ANTONINE UNIVERSITY

Faculty of Engineering

Department of Computer and Communications

E-Dispensary

Student(s)	Hassan Afrah, 201720102
	Obeid Ahmad, 201720443
Major	SW, SN
Campus	BA
Supervisor(s)	Prof. Chaccour Kabalan

A Final Year Project Report submitted in partial fulfilment of the requirements of the degree of Bachelor of Engineering

Fall / Spring 2023

Leave this page blank

Insert a brief dedication (Optional)

ACKNOWLEDGMENT

We would like to express my heartfelt gratitude to everyone who has contributed to the completion of this final year report. Their unwavering support, guidance, and encouragement have played an indispensable role in shaping this work.

First, we are indebted to Antonine University and its Secretary General, Father Michel JALAKH, for giving me the opportunity to pursue my education and engage in meaningful academic endeavors.

I extend my sincere thanks to the faculty's dean Dr. Chady ABOU JAOUDE for his drive for excellence in the faculty.

Our sincere thanks also go to the head of engineering and our supervisor Dr. Kabalan CHACCOUR, for his leadership and continuous support. Your commitment to academic excellence has been a driving force behind our pursuit of knowledge.

We would like to acknowledge the unwavering encouragement and understanding of our families and friends. Their love and belief in our abilities have been my pillars of strength, motivating me to overcome challenges and strive for success.

ABSTRACT

This software project addresses the critical issue of inadequate digital infrastructure in Lebanon's E-Health sector, particularly affecting small medical centers. In a context where E-Health plays a pivotal role, the project's proposed solution involves the creation of a comprehensive web portal. This portal enables efficient management of medicines, appointments, and events for medical centers, fostering streamlined operations and enhanced patient care. As a result, the implemented web application stands as a tangible achievement, boasting multiple user roles to facilitate seamless collaboration and utilization within the medical community. This web-app has been tested to ensure a simple user experience with reliable data against expected bad connections and incorrect use by inexperienced users. Throughout the solution, the project strives to be an empowering step in the lacking Lebanese E-Health sector, helping medical centers to provide more accessible and organized healthcare services.

Ce projet de logiciel aborde le problème critique de l'inadéquation de l'infrastructure numérique dans le secteur de la santé en ligne au Liban, qui affecte particulièrement les petits centres médicaux. Dans un contexte où l'e-santé joue un rôle central, la solution proposée par le projet implique la création d'un portail web complet. Ce portail permet une gestion efficace des médicaments, des rendez-vous et des événements pour les centres médicaux, favorisant ainsi la rationalisation des opérations et l'amélioration des soins aux patients. En conséquence, l'application web mise en œuvre représente une réalisation tangible, avec des rôles d'utilisateurs multiples pour faciliter la collaboration et l'utilisation sans faille au sein de la communauté médicale. Cette application web a été testée pour garantir une expérience utilisateur simple avec des données fiables contre les mauvaises connexions attendues et l'utilisation incorrecte par des utilisateurs inexpérimentés. Tout au long de la solution, le projet s'efforce d'être une étape habilitante dans le secteur libanais de l'e-santé qui fait défaut, en aidant les centres médicaux à fournir des services de santé plus accessibles et mieux organisés.

TABLE OF CONTENT

CHAPTER 1: INTRODUCTION

This project falls into E-Health, particularly health surveillance and health care services. The health sector in Lebanon has not been digitalized yet, with minimal efforts being geared towards medical centers. Processes used are outdated making the governance of records especially challenging. An initiative has been made with the MJO Beirut Social and Medical Center in Achrafieh to have their digital infrastructure developed with web portals for their management, staff, and patients aimed at enhancing efficiency, patient care and overall operational effectiveness. The solution consists of multiple microservices to cater for the different services they provide at their center, and to take future inevitable scalability into consideration. In this report, the development of this solution has been tackled starting with an introduction chapter to identify the problem statement. Throughout the report the rationale behind the project will be highlighted along with the steps to be taken to identify and develop the problems faced, as well as the outcomes achieved through the solution.

1. Problem identification (context, general problem)

The goal of the project is to have a digital infrastructure suitable for an independent medical center, along with a user-friendly web portal for the admins to do their processes, staff to organize and update records and patients to see their medical records and schedule appointments.
Meetings with the stakeholders at MJO social and medical center have shed light on the key issues, the paper-based system and outdated software that have led to inefficiencies, delayed processes, and potential errors. This absence of an infrastructure impedes the collaboration between members, all of which aim for the most efficient patient care. Some of the stakeholders are future users of the solution, and they have distinct roles. The user interface (UI) presented will have to be understood and useful to diverse types of users with varied technical knowledge.
As there are no official governmental rules and standards in the country on the access and security of digital records, they have been discussed with stakeholders. Role-based access control is needed to manage different actions. The center's custom policies are to be applied when developing the solution.
The engineering perspective presents technical problems that surround the project as well, first of them being the ethical handling of medical information, especially with the absence of official rules and regulation on digital ones in the country, this goes hand in hand with security, as any breach cannot be tolerated. In addition, the solution must be perfectly available and efficient, and most importantly easy to use as it is the first of its kind on this scale in our country, with a target audience that is not used to managing their medical information digitally. Thus, the implementation of the solution needs software, systems, and multimedia knowledge.

It is a hope that the solution will be the first of many similar initiatives to push modern practices into our health field, hopefully gaining the attention of officials to start updating Lebanon’s laws to protect patients and the healthcare field.

2. Problem statement / formulation

Medical centers of the smaller scale lack the digitized infrastructure to manage their activities and automate the tedious ones. There is no streamlined digital infrastructure to effectively manage their operations. The sector still relies on physical documents to keep track of medical records and communicate with patients and doctors alike.
Current initiatives only cater for large hospitals, leaving smaller centers technologically behind. There is no middle ground that suits medical centers, especially Lebanese dispensaries.
This problem touches all the pillars of e-health. What it lacks is: a user-friendly solution to structure, organize and plan a dispensary’s data and work in a secure manner.

- Choosing cost-effective tools especially for the infrastructure
- Automating repetitive tasks like e-mails and event publication.
- Having an up-to-date client portal

The two main benefactors of building a satisfactory solution are:

- The people, as they will have secure access to their data without the hassle of having to commute or reach the center.\
- The health sector – medical centers included, as their daily efficiency will increase.

Solving those solutions will shed light on a bigger problem in Lebanon which is the lack of surveillance and practical laws targeting the digital world, an inevitable incorporation in today’s age.

3. Solution approach / methodology

The proposed solution is a multi-role web-app. It can be seen as two functions:

- A user portal to manage their records and appointments and see a medical center’s published information.
- An internal tool for medical center’s doctor, nurses, and admin.

Multiple services are required to implement such a solution. Thus, the Software Development Life Cycle (SDLC) was followed throughout the conception of the project. In addition, as a team of two is working on it, and as the solution has multiple sections, agile methodology was followed to have multiple cumulative outcomes that can be quantified and ready for review. Particularly, ‘kanban’ method of the agile method was followed for its simplicity, which is more applicable in a small team (Ref. Appendix A for more about Agile and Kanban.)

The nature of the project as well as the initial research and feasibility study concluded that following the SDLC is best as it encompasses:

1. **Definition of objectives**: communicated by the supervisor.
2. **Requirements gathering, analysis and feasibility** done first through multiple meetings with the stakeholders at the MJO medical center then by laying out collected information and analyzing them against technology, time, risks, and budget.
3. **General software design:** through use case diagrams and activity diagrams (refer to chapter 4 where a general Software Requirement Specification SRS is laid out with unified modelling language UML diagrams).
4. **Implementation and development:** the direct translation of the features
5. **Testing:** to make sure that the developed feature works against unexpected use cases.
6. **Integration:** as it is multi-module software with one dynamic interface.
7. **Qualification:** to check the compliance with collected requirements.
8. **Deployment:** to go production in actual centers.
9. **Maintenance:** to apply any corrective or evolving actions on the finished software.

Step 1 was done during the proposal. Steps 2 and 3 were done after the proposal acceptance. Meetings were set up with the stakeholders. It was established that this medical center is but one part of a bigger planned system with a digital portal for a bigger organization, with the medical center solution being a smaller independent section. The software design was set then for multiple modules of a center; briefly:

- A pharmacy management module.
- Appointment management module.
- An event publisher module.

Steps 4 to 7 were done for each module (hence the Agile approach). Step 8 was undecided upon since there is no budget for deployment and no agreement for maintenance. (Details about deployment are laid out in later chapters.)

4. Report outline:

This report begins with an introductory chapter to get familiar with the project, the problems to be solved and the general approach taken. Then, Chapter 2 dives into the different requirements and constraints faced along with policies. Chapter 3 presents background research of existing solutions, then Chapter 4 details the development of the project and the steps taken to achieve the studied requirements. Finally, Chapter 5 presents tests of the results achieved.

CHAPTER II: PROJECT REQUIREMEMTS AND CONSTRAINTS

5. Introduction

This chapter details the project's different requirements from a technical and functional/business perspective. The planning of the project will be presented then a detailed listing of the requirements. On the other hand, the constraints and risks will also be presented. This chapter is a translation of the analysis and feasibility requirements phase of the SDLC, considering standards and policies.

6. Project planning

2.1. Team management (if applicable)

The team has two members that report to a supervisor as shown in the diagram below:

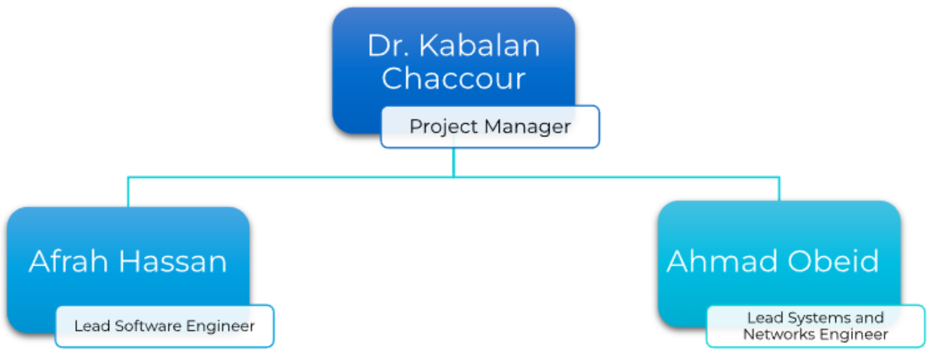


Figure 1 Team Structure Diagram

As shown, the various aspects of the project will benefit most from having a software and a systems engineer.

Team member	**Contributions**
Afrah Hassan	- Software architecture - Full stack knowledge - Coding

Ahmad Obeid	- Database administration/management - Server, cloud, and containerization skills - Coding

2.2. Time management

The work breakdown structure's (WBS) diagram is shown below:

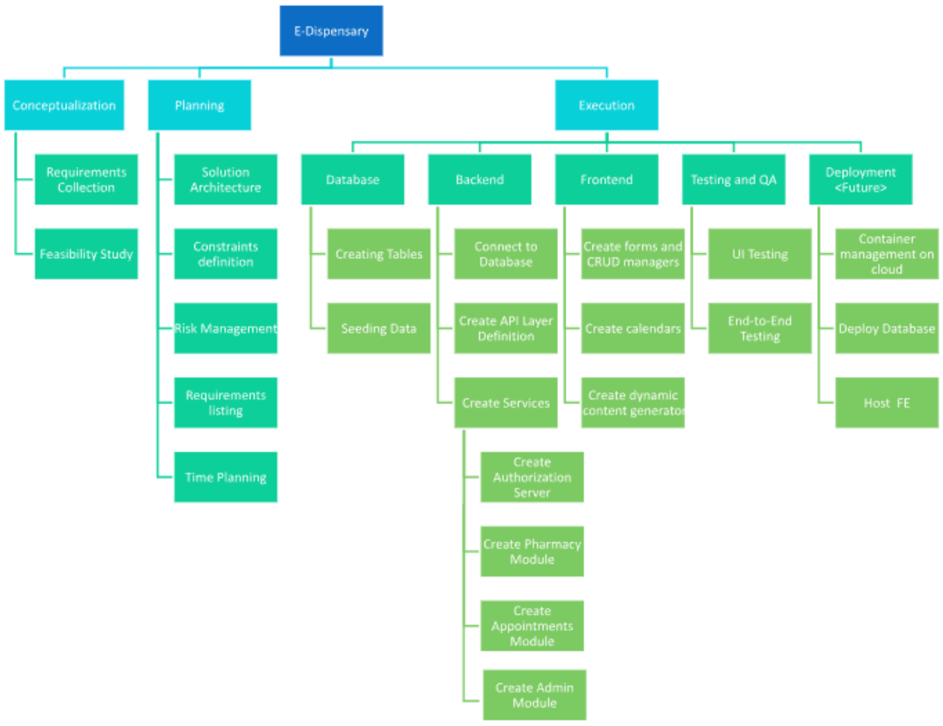


Figure 2 Work Breakdown Structure Diagram

2.3. Budget

There was no allocated budget for this project due to its charitable nature and as per stakeholder's request. This did not hinder development as all possible freely available resources were taken advantage of.

7. Project functional requirements

The project is a software product driven by user input and activity.
PS: In the table the word 'user' is defined for all users of the app. For users with a specific set of privileges, a role is assigned.

Table 1 Functional Requirements

User Story		Requirement
Authentication Module	As a user I want to log in to existing account	The system should verify existing credentials to log in via email/password or via third party
	As a user I want to reset my password	The system should send a verification link to reset the password
	As a patient I want to sign up	The system should collect basic information about patients and store them securely
Pharmacy Module	As a user I want to browse medicines	The system should show a paginated view of available medicines
	As a patient I want to browse medicines including mine	The system should show a paginated view of available medicines including the patients' bookmarked ones
	As a patient I want to reserve a medicine	The system should show allow reservations of a medicine
	As a nurse I want to manage purchases	The system should set a reserved medicine as bought.
	As a patient I want to view my purchase history	The system should show reservations of a user
	As a nurse I want to view all purchase history	The system should show all purchase information to the nurse
	As a nurse I want to add medicines	The system should let the assistant update medicines
Appointments Module	As doctor I want to set my availability for a specified amount of time	The system should manage availability dates during a clear set interval of time, during working hours of 9 to 5.
	As a doctor I want to see my availability on a calendar	The system should get corresponding dates by week, month (or year for supported calendars)
	As a patient I want to make an appointment	The system should let a patient make an appointment for a doctor during work hours and the Dr’s availability.

	As a doctor I want to see my patients and appointments	The system should let the Dr see his appointments with access to patients' information.
Events Module	As an admin I want to construct and publish events	The system should let the admin dynamically construct events with a simple UI.
	As a guest I want to see available events on landing page	The system should show constructed events.
Medical Records Module	As a patient I want access to medical records	The system should track information of appointments of patients
	As a doctor I want access to my patients' medical records	The system should allow access and update of medical records of a specific patient.
Admin related	The admin can turn on or off a backend service	System should include introspection of its own modules

8. Project constraints

4.1. Technical constraints

Due to the budget, there have been some technical constraints regarding the choices of technologies that can be used.

- Cloud can be used only partly and has been chosen to keep sensitive data (as seen in non-technical constraints)
- The logo colors hence theme colors already chosen.
- Scalable framework, so a stable one, for both front-end and backend.

4.2. Non-technical constraints

Examining the nature of the project and its applicability, the most prominent constraint is related to security and data protection. Data should be available to its owners only and to specified users with negligible risk of any breach.
As for impact, the project aims to have a positive social impact on the medical field and inspire further change of culture in how the local centers operate.
On one hand, the ethical side of data security can be ensured with the choice of technology, coding practices and preventive measures. As for the impact, however, it can only be quantified once the project is over.
In addition, the ease of use of the UI is to be highlighted, as the target audience is not the most familiar with technology.

4.3. Standards / codes / regulations / policies

As it is crucial to adhere to international standards and policies, the following were taken into consideration and/or applied:

- HIPAA as it related to medical data, particularly authorization and risk assessment.
- OWASP to access the used backend libraries
- SDLC as agile was followed.
- Ethical guidelines and code of conduct, when communicating with stakeholders and team members and choosing the technologies.

9. Conclusion

In conclusion, there are a set of constraints and defined requirements to work with next for this project. This chapter highlighted technical requirements, technical and non-technical constraints, and the regulations to be followed.

CHAPTER III: EXISITING SOLUTIONS

1. Introduction

This Chapter highlights the context and domain of applications and provides background research about related solutions along with a comparative table.
After reading this chapter, the reader should have an in-depth knowledge of the overall context and the domain of application and how the project relates to it, the related works and existing solutions, the problem to be solved and the objectives to be achieved.

2. Context and Domain of application

2.1 E-Health: What is it and what are its benefits

The digital transformation of this century has been affecting many, if not all, domains of human life. Healthcare has not fallen behind this necessary trend. Big health corporations across manty developed countries have digitized their infrastructure, making all their services electronic from patient monitoring, medical records, clinical interventions, and others.
The WHO defines e-health as:

"The cost-effective and secure use of information and communications technologies in support of health and health-related fields, including health care services, health surveillance, health literature, and health education, knowledge and research."

This field has become so important that WHO has put forth a global strategy for the next few years, aiming to make patients the center of the healthcare systems. This implies:

- Easy access to one's medical data from any place, at any time.
- Efficient governance of medical records and research that comes along with it.
- Minimizing the effort, it takes to get medical care.

The recent pandemic has also significantly increased the demand for e-health solutions.

In developed countries, e-health, particularly health information exchanges, are a top priority in improving their healthcare field. The digitization of medical records, registries, pharmacies are becoming the backbone of the healthcare sector, for good reasons. In fact, e-health increases the efficiency of its workers, reduces costs in the long run, and eases access to information. Not only that, but it also benefits the patient, by giving them trusted access to their information, exposure to reliable and helpful information.

This shift in methodology is resulting in a patient-centered mentality that has yet to reach countries and organizations on a smaller scale.

3. Existing Solutions /Methods

Different solutions appeared that tries to bring health services to the electronic age however, few presented an e-dispensary service, and even fewer in Lebanon. We will present the existing solutions and discuss the problems they target and their approach to bringing a solution to those problems. In addition, we will highlight their advantages and disadvantages in relation to our project proposal.

3.1. Solution 1: Ninsiima eDispensary

Ninsiima eDispensary is an existing service who has as objective to improve the access to health care services and their quality in Africa, and Tanzania specifically using available technologies. The problem this service is trying to solve is the hard-to-get health services in Africa. They offer a website where the user can choose between multiple options:

- Consultancy: Through which the user gets to chat via call or text with a medical professional.
- Book a doctor's visit: Through which the user can book a visit with a medical officer.
- Book Home Lab Test: Through the which the user can book medical laboratory test
- Subscribe to Palliative Care: Through which the user can subscribe to medical care for serious people with serious illnesses.
- Subscribe to Health Articles: Through which the user can subscribe to get health-related articles.

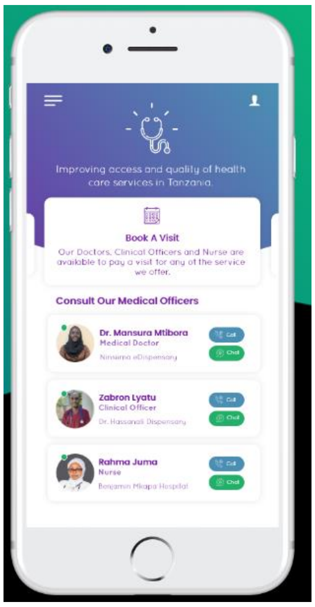
Those proposed solutions, while accessible from their website, are run from a business Whatsapp account. Choosing any option from the website will direct the user to the same Whatsapp chat. Considering the users this dispensary is trying to reach this method offers its users the simplicity of use. Chatting with a human on a popular chat application is a straightforward way of getting the services needed and especially when the user is not acknowledgeable with general technology.

On another note, this eDispensary lack a better integration with technology:

- The use of a chat app means that the service does utilize automation. The users must rely on an operator to provide them with the services they need and not a simple usable interface that can do provide the user needs with simple clicks. For example, the user cannot book an appointment by checking the available appointment times.

- The use of the chat app also means that the Ninsiima eDispensary does not keep track of user data efficiently. They must rely on traditional methods to keep their user data updated, stored, queried, and filtered according to their needs.
- Their system is still under improvement and further construction from 2020.

In addition to their general weak points, they also have shortcomings in terms of our project proposal, this service does not offer a news and events page, nor a way to look up available medications, nor a page to check medical officers' schedules.



Improving access and quality of health care services in Tanzania.

Book A Visit

Our Doctors, Clinical Officers and Nurse are available to pay a visit for any of the service we offer.

Consult Our Medical Officers

Dr. Mansura Mtibora
Medical Doctor
Ninsiima eDispensary

Zabron Lyatu
Clinical Officer
Dr. Hassanali Dispensary

Rahma Juma
Nurse
Benjamin Mkapa Hospital

Figure 3 Page from Ninsiima App

3.1.0. Solution 1: Ninsiima eDispensary

3.1.1. Method 1

3.2. Solution 2: DRAPP

DRAPP is a web-app and mobile-app for online health consultation. At its core, it is a messaging app that connects patients and healthcare professionals for on-the-go consultations. They offer a formal and secure platform alongside a chatbot to help both the patients and doctors for smooth interaction.

It is a simple concept, allowing the users to get a general view of the problems they have, account creation for patients is incredibly simple, and for doctors a specific and professional account setup. It offers notable features:

- <u>Appointment Booking</u>: By choosing a specific date for a doctor's appointment, with ability to reschedule or cancel. In addition, a reminder.
- <u>Choose and Find Doctors</u>: Through searching by field and location.
- <u>Video Conferencing</u>: Through which the online consultation happens.

While a great tech-based solution, it barely solves the problem. DRAPP said it themselves, “…the online consultation doesn’t replace the physical examination….” The chosen doctors are just users you know nothing about. The app is just a health-related chat application.

In addition, they also have shortcomings in terms of our project proposal. This service does not offer a news and events page, nor a way to look up available medications, nor a page to check medical officers' schedules. It is not an extension or an addition to a Dispensary.

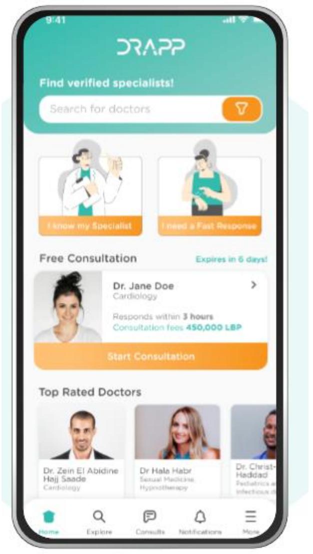


Figure 4 From Drapp App

3.3. Solution 3: Teladoc Health

Teladoc Health is the world's only integrated virtual care system. They deliver health related packages from wellness to complex healthcare needs. Teladoc offers plans for Employers, Health Plans, Hospitals & Health Systems, and Individuals. Since a dispensary is a Health Systems, we will discuss the case of Hospitals & Health Systems. They offer:

- Virtual Care: An easy to integrate service that merges virtual and in-person care to support both doctors and patients, while optimizing care delivery through automatically created insights to accelerate clinical decisions.
- Customer Care: To give patients access to on-demand or scheduled access to the care team.
- High-Acuity Care: To provide life-saving care when needed.
- Chronic Care: To coach patients and provide health plans to encourage them to live healthier

- Telehealth Devices: To create the perfect on-site environment for the best tech health solutions possible

Teladoc Health is the world leader in virtual care technology, and the solutions they proposed are extremely modern and useful in the sense it helps both the dispensary and the patients; however, it comes at an expensive cost.
When it comes to our project proposal, Teladoc Health is too complex for our dispensary use case situation; it is oriented more toward hospitals – patients' relation and not dispensary – patients.' It also does not offer a news and events page, nor a way to look up available medications.

3.4. Solution 4: Dispensary Management System for "Miyeh W Miyeh"

Dispensary Management System for "Miyeh W Miyeh" is a solution that consists of one part mobile-app and one part CMS webapp. The solution aims at providing a user-friendly dispensary platform for "Miyeh W Miyeh" village's dispensary in Lebanon.
This solution offers a good number of features for patients:

- User Login: To manage a profile for quick access to personal information.
- Appointment Booking: For users to book an appointment with a doctor
- Medicine Donation: To donate medicine to the other habitants in the village
- Blood Donation: To donate blood to the other habitants in the village
- Search for Doctors: Search for doctors depending on the field to book an appointment with.
- Search for Medicines: Search for medicines.

The solution also offers the ability for doctors to create an account.

The problem, however, appears in the CMS platform, as the administrator must actively and consistently manage the application and the local health market needs. Forms used to add doctors or events are not very well-organized, allowing for a lot of human errors. When publishing an event, with a lack of a dynamic aspect in the events UI.

3.5. Solution 4: Pharmacare – Pharmacy Software

Pharmacare is a pharmacy management software that has the goal to ensure a well-organized functioning, modern invoicing system revenue management, inventory track mechanism, and boosting up pharmacies' business.

It offers only CMS for Pharmacies:

- Medicine Management: To manage available and new medicines

- Manufacturing Management: To store manufacturers and providers information and buy stocks from the best suppliers.
- Purchase Management: For receipt printing and general purchasing information
- Reporting Functions: To monitor profits and losses
- Employee Information System: To manage employee information alongside salary and tax reports.
- Invoice System: Print invoices for customers
- Stock Management: For storage management
- Return Management: For returning perishable commodities.
- Accounts Management: To transfer money to supplier after customer payments

Pharmacare offers a complete Pharmacy CMS solution, coupled with a great graphical interface and general ease of use. The solution is, however, expensive for the customer highlighted for this final year project. This solution also does not offer any interaction with the patients, nor does it offer doctors or health professionals related solutions.

Thus, for this project proposal, this solution is not enough, as it only offers CMS and does not offer patient specific webapp.

4. Comparative study

After reviewing and discussing the different proposed solutions for the stated problem, by evaluating their features, weaknesses and specifically in the domain of our project proposal we can create a comparative table to summarize the key elements of each one of those solutions in comparison to our proposed solution.

Table 2 Comparative study of existing solutions

Solution	**Type of Solution**	**Features**	**Weaknesses**
Ninsiima eDispensary	eDispensary	Consultancy, through which the user gets to chat via call or text with a medical professional. Book a doctor's visit, through which the user can book a visit with a medical officer. Book Home Lab Test, through which the user can book medical laboratory test.	On another note, this eDispensary lack a better integration with technology: The use of a chat app means that the service does utilize technology to its full potential. Patients rely on human operators. They do not keep automatic track of user data.

		subscribe to Palliative Care, through which the user can subscribe to medical care for serious people with serious illnesses. Subscribe to Health Articles, through which the user can subscribe to get health related articles. Those proposed are ran from a business WhatsApp account allowing simplicity of use for the targeted audience.	The system is still under improvement and further construction from 2020. The service does not offer a news and events page, nor a way to look up available medications, nor a page to check medical officers' schedules.
DRAPP	Chat Mobile App	Appointment Booking by choosing a specific date for a doctor's appointment, with the ability to reschedule or cancel. In addition, a reminder. Choose and Find Doctors through searching by field and location. Video Conferencing through online consultation happens.	It barely solves the problem. The chosen doctors are just users you know nothing about. The app is just a health-related chat application. The service does not offer a news and events page, nor a way to look up available medications, nor a page to check medical officers' schedules. It is not an extension or an addition to a Dispensary.
Teladoc Health	Hospitals & Health Systems	Virtual Care service that merges virtual and in-person care to support both doctors and patients, while optimizing care delivery through automatically created insights to accelerate clinical decisions. Customer Care to give patients access to on-demand or scheduled access to the care team. High-Acuity Care to provide life-saving care when needed. Chronic Care to coach patients and provides health plans to encourage them to live healthier. Telehealth Devices to create the perfect on-site environment for the best tech health solutions possible	It comes at an expensive cost. It is too complex for our dispensary use case situation; it is oriented more toward hospitals – patients' relation and not dispensary – patients.' It also does not offer a news and events page, nor a way to look up available medications.

Dispensary Management System for “Miyeh W Miyeh”	Mobile App eDispensary with CMS	User Login to manage a profile for quick access to personal information. Appointment Booking for users to book an appointment with a doctor. Medicine Donation to donate medicine to the other inhabitants in the village. Blood Donation is to donate blood to the other inhabitants in the village. Search for Doctors depending on the field to book an appointment with. Search for Medicines.	The problem, however, appears in the CMS platform, as the administrator must actively and consistently manage the application and the local health market needs. Forms used to add doctors or events are not very well-organized, allowing for a lot of human errors. In addition, the data shown for the admin is not detailed enough. From the patient level, there is no webapp/website, and the user must always conduct his health-related problems from his phone.
Pharmacare – Pharmacy Software	Pharmacy Software	Medicine Management to manage medicines. Manufacturing Management to store manufacturers and providers information and buy stocks from the best suppliers. Purchase Management for receipt printing. Reporting Functions to monitor profits and losses. Employee Information System to manage employee information alongside salary and tax reports. Invoice System to print invoices for customers. Stock Management for storage management Return Management for returning perishable commodities. Accounts Management transfers money to supplier after customer payments.	The solution is expensive for the eDispensary. This solution also does not offer any interaction with the patients, nor does it offer any doctors or health professionals related solutions. It only offers CMS and does not offer patient specific webapp.

10.Problem Statement and Objectives

Medical centers of the smaller scale lack the digitized infrastructure to manage their activities and automate the tedious ones. Based on the background studies, current solutions are either too simple/targeted or too complex and suitable for bigger health organization. There is no middle ground that suits medical centers, especially Lebanese dispensaries.

This problem touches all the pillars of e-health, as mentioned previously. We need: a user-friendly solution to structure, organize and plan a dispensary's data and work in a secure manner.

The background studies presented challenges to keep in mind:

- Choosing cost-effective tools especially for the infrastructure
- Automating repetitive tasks like mails and FAQs
- Having an up-to-date client portal
- Managing the different actions that the roles can make

In addition, the nature of this project present two main challenges to keep in mind:

- Security: there are patient medical information in question
- Scaling: choosing the appropriate infrastructure to scale well in the future.

The efforts achieved thus far have only focused on one of the problems that medical centers face. They either cater very well for internal management or for the patient's care and experience.

Our solution aims to bridge this gap. Roles will be introduced on the level of the clients in an optimized webapp, whose content is managed by a back-office CMS.

The target users are:

Table 3 The distinct types of users and their roles

User type	Description	Goal
Guest	Public users visiting the webapp	● Increase exposure of medical center. ● Attract both potential donators and people in need.
Patient	Registered user benefiting from the services	● Benefit from programs, medicine, and doctors
Admin	Backoffice manager	● Manages events and news and has access on the whole scope of the app.
Nurse	Assistant	● Is online when requested by a user for information

		● Has write access on modules to assist doctors and manage pharmacy.

11.Conclusion

This chapter provided full insight on existing solutions from which, along with the laid-out constraints, the problem statement and objective were extracted. The existing solutions were listed, and the problem statement related to the context was extracted.

CHAPTER IV: PROPOSED SOLUTION AND STRUCTURE

(5 to 8 pages)
(on a separate page)

1. Introduction

This chapter details the structure of the proposed solution, highlighting the different modules it includes, along with the high-level software design.
The proposed solution is a multi-role web-application connected to a multi-service backend through REST (Representational State Transfer) calls. The authentication and critical data are delegated to a cloud service to improve security and simplify maintenance.

2. Design of the proposed solution

2.1. Solution architecture (project architecture)

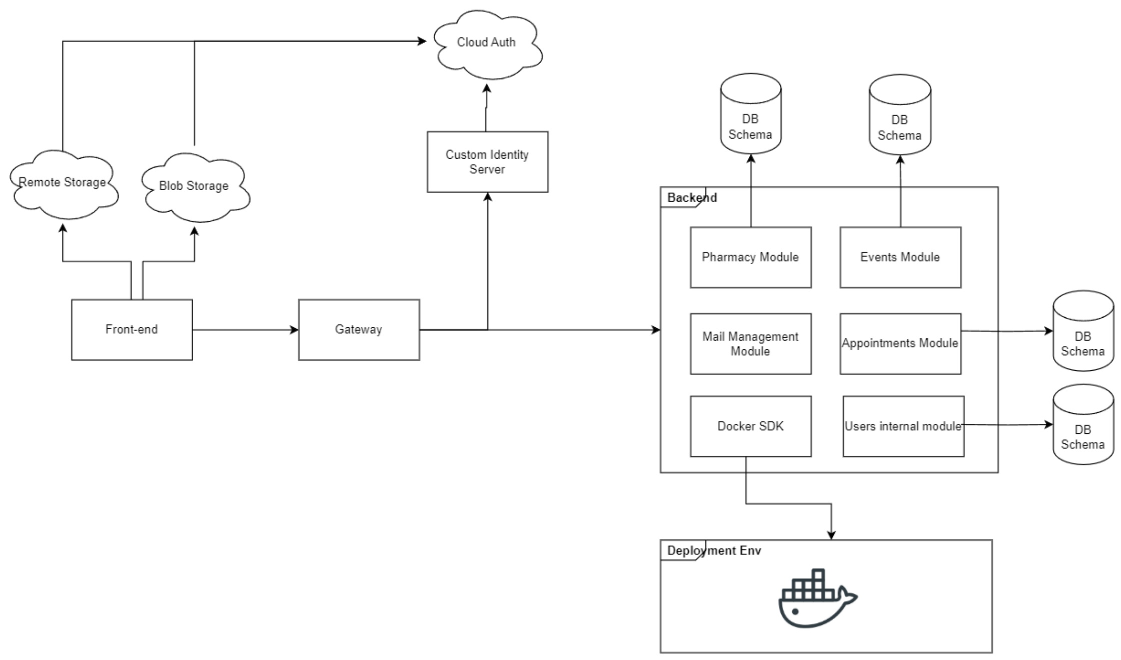


Figure 5 Diagram Showing High Level Structure of proposed solution

The diagram above shows the general structure. There are three high level layers:

- The frontend
- The gateway
- The backend

The front-end is the user interface. It takes its credentials from the cloud provider and utilizes cloud services for storage of some user data and blob storage. References of this remote data and the rest of the services are in the backend. The backend is reachable through the gateway, as the backend has no knowledge or decisions in authentication or role-based authorization and redirections. These are done on the level of the gateway. It checks the validity of the request credentials and role and redirects it to the appropriate backend module.
Each backend module has its own independent database schema. This approach applies the microservices architecture:

- It isolates the different services, reducing single point failures, so any issues that happen can be more easily located without affecting the rest of the system. So, if an issue arises in the pharmacy module and we needed to bring the service down for maintenance, the rest of the services would still be available.
- It offers scalability, so if there is more demand for one service multiple instances of it can be run to satisfy the demand without having to waste resources on the rest. Each service can also be modified independently.
- Continuous integration: as isolation and scalability are ensured, the deployment process is also streamlined for each service. Updates can be pushed faster, and new features done without messing with unrelated code.
- Language agnostic, which is something we have implemented as we will detail in the next chapter. Different features have unique needs, hence various technologies. In a monolith it is impossible to switch languages for a feature even if it will perform better, so it becomes restrictive.

The technologies chosen are detailed in the next chapter.

2.2. Internal code architecture

A multi-tier structure was followed both on the front-end and the backend.
Each backend module included:

- An infrastructure layer handling connections to the database and any needed request.
- A data access layer for the repositories and ORM (object relational mapper) configuration.
- A business logic layer isolated by mediators.
- A presentation layer housing the REST endpoints.

As seen below

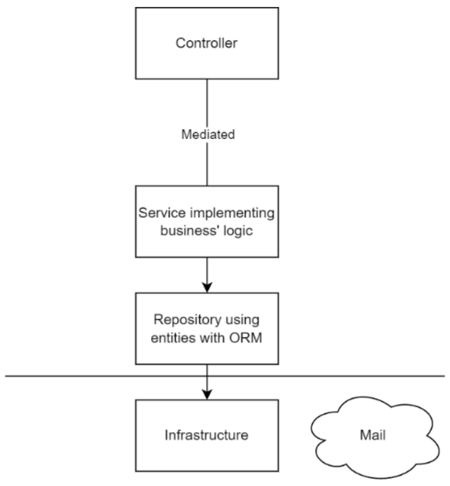

Controller
Mediated
Service implementing business' logic
Repository using entities with ORM
Infrastructure
Mail

Figure 6 Tiers of one endpoint flow

The figure above shows the different layers one endpoint passes through from top to bottom. Controllers are mere presentational layers that only delegate the request internally through mediators. The appropriate service injected will handle the business logic of the request and handle both system errors and expected user errors, making sure the system does not crash. Any data exchange must happen on a data access layer, as business logic stays unaware of the persistence medium, applying dependency inversion principle. The infrastructure layer handles barebone connections to the system or any third-party connections. In this project it has been used to communicate with the Docker API in an independent module.

3. Use case diagrams (Design Blocks)

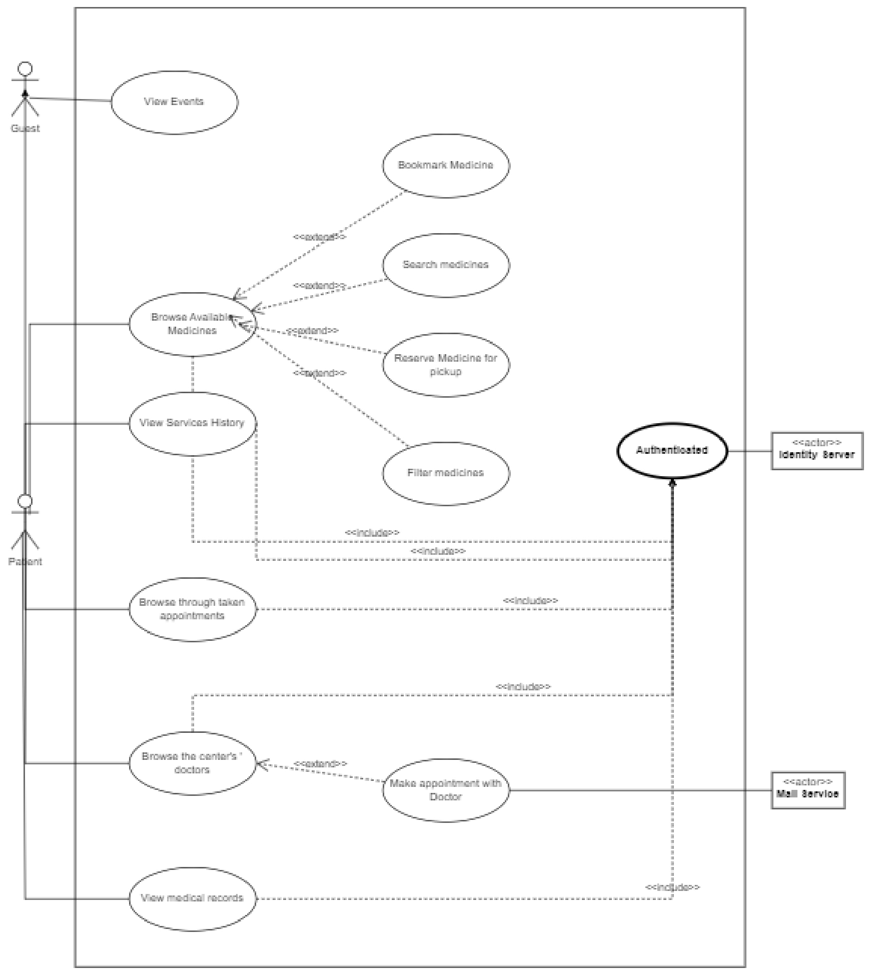
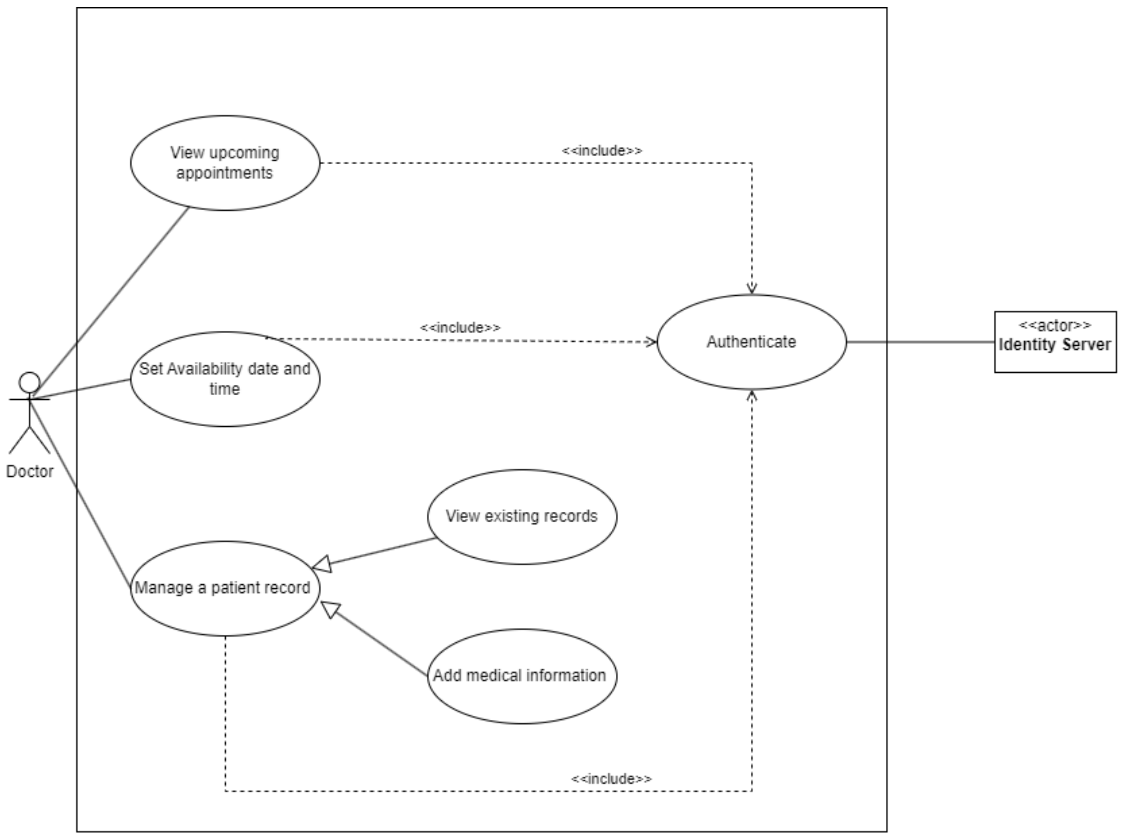


Figure 7 UCD of the patient role

Above is the UCD – use case diagram representing the translation of the collected functional requirements of the patient. Aside from making appointments, the user has more of a 'read' access to the services, as all the managerial tasks will be done by the nurse for the medicines, and doctors for their medical information. Secondary actors include mail services and the identity server.

As for the doctor:



The doctor's main objective is patient care that must be managed through appointments. To enable appointments

12.Conclusion

This chapter represented the analysis and design phase of the SDLC process through the high level design of the different blocks needed and the use case diagrams.

CHAPTER V: DEVELOPMENT AND IMPLEMENTATION

(10 to 15 pages)

(on a separate page)

1. Introduction

This chapter details the implementation of the project, the general structure will be shown following an explanation of the software and technologies used throughout the different components of the system and how it aided development, along with the cloud-based tools used surrounding the implemented system. The database design will be shown as well for the different services. While explaining technologies, their main usages and challenges were included as well.

2. Software development and implementation

2.1 System overview

2.2 Technologies used

We have chosen the technologies first based on the uses cases that we have. We need frameworks that facilitate our work without having unnecessary complications or inferior performance. The next criteria are how well it is supported in the current market. There would not be a point to all the implemented software architecture and design principles if the framework is deprecated or difficult to scale and maintain. A framework that has been around for enough time to have good support, and still well maintained to include new features and paradigms is needed. Lastly it would help, if both previous criteria were met, if we were familiar with the chosen technology, to prevent unexpected issues during the learning curve. This does not mean that we did not learn new skills and significantly broadened our technical horizons.
With all that in mind, the following was chosen:

Table 4 Chosen technology (tools, languages, frameworks for each need)

Needed technology	Purpose	Decision
Database	Storing generated user data	Postgres
	Storing sensitive user data	Firestore
Authentication service	Authenticating users and admins and supported extension for role-based access control	Firebase Authentication

File storage service	Storing static images from users	Firebase Cloud Storage
API gateway	Reverse proxy to handle requests to multiple backend	NGINX
Deployment	Containerization tool to handle local and future remote deployments	Docker
Front-end farmwork	Enterprise level, easy to use user interface with rich components	Angular
Backend frameworks	User related business logic	Java Spring Boot
	Internal tooling logic	Python FastAPI
	Firebase admin tooling	Python FastAPI

2.2.1 The front-end

Angular is a front-end framework by Google that is robust and scalable and used for enterprise level user interfaces. It is characterized by its solid internal code structure, following object-oriented programming and its four pillars, a technical advantage rarely seen in client-side frameworks. Angular provides many component frameworks to facilitate the development of UI components and unify design decisions like theming across the app.

As our app needs connection to Firebase to fetch access tokens in addition to using that token to validate it on the level of our gateway as well, efficient handling of dependent HTTP requests is needed. Angular includes out of the box native integration with a framework called RxJs (also known as reactive JavaScript), that interfaces with the browser through an event loop to add a-synchronicity to an otherwise single threaded language. With RxJs the requests made are more reliable.

For the UI, a framework called NgZorro was used. It follows the ‘Ant Design System’ and is widely used in enterprise level web applications.

For future reference, Angular also supports server-side rendering and mobile integration.

We made several modules extending on the recommended Angular architecture. All the modules are ‘Lazy Loaded;’ thus, they only load once the route first calls, improving the first-load time. In addition, an auth verification layer was also added to each route through ‘Route Guards’ that verify the existence of a token and role matching[1]. The architecture decoupled the UI code from the data fetching and formatting code.

2.2.2 The back-end frameworks

Starting with the main chosen framework which is Java Spring Boot. With minimal configuration, Spring boot enables the multi-tier flow mentioned previously that is most useful in small services. The biggest advantage is Spring Boot’s ORM, hibernate. Hibernate eliminates boilerplate code and links each defined entity to a repository ready to use and easy to extend through a generic interface. For example, for our Medicine entity we create a repository interface only inheriting Hibernate’s generic interface that has built-in CRUD (Create Read Update Delete) functions. Hibernate heavily uses code introspection in the repositories that it is not needed to write any additional SQL. For more complex queries like the appointments module that includes data and time management, some queries had to be built but also seamlessly through the interface, without having to manage database connection or sanitization of data.
In addition, Spring boot offers seamless dependency management through Maven and its pom.xml file. Spring boot handles the infrastructure layer through a resource file.

On the other hand, internal tools were also needed:

- Mail service
- Docker SDK (Software Design Kit) service
- Firebase Auth extension for roles

These tools are not user oriented, rather they include helper tasks that each has its independent purpose. Frameworks like Spring boot will only overcomplicate their development. A scripting language with good libraries is needed in this case, hence Python with FastAPI was used. Python has countless reliable libraries, and its framework FastAPI offers a simple and flexible REST (representational state transfer) integration.

2.2.2.1 The mail service

We want to notify a user when they have a near appointment. Patients can schedule appointments a week or two earlier, and they might not be visiting their profile often, so they need to be notified at least the day before the appointment. For that a scheduler has been created as a separate module that periodically checks the databases for near appointments and sends an email to the appropriate user. As all user info is in firestore, this module gets the email through the ID in firestore.

[1] This is only for the front-end and is independent from backend security.

2.2.3 Firebase services

Firebase was used because it is free and has a simple learning curve. We created a Firebase project on the console. To enable a front-end to use the provided authentication service we had to register the application and get the credentials. Those were registered in the Angular application's environment through the 'Angular Fire' library. The features provided through the front-end flows included signing in and signing up with custom username and password as well as with Google.
For any blob data needed like images we needed a cloud service to handle them efficiently thus the Firebase storage service.

2.2.3.4 The authentication flows

One major constraint is that the default given flow does not include custom tokens. And for user signups we need to indicate that they have a 'Patient' role, not other roles that are given by the admin. Thus, a middleware service was made with Python. It took admin credentials from the firebase console as well. Below a diagram showing the different

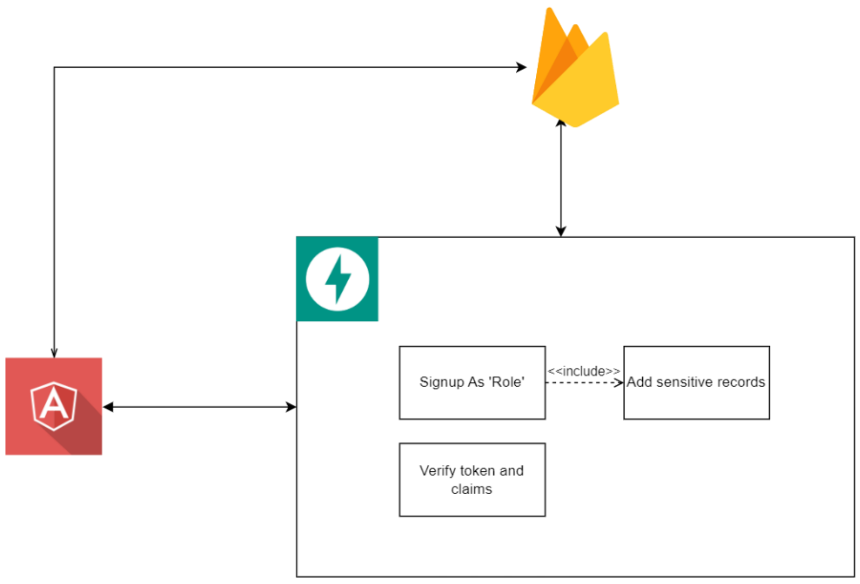


Figure 8 Authentication flow overview (High level view of components and functionality)

The figure above shows the different components that took part in the customized authentication flow. To dive deeper, below is a sequence diagram to clarify:

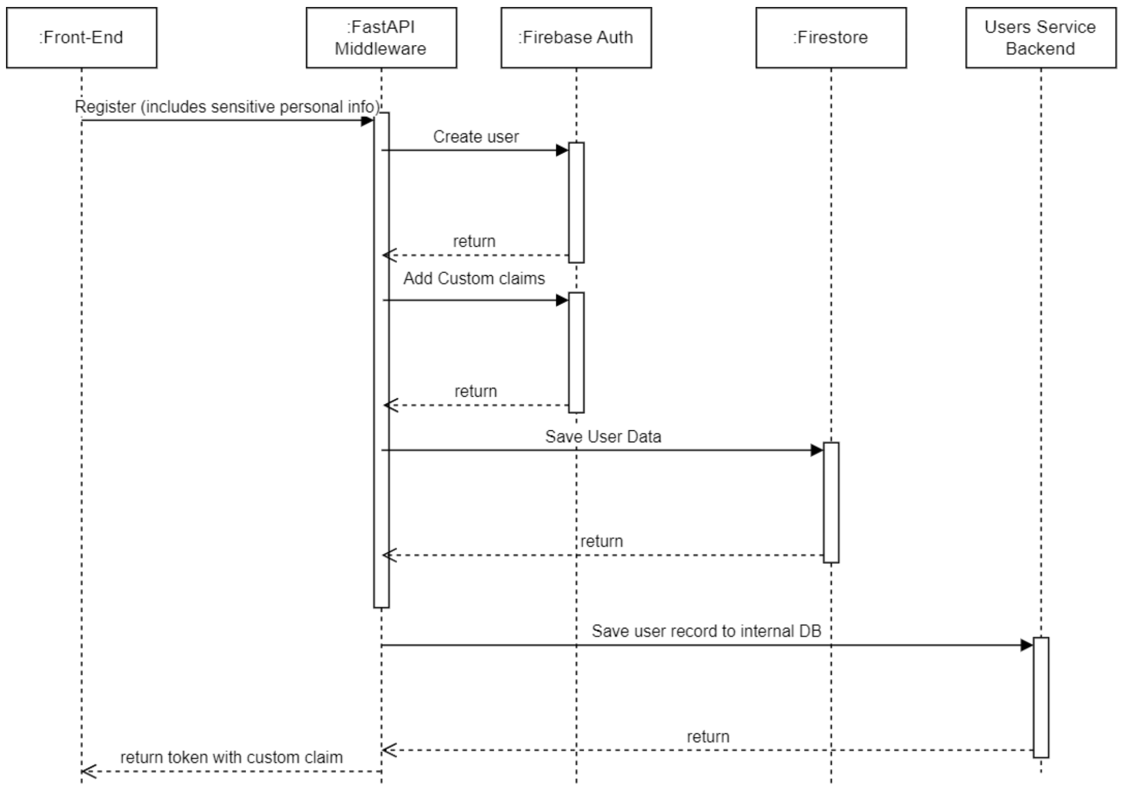


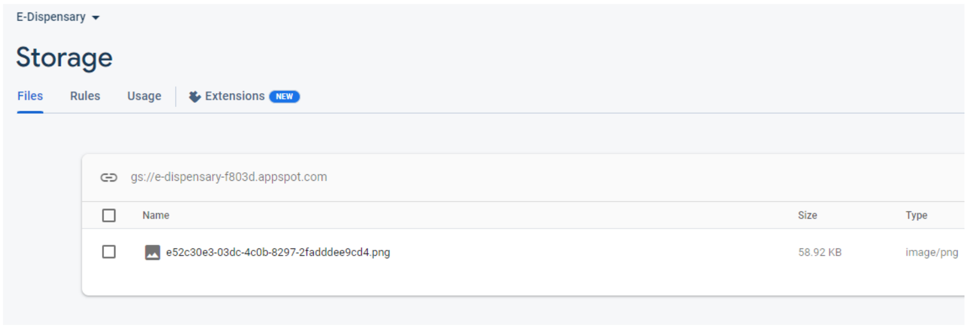
Figure 9 Auth sequence diagram (Sequence diagram showing the different calls needed for signup with classic user and password)

The sequence diagram above shows the steps took to create a new user and save its credentials and sensitive data using extended Firebase authentication and FireStore. The front-end sends the credentials along with profile data to the custom backend to extend authentication which, in turn, authenticates with firebase and sends the data to the FireStore. The mentioned credentials can be either a username and password or a Google account token.
While it would have been possible to the front-end to send information to FireStore directly, having it to one backend call causes less errors, in addition it stays as one transaction ensuring atomicity.
The rest of the authentication processes and additional FireStore calls are done with the default front-end flow provided by firebase as the token is now included in the request.

2.2.3.5 The storage services

Firebase provides its own configuration language to secure access to its database. For the created collection access rules has been set:

- The user must be authenticated
- The role must be set
- If the role is patient only the patient’s own data is accessible, checked through ID.



gs://e-dispensary-f803d.appspot.com

Name	Size	Type
e52c30e3-03dc-4c0b-8297-2fadddee9cd4.png	58.92 KB	image/png

2.2.4 NGINX

Nginx is an open-source web server and reverse proxy server. It is often used as a web server to serve static content like HTML, CSS, JavaScript files, and images. Additionally, Nginx is commonly used as a reverse proxy server to distribute incoming traffic to multiple backend servers, like application servers or other web servers.

We have set up a nginx server as a reverse proxy to connect our frontend to the multitude of backends available and notably our authentication server.

Nginx offers multitude of modules, one of those modules is the 'ngx_http_auth_request_module' module. The module implements client authorization based on the result of a sub-request. If the sub-request returns a 2xx response code (successful range), access is allowed. If it returns 401 or 403, the access is denied with the corresponding error code. Any other response code returned by the sub-request is considered an error.

Thus, the reverse proxy is used twice, once for authentication by taking the JWT token and verifying it in the auth server, and then if the authentication was successful the request will pass to the needed backend module.

The other module used is the 'ngx_http_core_module' module. This module contains many of the well-known established directives, like the server directive to setup the different servers in the nginx configuration, but in addition, the 'error_page' directive which was used to bypass nginx default error redirection and replace with our custom JSON error messages returned from the auth server. With this, we can create a reverse proxy with authentication capabilities and custom error redirecting.

It is important to note that nginx was used over other well supported options like Apache due to its robust security.

Refer to Appendix C to see sample code from nginx configuration.

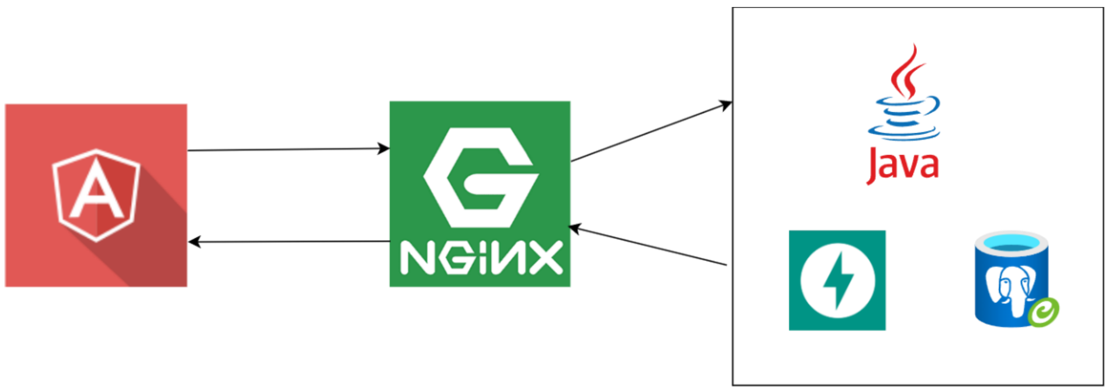


Figure 10 Nginx Reverse proxy

2.2.5 Docker

Docker is a containerization tool that facilitates code shipping and provides an isolated environment for development and production.
Each service developed has its own Dockerfile where dependencies are indicated for build and full environment set up. The services are registered in a 'docker-compose.yaml' file where they are in one internal network with only the needed ports exposed for security, and any external data needed mapped as Docker Volumes; otherwise, data is lost on container restarts as each container acts like a sandbox.
Our DockerSDK API manages containers programmatically through endpoints to start, stop and restart containers and view their status, that way the admin can stop services on command with zero technical knowledge needed.

3. Database structure

Internal data needed for the dispensary was stored in a PostgreSQL database. Each module has its own schema. For data that needs any relation to other schema the ID was used, and indexes created accordingly.
In the schemas below there is a duplication of users and roles. These have been added for representation only and live in their own 'General' schema.

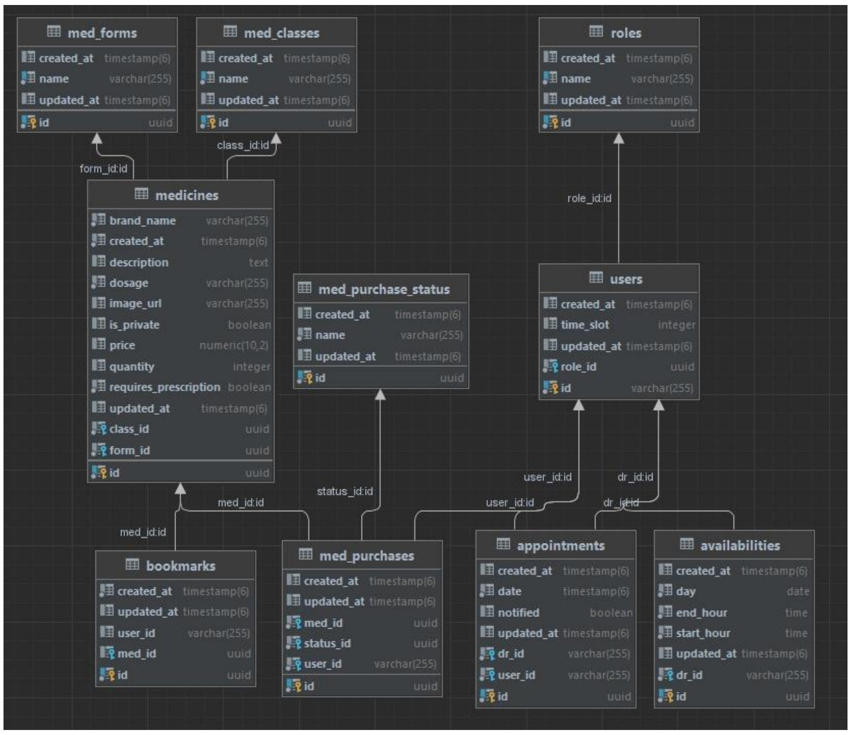


Figure 11 Pharmacy DB

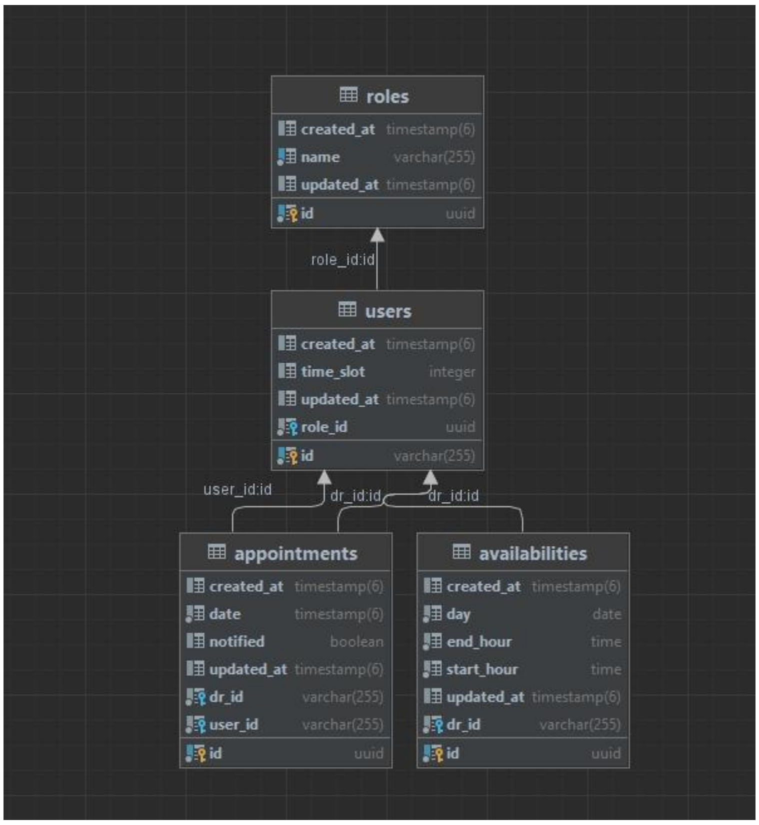


Figure 12 Appointments DB

The users table only includes the id, optional time slot (for users with role ‘doctor’). All sensitive data is in firestore. The ID generated by Firebase on signup acts as the source of truth for each user when it concerns created firestore documents for the user as well as the internal schema. For reference check the previous sequence diagram.

Below is the patient’s collection in FireStore.

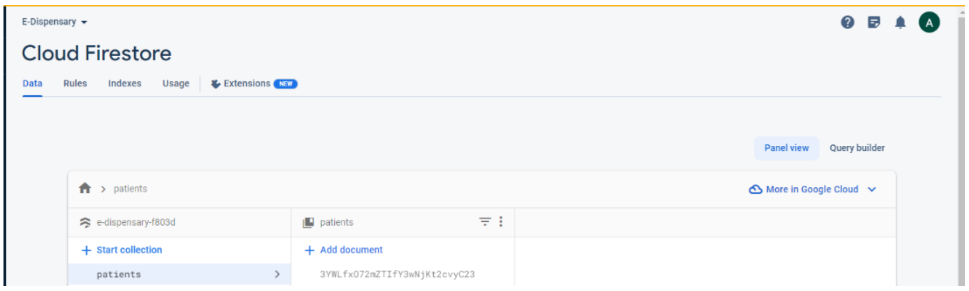


Figure 13 FireStore collection

3.1 Working with SQL vs No-SQL

Two different database paradigms had to be used: a traditional structural database with PostgreSQL, (note that PostgrSQL supports No-SQL features but none were used) and document based No-SQL database which is FireStore.
For the SQL, data had to be formatted into a DTO (data transfer object) before being sent out on the front-end, however the document-based data was simply read and sent as it is already structured like JSON, so the flexibility of NoSQL allowed for simpler and faster read access. However, NoSQL was only used for simpler discrete sets of data where complex relationships are not needed, as then it will fail to match the efficiency of SQL.

4. Conclusion

This chapter details the implementation of the different services. It went over each major component, explaining the technology and how it was used to apply the requirement within the mentioned module. The webapp was implemented through different services, each having their schema. Helping tools such as the notification module, Docker SDK module and auth extension modules were also needed in addition to the main modules implementing functional features.

CHAPTER VI: EXPERIMENTS AND RESULTS

(5 to 10 pages)

(on a separate page)

- Slow connection tests (show networks tab) (only firebase needed a second)
- Appointments not in time
- Diff roles (403)
- The debounce on reserve button

1. Introduction

This chapter includes the experimentation process done on the web app. The testing approach took the project features and user flows from end to end to simulate the user going through it.

2. Prototype/ application/

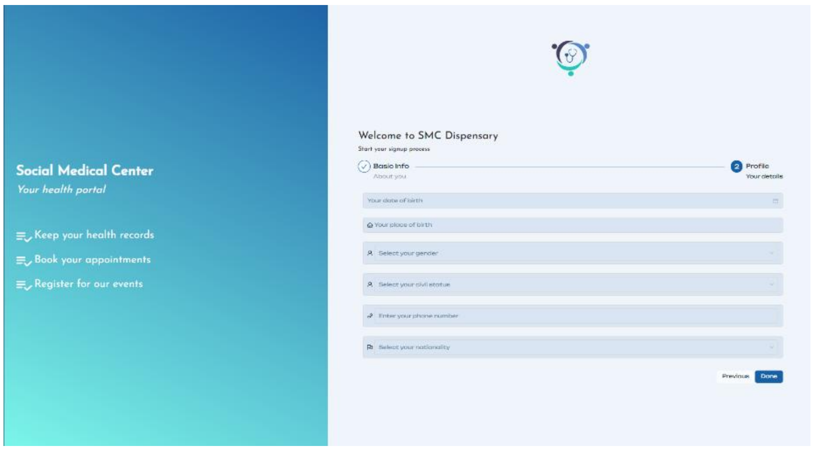


Figure 14 Sign up stepper (page in developed solution)

3. System tests/simulations/experiments

3. Testing slow connection to back-end and firebase

To simulate deployment on a different server and slow connections such as cellular or Ogero, the requests were throttled to see how well the system handles slow requests.

3.1.1. Test scenario (setup, users, constraints, parameters, variables, metrics)

- Frequent pages were chosen such as the medicines grid view and the login section to simulate real life usage with slow internet (common scenario in Lebanon).
- Any user using a wireless connection can experience slow connection.
- The browser's network tab 'Throttling' option was used to set up:

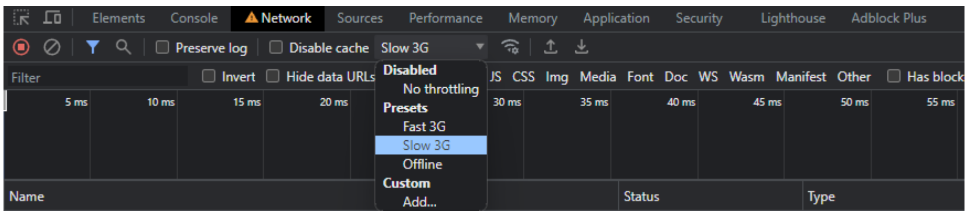


Figure 15 Throttling connection (throttling in browser)

It is expected to have a loading indicator until any data is received.

3.1.2. Test result

Table 5 Test result (connection throttling)

Time to load	10 seconds
Front-end result	The front-end kept the loading indicator
Error handling	The requests were tried to each other since RxJs library was used

3.1.3. Test interpretation

The interface kept a skeleton indicator the whole time of the request. The indicator is a pulsing skeleton, which gives a better user experience than a progress spinner.

3.1.4. Discussion

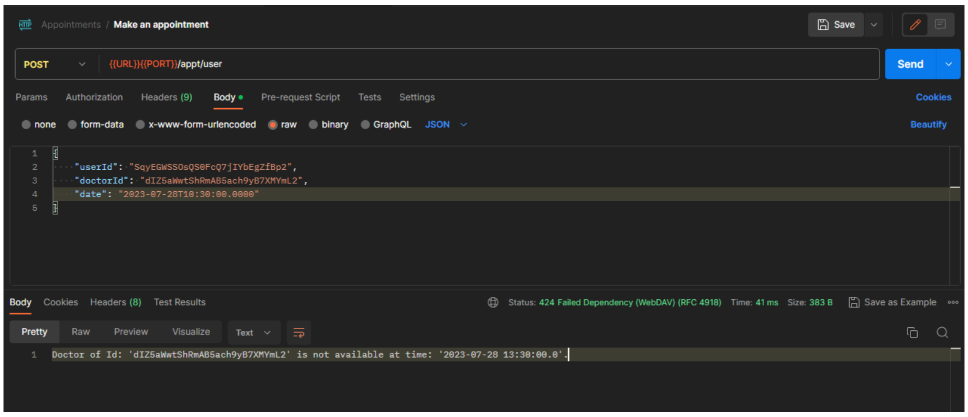
The usage of the RxJs library permitted efficient handling of errors and timeouts of HTTP (Hyper Text Transfer Protocol) requests, especially for the features that require more than one request to be handled (those that include 'front-end only' Firebase requests like authentication).

3. General E2E tests.

The most prominent test method throughout the project is E2E – end-to-end testing.

3.2.1. Test scenario: Appointments and Availability

In Postman, an API (Application programming interface) testing tool requests can be made with needed tokens and payloads. During development, each endpoint was tested. For the demonstration, the endpoint to make an appointment was tested.



```
POST {{URL}}{{PORT}}/appt/user
```

```
{
    "userId": "SqyEGWSSOsQS0FcQ7jIYbEgZfBp2",
    "doctorId": "dIZ5aWwtShRmAB5ach9yB7XMYmL2",
    "date": "2023-07-28T10:30:00.0000"
}
```

```
Status: 424 Failed Dependency (WebDAV) (RFC 4918)   Time: 41 ms   Size: 383 B
```

```
Doctor of Id: 'dIZ5aWwtShRmAB5ach9yB7XMYmL2' is not available at time: '2023-07-28 13:30:00.0'.
```

Figure 16 Postman test (sad path of endpoint)

As seen above the 'POST' endpoint was tested, user was trying to make an appointment at a time where there is no availability for the Doctor. The body includes the User Id, Doctor Id, and date.

3.2.2. Test result

As seen in the previous figure the case was handled accordingly.

3.2.3. Test interpretation

The system handles unexpected user flows with appointments:

- It checks if there is registered availability of the Doctor at the requested time.
- It checks if the requested time conflicts with another existing appointment.

3.2.3.1. Interpretation of end-to-end tests

E2E tests evaluate the whole flow of an endpoint/application. It is closest to a real-world scenario. E2E simulates the user's perspective whether tried from the UI or backend only, so it makes sure that the whole pipeline of services databases and network are tested for functionality and performance since they are tested through whole endpoints. This increases confidence in software and provides faster detection of defects.

3.2.4. Discussion

3.2.4.1. Discussion on end-to-end tests

While E2E tests provide many advantages and are cheaper to implement, a larger team would benefit from more types of tests, especially if maintenance and future support is to be included. E2E tests are done on a flow that is fully finished thus individual pieces of code were not tested and new features' integration are not verified against the existing codebase.
During testing the project's features, errors were clear, but they were related to different layers of the code, so sometimes substantial changes had to be made to fulfill requirements, including a refactoring of some services, changes that take a lot of precious development time.
A TDD – or Test-Driven Development (Ref- Appendix B) approach would have been beneficial especially if future maintenance or handover is expected. But such a drastic development approach change would require a higher budget due to the CI/CD runners needed, a larger team to handle the QA and the DevOps parts, as well as a longer timeline. So, this should be kept in mind if the project is to be deployed and scaled in the future.

As for the front-end, there are tools to help automate their tests, such as 'Storybook,' which helps build standalone testable components in isolation so they can be tested on their own. In the project manual front-end testing was used as a form of E2E tests, however the code structure still prioritized standalone components as they are re-usable and easier to maintain.

4. Auth Server Test

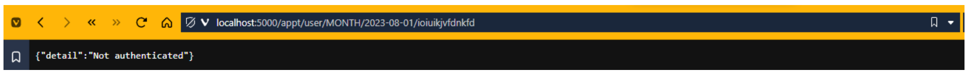
3.2.5. Test scenario: Testing Authentication

Each request needs to have a valid bearer token that includes the role.
A GET request was tested directly in the browser with no token included.

Table 6 Endpoint details (testing of an authenticated endpoint)

Request URL	***http://localhost:5000/appt/user/MONTH/2023-08-01/ioiuikjvfdnkfd***
Request description	Getting the appointment of a user.
Privileges needed	The authenticated user or an authenticated nurse.

3.2.6. Test result


localhost:5000/appt/user/MONTH/2023-08-01/ioiuikjvfdnkfd

```
{"detail":"Not authenticated"}
```

Figure 17 Test Result (unauthorized HTTP call)

As seen in the figure above no information was fetched.

Noting that on the front-end the page that loads the request does not show unless the user has privilege. It is a front-end level security that reflects the main backend security.

5. Impact of the proposed solution

This project aims to improve the health sector in Lebanon, in particular small medical centers.

- Positive impacts
  - Ease of access to medical services, reducing the people's need to commute, improving quality of life.
  - A new improvement on an outdated management approach that is not too complex to be overwhelming.
- Negative impacts
  - Even if appropriate measures were taken, this does not stop people with bad intentions from trying to cause breaches and steal data. For a project with important data to be deployed in production, it needs continuous maintenance and monitoring.
  - While it is not a short-term impact, going fully digital might be a problem for the elderly, as they still need assistance in electronics. Elderly assistance and accessibility were not tackled in this project.
- Local impact
  - It is hoped that this project, if small, might attract the attention of people in power to fund and improve such initiatives.

The people are the direct benefactors of this solution.

6. Conclusion

Tests aside, it was ensured that this phase of the project is functional and is a good milestone to future features and hopefully a scaled project that will change the health sector for the better. Both the front-end and back-end code were written with flexibility and scalability in mind. The backend has independent modules, so even if a drastic change was required such as a change in programming language (an extreme example) only a well-defined section must be re-written and that does not affect the infrastructure as everything is shipped in containers.

General formatting guidelines

A- Figures

- Figures should be placed in the middle of the page
- Titles of figures should be placed at the bottom of the figure in the middle
- Figure titles should be formatted as follows:

FIGURE DETAILS

Figure #: Title (caption) (size = 11)

- Figures should be cited in the text to indicate. Example:
    - Figure # illustrates the general solution architecture.
    - The graphs in Figures 5, and 7 represents the results of our simulation

B- Tables

- Tables should be placed in the middle of the page
- Titles of tables should be placed at the top of the figure to the left
- Tables titles should be formatted as below

Table #: Table title (caption)

Title (bold)	**Parameter 1**			
Feature 1	Values			

- Tables should be cited in the text to indicate. Example:
    - Table 1 shows a comparative study of the existing solutions

C- Text

- Plain text should be written in Times New Roman, size 12
- Interline spacing = 1.15
- Text should be justified

D- Margins

- Respect the margins of this template through the whole report

E- Titles of paragraphs

- Titles of paragraphs, sections, and subsections must follow the below rule:

1. Section 1 title

1.1. Subsection 1 title

1.2. Subsection 2 title

1.2.1. Subsubsection 1 title

1.2.2. Subsubsection 2 title

1.3. Subsection 3 title

2. Section 2 title

Etc.

F- Acronyms

- Acronyms should be detailed the first time they appear in the text

CONCLUSION

In conclusion, our proposed software project addresses a critical issue prevalent in Lebanon's healthcare landscape – the lack of a comprehensive digital infrastructure in the realm of e-health. This web application is a multi-role solution targeting medical centers, aiming to bridge the gap between traditional manual processes and modern digital advancements. The obtained web application is a comprehensible and easy user interface servicing patient to view their data and appointments, doctors to help with their schedule, nurses to assist doctors and manage the pharmacy, medication administration and patient care along with an administrative capability to dynamically create and manage events. Ethics were followed, security was ensured and tested along with end-to-end tests simulating real life scenarios like the misuse of the application. The latest containerization tools have been utilized, alongside a management API that enhances the application's scalability and maintainability. However, different decisions could have been made concerning the technologies, while Spring Boot checked all the boxes with its efficiency, ORM and diverse features, it was not the most developer friendly for the appointments' module. If new features were needed in the future another module could be done as a plugin using easier technology, an advantage easily implemented due to our architecture.

For future work, the admin module can be further extended to include data analysis on the application's data with more graphs and recommendations. In addition, multi-language support can be added for the front-end. The front-end's architecture already decouples the interface specific code from the rest of the HTTP services so adding the languages as JSON (JavaScript Object Notation) is not as challenging, the translations would have to be provided.
However, it must be highlighted that no matter how advanced the application gets, it would be useless without knowledgeable users. The patients of most medical centers include the elderly, poorer thus uneducated people, making the transition to fully online services slow, if not impossible in the short term.
This should not stop innovation however, as this application could be made into SAAS - software as a service, fully configurable for any medical center with the possibility of customization.

Our web application represents a significant step towards transforming the landscape of e-health in Lebanon. By addressing the dearth of digital infrastructure, we empower medical centers to optimize their operations, enhance patient care, and foster an interconnected healthcare ecosystem.

REFERENCES

The bibliographic references or sources (e.g., articles, journals, patents, books, webpages, etc.) that you examined during the preparation of your FYP proposal can be easily added and cited in MS-Word.

- https://www.insights10.com/report/lebanon-digital-health-market-analysis/
- https://www.telecomreview.com/articles/exclusive-interviews/2229-new-milestone-towards-e-health-in-lebanon

Refer to the Full FYP Proposal Template for the steps of how to insert the references

// TODO

LIST OF ACRONYMS

SDLC: Software Development Life Cycle.
SRS: Software Requirement Specification.
UML: Unified Modelling Language.
WBS: Work Breakdown Structure.
UI: User Interface
REST: Representational State Transfer.
HTTP: Hyper Text Transfer Protocol.
JSON: JavaScript Object Notation.
ORM: Object Relational Mapper.
UCD: Use Case Diagram.
TDD: Test-Driven Development.
E2E: End to End.
API: Application Programming Interface.
SAAS: Software As A Service
CRUD: Create Read Update Delete

LIST OF FIGURES

Provides a list of all the figure titles in the report for direct access and their page number

LIST OF TABLES

Provides a list of all the table titles in the report for direct access and their page number

APPENDICES

APPENDIX A

Agile Methodology

Agile is a project management and software development approach that emphasizes flexibility, collaboration, and iterative progress. It was originally formulated for software development but has since been adopted in other fields. The Agile methodology prioritizes customer satisfaction, adaptive planning, and continuous improvement. It contrasts with traditional, linear project management approaches by promoting flexibility, rapid iterations, and a close working relationship between cross-functional teams.

Noticeable advantages of agile for this project are:

1. **Flexibility and Adaptability**: Agile projects are divided into smaller, manageable iterations known as sprints. This allows project teams to adjust project requirements and priorities based on changing market conditions, customer feedback, or emerging opportunities. Thus, Agile embraces change as a natural part of the development process.

2. **Customer-Centric Approach**: Agile methodology places a strong emphasis on engaging customers throughout the project lifecycle. Regular interactions and feedback loops ensure that the product matches expectations.
3. **Improved Product Quality**: Agile promotes continuous testing, integration, and validation of the product. Thus, any errors can be detected before going to production.
4. **Risk Mitigation:** Agile's incremental approach mitigates project risks by breaking down complex steps into manageable units (milestones). This makes the project predictable

In software, Agile is implemented through many approaches depending on the size of teams, and, for companies as well depending on the technical decisions. Examples are Scrum, ‘Xtreme’ agile and Kanban.
Kanban is a simple visual management approach that places ‘Work Items’ on a board with labels ‘Todo,’ ‘Doing,’ ‘Done’ (and optionally more). This helps visualize the load on the team and quantify efforts.

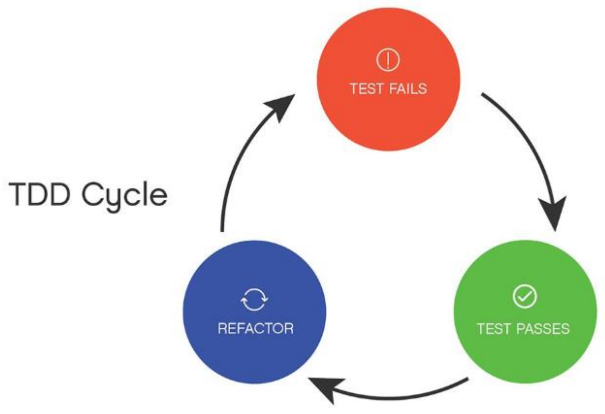
APPENDIX B
TDD Methodology

TDD methodology falls under the umbrella of Agile methodology as it goes hand in hand with the dynamic nature of the software market and its market centric profit.
TDD takes an evolutionary approach of writing the test case before writing a piece of code. And, that test must fail first then the code created (or refactored, thus considering recent changes), before that code is permitted to pass into production.
The benefits of TDD include:

- Higher productivity since the use cases are well understood by the developer before they start coding them.
- Less bugs after production since the code passes a more-than-average amount of tests.
- Automated tests are very precise and cover a wide variety of test cases.
- Code is well documented

In contrast there are some disadvantages to be taken into consideration:

- Functional tests cannot be well translated with TDD, which is used most in front-end development
- A steep learning curve is required to get into TDD.
- Tests are also codes, which must be maintained and updated throughout the project's lifetime, requiring more work hours.



Provides essential supplementary information for the comprehension of some sections of the report

APPENDIX C
Nginx Code

auth_request:

```
location / {
        ...
        auth_request /auth;
        ...
}

location /auth {
        ...
        proxy_pass http://auth_server;
        ...
}
```

As we can see in the example above, this is a simple auth_request configuration. auth_request uses the normal proxy_pass directive for reverse proxy.

error_page:

```
location / {
        ...
        auth_request /auth;
        error_page 500 404 403 @auth_error;
        ...
}

location /auth {
        internal;
        proxy_pass http://authserver;
        ...
}

location @auth_error {
     internal;
     proxy_pass http://authserver;
        ...
}
```

In the example above, we added the error_page directive to the auth_request directive. The error_page directive would intercept 500 404 and 403 errors and take the error message from the @auth_error internal location we created. That location would then communicate to the auth server to get the error message.