Movie Ticketing System

Software Requirements Specification

1.1

9/19/2024

Group 12

JJ Hafidh, Jianna Gapuz, Deepanshu Joshi, Claudia Gonzalez

Prepared for

CS 250- Introduction to Software Systems

Instructor: Gus Hanna, Ph.D.

Fall 2023

# Revision History

| **Date** | **Description** | **Author** | **Comments** |
| --- | --- | --- | --- |
| 09/19/2024 | TicketingSystem1 | Group 12 | First Version. Completed sections 1 through 3.7 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Document Approval

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

| **Signature** | **Printed Name** | **Title** | **Date** |
| --- | --- | --- | --- |
|  | 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 <Functional Requirement or Feature #1> 3*

*3.2.2 <Functional Requirement or Feature #2> 3*

3.3 Use Cases 3

*3.3.1 Use Case #1 3*

*3.3.2 Use Case #2 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

# 1. Introduction

Our movie ticketing system is a platform for users to browse and look for movies that they want to watch, they should be able to select seats and purchase the tickets as well. You can also log in as a user or checkout as a guest. Admins can adjust movies through the admin login and seat availability, and generate reports.

## 1.1 Purpose

The purpose of the document is to define and document the requirements of the software system, taking into account the needs of the consumers. This document predicts its intended use by the consumers and provides scenarios of how the system will be utilized. This SRS aims to specify the requirements of the software product, including its parameters and goals and it will be a foundation for development.

## 1.2 Scope

This is a system that will enable the consumers to access all functionalities such as purchasing tickets, signing up for rewards, allowing users to create accounts, accessibility features, and more. The website can not authenticate if the consumer is under the age of 18 or under the age of 21 to see a restricted film.

## 1.3 Definitions, Acronyms, and Abbreviations

FAQ: Frequently asked questions.

SRS: Software Requirements Specification

Admin: Administrator which is an authorized personnel who manage the system's backend operations

API: Application Programming Interface

## 1.4 References

IEEE SRS Standard (1998)

Perforce - How to Write an SRS Document (Software Requirements Specification Document)

Relevant Software - Your 2024 Guide to Writing a Software Requirements Specification – SRS Document

AMC Theatres

## 1.5 Overview

The SRS provides all the necessary information to start development including all its main functionalities and broad design features.

Section 1 introduces the SRS document outlining the software’s purpose, scope, abbreviations, references, and this section.

Section 2 provides a high-level overview of the system like how it interacts with other systems and it summarizes all the major functions of the software, it also states all the factors that the system depend on.

Section 3 details the explicit requirements of the system and it breaks it down into many different sections which includes the interfaces between the system and the users, hardware, and software. It also lists how users will interact with the system and what those functionalities can do.

Section 4 is analysis models which provides models to show the system’s structure and what it does.

Section 5 is the change management process which outlines the changes made to the requirements throughout the development of the system.

Section A is used to supplement material that supports the SRS.

# 2. General Description

## 2.1 Product Perspective

The online ticketing system will be interacting with an interface when securing payment details and detailing any bots from spamming a purchase of a large amount of tickets. However, the users will be able to access these features only through a device that has the ability to access a web browser.

## 2.2 Product Functions

* Purchasing tickets
* Creating an Account containing personal information
* Ticket and additional purchased item descriptions
* The ticketing system will allow the user to search for a movie by title, genre, category or time.
* The ticketing system should create and manage accounts for users to allow them to save their payment methods, view order history and for points. It should also manage an admin account in which you’re able to change movie titles and times.
* The ticketing system should allow users to sign up for rewards and redeem their points.
* The ticketing system allows users to enable accessibility features like zoom, enhanced texts, read-out-loud, and other languages.
* The ticketing system asks the user for their age when purchasing age-restricted tickets.
* The ticketing system allows users to order snacks while purchasing their tickets.
* The ticketing system allows users to inquire about private bookings like birthdays or private shows.

## 2.3 User Characteristics

The ticketing system will not teach the users how to get to the website, but when accessed, there will be clear instructions to guide the users through the many functionalities and features of the website. These include incorporating a “Sign In” function, a “Get Tickets” function, a “Search for a Location Near You”, and more. The website will be able to handle different types of users such as admin and customers whether it to be to fix a certain bug on the website or purchasing tickets.

## 2.4 General Constraints

Rules and regulations are necessary for the completion of the online ticketing system and the ones mentioned are only a general description of what the developers may face. These constraints include safety and security restrictions, hardware limitations, accessing any API’s needed, control functions, and higher order language requirements.

## 2.5 Assumptions and Dependencies

We will assume that this is an online ticket purchasing system and is used in the following application:

* Prompting the user for an account login where they will either log in or proceed as a guest.
* They will then search for movies, while asking for their age, and display available show times and dates.
* They will then select the date, the time, and how many tickets they wish to purchase
* When they pay, it will ask for a confirmation and the user will apply for any discounts if applicable.
* Confirmation will be shown and confirmation email will be sent immediately.

Therefore, we have successfully developed a system that will enable users to purchase tickets online along with all of its functionalities.

# 3. Specific Requirements

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

The user interface will be compatible with all major browsers including but not limited to Google Chrome, Mozilla, Safari, Opera, and Internet Explorer.

This interface may be implemented in JavaScript, React, or any web development software package.

### 3.1.2 Hardware Interfaces

The hardware necessary to run the website must have internet connecting capabilities. This can be through Modems, WAN/LAN, or Ethernet cable connections.

### 3.1.3 Software Interfaces

1. The ticketing system can communicate with the theater/content manager to edit showing times, prices, and available discounts.
2. The ticketing system can communicate with a payment gateway system to authenticate and validate the payment method chosen by the customer.
3. The ticketing system can communicate with a customer relationship management system to provide support
4. The ticketing system can communicate with an automated ticket delivery system and order fulfillment system to generate and deliver movie tickets purchased by the customer.
5. The ticketing system can communicate with an external tax system to calculate appropriate tax and exempt customers that are eligible.
6. The ticketing system can connect to a secure transaction processing system to allow users to complete a secure transaction.

### 3.1.4 Communications Interfaces

The movie ticketing system will use HTTP protocol to communicate over the internet.

## 3.2 Functional Requirements

This subsection contains the functions and features that the movie ticketing system possesses. Features from this section are refined and used in case diagrams to visualize how the requirements of the system would flow.

### 3.2.1 Provide Search Facility

3.2.1.1 The system shall allow the user to search by genre.

3.2.1.2 The system shall allow the user to search by actor names.

3.2.1.3 The system shall allow the user to search by movie titles.

3.2.1.4 The system shall allow the user to search by showing types such as iMax, 3-D, or regular screenings.

3.2.1.5 The system shall allow the user to filter search results by genre, age restrictions, showing times, locations, languages, and screening types.

### 3.2.2 Previewing a Specific Movie Title and Details

3.2.2.1 The system will show a prompt asking if the user meets age restriction requirements if applicable.

3.2.2.2 The system shall allow the user to view the title of the movie and its poster.

3.2.2.3 The system shall allow the user to view the trailer of the movie.

3.2.2.4 The system shall allow the user to see available seats.

3.2.2.5 The system shall allow the user to see keywords relating to the theme of the movie.

3.2.2.6 The system shall allow the user to view showing times.

3.2.2.7 the system shall allow the user to view nearby locations that are showing the currently selected movie.

3.2.2.8 The system shall allow the user to change to their preferred showing type if applicable.

3.2.2.9 The system shall show the top 10 returned search results.

3.2.2.10 The system shall notify the user if there are no movie titles that match the search prompt.

### 3.2.3 Customer Data Management

3.2.3.1 The system shall allow users to create a profile and set their credentials.

3.2.3.2 The system shall allow users to view the history of their profile, including past movies and transactions.

3.2.3.3 The system shall allow users to update their profile information and change their credentials.

### 3.2.4 Customer Support

The system shall give users access to customer support with the following cases.

3.2.4.1 The system shall allow users to connect to a specific location’s contact.

3.2.4.2 The system shall allow users to connect to the external payment system for disputes or inquiries on charges.

3.2.4.3 The system shall allow users to provide feedback on their experience at a location.

3.2.4.4 The system shall allow users to view FAQs.

### 3.2.5 Provide Shopping Cart Facility

The system shall allow users to view items to their shopping cart. The system shall allow users to add or remove items.

### 3.2.6 In-Theater Menu Display

3.2.6.1 Customers can view the snacks and refreshments menu online.

3.2.6.2 Customers can place orders for snacks and refreshments ahead of time to avoid waiting in line.

### 3.2.7 Email Confirmation

The system shall hold customer email information as part of their customer profile. The system will send confirmation to the user’s email address.

### 3.2.8 Customer Invoice

The system shall display the invoice for an order once it has been confirmed. This invoice will also be sent to the customer’s email address.

### 3.2.9 Allow Order Modification or Cancellation

3.2.9.1 The system shall allow the customer to modify an order’s details such as seat numbers or ticket quantities if available.

3.2.9.2 The system shall allow the customer to cancel an order and will issue refunds with accordance to the refund and cancellation policy.

### 3.2.10 Allow Event Booking

The system will provide the option to book a theater, multiple theaters, or the entire facility for events. Finalization of the booking will be through an employee of the theater.

### 3.2.11 Offer Promotions

3.2.11.1 The system will show promotions and deals on the homepage of the website.

3.2.11.2 The system will send emails to subscribed customers about offers and deals.

3.2.11.3 The system will display promotions at the checkout page.

3.2.11.4 The system will automatically apply promotions that apply to all users at checkout.

### 3.2.12 Online Purchase of Products

3.2.12.1 The system will display all contents of the shopping cart previous to checking out.

3.2.12.2 The system shall allow the user to confirm their purchase.

3.2.12.3 The system shall allow the user to enter payment details.

### 3.2.13 Allow Interface Modification for Accessibility

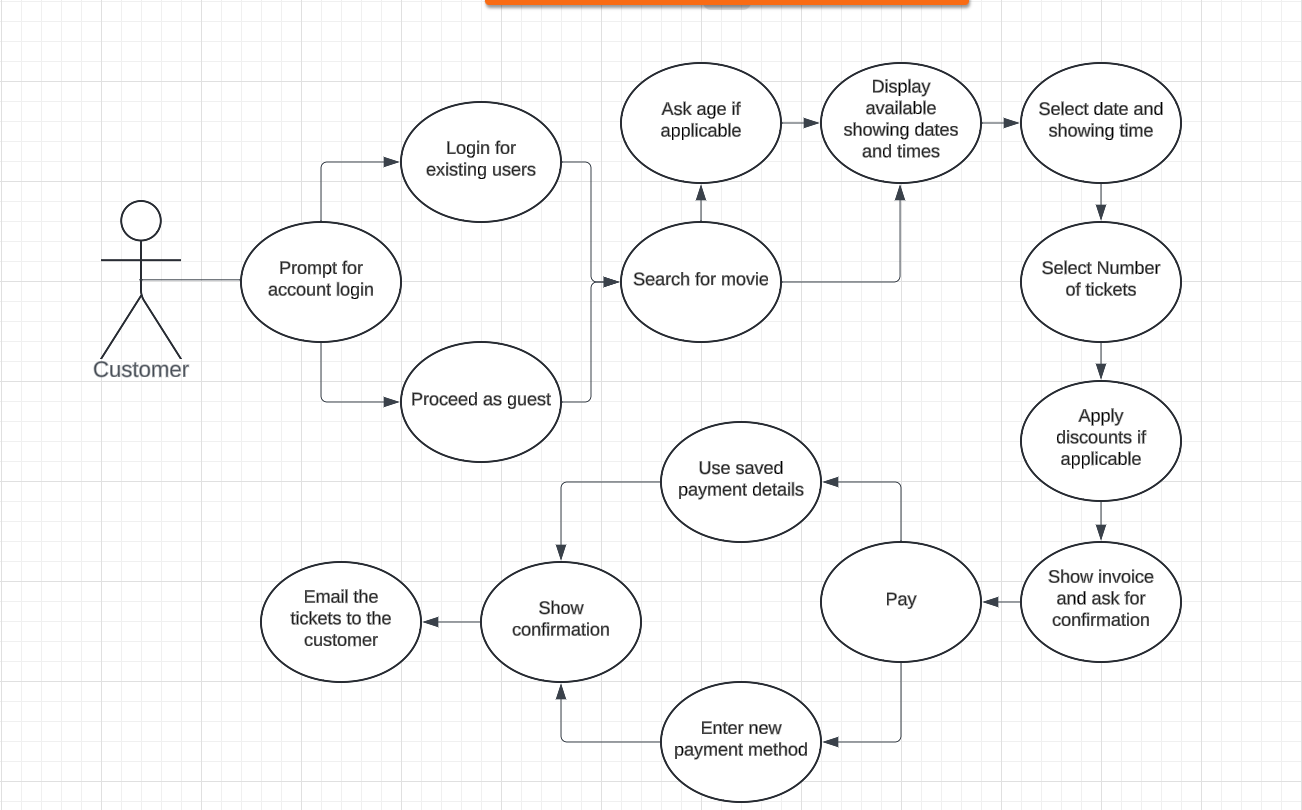
The system shall allow the user to toggle accessibility features for the customer.

3.2.13.1 The system shall allow users to toggle larger text

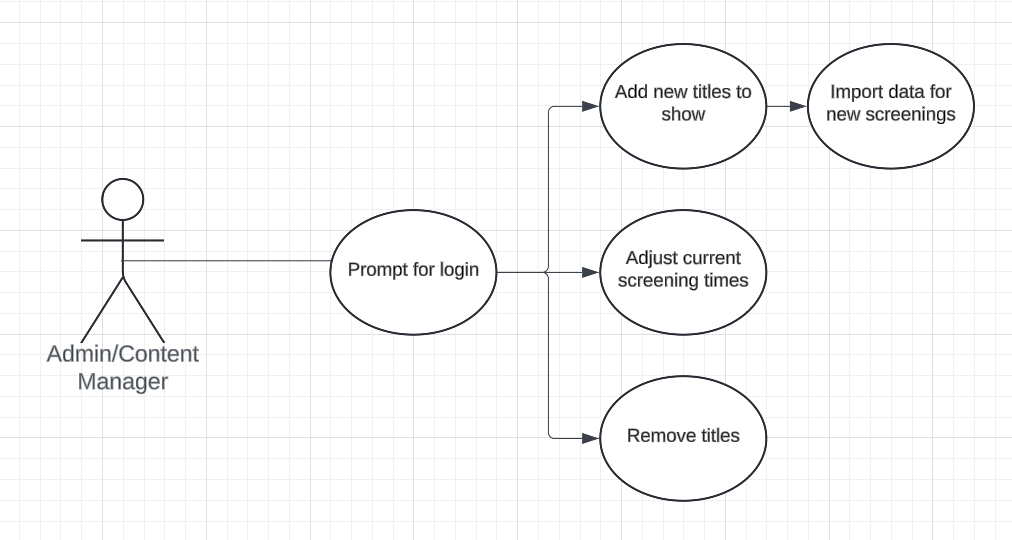
3.2.13.2 The system shall allow users to adjust the website's color scheme for color vision deficiencies.

## 3.3 Use Cases

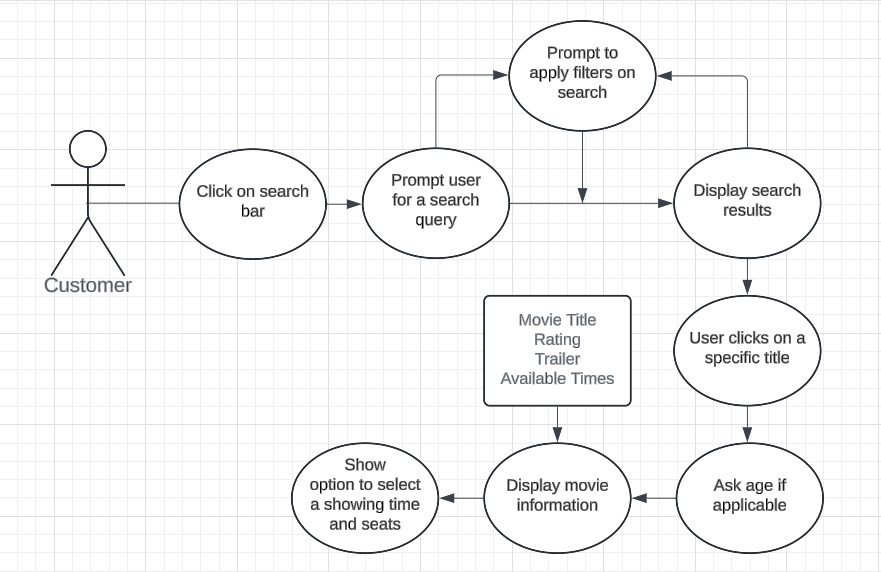
### 3.3.1 Buying a Ticket

Figure 1: Displaying the process of a customer purchasing a ticket on the online ticketing system.

### 3.3.2 Admin or Content Managers Editing Movie Showings

Figure 2: Displaying how an administrator will adjust information on the online site.

### 3.3.3 Search Movie Database

Figure 3: Displays the process of how the customer will search and view what movies are being shown.

## 3.4 Classes / Objects

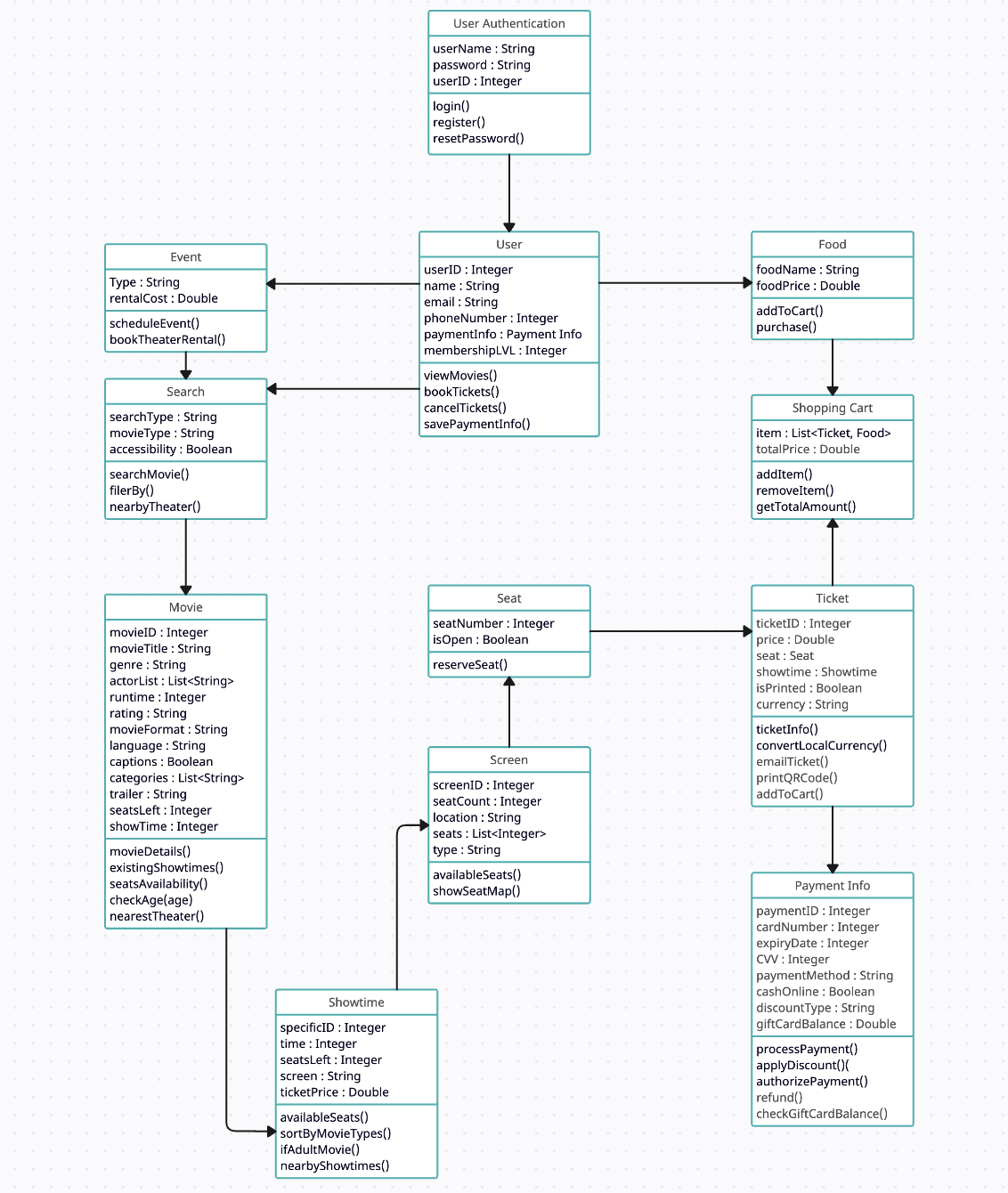


Figure 4: Displays the classes and objects that will be utilized when accessing the online ticketing system in a UML diagram.

**UML Diagram**

**Description below of the different classes, attributes and operations of UML Diagram**

**Class**:

**User Authentication**

The class user Authentication is meant to find out who the user is. It is our login page that is going to be shown if a user is trying to buy a ticket or do anything where knowing their identity would be important to the website.

**Attributes:**

**userName : String**

Username of the user that is saved in the system. Used for the logging in process by the user

**password : String**

Password of the user that is encrypted in our system and is entered by the user to verify that they are who they say they are

**userID : Integer**

The userID is how we actually find out who someone is. Everyone in our system is going to be given a specific number such as 4534231 which is going to be their user ID. This ID is going to be how we identify who is who

**Operations:**

**login()**

This operation is just the user signing into their account if they already have one. By entering their username and password we can identify who they are and keep track of all of their purchases and other transactions using their userID.

**register()**

The register operation is made so that users not already assigned a userID by the system can create a new account and get a userID. The user would be able to choose their username and password and if they are able to complete all the steps of registration successfully then the account will be assigned a userID.

**resetPassword()**

In the case that the user forgets their password we would hate for them to have to create a whole new account. As a result we have the reset password option where they will receive an email to their registered email account or text to their phone number where they will be able to change their number.

**Class:**

**User**

This class is the storage of user information. Here we store information that the user inputted to create their account. Alongside that we will also be storing additional information with the consent of the user. Things that will make the movie ticket buying process easier such as having

your credit card information or membership level will be stored in this class to make it easier to automatically fill the information later.

**Attributes:**

**userID : Integer**

Again as stated above the user ID is just how we get to recognize the user. Each account is assigned a unique ID and the ID that is assigned to this account is also stated in the user tab.

**name : String**

The user name will be defined in the User class to make it easier to verify the name of who is actually buying the movie tickets.

**email : String**

The email address of the user will be stored just so that we can send the confirmation email of a purchase to the user. Additionally if they decide to opt in we can also send promos or coupons for future purchases.

**phoneNumber** **: Integer**

The phone number is saved for almost the same reason as the email address. In the case that the user ends up buying a ticket we can also send them a confirmation text message to their phone if they would prefer that. The promos and coupons can also be sent to the phone instead of the email.

**paymentInfo : Payment info**

The payment information is something that we store just to make the checkout process easier for the user. By storing their payment information instead of having to deal with the hassle of something such as entering their credit card information every time they want to buy a movie ticket, they can just click once and they will have all their information filled in for them.

**membershipLVL : Integer**

Within our system we have a membership that users can either decide to opt in for or they can just buy tickets regularly. This integer that is saved in the system just shows what tier of membership they purchased so we can give them discounts and deals accordingly.

**Operations:**

**viewMovies()**

This operation stands to show what movies the user has already booked tickets for. You will be able to see both past movies that the user has already seen alongside movies that the user purchases tickets for but hasn’t gone to see yet.

**bookTickets()**

The operation just takes the user to the area where they can buy tickets for a movie through our search feature. From there they can find a movie that they would want and make the customizations that they want such as seat number before actually buying their tickets.

**cancelTickets()**

In the case that a user for some reason wants to cancel tickets we have an option where they can cancel their ticket. This would then remove the ticket from the users viewMovies() operation, give a refund to the user and open up the seats again back on the movie search tab.

**savePaymentInfo()**

Saving the user’s payment information within their user tab. This allows them to save their method of payment to their account so that they can easily checkout. They can use various different forms of payment such as paypal or a card.

**Class:**

**Event**

This class is meant for when there is an event that is being planned. If a birthday party or something similar is being held at our theater then there are some extra steps that we take to make the person feel more special. As a result there are some additional costs that come in to form of rentals that are also added to the subtotal.

**Attributes:**

**Type : String**

When we are setting up a room for a party we want to make sure that we are setting up for the correct occasion. As a result we take in a description from the user of what their party is about so that we can better understand how to accommodate them.

**rentalCost : Double**

Since we are going above and beyond for our customers we have to charge them a little bit more due to them using a little bit more of our facilities than other people

**Operations:**

**scheduleEvent()**

When we are hosting our customers we need to make sure we have enough rooms and we need to know when they are coming. As a result we have a scheduled event operation where we can get detailed information on how the party is going to go so that we can make sure everything runs smoothly.

**bookTheaterRental()**

In the case of birthday parties we offer them the option of watching any movie that they want. As a result they have to book the whole theater since they are going to be the only people watching.

**Class:**

**Search:**

When looking for a movie there are certain things that you can filter. This filtering will be done through our search function. You can search for a specific type of movie that you came into the site looking for or you can look for a type of movie that you want.

**Attributes:**

**searchType : String**

Finding out what kind of search you are doing. If you are looking for food or movies.

**movieType** **: String**

If you don’t know exactly what movie you want to watch you can use the movie type option to select a genre or mood of the movie. By entering something such as Fantasy we will show movies that are currently showing in the fantasy genre such as Harry Potter.

**Accessibility : Boolean**

For movies we will occasionally have screenings that have captions on or off. If having captions on is a necessity for you then you will have to look at the accessibility filter to see if we offer captions for the selected title.

**Operations:**

**searchMovie()**

This is the actual search operation. In this our search engine will take input from the user about what movie they are looking for and display that movie alongside movies that we think they would like because they are either similar to the searched movie or movies that we think they would like

**filterBy()**

This is our filtering operation. If the user has to have subtitles on or wants to see a movie about wildlife specifically they can toggle those options on and they will be presented movies that have those features.

**nearbyTheater()**

If a user finds a movie that they want to watch with the correct filters then our system will find the closest theater that is playing that movie with the appropriate filters.

**Class:**

**Food**

This class is for when you specifically want to buy food. It will both display the price and name of the option alongside adding the item to your cart.

**Attributes:**

**foodName: String**

Here we display the name of the food in the form of a String for the user to see

**foodPrice : Double**

Here we display the price of the food item in the form of a double for the user to see

**Operations:**

**addToCart()**

If the user likes a food option then the operation will be able to take the selected quantity of the food item that the user wants and adds it into the shopping cart alongside any items the user may be purchasing

**purchase()**

In the case that the user wants to purchase they can just purchase the food item directly and receive a ticket for the food tem.

**Class**

**Shopping Cart**

The shopping cart class is used to hold data about what the user wants to buy. If the user is considering buying items such as tickets or even food they can just add the items into the shopping cart and then checkout all at once.

**Attributes:**

**Item : List<Ticket, Food>**

This list is in place to keep track of what all is in the cart. Any ticket or food related item that the user decides they might consider buying will be added to this list to be kept track of.

**totalPrice : Double**

While the list above is in charge of keeping track of what items the user is trying to buy, this variable is just going to be the total price of all the items that the user has entered into their shopping cart.

**Operations:**

**addItem()**

This operation is simply the user adding an item to their cart. The operation will take the item that the user wants, add the name and quantity to the list while adding the price into the totalPrice.

**removeItem()**

When the user decides that they many not want an item that they had previously added to the cart this operation will both remove that item from the list and deduct the price from the totalPrice

**getTotalAmount()**

This operation is just going to get the value from totalPrice and display the price to the user as that is how much money they are going to be charged if they decide to purchase everything in the cart

**Class:**

**Ticket**

Class holding all the information for a specific ticket that a user buys. Once they buy the ticket it is going to be stored in a ticket ID so the information can be easily accessible and displayed anywhere.

**Attributes:**

**ticketID : Integer**

The ticket ID is an identification number attached to a ticket to identify things such as what movie it is for and what time it is for.

**price : Double**

The price for a ticket that the customer bought

**seat : Seat**

The seat that the ticket is for

**showtime : Showtime**

Specifics about the movie like where it is being shown and a age rating for the movie

**isPrinted : Boolean**

Showing if the ticket that the customer is going to bring is digital or if it is a physical ticket

**currency : String**

The currency that is being used to buy the ticket

**Operations:**

**ticketInfo()**

Sorting all the information that is important onto a either digital or physical copy of the ticket

**convertLocalCurrency()**

Converting whatever currency that the user has paid in to the currency of the local theater

**emailTicket()**

Email a copy of the ticket to the user just so that they have a backup in case anything happens

**printQRCode()**

Generate a qr code that can be scanned at the door as proof of ticket

**addToCart()**

Add ticket to cart so that the customer can check it out later

**Class:**

**Movie**

This is the big class. This is where everything pertaining to the specific movie booth is recorded. Everything that the user would need to book their tickets and a good description of the movie viewing is going to be saved here.

**Attributes:**

**movieID : Integer**

Just like above how each user has a unique userID each movie screening has a specific movieID. This ID is how we will recognize which booth we are talking about at a certain time.

**movieTitle : String**

Just as the variable name suggests this is simply the name of the movie that is being shown

**genre : String**

What genre the movie that is being shown is

**actorList : List<String>**

The list of the actors that are present in the movie

**Runtime : Integer**

How long the movie is

**Rating : String**

The rating of the movie was given by the Motion Picture Association. Using this there will be monitors of who is buying tickets depending on their age. (M, PG13, G)

**movieFormat : String**

How the movie is being shown. Some users may prefer IMAX or 3D for certain movies over a regular movie watching experience

**Language : String**

The language that the movie is going to shown in

**captions : Boolean**

Showing to the user if the captions are going to be on or if theta are going to be off

**categories : List<String>**

This list is meant to demonstrate what types of categories the movie fits into. Since movies can often fit into multiple categories this variable has to be stored as a list

**trailer : String**

Link to the official move trailer that is going to offer a sneak peak into the movie

**seatsLeft : Integer**

Showing how many seats are left in the theater for purchase

**showTime : Integer**

The times that the movie is going to be showing at different theaters

**Operations:**

**movieDetails()**

To make it easier for the user we will be bringing up all the important movie details to the user in one page

**existingShowtimes()**

In the case that a user is actually considering watching one of our movies we will find all the showtimes that have open seats and display those to the user

**seatAvailability()**

Finding out which seats are available in what movies theaters they are available

**checkAge(age)**

In the case that a movie is rated something like M where it isn’t suitable for all audiences we will display this information to inform the user

**nearestTheater()**

After you decide what kind of movie you would like to watch with all of your preferred customizations, this operation will help to find the movie theater that is closest to you.

**Class:**

**Seat**

The seat class is used to aid the user when it comes to choosing a seat and reserving it. The user will be able to see if the seat is open or reserved and if open, they will be given the option to reserve it, but not buy it.

**Attributes:**

seatNumber: For this attribute, the seat numbers will be labeled using integers on the online ticketing system website.

**isOpen**: Boolean was utilized for this attribute so the user will be able to see whether a seat is open or not.

**Operations:**

**reserveSeat():** This function will allow the user to choose what seat they want to reserve and hold that reservation until it comes to paying for the seats. The seat will be completely reserved when the user pays for the seats chosen.

**Class:**

**Screen**

The screen is just the specific movie theater booth that the movie is being shown at

**Attributes:**

**ScreenID : Integer**

The ID of the specific room that the movie is going to be shown in

**SeatCount : Integer**

This attribute will be an integer which will serve as displaying the numbers that are attached to the seats.

**Location : String**

The road address of the location where the movie is going to be shown

**Seats : List<Integer>**

This will display a list of the seats since they are in an aisle on the screen.

**Type : String:**

**Operations:**

**availableSeats():** This function will illustrate where the available seats are on the screen. There will be an indicator that will show the user whether the seat is available or vice versa. If they choose a seat that is unavailable, they will not be allowed to continue and told to choose another available seat.

**showseatMaps():** This function will show the overall structure of the movie theater seating. It will display where the screen is located, the available seats, the unavailable seats, handicapped seats, and the seats they want to select.

**Class:**

**Payment Info**

The payment Information class is present so that information about the users preferred form of payment can be stored into the system. By doing this the checkout process becomes much easier and faster for both the user and the system.

**Attributes:**

**paymentID : Integer**

The paymentID is just an ID that is given to every transaction that happens between the user and the system. Within the payment ID information about the transaction is going to be saved in the case of something such as a system crash, refund or fraudulent charge.

**cardNumber : Integer**

The card number that is present in the back of the card

**expiryDate : Integer**

The date that the card expires

**CVV : Integer**

The small 3 digit code that is on the back of a credit/debit card

**paymentMethod : String**

The way that the user is going to be paying for their items. This option could be online in forms such as paypal or card or it can be in person in forms such as cash.

**cashOnline : Boolean**

If the user is buying tickets online but wants to pay in cash they would have to check this button so that the tickets are reserved for them.

**discountType : String**

In the case that the user has a discount code or something of the sort that would be entered and then the appropriate discount is given to the user

**giftCardBalance : Double**

In the case that the user has a giftcard with some leftover balance the amount is saved in the payment information area since it is a way to pay for their items

**Operations:**

**processPayment()**

After taking all the information from the user the payment is then processed with an official banking system where we send the transaction ID to the bank to try and get our money.

**authorizePayment()**

In the case that the transaction is good the payment will be authorized and a message will be sent to the user that the purchase was successful

**applyDiscount()**

In the case that the user has a discount code that they want to use with payment the code is going to be processed and if it is valid a message showing it was valid and the discount amount will be shown. In the case that it isn't valid an error message will be shown to the user.

**refund()**

In the case that a user for some reason wants to cancel a ticket and get a refund then we have their saved banking information where we can easily send them their money back.

**checkGiftCardBalance()**

In the case that the user wants to either use their gift card or just check the balance we can do that for them and either apply the gift card or just show them the value.

**Class:**

**Showtime**

For a specific showtime there are going to be different variables that are available. As a result by creating a class for showtime you can easily organize all the information about a specific movie screening into one class.

**Attributes:**

**specificID : Integer**

The ID for showtime is the specific ID for the showing at a certain time for a certain movie. Again this ID is here just to help and organize information.

**time : Integer**

The time that the movie is going to be showing at

**seatsLeft : Integer**

How many seats are left for a specific movie showing

**screen : String**

The screen number that the movie is going to be showing at.

**ticketPrice : Double**

The ticket price for one ticket at this specific movie showtime

**Operations:**

**availableSeats()**

Showing seats that are available for a specific movie showcase.

**sortByMovieTypes()**

Within a movie theater there could be multiple showings of the same movie that each have different filters for differently abled people. This is going to allow you to sort movies within a certain showtime

**ifAdultMovie()**

In the case that the movie being shown is a movie rated R or above there is going to be a warning show about the fact

**nearbyShowtimes()**

Also display showtimes for the same movie at the same time at relatively close theaters.

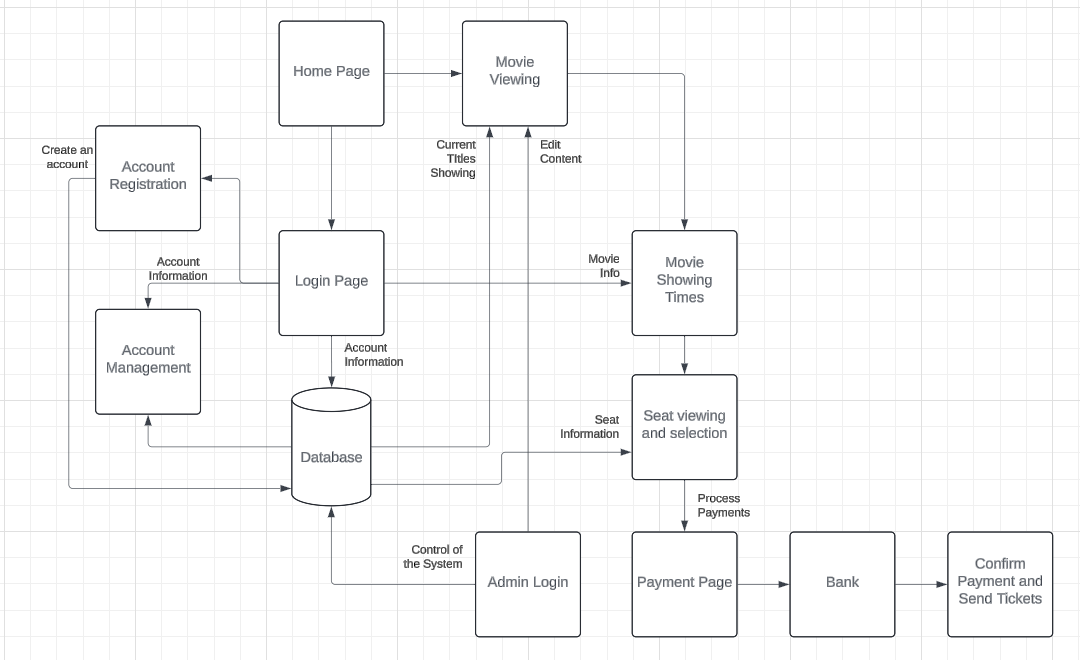
****

Figure 5: Displays the complete process of purchasing and accessing the tickets.

**Account Registration**

Customers can create accounts that will hold information about past movies and payment information. Will also be used to track available promotions.

**Account Management**

Users and customers can access or edit account information. Customers can change or reset passwords.

**Movie Viewing**

Displays all current movie titles. Customers can select a specific movie to view information about. Selecting a specific movie will display showing times.

**Seat Viewing and Selection**

Lets customers view available seats and select certain seats. A preview of selected seats will display as the user updates selection. Allows customer to add seats to shopping cart.

**Payment Page**

Customers can check out items in their shopping cart. Connects to the payment gateway and allows for checkout.

**Home Page**

Landing page for the movie ticketing website. Connects to Login Page and Movie Viewing.

**Login Page**

When the user searches up the movie ticketing site, there will be a login section that they will be able to access if they are members of the online system. They will be asked to input their username and password so they can access their information.

**Database**

When the user logins using their username and password, the database is where all their personal information regarding their online account will be stored. This includes their username, password, past items purchased including movies and snacks, returns, number, and email.

**Admin Login**

This login will differ from the user login because this one will be solely for administrators of the online ticketing system. These personnel will be capable of changing the content of the website in any type of way.

**Seat Viewing and Selection**

The users will be able to see all available, unavailable, and handicapped seats. They will be able to select all types of seats, but if they are unavailable the system will not allow them to continue to purchase them.

**Confirm Payment and Send Tickets**

Once the user finishes purchasing the tickets, a confirmation message will be displayed on the screen and the tickets will be sent to their email. In the email, their seats, their time of purchase, the date, the movie, and an option to cancel will be displayed.

## 3.5 Non-Functional Requirements

### 3.5.1 Performance

* Under normal circumstances the website should load in under 2 seconds
* Should be able to handle at least 20,000 concurrent users without and damage to performance of the website
* Ticket payments should be confirmed within 5 seconds
* In the case that the website crashes a crashes a crash report should be sent back
* Sending email within a minute of ticket purchase with ticket details

### 3.5.2 Reliability

* Website should stay online without problems and can have downtime for an hour every month
* In the case of a failure for some reason website should be down for no more than 10 minutes
* Save data on multiple server ensuring that in the case of a server failure multiple reports of ticket information are still available
* Data backups every half an hour

### 3.5.3 Availability

* Website should be up for at least 99.9 percent of the time and make it so that it is accessible at anytime
* Have checks in place to have backups plans in the case of unexpected maintenance or server failure

### 3.5.4 Security

* All user data is encrypted and protected.
* System itself is also protected with secure login and protection against hackers.
* Payment options are secure.

### 3.5.5 Maintainability

* Built in a way that there is space for updates and bug fixes without much downtime
* Allow for quick changes to things such as movie times or where the movie is playing

### 3.5.6 Portability

* Have both mobile and desktop capabilities alongside a designated application for ios and android.
* Works on different operating systems such as Windows Linux or even macOS issue free
* Works on different browsers such as chrome or firefox issue free.

## 3.6 Inverse Requirements

* Website should not allow for weak passwords to be used to log in to the website
* Website should not treat all users equally. Admins should be given a different view compared to a regular customer.
* Should not have outdated information (Old movies that arent showing, things the theater doesn’t offer anymore)
* Should not allow for a user to easily enter their payment information. Make sure that the card is actually theirs and don’t allow for fraudulent activity
* Should not show any incorrect information such as wrong ticket prices or wrong movie times
* Should not be slow or crash frequently
* Should not have no customer service. Should either have a real life representative always attending to the website or at least a number that customers could call if they are running into issues regarding the website.
* Should not be hard to navigate. Easily accessible for anyone who wishes to see a movie
* Should not only be available on one device. Website should be accessible through whatever a customer wishes to purchase a ticket off of.

## 3.7 Design Constraints

* The ticketing system will primarily support windows, web browsers, and mobile applications.
* The ticketing system will support all major payment options.
* The ticketing system will have the option to be accessible.

## 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.*

## 5.1 Test Plan

Claudia Gonzalez: The person responsible for this role will be developing and maintaining the requirements, capabilities, and functions of the system specification.

JJ Ammar & Deepanshu Joshi: This team will require 2 people as they are responsible for developing and verifying the software. However, the software is not compiled or completed during this procedure.

Jianna Gapuz: This role is responsible for creating statistical tests to assess the software after its development. Reliability growth models may be used to regulate when reliability is acceptable.

**The Testing Process**

The testing process is where we will be testing every single function thoroughly to see whether they work successfully. Exceptions are programmed to display when invalid information is entered. All functions will be tested, but they include payment processing, selecting seats, receiving a confirmation email, and more. The goal of the testing process is to have a 97% success rate to ensure the user has a successful experience on the online ticketing system.

**Requirements traceability:** Every single object and class will be tested to ensure all requirements are being met. After they are run, results will be collected to see how anything can be improved. For example, if the payment process is not working or taking too long, we would record those results and see how we can make the process more effective to satisfy the users.

**Tested Items:** The team will be testing the stages of the development such as the account registration, account management, loginpage, home page, admin login, payment processing page, and making sure all forms of display are correct.

**Testing Schedule:** Once development is done by the development team, testing will begin and results will be recorded.

**Test Recording Procedures:** The results will be recorded including effectiveness, time, error rate, and user satisfaction but most importantly making sure it is meeting all expectations at a reasonable time without error.

**Hardware and Software Requirements:**

1. Server Requirements: High-performance server for hosting the application, ensuring smooth testing of server-side operations
2. Testing Devices: Multiple devices (PCs, laptops, tablets, smartphones) to ensure compatibility across different screen sizes and operating systems.
3. Network Equipment: Reliable networking equipment and stable internet connection for testing functionality dependent on network connectivity, like login and payment processing. Testing with flagged IP addresses is also a possibility using multiple routers.
4. Browsers: Browser compatibility and testing making sure that it works on chrome, safari, firefox etc..
5. IDEs for developing and testing code.

**Constraints:** Some constraints include time, not enough people to help test, budget, and technological limitations.

# 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