Short description about each actor

* An ordinary (or regular) user is someone who employs the movie theater ticket reservation app to search for a movie, select a specific theater, view available showtimes and seats, select a seat, make their payment and receive tickets and confirmation, but does not have their information registered on the system’s database.
* A registered user is an ordinary user who has had their information saved on the system’s database and is therefore entitled to certain attendant perquisites contingent upon their payment of an annual account fee.
* A financial institution is a corporation that is responsible for processing and finalizing transactions related to the purchasing of movie tickets in the reservation app.
* A database engine is a system that interacts with the application in order to perform the actions needed to communicate data to and from the database.
* A manager is responsible for coordinating all of the activities occurring within the system of the movie theater ticket reservation app.

Use Case Scenario: Reserve Seats

This use case begins when the user has already selected a showtime for a chosen movie and theater combination. At this point, the system will display a graphical representation of the seating map of the theater room, showing the seats that have already been sold in one colour and those that are still available in another. For the films that have not yet had a public announcement, registered users will be able to purchase up to 10% of seats on a first come, first serve basis. After this 10% of seats has been reserved, registered users will need to reserve their seats after announcement. For all users, if there are no more or not enough seats available, the system will inform a user that the show is sold out and encourage them to consider another film. Otherwise, the user will have the opportunity to reserve one or more specific seats or to simply press cancel to annul seat selection entirely. Any available seat selected by the user will be highlighted and if a user selects a seat that has already been reserved and confirmed, the system will display a message to the user and ask them to pick another seat. After the correct seats have been highlighted, the user will confirm their selection with the system. After the user confirms that these seats are the ones they want to reserve, the system will immediately ensure that said seats may no longer be accessed by future users. The user will then be informed that they have successfully selected their preferred seat(s) and can move forward with the process. The database engine will craft an update with this new information and both the available seat count and images displayed in the app will change accordingly. After this point, the use case ‘Purchase Tickets’ will become available to the user.

Use Case Scenario: Purchase Tickets

After a user has chosen the seats to reserve for their performance, and proceeded to checkout, this use case commences. First, the system will show the user a summary of the ticket costs including taxes. The system will also inform the user at this point that, if they so wish, they may now apply any existing coupon codes or points towards the purchase price. If the user opts to do so, they will need to enter in the relevant information and apply it here. The system will accept this and modify the summary accordingly. If there is still an outstanding balance, the system will then prompt user to proceed to selecting a method of payment. Next, the system will determine which type of user it is dealing with. If a user is registered, they will be given the option to skip immediately to final confirmation of their booking details. An ordinary user will at this point be asked by the system to select their financial institution of choice, enter relevant personal information such as name, address, and email and to enter credit card number, expiry date, and security code (if credit card was chosen as form of payment) or other pertinent payment information. The system will communicate with the appropriate institution to establish the validity of the given information to accept payment. Then, for both ordinary, and registered users, the payment will be submitted. After the payment has been submitted, the system will confirm the completion of the payment process within the application and end the interaction with the user, shuttling them back to the application’s landing page. The system will update the database with the concluded purchase’s information through the database engine. It will also update the database regarding the used codes/points. The ‘Send Receipt’ use case will follow.

Use Case Scenario: Manage Annual Fee

When a user registers, they will be prompted by the system to make their first $20.00 annual fee payment to complete their registration. The system, which should now be keeping track of the registered user’s information, will from this point forward annually contact the user via their email to the remind them of their payment and ask that they make it before the deadline. This email will be sent one month before the registration anniversary. If the user makes their payment on time, the system will confirm that the fee has been paid and the user’s registration will remain in the system. If a user fails to pay their annual fee, the system will remove them from the list of registered users, and they will have to redo the process if they would like to rejoin.

Good-Candidate Objects: Reserve Seats

User, Movie, Showtime, Theater, Seat, Ticket, Map, Pattern, Message

Good-Candidate Objects: Purchase Tickets

User-information, Card-information, Payment, Coupon, Summary

Potential Operations: Reserve Seats

Display-availability, Select-seat, Cancel-seat-selection, Confirm-seat-reservation, Check-announcement-status, Inform-sold-out

Potential Operations: Purchase Tickets

Submit-payment, Confirm-payment, Apply-code, Present-summary, Choose-payment-method, Check-payment

Good-Candidate Objects: Manage Annual Fee

Reminder, RegisteredUser

Potential Operations: Manage Annual Fee

Check-Fee-Status, CheckDeadline, Email-reminder