Federal University of Technology – UTFPR
Academic Department of Electronics – DAELN
Academic Department of Informatics – DAINF
Computer Engineering
Integration Workshop 3 (ELEX22) – S72 – 2025/1

Technical Report
S.E.R.E.NA

Eduardo da Costa dos Santos – eduardosantos.2020@alunos.utf.edu.br
Lucas Vinicius Sillis do Vale – vale@alunos.utf.edu.br
Mateus Szepeilewicz Camillo – mateuscamillo@alunos.utf.edu.br
Regian Canuto dos Santos – regian@alunos.utf.edu.br
Roman Herrera Nery – romannery@alunos.utfpr.edu.br
Pedro Scarpin Ribeiro – pribeiro@alunos.utfpr.edu.br

June 2025

Abstract

S.E.R.E.N.A is a smart home healthcare device aimed primarily at the elderly and individuals with cognitive or visual impairments, with the goal of ensuring proper adherence to prescribed medical treatments. The system will assist users by administering medication, monitoring symptoms, and facilitating communication with caregivers and healthcare professionals through voice interaction, visual and audio alerts, and integration with a centralized dashboard. As a key feature, the device will include an automatic medication dispenser capable of releasing the correct medication at scheduled times, enhancing both patient autonomy and treatment safety.

The device operates, on the hardware side, through a rotary dispenser which contains 14 compartments for pills placement, a microphone and speakers for user feedback and interactive communication, a camera for medicine recognition, and a Raspberry Pi for central control of all devices. On the software side, it utilizes YOLO for medicine detection and LLM via LangChain for conversation understanding and processing. Additionally, there is also an application used for monitoring, giving prescriptions, alerts and dispenser management for the doctors and caregivers. The integration provides an effective and innovative way for home healthcare.

1 Introduction

The world's population is aging at an unprecedented rate: by 2050, individuals aged 65 and over will constitute nearly 17% of the global population, and in Brazil this demographic has grown by 30% over the past decade [1]. Alongside increased life expectancy comes a rise in chronic conditions requiring strict medication schedules; yet studies indicate that up to 50% of elderly patients fail to adhere properly to their prescribed regimens, leading to avoidable hospitalizations and estimated healthcare costs exceeding $100 billion annually [2].

These figures highlight a critical gap in continuous, at-home patient monitoring and support for those with cognitive or visual impairments.

Traditional care models—based on periodic visits and manual reminders—are both labor-intensive and reactive, often failing to detect missed doses or emerging adverse symptoms in time. The COVID-19 pandemic further compounded these challenges by limiting in-person interventions and accelerating the need for remote healthcare technologies [3]. Consequently, there is an urgent requirement for an intelligent, automated system that ensures treatment adherence while preserving patient autonomy.

S.E.R.E.N.A (Smart Embedded Responsive Entity for Elderly Needs and Assistance) addresses this gap by combining an automatic medication dispenser, multimodal user interaction, and a centralized dashboard. The device administers the correct dose at scheduled times, monitors self-reported symptoms (e.g., pain level, side effects), and facilitates communication with caregivers and healthcare professionals. All events—medication taken or missed, symptom logs, alert acknowledgments—are synchronized to a secure mobile and web application, enabling real-time oversight and data export for clinical review.

By automating critical aspects of home treatment management, S.E.R.E.N.A provides real-time visibility into treatment progress for both caregivers and doctors, while reducing missed doses and enhancing quality of life for elderly users. Section 2 details the functional and non-functional requirements of the system, including hardware specifications, software architecture, and security protocols.

2 Project Specification

The requirements defined and guided the entire system development process, clearly outlining the functional and non-functional specifications. They were established based on the project's initial conception and the envisioned interaction between the system and its potential users. To facilitate organization and understanding, the requirements were divided into distinct categories: software requirements, which include both the developed application and the device's programming (shown in Table 1); hardware requirements, which involve the mechanical and electronic components (Table 2); combined requirements, which describe functionalities that depend on the integration of both hardware and software (Table 3); and finally, documentation requirements, which cover user support materials and usage instructions (Table 4).

ID	Description
FR01	The device must notify the user at the correct time to take medications.
FR02	The system must allow medication confirmation through voice or IR sensor.
FR03	The device must collect symptom reports through interactive questions.
FR04	The system must activate upon detecting a predefined wake word, allowing hands-free interaction.
FR06	The system must identify the medication shown to the camera.

ID	Description
FR07	The system must allow repetition of instructions upon user request.
FR08	The system must support voice-to-text and text-to-voice conversion.
FR09	User commands must be processed by a Large Language Model (LLM), as well as the responses.
FR10	The system must allow doctors to register prescriptions and diagnosed illnesses.
FR11	The doctor must have access to the dashboard.
FR12	The doctor must have the option to export the report.
FR13	The prescription may include medication names, time windows for intake, and daily dosage restrictions for non-continuous medications.
FR14	The system must allow caregivers to receive notifications via the app.
FR15	The system must send alerts to caregivers if the user does not respond within the expected time.
FR16	The caregiver must register the medication times and dosages based on the medical prescription.
FR17	The caregiver must register the insertion of the medication into the dispenser.
FR18	The collected data must be sent to a dashboard.
FR19	The dashboard must display medication history, reported symptoms, and critical alerts.
FR20	The dispenser must automatically release the medication at the scheduled time according to the registered prescription.
FR23	The system must inform the caregiver of the current quantity of medications available in stock.
FR24	The system must calculate and inform the caregiver how many days of medication are available based on the recorded dosage schedule.
FR27	The system must allow the caregiver to remove obsolete or unused medications from the records.
FR28	The system must support three types of users: doctor, caregiver, and regular user (patient).
FR29	The system must provide an authentication system for users to securely log in.
FR30	The system must allow regular users to register a doctor who is associated with them.
FR31	The system must allow caregivers to register both doctors and regular users who are associated with them.
FR32	The system must notify the caregiver if the connection between the device and the application is lost, for example, due to a network failure.
FR35	The system must notify the caregiver if a medication presented to the camera is not recognized.
NFR01	The system must achieve at least 80% accuracy in voice recognition under typical home environments.
NFR03	Dashboard reports must be presented in PDF format.
NFR04	The device must operate using a Wi-Fi connection for data synchronization.
NFR06	The software must be compatible with Android devices.
NFR07	The camera must be capable of identifying medications with a minimum accuracy rate of 80%.
NFR09	The system must log all critical operations (e.g., medication dispensing, user authentication, voice commands) for auditing purposes.
NFR11	The language used to interact with the user must be english.

ID	Description
NFR12	The Large Language Model interaction will be created using LangChain Library.
NFR13	The Computer Vision algorithm will be an OCR model from tesseract.
NFR14	The application will be developed using the Flutter framework.
NFR15	The system will locate a medicine box in the video using a YOLO model using ultralytics package.
NFR16	The LLM will be an Open Source Model i.e LLAMA, Ollama, deepseek Etc.
NFR18	The Natural Language to Voice decoder will be created using gTTS.
NFR20	The Voice to Natural Language decoder will be created using speech-recognition library.

Table 1: Requirements - Software

ID	Description
FR05	The system must light up an LED while listening to the user.
FR22	The dispenser must emit an audible alert when releasing a medication.
FR33	The system must allow the caregiver to access the manual operation functionality of the dispenser in order to place or organize the medications.
FR34	The system must assign an identification method to each compartment, allowing the registration and control of the position of the stored pills.
NFR05	The system must operate on AC power for continuous usage.
NFR08	The dispenser must release all scheduled medications with a maximum delay tolerance of 1 minute.
NFR10	The dispenser must have a physical mechanism that allows easy access for cleaning and maintenance of the pill storage compartments.
NFR17	The station will include an option to increase the volume if necessary.
NFR19	The station will include an option to decrease the volume if necessary.

Table 2: Requirements - Hardware

ID	Description
FR21	The dispenser must identify the correct compartment to release the medication corresponding to the dose and time.
FR25	The system must detect and notify the caregiver of any malfunction in the dispenser's motors.
FR26	The system must provide an option for manual medication retrieval in case of a malfunction (e.g., power outage).
NFR02	The system's response time to voice commands must be less than 10 seconds under normal conditions.

Table 3: Requirements - Hardware and Software

ID	Description
NFR21	The station will include a user manual.
NRF22	The application will include a user manual.

Table 4: Requirements - Documentation

3 System Architecture

This section presents the overall architecture of the SERENA system, which integrates mechanical, electronic, and software components to deliver a smart, interactive healthcare assistant. The architecture is divided into three main layers. The mechanical structure covers the physical housing, slot layout, and dispenser mechanics. The electronic layer describes the hardware used for actuation, sensing, and power management. Lastly, the software layer details the firmware running on the embedded device, the backend infrastructure, and the mobile application that enables real-time interaction between patients, caregivers, and healthcare professionals.

3.1 Mechanical Structure

The S.E.R.E.N.A device features a compact, 3D-printed case with dimensions of 20 cm × 10 cm × 15 cm. All electronic modules and mechanical components are designed to fit neatly within this case.

A central element of the structure is the pill dispenser disk, which has a diameter of 8.6 cm and contains 14 compartments for storing and organizing medications. The disk is rotated using a stepper motor, and a solenoid is used to actuate the hatch located at the bottom of the disk, allowing precise pill release.

3.1.1 Case

The case was designed using Tinkercad and printed in white PLA, ensuring structural integrity and dimensional precision.

The front panel of the case includes:

- A cutout for the Raspberry Pi camera module;
- Perforated sections above and below the camera for the microphone and speaker, respectively;
- Additional cutouts for access to the Raspberry Pi's USB and power ports.

Internally, the design includes dedicated mounts for the camera module and a central support structure intended to house the stepper motor and solenoid, contributing to the overall alignment and stability of internal components.

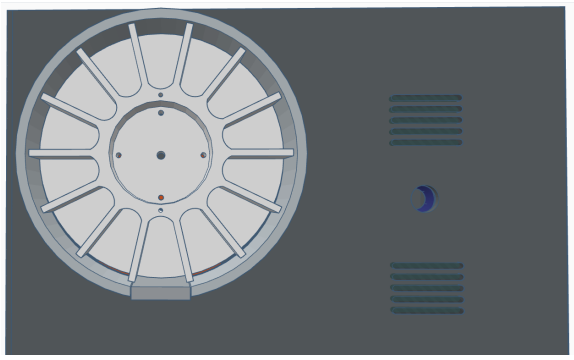

Figure 1: Front view of the case showing camera, microphone, and speaker cutouts

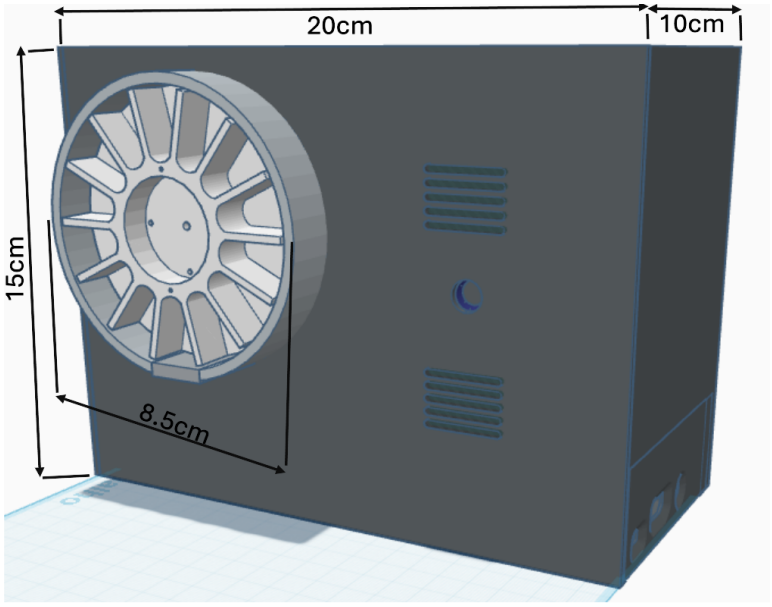

Figure 2: 3D render of the case design

3.1.2 Dispenser Mechanism

The dispenser mechanism consists of a rotating disk, a centrally mounted stepper motor, and a solenoid-actuated hatch. Both the motor and solenoid are securely attached to the central tower, enabling precise rotation and controlled pill release.

Figure 3: Internal view of the dispenser mechanism with motor and solenoid

3.2 Eletronic

The electronic structure was developed with the aim of simplifying connections and making the most of the capabilities of the components used. A key aspect of the system is the management of input voltages, as different components operate at different voltage levels. The electronic assembly is primarily responsible for controlling the motors and actuators that operate the medication dispenser, as well as processing image and audio inputs provided by the user, with image input being used for the medicine box recognition function, and playing the responses generated by the LLM through the speakers.

3.2.1 The electronic components of the system

- **Power Supply** - The system uses a 12-volt, 5-ampere power supply.
- **Raspberry Pi 3** – Responsible for the overall management of the device, controlling motors and actuators, acting as an intermediary between audio and video inputs and cloud processing, as well as managing audio output.
- **Solenoid JF-0530B (12V)** – Responsible for opening and closing the medication dispenser, activated when the correct medicine is in the proper position.
- **LM2596 Voltage Regulator Module (Step-Down 12V → 5V)** – Used to reduce the main power supply voltage (12V) to 5V, which is required for the Raspberry Pi.
- **Stepper Motor US-17HS4401S (5V)** – Controls the position of the dispenser, ensuring the correct medicine is aligned for release.
- **A4988 Driver** – Responsible for controlling the stepper motors, enabling precise movement of the mechanism.

- **Raspberry Pi Camera Module v1.3** – Captures images of the medicine boxes, used in the identification process through computer vision.
- **IRF520 MOSFET Module** – Controls the activation of the solenoid, allowing the opening and closing of the dispenser compartment.
- **USB Microphone** – Captures user voice input. The USB interface simplifies the connection and makes better use of the Raspberry Pi's USB ports.
- **USB Speaker** – Provides audio output to the user. Unlike the microphone, it uses the Universal Serial Bus (USB) interface solely for power supply, receiving five volts of power, while the audio signal is transmitted via the 3.5mm audio jack (P2 connector).

3.2.2 Electrical System

The system's power management is handled by a twelve-volt, five-ampere power supply, sized based on practical testing. During these tests, it was found that the speaker did not operate properly when powered solely through the Raspberry Pi's USB port. Therefore, an adaptation was made: the speaker now receives five volts of power via USB from the voltage regulator, while the audio signal is transmitted through the Raspberry's 3.5mm audio jack (P2 connector).

The overall schematic diagram of the system's electronic architecture is shown in Figure 4, detailing the various connections between the devices and the Raspberry Pi, as well as the interconnections among the components themselves.

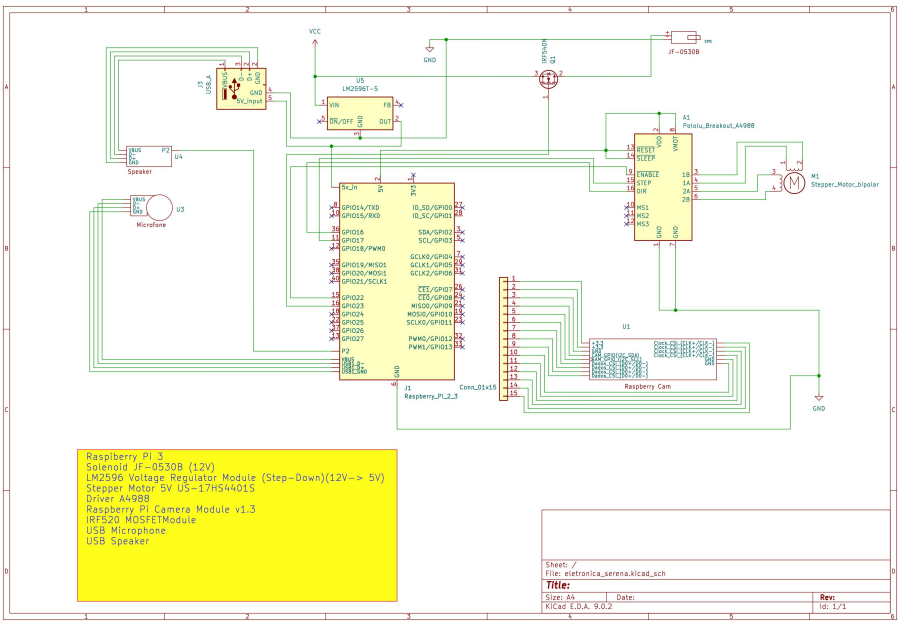

Figure 4: Overall electronic schematic of the project

3.3 Software

The S.E.R.E.N.A system is composed of three main software modules: the user-facing Serena App, the embedded software running on the medication dispenser device, and the central Serena API that handles all backend operations, data persistence, and AI integration. This section presents an overview of each component, their roles, and how they interact.

3.3.1 User Interface (Serena App)

The SERENA mobile app is the primary interface for doctors and caregivers to monitor and assist elderly patients. Developed using React Native, it runs on both Android and iOS platforms. The app supports secure login and signup, and its functionality is structured as follows:

- **Cross-platform Development:** Built with React Native to ensure code reusability between Android and iOS. Components were styled using the ‘StyleSheet‘ API and adapted with platform-specific customizations when needed.
- **Architecture:** Follows a modular MVVM architecture with separation between UI components, business logic, and data services. Screens are organized by feature modules.
- **State Management:** Utilizes Redux Toolkit for predictable and centralized state management, with separate slices for user, patient, dispenser, and prescription data.
- **Navigation:** Handled via React Navigation using Stack and Tab navigators. Deep linking is supported for push notification redirection.
- **Secure Authentication:** Login and token management are implemented using JWT, with credentials securely stored via ‘react-native-keychain‘ (leveraging Android Keystore and iOS Keychain).
- **API Integration:** All communication is performed over HTTPS using Axios. Service layers are abstracted for cleaner API usage and better error handling. Tokens are refreshed automatically when expired.
- **Push Notifications:** Real-time symptom alerts and medication intake confirmations are implemented using the `expo-notifications` library. The system handles notifications in both foreground and background, supporting deep linking to route users to the relevant app screens.
- **PDF Export:** Reports including medical logs and symptom history are generated and exported as PDFs using a native bridge or third-party libraries like ‘react-native-pdf-lib‘.
- **Multi-user Support:** Users can associate with multiple elderly profiles, switch between them, and manage their configurations individually via the Settings screen.
- **Accessibility and Responsiveness:** UI is designed with accessibility in mind, including proper semantic roles and labels. Layouts are responsive using Flexbox and percentage-based sizing.
- **Offline Support (optional):** Some critical data is cached locally using AsyncStorage to improve performance and allow limited offline usage.

- **Build and Distribution:** App builds and OTA updates are managed using Expo and EAS CLI. Environment variables are separated via '.env' files for development and production.

Figures 5, 6, 7, and 8 showcase the login interface, dispenser configuration, prescription CRUD interface, and user-patient association screen (settings), respectively.

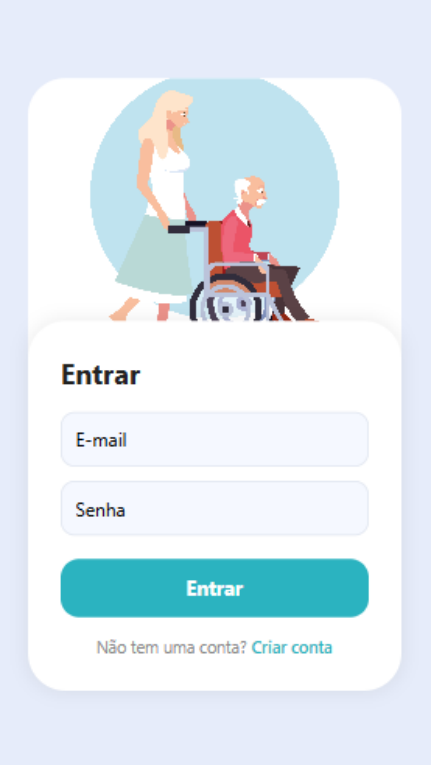

Figure 5: Login interface

Figure 6: Dispenser configuration interface

Figure 7: Prescription CRUD interface

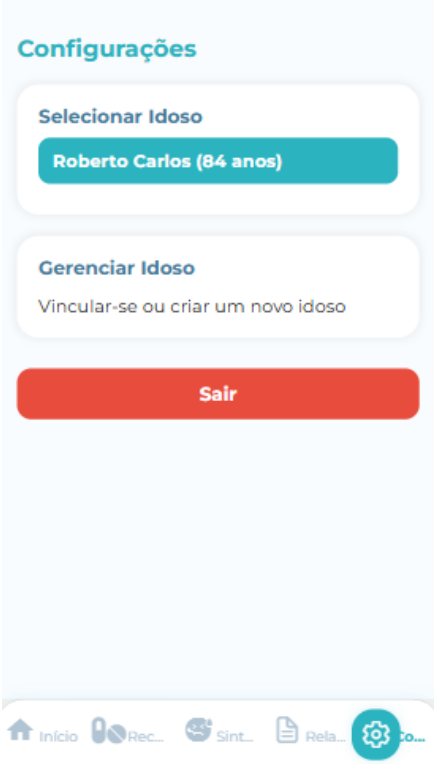

Figure 8: Settings interface

3.3.2 Embedded Software (Raspberry Pi)

The embedded Python-based firmware on the Raspberry Pi controls all interactions with the patient and the physical components of the dispenser. It runs a continuous loop for audio-based interaction, which includes wake word detection (*"Serena"*), transcription, LLM inference, and device control. A typical voice interaction loop can be seen below.

```
while True:
    if voice_detected():
        transcription = transcribe()
        command = send_to_llm(transcription)
        execute_command(command)
```

Commands interpreted may include inquiries about upcoming doses, symptom registration, or medication dispensing. The patient may choose to:

- Show the medication box to the camera for verification using YOLOv8 + OCR (EasyOCR), which checks against prescriptions.
- Request automatic dispensing, activating the rotating system via stepper motors to release medication.

For dispensing, the firmware includes logic to calculate the number of motor steps needed based on the desired dosage. Each slot is mapped to a specific angular position, and the system computes the rotation needed to align the correct slot with the dispensing mechanism. LEDs are used to visually indicate the active slot during the process, and safety checks ensure that only one slot is dispensed per cycle.

The firmware also manages:

- Audio capture (PortAudio + Google Speech Recognition)
- Audio feedback (gTTS for voice prompts)
- Camera handling (libcamera)
- Motor control (A4988 drivers)
- Power regulation (LM2596)

The Raspberry communicates with the cloud backend via Wi-Fi using HTTP protocol.

3.3.2.2 Voice Recognition Software The voice recognition software is responsible for mediating the interaction between the user and the embedded system through voice commands. This functionality is implemented by the `VoiceDecoder` class, which encapsulates a complete pipeline for both voice input and output.

The class integrates three main features:

- **Wake word detection:** A continuous loop listens to the environment for a predefined wake word (default: “Serena”). Upon detection, the system activates voice command capture.
- **Speech-to-text recognition:** Using the `speech_recognition` library, audio captured from the microphone is transcribed into text, with support for Brazilian Portuguese (`pt-BR`).
- **Text-to-speech synthesis:** System responses are converted into audio using the `gTTS` (Google Text-to-Speech) API and played using the `pydub` library.

The module also supports integration with auxiliary controllers, such as RGB lighting devices, allowing for visual feedback to the user during interactions.

This approach provides an accessible and responsive voice interface, ideal for screenless embedded applications, such as Raspberry Pi-based voice assistants.

3.3.2.3 Medicine Recognizer Software This component is responsible for identifying medicine names from camera-captured images of medication packages. It consists of two main parts: OCR processing and object detection via YOLO. The pipeline combines these technologies to extract and clean the names of medicines in real time.

- **OCR Pipeline:** The OCRPipeline class handles image preprocessing, text extraction using EasyOCR, and stopword removal using NLTK. The flow includes loading the image, applying filters to enhance text visibility, and generating a clean list of words by eliminating noise and common Portuguese stopwords.
- **Detection Pipeline:** The DetectionPipeline class captures video frames, uses a YOLO model to detect medicine boxes, and checks for bounding box stability before triggering OCR. When a stable detection is confirmed, it extracts the bounding box, preprocesses the image, and applies OCR to retrieve the medicine name.

3.3.2.4 LLM Interaction To provide intelligent and clinically aware responses, the system integrates with a Large Language Model (LLM) using the LangChain framework. This module enables the assistant to process natural language commands and respond based on medical prescriptions and patient data.

3.3.2.4.1 Prompt Engineering A custom prompt is defined using LangChain's `PromptTemplate`, tailoring the behavior of the assistant "Serena" to work within strict clinical and formatting constraints. The prompt:

- Instructs the assistant to recommend a single appropriate medication from the patient's prescriptions based on reported symptoms.
- Handles patient queries about prescriptions or general health by offering neutral or informative suggestions.
- Ensures that all outputs are returned in a fixed JSON schema, containing the following fields: `sintoma`, `medicamento_recomendado`, `dose`, and `sugestão`.

No explanations or text outside the JSON are allowed. The prompt enforces consistency and machine-readability of all language model outputs.

3.3.2.4.2 Model Configuration The system uses the Gemini 2.0 Flash model through the `ChatGoogleGenerativeAI` interface from LangChain. The model is initialized with a temperature of 0.0, ensuring deterministic responses suitable for clinical environments.

3.3.2.4.3 Environment Management Sensitive credentials, such as API email, password, and device ID, are securely loaded from environment variables via the `dotenv` package. This design improves security and allows seamless deployment across different systems.

3.3.2.4.4 Voice Integration Use Case When the patient interacts via voice (as outlined in Section 3.3.3.2), the transcribed command is passed to the LLM alongside their list of prescriptions. The assistant returns a structured JSON response, which is parsed to trigger actions like medication dispensing or issuing verbal guidance.

3.3.2.5 Dispenser Control The automated medicine dispenser is implemented using a stepper motor and relay mechanism. The class `MedicineDispenser` is responsible for controlling the rotation of a compartment wheel and triggering the dispensing action. It guarantees positional precision, reliable dose delivery, and safe reset after each operation.

3.3.2.5.1 Mechanical Operation The dispenser is built around a circular wheel divided into 14 compartments. Each compartment holds a dose of medicine. The system uses a stepper motor to rotate the wheel in a counterclockwise direction and stop precisely at the target slot. After dispensing, the motor rotates clockwise to return to a predefined reference position (slot 8).

3.3.2.5.2 Position Persistence To ensure the correct position is maintained between sessions or power cycles, the system stores the last known slot index in a file. This avoids misalignment in case of system restarts and ensures consistent operation. The position is loaded during initialization and updated after every dispensing action.

3.3.2.5.3 Motor and Relay Control The stepper motor is controlled via two GPIO pins: one for direction and another for step signals. Each step triggers a precise mechanical movement. A configurable delay between steps allows adjustment of motor speed. The relay is connected to a separate GPIO pin and is activated momentarily to release the medication from the selected slot.

3.3.2.5.4 Dispensing Algorithm The `run()` method receives a list of slot indices to activate. It sorts these slots based on their counterclockwise distance from the current position to optimize the rotation path. For each target slot:

- The motor moves to the slot.
- The relay is activated to dispense the dose.
- The new position is saved to file.

After completing all dispensing operations, the system automatically returns to the reference position to prepare for future commands.

3.3.2.5.5 Cleanup and GPIO Management At the end of execution or in case of shutdown, the `cleanup()` method releases the GPIO resources, preventing conflicts or warnings in subsequent executions.

3.3.3 Serena API (Backend Services)

The Serena API is implemented using FastAPI and serves as the central backend layer. It exposes endpoints to support the app, the embedded device, and the AI services. Major responsibilities include:

- Managing users, patients, and device associations
- CRUD operations on prescriptions, medications, symptoms, and dispenser configurations
- Logging and processing of medication intake, symptom reports, and image validations
- Integration with the AI layer to handle voice-based queries and structured decision flows
- Real-time alerting to caregivers

Data persistence is managed via a PostgreSQL relational database, while AI pipeline integrations rely on LangChain to send prompts and retrieve structured JSON responses from Gemini 2.0 Flash.

A high-level deployment diagram is presented in Figure 9, and Figure 10 illustrates a typical sequence for a medication event involving patient, device, API, and caregiver.

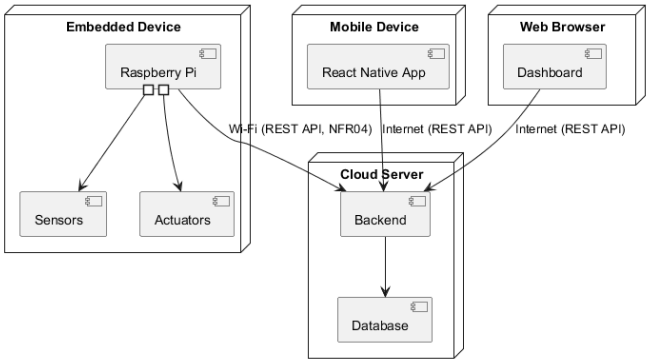

Figure 9: Deployment Diagram

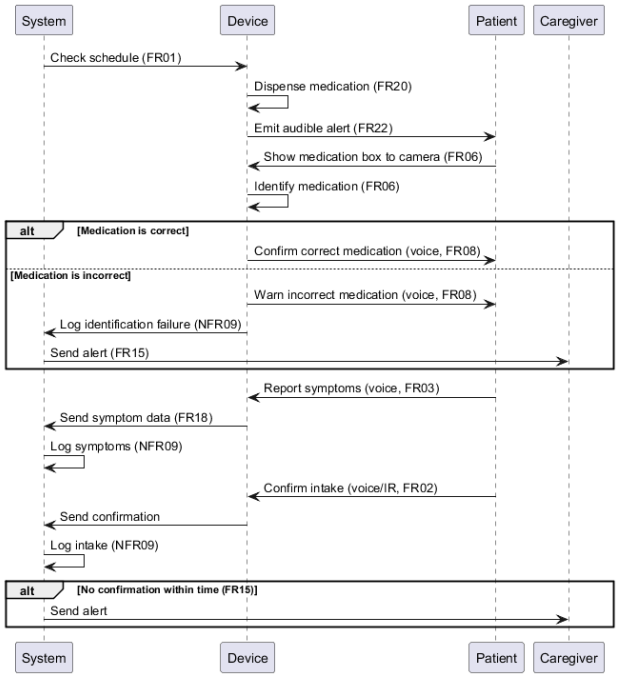

Figure 10: Sequence Diagram

Further details on API endpoints, device-level functions, software diagrams, and the overall system design are available in the project blog [4] and documentation.

The next section details the results of the project.

4 Results

The outcome of the project and its operation can be seen **here** and are also referenced in [5].

The project specifications and requirements were largely met, with all mandatory deliverables completed. Overall, the team followed the proposed schedule, with most tasks completed on time. Some activities took slightly longer than initially estimated, while others progressed more quickly. Despite these fluctuations, the final delivery remained aligned with the original planning.

In terms of workload, the number of hours dedicated to the project was slightly above the optimistic estimate of 330 hours but still below the extended estimate of 429 hours (optimistic +30%), demonstrating efficient use of time throughout the development.

The budget was adhered to, and the selected components performed effectively while remaining within financial constraints. Additionally, the risk analysis conducted during the planning phase proved valuable. Measures such as planning redundancies and selecting reliable components—like acquiring quality power regulation and backup hardware—helped mitigate potential issues and ensured the project's completion without exceeding the budget.

The following Table 5 shows a detailed cost analysis, presenting an itemized breakdown of the components along with their respective quantities, unit costs, and total expenses. This evaluation includes all necessary items for the system's implementation, covering hardware, materials, peripherals, and accessories.

Table 5: Cost Breakdown

Item	**Quantity**	**Unit Price**	**Subtotal**
Raspberry Pi 3	1	R$ 400,00	R$ 400,00
Microphone	1	R$ 90,00	R$ 90,00
Speaker	1	R$ 24,00	R$ 24,00
Raspberry Pi Camera rev 1.3	1	R$ 50,00	R$ 50,00
3D Printing	1	R$ 390,00	R$ 390,00
Step Motor Driver	1	R$ 15,00	R$ 15,00
Step Motor	1	R$ 80,00	R$ 80,00
Jack P4 Connector	1	R$ 10,00	R$ 10,00
Solenoid	1	R$ 40,00	R$ 40,00
Solenoid Relay	1	R$ 6,00	R$ 6,00
		Total	**R$ 1.131,00**

The project timeline was generally well adhered to, with most weekly deliverables being completed within the expected timeframe. Minor delays occurred in a few instances, primarily due to factors such as workload balancing and resource availability, but these were managed effectively to avoid impacting the overall schedule. As shown in Table 6, three out of five deliverables were completed on time, demonstrating the team's ability to maintain consistent progress. Although Deliverables 2 and 5 experienced delays of 3.28 and 6.12 hours respectively, the majority of the work aligned with the planned estimates, highlighting both the project's efficiency and the team's adaptability in managing challenges during execution.

Table 6: Project Work Progress

Deliverable	Worked Hours	Expected Hours	Delay (h)	Status
Deliverable 1	39	42.78	-	On Time
Deliverable 2	64	60.72	3.28	Delayed
Deliverable 3	95	102.12	-	On Time
Deliverable 4	179	184.92	-	On Time
Deliverable 5	111	104.88	6.12	Delayed

Source: Authors (2025).

5 Conclusion

The SERENA project delivers an innovative and inclusive solution for home healthcare, with a strong focus on assisting elderly individuals and those with cognitive or visual impairments. By combining automated medication dispensing, image-based medication recognition, and intuitive voice interaction, the system addresses key challenges in treatment adherence and patient autonomy.

All mandatory requirements established at the outset of the project were successfully met, demonstrating the team's commitment to both functional and user-centered design. The development process was also carried out within the predefined budget, reflecting effective planning and resource management.

Beyond meeting the technical goals, SERENA was designed with a strong focus on accessibility, safety and ease of use, which are essential aspects for any assistive healthcare device. The current version not only proves the feasibility of the solution but also builds a solid base for future improvements and potential use in real-world healthcare settings.

5.1 Future Work

To further enhance the effectiveness and usability of the SERENA system, several targeted improvements are planned for future development. One key area is the refinement of the automatic medication dispenser. Enhancing the precision, capacity, and modularity of the dispensing mechanism would allow the device to handle a wider variety of medication formats (e.g., pills, capsules, sachets), increasing flexibility and reliability in medication management.

Another important improvement involves expanding voice interaction capabilities to support multiple languages and regional accents. This would make the system more accessible to a broader user base, especially in multilingual households or communities. Enhanced natural language processing will also improve the system's ability to understand and respond to user commands more intuitively.

Additionally, upgrading the medication recognition module is a priority. By integrating more advanced image recognition models and sensor technologies, the system could more accurately identify medications based on packaging, shape, color, or embedded QR codes. This would reduce human error when refilling or verifying medication types, increasing overall treatment safety.

These improvements aim to make SERENA more inclusive, intelligent, and robust in real-world use, especially for elderly users and individuals with cognitive or visual

impairments.

Acknowledgements

We would like to express our sincere appreciation to all those who supported this project, whether through direct involvement or indirect encouragement. In particular, we extend our heartfelt thanks to Professors Gustavo Benvenutti Borba and Humberto Remigio Gamba for their exceptional guidance, insightful feedback, and continuous support throughout the entire development process. Their expertise was instrumental in shaping the direction and quality of this work.

References

[1] World Health Organization. Global aging report. Report, 2025. Accessed 2025.

[2] Institute of Gerontological Studies. Elderly fall incidence and costs. Report, 2024. Accessed 2025.

[3] A. Silva et al. Smart monitoring solutions for home healthcare. Technical Report, 2025. Accessed 2025.

[4] Team 4. Integration workshop 3 - team 4. https://integration-workshop-3.notion.site/Integration-Workshop-3-Team-4-1c9301496014807f829ecfe890a297c7?pvs=143, 2025. Accessed: 2025.

[5] Team 4. S.e.r.e.n.a - final video. https://drive.google.com/file/d/1dF3C764pqf6cd0N8kzH4veiXbiLlPWFG/view?usp=sharing, 2025. Accessed: 2025.