Movie Theater Ticketing System

Software Design Specification

Version 1.2

02/28/2024

Group 21
Alexander Kondan
Luiza Rocha
Shaun Romero

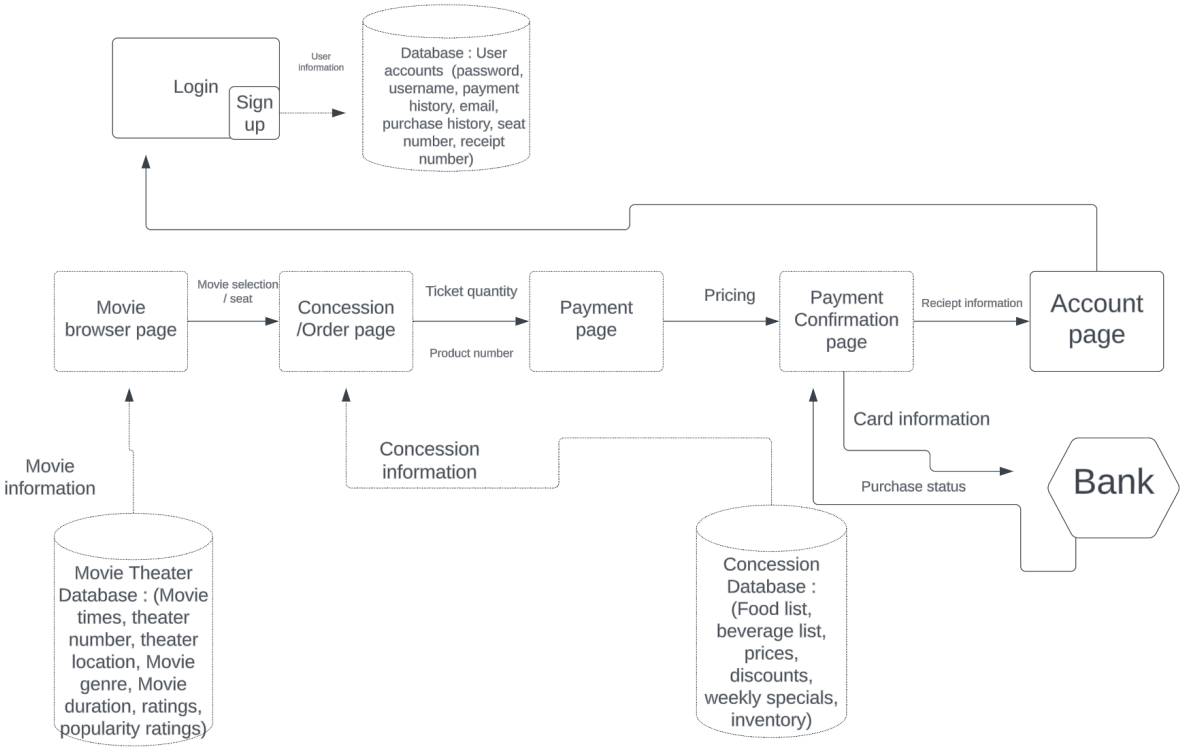
Prepared for
CS 250- Introduction to Software Systems
Instructor: Gus Hanna, Ph.D.
Fall 2023

System Description:

The scope of the Movie Theatre Ticketing System is to create a functional website to provide an integrated online real-time reservation, browse movies, list of theaters, payment transaction order and electronic information dissemination such as email confirmation and promotion to its members. This system will allow users to purchase tickets, create user accounts, and provide concession access before ever stepping into the theater. Users will be able to manage their accounts, purchase tickets, view movies, and purchase food and beverages all from their computer or mobile devices. The website's main objective is to provide movie going customers with an easy and pleasant experience.

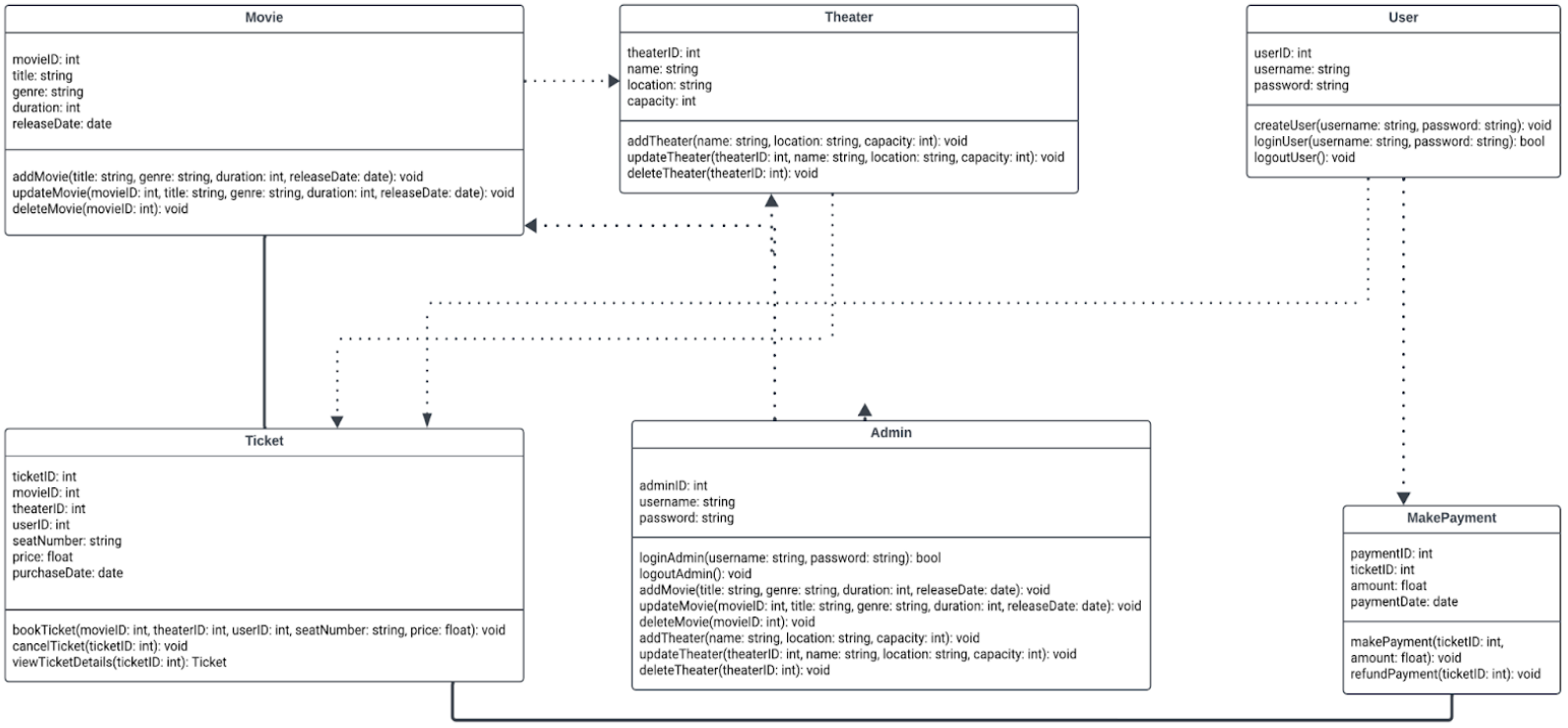
Software Architecture Overview:

- Architectural Diagram: (swa diagram)



- SWA Diagram Description

UML Diagram:



Student to GRADER NOTE: Couldn't get image to appear clear on doc. Please follow link to see more clearly:
https://lucid.app/lucidchart/01e71a52-b6bd-428c-b80f-12b644530d1e/edit?viewport_loc=-2358%2C-492%2C2750%2C1291%2C0_0&invitationId=inv_43cab007-a494-46d8-9da2-fb84c61e9685

UML Class Description:

1. Class: User

Description of Class:

This class represents a user of the movie ticketing system. Users can create accounts, log in, and book tickets for movies.

Attributes:

- userID: int - uniquely identifies each user in the system.
- username: string - stores the username of the user.
- password: string - stores the password of the user's account.

Operations:

- createUser(username: string, password: string): void
    - Description: Creates a new user account with the given username and password.
- loginUser(username: string, password: string): bool
    - Description: Logs in the user with the provided username and password. Returns true if login is successful, false otherwise.
- logoutUser(): void
    - Description: Logs out the current user from the system.

2. Class: Movie

Description of Class:

This class represents a movie available in the system. Admins can add, update, or delete movies.

Attributes:

- movieID: int - uniquely identifies each movie in the system.
- title: string - stores the title of the movie.
- genre: string - stores the genre of the movie.

- duration: int - stores the duration of the movie in minutes.
- releaseDate: date - stores the release date of the movie.

Operations:

- addMovie(title: string, genre: string, duration: int, releaseDate: date): void
    - Description: Adds a new movie to the system with the provided details.
- updateMovie(movieID: int, title: string, genre: string, duration: int, releaseDate: date): void
    - Description: Updates the details of the movie identified by movieID with the provided information.
- deleteMovie(movieID: int): void
    - Description: Deletes the movie with the specified movieID from the system.

3. Class: Theater

Description of Class:

This class represents a theater where movies are screened. Admins can add, update, or delete theaters.

Attributes:

- theaterID: int - uniquely identifies each theater in the system.
- name: string - stores the name of the theater.
- location: string - stores the location of the theater.
- capacity: int - stores the maximum seating capacity of the theater.

Operations:

- addTheater(name: string, location: string, capacity: int): void
    - Description: Adds a new theater to the system with the provided details.
- updateTheater(theaterID: int, name: string, location: string, capacity: int): void
    - Description: Updates the details of the theater identified by theaterID with the provided information.
- deleteTheater(theaterID: int): void
    - Description: Deletes the theater with the specified theaterID from the system.

4. Class: Ticket

Description of Class:

This class represents a ticket booked by a user for a movie screening.

Attributes:

- ticketID: int - uniquely identifies each ticket in the system.
- movieID: int - stores the ID of the movie for which the ticket is booked.
- theaterID: int - stores the ID of the theater where the movie is screened.
- userID: int - stores the ID of the user who booked the ticket.
- seatNumber: string - stores the seat number allocated for the ticket.
- price: float - stores the price of the ticket.
- purchaseDate: date - stores the date and time when the ticket was purchased.

Operations:

- bookTicket(movieID: int, theaterID: int, userID: int, seatNumber: string, price: float): void
    - Description: Books a ticket for the specified movie, theater, and user with the provided seat number and price.
- cancelTicket(ticketID: int): void
    - Description: Cancels the ticket with the specified ticketID.
- viewTicketDetails(ticketID: int): Ticket
    - Description: Retrieves and returns the details of the ticket with the specified ticketID.

Development Plan and Timeline:

Week 1-3: Project design preparation
Week 4-6: User Interface and database design
Week 5-6: User and Login System development
Week 7: Hardening: bug fixes
Week 8-9: Payment feature development
Week 10: Hardening: bug fixes
Week 11-12: Data integration
Week 13-14: Security Implementation and Testing
Week 15: Hardening: bug fixes
Week 16-17: Product testing
Week 18: Hardening: bug fixes
Week 19: Final push for development and release

Team Member Responsibilities:

[Alex]: Database design and management, data integration
[Shaun]: UI design, user registration and login system, security and final testing.