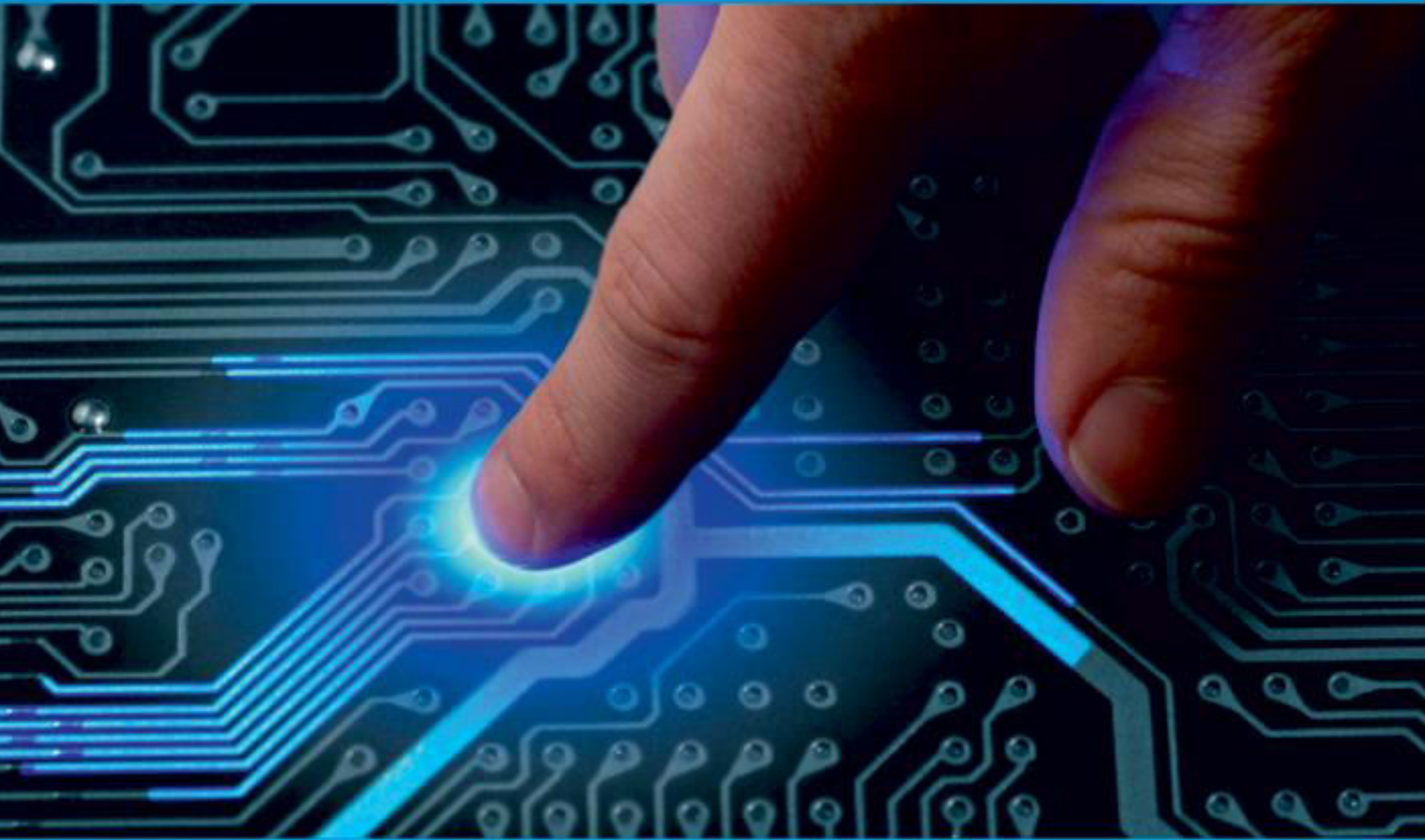
IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 10, Issue 5, May 2022

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.165

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 5, May 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1005273 |

Vaccine Management System Using Spring

Vamsi Chintala[1], Somarouthu Narasimha Rao[2], B. Avinash[3]

UG Student, IV Year Dept. of I.T., Vasireddy Venkatadri Institute of Technology, Guntur, India[1]

UG Student, IV Year Dept. of I.T., Vasireddy Venkatadri Institute of Technology, Guntur, India[2]

Assistant Professor, Department of Information Technology, Vasireddy Venkatadri Institute of Technology, Guntur, India[3]

ABSTRACT: All vaccination-related data is managed by the vaccine management system, which is a project. Users can make an appointment for vaccine doses using this system. The Vaccine Management System is a web-based application that primarily serves the general public. Users can use services like Register, Login, Appointment Booking, Display User Details, Check Vaccine Availability in respective Centers, Check Valid Dates for Booking Vaccination, the user will receive the remainder of the vaccine due date, and all vaccination statuses can be viewed at the same time. The Vaccine Management System is a self-contained application based on the Spring Framework and Microservices. Vaccine recipients can use the standalone application to register on the state or municipal vaccination website.As part of the registration process, prescreening checks will be conducted. Vaccine recipients will be able to use the app to arrange immunisation visits at a local institution. Vaccine recipients will receive a confirmation of their appointment, which they can produce at the clinic. The appointment service integrates with a provider's existing appointment scheduling system to reduce appointment overlap and give a positive immunisation recipient experience.

KEYWORDS: Mockito; Java; Spring MVC; HTML; CSS; Bootstrap; MySQL; Twilio.

I. INTRODUCTION

When the development of VMS was first contemplated, existing HSC systems such as Child Health and NIECR were taken into account. However, there were a number of technological issues, as well as the considerable design and development work that would have been required to ensure that existing systems could meet the requirements for booking, clinical process reporting, and analytics. The project's goal was to provide a fit-for-purpose digital Vaccine Management System (VMS) as quickly as possible so that Northern Ireland could return to its "new normal." The VMS was tasked with developing a digital vaccine appointment scheduling system for vaccinators, health and care providers, and then citizens of Northern Ireland. The system had to be capable of recording a pre-determined set of data points for reporting purposes, which would be subject to intense political and media scrutiny.Delivery of a digital vaccine appointment scheduling capability to assist the vaccinating of vaccinators, the wider NI health, social care, and allied clinical professionals across the public and private sectors, and the inhabitants of Northern Ireland in a variety of care settings (GP, Hospital, Community etc.).Mobilization, commissioning, and administration of a network of VMS provider expertise across Northern Ireland in order to develop the minimal viable product collectively (MVP). Collaboration with local delivery teams on VMS development, infrastructure, technology, and data needs.

II. LITERATURE SURVEY

Once vaccines and other commodities have been distributed from national shops, the Division of Vaccine and Immunization is having growing problems tracking their distribution. More issues are expected as new vaccinations are introduced, posing a major threat to Kenya's already overburdened vaccine supply chain. The ability of the existing supply and logistic system to distribute, track, and store the vaccines at suitable temperatures, condition, and quality, as well as ensure that they reach their destinations on time, will have been exceeded. This document serves as a system documentation for the Kenyan Division of Vaccine and Immunization's development of a Vaccine Stock Management Tool.The investigation, study, and review of current literature and systems architecture for vaccine stock supply management as well as the existing vaccine supply management system in use in Kenya were all part of the study. This exposed the Ministry of Health's Division of Vaccine and Immunization's present inadequacies in vaccine supply and stock management. The study's findings led to the creation and execution of a vaccination stock management tool that would address the country's regular stock outs by implementing a system based on a "push" or requisition model rather than the current "push" or allocation strategy. In order to accomplish this, an agile methodology was used to create a web-based vaccine stock management platform.The designed solution entails the development of a vaccine supply chain and cold chain management system that assures efficient and effective vaccine distribution while also allowing for improved and more prompt decision-making. The availability of high-quality, real-time data on vaccine supply and demand at the national, regional, and local levels would be crucial to improving vaccine management efficiency.

III. PROPOSED SYSTEM

This Vaccine Management System holds a huge quantity of vaccine data in the database, including user, admin profile, vaccine centres, and their corresponding counts. This system maintains user personal information such as passwords for both users and administrators, which are encrypted using an encryption technique. Aadhaar The user's number is also hidden for security reasons. By adding more features, the user can do a health checkup, which allows the user the chance to check for covid positive status while lowering lines of code and executing the application in less time. Admins can adjust the number of locations, see user profiles, and make changes to user data.

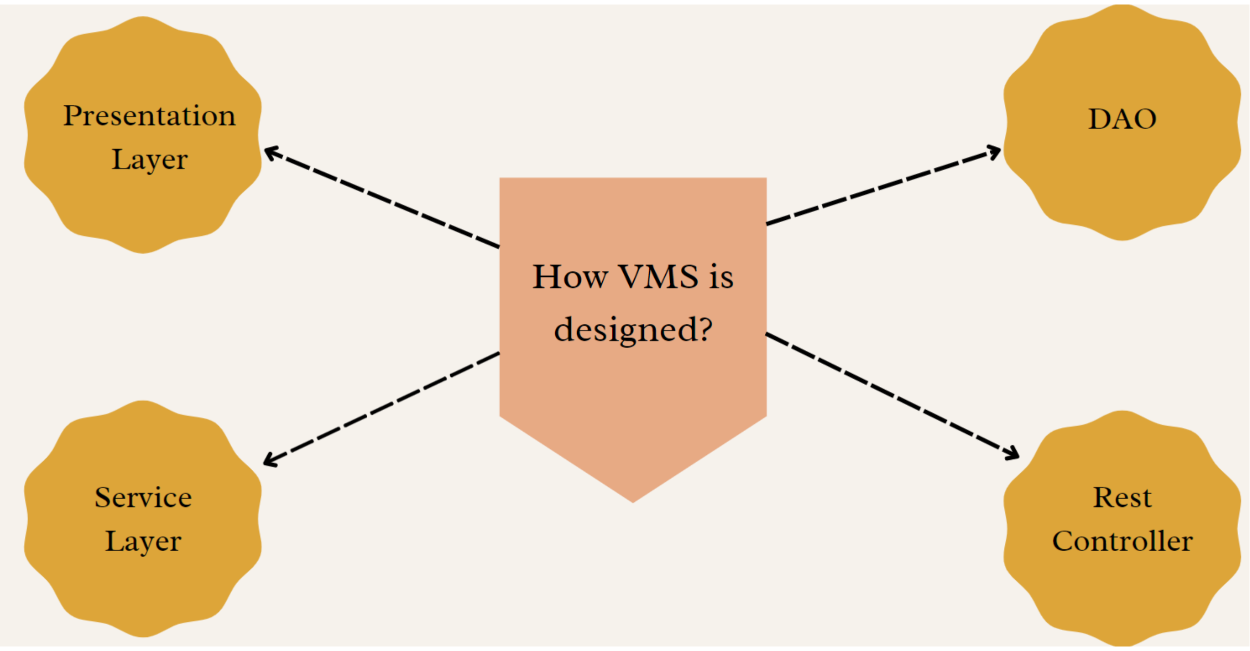


Figure 1: workflow diagram of proposed system

Four levels make up the Vaccine Management System:

Presentation Layer: Instead of just delivering data in the form of datagrams or packets between sites, application programmers consider data structure and presentation at the presentation layer. A presentation layer in Liquid Data is the software component responsible for the layout, formatting, and display of query results to users. You can use Liquid Data's JSP (Java Server Pages) tag library to generate the presentation layer in JSP pages.Java developers typically have two options when it comes to presentation layer technologies for n-tiered architectures: JSP (Java Server Pages) or Swing/AWT (Abstract Windowing Toolkit). JSP allows Java programmers to produce dynamic material that is very easy to distribute. In this scenario, the presentation layer is divided into two smaller layers: Files that deal with the view (JSP, Facelets, etc.). Servlets, @Controller from Spring MVC, @ManagedBean from JSF, and other files that regulate the interaction between the user and the view.

Service Layer: You can also have a business logic layer (often Service classes) and a data access layer in addition to these (DAOs or whatever you feel better to call them). If you think about it from the standpoint of designing graphical user interface desktop programs, your presentation will look like this: View controller classes are responsible for the user's interaction with the view.These classes are usually the same in this scenario, but keep in mind that they're just for show and should be related to your business logic layer. The service layer is responsible for implementing logic to process data provided to and from the DAO and the client. The service layer is made up of a set of Java classes that use one or more high-level methods to conduct business logic (data retrieval, updates, and deletions, for example). The service layer, in other terms, is in charge of workflow management.Domain logic is abstracted using the Service Layer.

It establishes a set of possible operations and coordinates the application's reaction to each operation through a layer of services that defines the application's boundary. The presentation layer is separated from the database layer by the service layer, which is the middle tier's entry point. Service layer classes expose methods that receive and/or return business objects in general. By calling down into business objects and the data access layer, the service layer wraps the business logic, making the code easier to maintain.In an application, a service layer supports communication between the controller and the persistence layer. The service layer also stores business logic. It has validation logic in it. To communicate between the controller and service levels, the model state is used. Treating validation as business logic has both advantages and cons, and Spring's validation (and data binding) architecture does not prevent either. Validation should not be restricted to the web layer, should be easy to localize, and should be able to use any validator accessible. Furthermore, client input data does not always flow through the REST controller process, and if we do not validate at the Service layer as well, undesirable data may pass through, resulting in a variety of difficulties.

Data Access Object Layer: Only the middle tier and the database communicate with the data access layer. The persistence technique used to persist business items is abstracted by classes in the DAL. ReIM can be linked to a different persistence engine thanks to the DAL. In certain circumstances, the update should ideally just require a change to the DAL. DAO is as light as feasible and exists merely to provide a connection to the database, which is occasionally abstracted to allow for the usage of several database backends. The service layer is responsible for implementing logic to process data provided to and from the DAO and the client. DAO is a class that performs CRUD activities such as save, update, and deletion.A DTO is simply a data-holding object. It's a JavaBean having setters and getters for instance variables. In a bean-like approach, the DTO is used to expose many values. The Data Access Object (DAO) pattern is a structural pattern that uses an abstract API to separate the application/business layer from the persistence layer (which is often a relational database but might be any other persistence mechanism). A data access object (DAO) is a pattern in software that creates an abstract interface to a database or other persistence mechanism. The DAO supports some specialized data operations without disclosing database information by mapping application calls to the persistence layer.The Data Access Object (DAO) pattern is a structural pattern that uses an abstract API to separate the application/business layer from the persistence layer (which is often a relational database but might be any other persistence mechanism). The main benefit of using data access objects is the relatively simple and strict separation between two important parts of an application that can but should not know about each other and are expected to evolve frequently and independently. Changes to business logic can use the same DAO interface, whereas changes to persistence logic have no impact on DAO clients as long as the interface is implemented appropriately.All storage information is hidden from the rest of the app. As a result, possible modifications to the persistence method can be made by just changing one DAO implementation without affecting the rest of the program. Between the application and the database, DAOs serve as a bridge. They transfer information between objects and database records. Unit Testing the code is made easier by replacing the DAO in the test with a test double, which makes the tests independent of the persistence layer. Data Access Objects as a design concept can be implemented in a variety of ways in the Java programming language.

Rest Controller Layer: The @RestController annotation is used to create restful web services with Rest Controller. This annotation is used at the class level, allowing the class to process client requests. Spring Boot is a microservice-based framework that requires relatively little time to turn into a production-ready application. In this post, we'll look at what a REST controller is and how to use it in Spring Boot. With the help of the @controller and @restcontroller annotations, there are primarily two controllers utilized in the spring: controller and Rest Controller. The key distinction between @restcontroller and @controller is that @restcontroller combines the @controller and @ResponseBody annotations.The @RestController annotation was added in Spring 4.0 to make developing RESTful web services easier. It's a handy annotation that combines @Controller and @ResponseBody, eliminating the need to use the @ResponseBody annotation on every request handling function in the controller class. The @Controller annotation is used to indicate that a class is a Spring MVC Controller, but the @RestController is a particular controller used in Restful web services that is the equivalent of @Controller + @ResponseBody. A convenient annotation for developing Restful controllers is @RestController. It's a subclass of @Component that's discovered automatically by searching the class path. It includes annotations for @Controller and @ResponseBody. The response is converted to JSON or XML.The @RestController annotation makes it easy to create Restful controllers. It's a specialization of @Component that's found automatically by scanning the class path. The annotations @Controller and @ResponseBody are added. It parses the response and converts it to JSON or XML. The methods cannot return ModelAndView because it does not work with the view technology. It's usually used in conjunction with @RequestMapping-annotated handler functions.

IV. Results & Discussion

Admin Dashboard: The admin is led to the admin Dashboard after successfully logging into the system, where he or she can perform a range of tasks, including updating the count at a given center for a specific vaccine, seeing all user profiles and their immunization status, and editing his or her own profile. If the login attempt fails, the administrator can create a new profile using the admin signup page.

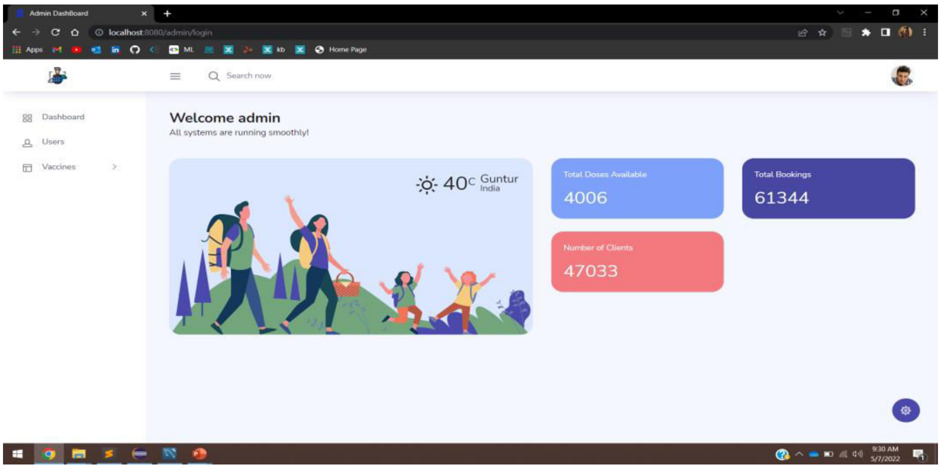


Figure 2: Admin Dashboard

User Dashboard: The user is sent to the user Dashboard after successfully logging into the system, where he or she can perform a range of actions, including booking doses, updating data related to user, visiting centres pages, and taking a free health checkup to determine the likelihood of being covid positive.If the user is unable to log in, they can create a new profile on the user signup page and then return to the login page.

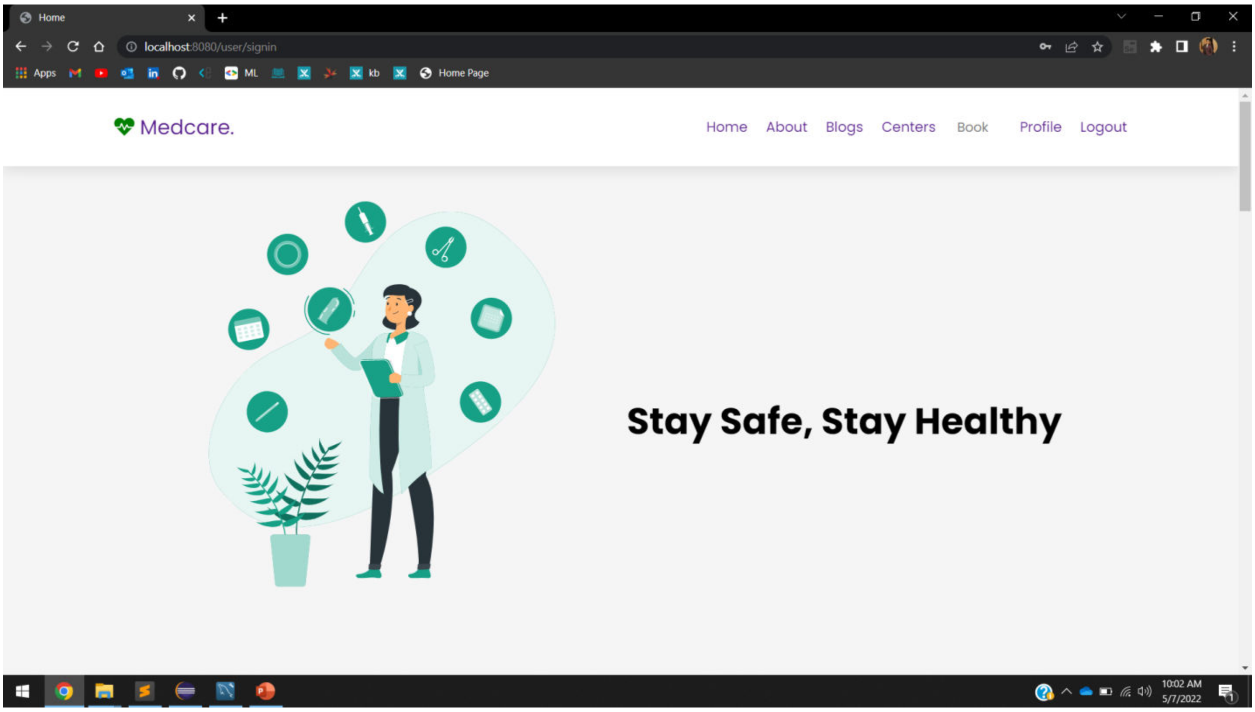
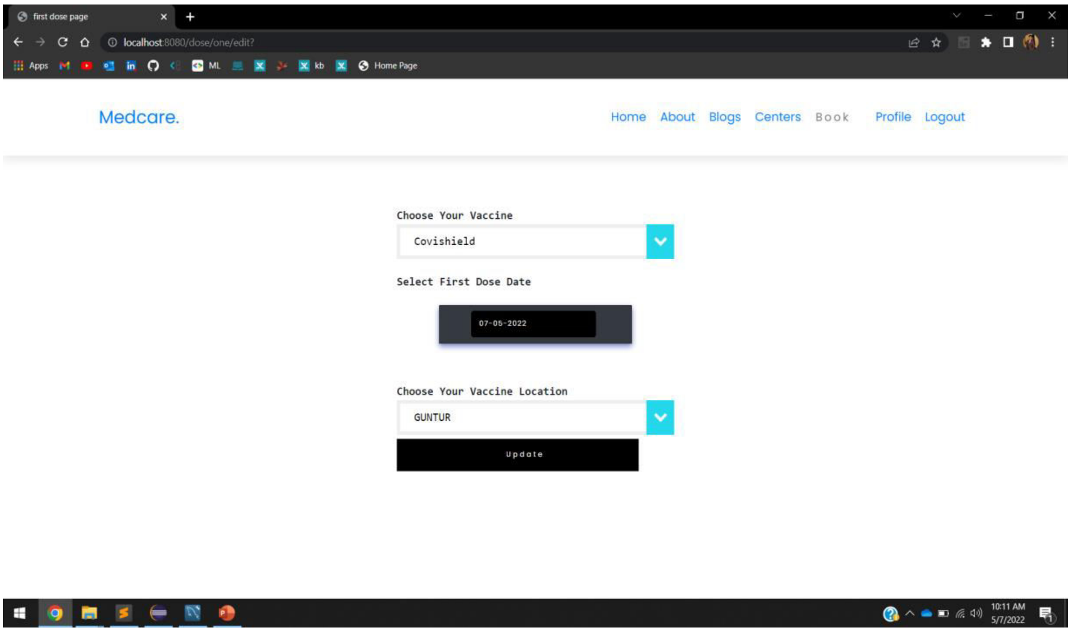


Figure 3: User Dashboard

Appointment Booking page: After logging in successfully, the user can plan a dose based on his availability after passing some validation tests. If the user has already taken dose 1, he is directed to the confirmation page, where he can update his current booking; if the user wants to take dose 2 without first taking dose 1, he is directed to dose 1; otherwise, the minimum gap between doses 1 and 2 must be greater than or equal to 45 days; otherwise, the user is not permitted to book a dose, and a confirmation is sent to the user's registered phone number.



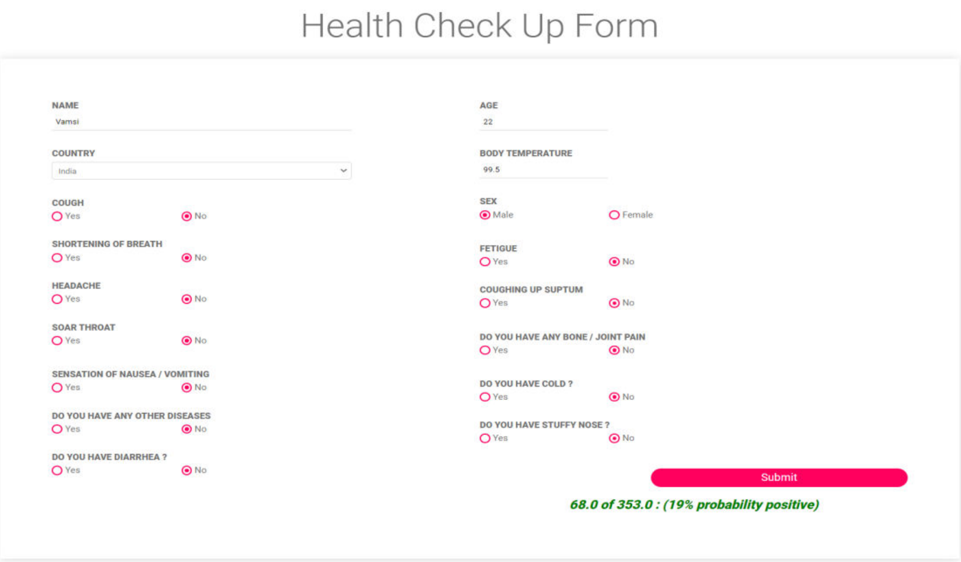
Choose Your Vaccine: Covishield

Select First Dose Date: 07-05-2022

Choose Your Vaccine Location: GUNTUR

Figure 4: Booking Page

Health Check Up Page: User can get a free health checkup by filling out a form based on the feedback received from the user, performing a few calculations, and receiving a result that determines the probability of receiving a positive result for covid.



NAME: Vamsi

COUNTRY: India

COUGH: ○ Yes ● No

SHORTENING OF BREATH: ○ Yes ● No

HEADACHE: ○ Yes ● No

SOAR THROAT: ○ Yes ● No

SENSATION OF NAUSEA / VOMITING: ○ Yes ● No

DO YOU HAVE ANY OTHER DISEASES: ○ Yes ● No

DO YOU HAVE DIARRHEA ?: ○ Yes ● No

AGE: 22

BODY TEMPERATURE: 99.5

SEX: ● Male ○ Female

FETIGUE: ○ Yes ● No

COUGHING UP SUPTUM: ○ Yes ● No

DO YOU HAVE ANY BONE / JOINT PAIN: ○ Yes ● No

DO YOU HAVE COLD ?: ○ Yes ● No

DO YOU HAVE STUFFY NOSE ?: ○ Yes ● No

68.0 of 353.0 : (19% probability positive)

V. CONCLUSION AND FUTURE WORK

A couple of improvements that can be made to the vaccine management system are that all passwords for users and administrators are encrypted using SHA-256, which is a weak algorithm, and an improvement that can be made is to use a bit more highly encrypted algorithm, such as PBKF2 (Password Based Key Definition Function), to provide more security. Twilio Sdk is used to send SMS to a user's registered mobile number. An improvement that can be made is to add an extra column to the registration page that accepts a unique user email id and sends an automatically triggered email to users about their vaccine status and confirmation messages using Spring JMS.

REFERENCES

1. Spring MVC: https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html.
2. Full stack Java development with spring MVC, hibernate, jQuery and bootstrap by Mayur Ramgir.
3. Mockito: https://javadoc.io/doc/org.mockito/mockito-core/latest/org/mockito/Mockito.html
4. Twilio: https://www.twilio.com/docs
5. Spring boot: https://docs.spring.io/spring-boot/docs/current/reference/pdf/spring-boot-reference.pdf
6. Spring boot in action by criag walls.

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details