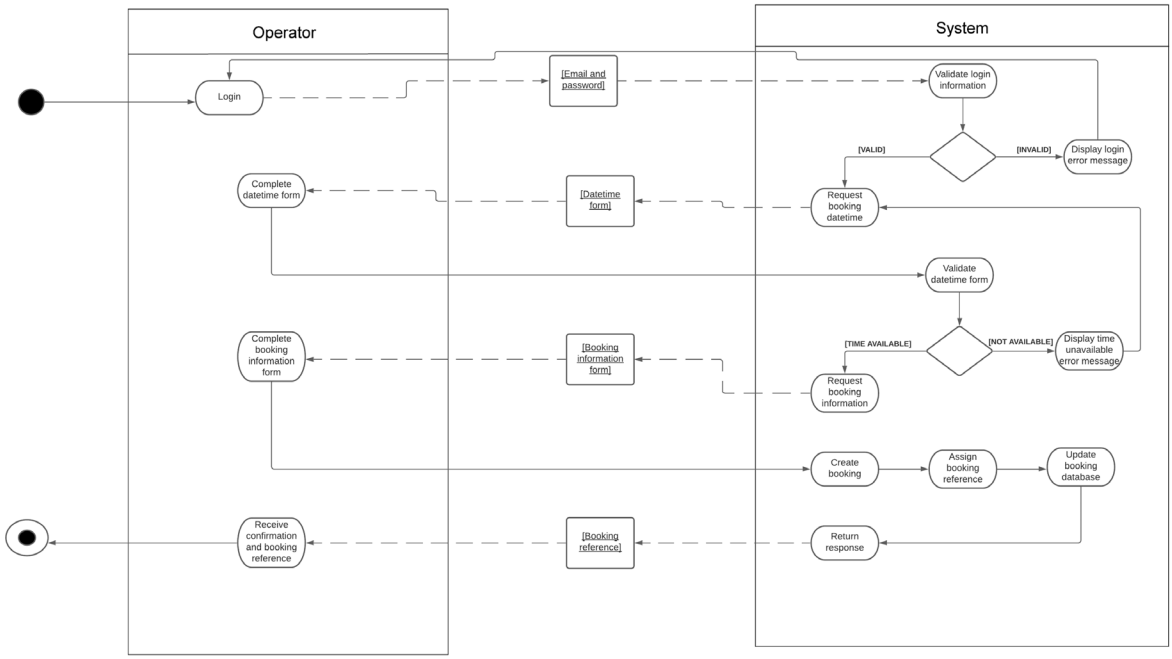
COMP6011 – Crownpass Vaccinator Subsystem D

1b) Activity Diagram

Vaccination operation staff – Add vaccination booking



Activity list

Use case name: Add vaccination booking

Participating actor: Vaccination operation staff (referred to as operator)

Entry conditions:

1) The operator has a registered account in the system
2) The operator has received a booking request

Exit conditions:

1) The booking is added to the system
2) The operator is given a booking reference with confirmation

Flow of events:

1) The operator logs into the system using their email and password.
2) The system validates the email and password
3) The system requests the booking date and time
4) The operator completes the form
5) The system validates if the requested slot is available

6) If the slot is available, request booking information

7) The operator completes the booking data form

8) The system creates the booking

9) The system assigns a booking reference number

10) The system updates the booking database

11) The system returns confirmation and the booking reference to the operator

Exceptional conditions and alternative flow of events:

There is an error in the email and password entered:

2.1) Display login information error message, return to 1)

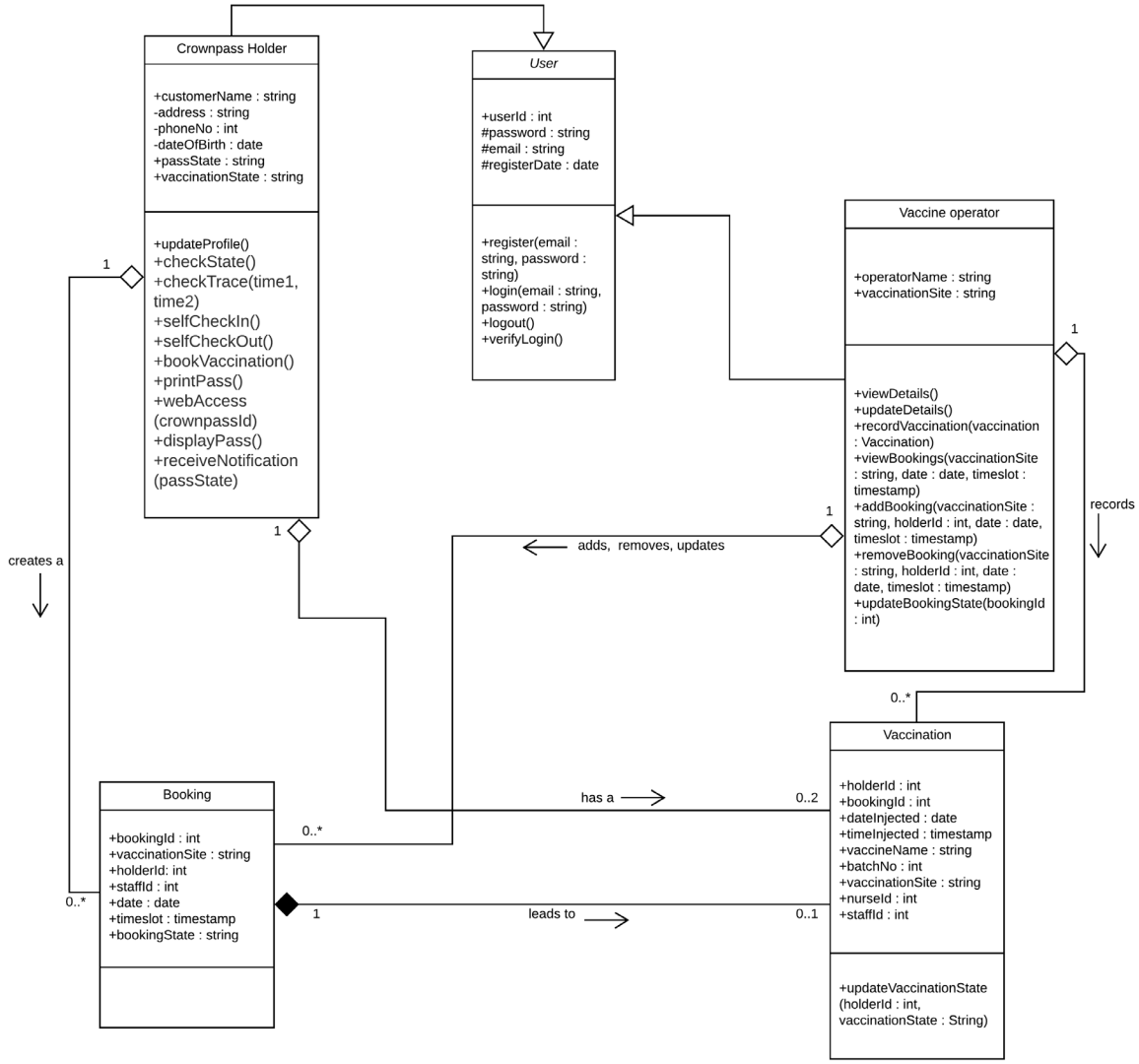
Requested datetime is not available:

6.1) Display slot unavailable error message, return to 3)

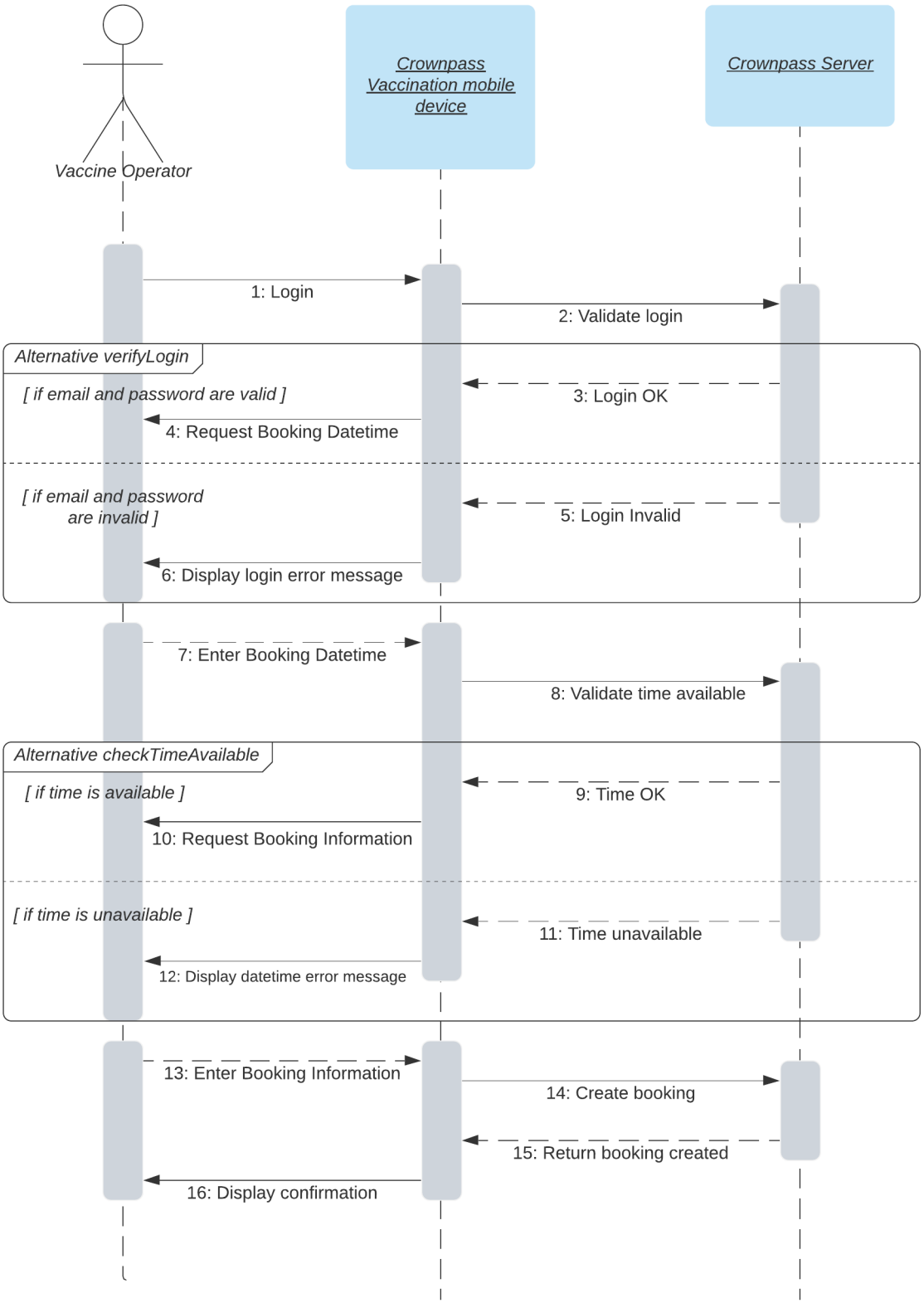
Special requirements:

1) The response time for each operation in the process of entering vaccination data should be no more than 3 seconds

1c) Class Diagram

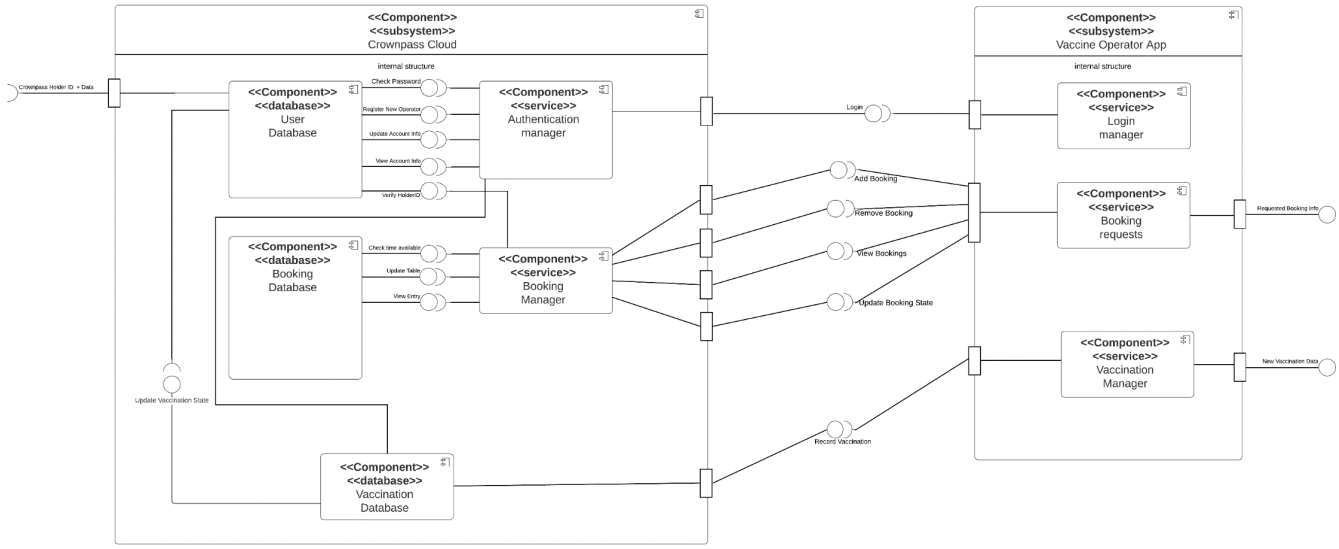


1d) Sequence Diagram



2a & 2b) Component Diagram

First iteration of the internal structure showing components & connectors within the app and the cloud of subsystem D:



<u>Subsystem D – Textual documentation of Components & Connectors</u>

Vaccine operator = Operation staff of vaccination stations/centres

Service Name	**Service Provided**
Login Manager	A microservice that allows users of the app to send requests relating to the vaccine operator accounts to the cloud
Authentication Manager	A microservice running on the cloud to process a vaccine operators requests related to managing (or creating) their user account by interacting with the User Database.
Booking Requests	A microservice that relates to queries a vaccine operator will have regarding bookings and allows an operator with some information regarding a booking to send a request to the cloud.
Booking Manager	A microservice running on the cloud that will query the Booking database depending on what request it has received. It is capable of verifying a booking can be created and also provides access to all bookings in the database.
Vaccination Manager	A microservice that allows a vaccine operator to send recorded vaccine information to the cloud which in turn allows for vaccination states to be updated.

Database Name	**Database Description**
User Database	A database that stores information on all registered users in the crownpass system, including emails, passwords, vaccination states, account roles and unique crownpass ID’s. For

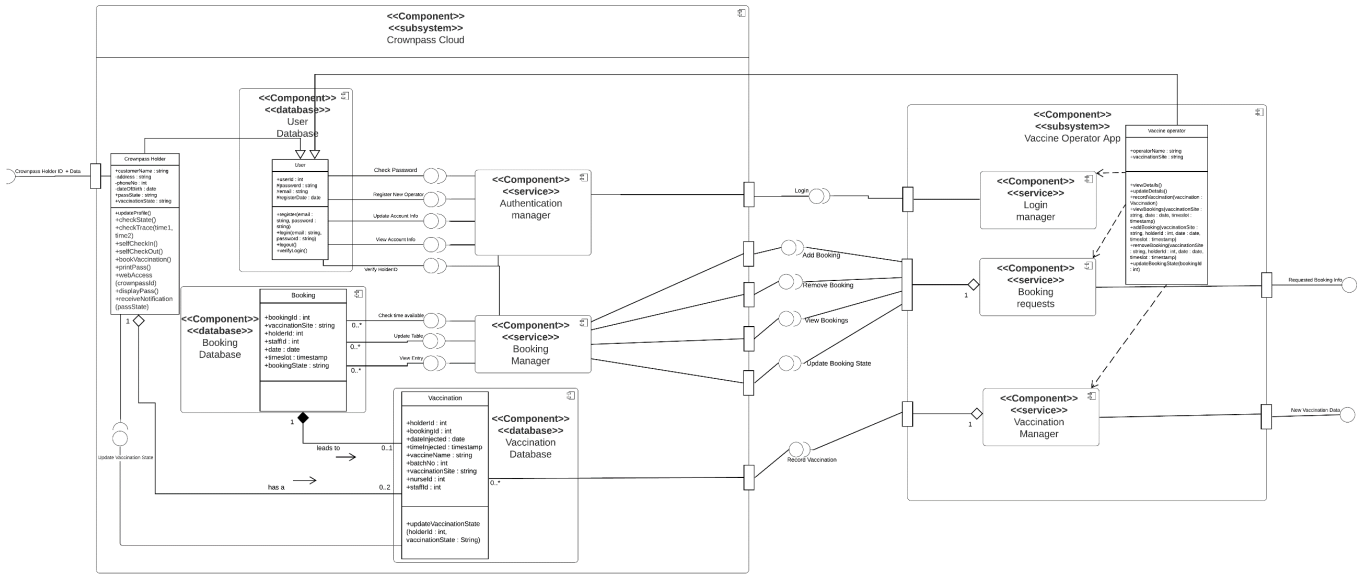
	my subsystem this is concentrated on crownpass holders and vaccine operators. Through queries with the authentication manager it can verify the correct password has been entered when logging in, update account information upon request from the mobile app and verify a Crownpass HolderID exists and is correct for the booking manager.
Booking Database	The booking database contains all information related to bookings. This includes the vaccination site associated, the date and time a booking is scheduled to take place and whether they are complete or not.
Vaccination Database	The vaccination database contains all information recorded in each unique vaccination. This includes the holderID associated, the bookingID, the date and time of injection, the vaccine name and batch number, the vaccination site and the ID of the Vaccine Operator that logged it. The vaccination state is stored on the cloud and the state transition is triggered whenever a new entry is recorded in the database by looking at the HolderID associated with it.

Connector Name	**Connector Description**
Login	Provided by the Authentication Manager on the cloud and required by the Login Manager on the app. This connector associates account information entered with the login manager with a request to the cloud to login with a password, or view / update data associated with the account.
Check Password	Provided by User Database and required by Authentication Manager. Verifies the password entered during login matches the one stored in the database (after hashing).
Register New Operator	Provided by User Database and required by Authentication Manager. Adds a new vaccine operator the database with a email and password, and assigns them a ID.
Update Account Info	Provided by User Database and required by Authentication Manager. Allows a vaccine operator to update their account information stored on the cloud.
View Account Info	Provided by User Database and required by Authentication Manager. Allows a registered vaccine operator to view their information stored on the cloud.

Verify HolderID	Provided by User Database and required by Authentication Manager, Booking Manager & Vaccination Database. Verifies that a HolderID does exist in the system and gives that holders information. Useful for booking manager such that they can verify when receiving booking data for an add request that the holderID does exist, similarly for Vaccination DB ensuring the holderID exists but also ensuring the updated vaccination state is associated with the correct Crownpass Holder.
Crownpass Holder ID + Data	Required by User Database. From subsystem A crownpass holder information regarding new crownpass accounts & current test states.
Requested Booking Info	Required by Booking Requests. Contains all booking information associated with a unique request, which could be some of a holderID, a date time or a vaccination site.
Add Booking	Provided by Booking Manager and required by Booking Requests. When a vaccine operator is asked to add a booking to the system they send through the requested time slot and holderID associated with the booking.
Remove Booking	Provided by Booking Manager and required by Booking Requests. Allows vaccine operators to remove a booking from the system by searching via a HolderID or datetime.
View Bookings	Provided by Booking Manager and required by Booking Requests. Allows vaccine operators to view all bookings at a vaccination site or on a specific date / time.
Update Booking State	Provided by Booking Manager and required by Booking Requests. Allows vaccine operators to update a booking state to completed when the associated vaccination has been recorded.
Check time available	Provided by Booking Database and required by Booking manager. Verifies that a request timeslot at a given vaccination centre is indeed available for a requested booking.
Update Table	Provided by Booking Database and required by Booking manager. Allows for the updating of bookings via add, remove or update booking state requests.
View entry	Provided by Booking Database and required by Booking manager. Allows a booking manager to return all booking entries given some criteria received from the app.
New Vaccination Data	Provided by Vaccination Manager. Includes all information when each vaccination is performed.

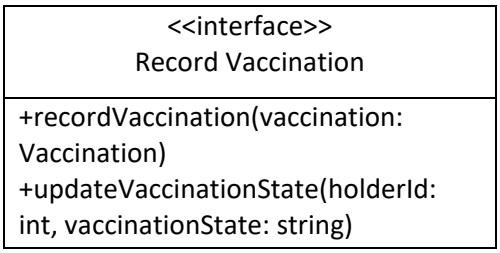
Record Vaccination	Provided by Vaccination Database and required by Vaccination Manager. Allows the vaccination manager to take the performed vaccination information and record it and send it to the cloud.
Update Vaccination State	Provided by Vaccination Database and required by User Database. Triggers the state transition of vaccination state associated with a HolderID when a vaccination is recorded concerning them.

Internal structure with classes:



3a) Subsystem D – Unit Test Plan

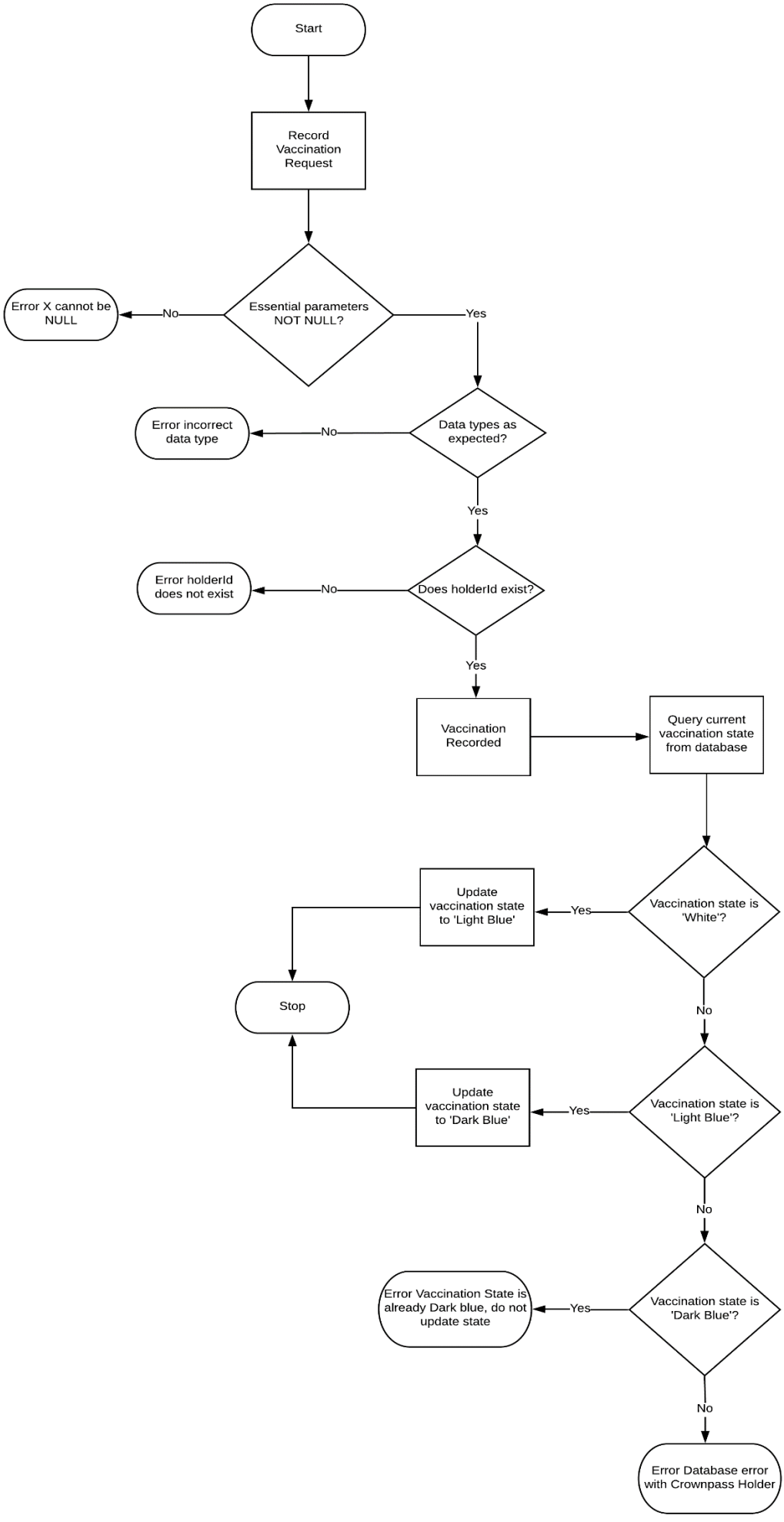
Selected class of objects from my subsystem, the interface class from the component diagram 'Record Vaccination':



Test Case	Pre-conditions	Method and Parameters	Expected Output
Vaccination recorded	No vaccinations are currently stored inside Vaccination Database	recordVaccination(vacc1) Vacc1 = Vaccination(00000001, 00000001, 25/11/2021, 15:30:35, 'Pfizer-02', 'XJ-149', 'Oxford Vacc Station 1', 00000002, 00000003)	HolderId, BookingId "Vaccination Recorded successfully" BookingId is returned for the updating of the booking status to 'complete'. The vaccination information is sent to the cloud to be stored in the Vaccination Database. Select * from the database should return an entry: 1, 00000001, 00000001, 25/11/2021, 15:30:35, Pfizer-02, XJ-149, Oxford Vacc Station 1, 00000002, 00000003
Vaccination recording failed due to holderId being null	holderId field cannot be NULL. Note: it should be stated that no ID field can be NULL.	recordVaccination(vacc2) Vacc2 = Vaccination(NULL, 00000005, 25/11/2021, 17:30:35, 'Pfizer-02', 'XJ-149', 'Oxford Vacc Station 1', 00000002, 00000003)	Error message: "Vaccination recording failed, holderId cannot be NULL."
Vaccination recording failed	Error handling complete with	recordVaccination(vacc3)	Error message: "Vaccination

due to incorrect data type for holderId	appropriate error messages	Vacc3 = Vaccination('Henry', 00000001, 25/11/2021, 17:30:35, 'Pfizer-02', 'XJ-149', 'Oxford Vacc Station 1', 00000002, 00000003)	recording failed, holderId is incorrect data type. (expected 'int', received 'string)"
Vaccination recording failed due to holderId not existing in the system	There is no holderId in User Database that is equal to 00000050	recordVaccination(vacc4) Vacc4 = Vaccination(00000050, 00000005, 25/11/2021, 17:30:35, 'Pfizer-02', 'XJ-149', 'Oxford Vacc Station 1', 00000002, 00000003)	Error message: "Vaccination recording failed, holderId 00000050 does not exist."
Vaccination State update triggered	Crownpass Holder associated with vaccination is registered & currently unvaccinated, Vaccination has just successfully been recorded	updateVaccinationState(00000001, 'Light Blue') Vaccination States are stored on the cloud and this method is triggered whenever a vaccination is recorded. I.e. it would check if current vaccinationState of the holderId = 'White' parse 'Light Blue' to the method	Select * from User Database and finding userId = 00000001 the associated vaccinationState has been updated to 'Light Blue'.
Vaccination State update failed	HolderId in question is already associated with a 'dark blue' vaccination state	updateVaccinationState(00000001, 'Dark Blue')	Vaccination state unchanged, still shows as dark blue. Returns error message "Cannot update vaccination state, holder is already in 'dark blue' state."

Unit test plan associated logic flow diagram:


Start
Record Vaccination Request
Essential parameters NOT NULL?
No
Error X cannot be NULL
Yes
Data types as expected?
No
Error incorrect data type
Yes
Does holderId exist?
No
Error holderId does not exist
Yes
Vaccination Recorded
Query current vaccination state from database
Vaccination state is 'White'?
Yes
Update vaccination state to 'Light Blue'
No
Vaccination state is 'Light Blue'?
Yes
Update vaccination state to 'Dark Blue'
No
Vaccination state is 'Dark Blue'?
Yes
Error Vaccination State is already Dark blue, do not update state
No
Error Database error with Crownpass Holder
Stop

3b)

Subsystem D – System Test Plan

Use case: **Add vaccination booking**

(1)

Scenario description: **Unsuccessful login data prevents operator from adding a booking**

Test case with test data (test data = colour red & italics)

Actor	**System**
1) Vaccine operator id *00000001* enters their login information as email *kevin1@gmail.com* and password *xyz123*.	2) Validates that the email & password combination exist in the cloud's user database, and finds it to be invalid.
	3) Displays an invalid login error message
4) Vaccine operator id *00000001* enters login information as email *kevin1@gmail.com* and password *xyz124*.	5) Validates that the email & password combination exist in the cloud's user database, and finds it to be invalid.
	6) Displays an invalid login error message

In this case the operator gave up as they could not resolve their password and email combination, and thus will query resetting their password.

Derivation of Test Data

	Variable	**Test Data 1**
Input	Vaccine operator ID	00000001
	Email	Kevin1@gmail.com
	Password	xyz123, xyz124
Stored data	Table of user information including passwords (normally hashed but not hashed for test example) and emails	00000001, Kevin1@gmail.com, xyz333 (not empty)
Expected output	passwordOK	False, False

Test Process

1. Set up test context

a. Vaccine Operator: Has received a booking request from a crownpass holder

b. Mobile device: Not already logged in

c. Cloud: User database contains user entry with id: 00000001, email: kevin1@gmail.com, password: xyz333

2. Vaccine operator enters login information

3. System gets login form

4. System validates login form against data in user database

a. Check if email & password combination match record in user database

b. Expected output: returns "Error, login information invalid" to mobile app

5. Vaccine operator repeats steps 2-4

a. Expected output: Vaccine operator cannot proceed with adding the booking

(2)

Scenario description: **Logged in vaccine operator cannot find a suitable time slot**

Test case with test data (test data = colour red & italics)

Actor	**System**
1) Vaccine operator id *00000001* enters their login information as email *kevin1@gmail.com* and password *xyz333*.	2) Validates that the email & password combination exist in the cloud's user database, confirms it is valid
	3) Requests booking datetime from operator
4) Operator enters booking datetime form giving the vaccination site as *'Oxford Vacc Station 1'*, date as *30/11/2021* and the time as *10:30am*.	5) Validates if the requested date and time are available at that station and finds there is already a booking scheduled for 10:30am on 30/11/2021.
	6) Displays timeslot unavailable error message
	7)Requests a new booking datetime
8) The operator enters vaccination site as *'Oxford Vacc Station 1'*, date *30/11/2021* and time as *10:45am.*	9) Validates if the requested date and time are available at that station and finds there is already a booking scheduled for 10:45am on 30/11/2021.
	10) Displays timeslot unavailable error message

In this case the operator could not find a suitable time in the range that the crownpass holder had asked for (over an email / phone call / in person) and so would now go and contact the holder and request a new range of times (or date), or to consider trying a different vaccination station.

Derivation of Test Data

	Variable	**Test Data 2**

Input	Vaccine operator ID	00000001
	Email	Kevin1@gmail.com
	Password	xyz333
	Vaccination Site	Oxford Vacc Station 1, Oxford Vacc Station 1
	Date	30/11/2021, 30/11/2021
	Time	10:30, 10:45
Stored data	Table of user information including passwords (normally hashed but not hashed for test example) and emails	00000001, Kevin1@gmail.com, xyz333 (not empty)
	Table of booking information including bookings associated with a vaccination site, a date, time as well as the relevant userId's	[OxfordVaccStation1, 30/11/2021(10:30(…),10:45(…)] *note '…' would be the id's associated with those bookings but are not relevant here, what matters is those times are taken
Expected output	passwordOK	True
	datetimeOK	False, False
	Table of booking information	[OxfordVaccStation1, 30/11/2021(10:30(…),10:45(…)]

Test Process

1. Set up test context

a. Vaccine Operator: Has received a booking request from a crownpass holder with a range of acceptable times (10:30, 10:45 on 30/11/2021 at Oxford Vacc Station 1)

b. Mobile device: Not already logged in

c. Cloud: User database contains user entry with id: 00000001, email: kevin1@gmail.com, password: xyz333, Booking database contains entries for Oxford Vacc Station 1, 30/11/2021 at 10:30 and at 10:45.

2. Vaccine operator enters login information

3. System gets login form

4. System validates login form against data in user database

a. Check if email & password combination match record in user database

b. Expected output: correct login information

5. System requests a booking datetime form to be completed by the vaccine operator

6. Operator inputs booking datetime Oxford Vacc Station 1, 30/11/2021, 10:30

7. System receives datetime form

8. System validates datetime form against booking entries in the Booking Database

a. Check if any records have all-identical fields i.e. is there a record already with Oxford Vacc Station 1, 30/11/2021, 10:30

b. Expected output: returns “Error, Booking datetime unavailable at selected Vaccination Centre” to mobile app

9. Vaccine operator repeats steps 5-8

a. Expected output: vaccine operator cannot proceed with the booking as the selected timeslots were unavailable.

(3)

Scenario description: **Logged in vaccine operator resolves time conflict and adds a new booking to the system.**

Test case with test data (test data = colour red & italics)

Actor	**System**
1) Vaccine operator id *00000001* enters their login information as email *kevin1@gmail.com* and password *xyz333*.	2) Validates that the email & password combination exist in the cloud’s user database, confirms it is valid
	3) Requests booking datetime from operator
4) Operator enters booking datetime form giving the vaccination site as *‘Oxford Vacc Station 1’*, date as *30/11/2021* and the time as *11:00am*.	5) Validates if the requested date and time are available at that station and finds there is already a booking scheduled for 11:00am on 30/11/2021.
	6) Displays timeslot unavailable error message
	7)Requests a new booking datetime
8) The operator enters vaccination site as *‘Oxford Vacc Station 1’*, date *01/12/2021* and time as *11:30am*.	9) Validates if the requested date and time are available and finds this is a free slot.
	10) Requests booking information
11) Operator completes booking information form giving holderId (belonging to the crownpass holder who requested the booking) as *00000010* and staffId as *00000001*	12) The system creates a booking with attributes vaccination site as ‘Oxford Vacc Station 1’, holderId as 00000010, staffId as 00000001, date as 01/12/2021 and time as 11:30am.
	13) The system appends a booking reference to the booking, bookingId *00000050*.
	14) The system inserts this booking object into the booking database
	15) The system returns success confirmation to the mobile app’s display with the booking reference attached

16) The operator receives the confirmation and booking reference	

In this test case the vaccine operator is successful in their use case of adding a booking to the system and would now confirm with holder who requested it that the booking has been created at Oxford Vacc station 1, 01/12/2021 at 11:30am.

Derivation of Test Data

	Variable	**Test Data 3**
Input	Vaccine operator ID	00000001
	Email	Kevin1@gmail.com
	Password	xyz333
	Vaccination Site	Oxford Vacc Station 1, Oxford Vacc Station 1
	Date	30/11/2021, 01/12/2021
	Time	11:00, 11:30
	Crownpass Holder ID	00000010
	Booking ID	00000050
Stored data	Table of user information including passwords (normally hashed but not hashed for test example) and emails	00000001, Kevin1@gmail.com, xyz333 (not empty)
	Table of booking information including bookings associated with a vaccination site, a date, time as well as the relevant userId's	[OxfordVaccStation1, 30/11/2021(11:00(…)), 01/12/2021(11:00(…))]
Expected output	passwordOK	True
	datetimeOK	False, True
	Table of booking information	[OxfordVaccStation1, 30/11/2021(11:00(…)), 01/12/2021(11:00(…), 11:30(00000050, 00000010, 00000001, INCOMPLETE)]

Test Process

1. Set up test context

a. Vaccine Operator: Has received a booking request from a crownpass holder with a range of acceptable times (11:00 on 30/11/2021 at Oxford Vacc Station 1, 11:30 on 01/12/2021 at Oxford Vacc Station 1)

b. Mobile device: Not already logged in

c. Cloud: User database contains user entry with id: 00000001, email: kevin1@gmail.com, password: xyz333, Booking database contains booking entry for Oxford Vacc Station 1, 30/11/2021 at 11:00.

2. Vaccine operator enters login information

3. System gets login form

4. System validates login form against data in user database

a. Check if email & password combination match record in user database

b. Expected output: correct login information

5. System requests a booking datetime form to be completed by the vaccine operator

6. Operator inputs booking datetime Oxford Vacc Station 1, 30/11/2021, 11:00

7. System receives datetime form

8. System validates datetime form against booking entries in the Booking Database

a. Check if any records have all-identical fields i.e. is there a record already with Oxford Vacc Station 1, 30/11/2021, 11:00

b. Expected output: returns “Error, Booking datetime unavailable at selected Vaccination Centre” to mobile app

9. Repeat step 5-8, but on step 6 Operator inputs Oxford Vacc Station 1, 01/12/2021, 11:30

a. Check if any records have all-identical fields i.e. is there a record already with Oxford Vacc Station 1, 01/12/2021, 11:30

b. Expected output: Booking time available

10. System requests a booking information form to be completed by the vaccine operator

11. Operator enters the holderId & their staffId into the booking form (00000010, 00000001)

12. System receives booking information form

13. System creates a booking with all attributes it has received (Vaccination Site: ‘Oxford Vacc Station 1’, holderId: 00000010, staffId: 00000001, date: 01/12/2021 and time: 11:30am, status: INCOMPLETE.

*NOTE: status automatically appended by system on all new bookings as ‘INCOMPLETE’

14. System creates a booking reference (00000050)

15. System inserts this booking object into the Booking Database

16. System returns confirmation to operator

a. Expected output: Booking successfully added to system

Checking my test cases against the activity diagram produced in 1b) & sequence diagram in 1d) I am satisfied that in terms of correctness each test case derived from the activity diagram does correspond to one path in the sequence diagram and thus they are consistent with each other.

In terms of adequacy the sets of test cases do cover all messages in the sequence diagram so I am satisfied with this as well.

Note: Images seem to lose quality when exported to PDF format, please see git repository for full quality diagrams