows users to configure their information, add new users and/or remove users. Adding a new user is quite basic, a system admin can add a user and give access permissions; these access permissions are meant to restrict non-administrative users from attempting to configure devices that are restricted to them, ie. Modules that control the fire-alarms are not accessible to non-administrative users unless the admin decides to give permission for that control and something as simple as removing a user from the database. The admin can give control to specific devices that are restricted either when adding a new user or by configuring the user profile at a later time, this would mark the user as limited in admin rights since they are unable to perform all administrative actions and only a select few that have been given to them. Removing a user as described before is also only possible by an administrator, this ensures a level of safety so normal users can't go about removing any user they wished, which might lead to issues down the road. Configure User, the selection of this brings up a dialogue which is slightly different for admin's and normal users. Normal user config window allows them to make profiles for device settings, change their information, and start/stop messaging services. The admin config window has all of the normal user's configuration options and another set of options which allow the admin to configure permissions for other users in the system. There is a user list window displayed for information purposes to show all the users in the database and the dates which they accessed the system.

b. Current Design and Implementations

Since we completed Reports 1 and 2, we decided to change the design of the website. We collectively came to the conclusion that it was not as user-friendly as we had anticipated it to be. The following pages will have pictures of our current implementation as well as brief descriptions of each part of the website.

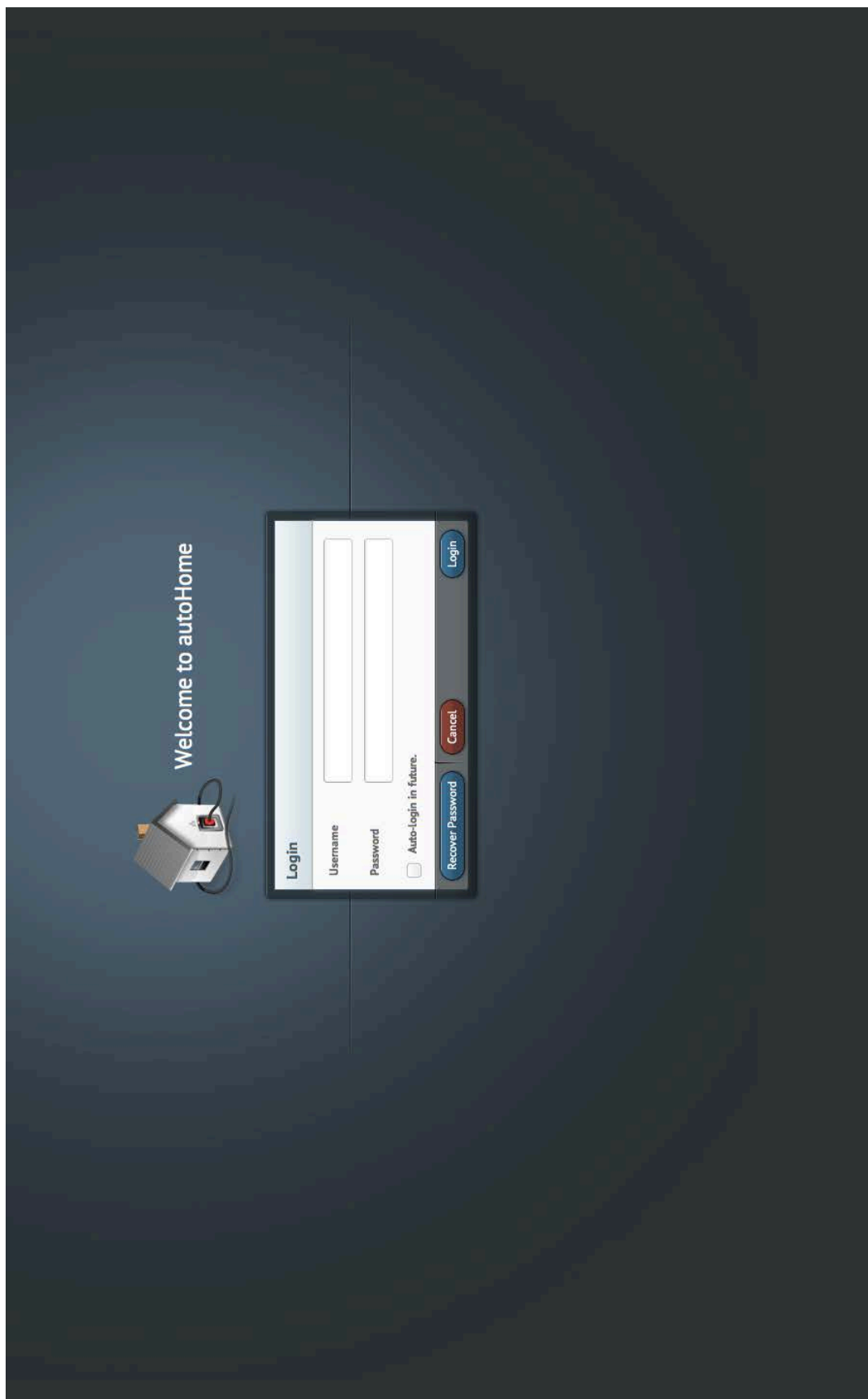


Figure 12.8 - Current Login Screen Implementation

From the beginning of the user's experience, autoHome presents the user with a clean, minimal, and very effective login screen. We found that user's felt they were using a quality product with this login screen, while also conveying the message that autoHome truly is revolutionary.

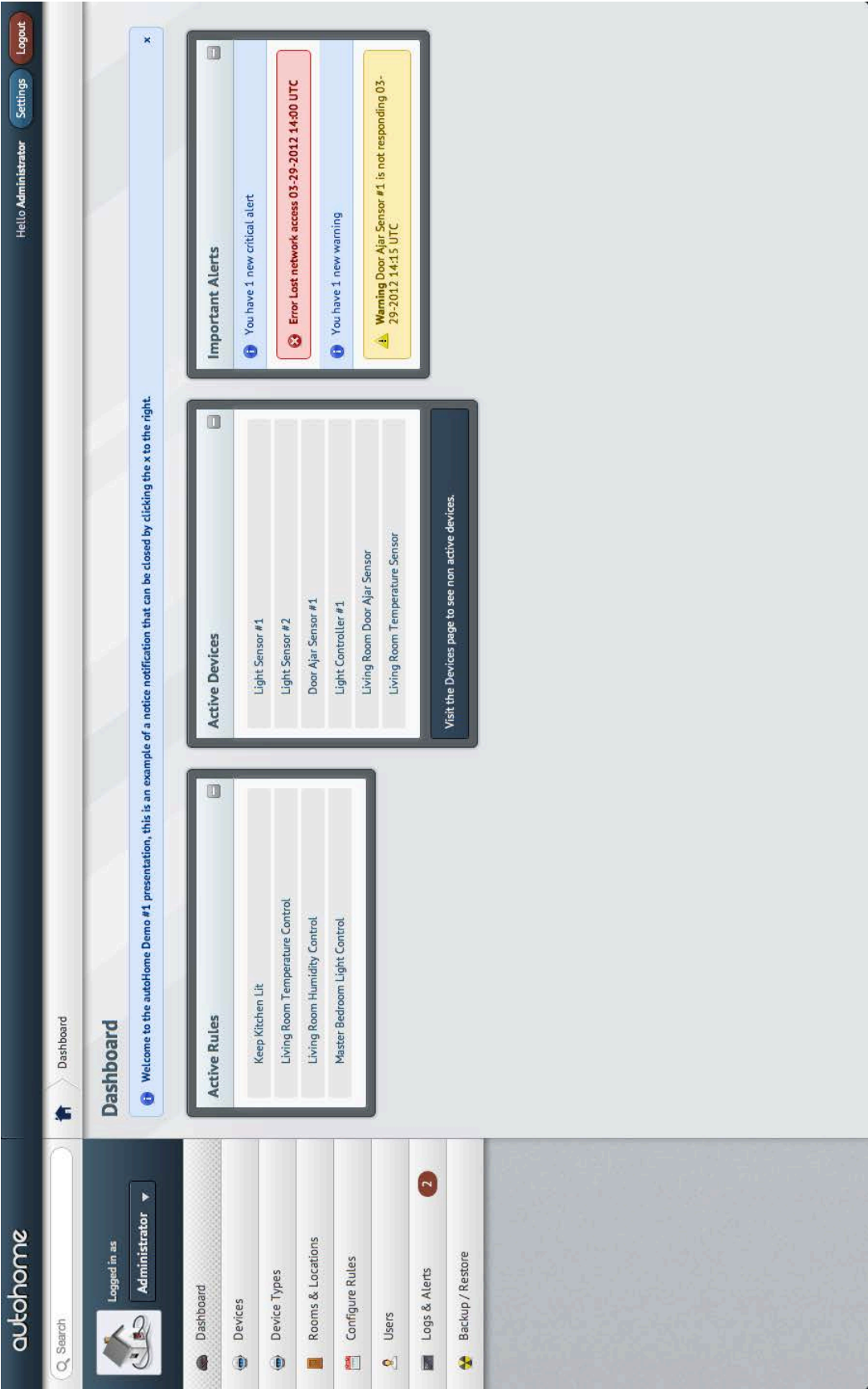


Figure 12.9 - Current Dashboard Page Implementation

The autoHome web interface was designed to be very user-friendly. Upon logging into the system, the user is greeted with their Dashboard. The autoHome Dashboard provides several important aspects including Active Rules, Active Devices, and Important Alerts. The listed Active Rules and Active Devices can be clicked on for additional information about the respective rules or devices. Important Alerts are color coded to catch the user's attention. Alerts are colored red for critical alerts, and yellow for warnings.

On the left side of the screen, the user is greeted with a number of options. The other pages that the user can access consist of Devices, Device Types, Rooms & Locations, Configure Rules, Users, Logs & Alerts and Backup/Restore. We will traverse the list from top to bottom and talk about each page.

Figure 12.10 - Current Devices Page Implementation

The Devices page shows the user all devices that are currently connected to the system. The Dashboard simply shows the user active devices on the system (devices that are currently “on” in the house), whereas on the Devices page, all devices are shown regardless of activity. Devices are listed in a tabular format to make it easier to read. The device table shows the device name, what type of device it is, what room the device is located in, the state of the device (if it is on or off), the default value of the device, and the options to show all information about the device in a new window (separate from the table), edit the device, or destroy (delete) the device from the system. In addition, on the top-right corner of the table, the user can click on “Add New Device” to add a new device to the system and specify their own values for each category in the table. The tabular format also allows for a number of additional functionalities. First, the user can sort the table based on any of the aforementioned categories. There is also the functionality to search the table for keywords that appear in any category. This is especially useful since in a house, there will be a lot of different devices, device types, and rooms. If a user types “kitchen” into the search box, all devices that are located in the kitchen will only appear. This proves to be especially useful when looking for a single device or a cluster of related devices (e.g. all devices in the kitchen). The tabular format makes it easier to have many entries (devices) and allows them to be displayed on multiple pages. Thus, our Devices page has page functionalities such as First, Previous, Next, Last, and a list of all page numbers for travelling to a specific page, directly built into the table.

autohome

Q Search

Logged in as Administrator

Dashboard

Devices

Device Types

Rooms & Locations

Configure Rules

Users

Logs & Alerts

Backup / Restore

2

Device Types

Device Types

Create New Device Type

Search:

Name	Module name	Data type	Data flow	Unit			
Door Ajar Sensor	door_ajar_sensor	binary	sensor	boolean	Show	Edit	Destroy
Light Controller	light_controller	analog	controller	Volts	Show	Edit	Destroy
Light Sensor	light_sensor	analog	sensor	Lux	Show	Edit	Destroy
Temperature Sensor	Temperature_Sensor	analog	sensor	Celsius	Show	Edit	Destroy

Showing 1 to 4 of 4 entries

First Previous 1 Next Last

Figure 12.11 - Current Device Types Page Implementation

Next on the list is the Device Types page. This page provides a very similar look to the Devices page in terms of the same tabular format. However, the categories in the table for this page consist of the name of the device type, the device type name that the system recognizes (module name), data type of the device (either analog or binary), data flow of the device (the units in which the device measures, e.g. volts, Celsius), and the same options of show, edit, and destroy. Just like the Devices page, users can search the table and sort it as they please. The user can also add a new device type to the table, specifying each value, in every category, as they desire.

autohome

Q Search

Logged in as Administrator

Dashboard

Devices

Device Types

Rooms & Locations

Configure Rules

Users

Logs & Alerts

Backup / Restore

Rooms and Locations

Rooms & Locations

Hello Administrator

Settings

Logout

List of Rooms

Show 1 entries

Name	Floor	Description	Show	Edit	Destroy
Adopted Child's Room	2	Minimal features included	Show	Edit	Destroy
Basement	0	BAT CAVE!	Show	Edit	Destroy
Dining Room	1	The place where family never eats together	Show	Edit	Destroy
Kitchen	1	The place of culinary delights.	Show	Edit	Destroy
Little Timmys Room	2	Timmy's Play House	Show	Edit	Destroy
Living Room	1	The place where everyone goes to waste their lives in front of televisions.	Show	Edit	Destroy
Master Bathroom	2	Cool spa jets and bubble bath	Show	Edit	Destroy
Master Bedroom	2	Where mommy and daddy go to.....sleep	Show	Edit	Destroy
Office	1	KEEP OUT SERIOUS BUSINESS	Show	Edit	Destroy
Upstairs Bathroom	2	Just a typical bathroom / Android game center	Show	Edit	Destroy

Showing 1 to 10 of 10 entries

List of Locations

Show 1 entries

Name	Room	Description	Show	Edit	Destroy
Bottom-most Cupboard	Kitchen	The cupboard right above the stove.	Show	Edit	Destroy
Main Entrance	Living Room	Main Entrance opposite to the television and closer to the stairs.	Show	Edit	Destroy
Topmost Cupboard	Kitchen	Cupboard most to the top above the stove.	Show	Edit	Destroy

Showing 1 to 3 of 3 entries

Figure 12.12 - Current Rooms & Locations Page Implementation

The next page is the Rooms & Locations page. This page is also in a tabular format and shows all rooms and locations in the house. The show, edit, and destroy options are also included in the same, tabular style as the other pages. It may seem confusing at first as to what the difference between rooms and locations are. Rooms are rooms in the house, whereas locations are the locations of where the devices are in a certain room. Clicking on a certain room or location will bring up another page with all devices in the specified room. Just like other page functionalities, the user can add a room or location if they desire, specifying the room or location name, the floor that it is on, and a description of it.

autohome

Q Search

Logged in as Administrator

Dashboard

Devices

Device Types

Rooms & Locations

Configure Rules

Users

Logs & Alerts

Backup / Restore

Configure Rules

Configure Rules

Settings

Logout

Search:

New Rule Set

Name	Condition set	Action set	User	Description	Active	Show	Edit	Destroy
Keep Kitchen Lit	Kitchen Lights are Dim	Turn on Kitchen Lights	Megaman Awesomesauce	Keeps the kitchen bright!	true	Show	Edit	Destroy
Living Room Humidity Control	People are in the house.	Living Room Humidity Control	Calvin the Pineapple Juice	Keep humidity constant at 70% to make the room comfy	true	Show	Edit	Destroy
Living Room Temperature Control	People are in the house.	Living Room Temperature Control	Calvin the Pineapple Juice	Keeps the living room at a comfy temperature	true	Show	Edit	Destroy
Master Bedroom Light Control	No one is in the master bedroom	Room Light Control	Calvin the Pineapple Juice	Turn off lights to save energy when no one is present	true	Show	Edit	Destroy

Showing 1 to 4 of 4 entries

Listing condition_sets

Name	Description	User
Kitchen Lights are Dim	Check to see if the kitchen is unlit.	Megaman Awesomesauce Show Edit Destroy
Living Room Doors Open	Are all the doors in the Living Room open?	Megaman Awesomesauce Show Edit Destroy
People are in the house.	People are in the house.	Sonic the Hedgehog Show Edit Destroy
No one is in the master bedroom	No one is in the master bedroom	Calvin the Pineapple Juice Show Edit Destroy

New Condition set

Listing action_sets

Name	Description	User
Turn on Kitchen Lights	Turns on all the kitchen lights that are being controlled.	Megaman Awesomesauce Show Edit Destroy
Living Room Temperature Control	Keep temperature at 78F	Calvin the Pineapple Juice Show Edit Destroy
Living Room Humidity Control	Keep Humidity at 70%	Calvin the Pineapple Juice Show Edit Destroy
Room Light Control	Turn off lights	Calvin the Pineapple Juice Show Edit Destroy

New Action set

Figure 12.13 - Current Configure Rules Page Implementation

Configure Rules page is the most complex of the pages. The user is provided with three different tables; Rule Sets, Condition Sets, and Action Sets. Each table has the same search and sorting functionalities as the other tables. Rule Sets use both an Action Set and Condition Set to form a Rule Set. Thus, the user should start by either making an Action Set or a Condition Set, let's start with an Action Set. The Action Set table provides the name of the Action Set, a description, the user who made it, as well as the show, edit, and destroy options found in the other tables. An Action Set is simply something that the system should do when a certain condition is met. So, this table allows the user to set the action they want to happen. However, now the user must make a condition set in order for the action to initiate. The Condition Set table is identical to the Action Set table, but obviously the user would want to set conditions rather than actions in the Condition Set table. Now that the Condition Sets and Action Sets table entries are entered, the user can categorize them into Rule Sets. The Rule Set table has the categories of name, the condition set it uses, the action set it uses, the user that created it, the description of the Rule Set, if the Rule Set is active or not, and the generic show, edit and destroy options.

autohome

Q Search

Logged in as Administrator

Dashboard

Devices

Device Types

Rooms & Locations

Configure Rules

Users

Logs & Alerts

Backup / Restore

Hello Administrator

Settings

Logout

Users

1 entries

Search:

Name	Last login	Administrator	Password hash	Last ip	Phone number			
Calvin the Pineapple Juice	2012-03-29 05:49:00 UTC	true	mixedfruit	192.168.1.1	978 564 3412	Show	Edit	Destroy
DJ Pauly Kay	2012-03-30 15:14:00 UTC	false	everydaymshufflin	192.168.1.1	9081974283	Show	Edit	Destroy
Elle Luau-Dancer	2012-03-30 03:19:00 UTC	false	strawskirt	192.168.0.1	1230124251	Show	Edit	Destroy
Megaman Awesomesauce	2012-03-28 06:43:00 UTC	true	testpass	192.168.1.1	7325555555	Show	Edit	Destroy
Rohith Hotsauce with Butter	2012-03-03 00:00:00 UTC	false	CountToPotato	192.168.0.1	852 99912345	Show	Edit	Destroy
Sonic the Hedgehog	2012-03-28 07:04:00 UTC	true	invalidpassword	192.168.1.1	7324445555	Show	Edit	Destroy

Showing 1 to 6 of 6 entries

First Previous 1 Next Last

New User

Figure 12.14 - Current Users Page Implementation

The Users page is very straightforward. The table lists the User Name, last login time, if the user is an administrator or not, their hashed password, last IP address they used to log in, their phone number, and the show, edit, and destroy options. The table adopts the same search and sort functionalities, and users can create additional user accounts for the system.

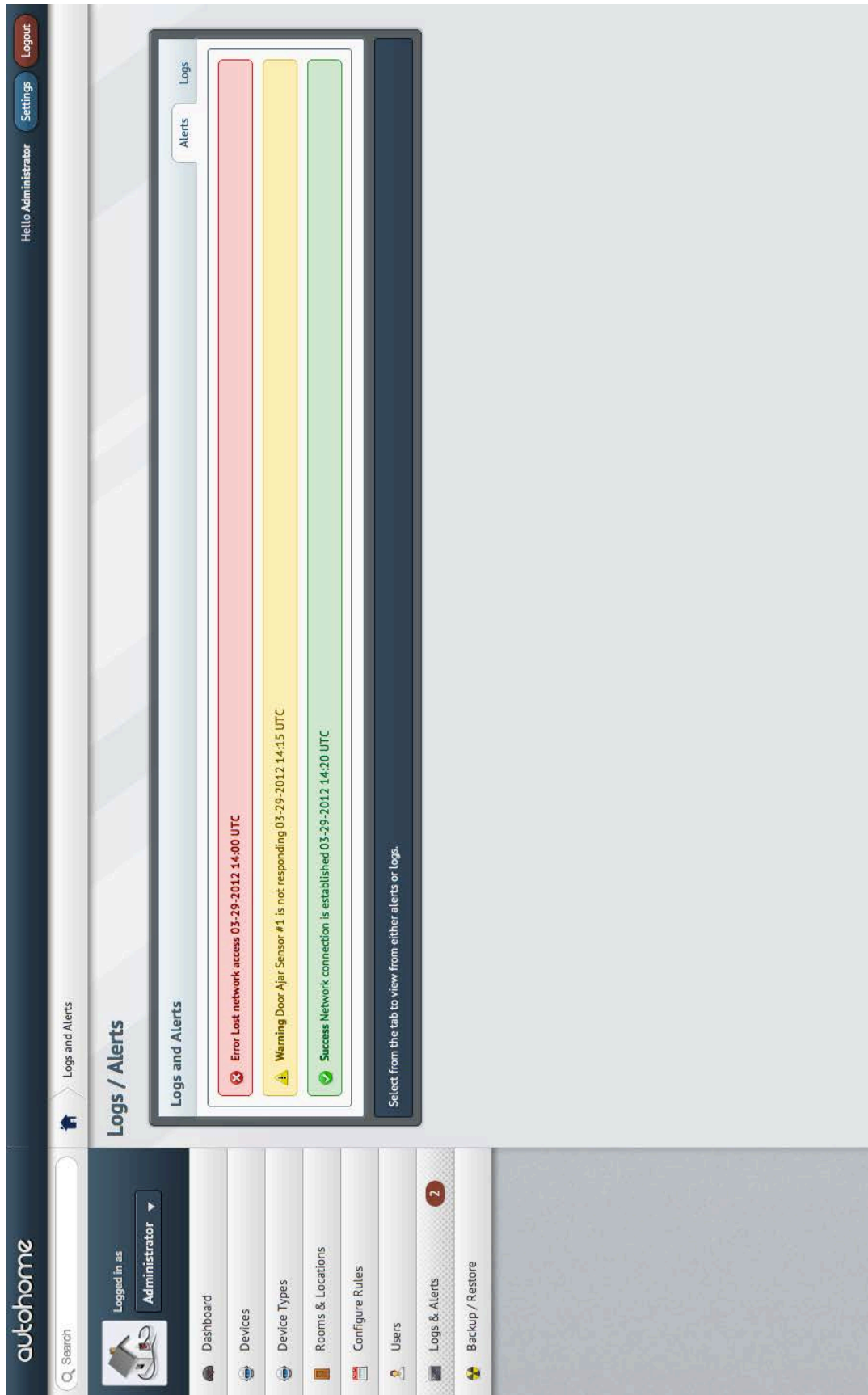


Figure 12.15 - Current Logs/Alerts Page Implementation

The Logs & Alerts tab on the left of the page displays a number next to it to inform the user of any alerts they may have missed in the Dashboard. The Logs & Alerts page shows alerts by default, including all critical alerts, warnings, and successes of the system. The Logs tab on the page shows all events that have happened in the system since the system was last booted.

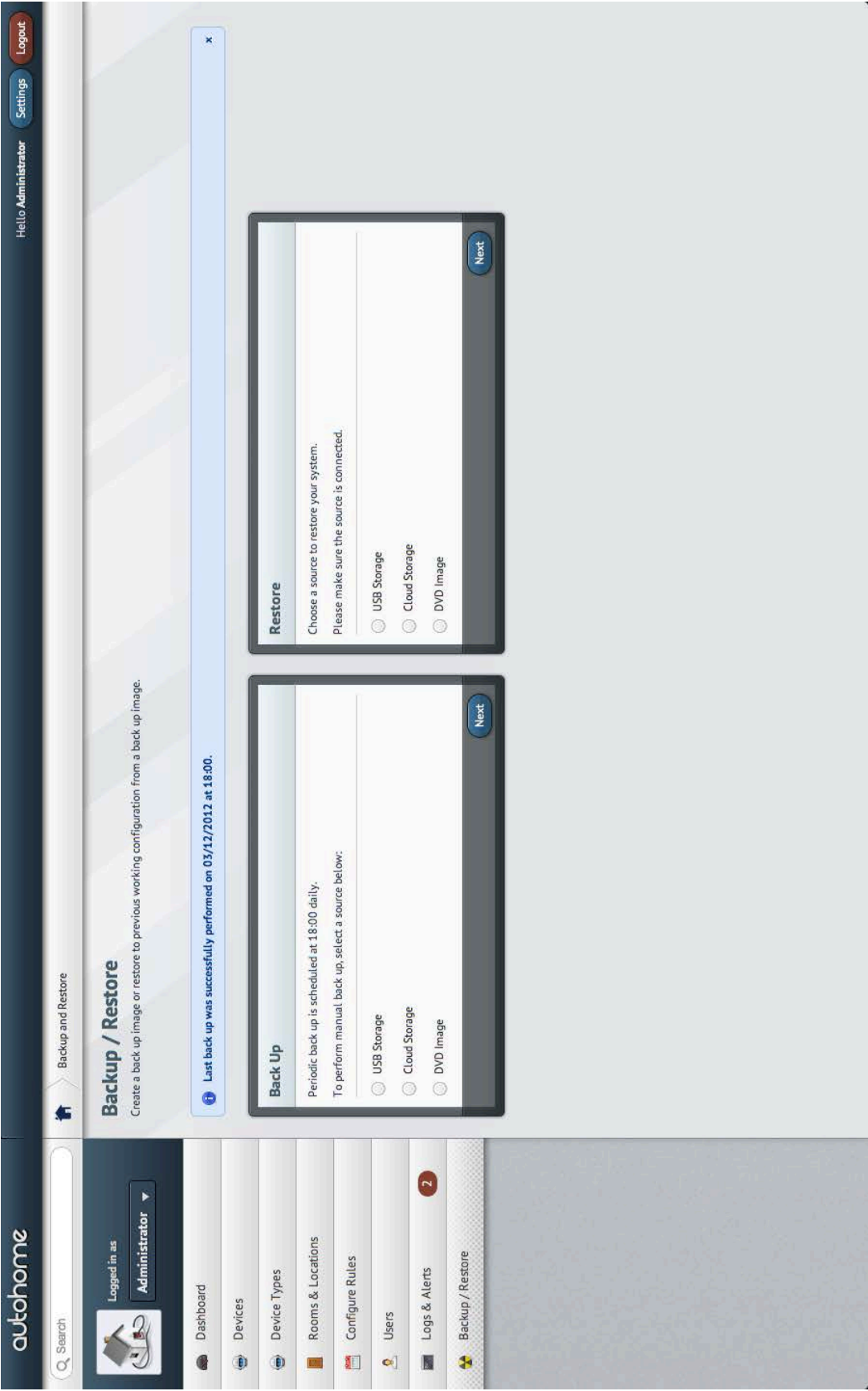


Figure 12.16 - Current Backup/Restore Page Implementation

The last tab on the side of the autoHome page is Backup/Restore. This page allows the user to start the system backup or restore process. The page prompts the user to select a backup or restore device, and to then continue the process to finish backing up or restoring the autoHome system. In addition, the top of the page provides a notification of when the last backup was performed. A backup is scheduled on a daily basis to ensure that if the system were to crash, the user would lose as little as possible.

Overall, the autoHome web interface is laid out in a clean manner, while also being highly effective for the user. We felt that this layout was much more effective and streamlined for a satisfactory user experience.

13. Design of Tests

a. Casual Test Cases

Use Case 1: Authenticate User

A user logs in to his/her account in order to configure the alarm system. The tests confirm that the correct passwords and usernames allow the logins to the correct account.

Use Case 4: Configure User Device Permissions

An administrator grants users the ability to access a specific device using the web-interface. Access to a device allows the user to create Rules polling from the device if it is a sensor or Actions commanding the device if it is a controller. The tests ensure that the device data is updated and log data is accessible.

Use Case 6: Display Device Information

A user displays information about a device that the user has access to. The web interface will bring up information including the device location, the device name, device ID, and data log information about the device's history in tabular or graph format.

Use Case 12: Add Device

An administrator will log in and configure a new device. The administrator shall add ID, driver type, name, and location to the database through the web interface. A successful test will result in a device usable to the autoHome system.

Use Case 14: Send Device Signal

A value is sent to a device driver to be converted to a correct voltage, and then sent through controlling conductors to set device values. Tests will assure that the software-set values will accurately reflect the values actually set in the physical device.

User Case 15: Receive Device Signal

A voltage is read by the device driver and converted to a valid value usable by humans. The tests assure that the values in the software reflect the values of sensors.

Use Case 18: Send Virtual Device Signal

A programmed routine is executed by the system with specified parameters. The routine can be anything from text messaging, to phone calls, to website updates. A successful test will result in these actions being successfully taken.

Use Case 19: Receive Virtual Device Signal

The autoHome system may receive signals from virtual devices such as system time, internet weather, or external information. The test will assure that the information pulled from external sources is correct.

b. Descriptive Test Cases

Use Case 1: Authenticate User

Success

- User inputs username and password successfully. The system has to verify if the username and password exists and matches in the system. The match is made and the user is logged into his/her account.

Failure

- User inputs username and password, but when the system goes to verify, the username is incorrect and it doesn't match any username in the database and it displays a "wrong username/ password" error message.
- User inputs username and password, but when the system goes to verify, the password does not match with the one saved in the database and it displays a "wrong username/ password" error message.
- User inputs username and password, but when the system goes to verify, neither the username nor the password match with the one saved in the database and it displays a "wrong username/ password" error message.

Use Case 4: Configure User Device Permissions

Success

- The authenticated administrator signs into his/her account successfully. From this point, he/she goes to the settings of the security device. Within this menu, he/she enters the menu of the individual sensors or controllers to the security device and enables the sensors and controllers to their given security device.
- The administrator is able to add users to command to a particular device. These users will now have access to use the device in their rule sets.

Failure

- The authenticated administrator signs into his/her account successfully. From this point, he/she goes to the settings of the security device. Within this menu, he/she enters the menu of the individual sensors or controllers to the security device and tries to enable the sensors and controllers of the given security device for another user to use, but one or more permissions did not follow through and it denies access to some or all of the security devices.

- The authenticated administrator signs into his/her account successfully. From this point, he/she goes to the settings of the security device. Within this menu, he/she tries to enter to the menu of the individual sensors or controllers to the security device but the autoHome interface displays a different menu from that isn't related to security devices or security device permissions.
- A user tries to change the permissions to the security device, but an error message pops up displaying that only an authenticated administrator has access to change the device permissions.
- A user with permissions to a device is denied access to use in their rule sets.

Use Case 5: Configure Device Rule Sets

Success

- An authenticated user logs into his/her account successfully. He/she then manages his/her way down to the security device list menu. In this menu, he/she will be able to select a security device and set the different schedules and conditions for the given security device and/or security sensors successfully. By doing this, the user will be able to turn on the light switches on the sensors and the motion sensors for the alarms successfully.
- When the condition sets for an enabled rule set are matched, the corresponding action set will be executed.

Failure

- The authenticated user logs into his/her account successfully. He/she then manages his/her way down to the security device list menu, but the autoHome interface displays a different menu from that isn't related to the security device list device rule sets.
- The authenticated user logs into his/her account successfully. He/she then manages his/her way down to the security device list menu. In this menu, he/she will want to select a security device and set the different schedules and conditions for the given security device and/or security sensors but the system configures a different schedule or condition instead of the one intended.
- The authenticated user logs into his/her account successfully. He/she then manages his/her way down to the security device list menu. In this menu, he/she will want to select a security device and set the different schedules and conditions for the given security device and/or security sensors but the system doesn't turn on the lights for the sensors at the given time, or it manages to set the time for a different device.
- The condition set does not trigger its related action set.

Use Case 6: Display Device Information

Success

- The authenticated user logs into his/her account successfully. He/she then manages his/her way down to the security device list menu. In this menu, he/she will be able to select any security device and be able to view where the security device is located on a floor map and view the data values the security device is producing, in text and/or graphs. The autoHome database will keep all of the data logs stored in memory in case of any error occurs within the system.
- Device settings in the database are displayed clearly and correctly in the user interface.

Failure

- The authenticated user logs into his/her account successfully. He/she then manages his/her way down to the security device list menu, but the autoHome interface displays a different menu from that isn't related to the security device list or its information.
- The authenticated user logs into his/her account successfully. He/she then manages his/her way down to the security device list menu. In this menu, he/she will be able to select any security device and be able to view where the security device is located on a floor map and view the data values the security device is producing, in text and/or graphs, but the interface only shows the floor maps to the security device, but not the values nor the text or graph data.
- The authenticated user logs into his/her account successfully. He/she then manages his/her way down to the security device list menu. In this menu, he/she will be able to select any security device and be able to view where the security device is located on a floor map and view the data values the security device is producing, in text and/or graphs, but the interface doesn't show the floor maps, and only shows the texts and/or graphs.
- The authenticated user logs into his/her account successfully. He/she then manages his/her way down to the security device list menu. In this menu, he/she will be able to select any security device and be able to view where the security device is located on a floor map and view the data values the security device is producing, in text and/or graphs, but the interface doesn't any type of data
- The authenticated user logs into his/her account successfully. He/she then manages his/her way down to the security device list menu. In this menu, he/she will be able to select any security device and be able to view where the security device is located on a floor map and view the data values the security device is producing, in text and/or graphs, but the interface shows data of other devices in the house and not the security devices.

Use Case 12: Add Device

Success

- An authenticated administrator logs into his/her account successfully. He/she enters the device list menu and clicks on the “add new device” option and then the user will install the device manually and the system will be able to read it automatically. Then the administrator will enable the device accordingly.

Failure

- An authenticated administrator logs into his/her account successfully. He/she enters the device list menu and clicks on the “add new device” and it fails to work.
- An authenticated administrator logs into his/her account successfully. He/she enters the device list menu and clicks on the “add new device” and it works, but then when the administrator attaches the security device manually, the system won’t read it.
- An authenticated administrator logs into his/her account successfully. He/she enters the device list menu and clicks on the “add new device” and it works, then the administrator attaches the security device and when he/she tries to enable it, it does not enable.

Use Case 13: Remove Device

Success

- An authenticated administrator logs into his/her account successfully. He/she enters the device list menu and clicks on the “remove device” option and then the administrator will disable the device accordingly. The user will remove the device manually.

Failure

- An authenticated administrator logs into his/her account successfully. He/she enters the device list menu and clicks on the “remove device” and it fails to work.
- An authenticated administrator logs into his/her account successfully. He/she enters the device list menu and clicks on the “remove device” and it works, but then when the administrator detaches the security device manually, the system still believes the security device is still attached.
- An authenticated administrator logs into his/her account successfully. He/she enters the device list menu and clicks on the “remove device” and it works, but when the administrator tries to disable the device, the system still leaves it enabled and it refuses to disable the security device, resulting the device to still is presentable within the autoHome user interface. This will result in false and inaccurate data when monitoring the entire house’s other devices.

Use Case 14: Send Device Signal

Success

- Without flaw, the system specifies values to the set security device that are currently connected. Then the device interfacing modules convert the values into meaningful voltages to set the controlling conductors. When no device is present, the interfacing modules provide a default value to be held for the controllers.

Failure

- The system specifies the wrong values to the set security device that is currently connected.
- The interfacing modules don't convert the values to meaningful voltages or they convert the values into a higher or lower than needed voltage for the specified security device value.
- When the device or devices are not present, the interfacing modules stay with the previously given voltage value for the device and it does not recognize that the device has been removed.
- The interfacing module configures the voltage and specified value from the security device to another device in the home, and/ or vice versa.

Use Case 15: Receive Device Signal

Success

- Without flaw, the system receives values from the set security device that are currently connected. Then the device interfacing modules convert the values into meaningful voltages to set the controlling conductors. When no device is present, the interfacing modules provide a default value to be held for the controllers.

Failure

- The system receives the wrong values from the set security device that is currently connected.
- The interfacing modules don't convert the values to meaningful voltages or they convert the values into a higher or lower than needed voltage for the received security device value.
- When the device or devices are not present, the interfacing modules stay with the previously given voltage value for the device and it does not recognize that the device has been removed.
- The interfacing module configures the voltage and received value from the security device to another device in the home, and/ or vice versa.

Use Case 18: Send Virtual Device Signal

Success

- The system successfully sets an active value to the system to use as a controller value in the rule set. The user can check the values of the text message profiles, email profile, and shell-script profile IDs in the interfacing module.

Failure

- The system unsuccessfully sets the active values to the system to use as a controller value in the rule set, and inputs different values or possibly none.
- The user is unable to enter the interfacing module to view the ID profiles for the security device that is wanted to be checked.

Use Case 19: Receive Virtual Device Signal

Success

- The system successfully receives an active value to the system to use as a sensor value in the rule set. The user can check the values of the system time, system date, and information such as when the sensor detected a movement in the interfacing module.

Failure

- The system unsuccessfully receives the active values to the system to use as a sensor value in the rule set.
- The system receives the active values to the system but configures invalid values or possibly none.
- The user is unable to enter the interfacing module to view the ID profiles for the security device that is wanted to be checked

Use Case 20: Evaluate Data Against Rule Sets

Success

- New data is received and it is checked against the relevant active rule sets that contain the data value as a condition in its condition set. Then when the condition is met, the associated action set will be executed. The user will try to break into his own home or try to move in front of the motion sensors and see if he/she would receive a text message or email, and to see if the alarm would go off.

Failure

- No data is received
- New data is received but when checked against the relevant active rules sets; it does not match towards its condition set.

- When the user tries to break into his/her own home or try to move in front of the motion sensors, he/she does not receive any alert messages
- When the user tries to break into his/her own home or try to move in front of the motion sensors, the alarm does not go off.

14. Past, Present and Future

a. History of Work

Our actual milestones and deadlines evolved quite a bit through the course of this project. Firstly, they evolved in such a way that all group members had various obligations for other classes including test, projects, presentations, etc. These obligations made our deadlines vary throughout the course of the semester. However, we still achieved our milestones by the deadline date. I believe it is very hard for college students to create a timeline, or plan of work, on such a large scale project when all the students working in the group have other obligations for classes or organizations. A plan of work is most efficiently used in a real-world setting where employees only have the project to worry about and can put all of their efforts towards certain parts of the project. Nonetheless, I think our group did a **very** good job at achieving what was planned in such a timely manner.

b. Key Accomplishments

- ▶ As a team, we started out rough, however, we quickly began to work more efficiently as a team.
- ▶ Following that, our overall grades increased with each piece. We started with an 80 on report 1, then a 90 on report 2 (very close to the highest grade in the class which was a 93), and then an 85 on demo 1 (the highest grade in the class). I believe that this not only shows continued efforts, but it also shows that the team learned how to work better together.
- ▶ Our first demo allowed us to show off an implementation of, what was once, merely a concept.
- ▶ Throughout the span of the project, the number of conflicts amongst the team were minimal.
- ▶ The autoHome web interface came out better than we had ever anticipated.
- ▶ For the second demo, hardware was used to show that the web interface can interact with actual hardware and proved the concept of our project effectively.

c. Current Status of Implementation

Currently, the team has autoHome in an excellent position for the time we were provided. Presently, the autoHome web interface is fully functional. Users are able to check their Dashboard for important updates regarding the autoHome system, as well as check the weather or even check their favorite stocks right through the Dashboard with our integrated widget system. Additionally, the functionalities of adding and removing users, rooms, devices, device types, rulesets, condition sets, conditions, action sets, actions, rooms, and locations. Our web interface is near fully-functioning and provides close to all functionalities needed to use autoHome most efficiently. In addition to the web interface, we have a “proof of concept” hardware demo where we have shown that the web interface can interact with hardware. We used a light sensor to give precise readings of light along with a series of five LED’s. With this, we showed off how the light sensor

can read a certain value, or range of values, and according to the rulesets in place, can change the states of the LED lights as a group or individually. Our hardware demonstration, paired with our web interface, truly showed that autoHome clearly went from a concept, to something much more realistic in the span of one semester.

d. Future Work

autoHome is a very sophisticated system. Despite our best efforts, the entire system that the team had in mind cannot be implemented in the course of only one semester. Currently, our web interface is implemented and provides the functionalities of adding and removing users, rooms, devices, device types, rulesets, condition sets, conditions, action sets, actions, rooms, and locations. Since the system is not in a consumer household, we did not see a need to implement the Backup / Restore option completely. This is one, among many things, that we plan to accomplish in our future work.

Additionally, for our second demonstration, we introduced the concept of widgets in the autoHome dashboard. We showed off weather and stocks, but we will obviously expand on these in the future. There are an endless number of widgets that could be made and implemented into the autoHome dashboard. Varying from a calendar, notepad, sport scores, or even a quick menu where the user can quickly set to turn on/off their favorite devices; the possibilities are endless.

During our second demonstration, we also demonstrated how autoHome's web interface can interact with hardware. We demonstrated a light sensor and how the light sensor interacted with a PC and displayed the light sensor readings on-screen. In the future, we plan to implement these readings into the web interface so that the user can see the exact reading of every sensor and set their rulesets accordingly.

We also expressed how simple it is to just remove the light sensor and implement a different sensor with a different set of devices (instead of the LED's we used). This showed how versatile our autoHome system can be, and how incredibly simple it is to install new sensors and new devices to the system.

In our reports, we mentioned the autoHome system being implemented into houses. We also made a point to realize that a lot of wiring would have to be done and it would be better suited for new houses that are still being built. However, we plan to further expand our horizons and do in-depth research on how we can use technology such as RFID tags to further simplify our system while also cutting down on necessary wiring in the house.

Lastly, we plan to further expand our autoHome web interface to include additional "user account settings" where a user can set a certain account to "child," allowing them only to access certain parts of the web interface. We also plan to expand upon the rooms and locations part of the website to provide a floor-plan of the house and have it highlighted in certain colors according to the number of devices on in that particular area of the house.

We also plan to monitor electricity usage with autoHome, and allow the system to record, and keep track of the usage on a daily basis. This would allow the user to see exactly what is using up the most electricity in the house.

Overall, despite the enormous milestones we have achieved during this project, there will always be room for improvement, and the new ideas will keep coming out, allowing us to make the autoHome system better and better with each passing day.

15. References

- [1] FURPS model - <http://en.wikipedia.org/wiki/FURPS>;
<http://www.ibm.com/developerworks/rational/library/4706.html#N100A7>
- [2] Requirements analysis - http://en.wikipedia.org/wiki/Requirements_analysis#Stakeholder_identification
- [3] Cloud based backup solution - <http://www.backblaze.com/>
- [4] jQuery documentation and usage examples - http://docs.jquery.com/Main_Page
- [5] Arduino microcontroller - <http://arduino.cc/>
- [6] Version control system - <http://www.github.com>
- [7] Infrared motion detector - http://en.wikipedia.org/wiki/Passive_infrared_sensor
- [8] Temperature / humidity monitor POE - <http://www.itwatchdogs.com/product-detail-microgoose-9.html>
- [9] Water level sensor - <http://www.amazon.com/United-Security-Products-Water-Sensor/dp/B001HLAOX4>
- [10] Light intensity sensor, outdoor - <http://www.onsetcomp.com/products/sensors/light-intensity>
- [11] How to measure electricity - http://en.wikipedia.org/wiki/Electric_power
- [12] How to measure water - http://en.wikipedia.org/wiki/Flow_measurement#Liquid
- [13] Coupled Induction for charging devices - http://en.wikipedia.org/wiki/Mutual_inductance#Coupled_inductors
- [14] House blueprint - <http://homeinteriortrend.com/wp-content/uploads/2011/06/sketch-smart-house-design-6.jpg>
- [15] Example as stated on the report format - http://en.wikipedia.org/wiki/Software_architecture#Examples_of_architectural_styles_and_patterns
- [16] United States Government Time - <http://time.gov/>
- [17] Android Operating System Documents - <http://www.android.com/about/>

- [18] FTDI communication for embedded devices - <http://www.ftdichip.com/>
- [19] The ActiveRecord development pattern http://en.wikipedia.org/wiki/Active_record_pattern
- [20] USB Relay <http://www.relaycontrollers.com/Relay/Device/UADR85PROXR>
- [21] Home Automation Coverage http://en.wikipedia.org/wiki/Home_automation
- [22] Serial Programming http://en.wikibooks.org/wiki/Serial_Programming/Serial_Linux
- [23] MySQL Reference <http://dev.mysql.com/doc/refman/5.6/en/index.html>
- [24] Use cases purpose http://en.wikipedia.org/wiki/Use_case