Vaccination Center - Algorithm

- Define the necessary HTML files (login.html, register.html, errorpage.html) to provide the user interface for the login and registration functionalities.
- Implement the entity class User with properties id, username, email, and password.
- Create the repository class UserRepository to provide database access methods for the User entity.
- Implement the service class UserService to handle user-related operations, such as user registration.
- Create the controller class UserController to handle HTTP requests related to user login and registration.
- Implement the showLoginPage method in UserController to handle the GET request for displaying the login page (login.html).
- Implement the processLogin method in UserController to handle the POST request for processing the login form submission.
- Within this method:
    - Retrieve the submitted username and password from the request parameters.
    - Use the UserRepository to find a user with the provided username.
    - Check if the user exists and the password matches.
    - If the user is valid, redirect to the user dashboard or any desired page (e.g."redirect:/vaccine-centers").
    - If the user is invalid, redirect to the error page (errorpage.html).
  - Implement the showRegisterPage method in UserController to handle the GET request for displaying the registration page (register.html).
  - Implement the registerUser method in UserController to handle the POST request for processing the registration form submission. Within this method:
    - Retrieve the submitted username, email, and password from the request parameters.
    - Create a new User instance with the provided information.
    - Use the UserService to register the user by calling the registerUser method.
    - Return a response indicating a successful user registration. Vaccination Center:
- When a user accesses the main page ("/vaccine-centers"), the controller method getAllVaccineCenters is invoked.
- The method retrieves all vaccine centers using the vaccineCenterService and adds them to the model.
- The "vaccine-center-details" Thymeleaf template is rendered, displaying a table of vaccine center details fetched from the model.
- The user can click on the "Add Vaccination Center" button to navigate to the"/vaccinecenters/add" page.
- On the "/vaccine-centers/add" page, the user can enter the details of a new vaccine center and submit the form.
- The form data is sent to the server using a POST request, invoking the controller method addVaccineCenter.

- The method creates a new VaccineCenter object with the provided details and saves it to the database using the vaccineCenterService.
- After saving, the user is redirected back to the main page ("/vaccine-centers").
- On the main page, the user can view the details of a specific vaccine center by clicking the "View" button next to a center in the table.
- The controller method viewVaccineCenter is invoked, retrieving the details of the selected vaccine center and the associated citizens using the vaccineCenterService and citizenService.
- The details are added to the model, and the "vaccine-center-view" Thymeleaf template is rendered, displaying the vaccine center details and a table of associated citizens.
- The user can click the "Edit" button next to a vaccine center in the table to navigate to the "/vaccine-centers/{id}/edit" page. the edit page, the current details of the vaccine center are pre-filled in the form fields.
- The user can modify the details and submit the form.
- The form data is sent to the server using a POST request, invoking the controller method updateVaccineCenter.
- The method retrieves the existing vaccine center from the database using the vaccineCenterService, updates its details with the new values, and saves it back to the database.
- After updating, the user is redirected back to the main page ("/vaccine-centers").
- The user can also delete a vaccine center by clicking the "Delete" button next to a center in the table.
- The controller method deleteVaccineCenter is invoked, which retrieves the selected vaccine center using the vaccineCenterService and deletes it from the database.
- After deletion, the user is redirected back to the main page ("/vaccine-centers").
- HTML Templates:
    - "citizen-details.html": Displays a table of citizen details, including ID, name, city, number of doses, vaccination status, vaccination center, and actions.
    - "add-citizen.html": Provides a form to add a new citizen, including fields for name, city, number of doses, vaccination status, and vaccination center.
    - "view-citizen.html": Displays the details of a specific citizen, including name,city, number of doses, vaccination status, and vaccination center.
    - "edit-citizen.html": Provides a form to edit the details of a specific citizen,including fields for name, city, number of doses, vaccination status, and vaccination center.
- Java Classes:
    - "Citizen.java": Represents the Citizen entity with attributes such as ID, name,city, number of doses, vaccination status, and vaccination center. Includes getters, setters, and constructors.
    - "CitizenService.java" and "CitizenServiceImpl.java": Define the service interface and its implementation for managing citizen data. Includes methods for adding, updating, deleting, and retrieving citizens.
    - "VaccineCenter.java": Represents the VaccineCenter entity with attributes such as ID and center name. Used for associating a citizen with a vaccination center.
    - "CitizenController.java": Handles HTTP requests and defines the application's endpoints. Includes methods for displaying the list of citizens, adding a citizen, viewing a citizen's details, and editing a citizen's details.

- JavaScript: The JavaScript code in the "add.html" and "edit.html" templates dynamically updates the vaccination status field based on the selected number of doses using event listeners.