Object Orientated Analysis & Design

CO560 – 20S1

For Bucks Centre for Performing Arts:

To create a system for the purpose of selling tickets for events and shows.

Luke Martin – 21610505

Owen Perry – 21905958

Date Submitted: 17th December 2020

Final Submission Date: 18th December 2020

Table of Contents

Part 1

Introduction to The Case Study

The Case study that we are working with is that of the Bucks Centre for Performing Arts. They require a system to allow them to operate their business online. This entails various essential features such as adding events and shows, holding and purchasing tickets, and manging various promotions. This project will cover the inclusion of multiple actors with various and differing use cases for the system.

As designers/analysists we will be aiming to produce detailed documentation that will clearly show the relevant information that was gathered from the brief in an effective way that clearly demonstrates these various aspects of the project. This will be done through different diagrams and such, which will be made using the UML framework to model these concepts clearly and thoroughly. By the end of the project the client will have access to these resources that we have generated according to the brief that we have been provided with.

Requirements Table

No.	Use case	Description
1	Login	To save your data and have an account that you make purchases through.
2	Verify information	To ensure that the account is secure and make sure the person logging in is the owner of the account.
3	View list of upcoming events	To show the customer a list consisting of all the upcoming events that are being hosted.
4	View scheduled shows by date	To show the customer a list consisting of all the upcoming shows of a given event in consequential order by date starting by the soonest show.
5	Display seat price	To show the customer a chart of the seats in the venue, along with which ones are available, which ones are not, as well as the different prices that the different seats in the chart are listed as costing.
6	Select seats from a seating chart	To allow the customer to select seats from the chart of available seats.
7	Hold seats	To keep seats held so other users to not take them while the customer is making their purchase.
8	Purchase seats	To pay for seats that have been selected and held by the customer.
9	Order tickets	To pay for seats that have been selected and held by the customer.
10	Manage promos	To allow the venue manager to oversee promotions, as well as accounting for differing prices such as having child, student, adult and senior ticket prices.
11	Add	To allow the Venue Manager to add new events and shows to the system.
12	Reschedule	To allow the Venue Manager to reschedule events and shows already in the system.
13	Cancel	To allow the Venue Manager to cancel events and shows already in the system.
14	Change max seats	To allow the venue manager to change the maximum seats per Customer value in each show.

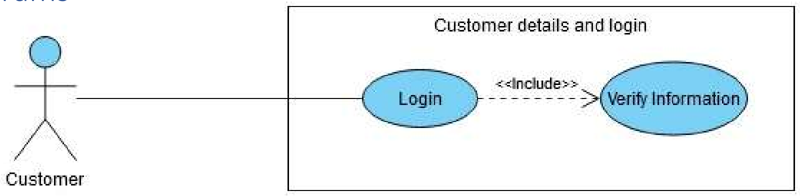
“High” Level Use Case Descriptions

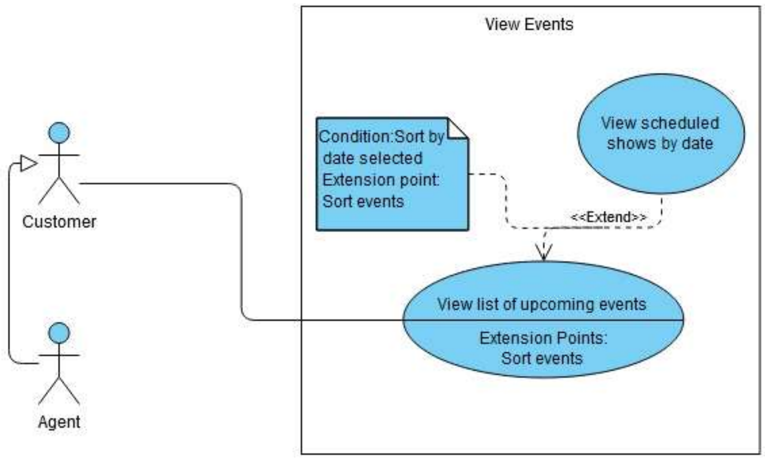
Use case	Description
Login	The customer will be prompted to provide login details when they attempt to do anything on the system such as hold a seat, purchase a ticket, etc.
Verify information	After the customer provides correct login details, they will be prompted to verify their account by providing some form of information linked to it, for example a security question, or verification number sent to the email attached to their account.
View list of upcoming events	Shows the customer a list consisting of all the upcoming events that are being hosted.
View scheduled shows by date	Shows the customer a list consisting of all the upcoming shows of a given event in consequential order by date starting by the soonest show.
Display seat price	A chart of the seats in the venue is displayed to the customer along with which ones are available, which ones are not, as well as the different prices that the different seats in the chart are listed as costing.
Select seats from a seating chart	The customer can select seats that are available on the seating chart, ready to be held, purchased, or deselected by the customer.
Hold seats	Allows the customer to save a selection of seats so they are not lost or purchased by someone else while the customer does other things with the system, these seats will also be displayed as unavailable to other users of the system while being held.
Purchase seats	The customer can pay for a collection of seats that they have selected or held by either inputting their banking information or using pre-saved banking information that is linked to their account.
Order tickets	The customer can pay for a collection of seats that they have selected or held by either inputting their banking information or using pre-saved banking information that is linked to their account.
Manage promos	The Venue Manager is responsible for setting promotions and discounts for each show. This sets the priority structure for seats, accommodating different prices for Adult, Student, Child, and Senior citizens. Promotions can be unique to each separate showing and only for specific seats within a show. They can also be re-used.
Add	The Venue Manager can add new events and shows to the system.
Reschedule	The Venue Manager can reschedule events and shows already in the system.
Cancel	The Venue Manager can cancel events and shows already in the system.
Change max seats	The venue Manager can change the maximum seats per Customer to accommodate for demand, or if a Customer wishes to pay for a large group of people.

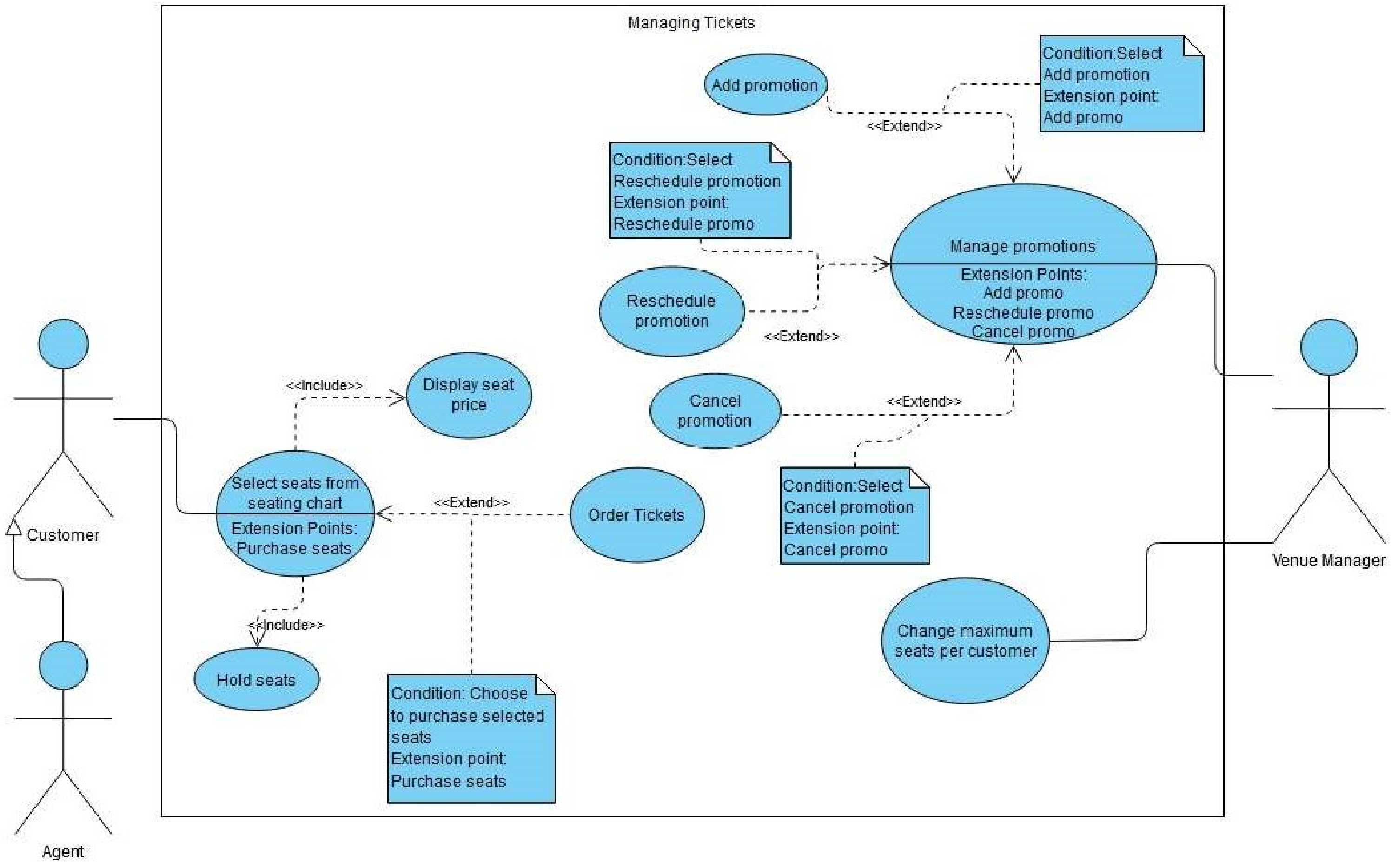
Actors Table

Actor	Description
Customer	The Customer is anyone who is using the online system to view events for the Bucks Centre for Performing Arts and can purchase tickets for them.
Venue Manager	The manager of the Bucks Centre for Performing Arts. They are responsible for the organization of the shows / events, their pricing and promotions and setting the limitations of the maximum number of seats per customer.
Agent	Agents are given a contract which allocates them their designated seats to sell along with the Terms and Conditions to follow while doing so.

Use Case Diagrams

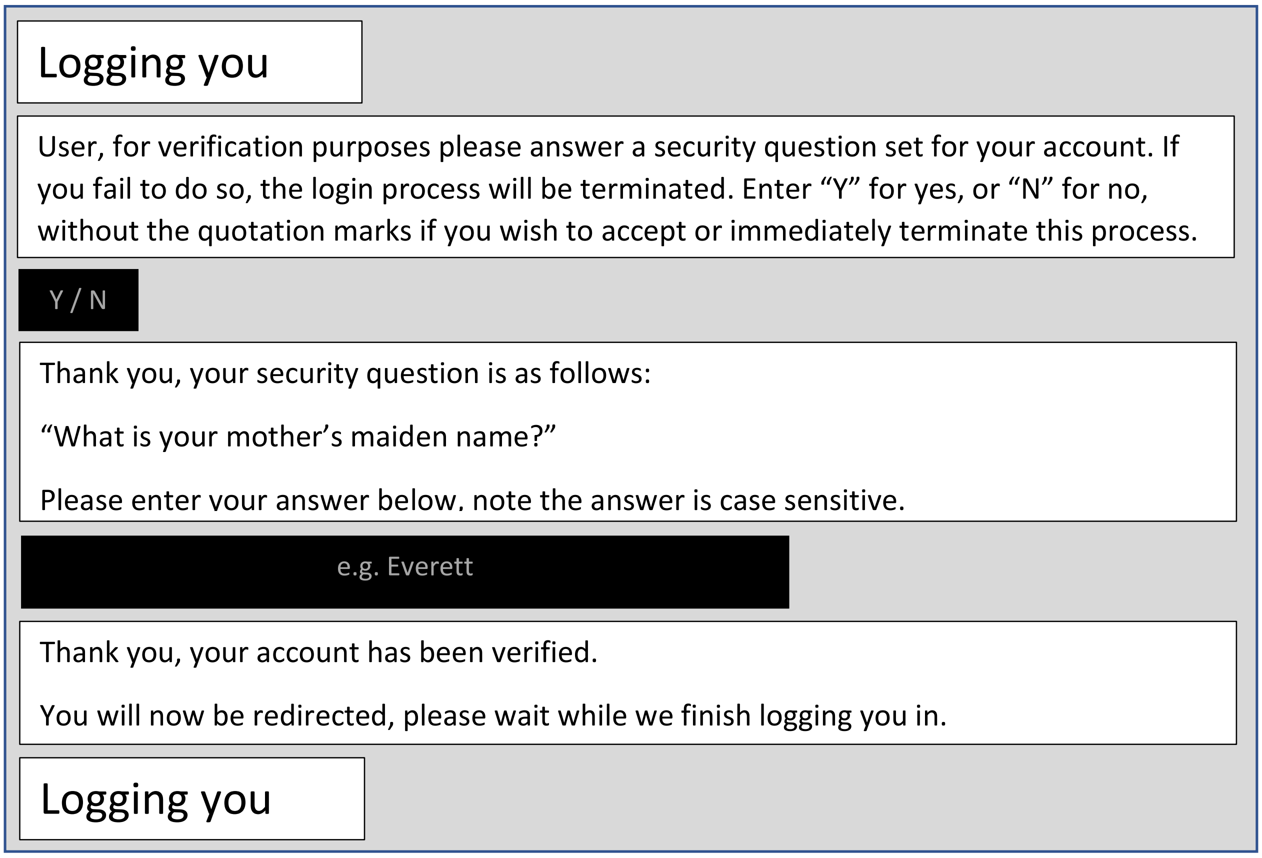



View Events
Customer
Agent
Condition:Sort by date selected Extension point: Sort events
View scheduled shows by date
<<Extend>>
View list of upcoming events
Extension Points: Sort events


Managing Tickets
Add promotion
Condition:Select Add promotion Extension point: Add promo
<<Extend>>
Condition:Select Reschedule promotion Extension point: Reschedule promo
Manage promotions
Extension Points: Add promo Reschedule promo Cancel promo
Reschedule promotion
<<Extend>>
Cancel promotion
<<Extend>>
Condition:Select Cancel promotion Extension point: Cancel promo
<<Include>>
Display seat price
Select seats from seating chart
Extension Points: Purchase seats
<<Extend>>
Order Tickets
<<Include>>
Hold seats
Condition: Choose to purchase selected seats Extension point: Purchase seats
Change maximum seats per customer
Customer
Agent
Venue Manager

Screen Mock-Up’s

Mock-Up: Verify Information



Logging you

User, for verification purposes please answer a security question set for your account. If you fail to do so, the login process will be terminated. Enter “Y” for yes, or “N” for no, without the quotation marks if you wish to accept or immediately terminate this process.

Y / N

Thank you, your security question is as follows:

“What is your mother’s maiden name?”

Please enter your answer below, note the answer is case sensitive.

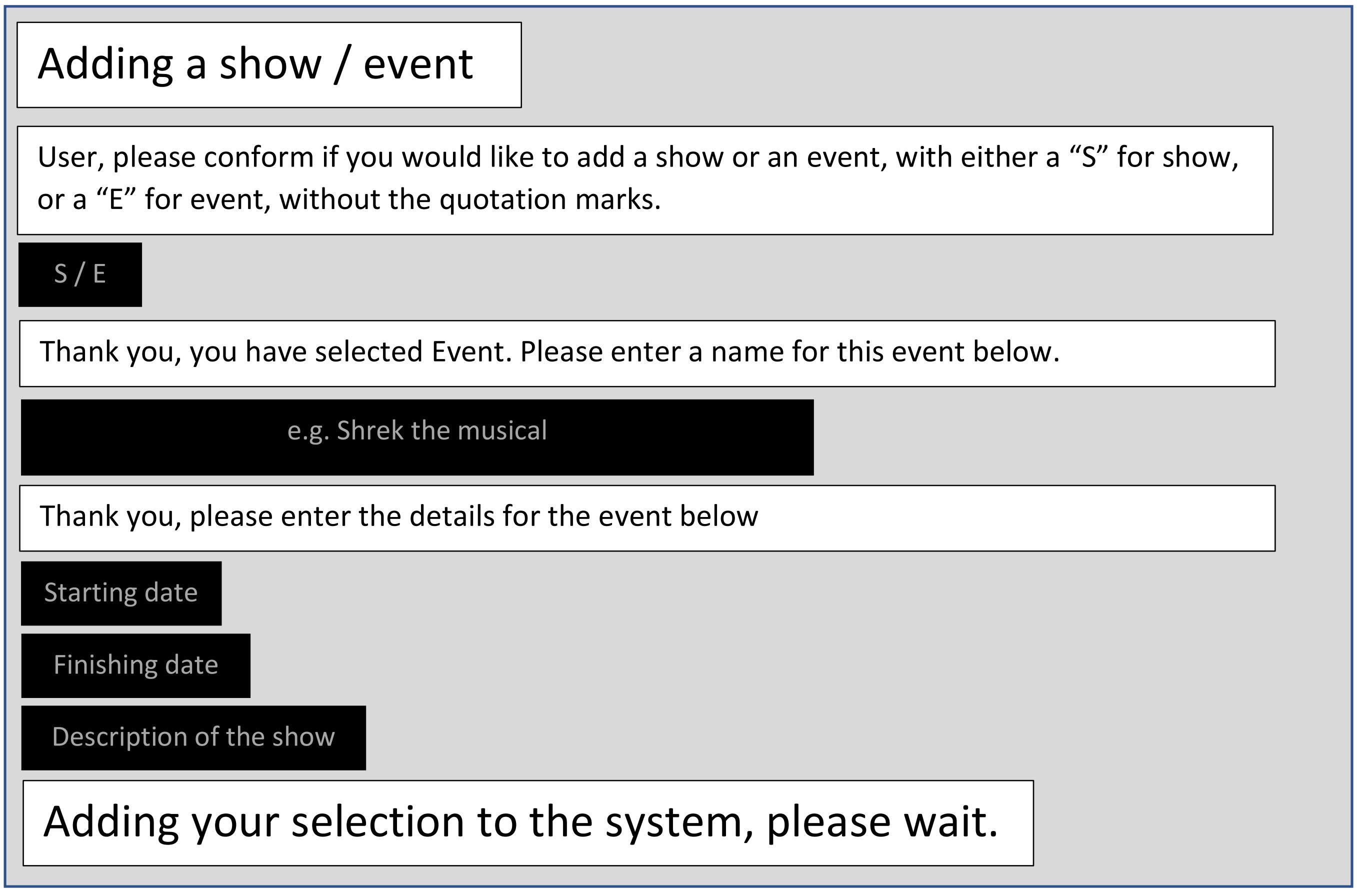
e.g. Everett

Thank you, your account has been verified.

You will now be redirected, please wait while we finish logging you in.

Logging you

Mock-Up: Add Show



Adding a show / event

User, please conform if you would like to add a show or an event, with either a “S” for show, or a “E” for event, without the quotation marks.

S / E

Thank you, you have selected Event. Please enter a name for this event below.

e.g. Shrek the musical

Thank you, please enter the details for the event below

Starting date

Finishing date

Description of the show

Adding your selection to the system, please wait.

"Low" Level Use Case Descriptions

Use Case: Verify Information

Actor Action	System Response
1. Enters correct login information	2. Prompts user to answer a security question
3. Accepts prompt	4. Security question is displayed
5. Enters answer	6. Presents message displaying the user has been verified

Use Case: Add Show

Actor Action	System Response
1. None	2. Prompts user to choose to add either a show or an event
3. Makes selection	4. Prompts user to name it
5. Enters name	6. Prompts user to enter details: • Date • Time • Description
7. Enters details	8. Adds the show / event to the system

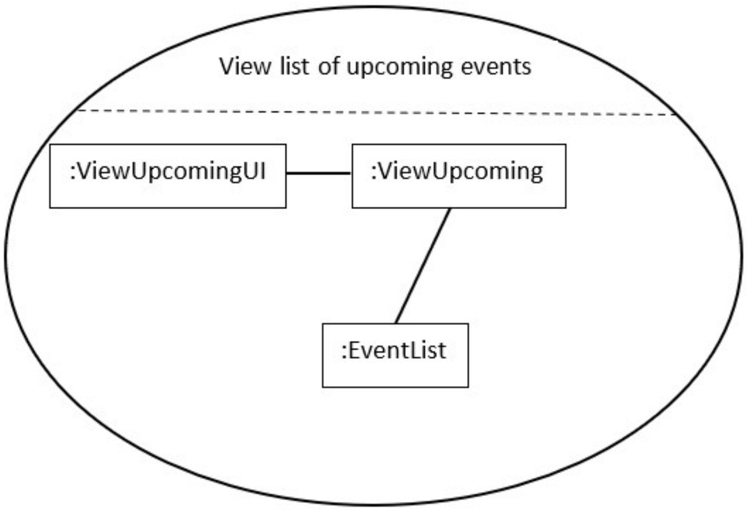
Glossary

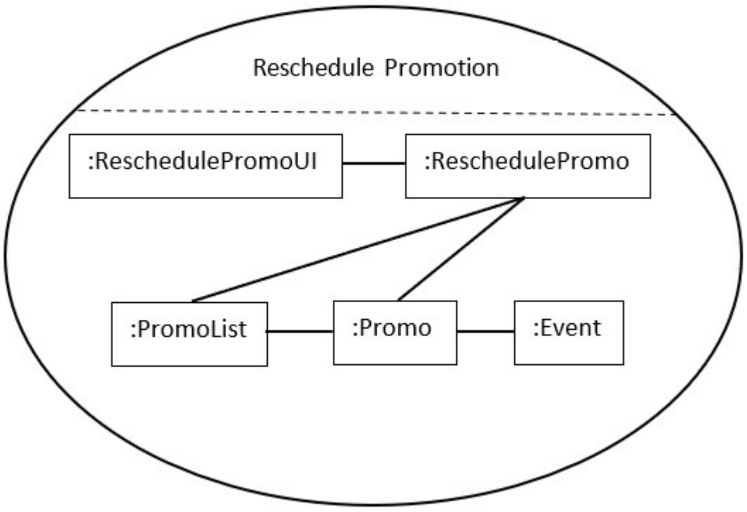
Term	Description
Customer	A person who intends to use the system for its intended purpose, e.g., buy tickets for a show
Venue manager	A person who oversees and manages the various events and shows on the system, as well as dealing with situations that require administrator permissions, for example hanging the maximum number of seats per customer.
Agent	A person who will be assigned seats at a show attached to an event that they are to try to sell to the public.
Bucks Centre for Performing Arts	An institution that puts on shows for the public who purchase tickets for them. They host events which are to be shown for a variety of dates.
Event	A given production. This will be performed in multiple shows.
Show	An instance of an event that is one of many for said given event.
Actor Action	What a given actor does in line with the system.
System response	What the system does in response to an actor's actions within it.

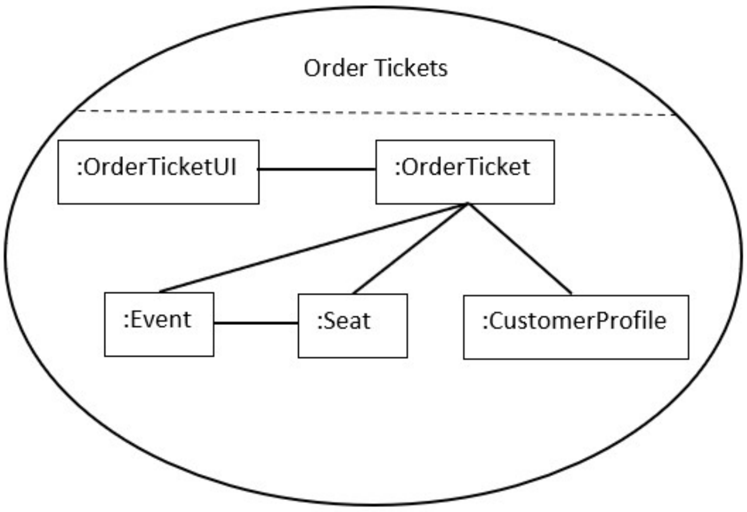
Part 2

5C's method

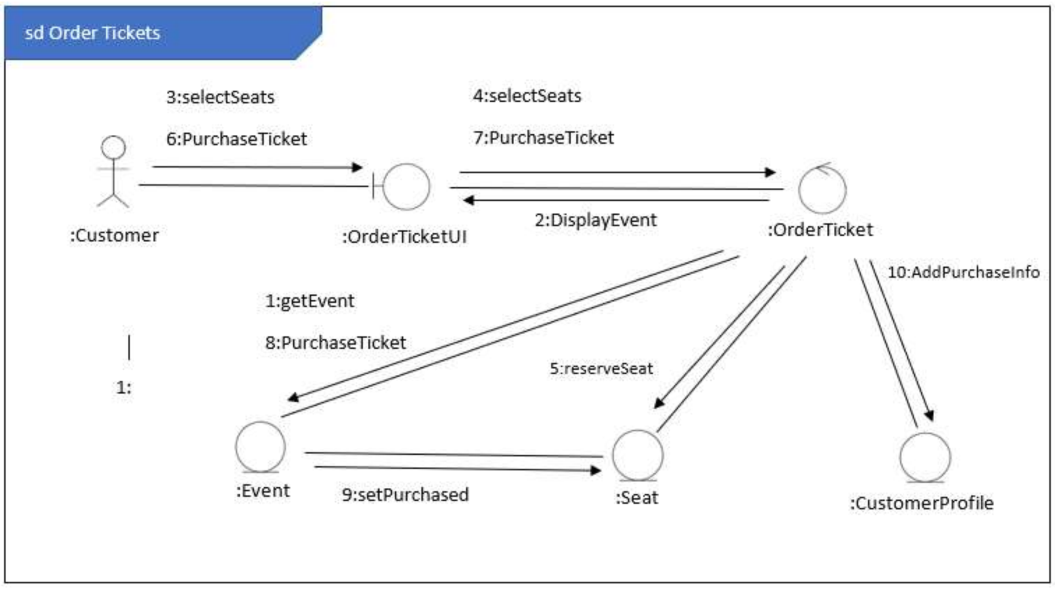
Collaboration Diagrams

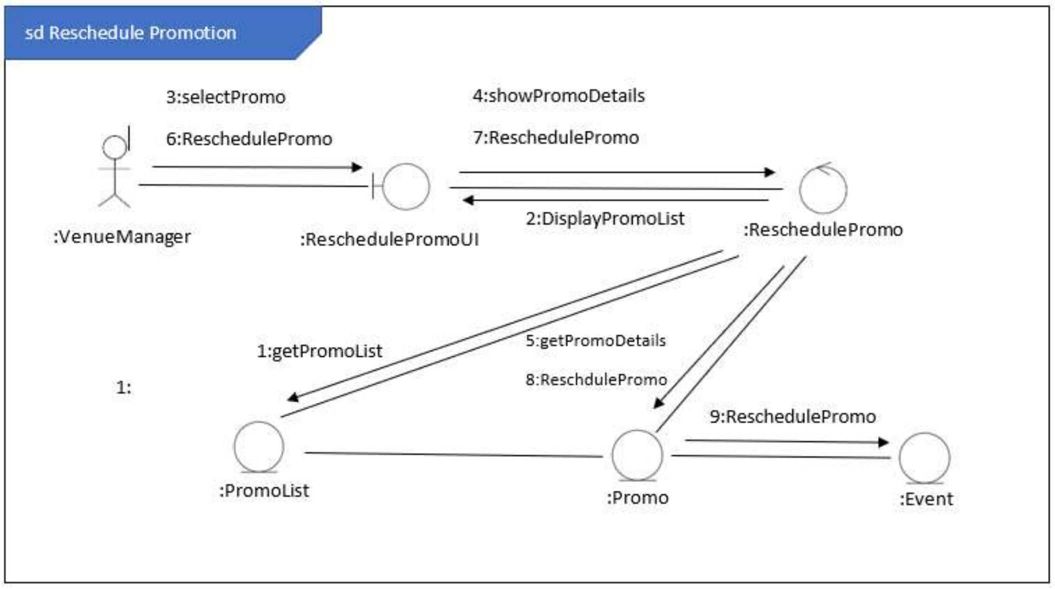


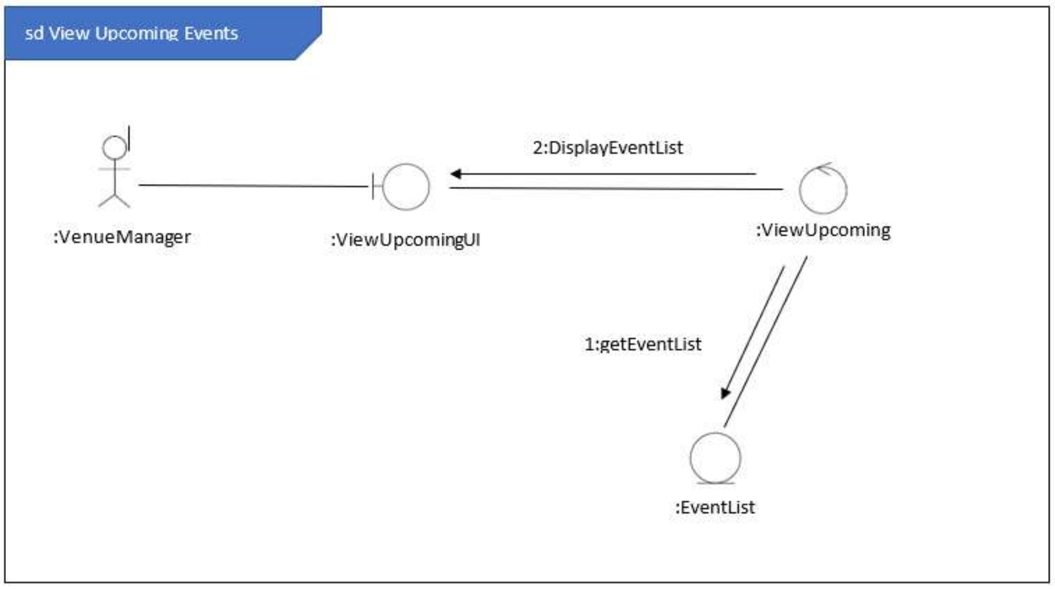




Communication Diagrams

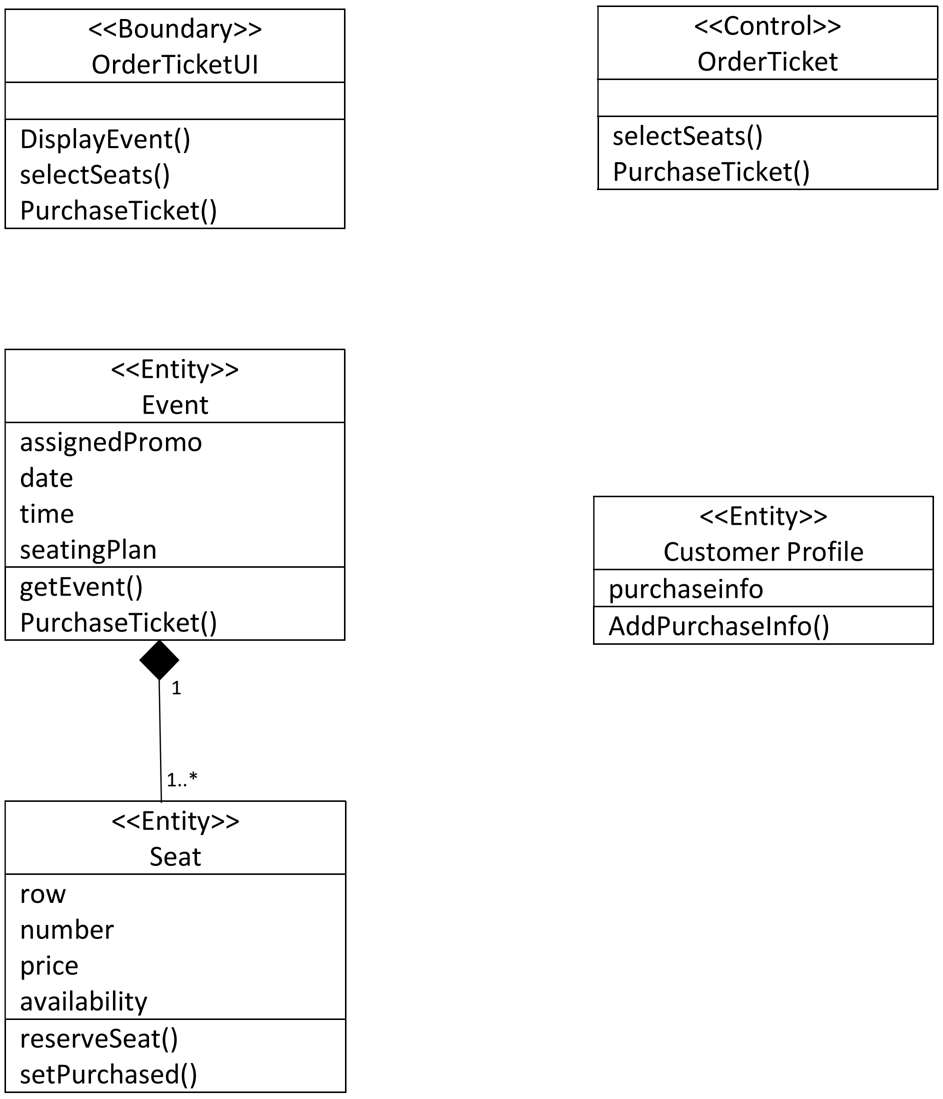




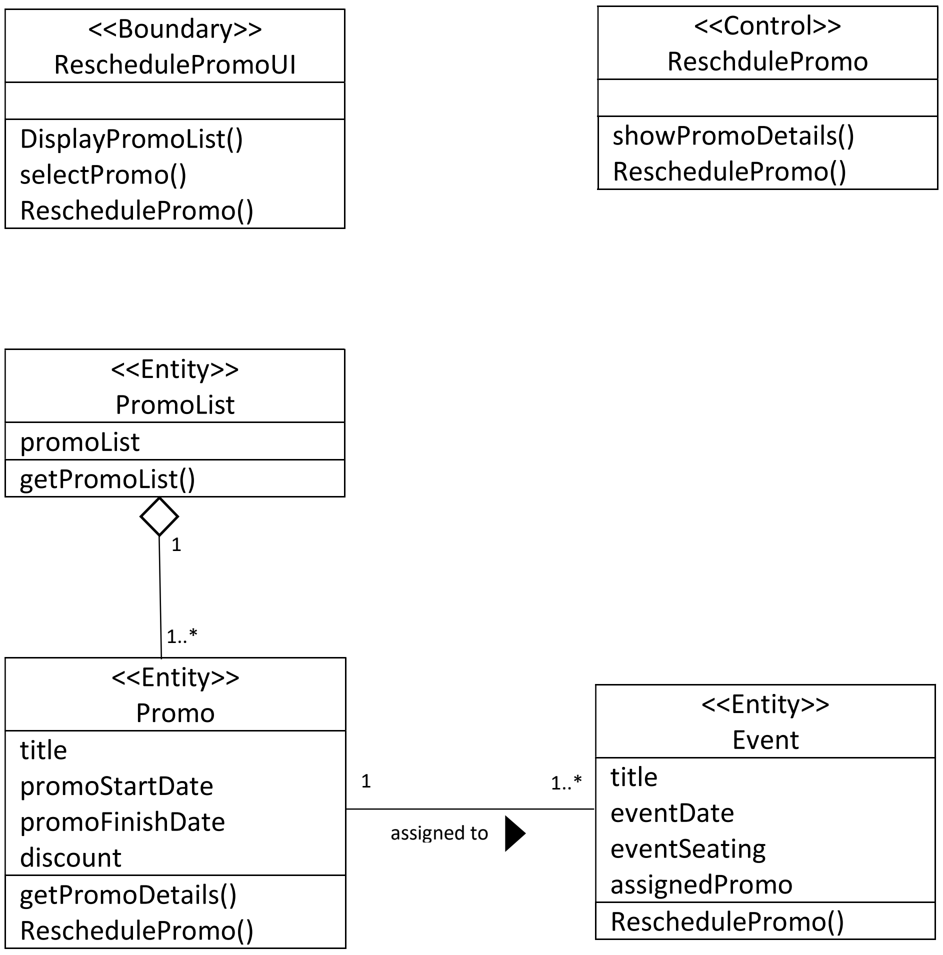


Semi-Class Diagrams

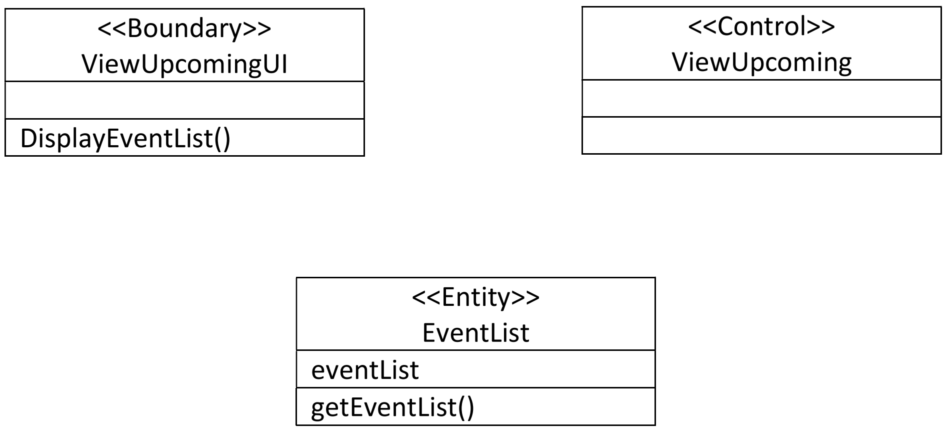
Order Tickets



Reschedule Promotion



View Upcoming Events



CRC Method

CRC Cards

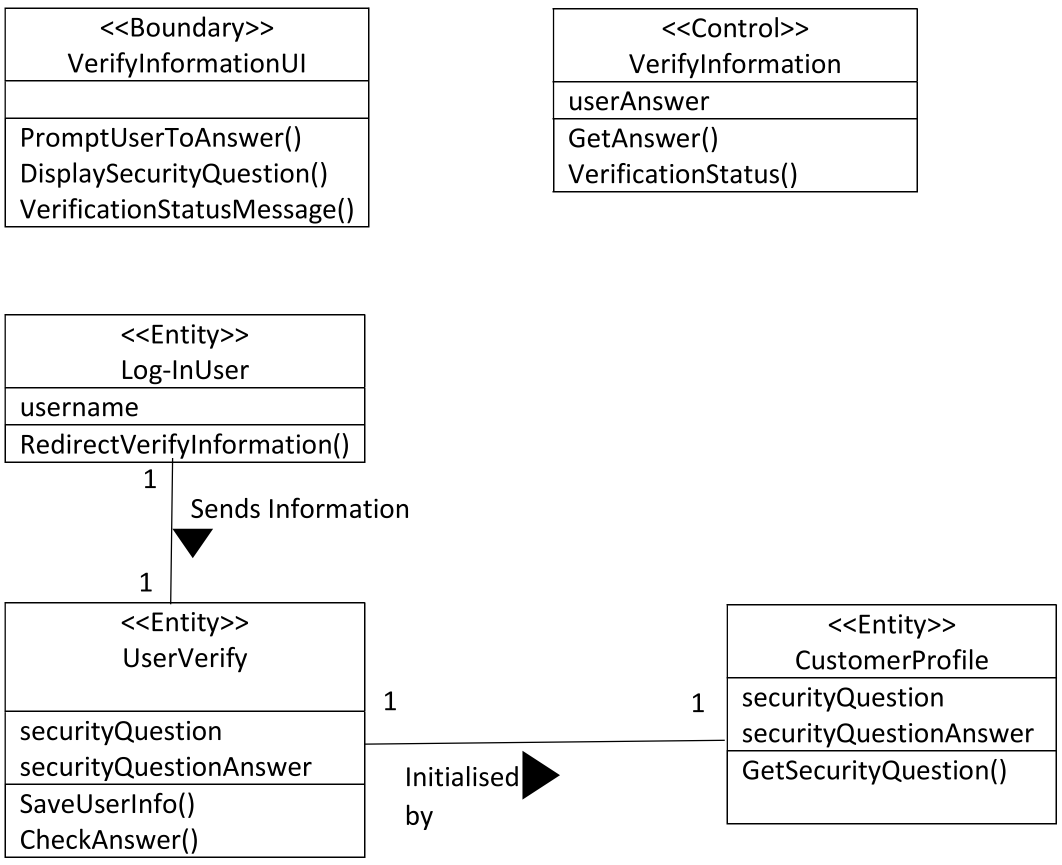
Log-inUser	
Store Users log-in details **Redirect user to verify their information**	Initiates VerifyInformation

VerifyInformation	
Constructs UI **Displays messages** **Gets responses**	UserVerify provides the security question User information provides information on if the answer was correct or not

UserVerify	
Make a temporary save of user information **Collect and store relevant security question and answer** **Check the users answer against the saved answer**	Log-inUser provides the user details CustomerProfile provides the security question and correct answer VerifyInformation provides the users answer

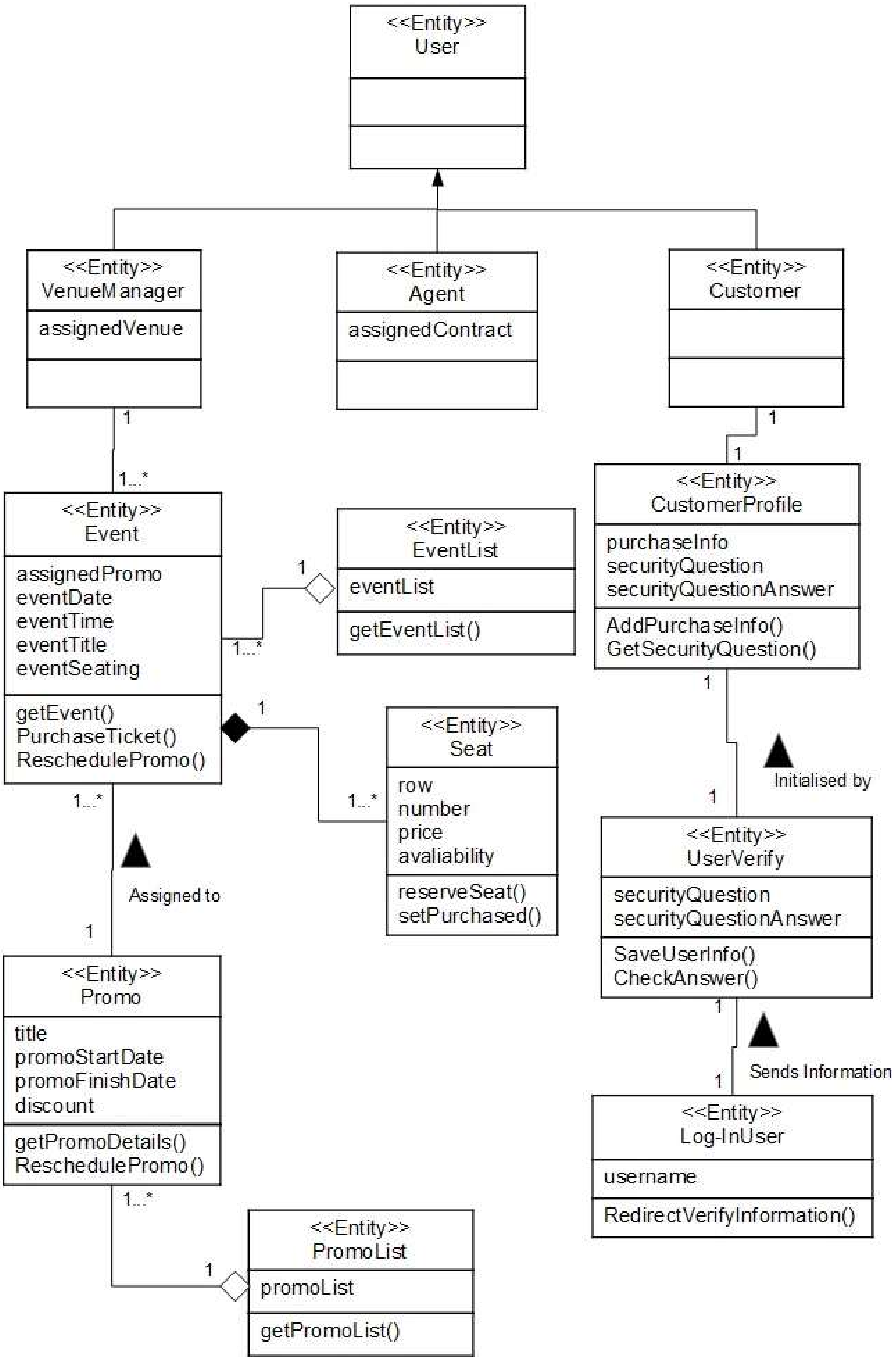
CustomerProfile	
Keeps all data from all users who have an account on the system	UserVerify provides user details to check

Resulting Semi-Class Diagram



Part 3

Class Diagram



Part 4

Conclusion

Throughout part one of this project the goal was to make sure that by the end of it we had a firm understanding and high comprehension of the brief. This entailed making sure we had nailed down all the relevant actors as well as making some crucial use cases in line with the brief that the system could theoretically accommodate. This was achieved effectively, and we believed it set us up well for the rest of the project and led to a higher level of understanding which helped us create a more relevant set of classes. As well as this we aimed to come out of it with solid foundations in the technical specifications that were relevant. This was achieved through the early stages by going through the brief looking for certain attributes, key phrases and uses of language that could later be developed upon and fleshed out into things like functions and variables, not just classes, over time.

Part 2 of the project went accordingly. One of us developed some class diagrams using the 5C's method. This gave us Collaboration diagrams, Communication diagrams, and finally Semi-Class diagrams for 3 separate use cases. They were detailed and allowed us to see the ways in which the classes would interact and perform their functions in relation to the relevant use cases chosen. The other person used the CRC method, developing CRC cards that showed the classes acting, the functions they were performing and the ways in which they communicated with other classes in the process. Through this a Semi-Class diagram was developed in accordance with the CRC cards that were generated, again showing the relevant, boundary, control, and entities within the process of the use case.

Part 3 of the project went smoothly. To complete it first the classes were taken from the three 5Cs models and the CRC models and any of the same or similar classes were consolidated. Then the "user" class was added to act as a base for the three actors to allow ease of use when any changes or updates are needed down the line. Finally, the prior class diagrams and their respective links were taken and worked on from the top down, taking the classes linked closest to the actors and adding them first (Event and CustomerProfile). Then the classes were as shown in the initial diagrams.

A benefit of using the OO approach, specifically for this project is for one that it can be developed on a component basis. This means that it has high potential for re-using existing code from other projects to make the system. This allows us to cut down on time taken to complete the system. This is important because it also helps with cost because we are not having to make all new code and components for the project when things from other projects could effectively work and fulfil the roles that are required of the system being developed. This does however have its drawbacks. For example, depending on how long ago the existing code was written, it could potentially be legacy code that does not have the security that the latest software does, along with this you are also inheriting all and any of the bugs stemming from this code. So, while you may be saving time by re-using code, you could end up spending twice as long trying to iron out bugs in code that has not been looked at in years.

In the project we achieved inheritance successfully. For example, all three of the actors identified are linked to one generalised "user" class. This was done as to allow adjustments to be made to both the specific actors themselves as well as the base attributes that they all share that are stored within the generalised "user" parent class.

Encapsulation, while perhaps a little harder to visualise was achieved here. You can see that data is kept protected by its relevant class. For example, in the CRC method there is a dedicated "UserVerify" class that handles all the potentially sensitive data. Information is pulled from the "CustomerProfile" class since that was also protected, this was done because the data was just needed temporarily to hold and save both the relevant customer security question and the security question answer.

To achieve polymorphism, you will see through the class diagrams we have kept some of the function names the same while they get passed through multiple classes, allowing for said classes to use them differently for their respective use cases. Allowing us to save time from having the same function written in every class, additionally making the system more efficient.

So far as coupling goes. We have ensured that only those classes relevant to each other, specifically that are needed for the function of a use case are linked, and only when necessary so that the use case cannot be completed if they are not linked. While this does add some time to the analysis and design side of the project it does again make the system that little bit more efficient and means that if one thing breaks then it will not start a chain reaction leading to multiple classes and thus use cases ceasing to function properly. Thus, we have spent time ensuring that our classes all have low coupling unless it is necessary to the functioning of the system.

During the design and analysis segment of the project we ensure that while making the various diagrams and models that have been displayed in this document that the content of the individual classes is as cohesive as possible. This meant combing through them more than once ensuring that all the things in them are only there because they are relevant to that class, and furthermore that that class is in of itself relevant to a use case that we were focusing on. This means that in the case that there are things that need changing with the system in the future that the code is easy to navigate and everything that is needed is in its correct place and that the code itself is not too bloated thus making it more difficult to comb through it and look for bugs, and less coding general means fewer potential points of failure in the logic, thus fewer potential bugs in the system.

We would like to finish this document off by saying we think this system has great potential. Not to mention that we things that it specifically would work well with an object orientated approach. There are areas in which the brief made it more difficult as well as less difficult to fulfil this in various places, however we believe, and thing we have adequately displayed that this approach is a very viable approach and a recommended method for the further development of the system being made for Bucks Centre for Performing Arts for the purpose of selling tickets for shows and events.