**Vaccine Management System**

**(Drug\_Lord\_v.1.10)**

Index

[Project Synopsis 1](#_Toc114672491)

[Title. 1](#_Toc114672492)

[Problem Statement 1](#_Toc114672493)

[Why this Topic? 1](#_Toc114672494)

[Objective and Scope. 1](#_Toc114672495)

[Methodology for developing project. 1](#_Toc114672496)

[Proposed Architecture 1](#_Toc114672497)

[Requirements 2](#_Toc114672498)

[Platform 2](#_Toc114672499)

[Contribution 3](#_Toc114672500)

[Conclusion 3](#_Toc114672501)

[1. Introduction 4](#_Toc114672502)

[1.1 Background 5](#_Toc114672503)

[1.2 Objective 5](#_Toc114672504)

[1.3 Purpose 5](#_Toc114672505)

[1.4 Application 5](#_Toc114672506)

[1.5 Scope 5](#_Toc114672507)

[1.6 Achievements 5](#_Toc114672508)

[2. Survey of Technology 7](#_Toc114672509)

[3. Requirement and Analysis 10](#_Toc114672510)

[3.1 Problem Definition 11](#_Toc114672511)

[3.2 Requirements Specification 12](#_Toc114672512)

[3.3 Planning & Scheduling 16](#_Toc114672513)

[3.4 Hardware & software requirements 18](#_Toc114672514)

[3.5 Conceptual Model 19](#_Toc114672515)

[3.5.3 Use-Case Diagram 24](#_Toc114672516)

[3.5.4 Sequence Diagram 27](#_Toc114672517)

[3.5.5 Activity State Diagram 33](#_Toc114672518)

[3.5.6 Activity Diagram 34](#_Toc114672519)

[Bibliography 41](#_Toc114672520)

[websites 41](#_Toc114672521)

[Reference books 41](#_Toc114672522)

# 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

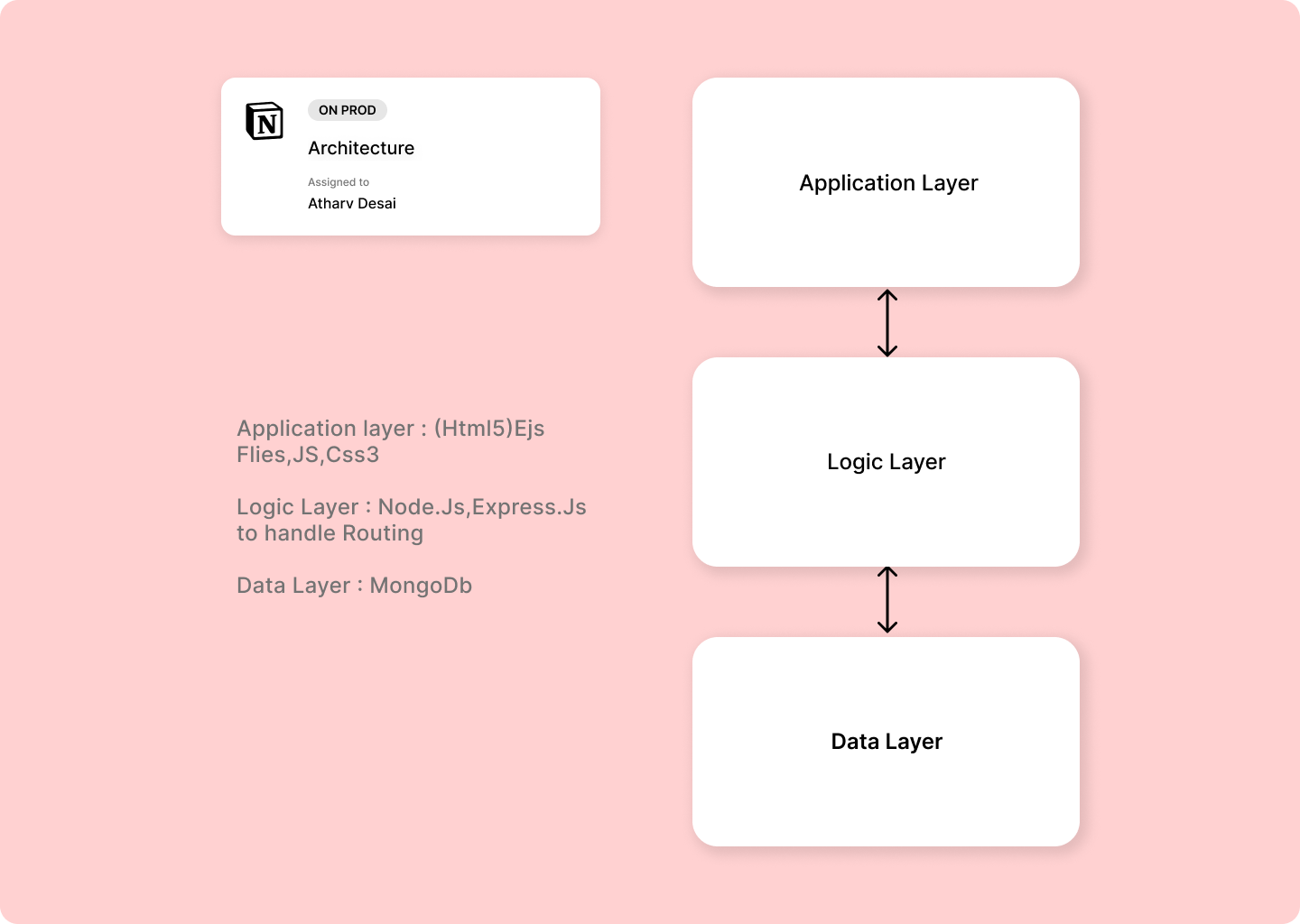
## Methodology for developing project.

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 1.7.1*



## 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.

# **Introduction**

## **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 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](https://www.intelex.com/products/applications/vaccine-management%7d%7bIntelex%20Vaccine%20Solution)

## **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.

## **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.

## **Application**

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

## **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.

## **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 relations

# **Survey of Technology**

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

# **Requirement and Analysis**

## **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

### 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 to schedule appointments for the people?
* For Producers
  + How to set up your vaccine Stocks?
  + How to process Orders?
  + How to Deliver?

### 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

## **Requirements Specification**

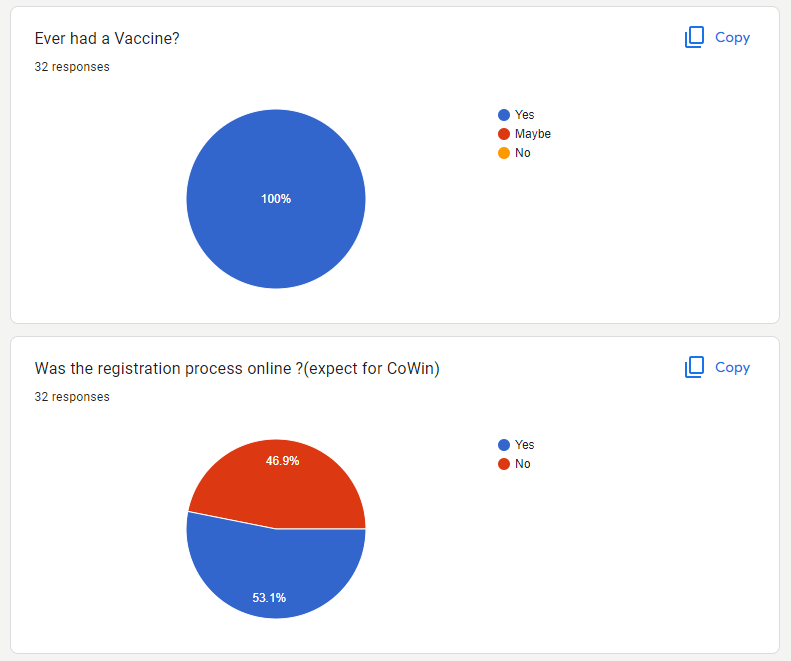
### Requirement analysis

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

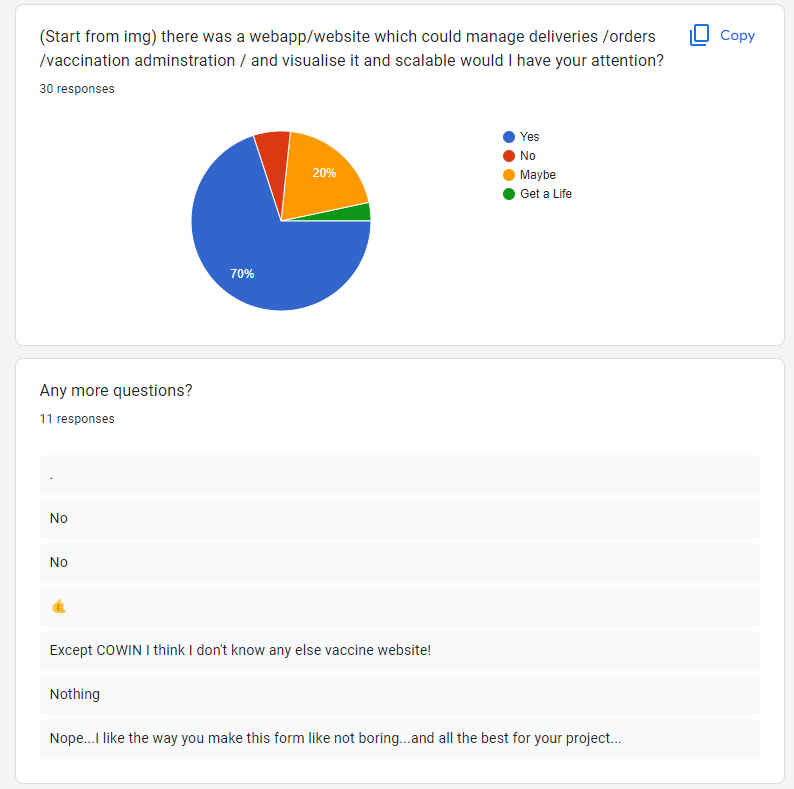
* + - Capture requirements
      * Holding One-One interviews
      * Conduct Group Workshops
      * Get Feedbacks
      * Build Small Prototypes

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

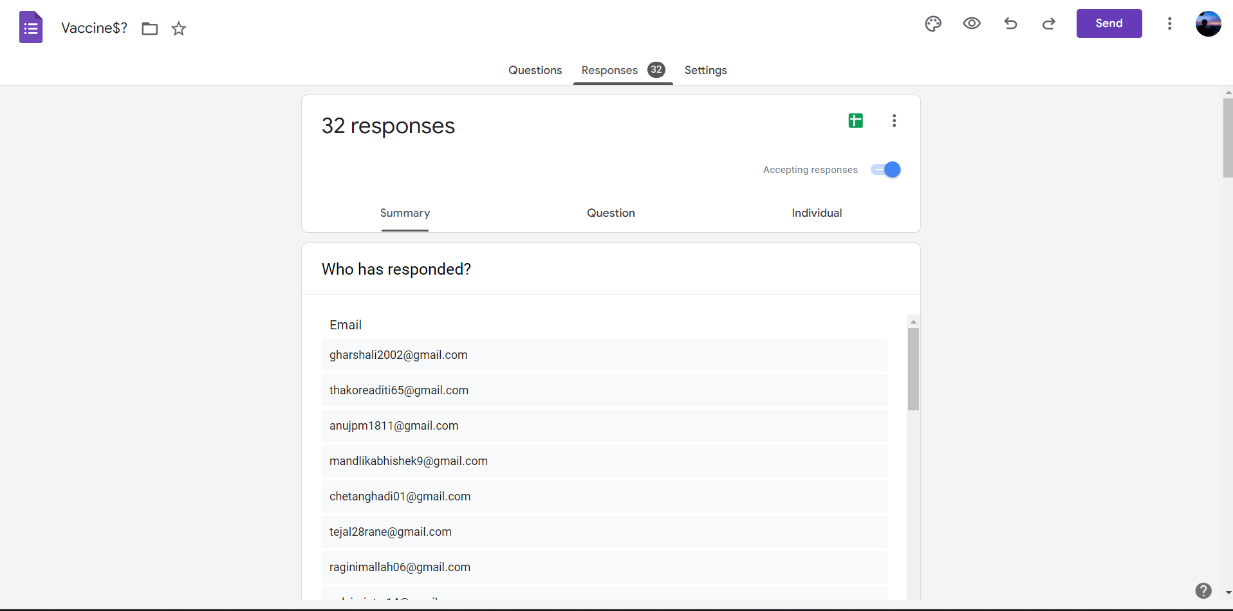
The below figures are the collected data that was generated.



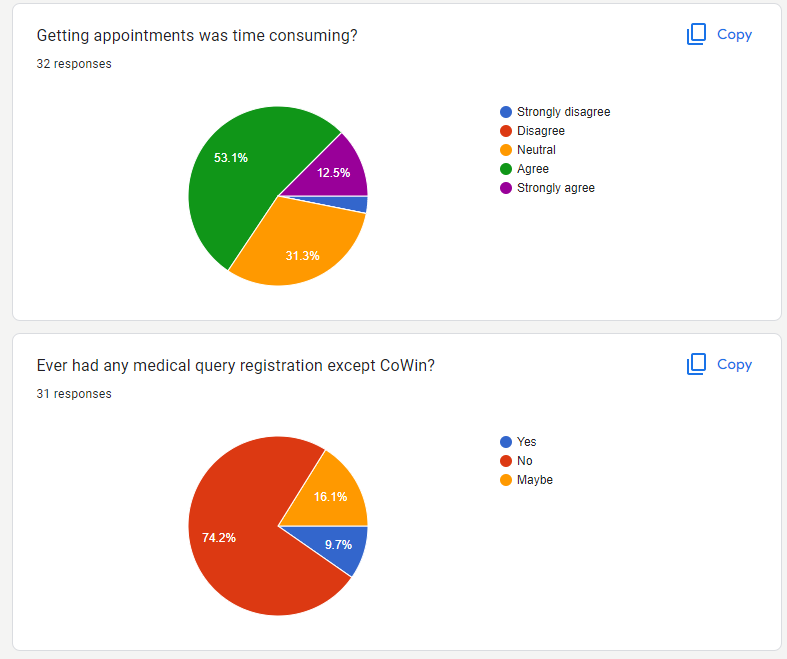
*Figure 3.2.1*



*Figure 3.2.2*



*Figure 3.2.3*



*Figure 3.2.4*

### 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

### 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

### 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: Error message is shown if credentials don’t match

* Register

Function: Used for registering account

Input: User, Provider, Producer all details

Pre-Condition: None

Source : From User, Provider, Producer

Post-Condition: Redirected to Login if details are valid and correct

Exception: 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: Users details are updated

Exception: 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: Users reviews get added

Exception: 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: Error message is shown if anything goes wrong

* Set appointments

Function: Used for setting appointments

Input: Provider, Producer all details for setting appointment

Pre-Condition: None

Source : Provider

Post-Condition: Appointment is generated

Exception: 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: 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 is updated or created

Exception: Error message is shown if anything goes wrong

//three more remaining

## **Planning & Scheduling**

### Activity Table

|  |  |  |
| --- | --- | --- |
| Activities | Start-Date | End-Date |
| Synopsis | 27/04/22 | 16/06/22 |
| Introduction | 20/06/22 | 25/06/22 |
| Survey of Technology | 20/06/22 | 25/06/22 |
| Requirement and analysis | 27/06/22 | 15/09/22 |
| Problem Definition | 4/07/22 | 9/07/22 |
| Requirement Specification | 11/07/22 | 16/07/22 |
| Planning & Scheduling | 18/07/22 | 23/07/22 |
| Hardware & Software Requirement | 18.07/22 | 23/07/22 |
| Conceptual Models | 26/07/22 | 15/09/22 |

### Gantt Chart



*Figure 3.3.1*

## **Hardware & software requirements**

### 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

### Software

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

## **Conceptual Model**

### Data Model

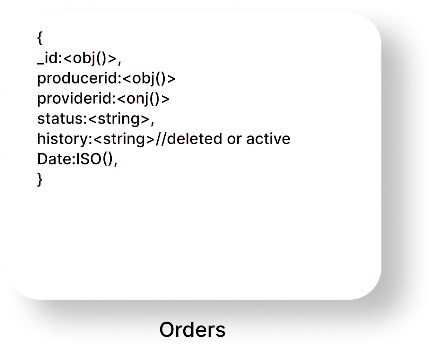
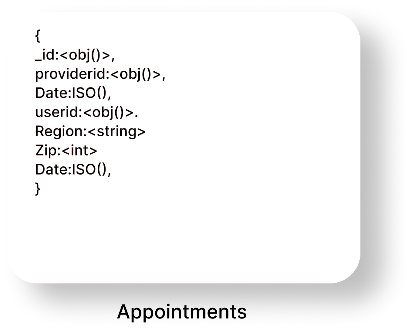
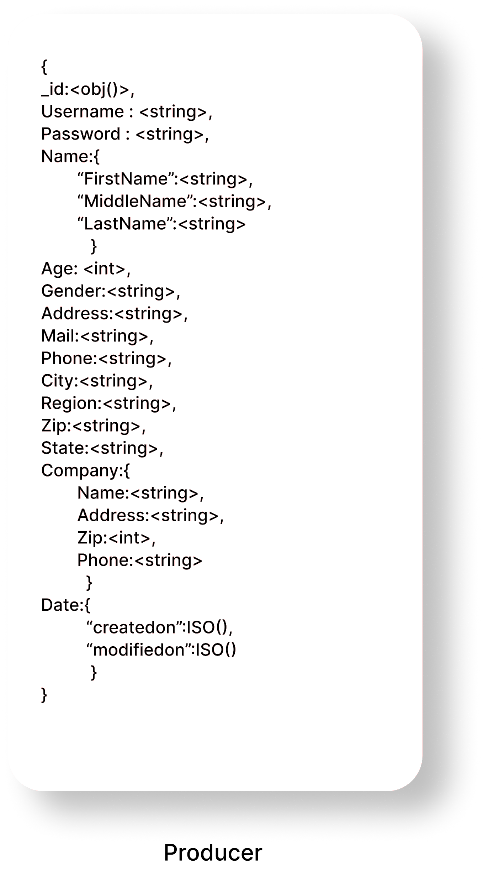
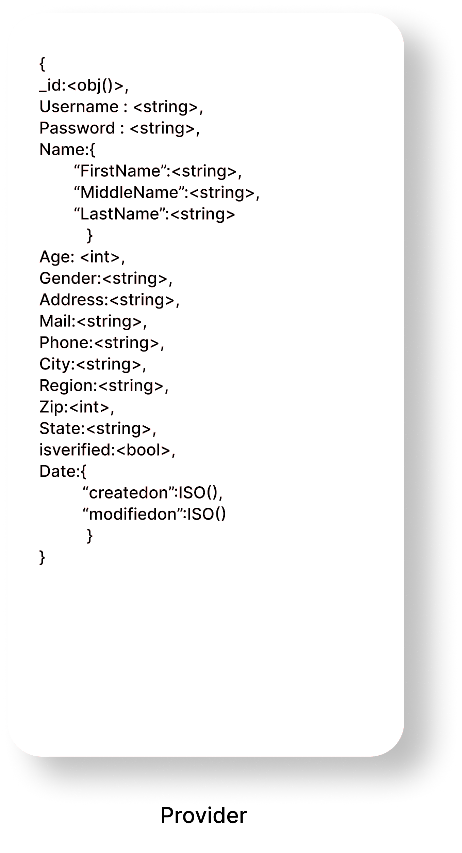
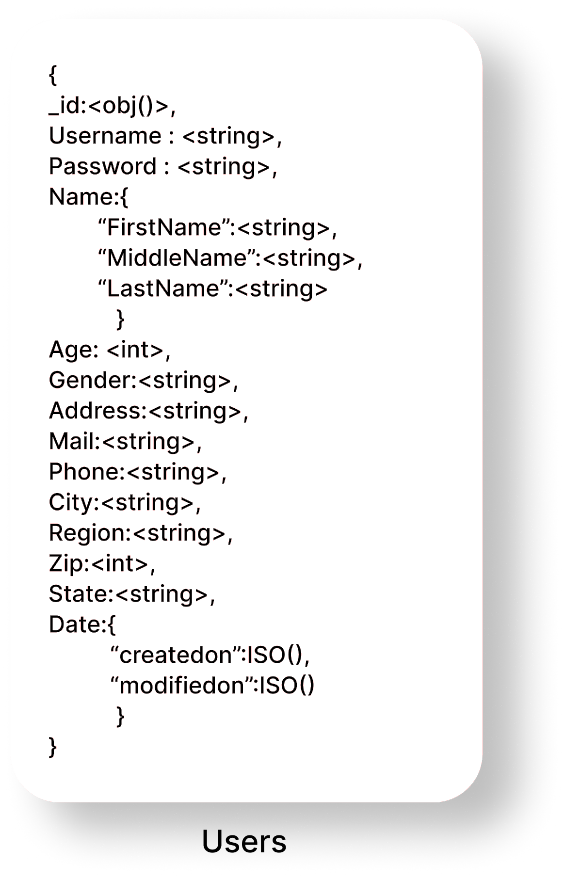
Unlike SQL databases, where you must determine and declare a table's schema before inserting data, MongoDB's [collections](https://www.mongodb.com/docs/manual/reference/glossary/#std-term-collection), by default, do not require their [documents](https://www.mongodb.com/docs/manual/core/document/) to 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.

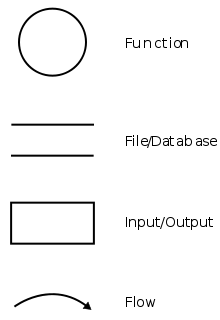
The key decision in designing data models for MongoDB applications revolves around the structure of documents and how the application represents relationships between data. MongoDB allows related data to be embedded within a single document.



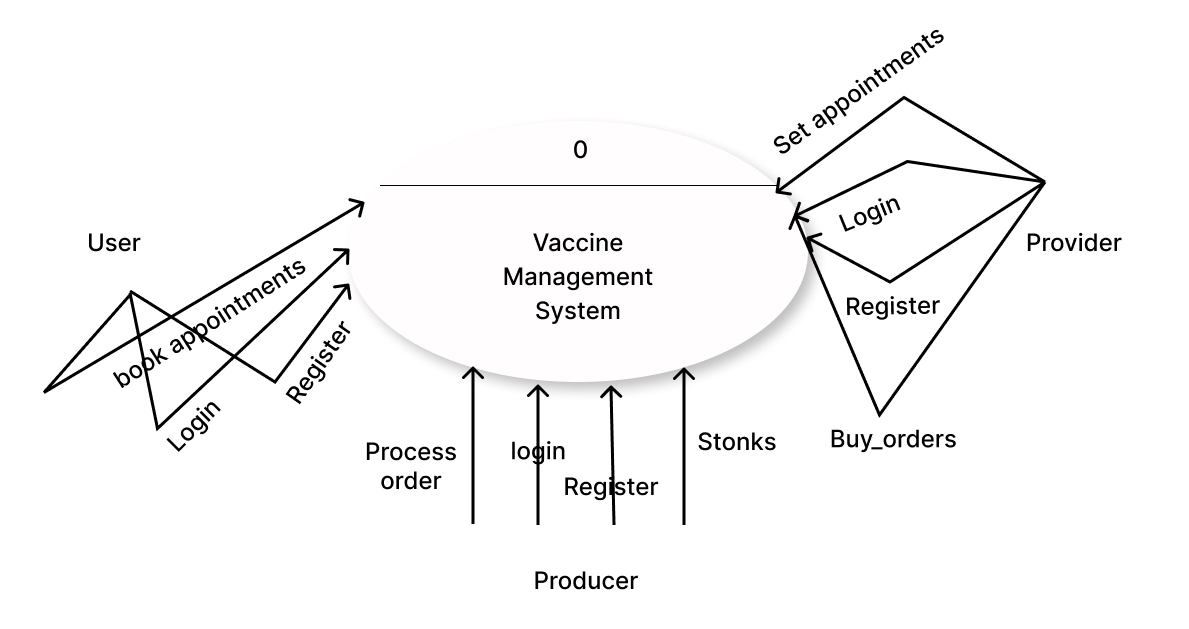
*Figure 3..1*

### Data-Flow Diagram

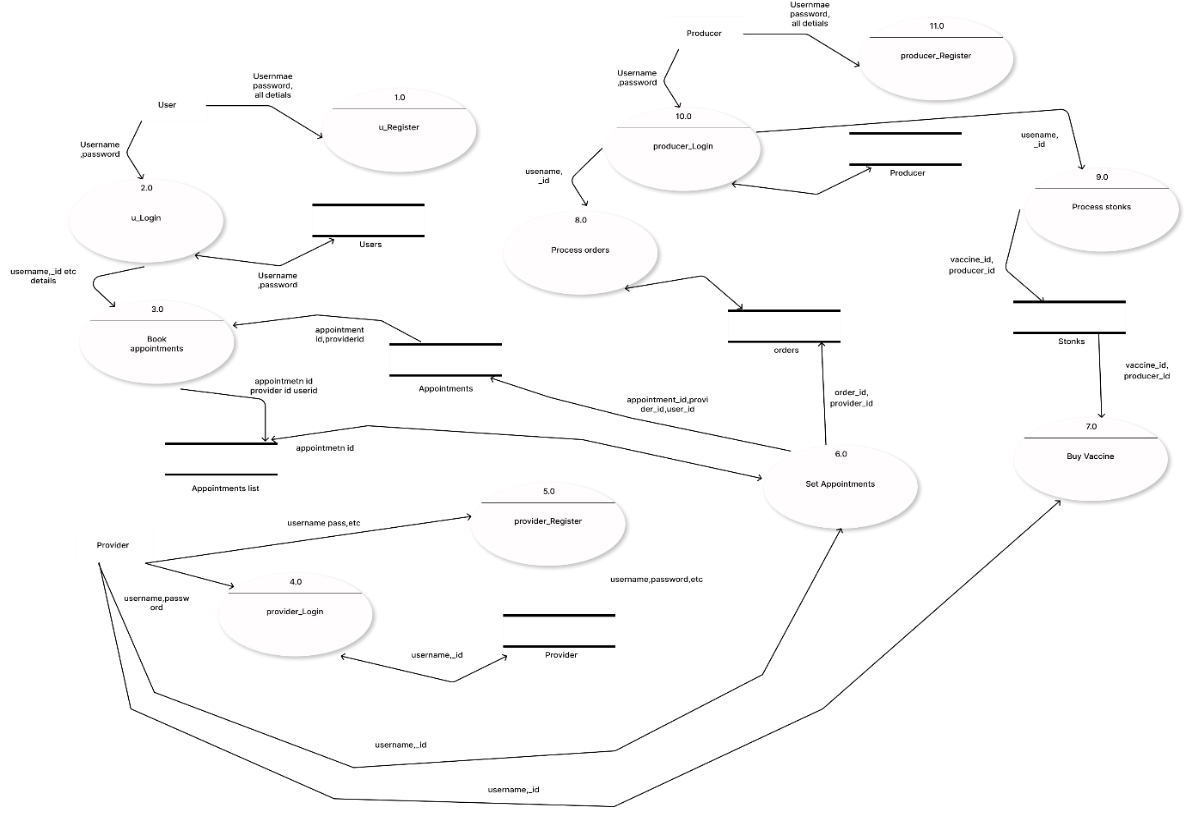
A **data flow diagram** (or DFD) is a graphical representation of the information flow in a business process. It demonstrates how data is transferred from the input to the file storage and reports generation. By visualizing the system flow, the flow charts will give users helpful insights into the process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



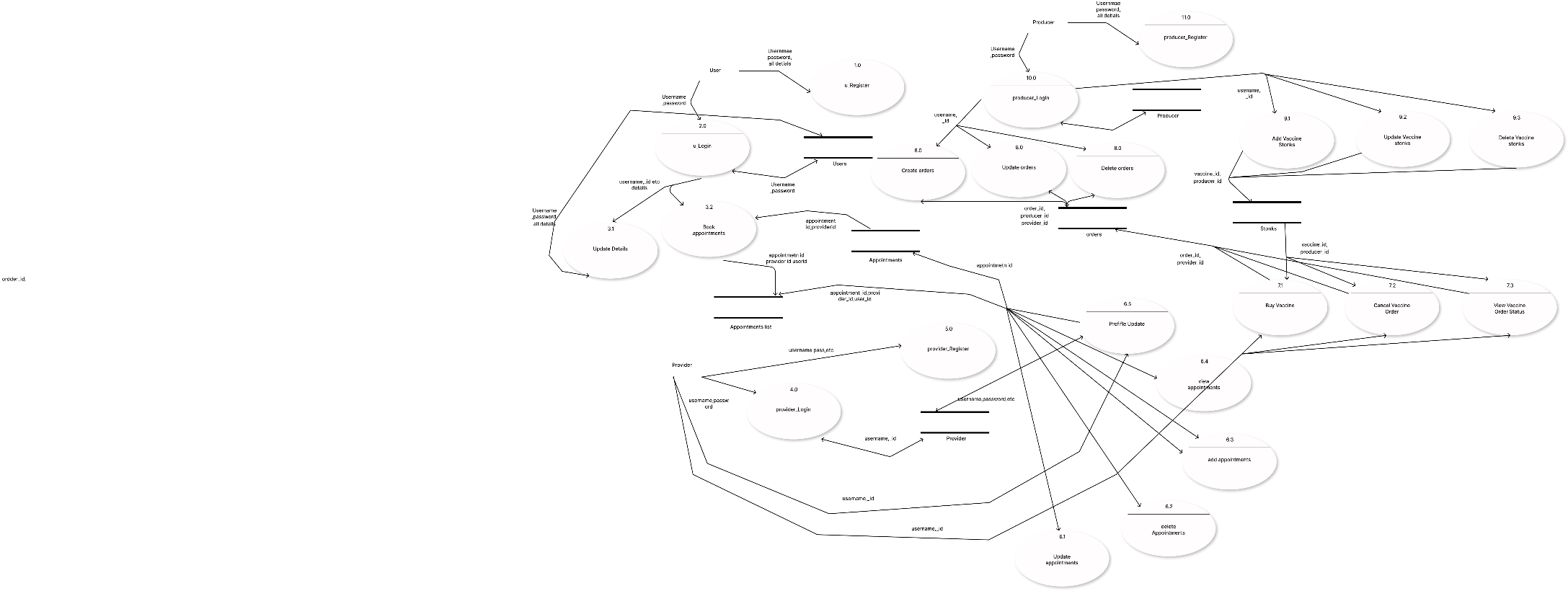
*Figure 3..2*



*Figure 3.3*



*Figure 3.4*

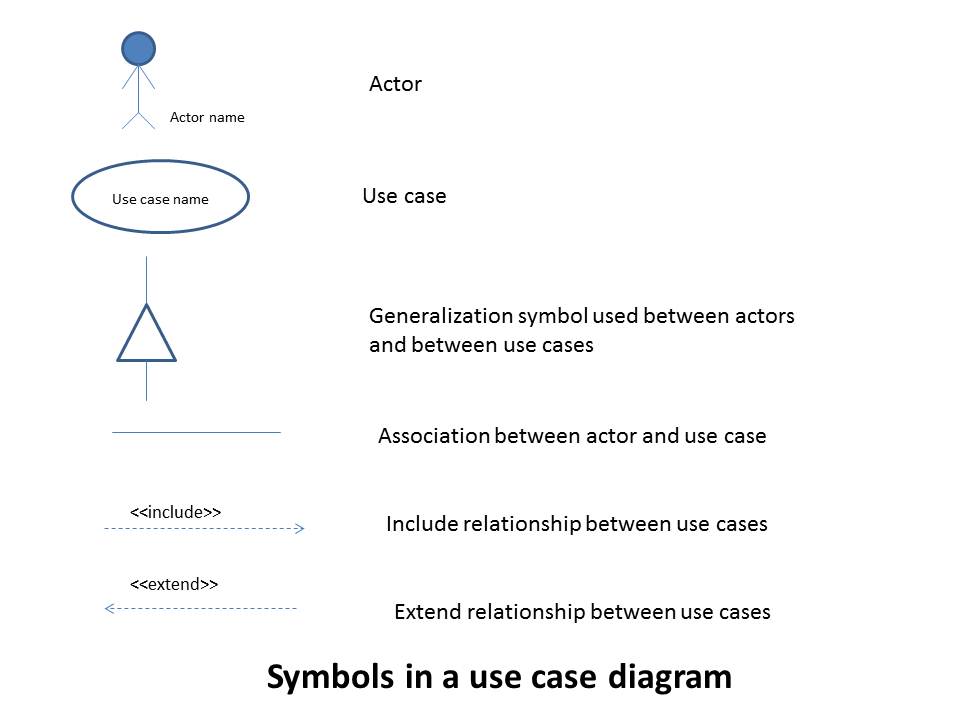


*Figure 3.5*

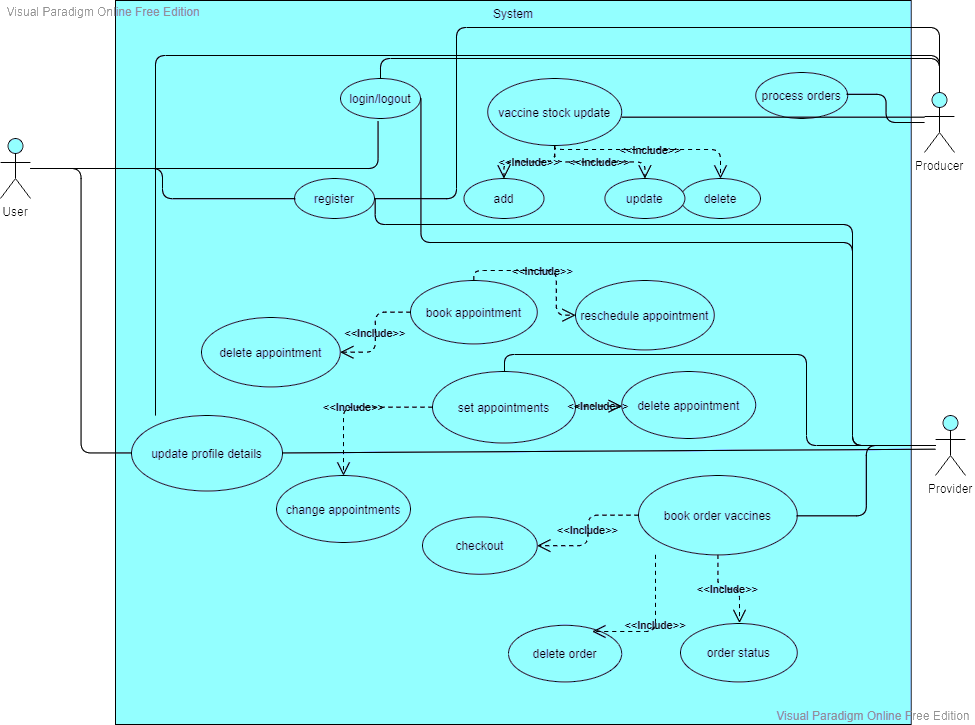
## Use-Case Diagram

In the Unified Modeling 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



*Figure 3.6*



*Figure 3.7*

Use Case Description:

* Login

Summary: Used to Login 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 and 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 a 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 and account in the service by fill certain details

Exception: If slots gets 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.

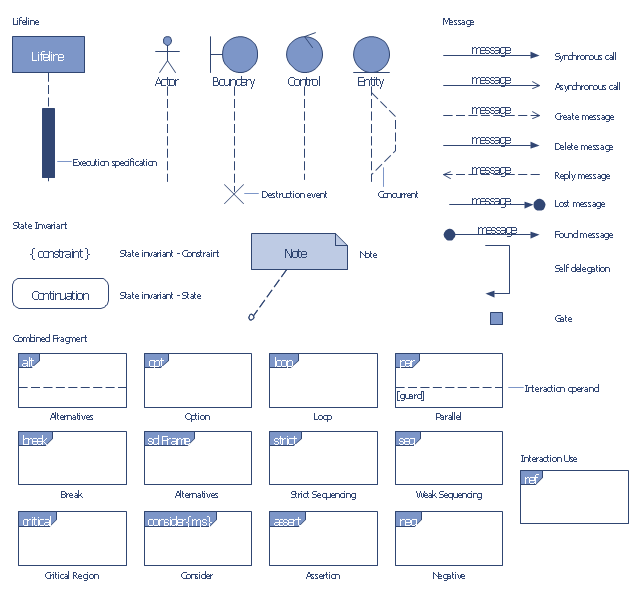
Exception: If an internal problem happens error message is thrown.

Post-condition: Order is successfully processed

## Sequence Diagram

[UML](https://en.wikipedia.org/wiki/Unified_Modeling_Language) 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.

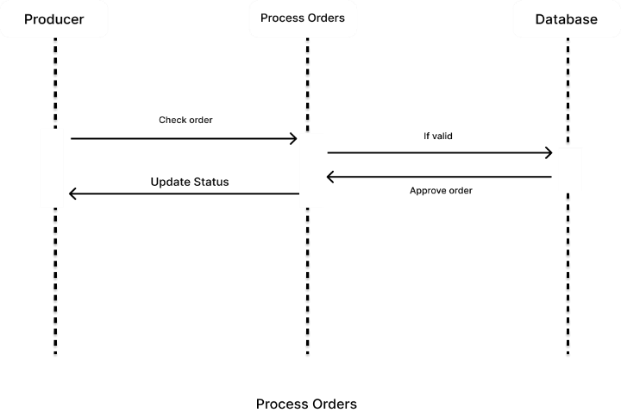
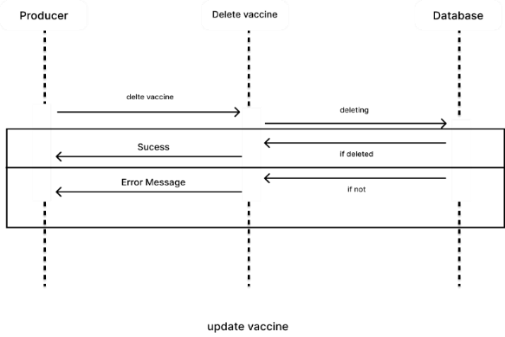
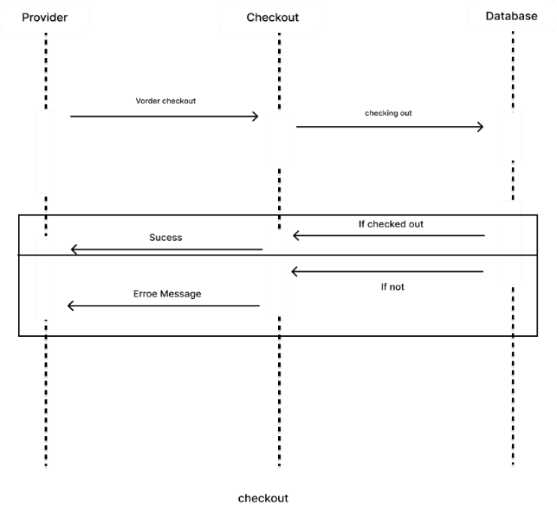
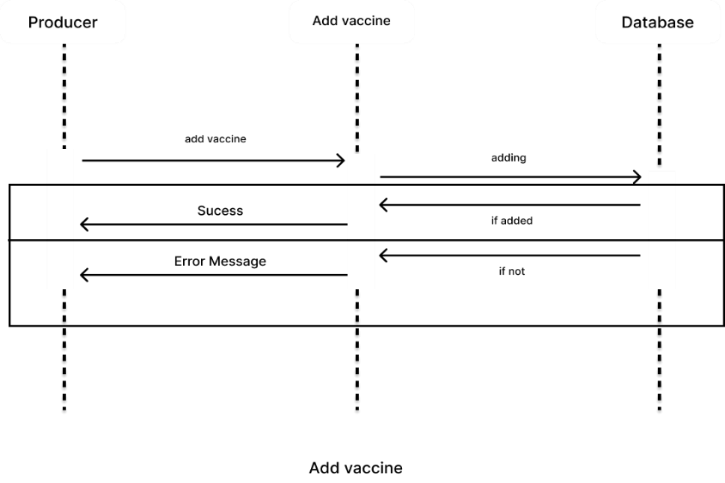
* Model high-level interaction between active objects in a system
* Model the interaction between object instances within a collaboration that realizes a use case
* Model the interaction between objects within a collaboration that realizes an operation
* Either model generic interactions (showing all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



*Figure 3.8*

*Figure 3.11*

*Figure 3.9*

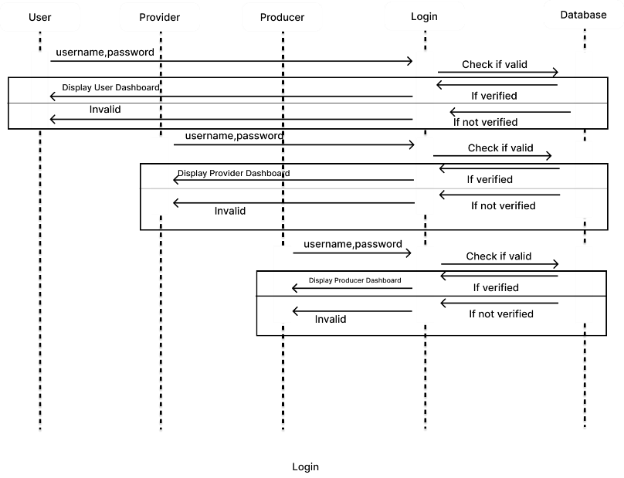
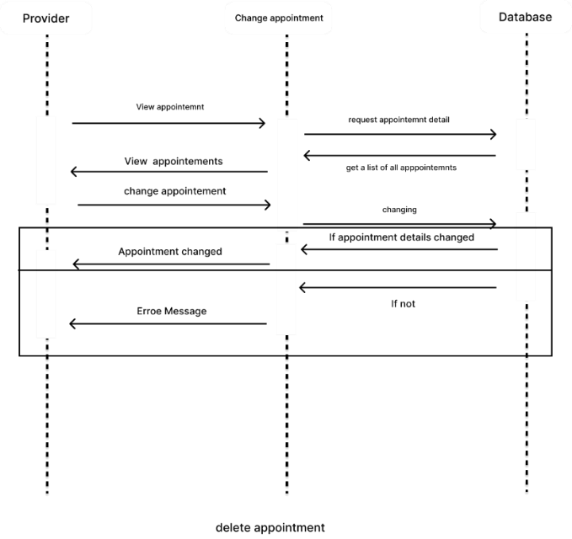
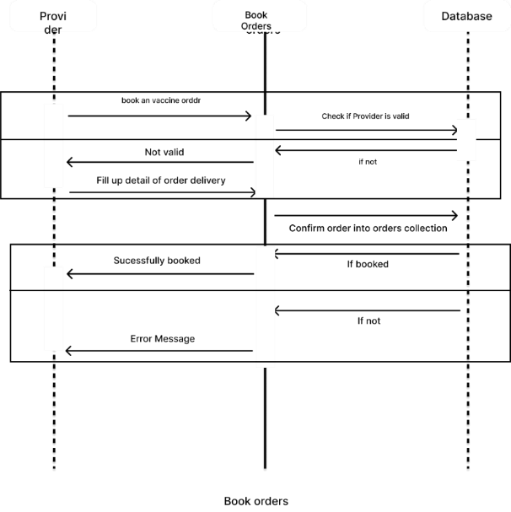
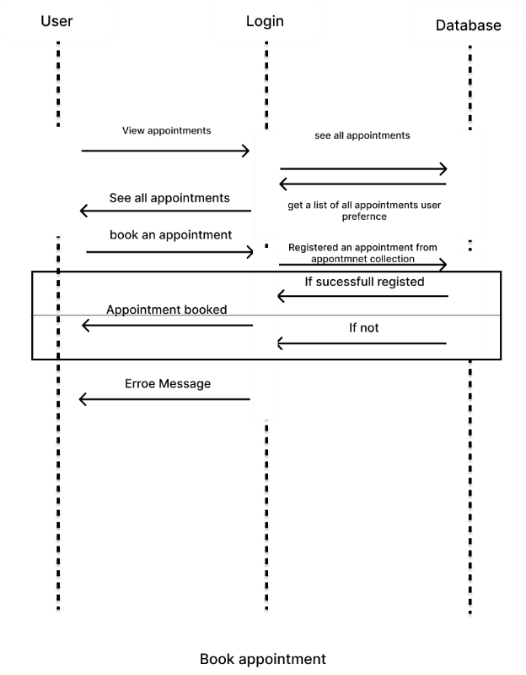


*Figure 3.10*

*Figure 3.12*

*Figure 3.13*

*Figure 3.14*

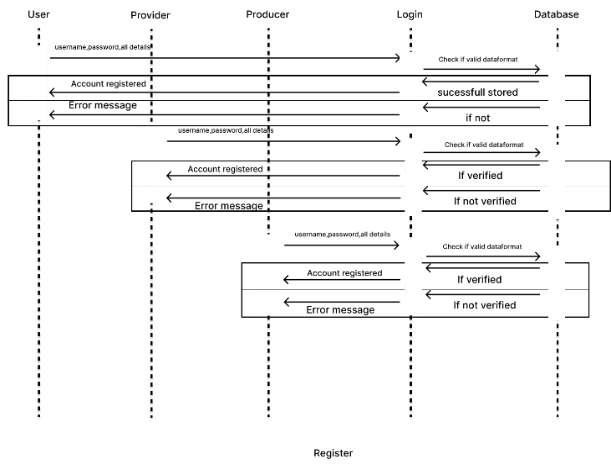
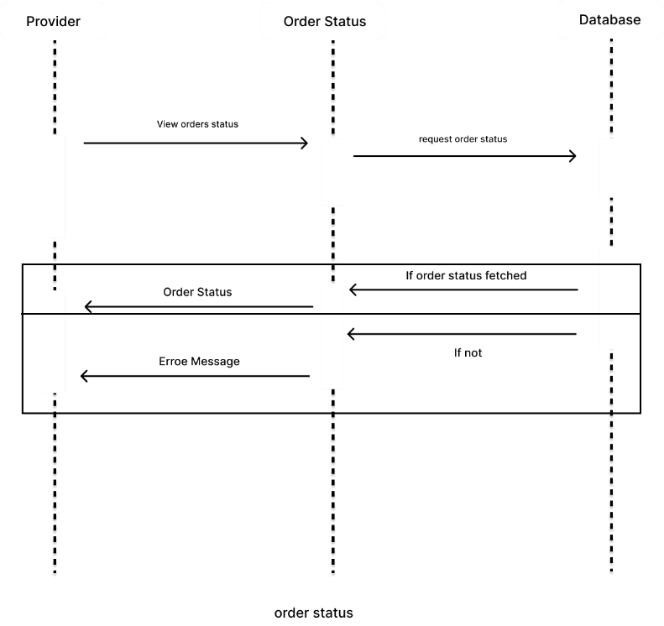
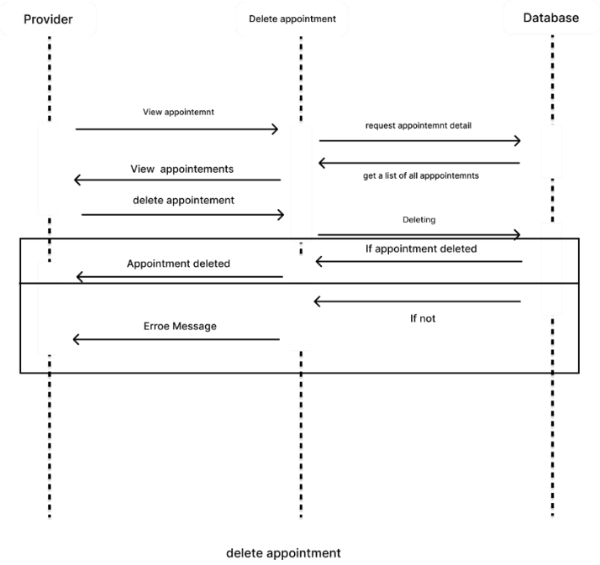


*Figure 3.16*

*Figure 3.15*

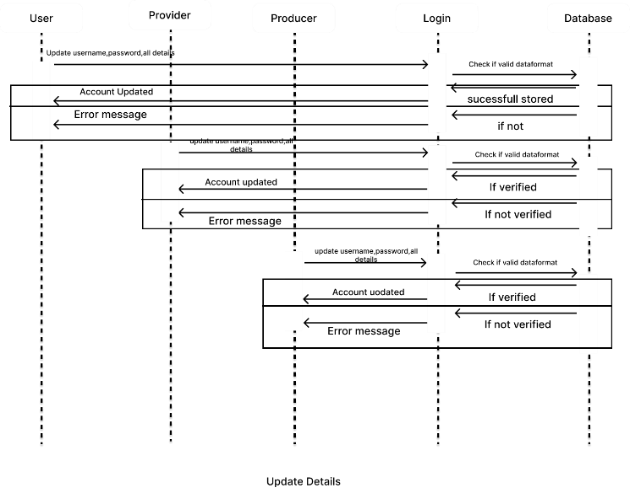
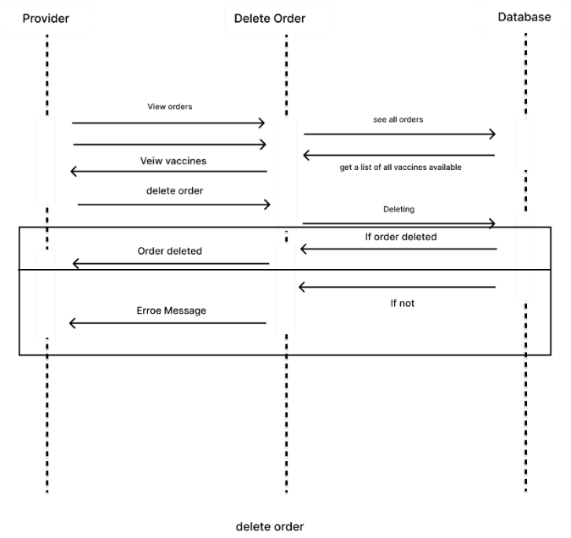
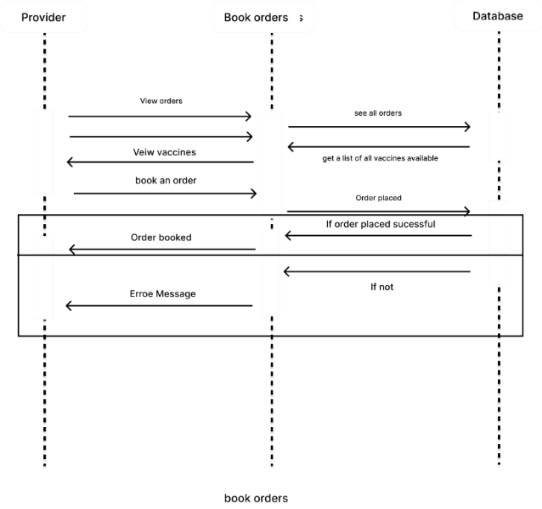
*Figure 3.17*

*Figure 3.18*



*Figure 3.19*

*Figure 3.20*

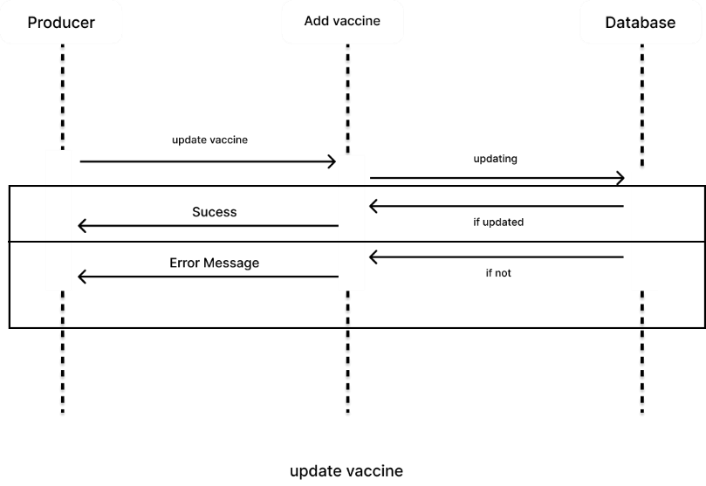
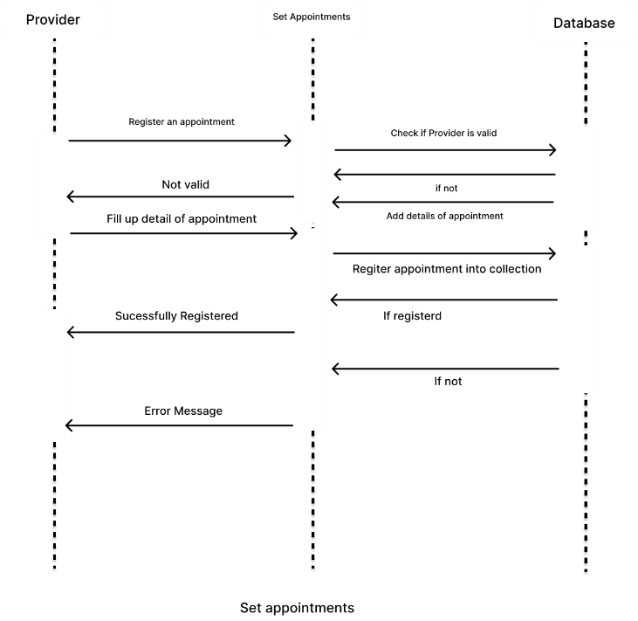


*Figure 3.21*

*Figure 3.22*

*Figure 3.23*

*Figure 3.24*



*Figure 3.25*

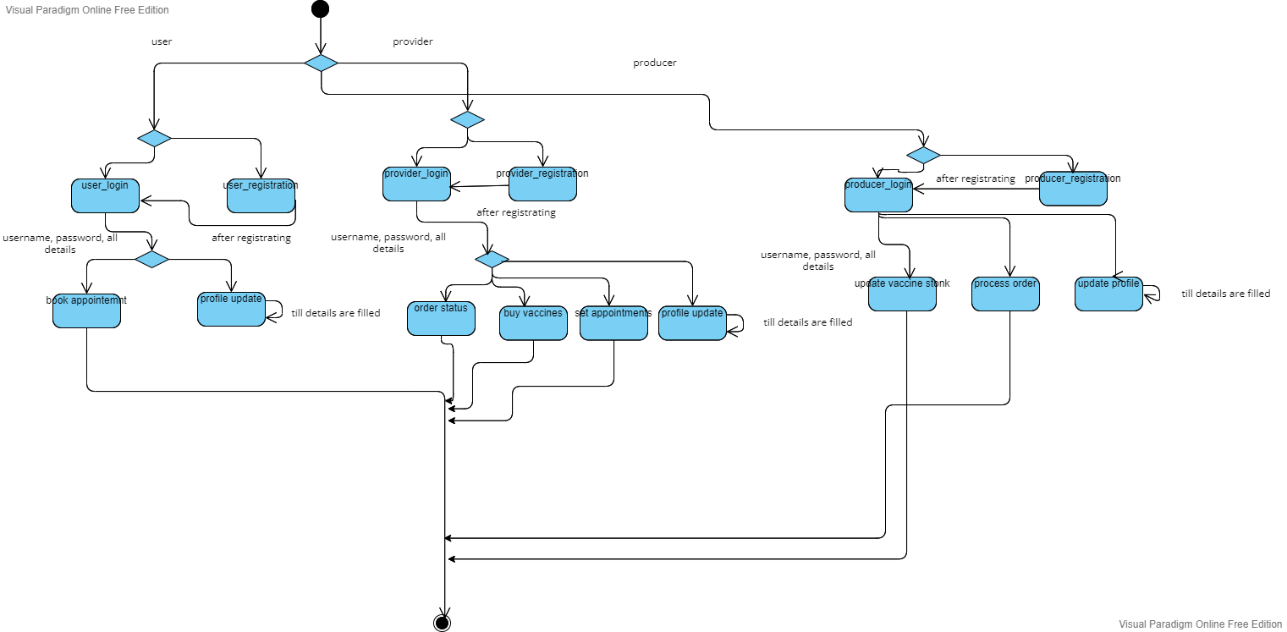
## Activity State Diagram

Activity diagrams present a number of 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.



*Figure 3.26*

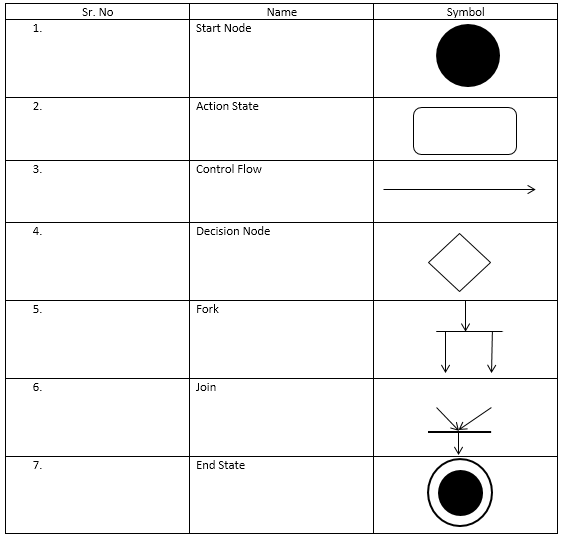


*Figure 3.27*

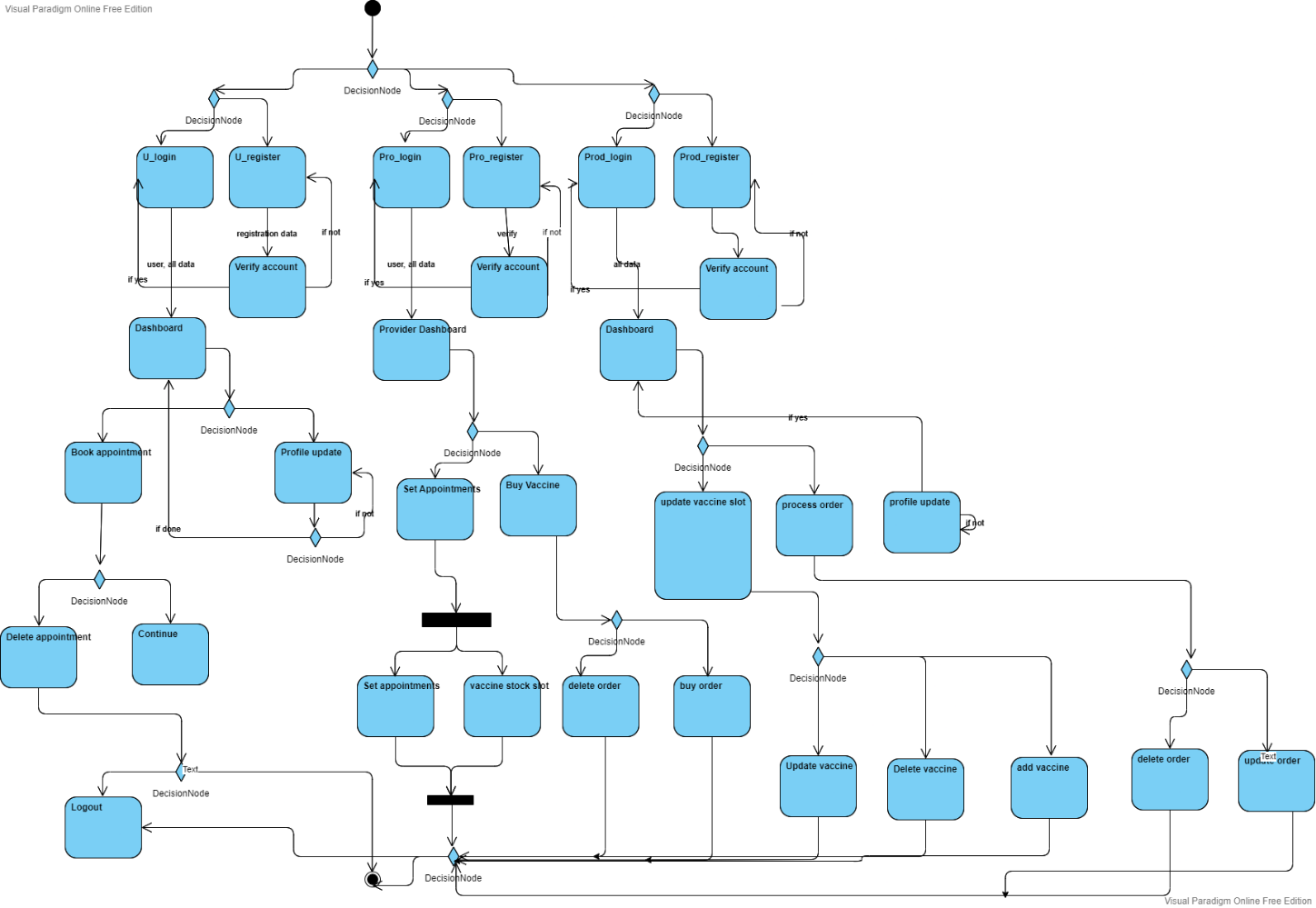
## Activity Diagram

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 about the elements used in 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.



*Figure 3.28*



*Figure 3.29*

# **System Design**

## Interface Design

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna.

Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci.

Aenean nec lorem. In porttitor. Donec laoreet nonummy augue.

Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis, nunc. Mauris eget neque at sem venenatis eleifend. Ut nonummy.

## Test Cases

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna.

Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci.

Aenean nec lorem. In porttitor. Donec laoreet nonummy augue.

Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis, nunc. Mauris eget neque at sem venenatis eleifend. Ut nonummy.

# Bibliography

## websites

* <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](https://www.intelex.com/products/applications/vaccine-management%7d%7bIntelex%20Vaccine%20Solution)
* <https://www.zenflowchart.com/guides/data-flow-diagram>

## Reference books

* Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna.
* Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus.
* Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci.
* Aenean nec lorem. In porttitor. Donec laoreet nonummy augue.
* Suspendisse dui purus, scelerisque at, vulputate vitae, pretium mattis, nunc. Mauris eget neque at sem venenatis eleifend. Ut nonummy.