Health Care

NETWORK AND COMPUTER SECURITY

28 December 2021 | Campus Alameda - Group 26

THE TEAM:

Ana Albuquerque
ist1102209

André Proença
ist1102327

Joel Russo
Ist1102098

1. Problem

Health care institutions gather and store sensitive information from patients. The information systems should allow fine-grained and contextualized access to the records to the relevant staff.
One of the relevant types of data stored are test results. Some of the medical tests can be performed inside a hospital lab, but in many cases, tests are done in partner labs, that have a distinct infrastructure, remote from the infrastructure of the hospital, that need to be interconnected. In addition, the privacy clause is a key issue for safe and successful access to patient health information. Current approaches do not always provide patients with the ability to establish appropriate rules for accessing their information in a secure manner.

A patient's medical records are extremely sensitive data. The historical records made by doctors facilitate the process of diagnosing a patient, ensuring their quality, which helps clinical staff to treatquickly and accordingly.
This data should be kept private, allowing only the discriminating staff to access it. We believe thatall healthcare facilities should have access to this type of information so that patients can receive healthcare anywhere and at any time. Therefore, the data should be protected from external agents(i.e., outside the medical institutions) and from unauthorized people within the institutions.

This report aims to demonstrate a secure system that allows secure and confidential access to patient medical records of certain healthcare organizations. It also demonstrates the secure interconnection (sending and receiving of medical records) of partner healthcare organizations. In such manner, a patient will be able to access his/her medical records in every healthcare organization where he/she is registered.

1.1. Solution Requirements

Client application requirements:

As a <u>user</u> (depending on my privilege) I am able to...

- Read my medical records (send requests to the system);
- Receive responses from the system (receive replies from the system).
- Change my system password

As a <u>Doctor</u> and <u>Nurse</u>, I am able to...

- Create/Read/update medical records.

As a <u>Patient Service Assistant</u> and <u>Porter</u>, I am able to…

- Read specific information about which patient is assigned to me.

As a <u>Ward Clerk</u> I am able to…

- Register a patient in the system.

As a Patient I am able to…

- Read my medical records.

As the System Administrator, I am able to…

- Create new personnel within the organization.

Server application requirements:

The System…

- Ensures confidentiality and integrity of medical records;
- Ensures confidentiality and integrity of communications with the web application;
- Ensure successful authentication of citizens;
- Authenticates citizens in a secure way;
- Ability to change password if citizen is authenticated;
- Authenticated user credentials confirmation sent to email;
- Ensures that only authorized staff and patients have an account;
- Ensures different "roles" have access to different privileges;
- Ensures there is only one account per citizen, using citizen ID card number;
- Prevents access to medical records if the citizen does not have privileges;
- Allows user A to change the privileges of user B, if user A is a system administrator;
- Validates and sanitize form input;
- Uses HTTPS to encrypt communications;
- Stores and manage symmetric and asymmetric keys;
- Defines a restricted set of rules on the firewall;
- Establishes mutual authentication and shares medical records with partner institutions;
- Minimizes the impact of attacks inside the system.

2. Proposed solution

In order to simulate real systems and their interconnection, our solution is based on the developmentof two systems representing healthcare institutions. A hospital and a partner laboratory. The goal is to have two completely independent and functional healthcare institutions, each with its own data storage system and independent web platform, and simultaneously simulate the sending of confidential patient's medical records in a secure way from one institution to the other. Authorized hospital and laboratory staff as well as patients will be able to access their respective hospital and laboratory remotely or locally.

2.1. Overview

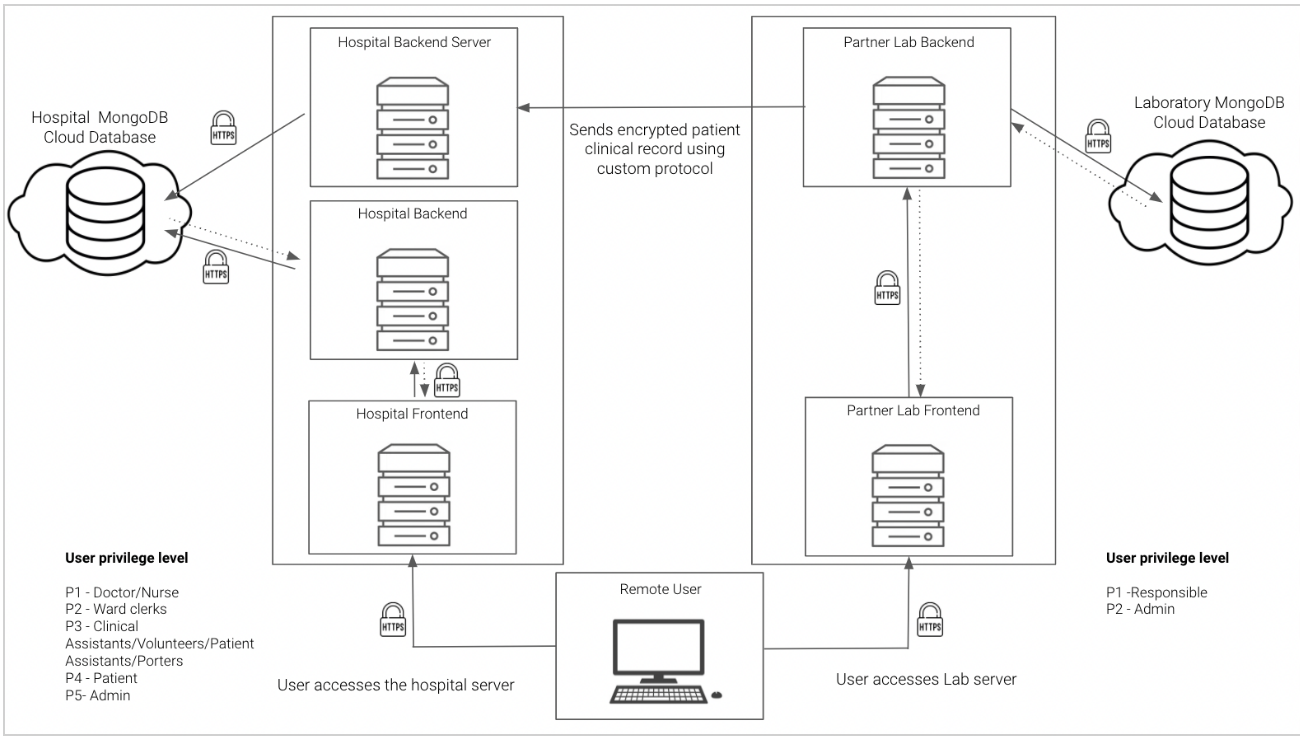


Figure 1 General Architecture.

Spring Security, Security policy language

Spring Security is a powerful and highly customizable authentication and access-control framework. It is the de-facto standard for securing Spring-based applications.
Spring Security is a framework that focuses on providing both authentication and authorization to Java applications. It has features like Comprehensive and extensible support for both Authentication and Authorization. Protection against attacks like session fixation, clickjacking, cross site request forgery, etc…

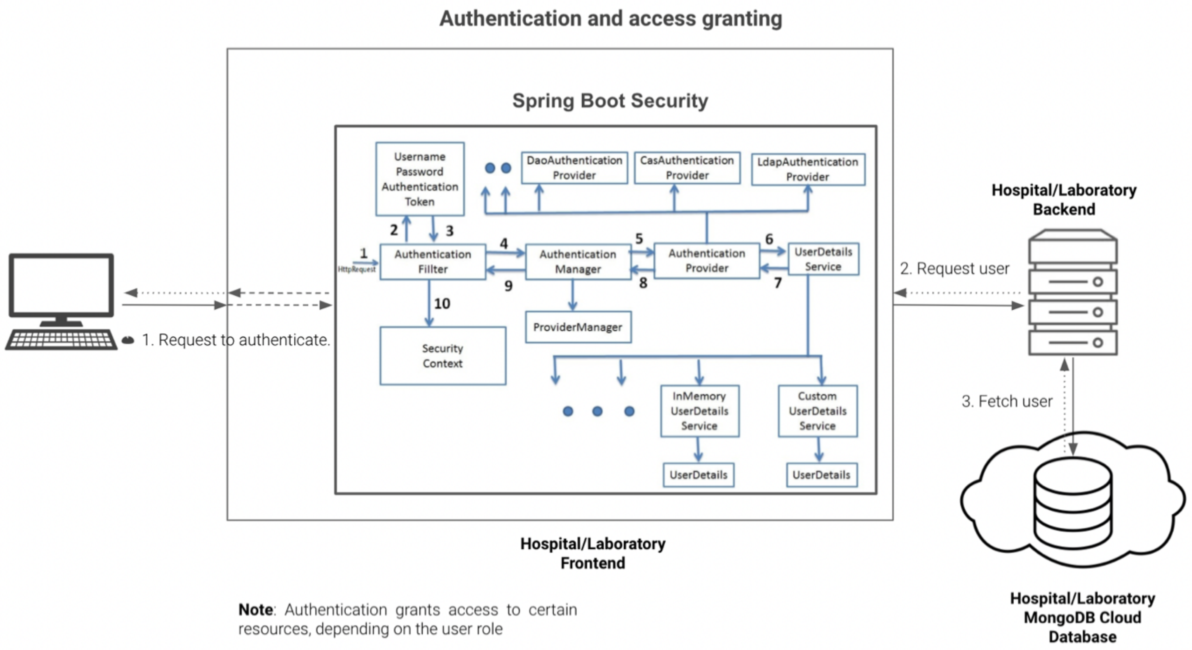

Authentication and access granting
Spring Boot Security
Username Password Authentication Token
DaoAuthentication Provider
CasAuthentication Provider
LdapAuthentication Provider
1
HttpRequest
2
3
4
5
6
7
8
9
10
Authentication Filter
Authentication Manager
Authentication Provider
UserDetails Service
ProviderManager
Security Context
InMemory UserDetails Service
Custom UserDetails Service
UserDetails
UserDetails
1. Request to authenticate.
Hospital/Laboratory Backend
2. Request user
3. Fetch user
Hospital/Laboratory Frontend
Hospital/Laboratory MongoDB Cloud Database
Note: Authentication grants access to certain resources, depending on the user role

Figure 2 Authentication and access granting.

2.2. Deployment

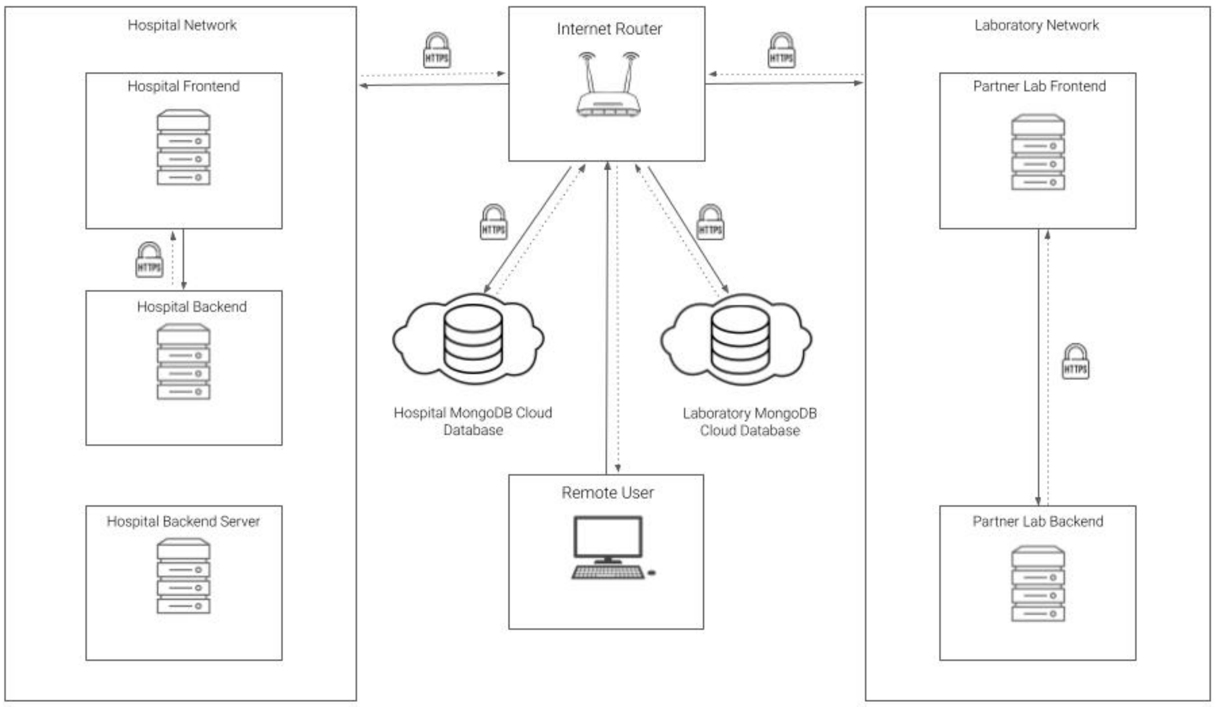


Figure 3 Network Architecture.

As you can see, 6 virtual machines running Ubuntu Server and the router running Seed Ubuntu are deployed.

- 1 for the router that will simulate the internet;
- 1 for Hospital Server, which will receive the clinical records from Partner Lab Backend;
- 2 for the hospital and laboratory respective frontends;
- 2 for the hospital and laboratory respective backends;
- 1 for local or remote users which will access the frontend servers. To change the location of users on the network, just change the properties of the VM in the hypervisor (VirtualBox, for example). This way we avoid creating multiple VM's for each local and remote user.

For the configuration of the each vm, the hostname, host, credentials, network were configured. For each machine in the laboratory and hospital networks, a static ip was set. The patient is connected via DHCP to the NAT Network.
For the router, there were added the hospital network, laboratory network and NAT adapters, so that the communication could flow between the three interfaces.

2.3. Firewall Rules

Router

For router firewall rules, the following ufw rules were used in order to simplify the iptables:

- sudo ufw route allow in on enp0s3 out to enp0s8
- sudo ufw route allow in on enp0s3 out to enp0s9
- sudo ufw route allow in on enp0s8 out to enp0s3
- sudo ufw route allow in on enp0s8 out to enp0s9
- sudo ufw route allow in on enp0s9 out to enp0s3
- sudo ufw route allow in on enp0s9 out to enp0s8
- sudo ufw enable # ativar a firewall

The following changes in /etc/ufw/sysctl.conf were uncommented in order to allow for forwading:

- net/ipv4/ip_forward=1
- net/ipv6/conf/default/forwarding=1
- net/ipv6/conf/all/forwarding=1

With this changes we were able to forward packets between interfaces.

Hospital Frontend and Laboratory frontend

The following ufw rules were created:

- sudo ufw enable
- sudo ufw allow 8443

Hospital Backend and Laboratory Backend

The following ufw rules were created:

- sudo ufw enable
- sudo ufw allow from 10.x.0.4 to any proto tcp port 3000

The IP 10.0.0.4 is the IP from Hospital Backend, for Laboratory Frontend the IP is 10.100.0.4.

Hospital Backend server

The following ufw rules were created:

- sudo ufw enable
- sudo ufw allow from 10.100.0.5 to any proto tcp port 4000

The IP Laboratory Backend the IP is 10.100.0.5.

2.4. Jar files

Auto running of jars

In order to auto run the jars on startup a cronjob was created.
Run crontab -e, put in file:
@reboot /path/to/bash-script

The script contains the following code so that the jar runs in the background:
nohup java -jar /path/to/jar &

3 Secure channels configured

Who is communicating?

- A user accesses any of the health institution platforms through the HTTPS protocol, thus encrypting the communication and making it secure. Each of the health institutions has a frontend that authenticates a user according to his/her "role", processes, validates, sanitizes form data and makes requests to the api of the respective backend institution to authenticate users, or to fetch a user or a medical/clinical specific information, which then makes a request to the respective database. The connections to the databases, as well as the connections from frontends to backends, use the HTTPS protocol, thus ensuring content integrity and confidentiality.

SSL certificates

- By default, the backends connect to their databases via HTTPS. SSL is configured in the backends application.properties. For the frontends and backends, self-signed certificates were created with the help of the keytool. To establish the HTTPS connection were also configured the application.properties files in the frontends and backends of both institutions.

4 Secure custom protocol developed

Who is communicating?

- Each time a partner Laboratory responsible creates a clinical record, the record is automatically sent to the hospital in a secure and confidential way. To establish this connection, it is necessary to ensure a mutual authentication by both the laboratory and the hospital server;
- When the connection is first requested, the hospital server and the laboratory must both authenticate each other and exchange a symmetric key in order to encrypt the messages that will be sent;
- After the exchange of the symmetric key, the laboratory makes a request to the hospital server, asking if the patient exists in the hospital's database. If the patient exists, the clinical record will be sent and then registered in the hospital's database.

Which keys will exist and how will they be distributed?

- Each Health Institution is a certifying entity that issues its own certificate. Each of theseinstitutions have an asymmetric key pair, stored in a keystore, which will be used to communicate and establish secure connections and in a truststore the public key of the other service, this is, the hospital backend server has the public key of the laboratory backend. Symmetric keys are generated when establishing a connection between laboratory backend server and hospital server;
- When the connection is first requested by the laboratory, the hospital server sends its signed certificate, so the laboratory can verify its legitimacy. The second step is laboratory is to verify the server legitimacy, in order to do this, the server sends a random string to the laboratory, so that the laboratory can sign that string with its private key and the server can verify the laboratory authenticity by decrypting the same random string using the laboratory public key;

- If the authentication is successful, the laboratory creates a symmetric key that is sent to the laboratory, encrypted with its public key. This key is generated so that the communication is done without overexposing the private keys;
- The symmetric key is temporary and is changed for every medical record that is sent to the hospital server.

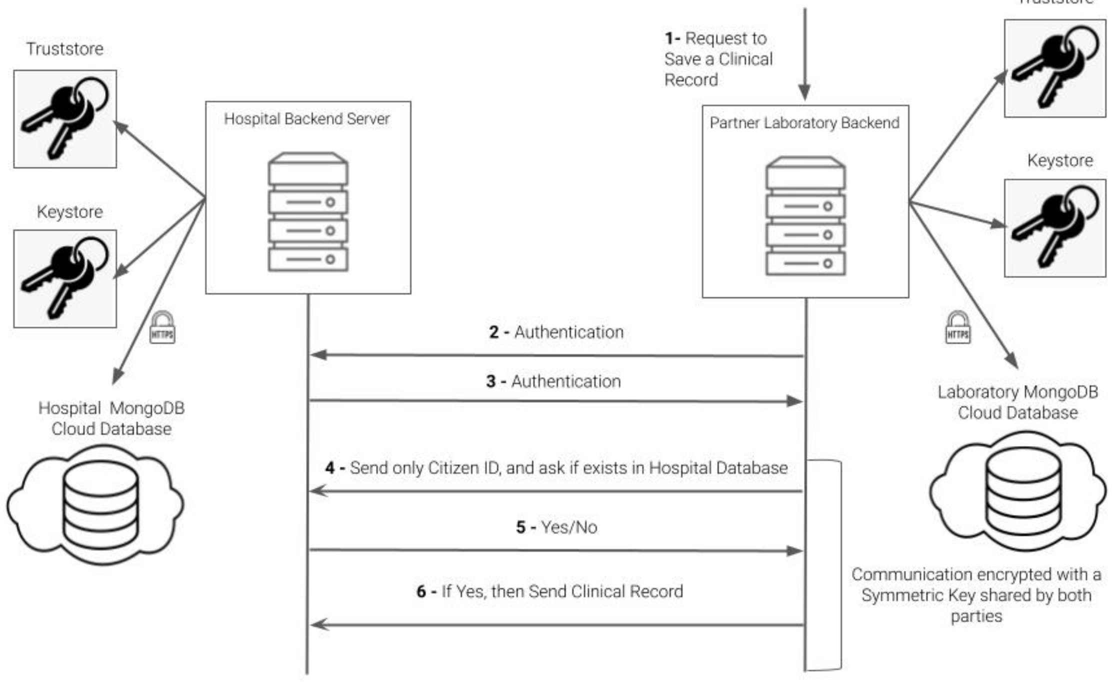

Truststore
Keystore
Hospital MongoDB Cloud Database
Hospital Backend Server
1- Request to Save a Clinical Record
Partner Laboratory Backend
Truststore
Keystore
Laboratory MongoDB Cloud Database
2 - Authentication
3 - Authentication
4 - Send only Citizen ID, and ask if exists in Hospital Database
5 - Yes/No
6 - If Yes, then Send Clinical Record
Communication encrypted with a Symmetric Key shared by both parties

Figure 4 Communication between hospital server and laboratory

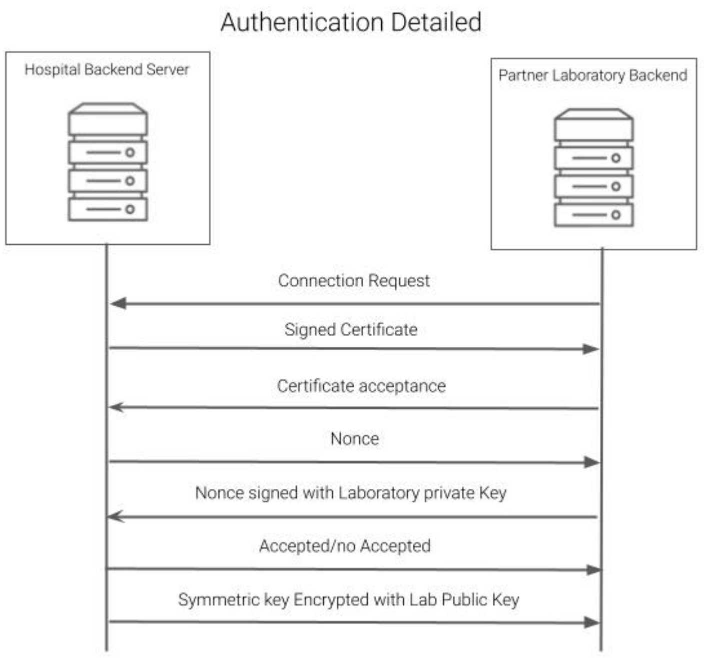


Figure 5 Authentication and exchange of symmetric key

Security properties ensured:

- The Protocol ensures integrity, confidentiality.

5 Secure custom protocol developed

- Frontend: Java Spring Boot
- Backend: Java Spring Boot
- Database: MongoDB
- Security policy language: Spring Boot Security
- Virtual Machines: Ubuntu Server, Seed Ubuntu
- Firewall: UFW (Uncomplicated firewall)

6 Results

6.1 Milestones

Basic Version:

- Design system, set up the login method in Health Institutions frontend servers, set up andconfigure the security policy language (Spring Boot Security)

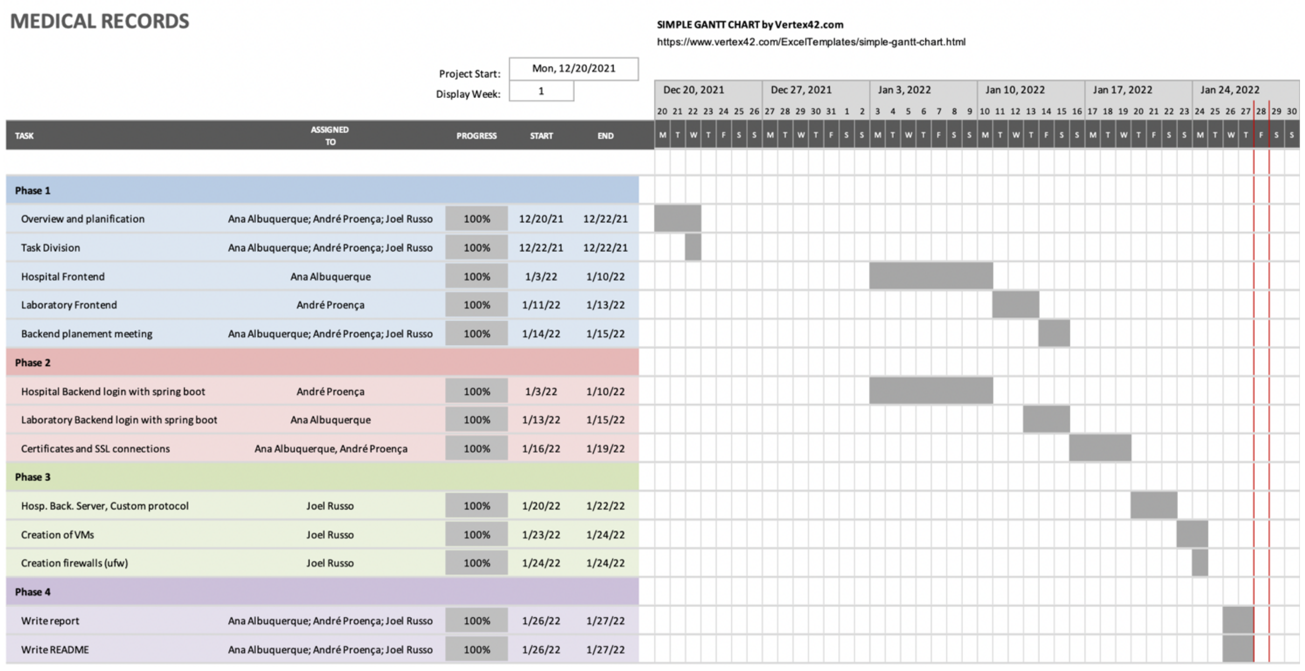
Intermediate Version:

- Design and implement the secure custom protocol. Ensure secure communication via TLS, Use asymmetric cryptography to authenticate the connection establishment and symmetric cryptography to guarantee integrity and confidentiality in communication.

Advanced Version:

- Backup servers.

7 Effort Commitments



MEDICAL RECORDS

SIMPLE GANTT CHART by Vertex42.com
https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html

Project Start: Mon, 12/20/2021
Display Week: 1

TASK	ASSIGNED TO	PROGRESS	START	END
Phase 1				
Overview and planification	Ana Albuquerque; André Proença; Joel Russo	100%	12/20/21	12/22/21
Task Division	Ana Albuquerque; André Proença; Joel Russo	100%	12/22/21	12/22/21
Hospital Frontend	Ana Albuquerque	100%	1/3/22	1/10/22
Laboratory Frontend	André Proença	100%	1/11/22	1/13/22
Backend planement meeting	Ana Albuquerque; André Proença; Joel Russo	100%	1/14/22	1/15/22
Phase 2				
Hospital Backend login with spring boot	André Proença	100%	1/3/22	1/10/22
Laboratory Backend login with spring boot	Ana Albuquerque	100%	1/13/22	1/15/22
Certificates and SSL connections	Ana Albuquerque, André Proença	100%	1/16/22	1/19/22
Phase 3				
Hosp. Back. Server, Custom protocol	Joel Russo	100%	1/20/22	1/22/22
Creation of VMs	Joel Russo	100%	1/23/22	1/24/22
Creation firewalls (ufw)	Joel Russo	100%	1/24/22	1/24/22
Phase 4				
Write report	Ana Albuquerque; André Proença; Joel Russo	100%	1/26/22	1/27/22
Write README	Ana Albuquerque; André Proença; Joel Russo	100%	1/26/22	1/27/22

8 References

1. Spring Quickstart Guide. (2022). Spring. https://spring.io/quickstart
2. Building REST services with Spring. (2022). Spring. https://spring.io/guides/tutorials/rest/
3. Securing a Web Application. (2022). Spring. https://spring.io/guides/gs/securing-web/
4. Understand Spring Security Architecture and implement Spring Boot Security | JavaInUse. (2022). https://www.javainuse.com/webseries/spring-security-jwt/chap3
5. RestTemplate (Spring Framework 5.3.15 API). (2022). Spring. https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/client/RestTemplate.html
6. Validating Form Input. (2022). Spring. https://spring.io/guides/gs/validating-form-input/
7. KeyStore (Java Platform SE 7). (2020, June 24). Oracle. https://docs.oracle.com/javase/7/docs/api/java/security/KeyStore.html
8. To Use keytool to Create a Server Certificate (The Java EE 6 Tutorial). (2022). Oracle. https://docs.oracle.com/cd/E19798-01/821-1841/gjrgy/
9. Krout, E. (2021, August 20). How to Configure a Firewall with UFW. Linode Guides & Tutorials. https://www.linode.com/docs/guides/configure-firewall-with-ufw/
10. Get Server | Download. (2022). Ubuntu. https://ubuntu.com/download/server
11. Tutorial: Thymeleaf + Spring. (2018). Thymeleaf. https://www.thymeleaf.org/doc/tutorials/3.0/thymeleafspring.html
12. To Use keytool to Create a Server Certificate (The Java EE 6 Tutorial). (2010). Docs.Oracle.Com. https://docs.oracle.com/cd/E19798-01/821-1841/gjrgy/