Architectural and Object Oriented interaction design

For

Online Ticket Booking System

(Assignment 2)

Prepared by

Abdul Mannan Omi

Zhiyun Chen

Yuchao Su

Wenjing Wang

Tanvir Ahmed Akash

Flinders University

22 October 2023

Table of Contents

1. Introduction

This document provides a comprehensive overview of the Adelaide Fringe Ticket Booking System, outlining its various components and design elements. It serves as a valuable resource for both stakeholders and system developers involved in the design and development of the system. The Adelaide Fringe Ticket Booking System is an integral part of the Adelaide Fringe website, offering a range of services and benefits to both registered members and non-members.

In this document, we will delve into the various aspects of the system's architecture and design, starting with a system-context diagram to provide a high-level view of the system's interactions with its external environment. This will be followed by box-and-line diagrams that identify key system components and their relationships, allowing a deeper understanding of the system's structure.

We will also explore conceptual class diagrams, system sequence diagrams, partial design class diagrams, and object interaction diagrams, each of which offers insights into different facets of the system's design, including class structures, use case visualization, and method interactions.

Additionally, we will provide a Class Dictionary in the appendix, which will help clarify the roles and responsibilities of various classes within the system, especially those related to controllers and interfaces in the context of use cases.

1.1 System Context and Architecture

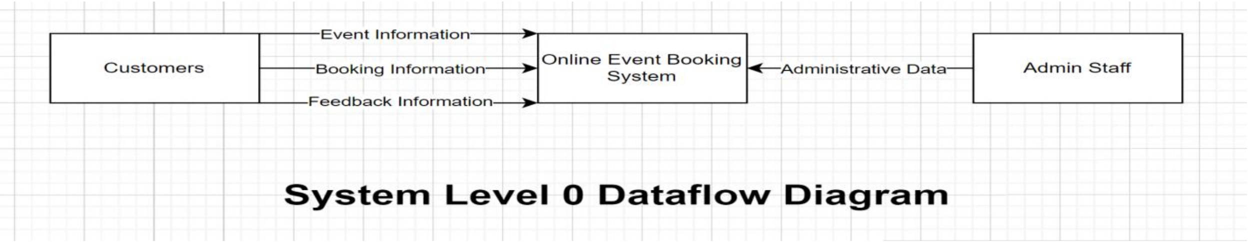


Figure: System level 0 Data Flow Diagram

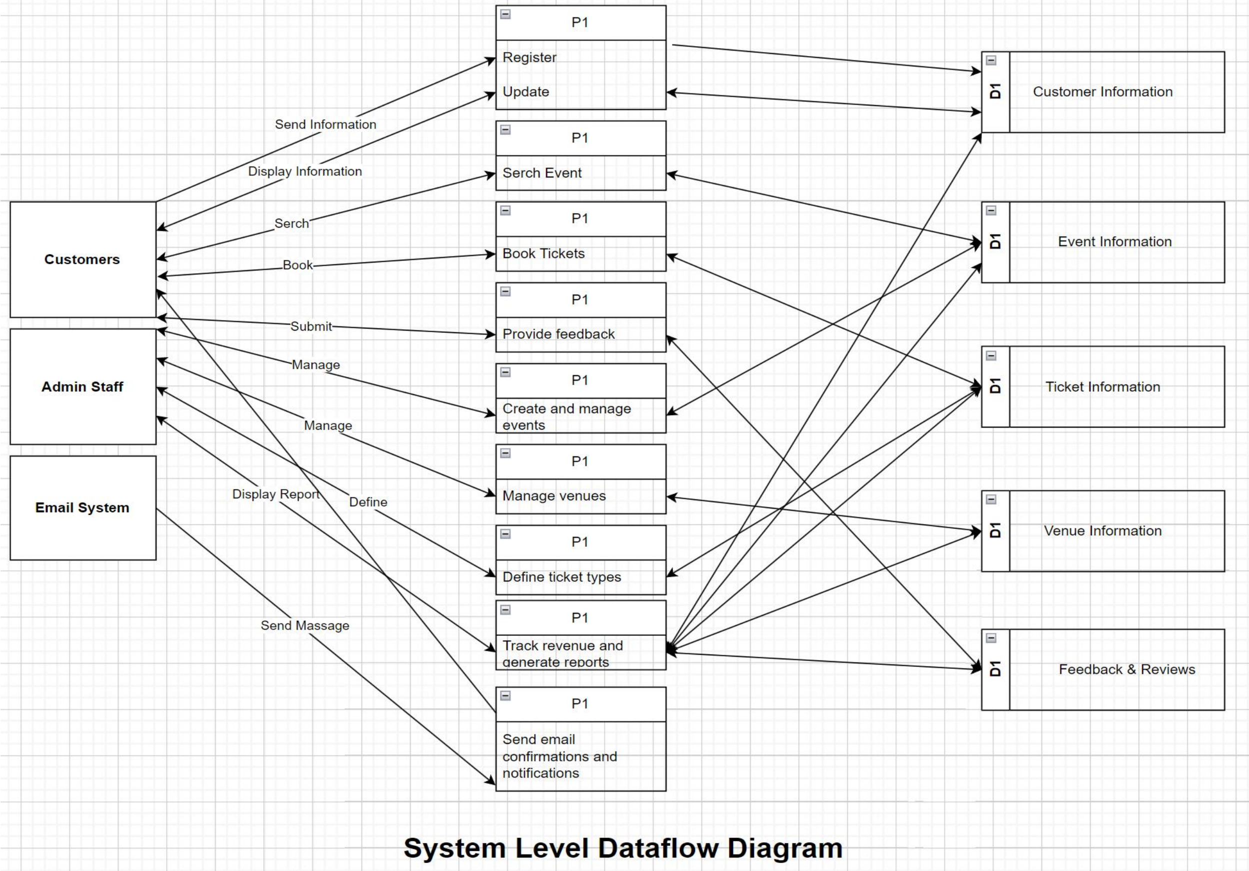


Figure: System level 1 Data Flow Diagram

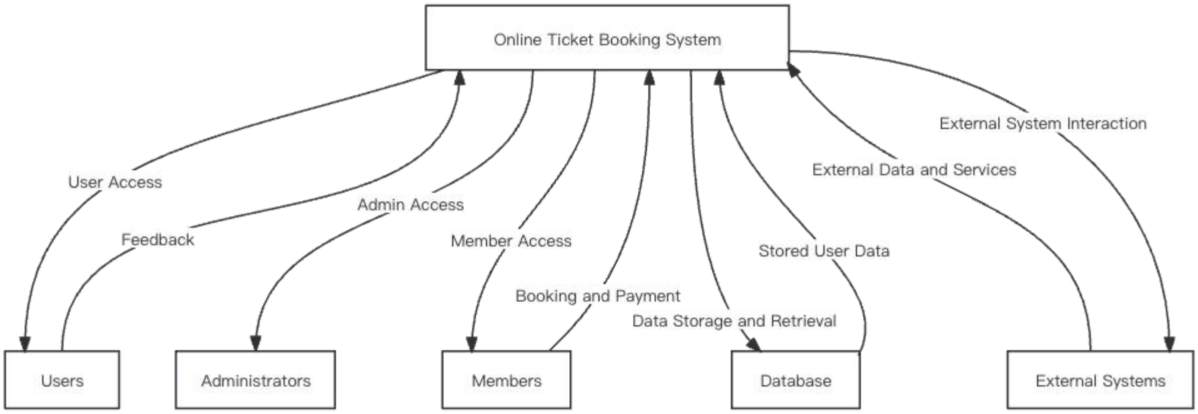


Figure: System Context Diagram

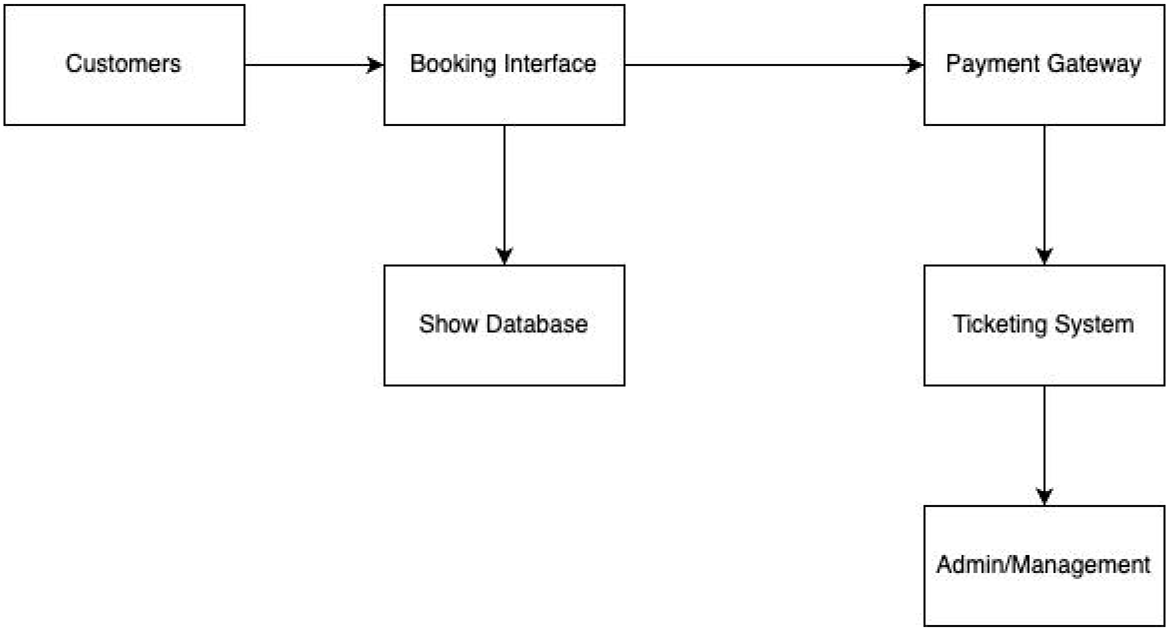


Figure: Box-and Line Diagram

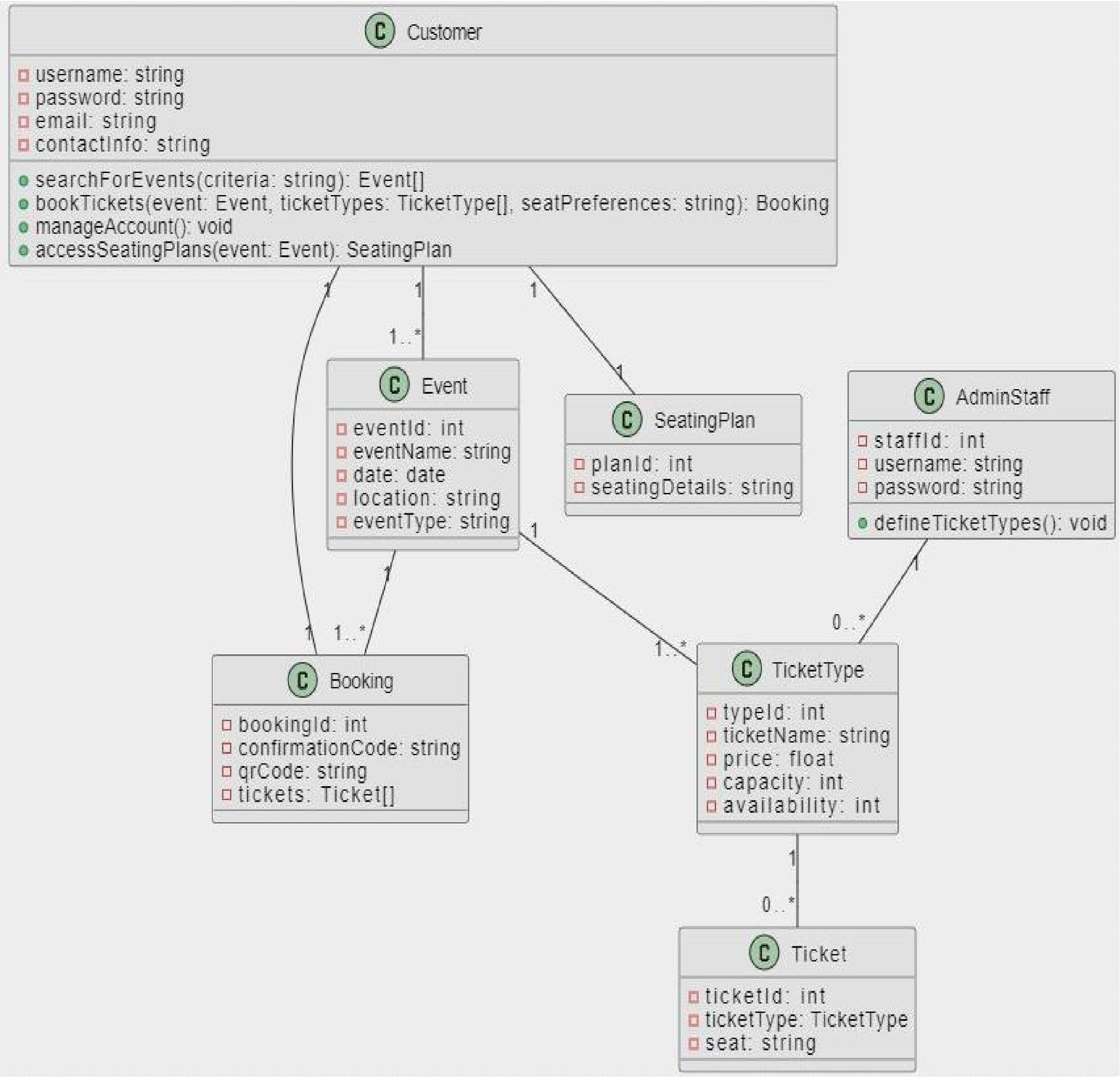
1.2Justification of design decisions

1. **System-Context Diagram**: The system-context diagram provides a high-level overview of the system's interactions with its external environment. This decision was made to help stakeholders and developers understand how the system interfaces with external entities such as users, the Adelaide Fringe website, and payment gateways. It offers a clear visual representation of the system's boundaries and its connections to the outside world.

2. **Box-and-Line Diagrams**: The use of box-and-line diagrams to identify key system components and their relationships is essential for providing a simplified yet informative representation of the system's architecture. This decision was made to facilitate a quick understanding of the system's structural components, making it easier for both developers and non-technical stakeholders to grasp the system's organization.

2. Conceptual Model

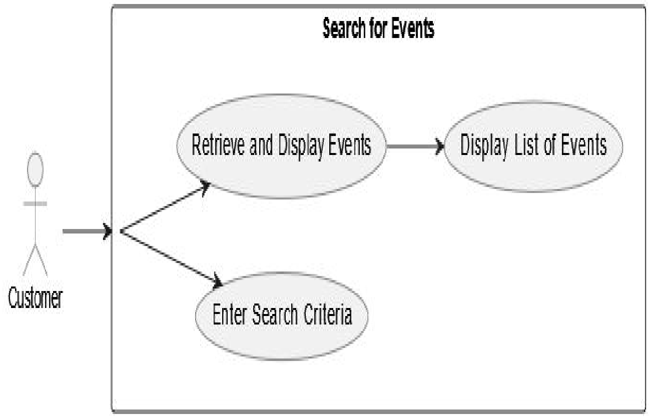
2.1 Conceptual Class Diagram



3. Use-case Level Behavior Design

Use case 1 (Created by Abdul Mannan Omi):

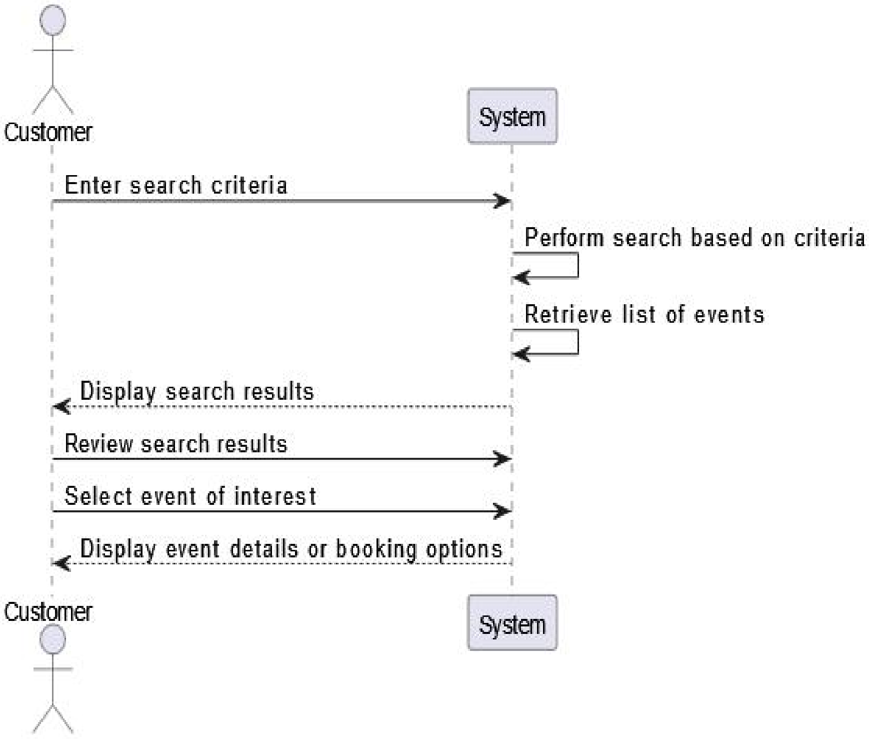
Search for Events	

- Actors: Customer (Primary) - Type: Primary - Description: The customer initiates this use case by entering search criteria (e.g., date, location, event type) to discover events that match their preferences. The system then retrieves and displays a list of relevant events.	 Search for Events Retrieve and Display Events Display List of Events Enter Search Criteria Customer

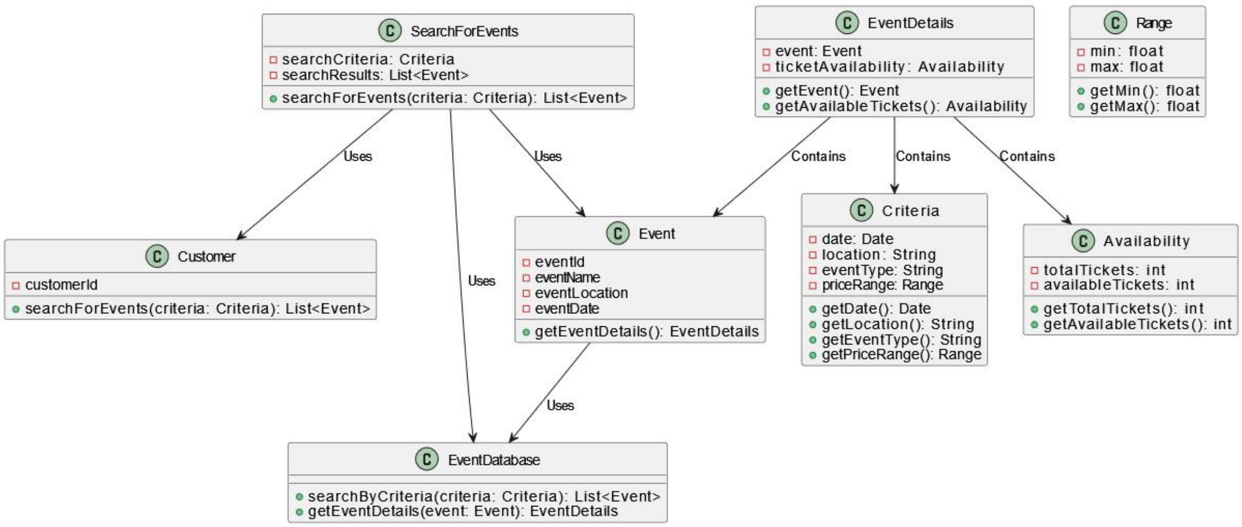
3.1 Process Description:

- The process begins when a customer decides to search for events.
- The customer enters search criteria, which can include parameters like date, location, event type, and any other relevant filters.
- The system processes the search criteria and queries the event database for events that match the customer's preferences.
- The system retrieves a list of relevant events from the database.
- The system displays the list of events to the customer, including event names, dates, locations, and brief descriptions.
- The customer can view additional details about each event, such as pricing, available ticket types, and seating options.
- The customer can then select an event from the search results to access more information or proceed to book tickets.

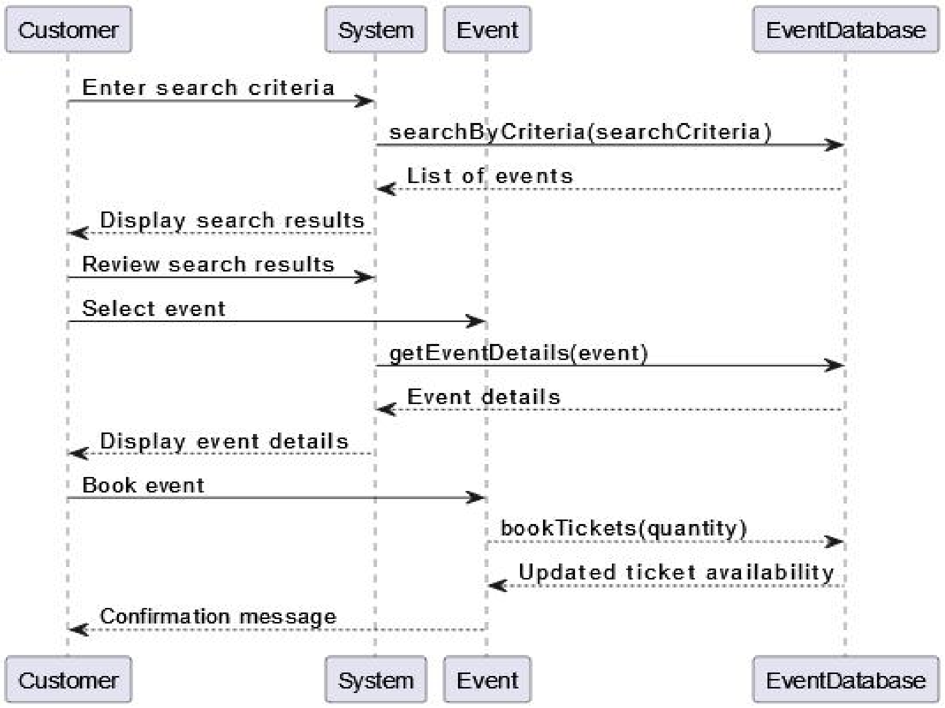
3.2 System Sequence Diagram



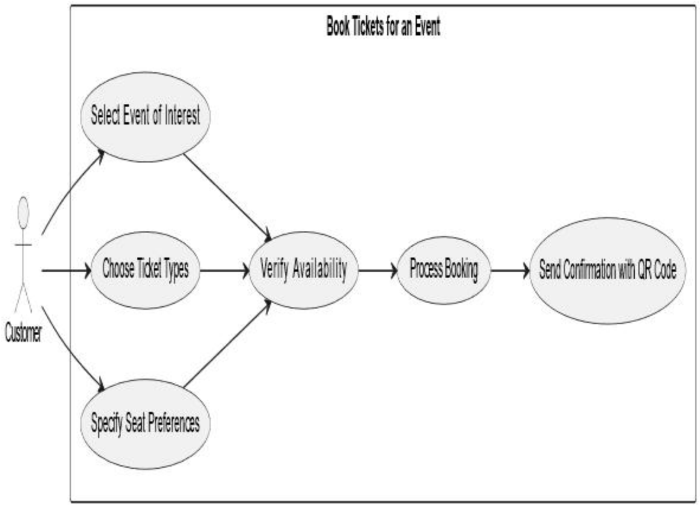
3.3 Partial design class diagram



3.4 Object interaction diagram (sequence diagram)



Use case 2 (Created by Yuchao Su):

Book Tickets for an Event	
▪ Actors: Customer (Primary) ▪ Type: Primary ▪ Description: The customer selects an event of interest, chooses ticket types, specifies seat preferences (if applicable), and proceeds to book tickets. The system verifies availability, processes the booking, and sends a confirmation with a QR code for admission.	 Book Tickets for an Event Select Event of Interest Choose Ticket Types Verify Availability Process Booking Send Confirmation with QR Code Specify Seat Preferences Customer

3.1 Process description

- The process begins with the customer selecting an event from the search results.

- The system displays event details, including event name, date, location, ticket types, and seating plans (if applicable).
- The customer chooses the desired ticket types and specifies the number of tickets they wish to purchase.
- If the event offers assigned seating, the customer can select specific seats from the seating plan.
- The system verifies ticket availability and reserves the selected tickets.
- The system prompts the customer to review their booking and proceed to payment.
- The customer provides payment information.
- The system processes the payment.
- If the payment is successful, the system sends a booking confirmation via email, including a QR code for admission.
- In case all tickets for the selected event are sold out during the booking process, the system informs the customer and suggests they choose another event or date.
- If the payment transaction fails, the system provides an error message and prompts the customer to retry the payment process.

3.2 System Sequence Diagram

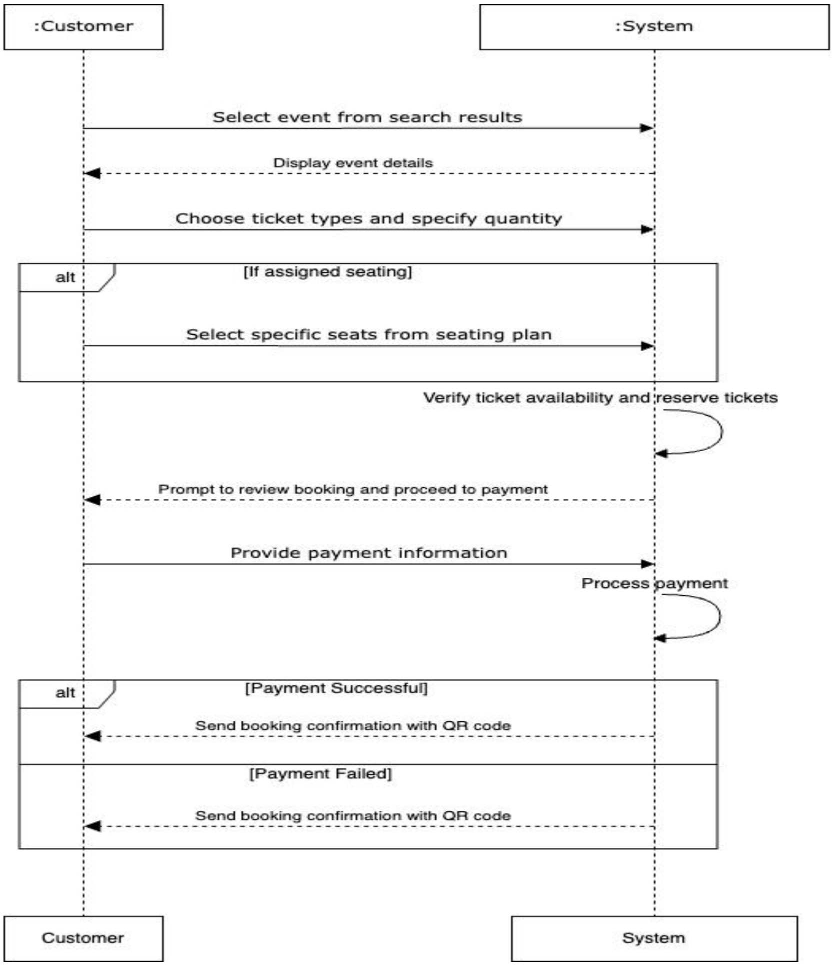
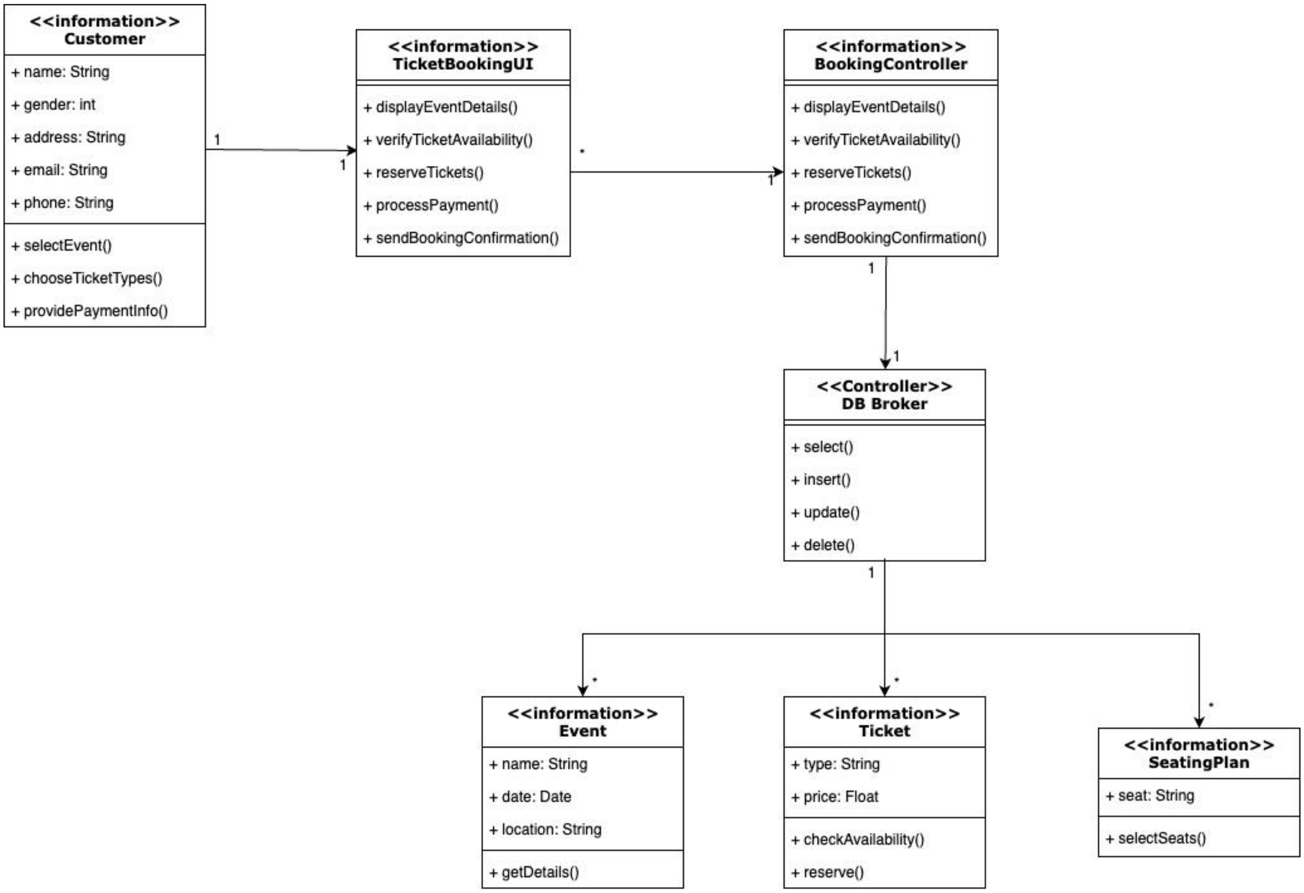
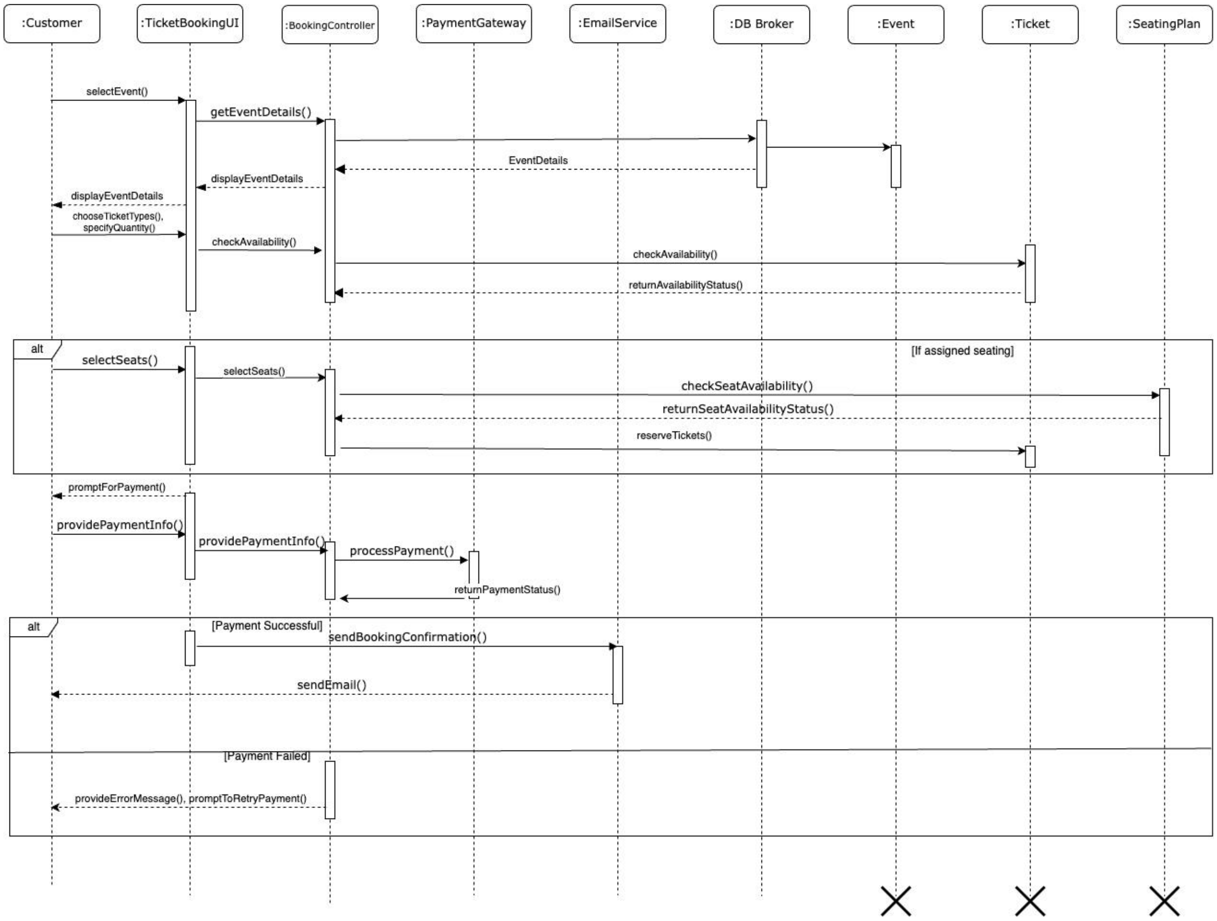

:Customer
:System
Select event from search results
Display event details
Choose ticket types and specify quantity
alt
[If assigned seating]
Select specific seats from seating plan
Verify ticket availability and reserve tickets
Prompt to review booking and proceed to payment
Provide payment information
Process payment
alt
[Payment Successful]
Send booking confirmation with QR code
[Payment Failed]
Send booking confirmation with QR code
Customer
System

Figure: System Sequence Diagram

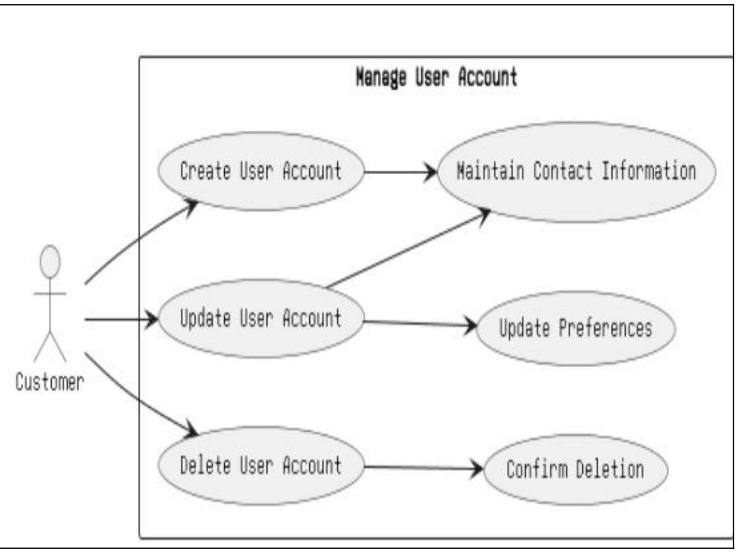
3.3 Partial design class diagram


<<information>>
Customer
+ name: String
+ gender: int
+ address: String
+ email: String
+ phone: String
+ selectEvent()
+ chooseTicketTypes()
+ providePaymentInfo()
1
1
<<information>>
TicketBookingUI
+ displayEventDetails()
+ verifyTicketAvailability()
+ reserveTickets()
+ processPayment()
+ sendBookingConfirmation()
*
1
<<information>>
BookingController
+ displayEventDetails()
+ verifyTicketAvailability()
+ reserveTickets()
+ processPayment()
+ sendBookingConfirmation()
1
1
<<Controller>>
DB Broker
+ select()
+ insert()
+ update()
+ delete()
1
*
*
*
<<information>>
Event
+ name: String
+ date: Date
+ location: String
+ getDetails()
<<information>>
Ticket
+ type: String
+ price: Float
+ checkAvailability()
+ reserve()
<<information>>
SeatingPlan
+ seat: String
+ selectSeats()

3.4 Object interaction diagram (sequence diagram)



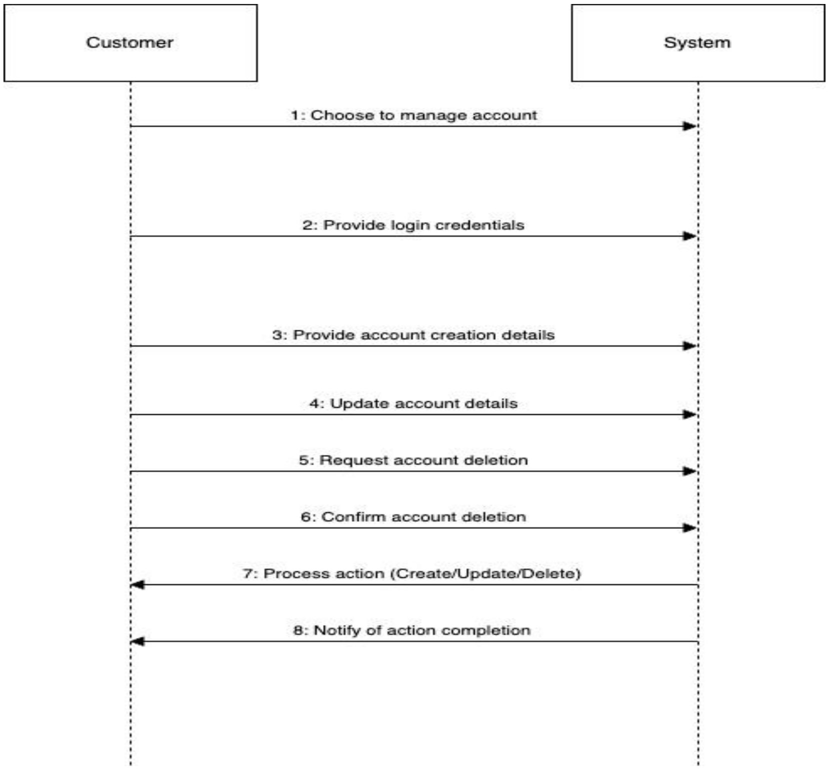
Use case 3 (Created by Wenjing Wang):

Manage User Account	 Manage User Account Create User Account Maintain Contact Information Update User Account Update Preferences Delete User Account Confirm Deletion Customer
▪ Actors: Customer (Primary) ▪ Type: Primary ▪ Description: Customers can create, update, or delete their user accounts. This use case allows customers to maintain their contact information and preferences for a seamless booking experience.	

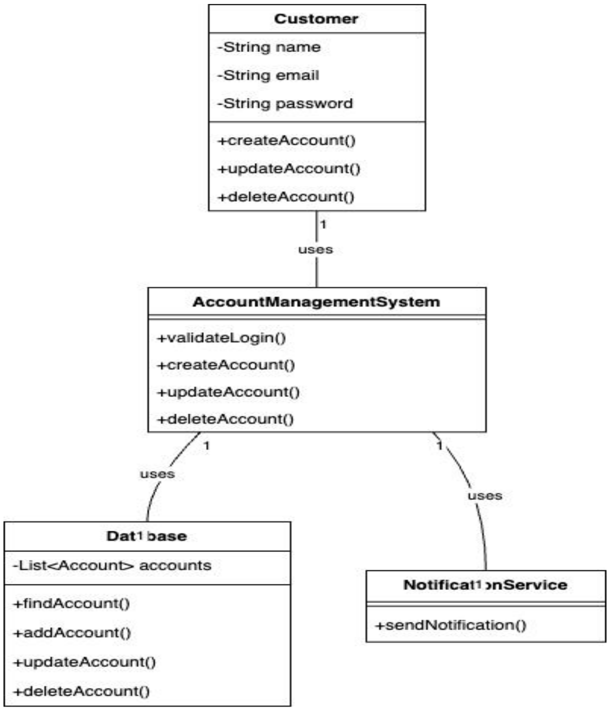
3.1 Process description

- The process begins when the customer decides to manage their user account.
- The customer can choose from several actions, including creating a new account, updating existing account information, or deleting their account.
- If the customer chooses to create a new account:
  - They provide their contact information and preferences.
  - The system verifies the provided information and creates a new user account.
- If the customer chooses to update their account:
  - They can modify their contact information and preferences as needed.
  - The system validates the changes and updates the user account.
- If the customer chooses to delete their account:
  - The system prompts the customer to confirm the deletion.
  - If confirmed, the system deletes the user account, removing all associated information.
- After completing the chosen action, the customer's user account is in the desired state.

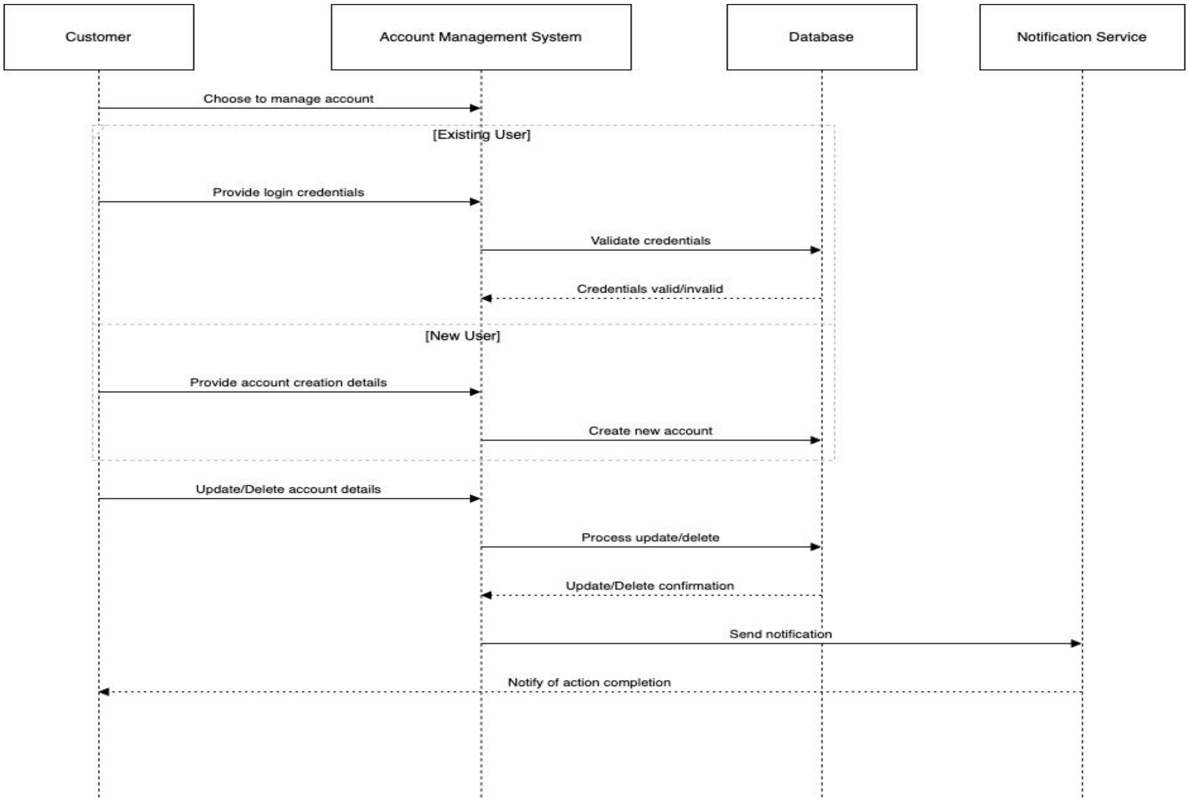
3.2 System Sequence Diagram


Customer
System
1: Choose to manage account
2: Provide login credentials
3: Provide account creation details
4: Update account details
5: Request account deletion
6: Confirm account deletion
7: Process action (Create/Update/Delete)
8: Notify of action completion

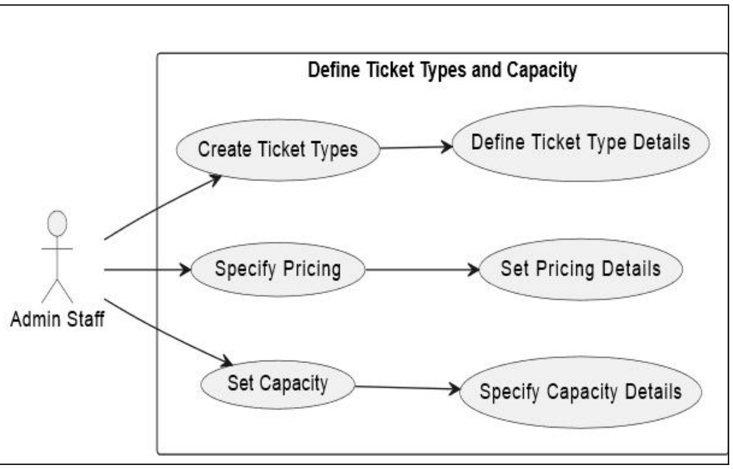
3.3 Partial design class diagram



3.4 Object interaction diagram (sequence diagram)


Customer
Account Management System
Database
Notification Service
Choose to manage account
[Existing User]
Provide login credentials
Validate credentials
Credentials valid/invalid
[New User]
Provide account creation details
Create new account
Update/Delete account details
Process update/delete
Update/Delete confirmation
Send notification
Notify of action completion

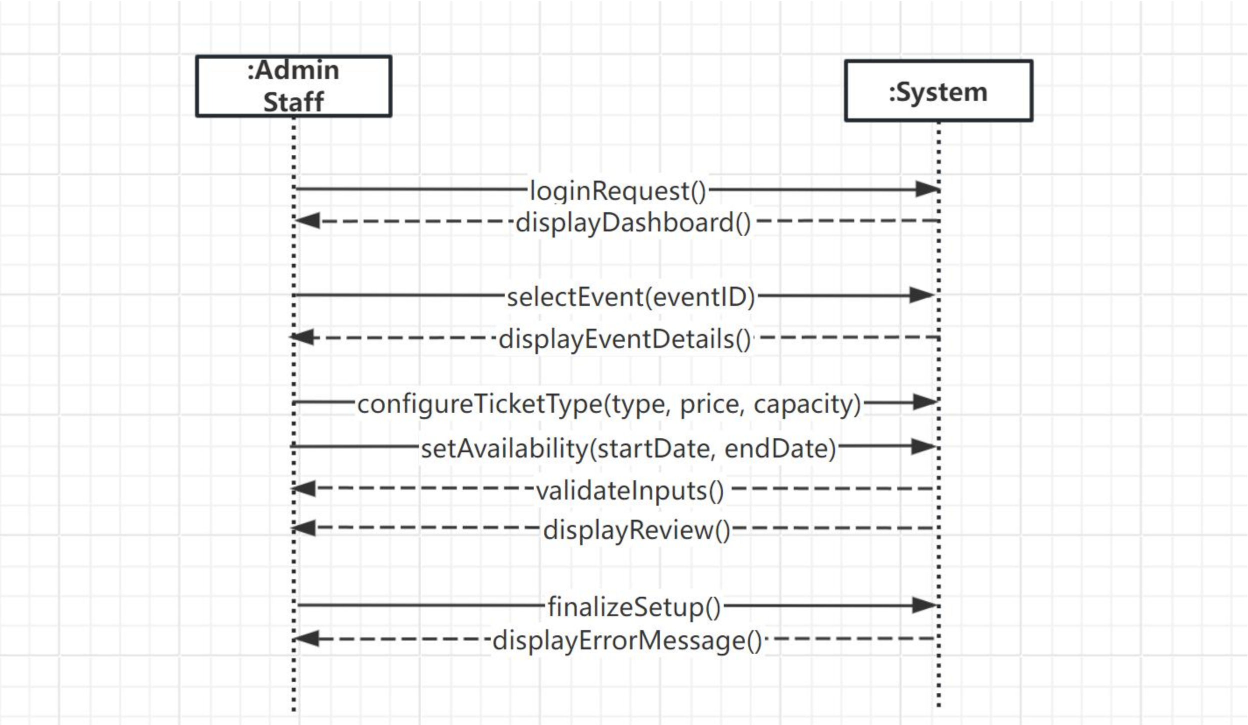
User case 4 (Created by Zhiyun Chen):

Define Ticket Types and Capacity (Chen)	 Define Ticket Types and Capacity Create Ticket Types Define Ticket Type Details Specify Pricing Set Pricing Details Set Capacity Specify Capacity Details Admin Staff
▪ Actors: Admin Staff (Primary) ▪ Type: Primary ▪ Description: Admin staff define different ticket types for events, specifying pricing, capacity, and availability. This use case ensures a variety of ticket options are available to customers.	

3.1 Process description

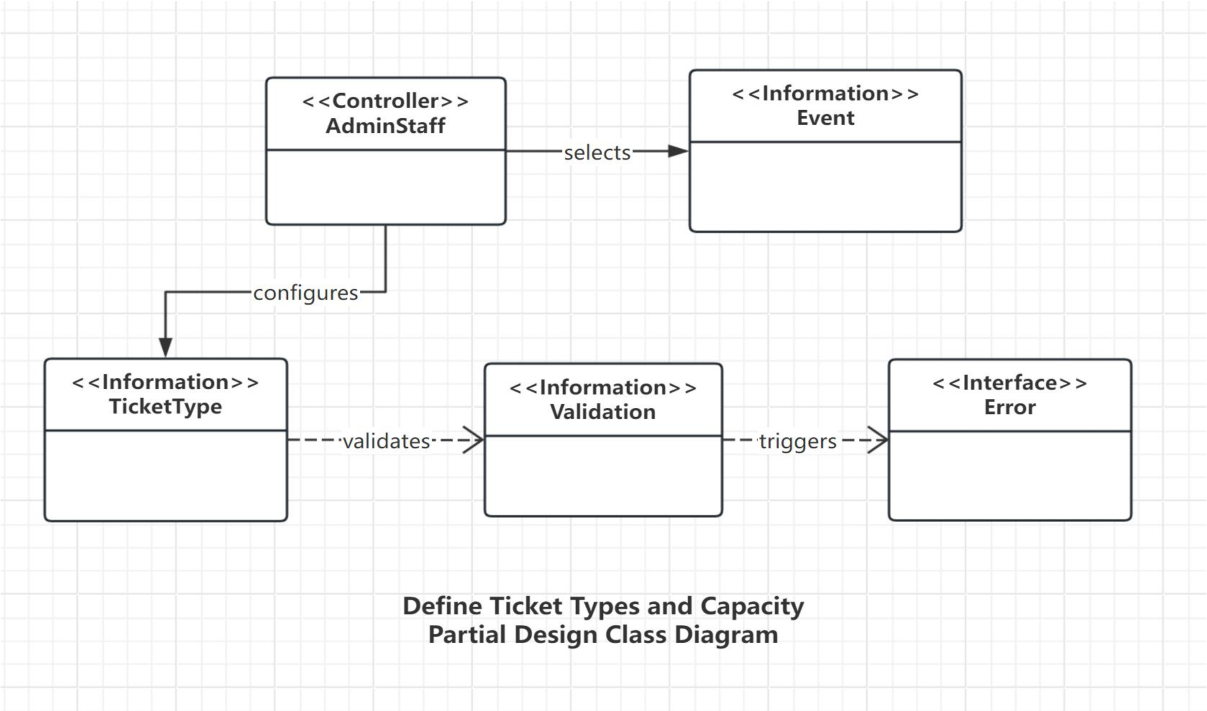
- The process begins when the admin staff decide to define ticket types and capacity for an event.
- The admin staff can create, update, or delete ticket types for an event, specifying details such as pricing, capacity, and availability.
- If the admin staff chooses to create a new ticket type:
    - They provide details such as the ticket name, price, capacity, and availability.
    - The system verifies the provided information and adds the new ticket type to the event.
- If the admin staff chooses to update an existing ticket type:
    - They can modify the ticket type's details, including price, capacity, and availability.
    - The system validates the changes and updates the ticket type.
- If the admin staff chooses to delete a ticket type:
    - The system prompts the admin staff to confirm the deletion.
    - If confirmed, the system removes the ticket type from the event, updating capacity and availability accordingly.
- After completing the chosen action, the ticket types for the event are in the desired state.

3.2 System Sequence Diagram



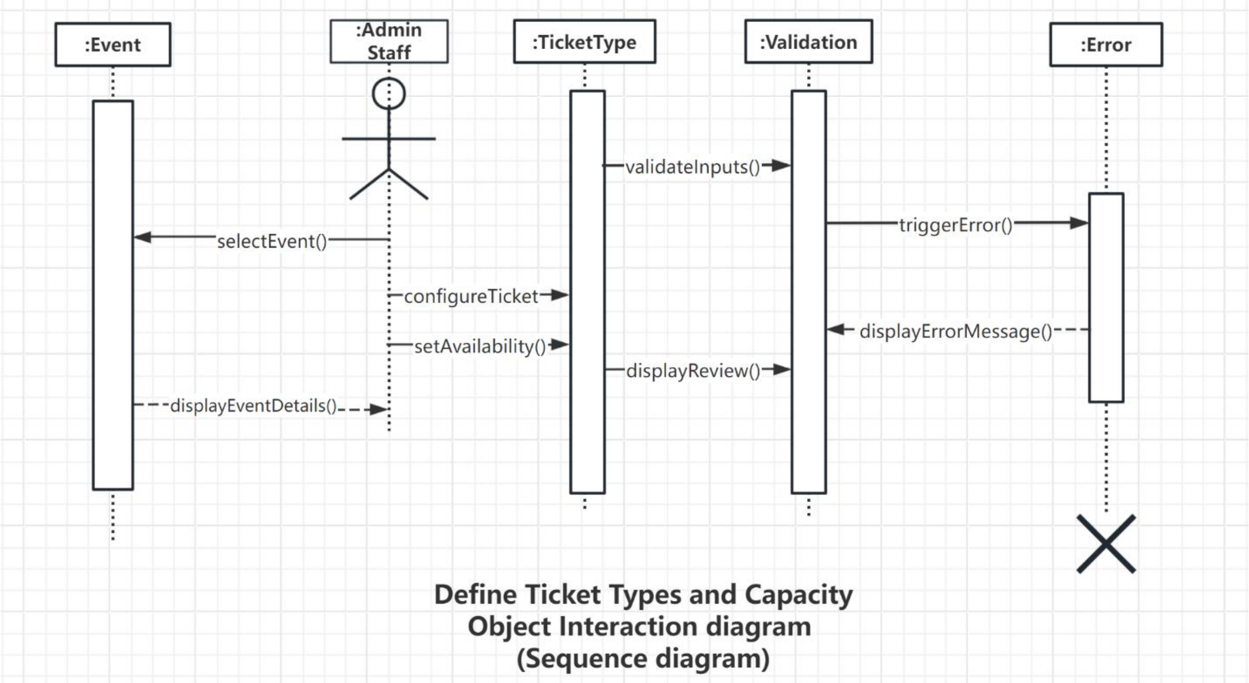
Define Ticket Types and Capacity System Sequence Diagram

3.3 Partial design class diagram



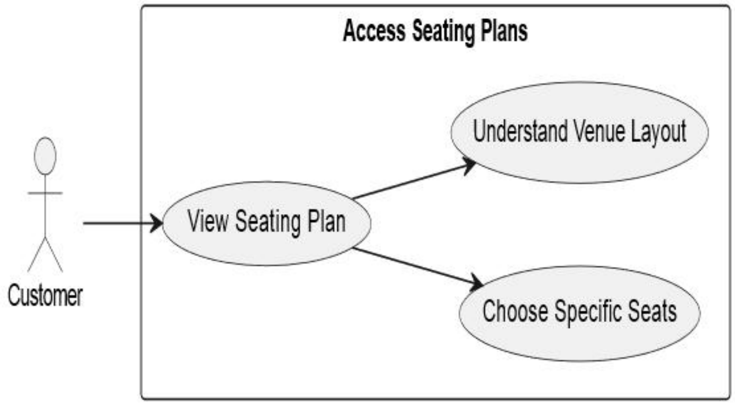
Define Ticket Types and Capacity
Partial Design Class Diagram

3.4 Object interaction diagram (sequence diagram)



Define Ticket Types and Capacity
Object Interaction diagram
(Sequence diagram)

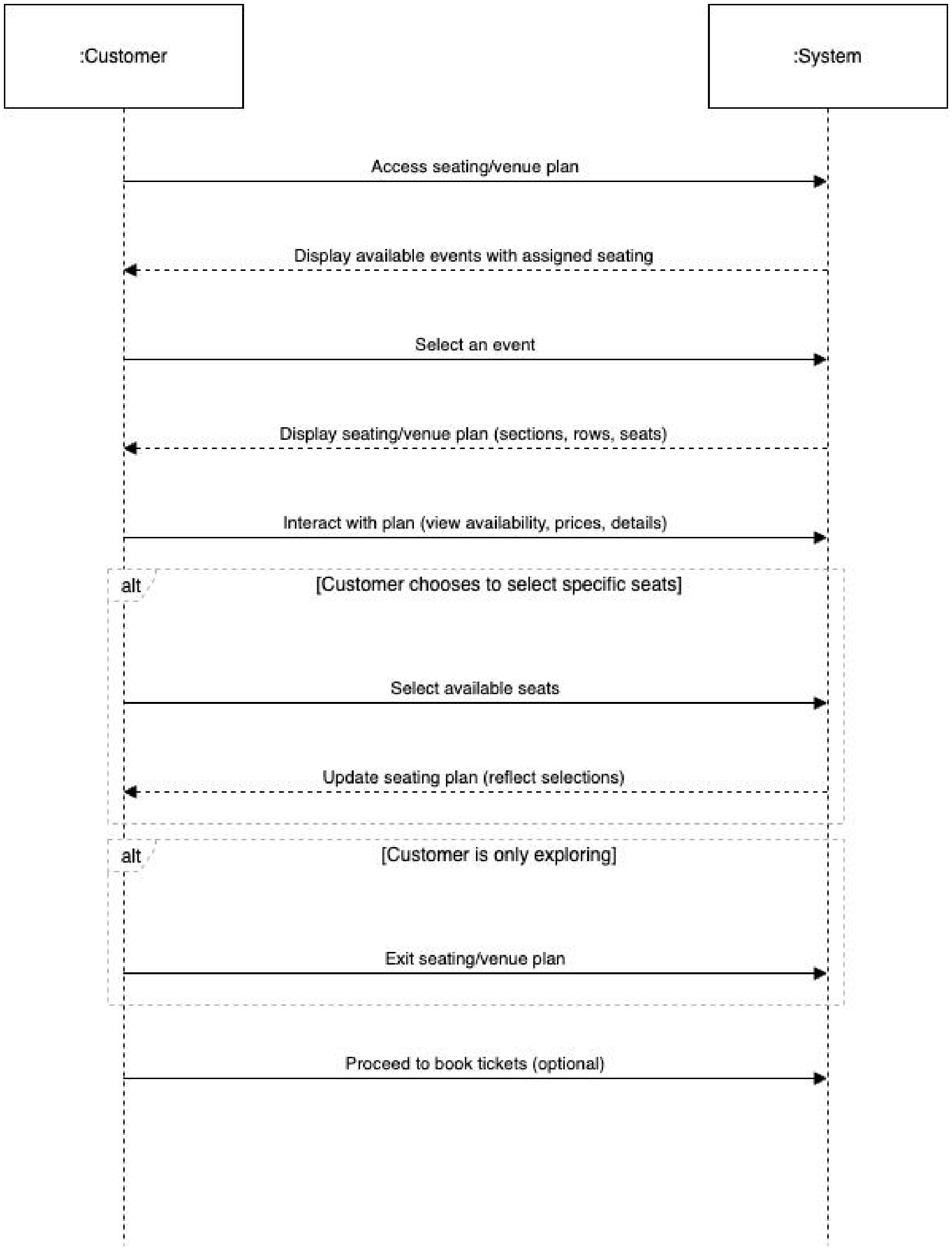
Use case 5 (Created by Tanvir Ahamed Akash):

Access Seating Plans (Tanvir)	
▪ Actors: Customer (Primary) ▪ Type: Primary ▪ Description: Customers can access seating/venue plans for events that offer assigned seating. This feature helps customers choose specific seats or understand the venue layout.	

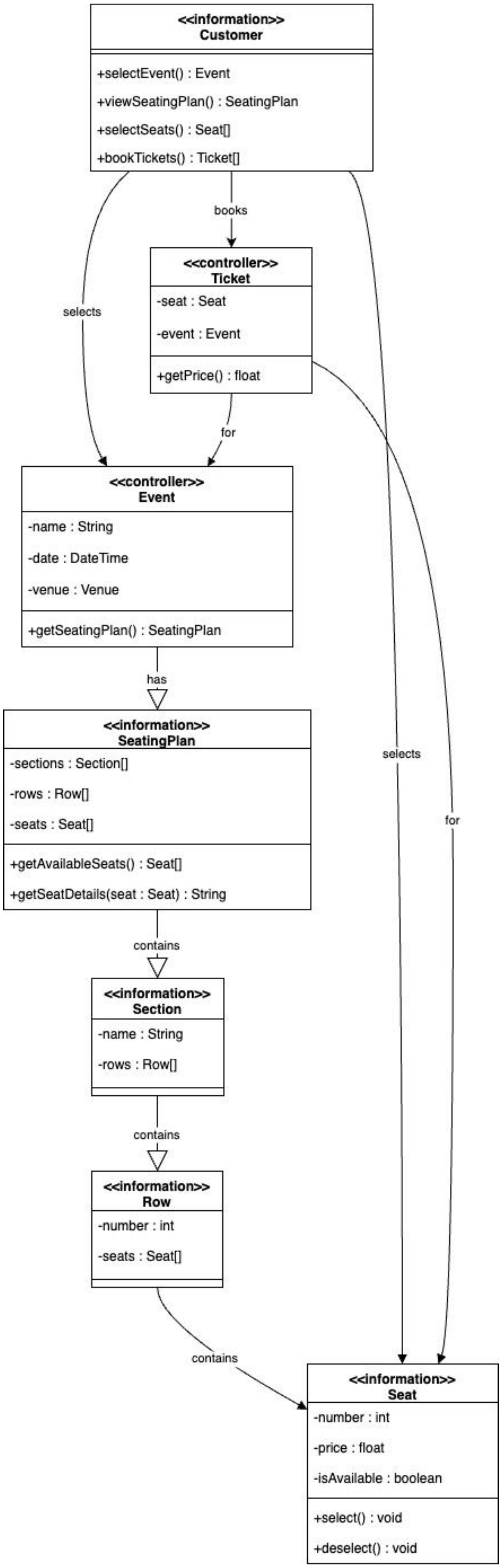
3.1 Process Description:

- The process begins when a customer intends to access seating or venue plans for an event.
- The customer selects an event that offers assigned seating.
- The system displays the seating/venue plan for the selected event, including information about available sections, rows, and seats.
- The customer can interact with the plan to view seat availability, seat prices, and other relevant details.
- If the customer chooses to select specific seats:
  - They can click on available seats to add them to their booking.
  - The system updates the seating plan to reflect the customer's selections.
- If the customer is only exploring, they can exit the seating/venue plan without making selections.
- After interacting with the seating/venue plan, the customer can proceed to book tickets if they wish.

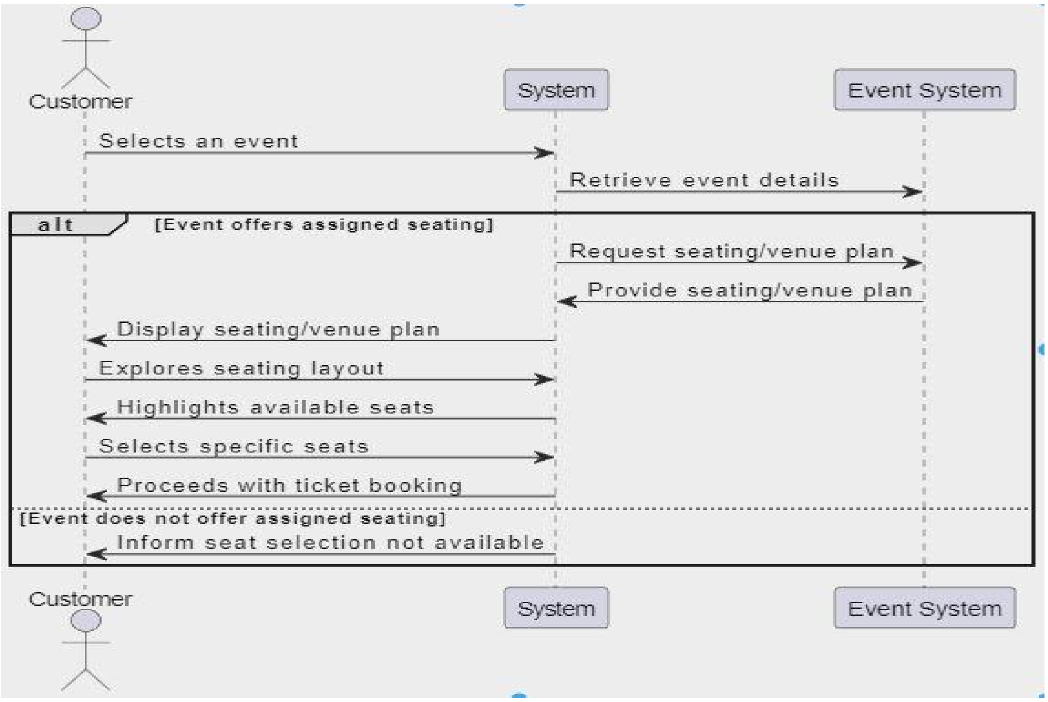
3.2 System Sequence Diagram


:Customer
:System
Access seating/venue plan
Display available events with assigned seating
Select an event
Display seating/venue plan (sections, rows, seats)
Interact with plan (view availability, prices, details)
alt
[Customer chooses to select specific seats]
Select available seats
Update seating plan (reflect selections)
alt
[Customer is only exploring]
Exit seating/venue plan
Proceed to book tickets (optional)

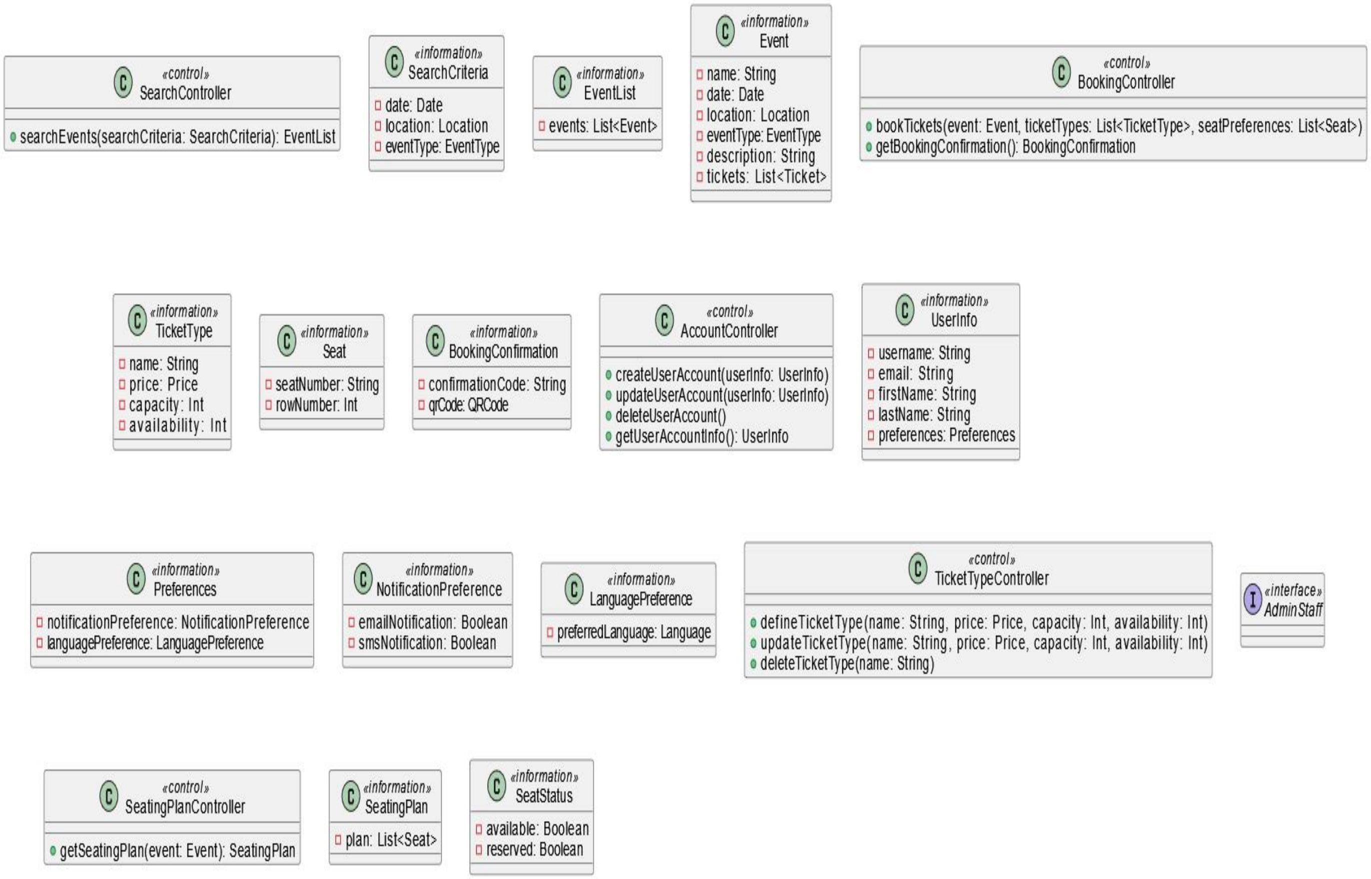
3.3 Partial design class diagram


<<information>>
Customer
+selectEvent() : Event
+viewSeatingPlan() : SeatingPlan
+selectSeats() : Seat[]
+bookTickets() : Ticket[]
books
selects
selects
<<controller>>
Ticket
-seat : Seat
-event : Event
+getPrice() : float
for
for
<<controller>>
Event
-name : String
-date : DateTime
-venue : Venue
+getSeatingPlan() : SeatingPlan
has
<<information>>
SeatingPlan
-sections : Section[]
-rows : Row[]
-seats : Seat[]
+getAvailableSeats() : Seat[]
+getSeatDetails(seat : Seat) : String
contains
<<information>>
Section
-name : String
-rows : Row[]
contains
<<information>>
Row
-number : int
-seats : Seat[]
contains
<<information>>
Seat
-number : int
-price : float
-isAvailable : boolean
+select() : void
+deselect() : void

3.4 Object interaction diagram (sequence diagram)


Customer
System
Event System
Selects an event
Retrieve event details
alt
[Event offers assigned seating]
Request seating/venue plan
Provide seating/venue plan
Display seating/venue plan
Explores seating layout
Highlights available seats
Selects specific seats
Proceeds with ticket booking
[Event does not offer assigned seating]
Inform seat selection not available
Customer
System
Event System

4. Appendix: Class Dictionary


«control» SearchController
+ searchEvents(searchCriteria: SearchCriteria): EventList
«information» SearchCriteria
date: Date
location: Location
eventType: EventType
«information» EventList
events: List<Event>
«information» Event
name: String
date: Date
location: Location
eventType: EventType
description: String
tickets: List<Ticket>
«control» BookingController
bookTickets(event: Event, ticketTypes: List<TicketType>, seatPreferences: List<Seat>)
getBookingConfirmation(): BookingConfirmation
«information» TicketType
name: String
price: Price
capacity: Int
availability: Int
«information» Seat
seatNumber: String
rowNumber: Int
«information» BookingConfirmation
confirmationCode: String
qrCode: QRCode
«control» AccountController
createUserAccount(userInfo: UserInfo)
updateUserAccount(userInfo: UserInfo)
deleteUserAccount()
getUserAccountInfo(): UserInfo
«information» UserInfo
username: String
email: String
firstName: String
lastName: String
preferences: Preferences
«information» Preferences
notificationPreference: NotificationPreference
languagePreference: LanguagePreference
«information» NotificationPreference
emailNotification: Boolean
smsNotification: Boolean
«information» LanguagePreference
preferredLanguage: Language
«control» TicketTypeController
defineTicketType(name: String, price: Price, capacity: Int, availability: Int)
updateTicketType(name: String, price: Price, capacity: Int, availability: Int)
deleteTicketType(name: String)
«interface» AdminStaff
«control» SeatingPlanController
getSeatingPlan(event: Event): SeatingPlan
«information» SeatingPlan
plan: List<Seat>
«information» SeatStatus
available: Boolean
reserved: Boolean