ARANEA

Digital Home
Software Architecture Document

Version 3.0
4/23/2017

Document Control

Approval

The Guidance Team and the customer shall approve this document:

Dr. Salamah Salamah and Elsa Tai

Document Change Control

Initial Release:	2/28/2017 V1.0
Current Release:	4/23/2017 V3.0
Indicator of Last Page in Document:	Ώ
Date of Last Review:	4/23/2017
Date of Next Review:	TBA
Target Date for Next Update:	TBA

Distribution List

This following list of people shall receive a copy of this document every time a new version of this document becomes available:

Customer:	Elsa Tai
Software Team Members:	Genesis Bejarano Eric Camacho Juan Ochoa Angel Ortega Aldo Perez

Change Summary

The following table details changes made between versions of this document

Version	Date	Modifier	Description
0.1	2/21/2017	ARANEA	Began §3.
0.1	2/27/2017	ARANEA	Review of §3 and corrections.
1.0	2/28/2017	ARANEA	Review and modification of §3. Addition of §2. Creation of Appendix. Initial release.
1.1	3/26/2017	ARANEA	Modification of Generalization and Layered Style.
2.0	3/26/2017	ARANEA	Review and release.
2.1	4/23/2017	ARANEA	Added Component and Connector Views.
3.0	4/23/2017	ARANEA	Review and release.

1 Introduction

1.1 Purpose and Intended Audience

The purpose of this document is to provide documentation on the software architecture that would be implemented for the desired system at hand. This document has the intention to establish an architecture style that must be followed by the development teams throughout the development of the product. Audience for this document are clients and the development team.

1.2 Scope of Product

DigitalHome (DH) would be a smart house in which the user shall be allow to manage devices that control the environment of a home. The user communicates through a personal web page or mobile application on the DigitalHome web server or on a local home server. The DH server shall communicate through a home wireless gateway device to all security sensors and appliance devices. Goal is to have a system in which allows homeowners to easily manage their daily lives by providing for a lifestyle that brings together security, environmental and energy management (temperature, humidity and lighting), entertainment, and communications.

1.3 Definitions, Acronyms, and Abbreviations

The following subsections contain definitions, acronyms and abbreviations, and an overview of the documents necessary for complete understanding of the document.

1.3.1 Definitions

The following section contains a list of terms and definitions that will be used in the rest of the document. The terms in this section might have multiple or ambiguous definitions, for this reason, we specify the definition for the term we will be using throughout the document.

Term	**Definition**
View	A View is a set of styles that describe different aspects of the system.
Style	A Style highlight only the components relevant to the emphasis of the style in perspective.

1.3.2 Acronyms and Abbreviations

The following section will list the acronyms and abbreviations that will be used in the rest of the document.

Acronym/Abbreviation	**Definition**
DH	Digital Home
AADL	Architecture Analysis and Design Language

i.e.	For example
etc	And other things
SC	System Controller
DSC	Degraded System Controller
UI	User Interface
GUI	Graphical User Interface
DGeneralUser	Degraded General User
DMasterUser	Degraded Master Unit
DTechnicican	Degraded Technician
Backup_DB	Backup Database
DataLogger_DB	Data Logger Database
System Settings_DB	System Setting Database
Security_DB	Security Database
Report_DB	Report Database

1.4 Overview

The Software Architecture Document is divided into 5 sections, these sections are:

1. Introduction of the document where the purpose of the document can be found along with the scope of the product.
2. Quality Attribute Scenario is the section that contains scenarios that fulfill the different quality attributes that were determine by various stakeholders.
3. Module View section shows how of the different module styles are been applied to the system.
4. Component and Connector View section provides information in how the elements considered for this system have some runtime presence, such as processes, objects, clients, servers, and data stores.
5. Allocation View section show a mapping between different software elements (either module view or component view) and non-software elements.

1.5 References

[1] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, and R. Little, *Documenting Software Architectures: Views and Beyond*, 2nd ed. United States: Addison-Wesley Educational Publishers, 2010.

[2] J. Michael, *DigitalHome Software Requirements Specification*, version 1.3. October 27,2010.

2 Quality Attribute Scenarios

Section 2 provides a scenario of quality attributes that describe the structure and behavior of our Digital Home Architecture in Section 2.1. Quality Attributes ensure a system architecture is sound and reliable. They are the standards and characteristics a software architecture must meet in order to build a good software system.

2.1 Quality Attributes

Section 2.1 contains a list of scenarios for each of the following quality attributes, modifiability, testability, security, availability, performance, and usability.

2.1.1 Modifiability Scenarios

The ease and effort of making changes to the system.

2.1.2 Testability Scenarios

How much code coverage is done on the system and the effort required to test the system.

2.1.3 Security Scenarios

Ensures the safety of the software system by controlling access to the system, code principle security procedures and locates hotspots of the system that require protection.

2.1.4 Availability Scenarios

The software system's ability to handle crashes and prevent downtime.

2.1.5 Performance Scenarios

The speed and accuracy a software system can perform an operation.

2.1.6 Usability Scenarios

The easy or amount of effort that is required to perform an operation in a software system.

3 Module Views

Section 3 describes a series of different architectural module views that shall be implement in order to fulfill the quality attribute that were discussed among several stakeholders for the DH system. Module View refers to the documentation that establishes the principal implementation units, or modules, of a system, together with the relations between them. These module views are styles that would be applied to the system by the architecture team. For the scope of the DH we are only including decomposition view, uses view, layer view, and generalization view [1].

3.1 Decomposition View

The decomposition style is used for decomposing a system into units of implementation. A decomposition view describes the organization of the code as modules and submodules and shows how system responsibilities are partitioned across them.

3.1.1 Primary Presentation

Figure 3.1.1 represent how the decomposition style is implemented in the DH System. The AADL representation of the decomposition view can be found in the Appendix A.

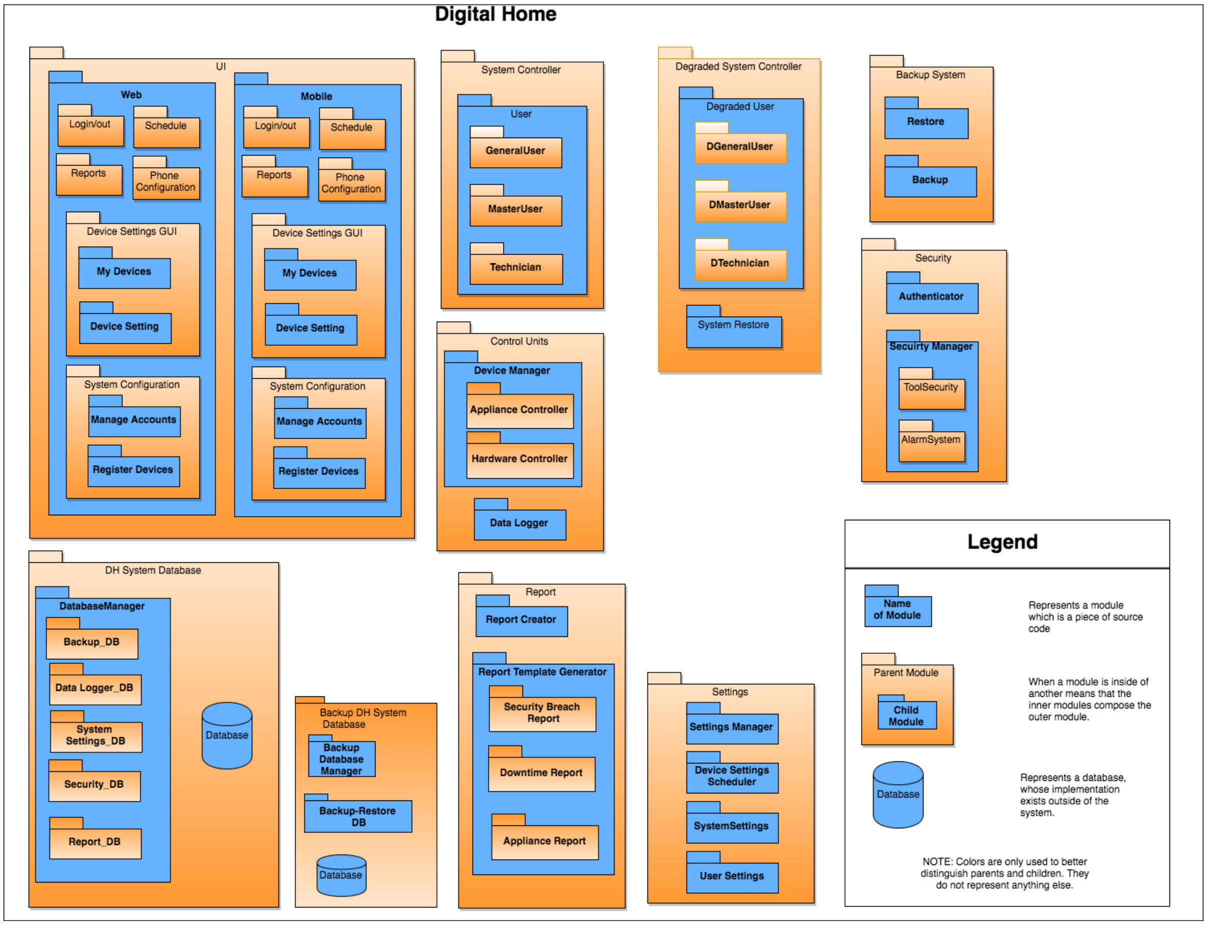

Figure 1: Decomposition View of Digital Home System

3.1.2 Element Catalog

3.1.2.1 Elements and Their Properties
The elements in the decomposition style are *Modules*. The modules and their properties can be seen in Table 1. Each element has been given an element number. Properties of an element can include: name, responsibility, visibility of interfaces, implementation information, mapping to source code units, test information, management information, and implementation constraints.

Element	**Properties**
E.1	**Name:** User Interface **Responsibility:** User interface is responsible to present the user with a graphical representation of the system, which will allow a user to do what they need to do on the system.
E.2	**Name:** Login/Logout **Responsibility:** The part of the GUI that allows a user to login and logout.
E.3	**Name:** Schedule **Responsibility:** This module is responsible for the implementing the part of the GUI that handles the scheduling of appliances and devices. **Constraints:** To perform this task it must be done in less than 4 clicks from the homepage. To use this part of the GUI the level of user experience must be a novice.
E.4	**Name:** Phone Configuration **Responsibility:** This module is responsible for implementing the part of the GUI that handles the mobile phone application. **Constraints:** This module shall be available only in normal environment.
E.5	**Name:** Device Setting GUI **Responsibility:** This module is responsible for implementing the part of the GUI that handles the modification of device and appliance settings.
E.6	**Name:** MyDevices **Responsibility:** This module is responsible for implementing the part of the GUI that handles the dashboard view of all the devices available to the user. **Constraints:** This part of the GUI must be less or 3 clicks away from the homepage. Time to display list of devices must be less than take 80ms.
E.7	**Name:** Device Settings **Responsibility:** This module is responsible for implementing the part of the GUI that handles the particular device settings view. Every kind of device available will have its own view to allow the user to change its settings. **Constraints:** Experience needed to develop this part of the GUI can be from an entry level higher.

E.8	**Name:** System Configuration **Responsibility:** This module is responsible for implementing the part of the GUI that handles the changing setting of the whole DH system. **Constraints:** For this part of the system consider developing for a novice user.
E.9	**Name:** Manage Accounts **Responsibility:** This module is responsible for implementing the part of the GUI that handles the adding and deleting accounts from the system.
E.10	**Name:** Register Devices **Responsibility:** This module is responsible for implementing the part of the GUI that handles the adding, deleting, and authenticating devices to the system.
E.11	**Name:** System Controller **Responsibility:** This module is responsible for implementing the part of the DH system that acts as the the system controller.
E.12	**Name:** User **Responsibility:** This module is responsible for implementing the part of the DH system that handles the logic for creating users.
E.13	**Name:** General User **Responsibility:** This module is responsible for implementing the part of the DH system that handles the logic for creating a user of type general.
E.14	**Name:** Master User **Responsibility:** This module is responsible for implementing the part of the DH system that handles the logic for creating a user of type Master.
E.15	**Name:** Technician **Responsibility:** This module is responsible for implementing the part of the DH system that handles the logic for creating a user of type Technician.
E.16	**Name:** Degraded System Controller **Responsibility:** This module is responsible for implementing the part of the DH system that acts as the the backup to system controller that will act when there is a problem with the main controller and when the system is not fully available. **Constraints:** This part of the system must only used 50% of the system resources. To test this part of the system must be complete in 4 months once fully implemented.
E.17	**Name:** Degraded User **Responsibility:** This module is responsible for implementing the part of the DH system that handles the logic for creating degraded users. These users do not have the whole functionality as normal users.
E.18	**Name:** Degraded General User **Responsibility:** This module is responsible for implementing the part of the DH system that handles the logic for creating a user of type general during degraded use.

E.19	**Name:** Degraded Master User **Responsibility:** This module is responsible for implementing the part of the DH system that handles the logic for creating a user of type Master during degraded use.
E.20	**Name:** Degraded Technician User **Responsibility:** This module is responsible for implementing the part of the DH system that handles the logic for creating a user of type Technician during degraded use.
E.21	**Name:** System Restore **Responsibility:** This module is responsible for implementing the part of the DH system that allows the change from a degraded state to a normal state. **Constraints:** This module shall detect when the system can go back to normal environment with 1sec of been available to get into normal environment. Person with 5 years of experience in machine learning should be used to implement this part of the system. Module shall be broken down into at least 10 subcomponents.
E.22	**Name:** Backup Subsystem **Responsibility:** This module is responsible for implementing the part of the DH system that create backups and restores the system from backup.
E. 23	**Name:** Backup **Responsibility:** This module is responsible for implementing the part of the DH system that is responsible to gather the information for a backup, as well as storing the backup information. This module is responsible to store data related to user credentials, devices with their appropriate settings and the planner in the backup database every 30 minutes. Any data that was change within every 30 minutes it should be transferred to the backup database. **Constraints:** This module shall be tested within a 2 weeks after been implemented. This module shall be broken down into smaller components that allows modification to the overall module to be done within a week.
E. 24	**Name:** Backup Restore **Responsibility:** This module is responsible for implementing the part of the DH system that is responsible for restoring a system to a previous state from backup information. **Constraints:** This component should be managed by a Security expert and should be a stable version of the system according to the expert's standards.
E. 25	**Name:** Database System **Responsibility:** This module is responsible for implementing the part of the DH system that is in charge of the storage and retrieval of information to and from the database. **Constraints:** The data going to the database should not be corrupted, if it is then it shall be disregarded. A database expert shall oversee the functionality of the database once a week.
E. 26	**Name:** Database Manager **Responsibility:** This module is responsible for implementing the part of the DH system that is in charge of handling requests to the database. **Constraints:** This module shall broken down to small sections that allows to make modification to be done in less than a week. This module and all the modules inside shall be implemented by a college hire. Every request shall be managed first in first out order following the prioritization set by the database expert.

E. 27	**Name:** Backup_DB **Responsibility:** This module is responsible for implementing the part of the DH system that is in charge of handling backup data requests to the database.
E. 28	**Name:** DataLogger_DB **Responsibility:** This module is responsible for implementing the part of the DH system that is in charge of handling device and application readings data requests to the database. **Constraints:** All the data shall be stored in a chronological fashion and no corrupted data
E. 29	**Name:** SystemSettings_DB **Responsibility:** This module is responsible for implementing the part of the DH system that is in charge of handling system settings data requests to the database.
E. 30	**Name:** Security_DB **Responsibility:** This module is responsible for implementing the part of the DH system that is in charge of handling security data requests to the database.
E. 31	**Name:** Report_DB **Responsibility:** This module is responsible for implementing the part of the DH system that is in charge of handling report data requests to the database.
E. 32	**Name:** Report Creator **Responsibility:** This module is responsible for implementing the part of the DH system that is responsible to create a report from information in the database.
E. 33	**Name:** Report Template Generator **Responsibility:** This module is responsible for creating specialized reports from data gathered from the system. **Constraints:** This module can be implemented by an entry level college hire.
E. 34	**Name:** Security Breach Report **Responsibility:** This module is responsible for creating reports about security breaches that occurred during a certain period of time.
E. 35	**Name:** Downtime Report **Responsibility:** This module is responsible for creating reports about lapses that occurred during a certain period of time.
E. 36	**Name:** Appliance Report **Responsibility:** This module is responsible for creating reports about appliance settings that occurred during a certain period of time.
E. 37	**Name:** Control Units System **Responsibility:** This module is responsible for handling communication with hardware and appliances. **Constraints:** Not applicable at this time
E. 38	**Name:** Device Manager **Responsibility:** This module is responsible for handling and directing traffic that is intended to go to hardware or appliances.

	Constraints: Not applicable at this time
E. 39	**Name:** Appliance Controller **Responsibility:** This module is responsible for the talking directly to appliances. **Constraints:** This module should be implemented in 2 weeks.
E. 40	**Name:** Hardware Controller **Responsibility:** This module is responsible for interfacing directly to simple hardware, such as sensors. **Constraints:** An expert with 5 years in embedded systems must be the lead in the development of this module.
E. 41	**Name:** Data Logger **Responsibility:** This module is responsible for querying directly with the devices regarding their status or settings. This module takes this information and stores it into the database. **Constraints:** This module should be tested with Appliance and Hardware Controllers. This module should take 2 weeks to implement.
E. 42	**Name:** Security **Responsibility:** This module is responsible to implement the code that only allows certain people or devices access the system, as well as the extent of how much each one of these can do. **Constraints:** This module should be completed by a developer with at least 3 years experience in security.
E. 43	**Name:** Authenticator **Responsibility:** This module is responsible for ensuring that the username provided matches records in the database, and that the password matches that that records password. **Constraints:** This module should be completed by a developer with at least 3 years experience in security. This module should be tested extensively to ensure verification.
E. 44	**Name:** Security Manager **Responsibility:** This module is in charge of handling the calls to security components by redirecting calls to the appropriate recipient. **Constraints:** This module must be implemented by same team developing security and authenticator.
E. 45	**Name:** Tool Security **Responsibility:** This module is in charge of implementing the logic to carry out the authorization procedure of devices and appliances into the system. **Constraints:** This module should take 1 intermediate-level developer about 5 days to implement. It should take <1 day to test this code. The developer should be familiar with device authentication protocols.
E. 46	**Name:** Alarm System **Responsibility:** This module is in charge of handling the alarm system logic. This involves engaging and disengaging the alarm, handling triggers from sensors, and turning on sound and/or light alarms when an intrusion is detected. **Constraints:** This module will take about a month to implement by 3 programmers with embedded systems experience.

E. 47	**Name:** Settings **Responsibility:** This module is in charge of handling all the settings in the system. **Constraints:** N/A
E. 48	**Name:** Settings Manager **Responsibility:** This module is in charge of handling the calls to settings components by redirecting calls to the appropriate recipient. **Constraints:** This module will take a week to implement by an intermediate programmer that is familiar with the system. It will take about a day to test.
E. 49	**Name:** Device Settings Scheduler **Responsibility:** This module is in charge of providing the logic to create a schedule for the devices that are connected to the system. This scheduler should be flexible enough to be able to change settings on a variety of components. **Constraints:** It will take 1 month to implement using 5 experienced people. It will take about a week to test. This module should should be able to be modified in less than 2 weeks.
E. 50	**Name:** System Settings **Responsibility:** This module is in charge of handling all system wide settings. **Constraints:** Will take about 3 weeks to implement and 1 week to test fully. This must be implemented by a person that know the system intimately because these settings are crosscutting.
E. 51	**Name:** User Settings **Responsibility:** This module is in charge of handling all request to add, change or delete user related activities such as adding new users, deleting new users, or updating user information. **Constraints:** This module will take 2 weeks two implement and 1 day to test. This should be implemented by a person with at least two years of experience with web systems.

Table 1: Elements and their Properties and Behaviors.

3.1.3 Context Diagram

A context diagram is not applicable for this view because the digital home system is not a submodule of a bigger system, but rather the whole system itself.

3.1.4 Rationale

The modules have been decomposed based on the deployment time of 1 month. This means that any module shown should be able to be implemented and unit tested in one month.

3.2 Uses View

The following view comes from the uses style in which show how modules depend on each other to fulfill their own responsibilities. It's helpful for planning because it helps explain subsets and increments of the product being developed [1].

3.2.1 Primary Presentation

Figure 3.2.1 represent how the uses style is implemented in the DH System.

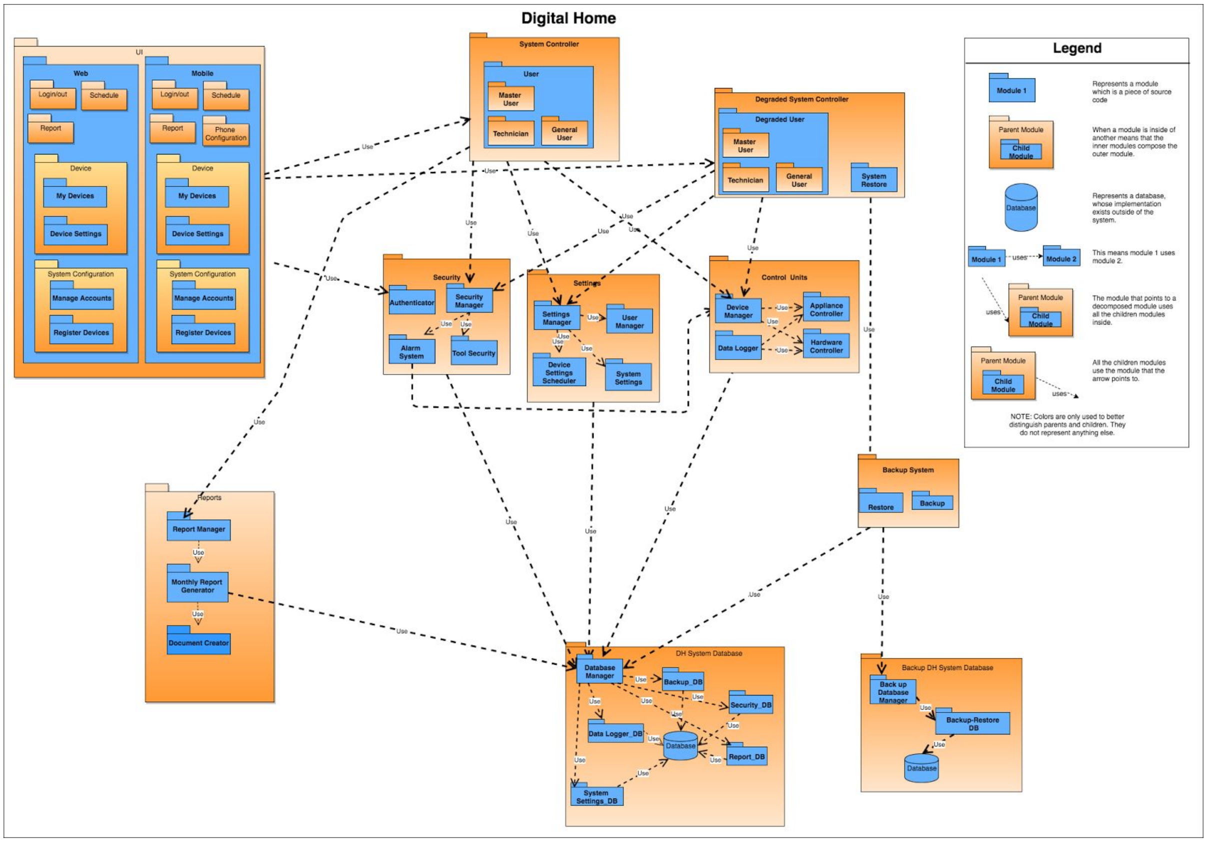

Figure 2: Uses Graphical View of Digital Home System.

3.2.2 Element Catalog

3.2.2.1 Elements and Their Properties

The elements in this view are the same as in the Decomposition View (Section 3.1). If you would like to see the elements and their properties, please refer to Table 1.

3.2.2.2 Relations and Their Properties

Relations and properties of elements can be seen in Section 3.1.

3.2.2.2.1 Elements Interfaces

The following table shows the interface that are considered in the uses views architecture of this product. The table is divided into columns that show the interface, what modules can use that interface, and the properties of that interface. Properties include what information is exchange to accomplish the intended task.

Module	Interface With	Properties (What they share)
Security Manager	System Controller Degraded System Controller	Information regarding the management of alarms or security services. Management include view, turn on/off.
Alarm System	Security Manager	Alarm System would provide the control to the emergency alarms in the system. Used to set or turn off alarms.
Tool Security	Security Manager	Information regarding the management of the different security services . Used to set or turn off security services, view security services that are available and the current status of each service.
Authenticate	UI	Data that contains user credentials would be gathered by the UI and used by the authenticate interface in order to give access to an authorized user or block unauthorized users to access the system.
Settings Manager	System Controller Degraded System Controller	Settings Manager would provide the availability to add/modify/remove users, manage the DH planner and the values that relate to configuration of the system itself.
Device Settings Scheduler	Settings Manager	Information regarding the values and time needed to set various preset home parameters for the DH Planner
User Manager	Settings Manager	Information that is needed to delete, create or delete a user is handle by the User Manager.
System Settings	Settings Manager	Values that would be used to configure the system.

Device Manager	System Controller Degraded System Controller	Information that contains the values to set or control an appliance and hardware.
Appliance Controller	Device Manager	Information that is used to turn on/off or set certain values to an appliance would be handle by the Appliance Controller.
Appliance Controller	Data Logger	Would provide an interface to gathered data values from appliances to be used by the Data Logger
Hardware Controller	Device Manager	Information that is used to turn on/off or set certain values to a hardware component would be handle by the Hardware Controller.
Hardware Controller	Data Logger	Would provide an interface to gathered data values from hardware components to be used by the Data Logger
Backup Restore	System Restore	Data in order to restore to a previous specific state of the system would be provided by the Backup Restore. System Restore would provide the information to determine the degraded current state of the system.
Database Manager	Reports Creator Security Settings Control Units Backup	This interface handles the queries to retrieve or store data from the DH database.
BackUp_DB	Database Manager	This interface would handle the execution of a query related to Backing up a state of the system.
Security_DB	Database Manager	The interface would handle the execution of a query coming from the security module.
Settings_DB	Database Manager	The interface would handle the execution of a query coming from

		the Settings module.
Data Logger_DB	Database Manager	The interface would handle the execution of a query coming from the Data Logger module.
Report_DB	Database Manager	The interface would handle the execution of a query coming from the Report module.
Reports	System Controller	System Controller would provide the constraints require to create the desired report.
Report Template Generator	Report Creator	Report Creator would provide the results obtain from the database to be process by the report template generator according to the desired template.
Backup Database Manager	Restore Backup	Data that needs to be restore to the primary database and any data that needs to be store in the backup database.

Table 2: Elements Interfaces

3.2.3 Context Diagram

The following figure shows the context diagram of the DH System from a uses style perspective.

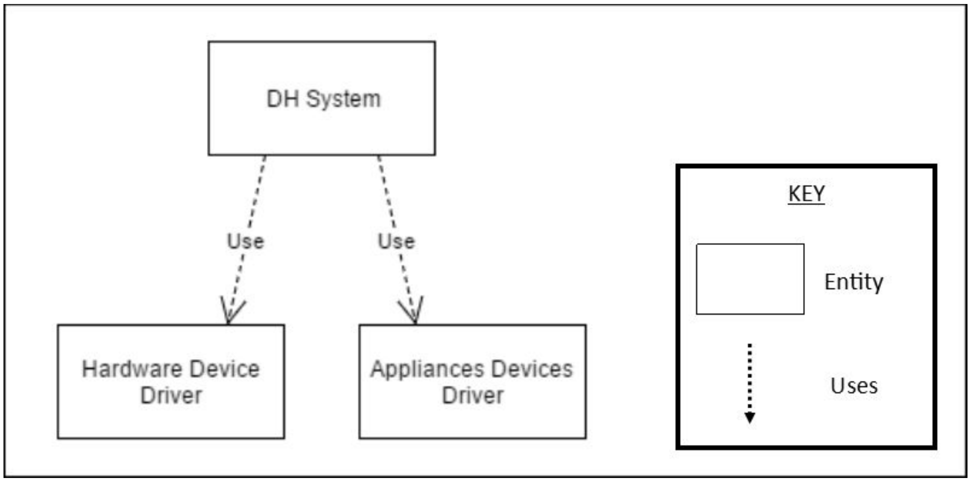

Figure 3: Uses Graphical View of the context of the Digital Home System.

3.2.4 Rationale

Rationale point that is considered for this view is that a module shall be developed within a month. Beside the time of implementation, there wasn’t any other significant design decision taken for this view. This view was largely based off of the Decomposition View in Section 3.1.

3.3 Generalization View

A Generalization View locates specialization relations among modules, and is used to show generalization relationship between modules. A generalization relationship is represented as inheritance, interface implementation, or both. This view captures and presents the commonalities between modules and supports reuse.

3.3.1 Primary Presentation

In section 3.3.1 the Generalization View Primary Presentation contains images of modules with a generalization relationship. Each module presents its commonalities and variations from Figure X to Figure X. These figure diagrams are represented in UML notation which is commonly used to denote Generalization [1].

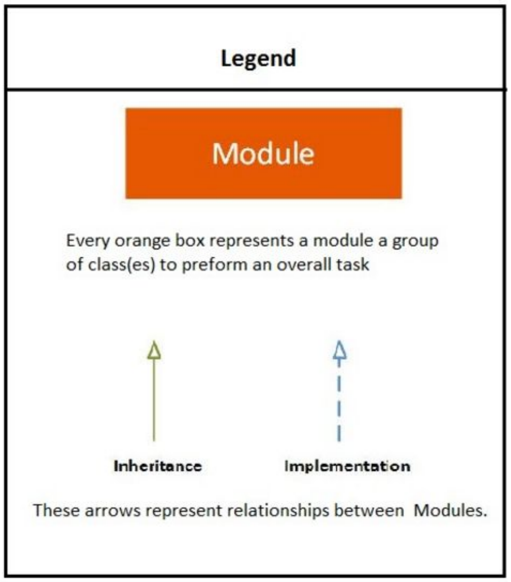
Legend
Module
Every orange box represents a module a group of class(es) to preform an overall task
Inheritance
Implementation
These arrows represent relationships between Modules.

Figure 4: Generalization View Legend Key

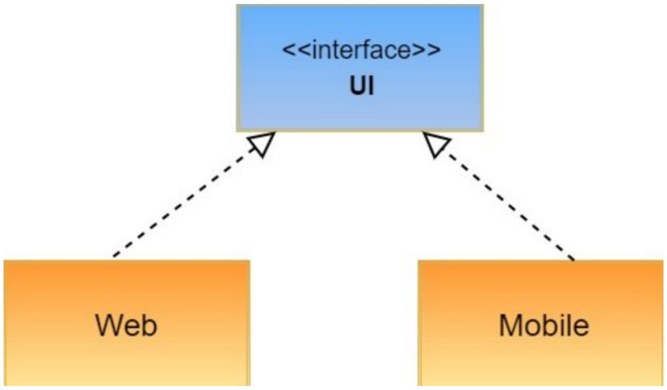

Figure 5: User Interface Generalization View

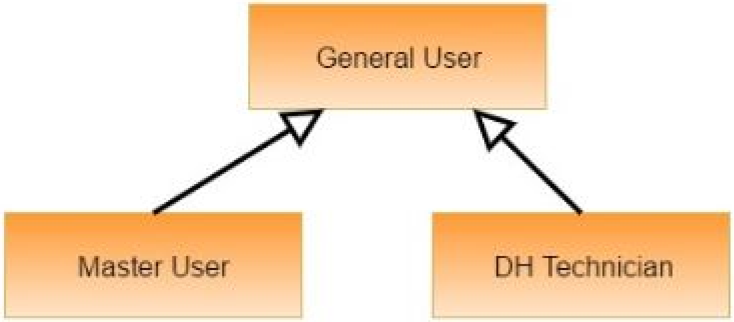

Figure 6: User Generalization View

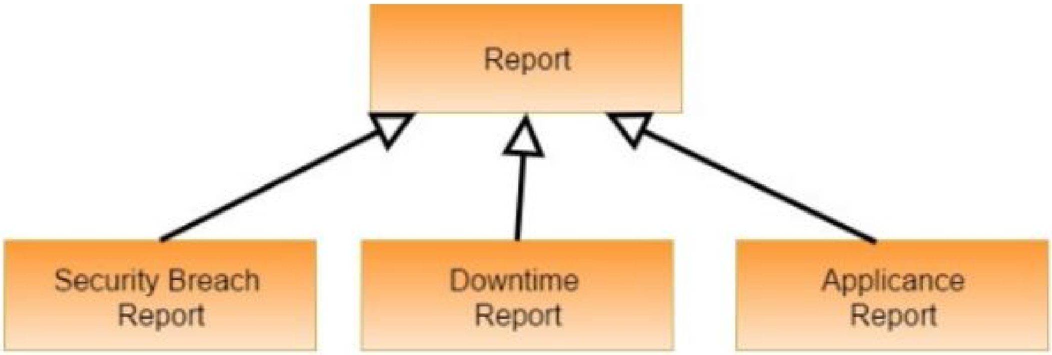

Figure 7: Report Generalization View

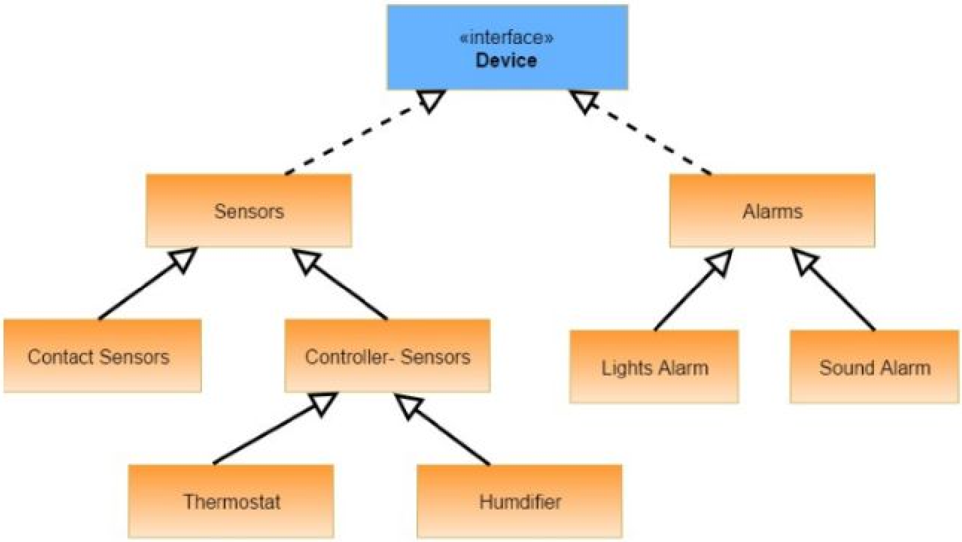

Figure 8: Control Unit Generalization View

3.3.2 Element Catalog

Section 3.3.2 Provides a verbal explanation of what is depicted in Figures X through X. Element Catalog explains and defines the elements shown in the Generalization View and list their properties.
Each element has been given an element number. Properties of an element can include: name, responsibility, visibility of interfaces, implementation information, mapping to source code units, test information, management information, and implementation constraints.

Element	Properties
E.1	**Figure 5** **Name:** User Interface Generalization View **Parent Module:** UI **Children Module:** Web & Mobile **Relation: Interface Shared Data:** Behaviors and constraints submodules should follow **Constraints:** Each User Interface(UI) must follow and provide all UI functionality and services stated in the SRS. A major constraint the Digital Home's UI contains, is that an interface is required to be developed for Web and Mobile. Each approach contains completely different implementation approaches. Preventing duplicate code, maximizing code reuse, and minimizing development time is a high priority. **Responsibility:** **Parent Module (UI)**: Is an interface that is responsible to list out all functionalities the system UI is required to provide to the user. **Child Module (Web)**: Is a type of user interface that is responsible to present the user with a web based graphical representation of the system and functionality declared from the UI parent module. **Child Module (Mobile)**: Is a type of user interface that is responsible to present the user with a mobile based graphical representation of the system and functionality declared from the UI parent module.
E 2	**Figure 6** **Name:** User Generalization View **Parent Module:** General User **Children Module:** Master User & DH Technician **Relation: (**Is -a) Inheritance **Shared Data:** General User properties (operations and fields) **Constraints:** Each user is permitted a specific set of operations and access level, for security purposes to prevent unauthorized access. According to the view this also provides

	modifiability, by allowing in future any new users categories allowed in the system. Every type of user has a baseline of operations and access level from the General User class. **Responsibility:** **Parent Module (General User)**: Parent module that is responsible to provide the general user its set of operations and access level. These properties are stated in the SRS requirements 3.2.1.1 through 3.2.1.3 [2]. **Child Module (Master User)**: Is a type of user that contains all the functions and properties from General User and additional properties that a Master User contains. These additional properties are stated in the SRS requirement 3.2.1.4 & 3.2.2.4 [2]. **Child Module (DH Technician)**: Is a type of user that contains all the functions and properties from General User and additional properties that a Master User contains. These additional properties are stated in the SRS requirements in Section 3.2.2 [2].
E.3	**Figure 7** **Name:** Report Generalization View **Parent Module:** Report **Children Module:** Security Breach Report, Downtime Report, & Appliance Report **Relation: (**Is -a) Inheritance **Shared Data:** General Report Outline **Constraints:** Reports module provides a general statistic of the system's performance and functionality described in the SRS Section 4.6.3 [2]. This system evaluation allows us to see if there are any weak areas that need improvement. Modifiability is possible since any additional type of report can be easily incorporated with Report module as the template. **Responsibility:** **Parent Module (Report):** Provides a general template a report over a month, year, for the past two years. **Child Module (Security Breach Report):** Provide a report of the day and time for which a security breach has been reported. **Child Module (Downtime Report):** Provide a report the periods of time the DH System was not in operation. **Child Module (Appliance Report):** Provide a report of appliance use statistics. Such has the maximum and minimum values of temperature and humidity.
E.4	**Figure 8** **Name:** Control Unit Generalization View

Interfaces: <<Device>>
Modules: Sensor & Alarm
Relation: Implementation
Shared Data: Power switch behavior

Constraints:
All Sensors and alarms must follow the Device interface constraints to be considered a device in the Digital Home System. This provides modifiability to the system by allowing any additional system devices to be incorporated to the DH System. Each device contains common operations and can be easily categorized.

Interfaces: <<Device>> is an interface that enforces all the modules to contain a basic functionality to be considered devices. One commonality all device operations share is a digital power switch or enabler/disabler. Every device shall have the ability to switch state to be turned on or off [2].

Parent Module: Sensors
Children Module: Contact Sensors & Control Sensors
Relation: (Is -a) Inheritance
Shared Data: General Report Outline
Constraints:
Sensor types such as Contact Sensors and Controller-Sensors must contain basic abilities to be considered Sensors.

Responsibilities:
Parent Module (Sensors): Provides a general operations a sensor shall contain such as read the sounding area.

Child Module (Contact Sensors): Is a type of sensor that keeps track of house entry. A magnetic contact sensor can detect if a window or door state is open or closed.

Child- Parent Module (Controller-Sensors): Is a type of sensor that also contains a control unit. It has all the functionalities a sensor has (parent class) and can be modified with its controller by containing a set point state [2].

Child-Parent Module: Controller- Sensors
Children Module: Thermostat & Humidifier
Relation: (Is -a) Inheritance
Shared Data: General Report Outline

Constraints:

	Control - Sensor types such as Thermostat and Humidifier systems must contain basic abilities to be considered Controller-Sensors. **Responsibilities:** **Child- Parent Module (Controller-Sensors):** A type of sensor that also contains a control unit. It has all the functionalities a sensor has (parent class) and can be modified with its controller by containing a set point state [2]. **Child Module (Thermostat):** A type of controller-sensor that controls the temperature of a room. **Child Module (Humidifier):** A type of controller-sensor that controls the humidity of a room. **Parent Module:** Alarms **Children Module:** Lights Alarm & Sounds Alarm **Relation: (**Is -a) Inheritance **Shared Data:** General Report Outline **Constraints:** Alarm types such as light and sound alarms must contain basic abilities to be considered an Alarm. **Responsibility:** **Parent Module (Alarms):** Notifies the owner a state has occurred. . **Child Module (Lights Alarm):** A type of alarm that notifies the owner with lights. **Child Module (Sound Alarm):** A type of alarm that notifies the owner with sound.

Table 3: Generalization Elements and their responsibilities.

3.3.3 Context Diagram

Not applicable because in all of the figures 5 through 8 they are just displaying relations among the modules. Implementing these figures in AADL would contain a system with an empty system implementation . Since the purpose of the view is to just show the communality among modules.

3.3.4 Rationale

The reason why we want to generalize common behaviors is to minimize redundant code. So grouping common behaviors will save time in development and testing. For future development the architecture components are organized in such a way where additional features will be easy to implement based on the grouping of common behaviors.

3.4 Layered View

The layered style puts together layers in a unidirectional allowed-to-use relation with each other and contains the layer segments that interact with the rest of the layers.

3.4.1 Primary Presentation

The following diagram shows the different layers of the system as well as their interaction.

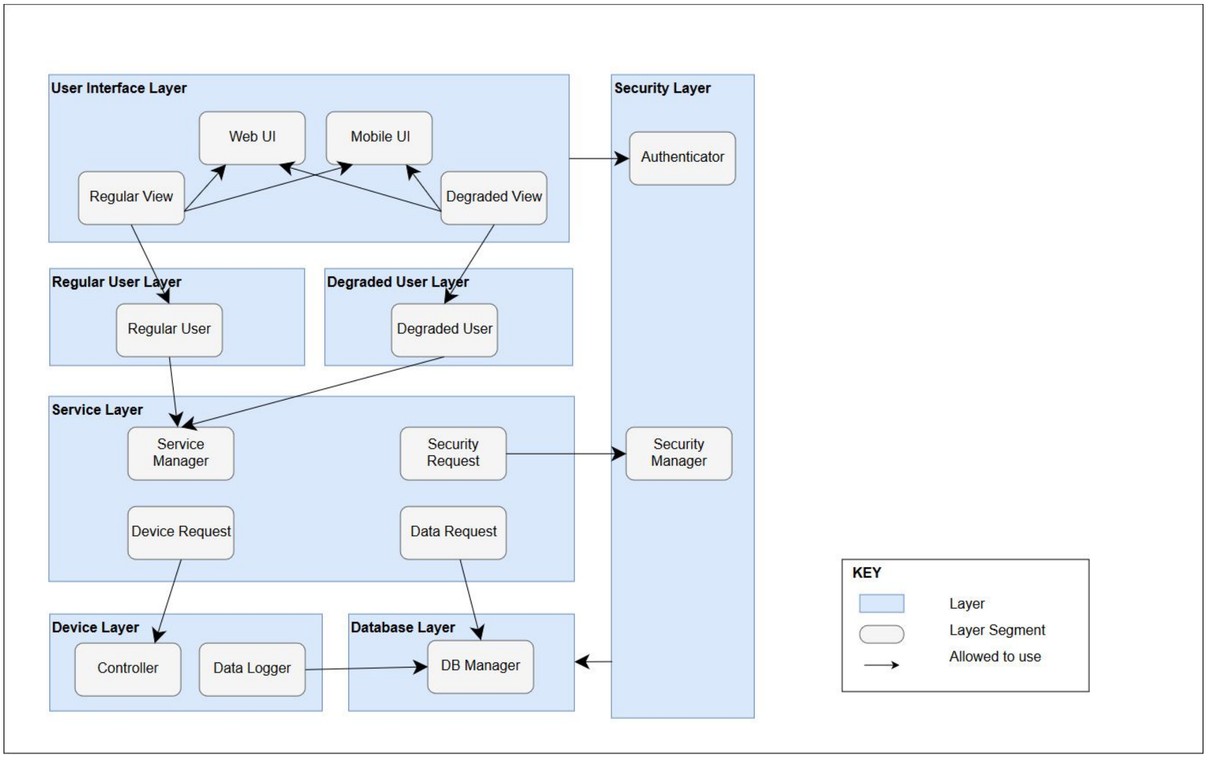

Figure 9: Primary representation of Layer View

3.4.2 Element Catalog

This section we would specify any new elements along with their corresponse information. Elements that are shown in the layered view primary presentation and are not shown under section 3.4.2.1 Elements, Properties and their Behaviors.

3.4.2.1 Elements Their Properties and Behaviors

The following section will have a list of the elements in the Layered style as well as their properties and behaviors.

Element	Details
L.1	**Name:** User Interface Layer **Properties:** This layer was decided to kept separated for the purpose of Modifiability, when future updates occur the Design Team will only have to deal with this layer to add or remove visual aspects of the system. **Responsibility**: The User Interface Layer will be responsible for interfacing with the user and showing him/her their respective functionality in an organized manner. **Layer Component:** 1. Regular View[*] 2. Degraded View[*] 3. Web UI a. Login in/out b. Schedule c. Report d. Device i. MyDevices ii. Device Settings e. System Configuration i. Manage Account ii. Register Account 4. Mobile UI a. Login in/out b. Schedule c. Report d. Phone Configuration e. Device i. MyDevices ii. Device Settings f. System Configuration i. Manage Account ii. Register Account
L.2	**Name:** Regular User Layer **Properties:** This layer was kept separated for the purpose of Modifiability, when the Design Team wants to introduce a new user type. **Responsibility**: The Regular User Layer will be in charge of handing out the agreed upon permissions to the regular users (General Users, Master User and DHTechnician) **Layer Components:** 1. Regular User[*] 2. System Controller a. User i. Master User ii. General User iii. Technician
L.3	**Name:** Degraded User Layer **Properties:** This layer was kept separated for the purpose of Modifiability, when the Design

	Team wants to introduce a new degraded user type. **Responsibility:** The Degraded User Layer will be in charge of handing out the agreed upon permissions to the regular users (Degraded General Users, Degraded Master User and Degraded DHTechnician) **Layer Components:** 3. Degraded User[*] 4. System Controller a. User i. Master User ii. General User iii. Technician 5. System Restore
L.4	**Name:** Service Layer **Properties:** This layer was kept separated for the purpose of modifiability. For example, when the design team decides to introduce new services. **Responsibility:** The Service Layer will be responsible for providing the required services (giving the user the monthly report and letting the user edit the settings of the devices) **Layer Components:** 1. Device Request[*] 2. Security Request[*] 3. Data Request[*] 4. Report a. Report Creator b. Report Template Generator i. Security Breach Report ii. Downtime Report iii. Appliance Report 5. Settings a. Manage Settings b. Security Manager c. Alarm System d. Tool Security 6. Control Unit a. Device Manager
L.5	**Name:** Device Layer **Properties:** This layer was kept separated for the purpose of modifiability. For example, when the design team decides to introduce new devices. **Responsibility:** The Device Layer will will be responsible for all the physical aspects of the system from sensors, and thermostats to household items such as televisions and lights. **Layer Components:** 1. Control Unit a. Appliance Controller b. Hardware Controller c. Data Logger[*] 2. Controller

L.6	**Name:** Data Layer **Properties:** This layer was kept separated for the purpose of modifiability. For example, when the design team decides to introduce new databases components. **Responsibility:** The Data Layer will be responsible for providing access to the database to the upper layers. **Layer Components:** 1. Database a. Database Manager b. Data Logger_DB c. System Settings_DB d. Security_DB e. Report_DB f. Database (Physical)
L.7	**Name:** Security Layer **Properties:** This layer was kept separated for the purpose of modifiability. For example, when the design team decides to introduce or update new security services.. **Responsibility:** The Security Layer will be responsible in keeping up with current protections of the system such as encrypting and authenticating users. **Layer Components:** 1. Security a. Authenticator b. Security Manager c. Alarm System d. Tool Security 2. Back up a. Backup restore b. Backup

Table 4: Layers and their responsibilities.

[*] These accessors will be explained in detail in section 3.4.2.1 Elements Interfaces

3.4.2.1.1 Elements Interfaces

The Security interfaces with the rest of the layers. The UI Layer only interfaces with the Communication Center Layer. The Communication Center Layer interfaces with the service Layer. The Service Layer interfaces with the Device and hardware Layer.

Layer	**Interfaces**
L.1	**Name:** User Interface Layer **Interfaces:** The User Interface Layer has access to the Regular User Layer and the Degraded User Layer through the regular and degraded view subcomponent respectively. It also has access to the Security Layer when authenticating the users. **Accessors**: 1. Regular View subcomponent gets the available service requests (i.e. available

	buttons) from the Regular User Layer. 2. Degraded View subcomponent gets the available service requests from the Degraded User Layer. 3. The User Interface layer gets its security aspects from the Authenticator subcomponent in the Security layer.
L.2	**Name:** Regular User Layer **Interfaces**: The Regular User Layer has access to the Service Layer through the Regular User subcomponent. It also has access to the Security Layer through the higher layers. **Accessors:** 1. The Regular User subcomponent gets the logic for the available services from the Service Layer.
L.3	**Name:** Degraded User Layer **Interfaces**: The Degraded User Layer has access to the Service Layer through the Degraded User. It also has access to the Security Layer through the higher layers. **Accessors:** 2. The Degraded User subcomponent gets the logic for the available services from the Service Layer.
L.4	**Name:** Service Layer **Interfaces:** The Service Layer is allowed to use the Device Layer though the Device Request subcomponent, and the Data Layer through the Data Request subcomponent. It is also allowed to use the Security Layer though the Security Request. **Accessors:** 3. The Device Request subcomponent gets control to the devices from the Device Layer. 4. The Data Request subcomponent gets access to the database from the Data Layer. 5. The Security Request subcomponent gets encryption and decryption from the Service Layer
L.5	**Name:** Device Layer **Interfaces:** The Device Layer is allowed to use the Data Layer through its Data Logger functionality. It also has access to the Security Layer through upper layers. **Accessors:** 1. The Data Logger subcomponent gets access to the database from the Data Layer.
L.6	**Name:** Data Layer **Interfaces:** This layer provides access to the database to the upper layers as well as the Device Layer.
L.7	**Name:** Security Layer **Interfaces**: This layer interfaces with the Data Layer by accessing it to store persistent data. This persistent data includes the user’s password and username. It also provide security services to the rest of the layers.

Table 5: Layers and their interfaces.

3.4.3 Context Diagram

The primary function of a context diagram is to show what’s in and what’s out of the system (or part of the system) that we're considering at the moment.

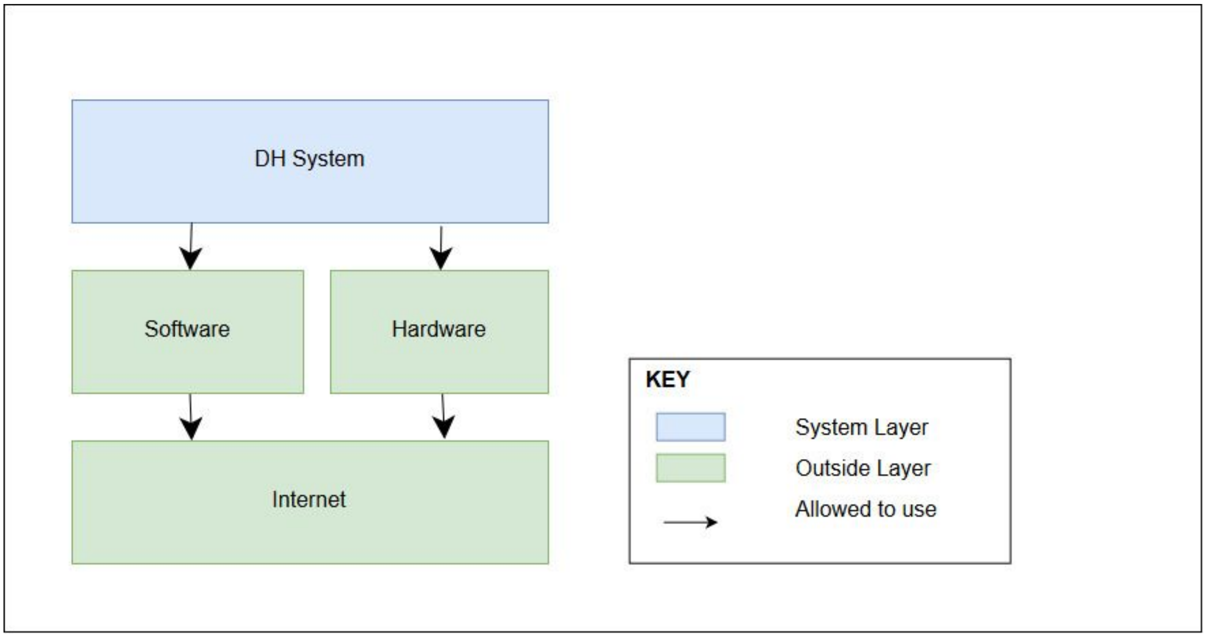

Figure 10: Layer Context Diagram

The following table describes the diagram above.

Element	Details
E.1	**Name:** DH System **Responsibility**: The system should satisfy all the requirements from the System Requirement Specifications[2]
E.2	**Name:** Software Layer **Responsibility:** This layer will deal with the Data base services (i.e. Oracle Corporation)
E.3	**Name:** Hardware Layer **Responsibility**: This layer will deal with all the physical devices that will be connected to the system (i.e. sensors, thermostats etc.)
E.4	**Name:** Internet Layer **Responsibility:** This layer will be in charge of connecting all the upper layers.

Table 6: Context diagram elements and their responsibilities.

3.4.4 Rationale

The team decided to separate the components that deal with displaying the provided services into a layer for modifiability and to assign a team with expertise in user interfaces. This layer is called the User Interface Layer.

We also decided to separate the collection of the known services in two layers, the Regular User Layer and the Degraded User Layer. The Regular User Layer will contain all the services that a regular user would normally have of the system. The Degraded User Layer will contain all the services that a user, that has been degraded, would have of the system. This was also decided for future upgrades, if a new functionality is introduced.

The logic behind the provided services was decided to kept separated for the purpose of modifiability and to assign a team specialized in the decided programming language that is going to be used to favor performance. This collection is the Service Layer.

The collection of the different devices that are available and programmable will be kept separated for the purpose of modifiability. When new devices are introduced or the software for this devices is updated, the development team will have to edit only this layer. The Device Layer.

The services from the database were decide to kept separated for the purpose of modifiability. When the development team decides to change database services, they will have to edit only this layer. The Database Layer.

The security services that are provided (i.e. encryption and decryption) will be kept separated to assign a team specialised in security to enhance the system security aspects.

3.5 Data Model View

TBD

4 Component and Connector Views

Section 4 comprises of modules that make up Home System. However Section 4 looks in these modules to find their components and connectors at runtime. Section 4.1 Pipe and Filter, Section 4.2 Batch Sequential , Section 4.3 Client/Server, Section 4.4 Publish Subscribe, Section 4.5 Service Oriented, Section Service Oriented , and finally Section 4.6 Shared Repository View.

Note: The tier property refers to the grouping components into processes:

- Tier 1 - Main System
- Tier 2 - Devices Subsystem
- Tier 3 - Database Subsystem
- Tier 4 - Restore Database

4.1 Pipe and Filter

4.1.1 Primary Presentation
4.1.2 Element Catalog
4.1.2.1 Elements and Their Properties
4.1.2.2 Relations and Their Properties
4.1.2.3 Elements Interfaces
4.1.2.4 Elements Behaviors
4.1.3 Context Diagram
4.1.4 Rationale

4.2 Batch Sequential View

This section contains the system as viewed in the Batch Sequential style.

4.2.1 Report Primary Presentation

Section 4.1.1 provides a visualization of the Report module at runtime, specifically under the Batch Sequential View.

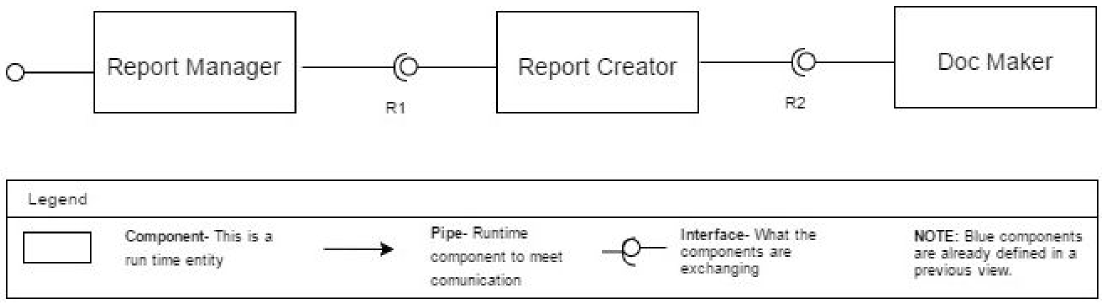
Report Manager
R1
Report Creator
R2
Doc Maker
Legend
Component- This is a run time entity
Pipe- Runtime component to meet comunication
Interface- What the components are exchanging
NOTE: Blue components are already defined in a previous view.

Figure 11: Report Batch Sequential View

4.2.1.1 Report Element Catalog

Section 4.2.1.1 contains a list of each component and connections found in Figure 11 with their list of properties. This section aids the reader in verbal understanding;
how components in Report work at runtime.

Components and Their Properties

Element	Properties
1	**Name:** Report Manager **Reliability:** 0.2% likelihood of failure. **Performance:** Latency <= 50ms. Response time <= 5ms **Functionality:** Report Manager handles all communication for the reporting functionality of the system. The lollipop in front of Report Manager represents all the outside (from different subsystems) to the reporting subsystem. The report manager communicates to the Report Creator to produce the requested report. . **Interface:** This component has the following interfaces: The following calls are redirected to Report Manager + createReport(reportType: int, extension:int):void; **Concurrency:** No concurrency in this component **Tier:** Report **Multiplicity:** Single instance of this component at a time. **Liveliness:** This component starts once a request to create a report is made

2	**Name:** Report Creator **Reliability:** 0.2 % likelihood of failure. **Performance:** Latency <= 50 ms. Response Time <= 10 ms **Functionality:** Makes calls to obtain information to create the report from the database by requesting from the database all alarm diffusions, system outages, and maximum, minimum, and average temperatures and humidity levels for a certain month and year. Once the data is obtained, it parses the data in order to create an XML file from the information, then makes calls to make the XML file into a report document. Finally, the document location is returned to be shown to the user. **Interface:** This component has the following interfaces: The following calls are redirected to Report Creator + createReport(reportType: int, extension:int, resultSet:Report):XML; **Concurrency:** No concurrency for this component. **Tier:** Report **Multiplicity:** There is only a single instance of this component at a time. **Liveliness:** This component starts when a report result set is available from the database. The result set contains all of the report information.
3	**Name:** Doc Maker **Reliability:** 5% likelihood of failure **Performance:** Latency <= 15ms **Functionality:** With the given information that is stored into an XML file. It would gathered the values and format them into a document. . **Interface:** This component has the following interfaces: The following calls are redirected to Doc Maker + createDocument(extension:int , report:XML):Doc; **Concurrency:** No concurrency for this component. **Tier:** Report **Multiplicity:** Single instance of this component at a time. **Liveliness:** This component is created once Report Creator has completed a XML file.

Table 7: Reports Batch Sequential Components and Properties

Connectors and Their Properties

Element	Properties
R1	**Name:** R1 **Data:** ● reportType- The type of report that will be created, which is represented by an integer ● extension - The type of extension the document will be formated in. Which is represented by an integer.

	Size: 9 bytes <= size <= 32 bytes **Reliability:** 0.3% possibility of failure **Performance:** 0.5 ms <= Latency <= 1 ms 9 bytes <= Load <= 32 bytes 0.4 ms <= Response time <= 1 ms **Functionality:** Relay the information needed to create a report. **Concurrency:** N/A **Multiplicity:** Only one instance at a time.
R2	**Name:** R2 **Data:** ● report- The XML version of result set to be formated to the extension type ● extension - The extension of the document to be formatted in. **Size:** 9 bytes <= size <= 32 bytes **Reliability:** 0.3% possibility of failure **Performance:** 0.5 ms <= Latency <= 1 ms 9 bytes <= Load <= 32 bytes 0.4 ms <= Response time <= 1 ms **Functionality:** Relay the information need to create a document. **Concurrency:** N/A **Multiplicity:** Only one instance at a time.

Table 8: Reports Batch Sequential Connectors and Properties

4.2.1.2 Report Context Diagram

Section 4.1.3 Provides the overall view of Report at runtime labeled Figure 12.

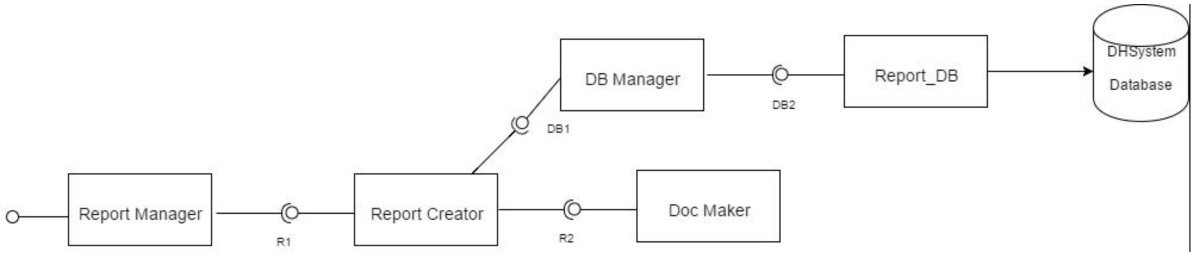

Figure 12: Report Context Diagram

4.2.1.3 Rationale

In Report Creator since we are not expecting the request to return a result to Report Creator component. Creating a document would become batch sequential.

4.2.2 Security Presentations

This view will have all the bach-sequential style flows that were used in the Security components of the system.

4.2.2.1 Security Primary Presentation: 1

This section provides a visualization of the Security module at runtime, specifically under the Batch Sequential View.

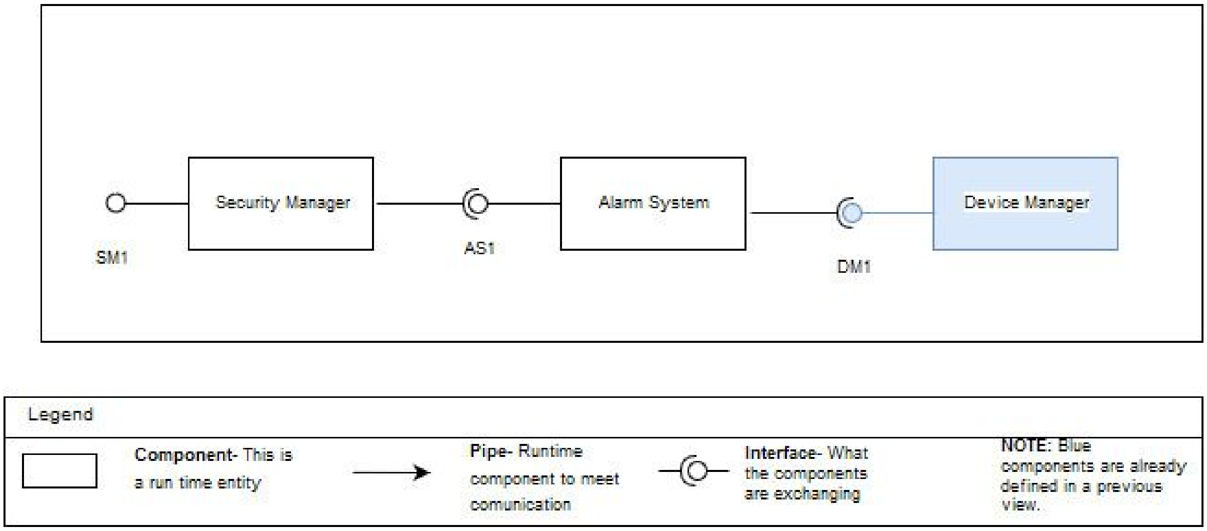
SM1
Security Manager
AS1
Alarm System
DM1
Device Manager
Legend
Component- This is a run time entity
Pipe- Runtime component to meet comunication
Interface- What the components are exchanging
NOTE: Blue components are already defined in a previous view.

Figure 13: Security Batch Sequential View

4.2.2.1.1 Security Element Catalog: 1

This section contains a list of each component and connections found in Figure 13 with their list of properties. It also aids the reader a verbal understanding, how components in Security work at runtime.

Components Catalog

Components	Properties
1	**Name:** Security Manager **Tier:** 1 **Reliability:** 0.99 % likelihood of failure. **Performance:** Latency <= 59ms-1min Response Time <= 10 ms **Functionality:** This module is in charge of handling the calls to security components by redirecting calls to the appropriate recipient. **Interface:** + setInstructions(sound :int , light:int, time:int):String **Liveliness:** Maintain the component live for the whole life of the system.
2	**Name:** Alarm System

	Tier: 1 **Reliability:** 0.99 % likelihood of failure. **Performance:** Latency <= 59ms-1min Response Time <= 10 ms **Functionality:** This module is in charge of handling the alarm system logic. This involves engaging and disengaging the alarm, handling triggers from sensors, and turning on sound and/or light alarms when an intrusion is detected. **Interface:** + setInstructions(sound :int , light:int, time:int):String **Liveliness:** Maintain the component live for the whole life of the system.
3	**Name:** Device Manager **Tier:** Device For properties on this component, refer to Section 4.6

Table 10: Security 1 Batch Sequential Components and Properties

Connectors Catalog

Connector	Properties
SM1	**Name:** SM1 **Data:** Strings **Reliability:** 0.05 % likelihood of failure. **Performance:** Latency <= 5ms Load <= 8 bytes **Multiplicity**: Only one instance at a time.
AS1	**Name:** AS1 **Data:** Strings **Reliability:** 0.05 % likelihood of failure. **Performance:** Latency <= 5ms Load <= 8 bytes **Multiplicity**: Only one instance at a time.
DM1	**Name:** DM1 For properties on this connector, refer to Section 4.6

Table 11: Security 1 Batch Sequential Connectors and Properties

4.2.2.2 Security Primary Presentation: 2

This section provides another visualization view of the Security module at runtime under Batch Sequential . Labeled as Figure 14.

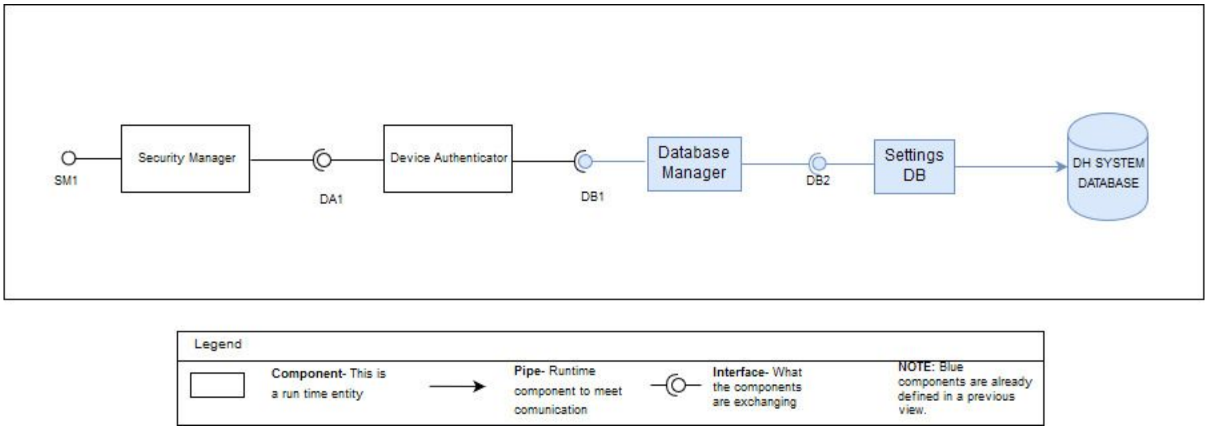
SM1
Security Manager
DA1
Device Authenticator
DB1
Database Manager
DB2
Settings DB
DH SYSTEM DATABASE
Legend
Component- This is a run time entity
Pipe- Runtime component to meet comunication
Interface- What the components are exchanging
NOTE: Blue components are already defined in a previous view.

Figure 14: Security Batch Sequential Device Authenticator View

4.2.2.2.1 Security Element Catalog: 2

This section contains a list of each component and connections found in Figure 14 with their list of properties. This section aids the reader a verbal understanding, how components in Security Manager requesting Device Authenticator at runtime.

Components and their Properties

Components	Properties
1	**Name:** Security Manager For properties on this component, refer to Section 4.2.1.1 component 1
2	**Name:** Device Authenticator **Tier:** 1 **Reliability:** 0.99 % likelihood of failure. **Performance:** Latency <= 59ms-1min Response Time <= 10 ms **Functionality:** This module is in charge of implementing the logic to carry out the authorization procedure of devices and appliances into the system **Interface:** + manage(instruction:String): String **Liveliness:** Maintain the component live for the whole life of the system.
3	**Name:** Database Manager For properties on this component, refer to Section 4.6

Table 12: Security 2 Batch Sequential Components and Properties

Connections and their Properties

Connector	Properties
SM1	**Name:** SM1 For properties on this Connector, refer to Section 4.2.2.1.1 connector SM1
DA1	**Name:** DA1 **Data:** Strings **Reliability:** 0.05 % likelihood of failure. **Performance:** Latency <= 5ms Load <= 8 bytes **Multiplicity**: Only one instance at a time.
DB1	**Name:** DB1 For properties on this Connector, refer to Section 4.6

Table 13: Security 2 Batch Sequential Connectors and Properties

4.2.2.3 Security Primary Presentation: 3

This section holds the Authenticator flow in the Batch-sequential style.

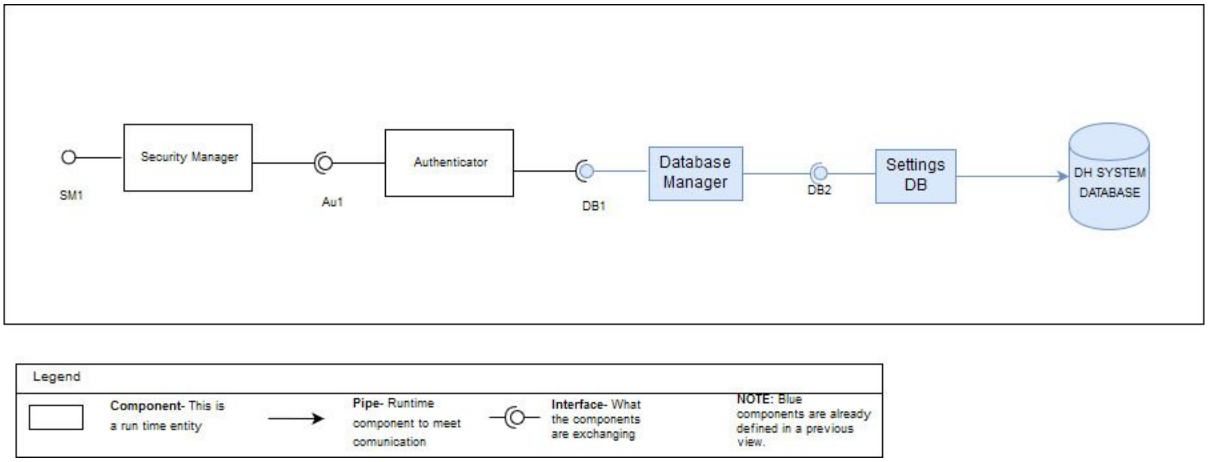
SM1
Security Manager
Au1
Authenticator
DB1
Database Manager
DB2
Settings DB
DH SYSTEM DATABASE
Legend
Component- This is a run time entity
Pipe- Runtime component to meet comunication
Interface- What the components are exchanging
NOTE: Blue components are already defined in a previous view.

Figure 16: Security Batch Sequential User Authenticator View

4.2.2.3.1 Element Catalog: 3

This section holds the elements that follow the Authenticator flow.

Components and their properties

Components	Properties
1	**Name:** Security Manager For properties on this component, refer to Section 4.2.1.1 component 1
2	**Name:** Authenticator **Tier:** 1 **Reliability:** 0.99 % likelihood of failure. **Performance:** Latency <= 59ms-1min Response Time <= 10 ms **Functionality:** **Interface:** + setToolSecurity(newTool: Tool): String **Liveliness:** Maintain the component live for the whole life of the system.
3	**Name:** Database Manager For properties on this component, refer to Section 4.6

Table 14 Security 3 Batch Sequential Components and Properties

Connectors and their properties

Connector	Properties
1	**Name:** SM1 For properties on this Connector, refer to Section 4.2.2.1.1 connector SM1
2	**Name:** Au1 **Data:** String **Reliability:** 0.05 % likelihood of failure. **Performance:** Latency <= 5ms Load <= 8 bytes **Multiplicity**: Only one instance at a time.
3	**Name:** DB1 For properties on this Connector, refer to Section 4.6

Table 15: Security 3 Batch Sequential Connectors and Properties

4.2.5 Backup Primary Presentation

Section 4.2.4 provides the components and connectors that are inside of Backup DB System and DH Database that come to existence with the backup and restore components.

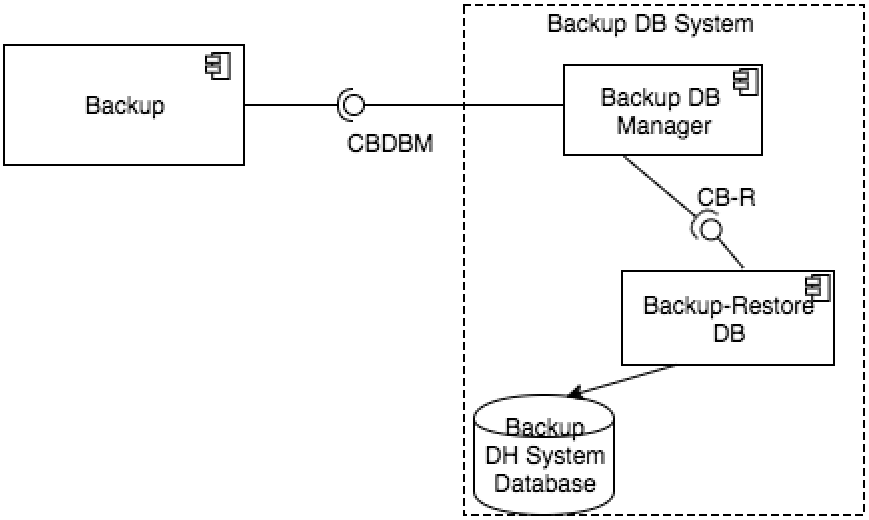

Figure 17: Backup Batch Sequential View

4.2.5.1 Element Catalog

Section 4.2.5.1 contains description and the properties of each of the components and connectors that were identified from the system based on the Batch Sequential style.

Components	Properties
1	**Name:** Backup **Reliability:** 0.99 % likelihood of failure. **Performance:** Latency <= 59ms-1min Response Time <= 10 ms **Functionality:** The purpose of this component is to request information from the main DH System Database and send it out to the backup database. This component would be run every 30min. Backup would be requesting information regarding; User's credentials, values that are set for the devices in the DH system, and information related to planner. **Interface:** This component would not be providing interfaces to anyone since it's role is to execute commands to other components. **Liveliness:** Would be running every 30 minutes and shall be terminated after sending the last data that is in the ResultSet obtain from the DH Database. The data from the ResultSet is sent to Backup DB Manager.

2	**Name:** Backup DB Manager **Reliability:** 0.05 % likelihood of failure. **Performance:** Latency <= 25ms-35ms Response Time <= 10ms **Functionality:** The purpose is to delegate the task the appropriate components that would be handling the request. Would take the JSON string that contains the full data obtained from the Resultset captured back the backup component, and break it into smaller JSON objects that contained a single row of data that would be inserted to the appropriate data table. **Interface:** + getBackup(): ResultSet - Provided to “Restore” component. + insertRestoreInfo(restoreJson: JSON): void - Provided to “Backup” component. **Liveliness:** Shall be initiated only when is invoked by “Restore” or “Backup” and be terminated when it provides the resultSet or when the last data of the restoreJson is set out.
3	**Name:** Backup-Restore DB **Reliability:** 0.05 % likelihood of failure. **Performance:** Latency <= 15ms-20ms Response Time <= 10ms **Functionality:** The purpose is to execute the appropriate queries that would be used to retrieve and insert information to the Backup DH System Database. **Interface:** + getBackup(): ResultSet - Provided to “Restore” component. + insertRestoreUserInfo(UserJson: JSON): void - Provided to “Backup DB Manager” component. + insertRestoreDeviceInfo(DeviceJson: JSON): void - Provided to “Backup DB Manager” component. + insertRestorePlannerInfo(PlannerJson: JSON): void - Provided to “Backup DB Manager” component. **Liveliness:** Shall be initiated only when is invoked by “Restore” or “Backup” and be terminated when it provides the resultSet or when the last data of the restoreJson is set out.

Table 16: Backup Batch Sequential Components and Properties

Connector	**Properties**
1	**Name:** CBDBM **Data:** ResultSet restoredata, JSON backupdata **Reliability:** 0.10 % likelihood of failure. **Performance:** Latency <= 5ms Load <= 16bytes **Multiplicity**: Only one instance at a time
2	**Name:** CB-R **Data:** Timestamp startTimeStamp, Timestamp endTimeStamp, and JSON restore.

	Reliability: 0.5 % likelihood of failure. **Performance:** Latency <= 5ms Load <= 1Kb **Multiplicity**: Only one instance at a time

Table 17: Backup Batch Sequential Connectors and Properties

4.2.6 Restore Primary Presentation

This section provides the sequential running components in order to perform a restore procedure.

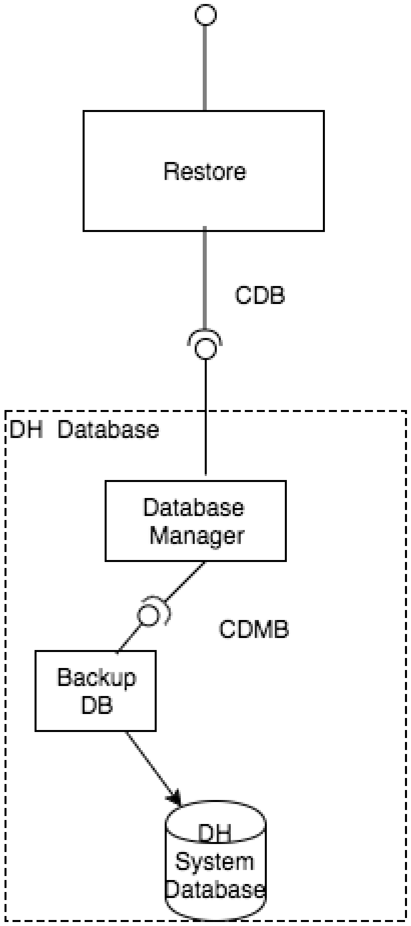

Figure 18: Backup Batch Sequential View

4.2.6.1 Element Catalog

Section 4.2.6.1 contains description and the properties of each of the components and connectors that were identified from the system based on the Batch Sequential style.

Element	Properties
1	**Name:** Restore

	Reliability: 0.99 % likelihood of failure. **Performance:** Latency <= 1min-3min. Response Time <= 10 ms **Functionality:** The main purpose of this component is to retrieve information that is stored in the Backup DH System Database and store it in the primary database whenever the system is available to move from degraded mode to regular mode. **Interface:** + initiateSystemRestore(): - Provided to “Degraded System” component. **Liveliness:** This component is initiated by the degraded subsystem and shall be terminated after sending the last data that is in the ResultSet obtain from the Backup DB System. The data from the ResultSet is sent to DH Database.
3	**Name:** Database Manager For properties on this component, refer to section 4.6
3	**Name:** Backup DB **Reliability:** 0.99 % likelihood of failure. **Performance:** Latency <= 5ms-8ms. Response Time <= 5 ms **Functionality:** The purpose of this component is that it would retrieve information regarding user’s credentials and values that are set for the devices in the DH system. This component is responsible to restore the main database using information from the backup database when a fault occurs to the data. **Interface:** + getBackup(startTimeStamp: TimeStamp, endTimeStamp): ResultSet - Provided to “Backup” component. + insertRestoreUserInfo(UserJson: JSON): void - Provided to “Restore” component. + insertRestoreDeviceInfo(DeviceJson: JSON): void - Provided to “Restore” component. + insertRestorePlannerInfo(PlannerJson: JSON): void - Provided to “Restore” component. **Liveliness:** Shall run only when is invoked by Database Manager. Shall be terminated and break the connection after returning a resultset to Backup component.

Table 18: Restore Batch Sequential Components and Properties

Connector	**Properties**
	Name: CDB **Data:** Timestamp startTimeStamp, Timestamp endTimeStamp, and JSON restore. **Reliability:** 0.05 % likelihood of failure. **Performance:** Latency <= 5ms Load <= 8 bytes **Multiplicity**: Only one instance at a time
	Name: CDMB **Data:** Timestamp startTimeStamp, Timestamp endTimeStamp, and JSON restore. **Reliability:** 0.05 % likelihood of failure. **Performance:**

	Latency <= 5ms Load <= 1Kb **Multiplicity**: Only one instance at a time

Table 19: Restore Batch Sequential Connectors and Properties

4.2.7 Devices Primary Presentation

Section 4.2.7 provides a visualization of Devices at runtime under the Batch Sequential View Figure 9.

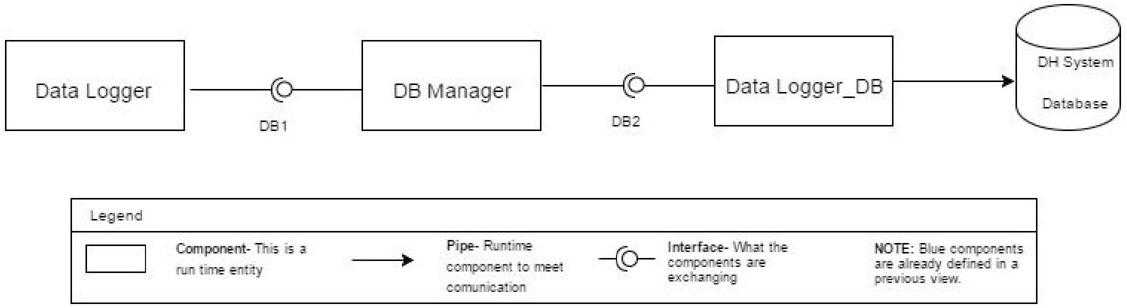
Data Logger
DB1
DB Manager
DB2
Data Logger_DB
DH System
Database
Legend
Component- This is a run time entity
Pipe- Runtime component to meet comunication
Interface- What the components are exchanging
NOTE: Blue components are already defined in a previous view.

Figure 19: Device Batch Sequential View

4.2.7.1 Device Element Catalog

Section 4.2.7.1 contains description and the properties of each of the components and connectors that were identified from the system based on Figure 19.

Element	Properties
1	**Name:** Data Logger **Reliability:** 0.7 % likelihood of failure. **Performance:** Latency: 30 ms - 50 ms. Response Time: 2 ms - 6 ms. **Functionality:** Logs current data from device drivers and writes its recordings into the DH Database.This information will be used to generate reports about the system. **Interface:** This component has the following interfaces: The following calls are redirected to DataLogger: + store_dataLog(device_ID: int, date:String, deviceData: String): Boolean **Concurrency:** N/A **Tier:** 1 **Multiplicity:** There is only one instance of these at a time. **Liveliness:** This component starts every 30 minutes or when a device has been sent a request. Ends once Data Logger has sent the query to DB Manager.
2	**Name:** DB Manager For properties on this component, refer to Section 4.6
3	**Name:** Data Logger_DB For properties on this component, refer to Section 4.6

Table 20: Devices Batch Sequential Components and Properties

Connector	Properties
DB1	**Name: DB1** For properties on this component, refer to Section 4.6
DB2	**Name: DB2** For properties on this component, refer to Section 4.6

Table 21: Device Batch Sequential Connectors and Properties

4.2.7.2 Context Diagram

Section 4.2.7.2 contains an overview of all the runtime components and connections found in the Device module.

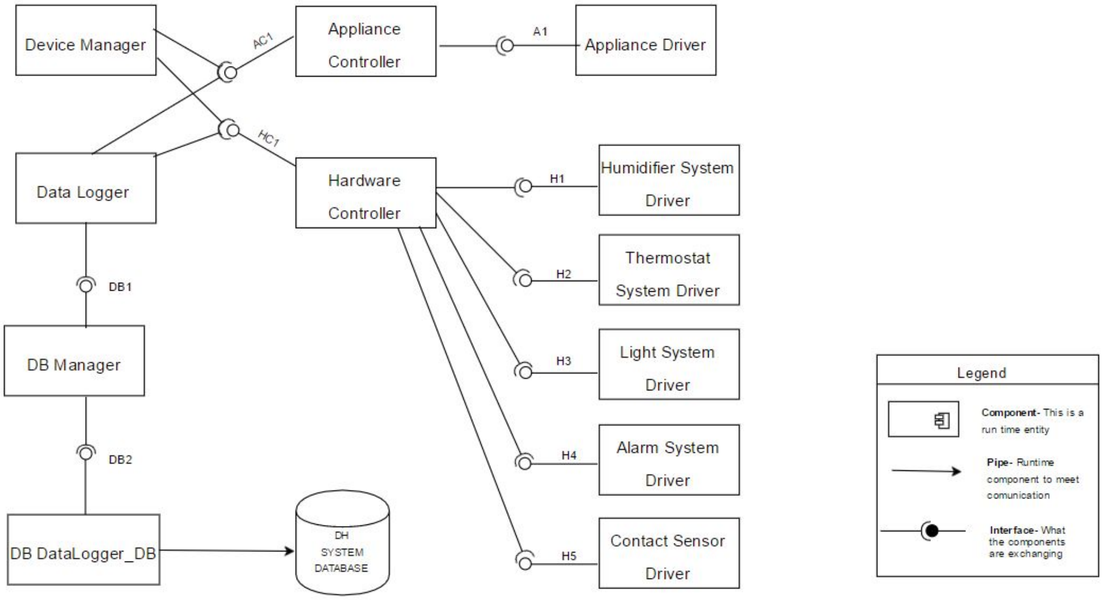

Figure 20: Device Context Diagram

4.2.7.3 Rationale

At runtime when Data Logger is storing data into the Database is batch sequential is because, this linear action can only be performed once a log of data is available to be stored in the Database.

4.3 Client-Server View

Section 4.3 provides a list of modules that fall under the client-server view for the Digital Home's System.

4.3.1 Backup Primary Presentation

This section shows how the client-server style is applied to the backup procedure within the DH System.

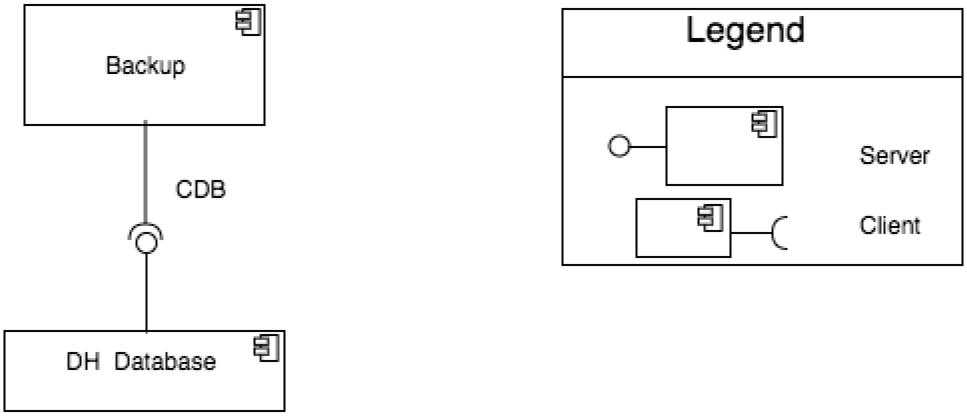

Figure 21: Backup Client DH Database Server

4.3.1.1 Element Catalog

The properties of the components and connectors from section 4.3.1. are found in section 4.2.5.1

4.3.1.2 Restore Primary Presentation

This section shows how the client-server style is applied to the restore procedure within the DH System.

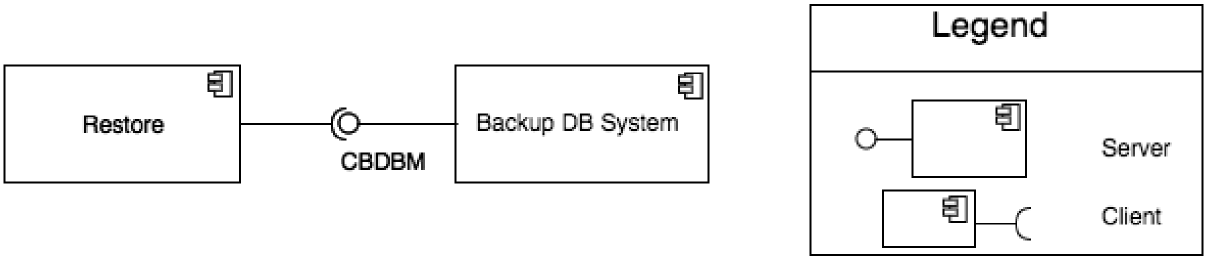

Figure 22: Restore Client Backup DB Server

4.3.1.3 Element Catalog

The properties of the components and connectors from section 4.3.2 are found in section 4.2.6.1

4.3.6 Device Primary Presentation

Section 4.3.6 provides a visualization of the Device module at runtime, specifically under the Client/Server shown in Figure 23.

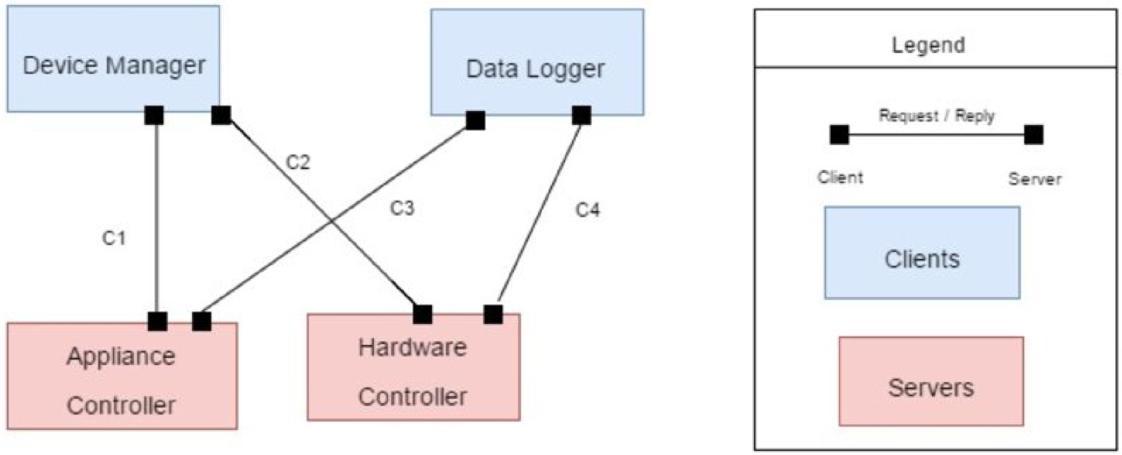

Figure 23: Device Primary Presentation

4.3.6.1 Element Catalog

Section 4.3.6.1.contains description and the properties of each of the components and connectors that were identified from the system based on the Client/Server style.

Components	Properties
1	**Name:** Device Manager **Reliability:** 0.7 % likelihood of failure. **Performance:** Latency: 30 ms - 50 ms. Response Time: 2 ms - 6 ms. **Functionality:** Request services that deal with Hardware and/or Appliance Drivers. **Interface:** This component has the following interfaces: The following calls are redirected to Device Manager + turnOnAppliance(appliance_ID: int , state: String): Boolean; + turnOffAppliance(appliance_ID:int , state: String): Boolean; + getApplianceState(appliance-ID, int, state:String):int; + turnOnSystem(system_ID: int): Boolean; + turnOffSystem(system_ID:int): Boolean; + getSystemState(system_ID, int):String; + contactSensor_ChangeState(sensorID: int, state:int)):Boolean; + getcontactSensorState(sensorID:int):String; **Concurrency:** N/A **Tier:** 2

	Multiplicity: There is only one instance of these at a time. **Liveliness:** This component starts when a device request is required and end when request has been completed.
2	**Name:** Data Logger **Reliability:** 0.7 % likelihood of failure. **Performance:** Latency: 30 ms - 50 ms. Response Time: 2 ms - 6 ms. **Functionality:** Requests the current state of Hardware and Appliance Drivers. **Interface:** This component has the following interfaces: The following calls are redirected to Data Logger + storeDataLog(device_ID: int, date:String, deviceData: String): Boolean + logAppliances():String; + logHardwareDevices():String; **Concurrency:** N/A **Tier:** 2 **Multiplicity:** There is only one instance of these at a time. **Liveliness:** This component starts every 30 minutes or when a device has been sent a request. Ends once Data Logger has sent the query to DB Manager.
3	**Name:** Appliance Controller **Reliability:** 0.7 % likelihood of failure. **Performance:** Latency: 30 ms - 50 ms. Response Time: 2 ms - 6 ms. **Functionality:** Provides services that require Appliance Drivers **Interface:** This component has the following interfaces: The following calls are redirected to Appliance Controller + turnOnAppliance(appliance_ID: int , state: String): Boolean; + turnOffAppliance(appliance_ID:int , state: String): Boolean; + getApplianceState(appliance-ID, int, state:String):int; **Concurrency:** N/A **Tier:** 2 **Multiplicity:** There is only one instance of these at a time. **Liveliness:** Starts when a Appliance request have been made, ends when request is fulfilled
4	**Name:** Hardware Controller **Name:** Data Logger **Reliability:** 0.7 % likelihood of failure. **Performance:** Latency: 30 ms - 50 ms. Response Time: 2 ms - 6 ms. **Functionality:** Provides services that require Hardware Drivers **Interface:** This component has the following interfaces: The following calls are redirected to Hardware Controller + turnOnSystem(system_ID: int): Boolean; + turnOffSystem(system_ID:int): Boolean; + getSystemState(system_ID, int):String;

	+ contactSensor_ChangeState(sensorID: int, state:int)):Boolean; + getcontactSensorState(sensorID:int):String **Concurrency:** N/A **Tier:** 2 **Multiplicity:** There is only one instance of these at a time. **Liveliness:** Starts when a Hardware request have been made, ends when request is fulfilled

Table 22: Device Client Server Components and Properties

Connector	Properties
1	**Name:** C1 **Data:** TBD **Size:** This connection can handle 30 kB of data **Reliability:** 98% Reliable **Performance:** Latency: 4 ms - 5 ms. **Functionality:** Channels all calls to the Appliance Controller **Concurrency:** N/A **Multiplicity:** Single instance
2	**Name:** C2 **Data:** TBD **Size:** This connection can handle 30 kB of data **Reliability:** 98% Reliable **Performance:** Latency: 4 ms - 5 ms. **Functionality:** Channels all calls to Hardware Controller **Concurrency:** N/A **Multiplicity:** Single instance
3	**Name:** C3 **Data:** TBD **Size:** This connection can handle 30 kB of data **Reliability:** 98% Reliable **Performance:** Latency: 4 ms - 5 ms. **Functionality:** Channels all calls to the Appliance Controller **Concurrency:** N/A **Multiplicity:** Single instance
4	**Name:** C4 **Data:** TBD **Size:** This connection can handle 30 kB of data **Reliability:** 98% Reliable **Performance:** Latency: 4 ms - 5 ms. **Functionality:** Channels all calls to Hardware Controller **Concurrency:** N/A **Multiplicity:** Single instance

Table 23: Device Client Server Connectors and Properties

4.3.6.3 Context Diagram

Section 4.3.6.3 Provides the overall view of Report at runtime labeled Figure 24.

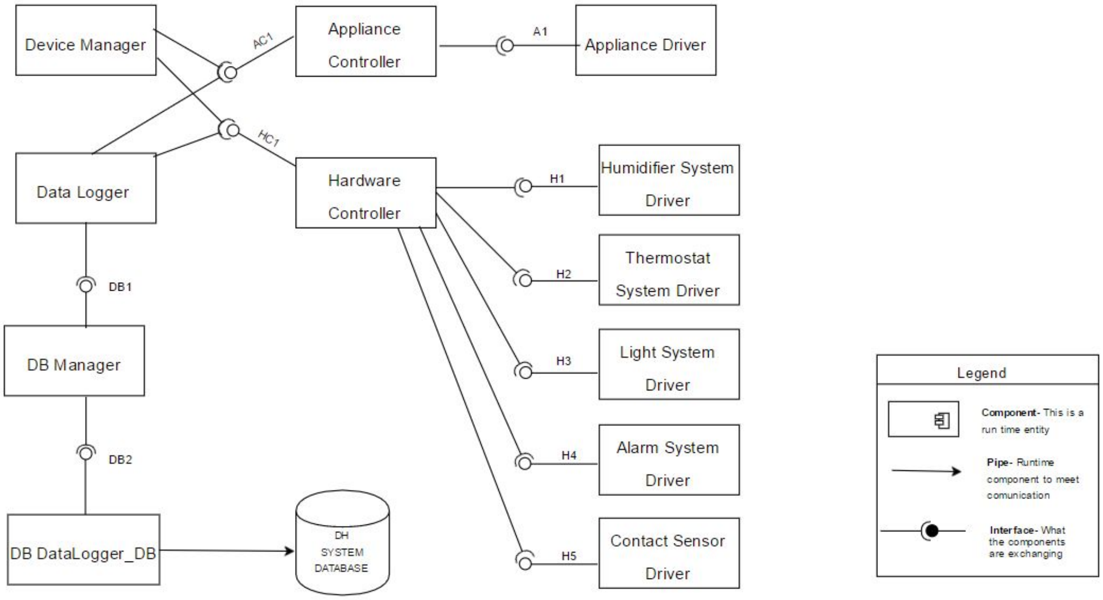

Figure 24: Device Context Diagram

4.3.6.4 Rationale

Device Manager and Data Logger request services from System Devices through their controllers Appliance Controller and Hardware Controller.

4.3.3 User Settings Primary Presentation

Section 4.3.3 provides a visualization of Settings Module at runtime labeled Figure 15

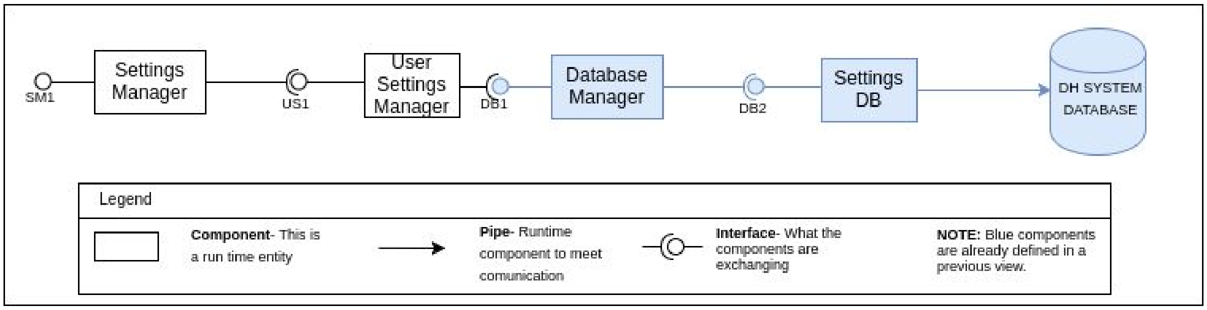
SM1
Settings Manager
US1
User Settings Manager
DB1
Database Manager
DB2
Settings DB
DH SYSTEM DATABASE
Legend
Component- This is a run time entity
Pipe- Runtime component to meet comunication
Interface- What the components are exchanging
NOTE: Blue components are already defined in a previous view.

Figure 25: User Settings Primary Presentation

4.3.3.1 Element Catalog

Section 4.3.3.1 contains description and the properties of each of the components and connectors that were identified from the system.

Element	Properties
1	**Name:** Settings Manager **Reliability:** 0.7 % likelihood of failure. **Performance:** Latency: 30 ms - 50 ms. Response Time: 2 ms - 6 ms. **Functionality:** Settings Manager handles all communication for the settings and scheduling functionality of the system. It redirects calls to the appropriate component to handle the call. **Interface:** This component has the following interfaces: The following calls are redirected to user settings + addUser(first name: String, last name: String, permissions: String): Boolean + removeUser(UserID: int): Boolean + modifyUser(UserID: int, first name: String, last name: String, permissions: String): Boolean + getUserSettings(UserID: int): void + getUsers(): void The following calls are redirected to system settings + modifySystemSetting(SSettingID: int, value: String): Boolean + getSystemSetting(UserID: int): Map + getAllSystemSettings(): Map The following calls are redirected to scheduler + createEvent(startDate: Date, endDate: Date, DeviceId: int, settings: Map, time: Time) + modifyEvent(EventID): Boolean + getSchedule(Month: int): Schedule + getSchedule(Year: int): Schedule + getSchedule(day: Date): Schedule + deleteEvent(EventID): Boolean The following calls are redirected to device settings + getDeviceList(): Map + addDevice(device_serial: int): Map + removeDevice(Device_Id: int): Boolean + setDevice(DeviceId: int, settings: Map): Boolean

	+ getDeviceSettings(DeviceId: int): Map **Concurrency:** N/A **Tier:** 1 **Multiplicity:** There is only one instance of these at a time. **Liveliness:** This component starts when the system starts and dies when the system dies.
2	**Name:** User Settings Manager **Reliability:** 0.2 % likelihood of failure. **Performance:** Latency: 30 ms - 50 ms. Response Time: 2 ms - 6 ms **Functionality:** This component gets all the calls from the system regarding changing and obtaining user information. It serves the user information, as well as add new users, modify their permissions, and even delete users. **Concurrency:** N/A **Tier:** 1 **Interface:** This component has the following interfaces: + addUser(first name: String, last name: String, permissions: String): Boolean + removeUser(UserID: int): Boolean + modifyUser(UserID: int, first name: String, last name: String, permissions: String): Boolean + getUserSettings(UserID: int): void + getUsers(): void **Multiplicity:** Only a single instance exists at a time. **Liveliness:** This component starts when a call for this service is called and dies when it is done doing its work.
3	**Name:** Database Manager For properties on this component, refer to Section 4.6
4	**Name:** Settings DB For properties on this component, refer to Section 4.6

Table 24: User Settings Client Server Components and Properties

Connector	Properties
SM1	**Name: SM1** **Data:** TBD **Size:** This connection can handle 30 kB of data **Reliability:** 98% Reliable **Performance:** Latency: 4 ms - 5 ms. **Functionality:** Channels all calls to the settings subsystem **Concurrency:** N/A **Multiplicity:** Single instance
US1	**Name: US1** **Data:** TBD **Size:** This connection can handle 10 kB of data **Reliability:** 97% Reliable **Performance:** Latency: 1 ms - 3 ms.

	Functionality: Channels all calls to the user settings component **Concurrency:** N/A **Multiplicity:** Single instance
DB1	For properties on this component, refer to Section 4.6
DB2	For properties on this component, refer to Section 4.6

Table 25: User Settings Client Server Connectors and Properties

4.3.4 System Settings Primary Presentation

Section 4.3.4 provides a visualization of Settings Module at runtime labeled Figure 26

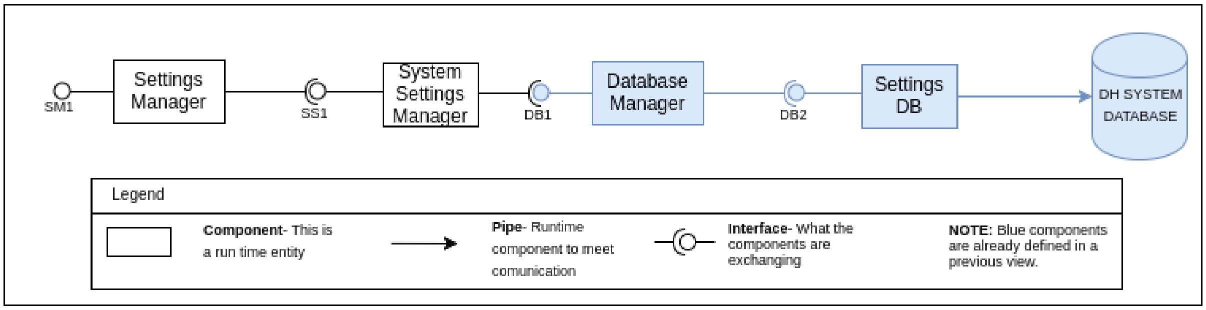
SM1
Settings Manager
SS1
System Settings Manager
DB1
Database Manager
DB2
Settings DB
DH SYSTEM DATABASE
Legend
Component- This is a run time entity
Pipe- Runtime component to meet comunication
Interface- What the components are exchanging
NOTE: Blue components are already defined in a previous view.

Figure 26: System Settings Primary Presentation

4.3.4.1 Element Catalog

Section 4.3.4.1 contains description and the properties of each of the components and connectors that were identified from the system based on the Client Server style.

Element	Properties
1	**Name:** Settings Manager For properties on this component, refer to Section 4.3.3
2	**Name:** System Settings Manager **Reliability:** 0.2 % likelihood of failure. **Performance:** Latency: 30 ms - 50 ms. Response Time: 2 ms - 6 ms **Functionality:** This component gets all the calls from the system regarding CRUD for system-wide settings. The system-wide settings are settings such as language, starting day of the week among others. **Concurrency:** N/A **Tier:** 1 **Interface:** This component has the following interfaces: + modifySystemSetting(SSettingID: int, value: String): Boolean + getSystemSetting(UserID: int): Map + getAllSystemSettings(): Map **Multiplicity:** Only a single instance at a time **Liveliness:** Starts up when the program starts and remains alive until the end of the program.
3	**Name:** Database Manager For properties on this component, refer to Section 4.6.
4	**Name:** Settings DB For properties on this component, refer to Section 4.6.

Table 26: System Settings Client Server Components and Properties

Connector	Properties
SM1	For properties on this component, refer to Section 4.3.3
SS1	**Name: SS1** **Data:** TBD **Size:** This connection can handle 15 kB of data **Reliability:** 96% Reliable **Performance:** Latency: 3 ms - 6 ms. **Functionality:** Channels all calls to the system settings component **Concurrency:** N/A **Multiplicity:** Single instance
DB1	For properties on this component, refer to Section 4.6
DB2	For properties on this component, refer to Section 4.6

Table 27: System Settings Client Server Connectors and Properties

4.3.5 Scheduler Primary Presentation

Section 4.3.5 provides a visualization of Settings Module at runtime labeled Figure 17

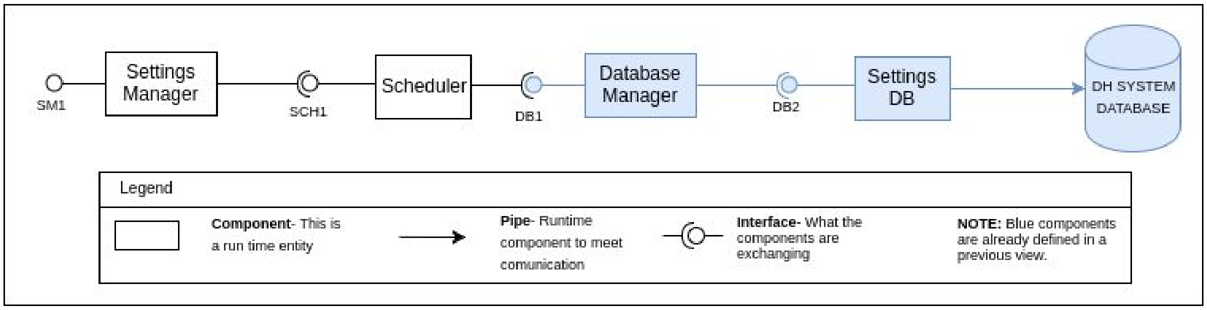
SM1
Settings Manager
SCH1
Scheduler
DB1
Database Manager
DB2
Settings DB
DH SYSTEM DATABASE
Legend
Component- This is a run time entity
Pipe- Runtime component to meet comunication
Interface- What the components are exchanging
NOTE: Blue components are already defined in a previous view.

Figure 27: Scheduler Primary Presentation

4.3.5.1 Element Catalog

Section 4.3.5.1 contains description and the properties of each of the components and connectors that were identified from the system based on the client-server style.

Element	Properties
1	**Name:** Settings Manager For properties on this component, refer to Section 4.3.3
2	**Name:** Scheduler **Reliability:** 0.2 % likelihood of failure. **Performance:** Latency: 30 ms - 50 ms. Response Time: 2 ms - 6 ms **Functionality:** This component is in charge of implementing the calendar function, where device events can be scheduled. The device events is what drives events to do certain actions at specified times. **Concurrency:** N/A **Tier: 1** **Interface:** This component has the following interfaces: + createEvent(startDate: Date, endDate: Date, DeviceId: int, settings: Map, time: Time) + modifyEvent(EventID): Boolean + getSchedule(Month: int): Schedule + getSchedule(Year: int): Schedule + getSchedule(day: Date): Schedule + deleteEvent(EventID): Boolean **Multiplicity:** Only a single instance at a time **Liveliness:** Starts when the program starts and dies when the program ends
3	**Name:** Database Manager For properties on this component, refer to Section 4.6
4	**Name:** Setting DB For properties on this component, refer to Section 4.6

Table 28: Scheduler Client Server Components and Properties

Connector	Properties
SM1	For properties on this component, refer to Section 4.3.3
SCH1	**Name: SCH1** **Data:** **Size:** This connection can handle 60 kB of data **Reliability:** 98% Reliable **Performance:** Latency: 3 ms - 7 ms. **Functionality:** Channels all calls to the scheduler component **Concurrency:** N/A **Multiplicity:** Single instance
DB1	For properties on this component, refer to Section 4.6
DB2	For properties on this component, refer to Section 4.6

Table 29: Scheduler Client Server Connectors and Properties

4.3.6 Device Settings Primary Presentation

Section 4.3.6 provides a visualization of Settings Module at runtime labeled Figure 18

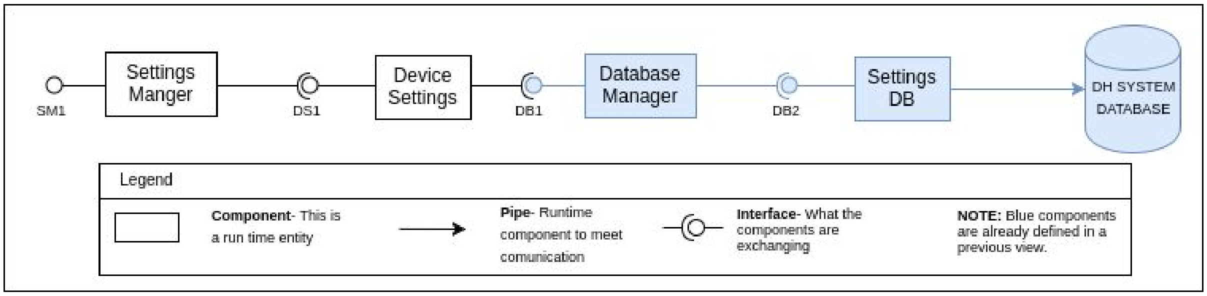
SM1
Settings Manger
DS1
Device Settings
DB1
Database Manager
DB2
Settings DB
DH SYSTEM DATABASE
Legend
Component- This is a run time entity
Pipe- Runtime component to meet comunication
Interface- What the components are exchanging
NOTE: Blue components are already defined in a previous view.

Figure 28 Device Settings

4.3.6.1 Element Catalog

Section 4.3.6.1 contains description and the properties of each of the components and connectors that were identified from the system based on the client-server style.

Element	Properties
1	**Name:** Settings Manager For properties on this component, refer to Section 4.3.3
2	**Name:** Device Settings **Reliability:** 0.2 % likelihood of failure. **Performance:** Latency: 30 ms - 50 ms. Response Time: 2 ms - 6 ms **Functionality:** This component is in charge of providing adding, deleting components, as well as for setting the settings for any device event. **Concurrency: N/A** **Tier:** 1 **Interface:** This component has the following interfaces: + getDeviceList(): Map + addDevice(device_serial: int): Map + removeDevice(Device_Id: int): Boolean + setDevice(DeviceId: int, settings: Map): Boolean + getDeviceSettings(DeviceId: int): Map **Multiplicity:** There is only a single instance of this component at a time **Liveliness:** This component exists only during changes to calendar and dies when the calendar is not being used.
3	**Name:** Database Manager For properties on this component, refer to Section 4.6
4	**Name:** Settings DB For properties on this component, refer to Section 4.6

Table 30: Device Settings Client Server Components and Properties

Connector	Properties
SM1	For properties, refer to section 4.3.3
DS1	**Name: DS1** **Data:** **Size:** This connection can handle 40 kB of data **Reliability:** 99% Reliable **Performance:** Latency: 4 ms - 7 ms. **Functionality:** Channels all calls to directed towards the device settings component. **Concurrency:** N/A **Multiplicity:** Single instance
DB1	For properties on this component, refer to Section 4.6
DB2	For properties on this component, refer to Section 4.6

Table 31: Device Settings Client Server Connectors and Properties

4.4 Publish-Subscribe View

4.4.1 Backup Primary Presentation

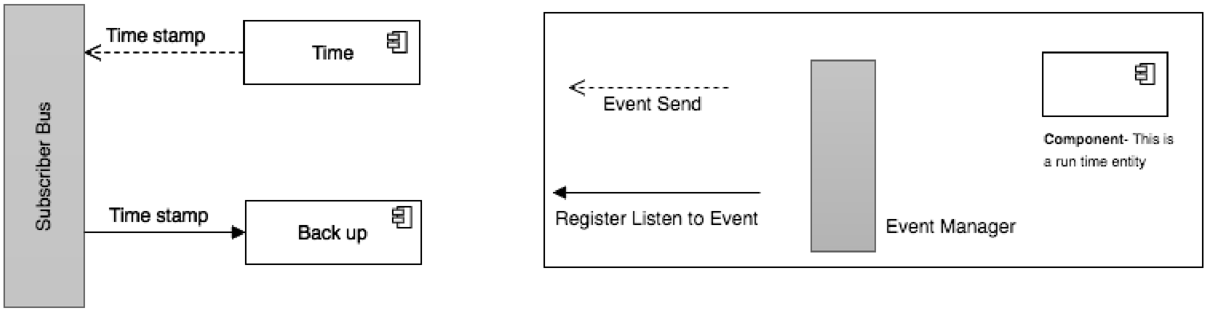

Figure 29: Backup Publish-Subscribe Primary Presentation

4.4.1.1 Element Catalog

Section 4.4.1.1 contains a table along with the properties of the components and connectors found in section 4.4.1.

Element	Properties
1.1	**Name:** Time **Reliability:** 0.03 % likelihood of failure. **Performance:** Latency <= 1ms-2ms. Response Time <= 2 ms **Functionality:** This component would provide time stamps to the subscriber bus that would be used by other runtime components. **Liveliness:** This component would initiate every time the DH system gets initiated and would not terminate until the DH system is shut down. This component would always be alive.

Table 32: Backup Publish Subscribe Components and Properties

4.4.2 Device Primary Presentation

This section contains the primary presentation of the devices of the system.

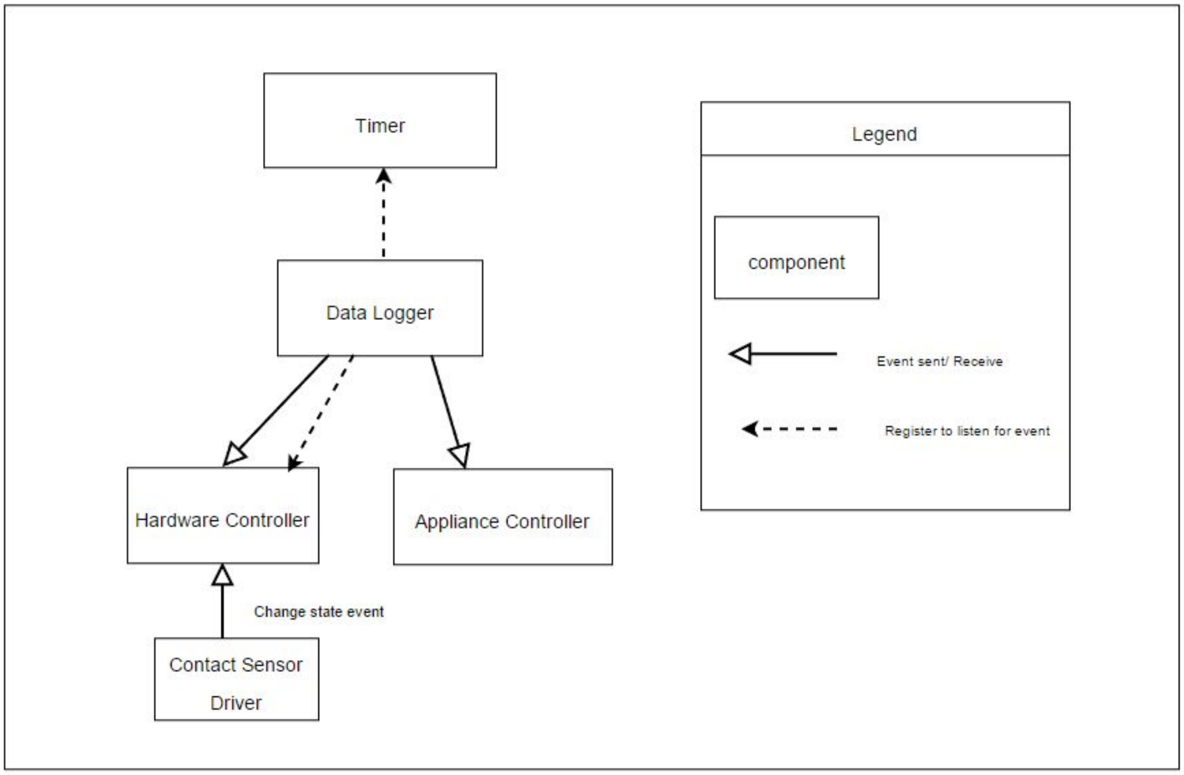

Figure 30: Device Primary Presentation

4.4.2.1 Element Catalog

Section 4.4.2.1 contains the components along with their properties found in Figure 30.

Element	Properties
1	**Name:** Timer **Reliability:** 0.7 % likelihood of failure. **Performance:** Latency: 30 ms - 50 ms. Response Time: 2 ms - 6 ms. **Functionality:**Broadcasts Alarm every 30 minutes **Interface:** This component has the following interfaces: The following calls are redirected to Timer + Alarm_Ring(): Boolean; **Concurrency:** N/A **Tier:** 2 **Multiplicity:** There is only one instance of these at a time. **Liveliness:** Every 30 minutes
2	**Name:** Data Logger **Reliability:** 0.7 % likelihood of failure. **Performance:**

	Latency: 30 ms - 50 ms. Response Time: 2 ms - 6 ms. **Functionality:** Subscribed to changes make in Hardware and Appliance Controller. Listens for alerts from Timer and Hardware Controller. Documents all events that may be triggered. **Interface:** This component has the following interfaces: The following calls are redirected to Data Logger + storeDataLog(device_ID: int, date:String, deviceData: String): Boolean + logAppliances():String; + logHardwareDevices():String + alarmBroadCast_Listener():Boolean; + alarmSystem_Listener():Boolean; **Concurrency:** N/A **Tier:** 2 **Multiplicity:** There is only one instance of these at a time. **Liveliness:** This component starts every 30 minutes or when a device has been sent a request. Ends once Data Logger has sent the query to DB Manager.
3	**Name:** Appliance Controller Refer to Section 4.3.6.1
4	**Name:** Hardware Controller **Reliability:** 0.7 % likelihood of failure. **Performance:** Latency: 30 ms - 50 ms. Response Time: 2 ms - 6 ms. **Functionality:** Subscribed to Contact Sensors Driver, for any state changes made on a contact sensor. **Interface:** This component has the following interfaces: The following calls are redirected to Hardware Controller + contactStateSignal():Boolean; + alertAlarmSystem(contactSensor_Data:String):void; **Concurrency:** N/A **Tier:** 2 **Multiplicity:** There is only one instance of these at a time. **Liveliness:** Begins when a state of the Contact Sensor is Changed
5	**Name:** Contact Sensor Driver **Reliability:** 0.7 % likelihood of failure. **Performance:** Latency: 30 ms - 50 ms. Response Time: 2 ms - 6 ms. **Functionality:**Broadcasts an alert anytime a state change from a contact sensor has occurred. **Interface:** This component has the following interfaces: The following calls are redirected to Data Logger + stateChange_Ring(): Boolean; + stateChange(sensor_ID:int):Boolean; + connect_Magnets()Boolean; + disconnect_Magnets():Boolean; **Concurrency:** N/A **Tier:** 2

	Multiplicity: There is only one instance of these at a time. **Liveliness:** This component starts whenever a contact sensor request is made or an unexpected state change from a sensor.

Table 33: Component Table of Device

Connector	Properties
1	**Name: C1** **Data:** **Size:** This connection can handle 30 kB of data **Reliability:** 98% Reliable **Performance:** Latency: 4 ms - 5 ms. **Functionality:** Listens for Timer's Alarm **Concurrency:** N/A **Multiplicity:** Single instance
2	**Name: C2** **Data:** **Size:** This connection can handle 30 kB of data **Reliability:** 98% Reliable **Performance:** Latency: 4 ms - 5 ms. **Functionality:** Event for Timer's alarm or System Alarm has been activated. Log current state **Concurrency:** N/A **Multiplicity:** Single instance
3	**Name: C3** **Data:** **Size:** This connection can handle 30 kB of data **Reliability:** 98% Reliable **Performance:** Latency: 4 ms - 5 ms. **Functionality:** Listens for a Hardware Controller request to trigger Alarm System. **Concurrency:** N/A **Multiplicity:** Single instance
4	**Name: C4** **Data:** **Size:** This connection can handle 30 kB of data **Reliability:** 98% Reliable **Performance:** Latency: 4 ms - 5 ms. **Functionality:** Event for Timer's alarm has been activated and Log current state **Concurrency:** N/A **Multiplicity:** Single instance
5	**Name: C5** **Data:** **Size:** This connection can handle 30 kB of data **Reliability:** 98% Reliable **Performance:** Latency: 4 ms - 5 ms. **Functionality:** Event to Broadcast Contact Sensor state has unexpectedly changed

	Concurrency: N/A **Multiplicity:** Single instance

Table 34: Connector Table of Device

4.2.2.2 Context Diagram

Section 4.2.2.2 Provides the overall view of Report at runtime labeled Figure 31

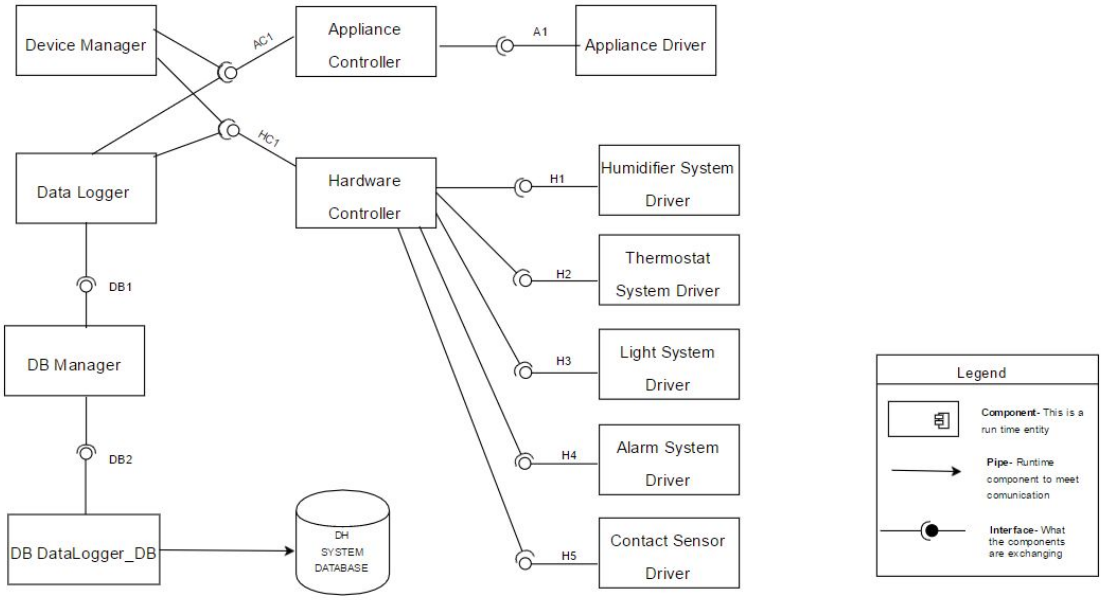

Figure 31: Context Diagram of Devices

4.2.2.3 Rationale

The additional Component Timer has been added to notify the Data Logger to document current system status periodically for accurate reports.

4.5 Service Oriented View

This section contains the Service Oriented style as applied to the system.

4.5.1 Primary Presentation

This section contains the primary presentation of the Service Oriented view.

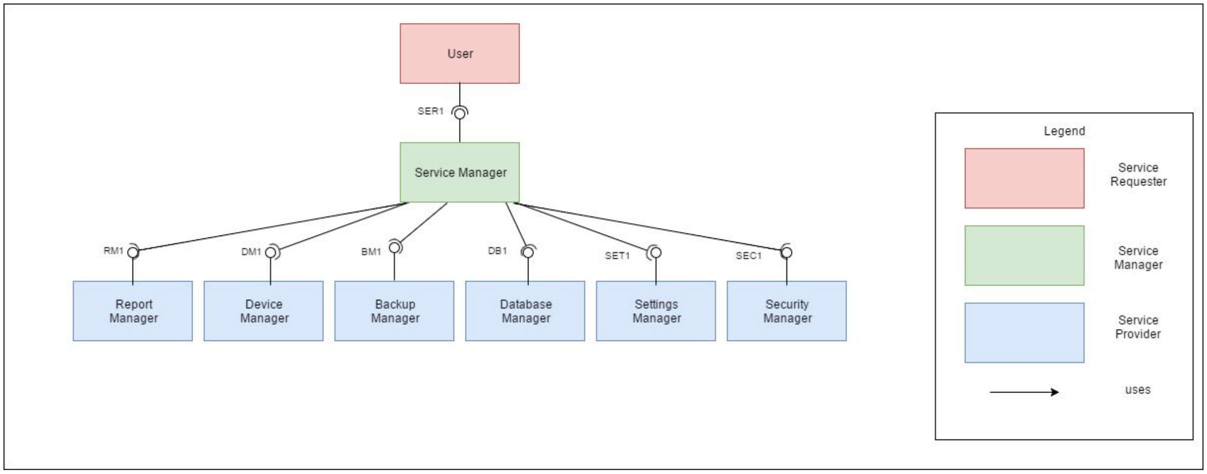

Figure 32: Primary Presentation of SOA

4.5.1.1 Element Catalog

Section 4.2.2 contains description and the properties of each of the components and connectors that were identified from the system based on the Batch Sequential style.

Element	Properties
1	**Name:** User **Reliability:** 0.7 % likelihood of failure. **Performance:** Latency: 30 ms - 50 ms. Response Time: 2 ms - 6 ms. **Functionality:** **Interface:**TBD **Concurrency:** N/A **Tier:** 1 **Multiplicity:** There is only one instance of these at a time. **Liveliness:** This component starts when the system starts and dies when the system dies.
2	**Name:** Settings Manager **Reliability:** 0.2 % likelihood of failure. **Performance:** Latency: 20 ms - 60 ms. Response Time: 3 ms - 7 ms. **Functionality:** This component reroutes all traffic to the services provided by the system. This is not

	an actual component, but it is still represented here to show the flow of data.
3	**Name:** Report Manager For properties on this component, refer to Section 4.3.1
4	**Name:** BackupManager For properties on this component, refer to Section 4.3.1
5	**Name:** SettingsManager For properties on this component, refer to Section 4.3.1
6	**Name:** Security Manager For properties on this component, refer to Section 4.3.1
7	**Name:** Device Manager For properties on this component, refer to Section 4.3.1
8	**Name:** DatabaseManager For properties on this component, refer to Section 4.3.1

Table 35: Components of SOA

Connector	Properties
SER1	**Name:** SM1 **Data:** TBD **Size:** This connection can handle 30 kB of data **Reliability:** 98% Reliable **Performance:** Latency: 4 ms - 5 ms. **Functionality:** Channels all calls to the Service Manager **Concurrency:** N/A **Multiplicity:** Single instance
RM1	For properties on this connector, refer to Section 4.3.1
DM1	For properties on this connector, refer to Section 4.3.1
BM1	For properties on this connector, refer to Section 4.3.1
DB1	For properties on this connector, refer to Section 4.3.1
SET1	For properties on this connector, refer to Section 4.3.1
SEC1	For properties on this connector, refer to Section 4.3.1

Table 36: Connector of SOA

4.5.1.2 Context Diagram

There is no context diagram available for this view.

4.5.1.3 Rationale

The User is an abstraction of the different possible types of users (i.e. Mobile User or Desktop User) in the different states of the system (i.e. Regular or Degraded).

4.6 Repository View

This section contains the system as viewed in the Repository style.

4.6.1 Primary Presentation

This section contains the primary presentation of the repository view.

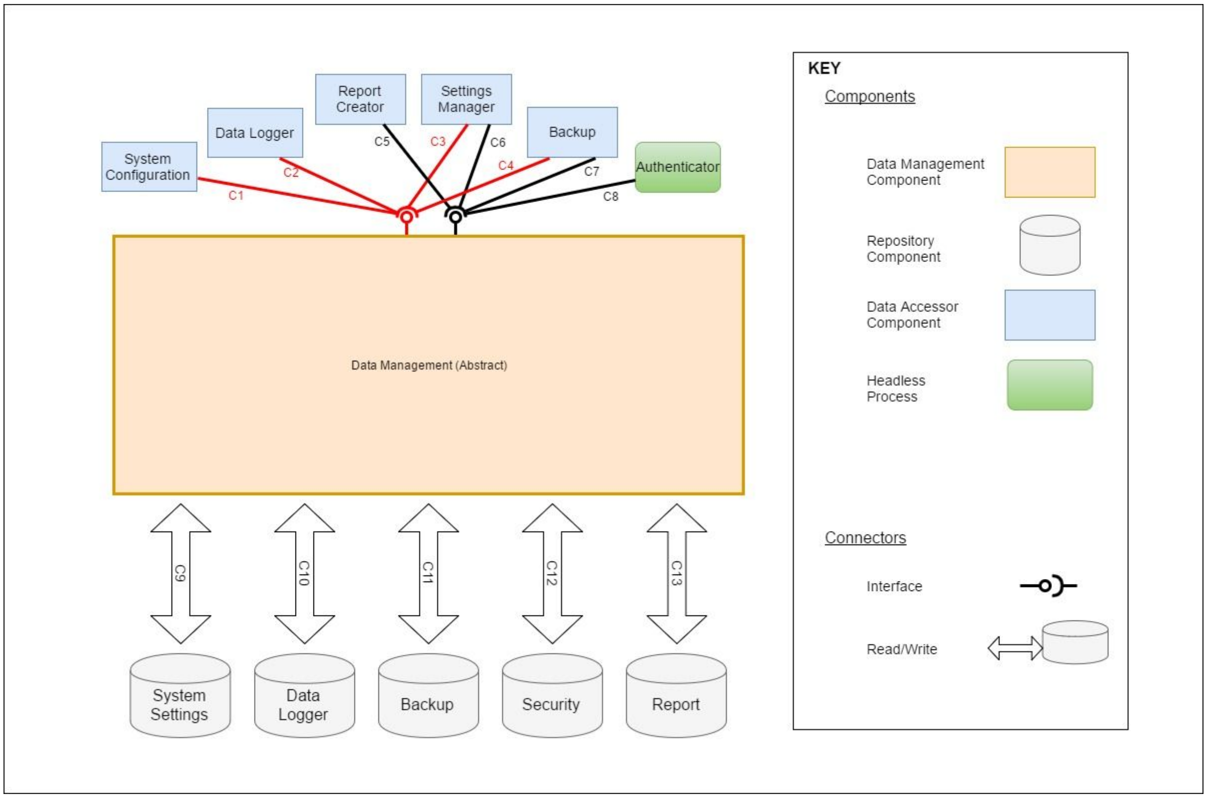

Figure 33: Shared-Data Repository Style

4.6.2 Element Catalog

4.6.2.1 Elements and Their Properties

Element	**Properties**
E1	**Name**: System Configuration **Functionality**: Data accessor component for system configuration data. **Performance**: Latency <= 20ms-40ms Response Time <= 10 ms **Interfaces**: No interface for this component in this view **Ports**: No ports for this component in this view **Events**: No events for this component **Tier**: 3

E2	**Name**: Data Logger **Functionality**: Data accessor component for log data. **Performance**: Latency <= 20ms-40ms Response Time <= 10 ms **Interfaces**: No interface for this component in this view **Ports**: No ports for this component in this view **Events**: No events for this component **Tier**: 3
E3	**Name**: Report Creator **Functionality**: Data accessor component for report data. **Performance**: Latency <= 20ms-40ms Response Time <= 10 ms **Interfaces**: No interface for this component in this view **Ports**: No ports for this component in this view **Events**: No events for this component **Tier**: 3
E4	**Name**: Settings Manager **Functionality**: Data accessor component for system settings data. **Performance**: Latency <= 20ms-40ms Response Time <= 10 ms **Interfaces**: No interface for this component in this view **Ports**: No ports for this component in this view **Events**: No events for this component **Tier**: 3
E5	**Name**: Backup **Functionality**: Data accessor component for backup data. **Performance**: Latency <= 20ms-40ms Response Time <= 10 ms **Interfaces**: No interface for this component in this view **Ports**: No ports for this component in this view **Events**: No events for this component **Tier**: 3
E6	**Name**: Authenticator **Functionality**: Headless process for security authentication data. **Performance**: Latency <= 20ms-40ms Response Time <= 10 ms **Interfaces**: No interface for this component in this view **Ports**: No ports for this component in this view **Events**: No events for this component **Tier**: 3

E7	**Name**: Database Management (Abstract) **Functionality**: Repository management component for database access. **Performance**: Latency <= 60ms-100ms Response Time <= 10 ms **Interfaces**: Provides interfaces to read/write data from the DH Database to E1-E6. +getLogData(Date DateStart, Date DateEnd, String [] DeviceID): Log +getSettingsData(Date DateStart, Date DateEnd, String [] SettingID): Setting +getBackupData(Date Date): Backup +getSecurityData(String Type, String ID): String +writeConfigData(Date DateStamp, String ConfigurationID, Configuration Config): Boolean +writeLogData(Date DateStamp, String DeviceID, Log Log): Boolean +writeSettingData(Date DateStamp, String SettingID, Setting Setting): Boolean +writeBackupData(Date DateStamp, String BackupID, Backup Backup): Boolean **Liveliness**: This element will remain active for duration of program's life. **Multiplicity**: This is an abstract element. This element is separated into five (5) different subcomponents. These subcomponents manage data storage and retrieval of specific information that is on different repositories. The five subcomponents are: Security DB Manager, Backup DB Manager, System Settings DB Manager, Data Log DB Manager, and Report DB Manager. **Ports**: This element has two interfaces. One is a data writer interface (Red). The other one is is a data reader interface (Black). **Events**: No events for this element. **Tier**: 3
E8	**Name**: DH Database **Functionality**: Repository component for entire database. **Performance**: Latency <= 20ms-60ms Response Time <= 10 ms **Interfaces**: Needs to provide access to Database Manager. Database Manager and its subcomponents is the only component that accesses the repository directly. **Liveliness**: This element will remain active for duration of program's life. **Multiplicity**: The database is composed of five different schemas: Security DB, Backup DB, System Settings DB, Data Log DB, and Report DB. Each repository contains separate data. Data is not linked from one repository to any other, however, some managers might retrieve information from one repository to store it in another repository in another format. **Ports**: This element must provide access to Database Management to read and write data into the different schemas. **Events**: No events for this element. **Tier**: 3
E9	**Name**: System Settings DB **Functionality**: Repository component for system settings. **Performance**: Latency <= 20ms-60ms Response Time <= 10 ms **Interfaces**: This repository element should be accessible by the database management abstract element.

	+getData(Query): Boolean +storeData(Query): Boolean **Liveliness**: This element is always active. **Ports**: This element provides a data port for system settings data. **Events**: No events for this element. **Tier**: 3
E10	**Name**: Data Logger DB **Functionality**: Repository component for log data. **Performance**: Latency <= 20ms-60ms Response Time <= 10 ms **Interfaces**: This repository element should be accessible by the database management abstract element. +getData(Query): Boolean +storeData(Query): Boolean **Liveliness**: This element is always active. **Ports**: This element provides a data port for log data. **Events**: No events for this element. **Tier**: 3
E11	**Name**: Backup DB **Functionality**: Repository component for backup data. **Performance**: Latency <= 20ms-60ms Response Time <= 10 ms **Interfaces**: This repository element should be accessible by the database management abstract element. +getData(Query): Boolean +storeData(Query): Boolean **Liveliness**: This element is always active. **Ports**: This element provides a data port for backup data. **Events**: No events for this element. **Tier**: 3
E12	**Name**: Security DB **Functionality**: Repository component for security data. **Performance**: Latency <= 20ms-60ms Response Time <= 10 ms **Interfaces**: This repository element should be accessible by the database management abstract element. +getData(Query): Boolean +storeData(Query): Boolean **Liveliness**: This element is always active. **Ports**: This element provides a data port for security data. **Events**: No events for this element. **Tier**: 3

E13	**Name**: Report DB **Functionality**: Repository component for report data. **Performance**: Latency <= 20ms-60ms Response Time <= 10 ms **Interfaces**: This repository element should be accessible by the database management abstract element. +getData(Query): Boolean +storeData(Query): Boolean **Liveliness**: This element is always active. **Ports**: This element provides a data port for report data. **Events**: No events for this element. **Tier**: 3

Table 37: Components, Properties and Behaviors of Shared Repository Style View.

4.6.2.2 Relations and Their Properties

Element	Properties
C1	**Name**: C1 **Data**: Method call to write system configuration data into database. **Size**: TBD **Reliability**: 0.05 % likelihood of failure. **Performance**: 0.5 ms <= Latency <= 1 ms 9 bytes <= Load <= 64 bytes 0.4 ms <= Response time <= 1 ms **Security**: This connector can only be used to request system configuration data by System Configuration. **Concurrency**: Connectors C1-C8 can run concurrently to request data from database management (abstract). **Multiplicity**: There is only one instance of this connector at a time.
C2	**Name**: C2 **Data**: Method call to write log data into database. **Size**: TBD **Reliability**: 0.05 % likelihood of failure. **Performance**: 0.5 ms <= Latency <= 1 ms 9 bytes <= Load <= 64 bytes 0.4 ms <= Response time <= 1 ms **Security**: This connector can only be used to write log data by Data Logger. **Concurrency**: Connectors C1-C8 can run concurrently to request data from database management (abstract). **Multiplicity**: There is only one instance of this connector at a time.

C3	**Name**: C3 **Data**: Method call to write system settings data into database. **Size**: TBD **Reliability**: 0.05 % likelihood of failure. **Performance**: 0.5 ms <= Latency <= 1 ms 9 bytes <= Load <= 64 bytes 0.4 ms <= Response time <= 1 ms **Security**: This connector can only be used to write system settings data by Settings Manager. **Concurrency**: Connectors C1-C8 can run concurrently to request data from database management (abstract). **Multiplicity**: There is only one instance of this connector at a time.
C4	**Name**: C4 **Data**: Method call to write backup data into database. **Size**: TBD **Reliability**: 0.05 % likelihood of failure. **Performance**: 0.5 ms <= Latency <= 1 ms 9 bytes <= Load <= 64 bytes 0.4 ms <= Response time <= 1 ms **Security**: This connector can only be used to write backup data by Backup. **Concurrency**: Connectors C1-C8 can run concurrently to request data from database management (abstract). **Multiplicity**: There is only one instance of this connector at a time.
C5	**Name**: C5 **Data**: Method call to read log data from database. **Size**: TBD **Reliability**: 0.05 % likelihood of failure. **Performance**: 0.5 ms <= Latency <= 1 ms 9 bytes <= Load <= 64 bytes 0.4 ms <= Response time <= 1 ms **Security**: This connector can only be used to request log data by Report Creator. **Concurrency**: Connectors C1-C8 can run concurrently to request data from database management (abstract). **Multiplicity**: There is only one instance of this connector at a time.

C6	**Name**: C6 **Data**: Method call to read system settings data from database. **Size**: TBD **Reliability**: 0.05 % likelihood of failure. **Performance**: 0.5 ms <= Latency <= 1 ms 9 bytes <= Load <= 64 bytes 0.4 ms <= Response time <= 1 ms **Security**: This connector can only be used to request system settings data by Settings Manager. **Concurrency**: Connectors C1-C8 can run concurrently to request data from database management (abstract). **Multiplicity**: There is only one instance of this connector at a time.
C7	**Name**: C7 **Data**: Method call to read backup data from database. **Size**: TBD **Reliability**: 0.05 % likelihood of failure. **Performance**: 0.5 ms <= Latency <= 1 ms 9 bytes <= Load <= 64 bytes 0.4 ms <= Response time <= 1 ms **Security**: This connector can only be used to request backup data by Backup. **Concurrency**: Connectors C1-C8 can run concurrently to request data from database **management** (abstract). **Multiplicity**: There is only one instance of this connector at a time.
C8	**Name**: C8 **Data**: Method call to read security data from database. **Size**: TBD **Reliability**: 0.05 % likelihood of failure. **Performance**: 0.5 ms <= Latency <= 1 ms 9 bytes <= Load <= 64 bytes 0.4 ms <= Response time <= 1 ms **Security**: This connector can only be used to request security data by Authenticator. **Concurrency**: Connectors C1-C8 can run concurrently to request data from database management (abstract). **Multiplicity**: There is only one instance of this connector at a time.

C9	**Name**: C9 **Data**: System settings data. **Size**: TBD (Dependent on Entity Relationship diagram and Repository design). **Reliability**: 0.05 % likelihood of failure. **Performance**: 0.5 ms <= Latency <= 1 ms Load: TBD 0.4 ms <= Response time <= 1 ms **Security**: This connection will be used by the System Settings_DB component that handles database management of system settings data. **Concurrency**: This connection might be used concurrently with connectors C10-C13. **Multiplicity**: There is only one instance of this connector at a time.
C10	**Name**: C10 **Data**: Log data. **Size**: TBD (Dependent on Entity Relationship diagram and Repository design). **Reliability**: 0.05 % likelihood of failure. **Performance**: 0.5 ms <= Latency <= 1 ms Load: TBD 0.4 ms <= Response time <= 1 ms **Security**: This connection will be used by the Data Logger_DB component that handles database management of log data. **Concurrency**: This connection might be used concurrently with connectors C9, C11-C13. **Multiplicity**: There is only one instance of this connector at a time.
C11	**Name**: C11 **Data**: Backup data. **Size**: TBD (Dependent on Entity Relationship diagram and Repository design). **Reliability**: 0.05 % likelihood of failure. **Performance**: 0.5 ms <= Latency <= 1 ms Load: TBD 0.4 ms <= Response time <= 1 ms **Security**: This connection will be used by the Backup_DB component that handles database management of backup data. **Concurrency**: This connection might be used concurrently with connectors C9,C10, C12, C13. **Multiplicity**: There is only one instance of this connector at a time.

C12	**Name**: C12 **Data**: Security data. **Size**: TBD (Dependent on Entity Relationship diagram and Repository design). **Reliability**: 0.05 % likelihood of failure. **Performance**: 0.5 ms <= Latency <= 1 ms Load: TBD 0.4 ms <= Response time <= 1 ms **Security**: This connection will be used by the Security_DB component that handles database management of security data. **Concurrency**: This connection might be used concurrently with connectors C9-C11, C13. **Multiplicity**: There is only one instance of this connector at a time.
C13	**Name**: C13 **Data**: Report data. **Size**: TBD (Dependent on Entity Relationship diagram and Repository design). **Reliability**: 0.05 % likelihood of failure. **Performance**: 0.5 ms <= Latency <= 1 ms Load: TBD 0.4 ms <= Response time <= 1 ms **Security**: This connection will be used by the Report_DB component that handles database management of report data. **Concurrency**: This connection might be used concurrently with connectors C9-C12. **Multiplicity**: There is only one instance of this connector at a time.

Table 38: Connectors for Repository View.

4.6.2.3 Elements Interfaces

Refer to Table 37.

4.6.2.4 Elements Behaviors

Refer to Table 37.

4.6.3 Context Diagram

This section contains the context context diagram repository view.

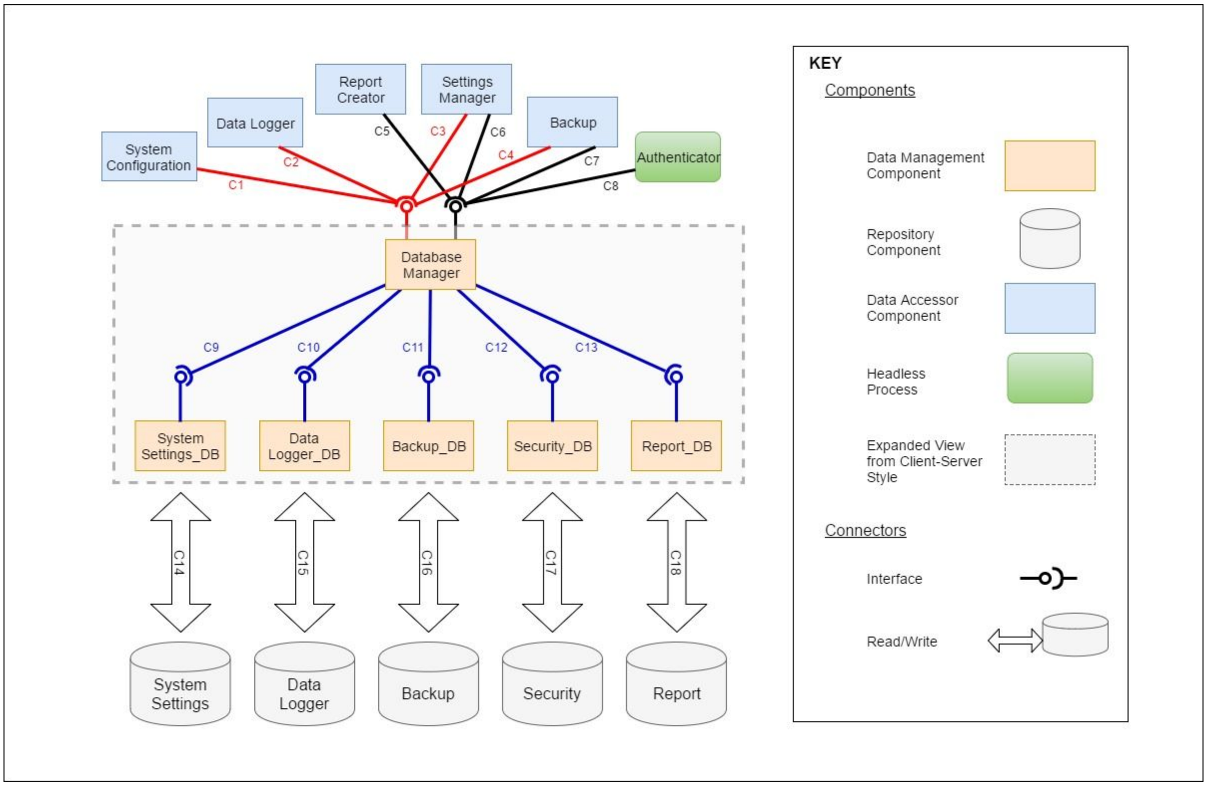

Figure 34: Shared-Data Repository Style with Expanded Client-Server Style View

4.6.4 Rationale

The entire Database Subsystem is shown in this style view. The Database Management abstract component is expanded in the context diagram in this section. This abstract component follows the client-server style and is described in the Client-Server View section of this document. There is only one database, DH-Database. This database, however, is broken down into five different schemas that can be accessed concurrently to aid in accessibility and performance of repository data requests.

5 Allocation Views

Introduction
Purpose and overview of your Allocation views.
Breakdown of the Section

5.1 Deployment View

5.1.1 Primary Presentation

5.1.2 Element Catalog

5.1.2.1 Elements and Their Properties

5.1.2.2 Relations and Their Properties

5.1.2.3 Elements Interfaces

5.1.2.4 Elements Behaviors

5.1.3 Context Diagram

5.1.4 Rationale

5.2 Install View

5.2.1 Primary Presentation

5.2.2 Element Catalog

5.2.2.1 Elements and Their Properties

5.2.2.2 Relations and Their Properties

5.2.2.3 Elements Interfaces

5.2.2.4 Elements Behaviors

5.2.3 Context Diagram

5.2.4 Rationale

5.3 Work-Assignment View

5.3.1 Primary Presentation

5.3.2 Element Catalog

5.3.2.1 Elements and Their Properties

5.3.2.2 Relations and Their Properties

5.3.2.3 Elements Interfaces

5.3.2.4 Elements Behaviors

5.3.3 Context Diagram

5.3.4 Rationale

6 Appendix A

6.1 Osate Layer View

6.2 Code Repository

The pdf files for the AADL Code can be found in the following link:

https://drive.google.com/open?id=0B_kWRxLZdmeJd1NROUY3RG5WMDA

The code is a work in progress. For the final version of this document, textual representation of the AADL Code will be made available here.

7 End of Document

Ώ