# Movie Ticketing System

## Software Requirements Specification

## v1.0.0

## 20 February 2025

Group 1
David Hernandez
Ryan Giangregorio
Anthony Krauss

Prepared for
CS 250 - Introduction to Software Systems
Instructor: Dr. Gus Hanna
Fall 2024

# Revision History

| Date | Description | Author | Comments |
|---|---|---|---|
| 20 Feb 2025 | Version 1 | David Hernandez<br>Ryan Giangregorio<br>Anthony Krauss | First Revision |
| | | | |
| | | | |
| | | | |

# Document Approval

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

| Signature | Printed Name | Title | Date |
|---|---|---|---|
| | David Hernandez | Software Eng. | 20 Feb 2025 |
| | Dr. Gus Hanna | Instructor, CS 250 | |
| | | | |

# Table of Contents

# 1. Introduction

The introduction to this Software Requirements Specification (SRS) provides a layout of all the SRS with purpose, scope, definitions, acronyms, abbreviations, references and overview of the SRS. The goal of this document is to provide an in-depth insight and analysis of the complete Movie Ticketing system. The full requirements of the Movie Ticketing system are written in this document.

## 1.1 Purpose

The purpose of the software requirement specification is to outline all aspects of the software's functionality which allows for proper execution and usability. Requirements are presented to communicate the customers' requirements and how those requirements are addressed through software development processes. This document will help developers as they work through the software development lifecycle (SDLC).

## 1.2 Scope

The scope of this document is confined to the features and requirements of the Movie Ticketing system. It primarily focuses on the online and electronic aspect of the Movie Ticketing system, which will process purchases, log data, and display movie information, among a variety of other processes defined in this document. The ticketing system will be user friendly, in the sense that it will be minimalistic and will constantly be updated, allowing the user to easily access and input data as needed.

## 1.3 Definitions, Acronyms, and Abbreviations

PCI DSS – Payment Card Industry Data Security Standard
API – Application Programming Interface

## 1.4 References

IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications.[1]

Your 2025 Guide to Writing a Software Requirements Specification – SRS Document, dated 30 January 2025.[2]

How to Write an SRS Document (Software Requirements Specification Document), dated 17 January 2023.[3]

## 1.5 Overview

The rest of this SRS document gives a description of the system, as well as who uses it, the components needed for it, and the data and code requirements for it as well. In section 2, it will provide a general description of the Movie Ticketing system. Section 3 dives into the specifics of the system, and what each of those specific components requires, both customer side and service side. Section 4 lists the analysis models used to gather and decide on requirements for the Movie Ticketing system.

# 2. General Description

## 2.1 Product Perspective

The Movie Ticketing system will function in relation to a constantly updated movie database(s), credit card companies, payment gateways, cloud hosting service(s), ticket printers, mapping service(s), and theater management systems.

## 2.2 Product Functions

This Movie Ticketing software will have the ability to process payment data, return movie data, alter movie seating data, email receipts, keep and return food purchasing data, and keep a database of user accounts.

## 2.3 User Characteristics

The users include anyone who decides to purchase a movie ticket(s). All the users need to know is how to do a simple aim, click, as well as enter in information, like browsing on other general websites.

## 2.4 General Constraints

Constraints include compliance with the PCI DSS, website compatibility with major web browsers, compliance with privacy laws, and 99.99% uptime.

## 2.5 Assumptions and Dependencies

Users are assumed to have a stable connection to the internet. The purchasing software is dependent on third party payment processers (e.g., PayPal). The movie information is reliant on external movie databases being updated and accurate. It is reliant on the functionality of movie theater ticket kiosks and ticket scanners.

# 3. Specific Requirements

## 3.1 External Interface Requirements

This section specifies user interface characteristics and requirements for interfacing with external hardware, software and communication means which support the systems operation.

### 3.1.1 User Interfaces

The system should present the user with movie posters.
The system should present a cart icon to navigate to the checkout page.
The system should present a profile icon to log in.
The system user interface should be easy to navigate.
The system should utilize a subtle color template.
The system should utilize uniform navigation between screens.
The system should support touchscreen navigation.
The system should include a plus symbol to increase the number of tickets.
The system should include a minus symbol to decrease the number of tickets.

### 3.1.2 Hardware Interfaces

The system should be hosted on a webserver.

### 3.1.3 Software Interfaces

The system should utilize a secure API to process payments.
The system should utilize a secure email sending service.
The system should query a database to present the users' requested information.

### 3.1.4 Communications Interfaces

The system should support an Ethernet internet connection.
The system should support a Wi-Fi internet connection.
The system should support a minimum of 4G cellular network.

## 3.2 Functional Requirements

This section outlines core requirements which ensure the system functions as necessary. Requirements contain input and outputs to provide a clear understanding of the system's functionality.

### 3.2.1 Purchasing of Movie Tickets

3.2.1.1 Introduction
The system should allow the user to purchase a movie ticket.
3.2.1.2 Inputs
The system should allow the user to utilize a payment method.
3.2.1.3 Processing
The system should ensure the user enters valid payment information.
3.2.1.4 Outputs
The system should email the user a receipt of their purchase.
3.2.1.5 Error Handling
The system should validate the user's payment information and provide notification if the payment method was not successful.

### 3.2.2 Purchasing of Concession Items

3.2.2.1 Introduction
The system should allow the user to purchase items from the concession stand.
3.2.2.2 Inputs
The system should allow users to utilize a payment method.
3.2.2.3 Processing
The system should ensure the user enters valid payment information.
3.2.2.4 Outputs
The system should email the user a receipt of their purchase.
3.2.2.5 Error Handling
The system should notify the user when a payment is unsuccessful

### 3.2.3 Displaying Movies

3.2.3.1 Introduction
The system should display all available movies for the user to purchase tickets.
3.2.3.2 Inputs
The system should respond to mouse clicks so the user may see further movie information.
3.2.3.3 Processing
The system should call a new page for the selected movie.
3.2.3.4 Outputs
The system should display a new page containing the clicked movies information
3.2.3.5 Error Handling
The system should notify the user when there are errors navigating to a new page.

### 3.2.4 Seat Selection

3.2.4.1 Introduction
The system should allow the user to select their desired seats.
3.2.4.2 Inputs
The system should allow the user to click the desired seats from an outline of the theater's seating arrangement
3.2.4.3 Processing
The system should ensure that the seat selected has not already been selected by another person.
3.2.4.5 Error Handling
The system should notify the user when selecting a seat that is unavailable.

### 3.2.5 Shopping Cart Functionality

3.2.5.1 Introduction
The system should allow the user to add various items to their shopping cart to be purchased at checkout.
3.2.5.2 Inputs
The system should allow the user a clickable option to add items.
3.2.5.3 Processing
The system should account for every new item that has been added to the shopping cart, including the total number of items and total price.
3.2.5.4 Error Handling
The system should notify the user if an item was not successfully added to their cart.

### 3.2.6 Navigation

3.2.6.1 Introduction
The system should contain a home page which displays a list of available movies
The system should display a summary page when clicking on a movie.
The system should have a checkout page where the user can input payment information.
The system should have an account creation page.
The system should have a page for password reset.
3.2.6.2 Inputs
The system should accept clicks to trigger page navigation.
3.2.6.3. Processing
The system should retrieve the page associated with the user's click.
3.2.6.4 Error Handling
The system should notify the user when there is an error retrieving the requested page.

### 3.2.7 Account Creation/Log In

3.2.7.1 Introduction
The system should allow the user to create an account.
The system should allow the user to log into their account.
The system should accept a password that is at least 10 characters long.
The system should accept a password that contains at least 1 number.
The system should accept a password that contains at least 1 special character.
3.2.7.2 Inputs
The systems should require an email for account creation.
The system should require a password for account creation.
3.2.7.3 Processing
The system should create and store the user's profile to a database.
The system should notify the user that their account was created.
3.2.6.4 Error Validation
The system should validate the entered email address.
The system should validate the entered password.
The system should notify the user if an invalid email address was entered.
The system should notify the user if an invalid password was entered.

### 3.2.8 Account Retrieval

3.2.8.1 Introduction
The system should allow the user to reset their password.
The system should send an authentication email to the users' email address.
3.2.8.2 Inputs
The system should respond to the users' clicks to reset their password.
3.2.8.3 Processing
The system should send an email address to the email address provided
3.2.8.4 Error Validation
The system should notify the user when the email address they entered is not that of an account.
The system should notify the user when an invalid email address is entered.

## 3.3 Use Cases

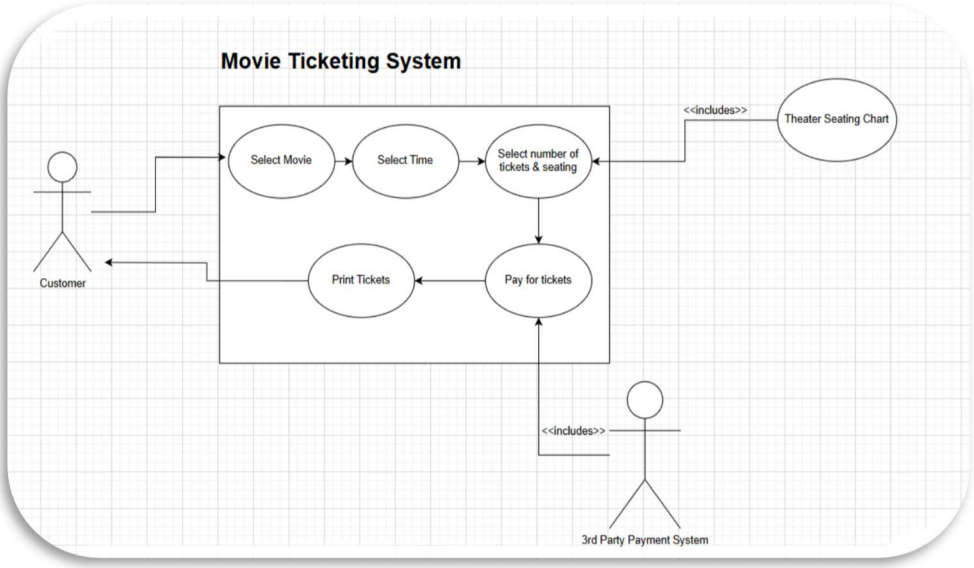
### 3.3.1 Purchasing Tickets No Sign in and Prints Tickets

Name: Movie Ticketing System
Actor(s): User
Flow of events:

1. User navigates to movie ticketing system.
2. User signs in to an account with the theater.
3. User selects a film.
4. User selects an available time for selected film.
5. User selects the total number of tickets and seating selection.
6. User pays for tickets.
7. User chooses to have tickets printed.
8. User receives printed tickets.

Entry Conditions:

1. Users must have successfully accessed the system.



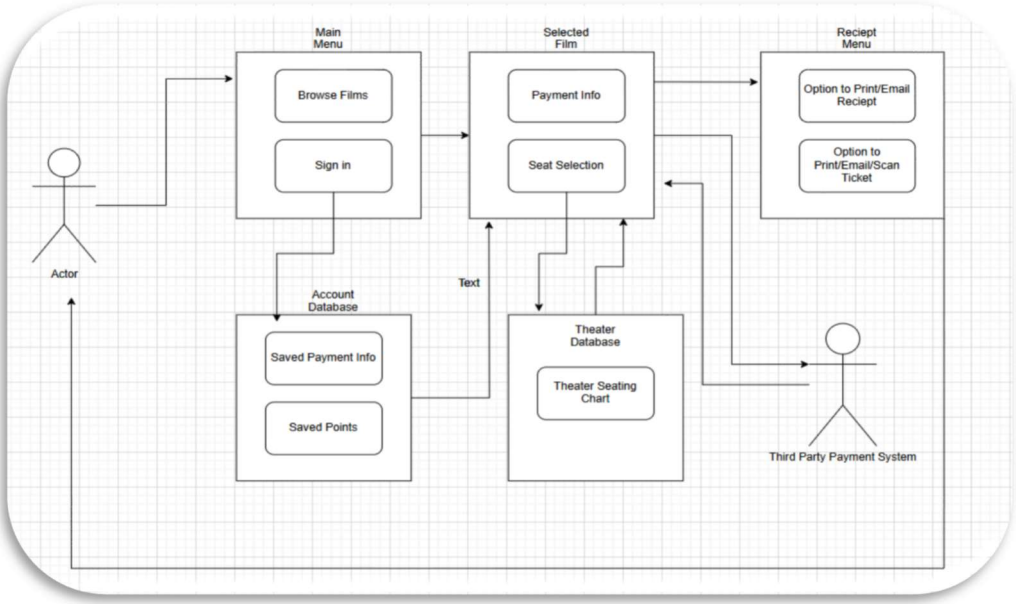
### 3.3.2 User Signs In To Purchase Tickets and Emails Tickets

Actor(s): Customer
Flow of events:

1. User navigates to movie ticketing system.
2. User signs in to an account with the theater.
3. User selects a film.
4. User selects an available time for selected film.
5. User selects the total number of tickets and seating selection.
6. User pays for tickets.
7. User chooses to have ticket emailed.
8. User receives email of ticket.

Entry Conditions:

2. Users must have successfully accessed the system.

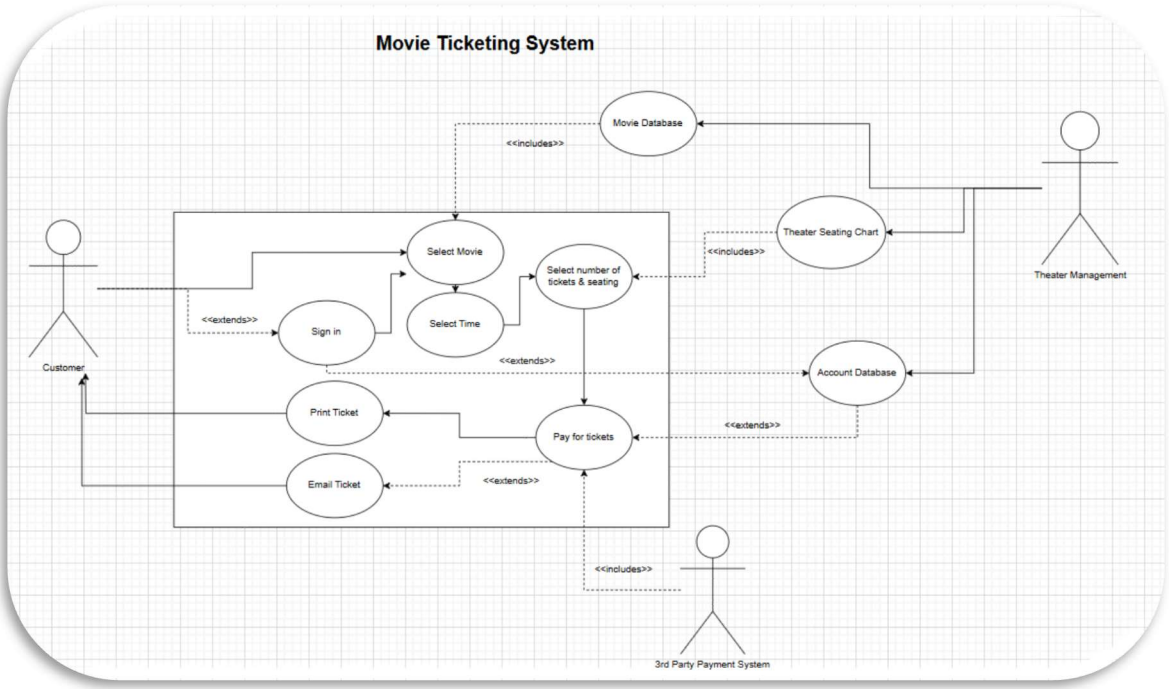


3.2.3 User Creates a Profile
Actor(s): User
Flow of events:

1. User navigates to movie ticketing system.
2. User selects to create a user account.
3. User enters email and enters a password
4. System validates that email and password are in proper format
5. System sends a confirmation email to the user's email
6. User confirms email
7. User information is saved to database

Entry Conditions:

3. Users must have successfully accessed the system.



## 3.4 Classes / Objects

### 3.4.1 Movie

3.4.1.1 Attributes
Name: Movies name
Rating: Movies rating
Length: Play time of the movie in minutes
Description: Brief description of the movie's plot

### 3.4.2 Movie Ticket

3.4.2.1 Attributes
Price:  US dollars
Date: Date movie will be playing
Serial Number: a unique identifier number for each ticket

### 3.4.3 Concession Item

3.4.3.1 Attributes
Name: Identifies the concession item
Price: US Dollars

### 3.4.4 User

3.4.3.1 Attributes
First Name: User's First name
Last Name: User's Last name
Email: To be associated with the account
Password: To access the account
3.4.3.2 Functions
Save: save/update users information to the database

### 3.4.5 Cart

3.4.3.1 Attributes
Movie Tickets: List of movie tickets the user has added to the cart
Concession Items: List of concession items the user has added to the cart
Total Price: US Dollars the user will pay at checkout
3.4.3.2 Functions
Checkout: Calls the payment API to process payment
Add Item: Adds the passed item to the cart

### 3.4.6 Payment Information

3.4.3.1 Attributes
Card Number: Payment card's number
Routing Number: Bank's routing number
Account Number: Bank's account number
3.4.3.2 Functions
Save: Save payment information entered to the database

### 3.4.7 Movie Theater

3.4.7.1 Attributes
Number: Identify the theater
Available Seats: List of available seats available in the theater
Movies: List of movies that are playing at this theater
3.4.3.2 Functions
RemoveSelectedSeats: Pass the selected user seats and remove them from list of available seats
ResetSeatAvailability: After the movie has shown, resets the list of available seats

### 3.4.8 Seat

3.4.8.1 Attributes
Row: Assigns an alphabetic value to the seat from A-H
Number: Assigns a number to the seat from 1-9

## 3.5 Non-Functional Requirements

The following requirements outline the behavior of the Movie Ticketing System. These requirements allow for assessing and measuring how the system performs as it is accessed and operated by the user.

### 3.5.1 Performance

The system should ensure that all pages are loaded within 3 seconds.
The system should operate at a minimum internet speed of 10mbps.
The system should operate on all major internet browsers.

### 3.5.2 Reliability

The system should operate with at least a 95% level of reliability.
The system should implement reliable database architecture.
The system should utilize a reliable web-based framework.

### 3.5.3 Availability

The system should be accessible at any time of day.
The system should be hosted on at least 2 servers for redundancy.
The system should accommodate at least 200 simultaneous users.

### 3.5.4 Security

The system should utilize a secure payment processing system.
The system should encrypt all user information.
The system should provide a secure means to retrieve passwords.

### 3.5.5 Maintainability

The system should allow for new items to be updated effectively.
The system should be modularized to increase debugging efficiency.

### 3.5.6 Portability

The system should be accessible on PC, Mac, and all mobile devices.

### 3.6 Inverse Requirements

The system should not allow a user to retrieve their password if they have confirmed their identity.
The system should not allow a user to buy tickets for a movie that is full.
The system should not allow a user to select a seat that has already been selected.

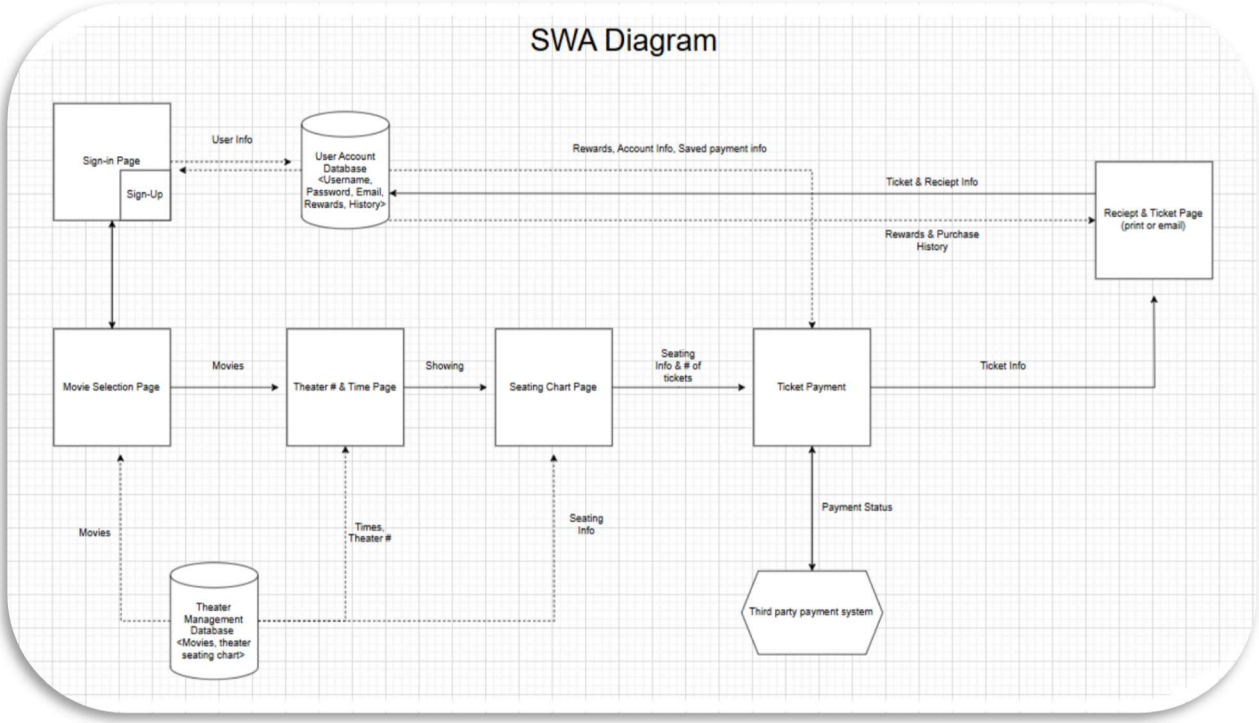
## 4. Software Design Specifications

### 4.1 Overview

This specification for the Movie Ticketing system outlines the system's architectural framework, and components' interactions with each other. The section gives an alternate view of the system, specifically aimed towards helping developers understand its implementation. The system uses an object-oriented design, helping to manage primary "objects" such as users, shopping carts, and payment information. The Software Architectural Diagram depicts, in a general sense, how the user interface will look, but also how the developer builds that interface, and how each of the components of it interact. The UML Diagram gives a detailed look at the classes and their attributes and methods. The development of this system will be divided into separate tasks, with a six-week timeline of completion. Ryan will handle the user authentication, as well as database integration, taking around three weeks. David will develop code for the movie selection page, seat reservation page, and the ticketing system; his work will last approximately four weeks. The payment processing and receipt generation will be taken care of by Anthony in three weeks. Following the completion of their respective tasks, team members will collaboratively test and debug for two weeks before deployment of the system.

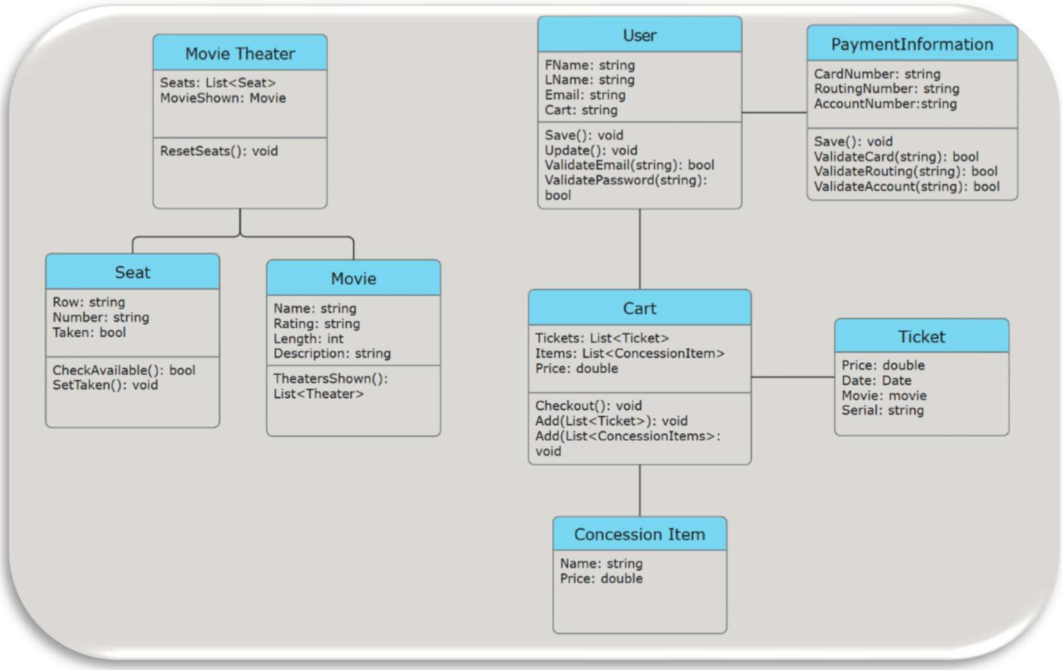
## 4.2 Software Architectural Diagram

The given Software Architectural Diagram illustrates a high-level visual depiction of the main components and how they interact with one another. The components consist of databases, web pages, and processes. It highlights modules such as user authentication, movie selection, reserving seats, and payment processing.



## 4.3 UML Diagram

The provided Unified Modeling Language (UML) Diagram provides a visual representation of the system design. Each container represents a class. Each class contains its variables, data types, and methods and methods' return type. A description of each class, variable and method is included below:




Movie Theater
Seats: A List of type Seat. This list contains all the available seats within the theater. Each seat must be tracked as assigned or unassigned.
Movie: A theater will be assigned a Movie which is currently playing

ResetSeats(): Allows for resetting all seats to available once the movie has finished playing

Seat
Row: This string will be a single character from A-H to represent the row placement.
Number: This string will be a single number from 1-9 to represent the seat number within the row.

CheckAvailable(): Checks if the seat is available. Returns true if available and false if it is taken.
SetTaken(): Sets the bool for taken to true.

Movie
Name: this string will store the movie's name
Rating: string will store the movie's rating allowing users to consider the movies content.
Length: an integer to show the movies running length time in minutes so the user may know how long they should expect to be at the theater

TheatersShown(): Returns the list of theaters which are playing the movie.

User
Fname: string to store the user's first name
Lname: string to store the user's last name
Email: string to store the users email
Password: string to store the users password
Cart: an instance of the cart class to store the items the user wishes to purchase

Save(): save the user's information to the database. Create a new entity in the database. Returns no value.
Update(): update existing user's information. Returns no value.
ValidateEmail(string): process and validate the passed email. Returns true if email is valid, otherwise returns false.
ValidatePassword(string): process and validate the passed password. Returns true if password is valid, else returns false;

Cart
Tickets: List of data type Tickets used to store the movie tickets the user has added to his cart.
Items: List of data type Items used to store the items the user has added to his cart.
Price: double variable to store the total price the user will have to pay at checkouts

Checkout(): adds up the price of all tickets and items in cart and prompts the user for payment.
Add(List<Ticket>): passes a list of tickets and adds each item to the cart's Ticket List
Add<List<Items>): passes a list of items and adds each item to the cart's Item List

Concession Item
Name: string to store the concession items name
Price: string to store the concession item's price

Ticket
Price: double to store the tickets price
Data: Data to store the date the ticket's movie will show
Movie: a ticket will be assigned a Movie class object to represent the movie the ticket was purchased for.
Serial: a unique identifier number to validate ticket

PaymentInformation
CardNumber: a string to store the user's credit cart number
RoutingNumber: a string to store the user's bank's routing number
Account Number: a string to store the user's bank's account number

Save(): method to save the user's payment information
ValidateCard(string): returns true if card number is valid, else returns false.
ValidateRouting(string): returns true if routing number is valid, else returns false.

ValidateAccount(string): returns true if account number is valid, else return false.

# 5. Test Plan

## 5.1 Test Plan Overview

This test plan provides the strategy and scope for the specific test cases laid out in the Excel spreadsheet; the test cases are used to confirm that the Movie Ticketing System works according to the SRS. The goal here is to make sure that all requirements (e.g. making and processing payments, picking seats, selecting movies, etc.) are tested at all levels.

## 5.2 Test Objectives

- To make sure that each class (component) functions as specified in the SWA
- To confirm correct module interaction (e.g., User to Cart to Ticket, Movie Theater to Seat or Movie)
- To make sure that the system meets requirements such as high performance, good reliability, and strong security.
- To eventually identify and document any issues with the software and anything out of the norm

## 5.3 Scope of Testing

In-Scope:
- Functional Requirements: Purchasing of movie tickets, purchasing of concession items, displaying movies, seat selection, shopping cart display, account management, and payment processing.
- Non-functional Requirements: Performance, reliability (availability), security

Out-of-Scope: Third-party payment service quality, physical printing of tickets

## 5.4 Test Strategy

Our testing strategy will cover three main levels:
1. Unit Testing
   - Focusing on testing the individual classes and their methods and variables
   - Performed as each class is implemented
2. Integration Testing
   - Checks the functionality of interactions between the related components (e.g., checking that adding a seat to cart correctly updates the availability of seats for the next user)
3. System Testing
   - Makes sure that the entire system (the UI and the backend) checks off the SRS requirements
   - Includes the usability, performance, and functional checks in a production simulated environment

## 5.5 Test Environment

OS: Test on Windows 10/11, macOS, mobile devices (iOS, Android)
Software:

- Browsers: Chrome, Firefox, Safari, Edge
- Database: MySQL
- Payment API: virtualized and sandboxed environment for credit card validations
- Tools; JUnit, Selenium (for UI automation tests)

### 5.6 Test Plan

Refer to the link below for the software test plan. The test plan is an excel spreadsheet which outlines testing steps along with expected results to ensure the system operates as expected.

Test Plan GitHub link: github.com/davidmht1

### 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.*

## 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