COMP1004

Computing Practice

2020/2021

Project Title

Doki-Doki Delight Management System

Contents

Introduction

This report will be segmented into six components which are to be presented to help the reader understand the current position of the project, but to also showcase my retrospective thoughts in relation to any complications or achievement. Therefore, the reader may freely interpret this document as a log of the project and all actions that I have made to prepare and develop my initial prototype.

- Software development lifecycle
- Project description
- Requirements
- Architecture
- Sprint Planning
- Reflection

Within each section, I hope to express a coherent and relevant argument as for why I have proceeded with certain tasks and how they are relevant to my project. This report is a testament to the professional and descriptive manner in which the web app has been regarded – such as in planning and designing a 'successful' solution for achieving the product vision.

Software Development Lifecycle

Long before the existence of software, people were already applying structured approaches to building infrastructure. These models were often linear, wherein certain phases would be completed within a certain order, one by one. And when software became an ongoingly mainstream revelation, the act of planning, designing, and implementing software was no exception to the rigid, linear models. This can be seen in the Waterfall model, which has been in action since the late 20th century.

Since then, the digital world has progressed much further and therefore new models have been created and are now widely regarded, such as the various Agile methodologies. Nowadays, developers have a variety of different models upon which they can structure their projects around – and this is exactly what the Software Development Lifecycle is – a concept wherein if developers apply a specific structure to their way of development then they can maximise the chance of success.

However, there are a multitude of different types of projects and therefore it is up to the developers, or their sponsors, to consider which methodologies are going to help them achieve the type of success they want to achieve. Success may be: low costs; changeability; scalability; or even, fast production. These are just a few types of successes that a developer may hope to achieve – but not all methodology will work for all of these factors. For instance, the Waterfall model might be perfect for a low cost, low risk project – but it is not suitable for a project where needs and functionality will always be changing to meet the demand of stakeholders.

But what makes one model different from another model? First, the reader must understand that each model is essentially a framework that is built upon phases/stages. There are generally 5 stages, but some models may be extended to be compatible with more. Finally, the way each model handles each stage sequence, and the tasks carried out within, will be different.

- Requirement Analysis
- Design
- Implementation
- Testing
- Evolution

The *Doki-Doki Delight Management System* is structured around the Agile framework. Agile is represented by 12 principles and together they express the priorities that the model tries to maintain. From these principles one may interpret Agile as the go-to model for customer satisfaction – which is maintained via a "continuous delivery of valuable software" (Martin and Martin, 2006), as well as the continuous interaction between developers, stakeholders, and sponsors.

One such factor of Agile that I have come to understand the importance of over the past few months is the concept of retrospection. In Agile, the motivation and confidence of the team is arguably one of the most important considerations that a team leader has to make when coming to decisions. Individual responsibilities and burdens should not exist within Agile – instead, the team should often reflect on their troubles and "adjust its behaviour" (Martin and Martin, 2006) to overcome issues. However, as the sole developer of this project I have come to learn that the *team* component is not essential to overcome issues. By reflecting on my own developments and struggles (within the weekly blog posts) I can personally understand my shortcomings and act on them.

In terms of the five stages, I have applied the Agile methodology by examining the project's requirements and then extracting a project backlog from these requirements. From the project backlog I have further decomposed the tasks into weekly sprints, wherein I devote my expertise to a specific set of tasks from the backlog. The bi-weekly meetings that I have been attending with Liz Stuart can be deemed as an 'inspection' of sorts, where she assumes the theoretical role as sponsor to the project and offers guidance, advice, and feedback.

Overall, up until this midpoint I have been working in sprints and reflecting on the quality of my work, and the extent of my productivity. As my initial prototype is nearing completion, I am expected to soon enter the cyclical process of Design, Implementation and Testing, to further strengthen my prototype. These prototypes are expected to be functional, but not complete, deliverables – where if requested, can be demonstrated as meeting the highest priority requirements.

Project Description

When thinking about the differences between a ‘good’ café and a ‘bad’ café it would be almost certain that it is the degree to which customer satisfaction can be proven. It is a business’s duty to serve their customers and fulfil their desire. Customer satisfaction is something that can be controlled and maintained by ensuring that services can be used intuitively and are easy to access.

In recognition of this, Doki-Doki Delight, a small Japanese inspired café, has realised that their current method of handling customer reservations is outdated. Using paper is not suitable in the modern world, and they therefore have issued a request for a web-application to help them grow as a business. Doki-Doki Delight requires this web-app to enable customers to book reservations, and for staff to manage these reservations. However, due to the ongoing pandemic they require Track and Trace functionality so that their customers can be made aware that they might have been in contact with someone who has COVID symptoms. Likewise, the webapp will be tailored to ensure that the pandemic regulations can be maintained and that laws are not breached.

The importance of a web related service for cafés is summarised within a 2013 article (Etemad-Sajadi, 2014) wherein it is stated, “A website … offers [customers] the chance to experience something of its atmosphere, level of service and genre of cuisine”. Used as a tool to gain a customer’s attention and to convince them to visit a business’s premises the webapp can be used as a type of interactive advertisement. Within the same article the author further draws a connection between repeat purchases and ‘highly satisfying’ websites, concluding why a webapp is a modern necessity for businesses.

Furthermore, to ensure that the project can be successfully deployed and maintained I have considered the legalities of the project. Some of these legalities also try to combat social inequalities and maintain ethical practices.

- The official government guidance for food businesses
- Companies Act 2006
- General Data Protection Regulation
- Equality Act 2010.

The government has restricted (at the time of writing) only to persons travelling in social bubbles of 6 and less and from this group customer contact information must be recorded; likewise, staff schedules should also be recorded (Guidance for food businesses on coronavirus (COVID-19), 2020). As personal information is being collected, the GDPR comes into effect as relates to ethically handling data and being transparent about how data is processed. Likewise, the Companies Act makes it mandatory to supply specific business information on the webapp. Finally, the Equality Act dictates that the webapp should not discriminate against individuals. The website should therefore maintain inclusivity for a variety of different target audiences.

Overall, the project has been designed to tackle a specific problem that Doki-Doki Delight faces, but in overcoming this problem, the other considerations and issues that have been mentioned should not be ignored, but also measured and maintained. Each deliverable should uphold the core values of usability and inclusivity (Scott, n.d.), and this is what the next topic outlines.

Requirements

Requirements Analysis is a critical stage of every SDLC methodology wherein the developers liaise with their client to form a mutual agreement on what the project is to be. Within this discussion they are certain to discuss the essentials of the project, such as the functionality that should be implemented within the prototype. This phase is significant in ensuring that the project is carried out 'successfully'. However, as Doki-Doki Delight is my own fabrication I had to singlehandedly evaluate the scale of the café management webapp, of which this section will present.

The first step in extracting the requirements from the project was by understanding that there are two main types of requirements: functional and non-functional. Functional requirements are categorised by their ability to identify what the prototype should do; whereas non-functional requirements detail the constraints of a prototype and should therefore be measurable.

- Business
- Administrative
- User
- System

The above functional requirement categories helped me determine that specific stakeholders should be able to carry out specific tasks (Cox, 2017) – but I intentionally abstracted out 'how' they might carry out the tasks as this unnecessarily restricts and complicates the analysis phase.

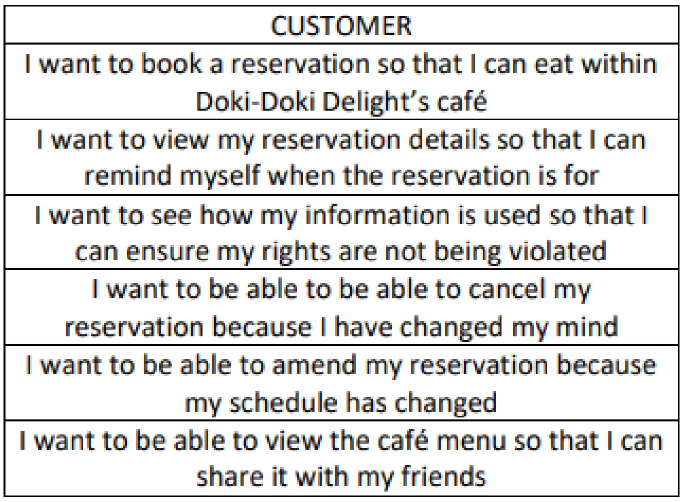
However, the following non-functional requirement classifications (Eriksson, 2012) made me explore the project in terms of statistics, but they also made me consider any external forces – such as laws and regulations.

- Usability
- Security
- Readability
- Social
- Availability
- Ethical
- Performance

At this stage, I had numerous unrefined requirements that were not in any sort of format. To further extend the analysis of my requirements it was mandatory that I gave them structure, and therefore resorted to a prioritisation technique called MoSCoW.

MoSCoW is a method of processing requirements by sorting them into the four tiers: must-have, should-have, could-have, wont-have. The must-have requirements are of the highest priority and these requirements must be satisfied for the project to be successful; the requirements on the opposite side of the spectrum are not essential. Through this process I refined and ordered the requirements – making it much easier to form my user stories.

User stories are concise explanations of the main functionality required from the main users' perspective (North, n.d.). Determining that my webapp will have two types of users – staff and customers, I formed stakeholder 'profiles'. My 'profiles' expressed all of the stakeholder's needs in terms of usage of the webapp – for example, customers want to make reservations with the intent of eating-in at the premises.



CUSTOMER
I want to book a reservation so that I can eat within Doki-Doki Delight's café
I want to view my reservation details so that I can remind myself when the reservation is for
I want to see how my information is used so that I can ensure my rights are not being violated
I want to be able to be able to cancel my reservation because I have changed my mind
I want to be able to amend my reservation because my schedule has changed
I want to be able to view the café menu so that I can share it with my friends

STAFF
As a front-door staff member I want to be able to view customer contact information so that I can contact them if any urgent news needs to be conveyed.
As a manager I want to be able to limit entry to the venue to only social bubbles of 6 or less so that I do not break the law
As a manager I want to be able to only accept a certain total amount of individuals into the café so that I do not break the law
As a manager I want to have a record of all staff and customer information so that I can contact them if a COVID outbreak is confirmed
As a front-door staff member I want to be able to add reservations to the system if a customer contacts the café over telephone so that I can provide exemplary customer service
As a front-door staff member I want to be able to cancel/delete customer reservations on the system if a customer contacts me so that I can provide exemplary customer service
As a staff member I want to be able to access the system on a desktop so that I can use mouse and keyboard to input data
As a staff member I want to use keyboard and mouse so that I can be more efficient when typing/inputting data
As a staff member I need to sign into the system so that I have permissions to look at business info and potentially amend it
As a staff member I need the system to automatically sign me out if I am inactive for more 10 minutes so that unauthorised access may be prevented

These user stories build upon the refined requirements, and I have directly used them within my product backlog, whilst maintaining the priority of each requirement:

REQUIREMENTS	MUST	SHOULD	COULD	WONT
Single page webpage application	🟧			
HTML5/CSS/JavaScript along with ASP.NET Core and a JavaScript SDK	🟧			
Staff authorisation	🟧			
Customer reservation booking	🟧			
Don't accept reservations with a social bubble of over 6 individuals	🟧			
Overbooking prevention	🟧			
Staff can amend venue details	🟧			
Desktop compatibility	🟧			
Functions at all times – 24/7	🟧			
Can support at least 10 people accessing it at the same time	🟧			
All hyperlinks must be fully functional	🟧			
User data should be encrypted and stored securely	🟧			
Customers cannot sign-in	🟧			
Record all staff working at the venue their shift times on a given day and their contact details	🟧			
Keep records of customers and staff for 21 days	🟧			
Identify company policies	🟧			
The system must adhere to the Companies Act 2006	🟧			
The system must adhere to the GDPR	🟧			
The system must adhere to the Equality Act 2010	🟧			
The customers need to be able to view their reservation		🟨		
The customers need to be able to cancel a reservation		🟨		
The customers need to be able to edit some of the reservation's details		🟨		
The staff need to be able to delete reservations		🟨		
The staff need to be able to add reservations		🟨		
The staff need to be able to view all current reservations		🟨		
The staff need to be able to edit venue details		🟨		
The staff need to be able to amend a reservation's details		🟨		
Must be a dynamic webpage		🟨		
Each request should process within under 7 seconds		🟨		
The website should provide users with information on how their information is used		🟨		
The website should be inclusive and provide features to provide greater customer satisfaction		🟨		
The system should produce a high-level monthly/weekly overview report			🟦	
The staff need to be able to see all shifts			🟦	
The customers need to be able to see a café menu			🟦	
Compatible with smartphone browsers			🟦	
If updates are to be applied this will occur during business downtime – at night			🟦	
Customer should receive SMS reservation confirmation within 10 minutes			🟦	
We need to contact customers with their reservation details			🟦	
The system will time-out after 10 minutes of inactivity			🟦	
Automated backup				🟩
Support for peripheral devices other than mouse or keyboard				🟩

(Requirements ordered by priority)

User Stories	Priority	Sprint	Status
As a staff member I want to be able to access the system on a desktop so that I can use mouse and keyboard to input data	Must		To be started
As a staff member I want to use keyboard and mouse so that I can be more efficient when typing/inputting data	Must		To be started
As a manager I want to be able to limit entry to the venue to only social bubbles of 6 or less so that I do not break the law	Must		To be started
As a manager I want to be able to only accept a certain total amount of individuals into the café so that I do not break the law	Must		To be started
As a manager I want to have a record of all staff and customer information so that I can contact them if a COVID outbreak is confirmed	Must		To be started
As a front-door staff member I want to be able to view customer contact information so that I can contact them if any urgent news needs to be conveyed.	Must		To be started
As a staff member I need to sign into the system so that I have permissions to look at business info and potentially amend it	Must		To be started
As a customer I want to book a reservation so that I can eat within Doki-Doki Delight's café	Must		To be started
As a front-door staff member I want to be able to add reservations to the system if a customer contacts the café over telephone so that I can provide exemplary customer service	Should		To be started
As a front-door staff member I want to be able to cancel/delete customer reservations on the system if a customer contacts me so that I can provide exemplary customer service	Should		To be started
As a customer I want to be able to be able to cancel my reservation because I have changed my mind	Should		To be started
As a customer I want to be able to amend my reservation because my schedule has changed	Should		To be started
As a customer I want to see how my information is used so that I can ensure my rights are not being violated	Should		To be started
As a staff member I need the system to automatically sign me out if I am inactive for more 10 minutes so that unauthorised access may be prevented	Could		To be started
As a customer I want to view my reservation details so that I can remind myself when the reservation is for	Could		To be started
As a customer I want to be able to view the café menu so that I can share it with my friends	Could		To be started
As a customer I want to be able to view the website on my smartphone's browser as it is my most readily available device.	Wont		-------------------

Further processing the user stories, I applied a development tool called Behaviour Driven Development (Hee, 2019) – of which depicts scenarios using the following framework:

Given…
When…
Then…

Title: customer books a reservation

As a customer
I want to book a reservation
so that I can eat within Doki-Doki Delight's café

Scenario 1: Venue has available seating/ dine-in space for the selected timeslot
Given the venue has less than max capacity
    And the reservation is for 6 or less individuals
    And the customer has input their full name and telephone number
    And the specific timeslot isn't already booked
When the customer clicks 'confirm reservation'
Then the website should return the reservation details

Scenario 2: Venue has no available seating/dine-in space at the selected timeslot
Given the timeslot is already reserved by another customer
When the customer clicks the date
Then the website should return 'this timeslot is already taken, please choose another'

Scenario 3: Customer does not input name or telephone number
Given the name or telephone number fields are empty
When the customer clicks 'confirm reservation'
Then the website should signal to fill in the empty input fields.

Title: webpage automatically disallows social bubbles of more than 6

As a manager
I want to be able to limit entry to the venue to only social bubbles of 6 or less
so that I do not break the law

Scenario 1: customer books a reservation with less than 6 attendees
Given that the attendees are 6 or less
When the customer clicks 'confirm reservation'
Then create a reservation

Scenario 2: customer books a reservation with more than 6 attendees
Given that the attendees are over 6
When the customer clicks 'confirm reservation'
Then return an error message saying 'social bubbles over 6 are not allowed'

Title: staff wants to sign into the system

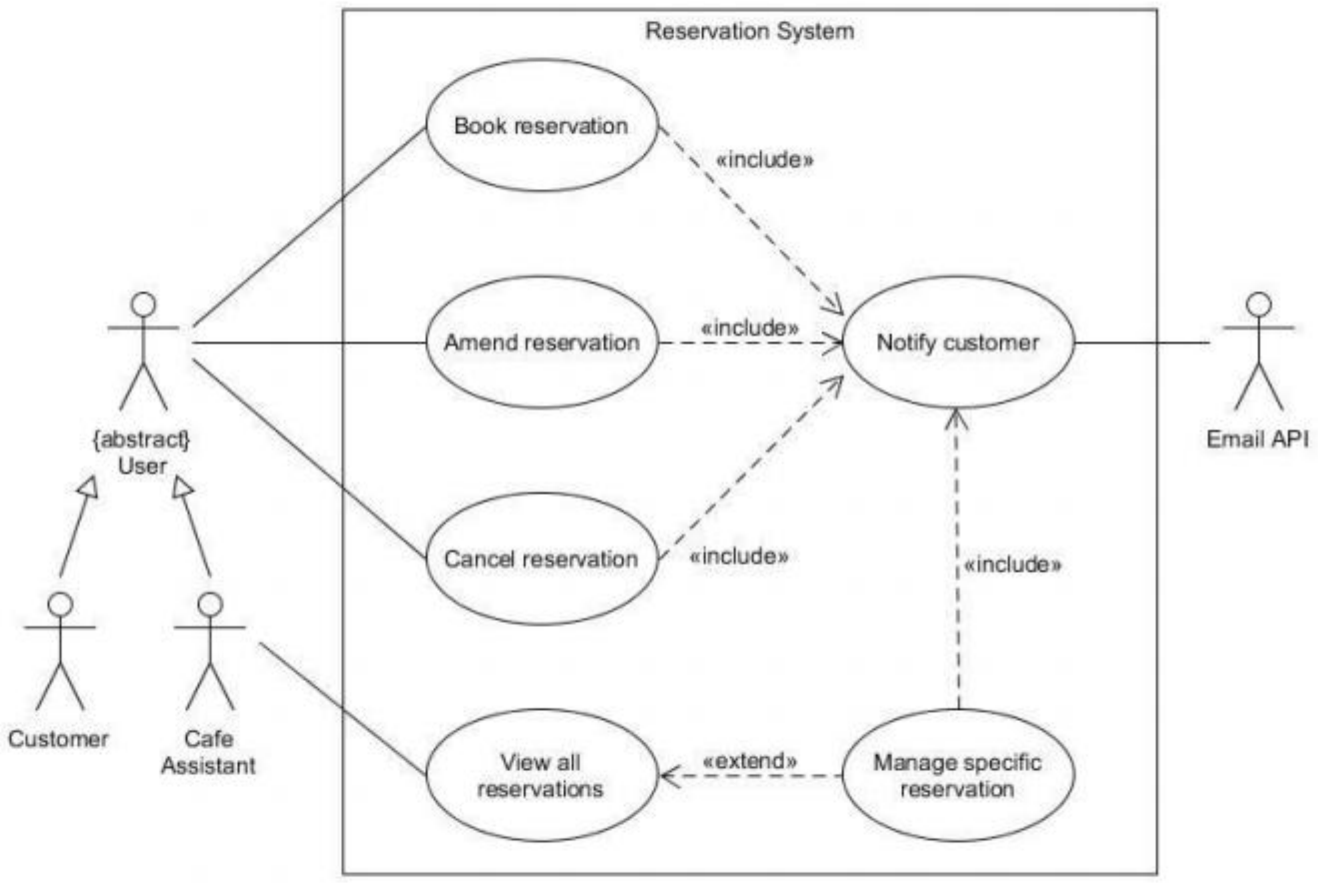
As a staff member
I need to sign into the system
so that I have permissions to look at business info and potentially amend it

Scenario: staff wants to sign in
Given that the staff member is not signed in
When they input a global username
    And password
Then give them authorised access to the system

(Some of many BDD illustrations)

Another tool that I have used at my disposal to further develop the User Stories was the infamous Use Case diagram. Using UML @ Classroom (Seidl et al., 2014) as my guide I converted my highest priority user stories into a visual format. I created a Use Case for each of the four critical sections of my project:

- Reservation booking
- COVID track and trace
- Venue management
- Notification system

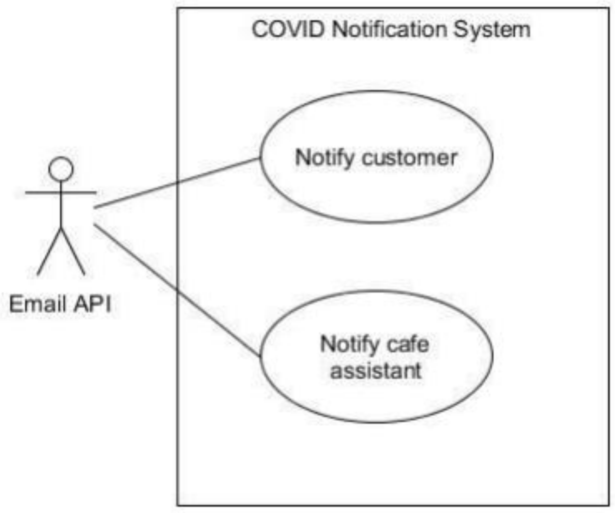


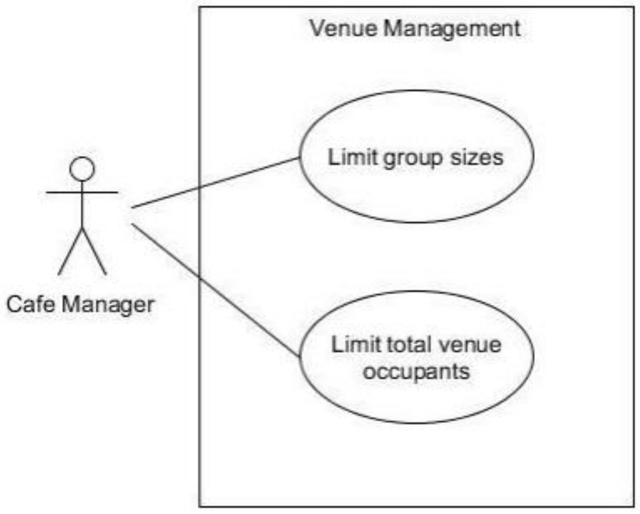
Name:	View all reservations
Short description:	A café assistant wants to view all reservations and perhaps make changes on behalf of a customer.
Precondition(s):	Café assistant is logged in to the system.
Postcondition(s):	Café assistant is viewing all placed reservations (high level overview)
Error 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	---

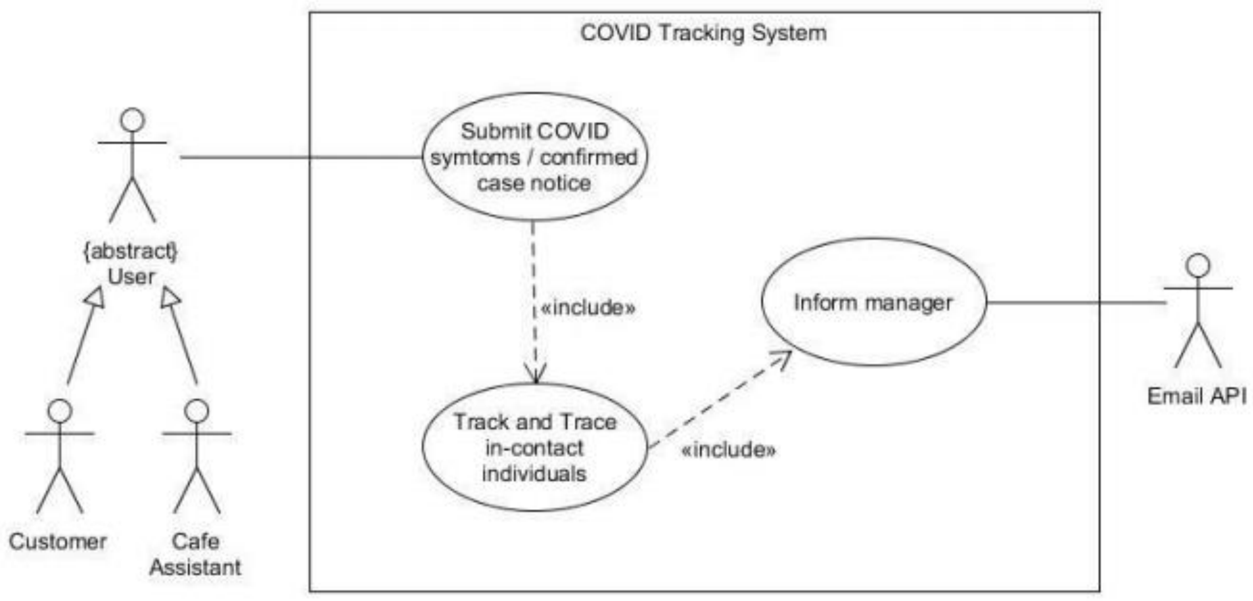
Name:	Book reservation
Short description:	A customer wants to create a reservation so they can dine-in at the venue. Can be done independently or through the customer assistant (both approaches utilise the website application)
Precondition(s):	Customer must have: full name, email address, telephone number, 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:	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

By using my user stories as templates for each Use Case diagram I can portray what each specific role within the system can accomplish by using the app. As stated within UML @ Classroom (Seidl et al., 2014) a Use Case diagram visually illustrates the answer to these questions:

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

Accompanying each diagram, the reader will find Use Case Descriptions, which intend to offer greater depth and understanding of the relevant diagram.

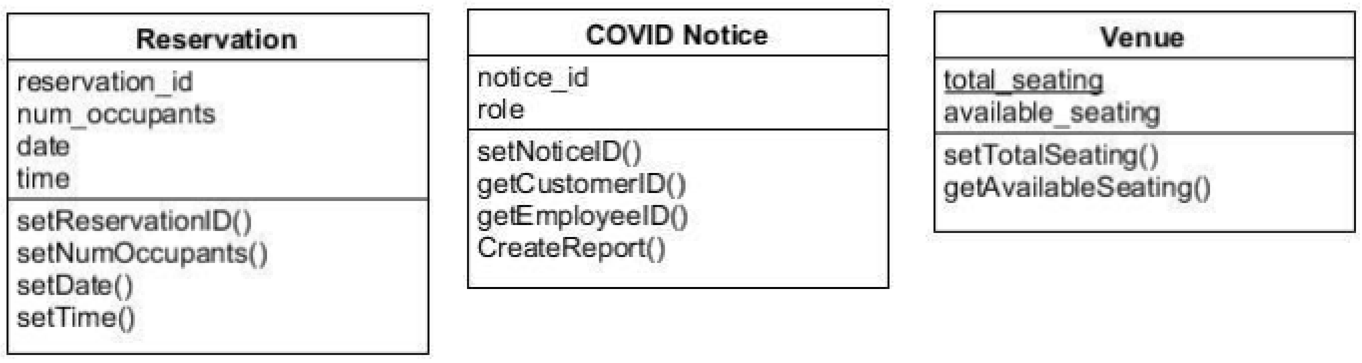
Architecture

As previously mentioned, when proceeding throughout the Design stage of my project I closely followed the book UML @ Classroom (Seidl et al., 2014) and this enabled me to understand and create the following diagrams:

- Class and Object
- State machine
- Sequence

I had decided to create class and object diagrams because I intend to implement my reservation booking system using the Object-Oriented Paradigm and to achieve this I will use C# through ASP.net. Using classes and objects I will hopefully be able to create a scalable system – and as I have greater experience with C# than JavaScript, I believe that I can implement a much more comprehensible system.

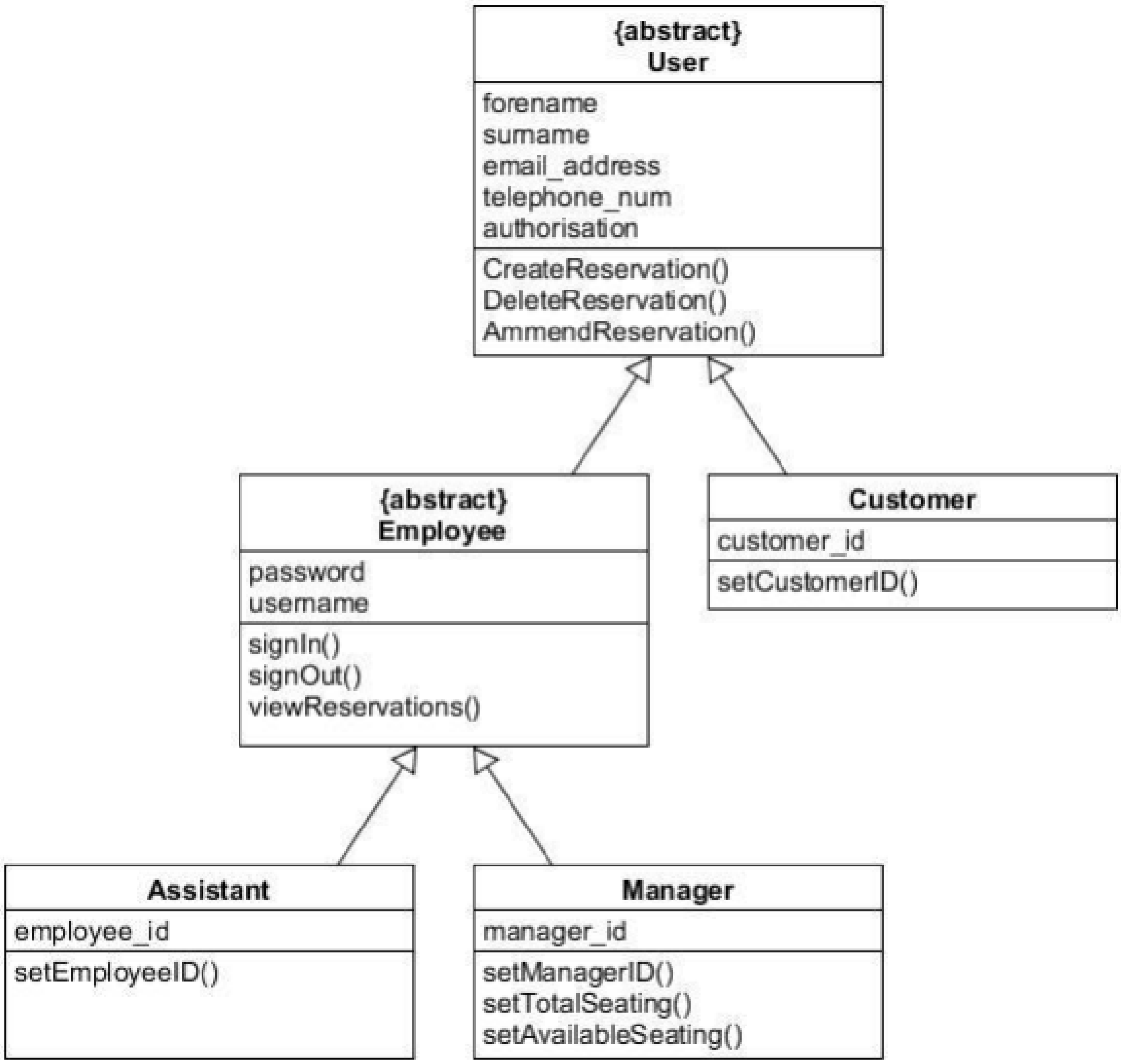
It is important to note that these diagrams are expected to be updated over time and are just setting a simple foundation of what the structure of my webapp code should look like. For example, these diagrams may be refined to be more practical as feedback is received by stakeholders via testing.



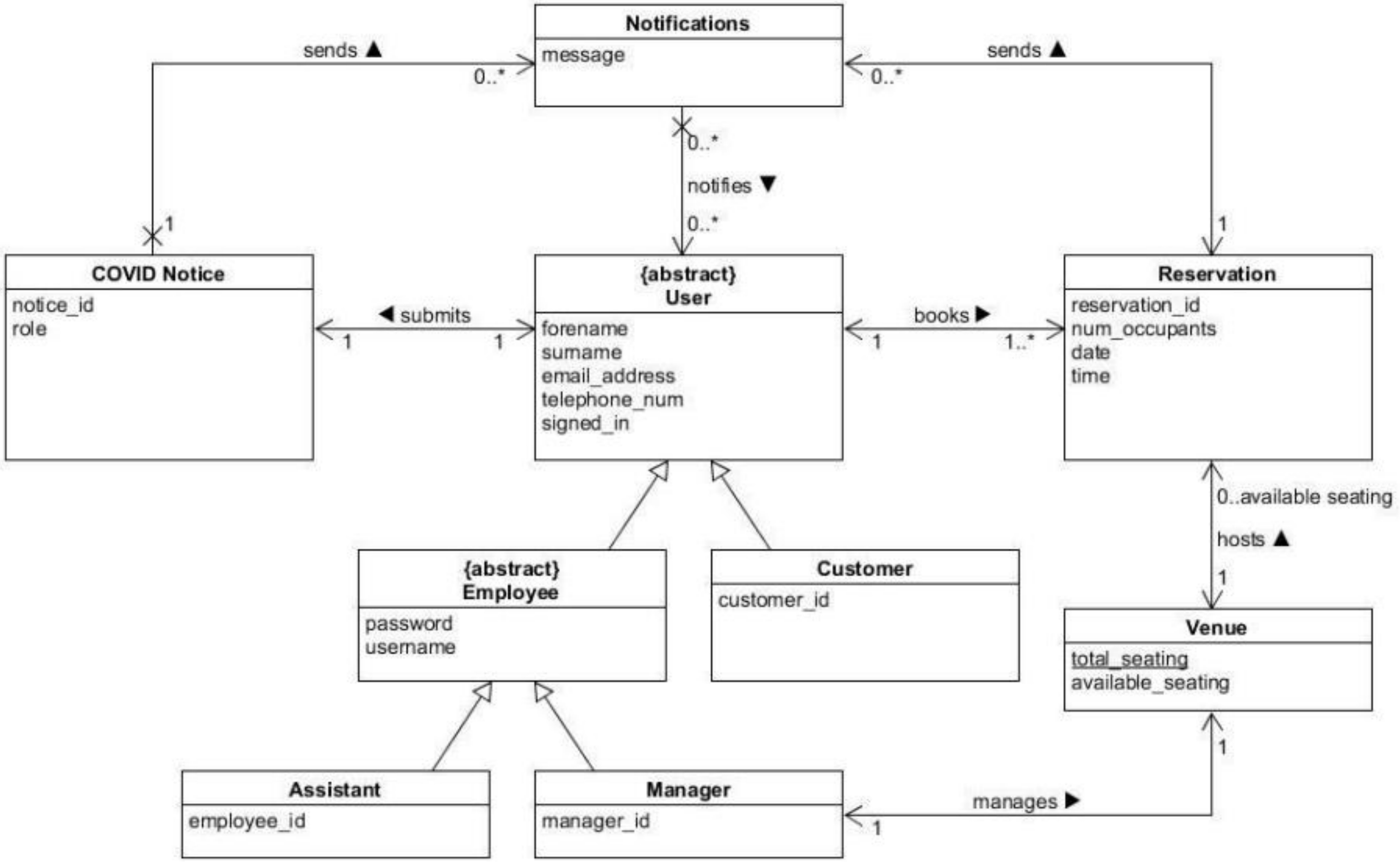
In some instances, classes will have ID attributes so that they can be uniquely identified, and this will eventually be used to locate specific objects – such as a customer via their reservation ID.

Alongside each class's attributes there are also methods, some of which are used to fetch or amend and store data. For example, the manager will have permission to update the venue's total seating as per the setTotalSeating() method.

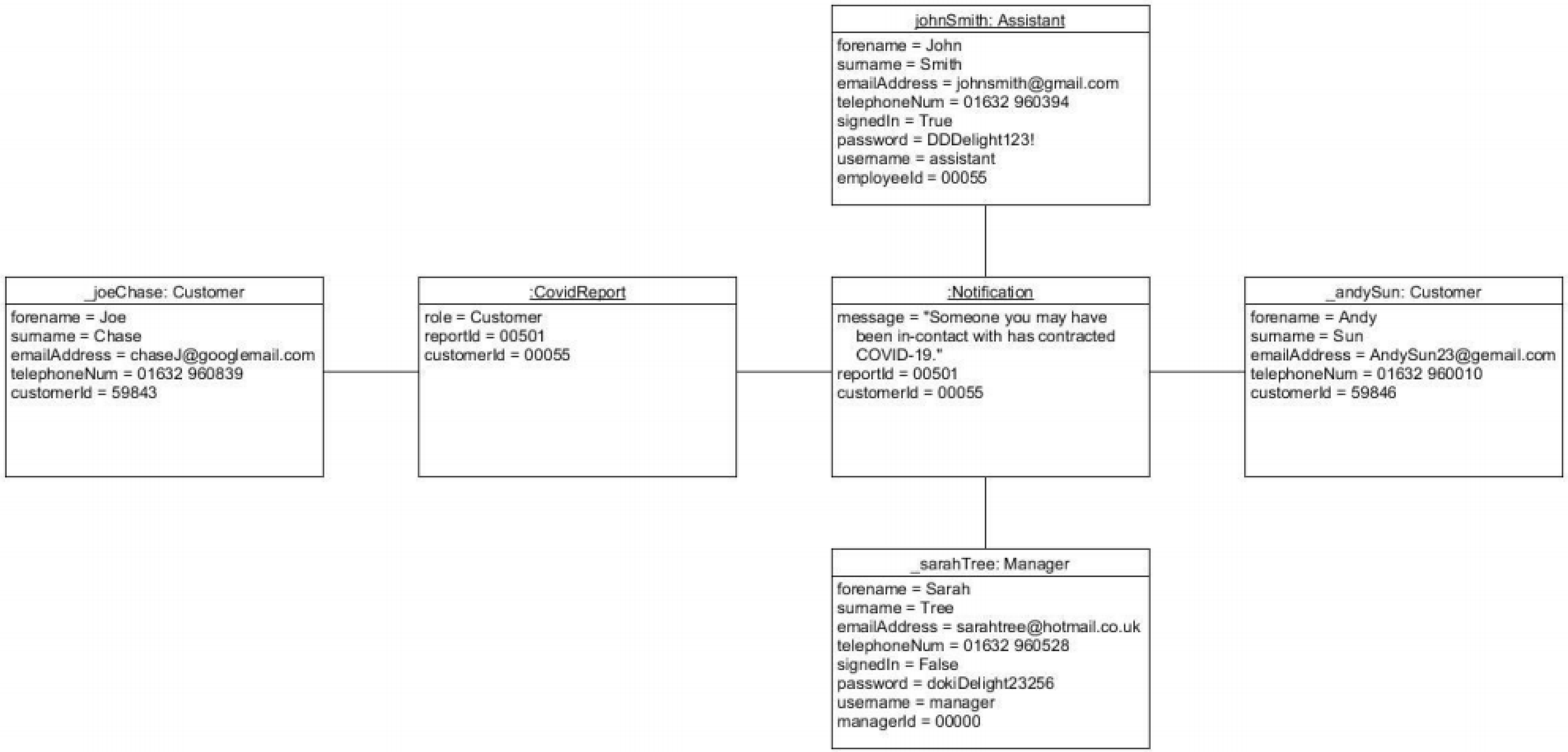
Within this next diagram, inheritance between the various user classes is shown. There is an overall, super class called “user”, of which all types of users will inherit from, but as there are two types of employee’s I have further implemented another super class that has two inheriting subclasses – assistant and manager. From observing the diagram the reader can observe that the manager has an additional set of activities that they can carry out via the webapp – namely, venue management.



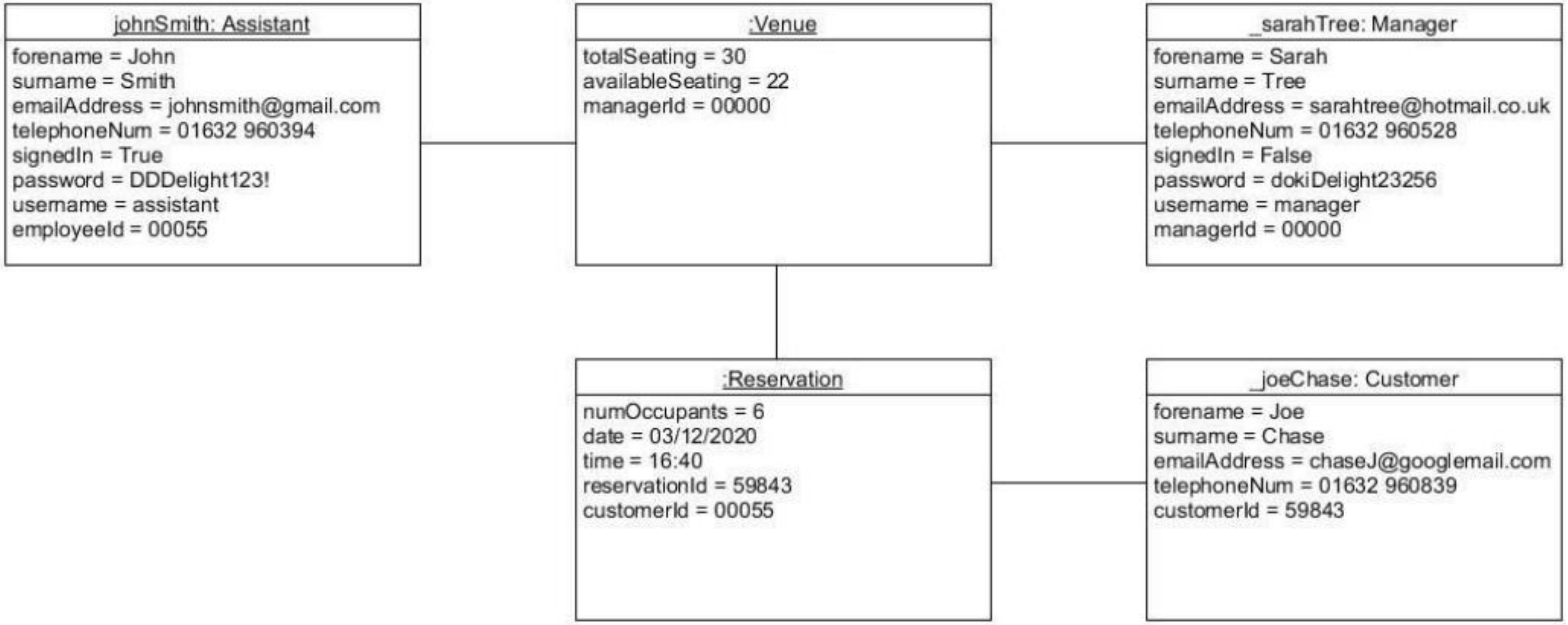
This diagram conveys a detailed structure of the relationships between each class and its interactions with other classes:



Furthermore, I have made additional artifacts to show the associations between each class within a given context:



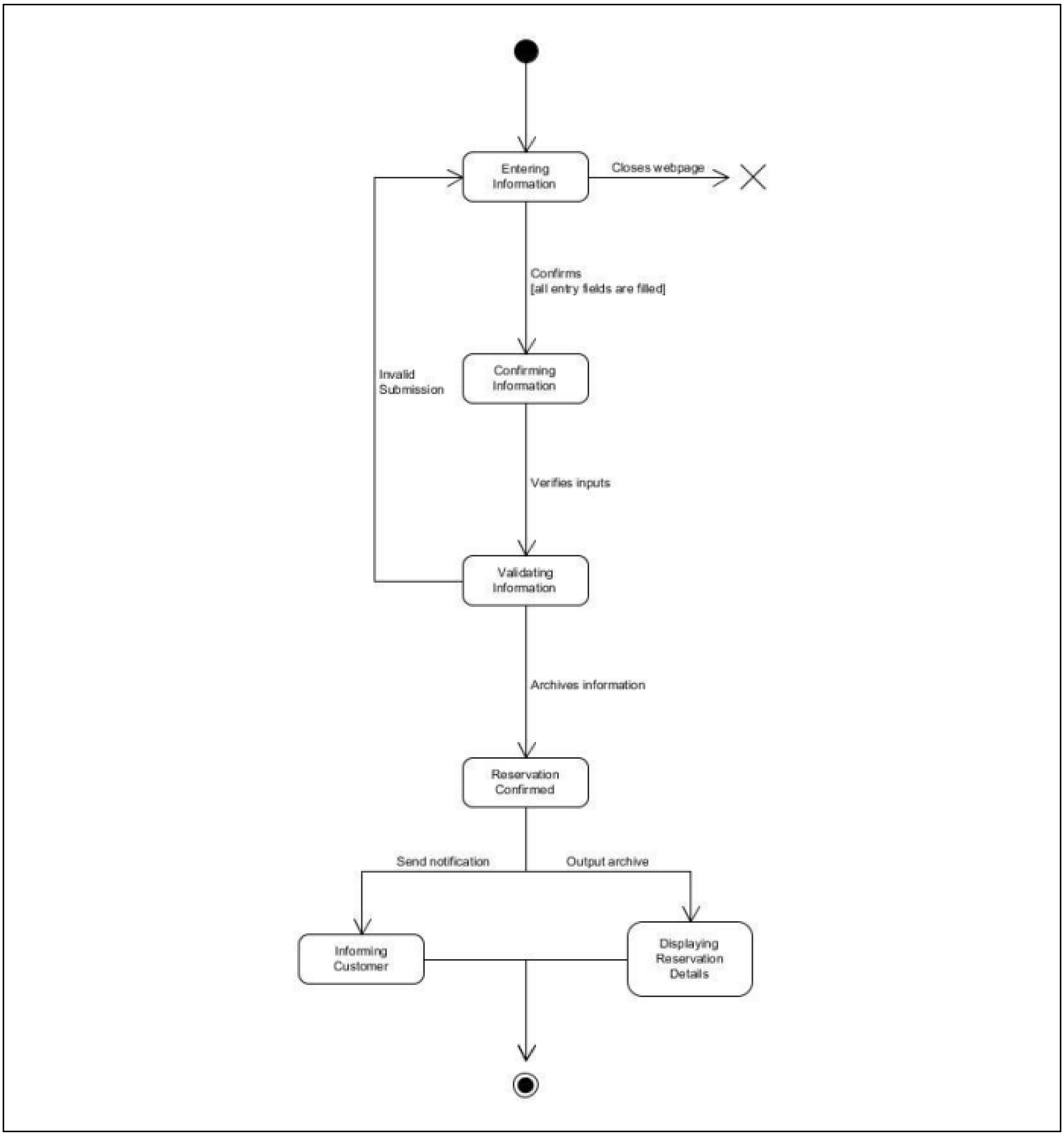
Scenario: A customer submits a COVID report and therefore notifications are sent to a collection of individuals that may have been in-contact with them. Details of implementing this tracing algorithm have been abstracted for simplicity.



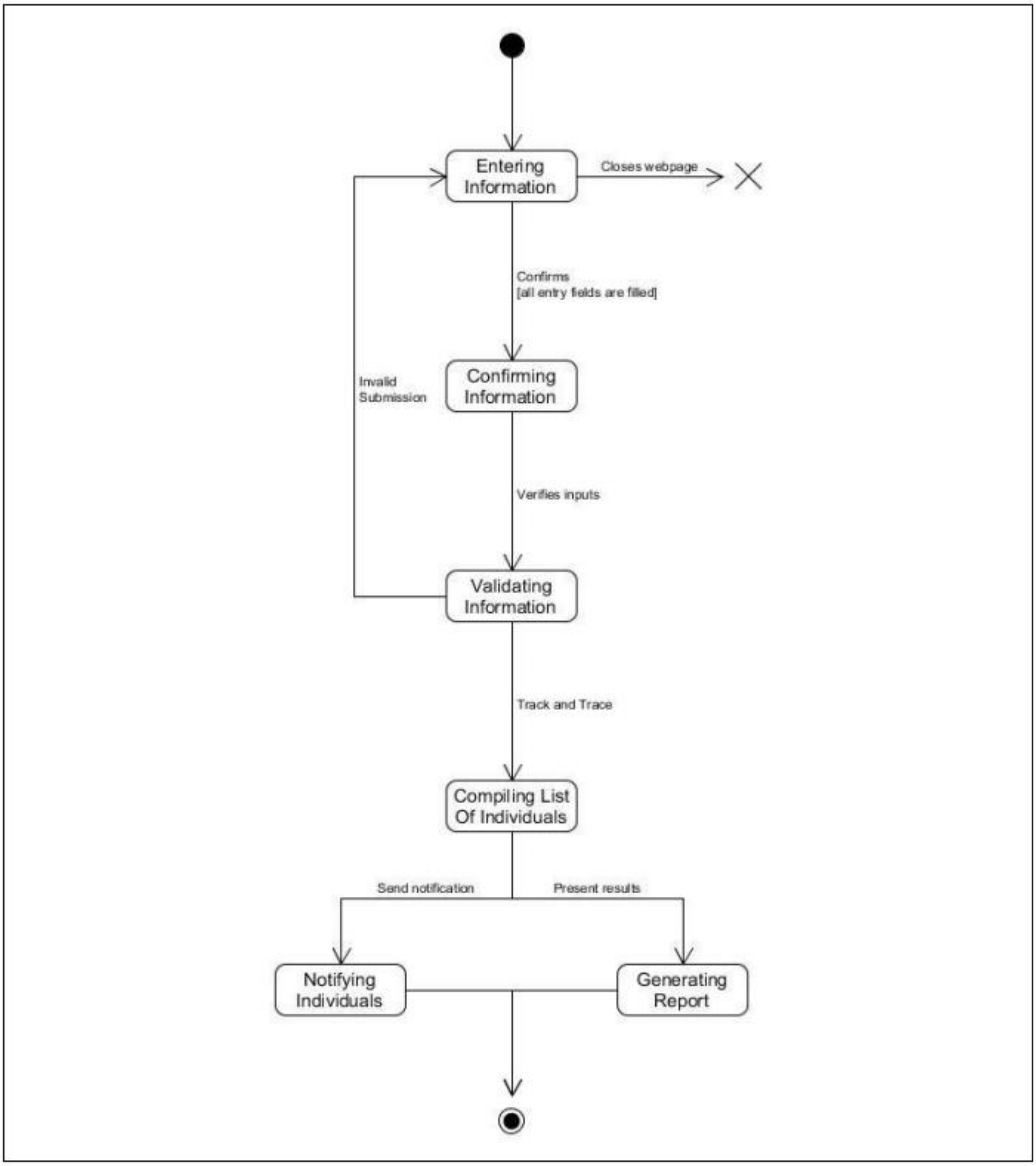
Scenario: A reservation can be booked by a Customer, and the Reservation object requires a Venue to be created – the venue will limit the number of reservations being created. A venue will be run by the business's Manager and Assistant. All users must input their personal information so that they can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

My state machine diagrams convey the responses and behaviours of my system when a trigger is activated.

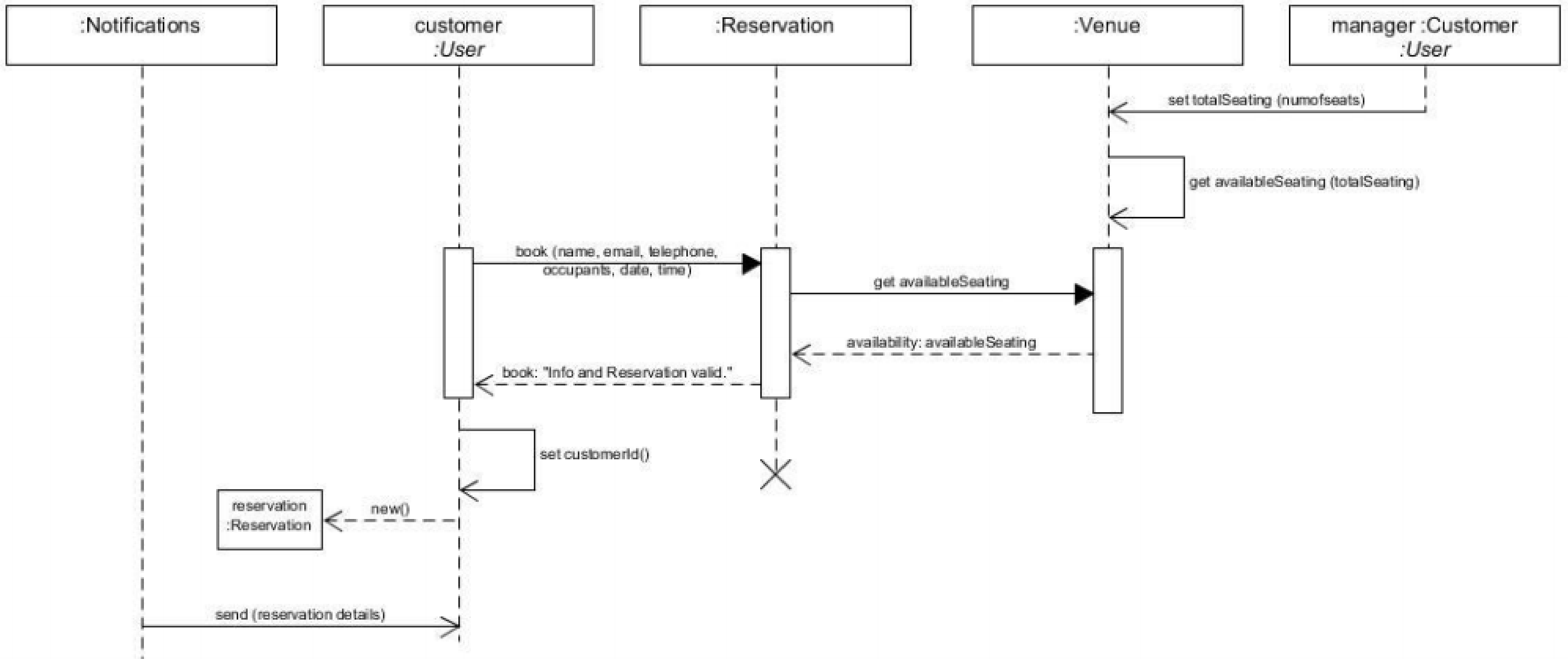
Reservation System:



COVID report system:



Finally, I have a single Sequence diagram that illustrates the event of a customer making a reservation:


: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

Sprint Planning

As previously seen within ***Requirements*** (page 6 of this report) I have formed a product backlog, and have given it the properties: sprint, priority, and status. Based on these fields, I have imported my backlog into Microsoft Planner so that I can easily manage what I need to do each sprint.

Microsoft Planner link:
https://tasks.office.com/live.plymouth.ac.uk/Home/PlanViews/KB8DrzSpokCUWjx6hWH765YACcBq?Type=PlanLink&Channel=Link&CreatedTime=637468573538540000

However, as of writing this report I have not gotten far into the sprints as most of the project has just been planning and designing so far. Sprints are intended to be iterated throughout on a 2-week basis, wherein a review of the deliverables produced will be conducted – and if required, the next sprint may be designated to the same backlog feature.

My planner is laid out using the Kanban structure, wherein the features are stored on the left of the board and as they are worked on, they are adjusted rightwards, until they reach the end and are marked as completed. Going from left to right, my planner is structured like this:

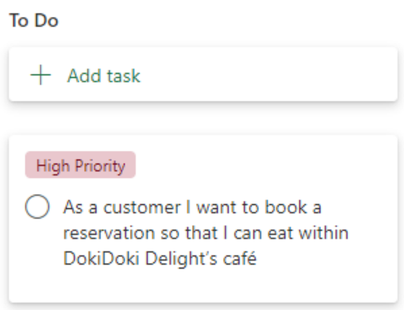
- Product backlog
- To do
- In progress
- Done

My product backlog is formed of user stories and they are 'tagged' with priorities that relate to the previous MoSCoW processing. For additional readability I simply renamed each tier within MoSCoW to: high, medium, and low. I intentionally left out the wont-have features as I wanted to keep the planner very simple.

At the start of a sprint, I will move a manageable/achievable amount of user stories to the "To-do" column. I will then move the current task I am working on to the "In progress" column and when the feature has been implemented it will be moved onto "Done" and reviewed before being marked as complete. If possible, I will refer back to my BDD artifacts to test/review each user story and if it fulfils the criteria then I know the feature has been implemented successfully. A new user story will then be fetched from To-do and this will repeat until the sprint is over.

I am currently also maintaining a weekly blog post on the DLE wherein I retrospectively explore how proactive I have been that week and list all that I have achieved. This is an important part of the sprints as Agile is about maintaining focus and recognizing weaknesses within a team so that they can be overcome and not inhibit the production of deliverables.

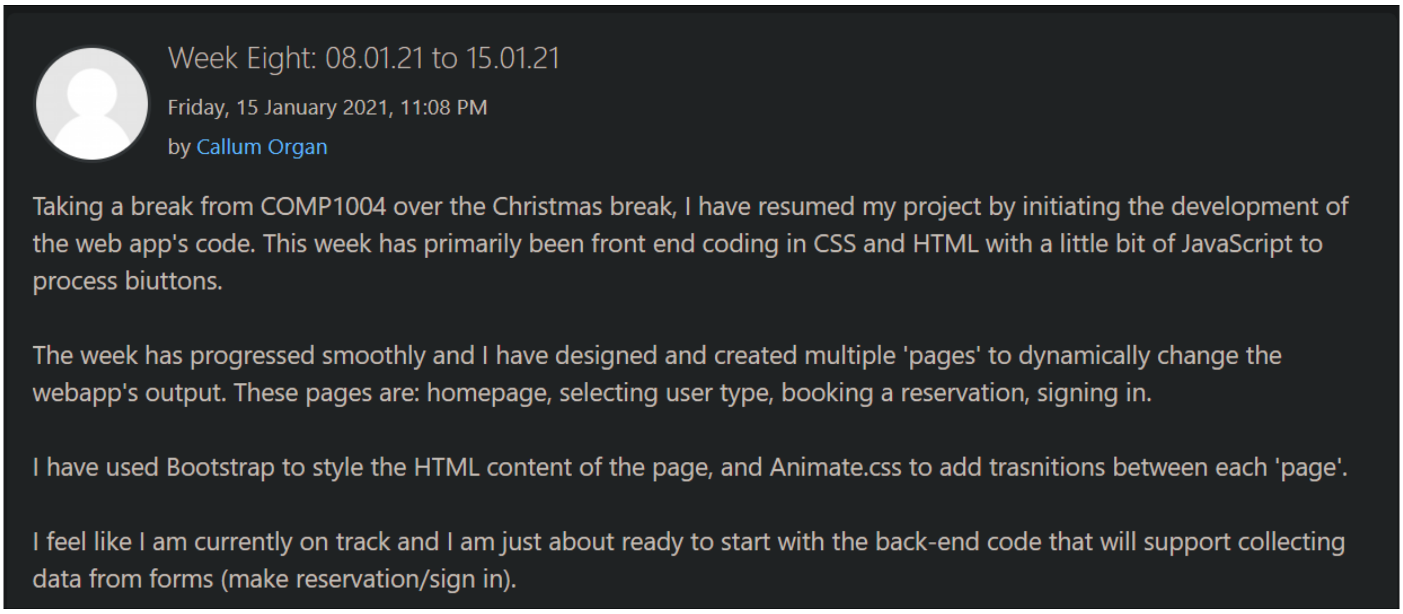
The backlog feature that I am intending to work on is the implementation of the reservation booking system, as shown here:



As a customer I want to book a reservation so that I can eat within DokiDoki Delight's café

As this user story is "high priority" I will most likely be spending the whole sprint implementing it within the webapp.

Overall, sprints are a basic way of managing the deliverables being produced and identifying if production is too slow – and the reviews work alongside this as they will help me understand the cause of this. For example, my latest review (start of implementation) is as follows:



Week Eight: 08.01.21 to 15.01.21

Friday, 15 January 2021, 11:08 PM

by Callum Organ

Taking a break from COMP1004 over the Christmas break, I have resumed my project by initiating the development of the web app's code. This week has primarily been front end coding in CSS and HTML with a little bit of JavaScript to process biuttons.

The week has progressed smoothly and I have designed and created multiple 'pages' to dynamically change the webapp's output. These pages are: homepage, selecting user type, booking a reservation, signing in.

I have used Bootstrap to style the HTML content of the page, and Animate.css to add trasnitions between each 'page'.

I feel like I am currently on track and I am just about ready to start with the back-end code that will support collecting data from forms (make reservation/sign in).

Reflection

Up until the time of writing I can confidently say that I have a professional pitch and, for a project being individually developed by myself, it is definitely manageable and not out of scale. A long portion of my time has so far been spent on the design and planning phases, but I have recently started implementing the front-end of my webapp.

Over the next few months of the project, I can see myself diving into the back-end code a lot more and making the website functional. I am aiming to fully implement the highest priority features within my backlog first and then work through the medium priority items, until I reach the lowest priority features. I do however realise that not all features may be implemented – it is only the highest priority ones that must definitely be created to ensure that the project is successful and is usable.

I am hoping to gain some really good advice on my ambitions towards the project and the user interface from the upcoming marketplace demo and will try to iteratively add the suggested changes to my project so that it becomes intuitive and easy to use for the user.

Overall, the project has really just started, and it is crucial that I maintain the planner and work inside of sprints. One change that I should make when approaching the project in the future is to reach out to others to gain as much feedback as possible by letting them interact with the webapp – this way of testing will help me collect a valid range of outsider opinions and Agile is about effective communication. I am aiming for an intuitive and interactable webapp that is easy to use by a range of target audiences and therefore it needs to be thoroughly critiqued and adjusted via iterative development procedures so that it can become a usable piece of software.

Bibliography

Cox, A., 2017. Business Requirements Vs Functional Requirements? Who Cares?. [online] Netmind. Available at: <https://netmind.net/business-vs-functional-requirements-who-cares/> [Accessed 7 November 2020].

Eriksson, U., 2012. Functional Vs Non-Functional Requirements - Understand The Difference. [online] ReQtest. Available at: <https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/> [Accessed 7 November 2020].

Etemad-Sajadi, R., 2014. The influence of a virtual agent on web-users' desire to visit the company. International Journal of Quality & Reliability Management, 31(4), pp.419-434.

GOV.UK. 2020. Guidance For Food Businesses On Coronavirus (COVID-19). [online] Available at: <https://www.gov.uk/government/publications/covid-19-guidance-for-food-businesses/guidance-for-food-businesses-on-coronavirus-covid-19> [Accessed 7 November 2020].

Hee, D., 2019. Applying BDD Acceptance Criteria In User Stories. [online] ThoughtWorks. Available at: <https://www.thoughtworks.com/insights/blog/applying-bdd-acceptance-criteria-user-stories> [Accessed 8 November 2020].

Martin, R. and Martin, M., 2006. Agile Principles, Patterns, And Practices In C. Prentice Hall.

North, D., n.d. What'S In A Story?. [online] Dan North & Associates. Available at: <https://dannorth.net/whats-in-a-story/> [Accessed 8 November 2020].

Scott, A., n.d. Ethical Web Development. [online] Ethicalweb.org. Available at: <https://ethicalweb.org/> [Accessed 7 November 2020].

Seidl, M., Scholz, M., Huemer, C. and Kappel, G., 2015. UML @ Classroom. Springer.