COMP1004

Computing Practice

2020/2021

Project Title

Doki-Doki Delight Management System

Contents

Executive Summary

This document will show my research into UML through the book UML @ Classroom (Seidl et al., 2014), and the results of this research will be various UML diagrams. Through the contents list above one may view all various types of UML diagrams that have been formed to represent the requirements of the Doki-Doki Management System.

Use-Case Diagrams

These diagrams describe the usage and actors of a system in an abstract method (meaning that detail is kept minimal). Often times these diagrams represent a role, and that a person in that role is trying to accomplish by using the system. As explained in UML @ Classroom (Seidl et al., 2014), Use Case Diagrams can be employed to cover (visually) these three questions:

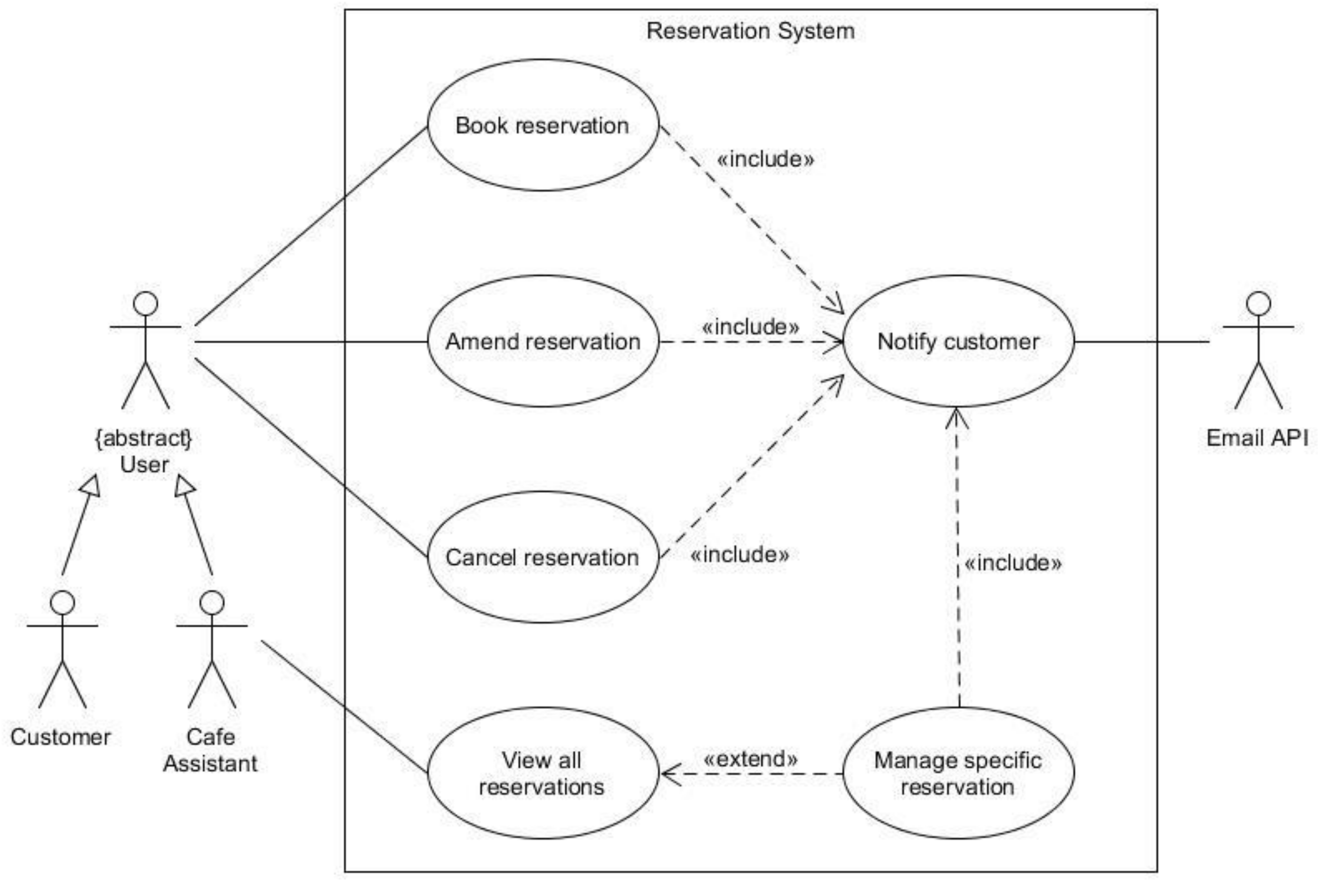
- What is being described?
- Who interacts with the system?
- What can the actors do?

When making these Use Case Diagrams, if the diagram is not self-explanatory or requires extra understanding, I will accompany the diagram with a Use Case Description. Descriptions are used to further explain the system and provide clarity to what the Diagram is trying to show – whether this is by specifying the preconditions or stating potential errors.

Reservation System

The reservation system is the main functionality of the webpage application – used by both the Doki-Doki Delight staff assistants and customers alike it will enable the management of:

- A customer's own reservation
- All of Doki-Doki Delight's current reservations



Name:	Book reservation
Short description:	A customer wants to create a reservation so they can dine-in at the venue. Can be done independently or through the customer assistant (both approaches utilise the website application)
Precondition(s):	Customer must have: full name, email address, telephone number, number of occupants.
Postcondition(s):	Reservation has been made in customer's name.
Error situations:	(a) invalid name (b) invalid email address (c) invalid telephone number (d) invalid number of occupants
System state in the event of an error:	Customer has not booked a reservation.
Actors:	Customer OR Café assistant + customer
Trigger:	Customer selects to book reservation OR customer contacts café assistant
Standard process:	(1) Customer selects time/date (2) Customer inputs full name (3) Customer inputs telephone number (4) Customer inputs email (5) Customer inputs number of occupants (6) Customer confirms details (7) Reservation ID is generated
Alternative processes	Customer tells the Café Assistant all of the above information, and they input it for them.

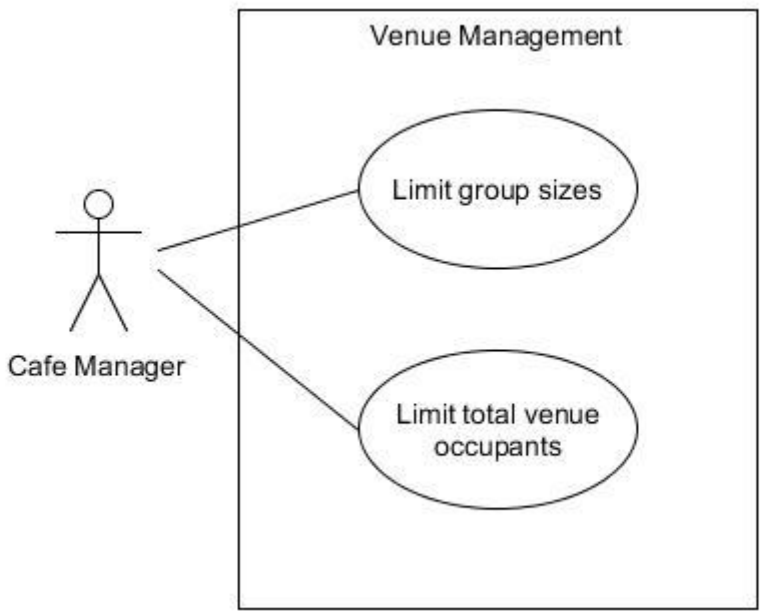
Name:	Cancel reservation
Short description:	A customer wants to delete their reservation from the system.
Precondition(s):	Customer must have: reservation ID, full name
Postcondition(s):	Customer's reservation has been deleted
Error situations:	(a) invalid name (b) reservation ID
System state in the event of an error:	Customer has not deleted reservation
Actors:	Customer OR Café assistant OR both
Trigger:	Customer selects cancel reservation OR customer contacts café assistant
Standard process:	(1) Customer inputs full name (2) Customer inputs reservation ID (3) Customer selects delete reservation (4) Reservation is deleted from system
Alternative processes	(1') Café assistant selects delete (2') Café assistant selects confirm

Name:	Amend reservation
Short description:	A customer wants to change their reservation details.
Precondition(s):	Must have a 6+ hour time difference between the present and the reservation timeslot. Customer must have: full name, reservation ID
Postcondition(s):	Customer’s reservation has had changes applied to it.
Error situations:	(a) invalid number of occupants (b) invalid timeslot
System state in the event of an error:	Customer has not made any changes
Actors:	Customer OR Café assistant + Customer
Trigger:	Customer selects amend reservation OR customer contacts café assistant
Standard process:	(1) Customer inputs reservation ID (2) Customer inputs full name (3) Customer changes information (4) Customer saves changes.
Alternative processes	Customer tells the Café Assistant the above information and they make the requested changes for them.

Name:	View all reservations
Short description:	A café assistant wants to view all reservations and perhaps make changes on behalf of a customer.
Precondition(s):	Café assistant is logged in to the system.
Postcondition(s):	Café assistant is viewing all placed reservations (high level overview)
Error situations:	(a) Cafe assistant signs out
System state in the event of an error:	Café assistant is not permitted to view reservations
Actors:	Café assistant
Trigger:	Café assistant selects view all reservations
Standard process:	(1) Café assistant see all the reservations and scroll through them (2) Café assistant can click on individual reservations to change them
Alternative processes	---

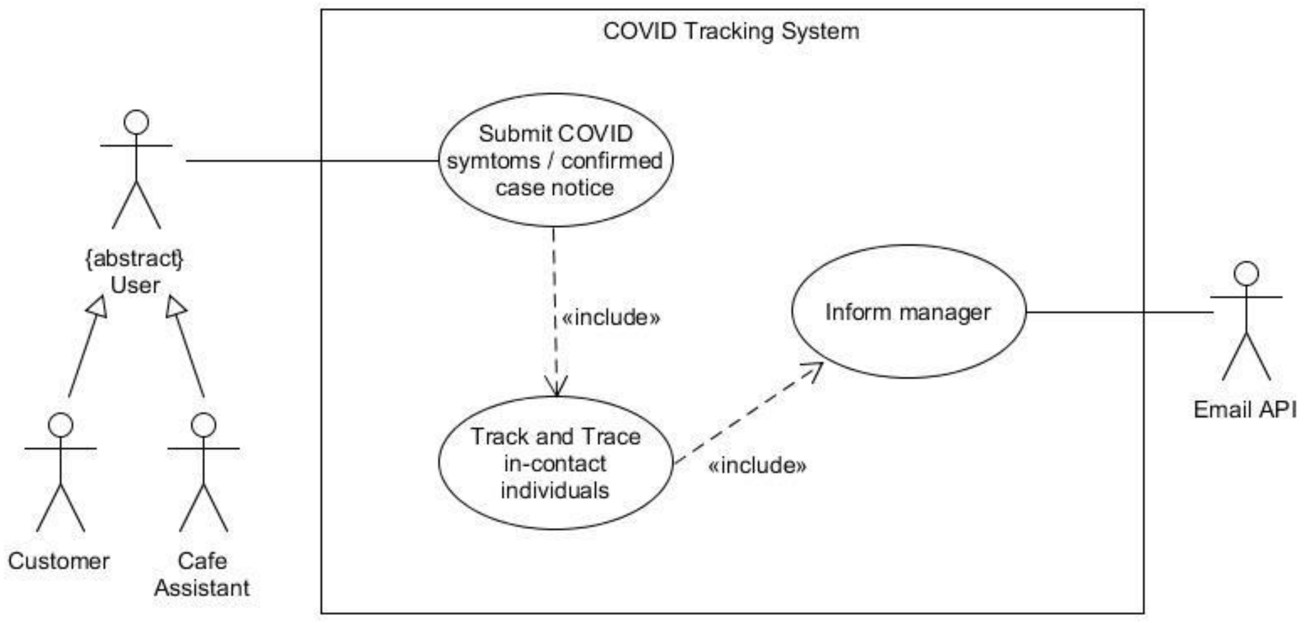
Venue Management

Managing the venue is a crucial part of ensuring that COVID guidelines are adhered to, but also making sure that the venue details can be adjusted – which may be required because of business expansion.



COVID-19 Tracking System

Track-and-Trace is an important part of identifying individuals that have been in-contact with someone who is reporting symptoms or is already confirmed to have COVID-19. This facility is important as the people who have been identified as "in-contact" need to be notified about this – in an effort to minimise the spread of the pandemic.

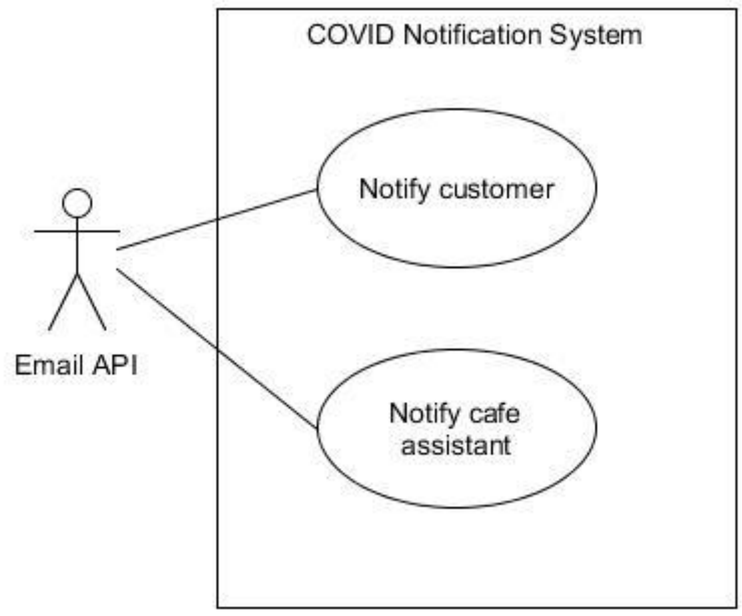


Name:	Submit COVID symptoms / confirmed case notice
Short description:	A café assistant or customer can submit a notice to the business to inform them that they have COVID or COVID symptoms. This will be used to control the spread of the pandemic and notify other individuals.
Precondition(s):	Must be a customer or café assistant
Postcondition(s):	Notice has been submitted
Error situations:	(a) invalid name (b) invalid email address (c) invalid telephone number (d) invalid reservation ID
System state in the event of an error:	Café assistant OR customer is not permitted to file a COVID notice
Actors:	Customer OR Café assistant
Trigger:	Café assistant selects to submit a COVID notice
Standard process:	(1) Customer inputs role as customer (2) Customer inputs reservation ID (3) Customer inputs full name (4) Customer inputs email address (5) Customer inputs telephone number (6) Customer confirms notice submission
Alternative processes	(1’) Cafe assistant inputs role as assistant (2’) Assistant inputs full name (3’) Assistant inputs email address (4’) Assistant inputs telephone number (5’) Assistant confirms notice submission

Name:	Track-and-Trace in-contact Individuals
Short description:	Finds all staff and customers that might have had contact with the individual submitting the notice.
Precondition(s):	COVID notice must be valid when submit
Postcondition(s):	A list of in-contact individuals and their contact information is formed
Error situations:	(a) Cannot find reservation ID in reservation history
System state in the event of an error:	Insufficient information provided
Actors:	Customer OR Café assistant
Trigger:	COVID notice has been submitted
Standard process:	(1) Find other reservations at the same time as reservation ID (2) Find assistants on shift at the time of the reservation ID
Alternative processes	(1’) Find all reservations during an assistant’s shift (2’) Find other assistants on shift at same time as the assistant who submitted the notice

COVID-19 Notification System

Staff and customers will need to be notified when the system has received feedback, from the café manager, that someone (whether they are staff or a customer) has notified the business that they have COVID or COVID symptoms. This system will work alongside the COVID Tracker component.



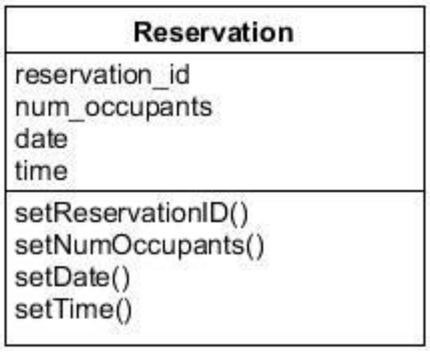
Class and Object Diagrams

A Class Diagram is used to model the structure of the system – and this structure will be concrete and never change. There are often different ways of representing class diagrams at different junctions within the SDLC, such as: within Analysis the Class Diagram will often be conceptual, but within Design the diagram may refine the previous version and go more into detail, such as by noting the relationships between the classes.

Object Diagrams are used to model objects, of which the classes provide the template/foundation for. Therefore, objects are instances of classes. However, what are objects? An object is an instance of a class and it defines the: attributes, behaviour, and state of an entity. Objects provide a structured approach to programming – therefore meaning that the characteristics of maintainability and extensibility are easier to be implemented.

(The above explanations were written with help from UML @ Classroom)

Reservation Class

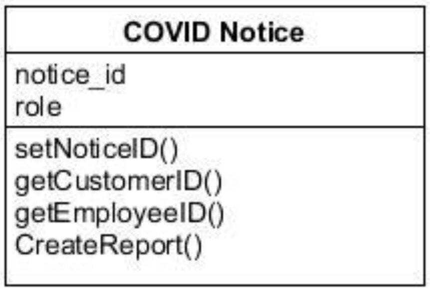


<u>Reservation:</u>

The reservation class defines the attributes and behaviours that this particular part of the system will require.

Within the diagram showing relationships, the reservation class is associated with a single user, a single venue, and many notifications.

COVID Notice Class

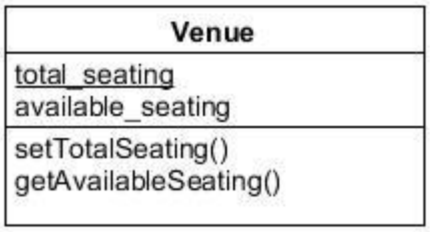


<u>COVID Notice:</u>

The notice class defines a unique identifier to itself and accepts the user's role as an input. Likewise, the behaviours will include fetching other customer and employees (via their IDs) to create a report of all possible contacted persons.

The notice class is associated with 1 user and many notifications.

Venue Class

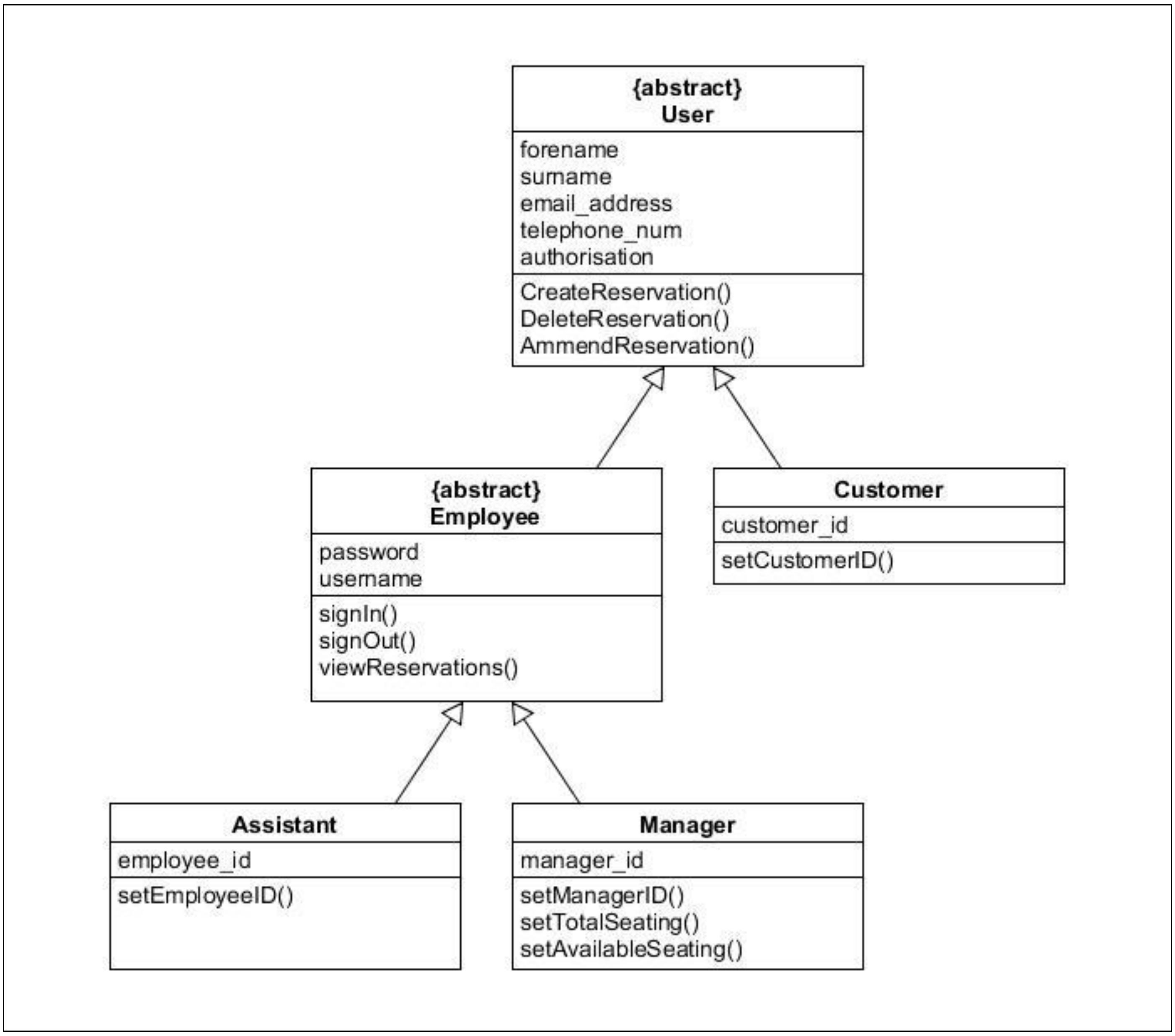


<u>Venue:</u>

The venue class defines the limitations of the café, such as through the total and available seating leftover (attributes).

The venue class is associated with many reservations and one manager.

Users Classes



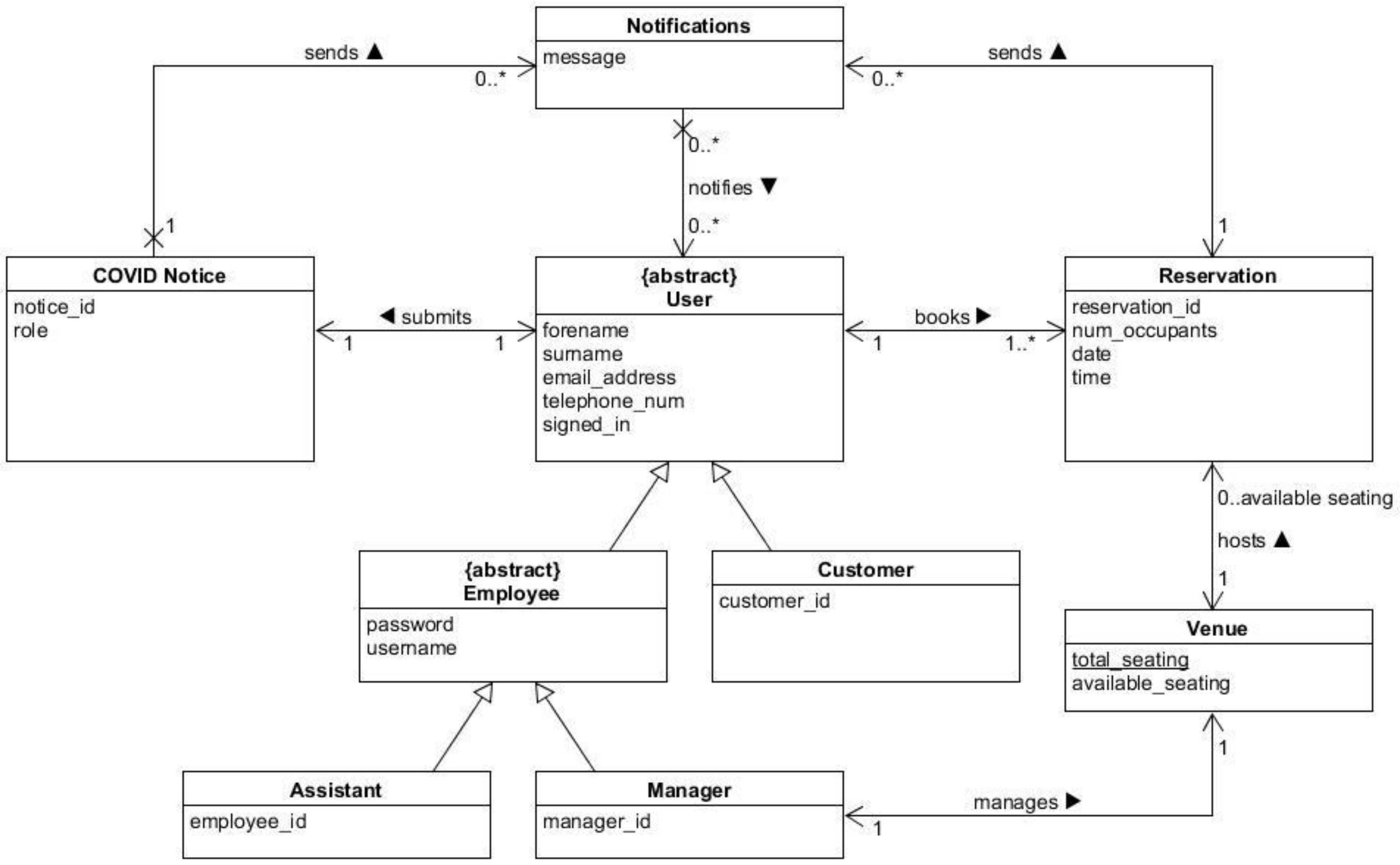
Within the above diagram I have depicted two {abstract} classes, which both express and depict the relationship between the audiences that my website is aimed at. A more descriptive word than audience is 'User' and therefore the super class is called User – however this class does not truly exist and is therefore an *abstract*. Likewise, the term 'Employee' can be better used to describe the relationship between manager and assistant.

The subclasses will inherit attributes and behaviours from their super classes and therefore all users will be able to create a reservation, but only the manager can define the seating arrangements of the venue. Likewise, an employee will be able to sign into the system, whereas a customer does not need this functionality.

Note:
all of these above classes are intended to be further refined over the duration of the project and therefore, the current Analysis is intended to be conceptual, and not concrete or mapped to the best of their ability.

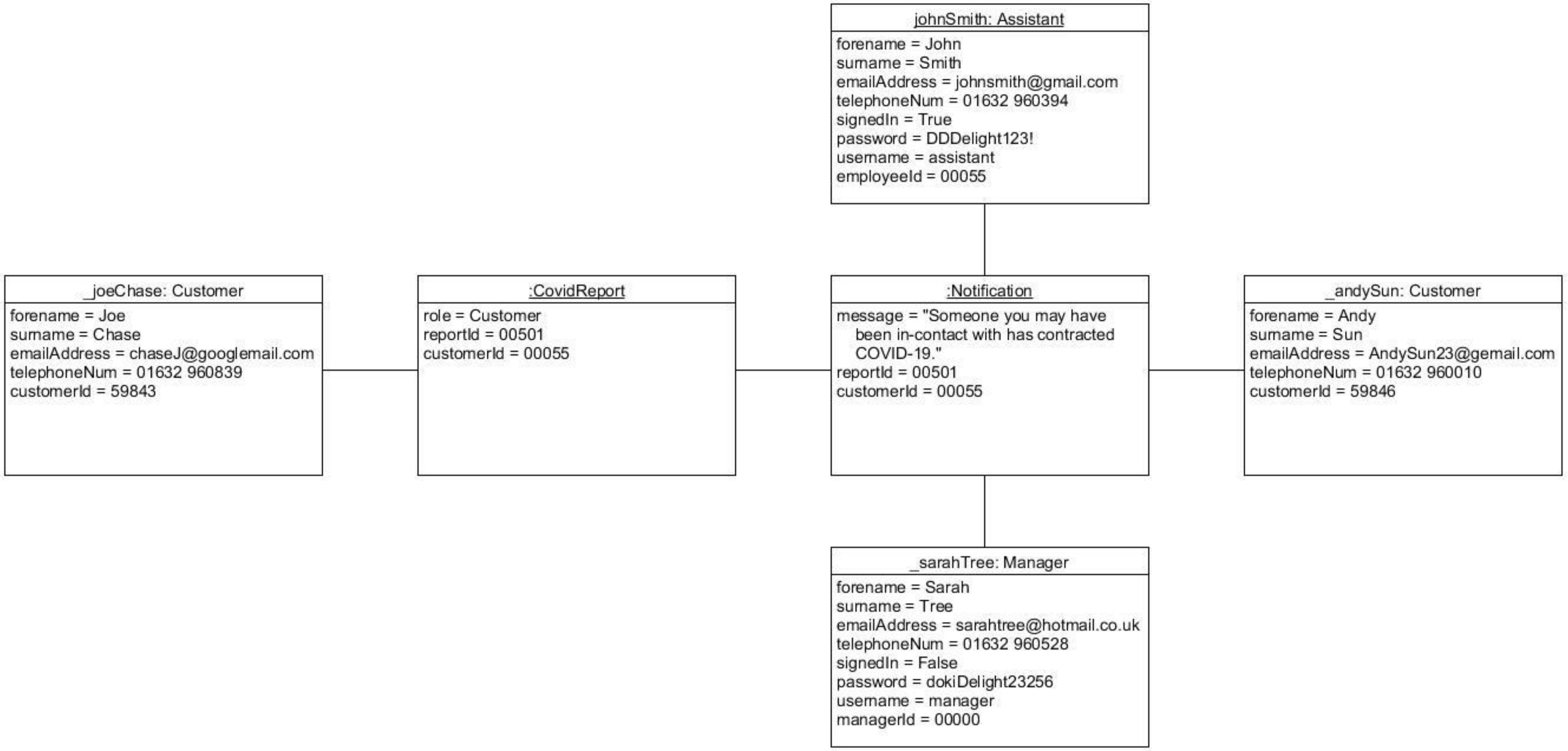
Class Relationships

This diagram describes the associations between each class within the Reservation System.



COVID Notice Object Association Diagram

This diagram describes the associations between each object within the COVID report/notice system.

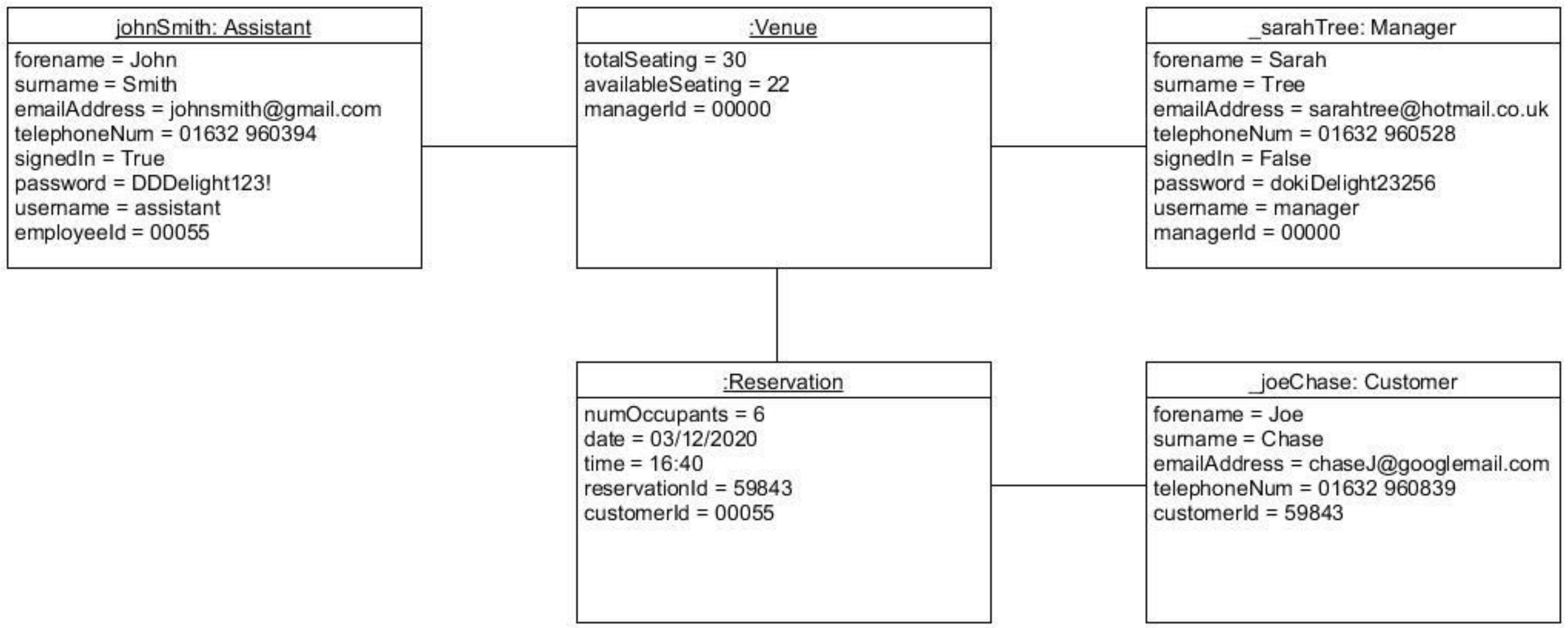


Scenario:

A customer submits a COVID report and therefore notifications are sent to a collection of individuals that may have been in-contact with them. Details of implementing this tracing algorithm have been abstracted for simplicity.

Reservation Object Association Diagram

This diagram describes the associations between each object within the restaurant booking system.



Scenario:

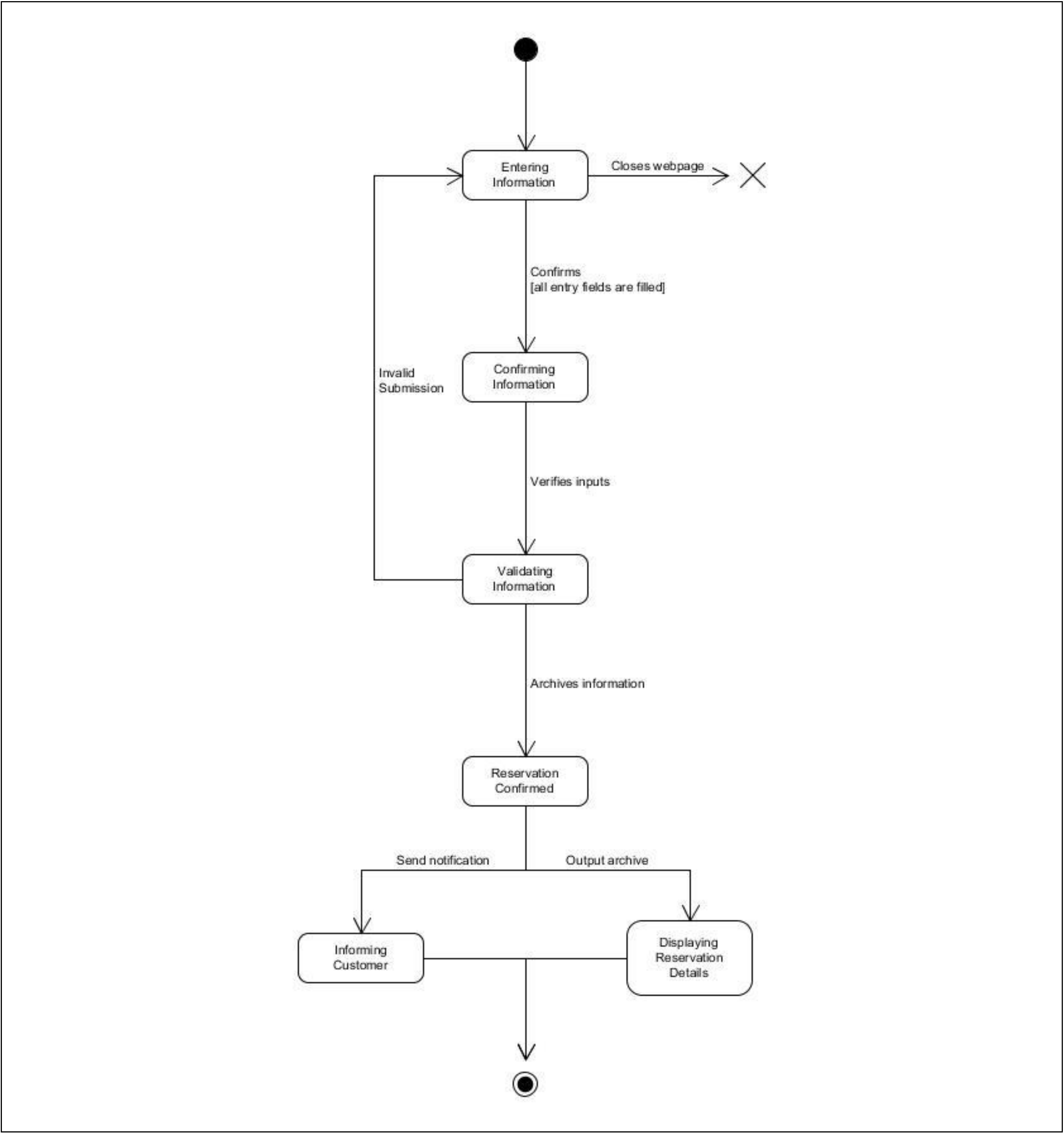
A reservation can be booked by a Customer, and the Reservation object requires a Venue to be created – the venue will limit the amount of reservations being created. A venue will be run by the business's Manager and Assistant. All users must input their personal information so that they can be contacted.

State Machine Diagrams

These diagrams model ‘states’ of objects/entities, but what is a ‘state’? A state can be defined as the condition of an object, which may fluctuate and change over periods of time – states are therefore not set/fixed. To cause a state to change an event or trigger must usually occur, and then the state can be changed to reflect an operation effectively. Therefore, a State Machine Diagram can model an object and its behaviour within the system. Likewise, the model shows a differing ‘life cycle view’ wherein the diagram determines a particular response.

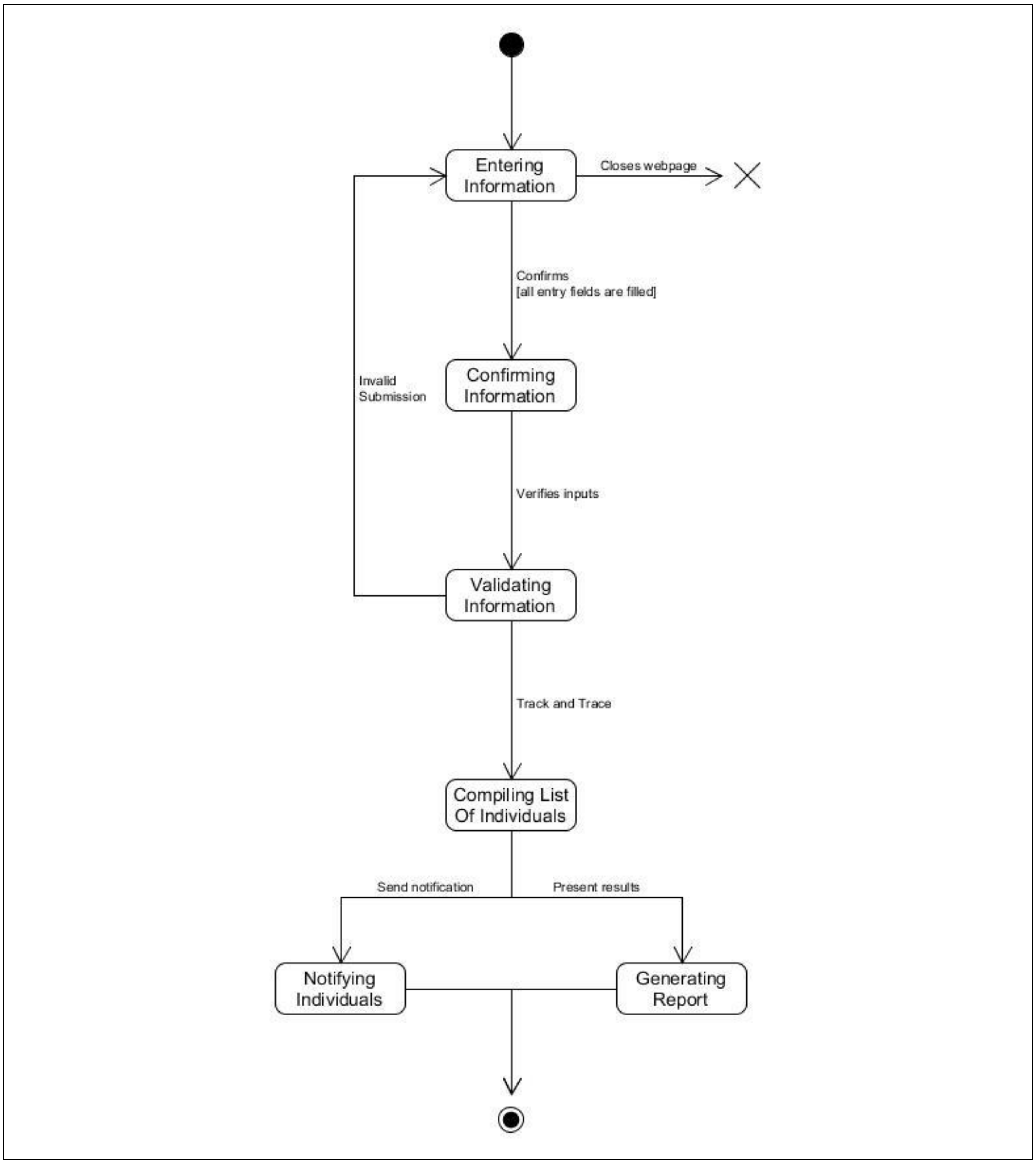
Reservation Process

This diagram presents the states of the Reservation System and the events (→) required to transition between states. The system starts off in the Entering Information state where the system will expect specific conditions to be met (in this case the user must input information) to transition.



COVID-19 Reporting Process

This diagram presents the states of the COVID report/notice submission system and the events (→) required to transition between states. The system starts off in the Entering Information state where the system will expect specific conditions to be met (in this case the user must input information) to transition.



This diagram is fairly similar to the Reservation System state diagram as they both follow the same procedure of receiving an input, receiving confirmation to process the input, validating the information and then either proceeding with tracking and tracing in-contact individuals or making the user resubmit their report.

After tracking all individuals, they will be notified, and a report will be produced for the manager.

Sequence Diagrams

These diagrams specify interactions by modelling scenarios and displaying exchanges. Within the model there are triggers/events and conditions, but also behaviours and objects. As ‘sequence’ means, the diagram portrays a chronological, ordered structure to conceptualising systems.

As part of the early SDLC I will provide simple diagrams and may revisit them throughout the iterations of the project. In those instances, the diagrams may go under drastic changes, or not change at all.

Notes:

A diagram for the Reservation System can be found on the following page

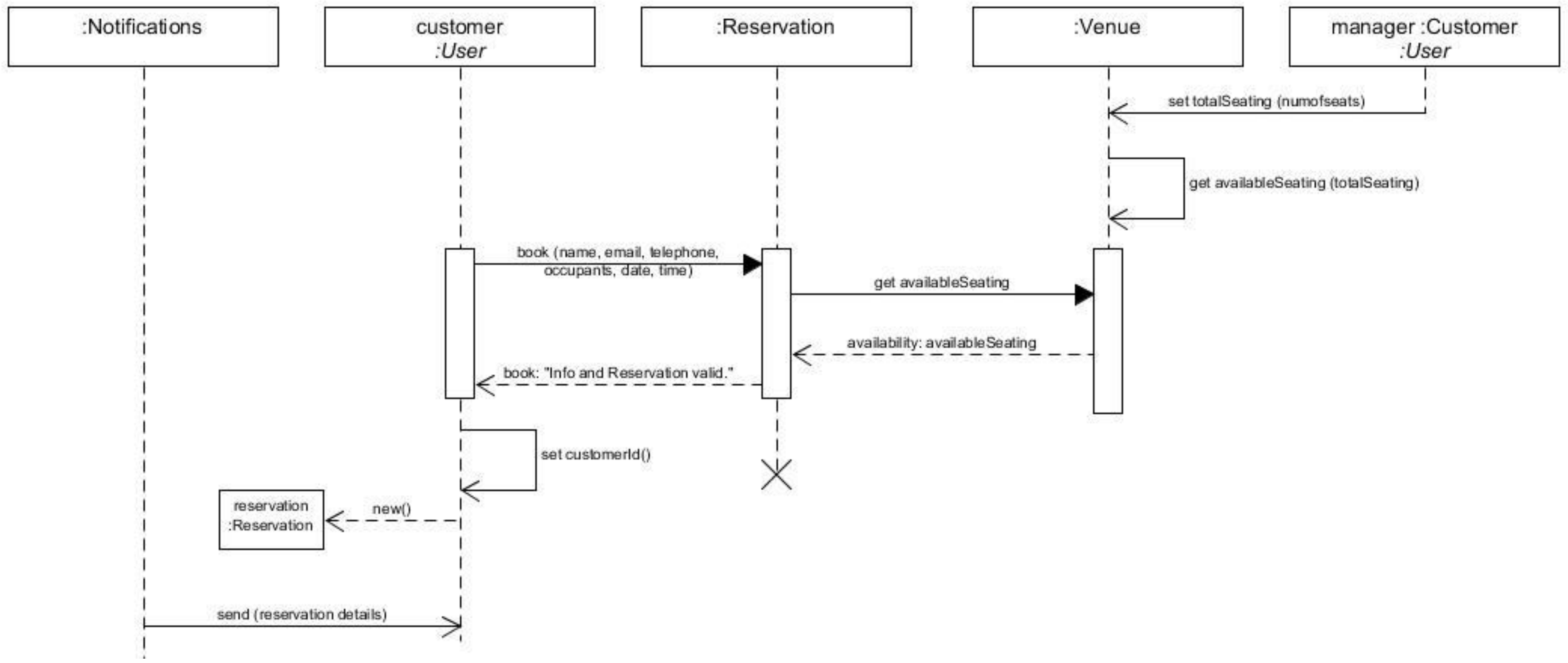
- This diagram has some inconsistencies with the other diagrams
  - Namely, the lack of implementation of receiving input data and verifying it
  - This has been left out to maximise simplicity.

- A diagram for creating a COVID report has not been created as I’m finding it urgent to start implementation so I can get through my first iteration of the prototype.

- When I reach the necessity to implement the COVID system I will create a diagram and paste it here.

Creating a Reservation

The below scenario depicts a customer creating a reservation whilst on the webpage. In this model the customer can be replaced with the subclass – assistant:Employee:*User*


:Notifications
customer :User
:Reservation
:Venue
manager :Customer :User
set totalSeating (numofseats)
get availableSeating (totalSeating)
book (name, email, telephone, occupants, date, time)
get availableSeating
availability: availableSeating
book: "Info and Reservation valid."
set customerId()
reservation :Reservation
new()
send (reservation details)

1. Manager provides venue seating information
2. Get the current available seating of the venue
3. Customer books a reservation with their input data, in parenthesis – implementation of setting this data has been abstracted out for simplicity.
4. View if there is enough space in the venue
5. Either – reservation is valid, or terminate
6. Set customerId
7. Create a reservation
8. Send a notification via email

Sources

Seidl, M., Scholz, M., Huemer, C., Kappel, G. and Duffy, T., 2014. *UML @ Classroom*. 1st ed. Cham: Springer, pp.11-161.

Planner:
https://tasks.office.com/live.plymouth.ac.uk/Home/PlanViews/KB8DrzSpokCUWjx6hWH765YACcBq?Type=PlanLink&Channel=Link&CreatedTime=637402140671510000

Edits / Notes

1. First iteration completed as of 07/12/20.