Bucks Centre for Performing Arts:  
Online Ticketing System

By Thomas Golbourn, Student ID: 21614843.  
Harvey Rush, Student ID: 21518237

# Introduction

The Bucks Centre for Performing Arts would like to enable their customers to order tickets through the internet. They have presented us with a document outlining the system they want. This document contains all the known requirements of the system from a functional perspective. Some of these functions lend themselves to non-functions as well. Here we will be outlining the requirements presented in the document given, both functional and non-functional, so that we can begin analysing them.

# Capturing Requirements

In this case, most of the requirement capture has been done for us, as we have been given a very specific outline of how the software should operate in a document. However, where there are gaps, or ambiguity in the document, requirement capture techniques can be used. These techniques can be broken down into 5 categories; interview, observation, document sampling, background reading, and questionnaire.

## Background Reading

In this project background reading is not entirely needed if the project is built to be scalable to suit a large theatre organisation with multiple venues, shows, and thousands of users. However, if building it for a smaller scale, then background reading would need to be undertaken to gauge how small a scale might be acceptable. We can also use background reading as a technique to learn about existing online ticket systems for theatres. This can help us avoid obstacles others have faced in the past. A major concern about background reading is that sometimes relevant or up to date information can be difficult to find, and can even have red herrings within them. In any case due to the unfamiliarity with the organisation background reading should take place to begin to understand the fact-finding questions we’ll need to ask, and to understand the business we’ll be working with.

### Results of Background Reading

The first part of the background reading was discovering any online presence, and how they currently book tickets through that. Their website is <http://bucksperformingarts.blogspot.co.uk/> and is quite simplistic. From this site we can see their most recent show used the Courtyard Theatre in London, but they have used Wycombe Town Hall, and multiple other venues in the past. This means we will absolutely have to accommodate multiple venues in the software. Their most recent method of ticket booking was via the Courtyard Theatre’s use of TicketWeb, with an alternative to phone and book tickets directly with the theatre. However, they have used other theatre’s own payment methods, such as Norden Farm theatre’s website, in the past. This could be the reason why they need their own system, as the inconsistent booking methods are not working for them.

## Interviewing

In this project interviewing is a readily available tool, as there are regular meetings with the client, who is also well-versed in computer science and programming. With just one point of contact there may be bias, but at least there shouldn’t be any conflicting accounts. Under normal circumstances it would be costly and time-consuming to conduct the interviews and transcribe the audio or compile the notes in a meaningful and useful way. However due to the regularity of the meetings, the knowledge of the client, and the university funding them, these disadvantages do not apply quite as much. Due to this interviewing is one of the most powerful tools at our disposal in this project.

### Interview Questions & Answers

Interview questions are shown in bold italics as headings with the paraphrased and reformatted answer indented afterwards. These were paraphrased and written down on-the-fly and agreed upon with the client during the interview as acceptable answers to the questions.

#### The document isn’t clear on which ‘customer information’ an agent must provide to purchase a ticket on a customer’s behalf, could you clarify which information should be given and how consent should be handled?

Consent from the user will be asked for when signing up, prior to entering credit card or personal information. This consent will explicitly state that agents may book on their behalf at their request whenever the requested tickets become available. Furthermore, it should state that the payment will be automated and can be subject to a processing time leading up to and including within 24 hours of the event, starting from the payment date. An agent needs to be able to find the customer in the system via any identifier or search term, such as the customer’s name or address. A unique identifier such as an email address, customer id number, or username, should also be available to use. Once an agent has identified the customer they should be able to make the purchase for them without knowing any further details about the customer. The customer should then be notified of this via email.

#### The document has no mention of target operating systems or hardware, do you have a preference? Are there hardware or software considerations on the server end we need to account for, or systems in place we need to integrate with?

The target operating system for the customer, venue manager, and agent is Windows 7, Windows 8, and Windows 10. The server should be hosted on a Windows Server 2012 machine. Payment systems will be set-up via a phoneline connected specialist machine. This machine has a local address on the network with the server and takes inputs at a rate of up to 10,000 per second in the following format:   
 [Unique ID (Int), CardNumber (Int), CVC Code (Int), Expiry (Int (MMYY), Charge (Float (####.##))]  
These inputs simply need to be parsed to a plain text file, line by line, on the server machine, in a windows network accessible location. The specialist card machine will read the inputs from there once it has been configured by the BCPA staff. Once it is running it will generate a file named ‘output.txt’ and a file named ‘log.txt’ in the same directory as the input file. Within the output file the results of operations parsed to input will be shown in the following format: [Unique ID (Int), DateTime (DateTime), Status (String)]  
The log file will show these inputs and outputs in this format: [[InputData], [OutputData]]  
The email system should use a text file in this same location called ‘email.txt’. This should be fed data line by line, after each line is read it will be deleted, the format specified is:  
[Email Address (Str), First Name (Str), Last Name (Str), Ticket Info (Any Multi-Line String)), MessageTemplateName (Str (Available Types: ‘PurchaseConfirmed’, ‘PurchaseProcessed’))]

#### Roughly how many employees does the BCPA have, and expect to have during the software’s expected lifecycle?

Less than 50 on both accounts.

#### Roughly what peak number of concurrent users do you think the software see during it’s expected lifecycle?

Between 100 and 600 concurrent users.

#### Who will be the Data Protection Officer should we need one, and will they ensure and finance the projects adherence to all current and future data privacy and e-commerce laws?

Yes, we will cover the cost of adherence to all data privacy and e-commerce law during the software’s development and lifecycle, and I am already the appointed Data Protection Officer for this software.

### Interview Conclusions

The key take-aways are: Mail and credit card charging systems are easy to use and access. Data privacy and e-commerce law adherence will be overseen by the client. The software should handle over 600 users at once.

# Non-Functional Requirements

Non-functional requirements are requirements that are concerned with system design rather than use. For example, the use of the ticket system is to enable customer purchases, however, the design of the system also determines several key parts of the program. Here I have created a list of non-functional requirements that should be considered, with the reasons why.

1. **Security**: Since the program requires users to create accounts with personal and financial details, and since agents and venue managers will be able to access these details, security measures are needed. If an unauthorised user managed to access one customer, or several customer accounts via an agent or venue manager account, the result would be a disaster. Customers would no longer trust the software with their financial or personal details.
2. **Quality**: If the program is of low quality, then users may lament having to use the program and call for its replacement. Customers especially may not trust the program with their information if it appears the program is of inadequate quality.
3. **Usability:** If the software is difficult to use for any user, they will be dissatisfied with it. In the venue manager’s case, where there are a lot of ways to configure a show, usability should be considered strongly. If it is not considered, then the program could easily end up feeling ‘bloated’, inundated with buttons and text, and hard to understand.
4. **Privacy:** Since Britain is still currently operating under the European Union law, the European Union’s laws must be considered. Some of these laws pertain to data privacy, control, and processing, this concerns how data can be collected, stored, used, and transported. Compliance with such laws, directives, and regulations, should be ensured during the design, build, and use of the software. Britain also uses its own laws, or clauses within EU laws, pertaining to data privacy, storage, transportation, and processing, that must be considered alongside EU law. In May of 2018 new EU legislation is coming into practice, this will also need to be considered and adhered to, along with any laws introduced by Britain following Brexit.
5. **Accountability:** Since the requirements dictate that the agent is allowed to make purchases on a customer’s behalf, actions taken within the software should trigger mechanisms that record or report such actions. This is to make the software safe to use in an online environment. If a security breach, error, or user mistake occurred and there was no accountability then discovering where, when, and how it occurred could be difficult.
6. **Configurability:** Since the software outline document does not include any seating layouts or price structures to work with, and theatres can vary these between shows, we need to ensure that the software is configurable to meet the theatre’s unique properties. These configurations will not be set-up by a software engineer, but instead by the venue manager. As such the software needs to be easily configurable on the user-level.
7. **Reliability:** If the software is unreliable, a multitude of problems can occur. Tickets may not go on sale at the right time, or the right price, seats may be oversold, users may not be able to access or use parts of or the entire system. These will hurt the business and the consumers trust in the business. This will have a ripple effect, and our software solutions may no longer be as trusted by other businesses, which could lose us clients. As such reliability is incredibly important. If a lift breaks half the times it is used, the stairs will be taken instead, regardless of how good the lift is.

# Requirements Model

## First Iteration, Basic Use Cases

|  |  |
| --- | --- |
| Actors | Descriptions |
| Venue Manager | The venue manager is responsible for the creation and editing of shows, including their prices, seating layouts, promotions, and discounts. They are also responsible for processing any refunds or amending tickets, as well as management of agent accounts. |
| Agent | The agent is responsible for booking tickets on behalf of customers. |
| Customer | The customer is able to access shows, select seats, and purchase tickets. |

### Customer and Agent



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Use Cases | Descriptions | | Sign Up | To allow the customer to execute any purchase requests they must first create an account via the sign up button, entering the critical information that will be needed to later complete a purchase. | | Login | Once a user has an account they can log in with their sign up email address and password, allowing them to access and edit account information and view order history. | | View Shows | The customer can view available shows via one of the search options either by show name, date or a list of all current shows. | | Select Show | The customer can then select their desired show to begin viewing extended show details and ticket information. | | Select Seat(s) | A seating plan for the venue will be shown, displaying all seats with information on availability and price ranges, they can then be selected by the customer to create a purchase request to be executed by the agent. | | Buy Ticket(s) | The customer must then enter their card details or select saved card details to be passed on to execute the order once available. | | Logout | The customer is then free to log out and their part of the purchase process is now complete. | | Find Customer | The agent will now locate and load customer details for his allocated list of purchases. | | Buy Ticket(s) on Customers Behalf | Once the customer is loaded and tickets are ready to be purchased, the agent will complete the purchase on the customer’s behalf and the customer will be automatically notified. | |  |

### Venue Manager



|  |  |
| --- | --- |
| Use Cases | Descriptions |
| Login | The venue manager will log in with his email and password, which are allocated special privileges and a separate interface from agents for management uses. |
| Add / Edit / View Show(s) | In the venue managers interface he will have access to creating new shows, editing existing shows or simply viewing shows the way an agent would.  Creating a new show requires entering details such as, show title; date; time; show description; ticket cost. |
| Add / Edit Seating Layout(s) | Once a show is created the venue manager can begin creating, editing or loading a seating layout to be applied to the show, the seating layout will reflect the venue layout and dictates the maximum seats per customer. |
| Add / Edit Seat(s) | Inside the seating layout, individual seats can then be added which hold the following information, seat price; seat name; seat ID; seat reserve timer; promotion ID; agent ID; seat booking status and seat position. |
| Add/ Edit Discount(s) | Discounts are added when certain criteria is met however can also be applied to the desired show as a whole.  Criteria can include, ticket type or volume of tickets purchased. |
| Add / Edit Promotion(s) | Promotions are controlled at two levels, show and seat. A show promotion is applied to all seats and can be saved, loaded and reused across multiple shows.  A seat promotion can be added at seat level to individual seats for desired reasons, e.g. clearance of sections. |
| View Customers | The venue manager can view customer details by searching from a list of available criteria such as, name; address; phone number or email address. |
| View Purchases | The venue manager can view purchase details either from customer purchase records when viewing a customer or via a provided purchase ID. |
| Edit Ticket(s) Post-Purchase | A ticket can be edited post purchase to reallocate customer seat location in certain circumstances. |
| Send Refund | If required, when viewing purchases the venue manager can issue a refund to the customer without affecting the customers’ ability to use the ticket or cancelling the seat. |
| Cancel Tickets | Should the customer express that they can no longer attend the event, the venue manager can choose to refund and cancel the ticket, reallocating it for purchase again. |
| Add / Edit Agent Account(s) | Agents must be given permissions by the venue manager once they have created their account through the standard email registration system. |
| Logout | The venue manager can log out of his account when desired. |

## First Iteration, Detailed Use Cases

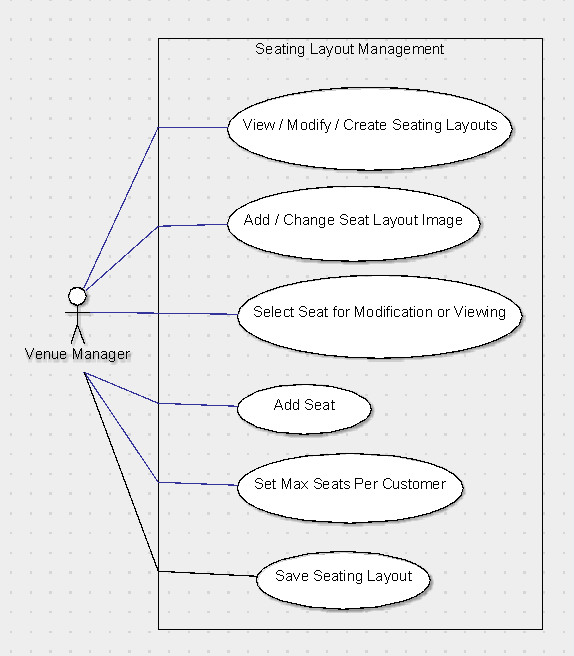
The following use cases are ‘zoomed in’ looks at the basic use cases. They are sorted by the actor that they apply to.

### Venue Manager

The following use cases apply to the venue manager actor only.

#### Seating Layout Management

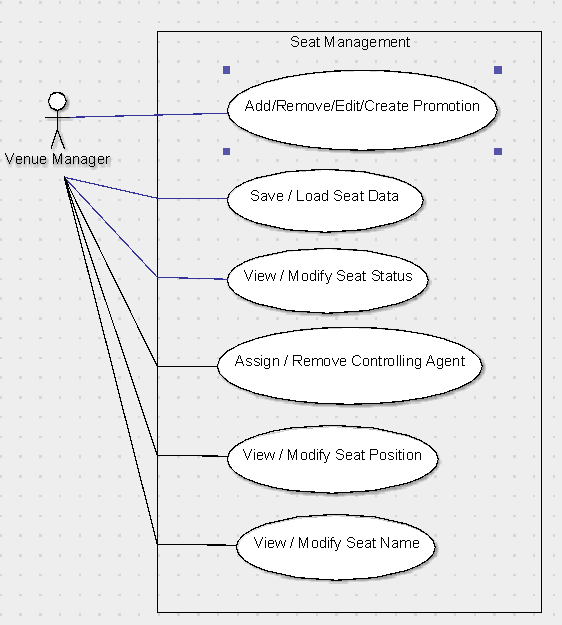
The seating layout management dialog has two parts, one is the select / create / delete dialog box, which is shown prior to the main interface which enables you to edit a seating layout. The select / create / delete dialog shows only a list box with 3 buttons: Create Layout, Modify Layout, and Delete Layout. The selection is done via a list box / drop down box that shows all existing seating layouts, sorted by ID, shown with both ID and name.



|  |  |
| --- | --- |
| Use Cases | Descriptions |
| View / Modify / Create Seating Layouts | This shows a dialog with a list box that contains all saved seating layouts, sorted by ID number, with the text showing the ID number and the layout name. To select a seating layout from here the user simply presses ‘Edit Seating Layout’ with the desired seating layout to edit selected in the list box. There will also be a button in this dialog that allows the user to create a new seating layout, and a button that enables the user to delete the selected seating layout. |
| Add / Change Seat Layout Image | This shows an open file dialog box that lets the user select a seating layout image from a location on the local computer. Once an image is selected the user must click a confirmation button and then the seating layout image will be applied to the currently selected seating layout. |
| Select Seat for Modification or Viewing | In the seating layout seats appear as objects that can be clicked, if the user selects a seat they will be given with options to change and view its properties, such as position, name, promotion ID, or controlling agent ID. |
| Add Seat | When the user clicks the seating layout in an empty space, a new seat will be created, a dialog will show allowing the manager to input the seats details. This is the same dialog that shows when modifying a seat. |
| Set Max Seats per Customer | An integral part of all seating layouts is that there is a maximum amount of seats each customer can purchase. This must be set for each seating layout. This is done via the seating layout interface with a small labelled text box. |
| Save Seating Layout | This option is a simple button in the main seating layout interface that saves the seating layout that is currently open. |

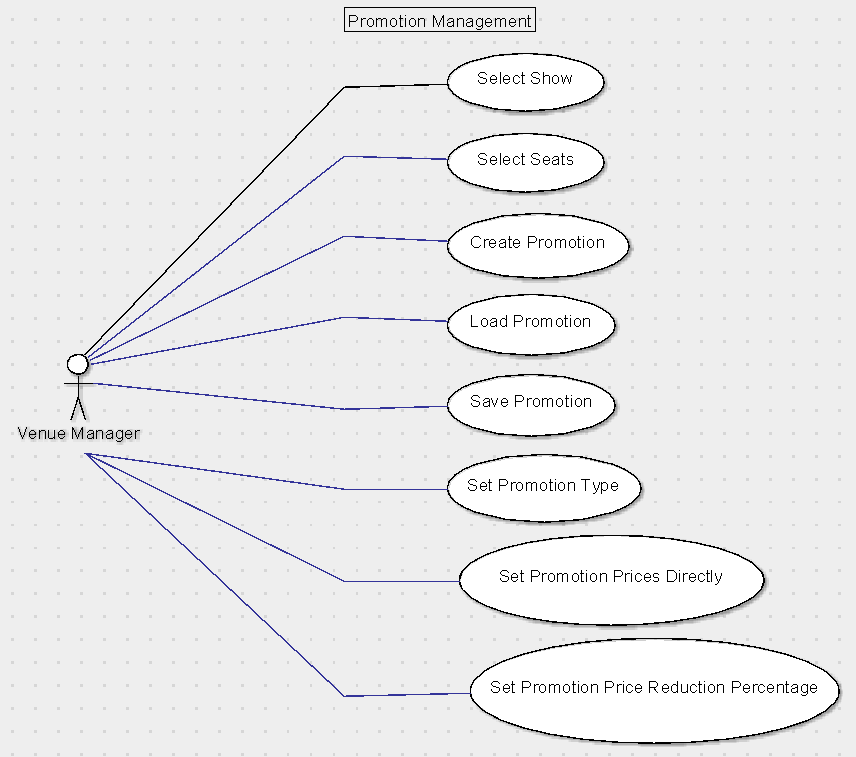
#### Seat Management

Seat management is done via the seat management dialog as well as the seating layout dialog. Within the seating layouts seats can be selected, created, deleted, and moved. Within the seat management dialog seats can be renamed, moved, assigned promotions or controlling agents, have their status changed, and have their data saved or loaded.

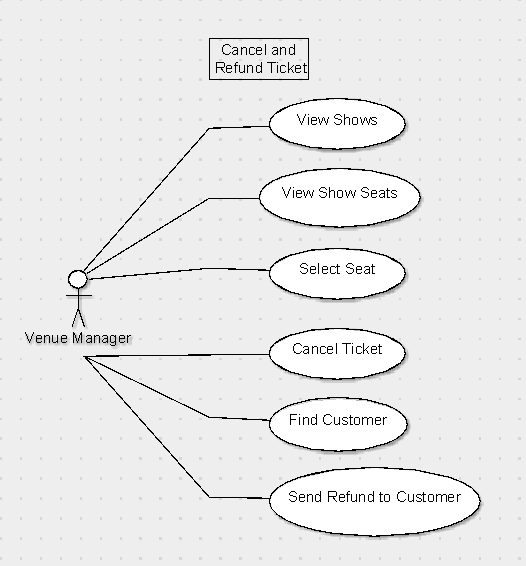


|  |  |
| --- | --- |
| Use Cases | Descriptions |
| Add / Remove / Create Promotion | This option opens the promotion selection dialog, which also includes buttons to add a selected promotion to the seat, create a new promotion, delete an existing promotion, and modify an existing promotion. |
| Save / Load Seat Data | This shows a dialog which enables the user to select from a list of existing saved seat data, and also enables the user to save the current seat data with a new name into the list, or delete a selected option from the list. |
| Assign / Remove Controlling Agent | This lets the user select from a list of agents, in that list is the ‘No Agent’ option as well. This is shown through a list box. |
| View / Modify Seat Status | This lets the user select from a drop down list of available status types, these are ‘Available’, ‘Held’, and ‘Booked’. |
| View / Modify Seat Position | This can be done in one of two ways, either by selecting and then dragging the seat, or multiple seats, across the seating layout to the new position, or by opening the seat management interface where the seat position can be set manually in pixel X and Y position. |
| View / Modify Seat Name | This can only be done through the seat management dialog which is opened by double clicking on a seat. Inside the dialog box there is a text box which shows the current seat name, and there is also an apply button to save the changes, and a cancel button to ignore the changes. |

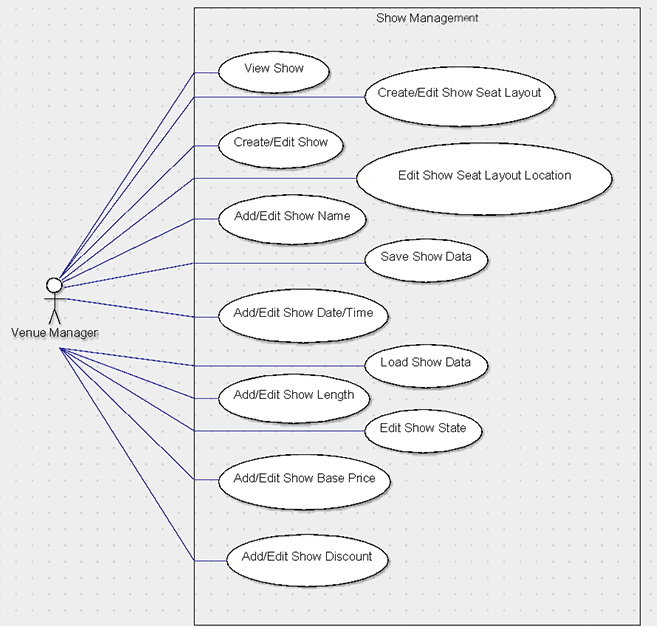
#### Promotion Management



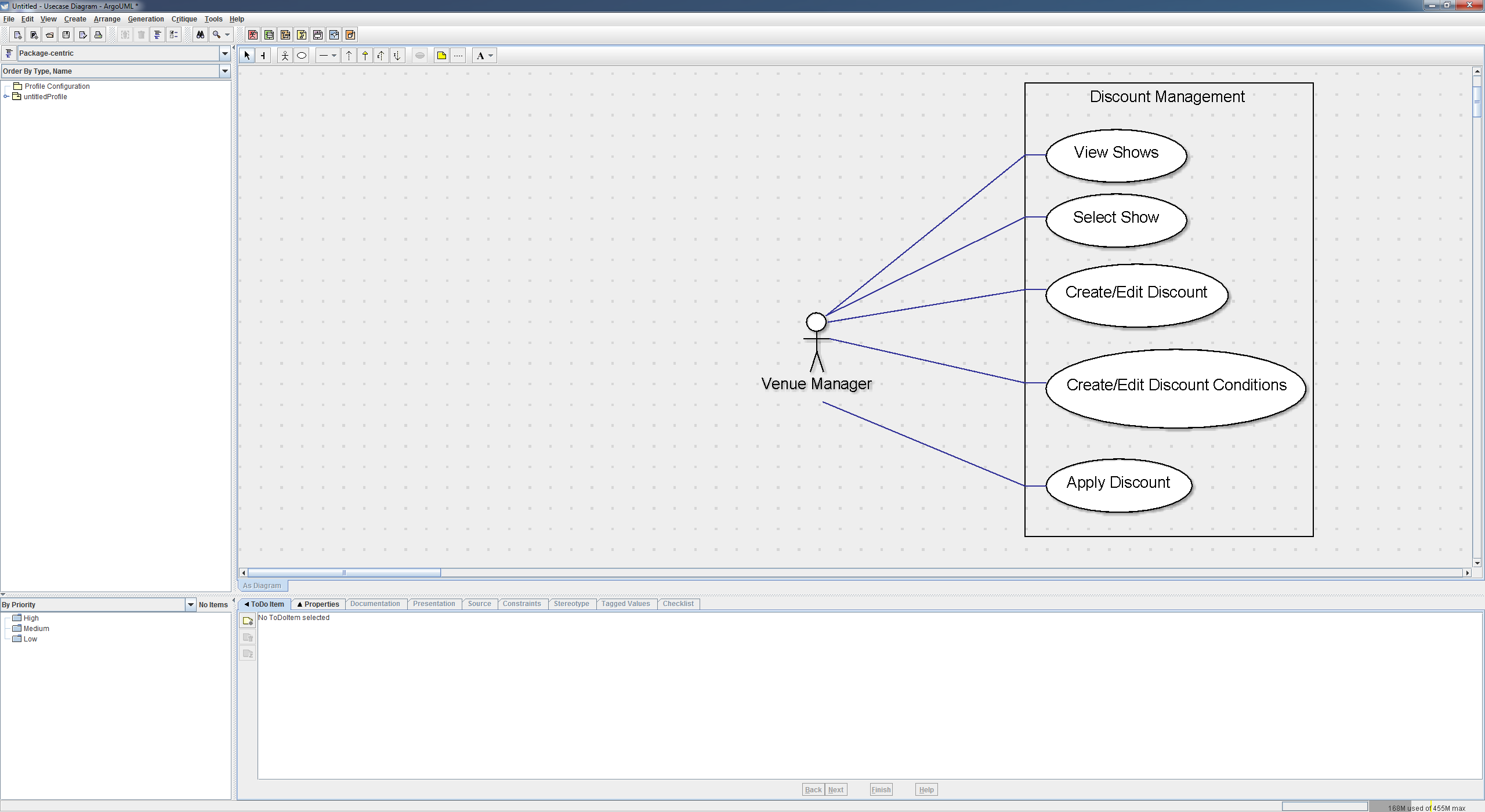
#### Cancel and Refund Ticket



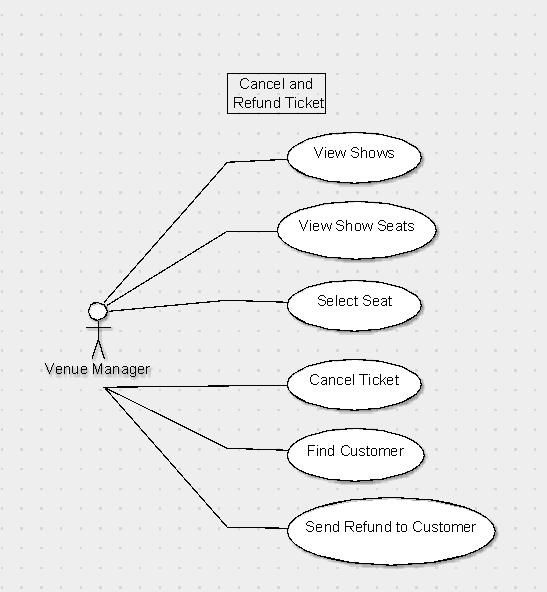
#### Show Management



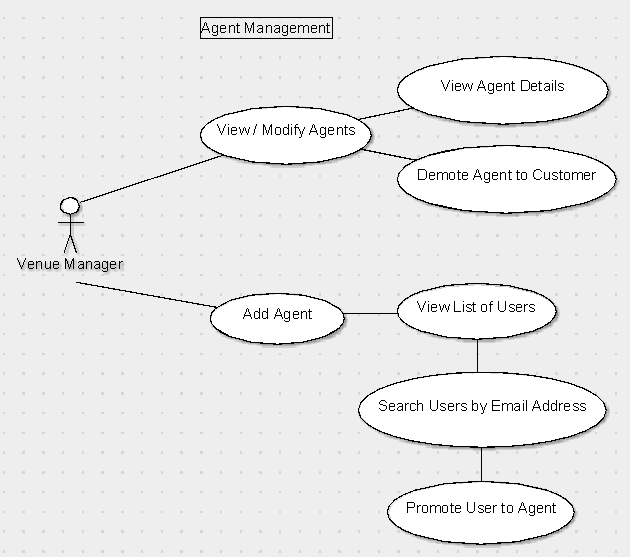
#### Discount Management



#### Cancel and Refund Ticket



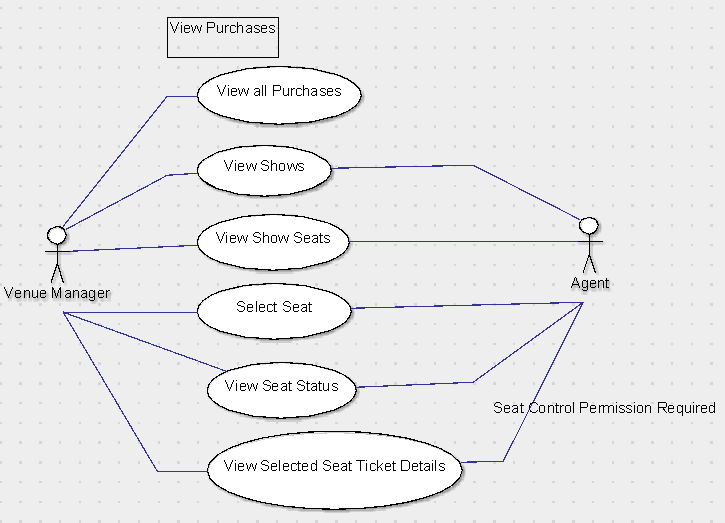
#### Agent Management



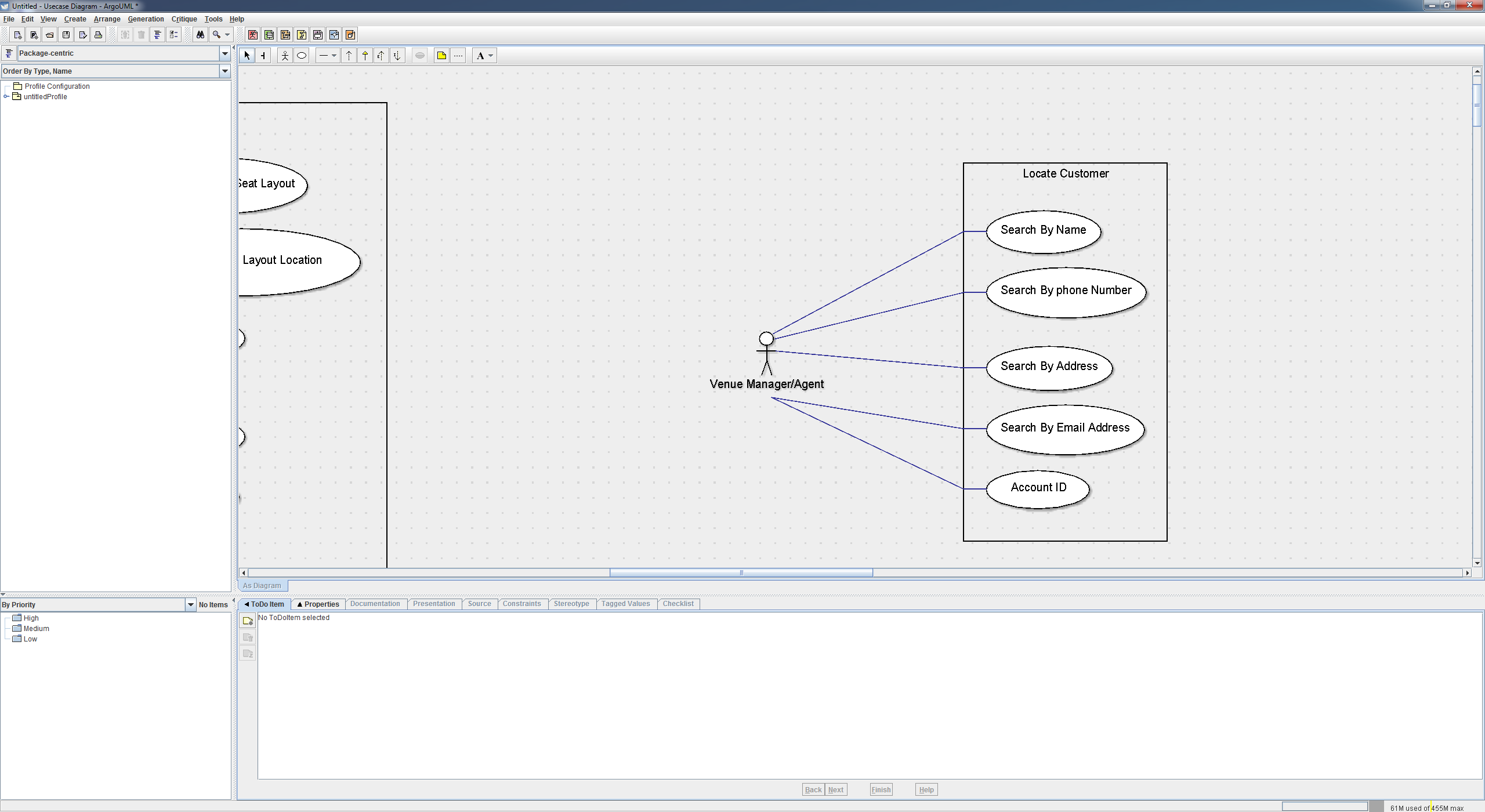
### Venue Manager & Agent Combined

The following are use cases that apply to both the venue manager and the agent actors.

#### View Purchases



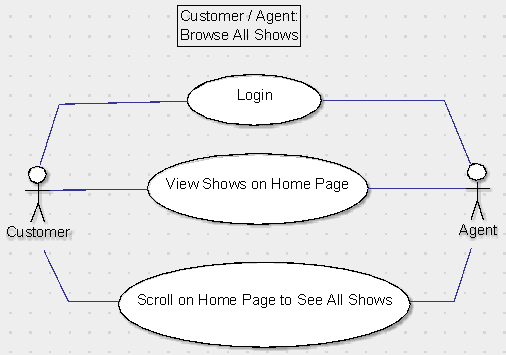
#### Locate Customer



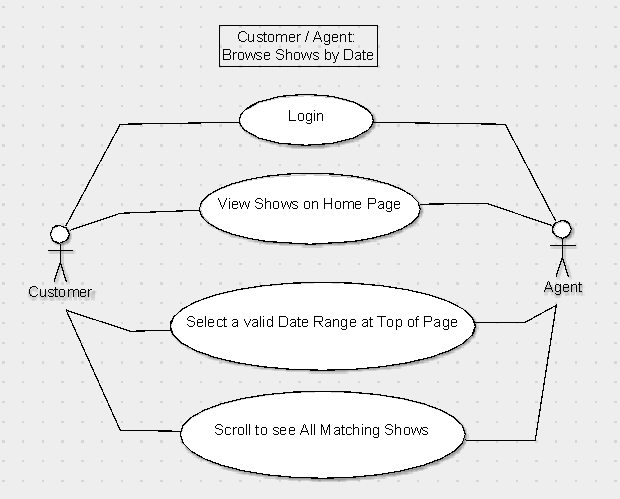
### Customer & Agent Combined

The following are use cases that apply to both the customer and the agent actors.

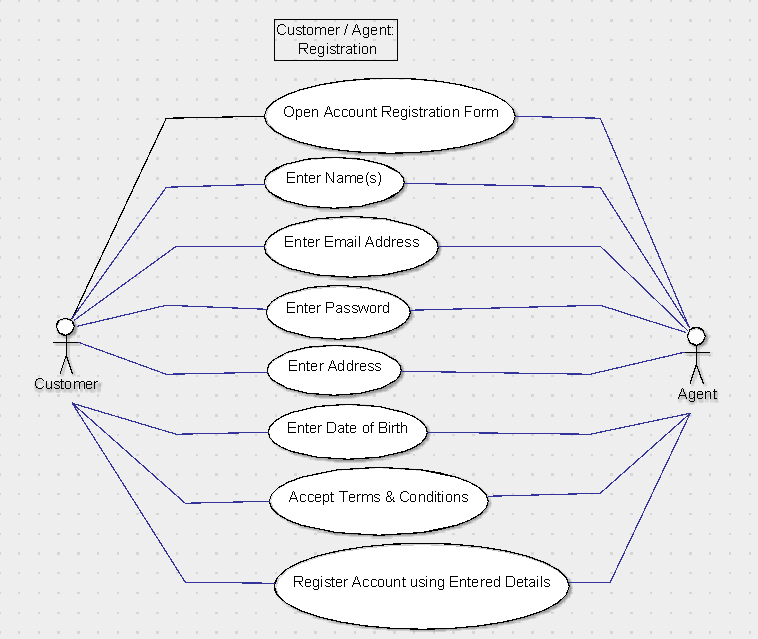
#### Browse All Shows



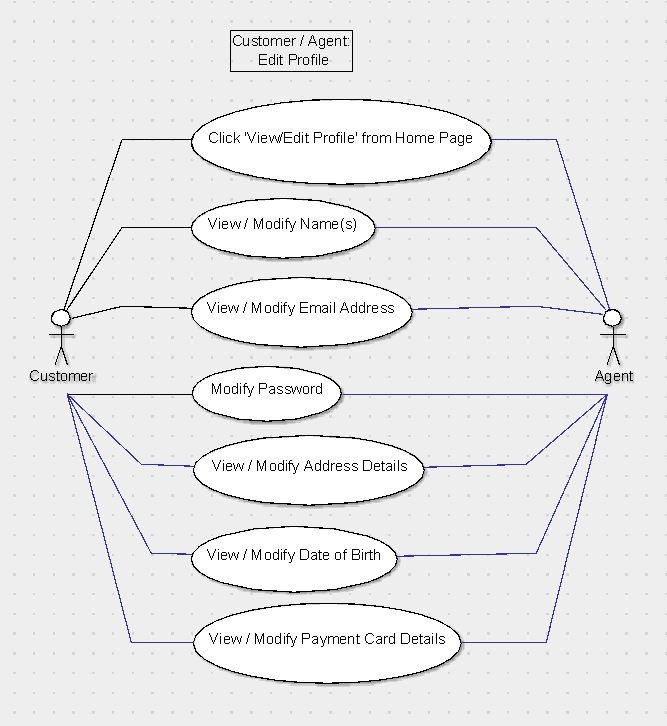
#### Browse Shows by Date



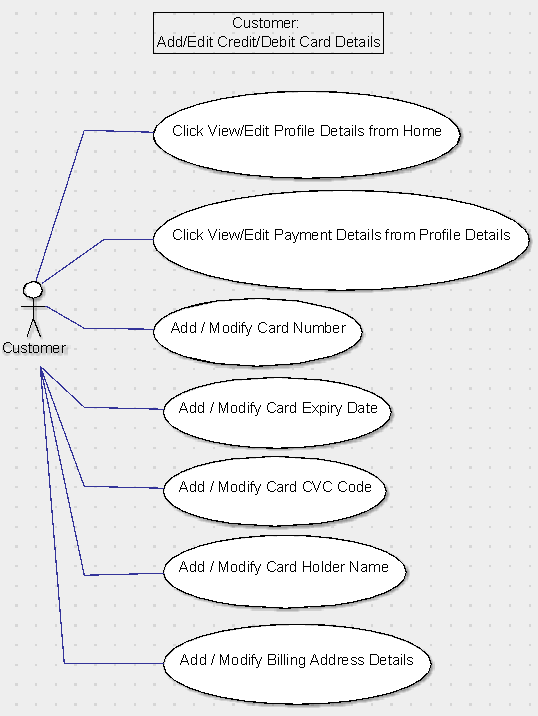
#### Register Account



#### Edit Profile



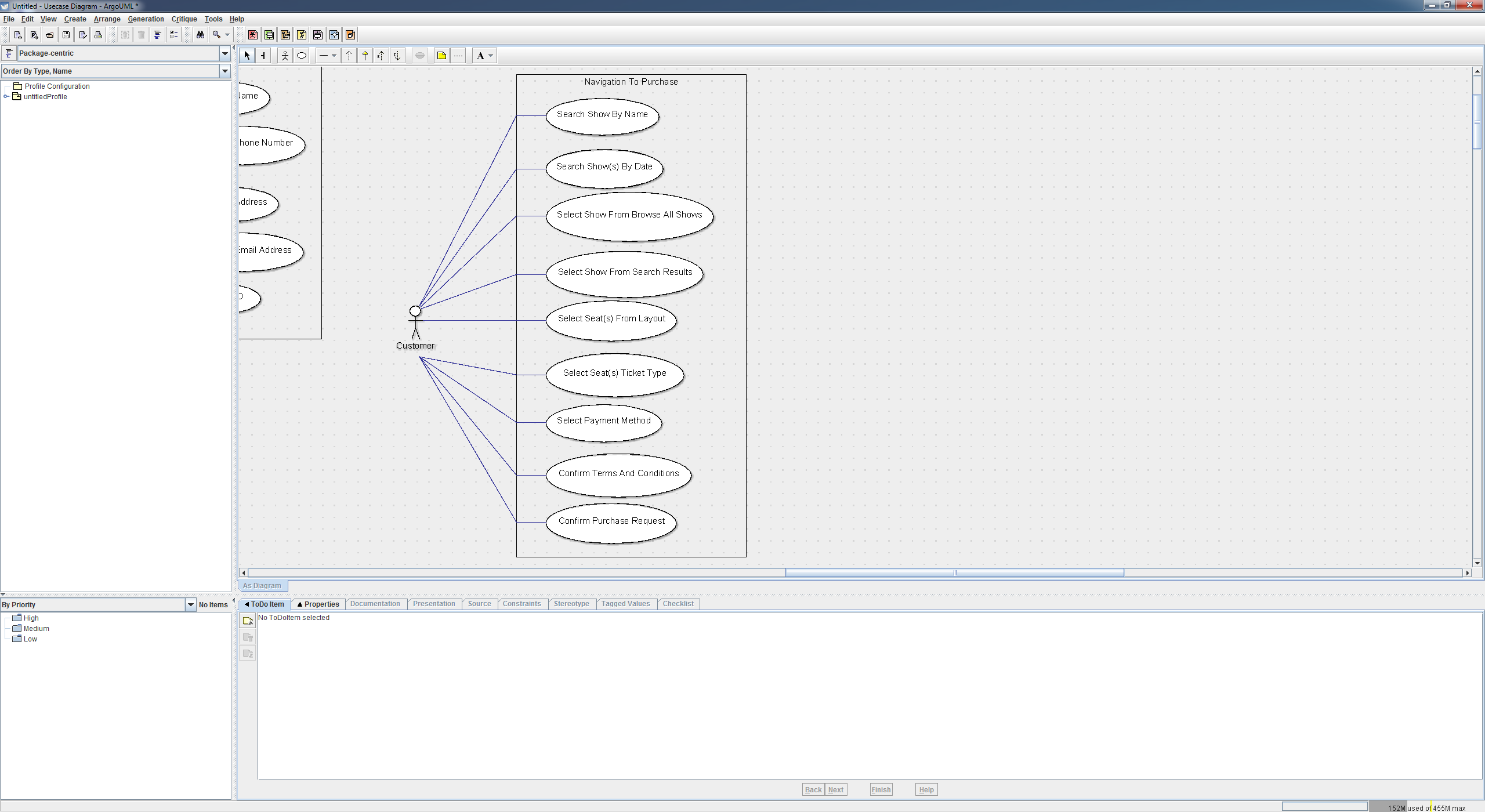
#### Add / Edit Credit/Debit Card Details



### Customer

The following apply to the customer actor only.

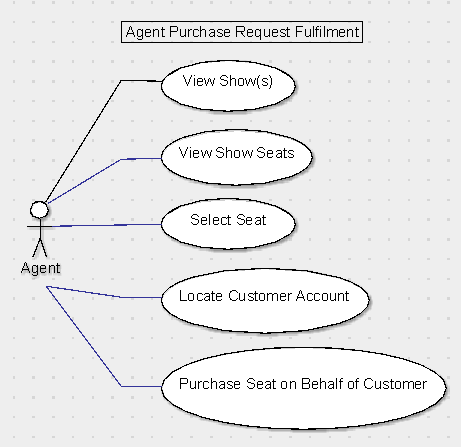
#### Navigate To Purchase



### Agent

The following apply to the agent actor only.

#### Agent Purchase Request Fulfilment



|  |  |
| --- | --- |
| Use Cases | Descriptions |
| View Show(s) | This is the default dialog that opens after an agent logs in, it displays a list of shows in date order and has an option to search for shows by date range or view the upcoming shows. |
| View Show Seats | After a show has been selected the user can open the seating layout dialog by clicking the ‘Open Seating Layout’ button. This opens the seating layout screen which allows the agent to view seats position and status on a map through coloured seating icons, as well as select seats within his control to view their details (ticket prices, discounts, promotions, seat id, and seat name) or purchase them on behalf of a customer. |
| Select Seat | A seat can be selected with other seats, each selection is done by one click. This use case describes the result of that click. When a user selects a seat it adds the seat to the selected list. There is a text display to the right of the layout with all of the selected seats information regarding status, price, name, and ID number. Held or booked seats cannot be selected, neither can seats the agent does not control, clicking these seats does nothing. It will not add them the selected seats list. After seats have been selected that are valid for purchase the ‘Purchase Seat(s)’ button is activated. |
| Purchase Seat on Behalf of Customer | After an agent has selected valid seat or seats, they can click the purchase button This opens a new dialog showing the current purchase details, such as the seats selected, the ticket types, the cost after discounts and promotions have been applied with details of such discounts and promotions, the show details, and the customer account it is being purchased through. To complete the purchase they need to go through the ‘Locate Customer Account’ use case. They will also need to within the purchase interface, select the ticket type for each seat from Adult, Student, or Child. |
| Locate Customer Account | This is an interface that has a simple search bar, with a list box to select the type of search to perform, from postcode, name, phone number, and email address. Once a single account is selected, an OK button is available, which confirms the selection. |

## Second Iteration, User Interface Prototypes

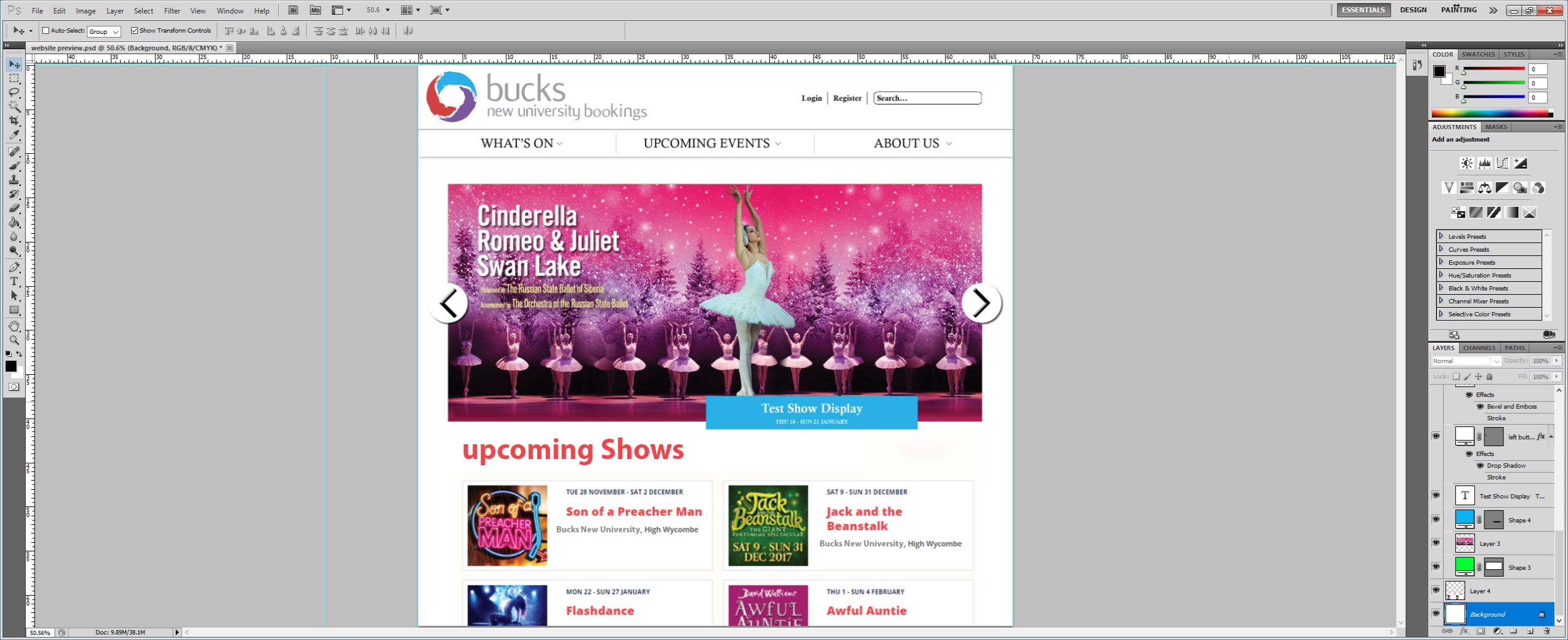
### Introduction

To help visualise and to test out ideas related to the program’s structure a prototype was built with the help of Visual Studio’s C# and the .NET framework. Since the target platform for the software is Windows we can use Windows Forms components within .NET and still be fairly certain that at minimum visual parity can be achieved between the prototype and final software. In most cases the prototype is ‘ugly’ in a sense that it has been stripped of any extraneous stylization elements. Since animations or video cannot be submitted on paper without resorting to a flip-book style, I will be annotating screenshots. The small red circles with numbers in correspond to a description in the table below the screenshot.

### Annotated Screenshots

#### Show Listing / Home

##### Screenshot



##### Screenshot Description Table

|  |  |
| --- | --- |
| No. | Annotation Description |
| 1 | The accounts bar will exist across the whole navigation section of the application and once successfully logged in will change “Login | Register” to “Account” while maintaining the search function from every page during navigation.  When log in is clicked it will load the below login screen which will also display another registration option. |
| 2 | The navigation bar allows the user to navigate shows in certain ways.  “WHAT’S ON” will present a list of current shows that are currently or soon to be showing.  “UPCOMING EVENTS” will show a full list of currently scheduled shows in date order, and also allow the user to search from the selection by either show name or show date, this can also be done in the “WHAT’S ON” tab.  “ABOUT US” is a page dedicated to the BCPA team and University to give a synopsis of the team and any relevant information they would like to share with the public. |
| 3 | This is a scrolling screen that will display current popular or prime events and draw attention to desired events. |
| 4 | This is a less detailed version of the “WHAT’S ON” tab which will just display a small amount of upcoming shows in date order, the user can’t search from this section. |

#### Show Details

##### Screenshot



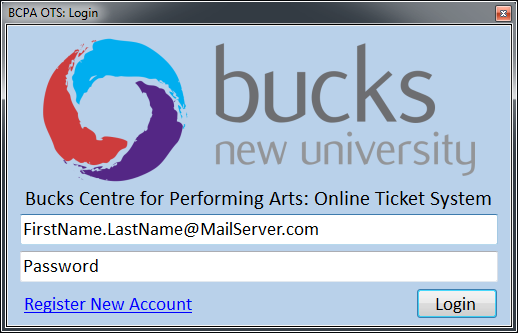
View available dates

##### Screenshot Description Table

|  |  |
| --- | --- |
| No. | Annotation Description |
| 1 | This button will allow the user to proceed with booking tickets for the current selected show. |
| 2 | This will display the selected show date and allow the user to view alternatively available dates. |
| 3 | This section is customised to show relevant, promotional or additional shows that the venue is actively promoting. |
| 4 | This section is a description of the selected show outlining show synopsis, cast and other key information. |

#### Login Interface

##### Screenshot

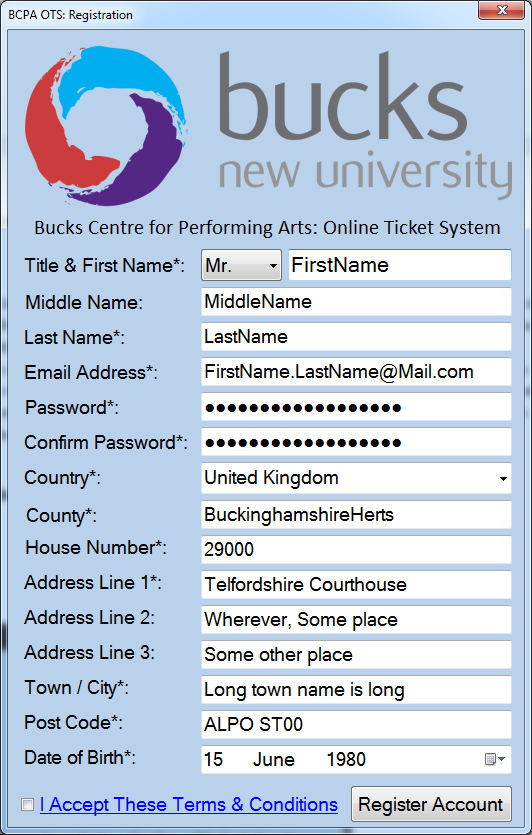


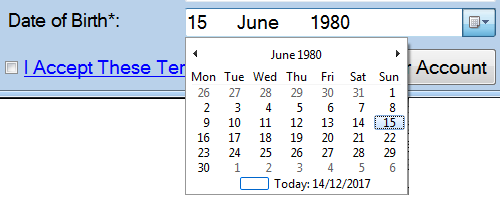
##### Screenshot Description Table

|  |  |
| --- | --- |
| No. | Annotation Description |
| 1 | This interface is loaded when a user clicks ‘Login’ from the Show Listing forms. It always asks for an email address and password to login. The Bucks logo and the ‘Bucks Centre for Performing…’ text is a placeholder only. |
| 2 | This email address acts as a username to login, the placeholder text in the field disappears on user focus. |
| 3 | This password field hides characters typed into it, and has placeholder text that disappears on user focus. |
| 4 | The register new account hyperlink opens the account registration form while leaving the login form open. |
| 5 | The login button checks the inputs for invalid characters, then sends the login details to the server. If there are any invalid characters a pop-up message box will show that tells the user where the error is. While it waits for a response the ‘Login’ button is disabled (greyed out) and the text changes to ‘Waiting’. If the login request is a success then the ‘Login’ button text changes to ‘Loading’ while the main interface loads, then after the main interface has loaded the login form closes. If the request to login is not successful then an error message is displayed, this could either be a connection error if no response was received in time, or a login invalid error if the request was received but incorrect login details were given. |

#### Registration Form Interface

##### Screenshot



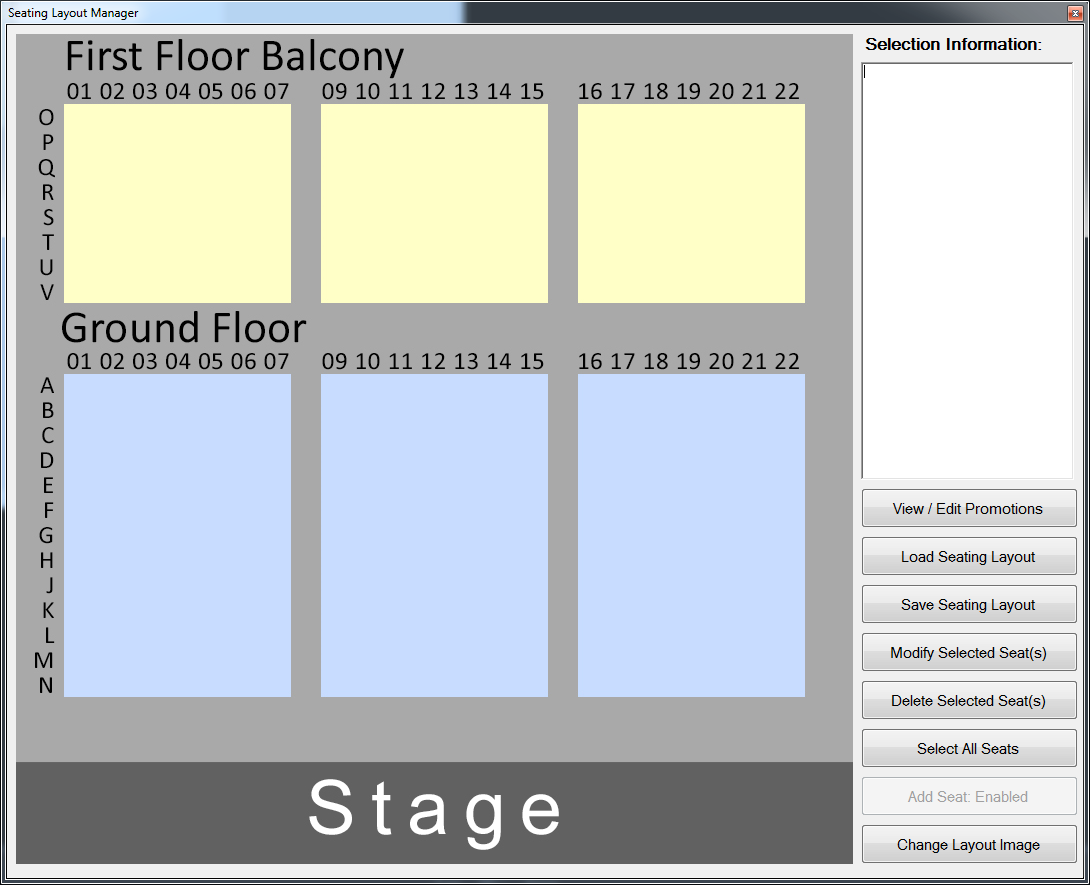


##### Screenshot Description Table

|  |  |
| --- | --- |
| No. | Annotation Description |
| 1 | This interface is the registration form, accessed by clicking ‘Register New Account’ in the login form or ‘Register’ in the Show Listing forms. It can be closed without any consequence by clicking the ‘X’ in the top right. This will bring you back to the login form which remained open in the background. In this screenshot all of the fields are filled in with dummy data. |
| 2 | The title box is a standard list box that shows a list of all titles when clicked, and allows the user to select one. The user must select a title to register an account. |
| 3 | As can be seen some fields are marked with an asterisk (\*), this is a universally known indicator that shows a field marked with it mandatory, that is that it must be completed to successfully register an account. |
| 4 | The ‘I Accept These Terms & Conditions’ link opens a popup box with a scrollable text that shows all of the terms and conditions. The registration form stays open in the background and regains focus when the user closes the terms and conditions popup. The small white box next to the link is a checkbox which must be checked to register. |
| 5 | The register account button can be clicked at any stage, however if any details have characters that are not allowed, or any mandatory fields are not completely filled in, then it will show the user an error message and ask them to either fill in the parts they missed, or remove special characters from certain fields. If the registration is successful it will display a success message then close the form, revealing the login form behind it. |
| 6 | This image just shows how the country list box looks when opened, with a scroll bar and a list of all the countries to select from. The user must select a country in the list and cannot type their own. |
| 7 | The date of birth control allows the user to both type their date of birth directly into the box (it does not accept invalid inputs), or to click the calendar button on the right hand side of the field, which opens a calendar menu as is shown. This allows the user to visually select the date rather than typing it in. |

#### Seating Layout Interface: Manager View, Blank Layout

##### Screenshot



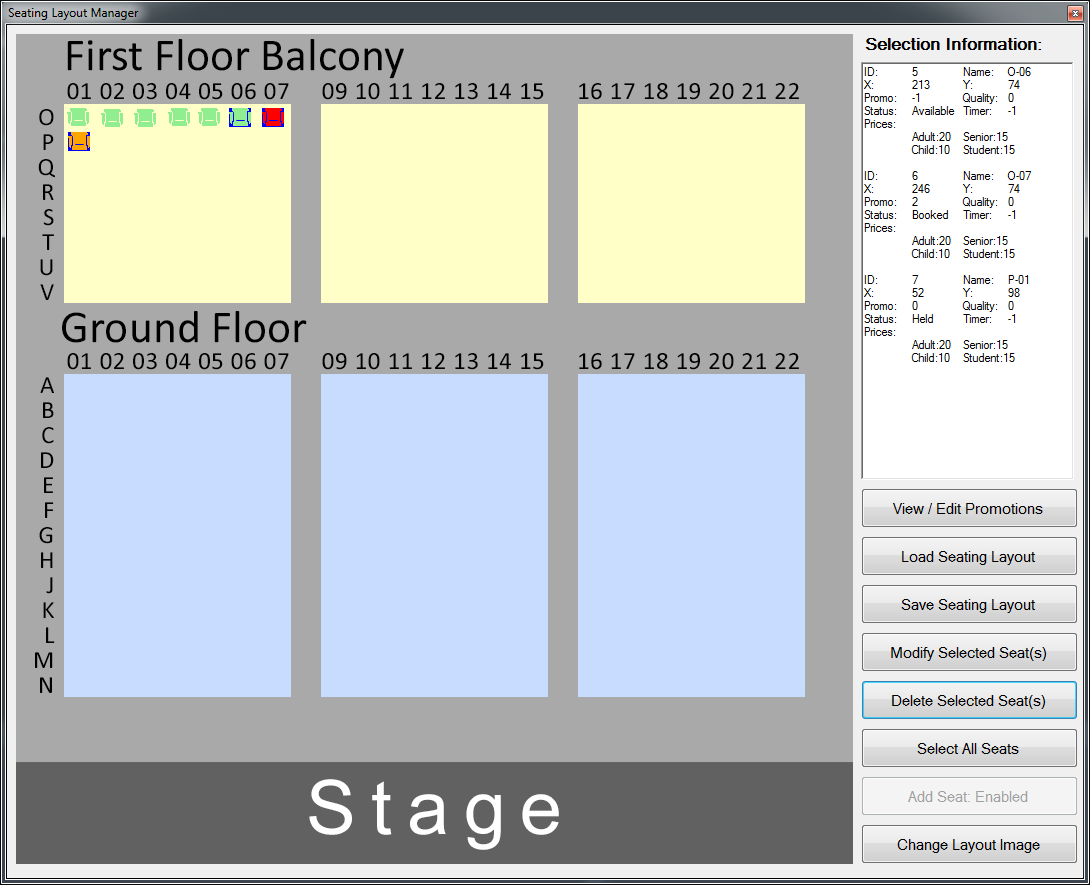
*The screenshot description table for this screenshot is on the next page…*

##### Screenshot Description Table

|  |  |
| --- | --- |
| No. | Annotation Description |
| 1 | This window is a pop-out from the show page itself. When clicking the ‘Seating Layout’ button from the show page, this window opens and automatically loads the clicked shows seating layout. The name and buttons in this window vary depending on user (Agent, Manager, Customer). If the user is a venue manager, the title is ‘Seating Layout Manager’ as seen in the above screenshot, since the above screenshot is from the Venue Managers point of view. |
| 2 | This section is the main seating layout area. In the shown screenshot no seats have been added to the layout, it is a blank canvas. The image in the background can be set by the Venue Manager for each individual seating layout via the ‘Change Layout Image’ button in the bottom right of the screen. The image size determines the size of the window, if the venue manager changes the layout image they must consider the size they want it to appear on the clients screen. This gives the venue manager an extra layer of control over how the seating layout appears to the client. The only restriction is that the image must be greater than 500 pixels in height so that there is room for the buttons and text box to the right hand side of the screen. The program will not allow images with a height less than this to be applied to the seating layout. |
| 3 | The selection information box is a rich text box, which means it allows for formatting options such as coloured, highlighted, bold, italic, etc. text. When a seat is selected it displays information regarding that seat. This box automatically resizes itself to fit the height of the window minus the space needed for the buttons. It also automatically adds horizontal or vertical scrollbars to ensure that no text can be cut off even when several selections are made. |
| 4 | These buttons vary based on the user, in this screenshot the user is a Venue Manager, and as such they have multiple buttons that allow control over the seating layout.  The first button is ‘View / Edit Promotions’. This loads the general promotion management interface. Promotions are applied to individual seats, and not layouts as a whole, so the interface this button loads contains all promotions, not just those that have been applied to seats within this layout.  The second button, ‘Load Seating Layout’, allows the user to select from a list of saved seating layouts, and upon selection of one, loads that layout into the interface.  The ‘Save Layout’ button asks the user for a layout name, then saves the layout and all of its contents.  The ‘Modify Selected Seat(s)’ button opens the seat management interface for each selected seat. It does this by opening the first selections seat management interface, then upon the closing of that interface, the next selections seat management interface is loaded, this repeats until all of the selected seats have had their management interface opened, potentially modified, and then closed.  The ‘Delete Selected Seat(s)’ button does as it describes, it deletes any selected seats from the layout entirely with no residual information or objects left behind.  The ‘Select All Seats’ button is also self-explanatory, it selects all the seats within the layout.  The ‘Add Seat: Enabled’ button, which is greyed out / disabled, is there to show the current mode. In the future this button will be enabled and act as a toggle between adding a single seat and an entire row, this would ease the user experience when creating or modifying seating layouts.  The ‘Change Layout Image’ button lets the user select a new image for the layout background. This image determines the size of the Seating Layout window itself and has a minimum height of 500 pixels to ensure there is room for the controls and text box on the right hand side of the window. |

#### Seating Layout Interface: Manager View, Including Seats in Layout

##### Screenshot

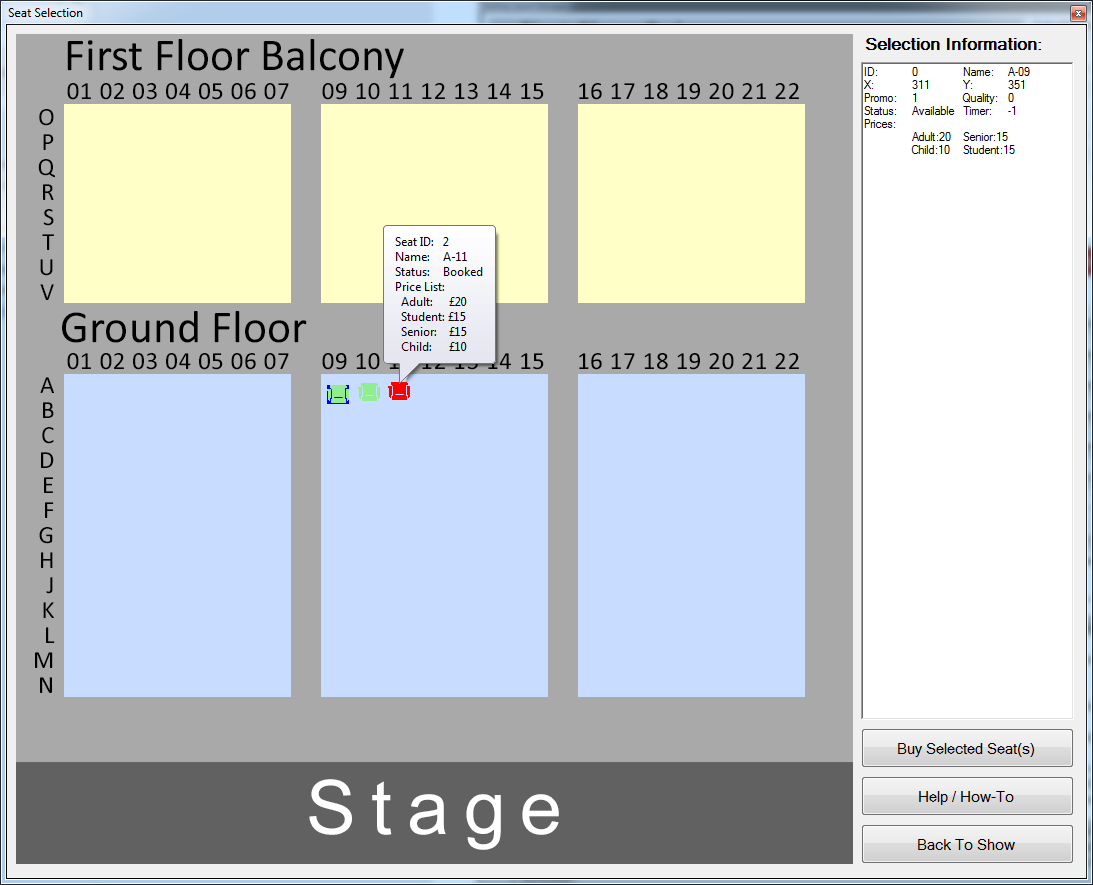


##### Screenshot Description Table

|  |  |
| --- | --- |
| No. | Annotation Description |
| 1 | In this area you can see that there are several seats placed into the layout, three of which are selected. |
| 2 | Within this area the manager can perform various actions depending upon varying conditions.  If the user has not got any seats selected, and they left click a blank space within the layout, a new seat is created and the Seat Management interface for that seat is shown. If the user has not got any seats selected, and they left click a seat, that seat will become selected.  If the user has got seats selected, and they left click a blank space within the seating layout, then their selection will be cleared and there will no longer be any selected seats. If the user has not got any seats selected, or if they do have seats selected, and they right click an existing seat then the seat management interface for the seat they right clicked will be shown.  If the user has a seat or multiple seats selected, and they left click one of those seats, then it puts the seat or seats into move mode. In this mode the seat they clicked on follows the cursor, and all of the other seats (if any) selected move relative to the seat they clicked movement. If the user attempts to move any of the seats out of bounds all seats will be deselected and stop in place. |
| 3 | This area shows information about the currently selected seats. As can be seen each seat has an individual name and ID. Two of them have a promo ID that isn’t -1 (-1 is no promotion). The green seat is available, the red seat is booked, and the orange seat is held (pending purchase). This textbox will automatically expand and include a scrollbar. |

#### Seating Layout Interface: Customer and Agent View, Including Seats in Layout

##### Screenshot

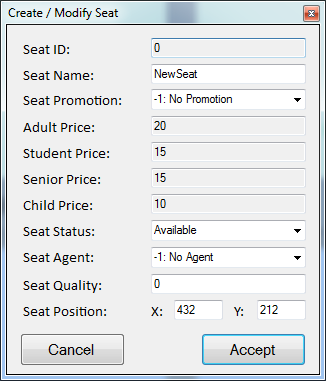
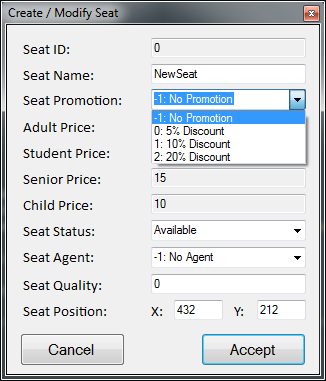
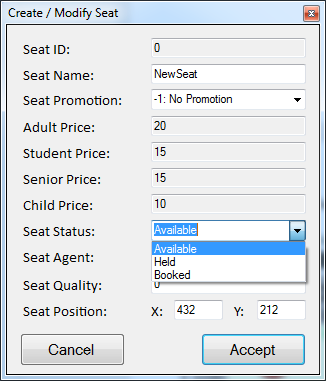
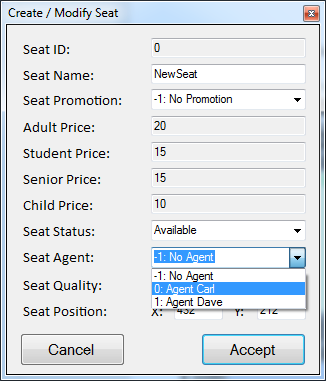


##### Screenshot Description Table

|  |  |
| --- | --- |
| No. | Annotation Description |
| 1 | Here you can see that the user has selected a seat, shown by the blue outline around it. That seats information is also shown in the ‘Selection Information’ box to the right of the screen. |
| 2 | In this screenshot the user has hovered their mouse over the booked seat A-11. There is a small amount of information is shown about it in a small pop-out window which automatically closes after a second if the user moves their mouse more than double the size of the cursor away from it in pixels. |
| 3 | Here information can be see about the seat. This is a prototype and does not reflect the level of detail this will show to the customer or agent. The textbox can use ‘rich text’ formatting, which means that it allows for different font sizes, colours, styles, and highlighting within the box to make the information stand out in the most readable way for the customer. The same can be done to make the agents job easier as well. |
| 4 | The topmost button allows the user to buy the seats they have selected, this is only active when the user has selected valid seats. In an agents case this refers to seats they control, for a customer this refers to seats that are available. The help / how-to button should open a page with instructions that vary depending on whether or not the user is a customer or an agent. The back to show button takes the user back to the show listing that the seating layout refers to. |

#### Seat Management Interface

##### Screenshot



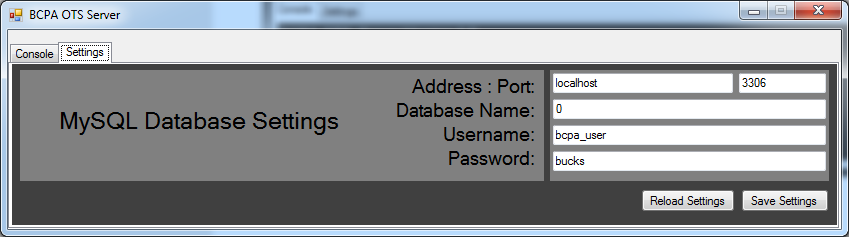
##### Screenshot Description Table

|  |  |
| --- | --- |
| No. | Annotation Description |
| 1 | This dialog lets you edit properties of a seat and view non-modifiable properties. All non-modifiable properties are in grey text boxes. Seeing the price as non-modifiable may seem confusing, however, seat prices are not set directly through the seat individually. Only promotions are individual to the seats themselves. The pricing structure is determined by the seating layout and this denotes the base cost prior to show-based discounts and seat-based promotions being applied. When this seat management interface is opened it re-calculates based on any seat-based promotions or show-based discounts. |
| 2 | This shows the promotion list box selection opened, it shows the ID of the promotion then the name of the promotion. Selecting a promotion automatically updates the prices shown for each ticket type. Each seat can only have one promotion applied to it. |
| 3 | This sets the seat availability, which updates the seat colour in the seating layout too. Available is green and means anyone can book the seat, held means a user is in the process of purchasing this seat, in which case a timer will be counting down. Red means that a seat is booked, the purchase has been confirmed and paid for. |
| 4 | This shows the agent selection list box, which allows the user to select from a list of existing agents and assign an agent to a seat. Each seat can only have one agent applied to it. |

#### Server Console Interface

##### Screenshot





##### Screenshot Description Table

|  |  |
| --- | --- |
| No. | Annotation Description |
| 1 | This is the server console interface, this runs on the server machine. It has two tabs, the console tab shows status messages, such as connection received, request received, etc. The settings tab allows the user to set-up the connection the server needs to a MySQL database, where it stores data. |
| 2 | This is where status messages are displayed, for example when a user connects to the server a message will be displayed showing the IP address of that user and their unique connection ID number, if they are logged in it may also show their username (email address). |
| 3 | This is the MySQL database settings section, the server requires a MySQL database connection so that it can store and retrieve data. |
| 4 | These buttons allow the user to refresh and save settings that they have entered. Pressing reload just loads the stored settings back into the fields, which is useful if for example you accidentally clear a field, and cannot remember what data was in there when the settings were last saved. |

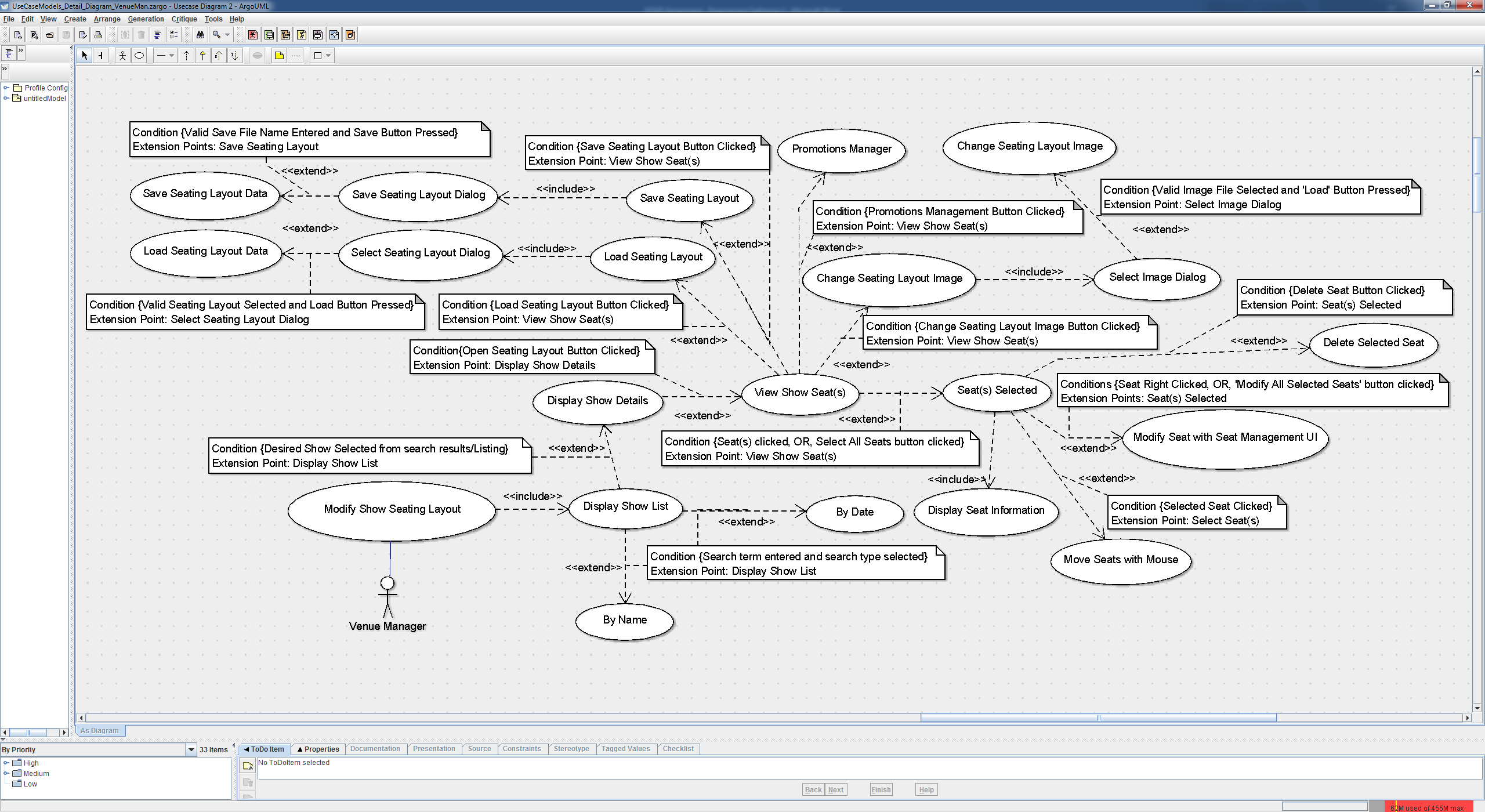
## Second Iteration, Use Cases with Extensions & Inclusions

The following are detailed use case diagrams that show the relationships between parts through extensions and inclusions. This will help with deciding upon inheritance and re-use of classes.

### Modify Seating Layout Use Case Description & Diagram

In the diagram it can be seen that the first step towards modifying a seating layout is opening the show listing, from there a show can be navigated to with the search methods or without, and a shows details can be displayed. After a shows details are displayed the seating layout can be displayed for that show. Without any seats selected there are several options, these are: Select a seat (or seats), Save the current or load an existing seating layout, open the promotions management interface, and change the seating layout image. If a seat or seats are selected there are yet more options, these are to delete, modify, or move the selected seats.

*See next page for diagram…*



Modify Seating Layout Use Case Diagram

### Purchase Seat on Behalf of Customer Use Case Description & Diagram

The Agent will select the “Purchase Seat on Behalf of Customer” button which will take him to a new window to view shows by Date; name or by default, display all shows in date order.

This will display a list of shows that the Agent can then select from to display show details.

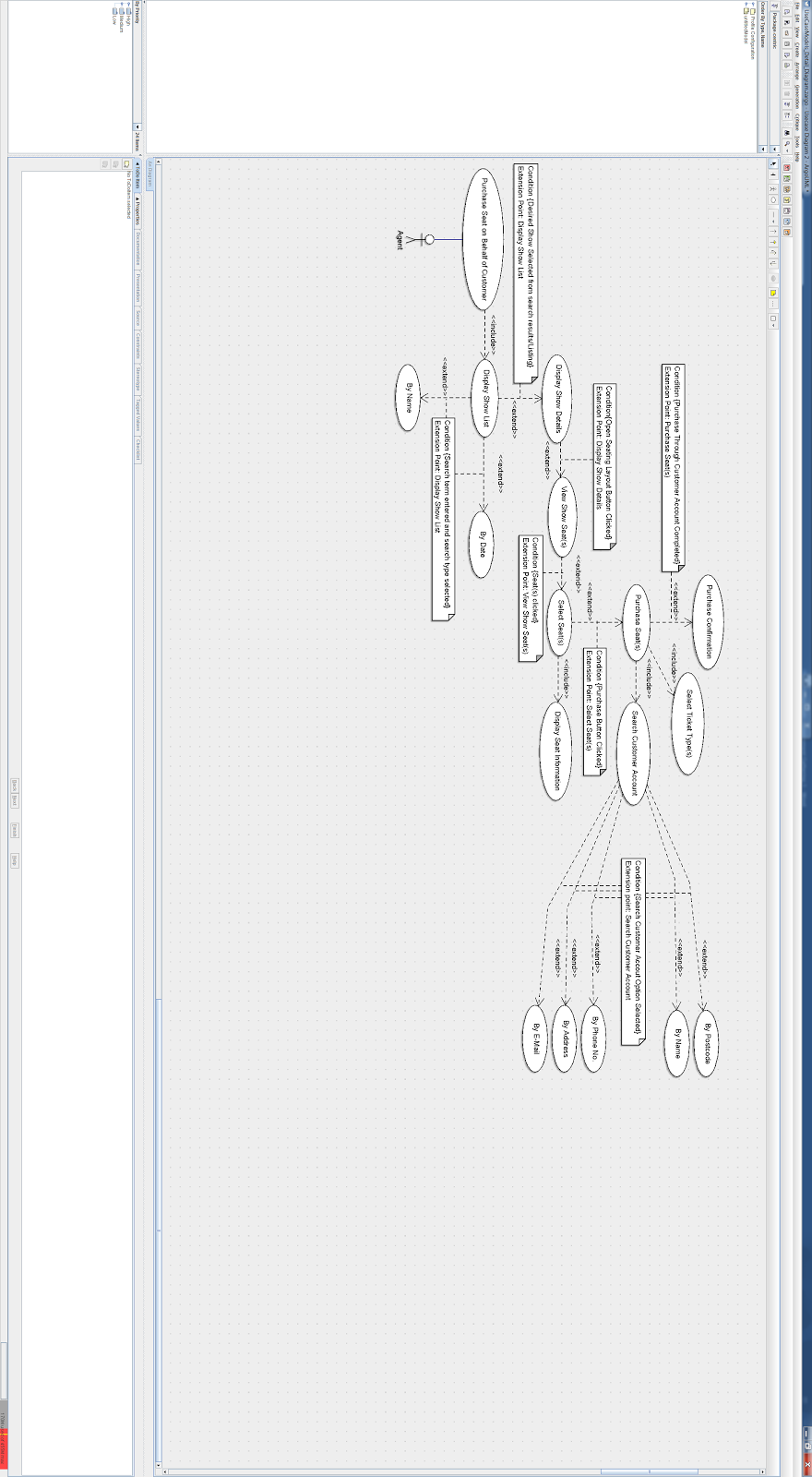
Once the desired show is selected the Agent may then select to view the seating layout where he can select the customer’s desired seats, up to a predetermined maximum (set by venue manager upon show creation) which will then display a box by the selected seats showing the price tiers of the selected seats; seat status and assigned agent.

Once the Agent has finalised the seats he must then select the purchase seat(s) button which will request him to assign the ticket types (I.E. Student, Senior, Child).

Once all seat types are assigned the Agent must search for the customer by; Postcode; Name; Phone Number; Address or Email Address.

Once the account is selected, the Agent can continue with entering Payment information to finalize the purchase, which will display purchase information such as order number, and automatically email the customer a digital receipt and order confirmation.

*See next page for diagram…*

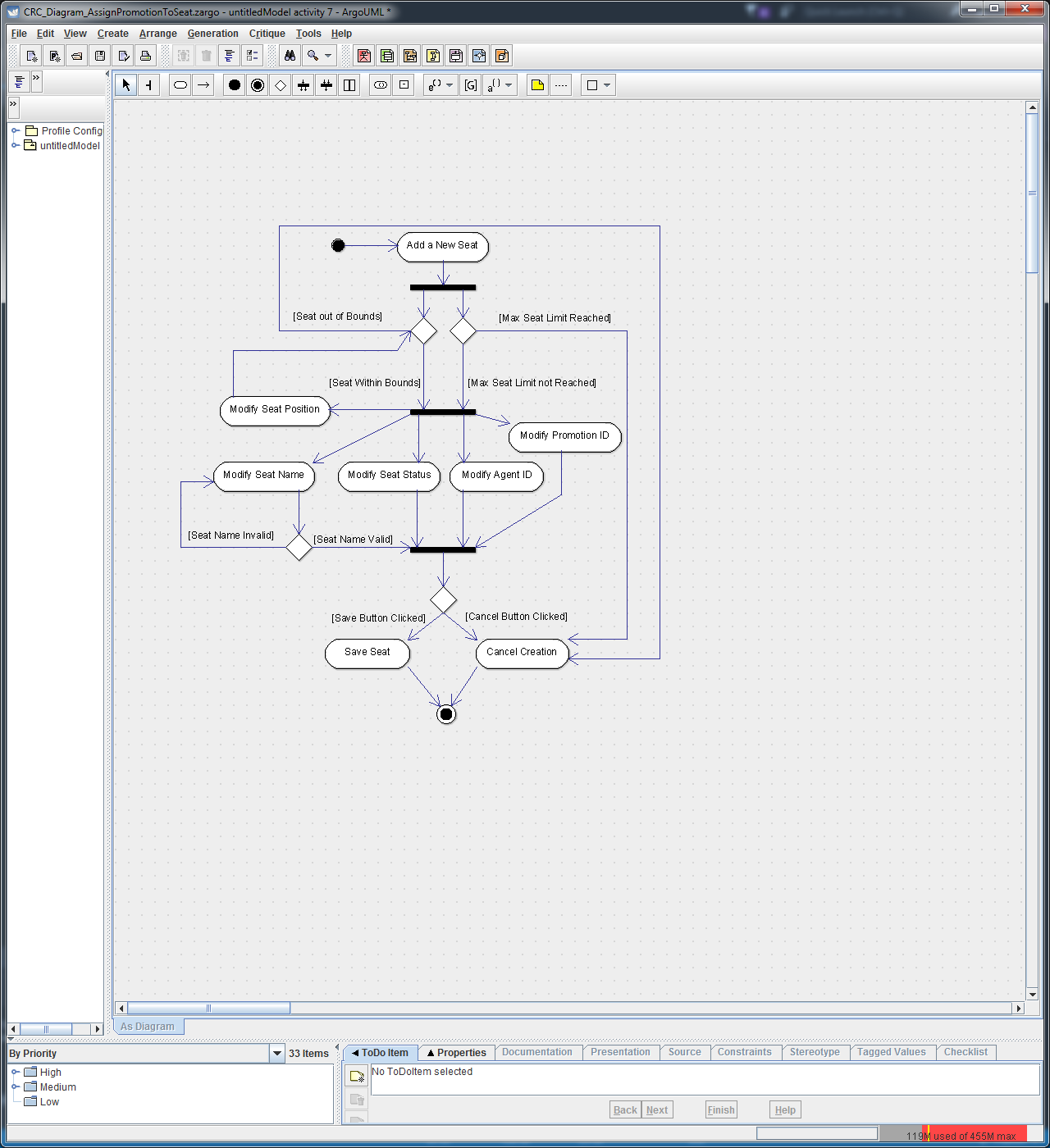


**Purchase Seat on Behalf of Customer** **Use Case Diagram**

## Activity Diagrams

### Add New Seat Activity Diagram & Description

#### Diagram

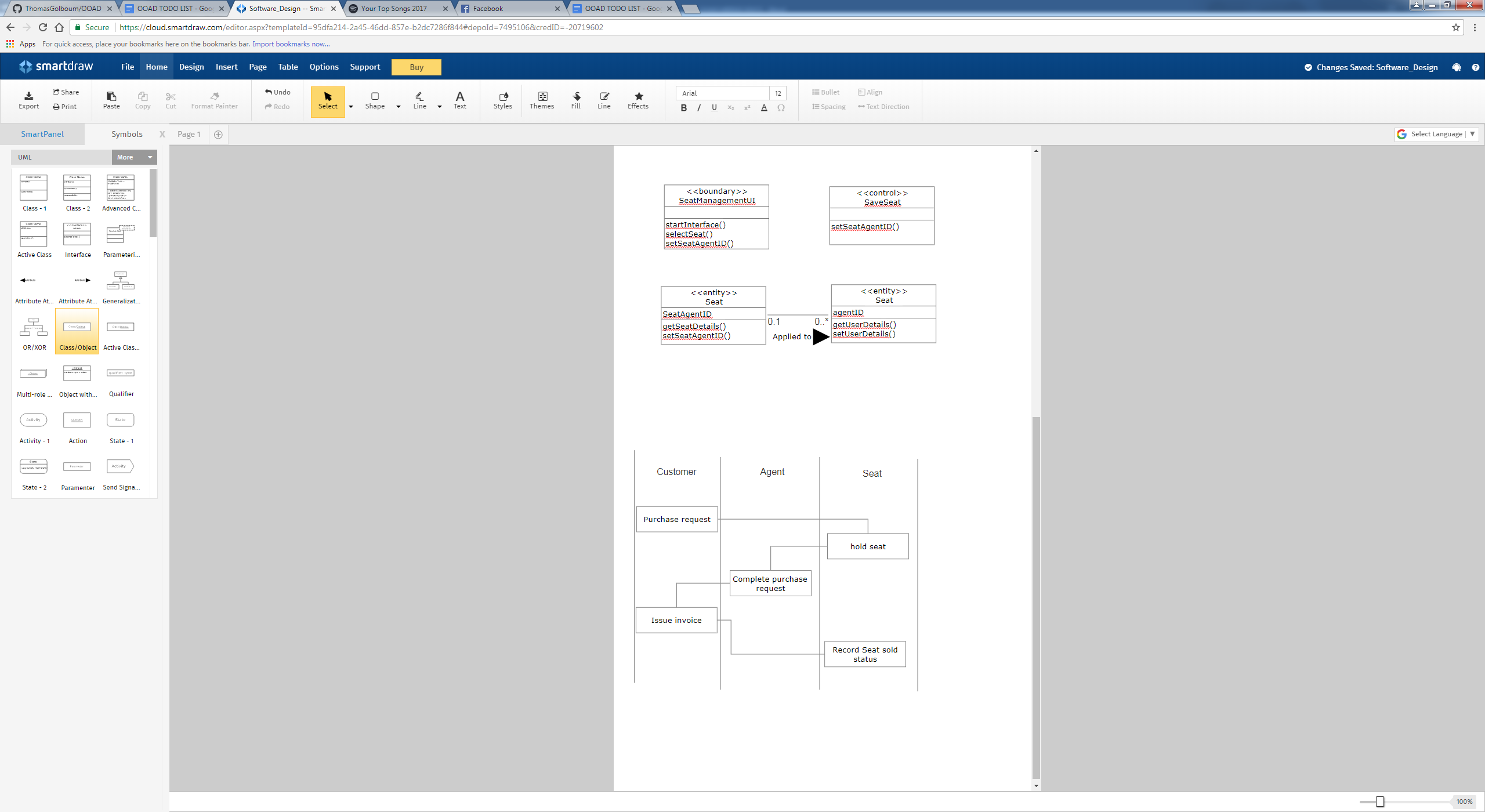


#### Diagram Description

The above shows the activity diagram for adding a new seat to an existing seating layout. You may notice that there are no decision nodes for invalid or unavailable agents or promotions, this is because the default option selected is ‘No Agent’ and ‘No Promotion’ which both have ID numbers of -1. The user interface automatically populates a list that the user can choose from and therefore the user cannot enter invalid data for this step, unlike the name step and the out of bounds step.

### Purchase Ticket via Agent Diagram & Description

#### Diagram



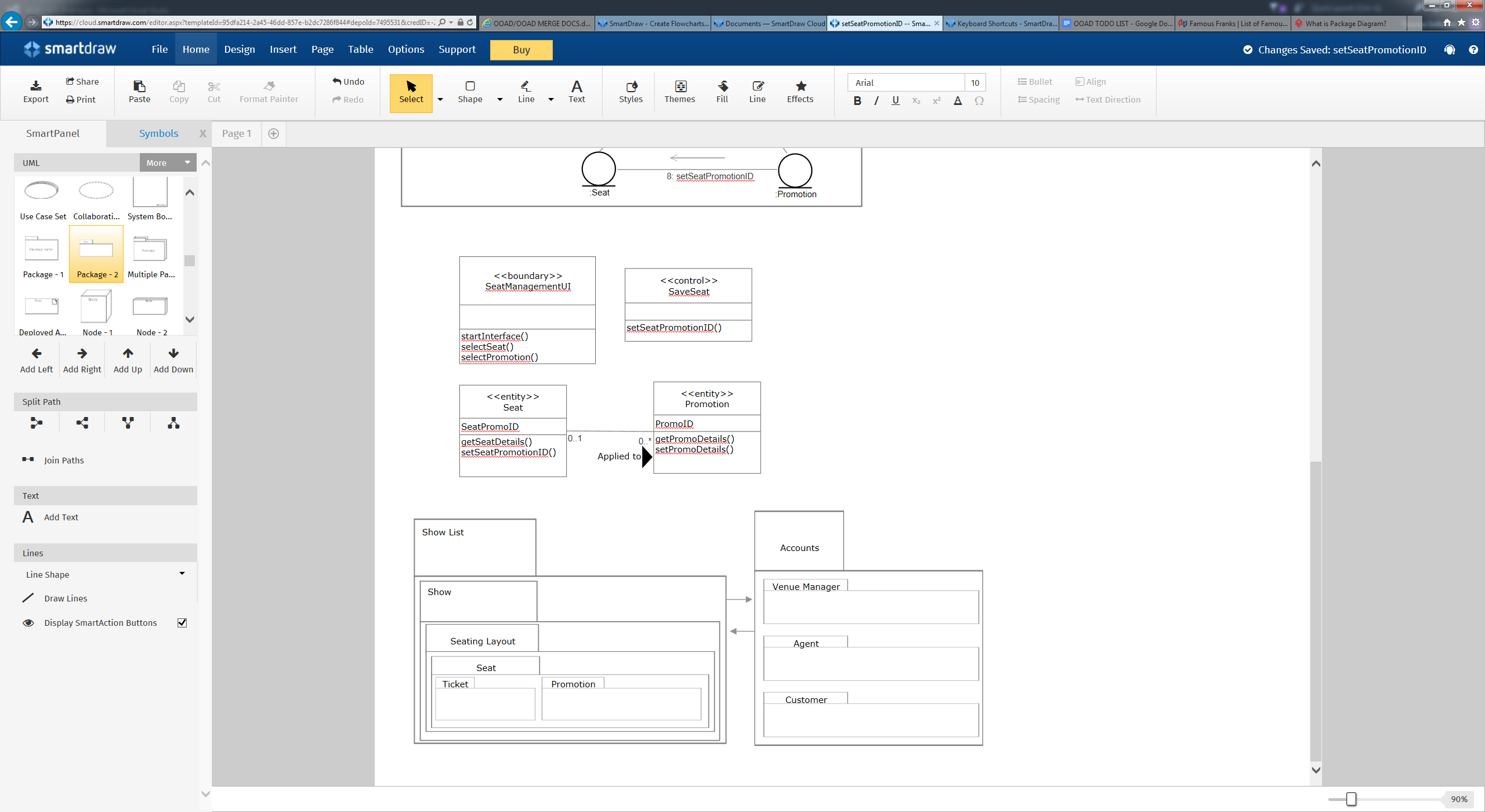
#### Description

This diagram shows the communication order from purchase request, throughout its lifetime across the customer agent and seat class.

## Architecture Models

### Package Diagram

This diagram is an initial realisation of the overall structure that classes will have and how they will interact with each other from an architectural viewpoint.



## Glossary of Functional Requirements

|  |  |  |
| --- | --- | --- |
| No. | Annotation Description | Use Case(s) |
| 1 | To create and record names, times, base price, descriptions and venue details for each show. | Add new show |
| 2 | To edit details such as name, date, time, description, and base price. | Edit show details |
| 3 | To create a venue seating plan reflective of the venue layout. | Create seating layout |
| 4 | To create and save individual seat objects to hold information such as, location, price, status, quality, assigned agent and assigned promotion. | Create seat |
| 5 | To save seating layouts to be used as templates for future shows. | Save seating layout |
| 6 | To load previously saved seating layouts. | Load seating layout |
| 7 | To add promotional offers to individual seats. | Add promotion |
| 8 | To save promotions for future use. | Save promotion |
| 9 | To load previously saved promotions. | Load promotion |
| 10 | To add discounts to individual showings of a show. | Add discount |
| 11 | To save discounts for future use. | Save discount |
| 12 | To load previously saved discounts. | Load discount |
| 13 | To create agent accounts and assign them Agent permissions to execute privileged tasks. | Assign agent permission |
| 14 | To assign seats in a show to an agent so that they can sell tickets for them. | Assign agent to seat |
| 15 | To allow agents to fulfil purchase requests on behalf of the customer. | Purchase ticket on behalf of customer |
| 16 | To allow venue managers to cancel tickets and or refund ticket cost. | Refund/cancel ticket |
| 17 | To allow agents and venue managers to find customer accounts, display account information and purchase history information. | Find customer account |
| 18 | To allow customers to register accounts and store mandatory account information such as; name; billing info; payment info; contact info etc. | Create customer account |
| 19 | To allow customers to edit and update their saved account information. | Edit customer account |
| 20 | To display a list of all available shows in date order. | Display all shows |
| 21 | To search shows by name and or date range. | Search show(s) |
| 22 | To search for the best seats based on price range. | Search seat(s) with price range |
| 23 | To select desired seats. | Select seat(s) |
| 24 | To hold desired seats while a customer purchases them. | Hold seat(s) |
| 25 | To allow the user to log into their account with their sign up email address and password. | Login |
| 26 | To allow the user to safely log out of their account. | Logout |
| 27 | To allow the customer to purchase a ticket or tickets either through seat selection or automatic selection of best available seats. | Purchase Ticket |
| 28 | To allow the venue manager to modify an existing seating layout. | Modify seating layout |
| 29 | To allow the venue manager to modify a seat within an existing layout. | Modify seat |
| 30 | To allow the venue manager to view all purchases or to allow a customer or agent to view purchases they have made. | View Purchase History |
| 31 | To allow the venue manager or agent to locate a customer account either to view its details or make a purchase on its behalf | Locate Customer |

# Requirements Analysis

This is the Requirement Analysis section which will be covering; Class diagrams; Communication diagrams and Analysis models.

## Class Diagrams & Models

The below classes demonstrate the available methods to each user type, the majority will be globally inherited with a few unique to individual user types, however the majority of data is stored in individual seat objects.

### Class Names

Based on the extensive requirements modelling these class names have been developed.

#### Non-Person Classes:

Show, Seating Layout, Seat, Promotion, Ticket

#### Person Classes:

Person, Venue Manager, Agent, Customer

### Class Structures

This is a rough outline of class inheritance structure, where the left-most class is the superclass / parent class.

#### Non-Person Classes:

Show 🡪 Seating Layout 🡪 Seat 🡪 Promotion 🡪 Ticket

#### Person Classes:

Person 🡪 Venue Manager 🡪 Agent 🡪 Customer

### Class Data & Methods

#### Non-Person Classes

|  |  |
| --- | --- |
| **Show** | |
| **Data** | ShowID, ShowDateTime, ShowName, ShowLengthMins, SeatLayoutData, ShowDiscount, ShowBasePrices, ShowSaveLocation, ShowState |
| **Methods** | setShowName(), setShowDateTime(), setShowLength(), setShowBasePrices(), setShowDiscount(), saveShow(), setSeatLayoutData (), setShowState(), getShowName(), getShowDateTime(), getShowLength(), getShowBasePrices(), getShowDiscount(), getSeatLayoutData(), getShowState(), loadSeatingLayout(), loadShow(), initialise() |

|  |  |
| --- | --- |
| **Seating** **Layout** | |
| **Data** | MaxSeatsPerCustomer, SeatLayoutImageLocation, SeatLayoutImageSizePx, SeatImagePositionsPx, SeatsData |
| **Methods** | setMaxSeatsPerCustomer(), setSeatLayoutImageLocation(), getImageSize(), setSeatLayoutImageSizePx(), setSeatImagePositionsPx(), setSeatsData(), getMaxSeatsPerCustomer(), getSeatLayoutImageLocation(), getImageSize(), getSeatLayoutImageSizePx(), getSeatImagePositionsPx(), getSeatsData(). getLocation(), loadSeatDataSet(), saveSeatingLayout(), loadExistingLayout(), initialise() |

|  |  |
| --- | --- |
| **Seat** | |
| **Data** | SeatPrice, SeatTimer, SeatImagePositionID, SeatPromotionID, SeatStatus, ControllingAgentID |
| **Methods** | setSeatPrice(), setSeatTimer(), setSeatImagePositionID(), setSeatPromotionID(), setSeatAgentID(), setSeatStatus(), getSeatPrice(), getSeatTimer(), getSeatImagePositionID(), getSeatPromotionID(), getSeatControllingAgentID(), getSeatStatus(), playTimerAlarm(), saveSeat(), loadExistingSeat(), getSeatDetails(), initialise() |

|  |  |
| --- | --- |
| **Promotion** | |
| **Data** | promotionName, promotionType, promoPriceReductionPercentages, fixedPromotionalTicketPrices |
| **Methods** | setPromoName(), setPromoType(), setPriceReductionPercentages(), setPromoPriceReductionPercentages(), getPromoName(), getPromoType(), getPriceReductionPercentages(), getPromoPriceReductionPercentages(), savePromotion(), loadExistingPromotion(), getPromoDetails(), initialise() |

|  |  |
| --- | --- |
| **Ticket** | |
| **Data** | TicketID, TicketOwner, TicketText, TicketState |
| **Methods** | setTicketOwner(), setTicketText(), setTicketState(), getTicketOwner(), getTicketText(), getTicketState(), saveTicketImage(), createTicketImage(), initialise() |

#### Person Classes

|  |  |
| --- | --- |
| **Person** | |
| **Data** | Title, FirstName, MiddleName, LastName, DOB, Gender, PhoneNumber, EmailAddress, AddressL1, AddressL2, AddressTown, AddressPostCode, AddressCounty, AddressCountry, Password, UserLevel |
| **Methods** | setTitle(), setFirstName(), setMiddleName(), setLastName(), setDOB(), setGender(), setPhoneNumber(), setEmailAddress(), setAddressL1(), setAddressL2(), setAddressTown(), setAddressPostCode(), setAddressCounty(), setAddressCountry(), setPassword(), SetUserLevel(), getTitle(), getFirstName(), getMiddleName(), getLastName(), getDOB(), getGender(), getPhoneNumber(), getEmailAddress(), getAddressL1(), getAddressL2(), getAddressTown(), getAddressPostCode(), getAddressCounty(), getAddressCountry(), getPassword(), getUserLevel(), getUserDetails() |

|  |  |
| --- | --- |
| **Venue Manager** | |
| **Data** | LoginHistory |
| **Methods** | setLoginHistory(), getLoginHistory(), |

|  |  |
| --- | --- |
| **Agent** | |
| **Data** | BookingHistory, ActiveAssignedSeats |
| **Methods** | setBookingHistory(), getBookingHistory(), setActiveAssignedSeats, getActiveAssignedSeats() |

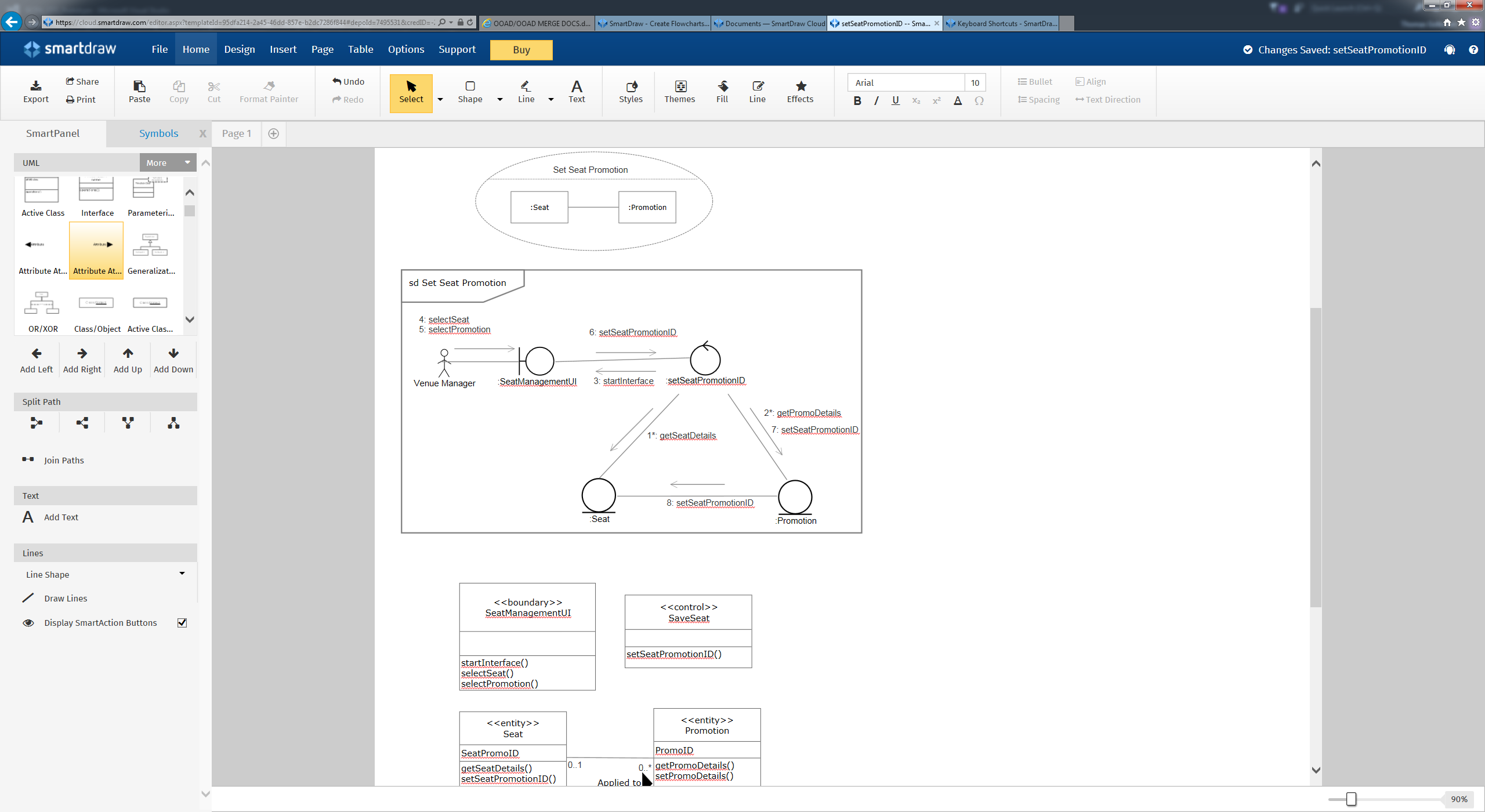
|  |  |
| --- | --- |
| **Customer** | |
| **Data** | CardNo, CVC2, ExpiryDate, CardName, CardBillingAddress, DefaultPaymentMethod |
| **Methods** | setCardNo(), setCVC2(), setExpiryDate(), setCardName(), setCardBillingAddress(), setDefaultPaymentMethod(), getCardNo(), getCVC2(), getExpiryDate(), getCardName(), getCardBillingAddress(), getDefaultPaymentMethod() |

### Collaboration, Communication, and Class Diagrams

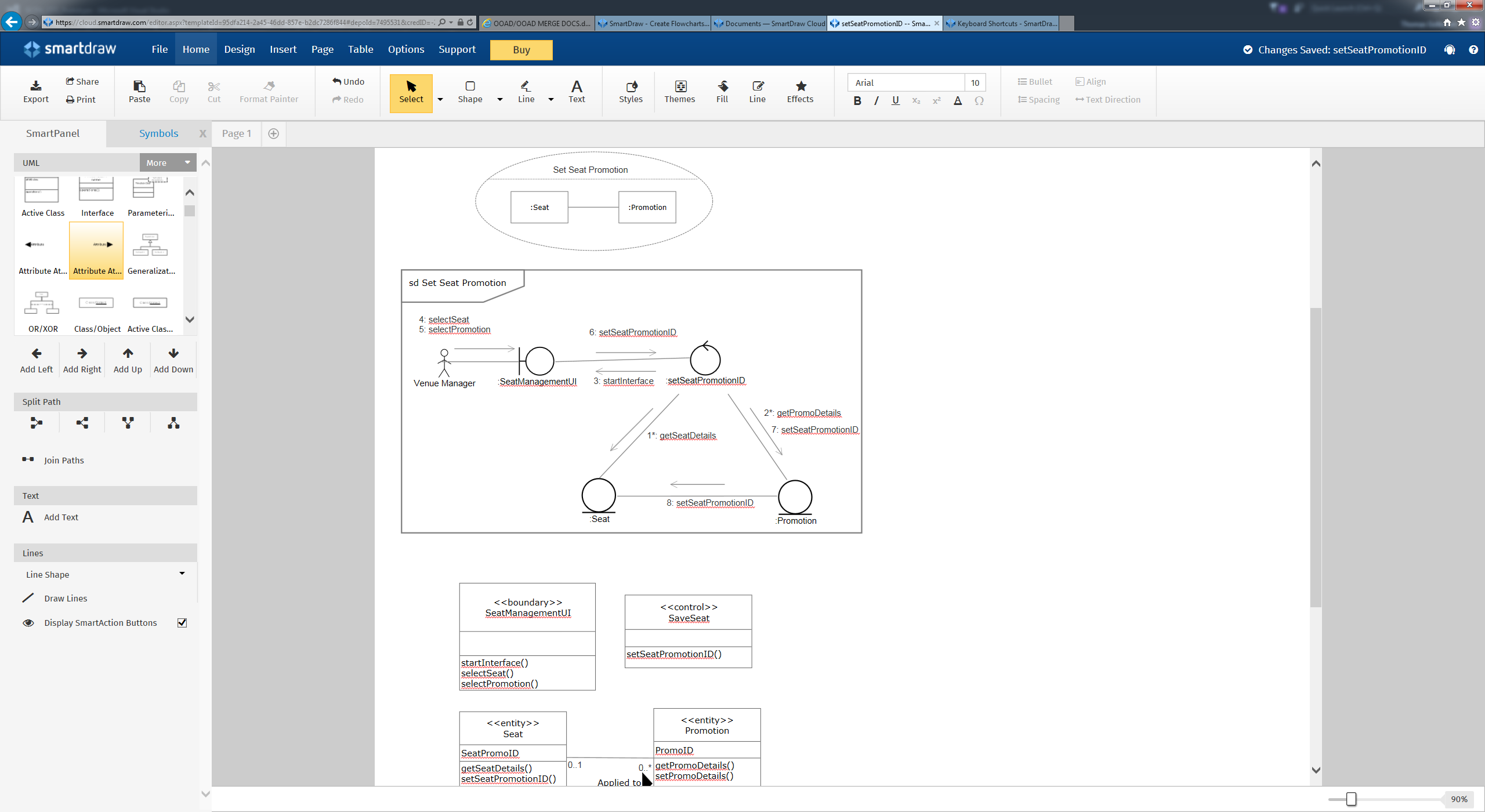
#### Set Seat Promotion

This set of diagrams refers to setting the promotion that is applied to a seat.

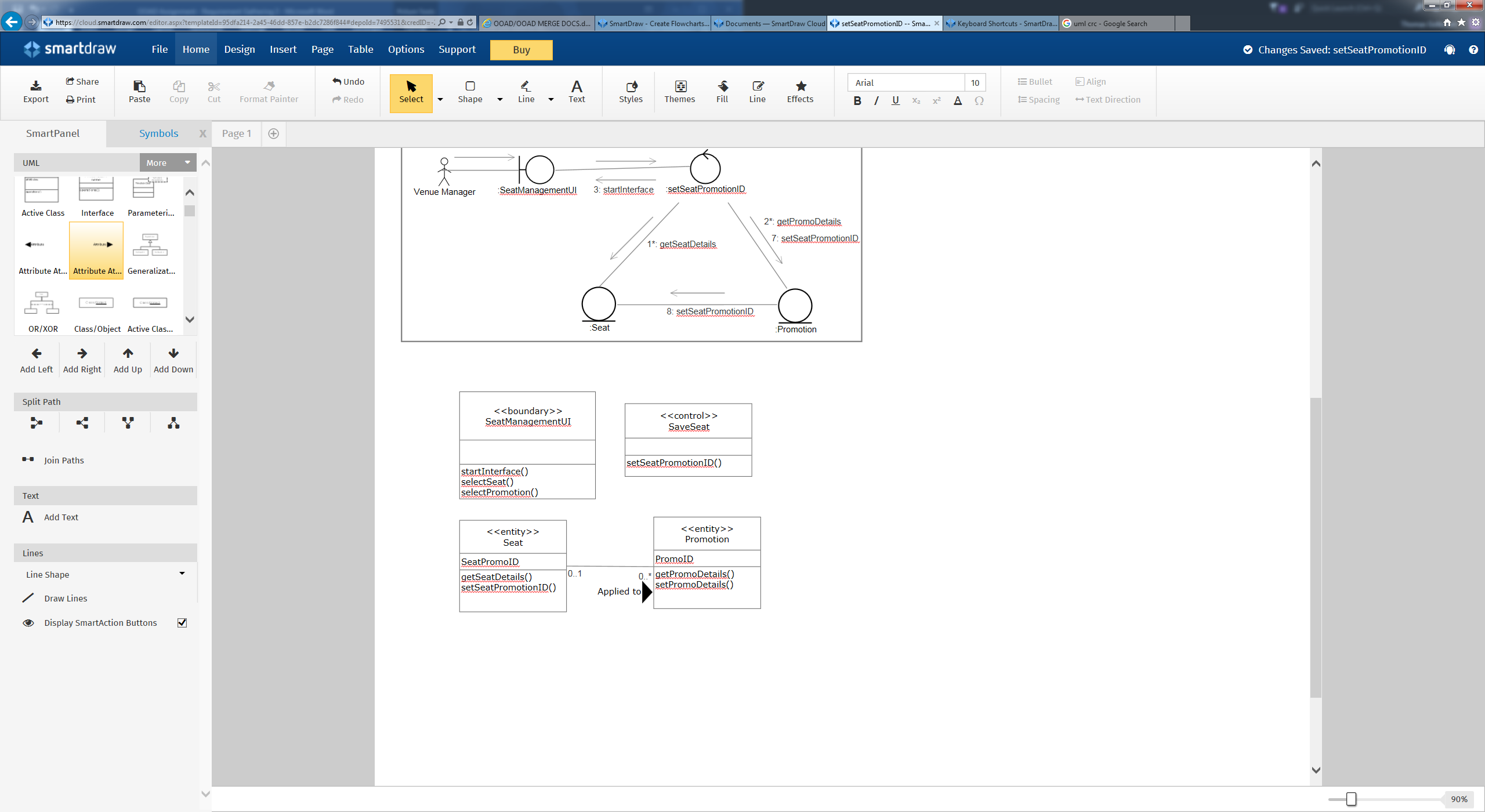
##### Collaboration Diagram



##### Communication Diagram



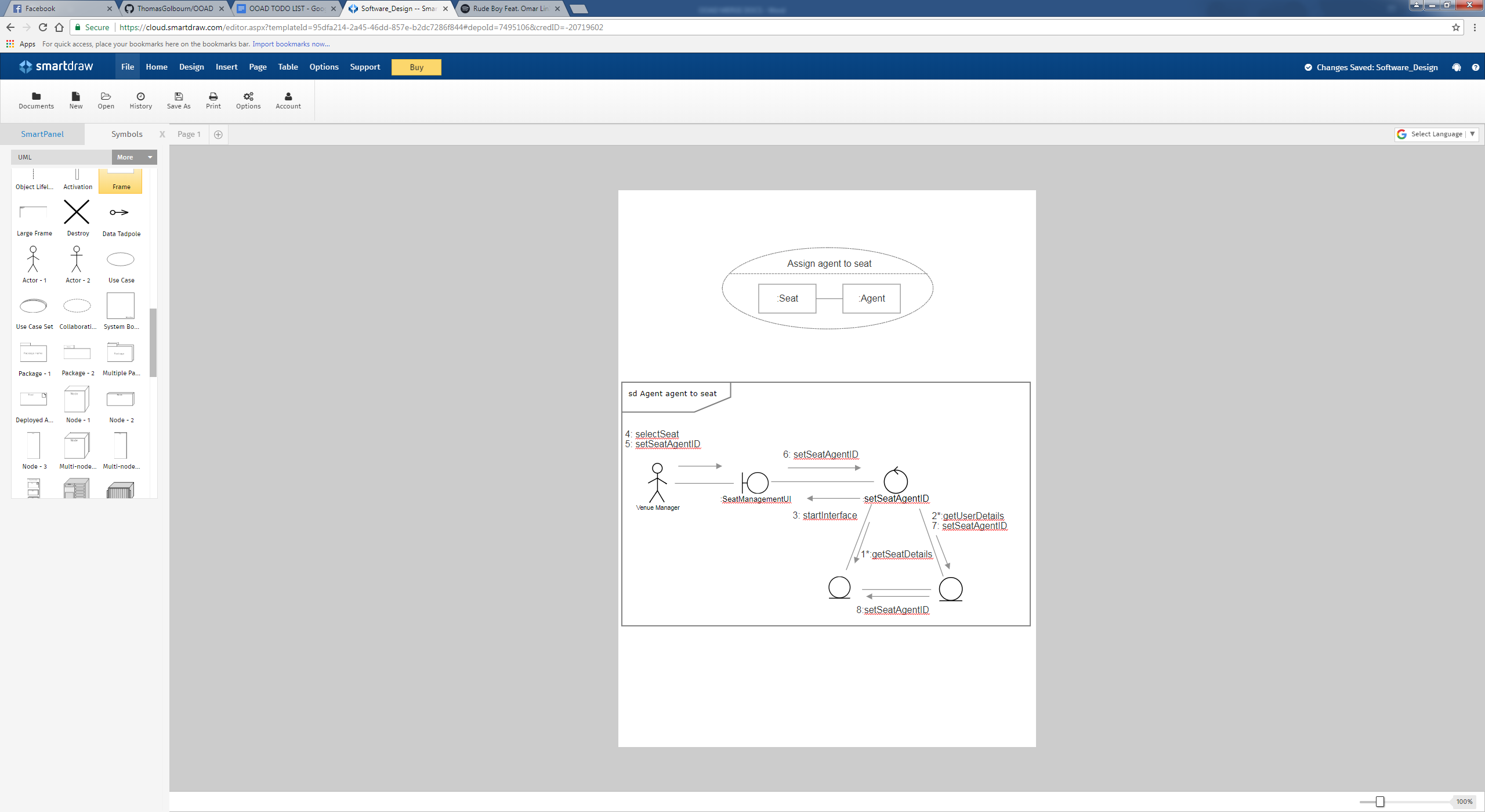
##### Class Diagram



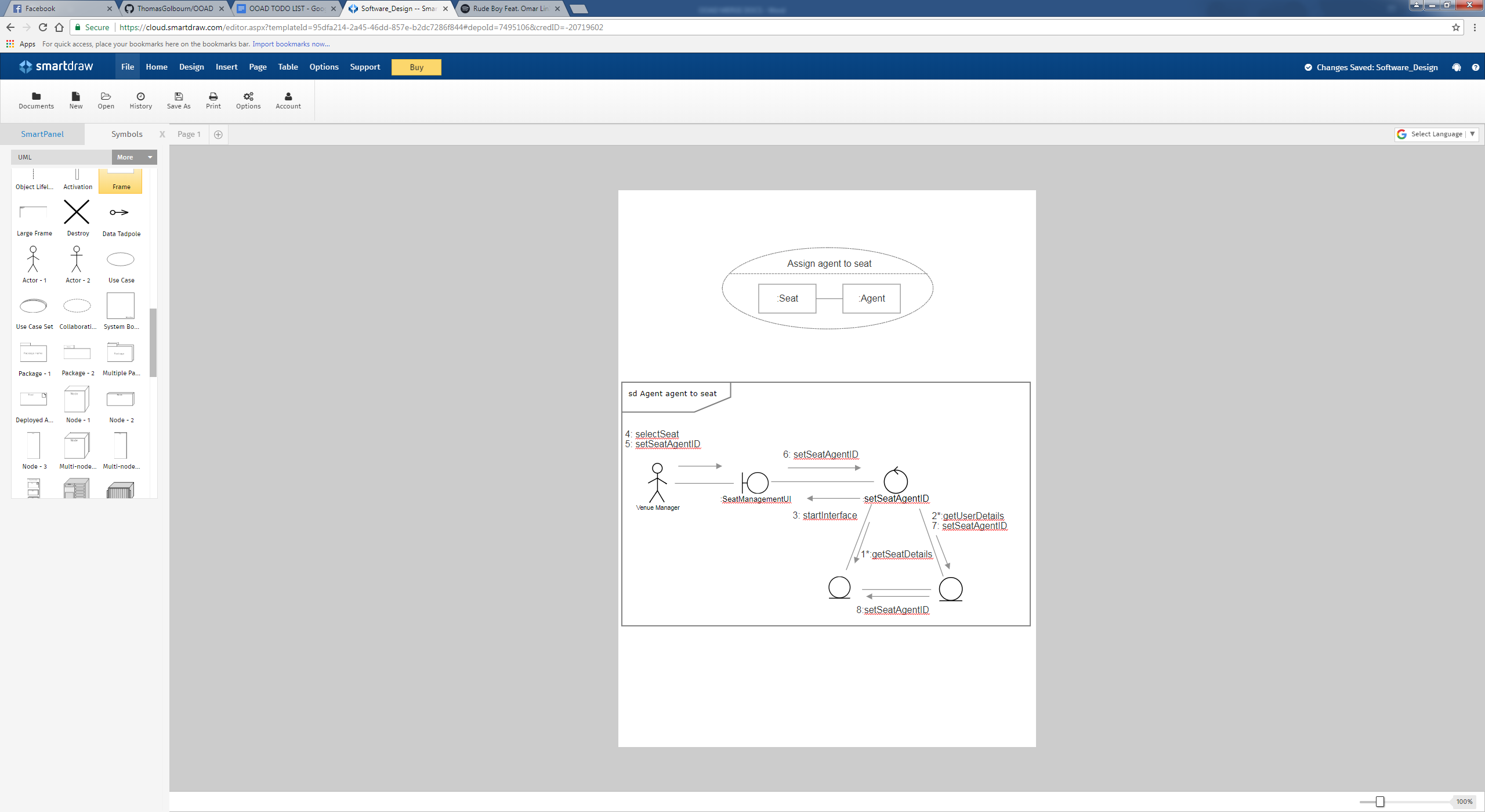
#### Assign Agent to Seat

This set of diagrams refers to assigning a controlling agent to a seat.

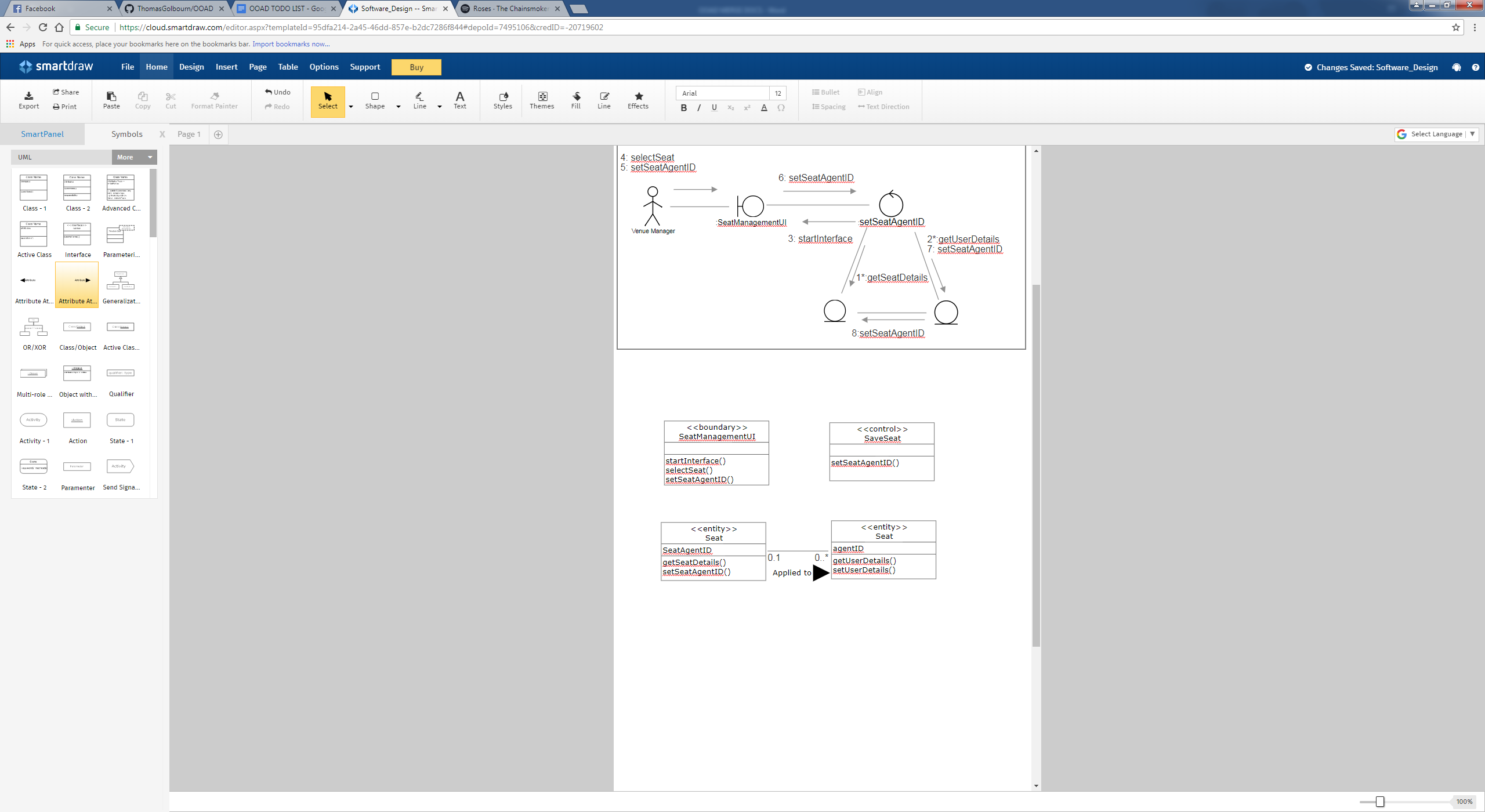
##### Collaboration Diagram



##### Communication Diagram



##### Class Diagram



### Class Responsibility Collaboration (CRC) Cards

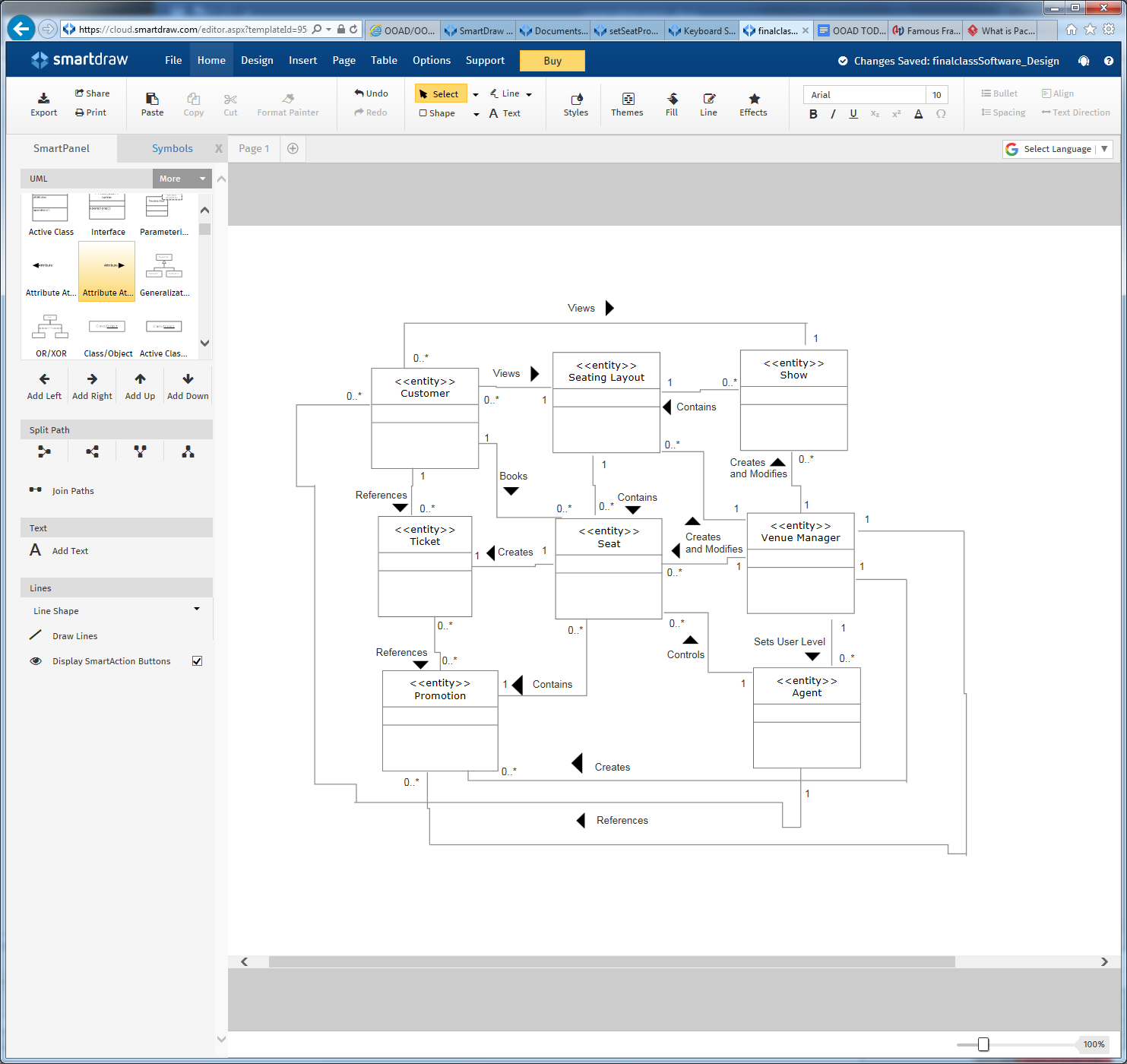
|  |  |
| --- | --- |
| **Seat** | |
| **Responsibilities** | **Collaborations** |
| **Stores information such as seat price, promotions, owned agent, seat location, seat name** | **The seat object collaborates with the seating layout manager and discounts** |

|  |  |
| --- | --- |
| **Agent** | |
| **Responsibilities** | **Collaborations** |
| **To fulfil Purchase requests on behalf of the customer.** | **Collaborates with seat to fulfil purchases** |

|  |  |
| --- | --- |
| **Customer** | |
| **Responsibilities** | **Collaborations** |
| **Purchases tickets or creates Ticket Purchase Requests.**  **Updates account information on file** | **Collaborates with seat to purchase ticket**  **Collaborates with Person to edit account information** |

## Analysis Class Diagram

This analysis class diagram intends to show the way in which all of the classes interact in a broader sense rather than focusing on individual use cases.



*Contained Methods & Data can be found in Section 5.1.2*