SAPIENZA
UNIVERSITÀ DI ROMA

DEPARTMENT OF COMPUTER SCIENCE

Project Report

MULTIMODAL INTERACTION A.Y. 23-24

Professors:
Maria De Marsico

Students:
Lucia Fores 1836451

Academic Year 2023/2024

Contents

1 Introduction

With the increase in life expectancy and the increase of the average health conditions of the elders in recent years, the percentage of elderly people who prefer to continue living alone and in their own home has also increased (it is estimated that in 2019 there were about 66% of the over 75 there were considered still auto-sufficient [1]). In this context, the use of technological tools such as IoT or AAL (Ambient/Active Assisted Living) devices is gaining ground.
Their use, thanks also to integration with artificial intelligence, makes possible to introduce a new way of helping and assisting the elderly people by caregivers and family members. [2]

1.1 Report Structure

In this report it will be given an overview of the design and developing journey addressed in order to produce the first release of an AAL system to support elders in the daily activity of taking medications.
Specifically in section 2 it will be given an overview of the background in which the system wants to be inserted, in section 3 all the use-cases developed for the system together with the modalities used in order to develop them will be listed and explained, in section 4 it will be presented the system architecture at different levels: at first in a general way, then more specifically by explaining how every part of the system works and finally focusing on features specifically; in section 5 it will be given an overview of the libraries used and an overview of the new code that has been developed in order to make the system work, in section 6 it will be presented an evaluation of the interface by following the Nielsen Heuristics, in section 7 it will be provided all the possible scenarios in which all the system's user can take part; in section 8 it will be given an overview of all the limitations of the system and a brief description of the future work that are related to the project, finally in section 9 it will be given a final overview of the job done.

2 Background

2.1 Elderly Care

The **Elderly Care** (a.k.a. **eldercare**) refers to services older people often need for physical or mental impairment.
There are many services that fall under the definition of eldercare like:

- **Assisted Living**: a housing facility for people with disabilities or for people who do not want or cannot live independently;
- **Adult Daycare**: non-residential facility that supports the health, nutritional, social, and daily living needs of adults in a professionally staffed, group setting;
- **Long-Term Care**: variety of services which help meet both the medical and non-medical needs of people with a chronic illness or disability who cannot care for themselves for long periods. It is actually divided into **Formal Long-Term Care** (that is provided in formal structures or at home by specialists) and **Informal Long-Term Care** (that is provided by family members, friends and unpaid voluntaries);
- **Nursing Homes**: facility for the residential care of older people, senior citizens, or disabled people;
- **Hospice Care**: health care that focuses on the palliation of a terminally ill patient's pain and symptoms also attending to their emotional and spiritual needs at the end of life;
- **Home Care**: health care or supportive care provided in the individual home where the patient or client is living, generally focusing on paramedical aid by professional caregivers, assistance in daily living for ill, disabled or elderly people, or a combination thereof.

2.2 Caregivers

Someone who takes care of a person who is young, old, ill, or disabled (= having an illness, injury, or condition that makes it difficult for them to do some things that other people do), either as a family member or friend, or as a job [3]

The roles of a caregiver are multiple and include all the activities that the assisted, in this specific case the elderly person, may need. An illustrative but non-exhaustive list of the roles covered by a caregiver is the following:

- **Help for physical needs**: include helping the assisted in dressing up, toileting and conducting household tasks;

- **Help for health and medical needs**: include helping the assisted in exercising, eating balanced meals and taking medication in a timely manner;
- **Help for emotional and psycho-social needs**: include helping the assisted by listening to them and providing care and support;
- **Help for financial and legal needs**: include helping the assisted in managing daily expenses, insurance, assets, and managing finances for their future.

2.3 Goals

The goal of this project is to propose the initial development of an AAL system that wants to be inserted in the context of an assisting system for old people and caregivers. The idea behind the project is to create a system that will help old people in remembering which medications they must take in a day, when to take them and help them in be sure to take the correct medication at the correct time; the system will also work as a constant assistant to send help messages to the caregivers. As for the caregiver point of view the system will serve as a alert system that will notify them when their assisted is in need of help and what is their health status when they take the medication; the system will also notify to the caregivers which are the medication taken (with proofs) and at what time their assisted took them.

3 Use Case

3.1 UML Diagram

In Figure 1 is presented the UML diagram for the use cases of the project

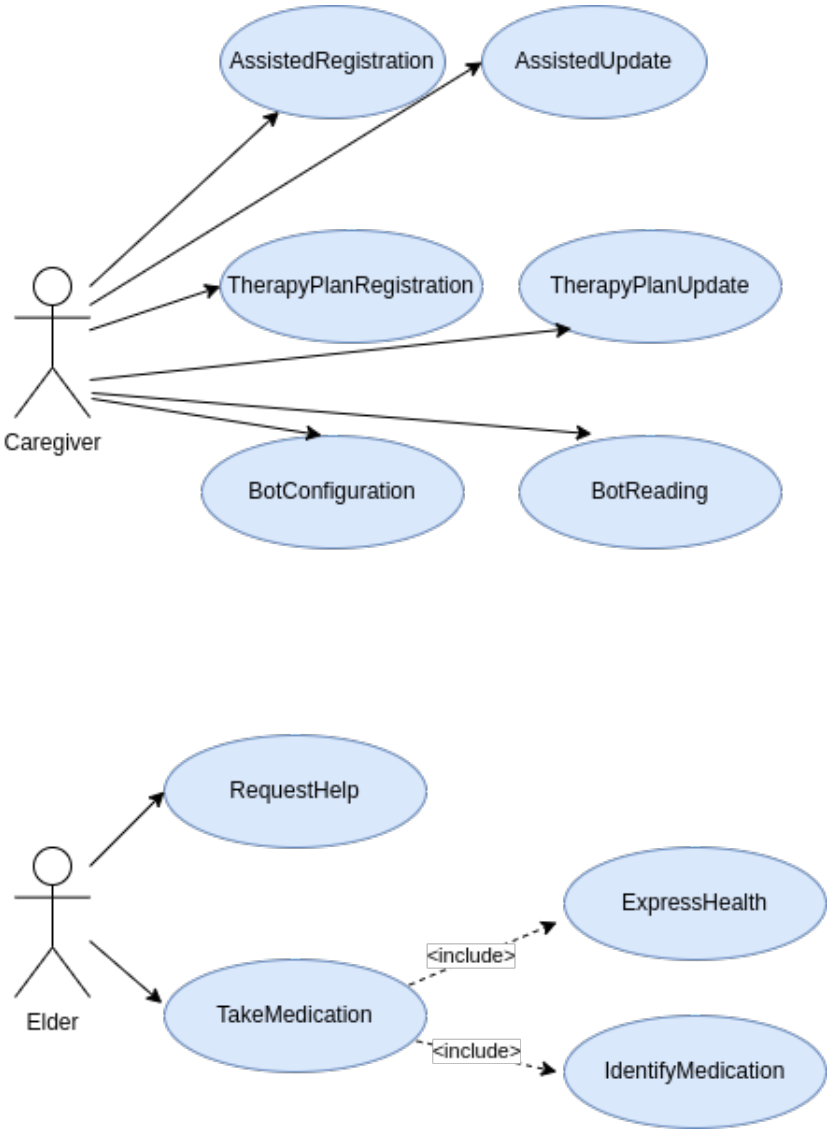


Figure 1: Use-Case UML Diagram

3.2 Multimodality

In order to provide a more complete description of the use cases of the system it is necessary to present the **multi-modal techniques** used in the project:

- **Speech Recognition and Voice Control**: at the base of the project there is the idea to create a system that the elders will find intuitive to use; in order to do that the primary multi-modal technique chosen has been the one of the speech recognition and voice control: thanks to that the elders will be able to

communicate spontaneously their needs to the system and will be able to answer spontaneously to the requests of the system;

- **Speech Synthesis**: in order to maintain the communication between the elder and the system as much spontaneous as possible, the system will answer to any of the possible stimulus in natural language;
- **Text Recognition in Pictures**: always with the goal to maintain a spontaneous communication between the system and the human user, it has been decided to also simulate vision and text comprehension since this will be extremely useful in the recognition of the right medication to take (Figure 10).

3.3 Use Case Description

In the following tables are presented the description for all the use-cases identified for the project.
For each use-case it will be highlighted the **name**, the **goal**, the **pre-conditions**, the **post-conditions** and the **execution flow**.
In subsubsection 3.3.1 are reported all the use-cases for the users of the system with the **caregiver** role, while in subsubsection 3.3.2 are reported all the use-cases for the users of the system with the **elder** role.

3.3.1 Caregiver Use-Cases

Name	AssistedRegistration
Goal	The caregiver of the assisted user wants to register their information.
Pre-conditions	None
Post-conditions	The information of the assisted user will be available in the system.
Flow	
User	System
1. Opens the registry file for the assisted user; 2. Compiles the file.	

Table 1: Assisted Registration Description

Name	AssistedUpdate
Goal	The caregiver of the assisted user wants to update their information.
Pre-conditions	The registry file for the assisted user already contains some information
Post-conditions	The information of the assisted user will be updated in the system.
Flow	
User	System
1. If the system is running, stop it; 3. Opens the registry file for the assisted user; 4. Updates the file.	2. The system stops;

Table 2: Assisted Update Description

Name	TherapyPlanRegistration
Goal	The caregiver of the assisted user wants to register their therapy plan.
Pre-conditions	None
Post-conditions	The information of the therapy plan will be available in the system.
Flow	
User	System
1. Opens the folder containing the therapy plan files; 2. For each one of the files, compiles the therapy plan.	

Table 3: Therapy Plan Registration Description

Name	TherapyPlanUpdate
Goal	The caregiver of the assisted user wants to update their therapy plan information.
Pre-conditions	The therapy plan files for the assisted user already contains some information
Post-conditions	The information about the therapy plan of the assisted user will be updated in the system.
Flow	
User	System
1. If the system is running, stop it; 3. Opens the folder containing the therapy plan files; 4. For each one of the files, compiles the therapy plan.	2. The system stops;

Table 4: Assisted Update Description

Name	BotConfiguration
Goal	The caregiver of the assisted user wants to configure the alert bot on their mobile phone.
Pre-conditions	The caregiver has Telegram installed on their mobile phone
Post-conditions	The caregiver will be ready to receive information about their assisted on the mobile phone
Flow	
User	System
1. If the caregiver has no username set on Telegram, set it; 2. The caregiver searches for the bot on Telegram; 3. The caregiver starts a conversation with the bot;	4.The bot replies with a standard message.

Table 5: Bot Configuration Description

Name	BotReading
Goal	The caregiver of the assisted user reads a message sent from the alert bot.
Pre-conditions	The caregiver has the bot already configured and the assisted user triggered the sending of a message.
Post-conditions	The user received a new message from the bot
Flow	
User	System
2. The caregiver reads the message;	1. Given the trigger activated by the assisted user, the bot sends the relative message to the caregiver;

Table 6: Bot Reading Description

3.3.2 Elder Use-Cases

Name	RequestHelp
Goal	The elder user sends an help request to the caregiver through the alert bot.
Pre-conditions	The caregiver has configured the system, has the bot already configured and the system is up and running.
Post-conditions	The caregiver received an help request through the bot
Flow	
User	System
2. The user says "aiuto";	1. The system is running and waiting for a command by the elder user; 3. The system replies to the user and tell them that is sending an help request to the caregiver; 4. The system triggers the send of an help request to the caregiver; 5. The system tells the user that the message has been sent; 6. The system goes back to the wait of a trigger.

Table 7: Request Help Description

Name	TakeMedication
Goal	The elderly person is guided by the system in the correct intake of the medications to be taken at a specific time.
Pre-conditions	The caregiver has configured the system, has the bot already configured and the system is up and running.
Post-conditions	The elder took a medication.
Flow	
User	System
3. The user express their health;	1. The system is running and recognize that is time for the elder to take a medication; 2. The system welcomes the user and ask them how are they feeling; 4. The system lists the medication to be taken and starts the identification process; 5. The system says goodbye to the user and sends a recap message to the caregiver. 6. The system goes back to the wait of a trigger.

Table 8: Take Medication Description

Name	ExpressHealth
Goal	The elderly person communicate to the system their current health.
Pre-conditions	The elder is in the process of taking a medication
Post-conditions	The elder expressed their health and the value is saved by the system.
Flow	
User	System
3. The user speaks and tell the system how they are feeling; 7. The user reply to the system;	1. The system is running a procedure to let the elder take their medication; 2. The system welcomes the user and ask them how are they feeling; 4. If the user said a word starting with "ben" or "mal" it proceeds, otherwise it asks the user to express again their health; 5. If the user said a word starting with "ben" the system goes back to the original procedure to let the user take the medication; 6. If the user said a word starting with "mal" the system asks them if they want to send an help message to the caregiver; 8. If the user said "sì" or "no" it proceeds, otherwise it asks the user to answer again; 9. If the user said "sì" the system triggers the send of the help message through the bot; 10. If the user said "no" the system goes back to the original procedure to let the user take the medication.

Table 9: Express Health Description

Name	IdentifyMedication
Goal	The system tells the elder if the medication box from which they want to take the medication is the correct one.
Pre-conditions	The elder is in the process of taking a medication
Post-conditions	The system recognized and saved the box from which the elder is taking the medication.
Flow	
User	System
4. The user says ”foto”; 7. When ready, the user says ”avanti”;	1. The system is running a procedure to let the elder take their medication; 2. The system lists the medication to be taken and for each one of them does the identification process; 3. The system asks to the user to take the box corresponding to the medication to take and asks them to say ”foto” in order to start the identification process; 5. The system checks if the name on the box corresponds to the one of the medication that the user is supposed to take; 6. If it is correct the system says to the user how much medication it has to take and waits for the signal from the user that says that is ready to proceed, otherwise it asks them to show another box; 8. If the medication was not the last one to be taken the process starts again, otherwise the system goes back to the main procedure TakeMedication;

Table 10: Identify Medication Description

4 System Architecture

4.1 Architecture Diagram

In Figure 2 is presented the architecture diagram of the system

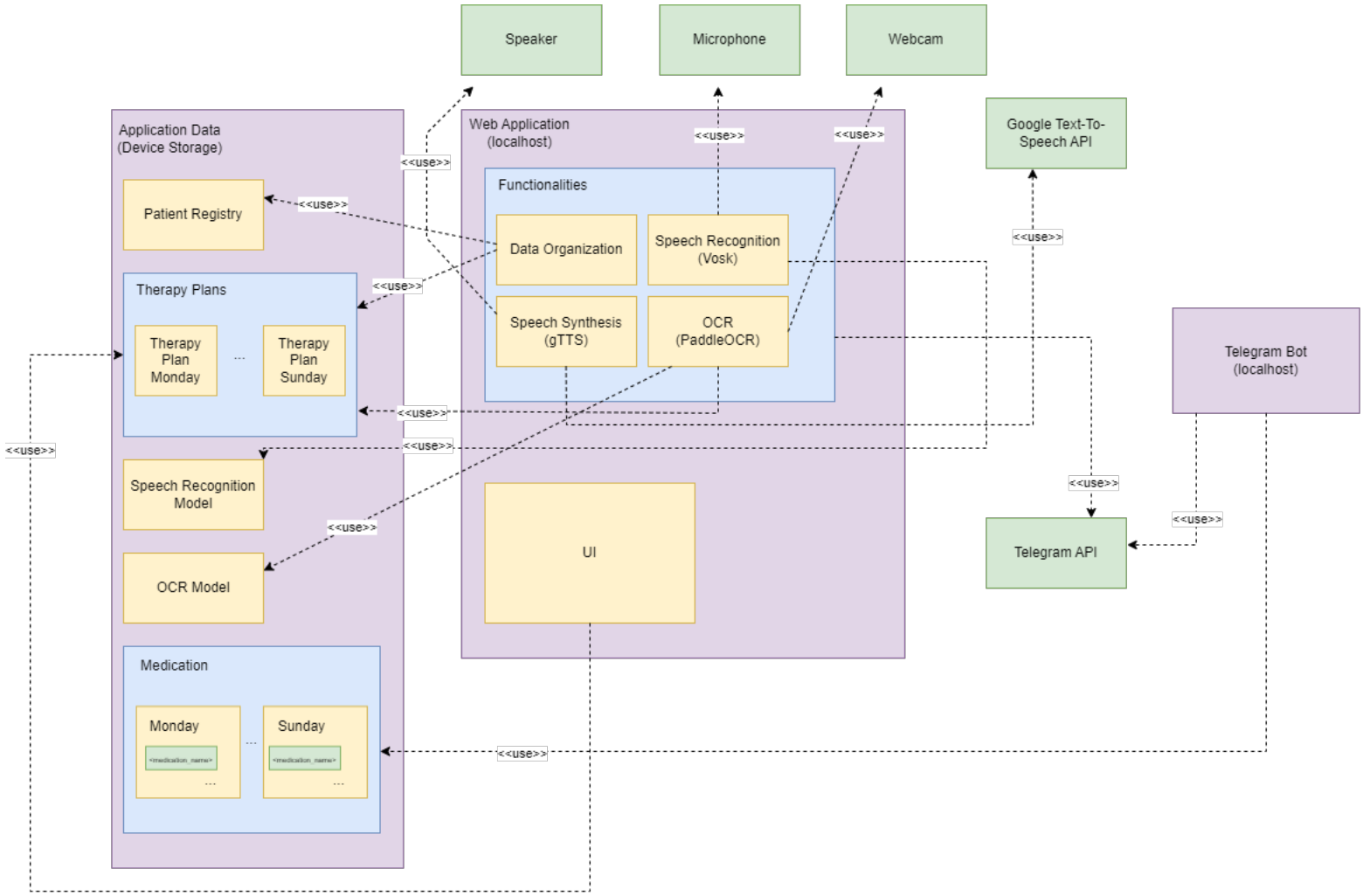


Figure 2: Architecture diagram

4.1.1 Diagram Description

In the following section it will be presented a description of the architecture diagram of the system (depicted in Figure 2).
For each one of the components it will be described the location of the component, the sub-components and the hardware and software dependencies.

Application Data The data used by the application, in this version of the system everything is stored in the device storage. The application data is made up by:

- **Patient Registry**: CSV file that stores the informations about the patient;

specifically the file stores the **name**, **gender**, **age** and the **caregivers' telegram usernames** (Figure 13);

- **Therapy Plan**: folder containing 7 CSV files each one representing the therapy plan for each day of the week; the files are structured in the following way: each file is divided into 36 sections (one for each half an hour that goes from 6:00 to 23:30), for each section is possible to specify as many medication as needed and for each medication is possible to specify the quantity of medication to be taken (Figure 13);

- **Speech Recognition Model**: Vosk model to perform the speech recognition, the model chosen is *vosk-model-small-it-0.22*; it has been decided to use this model mainly for two reasons: the first one is the fact that the model is optimized for the use on embedded devices and the second one is because by using the Italian language to communicate it has been possible to conduct more precise tests during the deployment.

- **OCR Model**: Paddle model to perform OCR, the model chosen is the Italian one offered by PaddlePaddle;

- **Medication**: folder containing 7 folders (one for each day of the week) in which, for every medication taken it is saved the picture of the medication box recognized as the correct one from the system; this data is not a persistent information (it is needed only to send the recap message to the caregiver) and is deleted every time that the procedure needed to take the medication is started.

Web Application The web application implementing the functionalities of the system and the user interface that can be used by the users. In the context of the project the application in run in a local environment and so it is hosted when needed on the machine localhost. The web application can be divided into the following two big components:

- **Functionalities**: contains all the functionalities of the system specifically:
  - **Data Organization**: the functions needed to pre-process the data so that they can be used by the application; in order to work it needs the *Patient Registry* and the *Therapy Plans*. The data pre-process also needs to use the Telegram API in order to find the ***chat ids*** of the caregivers so that it can correctly configure the interaction with the Telegram Bot;
  - **Speech Recognition**: performed thanks to the use of the Vosk library, the speech recognition of the system uses the device microphone and the speech recognition model; the use of the Vosk library made the system able

to perform fast speech recognition thanks to the use of the offline model that led to the minimization of the wait for the results;

  - **Speech Synthesis**: performed thanks to the use of the gTTS library, the speech synthesis of the system uses the device speaker and the Google Text-To-Speach API; early in the development of the system it was chosen to use an offline synthesizer called *pyttsx3* so to minimize the wait for the results also in this case, however, it was then decided to switch to the gTTS library (an online one) in order to provide to the users a better interaction since the voice synthesizing of *pyttsx3* was strongly robotic and it was not really good in providing the correct word pronunciations;
  - **OCR**: performed thanks to the use of the PaddleOCR library, the OCR of the system uses the device webcam to capture the image with the text to be recognized and the therapy plan of the given day in order to know the medication's name that must be recognized; during the early stages of the development of the system many OCR systems like *Tesseract* and *EasyOCR* but the final choice was to use *PaddleOCR* for its ease of use and very high accuracy of results [4]

- **UI**: the UI of the Web Application has been developed thanks to the use of the Flask library; the development of the web page has been made thanks to the use of HTML, CSS anf Javascript. The web application launches the interaction as a background thread and the communication between the user interface and the functionalities has been made possible thanks to the use of the *socketio* library.

Telegram Bot Telegram bot created thanks to the use of the Telethon library. The bot is used to send messages to the caregivers and uses the Telegram API in order to establish communication between the web application and the bot itself; moreover the bot uses the medication pictures to send the proofs of the medication taking to the caregivers.
During the early development of the system it was thought to use Whatsapp as messaging platform to contact the caregivers since it's the most used social messaging platform in the 16-64 age group [5], however it was much more complicated to create an automated bot that didn't need the use of an external phone number to correctly work; moreover in this case it was necessary for the caregivers to share their personal phone number in order to let the automated messaging system work. So it was decided to prioritize the usage of a simpler and "safer" tool like a Telegram Bot.

4.2 Class Diagram

In Figure 3 is presented the class diagram of the system, each component of the system is decomposed in the files that compose it; for the *app.py* file it has been decided to

further decompose the functions into two main categories that are the one about the functionalities of the system and the one about the user interface. In the class diagram are not presented the functions that has been taken from the libraries but only the ones that has been newly written.

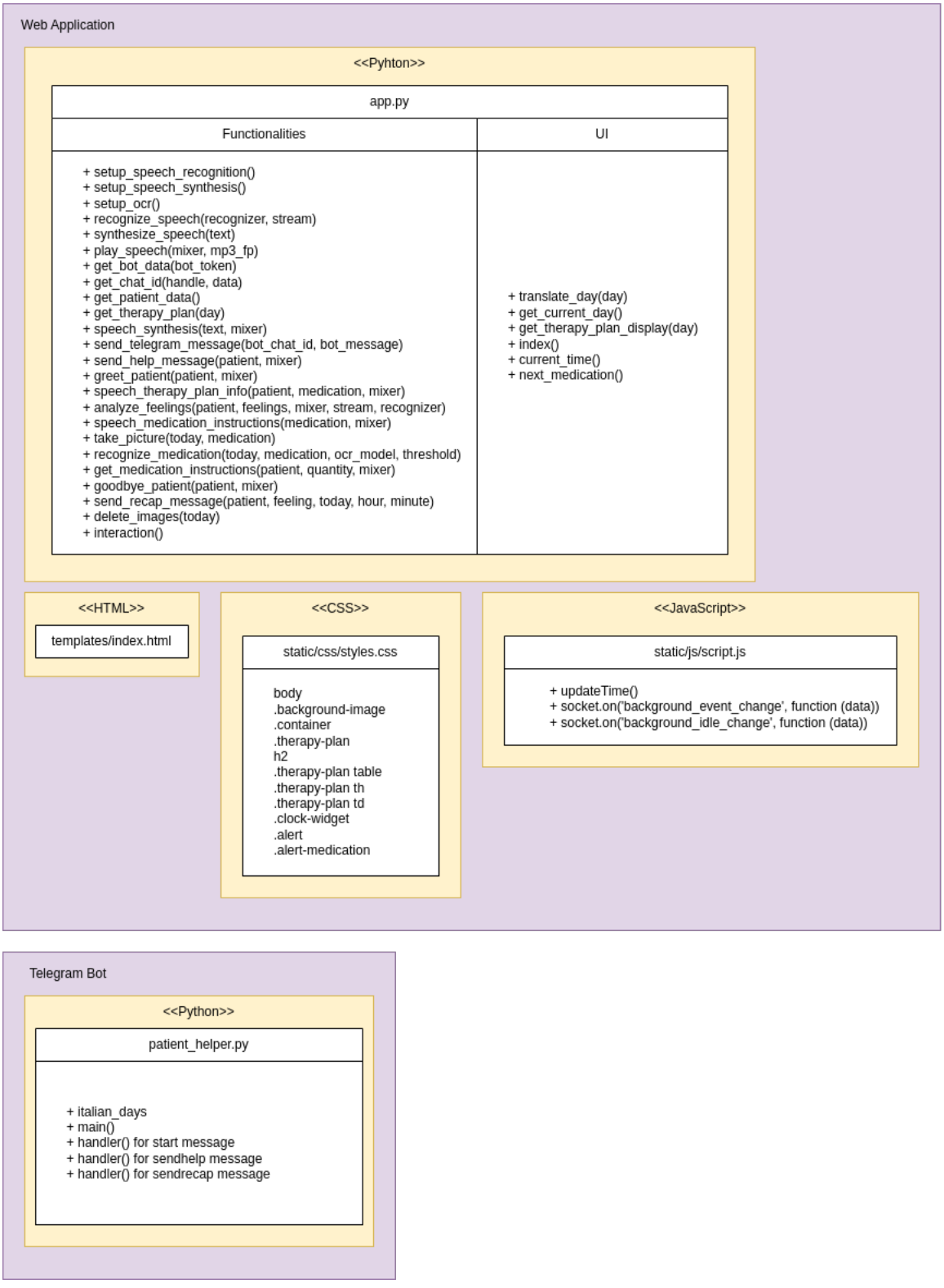


Figure 3: Class Diagram of the system

4.3 Activity Diagrams

In the following section are presented the execution flow for each one of the identified use-case (available in section 3)

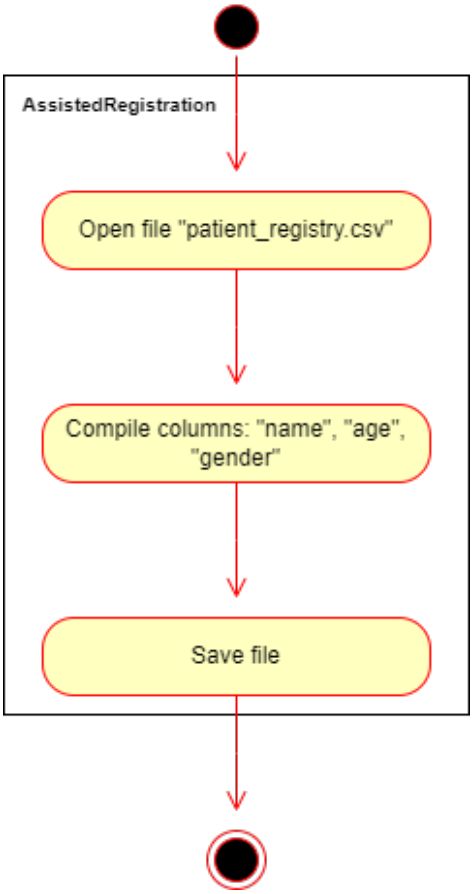


Figure 4: AssistedRegistration Activity Diagram

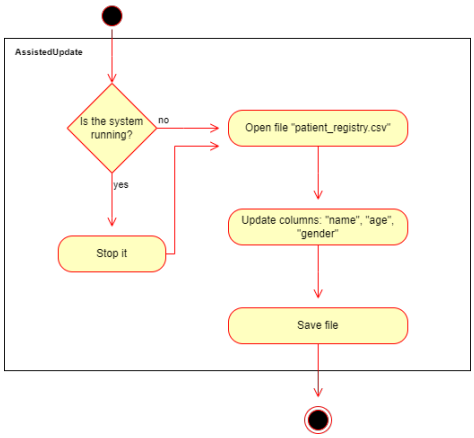


Figure 5: AssistedUpdate Activity Diagram

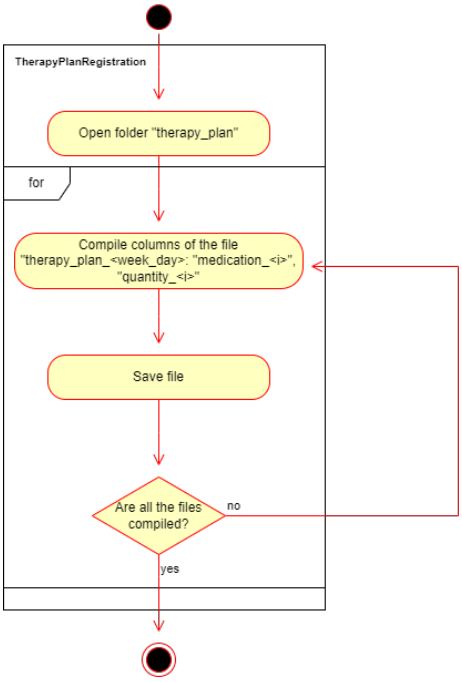


Figure 6: TherapyPlanRegistration Activity Diagram

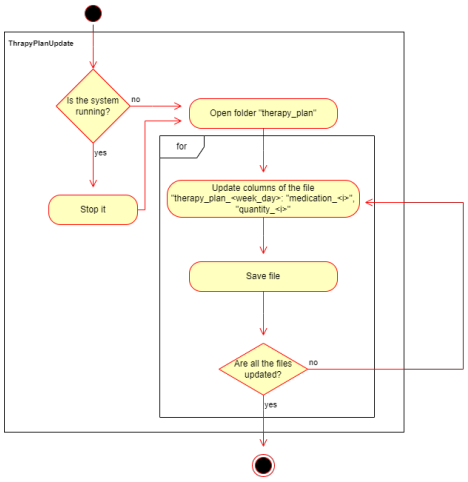


Figure 7: TherapyPlanUpdate Activity Diagram

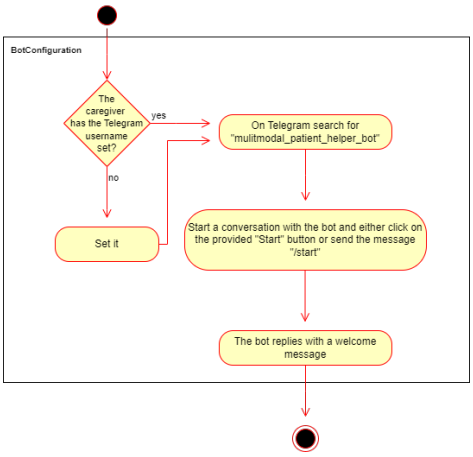


Figure 8: BotConfiguration Activity Diagram

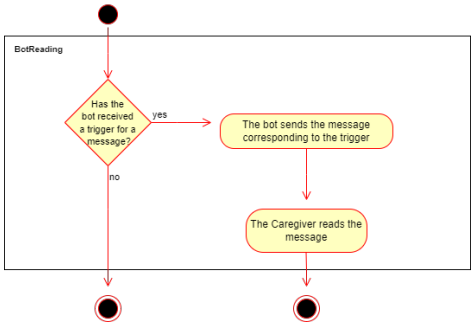


Figure 9: BotReading Activity Diagram

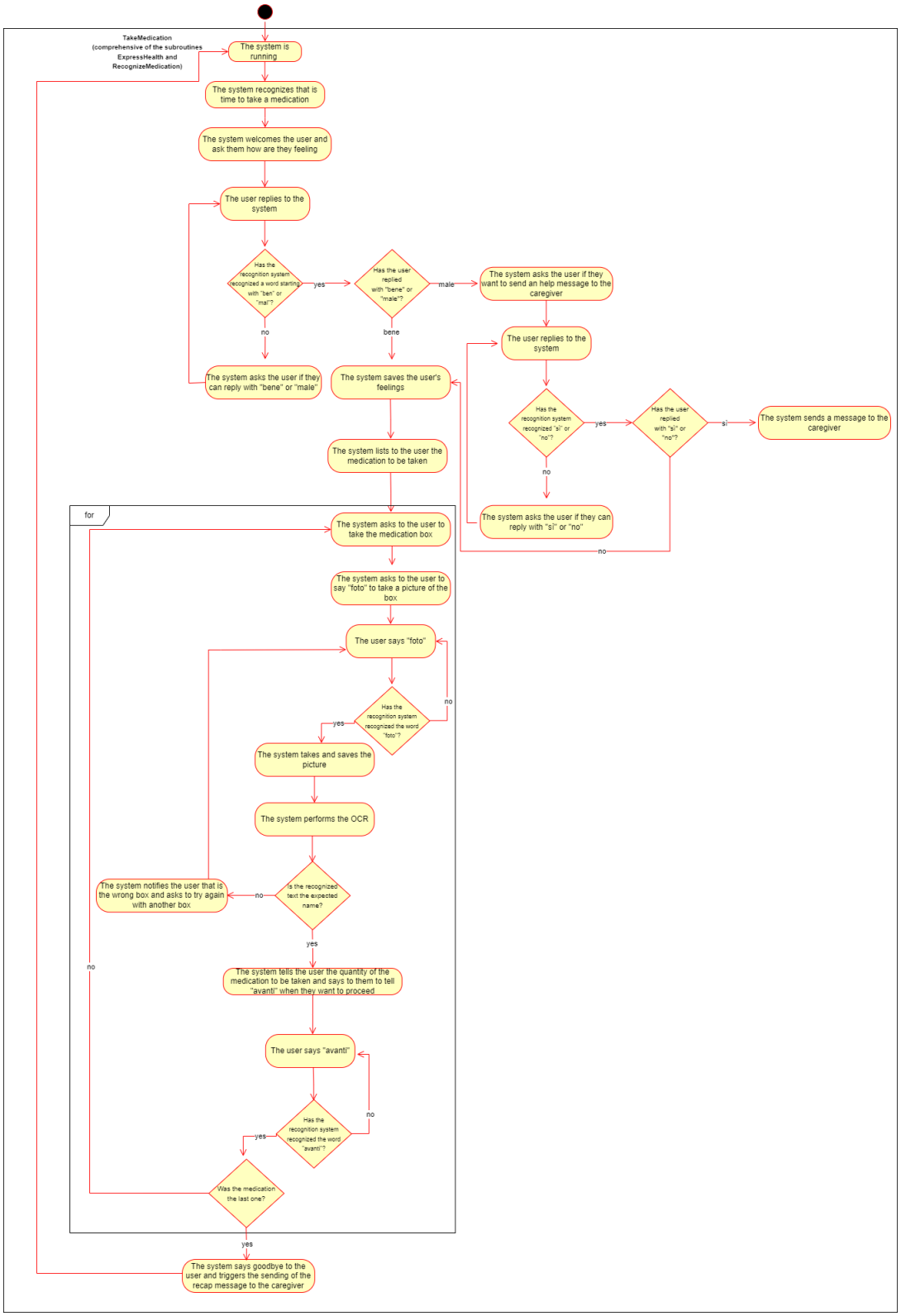
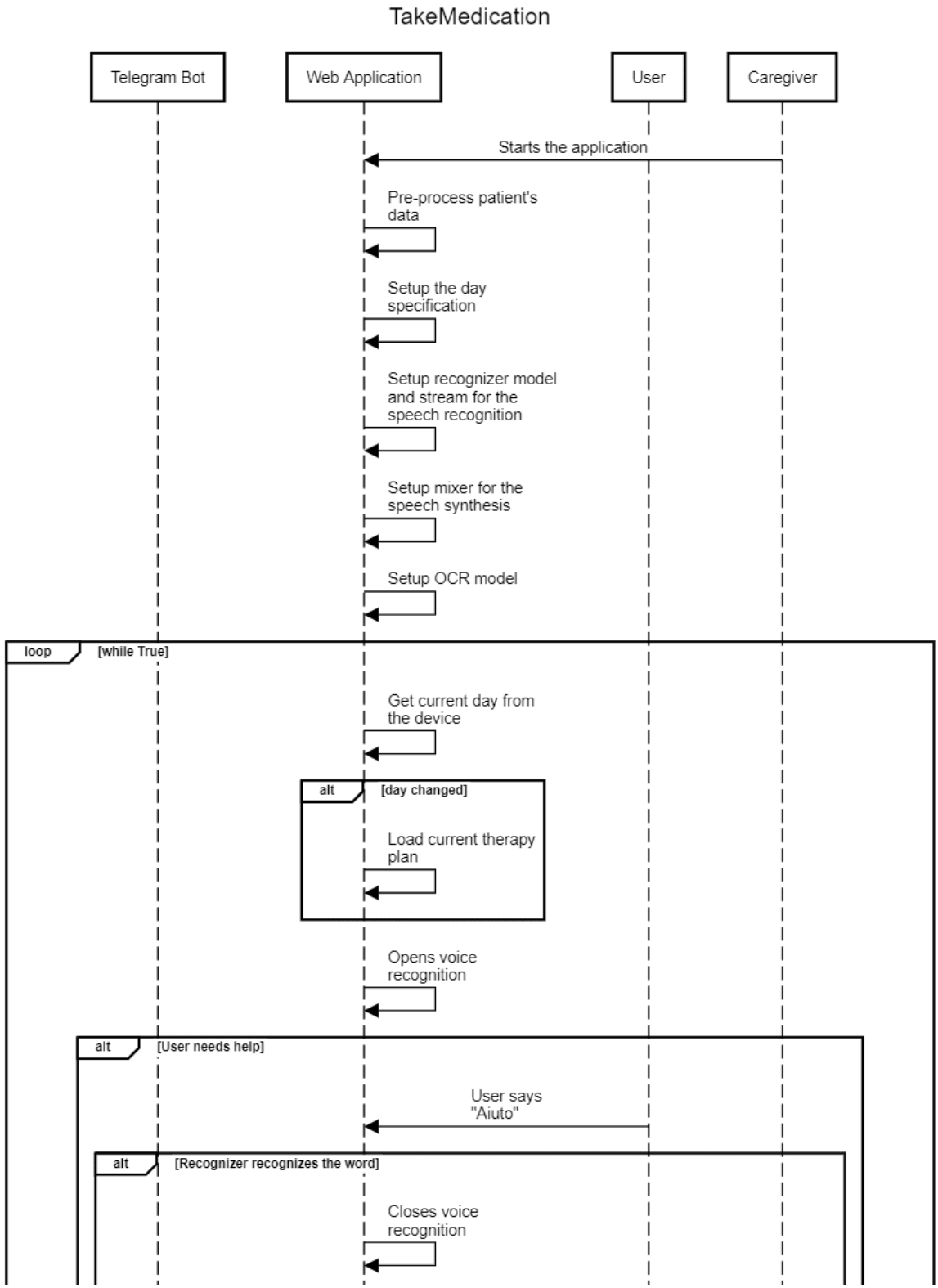
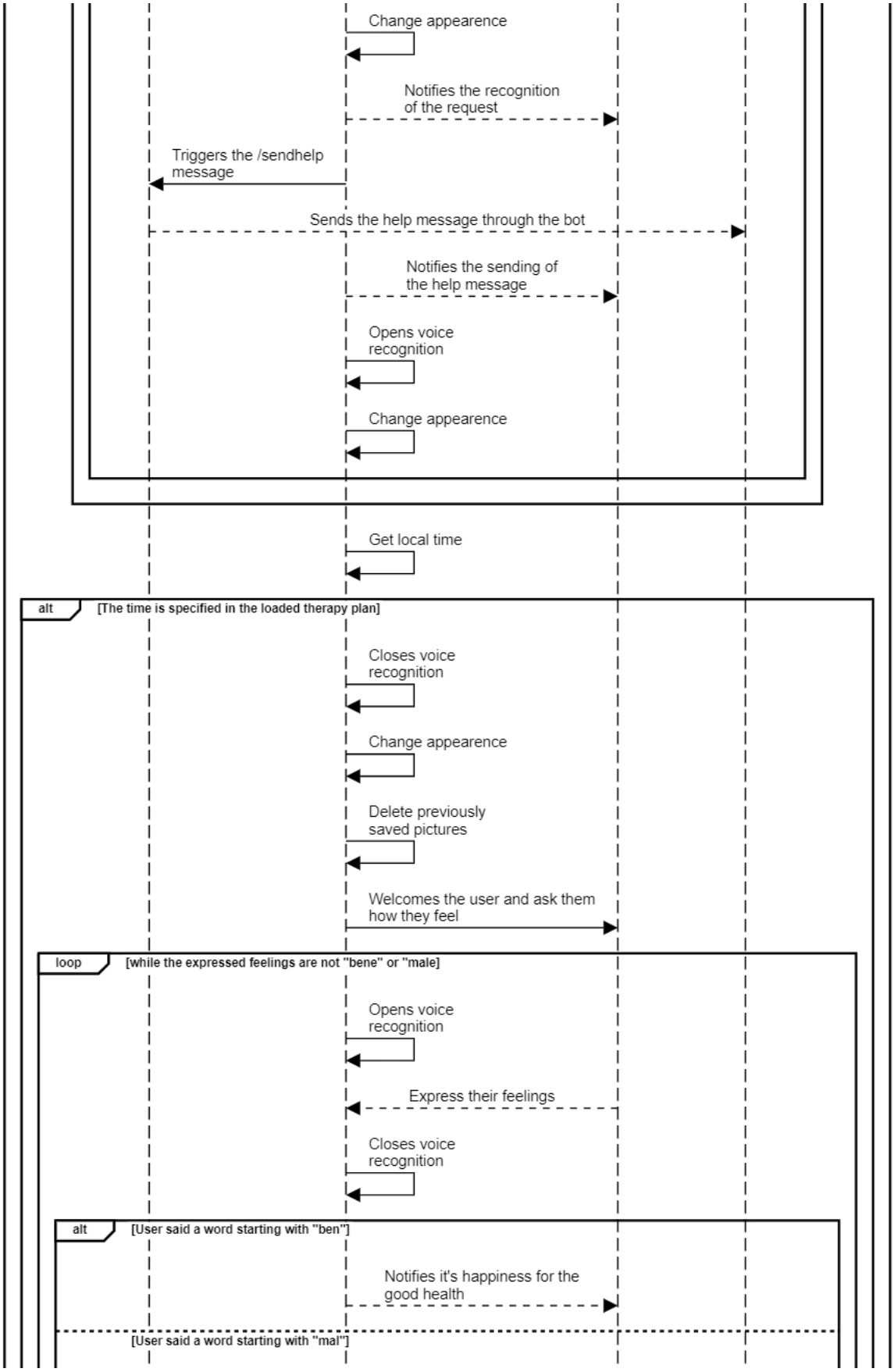


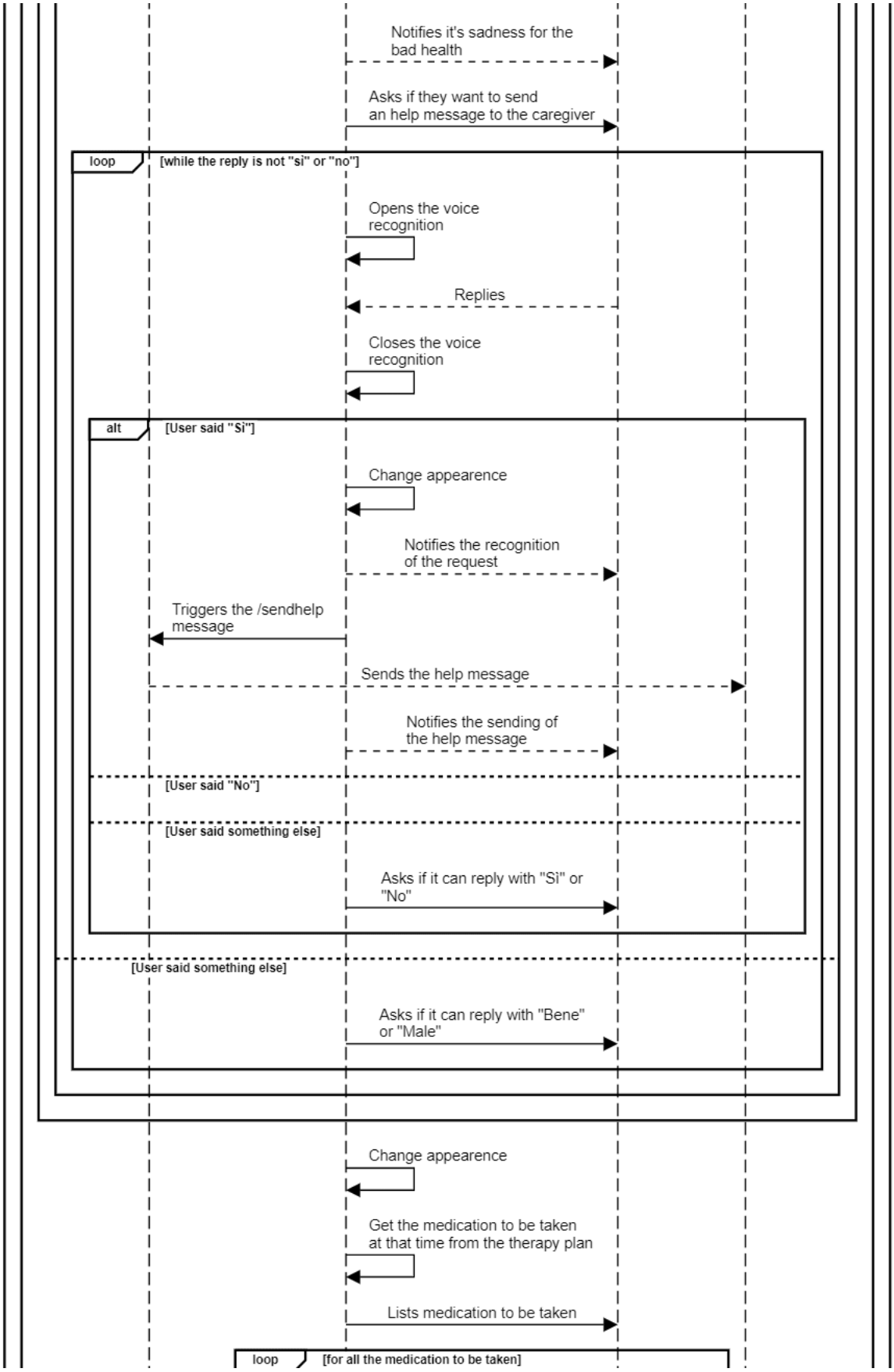
Figure 10: TakeMedication (with sub-routines ExpressHealth and RecognizeMedication) Activity Diagram

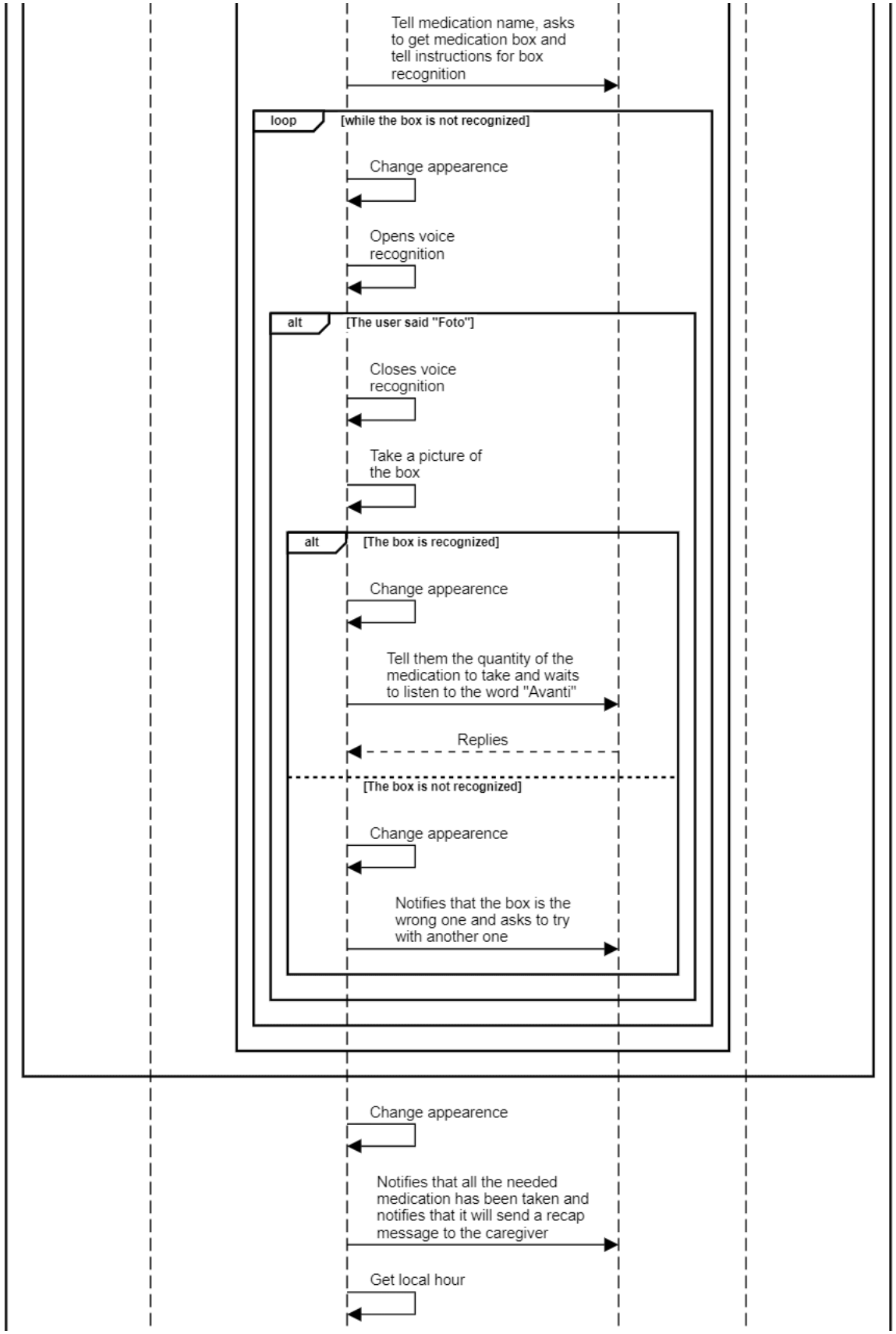
4.4 Sequence Diagrams

In Figure 11 is presented the sequence diagram of the main interaction with the system (whose execution flow is presented in Figure 10)




Change appearence
Notifies the recognition of the request
Triggers the /sendhelp message
Sends the help message through the bot
Notifies the sending of the help message
Opens voice recognition
Change appearence
Get local time
alt
[The time is specified in the loaded therapy plan]
Closes voice recognition
Change appearence
Delete previously saved pictures
Welcomes the user and ask them how they feel
loop
[while the expressed feelings are not "bene" or "male]
Opens voice recognition
Express their feelings
Closes voice recognition
alt
[User said a word starting with "ben"]
Notifies it's happiness for the good health
[User said a word starting with "mal"]


Notifies it's sadness for the bad health
Asks if they want to send an help message to the caregiver
loop
[while the reply is not "si" or "no"]
Opens the voice recognition
Replies
Closes the voice recognition
alt
[User said "Si"]
Change appearence
Notifies the recognition of the request
Triggers the /sendhelp message
Sends the help message
Notifies the sending of the help message
[User said "No"]
[User said something else]
Asks if it can reply with "Si" or "No"
[User said something else]
Asks if it can reply with "Bene" or "Male"
Change appearence
Get the medication to be taken at that time from the therapy plan
Lists medication to be taken
loop
[for all the medication to be taken]


Tell medication name, asks to get medication box and tell instructions for box recognition
loop
[while the box is not recognized]
Change appearence
Opens voice recognition
alt
[The user said "Foto"]
Closes voice recognition
Take a picture of the box
alt
[The box is recognized]
Change appearence
Tell them the quantity of the medication to take and waits to listen to the word "Avanti"
Replies
[The box is not recognized]
Change appearence
Notifies that the box is the wrong one and asks to try with another one
Change appearence
Notifies that all the needed medication has been taken and notifies that it will send a recap message to the caregiver
Get local hour

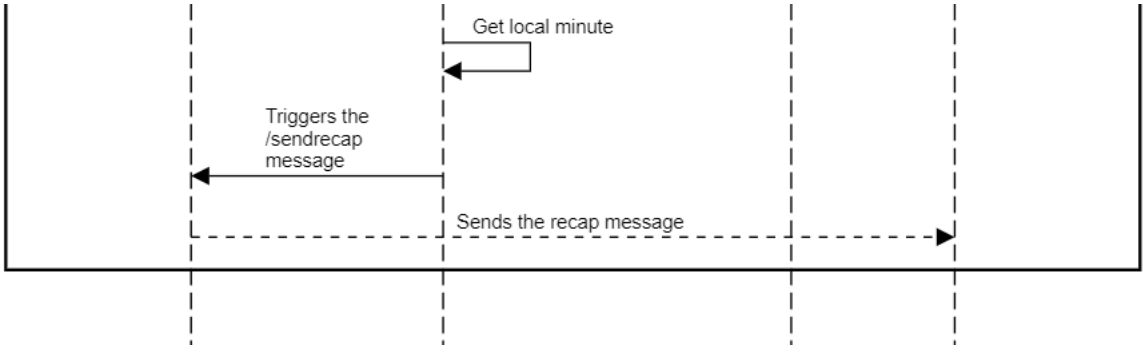


Figure 11: System's Sequence Diagram

5 Implementation

5.1 Libraries Used

The project lays its foundations on the usage of several libraries; for each of the chosen ones it will be now said for what they are used and why it was decided to use them.

5.1.1 Web Application

In the following section are presented the libraries used in the Web Application component of the project:

- **Vosk**: library to implement the speech recognition. It has been chosen for its offline models that made possible the implementation of a fast speech recognition; moreover the chosen model is optimized to be used on embedded system and given the nature of the future work (subsection 8.2) this has been chosen as the best library for the speech recognition. It is important to notice that in order to overcome difficulties in the recognition due to a not-so-well English pronunciation, the Italian model has been chosen for the recognition;
- **PyAudio**: library to access to the device microphone through a python script;
- **io**: library needed to store the audio from the stream into a .mp3 file;
- **PyGame**: library used to access to a real-time mixer and thus avoiding the necessity to save all the streams for the speech recognition, this solution has been thought in order to make the speech recognition as fast as possible;
- **PaddleOCR**: library to perform OCR; the Paddle OCR, based on the PaddlePaddle deep learning platform is at the moment one of the most accurate open source OCR available and also provides a recognizer model for the Italian language;
- **gTTS**: library to perform the speech synthesis. Even though is an online one is one of the best synthesizer available minimizing the "robot-like" voice and having an accurate pronunciation of the words even in Italian;
- **Pandas**: utility library used to manipulate CSV files;
- **dotenv**: utility library used to open .env files;
- **requests**: library used to send HTTPS requests to the Telegram API in order to use the Telegram Bot;
- **datetime**: utility library used to retrieve the current time information from the device;

- **time**: utility library used to set sleeps in the execution when needed;
- **OpenCV**: library needed to access to the webcam through the python script and to perform the needed image pre-process by the PaddleOCR library;
- **thefuzz**: library needed to automatize the process of the string matching;
- **os**: utility library used to access to all the os functions throught the python script;
- **Flask**: library used to develop a simple and intuitive web application able to integrate with the functionalities of the system (also written in Python);
- **flask_socketio**: library used to pass information among the functionality layer and the user interface one making in this way possible the dynamic change of the UI;
- **threading**: utility library needed to execute the different part of the web application in different threads.

5.1.2 Telegram Bot

In the following section are presented the libraries used in the Telegram Bot component of the project:

- **os**: utility library used to access to all the os functions throught the python script;
- **dotenv**: utility library used to open .env files;
- **Telethon**: MTProto library to interact with Telegram's API as a user or through a bot account (bot API alternative); Telethon has been chosen among the several possible libraries because it uses the MTProto (Telegram communication protocol) as communication protocol minimizing in this way all the waits in the interaction; moreover Telethon is a highly user-friendly library to use;
- **asyncio**: utility library to create asynchronous functions in Python;
- **re**: utility library to use regular expressions (particularly useful to define the message triggers).

5.2 New Code

Aside from the use of the library mentioned in subsection 5.1, the entire code for the interaction in system, the UI and the Telegram Bot is newly developed; in the following sections for each one of the components will be reported the newly written functions

and the goal for the functions (notice that no arguments will be mentioned since they can be found in the class diagram depicted in Figure 3)

5.2.1 Web Application

In the following section is presented the newly written code for the Web Application. In order to maintain the logic that has been used in the report also here the functions will be divided into the **functionalities** ones and the **UI** ones.

- **Functionalities**
  - **setup_speech_recognition**: function needed to load the speech recognition model and to setup the microphone of the device;
  - **setup_speech_synthesis**: function needed to setup the mixer of the device;
  - **setup_ocr**: function needed to setup the OCR model;
  - **recognize_speech**: function needed to capture the audio stream and process the recognizer results;
  - **synthesize_speech**: function needed to access the synthesization of the given text;
  - **play_speech**: function needed to play the synthesized text;
  - **get_bot_data**: function needed to retrieve the data from the Telegram Bot; the data retrieved are the messages that user written to the bot together with the writer information, since the users can only write standard bot commands (i.e. */start*, */help*) no sensitive information can be retrieved in this way;
  - **get_chat_id**: function thanks to which is possible to retrieve the chat id for every specific user with the bot, in this way is possible to send private message from the bot to the wanted users;
  - **get_patient_data**: function to pre-process the data of the patient available in the *patient_registry.csv* file so to use them in the application;
  - **get_therapy_plan**: function to pre-process the data of the therapy plan for a specific day of the week;
  - **speech_synthesis**: function to combine the synthesizing of a text and its reproduction;
  - **send_telegram_message**: function to send a specific trigger for the bot through the Telegram API;
  - **send_help_message**: specific function to send the trigger for a help message; the function also takes account to the interaction with the user by specifying them all the passages of the consignment;

- **greet_patient**: function to specify the text to say hello to a patient;
- **speech_therapy_plan_info**: function to automatize the creation of the text to be synthesized that is used to explain to the user which medication they have to take at a specific time;
- **analyze_feelings**: function used to manage the interaction with the user when the system ask them how they feel;
- **speech_medication_instruction**: function to automatize the creation of the text that is used to tell the user which medication box should they take for the picture;
- **take_picture**: function used to capture and save a picture of the medication box;
- **recognize_medication**: function used to perform the OCR on the taken picture and to perform the string matching with the provided medication name in the therapy plan;
- **get_medication_instructions**: function to automatize the creation of the text to be synthesized that is used to tell the user for each medication the quantity that must be taken;
- **goodbye_patient**: function needed to automatize the creation of the text that is used by the system to say goodbye to the user at the end of the routine for the medication take;
- **send_recap_message**: function needed to trigger the send of the recap message by the Telegram bot;
- **delete_images**: utility function needed to delete the previously taken picture at the start of the loop for the medication take;
- **interaction**: main function of the functionalities, it's the one that connect all the previously specified functions.

- **UI**
  - **translate_day**: function needed to translate the English names of the days of the week into the Italian ones;
  - **get_current_day**: function needed to retrieve the local current timestamp of the device;
  - **get_therapy_plan_display**: function needed to correctly display the therapy plan in the UI;
  - **index**: function needed to specify the elements to be displayed in the index page of the web application;

- **current_time**: function needed to correctly put data about the current time of the device in the */current_time* location of the web application (needed in order to let the JavaScript functions work)
- **next_medication**: function needed to correctly put data about the next medication the user must take in the */next_medication* location of the web application (needed in order to let the JavaScript functions work)
- **CSS classes**: classes needed to style the elements of the web page;
- **JavaScript handlers**: handlers needed to dynamically change the web page appearance.

5.2.2 Telegram Bot

In the following section is presented the newly written code for the Telegram Bot.

- **main**: main function that contains all the logic for the Bot. It starts the client and then defines all the handler given the possible messages received; specifically:
  - pattern `/start`: replies with a welcome message;
  - pattern `/help`: replies with a description of the bot goal;
  - pattern `/sendhelp<([a-zA-Z' ]+)>`: replies with the help message for the patient with the name specified in the pattern;
  - pattern `/sendrecap<([a-zA-Z' ]+)><(bene|male)> <(monday|tuesday|wednesday|thursday|friday|saturday|sunday)-([01]?[0-9]|2[0-3]):([0-5][0-9])>`: replies with the recap message whose information are passed in the message trigger pattern.

6 Interface Evaluation (Nielsen Heuristics)

The interface evaluation is conducted by observing if it is compliant to the Nielsen Heuristics. For a completeness matter, the heuristics will be at first listed and explained (subsection 6.1) and then it will be checked if the produced interface respect them (subsection 6.2).

6.1 Nielsen Heuristics

Jakob Nielsen provided **ten principles** for interaction design that can be used in order to identify usability issues in user interfaces. The principles are the following [6]:

1. **Visibility of System Status**: the design should always keep users informed about what is going on, through appropriate feedback within a reasonable amount of time.

2. **Match Between the System and the Real World**: the design should speak the users' language. Use words, phrases, and concepts familiar to the user, rather than internal jargon. Follow real-world conventions, making information appear in a natural and logical order.

3. **User Control and Freedom**: users often perform actions by mistake. They need a clearly marked "emergency exit" to leave the unwanted action without having to go through an extended process.

4. **Consistency and Standards**: users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform and industry conventions.

5. **Error Prevention**: good error messages are important, but the best designs carefully prevent problems from occurring in the first place. Either eliminate error-prone conditions, or check for them and present users with a confirmation option before they commit to the action.

6. **Recognition Rather than Recall**: minimize the user's memory load by making elements, actions, and options visible. The user should not have to remember information from one part of the interface to another. Information required to use the design (e.g. field labels or menu items) should be visible or easily retrievable when needed.

7. **Flexibility and Efficiency of Use**: shortcuts — hidden from novice users — may speed up the interaction for the expert user so that the design can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

8. **Aesthetic and Minimalist Design**: interfaces should not contain information that is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility.

9. **Help Users Recognize, Diagnose, and Recover from Errors**: error messages should be expressed in plain language (no error codes), precisely indicate the problem, and constructively suggest a solution.

10. **Help and Documentation**: it's best if the system doesn't need any additional explanation. However, it may be necessary to provide documentation to help users understand how to complete their tasks.

6.2 Evaluation

The evaluation of the interface (whose screens are provided in subsection 7.2) will be conducted by checking how each heuristic is implemented:

1. **Visibility of System Status**: in the provided system the status is always visible, specifically in the idle screen in which is visible when the next step of the routine will take place (by indicating the time in which it will take place and the current time); moreover when each routine starts (either the one to send help or the one to take the medication) the system always inform the user about what it is going to happen by providing them visual feedback in the change of appearance and by providing them vocal feedback;

2. **Match Between the System and the Real World**: the system is designed to only communicate to the user with the natural language and is designed to explain to the user how everything it's happening; the system uses only concepts familiar to the user like *messages*, *medication boxes*, *feelings*;

3. **User Control and Freedom**: no extended process can be activated by the users by mistake, the most extended process is always started automatically by the system and as for the help request routine is a short one that cannot be early ended (it is thought to not be blocked and to not ask further confirmations because it's an emergency routine);

4. **Consistency and Standards**: there are no duplicated commands, situations or actions in the interface and all the execution flows are deeply explained to the user while in the execution;

5. **Error Prevention**: all the possible user errors (either the the pronunciation of the wrong word or the wrong recognition of the words) are catch in the sense that the routine will go on and on up until the right command is not given; every

time that is given to the user the possibility to answer with a wrong command and the user do that the system notify them the error and provides a solution (e.g. when it is given to the user the possibility to choose how they can reply, for example when it is asked them how they feel, if the user reply with something different than a word that starts with `ben` or `mal` the system ask them to reply only with `bene` or `male`; in other cases, when the system specifically ask the user to say a specific word, for example when it is time to take a picture and the system asks the user to say `foto`, the system simply waits for the user to say it);

6. **Recognition Rather than Recall**: no things must be remembered by the user, the system, either with vocal feedback or textual ones, always say to the user which are their possibilities or which are the things that must be done in order to continue the routines;

7. **Flexibility and Efficiency of Use**: given the target and the goal of the system no shortcuts were provided;

8. **Aesthetic and Minimalist Design**: all the screens are minimalist; as for the idle screen it only provides to the user useful information about their daily therapy plan, the next medication to be taken, the current day and hours and a reminder about the possibility to send an help request through a specific command. All the other screens are only a visual feedback that reacts to the behavior of the user;

9. **Help Users Recognize, Diagnose, and Recover from Errors**: all the errors are instantly notified to the users and for each one of them the system also takes account to give the user a solution for them;

10. **Help and Documentation**: the additional documentation on how to perform actions are given to the user through the vocal feedback during the routine and thanks to the textual feedback given in the idle screen about the next medication to be taken and about the instruction to send an help request;

7 Scenarios

In the following sections are presented all the possible usage scenarios of the system. The presentation will be divided into two parts, one for the scenarios of the caregiver (subsection 7.1) and one for the scenarios of the elder patient (subsection 7.2). For each scenario it will be presented the steps that the user will have to take and also the associated screens.

7.1 Caregiver Scenarios

The scenarios in which a caregiver can take part can be divided into the following categories: **first registration of the patient and the therapy plan**, **later update of the patient and the therapy plan**, **first configuration of the bot**, **later reading of the bot**, **first boot of the system**.

7.1.1 Registration of the patient and of the therapy plan

Before the first boot of the system it is asked to the caregiver of the patient to compile the files in which are stored the information about the patient and about the therapy that they have to follow.
In this first release, it is asked to the caregiver to compile several different CSV files (this procedure is subject to future improvements and changes as described in subsection 8.2) specifically the caregiver will have to compile the templates that are distributed together with the system in the following way:

- ***patient_registry.csv***: this file contains the personal information about the patient, it is asked to the caregiver to compile the columns *name*, *age*, *gender* and to compile (and add) as many columns as needed to specify all the **caregivers Telegram username** (columns *cg_handle_i*).
  The file is presented to the user like in Figure 12;

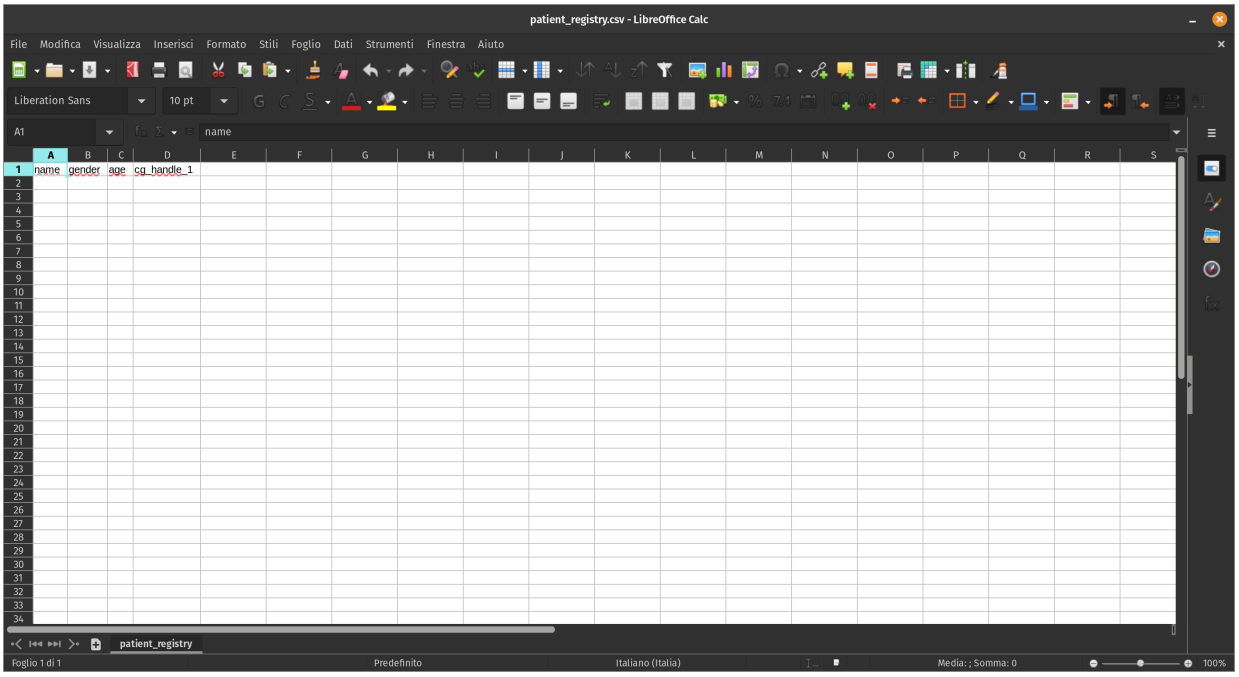


Figure 12: Sample patient_registry.csv file

- ***therapy_plan folder***: the folder contains 7 CSV files that represents the daily therapy plans of the patient. Each therapy plan is structured in the following way: there are 36 rows that must be filled with the information about the medication that the user must take at a specific hour; the files are pre-compiled in the hour column (going from 06:00 to 23:30) and the caregiver must fill the rows corresponding to the hours in which the patient must take a medication in the following way: they have to fill (and add if needed) couples of columns (with the names *medication_i*, *quantity_i*) in which they have to specify the medication name and the quantity of the medication that must be taken; it is important to notice that there are no limitations in how many medication the patient can take at a specific hour, it is just required for the caregiver to add the needed columns.
  The files (that have all the same template) are presented to the user like in Figure 13

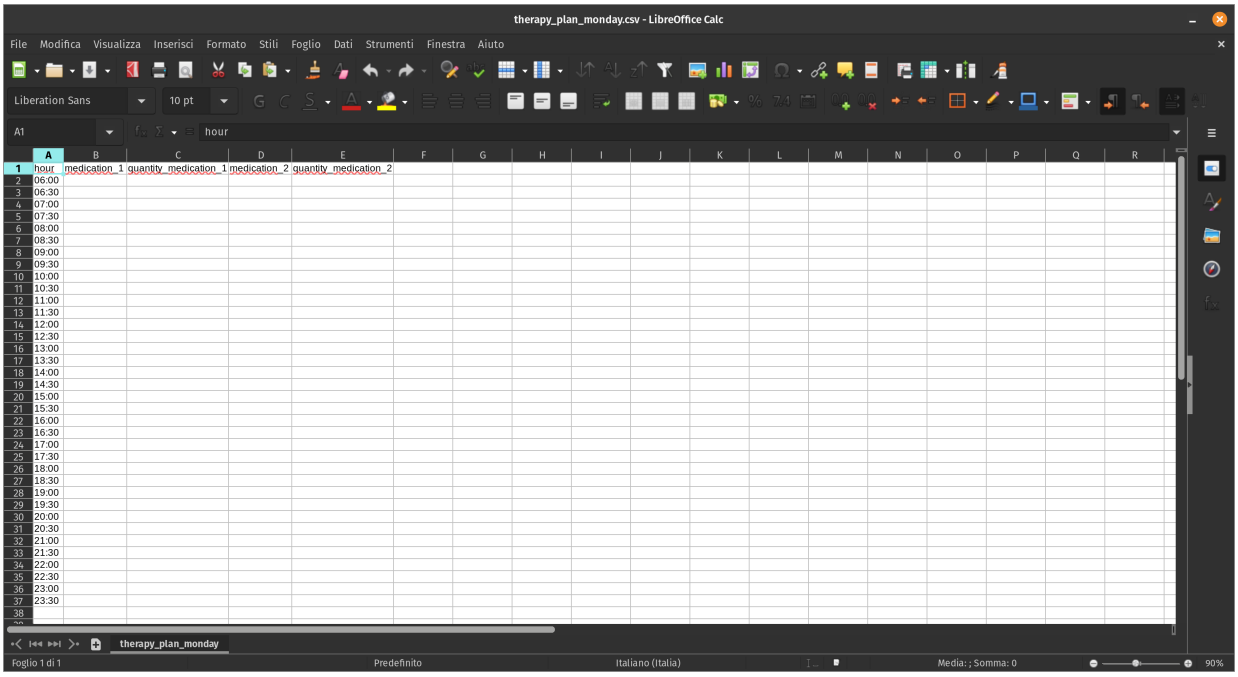


Figure 13: Sample therapy plan file

7.1.2 Update of the patient and of the therapy plan

It may happen that the caregiver has the need to update the information about the patient or, more probably, the information about the therapy plan. In this case it is asked to the user to stop the system, in case the system is already running, and then proceed with the update whose flow is similar to the one explained in subsubsection 7.1.1 for the registration. In this first release of the system it is asked to the user to manually stop the system process by pressing `CTRL+C` in the terminal. The screen presented to the user when they wants to stop the system is similar to the one depicted in Figure 14;

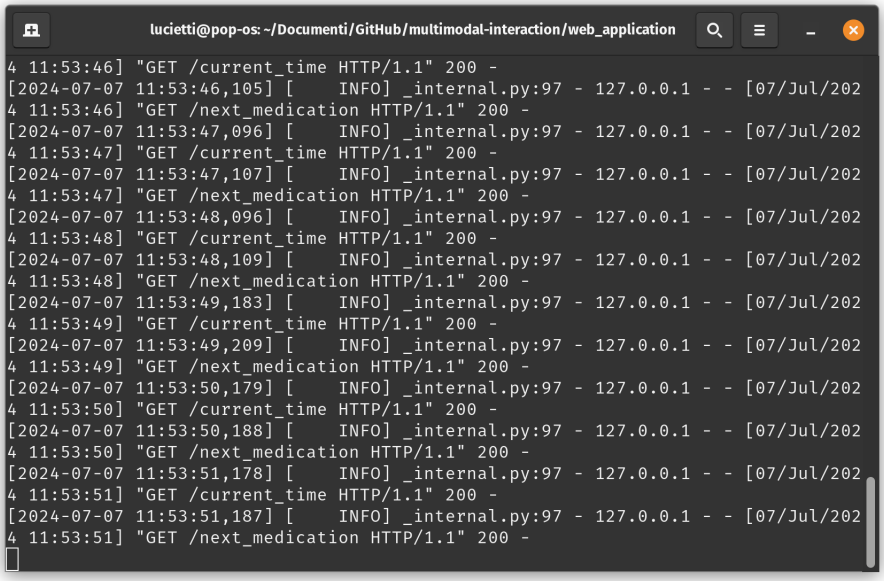


Figure 14: Sample screen when stopping the system

7.1.3 Configuration of the Telegram Bot

Before the first boot of the system it is also requested to the caregiver to activate the Telegram bot on their mobile phone. In order to do that the caregiver must at first check if it has an username set on Telegram (and if not they must set it). In order to find where to set it they must follow the passages depicted in Figure 15, Figure 16, Figure 17, Figure 18

Figure 15: Telegram Username Set Step 1

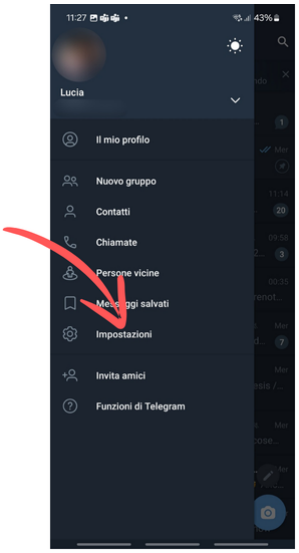


Figure 16: Telegram Username Set Step 2

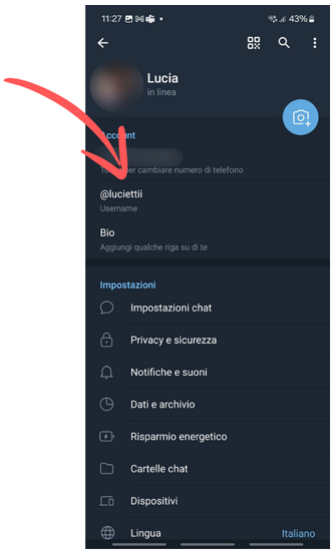


Figure 17: Telegram Username Set Step 3

Figure 18: Telegram Username Set Step 4

After that the caregiver is asked to search for the bot (called ***multimodal_patient_helper_bot***) in the Telegram search bar (Figure 19), select the bot and then click on the `AVVIA` button on the screen (Figure 20). By doing so the bot is correctly configured for the caregiver and it is ready to be used as an alert notification system.

Figure 19: Bot Search Step 1

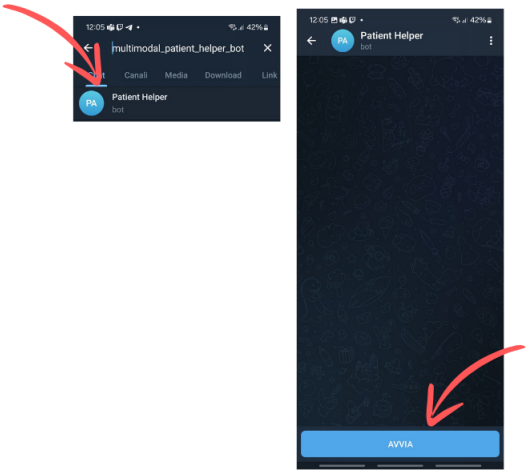


Figure 20: Bot Search Step 2 and 3

7.1.4 Reading of the Telegram Bot

After configuring the bot the caregiver will receive notification messages whenever the elder patient asks for help and for the recap of the medication taken. The messages have a standard form depicted in (Figure 21 and Figure 22)

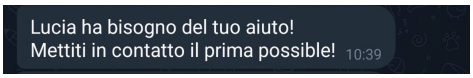


Figure 21: Sample Help Message

Figure 22: Sample Recap Message

7.1.5 First Boot of the System

When everything has been correctly configured the caregiver will be ready to first boot the system; given the state of the first release of the system it is required to go to the main folder of the system, open a terminal and then type `python3 app.py`, in this way the web application will be visible for the elder patient and all the recognizing systems will be booted (Figure 23).

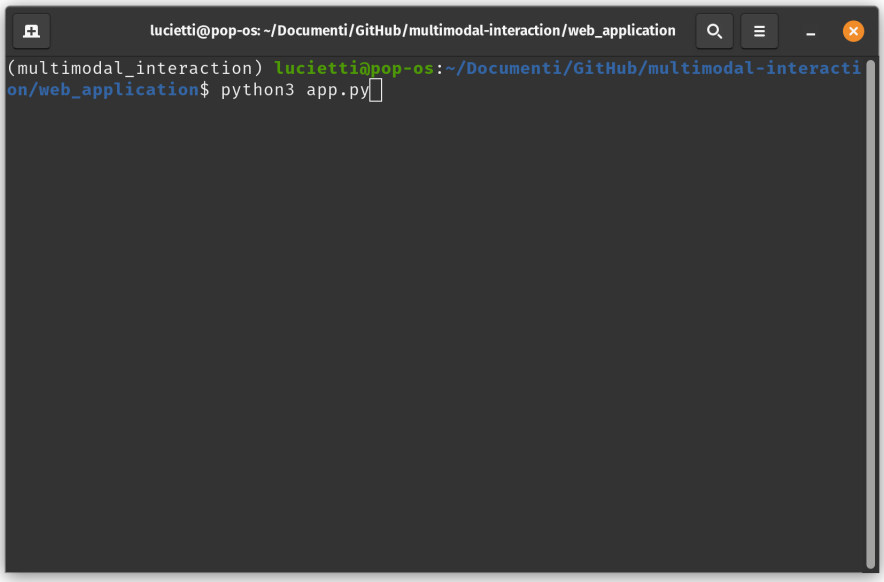


Figure 23: Sample start system screen

7.2 Patient Scenarios

The scenarios in which a patient can take part can be divided into the following categories: **asking for help**, **taking medication**.

7.2.1 Asking For Help

After the boot of the system the patient will have the possibility to ask for help in any moment in which the system is in idle (waiting for the moment in which the patient must take a medication). When the system is in idle it is presented to the user a screen like the one depicted in Figure 24: by saying the word `aiuto` the user will be able to trigger the automatic send of an help message through the bot: the web application will change screen (and become like the screen depicted in Figure 25) and the system will notify the user about every step is taking (meaning it will tell them that is sending the message and will notify them when the message has been sent). After the send of the help message the system will again enter into the idle state (Figure 24).

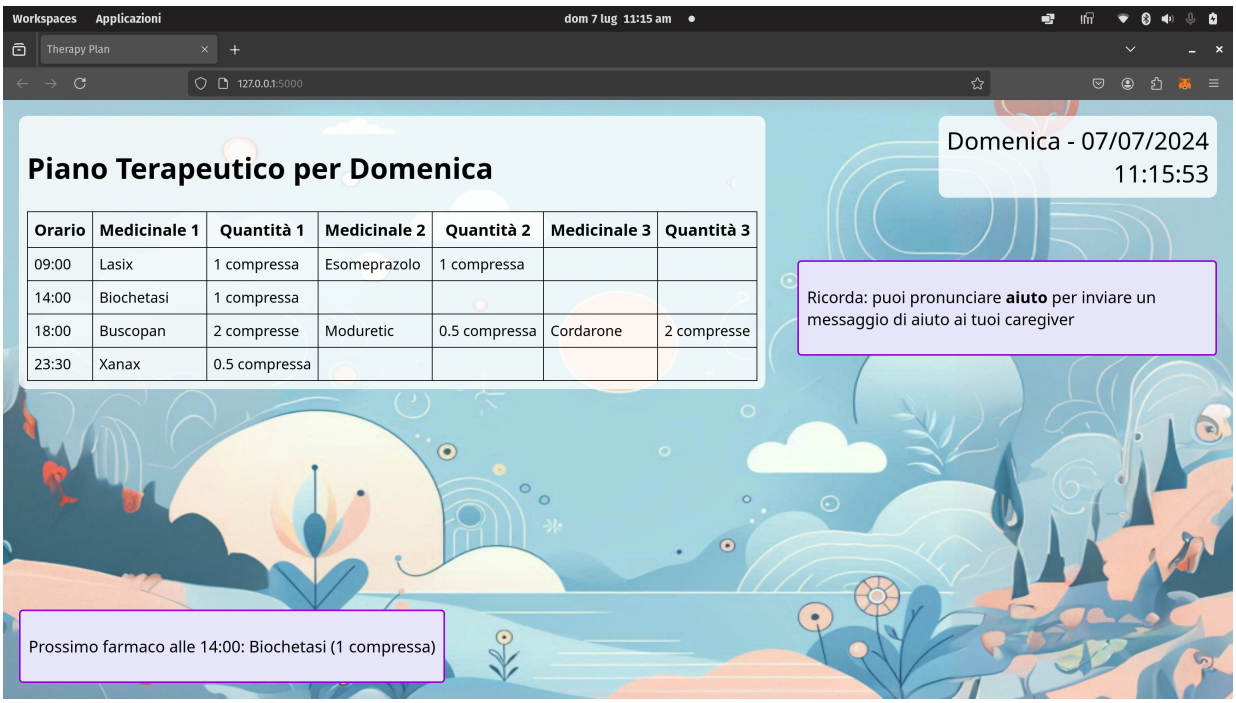


Figure 24: Web Application's idle screen

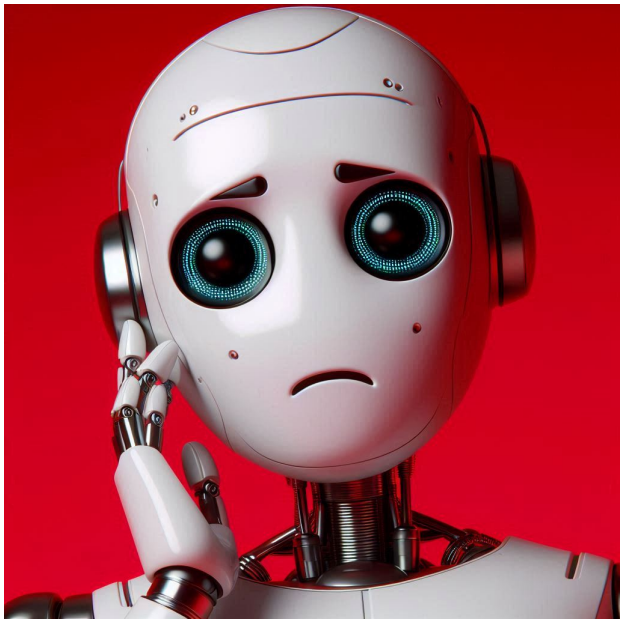


Figure 25: Web Application's alert screen

7.2.2 Taking Medication

When the system recognize that is time for the patient to take a medication it will start the procedure to help the patient take the medication. First of all the system will change appearance and greet the patient (Figure 26), it will ask them how they feel and behave accordingly to the reply of the user (Figure 27 or Figure 28 accordingly to the fact that the user says that they are feeling good or bad, in the case in which the user feels bad and wants to send an help message to the caregiver during the send the screen will become like in Figure 25). After that it will start the recognition of the medication box and also there behave accordingly to the user information (specifically it will have Figure 26 when giving instructions, Figure 29 when is ready to capture the image and Figure 27 or Figure 28 accordingly to the fact that the medication box is the correct or wrong one). After checking that all the needed medication were taken the system will say goodbye (by changing to the screen like in Figure 26) to the user and send the recap message to the caregiver. After that the system will again enter into an idle state (Figure 24).

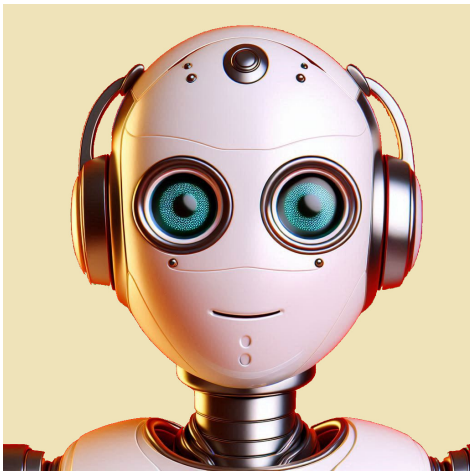


Figure 26: Web Application's screen to take a medication

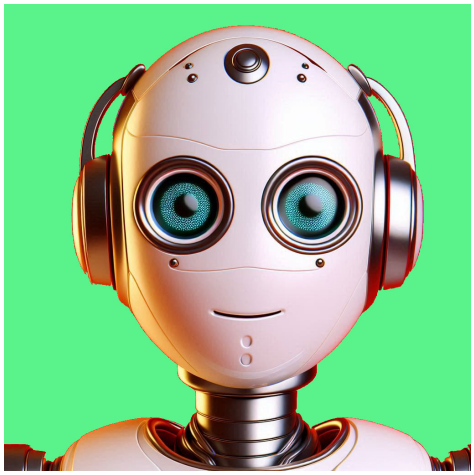


Figure 27: Web Application's screen for positive user behaviors

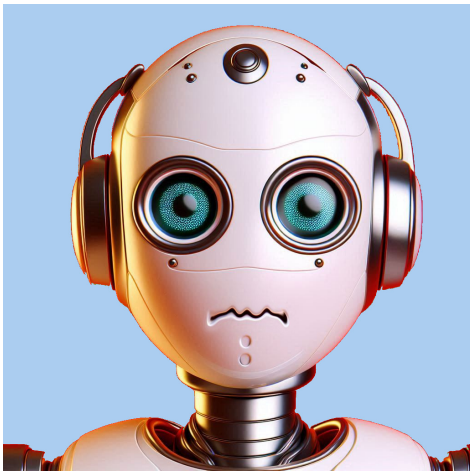


Figure 28: Web Application's screen for negative user behaviors

Figure 29: Web Application's screen for image taking

8 Limitations and Future Developments

8.1 Limitations

Since the system is still in an early stage of the development there are naturally several limitations that will be mitigated in future version of the project.
The biggest limitations found are the following:

- **Slow voice recognition**: even though the voice recognition is always correct (meaning that when a word is recognized is always exactly the word pronounced) it may happen that sometimes it is required to the user to tell the word more times before it is actually recognized. During the tests the words that gave much this problem were the ones starting with the letter `s` (in fact the command to take a picture was actually changed from `scatta` to `foto`) and the ones starting with letter `b`;

- **Voice recognition setting**: in order to let the voice recognition work it is required to the user to speak loudly and in a relatively silent setting;

- **Speech synthesis with wrong accents**: it may happen that in long sentences the speech synthesis get a few pronunciations wrong; this was not spotted every time for specific words but only sometimes in the context of a long sentence (for example the word `caregiver` is always pronounced correctly with the exception of the sentence used to say goodbye to the patient);

- **Difficulty for the OCR to recognize text written not horizontally**: it has been found during a test with a medication box with the name of the medication written obliquely that the OCR is not able to completely capture the name of the medication; it was tried to mitigate this limitation with the use of the fuzzer but using a sufficient high threshold in order to have correct results did not let the string matching in this case have a positive result.

8.2 Future Developments

The future development for the project are the following one:

- **Creation of a robot as interface for the patient**: thanks to the use of micro controllers (from a first fast search the Arduino Giga R1 seems to be a good choice) it will be developed a new interface of the system for the patient in the form of a robot with a screen that will give, together with the vocal feedback, also a visual feedback through the introduction of robots movements;

- **Creation of an easier system to compile and update the patient data**: in order to make the system more user friendly it will be developed a web portal

thanks to which the caregiver will have a better user interface to compile the information about the patient (and the therapy plan), the data collected in this way will be then sent to the main system that will use them as described in the report.

9 Conclusions

In this report it has been described the research and develop work that led to the production of the first iteration of an AAL system whose objective is to support the patient in the daily task of taking the medication.
After a brief description of the context in which the system wants to be inserted, it has been described all the principal use-cases thought for the first version of the system; it was then described the architecture of the presented prototype and it were presented all the libraries and the code written in order to let the system work. Then it was presented a brief evaluation of the prototyped interface by following the Nielsen Heuristics and it has been presented all the main scenarios in which every type of user can found themselves while using the system. Finally a description of the limitation of this first version of the system and a brief description of the future developments has been presented.

References

[1] Istat. https://www.istat.it/it/archivio/259588. Accessed: 2024-06-30.

[2] Shengzhi Wang, Khalisa Bolling, Wenlin Mao, Jennifer Reichstadt, Dilip Jeste, Ho-Cheol Kim, and Camille Nebeker. Technology to support aging in place: Older adults' perspectives. *Healthcare (Basel)*, 7(2), April 2019.

[3] Cambridge Dictionary. https://dictionary.cambridge.org/dictionary/english/caregiver. Accessed: 2024-06-30.

[4] Yuning Du, Chenxia Li, Ruoyu Guo, Xiaoting Yin, Weiwei Liu, Jun Zhou, Yifan Bai, Zilin Yu, Yehua Yang, Qingqing Dang, and Haoshuang Wang. Pp-ocr: A practical ultra lightweight ocr system, 2020.

[5] Valentina Turrini. Digital 2024 - i dati italiani. https://wearesocial.com/it/blog/2024/02/digital-2024-i-dati-italiani/, Feb 2024. Accessed: 2024-07-04.

[6] Jakob Nielsen. 10 usability heuristics for user interface design. https://www.nngroup.com/articles/ten-usability-heuristics/, Feb 2024. Accessed: 2024-07-05.