EASY CARE

A multimodal web application for older adults

Giuseppe Giuliano – Matr: 1973880

giuliano.1973880@studenti.uniroma1.it

Sommario

1 Introduction

The following pages describe a web application which can be considered as a help to elderly people. Maintaining independence and staying organized can become increasingly challenging. The web application is specifically designed to support older adults by providing tools that simplify taking notes to remember during their daily routines, promote mental well-being, and facilitate health monitoring. The interface can be used/navigated either using voice interaction or using a keyboard and mouse, in this way this application aims to empower elderly users to use application according to their ability and needing.

The application, now, supports only Italian language voice commands interaction, further improvement can take into account a multilanguage voice interaction. For coherence with the application language also the feedback and labels are in Italian.

2 Web Application Overview

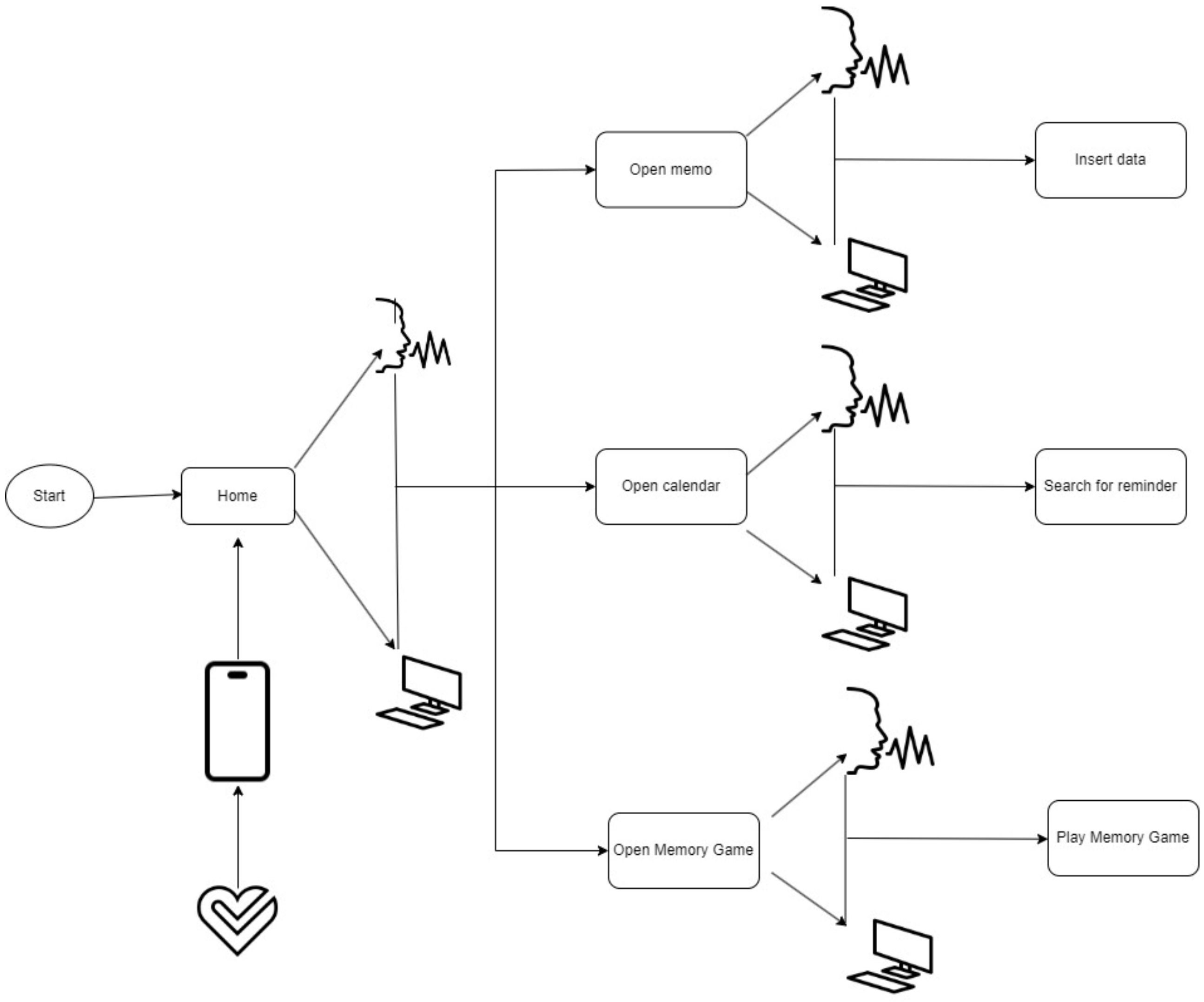


Figure 1- Web Application map interaction

2.1 Key Features

The following are the key features of the web application:

1. **Reminders and Calendar Integration**: The application enables users to easily create, view, and manage reminders for appointments, medication schedules, or important events. The integrated calendar provides a clear and straightforward overview, helping older adults stay organized without feeling overwhelmed.

2. **Memory Game**: To support cognitive health, the application includes a memory game tailored to engage and stimulate mental activity, helping users sharpen their memory in an enjoyable and interactive way.

3. **Health Data Visualization**: By connecting with Health Connection (Google Fit) on mobile devices, the application provides an accessible way to monitor average heart rate trends through visually intuitive graphs. This feature helps users and their caregivers stay informed about their cardiovascular health. The application by monitoring heart rate can warn the user and ask for sending a help message.

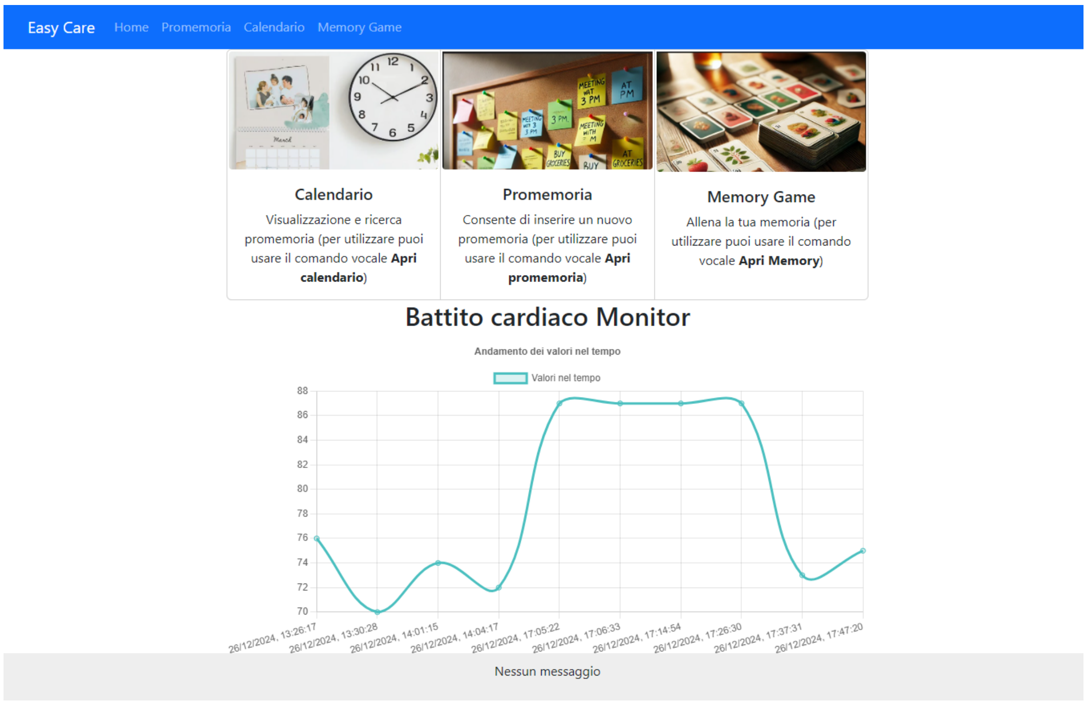
4. **Gesture Interaction:** In case of an emergency, the user can use a gesture-based interaction to send a command to the system, requesting the dispatch of a WhatsApp message containing a call for help.

2.2 User Interaction

The application interaction is designed to let the user use voice commands, allowed for a hands-free experience, making it particularly useful for users with mobility challenges or limited familiarity with technology. For those who prefer a more traditional method, the standard keyboard input ensures ease of use.

By focusing on the needs of older adults, this web application acts as an assistant that promotes mental sharpness, helps in scheduling reminders and monitor heart rate for uncommon episodes.

2.3 Home Page



Calendario

Visualizzazione e ricerca promemoria (per utilizzare puoi usare il comando vocale **Apri calendario**)

Promemoria

Consente di inserire un nuovo promemoria (per utilizzare puoi usare il comando vocale **Apri promemoria**)

Memory Game

Allena la tua memoria (per utilizzare puoi usare il comando vocale **Apri Memory**)

Battito cardiaco Monitor

Nessun messaggio

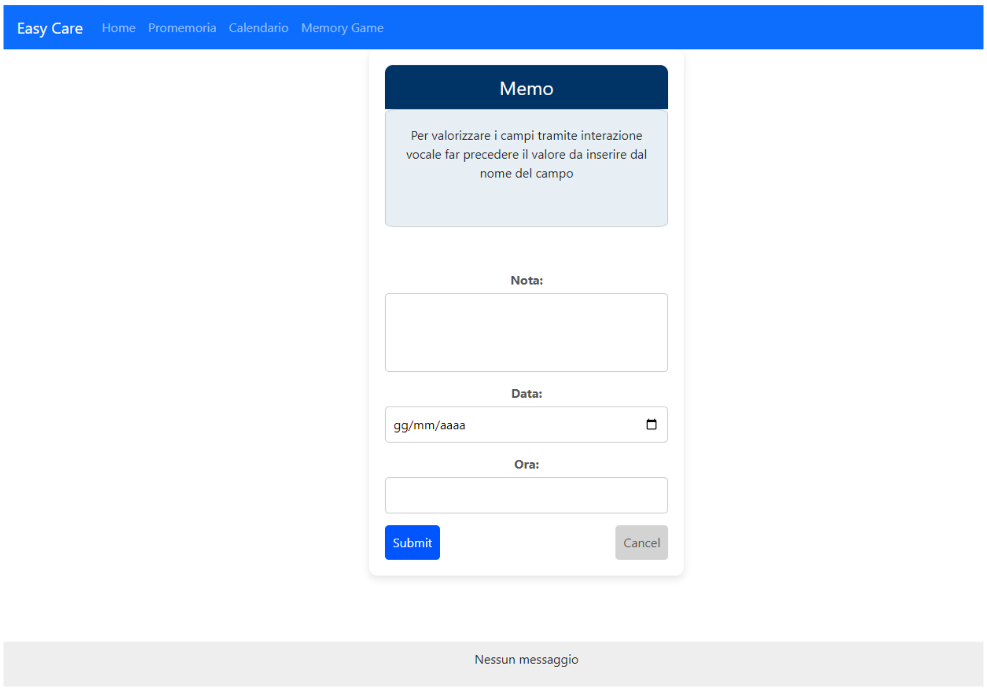
Figure 2 - Home Page

The Home page of the application serves as the central hub for users to access various features and monitor their health data. This page is designed to provide an overview of the application's functionalities and monitor health condition by checking the heart rates.

Key Sections:

- **Feature Navigation Cards**:
  - Calendar: Offers a shortcut to the calendar page for viewing and searching reminders. Users can utilize the voice command "Apri calendario" (Open calendar) to access this feature quickly or as an alternative the mouse can be used to perform the same action;
  - Memo: Provides access to the memo creation page for adding new reminders. Users can activate this feature using the voice command "Apri promemoria" (Open memo) or as an alternative the mouse can be used to perform the same action;
  - Memory Game: Directs users to a cognitive training game designed to improve memory retention. It can be accessed using the voice command "Apri Memory" (Open Memory) or as an alternative the mouse can be used to perform the same action;
- **Heart Rate Monitor**:
  - The lower section of the dashboard features a dynamic chart titled "Battito cardiaco Monitor" (Heart Rate Monitor). This graph visualizes the user's average heart rate data over time, collected from the Health Connection (Google Fit) on a mobile device.
  - The chart provides real-time insights into cardiovascular trends, allowing users to monitor their well-being effectively.
- **Informational Design**:
  - The dashboard is structured to offer clear, actionable pathways to each application feature. The integration of visual elements, such as feature-specific images and a graph, enhances usability and accessibility.
- **Voice Command Integration**:
  - Each feature supports voice commands, ensuring hands-free navigation for users who prefer or require this interaction method.
- **Speech synthesis:**
  - When the average heart rate recorded by the application over the past two days drops below 50 (bradycardia), the system uses voice synthesis to ask if the user wants to send a help request.
    - If the user responds "non chiedere aiuto" or "non chiamare" or "non richiedere" the voice alert will not be repeated;
    - if the user responds with the words "chiedi aiuto" or "invia messaggio" or "invia richiesta" to confirm a help request, the system sends a WhatsApp message to the designated emergency contact number.

2.4 Promemoria



Per valorizzare i campi tramite interazione vocale far precedere il valore da inserire dal nome del campo

Nota:

Data: gg/mm/aaaa

Ora:

Nessun messaggio

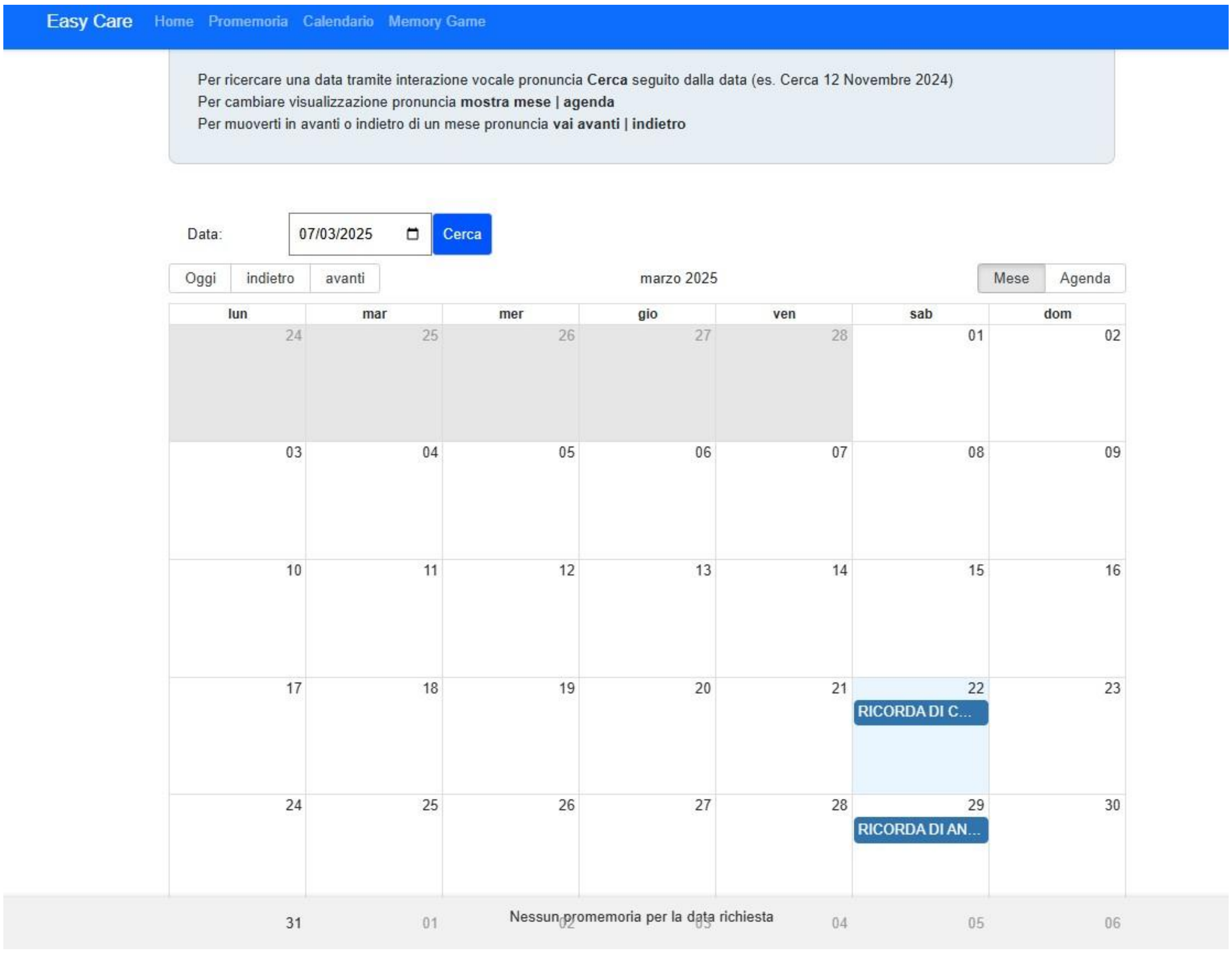
Figure 3 – Promemoria

The Memo page is designed to allow users to create and save reminders. This page provides an interface for inputting memo details and offers support for both voice commands and manual keyboard interaction.

Key Functionalities:

- **Input Fields:**
  - "Nota" (Note Field): A text box where users can enter the details or description of the reminder.
  - "Data" (Date Field): A calendar input to specify the date for the memo. Users can either type the date or select it from the dropdown calendar widget.
  - "Ora" (Time Field): A dedicated input box for specifying the time of the reminder, ensuring accurate scheduling.
- **Voice Command Integration:**
  - Users can populate the fields using voice commands by preceding the value with the field name. For example, saying "Nota ricorda di prendere le pillole" will automatically populate the note field with the text "ricorda di prendere le pillole."
- **Action Buttons:**
  - Submit Button: Saves the memo with the entered details and schedules it in the system.
  - Cancel Button: Clears all entered data and exits the memo creation process.
  - Voice command "Conferma" can be also used to save the memo.
- **Instructional Message:**
  - A prominent message at the top of the page guides users on how to use voice commands for populating the fields, ensuring accessibility for all users.

2.5 Calendario



Per ricercare una data tramite interazione vocale pronuncia **Cerca** seguito dalla data (es. Cerca 12 Novembre 2024)
Per cambiare visualizzazione pronuncia **mostra mese | agenda**
Per muoverti in avanti o indietro di un mese pronuncia **vai avanti | indietro**

Data: 07/03/2025

marzo 2025

lun	mar	mer	gio	ven	sab	dom
24	25	26	27	28	01	02
03	04	05	06	07	08	09
10	11	12	13	14	15	16
17	18	19	20	21	22 RICORDA DI C...	23
24	25	26	27	28	29 RICORDA DI AN...	30
31	01	02	03	04	05	06

Nessun promemoria per la data richiesta

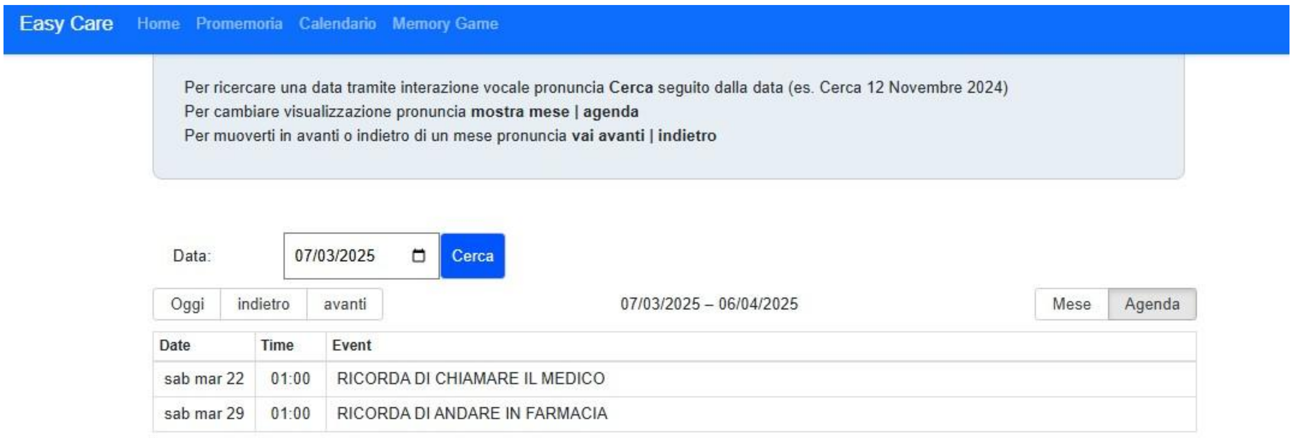
Figure 4 – Calendario – Month View

The page provides users with a view of their scheduled reminders and events. Users can interact with the calendar through both voice commands and keyboard inputs.

Key Functionalities:

- **Date Navigation:**
  - Users can navigate through different dates using the "Oggi", "Indietro" e "Avanti" buttons. These controls enable easy movement between months.
- **Reminder Display:**
  - Scheduled reminders are visually represented as blocks within the calendar grid. Each block displays a brief description of the reminder.
- **Search Feature:**
  - The search functionality allows users to find reminders by specifying a date using voice commands or manual input. For example, users can say "Cerca 12 Dicembre 2024" to locate reminders for that date.
- **View Modes:**
  - The calendar can be viewed in Month or Agenda, providing users with flexibility in how they visualize their schedules.
- **Empty Date Notification:**
  - If no reminders are set for a selected date, the application displays a message indicating "Nessun promemoria per la data richiesta" ("No reminders for the requested date") ensuring clarity for the user.

- **Speech synthesis:**
  - When a search is performed, the result is displayed in the footer of the page and also announced by the system using text-to-speech synthesis



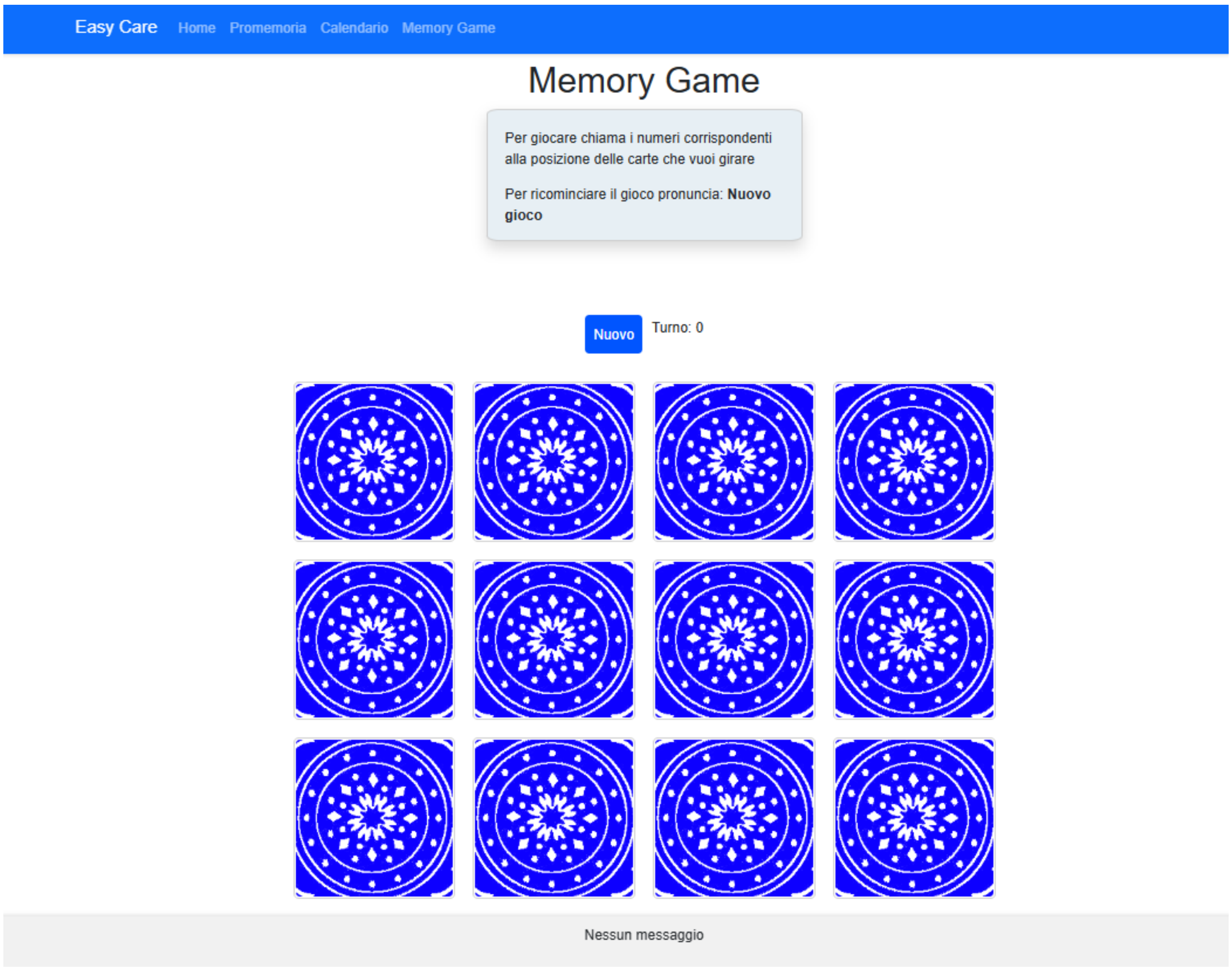
Per ricercare una data tramite interazione vocale pronuncia **Cerca** seguito dalla data (es. Cerca 12 Novembre 2024)
Per cambiare visualizzazione pronuncia **mostra mese | agenda**
Per muoverti in avanti o indietro di un mese pronuncia **vai avanti | indietro**

07/03/2025 – 06/04/2025

Date	Time	Event
sab mar 22	01:00	RICORDA DI CHIAMARE IL MEDICO
sab mar 29	01:00	RICORDA DI ANDARE IN FARMACIA

Figure 5 - Agenda VIew

2.6 Memory Game



Memory Game

Per giocare chiama i numeri corrispondenti alla posizione delle carte che vuoi girare

Per ricominciare il gioco pronuncia: **Nuovo gioco**

Nessun messaggio

Figure 6 - Memory game

The Memory Game page is designed to provide users with an engaging way to train their memory. This interactive game challenges users to match pairs of cards, promoting cognitive health and improving focus.

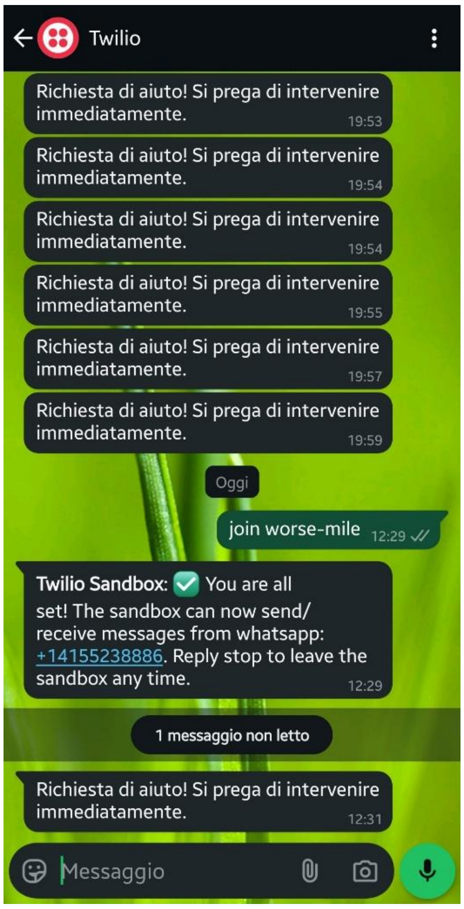
Key Features:

- **Game Instructions**:
  - An instruction box at the top of the page explains how to play the game:
    - Users call out the numbers corresponding to the positions of the cards they wish to flip, leveraging the application's voice command functionality.
    - To restart the game, users can say the voice command “Nuovo Gioco” (“New game”).
- **Game Grid**:
  - The main section of the page features a grid of cards, each displaying a uniform design initially.
  - When a card is selected (flipped), it reveals a hidden pattern or image. The objective is to find matching pairs by remembering the positions of the flipped cards.
- **Turn Counter**:
  - A Turn Counter is displayed to keep track of the number of attempts the user has made during the game, providing a measurable goal for improving performance.
- **Voice Command Integration**:
  - The game supports voice commands for flipping cards and restarting the game, enabling hands-free interaction and enhancing accessibility.
- **Real-Time Feedback**:
  - The game dynamically updates as the user flips cards, displaying matched pairs and tracking progress seamlessly.
- **Restart Option**:
  - Users can quickly reset the game at any time to start fresh, encouraging replayability and continuous engagement.

2.7 Hand Gesture Emergency Assistance

The web application includes a feature designed to enhance user safety and accessibility through gesture recognition.

When the user raises the hand, the system detects this gesture and automatically sends a WhatsApp message to a pre-specified phone number to request help. This functionality is particularly useful in emergency situations where immediate assistance is required, offering a hands-free and efficient way to communicate distress. By leveraging gesture recognition technology, the application ensures that users can seek help quickly, providing an added layer of security.


Twilio

Richiesta di aiuto! Si prega di intervenire immediatamente. 19:53

Richiesta di aiuto! Si prega di intervenire immediatamente. 19:54

Richiesta di aiuto! Si prega di intervenire immediatamente. 19:54

Richiesta di aiuto! Si prega di intervenire immediatamente. 19:55

Richiesta di aiuto! Si prega di intervenire immediatamente. 19:57

Richiesta di aiuto! Si prega di intervenire immediatamente. 19:59

Oggi

join worse-mile 12:29

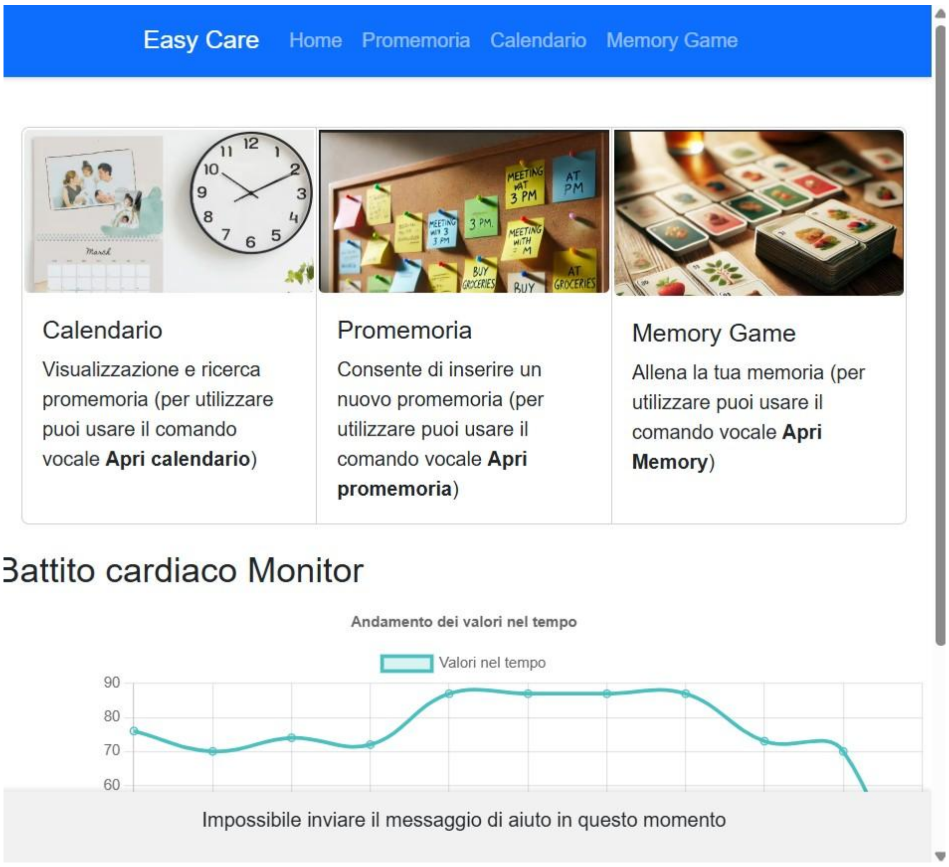
Twilio Sandbox: ✅ You are all set! The sandbox can now send/receive messages from whatsapp: +14155238886. Reply stop to leave the sandbox any time. 12:29

1 messaggio non letto

Richiesta di aiuto! Si prega di intervenire immediatamente. 12:31

Figure 7 - Help Message Received

The application, after a successful sent of the message, to avoid compulsive sends, checks for 5 minutes to let the user ask again for a help message to send otherwise it responds by warning that it's impossible to send another message.



Calendario

Visualizzazione e ricerca promemoria (per utilizzare puoi usare il comando vocale **Apri calendario**)

Promemoria

Consente di inserire un nuovo promemoria (per utilizzare puoi usare il comando vocale **Apri promemoria**)

Memory Game

Allena la tua memoria (per utilizzare puoi usare il comando vocale **Apri Memory**)

Battito cardiaco Monitor

Impossibile inviare il messaggio di aiuto in questo momento

Figure 8 - Warning Message Display

2.8 Mobile application

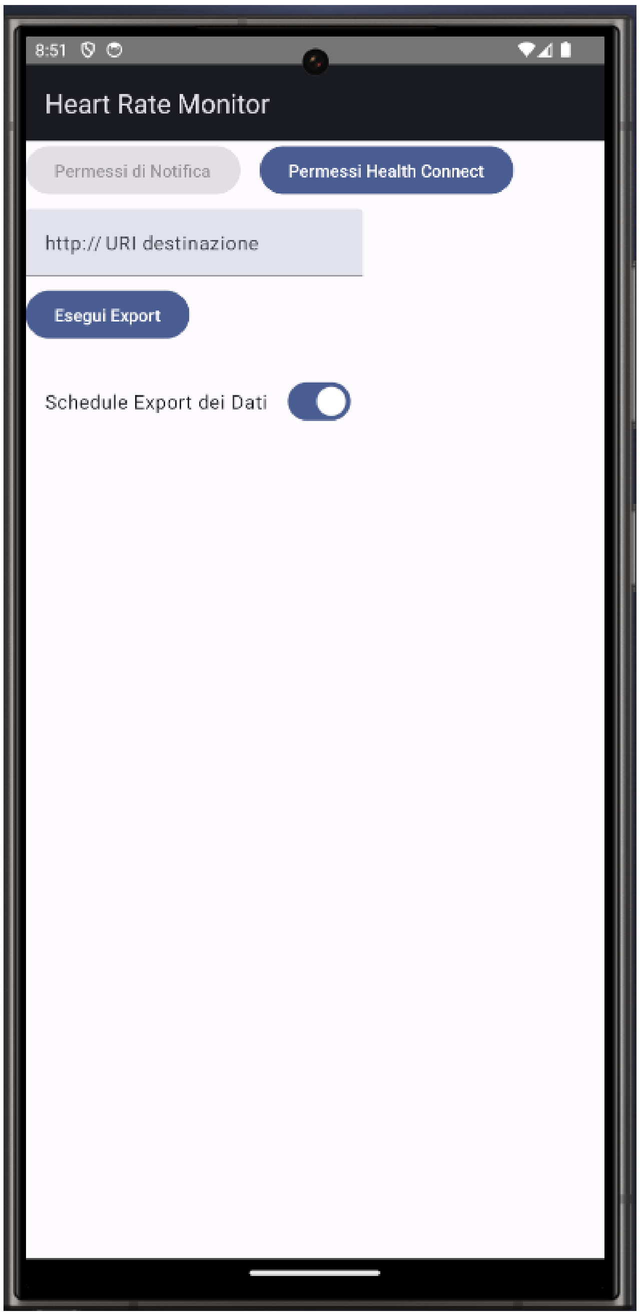


Figure 9 – Mobile

The mobile application let the user to collect his heart rate average beats and send them to the main web application.

Key features:

- **Permission Management**: Users must grant specific permissions to enable Health Connect integration or notifications.
- **Data Export Configuration**: The application provides an input field for setting a custom URI, allowing seamless integration with external platforms for advanced data analysis.
- **Manual and Scheduled Data Export**: Users can initiate data export manually using the 'Run Export' button or activate the 'Schedule Data Export' toggle for automated periodic exports, ensuring continuous and effortless data management.

3 Architecture

The system integrates different input modes:

- Voice can be used to navigate across the web pages, inputting data into the forms, playing the memory game, ask for sending a help message
- Navigate across pages and inputting data can also be done by the use of keyboard and mouse
- Video capture is also integrated into the system to trigger a help request if needed by monitoring a specific hand gesture

3.1 Software Overview

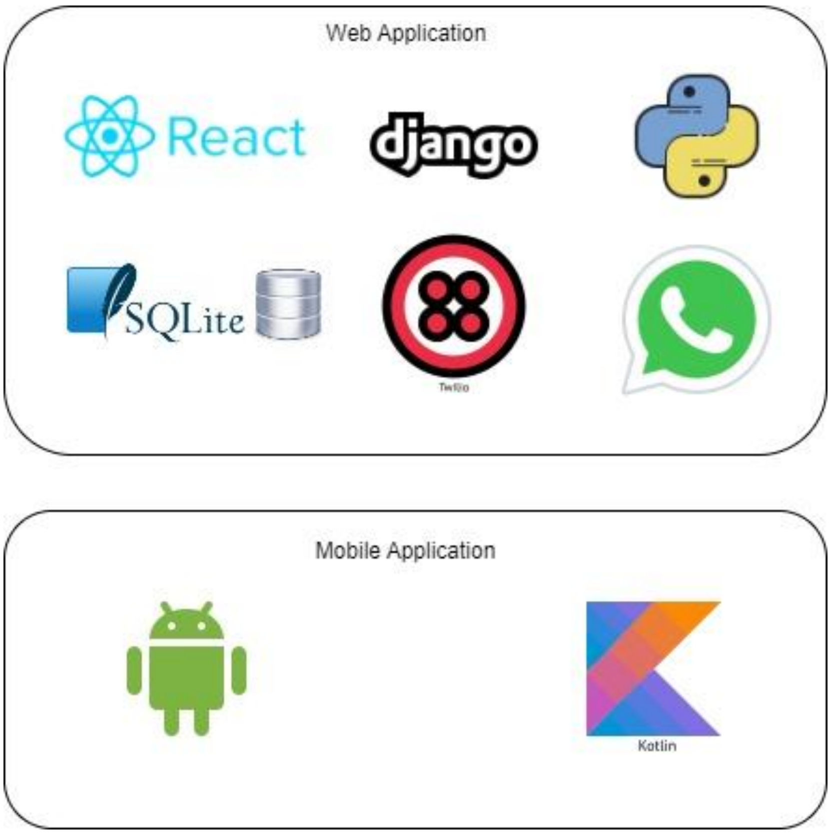


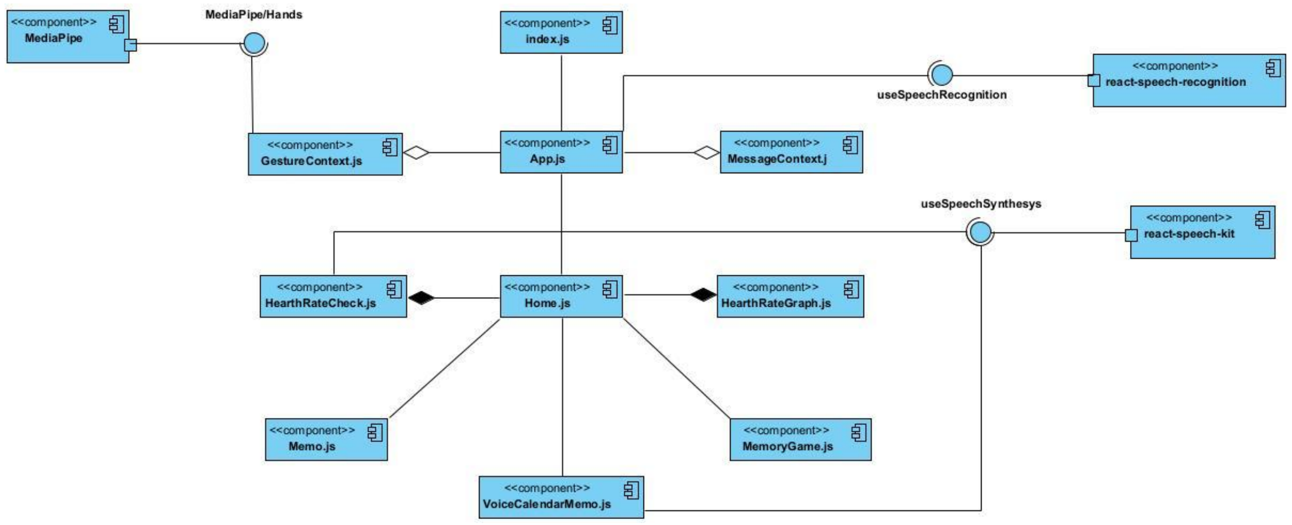
Figure 10 – Software Items

The software architecture is designed with two primary components: the **web application** and the **mobile application**.

The **web application** is divided into two distinct parts: the **frontend** and the **backend**.

- **Frontend:**
    - developed using **React**, a JavaScript library, to create a responsive, dynamic, and interactive user interface.
    - To manage speech recognition the **react-speech-recognition** component has been used. This library is a wrapper that simplifies the use of native browser speech recognition APIs, such as the **Web Speech API**. React-speech-recognition allows defining custom commands that trigger specific functions when recognized. While listening, the library handles the onresult event, which receives transcription results in real-time. The defined commands are compared against the transcribed text, when a command matches the text, its associated callback function is executed.
    - To manage speech synthesis the library **react-speech-kit** has been used

- **Mediapipe.js** has been imported into the React frontend part of the project to handle gesture recognition, used to capture the open hand to send an help message if needed.



<<component>> MediaPipe
MediaPipe/Hands
<<component>> index.js
useSpeechRecognition
<<component>> react-speech-recognition
<<component>> GestureContext.js
<<component>> App.js
<<component>> MessageContext.j
useSpeechSynthesys
<<component>> react-speech-kit
<<component>> HearthRateCheck.js
<<component>> Home.js
<<component>> HearthRateGraph.js
<<component>> Memo.js
<<component>> MemoryGame.js
<<component>> VoiceCalendarMemo.js

Figure 11 - Front End Component Diagram

- **Backend**
  - implemented with **Django**: a high-level Python web framework. Django handles the business logic, server-side processing, and integration with the database. The data for the web application is stored in **SQLite**, a lightweight, serverless database that ensures quick and reliable data retrieval, making it suitable for small to medium-scale applications.
  - **Twilio** Django component has been used to manage the send of a WhatsApp message.

The **mobile application** is built using **Android** as the target platform and **Kotlin** as the primary programming language. The mobile application is optimized for Android devices.

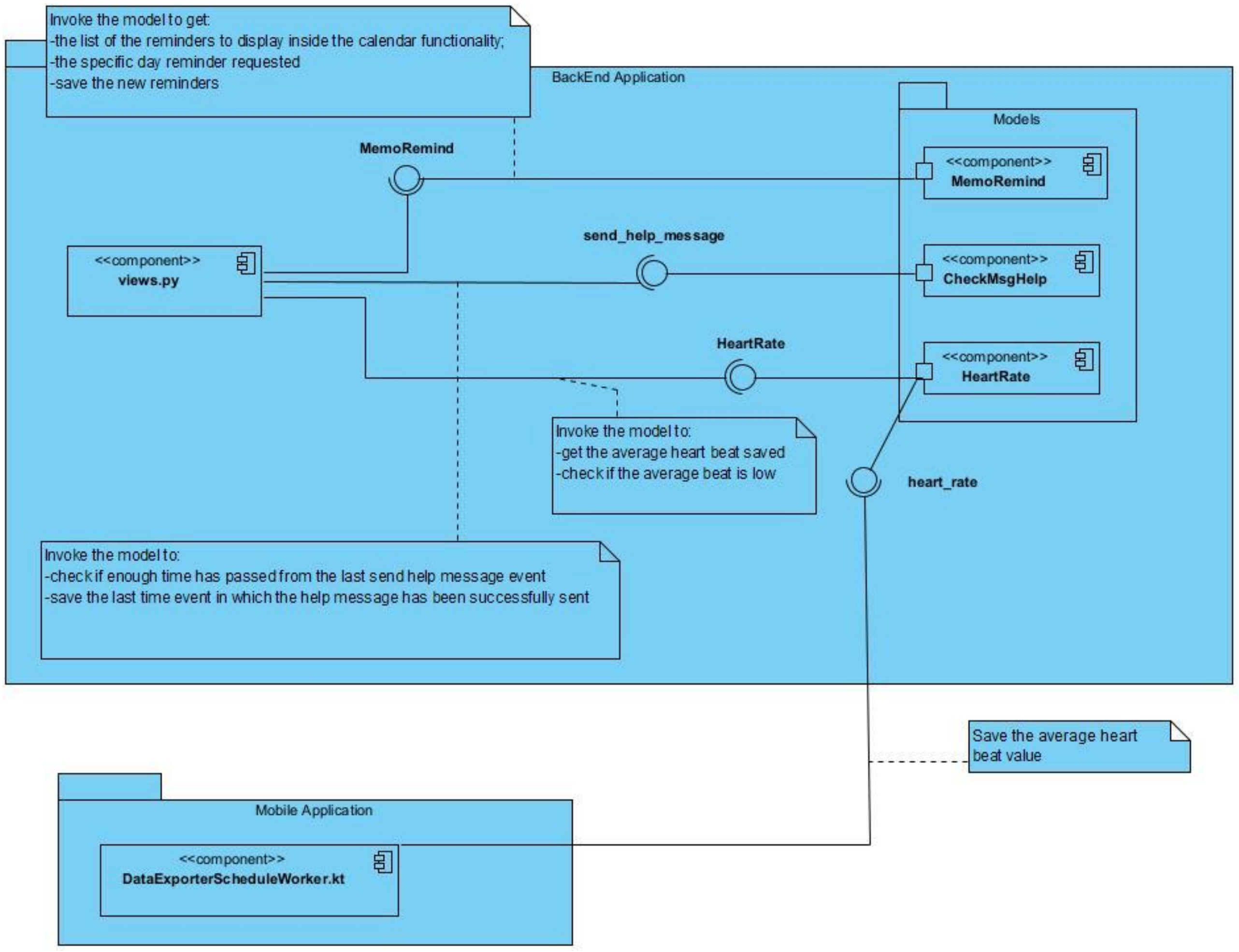

Invoke the model to get:
-the list of the reminders to display inside the calendar functionality;
-the specific day reminder requested
-save the new reminders
BackEnd Application
Models
MemoRemind
<<component>>
MemoRemind
send_help_message
<<component>>
views.py
<<component>>
CheckMsgHelp
HeartRate
<<component>>
HeartRate
Invoke the model to:
-get the average heart beat saved
-check if the average beat is low
heart_rate
Invoke the model to:
-check if enough time has passed from the last send help message event
-save the last time event in which the help message has been successfully sent
Save the average heart beat value
Mobile Application
<<component>>
DataExporterScheduleWorker.kt

Figure 12 - BackEnd and Mobile Component Diagram

The web and mobile applications are designed to work together seamlessly. The backend of the web application provides APIs that allow the mobile application to communicate with it in real-time, ensuring data consistency and reliability across both platforms. This modular and well-structured architecture enables scalability, maintainability, and a consistent user experience across devices.

3.2 Use case

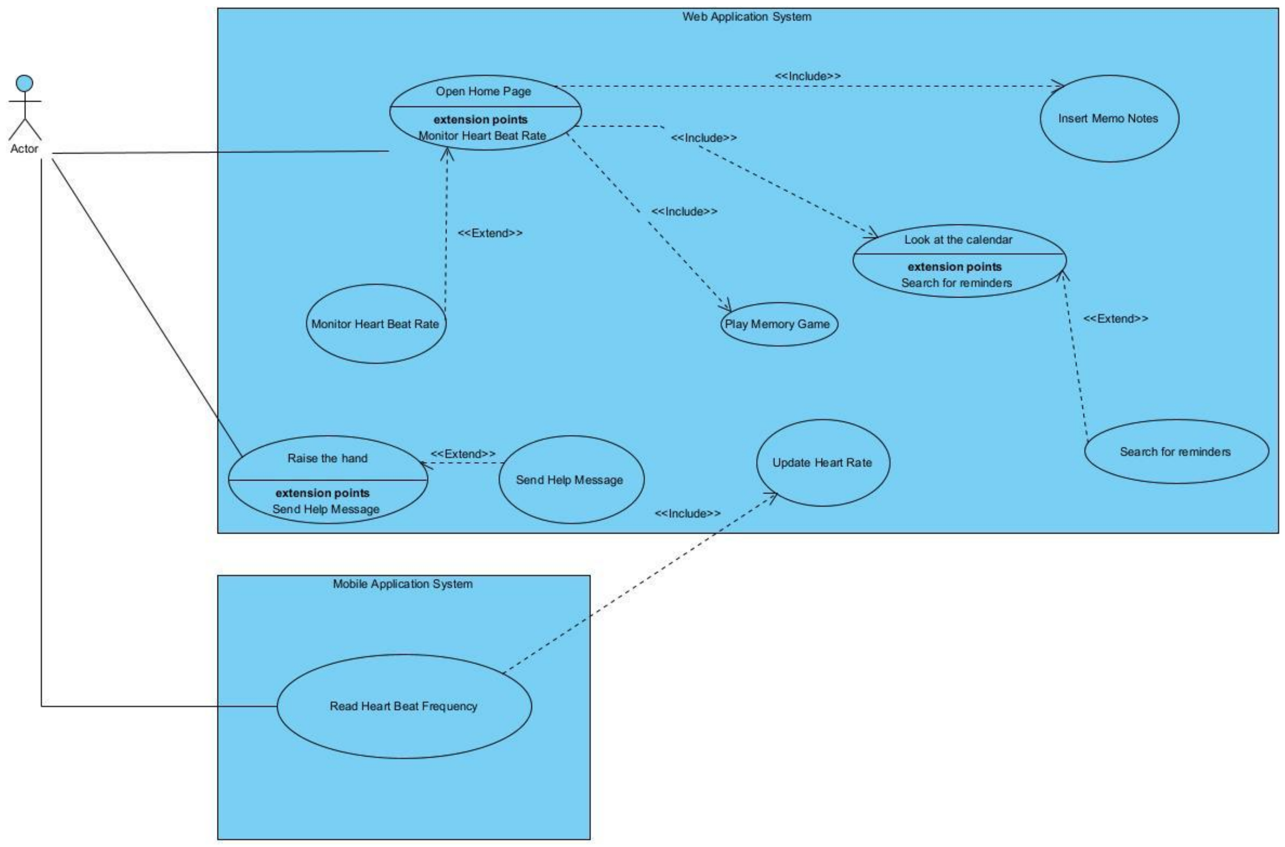


Figure 13- Use case

The use case diagram illustrates the interactions between the user (actor) and a system composed of two integrated components.

The functionalities provided by the **web application system** and by the **mobile application system** are the ones indicated in the preceding paragraphs.

3.3 Gesture Interaction - Send help message

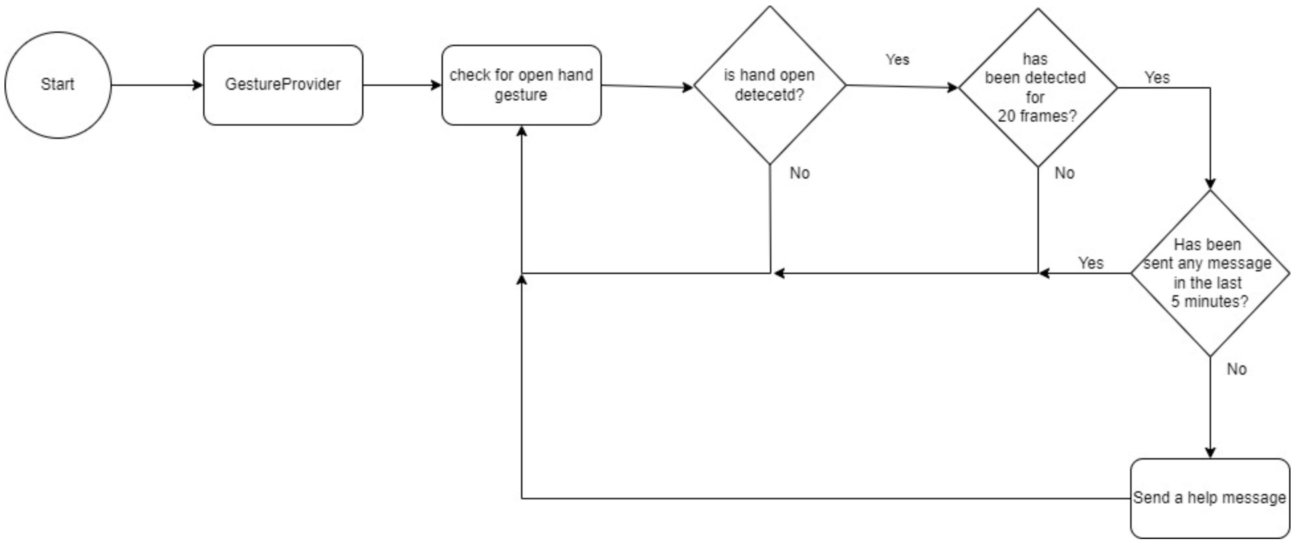

Start
GestureProvider
check for open hand gesture
is hand open detecetd?
Yes
has been detected for 20 frames?
Yes
No
No
Has been sent any message in the last 5 minutes?
Yes
No
Send a help message

Figure 14- Gesture Help Message

The **GestureProvider** component is active throughout the entire lifecycle of the application.

The flowchart (fig. 9) illustrates the functionality of the **GestureProvider** component, focusing on detecting an open-hand gesture and triggering a help message under specific conditions. Below is a detailed description of the key features of the diagram:

- **Initialization and Gesture Monitoring**
    - The process begins with the activation of the **GestureProvider** component, which remains active throughout the application's lifecycle.
    - The component continuously checks for the presence of an open-hand gesture performed by the user.
- **Gesture Detection Logic**
    - **Open-Hand Detection**: The system evaluates whether an open hand is detected. If no gesture is recognized, it loops back to continue monitoring.
    - **Gesture Stability**: If an open hand is detected, the system verifies whether the gesture has been continuously present for **20 frames**, ensuring the gesture is stable and not accidental. If this condition is not met, it resumes monitoring.
- **Message Sending Condition**
    - **Time-Based Check**: Once the gesture is confirmed, the system verifies if a help message has already been sent within the **last 5 minutes**. This ensures that repeated messages are not sent unnecessarily.
    - If no message has been sent within the specified time, the system proceeds to trigger a help message.
- **Help Message Execution**
    - Upon meeting all conditions, the system sends a help message to the appropriate recipient completing the process.
    - A timestamp value is stored in the database to verify the time interval during which it is possible or not to send another help message.

The following sequence diagram shows the situation depicted in the points above.

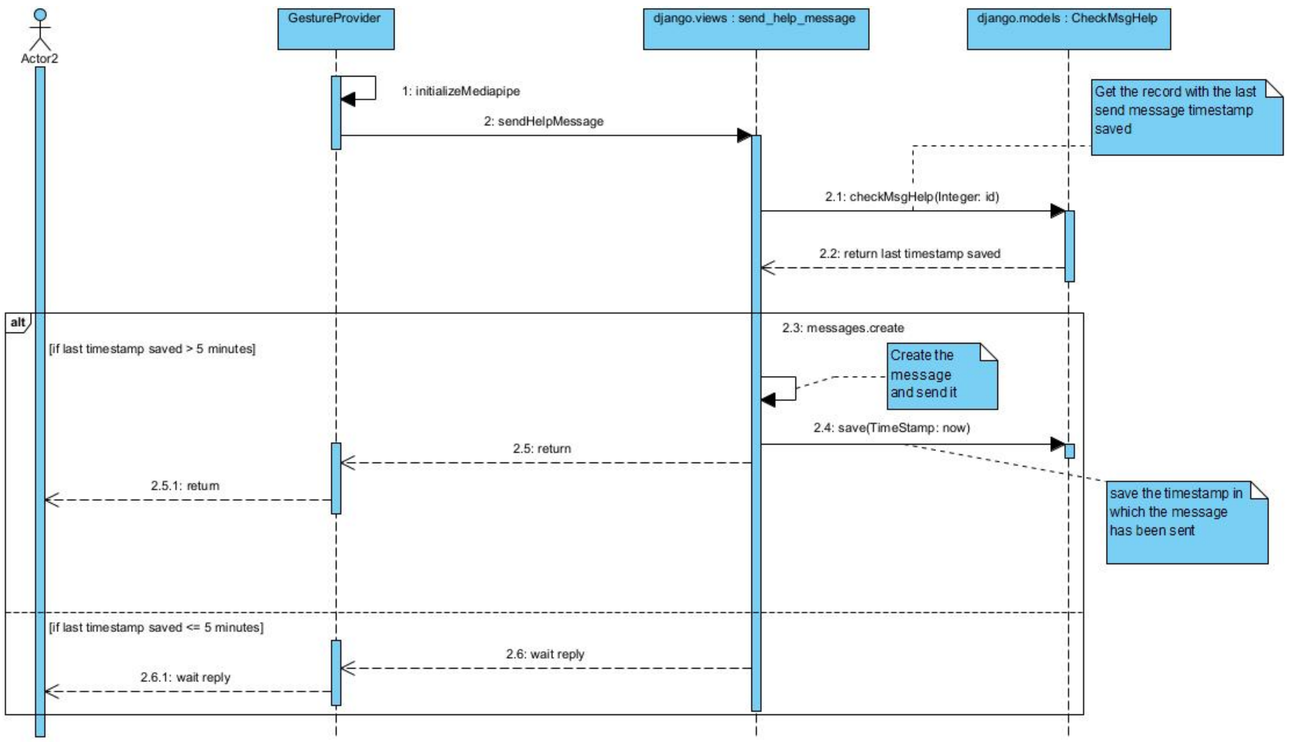

Actor2
GestureProvider
django.views : send_help_message
django.models : CheckMsgHelp
1: initializeMediapipe
2: sendHelpMessage
Get the record with the last send message timestamp saved
2.1: checkMsgHelp(Integer: id)
2.2: return last timestamp saved
alt
[if last timestamp saved > 5 minutes]
2.3: messages.create
Create the message and send it
2.4: save(TimeStamp: now)
2.5: return
2.5.1: return
save the timestamp in which the message has been sent
[if last timestamp saved <= 5 minutes]
2.6: wait reply
2.6.1: wait reply

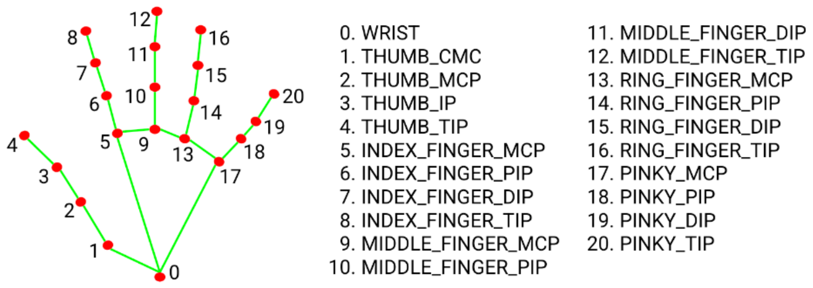
Figure 15- Gesture interaction with save event

3.3.1 MediaPipe

The library used to monitor and check gestures is **MediaPipe.**

I used the hand landmarks detection functionality of the library which uses machine learning to detect and track hands in an image or video stream, identifying 21 distinct hand landmarks for each detected hand. The model works by first identifying the hand's bounding box using a palm detection model and then refining the detected landmarks.

Mediapipe supports gestures such as open hand, closed fist, and pointing fingers by analyzing the spatial relationships of these landmarks.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 16- Mediapipe Hand Landmarks

MediaPipe Hands employs a machine learning pipeline combining multiple models: a palm detection model and a hand landmark model, which operates on the cropped region from the palm detector to predict precise 3D hand keypoints.

The pipeline also leverages temporal coherence, using hand landmarks from previous frames for cropping, and invoking palm detection only when landmarks are no longer detectable.

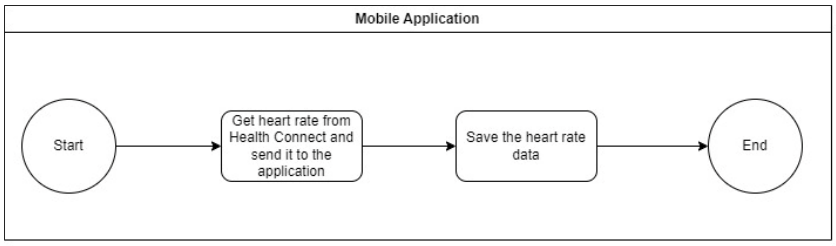
3.3.2 Twilio

Twilio library components have been used to send WhatsApp help messages.

Twilio provides APIs for integrating WhatsApp messaging into applications, enabling to communicate directly with users via the WhatsApp platform. With Twilio's **Programmable Messaging API**, developers can send WhatsApp messages programmatically. A twilio account is needed, then you need to configure a WhatsApp-enabled Twilio phone number. Messages are sent using RESTful API calls, specifying the recipient's WhatsApp number, the sender's Twilio WhatsApp number and the message content.

Twilio supports rich media messaging, allowing the inclusion of text, images, videos, and document attachments, enhancing user interaction. It also provides features to handle delivery receipts and read statuses, ensuring communication tracking. Twilio is a paid service with pricing based on message volume and region; however, it offers a **free trial** with limited features, enabling developers to send WhatsApp messages for testing purposes without incurring costs. This trial includes a communication with verified numbers. This trial version has been used in the application.

3.4 Heart Rate Monitor functionality


Mobile Application
Start
Get heart rate from Health Connect and send it to the application
Save the heart rate data
End

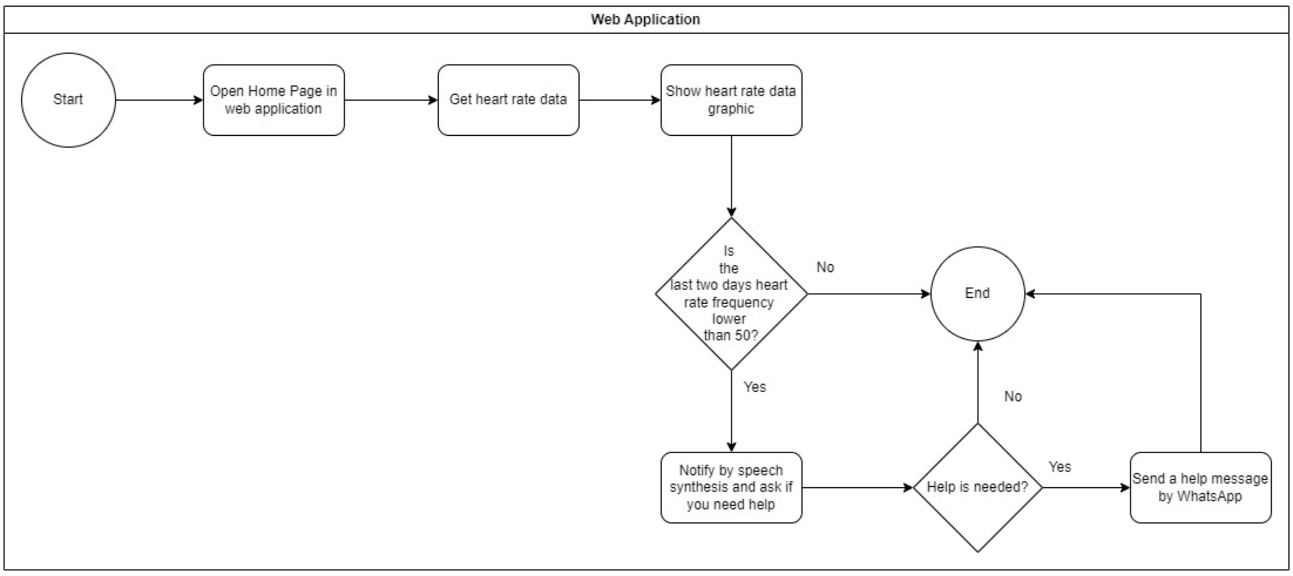

Web Application
Start
Open Home Page in web application
Get heart rate data
Show heart rate data graphic
Is the last two days heart rate frequency lower than 50?
No
End
Yes
No
Notify by speech synthesis and ask if you need help
Help is needed?
Yes
Send a help message by WhatsApp

Figure 17- Heart Rate Monitor

The Heart Rate Monitor functionality is divided into two parts:

- **Mobile Application**:
  - The mobile application communicates with Google Health Connect (Google Fit) which stores health data collected (usually) by a wearable device as a smartwatch
  - Captures the average heart rate and sends it to the backend server to save the data.
  - The heart rate readings can be scheduled to be sent periodically to the backend.
- **Web Application:**

  1. The Home Page retrieves data from the backend.
  2. Displays the data using a line graph.
  3. If the heart rate is below 50 bpm over the past two days
     - a speech synthesis functionality warns the user about it and asks the user whether they would like to request help:
       - If the user responds “non chiamare” or “non chiedere aiuto” or “non richiedere” the voice prompt will stop asking about the need to send a message.
       - If the user responds with "Invia richiesta" ("send request"), "Chiedi aiuto" ("ask for help"), or "Invia messaggio" ("send message"), a message is sent to the designated WhatsApp number.

The following sequence diagram shows the Heart Rate Monitor part of the web application:

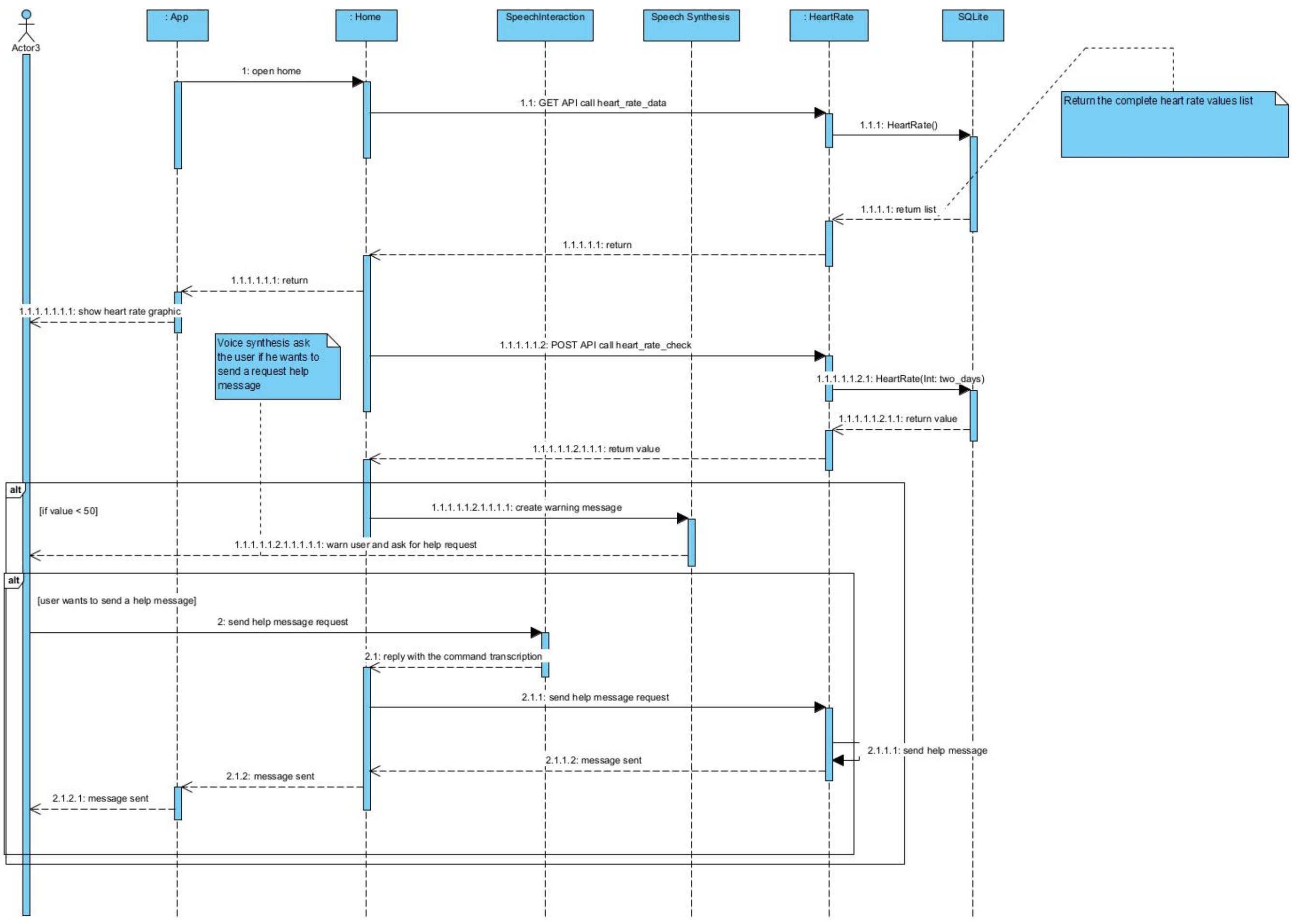

Actor3
: App
: Home
SpeechInteraction
Speech Synthesis
: HeartRate
SQLite
1: open home
1.1: GET API call heart_rate_data
1.1.1: HeartRate()
Return the complete heart rate values list
1.1.1.1: return list
1.1.1.1.1: return
1.1.1.1.1.1: return
1.1.1.1.1.1.1: show heart rate graphic
Voice synthesis ask the user if he wants to send a request help message
1.1.1.1.1.2: POST API call heart_rate_check
1.1.1.1.1.2.1: HeartRate(Int: two_days)
1.1.1.1.1.2.1.1: return value
1.1.1.1.1.2.1.1.1: return value
alt
[if value < 50]
1.1.1.1.1.2.1.1.1.1: create warning message
1.1.1.1.1.2.1.1.1.1.1: warn user and ask for help request
alt
[user wants to send a help message]
2: send help message request
2.1: reply with the command transcription
2.1.1: send help message request
2.1.1.1: send help message
2.1.1.2: message sent
2.1.2: message sent
2.1.2.1: message sent

Figure 18 - Web Application part of Heart Rate Monitor

3.5 Web Application and Voice Interaction

To manage user voice interaction library, React-Speech-Recognition has been used.

React-Speech-Recognition is a lightweight library that provides a React-friendly wrapper around the native **Web Speech API**, simplifying the integration of speech recognition capabilities into React applications. The library enables to capture and process spoken input from users and react to specific phrases or commands. At its core, it leverages the SpeechRecognition interface of the Web Speech API to start, stop, and manage speech recognition sessions. It supports features such as real-time transcription of speech into text, language configuration (e.g., setting the recognition to en-US or it-IT), and continuous listening for prolonged interactions.

The library provides a **useSpeechRecognition hook**, which exposes functionalities such as:

- **transcript**: Real-time access to the text transcribed from the user's speech.
- **listening**: A boolean indicating whether the microphone is actively capturing audio.
- **resetTranscript**: A method to clear the current transcription.
- **Custom Commands**: Developers can define specific phrases or patterns that trigger callback functions when recognized, allowing for dynamic and context-aware voice interactions.

React-Speech-Recognition also handles browser compatibility, detecting whether the user's browser supports the Web Speech API. It supports both continuous and non-continuous listening modes, where continuous mode enables uninterrupted listening for long-form speech input.

To manage speech synthesis, **react-speech-kit** has been used. This is a library for integrating text-to-speech (TTS) capabilities into React applications. It provides an interface to harness browser-native Web Speech APIs. The library includes hooks such as **useSpeechSynthesis** for converting text into spoken words

3.5.1 Memo functionality interaction

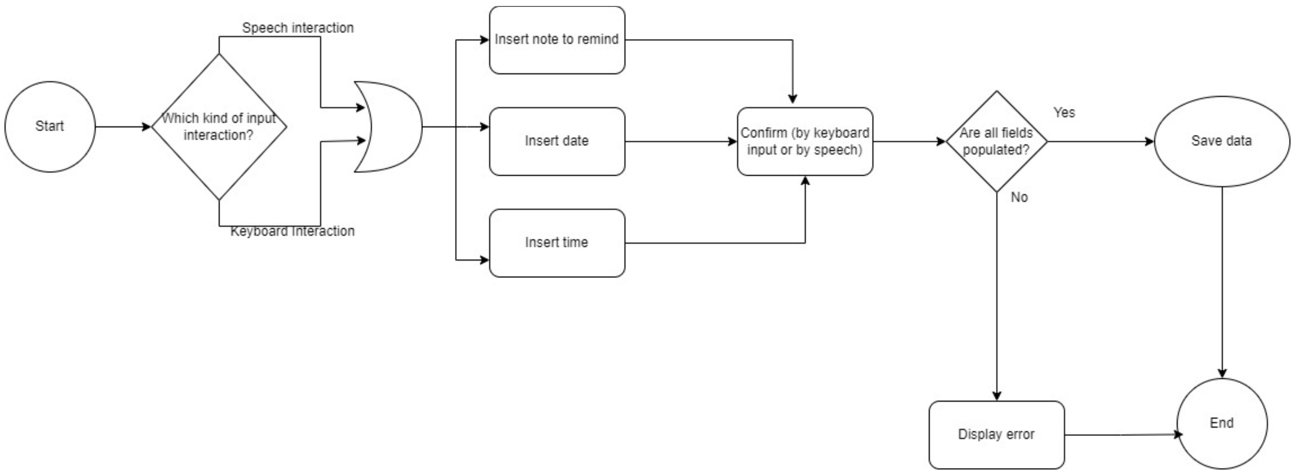


Figure 19- Memo – general representation

The following sequence diagram describes the speech interaction for the memo functionality:

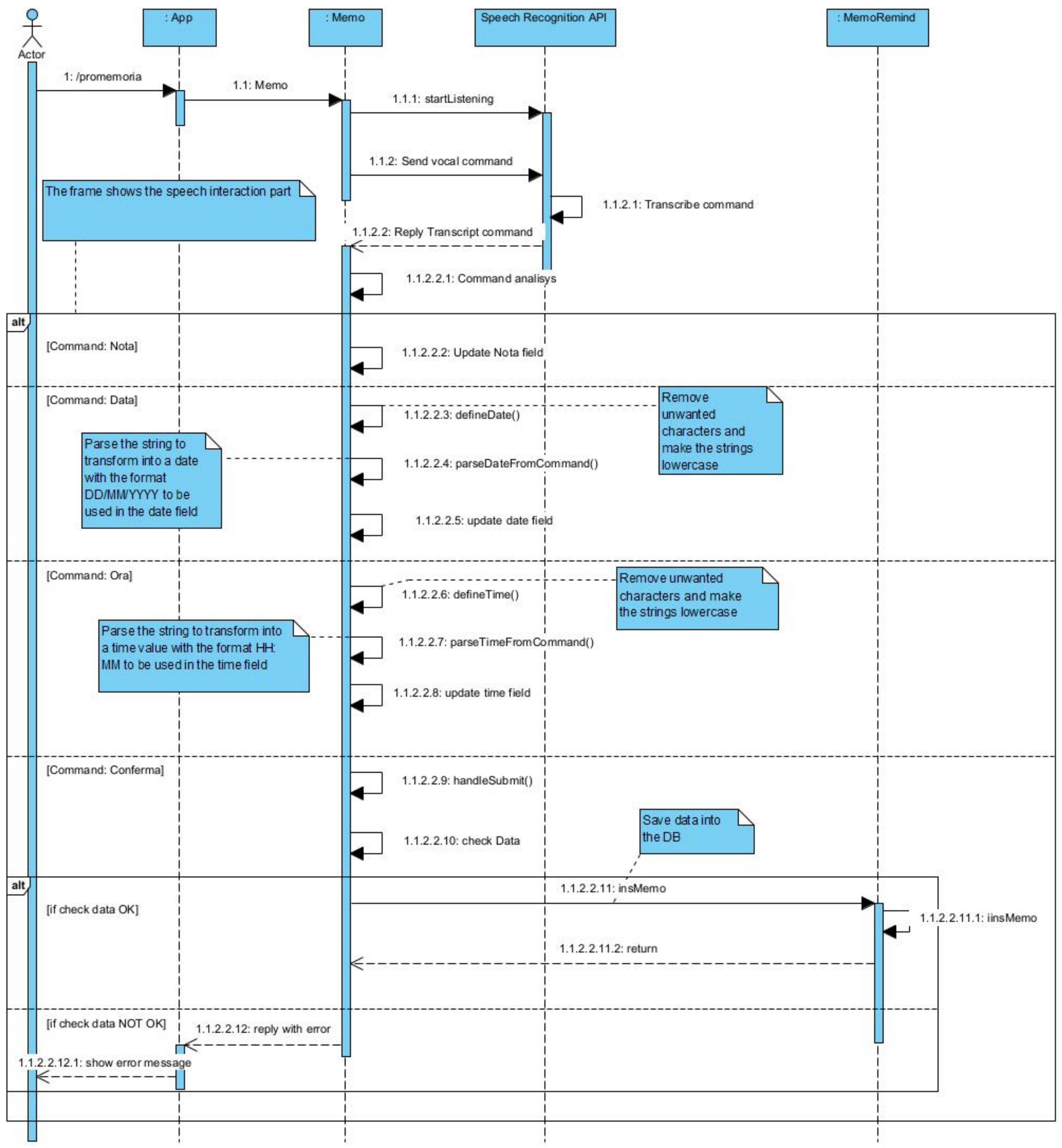

Actor
: App
: Memo
Speech Recognition API
: MemoRemind
1: /promemoria
1.1: Memo
1.1.1: startListening
1.1.2: Send vocal command
1.1.2.1: Transcribe command
1.1.2.2: Reply Transcript command
1.1.2.2.1: Command analisys
The frame shows the speech interaction part
alt
[Command: Nota]
1.1.2.2.2: Update Nota field
[Command: Data]
1.1.2.2.3: defineDate()
Remove unwanted characters and make the strings lowercase
Parse the string to transform into a date with the format DD/MM/YYYY to be used in the date field
1.1.2.2.4: parseDateFromCommand()
1.1.2.2.5: update date field
[Command: Ora]
1.1.2.2.6: defineTime()
Remove unwanted characters and make the strings lowercase
Parse the string to transform into a time value with the format HH: MM to be used in the time field
1.1.2.2.7: parseTimeFromCommand()
1.1.2.2.8: update time field
[Command: Conferma]
1.1.2.2.9: handleSubmit()
Save data into the DB
1.1.2.2.10: check Data
alt
[if check data OK]
1.1.2.2.11: insMemo
1.1.2.2.11.1: iinsMemo
1.1.2.2.11.2: return
[if check data NOT OK]
1.1.2.2.12: reply with error
1.1.2.2.12.1: show error message

Figure 20- Memo voice interaction

The following is a resume of the sequence diagram above.

1. **User Interaction**:
    a. The user initiates the memo functionality by invoking the page at the URL `/promemoria` in the application.
2. **Starting Speech Recognition**:
    a. The `App` module triggers the `Memo` module to start listening to vocal input
    b. The vocal command is sent to the `Speech Recognition API` for processing
3. **Speech Recognition API Response**:
    a. The `Speech Recognition API` transcribes the vocal command into text and sends the result back to the `Memo` module
4. **Command Processing**:

   a. The `Memo` module analyzes the transcribed command identifying specific commands such as "Nota", "Data", "Ora", or "Conferma".

5. **Handling Commands:**
   a. Command “Nota”:
      i. The system identifies that the input relates to a note and updates the `Nota` field
   b. Command “Data”:
      i. The system parses the string to extract a date in `DD/MM/YYYY` format translating the possible text name of the month into the corrispondent month number
      ii. The `parseDateFromCommand()` function processes the string and removes unwanted characters while converting it to lowercase
      iii. The date is then used to update the `date` field
   c. Command “Ora”:
      i. Similarly, the system parses the string for a time value in `HH:MM`
      ii. The `parseTimeFromCommand()` function performs the parsing and cleaning operation of the string
      iii. The extracted time updates the `time` field
6. **Final Submission and Validation**:
   a. Command “Conferma”:
      i. The system processes the `Conferma` command by invoking the method `handleSubmit()` to validate and save the memo data
      ii. The `check Data` operation verifies whether the provided data is correct
7. **Database Interaction**:
   a. If the data is valid, the `Memo` module saves it into the database using the `insMemo` function.
8. **Error Handling**:
   a. If the data is invalid, the system generates an error message
   b. The error is displayed by the user

3.5.2 Calendar functionality interaction

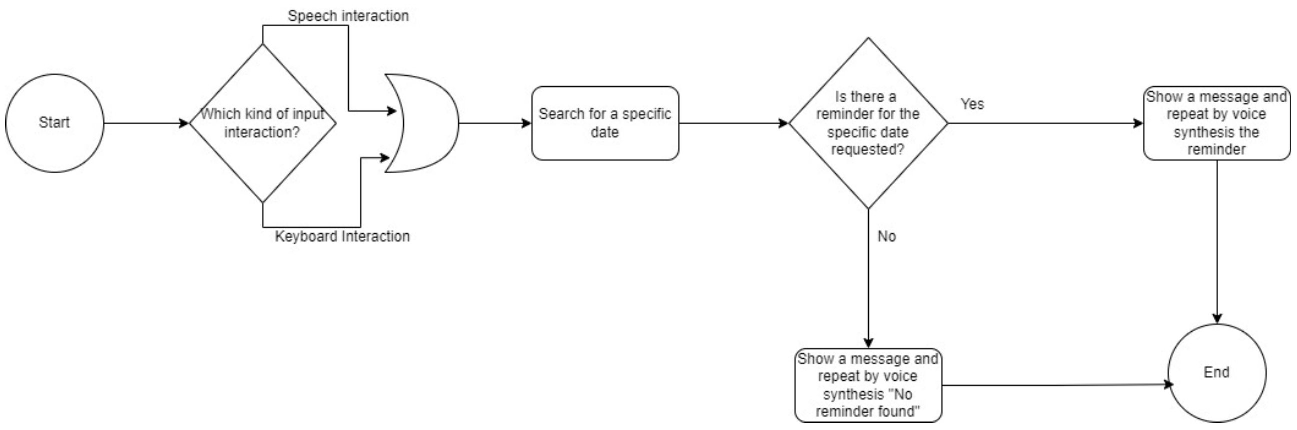

Start
Which kind of input interaction?
Speech interaction
Keyboard Interaction
Search for a specific date
Is there a reminder for the specific date requested?
Yes
No
Show a message and repeat by voice synthesis the reminder
Show a message and repeat by voice synthesis "No reminder found"
End

Figure 21- Search calendar functionality overview

The diagram above illustrates the search functionality for the calendar/reminder functionality:

- The user can interact by voice or by keyboard;
  1. After the users specifies a date can choose to pronounce the word “Conferma” (Confirm) or click on the button to trigger the search;
  2. If there exist a reminder for the date chosen the system communicate it by voice synthesis and by displaying a message on the footer

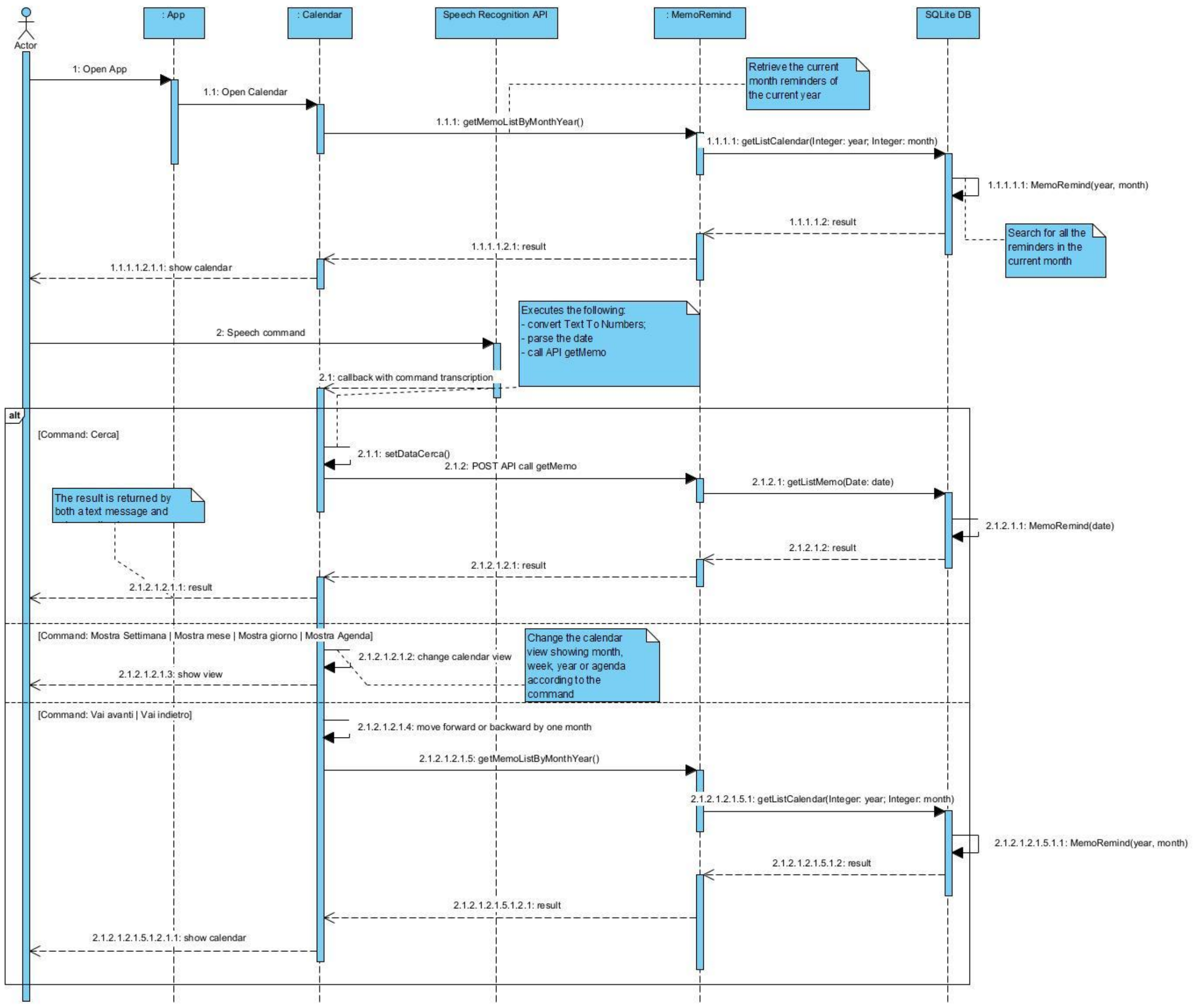
3. If a reminder does not exist for the date chosen, the system communicate it by voice synthesis and by displaying a message on the footer

If the user doesn't hear the vocal memo can ask the system to repeat the last vocal message by using the vocal command: "puoi ripetere?" (Can you repeat?).

The user can interact with the calendar by voice command and by keyboard interaction also to perform the following actions:

- Change the calendar view: month or agenda
- Move backward and forward the current view by one month

The following sequence diagram describes the speech interaction for the calendar/reminder functionality:



Actor
: App
: Calendar
Speech Recognition API
: MemoRemind
SQLite DB
1: Open App
1.1: Open Calendar
Retrieve the current month reminders of the current year
1.1.1: getMemoListByMonthYear()
1.1.1.1: getListCalendar(Integer: year; Integer: month)
1.1.1.1.1: MemoRemind(year, month)
1.1.1.1.2: result
Search for all the reminders in the current month
1.1.1.1.2.1: result
1.1.1.1.2.1.1: show calendar
Executes the following:
- convert Text To Numbers;
- parse the date
- call API getMemo
2: Speech command
2.1: callback with command transcription
alt
[Command: Cerca]
2.1.1: setDataCerca()
2.1.2: POST API call getMemo
2.1.2.1: getListMemo(Date: date)
The result is returned by both a text message and
2.1.2.1.1: MemoRemind(date)
2.1.2.1.2: result
2.1.2.1.2.1: result
2.1.2.1.2.1.1: result
[Command: Mostra Settimana | Mostra mese | Mostra giorno | Mostra Agenda]
2.1.2.1.2.1.2: change calendar view
Change the calendar view showing month, week, year or agenda according to the command
2.1.2.1.2.1.3: show view
[Command: Vai avanti | Vai indietro]
2.1.2.1.2.1.4: move forward or backward by one month
2.1.2.1.2.1.5: getMemoListByMonthYear()
2.1.2.1.2.1.5.1: getListCalendar(Integer: year; Integer: month)
2.1.2.1.2.1.5.1.1: MemoRemind(year, month)
2.1.2.1.2.1.5.1.2: result
2.1.2.1.2.1.5.1.2.1: result
2.1.2.1.2.1.5.1.2.1.1: show calendar

Figure 22 - speech interaction for the calendar/reminder functionality

3.5.3 Memory Game Application

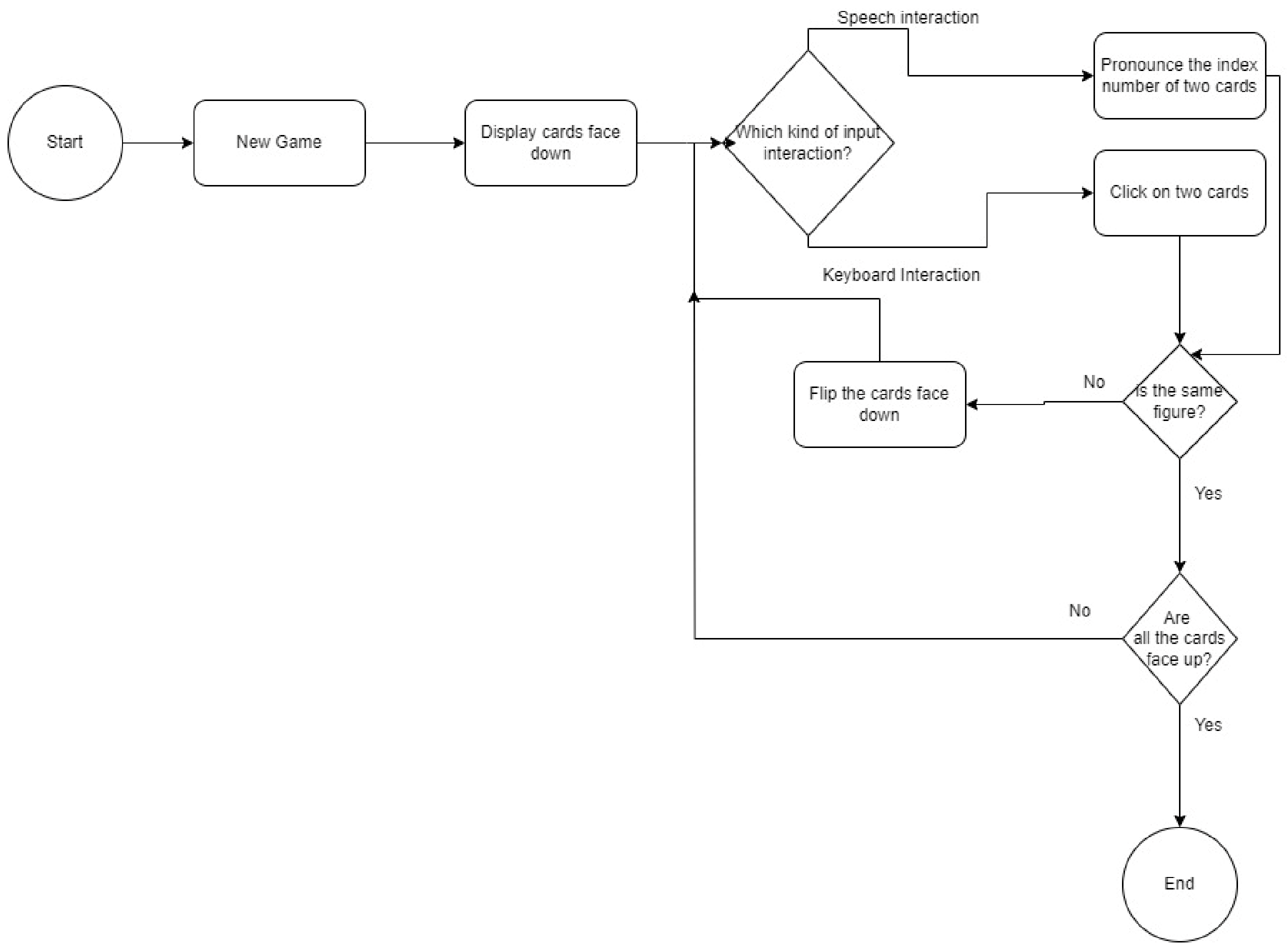


Figure 23 - Memory Game overview

The diagram above illustrates a classical memory game the user can play.

- **Game Initialization:**
  - The game starts with the option to begin a new game.
  - All cards are displayed face down at the start.
- **Input Interaction Selection**:
  - The user selects the type of input interaction:
    - Speech Interaction: The user pronounces the index numbers of two cards.
    - Keyboard/Mouse Interaction: The user clicks on two cards.
- **Card Selection:**
  - Two cards are selected based on the user's chosen interaction method (voice or keyboard/mouse).
- **Card Matching Check:**
  - The system checks if the selected cards have the same figure:
    - If they match, the cards remain face up.
    - If they do not match, the cards are flipped back face down.

- **Game Progression:**
    - The game continues iteratively with the user selecting pairs of cards.
- **Game Completion Check:**
    - The system checks if all cards are face up:
        - If yes, the game ends successfully.
        - If no, the process repeats until all cards are matched.

4 Conclusions

The project demonstrates the potential of integrating voice interaction, gesture recognition, and real-time health monitoring into a single platform designed to enhance the quality of life for older adults. By addressing common challenges such as maintaining organization, monitoring health, and ensuring safety, the application bridges the gap between technology and user accessibility.

Key innovations include:

- **Multimodal Interaction**: The combination of voice commands, keyboard inputs, and gesture recognition ensures usability for diverse user needs and preferences.
- **Health and Safety Features**: The integration of heart rate monitoring and emergency gesture-based assistance enhances user well-being and provides peace of mind.
- **Cognitive Engagement**: The inclusion of a memory game highlights the application's focus on promoting mental sharpness.

The system's modular architecture, utilizing frameworks like React and Django ensures scalability and adaptability for future improvements. However, the project also faces limitations, such as support for only Italian voice.

Future work can focus on

- expanding language support to handle different languages
- enhancing gesture to add a new web page interaction modality
- enhancing health status check by the mean of a model which analyses the voice to get possible suffering situations

References

https://webspeechrecognition.com/

https://developers.google.com/mediapipe/

https://www.twilio.com/

https://www.djangoproject.com/

https://nodejs.org/

https://www.python.org/