<Fandango>

Robert Ashe

Jaden Perleoni

Matthew Press

Mohammed Almousawi

System Description:

The Movie Theater Ticketing System is a web-based application that is to be designed using a three-tier architecture, made up of the Presentation Layer, Business Logic Layer, and Data Access Layer. Users will have various capabilities depending on their status as a customer, theater staff, or administrator.

The Presentation Layer will provide the user interface (UI) of the system. It is designed to be responsive to ensure compatibility with various devices like laptops, desktops, tablets, and smartphones. The UI will provide functionality for customers to browse movies, view showtimes, and purchase tickets.

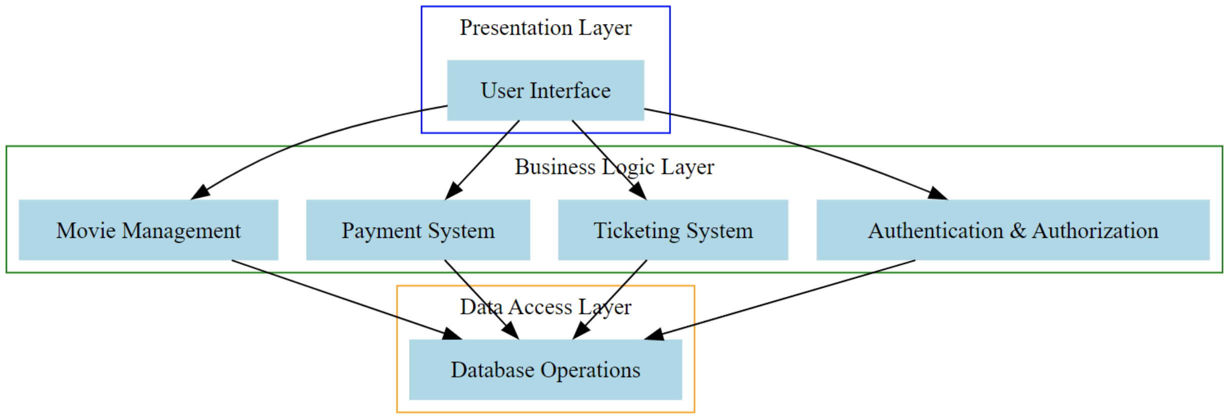
The Business Logic Layer will contain the core logic of the system. It manages operations such as user authentication, ticket purchasing, movie listing updates, and payment processing. The logic will ensure that customer data is handled securely and will comply with all current data integrity and security standards.

The Data Access Layer interacts with the database, handling all database operations. It ensures consistent and current data, and transaction history management.

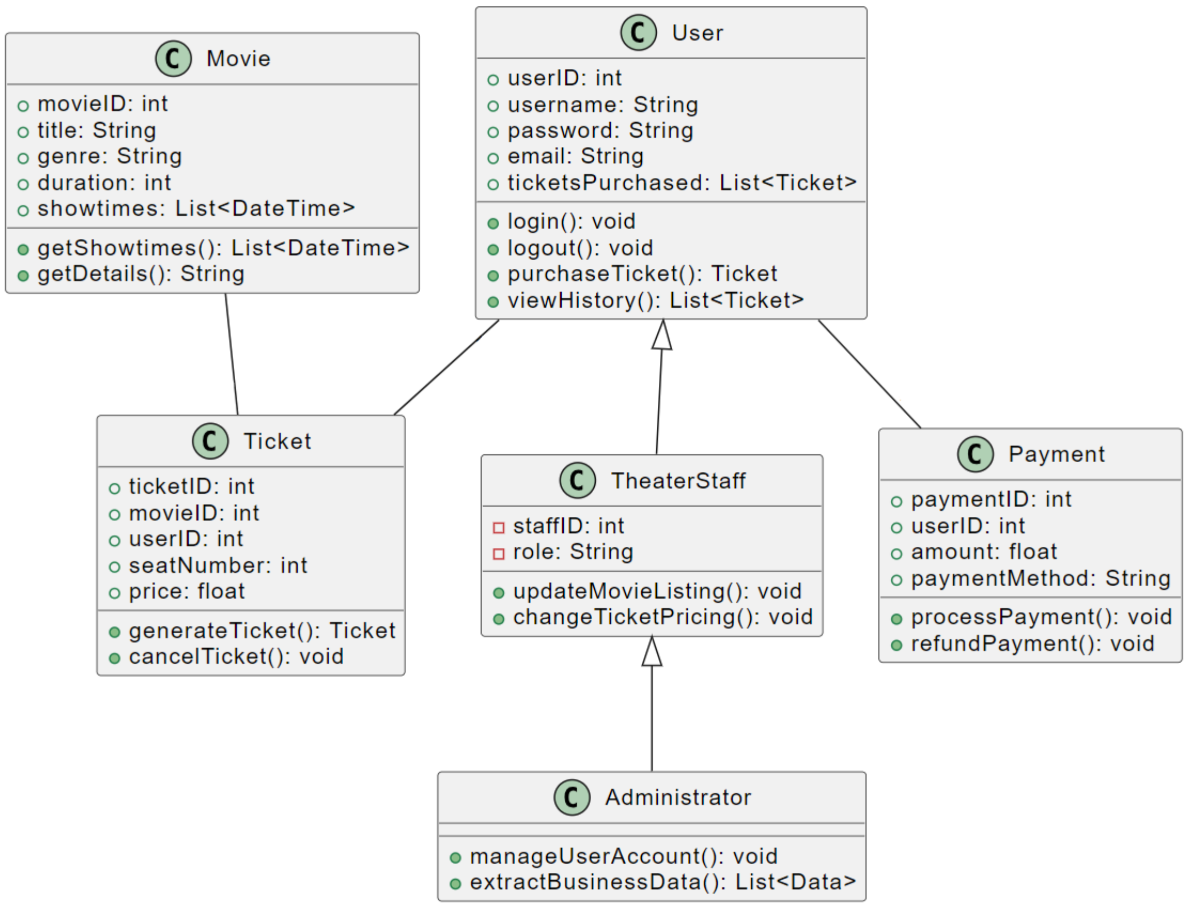
Software Architecture Overview:

<The architecture of the system is comprised of three layers, the Presentation Layer, Business Logic Layer, and Data Access Layer>

Architectural Diagram:



UML Diagram:



Description of Classes:

User: Represents the end-users of the system. Users can login and logout, purchase tickets, and view their purchase history.

TheaterStaff: Inherits from the User class and represents an employee of the theater. In addition to the capabilities of the User class, they can update the movie listings and change ticket pricing.

Administrator: Inherits from the TheaterStaff class and represents the system administrator. They have all the capabilities of the User and TheaterStaff classes, with the additional ability to extract business data. Administrators are also responsible for managing user accounts.

Movie: Represents the movies listed by the system. Each movie has details like title, genre, duration, and showtimes.

Ticket: Represents the tickets purchased by users. Each ticket is associated with a Movie and a User.

Payment: Represents the payment transactions in the system. Each payment is associated with a User and has a specific transaction amount.

Description of Attributes:

User:

- userID: Unique identifier for the user. Automatically generated by the Business Logic Layer. Integer type. Public.
- username: User's chosen name for login. Chosen by user in the Presentation Layer, verified to be unique by the Business Logic Layer. String type. Public.
- password: User's password for authentication. Chosen by the user in the Presentation Layer, verified by the Business Logic Layer to include at least one capital letter, one number, and one special character. String type. Public.
- email: User's email address. Entered by the user in the Presentation Layer, confirmed to be authentic by the Business Logic Layer. String type. Public.
- ticketsPurchased: List of tickets that the user has purchased. Maintained by the business logic layer and presented to the user by the Presentation Layer. List<Ticket> type. Public.

TheaterStaff:

- staffID: Unique identifier for the employee of the movie theater. Automatically generated by the Business Logic Layer. Integer type. Private.
- role: Specific role for the staff member (Cashier, Manager, etc.). Created by the Administrator in the Presentation Layer. String type. Private.

Administrator:

- Inherits attributes from TheaterStaff.

Movie:

- movieID: Unique identifier for the movie. Automatically generated by the Business Logic Layer. Integer type. Public.
- title: Name of the movie. Updated by the Business Logic Layer and sourced from the Data Access Layer. String type. Public.
- genre: Category of the movie (Action, Drama, etc.). Updated by the Business Logic Layer and sourced from the Data Access Layer. String type. Public.
- duration: Length of the movie in minutes. Updated by the Business Logic Layer and sourced from the Data Access Layer. Integer type. Public.
- showtimes: List of showtimes for the movie. Updated by the Business Logic Layer and sourced from the Data Access Layer. List<DateTime> type. Public.

Ticket:

- ticketID: Unique identifier for the ticket. Automatically generated by the Business Logic Layer. Integer type. Public.

- movieID: Unique identifier for the movie. Automatically generated by the Business Logic Layer. Integer type. Public.
- userID: Unique identifier for the user. Automatically generated by the Business Logic Layer. Integer type. Public.
- seatNumber: Seat number for the ticket. Automatically generated by the Business Logic Layer. Integer type. Public.
- price: Cost of the ticket. Updated by the Business Logic Layer. Float type. Public.

Payment:

- paymentID: Unique identifier for the payment. Automatically generated by the Business Logic Layer. Integer type. Public.
- userID: Unique identifier for the user. Automatically generated by the Business Logic Layer. Integer type. Public.
- amount: Amount of the transaction. Calculated by the Business Logic Layer. Float type. Public.
- paymentMethod: Method used to pay the transaction amount (Credit Card, Debit Card, etc.). Chosen by the user in the Presentation Layer and verified by the Business Logic Layer. String type. Public.

Description of Operations: < * descriptions should be detailed and specify datatypes, function interfaces, parameters, etc..>

User:

- login(username: String, password: String), Boolean: Allows the User to log into the system. Takes String parameters for username and password. Returns 'true' if the login was successful, 'false' otherwise.
- logout(), void: Allows the User to log out of the system. Does not return any value.
- purchaseTicket(movieID: Integer, showtime: DateTime, seatNumber: Integer), Ticket: Allows the user to purchase a ticket for a specific movie, showtime, and seat. Returns the purchased 'Ticket' object.
- viewHistory(), List<Ticket>: Allows the User to view their purchase history. Returns a list of 'Ticket' objects.

TheaterStaff:

- updateMovieListing(), Boolean: Allows the theater staff to update the movie listings using the Movie Listing Database. Returns 'true' if the listing was updated successfully, 'false' otherwise.
- changeTicketPricing(ticketID: Integer, newPrice: float), Boolean: Allows the staff to modify ticket pricing using the ticket's ID and a new price. Returns 'true' if the change was successful, 'false' otherwise.

Administrator:

- manageUserAccount(userID: Integer, action: String), Boolean: Allows the administrator to manage user accounts using the user's ID and a specific action (Delete User, Add User, Add TheaterStaff, etc.). Returns 'true' if the action was successful, 'false' otherwise.
- extractBusinessData(dateRange: DateTIme), List<Data>: Allows the administrator to extract business-related data for a specific date range. Returns a list of 'Data' objects.

Movie:

- getShowtimes(movieID: Integer), List<DateTime>: Retrieves the list of showtimes for a specific movie using the movie's ID. Returns a list of 'DateTime' objects.
- getDetails(movieID: Integer), String: Retrieves the detailed information about a specific movie using the movie's ID. Returns a String that contains the details related to the movie.

Ticket:

- generateTicket(userID: Integer, movieID: Integer, seatNumber: Integer), Ticket: Allows the User to purchase a ticket for a specific movie, showtime, and seat. Returns a 'Ticket' object that represents the purchased ticket.
- cancelTicket(ticketID: Integer), Boolean: Allows the User to cancel a purchased ticket using the ticket ID. Returns 'true' if the cancellation was successful, 'false' otherwise.

Payment:

- processPayment(userID: Integer, amount: Float, paymentMethod: String), Boolean: Processes the payment transaction for a specific User, amount, and payment method. Returns 'true' if the payment was successful, 'false' otherwise.
- refundPayment(paymentID: Integer), Boolean: Refunds a payment using the payment ID. Returns 'true' if the refund was successful, 'false' otherwise.

Development Plan and Timeline:

Task	**Description**	**Estimated Date**	**Team Member**
Create Movie class	Create the movie class and design its functions. Write logic for the getShowTImes() and cancelTicket() functions. The class must also define variables movieID, title, genre, duration, and showtimes.	10/18/2023	Robert Ashe

Create TheaterStaff class	Create a class for the theater staff and its functions. Define variables staffID and role. Write logic for functions: updateMovieListing() and changeTicketPricing().	11/4/2023	Matthew Press
Create User class	Create the user class and its functions. Variables userId, username, password, email, and ticketsPurchased must be defined in this class. Write the logic for the login(), logout(), purchaseTicket(), and viewHistory() functions.	10/31/2023	Jaden Perleoni
Create Administrator class	Create the administrator class and its two functions: manageUserAccount() and extractBusinessData().	10/16/2023	Mohammed Almousawi
Create Ticket class	Create the movie class, its variables, and its functions. Define variables movieId, title, genre, duration, and showtimes. Design the functions getShowtimes() and getDetails().	11/09/2023	Matthew Press
Create Payment class	Create the payment class, its variables, and its functions. Define variables paymentID, userID, amount, and paymentMethod. Write the logic for functions processPayment() and refundPayment().	11/01/2023	Jaden Perleoni