COMP1004

Computing Practice

2020/2021

Restaurant Management System

Introduction

In the current climate of the coronavirus pandemic, restaurant venues have had to adapt the way in which they operate, such as needing to take contact details for contact tracing and limiting the number of people that can sit at a table. I am developing a restaurant management system to support businesses with these changes. This report will go through the stages of the software development lifecycle and how these are being applied to my project, while also talking about various software development methodologies. It will then talk through the wider context around my project and any social, legal and ethical concerns that may arise. After this I present the requirements and architecture of the system and will finally go into the discussion of my sprint plans for implementation and evaluating the work I have done so far.

Software Development Lifecycle

The software development lifecycle brings structure to software development to ensure quality software. There are 5 stages of the SDLC:

- Requirements Analysis
- Design
- Implementation
- Testing
- Evolution

Without following this, projects can fail or take longer than expected, which leads to companies losing money. There are various models of the SDLC, such as Waterfall or Agile. There is a standard set for SDLC by the International Organisation for Standardization (ISO) titled 'ISO/IEC/IEEE 12207:2017 Systems and software engineering – Software life cycle processes'

Requirements Analysis
The first stage of the SDLC about gathering requirements of your software. These can be broken into 3 parts. Functional, Non-Functional and usability. Functional requirements are the tasks the system performs and how it responds to inputs. Non-functional requirements describe the constraints of software and how well it meets functional requirements. Another part of this is external requirements, these are social, legal and ethical considerations. Finally, usability requirements describe how easily the software can be used.

Using functional requirements, you can create use case diagrams that will form part of a projects UML. These are a way to visually represent the actions the software will perform based on input. Use case description can then be used to expand upon use cases.

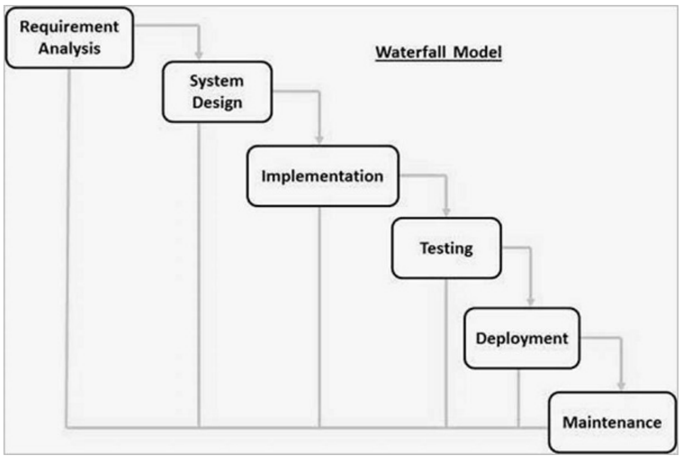
Design
After this, you need to design software based on requirements. There are two parts of this, detailed and style. Style considers architecture and basic design of how software will work, and you can see this in my own project in a later section. Another way of viewing software is using the object-oriented paradigm. This is the way of developing software uses classes, which are like templates of objects that have set attributes and methods, making code reusable.

You also create UML such as class diagrams, state diagrams and sequence diagrams. Class diagrams allow you to present how classes/objects will look in software by giving it its attributes and methods and how they inherit from other classes if necessary.

Following on from this is implementation, testing and evolution. In these stages you take the designs made and implement them in code. After this comes testing, which is done to ensure software doesn't contain errors and fixing errors that are found. After this the software is complete. Evolution is the stage where if bugs are found or new features are required the developers will return to working on the software.

Agile
Agile is a needs focus approach for seeing quick results. Agile projects are completed in sprints, which are normally two-week periods where you create a small chunk of software, known as the minimum viable product. Agile teams meet throughout sprints to discuss progress. Unique aspects to agile are user stories and a product backlog. Requirements can be made understandable to anyone using user stories and come in the form "As a <who>, I want to <what> so that <benefit/why>" (What is User Story?,2020) This makes it easy for anyone to understand tasks software should perform. The second thing is a product backlog, this holds all the features that your software will have and each sprint some of these get worked on.

Waterfall
The waterfall model has been around for decades and sets a structure to how software should be developed. Each phase must be completed before the next, ensuring no overlap. This gives expectations of what should be completed at the end of each phase. Having these deadlines means the project is clearly documented and there is no uncertainty. This means that requirements are clear however, changes can cause a loss of time and money.



Waterfall model diagram – Tutorials point https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm

Applying the SDLC to my project

I am taking an agile approach to development when it comes to developing, meaning I will be using sprints. To begin with I will develop functional requirements, ranking them using MoSCoW prioritisation which will inform my user stories and use cases. After this I will create a product backlog ready for implementation. I will then create class, state and sequence diagrams as well as use case descriptions to develop the architecture.

Project Description

In the current climate of a global pandemic, businesses are now required by law to take contact details to be shared to NHS Test and Trace if needed. The Gov.UK website states that "

- *Businesses and other public settings where people meet socially including hospitality, close contact and leisure venues must record contact details of customers, visitors and staff on their premises to tackle the spread of coronavirus*
- *Details must be stored for 21 days and shared with NHS Test and Trace, if requested*
- *Fixed penalties for organisations that do not comply"*

(Venues required by law to record contact details,2020)

Business have been advised to take contact details since they reopened on 4th July however since 18th September, it is now required by law to do so. Contact details the that are expected to be recorded are:

- *Name*
- *Contact number*
- *Date and time of visit*
- *Departure time (if possible)*
- *All collected data must comply with GDPR and will not be kept for longer than necessary.*

The restaurant management system will meet the requirements set by government in terms of contact details required and limiting the number of people that can meet inside a hospitality venue which is currently 6 when they can be open dependent on government guidelines. It will also provide the means for marking a customer as Covid-19 positive so that these details can be passed to Test and Trace if needed. It will also allow staff to set the restaurant opening time and the number of tables that the restaurant has.

As well as the legal considerations around Covid-19 restrictions as stated above, the data being collected for contact details also needs to comply with GDPR. The data must be kept for 21 days for Test and Trace due to the incubation period of the virus and time needed for testing and tracing. After this the data should be securely deleted. According to the ICO's guide to GDPR principle (b): Purpose Limitation – "You must be clear about what your purposes for processing are from the start." (Principle (b): Purpose limitation, 2018) This means that I will need to explain to customers why data is being collected and how it is used. Another part of GDPR that the Covid-19 regulations mention is Principle (e): Storage Limitation, which is about only keeping personal data for as long as needed. (Principle (e): Storage limitation, 2018) While the COVID regulations state 21 days, this is only for if the details were being collected for contact tracing but because there is another use to them which is bookings, they can be kept for longer, but you would need to justify why ensuring that when not needed it is either erased or anonymised.

Requirements

To begin with, I started by gathering functional requirements of my product and ranking each using MoSCoW, a prioritisation technique used in agile, ranking how important each feature is to decide what to focus on. The acronym stands for Must have, should have, could have, and Won't have. (Chapter 10: MoSCoW Prioritisation, 2021) Doing this means some requirements that I produced may not be in the final product depending on complexity and whether it is needed. By defining and refining requirements, you can create validation list what won't be accepted.

Functional Requirements

1. System will allow a customer to book a table, taking name and phone number for track and trace purposes
2. System will allow restaurant staff to set the opening times
3. System will allow restaurant staff to set the number of tables
4. System will prevent a booking size larger than currently allowed based on government guidelines
5. System must respond to an invalid phone number by telling user that phone number is invalid
6. System should allow staff to make a booking on behalf of customer that phones up
7. System will allow bookings to be amended
8. System will allow the restaurant to be able to set tables of various sizes

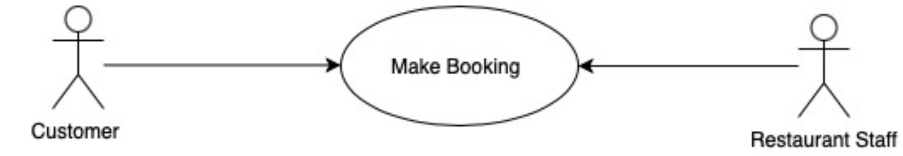
9. System will allow restaurant staff/ health workers to mark a customer as testing positive for coronavirus
10. Health worker will then be able to get a list of customers that were in the restaurant at around same time
11. System will have login system so only staff members have access to view bookings and can set restaurant details

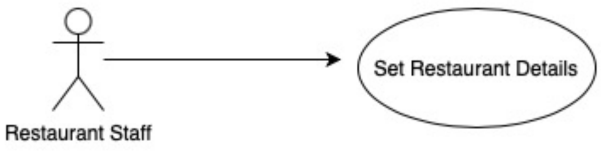
Must– 1-5,9
Should– 6,7, 10,11
Could– 8

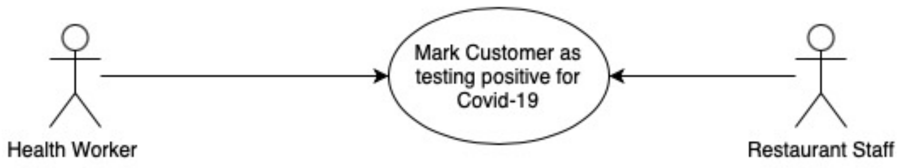
User Stories

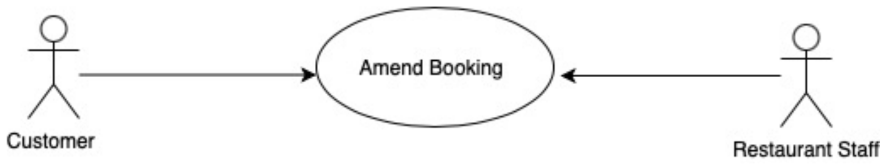
Customer	Restaurant Staff	Health Worker
As a customer, I want to be able to book a table at a restaurant so that I can go out for a meal	As a restaurant owner, I want to take name and contact details when a booking is made so that I can conform to government guidelines for contract tracing	As a health worker, I want to be able to get the details of customers that have tested positive so that I can trace other customers that could have been in contact with the customer that tested positive
As a customer I want to be able to amend bookings so that I can still go to the restaurant if circumstances change	As a restaurant owner, I want to be able to set restaurant information so that we have up to date details for customers	As a health worker I want to be able to mark a customer as testing positive for Covid-19 (Restaurant workers should also allowed to do these)
	As a restaurant owner, I want to be able to make a booking on behalf of a customer, so I can bring in customers who may not be IT literate or have internet access	

After this I can then develop my use case diagrams that form part of the overall UML for my project. Not all my user stories translate into use case diagrams as they are requirements that are fulfilled in other parts of the system, e.g., details that restaurant owner wants to be able to take is a part of making a booking but from the perspective of a business and not the customer.



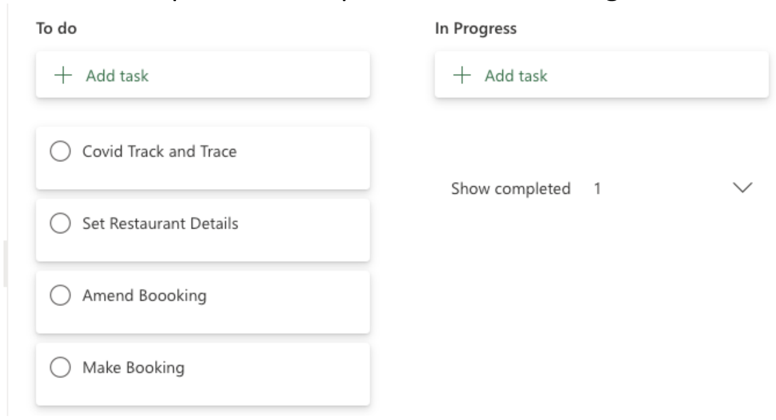






Product Backlog

The final part of the requirements phase is to develop my project backlog, this will hold all the tasks that will need to be completed throughout my project, for the minute these tasks only represent the use cases/ epics of my project and as I start making designs, tasks will be added to the planner to represent what is being worked on.


To do

+ Add task

Covid Track and Trace

Set Restaurant Details

Amend Boooking

Make Booking

In Progress

+ Add task

Show completed 1

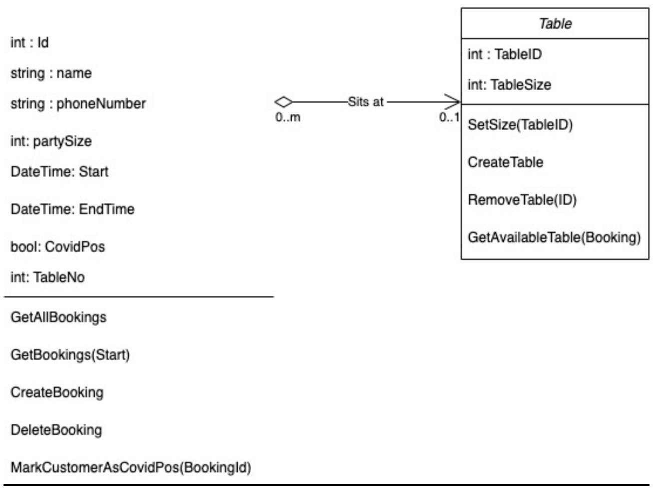
I will also be labelling tasks with priorities as you will see in my sprint planning.

Architecture

Class Diagrams

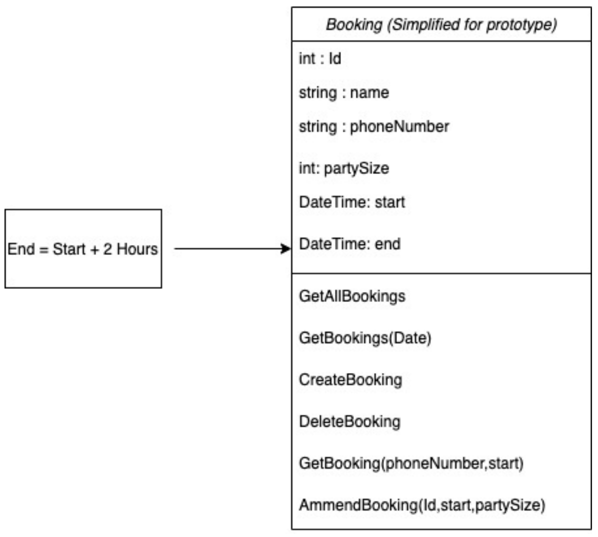
The first thing is to develop class diagrams to show how classes are laid out. A standard implementation for a system like this would use a database but for my prototype I will be storing bookings as objects that are read from a text file. Using a database is something that could be considered as it gives a persistent datastore without needing to load data each time as in this implementation, each booking is added to a list in the program and that is how they can be accessed and manipulated.

Class Diagram – Complex



I started off with a more complex diagram that used 2 classes, bookings and tables, as tables in the restaurant could be of varied sizes and so you need to make sure there is an available table at time of booking. However, this creates complexity that isn't relevant to developing the MVP (Minimum Viable Product) which is the aim of this report.

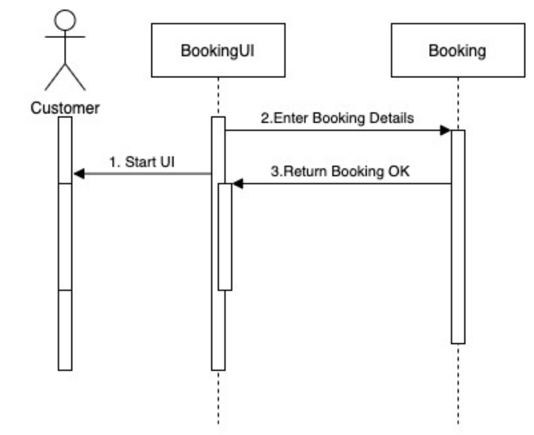
Class Diagram – Simplified



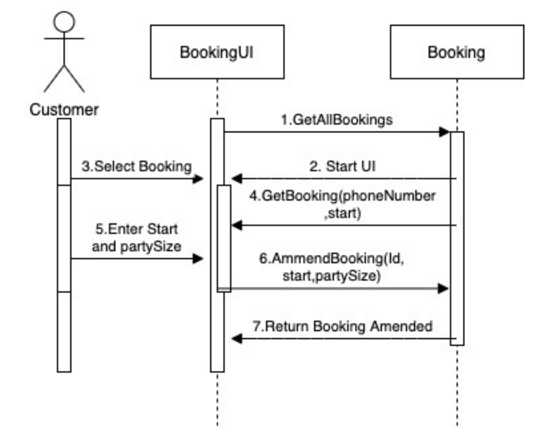
This is the simplified version of the class diagram that will be used in my prototype and provides all relevant attributes and methods needed to carry out basic functions of the system. The only omission from this is marking customers as COVID positive, however this is something that could be refactored.

Sequence Diagrams

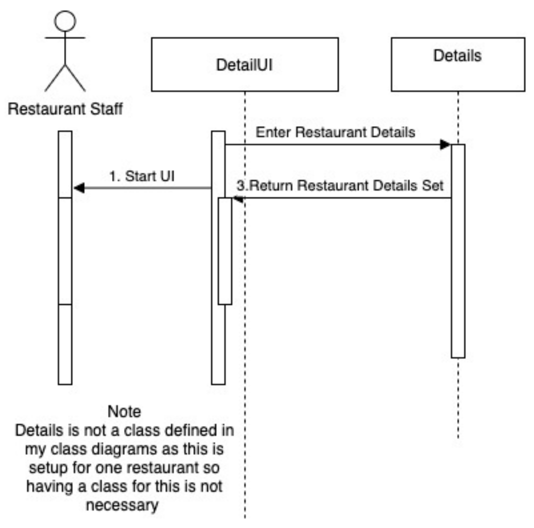
Make Booking



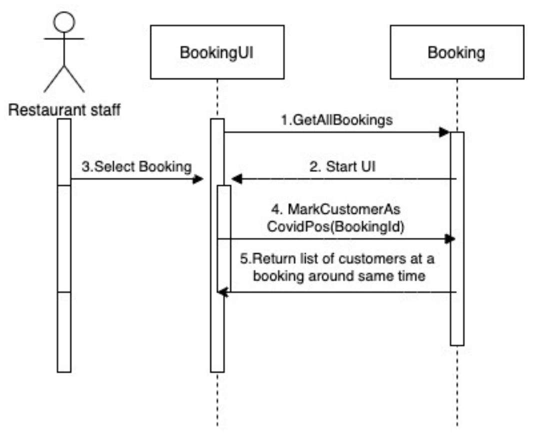
Amend Booking


Customer
BookingUI
Booking
1.GetAllBookings
3.Select Booking
2. Start UI
4.GetBooking(phoneNumber ,start)
5.Enter Start and partySize
6.AmmendBooking(Id, start,partySize)
7.Return Booking Amended

Set Restaurant Details

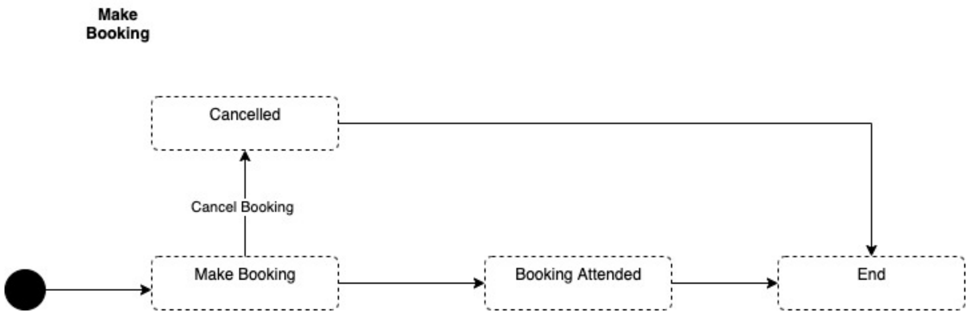

Restaurant Staff
DetailUI
Details
Enter Restaurant Details
1. Start UI
3.Return Restaurant Details Set
Note
Details is not a class defined in my class diagrams as this is setup for one restaurant so having a class for this is not necessary

Mark Customer Testing Positive for Covid


Restaurant staff
BookingUI
Booking
1.GetAllBookings
3.Select Booking
2. Start UI
4. MarkCustomerAs CovidPos(BookingId)
5.Return list of customers at a booking around same time

Sequence diagrams show how an actor interacts with the user interface of a system and how that interacts with the object being used. As you can see, these reference the class and methods that are used in the class diagram. So when you get to coding, you have coherent references in your UML.

State Diagrams



State diagrams show the various states that an object can have whilst the program is running, the only use case that can have various states is making a booking, as a booking can either be attended or cancelled before being ended.

Use Case Descriptions

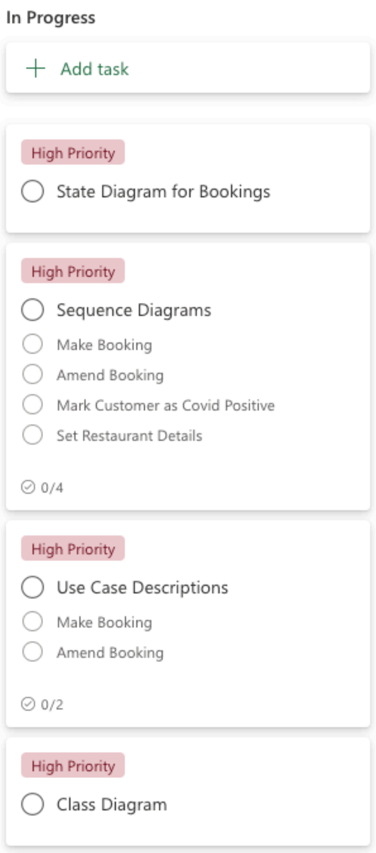
Use case descriptions are a way to expand upon use cases, explain how they work, the processes they follow, and conditions needed for the use case to run. I am using these to further explain make and amend booking as it's easy to expand upon as the condition that needs to be met is that a table needs to be available to make a booking. The template for this is from UML @ Classroom book (Seidl, Brandsteidl, Huemer and Kappel, 2012)

Name	Make Booking
Short Description	Books table at restaurant
Precondition	Customer has valid phone number
Post Condition	Table has been booked
Error Situations	No free table
System state in the event of an error	Table cannot be booked at this time
Actors	Customer
Triggers	Customer requires a table
Standard Process	1. Customer enters name and phone number 2. Customer choses party size 3. Customer enters date/time 4. System confirms there is a table free 5. Customer confirms the booking
Alternative Process	3’ Table is not available 4’ System gives next time slot for a free table 5’ Customer selects new time and confirms booking

Name	Amend Booking
Short Description	Amend a booking
Precondition	Customer has already made a booking
Post Condition	Booking amended
Error Situations	There is not a free table at amended time
System state in the event of an error	Table cannot be booked at this time
Actors	Customer
Triggers	Customer needs to amend booking
Standard Process	1. Customer enters booking details 2. System finds booking 3. Customer selects new date/time for booking 4. Customer changes party size (if needed) 5. System confirms there is a table free 6. Customer confirms booking
Alternative Process	5’ Table is not available 6’ System gives the next time slot for a free table 7’ Customer selects new time and confirms booking

Sprint Planning

Sprint 1
After setting up my backlog when setting out my requirements, I then added the design tasks for my project relevant to the architecture section to this report. I have also labelled these in progress tasks with high priority to show that they need to be completed in this sprint. Once they have been completed, they can be marked as complete and disappear from the board. In review, more work needs to be done in considering how I am going to store the data, how this affects my class diagram and whether I will need an ERD and database instead.


In Progress

+ Add task

High Priority
State Diagram for Bookings

High Priority
Sequence Diagrams
- Make Booking
- Amend Booking
- Mark Customer as Covid Positive
- Set Restaurant Details

0/4

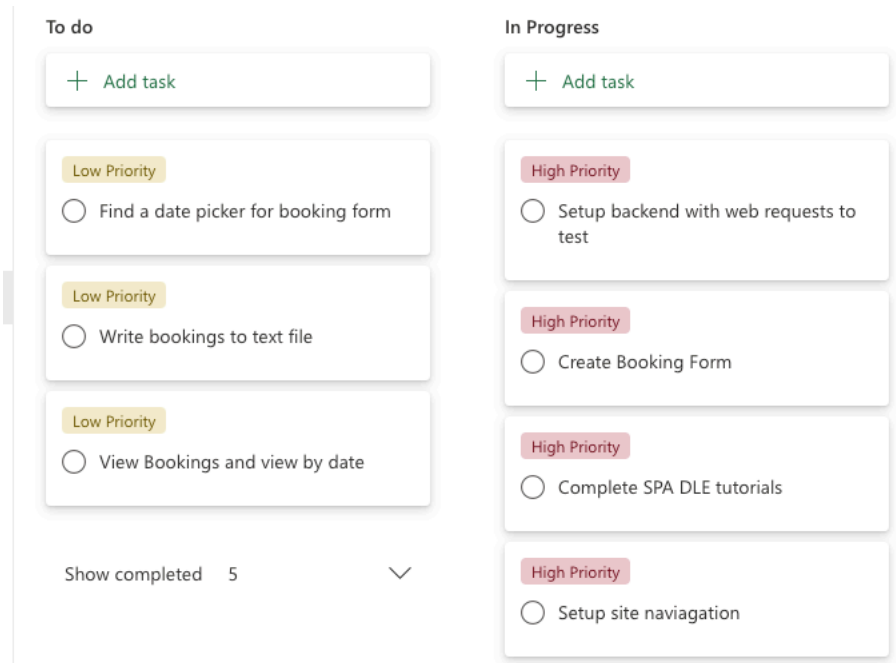
High Priority
Use Case Descriptions
- Make Booking
- Amend Booking

0/2

High Priority
Class Diagram

Sprint 2

The sprint began by working through SPA 01 – 03 to understand the basics of a single page application and commit those to my repository. I then decided to delete this project and create a new project using the ASP.NET React template. React is a JavaScript library that is widely used in web pages currently. Along with this, I'm using Reactstrap, which supplies bootstrap within React. I have then setup my planner with tasks that need completing in preparation for demoing my web app. This means basic site navigation should be setup regardless of content currently displayed and allow users to make bookings. I have also added viewing bookings, however this may not be something I get to in the current sprint. I have also added yellow labels for low priority, these are tasks that need to be completed but can be left until later.



To do

+ Add task

Low Priority
Find a date picker for booking form

Low Priority
Write bookings to text file

Low Priority
View Bookings and view by date

Show completed 5

In Progress

+ Add task

High Priority
Setup backend with web requests to test

High Priority
Create Booking Form

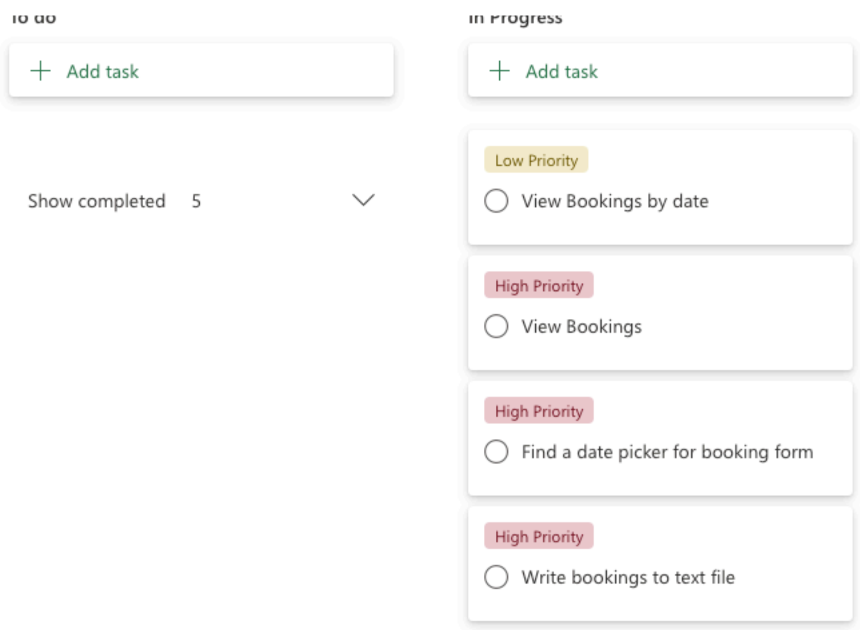
High Priority
Complete SPA DLE tutorials

High Priority
Setup site naviagation

As you can see above, I have updated my planner with the relevant tasks and hidden use cases so that it looks better organised for these screenshots. I have also included my GitHub commit history to show commits that I have made during this sprint. In review, the main thing that I need to look at to improve my next sprint is to find a date/time picker that can be used in my booking form as Reactstrap does not provide this. I have focussed on basic functionality in these sprints over looks and will consider this in later sprints.

Sprint 3

In this sprint, the focus was to write bookings to a text file and allow staff to view bookings made and filter by date, using the date picker that I have also used for the date/time selector for my booking form. I have used a library called Material-UI, an alternative to Reactstrap for components. This then meant I could easily input a booking and when submit is pressed, the booking will be written to a text file. While I have View Bookings working fine, I need to bring view bookings by date into the next sprint with me as there are still some things that could be changed with it. As currently, once a date has been selected you cannot go back to viewing all bookings.



To do

Add task

Show completed 5

In Progress

Add task

Low Priority
View Bookings by date

High Priority
View Bookings

High Priority
Find a date picker for booking form

High Priority
Write bookings to text file

Reflection

So far, I think I have done well to produce a sufficient project in the timescale provided and this will put me on track to produce a fully functioning solution at the end of the year. Looking back at my requirements, I have already met functional requirements 1 and 6 from the list provided, which were to allow customers to make a booking and only let them select a maximum party size of 6 people to meet government guidelines. Meeting that first requirement also means that I have produced a minimum viable product for the "Make Booking" use case that I have designed.

Looking back at my requirements, I do need to reconsider and refactor my class diagrams as I move through the project and implement having tables assigned to a booking and as to whether this is more suited to being placed in a database instead of each object being a line in a text file as currently, any changes made would need to be made in both the list that is created on load of the text file and the text file itself would need to be modified.

More organisation is needed so that I plan my sprints more effectively so that I know exactly what I want to have completed in each sprint and it makes it easier to review them and plan.

In conclusion, I think that I have made a good start to the project but there is still more work to be done to meet the requirements set out and to also ensure it follows legal requirements such as GDPR as currently I have nothing on pages telling customers why exactly this data is being collected and a requirement of GDPR is to ensure that users know what their data is being collected for.

References

Agile Business Consortium. 2021. *Chapter 10: Moscow Prioritisation*. [online] Available at: <https://www.agilebusiness.org/page/ProjectFramework_10_MoSCoWPrioritisation>

GOV.UK. 2020. *Venues Required By Law To Record Contact Details*. [online] Available at: <https://www.gov.uk/government/news/venues-required-by-law-to-record-contact-details>

Ico.org.uk. 2018. *Principle (B): Purpose Limitation*. [online] Available at: <https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/principles/purpose-limitation/>

Ico.org.uk. 2018. *Principle (E): Storage Limitation*. [online] Available at: <https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/principles/storage-limitation/>

Seidl, M., Brandsteidl, M., Huemer, C. and Kappel, G., 2012. *UML @ Classroom*. Heidelberg: Dpunkt.verlag, p.Page 36.

Visual-paradigm.com. 2020. *What Is User Story?*. [online] Available at: <https://www.visual-paradigm.com/guide/agile-software-development/what-is-user-story/>