RJJ’s Theater Movie Ticketing System

Software Requirements Specification

Version 4

4/10/2025

Group 6

Ross Hacket, John Ton, Jared Williams

Github: https://github.com/johntton/Group-6-Software-Specification-Requirements

Prepared for
CS 250- Introduction to Software Systems
Instructor: Gus Hanna, Ph.D.
Spring 2025

Revision History

Date	Description	Author	Comments
2/20/25	Version 1	Ross Hackett, John Ton, Jared Williams	Completed Introduction, General Description, External Interface Requirements, Functional Requirements, Use Cases, Classes/Objects, Non-Functional Requirements, and Inverse Requirement
3/6/25	Version 2	Ross Hackett, John Ton, Jared Williams	Analysis models: Sequence Diagrams, Data Flow Diagrams, and State Transition Diagrams
3/20/25	Version 3	Ross Hackett, John Ton, Jared Williams	Test Plan, Introduction, Test Details. Added Failure Handling to 4.3
4/10/25	Version 4	Ross Hackett, John Ton, Jared Williams	Data Management Strategy, Trade-offs, Data Organization, Data Design Diagrams

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 entire SRS with purpose, scope, definitions, acronyms, abbreviations, references, and overview of the SRS. The aim of this document is to gather and give an in-depth insight of the complete **RJJ Theaters Movie Ticketing System** by defining the problem statement in detail. It also concentrates on the capabilities required by stakeholders and their needs while defining high-level product features. The detailed requirements of **RJJ Theaters Movie Ticketing System** are provided in this document.

1.1 Purpose

The purpose of this document is to provide software engineers and web developers with an adequate description and understanding of the design structure and features intended for the online movie ticketing system that will be used by RJJ Theaters and describe the intended user experience.

1.2 Scope

The intended product will allow users to purchase and receive tickets electronically, notable features include, but are not limited to

1.) Searching for all movies playing at a specific theater, all showings of a certain movie in a given area and filtering movies recommended on the homepage by user rating and age rating.
2.) Purchasing tickets online and selecting preferred seating on the website
3.) Receiving tickets/receipts via email or text

Users will be able to pay for tickets with credit or debit card and save the payment info and preferred receipt delivery method if they choose to make an account.

1.3 Definitions, Acronyms, and Abbreviations

RJJ stands for the names of the product's creators: Ross, John, and Jared. IEEE stands for Institute of Electrical and Electronics Engineers. API stands for Application Programming Interface. APIs are used to communicate between different applications. SMS stands for Short Message Service. SMS is a text-messaging service used for communication through telephone numbers. NFT stands for non-fungible tokens. NFTs are a form of currency that is not compatible with this product. The QR in QR code stands for quick response. QR codes are an image composed of black and white pixels used to store digital data.

1.4 References

1) IEEE Recommended Practice for Software Requirements Specifications by IEEE Computer Society; Published 20 October 1998
2) Software Requirements Specification document with example by Ravi Bandakkanavar; Published 8 May 2023

1.5 Overview

The remaining sections of this document provide a general description, including characteristics of the users of this project, the product's hardware, and the functional and data requirements of the product. General description of the project is discussed in section 2 of this document. Section 3 gives the functional requirements, data requirements, and constraints and assumptions made while designing the ticketing system. It also gives the user viewpoint of the product. Section 3 also gives the specific requirements of the product. Section 3 also discusses the external interface requirements and gives detailed description of functional requirements. Section 4 is for diagrams and Section 5 is for any change that may occur.

2. General Description

The product is a ticket processing system for movie theaters, incorporated through a website. The ticket purchasing process will directly interact with each theater database, which includes the movie list, seating arrangement, and pricing.

2.1 Product Perspective

The RJJ Theater Movie Ticketing System was developed to provide a seamless experience for users when purchasing movie tickets. Further, the system will help in the daily functions of RJJ Theater with tracking ticket sales, seat availability, etc. The system is independent and self-contained, as we only have theaters in San Diego.

This product will work in tandem with a credit/debit card verification system. Further, it will be integrated with the Apple Pay API and Google Pay API, allowing for flexibility in payment. Once the ticket is purchased, the user will be sent an SMS message for verification and a QR code to their email. Captcha will be integrated into the account system for ensuring security against bots. This product will have direct access to the theater database for instant updating of the database.

2.2 Product Functions

Although a log-in system will be implemented for repeat customers, newcomers are allowed to purchase tickets through a guest account. The account system will prevent bots from creating an account. Users with an account will be able to purchase tickets and reserve seats through the website. Tickets will be received through the user's phone or email. The users will be able to discover popular, new movies via the homepage or select specific films or theatres and conveniently filter results by time and location. Customer support will be accessible through the website as well, allowing users to discuss and troubleshoot possible problems with a chatbot before being redirected to a customer service representative in the result that their problem is not resolved by this.

2.3 User Characteristics

The users of this product are of the general public, in particular movie theater customers. The user is expected to know the basics of navigating a website. The user must have either a credit/debit card, Apple Pay, or Google Pay to purchase a ticket and must have an email or phone number to receive the ticket. Security features will prohibit access from bots.

2.4 General Constraints

As this product is meant for exclusive use of the theaters connected with this system, it will not be able to communicate with external databases, limiting scaling with different theaters. This product will not accept other forms of payment like cryptocurrency or NFT, and will only process credit/debit cards, Google Pay, and Apple Pay. The maximum number of tickets able to be purchased at once is 20 tickets. Theater room rentals are not allowed through this product and inquiries must be through our customer support system.

2.5 Assumptions and Dependencies

The website will be made for access from computers, laptops, tablets, and smartphones. The recommended browser is Chrome, but will be compatible with Microsoft Edge, Safari, and Firefox. The website will be functional on all operating systems, including Android, IOS, macOS, and Linux on their devices.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

RJJ Theater Movie Ticketing system will be a website that users are able to access through their web browser. The homepage will display new and upcoming movies with several tabs that can relocate them to other pages such as account info, movie search or theater search. Films on the homepage will be interactable and take the user to their own specific page featuring info on the film. The seating selection page will display taken and available seats separately. The payment page can be accessed after confirming seat selection. The payment page will include the number of tickets sold, total price of all tickets, and payment options, which will allow users to enter credit, debit or use Apple pay or Google pay.

3.1.2 Hardware Interfaces

If accessing through a computer, the user will require a keyboard and mouse to navigate through the website. The user will be able to utilize the search function through their keyboard. The scrolling function will be implemented to ensure seamless navigation when using a mobile device. Through these functions, the user will be able to navigate through the website and pick their desired movie(s).

3.1.3 Software Interfaces

The website will utilize the Network Time Protocol in order to accurately display the time and date to the user. With this, the website will be able to accurately display which movies are showing and which movies have passed. Further, the website will utilize Simple Mail Transfer Protocol in order to send the users their QR code. The system will send the confirmation text through the SMS API. The system will connect with the Apple Pay API and Google Pay API to process payments if the user chooses not to pay with their credit/debit card.

3.1.4 Communications Interfaces

The system will integrate with the SMS API and Simple Mail Transfer Protocol to communicate with the user, whether it's for ticket confirmation or cancellation. Communicate with the user's system in order to access time and date. Ensure that the system has a stable and safe connection between the user and server sides.

3.2 Functional Requirements

3.2.1 <Account System>

3.2.1.1 Introduction
The user must create an account before purchasing a ticket. The user may create a guest or member account, however, the user cannot purchase a ticket without an account.

3.2.1.2 Inputs
For the guest account, the user can only provide a phone number. For the member account, a phone number is optional and the user may provide an email address, username, and password.

3.2.1.3 Processing
The phone number will be verified through an SMS text message. The email address will be verified by sending an email with a website link. A new member account may not use an already existing email address, phone number, or username in the database, but guest accounts will allow previously used phone numbers. The password will be processed securely and will not be stored in raw text.

3.2.1.4 Outputs
The output is the corresponding account information stored in the database. Data will be sent with encryption.

3.2.1.5 Error Handling
Duplicate email addresses, phone numbers, or usernames for new member accounts will be prohibited and users will be prompted to change their input. If all the necessary information is not received, an account will not be created and the user will have to restart the process. A "forgot my password" function will allow users to recover their account. A link will be sent to their email address.

3.2.2 <Payment Processing>

3.2.2.1 Introduction
When purchasing a ticket, the user will be prompted to enter a form of payment or select from a previously stored form of payment.

3.2.2.2 Inputs
The user will be prompted to select from using a credit/debit card, Apple Pay account, or Google Pay account. The user may choose to store their form of payment, which will be linked to their member account.

3.2.2.3 Processing
The payment processing system will not be native to this product and will instead use external verification systems.

3.2.2.4 Outputs
If the user chooses to store their form of payment, it will be stored in the database, but not in raw text. Data will be sent with encryption.

3.2.2.5 Error Handling
The payment processing system will reject invalid card information or nonfunctional Apple Pay/Google Pay accounts. The user will be prompted to try again. If there are insufficient funds, the tickets will not be provided.

3.2.3 <Customer Support>

3.2.3.1 Introduction
The user may get help through customer support. The user will first be directed to a chatbot before connecting to a customer support agent.

3.2.3.2 Inputs
The user will communicate with the chatbot through a messaging chat box in the website. The user will need to have a phone number to contact a customer support agent.

3.2.3.3 Processing
Customer support through the chatbot is intended to cover most general problems or inquiries. The chatbot will integrate artificial intelligence with natural language processing. If the chatbot sees that the user is not getting their questions answered, the user will be asked if they want to communicate with a customer support agent, which will happen through a phone call.

3.2.3.4 Outputs
The database will record both the user's chat logs with the chatbot and the phone calls with the customer support agent.

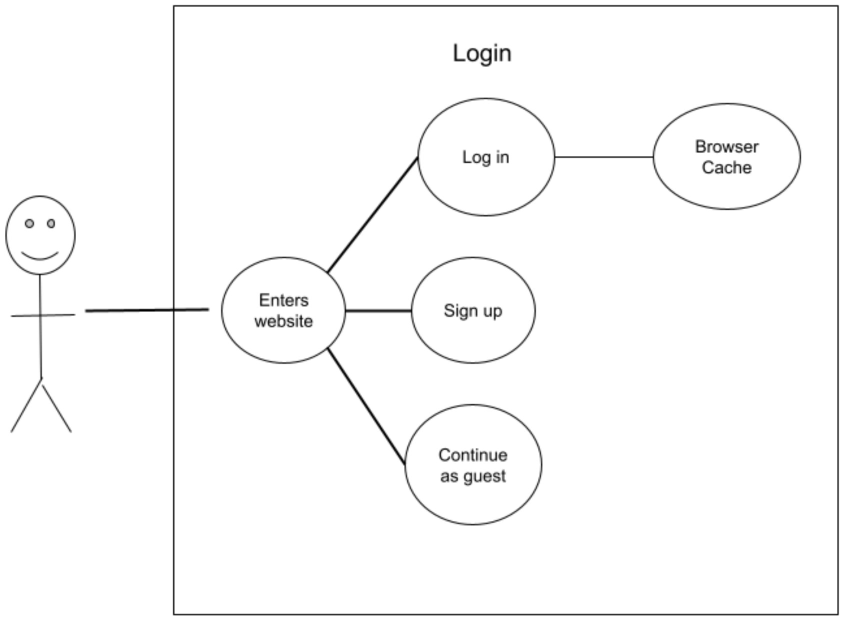
3.2.3.5 Error Handling
If a customer support agent is not available, the user will be placed on hold and be put in a queue until an agent is available.

3.3 Use Cases

3.3.1 Use Case #1: User Log-in

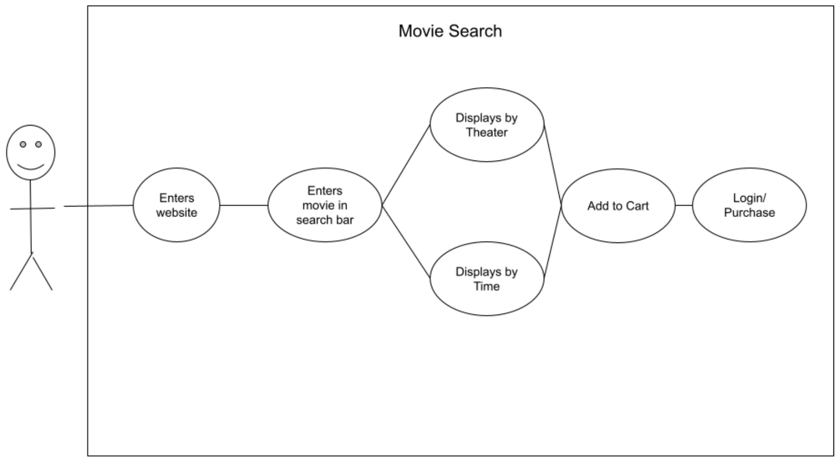
Actor: User
1. The user will add the ticket to their cart.
2. After going to their cart, the user will be prompted to login, sign up, or continue as a guest.

3. If they choose to login, they'll enter their username and password. If they choose to continue as a guest, then they'll have to input in their phone number. If they choose to sign up, they'll be prompted to create a username and password.
4. When logging in, the system will check the database to ensure that username and password were inputted correctly and there's a record of the account.
5. Once the user has successfully logged in, they'll be permitted to continue with their purchase and move on to the payment method.
6. Once payment is successful, the user can continue to shop or log out



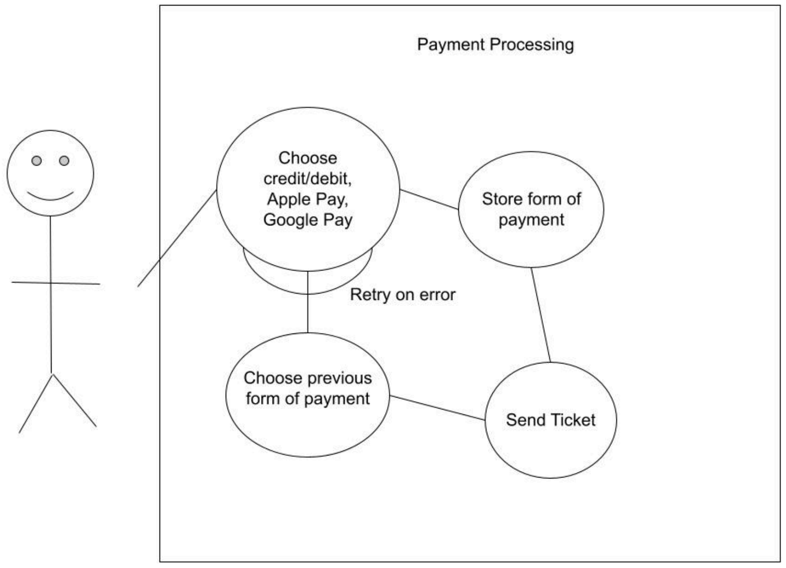
3.3.2 Use Case #2: Ticket Browsing/Searching

Actor: User
1. The user will access the movie theater website through their web browser
2. The search bar will be located at the top right corner of the tab. Here, the user is able to click on it and search up their desired movie.
3. Once the user puts in their movie, tabs will automatically populate below the search bar with recommendations similar to what the user inputted
4. The user can press enter or any of the recommended listings.
5. The user will be able to see all the times and locations that the movie will be playing at
6. The user can select a certain time and location and add it to their cart and allow the user to purchase the ticket.



3.3.3 Use Case #3: Payment Processing

Actor: User
1. The user will be prompted to select from using a credit/debit card, Apple Pay account, or Google Pay account.
2. If the user's account has a stored form of payment, it will be displayed as an option as well.
3. The user will be redirected to the corresponding payment processing system.
4. If an error occurs, the user will be prompted to try again
5. If the form of payment is verified, then the user may choose to store their new form of payment, which will be linked to their member account.
6. After the payment is processed, the ticket will be sent to the user.


Payment Processing
Choose credit/debit, Apple Pay, Google Pay
Store form of payment
Retry on error
Choose previous form of payment
Send Ticket

3.4 Classes / Objects

3.4.1 <Class / Object #1> User

3.4.1.1 Attributes

1. userName: user's username for logging in and identification
2. userPassword: user's password for logging in
3. userEmail: used for password retrieval and receipt delivery
4. userNumber: user's phone number for notifications receipt delivery and password recovery

3.4.1.2 Functions

1. login(userName, userPassword): allows the user to login to the website
2. selectSeat(showingID, seatID): allows the user to reserve a seat for 5 minutes while they enter payment info for purchase.
3. searchFilm(filmTitle): allows user to search for movies by title and displays most relevant results by distance or time
4. searchTheater(address): finds theaters in order of closest to furthest within a 20 mile radius of the entered address
5. purchaseTicket(showingID,seatID): initiates purchase process of a ticket for a selected seat.

3.4.2 <Class / Object #2> Ticket

3.4.2.1 Attributes

1. seatID: letter and number combination for seat location
2. showingID: contains information of the movie title, time the movie starts, and room number
3. price: price of the ticket
4. uniqueID: unique ID only for this ticket for identification

3.4.2.2 Functions

1. sendTicket(userEmail): sends email containing ticket to user's email

3.4.3 <Class / Object #3> Theater

3.4.3.1 Attributes

1. theaterName:name displayed for users upon searchTheater execution
2. theaterAddress: address of theater used to determine whether or not to show in search results for searchTheater and in what order according to proximity of the entered address
3. showings: list of all movies playing and their showtimes

3.4.3.2 Functions

1. getFilms(): displays all available films showing at this theater
2. getShowTimes(): displays all times films are being showed

3.5 Non-Functional Requirements

3.5.1 Performance

The response time for selecting movies, theaters, or displaying search results should be fast and responsive. Ticket purchase confirmations and seat reservation confirmations should all take less than ten seconds. The system should function identically under high and low user traffic.

3.5.2 Reliability

Website should have 99.5% uptime with 0.5% accounted for maintenance between 5am to midnight.

3.5.3 Availability

The website should be accessible at all hours of the day every day of the week not including scheduled maintenance.

3.5.4 Security

All sensitive customer information should be encrypted (password, username, payment method information, and other personal information)
Ensuring Payment Security: All payment processing must adhere to the Payment Card Industry Data Security Standard (PCI DSS) to safeguard customers' sensitive financial information. Compliance with these standards helps prevent fraud, unauthorized access, and data breaches.

3.5.5 Maintainability

Implement version control to facilitate backups in order to ensure recovery from unexpected errors. The system will be designed in a modular fashion in order to allow updates to individual components without affecting the entire system. Regular maintenance and updating of documentation for all system components will ensure consistency.

3.5.6 Portability

The system will be accessible through different browsers depending on what the user is running/using. This includes Chrome, Safari, and Firefox. Further, users are able to run the website on multiple operating systems, including Android, IOS, MacOS, and Linux. Users are able to log-in to their account on multiple devices, though not at the same time.

3.6 Inverse Requirements

Users are not able to log into their account on multiple devices at the same time. This will ensure website security and stability.
Users are not able to stay on the website for more than 60 minutes at a time. They will be automatically logged out of their account. This will also assist with security.
Users will not be able to double book a seat. If a seat has been taken by either the user or another user, they will not be able to book that same seat.
Users will not be able to book movies that have already been shown or past its showing window.
User’s sensitive information, such as their password and credit card information, will be displayed using “*” to maintain security.

3.7 Design Constraints

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

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.

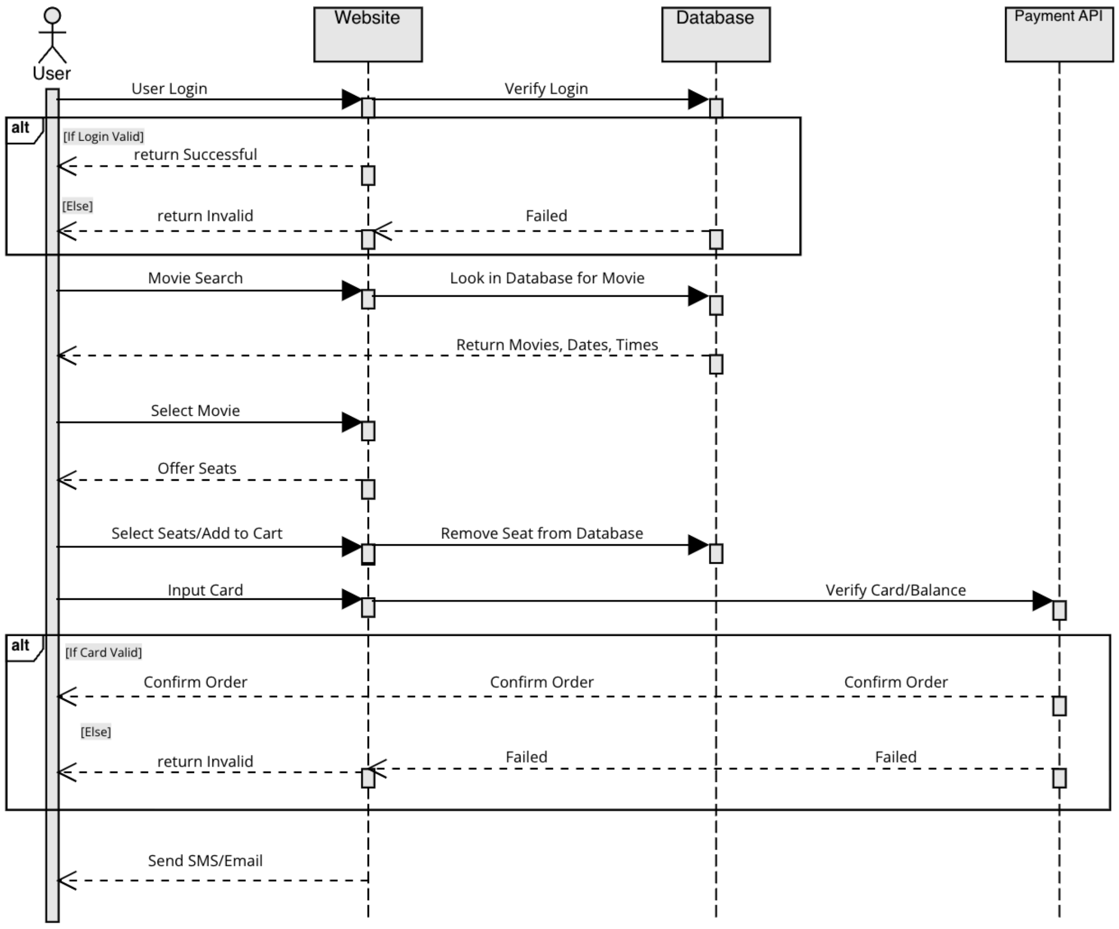
3.9 Other Requirements

Catchall section for any additional requirements.

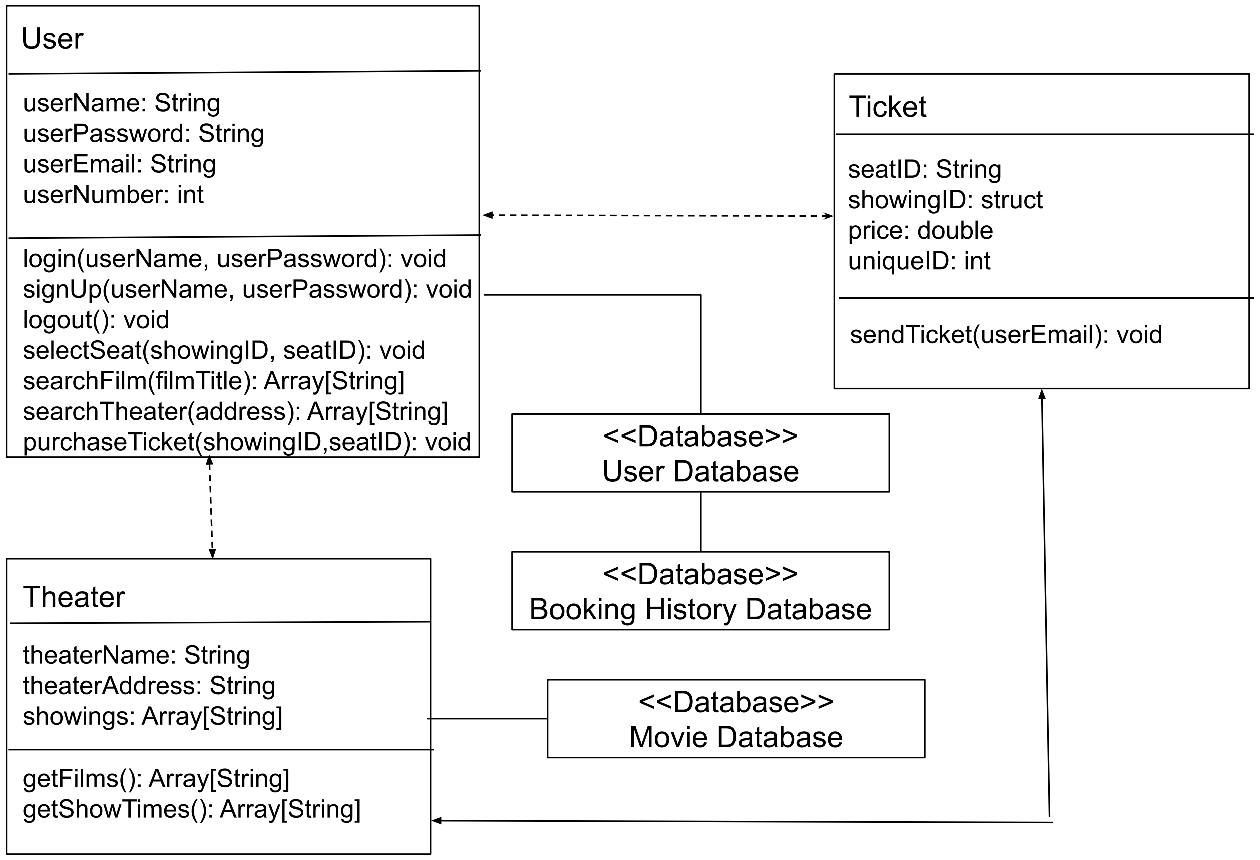
4. Analysis Models

This section presents a list of the fundamental sequence, data flow, and state-transition diagrams that satisfy the system's requirements. The purpose is to provide an alternative, "structural" view of the requirements stated above and how they might be satisfied in the system.

4.1 Sequence Diagrams



4.2 Unified Modeling Language Diagram (UML)

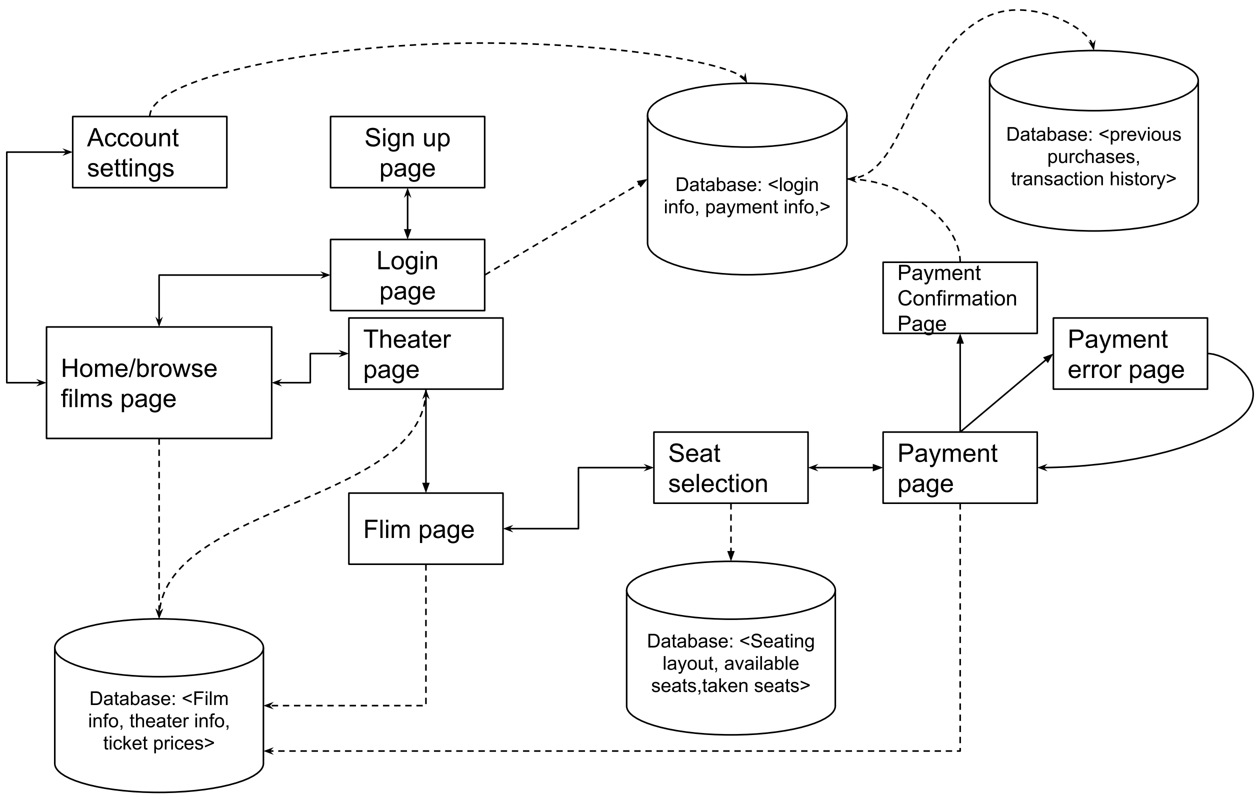


The **User class** contains the information pertaining to a member's personal information used when logging in, in particular the userName, userPassword, userEmail, and userNumber. All of the objects of this class are stored in the user database. The login() function allows the user to proceed with their member account. The User class interacts with the Ticket class with the selectSeat() and purchaseTicket() function using the corresponding parameters in the Ticket class. The User class interacts also with the Theater class with the searchFilm() and searchTheater() functions with the corresponding parameters in the Theater class.

The **Ticket class** contains the information used for each movie ticket including the seatID, showingID, price, and uniqueID. The struct for showingID contains a string for the movie title, a string formatted in hours and minutes for the time the movie starts, and an int for the room number. Its function sendTicket() uses the userEmail parameter from the User class.

The **Theater class** contains the identifying information and each theater's movie list in the parameters theaterName, theaterAddress, and showing. Each Theater object is stored in the theater database. The getFilms() and getShowTimes() functions receive the corresponding information from the database to be used for display in the website when calling the search functions in the User class.

4.3 Software Architecture Diagram(SWA)


Account settings
Sign up page
Database: <login info, payment info,>
Database: <previous purchases, transaction history>
Login page
Payment Confirmation Page
Home/browse films page
Theater page
Payment error page
Seat selection
Payment page
Flim page
Database: <Seating layout, available seats,taken seats>
Database: <Film info, theater info, ticket prices>

This diagram demonstrates the key components and their interaction with one another.

- **Account Settings**:
   - Users will be able to login and access account information via the account settings page.
- **Signup/Login page**:
   - Users will be able to create an account if they prefer to save payment data and view past purchase history or login if an account already exists.
   - Users' usernames, passwords, and purchase history are stored in the User Database
   - **Fail Handling**: If the user inputs an incorrect username or password, they aren't able to log in and are prompted to try again. The user will not be able to successfully log in if they're inputting incorrect information. Further, when the user is signing up, they will not be able to create an account if their username has already been taken. The phone number input will ignore characters that are not numbers, such as dashes or parentheses. The email input must have text followed by an @, a domain, and end with a top level domain. The password may contain any text, but must be at least 8 characters long.

-**Home page**:
   - Allows users to browse through popular movies showing at theatres nearby.
   - The home page is interacts with the signup/login page as users aren't able to access this page without a valid login
   - The home page is accesses the Theater Database in order to display a short list of movies
   - **Fail Handling**: If the page doesn't display all the movies in the database, the user should refresh the page to try to connect the system and the database. If it still presents as an issue, the

system will continue to connect the system and the database. If that still doesn't resolve the issue, the page will show an error message, to where a developer will try to resolve the issue.

-**Theater page**:

- Displays popular films and all showtimes for the selected theater.
- The theater page interacts with the Theater Database in order to display and search all the movies that are available
- The theater page allows the user to access the home page and film page
- **Fail Handling**: If the user does not input any text before pressing enter to search, the search will not go through. The letters of the search will not be case-sensitive.

-**Film page**:

- Displays relevant info about the selected film with the option to select seats.
- The film page interacts with the Theater Database
- The film page allows users to access the theater page and seat selection

-**Seat selection**:

- Interactive display that allows the user to reserve selected available seats.
- The seat selection page interacts with the Seat Database
- Allows users the interact with film page and payment page
- **Fail Handling**: The user is not allowed to reserve seats that were reserved in the past. These seats will be grayed out and not be able to be selected by the user.

-Payment Page:

- Allows the user to input their payment information.
- **Fail Handling**: Will redirect to the *Payment error page* if the payment information is not valid, prompting the user to try again. The system will try to communicate with the payment API in order to take the payment from the user. If this issue persists, the system will suggest that the user pick another form of payment. For example, if the system isn't taking Apple Pay for whatever reason, the system will suggest that the user either use Google Pay or Credit Card.

- **Payment confirmation page**

- Notifies the user of successful purchase and phone number or email address receipt has been sent to.
- **Fail Handling**: If the user doesn't receive the SMS message or Email of their order confirmation, the user is prompted with an option or resent those messages after 30 seconds. If the user still doesn't receive a message, the user is able to download a pdf of the QR code.

4.4 Test Plan

4.4.1 Test Plan Introduction

The test plan will verify and validate the components and methods of our Ticketing System. It will ensure that the functionalities that we've included in the system are working as expected and will for the user. The **scope** of the test plan will focus on the user login and signup, search system, ticket purchasing and reservation, payment processing, and ticket purchasing process. The tests will be of several types, including **unit tests, functional tests, and system tests**. All these tests are of different granularities.

Unit tests on testing individual components or functions of the software in isolation to ensure they work as expected.

Functional tests verify that the software performs its intended functions correctly, often based on requirements or specifications.
System tests involve testing the entire system as a whole to ensure all components interact correctly and meet overall system requirements.

4.4.2 Test Details

Unit Tests:

- Movie_Search_1:
    - Test the Search Bar Functionality
    - The test will test the getFilms() and getTimes() functions in our Theater Class. When the user searches a movie, those two functions are called in order to retrieve dates, times, and movies. Tests are conducted to ensure those functions are working properly and retrieve accurate movies, dates, and times. We are trying to discover any bugs present and any discrepancies between the functions and the database.
    - Test Steps:
        - Launch website from user's internet browser
        - Search for a movie in the search bar
        - Press enter
        - Compare movie dates/times to the dates/times in the database and verify for accuracy
    - Expected Outcome: The function should present the user with accurate movie dates/times depending on the movie that they choose. If no movies are available during the time, an error message will appear stating that there aren't any movies, times, or dates available at the time.
- User_Login_1:
    - Tests the Login Functionality
    - This will test the login() function in the User Class. If the user has inputted the correct username and password, they will be able to log into their account and access all their information. This test is trying to find a bug where the user is still able to login despite inputting the wrong username or password.
    - Test Steps:
        - Launch movie theater website from web browser
        - Press "Log in account"
        - Enter username and password
        - Press "Log in"
    - Expected Outcome: It is expected that the user is only able to log in if the information that they inputted was corrected and matched what is present in the user database. If the user is still able to log in despite wrong information, then this test has failed.
- User_Signup_1:
    - Tests the Sign Up Functionality

    - This test will test that the user is able to sign up with their email address and new password. When the user signs up, the system checks in the database if there's a matching username. This will test the signup functionality in the User Class, where it checks the database for matching usernames. The user should not be able to sign up if they have a matching username.
    - Test Steps:
        - Launch the website from browser
        - From homepage or login page select "Sign Up"
        - Enter email or phone number and create a password
        - Click "Create Account"
    - Expected Outcome: The user should be able to create an account if their username does not currently exist in the database. If the user chooses a username that already exists, they will be prompted to create a new username.
- User_Logout_1:
    - Tests the Logout Functionality
    - Tests the logout() function to ensure that whenever a user logs out of their account, that they're no longer logged in to the system. This ensures that their information will be safe and secure while they are logged out.
    - Test Steps:
        - Press "Log out" on the top right of the screen
    - Expected Outcome: It's expected that the user is no longer logged into the system once they log out and confirm that they want to log out. This will ensure the user's information security. This test will fail if the user is still logged in despite them logging out.

Functional Tests:
- Payment_1:
    - Tests the Payment Functionality
    - Tests that the payment function on the website is able to accurately and efficiently communicate with the payment APIs. This is testing if the system and API aren't currently communicating with each other. Further, it's testing if the money is being transferred properly and accurately.
    - Test Steps:
        - Launch website from web browser
        - Add several tickets to cart
        - Go through with purchasing tickets using credit card, Apple Pay, Google Pay, etc.
    - Expected Outcome: It's expected that the system will receive accurate and efficient payment from the user and it will show up on the designated account that takes payment.
- Purchase_Refund_1:

    - Tests the Refund Functionality
    - Tests that the user is able to cancel purchases before time of film screening and receive their payment amount back
    - Test Steps:
        - 1.User opens the receipt link sent to their email or phone
        - 2. Select "Cancel ticket"
    - Expected Outcome: The user will get a second email or text confirming their cancelation and they should receive their refund within 3 days
- Database_1:
    - Tests that the purchased seats are removed from available selection in the database
    - Purchased seats will no longer be able to be selected for reservation or purchase by other users and will appear as taken on the seating selection screen
    - Test Steps:
        - Add tickets to cart and purchase tickets
        - Check database to see if it's removed
    - Expected Outcome: It's expected that when a user purchases a ticket or adds it to their cart, future users will not be able to access or purchase the same ticket. This will ensure that there aren't any duplicate tickets. The test will be considered as a failure if users are able to purchase the same seat in the same movie.
- Purchase_Verification_1:
    - Test that user successfully receives receipts through email and text
    - User should receive a receipt along with a QR code that will be scanned at the front to confirm purchase
    - Test Steps:
        - User launches website
        - Select film
        - Purchase tickets
        - Choose email and text delivery options for receipts
    - Expected Outcome: The user should receive an email and text containing a link to both the receipt and QR code

System Tests:

- Website_Launch_1:
    - This will test if the user can launch the website using different browsers. This will assume that the user's computer reaches the spec requirements of each corresponding browser. This will not test access and interaction of the website using multiple tabs on the same browser.
    - Test Steps:
        - 1. Launch web browser
        - 2. Enter the link into the search tab

        - 3. Press Enter
    - Expected Outcome: The website is able to be launched on all tested web browsers and no issues occurred.
- Website_Bandwidth_1:
    - This will test if multiple users are able to access the website at the same time. Each user is expected to only have one instance of the website open. Other concurrent website functions will also be tested, such as changing pages on the website or writing and reading from the database.
    - Test Steps:
        - 1. Launch the website from the web browser on one computer
        - 2. On another computer, launch the website from the web browser
        - 3. Repeat for at least 5 computers
    - Expected Outcome: Each user successfully enters the website.

4.5 Data Management Strategy

Software systems contain persistent and sensitive information regarding the users and the company/organization. For this system, we decided to go implement a SQL-based approach with multiple databases. We chose this approach to maximize reliability and speed in order to create the best possible experience for our customers.

4.5.1 Trade-offs

- **SQL vs. NoSQL**
    - SQL databases are relational, structured, table-based, and vertically scalable. They are better suited for applications that require complex multi-row transactions and strong data consistency.
    - NoSQL databases are non-relational, flexible in structure, and horizontally scalable.
    - In general, SQL is ideal for applications that require complex queries and reliable transactional support, while NoSQL is better suited for applications that demand high scalability, fast performance, and can handle rapidly evolving data models.
- **Single Database vs. Multiple Databases**
    - Single databases are easier to manage and maintain as all of the data lives in one place and is cheaper as fewer resources are needed. However, single databases are really only beneficial for smaller to mid-sized projects as scalability can be limited.
    - Multiple databases separate data across different databases based on specific parts of the system, enabling independent scalability. However, managing multiple databases requires more resources to upkeep. Further, more resources are required to maintain consistency across multiple databases.
    - In general, single databases are a better fit for smaller systems with limited resources, whereas multiple databases are more suitable for larger systems that need to scale different components independently and can afford the added

complexity. Multiple databases might not be as fast as single databases, but is better for this given situation.

4.5.2 Design Decisions

- **SQL Database**
    - SQL thrives on well structured and relational databases, which is what we're working with. Our entities, such as users, theater, and bookings, have clear and defined relationships with one another.
    - SQL's ACID(Atomicity, Consistency, Isolation, Durability) properties ensures reliable transactions, which can prevent double bookings or lost payments
    - SQL also allows for complex queries and reports, such as how many books were made in a certain period
    - Our system doesn't require horizontal scaling, but rather, vertical scaling, which SQL is preferred for
    - While we don't anticipate extremely high traffic on our website, SQL will be better equipped to handle increased activity.
- **Multiple Database Approach**
    - Multiple databases allow us to scale and modify each database independently. For example, we are able to update, maintain, and scale the users database separately from the theaters database
    - Multiple databases allow for fault isolation. In other words, if one database fails or is faulty, the other databases aren't affected as they're independently managed and maintained

4.5.3 Data Organization

We will be utilizing data tables. A data table is a collection of rows and columns that hold data in a structured format. Each table represents an entity within the database, such as User, Movies, Theater, etc. This allows for a structured way to store data and be able to quickly access the data based on the entity that it belongs to. In other words, data tables are like digital spreadsheets that allow for efficient management and storage of data.

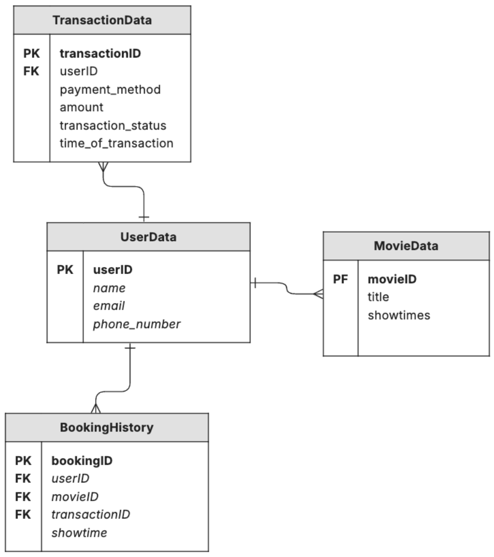
Data Tables:

- **UserData**: Contains name, email, phone number, payment history, booking history. Purpose is to store personal information of the user.
    - **BookingHistory**: Contains previous bookings of the user.
- **MovieData**: Contains movie title, rating, showtimes, language, showtime, movieID. Purpose is to provide details regarding specific movies.
- **TransactionData**: Contains payment ID, transaction status, payment method, amount, time of transaction. Purpose is to record details of transactions and to keep track of it for future reference.

4.5.4 Database Design Diagrams

Refer to sections **4.2 Unified Modeling Language Diagram (UML)** and **4.3 Software Architecture Diagram(SWA)** to see how the databases interact with the classes and variables of the system. These diagrams have been updated to account for the data tables that were considered in this section.

Data Table Diagram:



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