Movie Ticketing System

Software Requirements Specification

Version 1

9/21/2023

Group: 9
Omar Abdirahman
Brian Nguyen
Riley Phan

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

Revision History

Date	Description	Author	Comments
9/11/23	Version 1	Group 9	Filled out all sections except 3.4 and 3.6 onwards
10/05/23	Version 2	Group 9	Brief overview of system, UML diagram and SWA diagram.

Document Approval

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

Signature	Printed Name	Title	Date
	Omar Abdirahman	Software Eng.	9/11/2023
	Brian Nguyen	Software Eng.	9/11/2023
	Riley Phan	Software Eng.	9/11/2023
	Dr. Gus Hanna	Instructor, CS 250	

Table of Contents

Revision History ii

Document Approval ii

1. Introduction 1

1.1 Purpose 1
1.2 Scope 1
1.3 Definitions, Acronyms, and Abbreviations 1
1.4 References 1
1.5 Overview 1

2. General Description 2

2.1 Product Perspective 2
2.2 Product Functions 2
2.3 User Characteristics 2
2.4 General Constraints 2
2.5 Assumptions and Dependencies 2

3. Specific Requirements 2

3.1 External Interface Requirements 3
3.1.1 User Interfaces 3
3.1.2 Hardware Interfaces 3
3.1.3 Software Interfaces 3
3.1.4 Communications Interfaces 3
3.2 Functional Requirements 3
3.2.1 Ticket Booking 3
3.2.2 Communication 3
3.3 Use Cases 3
3.3.1 Register as a new account 3
3.3.2 Browse Movies
3.3.3 View Purchased Tickets 3
3.4 Classes / Objects 3
3.4.1 <Class / Object #1> 3
3.4.2 <Class / Object #2> 3
3.5 Non-Functional Requirements 4
3.5.1 Performance 4
3.5.2 Reliability 4
3.5.3 Availability 4
3.5.4 Security 4
3.5.5 Maintainability 4
3.5.6 Portability 4
3.6 Inverse Requirements 4
3.7 Design Constraints 4
3.8 Logical Database Requirements 4
3.9 Other Requirements 4

4. Analysis Models 4

4.1 Sequence Diagrams 5
4.3 Data Flow Diagrams (DFD) 5
4.2 State-Transition Diagrams (STD) 5

5. Change Management Process 5

A. Appendices 5

A.1 Appendix 1 5

A.2 Appendix 2 5

6. Software Design Specification 5

6.1 Unified Modeling Language 5

6.1.1 UML Diagram 6

6.1.2 UML Descriptions of Classes, Attributes and Operations 6

6.1.2.1 Movies 6

6.1.2.2 TheaterInformation 6

6.1.2.3 Showtimes 6

6.1.2.4 Ticket 6

6.1.2.4 MakePayment 6

6.1.2.5 Reserved 6

6.1.2.6 AccountInfo 6

6.1.2.7 EmployeeInfo 6

6.1.2.8 UserInfo 6

6.2 Software Architecture 6

6.2.1 Software Architecture Diagram *6*

6.2.2 User Interface *6*

6.2.3 Databases *6*

6.2.4 Connectors *6*

6.3 Development Plan and Timeline 6

6.3.1 Overview *6*

6.3.2 Partitioning of Tasks *6*

7. Test Plan and Cases 7

7.1 Test Plan 7

7.1.1 Test Plan *7*

7.2 Test Cases 7

7.2.1 Test Cases *7*

1. Introduction

1.1 Purpose

This software is responsible for the purchase, distribution, and sale of tickets for many movies and live viewings. This document will outline the requirements, scope, and constraints of this software. As well as define the targeted user audience and user-friendly interfaces. This document also outlines the intended ways to use the software by each user.

1.2 Scope

(1) "Swift Wave" is the name of the software to be created.

(2) It will provide an online mobile platform where movie theaters can list their movies and customers are able to search and purchase tickets listed by those studios/producers. This software will have a database of events that have been listed and make it easy for customers to be able to access information about new and upcoming movies. Swift Wave will be able to accept many various payment methods for financial transactions such as debit cards, credit cards, and various online wallets such as "Paypal, Apple Pay, or Venmo" related to purchasing tickets. After purchase, tickets will be distributed electronically to customers who then display and confirm with movie theaters to gain access to watch these movies.

(3) The software will include well-organized to-do lists that help coordinate the team in managing issues and tasks. The software will be user-friendly and easy to navigate.

(a) The benefits will be easier usage that allows people to buy tickets at any time and anywhere. The easy usage of the software will allow. This will eliminate the necessity for in-person queueing of ticket purchases.

(b) With the design of this software product will introduce a comprehensive ticketing system that will facilitate booking and purchases of movie tickets.

1.3 Definitions, Acronyms, and Abbreviations

- *SRS: Software Requirements Specification*
- *SMS: Short Message Service*
- *IMDB: Internet Movie Database*
- *IEEE: The Institute of Electrical and Electronics Engineers*

1.4 References

1. Title: Theater Ticketing Requirements.rtf.
   Source: Canvas
2. Title: Scenarios and Use Cases by William Y. Arms, Cornell University Computing and Information Science.
   Source: Canvas
3. Title: IEEE Recommended Practice for Software Requirements Specifications by IEEE Computer Society (Revision of IEEE Std 830-1993).

Source: Canvas

1.5 Overview

1. Content: The following sections of the SRS contain the detailed requirements of a movie ticketing system that works on mobile and web browsers.
2. Organization: The SRS is organized into multiple sections that detail a description of the product, interfaces, functional/non-functional requirements, use cases, constraints, and other requirements.

2. General Description

2.1 Product Perspective

2.1.1 System Interfaces

- Email and SMS services to send ticket confirmations and updates to users either through email or phone notifications
- Payment Services with third-party payment systems such as using credit cards or debit cards, Apple Pay, Venmo, and other services to process payments.

2.1.2 User Interfaces

- Checkout / Payments to facilitate order confirmation and payment processing.
- Admin dashboard to provide administration tools to manage movies by either adding, editing, or removing them along with managing user accounts.
- User Profile Management Page to update personal information such as email, name, and address. Along with that view previous and current tickets.
- Customer Support Page to contact any customer support.
- Homepage to display all listed movies and search button.
- Movie detail page to display information about the movie and current showtimes.
- User registration and login pages for user account creation and login.

2.1.3 Hardware Interfaces

- This theater ticketing system relies on users having a mobile device or any compatible hardware that can run a simple web browser.

2.1.4 Software Interfaces

- Database system for storing movie information and user information.
- Connection to popular movie databases such as IMDB
- Integration of third-party payment systems for processing payments.

2.1.5 Communications Interfaces

- Customer Support Communication that lets users contact customer support for any reason.
- Notification Services to send emails or updates to the users about their tickets, showtimes, etc.

2.1.6 Memory

- There must be enough server RAM and database storage for the system to manage user accounts, booking records, movie information, and customizations. Memory and storage space should be expandable to meet growing data volumes.

2.1.7 Operations

- Movie information display.
- Secure payment processing
- User management
- Customer support
- Admin management
- User registration and authentication
- Movie and seat reservation/booking

2.1.8 Site Adaptation Requirements

- The system must be responsive and adaptable to varied screen sizes and resolutions in order to give a consistent user experience across devices.
- From the standpoint of its product, this data provides a complete insight into the interactions, interfaces, and adaptability requirements of the ticketing system.

2.2 Product Functions

Swift Wave is an online mobile platform open to movie theaters and customers to purchase and manage tickets. Movie theaters can create and manage their movies. Customers can search from a database of events listed by movie theaters, view event details, purchase tickets, and manage what tickets they have purchased.

2.3 User Characteristics

The users of this product will be people ages 18 and up. The users are expected to have a basic level of understanding of how online ticket reservation works and mobile devices. It is to be used by people who have a basic general education.

2.4 General Constraints

- **Regulatory policies:** The theater ticketing system must abide by a number of legal requirements governing ticket sales and accepting online payments.
- **Hardware limitations**: The ticketing system's hardware requirements must be met by the system in order for users to use their devices to access it. This involves taking various screen sizes, processing capabilities, and network connectivity into account.

- **Interfaces with other applications:** The system needs to connect to external programs and services, including payment gateways, and theater management systems. It is necessary to maintain these systems' compatibility and data sharing.
- **Parallel operation:** The system needs to allow parallel operation to handle a large number of concurrent users during periods of strong ticket sales. Critical elements to take into account are scalability and load balancing.
- **Audit Functions:** To monitor user activity, financial transactions, and system modifications, robust audit tools are needed. For the purposes of compliance, security, and troubleshooting, these audit logs are essential.
- **Control Functions:** The system must provide control features for managing user access, roles, and permissions. Administrators should be able to set and alter many aspects of the system.
- **Higher-Order Language Requirements:** High-level programming languages and frameworks must be utilized during the system's design and implementation if it is to be scalable, manageable, and quickly developed.
- **Signal Handshake Protocols:** To enable reliable data transmission between the ticketing system and external hardware or systems, communication protocols such as XON-XOFF or ACK-NACK may be required.
- **Reliability Requirements:** High-reliability standards must be met by the ticketing system in order to make sure that ticket purchases, payments, and user interactions are processed accurately and without flaws.
- **Criticality of the Application:** This is extremely crucial because it sells tickets for movies. If it malfunctions or breaks, it might cost money and aggravate users. As a result, it must always function and respond quickly to issues.
- **Safety and Security Considerations:** It's crucial to keep things safe and secure. We must make sure that no one may misuse the ticket system, steal user's personal information, or take their money.

2.5 Assumptions and Dependencies

- ***Availability of Internet Connectivity:*** *We believe that those who utilize the ticketing* system have high-quality internet. However, individuals can lose access to the system if their internet service is interrupted.
- **Third-Party Systems:** Other third-party services are used by the system, including emailing, movie information, and online payments. The system might not function properly if certain external services encounter issues or undergo changes.
- **Movie Release Schedules:** Based on data from the movie studios, the system displays when movies are screening and whether any tickets are still available. The system might not display the correct information if the movie schedules are altered or run behind schedule.
- **Hardware and Software Compatibility:** The ticketing system assumes that it is compatible with the majority of users' devices, including web browsers and operating systems. The ticketing system might need changes if these technologies undergo any significant updates or changes.

- **Scalability:** If more people choose to utilize it and make ticket reservations, we expect that the system architecture will expand. However, we must make sure we have enough technology infrastructure, such as storage, servers, and the internet, to accommodate additional users.
- **Customer Support Availability:** The ticketing system believes that there are customer support staff available at specific times to assist users with issues and inquiries.
- **Data Backup and Recovery**: The system must periodically backup its data and be able to restore it if something goes wrong. It believes that these procedures are well-maintained and effective.
- **Ticket Fraud Prevention:** To stop people from faking tickets, the system assumes security *is up to date and good at catching fake tickets.*

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

3.1.1.A Ticket booking interface

- A simple easy-to-use grid layout for new movies with 10 rows and 5 columns
- Movies can be ordered by latest, showtimes, and most popular.
- Within each row and column, an image of the movie and its name will be underneath it.
- When a movie is selected movie details along with all possible showtimes are shown
- The user will have a few buttons to add tickets to their shopping cart or list if the movie is currently unavailable.
- The interface will allow the user to choose any available seat and make payments.
- There will be a confirmation image before any major actions like payment and adding to the cart.
- The interface should look as similar as possible despite the device in use.

3.1.1.B Account Management

- There will also be a simple account management interface that offers users the ability to change account details such as passwords, profile pictures, and usernames.
- The user will also be able to view previous purchases.
- There will be the ability for users to activate two-factor authorization on their accounts and any account that does not have it enabled will be asked if they would like to enable it.

3.1.2 Hardware Interfaces

- This theater ticketing website for the user is any hardware that can run a simple browser app.

3.1.3 Software Interfaces

- Database system for storing movie information and user information.
- Connection to popular movie databases such as IMDB
- Integration of third-party payment systems for processing payments.

3.1.4 Communications Interfaces

- Email and phone services for sending confirmation of purchased tickets or updates to the customers.
- Connection to the Internet in order to use the application.
- encryption of user communication data.
- mobile push notifications on orders or suggested movies.

3.2 Functional Requirements

3.2.1 Ticket Booking

3.2.1.1 Introduction

Allow the user to browse all available movies at a selected theater, select showtimes, choose seats, payment, and process their booking.

3.2.1.2 Inputs

- The user selects a movie from the catalog
- Then showtime
- enters the number of tickets they would like to purchase
- selects what type of payment and confirmation buttons

3.2.1.3 Processing

- The system checks all of the user input and validates it
- Checks to see the availability of selected showtime and selected seats.
- Calculate pricing
- Reserves selected seats

3.2.1.4 Outputs

- Confirmation for actions like payment or adding tickets
- Display bought ticket details and unique ID

3.2.1.5 Error Handling

- Display detailed error messages on invalid actions or requests such as taken seats

3.2.2 Communication

3.2.1.1 Introduction

Communications will be facilitated through various means including but not limited to email, mobile push notifications, and SMS.

3.2.1.2 Inputs

- User information
- Option to opt-in for notifications about account actions and news

3.2.1.3 Processing

- Notifications will be sent via available communication methods

3.2.1.4 Outputs

- Specific content that caters to the user about their account
- for those who opt-in display newsletter and promotion information

3.2.1.5 Error Handling

- Message failed to deliver errors
- inform the user of invalid communication pathways through their account inbox on the application

3.3 Use Cases

3.3.1 Register as a new account

Use Case: The user creates an account

Primary Actor: User

Brief description: A new user needs to create an account in the software to get the ability to view, purchase, and manage tickets.

Pre Conditions:

1. The user is not signed in.
2. The user has a valid email address without an existing account.

Post Conditions:

1. The user has created an account and their information is stored in a database.
2. The user will be sent an e-mail confirmation which will need to be confirmed to access the website.

Flow:

1. The user opens the app.
2. The user clicks the "Create Account" button.
3. The system shows registration requirements.
4. The user inputs email, username, and password.
5. The user enables two-factor authentication via phone number.
6. The user clicks "Submit."
7. If all fields are correct, the system stores the user's info.
8. The system sends a confirmation email and a text to the phone number.
9. The user verifies the email by clicking a link in the email.
10. The system sends an authentication code to the user's phone.
11. The user inputs the code sent to their phone.
12. The account is activated, and the user can log in.

Alternate Flow:

1. If any field is incorrect or empty, show an error message and return the user to step 3 to re-enter information.
2. If the two-factor authentication code is not used within 5 minutes, it becomes invalid and the system prompts the user to request a new code. Ask the user to enter the new code. After three failed attempts do not allow the specific phone number.

System/SubSystem:

1. User account Management and security
    a. Manage user data such as email and phone for setting up a new account to log in.
    b. Make sure user data information is safe by using two-factor authentication when logging in.

Special requirements:

1. Data security
    a. Keep user data and information safe.
2. Two-factor Authentication
    a. An additional step to ensure that the account has the original user logging in, is the verification code through a phone number.
3. Email verification
    a. Send a link to the user's inputted email that the user needs to click to activate the account.

3.3.2 Use Case #2 Browse Movies

Use Case: Browsing Movies

Primary Actor: User

Pre-Conditions:

1. User logs into the software (app)

Brief Description: The user browses through a list of movies within the app. They can also filter the movies by genre or using search keyword characters. Each movie shown will include the details, reviews, genre, and rating.

Post-Conditions:

1. The user has viewed a list of movies and potentially filtered or searched for specific titles

Flow:

1. The user clicks "View Movie List."
2. The system shows a list of movies with titles, images, and genres.
3. The user has the option to filter movies by genre, showtime, and availability.
4. The user can type keyword characters into a search bar.
5. The System displays movies matching the keyword.
6. Users can see which movies have reviews and ratings per movie.

Alternate Flow:

1. The User opens the movie to see more details including ratings, cast, and movie duration.
2. The system will also show more detailed information about the selected movie in which the trailer is shown.

System/Sub System:

1. Movie Browser
    a. Movie listing shows all movies to the user with title, image, and genre
    b. Filters and search allow users to narrow down movies by filtering

Special Requirements:

1. Speed
    a. The app must load the list of movies and details quickly for the user to ensure a good experience
2. Network and Internet
    a. An Internet connection is required, if there is no Internet then the system will display an error.

3.3.3 Use Case #3 View Purchased Tickets

Use Case: View purchased tickets

Brief Description: The user logs into the app and checks the purchased tickets. They can view the details such as movie title, showtime, and scannable QR code. If the user has no tickets due to a cancellation refund or none existing bought ticket, an error message will display.

Primary Actor: User

Pre-Conditions:

1. The user logs into the software (app)

Post-Conditions:

1. User can view their list of purchased movie tickets

Flow:

1. User logs into their account
2. The user navigates to the "View Purchased Tickets" section on the website or app.
3. The software will display a list of tickets that have been purchased.
4. In this list, it will display information about the title of the movie, showtime date and time, theater location, seat number, scannable QR code, and a unique ID number.

Alternate Flow

1. The user has no purchased tickets found, showing the user a message stating "You have no purchased tickets"

System/Sub System:

1. Core system
    a. This element serves as the foundation of the movie ticketing system, coordinating interactions between subsystems and preserving the integrity of the entire system.
2. Scalability
    a. The system can handle traffic flow easily, involving thousands of users
3. User Account Management:
    a. Handles the user login and user information
4. Ticket database
    a. Manages the storage and data of the purchased ticket information
5. Ticket display
    a. Shows ticket information to the user

Special Requirements:

1. Data Security and Privacy

a. All user information will be secure and safe
b. Two-factor authentication will be implemented for additional security

2. Performance and Accessibility
    a. The system is quick and fast-loading
    b. user-friendly interface ensuring a straightforward experience for all users which even accommodates people with disabilities
3. QR Code Readability
    a. QR codes will be scannable at the theater
4. Multiple platforms support
    a. Will be available across platforms such as mobile devices, computers, and web browsing software.
5. Ticket notification
    a. Purchase of a ticket will notify via email confirmation
    b. Changes involving purchased tickets such as cancellation will be notified via email
    c. Ticket refund confirmation will be notified via email.

3.4 Classes / Objects

3.4.1 <Class / Object #1>

3.4.1.1 Attributes
3.4.1.2 Functions
<Reference to functional requirements and/or use cases>

3.4.2 <Class / Object #2>

…

3.5 Non-Functional Requirements

3.5.1 Performance

- The system should be able to support at least 2,000 concurrent users.
- The average page load time should be 2 seconds
- 90% of transactions will be processed in under 1.5 seconds

3.5.2 Reliability

- 1.5 hours of maintenance a month
- The MTBF value of the system should be at least 20 days
- User data, especially bought tickets shall be maintained by all means to avoid data corruption or loss.
- Data should be automatically backed up daily. This data should include transaction history, user settings/data, and other irreplaceable data.
- The program should have error logs for analysis

3.5.3 Availability

- The application will be only available in America for its launch.
- The system should be consistently running unless under maintenance.
- Backups to ensure data availability.

3.5.4 Security

- Sensitive user data such as payment information, passwords, emails, etc. will be stored in a secure and encrypted database.
- Access to sensitive data will be limited to authorized personnel only.
- Logs shall be monitored to detect any security breaches.
- Security will be regularly audited to ensure safety and security

3.5.5 Maintainability

- Regular software updates to keep up with industry standards and increasing users.
- All updates will be roughly documented and patch notes will be available to users.
- Hotfixes and patches will be released within 3 days of discovering the issue.

3.5.6 Portability

- The application will be available on major mobile operating systems such as Android and IOS as well as major web browsers (Chrome, Safari, Edge…).
- Mobile quality shall be ensured to be compatible with multiple different devices/screen sizes.

3.6 Inverse Requirements

*State any *useful* inverse requirements.*

3.7 Design Constraints

Specify design constraints imposed by other standards, company policies, hardware limitations, 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.

4. Analysis Models

List all analysis models used in developing specific requirements previously given in this SRS. Each model should include an introduction and a narrative description. Furthermore, each model should be traceable the SRS's requirements.

4.1 Sequence Diagrams

4.3 Data Flow Diagrams (DFD)

4.2 State-Transition Diagrams (STD)

5. Change Management Process

Identify and describe the process that will be used to update the SRS, as needed, when project scope or requirements change. Who can submit changes and by what means, and how will these changes be approved.

A. Appendices

Appendices may be used to provide additional (and hopefully helpful) information. If present, the SRS should explicitly state whether the information contained within an appendix is to be considered as a part of the SRS's overall set of requirements.

Example Appendices could include (initial) conceptual documents for the software project, marketing materials, minutes of meetings with the customer(s), etc.

A.1 Appendix 1

A.2 Appendix 2

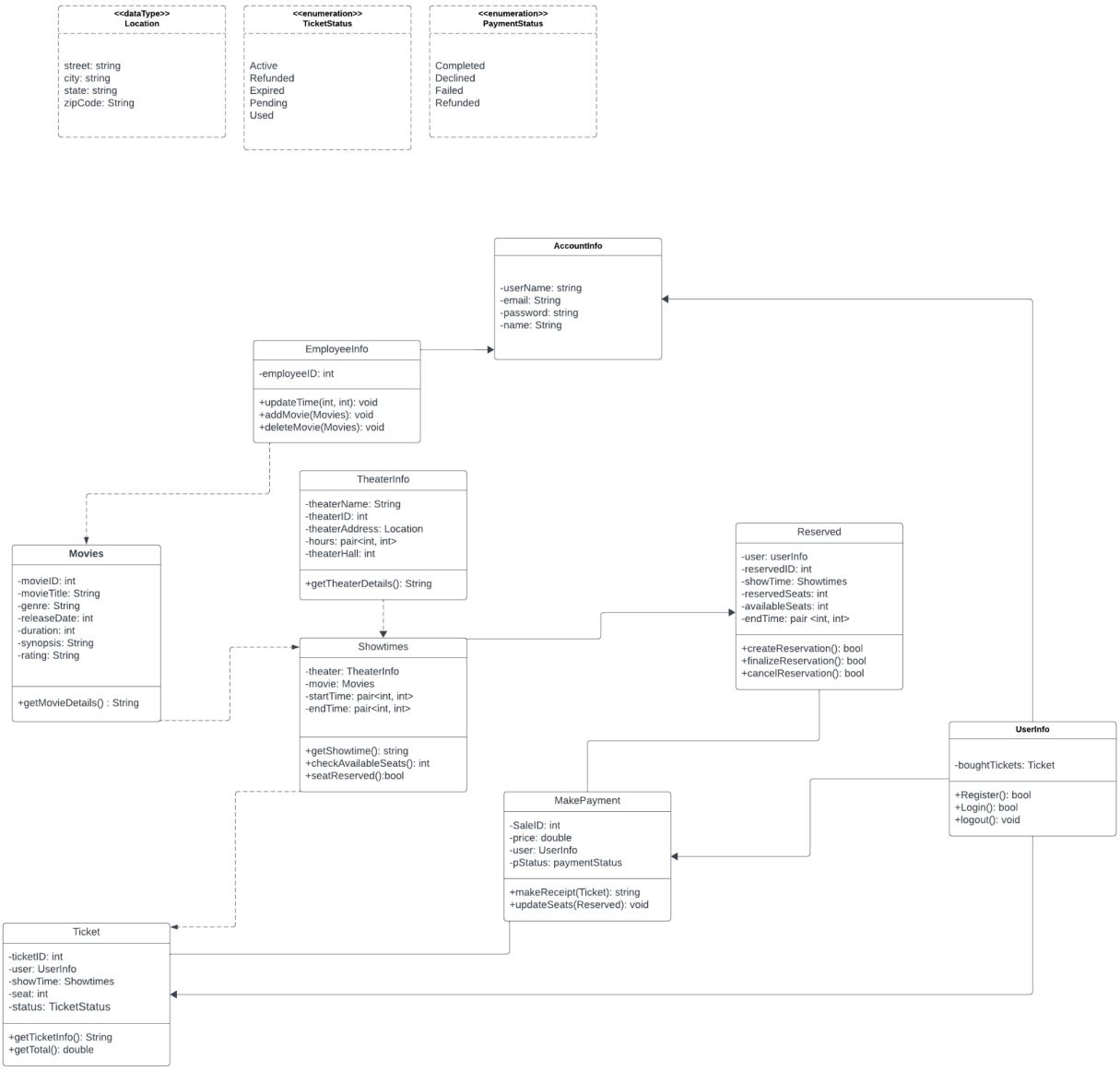
Software Design Specification
SUBMISSION BELOW 10/5

6. Software Design Specification

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

6.1 Unified Modeling Language

6.1.1 UML Diagram



6.1.2 UML Descriptions of Classes, Attributes and Operations

6.1.2.1 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.

6.1.2.2 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.

6.1.2.3 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.

6.1.2.4 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.

6.1.2.4 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.

6.1.2.5 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.

6.1.2.6 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.

6.1.2.7 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 Movie

6.1.2.8 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:

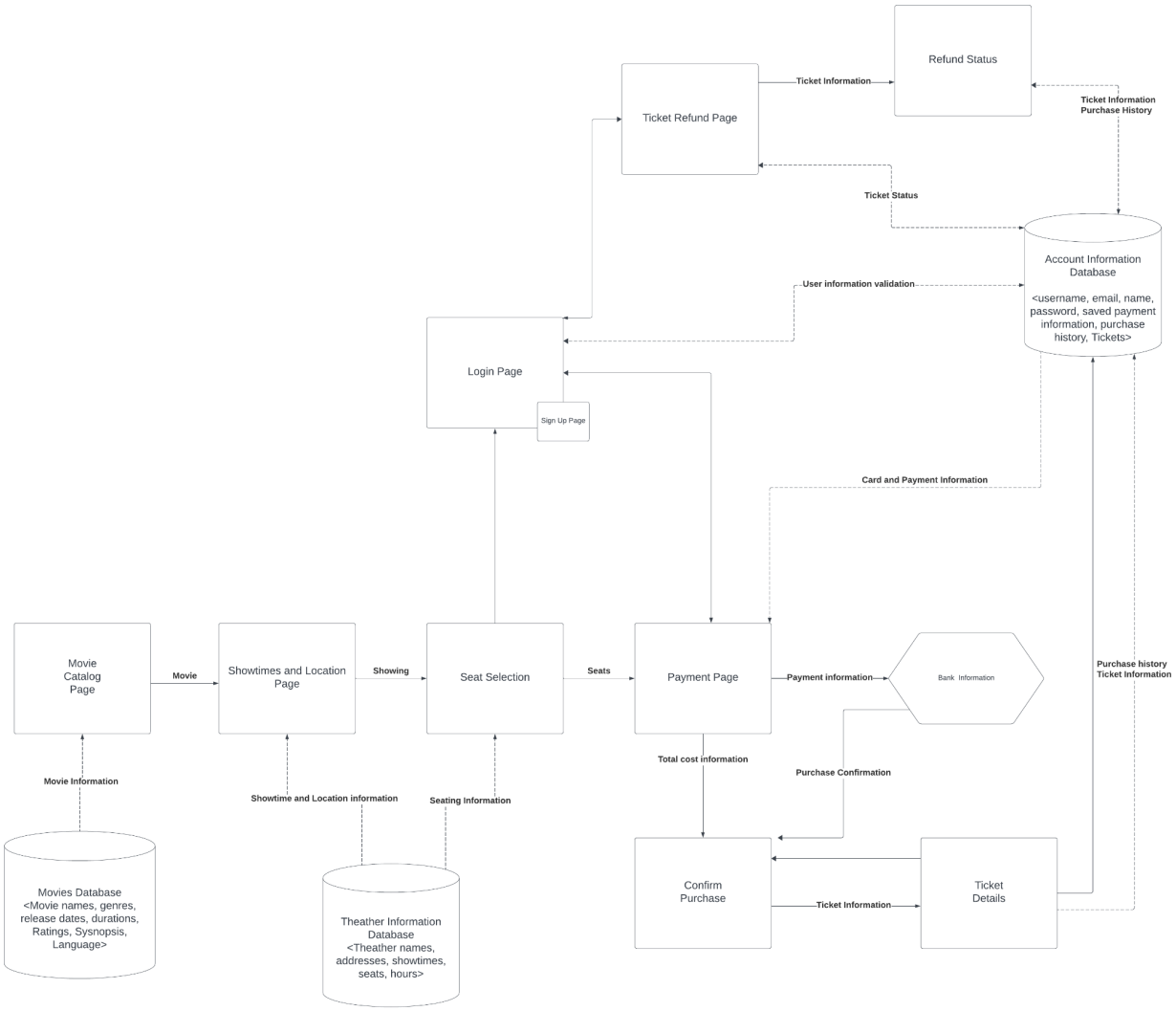
- boughtTickets (Ticket): A list of purchased tickets associated with the user.

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.

6.2 Software Architecture

6.2.1 Software Architecture Diagram



6.2.2 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 in order 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.

6.2.3 Databases

- Movie Catalog: a database for details on the movies that customers are able to 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 in choosing 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.

6.2.4 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 Inforamtion to authenticate access.
- User will have to use 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 login 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.

6.3 Development Plan and Timeline

6.3.1 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.

6.3.2 Partitioning of Tasks

The project has been split up into particular tasks, each of which focuses on a vital component of the system in order 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 included with the main objective for the week, included with task, estimated time, and the member(s) who will be responsible for.

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

7 Verification Test Plan and Cases

7.1 Test Plan

Verification Test Plan: Swift Wave Movie Ticketing System

7.1.1 Test Plan Introduction

- Validate that the theater ticketing system aligns with its design criteria and meets operational needs by executing a comprehensive series of testing cases. This plan covers key operations like ticket booking, user account management, film listing, payment, databases, and the suggestion system.

7.1.2 Test Granularities

7.1.3 Unit Testing

- To test individual units or components in isolation. Unit testing ensures that each component functions correctly and meets its design specifications.

7.1.4 Integration Testing

- To test the interactions and interfaces between various components/modules.Integration testing ensures that the various components interact seamlessly and that data is accurately exchanged between them.

7.1.5 System Testing

- To evaluate the entire system's behavior in a real-world environment. System testing ensures that the system as a whole meets the specified requirements and performs as intended.

7.2 Test Cases

7.2.1 Test Sets and Test Cases

We will test a large number of cases that focus on validating certain activities or functions, usually in the context of a system's overall functionality (ex: validates the complete user registration functionality, ensuring smooth and secure user registration). In addition, we will be making sure the entire product satisfies the standards and operates as intended in the intended setting (ex: checks that the system correctly identifies and displays movie titles based on the search criteria by running the movie search function through a test). Additionally to guarantee that every part of a class's object within the code carries out its designated function when used alone (ex: displaying movie title correctly).

7.2.2 Test case examples:

- Verifies that, upon clicking on a movie's thumbnail, the navigation process functions as intended. (unit)
- Validates the seat selection process and navigation. (unit)
- Assesses various login scenarios when typing in the wrong password, right username or right password and right username. (unit / functional)
- Focuses on handling data, validating input data, and guaranteeing successful transactions during the payment process. (functional)
- Checks that the system correctly displays movie titles based on the search criteria by running the movie search function through a test. (functional)
- Tests the entire ticket purchasing process, including choosing a movie, a seat, and making a payment. (system)
- Ensures a seamless and safe user registration process by validating the entire user registration functionality. (system)
- Evaluates the functionality and appearance of the system across various browsers whether on phone or PC. (system)

TestCaseId	Component	Priority	Description/Test Summary	Pre-requisites	Test Steps	Test Data	Expected Result	Actual Result	Status	Test Executed By
MovieSelect_1	Movie Catalog Page	P0	Verify that clicking on a movie from the Movie Catalog Page takes the user to the Seat Selection Page.	Movie is selected and user is on the Movies Page.	1. Navigate to the Movie Catalog Page. 2. Select the desired movie. 3. Click on the movie to view showtimes. 4. Pick a showtime and theater location. 5. Click on "Proceed" or equivalent action.	1. Selected Movie: "TEST_MOVIE" 2. Showtime: "9:15 PM" 3. Theather Location: "1234 AMC TEST"	The Seat Selection Page opens, showing available seats for the chosen showtime and location.	Seat Selection Page displays available seats for chosen time and location	PASS	OMAR
SeatSelection_2	Showtime and Location Page	PO	Verify that after selecting a showtime and location, the user is redirected to the Seat Selection Page.	Movie is selected and user is on the Movies Page.	1. Navigate to the selected movie's Showtime and Location Page. 2. Pick a showtime and theater location. 3. Click on "Proceed" or equivalent action	Selected Movie: "MOVIE" Chosen Showtime: "3:00 PM Chosen Theater Location: "A	The Seat Selection Page opens, showing available seats for the chosen showtime and location.	The Seat Selection Page opened as expected	PASS	RILEY
SWIFTWAVE_3	Login Page	P1	Verify that registered users can successfully log into the system using valid credentials.	1.User has already signed up and is a registered member of SWIFT WAVE. 2.User is on the Login Page.	1.Navigate to the Login Page. 2.Enter a valid username in the username field. 3.Enter the corresponding valid password in the password field. 4.Click on the 'Login' button.	1. Valid username 2. Valid password	User is successfully logged in and redirected to the Movie Catalog Page or a personalized dashboard.	Login sucessful	PASS	BRIAN
SWIFTWAVE_3	Login Page	P1	Verify that any registered users that types in a incorrect matching password but correct username and cannot successfully log into the system using invalid credentials.	1.User has already signed up and is a registered member of SWIFT WAVE. 2.User is on the Login Page.	1.Navigate to the Login Page. 2.Enter a valid username in the username field. 3.Enter the corresponding invalid password in the password field. 4.Click on the 'Login' button.	1. Valid username 2. Invalid password	User is not successfuly logged in and prompt with message that says "incorrect password or username."	Login unsuccessful	PASS	RILEY
SWIFTWAVE_3	Login Page	P1	Verify any registered users that types in a incorrect username but correct password and cannot successfully log into the system using invalid credentials.	1.User has already signed up and is a registered member of SWIFT WAVE. 2.User is on the Login Page.	1.Navigate to the Login Page. 2.Enter in invalid username in the username field. 3.Enter the corresponding valid password in the password field. 4.Click on the 'Login' button.	1. Invalid username 2. Valid password	User is not successfuly logged in and prompt with message that says "incorrect password or username."	Login unsuccessful	PASS	BRIAN
SWIFTWAVE_3	Login Page	P1	Verify any users that types in a incorrect username and password and cannot successfully log into the system using invalid credentials.	1.User has already signed up and is a registered member of SWIFT WAVE. 2.User is on the Login Page.	1.Navigate to the Login Page. 2.Enter in invalid username in the username field. 3.Enter in invalid password in the password field. 4.Click on the 'Login' button.	1. Invalid username 2. Invalid password	User is not successfuly logged in and prompt with message that says "incorrect password or username."	Login unsuccessful	PASS	BRIAN
PaymentInfo_1	Payment Page	P0	Verify any users's payment information to process the payment	1. User is logged into a registered account. 2. User has added a valid ticket to the cart. 3. User is on the payment page/Cart.	1.Navigate to payment page or shopping cart page. 2. Input a valid card number, cvc, exp date, and card holder name. 3.Valid zip code and billing address. 4. click the confirm payment button	1. Valid payment credentials	The payment is processed and the user is shown a processing prompt followed by a sucessful payment message. The user is then redirected to the ticket details page.	Successful payment	PASS	Omar
PaymentInfo_2	Payment Page	P0	Verify that if a user inputs an incorrect card information then the payment is rejected and the user is informed about the incorrect information.	1. User is logged into a registered account. 2. User has added a valid ticket to the cart. 3. User is on the payment page/Cart.	1.Navigate to payment page or shopping cart page. 2. Input a invalid card number, cvc, exp date, and card holder name. 3. Valid zip code and billing address. 4. click the confirm payment button	1. incorrect card information 2. correct zip code and address	The payment will attempt to process then return the user to the payment page. The incorrect card information will be highlighted red and a message prompt saying incorrect card information has been entered.	Successful payment	FAIL	Omar
PaymentInfo_3	Payment Page	p0	Verify that if a user inputs the wrong zip code or address then the payment is denied and the user is told to input correct information.	1. User is logged into a registered account. 2. User has added a valid ticket to the cart. 3. User is on the payment page/Cart.	1.Navigate to payment page or shopping cart page. 2. Input a valid card number, cvc, exp date, and card holder name. 3.Invalid zip code and billing address. 4. click the confirm payment button	1. correct card information 2. Incorrect zip code and address	The payment will not process and show the user a message prompt stating that either the zip or address is incorrect with a red highlight on the incorrect information.	Failed Payment	PASS	Omar
TicketBooking_1	Ticket		Test the beginning to end flow of the ticket reservation process. Select a movie, showtime, seat, add ticket, and complete a ticket purchase.	1.User is logged in 2.Has a payment method	1. Select a movie 2. Select a showtime 3. Select an available seat 4. Add to cart 5. Go to cart 6. Click pay now button and proceed to payment 7. complete the purchase	1. Movie 2. Showtime 3. Seat 4. payment	The user will recieve a confirmation of purchase and the end to end process works fully.	The process works and confirmation prompt is given	PASS	Omar
MovieSearch_1	Movie Database		Test the movie search functionality so users are able to find movies by their titles.	1. User is on the movie selection page	1. Enter the name of a movie into the search bar 2. Click search or hit enter 3. Exam the search results.	1. Movie Title "Valid Movie Title"	The system will display the movie or movies matching the search query. If nothing matches the system will display no results.	The system displays the movie or movies that match the search.	PASS	Omar
SetterGetterMovieTitleTest_01	Movies Class	P0	Make sure that the Movies class sets and retrieves the movie title correctly.	No pre-requisities required.	1. Create a Movies object. 2. Set the Movies' object title to "THE WALKING DEAD TEST". 3. Use a getter function to retrieve the title from the Movies object.	Movies object that has the title of "THE WALKING DEAD TEST"	The returned function should match whatever the movies object was set to.	The returned movie title matches the set movie title.	PASS	Riley
SetterGetterShowtimes_01	Showtime Class	P0	Make sure that the Showtime class correctly displays a movie's showtimes.	Movie object has to be created in order to have a listing for a showtime.	1. Schedule a showtime for a selected Movies object and fill out information. 2. Verify that the showtime is display correctly for the movie.	Movies object, showtime date, theater location and start & end time.	Correctly displays a movie's showtimes information with given attributes.	Infomation about theater location, startEend time and selected movie is displayed corretly.	PASS	Riley
UserLogin_01	User Registration	P0	Test the integration of user registration with movie selection, ensuring that after user registration, a user can select a movie and associated showtime.	User registration functionality is operational, movies and their respective showtimes are available in the system	1. Register using Userinfo. 2. Log in with Userinfo. 3. Select a movie in Movies. 4. Choose a showtime in Showtimes. 5. Reserve seats in Reserved.	User registration data (username, email, password), selected movie, chosen showtime.	The user can successfully select a movie and its showtime after registering.	The user successfully selects a movie and its showtime after registering.	PASS	Brian
BrowerCompatibilityPhonePC_01	System Browser Compatibility	P1	Test the SwiftWave system compatibility with different browsers across PC or moblie devices.	Any browser available either on moblie or PC devices	1.Access the system using different browsers such as Google Chrome, Firefox or Safari. 2. Perform actions as acessing the website to browser, logging, viewing listed movies, etc...	Browsers (Google chrome, Safari, Firefox, Mircosoft Edge, etc...)	The system should work and display correctly across different web browsers either on PC or moblie devices.	Using chrome on IOS works perfectly, everything display correctly, all buttons performed the correct actions.	PASS	Brian