Connected Housing Solutions  
“A low-income Housing Automation Service”

Ben Seiber

Daniel Wilmot

Jeff Wallace

Jheryl Lezama

Thomas Pionk

## 

## Table of Contents

|  |  |
| --- | --- |
| **Housing Automation (Cover Page)** | Page # |
| **Table of Contents** | 2 |
| **Section 1.0 – Project Vision** | 5 |
| 1.1 - Project Background | 5 |
| 1.2 - Socio-Economical Impact, Business Objectives, and Gap Analysis | 5 |
| 1.3 - Security and ethical concerns | 7 |
| 1.4 - Glossary of Key Terms | 8-9 |
| **Section 2.0 – Project Planning & Execution** | 10 |
| 2.1 - Team Information | 10 |
| 2.2 - Tools and Technology | 10-11 |
| 2.3 - Project Plan | 12 |
| 2.4 - Best Standards and Practices | 12 |
| **Section 3.0 – System Requirements Analysis** | 13 |
| 3.1 - Functional Requirements | 13-16 |
| 3.2 - Non-Functional Requirements | 16 |
| 3.3 - On-Screen Appearance of landing and other pages requirements | 16-17 |
| 3.4 - Wireframe Designs | 17-19 |
| **Section 4.0 – System Requirements Specifications and Diagrams** | 19 |
| 4.1 - Stakeholders | 19-20 |
| 4.2 - Actors and Goals | 20 |
| 4.3 - User Stories, Scenarios and Use Cases | 20-31 |
| 4.4 - System Sequence / Activity Diagrams | 31 |
| **Section 5.0 – User Interface Specifications** | 32 |
| 5.1 - Preliminary Design | 32-33 |
| 5.2 - User Effort Estimation | 33-35 |
| **Section 6.0 – Static Design** | 35 |
| 6.1 – Class Model | 35 |
| 6.2 – System Operation Contracts | 36-38 |
| 6.3 - Mathematical Model | 38-39 |
| 6.4 - Entity Relation | 39 |
| **Section 7.0 – Dynamic Design** | 40 |
| 7.1 - Sequence Diagrams | 40 |
| 7.2 - Interface Specification | 40 |
| 7.3 - State Diagrams | 40 |
| **Section 8.0 – System Architecture and System Design** | 40 |
| 8.1 – Subsystems / Component / Design Pattern Identification | 40-43 |
| 8.2 – Mapping Subsystems to Hardware (Deployment Diagram) | 43-44 |
| 8.3 - Persistent Data Storage | 44 |
| 8.4 - Network Protocol | 44-47 |
| 8.5 - Global Control Flow | 47 |
| 8.6 - Hardware Requirement | 47-55 |
| **Section 9.0 – Algorithms and Data Structures** | 55 |
| 9.1 - Algorithms | 55-56 |
| 9.2 - Data Structures | 56-58 |
| **Section 10.0 – User Interface Design and Implementation** | 58 |
| 10.1 - User Interface Design | 58-59 |
| 10.2 - User Interface Implementation | 59-60 |
| **Section 11.0 – Testing** | 60 |
| 11.1 - Unit Test Architecture and Strategy/Framework | 60 |
| 11.2 - Unit test definition, test data selection | 60-62 |
| 11.3 - System Test Specification | 62-63 |
| 11.4 - Test Reports per Spring | 63-68 |
| **Section 12.0 – Project Management** | 68 |
| 12.1 - Project Plan | 68 |
| 12.2 - Risk Assessment | 68 |
| **Section 13.0 - References** | 69 |

## 

## Section 1.0 – Project Vision

**1.1 – Project Background**

Home automation or “the internet of things” is not a new concept. However, the benefit of an effective automation and monitoring system is something that cannot be understated. The Connected Housing Solutions system strives to combine the benefits of housing security, utility monitoring, and cost-saving analytics into one beneficial package. It is our belief that such a system would aid in alerting landlords and tenants to possible issues before a potentially larger incident occurs. This system provides the benefits of receiving live sensor updates regarding water pressure, temperature, humidity, and lighting through the convenience of a website. Additionally, landlords are able to control specific items such a lighting and temperature in hallways and public spaces when the need arises. Furthermore, sensor logs are recorded so that analytics can be later generated in hopes of identifying high-cost utilities. As a result, the Connected Housing Solutions system will allow for the automation of a housing facility by using both monitoring and control elements, focusing on building safety and energy savings.

**1.2 – Socio-Economical Impact**

**Housing & Health**

The automation of housing utilities not only increases the convenience of the user, but is also a precautionary step taken towards improving the quality of life for both residents and landlords. The automated monitoring of housing elements such as the air quality control of a location can assist in detecting potential hazards and avoid negative impacts on both child and adult health. As a result, these systems can be implemented to increase both awareness and detection therefore leading to an overall improved quality of life.

**Energy & Efficiency Improvements**

Energy waste leads to an increased long-term operating cost for many low-income housing facilities and can be factored into rising housing costs. These costs leave low-income families with a minimal remaining budget for other necessary expenses and limit their chances of accumulating personal savings. The automation and ability to control high-cost factors such a property’s heating can cut the costs on unnecessary or wasteful energy expenses.

**Housing & Security**

Safety and peace of mind are two factors that can drastically reduce stress within the personal lives of many individuals. Through the automation of security cameras and motion detectors, one can lock their doors and monitor activity with the click of a button. This greatly reduces the vulnerability to possible break-ins and other security concerns for both residences and responsible landlords.

**1.2 – Business Objectives**

**Core Values**

* The Connected Housing Solutions System strives to provide landlords with the tools to reduce long-term utility expenses, increase the level of housing security and identify potential hazards before a possible larger incident occurs.

**Growth**

* The implementation of the Connected Housing Solutions system could be expanded to reach a wider market in addition to low-income housing. The cost-saving functionality could prove beneficial to manufacturing plants, larger corporations and even public stadiums and arenas in hopes of providing a convenient method of monitoring and cutting long-term energy expenses.

**Productivity**

* Provide landlords with a convenient method of remotely monitoring numerous property utilities.
* Provide landlords with the tools to remotely control their various property utilities.
* Provide landlords with a means of sending alert notifications to the respective property tenants should there be any utility failures or health hazards.
* Provide landlords with generated cost-saving analytics taken from the various sensor readings.
* Provide landlords with the ability to access a log of previously recorded, in addition to live sensor information.

**1.2 – Gap Analysis**

* **Current State (1/24/18):** In terms of productivity, there does not exist any means for a landlord to register properties in order to increase the user convenience as outlined by the project goal. Therefore, we should include a property management system before proceeding further that allows a landlord to actively register properties under their own unique account. This implementation will allow for the landlord to not only keep track of their list of properties, but also visually locate the sensors that are tied to each location.
* **Future State (2/7/18):** The property management system has been implemented and any further refining to this feature will be done as development of the overall Connected Housing Solutions system continues. An important aspect to keep in mind is that the system is being designed from the perspective of the landlord/building supervisor and any features that assist with monitoring productivity or the system configuration should be heavily considered to meet productivity goals.
* **Current State (3/10/18):** The website application requires a way for the landlord or property manager to view any alerts directly on the property management dashboard without having to navigate to an alerts page. Currently, there exist three separate options for each registered property. Option one is “manage”, intends to provide the monitoring functionality as outlined in UC-#06 “Remote Monitoring” in addition to viewing alerts. Option two “edit”, intends to provide the landlord the options to both “edit” and “delete” property information as outlined in use cases UC-#04 and UC-#05 respectively. The final option “Analytics”, intends to navigate to the cost-savings analytics page. However, this current U.I. design needs to be overhauled to accommodate an easier method of viewing alert notifications without clicking through these various pages.
* **Future State (3/25/18):** The property management dashboard has been redesigned and a notifications center “at a glance” feature has been implemented. This allows quick access to view any alerts for a given property. This was to address the concerns of receiving alerts as soon as possible as to avoid both tenant injuries and potential property damage. The notifications center is located under the property image on the property management dashboard and alerts are immediately highlighted in red as to draw attention. A designated “Good” status will not be highlighted and will simply display the current status of the system. Please note that the time to implement this U.I. change into the website did set back development time and therefore, all further U.I. concerns should be raised before development on a new feature begins as to avoid future setbacks.

**1.3 – Security Concerns & Ethical Concerns**

**Hacking & Stealing Sensitive Information**

* Given that the system design will include the use of a MySQL database, hacking techniques such as MySQL injection and exploiting the default security parameters associated with the MySQL functionality will be addressed. The first precaution will be making the database accessible on a secured private network to limit the number of potential attackers. The second precaution is to research and implement techniques to reduce to the overall risk of SQL data injection or the unauthorized manipulation of using a register function to drop database tables.

**Tenant Privacy**

* While the system shall allow for the controlling of various building utilities such as lighting and temperature, this functionality is strictly limited to public spaces within the property. These spaces will include - but are not limited to - the building’s front doorway, main hallways, front reception area, public laundry rooms, basement, attic and other maintenance or social spaces as to avoid invasion of individual tenant privacy. However, the monitoring of the various property utilities will include the monitoring of individual tenant’s utilities for the purpose of generating more efficient cost-saving analytics based on reducing long-term utility costs.

**Unauthorized Access**

* The system will require landlords to register an account with our database in order to create a verification process that will be used to access the Connected Housing Solutions System. To improve the security of this verification process, the system will also hash the password entries during both the account registration and login procedure. In addition, the use of the Connected Housing Solutions System shall be limited to the end-users that register an account and securely login to the system.

**1.4 – Glossary of Terms**

* **Android** - Refers to the operating system found in the end-user mobile application for use with the Connected Housing Solutions System.
* **Android Studio -** Refers to the software framework that was used to develop the end-user mobile application.
* **Application Program Interface (API) –** Refers to a set of protocols and tools for building software applications and allowing two applications to communicate with one another.
* **Arduino -** Refers to the open-source microcontroller that houses the logic for the various system sensors including lighting, temperature sensor and security.
* **Asynchronous JavaScript and XML (AJAX) -** Refers to the scripting language that was used to dynamically generate HTML elements within the end-user web application.
* **Cascading Style Sheets (CSS) -** Refers to the scripting language that was used to visually style the end-user web application.
* **End-User -** The intended users of the Connected Housing Solutions system, which are referred to in this document as 1) Landlord, alternatively building supervisor, and 2) Tenant.
* **End-User Application -** Refers to both the web and mobile applications that function as the user interface for the Home Automated System.
* **Gantt Chart -** A visual timeline chart found in Microsoft Project 2016 © that outlines the project timeframe and various deadlines that was used in the team planning and development of the Connected Housing Solutions System project.
* **Hypertext Markup Language (HTML) -** Refers to the scripting language that was used to develop the end-user web application.
* **Hypertext Preprocessor (PHP) -** Refers to the scripting language that was used to handle the information passed between the end-user web application and the various web servers.
* **Landlord -** Refers to the building manager/supervisor who also functions as an end-user for the Connected Housing Solutions system.
* **Microsoft Project 2016 © -** The software framework developed by Microsoft © that was used in the team planning and creation of a project schedule for development of the Connected Housing Solutions System project.
* **MQTT Server -** Refers to the network server that provides communication between the various system sensors (temperature, lighting, etc.), the system database and the end-user software applications.
* **MySQL Server -** Refers the database that will store 1) end-user account information and 2) sensor readings.
* **NodeRed -** Refers to the software framework that supports the communication between the following; MQTT server, MySQL server, Apache Web server, Arduino microcontrollers in addition to the website & mobile applications.
* **Operating System (OS)** - Refers to the software that supports a computer’s or mobile device’s basic functionality including any additional peripheral devices.
* **Property -** Refers to the building or residency that is owned by the landlord.
* **Python -** Refers to the programming language that utilized by the Arduino microcontrollers.
* **Raspberry Pi -** Refers to the miniature computer device that will function to host the MQTT server, MySQL server, NodeRed server and the Apache Web server.
* **Sensor/Asset -** Refers to the individual system sensors including - but not limited to - temperature, water pressure, motion detectors, etc. and property lighting.
* **Tenant -** Refers to the residents that reside at a building/property.
* **User Interface (U.I.) -** The interface that the end-user will interact with in order to use the Connected Housing Solutions System and encompasses both the visual style and layout of the application.
* **Wamp Web Server -** Refers to the server that will host the Connected Housing Solutions System end-user web application.

Section 2.0 – Project Execution & Planning

**2.1 – Team Information**

* **Ben Seiber**
  + Project Lead & Designer of the Connected Housing Solution’s physical model including the CNC laser cutting of the model itself, the electrical wiring of the model, the coding of the Arduino logic and designer of the physical hardware layout.
* **Daniel Wilmot** 
  + Lead programmer of the web application including the HTML, CSS, PHP and AJAX code and co-designer of the web application U.I. design. In addition, responsible for implementing the database schema using PHPmyAdmin and troubleshooting & testing website code.
* **Jeff Wallace**
  + Lead designer of the Connected Housing Solution’s user interface and database schema. Tasks included designing the application page layout & orientation and wireframe schematics. Also responsible for assisting with the coding of both the MySQL queries and the website application.
* **Jheryl Lezama**
  + Project Planner and responsible for documenting the development of the Connected Housing Solutions System. Created the use cases, user stories, functional and nonfunctional requirements, Activity & Sequence Diagrams and the Risk Management Matrix. Also assisted with the coding of the website application.
* **Thomas Pionk**
  + Lead designer of the back-end system network including the server configurations and various PowerShell scripts that allowed for communication between the various hardware elements. Also designed the backend system U.I. dashboard used for controlling the various system sensors and configuring the NodeRed networking schema.

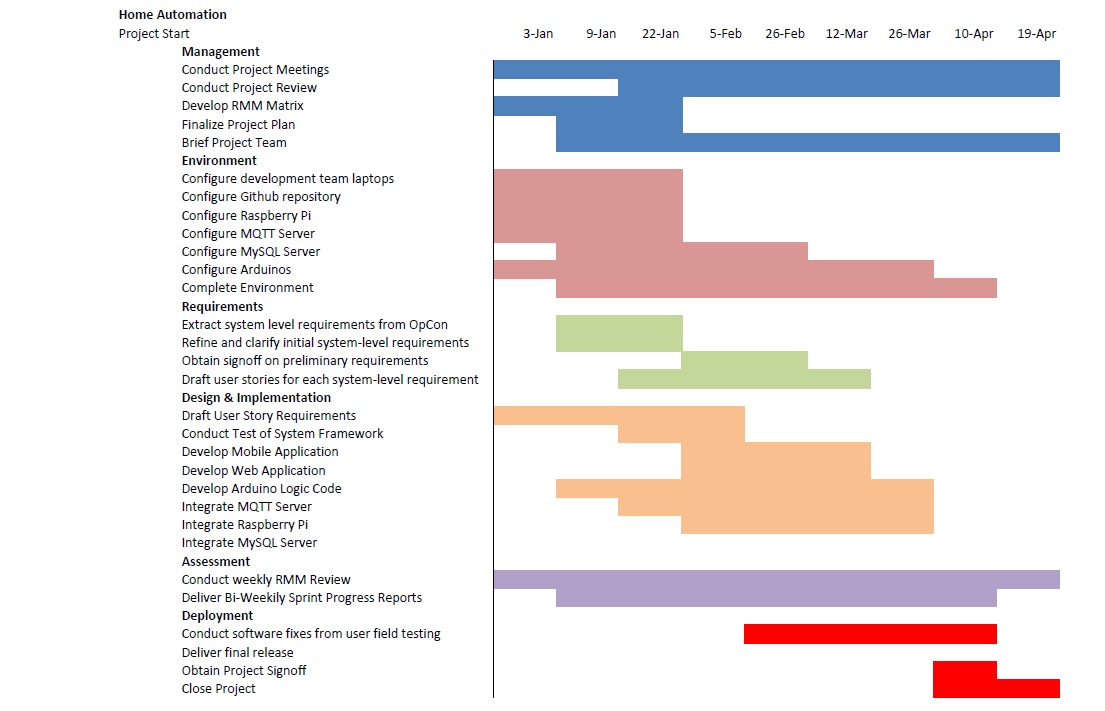
**2.2 – Tools and Technology**

The tools and technology for this project was broken down into two categories consisting of both hardware and software. Therefore, each of these frameworks has been represented within the respective table below.

|  |  |  |
| --- | --- | --- |
| **Hardware** | | |
| **Type of Hardware** | **Purpose of Hardware** | **Device** |
| Individual | Development | Dell Latitude © |
| Individual | Development | Apple Macbook Pro © |
| Individual | Development | CNC Laser Cutter |
| Individual | Development | Mobile Phone (Android ©) |
| On-Premise | Deployment & Testing | Arduino © |
| On-Premise | Deployment & Testing | Raspberry Pi © |
| On-Premise | Deployment & Testing | Air Purity Sensor |
| On-Premise | Deployment & Testing | LED Lighting |
| On-Premise | Deployment & Testing | Humidity Sensor |
| On-Premise | Deployment & Testing | Motion Sensor |
| On-Premise | Deployment & Testing | Temperature Sensor |
| On-Premise | Deployment & Testing | Wireless Router |

|  |  |  |
| --- | --- | --- |
| **Software** | | |
| **Type of Framework** | **Purpose of Framework** | **Software Program** |
| Individual | Team Collaboration | GroupMe © |
| Individual | Development | Android Studio © |
| Individual | Development | Arduino C-Scripting |
| Individual | Development | Sublime Text Editor © |
| Individual | Development | Python © |
| Individual | Development | Text Editor |
| Individual | Development | Command Line, PowerShell © |
| Individual | Project Management | Microsoft Project 2016 © |
| Individual | Project Management | Microsoft Excel 2016 © |
| Open API | Development | NodeRed © |
| Open API | Development | Wamp Server © |
| On-Premise | Development | Fusion 360 © |
| On-Premise | Deployment & Testing | NodeRed © |
| On-Premise | Deployment & Testing | MySQL Server © |
| On-Premise | Deployment & Testing | MySQL Workbench © |
| On-Premise | Deployment & Testing | PHPmyAdmin © |
| On-Premise | Deployment & Testing | MQTT Server |
| On-Premise | Deployment & Testing | ChartJS © |
| Cloud-Based | Team Collaboration | Github © |
| Cloud-Based | Team Collaboration | Google Drive © |
| Cloud-Based | Team Collaboration | Google Hangouts © |

**2.3 – Project Plan**



**2.4 – Best Standards and Practices**

**Better Targeting**

* Alerts and notifications are less likely to be intercepted by an unintended party if the system is hosted on a private network.

**Lower Costs**

* Generating cost-saving analytics that can be used to reduce long-term utility expenses and energy expense.

**Reliability**

* The website application for managing the system will be made available 99.98% of the time.
* The end-user account information will be stored within the database with no data loss.

**System Security**

* The website will hash the password entry during both the registration and login process to increase system security.
* The MySQL database will store passwords as hashes in addition to protecting itself from potential SQL injections.
* The MQTT server will only be accessible on the private network.

## Section 3.0 – System Requirements Analysis

**3.1 – Functional Requirements**

|  |  |
| --- | --- |
| General Requirement ID: 01 | Priority: Low |
| Requirement: Account Registration | |
| Actor(s): Landlord | Assigned to: Danny Wilmot, Jheryl Lezama, Jeff Wallace |
| Description:   1. The system shall provide landlords the ability to register a unique account using either the website or mobile application. 2. The system will require the following information when registering an account:  1) Email 2) Password 3. Further Notes: 1) The system will hash the password entry in order to increase system security.  2) The system will save a landlord’s account information within the database with no data loss.  3) The landlord’s account will unique from other accounts stored within the MySQL database. | |

|  |  |
| --- | --- |
| General Requirement ID: 02 | Priority: Low |
| Requirement: Account Login | |
| Actor(s): Landlord | Assigned to: Danny Wilmot, Jheryl Lezama, Jeff Wallace |
| Description:   1. The system shall provide landlords the ability to login to their own unique account via the website or mobile application. 2. The system will require the following information when logging into an account:  1) Email 2) Password 3. Further Notes: 1) The system will hash the password entry in order to increase system security.  2) The system will save a landlord’s account information within the database with no data loss.  3) The system will only access relevant account information that is associated with that particular account. | |

|  |  |
| --- | --- |
| General Requirement ID: 03 | Priority: High |
| Requirement: Remotely Monitoring the Sensors/Assets | |
| Actor(s): Landlord | Assigned to: Ben Seiber, Thomas Pionk |
| Description:   1. The system shall provide landlords the ability to remotely monitor the information being recorded by sensors/assets including lighting, temperature, humidity, etc. via the website or mobile application. 2. Further Notes: 1) The ability to remotely monitor the various system sensors shall only be granted to users who have both registered and securely logged into the system via the website or mobile application.  2) The monitor feature will not be limited only to public spaces and will extend to individual tenant’s spaces for the purpose of generating more accurate cost-saving analytics.  3) The monitoring of the sensor/asset information will be unique to each registered property associated with that particular landlord’s account. | |

|  |  |
| --- | --- |
| General Requirement ID: 04 | Priority: High |
| Requirement: Remotely Controlling the Sensors/Assets | |
| Actor(s): Landlord | Assigned to: Ben Seiber, Thomas Pionk |
| Description:   1. The system shall provide landlords the ability to remotely control the various sensors/assets including lighting, temperature, humidity, etc. via the website or mobile application. 2. Further Notes: 1) The ability to remotely monitor the various system sensors shall only be granted to users who have both registered and securely logged into the system via the website or mobile application.  2) The control feature will be limited only to public spaces and will not extend to individual tenant’s spaces as to avoid breach of privacy.  3) The controlling of the sensor/asset information will be unique to each registered property associated with that particular landlord’s account. | |

|  |  |
| --- | --- |
| General Requirement ID: 05 | Priority: High |
| Requirement: Accessing Sensor/Asset Logging Information | |
| Actor(s): Landlord | Assigned to: Ben Seiber, Daniel Wilmot, Jheryl Lezama, Jeff Wallace, Thomas Pionk |
| Description:   1. The system shall provide landlords the ability to view the logging of sensor/asset information including lighting, temperature, humidity, etc. via the website or mobile application. 2. Further Notes: 1) The system will use a MySQL database to store the sensor/asset readings.  2) The sensor/asset information will be unique to each registered property associated with that particular landlord’s account.  3) The website or mobile application will provide the means to interface with the MySQL database in order to view the stored sensor readings. | |

|  |  |
| --- | --- |
| General Requirement ID: 06 | Priority: High |
| Requirement: Generating Cost-Saving Analytics | |
| Actor(s): Landlord | Assigned to: Ben Seiber, Daniel Wilmot, Jheryl Lezama, Jeff Wallace, Thomas Pionk |
| Description:   1. The system shall provide landlords with cost-saving analytics based on the logged sensor/asset information that could then be used to reduce long-term utility expense. 2. Further Notes: 1) The system will use a MySQL database to store the sensor/asset readings.  2) The cost-saving analytics will be unique to each registered property associated with that particular landlord’s account.  3) The website or mobile application will provide the means to interface with the MySQL database in order to generate the cost-saving analytics. | |

|  |  |
| --- | --- |
| General Requirement ID: 07 | Priority: Medium |
| Requirement: Registering Properties | |
| Actor(s): Landlord | Assigned to: Daniel Wilmot, Jheryl Lezama, Jeff Wallace |
| Description:   1. The system shall provide landlords with the ability to register a list of multiple properties under their own account. This will provide landlords with a means to keep track of their various properties in addition to the sensors that occupy each one. 2. Further Notes: 1) The property information will be stored in a MySQL database.  2) Each newly registered property will be unique from each other (no duplicates).  3) Each registered property will be unique to the account to which it was registered to and will not be accessible by another account. | |

**3.2 - Non-Functional Requirements**

**MySQL Database**

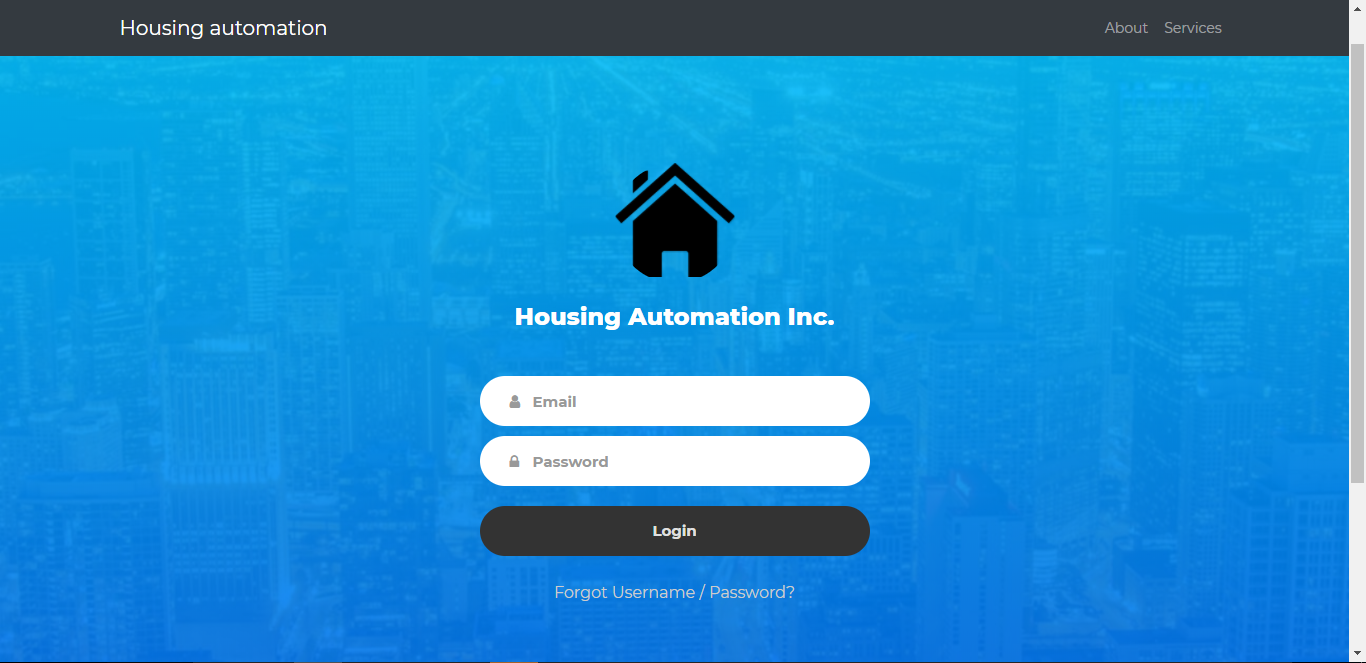
* The MySQL database will store landlord account information with no data loss.
* The MySQL database will be able to store at least 100 sensor/asset information snapshots within a log file.
* The MySQL database will be implemented with an industry standard relational database schema.

**Server Availability**

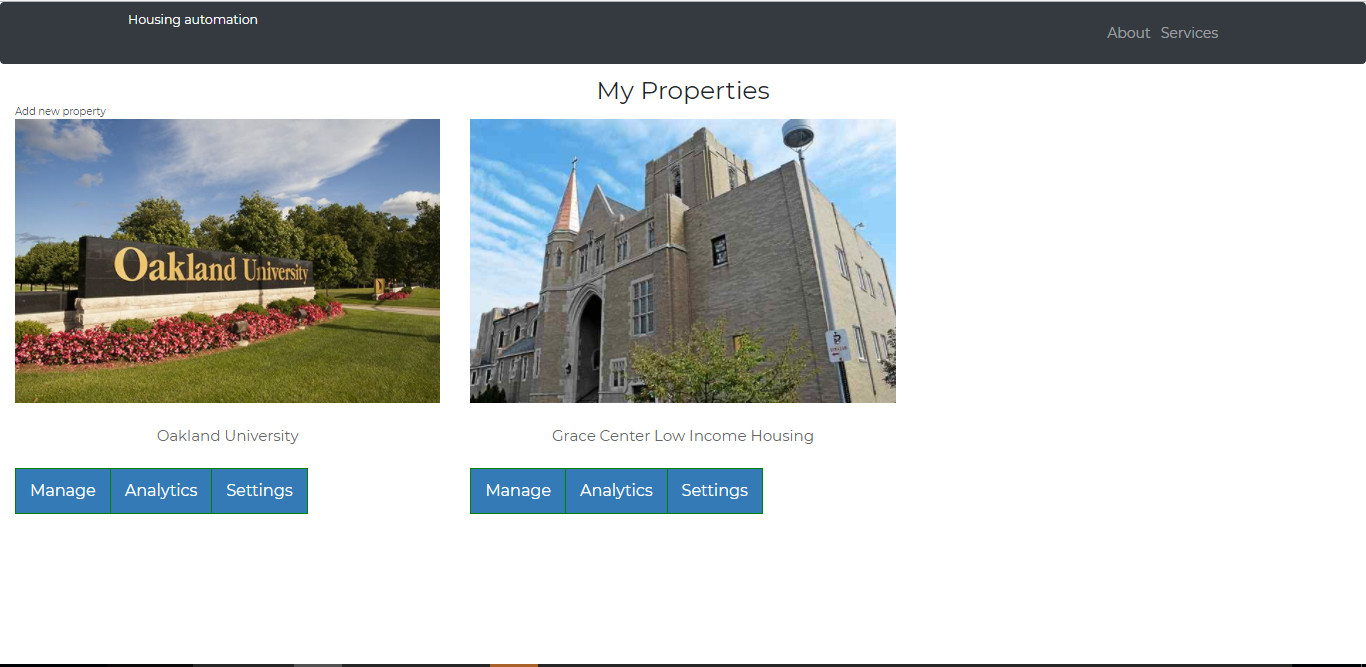
* The MQTT Server shall provide the back-end network functionality and be available 99.98% of the time.
* The MQTT will be hosted on a private network only accessible to those who have securely registered and logged into the Connected Housing Solutions system.

**Website Requirement**

* The website shall be accessible on both Windows and Mac OS devices.
* The website shall be able to load on both computer and mobile phone devices.
* The website application pages shall be compatible with HTML 4.0 or higher.
* As a minimum, the website application pages shall be compatible Firefox, Chrome, Safari and Internet Explorer.

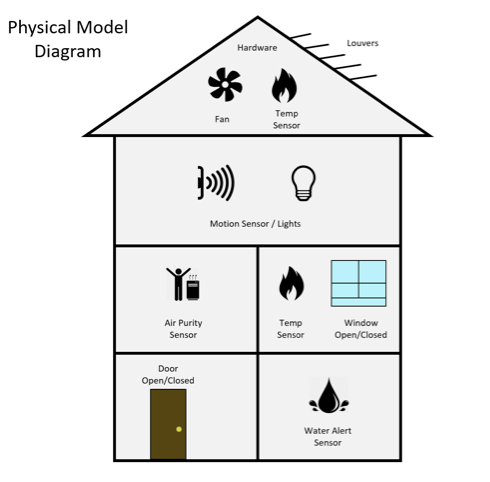
**3.3 – On-Screen Appearance of landing and other pages requirements  
  
1) Login:** Includes an account registration form as outlined in use case (UC-#01) and a forgot password recovery option.

**2) Dashboard:** Includes a property registration form as outlined in use case (UC-#03), includes the ability to upload photos for each property using a generated c-drive folder tied to each landlord account. Note: The properties being displayed are tied to each unique account.

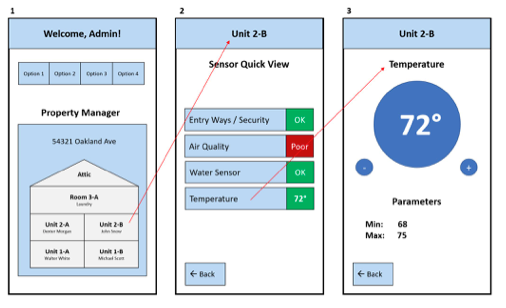


**3.4 – Wireframe Designs**

* The following was an early concept developed during Sprint 2 for a physical model to demonstrate the following features.

****

* The following was an early concept for a mobile application that was later dropped in favor of focusing on improving the website application’s functionality and responsiveness. The concept was later integrated into the property management dashboard.

****

Section 4.0 – Functional Requirements Specification

**4.1 – Stakeholders**

|  |  |
| --- | --- |
| **Customers** | 1) Grace Centers of Hope 2) Landlords and Property Managers 3) Dr. Nilesh Patel |
| **Project Manager** | Jheryl Lezama |
| **Project Team Members** | 1) Ben Seiber 2) Daniel Wilmot 3) Jeff Wallace 4) Jheryl Lezama 5) Thomas Pionk |
| **Project Sponsor** | Grace Centers of Hope |
| **Steering Committee** | 1) Grace Centers of Hope 2) Dr. Nilesh Patel |
| **Project Executives** | 1) Grace Centers of Hope 2) Dr. Nilesh Patel 3) Oakland University |
| **Resource Manager** | Oakland University |

**4.2 – Actors and Goals**

|  |  |
| --- | --- |
| **Actor(s)** | **Goal(s)** |
| **Landlord** | 1. Register an account for use with the system 2. Register properties for use with the system 3. Ability to remotely control sensors 4. Ability to remotely monitor sensors |
| **MQTT Server** | 1. Allow for communication between the website application and MySQL database 2. Allow for communication between the website application and various housing sensors 3. Allow for logging of sensor data |
| **MySQL** | 1. Stores landlord account information 2. Stores sensor readings/data |
| **Node Red** | 1. Allows for easier integration between hardware and software 2. Allows for easier configuration of the back-end network support |

**4.3 – User Stories**

|  |  |  |
| --- | --- | --- |
| **Identifier** | **User Story** | **Size** |
| ST-1 | As a landlord, I want a secure registration process that safely stores my personal information within a database. | 3pts |
| ST-2 | As a landlord, I want a secure login process that grants me access to my own unique account separate from others. | 3pts |
| ST-3 | As a landlord, I want a way to view sensor/asset information as gathered from my various properties. | 9pts |
| ST-4 | As a landlord, I want a way to control the various utilities within the public spaces of each of my property. | 9pts |
| ST-5 | As a landlord, I want to see generated cost-saving analytics that could help me save money in regard to long-term utility expenses. | 6pts |

**4.3 – Scenarios**

**User Scenario – Becoming a new user**

Upon deciding that a customer (being a housing landlord) would like to use our service, they would log on to our website (www.HomeAutomation.com) and register as a new user. The website will prompt them for a username and password, their first and last name, as well as their email address. After such information has been entered, the customer will click the register button, and given there is no conflict with the information (i.e. the username already exists, or the password format is unacceptable), the user will be registered.

**User Scenario – Logging in and viewing sensor data/statuses of a property**

Upon wanting to login to their user account and view the status of their housing complex, the landlord would visit our Connect Housing Solutions website and click the login button (given they have already completed the user/property registration process). The landlord will be prompted for their username and password; upon entering the correct information, they will be logged in.

Once logged in, the landlord will click on the manage button associated with the property of their choosing. On the manage page, the customer will be shown the data related to that property. More to come on this paragraph.

**User Scenario – Generate cost-saving analytics**

Upon wanting to view Home Automation analytics related to an owned property, the landlord would login to their user account, click on the property they wish to view metrics on, and then click on the “View Analytics” button.

**4.3 – Use Cases**

|  |  |  |
| --- | --- | --- |
| **UC-#01** | | **Account Registration** |
| Related Requirements: | | None |
| Initiating Actor: | | Landlord |
| Actor’s Goal: | | Register an account within the database for use with the Connected Housing Solutions system. |
| Participating Actors: | | MySQL Database, Website Application, Mobile Application. |
| Trigger Action: | | User requests to register an account. |
| Pre-Conditions: | | System does not already contain any prior account information regarding the landlord. |
| Post-Conditions: | | System stores the landlord account information within the MySQL database. |
| **Flow of Events for Main Success Scenario:** | | |
| 🡪 | 1. | Landlord selects register new account on the login page. |
| 🡨 | 2. | The system responds by displaying an account registration form for the landlord to enter personal account information. |
| 🡪 | 3. | The landlord enters personal account information into the account registration form. |
| 🡪 | 4. | The landlord presses the submit button once they have completed filling out the account registration form. |
| 🡨 | 5. | The system responds by first checking that all of the entry fields have information entered. |
| 🡨 | 6. | The system responds by checking the provided information against the information stored within the database to insure that there exists no duplicate account information. |
| 🡨 | 7. | **(Exit Condition)** The system responds with visual message confirmation that an account has been successfully registered within the database. |
| **Flow of Events for Main Success Scenario (Alternate Scenario):** | | |
| 🡨 | 5a. | The system responds with a visual error message that all of the entry fields have not been completed. |
| 🡨 |  | The system does not create an account until all of the form fields contain the necessary information. |
| 🡨 |  | Same as in Step 3 above. |
| 🡨 | 6a. | The system responds with a visual error message that the information entered within the account registration form already exists within the database. |
| 🡨 |  | The system does not create a new account given the existence of duplicate account information within the database. |
| 🡪 |  | Same as in Step 3 above. |
| 🡨 | 6a. | The system responds with a visual error message that there was an error connecting to the database. |
| 🡨 |  | The system does not create a new account within the database. |

|  |  |  |
| --- | --- | --- |
| **UC-#02** | | **Account Login** |
| Related Requirements: | | UC-#01 |
| Initiating Actor: | | Landlord |
| Actor’s Goal: | | Gain access to an account that was previously registered to the database in order to begin using the features of system. |
| Participating Actors: | | MySQL Database, Website Application, Mobile Application. |
| Trigger Action: | | User requests to access their previously registered account in order to gain access to the features of the system. |
| Pre-Conditions: | | System already contains previously registered account information. |
| Post-Conditions: | | Landlord logs into their unique account. |
| **Flow of Events for Main Success Scenario:** | | |
| 🡪 | **1.** | Landlord enters account information on the login page. |
| 🡪 | 2. | Landlord presses the login/submit button after entering the required information. |
| 🡨 | 3. | The system responds by checking the provided login account information against the information stored within the database. |
| 🡨 | 4. | The system logs the landlord into their unique account. |
| 🡨 | 5. | **(Exit Condition)** The system further responds with visual confirmation message that the landlord has successfully logged into their account. |
| **Flow of Events for Main Success Scenario (Alternate Scenario):** | | |
| 🡨 | 3a. | The system responds with a visual error message that the login account information provided was incorrect. |
| 🡪 |  | Same as in Step 1 above |

|  |  |  |
| --- | --- | --- |
| **UC-#03** | | **Register Property** |
| Related Requirements: | | UC-#01, UC-#02 |
| Initiating Actor: | | Landlord |
| Actor’s Goal: | | To register a new property within the database. |
| Participating Actors: | | MySQL Database, Website Application, Mobile Application. |
| Trigger Action: | | The landlord has selected the option to “add new property”. |
| Pre-Conditions: | | The landlord has successfully logged into their unique account and has requested to “add a property”. |
| Post-Conditions: | | The property information has been successfully registered within the database and is now displayed on the property management dashboard. |
| **Flow of Events for Main Success Scenario:** | | |
| 🡪 | 1. | Landlord selects “add a property” feature from the property management dashboard. |
| 🡨 | 2. | The system responds by receiving the request and displays a form for the landlord to enter in property information. |
| 🡪 | 3. | The landlord enters the property information into the form fields. |
| 🡪 | 4. | The landlord presses the “add” button after entering the property information into each of the form fields. |
| 🡨 | 5. | The system responds by first checking that all of the entry fields have information entered. |
| 🡨 | 6. | The system then opens a PHP request to store the property information within the database. |
| 🡨 | 7. | The system then returns the landlord back to the property management dashboard. |
| 🡨 | 8. | **(Exit Condition)** The system then displays the newly registered property information on the property management dashboard. |
| **Flow of Events for Main Success Scenario (Alternate Scenario):** | | |
| 🡨 | 5a. | The system responds with an error message that “all of the required form fields have not been completed”. |
| 🡪 |  | (Same as in step 3) |
| 🡨 | 8a. | The system displays an error message stating that, “no properties have been registered due to a PHP connection error”. |
| 🡪 |  | (Same as in step 1) |

|  |  |  |
| --- | --- | --- |
| **UC-#04** | | **Edit Property** |
| Related Requirements: | | UC-#01, UC-#02, UC-#03 |
| Initiating Actor: | | Landlord |
| Actor’s Goal: | | To edit listed property information stored within the database |
| Participating Actors: | | MySQL Database, Website Application, Mobile Application. |
| Trigger Action: | | Landlord desires to update/change a registered property’s information. |
| Pre-Conditions: | | The landlord already has a registered property and has selected the option to edit property information. |
| Post-Conditions: | | The property information has been successfully updated within the database. |
| **Flow of Events for Main Success Scenario:** | | |
| 🡪 | 1. | The landlord selects the “settings” option on the property task bar for the property they wish to edit. |
| 🡨 | 2. | The system responds by receiving the request and opening the property management page for that specific property. |
| 🡪 | 3. | The landlord selects the “edit” option on the property management page. |
| 🡨 | 4. | The system responds by displaying an edit property information form. |
| 🡪 | 5. | The landlord makes the necessary changes to the respective form fields regarding the information they wish to change. |
| 🡪 | 6. | The landlord presses the “submit” button to confirm the changes they have made to the property information. |
| 🡨 | 7. | The system responds by first checking that all of the entry fields are complete. |
| 🡨 | 8. | The system uses PHP to connect to the database and performs a update to the stored property information. |
| 🡨 | 9. | The system then returns the landlord back to the main page. |
| 🡨 | 10. | **(Exit Condition)** The system then displays the newly edited property information on the main page. |
| **Flow of Events for Main Success Scenario (Alternate Scenario):** | | |
| 🡨 | 7a. | The system responds with an error message that “all of the required form fields have not been completed”. |
| 🡪 |  | (Same as in step 5) |

|  |  |  |
| --- | --- | --- |
| **UC-#05** | | **Delete Property** |
| Related Requirements: | | UC-#01, UC-#02, UC-#03 |
| Initiating Actor: | | Landlord |
| Actor’s Goal: | | To delete property information stored within the database. |
| Participating Actors: | | Web Application, Mobile Application, MySQL Database. |
| Trigger Action: | | The landlord wishes to delete a property listed on their property management dashboard. |
| Pre-Conditions: | | The landlord already has a registered property and has selected the option to delete property information |
| Post-Conditions: | | The property information has been successfully deleted from the database and is no longer displayed on the property management dashboard. |
| **Flow of Events for Main Success Scenario:** | | |
| 🡪 | 1. | The landlord selects the “manage” option on the property task bar for the property they wish to edit. |
| 🡨 | 2. | The system responds by receiving the request and opening the property management page for that specific property. |
| 🡪 | 3. | The landlord selects the “delete” option on the property management page. |
| 🡨 | 4. | The system receives the request and responds with a dialogue box asking if they wish to confirm the deletion of the property. |
| 🡪 | 5. | The landlord confirms the action to delete the property information using the dialogue box. |
| 🡪 | 6. | The system opens a PHP connection to the database to delete the associated property information. |
| 🡨 | 7. | The system then returns the landlord back to the property management dashboard. |
| 🡨 | 8. | **(Exit Condition)** The system then removes the property information from the property management dashboard. |
| **Flow of Events for Main Success Scenario (Alternate Scenario):** | | |
| 🡪 | 5a. | The landlord does not confirm the property deletion from the dialogue box and cancels the decision to delete the property information. |
| 🡨 |  | The system then returns the landlord back to the property management dashboard. |
| 🡨 |  | **(Exit Condition)** The property information is still displayed on the property management dashboard and has not been deleted. |

|  |  |  |  |
| --- | --- | --- | --- |
| **UC-#06** | | | **Remote Monitoring** |
| Related Requirements: | | | UC-#01, UC-#02, UC-#03 |
| Initiating Actor: | | | Landlord |
| Actor’s Goal: | | | To remotely monitor the sensor/asset information within a particular property via the web or mobile application. |
| Participating Actors: | | | Sensor/Asset, MQTT Server, Web Application, Mobile Application. |
| Trigger Action: | | | Landlord requests to view the sensor/asset log information of a particular property. |
| Pre-Conditions: | | | Landlord already has a registered property and has selected the option to view sensor/asset information from property management screen. |
| Post-Conditions: | | | Landlord is taken to a dashboard that displays the sensor/asset information for that specified property. |
| **Flow of Events for Main Success Scenario:** | | | |
| 🡪 | | 1. | Landlord selects a registered property. |
| 🡨 | | 2. | The system responds by visually displaying the desired property information. |
| 🡪 | | 3. | The landlord selects the option to view sensor/asset information for that property. |
| 🡨 | | 4. | The system receives the request to view sensor/asset information and forwards the request to the MQTT Server. |
| 🡨 | | 5. | The MQTT server receives the request and sends the requested information to the application. |
| 🡨 | | 6. | The application receives the sensor/asset information data from the MQTT server. |
| 🡨 | | 7. | **(Exit Condition)** The system visually displays the sensor/asset information dashboard for the specified property to the landlord. |
| **Flow of Events for Main Success Scenario (Alternate Scenario):** | | | |
| 🡨 | 2a. | | The system responds with a visual error message that the property information cannot be retrieved from the database. |
| 🡪 |  | | (Same as in step 1) |
| 🡨 | 4a. | | The system responds with a visual error message that the property sensor/asset information cannot be retrieved. |
| 🡪 |  | | (Same as in step 3) |

|  |  |  |
| --- | --- | --- |
| **UC-#07** | | **Remote Control** |
| Related Requirements: | | UC-#01, UC-#02, UC-#03 |
| Initiating Actor: | | Landlord |
| Actor’s Goal: | | To remotely control a property sensor/asset via the web or mobile application (ex: turn off the lights) |
| Participating Actors: | | Sensor/Asset, MQTT Server, Web Application, Mobile Application |
| Trigger Action: | | The landlord selects the option to make an adjustment to a utility asset based on sensor readings. |
| Pre-Conditions: | | Landlord already has a registered property and has selected the option to control a sensor/asset from property management screen. |
| Post-Conditions: | | Landlord has visually changed the status of a sensor/asset setting. |
| **Flow of Events for Main Success Scenario:** | | |
| 🡪 | 1. | Landlord selects a registered property. |
| 🡨 | 2. | The system responds by visually displaying the desired property information. |
| 🡪 | 3. | The landlord selects the option to control a utility for that property. |
| 🡨 | 4. | System responds by sending the request to the MQTT server. |
| 🡨 | 5. | System further responds by visually displaying a system control dashboard to the landlord. |
| 🡪 | 6. | Landlord inputs the necessary changes to the sensor/asset dashboard. |
| 🡪 | 7. | Landlord submits the change to the sensor/asset information. |
| 🡨 | 8. | The MQTT Server receives the request and processes the change in sensor/asset setting. |
| 🡨 | 9. | System visually displays the change on the sensor/asset dashboard. |
| **Flow of Events for Main Success Scenario (Alternate Scenario):** | | |
| 🡨 | 2a. | The system responds with a visual error message that the property information cannot be retrieved from the database. |
| 🡪 |  | (Same as in step 1) |
| 🡨 | 4a. | The system responds with a visual error message that the request could not be sent to the MQTT Server. |
| 🡪 |  | (Same as in step 3) |

|  |  |  |
| --- | --- | --- |
| **UC-#08** | | **Generating Cost-Saving Analytics** |
| Related Requirements: | | UC-#01, UC-#02, UC-#03, UC-#06 |
| Initiating Actor: | | Landlord |
| Actor’s Goal: | | To view cost-saving analytics for a desired property based on sensor readings/logs. |
| Participating Actors: | | Sensor/Asset, MQTT Server, Web Application, Mobile Application |
| Trigger Action: | | The landlord selects the option to view cost-saving analytics based on sensor readings. |
| Pre-Conditions: | | Landlord already has a registered property and wishes to generate cost-saving analytics based on the sensor readings/logs. |
| Post-Conditions: | | Landlord is viewing various cost-saving analytics for the desired property within the property management dashboard. |
| **Flow of Events for Main Success Scenario:** | | |
| 🡪 | 1. | Landlord selects a registered property. |
| 🡨 | 2. | The system responds by visually displaying the desired property information. |
| 🡪 | 3. | The landlord selects the analytics option for that property. |
| 🡨 | 4. | System responds by sending the request to the MQTT server to view the live sensor data. |
| 🡨 | 5. | The system further responds by also reading the sensor data stored in the MySQL database for each of the various sensors. |
| 🡨 | 6. | The website application receives the sensor data from both sources and plots the information into Chart.JS. |
| 🡨 | 7. | **(Exit Condition)** The Chart.JS graphs are then displayed on the analytics page for the landlord to view an analytics breakdown. |
| **Flow of Events for Main Success Scenario (Alternate Scenario):** | | |
| 🡨 | 4a. | The system responds with a visual error message that the MQTT server is currently unavailable. |
| 🡪 |  | (Same as in step 1) |
| 🡨 | 5a. | The system responds with a visual error message that the MySQL server is currently unavailable. |
| 🡪 |  | (Same as in step 1) |

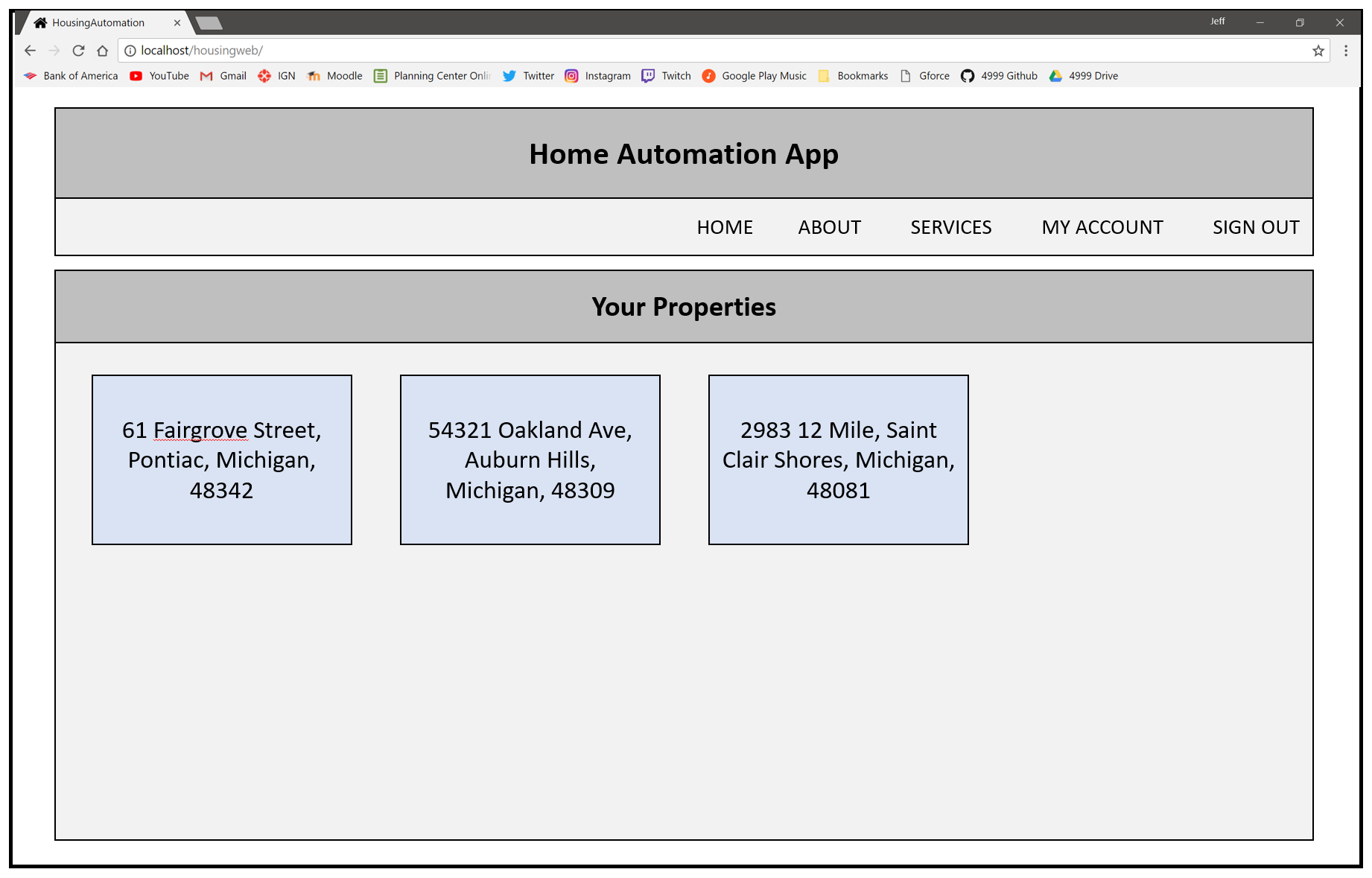
**4.4 – System Sequence / Activity Diagrams**

1. Please see the “ConnectedHousingSolutions.asta” for the Activity and Sequence Diagrams corresponding to each use case.

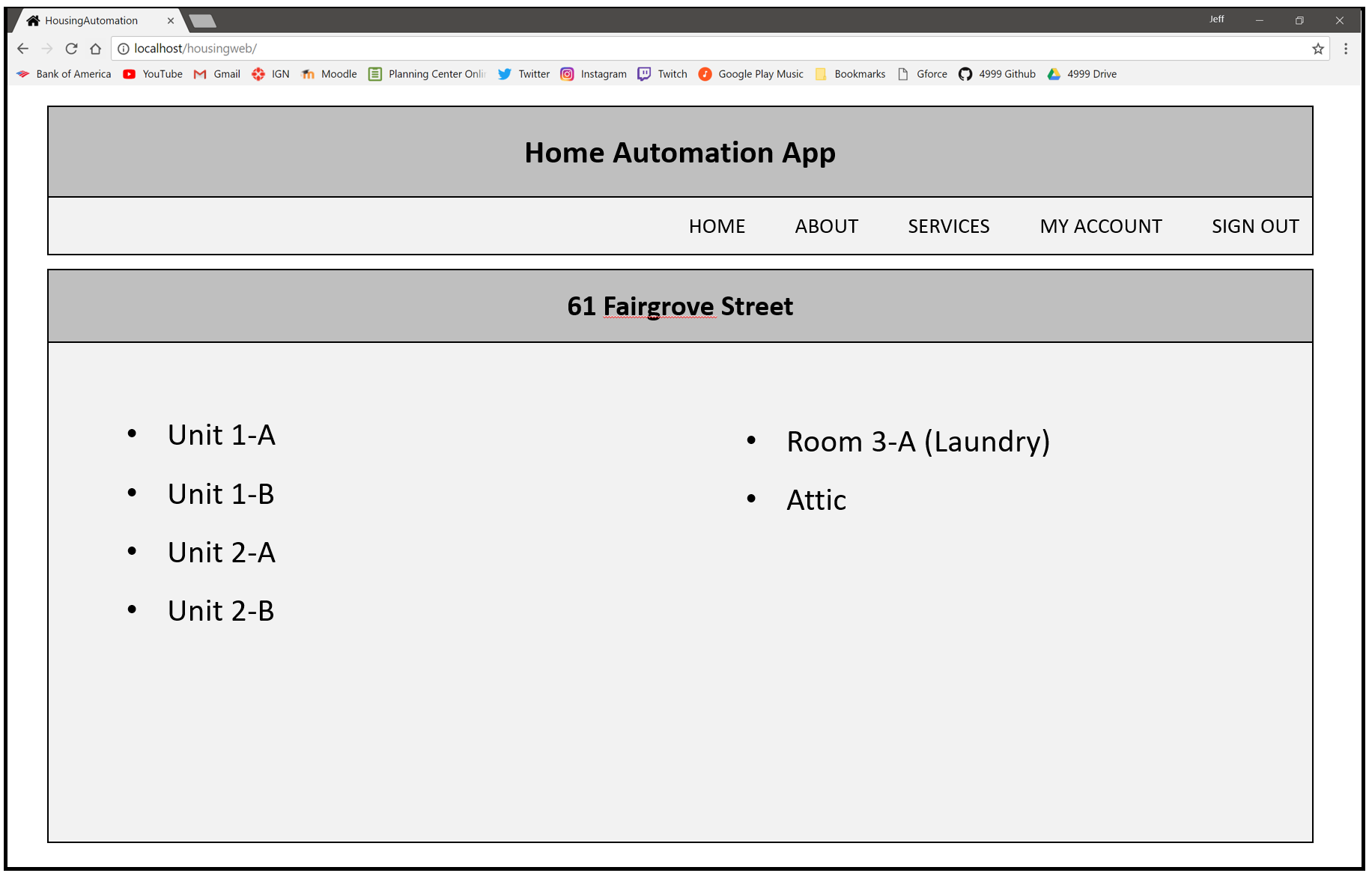
Section 5.0 – User Interface Specifications

* 1. **– Preliminary Design**

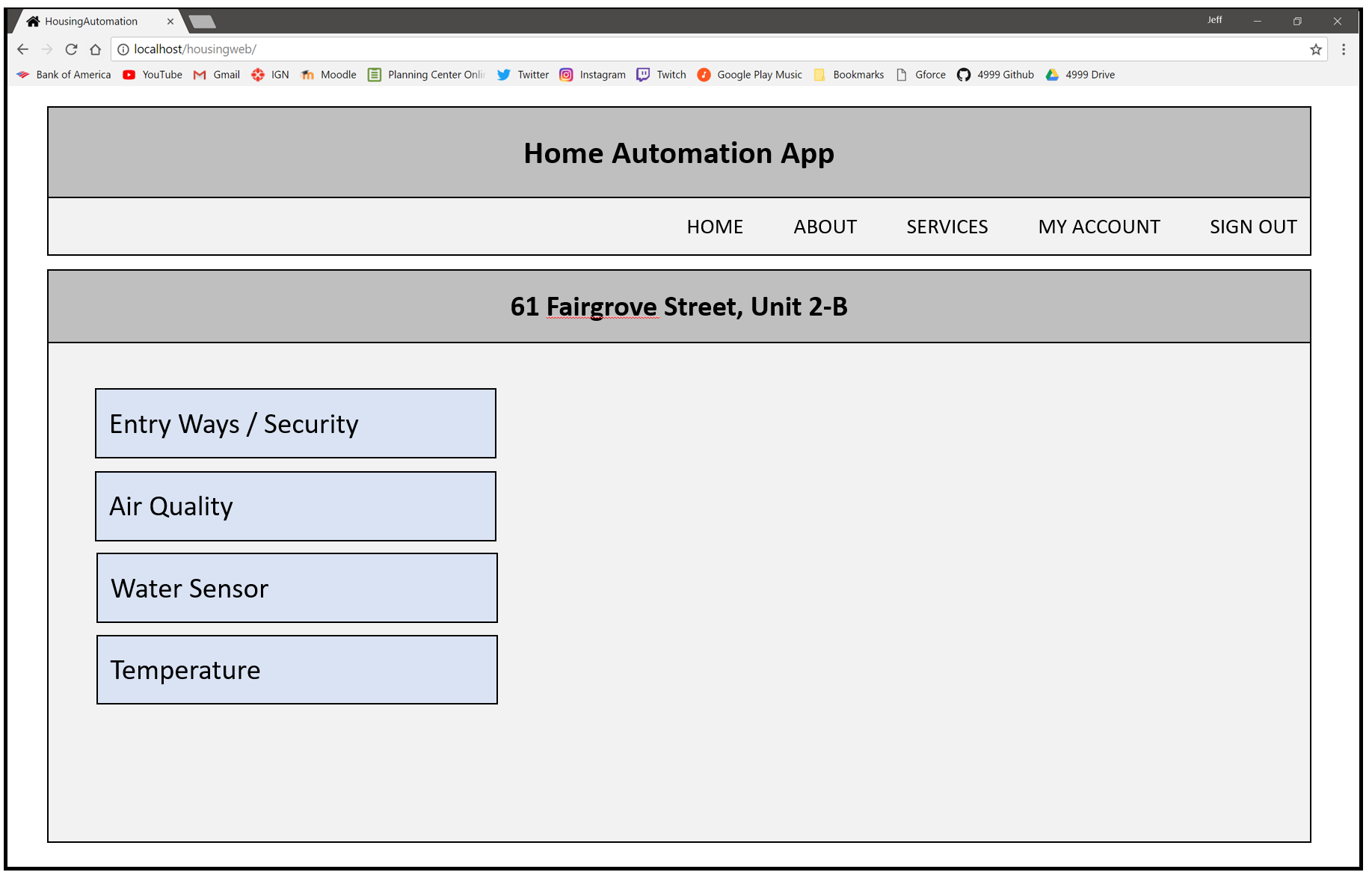
Example Page 1



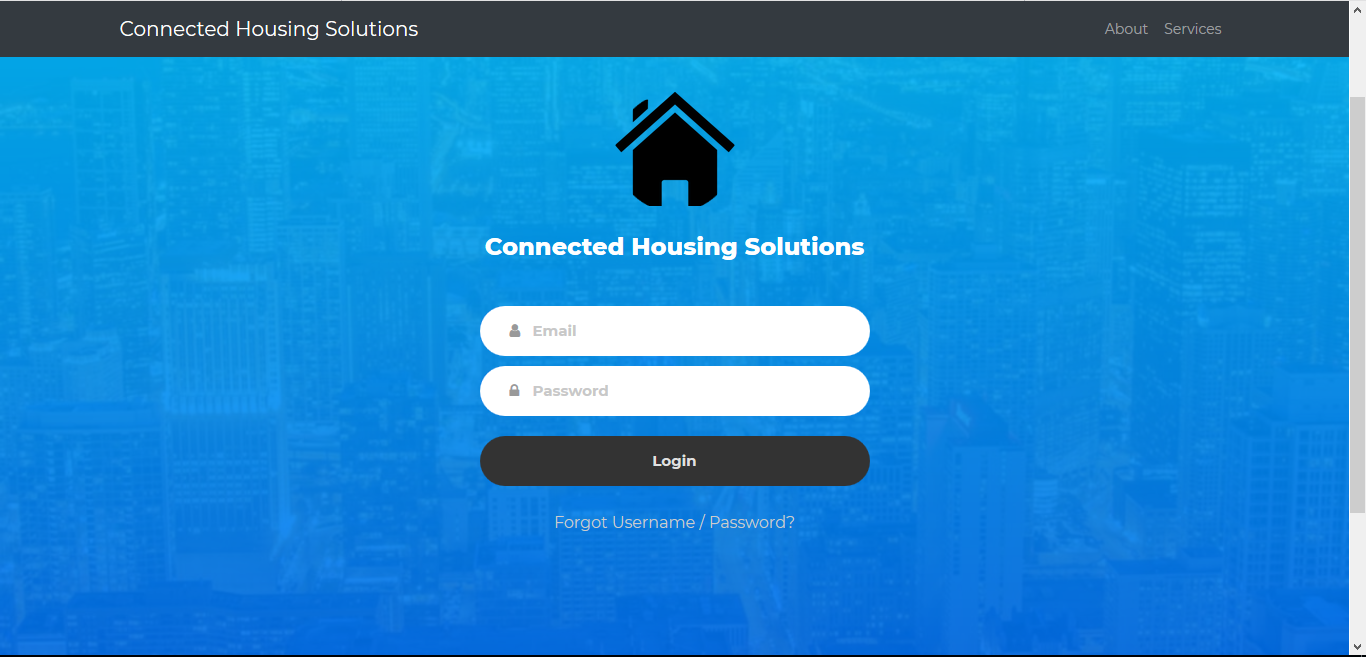
Example Page 2



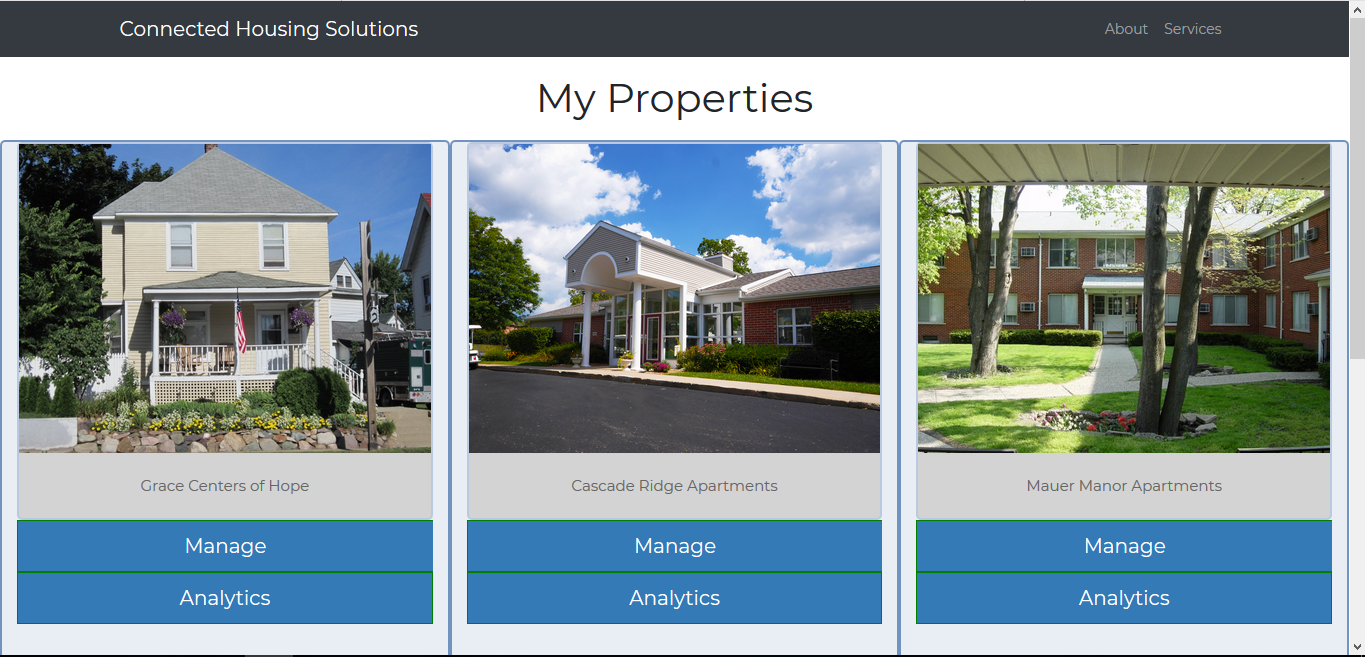
Example Page 3



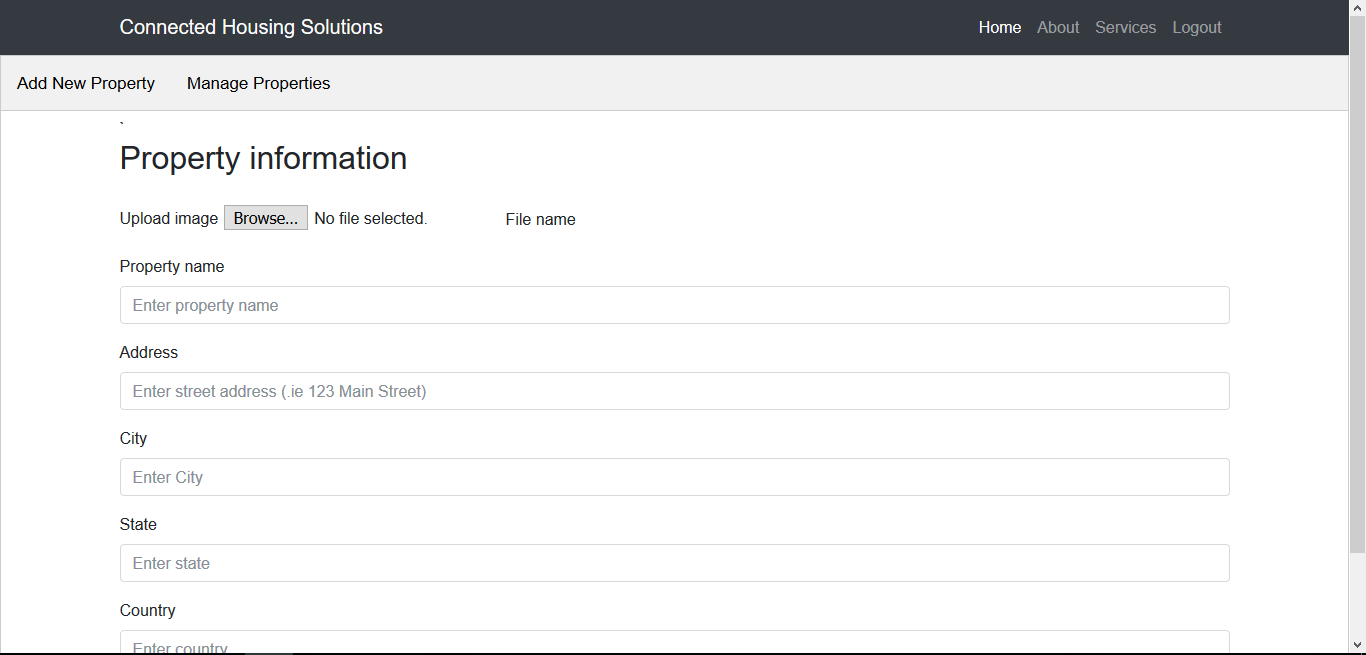
* 1. **– User Effort Estimation**

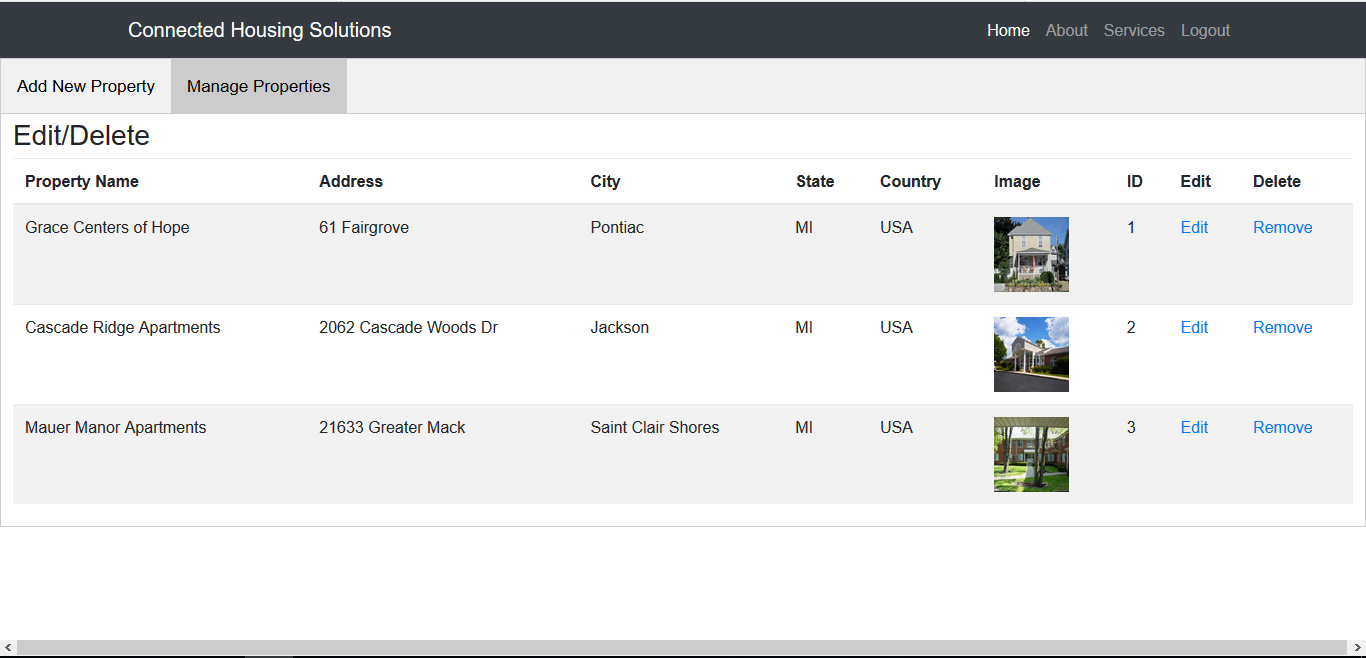
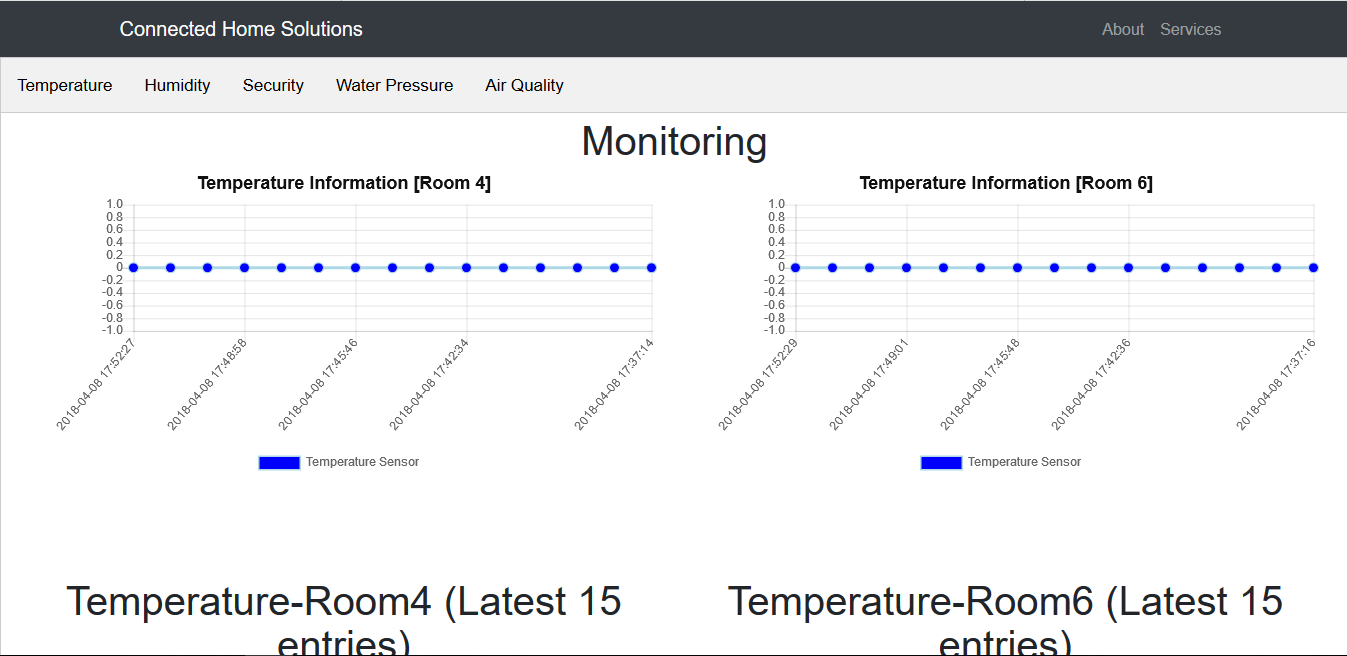
****

Clean, simple, and user-friendly Login Page

****

Consistency shown in various places throughout the webpage (e.g. Menu Bar)

****

****

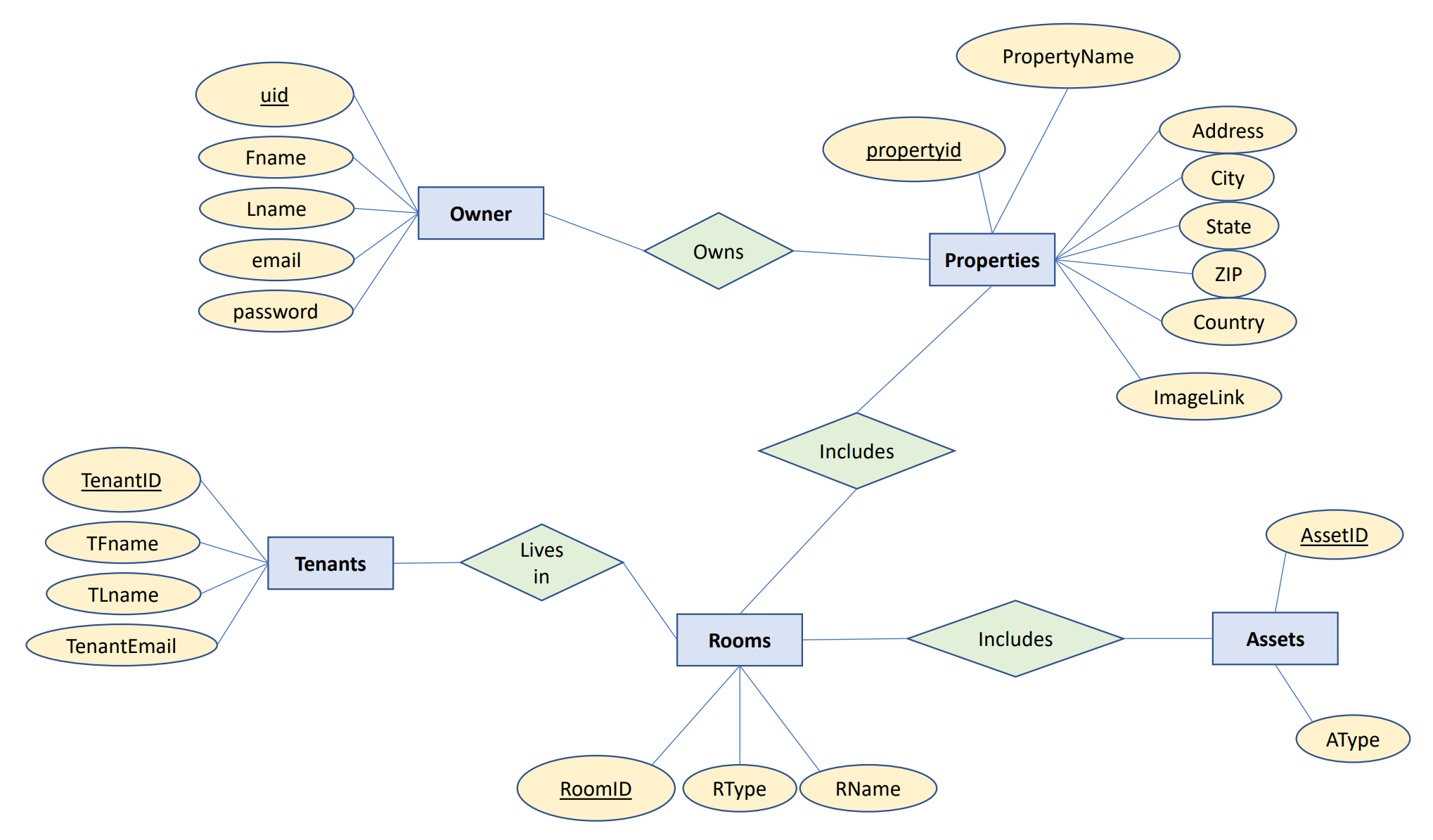
Interactive tables and vibrant colors make for a great visual tool when assessing data.

Alternating row colors in tables. This enhances ease-of-use and makes the information very accessible to the customer.

High-contrast text and easily readable pages

Section 6.0 – Static Design

**6.1 – Class Model**



**6.2 – System Operations Contract**

|  |  |
| --- | --- |
| **Name:** Account Registration  **Responsibilities:** Register a user account  **Cross-Reference:** UC-#01, General Requirement ID 01 | |
| **Pre-condition**   * User is on the account registration page | **Post-condition**   * User is returned to the account login page * “Account Created” notification is displayed |

|  |  |
| --- | --- |
| **Name:** Account Login  **Responsibilities:** User logs into their unique account  **Cross-Reference:** UC-#02, General Requirement ID 02 | |
| **Pre-condition**   * User is on the account login page | **Post-condition**   * User is on the property management dashboard * “Successful Login” notification is displayed |

|  |  |
| --- | --- |
| **Name:** Register Property  **Responsibilities:** Register a new property tied to a user’s unique account  **Cross-Reference:** UC-#03, General Requirement ID 03 | |
| **Pre-condition**   * User is on the property management dashboard | **Post-condition**   * User is returned to the property management dashboard * The newly registered property information is displayed on the property management dashboard |

|  |  |
| --- | --- |
| **Name:** Edit Property  **Responsibilities:** User makes changes to property information  **Cross-Reference:** UC-#04, General Requirement ID 04 | |
| **Pre-condition**   * User is on the property management dashboard | **Post-condition**   * User is returned to the property management dashboard * The updated property information is displayed on the property management dashboard |

|  |  |
| --- | --- |
| **Name:** Delete Property  **Responsibilities:** User deletes a property from the property management dashboard  **Cross-Reference:** UC-#05, General Requirement ID 05 | |
| **Pre-condition**   * User is on the property management dashboard | **Post-condition**   * User is returned to the property management dashboard * The property information is no longer displayed on the property management dashboard |

|  |  |
| --- | --- |
| **Name:** Remote Monitoring  **Responsibilities:** User views the sensor/asset information of the desired property  **Cross-Reference:** UC-#06, General Requirement ID 06 | |
| **Pre-condition**   * User is on the property management dashboard | **Post-condition**   * The sensor/asset dashboard displaying the live readings is displayed |

|  |  |
| --- | --- |
| **Name:** Remote Control  **Responsibilities:** User makes changes to sensor/asset information for the desired property  **Cross-Reference:** UC-#07, General Requirement ID 07 | |
| **Pre-condition**   * User is on the property management dashboard | **Post-condition**   * User is returned to the property management dashboard * The changes are reflected in the sensor/asset dashboard |

|  |  |
| --- | --- |
| **Name:** Cost-Saving Analytics  **Responsibilities:**  **Cross-Reference:** UC-#08, General Requirement ID 08 | |
| **Pre-condition**   * Use has a registered property and selected the option to view analytics | **Post-condition**   * System visually displays the cost-saving analytics breakdown page to the landlord/property manager |

**6.3 – Mathematical Model**

**Below is the algorithm used to script the cooling fan within the attic of the physical model to activate once the temperature reached an unsafe threshold. Please note that the cooling fan algorithm also appears in section 8.6 “Hardware Requirements”:**

cpu temp

topic/cpuTemp

PID Controller config

set point 53

K proportional 85

K integral 1

K differential 5

fan controller PID script

var PID = msg.payload; // PID Value

var cycle; // PWM duty cycle

var adjustment = 0; // duty cycle adjustment

var cycle = parseFloat(adjustment) + parseFloat(PID);

if(cycle<0){

msg.payload=0;

return msg;

}

else if(cycle>100){

msg.payload=1;

return msg;

}

else

{

msg.payload = parseFloat(cycle);

// 0 - 100

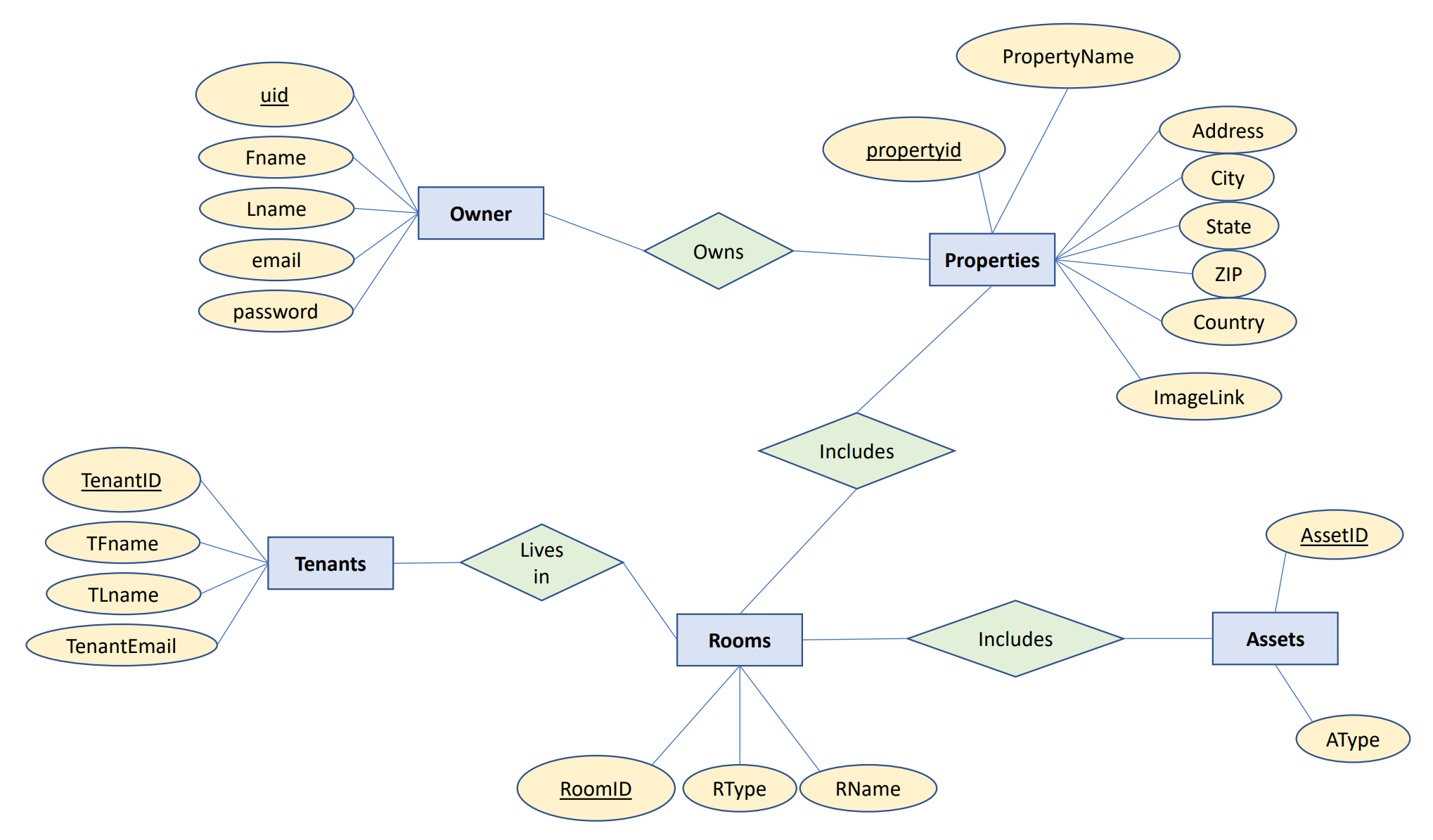
return msg;

}

return null;

input / output > topic/pidController

**6.4 – Entity Relation**



Section 7.0 – Dynamic Design

**7.1 – Sequence Diagram**

* Please see the “ConnectedHousingSolutions.asta” for the Activity and Sequence Diagrams corresponding to each use case.

**7.2 – Interface Specification**

* The Navigation bar will remain static at the top of the screen at all times.
* The property dashboard will constantly display all of the registered properties linked to each unique landlord account.
* A system dashboard will be displayed when interacting with the various system sensors/assets.

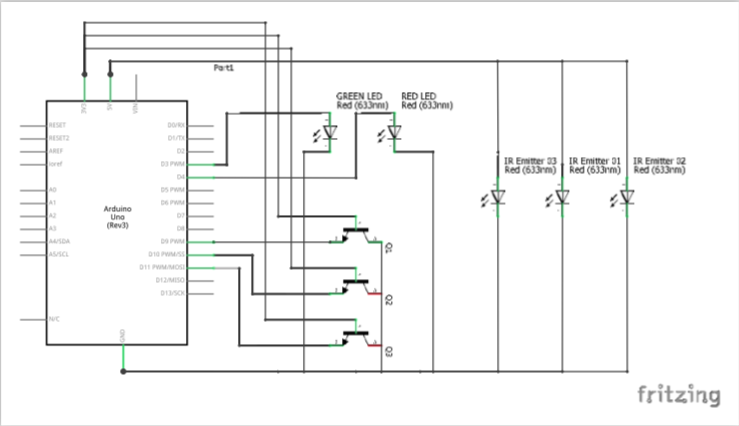
**7.3 – State Diagrams**

* Please see the “ConnectedHousingSolutions.asta” for the State Machine Diagram that outlines the overall system functionality.

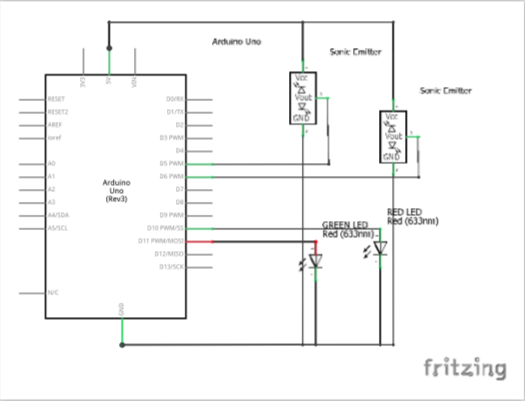
Section 8.0 – System Architecture and System Design

**8.1 – Subsystems / Component / Design Pattern Identification**

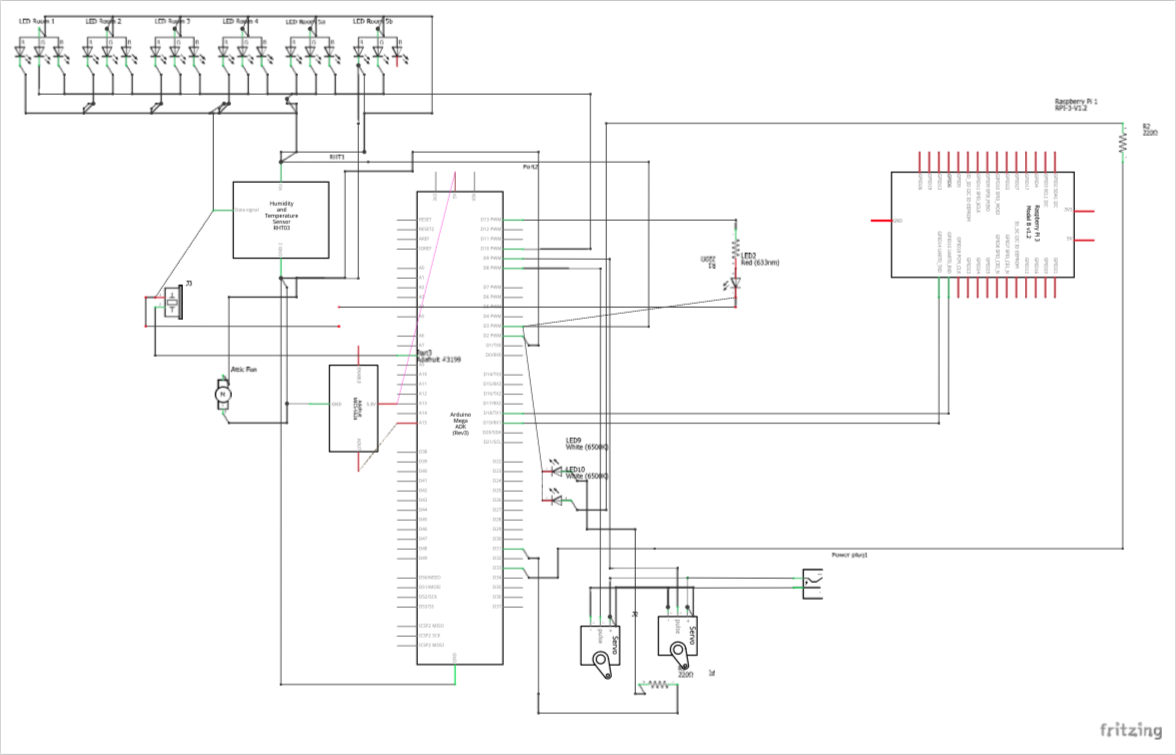
* The following wiring diagram is an overview of the Infrared Emitter system found within the Physical model.



* The following wiring diagram is an overview of the sonic emitter system found within the Physical model.



* The following wiring diagram is an overview of the lighting system found within the Physical model.



**8.2 – Mapping Subsystems to Hardware (Deployment Diagram)**

**Raspberry Pi (x2):** The Raspberry Pi devices are used to host the various servers that are required to run the connected Housing Solutions System. One raspberry pi hosts the MQTT server and MySQL database while the second raspberry pi device hosts the Wamp server used with the website application.

**Arduino Microcontrollers (x3):** The Arduino Microcontrollers are used to control the various sensors that occupy each room of the physical model. Details on the specific code used to script each sensor can be found in section 8.6 “Hardware Requirements”.

**Wireless Router (x1):** The wireless router occupies the attic of the physical model and is used to create a private VPN network that allows for the wireless communication between both the physical model and the laptop that is local hosting the website application. In addition, the router is used alongside the Node Red mobile dashboard that is hosted on the tablet device and allows for the remote control of the physical model’s lights and attic servo.

**Cooling Fan (x1):** The cooling fan is located in attic of the physical model and was installed alongside the rood servo to address the rick of overheating the raspberry pi devices within the attic. Due to the design of the physical model, the clustering of the various electronics within a confined space led to unsafe CPU temperatures within the raspberry pi devices. This resulted in input lag and device failure during original testing, therefore the cooling fan and servo used to lift the roof was added to address the heat build-up. For more details on the cooling algorithm used, please refer to either the previous section 6.3 “Mathematical Model” or the later section 8.6 “Hardware Requirements”.

**8.3 – Persistent Data Storage**

* MySQL Database will be utilized for storing the sensor/asset information in addition to the Landlord account information.
* The system uses two raspberry pi hardware devices as to split the CPU load and reduce the chances of overheating the overall system.
* The MySQL Database will store information with no data loss and feature security measures against MySQL data injection.
* For more information regarding the data structure of the MySQL database, please refer to section “9.2 – Data Structures”.

**8.4 – Network Protocol**

**To Install node red packages, implement the following script on nodered:**

Npm install 'package name here'

Node-red-contrib-bigtimer

Node-red-dashboard

Node-red-contrib-mqtt-broker

Node-red-contrib-camerapi

Node-red-contrib-usbcamera

Node-red-node-mysql

**Reboot pi to verify installs worked and did not mess up loading of node-red**

**The below process of changing the config file and changing / creating the admin account took around an two hours to do with much trial and error....**

**Process to generate admin password hash for Node red master login:**

cd /usr/lib/node\_modules/node-red/node\_modules

node -e "console.log(require('bcryptjs').hashSync(process.argv[1], 8));"   
password hash to generate goes here

**Adding the Password hash to the config file via nano. This also includes some modifications to the defualt config file:**

Cd /home/pi/.node-red

Sudo nano settings.js

adminAuth: {

type: "credentials",

users: [{

username: "admin",

password: "$2a$08$zZWtXTja0fB1pzD4sHCMyOCMYz2Z6dNbM6tl8sJogENOMcxWV9DN.", **>>>>>> THIS IS NOT THE REAL HASH!!!! <<<<<<<<**

permissions: "\*"

}]

},

**Additional config file changes to change the information storage path…**

userDir: '/home/pi/.node-red/',

nodesDir: '/home/pi/.node-red/nodes',

**After config file changes are made reboot and test. Then back up initial configuration to flash drive before continuing further…**

**To install the MySQL Server, implement the following script on the raspberry pi device:**

sudo apt-get install mysql-server python-mysqldb

sudo su

mysql -u root -p

CREATE DATABASE nodered;

USE nodered;

CREATE USER 'nodered'@'localhost' IDENTIFIED BY '\*\*\*\*';

GRANT ALL PRIVILEGES ON nodered.\* TO 'nodered'@'localhost';

FLUSH PRIVILEGES;

CREATE TABLE mqtt (id int(11) NOT NULL AUTO\_INCREMENT, timestamp datetime NOT NULL, topic text COLLATE utf8\_unicode\_ci NOT NULL, data text COLLATE utf8\_unicode\_ci NOT NULL, UNIQUE KEY id (id)) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;

quit;

**The above section creates a database of nodered and a user of nodered and the password for the user and grants access to the database. The above commands proceed to create a table of mqtt and columns of id; timestamp; topic; data**

Collation of utf8\_unicode\_ci was used to help with compatibility of MQTT

sudo /etc/init.d/mysql restart <--- resetarts mysql

**Grant access to external computers...**

sudo su

mysql -u root -p

GRANT ALL ON \*.\* TO 'nodered' @'192.168.1.%' IDENTIFIED BY 'passwordhere' with grant option;

Exit

Exit

Sudo nano /etc/mysql/my.cnf

Added the below line to the config file…

Bind-address = 0.0.0.0

**………………End config**

**Ctrl x**

**Y**

**Enter**

Sudo /etc/init.d/mysql restart

**Added login table for user logins…**

CREATE TABLE `nodered`.`login` (

`idlogin` INT NOT NULL AUTO\_INCREMENT,

`UserName` VARCHAR(45) NOT NULL,

`Password` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idlogin`));

**To Install the WAMP server, implement the following script on the raspberry pi device:**

**Ver: Apache2**

Sudo su

apt-get install apache2 -y

a2enmod rewrite

systemctl restart apache2

nano /etc/apache2/apache2.conf

<Directory /var/www/>

Options Indexes

FollowSymLinks

AllowOverride All

Require all granted

</Directory>

systemctl restart apache2

**To Install the necessary PHP support, implement the following script onto the raspberry pi device:**

apt-get install php libapache2-mod-php -y

systemctl restart apache2

**To Install the nescessary phpMyAdmin support, implement the following script onto the raspberry pi device:**

apt-get install phpmyadmin -y

**Verify it works….**

http://192.168.1.105/phpmyadmin/

**Grant full access to database…**

Sudo nano /etc/mysql/my.cnf

**Remove….**

Bind-address=0.0.0.0

Sudo /etc/init.d/mysql restart

Mysql -u root -p

**Grant all privileges on \*.\* to phpmyadmin@192.168.1.% identivied by '\*\*\*\*' with grant option;**

Exit

Sudo nano /etc/mysql/my.cnf

**Add…**

Bind-address=0.0.0.0

Sudo /etc/init.d/mysql restart  
 **8.5 – Global Control Flow**

* Please see the “ConnectedHousingSolutions.asta” to view the Global Control Diagram.

**8.6 – Hardware Requirement**

**-------------------------------**

**Raspberry Pi (x2)  
-------------------------------**

Model: 3B  
Version: 1.2

Memory: Recommended 32GB, Minimum 8GB

**To configure the raspberry pi that will host the MQTT server, use the following script:**sudo apt-get update | sudo apt-get upgrade -y   
chmod -R 777 /home/pi  
update-nodejs-and-nodered  
update-nodejs-and-nodered

**Then do the following…**  
Reboot pi  
sudo systemctl enable nodered.service <--- Bootup node red with PI  
Reboot Pi and try to get to web page to make sure it works   
Reboot pi a second time to verify….

**------------------------------  
Arduino (x3)  
------------------------------**Model: Arduino Mega 2560 REV3  
Code: A000067  
  
**To configure the Arduinos that will control the various sensors/assets, use the following script:**

**The following code is used to script both the door and water pressure sensors:**

#define cs1 7

#define cs2 8

#define p1 5

#define alertled 2

// Variables will change:

int doorState = 0;

int windowState = 0; // current state of the button

int lastdoorState = 0; // previous state of the button

int lastwindowState = 0;

int watersensorState = LOW;

int lastwaterState = 0;

void setup() {

// initialize the sensor pins as a input:

pinMode(cs1, INPUT);

pinMode(cs2, INPUT);

digitalWrite(cs1, HIGH);

digitalWrite(cs2, HIGH);

pinMode(p1, INPUT);

// initialize the LED as an output:

pinMode(alertled, OUTPUT);

// initialize serial communication:

Serial.begin(9600);

}

void loop() {

// read the sensors

windowState = digitalRead(cs1);

doorState = digitalRead(cs2);

watersensorState = digitalRead(p1);

// compare the state to its previous state

if (windowState != lastwindowState) {

if (windowState == LOW) {

Serial.println("w1closed");

lastwindowState = windowState;

digitalWrite(alertled, LOW);

}

if (windowState == HIGH) {

Serial.println("w1open");

digitalWrite(alertled, HIGH);

lastwindowState = windowState;

}

}

if (doorState != lastdoorState) {

if (doorState == LOW) {

Serial.println("d1closed");

lastdoorState = doorState;

digitalWrite(alertled, LOW);

}

if (doorState == HIGH) {

Serial.println("d1open");

digitalWrite(alertled, HIGH);

lastdoorState = doorState;

}

}

if (watersensorState != lastwaterState){

if (watersensorState == LOW){

digitalWrite(alertled, HIGH);

Serial.println("p1closed");

lastwaterState = watersensorState;

}

if (watersensorState == HIGH){

digitalWrite(alertled, LOW);

Serial.println("p1open");

lastwaterState = watersensorState;

}

}

}

**The following code is used to script the infrared laser sensor for motion detection:**

#define TRIG\_PIN 9

#define ECHO\_PIN 10

#define alertled 2

#define goodled 11

const unsigned int BAUD\_RATE=9600;

int lastDistance = 0;

void setup() {

pinMode(TRIG\_PIN, OUTPUT);

pinMode(ECHO\_PIN, INPUT);

pinMode(alertled, OUTPUT);

pinMode(goodled, OUTPUT);

Serial.begin(BAUD\_RATE);

}

void loop() {

digitalWrite(TRIG\_PIN, LOW);

delayMicroseconds(2);

digitalWrite(TRIG\_PIN, HIGH);

delayMicroseconds(10);

digitalWrite(TRIG\_PIN, LOW);

const unsigned long duration= pulseIn(ECHO\_PIN, HIGH);

int distance= duration/29/2;

delay(300);

if(lastDistance-distance>15){

digitalWrite(alertled, HIGH);

digitalWrite(goodled, LOW);

lastDistance = distance;

delay(2000);

}

if(lastDistance-distance>5){

digitalWrite(alertled, LOW);

digitalWrite(goodled, HIGH);

lastDistance = distance;

delay(2000);

}

if(lastDistance = distance){

digitalWrite(alertled, LOW);

digitalWrite(goodled, LOW);

}

}

**The following Arduino code is used to control the attic servo (used to lift the roof) and the led lighting within the physical model.**

#include <Servo.h>

Servo myservo;

int pos = 0;

#define alertled 22

#define cs1 24

#define cs2 25

#define r1led1 30

#define r1led2 31

#define r1led3 32

#define r2led1 33

#define r2led2 34

#define r2led3 35

#define r3led1 36

#define r3led2 37

#define r3led3 38

#define r4led1 39

#define r4led2 40

#define r4led3 41

#define r5led1 42

#define r5led2 43

#define r5led3 44

#define r5led4 45

#define aled1 46

#define aled2 47

#define aled3 48

#define aled4 49

void setup() {

Serial.begin(9600);

while (!Serial);

myservo.attach(9);

myservo.write(0);

pinMode(r1led1, OUTPUT);

pinMode(r1led2, OUTPUT);

pinMode(r1led3, OUTPUT);

pinMode(r2led1, OUTPUT);

pinMode(r2led2, OUTPUT);

pinMode(r2led3, OUTPUT);

pinMode(r3led1, OUTPUT);

pinMode(r3led2, OUTPUT);

pinMode(r3led3, OUTPUT);

pinMode(r4led1, OUTPUT);

pinMode(r4led2, OUTPUT);

pinMode(r4led3, OUTPUT);

pinMode(r5led1, OUTPUT);

pinMode(r5led2, OUTPUT);

pinMode(r5led3, OUTPUT);

pinMode(r5led4, OUTPUT);

pinMode(aled1, OUTPUT);

pinMode(aled2, OUTPUT);

pinMode(aled3, OUTPUT);

pinMode(aled4, OUTPUT);

pinMode(cs1, INPUT);

pinMode(cs2, INPUT);

pinMode(alertled, OUTPUT);

digitalWrite(cs1, HIGH);

digitalWrite(cs2, HIGH);

}

void loop() {

if(digitalRead(cs1)==LOW){

digitalWrite(alertled, LOW);

}

else{

digitalWrite(alertled, HIGH);

// Serial.write("d1open");

}

if(digitalRead(cs2)==LOW){

digitalWrite(alertled, LOW);

}

else{

digitalWrite(alertled, HIGH);

// Serial.write("w1open");

}

if (Serial.available())

{

int state = Serial.parseInt();

if (state == 0)

{

digitalWrite(r1led1, LOW);

digitalWrite(r1led2, LOW);

digitalWrite(r1led3, LOW);

digitalWrite(r2led1, LOW);

digitalWrite(r2led2, LOW);

digitalWrite(r2led3, LOW);

digitalWrite(r3led1, LOW);

digitalWrite(r3led2, LOW);

digitalWrite(r3led3, LOW);

digitalWrite(r4led1, LOW);

digitalWrite(r4led2, LOW);

digitalWrite(r4led3, LOW);

digitalWrite(r5led1, LOW);

digitalWrite(r5led2, LOW);

digitalWrite(r5led3, LOW);

digitalWrite(r5led4, LOW);

digitalWrite(aled1, LOW);

digitalWrite(aled2, LOW);

digitalWrite(aled3, LOW);

digitalWrite(aled4, LOW);

myservo.write(0);

delay(15);

}

if (state == 1)

{

digitalWrite(r1led1, HIGH);

digitalWrite(r1led2, HIGH);

digitalWrite(r1led3, HIGH);

}

if (state == 2)

{

digitalWrite(r2led1, HIGH);

digitalWrite(r2led2, HIGH);

digitalWrite(r2led3, HIGH);

}

if (state == 3)

{

digitalWrite(r3led1, HIGH);

digitalWrite(r3led2, HIGH);

digitalWrite(r3led3, HIGH);

}

if (state == 4)

{

digitalWrite(r4led1, HIGH);

digitalWrite(r4led2, HIGH);

digitalWrite(r4led3, HIGH);

}

if (state == 5)

{

digitalWrite(r5led1,HIGH);

digitalWrite(r5led2, HIGH);

digitalWrite(r5led3, HIGH);

digitalWrite(r5led4, HIGH);

}

if (state == 6)

{

digitalWrite(aled1, HIGH);

digitalWrite(aled2, HIGH);

digitalWrite(aled3, HIGH);

digitalWrite(aled4, HIGH);

myservo.write(130);

delay(15);

}

}

}

**--------------------------------  
Cooling Fan (x1)**

**--------------------------------**

**To configure the small electronic cooling fan to cool the raspberry pi system, implement the following algorithm:**cpu temp

topic/cpuTemp

PID Controller config

set point 53

K proportional 85

K integral 1

K differential 5

fan controller PID script

var PID = msg.payload; // PID Value

var cycle; // PWM duty cycle

var adjustment = 0; // duty cycle adjustment

var cycle = parseFloat(adjustment) + parseFloat(PID);

if(cycle<0){

msg.payload=0;

return msg;

}

else if(cycle>100){

msg.payload=1;

return msg;

}

else

{

msg.payload = parseFloat(cycle);

// 0 - 100

return msg;

}

return null;

input / output > topic/pidController

Section 9.0 – Algorithms and Data Structures

**9.1 – Algorithms**

**The following is an algorithm for the cooling fans found on the physical model that are responsible for keeping the system in safe operating temperatures:**

**-----------------------------**

**Fan Cooling**

**-----------------------------**

cpu temp

topic/cpuTemp

PID Controller config

set point 53

K proportional 85

K integral 1

K differential 5

fan controller PID script

var PID = msg.payload; // PID Value

var cycle; // PWM duty cycle

var adjustment = 0; // duty cycle adjustment

var cycle = parseFloat(adjustment) + parseFloat(PID);

if(cycle<0){

msg.payload=0;

return msg;

}

else if(cycle>100){

msg.payload=1;

return msg;

}

else

{

msg.payload = parseFloat(cycle);

// 0 - 100

return msg;

}

return null;

input / output > topic/pidController

**9.2 – Data Structures**

**The queries for creating the Connected Housing Solution’s MySQL Database schema that will house all of the necessary system information is as follows…**

CREATE TABLE `nodered`.`owners` (

`ownerid` INT NOT NULL AUTO\_INCREMENT,

`ofname` VARCHAR(45) NOT NULL,

`olname` VARCHAR(45) NOT NULL,

`owneremail` VARCHAR(45) NOT NULL,

`password` VARCHAR(45) NOT NULL,

PRIMARY KEY (`ownerid`),

UNIQUE INDEX `ownerid\_UNIQUE` (`ownerid` ASC));

CREATE TABLE `nodered`.`properties` (

`propertyid` INT NOT NULL AUTO\_INCREMENT,

`streetaddress` VARCHAR(45) NOT NULL,

`city` VARCHAR(45) NOT NULL,

`state` VARCHAR(45) NOT NULL,

`zip` VARCHAR(45) NOT NULL,

`aptname` VARCHAR(45) NOT NULL,

`ownerid` INT NOT NULL,

PRIMARY KEY (`propertyid`),

UNIQUE INDEX `propertyid\_UNIQUE` (`propertyid` ASC),

INDEX `ownerid\_idx` (`ownerid` ASC),

CONSTRAINT `ownerid`

FOREIGN KEY (`ownerid`)

REFERENCES `nodered`.`owners` (`ownerid`)

ON DELETE NO ACTION

ON UPDATE NO ACTION);

CREATE TABLE `nodered`.`rooms` (

`roomid` INT NOT NULL AUTO\_INCREMENT,

`rtype` VARCHAR(45) NOT NULL,

`rname` VARCHAR(45) NOT NULL,

`propertyid` INT NOT NULL,

PRIMARY KEY (`roomid`),

UNIQUE INDEX `roomid\_UNIQUE` (`roomid` ASC),

INDEX `propertyid\_idx` (`propertyid` ASC),

CONSTRAINT `propertyid`

FOREIGN KEY (`propertyid`)

REFERENCES `nodered`.`properties` (`propertyid`)

ON DELETE NO ACTION

ON UPDATE NO ACTION);

CREATE TABLE `nodered`.`tenants` (

`tenantid` INT NOT NULL AUTO\_INCREMENT,

`tfname` VARCHAR(45) NOT NULL,

`tlname` VARCHAR(45) NOT NULL,

`tenantemail` VARCHAR(45) NOT NULL,

`roomid` INT NOT NULL,

PRIMARY KEY (`tenantid`),

UNIQUE INDEX `tenantid\_UNIQUE` (`tenantid` ASC),

INDEX `roomid\_idx` (`roomid` ASC),

CONSTRAINT `roomid`

FOREIGN KEY (`roomid`)

REFERENCES `nodered`.`rooms` (`roomid`)

ON DELETE NO ACTION

ON UPDATE NO ACTION);

CREATE TABLE `nodered`.`assets` (

`assetid` INT NOT NULL AUTO\_INCREMENT,

`atype` VARCHAR(45) NULL,

`assetlocation` INT NULL,

PRIMARY KEY (`assetid`),

UNIQUE INDEX `assetid\_UNIQUE` (`assetid` ASC),

INDEX `assetlocation\_idx` (`assetlocation` ASC),

CONSTRAINT `assetlocation`

FOREIGN KEY (`assetlocation`)

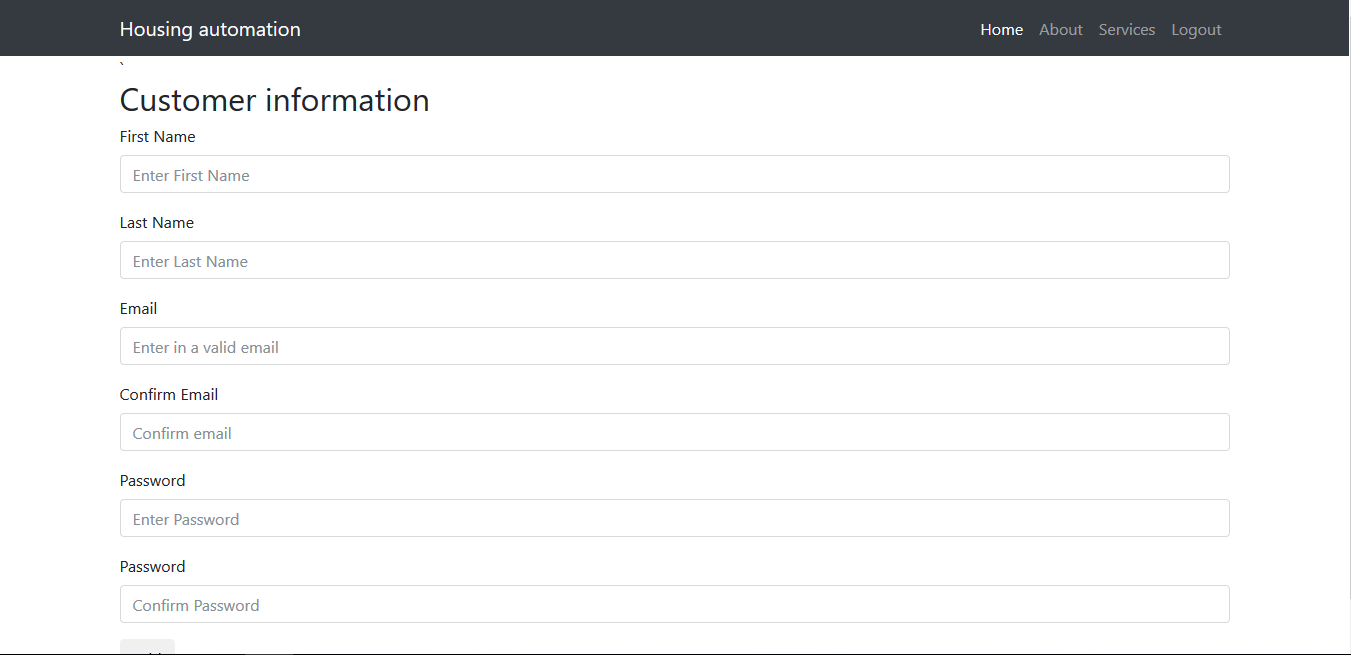
REFERENCES `nodered`.`rooms` (`roomid`)

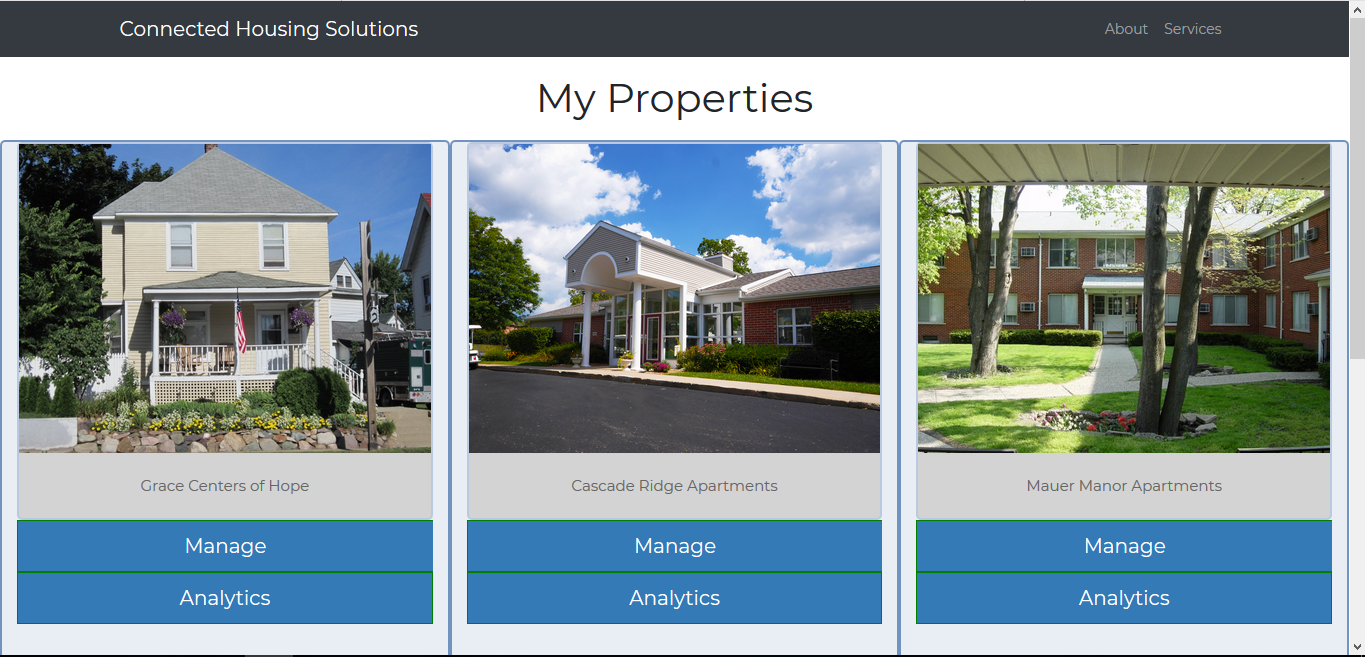
ON DELETE NO ACTION

ON UPDATE NO ACTION);

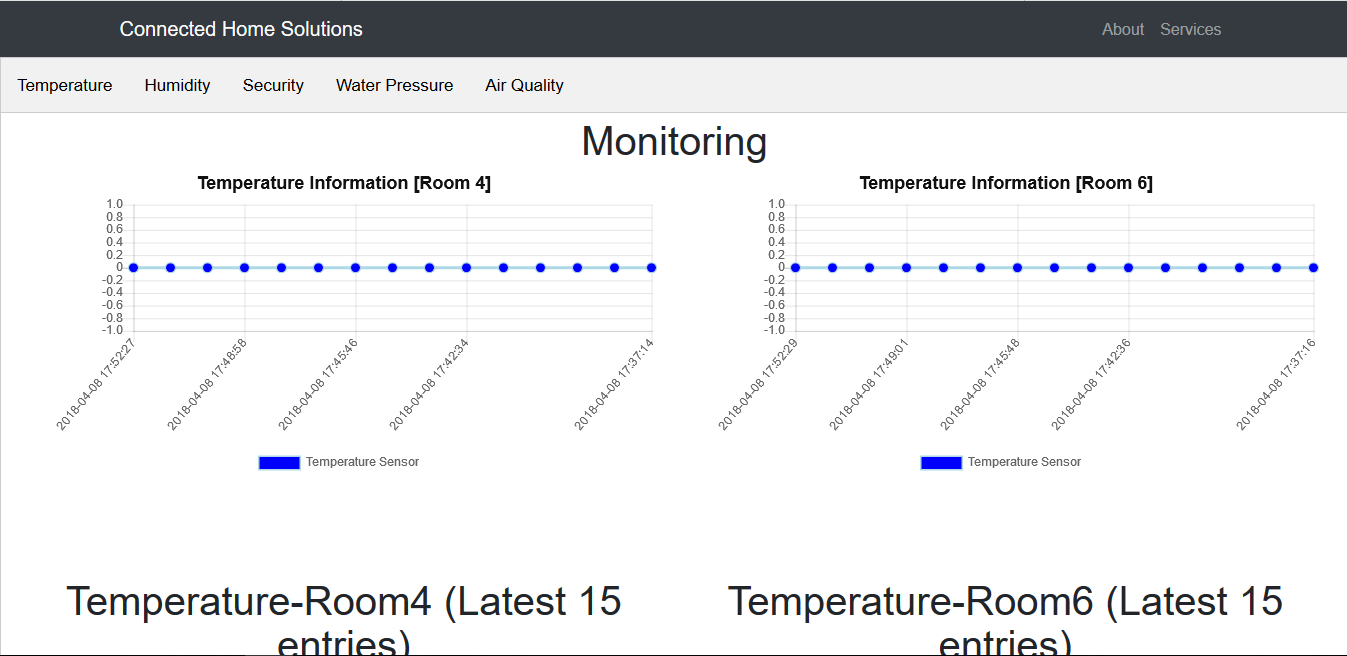
Section 10.0 – User Interface Design and Implementation

**10.1 – User Interface Design**

* Below is the landlord registration form that must be completed for use with the home automation system. The registration form is straightforward and provides a confirmation message upon successful registration.  
    
  
* Below is the property management dashboard that lists all of the registered properties that are linked to a landlord’s account. The property information takes center stage within the design as to avoid unnecessary page navigation. In addition, the navigation bar remains relatively simple throughout the application as to further emphasize user simplicity.

****

* Below is the analytics page that provides a cost-analysis breakdown using the readings from the various sensors. The graphs are generated using “Chart JS” and the colors selected are chosen to emphasize clarity and ease of viewing.



**10.2 – User Interface Implementation**

* The end-user interface was implemented using the following frameworks; HTML, CSS, JavaScript, PHP Script, AJAX and NodeRed. The Node Red framework was used specifically for use with the mobile dashboard application that is hosted on the tablet device and used to control the physical model’s lighting and attic servo.
* The layout for both the website and mobile dashboard applications were kept as intuitive as possible by limiting the number of page navigations required to access each feature. For example, the decision to list the notifications center directly under the property was chosen to be both convenient and reduce the time required to receive urgent alerts.
* The mobile dashboard used to control the physical model was styled specifically to fit both tablet and building control panels similar to the digital touch screens used for controlling a building’s heating. This decision was due to the fact the NodeRed Dashboard was to be made available to local emergency services that may require the information to combat a specific threat. For example, should a building catch on fire and local fire department arrives, the control panel would allow them to view the temperature which could potentially assist them in locating the origin and location of the fire.

Section 11.0 – Testing

**11.1 – Unit Test Architecture and Strategy/Framework**

The following is a breakdown of the overall scale of testing to be performed on the Connected Housing Solutions system. The order descends from top-level to each functionality contained within. (Ex: In order to test the cost-saving analytics feature, a property must be registered, and a user must be logged in, however the user must first be registered in order to access the system. Therefore cost-saving analytics contains the use cases for register property, user login and register user respectively):

1. Cost-Saving Analytics 🡪 Register Property 🡪 User Login 🡪 Register User
2. Physical Model 🡪 NodeRed Mobile Dashboard 🡪 Raspberry Pi 🡪 Arduino Microcontrollers

**11.2 – Unit test definition, test data selection**

* The test data for this project was separated into both hardware testing as well as software testing. As such, the data for testing each system feature individually can be found in the respective tables below:

|  |  |
| --- | --- |
| **Hardware** | |
| Device | Test Procedure |
| Raspberry Pi | Test involved pinging the servers hosted on each device from within the VPN to ensure that they were properly hosted and responding. |
| Arduino Microcontrollers | Test involved building small proof of concept mockups that were presented each sprint. These mockups allowed for testing the Arduino code logic as well as the sensor being demonstrated before moving each into the physical model. |
| Cooling Fan | Test involved holding a hairdryer to the raspberry pi device and waiting for the fan to spin up once the temperature range reached the desired threshold. In addition, a red led light accompanied this demo as a visual representative that the algorithm worked as intended. |
| Attic Servo | Test involved activating the servo a number of times to test both the durability of the servo as well as the physical construction of the Attic’s hinge. |

|  |  |
| --- | --- |
| **Software** | |
| Device | Test Procedure |
| Register / Login | 1. Register a new user and check the MySQL Database for confirmation that the account was created. 2. Login to the newly registered user account and ensure that it takes you to the property management dashboard page. 3. Attempt to register the same user twice and ensure that the database doesn’t add the same user again. 4. Attempt to register a new user with non-valid account information (ex: Password is too short). 5. Attempt to login with no account information entered (Error checking of login feature). |
| Register Property | 1. Register a new property and check the MySQL database for confirmation that the property was created. 2. Attempt to register a blank property (Error checking of register property feature). |
| Generating Dynamic Property Information | 1. Register a new property and ensure that the property information is within the MySQL database and that the AJAX script is dynamically generating the property information. 2. Login to another user in order to make sure that the property information isn’t being displayed to every user and that the property information is unique to the specific user account. |
| NodeRed Mobile Dashboard | 1. Test each individual room’s control functionality within the physical model to make sure that there are no control issues. 2. Check the packets that are being sent out to both the mobile interface and website’s notification center from the NodeRed server configuration interface at NodeRed.com. |
| Cost-Saving Analytics | 1. Test the live updates against the physical model (ex: opening the front door and checking the status on the website application). 2. Launching the analytics page and making sure that each sensor is providing data to be graphed by the chart.js API. |

**11.3 – System Test Specification**

* The system tests for the website application were carried out on both Windows and Macintosh devices.
* The individual sensors were presented in mockup models that were deliverables for each sprint to demonstrate the proof of concept.
* The physical model was tested alongside the mobile NodeRed Dashboard as well as the website application to ensure that the network hosted within the attic would consistently deliver the correct information and respond as expected. For more information of the issues found and addressed during this phase of testing please see ‘ConnectedHousingSolutionsRMM.xlsx”.

**11.4 – Test Reports per Sprint**

|  |  |
| --- | --- |
| **Test-Case Identifier:** TC-1  **Sprint Iteration(s):** Sprint 1, Sprint 2  **Use Case Tested:** UC-#01 Account Registration, main success scenario  **Pass/Fail Criteria:** The test passes if the user’s account information is registered within the database.  **Input Data:** Characters, Numeric | |
| **Test Procedure:** | **Expected Result:** |
| **Step 1:** User enters valid account information and presses the submit button.  **Step 2:** User enters invalid account information (ex: duplicate information) and presses the submit button. | The system registers the account information within the database and returns the user to the login screen.   The system does not register the account information within the database and prompts the user to “try again”. |
| **Pass/Fail:** | **Input Data:** |
| Pass | Test with valid account information. |
| Pass | Test with invalid account information. |
| Fail | Test with duplicate account information. |
| Pass | Test with leaving entry fields blank. |

|  |  |
| --- | --- |
| **Test-Case Identifier:** TC-2  **Sprint Iteration(s):** Sprint 1, Sprint 2  **Use Case Tested:** UC-#02 Account Login, main success scenario  **Pass/Fail Criteria:** The test passes if the user successfully logs into their unique account that was previously registered within the database.  **Input Data:** Characters, Numeric | |
| **Test Procedure:** | **Expected Result:** |
| **Step 1:** User enters valid account information and presses the login button.  **Step 2:** User enters invalid account information (ex: wrong username and/or password) and presses the login button. | The system checks the account information against the database and logs the user into the main page.   The system checks the account information against the database and does not login the user. In addition, an error message is displayed that prompts the user to “try again”. |
| **Pass/Fail:** | **Input Data:** |
| Pass | Test with valid login account information. |
| Pass | Test with invalid login account information. |
| Pass | Test with leaving entry fields blank. |

|  |  |
| --- | --- |
| **Test-Case Identifier:** TC-3  **Sprint Iteration(s):** Sprint 3  **Use Case Tested:** UC-#03 Register Property, main success scenario  **Pass/Fail Criteria:** The test passes if the user can successfully register a property tied to their unique account within the database.  **Input Data:** Characters, Numeric | |
| **Test Procedure:** | **Expected Result:** |
| **Step 1:** User enters valid property information and presses the submit button.  **Step 2:** User enters invalid property information (ex: empty field) and presses the submit button. | The system registers the property information within the database and returns the user to the property management screen.  The system does not register the property information within the database and prompts the user to “try again”. |
| **Pass/Fail:** | **Input Data:** |
| Pass | Test with valid property registration information. |
| Pass | Test with invalid property registration information. |
| Pass | Test with leaving entry fields blank. |

|  |  |
| --- | --- |
| **Test-Case Identifier:** TC-4  **Sprint Iteration(s):** Sprint 4  **Use Case Tested:** UC-#04 Edit Property, main success scenario  **Pass/Fail Criteria:** The test passes if the user can successfully update or change the property information registered within the database in addition to having the new information displayed on the property management screen.  **Input Data:** Characters, Numeric | |
| **Test Procedure:** | **Expected Result:** |
| **Step 1:** User edits the property information using the provided form and presses the submit button.  **Step 2:** User enters invalid property information (ex: empty field) and presses the submit button. | The system registers the new property information within the database and returns the user to the property management screen.   The system does not register the new property information within the database and prompts the user to “try again”. |
| **Pass/Fail:** | **Input Data:** |
| Pass | Test edit property feature with valid updated property information. |
| Pass | Test edit property feature with invalid updated property information. |
| Pass | Test with leaving entry fields blank. |

|  |  |
| --- | --- |
| **Test-Case Identifier:** TC-5  **Sprint Iteration(s):** Sprint 5  **Use Case Tested:** UC-#05 Delete Property, main success scenario  **Pass/Fail Criteria:** The test passes if the user successfully deletes the property information registered within the database and the property is no longer displayed on the property management screen.  **Input Data:** Boolean (“Yes” or “No”, at delete property confirmation message) | |
| **Test Procedure:** | **Expected Result:** |
| **Step 1:** User confirms selection of the “delete” property option.  **Step 2:** User does not confirm selection of the “delete” property option. | The system deletes the property information registered within the database and also removes the property from the property management page.  The system does not delete the property information registered within the database and does not remove the property from the property management page. |
| **Pass/Fail:** | **Input Data:** |
| Pass | Test with confirming the option to delete the property information. |
| Pass | Test with denying the option to delete the property information. |

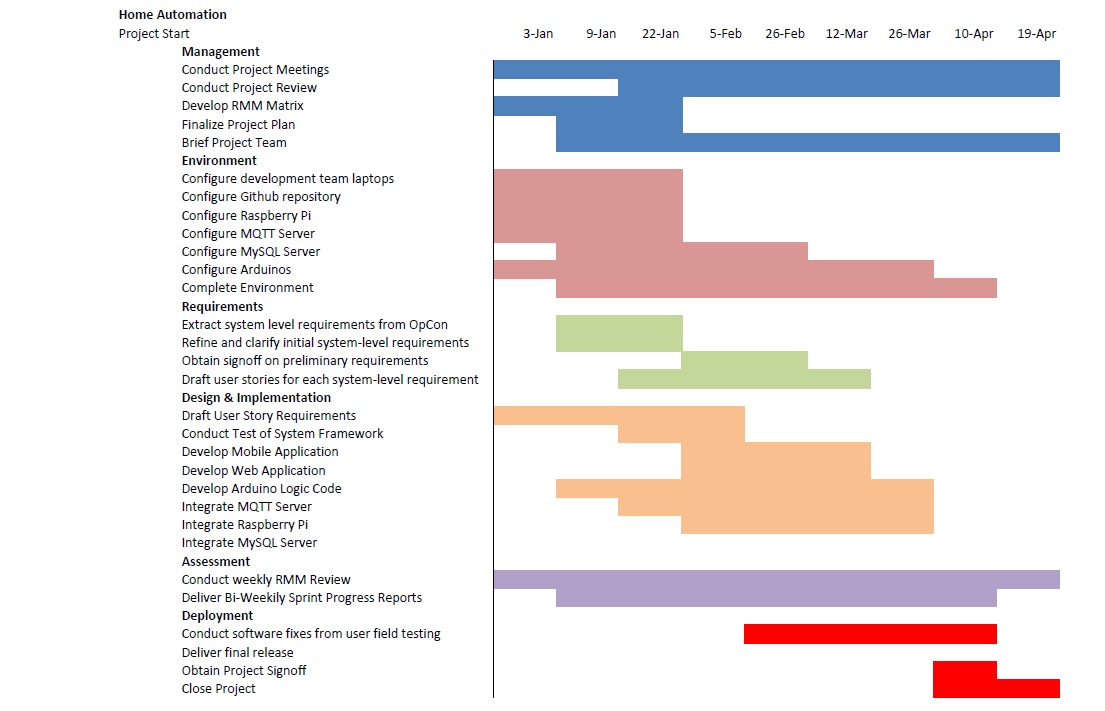
|  |  |
| --- | --- |
| **Test-Case Identifier:** TC-6  **Sprint Iteration(s):** Sprint 5  **Use Case Tested:** UC-#06 Remote Monitoring, main success scenario  **Pass/Fail Criteria:** The test passes if the user successfully views sensor/asset information for the desired property.  **Input Data:** None | |
| **Test Procedure:** | **Expected Result:** |
| **Step 1:** User selects a property and then proceeds to select the option to view the live sensor/asset information. | The system displays the sensor/asset dashboard along with the live readings as reported by the MQTT Server. |
| **Pass/Fail:** | **Input Data:** |
| Pass | Test the feature to provide live readings for a pre-configured test property. |

|  |  |
| --- | --- |
| **Test-Case Identifier:** TC-7  **Sprint Iteration(s):** Sprint 6  **Use Case Tested:** UC-#07  **Pass/Fail Criteria:** The test passes if the user successfully changes sensor/asset information for the desired property.  **Input Data:** None | |
| **Test Procedure:** | **Expected Result:** |
| **Step 1:** User selects a property and then proceeds to select the option to change the live sensor/asset information. | The system displays the sensor/asset dashboard along with the live changes made as reported by the MQTT Server. |
| **Pass/Fail:** | **Input Data:** |
| Pass | Test the feature to make live changes for a pre-configured test property. |

|  |  |
| --- | --- |
| **Test-Case Identifier:** TC-#08  **Sprint Iteration(s):** Sprint 6  **Use Case Tested:** UC-#08  **Pass/Fail Criteria:** The test passes if the user successfully can successfully view cost-saving analytics based on sensor/asset information for the desired property.  **Input Data:** None | |
| **Test Procedure:** | **Expected Result:** |
| **Step 1:** User selects a property and then proceeds to select the option to generate cost-saving analytics based on the live sensor/asset information. | The system displays the various cost-saving analytics within the property dashboard for the desired property. |
| **Pass/Fail:** | **Input Data:** |
| Pass | Test the feature to provide cost-saving analytics based on sensor readings for a pre-configured test property. |

Section 12.0 – Project Management

**12.1 – Project Plan**



**12.2 – Risk management**

* Please see “Connected Housing Solutions RMM.xlsx” for details regarding the risk management matrix.

Section 13.0 – References

* **The following reference links were used within the development of this project documentation:**

1. “Best Practice.” Info-Entrepreneurs, Canada Business Network, [www.infoentrepreneurs.org/en/guides/best-practice/](http://www.infoentrepreneurs.org/en/guides/best-practice/)
2. Brisson, Amy. “Impact of Affordable Housing on Families and Communities.” *Impact of Affordable Housing on Families and Communities.pdf*, Enterprise Community Partners Inc., May 2011, <http://homeforallsmc.com/wp-content/uploads/2017/05/Impact-of-Affordable-Housing-on-Families-and-Communities.pdf>
3. Critchley, Will, and Klaus Siegert. “Socio-Economic Factors and Project Management.” <http://www.fao.org/docrep/U3160E/u3160e09.htm>
4. Delgado, Rick. “Top 10 Benefits of Automating Your Home.” Freshome.com, Freshome, 7 May 2014, <https://freshome.com/2013/01/17/top-10-benefits-of-automating-your-home/>
5. “Ethical & Security Issues in Information System.” Free Training Tutorials & Video for IT Courses, Guru99, 2018, <https://www.guru99.com/mis-ethical-social-issue.html>
6. “Project Management Guide.” Https://Www.wrike.com/, Wrike, <https://www.wrike.com/project-management-guide/faq/what-is-a-stakeholder-in-project-management/>
7. Root, George N. “10 Most Important Business Objectives.” *Chron.com*, Chron, 26 Mar. 2018, <http://smallbusiness.chron.com/10-important-business-objectives-23686.html>
8. Thiry, Michael. “The Program Management Actors.” Project Manager, ProjectManager.com.au, 6 June 2012, <http://projectmanager.com.au/program-management-actors/>
9. “UML Operation Contract.” UML Operation Contract, <http://www.comptechdoc.org/independent/uml/begin/umlopcontract.html>
10. “UML 2 Activity Diagram.” Sparx Systems - UML 2 Tutorial, Sparx Systems, 5 July 2017, <https://www.sparxsystems.com/resources/uml2_tutorial/uml2_activitydiagram.html>