Department of Computer Science and Engineering
The University of Texas at Arlington

GLaDOS

Christian Burkhart
Shawn Chandwani
Jose Flores
George Oroni
John Podolanko

Last Updated: October 7, 2014 3:26 AM

Table of Contents

Document Revision History

Revision Number	Revision Date	Description	Rationale
0.1	9-Aug-14	ADS First Draft	Initial ADS distribution
0.2	11-Aug-14	ADS Second Draft	Completed Requirement Mapping section
1.0	12-Aug-14	ADS Gate Review	Redefined entire architecture design layers and subsystems
1.1	4-Sept-14	ADS Gate Review Revisions	Made peer and stakeholder recommended revisions
2.0	5-Sept-14	ADS Baseline	Final revisions made
2.1	7-Oct-14	ADS Baseline - Revised	Updated to match DDS

List of Figures

List of Tables

1. Introduction

This document describes the architecture for Presence, the RFID/sensor-based home management system. It will also elaborate on the system's meta-architecture and proceed to outline the details of each architecture layer and subsystem. It will introduce the system's various layers, subsystems, interfaces, and data flows as well as provide the set of guiding principles upon which the system will be developed. Lastly, it will describe operating system dependencies and testing considerations. This section will include the product concept, scope and key requirements necessary to design the architecture.

1.1 Product Concept

Presence has several functionalities and requirements that it is expected to perform for its user. It automates several common tasks at home such as turning lights off/on, regulating the temperature, locking/unlocking doors, and other tasks. Presence can allow tasks to be automated via the use of sensors or manually configured via an Android™ application. The purpose behind these features is to provide the user with a system that automates the most common and menial of daily tasks and to give the user a single platform to manually control and configure these tasks.

Since the main purpose of this product is aimed at helping the user automate certain aspects of their daily life, Presence will configure household peripheral settings based on information received from sensors or manual input. In addition, Presence is designed to simplify the user's daily life.

One scenario in which Presence will be useful is when a user returns home with their hands full carrying grocery bags or a briefcase. This situation is where Presence really works on behalf of the user. By utilizing sensors and having a centralized server to process this data, the user does not have to set something down to find their keys to unlock the door and then turn on a light.

1.2 Product Scope

Presence will consist of a server, RFID tags and sensors, and infrared transmitters and receivers. They will all work in conjunction with an Android™ application. The server will run a lightweight Linux operating system that enables control of sensors and various switches and devices in the home from lights to ceiling fans to door locks to security cameras. Users will carry an RFID tag in their possession, which will trigger events to happen in the home when in close proximity to the RFID sensors.

User settings can be automated. For instance, when a user comes home, the RFID sensor detects the user and sends a signal to the server which unlocks the door and turn on the lights. The system also adjusts the thermostat to the user's preferred settings. When the user leaves and the sensor no longer detects a targeted presence, the door is locked, the lights are turned off, and the thermostat is shut off or adjusted to conserve energy.

Presence's associated mobile application allows the user to create, modify, and delete various home settings on a per-user basis. Presence also allows manual control over various household settings. For instance, if a user is at home and decides to watch a movie, the user can select a setting mode from the mobile application that turns off the lights in the room while the user is present. When the movie is finished, the user can select a different setting mode that turns the lights back on. If the user wishes to go to sleep, the user can select a setting mode that turns off the lights, adjusts the thermostat, and turns on a fan. These are just a few of several configurable possibilities.

Presence will accommodate several users, each of which can have their own settings configurable on the server from the mobile application. Some settings such as thermostat control will have a hierarchal list of users whose preferences override those of others.

Lastly, the administrator side of the mobile application allows a user to monitor conditions and override settings in the home when no one is present or when multiple users are present. The user may have the ability to view the feed from security cameras and may also manually adjust settings. For instance, if a user is out of town for a few days, the user may wish to turn on the lights and the TV in the evenings to indicate the presence of a person in the home to would-be burglars.

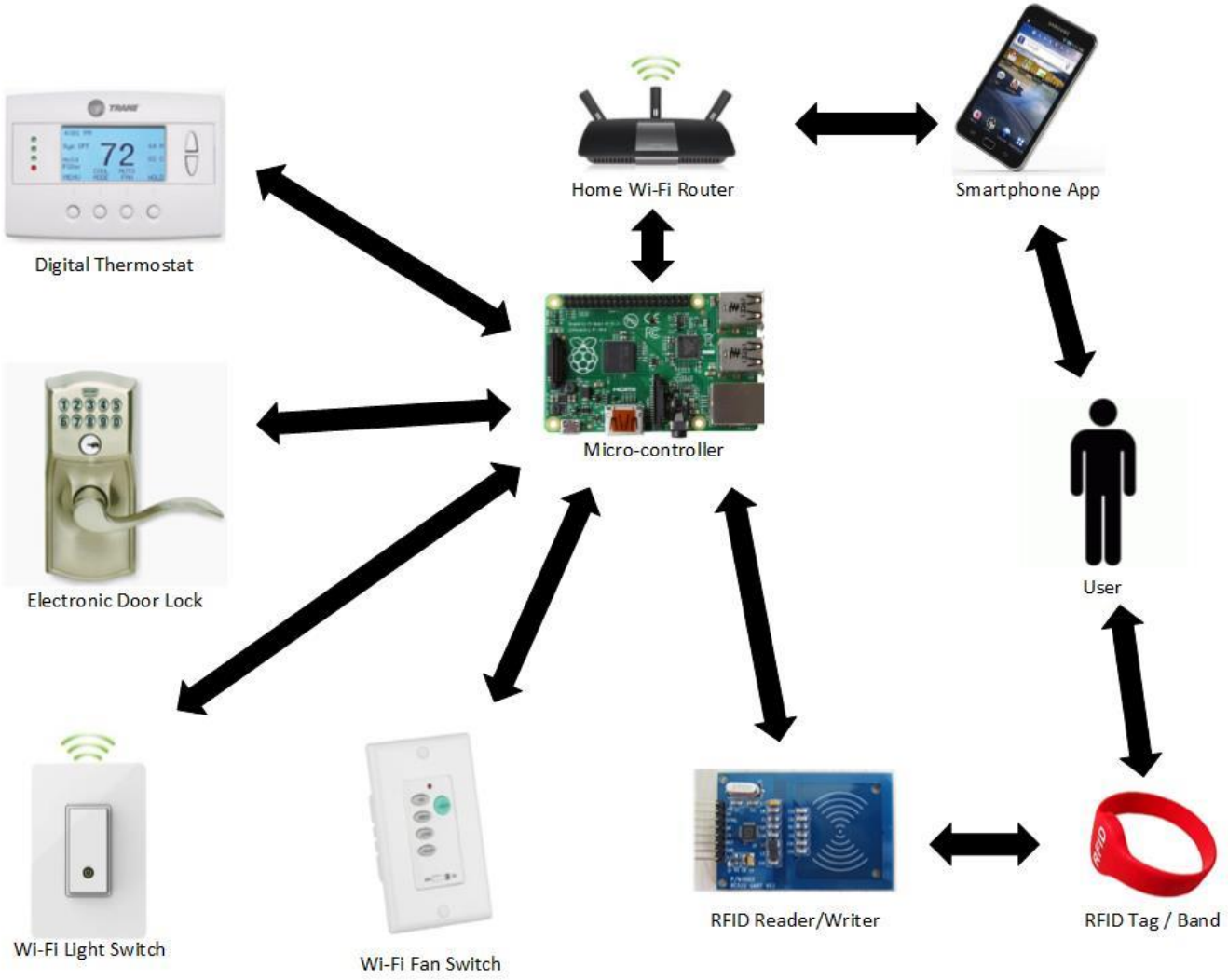

Figure 1-1 High-level concept diagram

1.3 Key Requirements

Requirement Number	Requirement Name	Description
3.1	Centralized Server	There will be a centralized server that controls all of the required household peripherals, stores user accounts and settings, and maintains the database. It will also interface with the RFID reader/writer to associate RFID tags to specific users.
3.2	Compatible Light/Fan Switches	Light and fan switches will need to be integrated into the system either by wire or wirelessly to receive commands from the server.
3.3	Electronic Door Lock	An electronic door lock will need to be integrated into the system either by wire or wirelessly to receive commands from the server.
3.4	Digital Thermostat	A digital thermostat will need to be integrated into the system either by wire or wirelessly to receive commands from the server.
3.5	RFID Reader/Writer	An RFID reader/writer will be used to interface RFID bands with the server.
3.6	Programmable RFID Bands/Tags	RFID bands or tags will be associated to individual users for identification and must be programmable.
3.7	Android Application	The Android™ application will be used to interface with Presence. The application will give the user the ability to create an account that will be associated with their RFID tag and manually set the settings for subsystems associated with Presence such as lights, fans, door locks, and thermostats.
3.8	Database Management System	There will be a backend database management system set up on the server to store the information and preferences of the users. The server will create queries based on the RFID tag information when an RFID tag has been scanned.
3.9	User Account Administration	The Android™ application will allow a user to create, modify, and delete user accounts and associate them with RFID tags. This information will be stored in the DBMS.
3.11	User Login	The first screen shown by the Android™ application is the ‘Login’ screen, where the user can enter their username and password. The user can also choose whether or not to have the device remember the login information to skip the ‘Login’ screen in the future. The user’s credentials will be authenticated by the DBMS on the server.
3.13	Switch Lights On/Off	The server can switch lights on/off automatically or from manual configuration via the Android™ application. The application will display the current switch position of each light switch under Presence control.

3.14	Set Fan Speeds	The server can adjust fan speeds to off/low/medium/high. It can be done automatically or from manual configuration via the Android™ application. The application will display the current switch position of each fan under Presence control.
3.15	Lock/Unlock Door	The server can lock/unlock the door automatically or from manual configuration via the Android ™ application. The application will display the current lock position of the door under Presence control.
3.16	Adjust Thermostat	The server can adjust the temperature on the digital thermostat automatically or from manual configuration via the Android™ application. The application will display the current temperature setting and let you increase/decrease the temperature number for the thermostat.
8.4	Manual Override	In the event of loss of power to the system or other system failure, the user should not be locked out of the home. A physical key will still unlock the door and other utilities may be manually turned on and off by the user in any case.

Table 1-1 Key requirements

2. Meta Architecture

This section describes the various design principles that were used in the construction of the system's architecture. It elaborates on the architectural vision of the team, the guiding principles that serve as the foundation for the system architecture, and assumptions as well as tradeoffs associated with the architecture design of Presence.

2.1 Architectural Vision

The architecture design of Presence is based on the principle of simplicity. The architecture design allows for ease of development and easy future modification. Communication between layers and communication between subsystems will be seamless and efficient by design.

The architecture consists of five layers with several subsystems interfacing with each other within and between layers. The Presentation Layer serves as a two-way user interface allowing for segregation between the multiple forms of input and packaging data accordingly. The Data Processing Layer analyzes the data and executes tasks accordingly. The File Storage Layer handles storage of data and the database management system, and the Output Layer determines the appropriate output based on commands received. Altogether, these layers represent the high-level architectural view of Presence.

2.2 Guiding Principles

Before architecture design can be started, it is important to define the guiding principles upon which Presence will be built. This allows the team to have a better understanding of how the system should be designed. The guiding principles for Presence – user-friendly, modularity, reliability, updatability, and safety and security – are described in this subsection.

2.2.1 User-Friendly

Presence will have an intuitive user interface from the start. Once the user has installed the peripherals and installed the application on their mobile device, the user will be able to quickly determine how to setup an account, build settings profiles, and manually configure household peripherals from the UI. Physical hardware installation will be covered in detail in the user manual so that devices can be installed quickly and with basic tools.

2.2.2 Modularity

Presence will be designed so that each hardware peripheral is independent of each other and so that hardware and software are independent of each other. If a user doesn't meet some of the required assumptions, the system should still be configurable to perform a subset of built-in functions. The system is also open for adding additional peripherals not covered in the System Requirements Specification such as security cameras or other sensors.

2.2.3 Reliability

Presence will be reliable and responsive. The user should be able to connect to the system at any time via Local Area Network or via cellular network (3G/4G) to check the status and adjust settings for peripherals in the user's home.

2.2.4 Updatability

Presence will be easy to update for its users. If additional storage is needed on the system, a quick swap of memory cards will do the trick provided the user transfers all files from the original memory card to another.

2.2.5 Safety and Security

Presence will be secure so that users' personal information is kept private. Presence will also be designed so that malicious intent will be easily mitigated and unauthorized intrusion will be easily prevented.

2.3 Assumptions

In order to develop Presence, the team has made assumptions that each user will have certain existing hardware and software features to use with the system. That list of assumptions are described in this subsection.

2.3.1 Android™ Device

Users must have an Android™ device to run the Presence application for optimized use of the system. The Android™ device must be capable of running Android™ software version 4.1.2 or higher (API 16 "Jelly Bean").

2.3.2 Internet Access + Wireless Internet Router

High-speed Internet access will be required to maximize the user's experience. The user must have some form of wireless router to provide Internet access to Presence so remote configuration may be possible. It will also be used to allow wireless communication between the Presence server and the household peripherals it controls. The router should be capable of Universal Plug-and-Play (UPnP) or port-forwarding.

2.3.3 Tools

The user must have some basic tools such as a screwdriver in order to install the hardware peripherals such as the door lock and light and fan switches around the house.

2.3.4 Technical Experience

The user is expected to have some basic technical experience regarding computer installation and smart device configuration. This will remove the need for professional installation of a new system.

2.3.5 Device Controllers

The user's home is expected to have device controllers installed that can be accessed and controlled by the system.

2.4 Tradeoffs

A few tradeoffs must be considered when designing Presence from both hardware and software perspectives. Those tradeoffs are described in this subsection.

2.4.1 One Input Layer vs. Two Input Layers

There are several advantages of having separate input layers segregating hardware and software, but the team decided that it is still possible to consolidate them into one layer since the hardware and software maintain autonomy from one another within that layer.

2.4.2 Security vs. Simplicity

By taking the simple device development route, more requirements can be met and a greater feature set included in the final product. However, doing so greatly sacrifices the safety and security of the user. Securing the system will be more complex to configure and will take more time. That increases the risk for lower-priority requirements to be pushed into the future development category. As user safety and security is paramount to the team and the customer, the team has decided it is more important to design a secure product.

2.4.3 Performance vs. Reliability

Presence is a system in which real-time information and high performance can be useful but is not necessary to deliver positive results to the customer and the user. With that in mind, the team opted to design a more reliable product that stands far less of a chance of failing under stressful conditions.

3. Layer Definitions

This section provides a high-level description of Presence and the three layers it is broken down into. Those layers include the Presentation Layer, Data Processing Layer, and File Storage Layer.

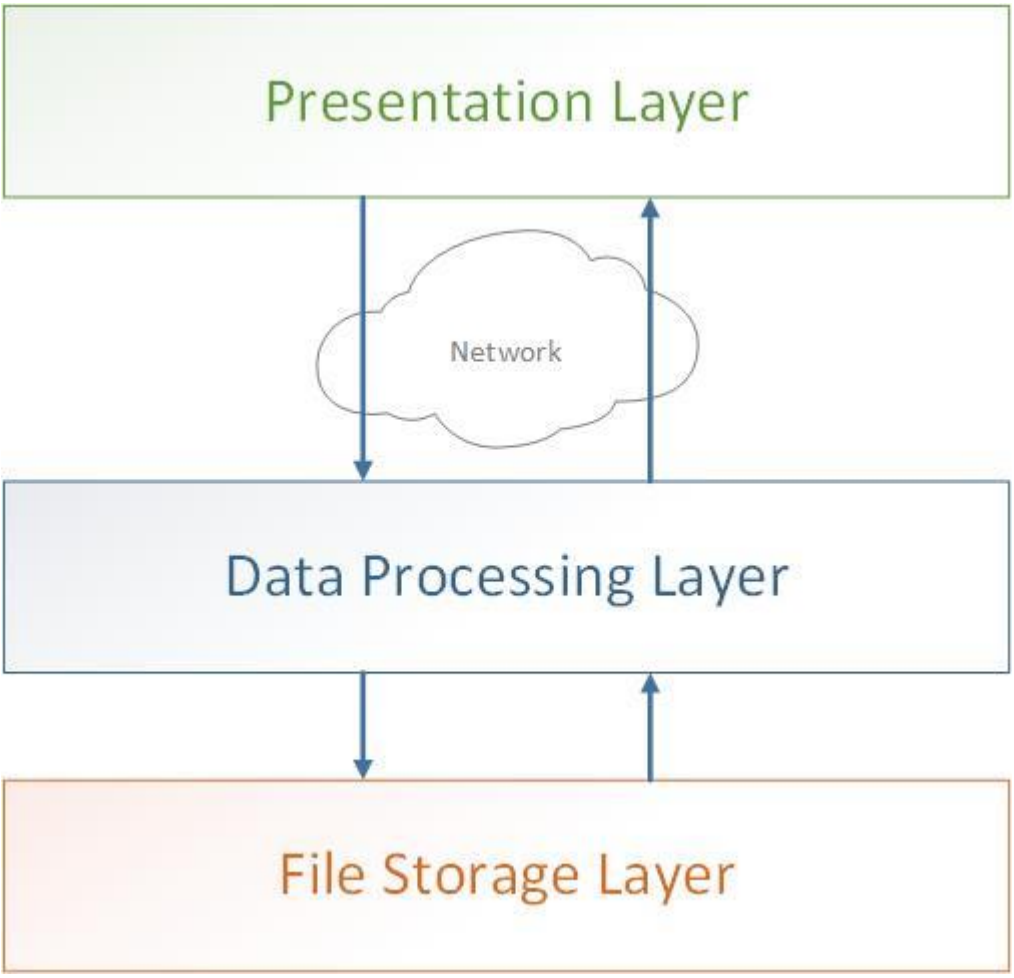

Figure 3-1 High-level architecture layer diagram

3.1 Presentation Layer

The purpose of the Presentation Layer is to accept input from the user and to trigger events that correspond with the needs of the user as well as present the appropriate output. The main sources of input for the user to the rest of the system will be in this layer. There are two types of input that the system accepts: hardware input and software input. Once the user provides input from any of the subsystems, it will be formatted before it is sent to the Data Processing Layer. This layer is also responsible for changing the states of the household peripherals controlled by the system (off/on,

temperature setting, et cetera) and displaying the appropriate information to the user via the Android™ application user interface.

3.2 Data Processing Layer

The purpose of the Data Processing Layer is to analyze and process the data that has been received from the Presentation Layer. This layer contains a data analyzer that interprets the incoming data for the central processing subsystem. This layer also generates database queries and requests for communicating with the File Storage Layer as well as translates information received from the File Storage Layer making a link between the two layers bi-directional.

3.3 File Storage Layer

The purpose of the File Storage Layer is to read and write the data received from the Data Processing Layer. The database subsystem in this layer manages all data reads and writes for Presence. After receiving a formatted query, the database sends formatted data back to the Data Processing Layer.

4. Inter-Subsystem Data Flow

This section illustrates how the various subsystems interact within Presence. Each flow of data is identified with a marker to be described in the following subsections. The subsections provide a high-level overview of the system and the data flows between each of the layers and subsystems as well as describe how individual data elements are used. They also show the relationship between data producers and their respective consumers.

4.1 Data Flow Diagram

The dataflow diagram consists of five layers. Data starts at the Input Layer which has inputs for both the Presence application as well as the hardware. The input layer then flows to the Communications Layer which is responsible for sending the input to the Data Processing Layer. The Data Processing Layer is where the input data is handled. The File Storage Layer is where information will be stored and it communicates back and forth with the Data Processing Layer. Finally, the Data Processing Layer will send the proper information back through the Communications Layer to the Output Layer.

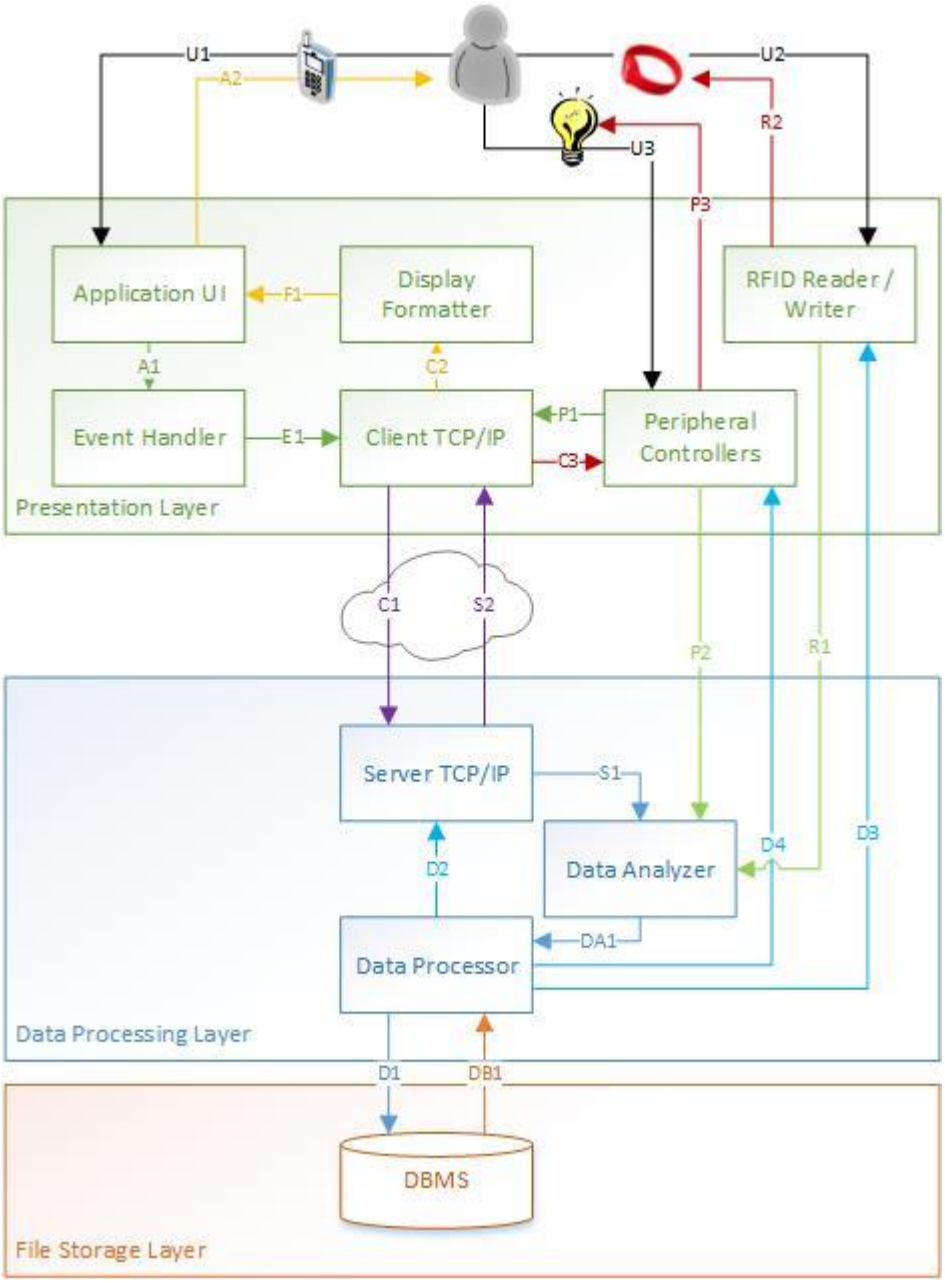

Figure 4-1 Data flow diagram

4.2 Data Flow Definitions

The table below gives a description of each element in the data flow diagram. Each description includes how the data element will be used and passed between subsystems.

Element	Description
U1	User Application input. This includes the user interacting with the presence GUI to perform an action.
U2	User RFID input. The User scans their RFID tag on the RFID reader. This is raw RFID tag data.
U3	User interaction with peripherals and switches. This is raw I/O data for opening and closing switches.
A1	This is the event information derived from user input that is sent to the event handler.
A2	This is visual UI data sent to the user's smartphone. It shows the user the updated status.
E1	The Event Handler will send the action input string to the TCP/IP Subsystem to traverse the network.
P1	The peripherals on the system send their status at regular intervals.
P2	This is the peripheral status signal that gets sent to the Data Analyzer.
P3	This is the opening or closing of a circuit which turns peripherals off/on.
R1	The RFID reader will sent the tag information to the Hardware Data Analyzer. This is an action input string.
R2	This is raw RFID tag information which gets written to the RFID band.
C1	Application events and peripheral statuses are encapsulated into TCP/IP packets and sent across the network.
C2	The application will update the display of the Peripherals' switch positions depending on the output message received from the network. This is a JSON object that is interpreted by the Application UI.
C3	The peripherals will update their status depending on the output message received from the network. This is a JSON object that is interpreted by the peripheral controller.
S1	The action input string from the application and/or peripherals will be sent to the Data Analyzer Subsystem from the network.
S2	Output commands are encapsulated into TCP/IP packets and sent across the network.
DA1	After Analyzing the JSON Object, the Data Analyzer will determine which action to perform and call that action in the main processing subsystem.
D1	The main processing subsystem will query the database for certain actions as needed and send read/write requests for storage data. This data will be SQL data.
D2	The processing subsystem will return an output string for the application or peripheral to the TCP/IP subsystem. This data will be output commands.
D3	The processing subsystem sends command data to the RFID Writer.
D4	The processing subsystem sends command signals to any hard-wired peripheral controllers.
DB1	The database will return results of the query to the processing subsystem when called. This is formatted SQL return data.

Table 4-1 Data flow descriptions

4.3 Producer-Consumer Relationships

This table demonstrates the relationships between data producers and their respective consumers. Producers are represented in the rows on the left and the consumers are represented in the columns on the top.

	User / Device	**Application UI**	**Event Handler**	**Display Formatter**	**RFID Reader/Writer**	**Peripheral Controllers**	**Client TCP/IP Subsystem**	**Server TCP/IP Subsystem**	**Data Analyzer**	**Data Processor**	**DBMS**
User / Device		U1			U2	U3					
Application UI	A2		A1								
Event Handler							E1				
Display Formatter		F1									
RFID Reader/Writer	R2								R1		
Peripheral Controllers	P3						P1		P2		
Client TCP/IP Subsystem				C2		C3		C1			
Server TCP/IP Subsystem							S2		S1		
Data Analyzer										DA1	
Data Processor					D3	D4		S2			D1
DBMS										DB1	

Table 4-2 Producer-consumer relationships

5. Presentation Layer

The purpose of the Presentation Layer is to accept user input from the Presence Android™ application and the RFID reader as well as display output to the user via the application and via the peripherals. The responsibilities entailed in this section include triggering events, executing tasks, sending input to the Data Processing Layer, accepting output from the Data Processing Layer, and network communications.

5.1 Application UI Subsystem

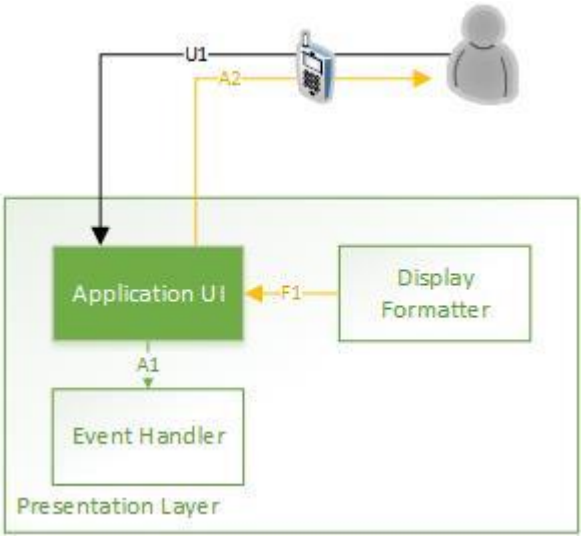

Figure 5-1 Application UI subsystem

5.1.1 Description: The User Interface (UI) subsystems will provide the user with an interface in which they will interact with Presence via their Android™ devices.

5.1.2 Assumptions: The Android™ APK will support all the graphical features envisioned for the application.

5.1.3 Responsibilities: This subsystem will be responsible for rendering and displaying the application's Graphical User Interface (GUI). Features of the GUI will include buttons, dialog boxes, drop-down menus, and the Presence logo among other things. The UI subsystem will also generate events to the system when the user taps or clicks on the responsive or clickable parts of the UI.

5.1.4 Inter-Layer Interfaces: This subsystem does not interface with another layer.

5.1.5 Public Interfaces:

Method	Description	Info Required	Info Returned
getUserInput	The application receives input from the user.	None	None

Table 5-1 Application UI public interfaces

5.2 Application Event Handler Subsystem

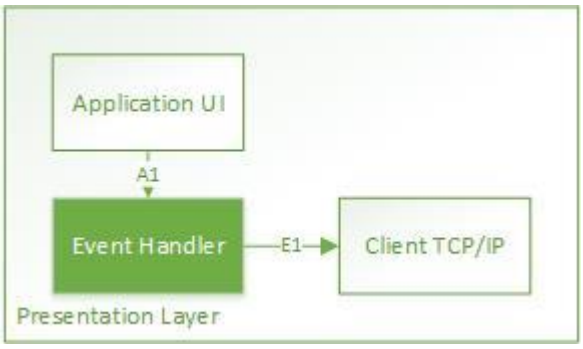

Figure 5-2 Application event handler subsystem

5.2.1 Description: The Event Handler will accept input from the UI subsystem and respond to different events generated by user actions on the application's UI.

5.2.2 Assumptions: The Event Handler will be able to differentiate events from different UI elements.

5.2.3 Responsibilities: This subsystem will uniquely identify events generated by the user through the Presence Android™ application. It will then partially process the event data, create a data package, and then hand it over to the Client TCP/IP subsystem for delivery to the Data Processing Layer.

5.2.4 Inter-Layer Interfaces: This subsystem does not interface with another layer.

5.2.5 Public Interfaces: This subsystem does not have any external interfaces.

5.3 Display Formatter Subsystem

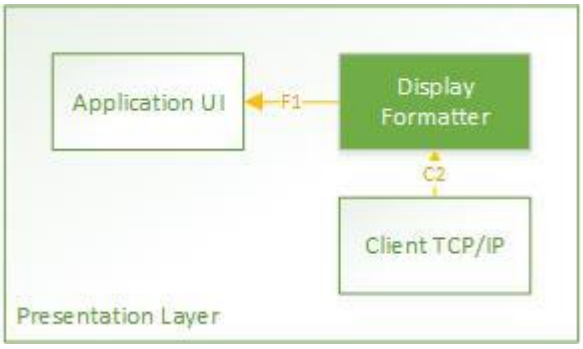

Figure 5-3 Display formatter subsystem

5.3.1 Description: The Event Handler will accept input from the UI subsystem and respond to different events generated by user actions on the application's UI.

5.3.2 Assumptions: The Event Handler will be able to differentiate events from different UI elements.

5.3.3 Responsibilities: This subsystem will uniquely identify events generated by the user through the Presence Android™ application. It will then partially process the event data, create a data package, and then hand it over to the Client TCP/IP subsystem for delivery to the Data Processing Layer.

5.3.4 Inter-Layer Interfaces: This subsystem does not interface with another layer.

5.3.5 Public Interfaces: This subsystem does not have any external interfaces.

5.4 RFID Reader / Writer Subsystem

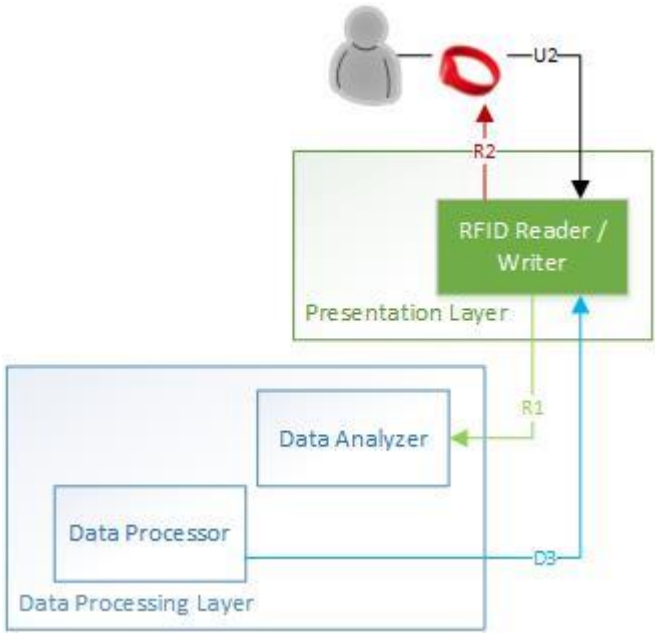

Figure 5-4 RFID reader/writer subsystem

5.4.1 Description: Upon scanning an RFID band/tag, the RFID reader will capture the band's/tag's unique id and generate an event. This event will be sent along with tag data to the Data Processing Layer.

5.4.2 Assumptions: The RFID band/tag will be readable and contains a unique ID associated with a particular individual.

5.4.3 Responsibilities: This subsystem will detect, read, and decode the RFID signal from the RFID band/tag. It will convert that data into an analog signal and pass it to the Data Processing Layer.

5.4.4 Inter-Layer Interfaces:

Method	Description	Info Required	Info Returned
sendRFID	The RFID reader sends tag info to Data Analyzer	RFID number	None
receiveCommand	The RFID writer receives tag data from the Data Processor	RFID number	None

Table 5-2 RFID reader/writer inter-layer interfaces

5.4.5 Public Interfaces:

Method	Description	Info Required	Info Returned
getRFID	The user scans the RFID tag	RFID tag ID	None
writeToRFID	The RFID writer writes tag ID data to an RFID band	RFID number	None

Table 5-3 RFID reader/writer public interfaces

5.5 Hardware Peripheral Controller Subsystem

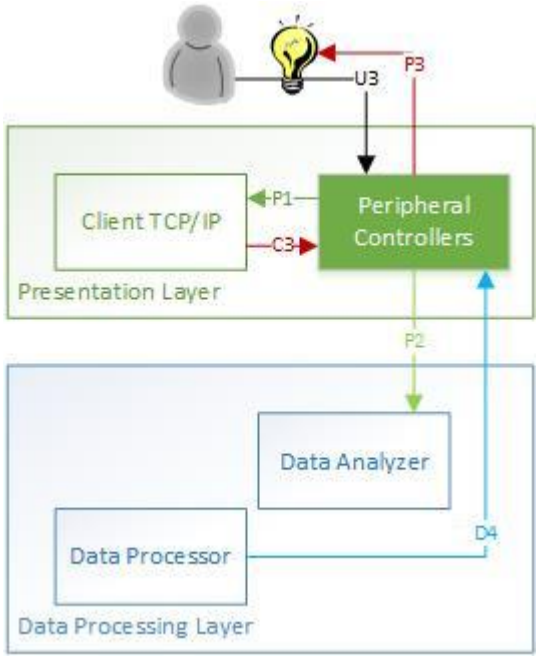

Figure 5-5 Hardware peripheral controller subsystem

5.5.1 Description: The peripherals such as the thermostat, door locks, and light and fan switches generate status input and send that data to the Data Processing Layer via the Client TCP/IP subsystem.

5.5.2 Assumptions: Each peripheral generates a unique input that can be differentiated from the inputs of other peripherals.

5.5.3 Responsibilities: This subsystem will be responsible for capturing peripheral inputs, converting the inputs from analog to digital, and handing the data over to the Client TCP/IP subsystem for delivery to the Data Processing Layer.

5.5.4 Inter-Layer Interfaces:

Method	Description	Info Required	Info Returned
sendSignal	Peripheral state info sent to Data Analyzer	Peripheral state	None
recvCommand	Receives command signal from Data Processor to change state	Command signal	None

Table 5-4 Peripheral controllers inter-layer interfaces

5.5.5 Public Interfaces:

Method	Description	Info Required	Info Returned
getStatus	The user scans the RFID tag	None	Peripheral state
switchCircuit	Switch circuit position of peripheral	Data to be output via hardware	None

Table 5-5 Peripheral controllers public interfaces

5.6 Client TCP/IP Subsystem

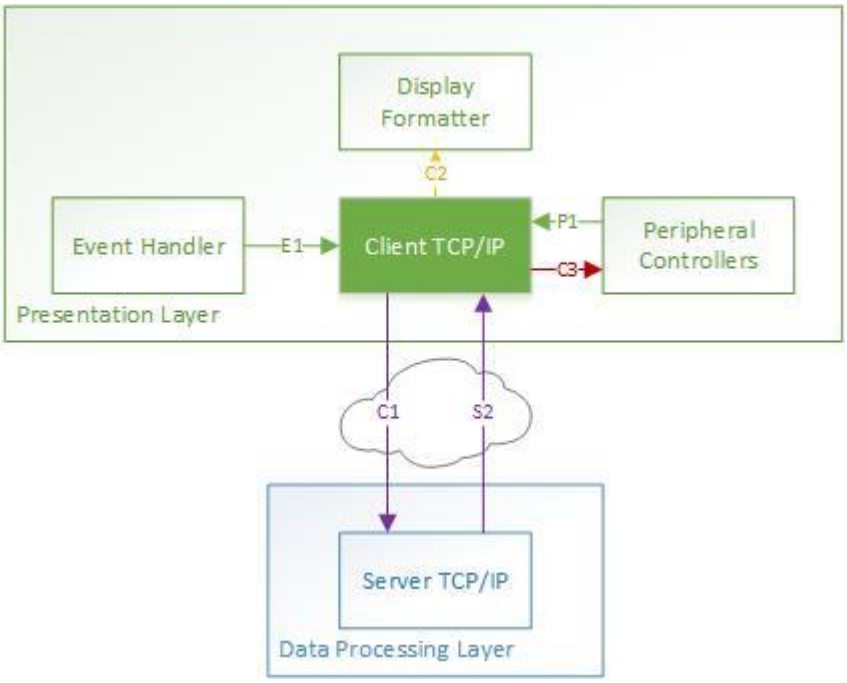

Figure 5-6 Client TCP/IP subsystem

5.6.1 Description: Once user input has been captured and formatted by both hardware and application event handlers, the Client TCP/IP subsystem will facilitate the transmission of the formatted processed input data to the Data Processing Layer and send output data received from the Data Processing Layer to the appropriate device.

5.6.2 Assumptions: Both the sending application/hardware and the receiving hardware/application support the communication protocol used. The communication channel is free from obstruction, interference, or other issues that may alter the nature of the data transmitted. It is assumed that the channel used has security features.

5.6.3 Responsibilities: This subsystem will be responsible for delivering input data from the Presence Android™ application and the peripherals input subsystems to the Data Processing Layer. It will also use Wi-Fi technology to transmit data from Wi-Fi enabled peripherals to the Data Processing Layer.

5.6.4 Inter-Layer Interfaces:

Method	Description	Info Required	Info Returned
sendNetTraffic	Sends input to Data Processing Layer	Input data from application or hardware	None
recvNetTraffic	Receives output from Data Processing Layer	None	Output data from Data Processor

Table 5-6 Client TCP/IP subsystem inter-layer interfaces

5.6.5 Public Interfaces: This subsystem does not have any external interfaces.

6. Data Processing Layer

The purpose of the Data Processing Layer is to analyze and process data as well as send tasks to the Output Layer. The sections below provide a detailed description of this layer and its three subsystems – the Server TCP/IP subsystem, the Data Analyzer, and the Data Processor subsystem.

6.1 Server TCP/IP Subsystem

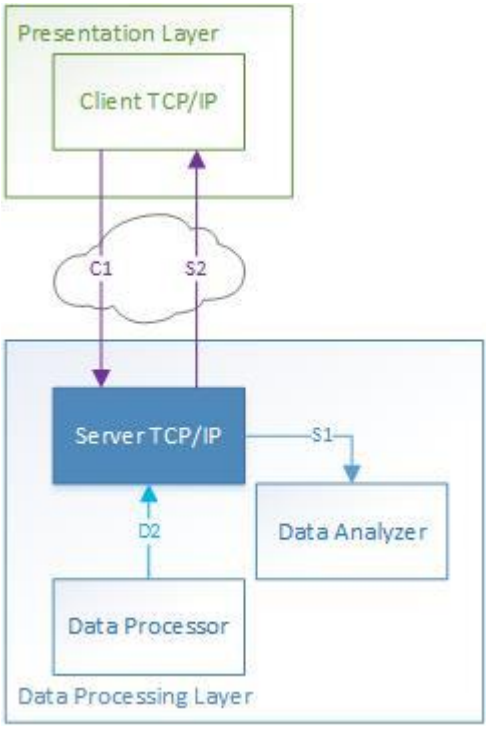

Figure 6-1 Server TCP/IP subsystem

6.1.1 Description: Once user output has been generated for hardware or the application, the Server TCP/IP subsystem will facilitate the transmission of the formatted processed output data to the Presentation Layer and segregate input data received from the Presentation Layer to the appropriate data analyzer.

6.1.2 Assumptions: Both the sending application/hardware and the receiving hardware/application support the communication protocol used. The communication channel is free from obstruction, interference, or other issues that may alter the nature of the data transmitted. It is assumed that the channel used has security features.

6.1.3 Responsibilities: This subsystem will be responsible for delivering output data from the Data Processor to the Presentation Layer. It will also use Wi-Fi technology to transmit data to Wi-Fi enabled peripherals.

6.1.4 Inter-Layer Interfaces:

Method	Description	Info Required	Info Returned
sendNetTraffic	Sends output to Presentation Layer	Output data for application or hardware	None
recvNetTraffic	Receives input from Presentation Layer	None	Input data from Presentation Layer

Table 6-1 Server TCP/IP subsystem inter-layer interfaces

6.1.5 Public Interfaces: This subsystem does not have any external interfaces.

6.2 Data Analyzer Subsystem

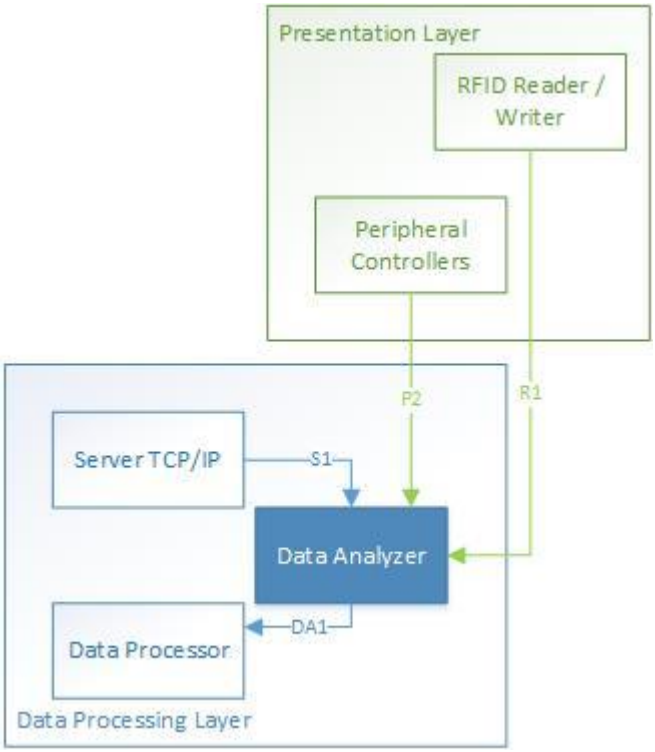

Figure 6-2 Data analyzer subsystem

6.2.1 Description: This subsystem is tasked with accepting, analyzing, and classifying data originating from the Android™ application. This is a component of the server that will actively listen on specific ports for requests and data from the Android™ application.

6.2.2 Assumptions: Data integrity and confidentiality were not compromised during the transmission.

6.2.3 Responsibilities: This subsystem will be responsible for listening for incoming HTTP connections from the Android™ application, analyzing and formatting the data, and passing the data to the central processor.

6.2.4 Inter-Layer Interfaces:

Method	Description	Info Required	Info Returned
DataListener	The Data Analyzer listens for new signal data from the peripherals and RFID R/W in the Presentation Layer	None	Analyzed and classified signal data

Table 6-2 Data analyzer inter-layer interfaces

6.2.5 Public Interfaces: This subsystem does not have any external interfaces.

6.3 Data Processor Subsystem

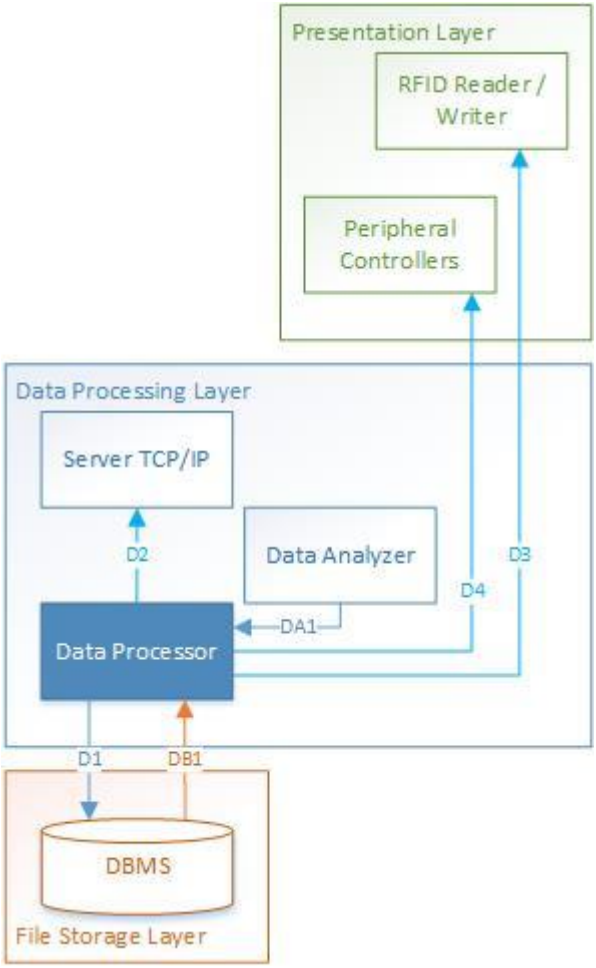

Figure 6-3 Data processor subsystem

6.3.1 Description: The Data Processor will be responsible for making decisions as to what response to send to the user via the Android™ application and what commands to send to which hardware peripherals. It will receive data for decision making from the Data Analyzer subsystems.

6.3.2 Assumptions: The Data Processor subsystem has access to the database. Data received is correctly formatted, and enough memory exists to store and reference data.

6.3.3 Responsibilities: This subsystem will accept data from Hardware and Software Analyzer subsystems, request data from the DBMS subsystem, send data to the DBMS for storage, process combined data, and send output responses to the peripherals and application output subsystems.

6.3.4 Inter-Layer Interfaces:

Method	Description	Info Required	Info Returned
storeData	The Data Processor sends any data that needs to be stored for future use to the database	Data that needs to be saved	None
getData	The Data Processor requests data stored in the database	Address pointer	Data that was requested
sendCommand	Send command signal to peripheral to change its state	Output data for the Presentation Layer	None
writeRFID	Send command with RFID tag information to RFID writer	RFID number	None

Table 6-3 Data processor inter-layer interfaces

6.3.5 Public Interfaces: This subsystem does not have any external interfaces.

7. File Storage Layer

The purpose of the File Storage Layer is to store files and settings associated with Presence. It also responds to database queries and read requests for information stored in memory. The section below provides a detailed description of this layer which contains the Database Management Subsystem.

7.1 Database Management Subsystem

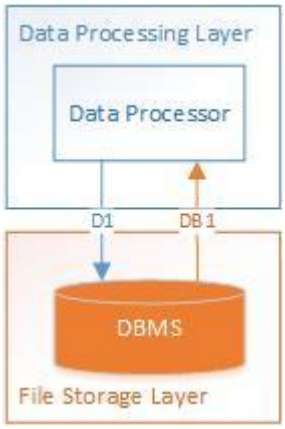

Figure 7-1 Database management subsystem

7.1.1 Description: This subsystem will store both user and peripheral information. The DBMS will provide a way to store collected data for future use. It will also allow read requests for stored data to the Data Processor in the Data Processing Layer.

7.1.2 Assumptions: The database is writable and readable. Also, the database locally resides on the Presence server, and there is enough storage space in the disk.

7.1.3 Responsibilities: This subsystem will be responsible for accepting data from the processing engine, change data format if necessary, locate empty space on the storage disk, and write data to the database. It will also be responsible for locating and retrieving stored data from the database, changing data format if necessary, and passing it to the Data Processor in the Data Processing Layer.

7.1.4 Inter-Layer Interfaces:

Method	Description	Info Required	Info Returned
getSQLQuery	The DBMS receives an SQL query from the Data Processor	None	SQL Query
sendSQLResponse	The DBMS sends an SQL formatted response to the Data Processor	SQL Response	None

Table 7-1 Database management system inter-layer interfaces

7.1.5 Public Interfaces:

Method	Description	Info Required	Info Returned
readMemory	The DBMS retrieves formatted data from the database	Query for data	Data that was requested
writeMemory	The DBMS stores formatted data to the database	Formatted data to be stored	None

Table 7-2 Database management system public interfaces

8. Requirements Traceability

This section outlines the key system requirements and how each subsystem provides functionality to meet them. By analyzing how each layer and subsystem contributes to the app as a whole, insight is gained as to the modularity of the system as well as the overall complexity.

8.1 Requirements Traceability Matrix

Requirement Number	Requirement Name	Presentation	Data Processing	File Storage
3.1	Centralized Server		X	X
3.2	Compatible Light/Fan Switches	X		
3.3	Electronic Door Lock	X		
3.4	Digital Thermostat	X		
3.5	RFID Reader/Writer	X		
3.6	Programmable RFID Bands/Tags	X		X
3.7	Android Application	X		
3.8	Database Management System			X
3.9	User Account Administration	X	X	X
3.10	User Settings Profiles	X	X	X
3.11	User Login	X	X	X
3.13	Switch Lights On/Off	X	X	
3.14	Set Fan Speeds	X	X	
3.15	Lock/Unlock Door	X	X	
3.16	Adjust Thermostat	X	X	
3.17	Administrator Notifications	X	X	
3.18	Custom Peripheral Naming	X	X	X
3.19	Peripheral Status Updates	X	X	
3.20	User Check-In / Check-Out	X	X	X
3.21	Peripheral Malfunction	X	X	
5.1	RFID Recognition Distance	X		
5.2	Settings Activation Latency	X	X	

5.3	Database Query Response Time		X	X
5.4	Server Response Time Over WLAN	X	X	
5.5	Server Response Time Over 3G/4G Networks	X	X	
5.6	Android™ Application Startup Delay	X		
5.7	Peripheral Malfunction Timeout	X		
7.2	Android™ Version Support	X		
7.8	Multiple User Support	X	X	X
7.9	Additional Peripheral Support	X	X	
8.1	Connectivity Tolerance	X	X	
8.2	Application Security & Privacy	X	X	X
8.3	Intrusion Prevention	X	X	X
8.4	Manual Override	X		
8.5	RFID Tag Deactivation	X	X	X

Table 8-1 Requirements traceability matrix

8.2 Requirements Traceability Analysis

From the table above, the team has concluded that the architecture as designed is appropriate for Presence. The client-server model – the client being the Presentation Layer and server being the Data Processing and File Storage Layers – addresses the needs of the project. The Presentation and Data Processing Layers do make up the bulk of the functionality for Presence. The Presentation Layer is the main user interface with the system as it is the only layer with public interfaces. The Data Processing Layer is a bit more complex in that it must analyze and classify all data coming in from the Presentation Layer. The File Storage Layer is the most simplified layer as it only deals with data storage.

Overall, each layer serves unique and specific functions. With each layer providing distinct functionality, the team has concluded that the layers maintain autonomy from each other allowing for modularity.

9. Operating System Dependencies

This section describes any dependencies Presence may have in the form of libraries, operating system and interfaces. The system is designed to be as modular as possible in order to minimize any dependencies.

9.1 Presentation Layer

The Presence Android™ application will be dependent on the Android™ operating system version 4.1.2 or later (API 16 "Jelly Bean"). All user interaction with Presence will require at least a one-time use for setup which will be done via the Android™ application UI. Since Android™ is built on the Java platform, the Java Development Kit and Java Runtime Environment is required along with any libraries needed to communicate with and control various hardware components. The Android™ device must use TCP/IP for network communications.

For hardware peripherals, the libraries used are subject to the operating system of the microcontroller and the different communication protocols for each hardware module. The server will run a Linux distribution, so the hardware peripherals will be dependent on there being an API compatible with the particular flavor of Linux to ensure two-way communications.

9.2 Data Processing Layer

This layer is dependent on the Linux operating system installed on the server. The data processor subsystem is dependent on libraries through an API for each individual hardware component. In order to transfer data needed for processing, the system will utilize networking protocols such as TCP/IP and even UDP as well as different forms of local short-range communications.

9.3 File Storage Layer

This layer is only dependent on Linux operating system installed on the server. All data will be stored and controlled by a database management system installed on the server. Data will be accessible by the hardware components and the Presence application via the database.

10. Testing Considerations

This section describes the testing considerations with respect to the different layers of the system. These considerations are used to aid in development and the testing process. To effectively test the functionality of each layer, communication between layers is kept to a minimum through certain specifications. Testing is conducted to ensure that the functionality of each layer can be reasonably verified for reliability, intuitiveness, and responsiveness.

10.1 Overall Considerations

10.1.1 Code Review: Code reviews will be used frequently to verify that the functions being implementing correspond to one or more requirements. It also provides an opportunity to catch bugs early on that may cause bigger problems in the later stages of development.

10.1.2 Ease of Use: Presence should be intuitive to the user. The user shouldn't be required to understand the implementation of a function in order to use the system.

10.1.3 Beginning-to-End: Upon finishing use cases, the system will be tested from the beginning to the end. The results will be compared to the expected result in order to validate the overall system with respect to the requirements.

10.2 Presentation Layer

10.2.1 Modularity: The Presentation Layer must not be dependent on the internal subsystems of other layers.

10.2.2 Internal System Interaction: The Android™ application UI sends input data to an Event Handler. Hardware peripherals maintain autonomy from the application.

10.2.3 External System Interaction: This layer interacts with the Data Processing Layer via the network. It also receives data from the user.

10.3 Data Processing Layer

10.3.1 Modularity: The Data Processing Layer must not be dependent on the internal subsystems of other layers.

10.3.2 Internal System Interaction: Hardware and software data analyzers will parse and translate data and send it to the Data Processor subsystem for task execution. Hardware and software data analyzers will have no interaction with each other. All subsystems interact with the TCP/IP subsystem to send and/or receive network traffic.

10.3.3 External System Interaction: This layer will interact with the Presentation Layer via networking protocols and direct connection with the RFID reader. It also interacts with the File Storage Layer via database queries and responses.

10.4 File Storage Layer

10.4.1 Modularity: The File Storage Layer must not be dependent on the internal subsystems of other layers.

10.4.2 Internal System Interaction: The only internal interaction will be between the database management system and the SD card where the data is stored.

10.4.3 External System Interaction: The File Storage Layer will receive formatted database queries from the Data Processing Layer and respond with formatted replies.