

SOEN 6471
ADVANCE SOFTWARE ARCHITECTURE
(SUMMER-1 2022)

Deliverable 1

VAXSYS

Declaration

We, the members of the team, have read and understood the Fairness Protocol and the Communal Work Protocol, and agree to abide by the policies therein, without any exception, under any circumstances, whatsoever.

TEAM-G

BARIQ ISHTIAQ MOHAMMED	40208194
MANTHAN PARESHBHAI MORADIYA	40156072
SHRAVYA NALLA	40162273
SIDDHARTHA NANDA	40200496
PARTH NAVSARIWALA	40178800

Table of Contents

1. VISION	6
1.1 Overview	6
1.2 Business Goals	6
2. STAKEHOLDERS AND CONCERNS	9
2.1 Mind map of stakeholders	9
2.2 Stakeholders and Concerns	10
2.2.1 Group of Stakeholders	10
2.2.2 Owner	10
2.2.3 End User	10
2.2.4 Software Provider Organization	12
2.2.5 Customer Support	16
3. Problems	17
3.1 Non-Functional Requirements	17
3.2 Functional Requirement	20
4. VIEWPOINTS AND VIEWS	25
4.1 Use Case View	26
4.2 Process View	28
4.3 Implementation View	30
4.4 Logical View	32
4.4.1 Related viewpoints	32
4.5 Deployment View	34
5. ARCHITECTURAL DECISIONS	35
5.1 Design Practices	35
5.1.1 Keep Modules Small	35
5.1.2 Include only concerned functions in any module	35
5.1.3 No Redundant Functionality	36
5.1.4 Module reusability	36
5.1.5 Testing	36
5.1.6 Standard Namespacing, conventions, and Coding styles	37
5.1.7 Indentation, Coding Style, Documentation	37
5.2 Design Principles	38

5.2.1 SOLID Principles	38
5.2.2 Testing	40
5.2.3 Consistency	40
5.2.4 YAGNI (You Aren't Going to Need It).....	40
5.2.5 DRY (Do not Repeat Yourself).....	40
5.3 Design Style and Pattern.....	41
5.3.1 Architecture Design style.....	41
5.3.2 Architecture Design Patterns	41
6. PROOF OF CONCEPT	45
6.1 Requirement 1: Users can book an appointment.	48
6.2 Requirement 2: Users can see their future appointment.	50
6.3 Requirement 3: User can view their profile with all details.....	51
6.4 Requirement 4: Admin can see all appointments by particular date.	52
6.5 Requirement 5: Preventing users from scheduling multiple appointments for the same vaccine.	54
7. Glossary.....	55
8. References	59

❖ Contribution

Team Member	Contribution
Bariq Ishtiaq Mohammed Status: OK	<ul style="list-style-type: none"> ➤ Prepared Stakeholders and Concerns and prepare a table for quality attribute mapping. ➤ Reviewed business goals and vision. ➤ Reviewed Software Requirement. ➤ Contributed to Software Architecture Pattern. ➤ Build and Reviewed Glossary. ➤ Reviewed ViewsPoints and View. ➤ Citations in the final report.
Manthan Pareshbhai Moradiya Status: OK	<ul style="list-style-type: none"> ➤ Prepared Business Goals of System and Review Overall vision of the Project. ➤ Prepared an Expressive Description for Use Case and Sequence Diagram. ➤ Contributed to Package Diagram Implementation in GitMind. ➤ Implemented backend API endpoints. ➤ Analyzed the various quality attributes related to each stakeholder. ➤ Contributed to preparing and documenting software Principles. ➤ Tested the API endpoints using postman.
Shravya Nalla Status: OK	<ul style="list-style-type: none"> ➤ Prepare and Documented Software Requirements. ➤ Reviewed business goals and vision. ➤ Contributed to Software Architecture Pattern. ➤ Contributed to Stakeholders and Concerns. ➤ Reviewed ViewsPoints and View. ➤ Identified and documented the best architecture decision. ➤ Build and Reviewed Glossary.

Siddhartha Nanda Status: OK	<ul style="list-style-type: none">➤ Setup project Trello board to track project progress.➤ Implemented the frontend user interface of the application.➤ Prepared a Logical and Deployment view.➤ Prepared overview of the project.➤ Analyzed the functional and non-functional requirements.➤ Build a model for the project.➤ Identified and documented a list of requirements and technical definitions.➤ Prepared Software Architecture Design Practices for the project.
Parth Navsariwala Status: OK	<ul style="list-style-type: none">➤ Arranged and setup meetings.➤ Worked on the documentation part.➤ Implemented backend API endpoints.➤ Tested the API endpoints using postman.➤ Setup database for the project.➤ Prepared Business Goals of System and Review Overall vision of the Project.➤ Created a mind-map for stakeholders.➤ Created UseCase and Sequence Diagram in GitMind.➤ Contributed to preparing and documenting software Principles.

1. VISION

Vaccine management system with VAXSYS application.

1.1 Overview

In recent years, there has been a surge in the number of infectious diseases. With the roll-out of widely approved vaccines, the primary task for the government is to immunize citizens of Montreal with the hope of eradicating deadly diseases. The major concern for the government is the shortage of equipment and vaccine procurement. Therefore, the Vaccination Management System aids in the reduction of medical sector burdens, by providing reliable, transparent, scalable, and authenticated solutions. Nowadays, everyone wants to adopt online solutions thereby making VAXSYS vaccination services more accessible to patients.

The VAXSYS Vaccination Management solution is a robust and scalable solution that aims to strengthen the entire process of vaccines, starting from arranging vaccines from the supplier to administering vaccines to the citizens. The VAXSYS is a one-stop solution for all kinds of vaccines and is subdivided into multiple steps.

The VAXSYS project will be implemented for doctors, nurses, and patients in a specific hospital or clinic. The key feature of this software is that it allows patients to schedule vaccination appointments online. This application also provided potential dose information to the patients. In other words, patients will be able to see their appointment history and schedule/reschedule vaccination appointments through a web application.

Scope: We opted to scope the project to the following limits due to the short time of the course.

Location: Montreal but accessible from anywhere in Canada

Constraints: Single Hospital

1.2 Business Goals

For business goals we decomposed our goals into subgoals and to represent goals, we use the GQM (only goals and question) framework.

Goal: VAXSYS must be accessible within Canada by people using a computing device.

Question: What kind of platform is flexible for users is it a mobile Application or Website?

Goal: Only authorized personnel can access specific information.

Question: Who is the authorized person to access the information?

Goal: VAXSYS should not allow an individual to receive more than the number of authorized doses of a vaccine for a particular infectious disease.

Question: What kind of Software/code can be used to map the doses received concerning that disease?

Goal: VAXSYS must aim to support the privacy of the information of vaccine recipients.

Question: Who provides security to personal information?

Goal: VAXSYS will allow authorized personnel to enter pertinent information (including, but not limited to, any side-effects of vaccination) about a vaccine recipient.

Question: Why do we need to validate the person's authorization to enter respective feedback?

Goal: VAXSYS will allow users to modify and search the information (i.e., clinic).

Question: What do we do to implement the search module?

Goal: VAXSYS will send the proof of vaccination to each recipient.

Question: What kind of communication method do we need to use to send the proof of vaccination?

Goal: VAXSYS will remind users about their scheduled vaccination appointment.

Question: How many hours/days earlier does the user need to be reminded about their appointment?

Goal: VAXSYS allows the patient to schedule appointments from any location within Montreal.

Question: Are patients able to book appointments from any location or restricted to particular locations?

Goal: VAXSYS allows doctors to have information from records about their patients.

Question: What and how much data has to be included in the records?

Goal: VAXSYS allows users to access information about the specialist.

Question: What information does the profile have?

Goal: VAXSYS does not allow an individual without a valid personal identification to schedule an appointment.

Question: Who verifies the personal identification provided by the user?

Goal: VAXSYS should allow an individual to receive mixed doses of different types of vaccines for the same infectious disease.

Question: How to map between two different doses of two different vaccine companies?

Goal: VAXSYS provides good service to the users by making itself available handy.

Question: To what extent users are satisfied with the system?

2. STAKEHOLDERS AND CONCERNS

2.1 Mind map of stakeholders

Stakeholders are those individuals, teams, or organizations that have an interest in the system. The following stakeholders can be considered for VAXSYS System.

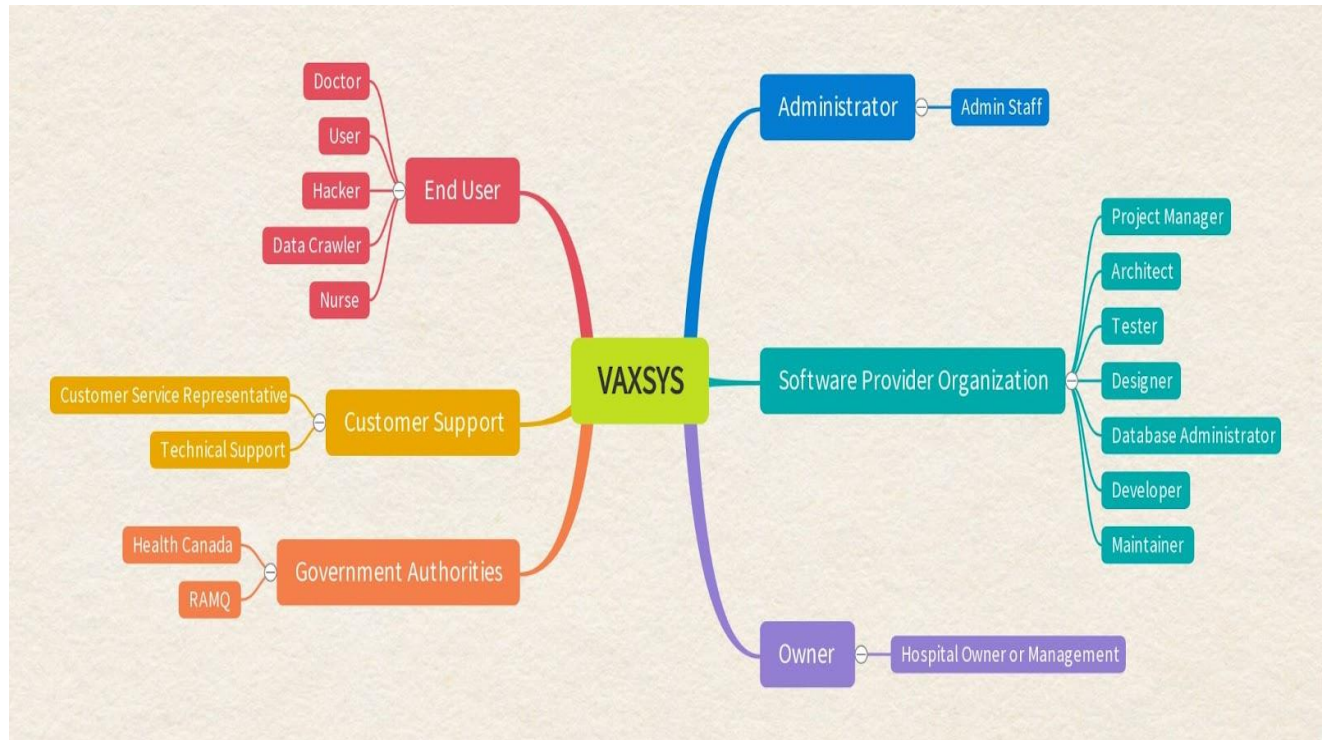


Figure 2.1 Mind Map of Stakeholders

2.2 Stakeholders and Concerns

The below tables show different stakeholders with concern and quality attribute and for that we divide the stakeholders into several groups.

2.2.1 Group of Stakeholders

Owner	Hospital or Organization
End-User	Physician, Patient, Hacker, User, Nurse, Data crawler, Pharmaceutical Company
Administrator	Admin Staff
Software Provider Organization	Project Manager, Architect, Tester, Designer, Database Administrator, Developer, Maintainer, Security Specialist
Customer Support	Customer Service Representative, Technical Support
Government Authorities	Health Canada, RAMQ

Table 2.1 Group of Stakeholders

2.2.2 Owner

Hospital or Organization: The hospital's main tasks are to create standards to which the hospital and its workers must follow, to serve patients by providing the best possible care, and to allocate resources in such a way that the community's requirements are addressed as efficiently as possible.

Concerns	Quality Attribute
Check if the patient's data is verified or not?	Correctness
Check the availability of resources required	Availability
Are the rules and regulations of the hospital maintained?	Security
Is the information of medical staff and patient is appropriate	Correctness

Table 2.2 Owner's Concern and Quality Attribute

2.2.3 End User

Physician: A qualified medical professional; a doctor.

Concerns	Quality Attributes
Check if the patient's data is verified or not?	Correctness
Check if the patient's data is up to date?	Availability
Does the system check for illegal activities and protect them from them?	Security
Check if the System is easy to use?	Usability

Table 2.3 End User's Concerns and Quality Attribute

Patient: A patient is somebody who receives a vaccine from a healthcare provider.

Concerns	Quality Attributes
Check if the doctor's information is verified or not?	Correctness
Does the information provide sufficiently for physicians?	Availability
Does the system check for illegal activities and protect them from them?	Security
Check if the System is easy to use?	Usability

Table 2.4 Patient's Concerns and Quality Attribute

Nurse: Patients are treated by medical personnel who have been vetted and accredited.

Concerns	Quality Attributes
Check if the patient's data is verified or not?	Correctness
Does the system check for illegal activities and protect them from them?	Security
Check if the System is easy to use?	Usability
Check if the System gives quick and accurate responses?	Performance
Check if the System is accessible to users all the time?	Availability

Table 2.5 Nurse's Concerns and Quality Attribute

Negative User:

Hackers: A user group that attempts to get access to unlawful system information.

Concerns	Quality Attribute
Is the system secure against all illegal access?	Security

Table 2.6 Negative User's Concerns and Quality Attribute

Data crawler: A user who tries to crawl unauthorized information to build web search indices.

Concerns	Quality Attribute
What information should the system publicly disclose according to the search?	Security

Table 2.7 Data Crawler's Concerns and Quality Attribute

Admin Staff: They are in responsible of the center's staff, financial, buildings, equipment, and supplies, as well as planning, organizing, directing, and regulating all resource divisions and services. Manages the facility's whole budget and takes an active role in planning with all senior health centers and government officials.

Concerns	Quality Attribute
Check the availability of resources required	Availability
Check if the vaccinations slots are updated as per schedule or not	Maintainability
Are the rules and regulations maintained?	Security

Table 2.8 Admin Staff's Concerns and Quality Attribute

2.2.4 Software Provider Organization

Project Manager: Planning, organizing, and managing projects to ensure they are finished on time, on budget, and within scope.

Concerns	Quality Attribute
Is the software going to be delivered on time in a given budget, product quality, and scope?	Feasibility
Which methodology to be used for development.	Effectiveness
Is it possible to change or modify the system's implementation?	Maintainability
Is there effective communication taking place within the project teams?	Effectiveness

Table 2.9 Project Manager's Concerns and Quality Attribute

Software Architect: An individual is in charge of selecting the project's high-level architecture and enforcing code standards.

Concerns	Quality Attribute
Is the system capable of adapting to the new environment?	Portability
Is it possible to pass along a system design that is comparable to ours?	Effectiveness
Is the technology that was utilized to create the system outdated? Is it simple to upgrade the system if so?	Maintainability
Is it possible to expand the architecture? (To create a product line)	Scalability
Are the components of a system able to communicate with one another?	Interoperability

Table 2.10 Software Architect's Concerns and Quality Attribute

Tester: A person who tests the project's functional and non-functional requirements.

Concerns	Quality Attribute
Is it easy to test the code?	Testability

Are all probable circumstances for system validation covered by the test cases?	Correctness
---	-------------

Table 2.11 Tester's Concerns and Quality Attribute

Designer: Problem-solving and software solution planning are the responsibilities of a software designer.

Concerns	Quality Attribute
Is the design easy to understand and correctly constructed	Correctness
Is it possible to expand the architecture? (To create a product line)	Scalability
Is the technology that was utilized to create the system outdated? Is it simple to upgrade the system if so?	Maintainability

Table 2.12 Designer's Concerns and Quality Attribute

Database Administrator: Database Administrators, sometimes known as Database Analysts, are professionals who store, manage, and maintain a company's data using specialist software. Their responsibilities include database planning and development, data integrity, and technical problem-solving.

Concerns	Quality Attributes
Implementing changes in database operations and ensuring data is readily available to end-users	Availability
Maintaining the database's efficiency through frequent reviews and data updates	Maintainability
Database downtime is minimized, and data delivered is accurate.	Correctness
Managing database access and keeping stored data up to date.	Effectiveness

Table 2.13 Database Administrator's Concerns and Quality Attribute

Developer: A group of users who develop software.

Concerns	Quality Attributes
What is our system's environment? (Hardware, Operating System)	Adaptability
Is it possible to pass along a system design that is comparable to ours?	Effectiveness
Is the technology that was utilized to create the system outdated? Is it simple to upgrade the system if so?	Maintainability
Is it possible to expand the architecture? (To create a product line)	Scalability
Are the components of a system able to communicate with one another?	Interoperability

Table 2.14 Developer's Concerns and Quality Attribute

Security Specialist: They examine current procedures and protections and make recommendations for improvements, as well as create standard organizational rules. They keep an eye on systems for signs of infiltration or possible vulnerabilities, and they develop rules and processes for reporting problems or receiving warnings.

Concerns	Quality Attributes
Is the data of the organization safe from cyber-attacks?	Security
Are the action plans, project plans, incident, issue, and risk registers?	Maintainability

Are the cameras, phones, intercom, and staff and youth moments monitored?	Surveillance
---	--------------

Table 2.15 Security Specialist's Concerns and Quality Attribute

Maintainer: A person who adds new features to software or moves it from one platform to another.

Concerns	Quality Attributes
Is the system flexible enough to adjust to the changing circumstances?	Portability
Is upgrading to a new version of the system simple?	Maintainability

Table 2.16 Maintainer's Concerns and Quality Attribute

2.2.5 Customer Support

Customer Service Representative: Deal with consumer concerns and offer system information.

Concerns	Quality Attributes
Quality of the system	Usability

Table 2.17 Customer support's Concerns and Quality Attribute

Technical Support: Assist those who are having trouble with a software of any sort.

Concerns	Quality Attributes
Maintain the system.	Maintainability

Table 2.18 Technical Support's Concerns and Quality Attribute

3. Problems

VAXSYS is an online vaccination management software that allows people to register for vaccinations from any location within the designated territory. It also allows healthcare providers to access and manage immunization appointments as well as patient information.

The software is a user-friendly interactive application that may be accessed from anywhere in Quebec.

It is a mobile-friendly online application that can be used from any gadget with a browser with an active internet connection.

In this part, we only offer the Architecturally Significant Requirements since they influence architectural design decisions. The VAXSYS software requirements are shown below. Stakeholder requirements, system requirements, and system element requirements are all included in the requirements.

3.1 Non-Functional Requirements

Requirement 1: User will be able to login or register if using VAXSYS for the first time.	
Quality Attribute	Usability
Sub-Quality Attribute	Accessibility
Subject / Use case /Actor	VAXSYS System
Action	Can be accessible
Constraints of Action	Mobiles are easily carriable which supports the VAXSYS system and can be accessed from anywhere.

Table 3.1 Non-Functional Requirement-1

Requirement 2: Allow the new user to register in the VAXSYS system before using any of the function, and a profile will be established as a result.	
Quality Attribute	Security
Sub-Quality Attribute	Authenticity, Functionality
Subject / Use case /Actor	Patient or Doctor
Action	Login or Register
Constraints of Action	Only users with valid login ID can access

Table 3.2 Non-Functional Requirement-2

Requirement 3: Ascertain that data is saved and reused.	
Quality Attribute	Reusability
Sub-Quality Attribute	Dependability, Maintainability, Usability
Subject / Use case /Actor	VAXSYS System
Action	Usability of data
Constraints of Action	Data stored and accessed easily

Table 3.3 Non-Functional Requirement-3

Requirement 4: VAXSYS's back-end should not be dependent on the front-end and vice versa.	
Quality Attribute	Maintainability
Sub-Quality Attribute	Testability, Modularity, Reusability
Subject / Use case /Actor	VAXSYS System
Action	Develop back-end and front-end
Constraints of Action	Both back-end and front-end are developed, maintained, and used independently

Table 3.4 Non-Functional Requirement-4

Requirement 5: Ensure that the information individuals enter in the application is kept private and secure.	
Quality Attribute	Privacy
Sub-Quality Attribute	Confidentiality
Subject / Use case /Actor	Patient or Doctor
Action	Data is encrypted
Constraints of Action	Maintain both patient's and doctor's private information

Table 3.5 Non-Functional Requirement-5

Requirement 6: Utmost inoperable time expected to be not more than 5%	
Quality Attribute	Reliability
Sub-Quality Attribute	Availability
Subject / Use case /Actor	VAXSYS System
Action	System is expected to be working almost all the time

Constraints of Action	Availability of the system is expected to be more than 95 percent
-----------------------	---

Table 3.6 Non-Functional Requirement-6

Requirement 7: VAXSYS supports multiple users accessing the system at the same time	
Quality Attribute	Performability
Sub-Quality Attribute	Capacity
Subject / Use case /Actor	VAXSYS System
Action	Accessing the system
Constraints of Action	System is expected to have more than 10 users at a time

Table 3.7 Non-Functional Requirement-7

Requirement 8: Response time of the VAXSYS is expected to be not more than 5 seconds.	
Quality Attribute	Performability
Sub-Quality Attribute	Availability, Usability
Subject / Use case /Actor	VAXSYS System
Action	Response from the system
Constraints of Action	Expected to react in 5 seconds or less

Table 3.8 Non-Functional Requirement-8

Requirement 9: VAXSYS software should be adaptable to any platform	
Quality Attribute	Modifiability
Sub-Quality Attribute	Maintainability, Reusability, Agility, Compatibility
Subject / Use case /Actor	Developer
Action	Accessing from anywhere on any device
Constraints of Action	Developed on any platform which can be accessed by a user

Table 3.9 Non-Functional Requirement-9

3.2 Functional Requirement

Requirement 1: The user will be able to login or register if using VAXSYS for the first time.	
Quality Attribute	Usability
Sub-Quality Attribute	Accessibility
Subject / Use case /Actor	Doctors or Patient
Action	Being able to log in or register with the system
Constraints of Action	Will be asked to enter basic information during the registration

Table 3.10 Functional Requirement-1

Requirement 2: Allow eligible users to retrieve or change their password if the user wants to.	
Quality Attribute	Usability
Sub-Quality Attribute	Accessibility, Operability
Subject / Use case /Actor	Doctor or Patients
Action	Recover or change the password
Constraints of Action	Only a valid user can perform these actions

Table 3.11 Functional Requirement-2

Requirement 3: Only Authorized users can access the VAXSYS.	
Quality Attribute	Security
Sub-Quality Attribute	Authenticity
Subject / Use case /Actor	VAXSYS
Action	Login or Registration required
Constraints of Action	Shows error message for invalid user

Table 3.12 Functional Requirement-3

Requirement 4: Users will be able to book cancel or reschedule their appointment till a day before the appointment.	
Quality Attribute	Usability
Sub-Quality Attribute	Accessibility
Subject / Use case /Actor	VAXSYS, Patient
Action	Book or Edit booking

Constraints of Action	Can make changes or cancel the appointment before 24hrs.
-----------------------	--

Table 3.13 Functional Requirement-4

Requirement 5: Patients can be able to view their appointments.	
Quality Attribute	Usability
Sub-Quality Attribute	Security, Accessibility
Subject / Use case /Actor	VAXSYS, Patient
Action	View patient's schedule
Constraints of Action	The patient should have valid login

Table 3.14 Functional Requirement-5

Requirement 6: Users will be able to view their profile and add additional information about their medical records.	
Quality Attribute	Usability, Security
Sub-Quality Attribute	Authenticity, Accessibility
Subject / Use case /Actor	VAXSYS, Patient, Doctor
Action	View and edit profile or add information
Constraints of Action	Users should have valid login

Table 3.15 Functional Requirement-6

Requirement 7: Doctors or Specialists will be able to view their patient's profile	
Quality Attribute	Security
Sub-Quality Attribute	Authenticity
Subject / Use case /Actor	VAXSYS, Doctors
Action	View the patient's profile
Constraints of Action	A doctor should have valid login

Table 3.16 Functional Requirement-7

Requirement 8: Preventing users from scheduling multiple appointments for the same vaccine	
Quality Attribute	Usability, Dependability
Sub-Quality Attribute	Accessibility

Subject / Use case /Actor	VAXSYS, Patients
Action	Ensure to have a single appointment for the respective vaccine
Constraints of Action	Shows an error message if the user tries to book more than one appointment

Table 3.17 Functional Requirement-8

Requirement 9: Sending reminders to users about their upcoming vaccination appointments.	
Quality Attribute	Traceability, Maintainability
Sub-Quality Attribute	Analyzability
Subject / Use case /Actor	VAXSYS
Action	Maintain and trace the appointments data
Constraints of Action	Sending reminders to patients having appointments within a week

Table 3.18 Functional Requirement-9

Requirement 10: Provide a comprehensive list of clinics along with contact information	
Quality Attribute	Usability
Sub-Quality Attribute	Accessibility
Subject / Use case /Actor	VAXSYS
Action	View clinics information on the Website
Constraints of Action	The system administrator should have VAXSYS updated with clinic information

Table 3.19 Functional Requirement-10

Requirement 11: Allow VAXSYS to map patients' previous vaccine doses and suggest appropriate doses for future vaccine appointments	
Quality Attribute	Traceability
Sub-Quality Attribute	Analyzability
Subject / Use case /Actor	VAXSYS
Action	Map previous doses from patients' profile
Constraints of Action	Suggest proper doses to be taken in for future appointments

Table 3.20 Functional Requirement-11

Requirement 12: System Administrator will be able to update the opening & closing times of clinics and add basic information about the vaccines available in respective clinics.	
Quality Attribute	Maintainability
Sub-Quality Attribute	Maintainability
Subject / Use case /Actor	VAXSYS
Action	Maintain vaccine data and keeps track of Clinic timings
Constraints of Action	Update basis information about the vaccine and change timings of clinics if necessary

Table 3.21 Functional Requirement-12

Requirement 13: The system administrator will be able to add and remove appointment slots	
Quality Attribute	Maintainability
Sub-Quality Attribute	Maintainability
Subject / Use case /Actor	VAXSYS
Action	Keep trace of no. of appointments
Constraints of Action	Adding or removing appointment slots to a clinic if necessary

Table 3.22 Functional Requirement-13

Requirement 14: System administrators will be able to see today's appointments as well as future appointments. (Ex-Dashboard)	
Quality Attribute	Maintainability
Sub-Quality Attribute	Accessibility, Usability
Subject / Use case /Actor	VAXSYS System
Action	Get knowledge on the appointments
Constraints of Action	An administrator can check for appointments

Table 3.23 Functional Requirement-14

Requirement 15: The patient will be able to download the respective Proof of Vaccination	
Quality Attribute	Usability
Sub-Quality Attribute	Accessibility

Subject / Use case /Actor	VAXSYS System and Patient
Action	Able to download proof of vaccination
Constraints of Action	Access and download proof of vaccine from the system

Table 3.24 Functional Requirement-15

4. VIEWPOINTS AND VIEWS

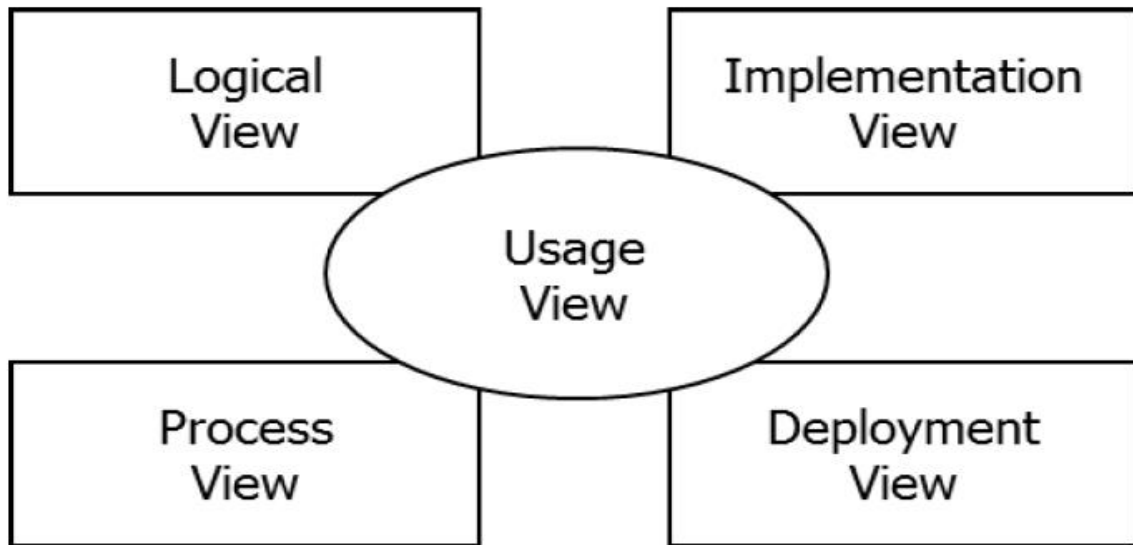


Figure 4.1 4+1 View Model of Software Architecture

We followed 4+1 architecture for our project. Each type of view serves specific stakeholder that has concern described in that view. Here meaning of Usage View is USE CASE view. The 4+1 View Model of Software Architecture has been adopted by the Rational Unified Process (RUP).[\[3\]](#) The remaining 4 views is as follow:

1. Logical View
2. Process View
3. Deployment View
4. Implementation View

4.1 Use Case View

A use case diagram is used to illustrate the use-case view. This diagram depicts the actor as well as the system's function. This diagram also shows how the specified actor interacts with the system.

Stakeholders: user, nurse, and doctor

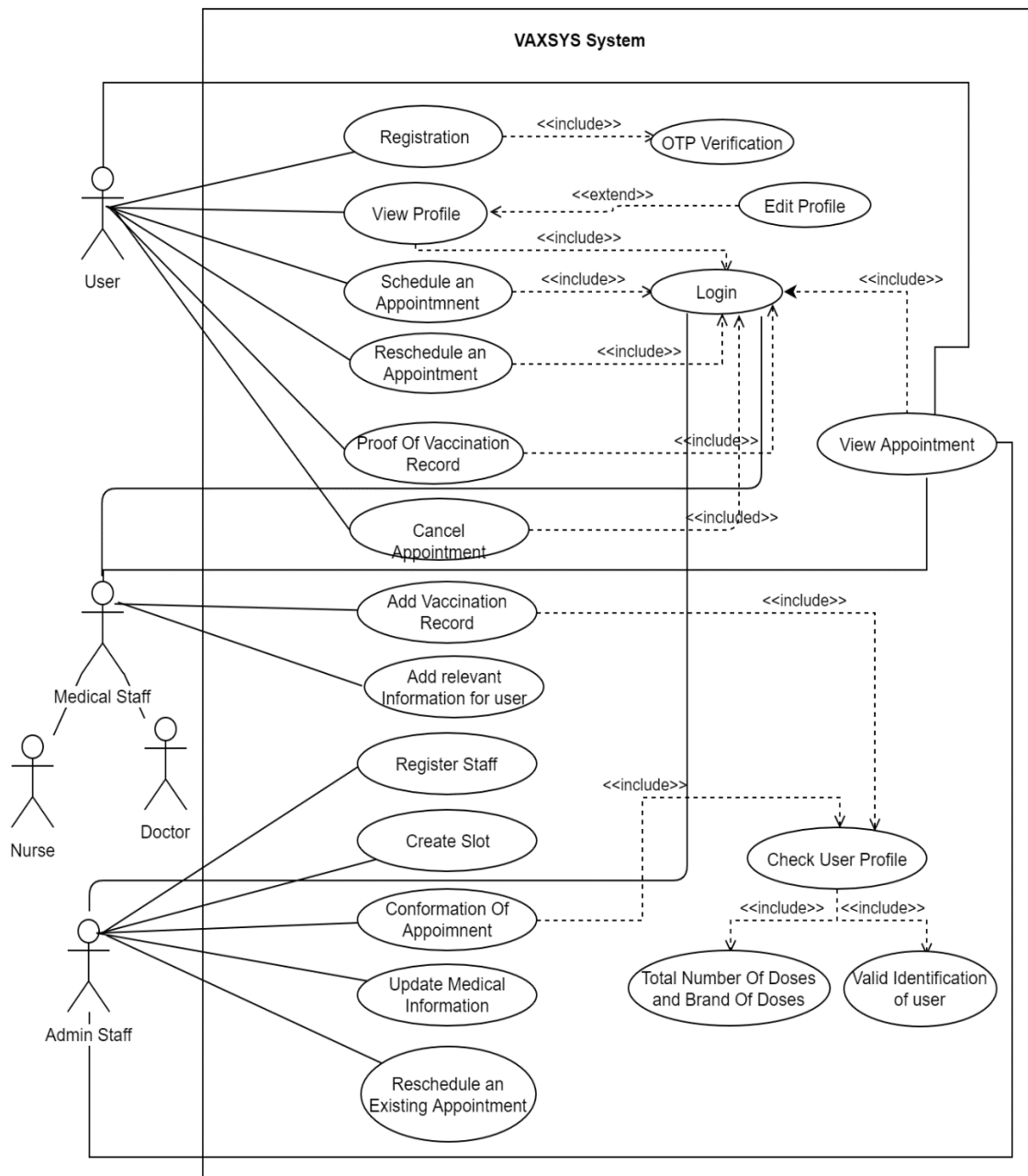


Figure 4.2 Use case View

Use case Description:

Function	Actor	Description
Registration	user	Users can create an account in VAXSYS using their personal information.
OTP verification	user	A one-time password is used to verify the identity of the user.
View profile	user	User can view their personal information.
Edit profile	User	User can edit their personal information.
Login	User, medical staff, admin staff	Users, medical staff, and admin staff can log in to the system using their user ID and password.
Schedule an appointment	User	Users can schedule an appointment for the vaccination.
Reschedule an appointment	User	User can reschedule their previously booked vaccination appointment.
Cancel an appointment	User	User can cancel their vaccination appointment.
Proof of vaccination record	User	Users can download the proof of vaccination.
View appointment	User, Medical staff, Admin staff	User can view their appointment. Medical staff and admin staff can view the user's appointment.
Add vaccination record	Medical staff	Medical staff can add vaccination records to the user profile.
Add relevant information for a user	Medical staff	Medical staff can add the relevant information about the vaccine.
Register staff	Admin staff	Admin staff can register the nurse and doctor.
Create slot	Admin staff	Admin staff can create a vaccination slot as per the availability of the vaccine.

Confirmation of appointment	Admin staff	Admin staff can confirm the appointment, booked by the user.
Update medical information	Admin staff	Admin staff can update the information(i.e. opening and closing hours) of the vaccination center.
Check user profile	Admin staff, medical staff	Admin staff and medical staff can be able to check the user history. For example, identification of the user, how many doses the user received, and the brand of the previous vaccine.
Reschedule an existing appointment	Admin staff	Admin staff can reschedule the user's appointment.

Table 4.1 Use Case View Description

4.2 Process View

A sequence diagram is used to describe the process view of the system. This view illustrates the run time behavior of the system, and how the users are interacting with the system.

Stakeholders: user, nurse, and doctor

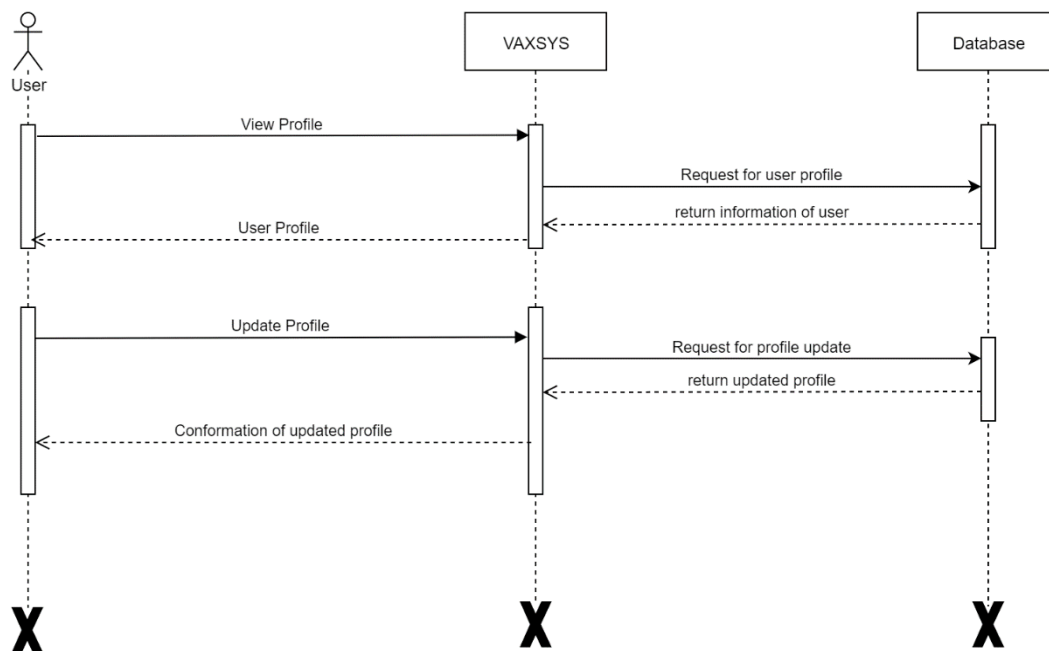


Figure 4.3 View and Update Profile

View profile:

Step 1: If a user wants to see their profile, then he/she can click the view profile option of the VAXSYS.

Step 2: VAXSYS sends the request to the database to fetch the user's personal information.

Step 3: The database sends the personal information to the VAXSYS. And system displayed the personal information to the user.

Update profile:

Step 1: If a user wants to update their profile, then he/she can click the update profile option of the VAXSYS.

Step 2: VAXSYS sends the updated personal information to the database.

Step 3: The database sends the confirmation to the VAXSYS and the VAXSYS sends the confirmation to the user.

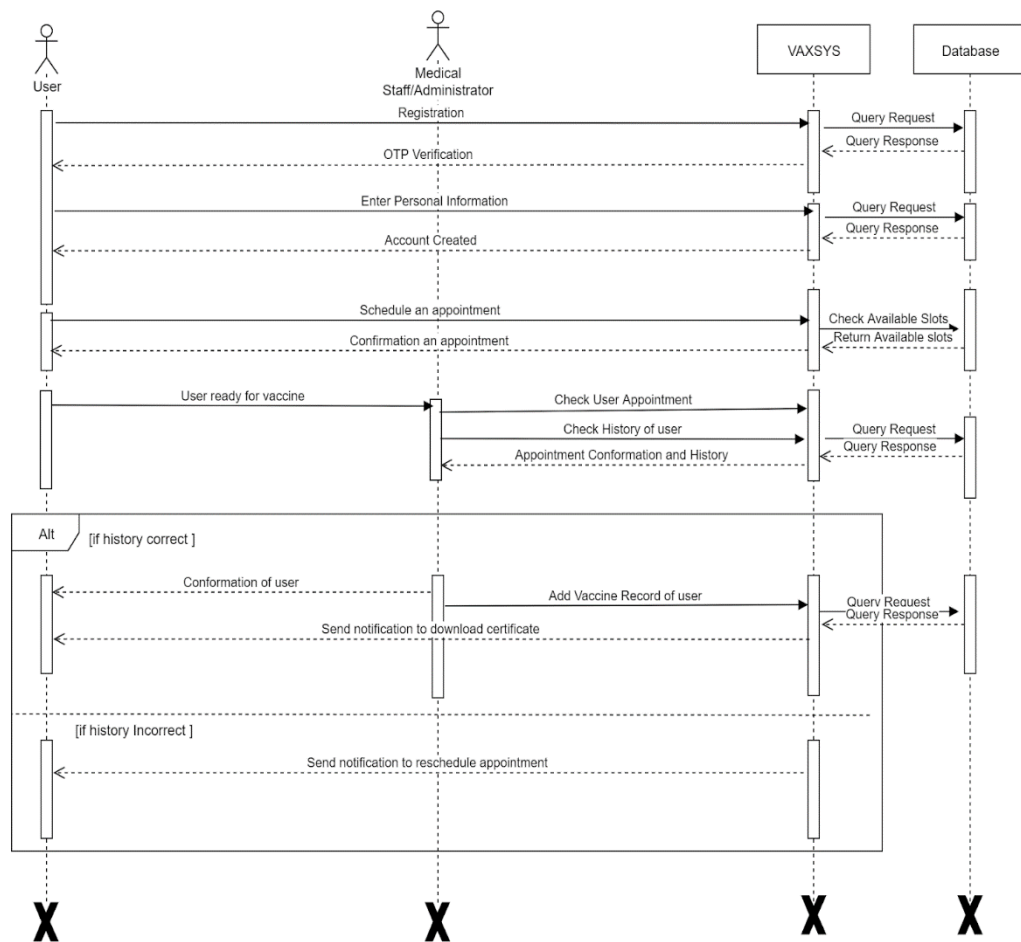


Figure 4.4 Booking an appointment

Appointment booking:

Step 1: Users register to VAXSYS using their email addresses.

Step 2: VAXSYS send the one-time password to the user to verify the user's identity.

Step 3: Users enter their personal information in VAXSYS.

Step 4: VAXSYS sends the confirmation of account creation to the user.

Step 5: Users schedule an appointment for vaccination.

Step 6: If the slots are available then VAXSYS sends the confirmation of an appointment to the user.

Step 7: At the time of an appointment, medical staff check if the appointment is valid or not and medical staff also check the history of the user.

Step 8: VAXSYS provides information on user appointments and user history to the medical staff.

If the user history is correct,

Step 9: Medical staff adds the record of the vaccine to the user's profile on VAXSYS.

Step 10: VAXSYS sends the notification to download the proof of vaccination to the user.

If the user history is not correct,

Step 9: VAXSYS sends a notification to reschedule a vaccination appointment to the user.

4.3 Implementation View

This view provides information on the software systems from the programmer's side. It is also known as the development view. To illustrate the implementation view package diagram is used.

Stakeholders: developer, tester

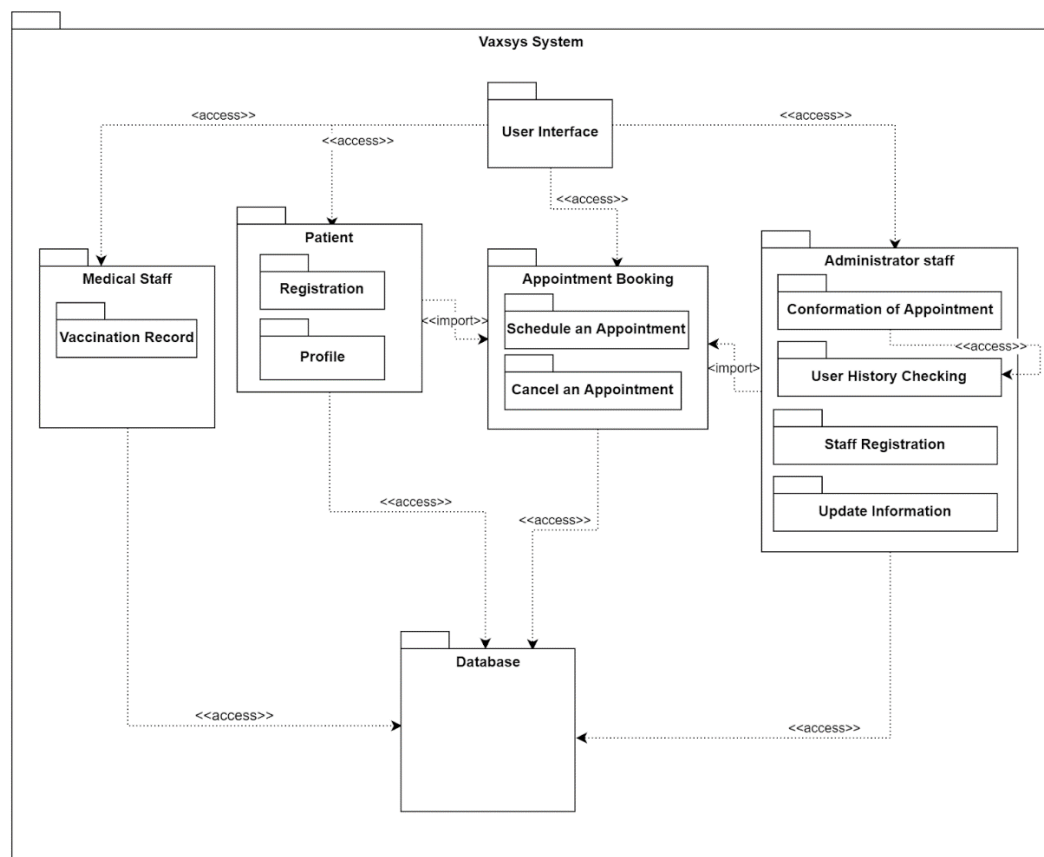


Figure 4.5 Package Diagram

Implementation View Description:

Package Name	Sub package Name	Description
User Interface		This package includes the UI of VAXSYS.
Medical Staff	Vaccination record	Medical staff packages include the functionalities used by the medical staff i.e., adding vaccination records of the patient's profile. This package accesses the database for the operations.
User	Registration, Profile	This package includes user registration and profile-related functionality. This package imports the functionality of the appointment booking package, to schedule an appointment. This package accesses the database in order complete the operations.
Appointment booking	Schedule an appointment, Cancel an appointment	This package is used to schedule and cancel an appointment. This package accesses the database to complete transactions.
Administrator staff	Confirmation of an appointment, User history checking, Staff Registration, Update information	This package is used to perform admin functionality like confirming an appointment, staff registration, etc. This package imports the appointment booking package and accesses the database.
Database		This package is used to store information about users, staff, and vaccination in the database.

Table 4.2 Implementation View Description

4.4 Logical View

The logical view primarily considers the functionality that the system should provide to its end users. The system is decomposed into a smaller set of abstraction, encapsulation, and inheritance components generated from the problem domain. A UML diagram consists of a state diagram and class diagram which are used in this view, thereby providing the high-level functionality of all the components in each layer of abstraction. This is responsible for the conceptual organization of these layers and how the inner components are connected.

Stakeholders: User, nurse, and doctor

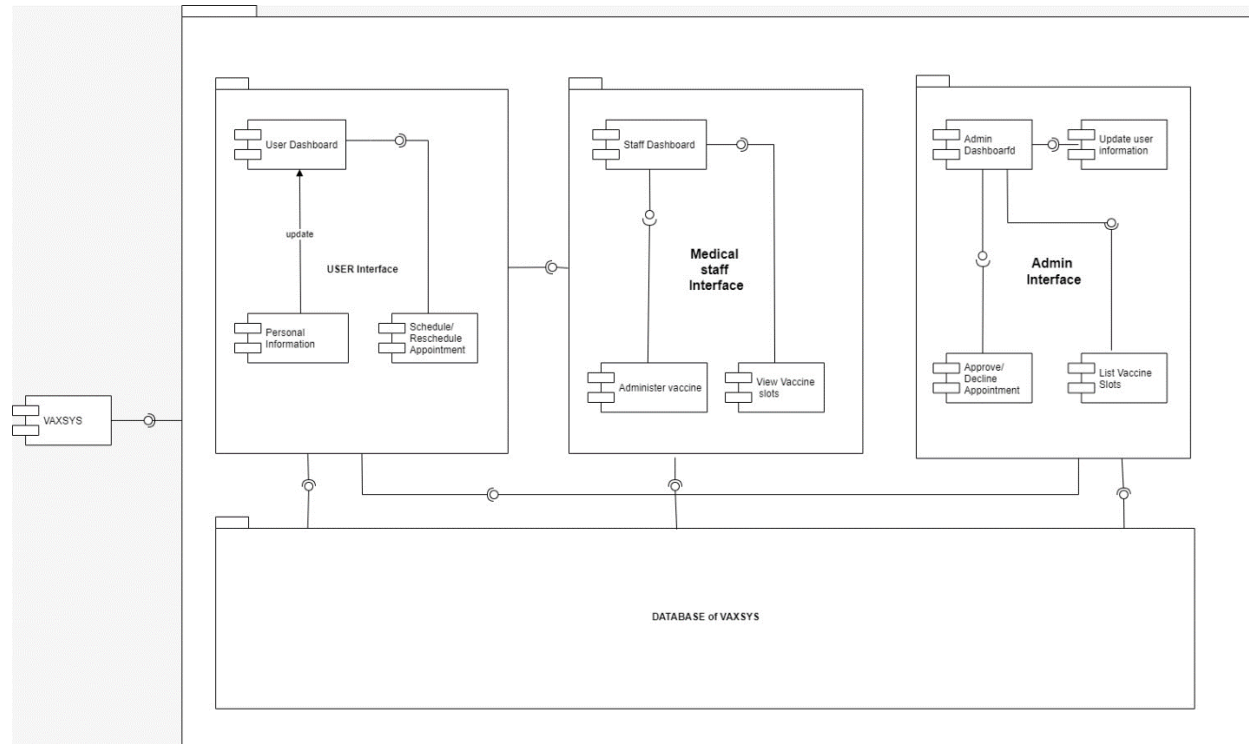


Figure 4.6 Component Diagram

4.4.1 Related viewpoints

Problem domain viewpoint, maintainers viewpoint

Logical View Description

Components associated with the logical view are mentioned below: -

1. User Interface
2. Medical staff Interface
3. Admin Interface
4. Database

User Interface:

User View is related to the end customer's profile, scheduling/rescheduling appointments, and downloading proof of vaccination.

Users can register themselves on the website and can create their profiles. In addition to that, they are allowed to edit their profile for correction or to add some useful information. Users can schedule an appointment from the given slots which can later be modified by rescheduling or canceling. Also, they can download the proof of vaccination of the administered dose.

Medical staff Interface:

Medical staff view relates to checking appointments to be administered to the user, making certificates available to the user, and updating relevant information for the user.

Medical staff like nurses and doctors can log in to check daily appointments that they need to administer to the users. In addition to that, they can update users' data with the vaccination details along with generating vaccination proof for the administered dose.

Admin Interface:

Admin view is related to staff registering, slot creation, appointment confirmation, appointment rescheduling, and user's medical information.

Admins have control over the entire application where they can create slots for the vaccination. They have the authority to register a staff member in the system portal. They can confirm the requested vaccination appointment. They are also entitled to update the medical history of the user and reschedule the user's vaccination appointment.

Database:

Database, an organized collection relates to storing structured data in places like the cloud or computer systems.

Database houses all the vaccination details, the entire system data like - users, medical staff, and admin's basic details. The database also includes their personal information and appointment scheduled by the user with the selected slot with vaccination details. The database contains general information about medical staff. If any user schedules a new appointment, the same information will be stored in a structured way in the database which can be later queried to get details about vaccination or user's details.

4.5 Deployment View

The deployment view describes how the application relies on the environment for its execution. It includes hardware, software execution environments, and the middleware or interlinking connecting them. It can be related to getting the right configuration to run the system on the server.

Stakeholders: Users, Doctor, Nurse

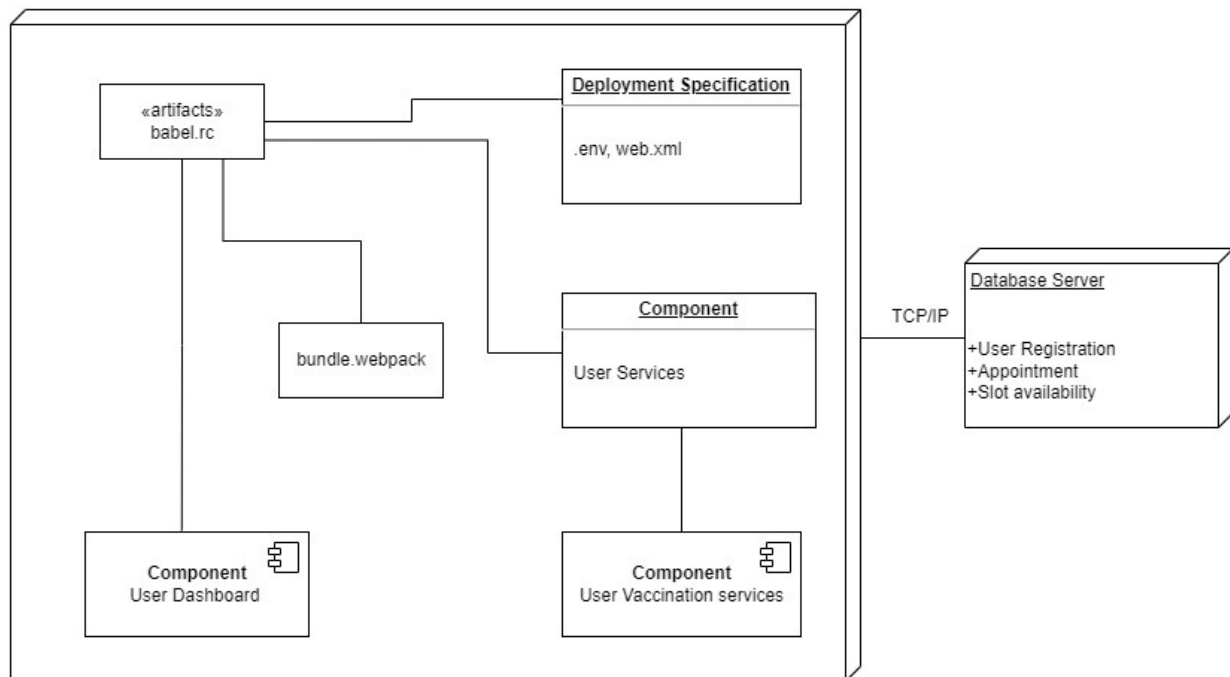


Figure 4.7 Deployment Diagram

Deployment View Description:

The deployment view can be comprehended as the mapping of two views– a logical view which includes conceptual modeling of views and the physical view which includes development environment and run environment configurations and database tables which stores data are and available to query even at a server. It includes an application server to run the system, a database that manages and stores system details, middleware connecting the database to the logic, and an end-user device to access the system.

The deployment view can be seen as an important factor as it helps in analyzing the overall performance of the system running on the server like load balancing, scalability, and setup configuration.

The user has to log in to the system to schedule/reschedule an appointment if he/she is using our web application. To book a slot, the user will check available slots by querying the database. The database in return provides all the slots available with their details which can be used to book the vaccination appointment. All the details like - personal information and vaccination certificate will be stored in the database and can be accessed by the users.

5. ARCHITECTURAL DECISIONS

5.1 Design Practices

Design practices are considered a thoughtful process that helps developers to solve complex problems keeping in mind software usability and effective solutions to the problem domain.[\[31\]](#)

5.1.1 Keep Modules Small

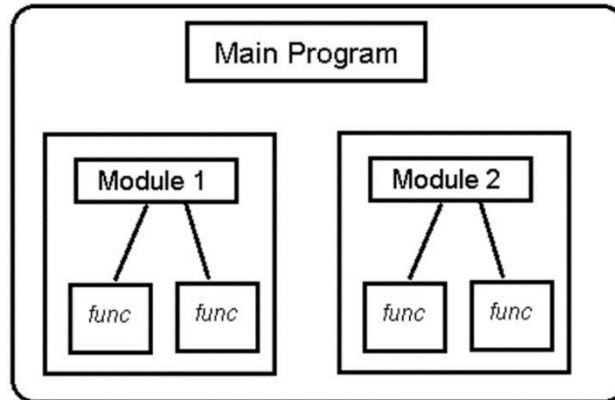


Figure 5.1 Keep Module Safe [\[32\]](#)

With the evolution of software, the code becomes more complex and difficult to maintain. As there are changes in the requirement and the introduction of new features with each iteration, the software becomes boggy. So, to cater to those points, we will keep our modules as simple and small as possible. This will help us to test the modules easily and even if we encounter bugs, that should not take long with the compact code.[\[32\]](#)

5.1.2 Include only concerned functions in any module

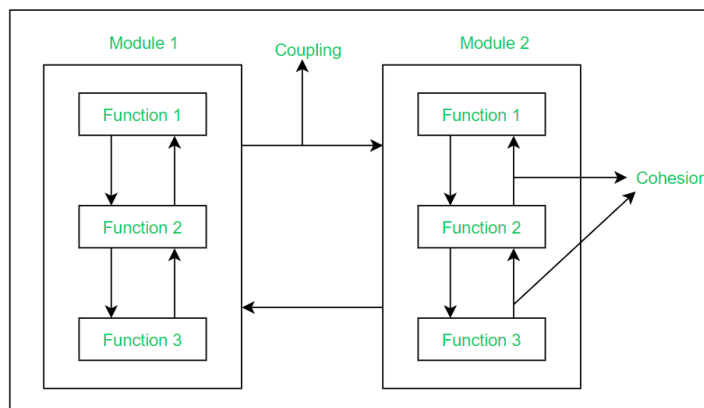


Figure 5.2 Keep Module Safe [\[33\]](#)

The VAXSYS management system will be implemented in such a way that each module will have only one concerning function and a few sub-functions to increase code bifurcation and cohesion. This will help

us to maintain the separation of concerns principle to increase cohesion and reduce coupling between the software modules. Since each component is performing separate functionality, the testing of such simple modules will be much easier, and the code deployed will be bug-free. Even if there are bugs in the software system, they can be easily traced out.[\[33\]](#)

5.1.3 No Redundant Functionality

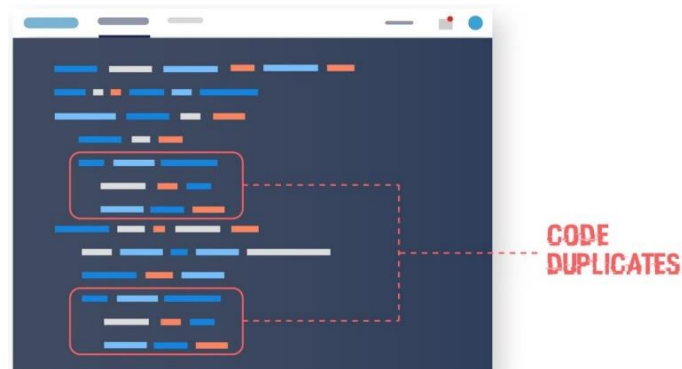


Figure 5.3 No Redundant [\[37\]](#)

The approach of adopting concerning functions in each module eliminates the possibility of redundant code/ functionality in our system. Redundant code in different components introduces inconsistencies and increases the chances of new defects. If a change requires updating a variable name in one place, the developer might miss changing the same variable name in duplicate code which in turn will break code functionality.

5.1.4 Module reusability

Module reusability is considered one of the best design practices. We have segregated some of the candidate components for module reusability. In the VAXSYS management system, the module of slot booking for the vaccination should be available to the admin, hospital staff as well as the -user, booking is the perfect candidate for module reusability.

5.1.5 Testing

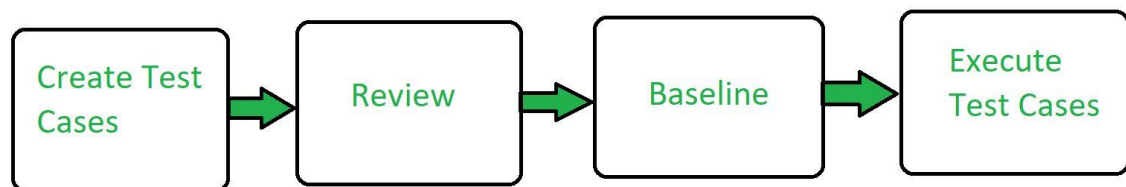


Figure 5.4 Testing [\[38\]](#)

Software testing has emerged as one of the core parts of the development of software systems. We have decided to use a unit testing approach to detect and catch bugs in each module as they are developed. Since we are using one functionality per module, finding bugs will be easier as the code would not be tightly coupled.

5.1.6 Standard Namespacing, conventions, and Coding styles



Figure 5.5 Conventions [34]

The development team should follow standard naming conventions like meaningful names to the functions, and consistent use of character sequence to declare variable names. This will help us to avoid any naming conflicts and improve debugging, remove dead codes, and support better maintainability.

5.1.7 Indentation, Coding Style, Documentation

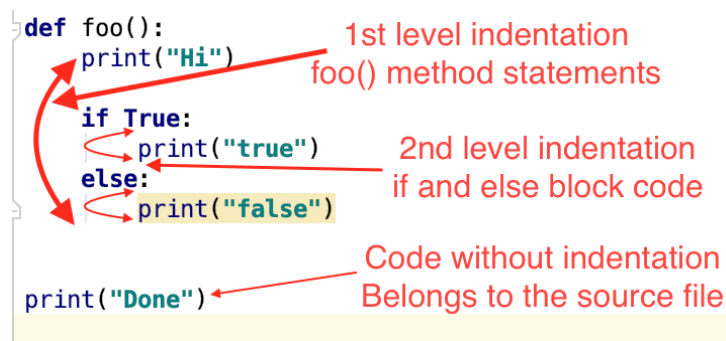


Figure 5.6 Documentation [39]

The proper use of indentation and documentation will help fellow developers to read others code and make any changes with a reduced probability of introducing new bugs in the system. The cleaner the code, the easy it will be to maintain and to be efficient.

5.2 Design Principles

5.2.1 SOLID Principles

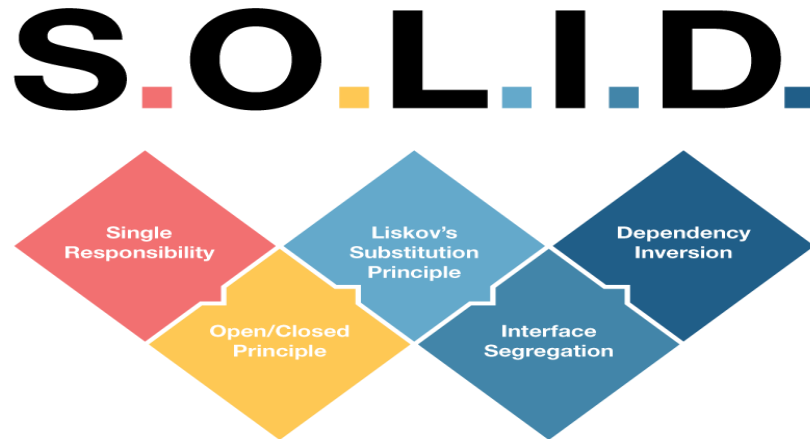


Figure 5.7 SOLID Principle [36]

For our VAXSYS application, we used the SOLID principle which is the most important and most common principle in software engineering. It is combined of 5 principles which are as follows:

1. Single Responsibility Principle (SRP)
2. Open-Closed Principle (OCP)
3. Liskov Substitution Principle (LSP)
4. Interface Segregation Principle (ISP)
5. Dependency Inversion Principle (DIP)

Main benefit of using the SOLID principle is it helps to produce less coupled code and less coupled code which assist in increasing code maintainability and readability because classes are Independent in less coupled code. [29]

5.2.1.1 Single Responsibility Principle (SRP)

This principle states that we have to design a class in such a way that each class has only a single purpose or in another way, we can say that for a class there should not be more than one reason to change. SRP does not state that a class has only one method, variable, or property; it simply states that everything in the class should be related to that single purpose. [29]

In our VAXSYS application, each class has its own responsibility and here the login class is responsible for all login activity whereas the Registration class contains all methods and variables related to registration like this all functionality has its own class.

5.2.1.2 Open-Closed Principle (OCP)

In this principle the meaning of Open is “Open for extension” and it states that new functionality or new requirements can be added to existing code. Close stands for “Close for modification” and it states that except to fix errors, you should never change a class once it has been created. Both parts in principle contradict each other. [\[29\]](#)

In our VAXSYS application VACCINE base class contains all basic information about each vaccine and this class can be extended by different disease vaccines so this class is open for extension but closed for modification, OCP principle makes more robust code and reduces the risk of error in existing code. Once code is written, debug, and tested it will apply limits to change the source code.

5.2.1.3 Liskov Substitution Principle (LSP)

This principle states that methods that use base class pointers or references must be able to use derived class objects without being aware of them. [\[29\]](#)

In our VAXSYS application, we have an inheritance hierarchy with medical staff and Doctors. Wherever you can use Medical Staff, you should also be able to use a doctor, because Doctor is a subclass of Medical Staff.

5.2.1.4 Interface Segregation Principle (ISP)

Clients should not be forced to rely on interface members they do not use, according to the Interface Segregation Principle (ISP). The ISP directs us to establish many, smaller, cohesive interfaces when we have non-cohesive interfaces. Creating a small interface helps us to achieve this principle. So, instead of generalizing the whole thing into one interface, we will try to make a small interface whenever possible. [\[29\]](#)

In VAXSYS, we will make different interfaces of “View appointment” for users, medical staff, and admin staff. Therefore, if we want to add new functionality to the “View appointment” section of admin or medical staff, we can do that without redeploying the user module.

5.2.1.5 Dependency Inversion Principle (DIP)

The Dependency Inversion Principle states that:

1. High-level modules should not depend on low-level vol modules. Both should depend upon abstractions.
2. Abstractions should not depend upon details. Details should depend upon abstractions.

It ensures that high-level modules rely on abstractions rather than concrete implementations of lower-level modules, which aids in the development of loosely coupled programming. [\[29\]](#)

5.2.2 Testing

Software testing is an important element of software development since it ensures that the system is fault-free. We chose unit testing because it allows us to test each component individually and uncover issues, allowing us to improve the system's quality.[\[30\]](#)

5.2.3 Consistency

When we understand the context of software systems, it becomes much easier to work with them. Consistency might be high-level, ensuring that the system recognizes and uses the same language, or low-level, focusing on coding styles.[\[30\]](#)

5.2.4 YAGNI (You Aren't Going to Need It)

This principle states that, before starting implementation, think that this functionality is necessary or not. If we need to implement these features, then any other simpler method is available to build this functionality or can we reuse the existing code. In VAXSYS, we will think about these before implementing every functionality.[\[30\]](#)

5.2.5 DRY (Do not Repeat Yourself)

This principle emphasized reusing the code whenever possible. In VAXSYS, we will try to implement relevant code into one function, and we will keep relevant functions in one class. It helps us to reuse the code.[\[30\]](#)

5.3 Design Style and Pattern

5.3.1 Architecture Design style

As VAXSYS is a web application, it mixes two different architecture approaches. By employing layered architecture, we may effectively isolate our presentation logic from our business logic and data access logic by employing layered architecture. We will deploy the application in a three-tiered approach to better meet our security standards. Data rules, such as Health Canada regulatory guidelines and government laws, will make up our business logic tier. This layer's components will enforce information security procedures in addition to data rules that keep data structures consistent. The data tier, which is the database access layer, will be accessible through the business services layer. Data access components and web services for exchanging data and resources make up this layer.

5.3.2 Architecture Design Patterns

Software architecture patterns are structural layouts that are used in the software development industry to overcome common design difficulties.

The accompanying development, maintenance, and enhancement responsibilities have gotten increasingly laborious and time-consuming as the complexity of today's software have increased. As a result, software architectural patterns have been developed to help software architects, developers, and operators reduce their effort.

These architecture patterns offer a reusable solution to an issue that arises frequently in software design. Software architecture patterns are a description or template for the structural framework or method for addressing a problem that may be used to solve a wide range of issues.

5.3.2.1 Model View Controller (MVC) Design Pattern

MVC is a design pattern that splits an application into three logical components: Model, View, and Controller. MVC was formed as a consequence. Each component of the architecture is intended to handle a certain application development element. MVC separates the business logic and presentation layers. On desktop computers, it was usually utilized for graphical user interfaces (GUIs).[\[26\]](#)

MODEL: The model (for example, the data information) solely comprises pure application data; it does not include any logic for presenting the data to a user.

VIEW: The user sees the model's data through the view (for example, the presentation information). The view understands how to get to the data in the model, but it has no idea what that data represents or what the user may do with it.

CONTROLLER: The controller resides between the view and the model (for example, the control information). It responds appropriately to events that are triggered by the view (or any external source). In most cases, the best option is to use a model method. Because the view and the model are linked by a notification system, the result of this operation is automatically reflected in the view.

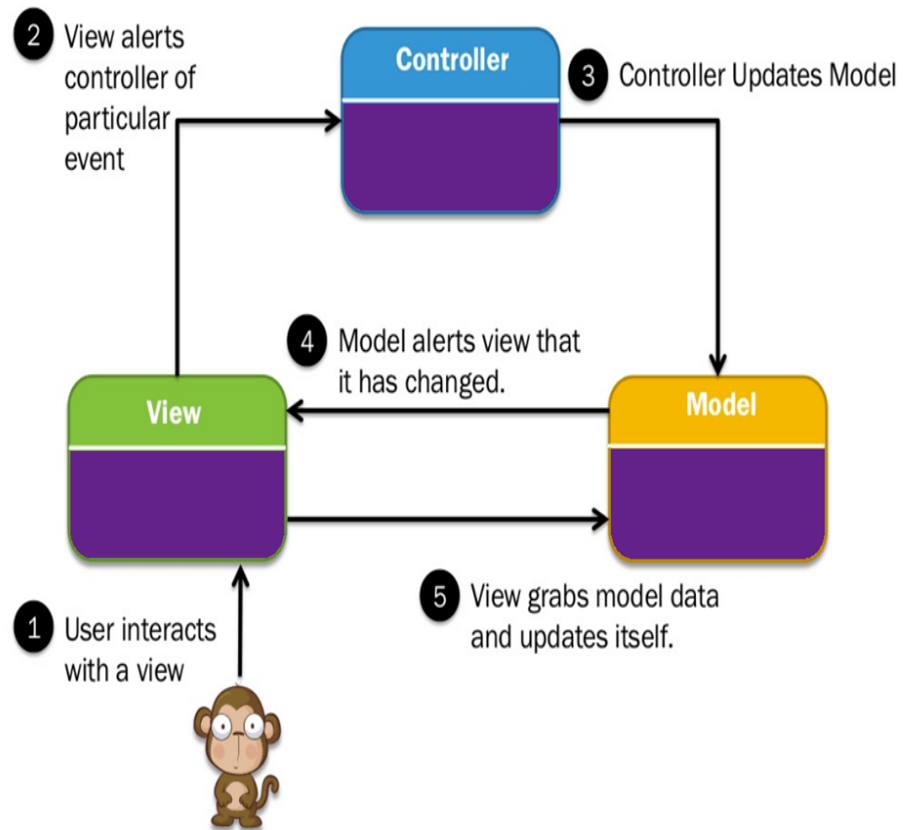


Figure 5.8 MVC architecture [\[35\]](#)

5.3.2.2 Observer Pattern Design

The purpose of the Observer Pattern is to build a one-to-many dependence between objects such that when one object changes state, all its dependents are immediately alerted and updated. The one-to-many connection is defined by the subject and observers. The observers are reliant on the subject; thus, they are alerted when the subject's status changes. [27]

It is appropriate for the VAXSYS system since the user will register and be alerted by the system. As a result, the system will notify the user anytime they make a change to their profile, such as amending personal information, scheduling, canceling an appointment, and so on.

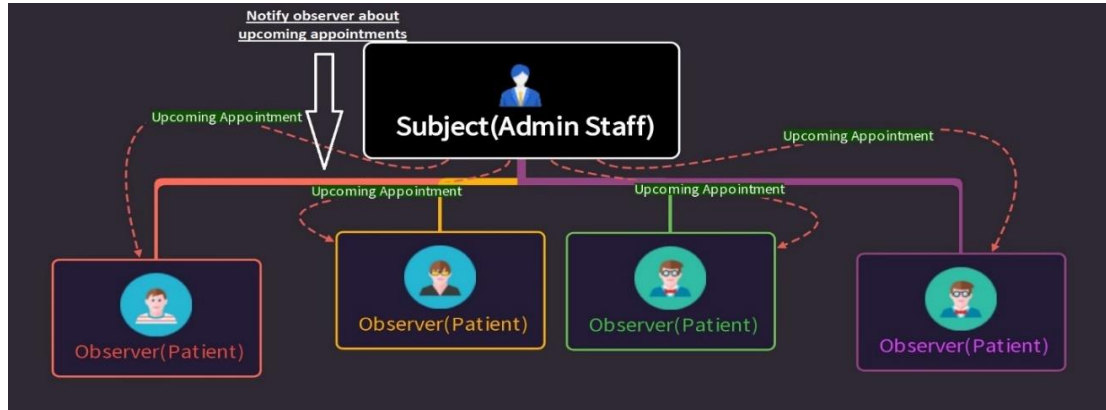


Figure 5.9 Observer Pattern

5.3.2.3 Multi-Tier Architecture:

A multi-tier or n-tier architecture is a collection of software components organized in levels (tiers) to fulfill various functions. The most common multi-tier design is a three-tier system, which consists of a data management layer (typically covering one or more database servers), an application tier (business logic), and a client tier (interface functionality). VAXSYS uses a three-tier system.

Three-tier Architecture:

- 1. Presentation tier:** The application's user interface and communication layer, which is where the end-user interacts with it, is called the presentation tier. Its main purpose is to give information to the user and collect data from them. In VAXSYS, we use React JS in developing this interface which runs on a web browser. [28]
- 2. Application tier:** The logical or intermediate tier is another name for the application layer. This tier handles all dynamic material as well as interactions with the presentation and database tiers. It takes data from the persistent or database tier and processes it. The information will subsequently be returned to the presentation tier. Node JS is used for the implementation of the logical part of the VAXSYS. [28]
- 3. Database tier:** The application's data is saved and maintained at this layer. This tier may also be referred to as the storage tier. VAXSYS uses MongoDB for storing and managing data access. [28]

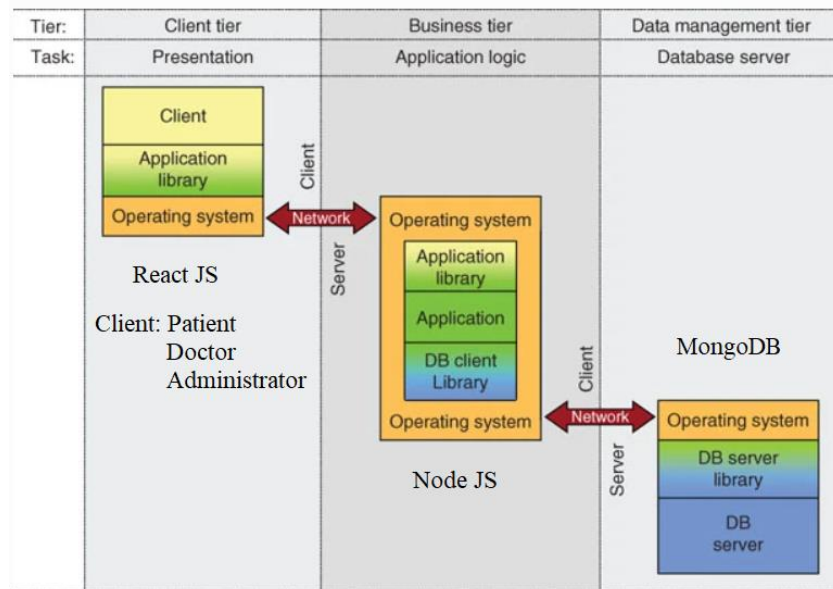


Figure 5.10 Three Tier Architecture [28]

Advantages of using 3-tier for VAXSYS:

- Helps for implementing all the layers concurrently i.e., fast development
- Improper working of any tier doesn't affect the functionality of other tier i.e., reliability.

6. PROOF OF CONCEPT

For the communication and update with the team members we used Trello Board for this project.

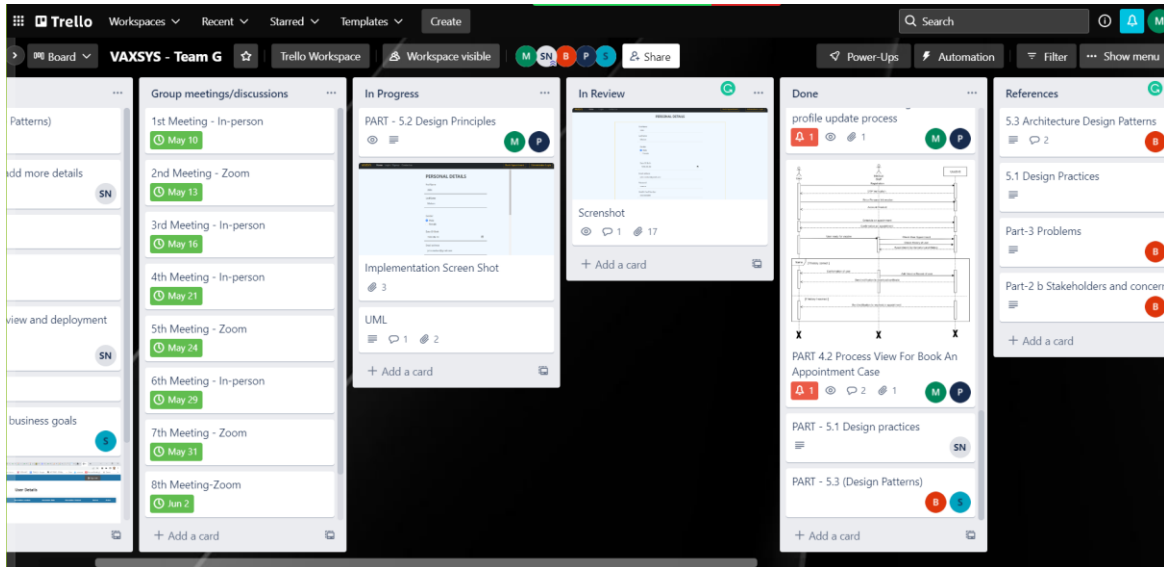


Figure 6.1 Trello Board

For the VAXSYS application, we implement login and registration functionality but don't consider it as our requirement because it's not an application-specific requirement it's a general requirement that every application provides.

Sign Up/Registration:

In the below-mentioned snapshot of registration of users using the for VAXSYS app first-time users has to register their selves to use the VAVSYS application to book an appointment.

MongoDB Atlas is a cloud-based database hosting service that allows developers to host their databases in the cloud.

Data from our application is stored in the database, as seen in the image. The VAXSYS API accepts JSON data and stores it in the database.

VAXSYS Home Login Contact us Book Appointment Administrator Login

PERSONAL DETAILS

FirstName
John

LastName
Molson

Gender
☒ Male
☐ Female

Date Of Birth
1998-03-04

Email address
john.molson@gmail.com

Password

Health Card Number
H54HD45DR

Telephone

Figure 6.2 Signup Page-1

VAXSYS Home Login Contact us Book Appointment Administrator Login

Email address
john.molson@gmail.com

Password

Health Card Number
H54HD45DR

Telephone
514587840

Apt Number
1102

Street
du fort

City
Montreal

Postal Code
H3H 2N7

Save

Figure 6.3 Signup Page-2



Figure 6.4 Postman for Signup Page

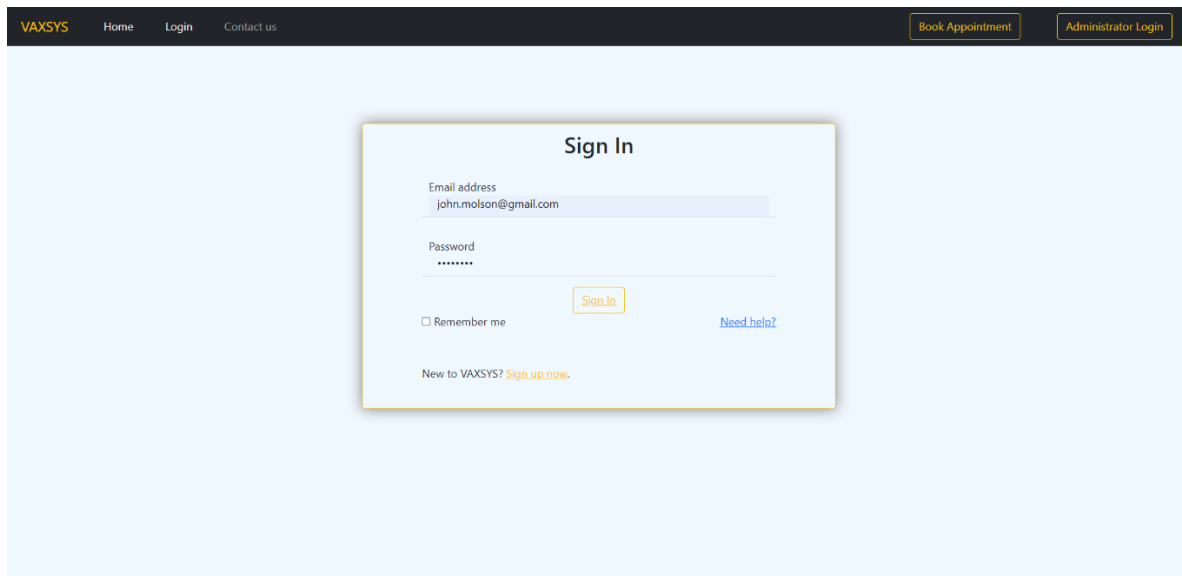
```

_id: ObjectId('62a35c106707ed20109fcc1e')
> medicalRecords: Array
  firstName: "John"
  lastName: "Molson"
  DOB: 1998-03-04T00:00:00.000+00:00
  role: "patient"
  email: "john.molson@gmail.com"
  password: "$2a$10$GmVwwRntJjfkZvL4dJ6yO.u0R2/OiBS.Nr1TEKddZU3x9uU/8XWUO"
  telephone: "514587840"
  address: "1102 du fort Montreal"
  postalcode: "H3H 2N7"
  healthcard: "H54HD45DR"
  gender: "Male"
  createdAt: 2022-06-10T14:58:24.203+00:00
  __v: 0
  
```

Figure 6.5 MongoDB for Signup Page

Sign In/Login:

The below-mentioned snapshots represent the login functionality of the VAXSYS application.



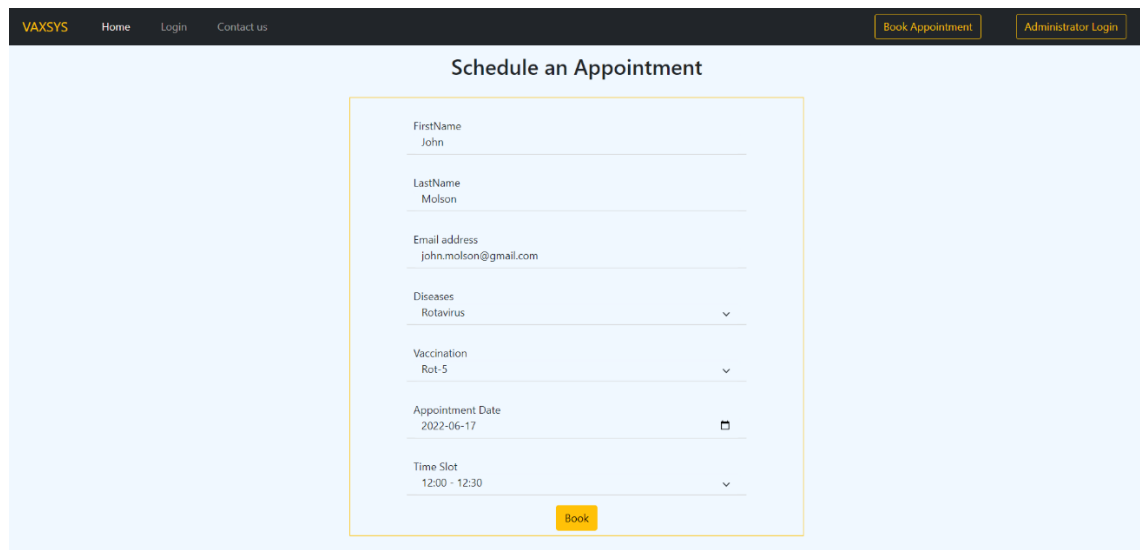
The screenshot shows the VAXSYS application's Sign In page. The header includes the VAXSYS logo and navigation links: Home, Login, and Contact us. On the right, there are buttons for 'Book Appointment' and 'Administrator Login'. The main content area features a 'Sign In' form with fields for 'Email address' (containing 'john.molson@gmail.com') and 'Password' (masked with dots). Below the password field is a 'Remember me' checkbox and a 'Sign In' button. A link for 'Need help?' is also present. At the bottom of the form, it says 'New to VAXSYS? Sign up now.'

Figure 6.6 Sign in Page

6.1 Requirement 1: Users can book an appointment.

The above-mentioned figures show how users can book an appointment for the vaccination. To book an appointment user must have to login into the system if the user tries to book an appointment without login, then the system will redirect user to the login page. User can book appointment for other people like for his children or wife etc.

Every entry of each user stored in the MongoDB database with all required details as shown in the figure.



The screenshot shows the VAXSYS application's 'Schedule an Appointment' page. The header is identical to the previous page. The main content area features a form for scheduling an appointment. The form includes fields for 'FirstName' (John), 'LastName' (Molson), and 'Email address' (john.molson@gmail.com). It also has dropdown menus for 'Diseases' (Rotavirus), 'Vaccination' (Rot-5), and 'Time Slot' (12:00 - 12:30). The 'Appointment Date' is set to 2022-06-17. A 'Book' button is located at the bottom of the form.

Figure 6.7 Book an appointment page

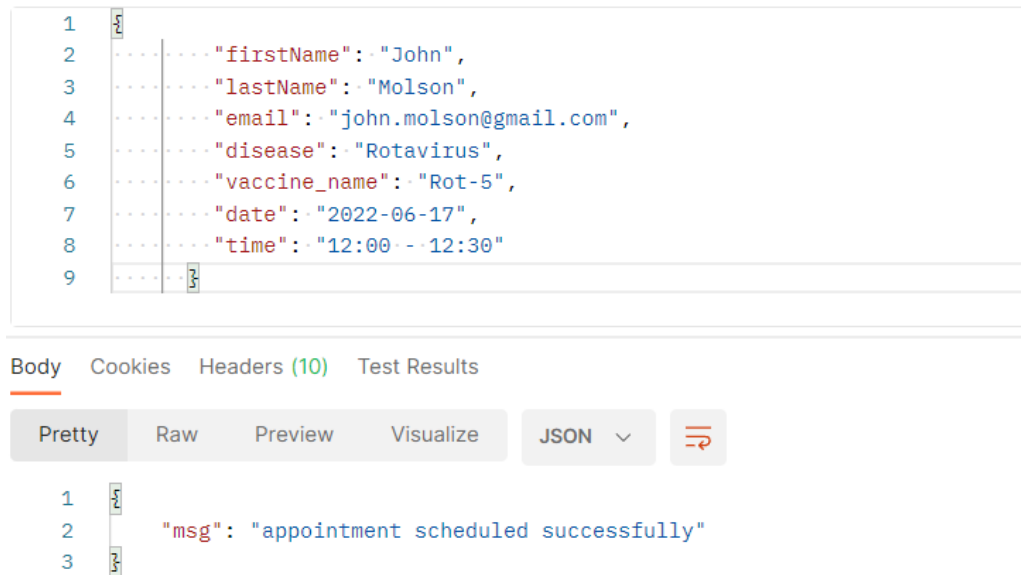


Figure 6.8 Postman for Book an appointment page



Figure 6.9 MongoDB Book an appointment page

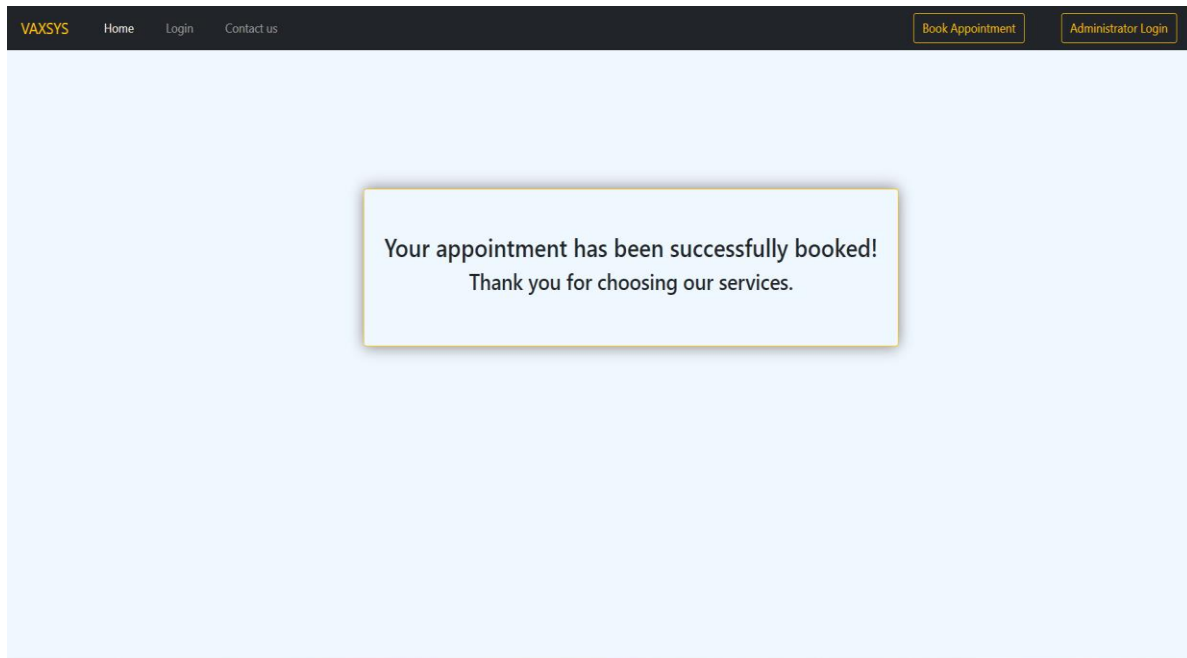


Figure 6.10 Appointment Confirmation Page

6.2 Requirement 2: Users can see their future appointment.

The below-mentioned figures show how users can see his all appointments that book through the VAXSYS system.

For each user, we maintained separate entries in appointment database table and using get method we fetch details of specific user from unique Id that provide at a time of registration.

FIRST NAME	LAST NAME	EMAIL	DISEASES	VACCINATION	APPOINTMENT DATE	TIME SLOT	RESCHEDULE?
Dan	Timothy	dtimothy@gmail.com	Covid19	Pfizer	2022-06-15	12:00 - 12:30	
Junior	Timothy	dtimoty@hotmail.com	Polio	DTaP-HB-IPV-Hib	2022-06-15	12:30 - 13:00	

Figure 6.11 User Appointment Dashboard

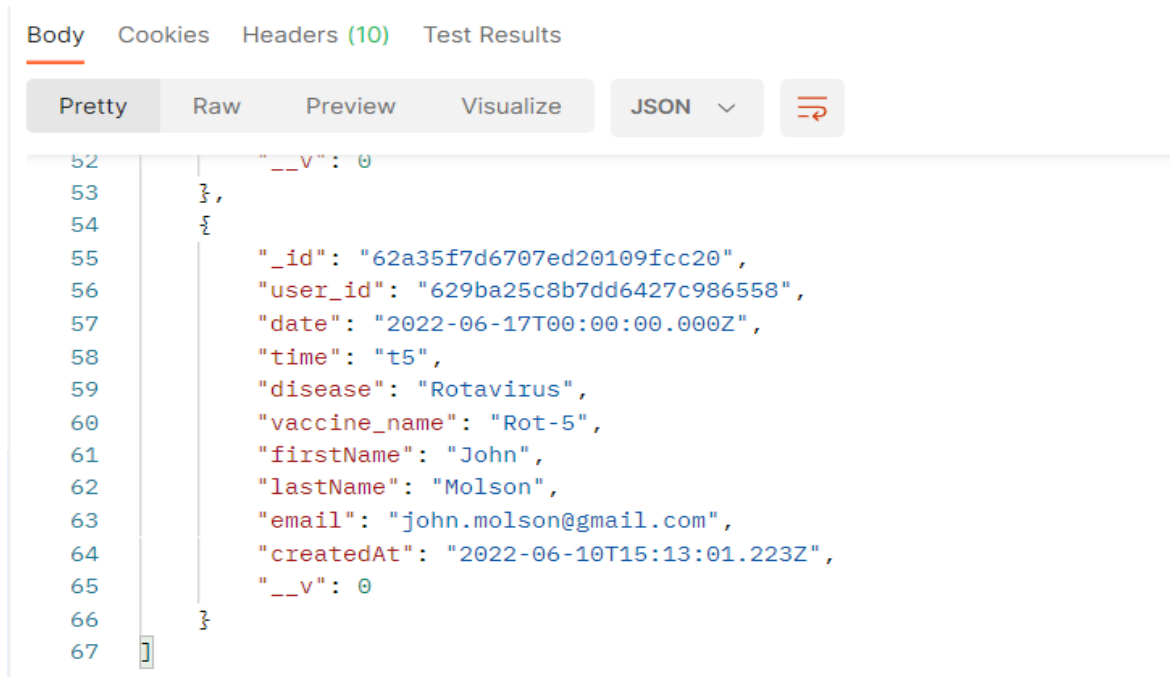


Figure 6.12 Postman for User Appointment Dashboard

6.3 Requirement 3: User can view their profile with all details.

The above-mentioned figures show that after registration user can view their profile and can also update it.

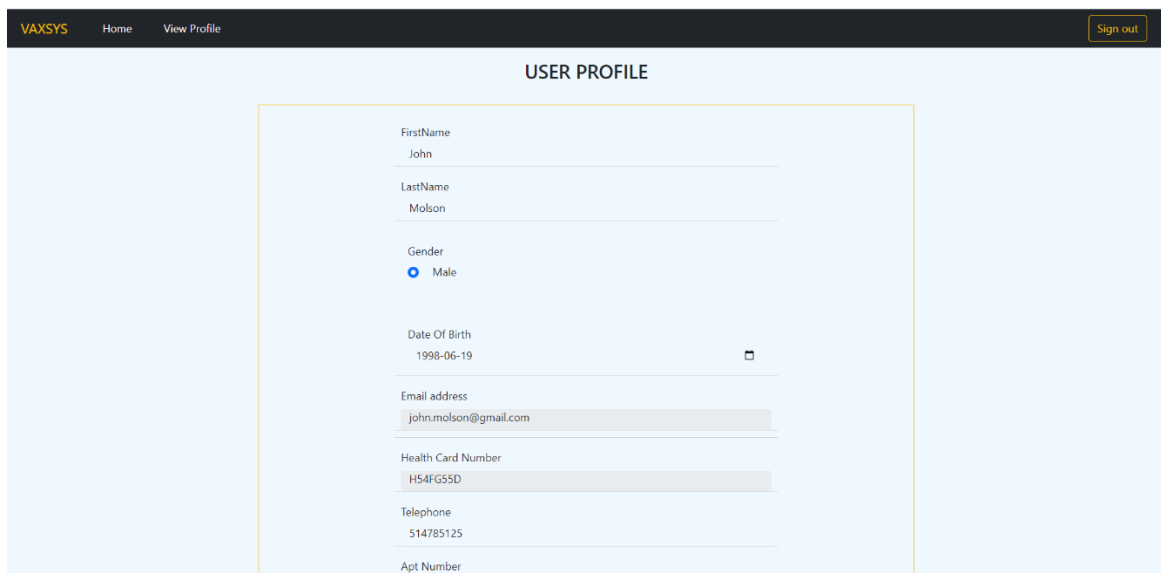


Figure 6.13 View Profile Page-1

The screenshot shows the 'View Profile' page of the VAXSYS application. The header includes the VAXSYS logo, navigation links for Home and View Profile, and a Sign out button. The main content area displays a user profile for 'John Molson' with the following details:

- Date Of Birth: 1998-06-19
- Email address: john.molson@gmail.com
- Health Card Number: H54FG55D
- Telephone: 514785125
- Apt Number: 1102
- Street: du fort
- City: Montreal
- Postal Code: H3H 2N7

Figure 6.14 View Profile Page-2

6.4 Requirement 4: Admin can see all appointments by particular date.

The above-mentioned figures represent the dashboard for the hospital admin. The administrator can see all appointment for a particular date.

The screenshot shows the 'Admin Login' page of the VAXSYS application. The header includes the VAXSYS logo, navigation links for Home, Login, and Contact us, and buttons for Book Appointment and Administrator Login. The main content area features a central 'Admin Login' form with the following fields:


- UserName: admin
- Password: ****


A 'Sign In' button is located at the bottom right of the form.

Figure 6.15 Admin Login Page

SCHEDULED APPOINTMENTS							
FIRST NAME	LAST NAME	EMAIL	DISEASES	VACCINATION	APPOINTMENT DATE	TIME SLOT	ACCEPT / CANCEL
Jaynil	Savani	jaynil@gmail.com	Typhoid	IV-14	2021-02-02T00:00:00.000Z	10:00 - 10:30	<input checked="" type="checkbox"/> <input type="checkbox"/>
Manthan	Moradiya	manthan@gmail.com	Measles	Moderna	2022-06-19T00:00:00.000Z	1:00 - 1:30	<input checked="" type="checkbox"/> <input type="checkbox"/>
Parth	Navsariwala	parth@gmail.com	Rotavirus	Pneu-C-10	2022-06-24T00:00:00.000Z	11:00 - 11:30	<input checked="" type="checkbox"/> <input type="checkbox"/>
John	Molson	john.molson@gmail.com	Rotavirus	Rot-5	2022-06-17T00:00:00.000Z	12:00 - 12:30	<input checked="" type="checkbox"/> <input type="checkbox"/>

Figure 6.16 Admin Dashboard Page

Body Cookies Headers (10) Test Results  Sta

Pretty Raw Preview Visualize JSON 

```
1  {
2
3      "_id": "629e8563e0c3b516d0116768",
4      "user_id": "629ba25c8b7dd6427c986558",
5      "date": "2021-02-02T00:00:00.000Z",
6      "time": "10:00 - 10:30",
7      "disease": "Typhoid",
8      "vaccine_name": "IV-14",
9      "firstName": "Jaynil",
10     "lastName": "Savani",
11     "email": "jaynil@gmail.com",
12     "createdAt": "2022-06-06T22:53:23.995Z",
13     "__v": 0
14 },
15 {
16     "_id": "629e884fe0c3b516d0116769",
17     "user_id": "629ba25c8b7dd6427c986558",
18     "date": "2022-06-19T00:00:00.000Z",
19     "time": "1:00 - 1:30",
20     "disease": "Measles",
21     "vaccine_name": "Moderna"
```

Figure 6.17 Postman for Admin Dashboard

6.5 Requirement 5: Preventing users from scheduling multiple appointments for the same vaccine.

One of the main functionality of the VAXSYS system is to preventing user from scheduling multiple appointments for the same vaccine.

The screenshot displays the VAXSYS web application interface. At the top, a dark navigation bar contains the 'VAXSYS' logo and links for 'Home', 'Login', and 'Contact us'. On the right side of this bar are buttons for 'Book Appointment' and 'Administrator Login'. A central alert box, titled 'localhost:3000 says', contains the message: 'You already booked vaccine slot for the same (date/person/disease) !!!!!'. Below the alert, a form is visible with the following fields: 'FirstName' (John), 'LastName' (Molson), 'Email address' (john.molson@gmail.com), 'Diseases' (Rotavirus), 'Vaccination' (Rot-5), 'Appointment Date' (2022-06-17), and 'Time Slot' (12:30 - 13:00). A yellow 'Book' button is positioned at the bottom of the form.

Figure 6.18 User Vaccine Alert Page

7. Glossary

Accessibility	It is the practice of making your websites usable by as many people as possible. The concept of accessible design and practice of accessible development ensures both "direct access" (i.e. unassisted) and "indirect access" meaning compatibility with a person's assistive technology. [6]
Accuracy	accuracy is how close or far off a given set of measurements (observations or readings) are to their true value. [7]
Adaptability	It refers to users that can substantially customize the system through tailoring activities by themselves. [8]
Agility	It is the ability of a business as a whole to respond quickly to changes, especially external changes. [9]
Authenticity	Authentication is the process of determining whether someone or something is, in fact, who or what it says it is. Authentication technology provides access control for systems by checking to see if a user's credentials match the credentials in a database of authorized users or in a data authentication server. [10]
Availability	It is part of reliability and is expressed as the ratio of the available system time to the total working time. [11]
Confidentiality	The context of computer systems allows authorized users to access sensitive and protected data. [12]
Correctness	The ability of software products to perform their exact tasks, as defined by their specification. [13]
Data Crawler	A data crawler, often known as a spider or web crawler, is an Internet bot that regularly browses the World Wide Web in order to build search engine indices. [14]
Dependability	Dependability is the ability to provide services that can be trusted within a time period. [15]

Effectiveness	A system's effectiveness is a measure of its ability to perform the functions necessary to achieve goals or objectives. [16]
Extensibility	Extensibility allows required modifications at the appropriate locations to be made without undesirable side effects.
Fault tolerance	A fault Tolerance is the property that enables a system to continue operating properly in the event of the failure of one or more faults within some of its components. [17]
Interoperability	Interoperability is an attribute of the system or part of the system that is responsible for its operation and the transmission of data and its exchange with other external systems. [18]
Maintainability	Maintainability is the ability of the system to support changes. [18]
Modifiability	Modifiability determines how many common changes need to be made to the system to make changes to each individual item. [18]
Modularity	Modularity is the measure of the extent to which software is composed of separate, interchangeable components, each of which accomplishes one function and contains everything necessary to accomplish this. [19]
Observability	Observability is a measure of how well internal states of a system can be inferred from knowledge of its external outputs. [20]
Performance	Performance shows the response of the system to performing certain actions for a certain period. [18]
Portability	The ease with which a software system can be adapted to run on computers other than the one for which it was designed.
Privacy	Privacy is the ability of an individual or group to seclude themselves or information about themselves, and thereby express themselves selectively. [21]

RAMQ (<i>Régie de l'assurance maladie du Québec</i>)	RAMQ is responsible for the sound management of the Québec Health Insurance Plan and of the Public Prescription Drug Insurance Plan. As part of its mission, it manages the eligibility of persons for the plans, monitors the remuneration of health professionals and facilitates access to health care. [22]
Reliability	The ability of a system to perform its requested functions under stated conditions whenever required having a long mean time between failures.
Reusability	Reusability is a chance of using a component or system in other components/systems with small or no change. [18]
Scalability	Scalability is the ability of the system to handle load increases without decreasing performance, or the possibility to rapidly increase the load. [18]
Security	Security is responsible for the ability of the system to reduce the likelihood of malicious or accidental actions as well as the possibility of theft or loss of information. [18]
Simplicity	Simplicity is the state or quality of being simple. Something easy to understand or explain seems simple, in contrast to something complicated. [23]
Surveillance	Observes and tracks the operations and activities of users, applications and network services on a computer or enterprise systems.
Testability	Testability shows how well the system allows performing tests, according to predefined criteria. [18]
Traceability	Traceability is the capability to trace something. [1] In some cases, it is interpreted as the ability to verify the history, location, or application of an item by means of documented recorded identification. [24]
Usability	Usability is one of the most important attributes, because, unlike in cases with other attributes, users can see directly how well this attribute of the system is worked out. [18]

User Interface

The user interface is the point at which human users interact with a computer, website, or application.[\[25\]](#)

8. References

1. P. Kamthan, Summer 2022, "Understanding Software Architecture", Department of Computer Science and Software Engineering, Concordia University.
2. P. Kamthan, Summer 2022, "Introduction to Software Product Quality", Department of Computer Science and Software Engineering, Concordia University.
3. P. Kamthan, Summer 2021, "Views and Viewpoints of Software Architecture", Department of Computer Science and Software Engineering, Concordia University.
4. "ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes -- Requirements engineering," in ISO/IEC/IEEE 29148:2018(E) , vol., no., pp.1-104, 30 Nov. 2018, doi: 10.1109/IEEESTD.2018.8559686.
5. ISO/IEC/IEEE Systems and software engineering -- Architecture description," in ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000) , vol., no., pp.1-46, 1 Dec. 2011, doi: 10.1109/IEEESTD.2011.6129467.
6. 'Accessibility'. Wikipedia, 4 June 2022. Wikipedia, <https://en.wikipedia.org/w/index.php?title=Accessibility&oldid=1091414992>.
7. 'Accuracy and Precision'. Wikipedia, 23 May 2022. Wikipedia, <https://en.wikipedia.org/w/index.php?title=Accuracy and precision&oldid=1089401220>.
8. 'Adaptation (Computer Science)'. Wikipedia, 12 June 2021. Wikipedia, [https://en.wikipedia.org/w/index.php?title=Adaptation \(computer science\)&oldid=1028140073](https://en.wikipedia.org/w/index.php?title=Adaptation (computer science)&oldid=1028140073).
9. 'Agile vs. Agility. What Are the Differences?' AgileXL, <https://agilexl.com/blog/agile-vs-agility-what-are-the-difference>. Accessed 7 June 2022.
10. 'What Is Authentication?' SearchSecurity, <https://www.techtarget.com/searchsecurity/definition/authentication>. Accessed 9 June 2022.
11. '12 Software Architecture Quality Attributes and Their Types'. Syndicode - Custom Software Development Company, 3 May 2018, <https://syndicode.com/blog/12-software-architecture-quality-attributes/>.
12. 'What Is Confidentiality? - Definition from Techopedia'. Techopedia.Com, <http://www.techopedia.com/definition/10254/confidentiality>. Accessed 7 June 2022.
13. ACCU. Quality Matters: Correctness, Robustness and Reliability. https://accu.org/journals/overload/17/93/wilson_1585/#:~:text=Correctness%20%3A%20The%20ability%20of%20software,concern%20encompassing%20correctness%20and%20robustness. Accessed 7 June 2022.
14. Data Crawler - Web Scraping Tool & Free Web Crawlers | Octoparse. <https://www.octoparse.com/DataCrawler#:~:text=What%20Is%20a%20Data%20Crawler,the%20data%20all%20the%20time>. Accessed 7 June 2022.
15. 'Dependability'. Wikipedia, 27 Mar. 2022. Wikipedia, <https://en.wikipedia.org/w/index.php?title=Dependability&oldid=1079499702>.
16. Effectiveness (Glossary) - SEBoK. [https://www.sebokwiki.org/wiki/Effectiveness_\(glossary\)#:~:text=From%20SEBoK,Ackoff%201971](https://www.sebokwiki.org/wiki/Effectiveness_(glossary)#:~:text=From%20SEBoK,Ackoff%201971). Accessed 9 June 2022.

17. 'Fault Tolerance'. Wikipedia, 9 Apr. 2022. Wikipedia, https://en.wikipedia.org/w/index.php?title=Fault_tolerance&oldid=1081781629.
18. '12 Software Architecture Quality Attributes and Their Types'. Syndicode - Custom Software Development Company, 3 May 2018, <https://syndicode.com/blog/12-software-architecture-quality-attributes/>.
19. Software Quality Attributes | ADVOSS. <https://advoss.com/software-quality-attributes/./index.html>. Accessed 9 June 2022.
20. 'Observability'. Wikipedia, 8 June 2022. Wikipedia, <https://en.wikipedia.org/w/index.php?title=Observability&oldid=1092084539>.
21. 'Privacy'. Wikipedia, 4 June 2022. Wikipedia, <https://en.wikipedia.org/w/index.php?title=Privacy&oldid=1091421451>.
22. About RAMQ | Régie de l'assurance Maladie Du Québec (RAMQ). <https://www.ramq.gouv.qc.ca/en/about-us>. Accessed 9 June 2022.
23. 'Simplicity'. Wikipedia, 17 Apr. 2022. Wikipedia, <https://en.wikipedia.org/w/index.php?title=Simplicity&oldid=1083188164>.
24. 'Traceability'. Wikipedia, 2 Mar. 2022. Wikipedia, <https://en.wikipedia.org/w/index.php?title=Traceability&oldid=1074841257>.
25. 'What Is User Interface (UI)? | Indeed.Com'. Indeed Career Guide, <https://www.indeed.com/career-advice/career-development/user-interface>. Accessed 9 June 2022.
26. What Is MVC? Simple Explanation. www.youtube.com, <https://www.youtube.com/watch?v=pCvZtjoRq1I>. Accessed 29 May 2022.
27. What Is the Observer Pattern? (Software Design Patterns). www.youtube.com, <https://www.youtube.com/watch?v=1ANfXJdXe34>. Accessed 9 June 2022.
28. Schuldt, Heiko. 'Multi-Tier Architecture'. Encyclopedia of Database Systems, edited by LING LIU and M. TAMER ÖZSU, Springer US, 2009, pp. 1862–65. Springer Link, https://doi.org/10.1007/978-0-387-39940-9_652.
29. SOLID: The First 5 Principles of Object Oriented Design | DigitalOcean. https://www.digitalocean.com/community/conceptual_articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design. Accessed 9 June 2022.
30. Different Types of Software Design Principles. <https://www.dotnettricks.com/learn/designpatterns/different-types-of-software-design-principles>. Accessed 29 May 2022.
31. 'List of Software Architecture Styles and Patterns'. Wikipedia, 2 May 2022. Wikipedia, https://en.wikipedia.org/w/index.php?title=List_of_software_architecture_styles_and_patterns&oldid=1085784250.
32. 'Product Design Trends in 2020: Modular Hardware vs. Modular Software'. Altium, 5 Nov. 2019, <https://resources.altium.com/p/product-design-trends-in-2020-modular-hardware-vs-modular-software>.
33. 'Effective Modular Design in Software Engineering'. GeeksforGeeks, 20 Dec. 2019, <https://www.geeksforgeeks.org/effective-modular-design-in-software-engineering/>.
34. Camel Case vs. Snake Case vs. Pascal Case — Naming Conventions | Khalil Stemmler. <https://khalilstemmler.com/blogs/camel-case-snake-case-pascal-case/>. Accessed 1 June 2022.

35. MVC Architecture Image.
https://www.guru99.com/images/1/122118_0445_MVCTutorial1.png
36. Rawal, Kajal. 'SOLID — Design Principles'. Medium, 15 Feb.
<https://kajalrawal.medium.com/solid-design-principles-82fc4bdebb8>.
37. 'What Is Duplicate Code?' Codegrip, 20 Nov. 2019,
<https://www.codegrip.tech/productivity/what-is-duplicate-code/>.
38. 'Unit Testing | Software Testing'. GeeksforGeeks, 18 Apr. 2019,
<https://www.geeksforgeeks.org/unit-testing-software-testing/>
39. Python Indentation - AskPython. 19 June 2019,
<https://www.askpython.com/python/python-indentation>.