Theater Ticketing System Design Specification

Prepared by Aeron Flores, Jacob Lorenzo, Jayce Ferris, Amarachi Duru, and Jeffrey DeOcampo

March 1, 2023

Overview

The theatre ticketing system is meant to work with the customer so that they may choose the movie they want to watch, the time they want to watch, and then pay for the movie. In this part of the process, the user must enter personal information required to make a transaction. One key feature of the software would be its ability to schedule movie showtimes and assign theater rooms. This would involve configuring the system to allow for different movie lengths, trailer times, and scheduling restrictions, avoiding overlaps between certain films. The software would need to ensure that the schedule is optimized for maximum profitability and that the necessary staffing is in place. Another essential aspect of the software would be the management of ticket sales. The system would need to be able to handle multiple ticket types, including discounts for seniors, children, and students, as well as group bookings. It would need to be integrated with the theater's website and mobile app to allow customers to book and pay for tickets online, and to generate e-tickets for scanning at the theater. The software would also need to handle the management of concessions, including inventory control and sales tracking. It would need to allow for the creation of menus and pricing structures, and to integrate with payment systems to process cash, credit, and debit transactions. Finally, the software would need to provide reporting and analytics functionality to help theater management make data-driven decisions. This would involve generating reports on ticket sales, concession sales, staffing costs, and customer behavior. The system could also provide insights into movie performance, allowing theater operators to adjust schedules and pricing to optimize revenue.

Software Architecture Overview

Architectural diagram of all major components

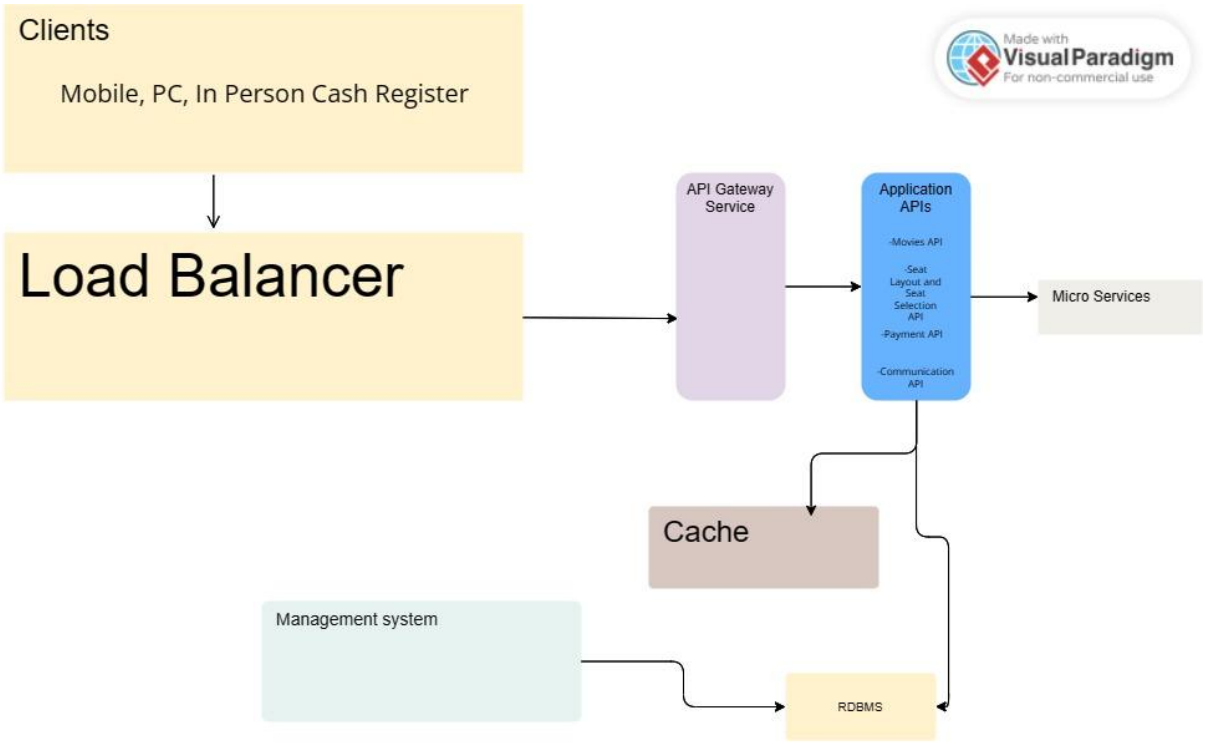
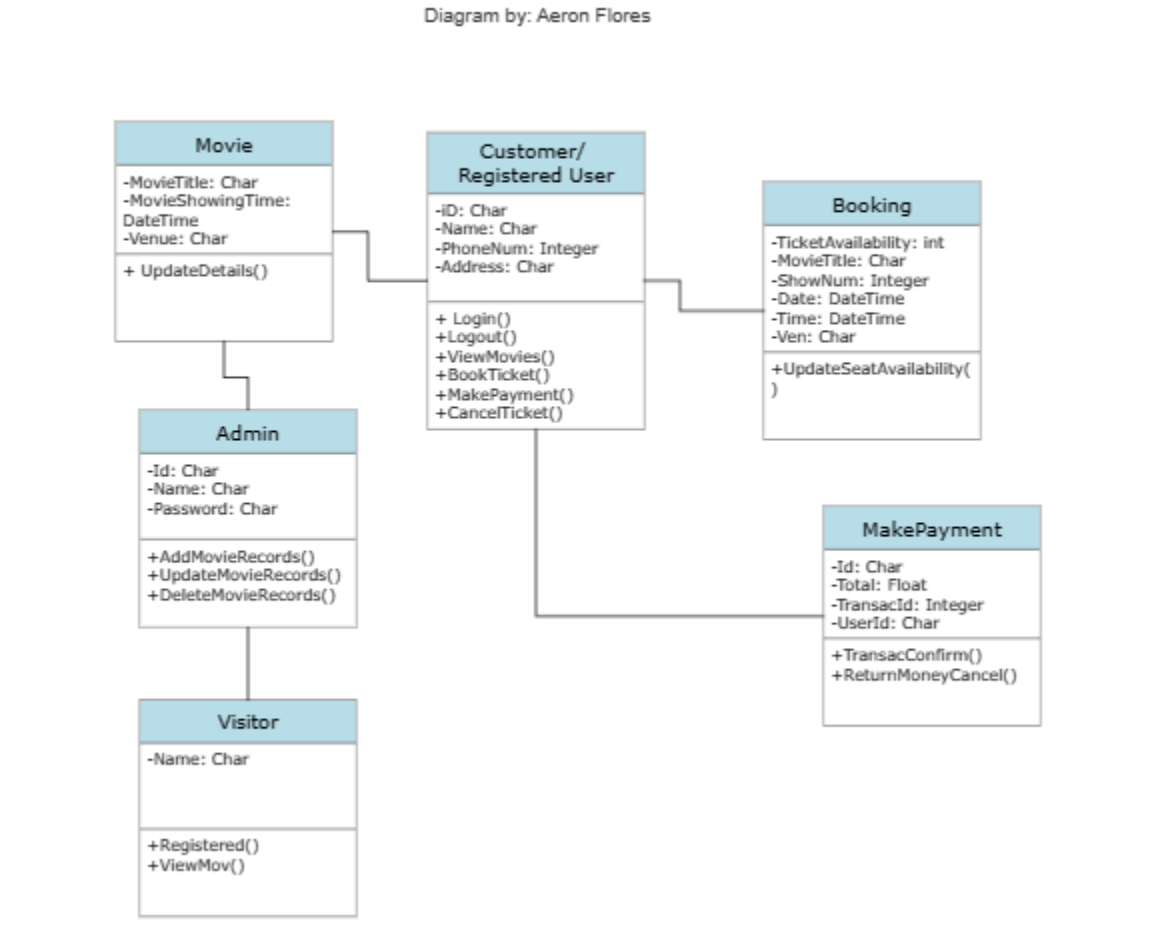


Diagram by: Aeron Flores

UML Class Diagram



Description of classes: (Jayce Ferris)

Movie:

Used in order to organize and manage the available movies, including all information associated with each movie including name, time, and location. Used by theater admin in order to create and update movie information.

Description of Attributes:

- MovieTitle
    - Type: Char
    - Use: Identifies the title of the movie in order for users and admin to keep track of the various films available at a give time.
- MovieShowingTime
    - Type: DateTime

    - Use: Specifies the individual screenings of each title
- Venue
    - Type: Char
    - Use: Identifies the location of the given film

Description of Operations/Functions Available in Movie Class: (Amara Duru)

- UpdateDetails()
    - Function: allows the updating of information; public void function
    - Precondition: values that are needed to change movieTitle, movieShowingTime, and venue can be accessed by this function.
    - Postcondition: movieTitle, movieShowingTime, and venue all contain values.

Admin:

Allows theater staff to manipulate movie class in order to keep up to date information about what movies are available at each time and location.

Description of Attributes:

- ID
    - Type: Char
    - Use: Helps organize and keep track of which admin changes are being completed by what employee/admin.
- Name
    - Type: Char
    - Use: Identifies an admin/employee by their name in order to more easily recognize who is making changes
- Password
    - Type: Char
    - Use: Secures admin accounts in order to ensure no unauthorized access by allowing for a sign in process.

Description of Operations/Functions Available in Admin Class:

- AddMovieRecords()
    - Function: adds movie records; public void function
    - Precondition: movieTitle, movieShowingTime, and venue all contain values (char)
    - Postcondition: new movie record has been added
- UpdateMovieRecords() and DeleteMovieRecords()
    - Function: updates or deletes movie records; public void functions
    - Precondition: AddMovieRecords() has been instantiated

    - Postcondition: movie record has been updated or deleted

Customer/Registered User:

Organizes customer and user base including personal information about each customer in order to create a personalized experience. Will allow user to access an account and purchase/manage movie viewings as well as view information about movies.

Description of Attributes:

- ID
    - Type: Char
    - Use: Identifies each individual user by a unique ID
- Name
    - Type: Char
    - Use: Identifies user by their name in order to track what products they are purchasing and to have a way to track by name.
- PhoneNum
    - Type: Integer
    - Use: Allows for contact to be made between employee and user in case of issues.
- Address
    - Type: Char
    - Use: Allows for personalized experience including the assistance in locating nearest theater.

Description of Operations/Functions Available in Customer Class:

- Login() and LogOut()
    - Function: allows user to log in or log out; public void functions
    - Precondition: to logout, Login() must be instantiated
    - Postcondition: user has either logged in or logged out
- ViewMovies()
    - Function: allows user to view movies; public void function
    - Precondition: movieTitle (char), venue (char), and movieShowingTime must contain values
    - Postcondition: user is able to view movie
- BookTicket() and CancelTicket()
    - Function: allows user to book or cance tickets; public void functions
    - Precondition: movieTitle, venue, and movieShowingTime must contain values
    - Postcondition: user has booked or canceled ticket
- MakePayment()

    - Function: allows user to make a payment; public void function
    - name, id (char), phoneNum (integer), and address (char) all contain values
    - Postcondition: user has made a payment

Booking:

Deals with the process of purchasing and managing tickets for individual movie viewings. Automatically manages how many tickets are left for each individual viewing in order to organize what movies are available at what time in each location. Mainly information based with minimal intervention from admin or user.

Description of Attributes:

- TicketAvailability
    - Type: Integer
    - Use: Tracks the availability of a given screening for various purposes including revenue tracking, employee needs, as well as avoiding the overselling of a screening
- MovieTitle
    - Type: Char
    - Use: Associates the title of the movie with the other various information
- ShowNum
    - Type: Integer
    - Use: Identifies the precise screening the availability is for.
- Date
    - Type: DateTime
    - Use: Identifies the precise date of the screening the availability is for.
- Time
    - Type: DateTime
    - Use: Identifies the precise time of the screening the availability is for.
- Ven
    - Type: Char
    - Use: Identifies the precise location the screening is for.

Description of Operations/Functions Available in Booking Class:

- UpdateSeatAvailability()
    - Function: updates the number of seats available by subtracting from or adding to ticketAvailability (int); public void function
    - Precondition: All Booking variables must contain values

    - Postcondition: ticketAvailability has been changed, and the number of available seats has been updated

Visitor:

Deals with non-registered users, allowing them access to the system in order to become registered or to view a movie.

Description of Attributes:

- Name
    - Type: Char
    - Use: Allows for system to have some type of identification of a user even if they are unregistered.

Description of Operations/Functions Available in Visitor Class:

- Registered()
    - Function: displays whether or not a user is registered; public void function
    - Precondition: user is not registered
    - Postcondition: allows user to enter values and create an account, user is registered and information can be inputted into Login() in Customer class
- ViewMov()
    - Function: checks Registered() before allowing a user to view a movie; public void function
    - Precondition: user has accessed Registered(), all variables in Movie class contain values
    - Postcondition: user is able to view movie

MakePayment:

Deals with all transactions between system and user. Allows user to purchase tickets for movies as well as to cancel any unwanted transactions.

Description of Attributes:

- ID
    - Type: Char
    - Use: Allows for each individual transaction to identify who is creating the transaction.
- Total
    - Type: Float
    - Use: Communicates to the user how much is owed.
- TransacID

    - Type: Integer
    - Use: Allows for each individual transaction to have an identifier
- UserId
    - Type: Char
    - Use: Tracks which user is making a transaction

Description of Operations/Functions Available in MakePayment Class:

- TransacConfirmed():
    - Function: sends confirmation message to user after a transaction has occurred; public void function
    - Precondition: user must be registered, all variables in Customer class should contain a value.
    - Postcondition: transacID (int) contains a value
- ReturnMoneyCancel():
    - Function: Cancels transaction (updates transacID to a new integer) and returns money to user; public void function
    - Precondition: TransacConfirmed() has been previously called, transacID (int) contains a value
    - Postcondition: transacID has been updated, user has had money returned to them

<u>Development plan and timeline</u> (Jacob)

- If everything goes according to plan then we should be able to complete the project after a few months. Over these few months, the team will create the software, test it, and make changes based on feedback. There will be multiple consultations with the customer throughout the timeline. The team consists of developers, designers, testers, managers, and specialists. The software team will be responsible for creating everything front end and back end. Front end development primarily focuses on what you see on the screen via visual elements. Back end development is made up of the data and logistical side of everything. The designers work with the developers to make everything look ergonomic and functional. Managers and project executives are there to maintain a steady workflow and enforce deadlines where need be. They also have the job of creating a workspace with flexibility and transparency so that there is an efficient use of time. The feedback team must work closely together with the development team and the customer so that we can produce a software that they want. Lastly, specialists will work in their areas of expertise aiding in all the the processed throughout the project timeline.