<Theater Ticketing System>

Software Design Specification

<2/27/2025>

<Group 7>
<Anthony Nguyen, Steven To, Charlie Pham>

Prepared for
CS 250- Introduction to Software Systems
Instructor: Gus Hanna, Ph.D.
Fall 2023

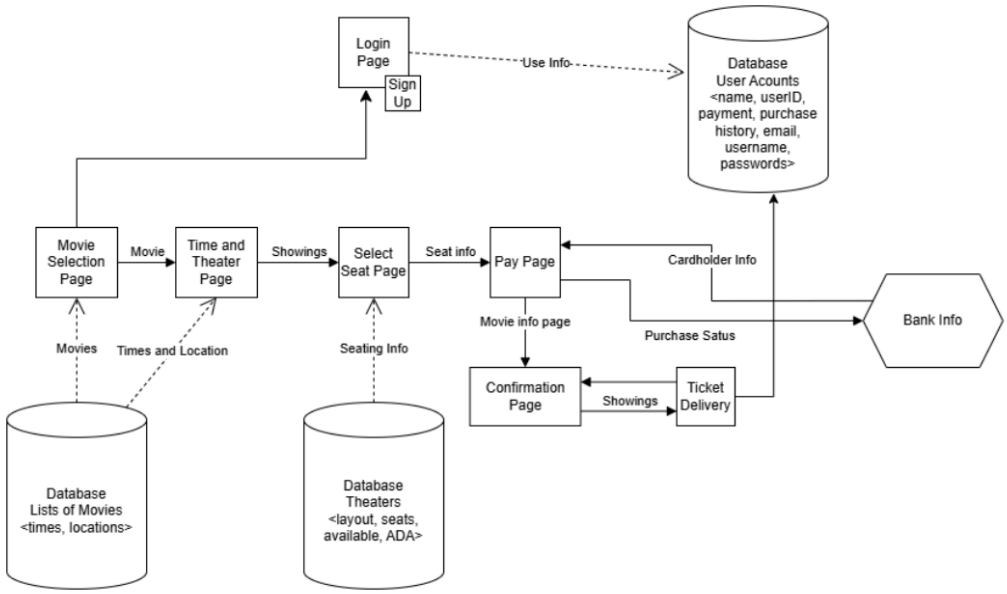
System Description

1. System Description

The theater ticketing system is a web-based application that allows users to browse moving screenings, select seats, and purchase digital tickets. The system offers real-time seat availability, secure payment integration, and automated notifications. It also includes administrative functionalities for managing movie schedules and ticket pricing. The system aims to enhance convenience for customers and optimize theater operations.

2. Software Architecture Overview

2.1 SWA Diagram

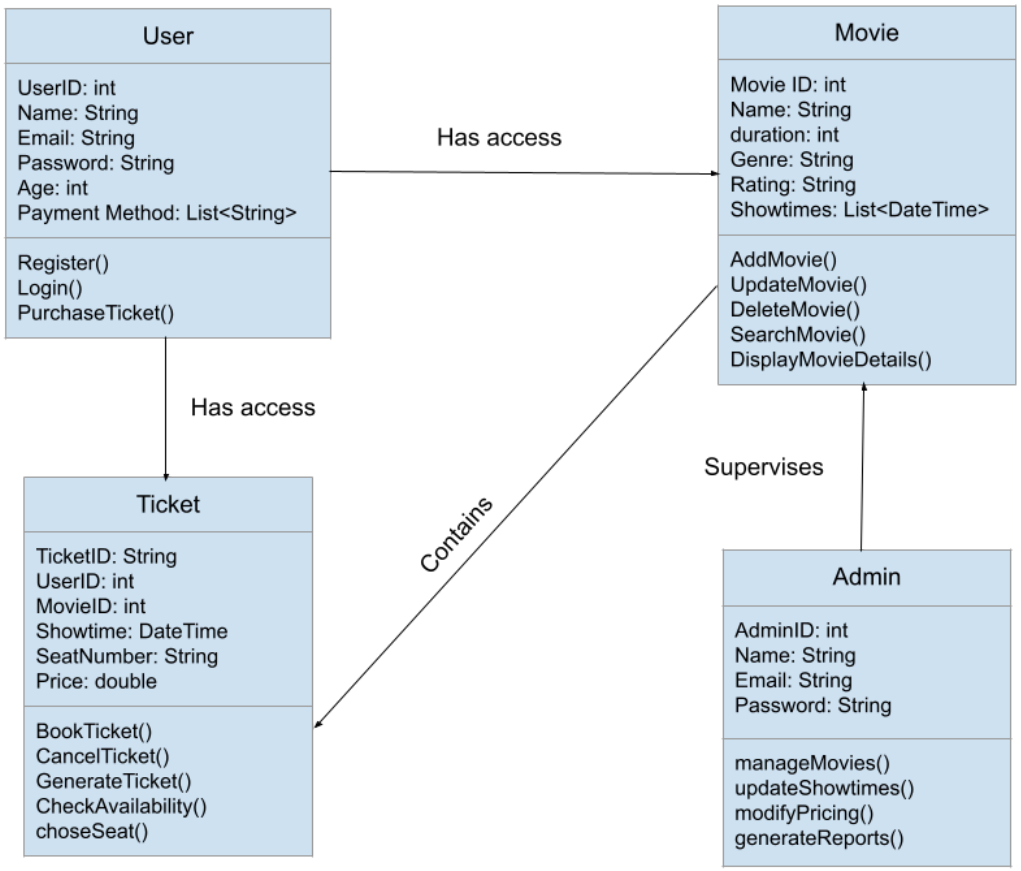


2.2 SWA Description

The user starts at the login page where they access their account, either regular customer or admin. This data from the database takes them to the movie selection page with certain attributes that correlate to the user type. If they get the email or password wrong, an error message will be displayed. At the bottom of the login page, there is also a “Forgot Password” link. If the user forgets their password, they can click the “Forgot Password” link and an email will be sent to them to change their password. Then once in, they are brought to the movie selection page. Here the user can search for a movie, or look through the genre list. After that, they are taken to the time and theater arrangement page. After selecting the time you want to watch the movie, you are taken to the seating arrangement page. Then after selecting your seat, you are taken to the payment page. Once a payment type is selected and paid for, the

system accesses the bank account and takes you to a confirmation page. From here you would confirm your purchase, and the receipt of the payment gets sent to the user account database. Your ticket gets delivered via email and also saved in the account database. With the email, you can then go in and enjoy your movie.

2.3 UML Class Diagram



2.4 Description of Classes

User

- Purpose: Represents an individual user who can register, log in, and purchase movie tickets.

Movie

- Purpose: Stores details of movies available in the system.

Ticket

- Purpose: Handles movie ticket information and seat reservations.

Admin

- Purpose: Manages the backend operations, including movie listings and reports.

2.5 Description of Attributes

- **User**
  - UserID: int
    - A unique identifier for each user.
  - Name: String
    - The full name of the user.
  - Email: String
    - The user's email address, and is used for account creation and communication.
  - Password: String
    - A secure password for account authentication.
  - Age: int
    - The user's age, and is used to enforce the minimum age requirement (14+).
  - PaymentMethods: (List<string>)
    - A list of payment methods associated with the user.
- **Movie**
  - MovieID: int
    - Unique identifier for the movie.
  - Title: String
    - Title of the movie.
  - Description: String
    - Brief synopsis of the movie.
  - Duration: int
    - Runtime of the movie in minutes.
  - Genre: String
    - Category of the movie (e.g., "Action", "Comedy").
  - Rating: String
    - Movie rating (e.g., "PG", "PG-13").
  - Showtimes: List<DateTime>
    - List of available showtimes.
- **Ticket**
  - TicketID: String
    - A unique identifier for each ticket.
  - UserID: int
    - The ID of the user who purchased the ticket.
  - MovieID: int
    - The ID of the movie associated with the ticket.
  - Showtime: DateTime
    - The date and time of the movie screening.
  - SeatNumber: String
    - The seat number assigned to the ticket.
  - Price: double

        - The price of the ticket is converted to the user's local currency.
    - choseSeat: String
        - The seat is selected.
- **Admin**
    - AdminID: int
        - Unique identifier for administrators.
    - Name: String
        - Name of the administrator.
    - Email: String
        - Email for authentication.
    - Password: int
        - A secure password for account authentication.

2.6 Description of Operations

- **User**
    - Register(name, email, password, age): void
        - Allows a new user to create an account.
    - Login(email, password): bool
        - Authenticates the user and grants access to the system.
    - PurchaseTicket(movieID, showtime, seatNumber): Ticket
        - Purchases a movie ticket.
- **Movie**
    - AddMovie(title, description, duration, genre, rating): void
        - Adds a new movie to the system.
    - UpdateMovie(movieID, title, description, duration, genre, rating): void
        - Updates movie details.
    - DeleteMovie(movieID): void
        - Removes a movie from the system.
    - SearchMovie(keyword): List<Movies>
        - Searches for movies by title, genre, or keyword.
    - DisplayMovieDetails(movieID): void
        - Movie Displays detailed information about a movie.
- **Ticket**
    - BookTicket(userID, movieID, showtime, seatNumber, price): Ticket
        - A user can book a ticket for a specific movie and showtime.
    - CancelTicket(ticketID): void
        - This enables a user to cancel a booked ticket.
    - GenerateTicket(ticketID): string
        - Generates a digital ticket for the user.
    - CheckAvailability(movieID, showtime): int
        - Check seat availability for a specific showtime.
    - ChoseSeat(SeatNumber): void
        - This allows the user to select their seat and seat number.
- **Admin**
    - manageMovies(): void

        - Adds, edits, and removes movie listings.
    - updateShowtimes(): void
        - Modifies showtimes for movies.
    - modifyPricing(): void
        - Adjusts ticket pricing.
    - generateReports(): List<Report>
        - Produces sales and user activity reports.

3. Development Plan and Timeline

3.1 Partitioning of Tasks

Task	Estimated Completion Date
System Architecture and Design	Week 1-2
Testing and Debugging	Week 2-3
Payment Integration	Week 3-4
Frontend Development and UI Design	Week 4-5
Database and Backend Development	Week 5-6
Documentation and Finalization	Week 6-7
User Authentication and Role Management	Week 7-8
Seat Selection and Booking System Development	Week 8-9
Notification and Confirmation System	Week 9-10

3.2 Team Member Responsibilities

	Member
Title Page	Steven To
System Overview	Charlie Pham
Architectural Diagram	Anthony Nguyen
UML Diagram	Steven To

Classes Description	Charlie Pham
Attributes Description	Anthony Nguyen
Operations Description	Steven To

4. Verification Test Plan

UNIT TEST

Test Set 1: login(): boolean - Method

Feature Targeted: The *login()* method is within the user class. This method takes in two parameters: the email address and password. This checks for a user's account and authenticates the user hence granting them access to the system.

Explanation: This test validates the correctness of the *login()* method by checking for accurate inputs for the email and password. This method is split into two partitions, When the correct credentials are inserted, and when the wrong credentials are entered.

1. The correct credentials
2. The incorrect credentials

Test Cases:

- TC001 Correct Credentials
    - Preconditions: The user has an active account
    - Expected Results: The user should be redirected to the dashboard
    - Actual Results: The user is redirected to the dashboard
- TC002 Incorrect Credentials
    - Preconditions: The user account exists
    - Expected Results: An error message should be displayed

    - Actual Results: Error message displayed

Status: Pass

Test Set 2: Register(String, String, String, int) - Method

Feature Targeted: The register(String, String, String, int) method derives from the user class. This method depends on the input from the user to check if there is a preexisting account.

Explanation: While registering, this method asks for a name, email, password, and age. The test cases present are to check whether there is already an account linked to the email address that is being used to register.

Test Cases:

- TC007 Successful Registration
    - Preconditions: The user does not have an account
    - Expected Results: The account was created successfully
    - Actual Results: The account was created successfully
- TC008 Unsuccessful Registration
    - Preconditions: The user already has an account with the email used
    - Expected Results: The account already exists
    - Actual Results: The account already exists

Status: Pass

Test Set 3: SearchMovie(String): Method

Feature Targeted: The SearchMovie(String) method allows the user to input a movie title, genre, or keyword, and the system goes into the Movie database and displays movie's the are similar to the inputs from the user.

Explanation: It takes in the movie chosen as its parameter and goes into said movie's database. Each movie has a genre attached to it, so would be displayed if the user inputs a genre that correlates with the movie. If a movie title in taken, it checks to see if the movie exists and is in the database and displays the movie relating to the search.

Test Cases:

- TC009 Validate Search Functionality
    - Preconditions: Movies are available in the database
    - Expected Results: Matching movies should be displayed
    - Actual Results: Matching movies should be displayed

Status: Pass

INTEGRATION TEST

Test Set 1: choseSeat(String): String - Method

Feature Targeted: The *choseSeat(String)* method is in the ticket class. This allows the user to select their seat, and check if the seat they want is available.

Explanation: It takes in the movie chosen as its parameters and goes into said movie's database, then allows the user to select a seat. It also uses a boolean response as the user is selecting the seat. If the seat is available, then it will allow the user to add it to their ticket, if not, there will be an error message and return to the seat selection page.

Test Cases:

- TC003 Available Seat
    - Preconditions: The movie has available seats
    - Expected Results: Seat should be booked successfully
    - Actual Results: Seat is booked successfully
- TC004 Unavailable Seat
    - Preconditions: Another user already booked the same seat
    - Expected Results: An error message should be displayed
    - Actual Results: Error message displayed

Status: Pass

Test Set 2: PaymentMethods: (List<String>) - Method

Feature Targeted: The *PaymentMethods: (List<String>)* method in the user class. This allows the user to add and save payment methods that are associated with their account.

Explanation: It takes in the payment method as its parameter and is encrypted to ensure security. It also makes sure that the payment method is valid and if it is invalid, then the payment fails.

Test Cases:

- TC005 Successful Payment
    - Preconditions: The user has valid payment details
    - Expected Results: Payment successful, receipt should be generated
    - Actual Results: Payment successful, receipt generated
- TC006 Failed Payment
    - Preconditions: The user enters an expired card or not enough funds
    - Expected Results: Payment should be declined
    - Actual Results: Payment declined

Status: Pass

SYSTEM TEST

Test Set 1: Logging Out

Feature Targeted: Logging Out Functionality

Explanation: The user will be given the option to sign/log out of their account.

Test Cases:

- TC010 Logging Out
    - Preconditions: The user is logged in
    - Expected Results: The user is logged out, and redirected to the home page
    - Actual Results: The user is logged out, and redirected to the home page

Status: Pass

Test Set 2: Email Confirmation

Feature Targeted: Email Confirmation Functionality

Explanation: After ticket(s) is successfully booked, an email confirmation with movie details, seat number(s), etc. will be sent to the email associated with the account used to book the ticket.

Test Cases:

- TC011 Email Confirmation After Booking
    - Preconditions: The user has a valid email
    - Expected Results: A confirmation email is sent
    - Actual Results: A confirmation email is sent

Status: Pass