



Software Engineering Department
ORT Braude College
Capstone Project Phase B – 61999

MedCall

24-1-D-10

Supervisor: Mr. Alex Keselman

Tareez Ghandour - 211515838 | Email: Tareez.Ghandour@e.braude.ac.il

Leen Hawa - 314655309 | Email: Leen.Hawa@e.braude.ac.il

GIT Link: <https://github.com/tareezgh/MedCall>

Platform Link: <https://medcall-client.web.app/>

Abstract

Our MedCall project is an online platform that aims to transform emergency medical services (EMS) by implementing modern technology and algorithms to speed up response times. It enables real-time communication between people in need of help and medical services. Our system uses the Nearest Neighbor algorithm to quickly locate and track the nearest ambulance. Users can see in real-time the location of the ambulance; this feature ensures that all parties involved can monitor the ambulance's progress and estimated time of arrival. MedCall is designed to be user-friendly and easy to use, making it accessible to all users. Our goal is to significantly improve the speed and reliability of medical interventions in emergencies, helping to save lives and enhance patient outcomes.

Table of Contents

1. Introduction	5
2. Background and Related Work	7
2.1 Related Work.....	7
2.2 Algorithms.....	8
2.3 Technologies Used	9
2.3.1 Node.js.....	9
2.3.2 Preact.....	10
2.3.3 MongoDB.....	10
3. Expected Achievements.....	11
3.1 Outcomes	11
3.2 Unique Features	11
3.3 Criteria for Success	12
4. The Process	13
4.1 Research – Emergency Medical Services (EMS)	13
4.2 Methodology and Development Process:	13
4.3 Challenges and Constraints.....	14
5. Product.....	15
5.1 Requirements.....	15
5.2 Architecture overview.....	16
5.3 Diagrams.....	17
5.3.1 Use case:.....	17
5.3.2 Class diagram:	17
5.3.3 Activity Diagram:	18
6. User Documentation.....	19
6.1 Home Page and Request Ambulance.....	19
6.2 Registration, Login and Reset password	22
6.3 User Dashboard	26
6.4 Driver Dashboard	29
6.5 Dispatch Center Dashboard.....	31
6.6 Responsive Screens	34

7. <i>Maintenance Guide (User Help)</i>	37
8. <i>Verification and Evaluation</i>	39
8.1 Tests Description	39
8.2 Tests Results.....	42
9. <i>Results and Conclusions</i>	43
10. <i>References</i>	44

1. Introduction

Emergency medical services are of great importance in any community, playing a crucial role in keeping public safety and health. They offer a wide range of public services responsible for providing immediate assistance and support during life-threatening situations, like accidents, natural disasters, fires and many more incidents that require a skilled response. First responders often face challenges in reaching individuals quickly, especially in remote areas or during times of heavy traffic. This problem prevents people from getting the proper treatment within the required time.

Nowadays, people must make a phone call to get emergency care, yet the process of calling an ambulance usually involves several crucial stages. It begins with the recognition of a medical emergency, where individuals evaluate the severity of the situation. Once recognizing the need for urgent medical attention, the first stage involves immediately dialing the emergency number, to connect with the dispatch center. During the call, the dispatcher gathers essential information about the nature of the emergency, the location, and the condition of the patient. In the second stage, emergency services are sent to the location, while the dispatcher may provide pre-arrival instructions to the caller, guiding them on basic first aid actions. The third stage involves the arrival of the ambulance at the location where trained paramedics assess the patient and provide help.

The current emergency care scenario has several flaws. Making a phone call may present challenges in recognizing medical emergencies. Communication challenges such as language barriers, unclear descriptions, or the caller's emotional state may set back effective communication. In addition, the entire emergency response system relies on a functional and reliable telecommunications network. Areas with poor connectivity may face challenges in reaching emergency services promptly. Furthermore, answering an emergency phone call and processing information about a critical situation can be highly stressful for the phone operator. This psychological impact may affect their ability to think and communicate clearly.

Our project aims to significantly reduce emergency response times, by offering a user-friendly platform that connects people with the dispatch center in real-time, our platform locates the nearest ambulance to the patient's location, providing crucial assistance choosing the proper ambulance for specific situations. The technology we will use ensures

that both sides benefit from this solution, the dispatch centers gain access to real-time patient location and status information, enabling them to allocate appropriate resources efficiently. Meanwhile, individuals can track the ambulance's location in real-time and receive pre-arrival instructions to help the patient if needed, this will save time and effort for both individuals and the emergency services.

Patients, ambulance drivers, and the public who depend on this information are among the project's stakeholders. Patients receive timely medical attention, while ambulance drivers and paramedics will be better equipped to respond to emergencies swiftly and provide critical care to those in need. By enhancing the speed and effectiveness of emergency response efforts, the solution directly addresses the needs and concerns of these stakeholders.

2. Background and Related Work

2.1 Related Work

Before we dive into our project, let us look at what others have done in emergency medical services (EMS). Learning from existing ideas can help us make our solution better.

- **MyMDA App:** Magen David Adom offers the MyMDA mobile application, which allows individuals to request emergency medical assistance from their smartphones. The MyMDA app streamlines the process of initiating emergency calls and transmitting location data to responders.
- **United Hatzalah:** In Israel, United Hatzalah uses smart technology to send volunteer medics quickly to emergencies. They track medics using GPS and communicate in real-time, making sure help gets to people fast, especially in busy cities.

These examples show how technology and smart thinking can make EMS better. By looking at what others have done, we can get ideas to make our project even more helpful for people in emergencies.

2.2 Algorithms

- **Nearest Neighbor Algorithm [2]**

With life and death situations time is an extremely important factor. The Nearest Neighbor algorithm is essential for the emergency response platform as it plays a crucial role in minimizing response times and optimizing resource allocation. The algorithm ensures swift medical assistance by efficiently finding the nearest available ambulance to an emergency location. The algorithm's simplicity and fast computation further contribute to quick decision-making, crucial in emergency situations where time is crucial.

- **Real-time algorithms [3]**

Real-time algorithms help update decision making based on constantly changing data and circumstances, by analyzing incoming information, like traffic and the availability of medical assistance. These algorithms help in optimizing response strategies in real-time, so emergency teams can navigate through obstacles efficiently.

- **Geolocation and Mapping Algorithm**

Numerous geolocation technologies can pinpoint a person's or an object's position on the Earth [4]. The algorithms are responsible for determining the geographical location of an object or user based on various data sources such as GPS signals. These algorithms help in optimizing arrival and response time, in addition to accurate location tracking.

- **Machine Learning Live Chat Support**

Live chat support has emerged as a powerful solution for bridging the gap between businesses and customers in the digital era. Its ability to provide prompt assistance and personalized interactions that yield higher customer satisfaction rates has turned it into an essential tool for businesses across various industries [5].

To utilize machine learning for live chat support using Natural Language Processing (NLP) techniques to analyze and understand the intent and context behind each inquiry to generate appropriate and timely responses. And to generate appropriate responses to customer queries. The system is trained to recognize various intents and maintain context throughout the conversation, ensuring accurate and context-aware responses. Additionally, the system's capabilities extend to assisting in selecting the appropriate ambulance type based on the emergency.

2.3 Technologies Used

For development, we have decided to use Preact for the frontend, Node.js for the backend, and MongoDB to manage our database.

These technologies offer a robust and flexible foundation for building modern web applications, providing scalability, performance, and ease of development.

2.3.1 Node.js



Node.js is a runtime environment that enables JavaScript to be executed on the server side. It is known for its high performance and fast execution, thanks to its use of Chrome's V8 JavaScript engine. Node.js provides several advantages over other server-side technologies, making it a popular choice among developers.

One of the key features of Node.js is its event-driven, non-blocking I/O model. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time web applications. [1] This architecture allows Node.js to handle multiple connections simultaneously without being slowed down by operations like file reading or database queries.

Another advantage is that developers can use JavaScript on both the client and server sides, simplifying application development. Additionally, npm (Node Package Manager) provides access to a vast ecosystem of reusable libraries and tools, further accelerating the development process.

Node.js is highly compatible with major operating systems, including Windows, macOS, and Linux. This cross-platform compatibility provides developers with the flexibility to create server-side applications that are not limited to a single platform. Applications built with Node.js can run smoothly on different systems, ensuring broader accessibility and deployment options.

In summary, Node.js is favored by developers for its efficient event-driven architecture, cross-platform compatibility, high performance, and suitability for modern web development.

2.3.2 Preact



Preact is a fast, lightweight JavaScript library used for building user interfaces, offering similar functionality to React but in a smaller package. It is especially well-suited for applications where performance and file size are critical.

Preact employs a component-based architecture, allowing developers to break down complex user interfaces into smaller, reusable parts. This modular approach simplifies the development process, enabling faster iteration and reducing the likelihood of errors.

Preact uses a declarative syntax, which enables developers to describe what the UI should look like based on the application's current state. When the state changes, Preact efficiently updates the Document Object Model (DOM), ensuring the UI reflects the new state. This makes the code more predictable and easier to maintain.

Like React, Preact uses a virtual DOM to optimize performance. The virtual DOM is a lightweight copy of the real DOM. When the application state changes, Preact updates the virtual DOM first and then determines the most efficient way to update the actual DOM, minimizing performance overhead.

Preact also follows a unidirectional data flow, where data is passed from parent components to child components. This flow simplifies data management and makes it easier to track changes, particularly in large-scale applications.

In summary, Preact's small footprint, component-based architecture, declarative syntax, virtual DOM, and unidirectional data flow make it an excellent choice for building fast, interactive user interfaces, especially when performance is a concern.

2.3.3 MongoDB



MongoDB is a NoSQL database that provides flexible, scalable, and high-performance solutions for managing data. Its document-oriented structure allows for the storage of data in a flexible, JSON-like format, which is ideal for handling unstructured data.

One of the primary benefits of MongoDB is its horizontal scalability. As data grows, it can be distributed across multiple servers through a process known as sharding. This ensures that large amounts of data can be managed efficiently without degrading the application's performance.

MongoDB offers high performance, particularly with regard to read and write operations. Its support for indexing improves query speed, making it suitable for applications that require fast data access. MongoDB also features a powerful query language, which allows developers to create complex queries and perform data aggregations easily.

Another key advantage is MongoDB's high availability and data recovery capabilities. MongoDB uses replica sets, where data is copied across multiple servers. In the event of a server failure, another server can take over, ensuring that the application remains available and data integrity is maintained.

In conclusion, MongoDB's flexibility, horizontal scalability, high performance, and strong data recovery features make it an excellent choice for modern web applications requiring efficient data management and minimal downtime.

3. Expected Achievements

3.1 Outcomes

The outcome we expect to achieve in this project is to develop a comprehensive platform for emergency call services that efficiently locates the nearest ambulance to the accident location, considering availability and type. Our platform will offer a range of features to make the emergency call more easily and efficiently, including real-time ambulance tracking capabilities.

We aim to significantly reduce emergency response times, ultimately saving lives and improving outcomes for individuals in critical situations.

3.2 Unique Features

- **Finding the Nearest Ambulance:** One of the primary and unique features of our platform is the utilization of the Nearest Neighbor Algorithm to determine the nearest available ambulance to the site of an emergency. This functionality ensures swift response times and optimal allocation of resources during critical situations. One challenge in this field lies in the implementation of the algorithm to ensure its seamless integration into our platform.
- **Geolocation Services:** Our platform relies on Geolocation Services, which require compatible devices equipped with GPS technology to accurately determine the locations of emergencies and ambulance units in real-time. This technology streamlines

coordination between dispatch centers and responders, ensuring efficient response efforts. One challenge is ensuring that all devices used in our system are equipped with GPS functionality for reliable geolocation services.

- **Emergency Response Dashboard:** feature that offers dispatch centers and responders a clear overview of ongoing emergencies, ambulance statuses, and response activities. With real-time updates, a challenge is ensuring the dashboard's simplicity and ease of use.
- **Real-Time Communication:** Our platform provides real-time communication via chat, ensuring immediate interaction and information exchange among dispatch centers, responders, and other stakeholders involved in emergency situations. This feature enables quick collaboration and timely decision-making, essential for effective emergency response.
- **User Authentication:** Our platform includes user authentication and profile management, catering to various user roles such as responders, dispatch centers, and individuals. This feature ensures secure access while allowing users to customize their profiles.

3.3 Criteria for Success

The success criteria for the project revolves around reducing emergency response times and improving the overall efficiency of the emergency medical services system. The primary goal is to offer a user-friendly platform that connects individuals with the dispatch center in real-time, providing immediate assistance during critical situations. Success relies on the platform's ability to significantly decrease response times by locating the nearest ambulance to the patient's location and facilitating effective communication between the dispatch center and individuals in need. The project's effectiveness is measured by its ability to provide timely medical care to patients, enhance the efficiency of ambulance drivers and paramedics in responding swiftly, meeting the needs and concerns of the stakeholders involved.

4. The Process

4.1 Research – Emergency Medical Services (EMS)

In phase A of our project, we evaluated how important urgent medical care is in our lives and decided to help in that matter. We did some research and found the reason that holds back EMS. In phase B of this project, we are building an online platform that will decrease emergency medical services response times. To help achieve our goal and widen our knowledge in EMS, we focused on answering the following questions:

- What are the primary challenges in the current emergency response system?
- How does the process of calling an ambulance work in the current system, and what are the potential drawbacks?
- What are the potential consequences of delays in emergency response times?
- Who are the stakeholders involved in the project, and what are their respective needs and concerns?
- What potential ethical considerations should be considered when developing and deploying an emergency response platform?

To answer these questions and expand our knowledge, we read articles, watched videos and met up to discuss our thoughts and share our ideas and the main points we should focus on developing our website. Some of the conclusions we reached while researching EMS with regards to developing our idea, there exists a need to add a questionnaire after using our website, this will help us validate our website and improve it. Some of the questions may be:

- Was it easy or difficult to use the website?
- How would you rate the efficiency of the website?
- How satisfied are you with the website and its services?
- What additional features would you like to see in the future?

If we find a connection between the questionnaire and our final analysis, it will confirm the validity of our emergency medical services website.

4.2 Methodology and Development Process:

In crafting our project, we have adopted an agile methodology, recognizing its suitability to the dynamic needs of our MedCall platform. Agile methodology enables us to break down the project into manageable increments, known as sprints, each focusing on delivering specific features or functionalities. By prioritizing user stories and features based on their importance

and value, we ensure that the development process remains responsive to evolving requirements.

Our development process involves two main phases. We start with planning and preparation, where we define project objectives, identify user stories and requirements, develop diagrams, and create a roadmap for development. Then, we move to the implementation stage, where development activities take place. During this stage, we follow an iterative development approach, continuously integrating feedback from stakeholders and conducting regular testing to identify and address issues early in the development process.

4.3 Challenges and Constraints

One of the significant challenges we anticipate in the project's life cycle involves acquiring the requisite development skills essential for building the website and implementing complex algorithms to locate the nearest ambulances and determine their availability. Developing a comprehensive emergency medical services platform requires proficiency in diverse programming languages and algorithms, which may currently be unfamiliar to us.

To overcome this challenge, we plan to invest time and resources in continuous learning and skill enhancement. This will entail participating in courses and leveraging online resources to deepen our understanding and proficiency in the relevant programming languages and algorithms. By dedicating ourselves to ongoing education and professional development, we aim to overcome these obstacles and deliver a high-quality emergency medical services platform that meets the needs of our users and stakeholders.

5. Product

5.1 Requirements:

Functional:

Number	Requirements
1	The system allows user registration via email and password
2	The system allows user registration via Google authentication
3	The system allows users to reset forgotten passwords
4	The system allows grant appropriate permissions based on user roles
5	The system allows user to initiate emergency calls
6	The system allows prompt users to provide necessary information
7	The system allows notifying dispatch centers about the ambulance's request
8	The system allows dispatch centers to select appropriate ambulance type
9	The system allows notifying the nearest available ambulance drivers of incoming emergency calls
10	The system allows displaying essential information about assigned emergency
11	The system allows ambulance drivers to accept or decline emergency calls
12	The system allows communication via chat between individuals and drivers
13	The system allows providing real-time updates to users
14	The system allows confirming ambulance arrival and completion of the request

Non-functional:

Number	Requirements
1	The platform should be user friendly and easy to use
2	The platform should be handling large amounts of traffic at the same time
3	The platform must keep user data secure and protected
4	The platform should support multiple languages
5	The registration via unique email
6	The user roles are dispatch center, ambulance drivers and individuals
7	The information for each emergency is the patients' status and location
8	The ambulance types are regular and intensive care (ICU)

5.2 Architecture overview

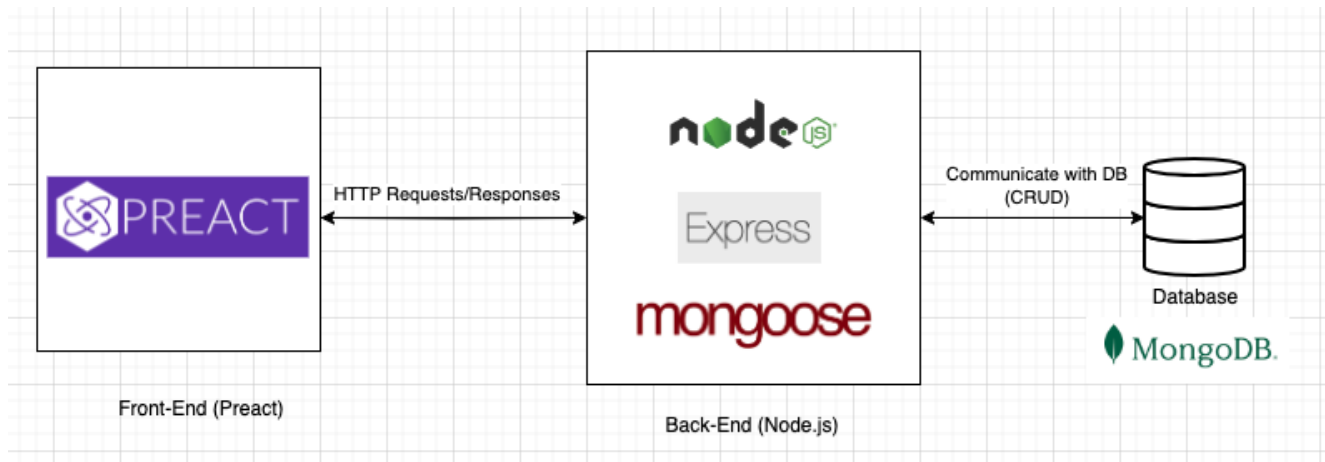


Figure 1- Architecture scheme.

The technologies: MongoDB, Express, Preact and Node.js. Each of these technologies plays a specific role in developing a full-stack application, enabling developers to create dynamic and responsive web experiences.

MongoDB is a NoSQL database that stores data in a flexible and scalable matter, making it suitable for handling large amounts of data. It allows developers to easily store, search and retrieve data, which is essential for building modern web applications.

Express.js is a backend web application framework for Node.js. It offers a robust set of features designed to simplify the development of web servers and APIs.

Preact is a popular frontend library for building user interfaces. It enables developers to create reusable UI components and efficiently updates the UI by rendering only the components that have changed, ensuring a fast and responsive user experience.

Node.js is an open-source, cross-platform runtime environment that allows JavaScript to be used for server-side scripting. Built on Chrome's V8 JavaScript engine, it is known for its scalability and high performance. Node.js enables developers to use JavaScript for both client-side and server-side code, simplifying the development process.

Overall, it's a powerful combination of technologies that allows developers to build web applications using JavaScript. It includes MongoDB as the database, Express for the backend framework, Preact for the frontend, and Node.js as the runtime environment. Together, these technologies enable the creation of full-stack web applications.

5.3 Diagrams

5.3.1 Use case:

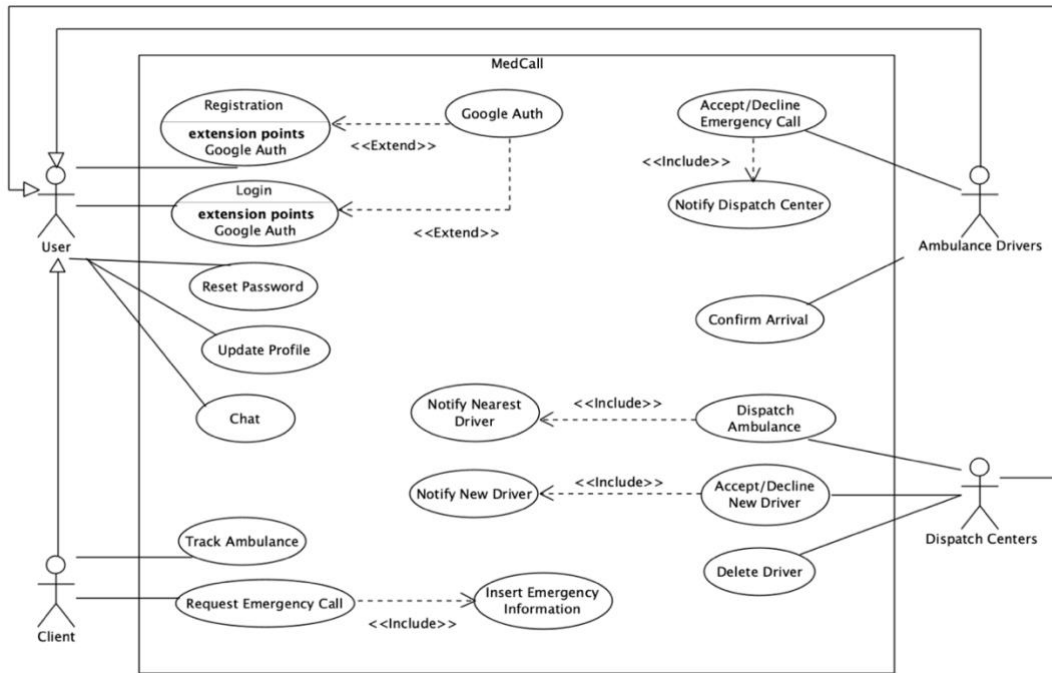


Figure 2 - Use case diagram

5.3.2 Class diagram:

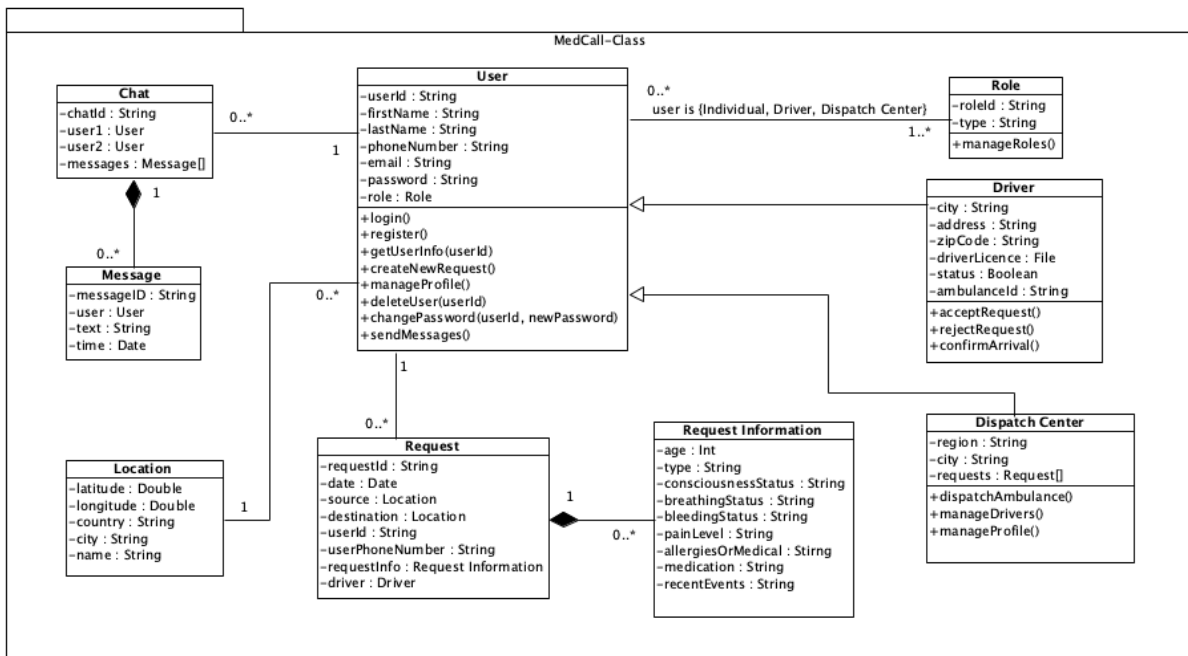


Figure 3 - Class Diagram

5.3.3 Activity Diagram:

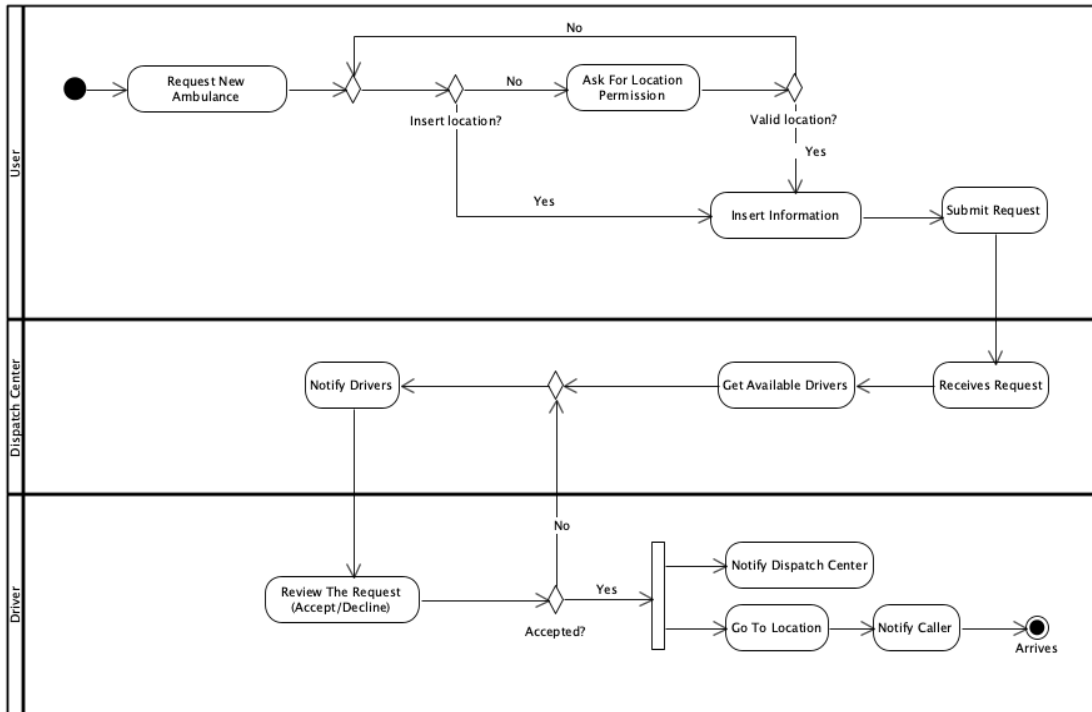


Figure 4 - Activity Diagram

Sequence:

- The user initiates a request for a new ambulance.
- The system checks if user has inserted a location.
- If the location was not inserted, the system asks location permission from user.
- If location permission is granted, the system checks if the location is valid.
- The user inserts information.
- The user submits the ambulance request.
- The dispatch center receives the ambulance request.
- The dispatch center gets available drivers.
- The dispatch notifies available drivers about the request.
- The ambulance driver reviews the request and can either accept or decline.
- If the driver declines the request, the system notifies the dispatch center to find another available driver.
- If the driver accepts the request, he will go to the requested location and the system will notify the dispatch center.
- The system will notify the user that the ambulance is on its way.
- The ambulance arrives at the requested location.

6. User Documentation

6.1 Home Page and Request Ambulance

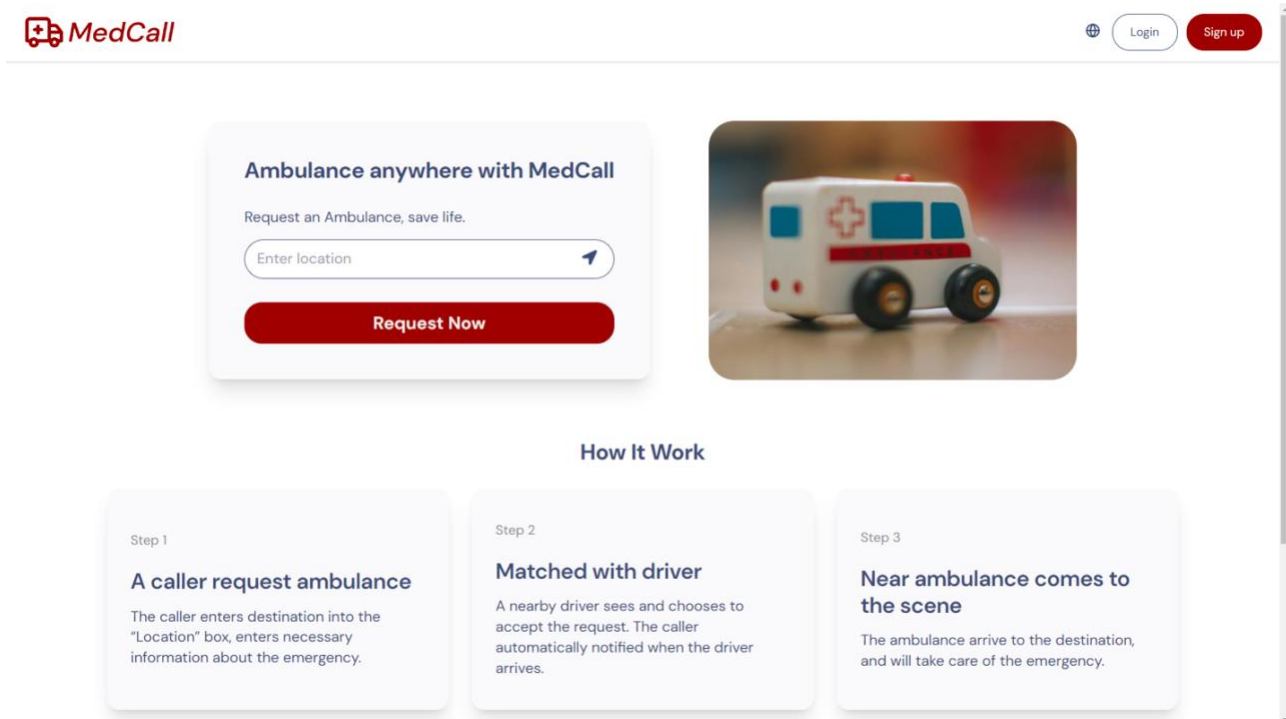


Figure 5 - Home Page

The main screen of MedCall, the first page users encounter, features a clean and intuitive layout designed to facilitate quick access to emergency services. At the top, users can log in or sign up, each leading to dedicated screens for account management. A prominent call-to-action (CTA) button allows users to swiftly request an ambulance, guiding them to the next steps. Below the CTA, an informative section outlines how MedCall operates, helping users understand the process and what to expect when using the platform.

Necessary Information

Please provide the following information to request emergency medical assistance

Caller's Contact Information:

John 0527063291

Patient's Age 21

Type of emergency (Select one)

Accident Injury Fire Difficulty Breathing Severe Bleeding Loss of Consciousness Allergic Reaction Heart Attack Cardiac Arrest Seizure Overdose or Poisoning Other

Patient's Consciousness (Select one)

Conscious Unconscious Unsure

Breathing Status (Select one)

Breathing Normally Difficulty Breathing Not Breathing Unsure

Bleeding (Select one)

Not Bleeding Minor Bleeding Severe Bleeding Unsure

Pain Level (Select one)

Mild Moderate Severe Unsure

Continue

Figure 6 - Request Ambulance Necessary Information

After clicking the call-to-action (CTA) button, users are directed to the "Request Ambulance" screen, where they can input essential information about the emergency. This screen is designed for simplicity and efficiency, allowing users to provide details through predefined options. Users are guided through a series of questions, such as the type of emergency, the patient's condition, and other critical factors. Each question offers selectable answers, ensuring that users can quickly and accurately convey the necessary information to dispatch centers.

Optional Information

Please provide the following information if you know the patient to assist medical emergency.

Allergies or Medical Conditions

Medications

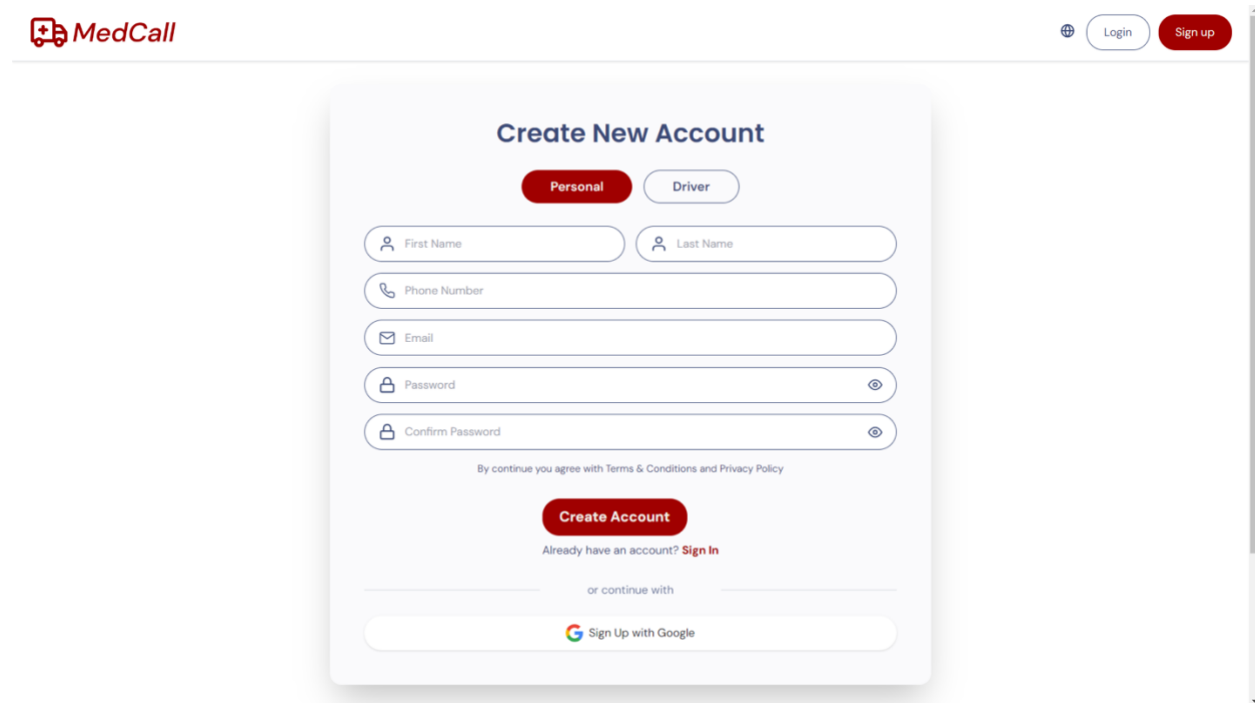
Recent Events or Activities

[Back](#)
[Request Ambulance](#)

Figure 7 - Request Ambulance Optional Information

After answering the predefined questions, users are directed to an optional input screen where they can provide additional details that were not covered earlier. Users can type their responses directly into fields such as allergies, medications, or recent events. Once they submit this information, they are taken to their dashboard. If the user is not logged in, they will be directed to the guest dashboard.

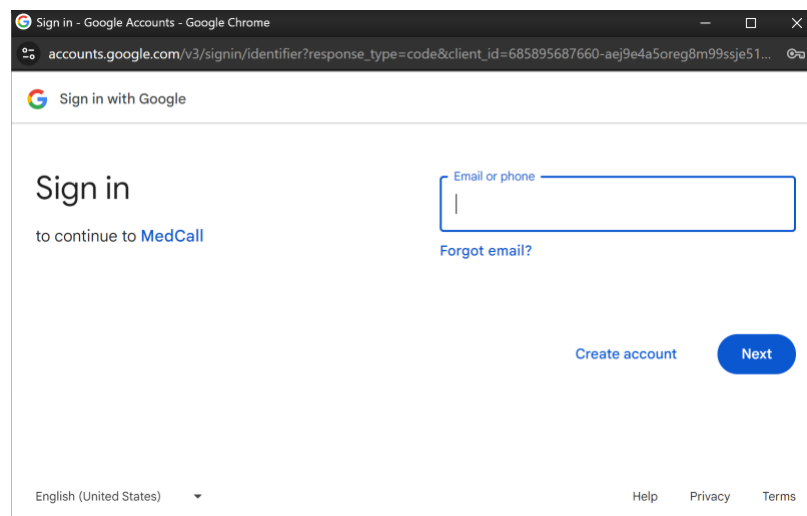
6.2 Registration, Login and Reset password



The image shows a web browser window with the MedCall logo in the top left corner. In the top right corner, there are links for 'Login' and 'Sign up'. The main content is a 'Create New Account' form. At the top of the form, there are two tabs: 'Personal' (selected) and 'Driver'. Below the tabs, there are input fields for 'First Name', 'Last Name', 'Phone Number', 'Email', 'Password', and 'Confirm Password'. Each field has a corresponding icon (person, phone, envelope, and padlock). Below the 'Confirm Password' field, there is a small text line: 'By continue you agree with Terms & Conditions and Privacy Policy'. Below this, there is a red 'Create Account' button. Underneath the button, it says 'Already have an account? Sign In'. At the bottom, there is a section 'or continue with' followed by a 'Sign Up with Google' button with the Google logo.


Figure 8 - Sign up


The signup screen enables users to create a new account. Users can select between creating a personal or driver account by clicking on the respective tabs at the top. For a personal account, users are required to provide basic information such as their name, phone number, email, and password. Additionally, there is an option to sign up using a Google account for quicker access.




The image shows a Google Chrome browser window with the address bar displaying 'accounts.google.com/v3/signin/identifier?response_type=code&client_id=685895687660-aej9e4a5oreg8m99sje51...'. The page title is 'Sign in - Google Accounts - Google Chrome'. The main content is a 'Sign in' form. At the top, there is a 'Sign in with Google' button with the Google logo. Below this, there is a 'Sign in' heading followed by 'to continue to MedCall'. To the right of the heading is a large input field labeled 'Email or phone'. Below the input field is a link 'Forgot email?'. At the bottom right, there are two buttons: 'Create account' and 'Next'. At the bottom left, there is a language selector 'English (United States)' with a dropdown arrow. At the bottom right, there are links for 'Help', 'Privacy', and 'Terms'.


Figure 9 - Google sign up




 [Login](#) [Sign up](#)

One Last Step


 City

 Address

 Zip Code

[Back](#) [Create Account](#)

Already have an account? [Sign In](#)



copyright © 2024




 support@medcall.com


Figure 10 - Sign up Driver More Info



When a user chooses to sign up as a driver, a new screen appears asking for additional information, such as the driver's city, address, and zip code. Once completed, they can create their account by clicking the "Create Account" button.



 [Login](#) [Sign up](#)

Sign in MedCall

 Email


 Password 


☐ Remember me [Forget Password?](#)

[Sign in](#)

Don't have an account? [Sign Up](#)

or continue with

 Sign in with Google



copyright © 2024

[About MedCall](#)



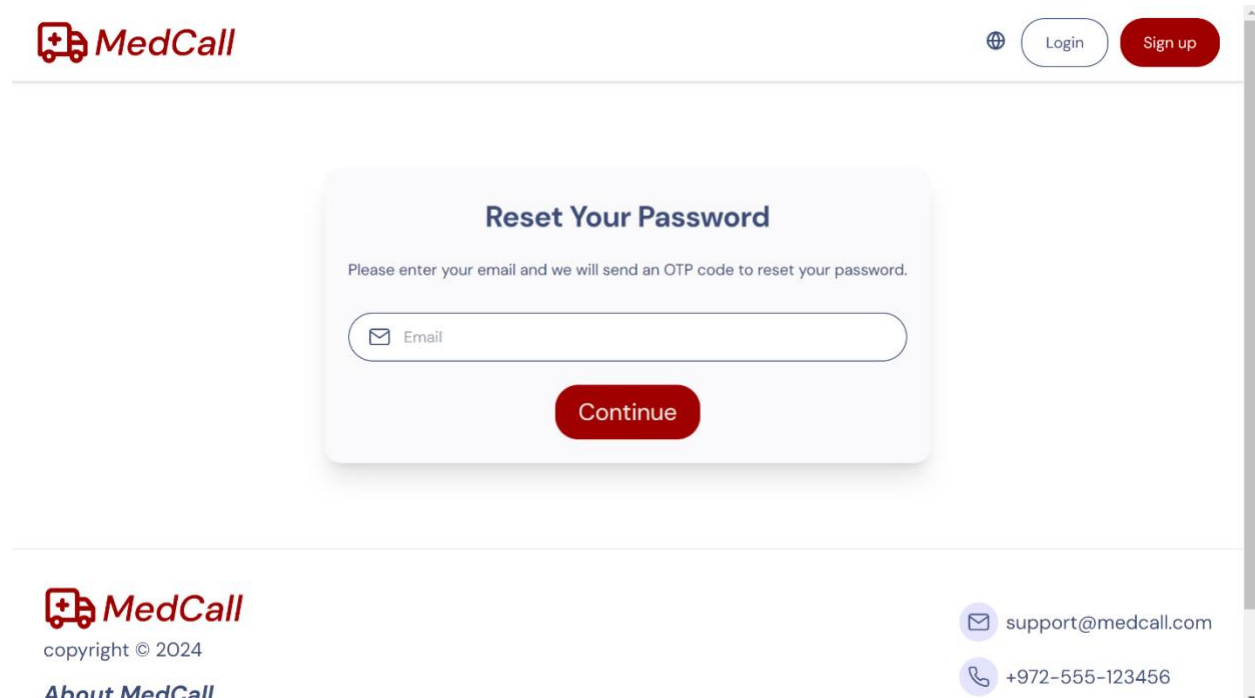
 support@medcall.com
 +972-555-123456

Figure 11 - Login Screen

If the user already has an account, they can sign in by entering their email and password. After a successful login, they will be directed to a dashboard tailored to their specific role. Users also have the option to sign in using their Google account for a faster login process.



The screenshot shows the 'Reset Your Password' form on the MedCall website. The form is centered on a light gray background. At the top left is the MedCall logo, and at the top right are 'Login' and 'Sign up' buttons. The form itself has a title 'Reset Your Password' and a subtitle 'Please enter your email and we will send an OTP code to reset your password.' Below this is a text input field with an email icon and the placeholder text 'Email'. A red 'Continue' button is positioned below the input field. The footer contains the MedCall logo, copyright information, an 'About MedCall' link, and contact details for support@medcall.com and +972-555-123456.

MedCall

Reset Your Password

Please enter your email and we will send an OTP code to reset your password.

Email

Continue

MedCall
copyright © 2024
[About MedCall](#)

support@medcall.com
+972-555-123456

Figure 12 - Reset Password

The password reset process consists of three steps. First, users enter their registered email address on the email input screen. After submission, an OTP (One-Time Password) is sent to their email for verification on the OTP verification screen. Once the OTP is confirmed, users proceed to the new password screen, where they can set and confirm their new password. This streamlined process ensures users can reset their passwords securely and efficiently.

OTP code verification


We sent an OTP code to your email **tareez@gmail.com**. Enter the OTP code below to verify.

Didn't receive code? [Send again](#)

Figure 13 - OTP Screen

Create New Password

Please choose a new password.






Figure 14 - Create New Password Screen

6.3 User Dashboard

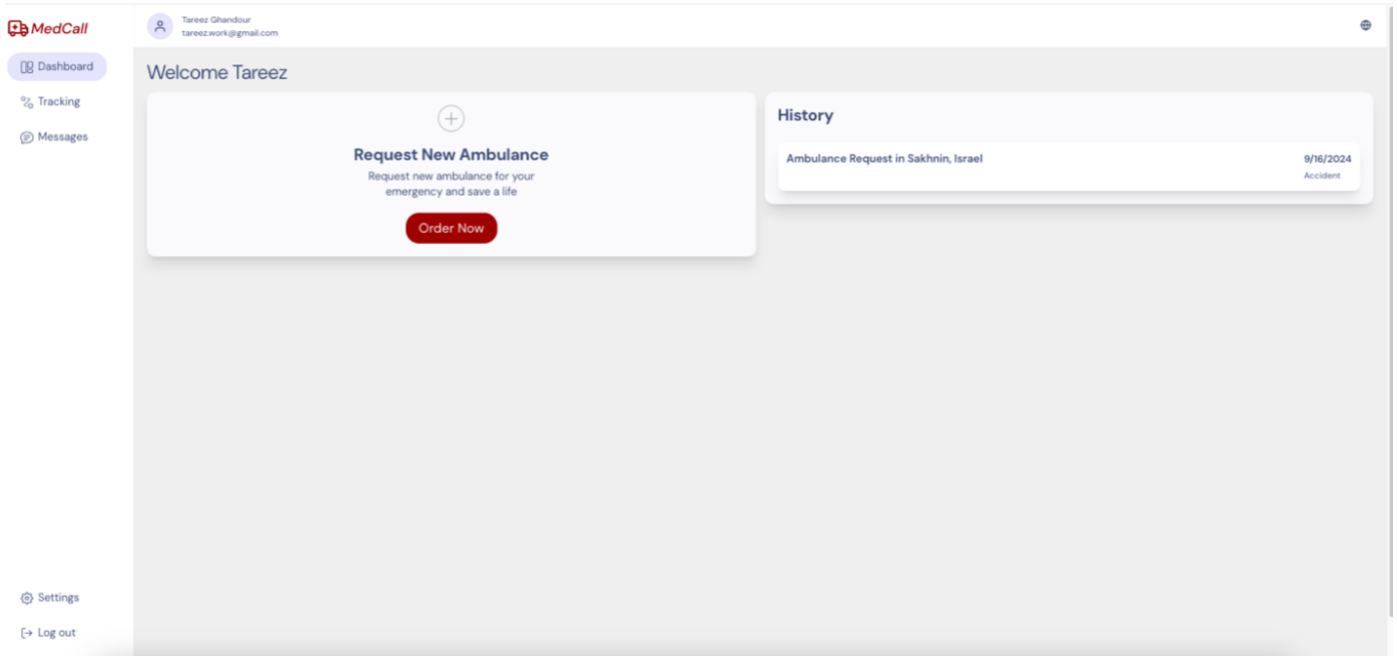


Figure 15 - User dashboard, no active request

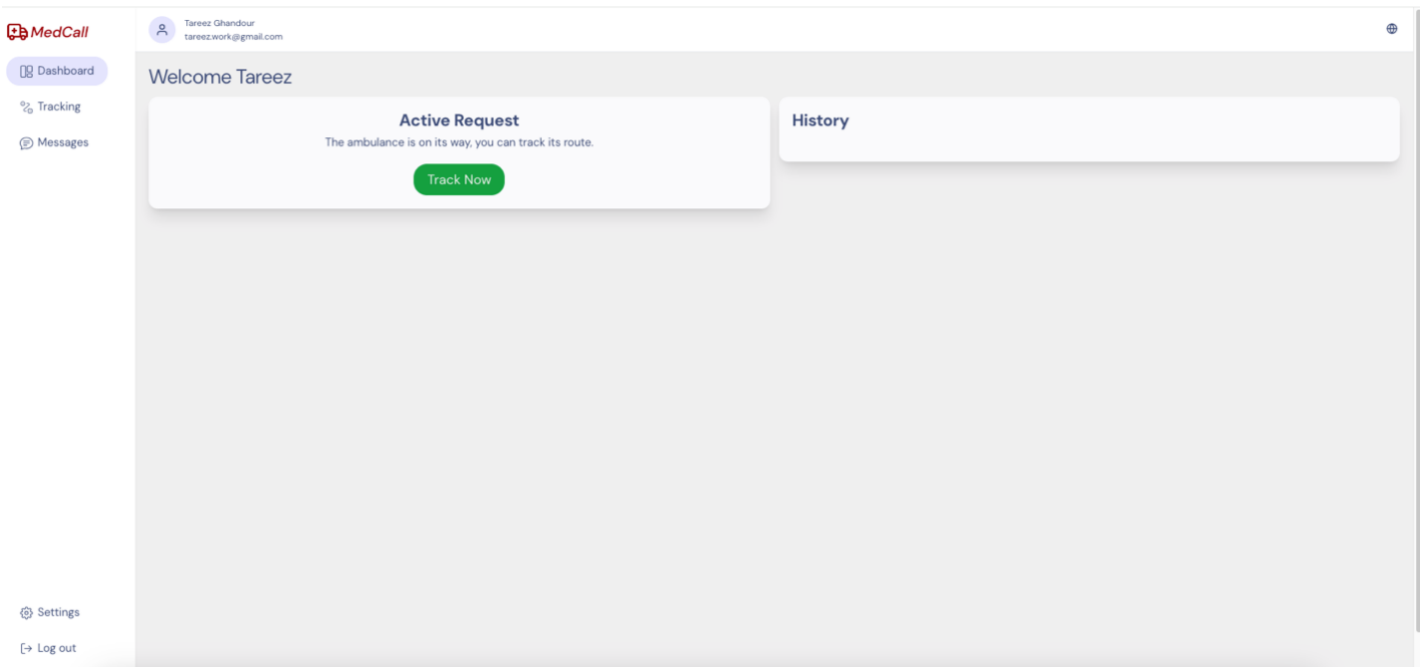


Figure 16 - User dashboard, with active request

The user dashboard screen serves for managing account activities, a sidebar menu allows users to navigate between different sections, including tracking, messages, and settings.

Users can easily track ongoing emergencies, view messages, make a new request, and adjust account settings. When the user makes a new request, the status will be marked as “Pending” until the nearest driver accepts it. Once accepted, the user can track the status of their request in real-time. At the top users can change the language of the platform. Additionally, users have the option to log out from their account.

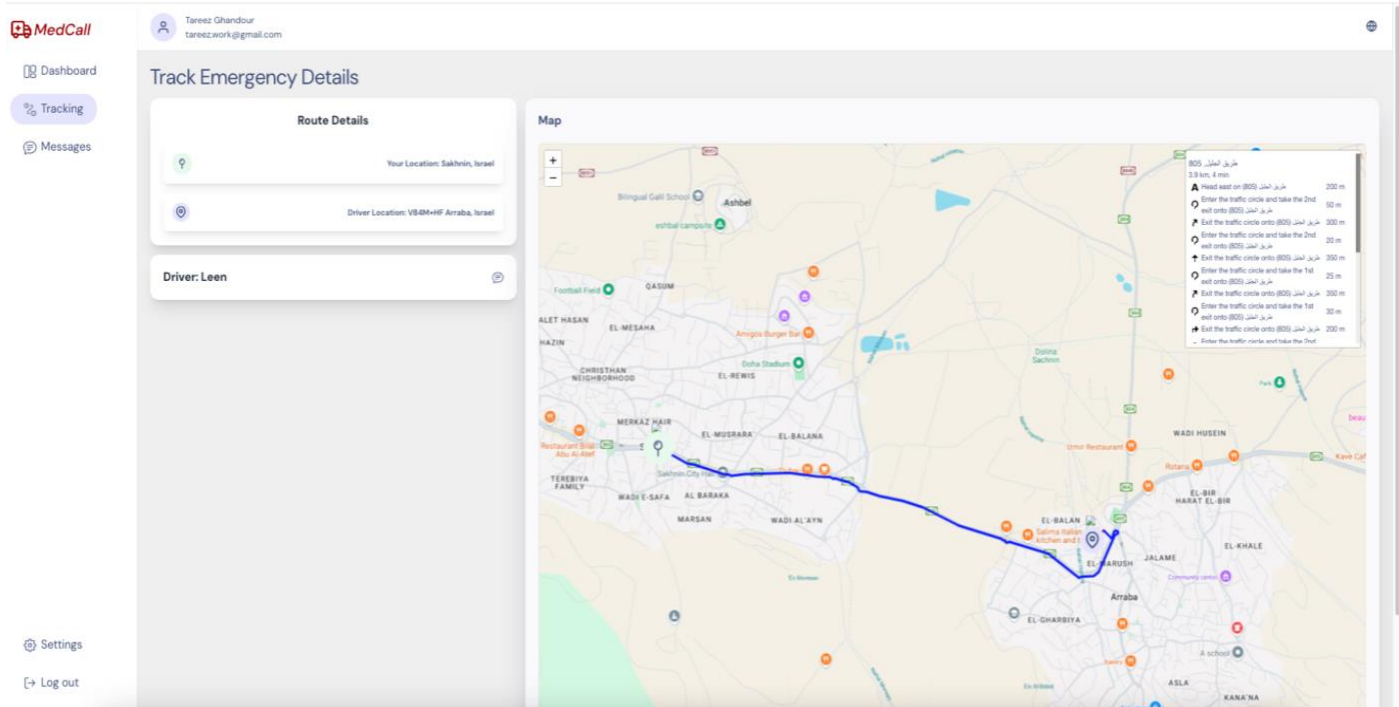


Figure 17 - User dashboard, track active request

The track screen allows users to monitor the real-time location of the ambulance as it responds to their request. The ambulance and the ambulance’s driver details are also shown.

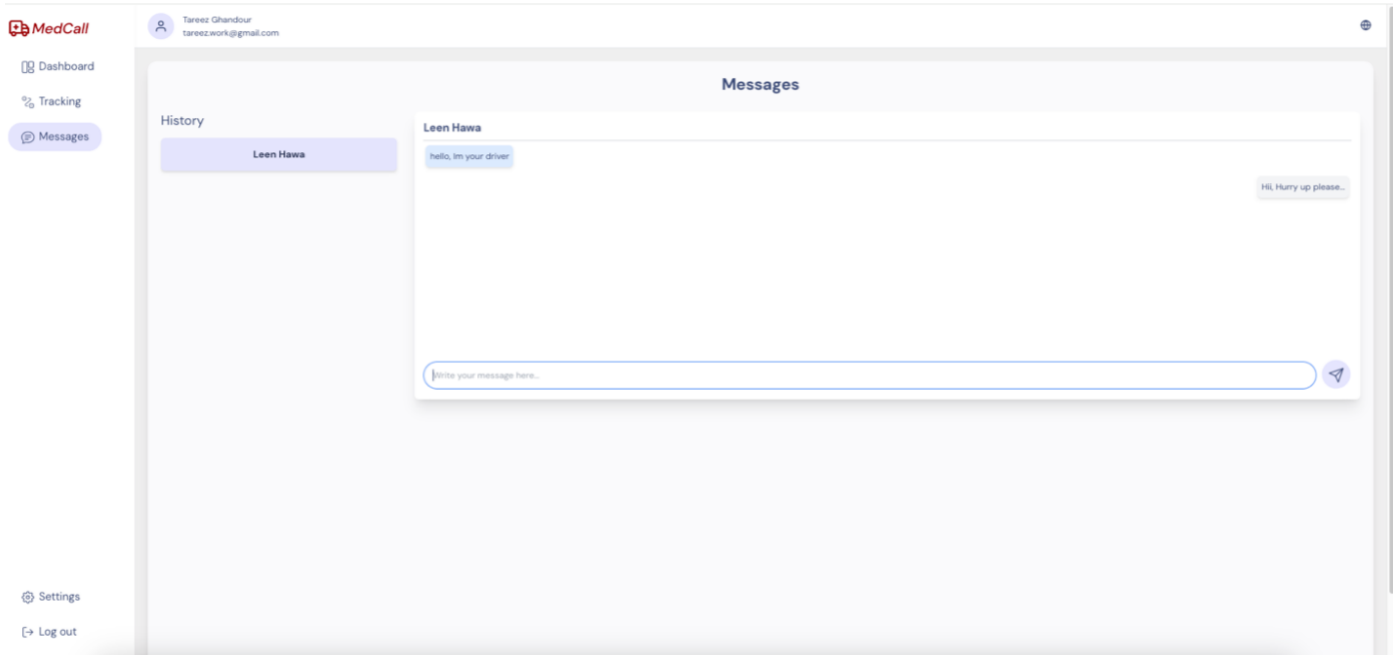


Figure 18 - messages

The message screen serves as a communication hub between users and drivers. Users can view and send messages related to their emergency. Additionally, users can access their previous messages.

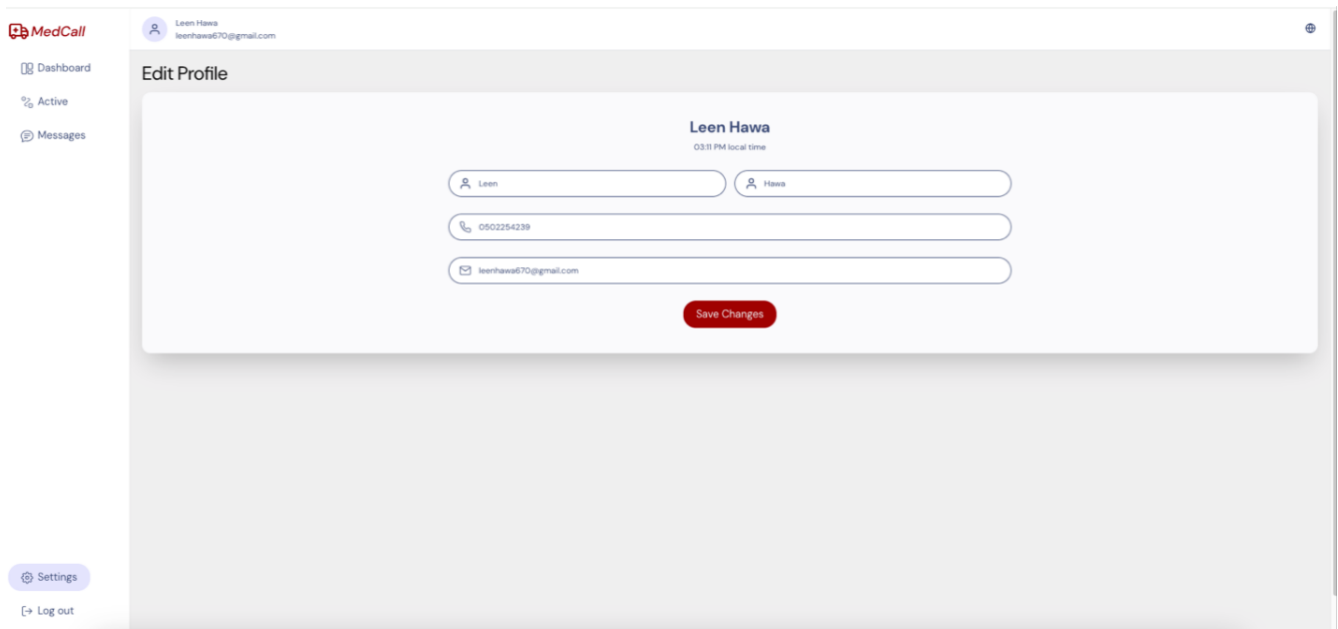


Figure 19 - Edit profile information

The settings screen allows users to edit their profile information. By pressing edit, making changes becomes available. When they finish, they should press “Save Changes”.

6.4 Driver Dashboard

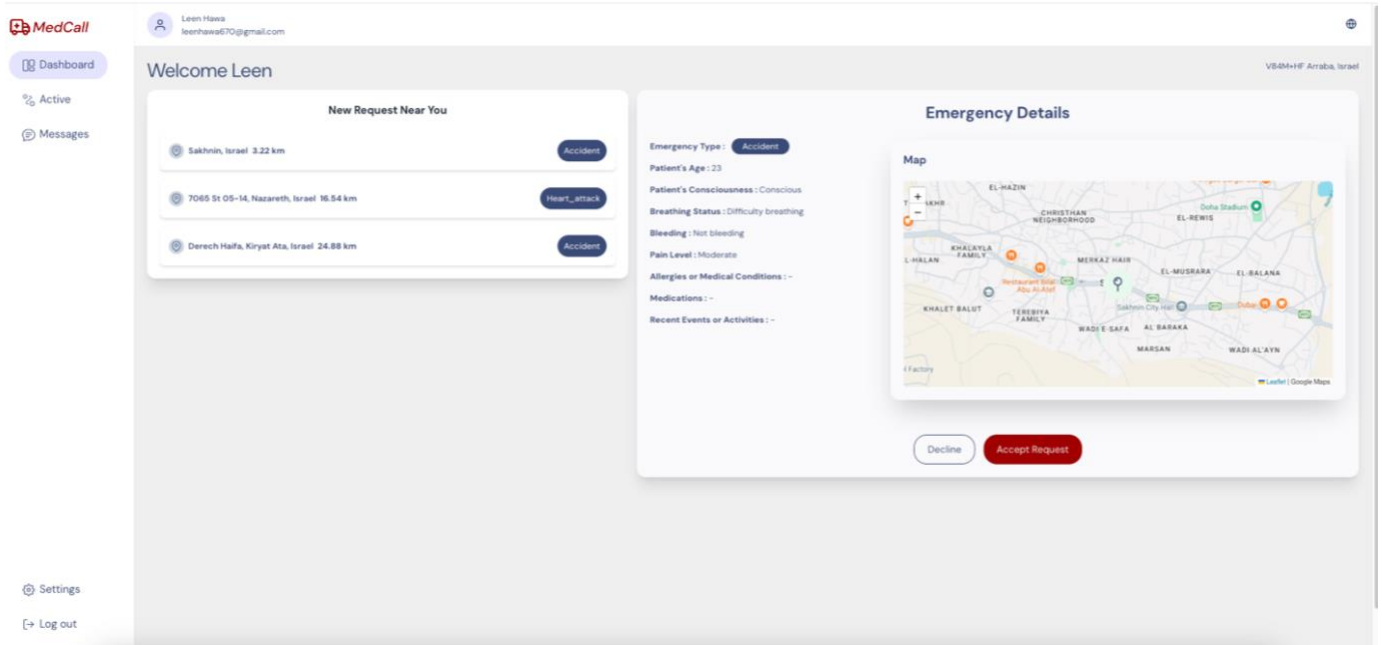


Figure 20 - Driver dashboard

The driver dashboard displays incoming emergency requests. Drivers can see a list of new requests and view details like the type of emergency and location. Clicking on a request shows more information, including the patient's condition. The driver can choose whether to accept or decline the request. This setup helps drivers quickly assess and respond to emergencies, ensuring efficient service delivery.

The driver dashboard has a sidebar menu that allows navigating through the dashboard, view active requests and messages.

At the top users can change the language of the platform.

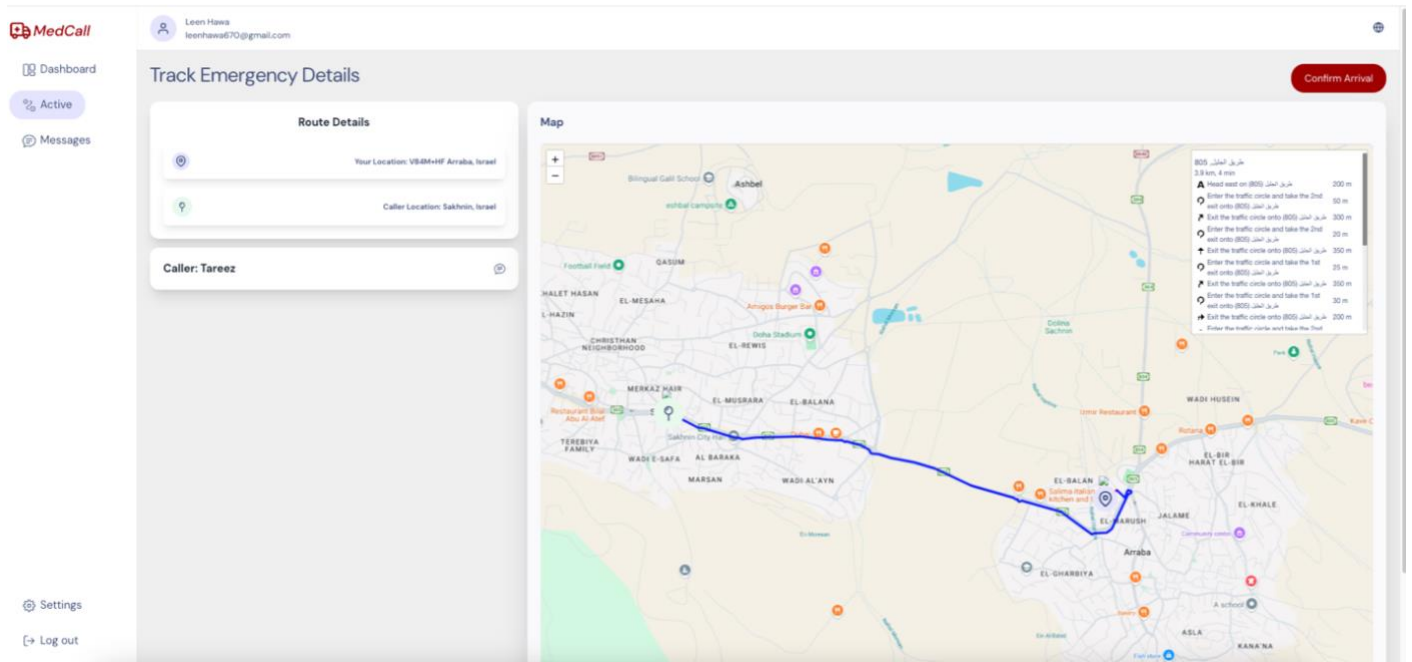


Figure 21 - Driver dashboard, track active request

The active emergency screen allows users to monitor the real-time location of the ambulance and the patient. The route and patient's details are shown.

There is a confirm arrival button that the driver can press when they arrive at the location.

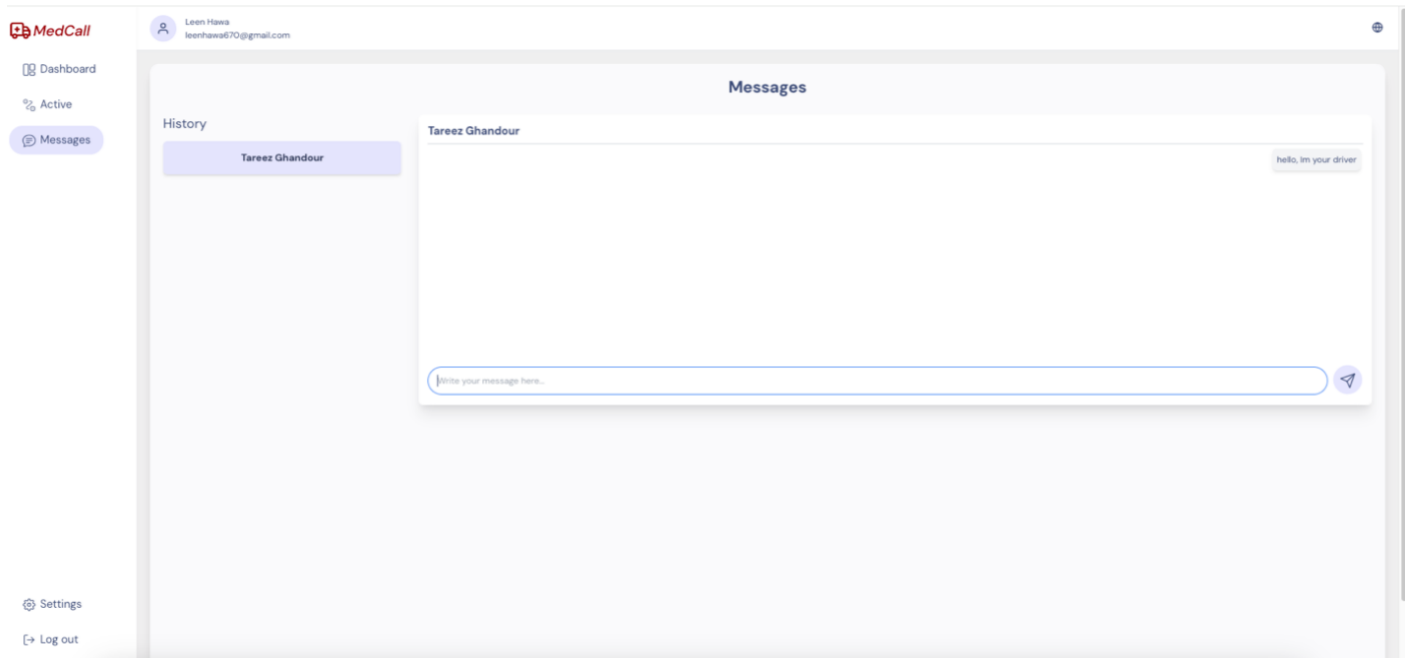


Figure 22 - driver messages

The message screen allows communication between the drivers and users. Drivers can view and send messages to the patients. Also, drivers can see their previous messages history.

6.5 Dispatch Center Dashboard

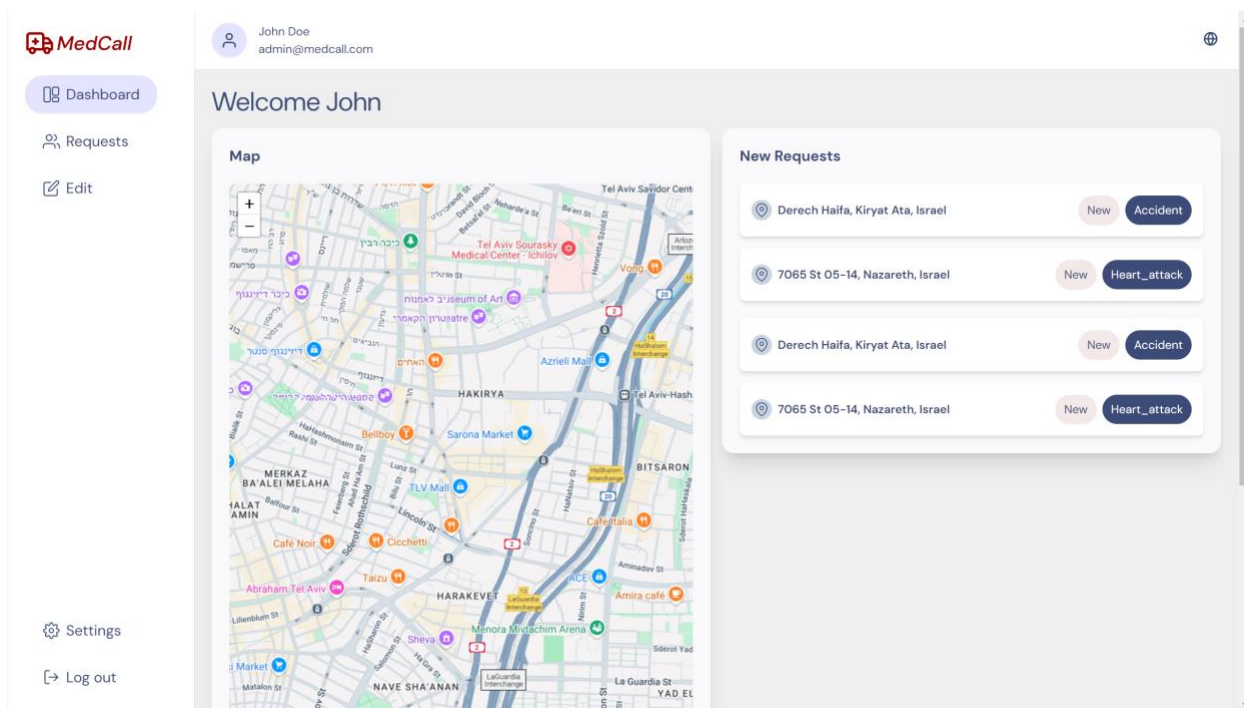


Figure 23 - Dispatch center dashboard

The dispatch center dashboard screen provides a comprehensive overview of incoming emergency requests. The dashboard has a sidebar menu that allows the dispatch worker to navigate through the dashboard, view new drivers request to become active and edit existing drivers.

A map interface displays the real-time locations of requests, clicking on a request displays a popup with detailed information about location.

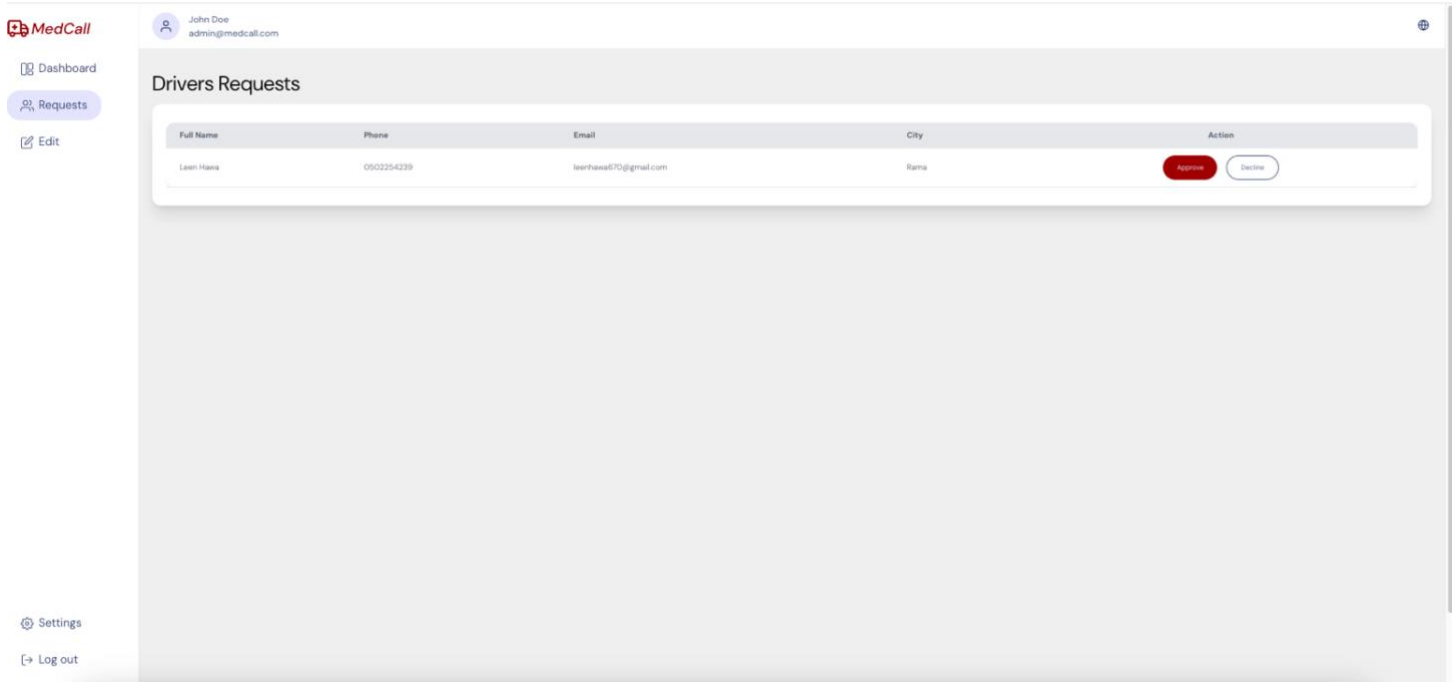


Figure 24 - Dispatch center, review drivers requests

The driver requests review screen includes a table that displays the driver's applicant details including full name, phone, email, and city. Dispatch center staff can quickly assess and decide to accept or decline driver applications, ensuring efficient approval processes.

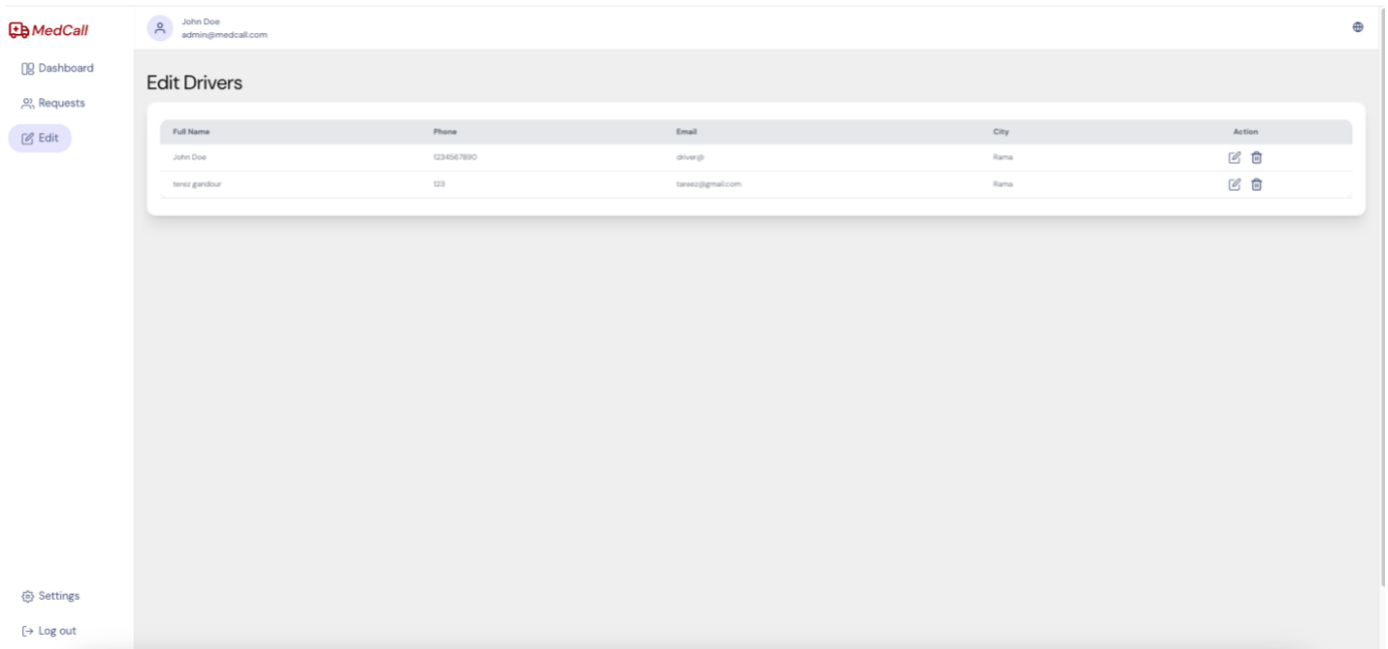


Figure 25 - Dispatch center, edit drivers' info

The edit drivers screen includes a table that displays the driver's details including full name, phone, email, and city. Dispatch center staff can quickly assess, edit, or delete drivers.

When click on delete icon will delete the driver and when click on edit icon will open a modal to edit data

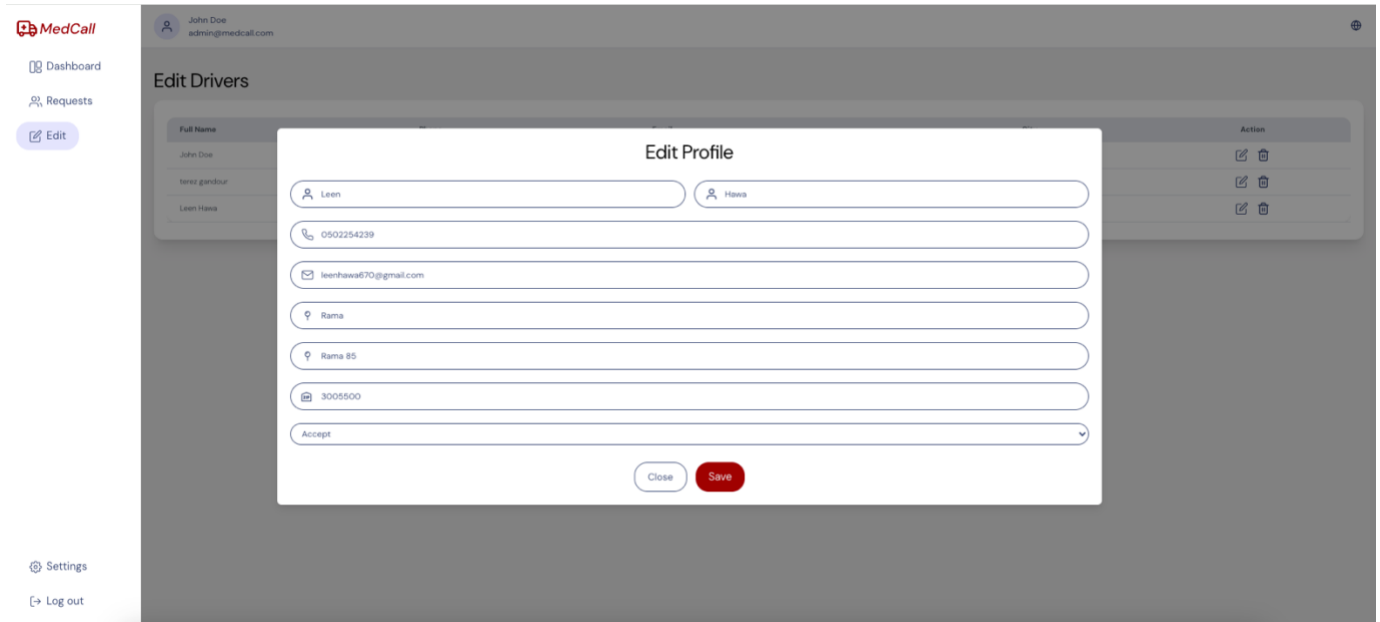
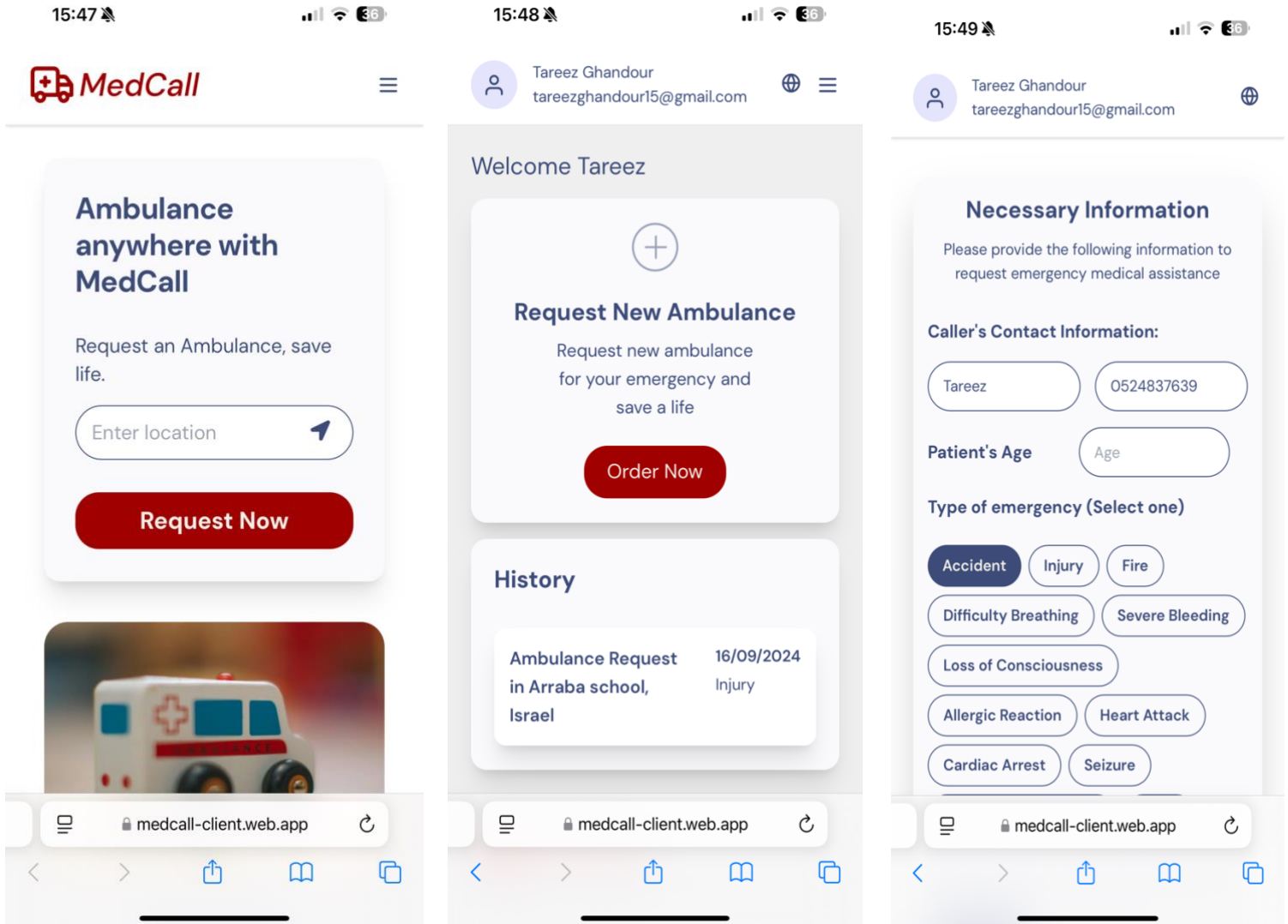
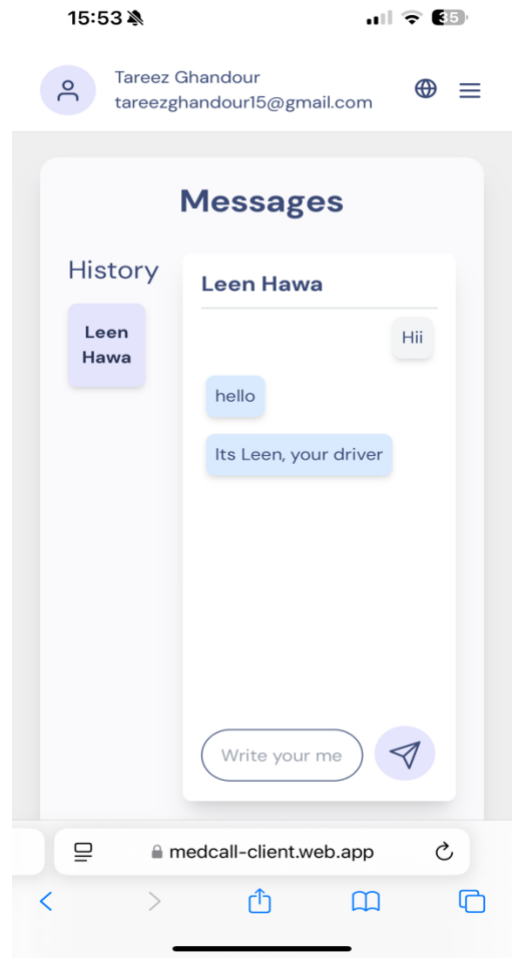
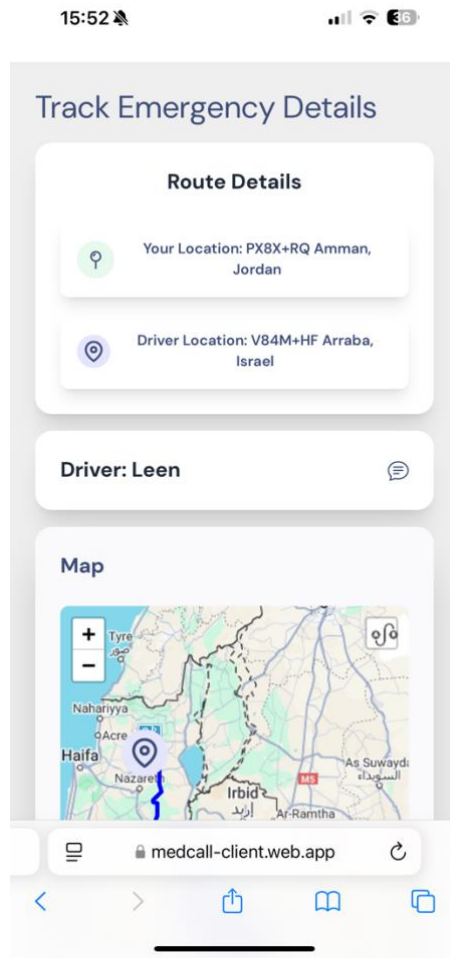
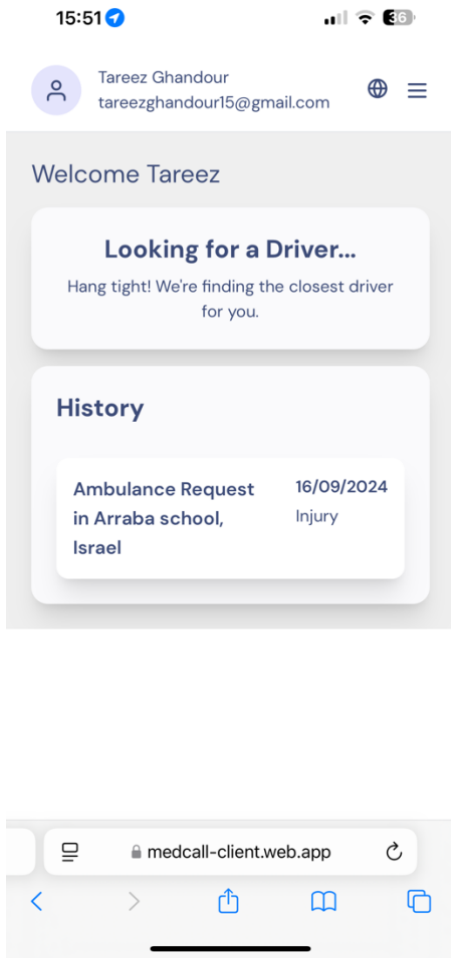


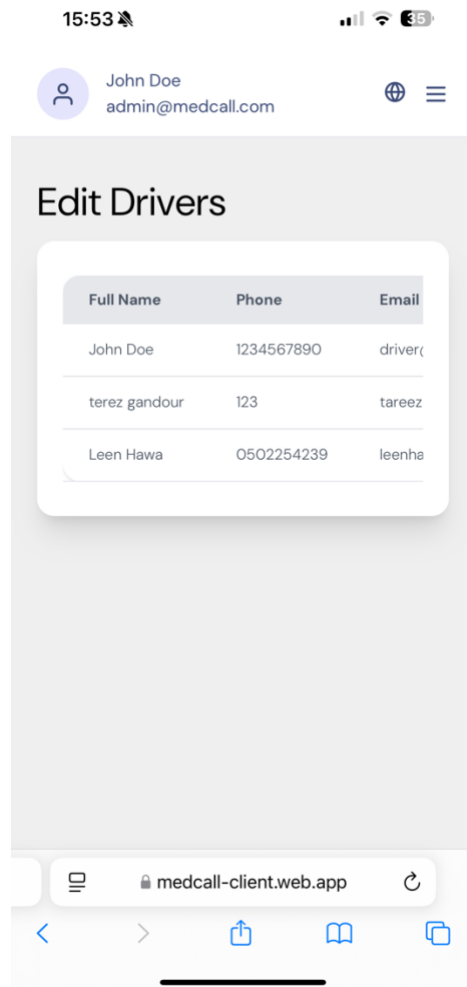
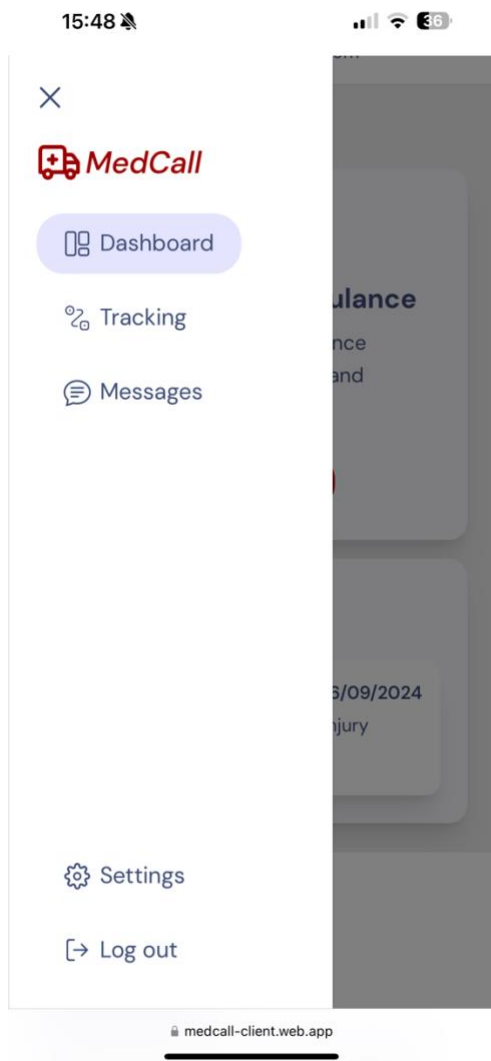
Figure 26 - Dispatch center, edit driver modal

6.6 Responsive Screens

Here some screens of the platform responsive design (mobile)







7. Maintenance Guide (User Help)

Initial Setup:

To get started with the MedCall system, follow these steps:

1. Download the Project:
 - Clone the Repository by run the command: `git clone https://github.com/tareezgh/MedCall`
This will include both the frontend and backend projects.
 - Download ZIP: Download the ZIP file from the repository and extract it
2. Open the projects in Visual Studio Code (VSCode): Open the `medcall-client` project and the `medcall-server` project in separate VSCode windows. Install Dependencies: Open the terminal and run the command `"npm install"` in each file to download any missing packages. This will allow you to work on each project independently.
3. Run the Application:
 - **Start the Backend Server:** In the backend terminal, run: `"npx ts-node src/app.ts"`
 - **Start the Frontend Client:** In the frontend terminal, run: `"npm run dev"`
4. Access the Platform: Open Google Chrome and navigate to the provided link (e.g., <http://localhost:5173/>) to use the platform

By following these steps, you can setup the frontend and backend projects of the platform.

Libraries and Tools:

Frontend (medcall-client):

- **Preact:** Lightweight alternative to React.
- **Vite:** Build tool and development server.
- **Tailwind CSS:** Utility-first CSS framework.
- **Leaflet:** Library for interactive maps.
- **Redux Toolkit:** For state management.
- **Firebase:** For authentication and real-time data.
- **i18next:** For internationalization (supports English, Arabic, Hebrew, and Russian).

- **Axios:** For HTTP requests.
- **React-Leaflet:** Integration of Leaflet with Preact.
- **React-Toastify:** For notifications.

Backend (medcall-server):

- **Express:** Web framework for Node.js.
- **MongoDB and Mongoose:** For database management.
- **JWT:** For authentication.
- **Nodemailer:** For email sending.
- **WebSocket (ws):** For real-time communication.
- **bcrypt:** For password hashing.
- **dotenv:** For environment variable management.
- **Jest:** For testing.

Deployment

- Deployed to Firebase, and deployed to Heroku (for WebSocket support).

Operating System Environment

The operating system environment for this web project revolves around the Google Chrome browser, where the application will be launched and used.

Since the project is designed to be lightweight and flexible to various computing environments, it can be launched on any operating system that can run Google Chrome, like computers and smartphones.

Software wise, the project employs web technologies like HTML, CSS, and JavaScript to develop interactive user interfaces. These languages are compatible with modern web browsers, including Google Chrome, guaranteeing smooth functionality and compatibility.

8. Verification and Evaluation

8.1 Tests Description

	Test N.	Test Description	Expected Result
Registration	1	Attempt to register a new user with valid input data	The user registration should be successful, and the user should receive a confirmation message indicating successful registration.
	2	Attempt to register a new user with and empty input	The registration process should fail, and the user should receive an error message "Please fill in all the fields."
	3	Attempt to register a new user with an invalid email format (e.g. missing "@")	The registration process should fail, and the user should receive an error message "Please enter a valid email."
	4	Attempt to register a new user with a weak password	The registration process should fail, and the user should receive an error message "Password must be at least 8 characters with uppercase, lowercase, and number."
	5	Attempt to register a new user with an email address that is already registered in the system.	The registration process should fail, and the user should receive an error message "Email already in use. Please choose another."
Login	6	Attempt to log in with valid credentials	The login process should be successful, and the user should be redirected to their account dashboard.
	7	Attempt to log in with an empty input	The login process should fail, and the user should receive an error message "Please fill in all the fields."
	8	Attempt to log in with invalid credentials	The login process should fail, and the user should receive an error message "Email or password is wrong. Please try again!"
	9	Attempt to log in with an email address that is not registered in the system.	The login process should fail, and the user should receive an error message "Email not found. Please signup first."
	10	Handle internal server error while login	The login process should fail, because of server error and get http code 500

New Ambulance Request	11	Attempt to request a new ambulance with valid information.	The request should be successfully submitted, and the nearest available ambulance should be dispatched to the specified location.
	12	Attempt to request a new ambulance without providing a location.	The request should fail, and the user should receive an error message "Must enter a location."
	13	Attempt to accept an ambulance request without providing all necessary information.	The request should fail, and the user should receive an error message "Please fill in all the fields."
	14	Verify that the user receives a confirmation message after successfully requesting a new ambulance.	The user should receive a confirmation message confirming that their request has been received.
Accept/Decline/Edit Drivers	15	Attempt to accept a driver request with valid information.	The request should be successfully accepted, and the driver should receive a confirmation message.
	16	Attempt to accept a driver request with an invalid information	The acceptance process should fail, and the driver should receive an error message "Invalid request. Please provide all necessary information."
	17	Attempt to decline a driver request with valid information.	The request should be successfully declined, and the driver should receive a confirmation message.
	18	Attempt to edit driver information with valid input data.	The driver information should be successfully updated, and the driver should receive a confirmation message.
	19	Attempt to edit driver information with missing necessary information.	The update process should fail, and the dispatch center should receive an error message "Please fill in all the required fields."
Accept/Decline Request	20	Driver accepts an ambulance request nearby with valid information.	The driver should successfully accept the request, and the caller and dispatch center should receive confirmation.
	21	Driver declines an ambulance request nearby.	The driver should successfully decline the request, and the dispatch center should receive confirmation.

	22	Driver attempts to accept an invalid ambulance request nearby	The acceptance process should fail, and the driver should receive an error message "Invalid request."
	23	Handle internal server error while accept request	The acceptance process should fail, because of server error and get http code 500

8.2 Tests Results

✓ 23/23

7.7s ↺

- ✓ medcall-server (watch)
 - ✓ tests
 - ✓ requests.test.ts
 - ✓ Request API
 - ✓ should create a new request successfully
 - ✓ should fail to create a request with missing information
 - ✓ should get all requests successfully
 - ✓ should update request successfully
 - ✓ Update Ambulance Request API
 - ✓ should accept ambulance request with valid information
 - ✓ should decline ambulance request with valid information
 - ✓ should fail to accept invalid ambulance request
 - ✓ should handle internal server error during request update
 - ✓ users.test.ts
 - ✓ User API
 - ✓ Login API
 - ✓ should login successfully with correct credentials
 - ✓ should fail login with empty input
 - ✓ should fail login with invalid credentials
 - ✓ should fail login with unregistered email
 - ✓ should handle internal server error
 - ✓ Register API
 - ✓ should register successfully with valid input data
 - ✓ should fail registration with empty input
 - ✓ should fail registration with invalid email format
 - ✓ should fail registration with weak password
 - ✓ should fail registration with already registered email
 - ✓ Accept/Decline/Edit Drivers API
 - ✓ should accept driver request with valid information
 - ✓ should decline driver request with valid information
 - ✓ should update driver information with valid input data
 - ✓ should fail to update driver information with missing necessary fields
 - ✓ should handle internal server error during driver data update

9. Results and Conclusions

We have successfully built a user-friendly platform, that connects individuals in need of help with medical responders in real-time. The platform effectively integrates new and efficient technologies that help optimize EMS response times, which is crucial in emergency situations where every second counts.

The technologies we used were essential to the success of this project. We worked with MongoDB, Express.js, Preact, and Node.js. MongoDB gave us a flexible database that could easily handle large amounts of data, which was important for real-time tracking and storage. Express.js served as the backend framework, making sure the server and client-side application communicated smoothly. Preact helped us build a dynamic and responsive user interface, ensuring a smooth experience for all users. Node.js, with its ability to handle multiple tasks at once without slowing down, was crucial for managing real-time operations, making it perfect for a platform that needed quick data processing and fast responses.

In conclusion, the successful implementation of MedCall shows how modern web technologies can address critical issues in EMS. By combining advanced algorithms with real-time data processing, we created a platform that not only meets the needs of its users but also helps optimize public safety and improve the efficiency of emergency medical services.

10. References

1. <https://en.wikipedia.org/wiki/Node.js>
2. **Nearest Neighbor Algorithm:**
Taunk, K., De, S., Verma, S., & Swetapadma, A. (2019, May). A brief review of nearest neighbor algorithm for learning and classification. In 2019 international conference on intelligent computing and control systems (ICCS) (pp. 1255-1260). IEEE.
3. **Real-time Algorithm:**
Ramamritham, K., Stankovic, J. A., & Shiah, P. F. (1990). Efficient scheduling algorithms for real-time multiprocessor systems. IEEE Transactions on Parallel and Distributed systems, 1(2), 184-194.
4. **Geolocation Algorithm:**
Djuknic, G. M., & Richton, R. E. (2001). Geolocation and assisted GPS. Computer, 34(2), 123-125.
5. **Live chat services and chatbots:**
<https://www.touchpoint.com/blog/live-chat-support/>