Software Requirements Specification Template

Software Engineering

The following annotated template shall be used to complete the Software Requirements Specification (SRS) assignment.

Template Usage:
Text contained within angle brackets (‘<’, ‘>’) shall be replaced by your project-specific information and/or details. For example, <Project Name> will be replaced with either ‘Smart Home’ or ‘Sensor Network’.

Italicized text is included to briefly annotate the purpose of each section within this template. This text should not appear in the final version of your submitted SRS.

This cover page is not a part of the final template and should be removed before your SRS is submitted.

<Theater Ticketing>

Software Requirements Specification

<Version 4>

<3/28/2024>

<Group #3>
<Elizabeth Reynante, Paris Cabatit, Jazmin Gallegos>

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

Revision History

Date	Description	Author	Comments
<2/15>	<Version 1>	<Elizabeth Reynante, Paris Cabatit, Jazmin Gallegos>	<First Revision>
<2/29>	<Version 2>	<Elizabeth Reynante, Paris Cabatit, Jazmin Gallegos>	<Second Revision>
<3/14>	<Version 3>	<Elizabeth Reynante, Paris Cabatit, Jazmin Gallegos>	<Third Revision>
<3/28>	<Version 4>	<Elizabeth Reynante, Paris Cabatit, Jazmin Gallegos>	<Fourth Revision>

Document Approval

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

Signature	Printed Name	Title	Date
	<Elizabeth Reynante, Paris Cabatit, Jazmin Gallegos>	Software Eng.	3/28
	Dr. Gus Hanna	Instructor, CS 250	

Table of Contents

1. Introduction

The introduction to the Software Requirement Specification (SRS) document should provide an overview of the complete SRS document. While writing this document please remember that this document should contain all of the information needed by a software engineer to adequately design and implement the software product described by the requirements listed in this document. (Note: the following subsection annotations are largely taken from the IEEE Guide to SRS).

1.1 Purpose

What is the purpose of this SRS and the (intended) audience for which it is written?

This document intends to help the developers on this project understand the requirements of the online system, as well as its capabilities and limitations in a detailed manner.

1.2 Scope

This subsection should:
(1) Identify the software product(s) to be produced by name; for example, Host DBMS, Report Generator, etc
(2) Explain what the software product(s) will, and, if necessary, will not do
(3) Describe the application of the software being specified. As a portion of this, it should:
(a) Describe all relevant benefits, objectives, and goals as precisely as possible. For example, to say that one goal is to provide effective reporting capabilities is not as good as saying parameter-driven, user-definable reports with a 2 h turnaround and on-line entry of user parameters.
(b) Be consistent with similar statements in higher-level specifications (for example, the System Requirement Specification) , if they exist.What is the scope of this software product.

This SRS will define the specifications for a movie theater ticketing system. The system will allow customers to efficiently purchase tickets for their desired movie. In addition, customers will be able to choose the time and place that the movie will be played. Based on the user's location and if location access is granted, users will be able to choose a movie theater from a list of theaters within a set distance from their location. We will also provide customers the ability to select how many tickets they want as well as where their seats will be. The system will display various categories of tickets such as kids, adults, and seniors that differ in price. Once the customer has chosen everything to their desire, the system will redirect them to input their payment information and email and confirm that all ticket details are correct. During the ticket selection and payment process, customers' chosen seats will be held for a limited amount of time (i.e. 10 minutes) and once that time has passed, a screen will pop up explaining that they ran out of time. The customer will then click a button that will return them to the home page. After payment is complete, the system will send a confirmation email and receipt to the customer that includes a QR code to be scanned upon arrival at the movie theater.

1.3 Definitions, Acronyms, and Abbreviations

This subsection should provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS. This information may be provided by reference to one or more appendixes in the SRS or by reference to other documents.

No definitions, acronyms, or abbreviations are needed yet.

1.4 References

This subsection should:
(1) Provide a complete list of all documents referenced elsewhere in the SRS, or in a separate, specified document.
(2) Identify each document by title, report number - if applicable - date, and publishing organization.
(3) Specify the sources from which the references can be obtained.
This information may be provided by reference to an appendix or to another document.

No references yet.

1.5 Overview

This subsection should:
(1) Describe what the rest of the SRS contains
(2) Explain how the SRS is organized.

The rest of the SRS document is split into sections to further explain the details of our expectations for our movie theater ticket system. Section 1 introduces our ideas and what we expect to see in developing the movie theater system. Section 2 describes the broad issues that affect the product and its requirements. It's crucial to remember that this section only serves to clarify the requirements, it doesn't include specifics. Section 3 contains the Design Requirements (D-requirements). These are essential for guiding the software design, development, and testing phases of the project. The last section, section 4, will list all of the analytical models that were used to create the specific requirements that were previously provided in this SRS. A narrative description and an introduction should be included in every model. Every model should also be able to track back to the SRS's needs.

2. General Description

This section of the SRS should describe the general factors that affect 'the product and its requirements. It should be made clear that this section does not state specific requirements; it only makes those requirements easier to understand.

2.1 Product Perspective

This subsection of the SRS puts the product into perspective with other related products or projects. (See the IEEE Guide to SRS for more details).

The system will need access to the databases of all movie theaters in order to display the layout of each individual theater and what specific seats are available (i.e. wheelchair/handicapped or regular). We will also use an already established payment system that users will be redirected to when it is time for them to complete their payment.

2.2 Product Functions

This subsection of the SRS should provide a summary of the functions that the software will perform.

This product will ensure movie tickets are purchased more quickly and efficiently, making it easier for the user and those who receive the data to maneuver the information.

2.3 User Characteristics

This subsection of the SRS should describe those general characteristics of the eventual users of the product that will affect the specific requirements. (See the IEEE Guide to SRS for more details).

Characteristics to consider for eventual users that shape the development of our system include age, technical expertise, disability, and possession of alternate electronic payment methods. Age will determine what kind of ticket the customer will purchase and therefore the price of the ticket. Technical expertise is dependent on how familiar the user is with using online applications and web browsers to purchase items. If the user has a disability, they will be able to choose a seat that caters to the space and comfort they need. Lastly, the system will need to accommodate users who wish to pay using another method of payment besides a credit card (i.e. Apple Pay).

2.4 General Constraints

This subsection of the SRS should provide a general description of any other items that will limit the developer's options for designing the system. (See the IEEE Guide to SRS for a partial list of possible general constraints).

Some general constraints we will face in designing the system include interfaces to other applications, safety and security, and regulatory policies. We will be redirecting the user to a separate payment system which could create issues since we are not the developers/owners of that specific system. While users are on this separate payment screen we also need to ensure that it is on a secure domain so that none of their personal information can be accessed/stolen. In addition, we cannot guarantee that minors will not be able to purchase rated R tickets because users can easily lie about their age. To follow this policy, the theaters themselves would have to check IDs to make sure that customers are 18 years of age or older.

2.5 Assumptions and Dependencies

This subsection of the SRS should list each of the factors that affect the requirements stated in the SRS. These factors are not design constraints on the software but are, rather, any changes to them that can affect the requirements in the SRS. For example, an assumption might be that a specific operating system will be available on the hardware designated for the software product. If, in fact, the operating system is not available, the SRS would then have to change accordingly.

Assumptions and dependencies that affect the requirements are stable internet connection. To access the website internet connection must be secured and strong. A successful secure payment system will introduce a secure payment window to allow the customer to safely pay for their designed tickets. The website updates accordingly when people buy tickets and will keep those reserved seats off-limits. This helps prevent customers from buying the same tickets and causing confusion and stress among them. The website is compatible with the IOS and Android systems, to allow a more widespread number of users. The system will have regular updates to keep it

running smoothly.

3. Specific Requirements

This will be the largest and most important section of the SRS. The customer requirements will be embodied within Section 2, but this section will give the D-requirements that are used to guide the project's software design, implementation, and testing.

Each requirement in this section should be:

- *Correct*
- *Traceable (both forward and backward to prior/future artifacts)*
- *Unambiguous*
- *Verifiable (i.e., testable)*
- *Prioritized (with respect to importance and/or stability)*
- *Complete*
- *Consistent*
- *Uniquely identifiable (usually via numbering like 3.4.5.6)*

Attention should be paid to carefully organize the requirements presented in this section so that they may be easily accessed and understood. Furthermore, this SRS is not the software design document, therefore one should avoid the tendency to over-constrain (and therefore design) the software project within this SRS.

3.1 External Interface Requirements

3.1.1 User Interfaces

3.1.1.1 The system is compatible with different web browsers (Chrome, Internet Explorer, Safari, etc.).
3.1.1.2 The user interface can be implemented with any tool or software package (Java, Applet, etc.).

3.1.2 Hardware Interfaces

3.1.2.1 Since the application must run over the internet, all the hardware shall be required to connect to the internet e.g. Modem, WAN–LAN, Ethernet Cross-Cable.

3.1.3 Software Interfaces

3.1.3.1 The theater ticket system will communicate with the configurator to identify all the available components to configure the product.
3.1.3.2 The theater ticket shall communicate with the content manager to get the product specifications, offerings, and promotions.
3.1.3.3 The theater ticket system shall communicate with the billPay system to identify available payment methods, validate the payments, and process payment.
3.1.3.4 The theater ticket system shall communicate to a credit management system for handling financing options.
3.1.3.5 The theater ticket system shall communicate with the external Tax system to calculate tax.

3.1.3.6 The theater ticket system shall be verisign like software which shall allow the users to complete secured transactions. This usually shall be the third party software system that is widely used for internet transactions.
3.1.3.7 The theater ticket system shall communicate with the Sales system for order management.

3.1.4 Communications Interfaces

The system shall use HTTP protocol for communication over the internet and for the intranet communication will be through TCP/IP protocol suite.

3.2 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

3.2.1 <Allows high website traffic>

3.2.1.1 The system should be fully functional no matter how many people are using the system.

3.2.2 <Interface with database>

3.2.2.1 The system should have access to the movie theater database of showtimes, tickets available, and ticket prices.
3.2.2.2 The system should be able to alter databases as each ticket is bought.
3.2.2.3 The system should be able to alter prices based on the ticket type purchased.

3.2.3 <Supports administrative control>

3.2.3.1 The system should allow employees to have separate access from users.

3.2.4 <Provide customer support>

3.2.4.1 System should have 24/7 support for the customers. (Ex: a phone number to call/text their questions, or an email to ask questions)

3.2.5 <Supports discounted tickets>

3.2.5.1 The system should allow users to purchase adult tickets.
3.2.5.2 The system should allow users to purchase child tickets.
3.2.5.3 The system should allow users to purchase senior tickets.

3.2.6 <Maintain customer profile>

3.2.6.1 The system should allow users to create a profile and establish login credentials.
3.2.6.2 The system should allow users to update credentials and profile information.

3.2.7 <Email confirmation>

3.2.7.1 The System should send an email to the user confirming their purchase right after payment and information has been verified.
3.2.7.2 Email should provide the tickets bought/how many, the seat number(s), what time the movie begins, and their form of payment.

3.2.8 <Provide shopping cart facility>

3.2.8.1 The system should provide a shopping cart during online purchases.
3.2.8.2 The system should allow users to add and remove items from the shopping cart.

3.2.9 <Provide online tax calculations>

3.2.9.1 The system should calculate the tax on the purchase.
3.2.9.2 The system should provide the price after the tax has been calculated.

3.2.10 <Allow multiple payment methods>

3.2.10.1 The system should allow payment through credit and debit cards.
3.2.10.2 The system should allow payment through Apple Pay.

3.2.11 <Allow cancellation of order>

3.2.11.1 The System should allow cancellation up to 2 hours before the movie begins.
3.2.11.2 The system will let the user know the refund will take two business days to fully process and return to their bank account.

3.3 Use Cases

3.3.1 Use Case #1

Name of case: Create An Account
Actor(s): TicketBuyer
Flow of events:

1) TicketBuyer goes to the website's home page.
2) TicketBuyer navigates to “Join Now” on the home page.
3) TicketBuyer enters name, email, birthday, and password.
    - The email acts as the username for future logging in.
    - The password must be a minimum of 8 characters and must contain at least two of the following: one uppercase letter, one number, or one special character.
4) TicketBuyer must tick a box to agree to the terms and conditions.
5) TicketBuyer is taken to a confirmation screen where they have the option to view their account or return to the home page.

3.3.2 Use Case #2

Name of case: Select Movie Time
Actor(s): TicketBuyer

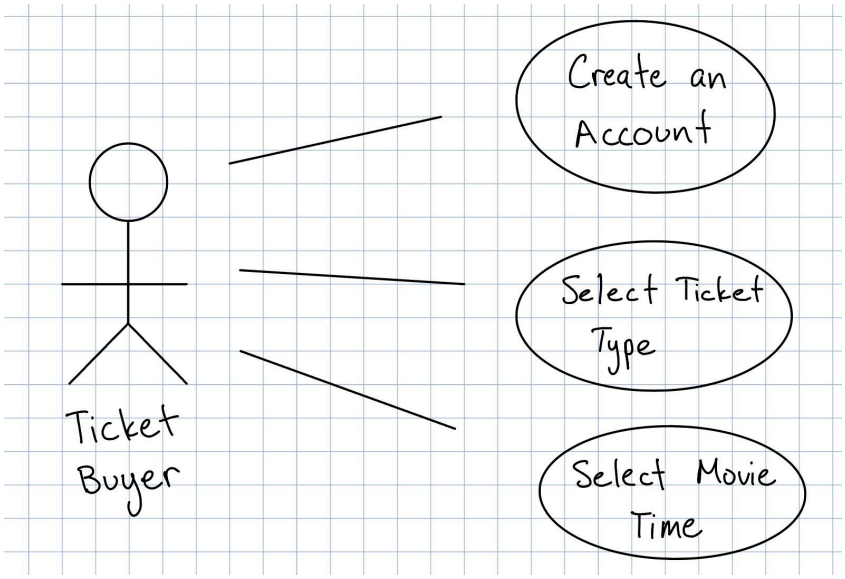
Flow of events:

1) After selecting the desired movie from the movie catalog, TicketBuyer will see a list of the different locations of theaters that are showing that movie as well as several movie showing times underneath each location.
2) TicketBuyer will scroll through the list and click on the time they wish to watch the movie.
3) TicketBuyer is directed to a screen where they can then select which seats they wish to be in.

3.3.3 Use Case #3

Name of case: Select Ticket Type
Actor(s): TicketBuyer
Flow of events:

1) After the location of the seat(s) is chosen, TicketBuyer is taken to a screen where the different types of tickets are listed: Adult, Child, and Senior. Next to the names of the types of tickets are the corresponding prices and the current quantity of that type of ticket. On either side of the quantity are "+" and "-" buttons TicketBuyer can click on to increase or decrease the number of tickets.
2) TicketBuyer will use the "+" and "-" buttons to choose how many of each kind of ticket they want to purchase. Based on how many seats were chosen on the previous page, TicketBuyer will only be allowed to select that many tickets.
3) Once all tickets have been selected, TicketBuyer may continue to the next page.



3.4 Classes / Objects

3.4.1 <Ticket Type Class>

3.4.1.1 Attributes

The class will have the attributes adult, child, and senior ticket type.
3.4.1.2 Functions
<Interface with database> The system will access the database and determine the pricing of the ticket based on the attributes chosen.

3.4.2 <Movie Class>

3.4.2.1 Attributes
The class will contain every movie available for watching in the catalog as well as the date, time, rating, which theater number (location) and number of available seats in the movie.
3.4.2.2 Functions
<Interface with database> The system will access the database and show the user different showtimes, seating options, and locations.

3.5 Non-Functional Requirements

Non-functional requirements may exist for the following attributes. Often these requirements must be achieved at a system-wide level rather than at a unit level. State the requirements in the following sections in measurable terms (e.g., 95% of transaction shall be processed in less than a second, system downtime may not exceed 1 minute per day, > 30 day MTBF value, etc).

3.5.1 Performance

3.5.1.1 The performance of the system's downtime cannot exceed 1 minute per day.
3.5.1.2 The performance system always needs to send a confirmation email immediately.
3.5.1.3 The performance system will process 95% of transactions in less than 5 seconds.
3.5.1.4 The performance redirection of the system should not exceed 10 seconds.

3.5.2 Reliability

3.5.2.1 The system's reliability will make sure it does not crash if too many people are on the website.
3.5.2.1 The system's reliability will ensure if two people click on the same seat, the first person actually gets it.

3.5.3 Availability

3.5.3.1 The system needs to have a good internet connection.
3.5.3.2 The system ensures you are able to go back to previous stages of ticket selection and checkpoints.

3.5.4 Security

3.5.4.1 The security system will ensure the payment will be secure.
3.5.4.2 The security system will ensure payment information stored for users who have accounts is safe and secured.
3.5.4.3 The security system will suggest strong passwords if creating an account.

3.5.5 Maintainability

3.5.5.1 The maintainability of the system will have frequent updates to keep it up to date.
3.5.5.2 The maintainability of the system will make sure movies and showing times are up to date.

3.5.6 Portability

3.5.6.1 The portability of the system will allow it to be used on different devices.
3.5.6.2 The system portability of the format will be adaptable.
3.5.6.3 The system can be used by different operating systems.

3.6 Inverse Requirements

3.6.1 The system shall not allow users to purchase more than 15 tickets at a time.
3.6.2 The system shall not allow users to make more than one account with the same email.
3.6.3 System maintenance and updates shall not occur during business hours and should not take longer than three hours.

3.7 Design Constraints

Specify design constraints imposed by other standards, company policies, hardware limitations, etc. that will impact this software project.

Design constraints will refer to the specific limitations and certain conditions that will impact the design and development of the software system. The system constraints by other standards are that they will be compliant with industry and regulatory guidelines. The system constraints by company policies is that they will adhere to the policies of security and privacy. Hardware limitations will include the capability with hardware specifications and limitations. Security requirements will include compliance with security policies and encryption standards. Resource limitations include budget and project timelines.

3.8 Logical Database Requirements

Will a database be used? If so, what logical requirements exist for data formats, storage capabilities, data retention, data integrity, etc.

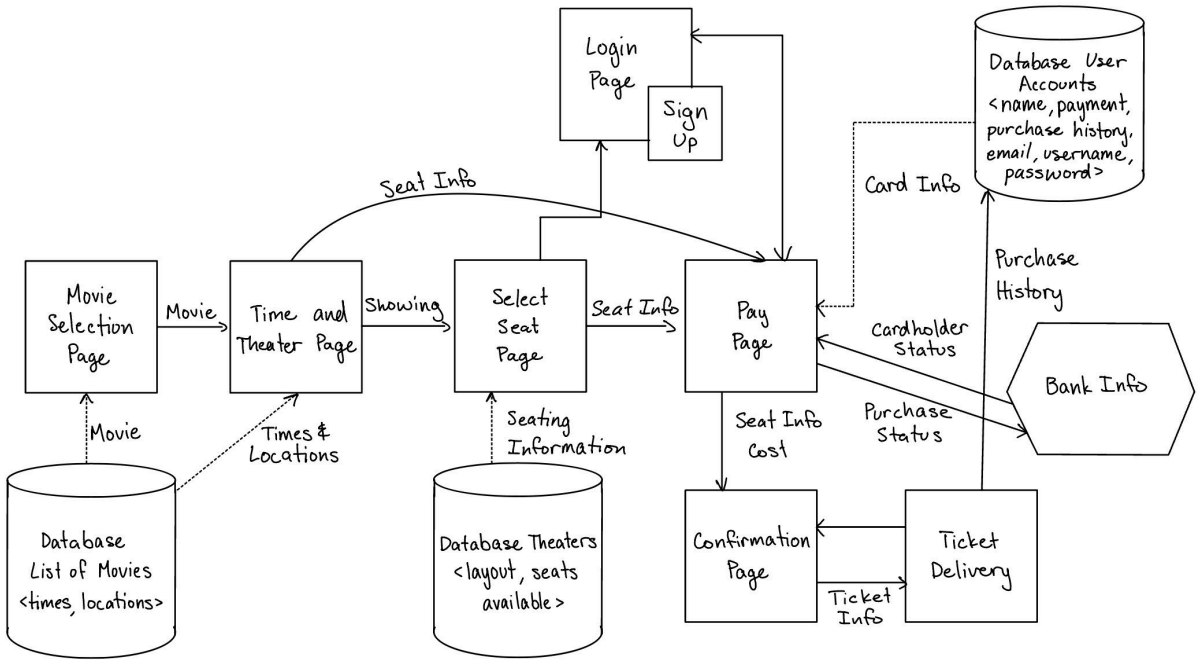
3.9 Other Requirements

Catchall section for any additional requirements.

4. Analysis Models

List all analysis models used in developing specific requirements previously given in this SRS. Each model should include an introduction and a narrative description. Furthermore, each model should be traceable to the SRS's requirements.

4.1 Data Flow Diagrams (DFD)



4.1.1<Login Page>

4.1.1.1 Attributes: This class will provide the option to enter their username and password. It allows the system to cross check if they are already a member with us and pull up their information.
4.1.1.2 Functions: The system will allow the user to create a login for first time guests, or allow guests who already have an account to sign in.

4.2.1< Database List Of Movies >

4.2.1.1 Attributes: This class will provide a list of movies, their times and locations
4.2.1.2 Functions: The system will allow you to go through each page of “Movie Selection Page” and “Time and Theatre Page” and have you make selections allowing you to move on as all selections possible are made.

4.3.1< Database Theatres>

4.3.1.1 Attributes: This class will allow the user to view the layout of the movie theater and the seats available.
4.3.1.2 Functions: The system will allow the user to select the available seats they wish. While allowing the user to have an idea of how the layout of the theater will be.

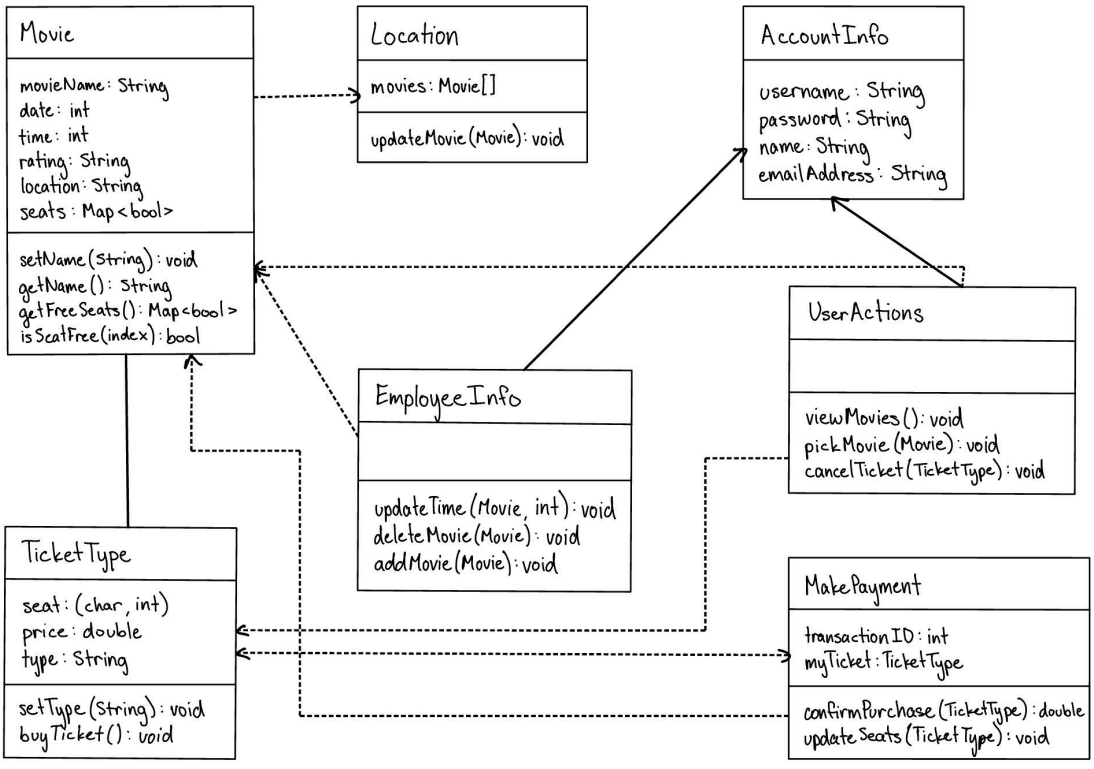
4.4.1<Database User Accounts>

4.4.1.1 Attributes: This class will allow the user to access and save their name, payment method, purchase history, email, username and password into our system. It also allows the system to access the information for a quicker and easier way to checkout.

4.4.1.2 Functions: The system will allow the user to save payment methods, usernames and passwords. Allowing the user to have a quick and easy login in and checking out.

4.5.1<Movie Selection Page>
4.5.1.1 Functions: The system allows guests to select a movie and then move on to the next page
4.6.1<Time and Theatre Page>
4.6.1.1 Functions: The System allows guests to see movie times and locations and prompt them to choose which one bests fits what they are looking for
4.7.1<Select Seat Page>
4.7.1.1 Functions: The system allows guests to select their seat and provide them with seat information
4.8.1<Pay Page>
4.8.1.1 Functions: The system will access both your card and bank information to confirm the purchase can be made
4.9.1<Confirmation Page>
4.9.1.1 Functions: The system will prompt a confirmation page that tells guests the seats they chose, the amount they paid and letting them know they will receive a confirmation email as well. It will also send them their tickets via the email which will be known to the guest.

4.2 UML Diagram



4.2.1 <Movie Class>
4.2.1.1 Attributes

The class will contain every movie available for watching in the catalog as well as the date, time, rating, which theater number (location) and number of available seats in the movie.
4.2.1.2 Functions
<Interface with database> The system will access the database and show the user different showtimes, seating options, and locations.

4.2.2 <TicketType Class>
4.2.2.1 Attributes
The class will have the attribute seat that has a character and int to identify the seat location in the theater, another attribute that holds the price of the ticket as a double and a final attribute called type that holds a string to identify whether the ticket is for a child, senior adult.
4.2.2.2 Functions
<Interface with database> The system will access the database and determine the pricing of the ticket based on the attributes chosen. It will also have functions setType and buyTicket.

4.2.3 <Location Class>
4.2.3.1 Attributes
The location class has an attribute called movies that creates an instance of the Movie class.
4.2.3.2 Functions
This class has a function named updateMovie that has the ability to update the location of the theater that the movie will be showing in.

4.2.4 <UserActions Class>
4.2.4.1 Functions
This class will have functions viewMovies, pickMovie and cancelTicket that use objects of the Movie and TicketType class.

4.2.5 <MakePayment Class>
4.2.5.1 Attributes
Attributes in this class include transactionID that holds an int for the user's order number and myTicket that creates an instance of the TicketType class.
4.2.5.2 Functions
This class will have a confirmPurchase function that returns the final price of the ticket plus tax as a double as well as an updateSeats function that allows the user to change the location of their seats if they wish to.

4.2.6 <AccountInfo Class>
4.2.6.1 Attributes
The user's username, password, name and email are all stored as strings.

4.2.7 <EmployeeInfo Class>
4.2.7.1 Functions
This class has functions updateTime, deleteMovie, and addMovie which allow an employee to make these changes from the administrative side of the system should they need to.

4.3 Description, Development Plan, and Timeline

Our software system will implement all previously mentioned classes and their attributes & functions in section 4.2. Overall, tasks to be completed include creating all code to implement the classes, debugging, and several rounds of testing. The first stage of creation of the code should not take more than a month considering that it is executed thoroughly and for the most part, correctly. The debugging stage should not take any longer than two weeks and the testing stage should not take longer than two weeks. As for task assignment, it is ideal to have at least 10 people involved in each stage of the development process. It is best if no individual is involved in more than one of these stages so they may direct their attention to only one task.

5. Test Plan

TEST CASE SAMPLES: TestCases

5.1 <availableSeats>

In this portion of testing, the established tester named Tester will determine if the system correctly displays how many seats are available in a specific theater at a specific time for a specific movie. In addition, the Tester will also determine if the system accurately shows seat rows and numbers. This is assuming that the web browser can be launched, the Tester was able to log in to their established account, and the steps of choosing a movie from the movie catalog as well as selecting an available time of day to watch the movie is functioning correctly. The steps that the tester will take for this test case is first searching our URL (pljtheaters.com) and loading the website. The tester will then log into their established account. The tester will then select a movie, location, and time that they know still has available seats. Once the tester gets to the seat selection page, they will try to pick seats in that theater. A passing test will show an updated organization of “seat” icons that is in accordance with the actual layout of the seats in the selected theater, with accurate seat and row numbers. When Tester’s mouse hovers over an available seat icon, colored white, they are able to click on it. However, a seat that has already been purchased should be gray and when Tester’s mouse hovers over it, they should not be able to click on it. If Tester is able to click on a gray seat icon and proceed with the ticket purchasing process, the test fails.

5.2 <noAvailableSeats>

For this test case, Tester will determine if the system correctly displays that no seats are available in a specific theater for a specific movie. In addition, the Tester also needs to determine that the system won’t let the user click on any seats that are no longer available. This is assuming that the web browser can be launched, the Tester is able to log in to their established account, a specific movie can be selected, and a specific time and location can be selected. The steps that the tester will take for this test case is first searching our URL (pljtheaters.com) and loading the website. The tester will then log into their established account. The tester will then select a movie, location, and time that they know no longer has any available seats. Once the tester gets to the seat selection page, they will try to pick seats in that theater. A passing test will show that all the seats are colored gray (indicating that they aren’t available), and the tester will not be able to

select any of the seats in the theater. However, it is a failing test if some of the seats are still colored white (indicating availability) and/or the tester can still select seats that are colored gray.

5.3 <emailConfirmation>

For this test case, Tester will determine if a confirmation email is sent in a timely manner and contains all relevant information regarding their purchase. In addition, the Tester will determine if the confirmation email includes a link to cancel the tickets. This is assuming that the web browser can be launched, Tester was able to log in to their established account, a specific movie can be selected, a specific time and location can be selected, seat(s) can be selected, ticket type can be selected, and payment went through successfully. The steps that the tester will take for this test case is first searching our URL (pljtheaters.com) and loading the website. The tester will then log into their established account. The tester will then select a movie, location, time, and seats they wish to purchase as well as what kind of ticket they wish to purchase. Tester will then enter their credit card, debit card, or ApplePay information and complete the purchase. The confirmation email will be sent to the email address associated with their account. A passing test requires the confirmation email to be sent to the correct email address in under 5 minutes and contains all relevant information about their purchase. This includes what movie they chose, time, date, theater location, seat location (row letter and seat number) and a link that allows them to cancel their order.

5.4 <movieTimes>

In this test case, Tester will be looking out for an accurate selection of movie times for the movie they wish to watch. This is assuming the web browser can be launched, the Tester can log into their established account, and Tester is able to choose a movie from the movie catalog. If user location is granted, it is assumed that the selection of theaters under which the movie times are listed is also accurate (explained more in 5.5). The steps that the tester will take for this test case is first searching our URL (pljtheaters.com) and loading the website. The Tester will then log into their established account. The tester will then select a movie and will be led to a list of available locations and times. A passing test will show correct movie times for each location and no repeated times. The test will fail if there are repetitive movie times, missing movie times, and/or an incorrect display of options for movie times.

5.5 <movieLocation>

For this test case, Tester will determine if the system correctly displays the movie theater locations that are showing a specific movie. In addition, the Tester will also determine if the system correctly displays theater locations that are within a 25 mile radius of the user (only applicable when location settings are turned on). This is assuming that the web browser can be launched and a specific movie can be selected. The steps that the tester will take for this test case is first searching our URL (pljtheaters.com) and loading the website. The tester will then select a movie. Once the tester is brought to the time and location selection page they will verify the theater locations that are presented on the website. A passing test will show every single movie theater within a 25 mile radius that is showing the specific movie that was selected. A failing test will either show theaters that aren't showing the selected movie, will be missing theaters that are showing the selected movie, or will include theaters that are farther than 25 miles away from the user.

5.6 <payment>
For this test case, Tester will determine if the system can receive payments from only credit cards, debit cards, and apple pay in a timely manner. This is assuming that the web browser can be launched, a specific movie can be selected, a specific time and location can be selected, seats can be selected, and the ticket type can be selected. The steps that the tester will take for this test case is first searching our URL (pljtheaters.com) and loading the website. The tester will then select a movie, location, and time that they know no longer has any available seats. Then the tester will select any available seats for that movie, location, and time. Once seats have been selected the tester will then select a ticket type and then be brought to the payment screen. At this screen the tester will then input payment information and verify that it loaded within 5 seconds. The tester will do this a total of four times, each time trying a different payment method (credit, debit, apple pay, and an expired card). A passing test will show that the payment methods credit, debit, and apple pay are all accepted and are loaded within 5 seconds of the purchase. A failing test will show that either the credit, debit, or apple pay were not accepted, the expired card was accepted, or the payment took longer to load than 5 seconds.

5.7 <password>
For this test case, the tester will verify the password to be a minimum of 8 characters and must contain at least two of the following: one uppercase letter, one number, or one special character. Failure to complete the request will not allow the user to create the password and will be encouraged to change or add to make it valid. This is assuming that the web browser can be launched. The steps that the tester will take for this test case is first searching our URL (pljtheaters.com) and loading the website. They will then navigate to the “Sign In” button in the upper right corner and choose the option to make an account. A passing test will not let Tester make a password that does not follow the established guidelines and allows the creation of an account with a valid password. A failing test will allow Tester to create a password that does not follow the established guidelines i.e. a password that has all capital letters.

5.8 <username>
For this test case, the tester will use the email to act as the username for future logging in. Which will then be used to identify the user and recognize them for future login ins. Tester will make sure the username is valid and not been used before in order to allow it to become the username. Assuming the tester is already on the web browser and is ready to select and purchase tickets. The system will prompt the tester to make an account, allowing them to type in their email, letting the tester know this will count as the username as well. Checking the email if it is valid, the email has become the username and test case should have passed. A failed test case will allow the tester to create two accounts under the same email account.

5.9 <cancelTicket>
For this test case, the tester will confirm that cancellations are allowed, works and will be processed fully in two business days. A refund will appear in their bank account in those business days. The first step would be assuming the tester has already selected and paid for the movie tickets before going ahead to cancel. In the confirmation email sent there will be a link to

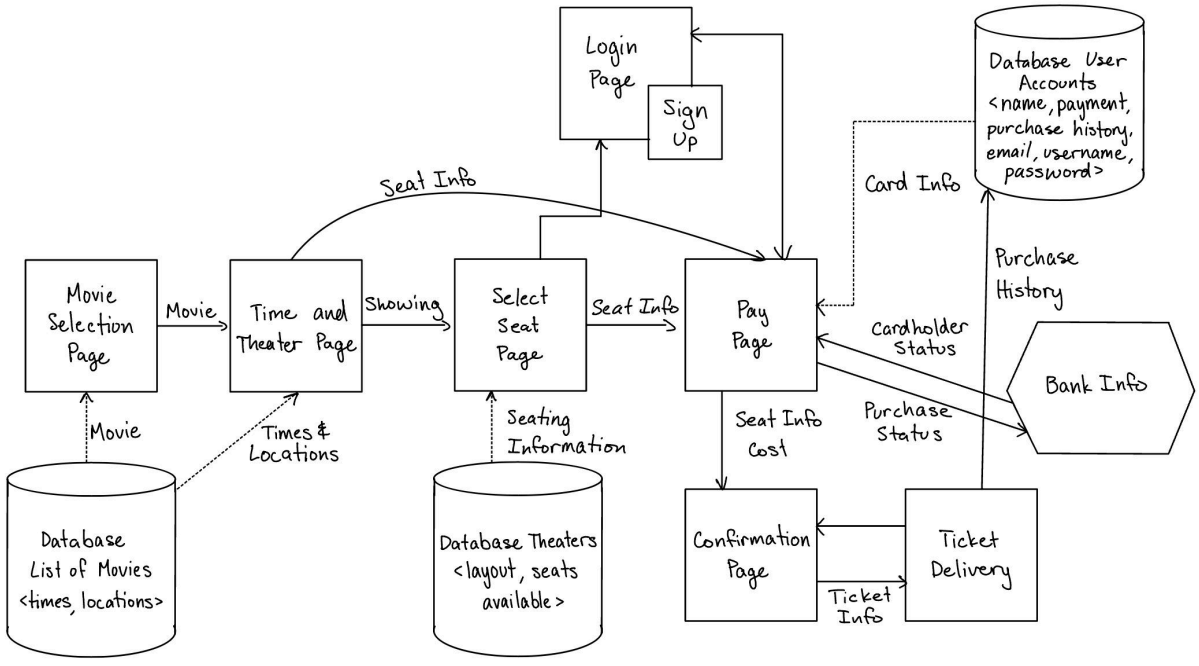
cancel and receive a refund for the tickets purchased. Tester will make sure the cancellation is two hours prior to the movie start time, once confirming it is two hours prior, cancellation will go through. Confirming the tickets once again and the movie for the one being canceled for. Tester shall be notified the refund went through and be known when the refund will be fully processed by. A failed test case will allow cancellations after the two hour window.

5.10 <searchBar>

For this test case, the tester will search specific words to allow the system to load anything that is related to what is searched up. Test case will pass if search results will be related to what is being searched up. This is assuming that the web browser can be launched and the Tester can log into their established account. The steps that the tester will take for this test case is first searching our URL (pljtheaters.com) and loading the website. Then the Tester will log into their established account. Tester will then go up to the search bar. Tester searches for a movie title and the system should show when that movie is playing and which theater it is playing at. If their designated movie location is not showing that movie, then the next closest location will show, allowing the tester to view and do as they please. Test case will pass if all related content to what the tester searched up gets displayed. A failed test will show Tester no movies or unrelated movies after their search.

6. Architecture Design with Data Management

6.1 Software Architecture Diagram



6.2 Data Management Strategy

We will be using SQL for our data management strategy as we have three databases part of our system. There is one database for the catalog of movies available for users to watch, another for the theater layouts (formatting of seats), and the last database for user accounts and the corresponding information i.e. name, email, username, password. The information that was needed for our system could be grouped together and so it made sense for us to combine it all into their own databases. Because each database has its own attributes and relationships, SQL also seemed to be the best choice to organize our data. As an alternative, the movie catalog and theater databases could be combined into one, but we believe keeping them separate is best because the data in the movie catalog database cannot be altered, but the data in the theater database (such as seat availability) can change. We could have also added the user's bank information into the user account database, however we chose not to as this would be a highly potential security risk.

7. 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