# RJJ's Theater Movie Ticketing System

## Software Requirements Specification

Version 2

2/20/2025

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 |
| | | | |
| | | | |

## 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.

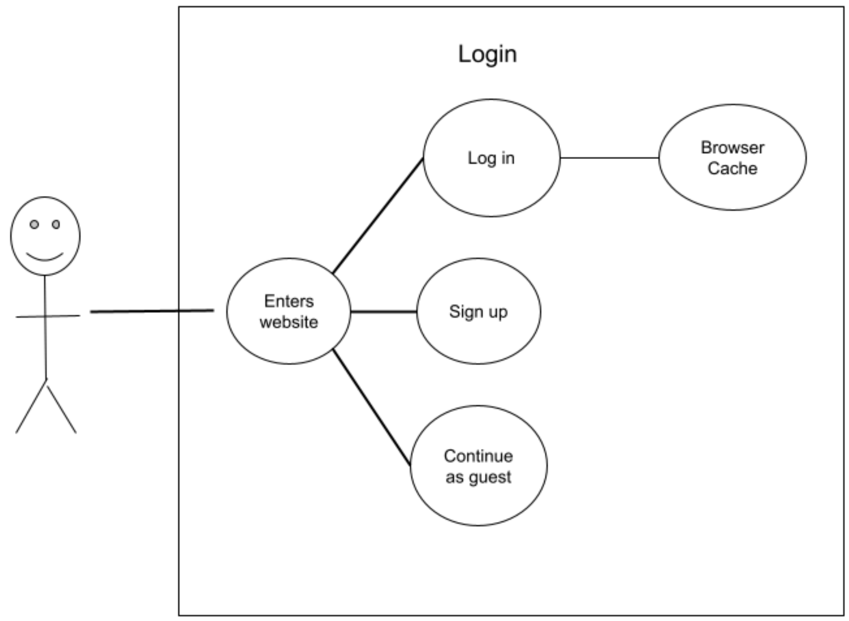
3.2.3.5 Error Handling
If a customer support agent is not available, the user will be placed on hold and be put in a queue until an agent is available.

## 3.3 Use Cases

### 3.3.1 Use Case #1: User Log-in

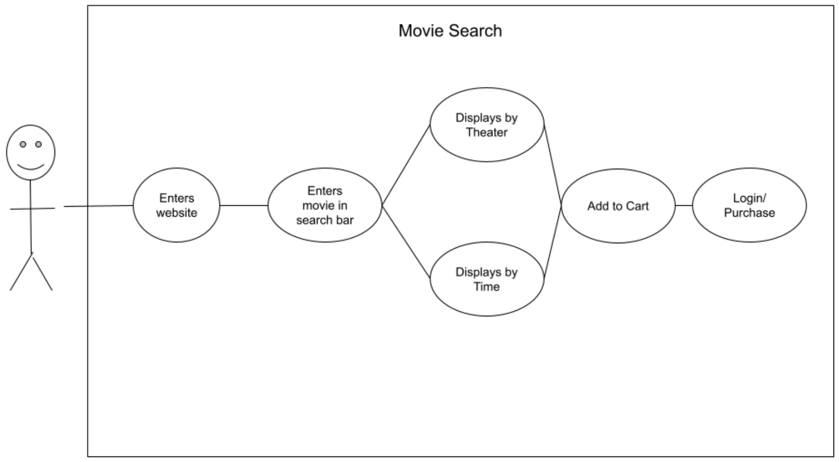
Actor: User
1. The user will add the ticket to their cart.
2. After going to their cart, the user will be prompted to login, sign up, or continue as a guest.

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



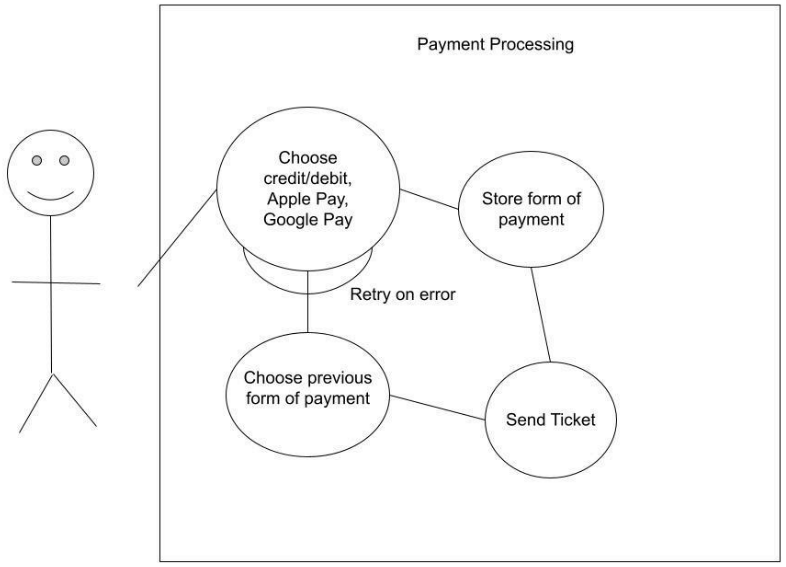
**3.3.2 Use Case #2: Ticket Browsing/Searching**

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.*

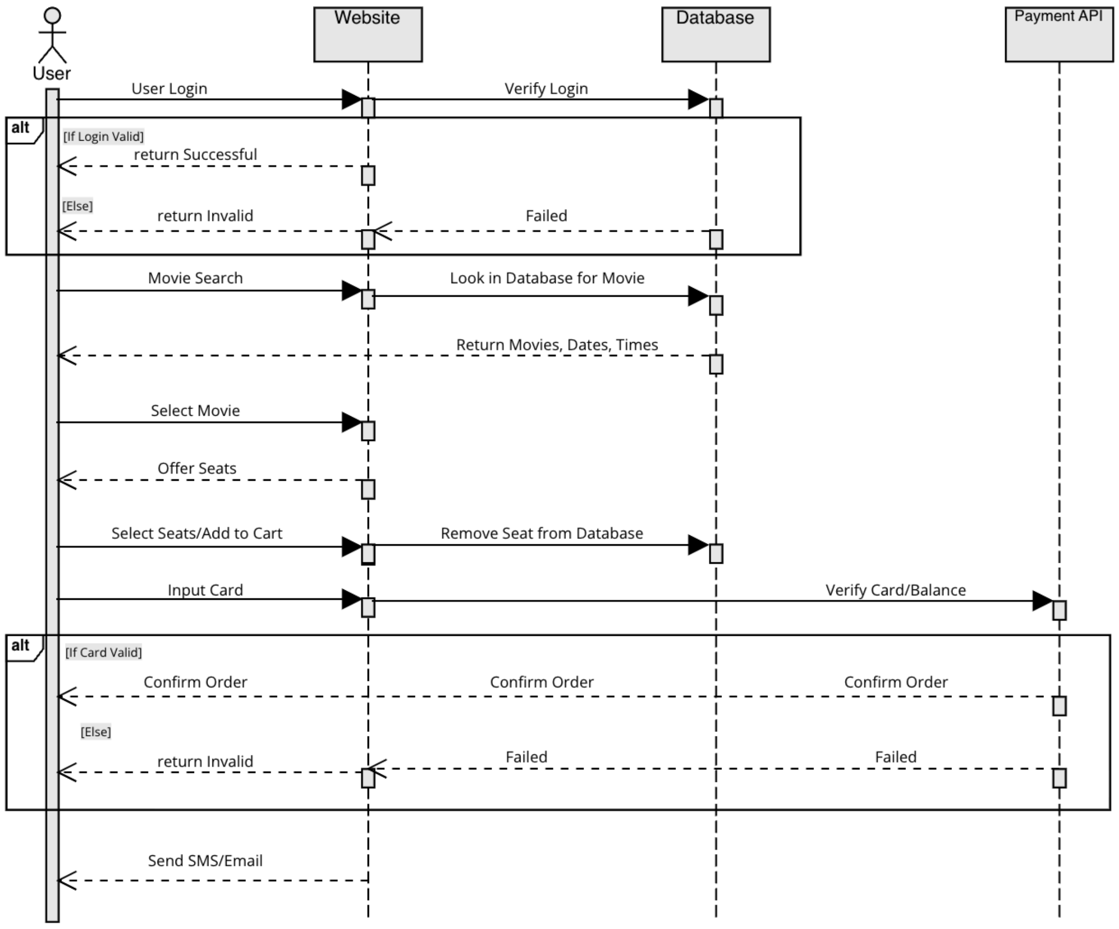
### 3.9 Other Requirements

*Catchall section for any additional requirements.*

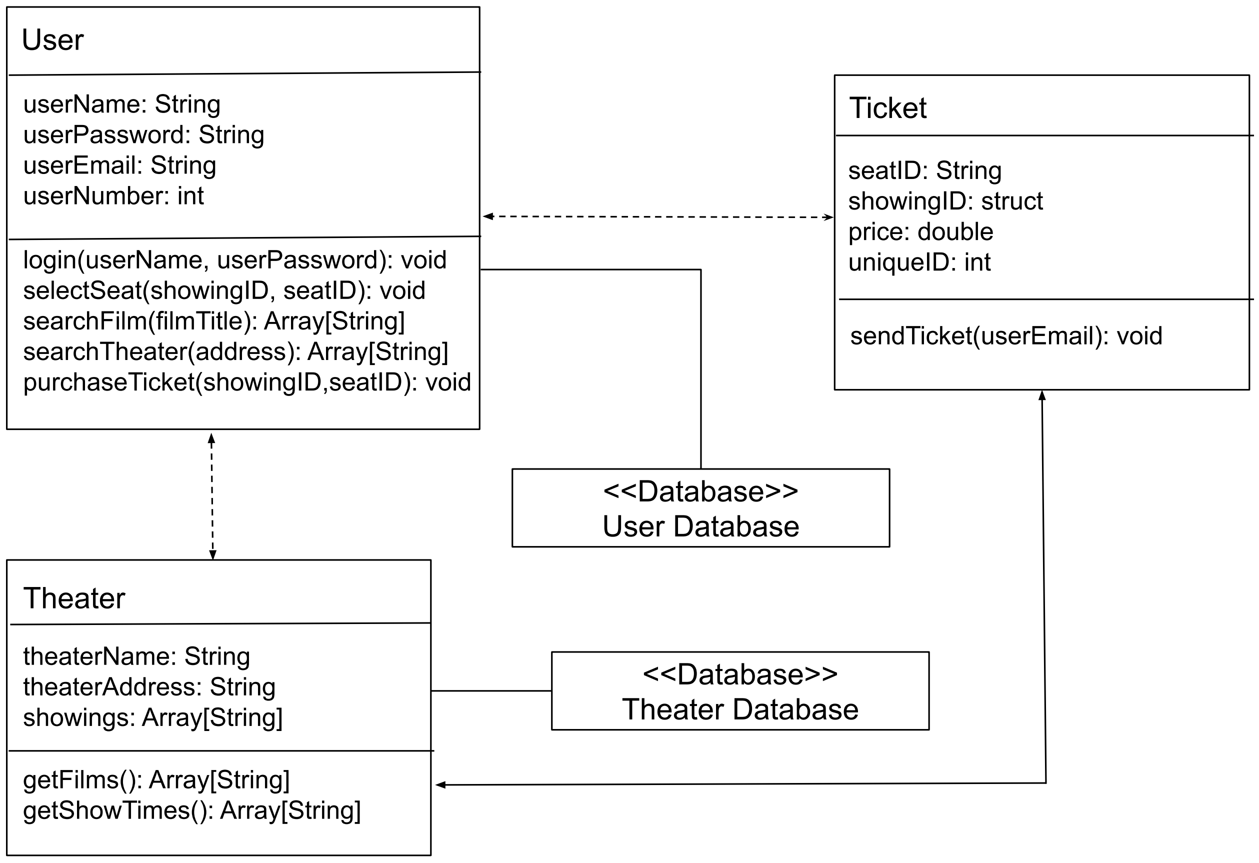
## 4. Analysis Models

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)

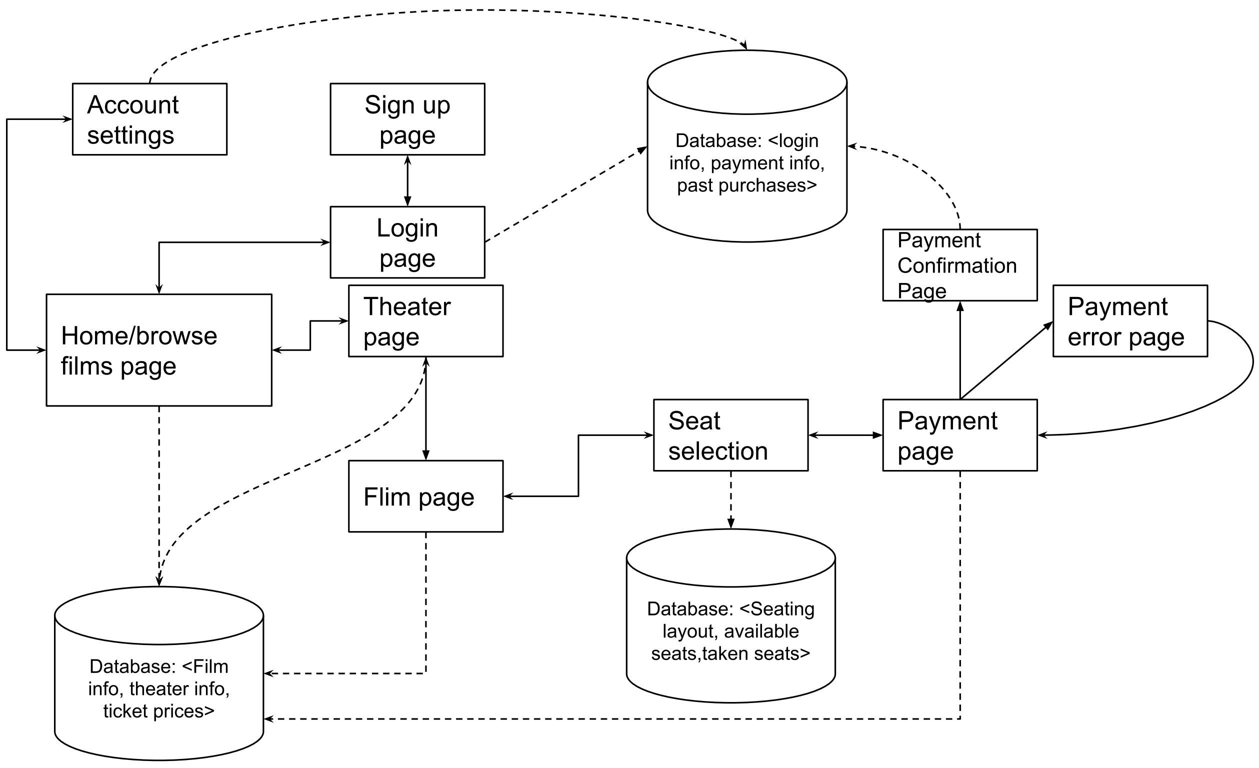

User
userName: String
userPassword: String
userEmail: String
userNumber: int
login(userName, userPassword): void
selectSeat(showingID, seatID): void
searchFilm(filmTitle): Array[String]
searchTheater(address): Array[String]
purchaseTicket(showingID,seatID): void
Ticket
seatID: String
showingID: struct
price: double
uniqueID: int
sendTicket(userEmail): void
<<Database>>
User Database
Theater
theaterName: String
theaterAddress: String
showings: Array[String]
getFilms(): Array[String]
getShowTimes(): Array[String]
<<Database>>
Theater Database

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, past purchases>
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
-**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
-**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
-**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

-**Payment Page**:

- Allows the user to input their payment information.
- Will redirect to the **Payment error page** if the payment information is not valid, prompting the user to try again.

- **Payment confirmation page**

-Notifies user of successful purchase and phone number or email address receipt has been sent to.

## 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