POLITECNICO DI BARI

DEPARTMENT OF ELECTRICAL AND INFORMATION ENGINEERING

Master's Degree in Computer Engineering Cybersecurity and Cloud

---

Project Work in Software Architecture and Pattern Design

MEDANGEL - Your Personal Health Assistant

Design and Development Report

Professor:

Prof. Marina Mongiello

Team Members:

Leonardo Coccia

Domenico Notarangelo

Academic Year 2023/2024

SUMMARY

1. Introduction

1.1 Context and Motivation

The decision to develop *MEDANGEL* stems from a confluence of societal needs, technological advancements, and the desire to improve the quality of life for elderly individuals. As the global population ages, the demand for accessible health management tools has intensified. Elderly individuals often face numerous challenges in navigating the complexities of modern healthcare, including managing medications, understanding health conditions, and accessing necessary resources. Traditional support systems, while essential, may not always meet the evolving demands of this demographic.

One significant motivation for creating *MEDANGEL* was the recognition that many elderly people lack familiarity with modern technology, which can exacerbate feelings of isolation and dependency. By designing a user-friendly web application specifically tailored to this audience, we aim to reduce the technological barriers that prevent older adults from accessing vital health information and services. The inclusion of features such as a calendar for medication tracking and a chatbot for medical inquiries addresses these barriers directly, providing a straightforward interface that encourages users to engage actively with their health management.

Moreover, studies indicate that timely medication adherence is crucial for effective health outcomes, particularly for the elderly, who often manage multiple prescriptions. However, forgetting to take medications is a common issue that can lead to serious health complications. *MEDANGEL* directly addresses this challenge by offering reminder notifications and an organized calendar view, empowering users to take charge of their health routines.

Additionally, the integration of a pharmacy locator feature underscores our commitment to ensuring that users have easy access to essential medications, thereby reducing the barriers to healthcare access. By facilitating this connection, *MEDANGEL* aspires to foster a sense of autonomy and confidence in elderly users, ultimately enhancing their overall well-being.

In summary, the development of *MEDANGEL* is driven by a profound understanding of the challenges faced by elderly individuals in managing their health. Through a combination of targeted features and user-centric design, we aim to create a solution that not only meets their needs but also enriches their lives, promoting independence and improved health management in a supportive and accessible manner.

1.2 Project Overview

MEDANGEL is a web-based application designed to assist elderly individuals in managing their health and daily routines. The application focuses on three main features:

- **Medical Chatbot:** Provides quick assistance and answers to common questions, helping users to manage their day-to-day medical needs.

- **Medication Reminders:** Users can schedule reminders to ensure timely medication intake, reducing the risk of missed doses.

- **Calendar View**: Users can visualize their daily, weekly, or monthly medication schedules, making it easier to track upcoming doses.

- **Pharmacy Locator Map:** A built-in map displays nearby pharmacies, allowing users to quickly find and access essential medications.

By integrating these functionalities, *MEDANGEL* aims to simplify health management for the elderly, promoting independence and well-being.

1.3 Objectives

The primary objectives of the *MEDANGEL* project are:

- **Enhance Independence:** Empower elderly users to manage their health effectively, reducing reliance on caregivers for medication management.

- **Improve Health Management:** Facilitate timely medication adherence through reminders and a comprehensive calendar view, thus minimizing health risks.

- **Provide Accessible Information:** Offer accurate medical information through the chatbot, helping users make informed decisions regarding their health.

- **Streamline Access to Pharmacies:** Simplify the process of locating nearby pharmacies, ensuring that users can obtain necessary medications easily.

Through these objectives, *MEDANGEL* not only aims to address immediate health management needs but also seeks to foster a sense of confidence and self-reliance in elderly individuals, ultimately contributing to a healthier and more engaged lifestyle.

2. Market Analysis

The market for health management solutions for the elderly is rapidly expanding, driven by demographic shifts and the increasing prevalence of chronic conditions among older adults. As global populations age, the number of individuals aged 65 and older is projected to reach 1.5 billion by 2050, a stark increase from approximately 703 million in 2019. This demographic shift necessitates innovative solutions that suit specifically to the unique health needs of older individuals.

Currently, the market features a mix of traditional healthcare services and emerging digital solutions, including mobile applications and web platforms aimed at promoting health and well-being among the elderly. However, many existing applications often lack user-friendly interfaces, comprehensive features, or accessibility, which can deter elderly users from engaging with technology. *MEDANGEL* aims to addresses this gap by providing a tailored experience that simplifies medication management and health inquiries. Moreover, many apps fail to integrate essential elements like pharmacy locators or reliable medical chatbots that address specific health concerns.

Furthermore, the trend towards telehealth and digital health solutions accelerated during the COVID-19 pandemic, as many elderly individuals sought alternatives to in-person visits for safety reasons. This has led to increased acceptance of technology among older adults, providing an opportune moment for *MEDANGEL* to penetrate the market. By offering features that enable users to manage their health from home, the app capitalizes on the current shift towards remote healthcare solutions.

Market research indicates that older adults are increasingly seeking tools that enhance their autonomy and provide reliable health information. *MEDANGEL* responds to this demand by ensuring that its features are not only practical but also designed with the user's experience in mind. By prioritizing accessibility and ease of use, the application aims to attract a loyal user base that values the combination of technological support and health management.

3. Project Requirements

This section is divided into functional and non-functional requirements, addressing both the core features of the application and the overarching quality standards it must meet. Functional requirements describe the specific tasks the web app must perform, such as user authentication, medication reminders, and interactive calendar management. Non-functional requirements, based on the URPS+ framework, focus on aspects like usability, reliability, performance, and security, ensuring a seamless and secure user experience.

3.1 Functional Requirements

- **User Authentication:** The app allows users to register and log in with validated credentials. Registration requires fields such as password (minimum 8 characters, one uppercase, one lowercase letter, a number, and a symbol), and checks are in place to ensure data consistency.

- **Interactive Calendar:** Once logged in, users can view an interactive calendar that highlights days with scheduled events. Upon clicking a day, users can see a detailed list of events for that day.

- **Medication Reminders:** For each event scheduled on the calendar, reminders are set at specific times to notify the user. Users can confirm the intake of medication by marking checkboxes next to each event.

- **Chatbot:** The chatbot allows users to send messages in Italian, restarting conversations as needed. It is restricted to answering questions related to medical topics.

- **Pharmacy Locator:** An integrated map helps users locate nearby pharmacies. Clicking on blue pins displays the pharmacy’s address, while the red pin shows the user’s home location.

- **Profile Management:** Users can look at their profile info and update it (username, password, name, surname, gender, date of birth, ZIP Code, address, city) through a form similar to the registration process.

- **Event Management:** Users can create, edit, or delete events in the calendar. The "Create New Event" or “Edit Event” buttons allows users to input event details like name, description, time schedule (up to 3 time slots), start and end dates, frequency, and days of the week via a form.

3.2 Non-Functional Requirements

- **Usability:** The web app is designed to be simple, intuitive, and accessible for elderly users, with an uncluttered layout, large fonts, and clear labels. The goal is to minimize the learning curve and ensure that users can easily navigate through key features like the calendar, reminders, and chatbot thanks to the FAQ section in the home page.

- **Reliability:** The system must function consistently, especially for time-sensitive notifications like medication reminders. High uptime and dependable reminder functionality are crucial for user trust and adherence to medical schedules.

- **Performance:** The application needs to be responsive and efficient. Quick load times and minimal lag when interacting with the calendar, reminders, chatbot, and map are essential to maintain a smooth user experience.

- **Security:** Given the sensitive nature of user data (e.g., personal information and medical schedules), strong security protocols are required. Password complexity and hashing, data validation, and secure handling of profile updates are necessary to protect against breaches and unauthorized access.

- **Localization:** The web app is localized for Italian, ensuring that all features, including the chatbot and interface, fully support the language. This enhances usability and accessibility for the target demographic.

4. Project Planning

4.1 Project Phases

The development of *MEDANGEL* followed a structured approach, divided into several key phases to ensure organized progress and effective collaboration.

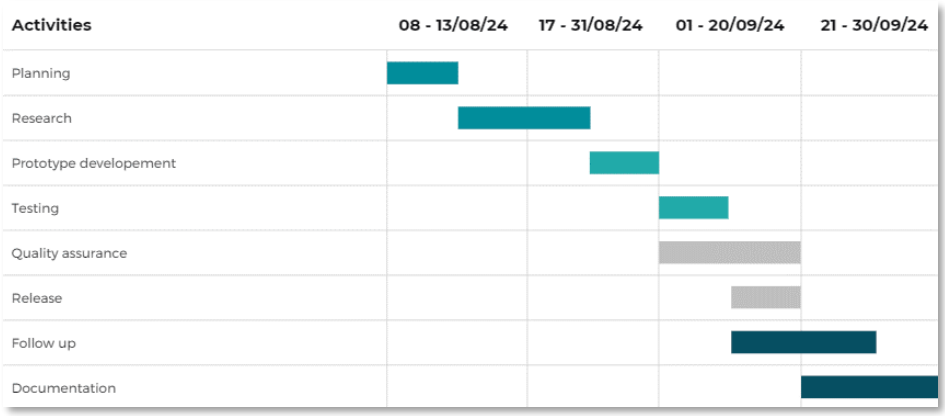
The first phase was Requirement Gathering, where we identified the core functionalities needed to meet the users' needs, such as medication reminders, the interactive calendar, and the chatbot. This stage involved both brainstorming and stakeholder input to ensure that the features would be practical and accessible for elderly users.

Next, we moved into the Design phase. Here, the architecture of the web app was defined, outlining how the frontend and backend components would interact. User experience (UX) design also played a key role, ensuring that the interface would be simple, intuitive, and user-friendly for non-technical users. We mapped out how the calendar, chatbot, and reminders would function within the app, and created wireframes for the core screens.

In the Development phase, the team began coding both the frontend and backend. The backend was implemented using Node.js to handle user authentication, data storage, and event management, while the frontend was developed with React, allowing for dynamic interactions and an interactive user interface. The chatbot integration was handled during this phase as well, ensuring that it could effectively respond to medical-related queries in Italian.

After development, we entered the Testing phase. We conducted both unit testing and functional testing to validate the core features and ensure that the app functioned as expected. This phase also included user acceptance testing (UAT), with feedback from real users who tested the app to identify any usability issues or bugs that needed fixing.

At the end of this section, the **Gantt chart** provides a visual representation of the project timeline, showing the progression through each phase and highlighting key milestones.



4.2 Task Allocation and Time Management

The development of *MEDANGEL* involved a two-person team, allowing for close collaboration and effective communication. With the project starting on August 8, 2024, and concluding on September 30, 2024, we maintained a rigorous schedule, holding Microsoft Teams calls almost every morning, except during the mid-August break. This frequent communication enabled us to work in parallel on various aspects of the web app.

Both team members actively participated in all phases of development, sharing responsibilities equally. For instance, we collaboratively tackled backend development using Node.js, while also focusing on the frontend design with React. GitHub was utilized as our primary collaboration tool, allowing us to manage code changes effectively, track progress, and resolve any issues collaboratively.

Our approach combined agile methodologies with daily updates during our calls, ensuring we remained aligned on tasks and deadlines. This synergy between task allocation and time management facilitated a smooth development process, leading to the successful completion of the project within the designated timeframe.

4.3 Challenges and Solutions

Throughout the development of *MEDANGEL*, we encountered several challenges that required effective problem-solving and adaptability. One of the primary challenges was ensuring that the interactive calendar functioned seamlessly with the reminder system. Integrating these features required extensive testing and debugging to ensure that users received notifications at the correct times.

Another challenge arose during the implementation of the chatbot, particularly in ensuring that it accurately responded to medical-related queries. To address this, we conducted thorough research on the intended topics and continuously tested the chatbot's responses with real user scenarios, refining its capabilities based on feedback.

Additionally, managing our time effectively proved crucial, especially considering our commitment to daily calls and the need for concentrated work periods. We learned to prioritize tasks based on urgency and complexity, which helped us stay on schedule while maintaining the quality of our work.

Overall, these challenges fostered a collaborative environment where we could share ideas and solutions, ultimately enhancing the overall quality of the *MEDANGEL* application.

5. Platform and Technologies

5.1 Overview of Architecture

The architecture of *MEDANGEL* follows a client-server model, where the frontend (client) interacts with the backend (server) through well-defined API endpoints. This separation allows for modular development, where the frontend handles the user interface and interaction, while the backend manages data processing, storage, and retrieval. The backend ensures efficient data handling, user authentication, and event management, while the frontend provides a responsive user experience. The application follows the Model-View-Controller (MVC) pattern on the backend. In this architecture:

- **Model** refers to the database schemas and the management of data, stored in MongoDB.
- **View** represents the frontend elements that the user interacts with, built using React.
- **Controller** handles the logic that processes incoming requests and communicates between the Model and View, implemented using Express.js.

This separation of concerns allows for easier maintenance, testing, and scalability. The choice of this architecture facilitates a clear structure for handling user interactions, data validation, and processing, providing an efficient system for managing medication reminders and user profiles. The client-server model further enables seamless communication between the frontend and backend, ensuring that requests are handled asynchronously and efficiently.

5.1.1 Backend

Node.js

The backend of *MEDANGEL* was built using Node.js due to its non-blocking, event-driven architecture, which is highly efficient for handling multiple simultaneous requests. Node.js is powered by Chrome's V8 JavaScript engine, allowing the use of a single programming language (JavaScript) across both the frontend and backend. This streamlines the development process and is particularly beneficial for managing asynchronous tasks such as reminders, fetching user data, and responding to API calls.

Node.js also integrates seamlessly with databases like MongoDB, which offers scalability and flexibility, ensuring that the backend can grow alongside the user base.

5.1.2 Database

MongoDB

For the database, MongoDB was selected due to its NoSQL structure, which is ideal for managing unstructured data efficiently. The flexibility of MongoDB's document-based storage model suits applications like *MEDANGEL* that require frequent changes to data structures, including user profiles, medication events, and dynamic reminders. Additionally, MongoDB's scalability allows the app to grow without significant changes to the database architecture, ensuring future-proof development.

5.1.3 Frontend

React + Vite & CSS

On the frontend, React was the framework of choice due to its component-based architecture and virtual DOM, which ensures fast rendering of UI elements. React allowed us to create a highly interactive and responsive interface, including features such as the interactive calendar, chatbot interface, and map display. React's reusable components provided a clean and modular approach to frontend development, making it easier to maintain the application.

To enhance the development experience and performance, we also used Vite, a modern frontend build tool. Vite provides a much faster development environment compared to traditional bundlers, thanks to its native ES module support and instant server start-up. It dramatically reduces the time required for building and serving the application during development, which is particularly useful when working on a React-based app. Vite's efficient hot module replacement (HMR) ensures quick feedback on code changes, boosting developer productivity.

For the UI styling, we used CSS, which allowed us to customize the look and feel of the application. Integrating CSS with React enabled us to apply styles specific to each component, maintaining a consistent visual structure and enhancing the user experience.

5.1.4 Development Environment

Visual Studio Code

The development environment for building *MEDANGEL* was Visual Studio Code (VS Code), a highly versatile and lightweight code editor. VS Code was chosen for its rich ecosystem of extensions, which greatly enhanced productivity through tools like live server integration, real-time code linting, and debugging support. Its seamless Git integration allowed the team to manage version control efficiently, and the customizable interface made it easier to tailor the workspace to the specific needs of frontend and backend development. Furthermore, VS Code's powerful search, refactoring, and collaboration features helped streamline the development workflow, ensuring that the team could work efficiently and cohesively.

5.1.5 Framework

Express.js

Express.js was used as the framework for building the Node.js server. It streamlines the creation of API endpoints for handling all backend functionalities, including user authentication, data storage, and event management.

Express.js is lightweight and flexible, making it an ideal choice for building RESTful APIs. This enables smooth communication between the frontend and backend, ensuring that user actions on the web app are processed efficiently.

5.2 API and Endpoints

The following section provides an overview of the available APIs, the respective endpoints, and the type of HTTP requests (GET, POST, PUT, DELETE) that each one accepts. These endpoints enable the application to retrieve, create, update, and delete data, ensuring full CRUD (Create, Read, Update, Delete) functionality across all core features of the web app.

5.2.1 Auth Router

The Auth Router handles all user authentication-related functions, ensuring secure access to the web app. It manages essential operations like user registration, login, and session validation, while implementing safeguards such as input validation and password encryption. In the following list, each endpoint and its accepted HTTP methods are outlined.

- **(GET) '/auth/check':** Verifies if the current user session is valid and authenticated, returning the user's authentication status.
- **(PUT) '/auth/update':** Allows the user to update their profile information, such as their password or personal details.
- **(POST) '/auth/register':** Registers a new user by accepting their details (***e.g.,*** name, email, password), performing necessary validations, and storing them in the database.
- **(POST) '/auth/login':** Authenticates a user by verifying their ***username*** and password, creating a new session upon successful login.
- **(POST) '/auth/logout':** Logs the user out by terminating their active session and clearing authentication data.

5.2.1 Api Router

The API Router handles various functionalities related to event management, reminders, and retrieving pharmacy locations. These endpoints are central to the app's core services,

allowing users to interact with the calendar, set reminders, and access relevant medical information. Below is a list of available endpoints with their HTTP methods:

- **(POST) '/api/calendar/events':** This endpoint allows users to create a new calendar event by providing necessary details such as the medicine name, description, time, frequency, days of the week, and date range (start and end dates). The event is then saved to the user's calendar.

- **(POST) '/api/calendar/getEvents':** Fetches all non-recurring events from the user's calendar. This endpoint returns a list of the events based on the user's ID, including the scheduled medicine reminders and descriptions.

- **(POST) '/api/calendar/getRecurringEvents':** Retrieves all recurring events from the user's calendar. These events are automatically calculated based on the frequency (weekly, bi-weekly, etc.), ensuring that the user sees all upcoming recurring reminders.

- **(GET, PUT, DELETE) '/api/calendar/events/:eventId':**

  - **(GET):** Retrieves a single event by its unique event ID, allowing the user to view details of a specific event.

  - **(PUT):** Updates an existing event by its event ID. The user can modify details such as the medicine name, description, or reminder schedule.

  - **(DELETE):** Deletes a specific event from the user's calendar using the event's ID.

- **(GET) '/api/pharmacy/findNearbyPharmacies':** The user's home coordinates are fetched from their profile, and the endpoint uses these coordinates to calculate a bounding box. It then queries the Nominatim API to search for pharmacies in that area and returns a list of the closest ones, including their addresses. If no pharmacies are found or an error occurs during the search, an appropriate error message is returned.

5.3 Database Schema

For data management, we utilized MongoDB, a NoSQL database that provides the flexibility needed for managing unstructured data. The database schema is structured around two primary collections: **Users** and **Events**. The use of these collections enables efficient querying and management of user profiles and event data, ensuring the app's functionality remains robust as it scales.

5.3.1 Users Collection

This collection contains essential information for user management. Each document includes fields for *username*, *password* (hashed for security), *nome*, *cognome*, *dataNascita*, *sesso*, *citta*, *indirizzo*, *cap*, as well as *residenzaLat* and *residenzaLon* for geographical data. The password is encrypted using bcrypt.js, ensuring both secure storage and protection during transmission. The comparePassword method allows for secure validation of user credentials.

5.3.2 Events Collection

This collection is designed to track medication reminders, using a structure that includes the user's *ID*, *medicine-name*, *description*, and a list of times (specified in hours and minutes) for reminders. It also features fields for *days_of_week*, *week_pattern* (to indicate the frequency of reminders), and *start_date* and *end_date* to specify the event duration. This schema supports dynamic reminders, catering to user preferences effectively.

5.4 Frontend Components

The frontend of the *MEDANGEL* application is composed of various components designed to enhance user experience and facilitate interaction with the application. Each component serves a distinct purpose, ensuring that users can effectively manage their medication schedules, interact with support features, and navigate the application seamlessly. This structured approach allows for a modular and maintainable frontend architecture, ensuring that users can efficiently navigate and utilize the *MEDANGEL* application. Below is a list of the frontend components developed with a briefly description.

5.4.1 BannerEvents

Displays a banner with a message guiding the user on how to view and modify an event. This component enhances usability by offering clear instructions without requiring additional page loads, improving the user experience when managing reminders and events.

5.4.2 CalendarWithEvents

The component renders a calendar view with integrated event data. Users can interact with the calendar to select a date, and events associated with that date are displayed. It allows users to view and manage reminders by clicking on specific dates. The component supports features like marking completed events, making it a core feature of the application for medication management.

5.4.3 ChatBot

Integrates a chatbot powered by Botpress, enabling real-time communication with users. This component acts as a virtual assistant within the app, offering quick responses to user queries.

The chatbot interface is user-friendly, making the app more accessible and offering additional support to users navigating the platform.

5.4.4 Event

Represents an individual event on the app. This component is central to the event management system, displaying event details such as medicine name ***and*** description. It also provides buttons to modify or delete the event, and it includes modals (pop-ups) for confirming deletion and handling errors. The clean layout and interactive features make event management seamless, ensuring that users can easily edit or remove events.

5.4.5 EventForm

This component is used to create or edit an event. It contains input fields for entering the medicine name, description, time, and recurrence details, such as days of the week or start and end dates. Validation mechanisms ensure that all required fields are properly filled before submission. The form interacts with backend services to update or create new event records.

5.4.6 EventList

Displays a scrollable list of all events associated with the current user. Each list item corresponds to a single event, and clicking on an item allows the user to view or edit the event details. The component supports dynamic updates when new events are added or old ones are modified, ensuring that the user always sees the most up-to-date information.

5.4.7 Loading

The component provides a visual indicator to the user when the app is processing data or waiting for a response from the backend. It typically displays a spinner along with a waiting message. This component ensures a smooth user experience during transitions or longer load times, maintaining user engagement and minimizing frustration.

5.4.8 Navbar

Navbar is the top navigation bar of the app, providing easy access to essential sections such as the *home* and *personalizza calendario*. It also includes a dropdown menu for actions like *logout* and *profile*. It serves as the main hub for accessing different app functionalities quickly.

5.4.9 UserForm

Allows users to update their profile information, including personal details like name, email, date of birth and address. This form validates the data before submitting updates to the backend, ensuring data integrity. By providing an intuitive interface, it facilitates easy modification of user profiles, helping users keep their information up-to-date.

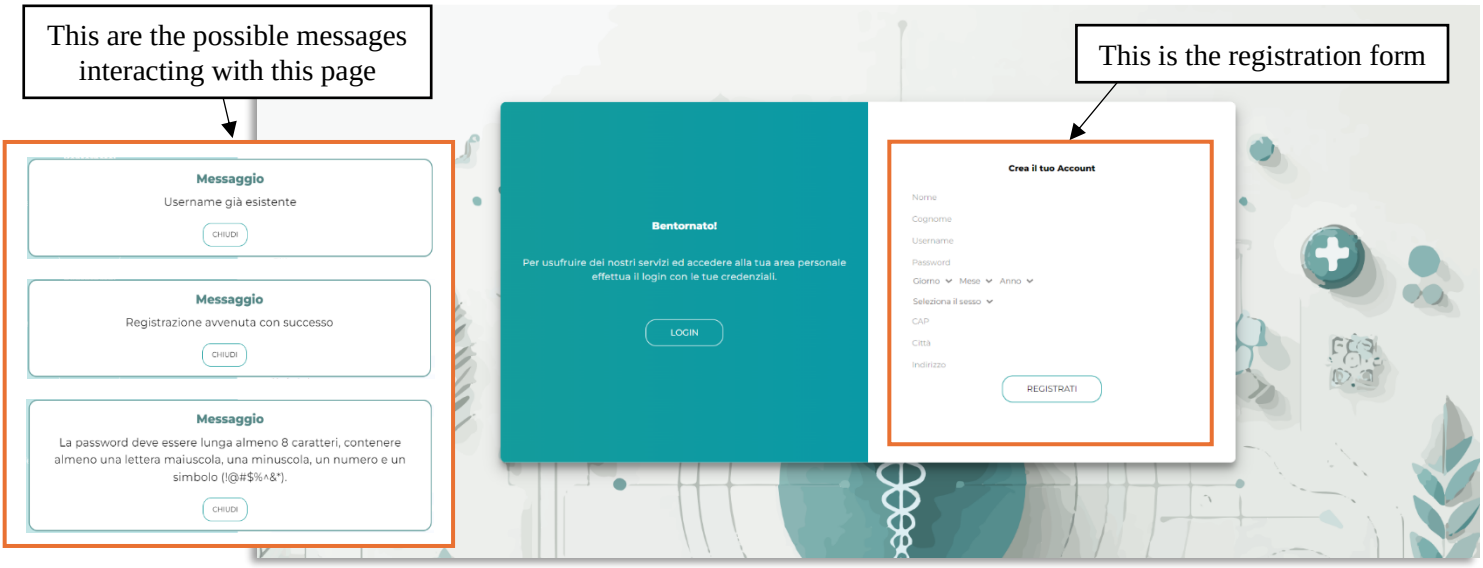
5.4.10 UserInfo

Displays user information, including username, first name and last name, along with a profile icon. This component is displayed in a clean, accessible layout. It allows users to review their data and integrates seamlessly with the app's authentication system to ensure accurate information display. In conjunction with the *UserForm* component, users can easily switch between viewing and editing their information, improving user experience and personalization.

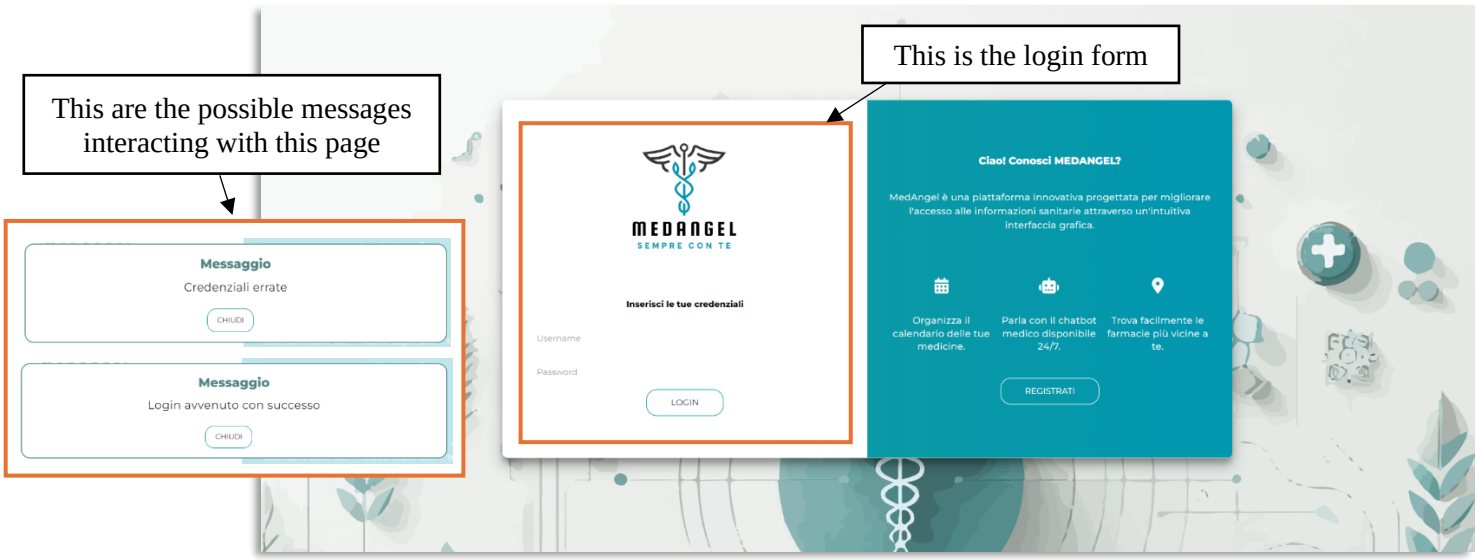
6. Web-App Layers

In this chapter, we will explore the pages that compose our web application. Each section includes a representation of the respective page along with a brief description outlining its functionality and purpose. This visual walkthrough showcases how users interact with the app, manage their profiles, set reminders, and navigate through its core features. The goal is to highlight both the design and the user experience, demonstrating how the app's layout and functions support seamless interaction across various use cases.

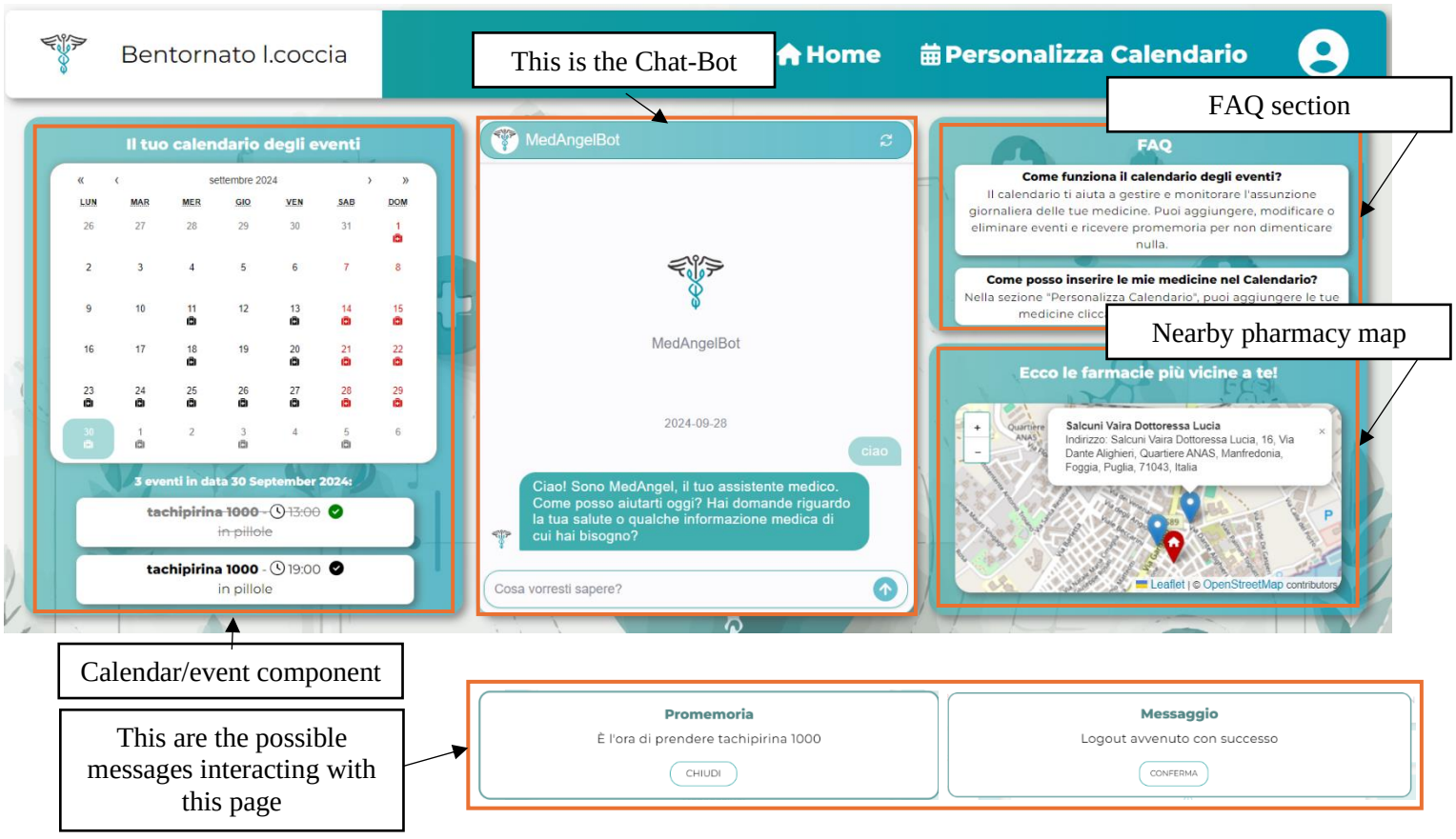
6.1 Registration



6.2 Login

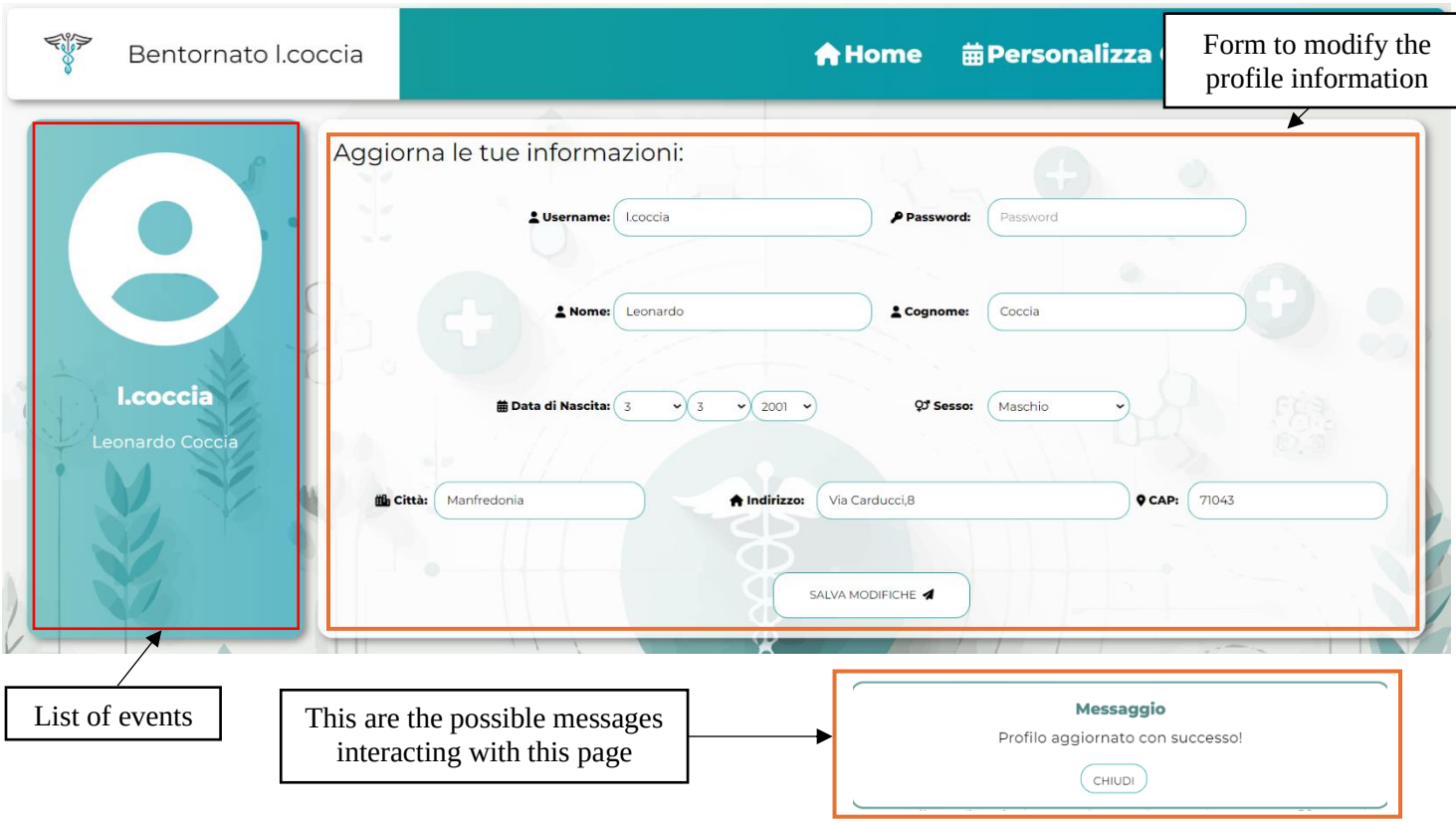


6.3 Home page



6.4 Calendar customization

6.5 Profile


Bentornato l.coccia
Home
Personalizza
Form to modify the profile information
l.coccia
Leonardo Coccia
Aggiorna le tue informazioni:
Username: l.coccia
Password: Password
Nome: Leonardo
Cognome: Coccia
Data di Nascita: 3 3 2001
Sesso: Maschio
Città: Manfredonia
Indirizzo: Via Carducci,8
CAP: 71043
SALVA MODIFICHE
List of events
This are the possible messages interacting with this page
Messaggio
Profilo aggiornato con successo!
CHIUDI

7. Testing and Validation

Testing and validation are essential to ensure that the application meets the required functionality, performance, and reliability standards. In this project, manual testing was employed to verify each feature's behaviour and functionality during development. The primary focus was on validating user interactions and processes across the app, ensuring they align with the project's requirements.

Testing was done incrementally, with each feature being rigorously checked for functionality, usability, and consistency. Each module, including authentication, event management, and profile handling, was tested to ensure correct behaviour under various conditions. This approach allowed for immediate identification and correction of issues, leading to more refined and reliable functionality. Although no automated testing tools were utilized, manual validation effectively ensured the overall stability and performance of the application. The validation process further involved ensuring that user inputs and system responses met predefined requirements, minimizing the risk of errors during normal usage.

7.1 Manual Testing Approach

The testing approach for the project primarily revolved around manual validation to ensure all features functioned as expected. Since automated tools were not used, a hands-on, iterative method was employed. Each component of the web application was examined thoroughly through a series of functional tests.

The process began by setting up specific scenarios reflecting real-world usage, including both typical and edge cases. Each feature—such as user registration, login, event creation, and reminders—was tested step by step. This iterative process allowed for bugs or inconsistencies to be caught early in development.

Testing also involved cross-checking features on multiple devices and browsers to ensure consistency and performance under varying environments. Whenever a bug or issue was detected, the cycle involved debugging, fixing, and retesting until the desired functionality was achieved.

The goal of the manual testing approach was to ensure that every core feature of the application worked seamlessly, and to validate the overall user experience, ensuring a robust and stable product.

7.2 Tested Feature

The following is a comprehensive list of the key features that were manually tested to ensure proper functionality:

1. **User Registration:**
   - Account creation with mandatory fields (username, password, etc.).
   - Validation of form inputs and error handling for invalid entries.
   - Confirmation of successful account creation.
2. **User Login/Logout:**
   - Correct login with valid credentials.
   - Error messages for invalid login attempts.
   - Functionality of the logout process.
3. **Profile Management:**
   - Update user details (name, password, etc.).
   - Validation of updated information.
4. **Event Creation and Modification:**
   - Create new events with proper input validation.
   - Edit and delete existing events.
   - Date, time, and reminder validation.
5. **Calendar Integration:**
   - Accurate display of events on the calendar.
   - Proper functioning of date selection.

6. **Medication Reminders:**
    - Proper setup and scheduling of reminders.
    - Notifications based on event settings.
7. **Pharmacy Search:**
    - Search for nearby pharmacies using home address.
    - Error handling for invalid locations.
8. **ChatBot Integration:**
    - Functionality of chatbot interaction.
    - Response accuracy and relevance.

Each of these features was tested across multiple devices and browsers to ensure stability, consistency, and user experience quality.

7.3 Validation Process

The validation process focused on ensuring the correctness and reliability of the web application's functionalities. Although no formal automated testing tools were utilized, the validation was conducted through:

1. **Cross-Browser Compatibility:** Tested on major browsers (Chrome, Firefox, Edge) to ensure consistent behaviour and layout.
2. **Edge Case Testing:** Inputs such as invalid dates, special characters in forms, and extreme values were tested to verify proper handling and error messaging.

This manual testing and validation ensured that the web application functioned as intended under various scenarios and across multiple environments.

8. Conclusions

This chapter reflects on the development and outcomes of the *MEDANGEL* web application project. The main goals of improving medication adherence through an intuitive reminder system were successfully achieved, leveraging modern technologies like MongoDB, Node.js, and React. Throughout the development process, several challenges were encountered and addressed effectively, leading to a functional, user-friendly platform. This section summarizes key learnings, the app's impact, and identifies potential areas for future enhancements to expand its functionality and usability further.

8.1 Overall Project Evaluation

The *MEDANGEL* project was evaluated through various lenses, considering its functional performance, user experience, and overall impact on medication adherence. The application was developed with a focus on providing users with a streamlined interface for managing their medication schedules, ensuring that critical reminders are accessible and actionable. Some areas for improvement were identified, such as enhancing the mobile accessibility of the platform and adding advanced features for customization. These insights will guide the ongoing development of *MEDANGEL*, ensuring it continues to meet the evolving needs of its users while maintaining a user-friendly experience.

In summary, the project has successfully laid the foundation for future enhancements, with a clear path for growth and improvement based on user needs and technological advancements.

8.2 Future Developments

This section outlines potential future developments for *MEDANGEL*, divided into two key areas: Web-App Improvements and Mobile Implementations.

8.1.1 Web-App Improvements

- **Avatar:** Introducing an avatar-based interface for the chatbot would enhance user interaction by replacing the current text-based communication with a more engaging virtual character. This avatar would use visual and voice interaction to translate the conversation into text, seamlessly communicating with the underlying chatbot. This adds a layer of personality and immersion to the user experience, making interactions feel more natural, especially for those who may prefer a more human-like interface compared to a purely text-based chatbot.

- **Dynamic pharmacy retrieval:** Enhancing the dynamic retrieval of nearby pharmacies would allow users to access updated information about pharmacies based on their current location.

- **Personal storage area:** Implementing a personal storage area would enable users to securely save their medical data, prescriptions, and health-related documents.

- **Third party log-in:** Adding the option for third-party logins, such as Google or Facebook, would streamline the user onboarding process, allowing new users to quickly sign up and log in using their existing accounts. This would enhance user convenience, reduce friction, and increase overall user retention. Additionally, it could provide extra layers of security through multi-factor authentication available via these services.

8.1.2 Mobile Implementations

- **Smartwatch app:** Integrating *MEDANGEL* with a smartwatch app would allow users to receive medication reminders directly on their wrist. This would add convenience, especially for those who are on the go and may not always have their phone nearby. The lightweight nature of smartwatch notifications ensures users are reminded in a non-intrusive way, promoting better adherence to medication schedules.

- **Smartphone & tablet app:** Developing a dedicated smartphone and tablet app would enhance user experience, providing a more optimized and responsive interface compared to a web application. Native apps can leverage device-specific features, such as push notifications and biometric authentication.

- **Telegram integration:** Integrating *MEDANGEL* with Telegram would allow users to receive reminders and updates through a platform they already use frequently. This would offer an alternative communication channel that fits into users' existing habits. Telegram bots could be programmed to send reminders, allow users to mark medications as taken, and even provide real-time support or updates about their health management routine through a simple chat interface. Trough Telegram mini app is possible to make the bot interface even more flexible, so more usable for the user.