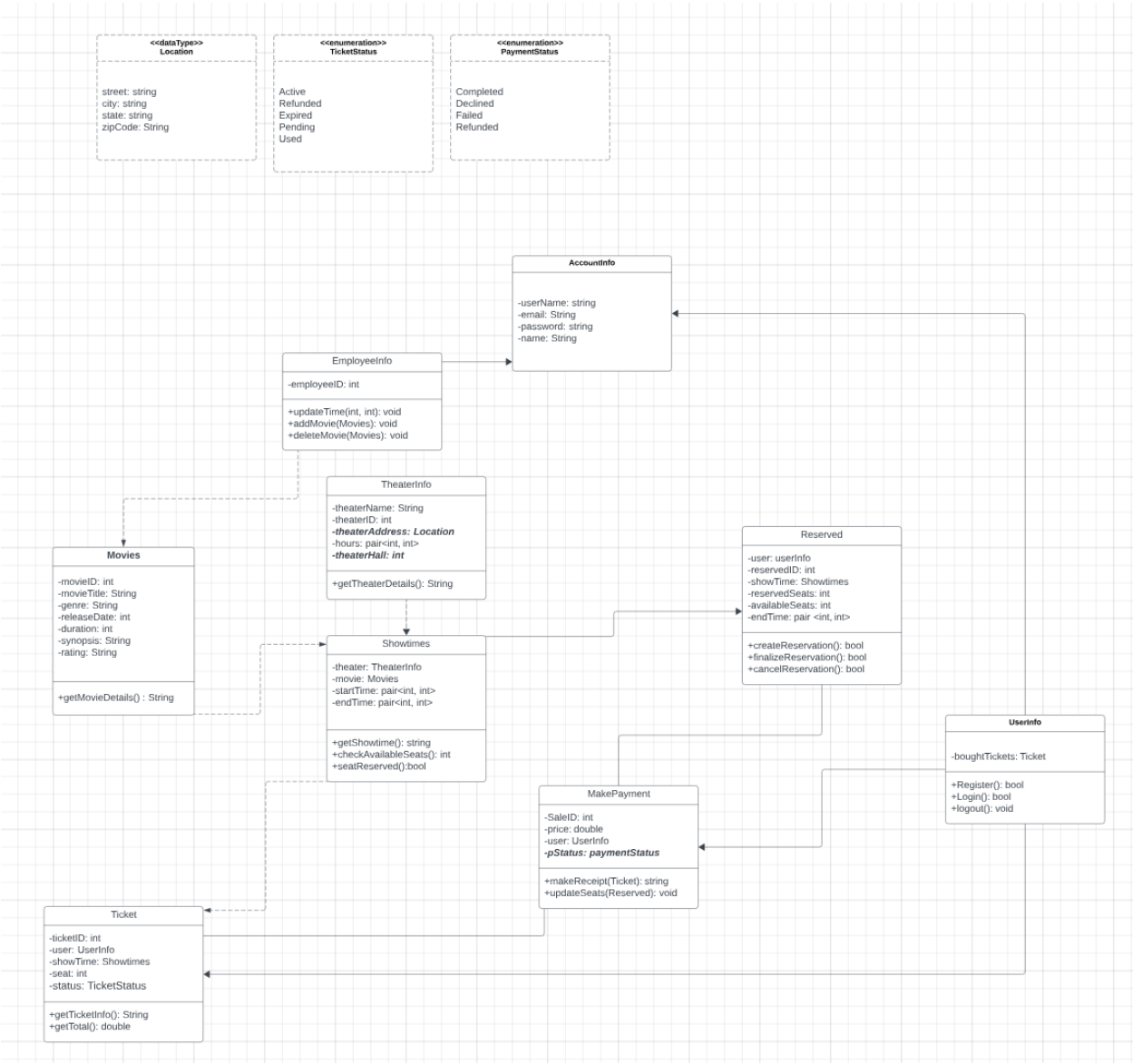
“Swift Wave”
Movie Ticketing System

Group: 9
Omar Abdirahman
Brian Nguyen
Riley Phan

System Description:

SWIFT WAVE is a state-of-the-art, user-friendly platform made to improve and streamline moviegoing for patrons while giving theater operators effective management capabilities. By combining several capabilities into a single, seamless digital solution, this technology seeks to fundamentally alter how moviegoers choose, reserve, and attend film screenings.

UML:



Movies

Description:

The "Movies" feature in our system uniquely identifies each film with attributes such as its title, genre, release date, duration, brief synopsis, and suitability rating. Additionally, the getMovieDetails() function ensures users can swiftly access a comprehensive view of their selected movie.

Attributes:

- movieID(int): Serves as the unique identifier for every film. The integer ensures that each movie has a distinct reference.
- movieTitle (String): Represents the name or title of the movie. It's a string type that can accommodate varying lengths of movie names, ensuring flexibility.
- Genre (String): Categorizes the film into specific genres for easy filtering and searching. This string attribute allows the filtering to be easy for users who want to search by filtering genre.
- releaseDate (int): Specifies when the movie was or will be released to the public.
- Duration (int): Represents the total runtime of the movie in minutes. This enables the users to view the duration and length of the film beforehand.
- Synopsis (String): Provides a concise overview or summary of the movie's plot.
- Rating (String): Indicates the audience suitability based on standard rating systems.

Functionality:

- The getMovieDetails() function is tailored to retrieve and present all these attributes, offering users a comprehensive view of their movie selection.

TheaterInformation

Description:

The TheaterInfo class centralizes key details about individual theaters associated with the movie ticketing system. It catalogs information about a theater's identity, location, and operational timings, ensuring users and system administrators have a comprehensive understanding of each venue.

Attributes:

- theaterName (String): Denotes the official name or title of the theater, helping users and administrators quickly identify each venue.

- theaterID (int): Serves as a unique identifier for every theater. As an integer value, it ensures that each theater has its distinct reference within the system.

- theaterAddress (Location): Represents the theater's physical location using the Location datatype. Such attributes as a street address, city, state, or postal code may be included in this custom type. Users need to be able to integrate directions or a map while organizing their visit.

- hours (pair<int, int>): Represents the theater's operating timings. This pair of integers typically captures the opening and closing times. Including informing users about when they can visit.

- gives the cinema or movie theater complex's hall number or identification. This helps users locate the specific screening hall once they arrive at the theater complex.

Functionality:

getTheaterDetails(): String: This function's purpose is to retrieve and display all of a theater's attributes. When called, it gathers information such as the name, ID, location, and operating hours of the theater and delivers it in a structured string for easier consumption.

Showtimes

Description:

The Showtimes class represents specific showtimes for a movie at a theater. It includes details about the movie being shown, such as the theater location, title, start time, end time, and available seats.

Attributes:

- Theater: A reference to the theater where the movie is being shown at.
- Movie (Movie): A reference to the movie object being shown during this showtime.
- startTime: The date and time when the movie starts.
- endTime: The date and time when the movie ends.

Functionality:

- getShowtime(): String: This function retrieves and returns specific information about the showtime, such as the theater, the film, the start time, the end time, and the number of seats that are still available.
- checkAvailableSeats(): Int: This function returns the number of seats that can be reserved for a movie.
- seatReserved(): bool: This function is used to show that a seat has been reserved for this movie returning true or false.

Ticket

Description:

Represents ticket information for a specific showtime purchased by a user such as who bought it, ticket identification number, movie's showtime, seat number, and price.

Attributes:

- ticketID (int): Serves as the unique identifier for each ticket. Which creates a special number for the ticket.

- user (UserInfo): Contains the details of the user who has purchased or reserved the ticket.

showTime (Showtimes): When and where the movie plays.

- seat (int): Indicates the assigned seat number for the ticket, providing the user a specific spot in the theater.

- status (TicketStatus): Specifies the current state of the ticket if ticket is paid fully leading to official confirmation as proof.

Functionality:

- getTicketInfo() String: This function retrieves detailed information about the ticket, such as the user's details, showtime and seat number. It then returns this information in a structured string format for easy reference and display.

- getTotal() double: Calculates the user's total ticket cost for a movie including fees and taxes while taking any discounts or special offers into account.

MakePayment

Description:

Represents a payment transaction for movie tickets made by a user.

Attributes:

- SaleID (int): Acts as the unique identifier for every sale, typically a sequential integer, ensuring

- that each sale has a distinct reference within the system.

- User (UserInfo): The user information who purchased the ticket.

- price (double): Represents the total amount paid for the ticket.

- pStatus (paymentStatus): Denotes the current status of the payment using the PaymentStatus enumeration. (Completed, Declined, Failed, Refunded)

Functionality:

- makeReceipt(Ticket): string: Creates a receipt for the user with their information about their transaction and the associated purchased ticket. Taking in a Ticket object, it will use data from the ticket object to create the receipt.

- updateSeats(Reserved): void: Takes in the Reserved object, takes information from that class, and uses it to update the seating availability of the movie that the user just bought after a successful payment.

Reserved

Description:
Reserved Class represents a reservation made by a user for a certain showtime, including with the user's details, the seats they have reserved, and the reservation ID. Methods include making, completing, and canceling reservations and is intended to manage the user's reservations.

Attributes:

- reservedID (int): A unique identification number generated for the reservation.

- user (userInfo): The user information who made the reservation which includes their user details.

- showTime (Showtime): The showtime for which the reservation is made, providing details about the movie, theater location, date, and time.

- AvailableSeats (int): The number of seats that are available for a requested movie.

- reservedSeats (int): The number of seats reserved by the user.

- endTime (pair<int, int>): Two integers that represent the reservation's end time and often represent the beginning and ending timestamps of the reservation's time.

Functionality:

- createReservation(): bool: Creates a reservation for the user for the chosen showtime and amount of seats; returns true if successful, false otherwise.

- finalizeReservation(): bool: This usually happens after the user has finished the payment procedure. When it finalizes the reservation and returns true if it was successful or false if it wasn't.

- cancelReservation(): bool: Returns true if the reservation is successfully completed or false if it is not, canceling the reservation and making the reserved seats available to other customers.

AccountInfo

Description:

The AccountInfo is the superclass of both EmployeeInfo and UserInfo

Attributes:

- name (String): Captures the full name of the employee/User. As a string type, it offers the flexibility to accommodate both short and long names.

- email (String): Records the official email address of the employee/User, serving as a primary point of contact and often used for system notifications and communications.

- password (String): Stores a secure, encrypted version of the User/employee's password, crucial for login and authentication purposes.

- userName (String): Stores the username of an employee or user.

EmployeeInfo

Description:

The EmployeeInfo subclass contains crucial information about employees with access privileges to the backend of the movie ticketing system. It not only keeps a worker's essential personal information, but it also has features for managing movies inside the system.

attributes:

- employeeID (int): Acts as the unique identifier for every employee, typically a sequential integer, ensuring that each staff member has a distinct reference within the system.

Functionality:

- updateTime(int, int): void: Enables employees to modify movie showtimes. It typically accepts parameters such as the movie's ID and the updated time, making necessary adjustments in the system.

- addMovie(Movies): void: Provides employees the capability to introduce new movies to the system. It takes a Movies object as a parameter and appends it to the movie database.

- deleteMovie(Movies): void: Allows employees to remove existing movies from the system. By accepting a Movies object, it identifies and deletes the corresponding movie record from the database.

UserInfo

Description:

The Ticket class represents the information associated with a ticket purchased or reserved by a user. It includes important information about the ticket, including the particular showtime, seat number, the user who made the purchase, and the unique ticket ID.

Attributes:

- boughtTickets (Ticket): A list of purchased tickets associated with the user.

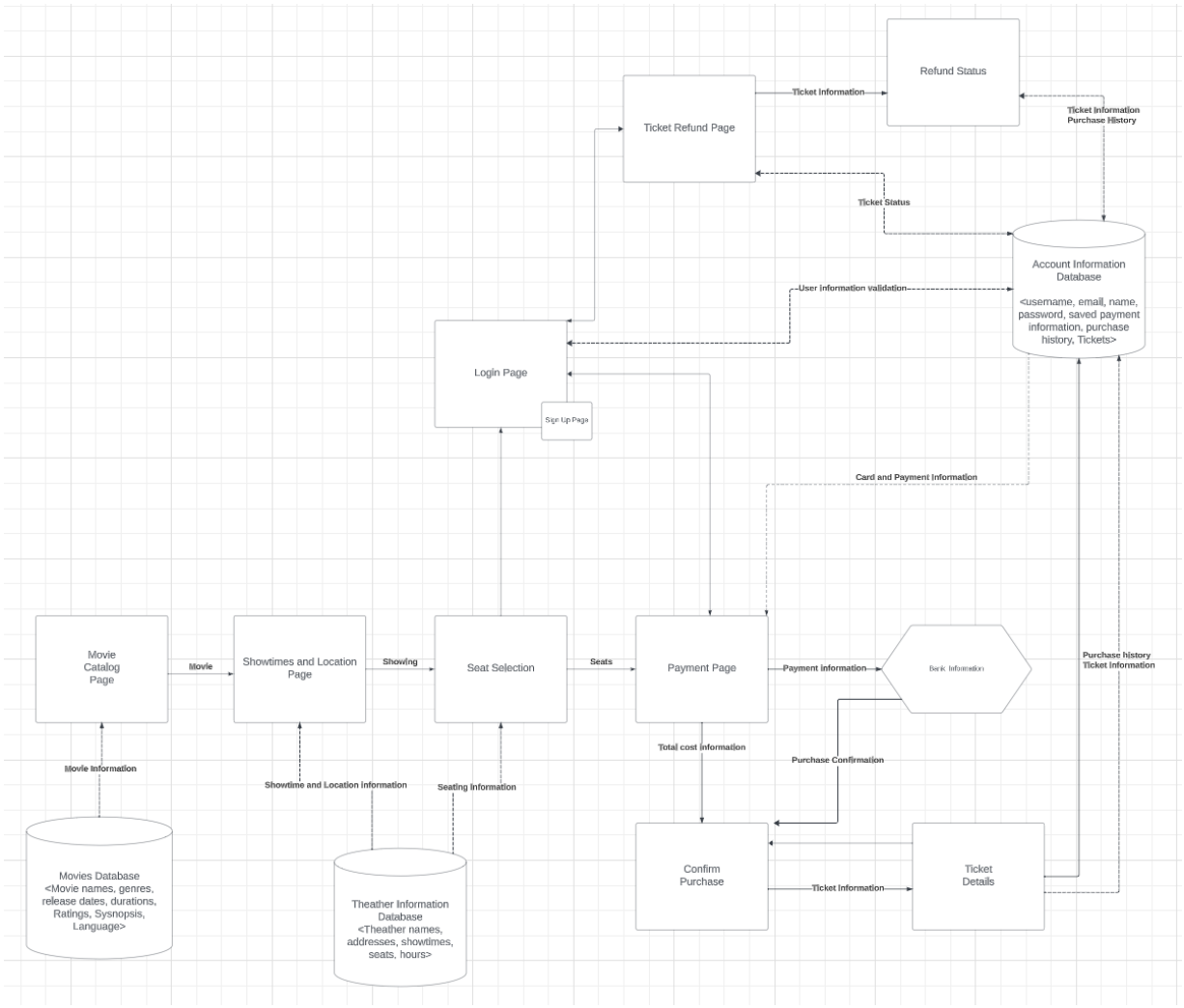
Operations:

- register(): bool: returns a new user's registration information for the system. Upon successful registration, return true; otherwise, false.

- login(): bool: compares the user's credentials (username and password) with previously saved user information to verify their identity. In case of successful login or not, return true or false.

- logout(): bool: returns a boolean after logging the user out of the system, if the logout process is successful, True; otherwise, False.

Software Architecture:



User Interface:

- Movie Catalog Page: It provides movie information, titles, synopses, and reviews. Clicking on a movie will take users to additional information or a booking page.
- Movies Page: Users can explore and choose a movie of interest from the list of movies on this website. Users can view movie details and proceed to select a movie for booking.
- Showtime and Location Page: It displays a calendar or timetable of the various theaters' open showtimes for the selected movie. With user interests and availability, users can choose a convenient showtime and venue.
- Seat Selection: Users often see a picture of the theater's layout with the seats that are still available highlighted. Users can check seating prices, pick the seats of their choice, and proceed with the reservation.

- Login Page: Users enter their credentials on the login page to be entered by the system. By comparing user credentials with the Accounts Database, it confirms user identification and provides authorized users access.
- Payment Page: Securely manages financial transactions. Users enter payment data (like credit card or debit card information), which the Payment Service uses to process payments by integrating Bank information.
- Confirm Purchase: Before completing the transaction, users can confirm their reservations or make any modifications. Users can check a summary of their booking information on the Confirm Purchase page.
- Ticket Details: Users can get information about their movie reservations/purchases, such as the title of the film, the showtimes, the location of their seats, and any other reservation information.
- Ticket Refund Page: Users that have a bought ticket that has not expired can refund their tickets here. It requires the user to be logged in. It uses the Accounts Database to retrieve information.
- Refund Status Page: Shows the current status of refund requests in the last 30 days. It can be accessed from the Ticket Refund Page.

Databases:

- Movie Catalog: a database for details on the movies that customers can view and reserve tickets. It contains data about each movie, such as titles, descriptions or synopsis, Genres, reviews, and trailers.
- Accounts Database: Maintains user information (such as names and email addresses), user credentials (such as usernames and passwords), and other user-related information. This database is used by the Login Page to handle user profiles and provide user authentication.
- Theater Info: It contains information about showtimes, seating, location information, and theater names, addresses, and phone numbers. The "Showtime and Location Page" and the "Seat Selection" sections of the movie ticketing system both rely on information from the "TheaterInfo" database. By offering seating, it helps users choose their best theater location and helps them choose seats. Users can choose showtimes conveniently thanks to the database's retrieval of theater-specific showtimes suited best for them.

Connectors:

- Users access the system through the Homepage.
- The Movie retrieves movie information from the Movie Catalog, displaying a list of available movies.
- Users can select a movie, leading to the Showtimes and Location Page.
- From the Movie Catalog Page, users can navigate to the Showtime and Location Page to select a showtime and theater location.

- Seat Selection allows users to choose their preferred seats,
- Users must log in via the Login Page to proceed with the reservation.
- User information validation is required from the database of Account Information to authenticate access.
- Users will have to use the Sign Up Page if not signed up.
- The Payment Page handles the payment process, integrating with the Bank information for transactions.
- After successful payment, users are directed to the Confirm Purchase page to review the purchase.
- Users can access their ticket details from the Ticket Details page.
- Users when logging in, can request the Ticket Refund Page to start a refund process.
- Ticket Refund Page can gather ticket information and start the refund process leading users to the Refund Status if users want to commit to a refund.
- User account information is stored and managed in the AccountInfo Database, ensuring secure authentication and personalization.

Development Plan and Timeline

Overview:

The objective is to create SWIFT WAVE, a user-friendly platform created to improve the experience of watching movies. The system will manage user/employee accounts and handle user/employee account management, movie details, theater information, showtimes, ticketing, reservations, and payment transactions.

Partitioning of Tasks:

The project has been split up into particular tasks, each of which focuses on a vital component of the system to speed up the development process. This division guarantees effective task distribution, prompt completion, and efficient resource use. Representation of each week will be shown including the main objective for the week, including with task, estimated time, and the member(s) who will be responsible for it.

Week 1-2: Initialize Classes and System setup/System initialization

Task:

- Set up databases and development environment
- Estimated Time: 2 Days

- Responsibility: Brian

Task:

- Implement ‘Movies’ and ‘TheaterInfo’ classes.
- Estimated Time: 1 week
- Responsibility: Omar

Week 3-4: User & Employee Account Management

Task:

- Develop AccountInfo, EmployeeInfo, and UserInfo classes with relevant functions (Register, Login, Logout, addMovie, etc.).
- Estimated Time: 2 weeks
- Responsibility: Riley

Week 5-6: Reservations and Showtimes

Task:

- Implement Showtimes class with functionalities for checking available seats and reserving seats.
- Estimated Time: 1 week
- Responsibility: Brian

Task:

- Develop the Reserved class to manage user reservations.
- Estimated Time: 1 week
- Responsibility: Omar

Week 7-8: Payment & Ticketing

Task:

- Design and implement the MakePayment class.
- Estimated Time: 1 week
- Responsibility: Riley

Task:

- Develop the ‘Ticket’ class for ticketing functionality.
- Estimated Time: 1 week
- Responsibility: Brian

Week 9-10: User Interface Design and System Integration

Task:

- Design interface prototypes and construct interactive elements.

- Estimated Time: 1 week
- Responsibility: Omar

Task:

- Merge interface elements with system functionality.
- Estimated Time: 1 week
- Responsibility: Riley

Task:

- Testing and debugging.
- Estimated Time: 1 week
- Responsibility: All team members

Week 11: Debugging, Testing, Review

Task:

- Thorough testing and debugging
- Estimated Time: 1 week
- Responsibility: All Members

Task:

- Evaluation phase to monitor progress and make necessary adjustments.
- Estimated Time: 3 days
- Responsibility: All members