Princess Sumaya University for Technology
King Hussein School for Computing Sciences

E-s3af.com

An Online Emergency-Response Application

Prepared By:
Saif Ali (20160754)
Loay AlBarmawi (20170111)
Tariq Al-Madhoun (20160711)

Supervised By:
Dr. Ammar Odeh

Project Submitted in partial fulfillment for the degree of Bachelor of Science in Computer Science

Summer-2020

E-s3af.com

Declaration of Originality

This document has been written entirely by the undersigned team members of the project. The source of every quoted text is cited and there is no ambiguity in where the quoted text begins and ends. The source of any illustration, image, or table that is not the work of the team members is also clearly cited. We are aware that using non-original text or material or paraphrasing or modifying it without proper citation is a violation of the university's regulations and is subject to legal actions.

Names and Signatures of team members:

Saif Eddein Ali Far'oun

Loay Menah AlBarmawi

Tariq Abdallah Almadhoun

E-s3af.com

Summary

This work was mainly inspired by the help of Professor Ammar, Our motive was leaning towards making something that can take part in human life by being there when needed.

The main purpose of this, is simply that it's not an application you would use daily or be addicted to, but something that will be handy and useful when the time calls for it. It will be here to ensure that in case of an emergency, the process of communicating with your escort being it an ambulance or the dispatcher following through with your emergency until help arrives will be as smooth as possible. while the main purpose is handling emergency cases, there are secondary options as such a Medicine barcode scanner which allows finding descriptions of a certain medicine based on the barcode given, as well as a First-Aid guide-book that shows detailed actions to take to provide First-Aid to someone or yourself, until help arrives or you are escorted to seek medical help.

Our methodological approach was to build this application to be available to all kinds of platforms, thus allowing us to cover support for whatever accessible device.

Hence, it was decided that a Web-Application is the most suitable approach to this as it is much easier to port it to other devices rather than developing an application for each separately.

This system should achieve faster response times for emergency cases by the fact it relays location information immediately when you request rather than having to provide it over the phone. It will also be a guide for first-aid cases to know the procedures when needed.

Before starting with the development and writing detailed documentation, we saw that we should research for any existing software/application that contain the same ideology and concept, we stumbled across a few but each had something missing out that the other one had, therefore, it inspired us to combine the missing parts all-together to form an All-in-One Web-Application that will portrait our vision of an improved emergency system in our country and expanding nation-wide.

E-3af.com

List of Abbreviations

Abbreviation	Meaning
UML	Unified Modeling Language
EERD	Enhanced Entity Relationship Diagram
XML	Extensible Markup Language
PK	Primary Key
NN	Not Null
UQ	Unique Index
B	Binary Column
UN	Unsigned Data Type
ZF	Zero Fill
AI	Auto-Increment
G	Generated Column

E-s3af.com

Table of Contents

Chapter 1 Introduction 10

1.1 Overview 10

1.2 Problem Statement 10

1.3 Related Work 11

1.4 Document Outline 13

Chapter 2 Project Plan 14

2.1 Project Deliverables 14

2.2 Project Tasks 15

2.3 Roles and Responsibilities 17

2.4 Risk Assessment 17

2.4.1 Risk Identification and Analysis 17

2.4.2 Risk Response 18

2.5 Cost Estimation 19

2.6 Project Management Tools 20

Chapter 3 Requirements Specification 21

3.1 Stakeholders 21

3.2 Platform Requirements 23

3.3 Functional Requirements 24

3.4 Non-Functional Requirements 27

3.5 Domain Requirements 28

Chapter 4 System Design 29

4.1 Logical Model Design 29

4.1.1 Use Case Diagram: 30

4.1.2 Class Diagram: 35

4.1.3 Component Diagram: 36

4.1.4 Activity Diagrams: 37

4.1.5 39

4.1.6 41

4.1.7 Communication Diagrams: 44

4.1.8 Database Schema: 46

4.2 Physical Model Design 48

4.2.1 Reports Design 48

E-S3af.com

Userbase report 48
Patient Report 48
Ambulance Report 48
4.2.2 User Interface Design 49
4.2.2.1 Patient Interface: 49
4.2.2.2 Patient Interface: 50
4.2.2.3 Dispatched Unit Interface: 51
4.2.2.4 Dispatcher Interface: 52
4.2.3 Database Design: 52
Chapter 5 Implementation 54
Programming Languages, tools and APIs 54
Overview 55
Main Functionalities 57
Normal User 57
Sign-In using phone number 57
Request Ambulance / Distress Signal 58
Medicine Scanner 59
First-Aid Tips 61
Dispatched Unit 62
Get Current Request Information 62
Update Case Information 62
Duty Hour Control 64
Dispatcher 65
Sign-In using email and password 65
View Active Markers and Related Details with Live-Tracking 66
Assign Dispatched Unit based on Phone Numbers 70
Cancel Patient Request 73
Track Patient Requests Using Phone Number 74
Send Patient Data & Notification to Dispatched Unit through POST request 76
Send Notification to Patient through POST Request 77
Features 77
Chapter 6 Testing 78
6.1 Testing Approach 78
6.1.1 Functional Testing 78

E-S3af.com

6.1.2 Non-Functional Testing 79
6.2 Testing Tools 81
6.3 Tested Features 82
6.4 Discussion 83
Chapter 7 Conclusions and Future Work 85
Appendix A Users' Manual 86
Regular User 86
Dispatched Unit 93
Dispatcher 95
Appendix B Document Changes 104
Appendix C Code Documentation 105
References 106

E-s3af.com

Table of Figures

Figure 1: Project Tasks Gantt Chart 15
Figure 2: Use Case Diagram for Es3af.com 29
Figure 3: Class Diagram for Es3af.com 34
Figure 4: General Component Diagram for Es3af.com 35
Figure 5: Activity Diagram for Requesting an Ambulance 36
Figure 6: Activity Diagram for Medicine Scanning 37
Figure 7: Login Sequence Diagram**Error! Bookmark not defined.**Figure 8: SOS Request Sequence Diagram 39
Figure 9: State Transition Diagram for Requesting an S.O.S 40
Figure 10: State Transition Diagram for Dispatched Units Assignment**Error! Bookmark not defined.**Figure 11: State Transition Diagram for Dispatcher Receiving an Emergency Request 42
Figure 12: Login Communication Diagram 43
Figure 13: First Aid Display Communication Diagram**Error! Bookmark not defined.**Figure 14: Enhanced Entity Relation Ship Diagram for Es3af.com 45
Figure 15: Entity Relationship Diagram for Es3af.com 46
Figure 16: Patient UI for Main Menu**Error! Bookmark not defined.**Figure 17: Patient Login UI 49
Figure 18: Dispatched Unit UI After Being Assigned to a Request 50
Figure 19: Dispatcher UI Main Dashboard 51
Figure 20: User Table 51
Figure 21: Next of Kin Table 51
Figure 22: Medical Case Table 51
Figure 23: Medical History Table 52
Figure 24: First Aid Table 52
Figure 25: Login Table 52

E-s3af.com

Table of Tables

Table 1: Related Works Comparison Table 11
Table 2: Document Outline 12
Table 3: Project Deliverables 13
Table 4: Project Tasks Description 15
Table 5: Team Members Roles & Responsibilities 16
Table 6: Risk Identification & Analysis 16
Table 7: Risk Response 17
Table 8: Project Cost Estimation 18
Table 9: Project Management Tools 19
Table 10: Project Stakeholders 21
Table 11: Mobile Client Requirements & Their Importance 22
Table 12: Web Requirements & Their Importance 22
Table 13: Server Requirements & Their Importance 22
Table 14: Functional Requirements Description 23
Table 15: Non-Functional Requirements Types 26
Table 16: Userbase System-Generated Report 47
Table 17: Patient System-Generated Report 47
Table 18: Ambulance System-Generated Report 47

E-s3af.com

Chapter 1
Introduction

1.1 Overview

E-s3af.com is a web-based mobile application that allows users to call an ambulance in case of medical emergencies using their mobile devices, it utilizes the GPS feature to give the dispatcher an accurate location of the person needing medical attention so that they send an ambulance to them as soon as possible, it also allows the authorities to check the identity of the person calling for help and find if they have underlying health problems or disabilities so that they deal with the situation properly according to protocol.

The application also contains features such as a barcode scanner for medicines to identify it and its uses and the instructions of taking the drug and the proper or recommended dosages and has an SOS feature for people in rural areas such as the desert or the sea to call for help.

1.2 Problem Statement

Within the rise of a pandemic, and a full lockdown being issued, we started wondering on how to improve the situation for the citizens in cases of emergency, and how to improve the efficiency and the accessibility towards treating emergencies in a faster manner and minimal room for error. No matter who you are, or where you are, there should be a way where you can seek help in cases of emergency.

Discussing many cases, we've deduced that human error, for example, can be a critical problem when giving a location or if the person didn't know where they exactly were located if they were in a foreign district, thus, we proposed a built-in feature that relays the geo-location and coordinates, alongside the contact details to the dispatcher and the dispatched unit, therefore, allowing a more accurate track of where the request came from.

Rural Areas also were a problem in our research as you can't tell someone directions in the middle of a dessert or while you're sailing in the sea, therefore we expect our application to not only cover ground, but sea emergencies to provide further safety measures to everyone.

Our Application will be a web-based application that is integrated onto mobile devices to allow accessibility for anyone in need, as limiting our options to mobile devices only would essentially affect a certain criterion of people, even if, it was a small minority, thus we will target as many people to be able to use this application as possible when needed.

E-s3af.com

1.3 Related Work

Firstly, we would like to state that an application of this type has not appeared in the Jordanian market yet, and we are working on the assumption that there is a need for such applications to provide services as e-government effectively.

NA998 [1]: This application belongs to the UAE government, where it calls an ambulance and sends the location via smartphone, it also provides first aid tips and procedures for most cases.

The drawbacks are the lack of rural area SOS features, and lack of biometric login or voice XML, so special needs users would have a hard time using the application, as well as lack of pharmacy data and locations and no access to database of medicine where users can see the medicine they scanned.

As'efny (أسعفني) [2]: This is a Saudi-Arabian application that sends an ambulance via smartphone location, it also provides SOS calls and pharmacy locations as well as first aid tips, it also has a medical history where a user specifies if they have a heart condition or diabetes and if they are on certain medications.

The drawbacks are a lack of special needs user support and no access to the medicine database.

Indian Red Cross First Aid [3]: This application calls an ambulance and provides emergency numbers as well as first aid tips for many emergencies including animal attacks.

The drawbacks of this application are the lack of special needs support, a medical history page, medicine and pharmacy data, user data, and the user interface is very weak. India is a very large country that shouldn't have one system, but more than one for each region to ease the load on servers.

Our application will take from the best features of all 3 applications and will add the features that were missing from them such as Special needs support, where login would be easier for people with disabilities by using voice instead of typing or fingerprint scanners on the Smartphone of the user, it would also have a more accurate and specific medical history folder for each user, not only specifying heart problems and diabetes but allergies to foods, medicines, and other conditions, it would also include the locations of pharmacies and they are open or not along with their numbers and will contain an offline list of first aid tips with animated GIFs to help the user, the applications would also include access to the database of medicines and their prices as specified by the Jordanian government.

Essaf.com

For comparison purposes, the table below simplifies the main differences between the previously mentioned related works and our application:

Table 1: Related Works Comparison Table

	Es3af.com	NA998	As'efny	Indian Red Cross First Aid
Request an Ambulance	X	X	X	X
Rural SOS	X		X	
Voice XML	X			
Pharmacy Locations	X		X	
First Aid Tips	X	X	X	X
Medicine Scanning	X			
Medical History	X	X		

E-s3af.com

1.4 Document Outline

Table 2: Document Outline

Chapter Title	Description
Chapter 1: Introduction	**Brief overview of the project and its problem statement, it shows any prior work and describes the audience targeted**
Chapter 2: Project Plan	**Detailed description of the project's flow and the tasks with their distribution amongst team members, it also contains the risk assessment and cost estimation and tools required.**
Chapter 3: Requirements Specification	**The functional and non-functional requirements and required software and hardware specifications with the declaration of stakeholders**
Chapter 4: System Design	**Detailed analysis for the system components for both the high- and low-level design using UML diagrams and such**

E-s3af.com

Chapter 2
Project Plan

2.1 Project Deliverables

Table 3: Project Deliverables

Name	Description
Source Code	The code for the website, the mobile application, and the server will be available.
Design Drawings	Drawings explaining the system and its components will be provided in the documentation.
Databases	Databases including a user database, dispatcher database, the dispatched unit database will be provided along with additional databases that contain medical information.
Executables	An android executable file will be provided by the end of the project.
Documentation Files	Code Documentation, System Documentation
Website	A website with the service implemented on it will be provided.
Product Prototype	A functional prototype of the project will be provided and will include the user, dispatcher, and dispatched unit views along with the logs and access to the features specified in the documentation.
Progress Report	A timeline specifying the amount of work that was finished each week will be provided in the documentation.
Logs	Logs of the interactions between the stakeholders and the system will be saved and able to be accessed by the meant parties.
Faster Emergency Vehicles Response Time	The application will help organize and optimize the way the ambulance system works and it is expected to minimize the time needed for an ambulance to arrive at a given destination.

E-ssaf.com

2.2 Project Tasks

For the project tasks, we split up the work accordingly with our time-space to be as efficient as possible, since we're going by a waterfall model, our tasks were pretty dependent on putting together our assigned work.

The main achieved tasks in this documentation are:

- Research
  - Define Project Scope: This involved finding the idea to be made and how to portray it
  - Research Review: This was mainly user research by the team to see a prior work and find several approaches and features to be added into our idea
  - Requirements Gathering: After doing the main research, it was time to dig deeper into the topic to see what we would need to get things running and what we would need for a smooth operation
  - Kickoff Meeting: Having completed the previous task, it was time for a meeting to wrap up the Research part and discuss it with our mentor.

- Design Phase
  - Development Strategy: After completing our research, it was time to determine our strategy and how we want to develop this application as a regular software or web-based software.
  - High-Level Design: Mainly the Logical Design of our project including diagrams and sample scenarios.
  - Database Design: A structured third normal form database that shows our expected database design.
  - UI Design: Simple interfaces from several perspectives of different users
  - Design Review w/ Stakeholders: a small meeting with our mentor to check our current progress and instruct us for any changes.
  - Design Revisions: These are optional tasks if the previous meeting indicated some changes to our work.

E-saf.com

Table 4: Project Tasks Description

Task Number	Task Name	Time
TNR1	Define Project Scope	1 day
TNR2	Research Review	6 days
TNR3	Requirement Gathering	1 week 2 day
TNR4	Kickoff Meeting	1 day
TND1	Development Strategy	3 days
TND2	High-Level Design	1 week 4 days
TND3	Database Design	3 days
TND4	UI Design	2 days
TND5	Design Review	1 day
TND6	Design Revisions	3 days
Total Time		
43 days approx. 6 weeks		

3af.com

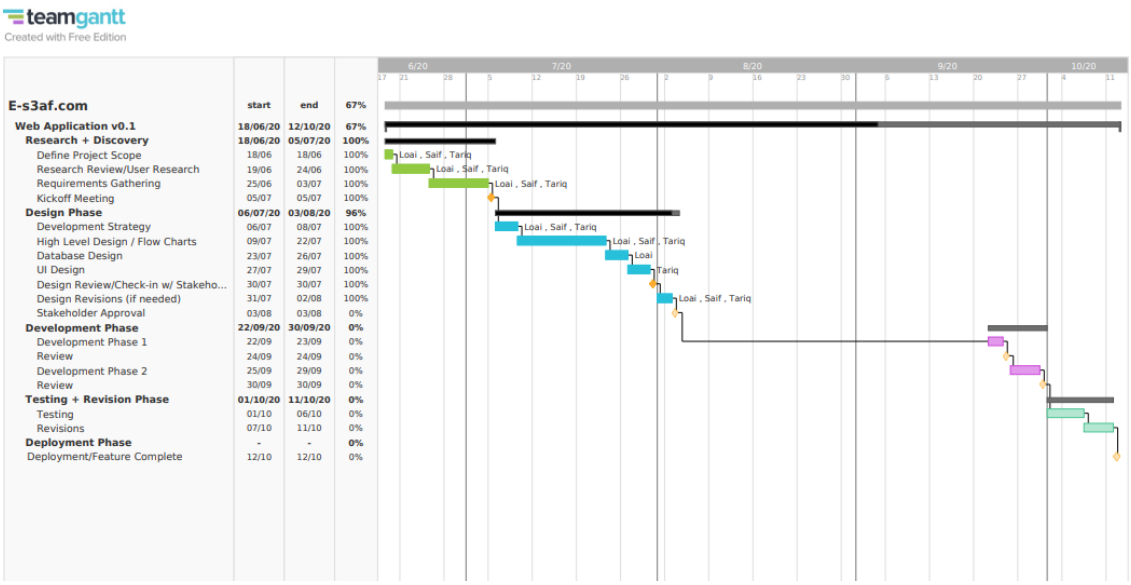


Figure 1: Project Tasks Gantt Chart

2.3 Roles and Responsibilities

Table 5: Team Members Roles & Responsibilities

Team Member Name	Responsibilities
Loay AlBarmawi	Front-End, Back-End, Documentation.
Saif Ali	Back-End, Server Programming, Documentation.
Tariq Al-Madhoun	Graphics Design, Front-End, Documentation.

2.4 Risk Assessment

2.4.1 Risk Identification and Analysis

Table 6: Risk Identification & Analysis

Risk ID	Task ID	Risk Description	Probability
RS1	**T1.1** **T1.2**	User Data Entry Error	Medium
RS2	**T2.1** **T2.2**	Dispatcher Accidental Decline Of User Requests	Low
RS3	**T3.1**	Server Stops Working	Low
RS4	**T4.1**	False Emergency Order By User	High

E-S3af.com

2.4.2 Risk Response

Table 7: Risk Response

Risk ID	Task ID	Task	Response
RS1	**T1.1**	Confirm Entry	Popup to confirm correct data entry of User urging them to double-check
	T1.2	Check Repeated Information	Search DB for repeated data such as National ID and Phone Number and notify user they are already used upon registration and insertion of medical history data
RS2	**T2.1**	Confirm Decline	Popup to confirm Dispatcher's declination of the order along with the reason (Spam, False Alarm...etc)
	T2.1	Rollback Declination	The ability for Dispatcher to rollback declination upon realizing the mistake
RS3	**T3.1**	Emergency Number List	Provides User with a list of emergency numbers in case of server failure
RS4	**T4.1**	Emergency Order Cancellation	Allows User to cancel emergency order within 10 seconds of submitting

E-s3af.com

2.5 Cost Estimation

After researching the costs required to have a fully operational system with scalability and efficiency in mind; we have decided on the following costs:

- Website Domain Name: Hosting a website is an extremely important requirement, where the service can't work if we don't have one, a ".com" domain costs $11 a year making it a cheap but necessary cost.

- App Hosting Platform: To make the app easier to install, we must put it up on Google Play so that our users don't have to install it as a third-party app, Google Charges a $25 one-time fee.
- App Hosting Server: For us to be able to maintain a large sum of data and make sure that scalability is not going to be a problem, we concluded that we need an external server to host most of the processing required by the application, servers that can support large databases and many users can range from $960 to $3000 a year depending on the specifications and constraints of the server.

- Test Devices: Test devices are an important part of development, where we'll need a variety of devices of different generations and versions of the mobile operating system we will be developing the application for, we have personal devices that could be used but we will require more devices, used devices can be found cheaply on the internet, so a maximum of $700 will be needed.

Table 8: Project Cost Estimation

Name	**Cost Estimate**	**Description**
Website Domain Name	USD 11 -Annually	A ".com" domain averages at around USD 11 annually, we will use it for the E-s3af website.
App Hosting	USD 25 - One time	**Google** charges a one-time $25 fee to get a **developer** account on **Google Play**
App Hosting Server	$960 to $3000 -Annually	A server to host the application and handle the processing. An App like this requires a dedicated host as it will have nationwide coverage
Test Devices	USD 700	Different devices to check for compatibility issues in rendering the web-app and such.
	Yearly	**One-Time**

E-s3af.com

Total Cost	$971-$3011	$725

2.6 Project Management Tools

Table 9: Project Management Tools

Name	**Description**	**Where it was used**
TeamGantt	Gantt Chart software	It was used to make Gantt charts
LucidChart	UML Diagrams Web-app	It was used to design most of the UML Diagrams
Bootstrap Studio	A bootstrap framework design software	It was used to make a simple interface to display the idea
Visual Paradigm	UML Diagrams tool	It was used to design some UML Diagrams
MySQL Workbench	Database creation tool	It was used to create the database and the EERD
Adobe Photoshop	Graphic Design Tool	It was used in designing the logo and assisted in creating the preliminary interface
Microsoft Word	Documentation Tool	It was mainly used in creating the documentation and formatting
Google Drive	Cloud Hosting Tool	It was used in hosting our project's files and organize the workflow

E-S3af.com

Chapter 3
Requirements Specification

3.1 Stakeholders

Stakeholders in our system and their specific interaction can be broken down to three points:

- User: The user in our system is divided into two categories as follows:
  - Regular User: The regular user is the one looking for medical information or medical tips and first aid instructions for daily life injuries, including (but

not limited to): First- and second-degree burns, bruises, scratches, nosebleeds, etc…

The roles of this type of user can be broken down into the following roles:

- Use the first-aid tips to find the required procedure.
- Search for the required medicine through the database included in the application.
- Look for open pharmacies in their area through our application using Google maps.

- Patient: A patient is also considered as the other type of users in which they require immediate medical assistance from a third-party, such cases include (but not limited to): Third Degree burns, car accidents, animal attacks, heart attacks, etc…

  The roles of this type of user can be broken down into the following roles:

  - Request an ambulance through the application.
  - Request an emergency vehicle through the SOS feature in the application.

- Dispatcher: A dispatcher is considered to be the person responsible for arranging and responding to all user requests, the dispatcher's interaction with our system can be broken down into the following roles:
  - Validate user requests; both SOS and Ambulance.
  - Display active requests.
  - Display Logs
  - Assign/Unassign dispatched units according to the given scenario.
- Dispatched Units: A dispatched unit is considered to be the person/s responsible for driving the requested vehicle and performing the medical duties required of them, the dispatched unit's interaction with our system can be broken down into the following roles:
  - Get to the destination provided by the dispatchers through the application.
  - Decide what actions to take depending on the description of the scenario and the patient's medical history.
  - Verify the arrival to the destination.

The below table summarizes the above:

Table 10: Project Stakeholders

Stakeholder	Roles
User (Regular)	**Search for Medicine Data, Open Pharmacies & First-Aid Procedures**
User (Patient)	**File an emergency request**
Dispatcher	**Validates Requests, Assigns Units & Moderates the system actively as part of their duty**

E-sbat.com

Dispatched Unit	**Accepts/Gets assigned to their mission and submits fulfillment of duty to preserve accurate logs.**

Color codes:

Red - Critical Role

Green - Non-Critical Role

E-s3af.com

3.2 Platform Requirements

Mobile Client

Table 11: Mobile Client Requirements & Their Importance

Requirement	Description	Importance
PRMC1	Access to the Internet	Required
PRMC2	Access to Google Services	Required
PRMC3	Android OS/iOS	Required
PRMC4	Browser	Recommended

Web Client

Table 12: Web Requirements & Their Importance

Requirement	Description	Importance
PRWC1	Access to the Internet	Required
PRWC2	Access to Live Location	Recommended
PRWC3	Operating System with GUI	Required
PRWC4	Browser	Required

Server(s)

Table 13: Server Requirements & Their Importance

Requirement	Description	Importance
PRS1	Access to the Internet	Required
PRS2	Large Storage Units	Required
PRS3	High-end Components	Recommended
PRS4	Host Software	Required

3.3 Functional Requirements

Our functional requirements are listed as follows:

E-3af.com

Table 14: Functional Requirements Description

Requirement	Description
FR1	Login
FR2	Request an Ambulance
FR3	Medicine Reader
FR4	SOS Signal
FR5	First-Aid
FR6	Display Active Requests
FR7	Display Request Details
FR8	Display Logs
FR9	Duty Hours Control
FR10	Display Notifications for Case/Request/Emergencies
FR11	Display Patient Request Details and Personal Information
FR12	Language Control

- Login: This is considered the main feature in which a user can access the application and its features, and to achieve a hassle-free experience, we have decided on a mobile number-based login, in which users can sign into our application without any frequently faced problems that may delay the process; i.e.: forgetting your password.
- Request an ambulance: This requirement is the heart and soul of our application, and it is the most important feature in the entire system. This functionality can be demonstrated when the patient requires immediate medical assistance, in which they request the ambulance with minimal effort by tapping on the respective button on the application, or by using our integrated voice XML, the application then takes it from there and a dispatched unit is assigned to the patient who needs help.
- Medicine Reader: The medicine reader feature is considered one of the important functionalities as well, as it provides users with the required information about any scanned medicine that exists within our database. This feature properly advises the doses, side effects, as well as the price of said medicine.
- SOS Signal: Our application can be distinguished from any other application for having such functionality. This can be demonstrated in rural areas where the required medical services can prove to be difficult to provide, however, our

E-3af.com

application makes sure to flash an SOS signal when needed, and the responsible department will be able to send immediate help regardless of the location.

- Preliminary First Aid: We strongly believe that our application can be useful to everyone who has a smartphone in hand, this feature helps educate the users on how to deal with their everyday injuries from papercuts to minor burns and scratches, as it provides an in-depth explanation on how to treat each one, with gifs to further explain the exact procedure.
- Display of Active Requests: We make sure to provide the needed service not also to the users, but dispatchers as well. This feature can help dispatchers display, track, and manage active requests sent from the users, so they can act accordingly while advising the proper instructions to any dispatched unit that is on its way to help.
- Display Request Details: This functionality can branch off from the previous one, as it allows the dispatcher to further check regarding the selected request, and display all the medical information and medical records of the patient at hand. This functionality will allow the dispatcher to prepare the units ahead of time upon their arrival to the patient, and substantially accelerate the process in which the situation is dealt with.
- Display Logs: For an application that functions in such a critical environment, we believe that keeping logs is an extremely important part of the process, as it further enriches the medical records of all the patients using the application (for the reason mentioned previously), as well as keep all the cases in check for any expected incidents with the very same patients.
- Duty Hours Control: This feature allows the dispatched units to be able to control their duty hours based on their work schedules, such that no requests/notifications will be sent on an inactive unit, instead, all requests will be rerouted to the closest active units to the requester.
- Display Notifications for Cases/Requests/Emergencies: Dispatched units will be able to view their ongoing notifications for any cases/nearby requests/nearby emergencies, and respond to them accordingly. The response time on each case will prove to decrease due to this feature over time, which will further increase the chance of reaching the patient on time without any unnecessary delays.
- Display Patient's Request Details & Personal Information: In addition to the previous point, all dispatched units will be able to view the request details as well as the patient's personal information and medical records, to provide the assistance required while making sure not to harm the patient in the process.
- Language Control: We believe that our application should be accessible by every citizen in our country, and the application being bilingual will greatly assist in achieving this purpose. Any user will be able to switch to their language of preference (EN/AR) when needed.

H-3af.com

E-s3af.com

3.4 Non-Functional Requirements

We strongly believe in an application that offers users with non-visible, yet extremely important features as they perform as the heart and mind of our application.

The non-functional requirements are elaborated as follows:

Table 15: Non-Functional Requirements Types

Requirement	Type
NFR1	Accessibility
NFR2	Data Integrity
NFR3	Effectiveness
NFR4	Platform Compatibility
NFR5	Privacy
NFR6	Security
NFR7	Reusability
NFR8	Transparency

- Accessibility: Accessibility plays a role as one of our major goals to offer users who suffer from various conditions and/or disabilities, where they normally cannot operate such applications, to be able to benefit from our services when needed. And this can be seen within our application as it can operate using voice commands that are powered by Siri for iOS or Google Assistant for Android devices. This will allow blind people and even people who have suffered severe injuries (Say in a car accident for example), to call for help without moving a muscle.

- Data Integrity: Es3af.com operates in cases where accurate information is a critical factor, and any misleading information can lead to potential harm on both ends of the rope.

- Effectiveness: Effectiveness is translated within our application in which the task performed requires the least amount of effort, and for the weight of the task performed, it makes the effort made in having it done be extremely minimal, in which something as simple as a voice command can save a life.

- Platform Compatibility: Es3af.com is made to be a web-based application that can operate on multiple platforms, including but not limited to: Mobile Phones, Tablets, Computers, Car Tablets, etc... And this can come in handy in various situations accordingly.

E-s3af.com

- Privacy: We believe in privacy being a top priority in our applications in which other users' data is not visible to one another, however, it is shown only to those who have a direct concern to dealing with cases. i.e.: Dispatched units, Dispatchers, etc... This requirement is achieved through keeping our databases secure and having users only perform actions on certain webpages, as well as utilizing roles and flags to prevent unwanted access of data, and limiting one's access within the application itself.

- Security: Security is considered the other side of the coin, in addition to privacy, and we believe that it is as important as well. Security shows in the way we have decided to let users log into the application, as they will be using their mobile numbers to do so and to further verify their login, an SMS is sent to their side to verify the integrity of the provided number.

- Reusability: This feature will not be very visible to the users as its main use is in our way of implementing the program. There will be snippets of the code that will be used in multiple places within the application, and this will greatly improve on optimizing the application later on, as well as help us update it more smoothly, as well as assisting us in troubleshooting when necessary.

- Transparency: We lastly believe in offering users a transparent experience in which they know exactly what is happening with every action that is taken within the application. And this shows as users can track the dispatched units, as well as be provided with information on the request they have sent and how it is being handled by the dispatchers, in addition to that, they will be offered full information about medicines they scan, and the closest merchant pharmacy for the corresponding requested medicine. The dispatcher side of this is having to know every single detail about where all the dispatched units are who are on duty, as well as the requests that are being submitted all in real-time. The dispatched units will also have access to the nearest available hospitals, as well as the request they are assigned to.

E-saf.com

3.5 Domain Requirements

Those requirements depend on government regulations and rules by the ministry of health.

Chapter 4
System Design

4.1 Logical Model Design

As a result of analyzing the previously mentioned requirements, we concluded that our system requires us to use the Object-Oriented approach. The Object-Oriented Approach allows us to capture the structure and behavior of our system in a more organized manner, in which our system can be broken down into smaller modules that combine both data and process.

As for other reasons that supported our decision, we can specify multiple such as reusability, as well as the Object-Oriented Design is done concurrently with other phases of our system, and the line between design and implementation being blurry.

Our system offers a variety of benefits that can be concluded in the points below upon choosing the Object-Oriented Design Approach:

- Encapsulation: This can be demonstrated through the hierarchy of our system's entities, as we strongly believe that our system needs to have a maximized level of independence in order to avoid further security-related complications, as well as allow us to properly update one part of our system in case of updating it as a whole.
- Abstraction: Our system provides an easy-to-use interface in which all stakeholders can interact with the system to have the job done with the least amount of effort. Abstraction can be shown in our system as no stakeholder needs to know how our system works in full detail, as long as it is doing the right job. For example, You are a patient who needs an ambulance, you know that the only required step is to tap on the "Request Ambulance" button to have an ambulance sent your way, but in reality, the backend of our system does not show to the user as it is unnecessary to complete the given task.
- Relationships: Our system, as independent as the entities can be individual, we believe that the relationships between entities are as important for an efficient system, as well as maximizing the throughput while maintaining the least amount of runtime.

Upon choosing the Object-Oriented Approach, we would like to present the following diagrams to further explain the project:

E-3af.com

4.1.1 Use Case Diagram:

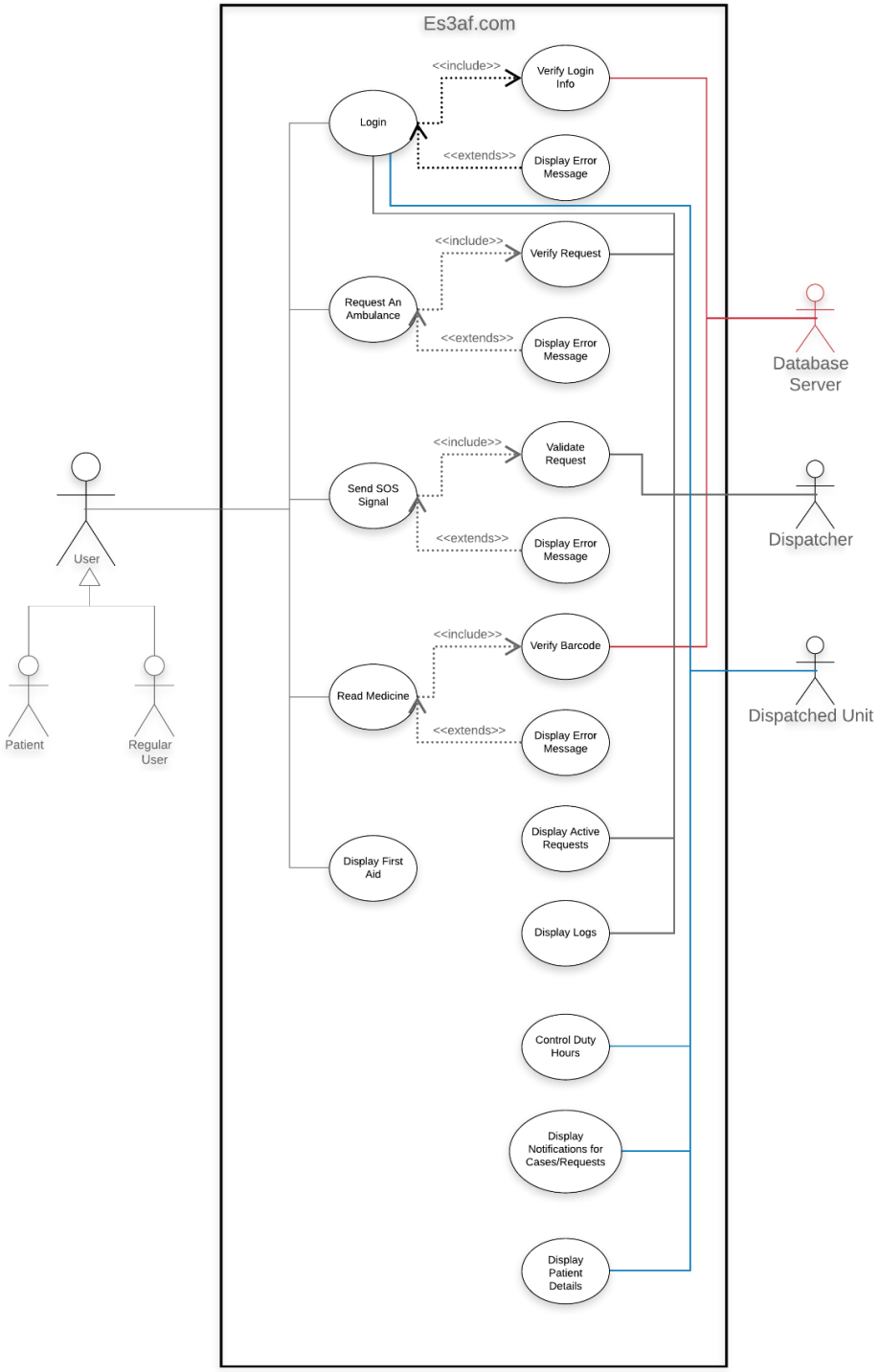


Figure 2: Use Case Diagram for Es3af.com

As mentioned below, we have depicted multiple use cases derived from the above figure as follows:

Use Case: Login
Actors: User/Dispatcher/Dispatched unit, Database server
Purpose: For users of the system to log into the system
Overview (Success scenario):
A user inserts their login information into the system, the database server checks for validity. On completion, the user is allowed to use the system.
Type: Primary
Cross References: FR1
Typical course of actions:

1-User inserts password	2-Application sends the password to the database server for verification
3-database server searches for matching password	
4-database server sends verification to the system along with user information	5-Application displays main menu to the user

Alternative

Line 3-If password is incorrect, the database server sends an error message and system displays it to user

Use Case: Request Ambulance
Actors: Patient, Dispatcher, Dispatched unit
Purpose: For the patient to request an ambulance
Overview (Success scenario):
The patient requests an ambulance, the request is accepted by the dispatcher, the dispatcher assigns a dispatched unit to go to the patient's location. On completion, the patient should get a confirmation message stating that an ambulance is on the way.
Type: Primary
Cross References: FR2
Typical course of actions:

1-Patient requests ambulance	2-Application sends a validation request to the dispatcher

E-S3af.com

3-Dispatcher validates the request and sends patient information to dispatched unit nearest to patient	4-Application sends alert to the dispatched unit along with the patient's location and information as well as a notification to the patient verifying that a unit is arriving
5-dispatched unit goes to the patient location and confirms their arrival	6-Application redirects confirmation to the dispatcher

Alternative

Line 1-If server error occurs, the application displays the error message Line 2-If dispatcher denies the request, the application will display the error message to the user

Use Case: Send SOS Signal

Actors: Patient, Dispatcher, Dispatched unit

Purpose: For the patient to request an emergency vehicle

Overview (Success scenario):

The patient sends a SOS signal, the request is accepted by the dispatcher, the dispatcher assigns a dispatched unit to go to the patient's location. On completion, the patient should get a confirmation message stating that an ambulance is on the way.

Type: Primary

Cross References: FR4

Typical course of actions:

1-Patient sends SOS signal	2-Application sends a validation request to the dispatcher
3-Dispatcher validates the request and sends patient information to dispatched unit nearest to patient	4-Application sends alert to the dispatched unit along with the patient's location and information as well as a notification to the patient verifying that a unit is arriving
5-dispatched unit goes to the patient location and confirms their arrival	6-Application redirects confirmation to the dispatcher

Alternative

Line 1-If server error occurs, the application displays an error message Line 2-If dispatcher denies the request, the application will display an error message to the user

Use Case: Read Medicine

Actors: User, Database Server

Purpose: For the user to view medication information

Overview (Success scenario):

E-s3af.com

The user reads the barcode of the medicine box using his phone camera, the barcode number is sent to the database. On completion, the user should be able to view that medication's information.
Type: Secondary
Cross References: FR3
Typical course of actions:

1-User opens barcode reader and scans medicine barcode	2-system detects barcode and sends barcode number to database server
3-Database server looks up medicine and sends medicine information to the system	4-System displays medicine information to the user

Alternative

Line 2-If database does not find matching barcode number, it sends an error message to the system and system displays it to user

Use Case: Display First-Aid
Actors: User, Database Server
Purpose: For the user to view first-aid tips for a given scenario
Overview (Success scenario):
The user opens the first aid scenarios and chooses the needed scenario. On success the user should see first-aid tips and images describing how to deal with the scenario.
Type: Secondary
Cross References: FR5
Typical course of actions:

1-User opens first-aid tips and selects a needed situation	2-system displays situation with images and animated explanation of how to deal with the situation

E-s3af.com

Use Case: Display Logs
Actors: Dispatcher
Purpose: For the dispatcher to view system logs
Overview (Success scenario):
The dispatcher searches for logs within certain criteria. On success, the system should display the logs to the dispatcher.
Type: Secondary
Cross References: FR8
Typical course of actions:

1-Dispatcher clicks on display logs and inserts search criteria	2-System looks up logs containing the information specified and displays them

Alternative

Line 1- If criteria are not found system displays an error message to the dispatcher

Use Case: Display Active Requests
Actors: Dispatcher
Purpose: For the dispatcher to view active requests in the system
Overview (Success scenario):
The dispatcher looks for active requests through the system. On success, the dispatcher should be able to see every active request in their area.
Type: Secondary
Cross References: FR6
Typical course of actions:

1-Dispatcher clicks on display active requests	2-System looks up unvalidated requests and displays them to a dispatcher

Alternative

Line 1- If not found system displays an error message to the dispatcher

E-s3af.com

4.1.2 Class Diagram:

The below class diagram emphasizes on our take on the hierarchy of our system, as we believe that we should provide the least dependability possible and trade it for higher efficiency, which successfully meets our view on how critical the performance and task flow of the program is and how it should be handled.

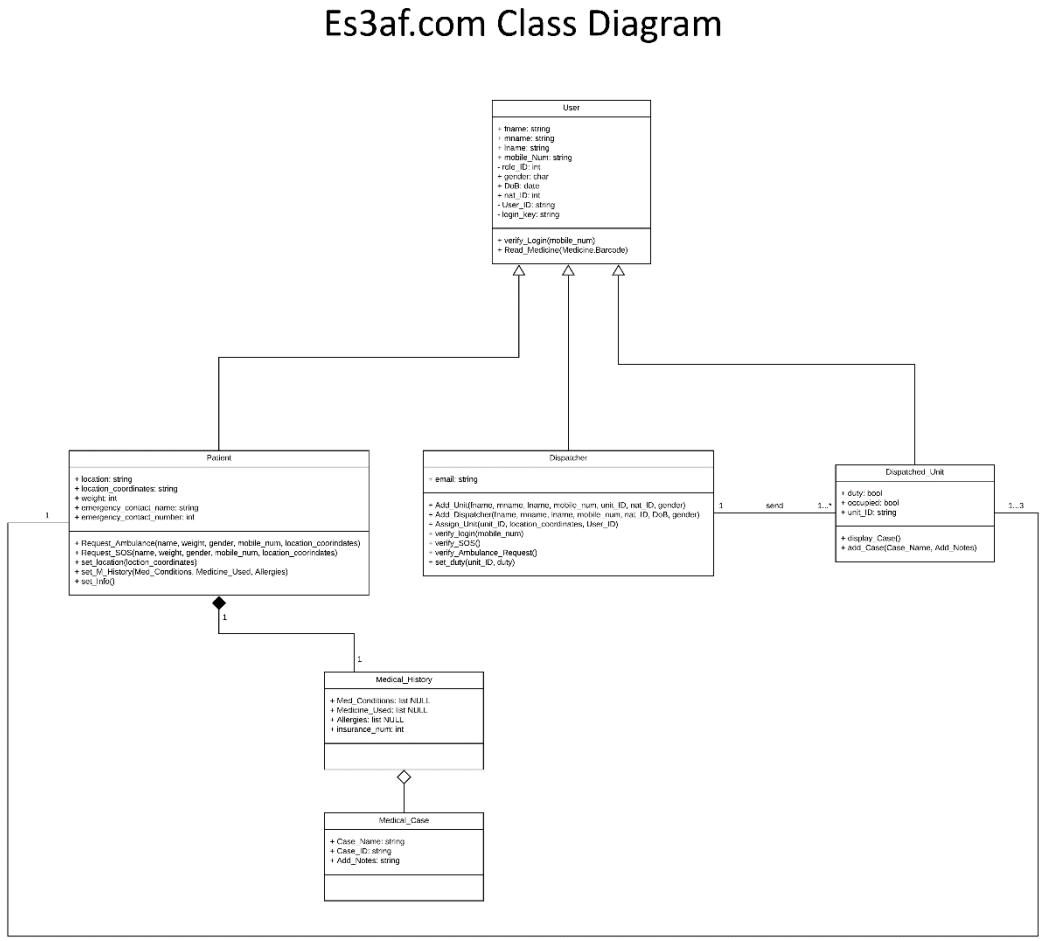


Figure 3: Class Diagram for Es3af.com

4.1.3 Component Diagram:

The below component diagram displays the general idea of how the system components should work, the main class is the User class where (Patient, Dispatcher, and Dispatched Unit) inherit the main attributes of a user then add their attributes to it.

The User Interface is the main Interface for Dispatched Units and Patients; thus, it has an edge that accesses the data in Patient class with the help of the Server being the provider, the Web Dashboard is mainly for Dispatchers usage for monitoring and other work to be done. The Server relies on the provided database to fetch all the information needed to provide it to both User Interface and Web Dashboard.

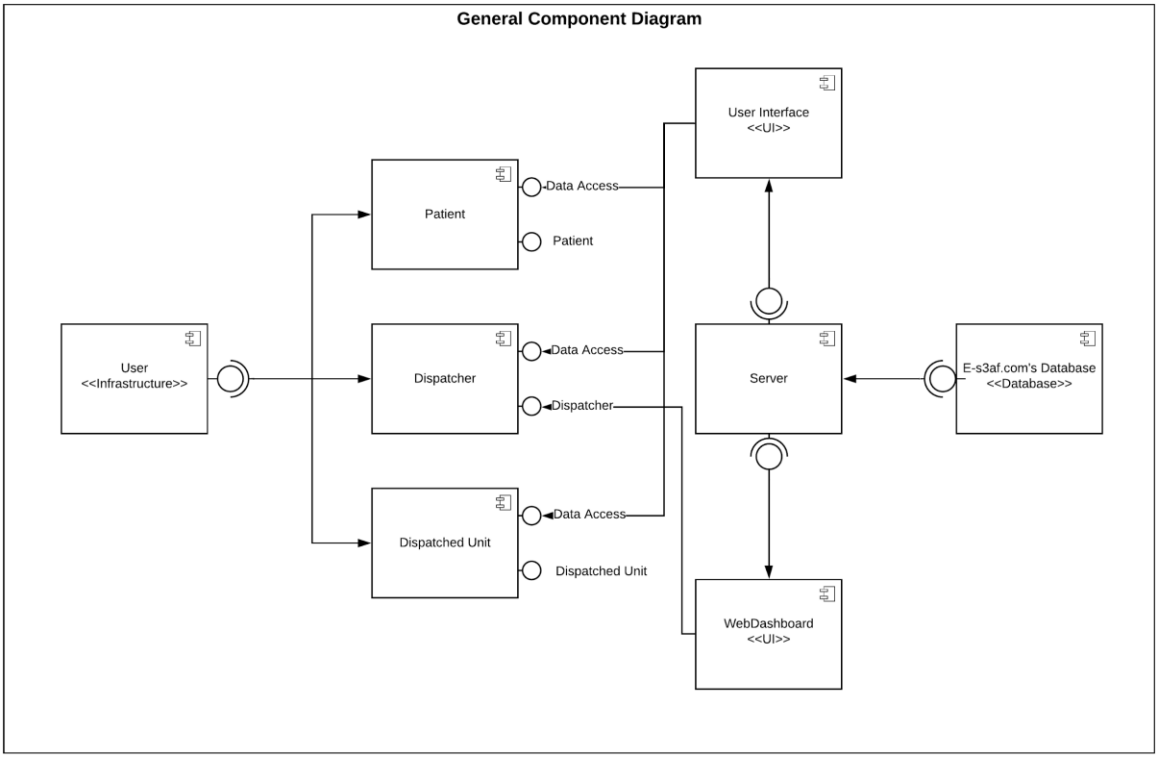


Figure 4: General Component Diagram for Es3af.com

4.1.4 Activity Diagrams:

The below diagram represents the activity present when attempting to request an ambulance and what each "Role" does in making this happen. In a normal flow, the user is logged in, the main menu is displayed and with a simple click on "Request Ambulance", the system processes the request towards the dispatcher, after the dispatcher verifies the source, the system sends a global signal to the closest dispatched units to maintain a faster response time.

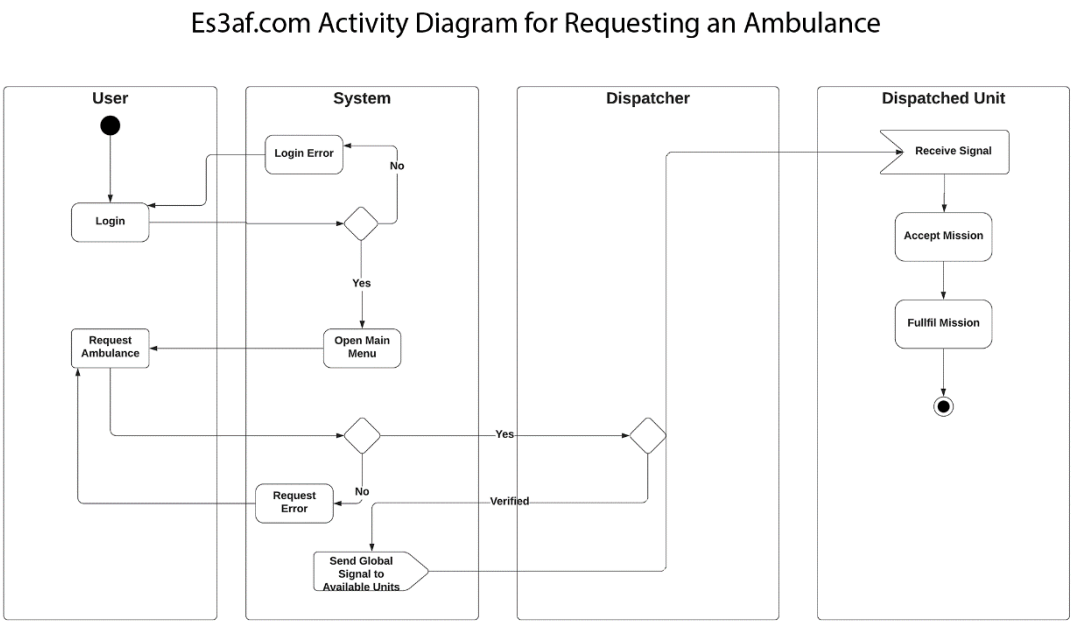


Figure 5: Activity Diagram for Requesting an Ambulance

The below diagram represents the default process of scanning barcodes on medicine packs, the response will be found or not found while remaining on the same tab of scanning barcodes to allow scanning others or simply leaving that tab if you found the information needed.

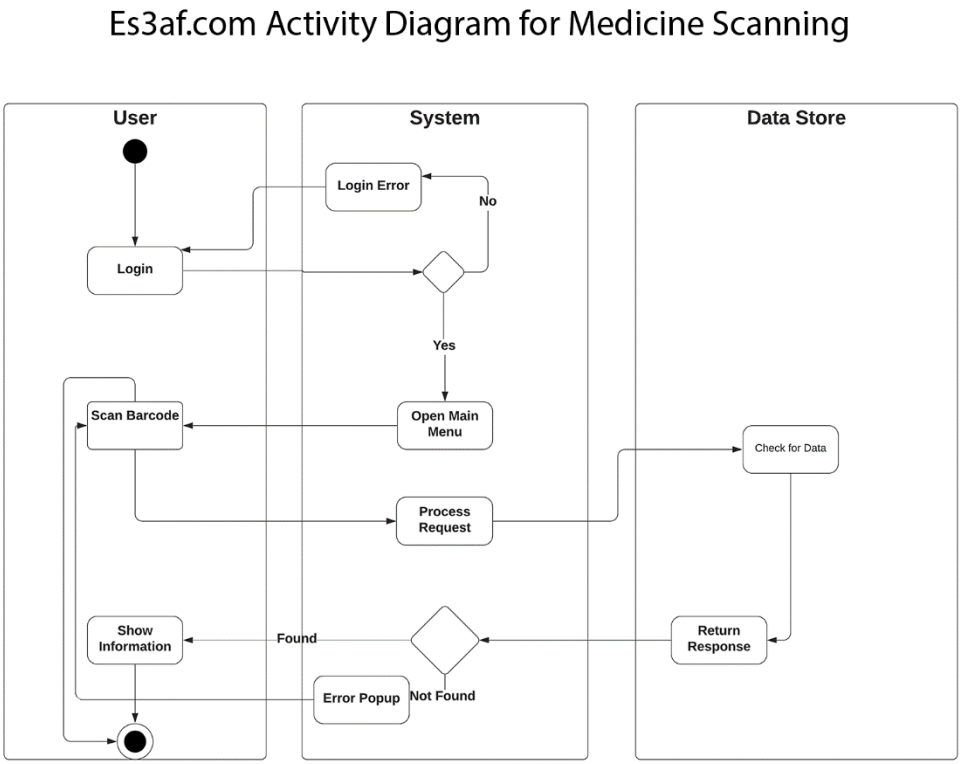
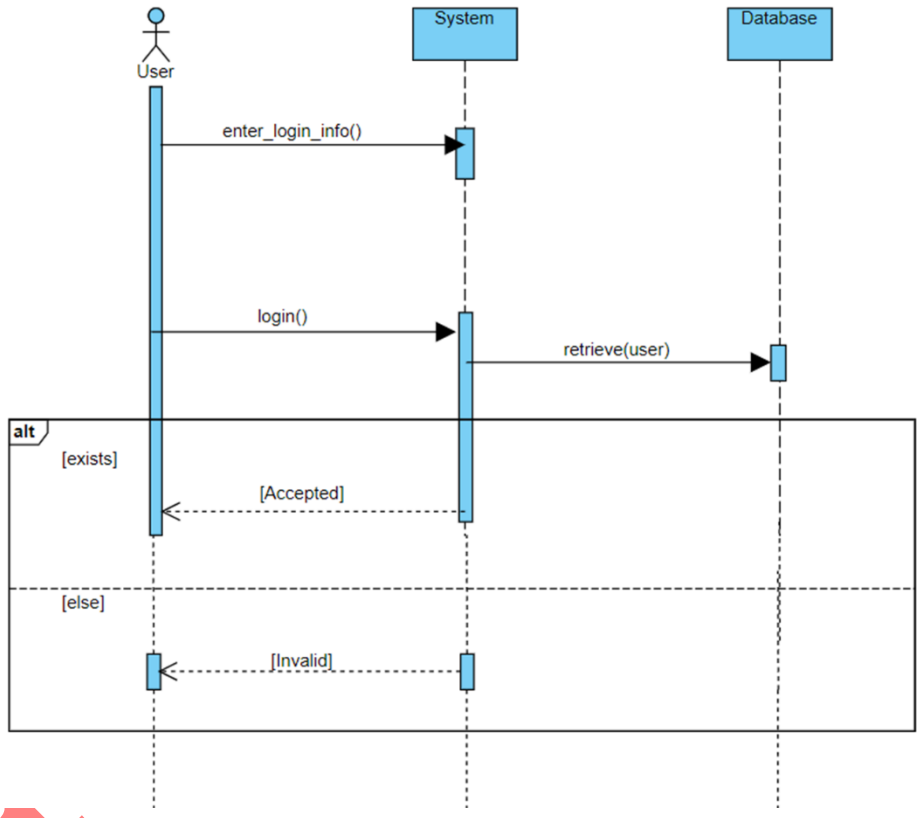


Figure 6: Activity Diagram for Medicine Scanning

4.1.5 Sequence Diagrams:

The below diagram shows the requirement FR1 in the form of a UML sequence diagram where the user enters their login information to the system, then the system checks it through the database, and if it is correct accept their login attempt and if the information is invalid it declines the attempt.



The below diagram shows the requirement FR4 in the form of a UML sequence diagram where the user requests an S.O.S signal, and the request is redirected to the dispatcher where if it accepts it; it will confirm the request to the user and dispatch an emergency vehicle towards the user's location, and if the dispatcher denies the request it will display a declination message to the user.

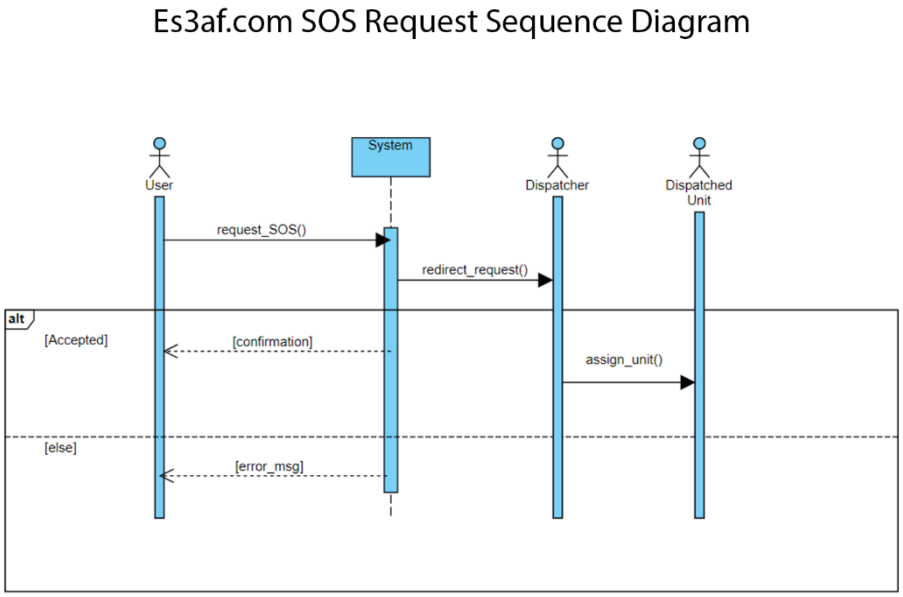


Figure 8: SOS Request Sequence Diagram

E-s3af

4.1.6 State Transition Diagrams:

This state diagram represents the case where you are in a rural area or lost somewhere and in-danger and would like to request some sort of help/emergency to escort you, selecting the S.O.S option from the main menu will trigger a composite state where your SOS Signal is active and an emergency mode is there until help is there.

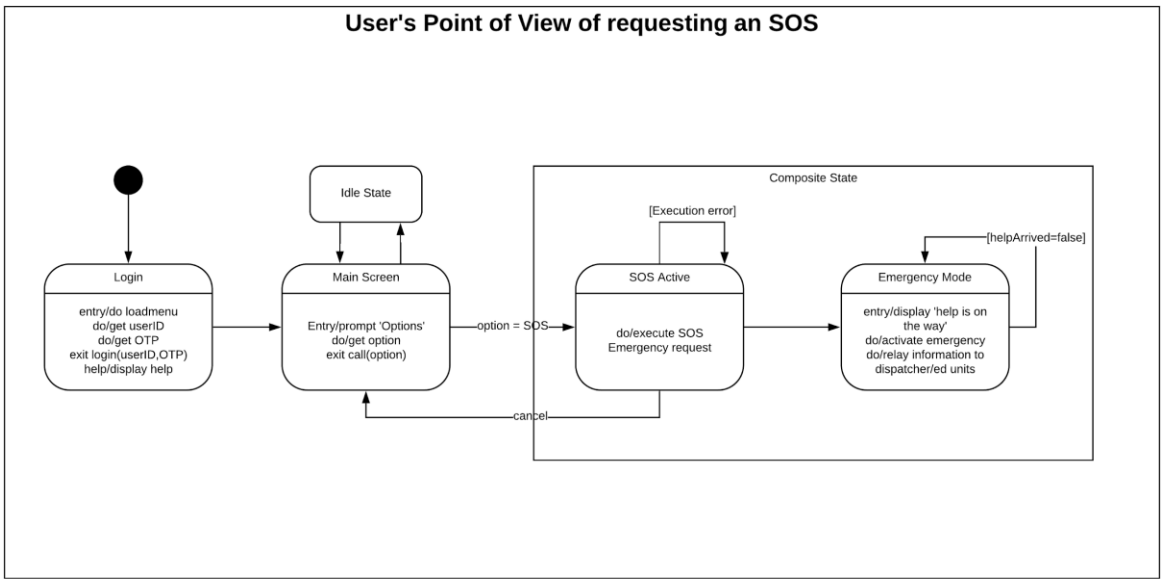
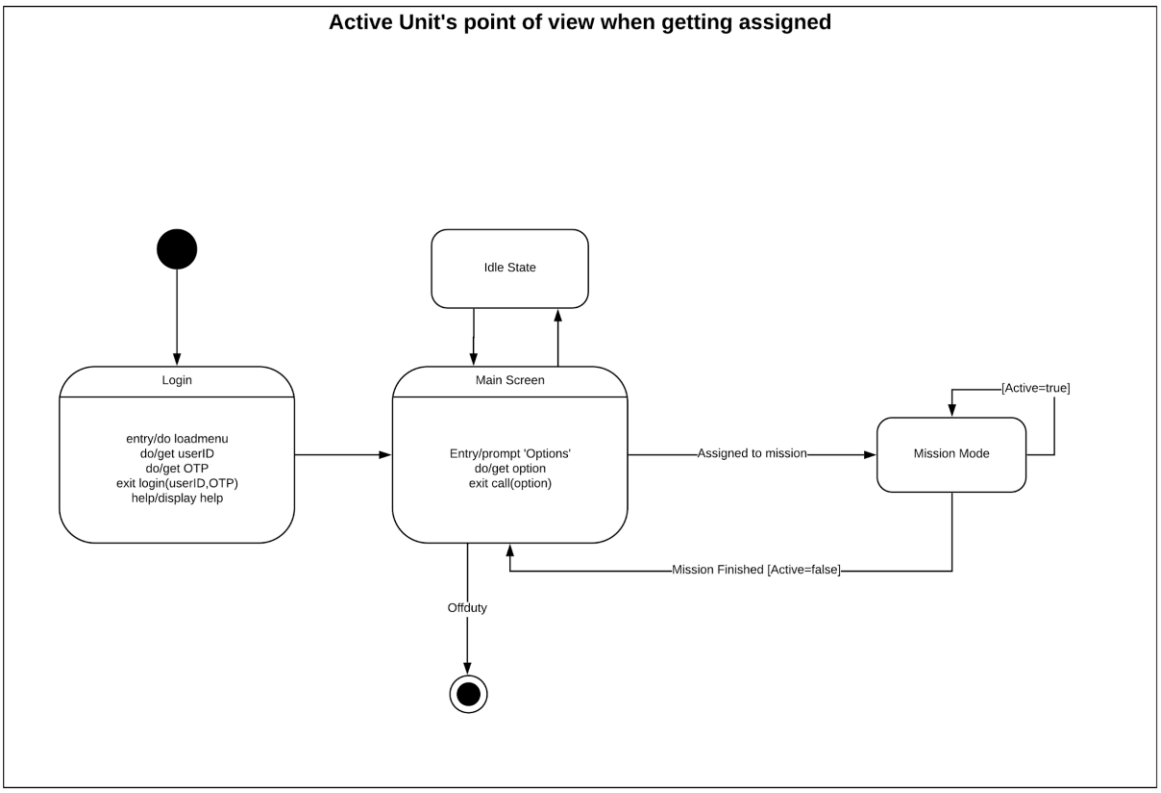


Figure 9: State Transition Diagram for Requesting an S.O.S

E-s3ar

This state diagram is the perspective of an active unit nearby, you have an active listener on the main menu and it remains in an idle state until you're assigned to a specific mission, be it an SOS or Ambulance request, etc., you will be in mission mode state until it is complete.



E-S

The below state diagram is the perspective of a Dispatcher, they are there to maintain the operation and moderate over everything happening and act as some sort of median between Patients, Dispatched Units and System (in some cases). This one portrait the request of an emergency and how it's verified, it also shows a manual assignment functionality to assign a specific unit that is close in radius to enhance the response time. The dispatcher will be able to monitor the information being relayed between user and dispatched unit as well as relay additional information to both sides to provide help.

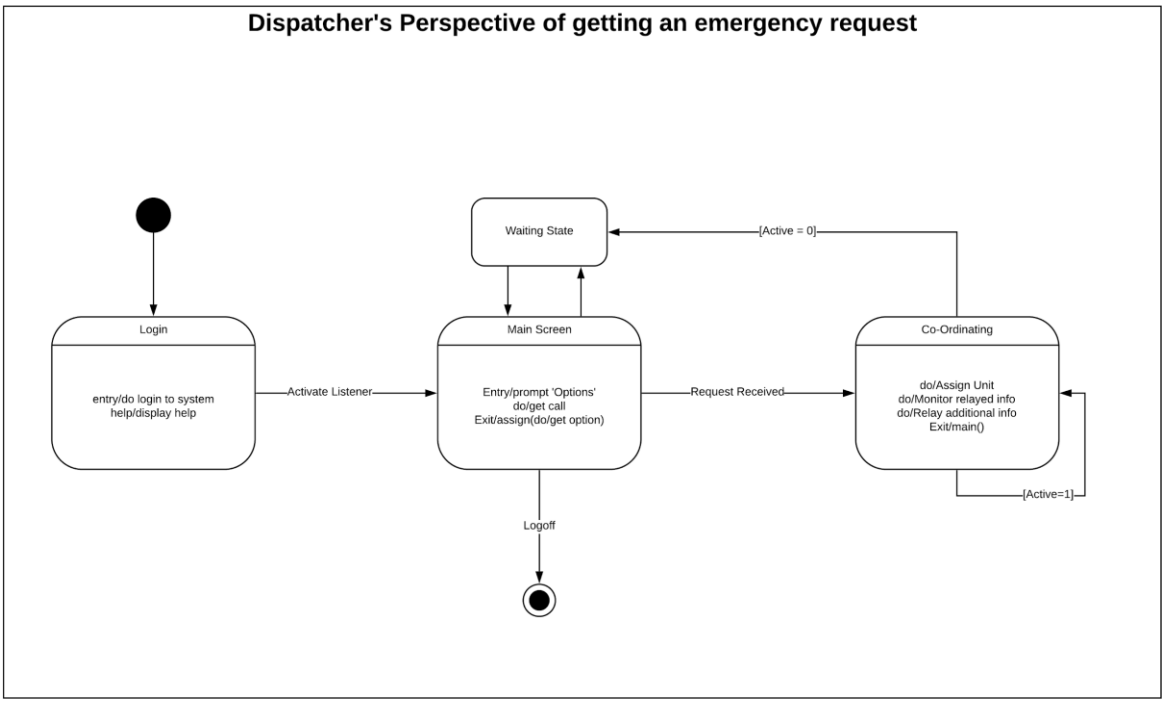


Figure 11: State Transition Diagram for Dispatcher Receiving an Emergency Request

E-S

4.1.7 Communication Diagrams:

The below communication diagram shows the requirement FR1 in the form of a UML communication diagram where the user interacts with the UI and sends the login information through it, then it will check if the login information is valid from the user login database, if it is valid it will display the appropriate UI depending on the user type taken from the E-S3af Database, and if the information is invalid it will display a login error message to the user through the UI.

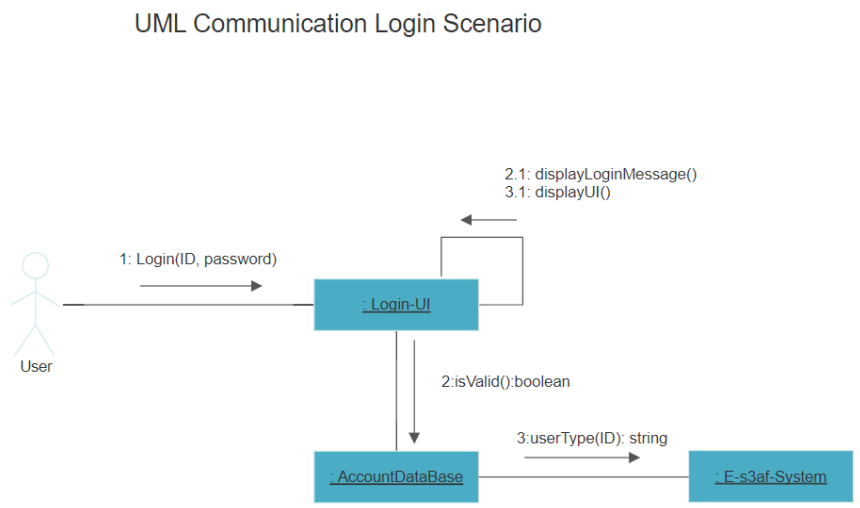
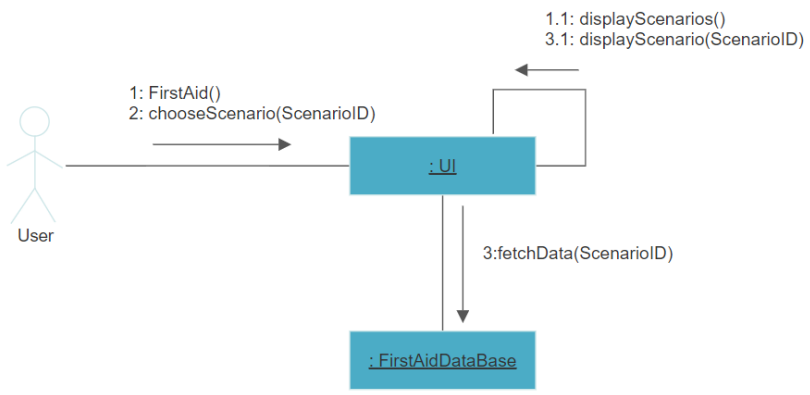


Figure 12: Login Communication Diagram

The below Communication Diagram shows the requirement FR5 in the form of a UML communication diagram where the user interacts with the UI and selects the first-aid option, then the UI would display the scenarios available, after that the user chooses the scenario needed, then the UI would fetch the scenario data from the first-aid database and display it to the user.

Communication Diagram: First-Aid Scenario



E-s3a

4.1.8 Database Schema:

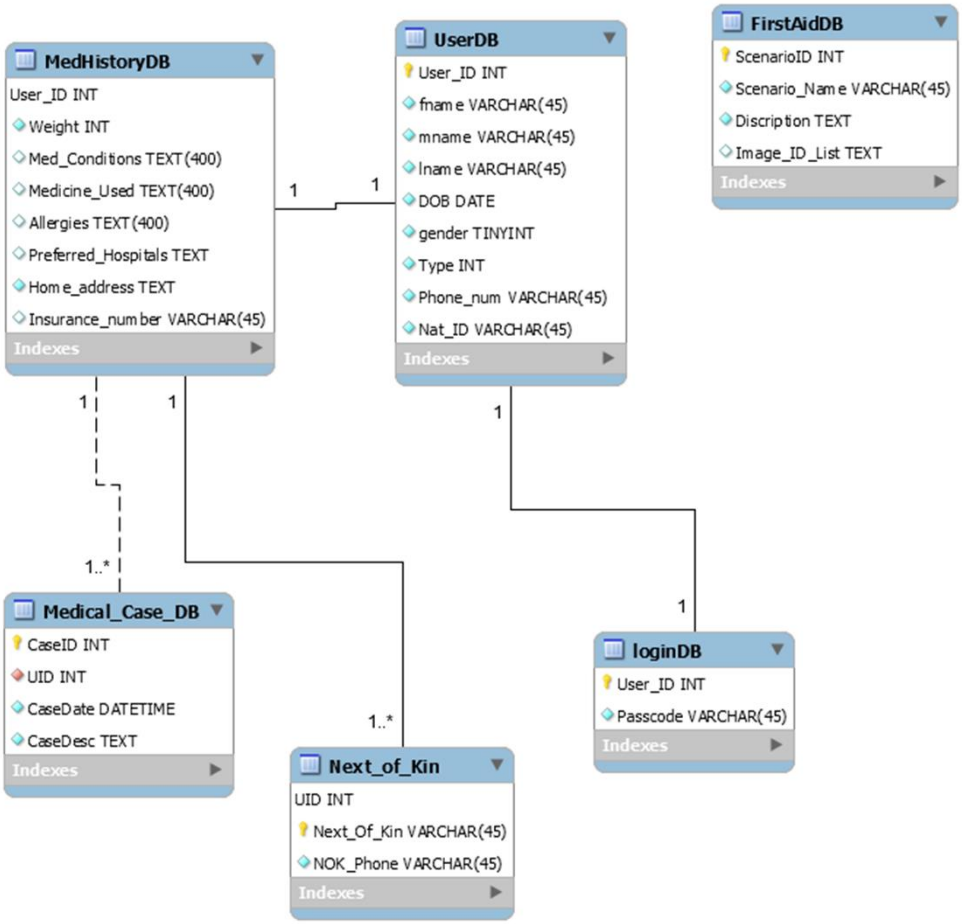


Figure 14: Enhanced Entity Relation Ship Diagram for Es3af.com

E-S3af Database Entity Relationship Diagram

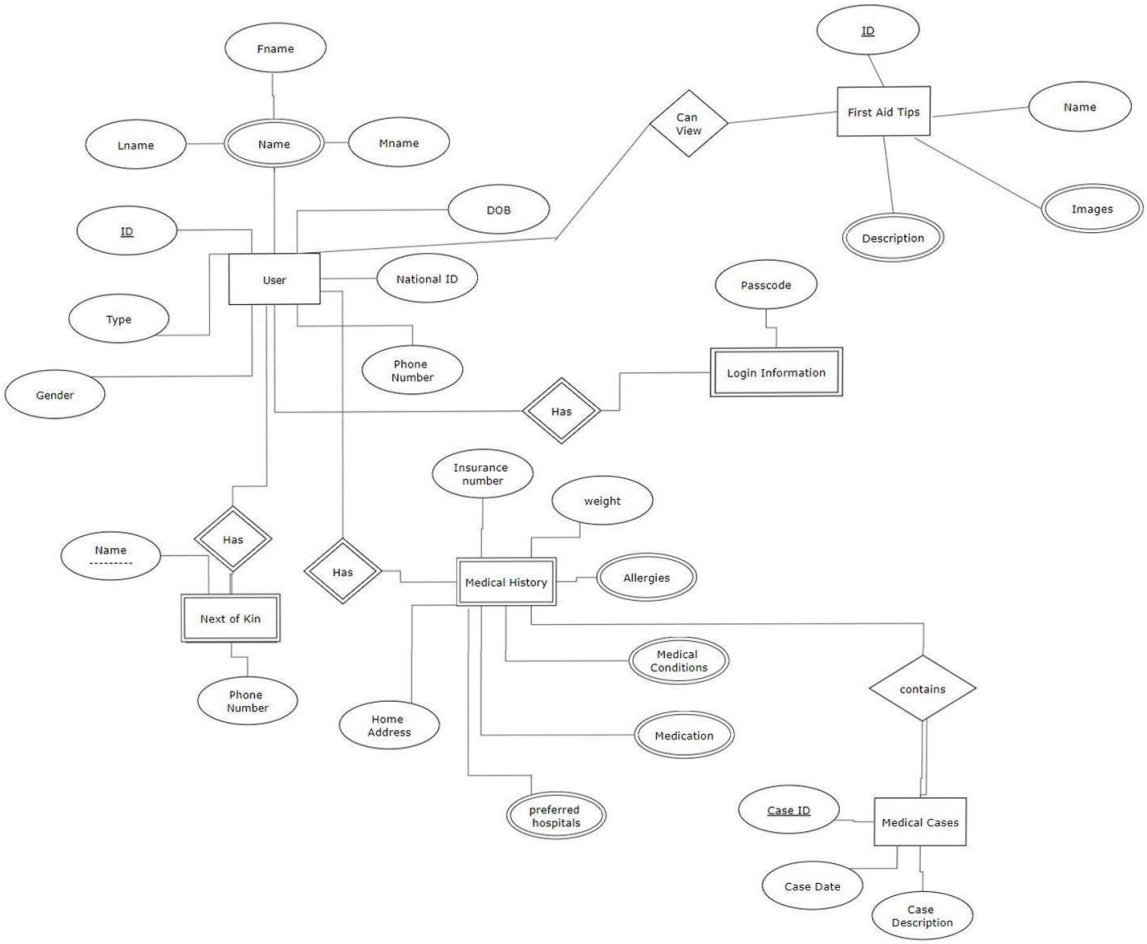


Figure 15: Entity Relationship Diagram for Es3af.com

Describe also any design choices that are related to the user interface. Describe the different screens (or web-pages), and how the flow moves between them. Use appropriate diagrams like trees to describe the structure of web-pages (for example).

Make sure to organize this chapter into sections and subsections in a manner that is appropriate to the provided information. Make sure also to stick to the formatting used in the other chapters.

4.2 Physical Model Design

4.2.1 Reports Design

Userbase report

Table 16: Userbase System-Generated Report

User ID	First Name	Last Name	Date of Birth	Gender	Type	Phone Number	National ID

Patient Report

Table 17: Patient System-Generated Report

User_ID	Weight	Medical Conditions	Medicine Used	Allergies	Preferred Hospitals	Home Address	Insurance Number

Ambulance Report

Table 18: Ambulance System-Generated Report

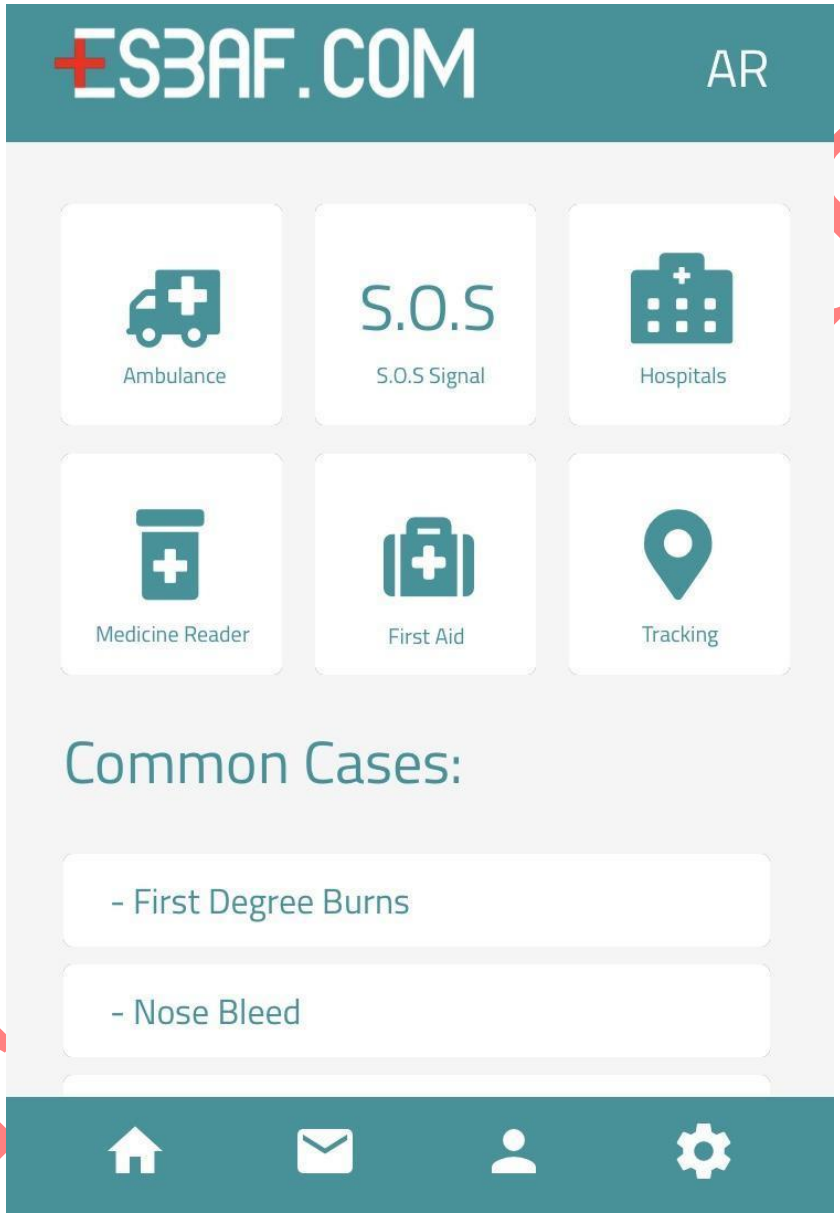
Dispatcher ID	Dispatched Unit ID	Patient ID	Patient Name	Date	Response Time	Time Taken

E-s3af.com

4.2.2 User Interface Design

The following UI design is in its preliminary state and is subject to change with the upcoming work as seen best fit for the user experience.

4.2.2.1 Patient Interface:



4.2.2.2 Patient Interface:

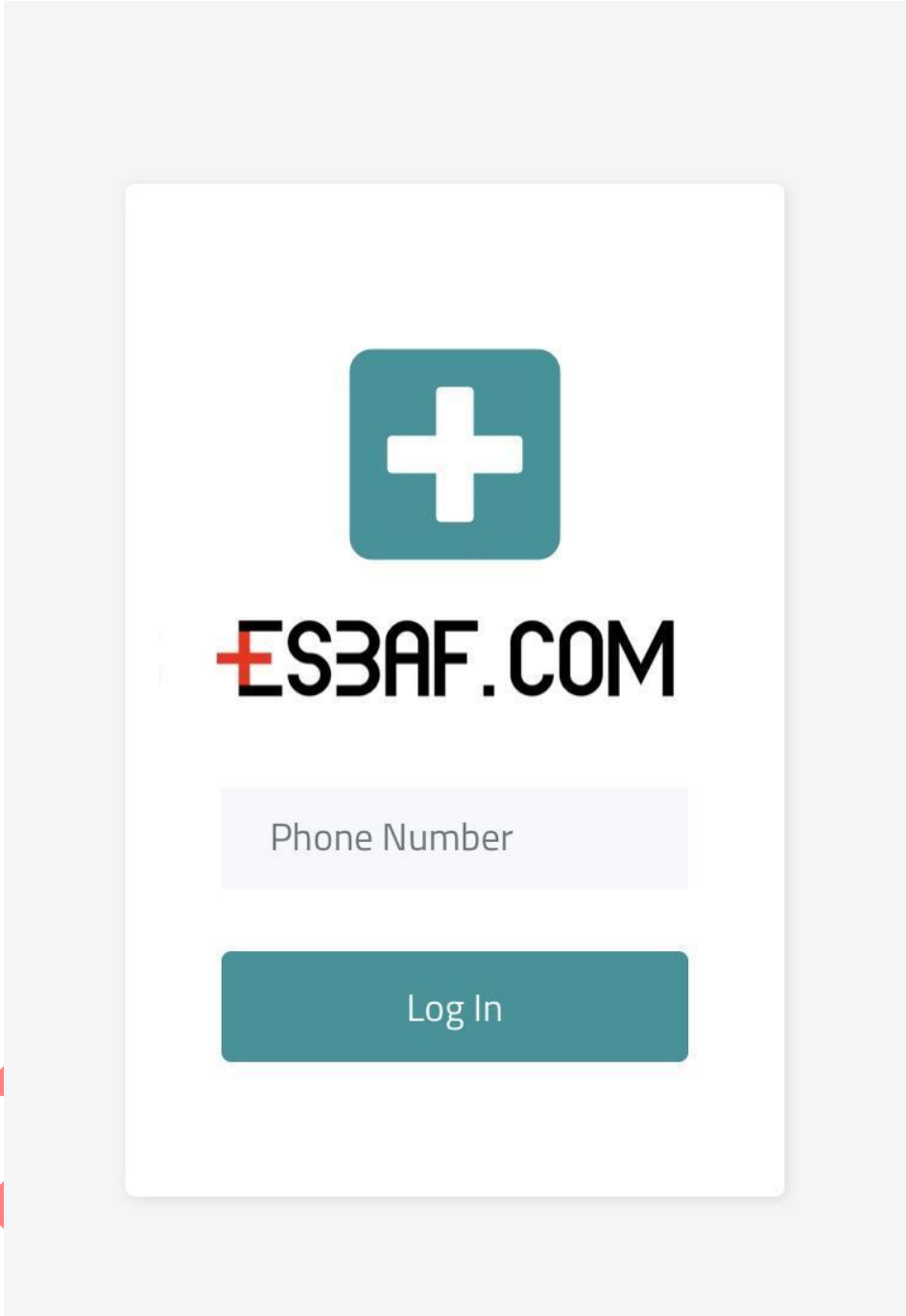


Figure 17: Patient Login UI

4.2.2.3 Dispatched Unit Interface:

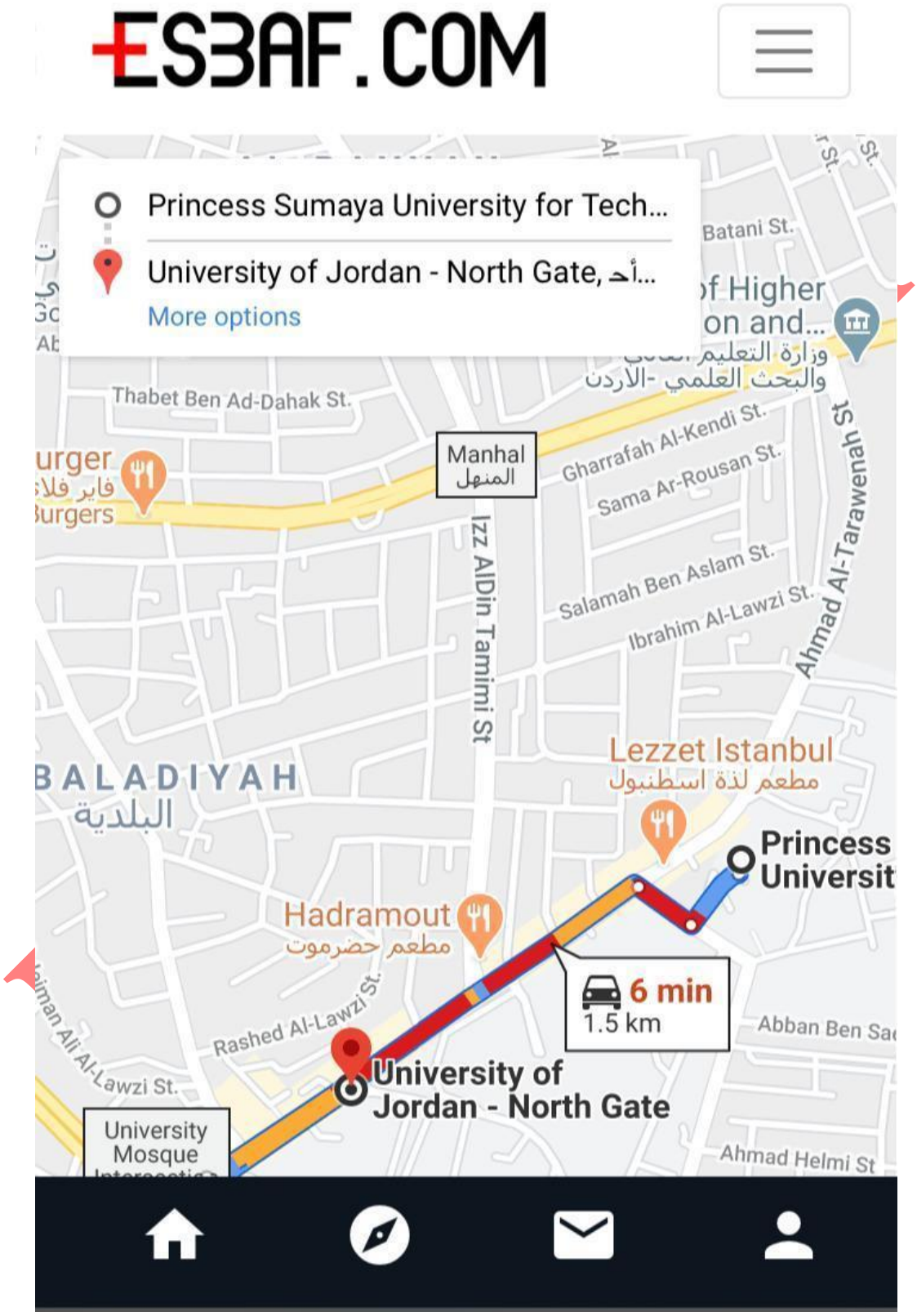


Figure 18: Dispatched Unit UI After Being Assigned to a Request

4.2.2.4 Dispatcher Interface:

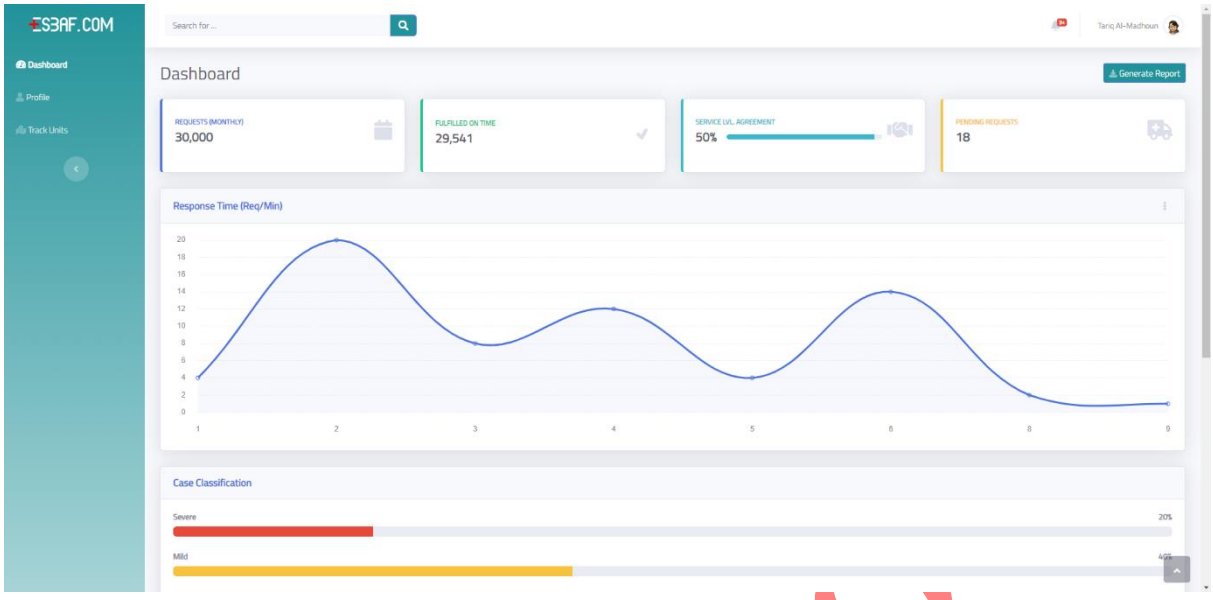


Figure 19: Dispatcher UI Main Dashboard

4.2.3 Database Design:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
User_ID	INT	☑	☑	☐	☐	☐	☐	☑	☐	
fname	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
mname	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
lname	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
DOB	DATE	☐	☑	☐	☐	☐	☐	☐	☐	
gender	TINYINT	☐	☑	☐	☐	☐	☐	☐	☐	
Type	INT	☐	☑	☐	☐	☐	☐	☐	☐	
Phone_num	VARCHAR(45)	☐	☑	☑	☐	☐	☐	☐	☐	
Nat_ID	VARCHAR(45)	☐	☑	☑	☐	☐	☐	☐	☐	

Figure 20: User Table

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
UID	INT	☑	☑	☐	☐	☐	☐	☐	☐	
Next_Of_Kin	VARCHAR(45)	☑	☑	☐	☐	☐	☐	☐	☐	
NOK_Phone	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

Figure 21: Next of Kin Table

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
CaseID	INT	☑	☑	☐	☐	☐	☐	☑	☐	
UID	INT	☐	☑	☐	☐	☐	☐	☐	☐	
CaseDate	DATETIME	☐	☑	☐	☐	☐	☐	☐	☐	
CaseDesc	TEXT	☐	☑	☐	☐	☐	☐	☐	☐	
		☐	☐	☐	☐	☐	☐	☐	☐	

Figure 22: Medical Case Table

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
User_ID	INT	☑	☑	☐	☐	☐	☐	☐	☐	
Weight	INT	☐	☑	☐	☐	☐	☐	☐	☐	
Med_Conditions	TEXT(400)	☐	☐	☐	☐	☐	☐	☐	☐	
Medicine_Used	TEXT(400)	☐	☐	☐	☐	☐	☐	☐	☐	
Allergies	TEXT(400)	☐	☐	☐	☐	☐	☐	☐	☐	
Preferred_Hospitals	TEXT	☐	☐	☐	☐	☐	☐	☐	☐	
Home_address	TEXT	☐	☑	☐	☐	☐	☐	☐	☐	
Insurance_number	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	

Figure 23: Medical History Table

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ScenarioID	INT	☑	☑	☐	☐	☐	☐	☑	☐	
Scenario_Name	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	
Discription	TEXT	☐	☑	☐	☐	☐	☐	☐	☐	
Image_ID_List	TEXT	☐	☐	☐	☐	☐	☐	☐	☐	

Figure 24: First Aid Table

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
User_ID	INT	☑	☑	☑	☐	☐	☐	☐	☐	
Passcode	VARCHAR(45)	☐	☑	☐	☐	☐	☐	☐	☐	

Figure 25: Login Table

E-s3af.com

Chapter 5
Implementation

Programming Languages, tools and APIs

Mobile users have been leading in traffic against desktop users which put this application preference towards being mobile, therefore we picked flutter as our Software Development Kit as it allows the porting of our application towards both Android OS and iOS.

"Flutter is Google's UI toolkit for building beautiful, natively compiled applications for mobile, web, and desktop from a single codebase." [1]

Flutter certainly has interesting interface functionalities that have a very friendly user interface which can be exported towards any mobile platform easily. For this project, we have tested out the capabilities of flutter web and flutter on android, reasons being Android is Open-Source which allows us to have our application tested easily when using an Emulator or a physical device with USB-Debugging.

As for the development using Flutter, Visual Studio Code was used to run Flutters SDK and the development and code was written using it. Visual Studio Code is lightweight compared to regular Visual Studio and allows for faster access to emulators and compiling as you are not obligated to download other SDKs for languages that you don't need to use.

The API and Back-end was based on Firebase which is also a Google's service that helps you develop mobile applications offering you a lot of services such as authentication, file storage, databases, analytics and many other useful features. Therefore, it saves the effort of having servers that host your API as it has the backend in its hands. We mainly used it to store the user base and other vital information using Firebase's Firestore.

Node.js was also used to host a server that forwards notifications to the user base, be it a dispatched unit or a patient. Node.js is quite reliable in handling requests sent to it and making API Requests to Firebase.

E-Saf.com

Overview

The following table lists the number of implemented and imported classes used and the total number of lines of code found in our project:

Class Name	Class Description
services	
material	
firebase_auth	Firebase Authentication Services
cloud_firestore	Firebase Cloud Storage
geolocator	Accurate Location/Position finder
google_maps_flutter	Google Maps API on Flutter
age	Accurate Age calculation
Firebase_messaging	Notifications Handler
Firebase_core	Core service for firebase
url_launcher	Handles launching MAP URLs
location	Provides location based on position
message_handler	Handles received mission notifications
Management	Handles main application functionalities
UserMng	Handles user data retrieval and such
FLogin	Mobile SMS Login system

E-s3af.com

Login	Dashboard Email/Password Login system
Dash	The dashboard page and its functionalities
Track	The tracking page and its functionalities
Profile	Profile page display and user data retrieval
AddDu	Handles Adding Dispatched Units
Patient	Handles fetched patient data and sends a notification to targeted Dispatched Unit
PatientNotif	Handles the notification to be sent to the patient
Translations	Handles localization of the application
Total Lines of Code	**14222**
Total Number of Classes	**22**

E-s3af.com

Main Functionalities

Normal User

Sign-In using phone number

```
signIn() async {
  try {
    final AuthCredential credential = PhoneAuthProvider.credential(
      verificationId: verificationId,
      smsCode: smsOTP,
    );
    final User user = (await _auth.signInWithCredential(credential)).user;
    final User currentUser = _auth.currentUser;
    /*
    DocumentReference account = FirebaseFirestore.instance
      .collection('users')
      .doc(_auth.currentUser.uid);
      */
    await FirebaseFirestore.instance
        .doc("users/${currentUser.uid}")
        .get()
        .then((doc) {
      if (!doc.exists) {
        //add navigatorpush if u want etc idk
        Navigator.of(context).pop();
        showAlertDialogReg(context, user.uid);
        UserMng.addUser();
        print("created new acc lol");
      }
    });

    assert(user.uid == currentUser.uid);
  } catch (e) {
    handleError(e);
  }
}
```

Request Ambulance / Distress Signal

```
//  This code checks if the user has an active request first then proceeds accordingly
  static void requestHelp(
      Position position, bool input, BuildContext context) async {
    print("Current location:" + position.toString());
    FirebaseAuth _auth = FirebaseAuth.instance;

    DocumentReference account = FirebaseFirestore.instance
        .collection('users')
        .doc(_auth.currentUser.uid);

    if (await hasActiveRequest(_auth)) {
      showDialog(
        context: context,
        builder: (context) => AlertDialog(
          content: ListTile(
            title: Text("Ambulance Already Requested"),
            subtitle: Text("Please wait until you are contacted"),
          ),
          actions: <Widget>[
            FlatButton(
              child: Text('Ok'),
              onPressed: () => Navigator.of(context).pop(),
            ),
            FlatButton(
              child: Text('First Aid'),
              onPressed: () => Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) => HomeScreen(),
                ),
              ),
            ),
          ],
        ),
      );
```

```
      print("Error, user already has an existing case.");
    } else {
      DocumentReference activeReq = FirebaseFirestore.instance
          .collection('active-cases/')
          .doc(_auth.currentUser.uid);
      //if Type = true it means ambulance if false SOS
      activeReq
          .set({
            'Name': _auth.currentUser.displayName,
            'Phone': _auth.currentUser.phoneNumber,
            'longitude': position.longitude,
            'latitude': position.latitude,
            'location': GeoPoint(position.latitude, position.longitude),
            'Type': input,
            'AssignedUnit': "0"
          })
          .then((value) => print("case created"))
          .catchError((error) => print("Error creating case: $error"));
      account
          .update({
            'hasActiveRequest': true,
            'longitude': position.longitude,
            'latitude': position.latitude,
            'location': GeoPoint(position.latitude, position.longitude)
          })
          .then((value) => print("user has active case = 1 now"))
          .catchError((error) => print("$error"));
    }
  }
```

Medicine Scanner

```
//The following code is to scan the medicine then another function is
used to look it up
```

```
Future<void> scanbc() async {
    print("camera started");
    String barcodeScanRes = await FlutterBarcodeScanner.scanBarcode(
        '#FFFFFF', "Exit", true, ScanMode.BARCODE);
    Navigator.push(
      context,
      MaterialPageRoute(
          builder: (context) => Splashscreen4(
                barcode: barcodeScanRes,
              )),
    );
  }
```

E-s3af.c

First-Aid Tips

```
//This code is the builder that retrieves first-aid tips which are hard-
coded for offline availability

Widget build(BuildContext context) {

    Size size = MediaQuery.of(context).size;

    return SingleChildScrollView(

      child: Column(

        children: <Widget>[

          Header(

            size: size,

            name: Translations.of(context).text('firstaid'),

          ),

       Titlewithunderline(text:Translations.of(context).text('choke')),

          TipsScroll(),

       Titlewithunderline(text:Translations.of(context).text('burns')),

          TipsScroll(),

        Titlewithunderline(text: Translations.of(context).text('cuts')),

          TipsScroll3(),

        ],

      ),

    );

  }
```

E-S

Dispatched Unit

Get Current Request Information

```
static Future<void> getReqInfo(FirebaseAuth _auth, String uid, code)
async {
    var payload;
    if (code == 1) {
      DocumentReference userInfo =
          FirebaseFirestore.instance.collection('users').doc(uid);
      Future<DocumentSnapshot> userI = userInfo.get();
      await userI.then((DocumentSnapshot docSnapshot) => {
            if (docSnapshot.exists)
              {
                payload = docSnapshot.data(),
              }
            else
              {print("error, data does not exist")}
          });
    } else if (code == 2) {
      DocumentReference userMedInfo =
          FirebaseFirestore.instance.collection('med-
history').doc(uid);
      Future<DocumentSnapshot> userMedI = userMedInfo.get();
      await userMedI.then((DocumentSnapshot docSnapshot) => {
            if (docSnapshot.exists)
              {payload = docSnapshot.data()}
            else
              {print("error, data does not exist")}
          });
    } else {
      payload = "Wrong Code Error";
    }
    return payload;
  }
```

Update Case Information

```
  //Upon case completion, necessary cloud storage updates are processed
static Future<void> updateCase(FirebaseAuth _auth, String uid,
      String description, String timeReq, GeoPoint loc) {
    CollectionReference caseH = FirebaseFirestore.instance
        .collection('case-history/' + uid + '/history');
    CollectionReference users =
        FirebaseFirestore.instance.collection('case-history');
    DocumentReference init = users.doc(uid);
    init
        .update({'cases': FieldValue.increment(1)})
        .then((value) => print("cases incremented"))
        .catchError((error) => print("Failed to increment: $error"));

    var now = DateTime.now().millisecondsSinceEpoch;
    DocumentReference dName = caseH.doc("case-" + timeReq.toString());
    DocumentReference account =
        FirebaseFirestore.instance.collection('users').doc(uid);
    account
        .update({
          'hasActiveRequest': false,
        })
        .then((value) => print("user has active case = 1 now"))
        .catchError((error) => print("$error"));
    return dName
        .set({
          'Status':     "Completed",     'Description':     description,
'RequestedTime': timeReq,'FulfiledTime': now, 'Location': loc
        })
        .then((value) => print("case added"))
        .catchError((error) => print("Failed to case: $error"));
    }
```

Duty Hour Control

```
Future<bool> setStatus(FirebaseAuth _auth) async {
    try {
      DocumentReference account = FirebaseFirestore.instance
          .collection('active-DU')
          .doc(_auth.currentUser.uid);
      await FirebaseFirestore.instance
          .doc("active-DU/${_auth.currentUser.uid}")
          .get()
          .then((doc) {
        if (!doc.exists) {
         print("User is not a dispatched unit or doc does not
exist.");
        } else {
          if (doc.get('isAvailable') == true) {
            account.update({'isAvailable': false});
            print("User is now offline");
            return false;
          } else if (doc.get('ActiveCase') == '0') {
            account.update({'isAvailable': true});
            print("User is now online");
            return true;
          }}});
    } catch (e) {print("Error: " + e.toString());}
    //reaching here means something went wrong with this and the
user is offline by default to prevent issues
    return false;
  }
```

Dispatcher

Sign-In using email and password

```
Future<dynamic> signIn() async {
  dynamic result = await _auth.signInWithEmailAndPassword(
      email: _emailController.text, password: _passwordController.text);
  return result;
}
```

```
try {
  if (signIn() != null) {
    Navigator.of(context).push(
        MaterialPageRoute(
            builder: (context) => Body())); // MaterialPageRoute
  }
} catch (e) {
  _emailController.clear();
  _passwordController.clear();
  showDialog(
      context: context,
      builder: (context) {
        return AlertDialog(
          title: Text("Login Error"),
          content: Text(
              "Wrong email/password. Please contact your system administrator."), // Text
          actions: <Widget>[
            FlatButton(
              child: Text("OK"),
              onPressed: () {
                Navigator.pop(context);
              },
            ) // FlatButton
          ], // <Widget>[]
        ); // AlertDialog
      });
}
```

View Active Markers and Related Details with Live-Tracking

```
void activeUserManagement() async {
  String namePlaceHolder;
  String phoneNumberPlaceHolder;
  Marker markerPlaceHolder;

  CollectionReference dispatchedUnitCollection =
      FirebaseFirestore.instance.collection('active-DU');

  CollectionReference activeCasesCollection =
      FirebaseFirestore.instance.collection('active-cases');

  Future<QuerySnapshot> activeCasesQuerySnapshot =
      activeCasesCollection.get();

  Future<QuerySnapshot> activeDispatchedUnit =
      dispatchedUnitCollection.where('isAvailable', isEqualTo: true).get();

  Future<QuerySnapshot> nonActiveDispatchedUnit =
      dispatchedUnitCollection.where('isAvailable', isEqualTo: false).get();

  await activeCasesQuerySnapshot.then((querySnapshot) {
    querySnapshot.docs.forEach((element) {
      if (element.data()['AssignedUnit'] == "0") {
        namePlaceHolder = element.data()['Name'];
        phoneNumberPlaceHolder = element.data()['Phone'];
        markerPlaceHolder = Marker(
            markerId: MarkerId(element.id),
            position: LatLng(
                element.data()['latitude'], element.data()['longitude']), // LatLng
            icon: pinLocationIconPatient,
            infoWindow: InfoWindow(
                title: "Case " + (requestNumber + 1).toString(),
                snippet: namePlaceHolder)); // InfoWindow // Marker

        if (!patientMarkersMap.containsKey(element.id)) {
          setState(() {
            patientMarkersMap[element.id] = markerPlaceHolder;
            patientNameMap[element.id] = namePlaceHolder;
            patientPhoneNumberMap[element.id] = phoneNumberPlaceHolder;
            if (element.data()['Type']) {
              sosColor[element.id] = Colors.red;
            } else {
              sosColor[element.id] = Colors.white;
            }
          });
        }
      }
    });
  }).catchError((onError) {
    print(onError);
  });
```

```
  await activeDispatchedUnit.then((querySnapshot) {
    querySnapshot.docs.forEach((element) {
      if (element.data()['ActiveCase'] == "0") {
        namePlaceHolder = element.data()['name'];
        phoneNumberPlaceHolder = element.data()['phone'].toString();
        markerPlaceHolder = Marker(
            markerId: MarkerId(element.id),
            position: LatLng(
              double.parse(element.data()['Latitude']),
              double.parse(element.data()['Longitude']),
            ), // LatLng
            icon: pinLocationIconAmbulance,
            infoWindow: InfoWindow(
                title: namePlaceHolder, snippet: phoneNumberPlaceHolder)); // InfoWindow // Marker

        setState(() {
          dispatchedUnitMarkerMap[element.id] = markerPlaceHolder;
          dispatchedUnitNameMap[element.id] = namePlaceHolder;
          dispatchedUnitPhoneNumberMap[element.id] = phoneNumberPlaceHolder;
        });
      }
    });
  }).catchError((onError) {
    print(onError);
  });
  await nonActiveDispatchedUnit.then((querySnapshot) {
    querySnapshot.docs.forEach((element) {
      namePlaceHolder = element.data()['name'];
      phoneNumberPlaceHolder = element.data()['phone'].toString();
      markerPlaceHolder = Marker(
          markerId: MarkerId(element.id),
          position: LatLng(
            double.parse(element.data()['Latitude']),
            double.parse(element.data()['Longitude']),
          ), // LatLng
          icon: pinLocationIconAmbulance,
          infoWindow: InfoWindow(
              title: namePlaceHolder, snippet: phoneNumberPlaceHolder)); // InfoWindow // Marker

      setState(() {
        dispatchedUnitMarkerMap.remove(element.id);
        dispatchedUnitNameMap.remove(element.id);
        dispatchedUnitPhoneNumberMap.remove(element.id);
        dispatchedUnitEntryMap.remove(element.id);
      });
    });
  }).catchError((onError) {
    print(onError);
  });
}
```

```
activeCaseListener() {
  String namePlaceHolder;
  String phoneNumberPlaceHolder;
  Marker markerPlaceHolder;

  CollectionReference activeCasesCollection =
      FirebaseFirestore.instance.collection('active-cases');

  Stream<QuerySnapshot> activeCasesStream = activeCasesCollection.snapshots();

  activeCasesStream.listen((event) {
    event.docs.forEach((element) {
      if (!element.exists) {
        setState(() {
          patientMarkersMap.remove(element.id);
          patientNameMap.remove(element.id);
          patientPhoneNumberMap.remove(element.id);
          sosColor.remove(element.id);
        });
      } else {
        if (element.data()['AssignedUnit'] == "0") {
          namePlaceHolder = element.data()['Name'];
          phoneNumberPlaceHolder = element.data()['Phone'];
          markerPlaceHolder = Marker(
              markerId: MarkerId(element.id),
              position: LatLng(
                  element.data()['latitude'], element.data()['longitude']), // LatLng
              icon: pinLocationIconPatient,
              infoWindow: InfoWindow(
                  title: "Case " + (requestNumber + 1).toString(),
                  snippet: namePlaceHolder)); // InfoWindow // Marker

          setState(() {
            patientMarkersMap[element.id] = markerPlaceHolder;
            patientNameMap[element.id] = namePlaceHolder;
            patientPhoneNumberMap[element.id] = phoneNumberPlaceHolder;
            if (!element.data()['Type']) {
              sosColor[element.id] = Colors.red;
            } else {
              sosColor[element.id] = Colors.white;
            }
          });
        }
      }
    });
  });
}
```

```
_markerHandler() {
  setState(() {
    finalMarkers = patientMarkersMap.values.toList() +
        dispatchedUnitMarkerMap.values.toList();
    dispatchedUnitNumber = dispatchedUnitMarkerMap.length;
    requestNumber = patientMarkersMap.length;
  });
}
```

```
_expantionListHander() {
  setState(() {
    finalNames = patientNameMap.values.toList();
    finalPhoneNumbers = patientPhoneNumberMap.values.toList();
    finalColors = sosColor.values.toList();

    dispatchedUnitNameMap.forEach((key, value) {
      dispatchedUnitEntryMap[key] = dispatchedUnitNameMap[key] +
          " " +
          dispatchedUnitPhoneNumberMap[key];
    });

    dispatchedUnitsEntryList = dispatchedUnitEntryMap.values.toList();
  });
}
```

E-s3ar.com

Assign Dispatched Unit based on Phone Numbers

```
Future<void> assignDispatchedUnit(
    String dispatchedUnitPhoneNumber, String patientPhoneNumber) async {
  String dispatchedUnitDocumentId;
  String patientDocumentId;

  String tokenDispatchedUnit;
  String tokenPatient;
  String namePlaceHolder;
  String phoneNumberPlaceHolder;
  String agePlaceHolder;
  String genderPlaceHolder;
  LatLng locationPlaceHolder;
  bool typePlaceHolder;

  CollectionReference patientCollection =
      FirebaseFirestore.instance.collection('active-cases');

  CollectionReference usersCollection =
      FirebaseFirestore.instance.collection('users');

  CollectionReference dispatchedUnitCollection =
      FirebaseFirestore.instance.collection('active-DU');

  CollectionReference medHistoryCollection =
      FirebaseFirestore.instance.collection('med-history');

  Future<QuerySnapshot> patientQuerySnapshot =
      patientCollection.where('Phone', isEqualTo: patientPhoneNumber).get();

  Future<QuerySnapshot> dispatchedUnitQuerySnapshot = dispatchedUnitCollection
      .where('phone', isEqualTo: dispatchedUnitPhoneNumber)
      .get();

  await patientQuerySnapshot.then((querySnapshot) {
    querySnapshot.docs.forEach((element) {
      patientDocumentId = element.id;
      namePlaceHolder = element.data()['Name'];
      phoneNumberPlaceHolder = element.data()['Phone'];
      locationPlaceHolder =
          LatLng(element.data()['latitude'], element.data()['longitude']);
      typePlaceHolder = element.data()['Type'];
    });
  }).catchError((onError) {
    print(onError);
  });

  await dispatchedUnitQuerySnapshot.then((querySnapshot) {
    querySnapshot.docs.forEach((element) {
      dispatchedUnitDocumentId = element.id;
    });
  }).catchError((onError) {
    print(onError);
  });

  patientCollection.doc(patientDocumentId).update(
      {'AssignedUnit': dispatchedUnitDocumentId}).catchError((onError) {
    print(onError);
  });
```

```
dispatchedUnitCollection.doc(dispatchedUnitDocumentId).update({
  'ActiveCase': patientDocumentId,
  'isAvailable': false
}).catchError((onError) {
  print(onError);
});

setState(() {
  // Remove patient data from map and expanded list
  patientMarkersMap.remove(patientDocumentId);
  patientNameMap.remove(patientDocumentId);
  patientPhoneNumberMap.remove(patientDocumentId);

  //remove dispatched unit data from map

  dispatchedUnitMarkerMap.remove(dispatchedUnitDocumentId);
  dispatchedUnitPhoneNumberMap.remove(dispatchedUnitDocumentId);
  dispatchedUnitNameMap.remove(dispatchedUnitDocumentId);
  dispatchedUnitEntryMap.remove(dispatchedUnitDocumentId);
});
Future<DocumentSnapshot> usersDocumentSnapshot =
    usersCollection.doc(patientDocumentId).get();

await usersDocumentSnapshot.then((value) {
  agePlaceHolder = value.data()['age'].toString();
  genderPlaceHolder = value.data()['sex'];
  tokenPatient = value.data()['FCMToken'];
});

Future<DocumentSnapshot> dispatchedUnitUserDocumentSnapshot =
    usersCollection.doc(dispatchedUnitDocumentId).get();

await dispatchedUnitUserDocumentSnapshot.then((value) {
  tokenDispatchedUnit = value.data()['FCMToken'];
});
//send data here
Future<DocumentSnapshot> medHistoryDocumentSnapshot =
    medHistoryCollection.doc(patientDocumentId).get();
```

```
  await medHistoryDocumentSnapshot.then((value) {
    sendUserData(
        "1",
        tokenDispatchedUnit,
        namePlaceHolder,
        locationPlaceHolder.latitude.toString(),
        locationPlaceHolder.longitude.toString(),
        agePlaceHolder,
        genderPlaceHolder,
        phoneNumberPlaceHolder,
        value.data()['Blood_Type'],
        value.data()['Heart_Condition'],
        value.data()['Allergies'],
        value.data()['Medications'],
        value.data()['Pre-existing_Conditions'],
        patientDocumentId,
        value.data()['Diabetic'],
        typePlaceHolder.toString());
  });
  try {
    await sendPatientNotification("2", tokenPatient);
  } catch (e) {
    print(e);
  }
}
```

E-s3at.com

Cancel Patient Request

```
Future<void> cancelRequest(String phoneNumber) async {
  String patientCollectionDocId;
  CollectionReference patientCollection =
      FirebaseFirestore.instance.collection('active-cases');
  CollectionReference usersCollection =
      FirebaseFirestore.instance.collection('users');

  Future<QuerySnapshot> pateintQuerySnapshot =
      patientCollection.where('Phone', isEqualTo: phoneNumber).get();

  await pateintQuerySnapshot.then((value) {
    value.docs.forEach((element) {
      patientCollectionDocId = element.id;
    });
  });

  patientCollection.doc(patientCollectionDocId).delete();

  usersCollection.doc(patientCollectionDocId).update({
    'hasActiveRequest': false,
  });

  setState(() {
    patientMarkersMap.remove(patientCollectionDocId);
    patientNameMap.remove(patientCollectionDocId);
    patientPhoneNumberMap.remove(patientCollectionDocId);
    sosColor.remove(patientCollectionDocId);
  });
}
```

E-s3

Track Patient Requests Using Phone Number

```
_trackingHandler(String phoneNumber) async {
  String dispatchedUnitDocId;
  String namePlaceHolder;
  String phoneNumberPlaceHolder;
  Marker markerPlaceHolder;

  CollectionReference patientCollection =
      FirebaseFirestore.instance.collection('active-cases');
  CollectionReference dispatchedUnitCollection =
      FirebaseFirestore.instance.collection('active-DU');

  Future<QuerySnapshot> patientSnapshot =
      patientCollection.where('Phone', isEqualTo: phoneNumber).get();

  await patientSnapshot.then((querySnapshot) {
    if (querySnapshot.docs.length == 0) {
      showDialog(
          context: context,
          builder: (context) {
            return AlertDialog(
              title: Text("Alert"),
              content: Text("Please enter a valid mobile number."),
              actions: <Widget>[
                FlatButton(
                  child: Text("OK"),
                  onPressed: () {
                    Navigator.pop(context);
                  },
                ) // FlatButton
              ], // <Widget>[]
            ); // AlertDialog
          });
      throw Exception(
          "Mobile number provided does not exist in active cases");
    }
    querySnapshot.docs.forEach((element) {
      namePlaceHolder = element.data()['Name'];
      markerPlaceHolder = Marker(
          markerId: MarkerId(element.id),
          position: LatLng(
            element.data()['latitude'],
            element.data()['longitude'],
          ), // LatLng
          icon: pinLocationIconPatient,
          infoWindow:
              InfoWindow(title: namePlaceHolder, snippet: phoneNumber)); // Marker
      if (element.data()['AssignedUnit'] == "0") {
        showDialog(
            context: context,
            builder: (context) {
              return AlertDialog(
                title: Text("Alert"),
                content: Text(
                    "This patient does not yet have an ambulance assigned to it."), // Text
                actions: <Widget>[
                  FlatButton(
                    child: Text("OK"),
                    onPressed: () {
                      Navigator.pop(context);
                    },
                  ) // FlatButton
                ], // <Widget>[]
              ); // AlertDialog
            });
      } else {
        dispatchedUnitDocId = element.data()['AssignedUnit'];
      }

      setState(() {
        markerMap[element.id] = markerPlaceHolder;
      });
    });
  }).catchError((error) {
    print(error);
  });
```

```
  if (dispatchedUnitDocId != null) {
    Future<DocumentSnapshot> dispatchedUnitDocument =
        dispatchedUnitCollection.doc(dispatchedUnitDocId).get();

    await dispatchedUnitDocument.then((value) {
      phoneNumberPlaceHolder = value.data()['phone'];
      namePlaceHolder = value.data()['name'];
      markerPlaceHolder = Marker(
          markerId: MarkerId(value.id),
          position: LatLng(
            double.parse(value.data()['Latitude']),
            double.parse(value.data()['Longitude']),
          ), // LatLng
          icon: pinLocationIconAmbulance,
          infoWindow: InfoWindow(
              title: namePlaceHolder, snippet: phoneNumberPlaceHolder)); // InfoWindow // Marker

      setState(() {
        markerMap[value.id] = markerPlaceHolder;
      });
    }).catchError((error) {
      print(error);
    });
  }
  _markerHandler();
}
```

```
_markerHandler() {
  setState(() {
    finalMarkers = markerMap.values.toList();
  });
}
```

E-S

Send Patient Data & Notification to Dispatched Unit through POST request

```
Future<Patient> sendUserData(
    String msgType,
    String registrationToken,
    String name,
    String lat,
    String lng,
    String age,
    String sex,
    String phoneNo,
    String bloodtype,
    String heartCond,
    String allergies,
    String currentMeds,
    String preCond,
    String patientID,
    String diabetic,
    String type) async {
  var url = 'http://localhost:3001/firebase/notification';
  Map data = {
    'msgType': msgType,
    'registrationToken': registrationToken,
    'name': name,
    'age': age,
    'sex': sex,
    'phoneNo': phoneNo,
    'bloodtype': bloodtype,
    'heartCond': heartCond,
    'allergies': allergies,
    'currentMeds': currentMeds,
    'preCond': preCond,
    'patientID': patientID,
    'currLat': lat,
    'currLong': lng,
    'diabetic': diabetic,
    'type': type.toString()
  };
  var body = json.encode(data);
  final http.Response response = await http.post(url,
      headers: <String, String>{
        'Content-Type': 'application/json; charset=UTF-8',
      },
      body: body);

  if (response.statusCode == 201) {
    return Patient.fromJson(jsonDecode(response.body));
  } else {
    throw Exception(response.statusCode);
  }
}
```

Send Notification to Patient through POST Request

```
Future<PatientNotif> sendPatientNotification(
    String msgType, String registrationToken) async {
  var url = 'http://localhost:3001/firebase/notification';
  Map data = {
    'msgType': msgType,
    'registrationToken': registrationToken,
  };
  var body = json.encode(data);
  final http.Response response = await http.post(url,
      headers: <String, String>{
        'Content-Type': 'application/json; charset=UTF-8',
      },
      body: body);

  if (response.statusCode == 201) {
    return PatientNotif.fromJson(jsonDecode(response.body));
  } else {
    throw Exception(response.statusCode);
  }
}
```

Features

The table below lists the features that were mentioned and whether they've been implemented or not.

Feature	Is It Implemented?
SMS Login	Yes
Request An Ambulance	Yes
Medicine Reader	Yes
SOS Signal	Yes
First-Aid Tips	Yes
Display Active Requests	Yes
Display Request Details	Yes
Display Logs	No
Duty Hours Control	Yes
Display Notifications for Case/Request/Emergencies	Yes
Display Patient Request Details and Personal Information	Yes

E-3af.com

Language Control	Yes
Announcements	Yes
Real-Time Tracking	Yes

Chapter 6
Testing

6.1 Testing Approach

6.1.1 Functional Testing

For functional testing, we mainly utilized techniques such as:

1. Unit Testing: In unit testing, we tested each module separately before committing to it and integrating it with the rest of the system, allowing us to find errors or flaws before complicating the system even further.
2. Integration Testing: After integrating the different modules, we test the functionalities as a whole, for example, when testing the patient's interface, we tested requesting an ambulance and viewed the results directly on the database instead of seeing if they reached the dispatcher so that if we had an issue we would know for sure that it came from the patient's end and no the other applications in the system.
3. Regression Testing: While working on this project, changes to already implemented functionalities had to be changed either for efficiency or compatibility updates, so regression testing ensured that the units connected to one another were working perfectly after the change was made.
4. System Testing: After making sure that we had no bugs in the individual applications, we tested the entire system for different cases, such as requesting an ambulance and waiting for the dispatcher to accept it and assign a dispatched unit for that request, making sure the notifications and alerts needed are correctly sent.
5. User Acceptance Testing: The pandemic caused difficulties for us in the development phase of E-s3af.com where we were not able to meet up or recruit users to help with User Acceptance Tests or stress tests, but we tried emulating a real-life environment where we assigned physical devices and emulators as Patients and Dispatched Units and made sure that the accuracy of the location tracking and the speed of the database was up to par for an emergency response system.

E-s3af.com

Our approach for testing was Black box testing as well as White box, Black box testing came in the form of testing out the outputs based on our specific inputs to the system, such features tested using this technique were location tracking and the URL launcher used for nearby hospitals, as well as most other features. For White box testing we used print statements and breakpoints in our code to know what is happening to your inputs and commands inside the system, this approach was used in the most critical features in our system such as Read/Write operations in our database to solve problems with timing and false data, as well as hardware manipulation code such as using the phone's flash and diagnosing the issue if the hardware was not responding.

6.1.2 Non-Functional Testing

For non-functional testing, we have done the following tests:

1. Security: In terms of security, our application allows the user to always keep their information secure via cloud storage, where only the administrators can view a patient's data and the user itself, this is made possible by using "Firebase Authentication Rules'' which allows to white list certain groups to read and write data as well as the user reading/writing their own. While the information sent to the dispatched unit is only made once using a single post request therefore, it is harder to interfere with it. We also maintain the user's trust by using an SMS sent OTP so there is no way a user can have their account stolen.

2. Efficiency and Reliability: The system has been tested through and through for speed, where we store timestamps for each request during the testing phase and compare them with each other for consistency.

3. Load Testing: The system's ability to handle a large sum of data was tested in terms of Read/Write operations by subjecting it to 1 million sequential Read/Write instructions within a day, which is 10 times the expected maximum of a dispatcher working an 8 hour shift, which leads us to the conclusion that our system has the capability of handling a larger load of data making it suitable for real world use.

Our application has also fulfilled Nielsen's 10 principles for UI design, including:

1. Visibility of System Status: Our system tries to establish trust by making sure to display the system status continuously, as well as allow the users to know what actions they took and allow them to decide what to do next.
2. Aesthetic and Minimalist Design: Our design was meant to be simple and delightful for the eyes, prominent colors and round shapes were chosen in order to make it more usable and friendlier.
3. Consistency and Standards: We developed our application to be easy to use and navigate, which meets users' expectations when using a well-developed application, working on improving learnability has been a priority because our application might be used by a wide array of people with different depths of knowledge.

E-3af.com

4. Error Prevention: Showing error messages that are clear and understandable by a user with no technical experience in the field is apparent in our application.

E-s3af.com

6.2 Testing Tools

- Android Emulator
- Visual Studio Code debugging mode
- Physical Devices

E-s3af.com

6.3 Tested Features

Discuss here which features were tested and which were not and explain why.

Put title here

Requirement	Description	Name in Application	Tested
FR1	Login	FLogin()	Yes
FR2	Request an Ambulance	ReqAmb(), requestHelp()	Yes
FR3	Medicine Reader	BarcodeReader(), scanbc(), MedResult()	Yes
FR4	SOS Signal	SOS(), RequestSOS(), requestHelp()	Yes
FR5	First-Aid	HomeScreen()	Yes
FR6	Display Active Requests	activeUserManagement(), activeCaseListener(), _markerHandler()	Yes
FR7	Display Request Details	activeUserManagement(), activeCaseListener(), _expansionListHandler()	Yes
FR9	Duty Hour Control	SwitchStatus(), getStatus(), setStatus()	Yes
FR10	Display Notifications for Case/Request/Emergencies	activeUserManagement(), activeCaseListener(), _markerHandler(), _expansionListHandler()	Yes
FR11	Display Patient Request Details and Personal Information	activeUserManagement(), activeCaseListener(), _expansionListHander()	Yes
FR12	Language Control	Translations()	Yes

E-3af.com

6.4 Discussion

Discuss here any test results that have impacted the design or implementation of the system.

1- Real-Time tracking: Real-Time tracking has proven to be one of the obstacles we have encountered throughout our application, the dispatcher to be exact.

As the dispatcher, it is required to be able to view all the patient requests and active dispatched units simultaneously and in real time, in order to assign the units properly and avoid any possible errors that would cause unnecessary and unwanted delays.

Our initial approach was to use Lists in which the data for each marker appearing on the map is stored, however, that approach was not efficient in the long run as it was difficult to be able to update each list element without knowing the exact address of the data, as well as data removal.

To which we have changed our approach to using hash-maps instead, where the key is the user ID, and the value is for the markers that we would like to update the location for. This approach has allowed us to not only know the address of each element using its unique key, but also made updating and removing the data easier on our application.

Finally, with each time the location data of each dispatched unit is updated in our database, each dispatched unit marker is updated in real time, and a slight delay of 2 milliseconds was added in order to ensure the updates appearing on the map correctly and with very high accuracy.

2- Language Control: At first, we thought that we should make a button for changing the language of the application, but that came at the cost of making every widget with text a "stateful widget" which would make the code too long and impractical in terms of constructing classes, so we decided to utilize the localization features provided by flutter.

Flutter allows us to track the device's locale and find their preferred language, if the phone's language is in arabic then the entire application would be displayed in arabic and the app drawer and text would automatically become right to left, the same if the device was in english.

This turned out to be an advantage because of the ability to include many more languages in the future easily by creating a new JSON dictionary for the language.

E-3af.com

3- Database Design: Our application is mainly based on the manipulation and use of data from a large online database, so our choice of database was based on how fast, how reliable, and how scalable it is, we came to the conclusion that we should utilize Google's Firebase as our database, since it updates in real time and has the scalability needed to turn this into a nationwide project, as well as features that are important for our application such as SMS OTP and real-time tracking.

Since firebase is a unique database that is different from traditional databases, it consists of documents and a class-like structure, which resembles an object-oriented approach, we had to change our application's design, so we did not create a class for each user but we utilized the database's class-like structure in order to make it seem like each user was an object inside the application.

Since all the data is stored on the database, we had to develop our app with a lot of asynchronous functions that work hand in hand to fetch data from the database whenever it was needed, as well as update and upload new data, our new approach turned out to be very efficient and fast in terms of reads and writes.

E-s3af.com

Chapter 7
Conclusions and Future Work

Es3af.com has a lot of potential in expanding among the health sector in our country, there are many features that can be perceived and implemented in the future such as implementing the ability to find the closest hospital according to the patient's insurance that will be enlisted in their profile. As the application supports localization for both English and Arabic speakers, there are plans to implement more language support for the foreigners that come to visit our beloved country.

Furthermore, improvements on the existing features and services will always be on top of the list to keep the platform on point and provide the smooth experience it should.

Some of the features that were discontinued but to be moved as future work are as follows:

-Action Logging: For now, everything is logged per user on their own document that is stored on FireStore.

-Voice XML: Due to time constraints and lack of compatibility between XML and Flutter this feature will be re-evaluated and implemented differently in the future.

-Mission Assignment: Improved mission assignment and tracking overtime as consistency should be at its fullest.

E-s3af.com

Appendix A
Users' Manual

For the initial installation, the application will be available on Google Playstore, after the installation from there and launching the app the procedure will be as follows for each user case.

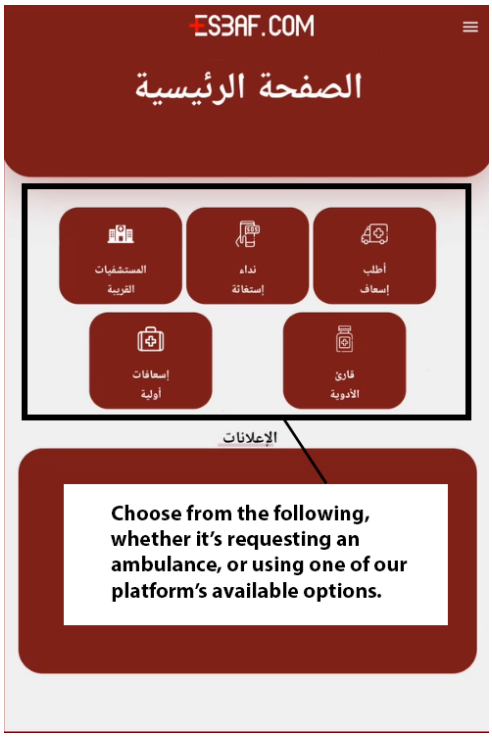
Regular User

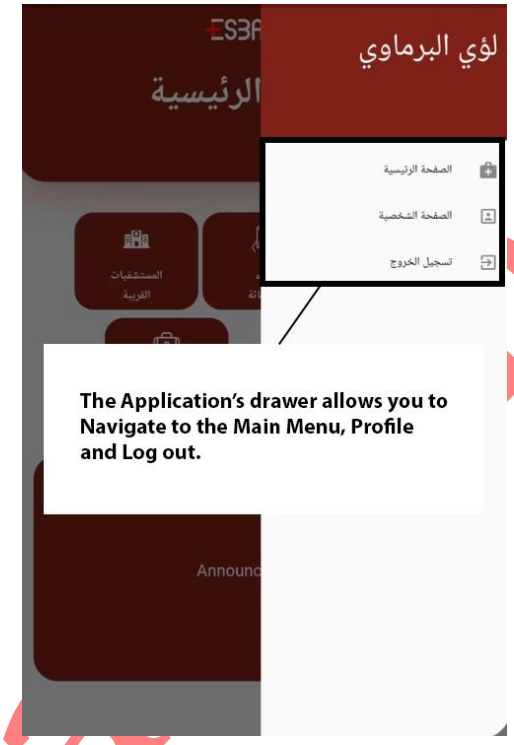
1. The main landing page for a new user will prompt you to login using your phone number, after entering your phone number you will be prompted with a SMS Code popup and you will be receiving a SMS Code in your texts to input.

Alternatively you can head straight to the First-Aid Tips section as it does not require a login.

2. After a successful login using your phone number, you will be prompted to enter basic personal information as shown, this will be a one-time thing per account

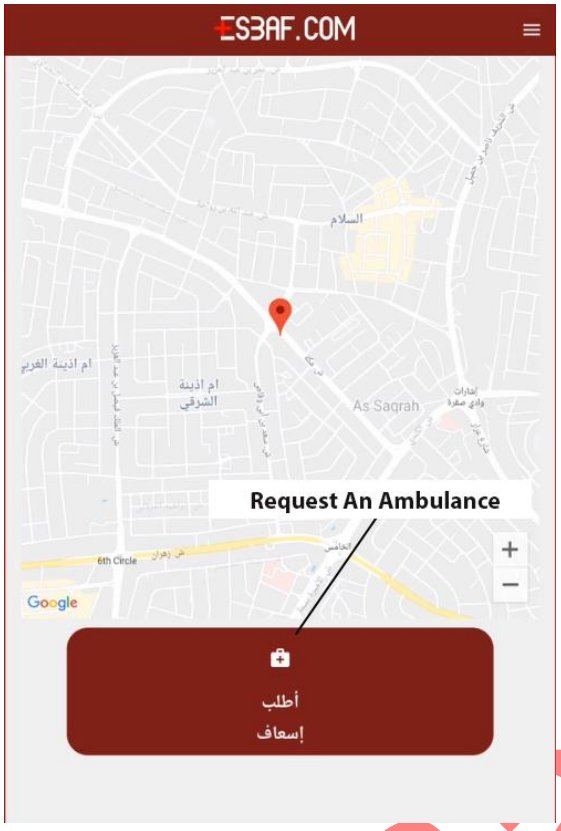
3. After filling out your information, you will be taken to the main landing page that loads by default if you have an existing account, from there you will have several options and to proceed with your goal of using the application, you will also notice the application drawer which will be visible on all pages, which allows for navigation to your Main Menu, Personal Profile and Log out.





E-s3af.com

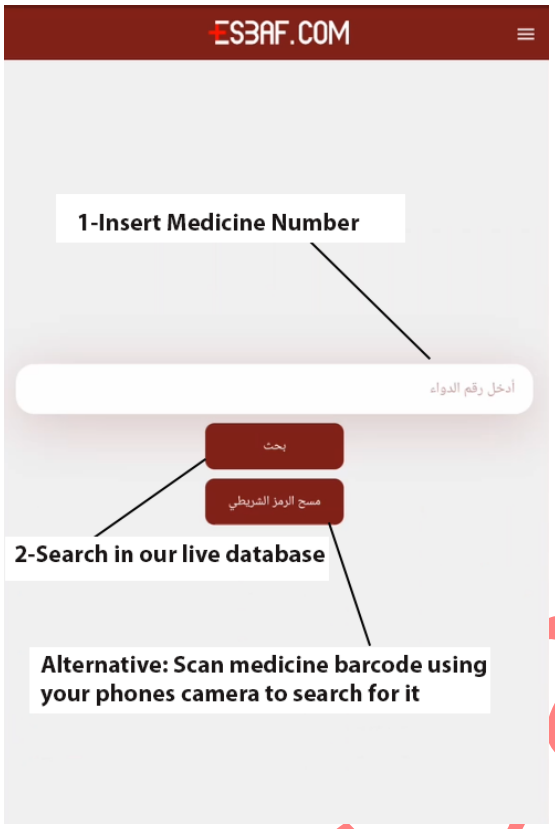
3.1 Request Ambulance, once you click the request button it will send a request to the dashboard to be handled by the dispatcher and an Ambulance will be sent your way as soon as possible.



E-s3af.com

3.2 Medicine Reader

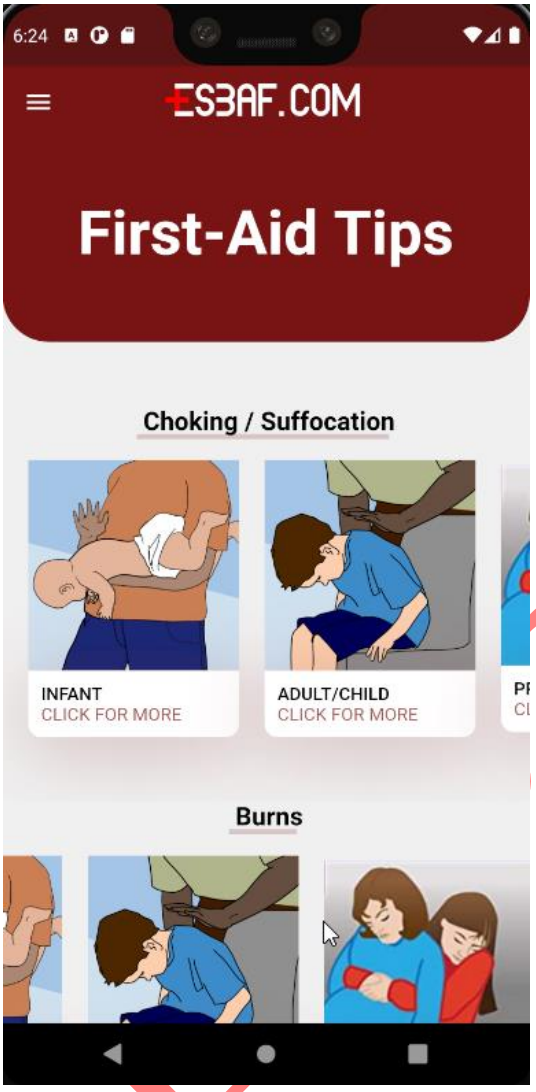
Medicine scanner can be used in two different ways, either by inserting the medicine number manually and tapping on “Search” or scanning it’s barcode using the phone's camera.

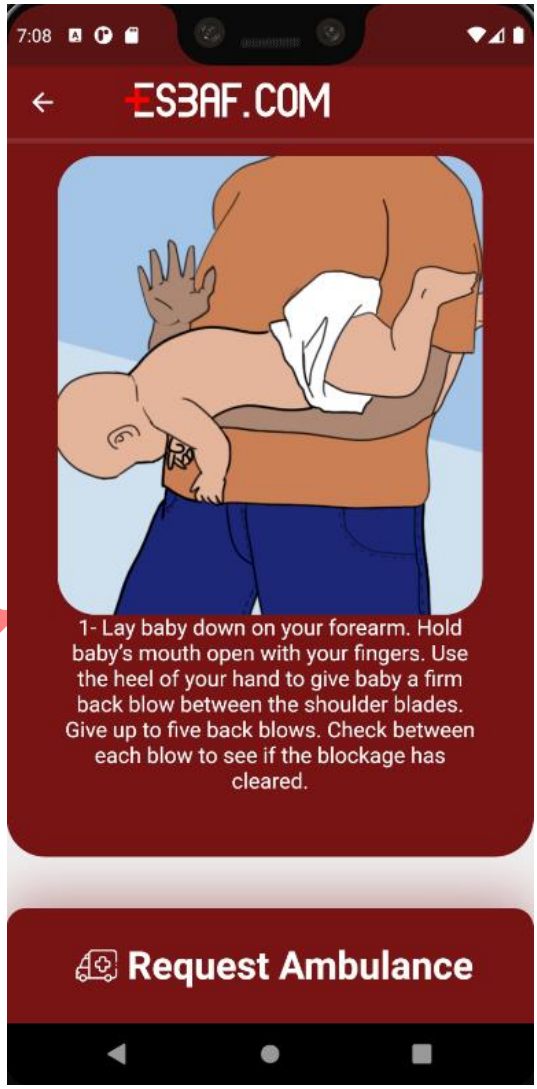


E-Sbaf.com

3.3 First-Aid Tips

On the first-aid tips screen, you can look for the guides based on categories and the navigation inside is by swiping left and right for the correct steps.





4 Language Configuration, Es3af.com is based on your mobile device's localization, meaning it will switch over the language of the application to match your phone's language, if it was in Arabic then the application will be in Arabic and the same applies for English.

Our Application uses your mobile's native language, if your device is in Arabic, it will switch to Arabic, if your device is in English, it will switch to English.

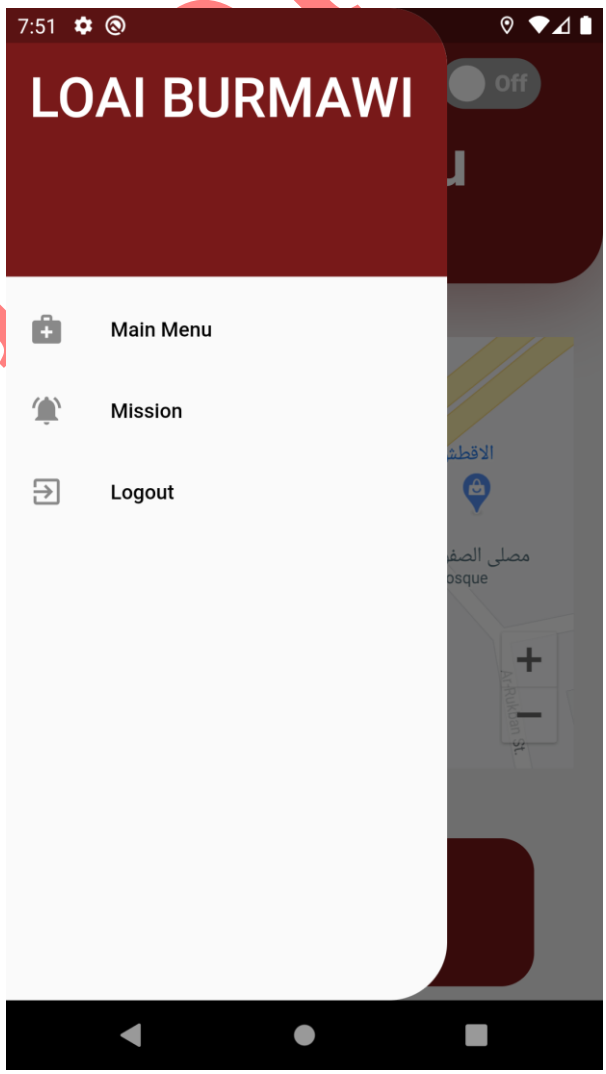
E-s3af.com

Dispatched Unit

The login procedure for the Dispatched Unit is the same as the regular user except they should have an authorized document on Firebase that allows them to login to this application.

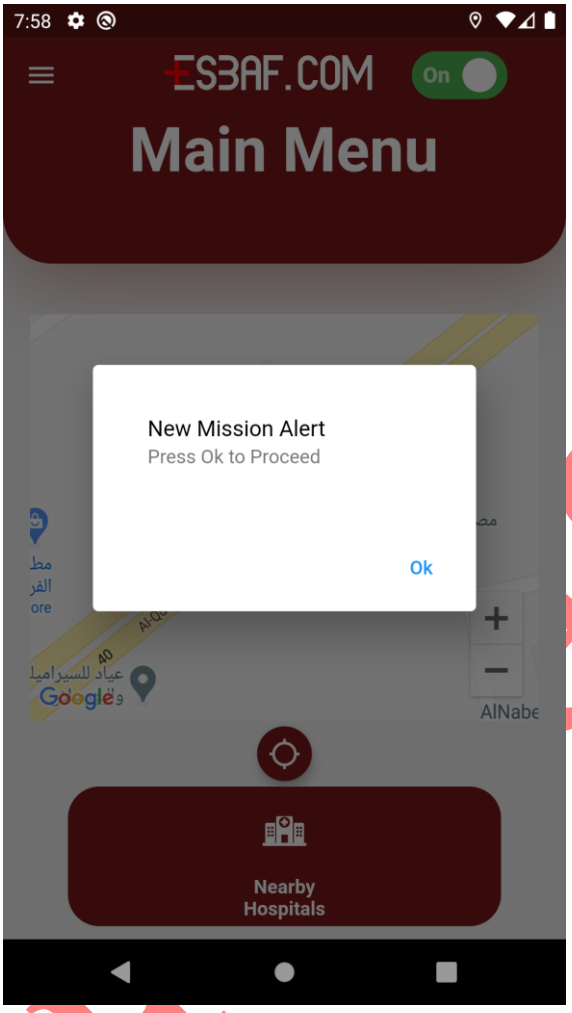
After logging in, they will be navigated to the main menu with a map of their current location and they will receive a notification popup with the mission once they are assigned one. Drawers are also available to load existing missions or navigate.

The Dispatched Unit can go on-duty and off-duty using the slider on the top right corner, they will not be able to receive any missions if they are off-duty.



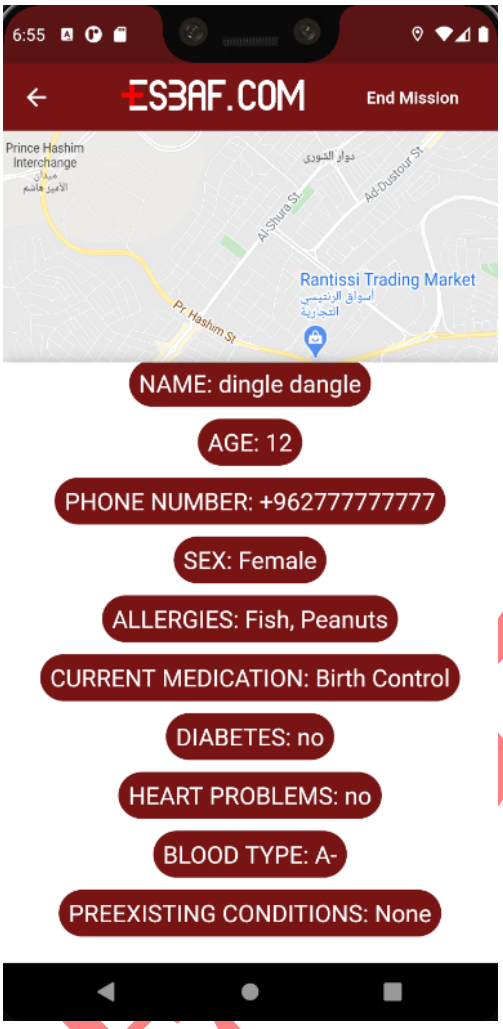
When being assigned a mission, they will be redirected to a map with the patient information and clicking on the pin will give the option to start navigating on the available map applications.

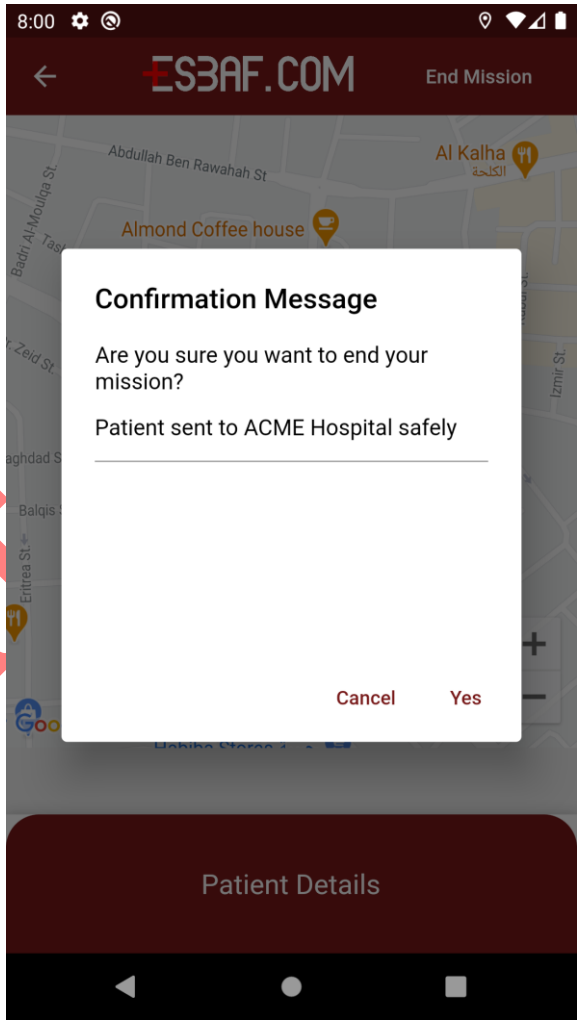
During a mission, patient details will be available by swiping the bottom bar up.



Esbaf.com

Upon finishing the mission, the dispatched unit is prompted with a pop-up where they will fill in brief information about the emergency request they just fulfilled then they will be redirected back to the main menu.



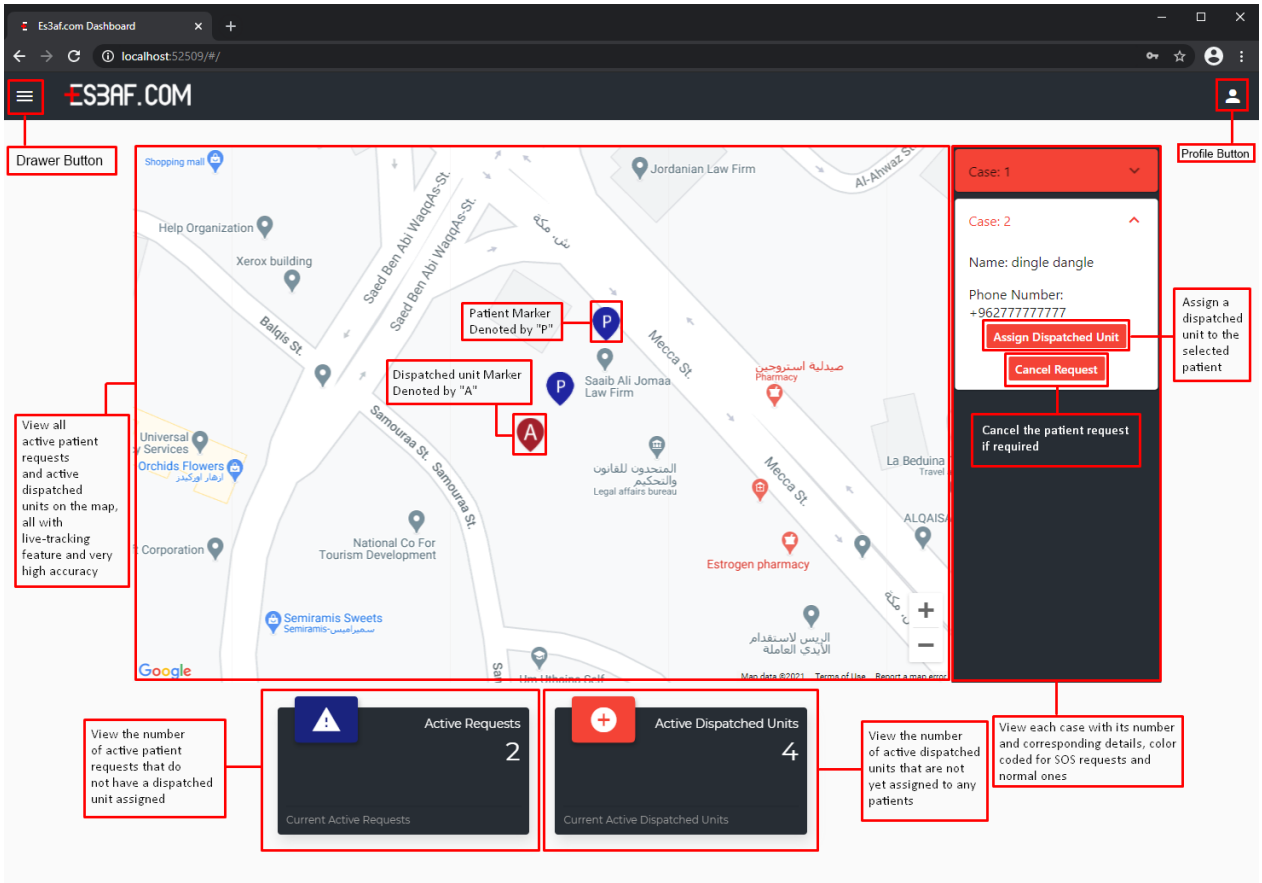


Dispatcher

The login procedure for the dispatcher is different from the mobile applications, as the dispatcher logs in using their email address and password as provided by the database administrators, which limits the ability to create one's account in order to avoid any security issues due to how critical the role is.

The dispatcher is then redirected to the main dashboard page, where they are able to view the current active patient requests, as well as the active dispatched units live on Google maps, and are able to show the details of each marker by tapping on it. Furthermore, as shown on the right side of the map, the dispatcher will also be able to view and navigate through each request with its corresponding details, as they are also shown in a color coded fashion where the dispatcher will be able to differentiate between normal requests and SOS requests coming from the patients.

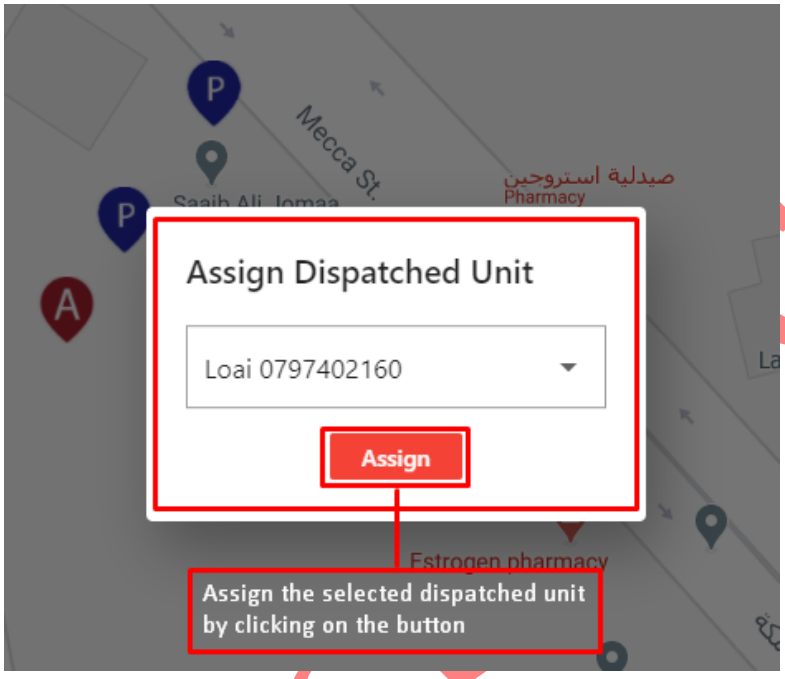
Dispatchers will be able to assign a dispatched unit to the selected patient request, or cancel it accordingly.


Es3af.com Dashboard
localhost:52509/#/
ES3AF.COM
Drawer Button
Profile Button
Case: 1
Case: 2
Name: dingle dangle
Phone Number:
+962777777777
Assign Dispatched Unit
Cancel Request
Assign a dispatched unit to the selected patient
Cancel the patient request if required
Patient Marker Denoted by "P"
Dispatched unit Marker Denoted by "A"
View all active patient requests and active dispatched units on the map, all with live-tracking feature and very high accuracy
View the number of active patient requests that do not have a dispatched unit assigned
Active Requests
2
Current Active Requests
Active Dispatched Units
4
Current Active Dispatched Units
View the number of active dispatched units that are not yet assigned to any patients
View each case with its number and corresponding details, color coded for SOS requests and normal ones

E-s3af

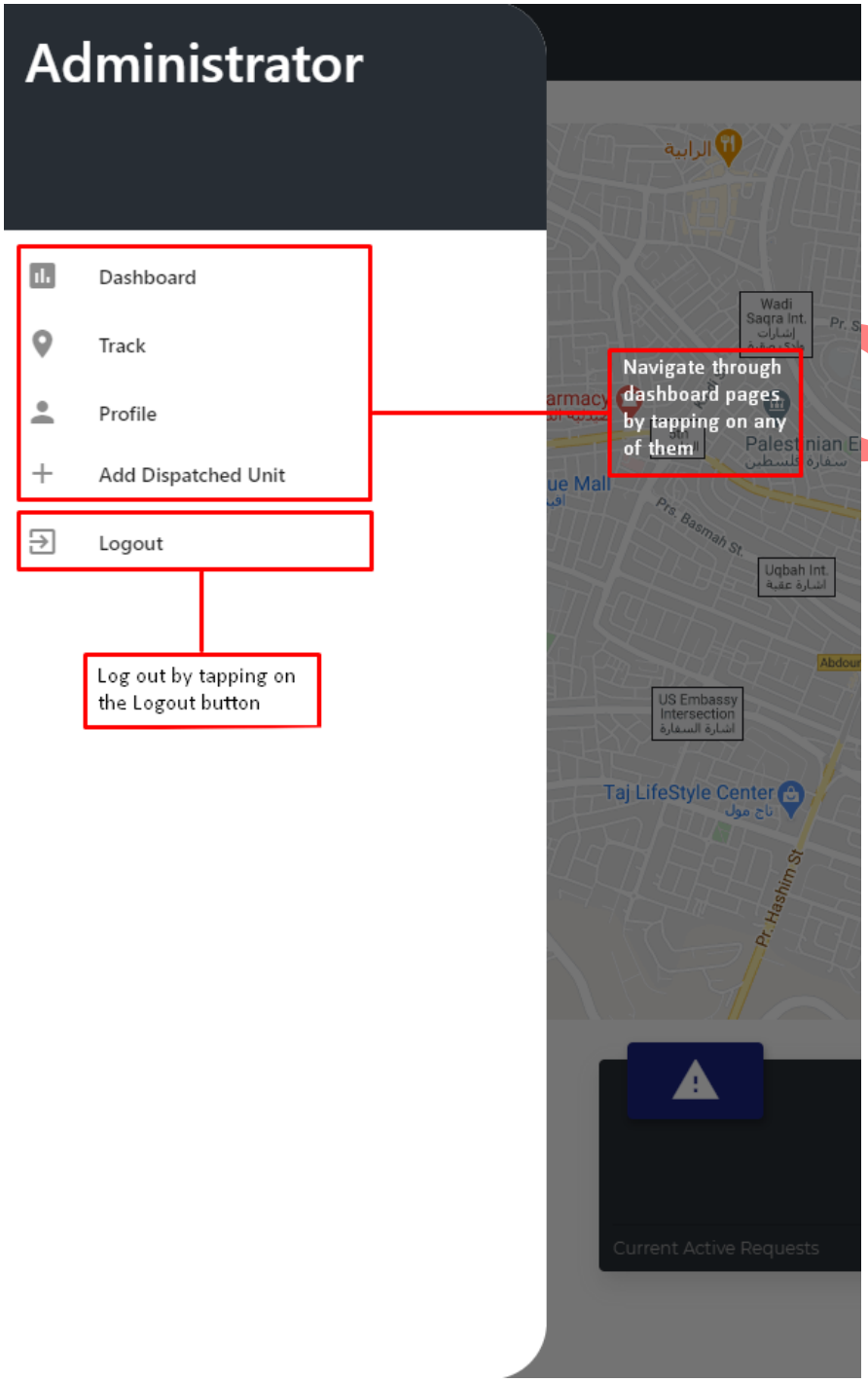
Assigning a dispatched unit can be done by selecting the closest ambulance to the patient from a dropdown list as shown in a prompt dialog. The dispatched unit can be identified on the map by tapping on its marker, and the corresponding data will be available in the dropdown list.

Once the dispatched unit is selected, the dispatcher needs to press on the "Assign" button, and the unit will be assigned to the patient successfully with minimal effort. The dispatched unit will not appear on the map until it is available again, and the patient which the unit was assigned to will also be removed from the map in order to improve the tracking clarity.



E-S com

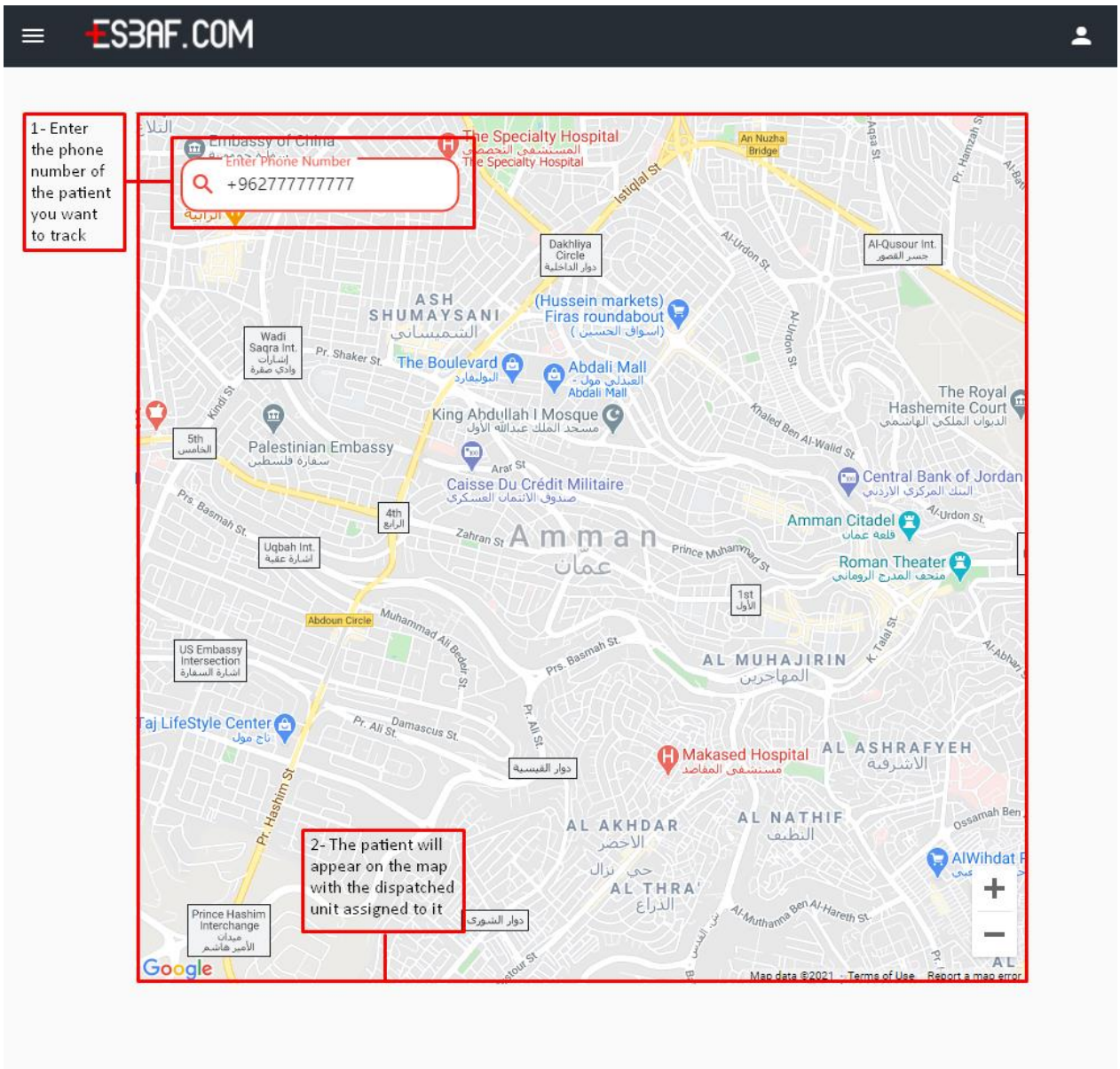
The drawer within the dashboard has a similar functionality as the one viewed previously in the user application, in which the dispatcher is able to navigate through the dashboard's pages seamlessly and with no issues, and is also able to log out once they are done.

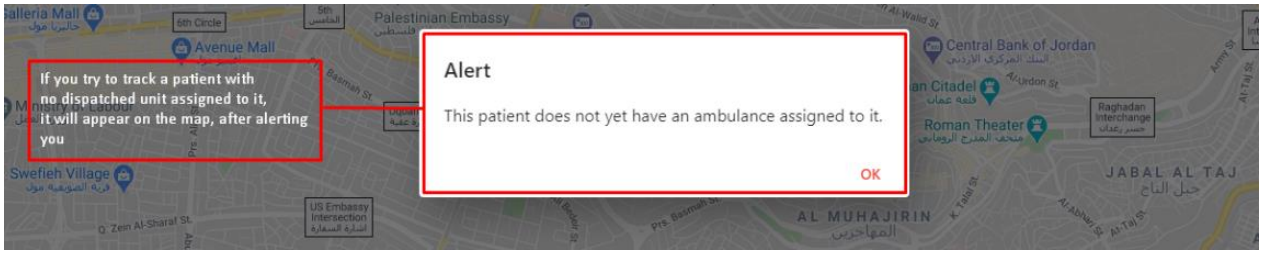


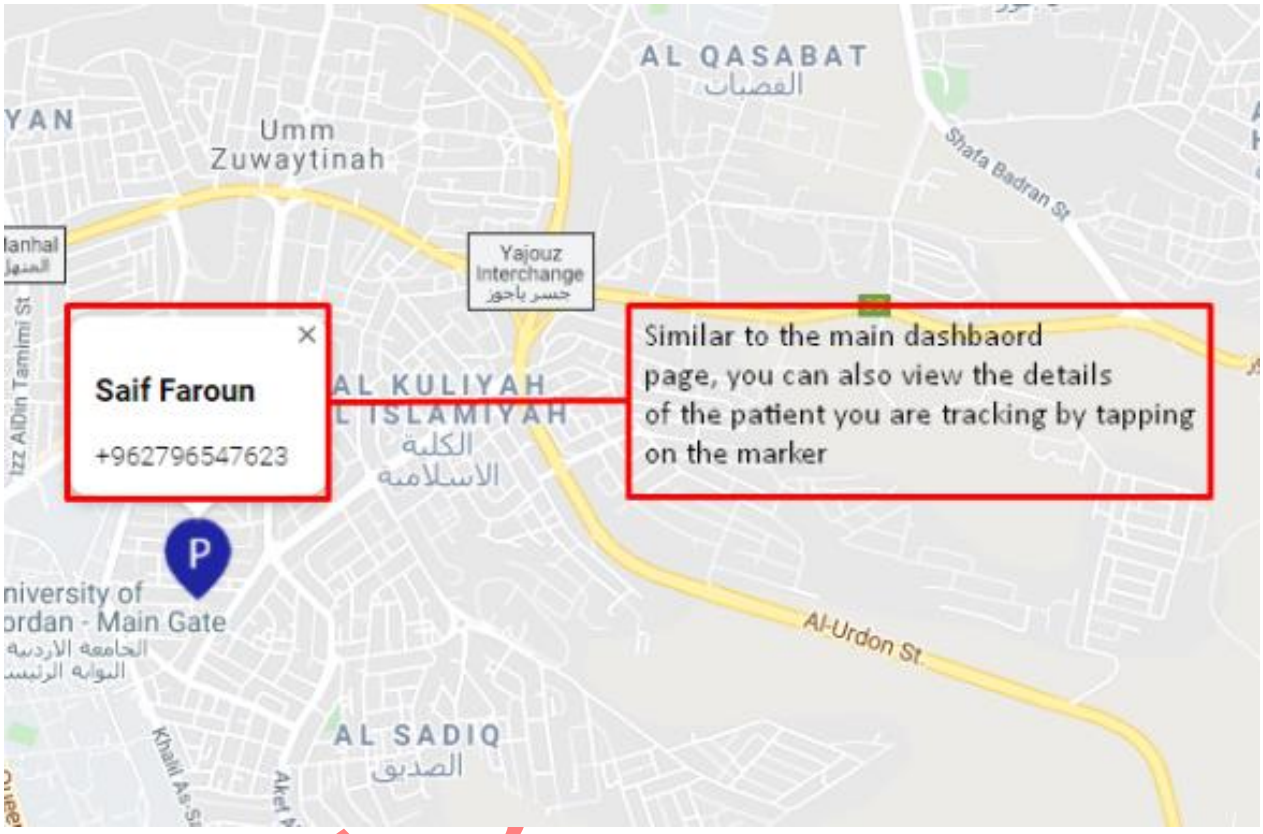
The tracking page allows the dispatcher to track patients that have an active request using their mobile number, as that is considered the most unique way of doing so. If the patient

did not have a dispatched unit assigned to it, the dashboard will display an alert message informing the dispatcher that the patient they are trying to track does not have an ambulance assigned to it, then it will show the patient regardless.

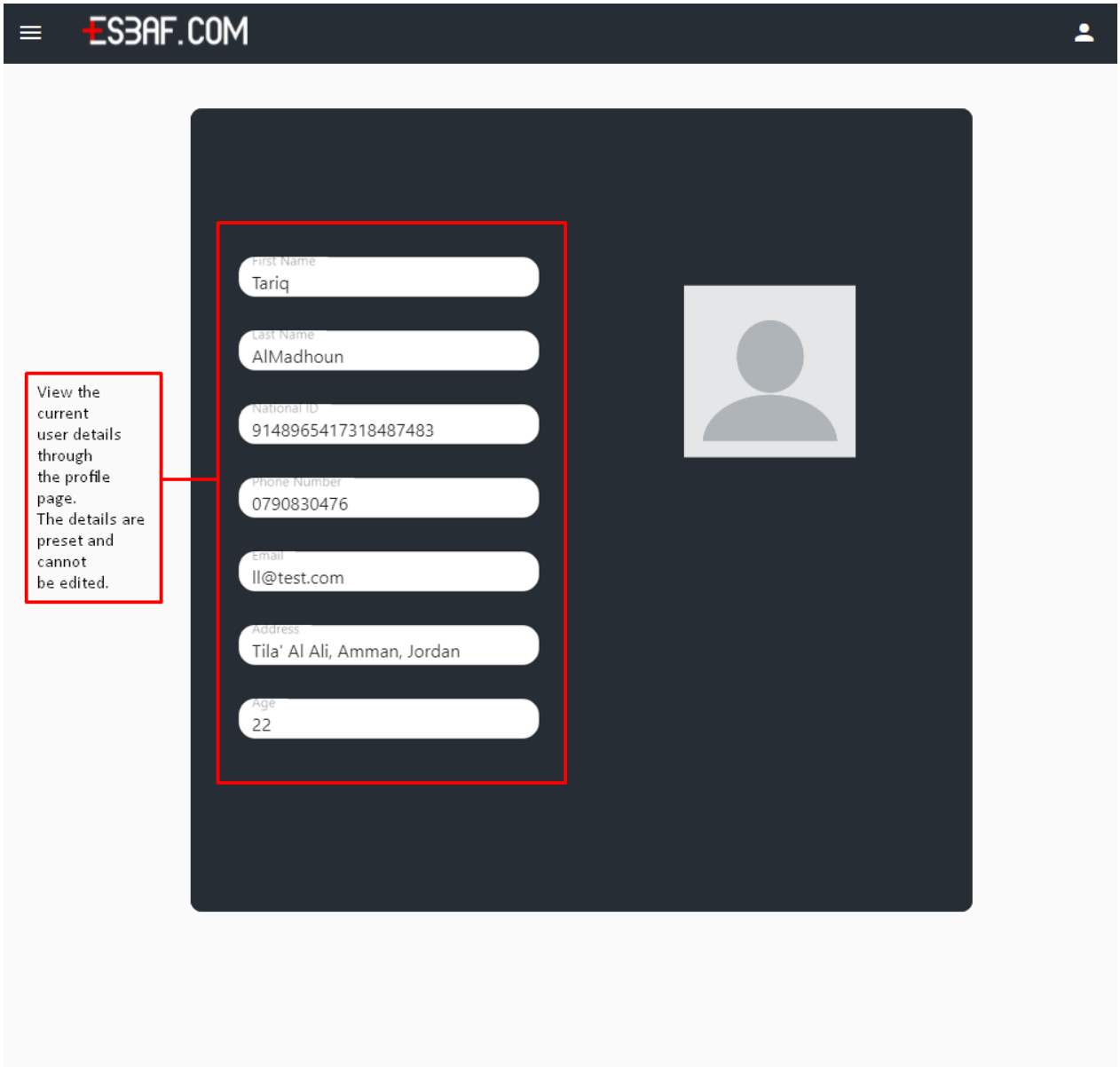
Needless to say, if the patient does have a dispatched unit assigned to it, it will show the patient marker on the map, as well as the marker of the dispatched unit that is assigned to it, and the dispatcher will also be able to view the details of each marker by tapping on them respectively.



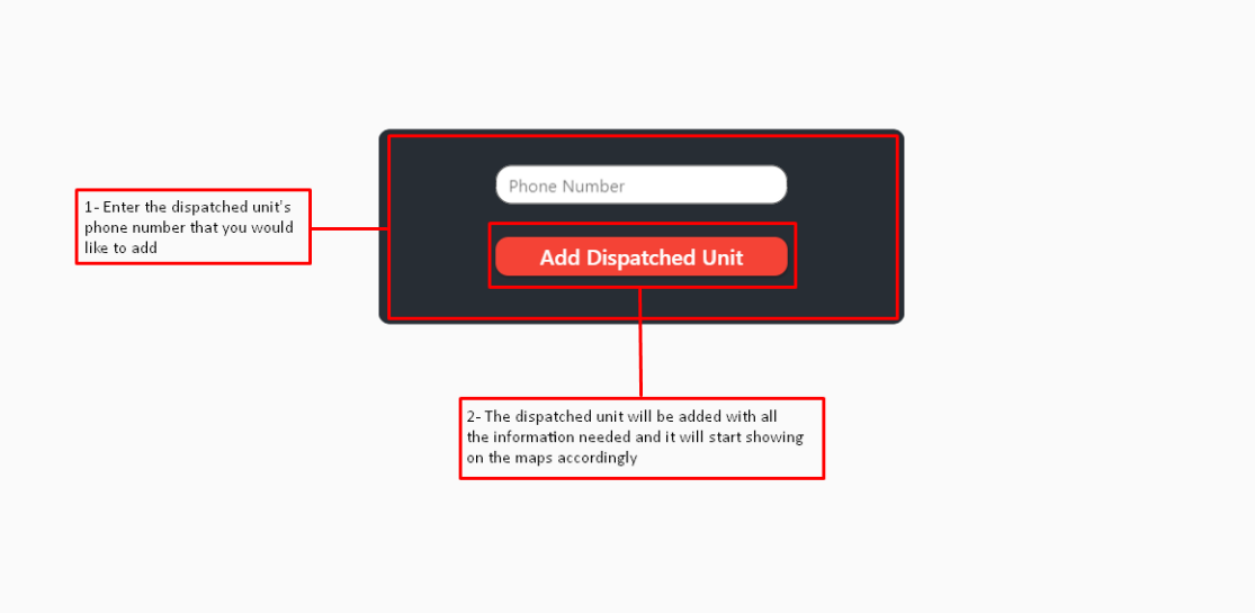

If you try to track a patient with no dispatched unit assigned to it, it will appear on the map, after alerting you
Alert
This patient does not yet have an ambulance assigned to it.
OK
Palestinian Embassy
Avenue Mall
Central Bank of Jordan
Roman Theater
Swefieh Village
JABAL AL TAJ
AL MUHAJIRIN


AL QASABAT
Umm Zuwaytinah
Shafa Badran St
Yajouz Interchange
Saif Faroun
+962796547623
AL KULIYAH
Similar to the main dashbaord page, you can also view the details of the patient you are tracking by tapping on the marker
Al-Urdon St
AL SADIQ

The dispatcher is also able to view their personal details through the profile page. The data previewed on the profile page is Read-Only.



Finally, the dispatcher can also add dispatched units through the Add Dispatched Unit page, where adding one can be done by entering their phone number and it will be added automatically; however, the dispatched unit needs to register on the application beforehand, otherwise, they will not be added.



E-s3at.

Appendix B
Document Changes

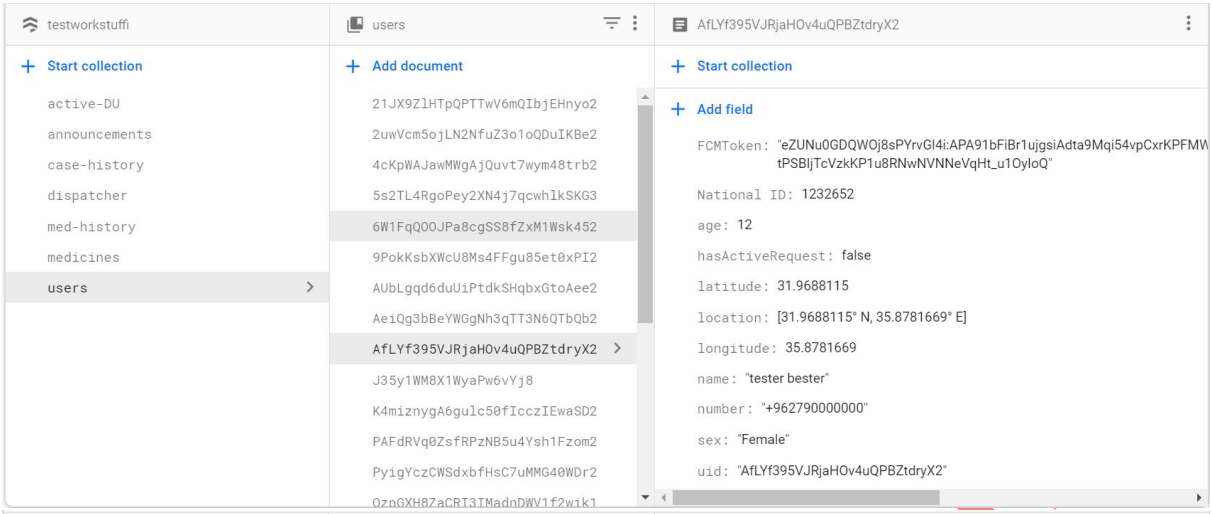
Throughout the entire planning of this project, there were some major and minor changes to the structure, some features were added and some were discontinued until further notice, as well as some changes to the requirements for the whole system that were based on the re-evaluation after running the application.

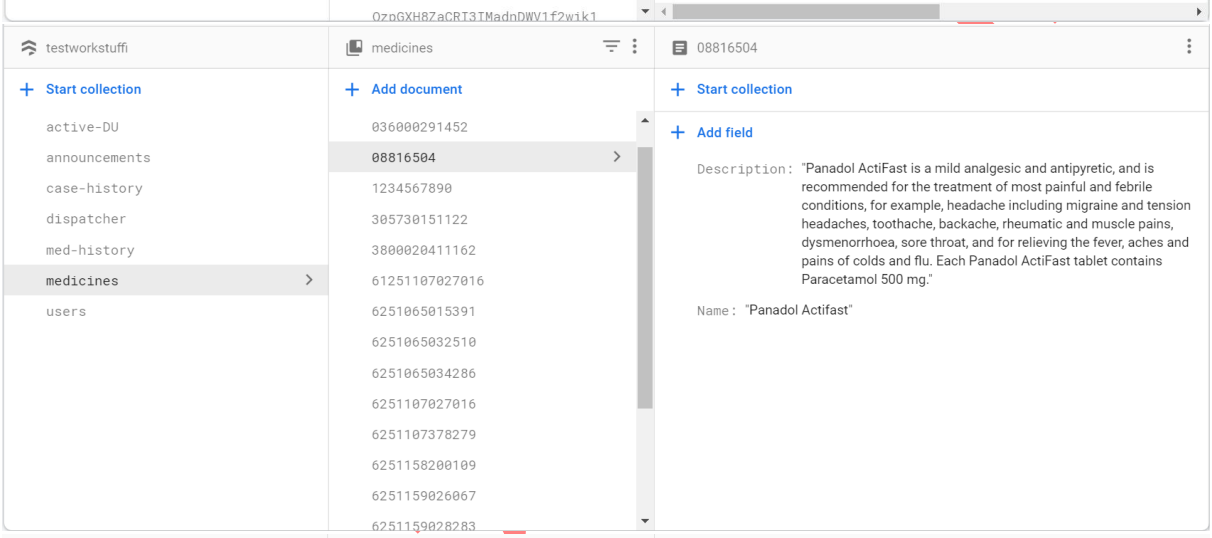
The Changes present are as listed below:

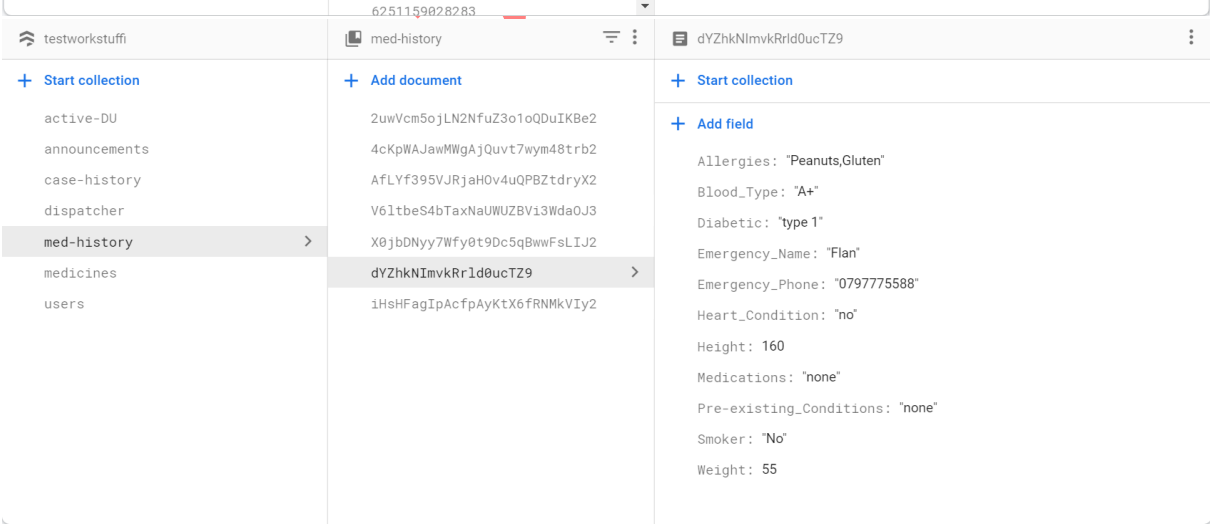
ChangeID	Change	Type of change	Reason for Change
DC1	Case History Availability	Modified	Only dispatchers have access to those for privacy and security reasons of the patient
DC2	VoiceXML	Removed	Lack of compatibility between Flutter and XML
DC3	Announcements	Added	Relaying mass messages to the entire user base
DC4	Database Schema	Modified	Cloudstore works differently
DC5	Platform Requirements	Modified	Only Android support was added.
DC6	Server Requirements	Modified	Servers are only needed for node.js hosting, Since firebase has cloud storage there are no high storage units required anymore.

E-3af.com

New Database Schema







testworkstuffi

Start collection

active-DU
announcements
case-history
dispatcher
med-history
medicines
users

dispatcher

Add document

c6TJfkc6cJUWd86J6wt1unJF0Uh2

c6TJfkc6cJUWd86J6wt1unJF0Uh2

Start collection

Add field

address: "Tila' Al Ali, Amman, Jordan"
age: "22"
email: "ll@test.com"
name: "Tariq AlMadhoun"
natID: "9148965417318487483"
phone: "0790830476"

testworkstuffi

Start collection

active-DU
announcements
case-history
dispatcher
med-history
medicines
users

announcements

Add document

Test Announcement

Test Announcement

Start collection

Add field

Content
0 "Announcement #1"
1 "Announcement #2"
2 "Announcement #3"

testworkstuffi

Start collection

active-DU
announcements
case-history
dispatcher
med-history
medicines
users

active-DU

Add document

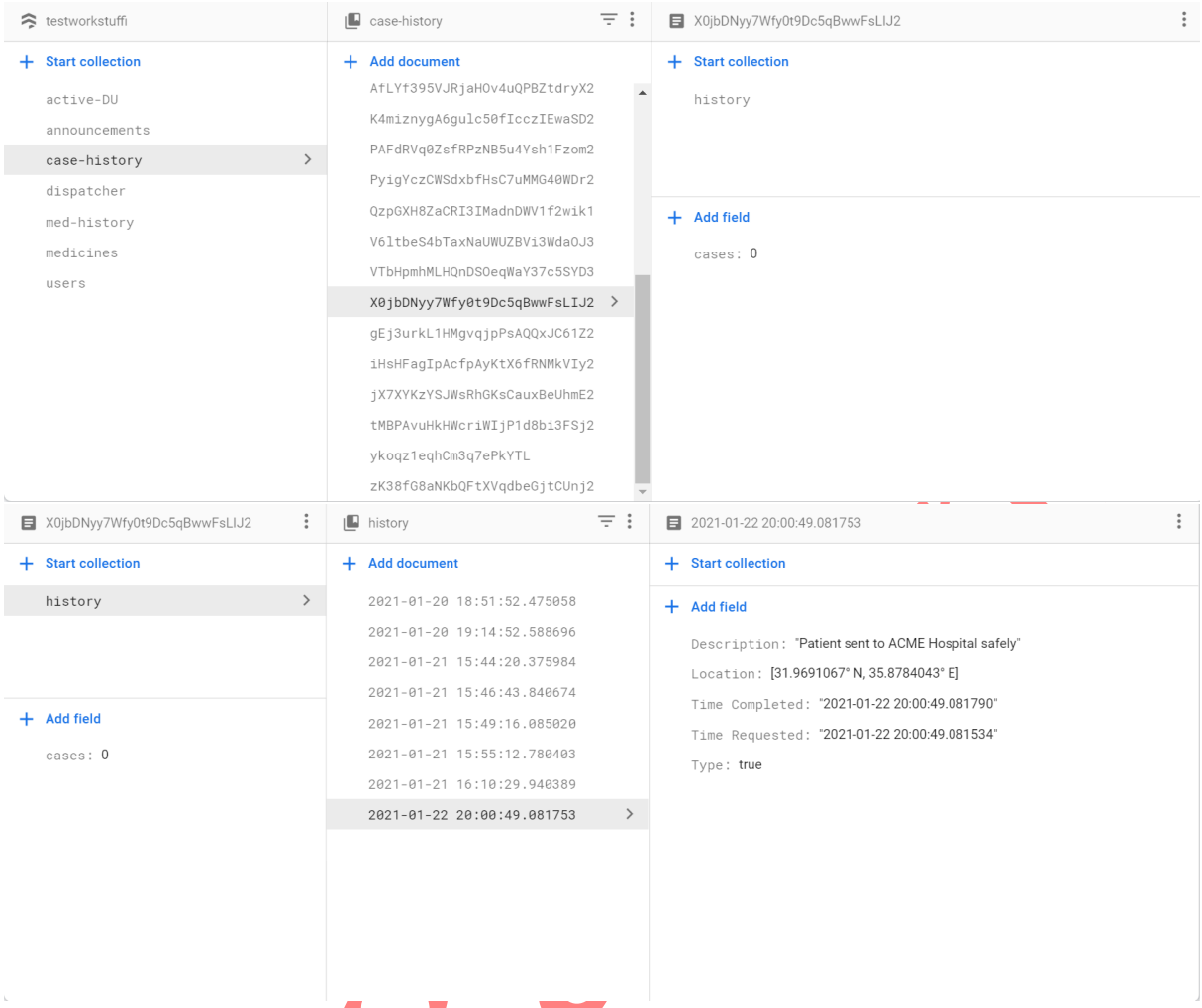
AfLYf395VJRjaHOv4uQPBZtdryX2
V6ltbeS4bTaxNaUWUZBVi3WdaOJ3

V6ltbeS4bTaxNaUWUZBVi3WdaOJ3

Start collection

Add field

ActiveCase: "0"
Latitude: "31.9121633"
Location: [31.9121633° N, 35.8982767° E]
Longitude: "35.8982767"
isAvailable: false
name: "Loai"
phone: "0797402160"


testworkstuffi
Start collection
active-DU
announcements
case-history
dispatcher
med-history
medicines
users
case-history
Add document
AfLYf395VJRjaHOv4uQPBZtdryX2
K4miznygA6gulc50fIcczIEwaSD2
PAFdRVq0ZsfRPzNB5u4Ysh1Fzom2
PyigYczCWSdxbfHsC7uMMG40WDr2
QzpGXH8ZaCRI3IMadnDWV1f2wik1
V6ltbeS4bTaxNaUWUZBVi3WdaOJ3
VTbHpmhMLHQnDSOeqWaY37c5SYD3
X0jbDNyy7Wfy0t9Dc5qBwwFsLIJ2
gEj3urkL1HMgvqjpPsAQQxJC61Z2
iHsHFagIpAcfpAyKtX6fRNMkVIy2
jX7XYKzYSJWsRhGKsCauxBeUhmE2
tMBPAvuHkHWcriWIjP1d8bi3FSj2
ykoqz1eqhCm3q7ePkYTL
zK38fG8aNKbQFtXVqdbeGjtCUnj2
X0jbDNyy7Wfy0t9Dc5qBwwFsLIJ2
Start collection
history
Add field
cases: 0
X0jbDNyy7Wfy0t9Dc5qBwwFsLIJ2
Start collection
history
Add field
cases: 0
history
Add document
2021-01-20 18:51:52.475058
2021-01-20 19:14:52.588696
2021-01-21 15:44:20.375984
2021-01-21 15:46:43.840674
2021-01-21 15:49:16.085020
2021-01-21 15:55:12.780403
2021-01-21 16:10:29.940389
2021-01-22 20:00:49.081753
2021-01-22 20:00:49.081753
Start collection
Add field
Description: "Patient sent to ACME Hospital safely"
Location: [31.9691067° N, 35.8784043° E]
Time Completed: "2021-01-22 20:00:49.081790"
Time Requested: "2021-01-22 20:00:49.081534"
Type: true

E-S

Appendix C
Code Documentation

Since this was split up into three parts, "Code Documentation" will be available under a folder named API for each application that will document the classes used in this project.

E-s3af.com

References

[1] National Ambulance Completes Campaign with 'Be Prepared' Call to the Community (2016). Retrieved from

http://www.nationalambulance.ae/newsroom/180-national-ambulance-completes-campaign-with-be-prepared-call-to-the-community.html

[2] KSA Mobile Government (2020). Retrieved from

https://www.my.gov.sa/wps/portal/snp/pages/mobileGovernment/!ut/p/z0/04_Sj9CPykssy0xPLMnMz0vMAfIjo8zivQIsTAwdDQz9LSw8XQ0CnT0s3JxDfA0MHA30g1Pz9AuyHRUBO1rEew!!/

[3] Indian Red Cross First Aid. Retrieved From

https://www.socialapphub.com/app/indian-red-cross-first-aid

[4] UML Diagrams Based on Eriksson, H., 2004. UML 2 Toolkit. 2nd ed. Indianapolis: Wiley.

[5] "Flutter - Beautiful native apps in record time", *Flutter.dev*, 2021. [Online].

E-s3af.com