Movie Tickets Master

Software Requirements Specification

Version 2.0

2/29/24

Group 5
Kalyn Sheffield
Megan Lovell
Aaron Alegre

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	Section 1-3.6
2/29/24	Version 2.0	Kalyn Sheffield Megan Lovell Aaron Alegre	Sections 3.7 & 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

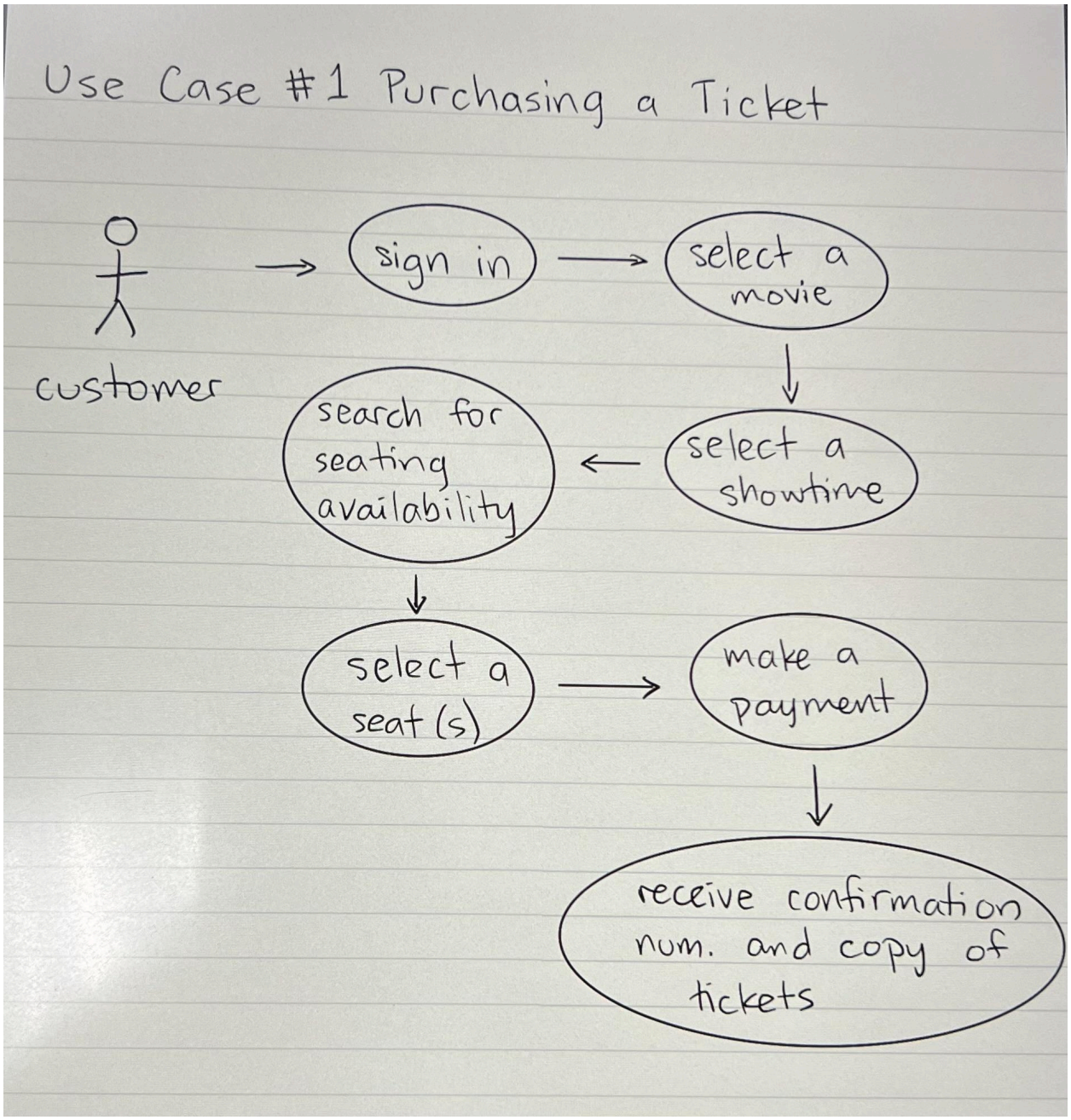
3.2.7.1 The system will allow a user to refund or exchange their ticket up until the time the movie starts or the ticket has been redeemed.
3.2.7.2 The system will allow a user to enter their confirmation number for their purchased tickets.
3.2.7.3 The system will display a message once the confirmation 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

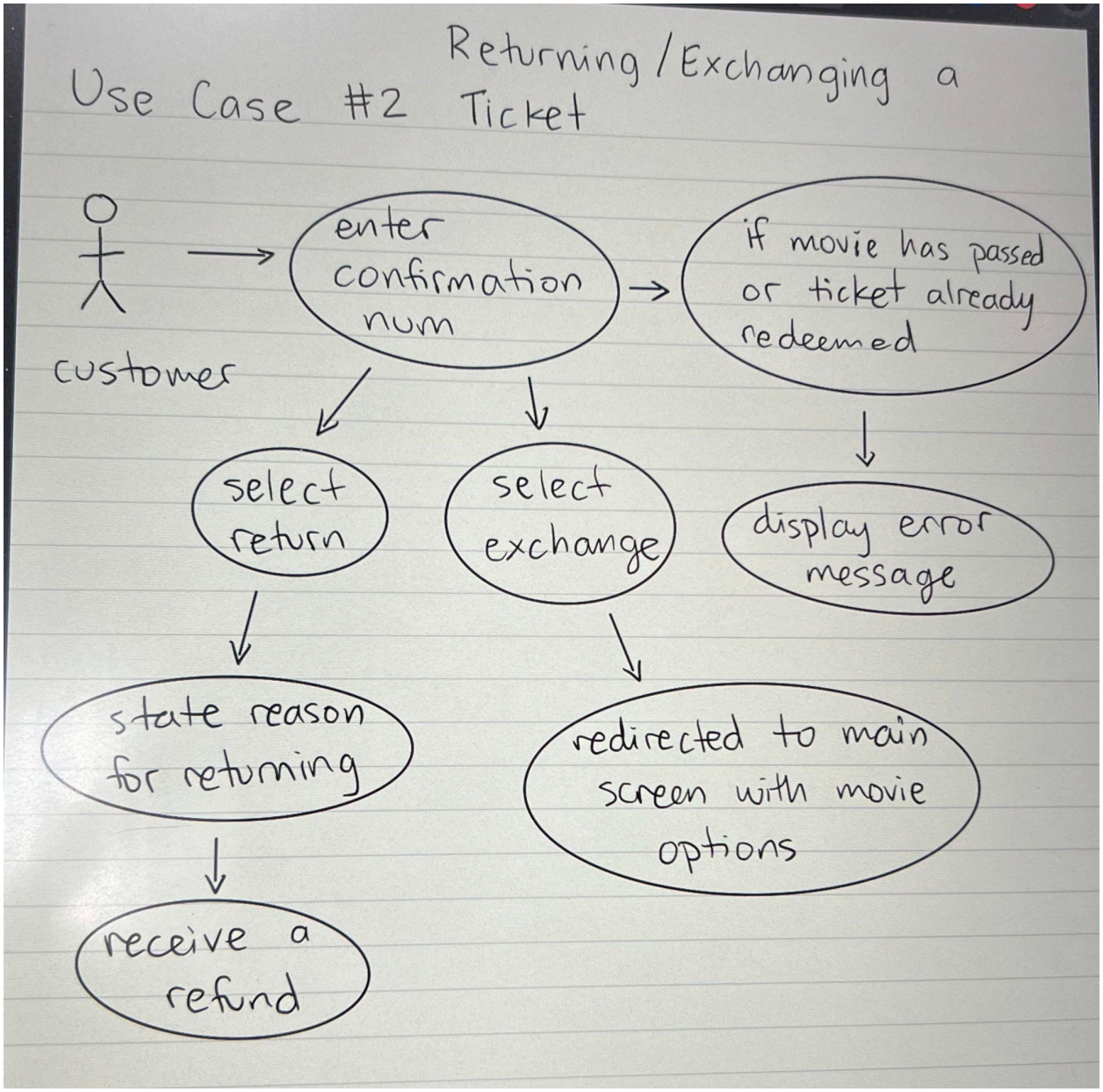


Actor(s): Customer
Flow of events:

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

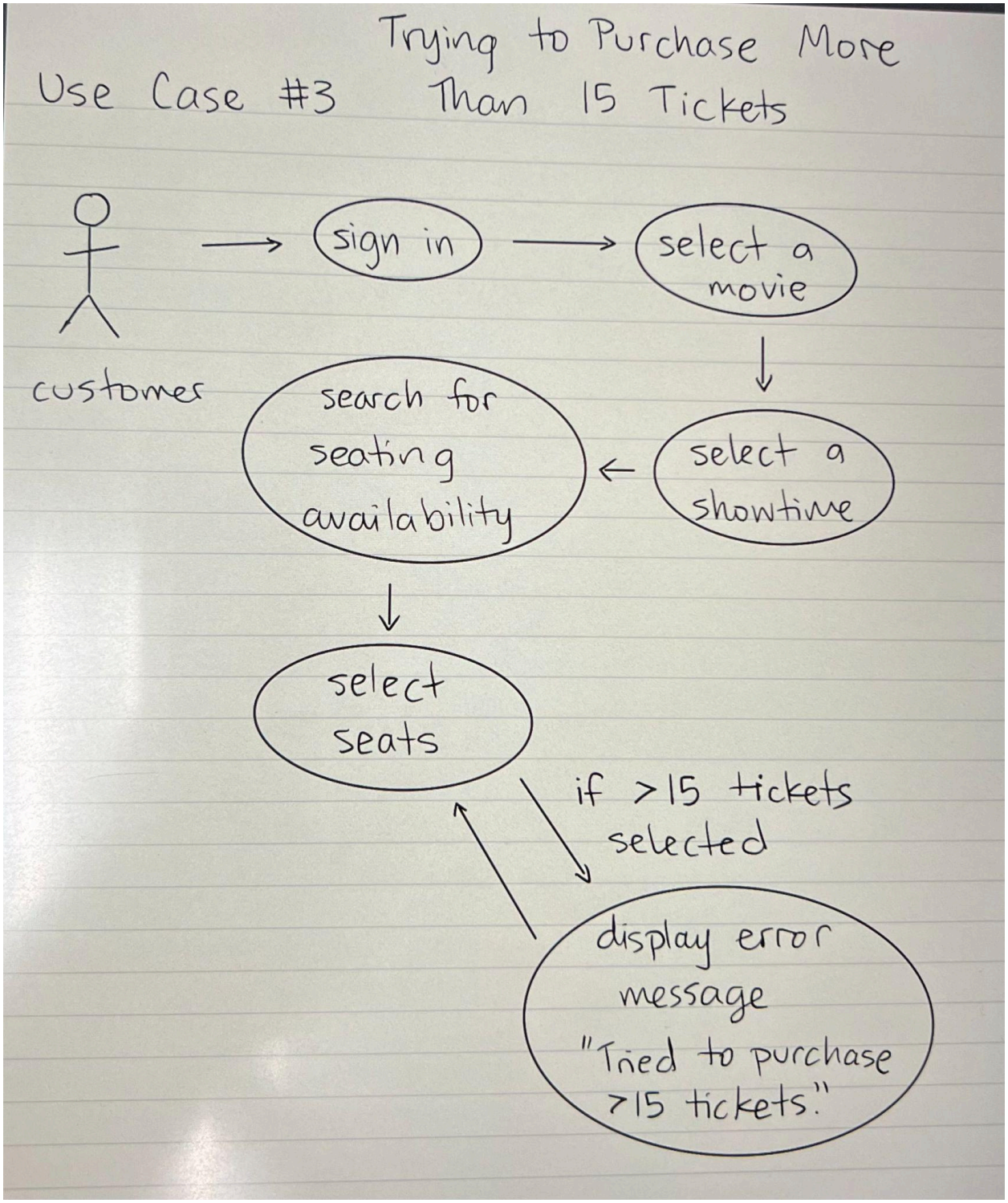


Actor(s): Customer
Flow of events:

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

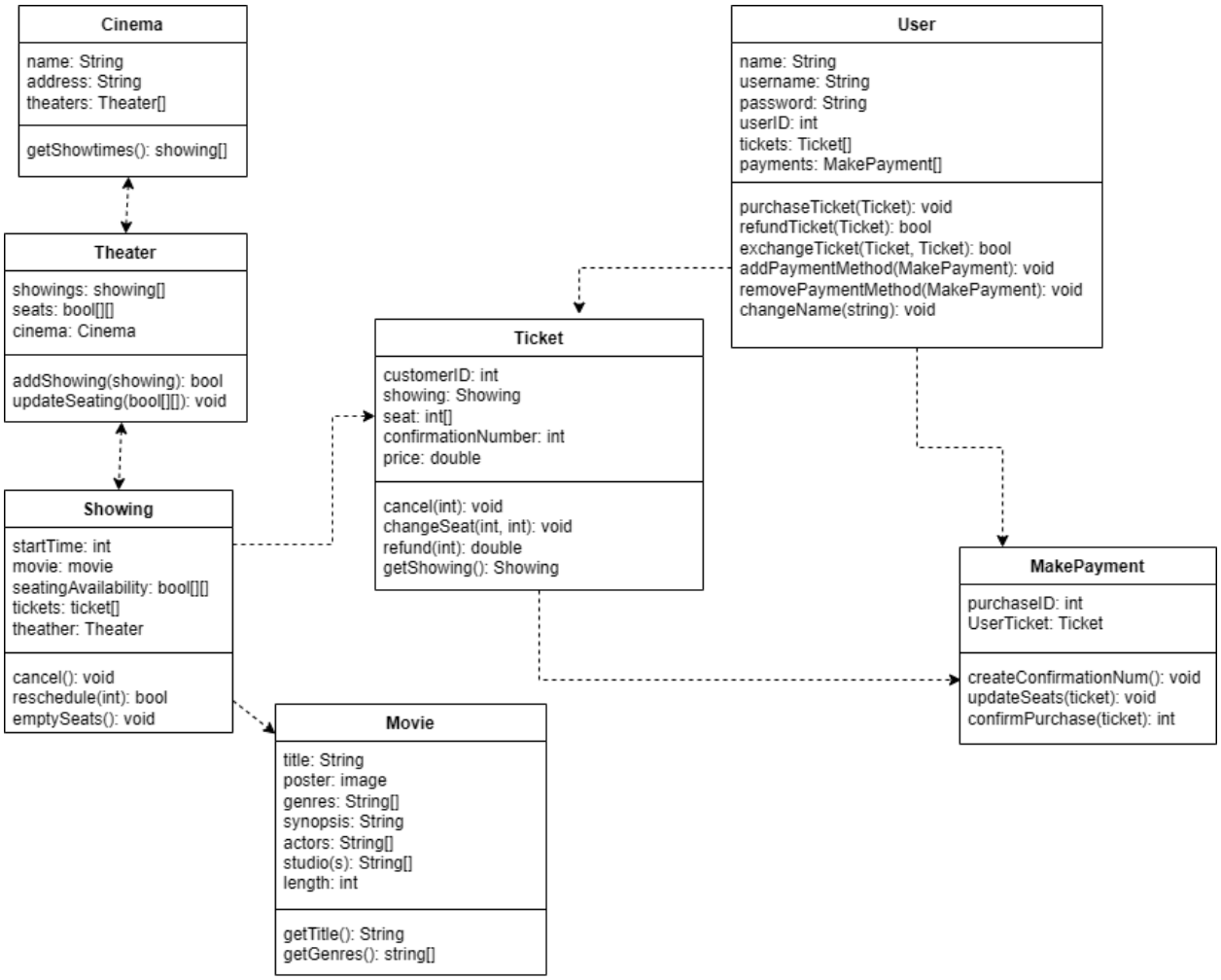
Catchall section for any additional requirements.

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[][].

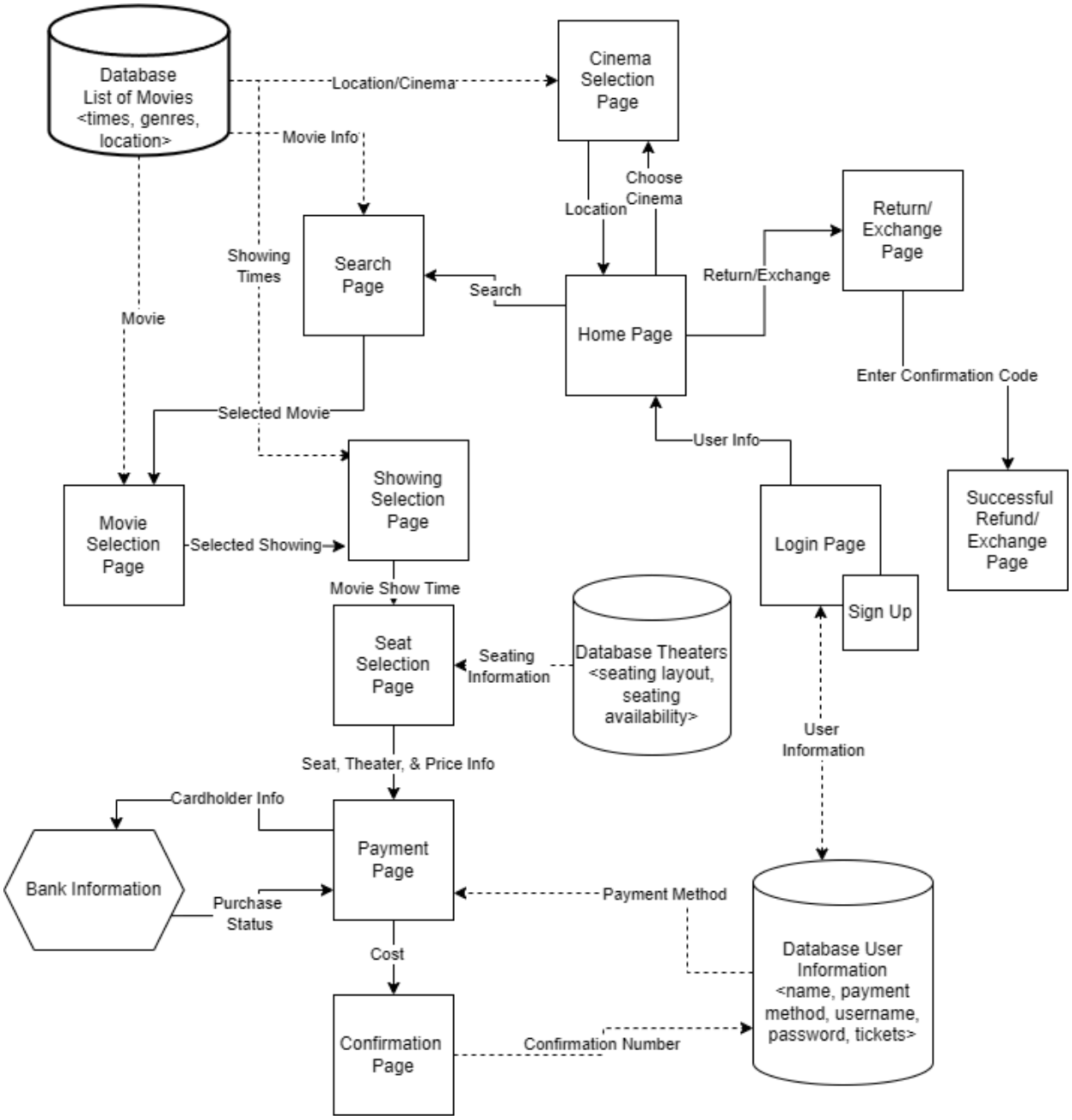
Movie Class: The movie class contains information about movies that are/were/will be available to watch
The title(string) attribute contains the title of the movie
The poster(image) attribute contains an image of the movie poster
The genres(stringp[]) attribute contains a list of genres applicable to 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, if successful, users will be redirected to the successful refund/exchange page
From the cinema selection page, users can choose a cinema near them, and are redirected to the home page
From the search page users can select a movie they are interested in and be directed to that movies 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

Timeline

- Requirements Specification (Sections 1-3.6) were completed by February 15th.
- Software Design Specification (Sections 3.7, and 4) were completed by Today(February 29th)
- SDS: Test Plan(Sections TBD) will be completed by March 14th
- Architecture Design with Data Management(Sections TBD) will be completed by March 28th
- Final Report will be completed by May 2nd

Tasks and Responsibilities

All members worked equally on the different sections of the SRS. We split up the larger sections and worked on our parts of that section at the same time. We did not have specific responsibilities or tasks because we worked together on each section.

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