Faculty of Engineering and Applied Science

ENGR 4941U Capstone Systems Design for ECSE II

Design and Development of Smart Medicine Cabinet

R5: Final Engineering Report

Team Members

Lyba Mughees 10075090

Massimo Albanese 100616057

Abida Choudhury 100700985

Hima Paul 100753261

Faculty Advisor: Dr. Ramiro Liscano

Capstone Coordinator (Fall 2023 & Winter 2024): Dr. Q. Mahmoud

Table of Contents

List of Tables

List of Figures

1. Executive Summary

In the growing era of technology and healthcare, current solutions like traditional pillboxes and medication reminder apps aren't enough to provide comprehensive medication management. Our Smart Medicine Cabinet is unique by offering an all-in-one solution that combines medication intake, reminders, and management without the hassle of downloading multiple apps or manually scuffing through calendars to look at a schedule. The Smart Medicine Cabinet solution provides convenience and efficiency for both patients and their caregivers.

Our Smart Medicine Cabinet tackles this by offering a product that promotes independent living amongst seniors and supports medication management altogether. Instead of relying on the patient to take their pills without any knowledge about them or having a caregiver constantly around, our cabinet allows patients to manage their medications independently while providing remote caregiver assistance if needed. This reduces concerns about frequent missed doses, that can cause adverse health effects and allows patients to have some control over their medications. Currently, our design is intended for a single patient in their home, but with the implementation of multiple profiles, our product can be expanded for larger settings such as retirement or nursing homes.

There are two major components to our project; hardware and software. The hardware includes a touchscreen monitor, webcam, seven pill dispensers, and to control it all, a Raspberry Pi. To control the hardware, we have designed software modules using the Magic Mirror Framework, which allows us to use a modular and adaptable system to meet the various requirements presented to us from our different stakeholders. We used a layered architecture with a Presentation Layer, Business Layer, Data Layer, and Cloud. In the Presentation Layer, modules like Medication-Input, Medication-Scheduler, and Medication-Alarm facilitate user interaction and display medication-related information. The Business Layer hosts core functions like data processing and medication management services. In the Data Layer, modules like SQLite Database Service and Cloud Data Sync Service manage storage and synchronization of medication data between local and cloud databases. The Cloud layer has modules like the Cloud API Interface, Data Analytics Service, and Cloud Database, to allow remote access, data analysis, and secure storage of patient profiles and medication schedules. These modules form an integrated system that ensures efficient medication management, user authentication, and data synchronization between local and cloud environments.

2. R1

1. Problem Identification

This project centers around the creation of a proof-of-concept for a smart mirror cabinet designed to remind dementia patients to take their medications. The proposed system offers elderly individuals an interactive interface suitable for home use. This smart mirror can

recognize medication details through sensors, responding to instructions provided by healthcare professionals. It offers users information about their medication schedules, including dosage and timing.

The mirror operates as a seamless user interface where medication bottles act as the primary input method, generating visual and audio feedback concerning medication timing and dosages. The primary goal of this project is to enhance the user experience and promote independent living among seniors dealing with chronic illnesses while residing at home. The smart mirror cabinet is designed to overcome common barriers older adults face when interacting with technological devices. It is intended to provide support for seniors who may struggle to maintain their independence and to minimize medication errors, a critical concern for healthcare safety. The mirror serves as a human communication interface, ensuring effective interaction with the elderly to deliver essential health services.

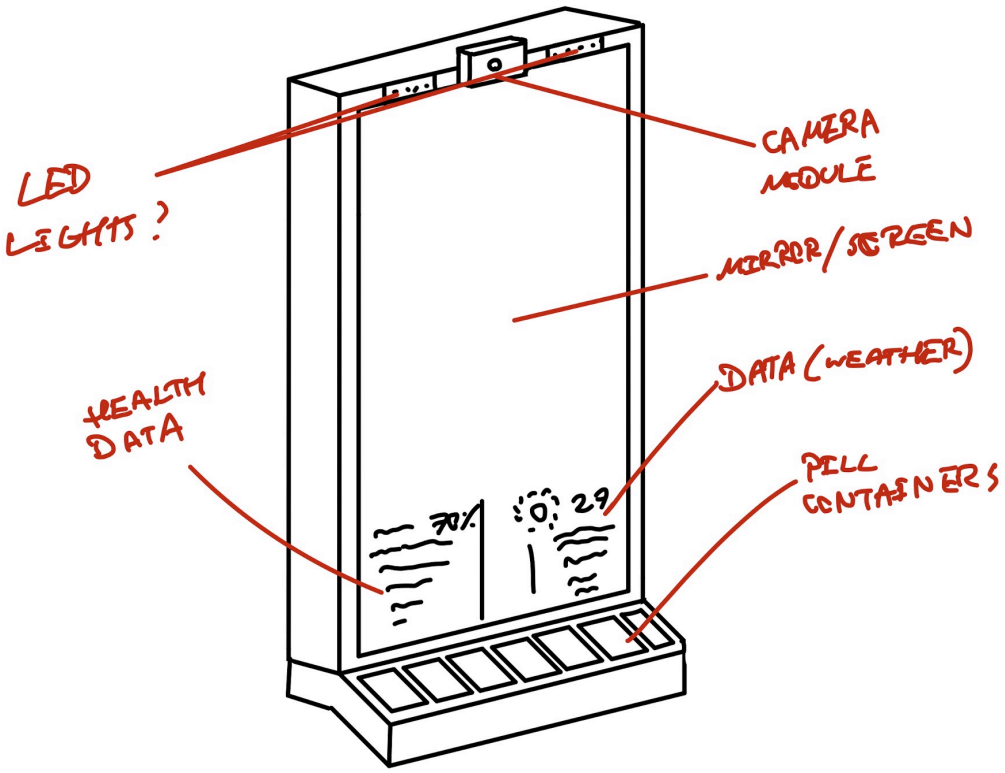


Figure 2.1: Prototype of Smart Mirror Cabinet

2. Project-related Background and Research Review

This capstone project builds upon previous students' research endeavors in the field of developing applications for smart cabinets under the guidance of Professor Liscano. Given the project's established history, we have access to a valuable repository of resources, documented expertise, and a network of individuals who can assist us in our project planning.

2.1 Research from past projects

In recent years, the dynamic realms of computer science and the Internet of Things (IoT) have witnessed a notable surge in applications related to patient health monitoring, as underscored by a myriad of comprehensive research endeavors [1]. These technologies have garnered significant recognition for their inherent capacity to furnish swift, secure, and cost-effective solutions[2]. Simultaneously, this period has witnessed a remarkable increase in the exploration and experimentation surrounding smart mirrors, broadening their potential use cases to encompass sophisticated applications in vehicular and residential settings [3]. Notably, a promising yet underdeveloped avenue within this expansive landscape is the integration of smart mirror technology into the healthcare domain, particularly for predictive and real-time health monitoring.

In stark contrast to the majority of contemporary smart mirrors, often characterized by simplicity, limited functionality, or inherent constraints stemming from the processing capabilities of onboard devices, our innovative smart mirror cabinet distinguishes itself through the incorporation of a diverse range of advanced features. These encompass vocal recognition, seamless integration with external web applications, and the robust integration of an image classification machine learning model. This unique amalgamation is conspicuously evident upon an exhaustive examination of both scholarly and commercial literature. Significantly, these advanced features, frequently conspicuous by their absence in other smart mirrors, either rely on a single feature or necessitate a reliance on online services. By skillfully amalgamating these multifaceted features, our focus remains directly on addressing the distinct requirements of our user base, with a pronounced emphasis on the elderly demographic, which can frequently encounter challenges when engaging with mainstream technology [4].

From a design perspective, our smart mirror cabinet is painstakingly crafted to confront the prevalent challenges that the elderly often encounter during the assimilation of novel technologies, encompassing cognitive, auditory, haptic, and visual dimensions. The system's meticulous architectural structure adheres to a multi-interaction paradigm, thoughtfully offering an array of diverse pathways for feedback and input. This adaptable approach ensures that users with auditory impairments can depend on visual feedback, while those with visual impairments can engage effortlessly through voice communication. Furthermore, to alleviate cognitive burdens, our interface is expertly designed to be user-friendly, ensuring the presentation of information in a lucid and easily understandable manner.

2.2 Research from discussions with experts

To gain a more comprehensive understanding of the desired outcome for our final project, we initiated consultations with stakeholders who will be utilizing our completed capstone. Among these stakeholders was a nurse deeply embedded in the operations of Ontario Shores. Through these interactions, we obtained valuable insights into the challenges faced by many patients, particularly those with mild dementia, who require constant medication reminders.

Initially, we contemplated the development of an iPad application as a potential solution to address this issue. However, subsequent discussions with Professor Quevedo and Professor Liscano prompted a shift in our perspective. They illuminated the competitive landscape of the current market, leading us to reevaluate our approach. Following thorough research and a comprehensive needs assessment conducted in collaboration with the nurse, we acquired a firm grasp of the requirements and considerations necessary for the successful implementation of an active solution. By synthesizing the outcomes of these deliberations, we conceived the concept of a "smart mirror cabinet." This innovation represents an extension of the existing smart mirror technology, augmented to accommodate a smart pill box, thereby providing a holistic solution to the identified medication management challenge.

2.3 Gathering information from related applications

Smart mirrors are a fairly new technology with the rise of Artificial Intelligence and Machine Learning. Several related applications and technologies can complement or be integrated with a smart mirror. Through an exploration of these areas, we can devise the design of our user interface and user experience by drawing upon the existing work carried out in the development of other smart applications. The key to successful integration with these related applications is to ensure seamless user experiences and data security while maintaining the primary focus on medication management and health monitoring in the smart mirror.

3. Design Process

We have chosen to embrace an iterative design methodology for this project, primarily due to its adaptability and the capacity to continuously enhance our project. Given our limited prior experience in the healthcare domain, we were in search of an approach that resonates with the experimental nature of developing this application. This approach will allow us to prioritize and fine-tune our core functionalities while incorporating ongoing feedback and testing, all with the added benefit of minimizing administrative complexities compared to methodologies such as Agile or Waterfall. Although Agile exhibits certain resemblances, we excluded it from consideration due to its additional intricacies and the requirement for a higher level of expertise for effective implementation. Furthermore, we dismissed the Waterfall approach due to its inflexibility, which is less accommodating of changes. Given our student status, the comprehensive scoping out of an entire project would be a formidable task, and we foresee inevitable modifications that Waterfall is not readily adaptable to.

4. Use Cases and Scenarios

Use Cases:

Case	Description
UC-1	Medication Storage Management and Cataloging Description: The mirror stores medication in a safe, organized manner.

	Actor(s): Caregiver Preconditions: The mirror is powered on, functional, and connected to the internet. Basic Flow: 1. Caregiver opens relevant pill containers. 2. The caregiver scans medication to input the medication type corresponding to the device database, amount, and quantities. The caregiver inputs the schedule, and the compartment number into the cabinet. 3. The caregiver stores medication in the respective pill containers. 4. Repeat steps until all medication is stored or the containers are full for that designated period. 5. The caregiver closes pill containers and saves the medication input. Postconditions: Medications are securely stored, organized, and easily accessible. Exceptions: 2a. Medication Scan Fails: 1. System Displays Error Message 2. Caregiver manually inputs medication data 2b. The cabinet is not connected to the internet/can't connect to the medication database: 1. System Displays Error Message 2. Caregiver manually inputs medication data 3. Data gets synced to the cloud when the internet connection is restored
UC-2	<u>Medication Reminders</u> Description: The mirror notifies and/or reminds the patient to take their medication. Actor(s): Patient Preconditions: - Cabinet is powered on, functional, and connected to the internet. - Medication Schedules are set. - Medication is stored in the cabinet. Basic Flow: 1. It's time for medication intake. 2. Cabinet system triggers visual and auditory notifications for the patient. 3. Cabinet system displays a prompt on the screen. 4. The patient acknowledges the reminder by tapping the prompt. 5. Proceed to UC-3, Visual Medication Indication. Postconditions: The patient is reminded of medication intake and is in front of the cabinet system. Exceptions: 4a. Patient fails to acknowledge reminder within the set time frame: 1. Cabinet system notifies caregiver.
UC-3	<u>Visual Medication Indicator</u> Description: The mirror visually indicates which pill containersthe medication to be taken is in.

	Actor(s): Patient Preconditions: - Cabinet is powered on, functional, and connected to the internet. - Medication Schedules are set. - Medication is stored in the cabinet. - Patient is in front of the cabinet. Basic Flow: 1. It's time for medication intake. 2. Cabinet visually indicates which pill containersthe medication is located in. 3. The medication pill containersis opened. 4. Proceed to UC-5, Medication Intake Verification Postconditions: The patient is ready to take the medication.
UC-4	Online Caregiver Monitoring Description: The Caregiver has access to all Patient data through an online portal. Actor(s): Caregiver Preconditions: - The cabinet is powered on, functional, and connected to the internet. - Patient has a history of taking medication with the system. Basic Flow: 1. Caregiver logs into an online portal connected to the cabinet system. 2. Caregiver accesses metrics and historical data of patient medication intake and behaviour, as well as view and modify patient data and medication scheduling if needed. 3. Caregiver logs out of the online portal. Postconditions: The caregiver is better informed and the information in the cabinet system is up to date.
UC-5	Medication Intake Verification Description: The mirror ensures that the patient has successfully taken their medication. Actor(s): Patient Preconditions: - Cabinet is powered on, functional, and connected to the internet. - Patient is ready to take medication and is facing the cabinet system. Basic Flow: 1. The patient picks up one pill from one of the containers. 2. The cabinet system tracks the change in quantity of pills in the pill containers using computer vision. 3. The patient ingests the pill. 4. The cabinet system tracks the patient's hand going across their mouth using computer vision and video recording. 5. Repeat until all needed medication is consumed. Postconditions:

	- The patient has successfully ingested the medication. - The system has tracked if the patient has taken the medication. Exceptions: 4a. Patient fails to ingest and quantity of medication in pill containers is different: - Notify the caregiver of the discrepancy - Store the video of the patient ingesting the pill
UC-6	<u>Interactive Features</u> Description: The mirror exhibits interactive features, such as the weather, for Patient enjoyment and utility. Actor(s): Patient, Caregiver Preconditions: - Cabinet is powered on, functional, and connected to the internet. Basic Flow: 1. Patient prompts the cabinet system by voice for information, such as the current weather. 2. The cabinet responds via audio. 3. The patient can see the provided data/response on screen. 4. The patient can see standard data such as motivational quotes and the weather that are displayed by default. Postconditions: The patient is uplifted and better informed.
UC-7	<u>Caregiver to Patient Messaging System</u> Description: The Caregiver can communicate with the Patient through the mirror via messages. Actor(s): Patient, Caregiver Preconditions: - Cabinet is powered on, functional, and connected to the internet. Basic Flow: 1. The caregiver accesses the online portal connected to the cabinet system. 2. The caregiver can write and submit messages to be displayed to the patient, with the choice to notify the patient or not. 3. The caregiver logs out of the portal. 4. Either the patient is notified of the new message or the message is displayed the next time the patient is in front of the cabinet system. Postconditions: The caregiver has communicated with the patient.

Table 1.1: Use Cases

Use Case Diagram:

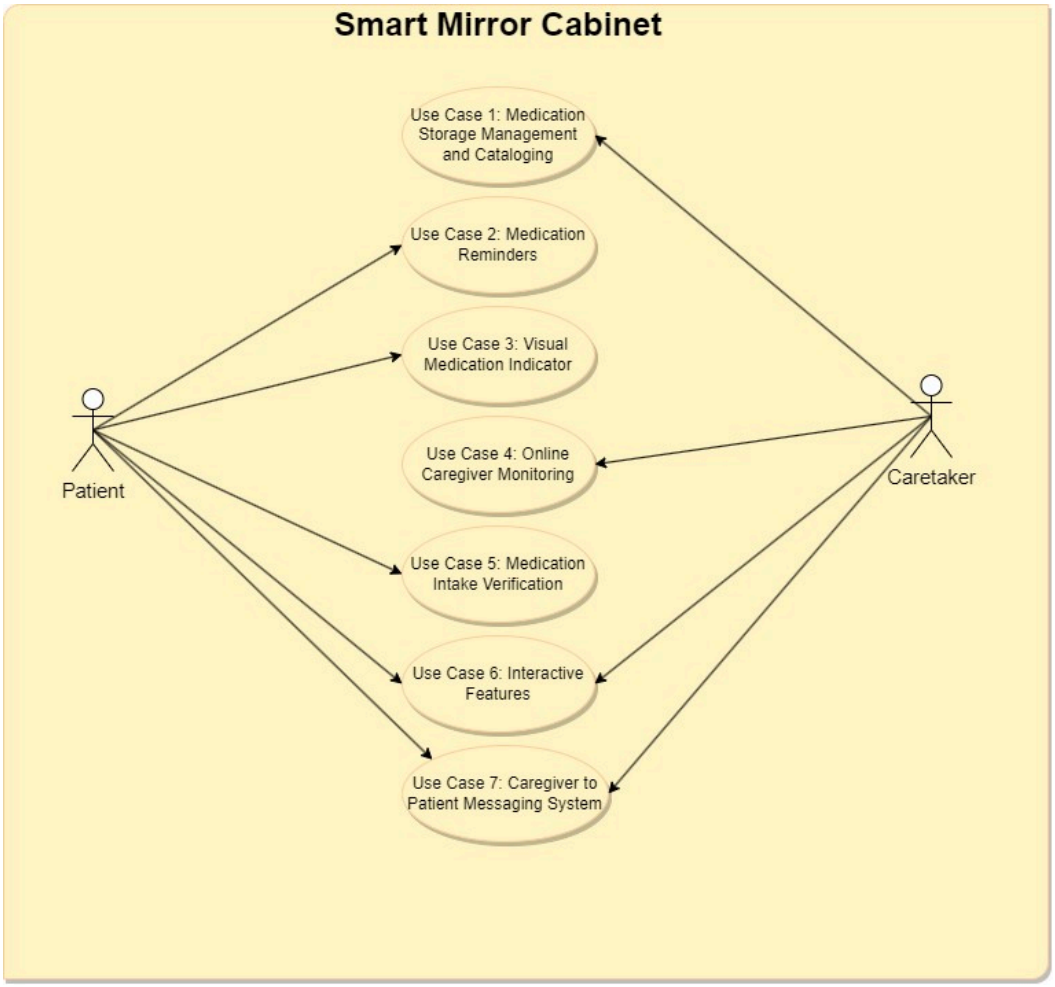


Figure 2.2: Use Case Diagram

5. Stakeholder Requirements and Traceability Matrix

Stakeholders:

C	Caretaker - Supervises the dementia patient and sets up application for them.
P	Patient - Individual who has dementia and takes medications for other health issues

Table 1.2: Stakeholders

Requirements:

Caretaker Requirements:

R1	Caretaker will be able to select and create profiles for patients

R2	Caretaker will be able to select alarm sounds
R3	The application must accurately display patient's personal information

Table 1.3: Caretaker Requirements

Patient Requirements:

R4	The application should provide clear and visual medication schedules with reminders, reducing the risk of missed doses or incorrect medications
R5	The application should play audio through the speaker
R6	Patient should be able to interact with the application by selecting mood for the day and through notification alerts

Table 1.4: Patient Requirements

Functional requirements:

F1	Application can be displayed on both smart mirror and smart watch
F2	Caretaker is able to look at multiple profile information at the same time

Table 1.5: Functional Requirements

Non-functional requirements:

NF1	The system must ensure the confidentiality of patient information and medication schedules. Access to patient data should be restricted to authorized individuals.
NF2	Medical data stored in the cloud must be encrypted during transmission and storage to protect against unauthorized access.
NF3	The system should have a high availability rate, ensuring that it is operational and accessible to caregivers and patients at all times.
NF4	The system should have a backup power source to ensure continued operation during power outages, preventing any disruption in medication management and reminders.
NF5	The system should respond to user interactions with minimal latency to ensure a seamless user experience.
NF6	The cabinet system's computer vision and video recording should be capable of real-time tracking with high accuracy and minimal lag during medication intake

	verification.
NF7	The system should be designed to accommodate an increasing number of patients and caregivers, allowing for easy scalability as the user base grows.
NF8	The system should support future enhancements and additional features, ensuring it can adapt to evolving needs and technologies.
NF9	The user interface should be intuitive and user-friendly, ensuring that caregivers and patients can easily navigate and interact with the system even with a limited technological background
NF10	The system's audio and visual notifications should be clear, user-friendly and compatible with a wide variety of cognitive capabilities, helping patients and caregivers to understand and respond to reminders effectively.
NF11	The system must be capable of integrating with external databases to update medication information and synchronize data when the internet connection is restored.
NF12	The cabinet system should have the ability to integrate with third-party devices or software for extended functionality (linking with digital health records or wearable devices).
NF13	The system should comply with relevant healthcare and data privacy regulations to ensure the protection of patient data and adherence to legal requirements.
NF14	The system should undergo regular updates to maintain compliance with changing healthcare regulations and industry standards.
NF15	The system must have robust data backup and recovery mechanisms to prevent data loss in case of system failures or data corruption.

Table 1.6: Non-Functional Requirements

Traceability Matrix:

Use Case	Stakeholders	Caretaker Requirements	Patient Requirements	Functional Requirements	Non-Functional Requirements
UC-1	Caregiver				NF11
UC-2	Patient		R4		NF11
UC-3	Patient				NF10

UC-4	Caregiver			F1, F2	NF1, NF2, NF7, NF8, NF13, NF14
UC-5	Patient				NF6, NF10
UC-6	Caregiver, Patient		R6	F1, F2	NF9, NF10, NF12
UC-7	Caregiver, Patient				NF15
Stakeholder Requirements		R1, R2, R3	R4, R5, R6	F1, F2	NF1, NF2, NF3, NF4, NF5, NF6, NF7, NF8, NF9, NF10, NF11, NF12, NF13, NF14, NF15

Table 1.7: Traceability Matrix

6. Acceptance Testing

Case	Acceptance Test
UC-1	1. Caregiver successfully opens relevant pill containers. 2. Caregiver can scan and input medication data accurately into the system. 3. Medication data, schedule, and compartment numbers are correctly stored. 4. Caregiver can successfully close pill containers and save medication input. 5. Medications are securely stored, organized, and easily accessible.
UC-2	1. Cabinet system triggers visual and auditory notifications for the patient at the correct time. 2. The system displays a prompt on the screen. 3. The patient acknowledges the reminder by tapping the prompt. 4. If the patient fails to acknowledge the reminder within the set time frame, the cabinet system should notify the caregiver.
UC-3	1. Cabinet visually indicates which pill container the medication is located in. 2. The medication pill containers open successfully.
UC-4	1. Caregiver can log into the online portal connected to the cabinet system. 2. Caregiver can access metrics and historical data of patient medication intake and behavior. 3. Caregiver can view and modify patient data and medication schedule.

	4. Caregiver can log out of the online portal. 5. The information in the cabinet system is up to date.
UC-5	1. The system tracks changes in the quantity of pills in the pill containers using computer vision. 2. The system tracks the patient's hand going across their mouth using computer vision and video recording. 3. If the patient fails to ingest medication or the quantity in the pill containers is different, the system should notify the caregiver and store the video of the patient ingesting the pill.
UC-6	1. The patient can prompt the cabinet system by voice for information. 2. The cabinet responds via audio. 3. The patient can see the provided data/response on the screen. 4. Standard data like motivational quotes and the weather are displayed correctly.
UC-7	1. The caregiver can access the online portal connected to the cabinet system. 2. The caregiver can write and submit messages to be displayed to the patient, with the choice to notify the patient or not. 3. The message is displayed correctly to the patient either immediately or the next time the patient is in front of the cabinet system.

Table 1.8: Acceptance Testing

7. Project Plan

Report R1 (Due Oct 20th, 2023):

- Commence the process of collecting prerequisites and use cases while laying the foundation for the project, utilizing the report sections as a reference point to shape the project concept.

Report R2 (Due Nov 17th, 2023):

- Initiate a comprehensive process of concept generation and development, while concurrently establishing integration tests to iteratively refine project requirements and scope. Once use cases and acceptance tests receive validation from diverse stakeholders, they can be further elaborated upon in conjunction with the technical approach.
- Generate preliminary design concepts in the form of UI mockups, accompanied by descriptive annotations for each interface component, to delve deeper into the product concept.
- Conduct an assessment of available software solutions that may enhance the overall quality and functionality of the application.

- Undertake a rigorous cost estimation for the entire project.

Team Presentation and Demo (Due Nov 27th, 2023):

- Proceed with the development of the Minimum Viable Product (MVP) intended for presentation, with a primary focus on UC-1 and UC-2.
- Solicit feedback from an array of stakeholders throughout the MVP development process to ascertain that we are crafting a product that meets usability standards and aligns with their expectations.

Team Retrospective Report (Due Dec 4th, 2023):

- Create a Report Reflecting on Previous Meetings and Communications, with a Focus on Enhancements for the Upcoming Semester

Detail Design and Integration Testing Report

- Design any algorithms and diagrams needed to make our product
- Perform the integration tests defined in Report R2
- Update project plan as needed

Acceptance Test Demonstration Report

- Perform and assess system based on acceptance tests defined in Report R1

Final Engineering Report

- Revise existing reports based on current viable product

Presentation & Demo Video

- Create a promotional video of the product
- Present product to peers

Capstone Design Annual Exhibition

- Create a poster with the project's accomplishments
- Demo fully functional smart mirror

8. Team Contribution Matrix

Name	Student Number	Contribution
Lyba Mughees	100750490	25%
Massimo Albanese	100616057	25%
Abida Choudry	100700985	25%
Hima Paul	100753261	25%

3. R2

1. Concept Generation and Analysis

1.1. Project Inputs and Architectural Drivers

1.1.1. Functional Requirements Overview

The functional requirements derived from the findings in Report 1 (Table 7.2) are crucial for decision making. The identified requirements play a pivotal role in influencing the decisions for the smart medicine cabinet. For our design, we decided to focus on UC-1, UC-2, UC-4 and UC-5 (Table 7.1). Since UC-2 and UC-5 were similar, they have been combined, and will be hereby referred to as UC-2.

Use Case	Functional Requirements Influencing Decision	Impact on Decision
UC-1	R1: Caretaker creating profiles for patients. R4: Clear and visual medication schedules with reminders.	● The ability for the caretaker to create patient profiles influences the design to ensure a personalized and organized medication storage system. ● Visual medication schedules align with the need for an organized and user-friendly interface, ensuring efficient cataloging.
UC-2 , UC-5	R2: Caretaker selecting alarm sounds. R5: Application playing audio through the speaker. R6: Patient interacting with the application through notification alerts.	● The choice of alarm sounds and audio capabilities enhances the effectiveness of medication reminders, ensuring patient engagement and timely intake. ● Patient interaction through notification alerts is essential for medication intake verification, ensuring patient engagement and system feedback.

UC-4	R3: Accurate display of patient's personal information. F2: Caretaker looking at multiple profile information simultaneously.	● Accurate patient information display is crucial for online monitoring, ensuring the caretaker has comprehensive and real-time insights. ● The ability to view multiple profiles simultaneously enhances the efficiency of caregiver monitoring.

Table 1.1.1.1: Functional Requirements Overview

The integration of these functional requirements in the design ensures a system that caters to the specific needs outlined in the identified use cases.

The use cases UC-3, UC-6, and UC-7 were discarded based on specific considerations that led to their exclusion from the final design.

Use Case	Reason Behind Discarding	Impact on Requirements
UC-3	The functionality of visually indicating the pill container for medication intake was not needed as our design opens up the corresponding pill container (UC-5).	Requirements related to visual medication indication were subsumed under UC-5, ensuring that the patient receives clear guidance about the location of the medication container.
UC-6	Since this is an existing project we are building upon, the interactive UI is already in place with modules like weather updates and the news.	Requirements associated with interactive features, voice prompts, and additional display content were discarded, aligning the system more closely with essential medication-related functions.
UC-7	The messaging system was excluded to maintain a clear scope centered around medication management. Introducing a messaging system would add complexity and potential distractions, deviating from the primary goal of the smart medicine cabinet.	Requirements related to caregiver-patient communication, message notifications, and display of messages were discarded, ensuring a streamlined focus on medication-related functionalities.

Table 1.1.1.2: Discarded Requirements Overview

In summary, the decision to discard these use cases and their associated requirements was driven by the need for a focused, efficient, and user-friendly smart medicine cabinet system. By consolidating

functionalities and excluding non-essential features, the design prioritizes core requirements related to medication storage, reminders, monitoring, and verification, contributing to a more effective and purposeful solution.

1.1.2. Quality Attributes Analysis

The integration of these functional requirements in the design ensures a system that caters to the specific needs outlined in the identified use cases. The prioritization of usability, reliability, performance efficiency, and security as quality attributes aligns with the critical aspects of medication management, user interaction, and data security in the smart medicine cabinet system.

ID	Quality Attributes	Scenario	Associated Use Case	Priority
QA-1	Usability	The web application's user-friendly design ensures easy profile creation and medication input for caregivers and straightforward medication intake for patients.	UC-1, UC-2, UC-4, UC-5	High
QA-2	Reliability	System reliability is crucial for accurate medication reminders, monitoring, and data storage. The system should consistently perform as expected to ensure patient safety.	UC-1, UC-2, UC-4, UC-5	High
QA-3	Performance	Efficient medication cataloging and reminders demand optimal system performance, preventing delays and ensuring timely notifications.	UC-1, UC-2, UC-4, UC-5	High
QA-4	Security	Patient data security is paramount, especially for online caregiver monitoring. Robust measures must be in place to protect sensitive health information.	UC-4	High

Table 1.1.2.1: Quality Attributes

1.1.3. Constraints Identification

In the process of developing a smart medicine cabinet, several constraints have been identified to guide and limit the project's scope. These constraints encompass financial, human resource, time, and technical considerations, shaping the parameters within which the project must operate. The careful management of these constraints is essential for ensuring the successful and timely completion of the project while optimizing available resources.

ID	Constraint
CON-1	The project must adhere to a predefined budget ($800 max), limiting the financial resources allocated for development, hardware, and software components.
CON-2	The project is constrained by the availability of human resources. The team size, skill sets, and availability of team members may impact the project's development timeline and scope.
CON-3	The project is subject to time limitations, with a specified deadline for completion. The project must be completed within the set time frame.
CON-4	The project's technical complexity is constrained by the knowledge and expertise of the development team. The team's familiarity with selected technologies and frameworks may influence design choices.

Table 1.1.3.1: Constraints

1.1.4. Key Architectural Drivers

The architecture must strike a balance among usability, reliability, performance, and security requirements while adhering to budgetary and resource constraints. Strategic decisions in technology selection, scalability, and security measures are crucial to the success of out smart medicine cabinet within the specified constraints.

1. Usability (QA-1): The emphasis on usability in requirements, especially for profile creation and medication input, necessitates an intuitive and user-friendly interface. The architecture must support a responsive and easily navigable web application accessible across various devices.
2. Reliability (QA-2): The reliability of the system is critical for accurate medication reminders and monitoring. The architecture must incorporate redundancy, fault tolerance, and robust error handling to ensure consistent and dependable performance.
3. Performance (QA-3): Efficient medication cataloging and timely reminders demand optimal system performance. The architecture needs to be designed for scalability, efficient data storage, and quick retrieval to prevent delays and ensure timely notifications.
4. Security (QA-4): Given the sensitivity of patient health information, security is paramount, especially for online caregiver monitoring. The architecture must implement robust security

measures, including data encryption, access controls, and secure data transmission, to protect patient privacy.

5. Budget Constraint (CON-1): The predefined budget constraint influences architectural decisions related to hardware and software components. The architecture must prioritize cost-effective solutions without compromising functionality or performance.
6. Human Resource Constraints (CON-2): Limited human resources may impact the project's development timeline and scope. The architecture must be designed with considerations for resource availability, ensuring that the system can be developed and maintained with the available team size and skill sets.
7. Technical Expertise (CON-4): The technical complexity constraint based on the team's expertise influences technology and framework choices. The architecture must align with the team's familiarity with selected technologies, ensuring a smoother development process and minimizing the learning curve.

1.2. Initial Architectural Concepts

1.2.1. Exploration of Architectural Options

The design decisions for the smart medicine cabinet project were carefully considered to align with key goals and requirements. The following table outlines the rationale behind the chosen design elements and the alternatives that were discarded, along with the reasons for their exclusion.

<table>
<tr><th>Design Decisions & Locations</th><th>Rationale</th></tr>
<tr><td>Web Application</td><td>The choice of a web application aligns with the goal of making the smart medicine cabinet user-friendly and widely accessible. This approach eliminates the need for users to install dedicated applications on their devices, streamlining the user experience and making it more convenient for both patients and caregivers.
<table>
<tr><th>Discarded Alternatives</th><th>Reason for Exclusion</th></tr>
<tr><td>Native Mobile Applications</td><td>We excluded native mobile applications for the smart medicine cabinet to avoid installation challenges and the resource-intensive development of separate applications for iOS and Android. Opting for a web-based solution ensures a standardized user experience accessible through any web browser.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr>
<td>Raspberry Pi with Monitor</td>
<td>The rationale for using a Raspberry Pi in the smart medicine cabinet project is rooted in its cost-effectiveness, compact form factor, and versatility. The Raspberry Pi serves as a reliable and affordable computing platform that can efficiently manage the cabinet's functionalities, including processing medication data, running the user interface, and handling connectivity, making it an ideal choice for this embedded system.
<table>
<tr><th>Discarded Alternatives</th><th>Reason for Exclusion</th></tr>
<tr><td>iPad/Apple Watch</td><td>iPads and Apple Watches were excluded due to cost considerations and flexibility. Apple Watches can be relatively expensive, ranging from $300 to $1000, while the Raspberry Pi and monitor combination is more cost-effective. Additionally, the Raspberry Pi offers greater control over the system's functionalities as a standalone computing device.</td></tr>
</table>
</td>
</tr>
</table>

Table 1.2.1.1: Architectural Options

These design decisions aim to prioritize user accessibility, cost-effectiveness, and efficient system control, contributing to the development of a more inclusive and functional smart medicine cabinet solution.

1.2.2. High-Level System Architecture

Given our decision to continue the development of the system given to us, a high-level system architecture can be put together based on the existing and planned hardware components. These hardware components will each be associated with respective software modules and interfaces. The directionality of the arrows in the following figures show the logical flow of data between each component.

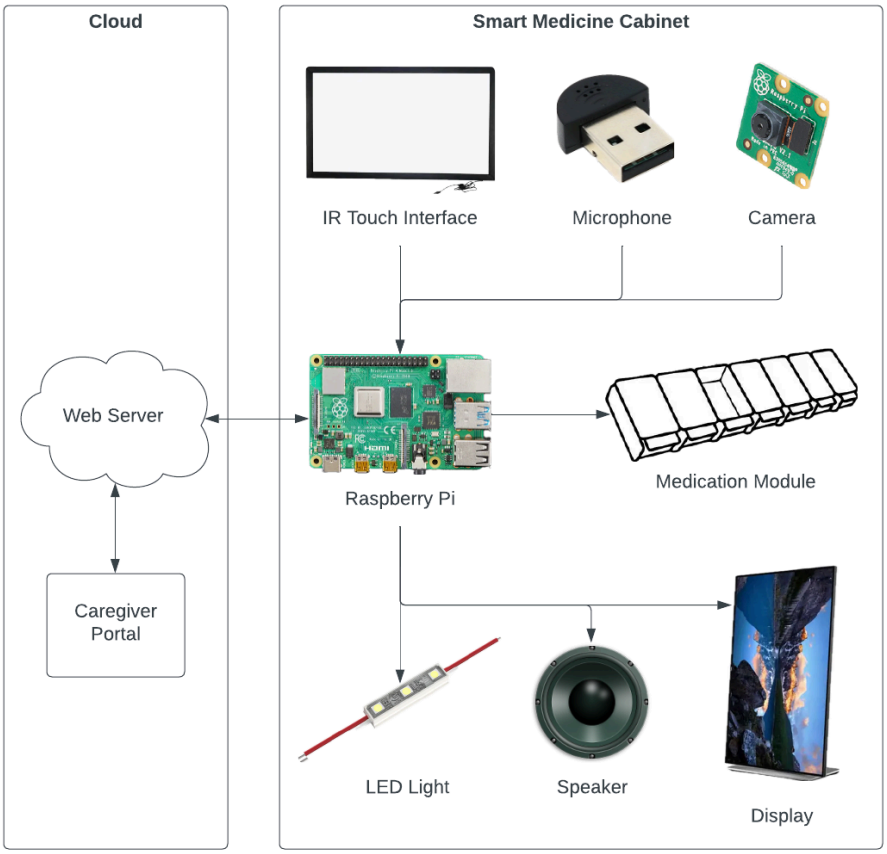


Figure 1.2.2.1: Preliminary Hardware Layout

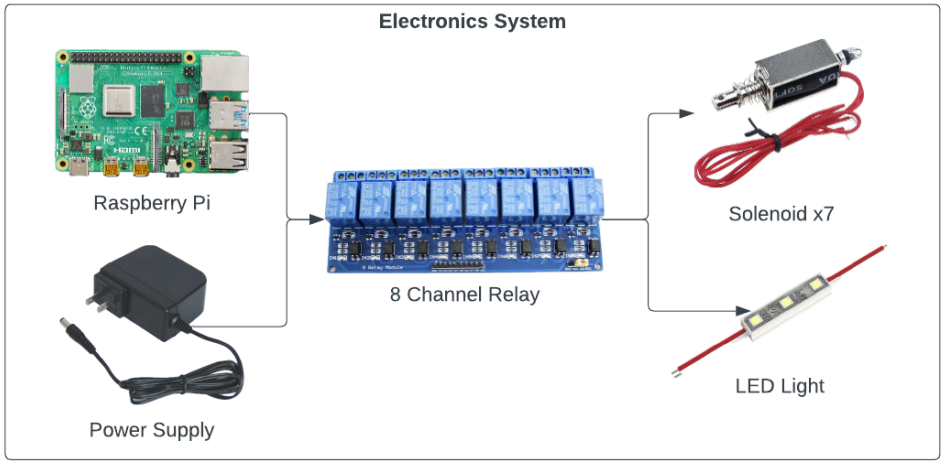


Figure 1.2.2.2: System Electronics Layout

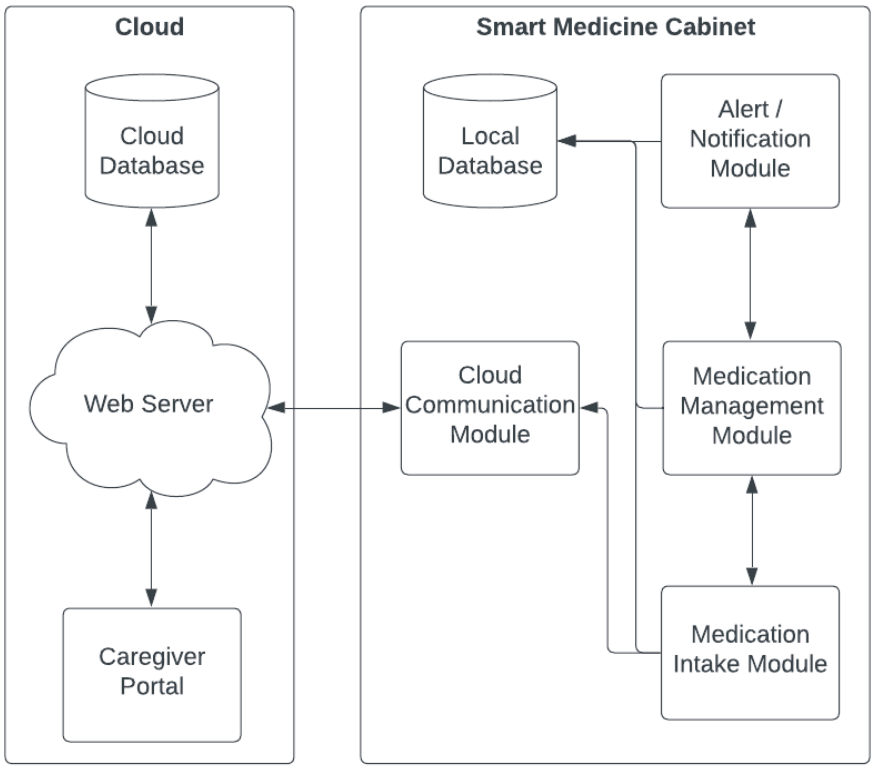


Figure 1.2.2.3: Preliminary Architecture Diagram

1.3. Preliminary Design of Responsibilities and Interfaces

1.3.1. Distribution of System Responsibilities

From the Preliminary System Layout, we can create software components based on our system's hardware components, and assign them responsibilities based on their associated use cases. Hardware components such as the Raspberry Pi, IR Touch Interface, and Display, are considered necessary for the base system and all use cases to function. We will focus only on accessory hardware components.

Component	**UCS**	**Responsibilities**
Medication Management Module	UC-1	- Handle Input Data from Caregiver - Scan Medications with the Camera - Save data to local database
Alert/Notification module	UC-2	- Keep track of medication timings - Use the LED Light and the Speaker to notify the patient of medication intake times
Medication Intake Module	UC-3, UC-5	- Verify the medication intake by patients - IR Touch Interface (for patient interaction), Camera (for verification), Raspberry Pi (to manage intake process)

Local Database	UC-1, UC-2, UC-5, UC-4	- Store and manage local data related to medication schedules, patient profiles and intake records - Raspberry Pi to host local database
Web Server	UC-1, UC-2, UC-5	- Host and manage the web-based application interface for user interaction
Caregiver Portal	UC-1, UC-4	- Provide a secure platform for caregivers to access patient data, schedules, and make necessary adjustments
Cloud Database	UC-1, UC-4	- Store and synchronize data securely over the cloud, enabling access to information from multiple locations

Table 1.3.1.1: Software Components, Associated Use Cases, and Responsibilities

1.3.2. Initial Components Interface Design

The interaction between components and modules within the architecture of the smart medicine cabinet system is crucial for its seamless operation. Below is an overview of how different elements will interact:

1. Web Application (User Interface): Interaction with Users (Caregiver and Patient): The web application serves as the primary user interface accessible through various devices. It allows caregivers to create profiles, input medication data, and monitor patient information. Patients interact with the application for medication reminders..
2. Raspberry Pi (Embedded System): Control of Cabinet Functions: The Raspberry Pi acts as the embedded system controlling the smart medicine cabinet. It manages the hardware, interfaces with sensors, and controls the opening and closing of pill containers based on medication schedules.
3. Database (Cloud and Local): Data Storage: The system utilizes a database to store patient profiles, medication schedules, and historical data. Cloud storage enables synchronization and backup, ensuring data persistence even in the absence of an internet connection.
4. Computer Vision Module: Medication Intake Verification: The computer vision module is responsible for tracking medication intake. It interfaces with cameras to monitor the patient's actions, such as picking up pills and ingesting them, ensuring accurate medication verification.
5. Audio Module: Alarm Sounds and Audio Alerts: The audio module generates alarm sounds for medication reminders. It also supports audio alerts for patient interaction, such as acknowledging reminders. This module interfaces with the Raspberry Pi to synchronize audio cues with visual prompts.
6. Notification Module: Patient and Caregiver Notifications: This module handles notifications for both patients and caregivers. It sends reminders to patients and notifies caregivers in case of patient non-compliance. The notification module interfaces with the web application and the Raspberry Pi.

7. Online Portal (Caregiver Monitoring): Remote Monitoring: The online portal allows caregivers to remotely monitor patient data. It interfaces with the database to retrieve real-time metrics, historical data, and facilitates modifications to patient profiles and medication schedules.
8. Cloud Connectivity Module: Synchronization and Updates: The cloud connectivity module enables synchronization between the local database and the cloud when an internet connection is available. It ensures that the system is up-to-date and that data is backed up securely.

In summary, the interaction between these components and modules is orchestrated to provide a comprehensive and reliable smart medicine cabinet system that meets user requirements while adhering to constraints and quality attributes. The collaboration between the web application, embedded system, databases, and specialized modules ensures a cohesive and effective user experience.

1.4. Feasibility and Risk Assessment

1.4.1. Conceptual Design Feasibility

The feasibility of the smart medicine cabinet design appears practical within the defined constraints and requirements. Budget limitations drive cost-effective hardware and software choices, leveraging a Raspberry Pi and a web-based application to ensure efficiency without compromising functionality. The team's skills and time constraints influence the design direction, focusing on leveraging existing modules to expedite development.

The design aligns closely with critical functional requirements, emphasizing personalized medication storage, visual schedules, and accurate patient information display. Prioritizing usability, reliability, performance, and security as quality attributes ensures the design addresses essential aspects of medication management and caregiver monitoring effectively.

Architectural choices, such as utilizing a web-based application and a cost-effective Raspberry Pi while excluding native mobile applications due to potential complexities, contribute to a streamlined and feasible approach. Overall, the design demonstrates practicality and promise within constraints, aligning functionalities, quality attributes, and architectural decisions with the project's objectives.

1.4.2. Potential Risks and Challenges

Technical Risks:

1. Hardware/Software Compatibility: Integrating diverse hardware components and ensuring their compatibility with the software could pose challenges, leading to technical glitches or system malfunctions.
2. Complex System Integration: The complexity of integrating various functionalities, such as medication storage, reminders, and caregiver monitoring, might lead to integration difficulties and require extensive testing and debugging.
3. Data Security Concerns: Ensuring robust data encryption and protection against cyber threats is crucial, considering the sensitive nature of patient information stored within the system.

4. Reliability of Sensors and Technology: Dependence on sensors for medication verification and system responsiveness might face reliability issues, impacting the system's accuracy and performance.

Operational Risks:

1. User Adoption and Training: The user interface's complexity might hinder user adoption, requiring comprehensive training for both caregivers and patients to ensure efficient usage.
2. Maintenance and Upkeep: The need for regular maintenance and software updates to ensure system functionality and security could pose operational challenges, especially for non-technical users.
3. Scalability Issues: Scaling the system to accommodate additional users or functionalities might encounter operational bottlenecks, affecting its effectiveness in larger settings or increased demand.

Managerial Challenges:

1. Resource Allocation: Limited financial and human resources could influence project schedules and goals, requiring efficient resource management to prevent delays or compromised project outcomes.
2. Project Expansion: Shifting or broadening project requirements beyond the original scope might escalate expenses, prolong timelines, and divert attention from vital features, affecting project focus and effectiveness.
3. Stakeholder Engagement: Inadequate or ambiguous engagement with stakeholders, including healthcare providers, caregivers, and patients, might lead to misinterpretations or misaligned expectations, potentially impacting project direction and successful execution.

2. Conceptual System Design

2.1. Refinement of Architectural Elements

2.1.1. Component and Module Design

Detailed design of system software components and modules, describing functions and interactions

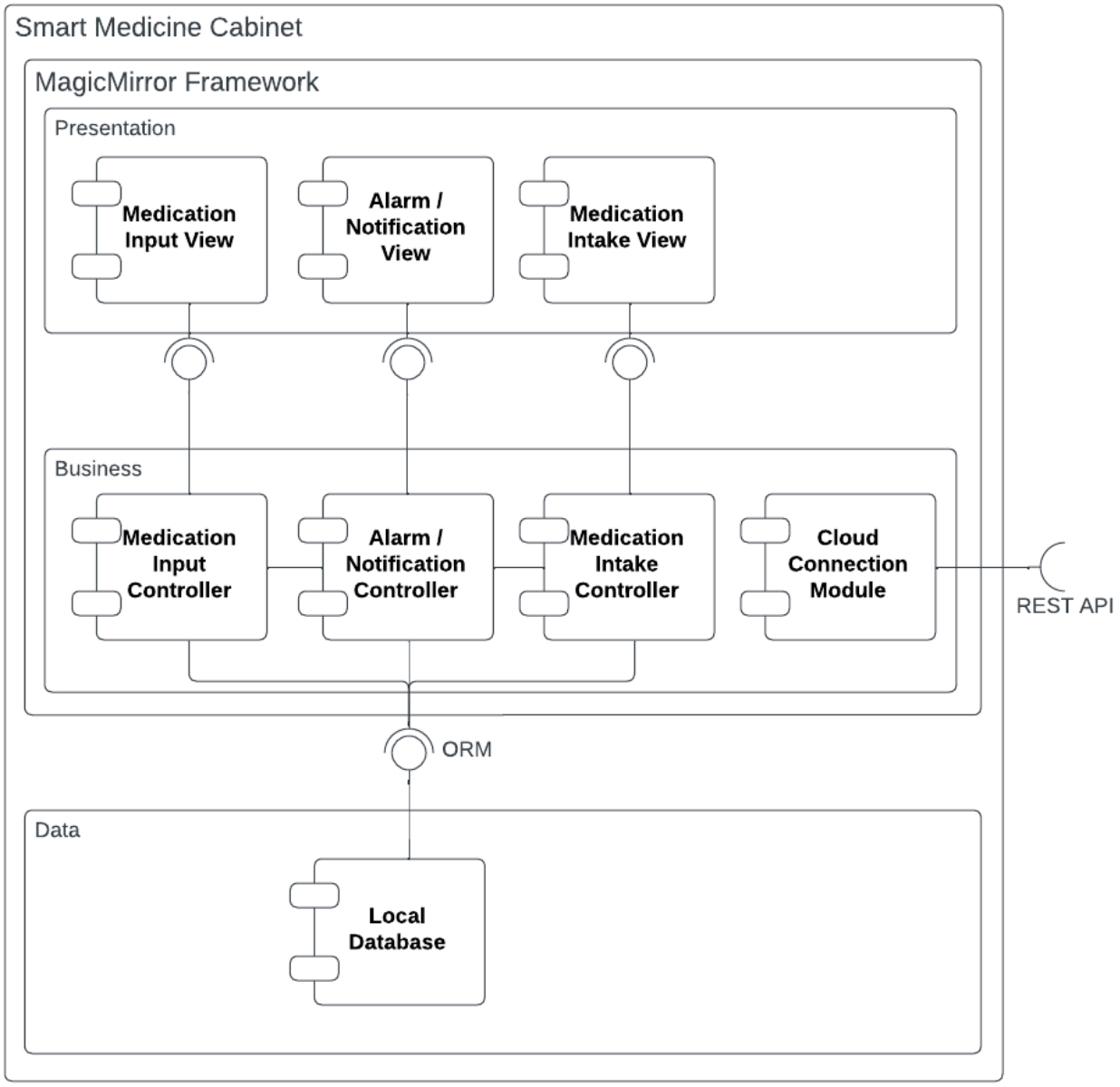


Figure 2.1.1.1: Smart Medicine Cabinet Architecture Diagram

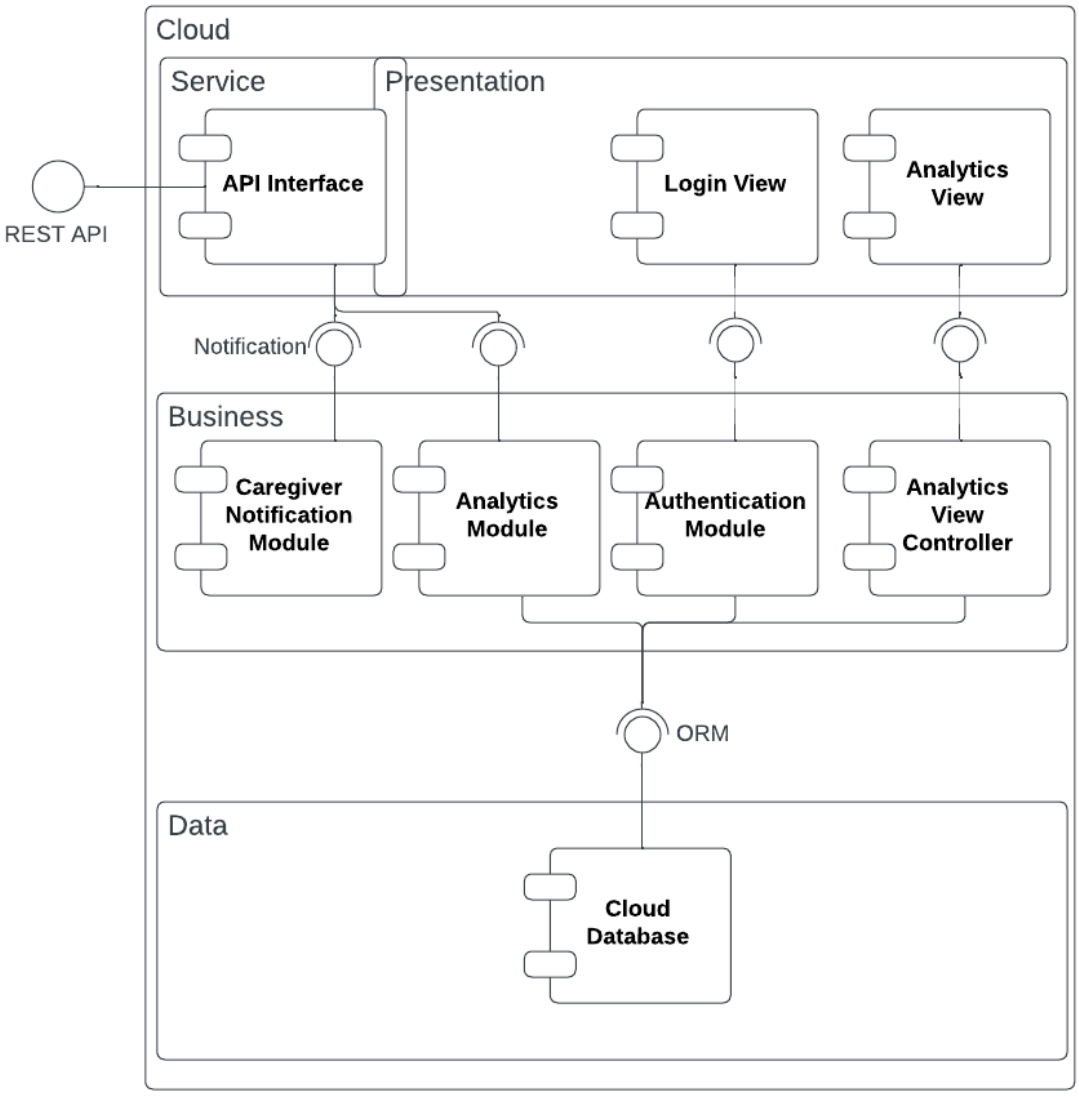


Figure 2.1.1.2: Cloud Server Architecture Diagram

2.1.2. Detailed Responsibility Allocation

Components	UCS	Responsibilities
Presentation layer	UC-1, UC-2, UC-5	Focuses on the user interface and presentation of information to caregivers and patients, ensuring user-friendly interactions and intuitive views for medication-related tasks.
Business layer	UC-1, UC-2, UC-5, UC-4	Handles the logic and processing of user inputs and system interactions, orchestrating the flow of data between the presentation layer and the data layer while implementing business rules and logic.

Data Layer	UC-1, UC-2, UC-5, UC-4	Manages the storage and retrieval of data, ensuring the persistence and security of medication-related information within the smart medicine cabinet system.
Medication Input View	UC-1	● Provides an interface for caregivers to input medication-related information such as medication schedules, dosage, and patient profiles. ● Allows caregivers to conveniently manage and update medication data for different patients, facilitating personalized medication scheduling.
Alarm/notification view	UC-2, UC-5	● Displays medication reminders and notifications for patients, including alarm settings and visual alerts ● Notifies patients about scheduled medication intake times, ensuring timely intake and adherence to the medication schedule
Medication Intake View	UC-2, UC-5	● Offers a user interface for patients to confirm medication intake and interact with the smart cabinet system during medication consumption ● Allows patients to verify and acknowledge medication intake, providing feedback to the system about adherence
Medication Input Controller	UC-1	● Manages user interactions and processes related to medication input, including validating and processing caregiver-entered medication information ● Orchestrates the flow of data between the Medication Input View and Data Layer, ensuring accurate and secure storage of medication details
Alarm/Notification Controller	UC-2, UC-5	● Controls the scheduling and triggering of medication reminders and notifications for patients based on their specific medication schedules. ● Coordinates with the Alarm/Notification View to display timely alerts to patients, facilitating adherence to medication schedules
Medication Intake Controller	UC-2, UC-5	● Manages interactions related to medication intake, including receiving and processing patient verification of medication consumption. ● Facilitates patient interactions through the Medication Intake View, capturing intake verification and updating relevant data in the system.
Cloud Connection Module	UC-4	● Handles communication between the smart medicine cabinet system and a cloud-based service via a RESTful API ● Enables secure and efficient exchange of data between the

		smart cabinet and external cloud services, facilitating features like remote monitoring and data synchronization
Local Database	UC-1, UC-2, UC-5 UC-4	• Stores and manages all locally relevant data related to medication schedules, patient profiles, intake verification, and system configurations • Provides a secure and reliable storage solution for all pertinent information within the smart medicine cabinet, ensuring quick access and retrieval of data
Login View	UC-1	• Provides a user interface for authentication, allowing users to log into the system securely • Validates user credentials and grants access to authorized users
Analytics View	UC-4	• Displays medication intake statistics, schedules, and personalized analytics for caregivers or administrators • Presents visualized data regarding medication adherence, schedule management, and other relevant analytics for informed decision-making
Caregiver Notification Module	UC-2, UC-5	• Manages and sends notifications/alerts to caregivers about medication schedules, cabinet status, or any relevant updates. • Ensures timely notifications to caregivers for improved medication management and cabinet monitoring
Analytics Module	UC-4	• Analyzes medication intake data, generating insights into patient adherence and schedule adherence • Processes medication-related data to provide meaningful analytics for better decision-making regarding patient care
Authentication Module	UC-1	• Handles user authentication processes, ensuring secure access to the Smart Medicine Cabinet system • Verifies user credentials and authorizes access to the system, maintaining security protocols
Analytics View Controller	UC-4	• Manages the functionalities of the Analytics View, retrieving and presenting medication-related analytics to authorized users • Coordinates the display of analytics data for caregivers aiding in decision-making regarding patient care and medication management
Service Layer	UC-1, UC-2, UC-5, UC-4	Acts as an intermediary between the Presentation, Business, and Data Layers, providing an interface for external communication and managing the API interactions while orchestrating the flow of data and functionalities within the cloud-based system

Table 2.1.2.1: Detailed Responsibility Allocation

2.1.3. Interface Specification

System Components:

1. Web Application:

Purpose: Interface for caregivers and patients.

Interface: HTTP/HTTPS for web-based access.

Data Formats: JSON for data exchange.

Communication Mechanism: RESTful APIs for CRUD operations.

2. Raspberry Pi:

Purpose: System control and hardware interface.

Interface: GPIO pins for hardware integration.

Data Formats: Interfacing with sensors and devices (e.g., camera, microphone).

Communication Mechanism: Python libraries for hardware control.

3. IR Touch Interface:

Purpose: Patient interaction.

Interface: Touchscreen interface for user input.

Data Formats: Coordinates and touch events.

Communication Mechanism: Direct interaction with the web application through the Raspberry Pi.

4. Cloud Database:

Purpose: Storage and synchronization of data.

Interface: Cloud API (e.g., AWS, Azure).

Data Formats: Encrypted data transmission (JSON or binary).

Communication Mechanism: HTTPS for secure communication.

5. Local Database:

Purpose: Storage of local system data.

Interface: SQLite or similar database system.

Data Formats: Structured database format (e.g., SQL).

Communication Mechanism: Local storage and retrieval through Python or similar.

Communication Protocols:

Web Application to Raspberry Pi: HTTP(S) requests and responses.

Raspberry Pi to Hardware Components: GPIO or hardware-specific communication protocols.

Cloud Database to Web Application: RESTful API calls over HTTPS.

Local Database to Raspberry Pi: Python libraries for database interaction.

IR Touch Interface to Web Application: Direct interaction via touchscreen events.

Data Formats:

JSON: For API requests and responses between the web application and databases.

Structured Data (SQL): For local database storage and retrieval.

Coordinates/Events: For IR Touch Interface interactions.

Binary (encrypted): For secure data transmission between the cloud and local databases.

Communication Mechanisms:

Web-Based Interface: Accessible via web browsers using HTTPS.

Hardware Integration: GPIO pins, sensors, and devices connected to the Raspberry Pi.

Cloud Integration: Secure API calls using HTTPS for data synchronization.

2.2. Architectural Design and Documentation

2.2.1. Detailed Architectural Views

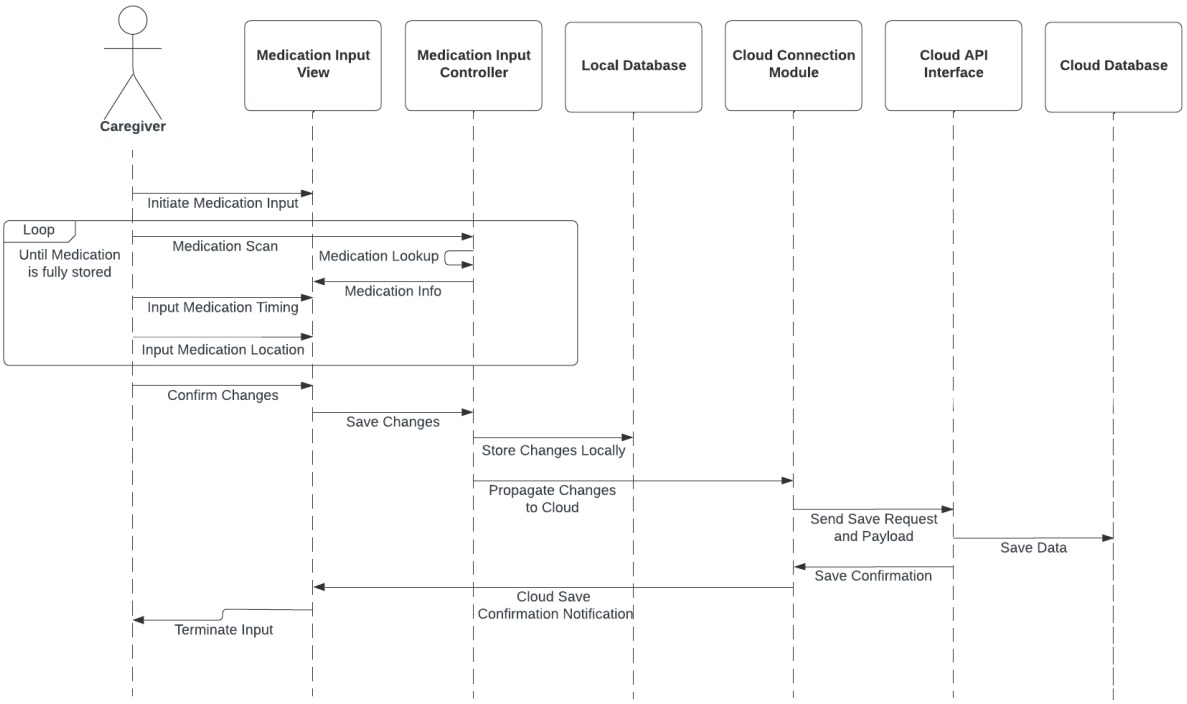


Figure 2.2.1.1: Medication Input Sequence Diagram

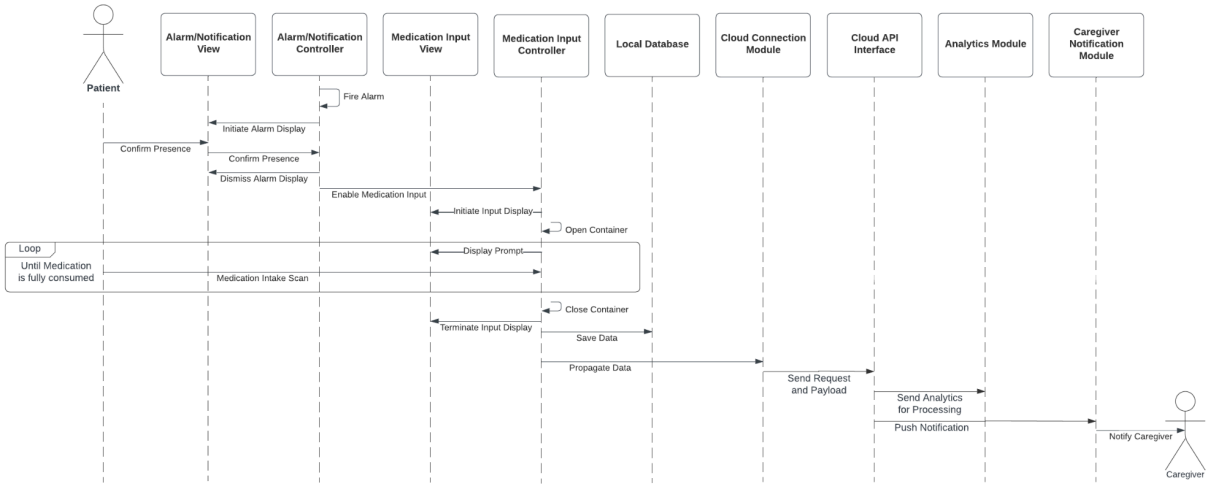


Figure 2.2.1.2: Medication Intake Sequence Diagram

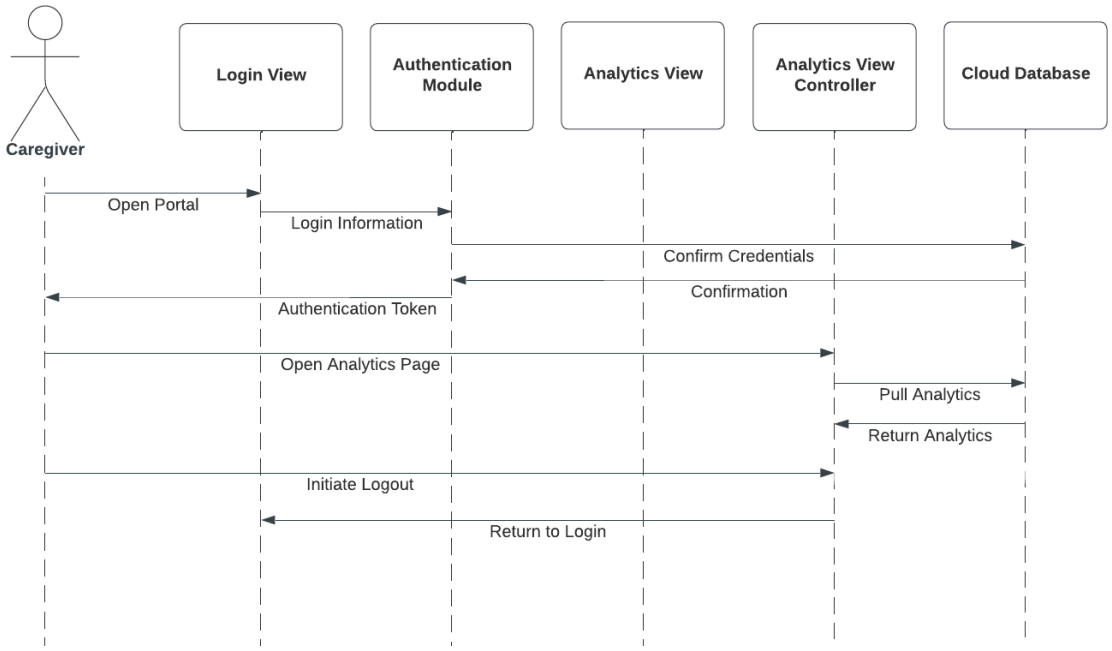


Figure 2.2.1.3: View Analytics Sequence Diagram

2.2.2. Design Decisions and Rationale

Design Decisions	**Rationale**
Cloud Database Integration (AWS/Azure) over Bare Metal	Accessibility: A web application ensures universal access without platform dependencies, enabling users to interact with the system via any web-enabled device. Standardization: Avoiding native apps simplifies maintenance, updates, and reduces the need for separate developments for iOS and Android platforms. Ease of Use: Provides a consistent user experience across devices without the need for installations, enhancing user accessibility and ease of adoption.
MagicMirror Framework over Custom Interface	Customization: MagicMirror framework offers extensive customization options for displaying information beyond medication-related data, promoting user engagement. Modularity: The framework's modular design enables easy integration of new features, facilitating potential future expansions or integrations. Community Support: Leveraging an established framework like MagicMirror ensures access to a supportive community and existing plugins/modules for rapid development.

Use of RESTful APIs for System Communication:	Interoperability: RESTful APIs enable seamless interaction between system components, allowing diverse hardware and software elements to communicate effectively. Scalability: APIs facilitate future integrations with other systems or devices, supporting the system's scalability and adaptability. Simplicity and Standardization: RESTful APIs offer a straightforward and standardized approach to data exchange, promoting system reliability and ease of maintenance.
Integration of Local and Cloud Databases:	Data Redundancy: Local databases ensure data availability even in the absence of internet connectivity, enhancing system reliability. Cloud Synchronization: Cloud databases enable secure data synchronization, allowing access from multiple locations and ensuring real-time data updates.

Table 2.2.2.1: Design Decisions

3. Definition of Integration Tests

3.1. Testing

Purpose: Integration tests aim to validate the interaction and collaboration among different system components—both hardware and software—ensuring they function coherently as a unified system.

Scope: These tests cover various aspects:

Hardware-Software Interactions: Verifying communication between hardware modules (e.g., Raspberry Pi, IR Touch Interface, Camera) and software components (e.g., web-based application, local databases).

Functionality Integration: Ensuring that functionalities such as medication management, reminders, and user interactions work seamlessly together.

Data Flow and Synchronization: Testing the flow of data between local and cloud databases, verifying synchronization and consistency.

User Interface and System Responsiveness: Assessing the responsiveness and usability of the user interface in response to hardware inputs and software triggers.

Specific Integration Test Scenarios:

1. Hardware-Software Interaction Test:

Objective: Verify the communication between hardware components (e.g., IR Touch Interface, Camera) and the software modules responsible for managing these interactions.

Scenario: Simulate hardware inputs triggering software responses (e.g., opening medication containers through touch, verifying medication intake via camera).

Validation: Ensure that hardware-triggered actions are accurately captured and processed by the corresponding software modules.

2. Functionality Integration Test:

Objective: Validate the cohesive functioning of various system functionalities (e.g., medication storage, reminders, patient interaction).

Scenario: Simulate a full cycle of medication intake, from caregiver input to patient acknowledgment, ensuring all related functionalities work together flawlessly.

Validation: Confirm that interdependent functionalities operate cohesively without disruptions or inconsistencies.

3. Data Flow and Synchronization Test:

Objective: Validate data synchronization between local and cloud databases.

Scenario: Introduce changes in the local database and verify synchronization with the cloud database.

Validation: Ensure that updates made in the local database are accurately reflected and synchronized in the cloud database in real-time.

4. User Interface Responsiveness Test:

Objective: Assess the responsiveness and usability of the user interface in coordination with hardware inputs.

Scenario: Simulate various user interactions (e.g., medication schedule adjustments, acknowledgment of reminders) and verify prompt system responses.

Validation: Confirm that user interactions trigger appropriate system responses without delays or errors.

Validation Metrics and Criteria:

Accuracy: Ensure that data transmitted between components is accurate and consistent.

Timeliness: Assess the system's response time to hardware triggers and user inputs.

Reliability: Verify that the integrated system functions reliably under varying conditions and usage scenarios.

Consistency: Confirm that data synchronization between local and cloud databases is consistent and up-to-date.

4. Estimated Cost

4.1. Bill of Materials

Component	Image	Price	Notes
Raspberry Pi 4 Model B		N/A - Provided	
27” 1920x1080p Display		N/A - Provided	
Mini USB Microphone		N/A - Provided	May need wired microphone
IR Touch Frame		N/A - Provided	
Camera		N/A - Provided	May need second camera
Speaker		N/A - Provided	May need better speaker

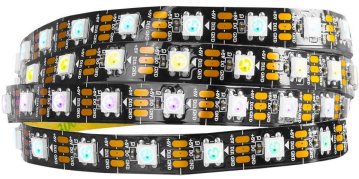
8 Channel 5V Relay		$10.99	
5V Power Supply		$28.99	
5V Push/Pull Solenoids		$19.35 x7	
Addressable RGB Strip		$16.99	
Breadboard + Wires		$18.99	
3D Printing Costs		TBD	
Wood Frame		TBD	
Subtotal		**$211.41**	Before Tax

Table 4.1.1: Bill of Materials

4.2. Manpower

A team comprising four members convenes weekly for a duration of three hours to advance the project. These sessions encompass discussions on project-related matters, feedback collection, research, implementation, and report-related activities. The COCOMO model is a valuable tool for gauging the anticipated effort in person-months and determining the overall development timeline in months. Our

capstone project falls under the semi-detached category, given that team members seek additional expertise and guidance, particularly in the integration of software and hardware components [5].

Semi-Detached constants: a = 3.0, b = 1.12, c = 2.5, d = 0.35 [5]

$E = a(KLOC)^b = 3(4)^{1.12}$
E= 14.17 persons-months

$time = c(Effort)^d = 2.5(14.17)^{0.35}$
time= 6 months

5. Updated Project Plan

5.1. Project Plan

Task Name	Details	Start Date	End Date	Total Days
Report 2	● Initiate a comprehensive process of concept generation and development, while concurrently establishing integration tests to iteratively refine project requirements and scope. Once use cases and acceptance tests receive validation from diverse stakeholders, they can be further elaborated upon in conjunction with the technical approach. ● Generate preliminary design concepts in the form of UI mockups, accompanied by descriptive annotations for each interface component, to delve deeper into the product concept. ● Conduct an assessment of available software solutions that may enhance the overall quality and functionality of the application. ● Undertake a rigorous cost estimation for the entire	10/20/2023	11/17/2023	28 days

	project.			
Build Application	• Proceed with the development of the Minimum Viable Product (MVP)	11/18/2023	11/26/2023	8 days
Build Web Server	• Develop the server alongside the application	11/18/2023	11/26/2023	8 days
Presentation + Demo	• Create presentation for current prototype • Solicit feedback from an array of stakeholders development process to ascertain that we are crafting a product that meets usability standards and aligns with their expectations.	11/26/2023	11/27/2023	1 day
Build Hardware	• 3D print the pill container • Add the pill container to the cabinet	1/4/2024	1/20/2024	16 days
Integrate hardware with software	• Connect the developed hardware components with the software.	1/21/2024	2/10/2024	20 days
Detail Design and Integration Testing Report	• Design any algorithms and diagrams needed to make our product • Perform the integration tests defined in Report R2 • Update project plan as needed	2/11/2024	2/20/2024	9 days
Acceptance Test Demonstration Report	• Perform and assess system based on acceptance tests defined in Report R1	2/21/2024	2/22/2024	1 day
Final Engineering Report	• Revise existing reports based on current viable product	2/23/2024	3/5/2024	11 days

Presentation & Demo Video	• Create a promotional video of the product • Present product to peers	TBA	TBA	
Capstone Design Annual Exhibition	• Create a poster with the project's accomplishments • Demo fully functional product	TBA	TBA	

Table 5.1: Project Plan

5.2. Gantt Chart

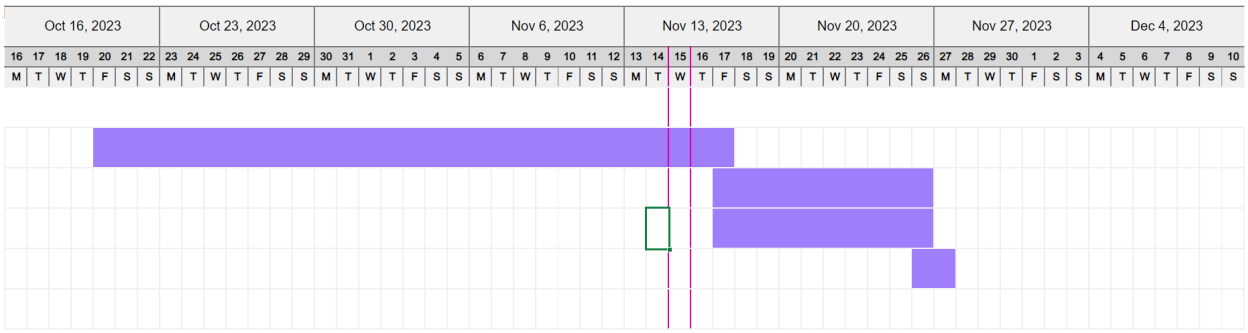


Figure 5.2.1: Gantt Chart For Semester 1

6. Contribution Matrix

Name	Student Number	Contribution
Lyba Mughees	100750490	25%
Massimo Albanese	100616057	25%
Abida Choudry	100700985	25%

Hima Paul	100753261	25%

4. Final revised design report from R#3 and R#4

Detailed Design

a. Hardware Components

i. Electronics

The final iteration of the Smart Cabinet's electronics systems is depicted in the following diagram. The Raspberry Pi is the central computer component of the entire project, where our MagicMirror software is running. The accessory devices, such as the speaker, camera, and IR touch interface, are all connected to and powered directly from the Pi, either via USB, direct connection, or GPIO pins.

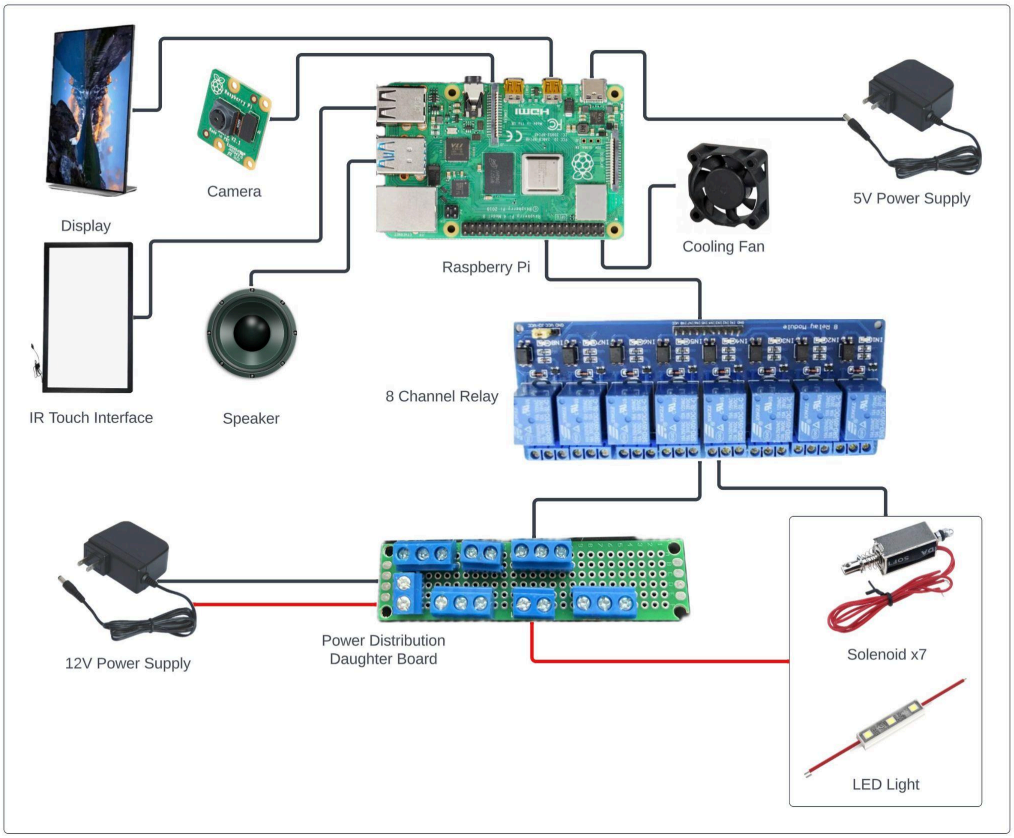


Figure 1.1.1.1 Detailed Electronics Systems Diagram

A two-voltage system (5V and 12V) was chosen for multiple reasons. The 5V supplied directly from the Pi was more than enough to power the USB accessory devices and the control of the Relay

Board, but the Pi couldn't provide enough current to properly power 6V versions of the solenoids or the LED light strip, especially at the same time. In addition, the subsequent power spike from switching on either of these components could pose a risk to the Pi's circuits. To reduce current and use thinner cabling, we settled on a 12V accessory power supply for the solenoids and the LED light strip since these parts were readily available. The relay serves as the interface between these two voltage systems, where the GPIO pins on the Pi actuate the relays on the board, closing the relevant circuit and powering the connected solenoid or light strip. There is also a 21.6V power supply, which is used exclusively to power the display.

Circuitry components like the Raspberry Pi, the Relay Board, and the Power Distribution board needed to be enclosed, preferably together in a box. This led to the design of the Electronics Box. This allowed all connections to be routed to one central location and protected the boards and delicate circuitry connecting them. They could also all be actively cooled with the fan from the Pi. Not directly shown in the assembled version is the Power Distribution Board, which is below a shelf with an array of barrel plugs, the green and black cylinders at the bottom of the box in Figure 1.1.1.2. These plugs were a last-second addition, enabling quick connection and disconnection of the solenoids/Medication Dispensers, in case of disassembly.

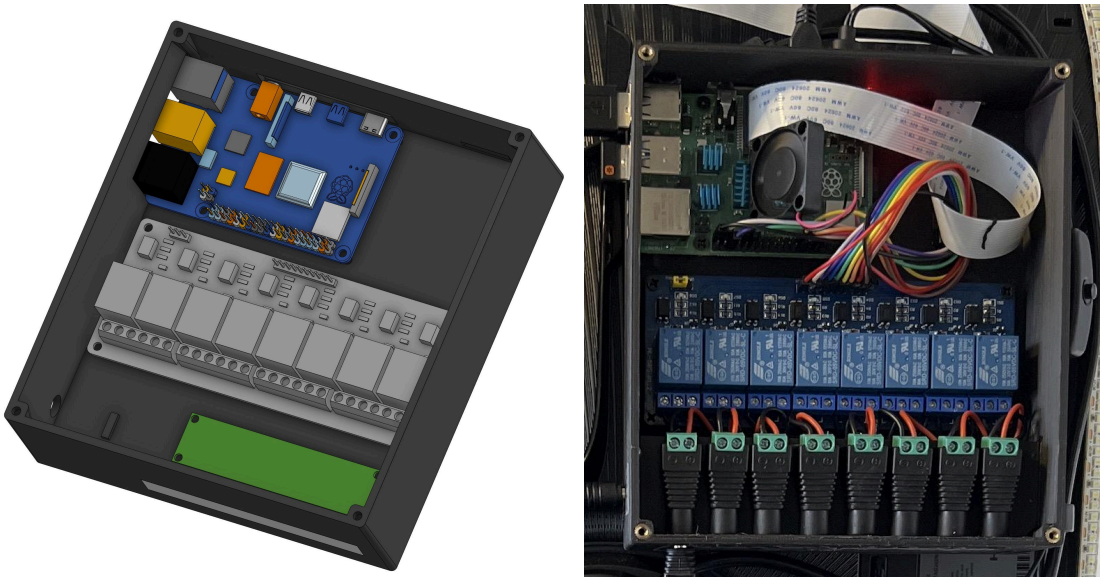


Figure 1.1.1.2 Electronics Box Design and Assembly (Enclosed version in figure 1.1.3.3)

ii. Medication Trays/Dispensers

In our original proposal, our system was designed to have 7 individual containers with lids that automatically opened and closed depending on the day and medication to be taken at that scheduled time. These lids were to be actuated by solenoids, which turned out to be a design oversight. Upon further research and testing, when these solenoids are kept powered for long periods (>1 minute), they get noticeably hot and potentially lose their magnetism. The originally selected 5N and replacement 25N solenoids both drew about 20W of power. This left us with two options; either we adapt the design to use a different means of actuation, or we pivot to a previously considered design. The only other means of

actuation would be some sort of motor, ideally linear actuators, which proved to be either too expensive or no suitable commercially available products were found. We ended up pivoting to having 7 medication dispensers, which while proving more mechanically complex, would preserve the use case. This decision will reduce the original design of (up to) 10 different medications stored for a patient, but considering this late design change, we deemed it an acceptable compromise.

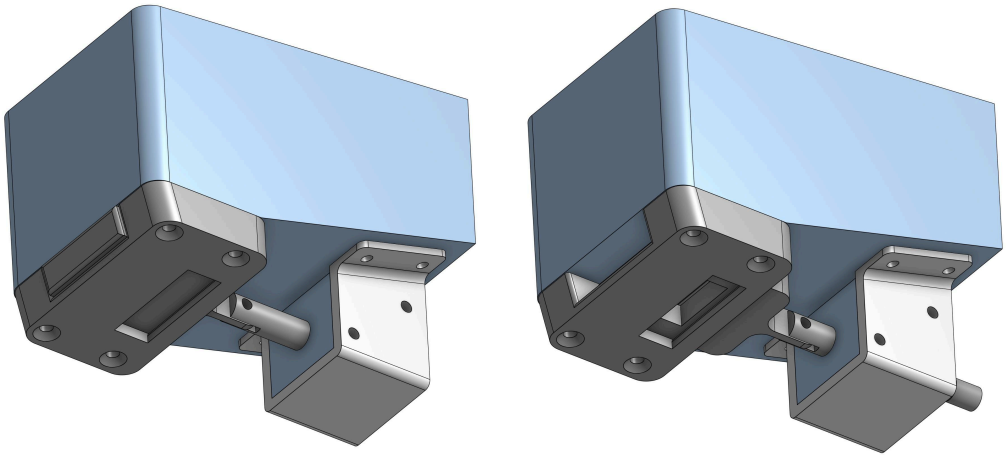


Figure 1.1.2.1. Medication Dispenser in Closed and Open Positions

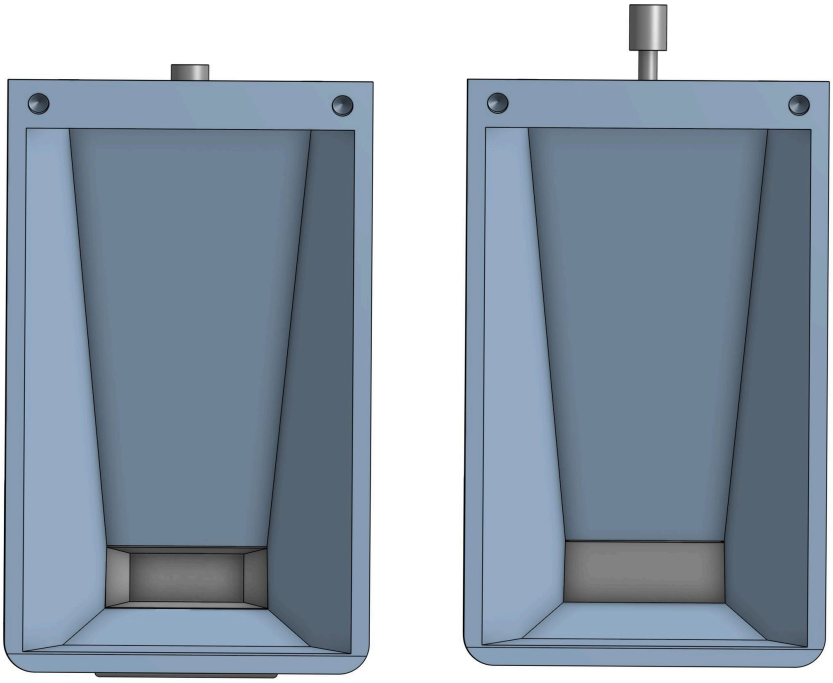


Figure 1.1.2.2. Medication Dispenser in Closed and Open Positions (Top View)

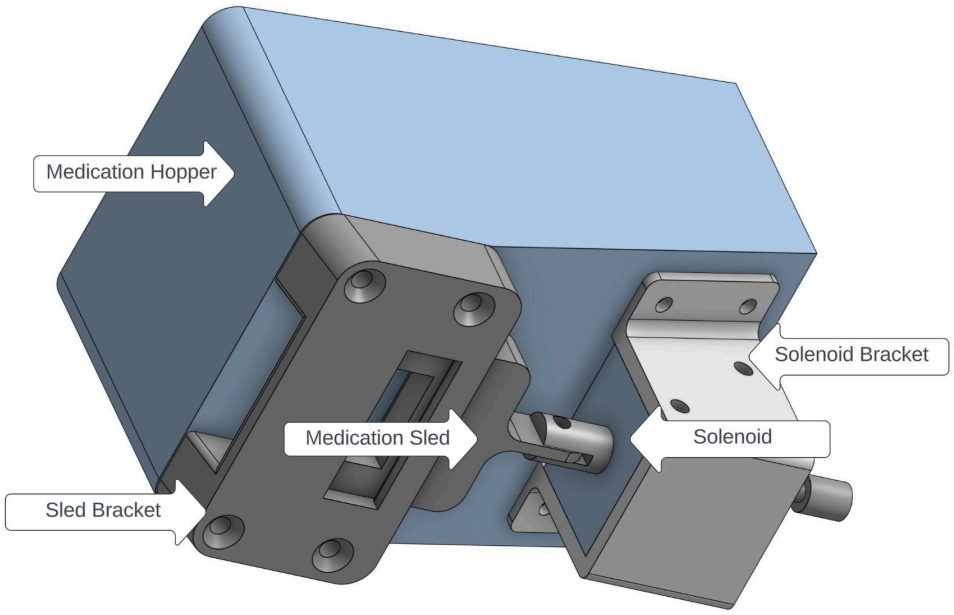


Figure 1.1.2.3. Annotated Medication Dispenser

These figures portray the most recent version of a single Medication Dispenser. The Hopper stores the medication and feeds it into the Sled. The Sled has a cutout (seen in the top view) that can contain one unit of medication, the dimensions of which can be changed for different medications. When the solenoid is actuated, it pulls the Sled back, exposing the cutout to the hole on the underside of the Sled Bracket, releasing the medication contained in the Sled. A sort of tray that captures dispensed medications is currently being designed. This assembly has been printed and tested with empty size 1 gelatin capsules, which exposed some design issues: if the capsules are poured into the hopper, there is no mechanism to prevent the capsules from being fed vertically into the sled, which jams the mechanism. Multiple variations of the Sled and Sled bracket are currently being tested to remedy the issue and work with different medication shapes. Assuming all the capsules were fed into the hopper horizontally, the mechanism feeds as expected and dispenses a pill per actuation of the solenoid.

iii. Assembly

The original smart mirror assembly provided to us consisted of a 27” display mounted on its original stand, with a thin piece of reflective plastic not covering the entire display taped on with electric tape, the IR touch frame mounted to the display with adhesive craft foam, and the front part of a wooden picture frame taped onto that. The rest of the electronic systems were left loose and not mounted to the display. This necessitated some major modifications for the creation of a properly assembled device.

We first started with the overall design of the smart cabinet. Building around the provided monitor, we modeled and designed components such as the medication trays, camera housing, and electronics box, giving us a general layout to work with. The camera housing and electronics box were 3D printed and assembled.

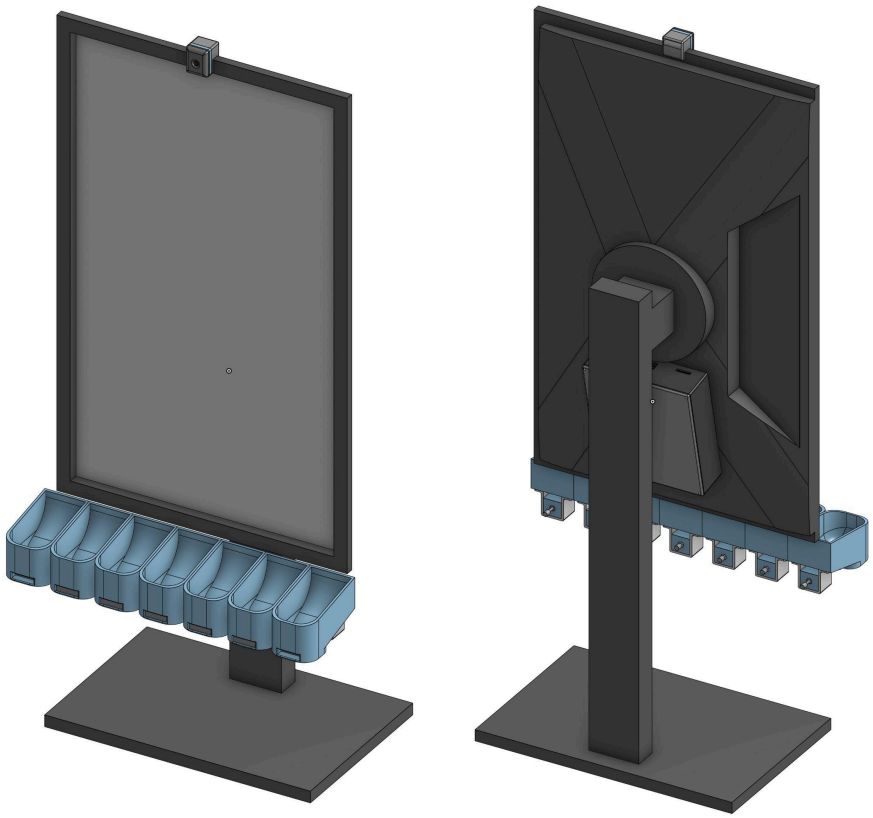


Figure 1.1.3.1. Modeled Design and Layout

To address the ‘mirror’ aspect of our project, we ordered a ¼” sheet of semi-reflective acrylic, which we then cut to fit seamlessly on the face of the display. The thickness was nearly identical to the bottom lip of the display, which enabled us to have a flush fit for the IR frame that was to be mounted onto the acrylic. There was a feature on the bottom lip that required some vigorous sanding to have it completely level. Once that was complete, the acrylic was attached to the display’s bezels with VHB (Very High Bond) mounting tape, with the IR frame attached to the acrylic in the same fashion.

The original stand of the display proved to be too short when the monitor was in a portrait orientation, especially considering the space needed below the display to house the medication storage trays. This required the replacement of the stand for one which is shown in the following images.

Figure 1.1.3.2. Display with Acrylic and IR Touch Frame with old stand (left) and new stand (right)

We wanted the unit to be powered with just one power cable, and all necessary components to be fixed onto the unit. Considering the sheer amount of electronics needed for the cabinet to function (elaborated on in the next section), and the equivalent amount of cables required to support them, strategic cable management was necessary. The Electronics Box, Power Supplies, and Power Lead are all attached via 3M DualLock Fastener strips, allowing the removal and reattachment if reconfiguration is needed.

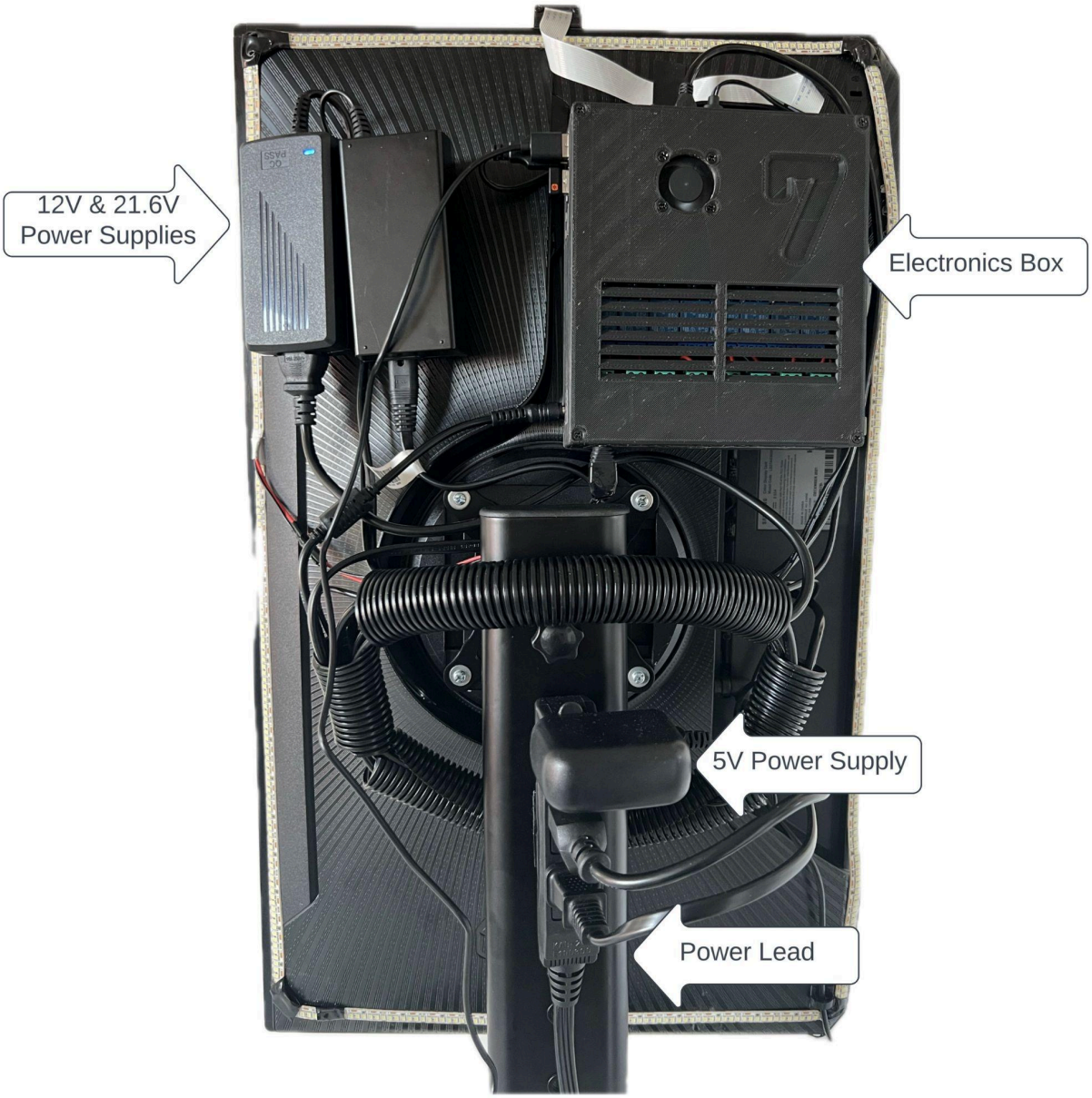


Figure 1.1.3.3. Photo showcasing the cable management on the rear of the unit

The LED light strip procured for visual notifications of the medication alarm was originally intended to be facing the patient, on the same face as the mirror, for maximum visibility. This was changed to lining the rear perimeter of the display, as the light strip was powerful enough that the light bouncing off the wall behind the cabinet in a deployment scenario was more than adequate to notify the patient, it was more aesthetically pleasing, and the light strip on the front could have proved to be too bright for some patients.

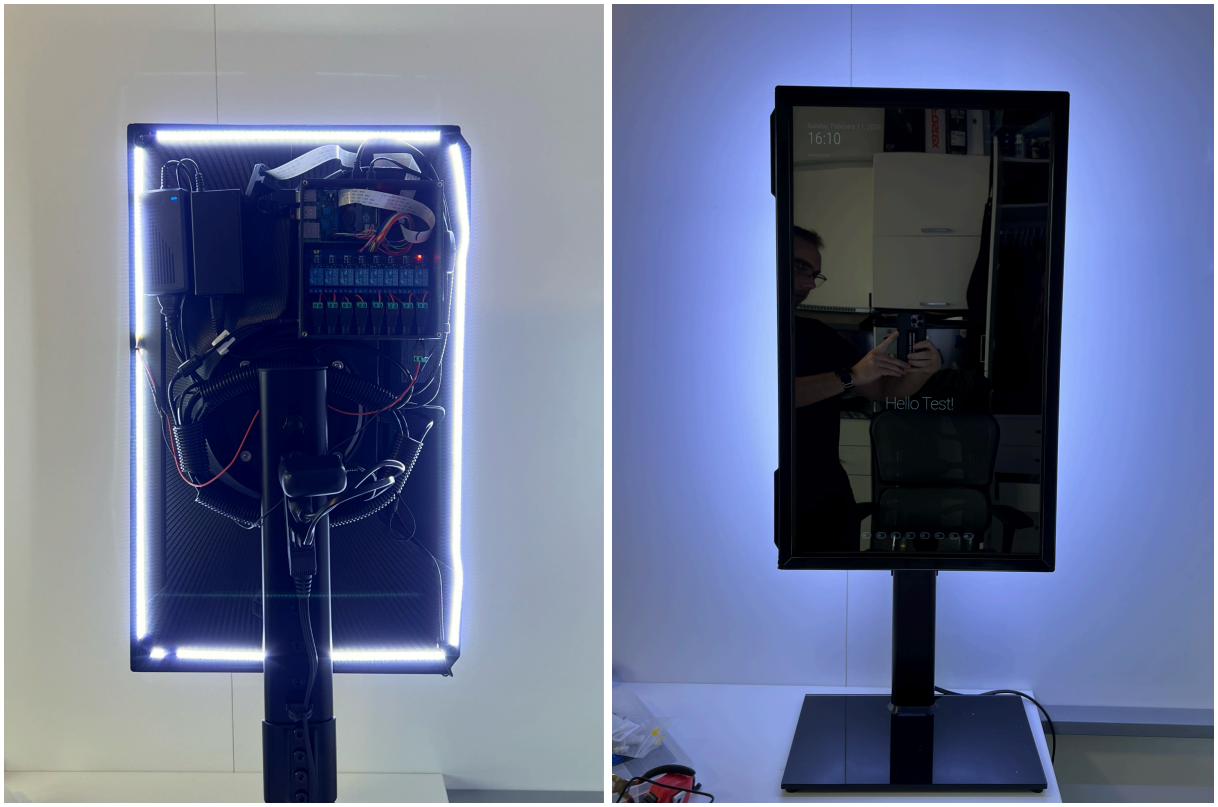


Figure 1.1.3.4. LED light strip illuminated on the rear of the unit, and the light bouncing off the wall

b. Software Components

i. **Cabinet**

ii. Layers

In the software design part for the smart medicine cabinet, we used a framework called “Magic Mirror”. This framework allowed us to build our product using different modules. These modules communicate with each other via notifications. With these modules' help, we can also control the hardware components. Our original design for the software is described with three layers; Presentation, Business, and Data.

The presentation layer consists of these modules. These modules are responsible for rendering all sorts of information, especially the medication management functionalities. Through this presentation layer, the caretaker can input medication information like schedules. The patient can also interact with the system to manage their medication intake effectively.

The business layer is responsible for implementing the system’s core functionalities. In this part of our application, our data is processed, and the interaction between the cloud occurs. There are many services present at this layer that handle specific aspects related to medication management.

The data layer is responsible for managing and storing the data related to medication management. This layer serves as the interface between the business logic and the data storage processes. Locally the cabinet has an SQLite database that stores the schedule for the patient. This data is also sent to the cloud along with the patient's medication list which is stored on the cloud as well. The data layer communicates with the cloud using REST API.

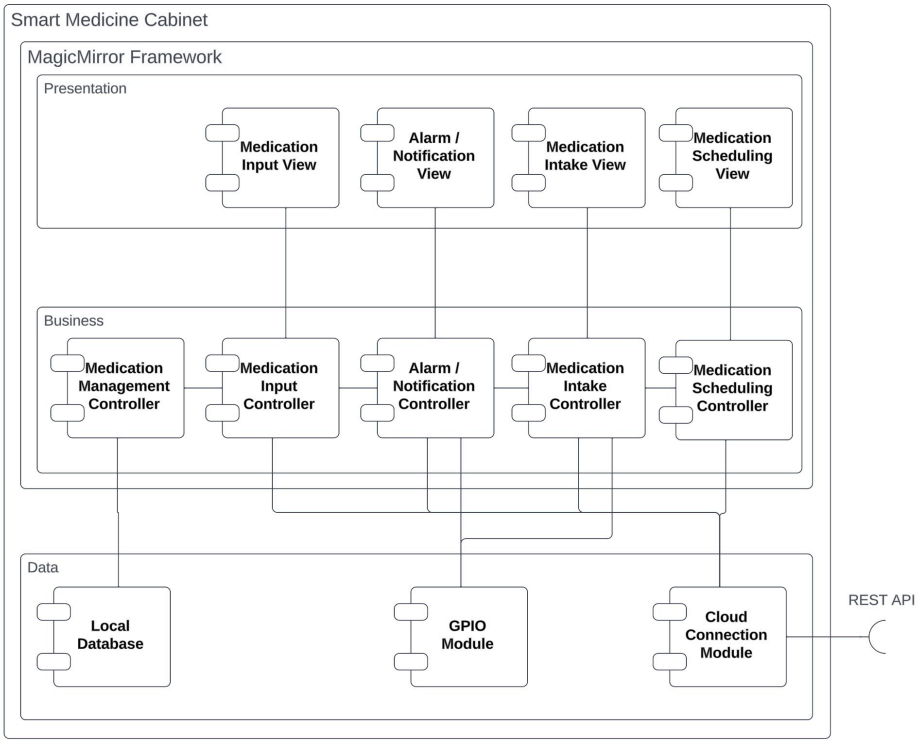


Figure 1.2.2.1. Smart Medicine Cabinet Architecture Diagram

iii. Modules

The Medication-Input module was the first module that we built for our cabinet. This module is the interface that allows a caretaker to input medication information. There are two parts in this module: a Node-Helper (the backend) and the Magic Mirror module (the frontend). The Node Helper handles operations such as saving medication data to the SQLite database and processing any notifications that are sent from the front end. When the helper receives the medication data from the front end, the information is processed and saved in the patient_medications table in the SQLite database. It verifies that the pillbox and medication ID are unique before inserting them into the database.

On the front end, there is an interface that allows the caretaker to input medication details. These details include searching the cloud for available medications and then assigning a quantity and pill box to the selected medication. This information is then sent to the Node Helper through a socket notification. This information is also sent to the cloud via a notification. Overall this module is the bridge between the Magic Mirror interface and the backend of medication management.

A similar module to the Medication-Input module that was built was the Medication-Scheduler module. Like the other module, this module also has two parts; Node Helper and the Magic Mirror module. This module is responsible for creating the schedules of medication intake for the patient. On the backend, the Node Helper listens for notifications. Once it receives the notification SCHEDULE_MEDICATION, it retrieves the medication details from the module and inserts the scheduling information into the database table medication_schedule. Similarly, the DELETE_SCHEDULE notification triggers the helper to remove the existing medication schedule. Additionally, it interacts with the patient_medications table to fetch medication options available based on the previous inputs in Medication-Intake.

The front end of the module presents the user interface to allow the caretaker to input the patient's medication schedule. It consists of the input fields for the medication name, days, and times along with the schedule and delete buttons. This information is then passed to the Node Helper. Moreover, these medication schedules are pushed to the cloud for synchronization. Overall the Medication-Scheduler module facilitates the scheduling of medication for the patient.

The last module that was built for the cabinet was the Medication-Alarm module. This module is responsible for triggering the notifications on the days and times the medication is scheduled. The node helper is responsible for using the node-schedule package to schedule medication checks every minute. The node helper uses the method "checkAndTriggerMedicationNotifications" which retrieves the medications scheduled for the current day and time from the database. The data in the table is filtered based on the current day and triggers notifications for each medication scheduled. This notification includes the details for the medication name as well as the time to take. This information is passed to the front-end display along with an auditory notification to alert the patient.

On the front end, there is a user interface created to display the medication alert to the patient. There is also a stop button to allow the patient to stop the alarm and acknowledge their medication intake. Once the stop button is clicked, the notification is deleted the the alarm sound is stopped. This also starts the medication verification process.

iv. Cloud

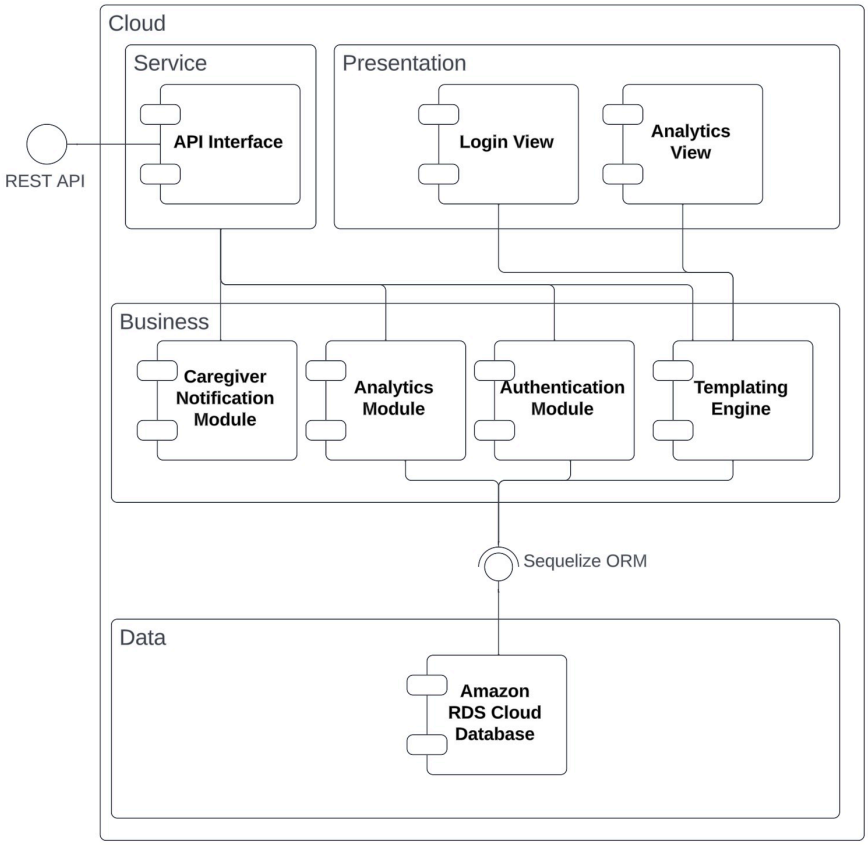


Figure 1.2.4.1. Cloud Server Architecture Diagram

Here in the cloud, the Service listens for incoming requests via a REST API, which defines a set of operations such as fetching patient data like medication information which includes the quantity of the pills, names of the medications along medication schedules. The API Interface processes incoming requests, converting them into commands that the application can recognize and respond to. Cloud services enable the synchronization of data, making sure that medication inventories and timetables saved locally are mirrored in the cloud. This way it guarantees that information can be retrieved and modified from anywhere which improves the system's adaptability.

In the login view, the caretaker can log in with their assigned credentials such as email address and password to login to gain access to the portal. The assigned caretaker will have administrator access to make changes in the caretaker portal where they can update any patient's information related to patient personal information, medication management, medication scheduling, and analytics. On the other hand, the analytics view displays statistics related to how long the patient takes to respond to the alarm for different medications and ingestion time per medication as well.

The Amazon RDS (Relational Database System serves as the primary storage for all our applications data, including caretaker, patient profiles and other relational data like scheduling and medications. This way, caretakers have access to up-to-date and accurate information. Sequelzie ORM

acts as the bridge between the caretaker portal and the relational database, eliminating the need to write raw SQL queries. By adding this layer of abstraction, interactions become more straightforward and intuitive, significantly reducing the likelihood of errors. The ORM tool makes dealing with databases much simpler letting us manage data through an object-oriented mindset instead. Additionally, Amazon RDS is crucial for handling data within the application. It enables various operations like adding, updating, and removing records based on user actions. For example, whenever a caretaker modifies a medication schedule or adds new medication details, these updates are promptly captured and saved in the Amazon RDS database. This process keeps the application up-to-date, ensuring its reliability and making the caretaker portal more efficient. On the other hand, our templating engine allows us to dynamically generate HTML pages by incorporating data into the HTML templates on the server side before displaying it on the client side. This way it allows us to customize data such as patient profiles or medication schedules according to the data stored in the database.

v. Computer Vision

1. Python Script

This Python script uses the MediaPipe library to track hand movements through a webcam.

1. Library Imports and Setup: The script imports necessary libraries, like OpenCV (‘cv2’) for video processing and MediaPipe (‘mediapipe’) for hand tracking. It starts the MediaPipe hand tracking module with specific configurations like detection and tracking confidence thresholds, to ensure accurate hand tracking.

2. Video Capture and Processing Loop: The script uses OpenCV's ‘VideoCapture’ class to capture frames from the default camera in real time. Each frame is processed for hand detection and tracking using the MediaPipe module.

3. Hand Detection and Action Recognition: When a hand is detected in frame, the script analyzes its characteristics. By measuring distances between specific landmarks (e.g., thumb tip and index fingertip), it determines the hand's proximity to the mouth, relevant for medication intake.

4. Display and User Interaction: Processed frames and hand landmarks are displayed to the user in real-time using OpenCV's ‘imshow’ function, which allows for instant visual feedback on hand tracking. The script waits for key press events for users to interact with the application. Pressing the 'q' key exits the program properly.

5. Resource Cleanup: When exiting the loop, the script then releases the camera resources and closes any OpenCV windows, to ensure proper cleanup and resource management.

2. Module

This module verifies medication intake and communicates relevant data with external systems.

1. Module Initialization and Configuration: When called, the ‘start’ method will log a message to the console, starting the Medication Verification module.

2. Notification Handling: The ‘notificationReceived’ method is the starting point for receiving notifications from other modules. When the module receives a notification "START_MEDICATION_VERIFICATION", it triggers the verification process by sending a socket notification to the helper module.

3. Socket Notification Handling: The ‘socketNotificationReceived’ method manages incoming socket notifications from the helper module or external sources. When it receives a notification "VERIFY_MEDICATION_RESULT", the module logs the payload data to the console for debugging, prepares the medication data extracted from the payload for integration with cloud systems, sends the medication data to the cloud for logging and additional processing, and logs the medication data locally for reference and auditing purposes. It also handles notifications "CLOUD_PUSH_SESSION_RESULT" by logging the received payload data.

4. Logging Utility: The ‘log’ method facilitates logging by sending log data as socket notifications to other modules.

c. Rationale for Software Design Change

During the design of the project, there were slight modifications we had to make. The first design change we made was the modification of UC-1 (Medication Storage Management and Cataloging). Originally, the caretaker was going to have the ability to scan the medication to input it into the device database. After further research, it was discovered that doing this with our system would lead to other complications. The only way to scan a medication would be based on a barcode but a barcode is not unique to a medication. Usually, barcodes are used by stores to keep track of inventory and stock so a single medicine can have multiple barcodes. This has little to no effect on our use case as the caretaker is still able to manually search for medicine in the FDA database and save it locally and to the cloud.

The next use case that was modified was UC-5 (Medication Intake Verification). When designing the use cases based on the requirements presented to us, we initially thought to use computer vision to track the quantity of the pills in the containers. Due to the solenoids not being able to open the containers at an angle higher than 90 degrees, we decided to omit this from our application. We instead decided to implement a pill dispenser as discussed in the hardware section.

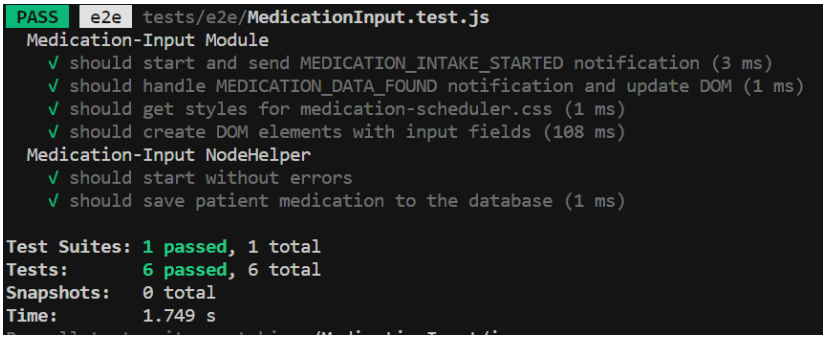
Unit and Integration Testing

d. Unit Testing

i. Modules

- Medication-Input Module: Creates a patient's personal medication table as well as sets the pillbox number and quantity of each of the patient's medication

1. Start and send MEDICATION_INTAKE_STARTED notification
2. Handle MEDICATION_DATA_FOUND notification and update DOM
3. Get the styles from medication-scheduler.css
4. Create DOM elements with input fields
5. The helper should start without any errors
6. Patient medication is saved in the database



- Medication-Scheduler Module: Creates a schedule for the medications the patient has to take

1. Start and send MEDICATION_SCHEDULER_STARTED notification
2. Handle MEDICATION_DATA_FOUND notification and update DOM
3. Get styles for medication-scheduler.css
4. Create DOM elements with input fields
5. Helper starts without errors
6. Handle SCHEDULE_MEDICATION notification
7. Handle DELETE_SCHEDULE notification
8. Handle MEDICATION_ALARM_TEST notification

```
PASS  e2e  tests/e2e/MedicationScheduler.test.js
  Medication-Scheduler Module
    √ should start and send MEDICATION_SCHEDULER_STARTED notification (9 ms)
    √ should handle MEDICATION_DATA_FOUND notification and update DOM (2 ms)
    √ should get styles for medication-scheduler.css (1 ms)
    √ should create DOM elements with input fields (69 ms)
  Medication-Scheduler NodeHelper
    √ should start without errors (1 ms)
    √ should handle SCHEDULE_MEDICATION notification (63 ms)
    √ should handle DELETE_SCHEDULE notification
    √ should handle MEDICATION_ALARM_TEST notification

Test Suites: 1 passed, 1 total
Tests:       8 passed, 8 total
Snapshots:   0 total
Time:        1.624 s
```

- Medication-Alarm Module: Notifies the patient through auditory alarm when it is time to take their medication

1. Start and send MEDICATION_ALARM_STARTED notification
2. Handle MEDICATION_ALARM_TEST notification and update DOM
3. Create DOM elements with button and notifications wrapper
4. Helper schedules medication checks every minute
5. Check and trigger notifications
6. The MEDICATION_ALARM_TEST notification is sent successfully to the mirror

```
PASS  e2e  tests/e2e/MedicationAlarm.test.js
  Medication-Alarm Module
    √ should start and send MEDICATION_ALARM_STARTED notification (7 ms)
    √ should handle MEDICATION_ALARM_TEST notification and update DOM (2 ms)
    √ should create DOM elements with button and notifications wrapper (2 ms)
  Medication-Alarm Node Helper
    start
      √ should schedule medication check (2 ms)
    scheduleMedicationCheck
      √ should schedule a job to check and trigger medication notifications (2 ms)
    triggerMedicationNotification
      √ should send MEDICATION_ALARM_TEST notification (2 ms)

Test Suites: 1 passed, 1 total
Tests:       6 passed, 6 total
Snapshots:   0 total
Time:        1.953 s
```

ii. Verification

- Medication-Verification Module: Verifies if the patient has taken their medication after the alarm goes off

1. Start and send START_MEDICATION_VERIFICATION notification
2. Handle the VERIFY_MEDICATION_RESULT notification and update DOM
3. Create DOM elements for medication verification
4. End successfully
5. Handle failed medication verification
6. Handle delayed medication intake

7. Handle abrupt stop of medication intake
8. Handle multiple alarm triggers for medication intake
9. Handle successful medication verification
10. Start without errors
11. Exit script once verification is received

```
PS C:\Users\abida\magicmirror> jest tests/e2e/MedicationVerification.test.js
 PASS  e2e  tests/e2e/MedicationVerification.test.js
  Medication-Verification Module
    √ should start and send START_MEDICATION_VERIFICATION notification (5 ms)
    √ should handle VERIFY_MEDICATION_RESULT notification and update DOM (2 ms)
    √ should create DOM elements for medication verification (102 ms)
    √ should end successfully (1 ms)
    √ should handle failed medication verification (1 ms)
    √ should handle delayed medication intake (2 ms)
    √ should handle abrupt stop of medication intake (1 ms)
    √ should handle multiple alarm triggers for medication intake (1 ms)
    √ should handle successful medication verification (1 ms)
  Medication-Verification NodeHelper
    √ should start without errors (1 ms)
    √ should exit script once verification is received (1 ms)

Test Suites: 1 passed, 1 total
Tests:       11 passed, 11 total
Snapshots:   0 total
Time:        2.338 s
Ran all test suites matching /tests\\e2e\\MedicationVerification.test.js/i.
```

iii. Cloud

Login test case:
-Starts by simulating a login attempt with a valid user credentials
-Tests that the login is successful and a session has started
-Verifies that after successful login, the caretaker is redirected to profile pages
-Attempls a login with invalid credentials to make sure the system correctly prevents access
-Gives error message when a login attempt fails

Registration test case:
-Starts with simulating with valid new user details
-Tests after successful login, the caretaker is redirected to login page
-Ensure that the user's password correctly hashes before saving
-Confirms that the portal renders the registration page again with an error message when it fails

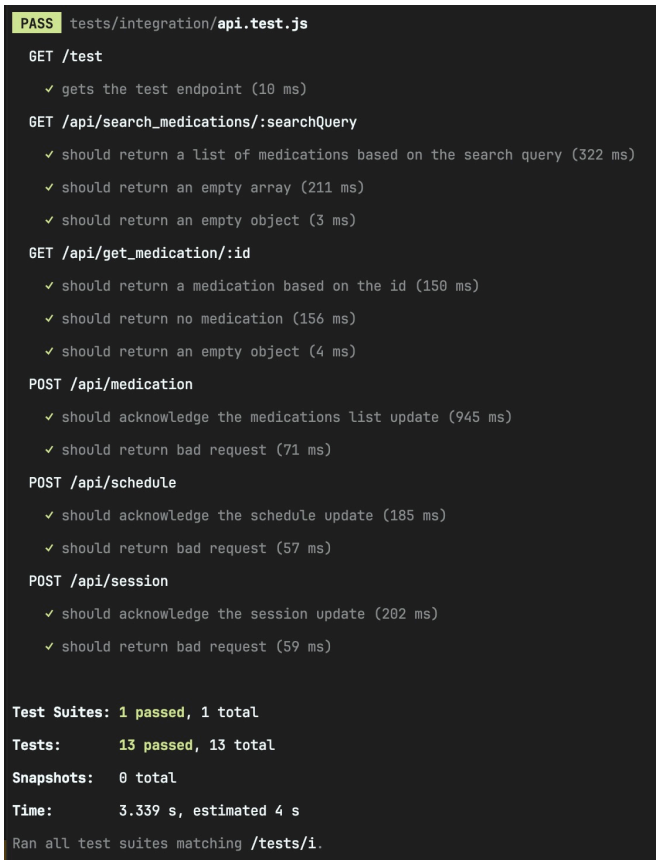
```
      at error (methods.js:290:17)
      at Object.<anonymous> (tests/integration/api.test.js:87:7)


Test Suites: 3 passed, 3 total
Tests:       7 passed, 7 total
Snapshots:   0 total
Time:        2.75 s, estimated 3 s
Ran all test suites matching /tests/i.
```

e. Integration Testing

Tested Components	Test description	Pass Criteria	Outcome
Hardware with the Software	Check that the software modules in charge of controlling these interactions are communicating with the hardware components (such as the camera and IR touch interface).	Verify medication intake with a camera by simulating hardware inputs and software responses.	When the alarm went off, the user was able to press the stop button triggering the stopping of the alarm. This started the medication verification which checked if the user's hand was on the mouth. Even though the camera on the hardware was positioned awkwardly, it eventually detected the hand on the mouth after the user went into the frame of the camera. **PASS**
All the modules on the cabinet	Confirm that the various modules work together (medication storage, reminders, patient interaction	The system successfully integrates all modules, and alarms are triggered correctly according to medication schedules, and the medication schedule is accurately created based on patient input.	The alarm goes off according to the medication schedule, and the system generates medication schedules based on inputted medication for individual patients. **PASS**
Data Flow and Synchronization	Validate the synchronization of data between the local database and the cloud.	Any changes to the schedule or patient medications that are made on the medicine cabinet are sent to the cloud.	Once a change is made on the local database, the change is promptly reflected on the database in the cloud. **PASS**

Table 2.2.1. Integration Tests

In addition to the above test cases, we also ran the following integration tests for our cloud component.



GET /test: Conducts a preliminary check to confirm the test environment is properly configured and the API endpoint is operational.

GET /api/search_medications/:searchQuery: Tests that the server processes a medication search query and confirms it can return a corresponding list of medications. Then, it verifies the server's ability to handle cases where no matches are found by returning an empty array and checks that an empty object is returned when appropriate.

GET /api/get_medication/:id: Tests the server's function in fetching specific medication details using a distinct identifier, ensuring it can successfully retrieve medication data or accurately indicate when no data is found for an ID.

POST /api/medication: Verifies the server's capacity to recognize and process updates to the medication list, including its response to erroneous requests that do not conform to expected data formats or validation criteria.

POST /api/schedule: Examines the server's acknowledgment of updates made to medication schedules, ensuring it responds correctly to successful changes and handles incorrect requests as needed.

POST /api/session: Tests the server's functionality in updating session information, a critical aspect of maintaining secure and consistent user sessions.

The testing outcomes show complete success across all 13 tests, showcasing that the API endpoints are effectively managing interactions, confirming the server's ability to interface with the backend services reliably, and communicating with the database to retrieve or update data accurately.

Hence these operations are functioning correctly on the mirror indicating that when the frontend component of the Magic Mirror issues requests to these API endpoints, the server is returning accurate data or confirmations. This is demonstrated, for example, when a medication schedule is modified via the Magic Mirror interface, and a POST request to /api/schedule successfully saves the updated schedule in the database, with the front end receiving a positive response.

Updated Project Plan

Task Name	**Details**	**Due Date**
Acceptance Test Demonstration Report	● Create an acceptance testing report ● Meet with stakeholders to perform and assess the system based on acceptance tests defined in Report R1	3/14/2024
Final Engineering Report	● Update reports (R1, R2, R3) based on the current, marketable product.	3/21/2024
Presentation & Demo Video	● Make a product promotional video. ● Show colleagues the product.	4/1/2024
Capstone Design Annual Exhibition	● Showcase a poster detailing the project's achievements. ● Demonstration of a working product	4/8/2024

Table 3.1. Project Plan

Contribution Matrix

Name	Student Number	Sections worked on

Lyba Mughees	100750490	Software Components:
		- Cabinet
		  - Layers
		  - Modules
		- Rationale for Design Change
		- Unit Testing
		  - Modules
		- Integration Testing
		- Project Plan

Massimo Albanese	100616057	Hardware Components: - Electronics - Medication Tray/Dispenser - Assembly - Rationale for Design Change - Integration Testing - Cloud

Abida Choudhury	100700985	Software Components:
		- Computer Vision
		  - Python Script
		  - Module
		- Unit Testing
		  - Verification

Hima Paul	100753261	Software Components:
		- Cloud
		- Unit Testing
		  - Cloud
		- Integration Testing
		  - Cloud

Table 4.1. Contribution Matrix

5. Test results from R#4

1. Test Results

Test #	Test Date & Tester	Result: Pass/Fail	Comments
FT1	Mar 14, 2024, Caregiver	PASS	- The input module and

			scheduler module are right on top of each other with no visual separation between them which caused some confusion while inputting medications.
FT2	Mar 14, 2024, Caregiver	PASS	- The input module and scheduler module are right on top of each other with no visual separation between them which caused some confusion while creating medication schedules.
FT3	Mar 14, 2024, Patient	PASS	- Alarm goes off at scheduled medication time.
FT4	Mar 14, 2024, Patient	PASS	- The result updates on the portal and an email is sent to the caretaker if medication is not taken on time.
FT5	Mar 14, 2024, Caregiver	PASS	- Caretaker can easily view patient data that has been pushed to the cloud.
UT1	Mar 14, 2024, Caregiver, Patient	FAIL	- There is some confusion for users navigating the interface. - Some extra instructions and guidance were required to understand how the UI works.
UT2	Mar 14, 2024, Caregiver, Patient	PASS	- No difficulty reading text or issues with current colour scheme
CT1	Mar 14, 2024, Caregiver, Patient	PASS	- Alarm, LED and containers work properly

			as expected
CT2	Mar 14, 2024, Caregiver	PASS	- Users can view weather and other personalized modules - Caretaker can push medication data and view on the cloud

Table 3.1: Test Results

6. Ethical Considerations[6]

The MagicMirror addresses the need for medication management, especially for those with complex medication plans or requiring assistance with following those plans. By automating pill dispensing and providing an intuitive user interface, the mirror enhances medication management and reduces the risk of medication errors which can significantly improve health outcomes, reduce healthcare costs associated with medication non-adherence, and enhance overall quality of life for patients and their caregivers.

Some aspects of the product design may raise some concerns related to privacy and data security. The use of cloud-based storage for patient and medication information and schedules raises questions about the security and confidentiality of sensitive medical data. To reduce these risks, robust data encryption and stricter access control measures to protect patient privacy could be implemented. Patients should also have an understanding of how their data will be used and shared.

7. Safety Considerations

Security and User Privacy Considerations:

User Authentication: In our caretaker portal, email and password are needed to view patient information. We have put in place robust measures for user authentication aimed at deterring unauthorized access.

Role-Based Access Control: Users get different levels of access depending on their roles. Caretakers can sign into the system as administrators and are therefore able to edit patients' profiles or schedules while other caretakers can only see their patients' schedules and not any other patient's information.

Secure Communication: Meanwhile, the cabinet communicates with the cloud server securely and uses HTTPS instead of HTTP to protect data exchange via secure channels.

Safety Concerns:

Electricity safety: Cabinet users should be aware of the electrical elements and handle them with care. The cabinet has been properly insulated and grounded to minimize the chances of electric shock.

Medicine Handling: Patients must adhere to the guidance given out by caregivers concerning their drug intake and use. The consequences that come with misusing drugs are harmful to the human body.

Maintenance checks: For proper working conditions, regular testing should be done on a hardware system within the cabinet to detect any threat that may cause danger.

Environmental Impact:

Energy Efficiency: Energy consumption and its environmental effects can be reduced by using efficient energy management techniques like dual voltage systems and LED lighting.

Remote Monitoring: Lower carbon emissions can result from fewer medical visits if we switch over to the use of a smart cabinet that is remotely controlled for drug schedule monitoring.

Electronic Waste: Disposing of used electronic components adds to electronic waste. There needs to be proper disposal methods and recycling of these parts to reduce this

Material Selection: The materials chosen for the cabinet's hardware affect the environment. It should be made with green materials which are biodegradable or recyclable.

8. Conclusion

Key Findings and Significance

Initially, this was an existing project that we had to redesign. The problem we had to solve was how to help dementia patients and their caretakers with medication management without it being a burden. After some deliberation with all stakeholders, we decided to take a different approach to the problem at hand than what was initially given to us. When we started our capstone project we were to make a smart mirror through the use of hardware and software to improve patient care and medication management. Encompassing a proof-of-concept for a medication reminder system, our purpose became an all-inclusive solution for elderly people with chronic illnesses who want to remain at home independently which caused us to change to the smart medicine cabinet approach.

Our product represents an innovative solution for medication management among patients and their caretakers. After eight months of planning and designing, both the hardware and software components have been refined to better fit the problem that dementia patients and their caretakers face. The software, which makes use of the Magic Mirror Framework, allowed us to have a modular and adaptable approach to our product.

The Smart Medicine Cabinet has the potential to be applied in a wide range of healthcare settings, specifically nursing homes. It addresses the critical concerns that caretakers have with ensuring their patients take their medications on time, reducing the burden on the caretaker, and allowing the patient to be more independent. Additionally utilizing the Magic Mirror Framework provides an adaptable approach for future enhancements beyond medication management.

Despite the advancements that can be in the healthcare field with the smart medicine cabinet, there are some limitations as well. Ensuring data security and privacy while using this in a setting like a nursing home remains a challenge to this day. This is due to the medication data that is stored for each patient locally on the device. In the future, this product can be modified to allow for user profiles for each patient where the data is stored separately for each patient. Future improvements can focus on enhancing accessibility features and simplifying the user interface further.

Learning Experience

As we used an iterative approach to design our product, we were able to continuously get feedback from stakeholders and adapt our product based on continuous improvement. Using this approach, allowed agility while responding to any evolving feedback, particularly in an area with limited prior experience.

While planning and designing, we were fortunate enough to collaborate with stakeholders who had experience from diverse backgrounds including healthcare professionals and engineers. This allowed us to get a comprehensive understanding of user needs and the challenges faced in addressing those needs. Due to the variety of input we received, we were able to ensure our project aligned with the real-world requirements.

The importance of invention and problem-solving in product development was highlighted when we addressed the problem presented to us and designed the Smart Medicine Cabinet. By utilizing existing frameworks and technologies to create novel solutions, the project demonstrated the potential to make meaningful technological advancements in healthcare.

9. Acknowledgments

We would first like to thank our Capstone Supervisor and Coordinator, Dr. Ramiro Liscano, and Dr. Qusay Mahmoud for assisting and guiding us through this project. Their knowledge and insight have been monumental in helping get our product where it is now. We would also like to thank Dr. Alvaro Quevedo for providing us with the preexisting hardware and literature that helped us get started on this project. We would also like to thank Dr. Winnie Sun

for her insight regarding the medical field and her knowledge of patient-caregiver interactions that allowed us to refine our design significantly. Those on the online MagicMirror forums deserve recognition for documenting their experience with this framework, significantly assisting us in troubleshooting and general knowledge on how it works. We have to thank our families for supporting us in this stressful time, balancing coursework and this project along with general post-grad tasks.

10. References

[1]H. Ben Hassen, W. Dghais, and B. Hamdi, "An E-health system for monitoring elderly health based on Internet of Things and Fog computing," *Health Information Science and Systems*, vol. 7, no. 1, Oct. 2019, doi: 10.1007/s13755-019-0087-z.

[2]H. Sun, H. Yu, G. Fan, and L. Chen, "Energy and time efficient task offloading and resource allocation on the generic IoT-fog-cloud architecture," *Peer-to-Peer Networking and Applications*, vol. 13, no. 2, pp. 548–563, Jul. 2019, doi: 10.1007/s12083-019-00783-7.

[3]C. Threewitt, "How Smart Rearview Mirrors Work," *HowStuffWorks*, May 16, 2014. Accessed: Oct. 16, 2023. [Online]. Available: https://auto.howstuffworks.com/under-the-hood/trends-innovations/smart-rear-view-mirrors.htm

[4]S. Bianco *et al.*, "A Smart Mirror for Emotion Monitoring in Home Environments," *Sensors*, vol. 21, no. 22, p. 7453, Nov. 2021, doi: 10.3390/s21227453.

Case	Description
UC-1	Medication Storage Management and Cataloging Description: The mirror stores medication in a safe, organized manner. Actor(s): Caregiver Preconditions: The mirror is powered on, functional, and connected to the internet. Basic Flow: 1. Caregiver opens relevant pill containers. 2. The caregiver scans medication to input the medication type corresponding to the device database, amount, and quantities. The caregiver inputs the schedule,

	and the compartment number into the cabinet. 3. The caregiver stores medication in the respective pill containers. 4. Repeat steps until all medication is stored or the containers are full for that designated period. 5. The caregiver closes pill containers and saves the medication input. Postconditions: Medications are securely stored, organized, and easily accessible. Exceptions: 2a. Medication Scan Fails: 1. System Displays Error Message 2. Caregiver manually inputs medication data 2b. The cabinet is not connected to the internet/can't connect to the medication database: 1. System Displays Error Message 2. Caregiver manually inputs medication data 3. Data gets synced to the cloud when the internet connection is restored
UC-2	**COMBINED WITH UC-5 -** Medication Reminders Description: The mirror notifies and/or reminds the patient to take their medication. Actor(s): Patient Preconditions: - Cabinet is powered on, functional, and connected to the internet. - Medication Schedules are set. - Medication is stored in the cabinet. Basic Flow: 1. It's time for medication intake. 2. Cabinet system triggers visual and auditory notifications for the patient. 3. Cabinet system displays a prompt on the screen. 4. The patient acknowledges the reminder by tapping the prompt. 5. Proceed to UC-3, Visual Medication Indication. Postconditions: The patient is reminded of medication intake and is in front of the cabinet system. Exceptions: 4a. Patient fails to acknowledge reminder within the set time frame: 1. Cabinet system notifies caregiver.
UC-3	**DISCARDED -** Visual Medication Indicator Description: The mirror visually indicates which pill containers the medication to be taken is in. Actor(s): Patient Preconditions: - Cabinet is powered on, functional, and connected to the internet. - Medication Schedules are set. - Medication is stored in the cabinet. - Patient is in front of the cabinet.

	Basic Flow: 1. It's time for medication intake. 2. Cabinet visually indicates which pill containers the medication is located in. 3. The medication pill containers is opened. 4. Proceed to UC-5, Medication Intake Verification Postconditions: The patient is ready to take the medication.
UC-4	<u>Online Caregiver Monitoring</u> Description: The Caregiver has access to all Patient data through an online portal. Actor(s): Caregiver Preconditions: - The cabinet is powered on, functional, and connected to the internet. - Patient has a history of taking medication with the system. Basic Flow: 1. Caregiver logs into an online portal connected to the cabinet system. 2. Caregiver accesses metrics and historical data of patient medication intake and behaviour, as well as view and modify patient data and medication scheduling if needed. 3. Caregiver logs out of the online portal. Postconditions: The caregiver is better informed and the information in the cabinet system is up to date.
UC-5	**<u>COMBINED WITH UC-2 -</u>** <u>Medication Intake Verification</u> Description: The mirror ensures that the patient has successfully taken their medication. Actor(s): Patient Preconditions: - Cabinet is powered on, functional, and connected to the internet. - Patient is ready to take medication and is facing the cabinet system. Basic Flow: 1. The patient picks up one pill from one of the containers. 2. The cabinet system tracks the change in quantity of pills in the pill containers using computer vision. 3. The patient ingests the pill. 4. The cabinet system tracks the patient's hand going across their mouth using computer vision and video recording. 5. Repeat until all needed medication is consumed. Postconditions: - The patient has successfully ingested the medication. - The system has tracked if the patient has taken the medication. Exceptions: 4a. Patient fails to ingest and quantity of medication in pill containers is different: - Notify the caregiver of the discrepancy - Store the video of the patient ingesting the pill

UC-6	**DISCARDED -** Interactive Features Description: The mirror exhibits interactive features, such as the weather, for Patient enjoyment and utility. Actor(s): Patient, Caregiver Preconditions: - Cabinet is powered on, functional, and connected to the internet. Basic Flow: 1. Patient prompts the cabinet system by voice for information, such as the current weather. 2. The cabinet responds via audio. 3. The patient can see the provided data/response on screen. 4. The patient can see standard data such as motivational quotes and the weather that are displayed by default. Postconditions: The patient is uplifted and better informed.
UC-7	**DISCARDED -** Caregiver to Patient Messaging System Description: The Caregiver can communicate with the Patient through the mirror via messages. Actor(s): Patient, Caregiver Preconditions: - Cabinet is powered on, functional, and connected to the internet. Basic Flow: 1. The caregiver accesses the online portal connected to the cabinet system. 2. The caregiver can write and submit messages to be displayed to the patient, with the choice to notify the patient or not. 3. The caregiver logs out of the portal. 4. Either the patient is notified of the new message or the message is displayed the next time the patient is in front of the cabinet system. Postconditions: The caregiver has communicated with the patient.

Table 7.1: Use Cases

R1	Caretaker will be able to select and create profiles for patients
R2	Caretaker will be able to select alarm sounds
R3	The application must accurately display patient's personal information
R4	The application should provide clear and visual medication schedules with reminders, reducing the risk of missed doses or incorrect medications
R5	The application should play audio through the speaker
R6	Patient should be able to interact with the application by selecting mood for the

	day and through notification alerts
F1	Application can be displayed on both smart mirror and smart watch
F2	Caretaker is able to look at multiple profile information at the same time

Table 7.2: Functional Requirements

[5] “Software engineering COCOMO model,” *GeeksforGeeks*, Apr. 13, 2018. Accessed: Nov. 17, 2023. [Online]. Available: https://www.geeksforgeeks.org/software-engineering-cocomo-model/

[6]"IEEE Code of Ethics," IEEE Computer Society, https://www.computer.org/education/code-of-ethics. Accessed: March 19, 2024

11. Appendices

a. Cabinet
   https://github.com/madlitch/magicmirror/
b. Cloud
   https://github.com/himapaul10/caretakerportal

Test #	Purpose	Preconditions	Inputs	Outputs	Process
FT1	The purpose of this test is to ensure that the system meets the conditions specified in UC-1 like medication data.	- Cabinet is powered on - Cabinet is functional and connected to the internet - Pill containers are opened	Medication is fetched from the FDA database based on generic name or brand name. Container number and quantity	Medications stored securely and in an organized manner being readily available to the patient when the alarm goes off. The medication	- The stakeholder opens the pill containers and searches the FDA database for the medication by brand name or generic name. - The stakeholder then inputs data for each medication along with the quantity

			inputted. Medications are then stored in the respective containers.	data is pushed to the cloud.	in the pill boxes from the mirror. - Medication data is then updated in the caretaker portal. - The medication quantity is then counted and put into their respective containers.
FT2	This test will validate that schedules can be created for each day and medication (UC-1).	- Cabinet is powered on - Cabinet is functional and connected to the internet - Medication being scheduled is already stored in a pill container	Medication list is retrieved from the cabinet_boxes database.	The schedule is pushed to the cloud and the local database.	- The stakeholders select a medication and add the days and time that medication is taken. - The schedule is updated on the portal under the respective date.
FT3	The purpose of this test is to ensure that the medication alarm runs as expected as per UC-2	- Cabinet is powered on - Cabinet is functional and connected to the internet - Medication schedules exist - The	Visual and auditory notifications triggered. Alert displayed on the screen.	Patient acknowledges the alert by pressing the stop button on the screen.	- When a medication is scheduled, a notification is triggered and a prompt is displayed on the screen telling the patient that it is time to take their medication.

		Medication is stored in the cabinet			
FT4	The purpose of this test is to check that medication intake occurs and the caretaker is notified via email if the intake is false as per UC-5.	- Cabinet is powered on - Cabinet is functional and connected to the internet - The alarm has been triggered - Patient is at the cabinet ready to take the scheduled medication	The computer vision script checks the patient's hand going over the mouth.	Patient ingests medication and data is sent to the cloud portal. If the medication is not ingested, then the caretaker gets an email and the data is also updated on the cloud.	- The alarm goes off - Stakeholder stops the alarm and the medication verification script begins. - One hand is detected covering the mouth, the script exists and the caretaker portal is updated to show the medication is taken. - If medication is not taken a minute, the process tries again for up to 3 times. If all 3 attempts are successful, portal updates to show medication isn't taken and caretaker receives email.
FT5	This test ensures that the patient data can be monitored from the Caretaker Portal as described in UC-4.	- The portal is functioning - The user has an internet connection - User has a valid login for the portal.	None	Caretaker can access and monitor patient data including the medication intake history, medication schedules and analytics.	- Stakeholder logins into the portal - Stakeholder then navigates to the patient view - Stakeholder views all the required analytics for the patient - Stakeholder logs out

Table 2.1: Functional Testing

Test #	Purpose	Preconditions	Inputs	Outputs	Process
UT1	This test will evaluate if the user interface is easy to navigate for patients and all other stakeholders.	- Cabinet is powered on - Cabinet is functional and connected to the internet - Caretaker logged into the portal and is using the cabinet simultaneously	Caretaker interacts with the cabinet UI to input medications and schedules. Caretaker also sees this update on the portal.	User can complete tasks with little to no confusion and independently.	- Stakeholder navigates through all the functional tests and completes all tasks
UT2	This test will validate the features of the user interface relating to the text size and colour schemas.	- Cabinet is powered on - Cabinet is functional and connected to the internet	Patient navigates through the system	Patient gives feedback on the readability, and the usability of the default settings	- Stakeholder interacts with the different modules in the cabinet - Stakeholder observes the default text that is used in the UI - Patient completes the interaction and provides and feedback

Table 2.2: Usability Testing

Test #	Purpose	Preconditions	Inputs	Outputs	Process
CT1	Validate that the system's software is integrated with the hardware.	- Cabinet is powered on - Cabinet is functional and connected to the	None	Hardware component LED flash when a notification is received Pill	- Alarm is started on the cabinet - LED flashes - Respective pill container dispenses pill

		internet		containers for each scheduled medication dispenses when it is time to take a pill	
CT2	Validate that the system can integrate with external system like the database	- Cabinet is powered on - Cabinet is functional and connected to the internet	None	Data is uploaded successfully to the local database and cloud Weather service from external service retrieved successfully.	- Stakeholder opens cabinet and sees the weather forecast for their area - Stakeholder pushes some medication data to the cloud and verifies that it is available on the portal

Table 2.3: Compatibility Testing

12. Contribution Matrix

Name	**Student Number**	**Contribution**
Lyba Mughees	100750490	- Executive Summary - R1 - Safety Concerns - Conclusion - Appendices
Massimo Albanese	100616057	- R2 - R4 - Test Results - Ethical Concerns - Acknowledgments

Abida Choudhury	100700985	- Executive Summary - Acknowledgments - Ethical Concerns - Safety Concerns
Hima Paul	100753261	- R3 - Acknowledgments - Safety Concerns - Conclusion - References