Content: Page

Activity 1: **'One Simple Booking System' (OSBOS)**

Introduction

This is a plan for renewing the current booking system to a fully computerised "One Simple Booking System", which will provide an easy, efficient and reliable way to manage data and processes. The Playhouse is a small and well respected theatre, located in a picturesque University town of Camford. It consists of two auditoria:

Auditoria	The Stearne Place	The Persephone's Pit (Pomegranate)
Can hold	623 people	about 250 people
Stages	- Conventional play - Public lectures - Musical performances - Pantomime	- Solo concert artists - Informal and amateur Events - Normal theatre productions

The theatre is hoping to be amongst the first to install such a system with a view to possibly selling it to other theatres.

The Current booking system

The theatre booking system is very old fashioned and is basically paper-based. Tickets can be purchased from the Theatre Box Office, via face to face transaction or telephone sales and this only during business and performance hours. Some tickets are allocated to local booking agencies, mainly travel agents, unsold tickets have to be returned within two days before the relevant performance, in order to be available for late bookings and avoid double booking problems.

The new functions of the system

The new functions of the system will make the booking procedure, easier, secure and more organised. Some of the key new functions are listed below:

- Customer mailing list via Mail Merger in Microsoft Word
- Banned list via Acces Reports
- Seat availability
- Refund money
- Handle discounts easier
- Management information data (e.g sales to performances) via Acces Reports
- Returning tickets / handle refunds
- Customer information

The new system will provide a structured way to handle bookings; giving the management all the information to make decisions on time and with real-time facts. These functions will supply your customer's better service and satisfy their needs.

Data storage requirements

Many Organisations are beginning to realise the importance of retaining their electronic files for extended periods of time. Therefore the relational database will store following information:

- Auditoria
- Performance
- Seat
- Ticket
- Transaction Type
- Transaction
- Discount
- Customer

This information is useful to see past, present statistics and make future predicitions from these facts.
The customers will be notified (on the ticket) that their personal details are documented, and will only be used for safety and internal statistics. (Data Protection Act 1998)

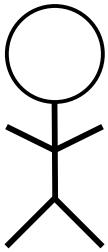
Overview of the whole system

The system/application will be Microsoft Windows compatible, the staff just needs to log on to Windows and start the application. Booking can be done at the box office personally which gives the management an exact figure of the attendance for a specific performance.
The new check for seat feature helps to have a better overview of the autitorias, providing better service to disabled people and allocations of seats to customers to their wishes.
A banned list is also implemented; customers are asked to show some proof of Identity, the banned check box is visible on the screen when you read the customer record. And if the check box is ticked the customer is banned from the theatre. If the check is not ticked, the customer is will be provided with a ticket, this ensures the safety of the customers. Customer information is also stored to help maintain/establish a new mailing list, which will contain information about future activities and the current monthly timetable of the theatre and its performances. The customer information will only be available to the management and the staff.
A new discount system is realised, to handle the different discounts types, this can help the management to see what ticket types were sold to which performance.
Customers which want to return their ticket will now have the facility to do so. They have to return their tickets latest two hours before the performance starts. They will be refunded with the amount spend on the ticket with some small cancellation fee deducted from the ticket price, due to the occured administration costs. As soon as the ticket is set as returned into the system, this seat will be available again to the customers.

Activity 2: **Detailed account of the use case entitled "check for seats"**

d) Identification of Actors

The main actor using this use case is:



Booking agent

e) A use case description

Use case number: 6	check for seats
Goal	Handle the functionality associated with checking for seats in a given performance.
Description	Customer approaches the system at the theatre to look for a performance (time and date) of their choice. Then the customer looks for available seat/s for this particular performance, by looking at the seat availability map.
Actors **Constraints**	Booking agent – a privileged user who may: ▪ Check for available seats ▪ Make booking ▪ Manage bookings & tickets ▪ Print ticket ▪ Manage customer accounts ▪ Update seats availability ▪ Manage booking cancellations ▪ Manage returns ▪ Manages mailing lists The booking agent has a windows user name and password and he uses it to log on to the computer. This use case must execute in under two minutes with a mean execution time of one minute or less. Booking agent should be able to learn the associated activities in under one hour. Screen dialogs should be readable to people with averagely poor eyesight.
Pre-conditions	For the main success scenario these are: • the booking system is running • the performance the customer wants to visit, is available • the customer is not banned • the customer has a valid user account
Main success scenario	The booking agent does successfully check for seats. The scenario has following stages: 1. Booking agent checks the status of the User 2. Booking agent locates performance data (time, date) 3. Booking agent looks up the current status of the seat availability for this given performance
Other scenarios	1. Customer attempts to visit a performance which does not exist 2. Booking agent is not trained enough to use the system and therefore books wrong performances
Related use case	Use Case 5 – Check customer status All of these use cases involve the Booking agent.

Frequency of occurrence in the system	The frequency of this use case will be one of the major scenarios in the system. It involves about 90% of the time that make booking occurs.
Notes	We assume that booking agent has no trouble getting into the computer system but any password-protected system will occasionally cause problems. We should check on theatre policy with respect to login problems. The following terms are defined in the system glossary ▪ Transactions ▪ Seats ▪ Tickets ▪ Customer ▪ Booking agent ▪ Book ticket/Make booking

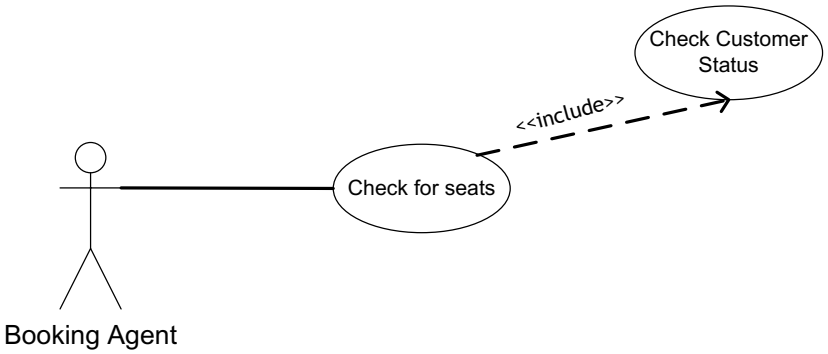


Figure 1: Use case - Check for seats

f) A high level account what the use case archives

Customer approaches Booking agent with a verbal request to book a ticket(s) for a certain performance. The Booking agent checks the customer data against his identity, if he/she is banned at the theatre. If they are not banned, the Booking agent uses the system to locate the available seats for the performance on a certain date/time per customer preference.

g) Identification of a "main success scenario" plus any other, appropriate, scenarios

Main success scenario

1. The customer approaches the Booking agent and tells him/her he would like to visit a performance on a certain date/time preference.
2. The booking agent asks the customer to show him/her some proof of identity.
3. Customers status/identity is then checked against the banned database.
4. If the Customer is not listed in the banned database; the booking agent gives the customer preference into the system, via the date & time.
5. The Booking agent then looks up the status of the available seats for the certain performance.

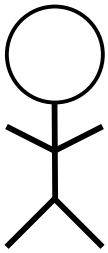
Possible other scenario

1. The customer approaches the Booking agent and tells him/her he would like to visit a performance on a certain date/time preference.
2. The booking agent asks the customer to show him/her some proof of identity.
3. Customers status/identity is then checked against the banned database.
4. The customer is listed in the banned database; the booking agent will cancel the request of the customer.

Activity 3 **Detailed account of the use case entitled "make booking"**

h) Identification of Actors

The main actor using this use case is:



Booking agent

i) A detailed use case description

Use Case Number: 2	**Make Booking**
Goal	Handle the functionality associated with making a booking for ticket.
Description	Customer approaches Booking agent with a verbal request to book a ticket(s) for a certain performance. Booking agent uses the system to locate the available seats for the performance. If requested seat is available or the customer selects one from the available seats, the performance preferences (ticket type, ticket quantity, drinks) are selected, payment process is completed and the booking is confirmed. The ticket is then printed and handed over to the customer. If no available seat is found for the performance requested by the customer, the booking process is cancelled.
Actors	Booking agent – a privileged user who may: ▪ Check for available seats ▪ Make booking ▪ Manage bookings & tickets ▪ Print ticket ▪ Manage customer accounts ▪ Update seats availability ▪ Manage booking cancellations ▪ Manage returns ▪ Manages mailing lists
Constraints	The booking agent has a windows user name and password and he uses it to log on to the computer. This use case must execute in under two minutes with a mean execution time of one minute or less. Booking agent should be able to learn the associated activities in under one hour. Screen dialogs should be readable to people with averagely poor eyesight.

Pre-conditions	For the main success scenario these are: ▪ The booking system is up and running. ▪ The box office is open for making ticket bookings. ▪ The customer requesting for booking ticket is not a banned customer. ▪ The customer wants to buy a ticket. ▪ The booking agent is an authenticated windows user and logged into the computer. ▪ If a customer doesn't exist already then he is added first.
Main Success Scenario	The booking agent successfully books a ticket. The scenario has these stages. 1. Booking agent checks customer status. 2. Booking agent checks available seats. 3. Booking agent selects seat. 4. Booking agent selects ticket type and quantity. 5. Booking agent enters drinks code. 6. Booking agent calculates bill. 7. Booking agent completes payment process. 8. Booking agent confirms booking. 9. Booking agent prints ticket.
Post conditions for main success scenario	▪ The transactions database is updated. ▪ The tickets table is updated.
Assumptions	1. The version 1 of the booking system only caters for box office bookings. 2. The version 1 depends on windows login module for safety ans security.
Other scenarios (named in brackets)	1. The PlayHouse theatre is not currently offering tickets for bookings. (System Failure/Holiday) 2. Information entered is not in valid format. (Incorrect format) 3. All mandatory fields are not provided. (Missing fields/info) 4. The booking agent fails to log in to the computer system. (System has been compromised?) 5. The performance of whose tickets are requested is completely sold out. 6. The performance requested by the customer doesn't exist, got cancelled or ended. 7. The customer doesn't want to book other available seats than what he requested. 8. The credit card is not validated and customer is not carrying cash.
Related use cases	Use case 5 – Check customer status Use case 6 – Check for seats Use case 7 – Make payment Use case 8 – Print Ticket All of these use cases involve booking agent.
Frequency of occurrence in the system	The main success scenario is one of the two commonest scenarios in the system (the other being Cancel booking). It represents about 46 per cent of the activities involving booking agent.
Test generation	▪ Each of the nine scenarios described must be tested. ▪ The timing constraint is fairly lax but should be checked. In practice it is unlikely to be an issue in this system but for contractual reasons it may be appropriate to include auxiliary code to collect data on how long booking agents are taking to book tickets, when the alpha system is available, and this may have a bearing on interface design.
Notes	We assume that booking agent has no trouble getting into the computer system but any password-protected system will occasionally cause problems. We should check on theatre policy with respect to login problems. The following terms are defined in the system glossary ▪ Transactions ▪ Seats ▪ Tickets ▪ Customer ▪ Booking agent

	▪ Book ticket/Make booking

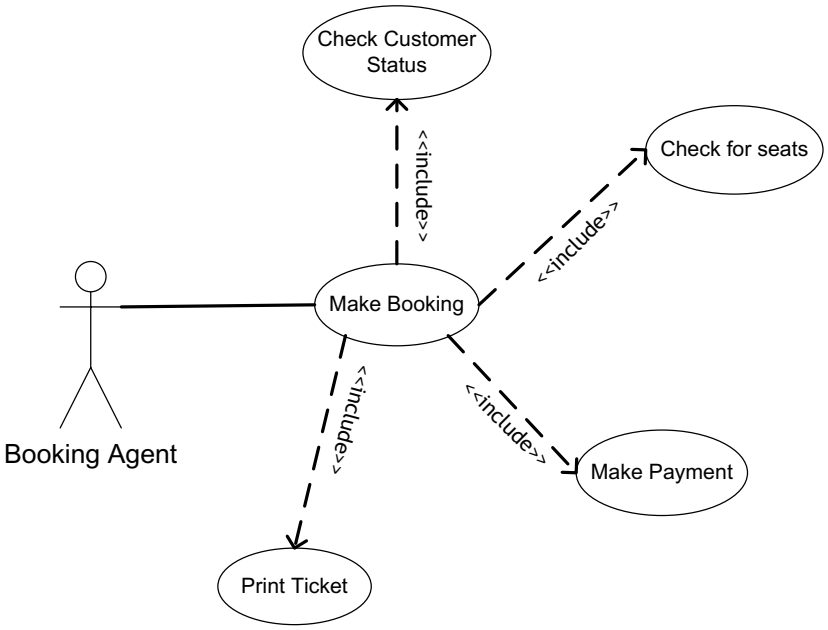


Figure 2: Use Case Diagram - Make Booking

j) A high level account of what the use case achieves

Customer approaches Booking agent with a verbal request to book a ticket(s) for a certain performance. Booking agent uses the system to locate the available seats for the performance on a certain date/time per customer preference. If requested seat is available or the customer selects one from the available seats, the performance preferences (ticket type, quantity of tickets, drinks) are entered, payment process is completed and the booking is confirmed. The ticket is then printed and handed over to the customer. If no available seat is found for the performance requested by the customer, the booking process is cancelled.

k) Identification of main success scenario, plus other appropriate scenarios

1. This use case involves the actor 'Booking agent'.
2. This use case occurs when someone comes to book a ticket and give their preference details for ticket request.
3. The **pre-condition** is composed of a booking agent who has been trained to use the booking software.
4. The booking agent checks the customer status.

5. If the customer is banned then the booking process is cancelled.
6. If the customer is not banned, the booking agent continues with the booking process.
7. The system shows the performance titles.
8. The booking agent selects performance title.
9. The system displays dates & time for the performance.
10. The booking agent selects a date and time (as requested by the customer).
11. The system shows the available seats list.
12. If no available seats are found, the booking process is cancelled.
13. If available seats are found, then booking agent selects a seat that customer requested.
14. If the requested seat is already booked, the booking agent asks from the customer to select from the other available seats.
15. If the customer doesn't want to go for other than his original preferred seat, the booking process is cancelled.
16. If the customer picks one of the available seats, the booking agent selects the seat.
17. The booking agent selects the ticket type and quantity.
18. The booking agent enters drinks code if user requested to order a drink.
19. The booking agent calculates bill.
20. The booking agent chooses the mode of payment.
21. If the customer is paying via a credit card, the booking agent enters the credit card.
22. If the credit card is not validated, the customer is informed and the booking process is cancelled or customer pays cash.
23. If the credit card is validated, the booking agent confirms the booking.
24. If the customer pays using cash, the process is manually done.
25. The booking agent confirms booking.
26. The booking agent prints the ticket and gives to the customer.
27. Success state changes include
    (i) Updating the transactions records
    (ii) Updating the tickets database
28. Point 27 represents the **post-conditions** of the main success scenario.
29. Notes:
    a) The price of a ticket depends on the performance and seat booked.
    b) There are categories of customers which have correspondingly different discount policies.
    c) Step 7-11 falls under the use case 'check seats'
    d) In version 1 we assume to keep security and safety using windows login procedure and the system login access should be added in version 2 of the booking system.
    e) If a customer doesn't exist in database then he is added to it and then we proceed with booking.

This use case description reveals that this analysis implies that other use cases are involved in its execution. We could identify:

Check customer status
Check for seats

Make payment
Print ticket

Main Success Scenario:

1. Booking agent checks customer status.
2. Booking agent checks for available seats.
3. Booking agent selects seat.
4. Booking agent selects ticket type and quantity.
5. Booking agent enters drinks code.
6. Booking agent calculates bill.
7. Booking agent completes payment process.
8. Booking agent confirms booking.
9. Booking agent prints ticket.

Other Possible Scenario:

1. Booking agent can not log in to the computer system.
2. The customer is banned by the Playhouse theatre.
3. The performance is sold out whose verbal ticket request customer made.
4. The performance requested by the customer doesn't exist, got cancelled or ended.
5. The customer doesn't want to book other available seats than what he requested.
6. The credit card is not validated and customer is not carrying cash.
7. The booking system is compromised.
8. The box office is not open for bookings.

Activity 4: **Technical specification of three distinct system functions**

Identifier	Add Customer
Parameters	(ID:Long Integer) (Title:String) (Firstname:String) (Lastname:String) (Email:String) (Status:Boolean) (Address:String) (Mailing:Boolean) These are the fields that are needed in order to add a customer to the Customer table in the database.
Return types	Boolean The return value will be true if the customer is added successfully and false if the customer is not successfully added to the database.
Informal description	This function is used when a new customer needs to be added to the database. There are two main times when this may occur, these are: • When a customer wishes to be added to the Mailing list to be kept informed of theatre productions or announcements • When a customer wishes to buy tickets for a performance Notes that customer details may also be needed for other function such as when a customer makes a cancellation but in that circumstance the customer must already exist on the database.
Notes	**Pre-conditions** A new (not previously existing on the database) customer needs to be added to the database. **Post-conditions** A new Customer record is added to the Customer table in the database. **Error-conditions** ▪ Errors will be generated if required fields for a customer are not correctly filled in, required fields (and all other fields) are described in full in the Entity Relationship Model. **Other methods used** n/a

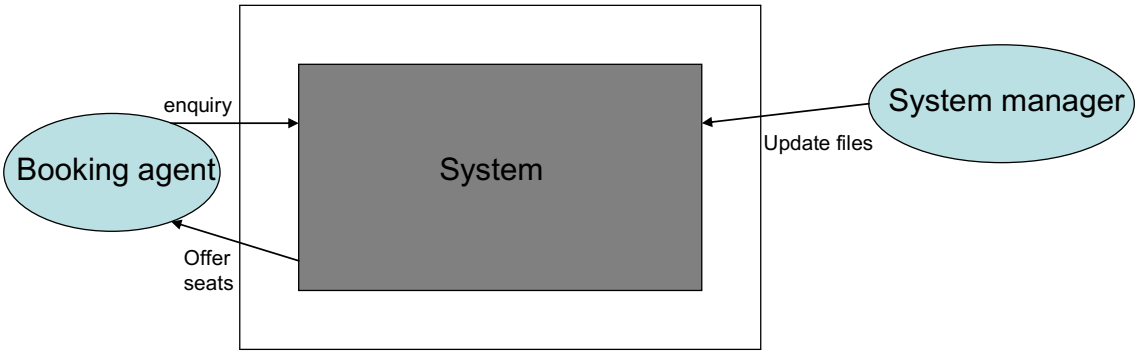
Identifier	Make booking
Parameters	n/a
Return types	Boolean The return value will be true if the booking is successfully made and false if the booking fails.
Informal description	Customer approaches Booking agent with a verbal request to book a ticket(s) for a certain performance. Booking agent uses the system to locate the available seats for the performance on a certain date/time as per customer preference. If requested seat is available or the customer selects one from the available seats, the performance preferences (ticket type, drinks) are entered, payment process is completed and the booking is confirmed.

	The ticket is then printed and handed over to the customer.
Notes	**Pre-conditions** ▪ An authenticated booking agent who has been trained to use the booking software ▪ A customer wants to buy tickets ▪ A customer record exists on the database ▪ The booking system is up and running ▪ The box office is open for making ticket bookings ▪ The customer requesting for booking ticket is not a banned customer **Post-conditions** ▪ Updating the transactions records ▪ Updating the tickets database **Error-conditions** ▪ Customer is banned ▪ No seats are available ▪ Payment fails ▪ Information is not entered in a valid format ▪ Mandatory fields are not completed - required fields (and all other fields) are described in full in the Entity Relationship Model. **Other methods (sub-functions) used** ▪ Check customer status ▪ Check seats ▪ Make payment ▪ Print Ticket

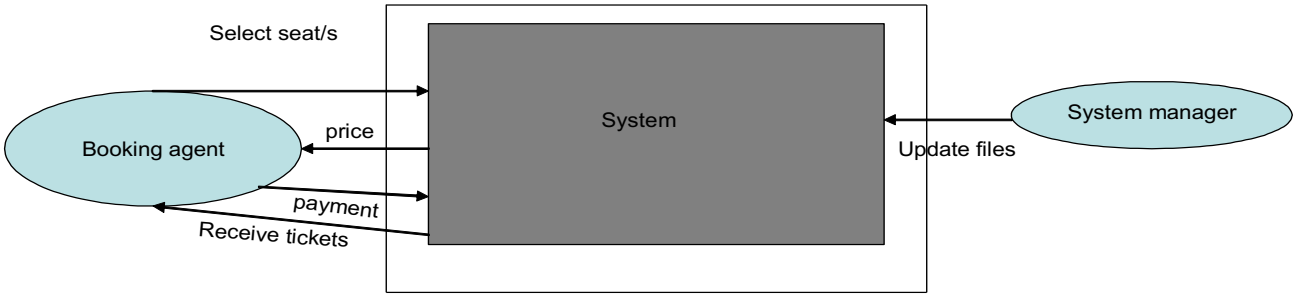
Identifier	Select seat
Parameters	(SeatNumber:String) (performanceID:LongInteger) This is the id of the seat that needs to be located
Return types	Boolean The return value will be true if the seat is available and false if the seat is not available.
Informal description	This is the function that is used when a customer wishes to make a booking, it is one of the sub-functions of that process and is used to select a seat for which a ticket(s) will be issued
Notes	**Pre-conditions** ▪ A customer wishes to make a booking **Post-conditions** ▪ The availability of a seat is decided **Error-conditions** ▪ Errors may occur if an invalid seat number is entered ▪ Errors may occur if an invalid Performance ID is entered **Other methods (sub-functions) used** ▪ Check performance ▪ Check seats

Activity 5: **Data flow diagrams relating to two distinct functions**

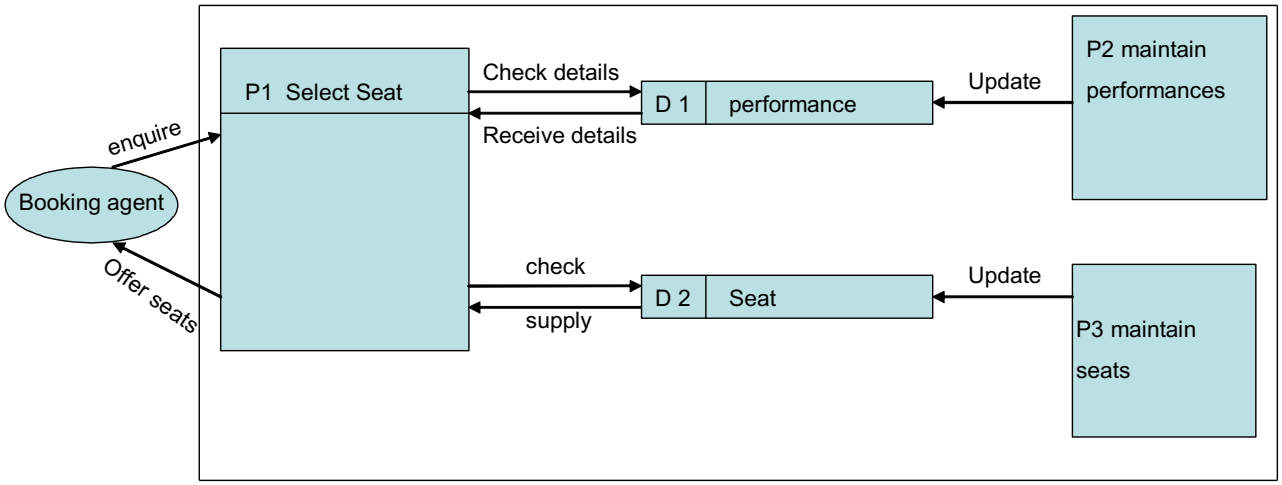
Level 0 DFD for Playhouse theatre, Select Seat



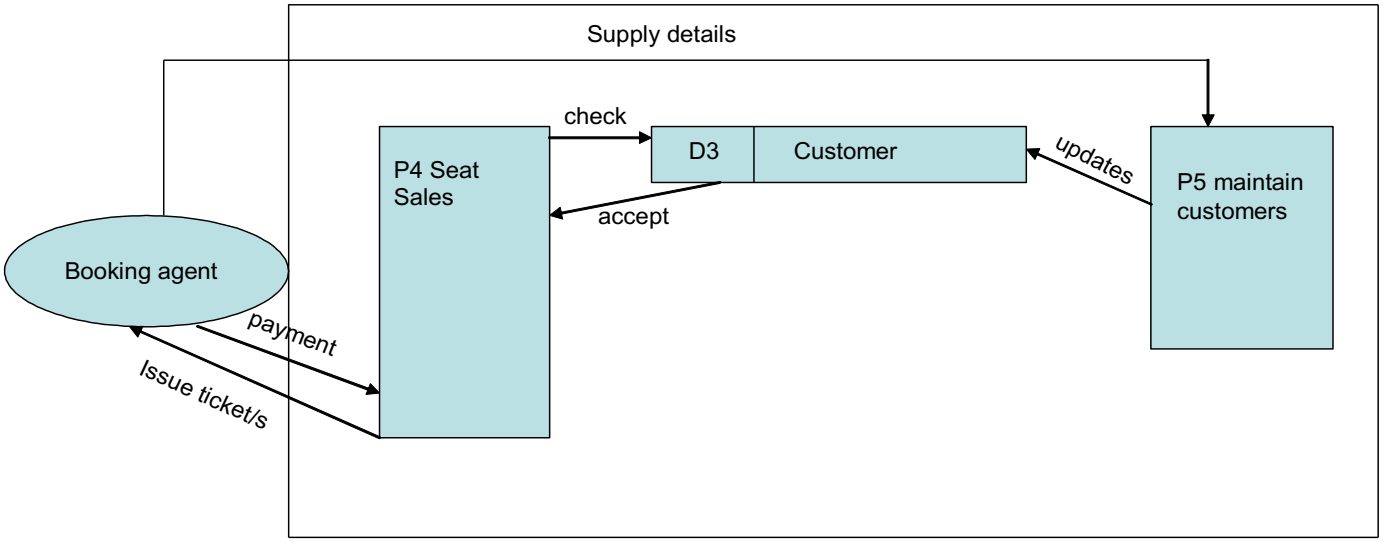
Level 0 DFD Make booking



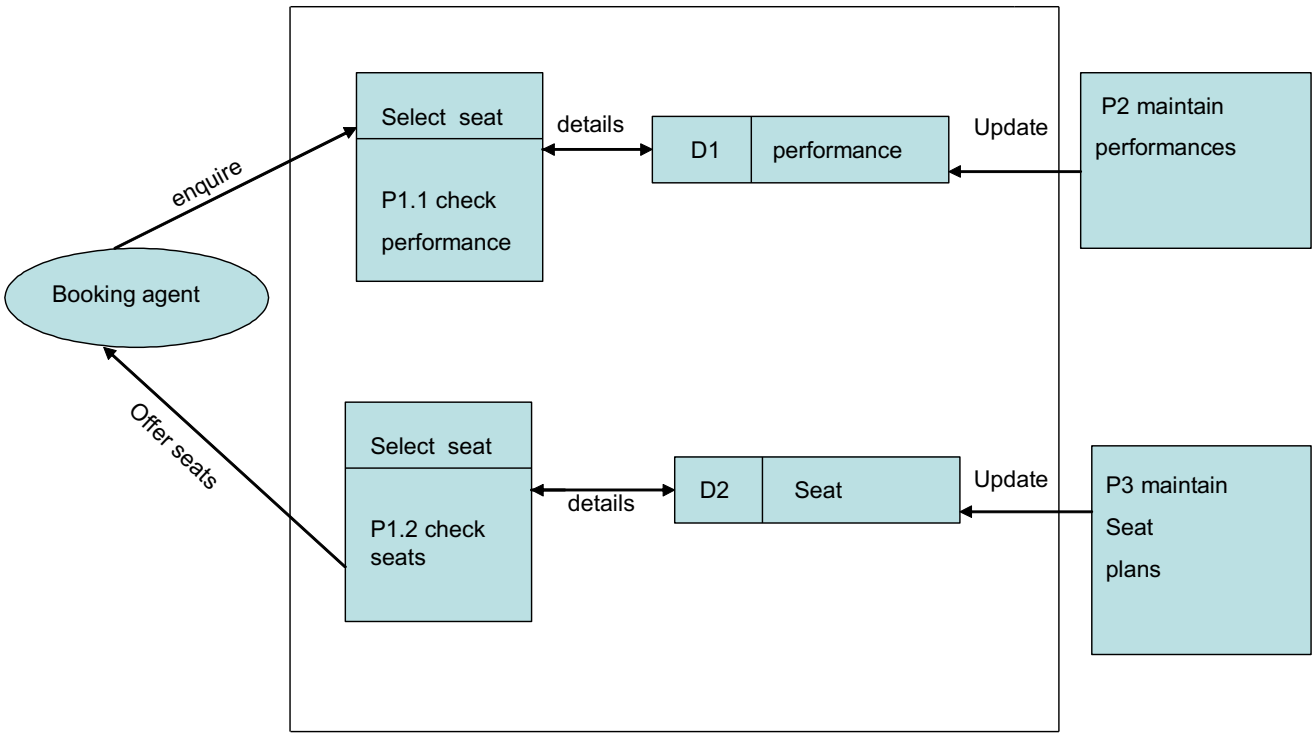
Level 1 DFD Select Seat



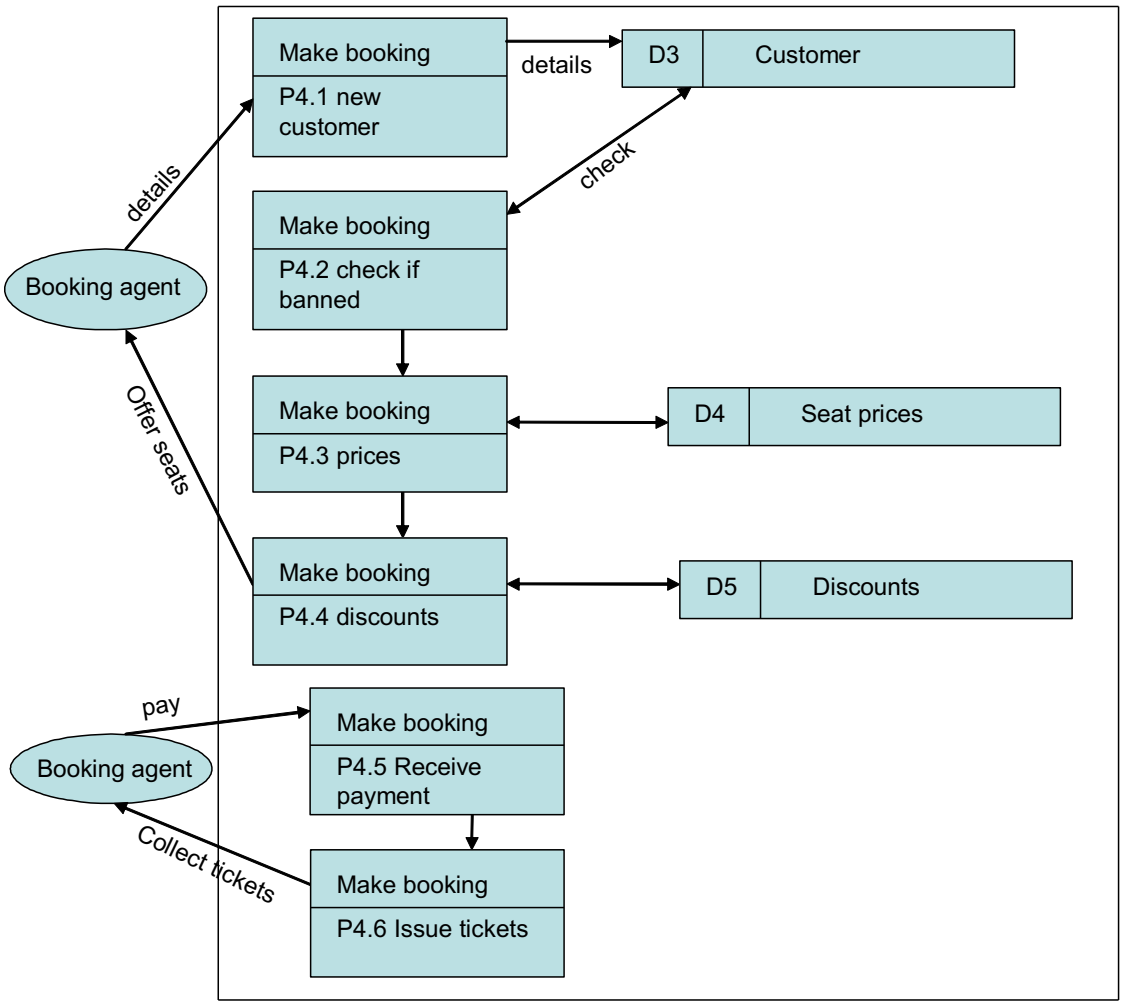
Level 1 DFD make booking



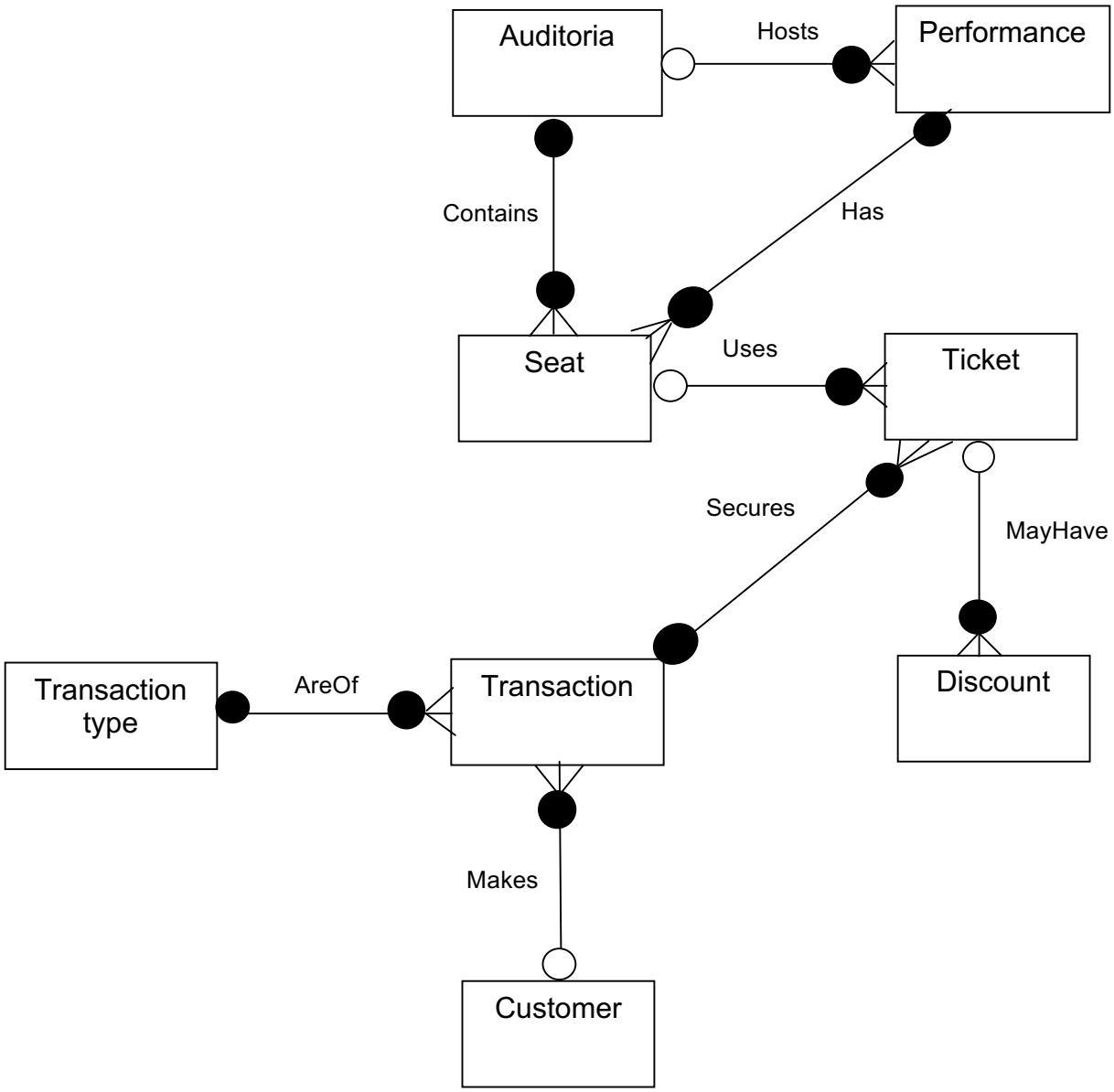
Level 2 DFD Select Seat



Level 2 DFD Make Booking



Activity 6 **An entity model for the theatre system**



Entity Description	
Identifier:	E1
Name:	Customer
Description:	This entity contains information about the customers (patrons) of the theatre
Attributes:	ID – Primary key Title – mandatory Firstname – mandatory Lastname – mandatory Email ** Status – mandatory Address ** Mailing – mandatory
Constraints:	• The Status attribute can be set to banned or not banned (1 or 0) • The Mailing attribute (which relates to whether a customer wants to be included on the mailing list) can be set to On or Off (1 or 0) • **Either email or address must be filled in
Assumptions:	• It is assumed all people who buy tickets (patrons) will be entered into the customer table • They are asked if they would like to go on the mailing list when they are entered onto the system at the same time as a document explaining their rights under the Data Protection Act is given to them • Agencies who sell tickets are also entered as a customer in this table • A customer may choose to be contacted via email or post
Comments:	None
Requirements:	• Used by the Make Booking function • Used by the Cancel Booking function • Used by the Return Tickets function • Used by the Maintain (Add/Modify/Delete) Customer functionality • Used by the Mailing list functionality
Entity Description	
Identifier:	E2
Name:	Performance
Description:	This entity contains information about the performances that are given at the theatre
Attributes:	ID – Primary key StartDate – mandatory EndDate – mandatory PerformanceTitle – mandatory AuditoriaID – mandatory Actors Description Category TotalSeats – mandatory SeatsSold – mandatory
Constraints:	• AuditoriaID is a foreign key which must exist in the Auditoria table • Seats sold is initially set to 0
Assumptions:	• All performances are entered into this table
Comments:	This may be used at a later date to include the Proposed Outreach scheme (AKTOR!)
Requirements:	• Used by the Make Booking function • Used by the Find Seat Availability function • Used by the Maintain (Add/Modify/Delete) Performance functionality
Entity Description	

Identifier:	E3
Name:	Seat
Description:	This entity contains information about the seats available in the auditoria
Attributes:	SeatNumber – Primary key Row – mandatory Section – mandatory PerformanceID – mandatory AudtoriaID – mandatory DisabilityAccess
Constraints:	• PerformanceID is a foreign key and must have a matching record in the Performance table. • AudtoriaID is a foreign key and must have a matching record in the Auditoria table.
Assumptions:	• DisabilityAccess will contain a setting to indicate if this seat has special disability access and what type
Comments:	None
Requirements:	• Used by the Make Booking function • Used by the Find Seat Availability function • Used by the Maintain (Add/Modify/Delete) Seat functionality
Entity Description	
Identifier:	E4
Name:	Discount
Description:	This entity contains information about the discounts that are available on the tickets that are sold for the performances at the theatre
Attributes:	ID – Primary key DiscountType – mandatory Amount – mandatory
Constraints:	None
Assumptions:	None
Comments:	None
Requirements:	• Used by the Make Booking function • Used by the Cancel Booking function • Used by the Return Tickets function • Used by the Maintain (Add/Modify/Delete) Discount functionality
Entity Description	
Identifier:	E5
Name:	Transaction
Description:	This entity contains information about ticket transactions, this includes the sales, returns and cancellations
Attributes:	TransactionID – Primary key Date – mandatory Time – mandatory CustomerID – mandatory TTID – mandatory Amount – mandatory Deduction
Constraints:	• CustomerID is a foreign key and must have a matching record in the Customer table. • TTID (transaction type ID) is a foreign key and must have a matching record in the Transaction Type table. • Deduction is the amount being deducted from the total amount, it represents either a Discount or Cancellation charge
Assumptions:	• Deduction defaults to 0 if no entry is made
Comments:	None
Requirements:	• Used by the Make Booking function • Used by the Cancel Booking function • Used by the Return Tickets function

Entity Description	
Identifier:	E6
Name:	Transaction type
Description:	This entity contains the transaction type identifiers – it is used by the transaction entity
Attributes:	ID – Primary key TransType – mandatory
Constraints:	• TransType represents the type of transaction, it can be either sale, return or cancellation
Assumptions:	None
Comments:	There are currently only 3 types of transaction as described in constraints
Requirements:	• Used by any function that accesses the Transaction table
Entity Description	
Identifier:	E7
Name:	Ticket
Description:	This entity contains information about the tickets
Attributes:	TicketID – Primary key SeatNumber – mandatory DrinksCode – mandatory DiscountID TransactionID – mandatory TicketDate – mandatory TicketTime – mandatory
Constraints:	• SeatNumber is a foreign key and must have a matching record in the Seat table. • DiscountsID is a foreign key and may have a matching record in the Discounts table if it is a value other than 0 • TransactionID is a foreign key and must have a matching record in the Transaction table.
Assumptions:	• DiscountID can be 0
Comments:	None
Requirements:	• Used by the Make Booking function • Used by the Cancel Booking function • Used by the Return Tickets function
Entity Description	
Identifier:	E8
Name:	Auditoria
Description:	This entity contains information about the Auditoria
Attributes:	ID – Primary key Name – mandatory
Constraints:	None
Assumptions:	None
Comments:	There are currently only 2 auditoria available, these are The Stearne and the Pomegranate
Requirements:	• Used by any function that uses the Seat table • Used by the Maintain (Add/Modify/Delete) Auditoria functionality

Attribute Description	
Identifier:	A1
Name:	ID
Description:	This is the Primary key in tables where it occurs
Data Type:	Long Integer
Data Values:	Numeric, integer
Constraints:	Mandatory
Owner:	Occurs in Customer, Performance, Discount, Transaction type and

	Auditoria
Attribute Description	
Identifier:	A2
Name:	Title
Description:	A persons mode of address, e.g. Mr, Mrs, Dr
Data Type:	String
Data Values:	Any acceptable form of address
Constraints:	Mandatory
Owner:	Customer
Attribute Description	
Identifier:	A3
Name:	Firstname
Description:	A persons first name or of an agency
Data Type:	String
Data Values:	Any acceptable mix of alphabetic optionally with hyphens or spaces
Constraints:	Mandatory
Owner:	Customer
Attribute Description	
Identifier:	A4
Name:	Lastname
Description:	A persons last name or the word “agency”
Data Type:	String
Data Values:	Any acceptable mix of alphabetic optionally with hyphens or spaces or apostrophe
Constraints:	Mandatory
Owner:	Customer
Attribute Description	
Identifier:	A5
Name:	Email
Description:	An email address for contact
Data Type:	String
Data Values:	Any valid form of email address
Constraints:	Either this or Address must be filled in
Owner:	Customer
Attribute Description	
Identifier:	A6
Name:	Status
Description:	This indicates whether a customer is banned or not
Data Type:	Boolean
Data Values:	Banned or not banned (1 or 0)
Constraints:	Mandatory
Owner:	Customer
Attribute Description	
Identifier:	A7
Name:	Address
Description:	This is a postal address for use if the customer wants to be contacted by post for the mailing list
Data Type:	String
Data Values:	Any valid form of postal address
Constraints:	Either this or Email must be filled in
Owner:	Customer
Attribute Description	
Identifier:	A8
Name:	Mailing – mandatory
Description:	A customer can opt in or out of the mailing list
Data Type:	Boolean
Data Values:	On or Off (1 or 0)

Constraints:	Mandatory
Owner:	Customer
Attribute Description	
Identifier:	A9
Name:	StartDate
Description:	The start date for the performance
Data Type:	Date
Data Values:	Valid UK date format (DD/MM/YYYY)
Constraints:	Mandatory
Owner:	Performance
Attribute Description	
Identifier:	A10
Name:	EndDate – mandatory
Description:	The end date for the performance
Data Type:	Date
Data Values:	Valid UK date format (DD/MM/YYYY)
Constraints:	Mandatory, must be equal to or later than the start date
Owner:	Performance
Attribute Description	
Identifier:	A11
Name:	PerformanceTitle
Description:	The title of the performance
Data Type:	String
Data Values:	Any mix of letters and numbers plus punctuation
Constraints:	Mandatory
Owner:	Performance
Attribute Description	
Identifier:	A12
Name:	AuditoriaID
Description:	The identifier for the Auditorium, this is a foreign key that link to the Auditoria table
Data Type:	Long Integer
Data Values:	Numeric, integer
Constraints:	Mandatory
Owner:	Performance
Attribute Description	
Identifier:	A13
Name:	Actors
Description:	The actors and/or performers appearing in the production
Data Type:	String
Data Values:	Any mix of letters and numbers plus punctuation
Constraints:	None
Owner:	Performance
Attribute Description	
Identifier:	A14
Name:	Description
Description:	A description of the production
Data Type:	String
Data Values:	Any mix of letters and numbers plus punctuation
Constraints:	None
Owner:	Performance
Attribute Description	
Identifier:	A15
Name:	Category
Description:	A one or two word categorisation of the production, e.g. Play, Pantomime, Musical recital
Data Type:	String

Data Values:	Any mix of letters and numbers plus punctuation
Constraints:	None
Owner:	Performance
Attribute Description	
Identifier:	A16
Name:	TotalSeats
Description:	The total number of seats available for the performance
Data Type:	Integer
Data Values:	Numeric, Integer
Constraints:	Cannot be greater than the total number of seats designated for an auditorium, mandatory
Owner:	Performance
Attribute Description	
Identifier:	A17
Name:	SeatsSold
Description:	The number of seats for a production that have been sold
Data Type:	Integer
Data Values:	Numeric, Integer
Constraints:	Cannot be greater than the total number of seats field, mandatory
Owner:	Performance
Attribute Description	
Identifier:	A18
Name:	SeatNumber
Description:	The number of the seat in the auditorium
Data Type:	String
Data Values:	Numeric, Integer
Constraints:	Primary key, Mandatory
Owner:	Seat
Attribute Description	
Identifier:	A19
Name:	Row
Description:	The row letter(s) in the auditorium
Data Type:	String
Data Values:	Two character, alphabetic only
Constraints:	Mandatory
Owner:	Seat
Attribute Description	
Identifier:	A20
Name:	Section
Description:	The section where the seating resides
Data Type:	Integer
Data Values:	Numeric, Integer
Constraints:	Mandatory
Owner:	Seat
Attribute Description	
Identifier:	A21
Name:	PerformanceID
Description:	A foreign key to the performance table
Data Type:	Long Integer
Data Values:	Numeric, Integer
Constraints:	Must exist in the performance table, mandatory
Owner:	Seat
Attribute Description	
Identifier:	A22
Name:	AudtoriaID
Description:	A foreign key to the auditoria table
Data Type:	Long Integer

Data Values:	Numeric, Integer
Constraints:	Must exist in the auditoria table, mandatory
Owner:	Seat
Attribute Description	
Identifier:	A23
Name:	DisabilityAccess
Description:	This indicates if a seat has disability provision
Data Type:	String
Data Values:	Description of the disability access for this seat - alphabetic
Constraints:	None
Owner:	Seat
Attribute Description	
Identifier:	A24
Name:	DiscountType
Description:	A description of a type of discount
Data Type:	String
Data Values:	Alphabetic, e.g. Child, Concession, Volunteer
Constraints:	Mandatory
Owner:	Discount
Attribute Description	
Identifier:	A25
Name:	Amount
Description:	An amount to be subtracted from the normal ticket cost based on the type of discount
Data Type:	Currency
Data Values:	An amount in pounds sterling, 2 decimal places
Constraints:	Mandatory
Owner:	Discount
Attribute Description	
Identifier:	A26
Name:	TransactionID
Description:	The ID for the transaction
Data Type:	Long Integer
Data Values:	Numeric, integer
Constraints:	Primary key, mandatory
Owner:	Transaction
Attribute Description	
Identifier:	A27
Name:	Date
Description:	The date of the transaction
Data Type:	Date
Data Values:	Valid UK date format (DD/MM/YYYY)
Constraints:	Mandatory
Owner:	Transaction
Attribute Description	
Identifier:	A28
Name:	Time
Description:	The time of the transaction
Data Type:	Time
Data Values:	HH:MM:SS (hours, minutes, seconds)
Constraints:	Mandatory
Owner:	Transaction
Attribute Description	
Identifier:	A29
Name:	CustomerID
Description:	A foreign key to the customer table
Data Type:	Long integer

Data Values:	Numeric, integer
Constraints:	Mandatory, must exist in the customer table
Owner:	Transaction
Attribute Description	
Identifier:	A30
Name:	TTID
Description:	Transaction type ID
Data Type:	Long integer
Data Values:	Numeric, integer
Constraints:	Mandatory, must exist in the transaction type table
Owner:	Transaction
Attribute Description	
Identifier:	A31
Name:	Amount
Description:	The cost of the transaction
Data Type:	Currency
Data Values:	An amount in pounds sterling, 2 decimal places
Constraints:	Mandatory
Owner:	Transaction
Attribute Description	
Identifier:	A32
Name:	Deduction
Description:	The amount to be deducted from the amount, it can be derived from discounts for a sale or cancellation charges for a cancellation (refund)
Data Type:	Currency
Data Values:	An amount in pounds sterling, 2 decimal places
Constraints:	Cannot be greater than the Amount
Owner:	Transaction
Attribute Description	
Identifier:	A33
Name:	TransType
Description:	TransType represents the type of transaction
Data Type:	String
Data Values:	It can be either sale, return or cancellation
Constraints:	mandatory
Owner:	Transaction Type
Attribute Description	
Identifier:	A34
Name:	TicketID
Description:	The ID for the ticket
Data Type:	Long Integer
Data Values:	Numeric, Integer
Constraints:	Primary key
Owner:	Ticket
Attribute Description	
Identifier:	A35
Name:	SeatNumber
Description:	The number of the seat the ticket applies to
Data Type:	String
Data Values:	This is a foreign key that must exist in the seat table, numeric, integer
Constraints:	Mandatory
Owner:	Ticket
Attribute Description	
Identifier:	A36
Name:	DrinksCode
Description:	A code used when a customer pre-orders interval drinks
Data Type:	Long integer

Data Values:	Numeric, integer
Constraints:	Mandatory
Owner:	Ticket
Attribute Description	
Identifier:	A37
Name:	DiscountID
Description:	An identifier to the discount that applies to the ticket (if any)
Data Type:	Integer
Data Values:	This is a foreign key to the discount table, numeric, integer
Constraints:	May be 0
Owner:	Ticket
Attribute Description	
Identifier:	A38
Name:	TransactionID
Description:	A transaction ID to link the ticket to the transaction
Data Type:	Long integer
Data Values:	This is a foreign key that must exist in the transaction table, numeric, integer
Constraints:	Mandatory
Owner:	Ticket
Attribute Description	
Identifier:	A39
Name:	TicketDate
Description:	The date of the performance
Data Type:	Date
Data Values:	Valid UK date format (DD/MM/YYYY)
Constraints:	Mandatory
Owner:	Ticket
Attribute Description	
Identifier:	A40
Name:	TicketTime
Description:	The time of the performance
Data Type:	Time
Data Values:	HH:MM:SS (hours, minutes, seconds)
Constraints:	Mandatory
Owner:	Ticket
Attribute Description	
Identifier:	A41
Name:	Name
Description:	The name of the auditorium
Data Type:	String
Data Values:	Alphabetic
Constraints:	Mandatory, currently only The Stearne or The Pomegranate
Owner:	Auditoria

Relationship Description	
Identifier:	R1
Name:	Hosts
Entities Linked:	Auditoria and Performance
Description:	Each auditorium can host a performance (in fact it will host many performances over the course of the year), a performance needs an Auditorium but an auditorium can exist without any performances.
Definition:	A auditorium may host 0, 1 or many performances A performance is hosted by 1 auditorium
Relationship Description	
Identifier:	R2

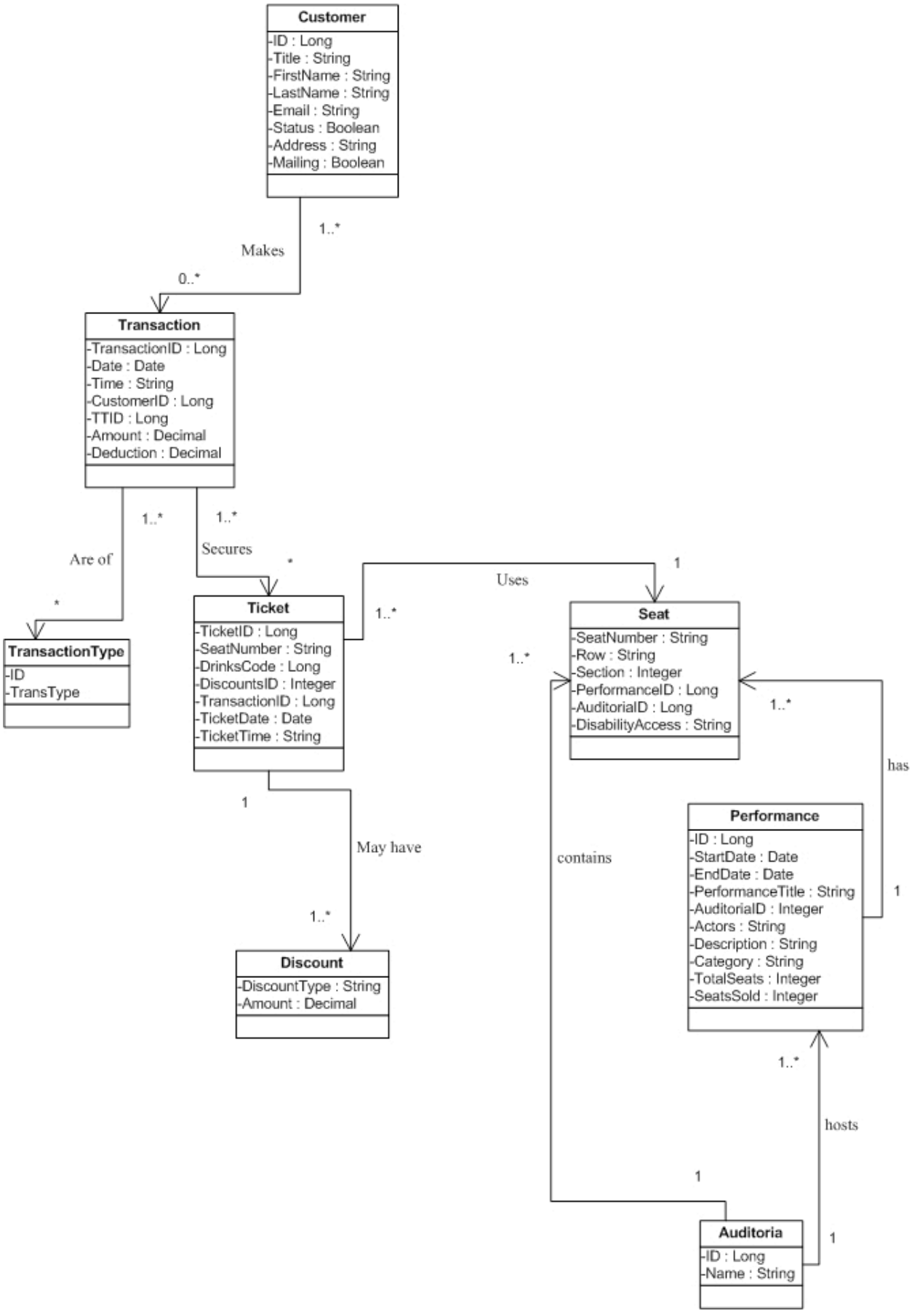
Name:	Contains
Entities Linked:	Auditoria and Seat
Description:	Each auditorium contains a number of seats
Definition:	A auditorium may contain 1 or more seats A seat is contained in 1 auditorium
Relationship Description	
Identifier:	R3
Name:	Has
Entities Linked:	Performance and Seat
Description:	Each performance has seats
Definition:	A performance may have 1 or more seats A seat is had by 1 performance
Relationship Description	
Identifier:	R4
Name:	Uses
Entities Linked:	Seat and Ticket
Description:	Each seat uses one or more tickets (note that a parent and child can use the same seat but 2 tickets are issued)
Definition:	A seat may use 0, 1 or many tickets A ticket is used by 1 seat
Relationship Description	
Identifier:	R5
Name:	Secures
Entities Linked:	Transaction and Ticket
Description:	Each transaction secures a ticket
Definition:	A transaction may secure 1 or more tickets A ticket is secured by 1 transaction
Relationship Description	
Identifier:	R6
Name:	MayHave
Entities Linked:	Ticket and Discount
Description:	Each ticket may have one or more discounts applied to its price
Definition:	A ticket may have 0, 1 or many discounts A discount may be applied to a ticket
Relationship Description	
Identifier:	R7
Name:	AreOf
Entities Linked:	Transaction and Transaction Type
Description:	Each transaction is of a certain transaction type (sale, return, cancellation)
Definition:	A transaction is of 1 transaction type A transaction type applies to 1 or many transactions
Relationship Description	
Identifier:	R8
Name:	Makes
Entities Linked:	Customer and Transaction
Description:	Each customer can make 0 or more transactions
Definition:	A customer may make 0, 1 or many transactions A transaction is made by 1 customer

Activity 7 **A class model for the system**

Performance		
Date		
Author	Drd	
Superclasses	None	
Subclasses	None	
Description	Know all performances and details for each date.	
Responsibilty	Collabration	Association
Performance date Performance time Performance auditorium Performance title	Seats Auditioria	1 : many 1 : many

Customer		
Date		
Author	Drd	
Superclasses	None	
Subclasses	None	
Description	Stores and handles all data for customers.	
Responsibilty	Collabration	Association
Know customer address Know customer names Know customer bookings Know banned customers Know customer discounts Change customer details Enter new customers	Transaction	1 : many

Seat		
Date		
Author	Drd	
Superclasses	None	
Subclasses	None	
Description	Know all seats in each auditorium and which have been booked.	
Responsibilty	Collabration	Association
Know if seat booked Know if for disabled	Performance Seat booking	1 : 1


Customer
-ID : Long
-Title : String
-FirstName : String
-LastName : String
-Email : String
-Status : Boolean
-Address : String
-Mailing : Boolean
1..*
Makes
0..*
Transaction
-TransactionID : Long
-Date : Date
-Time : String
-CustomerID : Long
-TTID : Long
-Amount : Decimal
-Deduction : Decimal
1..*
Are of
*
1..*
Secures
*
TransactionType
-ID
-TransType
Ticket
-TicketID : Long
-SeatNumber : String
-DrinksCode : Long
-DiscountsID : Integer
-TransactionID : Long
-TicketDate : Date
-TicketTime : String
1..*
Uses
1
1..*
Seat
-SeatNumber : String
-Row : String
-Section : Integer
-PerformanceID : Long
-AuditorialD : Long
-DisabilityAccess : String
1..*
has
1
1
May have
1..*
Discount
-DiscountType : String
-Amount : Decimal
contains
Performance
-ID : Long
-StartDate : Date
-EndDate : Date
-PerformanceTitle : String
-AuditorialD : Integer
-Actors : String
-Description : String
-Category : String
-TotalSeats : Integer
-SeatsSold : Integer
1..*
hosts
1
1
Auditoria
-ID : Long
-Name : String

Activity 8 **Sequence & activity diagrams associated with "make booking" use case**

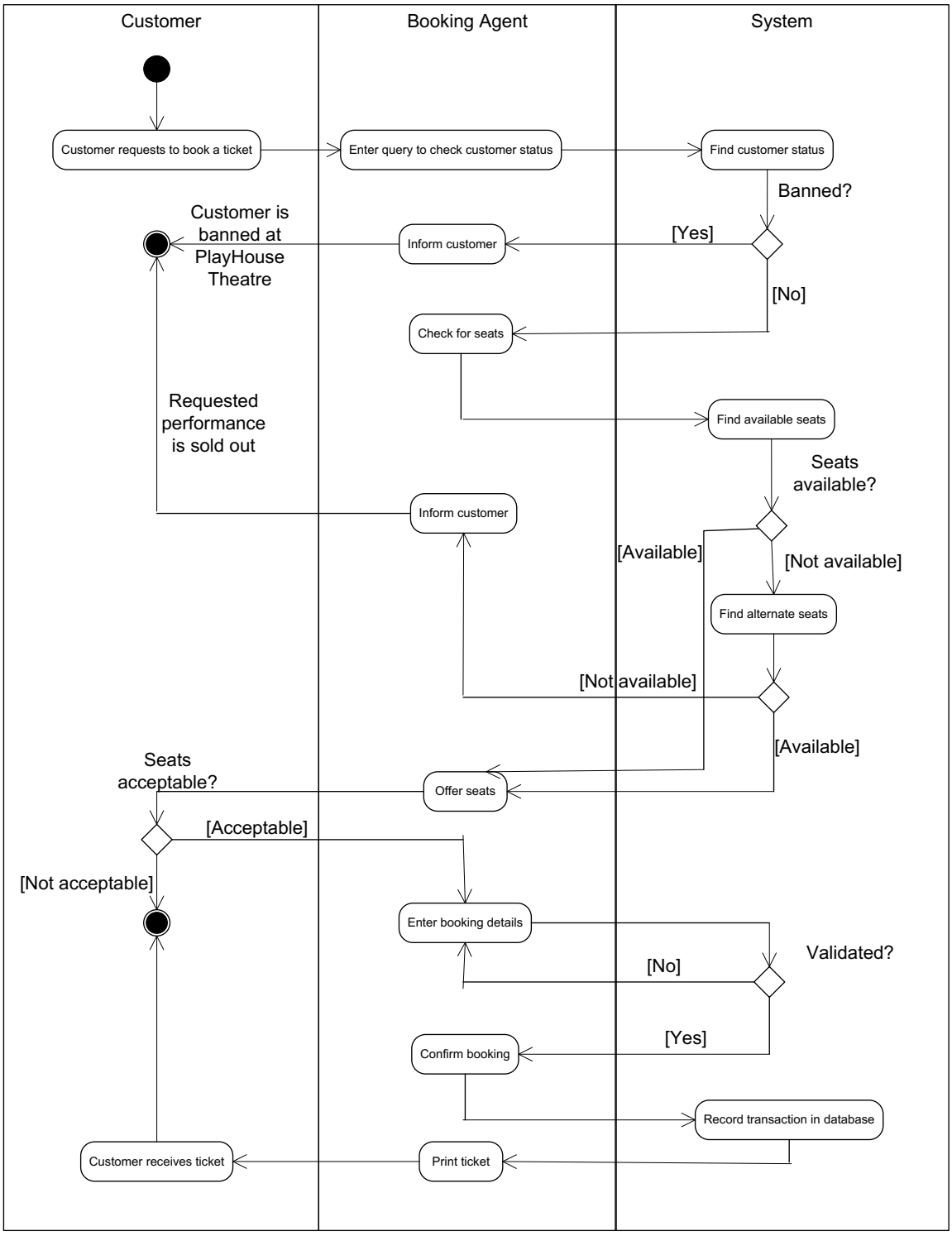


Figure 3: Activity Diagram - Make Booking

Booking agent activities:

1. Enter query to check customer status
2. Inform customer if his status is of a banned customer
3. Check for seats
4. If performance is sold out, inform customer
5. Offer the available seats to customer and ask him to select
6. Enter booking details (seat number, ticket type, quantity, drinks code & completes payment)
7. Confirm booking
8. Print Ticket

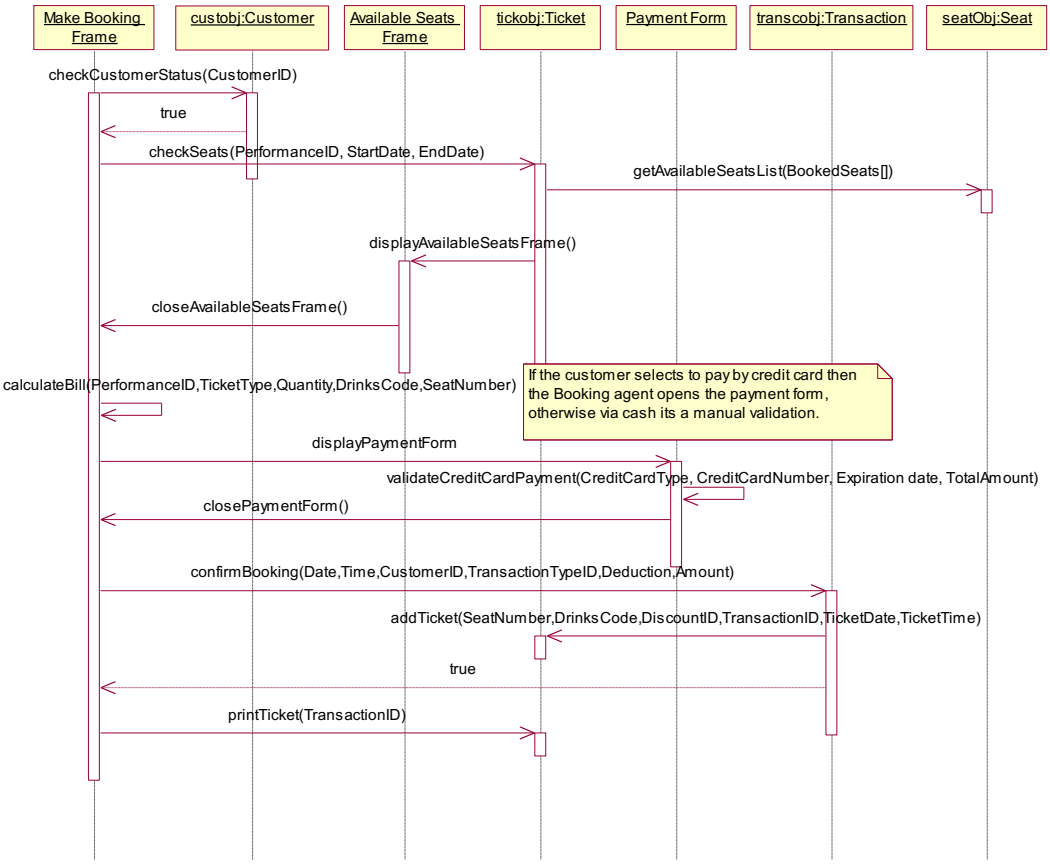


Figure 4: Sequence Diagram - Make Booking

Classes:

The following classes are involved in the make booking sequence diagram.

1. Customer
2. Ticket
3. Transaction
4. Seat

Functions:

The following functions are involved in the make booking sequence diagram.

1. Boolean checkCustomerStatus(CustomerID)
2. checkSeats(PerformanceID, StartDate, EndDate)
3. String[] getAvailableSeatsList(BookedSeats[])
4. Double calculateBill(PerformanceID,TicketType,Quantity,DrinksCode,SeatNumber)
5. Boolean validateCreditCardPayment(CreditCardType, CreditCardNumber, Expiration date, TotalAmount)
6. Boolean confirmBooking(Date,Time,CustomerID,TransactionTypeID,Deduction,Amount)
7. Boolean addTicket(SeatNumber,DrinksCode,DiscountID,TransactionID,TicketDate,Ticket Time)
8. Void printTicket(TransactionID)

Functions description:

On the request of the customer, the booking agent starts with the booking process. He opens the Make Booking Frame and asks for Customer ID in order to check his status.

checkCustomerStatus(CustomerID) function handles it. The Customer ID is passed as parameter. If the 'Status' of the customer has value "Banned" then the function would return false and the booking process would be cancelled. Otherwise the booking agent would continue with the booking process.

Assumption: If CustomerID is not found in database and it's a new customer, then the booking agent will first add the customer and then return to 'Make booking' feature.

Next the Booking agent checks for the available seats for the performance requested by customer.

The checkSeats(PerformanceID, StartDate, EndDate) functions searches for the available seats for all the dates the performance is going on. It searches for all the booked seats in ticket database and save the searched result in a string array. This array actually contains the attribute values of seat numbers.

This BookedSeats array is passed on by getAvailableSeatsList(BookedSeats[]) to the seats database. This function produces list of the available seats by comparing the BookedSeats with the seat numbers in the seat database.

The search results are shown on the 'Available seats frame'. Booking agent then asks the customer to choose from the available seats. Once customer is done selecting, booking agent returns to 'Make Booking Frame'.

The booking agent then enters the seat number, selects ticket type and quantity and enters Drinks Code if any drinks are requested by the customer.

Next the booking agent calculates bill.

The calculateBill(PerformanceID,TicketType,Quantity,DrinksCode,SeatNumber) function calculates the bill based on the seat selected and the performance and adjusts if any discounts are applicable based on the ticket type. The performance ID and seat number help in retrieving the price from the database, while the ticket type is used to get discount amount. The total amount is calculated.

The booking agent asks customer if he is paying by cash or credit card. If its cash, the process is completely manually. If by credit card, then the Booking agent opens the payment form and enters/selects the payment details like Credit card type, Credit card number, expiration date and amount. The validateCreditCardPayment(CreditCardType, CreditCardNumber, Expiration date, TotalAmount) function is initiated to validate the credit card and complete payment process and returns to ‘Make booking’ frame.

Once the payment is done, the Booking agent confirms booking.
The confirmBooking(Date,Time,CustomerID,TransactionTypeID,Deduction,Amount) and addTicket(SeatNumber,DrinksCode,DiscountID,TransactionID,TicketDate,TicketTime) are initiated then. The Transaction and Ticket databases are updated to record the booking details. In case the customer buys more than 1 ticket, then using programming we separate the ticket rows by running a while loop on the ticket’s dataset to separate each ticket record until the dataset is non-empty. This is done under confirmBooking(Date,Time,CustomerID,TransactionTypeID,Deduction,Amount) function, after updating transaction’s database, this function gives call to addTicket function to record all the sold tickets in the ticket database.

Once booking confirmation process is through, the booking agent prints the ticket. The printTicket(TransactionID)function is initiated. Using the Transaction ID, the ticket details are retrieved from the database and the ticket(s) are printed and handed over to the customer.