Adik Nasirov s19245

MAS Project Documentation s19245
Football game's ticket buying application

I have some problems with buying ticket to the match between Poland and Belgium, (laczynaspilke.pl). So, I decided to make the application based on this.

User requirements:

1. An app will have a functionality to view the list of all stadiums in the city and should contain data about those stadiums, including their names, addresses and unique id numbers.

2. Each stadium has at least one sector, which should be described mainly by its name and number of seats.

3. An app also should keep information about each seat in the sector by recording its seat number and its regular price.

4. Each game must be recorded by its unique id number, the name of the game(credentials of teams), its date, description, and players. The users should be able to view the list of all games, fitler games by different categories(competition like: world cup, champions league, europe cup) to make searches faster and find games by its title.

5. The users should be able to see the list of sessions(time of the game) for each game and in each stadium, which may occur more than one time in a day. Sessions must have its unique id number, the starting and ending date and time.

6. The users should be able to buy tickets online at chosen game, session, and seat. We also want to store availability of the seats at the chosen session, if there are no available seats, then show an error message popup. We should record every purchased ticket bought by the user as history and keep them in one tab for him to view. Every ticket will store its unique ticket id number, purchased date and its final price, which must be calculated by considering the users' discount. The purchased date cannot be later than the ending date of the current session.

7. The user information such as name, telephone number, email, login, password, unique id number and status will be kept in his profile page. The status should be described by name and the amount of discount in percentage. The users can log in to the app, log out, register, view their own profile page.

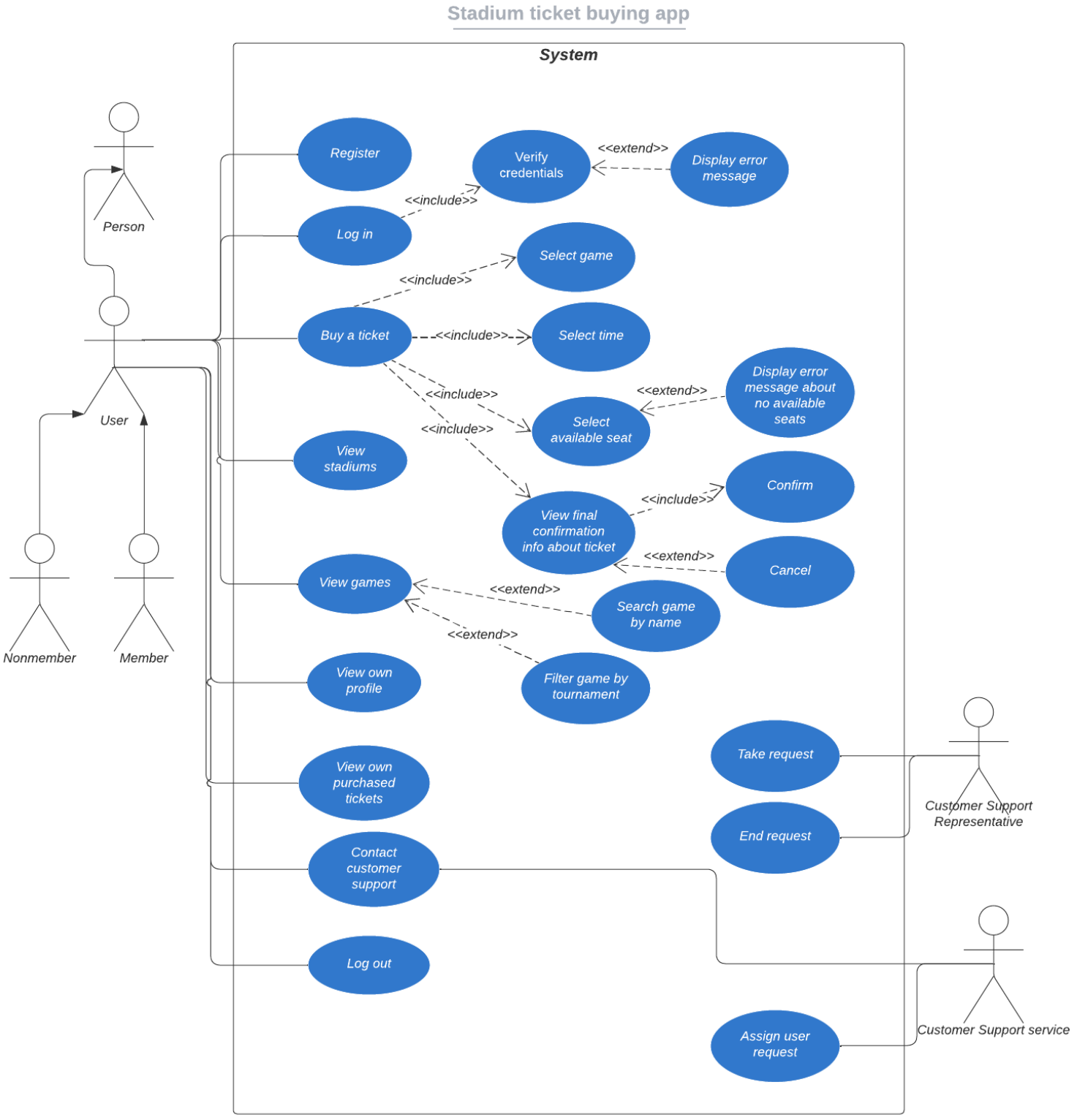
8. There are in general 2 types of users, they are members and nonmembers. Members have certain advantages such as member points, which can be collected and later used as a discount or in other promotions. The user at any moment can become a member and vice versa nonmember.

9. The user will be able to contact Customer Support either by telephone number or by email, in case they will have some issues.
10. Customer Support has a group of specialists - Customer Support Representatives, who will deal with users' requests. For every Customer Support Representative, we should record the name, telephone number, email, unique staff id number, the languages in which he speaks. Moreover, we should store the current availability of each employee, which is essential during assigning user requests.f

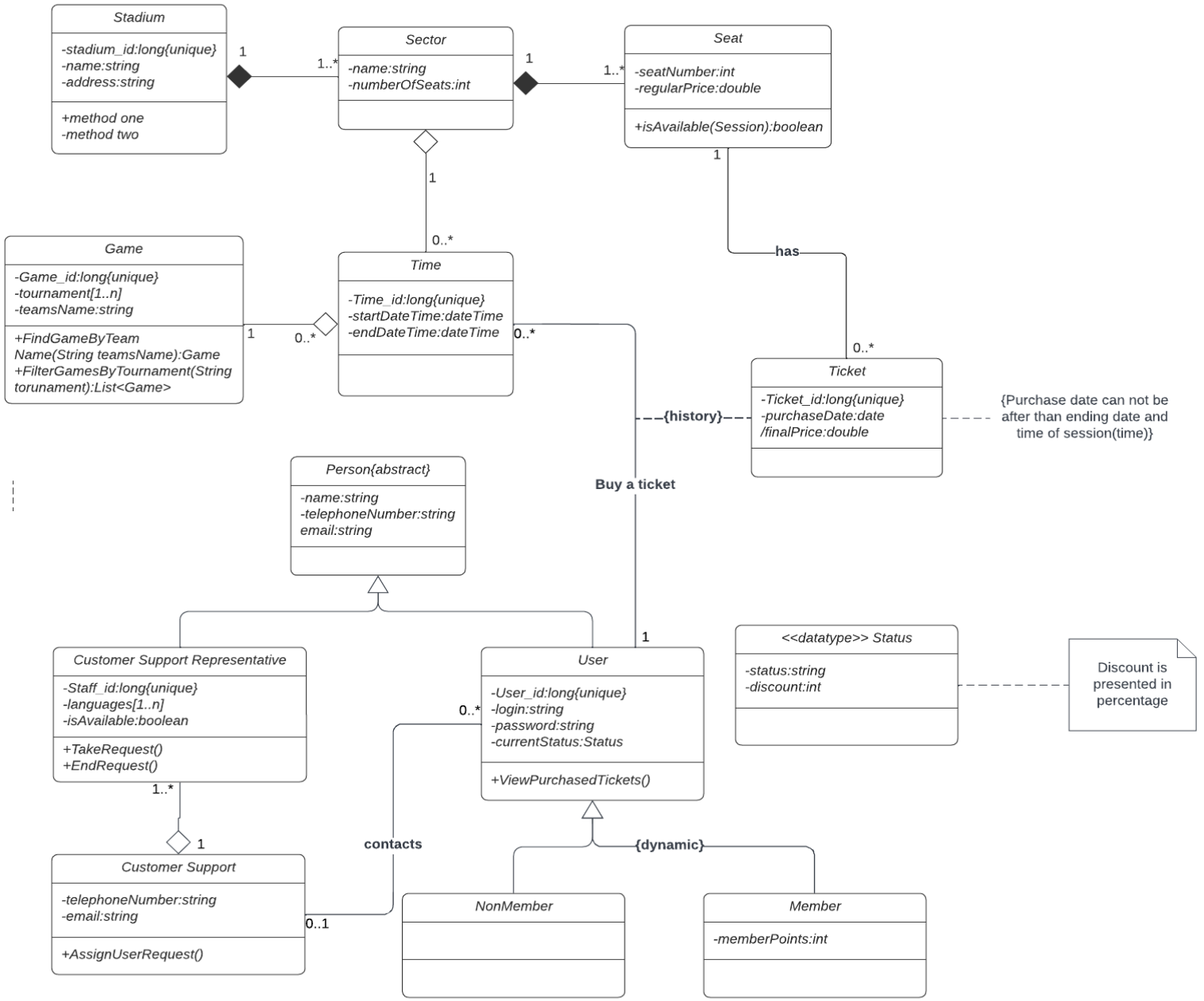
System should allow the user to:

1. Log in, log out, register
2. View all games
3. Quickly search game by its name
4. Quickly filter games by categories
5. View all stadiums
6. Buy tickets at chosen game, session(time) and seat
7. View own profile page
8. View all purchased tickets as history
9. Calculate the final price of tickets considering the discount of the user if he has
10. Properly assign user contact requests to Customer Support Representatives
11. Customer Support Representatives should be able to take the tasks and end them(mark their availability)

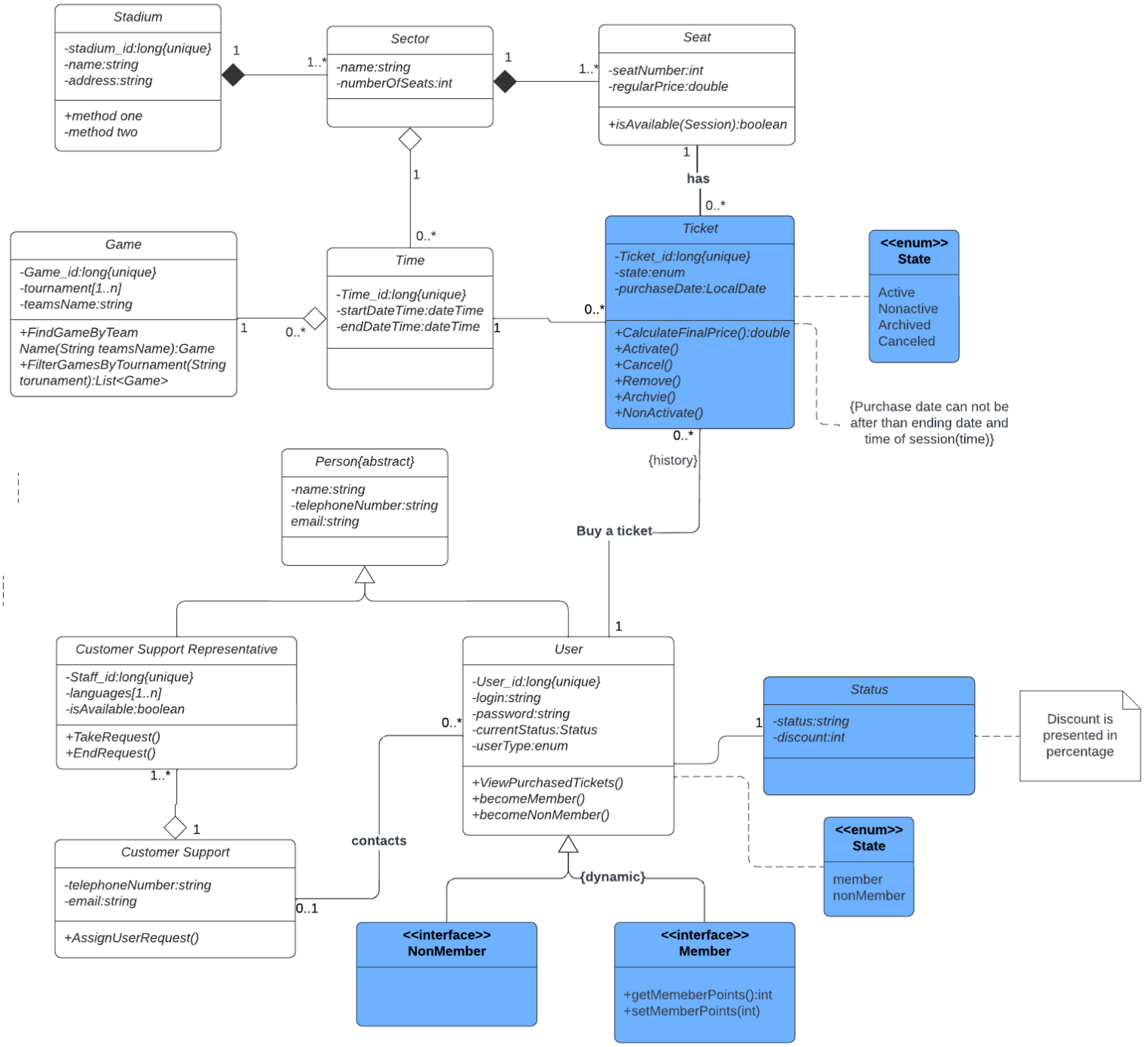
Use Case Diagram


Stadium ticket buying app
System
Person
User
Nonmember
Member
Register
Verify credentials
<<extend>>
Display error message
<<include>>
Log in
Select game
<<include>>
Buy a ticket
<<include>>
Select time
<<include>>
<<include>>
Select available seat
<<extend>>
Display error message about no available seats
View stadiums
View final confirmation info about ticket
<<include>>
Confirm
View games
<<extend>>
Cancel
<<extend>>
Search game by name
<<extend>>
Filter game by tournament
View own profile
View own purchased tickets
Contact customer support
Log out
Take request
End request
Customer Support Representative
Assign user request
Customer Support service

Analytical class diagram



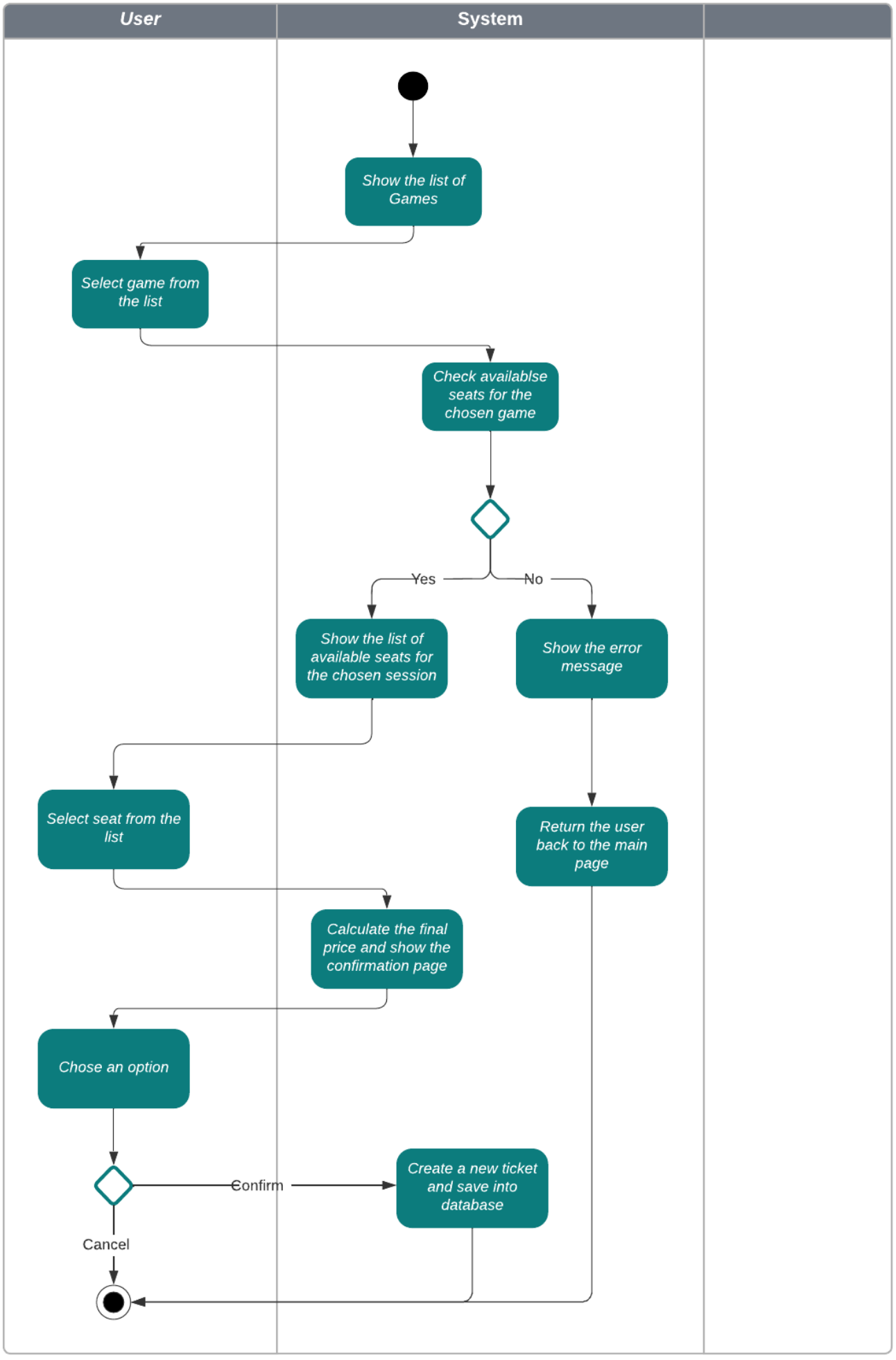
Design Class Diagram



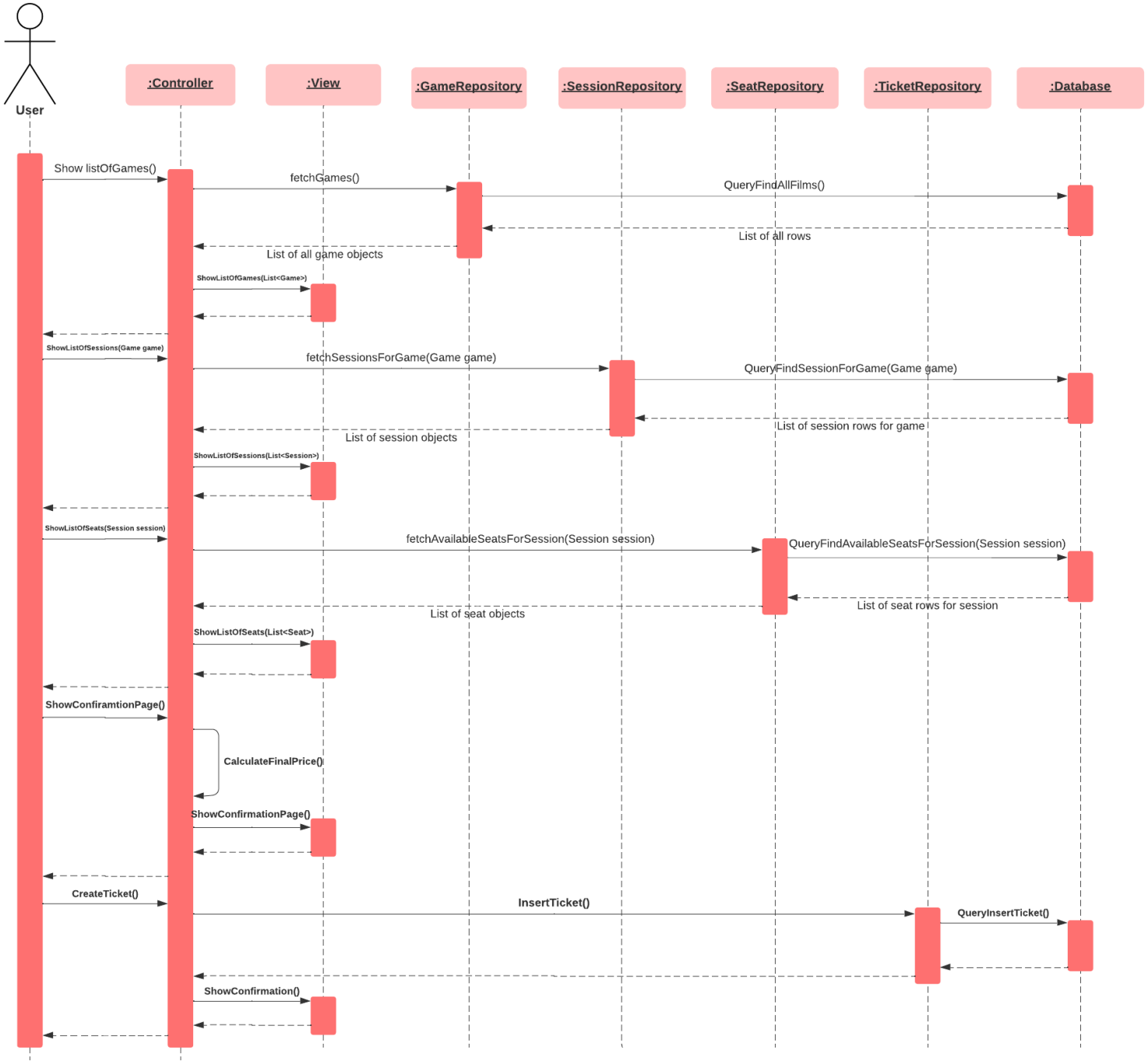
Selected use-case scenario

One of the apps functionalities is - buying ticket.
In order to start buying a ticket process, the user should select “Buy a ticket” option from the main page. First of all, the user should choose a game by selecting the appropriate option from the list of games. Next the system should show a list of available seats of the chosen game.
If there is no available seat, the system shows an error message popup, returns the user back to the main page and the process ends.
If there are available seats, the system shows them in a list and the user selects the appropriate option. After everything is selected, the system calculates the final price of the ticket and opens a confirmation page, it contains the name of the game, the name of the stadium, sector, seat, the date and time, and price. Then the user either confirms or cancels. If the user confirms, the system creates a new ticket with all data and saves into the database, the process ends. If the user decides to cancel the process ends immediately.

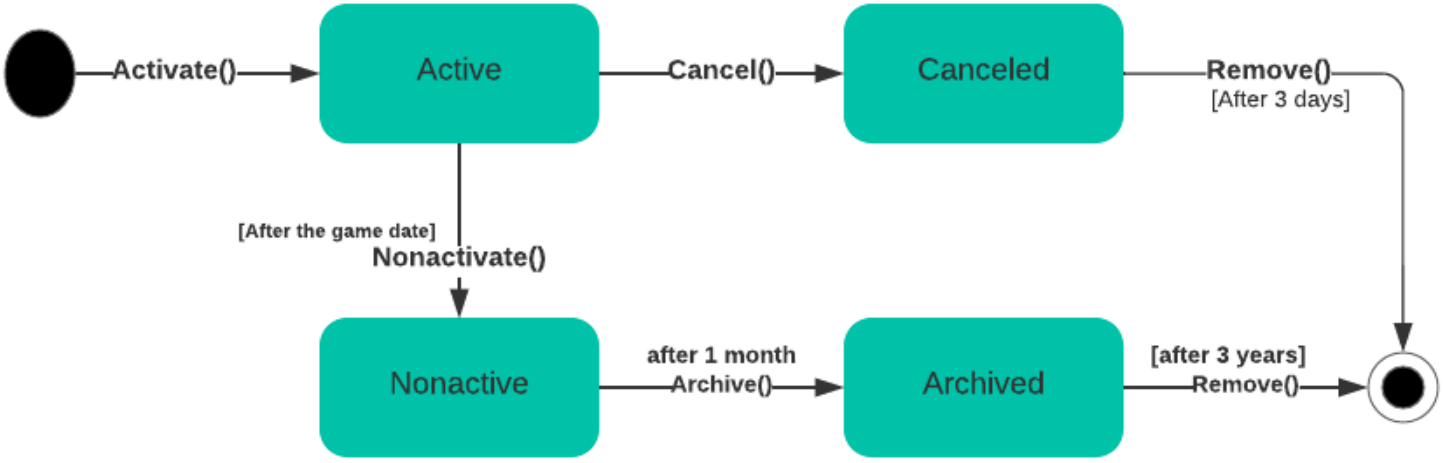
Activity diagram for selected use-case


User
System
Show the list of Games
Select game from the list
Check availablse seats for the chosen game
Yes
No
Show the list of available seats for the chosen session
Show the error message
Select seat from the list
Return the user back to the main page
Calculate the final price and show the confirmation page
Chose an option
Confirm
Create a new ticket and save into database
Cancel

Interaction(sequence) diagram for selected use-case

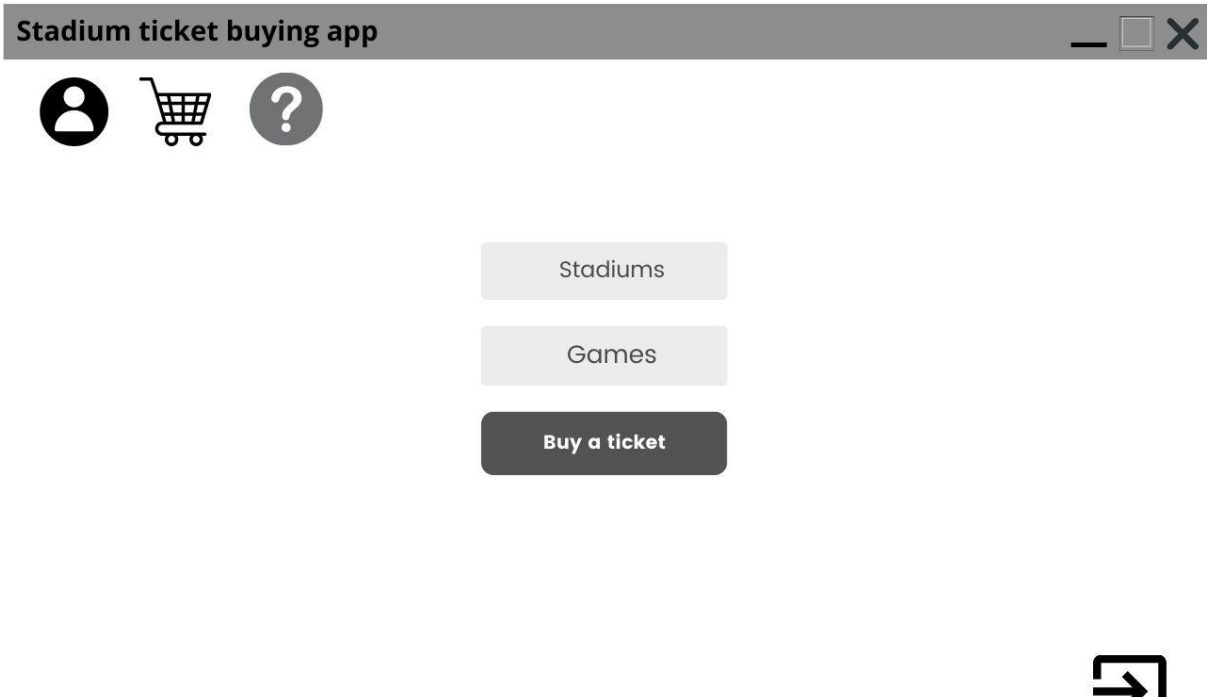


State diagram for selected class
Selected class - Ticket



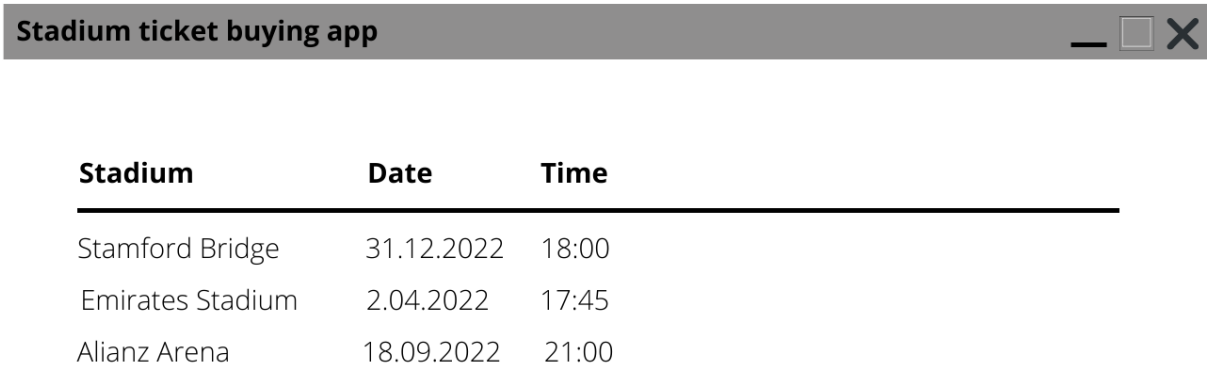
The GUI design for selected use-case

The main page, use-case starts from clicking the Buy a ticket button

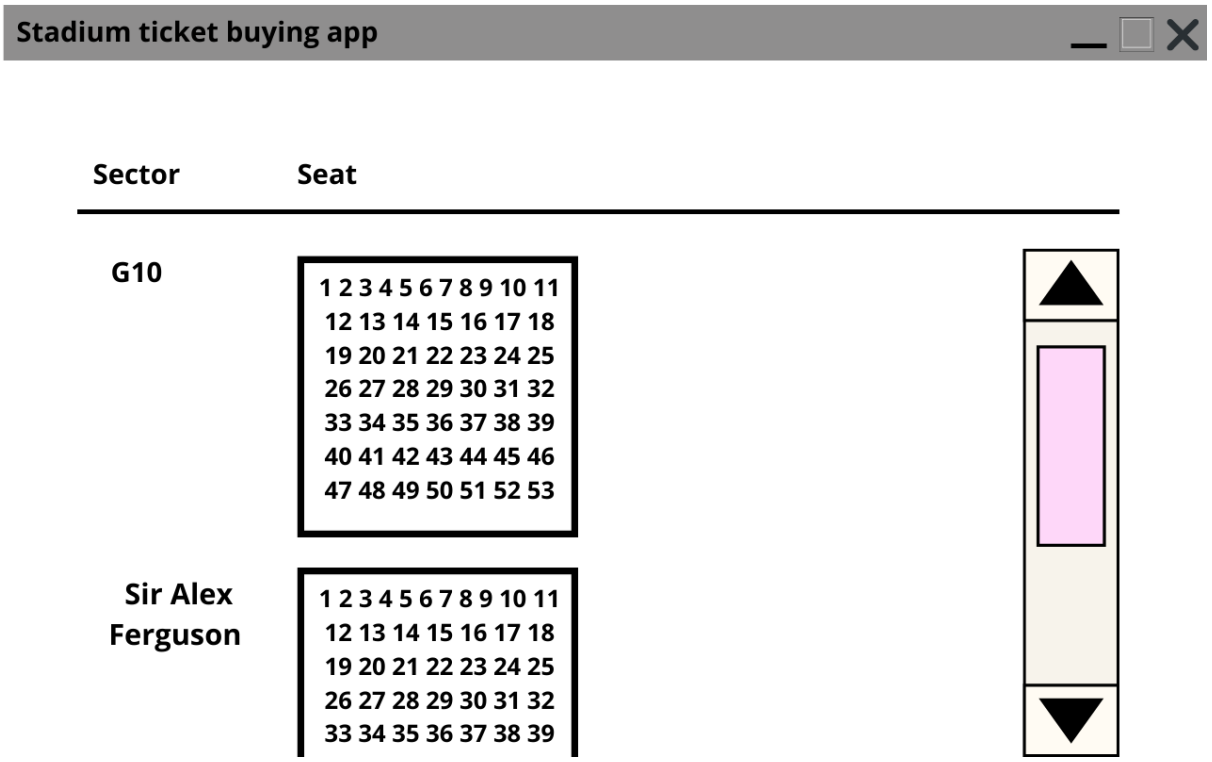


User selects a game from the list of games.

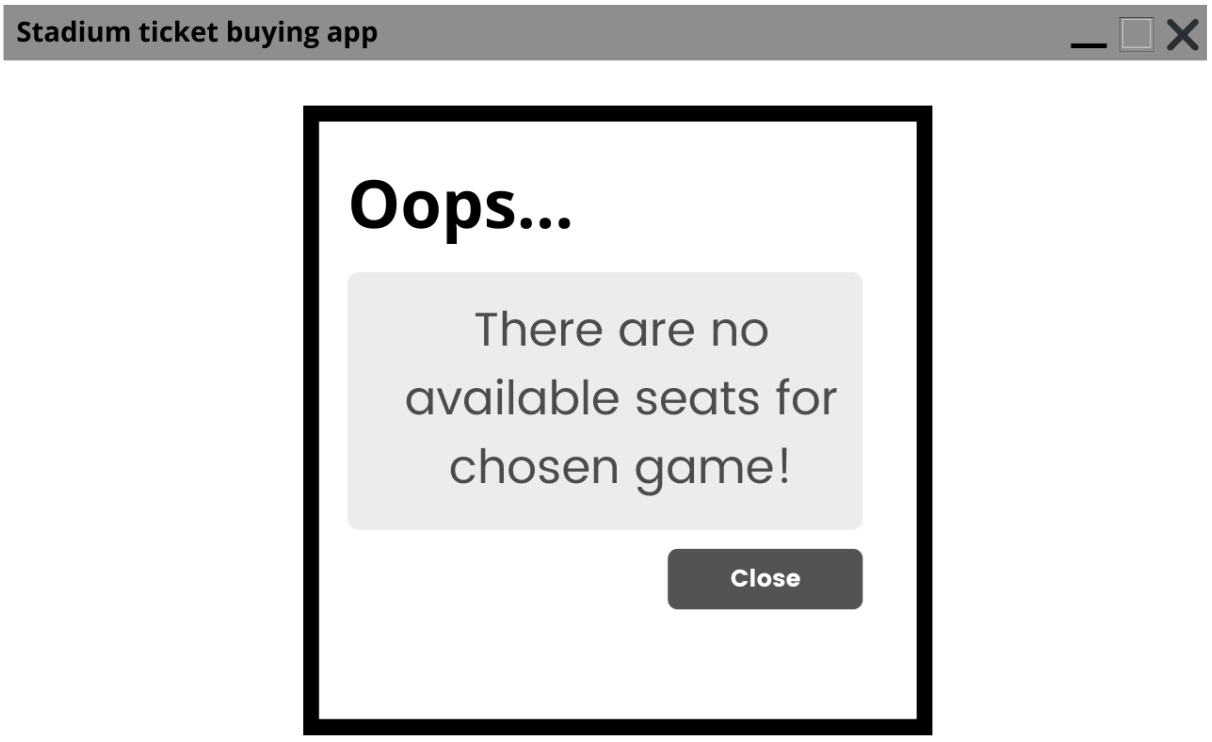
User selects session(stadium, date, time) from the list of sessions for the chosen game.



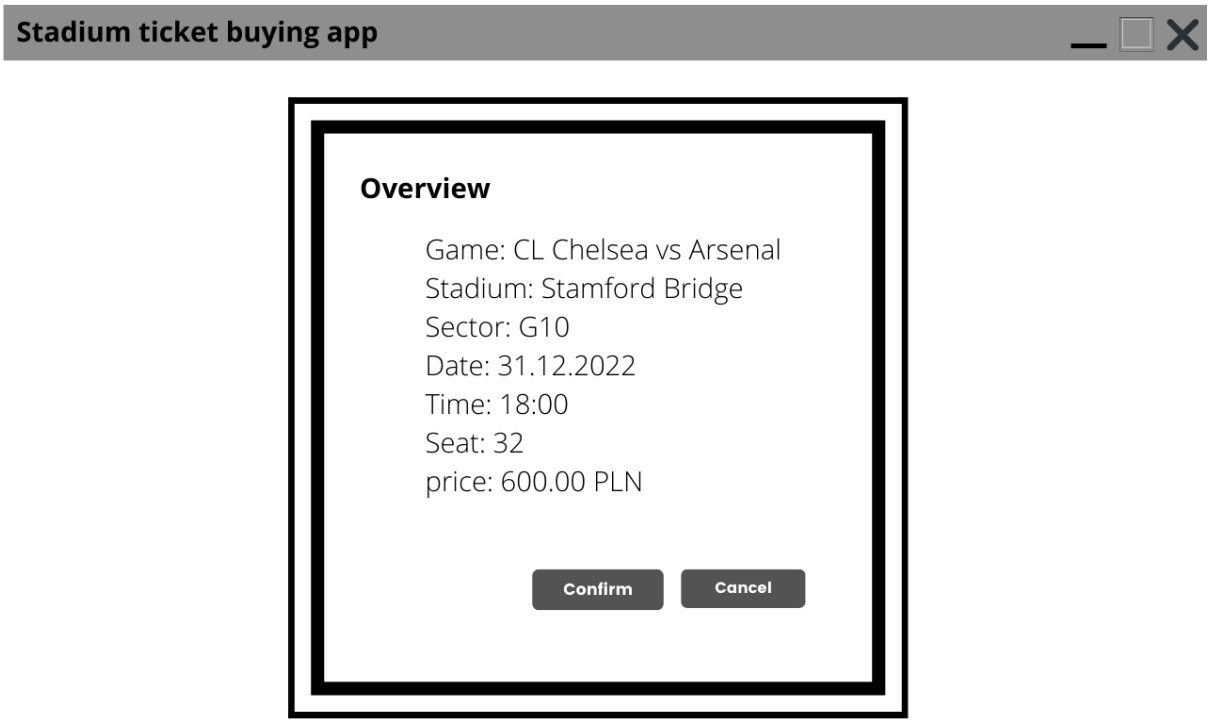
User selects a seat from the list of available seats for the chosen game.



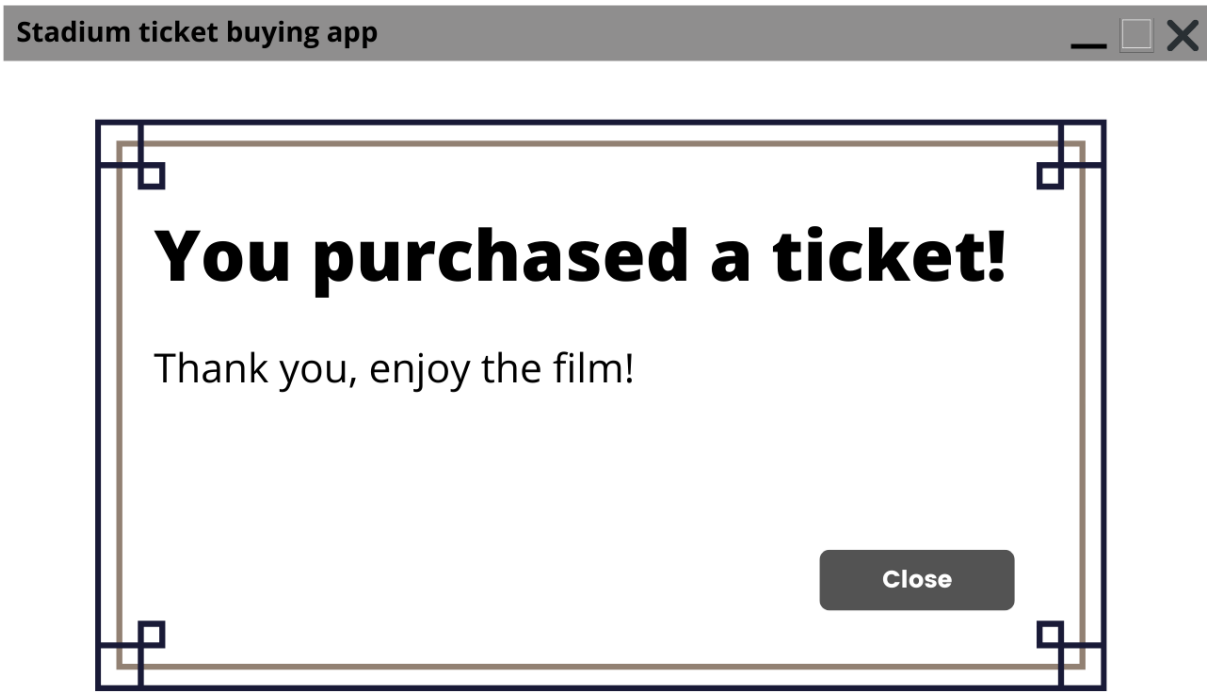
If there are no available seats for the chosen game, the error popup shows.



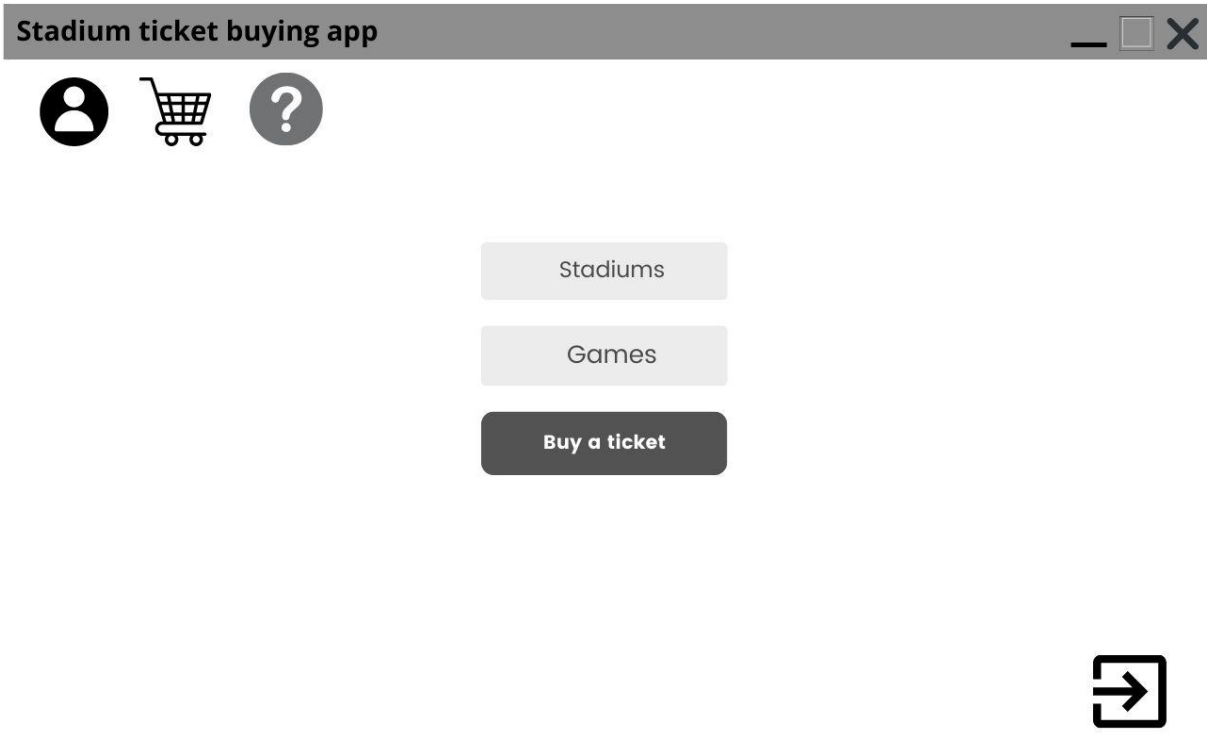
After selecting seat successfully, the user views the confirmation page with all data, then either confirms or cancels.



If the user confirms, the popup appears.



If user cancels, he returns to the main page.



The discussion of design decisions and the effect of dynamic analysis.

During the dynamic analysis, the following design decisions were made:

1. Adding new field ‘state’ into Ticket class, which will be represented by enum. There are 4 types of state: active, nonactive, canceled, archived. This decision was made after creating State diagram, where Ticket business class changes its state and this field is required in order to keep track of the current state of the ticket.
2. Adding 5 new methods into Ticket class. They are: Activate(), Cancel(), Nonactivate(), Archive(), Remove(). They will be helpful for changing current state of Ticket. This decision came from a State diagram. When a ticket is created it should be automatically activated by Activate() method in constructor. This means that the ticket is active and can be used to enter the stadium. There will be cases when the user decides to cancel his purchased ticket, in that case Cancel() method will change the current state of Ticket to ‘canceled’. If the ticket is canceled it should be removed, because the seat should be free again for another customer.

To remove ticket, I will use Remove() method. After game is finished the ticket is no longer active, Nonactivate() method will be invoked automatically by the system. After some time, ticket will be archived in the same way and kept in database for 3 years.
3. Adding CalculateFinalPrice() method into Ticket class, which returns double. In analytical diagram there is derived attribute /finalPrice in Ticket class. In implementation this derived attribute will become a method, which properly calculates the final price of the ticket considering the discount of the user and returns amount in double.
4. Adding new class Status. Status is a complex attribute of the User class.
It contains the name:String and discount:int. It should represented as another class because there can be different statuses such as: Student, Retired, Child and each of statuses can have different percentage of discount.
5. Dynamic inheritance, where User can become a Member and NonMember, will be implemented as follows. The User class will implement two interfaces, they are: Member, Nonmember. Member interface has two public abstract methods: getMemberPoints():int, setMemberPoints(int). The User class will have a field ‘userType’ which will be represented by Enum. There are two user types: member, nonMember. By ‘userType’ field I will keep track whether the user currently member or not. The user class will have two new methods, they are: becomeMember(), becomeNonMember(). These two methods will switch the user type to another.