Silver Screens Cinema

Software Requirements Specification

Version 1

February 15, 2024

Group #13

Janae Farinas, Jorden Kruzner, Rekik Degefa

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

Revision History

Date	Description	Author(s)	Comments
2/15/24	Version 1		<First Revision>
N/A		Janae Farinas	
N/A		Jorden Kruzner	
N/A		Rekik Degefa	

Document Approval

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

Signature	Printed Name	Title	Date
	<Your Name>		
	Dr. Gus Hanna	Instructor, CS 250	

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this SRS document is to hold the information discussed between the client and developers in order to maintain and create the client's wishes for the Ticketing System website project. Its main purpose is to serve customers digitally so that consumers may purchase movie tickets via mobile devices and increase sales. This SRS also serves to help the project developers push ideas into one place for efficiency and communication for it contains the layout of many functions and use cases.

1.2 Scope

The Silver Screens Cinema is an Online Ticketing service that allows customers to search for and purchase tickets for films. The SSC allows customers to watch trailers, read reviews, and film synopsis before purchasing a ticket. The SSC allows customers to accumulate points for each purchase and pre-order goods like food and drinks before the beginning of their film. Through the SSC, customers can browse through a catalog of current playing films in different ways by searching for any specific genre or movie theater location. The SSC is accessible and targeted to all users of different backgrounds as a helpful tool and a way of enhancing the movie ticket-buying experience.

1.3 Definitions, Acronyms, and Abbreviations

Word:	Meaning:
OTTS	Acronym for Online Theatre Ticketing System
SSC	Silver Screens Cinema
Acc	Acronym for "Account"
TBD	Acronym for "To Be Done"

1.4 References

1. E-Store Project Software Requirements Specification Version <4.0>, 15 April 2007, Marvel Electronics and Home Entertainment.

2. IEEE Recommend Practice for Software Requirements Specifications, 20 October 1998, Software Engineering Standards Committee.
3. Software Requirements Specification document with example, 8 May 2023, KrazyTech

1.5 Overview

From here on out, this form includes the main logic of how the OTTS will be broken down. It will include ideas built off of the client's specifications for the product. The SRS is designed in a way the development of the software is currently broken up into 4 sections overviewing different aspects of the project, including the introduction, General Description, specific requirements, and sequence diagrams. These sections work to include in-depth descriptions of OTTS functions, the different interfaces used, any specific, inverse, and non-functional requirements, and their performance ability.

2. General Description

2.1 Product Perspective

The SSC OTTS is built with the intention of being utilized with other products namely the Regal and AMC theaters, and be displayed mainly on mobile devices. The OTTS also utilizes and displays critic and general public reviews, namely from the Rotten Tomatoes site.

2.2 Product Functions

The OTTS website will hold many functions in order for it to run efficiently for both the user and the client. These functions include an A-Z database full of movies that is searchable by the user, movie ratings, location, and movie name. Each movie provides a brief description of the movie, the trailer, and reviews. Reviews are also able to be left by the user through the website. There will be tabs at the top of the websites to focus on certain categories such as new releases, movie ratings(PG, R), genre(comedy, thriller), etc. Other functions include making an account that allows users to become part of a rewards program and receive notifications about deals or coupons. Another important function is the shopping cart feature which keeps track of the tickets that the user wishes to purchase, and when they decide to check out, they are given a limited amount of time to checkout in which their seats remain reserved. Tickets range in price depending on whether it is for a child, senior, or adult, or whether they get a student discount.

2.3 User Characteristics

Since the SSC OTTS is a website for purchasing online tickets, the website is user-friendly and interactive, allowing customers to reserve and buy theater tickets, browse and search the movie catalog, and the ability to leave reviews.

2.4 General Constraints

During the time of SSC OTTS development, the team may face some general constraints regarding budget and time cost, performance abilities, and other external constraints. Maintaining a website takes a large amount of both time and money to keep up and running and a certain amount of work needs to be done within the allotted time that is negotiated with the client. There are also other factors to take into account when developing the SSC OTTS such as legal matters including copyright infringement.

2.5 Assumptions and Dependencies

The SSC OTTS assumes that the user has access to a form of electronic payment or credit/debit card. It assumes that the user has access to a stable internet connection when accessing the OTTS. It depends on the user to keep their device updated to the latest version. The OTTS assumes that the user is able to navigate the site has basic knowledge of online ticketing and payments and can easily interact with the site functions. The OTTS assumes that the user is familiar with the rules and regulations of the movie theater. The OTTS is also dependent on the user to have a compatible operating system and/or browser and will have to adapt and change accordingly if they are not compatible.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The OTTS should implement a Graphical User Interface and Command Line Interface for purposes of having a layout that is easy to understand, click on visuals to lead to new pages, and to use the search function to manipulate the information displayed on the screen. A menu-driven interface can also be useful for providing different categories of dropdown lists that provide different genres, locations, movie ratings, etc.

3.1.2 Hardware Interfaces

Necessary hardware interfaces include a working mobile device such as a computer, tablet, or phone, and any connected keyboards or a mouse through USB or wireless connection.

3.1.3 Software Interfaces

The OTTS will be implementing Javascript. It should work with various operating systems including Windows, macOS, Android, and also different variations of these systems such as Windows 10. Simple Mail Transfer Protocol (SMTP) email is also a necessary interface so that customers may have the ability to email the admin or report issues if they have any questions or concerns. It requires the use of digital payment software such as Apple Pay, Google Pay, PayPal, and Venmo. It also needs to access RottenTomatoes in order to obtain updated reviews on movies.

3.1.4 Communications Interfaces

The SSC OTTS should support all common web browsers and require a wireless internet connection.

3.2 Functional Requirements

3.2.1: Allow Searching for Films Through Search Bar

3.2.1.1 The OTTS should allow users to search for a specific film

3.2.1.2 The OTTS should allow users to input text in a text box to search for a media title

3.2.1.3 The OTTS searches through the catalog of movies titles and selects which ones include the word(s) typed into the search bar

3.2.1.4 The OTTS should showcase the film title and photo to the user with the different locations and movie times that this movie is available

3.2.1.5 If no movie titles match what was searched, display "No search results found"

3.2.2: Searching For Media

3.2.2.1 The OTTS should have a tab for "genres" on the homepage

3.2.2.2 The OTTS should display a list of film genres

3.2.2.3 The OTTS should allow users to select the total or less than total amount of genres by clicking a check box next to the genre

3.2.2.4 The OTTS should list films that include user-selected genres

3.2.3: Display Film Trailer, Synopsis and Reviews

3.2.3.1 The OTTS should display a film trailer and 2 official promotional photos in a slideshow manner to the user

3.2.3.2 The user selects their desired movie.

3.2.3.3 The OTTS should have a trailer, synopsis, and reviews that correspond to that movie.

3.2.3.4 The OTTS should display a synopsis of the film under the slideshow to the user and provide verified user reviews to the right of the slideshow and synopsis.

3.2.3.5 An error message may be necessary if the trailer doesn't load.

3.2.4: Allow Searching based on Location

3.2.4.1 The OTTS should allow users to search for movies in a specific location whether based on city, zip code, or specific theater

3.2.4.2 The OTTS should allow users to input a certain city, zip code, full address of a theater, or the specific name of that theater into the search bar

3.2.4.3 The OTTS searches through the different theater locations listed in the database for what cities, zip codes, and names of theaters match what was searched

3.2.4.4 The OTTS should showcase the theaters that are within that city or zip code, or that match the name typed into the search bar

3.2.4.5 If no locations match what was searched, display "No search results found"

3.2.5: Provide Customer Support

3.2.5.1 The OTTS should display a footer of customer support contact through email and phone number

3.2.5.2 The OTTS should have a tab on the homepage for customer support contact

3.2.5.3 The OTTS should provide the user with a 24/7 AI chatbot tool at the bottom of the screen.

3.2.5.4 The OTTS should be able to process the user's questions and provide feedback whether it is an answer or potential helpful links displayed in the AI chat box.

3.2.5.5 If the bot is unable to generate an answer to their question, display a message asking the user to try again with a different or re-worded question.

3.2.6: Maintain and Create Custom Profile

3.2.6.1 The OTTS should allow a user to create an account or continue as a guest

3.2.6.2 The OTTS should allow a user to select a profile photo from a provided set of images and input private information such as name, address, email, and phone number

3.2.6.3 The account is saved in the database and becomes part of programs such as receiving personalized suggestions, receiving points, and receiving emails regarding deals and coupons

3.2.6.4 The user's account is displayed with their selected corresponding profile picture and they receive an email confirming their subscription

3.2.6.5 If an error occurs while the user is creating an account, the page will display that an error has occurred and they will be asked to refresh the page. Let the user know that information may be lost.

3.2.7: Sign In to Your Account

3.2.7.1 The OTTS should verify that members sign in with their correct ID and password with dual authentication.

3.2.7.2 The user inputs their username and password and then is sent a code which they have to use to verify they are signing in.

3.2.7.3 The OTTS checks whether or not it matches the information in their account and checks whether the code sent matches the code entered.

3.2.7.4 The user is redirected to the home page if all the information is correct.

3.2.7.5 If the user inputs incorrect information a message displays "Username or password is incorrect" and for the code, it asks if the user wants to send another code.

3.2.8: Provide an Online Shopping Cart

3.2.8.1 The OTTS should allow users to put tickets into a shopping cart which displays the ticket details and how much money the user will have to pay for the tickets.

3.2.8.2 The user selects what movie, seats, and ticket type they want, and it is added to the shopping cart.

3.2.8.3 The shopping cart is updated with new information and the amount owed increases.

3.2.8.4 Ticket details are displayed in the shopping cart.

3.2.8.5 If someone reserves the seats that are in the user's shopping cart, a message will be displayed that those tickets are no longer available, and they will be redirected to the home page.

3.2.9: Purchase Film Tickets and Goods

3.2.9.1 The OTTS should allow a user to purchase a ticket for the film of their choice and pre-order goods like food, snacks, and beverages.

3.2.9.2 The OTTS should showcase the ticket price after tax calculations.

3.2.9.3 The OTTS should allow payment options of credit/debit, Paypal, Apple Pay, and Samsung Pay.

3.2.9.4 The OTTS should send out an email confirmation with a ticket QR code attached.

3.2.9.5 The OTTS will provide the user with a digital receipt as proof of purchase in the case that the QR code does not work.

3.2.10: Ticket Cancellations and Refunds

3.2.10.1 The OTTS should allow users to apply for a refund.

3.2.10.2 The OTTS processes refunds within 2-3 business days.

3.2.10.3 The OTTS will contact the theater and cancel the seating reservation.

3.2.10.4 The user will get an email confirmation after the system has processed and canceled their ticket.

3.2.10.5 If there are problems with fulfilling the user's refund request, the OTTS will provide a coupon code as compensation.

3.2.11: Handling Site Traffic Including Bots

3.2.11.1 The OTTS should allow for up to 1000 users to enter and use the site at one time, and actively work to ignore bot users.

3.2.11.2 The OTTS inputs and monitors the site traffic continuously.

3.2.11.3 The OTTS should inform the admin of any site crashes and overloads.

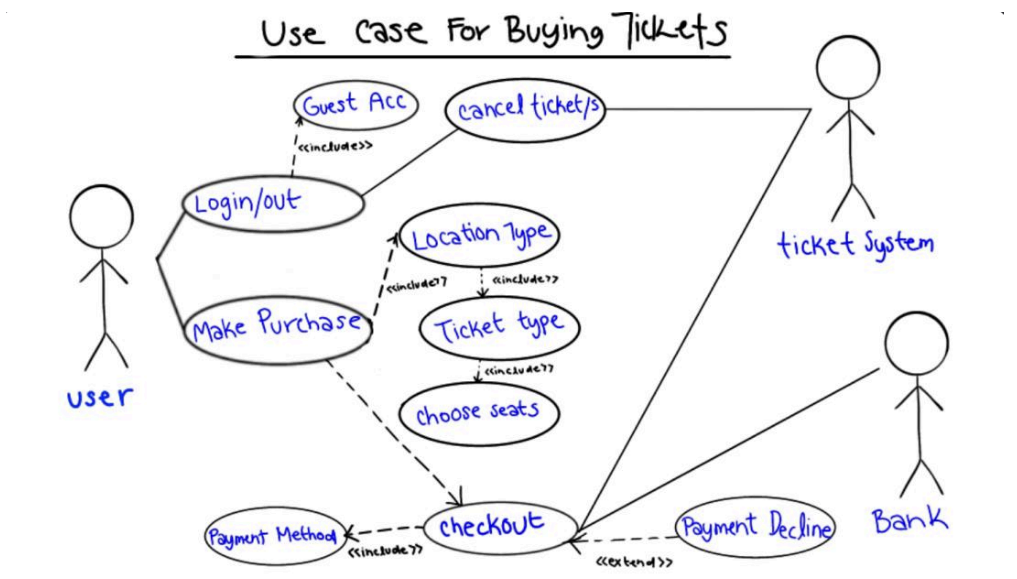
3.2.11.4 When the OTTS crashes it will display a screen letting the user know that the site is currently down, be redirected, and try again later.

3.2.11.5 The admin of the OTTS will resolve the issues, get the OTTS back up again within a couple of hours, and block any accounts that seem to be run by bots.

3.3 Use Cases

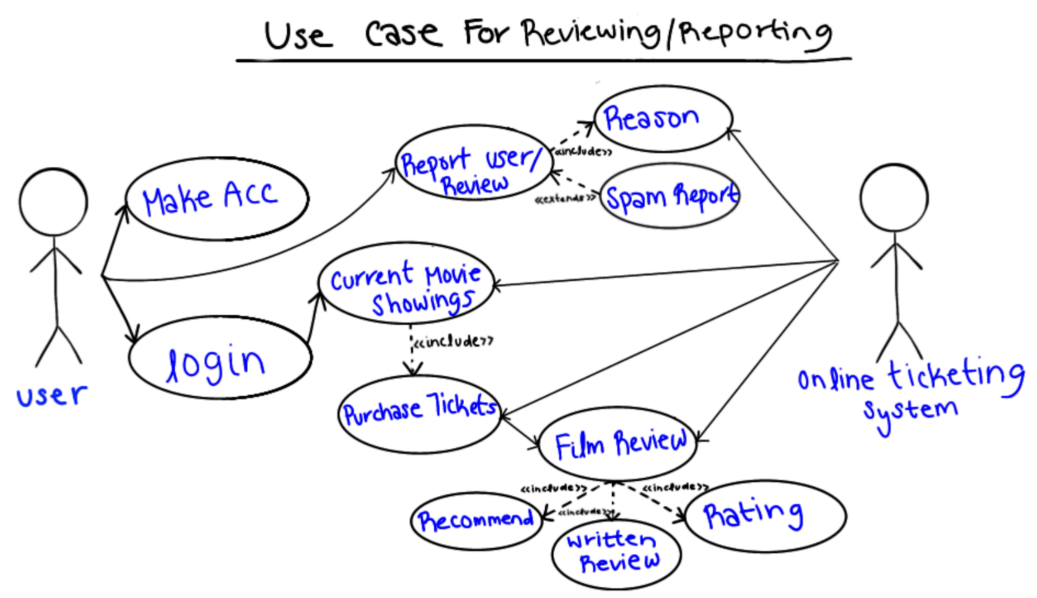
3.3.1 Use Case #1: Purchasing a Ticket

Step #:	Action:
1	Customer logs into an SSC account either through an email-linked account or a guest login.
2	Customer finds a movie and goes to purchase a ticket.
3	Customer inputs information of the wanted location, and ticket types which include child, student, adult, and senior.
4	After the user inputs the latter information, the OTTS takes them to the checkout screen that displays the ticket information, and final calculation of the total ticket(s) price.
5	When the user clicks the “Buy” button after confirmation, the payment is then processed by their bank.
Step #:	Branching Action:
2.a	If the user is not logged into an account and tries to purchase a ticket, the screen displays a login screen.
5.a	If there is an issue with completing the customer’s specified method of payment, a “Payment Declined” error method is displayed to the user.



3.3.2 Use Case #2: Reviewing and Reporting Customer Reviews

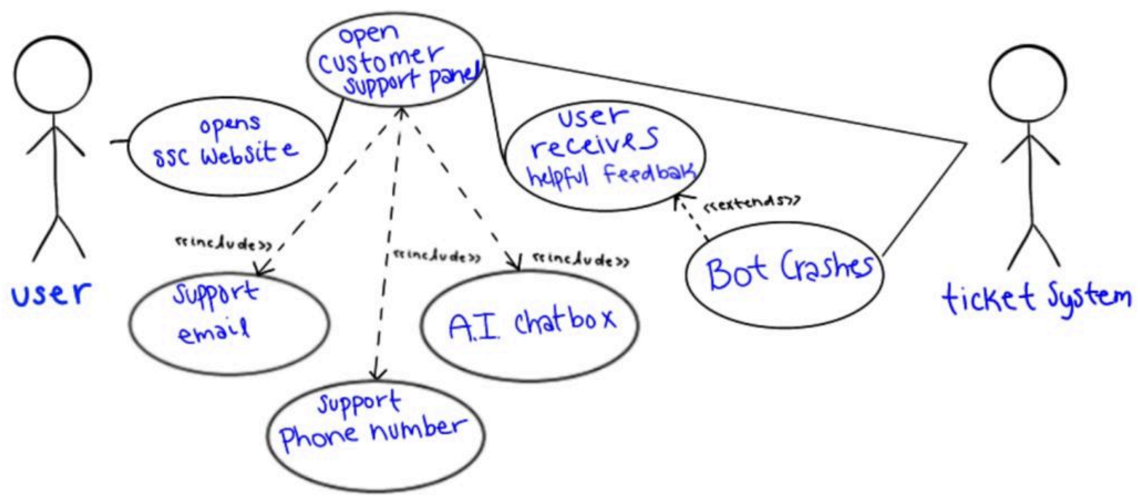
Step #:	Action:
1	User makes an SSC account and logs in.
2	The user browses the "Currently showing" movies and decides to purchase a ticket
3	After watching the film, the user decides to leave a review from their SSC account recounting their experience with, recommend or not recommend it, and leave a rating out of 5 stars.
Step #:	Branch Action:
1.a	User decides to file a report against another user/user review
1.a.1	A user spam reports and the spammer in question gets a notice of not being able to report for another 3 hours.



3.3.3 Use Case #3: Interacting and Handling with Customer Support

Step #:	Action:
1	User enters the SSC website.
2	User opens the customer support panel or sees their information at the foot of the webpage.
3	User chooses to contact the A.I. chatbox and receives helpful information regarding their issue.
Step #:	Branching Action:
3.a	A.I. chatbox crashes.
3.b	User does not receive helpful information.

Use Case For Customer Support



3.4 Classes / Objects

3.4.1 Movie

3.4.1.1 The movie class should include the movie title, movie description, and movie trailer alongside 2 official images from the film.

3.4.1.2 The movie class should be able to display the movie information visually to the user. When a user searches and clicks on a movie to view, section 3.2.3 should display all information in a visually appealing manner.

3.4.2 MovieTicket

3.4.2.1 The MovieTicket class should include a Unique QR code

3.4.2.2 The movie ticket should include the customer's theater location, room number, and seat number

3.4.2.3 The movie ticket should have a 9 digit number to uniquely identify it for different processes such as refunds.

3.4.2.4 The MovieTicket class should include the designated and discounted prices for different customers, i.e. Child Ticket, Adult Ticket, Senior Citizen Ticket, and Student Discount.

3.4.3 MovieTheater

3.4.3.1 The MovieTheater class should include city, state, zip code, street address, and the theater name

3.4.3.2 The functions of this class are to be able to display movie theaters in a desired location. When a certain city, state, zip code, address, or theater name is searched, the corresponding theaters should be displayed. This function is referenced in section 3.2.4.

3.4.4 UserReview

3.4.4.1 The Review Object should include a user-generated review and a rating out of 5 stars.

3.4.4.2 The Review Object should have a report flag

3.4.4.3 A user should be able to post a review and be able to report other reviews.

3.4.5 Account

3.4.5.1 The Account Class stores the profile information of the customer, their email address, payment information, purchase history, points history, etc.

3.4.6 CustomerAccount

3.4.5.1 The Customer object should be able to keep the user's information private and readily accessible for the user to access.

3.4.5.2 The Customer object includes any functions that allow the user to change and adapt any parts of their profile.

3.4.5.3 The Customer object saves the login information of the user.

3.4.7 GuestAccount

3.4.6.1 The GuestAccount allows the user to make a purchase without creating an official account and profile with the OTTS

3.4.6.2. The GuestAccount does not keep track of the user's point history

3.5 Non-Functional Requirements

3.5.1 Performance

When having a sufficient internet connection, our website quickly loads information, especially when using the search function and when being taken to a new page. It should remain up-to-date with current movies and their reviews. It should also stay updated regarding the availability of seats to ensure that when other people have purchased tickets, it shows that those seats are reserved. It must also be quick to provide a confirmation email with a working QR code that can be added to a digital wallet.

3.5.2 Reliability

Our ticketing system is fully functional with many people accessing it. It is able to hold approximately 1000 people. It must perform without failure in 95% of use cases and if recovering from handling an error, it should be able to continue without losing data such as whether the user had put something in their shopping cart, entered any personal information, or made a purchase.

3.5.3 Availability

Although a mobile website is used for its accessibility, the servers will go offline once a week for approximately 180 minutes in order for maintenance, updates, or backups to be performed. Ideally, this weekly scheduled maintenance will work to prevent any system failures during the week. Our website must be able to handle user traffic or display an error message to the user.

3.5.4 Security

Our OTTS uses an HTTPS website that utilizes the encryption protocol, TLS. This tool encrypts and decrypts information, safely securing and storing your information. This protocol will ensure card and other personal data safety, including emails, phone numbers, and addresses.

3.5.5 Maintainability

The goal of the SSC website is to ensure the efficiency and functionality of our software to the user. If there is an error that causes the system to fail, it should be fixed approximately within a 30-minute time frame. The SSC OTTS software should be neatly organized to maximize efficiency and allow the developer team to maintain and fix any irregularities. The software should also be efficient without using any redundant code.

3.5.6 Portability

Our OTTS should be accessible and perform consistently across various electronic devices such as laptops, mobile phones, and tablets. It should also be able to run on different operating systems such as Windows, macOS, Android, and Linux, as well as different models of these operating systems. The OTTS should behave the same on all the main web browsers such as Chrome, Safari, Firefox, and Opera GX.

3.6 Inverse Requirements

3.6.1 This OTTS can't be used to buy or rent movies for personal use, it is only to purchase movie tickets to be used in a theater.

3.6.2 This OTTS cannot be used to purchase any alcoholic beverages, and can only be used to buy non-alcoholic beverages like sodas and water.

3.6.3 This OTTS stops providing refunds for cancellation 24 hours before the film viewing officially starts.

3.6.4 This OTTS does not provide tickets for any live streams or television broadcasts, only films. If tickets placed in the shopping cart after an hour are not purchased, the OTTS will automatically remove them from the user's cart.

3.6.5 The OTTS does not provide tickets for any live productions, concerts, or performances.

3.6.6. The OTTS does not accept any forms of physical payments like cash or checks.

3.7 Design Constraints

The SSC team will have some constraints due to the limitation of time, budget, client policies, and varying hardware that may interrupt the designing and testing process during product creation. We will also need to take into consideration user feedback and discuss any changes that may need to happen to satisfy customer and/or client needs.

Within our software development there also exist some constraints within our code that will need extra care; such as unclosed loops or redundant code within our testing and debugging phases.

3.8 Logical Database Requirements *(LEFT BLANK PER PROFESSOR)*

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 *(LEFT BLANK PER PROFESSOR)*

Catchall section for any additional requirements.

4. Analysis Models

4.1 Software System Description

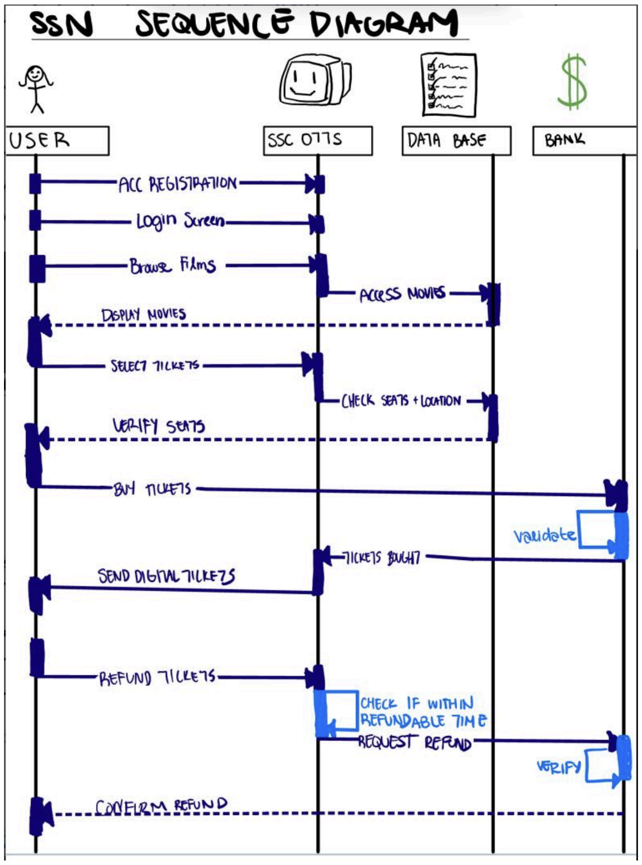
The SSN OTTS is a website designed for customers to purchase a movie ticket online with ease. Our team will be mainly working on Microsoft windows devices but also test on other systems like macOS, IOS, and Android.

Users should be able to open our homepage, because it is compatible with most browsers, and browse or search for movie(s) of their interest from our database. Once a user decides to purchase a ticket(s) for a movie of their choice, they can then input their zip code or a zip code from a specific theater in collaboration with SSN zip code and browse the tickets available through our system. Once they choose their ticket location, they are met with several ticket customization steps that should flow from ticket type (child, adult, senior, student) to then selecting seats based on the availability by displaying a map of the theater room to the user and a key to the side of the different symbols and colors (seats, screen, aisles, available, not available). Once choice of payment is confirmed, we will carry out transactions through the Stripe software which will securely encrypt user information.

Notable features of our software include customer support from our support team in real time with the help of the Zendesk software or the option of utilizing our website's own AI chatbot built specifically for handling frequent Q&A and is supported by over 10 languages.

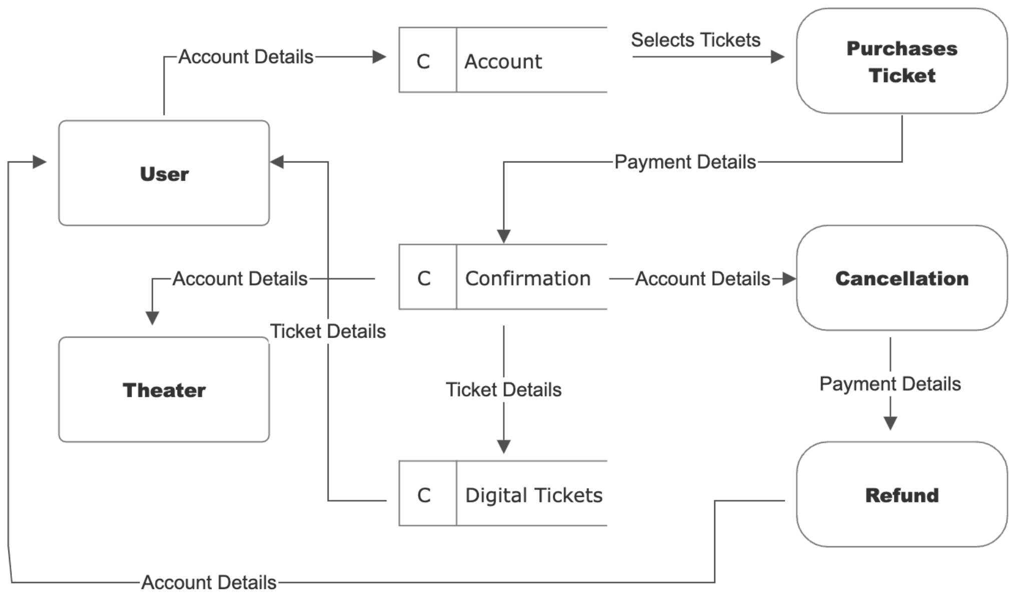
4.2 Sequence Diagrams

The diagram below includes a sequence of events and interactions between software that may occur if a user wishes to buy a ticket for a movie and chooses to refund their ticket within the allotted SSN OTTS refund period.



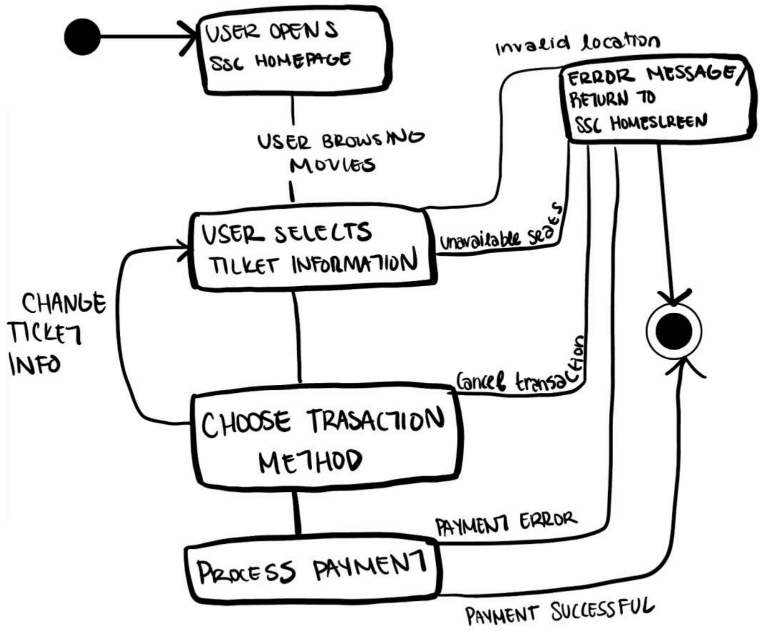
4.3 Data Flow Diagrams (DFD)

The diagram below depicts the overall flow of the information in our system when a user enters our OTTS and purchases tickets. It shows where the user data and information goes and how it is stored and later used.

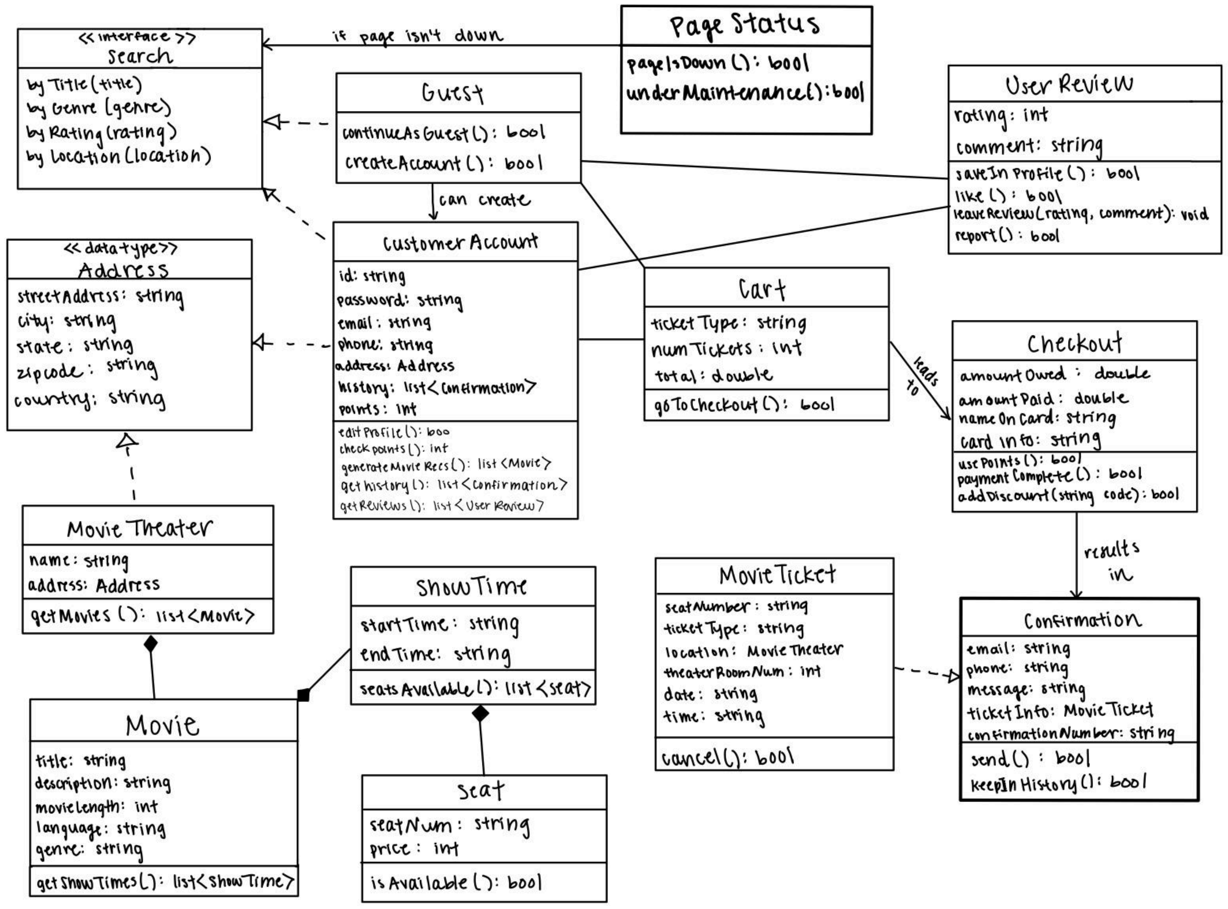


4.4 State-Transition Diagrams (STD):

The diagram below acts as a visualization of the different interactions and states that may occur whether or not a user successfully purchases a ticket through the SSN OTTS.



4.5 UML Class Diagram/Description



The Search interface provides the outline for methods to search by movie title, genre, rating, and location. This interface is used and implemented by the user, whether they are a guest or have an account.

Guests have the option to continue as a guest, or become a member by creating an account.

The CustomerAccount class contains the user's ID, password, personal information, and it keeps track of their past purchases and points they might have gained. They can edit their profile, check the amount of points they have, look at movie recommendations generated based on the movies they have previously watched, and they can look at their past purchases. Another part of their profile is they can see reviews they have left or reacted to.

The UserReview class can be accessed by a member of a guest who may choose to leave a movie review which consists of a rating out of 5 and their commentary. They can also simply react to a review by liking it, or they can report another user or their review.

The Address class is a data type used by CustomerAccount and MovieTheater so that they don't have to store as many attributes individually. It consists of the street address, city, state, zip code, and country that applies to either the individual or the movie theater location.

The MovieTheater class contains the address of the theater and the name of the theater. This class allows the user to access a list of the movies that are shown at a certain theater. Therefore, MovieTheater acts a bit as a container for Movies, since it generates a list of them.

The Movie class provides information for a specific movie, such as the title, description, genre, language, and duration of the movie. For each movie, a list of showtimes can be generated, so the Movie class is a container for the ShowTime class.

The ShowTime class contains the start and end time for a certain showing of a movie, and for a given show time you can access a list of how many seats are available. This class is another container, this time for the Seat class.

The Seat class contains the seat number and the price of that seat(may range depending on the type of theater or chair such as reclining). The operation isAvailable() checks whether that seat has been taken already or not.

The Cart class represents the shopping cart feature where the user's ticket(s) are held until they are ready to checkout. This contains the number of tickets in the cart, as well as the ticket types(child, adult, senior), as well as the total price. The user must pass through the cart feature to get to checkout.

The Checkout class contains the amount owed by the user and the amount paid(will show that there is still more owed if the user uses points but it doesn't cover the entire cost), and it is where the user enters their card information. The operations in this class give the user the option to use points or a discount code which they are only eligible for if they have an account. The operation paymentComplete() checks whether the payment went through successfully, if so, it initiates the confirmation class.

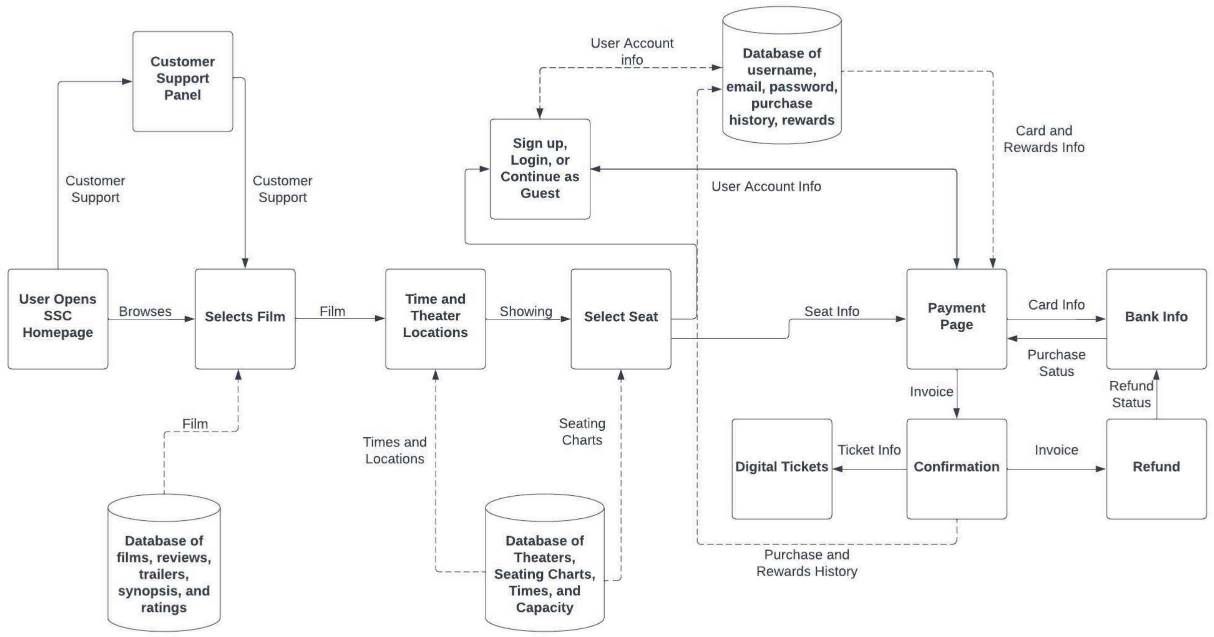
The Confirmation class contains the message that is sent to the user once their purchase has been completed. The user inputs an email and/or phone number(if the user has an account it will already be in the system), and the confirmation is sent to whichever method they chose. The confirmation email/text message also contains a unique confirmation number which can be used for cancellations and refunds, and lastly, it contains the digital movie ticket. If the user has an account, the confirmation of their purchase is kept in their history.

The MovieTicket class contains the user's seat number, ticket type, movie theater, movie name, the movie theater room number, the date, and the time of the movie. You can cancel your ticket through this class, which is implemented by the Confirmation class.

The PageStatus class keeps track of the status of the webpage. If there is a problem with the page, it will update pageIsDown(), display a message to the user on the screen, and the error will be dealt with. If the page is down due to maintenance, a message will be displayed on the screen.

4.6 Software Architecture Diagram (SWA)

4.6.1 Software Architecture Diagram



4.6.2 Software Architecture Diagram Description

The Software Architecture Diagram (SWA) starts off with the component "User Opens SSC Homepage" which has two connectors that lead to different components. If the user has questions and needs to access customer support, the customer support connector leads to the "Customer Support Panel" component that gives help and then leads/supports the user to the "Selects Film" component. If the user has no concerns or concerns once entering the SSC homepage, the next step would be choosing a film by browsing the site and directly accessing the "Selects Film" component. Through the "Select Film" component, access to a full database of the different films, reviews, and trailer options is given and a selected film is presented to the user. Once a desired film has been selected, the next component is the "Time and Theater Locations" which is connected to a database that includes all of the different theaters and showtimes available returned to the user. The selected, user-showing connector includes the time and location of the film chosen and points to the "Select Seat" component that is connected to the previous database that also includes the seating charts of the theater options. The seating chart information is given from the database to the "Select Seat" component then the chosen seat info is connected to the "Payment Page" component.

After the "Select Seat" component, the user is directed to either sign up and create an account for the OTTS, log in to their pre-existing account, or continue as a guest through the

“Sign up, Login, or Continue as Guest” component. The user account info is stored and accessed through a database that includes all usernames, passwords, email addresses, account purchase history, and rewards. Then the user account info connector connects the sign-up component to the payment page and links the account with the “Payment Page” component. The “Payment Page” component receives all of the cardholder and rewards information from the previously stated database. The card information connector connects the “Payment Page” to the “Bank Info " component that processes the card information of the user and returns the purchase status to the “Payment Page”. The user then receives a confirmation email through the “Confirmation” component that connects the invoice/receipt of the ticket transaction. The ticket information from the “Confirmation” is transferred to the “Digital Tickets” component which is where the user is able to scan their tickets for entry at the theater. The “Confirmation” component also updates the user’s purchase and rewards history to the database including the user’s account information. In the case of a cancellation and a refund is required, the invoice goes to the “Refund” component and then the refund status connector updates the status of the refund to the user’s “Bank Info” component.

4.7 Development Plans and timelines (Due Dates)

All TBD in 6 days (28 Feb. 2024), 1 day before submittal in order to go over once then submit/push section 4 updates

Janae Farinas(Commit located in “Janae’s Diagrams) - STD, Sequence Diagram, Design Constraints, Overview
git link:https://github.com/janaeF/Team-Design-Git-Uploads/blob/main/Janae's%20Diagrams.pdf

Jorden Kruzner(Commit located in “Jorden’s Random Upload”) - UML Class Diagram, UML Description
git link:
https://github.com/janaeF/Team-Design-Git-Uploads/blob/main/Jorden's%20Random%20upload.pdf

Rekik Degefa(Commit located in “Rekik’s Diagrams) - SWA, SWA Description, DFD
git link:https://github.com/janaeF/Team-Design-Git-Uploads/blob/main/Rekik's%20Diagrams.pdf

FINAL CHANGES DUE 29 Feb. 2024

Test Plans

We will attempt to do a total number of 150 tests that will take approximately 1.5 hours to complete individually. This leaves off with a total test execution of 225 hours. The majority of the testing will be automated when possible to leave us with a quicker and more efficient execution time. We have a test coverage goal of covering 95% of all possible user entries and possibilities leaving us a window of a 5% error that includes any test cases we could have missed and do not have the resources to add.

Test Plan #1

The first test plan focuses on the main functions of the website that every user, whether guest or member, must go through to purchase a ticket. The specific features that this will be testing are the various search functions, checking the seat availability of a movie, adding that seat to the shopping cart, and checking out successfully. The first set of tests consist of checking whether the search function works properly and as stated in the functional requirements section of this document. Since we should be able to search by location, movie title, genre, and movie ratings, we must test both valid and invalid inputs to see whether we obtain the desired output. If a valid movie theater location is entered into the search bar(which can be by zip code, movie theater name, city, or street address), that theater should show up, and the user should be able to click on that link which would lead them a page that allows them to navigate through all the movies offered at that theater. If a movie title is searched, then all the theaters and their showtimes for that movie should appear on the screen. For searching a genre, all the movies that fall under that genre should appear, then once selecting a movie, it will show the theaters and showtimes for that movie. Similar to searching by genre, if searching by movie rating, all the movies with the given rating or greater will be shown, then theaters and showtimes can be seen once selecting a certain movie. For any input into the search bar that is invalid, meaning it doesn't match any of the information in the database, "No search results found" should be displayed on the screen.

After navigating the search function, the user should test checking the availability of showtimes. Only showtimes that still have available seats should be shown. If the user clicks on a displayed showtime, but all the seats are full, then there is an error in the code that needs to be fixed. Once the user clicks on a showtime, they should be asked how many tickets they are interested in purchasing, and whether those are child, adult, or senior. This information should be correctly saved, and once entered, the layout of the theater should be accurately displayed on the screen, with the correct color-coded seats and/or symbols to show which are already reserved, emergency exit, or for handicap. The user should not be allowed to select more seats than the amount of tickets they entered. When they are satisfied with their choice, they should be able to select "Add to cart". This should then add the correct number of tickets, with the correct ticket information and total price into the cart. When testing this feature, it is crucial to make sure that the pricing for different ticket types have been saved and calculated correctly, as well as the movie, seat, and showtime information. If the user tries to add seats that are already reserved, then an error will occur, and that means that the seats were not displayed accurately and the code needs to be fixed. An acceptable error that may happen is when the user goes to checkout, but in the time they took to add seats to their cart, the ones they chose have been reserved, and they should be redirected to the search page and the seating chart must be updated. To test this, two users must try reserving seats at the nearly the same time, to ensure that only the first person gets the ticket, and the second gets redirected.

The final set of tests for this plan is to test the actual purchasing of the ticket, meaning through checkout. Once the user clicks checkout, they should be redirected to another page that asks them to enter payment details, email, and phone number. This mainly applies if the user chose to "continue as guest". If they are a guest, then the information they enter should be verified as valid information (valid card information, email address, and phone number). If it is invalid then a message saying that the information entered is invalid should appear. The user should test both valid and invalid entries to make sure that this feature works. If the user is using an account, then their personal information should already be available, they just have to enter the correct CVC number. Both the correct and an incorrect CVC number should be tried so as to make sure their information is being properly protected. More information about account users is explained in Test Plan #2. The last feature that should be tested is once the user has made their purchase, they should receive a confirmation email with all the correct movie ticket information. They should also get a confirmation from their bank saying they made a purchase.

Test Plan #2

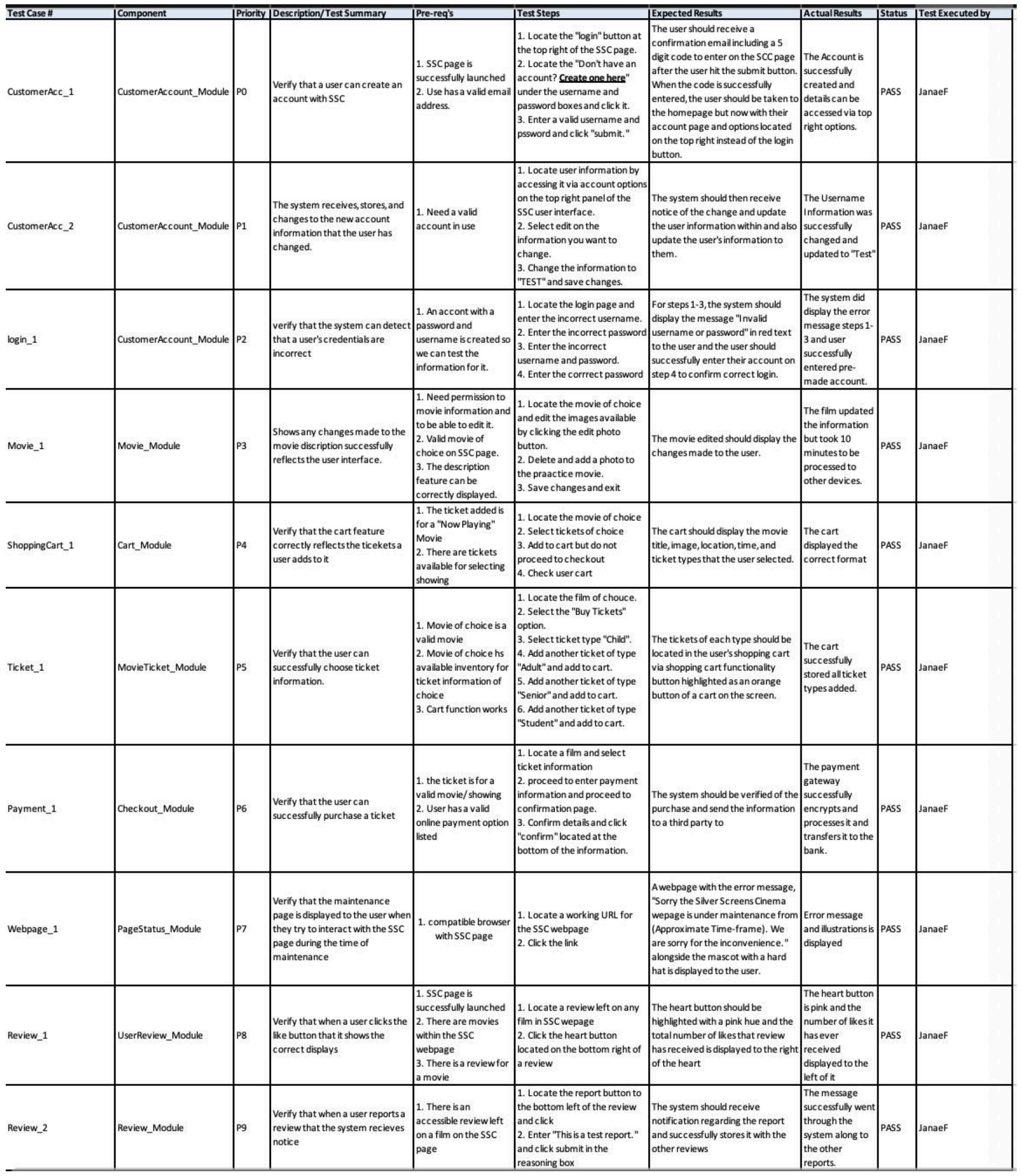
In this test plan, we will be checking out all of the features related to interacting directly with user-entered information in terms of their account and customer service procedures. The features we are testing include setting up and logging in to a user account, user interaction with customer service, and handling user account card/bank information. Testing the user account feature will include testing that passes when correct combinations for a username and password are entered and testing that fails when illegal and incorrect combinations, that do not match up to the user-stored information in the database, get entered. The testing will be specific and account for each user's entered data to be a match per character so that only the correct username and password pass. This also includes testing to see if the username and password themselves get stored correctly upon the initial creation of the account.

Testing the customer service feature will include testing that covers the transport of information that the user entered into the customer service panel. This testing will cover if the correct response to the user question is received and if the user response gets transferred correctly. Testing will include checking if the response matches the question of the right user when the customer panel is faced with multiple users asking different questions. Testing will also include checking to see if the customer questions are received by the panel or not. We will also be testing the entire interaction of the customer service panel to see if the chatbox can support an entire conversation with a user.

Testing the user's account card information will include testing that the user's information is protected from unauthorized access and if the card information is accurate to the bank’s. This includes testing that the user account’s card information is safely stored by verifying that the CVC details on the card match that of the stored information when attempting to purchase a ticket at checkout. We will also test incorrect entries and attempts to purchase so that the test will fail. We are also testing to see if the payment gets processed with the bank. We will also test the entire process of creating an account, interacting with customer service, and then purchasing a ticket as a user. More in-depth test case examples are stated in our Excel sheet.

Test Cases

Link to Repository



Test Case #	Component	Priority	Description/Test Summary	Pre-req's	Test Steps	Expected Results	Actual Results	Status	Test Executed by
CustomerAcc_1	CustomerAccount_Module	P0	Verify that a user can create an account with SSC	1. SSC page is successfully launched 2. Use has a valid email address.	1. Locate the "login" button at the top right of the SSC page. 2. Locate the "Don't have an account? **Create one here**" under the username and password boxes and click it. 3. Enter a valid username and pssword and click "submit."	The user should receive a confirmation email including a 5 digit code to enter on the SCC page after the user hit the submit button. When the code is successfully entered, the user should be taken to the homepage but now with their account page and options located on the top right instead of the login button.	The Account is successfully created and details can be accessed via top right options.	PASS	JanaeF
CustomerAcc_2	CustomerAccount_Module	P1	The system receives, stores, and changes to the new account information that the user has changed.	1. Need a valid account in use	1. Locate user information by accessing it via account options on the top right panel of the SSC user interface. 2. Select edit on the information you want to change. 3. Change the information to "TEST" and save changes.	The system should then receive notice of the change and update the user information within and also update the user's information to them.	The Username Information was successfully changed and updated to "Test"	PASS	JanaeF
login_1	CustomerAccount_Module	P2	verify that the system can detect that a user's credentials are incorrect	1. An accont with a password and username is created so we can test the information for it.	1. Locate the login page and enter the incorrect username. 2. Enter the incorrect password 3. Enter the incorrect username and password. 4. Enter the corrrect password	For steps 1-3, the system should display the message "Invalid username or password" in red text to the user and the user should successfully enter their account on step 4 to confirm correct login.	The system did display the error message steps 1-3 and user successfully entered pre-made account.	PASS	JanaeF
Movie_1	Movie_Module	P3	Shows any changes made to the movie discription successfully reflects the user interface.	1. Need permission to movie information and to be able to edit it. 2. Valid movie of choice on SSC page. 3. The description feature can be correctly displayed.	1. Locate the movie of choice and edit the images available by clicking the edit photo button. 2. Delete and add a photo to the praactice movie. 3. Save changes and exit	The movie edited should display the changes made to the user.	The film updated the information but took 10 minutes to be processed to other devices.	PASS	JanaeF
ShoppingCart_1	Cart_Module	P4	Verify that the cart feature correctly reflects the ticekets a user adds to it	1. The ticket added is for a "Now Playing" Movie 2. There are tickets available for selecting showing	1. Locate the movie of choice 2. Select tickets of choice 3. Add to cart but do not proceed to checkout 4. Check user cart	The cart should display the movie title, image, location, time, and ticket types that the user selected.	The cart displayed the correct format	PASS	JanaeF
Ticket_1	MovieTicket_Module	P5	Verify that the user can successfully choose ticket information.	1. Movie of choice is a valid movie 2. Movie of choice hs available inventory for ticket information of choice 3. Cart function works	1. Locate the film of chouce. 2. Select the "Buy Tickets" option. 3. Select ticket type "Child". 4. Add another ticket of type "Adult" and add to cart. 5. Add another ticket of type "Senior" and add to cart. 6. Add another ticket of type "Student" and add to cart.	The tickets of each type should be located in the user's shopping cart via shopping cart functionality button highlighted as an orange button of a cart on the screen.	The cart successfully stored all ticket types added.	PASS	JanaeF
Payment_1	Checkout_Module	P6	Verify that the user can successfully purchase a ticket	1. the ticket is for a valid movie/ showing 2. User has a valid online payment option listed	1. Locate a film and select ticket information 2. proceed to enter payment information and proceed to confirmation page. 3. Confirm details and click "confirm" located at the bottom of the information.	The system should be verified of the purchase and send the information to a third party to	The payment gateway successfully encrypts and processes it and transfers it to the bank.	PASS	JanaeF
Webpage_1	PageStatus_Module	P7	Verify that the maintenance page is displayed to the user when they try to interact with the SSC page during the time of maintenance	1. compatible browser with SSC page	1. Locate a working URL for the SSC webpage 2. Click the link	A webpage with the error message, "Sorry the Silver Screens Cinema wepage is under maintenance from (Approximate Time-frame). We are sorry for the inconvenience." alongside the mascot with a hard hat is displayed to the user.	Error message and illustrations is displayed	PASS	JanaeF
Review_1	UserReview_Module	P8	Verify that when a user clicks the like button that it shows the correct displays	1. SSC page is successfully launched 2. There are movies within the SSC webpage 3. There is a review for a movie	1. Locate a review left on any film in SSC wepage 2. Click the heart button located on the bottom right of a review	The heart button should be highlighted with a pink hue and the total number of likes that review has received is displayed to the right of the heart	The heart button is pink and the number of likes it has ever received displayed to the left of it	PASS	JanaeF
Review_2	Review_Module	P9	Verify that when a user reports a review that the system recieves notice	1. There is an accessible review left on a film on the SSC page	1. Locate the report button to the bottom left of the review and click 2. Enter "This is a test report." and click submit in the reasoning box	The system should receive notification regarding the report and successfully stores it with the other reviews	The message successfully went through the system along to the other reports.	PASS	JanaeF

Data Management Strategy

Revisions to architecture diagram and description are in section 4.6.

SQL Table #1: Movie Database

Films	Review	Trailer	Synopsis	Ratings
Kung Fu Panda 4	“My kid loved this movie…”	*insert trailer clip*	“The new dragon Warrior…”	3 stars
…	…	…	…	…
…	…	…	…	…
Oppenheimer	“Great story..”	*insert trailer clip*	“Follow the story of…”	5 stars

SQL Table #2: Theater Database

Theater	Seating Charts	Times	Capacity
National City	Standard and ADA	12:00 pm, 2:15 pm, …	200
…	…	…	…
…	…	…	…
La Jolla	Standard and ADA	1:30 pm, 3:00 pm, ….	305

SQL Table #3: Account Database

Username	Email	Password	Purchase History	Rewards
jkruz123	jmkruz@gmail.com	H@ppy:)5	“list of past purchases…” (shows movie, date, and price paid)	250 points
…	…	…	…	…
…	…	…	…	…
rekikdegefa_6	rdegefa08@gmail.com	blu6Ppl$	“list of purchases…”	56 points

Data Management Description

Our data management strategy uses SQL based databases, such as MySQL, because our information and data is relational and some data is built off of one another. We also have come to this decision due to the fact that this system is the best fit for our high transaction application of purchasing film tickets. Because we have to handle so much data, a well-structured database and high-powered computers are necessary in order to implement our system. If we go by SQL databases, we can work with tables that have a schema instead of having an unstructured system that would make it hard to keep track of information relationships.

Tradeoffs and Alternatives

Our team has decided that using 3 databases would be most beneficial and efficient to keep running as it makes accessing certain parts of our data easier. It is the most cost efficient choice because they each contain information that corresponds to one umbrella topic, so when searching and accessing data, it saves time rather than trying to find information that is all stored within the same database. SQL is also best for handling large scale updates.

A tradeoff of using more than 1 database is that we lose the ability to easily apply an update in one go since we are not all working with a single base code across our servers. With 1 database we can update all at once which can cut down our time on locating a database, but increases our time searching through the database.

An alternative would be using a NoSQL based database where we would split all of our data into multiple servers and computers, instead of using less computers that are more high-powered ones. With a NoSQL based database, our database would be a bit less structured but the developing and processing would become quicker.

REVISIONS

Our team decided to update some database descriptions in our architecture diagram in section 4.3. We just added more information to our movie database, such as a rating and synopsis category in our movie database and a capacity category in our theater database.

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