**Software Design Document (SDD) for Vaccination Center Application**

**1. Introduction**

The Vaccination Center application is a software platform designed to streamline and manage vaccination appointments and processes.

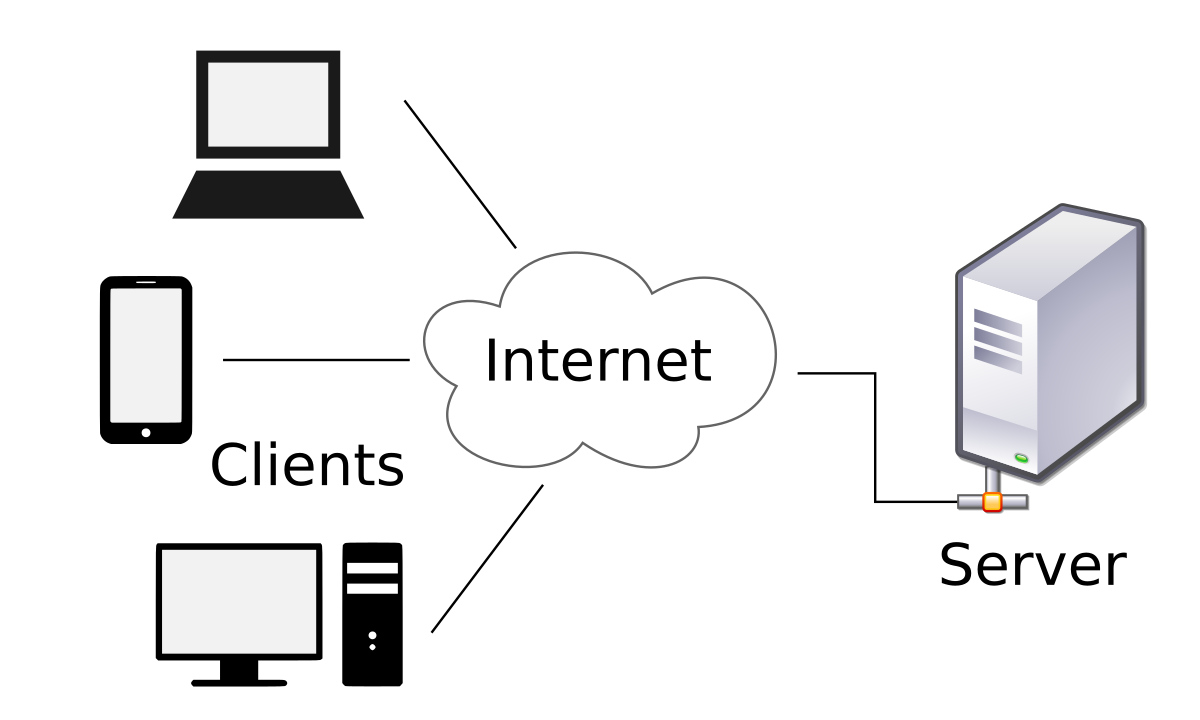
It provides a user-friendly interface for individuals to schedule their vaccinations and for vaccination center staff to manage appointments and vaccine administration.

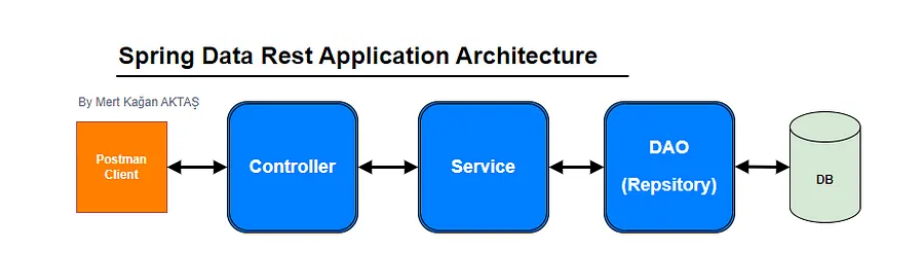
**Architectural Design**

The architecture of the Vaccination Center application follows **a client-server model**:

**Client Side**: The client-side application is a web-based user interface accessible through web browsers. It interacts with the server-side application via RESTful APIs.

**Server Side**: The server-side application is built using Spring Boot framework in Java. It includes components for handling user authentication, managing vaccination centers, scheduling appointments, and managing vaccines.



**Component Diagram**

**Description**:

**User Interface:**

Provides the user interface for interacting with the application. It consists of various UI components such as screens, forms, and buttons.

**Controller Layer:**

Handles incoming HTTP requests from the user interface and delegates them to the appropriate service layer based on the requested functionality.

**Service Layer:**

Implements business logic and orchestrates interactions between controllers and repositories. It contains the core logic of the application, including validation and authentication

**Repository Layer:**

Manages interactions with the database. It encapsulates database operations such as querying, inserting, updating, and deleting data.

**Database:**

Stores application data, including user information, vaccination center details, vaccine information, reservations, and certificates. It serves as the persistent storage for the application's data**.**

**Detailed Design**

**3.1 User Authentication**

The application supports authentication using username and password. Upon successful authentication, users are assigned roles (Admin, Patient, Vaccination Center) based on their credentials.

**3.2 Admin Module**

The Admin module allows administrators to manage vaccination centers and vaccines. It includes the following functionalities:

Create, read, update, and delete vaccination centers.

Manage vaccines for each vaccination center (create, read, update, delete).

**3.3 Patient Module**

The Patient module allows individuals to schedule vaccination appointments. It includes the following functionalities:

Register, login, and logout.

View list of vaccination centers and available vaccines.

Schedule first and second dose appointments.

View vaccination certificate after completing the vaccination process.

**3.4 Vaccination Center Module**

The Vaccination Center module allows staff at vaccination centers to manage appointments and vaccine administration. It includes the following functionalities:

View list of appointments scheduled at the center.

Approve or reject appointment requests.

Record vaccine administration and upload vaccination certificates.

**4. Data Design**

The application uses a relational database (**POSTGRES**) to store data.

**The main entities and their attributes include:**

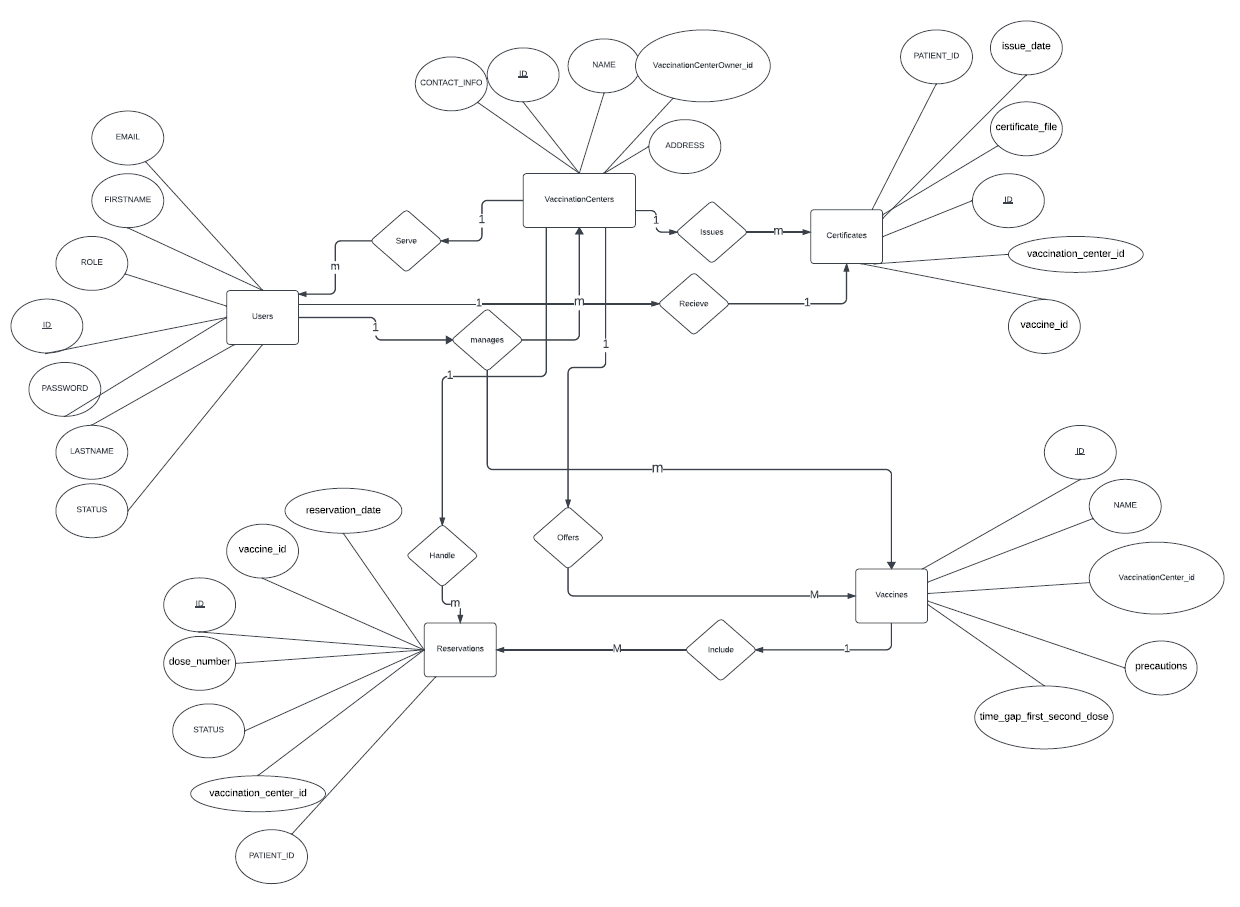
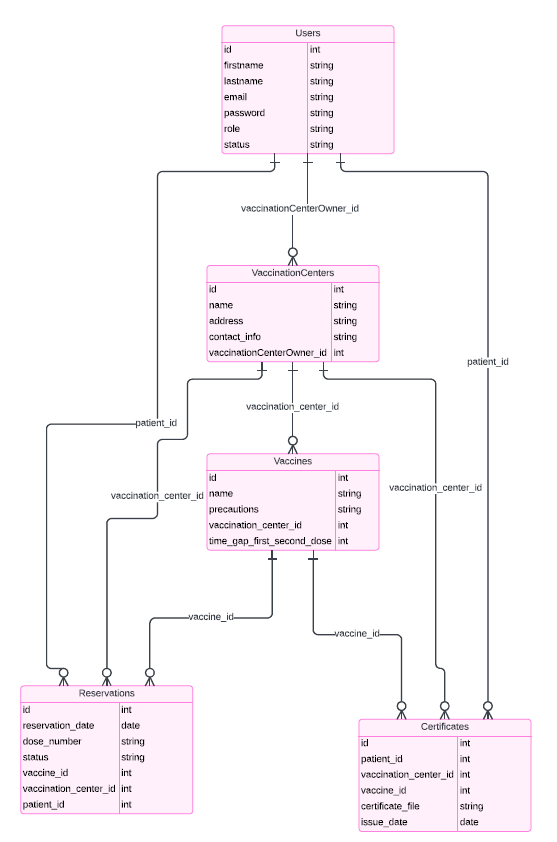
Users: id, username, password, role.

VaccinationCenters: id, name, address, contact\_info.

Vaccines: id, name, precautions, vaccination\_center\_id, time\_gap\_first\_second\_dose.

Reservations: id, patient\_id, vaccine\_id, dose\_number, reservation\_date, status.

Certificates: id, patient\_id, vaccination\_center\_id, vaccine\_id, certificate\_file, issue\_date.



**APIs explanation to understand how use it**

(for auth)

**http://localhost:8080/api/v1/auth/authenticate**

POST (login user , use it when user login in application)

**http://localhost:8080/api/v1/auth/register**

POST (register user (patient who use this end point))

**http://localhost:8080/api/v1/auth/{userId}/approve**

PUT (allow admin to edit patient statue to accepted to make patient can login)

(access user table)

**http://localhost:8080/api/User**

GET (get all users in all status & roles used by admin )

**http://localhost:8080/api/User/{userId}**

GET (get one user with specific id can be used by admin or vaccination center)

**http://localhost:8080/api/User/Patients**

GET (get users with statue Pending (users who waiting to accepted by admin)(patients))

(access vaccination center table)

**http://localhost:8080/api/VaccinationCenter**

GET (get all vaccination centers)

**http://localhost:8080/api/VaccinationCenter/{vaccinationCenterId}**

GET (get one vaccination center with specific id)

**http://localhost:8080/api/VaccinationCenter/{centerId}/Vaccines**

GET (get vaccines associated with a vaccination center)

**http://localhost:8080/api/VaccinationCenter/Owner/{OwnerId}**

Get (get a vaccination center associated with vaccination center owner)

**http://localhost:8080/api/VaccinationCenter**

POST (add new vaccination center)

**http://localhost:8080/api/VaccinationCenter/{vaccinationCenterId}**

DELETE (delete a specific vaccination center with id)

[**http://localhost:8080/api/VaccinationCenter/{vaccinationCenterId}**](http://localhost:8080/api/VaccinationCenter/%7bvaccinationCenterId%7d)

PUT (update a specific vaccination center with id)

(to access vaccines tables)

**http://localhost:8080/api/Vaccine**

GET (get all vaccines)

**http://localhost:8080/api/Vaccine/{vaccineId}**

GET (get one vaccine with specific id)

**http://localhost:8080/api/Vaccine**

POST (add new vaccine)

**http://localhost:8080/api/Vaccine/{vaccineId}**

DELETE (delete one vaccine with specific id)

**http://localhost:8080/api/Vaccine/{vaccineId}**

PUT (update one vaccine with specific id)

(to access reservation tables)

**http://localhost:8080/api/Reservation**

GET (get all reservations)

**http://localhost:8080/api/Reservation**

Post (make a new reservation reservations)

**http://localhost:8080/api/Reservation/{reservationId}/approveDose**

PUT (approve dose by vaccination center)

**http://localhost:8080/api/Reservation/pendingFirstDose/{vaccinationCenterId}**

GET (get first pending doses to view from vaccination center view to make it approve)

**http://localhost:8080/api/Reservation/pendingSecondDose/{vaccinationCenterId}**

GET (get second pending doses to view from vaccination center view to make it approve)

**http://localhost:8080/api/Reservation/{vaccinationCenterId}/patientsReserved**

GET (get patients who reserved)

(to access certificates tables)

**http://localhost:8080/Certificate/Upload**

POST (upload certificate by vaccination center to patient using vaccination center, vaccine and patient id)

**http://localhost:8080/Certificate/Upload/File**

POST (upload certificate by patient id)

**http://localhost:8080/Certificate/fileSystem/{fileName}**

GET (get photo by name )

[**http://localhost:8080/Certificate/fileSystem/ById/{patientId}**](http://localhost:8080/Certificate/fileSystem/ById/%7bpatientId%7d)

GET (get photo by patient id only )

**Aspect-Oriented Programming (AOP)**

**Overview**

Aspect-Oriented Programming (AOP) is a programming paradigm that enables modularization of cross-cutting concerns, such as logging, security, and transaction management, that cut across multiple modules or layers of an application. In the Vaccination Center application, AOP is utilized to address cross-cutting concerns and improve modularity and maintainability.

**Measuring Service Execution Time**

AOP is used to measure the execution time of services in the Vaccination Center application. This functionality is implemented using AOP aspects to intercept method calls to service layer components and calculate the execution time.

**Deployment Architecture**

**Docker Deployment**

The Vaccination Center application can be deployed using Docker containers to facilitate easy deployment, scalability, and portability across different environments.

**Docker Architecture Overview**

The deployment architecture using Docker consists of the following components:

* Docker Engine: The runtime environment for Docker containers.
* Docker Image: A lightweight, standalone, executable package that includes everything needed to run a piece of software, including the code, runtime, libraries, environment variables, and configuration files.
* Docker Container: An instance of a Docker image that runs as a process on the host machine's operating system.