**Software Requirements Specification** (SRS)

**For :** VaxScheduler application

**1-introduction**

Project called "VaxScheduler" (Vaccination Scheduling Platform), is an online platform designed to streamline and simplify the process of scheduling and managing vaccination appointments. The platform aims to provide a user-friendly interface for individuals to schedule their vaccinations

**● Actors**

(3 types of users):

• Admin

• Patient

• Vaccination Center

-Now, we need to determine the functional and non-functional

Requirements of this program.

**Function requirements**

• Admin can add manage vaccination centers (Create, list, update, delete).

• Admin can manage vaccines (create, list, update, delete) for every vaccination center.

• Accept or reject registered users.

• Patient can list all vaccination centers and its vaccines.

• Vaccine includes: name, precautions, vaccination center, time gap between first and second dose.

• Patient can reserve vaccine (first and second dose). Patient can’t reserve second dose if the first dose is not accepted yet by the vaccination center.

Also, after accepting first dose, you need to choose a date after the gap period descripted in the vaccine.

• Patient can view certificate after accepting second dose from vaccination center.

• Vaccination center and list his registered users (who registered vaccines from this center).

• Vaccination center needs to approve the first and second reservation.

**1. Admin:**

• Login/ Logout

• Manage vaccination centers (CRUD)

• Manage vaccines (CRUD)

• Accept/Reject registered Users

**2. Patient:**

• Login/ Logout/ Register

• List vaccination centers and its vaccines

• Reserve vaccination (first and second dose)

• Prevent reserving second dose before accepting first dose from center

**3. Vaccination Center:**

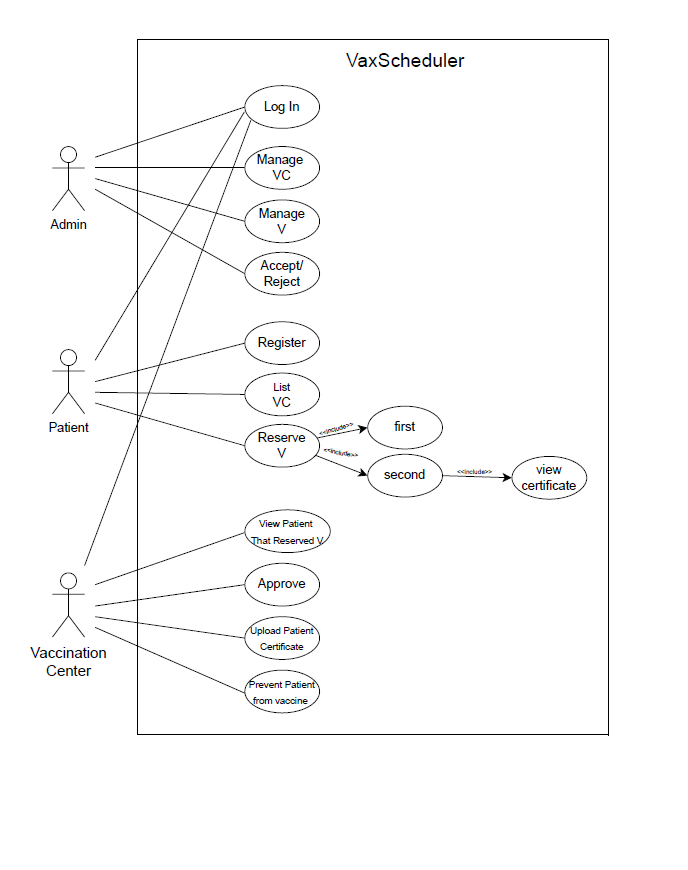
• Login/ Logout

• View patients that reserved vaccine from that vaccination center

• Approve (first and second dose)

• Upload patient certificate (after second dose)

• Prevent patient from reserving taken vaccine again

**Use case diagram**

|  |  |
| --- | --- |
| Use case name | Login |
| Number | 1 |
| Actor | Admin – patient – Vaccination Center |
| Precondition | When admin or patien or vaccinaton center need to login |
| Flow of event | They should to login  Enter UserName  Enter Password |
| Postcondition | You now can use website |

|  |  |
| --- | --- |
| Use case name | Manage VC |
| Number | 2 |
| Actor | Admin |
| Precondition | admin is logged in |
| Flow of event | 1. Admin navigates to the "Vaccination Centers" section. 2. Admin can create a new center by providing the necessary information (name, location, hours, etc.). 3. Admin can list all existing centers. 4. Admin can select a center to update or delete. 5. Changes are saved to the database. 6. Any invalid actions (e.g., deleting a center with active reservations) result in an error message. |
| Postcondition | Changes to vaccination centers are saved |

|  |  |
| --- | --- |
| Use case name | Manage V |
| Number | 3 |
| Actor | Admin |
| Precondition | Admin is logged in. |
| Flow of event | 1. Admin navigates to the "Vaccines" section. 2. Admin can create a new vaccine by providing necessary information (name, precautions, time gap between doses, etc.). 3. Admin can list all vaccines. 4. Admin can select a vaccine to update or delete. 5. Changes are saved to the database. 6. Errors are displayed for invalid operations (e.g., deleting a vaccine with active reservations). |
| Postcondition | Changes to vaccines are saved |

|  |  |
| --- | --- |
| Use case name | Accept/Reject |
| Number | 4 |
| Actor | Admin |
| Precondition | Admin is logged in |
| Flow of event | 1. Admin navigates to the "Registered Users" section. 2. Admin sees a list of pending user registrations. 3. Admin selects a user and chooses to accept or reject them. 4. The user is notified of the decision via email. 5. If accepted, the user gains access to the platform as a patient. If rejected, they are denied access. |
| Postcondition | User registration status is updated |

|  |  |
| --- | --- |
| Use case name | Register |
| Number | 5 |
| Actor | patient |
| Precondition | None. |
| Flow of event | 1. User navigates to the "Register" page. 2. User enters required information (name, email, password, etc.). 3. The system validates the information and checks for existing accounts. 4. If valid, the account is created, but requires admin approval. 5. The user is notified that their registration is pending approval. 6. If invalid, an error message is displayed. |
| Postcondition | Patient has a valid account. |

|  |  |
| --- | --- |
| List VC | List VC |
| Number | 6 |
| Actor | patient |
| Precondition | Patient is logged in |
| Flow of event | 1. Patient navigates to the "Vaccination Centers" page. 2. Patient can view a list of all centers along with the vaccines they offer. 3. Patient can click on a center to view more details, such as available appointment times. |
| Postcondition | none |

|  |  |
| --- | --- |
| Use case name | Reserve V |
| Number | 7 |
| Actor | patient |
| Precondition | Patient is logged in, Admin approved their registration |
| Flow of event | 1. Patient navigates to a specific vaccination center and chooses a vaccine to reserve. 2. Patient selects a date and time for the first dose. 3. The system checks if the selected appointment time is available. 4. If available, the reservation is made, pending approval by the vaccination center. 5. If the first dose reservation is accepted, the patient can later reserve the second dose after the required time gap. 6. If the first dose reservation is rejected, an error message is displayed. 7. Patient receives a notification once the vaccination center approves/rejects the reservation. |
| Postcondition | Reservation is made or rejected |

|  |  |
| --- | --- |
| Use case name | View Certificate |
| Number | 8 |
| Actor | patient |
| Precondition | Patient is logged in, and second dose is accepted |
| Flow of event | 1. Patient navigates to their dashboard and selects the "View Certificate" option. 2. The system checks if the second dose has been approved by the vaccination center. 3. If approved, the certificate is displayed, and the patient can download it. 4. If not approved, an error message is displayed indicating that the certificate is not yet available. |
| Postcondition | none |

|  |  |
| --- | --- |
| Use case name | View patient that Reserved V |
| Number | 9 |
| Actor | Vaccination Center |
| Precondition | Vaccination center personnel are logged in |
| Flow of event | 1. Vaccination center personnel navigate to the "Registered Patients" section. 2. They can view a list of all patients who have reserved vaccines at their center. 3. They can click on a specific patient to view reservation details, including dates and times. |
| Postcondition | None |

|  |  |
| --- | --- |
| Use case name | Approve |
| Number | 10 |
| Actor | Vaccination Center |
| Precondition | Vaccination center personnel are logged in |
| Flow of event | 1. Vaccination center personnel view the list of reservations for the first dose. 2. They can choose to approve or reject each reservation. 3. If approved, the patient is notified, and the reservation is finalized. 4. If rejected, the patient is notified with an explanation. 5. For the second dose, the system checks the time gap between the first and second dose before allowing approval. 6. If the second dose is approved, the patient is notified. If rejected, an error message is displayed indicating why |
| Postcondition | Reservation is approved or rejected. |

|  |  |
| --- | --- |
| Use case name | Upload Patient Certificate |
| Number | 11 |
| Actor | Vaccination Center |
| Precondition | Second dose has been approved |
| Flow of event | 1. Vaccination center personnel navigate to the patient's reservation for the second dose. 2. They can upload the patient's certificate after the second dose is administered. 3. The system verifies the uploaded file format and size. 4. If valid, the certificate is stored in the database and accessible to the patient. 5. If invalid, an error message is displayed, and the upload is rejected. |
| Postcondition | Certificate is uploaded and accessible to the patient |

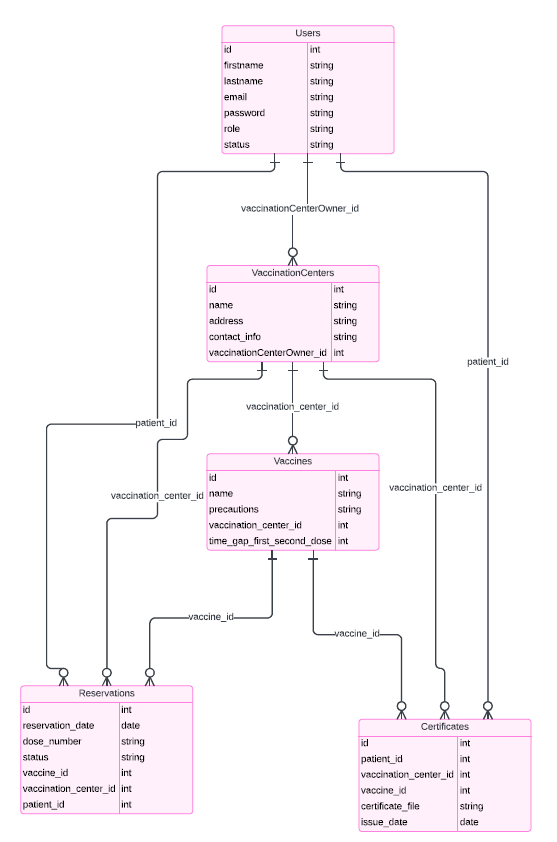
|  |  |
| --- | --- |
| Use case name | Prevent Patient from vaccine |
| Number | 12 |
| Actor | Vaccination Center |
| Precondition | Vaccination center personnel are logged in |
| Flow of event | 1. Vaccination center personnel check a patient's reservation history. 2. If the patient has already received the vaccine, they cannot reserve it again. 3. If the system detects an attempt at double booking, an error message is displayed, and the reservation is prevented. |
| Postcondition | Double booking is prevented |

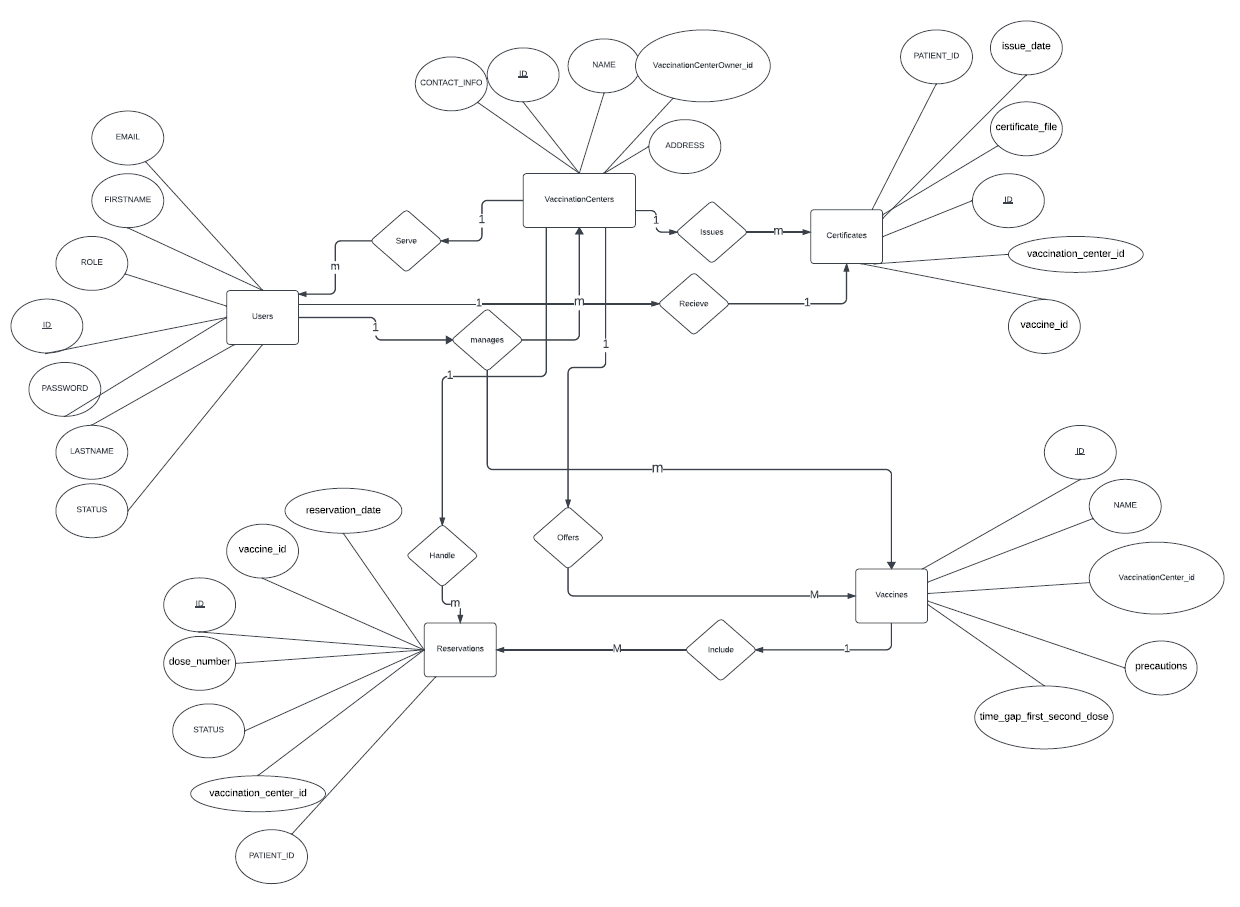
**Activity Diagram**

**A diagram of a company

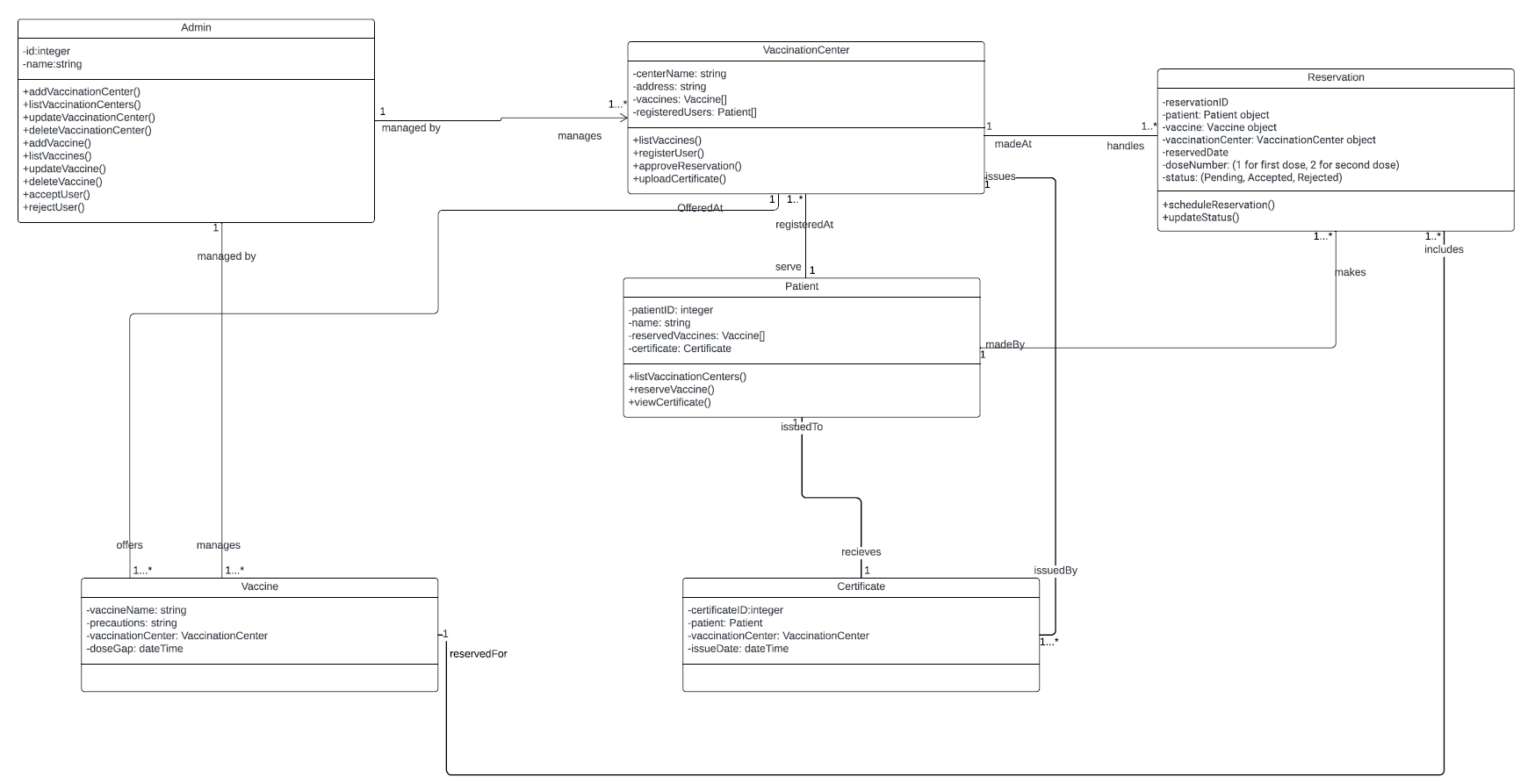
Description automatically generated**

**Database tables**

****

**ERD**

**Class diagram**

****

**OCL**

**Admin Class:**

***Invariants:***

context Admin

inv: self.manages->notEmpty()

inv: self.manages->forAll(vc | vc.managedBy = self

inv: self.managesVaccines->notEmpty()

inv: self.managesVaccines->forAll(v | v.managedByAdmin = self)

***Pre &Post Conditions:***

context Admin::deleteVaccinationCenter(vc: VaccinationCenter)

pre: self.manages->includes(vc)

post: not self.manages->includes(vc)

context Admin::addVaccine(v: Vaccine)

pre: v.name ->notEmpty()

post: self.managesVaccines->includes(v)

context Admin::addVaccinationCenter(newCenterName: String, newLocation: String)

pre: newCenterName ->notEmpty()and newLocation ->notEmpty()

post: self.manages->exists(vc | vc.centerName = newCenterName)

context Admin::listVaccinationCenters(): Set(VaccinationCenter)

pre: self.manages->notEmpty()

context Admin::updateVaccinationCenter(existingCenter: VaccinationCenter, newCenterName: String, newLocation: String)

pre: self.manages->includes(existingCenter) and newCenterName ->notEmpty() and newLocation ->notEmpty()

post: existingCenter.centerName = newCenterName and existingCenter.location = newLocation

context Admin::deleteVaccinationCenter(vc: VaccinationCenter)

pre: self.manages->includes(vc)

post: not self.manages->includes(vc)

context Admin::addVaccine(newVaccineName: String, newPrecautions: String, newTimeGap: Integer)

pre: newVaccineName ->notEmpty() and newPrecautions ->notEmpty() and newTimeGap > 0

post: self.managesVaccines->exists(v | v.name = newVaccineName

context Admin::listVaccines(): Set(Vaccine)

pre: self.managesVaccines->notEmpty()

context Admin::updateVaccine(existingVaccine: Vaccine, newVaccineName: String, newPrecautions: String, newTimeGap: Integer)

pre: self.managesVaccines->includes(existingVaccine) and newVaccineName ->notEmpty() and newPrecautions ->notEmpty() and newTimeGap > 0

post: existingVaccine.name = newVaccineName and existingVaccine.precautions = newPrecautions and existingVaccine.timeGap = newTimeGap

context Admin::deleteVaccine(v: Vaccine)

pre: self.managesVaccines->includes(v)

post: not self.managesVaccines->includes(v)

context Admin::acceptUser(patient: Patient)

pre: patient.name ->notEmpty()

post: No specific postcondition

context Admin::rejectUser(patient: Patient)

pre: patient.name ->notEmpty()

post: No specific postcondition

***Collection Operations:***

context Admin

def: listVaccinationCenters(): Set(VaccinationCenter) =self.manages

context Admin

def: listVaccines(): Set(Vaccine) =self.managesVaccines.vaccine

**VaccinationCenter Class:**

***Invariants:***

context VaccinationCenter

inv: self.centerName ->notEmpty()

inv: self.location ->notEmpty()

inv: self.offers->notEmpty()

inv: self.offers->forAll(v | v.offeredAt = self)

***Pre &Post Conditions:***

context VaccinationCenter::registerUser(patient: Patient)

pre: patient.name ->notEmpty()

post: self.registeredAt->includes(patient)

context VaccinationCenter::approveReservation(reservation: Reservation)

pre: reservation.status = 'pending'

post: reservation.status = 'approved'

context VaccinationCenter::uploadCertificate(patient: Patient, certificateID: String, issueDate: Date)

pre: patient.name -> notEmpty() and certificateID ->notEmpty() and issueDate <> null

post: No specific postcondition

context VaccinationCenter::listVaccines(): Set(Vaccine)

pre: self.offers->notEmpty()

***Collection Operations:***

context VaccinationCenter

def: listVaccines(): Set(Vaccine) =self.offers

**Vaccine Class**

***Invariants:***

context Vaccine

inv: self.name ->notEmpty()

inv: self.precautions ->notEmpty()

inv: self.timeGap > 0

**Patient Class**

***Invariants:***

context Patient

inv: self.patientID->notEmpty()

inv: self.name->notEmpty()

***Pre &Post Conditions:***

context Patient::reserveVaccine(v: Vaccine)

pre: v.offeredAt->notEmpty()

post: self.madeBy->exists(r | r.reservedFor = v)

context Patient::viewCertificate(certificate: Certificate)

pre: self.receives->includes(certificate)

post: No postconditions for viewing certificate

***Collection Operations:***

context Patient

def: listRegisteredVaccinationCenters(): Set(VaccinationCenter) =self.registeredAt

**Reservation Class:**

***Invariants:***

context Reservation

inv: self.reservationID->notEmpty()

self.reservedDate->notEmpty()

self.doseNumber->notEmpty()

self.status->notEmpty()

inv: self.madeAt->size() = 1

inv: self.includes->size() >= 1

inv: self.includes->forAll(vaccine | vaccine.reservedFor = self)

***Pre &Post Conditions:***

context Reservation::scheduleReservation()

pre: self.madeAt->notEmpty()

post: No postconditions for scheduling reservation

context Reservation::updateStatus(newStatus: String)

pre: newStatus = 'approved' or newStatus = 'rejected'

post: self.status = newStatus

**Certificate Class :**

***Invariants:*** context Certificate

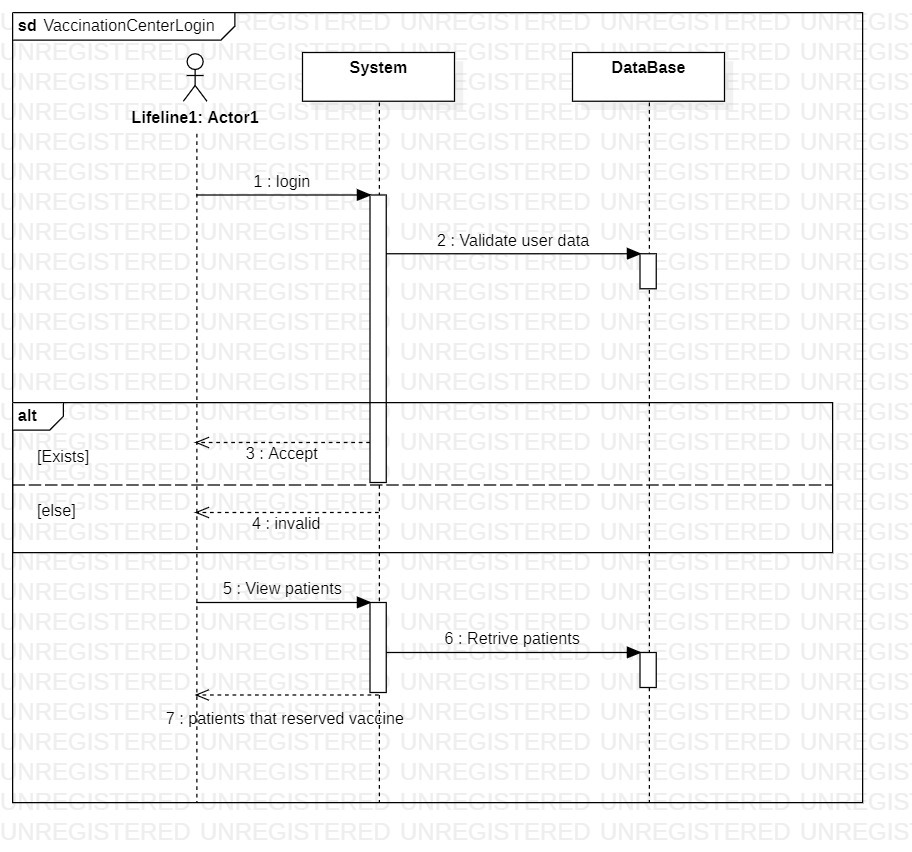
inv: self.certificateID->notEmpty()

self.issueDate->notEmpty()

inv: self.issuedBy->size() = 1

inv: self.issuedTo->size() = 1

**Sequence diagrams**

****