Movie Tickets Master

Software Requirements Specification

Version 4.0

3/28/24

Group 5
Kalyn Sheffield
Megan Lovell
Aaron Alegre

Project Github: [Sanmitr564/CS250-Github-Submission]

Prepared for
CS 250- Introduction to Software Systems
Instructor: Gus Hanna, Ph.D.
Spring 2024

Revision History

Date	Description	Author	Comments
2/15/24	Version 1.0	Kalyn Sheffield Megan Lovell Aaron Alegre	Completed Section 1-3.6
2/29/24	Version 2.0	Kalyn Sheffield Megan Lovell Aaron Alegre	Completed Sections 3.7, 4.1, & 4.2
3/14/24	Version 3.0	Kalyn Sheffield Megan Lovell Aaron Alegre	Updated Sections 4.1 & 4.2 Completed Section 4.3
3/28/24	Version 4.0	Kalyn Sheffield Megan Lovell Aaron Alegre	Updated Section 4.2 Completed Section 4.4

Document Approval

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

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

Table of Contents

1. Introduction

The introduction of the Software Requirements Specification (SRS) provides an overview of the document and the capabilities and uses of the Movie Tickets Master software.

1.1 Purpose

The purpose of this SRS is to describe the requirements needed for the Movie Tickets Master software and to list any limitations in the system's functionality to any potential developers.

1.2 Scope

The users of the Movie Tickets Master system are people interested in purchasing movie tickets, as well as returning or exchanging their tickets. Our goal with this system is to make it easier and faster to handle the purchasing of movie tickets.

1.3 Definitions, Acronyms, and Abbreviations

System	Refers to the Movie Tickets Master program in its entirety

1.4 References

These are the references:

- “Marvel Electronics and Home Entertainment/E-Store Project”(Software Requirements Specification - Example)
- “Software Engineering Software Requirements Specification(SRS) Document”(SoftwareRequirementsTemplate-Example)
- “IEEE Recommended Practice for Software Requirements Specifications”(IEEE SRS Standard(1998))

All references can be obtained from canvas resources.

1.5 Overview

The next sections of this document will have a description of the project, and what requirements are needed for the system to run. In section 2 of this document, we will describe what the project perspective is as well as the functionality, user characteristics, and constraints of the project. While section 3 will discuss all the different requirements needed for our project. Section 4 will show different diagrams and models to support the information mentioned in other sections before.

2. General Description

The Movie Tickets Master is a ticketing system run off of a web browser that is capable of handling at least 1000 customers at once and provides a variety of useful features like searching for movies based on genre and available showtimes.

2.1 Product Perspective

This product will make the entire process of purchasing and handling movie tickets faster and more accessible to users all around.

2.2 Product Functions

The system should be able to:

- Create reservations for movie showings
- Refund purchased reservations
- Exchange purchased reservations
- Convey information such as:
    - Showtimes
    - Reservation availability
    - Relevant information(synopsis, genres, standalone/series, actors, etc)
- Safely handle and store customer information
- Edit previously entered information
- Easily accept new information

2.3 User Characteristics

The eventual users of the Movie Tickets Master system are anyone with access to and capable of operating a web browser and has a method of online payment, such as all major credit cards and debit cards, PayPal, and ApplePay.

2.4 General Constraints

- Sensitive user information(names, credit card details, etc) must be encrypted
- The system should perform smoothly and reliably when there are less than 1000 concurrent users
- Information in the system should not be accessible by unauthorized actors

2.5 Assumptions and Dependencies

- The frontend of the system will be run on stable versions of Windows 10/11, macOS, Android, and iOS
- The frontend of the system will be accessed on stable versions of Chrome, Edge, Safari, Firefox, or Opera

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The user interface for the software will be usable on any web browser such as Google Chrome and Safari.

3.1.2 Hardware Interfaces

Our system needs to run on the internet, hardware that can connect to the internet shall be the hardware interface for our system.

3.1.3 Software Interfaces

Our system shall communicate with credit card companies for handling purchasing/returning options, the movie theaters that will be accessing our system for ticket management, and an external Tax system to calculate the tax.

3.1.4 Communications Interfaces

The Movie Tickets Master system will use the HTTPS protocol so that the users' information and the information displayed on our interface is secure and accurate.

3.2 Functional Requirements

3.2.1 System needs to be run on a web browser

3.2.1.1 The system will be able to be accessed and run from a web browser.
3.2.1.2 The system will allow users to make all of the movie and ticket selections they need to on their chosen web browser.
3.2.1.3 The system will need some level of network or Wi-Fi connectivity to operate on a web browser.
3.2.1.5 The system will notify the user of any connectivity issues.

3.2.2 Capable of handling a minimum of 1000 customers at once

3.2.2.1 The system should be able to handle 1000 customers without adversely affecting the user experience
3.2.2.2 When there are less than 1000 users using the system at once, assuming a download speed of 100 mb/s and a ping of 20 ms, webpages should load within 1-2 seconds
3.2.2.3 The system will not be expected to maintain the above constraints when handling more than 1000 users

3.2.3 Searching interface with the database of genre

3.2.3.1 The system shall display all different genres of movies
3.2.3.2 The system shall allow the user to select a specific genre
3.2.3.3 The system shall user to switch between genres
3.2.3.4 The system shall show movies in genre sections based on popularity
3.2.3.5 The system shall notify if there are no movies of that genre showing

3.2.4 Change Movie Availability Based on Selected Theater

3.2.4.1 The system will allow users to select one of the theaters supported by this website
3.2.4.2 The system will allow users to switch between theaters supported by this website
3.2.4.3 The system will change the movie catalog available to the user based on the selected theater
3.2.4.4 The system will adjust prices based on the pricing of the selected theater
3.2.4.5 The system will clearly display the currently selected theater on the webpage

3.2.5 Users can only buy 15 tickets at a time

3.2.5.1 The system will allow a user to buy 15 or less tickets at a time.
3.2.5.2 The system will allow the user to make ticket selections.
3.2.5.3 The system will allow users to choose a varying number of tickets less than 15.
3.2.5.4 The system will display the number of tickets the user has selected.
3.2.5.5 The system will display an error message if a user tries to purchase more than 15 tickets at one time.

3.2.6 Interface showing available movies and tickets

3.2.6.1 The system shall display movies available with the number of seats left.
3.2.6.2 The system shall allow users to search for movies.
3.2.6.3 The system will display movies based on the next showings available.
3.2.6.4 The system will show information about the movie if the user selects it.
3.2.6.5 The system will display a message if the user selects a showing after it has begun

3.2.7 Handle refunds/exchanges up until the point the movie starts or ticket is redeemed

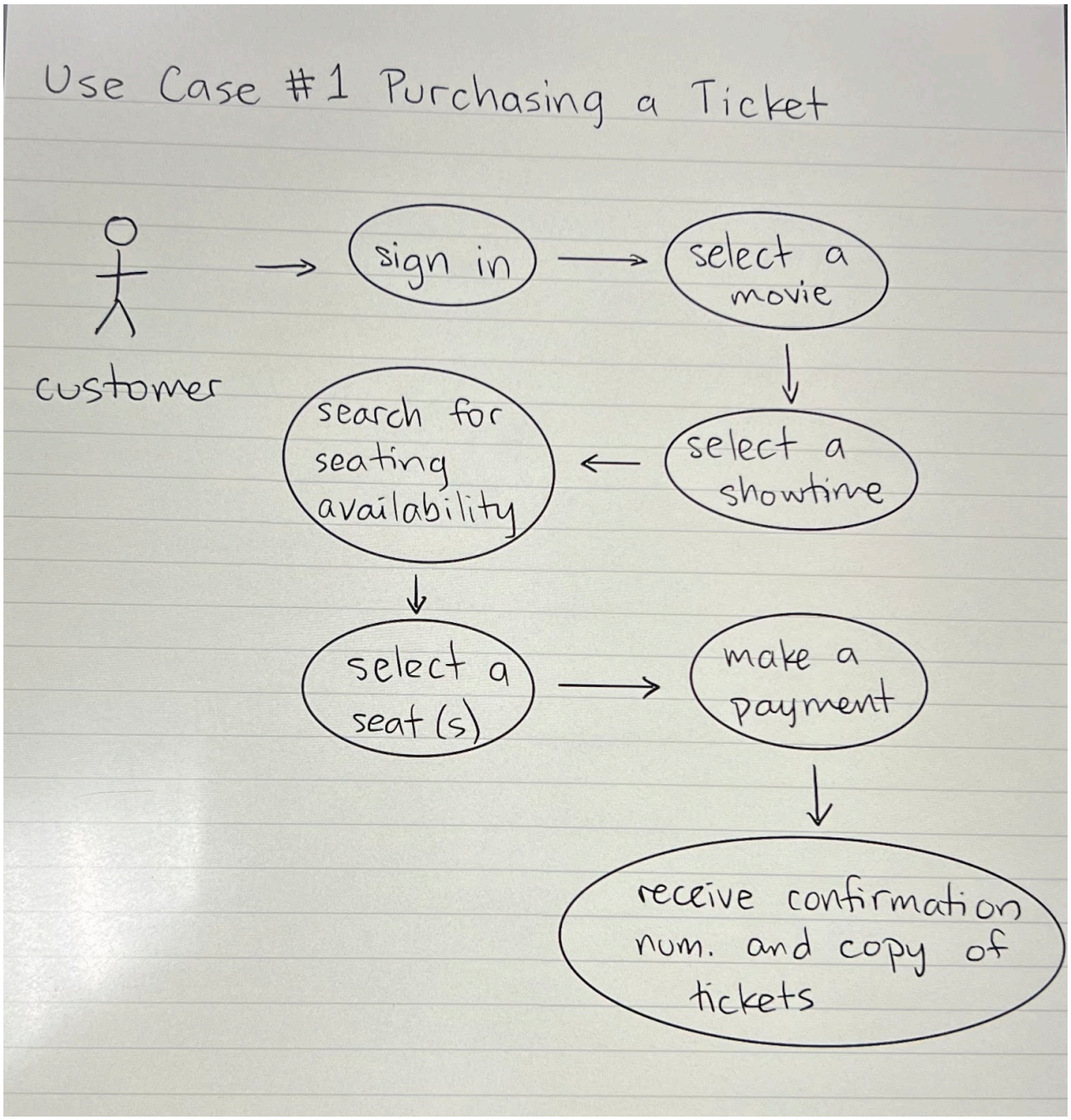
3.2.7.1 The system will allow a user to refund or exchange their ticket up until the time the movie starts or the ticket has been redeemed.
3.2.7.2 The system will allow a user to enter their confirmation number for their purchased tickets.
3.2.7.3 The system will display a message once the confirmation number has been entered and found.
3.2.7.4 The system will give the user a refund if they decide to return their tickets(s) or provide the user with a new confirmation number for their newly redeemed ticket.
3.2.7.5 The system will display an error message if the confirmation cannot be found.

3.2.8 Accepts and Protects User's Payment Method

3.2.8.1 The system allows payment methods to be all major credit/debit cards, Paypal, and Apple Pay.
3.2.8.2 The system will save a method of payment if you have a saved account.
3.2.8.3 The system will allow the user to pick which payment method they will use.
3.2.8.4 The system will display a message once the payment has been approved.
3.2.8.5 The system will display an error message if payment doesn’t go through.

3.3 Use Cases

3.3.1 Use Case #1 Purchasing a Ticket

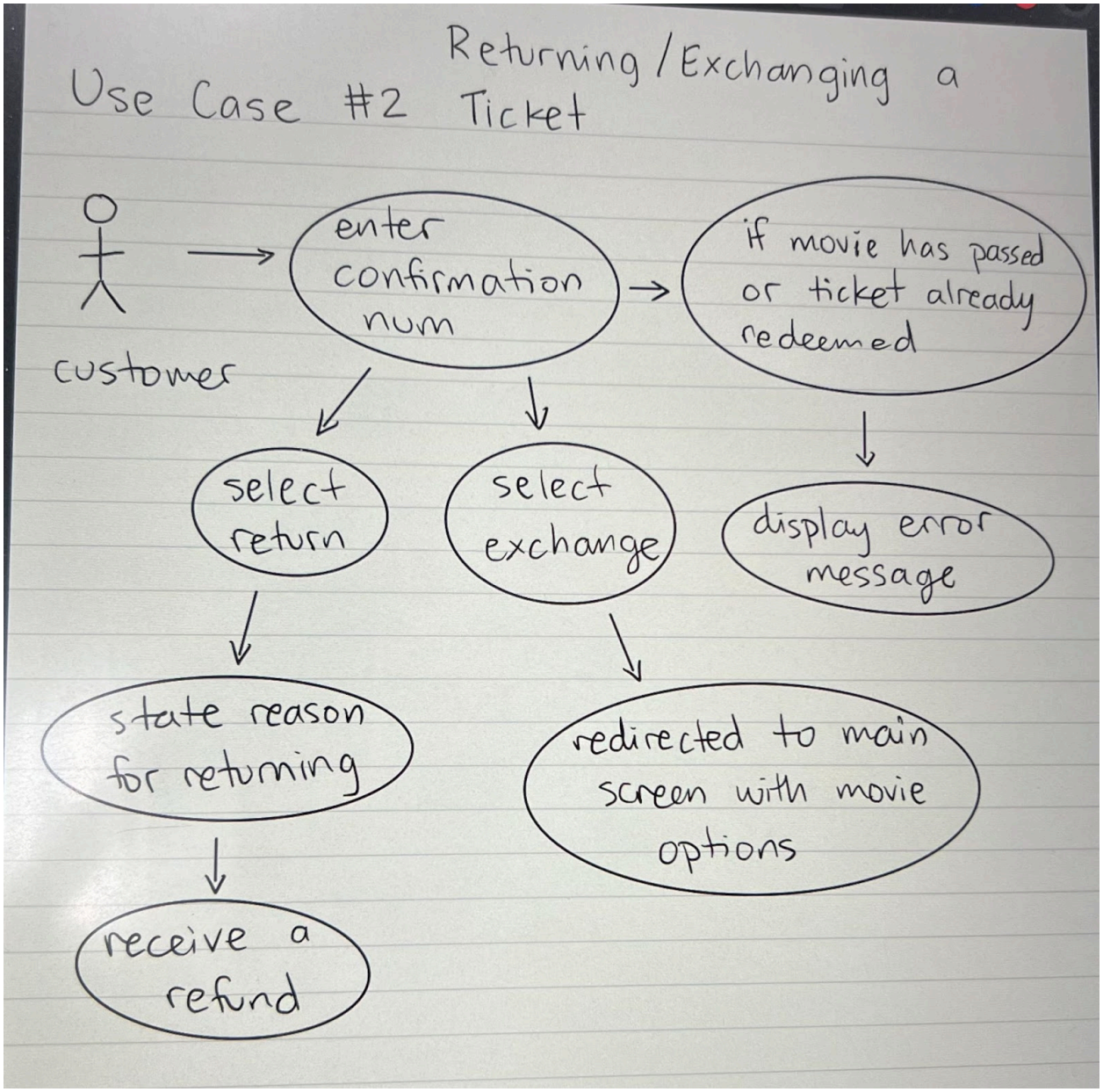


Actor(s): Customer
Flow of events:

1. Sign in
2. Select a movie
3. Select a showtime
4. Search for seating availability
5. Select a seat(s)
6. Make a payment

7. Receive a confirmation number and copy of tickets

3.3.2 Use Case #2 Returning/Exchanging a Ticket

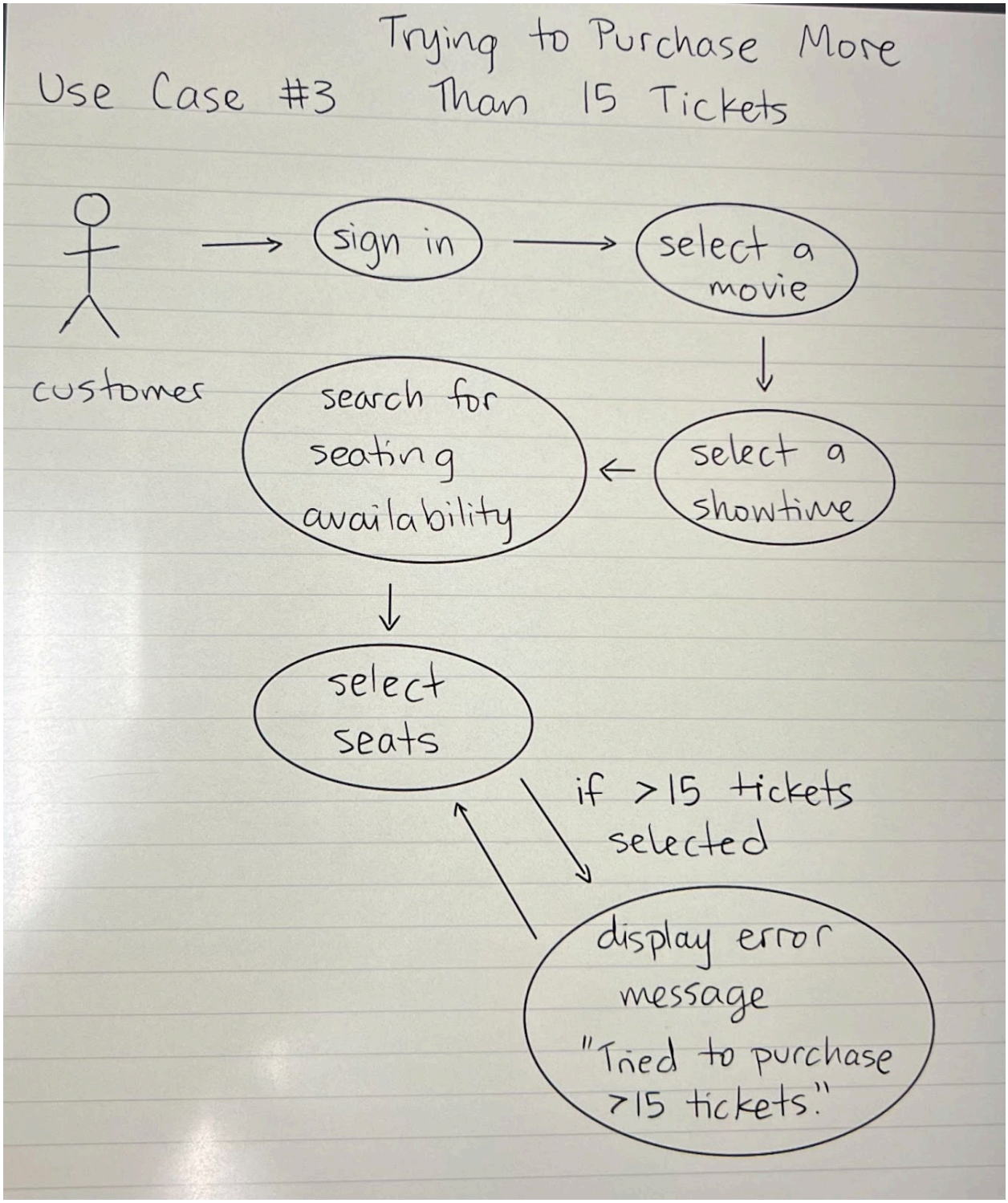


Actor(s): Customer
Flow of events:

1. Enter the confirmation number for tickets
2. If the movie time has already passed or the ticket has been redeemed, error message is displayed “Movie has already been played or ticket has already been redeemed”
3. Select if returning ticket for a refund or exchanging for a different ticket
4. If returning ticket for a refund, state reason for returning ticket and receive a refund

5. If exchanging ticket for a different ticket, get redirected to the main screen with the movie options

3.3.3 Use Case #3 Trying to Purchase More than 15 Tickets



Actor(s): Customer

Flow of events:

1. Sign in
2. Select a movie
3. Select a showtime
4. Search for seating availability
5. Select seats
6. Error message is displayed, “Tried to purchase more than 15 tickets”
7. Redirected back to seat selection

3.4 Classes / Objects

3.4.1 User

3.4.1.1 Attributes

3.4.1.1.1 Name: string
3.4.1.1.2 Username: string
3.4.1.1.3 Password: string
3.4.1.1.4 User ID: int
3.4.1.1.5 Tickets: ticket[]
3.4.1.1.6 Credit card information:

3.4.1.2 Functions

3.4.1.2.1 Purchase ticket
3.4.1.2.2 Refund ticket
3.4.1.2.3 Exchange ticket
3.4.1.2.4 Add credit card
3.4.1.2.5 Remove credit card
3.4.1.2.6 Change name

3.4.2 Movie

3.4.2.1 Attributes

3.4.2.1.1 Title: string
3.4.2.1.2 Poster: image
3.4.2.1.3 Genres: string[]
3.4.2.1.4 Synopsis: string
3.4.2.1.5 Actors: string[]
3.4.2.1.6 Studio(s): string[]
3.4.2.1.7 Series: string
3.4.2.2.8 Length: int

3.4.2.2 Functions

3.4.2.2.1 Get title
3.4.3.2.2 Get genres

3.4.3 Ticket

3.4.3.1 Attributes

3.4.3.1.1 Customer ID: int
3.4.3.1.2 Showing: showing
3.4.3.1.3 Seat: int
3.4.3.1.4 Confirmation number: int
3.4.3.1.5 Price: double

3.4.3.2 Functions

3.4.3.2.1 Cancel
3.4.3.2.2 Change seat
3.4.3.2.3 Refund

3.5 Non-Functional Requirements

3.5.1 Performance

Our system's performance is dependent on the internet connectivity the customer has and is run from a web server.

3.5.2 Reliability

To ensure the reliability of our system all of our databases by replicating our data to keep a backup of it so that we don't encounter any loss of data when inconvenient problems occur.

3.5.3 Availability

To ensure the availability of the system, we will have an agreement with an internet access provider who can guarantee that our system will have 99.9% availability to the internet.

3.5.4 Security

Our system needs to be secure so that the users can safely enter their personal data and payment information without the risk of their data being taken. The system will also log out of the users account after 10 minutes of inactivity to ensure the user's confidentiality. The user's password and credit/debit number will be hidden by different characters. Any data replicated as a backup will also be encrypted.

3.5.5 Maintainability

Our system will be maintained by configuration management tools to make sure that updates to the system's software run smoothly.

3.5.6 Portability

Our system will be portable because it can be run and accessed from any web browser, so any device that can access the internet can open a web browser to use our system.

3.6 Inverse Requirements

The system will not accept cash as a method of payment for the movie tickets. The system will not support the purchase of anything besides the movie tickets.

3.7 Design Constraints

The system will follow the standards set by modern graphical user interfaces, such as Microsoft's GUI or Google Chrome's GUI, such as having user-friendly features like a mouse, menus, and icons. The system will not have any requirements for minimum memory storage. The users must have access to the internet and have the ability to open a web browser. The response time of the system will take no longer than 3 minutes to load any of the interfaces the user might access.

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

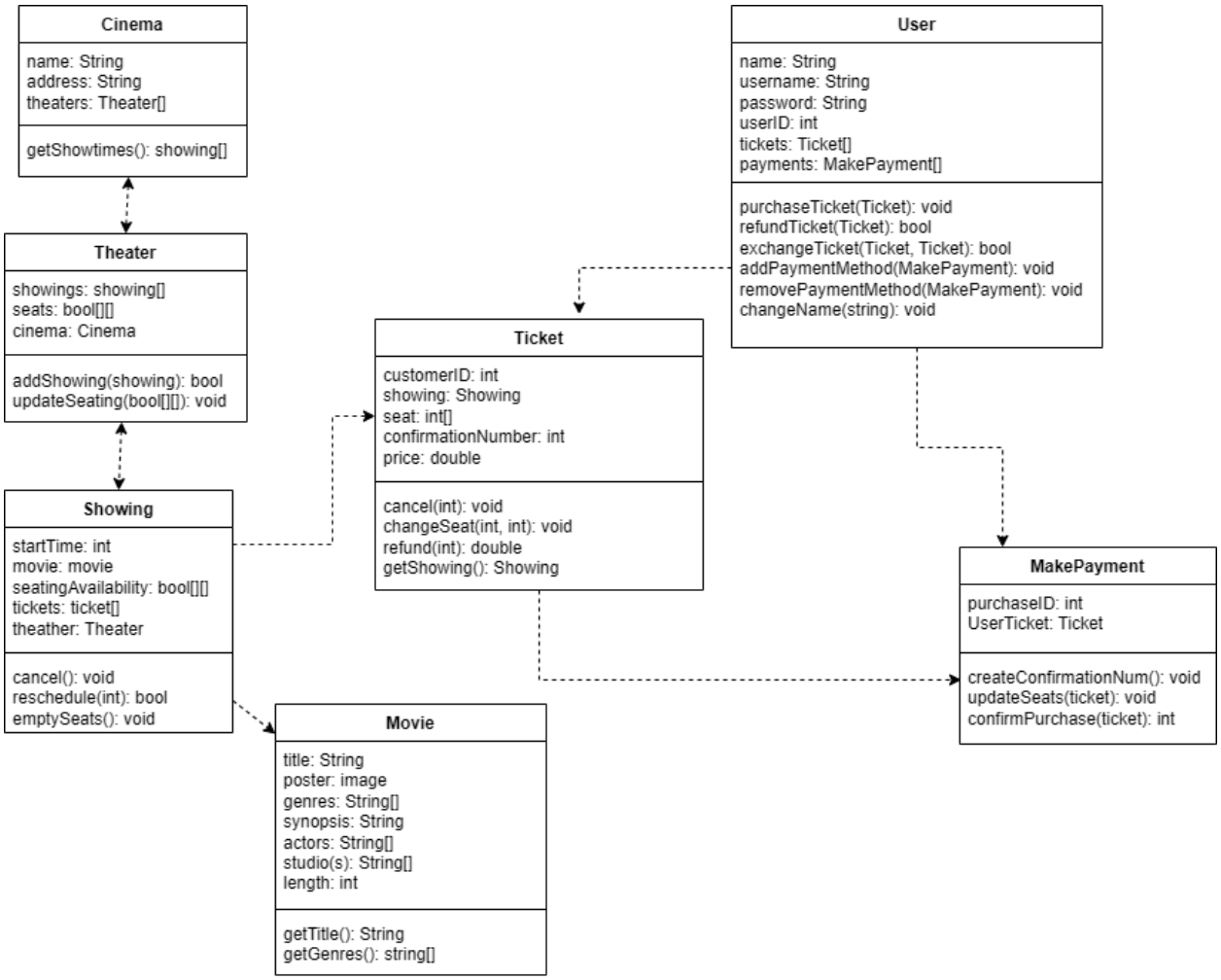
Catchall section for any additional requirements.

4. Analysis Models

Overview of System

The Movie Tickets Master system will allow users to maneuver between various interfaces and search for specific movies at different theaters and cinemas. The system's users will be able to make purchases for tickets on the website and have an account that saves their past ticket purchases and payment methods.

4.1 UML Diagram



4.1.1 UML Diagram Description

Cinema Class: The cinema class is for the cinema that the movies will play in.
The name attribute (String) is for the cinema name.
The address attribute (String) is for the address of the cinema.
The theaters attribute (Theater[]) is for the theaters in the cinema that the movie will be showing in.
The getShowTimes() operation (Showing[]) is for the showtimes of the movie in the cinema.

Ticket Class: The ticket class holds all the information for the showing of the user's choice.
The customerID(int) attribute is a unique combination of numbers that identifies the user's information.
The showing(showing) attribute contains all the details of the showing from the showing class.
The seat(int[]) attribute contains the details of which seat(s) was/were selected by the user.
The confirmationNumber(int) attribute is a unique combination of numbers that confirms the purchasing of the ticket.

The price(double) attribute shows how much the specific showing will be.
The cancel(int) operation cancels the ticket just purchased by inputting the confirmation number.
The changeSeat(int, int) operation changes to another available seat but inputs a column and row in the selected showing.
The refund(int) operation gives a refund for the ticket purchased by inputting the confirmation number.
The getShowing() operation gets the available showings.

Showing Class: The showing class contains relevant information about a movie showing
The startTime(int) attribute contains the time the movie is scheduled to start
The movie(movie) attribute contains an instance of the movie to be shown
The seatingAvailability(bool[][]) attribute contains a 2d array of seats that have and haven't been purchased
The tickets(ticket[]) attribute contains a list of tickets purchased for this showing
The theater(theater[]) attribute contains an instance of the theater the showing will take place in
The cancel() method cancels the showing, refunds, and informs all ticket holders
The reschedule(int) method reschedules the showing to the specified time, informs ticket holders, and offers a refund
The emptySeats() method empties all seats, refunds, and informs all ticket holders

Theater Class: The theater class is for the theater that the movie will play in.
The showings attribute (Showing[]) is for the showings of the movie in the theater.
The seats attribute (bool[][]) is for the seating in the theater.
The cinema attribute (Cinema) is for the cinema that the theater is located in.
The addShowing operation (bool) adds a showing to the theater with a given showing.
The updateSeating operation (void) updates the seating available in the theater with a given seating arrangement in type bool[][].

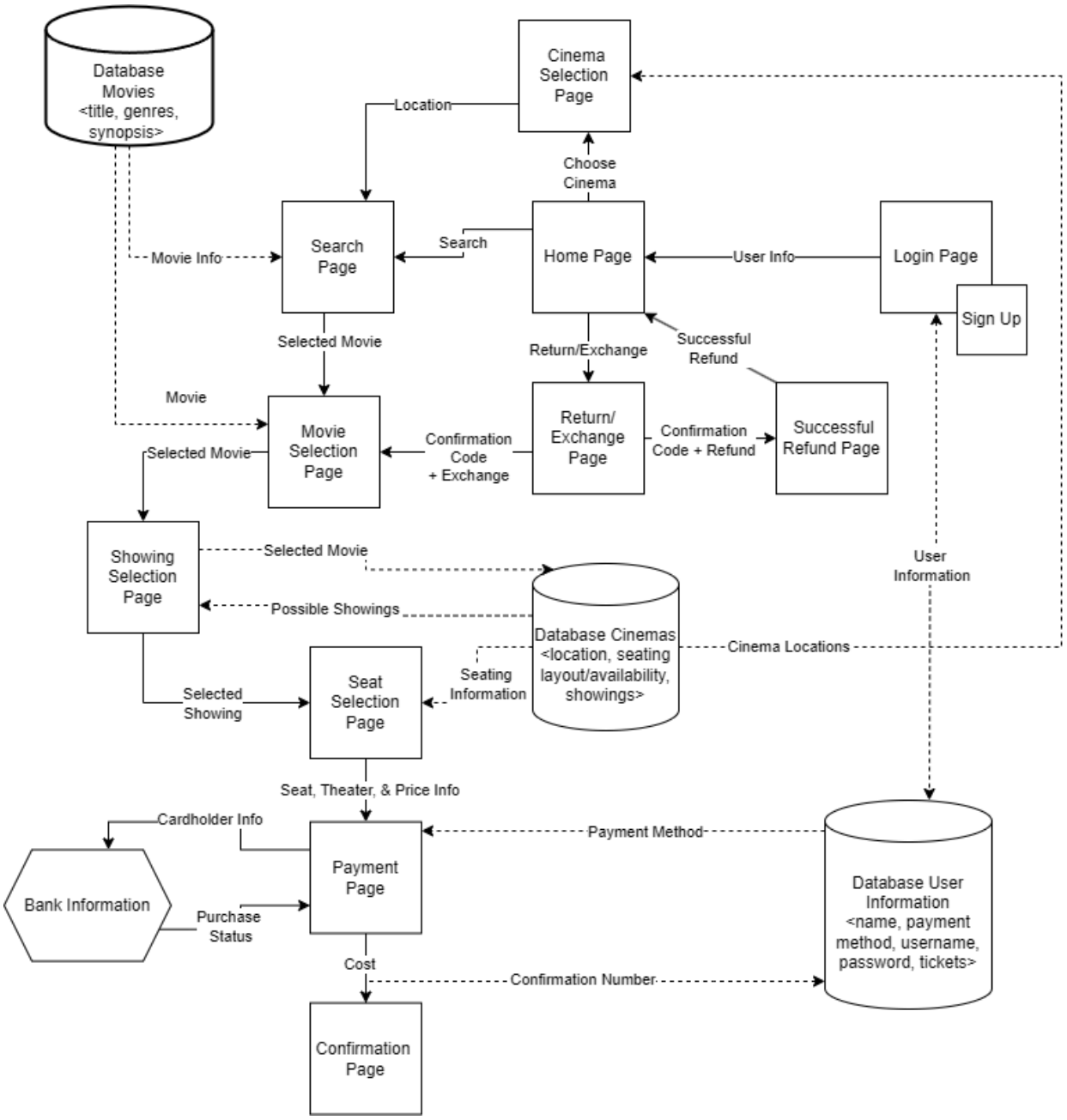
Movie Class: The movie class contains information about movies that are/were/will be available to watch
The title(string) attribute contains the title of the movie
The poster(image) attribute contains an image of the movie poster
The genres(stringp[]) attribute contains a list of genres applicable to the movie
The synopsis(string) attribute contains a short description of the film
The actors(string[]) attribute contains a list of actors starring in the movie
The studio(s)(string[]) attribute contains a list of studios involved in the production of the movie
The length(int) attribute contains the length of the movie in seconds
The getTitle() operation returns the title of the movie
The getGenres() operation returns the list of genres applicable to the movie

User Class: The user class is for the customers using the Movie Tickets Master.
The name(String) attribute is for the user's name.
The username(String) attribute is for the name saved for the user's account.

The password(String) attribute is for the password saved for the user's account.
The userID(int) attribute is for the user's unique userID saved to their account.
The tickets(Ticket[]) attribute is for the tickets the user has bought.
The payments(MakePayment[]) attribute is for the methods of payment the user has used for buying their tickets.
The purchaseTicket operation is for purchasing movie tickets.
The refundTicket(Ticket) operation refunds the entered movie ticket.
The exchangeTicket(Ticket, Ticket) operation exchanges the two tickets.
The addPaymentMethod(MakePayment) operation adds a payment method to the user's account.
The removePaymentMethod(MakePayment) operation removes the inputted payment method from the user's account.
The changeName(string) operation is for changing the name saved for the user's account.

MakePayment Class: The MakePayment class allows customers to purchase a ticket for a movie through Movie Tickets Master.
The purchaseID(int) attribute creates a unique combination of numbers to identify the purchase being made.
The userTicket(Ticket) attribute identifies the user ticket information from the ticket class.
The createConfirmationNum() operation creates a unique number that links back to the payment being made.
The updateSeats(ticket) operation updates the seat array in the ticket class with which seats are now available and unavailable after the purchase of the ticket goes through.
The confirmPurchase(ticket) operation confirms the payment of the ticket and the details that go along with it.

4.2 SWA Diagram



4.2.1 SWA Diagram Description

The users start at the login or sign-up page and can go to the home page via the user's information.

From the home page users can choose to refund/exchange a ticket, select a cinema to display movies from, or go to the search page.

From the return/exchange page, users will put in a valid confirmation code and if they choose to return, will get sent to the successful return page and then back to the home page. But if they chose to exchange the ticket, users would get sent to the movie selection page.

From the cinema selection page, users can choose a cinema near them, and are then redirected to the search page.
From the search page users can select a movie they are interested in and be directed to that movie's webpage.
From the movie selection page users can decide to purchase a ticket for that movie, at which point they are redirected to that showing's showing selection page.
From the showing page, users can select a showing to watch, and are redirected to that showing's seat selection page.
From the seat selection page, once the user has chosen their seats and has decided to pay, they are redirected to the payment page.
After payment info has been input, and after communicating with the bank to confirm payment, users are redirected to the confirmation page.

The user database exchanges information with the login page, to ensure that users are logged into the correct account.
The user database provides saved payment information to the payment page when a user is purchasing their ticket.
The user database receives a ticket purchase confirmation number from the confirmation page.
The theater database contains information pertaining to the layout and availability of seating and relays that to the seat selection page.
The movie database provides cinema locations to the cinema selection page.
The movie database provides movie info to the search page.
The movie database provides movie information to each movie page.
The movie database provides showing information to the showing selection page.

4.2.2 Most Recent SWA Diagram Description

- The Return and Exchange page was revised to have two separate output pages compared to one. When you do an exchange now, it takes you back to the Movie Selection page. When you do a refund now, it takes you to the Successful Refund page.
- The List of Movies Database was revised to be the Movies Database because the location of the theater the movie was playing in was moved down to the Theater Database.
- The Theater Database was revised to be called the Cinema Database because it contains data from the old Theater Database as well as the location of the theater that was brought over from the Movies Database.
- The Cinema Database was revised to point out to more pages, including the Showing Selection page and the Cinema Selection page. As well as to point back in from the Showing Selection page.

Timeline

- Design the structure and features of the Movie Tickets Master system by February 15th
- Coordinate with movie theaters and banks that will be connected with the Movie Tickets Master system by February 29th
- Create functional databases (insert, request, edit, delete) by March 14th

- Start building the Movie Tickets Master system and user interfaces by March 28th
- Release pilot of Movie Tickets Master for users to test by May 2nd

Tasks and Responsibilities

- Design the structure and features of the system will be completed by Department A
- Coordination with theaters and banks will be done by Department B
- Databases will be created by Department C
- Building of Movie Tickets Master system and user interface will be done by Department D
- Release the pilot of the ticketing system will be completed by Department E

4.3 Verification Test Plan

4.3.1 Test Plan #1

Selecting A Seat Test Case (System): The purpose of this test case is to make sure that the user can find and select the seating they want in the theater they chose, and that the seating availability is up to date. This test case is supposed to check that the MovieSelectionPage, ShowingSelectionPage, and SeatSelectionPage features of the Movie Tickets Master are working correctly.

Payment test case(Functional): The purpose of this test case is to make sure that when a user inputs a valid type of payment, that Movie Tickets Master is able to access that information from a valid bank provider. Then from that charge the user the amount that was displayed to them and in return give a valid confirmation number. This test case is checking the PaymentPage and the ConfirmationPage to see if all the functionalities work together to fulfill the task of paying for a ticket.

SignUp Test Case (Unit): The purpose of this test case is to ensure that the user can sign up and make a Movie Tickets Master account. This test case is supposed to check that the SignUpPage feature of the Movie Tickets Master is working correctly.

4.3.2 Test Plan #2

Exchanging a Ticket Test Case(System): The purpose of this test case is to make sure that should the user decide to exchange their ticket for a different one, the system in place for that is functional. This system will check to ensure that the HomePage, Return/ExchangePage, MovieSelectionPage, ShowingSelectionPage, PaymentPage, and ConfirmationPage are all working correctly,

Search Page Test Case (Functional): The purpose of this test case is to make sure that the user can search for and find the movie they are looking for at the cinema they selected. This test case is supposed to check that the Search Page feature of the Movie Tickets Master is working correctly and displaying the available movies.

Change Showing Selection(Unit): The purpose of this case is to make sure that if a user decides that they don't want the showing that they initially chose, they can change to a different one as

many times as they want to. This test case checks the ShowingSelectionPage to see if users have the ability to make as many changes.

4.3.3 Test Cases

CS250 Test Cases .xlsx

Test Cases									
TestCaseId	Component	Priority	Description/Test Summary	Pre-requisites	Test Steps	Expected Result	Actual Result	Status	Test Executed By
Login_1F	LoginAndSignUpPage HomePage	P1	Verify that when a user enters their username and password, it takes them to the home page	Tester has already made an account	1. Navigate to movieticketsmaster.com/login 2. Insert valid username and valid password into corresponding text boxes 3. Click "Log In" 4. Get directed to the Home Page	User should be logged in and directed to the Movie Tickets Master Home Page	User is logged in and directed to the Movie Tickets Master Home Page	Pass	TesterA
Payment_2F	PaymentPage ConfirmationPage	P5	Verify that when a user enters their payment method and information, it directs them to the Confirmation Page and their Confirmation Number	Tester has already selected their ticket and seating	1. Navigate to the seat selection page for a movie showing 2. Select seats and click "Head to Checkout" 3. Enter valid payment information into corresponding text boxes 4. Press "Confirm Purchase"	After entering payment information, user should be directed to the confirmation page and is given a confirmation number	After entering payment information, user is be directed to the confirmation page and is given a confirmation number	Pass	TesterB
SelectingASeat_1S	MovieSelectionPage ShowingSelectionPage SeatSelectionPage	P4	Verify that when a user wants to select a seat to the theater and movie that they are choosing if the seat is open	Tester has already selected their theater and showing	1. Navigate to seat selection for a certain movie showing 2. Check the displayed seating for the theater 3. Compare this seating diagram to the real seating diagram	The seating diagram displayed on the website should match the seat layout of the theater in real life	The user sees the correct seating availability to the theater they selected	Pass	TesterC
ExchangingATicket_2S	HomePage Return/ExchangePage MovieSelectionPage ShowingSelectionPage SeatSelectionPage PaymentPage ConfirmationPage	P6	Verify that when a user wants to exchange a ticket, after inputting a valid confirmation code it takes them back through the process of selecting a movie, showing and seating paying, and receiving a new confirmation code.	Tester has already successfully brought a ticket and has a valid confirmation code	1. Navigate to movieticketsmaster.com/refunds 2. Enter valid confirmation code 3. Select "Exchange" 4. Choose new movie 5. Choose seats 6. Make sure exchange discount is applied 7. Pay 8. Get new confirmation number	The user can successfully exchange their ticket for a new one	After the user exchanges their ticket, they are able to get a new one with a new Confirmation Number	Pass	TesterD
Login_3F	LoginAndSignUpPage HomePage	P1	Verify that when a user enters their username and password, it takes them to the home page	Tester has already made an account	1. Navigate to movieticketsmaster.com/login 2. Insert valid username and invalid password into corresponding text boxes 3. Click "Log In" 4. Stay on Login Page	User should be logged in and directed to the Movie Tickets Master Home Page	User stays on the login page, entries clear, and the "username or password is invalid" message is shown	Fail	TesterE
ChangeUsername_1U	HomePage	P7	Verify that when a user changes their username, it is correctly saved and displayed on home page	Tester has already a Movie Tickets Master account and is logged in	1. Navigate to movieticketsmaster.com 2. Select "Change username" from settings dropdown 3. Put new username into text box 4. Click "Save changes"	User should be able to successfully change their username and new username is displayed on home page	User changes their username and the new username is correctly displayed on the Home Page	Pass	TesterF
ChangeShowingSelection_2U	ShowingSelectionPage	P3	Verify that a user can select a showing and change their showing selection on the Showing Selection Page	Tester has already selected a movie	1. Navigate to movieticketsmaster.com/(valid-movie) 2. Select a showing 3. Select a different showing	User should be able to successfully select a different showing from what they originally selected	User can successfully select a different showing from what they originally selected	Pass	TesterG
SignUp_3U	SignUpPage	P0	Verfiy that a user can sign up and make a Movie Tickets Master account	Tester is already on the Movie Tickets Master Sign Up Page	1. Navigate to movieticketsmaster.com/sign-up 2. Fill out appropriate text boxes 3. Click "Create account" 4. Get directed to the login page	User should be able to create a new account and be redirected to the login page	User successfully created a new account and is redirected to the Login Page	Pass	TesterH
Payment_4F	PaymentPage ConfirmationPage	P5	Verify that when a user enters their payment method and information, it directs them to the Confirmation Page and their Confirmation Number	Tester has already selected their ticket and seating	1. Enter valid payment information into corresponding text boxes 2. Press "Confirm Purchase" 3. Get redirected to Confirmation Page with valid confirmation number	After entering payment information, user should be directed to the confirmation page and is given a confirmation number	After entering payment information, user stays on the payment page and error message is shown "invalid form of payment" and boxes clear	Fail	TesterI
SearchPage_5F	SearchPage	P2	Verify that a user can search for and find a movie with a showing at the cinema they selected	Tester has logged in to Movie Tickets Master and has selected a cinema	1. Navigate to movieticketmaster.com/search 2. Enter "Spirited Away" into the search bar 3. Check results	After entering the title the user should be able to click a link which takes them to the "Spirited Away" movie page	After entering the title, the movie does not show up in the list of selections	Fail	TesterJ

4.4 Data Management

4.4.1 Data Management Strategy

The data management strategy for Movie Tickets Master uses SQL technology with an organization system of multiple databases. The SWA diagram above has three databases: User Information Database, Cinema Database, Movies Database. The User Information Database was chosen to store user account information in one place and keep it more secure by having it separated from the other databases. The Cinema Database was chosen to store information about the locations of cinemas, seating in the theaters, and available showings of the movies. The Movies was chosen to store information about the titles, genres, and descriptions of movies.

4.4.2 Trade Off Discussion

One possible alternative to the SQL technology in the Movie Tickets Master data management strategy is a non-SQL technology. One trade off between SQL and non-SQL is that SQL is more structured in how it stores data while non-SQL is not structured. Another trade off between SQL and non-SQL is that non-SQL databases are horizontally scalable so they can be scaled by adding more servers to the system, while SQL databases are vertically scalable so they can only be scaled by increasing the capacity of the system's hardware.

One possible alternative to the multiple databases organization system in the Movie Tickets Master data management strategy is to use the traditional approach to data management where the data is usually stored in separate files. One trade off between having databases versus having files to organize data is that the database method has better protection of the data because it is easier to monitor and control. Another trade off between having databases versus having files is that databases reduce the amount of redundant data which leads to more efficient use of storage space.

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