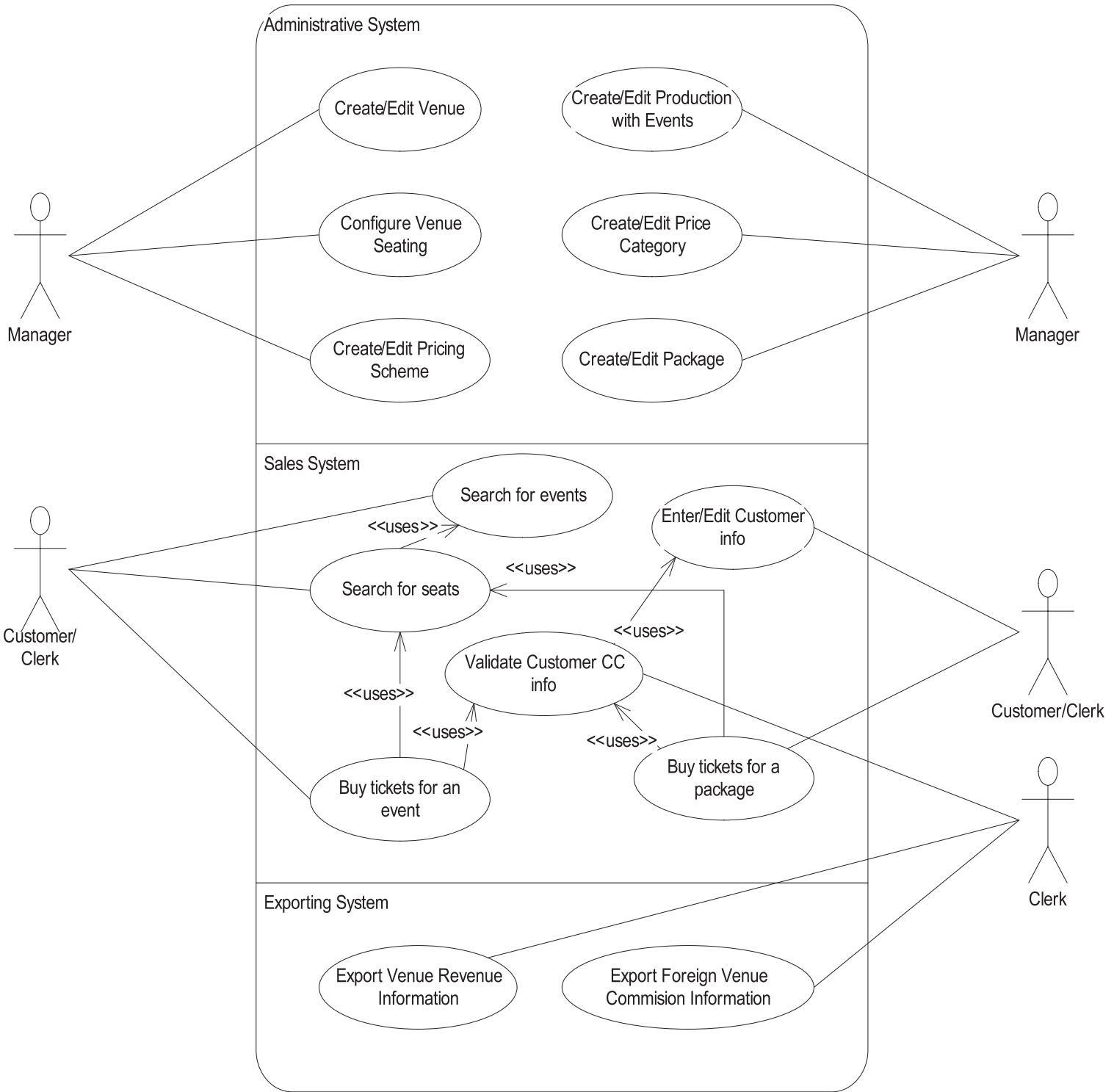


# Two Tickets Plus

## Appendix B.



## Two Tickets Plus. Use Case Descriptions

Use Case Name:	<b>Create/Edit Venue</b>	
Scenario:	<b>Create a Venue Edit an existing Venue</b>	
Triggering Event:	<b>Original set up of database with new venue A foreign venue needs to be listed in the database</b>	
Brief Description:	<b>First check to see if venue exists. Add or edit venue information. This is a simple single table update of the database.</b>	
Actors:	<b>Manager or clerk</b>	
Stakeholders:		
Preconditions:	<b>None</b>	
Postconditions:	<b>Venue exists in database</b>	
Flow of Events:	Actor	System
	<b>1. Search on venue name</b>  <b>2. Enter venue information</b>	<b>1. If venue exists, display venue information.</b> <b>1a. If no venue exists, display blank venue screen.</b> <b>2. Create or update venue information</b>
Exception Conditions:		

Use Case Name:	<b>Configure Venue Seating</b>	
Scenario:	Set up venue seating Edit existing venue seating	
Triggering Event:	Initial set up of venue Correction or change to existing seating	
Brief Description:	This use case builds all sections, rows and seats. For each section, it adds row information. For each row it adds seat information and pricing category foreign key. When editing existing information, it first populates the screens and then updates the database. The design of the forms, whether one form per object, or several records in a matrix is left up to the programmer.	
Actors:	Manager	
Stakeholders:		
Preconditions:	Venue must exist Pricing Categories should exist (although could be added later)	
Postconditions:	Section objects created Row objects created Seat objects created Rows associated with pricing category	
Flow of Events:	Actor	System
	<ol style="list-style-type: none"> <li>1. Enter venue name</li> <li>2. For each section, enter section information.</li> <li>3. Enter beginning row number and ending row number (numbers or letters, as per venue numbering scheme).</li> <li>4. For each row, enter seat numbers (scheme according to venue). Enter pricing category.</li> <li>5. Finish</li> </ol>	<ol style="list-style-type: none"> <li>1. Display venue information. Display form to enter section ID and description. If existing, show first, else show blank form.</li> <li>2. Update Section information. Display form to enter row information for that section. If existing, show existing data, else blank.</li> <li>3. Display form to enter seat information. If existing, show seat information for active row. If new, show blank form.</li> <li>4. Update database with new information or existing info updated. Update section with number of rows, and rows with number of seats. Create new section. Create Row objects. Create Seat objects. Associate all foreign keys.</li> </ol>
Exception Conditions:	Venue not found. Duplicate Section number. Price Category not found. Leave blank, but note as error condition.	

Use Case Name:	<b>Create/Edit seat pricing category</b>	
Scenario:	<b>Set up initial pricing categories Update existing pricing categories</b>	
Triggering Event:	<b>Initial set up of theater Manager decides to change pricing category layout</b>	
Brief Description:	<b>This use case sets up the pricing categories so that (1) they can be associated with rows, and (2) pricing schemes can be developed. It is a simple single table update.</b>	
Actors:	<b>Manager</b>	
Stakeholders:		
Preconditions:	<b>None</b>	
Postconditions:	<b>Price Category exists</b>	
Flow of Events:	Actor	System
	<b>1. Display all pricing categories</b>  <b>2. Enter price category ID and description</b>	<b>1. Find and display, else display blank screen</b>  <b>2. Create new object, or update existing</b>
Exception Conditions:	<b>2. Entering duplicate ID will cause exception.</b>	

Use Case Name:	<b>Create/Edit Pricing Scheme</b>	
Scenario:	<b>Set up initial pricing scheme</b> <b>Modify pricing scheme</b>	
Triggering Event:	<b>Initial set up of venue</b> <b>New pricing scheme is required</b> <b>Prices are changed</b>	
Brief Description:	<b>This use case adds/updates pricing information for a pricing scheme. All pricing category records must be updated.</b>	
Actors:	<b>Manager</b>	
Stakeholders:		
Preconditions:	<b>Price categories exist</b>	
Postconditions:	<b>Pricing Scheme object exists</b> <b>SchemeCategoryPrice objects exist</b>	
Flow of Events:	Actor	System
	<b>1. Display all pricing schemes</b>  <b>2. Enter new pricing scheme ID and description</b>  <b>3. For each pricing category (within this scheme), enter the ticket price, package price, foreign commission.</b>  <b>4. Finish</b>	<b>1. Find and display, else display blank screen</b>  <b>2. Create new object, or update existing. Update database.</b> <b>2a. Display pricing category information (one at a time, or as a table.)</b>  <b>3. Update SchemeCatPrice in database.</b>
Exception Conditions:	<b>2. No price categories exist</b> <b>3. Not all categories have pricing information causes exception.</b>	

Use Case Name:	<b>Create/Edit a Production</b>	
Scenario:	<b>Schedule a new production Update an existing production</b>	
Triggering Event:	<b>Manager schedules a new production</b>	
Brief Description:	<b>Events are always created within a production. First create a production, then add the necessary events. System must check for date/time conflicts. System must create all the Event seats (tickets) available for sale. Each event seat should contain section, row, and seat identifiers.</b>	
Actors:	<b>Manager</b>	
Stakeholders:		
Preconditions:	<b>Venue must exist Pricing Scheme should exist Venue configuration (sections, rows, seats) must exist</b>	
Postconditions:	<b>Production will exist Events will exist Event Seats will exist Events will have pricing scheme</b>	
Flow of Events:	Actor	System
	<b>1. Enter Production name</b>  <b>2. Enter production information (new or updates)</b>  <b>3. For each event, enter event date and time and pricing scheme</b>  <b>4. Finish</b>	<b>1. Find Production. Display info or blank screen.</b>  <b>2. Update production info. Display Events form.</b>  <b>3. Add/update event information.</b> <b>3a. Check date/time for conflicts.</b> <b>3b. Create EventSeats for event</b>  <b>4. Update database</b>
Exception Conditions:	<b>3. Event conflict raise exception condition</b>	

Use Case Name:	<b>Create/Edit Package</b>	
Scenario:	<b>Add SalesPackage Event Series</b>	
Triggering Event:	<b>Manager wants to create a package of events for sale</b>	
Brief Description:	<b>Create a sales package. Associate the desired productions with the package. Find minimum package price and maximum package price. Note: This does NOT handle sporting event series, only production series.</b>	
Actors:	<b>Clerk</b>	
Stakeholders:		
Preconditions:	<b>Productions must exist Events must exist Pricing scheme must exist Pricing category must exist</b>	
Postconditions:	<b>Package must exist EventsInPackage must exist</b>	
Flow of Events:	Actor	System
	<b>1. Enter new sales package name</b>  <b>2. Enter package information</b>  <b>3. From list of productions, choose a production to include. Repeat for all productions to include</b>  <b>4. Finish</b>	<b>1. If new display blank screen. If exists, display package information.</b>  <b>2. Create/update package information. Display productions information</b>  <b>3. Add production to package (create EventsInPackage). Find minimum and maximum seat price (low pricing scheme and low price category).</b> <b>3a. Accum min and max total amount.</b>
Exception Conditions:	<b>Preconditions not present, raise exception conditions</b>	

Use Case Name:	<b>Create/Edit Customer Info</b>	
Scenario:	<b>Customer needs to enter information</b>	
Triggering Event:	<b>Usually part of a purchase</b>	
Brief Description:	<b>Customer enters his/her personal information and the system updates the database.</b>	
Actors:	<b>Customer</b>	
Stakeholders:		
Preconditions:	<b>None</b>	
Postconditions:	<b>Customer record exists</b>	
Flow of Events:	Actor	System
	<b>1. Enter Customer name</b>  <b>2. Select correct name, or (none)</b>  <b>3. Enter corrections or new customer data.</b>  <b>Finish</b>	<b>1. Display list of existing customers with same name.</b>  <b>2. If selected, display complete customer info (not CC) info. Else display blank screen.</b>  <b>3. Create object and update database.</b>
Exception Conditions:	<b>NOTE: security is not addressed</b>	

Use Case Name:	<b>Validate Customer CC info</b>	
Scenario:	<b>As part of a sale validate CC info As a separate use case, clerk can verify CC info</b>	
Triggering Event:	<b>Sale requires CC validation</b>	
Brief Description:	<b>Collect CC information. Connect to validation site and return validation code.</b>	
Actors:	<b>Customer, Clerk</b>	
Stakeholders:	<b>Customer</b>	
Preconditions:	<b>Customer must exist</b>	
Postconditions:		
Flow of Events:	Actor	System
	<b>1. (Customer or clerk) Enters CC information and amount.</b>  <b>2. Finish</b>	<b>1. Convert to secure connection</b> <b>1a. Retrieve data.</b> <b>1b. Connect to validation site.</b> <b>1c. Return validation number or error message. Display message.</b>
Exception Conditions:	<b>Cannot connect to validation site.</b>	

Use Case Name:	<b>Search for events</b>	
Scenario:	<b>Customer is looking for events for a particular production As part of Serch for Available Seats, needs to find a production</b>	
Triggering Event:	<b>Customer wants to find available seats</b>	
Brief Description:		
Actors:	<b>Customer</b>	
Stakeholders:		
Preconditions:	<b>Production must exist Event must exist</b>	
Postconditions:	<b>None</b>	
Flow of Events:	Actor	System
	<b>1. Enter Search information (Venue Name and Production Name) (or part of name)</b>  <b>2. Finish</b> <b>2a. OR select an event to see Available Seats.</b>	<b>1. Find correct Venue. Find Productions available at that venue. Display full venue name, and full production information (start date, end date, etc.)</b> <b>1a. Display list of events available (date, time)</b>  <b>2. End</b>
Exception Conditions:	<b>Data not found.</b>	

Use Case Name:	<b>Search for available Seats</b>	
Scenario:	<b>Customer is browsing for available seats As part of purchase, customer/clerk searches for seats</b>	
Triggering Event:	<b>Customer wants to find available seats</b>	
Brief Description:		
Actors:	<b>Customer</b>	
Stakeholders:		
Preconditions:	<b>Event must exist EventSeat must exist</b>	
Postconditions:	<b>None</b>	
Flow of Events:	Actor	System
	<b>1. Enter Event information (Production, date, time) 1a. OR invoke Search for Events use case  2. Enter search criteria (price, price category, section, number of seats – any combination)</b>	<b>1. Display Event information  2. Display available seats according to criteria (may be textual or graphical)</b>
Exception Conditions:	<b>Data not found based on selection criteria</b>	

Use Case Name:	<b>Buy tickets for an event</b>	
Scenario:	<b>Purchase contiguous tickets for a single event Purchase all tickets in a box?</b>	
Triggering Event:	<b>Customer wants to buy tickets to an event.</b>	
Brief Description:	<b>Customer selects an event. Then system display available seats. Customer selects seats, enters customer information, enters CC info. System updates database and prints tickets.</b>	
Actors:	<b>Customer</b>	
Stakeholders:		
Preconditions:	<b>Production must exist Event must exist Event Seat must exist Venue/Seller must exist</b>	
Postconditions:	<b>Sale exists (and appropriate subclass exists) PaymentTransaction exists Customer Exists (optional) Ticket(s) must exist EventSeat updated</b>	
Flow of Events:	<b>Actor</b>	<b>System</b>
	<b>1. Customer selects event – Enter event info – production, date, time</b>  <b>2. Invoke “Search for available seats” as much as required.</b>  <b>3. Select desired seats.</b>  <b>4. More seats, return to 2.</b> <b>4a. No more seats.</b>  <b>5. Enter customer information</b>  <b>6. Enter CC information.</b>  <b>7. Finalize (Enter delivery request).</b>	<b>1. Display event information</b>  <b>2. Execute “Search for available seats.”</b>  <b>3. Create sale (possible in the form of a shopping cart). Create sale subclass. Create tickets and connect to sale. Update status of EventSeat.</b> <b>3a. Display “more seats” option</b>  <b>4. Display Total amount.</b> <b>4a. Invoke Enter/Edit Customer information (Display form)</b>  <b>5. Update customer information.</b> <b>5a. Invoke Validate Customer CC info (Display form)</b> <b>6. Create Payment Transaction</b>  <b>7. Update database. Print tickets with delivery instructions.</b>
Exception Conditions:	<b>5. Credit card not validated, mark tickets on hold.</b> *	

Use Case Name:	<b>Buy tickets for a Package</b>	
Scenario:	<b>Purchase contiguous tickets for a set of events in a package</b>	
Triggering Event:	<b>Customer wants to buy tickets for a package of events</b>	
Brief Description:	<p><b>Customer selects package. System displays productions in package.</b>  <i>Note: Your team has several options for solutions here. Choose any combination of one and two.</i></p> <p><i>1. Same night (such as Thursday night) for all productions versus select each date/time individually.</i></p> <p><i>2. Same price category for all events or different price category for each event.</i></p> <p><b>Customer/System select events and tickets. Customer enters customer information, CC information. System prints tickets.</b></p>	
Actors:	<b>Customer</b>	
Stakeholders:		
Preconditions:	<b>Package must exist    Production must exist    Event must exist</b> <b>Event Seat must exist    Venue/Seller must exist (?)</b>	
Postconditions:	<b>Sale exists (and appropriate subclass exists)</b> <b>PaymentTransaction exists    Customer Exists (optional)</b> <b>Ticket(s) must exist    EventSeat updated</b>	
Flow of Events:	Actor	System
	<ol style="list-style-type: none"> <li><b>1. Customer selects package of events.</b></li> <li><b>2. Customer selects event – Enter event info – production, date, time (for every event in the package)</b></li> <li><b>3. Invoke “Search for available seats” as much as required.</b></li> <li><b>4. Select desired seats.</b></li> <li><b>5. No more events in package.</b></li> <li><b>6. Enter customer information</b></li> <li><b>7. Enter CC information.</b></li> <li><b>8. Finalize (Enter delivery request).</b></li> </ol>	<ol style="list-style-type: none"> <li><b>1. Display package information (with included productions)</b></li> <li><b>2. Display event information</b>  2a. Create sale (possible in the form of a shopping cart). Create sale subclass.</li> <li><b>3. Execute “Search for available seats.”</b></li> <li><b>4. Create tickets and connect to sale. Update status of EventSeat.</b>  4a. Return to Actor 2. to select the next event in the package.</li> <li><b>5. Display Total amount.</b>  5a. Invoke Enter/Edit Customer information (Display form)</li> <li><b>6. Update customer information.</b>  6a. Invoke Validate Customer CC info (Display form)</li> <li><b>7. Create Payment Transaction</b></li> <li><b>8. Update database. Print tickets with delivery instructions.</b></li> </ol>
Exception Conditions:	<b>5. Credit card not validated, mark tickets on hold.</b>	

Use Case Name:	<b>Export Venue Revenue Information</b>	
Scenario:		
Triggering Event:	<b>Manager needs Venue Revenue Report</b>	
Brief Description:	<b>Note: It should have options to be export based on (1) Single event, (2) a Production, (3) a Time period.</b> <b>It should include data such as Production, Event, Seats available, Seats sold, seats unsold, revenue individual sales, revenue from package sales, etc.</b>	
Actors:	<b>Manager</b>	
Stakeholders:	<b>Manager</b>	
Preconditions:	<b>Venue must exist</b> <b>Sales must exist</b> <b>PaymentTrans must exist</b> <b>Events must exist</b> <b>EventSeats must exist</b>	
Postconditions:	<b>None</b>	
Flow of Events:	Actor	System
	<b>1. Enter export criteria</b>	<b>1. Generate XML File</b>
Exception Conditions:		

Use Case Name:	<b>Export Foreign Venue Commission Information</b>	
Scenario:		
Triggering Event:		
Brief Description:	<b>This information is used to create an invoice for a foreign Venue. The data is extracted from those sales of local tickets done by a foreign venue. Options should include (1) export for a particular foreign venue, (2) export for all foreign venues. Information will include by event (date, time), number of tickets sold, total revenue from sales, commissions allowed, amount to be remitted, etc.</b>	
Actors:	<b>Manager</b>	
Stakeholders:	<b>Manager</b>	
Preconditions:	<b>Manager</b>	
Postconditions:	<b>Venue must exist Sales must exist PaymentTrans must exist Events must exist EventSeats must exist</b>	
Flow of Events:	Actor	System
	<b>1. Enter export criteria</b>	<b>1. Generate XML file</b>
Exception Conditions:		