

# SOFTWARE DESIGN DOCUMENT

*for*

## **Vaccine management System**

September 2021

<b><u>1. INTRODUCTION</u></b> .....	<b>3</b>
<u>1.1. PURPOSE</u> .....	3
<u>1.2. SCOPE</u> .....	3
<b><u>2. ARCHITECTURAL DESIGN</u></b> .....	<b>3</b>
<b><u>3 SYSTEM INTERFACE DESIGN</u></b> .....	ERROR! BOOKMARK NOT DEFINED.
<b><u>4 DATABASE DESIGN</u></b> .....	ERROR! BOOKMARK NOT DEFINED.
<b><u>5 TESTING</u></b> .....	ERROR! BOOKMARK NOT DEFINED.

## 1. INTRODUCTION

This software design document (SDD) is a formal architectural blueprint for Vaccine Management System. The layout is structured to help the reader recognize the requirements imposed upon the system and how these requirements are being met.

### 1.1. Purpose

This document describes all the standards and design of VMS (Vaccine Management System) to be used by the developers to follow in the implementation phase. The SDD is an important reference not only for developers, but also for the testing programmers and editors of documentation.

### 1.2. Scope

VMS is software that needs to be deployed on a central server used by the admins. VMS will be used to run vaccine campaigns and will allow government agency to manage their campaigns. The users of the system are the people who need to get their vaccines. They should be able to book their appointments using the web interface of the software.

## 2. ARCHITECTURAL DESIGN

The applications web Interface will be built using C# and .NET whereas the Database will use Mongo DB.

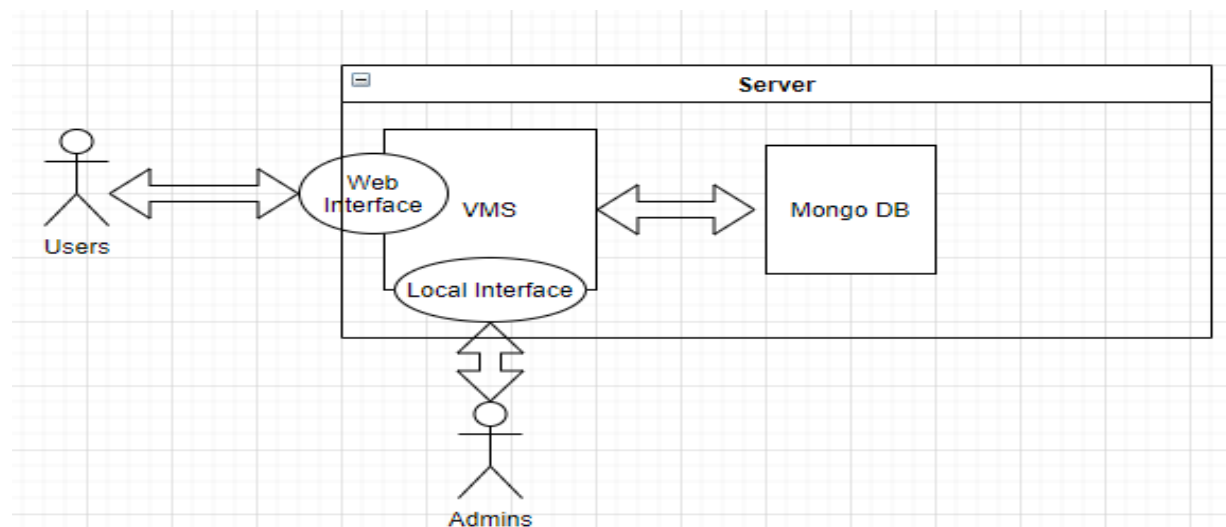


Figure 1: High level Architecture

The system also has two Crypto Keys that will be generated when the system starts for the first time and are stored in the specific files that are password protected.

1: Encryption Key: This key will be used to encrypt all the data that goes and comes out of the database.

2: Certificate Key: This key will be used to sign all the certificates.

Using two key ensures that if one system is compromised the other is still secure.

The users of the system namely admins and the users access it through web interface and local interface using two different URLs respectively. The admin URL will be only available on the Government agencies network and direct web access needs to be barred and is incumbent on the agency to handle this.

We use MVC : Model View and Controller framework of ASP .NET to implement the entire software. The use of MVC allows faster development by dividing the project into models(objects), Views (The actual UI) and the controller(methods to be invoked on user input). The MVC framework allows handling of the incoming http request and automatically invokes the correct controller.

For database we focus on using mongoDB.

### 3. SYSTEM DESIGN

#### 3.1 VMS Modules:

The VMS comprises of the following components:

##### 1: **Authentication**

The purpose of this module is to manage Signup and login for both users and admins. It will also implement Multi Factor Authentication using .NET core Identity library. The current MFA only consists of 2FA whereby the admin will receive a code on their registered email id.

This component also manages authorization and will only allow users who have verified their emails. For email verification user manager library of the asp.net core will be used. The Authentication component will send a link on the email to verify the users and admins.

It also needs to implement password criteria. The password must check all the criteria mentioned below:

- 1: Must be greater than 8 characters.
- 2: Must have numbers.
- 3: Must have capital letters.
- 4: Must have one special character.

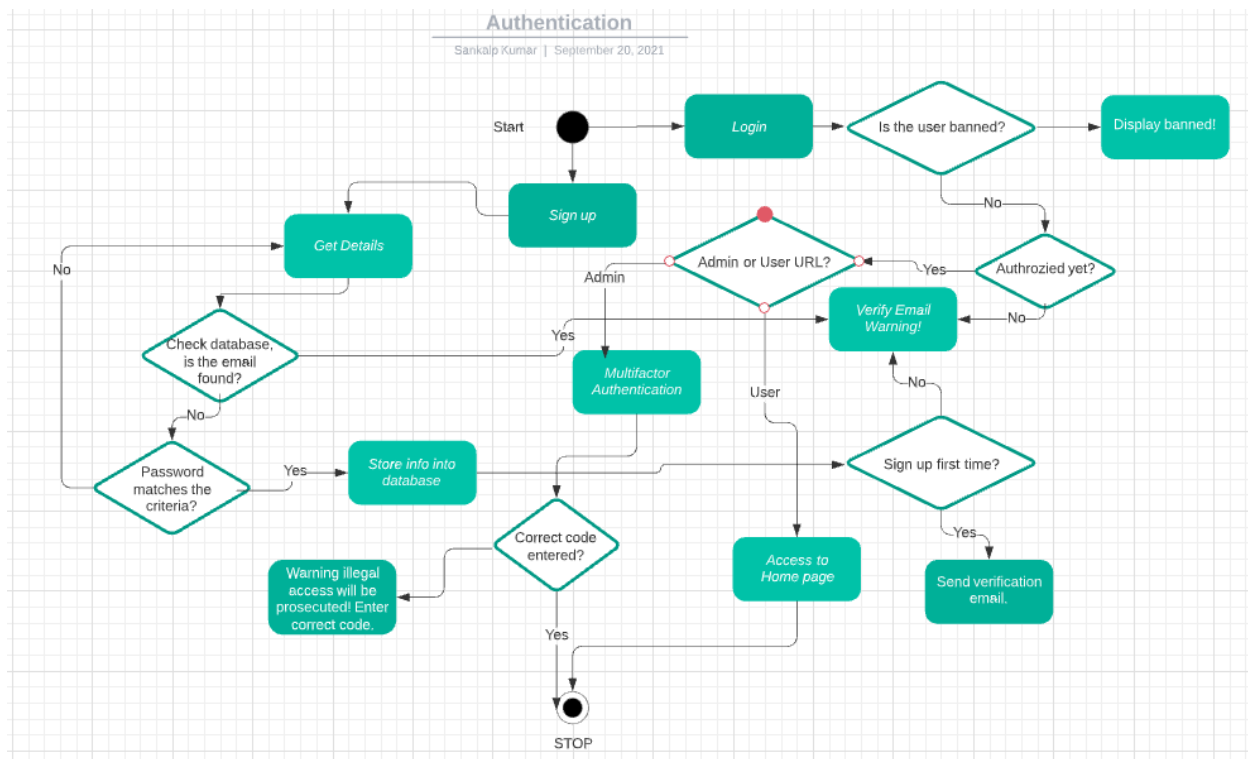


Figure 2: Flow Chart for Authentication

## 2: Key management layer

This layer manages all kinds of keys be it password or crypto keys. This is responsible for mandating password policies and recreating keys.

The module regenerates keys and re-encrypts the entire database on a yearly basis. Since, this could lead to all user's certificate getting invalidated suddenly in one shot, there could be a scenario where all users would want to renew their certificates, and this could lead to high load on server. This being a performance problem could be solved in a later release whereby the re-encryption happens in phases.

It also mandates admins to change their key on quarterly basis by running checks on date time stamps stored in databases.

## 3: Database Layer

The layer interfaces with the database and runs all transaction needed between the application and database.

Following are the function that the module must implement:

- 1: setUserDetails: sets the user details upon signup.
- 2: getUserInfo: takes input one or more fields to get from the database.

The setUserDetails also must implement the following functionality:

- 1: Hash the password using SHA256.
- 2: Add salt to the password.
- 3: Use Encryption key to encrypt all data before storing it in the database.

The encryption will be done using AES256-CBC using openssl library.

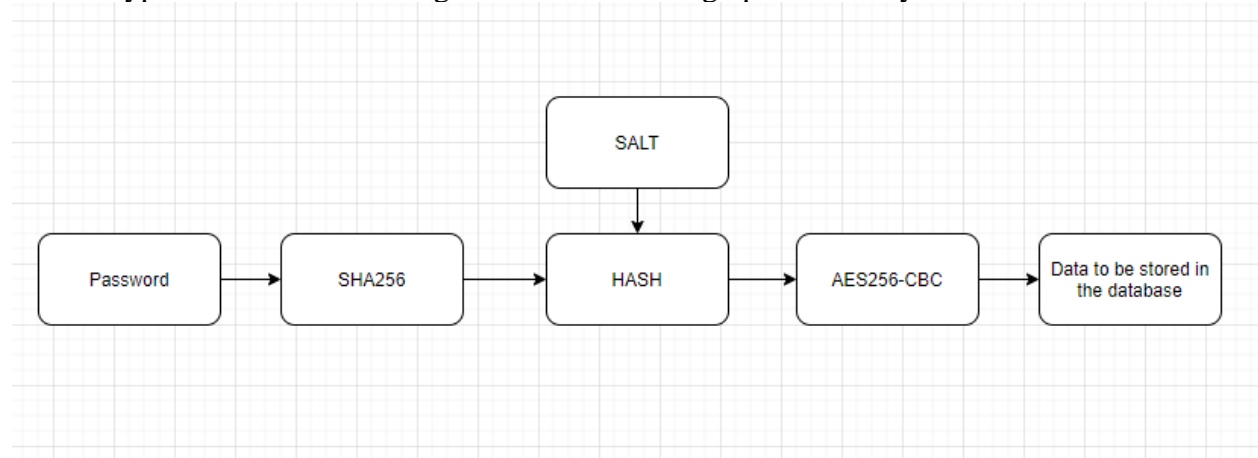


Figure 3: Storing the password

#### 4: Usage limit layer

The layer sees over banning of Users and admins. We track each user of the system using their email and overusing the limit leads to banning of the user for a week. We do not use IP as this could be misused easily. Hence, the single source of truth for tracking users is their email address.

Following are the limits set for User:

- 1: If the user attempts to look for more than 5 zip codes for vaccine centers.
- 2: If the user tries to book more than one appointment for the same vaccine. Once this happens thrice it will lead to ban. The website doesn't support functionality for cancelling an appointment. This will be considered for future release.

This layer runs a thread as well to check if the ban needs to be list. This thread is run every day at 7:00AM.

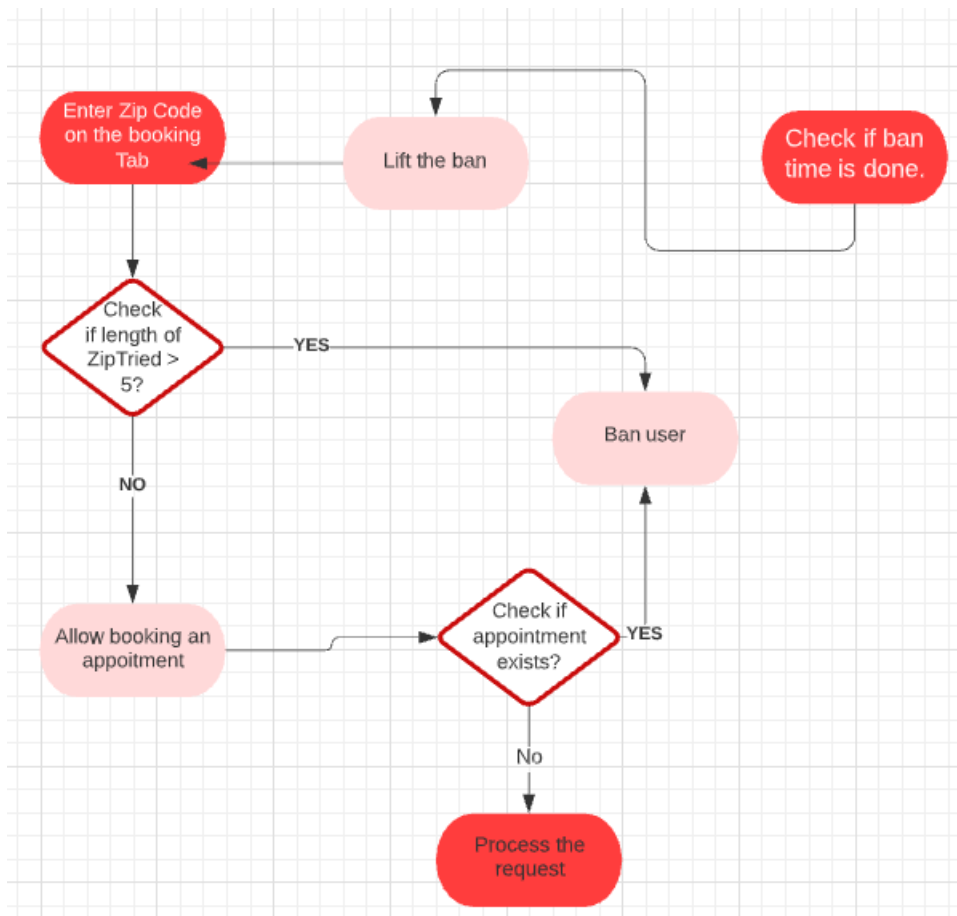


Figure 4: flowchart for user usage limit

Following are the limits to be set on Admins:

1: Admin attempts to create more than 3 vaccine campaigns in a day.

## 5: Appointment attendance verification

This layer generates an email just before the appointment and the user needs to enter this code in a screen that is only available via the admin URL of the application.

## 6: Certificate generation Layer

The Layer generates the certificate once the User has attended the appointment.

The certificate contains a hash created out of all user data using SHA256 and then encrypted using the Certificate key.

The algorithm to be used for encryption is AES256-CBC which is an approved symmetric key algorithm specified in FIP 140-2 published in the year 2019. We use openssl to generate keys and perform encryption and decryption. The paraphrase will change with each release. Due to security concerns the paraphrase has not been mentioned in this document.

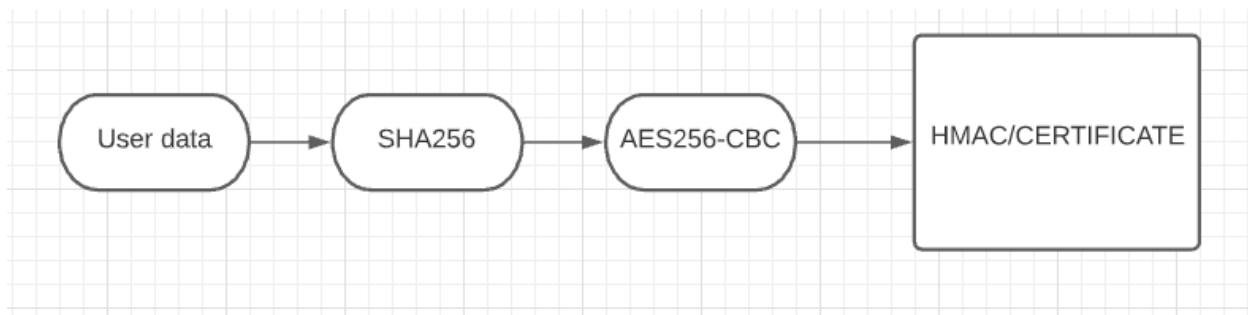


Figure 5: Certificate generation

## 7: Logging Layer

The logging layer maintains 10 files all of them compressed leaving out the current logging file. We build a wrapper called LOG that takes in multiple inputs to log to the files. The log will note down all actions being taken by the user or admins.

The log rotates after writing to the 10<sup>th</sup> file. Each file will be no larger than 50MB. All these files must be password protected that could be set by the admins.

## 8: Reminder:

This will be a background thread called bgReminder. The thread is scheduled to run at 7:00am each day and send email reminders to all users who have appointment that day.

## 9: Server time Check Layer

This will be a background thread running and checking every 30 mins if the time is correct or not on the server. It will use the NTP version 4 and sync time from a NIST time server.

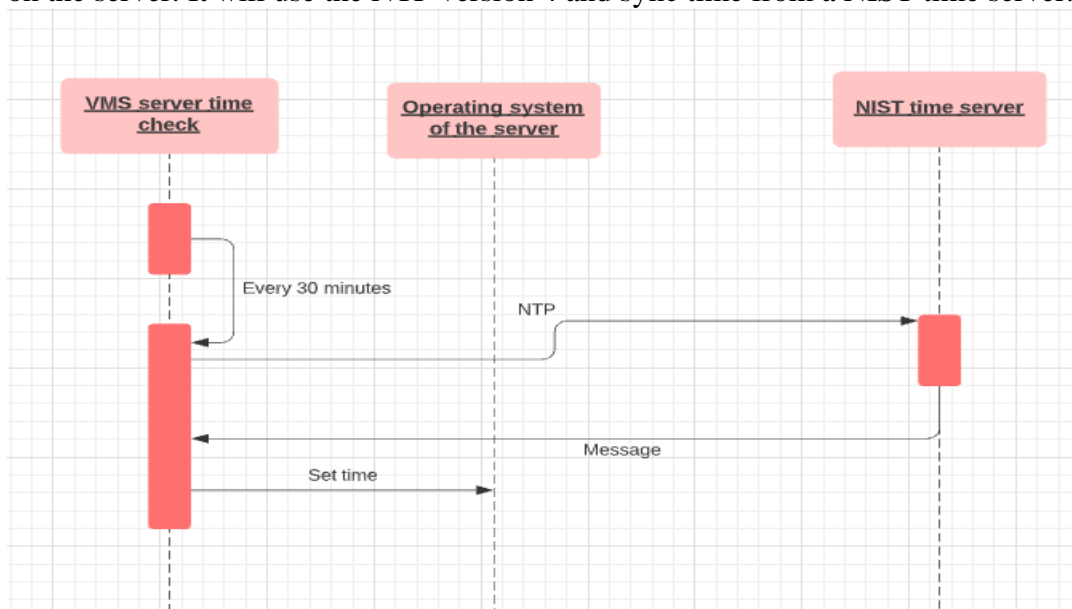


Figure 6: Sequence Diagram of time check layer

## 10: Feedback

Both admin and users interact with the feedback. Users add their feedback once they have attended the appointment. Admins can view the feedback. Searching or filtering through



- 1: isVerified: This is a flag that indicates that a user has verified their email.
- 2: zipTried: Number of the times the user tried to look through a zip code.
- 3: govId: A unique ID issued by the government.
- 4: bookings: These have the bookings that has been made by the user.
- 5: shots: These contain vaccine shots already received by the user.

#### Methods:

1: verifyLogin() : This method checks if the user has verified himself/herself and authenticates them.

2:searchZip() : This method takes a Zip Code input and shows the available vaccine centers for them.

3: makeAppointment(): Once the user selects an appointment this method is invoked and is responsible for performing the following operations:

- 1: Add the vaccine appointment to the users booking list.
- 2: Decrement the vaccine count at the chosen center.
- 3: save the booking date in the database.

4: generateCertificate(): When the user clicks on to download a certificate for their vaccine. The method gets invoked.

The method does the following:

- 1: Calculate the SHA256 hash of user's information : 1: Name 2: email 3: govId 4: vaccine name 5: vaccine center name.
- 2: Use AES256-CBC certificate key to sign this.
- 3: Create a pdf template and send the document to the user over https.

checkUsageLimit(): This method looks for if the user has attempted any of the following:

- 1: tried five different ZipCode for the same vaccine.
- 2: tried booking another appointment after already booking an appointment.

#### **Admin:**

##### Attributes:

1:campaignAdditionTried: Number of campaigns added in a day.

##### Methods:

addVaccineCampaign(): The method lets an admin add a new campaign.

addVaccineCenter(): The methods add a vaccine center to an existing campaign.

Following are the steps taken by it:

- 1: Check if the campaign exists, this should be provided as an argument.
- 2: If it does then create a vaccine center and add it to the center list.
- 3: Fill in the amount, vaccine name etc in the center.

isStillBanned(): Checks if an admin is still banned.

verifyLogin(): This method ensures password management such as when was the last password set etc. This also authenticates and authorizes the admin.

#### **Vaccine Campaign:**

##### Attributes:

- 1: name: This is the name of the campaign.
- 2: center: name of the vaccine centers under the campaign.

**Vaccine Center:**

Attributes:

- 1: name: Name of the vaccine center.
- 2: zipCode: The zipCode of the vaccine center.
- 3: stock: Types of vaccine available at the center.
- 4: amount: The number of corresponding vaccines available.

Methods:

isVaccineAvailable: The method is used to check if any stock is available at a given vaccine center.

**Vaccine:**

Attributes:

- 1: Name of the vaccine.

**Appointment:**

Attributes:

- 1: Name: The name of the user who booked the vaccine.
- 2: Email: email of the user who booked the vaccine.
- 3: zipCode : The zipCode of the center where vaccine was booked.
- 4: vax: The vaccine object for which booking was made.

Methods:

**System:**

System implements all the security requirements for VMS. Since all of the security requirements are asynchronous tasks running at a specific time or they run perpetually. The system will be a singleton class.

It contains the following methods:

- 1:startNTPCheck() : The method when called creates a thread that ensures the server time is correct using a NTP server.
- 2:startBanRemovalChecks() : The method creates a thread that checks if the ban period has been served by the client. If so, then remove the ban.
- 3:startActivityCleanUp(): The method invokes a thread that is responsible for resetting all usage limits once a given time period resets.
- 4:startReminder() : This method invokes a thread that sends reminders to the user for their appointments.
- 5:startAppointmentVerifications(): The method invokes a thread that sends the code to the user to check whether the appointment has been honored or not.
- 6:startKeyManagement(): The method invokes a thread that implements all the functionality of the key management layer as specified in the previous section.

## 4.2 Database Design

The plan is to use a non-relational database to store all the needs of our application.

We will use MongoDB as a database which is one of the most famous NoSQL databases.

MongoDB stores each table as BSON document and size of each of these documents is restricted to 16MB.

Following is the schema for the database:

User document:

```
{ _id : <objectId1>,  
  name : "xyz",  
  email : "xyz@gmail.com"  
}
```

auth document:

```
{ user_id : <objectId1>,  
  isBanned: false,  
  banTime: new Date()  
  isAuthroized: true  
}
```

Sec document

```
{ user_id: <objectId1>  
  Pass : "qwerty"  
  Salt : 4526718  
  PassSetDate: new Date()  
}
```

Appointment document

```
{ user_id:<objectId1>  
  name : "UMD Health Center"  
  zipCode : "27040"  
  dateBooking: new Date()  
  appointmentHonored: false  
}
```

feedback document

```
{ user_id:<objectId1>  
  feed : "Good campaign!"  
}
```

admin document

```
{_id : <objectId2>  
  email : "admin@xyz.com"  
}
```

asec document

```
{ admin_id: <objectId2>  
  Pass : "qwerty"  
  PassSetDate: new Date()
```

```
Salt : 4526718
}
```

```
aauth document:
{ user_id : <objectId2>,
  isBanned: false,
  banTime: new Date(),
  isAuthroized: true
}
```

```
vaccineCenter document
{ _id : <objectId3>,
  name: "UMD health center"
  zipCode: "20740"
}
```

```
vaccine document
{ _id : <objectId4>
  vaccineCenter_id : <objectId3>
  name: "AstraZeneca"
}
```

```
Certificate document
{user_id: <objectId1>
  vaccine_id : <objectId4>
  certHash: "8971923164127"
  signedhash: "asd8712t3419dhqwd101d"
}
```

## 5. GUI

### 5.1 Login page:



Welcome to VMS!

Username

Password

Sign In

Sign Up

## 5.2 Sign Up page:

Welcome to VMS!

Email

Password

Confirm Password

Government ID

Address

Sign Up!

### 5.3 User booking tab:

Booking

Certificate

feedback

Zip Code

### 5.4 User feedback tab:

Booking

Certificate

feedback

Covid

Good campaign!

Average campaign!

Bad Campaign!

Submit

General influenza

Good campaign!

Average campaign!

Bad Campaign!

Submit

### 5.5 User certificate download tab:



The interface for the 'Certificate' tab shows two sections for downloading certificates. The first section is for 'Covid' and the second is for 'General influenza'. Each section contains a 'Download' button.

Booking Certificate feedback

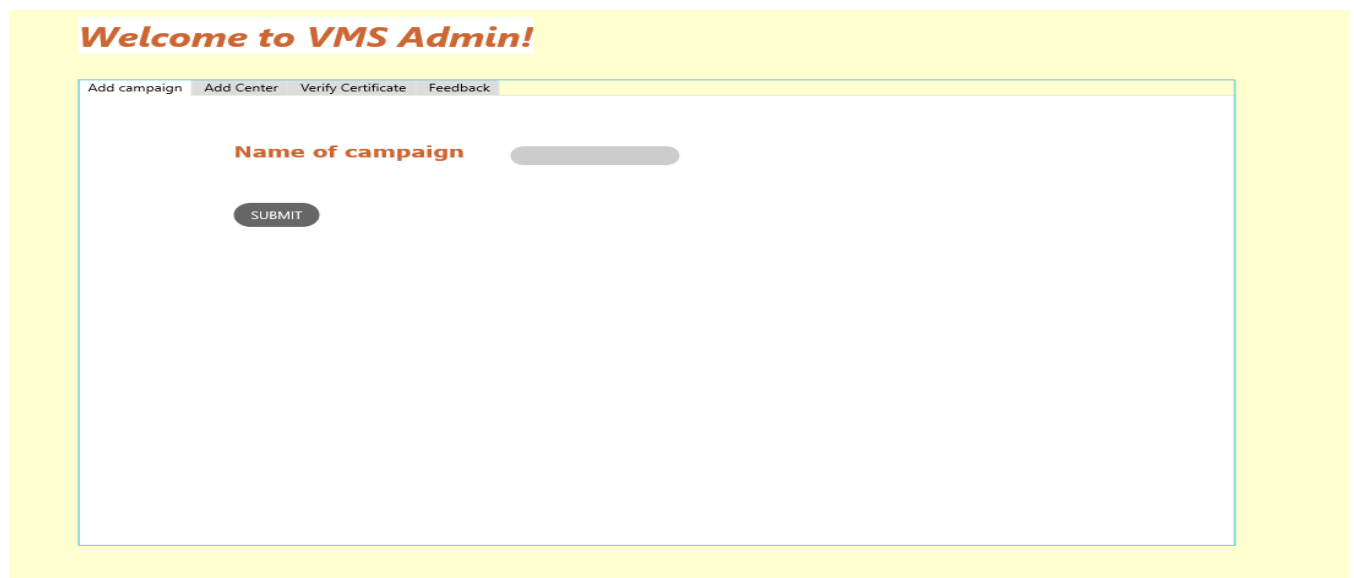
Covid

Download

General influenza

Download

### 5.6 Admin Homepage:



The Admin Homepage features a welcome message and a form to add a new campaign. The form includes a label 'Name of campaign', a text input field, and a 'SUBMIT' button. The navigation tabs are 'Add campaign', 'Add Center', 'Verify Certificate', and 'Feedback'.

**Welcome to VMS Admin!**

Add campaign Add Center Verify Certificate Feedback

Name of campaign

SUBMIT

### 5.7 Admin Add center tab:



## Welcome to VMS Admin!

Add campaign Add Center Verify Certificate Feedback

Name of Center

Vaccine Name

Quantity

Zip Code

SUBMIT

### 5.8 Admin Verification tab:

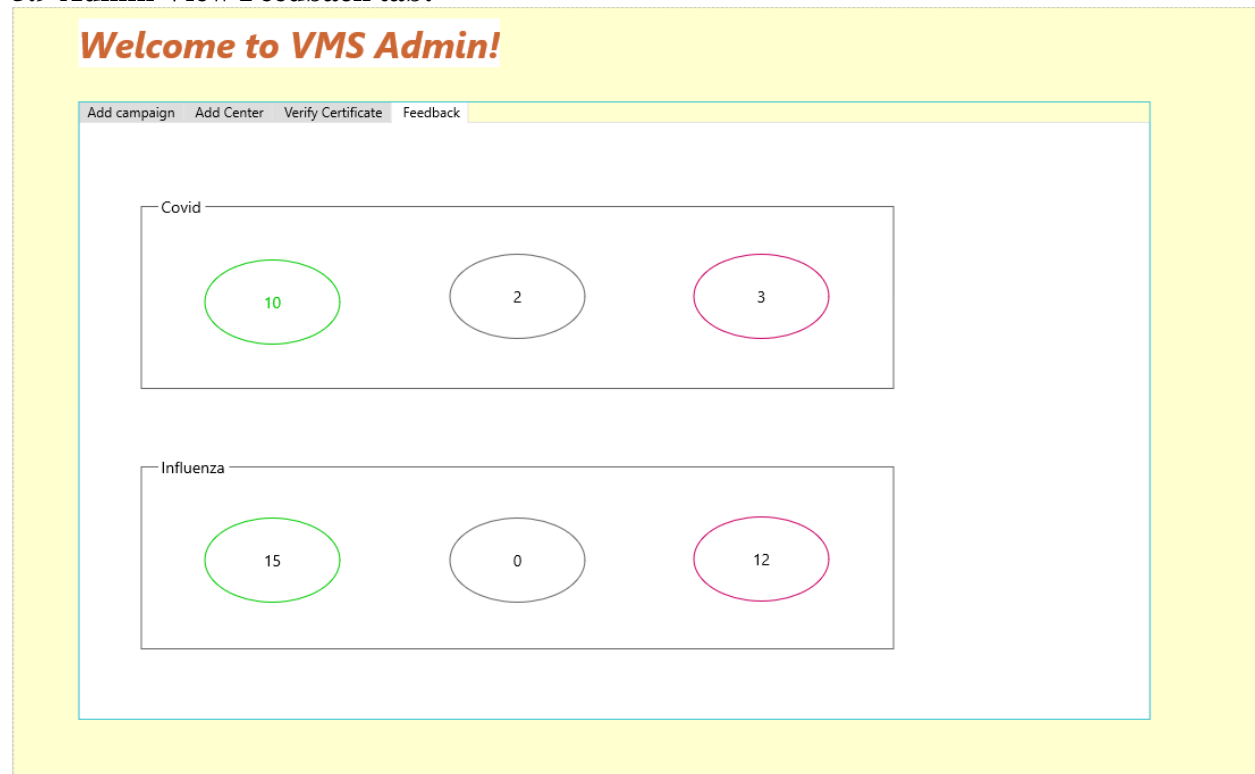
## Welcome to VMS Admin!

Add campaign Add Center Verify Certificate Feedback

Upload Certificate

Verify

## 5.9 Admin View Feedback tab:



## 6. Testing

The project takes an approach called Test Driven Development in which all the test cases are covered before development. This, section covers the unit test cases for each of the modules described in the system design.

Unit test cases for authentication:

### Signup

- 1: Leave one field blank and try signing up.
- 2: give an email id without the SMTP server name. Ex: Instead of [xyz@gmail.com](mailto:xyz@gmail.com) enter xyz.
- 3: Try signing up an already existing user with same information.
- 4: Try signing up an already existing user with the same email but different username or some other information.

### Login

- 1: Try to login a user after getting banned.
- 2: Try to login a user that has not signed up.
- 3: Try to login a user that has signed up but not verified their email yet.
- 4: Try to login a user after ban has been lift.
- 5: Try to login using correct 2FA code.
- 6: Try to login using wrong 2FA code.

- 7: Try with a blank code.
- 8: Try admin logging into user.
- 9: Try user logging into admin.

#### Book an appointment

- 1: Try booking an appointment again.
- 2: Try booking an again at a different zip code.

#### Test reminder

- 1: Check if reminder is received when booking is made.

#### Banning and removing ban

- 1: Attempt getting banned

#### Change time

- 1: Change time and see if the system corrects it.