Project Synopsis

Title.

Vaccine Management System Codename: DrugLordv1.10

Problem Statement

The main objective of this System is to maintain records of vaccinations to monitor the quality of vaccines and timely assurance in rural areas where health centres are not easily accessible.

Why this Topic?

To reduce the paperwork required for maintaining records instead digitizing them for data visualization to minimize reading of records and provide easy access to third-party bodies to get, deliver, and administer vaccines.

Objective and Scope.

- To reduce the paperwork required
- To maintain a detailed record of each individual with ease
- To analyse data easily for the person/body using this system
- To create a system that manages both deliveries along with a facility for vaccine administration

Methodology for developing project.

In this system, I am going to use Extreme Programming for developing an appropriate system as a solution for rapidly changing requirements

Advantages: Communication, Simple, Easy, Agile.

Proposed Architecture

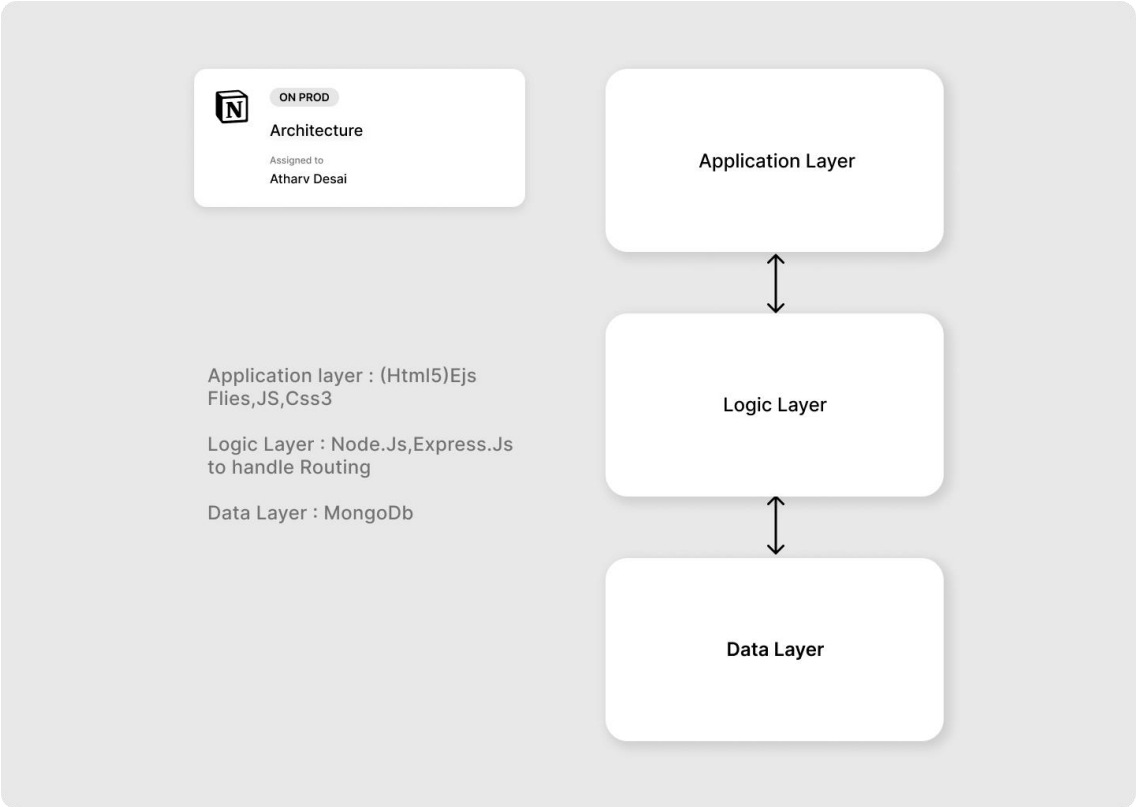


Figure 1.1 Architecture Design

Requirements

Software Requirements

- Front-end : HTML5, CSS , JS,Tailwindcss
- Back-end :,ExpressJs, NodeJS, MongoDb.
- Operating System: Windows 7.0 +.

Hardware Requirements

- Processor: Intel Core Duo 2.0 GHz or more.
- RAM: 2 GB or more.
- Monitor: 17 CRT or LCD.
- Hard disk: 500 GB or more.
- Keyboard: Normal or multimedia.

Platform

Visual Studio Code

Contribution

This system will help people in rural areas get access to vaccines along with reduced paperwork and record generation and keep each individual to date.

Conclusion

This system will help to reduce paperwork and provide access to people in rural areas.

1.1 Background

The current Vaccination Systems which are used are not centralized i.e., the data generated is confined to an organization or a company except due to a recent pandemic newer methods are cultivated

References

- https://www.servicenow.com/solutions/vaccine-management.htm
- https://www2.deloitte.com/us/en/pages/public-sector/solutions/vaccine-management-system.htm
- https://www.intelex.com/products/applications/vaccine-management}{Intelex Vaccine Solution

1.2 Objective

The main purpose of this project is to help benefit NGOs in providing vaccine access to rural areas people without using many resources, by providing verified personnel of NGOs or any individual to request access to medical vaccines under the guidance of a medical practitioner.

1.3 Purpose

- Provide rural areas with exposure to vaccine
- Create a platform where people can register for an appointment, and delivery as well as monitor the records.

1.4 Application

The idea can be fundamentally used in any management system of products in any medical field.

1.5 Scope

Creating a platform for all domestic producers to sell their vaccines to local consumers, NGOs, etc, and to simplify the delivery process, regulations and management.

1.6 Achievements

Through this project, I will get to learn various technologies used to develop a system. This system will give a digital platform to the all-Domestic producers of vaccines which will increase the vaccine production rate and simplify the delivery process as well as help them to manage their data. Once they get satisfactory results from the system, users will start trusting the system and eventually, most of the tasks will get digital. Encourage B2B and C2C relation

The number of Technologies available for the implementation is listed below

- ASP.NET(MVC) Core 5.0+
    - Is a framework developed by Microsoft to create enterprise-grade web apps in a short amount of time
    - Languages : C#, HTML, CSS, jQuery
    - Backend: Microsoft SQL Serve
- Django (MVC)
    - Framework to create scalable single-page web apps,web-API using Python as the main programming language
    - Language : Python,HTML,CSS,jQuery,JS
    - Backend: MongoDB
- Ruby on Rails
    - Framework based on Rudby Programming Language for easy file directory structure and dataflow subroutines
    - Language: Ruby,HTML,CSS,JS
    - Backend: MySql Server
- Node.js
    - Backend as well and Frontend using vanilla Javascript as main primary language useful in API creation
    - Language: JS (EMAC5), HTML(ejs), JS,jQuery
    - Backend: MongoDB
- Flask
    - Python Backend Framework, useful in developing fast prototypes of ML and AI searches, etc
    - Language: Python

For my current project, I am going to use Node.js as a development platform for easy implementation of the requirements proposed.

Why Node.js?

Easy routing through Express.JS library and ejs content to render with minimum requirements to hosting a server and is platform independent

Required: Node.js is installed

3.1 Problem Definition

What to expect about the system?

The system will manage and record all the deliveries along with orderly administration of vaccines It will also keep a detailed record of all the domestic suppliers, users requesting appointments, providers buying vaccines, and local company producers producing vaccines

- [Users, Providers, Producers]
    - Will be allowed to log in as each unique individual account.
- [Producers, Providers]
    - Help manage, and administer vaccine drives with all the information centralized for easy fetch
    - The system will keep a detailed record of all individual's information both registered for vaccine drives as well as seeking appointments
- [Provider]
    - To set up appointments for the user or use for delivery purposes.
- [Producer]
    - Set up vaccine stocks for order processing
- [User]
    - Book a nearby appointment for quick administration

3.1.1 Sub Problems

- For Users
    - How users can add their information?
    - How to schedule and select an appointment?
    - How to get the vaccine?

- For Providers
    - How to get verified?
    - How to select your order?
    - How do schedule appointments for the people?

- For Producers
    - How to set up your vaccine Stocks?
    - How to process Orders?
    - How to Deliver?

3.1.2 Problem Description.

The system will help find required clients for domestic producers as well as enable recipients to get vaccine appointments in their nearby locality

3.2 Requirements Specification

3.2.1 Requirement analysis

Identify stakeholders i.e., in case the people who are going to use this app

- Capture requirements
- Holding One-One interviews
- Conduct Group Workshops
- Get Feedbacks
- Build Small Prototypes

For the current situation, I used the Feedback method to identify the requirements for the project using Google Forms as a means to collect the data

The below figures are the collected data that was generated.

Responses were based on:

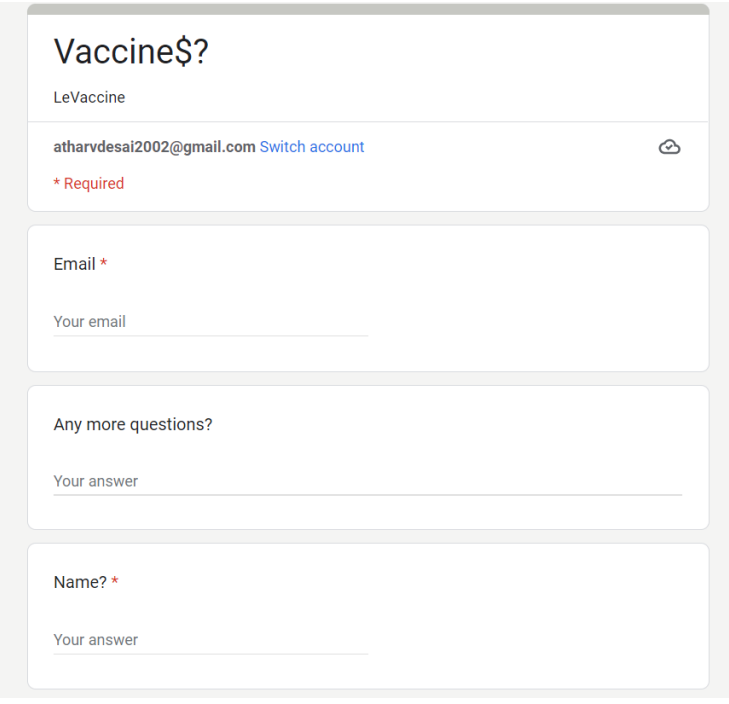


Figure 3.2.1

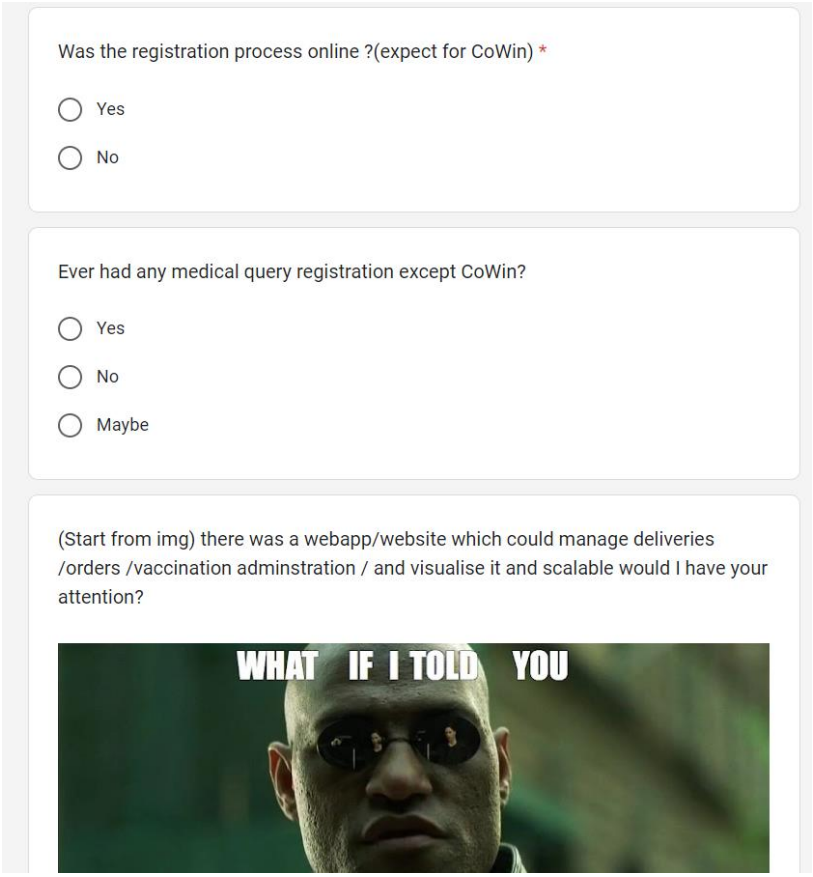


Figure 3.2.2

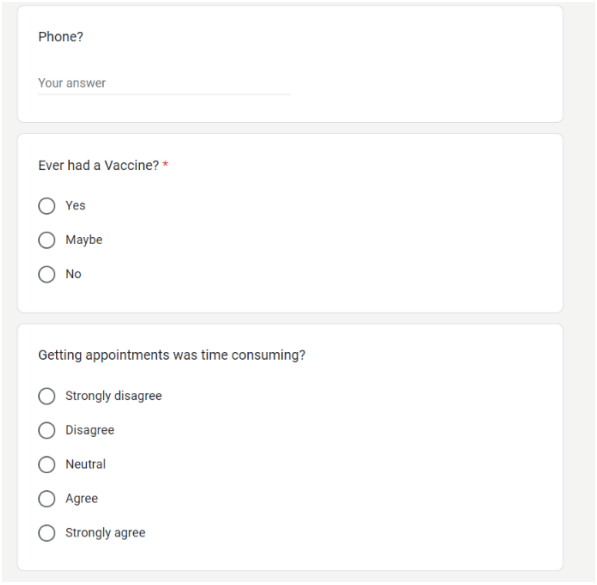


Figure 3.2.3

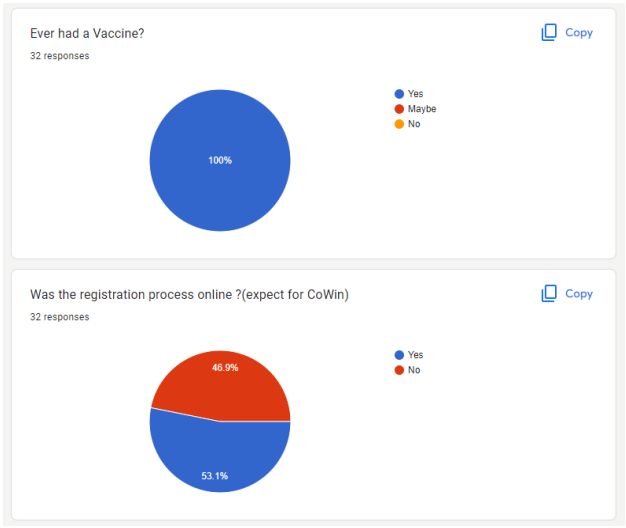


Figure 3.2.4

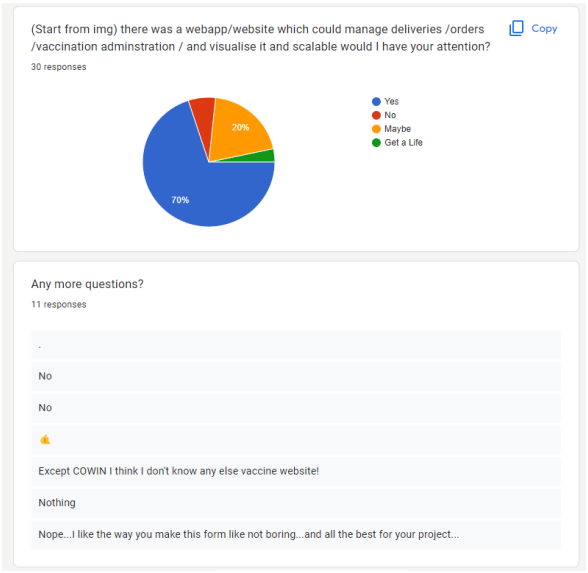


Figure 3.2.5

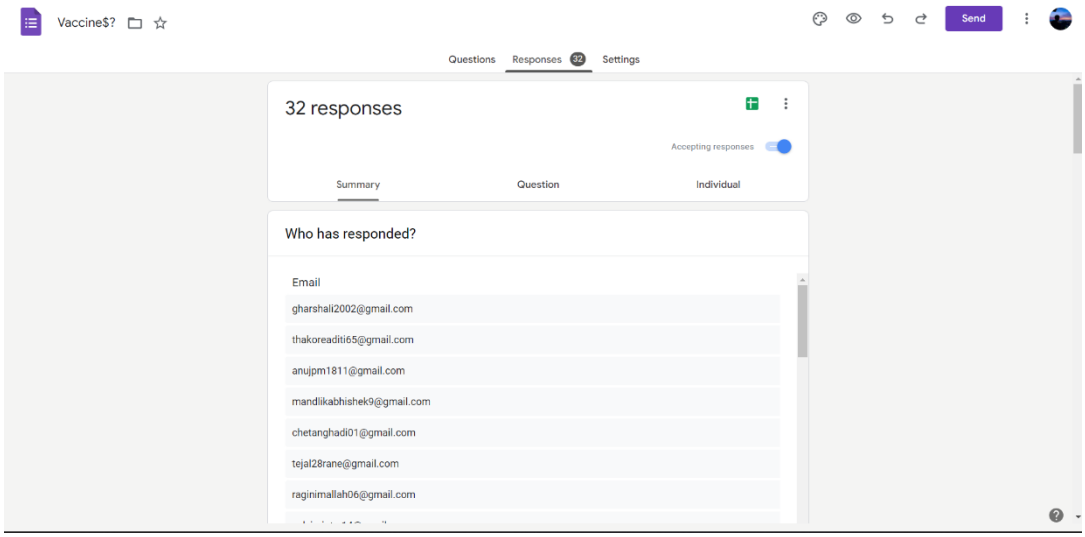


Figure 3.2.6

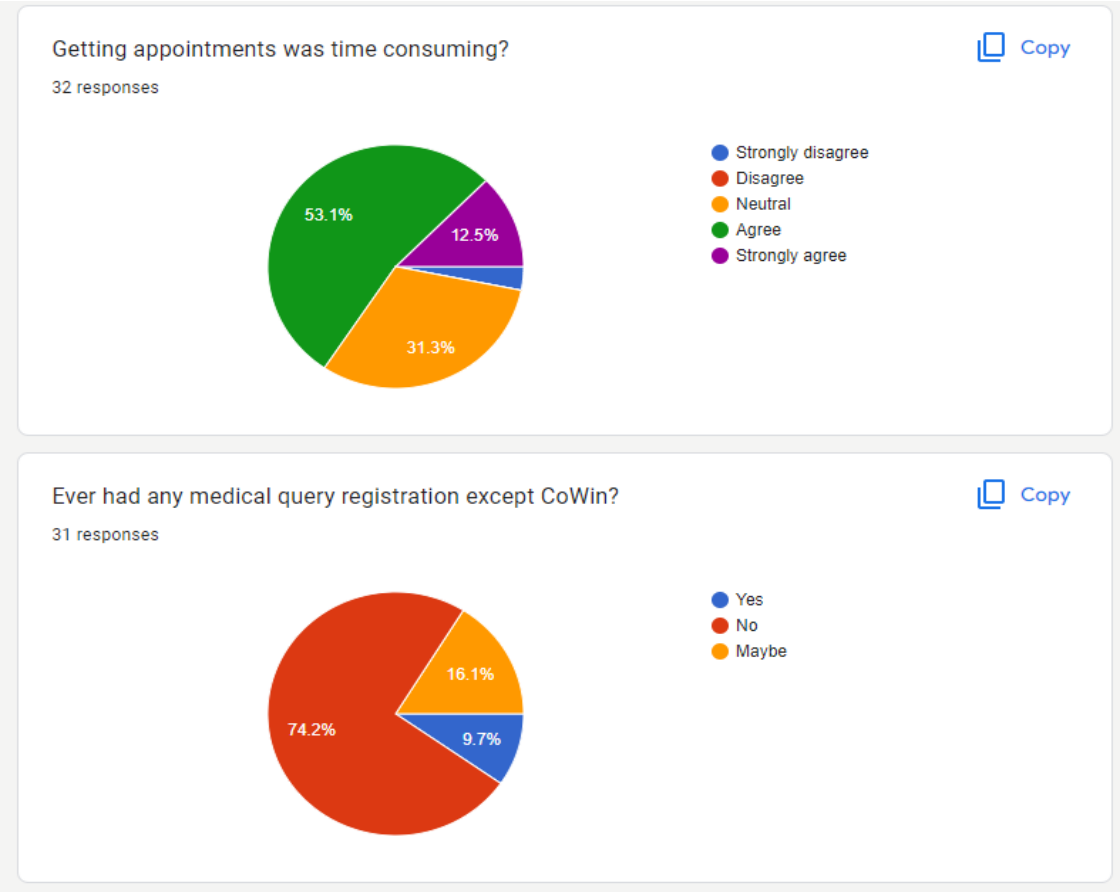


Figure 3.2.7

3.2.2 Functional Requirements

1. The system should allow users to fill up for an appointment in their locality
2. The system should keep a detailed record of user's info
3. The system should allow users to note any problems after vaccination is completed
4. The system should provide Producers, NGOs, and Companies to order and sell their vaccines
5. The system should keep a detailed record of all domestic producers, providers and along with the info of users

3.2.3 Non-Functional Requirements

1. Usability: Should be user-friendly and only required detail should be shown in a minimal way
2. Reliability: The system should be user-friendly to use.
3. Scalability: To increase the load of data handling
4. Flexibility: can run on any Platform

3.2.3 Sub- Systems

- Login

Function: Used for login
Input: User, Provider, Producer Login Details
Pre-Condition: None
Source: From User, Provider, Producer
Post-Condition: Redirected to Dashboard if Credentials are valid
Exception: An error message is shown if credentials don’t match

- Register

Function: Used for registering an account
Input: User, Provider, Producer all details
Pre-Condition: None
Source: From User, Provider, Producer
Post-Condition: Redirected to log in if details are valid and correct
Exception: An error message is shown if any field is wrong or not valid

- Users-Info Update

Function: Used for updating account
Input: User, Provider, Producer all details to change
Pre-Condition: None
Source: From User, Provider, Producer
Post-Condition: The user's details are updated
Exception: The error message is shown if any field is wrong or not valid

- Post-Review

Function: Used for Posting Reviews
Input: User Reviews
Pre-Condition: None
Source: From User
Post-Condition: User's reviews get added
Exception: The error message is shown if anything goes wrong

- Book Appointment
  Function: Used for booking an appointment
  Input: User to book appointment
  Pre-Condition: None
  Source: From User
  Post-Condition: Appointment is booked
  Exception: The error message is shown if anything goes wrong

- Set appointments

Function: Used for setting appointments
Input: Provider, Producer all details for setting the appointment
Pre-Condition: None
Source: Provider
Post-Condition: Appointment is generated
Exception: The error message is shown if anything goes wrong

- Buy Vaccines
  Function: Used for Buying vaccines
  Input: Provider all details for buying vaccines
  Pre-Condition: None
  Source: Provider
  Post-Condition: Order is Placed
  Exception: An error message is shown if anything goes wrong

- Update Vaccine Stocks
  Function: Used for updating vaccine stocks
  Input: Producer all details of vaccine
  Pre-Condition: None
  Source: Producer
  Post-Condition: Vaccine Stocks are updated or created
  Exception: The error message is shown if anything goes wrong

3.3 **Planning & Scheduling** (update)

3.3.1 Activity Table

Activities	Start-Date	End-Date
Synopsis	27/04/22	16/06/22
1. Introduction	20/06/22	25/06/22
2. Survey of Technology	20/06/22	25/06/22
3. Requirement and analysis	27/06/22	15/09/22

3.1 Problem Definition	4/07/22	9/07/22
3.2 Requirement Specification	11/07/22	16/07/22
3.3 Planning & Scheduling	18/07/22	23/07/22
3.4 Hardware & Software Requirements	18/07/22	23/07/22
3.5 Conceptual Models	26/07/22	15/09/22
3.5.1 Data Model	26/07/22	1/08/22
3.5.2 Data Flow	1/08/22	22/08/22
3.5.3 Class Diagram	22/08/22	29/09/22
3.5.4 Use Case Diagram	29/09/22	12/09/22
3.5.5 Sequence Diagram	12/09/22	15/09/22
3.5.6 State Machine Diagram	12/09/22	16/09/22
3.5.7 Activity Diagram	13/09/22	19/09/22
4. System Design	19/09/22	22/09/22
4.1 Interface Design	17/09/22	20/09/22
4.2 Test Cases	18/09/22	22/09/22

Table 3.1

3.3.2 Gantt Chart

Figure 3.1 GANTT Chart

Re-Engineering

Activities	Start-Date	End-Date
1. Re-Engineering	10/01/2022	12/05/2022
2. Introduction	10/01/2022	10/16/2022
3. Survey of Technology	10/08/2022	10/15/2022
4. System Design	10/19/2022	10/29/2022
User		
5. Login/Register	12/05/2022	12/12/2022
6. User Profile	12/12/2022	12/19/2022
7. Book Appointments	01/08/2023	01/15/2023
8. Appointments	01/15/2023	01/22/2023
Provider		
10. Login/Register	12/06/2022	12/13/2022
11. Provider-Profile	12/16/2022	12/23/2022
12. Set Appointments	12/25/2022	01/08/2023
13. Appointments	01/18/2023	01/25/2023
14. Buy-Vaccine	01/22/2023	01/29/2023
15. Orders	01/29/2023	02/05/2023
Producer		
16. Login/Register	12/07/2022	12/14/2022
17. Producer-Profile	12/20/2022	12/27/2022
18. Authorize-Providers	01/22/2023	01/29/2023
19. Process-Orders	02/12/2023	02/13/2023
20. Set-Stocks	01/29/2023	02/05/2023
21. APIs	11/26/2022	12/03/2022

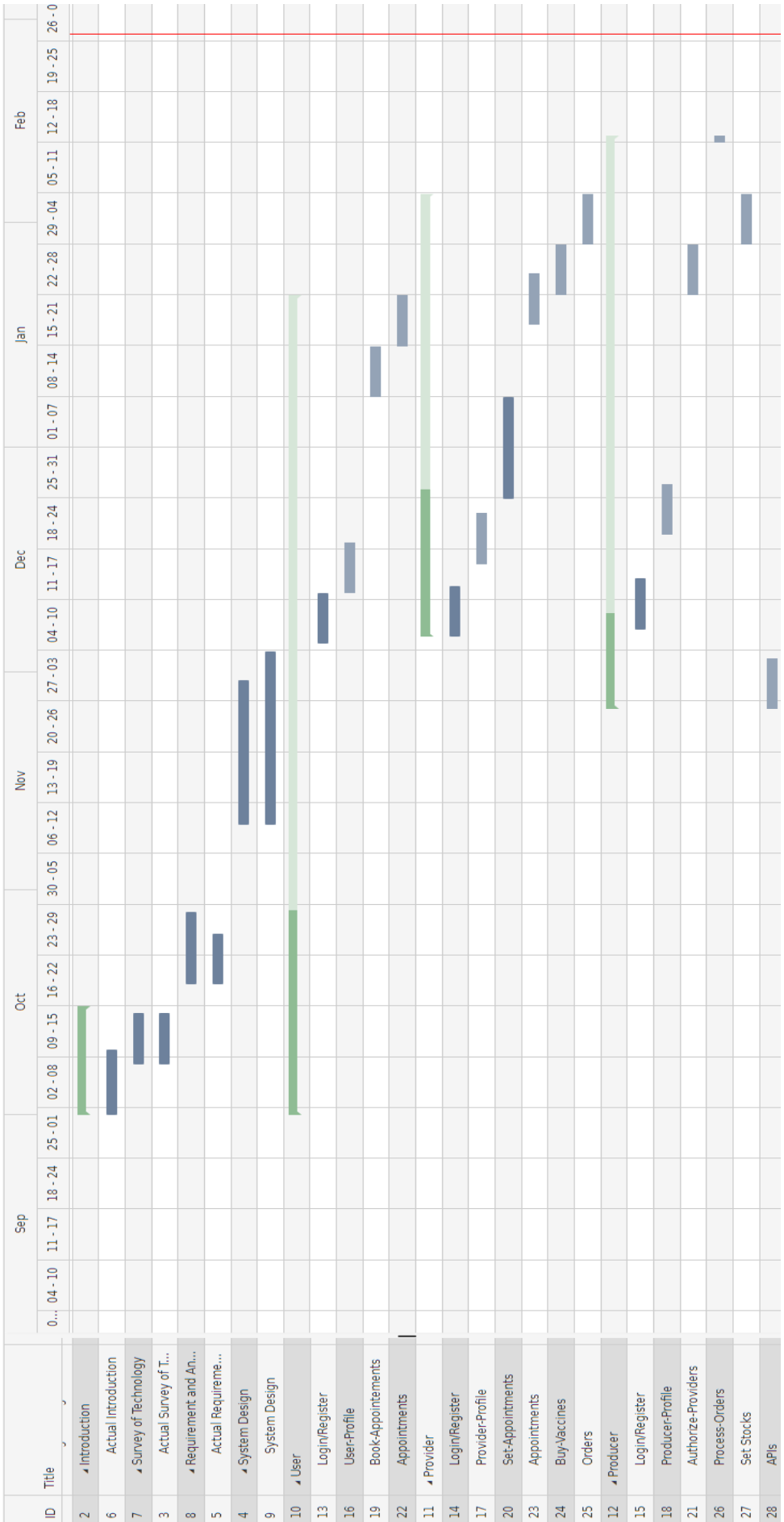

ID
Title
Introduction
Actual Introduction
Survey of Technology
Actual Survey of T...
Requirement and An...
Actual Requireme...
System Design
System Design
User
Login/Register
User-Profile
Book-Appointements
Appointments
Provider
Login/Register
Provider-Profile
Set-Appointments
Appointments
Buy-Vaccines
Orders
Producer
Login/Register
Producer-Profile
Authorize-Providers
Process-Orders
Set Stocks
APIs
Sep
Oct
Nov
Dec
Jan
Feb

Figure 3.2

3.4 Hardware & software requirements

3.5.1 Hardware

- Processor: Intel Core 3.0 2.3 GHz or more.
- RAM: 4GB or more.
- Monitor: 17 CRT or LCD, Plasma, etc.
- Hard-Disk: 256 or more (SSD preferable)
- Keyboard: Normal or multimedia.
- Mouse: Compatible

3.5.2 Software

- System O.S: Window or Linux (Debian or Arch).
- Frontend: HTML(ejs),JS, TailwindCss.
- Backend: Node.js,Express.js
- Database: MongoDB (NoSql)

3.5 Conceptual Model

3.5.1 Data Model

Unlike SQL databases, where you must determine and declare a table's schema before inserting data, MongoDB's collections, by default, do not require their documents to have the same schema. That is:

- The documents in a single collection do not need to have the same set of fields and the data type for a field can differ across documents within a collection.

Embedded Data

Embedded documents capture relationships between data by storing related data in a single document structure. MongoDB documents make it possible to embed document structures in a field or array within a document. These *denormalized* data models allow applications to retrieve and manipulate related data in a single database operation.

The key decision in designing data models for MongoDB applications revolves around the structure of documents and how the application represents relationships between data. MongoDB allows related data to be embedded within a single document.

References :

- Practical MongoDB: Architecting, Developing, and Administering MongoDB Author :Shakuntala Gupta Edward & Navin Sabharwal,
- https://www.mongodb.com/docs/manual/core/data-modeling-introduction/

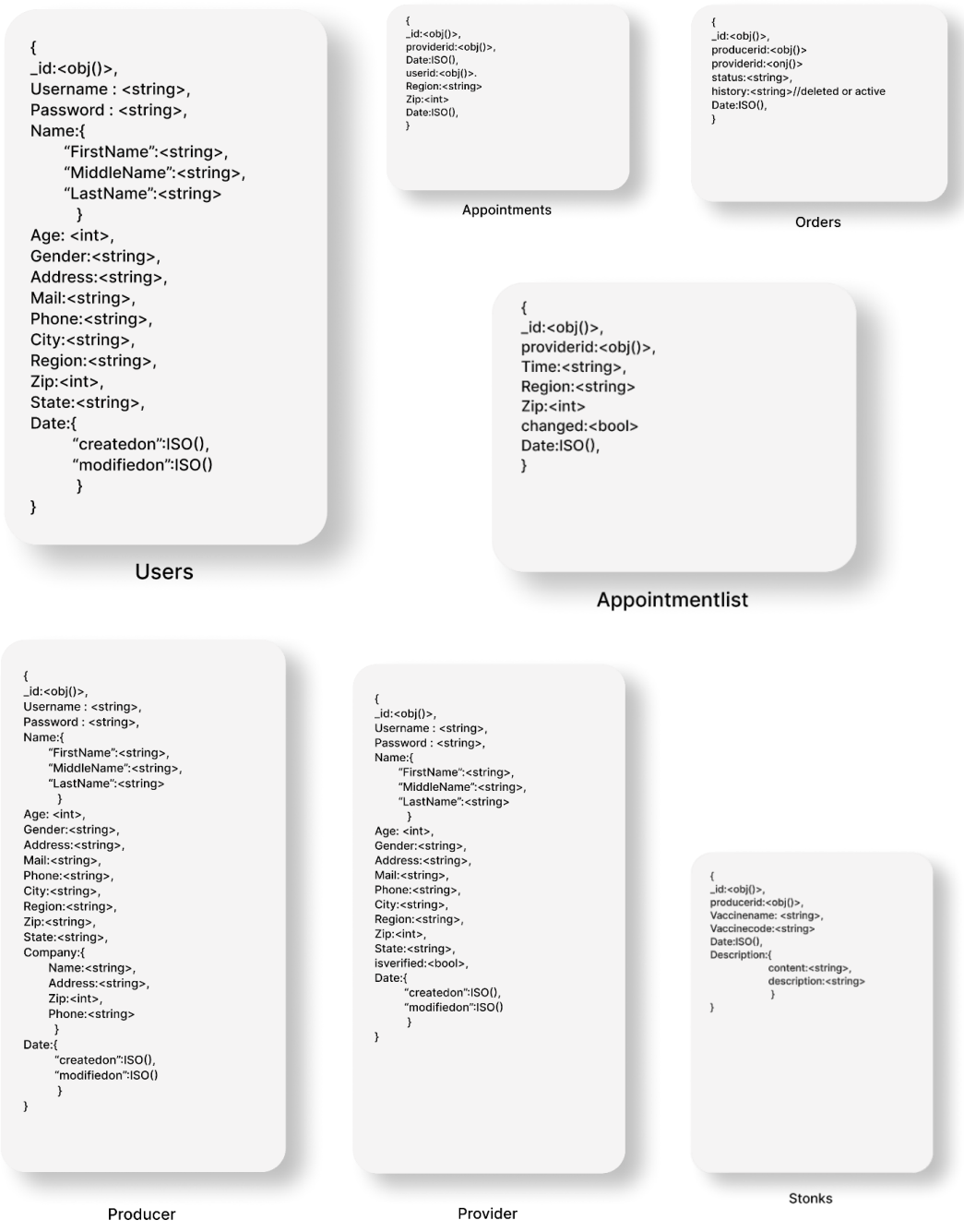


Figure 3.2 Data Model

3.5.2 Data-Flow Diagram

A **data flow diagram** (or DFD) is a graphical representation of the information flow in a business process. It demonstrates how data is transferred from the input to the file storage and reports generation. By visualizing the system flow, the flow charts will give users helpful insights into the process and open up ways to define and improve their business.

References:

- Software Engineering 9th Edition Author : Ian Sommerville

- https://www.lucidchart.com/pages/uml-data-flow-diagram

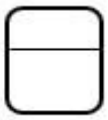
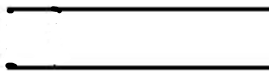
Name	Symbol	Description
Process		A process transforms incoming data flow into outgoing data flow.
Database		Data stores are repositories of data in the system.
Data Flow		Data flows are pipelines through which packets of information flow. Label the arrows with the name of the data that moves through it.
External Entity		External entities are objects outside the system, with which the system communicates

Table 3.2

DFD 0

DFD Level 0 is also called a Context Diagram. It's a basic overview of the whole system or process being analysed or modelled. It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities.

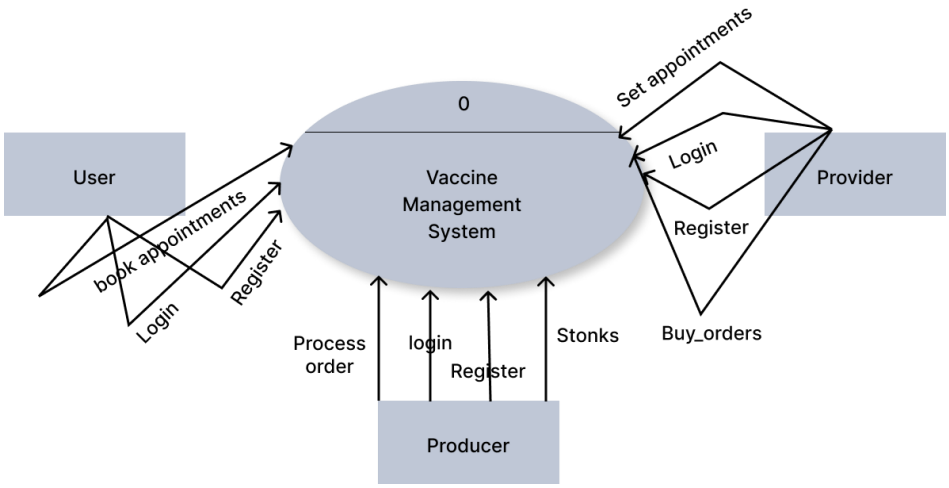


Figure 3.3 DFD 0

DFD 1

In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into subprocesses

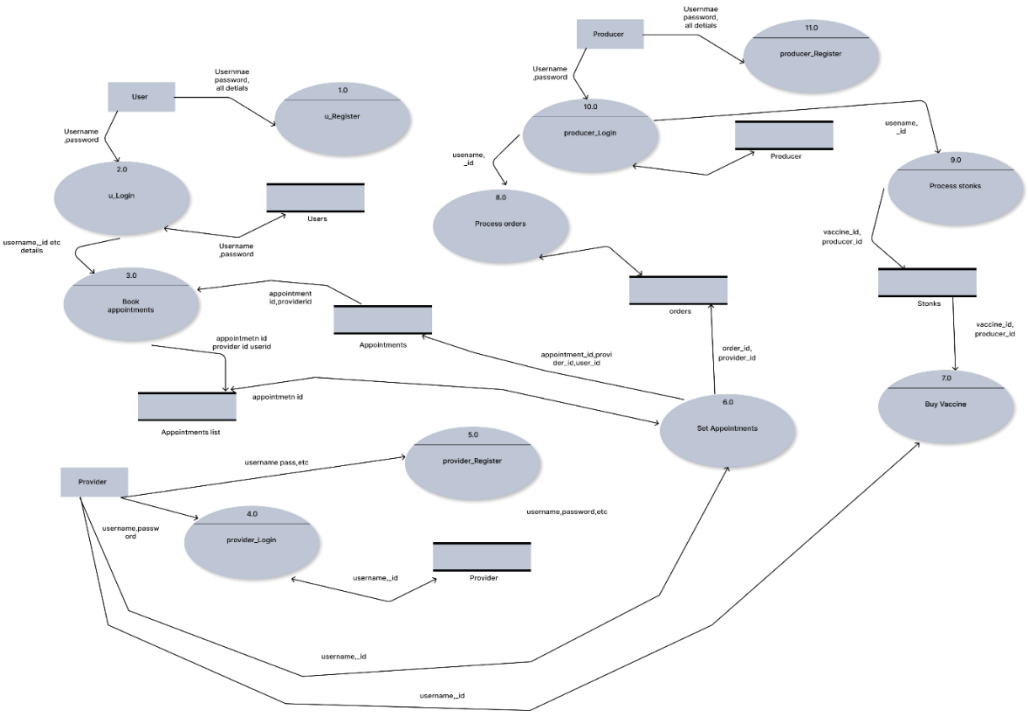


Figure 3.4 DFD1

DFD 2

2-level DFD goes one step deeper into parts of 1-level DFD. It can be used to plan or record the specific/necessary detail about the system's functioning.

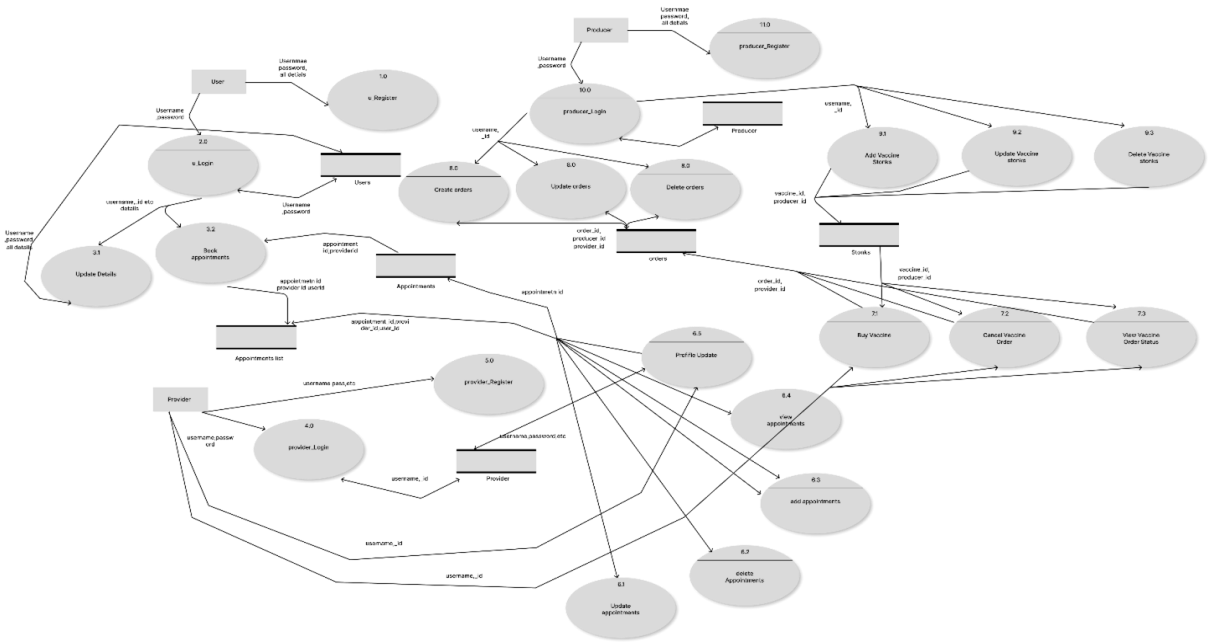


Figure 3.5 DFD 2

3.5.3 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

References:

- Software Engineering 9th Edition Author : Ian Sommerville

- https://www.lucidchart.com/pages/class-diagram

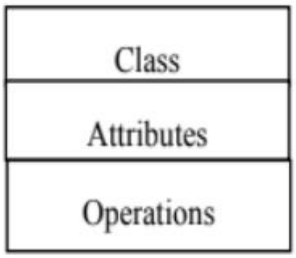
Name	Symbol	Description
Class		Classes and interfaces in UML show architecture and features of the designed system.
Association		Represents the static relationship shared among the objects of two classes

Table 3.3

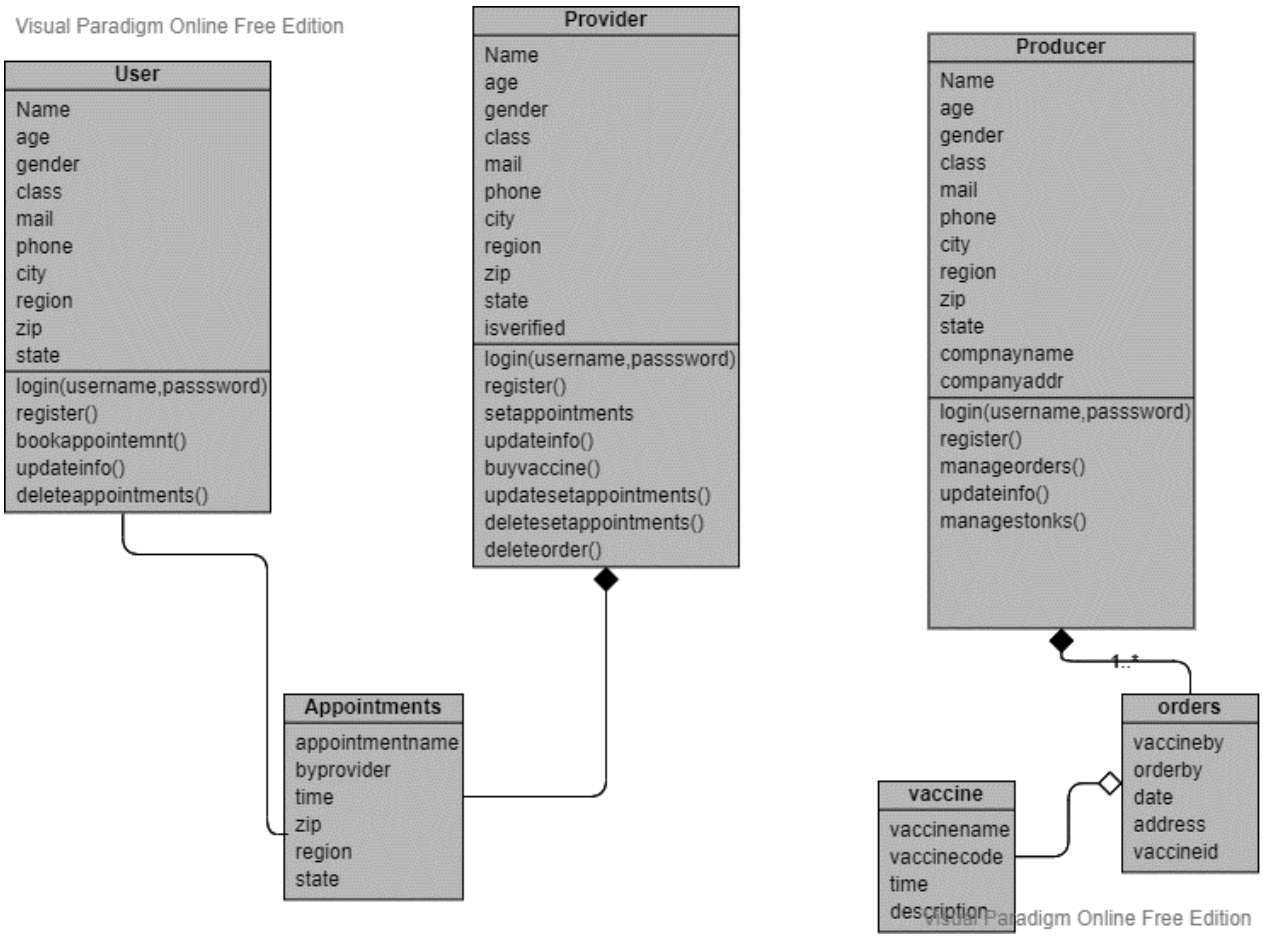


Figure 3.6 Class Diagram

3.5.4 Use-Case Diagram

In the Unified Modelling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

- Scenarios in which your system or application interacts with people, organizations, or external systems
- Goals that your system or application helps those entities (known as actors) achieve
- The scope of your system

References:

- Software Engineering 9th Edition Author : Ian Sommerville
- https://www.lucidchart.com/pages/uml-use-case-diagram

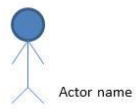
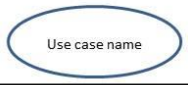
	Actor
	Use case
	Generalization symbol used between actors and between use cases
	Association between actor and use case
	Include relationship between use cases
	Extend relationship between use cases
Symbols in a use case diagram	

Table 3.4

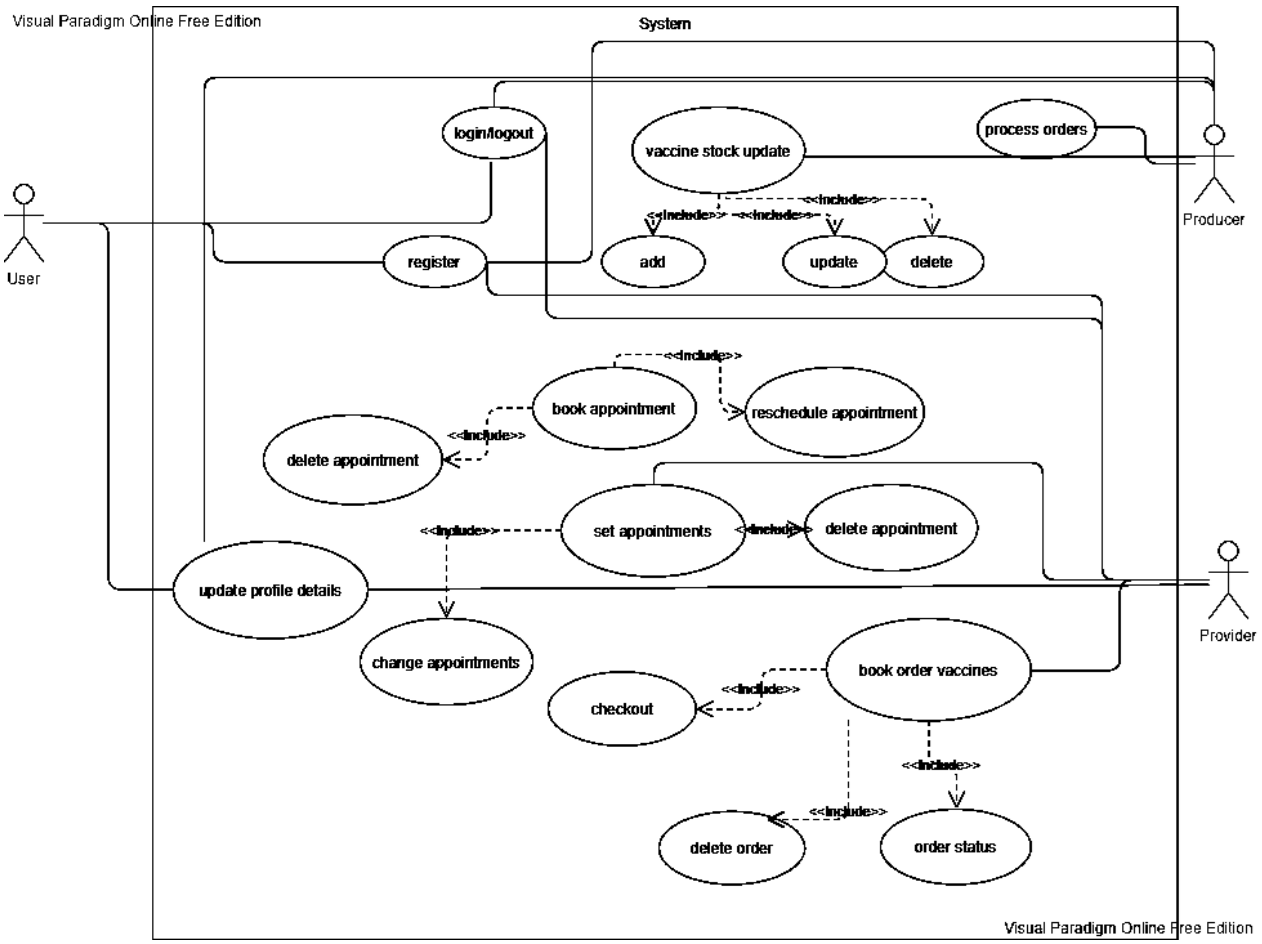


Figure 3.7

Use Case Description:

- Login

  Summary: Used to log in and Logout of the system

  Actor: User, Provider, Producer.

  Pre-condition: None.

  Description: The System waits for the credentials to be entered for validations and necessary for logout

  Exception: If credentials are not valid an error message is thrown

  Post-condition: Display user dashboard

- Register

  Summary: Used for registration

  Actor: User, Provider, Producer.

  Pre-condition: None.

  Description: Is used for creating an account in the service by fill certain details

  Exception: If proper fields are not valid an error message is thrown

  Post-condition: redirect to login

- Book an appointment

  Summary: Used for booking an appointment

  Actor: User.

  Pre-condition: The user should be logged in with proper account details.

  Description: Is used for creating an account in the service by fill certain details

  Exception: If slots get filled during registration error message is shown

  Post-condition: Appointment is booked.

- Book orders

  Summary: Used for buying vaccines.

  Actor: Provider.

  Pre-condition: Provider should be logged in and valid

  Description: For buying vaccines from a domestic producer

  Exception: If proper fields are not valid an error message is thrown

  Post-condition: Order is booked

- Set Appointments

  Summary: Used for setting up appointments

  Actor: Provider.

  Pre-condition: Provider should be logged in and valid

Description: For setting appointments for users in his/her locality

Exception: If proper fields are not valid an error message is thrown

Post-condition: Appointment is successfully registered.

- Process Orders

  Summary: Used for processing orders.

  Actor: Producer.

  Pre-condition: None.

  Description: For processing vaccines orders of a provider.

  Exception: If an internal problem happens error message is thrown.

  Post-condition: Order is successfully processed

3.5.5 Sequence Diagram

UML Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

- Model high-level interaction between active objects in a system
- Model the interaction between object instances within a collaboration that realizes a use case
- Model the interaction between objects within a collaboration that realizes an operation
- Either model generic interactions (showing all possible paths through the interaction) or specific instances of a interaction (showing just one path through the interaction)

References:

- Software Engineering 9th Edition Author : Ian Sommerville

- https://www.lucidchart.com/pages/uml-sequence-diagram

Name	Symbol	Description
Synchronous Message		An instantaneous communication between objects that conveys information, with the expectation that an action will be initiated as a result.
Activation Box		The period during which an object is performing an action.
Object		An object that is created, performs actions, and/or is destroyed during the lifeline

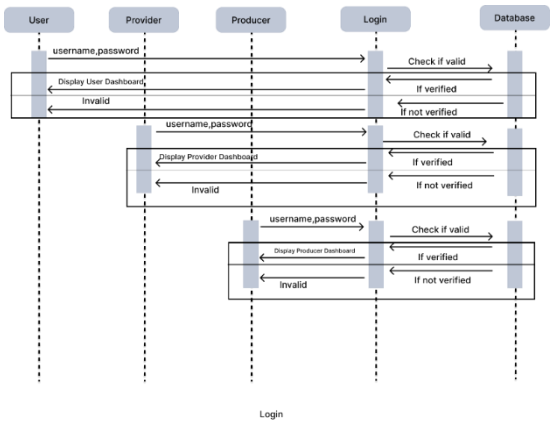


Figure 3.13

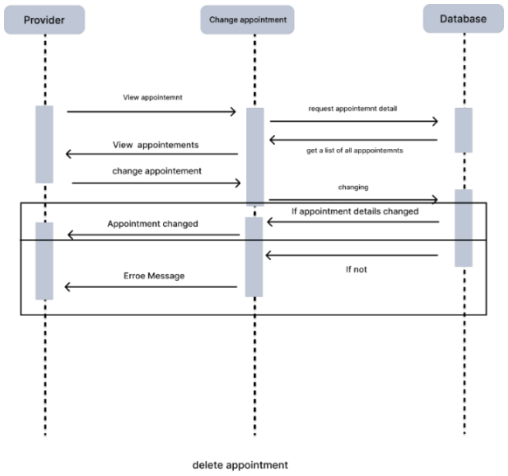


Figure 3.14

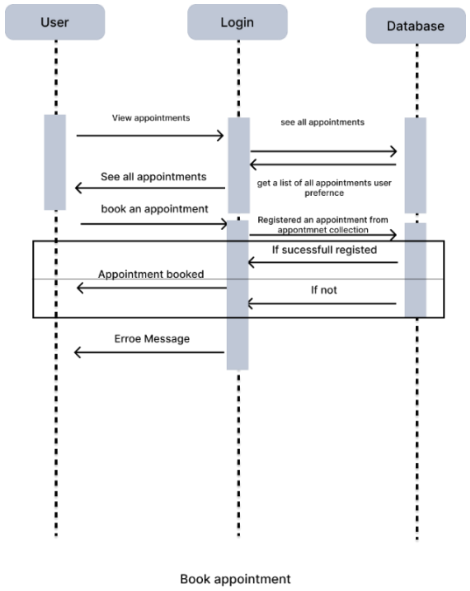


Figure 3.16

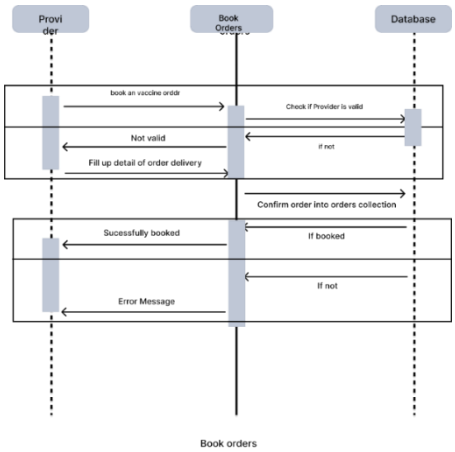
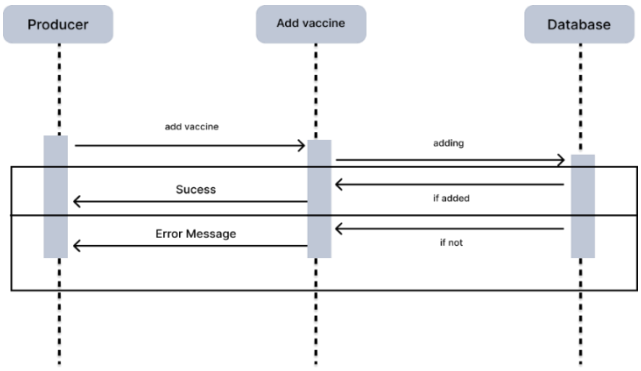
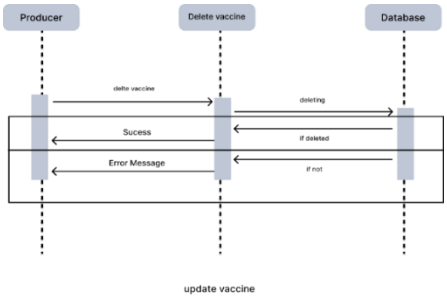


Figure 3.15



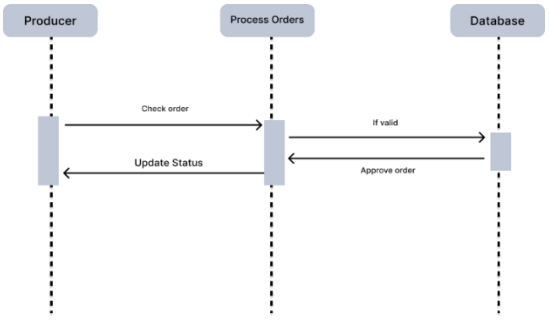
Add vaccine

Figure 3.9



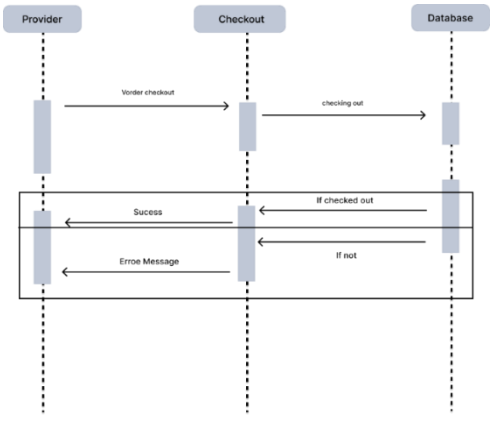
update vaccine

Figure 3.11



Process Orders

Figure 3.10



checkout

Figure 3.12

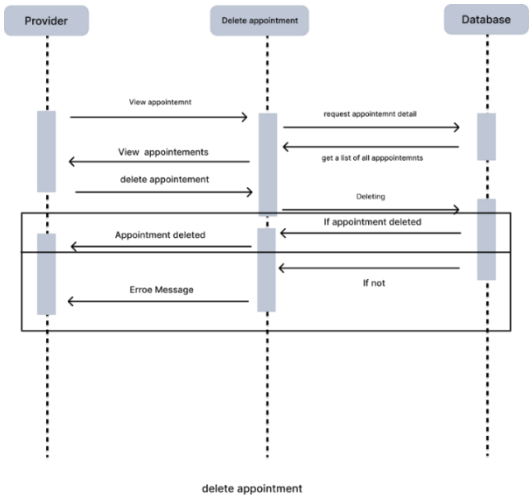


Figure 3.17

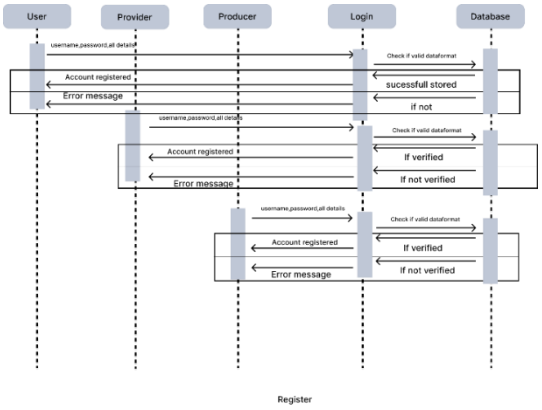


Figure 3.18

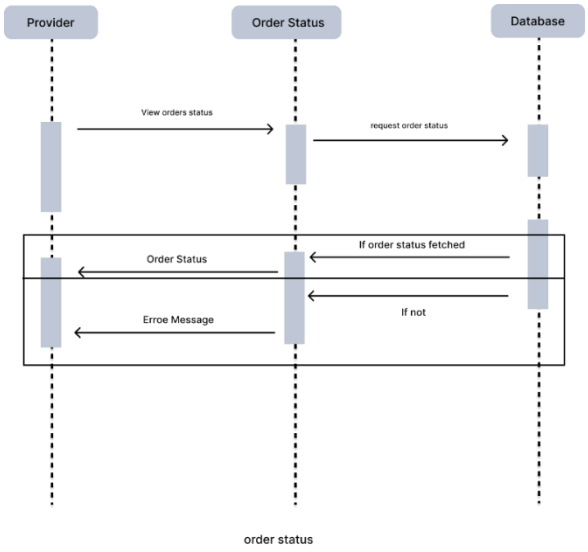


Figure 3.19

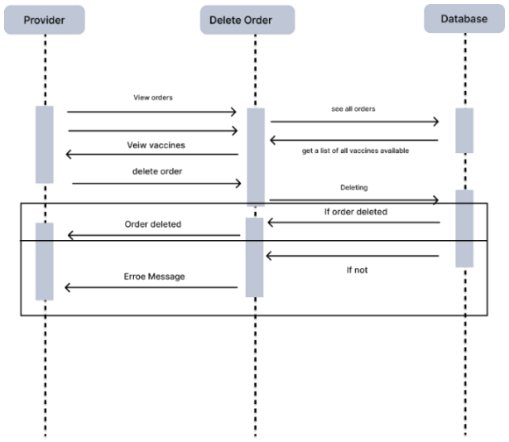


Figure 3.20

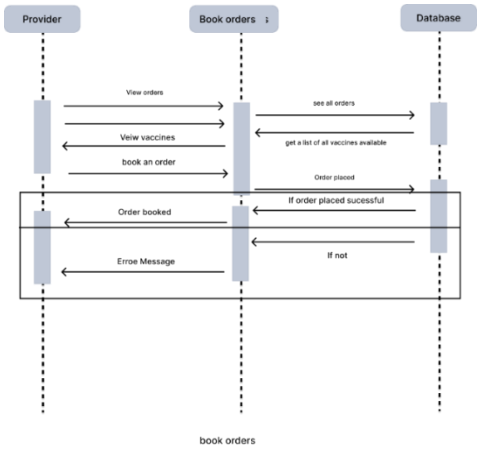


Figure 3.21

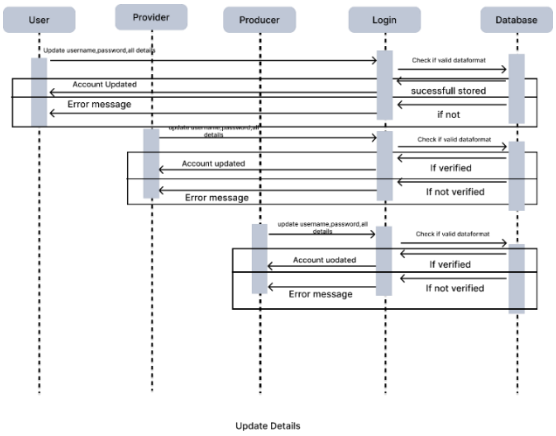
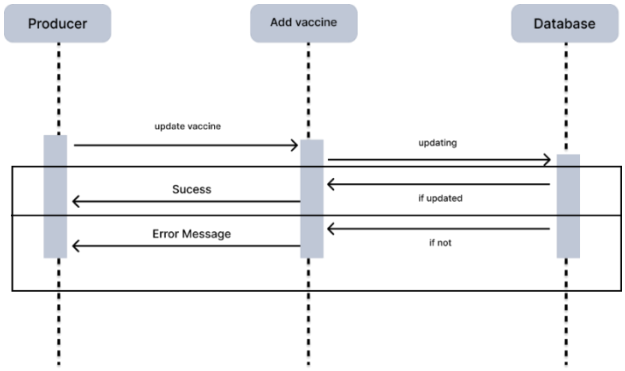
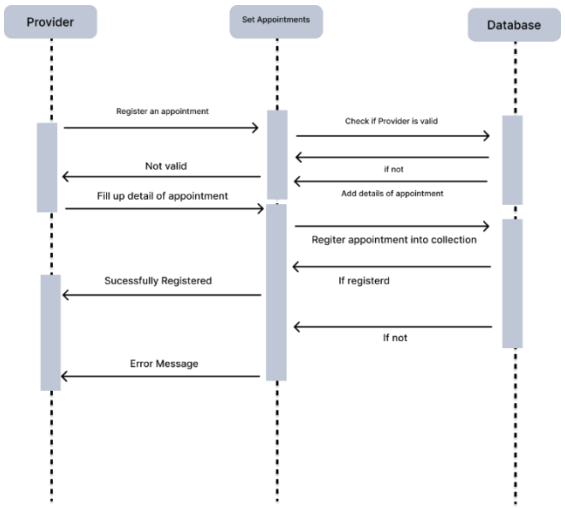


Figure 3.22



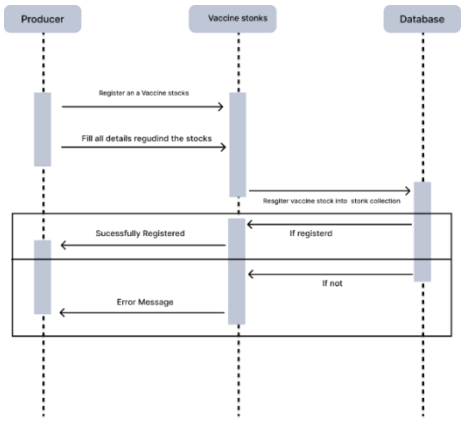
update vaccine

Figure 3.23



Set appointments

Figure 3.24



Vaccine Stocks

Figure 3.25

3.5.5 State Activity Diagram

Activity diagrams present several benefits to users. Consider creating an activity diagram to:

- Demonstrate the logic of an algorithm.
- Describe the steps performed in a UML use case.
- Illustrate a business process or workflow between users and the system.
- Simplify and improve any process by clarifying complicated use cases.
- Model software architecture elements, such as method, function, and operation.

References:

- Software Engineering 9th Edition Author : Ian Sommerville
- https://www.lucidchart.com/pages/uml-state-diagram

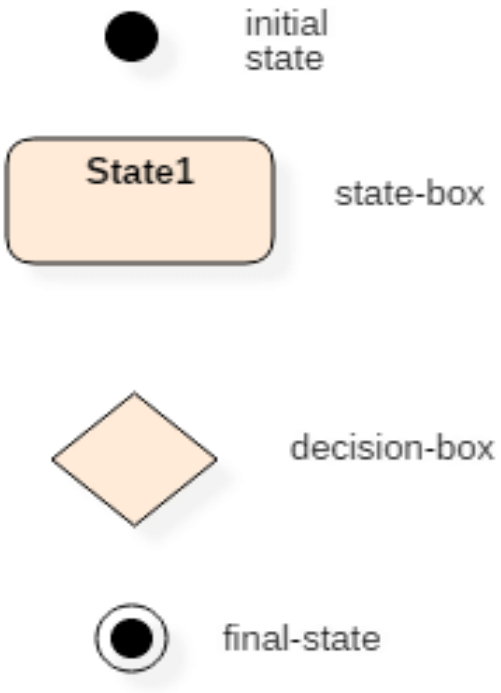


Table 3.6

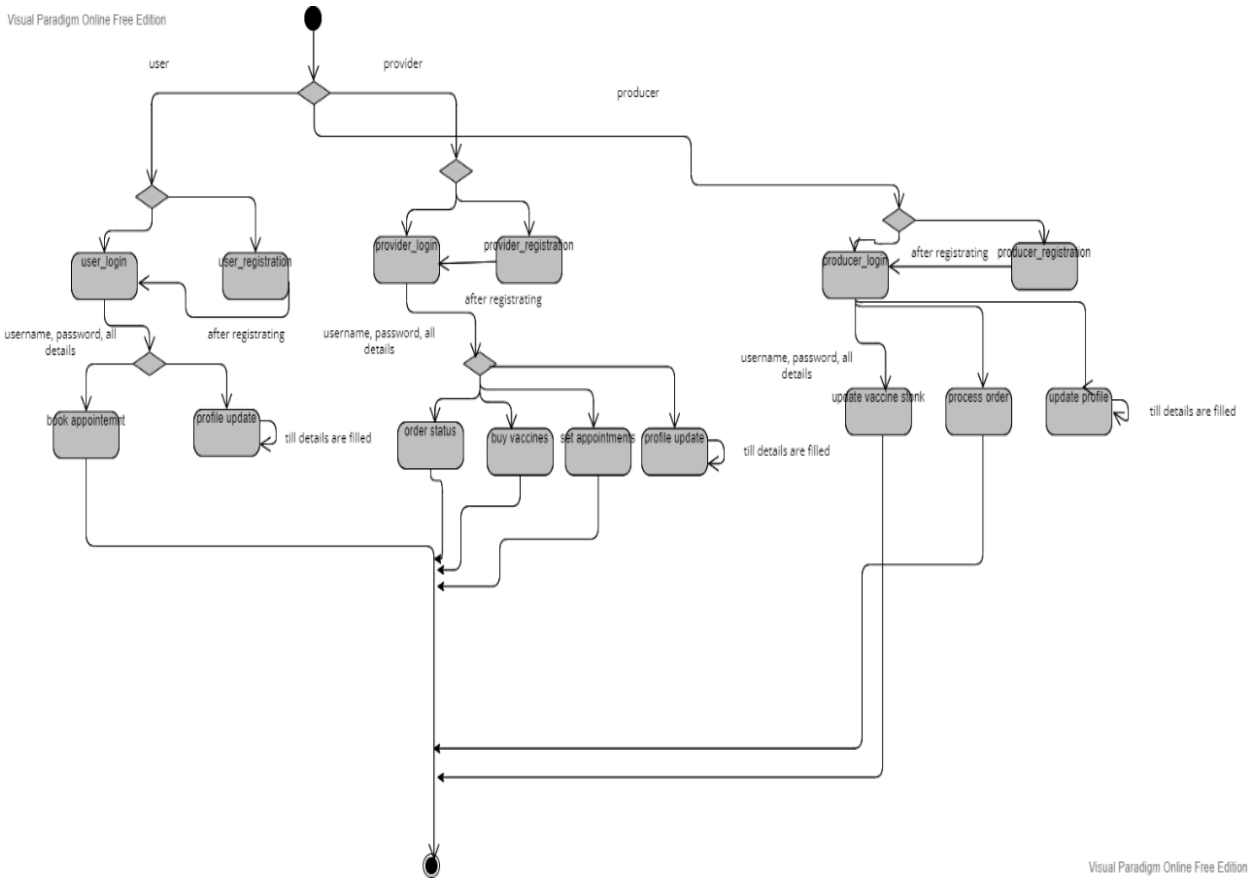


Figure 3.26 State Diagram

3.5.6 Activity Diagram

Activity diagrams are mainly used as a flowchart that consists of activities performed by the system. Activity diagrams are not exactly flowcharts as they have some additional capabilities. These additional capabilities include branching, parallel flow, swim lane, etc.

Before drawing an activity diagram, we must have a clear understanding of the elements used in the activity diagram. The main element of an activity diagram is the activity itself. An activity is a function performed by the system. After identifying the activities, we need to understand how they are associated with constraints and conditions.

References:

- Software Engineering 9th Edition Author : Ian Sommerville

- https://www.lucidchart.com/pages/uml-activity-diagram

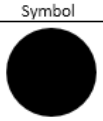
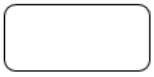
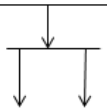
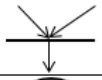
Sr. No	Name	Symbol
1.	Start Node	
2.	Action State	
3.	Control Flow	
4.	Decision Node	
5.	Fork	
6.	Join	
7.	End State	

Table 3.7

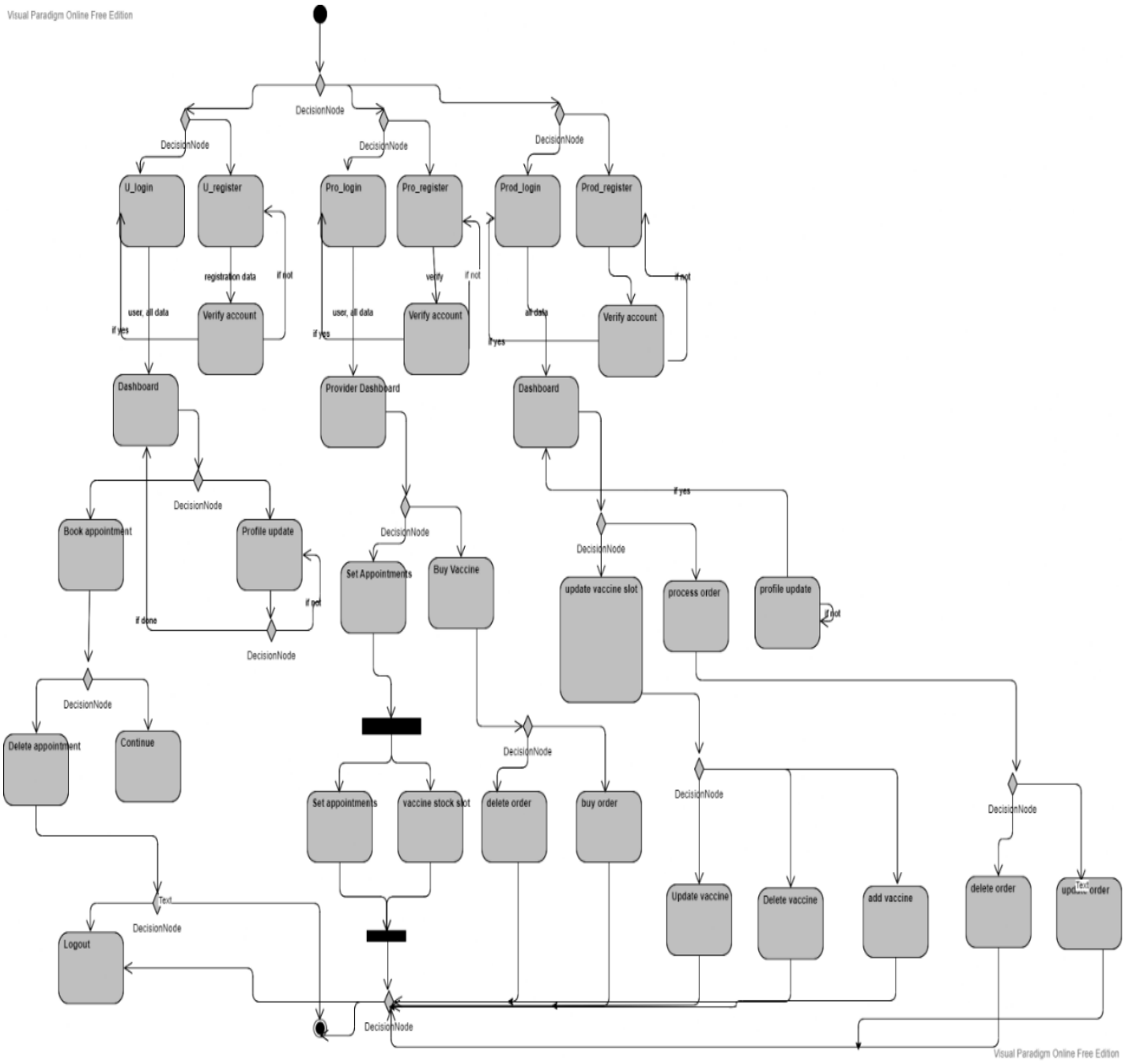


Figure 3.27 Activity Diagram

4. System Design

4.1 Interface Design

User Interface (UI) Design focuses on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions. UI brings together concepts from interaction design, visual design, and information architecture.

Interface elements include but are not limited to:

- Input Controls: buttons, text fields, checkboxes, radio buttons, dropdown lists, list boxes, toggles, date field
- Navigational Components: breadcrumb, slider, search field, pagination, slider, tags, icons
- Informational Components: tooltips, icons, progress bar, notifications, message boxes, modal windows
- Containers: accordion

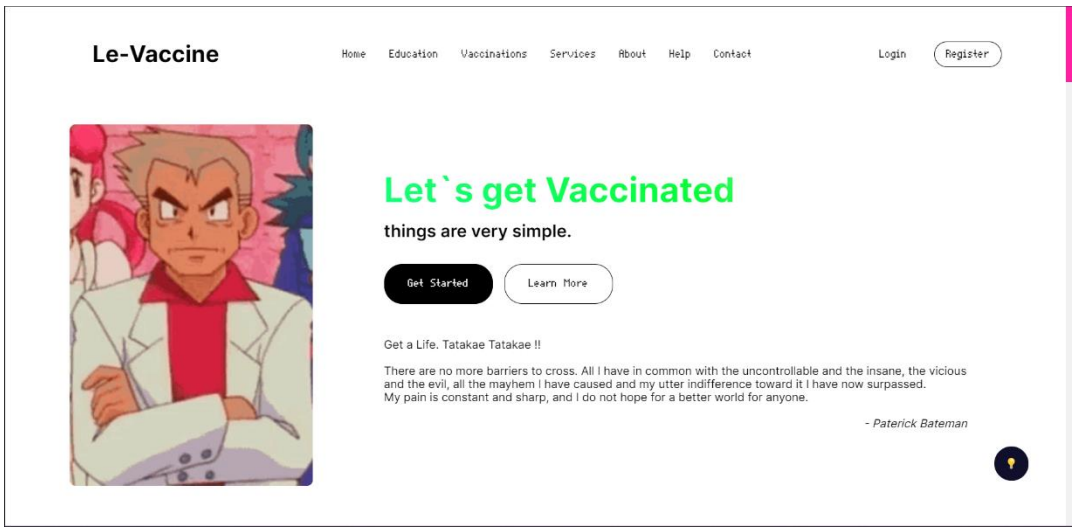


Figure 4.1

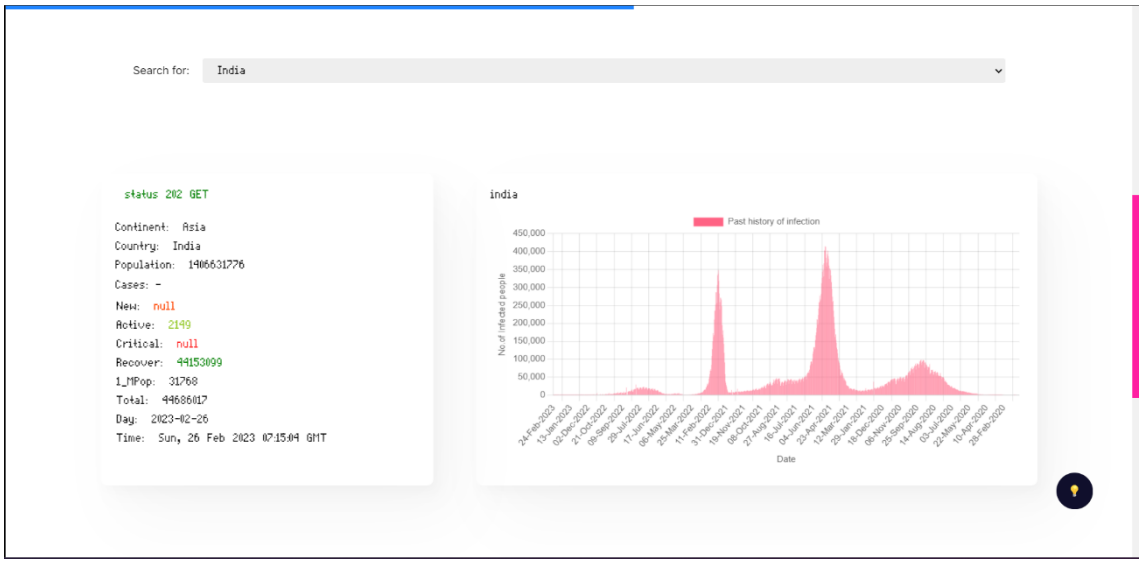


Figure 4.2

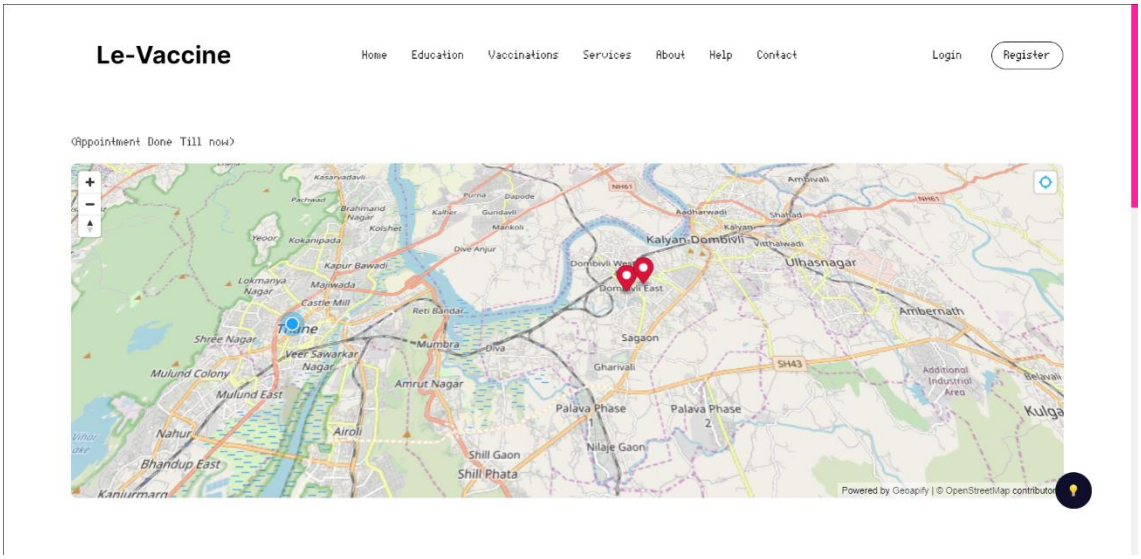


Figure 4.3

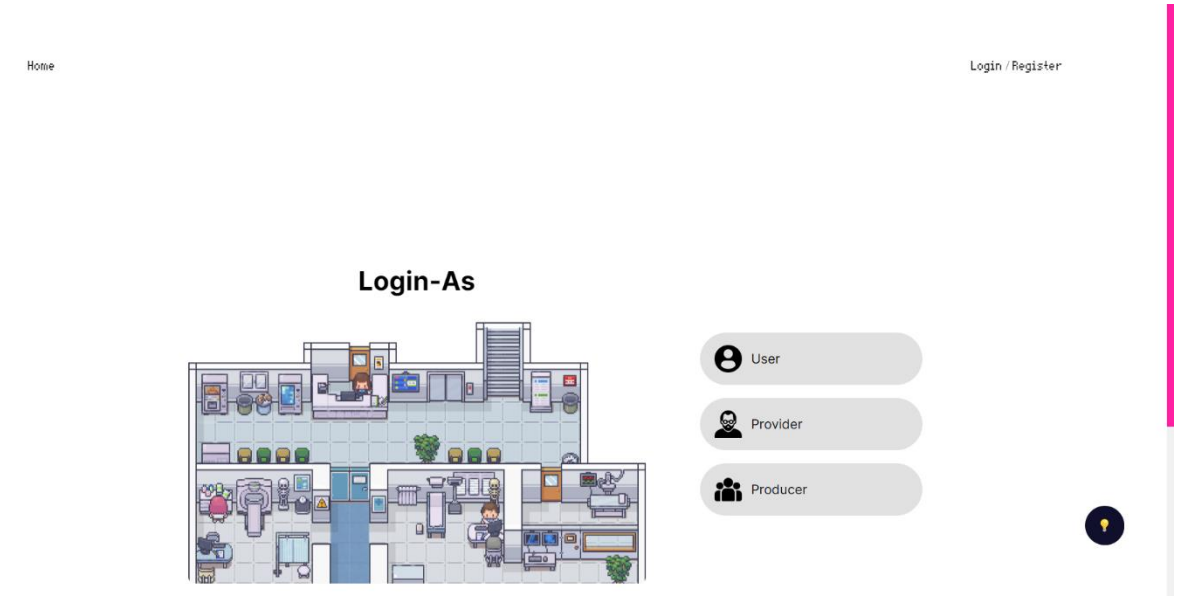


Figure 4.4

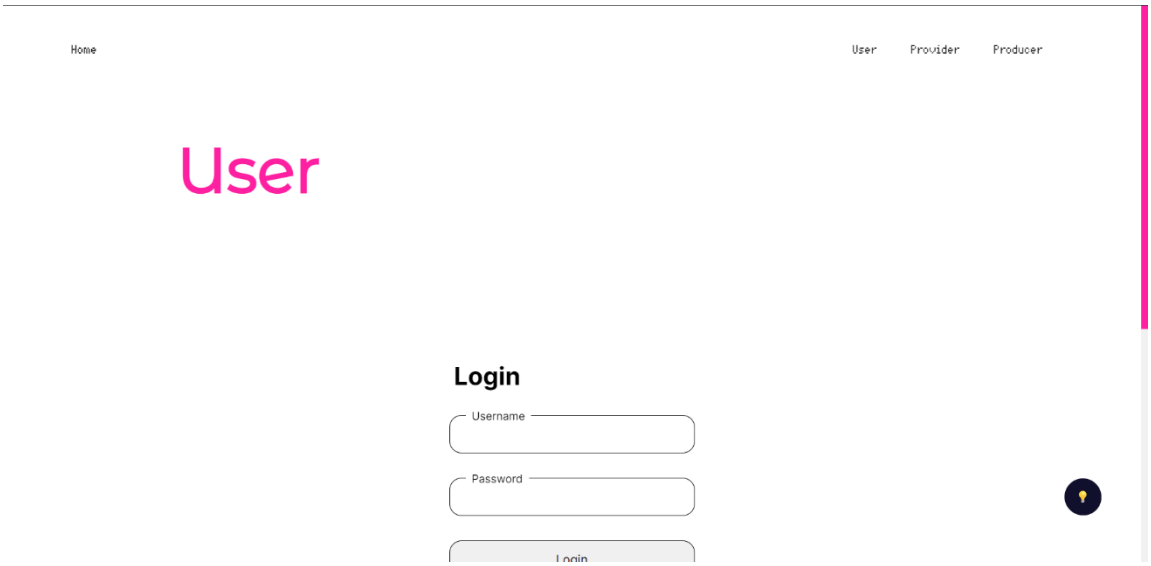


Figure 4.5

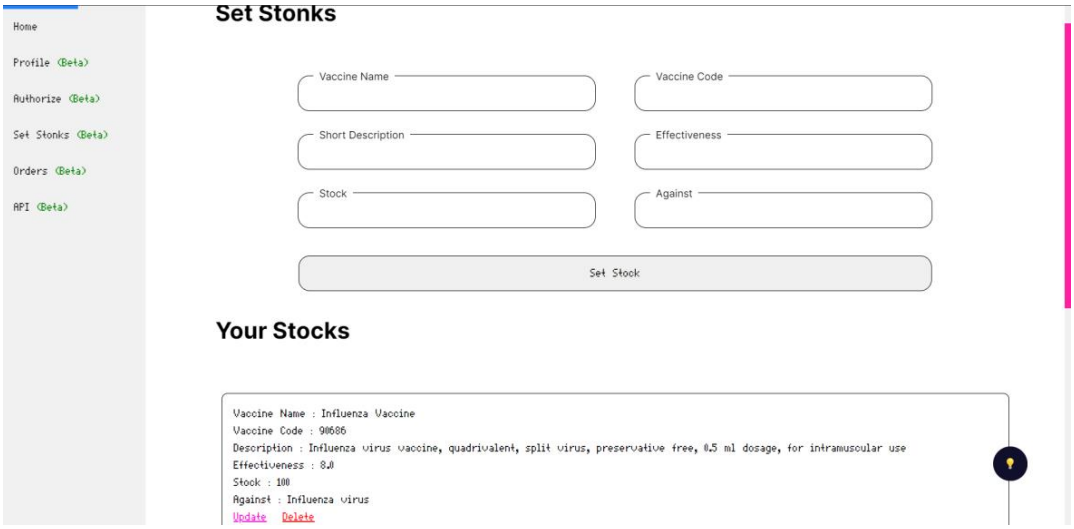


Figure 4.6

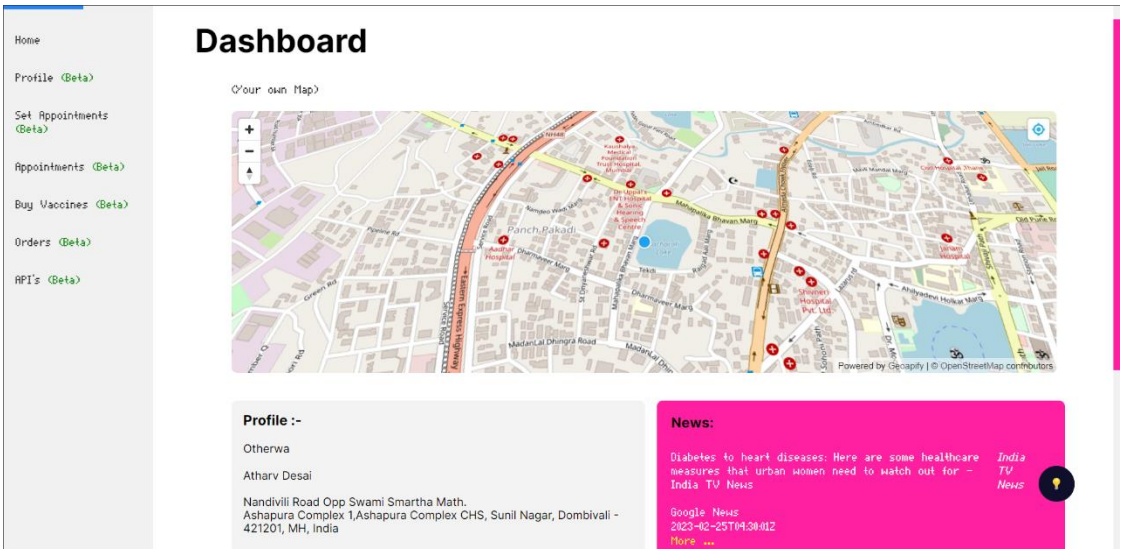


Figure 4.7

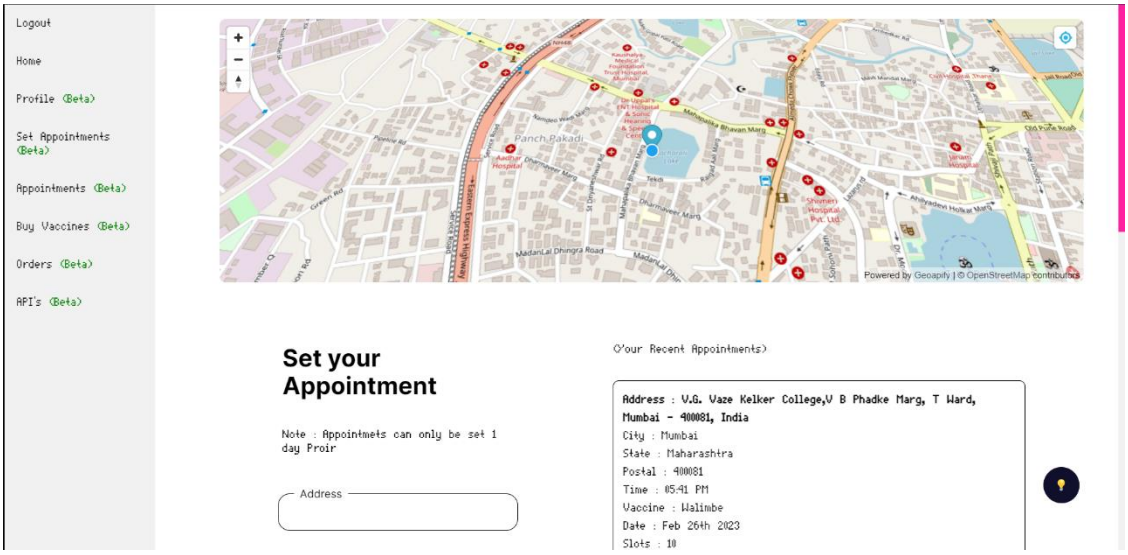


Figure 4.8

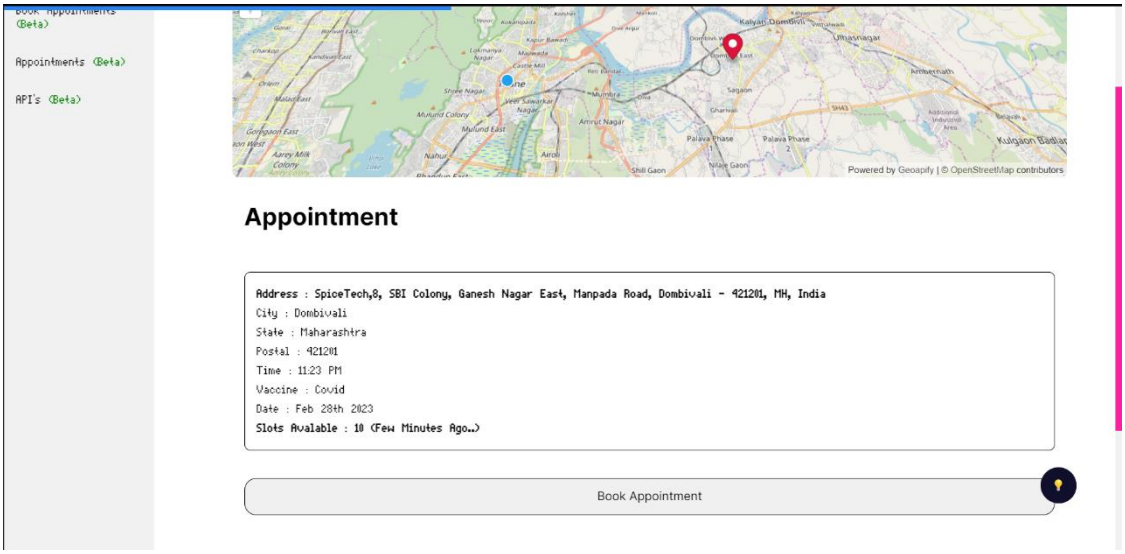


Figure 4.9

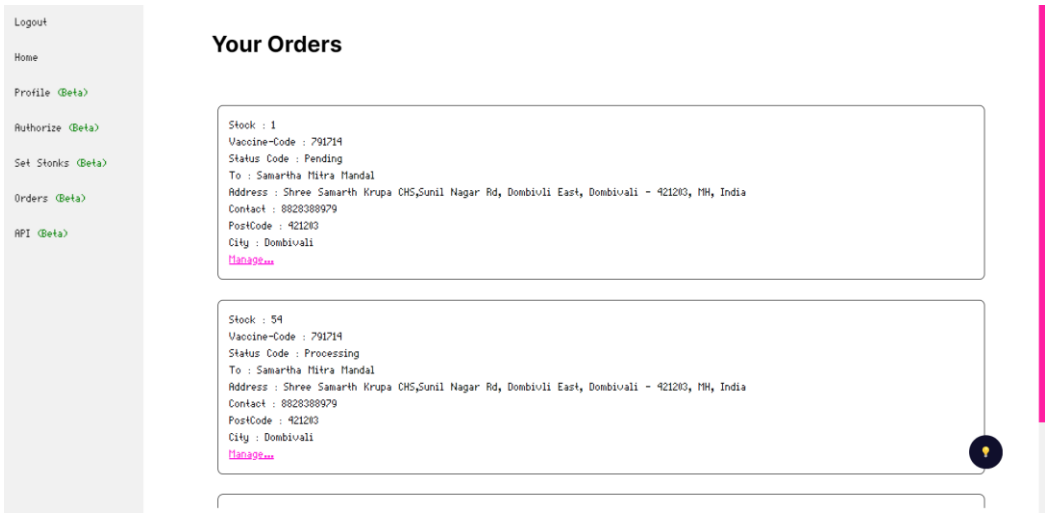


Figure 4.10

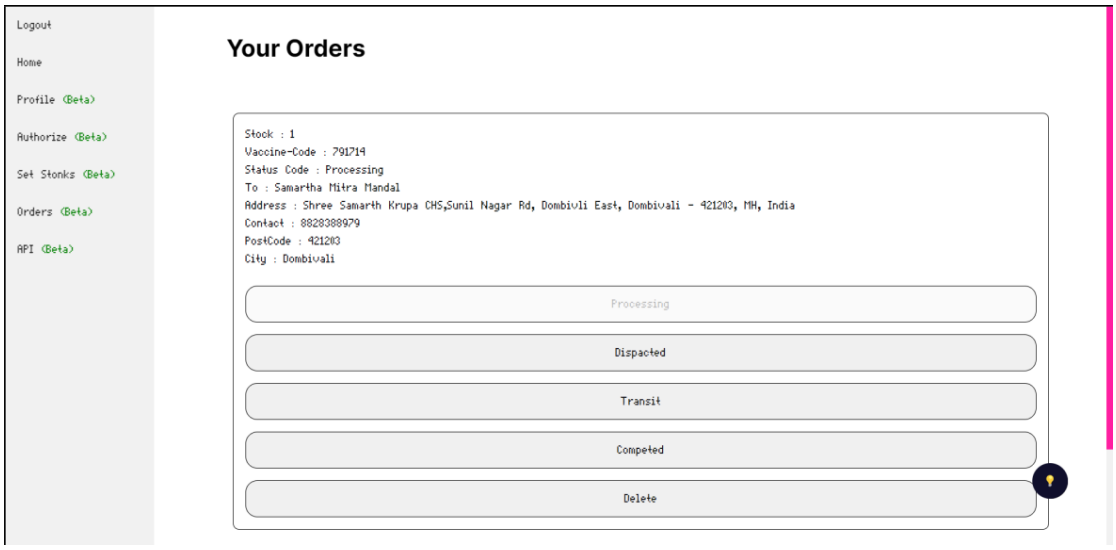


Figure 4.11

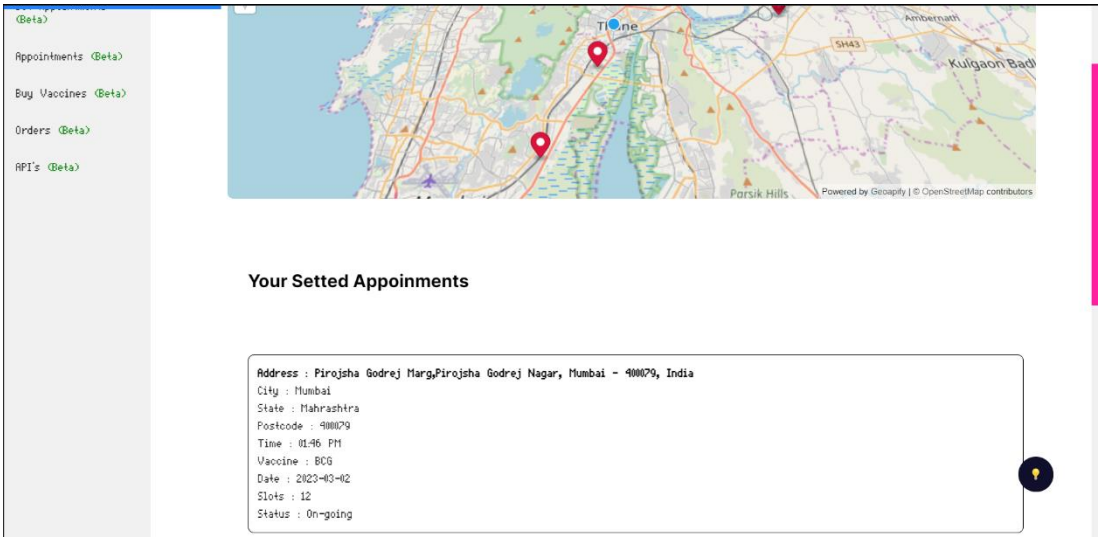


Figure 4.12

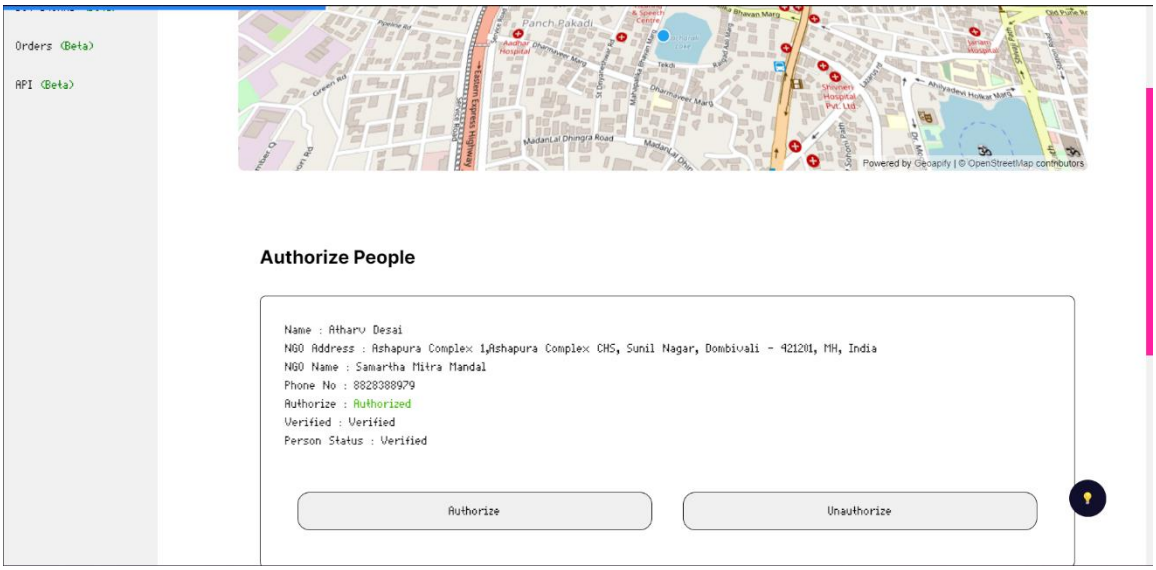


Figure 4.13

5.1 Implementation Approaches

The Vaccine Management System (Le-Vaccine) project was created utilising Extreme Programming Concepts (XP), which is intended to increase software quality and responsiveness to client needs. The extreme programming approach suggests scaling up the best methods that have previously performed successfully in programme development initiatives to extreme levels. Extreme Programming (XP) is a software development process that emphasises high-quality product delivery through frequent and continuous feedback, collaboration, and adaptability. With an emphasis on rapid, iterative development and deployment, XP promotes a close working relationship between the development team, the client, and stakeholders. If the user requirements change at any time, the appropriate component can be rebuilt, reimplemented, and tested again.

The requirements were analysed, and the project was promptly put into action by developing the necessary user interfaces. The interfaces were designed and built utilising Microsoft Studio Code and a database provided by MongoAtlas Service.The frontend of the project was coded in EJS with CSS for style and Js for adding functionalities, and the backend was coded in REST API routes using Express Library.The project was broken down into modules.These modules were built one at a time, and once each one was finished, unit testing was performed on it. The module was included into the main project as soon as it met its requirements. Following integration, each whole module capability was tested, which can also be referred to as integration testing. After adding all the modules to the main project, the final testing was performed to check whether the system was performing properly or not.

The validations were used wherever required. The system was made thinking about all the problems that it could face and thereby proper measures were taken to make sure that the system does not get affected by them. The final product would be delivered when it satisfies all the requirements and functionalities

5.2 Coding Details and Coding Efficiency

5.2.1 Code Design:

View All Appointments Route.

```
Router.get('/user/dash/bookappo', auth, livedata, async (req, res) => {
await connect()
const cookie = req.cookies.jwt
console.log(moment(new Date()).format('YYYY-MM-DD'))
console.log(moment(new Date()).format('hh:mm A'))

// nearby pincode match
let pincode = Number(req.user.detail.postcode)

// match nearby pincodes
pins = [pincode - 2, pincode - 1, pincode, pincode + 1, pincode + 2]
```

```
  console.log(pins)
  console.log(
    moment(new Date()).format()
  )
  appo.find({ 'status': false, 'postcode': { $in: pins }, 'details.date': { $gt: moment(new Date()).format('YYYY-MM-DD'), } }, (err, result) => {
    if (err) console.log(err)
    console.log(result)

    const pos = result.map(position);

    result.map(time);

    function time(item) {
      // change 24.00 to XX.XX AM/PM moment library
      item.details.time = moment(item.details.time).format('hh:mm A');
      item.details.date = moment(item.details.date).format("MMM Do YYYY");
      // console.log(item.details.time)
    }
    // console.log(pos)
    function position(item) {
      return (item.details.position);
    }

    res.render('account/user/bookappo', {
      data: req.user,
      token: cookie,
      appos: result,
      appos_: pos,
      csrf_token: req.csrfToken()
    });
  })
})
```

Book Appointements Route

```
Router.post('/dash/bookappo/:id', auth, livedata, async (req, res) => {
    console.log("heer")
    const id = req.params.id
    console.log(id.toString())
    console.log(req.user._id.toString())
    user.bookappo(req, res, id, req.user._id)
  })
```

Book Appoinment Method:

```
userSchema.prototype.bookappo = async (req, res, appoid, userid) => {
  // sleep
  await connect()


  async function awaitUpdate() {
    try {
      appos.findById(appoid).then((doc) => {
```

```
      // awaiting resposne
      if (doc.details.slots > 0 && Boolean(doc.status) == false) {
        appos.findByIdAndUpdate(appoid, { $inc: { 'details.slots': '-1' } }, (err, results) => {
          if (err) console.log(err)

          console.log(results)

          const appo = new appolist({
            appoid: appoid,
            userid: userid,
            date: new Date()
          })

          appo.save((err, result) => {
            if (err) console.error(err)

            console.log(result)
            // req.flash('msg', "Appointment Booked")
            res.json({ status: '200' })
            user_bookappo(req.user.email, req.user.username, results)
            return results
          })
        })
      } else {
        // req.flash('err', "Appointment Was Not Booked")
        res.json({ status: '404' })
      }
    })

  }
  catch (err) {
    handleError(err);
  }
}

let appoupdate = await awaitUpdate()
console.log(appoupdate + " asdas")

}
```

5.2.2 Code Efficiency

The code was kept as small as possible while maintaining functionality and dependability. For software, code efficiency is closely related to algorithmic efficiency and the speed of runtime execution. It is critical to achieving peak performance. Wherever there was code repetition, it was placed in functions, async callbacks and procedures, as well as the High Read Write Provided by MongoDb. Instead of rewriting the entire code, the functions were invoked.

5.3 Testing Approach

One effective testing approach for project would be to conduct a thorough review of the document to ensure accuracy and completeness. This review could include checking that all necessary information is present, such as the name and address of the site, the types of vaccines available, and the hours of operation. Additionally, testing could involve simulated scenarios, such as a patient arriving for their appointment or a staff member needing to access important information quickly. This can help identify any potential issues with the documentation and ensure that it is user-friendly and easy to navigate. Regular updates and revisions to the documentation based on feedback from staff and patients can also help improve its effectiveness over time

5.3.1 Unit Testing

A unit testing approach can be effective. This involves testing individual components or units of the documentation, such as the instructions for administering a vaccine or the procedures for recording vaccine doses. Each unit should be tested in isolation, with any dependencies or interactions with other units mocked or simulated.

Tests should be designed to ensure that each unit performs as expected and produces the correct outputs for various inputs or scenarios. This can include testing for accuracy, completeness, and readability of the documentation. Unit tests can be automated using testing frameworks, making it easier to run tests quickly and consistently.

Overall, a unit testing approach can help ensure that the documentation for a vaccination site is reliable, accurate, and easy to use for healthcare professionals administering vaccines.

ID	Test Case Description	Test Case Data	Expected Result	Actual Result	Remark
User Test Cases					
0	Login for user	Username: Otherwa Password: admin	Should get redirected to the dashboard	Gets redirected	Pass
2	Register for User	Username: Otherwa Password: admin Name: Atharv Ankush Desai, Age 20,gender male, Adhar no 12121243445. pincode 42120,phone-no 8828388979,email atharvdesai2002@gmail.com	Error due to Adhar no	As Expected	Pass

3	Book Appointments	Appointment Name XXY Appointment Time 13:00 – 14:00 Date: 21/09/222 (On Book Button Click)	The appointment Should be booked	As Expected	Fail
4	Update Profile for user	Username: Otherwa Password: admin Name: Atharv Ankush Desai, Age 20,gender male, Adhar no 2023023023. pincode 42120,phone-no 8828388979,email atharvdesai2002@gmail.com	User Profile Updated	As expected	Pass
5	Register for User	Username: Otherwa Password: admin Name: Atharv Ankush Desai, Age 20,gender male, Adhar no 1212232pincode 421201,phone-no 882838899,email atharvdesai2002@gmail.com	Error due to Phone number	As Expected	Pass
Provider Test Cases :					
6	Set Appointments	Appointment Name : XYZ Appointment Location : Nalasopara/202,22e Appointment Timming : 13:00 – 14:00 Appointment Date : 24/09/22	Appointemnt should be setted up	As Expected	Pass
7	Buy Vaccine	On certain vaccine buy confirm order button click and checkout	Order should be placed	Error in Buying Required Amount	Fail

8	Profile Update for provider	Username: Otherwa Password: admin Name: Atharv Ankush Desai, Age 20,gender male, Adhar no 1212232pincode 421201,phone-no 882838899,email atharvdesai2002@gmail.com NGO : Atmaram NGO	Profile Updated	As Expected	Pass
9	Profile Update for provider	Username: Otherwa Password: admin Name: Atharv Ankush Desai, Age 20,gender male, Adhar no 1212232pincode 421201,phone-no 882838899,email atharvdesai2002@gmail.com NGO :	Profile Not Updated NGO Field Blank	As Expected	Pass
10	Update profile for provider	Username: Otherwa Password: admin Name: Atharv Ankush Desai, Age 20,gender male, Adhar no 1212232pincode 421201,phone-no 8828388979,email atharvdesai2002gmail	Error due to invalid email	As Expected	Pass
11	Set Appointments	Appointment Name : XYZ Appointment Location : Nalasopara /202,22e Appointment Timming : 13:00 – 14:00 Appointment Date :	Error due to no date specified	As Expected	Pass
12	Set Appointments	Appointment Name : XYZ Appointment Location : Nalasopara /202,22e	Error due to no end time specified	As Expected	Pass

		Appointment Timming : 13:00 – Appointment Date : 20.09.22			
Provider Test Cases					
13	Process Orders	On button Click of delete order status order should be deleted	Order Should get deleted	As Expected	Pass
14	Process Orders	On button Click of Update order status order should be updated	Order Should get Updated	As Expected	Pass
15	Update Vaccine Stonks	Vaccine Name : BCG Vaccine Code : 2033 Description: wfwddssdsd Quantity:203	Vaccine stocks not Updated due to faulty description	As Expected	Pass
16	Profile Update For Producer	Username: Otherwa Password: admin Name: Atharv Ankush Desai, Age 20,gender male, Adhar no 1212232pincode 421201,phone-no 8828388979,email atharvdesai2002@gmail.com Company Name : Le-Vaccine Company Address : C/202 Shree Samartha Krupa	Should not get updated Successfully Not valid Company Name	As Expected	Pass
17	Profile Update For Producer	Username: Otherwa Password: admin Name: Atharv Ankush Desai, Age 20,gender male, Adhar no 1212232pincode 421201,phone-no 8828388979,email	Should get updated Successfully	As Expected	Pass

		atharvdesai2002@gmail.com			
18	Process Orders	On the button Click of Update order status order should be updated	Order Should get Updated	As Expected	Pass
19	View Order Status	On button click of view status	Order status should be shown	As Expected	Pass
20	Update Vaccine Stonks	Vaccine Name: BCG Vaccine Code: 2033 Description: For Immunity etc, Quantity:203	Vaccine stocks Updated	As Expected	Pass
21	Update Vaccine Stonks	Vaccine Name: BCG Vaccine Code: 2033 Description: For Immunity etc, Quantity:0	Vaccine stocks Updated	As Expected	Pass
22	Update Vaccine Stonks	Vaccine Name: BCG Vaccine Code: 2033 Description: For Immunity etc, Quantity:	Vaccine stocks are not Updated	As Expected	Pass
23	Profile Update For Producer	Username: Otherwa1 Password: admin Name: Atharv Ankush Desai, Age 20,gender male, Adhar no 1212232pincode 421201,phone-no 8828388979,email atharvdesai2002@gmail.com Company Name: Bhartia Biotech Company Address: C/202 Shree Samartha Krupa	The username is already taken error	As Expected	Pass

24	Profile Update For Producer	Username: Otherwa Password: admin Name: Atharv Ankush Desai, Age 20,gender male, Adhar no 1212232pincode 421201,phone-no 8828388979,email atharvdesai2002@gmail.com :	Should not get updated Successfully	As Expected	Pass
25	Profile Update For Producer	Username: Otherwa Password: admin Name: Atharv Ankush Desai, Age 20,gender male, Adhar no 1212232pincode 421201,phone-no 8828388979,email atharvdesai2002@gmail.com	Should not get updated Successfully	As Expected	Pass

5.3.2 Integration Testing

One possible testing approach for vaccination site documentation integration testing is to start by identifying the different components of the system that need to be integrated. This can include the user interface, the database, the API endpoints, and any third-party services that are used.

Once the components have been identified, test cases can be designed to validate the integration between them. This can include testing how information is captured and stored in the database, how it is displayed on the user interface, and how it is transmitted through the API.

It is important to also test the system's ability to handle different scenarios and edge cases, such as concurrent users, errors in data input, and unexpected system failures. This can be done through both automated and manual testing methods.

5.3.3 Beta Testing

The first step would be to identify the scope of the testing and the objectives to be achieved. Then, a list of test scenarios can be created based on the identified scope and objectives. These scenarios should cover different aspects of the documentation, such as clarity, completeness, and accuracy.

Next, a group of beta testers can be selected, who can represent the target audience for the documentation. The testers can then execute the test scenarios and provide feedback on their experience and observations. The feedback can be collected in a structured format, such as a survey or a feedback form.

Finally, the feedback can be analyzed, and the necessary changes can be incorporated into the documentation. This iterative process can be repeated until the documentation meets the desired quality standards

5.4 Modifications and Improvements

The Serializability Issue For Booking Appointments Was Fixed By Introducing Random 1000 – 5000 Ms Delay Along with An Async Promise Call-back which allows The Express Logic Layer to Read/Write the Correct Amount of Slots Remaining For the Appointments

Similar Approach is Done For Booking Vaccine Orders And Order Status

More Over a Map is Integrated Using Geoapify and Maplibre API which Provide Geospatial Data For a Set Of Co-ordinates Dragged by the Provider For Setting Appointments ,User Locations etc.

1 Test Reports

The testing phase of project development is critical. The testing step allows you to determine whether all of the capabilities are being executed correctly.

The testing phase began with the creation of test cases for each module as well as the design of the modules themselves. The process for integrating test cases was completed. Each module was then examined, and test cases were created based on the findings. The test cases provided input and the expected result after entering the values. After constructing the test cases, they were validated by actually entering the inputs and determining if the estimated and actual outputs were the same or not. If the estimated output corresponded to the actual output then the test cases were remarked to be passed else, they were remarked as a failure. Not all values were tried and tested but the process made sure the system would be able to cope up with any values. After performing all the testcases, it was concluded that there were no errors. So, no further modifications were needed in the respective modules.

Based on the performed test cases And modifications the problems such as concurrent booking of Appointments, Buying of Vaccines were capable of tackling the problem defined for the project objectives and commercial small scale use.

6.2 User Documentation

- Homepage:

The First Page the user can see Several Other tabs are also present which provide the aim and vision of the project

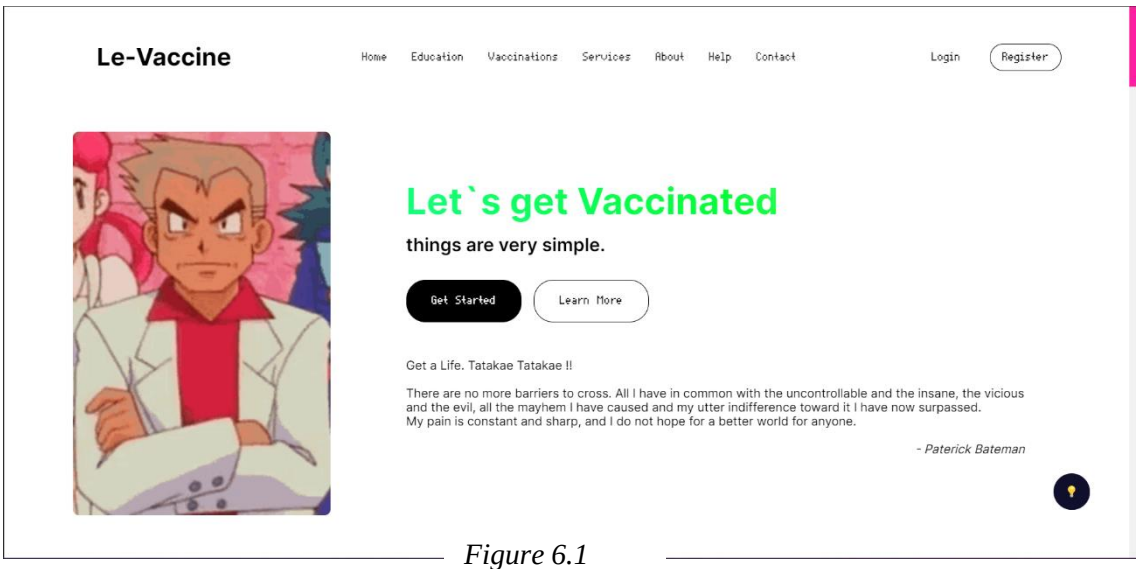


Figure 6.1

- Vaccinations:

This page gives the information of the actual vaccination done by using this site which is mapped and an external open API which provide daily records of Covid-19 cases all over the world

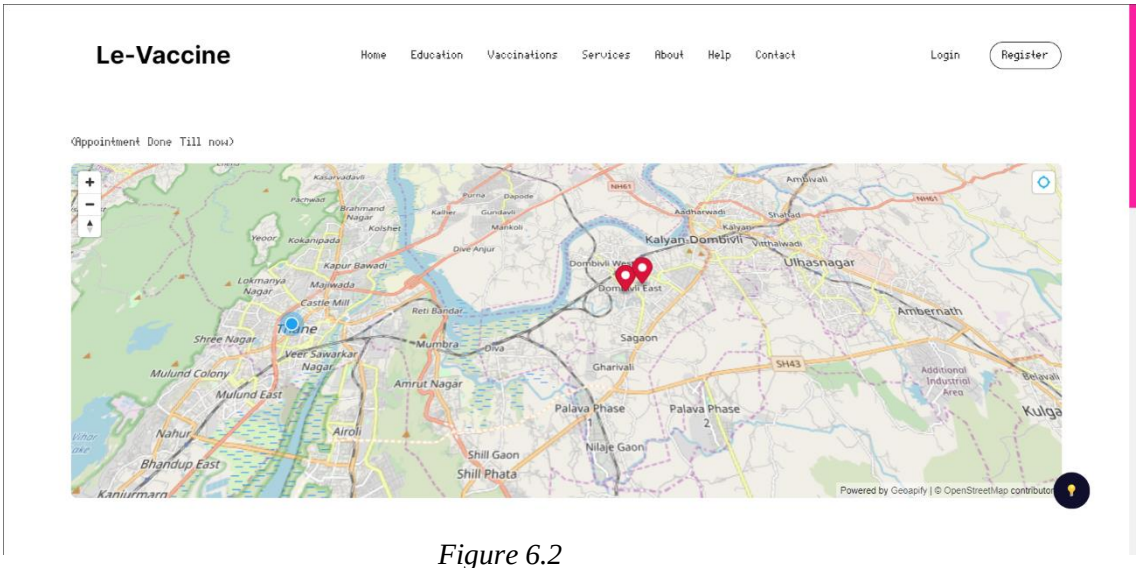


Figure 6.2

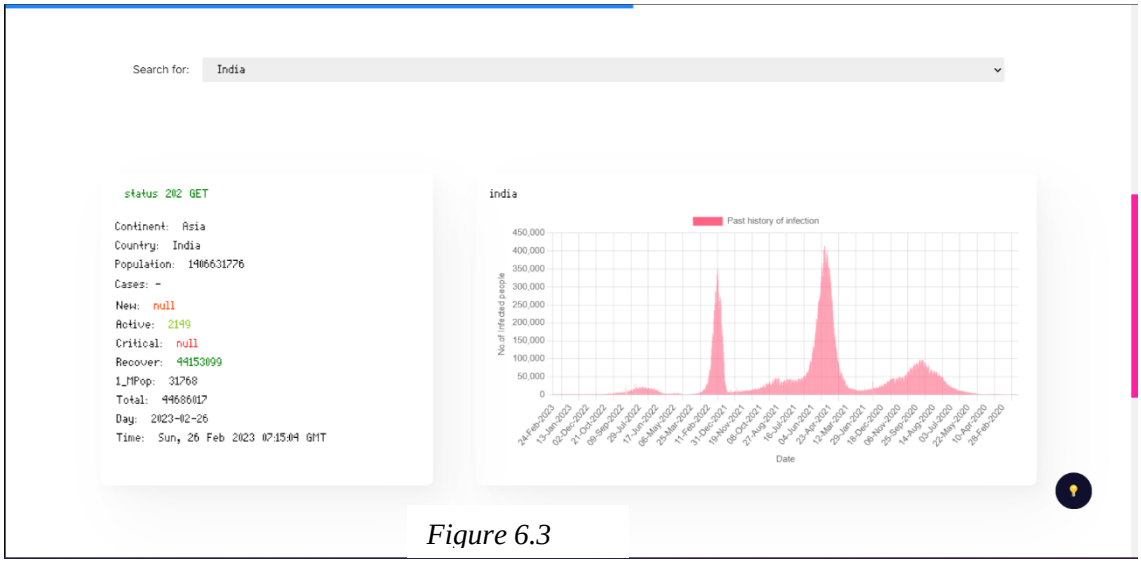


Figure 6.3

- APIs

  This page is only for registered users, provider, producer which can use vaccinations data generate by the site for their own use.

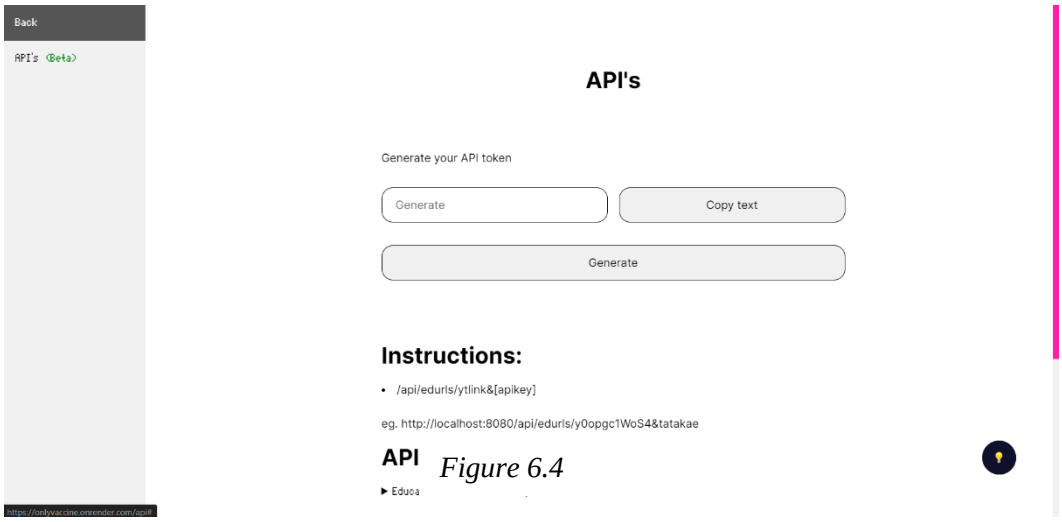


Figure 6.4

- Help:

Open AI Chat-bot is also integrated using API services.

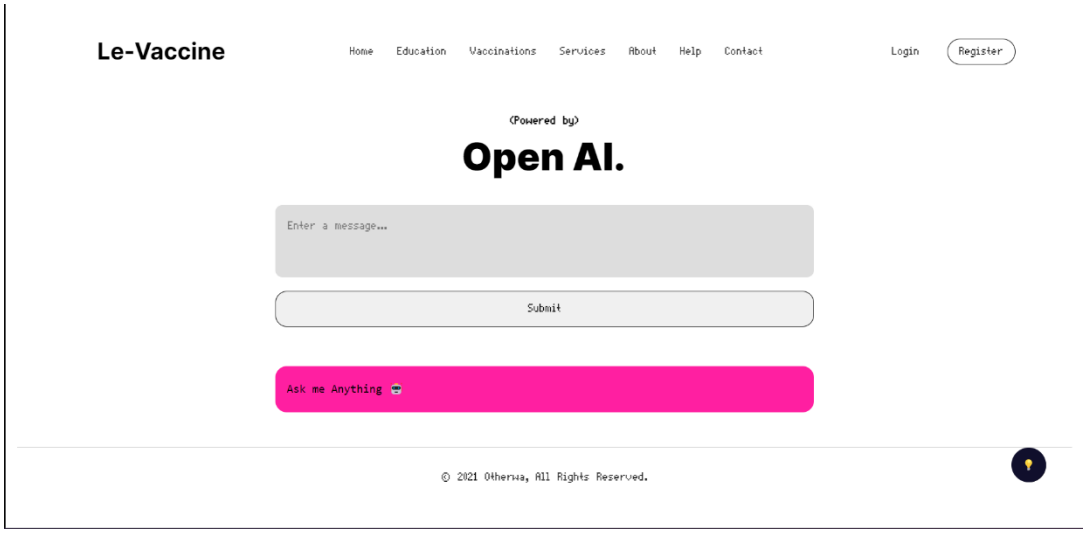


Figure 6.5

User :

- Login:

  The Required User can login through this page this page is a common point for different roles

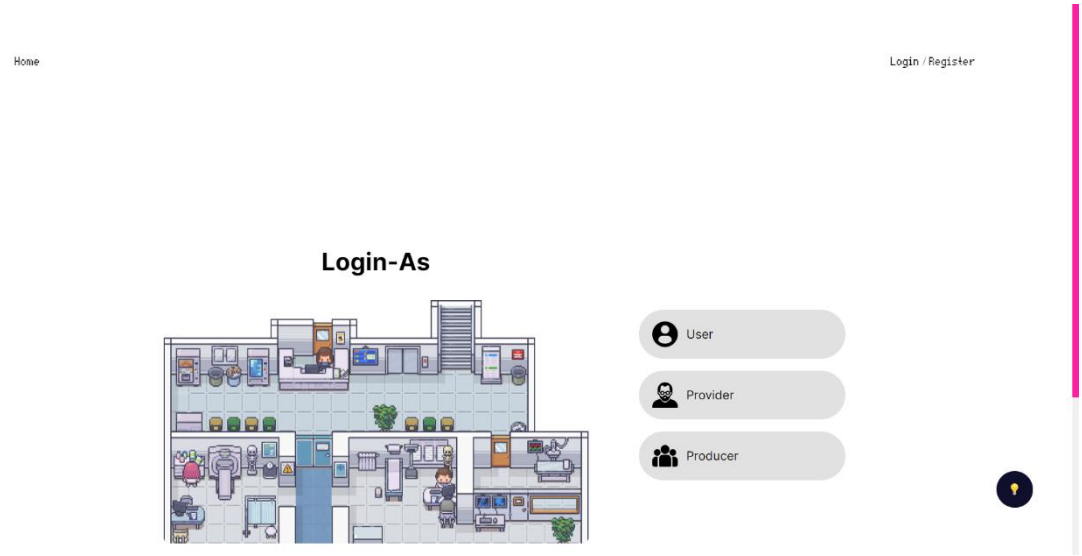


Figure 6.5

- Dashboard:

  This is the first page the user sees after logging in

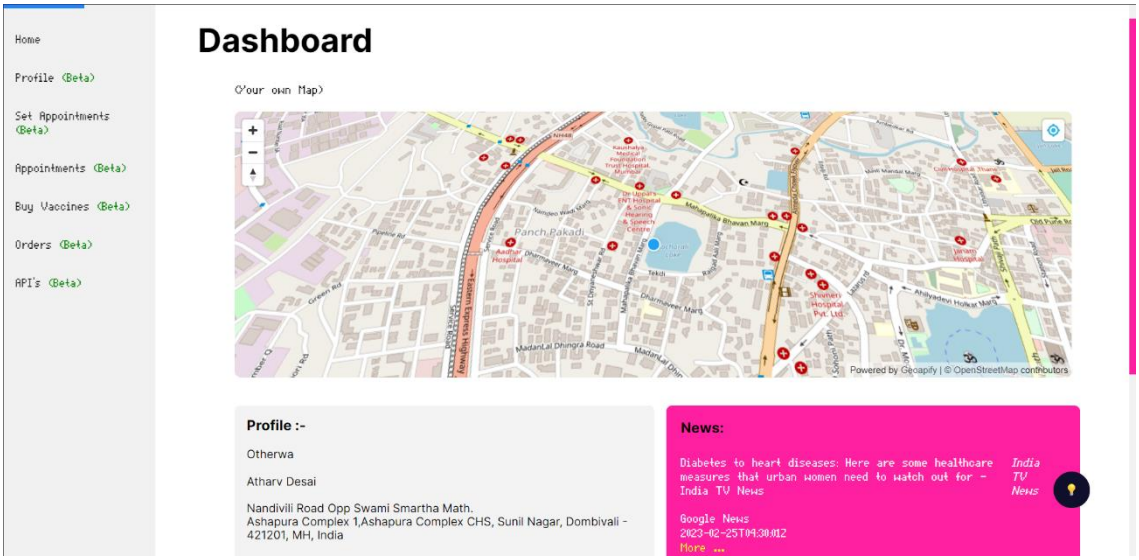


Figure 6.6

- Profile:

  Here the user can update his/her profile as per the need

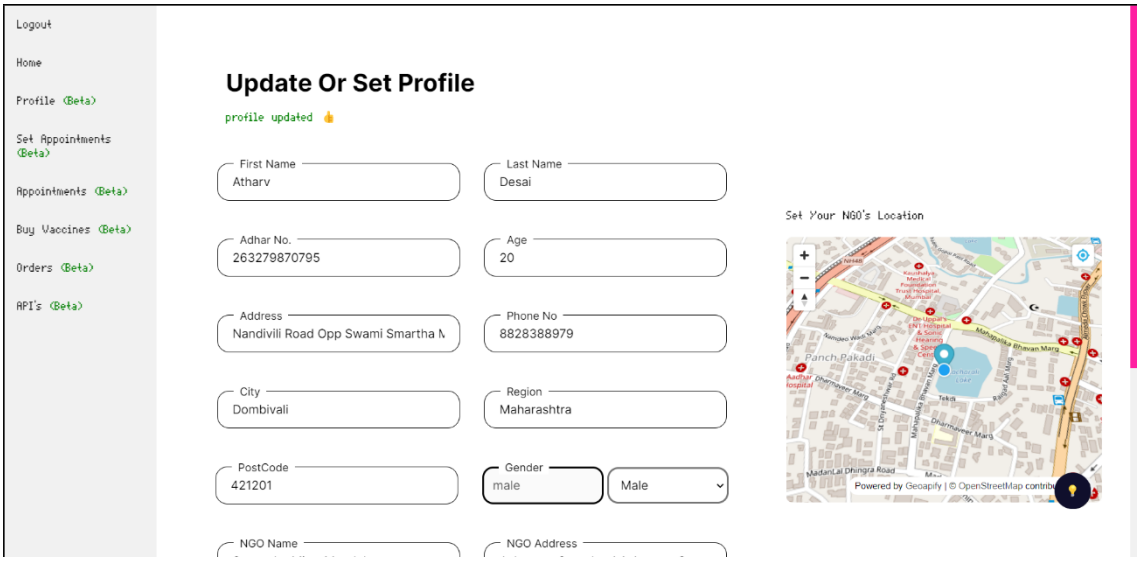


Figure 6.7

- Book Appointments:

  Here the user can book appointment which is nearby matching his /her by using pincode for lookup

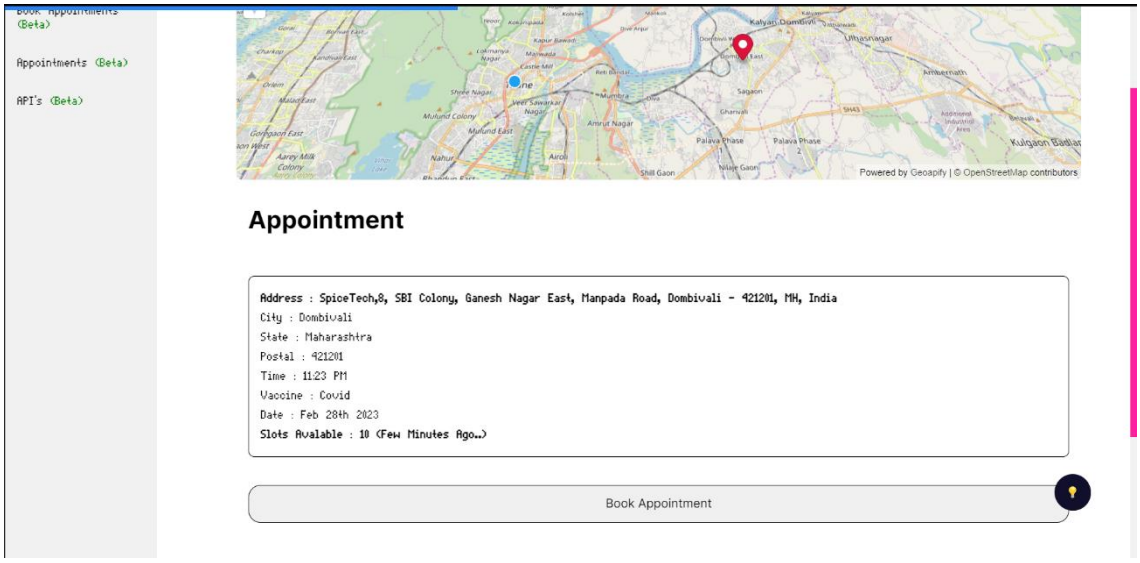


Figure 6.8

Provider :

- Login:

  Appropriate Provider can login

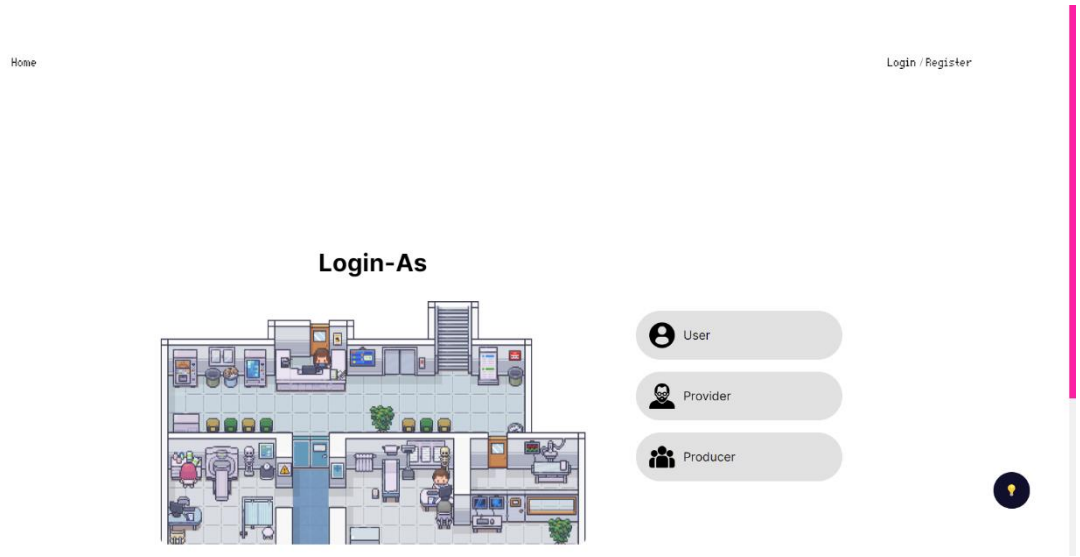


Figure 6.9

- Dashboard:

  The First page the Provider Sees

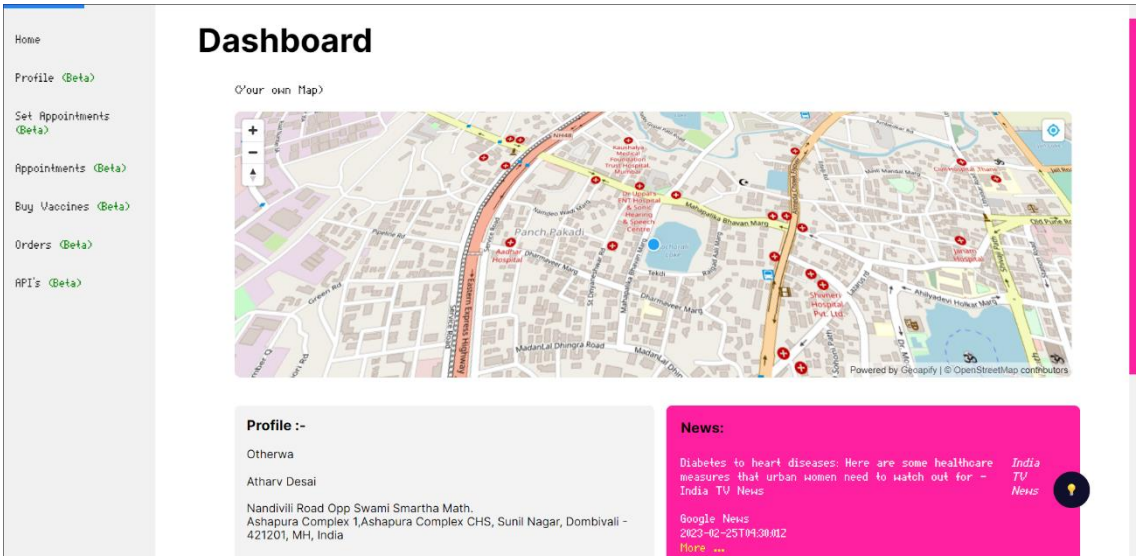


Figure 6.10

- Profile:

  Here the Provider can update his or her NGO profile etc.

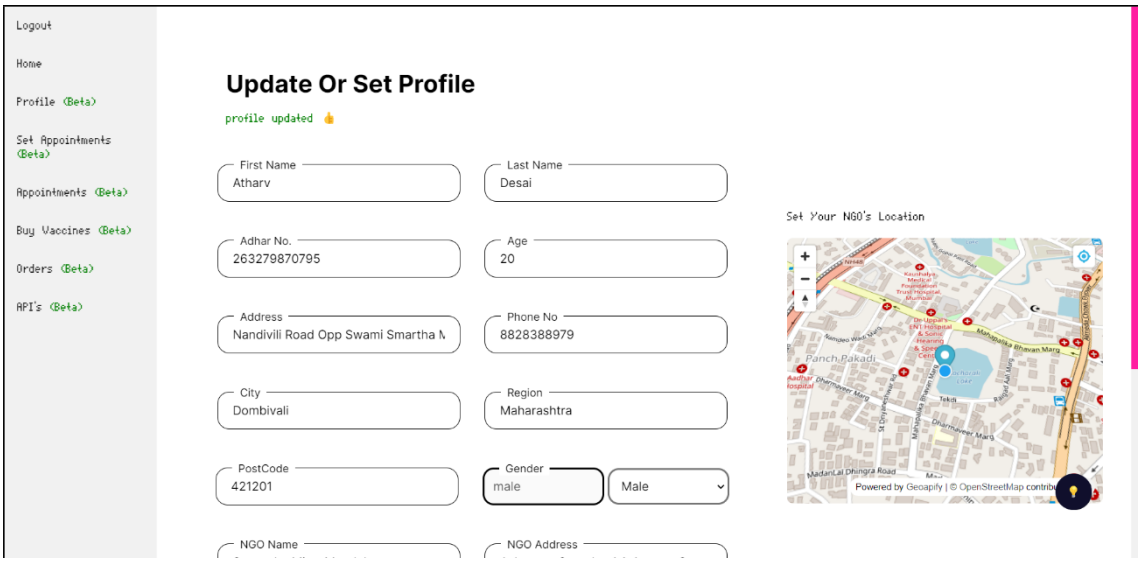


Figure 6.11

- Set Appointments:

  Here The Provider can set Appointments as per the need

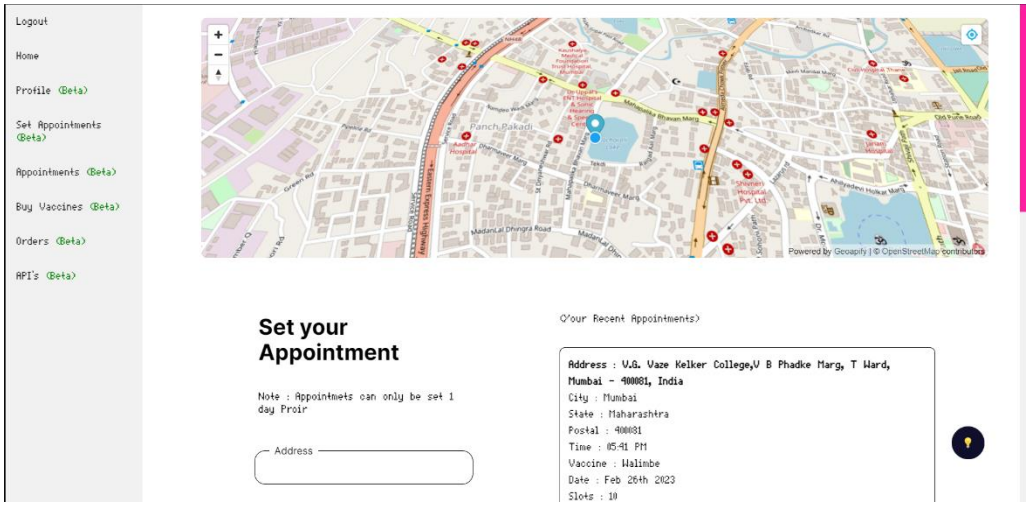


Figure 6.12

- Appointments:

  Here the Provider can see his/her appointments and can either stop or start for slots

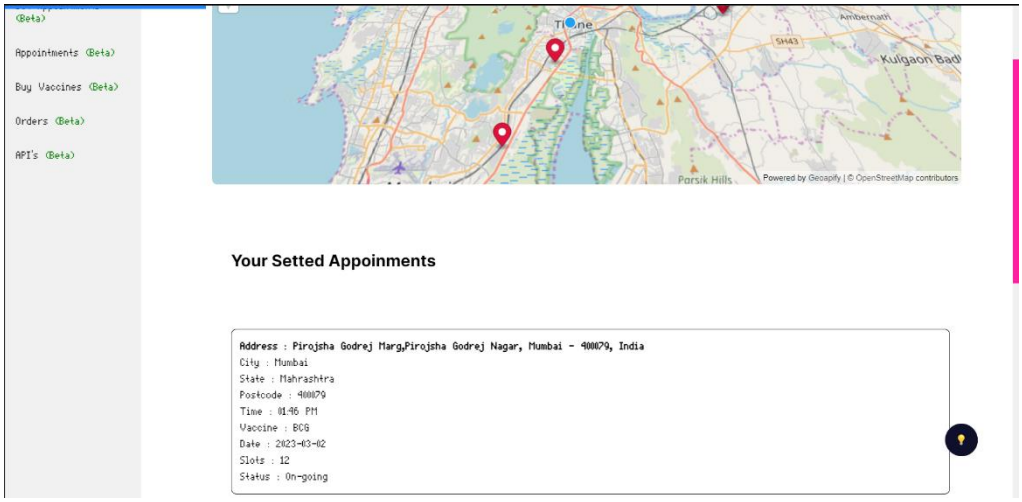


Figure 6.13

- Buy Vaccine

  Here the Provider can buy Vaccine which are required

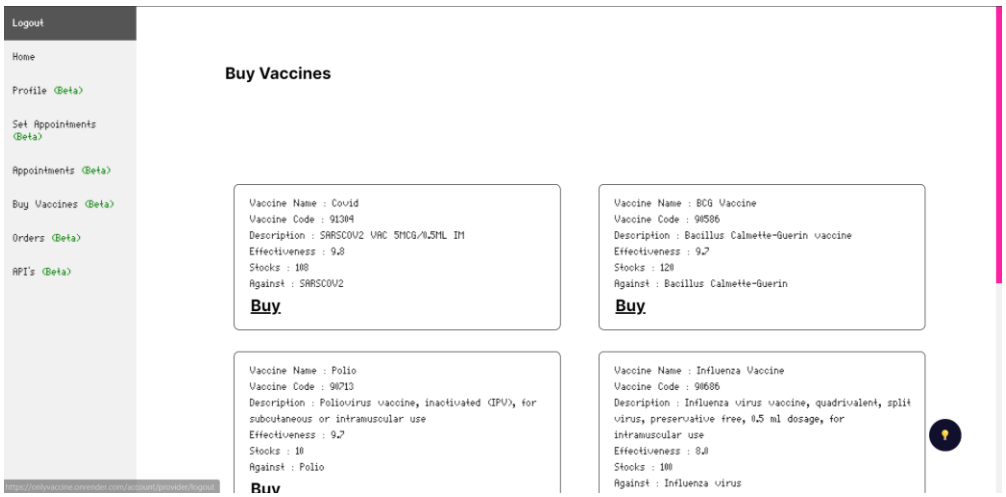


Figure 6.14

- Orders:

  Here The Provider can keep track of the Vaccine Orders Placed.

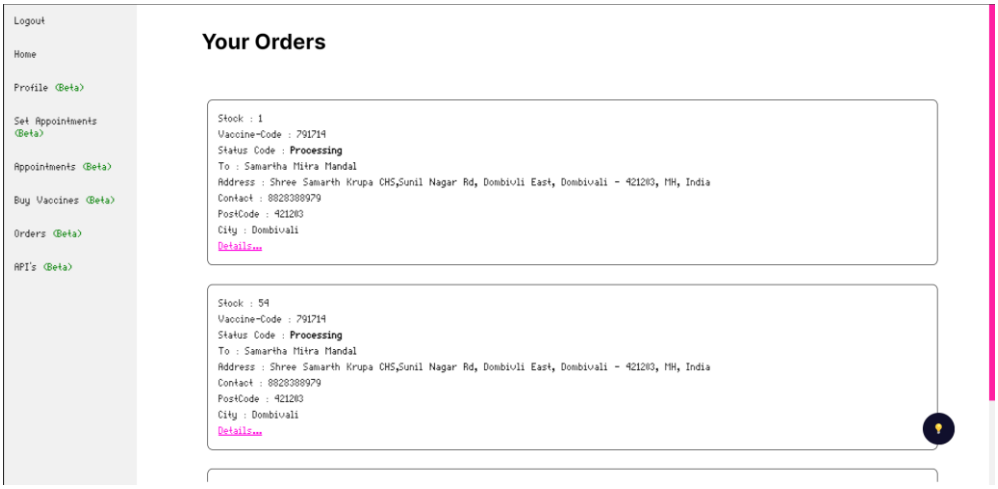


Figure 6.15

Producer:

- Login

  The First Page the Provider Sees For logging in

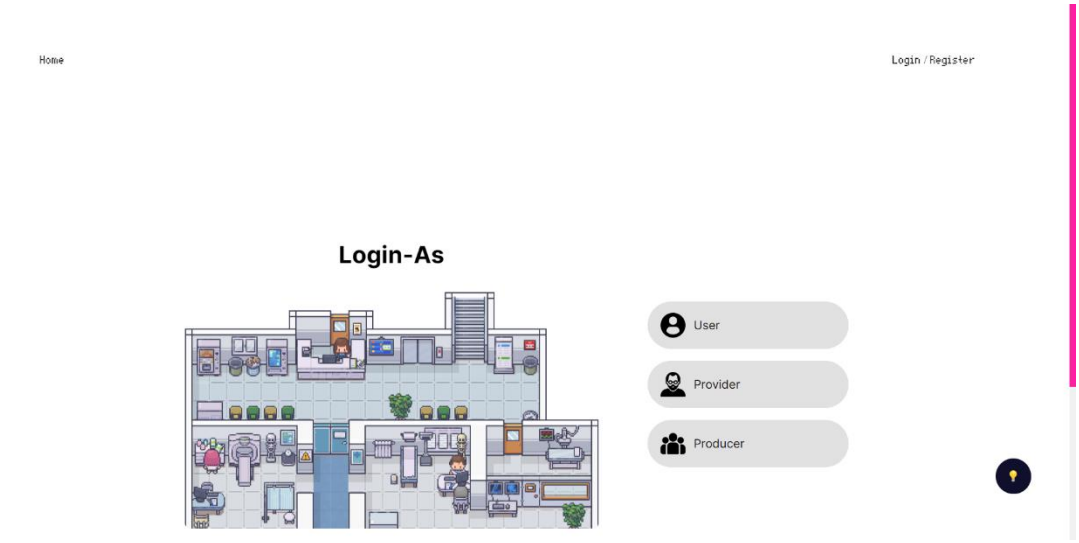


Figure 6.16

- Dashboard

This is the first page the producer sees after logging in

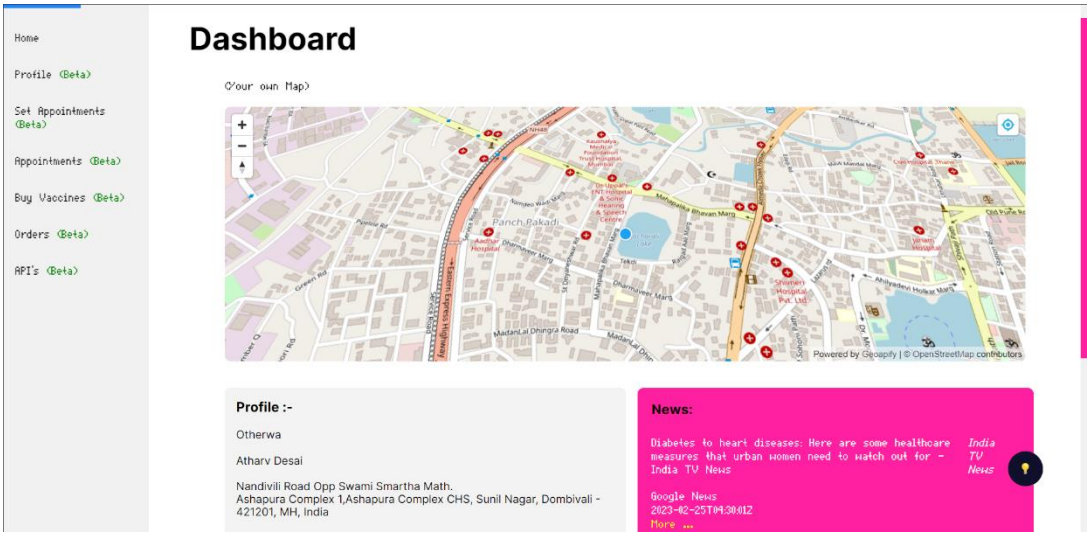


Figure 6.17

- Authorize

Here the Producer can Authorize or Unauthorized the Provider of certain NGO

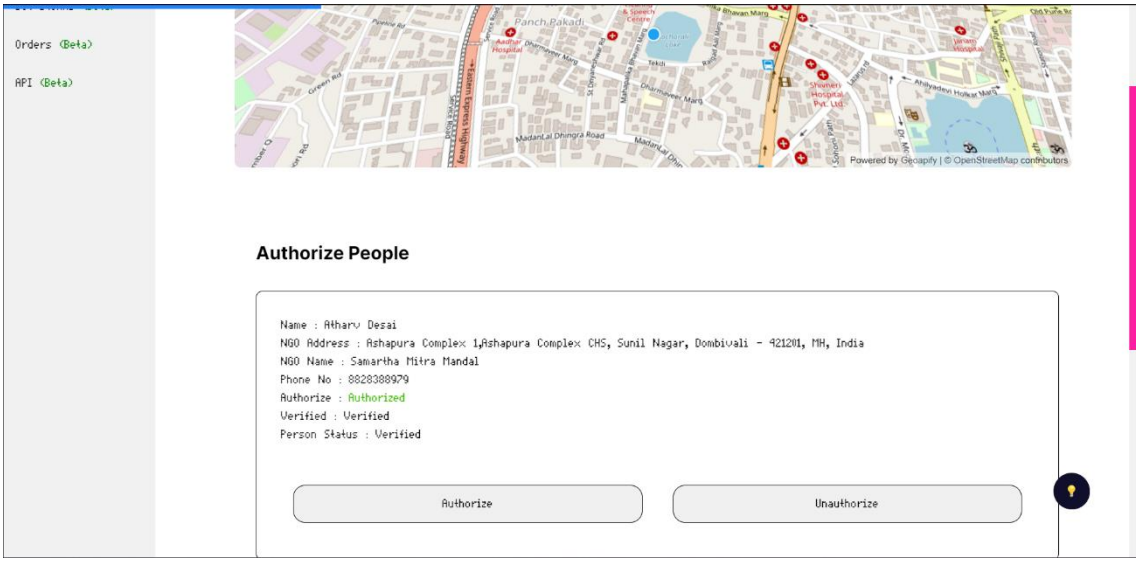


Figure 6.18

- Orders

  Here the Producer can Process the Orders Received From the Providers

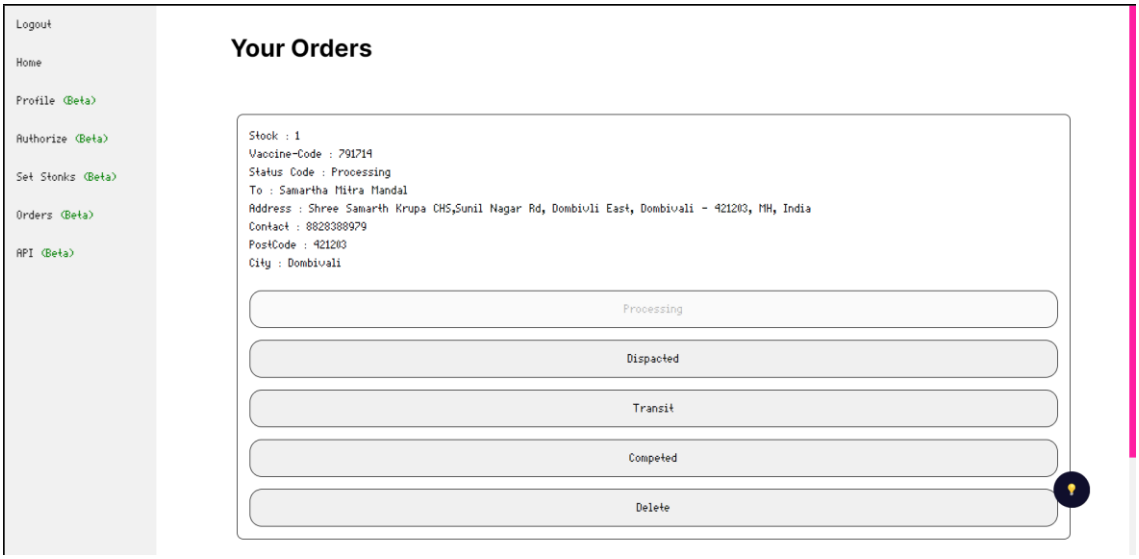


Figure 6.19

- Stocks:

  Here the Producer can set stocks for the vaccine they can manufacture

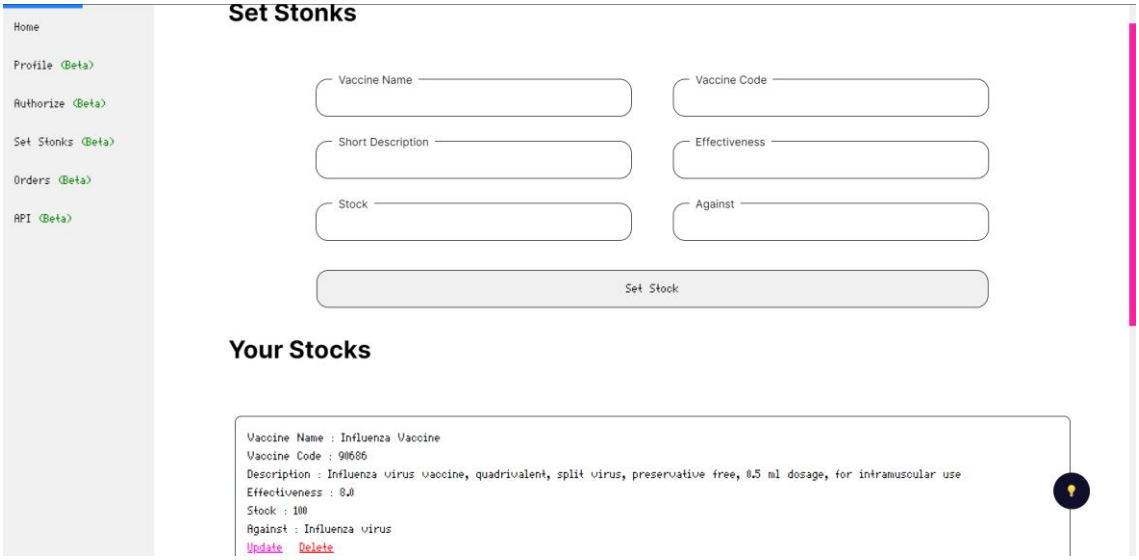


Figure 6.20

7.1 Conclusion

The most thing I learnt was time management, if we manage our time properly then we can finish anything before the deadline. I also learnt that designing and planning are one of the important things. This project took me through the various phases of project development and gave me real insight into the world of software engineering.

In conclusion, developing a vaccine management system using MongoDB and Node.js for our college project was a great learning experience. Through this project, we were able to gain practical experience in designing and implementing a full-stack application that can be used to manage vaccine distribution and administration.

One of the main benefits of using MongoDB as our database management system was its flexibility and scalability. Unlike traditional relational databases, MongoDB allowed us to easily store and retrieve data in a document-based format, which made it easy to manage and manipulate data in real-time. Additionally, the ability to scale our database horizontally across multiple servers allowed us to handle large amounts of data with ease, which is particularly important for a vaccine management system that requires fast and efficient data processing.

Node.js was also an excellent choice for our project, as it provided us with a fast and efficient platform for building server-side applications. Its event-driven architecture and non-blocking I/O model allowed us to develop a highly responsive and performant application that can handle a large number of requests simultaneously.

Throughout the development process, we faced several challenges, such as designing a user-friendly interface and ensuring data security. However, through collaboration and teamwork, we were able to overcome these challenges and deliver a fully functional application that met our project requirements.

One of the most rewarding aspects of this project was the knowledge and skills we gained from it. We learned about the importance of data management and security, as well as the benefits of using modern technologies such as MongoDB and Node.js. We also learned about the importance of collaboration and communication in a team-based project, which will undoubtedly be beneficial in our future careers.

In summary, our experience in developing a vaccine management system using MongoDB and Node.js was both challenging and rewarding. We gained valuable knowledge and skills, and we are proud of the final product we delivered. We believe that our application has the potential to make a positive impact in managing vaccine distribution and administration, and we hope to see it implemented in the real world.

Furthermore, this project has also given the opportunity to showcase our abilities to potential employers and stand out in a highly competitive job market. By demonstrating our ability to design and develop a functional application, we can confidently apply for positions that require experience in full-stack development, data management, and security. Overall, this project has been an invaluable experience that has prepared us for real-world challenges and opportunities.

In addition, the use of agile development methodologies helped us to streamline our development process, ensure that we were meeting project requirements, and keep track of progress in a timely manner. This allowed us to make necessary adjustments quickly, saving time and resources in the long run. Overall, this project has been a valuable learning experience that we can apply in our future endeavors.

7.2 Limitations

- Due to Financial Issues this Project is only Capable for users in Maharashtra.
- This Project is only capable to Handle Moderate Traffic as the hosting solution is based on 1 CPU core and 2GB RAM. (Financial Issue).
- Only has 512 MB of Database Storge Size (Financial Issue).

7.3 Future Scope:

- An Android or IOS application can be suitably developed reusing the RESTful APIs developed for the project
- Should be used anywhere in the world

5. Bibliography

Websites

- https://cdsco.gov.in/opencms/export/sites/CDSCO_WEB/Pdf-documents/biologicals/facilitiesLIST.pdf

  as of 19/05/22

- https://www.who.int/emergencies/diseases/novel-coronavirus-2019/question-and-answers-hub/q-a-detail/coronavirus-disease-(covid-19)-vaccines?adgroupsurvey={adgroupsurvey}

  as of 22/09/22

- https://www.servicenow.com/solutions/vaccine-management.htm
  as of 01/09/22
- https://www2.deloitte.com/us/en/pages/public-sector/solutions/vaccine-management-system.htm
  as of 02/06/22
- https://www.intelex.com/products/applications/vaccine-management}{Intelex Vaccine Solution
  as of 05/05/22
- https://www.zenflowchart.com/guides/data-flow-diagram
  as of 01/09/22
- https://www.newthinktank.com/2019/01/latex-tutorial/
  as of 28/06/22
- https://online.visual-paradigm.com/drive/#diagram:proj=0&type=GanttChart&workspace=mhujsaxv&id=2
  as of 12./07/22
- https://www.lucidchart.com/pages
  as of 16/08/22
- https://www.mongodb.com/docs/manual/core/data-modeling-introduction/
  as of 15/08/22
- https://www.lucidchart.com/pages/uml-use-case-diagram
  as of 21/08/22
- https://www.lucidchart.com/pages/uml-sequence-diagram
  as of 22/08/22
- https://www.lucidchart.com/pages/data-flow-diagram
  as of 23/08/22

- https://www.lucidchart.com/pages/uml-state-machine-diagram
  as of 21/08/22
- https://www.lucidchart.com/pages/uml-activity-diagram
  as of 20/08/22
- https://youtu.be/WnMQ8HlmeXc
  as of 07/09/22

Reference books

- Practical MongoDB: Architecting, Developing, and Administering MongoDB
  Author : Shakuntala Gupta Edward & Navin Sabharwal as of 16/07/22

- Software Engineering 9th Edition
  Author : Ian Sommerville as of 19/06/22

- Learning Node.js Development: Learn the fundamentals of Node.js, and deploy and test Node.js applications on the web
  Author : Andrew Mead as of 20/08/22

- Beginning Node.js, Express & MongoDB Development
  Author : Greg Lim as of 22/08/22

- Fundamentals of Creating a Great UI/UX
  Author: - Creative Tim as of 23/08/22