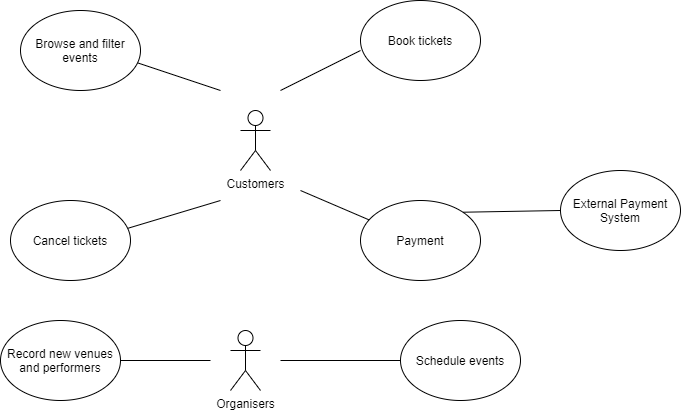
**3.1 Draw use case diagram**



**3.2 Describe use cases**

**Use Case 1**

Use case name: Book event

Primary actor: Customer

Supporting actors: Payment system

Summary: Customer books available events that they are interested in, gives required information and pays.

Precondition: There are still tickets left so that the events are available for booking and is taking place within 14 days.

Trigger: Customer is interested in viewing a show and operates on the system to browse on the events.

Guarantee: Booking information saved into system and booking status goes from “pending” to “active”.

Main success scenario:

1. Customer browses information on the events and uses filter to filter out wanted events.
2. Customer identifies an event of interest.
3. Customer books it if it has tickets left.
4. Customer provides information required by the system.
5. Customer pays.
6. Customer receives a booking number and confirmation.

Extensions:

3a. Customer orders more tickets than left and is notified.

3b. Customer wants to book for a show beyond the 14-day range which is not allowed.

4a. System checks on the information provided by customer and finds out illegal input. Customer is asked to enter the information again.

5a. Customer payment is unsuccessful and has to restart booking.

**Use Case 2**

Use case name: Schedule event

Primary actor: Organiser

Supporting actors: Performer (they have to negotiate with the organisers in advance of scheduling so that time and location can be determined.)

Summary: Organiser sets up information and schedules an event in the system.

Precondition: date and time do not clash with another event that is already in the same festival.

Trigger: Organiser has information of the event ready to be scheduled and logs into the system.

Guarantee: Event information is saved into the system.

Main success scenario:

1. Organiser provides a date and time.
2. For in person shows, organiser is informed of available venues and their capacity/3 to practise social distancing.
3. Organiser chooses a venue which schedules the event.
4. Organiser receives a confirmation sent by the system.

Extensions:

2a. For online shows, organiser provides duration, system then schedules the event and gives back confirmation.

2b. Organiser finds out no available venue returned by the system and is not able to carry on scheduling.

**3.3 Stakeholders-actors**

1. Venue owners who provide the venues such as opera theatres and stadiums. They are involved since they are paid by the organisers whenever the venues are used by the shows.

2. The external payment system which charges the customers for online ticket booking. This system is based outside of the festival organisation and has no financial involvement.

**3.4 Describe requirements**

**Non-functional requirements:**

Performance:

The system should be able to respond to at least **10,000** requests from the customers at once. Since these festivals in Edinburgh are popular and attract people from all around the globe.

Privacy:

The system should provide the customers the privacy terms that inform them how personal information will be collected and shared with any third parties. The system should obtain consent from the customers after the customers are notified of personal data terms.

**Functional requirement:**

Performance:

Optimised client-side performance is a great contributing factor towards fast loading. One of the ways to improve it is to optimise image usage by shrinking them down since there are possibly lots of images of artists and venues showing at once. Lazy loading (only load the images that you can see on the screen) should also be considered when we have pages that are long with lots of images.

Privacy:

Although cookies are fundamental to some functions on the system, they are exposed to severe risk to privacy. As sensitive information such as name, address and email address is involved via ticket booking stage, them being hijacked could lead to impersonation of the user and unauthorised access. Encryption of any cookies that contain sensitive information should be placed at top priority while building up the system.

**3.5 UML class model**

**Diagram

Description automatically generated**

**3.6 High-level description of the UML class model**

1. Singleton Design Pattern:

The “AllBookings” and “Schedule” classes both follow the singleton design pattern as there should only be one instance for each of them and they should be accessible by all other classes. An alternative is that to make organisers have their own collection of bookings that they themselves have made instead of having them as a whole. However that would make the system much more inefficient because looping over one big collection is faster than having to gather lots of collections then loop over each of them.

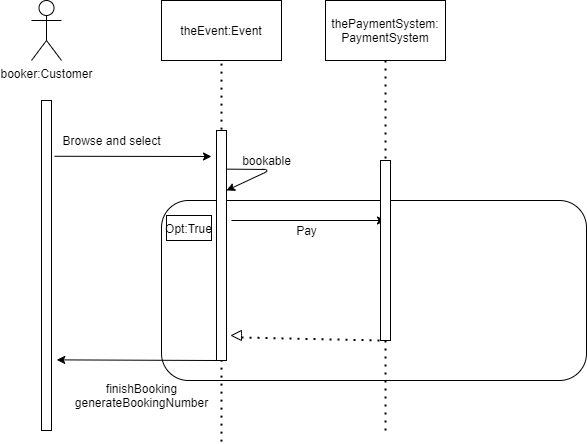
2. Increasing Cohesion:

I decided to group things that make sense together into one class. For instance, inside the class “Booking” it has everything related to booking operations within it. This will make it much easier to find things, therefore simplifying the system.

3. Reducing Coupling:

I decided to make an interface that is called “Event”. Both the online event class and in person class implement this interface. The two only share the similarity of two common auxiliary function and this way there is a very small degree of interdependence between the modules. As an alternative, It is possibly much easier to implement the two types of event as one single class however it will be much harder to fix, debug or modify a component.

**3.7 UML sequence diagram**

****

**3.8 Implementation**

I used consistent JAVADOC commenting and documentation as well as space indentation all along the program. I avoided some obvious comments such as the ones for getter methods.

I used consistent naming scheme in the program: camel case for the variables and capitalisation of first letter in words for the class names.

I avoided self-repetition by making auxiliary function separate so that they can be called multiple times.

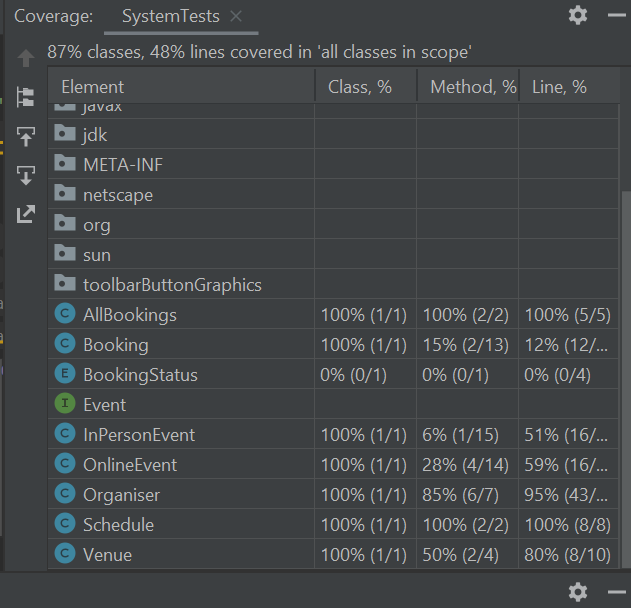
I limited line length and all the lines can be readable without scrolling right.

There are two appearances of **assertions** in my code. They can be found in the methods “isEligibleOnlineEvent” and “isEligibleInPersonEvent” under the organiser class.

**3.9 System-level Tests**

I tested the use cases which are booking event and scheduling event.

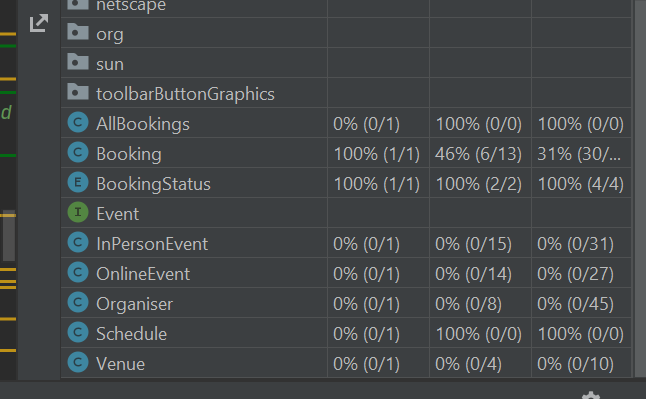
The test results came back positive.



**3.10 Unit Tests**

My tests covered

1. generating booking number
2. whether the event is bookable at the moment
3. if the name provided by the customer is valid
4. if the email address provided by the customer is valid
5. if the phone number provided by the customer is valid
6. if payment is OK

As the requirement asked, I only concerned these methods which do not use objects of other classes from the system.

**3.11 The Software Development**

Q1. Software project engineering would be more suitable in this case.

The need of generating such a software originates from a specific requirement from the organisers and the customers as COVID has made things working differently. Therefore, we need to watch closely to the needs and requirements presented by these people instead of only making some generic solutions.

Moreover, the system must be made long-lived since no-one can tell when COVID-19 ends. It is possible that the system will last 5 years or even longer. Therefore, modification and fixings should be expected throughout the time whenever it is used.

Q2. I have been using plan-driven software development process in this assignment. I was given very specified and detailed requirements and I have been programming entirely based on them. There is no agile process involved in this assignment since I was not able to communicate with any of the requirement providers thus no iteration of design is processed.

Q3. Agile process could possibly be better in this context since the pandemic situation could be very unstable. For example, few weeks of lockdown could lead much less infections in certain areas and that results in a shift of requirements. With an agile process, we are able to respond very rapidly to the changes therefore by the time the software is released, it would not seem out of date.

However agile process involves large amount of communication which has been harder to execute after COVID outbreak because of social distancing. It is possible that due to inconvenient communication the production of the software will be slowed down. An efficient and smooth way of exchanging information must be carried out when considering agile process.

Q4.

Part1:

Daily meetings are key to agile implementation. We wish to keep them short and concise, and each member should be able to explicitly state their progress of tasks and what is there to be done.

Customer approach throughout the production of the software would optimise the adaptability to customers of all ages, genders, etc. This way it is clearer what the customers expect and the problems that might arise while experiencing the software. These problems should all be fixed before the software gets released and it is better to figure them out earlier than later.

Part2:

It would be useful to have a project management software specifically built for development teams. It is key to keep everyone up to date and informed on a daily basis. Also with such a tool, it would be more efficient to share workflows and user issues.

A version control tool is a great practise in agile process as well since we need to track and manage changes to the software code. With daily iterated requirements, we often find things that are no more suitable after updates so a version control tool would come to play to revert to any previous version that we want.

4 Professional Issue Tasks

The practice of buying and reselling tickets for profit has always existed. The so-called scalpers would reserve popular concert tickets as soon as they go on sale in order to resell them for a profit. According to Derek Beres(2019), The secondary ticketing market is predicted to grow to $15.19 billion next year. What is scarier is that none of the artists and venues see this revenue—it all goes to scalpers and ticketing agencies.

As of today, in the digital era, the scale of scalping has increased substantially. This is due to multiple reasons. The scalpers enter and exit the ticket markets with sophisticated technology to harvest the tickets at regular price within usually seconds. It wouldn’t be hard to imagine such famous festivals that attract viewers from world-wide could potentially be huge targets to the scalpers. Since we have shifted the ticket platform to virtual, the ticket [bots](https://www.imperva.com/learn/application-security/what-are-bots/) that are automated software used to purchase tickets in bulk will become an operative tool for them. In fact, according to statistics(imperva), in most markets, over 40% of all online ticket booking is now done by automated software, in order to be resold later. Moreover, scalping bots are usually considerably cheap and easy to run in contrast to very high return on investment. As a solution which could be implemented at the booking stage, an anti-bot page that pops up before payment should be effective to most of the bot-driven scalping.

Another fundamental cause is that there has been very little regulatory control and seldom prosecution of scalpers, allowing them to continue operating unrestrained. In addition, lack of the original price, seat location recorded on the tickets helps them even more. In our case this would be less of a problem because all the price is the same. However, we do need put effort into legislation research to seek help lawfully.

According to economists (2017), the existence of a secondary market - where tickets are resold - is a sign that they have been undersupplied, underpriced or a combination of the two. The bad news is that in our case we are not able to stop undersupplying due to the outbreak of COVID-19. In fact, there are only 100 tickets available to the online event and the in-person event capacity has shrunken down to one third of its original size because of the practise of social distancing.

The performers are big victims from the shameful behaviours of the scalpers. There are cases where an artists’ show has been sold out for months yet looking in from the side of the stage, it appeared like about a fifth of the stadium was vacant. Bot-managed scalping is also a major concern to the fans. Elderly people can be too trusting to identify a scalper from the legitimate seller resulting in financial loss.

In conclusion, the threat of scalping to our stakeholders is immense and we are not in the best position right now to fight away from it.

reference:

<https://bigthink.com/politics-current-affairs/scalping>

<https://www.imperva.com/learn/application-security/ticket-scalping-bots/>

https://theconversation.com/the-economics-of-ticket-scalping-83434