COMP1004

Computing Practice

2020/2021

Project Title

Doki-Doki Delight Management System

GitHub Repository:	https://github.com/ORG4N/doki-doki-delight-management-system

Contents

Introduction

This report aims to outline the entire development process of the Single Page Web-Application (SPA) that has been created under the pretext of the business: Doki-Doki Delight, a fictional cafe.

An SPA is an online web application that, instead of reloading entire webpages, dynamically updates a single page's content in response to user actions. The overall goal of creating an SPA, rather than a traditional website, is to provide a richer user experience.

This aspect of an SPA is significant to Doki-Doki Delight as they require their web-app to be user friendly and provide a welcoming, easy to use service which enables customers to create reservations and do further research into the business.

To coherently document the SPA project, the report will be segmented into the following seven components:

- Software Development Lifecycle
- Project description
- Requirements
- Architecture
- Sprint planning
- Implementation
- Reflection

Within each section of this report, the reader will be shown justifications behind the manner in which the SPA project has been carried out, and by the end of the document, the reader will have a thorough understanding of the need for this SPA and how it has been created to be 'successful' in accordance with Doki-Doki Delight's 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 (W. Royce, 2021).

Since then, the digital world has progressed with the creation of new models, 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 (McConnell, 2014). These are just a few types of successes that a developer may hope to achieve – but not all existing methodologies 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 (W. Royce, 2021).

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. Models generally include the following five stages:

- Requirement Analysis
- Design
- Implementation
- Testing
- Evolution

However, some models may extend beyond five stages – furthermore, each model executes each stage in a different sequence from another model.

The *Doki-Doki Delight Management System* is structured around the Agile framework. Agile is represented by 12 principles (Principles behind the Agile Manifesto, 2021) 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 "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 influenced the direction of the project from start to finish was the concept of retrospection. In Agile, the current morale of the team an important factor that a team leader has to consider when coming to decisions (Principles behind the Agile Manifesto, 2021). 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. By reflecting on the present struggles and developments throughout the project, through writing weekly blog posts, issues we are made known and then able to be worked upon.

Use of the Agile methodology was applied by examining the project's requirements and then extracting a project backlog from these requirements. From the project backlog, tasks were further decomposed into weekly sprints, which are sets of tasks that should be completed within a set time period (Scrum Guide, 2021). For the first half of the project, outside guidance was sought from bi-weekly meetings with Liz Stewart, who can be interpreted as a project sponsor. Further criticism was received from other staff within the Marketplace Demo where the developer of the app showcased a formal prototype of the app.

Through analysing and applying this feedback to the project, the final deliverable has been constructively built upon to make a useable piece of software that is in line with what real clients would expect. This continuous development of deliverables is in line with what is expected from a project built upon an Agile framework.

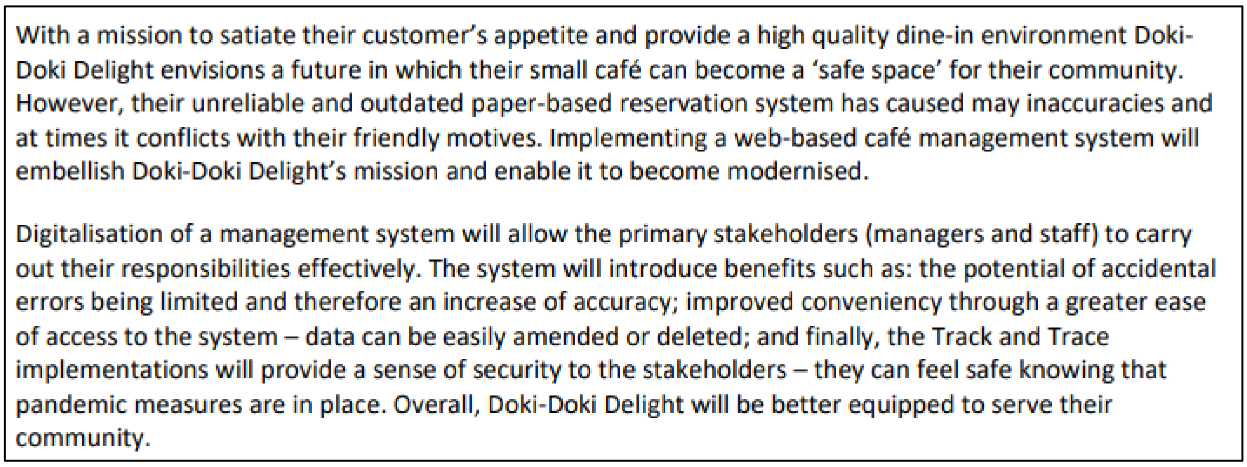
Project Description

To start this project, a lot of beforehand planning had to be carried out. As such, the following documents were created:

- Project Proposal Document (PID)
- Project Initiation Document

The latter document establishes the vision of Doki-Doki Delight, whilst the former translates this vision into a set of meaningful goals that this project should meet so that the vision can be realised.

Doki-Doki Delight's vision is as follows (taken from the PID):



With a mission to satiate their customer's appetite and provide a high quality dine-in environment Doki-Doki Delight envisions a future in which their small café can become a 'safe space' for their community. However, their unreliable and outdated paper-based reservation system has caused may inaccuracies and at times it conflicts with their friendly motives. Implementing a web-based café management system will embellish Doki-Doki Delight's mission and enable it to become modernised.

Digitalisation of a management system will allow the primary stakeholders (managers and staff) to carry out their responsibilities effectively. The system will introduce benefits such as: the potential of accidental errors being limited and therefore an increase of accuracy; improved conveniency through a greater ease of access to the system – data can be easily amended or deleted; and finally, the Track and Trace implementations will provide a sense of security to the stakeholders – they can feel safe knowing that pandemic measures are in place. Overall, Doki-Doki Delight will be better equipped to serve their community.

Furthermore, additional activities, such as research into requirements and potential issues, formed the basis of Requirements (see page 6). The final stage of planning included designing the structure of the software by creating various UML diagrams. Halfway through the project, these artifacts were reviewed by outside sources and feedback was relayed to the developer of the project, enabling them to reflect on the direction of the project. The project was further reviewed within a Marketplace Demo – allowing the project's scope to grow as further additional requirements and potential issues were highlighted.

The difference between a 'good' café and a 'bad' café can ultimately be drawn to the degree of customer satisfaction that the café provides. As a business it is their duty to fulfil their customer's needs and provide a quality service. Customer satisfaction is a concept that a business can control and maintain by ensuring their services are intuitive and easy to use/access.

Through developing a web-application for Doki-Doki Delight they will be able to grow their business. The app will enable customers to book reservations, of which staff can further manage. Also, as a result of the ongoing pandemic, the application requires Track and Trace functionality to alert customers when they have come into contact with another individual reporting COVID symptoms. The application will also have to be tailored towards the current pandemic regulations (which are constantly changing) to ensure 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 following legislation are significant in relation to developing the web-app:

- 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 application should therefore maintain inclusivity for a variety of different target audiences.

In 2011 the World Health Organisation estimated that near 15% of the human population has some sort of disability (World health organisation, 2011). Within this subset of the population, there are a wide range of disabilities that exit, but the ones that are most significant to consider within this project are mobility and visual disabilities. Without any sort of assistance, these stakeholders are prevented from accessing the service (Schmutz, 2016) as they have no way of interacting with the application.

There are four primary principles in which webpages can be improved for disabled users (Caldwell, 2008):

- Perceivable
- Operable
- Understandable
- Robust

Caldwell's guidelines suggest that different characteristics of the page can be changed to display content so that it is easier to view, but ways in which users interact with functionality can be different – such as by "making functionality available from a keyboard".

Overall, the project has been designed help market Doki-Doki Delight and make it easier for customers to create reservations, and whilst developing a solution the other considerations and issues that have been mentioned have not been ignored. Extra attention has been bought to them via the Proposal and Initiation documents and the legalities have been researched. Overall, the core values of usability and inclusivity (Scott, n.d.) have been significant when developing the project and this is reflected within the Requirements (page 6).

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 its own fabrication, the scale of the café management webapp was singlehandedly evaluated by the developer. This section will present an analysis of the project’s requirements.

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 determine that specific stakeholders should be able to carry out specific tasks (Cox, 2017) – but ‘how’ they might carry out the tasks was intentionally abstracted as this unnecessarily restricts and complicates the analysis phase.

However, the following non-functional requirement classifications (Eriksson, 2012) opened a deeper exploration of the project in terms of statistics, but also made outlining the consideration of external forces – such as laws and regulations.

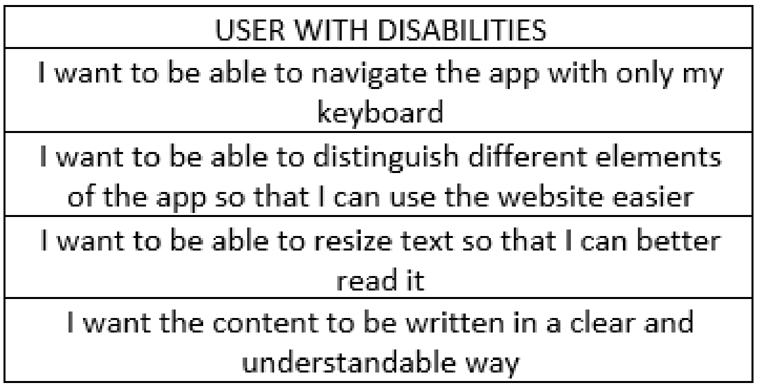
- Usability
- Security
- Readability
- Social
- Availability
- Ethical
- Performance

At this stage, there were numerous unrefined requirements that were not in any sort of format. To further extend the analysis of my requirements it was mandatory that they were given structure, and therefore resulting in the usage of 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 the requirements were refined and ordered – making it much easier to form the user stories.

User stories are concise explanations of the main functionality required from the main users’ perspective (North, n.d.). Determining that the webapp will have two types of users – staff and customers, allowed the creation of stakeholder ‘profiles’. The ‘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.

A final, more ambiguous profile has also been created for users with disabilities and it contains some general concerns/needs that a range of different impaired users might require consideration of within the application.



USER WITH DISABILITIES
I want to be able to navigate the app with only my keyboard
I want to be able to distinguish different elements of the app so that I can use the website easier
I want to be able to resize text so that I can better read it
I want the content to be written in a clear and understandable way

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 have been directly used within the 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	🟧			
Information or instructions should be written in a clear and understandable way	🟧			
Use colour to distinguish different elements of the app	🟧			
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 allow users to navigate using only a keyboard		🟨		
The website should be resizable (text, images)		🟨		
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 disabled user, I want to be able to navigate the app with only my keyboard	Should		To be started
As a disabled user, I want to be able to resize text so that I can better read it	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, a development tool called Behaviour Driven Development (Hee, 2019) was applied – 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

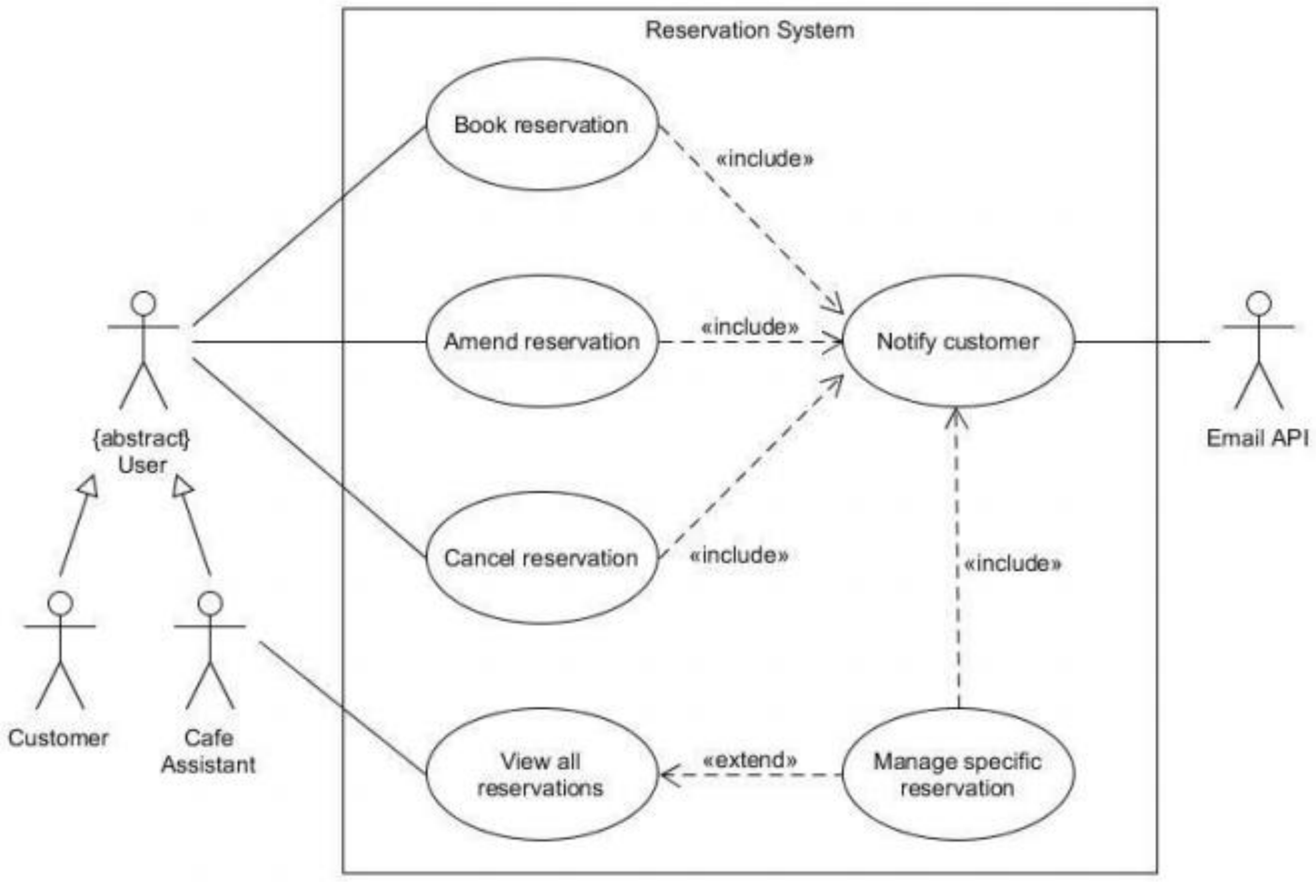
As a staff member
I need to sign into the system
so that I have permissions to look at business info and potentially amend it

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 has been used within this project to further develop the User Stories was the infamous Use Case diagram. Using UML @ Classroom (Seidl et al., 2014) as a guide, the highest priority user stories were converted into a visual format. A Use Case was created for each of the four critical sections of my project:

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

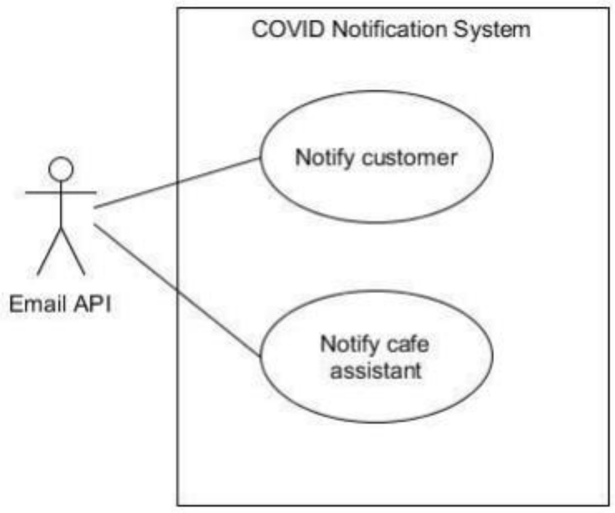


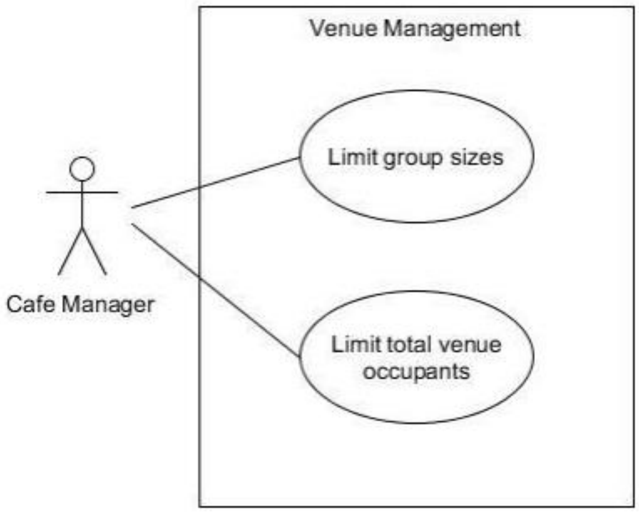
Name:	View all reservations
Short description:	A café assistant wants to view all reservations and perhaps make changes on behalf of a customer.
Precondition(s):	Café assistant is logged in to the system.
Postcondition(s):	Café assistant is viewing all placed reservations (high level overview)
Error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

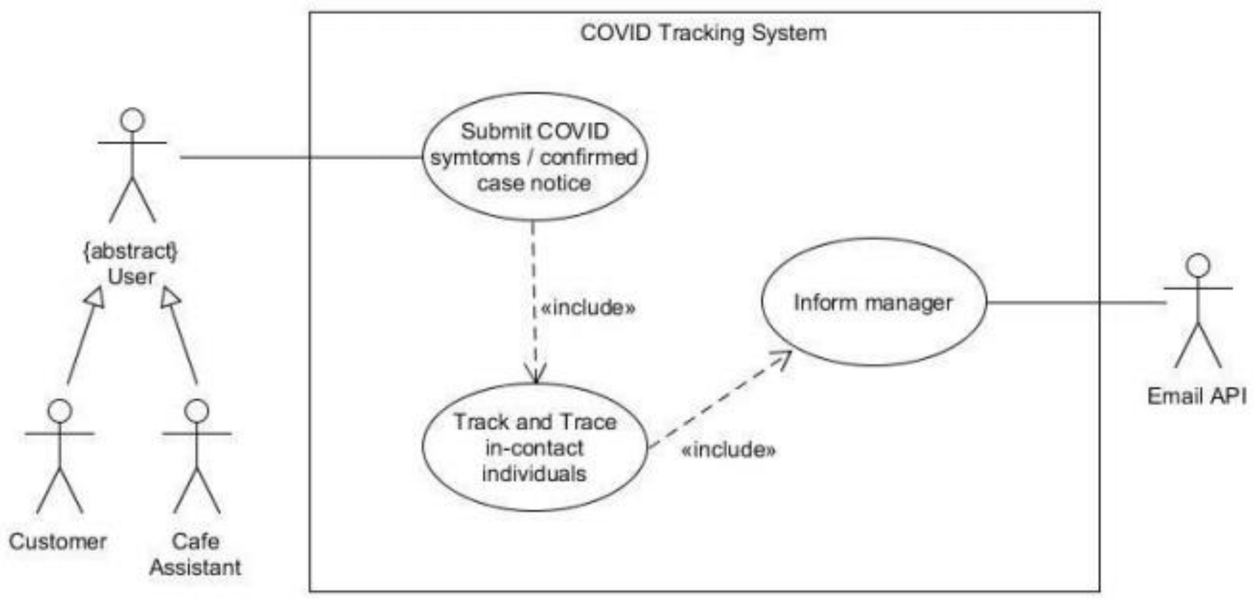
Name:	Book reservation
Short description:	A customer wants to create a reservation so they can dine-in at the venue. Can be done independently or through the customer assistant (both approaches utilise the website application)
Precondition(s):	Customer must have: full name, email address, telephone number, 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 user stories as templates for each Use Case diagram there is a portrayal of 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.

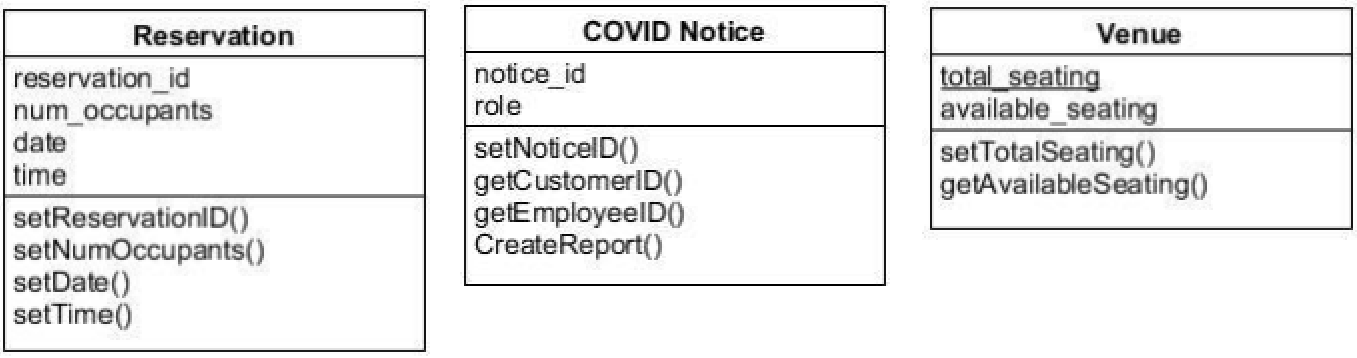
Architecture

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

- Class and Object
- State machine
- Sequence

The developer had decided to create class and object diagrams because it was intended from the start of the project to implement the reservation booking system using the Object-Oriented Paradigm and to achieve this they would use C# through ASP.net. By using classes and objects, the hopeful outcome would be a scalable system – and as the developer has greater experience with C# than JavaScript, they could implement a much more comprehensible system.

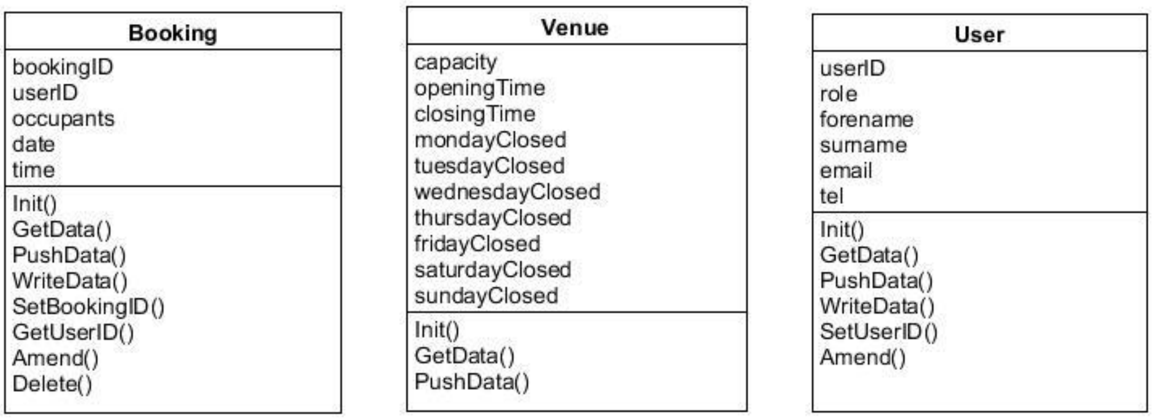
These were the initial diagrams that were created before coding the final project, and although they are similar to how the final system has been implemented, they are outdated. These diagrams served as a foundation, but as coding was started, it was realised that these objects could be simplified much more.



Within the initial diagrams it was notable that in some instances, classes would have ID attributes so that they can be uniquely identified, and this would eventually be used to locate specific objects – such as a customer via their reservation ID.

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

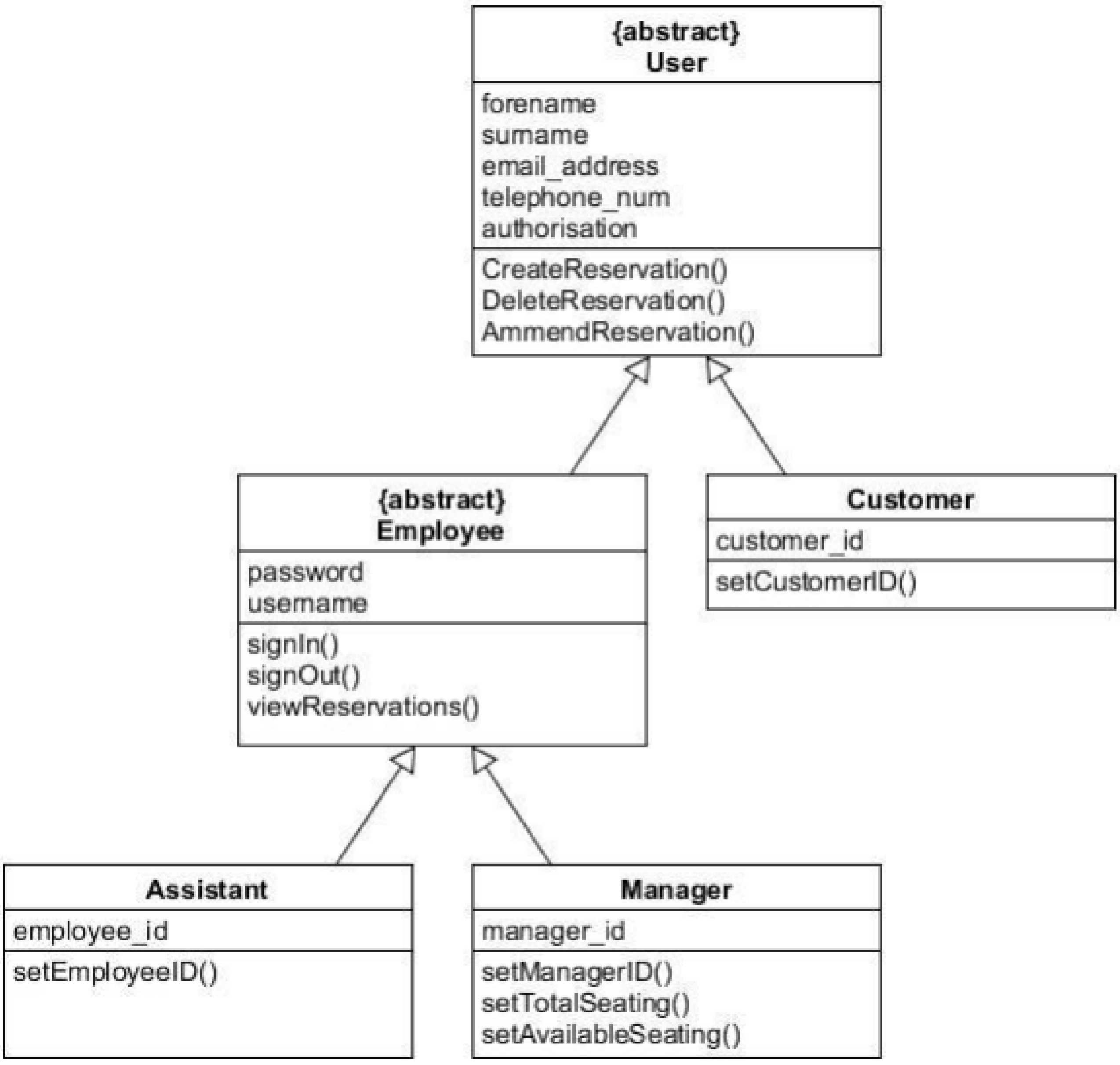
The updated class and object diagrams look like this:



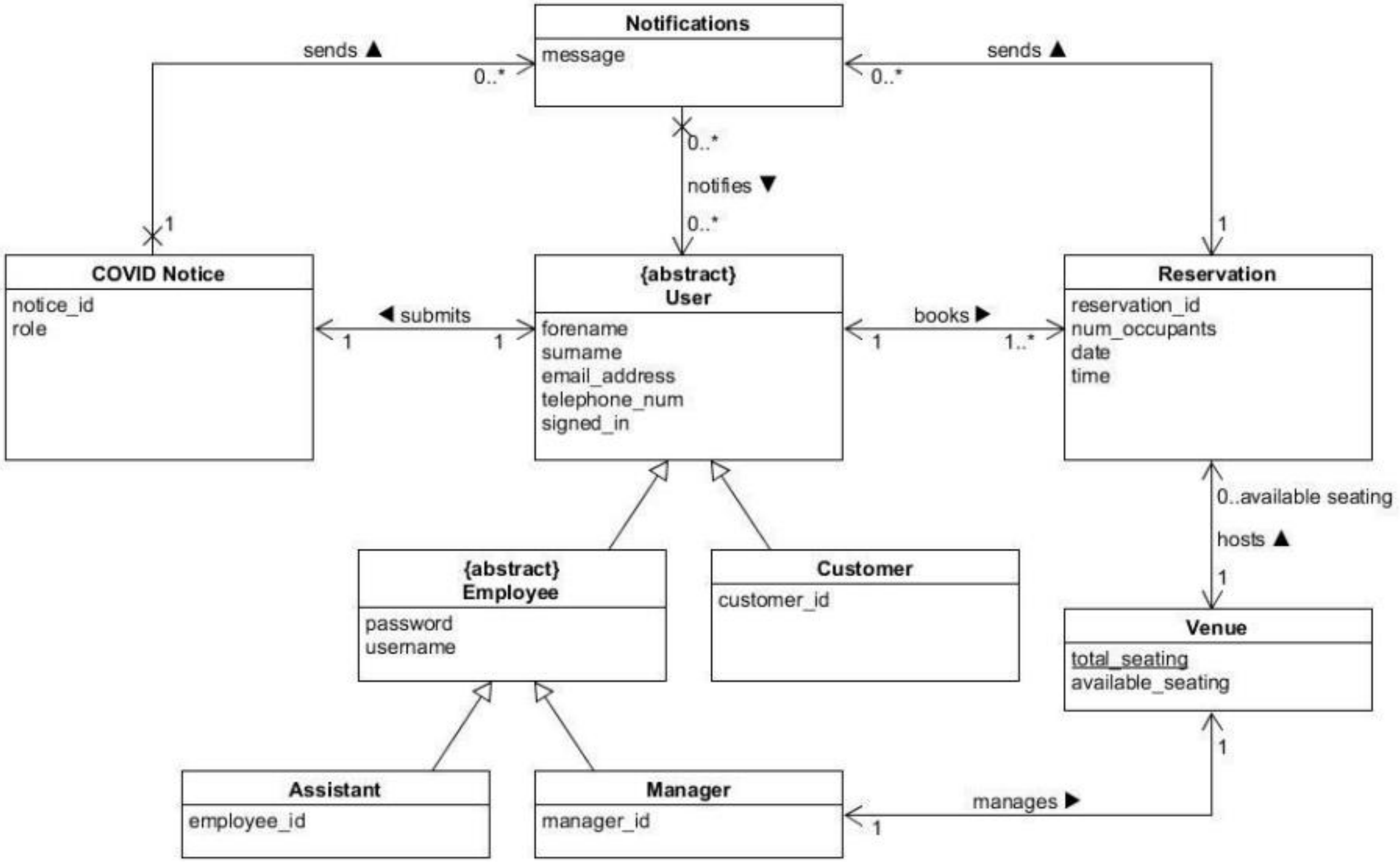
Alongisde the outdated class diagram there is also this outdated inheritance diagram that shows the relationship between the super class “user” and its two inheriting subclasses – Employee and Customer. From observing the diagram the reader can observe that the manager would have an additional set of activities that they can carry out via the webapp – namely, venue management.

However, within the developed solution the code was not structured this way and there was no inheritance between the model classes that were used. This diagram has been kept within this document to show that the intended architecture is different from the currently implemented architecture.

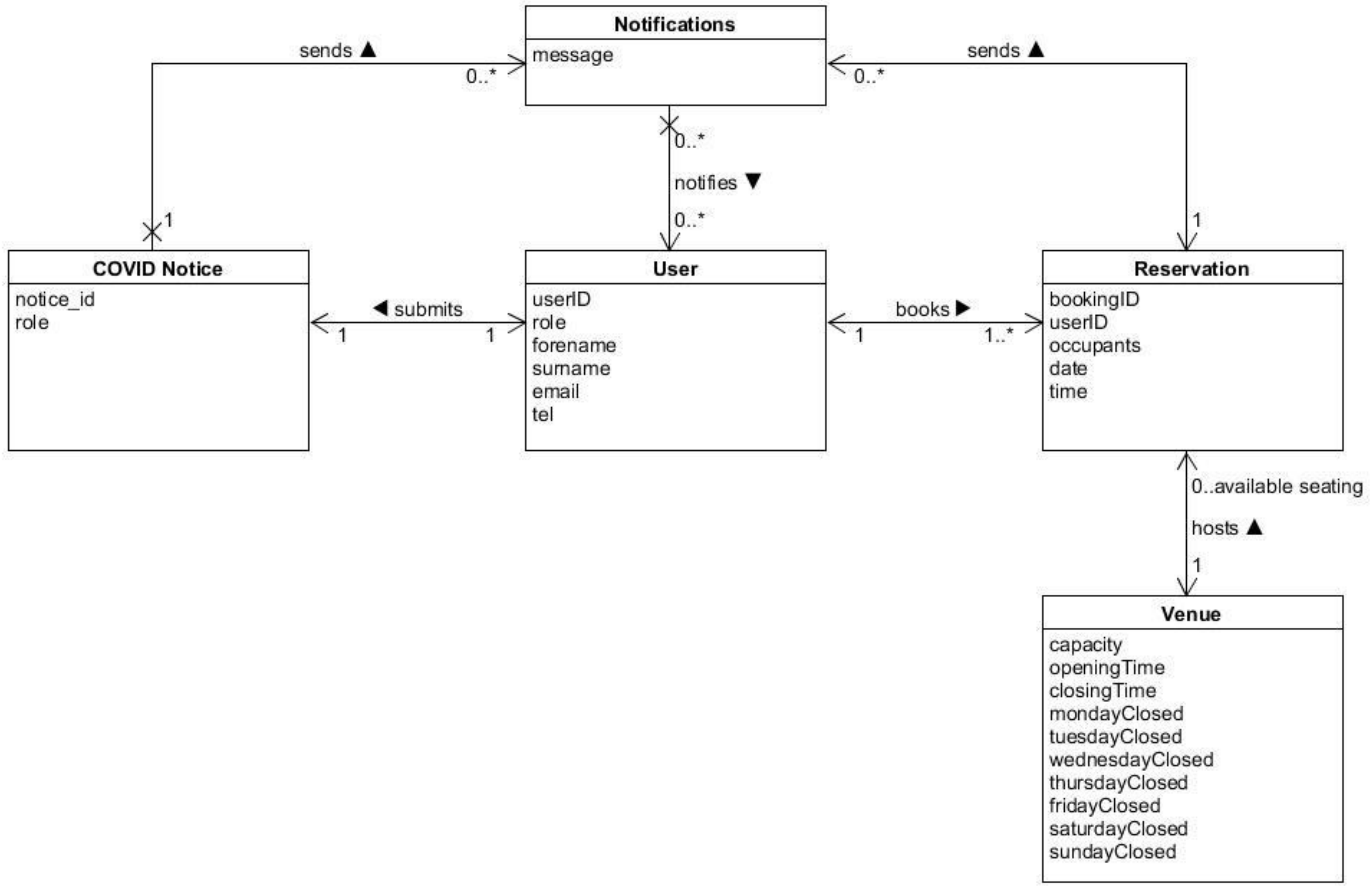
Likewise, because the other diagrams that were created as part of the original architecture are based upon these classes, they too are outdated and this document will provide updated versions.



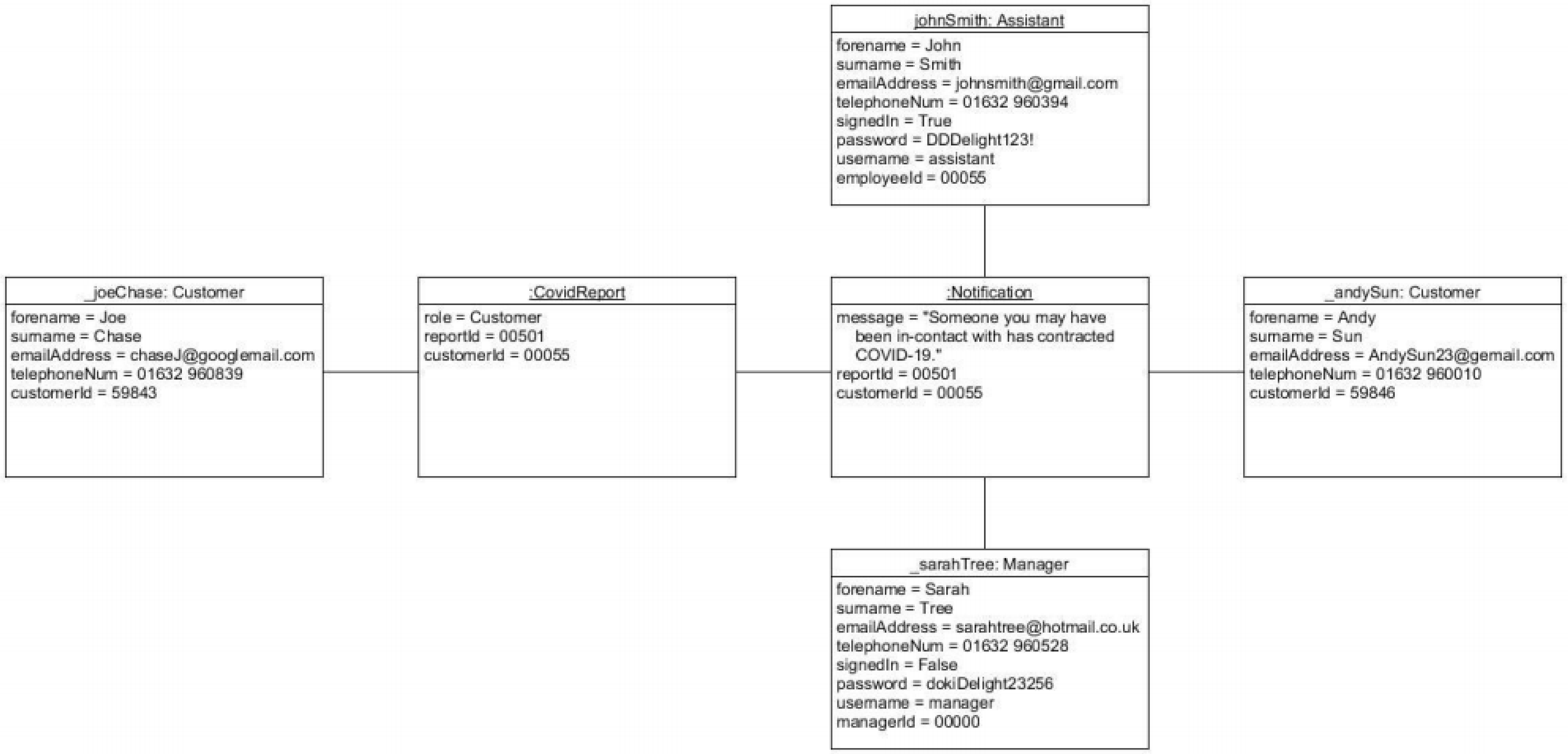
Outdated: This diagram conveys a detailed structure of the relationships between each class and its interactions with other classes:



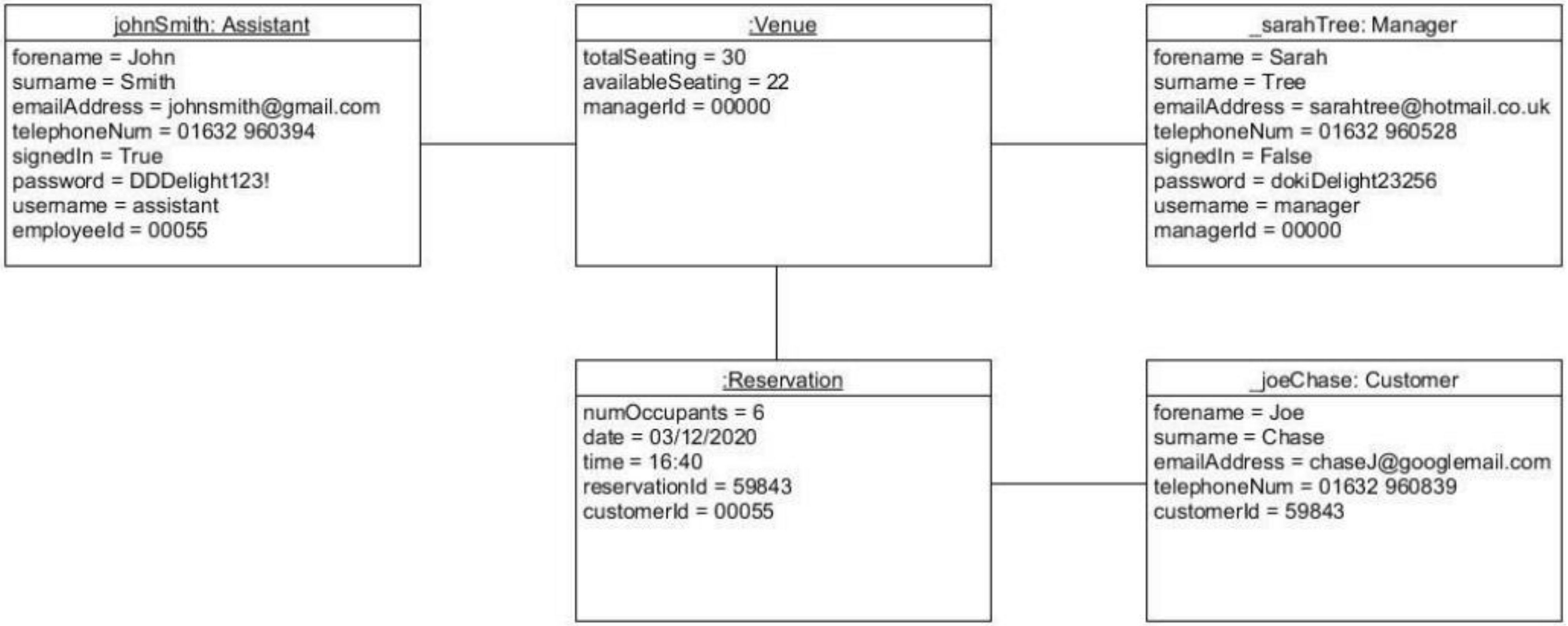
Updated: the classes' attributes have been updated and the subclasses have been removed.



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



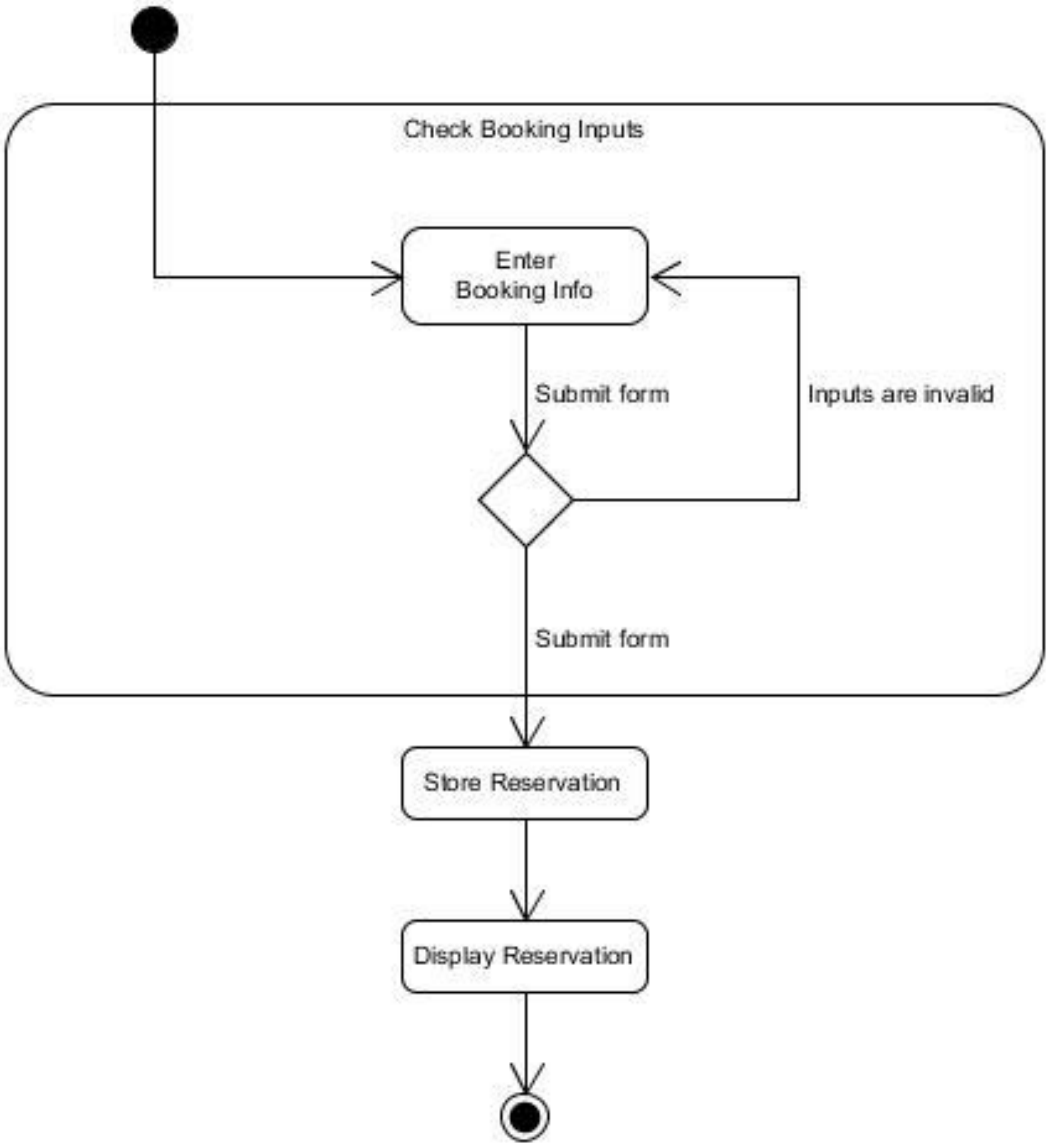
Scenario: A customer submits a COVID report and therefore notifications are sent to a collection of individuals that may have been in-contact with them. Details of implementing this tracing algorithm have been abstracted for simplicity.



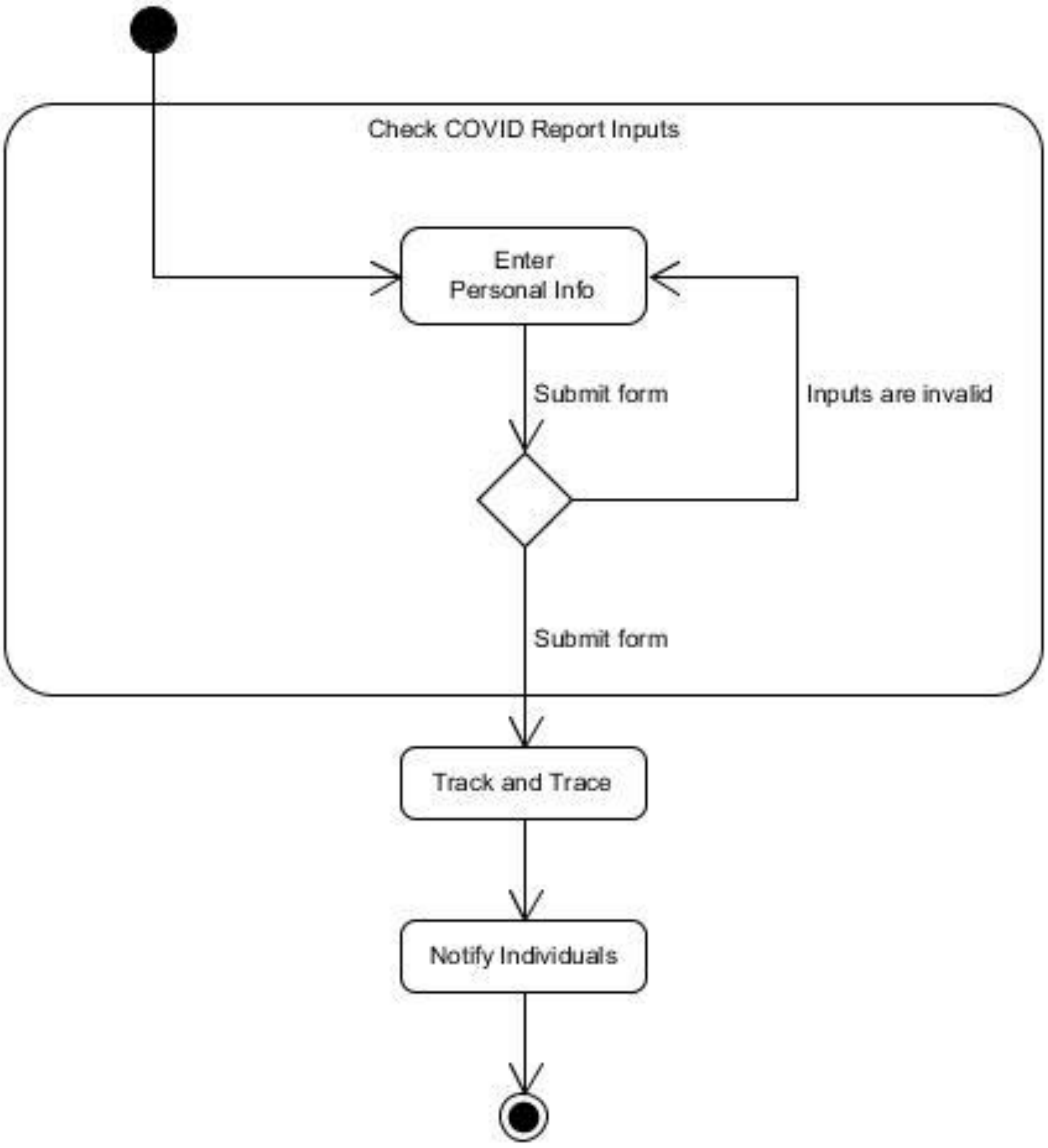
Scenario: A reservation can be booked by a Customer, and the Reservation object requires a Venue to be created – the venue will limit the number of reservations being created. A venue will be run by the business's Manager and Assistant. All users must input their personal information so that they can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

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

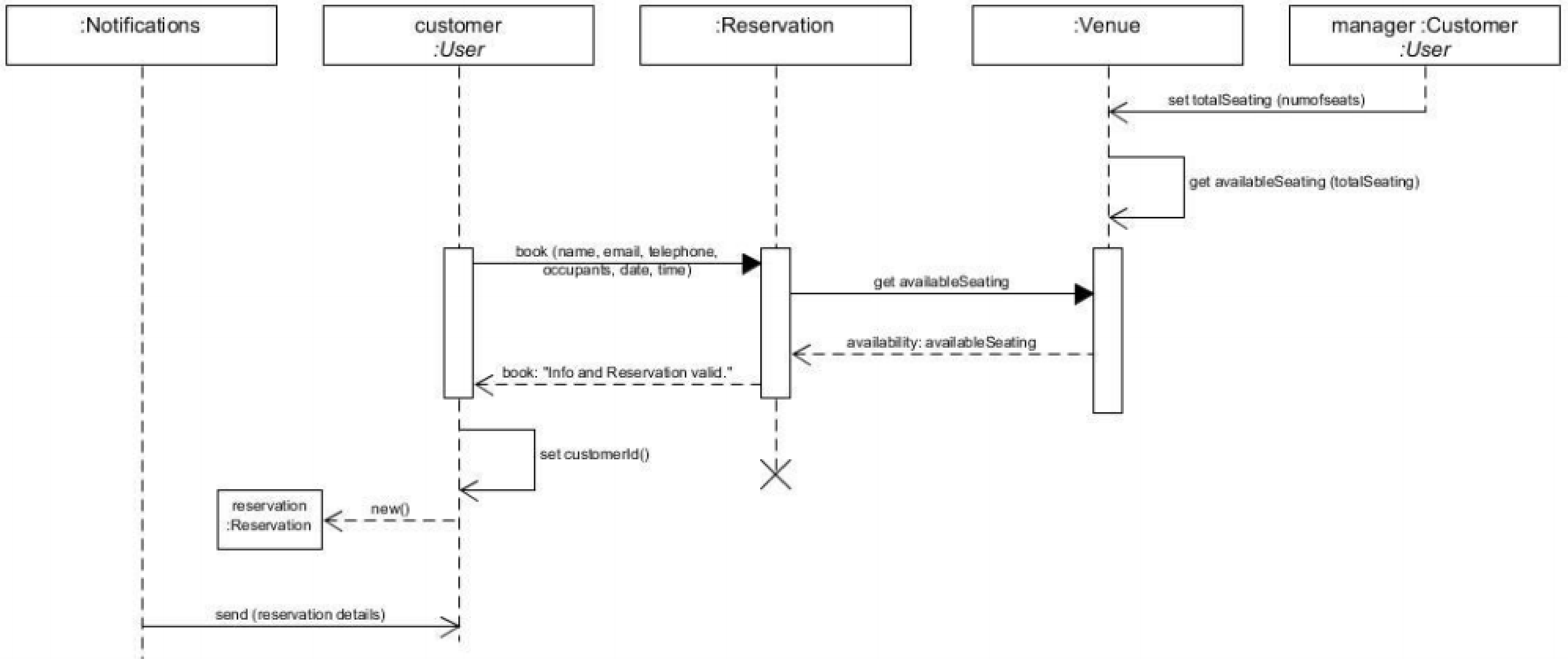
Reservation System:



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

Overall, the system has an ASP.NET backend API and reservations will be submitted to this API and multiple different components of the system will be able to access the backend to manipulate the data – for example, the COVID notification system will use the backend to find customer records. The application will also repeatedly request data from the API so that it can be written to the user.

Another example of how the API ties the entire application together is via the venue details and the creating a reservation process – the API will be used to fetch venue details and then evaluate whether the booking should be created or not based on capacity or the time/date the user has selected.

Sprint Planning

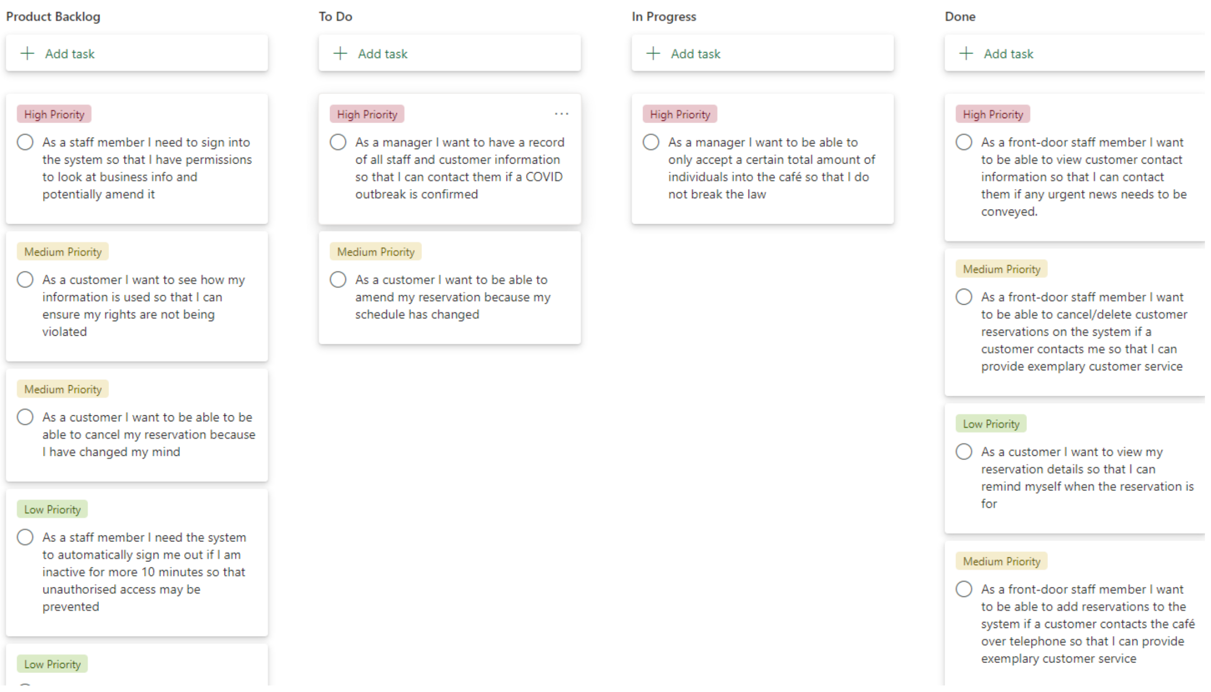
As previously seen within ***Requirements*** (page 6 of this report) a product backlog has been created, and given the properties: sprint, priority, and status. Based on these fields, the backlog has been translated over to Microsoft Planner so that the developer can easily manage what is needed to do each sprint. As the project uses the agile framework, they have sole responsibility of his schedule

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

Throughout this project, sprints have been planned to be worked on across a 2-week basis wherein a review of the deliverables produced will be conducted – and if it were required, the next sprint would have been designated to the same backlog feature.

The planner was 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, the planner is structured like this:

- Product backlog
- To do
- In progress
- Done



Product Backlog

+ Add task

High Priority
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

Medium Priority
As a customer I want to see how my information is used so that I can ensure my rights are not being violated

Medium Priority
As a customer I want to be able to be able to cancel my reservation because I have changed my mind

Low Priority
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

Low Priority

To Do

+ Add task

High Priority
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

Medium Priority
As a customer I want to be able to amend my reservation because my schedule has changed

In Progress

+ Add task

High Priority
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

Done

+ Add task

High Priority
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.

Medium Priority
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

Low Priority
As a customer I want to view my reservation details so that I can remind myself when the reservation is for

Medium Priority
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

The product backlog is formed of user stories and they are ‘tagged’ with priorities that relate to the previous MoSCoW processing. For additional readability, each tier within MoSCoW has been renamed to: high, medium, and low. The wont-have features were intentionally left out to keep the planner very simple.

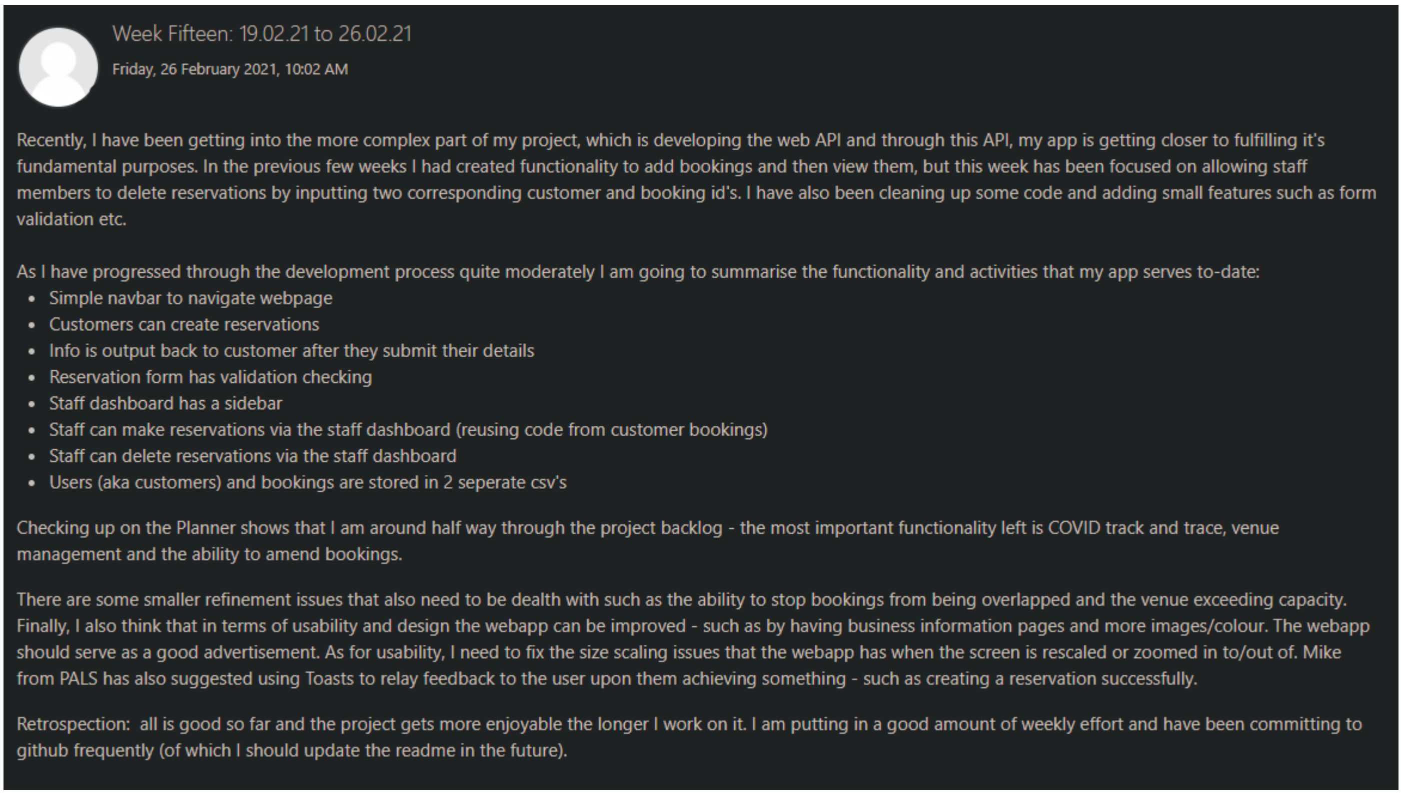
Throughout the project development phase, the process of carrying out a sprint followed this process:

At the start of a sprint, a manageable/achievable amount of user stories are moved to the "To-do" column. Then, the current tasks being worked on are moved 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, the developer will refer back to the BDD artifacts to test/review each user story and if it fulfils the criteria then they 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.

Alongside maintaining the Kanban board (also referred to as the planner), the developer maintained a weekly blog post wherein retrospective activities were carried out to determine how productive they have recently been – but to also highlight achievements. 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.

Overall, sprints are a basic way of managing the deliverables being produced and identifying if production is too slow – and the blog reviews work alongside this as they will help find the root cause of issues. From identifying the root cause, the problem can be solved. Throughout the entire project there were many issues that were encountered and without this process of analysing what can be done to overcome the issue, the project would have operated at a much slower pace.

An example of what a weekly blog review is as follows:



Week Fifteen: 19.02.21 to 26.02.21

Friday, 26 February 2021, 10:02 AM

Recently, I have been getting into the more complex part of my project, which is developing the web API and through this API, my app is getting closer to fulfilling it's fundamental purposes. In the previous few weeks I had created functionality to add bookings and then view them, but this week has been focused on allowing staff members to delete reservations by inputting two corresponding customer and booking id's. I have also been cleaning up some code and adding small features such as form validation etc.

As I have progressed through the development process quite moderately I am going to summarise the functionality and activities that my app serves to-date:

- Simple navbar to navigate webpage
- Customers can create reservations
- Info is output back to customer after they submit their details
- Reservation form has validation checking
- Staff dashboard has a sidebar
- Staff can make reservations via the staff dashboard (reusing code from customer bookings)
- Staff can delete reservations via the staff dashboard
- Users (aka customers) and bookings are stored in 2 seperate csv's

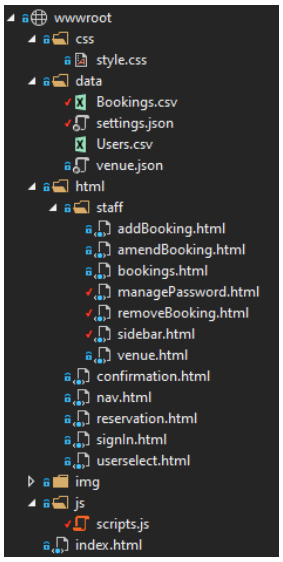
Checking up on the Planner shows that I am around half way through the project backlog - the most important functionality left is COVID track and trace, venue management and the ability to amend bookings.

There are some smaller refinement issues that also need to be dealth with such as the ability to stop bookings from being overlapped and the venue exceeding capacity. Finally, I also think that in terms of usability and design the webapp can be improved - such as by having business information pages and more images/colour. The webapp should serve as a good advertisement. As for usability, I need to fix the size scaling issues that the webapp has when the screen is rescaled or zoomed in to/out of. Mike from PALS has also suggested using Toasts to relay feedback to the user upon them achieving something - such as creating a reservation successfully.

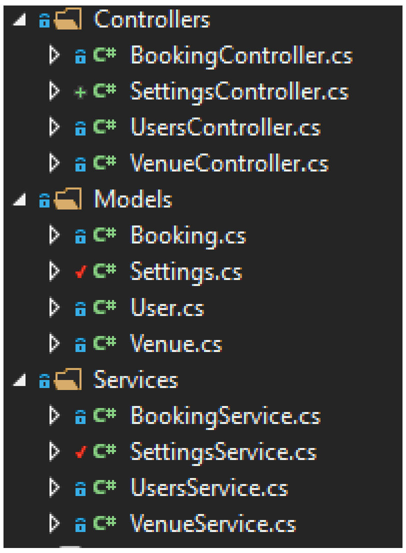
Retrospection: all is good so far and the project gets more enjoyable the longer I work on it. I am putting in a good amount of weekly effort and have been committing to github frequently (of which I should update the readme in the future).

Implementation

This project can be abstracted into two sections: the frontend and the backend. The frontend of the web application consists of html pages, CSS stylesheets and JavaScript scripts. Meanwhile, the backend has been implemented via ASP.NET, which is used to allow the application to create, store, read, delete, and amend data within text files.



```
wwwroot
  css
    style.css
  data
    Bookings.csv
    settings.json
    Users.csv
    venue.json
  html
    staff
      addBooking.html
      amendBooking.html
      bookings.html
      managePassword.html
      removeBooking.html
      sidebar.html
      venue.html
    confirmation.html
    nav.html
    reservation.html
    signIn.html
    userselect.html
  img
  js
    scripts.js
  index.html
```

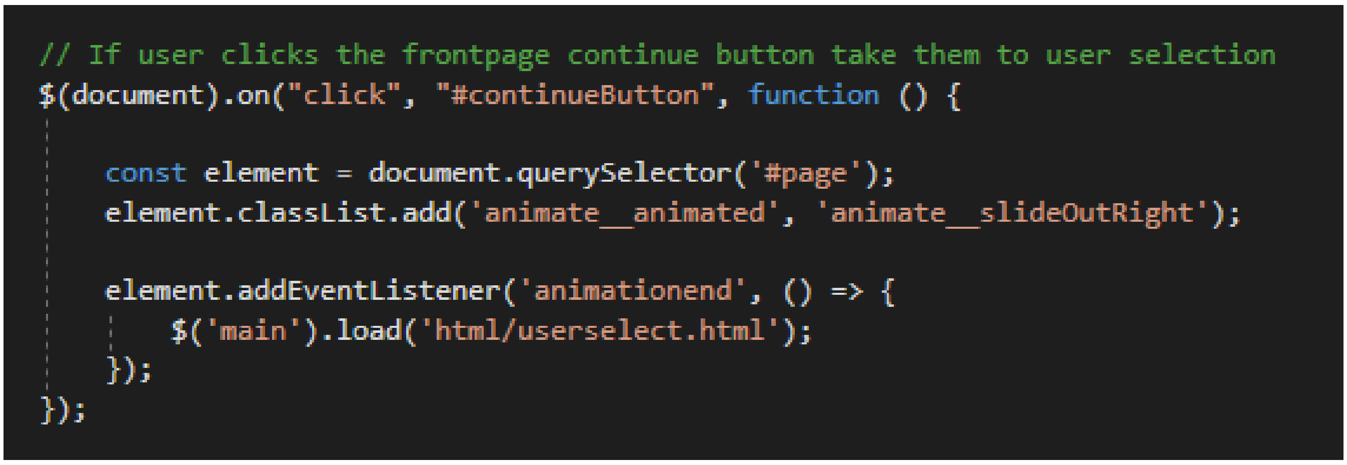


```
Controllers
  BookingController.cs
  SettingsController.cs
  UsersController.cs
  VenueController.cs
Models
  Booking.cs
  Settings.cs
  User.cs
  Venue.cs
Services
  BookingService.cs
  SettingsService.cs
  UsersService.cs
  VenueService.cs
```

The API is implemented in three factors – Controllers, Models and Services. Models serve as a template for the data being handled, whilst the Controllers act as a gateway between the application and the Services that the API hosts. These services range from read/write methods to methods that generate ID's for users and reservations.

Structuring the web-app in this manner allowed for an easy way to store data permanently and was relatively simple to learn. Storing data within .txt and .json files was seen as a better alternative to databases because it was simpler, and the application is not expected to be large scale. Another benefit of structuring the backend like this was the ease of finding errors or problems with the written code, because of the encapsulation and separation of different functionalities. Overall, the code was made manageable and easy to expand.

To adhere to the Single Page Application requirement, the application uses JavaScript and jQuery to dynamically load new content onto the index.html page. The html of each different aspect of the web app has been separated into their own html pages so that they are each easy to style and so that issues are easy to find. The code for loading the contents of a different html page into index.html can be seen below:

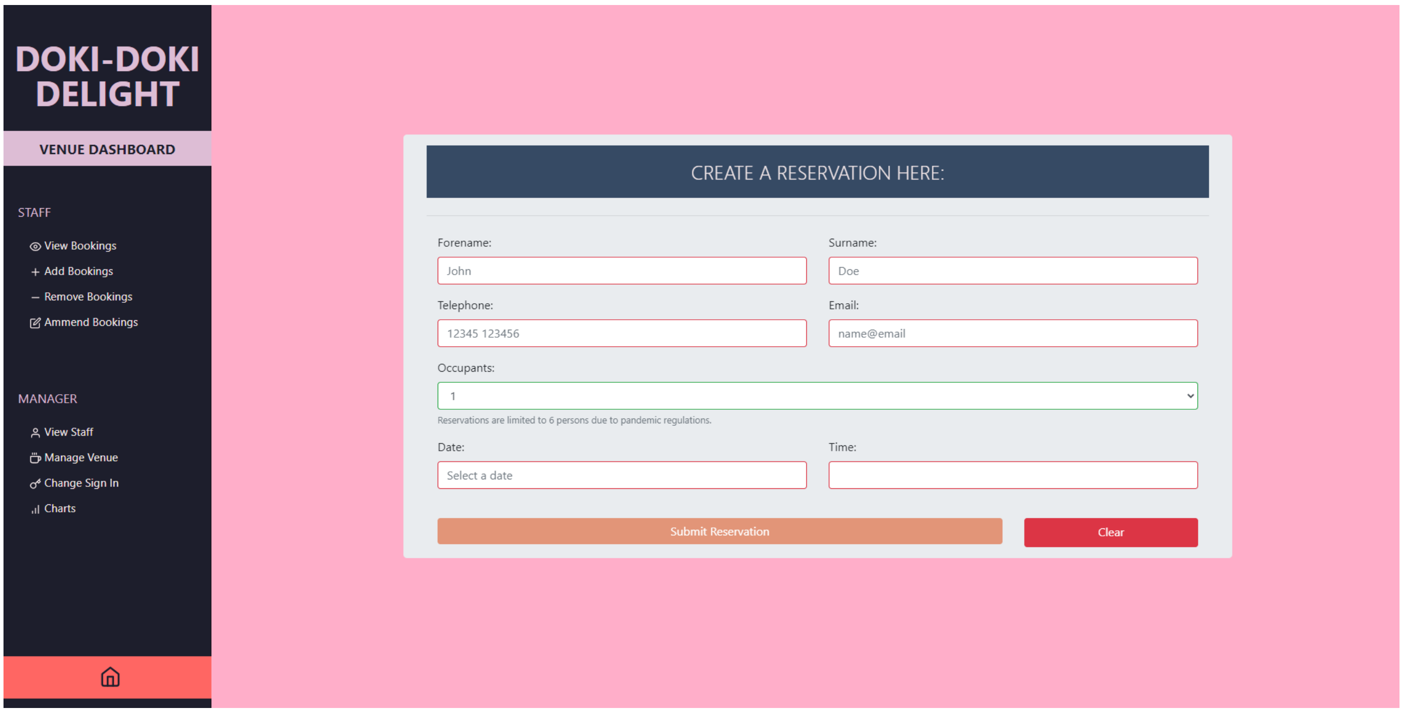


```javascript
// If user clicks the frontpage continue button take them to user selection
$(document).on("click", "#continueButton", function () {

    const element = document.querySelector('#page');
    element.classList.add('animate__animated', 'animate__slideOutRight');

    element.addEventListener('animationend', () => {
        $('main').load('html/userselect.html');
    });
});
```

Core Features of the Program:

As a café reservation and management system, the core functionalities of this application were to be able to create reservations and be able to manage them. Another core focus of the application was the COVID-19 Track and Trace functionality – however, this has yet to be implemented due to a greater focus being spent on the development of the hospitality side of the site.

Reservations are made by submitting data via forms, which are validated and POSTed to the API which then records them in a txt file. These reservations can be further amended or deleted by the PUT and DELETE html request methods. In these cases, the application requests the user's user ID and booking ID.



CREATE A RESERVATION HERE:

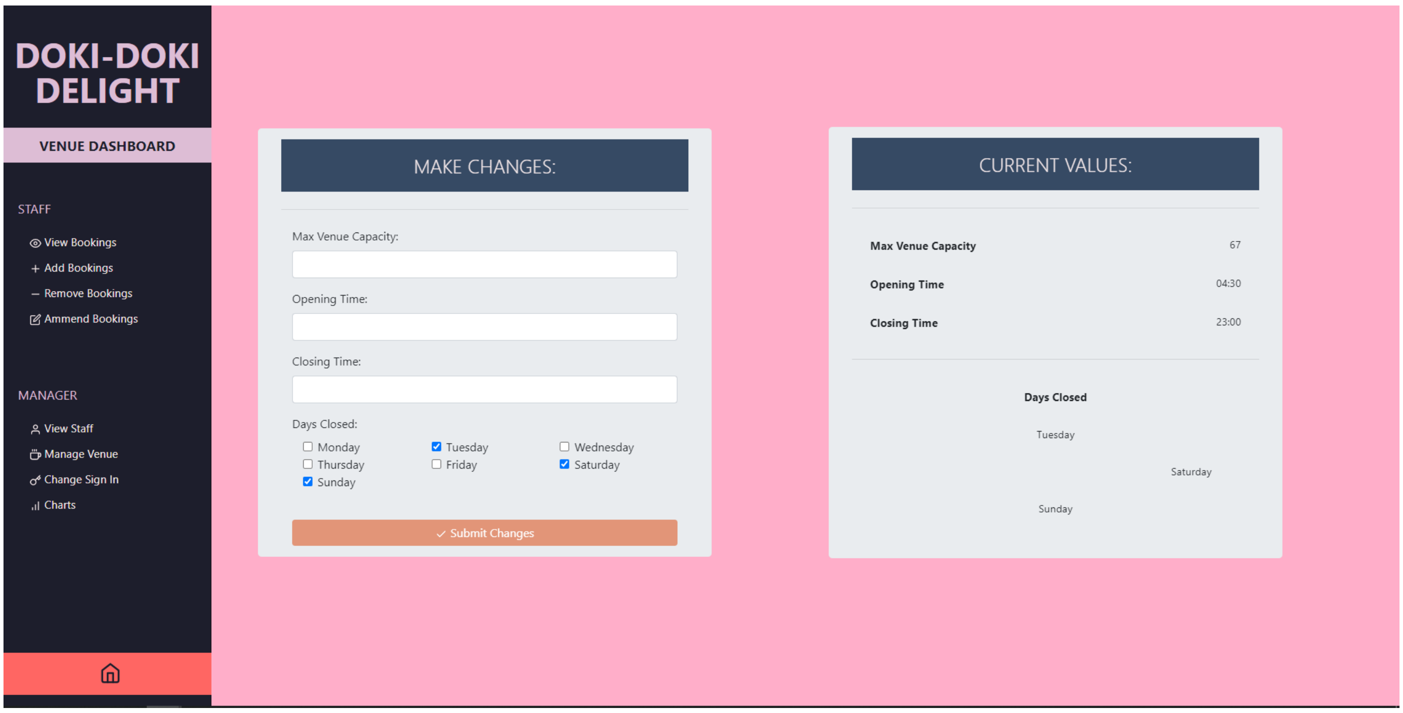
Forename: John	Surname: Doe
Telephone: 12345 123456	Email: name@email
Occupants: 1	
Date: Select a date	Time:

Reservations are limited to 6 persons due to pandemic regulations.

The reservation features were implemented to provide easy access for customers to the café – by allowing reservations to be created through the website, Doki-Doki Delight can potentially draw in new customers and offer them an easy way to eat out. Taking usability into consideration, the app uses colour to distinguish valid input fields from invalid ones (as seen in the red and green highlighting of the fields in the image above).

Colour has also been used within buttons to distinguish buttons with an undesirable affect from buttons that achieve something – such as submitting the form.

Other than reservations, the staff dashboard of the webapp also has a section where the venue details can be managed, and days open/closed can be toggled off as well as opening and closing times changed. This part of the website determines the possible selections shown to the user within the date and time pickers and is therefore important in running a business as customers should not be able to book reservations on closed days. Therefore, this feature adds redundancy.



MAKE CHANGES:

Max Venue Capacity:

Opening Time:

Closing Time:

Days Closed:

- [ ] Monday
- [x] Tuesday
- [ ] Wednesday
- [ ] Thursday
- [ ] Friday
- [x] Saturday
- [x] Sunday

Submit Changes

CURRENT VALUES:

Max Venue Capacity	67
Opening Time	04:30
Closing Time	23:00

Days Closed

Tuesday

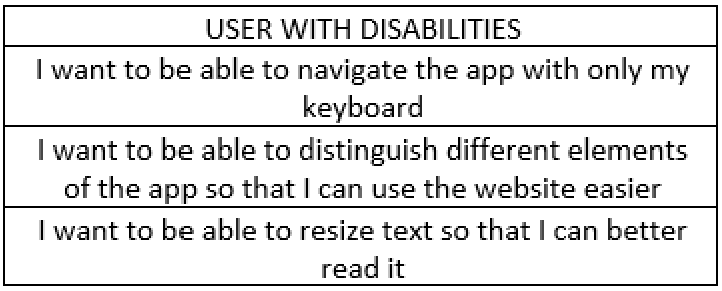
Saturday

Sunday

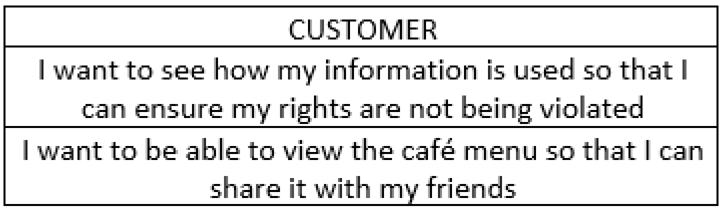
This table shows all user story requirements that have been completed or have been started:

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		Fulfilled
As a staff member I want to use keyboard and mouse so that I can be more efficient when typing/inputting data	Must		Fulfilled
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		Fulfilled
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		Half-completed
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		Fulfilled
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		Half-completed
As a customer I want to book a reservation so that I can eat within Doki-Doki Delight's café	Must		Fulfilled
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		Fulfilled
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		Fulfilled
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		------------------

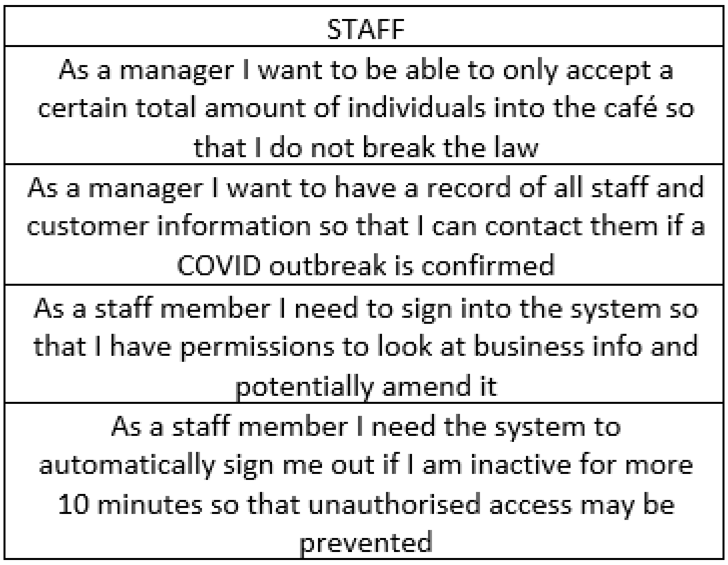
Features of the program that did not get implemented:
Unfortunately, not all of the User Stories got implemented within the final deliverable due to the final deadline closing in quicker than the developer planned. The user stories that did not get implemented are as follows:



USER WITH DISABILITIES
I want to be able to navigate the app with only my keyboard
I want to be able to distinguish different elements of the app so that I can use the website easier
I want to be able to resize text so that I can better read it



CUSTOMER
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 view the café menu so that I can share it with my friends



STAFF
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 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

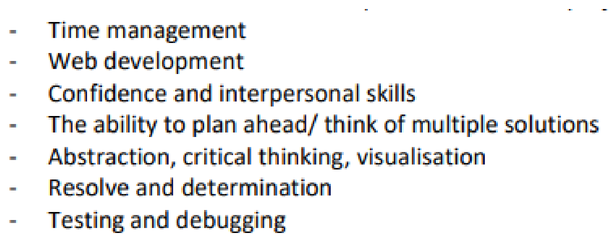
By observing the above user stories, it can be seen that a lot of quality-of-life features were not implemented – as well as staff authorisation as it was not a crucial part of the app – it did not offer any advancement towards Doki-Doki Delight's vision. Another feature that was not implemented was the maximum total capacity of the café, which would be used to stop reservations being made on specific days if the café were fully booked out.

The main reason for not developing these features was time and bad organisation efforts of the project. In the future, the developer of this project has learnt to use strict planning tools such as Gantt charts to keep themselves on track and to break tasks down into time slots.

Reflection

Since the submission of the Interim report and the presentation at the Marketplace Demo, the main software deliverable has been developed, and as highlighted within the Interim report, the project was of a manageable scale. Advancements were made in areas of which the developer behind this project had little or no experience within – such as tying in a front-end application with a backend API or using JavaScript to make a website dynamic and functional.

Within the planning stages, the developer outlined within the Project Proposal document that the following skills were necessary to develop to be able to carry out the project:



- Time management
- Web development
- Confidence and interpersonal skills
- The ability to plan ahead/ think of multiple solutions
- Abstraction, critical thinking, visualisation
- Resolve and determination
- Testing and debugging

Within this list there are both soft and hard skills; and by the end of the project the developer had certainly started to train most of these skills. Management skills were displayed via the planning of sprints; development skills were enhanced by experimenting with code and researching existing solutions and adapting them; interpersonal skills were shown within the Marketplace Demo and through interaction with peers.

Through thinking of the system as a whole, as well as individual processes/components, a display of critical thinking was shown and the ability to visualise what the application should do and then implementing this vision is also something that the developer tried to train their brain into doing. As a result, code was well structured.

Of the taught modules in this first year of study, the most relevant to this project was COMP1000 wherein C# skills were developed – and this applied to the project via the ASP.NET backend which required file read/write functionality that had been developed through the COMP1000 coursework. The ASP.NET backend also used classes as templates to create objects from, which is a concept that this module taught. Without the prior C# knowledge, the API would most likely have suffered from unreadable code.

Through this project the significance of planning the architecture and analysing the requirements has been made abundantly obvious – without these factors, the final deliverable would not have been usable or include the intended functionality. The UML diagrams provided a clear starting point, and although the architecture diverted from these plans, the diagrams help the developer think critically and decompose the project into manageable sizes.

Finally, the retrospective blog posts helped the developer maintain confidence and an orderly schedule as they served as an outlet for stress, but also a way to list accomplishments and express which requirements have been completed, are being worked on, and are due to be started.

One factor that should definitely be considered by the reader, the developer of this project, and other developers for their future projects is that stakeholders and clients should be contacted on a frequent basis. This project had some exposure during the Marketplace Demo and to the developers' close acquaintances, but as new features were constantly being added or changed, more and more feedback should have also been sought.

Therefore, this report will conclude by expressing that if anything could be changed, there are two activities that would be carried out: (1) creation of a focus group wherein users answer a questionnaire after spending an allocated time with the software, and (2) creation of a risk assessment document wherein all recognised issues and risks would be recorded for future reference within other projects, but to also outline the main issues with this current project.

Overall, the result of this project was satisfactory, and all components have built towards this in some way.

Bibliography

Agilemanifesto.org. 2021. Principles behind the Agile Manifesto. [online] Available at: <https://agilemanifesto.org/principles.html> [Accessed 16 May 2021].

Caldwell, B., Cooper, M., Reid, L. G., Vanderheiden, G. (2008). Web content accessibility guidelines 2.0, World Wide Web Consortium (W3C) recommendation.

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.

McConnell, S., 2014. Rapid development. Redmond (Washington): Microsoft Press.

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].

Schmutz, S., Sonderegger, A. and Sauer, J. (2016) 'Implementing Recommendations From Web Accessibility Guidelines: Would They Also Provide Benefits to Nondisabled Users', Human Factors, 58(4), pp. 611–629.

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

Scrumguides.org. 2021. Scrum Guide. [online] Available at: <https://scrumguides.org/scrum-guide.html> [Accessed 16 May 2021].

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

World health organisation. (2011). World report on disability. Geneva, Switzerland, World Health Organization.

W. Royce, W., 2021. Managing the development of large software systems: concepts and techniques. In: Proceedings of the 9th International Conference on Software Engineering. Washington, DC, USA: IEEE Computer Society Press, pp.328–338.