Software Requirements Specification Template

Software Engineering

The following annotated template shall be used to complete the Software Requirements Specification (SRS) assignment.

Template Usage:
Text contained within angle brackets (‘<’, ‘>’) shall be replaced by your project-specific information and/or details. For example, <Project Name> will be replaced with either ‘Smart Home’ or ‘Sensor Network’.

Italicized text is included to briefly annotate the purpose of each section within this template. This text should not appear in the final version of your submitted SRS.

This cover page is not a part of the final template and should be removed before your SRS is submitted.

Movie theater ticketing

Software Requirements Specification

Version <4.0>

8.4.2025

Group #6
Daniel Wen,
Lawrence Weathers,
Yangchen Lee

Prepared for
CS 250- Introduction to Software Systems
Instructor: Gus Hanna, Ph.D.
Summer 2025

Revision History

Date	Description	Author	Comments
<date>	<Version 1>	<Your Name>	<First Revision>
7/15/2025	<Version 1>	Group 6	Requirements Specification
7/22/2025	<Version 2>	Group 6	SDS addition pg 7
7/29	<Version 3>	Group 6	1. UML diagram updated ● Added reset password function in user class ● Added view ticket and get ticket held function for customer users ● Added view update log for Admin users ● Added bot and purchase limit check in shopping cart class. ● Corrected user, administrator class association *All changes applied to diagram description. 2. Added development plan and timeline 3. Added test plan description pg14
8/4	<Version 4>	Group 6	Added Architecture Diagram Added Data Management Strategy

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date

	Group 6 CS250	Software Eng.	7/22/2025
	Dr. Gus Hanna	Instructor, CS 250	

Table of Contents

1. Introduction

With the evolution of the movie theater industry, a robust ticketing system needs to be created to better serve customer needs. This document acts as a blueprint to guide the developer on how to create a movie theater ticketing system that will include assistive features and effective functionalities to simplify the ticketing process for today's consumer.

1.1 Purpose

The purpose of this document is to create an online system to facilitate the sale of movie tickets with the customer's satisfaction in mind. This system is intended to enable users to search movie schedules, select seats, and purchase tickets via browsers with ease. The document provides the outline of the goals that should be met in the final product.

1.2 Scope

The scope of the system includes the ability for users to browse movies available at their selected theater, see the number of tickets and seating available, and the price of a ticket. The system holds a database containing movie schedules and availability for all theaters. Users can see movie reviews and ratings, purchase discounted tickets, and see live updates on ticket availability. Feedback can be submitted by the users at the end of each purchase. Additionally, an administrator mode will be provided for theater managers to override the system in case of any error.

1.3 Definitions, Acronyms, and Abbreviations

User	An individual who uses the website to browse or buy movie tickets
Administrator / Admin	A manager who has access to override the system and handle errors
UI	User Interface
DB	Database
Discounted tickets	Tickets sold at a discounted price when certain condition is met
Rating	A score showing critics' opinions on a movie
Live Update	Real-time statistics shown to the user

1.4 References

The references are:

Marvel Electronics and Home Entertainment E-store project
IEEE Recommended Practice for Software Requirements Specifications
Software Requirements Specification document with example

1.5 Overview

The following sections of this document provide a general description and the requirements of the software. Section 2 discusses the functions of the product, characteristics of the users, and the constraints when developing the project. Section 3 describes detailed system functional requirements and the external interface requirements.

2. General Description

This section will discuss the general factors that are to be implemented into the Movie theater ticketing system.

2.1 Product Perspective

The Movie theater ticketing system provides a user-friendly interface that allows users to search for and purchase tickets on the website. It should streamline the user booking experience into an intuitive experience and catalog their records into an efficient database alongside ticket availability, film schedules, and theatre statistics.

2.2 Product Functions

The system provides a search feature for movie schedules, displays seat maps, and ticket pricing. Remaining tickets will be adjusted and displayed to the users according to real-time availability updates. Users can select from two seating options, regular or premium, and the pricing shown would be adjusted accordingly, with the choice to purchase a discounted ticket if certain conditions are met. They can also read movie reviews and ratings before making a purchase. Emojis like smiley, neutral, or angry faces will be shown to the users at the end for them to choose from, and collected as the user's feedback as a simple survey. An administrator mode is provided for the manager to make changes to movie schedules for each theater and resolve errors if a customer encounters an error during a purchase.

2.3 User Characteristics

Users of this system will primarily be customers who are able to view movie screening times, available seats, create ticket reservations, and eventually make purchases according to selected parameters. The user should be able to do the above through the theater's website, both on a mobile device and a PC, and have the option to store their data with their own account or to operate as a guest. The system also supports theater staff and privileged administrators with the ability to manage movie schedules, oversee purchases/reservations, general troubleshooting, and system adjustments.

2.4 General Constraints

Constraints include an intuitive user interface available on modern browsers such as Chrome or Edge, with customer data being protected by meeting webpage security standards. Showtimes, ticket availability, and film statistics should all be shown to the user live with a slight delay, whereas the background database is to be updated in real time to avoid accidents such as double-booking and fraud such as botting.

2.5 Assumptions and Dependencies

It is assumed that users will have a stable internet connection and access to a web browser. Movie schedules and seat availability are assumed to be updated by the theater staff with admin mode regularly. Admin users are assumed to have the necessary knowledge to perform a system override readily for a wide range of potential circumstances. The system depends on external APIs to process ticket payments and to display certain visual elements such as reviews.

3. Specific Requirements

The following section is designed to provide a comprehensive overview of developing the Movie theater ticketing system. Listed below are the functionalities and features that will be implemented along with use cases.

3.1 External Interface Requirements

3.1.1 User Interfaces

- Input Fields
    - Create user login
    - Continue as a “Guest”
    - Search bar
    - Reserve option
    - Purchase option
    - User survey option
    - Support ticket submission
- Drop-Down lists
    - List of Movies being shown
    - List of theater locations
    - Filter options
        - Genre (i.e. Romantic, Comedy, Action)
        - Ratings (Y, PG, PG-13, R)
        - Length of Movie
        - Theater type (3-D, iMax, Standard)
- Ability to browse both current and future screening times given calendar dates
- Progress updates (where you are in the ticket purchasing process)
- Real-time visual output of available seating based on selections

3.1.2 Hardware Interfaces

- Laptop, desktop, tablet, cell phone.
- The ability to connect via the internet.

3.1.3 Software Interfaces

The Movie theater ticketing system will support multiple web browsers. If the user is using an unsupported browser, they will receive a pop-up notification with a list of supported browsers and versions. Examples of supported browsers are below, with example version numbers:

- Google Chrome (ex. 25.9 and up)
- Firefox (ex. 25.10 and up)
- Safari (ex. 16.6 and up)
- Microsoft Edge (25.1 and up)
- Brave (10.5 and up)

3.1.4 Communications Interfaces

The user will be able to contact the theater by submitting support tickets and emails, which are then stored in the database, where theater staff are immediately notified. Other contact methods, such as social media and help phone, are not handled by this system.

3.2 Functional Requirements

This section covers the functionality of the Movie theater ticketing system. It covers functions that are necessary for the successful development of the system, including small details such as information retention and error handling when going back to previous pages.

3.2.1 Payment

- Should be able to process payments using Credit Cards, Debit Cards, PayPal, gift cards, Venmo, Zelle, or reward points.
- It should also offer an option to pay at the theater during ticket pickup.
- If there is an error during the payment process, the user will receive an error message. They will be given a chance to purchase again, or the purchase will fail, and the user will not be charged.
- Requested refunds are either returned to the user's payment method or via theater credit.

3.2.2 Login

- Users can create an account to buy tickets. This will allow them to save payment information, earn rewards from purchases.
- There is an option to proceed as a guest.
- If a user enters incorrect login information, they will see a message “incorrect email or password”.
- Ability to select “Forgot Password” or “Forgot Login”.

3.2.3 Ticketing

- After the desired number of tickets is selected, the user is taken to the seat selection screen.
- Users are given 10 minutes to complete the purchase after selecting seats.
- During that 10-minute period, seats are shown as filled, so that seats cannot be purchased while the customer is completing the purchase.

- Options for discounted tickets are shown to the users, including student, military, and senior discounts.

3.2.4 Administrator mode

- The system allows theater managers to add, remove, and update movie listings.
- Managers can override the system at any phase of a customer's purchase to handle errors.
- Customers are restricted from accessing administrator mode.

3.3 Use Cases

3.3.1 Use Case #1 Customer purchasing tickets

1. Users browse and choose a movie.
2. The system shows movie schedule, ticket price, and seating options.
3. Users choose ticket discount options and, if possible, seating.
4. Users proceed to payment.
5. System confirms purchase.
6. Ticket availability updated.

3.3.2 Use Case #2 Submit feedback

1. The system prompts the user to leave feedback
2. The user chooses from three separate emojis according to their level of satisfaction
3. The system stores the feedback

3.3.3 Use Case #3 Admin override system

1. Admin is notified of a showtime input error
2. Admin accesses the admin panel
3. Admin chooses a movie showtime
4. Admin manually removes the movie from the schedule
5. System updates the database
6. Interfaces update to reflect changes

3.4 Classes / Objects

3.4.1 <Ticket Class>

3.4.1.1 Attributes

- ticketID: Integer, unique ID for the ticket
- purchaseLimit: Integer, 20 per purchase
- userID: Integer, unique ID for each user
- seatNumber: String, user-selected seat
- Showtime: Integer, scheduled movie showtime
- Price: Float, ticket price
- holdStartTime: Timestamp, ticket purchase timer
- holdDuration: Integer, 10 minutes timer
- isExpired: Boolean, returns true if a user has exceeded the purchase time limit

3.4.1.2 Functions

- purchase(): Allows the user to choose a payment method and purchase tickets

- checkExpiration(): Check if the reserved session has ended
- holdSeat(): Temporary hold tickets during a session
- releaseSeat(): Unhold tickets if a user's session has expired
- updateAvailability(): Update ticket availability after a successful purchase
- changeLanguage(): Allows the user to change language into Spanish, Swedish, or English
- cancel(): End purchase

3.4.2 <Admin Class>

3.4.2.1 Attributes

- adminID: Integer, unique ID for an admin user
- username: String, login name

3.4.2.2 Functions

- login(username): Boolean, authentication for an admin user
- overrideSystem(): Boolean, allow staff to handle errors
- adjustShowtime(): Boolean, allow staff to change movie schedule

3.5 Non-Functional Requirements

3.5.1 Performance

- 95% of page loads and user actions should be complete within 2 seconds under normal conditions
- The system should support up to 1000 concurrent users

3.5.2 Reliability

- The system should recover to the normal state within 5 minutes with auto reset or admin override
- The UI should be easy for both the users and admins to understand and operate without additional assistance

3.5.3 Availability

- The monitoring system would notify System Administrators of the ticketing system being offline.
- Monthly maintenance on the system would ensure it is running at optimum levels (patching, purging of stale data)

3.5.4 Security

- Use MFA for login for registered users.
- Monitoring of purchases ensures bots are not buying tickets for the resell market.
- Ensure encryption of data
- Require employees to use specific authentication methods (i.e MFA, badges, hardware tokens)
- Bot mitigation techniques, such as CAPTCHA or reCAPTCHA, shall be implemented during ticket purchasing to ensure the actions are performed by human users

3.5.5 Maintainability

- Schedule weekly downtime for maintenance. Would be scheduled during low traffic hours (12am-8am) with no more than 2 hours of downtime
- Implement a monitoring system that would notify of ticketing systems being down or services being in a “Failed” state.
- Potential creation of a failover system in the event the main system goes down, traffic can be moved to the failover system to ensure minimal downtime.

3.5.6 Portability

- The system shall be usable on all modern browsers (Chrome, Edge, Firefox, etc.)
- The UI must adapt to both desktop and mobile devices

3.6 Inverse Requirements

*State any *useful* inverse requirements.*

3.7 Design Constraints

Specify design constrains imposed by other standards, company policies, hardware limitation, etc. that will impact this software project.

3.8 Logical Database Requirements

Will a database be used? If so, what logical requirements exist for data formats, storage capabilities, data retention, data integrity, etc.

3.9 Other Requirements

Catchall section for any additional requirements.

Starting point for Assignment #2

4. System Description

4.1 Brief overview of ticketing system

The following sections detail the Software Design Specifications for the Movie theater ticketing system. This will offer an outline of the architectural design of the project before implementation. This will provide a view of the flow of the ticketing system, starting from the user logging in, to completing the purchase.

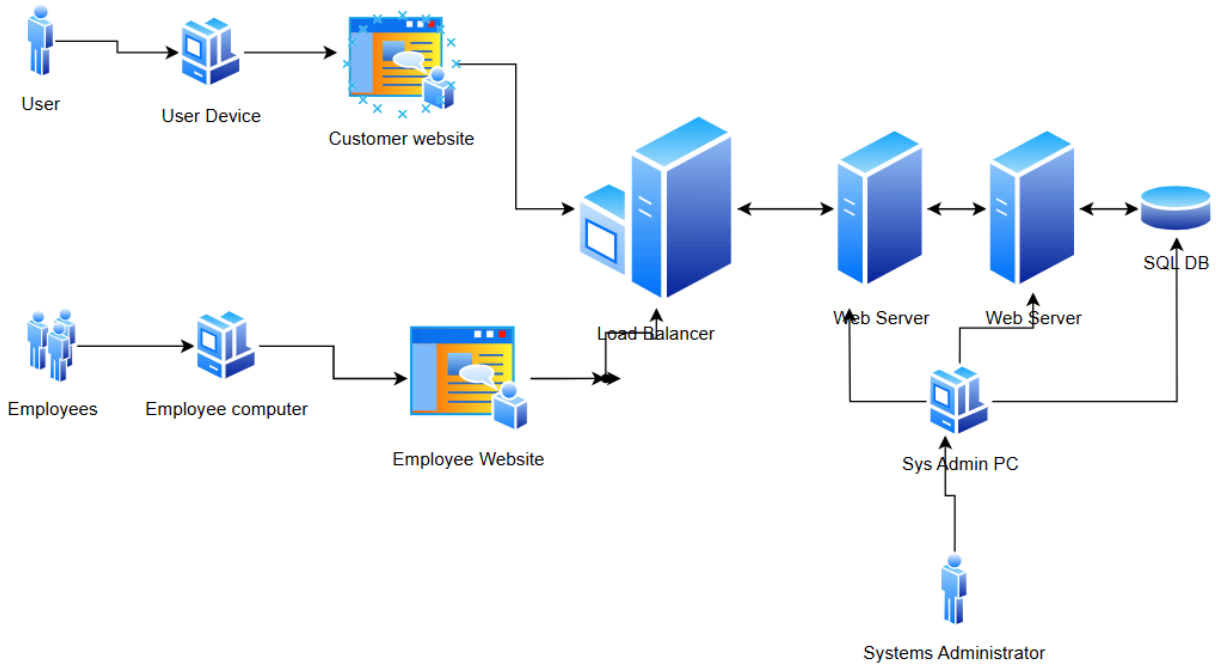
5. Software Architecture Overview

Below is a diagram of the system architecture. This serves as an overview for the developer on how the system should work. The diagram shows the flow from the user connecting to the ticketing system via their device. The data goes through the load balancer and connects to the web servers. The web servers then take the input from the user to search the database for the movie, and send the information back through the web servers to the user.

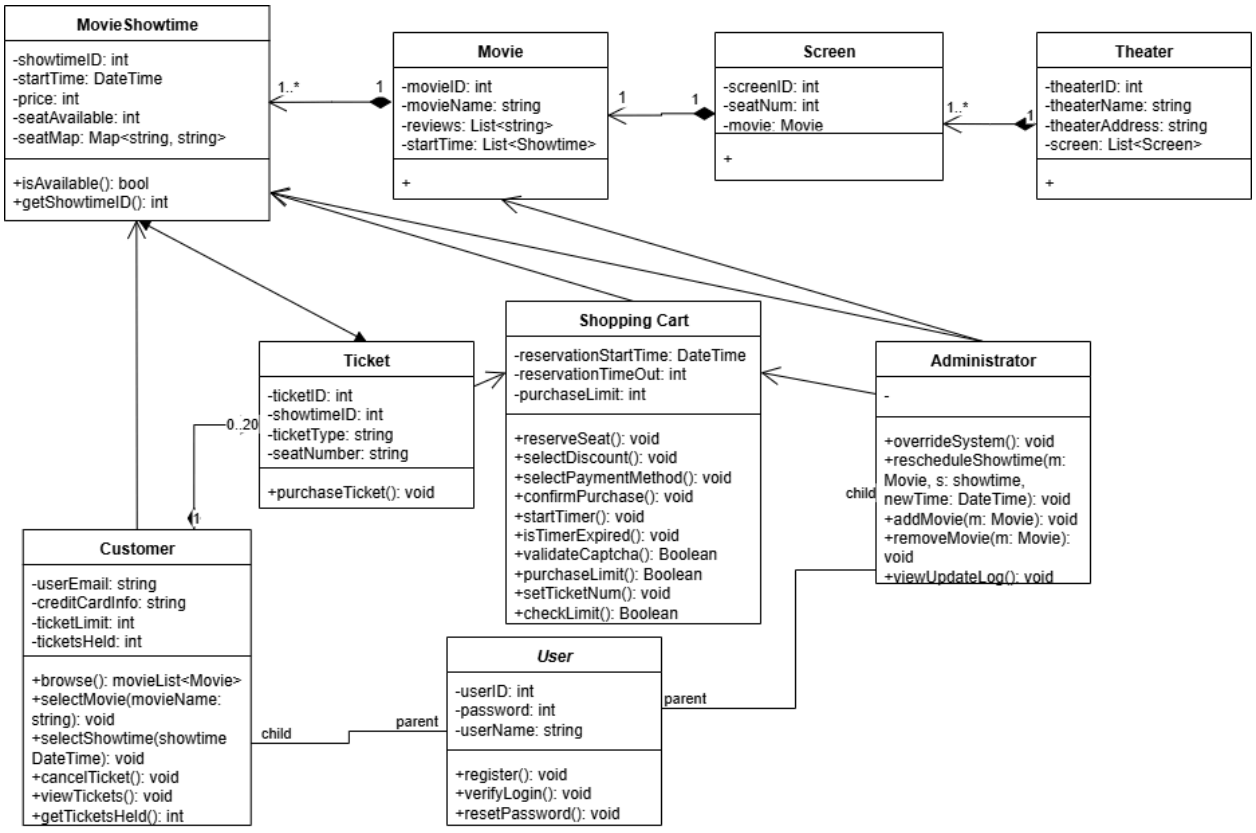
The access for employees is similar but instead of accessing the system through a personal device, the employees would be using computers at the theater or office.

System administrators will access the Web Servers as well as the SQL DB via system administrator PC which they will be the only users who have access to.

5.1 Architectural diagram



5.2 UML Diagram



6. Description of Classes, Objects & Attributes

Below are descriptions of the classes, objects, and attributes that will be used for the development of the Movie theater ticketing system. The objects and respective functions described will reside within the SQL database.

The database itself will use SQL and will be managed by the System Administrators.

6.1 Movie Showtime

Represents scheduled screening time of a movie

- Stores when a show occurs (startTime)
- Tracks ticket and seating info (price, seatAvailable, seatMap)

6.1.1 Attributes

- (int) showtimeID: Sequential identifier for movie showtime, starting at 1
- (DateTime) startTime: Exact time stamp for movie showtime.
- (int) price: Non-discounted price for ticket at that time.
- (int) seatAvailable: Holds the number of seats available left for a specific showtime. Used as the condition of isAvailable method.

- (Map<string, string>) seatMap: A map that holds the status and the location of a seat. A string is used for status (available, reserved or non-reserved). The other string holds the seat number that indicates the row and column location of a seat (e.g. “M17”).

6.1.2 Methods

- (bool) isAvailable(): Return false when seatAvailable == 0
- (int) getShowtimeID(): Return showtime ID

6.2 Movie

Holds movie information

- Stores a list of reviews for the film
- Stores a list of showtime set by the administrators

6.2.1 Attributes

- (int) movieID: Unique, auto-incremented ID for each movie
- (string) movieName: Name of the movie show to users
- (List<string>) reviews: List of critics reviews
- (List<Showtime>) startTime: List of movie showtime

6.3 Screen

Models an auditorium in a theater

- Identify the auditorium (screenID) and capacity (seatNum)
- References the movie that is currently showing (movie)

6.3.1 Attributes

- (int) screenID: Unique ID of the auditorium
- (int) seatNum: Number of capacity for an auditorium
- (Movie) movie: Movie object

6.4 Theater

Represent the physical theater location

- Holds cinema info (theaterID, theaterName, theaterAddress)
- Contain one or more Screen objects
- Managed by administrator

6.4.1 Attributes

- (int) theaterID: Unique, auto-incremented ID for each theater
- (string) theaterName: Name of a theater
- (string) theaterAddress: Address location of a theater
- (List<Screen>) screen: List of screens in the theater

6.5 Ticket

Records one purchased seat at a specific time

- Combine movieID and showtimeID for a customer’s purchase (ticketID)
- Identify each ticket (ticketType)
- Checked or removed by Customer.cancelTicket()

6.5.1 Attributes

- (int) ticketID: Unique ID for each tickets
- (int) showtimeID: ID that matches movie and showtime ID combined
- (string) ticketType: Ticket type purchased (Regular, Deluxe)
- (string) seatNumber: Status of seating or seat number (e.g. “A10”, “un-reserved”)

6.5.2 Methods

- purchaseTicket(): Calls purchase service

6.6 Shopping cart

Support step for processing a customer's reservation into a paid ticket

- Reserve a seat in a showtime (reserveSeat())
- Handle payment selection and processing (selectPaymentMethod(), confirmPurchase())
- Calls Showtime.isAvailable() and update seatAvailable
- Uses a timer to limit reservation time for a user's purchase

6.6.1 Attributes

- (DateTime) reservationStartTime: The time when the seat reservation began
- (int) reservationTimeOut: Time limit (minutes) to complete a purchase
- (int) purchaseLimit: Maximum ticket allowed to purchase at once, used in check limit function. In this case 20.

6.6.2 Methods

- reserveSeat(): Decrements available seats for a given showtime
- selectDiscount(): Show selections for regular and discounted prices (regular, students, military, senior, etc.)
- selectPaymentMethod(): Prompt user to select payment methods (credit card, gift card, etc.)
- confirmPurchase(): Finalize ticket purchase. Returns true if the payment is successful and the timer has not expired; return false otherwise.
- startTimer(): Starts countdown timer when a seat is reserved
- isTimerExpired(): Checks if reservationTimeOut number of minutes has passed; cancel purchase when true
- (Boolean) validateCaptcha(): Check for bot users. Returns false if bot check isn't passed.
- setTicketNum(): Allow user to set amount of ticket purchased, increment the number of ticket currently hold by the user
- (Boolean) checkLimit(): Check if the user has reached the maximum ticket purchased, calls get tickets held function in customer class. Prompt the user the maximum amount of ticket the user can get for the current purchase

6.7 Administrators

Special user with rights to manage movie schedules and handle customer errors

- Override system settings (overrideSystem())
- Add or remove movies (addMovie(), removeMovie())
- Change showtimes (rescheduleShowtime())

6.7.1 Methods

- overrideSystem(): Allow admins to take over the system for error handling or general support on behalf of a customer
- rescheduleShowtime(m: Movie, s Showtime, newTime: DateTime): Allows admins to change schedules to unpublished movie showtime
- addMovie(m: Movie): Allows admins to add movie to a screen
- removeMovie(m: Movie): Allows admins to remove unpublished movie showtime
- viewUpdateLog(): Display a list of changes in movie schedule done by all admins

6.8 Customer

End user that browses, selects, and purchase tickets

- Create and manage their account (register())
- Browse available movies and showtimes (browse(), selectMovie(), selectShowtime())
- Maintain payment info and track tickets held (creditCardInfo, ticketLimit, ticketsHeld)
- Uses purchaseService for booking

6.8.1 Attributes

- (string) userEmail: Unique email the customer uses to log in
- (string) creditCardInfo: Credit card info for payments
- (int) ticketLimit: Maximum number of unexpired tickets each customer can hold
- (int) ticketsHeld: Current number of a customer's unexpired, purchased tickets

6.8.2 Method

- register(): Creates new customer account; prompt invalid registration if email is already in use
- browse(): Show movies currently available for booking; return empty list if none
- selectMovie(movie: string): Choose a movie by clicking on the movie shown to the customer
- selectShowtime(showtime: DateTime): Choose an available showtime by clicking; showtime will not be shown if availability == 0
- cancelTicket(): Cancel the ticket selected; proceed to payment if otherwise
- viewTickets(): System display tickets owned by the user with detailed description of the movie (movie name, showtime, seating information, etc.)
- (int) getTicketsHeld(): Return the number of tickets held, called by check limit function in shopping cart class

6.9 User

Base class for everyone who logs into the system (Customer and Administrator)

- Stores authentication data (userID, userName, password)
- Verify credentials (verifyLogin())
- Extended by Customer and Administrator

6.9.1 Attributes

- (int) userID: System-generated unique user identifier
- (int) password: User entered code for authentication
- (string) userName: Display name set by the user

6.9.2 Methods

- verifyLogin(): Check credentials given by the user against the record; prompt forgot password message for invalid email and password. Calls reset password function if forgot password is clicked.
- register(): Prompt the user to create a new account. Reads an email input and two password inputs from the user. Check if the password entered matches and if the email is not registered.
- resetPassword(): Send a verification email to the user, read new password input from the user if email verification is successful

7. Development Plan and Timeline

7.1 Begin Development

- Consult with the design team to create a diagram showing the flow of the system.
    - Once design is approved, development team will begin working on the creation of the system
- After the creation of each module, the QA team will test functionality between modules to confirm functionality.
- Consult with System Administrators for security patching. Also, confirm with System Administrators that all proper security practices have been implemented to protect the customer as well as the theater.
- Discuss hiring customer service representatives to be available to assist users with issues.

7.1.2 Testing

- **Test login functionality**
    - User name/password
    - Guest Login
    - Forgot user name or Forgot password.
    - Create new account
- **Test search functionality**
    - Confirm can search Movie DB
        - Confirm filters functionality
    - Can search multiple theaters
    - Can search different days
- **Shopping Cart**
    - Test user can make purchase using different payment methods (credit card, pay pal, zelle, venmo, etc)
    - Test user can use rewards to make purchases
    - Test user account is credited with reward points when making purchases
    - Test user can request refund
- **Security**
    - Confirm CAPTCHA or MFA works
    - Confirm scheduling of updates during low peak hours (12am-5am)
    - Confirm encryption of purchases
- **System availability**
    - Stress testing, make sure the system can handle a high volume of requests.
    - Verify in the event that the main system goes down, the ticketing system fails over to the backup system to mitigate downtime.
    - Nightly backups of system
    - Disaster recovery plan in the event of unplanned outage

    - Hot site on standby
    - Ability to attach SQL DB to backup system

7.1.3 User testing
- Invite small group of users to test out ticketing system
- Take feedback from user testing to make changes to the system.

7.1.4 Final release
- Create event for release of ticketing system
    - Discounts for first 500 users to login

7.2 Team member responsibilities
Below is a list of who is responsible for what aspects of the development process for the Movie theater ticketing system

7.2.1 UX/UI Team
- Take an architectural diagram and implement.
- Take user input from user testing and implement into final Movie theater ticketing system
- Creates/manages GUI for ticketing system

7.2.2 Development team
- Develop code for the site.
- Create APIs for things like shopping carts, emails, etc.
- Implement user authentication.
- Collaborate with the UX/UI team to deliver and maintain the site.
- Implement proper security with the help of System Administrators (i.e MFA, CAPTCHA)
- Collaborate with System Administrators for patching and security updates to ensure the system is always running on latest versions.

7.2.3 QA
- Test a new version before being released to the site. Check for bugs, functionality that is failing, and report back to the development team to fix.

7.2.4 Database team
- Create database to manage Movie theater ticketing infrastructure
- Ensure database updates are installed when available during low peak hours
- Perform weekly indexing of DB to ensure the database has no errors.

7.2.5 Customer Service
- Work with customers having issues, front end support
- Escalate to the development team when issues with the site are found.

Starting Point Assignment #3

8. Test Plan

The purpose of this section is to cover verification testing for the following:

- Unit testing: Testing of class methods and functions
- Functional testing: test the interaction between the functions and methods
- System testing: Test behavior of the actual system.

8.1 Unit test

- isTimeExpired()
  - Description: The purpose of this test is to verify the functionality of the timeout function.
  - Input test: The user will select available seats for a movie showing. The user will not complete payment for the seats in the allotted time for purchasing.
  - Expected outcome: The user will receive a message stating that the purchase time has expired and the seats have been released for purchase by other customers.
  - Success/fail criteria
    - Success: Purchase session times out, seats are released for purchase
    - Failure: Session times out, user is still successfully purchasing seats.
- checkLimit()
  - Description: The purpose of this test is to verify that a single user can only buy up to the maximum number of tickets (20 per user). When the user goes to select seats, they will see a 2D layout of the theater. They will choose the seats and select continue/proceed.
  - Input test: Purchaser will select 21 seats at the seat selection screen for the movie. After selecting 21 seats, the purchaser will continue to proceed to the purchasing screen.
  - Expected outcome: Purchaser should receive an error message stating they have exceeded the maximum amount of tickets that 1 person can buy. Message will show how many seats need to be removed before the purchaser can proceed to complete the purchase.
  - Success/fail criteria
    - Success: Purchase fails, giving message that maximum seats allowed for purchase has been exceeded.
    - Failure: Purchase is successful.

8.2 Functionality

- Password reset

    - Description: This test will verify that when a user has forgotten their password, they can select the “Forgot Password” and be able to receive an email with instructions on how to reset the password. Password email should include a link that has a specific timeout (i.e. 20 minutes), and takes the user to a page where they can change the password. Part of the functionality is also that the user cannot use the same password. Uses “verify log in” and “reset password” function.
    - Test input: user goes to login page. Below password field is an option for password help (i.e. “Forgot Password”). The user clicks the option and is taken to the page to enter the email address. If the email address is valid, the user will receive an email with a link to reset password.
    - Expected outcome: Email is valid, user receives email, clicks link to go to password reset page, resets password successfully, and is able to login.
    - Success/failure criteria:
        - Success: User resets password, logs into account
        - Failure: User cannot login.
- Search bar
    - Description: The user is able to search for movies in the search bar using relevant key words as well as specific film titles.
    - Test input: The user types in a Movie title they believe is showing currently at the theater.
    - Expected outcome: User finds the movie using the search bar, is able to make a purchase.
    - Success/failure criteria:
        - Success: Finds movie using search bar
        - Failure: Does not find a movie using search bar, or receive an error message.

8.3 System

- user login
    - Description: Verify user authentication is working
    - Test input: The user logs in using the user name and password they created when they created their account
    - Expected outcome: The user logs in with user name and password. User sees account information
    - Success/failure criteria:
        - Success: User logs in
        - Failure: User is not able to login

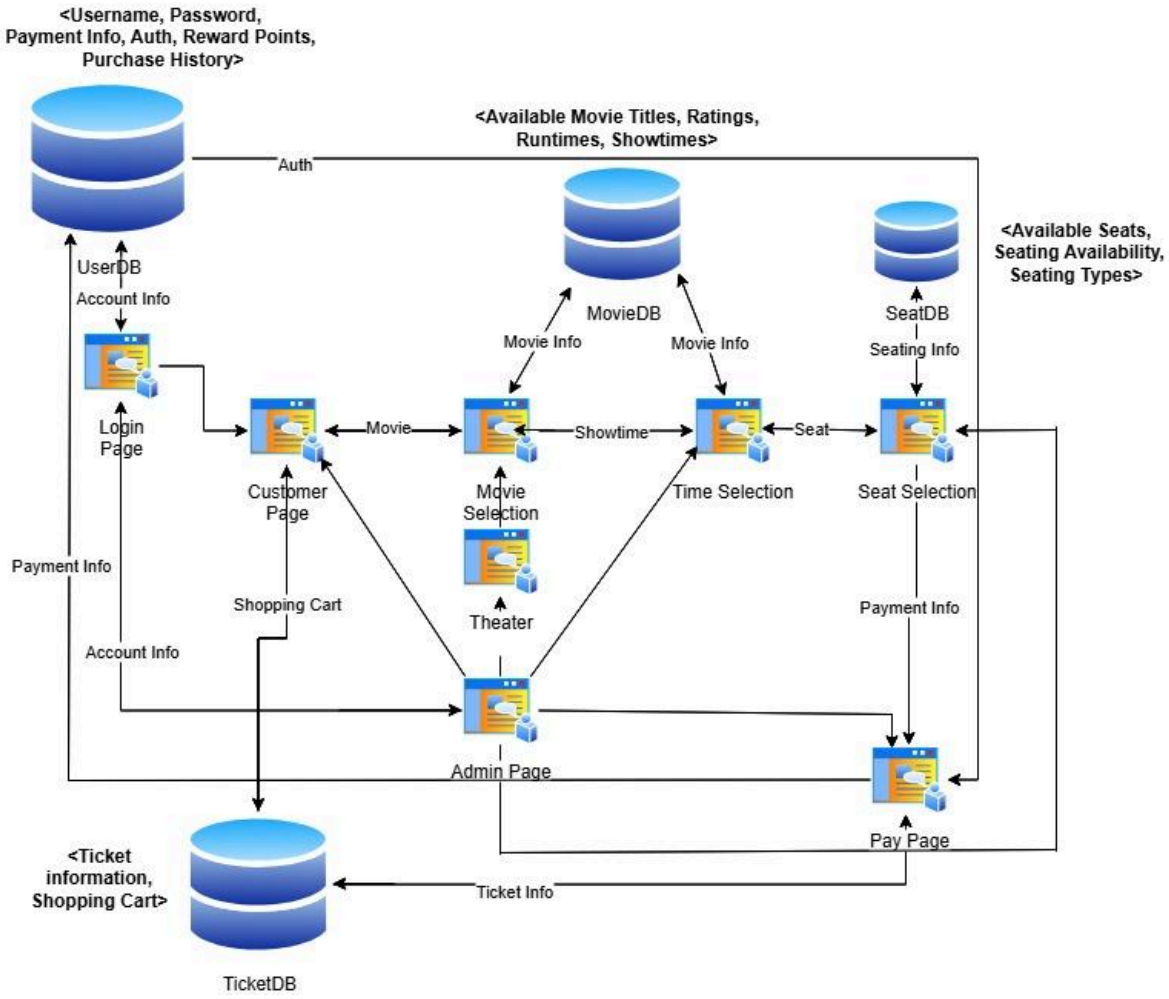
- View tickets purchased
    - Description: The test is used to check if the system correctly displays the tickets purchased by the user. A view tickets button will be displayed in the user menu. By clicking it, the system will display the tickets held by the user, including the number of tickets, movie name, and showtime.

- Test input: The user will proceed to purchase a ticket, then click the view ticket option in the user menu and be taken to a ticket display page after a successful purchase.
- Expected outcome: User should see the correct movie, showtime and number of tickets purchased displayed by the system.
- Success/failure criteria:
  - Success: Ticket displayed successfully, user can see all the details of the purchased ticket.
  - Failure: The system fails to display all the tickets purchased by the user. The system displayed the wrong information of the ticket.

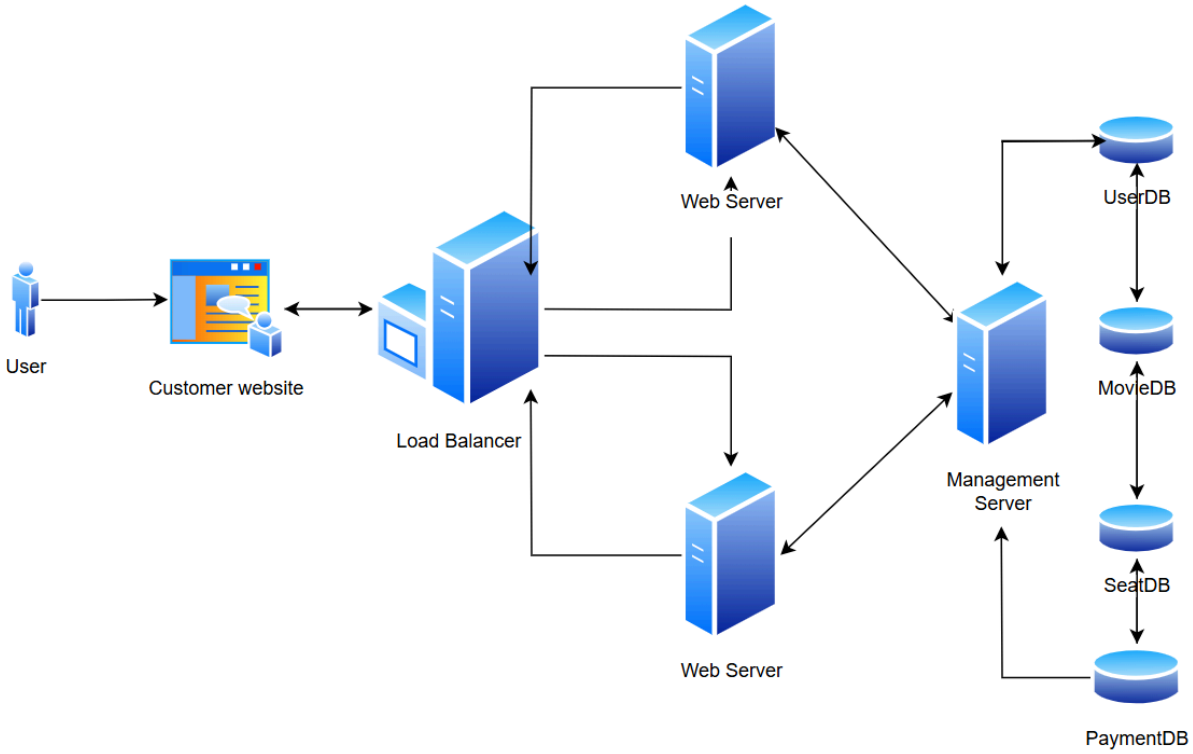
Start of Assignment #4

9. Architecture Design

- 9.1 Software Architecture Diagram



9.2 Data Management Strategy



The data will be managed using a web server farm hosted on AWS, positioned behind a load balancer and supported by SQL databases. This infrastructure ensures high availability, performance scalability, and fault tolerance. The system will perform nightly backups during non-peak hours (between 12AM - 5AM) to reduce operational disruptions. Using AWS allows the ticketing system to scale backend resources as demand increases, which is essential for handling spikes in traffic during high-demand events such as new movie releases. To improve security, database encryption will be enabled both at rest and in transit. Additionally, we chose to split the main database into smaller, purpose-specific databases, such as one for user accounts, one for bookings, and one for payment data. This compartmentalization minimizes the impact in the event of a breach on top of helping organize the data types into manageable categories. This approach offers a balance between performance, scalability, and security. The use of SQL databases allows for structured, relational data storage, something we believe is ideal for more complex operations such as seat reservations, user management, and payment processing. We considered alternatives such as NoSQL databases, which offer flexible schemas and easier horizontal scaling, but they lack the strong consistency and relational integrity required (or at the least simplifies) operations like ticket bookings and reservations. Overall, we are confident in our chosen architecture in providing the reliability and simplicity needed now, with the flexibility to transition to more advanced solutions in the future if the system or demands evolve.

9.2.1 UserDB

- The user's information is stored in this database. This includes Username, Password, Payment info, Auth, Reward Points, and Purchase History.
- This database will implement encryption/hashing for passwords.

- The database works with the Login page, providing the information needed to verify the user's credential. It also works with the Pay page to facilitate the ticket purchasing process using the user's payment information.

9.2.2. MovieDB

- The database will store all movie information including the movie title, rating, runtime, and showtime.
- This database works with the movie selection page and time selection page. The Movie Selection page will provide the user the list of available movies and also allows for the searching of specific titles. It will also show the user general movie information, such as movie availability, ratings, runtime, and showtime. The time selection page will show the user the schedule and enable them to choose a specific movie showing based on the available showtimes.

9.2.3 SeatDB

- The database is for seat availability information. The seating list can vary, depending on the theater type.

9.2.4 TicketDB

- The database is used to store ticket information for paid tickets as well as unpaid tickets reserved in a user's shopping cart.

9.2.5 TheaterDB

- The database stores the different theaters and their corresponding information, such as location.

9.2.6 Login page

- Allows the users to login or register an account
- Directs customers and administrators to their respective user page.

9.2.7 Customer page

- Allows customers to browse movies by the name or suggestions.
- Allows customers to access the shopping cart to finish unpaid order or view purchased tickets.

9.2.8 Admin page

- Allows admins to adjust movie schedule
- Allows admins to add and remove movies
- Allows admins to override and also interact with user-accessible pages

9.2.9 Movie selection page

- Retrieves information from Movie Database
- Works with both customer and admin users
- Avails to the user the list of available movies and enables movie searching.
- Displays general movie information, such as movie availability, ratings, runtime, and showtime.

9.2.10 Time selection page

- Retrieves information from Movie Database
- Allows the users to choose different movie showtimes

9.2.11 Seat selection page

- Retrieves data from Seat Database
- Allows the users to view seat map and reserve available seat(s)

9.2.12 Pay page

- Retrieves payment and user information from User Database
- Allows the users to view and confirm a purchase

A. Appendices

A.1 Appendix 1

Below is a link to our group's GitHub repository.

- https://github.com/DerpayJr/CS250MovieSRS

A.2 Appendix 2