<J-Cinemas Ticketing System>

Software Requirements Specification

<1.0>

<9/18/2025>

<Group #4>

<Jordan Kiryakoza, Justin Collett, Jonah Ale>

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

J-Cinemas Ticketing System

Date	Description	Author	Comments
9/25/2025	1.0	Jordan, Justin, Jonah	<First Revision up to and Including 3.5>
10/9/2025	Design	Jordan, Justin, Jonah	<Design Implementation>
10/23/2025	Test Plan & Test Cases	Jordan, Justin, Jonah	<Test Cases>
11/6/2025	Data Management Strategies	Jordan, Justin, Jordan	

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 section of the SRS establishes the infrastructure for the document as a whole by defining its purpose (1.1), scope (1.2), definitions, acronyms, and abbreviations (1.3), references (1.4), and overview (1.5) of the SRS for J-Cinemas, the movie ticketing system, including its intended audience. The purpose of this document is to act as a comprehensive blueprint, detailing all non-functional (defines how a system should perform) and functional (what the system does) requirements of developing a movie ticketing system.

1.1 Purpose

The movie ticketing system J-Cinemas' aim is to provide a convenient and smooth transactional account for the intended audience, the consumers, to search for movies, select their seats, and book their tickets online. By allowing the consumer to book online using our movie ticketing system at any given time, it can remove the need for standing in line at the physical affiliated location.

1.2 Scope

- The movie ticketing system, J-Cinemas, will function as a web-based application to facilitate the online ticket purchases for our local movie theater.
- J-Cinemas' objectives include accessibility, convenience, and efficiency for the movie theater ticket booking process.

The main purpose of J-Cinemas is to provide an alternative way for the user to purchase movie tickets. It will include functions such as user registration, movie listing browsing, seat selections, ticket booking, payment processing, and producing ‘e-tickets’ for user upon payment confirmation. On the admin side, it will include features for Admin role to manage movies and users’ accounts.

It will not include functions for purchasing food, drinks, or merchandise. It will also not include a review or ratings system that helps provide user-based feedback to customers to assist in determining movie choice(s). Processing refunds won’t be included in the MTS as it must be done in person at the box office location.

Additionally, our movie ticketing system will not save users' payment information into their account.

Software Requirements Specification

J-Cinemas Ticketing System

- Benefit: To successfully reduce the time for a customer to choose and purchase movie tickets by at least 40-50% compared to the manual physical system in standing in line at a movie theater's physical location.
- Objective: Allow users to view all available seats in real-time, enabling them to purchase their preferred seats for a specific showtime for a movie within first 6 minutes (session timeout) of their interaction with the movie showtime's seat selection window.

Our system should process movie ticket bookings and provide information to users in a set time frame (up to 1 month time frame before movies are shown). The MTS also needs to ensure and guarantee the security of user accounts as well as payment transactions and payment information. The MTS needs to be stable and available for users to interact with the website, browse movie listings, access their accounts, and book tickets without little to no interruptions from the system/server.

User Registration: Users can create account on our MTS website to be able to purchase movie tickets and get notifications about upcoming movie showtimes for tickets purchase. Admin will also need to create accounts so that they are provided with higher access to the website which will allow for modifications.

Login: Admin and users will log in to the website with their information. If successful, the page will refresh with the respected profile linked to the browser's page with account details when clicked. If failed, then password re-attempt. (3 Attempts = Lockout)

Change Password: User can change their password by inputting correct responses to security questions.

Movie Listings: Users can browse movie selections by name and date, with movies displaying a short, summarized description of plot, runtime duration, and the MPA (Motion Picture Association) rating system.

Ticket Booking – Users can select movie and showtime from listings and book their ticket.

Payment: User will apply payment with [approved] credit card(s) via electronic payment system to confirm ticket booking.

Ticket Confirmation: After successful payment, user will receive an online electronic ticket that displays either a QR code or barcode, along with important information detailing: Movie name & Showtime, Seat Row & Number, and confirmation number receipt.

Admin Functionalities:

Content Management: Admin can add, modify, and remove movie information and showtimes.
Theater and Seating Management: Setting up seat placement configuration and movie showings for each individual theater screening room.

The goals of our MTS are to:

1) Provide a service that can be accessed at any given time for the user.
2) Promote the movies online that are to be shown in the box office.
3) Assist in profit increase by providing an alternative method for booking tickets.

1.3 Definitions, Acronyms, and Abbreviations

Definitions:

System: J-Cinema, the web-based movie ticket booking foundation includes the user-based website (Front-end / what the users see) and the back end that is monitored and controlled by the admins that help operate it behind the scenes. (Back-end) User: An individual that interacts with the system.

Customer: A guest OR registered user that can browse movie listings, purchase tickets, and check on their purchased movie ticket bookings. If registered users, then they have the ability to edit their account details, change password.

Admin: A specific type of user with high-access privileges that manages content of the system, including movies and showtimes, user accounts.

Showtime: A set of specific dates, times, and theater number for a given movie showing.

Seat Selection / Layout: A graphical or visual representation of the seats in a particular movie theater room. Shows whether or not the theater room has any availability based on whether or not a seat is available or already been purchased, or unavailable due to technical issues.
Booking: The system process of confirming a reservation of one (or more) seats for a specific movie and showtime.

Payment Gateway: Third-party service that is integrated within the system to securely process payment transactions.

Transaction: Process for making a payment for purchasing a movie ticket / reserving a seat at a movie booking.

Software Requirements Specification J-Cinemas
Ticketing System

E-Ticket: Electronic ticket that contains information of movie with either a QR code or barcode with seat and movie details, generated by system after ticket purchase confirmation.

Acronyms and Abbreviations:

SRS: Software Requirements Specification
UI: User Interface (For the user to interact with when on website, selecting seats, etc)
DBMS: Database Management System (Software that maintains the system's database)
API: Application Programming Interface
HTTPS: Hypertext Transfer Protocol Secure
PCI DSS: Payment Card Industry Data Security Standard
SQL: Structured Query Language (Back-End)
PDF: Portable Document Format (Generating e-tickets)
Platform: Version of Windows or Apple (i.e. Windows 11, macOS) OS:
Operating System (i.e. Android, iPhone) SMPT:
Simple Mail Transfer Protocol HTTP: Hypertext
Transfer Protocol

1.4 References

3.0 System Requirements
3.1 External Interface Requirements

1.5 Overview

The next section of this document, General Description, gives an overview of the functionality of the movie ticketing system, J-Cinema. It describes the informal factors and requirements used to help provide the background and framework for the technical requirements discussed in the subsequent chapter. It also includes the characteristics of the users (consumers) of JCinema.

The third section of this document, Specific Requirements, is more tailored towards the developers and explains the functionality of the movie ticketing system in more technical terms. While both sections of this document discuss the movie ticketing system as a whole and describe J-Cinemas' functionalities, each section is intended for separate audiences and therefore explains each area in depth differently, using different language and terminology.

2. General Description

Our product, J-Cinema, will be a movie ticketing system that users will be able to interact with over the Web. They will be able to create an account and login to keep track of necessary information. They will be able to browse and search for movies to their liking. The user will be able to select a movie ticket for purchase, confirm it, and have their results displayed.

2.1 Product Perspective

This system shall be accessible through the web, and it is self-contained and doesn't require any application or service to run besides an internet connection. It is a system that runs on its own designed to increase the discoverability of movies, and to increase the speed of buying tickets due to not having to go to a box office.

2.2 Product Functions

- The system shall have a user registration system so they can keep track of their purchases and spending.
- The system shall allow users to log in to the accounts they have created
- The system shall allow users to change their password had they forgotten or need to change it.
- The system shall allow users to search for movies, and sort in alphabetical order for easier findability
- The system shall allow users to add items to their cart
- The product shall have a cart that can hold up to 20 movie tickets at a time and allow the user to remove unwanted or duplicate tickets.
- The system shall allow users to purchase no more than 20 items per cart
- The system shall provide the user with an invoice for their purchase and confirmation after their purchase has been made.
- The system shall allow users to review their purchased history

2.3 User Characteristics

The users are expected to mainly be teenagers and young adults, with the exceptions of some parents and elders. We expect the users to either be in high school, have a diploma/degree, or be in the workforce. We expect the users to know how to use and manage basic technology, as that's what the new generations are all about. We expect the users to not have much technical expertise, so we are designing our product to be as simple and easy to use as possible.

2.4 General Constraints

- Legal constraints must abide by laws regarding the users' data as it often includes sensitive information such as a payment method.
- Database constraints, must be able to store users' account data for all previous and active users.
- Age constraints, a user must be at least 18 to be able to digitally purchase a product in the system.
- Software constraints, the system shall be able to support up to 1,000 users. In the event that the system greatly surpasses that number, it could fail.
- Cost constraints, the development of this system cannot go over a certain cost threshold.

2.5 Assumptions and Dependencies

- Assume the users are accessing our system from a windows, mac, linux, apple, or android device
- We assume the users are accessing our system from Google Chrome, Opera, Microsoft Edge, Safari
- We assume that the required systems of the users are up to date, with at least 8gb of ram and a quad core processor.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The system will provide a secure login and registration process for the users.

- Login window that has users enter a correct ID and password.
    - Upon successful login (approved authentication of data stored in user database), user can edit their account details, purchase movie tickets, utilize rewards system.
    - If an account does not exist, the user must register a new account with their personal information to purchase tickets, utilize rewards system, check on

upcoming booked tickets, edit account details. Upon successful validation and creation (by the system) of account, users' login information will be stored into the user database for potential future movie purchases.

- If password is incorrect, user can attempt 4 times before being locked out of account. User will need to reconfirm account through Forgot Password by inputting correct security question details in order to change password details.

  The system will provide real-time seat availability for each movie and their respective showtimes. The users will be able to choose their preferred seats based on availability. The system will then update the seat availability in real-time to prevent potentially double bookings from additional users.
- Movie selection window for browsing and choosing through a list of movies and showtimes.
  - Clicking on a specific movie will open a new page, showing details of movie.
    - The system will include a brief synopsis, running time, and the associated MPA (Motion Picture Association) rating.
    - Underneath brief info will show clickable links in the form of Date & Showtimes for specific movie.
    - Clicking a specific showtime will load the seat selection window.
- Seat selection window for all movies and showtimes (individually) for users to choose their preferred seats based on selection availability.
  - Upon choosing a specific movie and showtime, the user will be loaded into seat selection window.
    - An interactive visual representation of a specific theater room's layout will load showing both rows of each seat and number, allowing the user to choose their preferred seat(s), up to max limit of 20 seats.
    - When finished selecting seat(s), user will hit the Book Seats button which will direct user to a new page to prompt the user to choose the type of tickets based on their max number of seat selections in previous window.
    - The system will put a temporary hold on selected seats during the payment method transaction process.
- Ticket Choices Window
  - User will choose what types of tickets are being purchased. Child: Ticket for children under 12
    - Adult: Ticket for people 12 and older (up to 59)
    - Senior: Ticket for people 60 and older
    - Matinee: **Only shows up for purchase if showtime is before 12 pm**
  - Confirming quantity and type of tickets and hitting Ok/Next button will take user to

the payment method / transaction window.

The system will support secure payment processing using standard payment gateways. The users will be able to enter their payment information and finalize purchase transactions securely.

- Payment Method / Transaction window for inputting payment method information in to pay for movie ticket(s).
  - User will be required to input payment information including Full name, billing address, city, state, zip code. Will also include credit card number, expiration date, CVV(card verification value), and zip code the credit card is tied to.

    User will also be prompt to choose between various credit card payment options (Visa, Mastercard)
  - Information is sent to payment gateway that sends a callback notification to the ticketing system, which tells the system to mark the ticket booking as Confirmed.
  - The system releases the temporary hold on selected seats and updates seat availability and mark selected seats as now unavailable.

The system will provide a confirmation of ticket purchase bookings that is used as the reference for the user’s booking reservation.

- e-Ticket Generation / Email Confirmation
  - An alphanumeric code is generated by the system’s database to uniquely identify the movie ticket booking, which is linked to the user’s account, along with the rest of the ticket booking information.
  - An email confirmation displaying the code along with its respective barcode and seat details including: Theater Number, Row and Seat Number, and the date and time of movie showing.

    All booking information data is stored safely on the system’s database.

3.1.2 Hardware Interfaces

- This product must be run over the internet. All hardware shall require connection to the internet either by Modem, Ethernet, LAN, etc.
- This product must be accessed using an electronic device, phone, tablet, computer, etc.
- This product shall allow users to navigate the system with their mouse and keyboard and or touch screen
- This product shall use the HTTP protocol when interacting with web servers
- This product shall interact with a SQL database to store and retrieve information
- This product shall use Email servers to send digital tickets and confirmation to the user.

3.1.3 Software Interfaces

The system will use a database to store and retrieve user information. It will also implement an online payment service to process transactions. Lastly, the system will use an email service to send booking confirmations and password recovery information to users.

3.1.4 Communications Interfaces

The system will use HTTPS for communication with web browsers and the payment service. Email communication will use SMTP.

3.2 Functional Requirements

3.2.1 <User Registration feature>

3.2.1.1 Introduction –

- This feature will allow users to register an account for a movie ticketing system.

3.2.1.2 Inputs-

- The system shall accept a username or email
- The system shall accept a user's full name
- The system shall accept a password that a user creates
- The system shall accept the user's created password

3.2.1.3 Processing-

- The system shall validate that there are no empty fields
- The system shall verify that username and email are unique to avoid duplicates
- The system shall verify that the password and password confirmation match
- The system shall verify that the password meets password strength
- The system shall encrypt the password than store it in the database

3.2.1.4 Outputs-

- The system shall create a new user in the database
- The system shall display a success message to the user if registration is successful.
- The system shall redirect the user to the log in screen after registration is complete.

3.2.1.5 Error Handling-

- The system shall display "Username already in use" if username exists.
- The system shall display "Email already registered" if email exists.
- The system shall display "Password to weak" if password strength doesn't meet requirements.

- The system shall display “Invalid email format” if email doesn’t match proper email format.
- The system shall display “The field is required” if a field is left empty.

3.2.2 <User Login feature>

3.2.2.1 Introduction-

- This feature will allow users to log in to a movie ticketing system with their registered credentials.

3.2.2.2 Inputs-

- The system shall accept a username or email entered by the user
- The system shall accept a password entered by a user

3.2.2.3 Processing-

- The system shall validate that the username or email exists in the database
- The system shall verify that the password matches the stored encrypted password for the account
- The system shall reset the failed login counter if login was successful

3.2.2.4 Outputs-

- The system shall redirect the user to a home screen if the login is successful

3.2.2.5 Error Handling-

- The system shall lock the user out for 10 minutes after 4 failed login attempts

3.2.3 <Password recovery feature>

3.2.3.1 Introduction-

- This system shall allow users to recover their password if they forgot

3.2.3.2 Input-

- The system shall accept the registered email or username from the user
- The system shall accept a new password and confirm password for password reset

3.2.3.3 Processing

- The system shall validate whether the provided username or email exists in the database
- The system shall generate a single-use reset token with an expiration
- The system shall invalidate any previous active token

3.2.3.4 Outputs-

- The system shall display a confirmation message that a recovery link has been sent • The system shall send a password reset link

- The system shall display a confirmation message after a successful reset
- The system shall send an email to confirm that the account has been reset

3.2.3.5 Error handling-

- The system shall display an error message if the email/username doesn't exist
- The system shall display an error message if the token expires
- The system shall display an error message if the reset process fails
- The system shall block requests if too many recovery attempts were made in a short period of time

3.2.4 <Online payment feature>
3.2.3.1 Introduction-

- This feature will allow users to make payments for movie ticket purchases

3.2.3.2 Input-

- The system shall accept the user's payment information
- The system shall accept the user's billing information
- The system shall accept the payment amount for the selected movie ticket

3.2.3.3 Processing

- The system must integrate with a third-party payment service (Stax) to handle transactions.
- The system must receive and interpret transaction responses from the payment gateway, such as success, failure, or pending statuses.
- The system shall validate whether the provided username or email exists in the database.
- The system shall validate the user's payment and billing information.
- The system shall calculate the final ticket price.

3.2.3.4 Outputs –

- The system shall display the final ticket price
- The system shall display a confirmation message for a successful payment • The system shall generate a digital receipt

3.2.3.5 Error Handling-

- The system shall display an error message if payment gets declined
- The system shall display an error message if billing information is invalid

- The system shall mark booked seats as unavailable if payment is successful
- The system shall release seats back into availability if payment is unsuccessful
- The system shall log failed payment attempts for troubleshooting

3.2.5 – <List movies & showtimes feature>

3.2.5.1 Introduction – This feature will allow users to browse and select movies based on their showtimes.

3.2.5.2 Input –

- The system shall accept the movie title that a user clicks on from the list or types in the search bar
- The system shall accept a date input from the user to filter out showtimes
- The system shall accept a location that a user chooses to watch a movie at

3.2.5.4 Processing

- The system shall search the database for movies that match the entered title
- The system shall filter out movies based on date and location
- The system shall gather the filtered results for displaying to a user

3.2.5.3 Output

- The system shall display a list of movies with corresponding showtimes
- The system shall display where the movie is available
- The system shall display an error message if no showtimes correspond to the selected movie

3.2.5.5 Error Handling

- The system shall display an error message if there are no matching results
- The system shall display an error message if the user enters an invalid date or location
- The system shall display an error message if the showtime is unavailable

3.2.6 - <Seat selection feature>

3.2.6.1 Introduction- This feature will allow a user to select a seat for a given movie and showtime.

3.2.6.2 Input-

- The system shall accept an input for the seat a user selects
- The system shall accept the number of seats that a user wants to book

3.2.6.3 Processing-

- The system shall display a list of seats that for the selected movie and showtime
- The system shall ensure that the selected seats are available
- The system shall mark seats as reserved if a user selects them

3.2.6.4 Output-

- The system shall display a visual update for available and unavailable seats
- The system shall confirm that a seat was selected
- The system shall display a message to the user if a seat becomes unavailable before payment is processed

3.2.6.5 Error Handling-

- The system shall display an error message if the seat is already reserved
- The system shall display an error message if seat information fails to load due to a system error

3.2.7<Ticket delivery feature>

3.2.7.1 Introduction – This feature will deliver purchased tickets to a user in a digital format.

3.2.7.2 Input –

- The system shall accept the users chose delivery method.
- The system shall accept the users contact information

3.2.7.3 Processing –

- The system shall create a digital ticket that displays movie title, showtime, seat selection, and a barcode
- The system shall update the database and mark the ticket as delivered once it is sent

3.2.7.4 Output –

- The system shall deliver the ticket to the users chosen delivery method
- The system shall display a confirmation message to ensure the ticket was delivered

3.2.7.5 Error Handling –

- The system shall display an error message if the ticket is not delivered due to system issues
- The system shall provide an alternative delivery method option if the initial delivery method fails

3.2.8<Ticket history>

3.2.8.1 Introduction – This feature will allow users to view their history of tickets they purchased.

3.2.8.2 Input –

- The system shall accept filter inputs

3.2.8.3 Processing –

- The system shall get tickets that were purchased in the past that are associated with a user’s account
- The system shall filter out results according to the users input

3.2.8.4 – Output

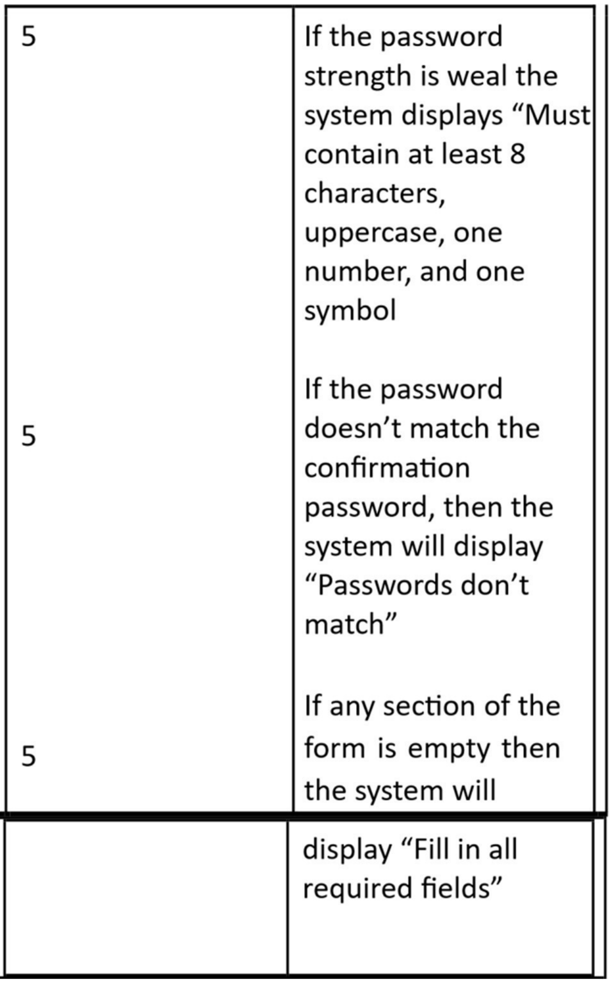
- The system shall display a list of tickets that were purchased by the user in the past
- The system shall display the number of results
- The system shall display “No history” if no records are found

3.3 Use Cases 3.3.1 Use Case #1 Register for an account

Use Case #1	Register for an account

	3	The user fills out the registration form
	4	The user submits the registration form
	5	The system validates the details in the registration form
	6	If validation is successful a confirmation message will be displayed
	7	If validation is unsuccessful then an error message will be displayed
Extensions or Variations		

Step	Branching Action



	5	If the password strength is weal the system displays “Must contain at least 8 characters, uppercase, one number, and one symbol
	5	If the password doesn’t match the confirmation password, then the system will display “Passwords don’t match”
	5	If any section of the form is empty then the system will display “Fill in all required fields”

3.3.2 Use Case #2 Log in to account

Use Case #2	Log in to account

Goal in Context	The user enters their username/email for their registered account.
Preconditions	The user has a registered account
Success End Condition	The user successfully logs into their account and is granted access to the system.

ount details.
rvice
credentials g in credentials ss to the ystem

lay an error message

edirect them to “Forgot password” recovery process

Use Case #3 3.3.3: Pay for a ticket

Use Case #3	Pay for a cket
Goal in Context	The user pays for a cket for a given movie that they have selected.

Preconditions	The user has sufficient funds.
Success End Condition	The system validates payment successfully.
Failed End Condition	The system declines the card due to insufficient funds.

Primary Actor	Cardholder	
Secondary Actors	Alternate payment methods, the CardHolders bank	
Trigger	The cardholder provides banking information	
Description		
	Step	Action
	1	The user enters their banking/card information.
	2	The system validates the banking/card information
	3	The payment goes through, and the ticket is purchased.
Extensions or Variations		
	Step	Branching Action
	1	If a user leaves a section empty, then display “Fill in all required fields”

	2	If a user’s card declines then permit them to enter another card
	3	If the payment fails due to system errors then display “System is down at the moment”

Use Case #4 3.3.4 Browsing and selecting movies

Use Case #4	Browsing and selecting movies
Goal in Context	The user will browse a list of movies and then select a movie based on their preference.
Preconditions	A user must be logged into their account.
Success End Condition	A user’s movie selection is displayed when they click on it.

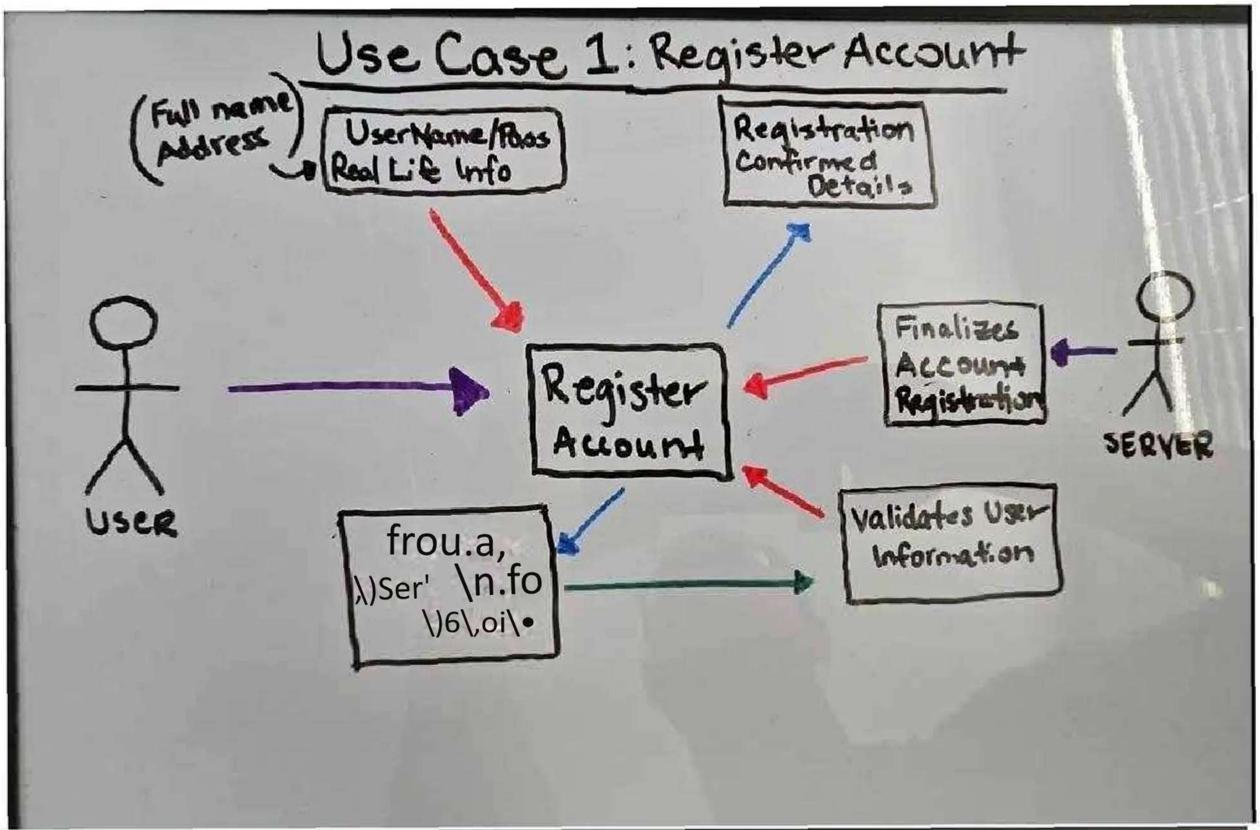
Failed End Condition	A user's movie selection is either buffered or doesn't process due to system error.
Primary Actor	A customer

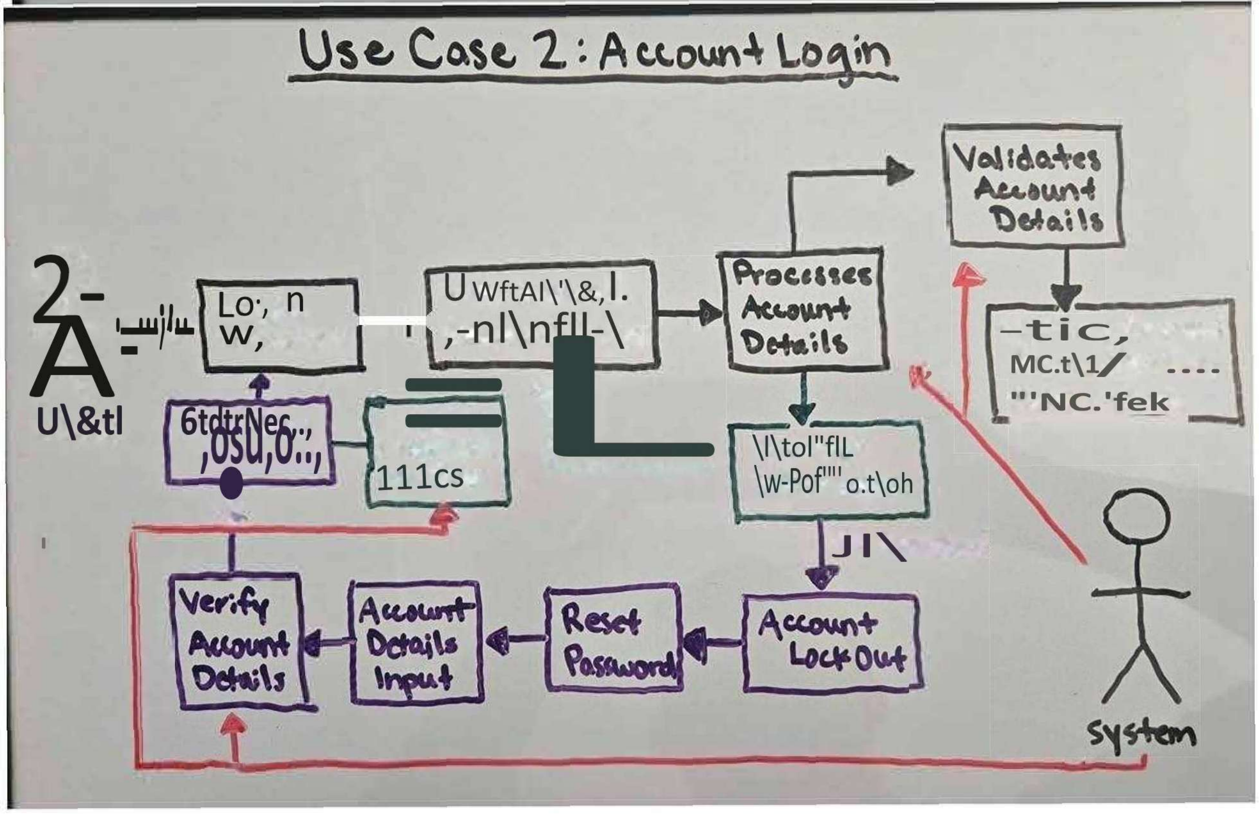
Secondary Actors	System, Movie database
Trigger	A user clicks on the movie they selected
Description	

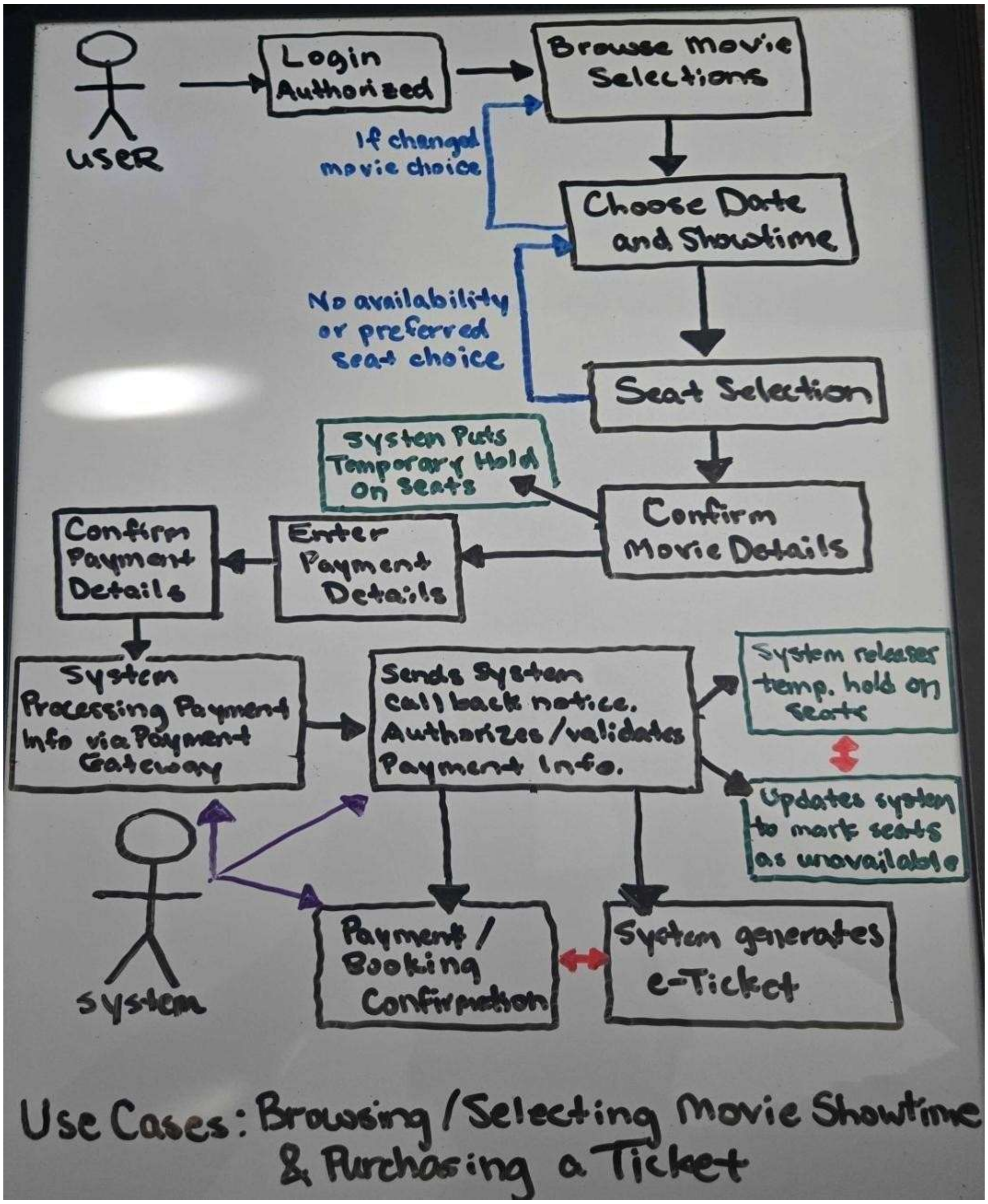
Step	Action

	1	The user clicks on a movie they selected.
	2	A movie is displayed.
Extensions or Variations		
	Step	Branching Action
	1 1	The system displays movie details when a user hovers over a movie

	2	The user clicks a movie that isn’t available so the system displays “The movie is not available”
	2	The user cancels the movie before the movie loads and the system returns the user to the list of movies
		The system loads the wrong movie because of a database mismatch. The system notifies the user and asks them to try again.


Use Case 1: Register Account
Full name
Address
UserName/Pass
Real Life Info
Registration Confirmed Details
Register Account
Finalizes Account Registration
Validates User Information
USER
SERVER


Use Case 2: Account Login
Validates Account Details
Processes Account Details
Verify Account Details
Account Details Input
Reset Password
Account Lock Out
System


USER
Login Authorized
Browse Movie Selections
If changed movie choice
Choose Date and Showtime
No availability or preferred seat choice
Seat Selection
System Puts Temporary Hold On Seats
Confirm Movie Details
Enter Payment Details
Confirm Payment Details
System Processing Payment Info via Payment Gateway
Sends System call back notice. Authorizes/validates Payment Info.
System releases temp. hold on seats
Updates system to mark seats as unavailable
System
Payment / Booking Confirmation
System generates e-Ticket
Use Cases: Browsing/Selecting Movie Showtime & Purchasing a Ticket

3.4 Classes / Objects

User class, Movie class, Showroom class, showtime class, booking class, seat class, theater class

3.4.1 <User class>
Attributes: Username -> string, Password -> string
Objects: User objects
Functions: login(username, password)
3.4.2 <Movie class>
Attributes: Title -> string, Genre -> string, Duration -> string, Rating -> double

Objects: Different types of films
Functions: AddMovie(title), DeleteMovie(title)

3.4.3 <Theater class>
Attributes: Name -> String, Address -> String, Number of rooms -> int
Objects: Different theaters in the region
Functions: no functions

3.4.4 <Showroom class>
Attributes: Room number -> int, Capacity -> int, Movie playing -> string, Seat-layout -> int, inherits movie class Objects: Different showrooms.
Functions: Addmovie(title, Room Number), GetMovie(MoviePlaying)

3.4.5 <Showtime class>
Attributes: Start time -> String, Movie Name -> String, Showroom number -> int, #ofavailableseats -> int, Date -> string
Objects The different movie showing and their showrooms will be inherited to this class.
Functions: DisplayStartDate(Movie Name, ShowRoom number, Start time, Date) DisplayAvailableSteats(#ofavailableseats)

3.4.6 <Seat class>
Attributes: Seat number -> int, Seat status -> string Objects: different seat numbers.
Functions: Getstatus(Seat number, Seat status)

3.4.7 <Booking class>
Attributes: Inherits username, Room number, Capacity, Start time, Movie name, #ofavailable Seats, Seat number, Seat status
Objects: User object, Showroom object, Showtime object, Seat object, all used to verify and book the user.
Functions: BookUser(Username, Room number, Start time, Movie name, seat number)
ViewTicketStatus(Username, Room number, Start time, Movie name, seat number)

3.4.8 <Payment class>
Attributes: paymentId, amount, payment method, status Objects:
Each payment order connected to an order.
Functions: processPayment(orderId, amount, paymentmethod) confirmPayment(paymentId)

3.4.9 <Ticket class>
Attributes: orderId, seatId, showtimeId, qrCode
Objects: Individual tickets created for each booking.
Functions: generateQrCode(ticketId) validateQrCode(qrCode)

3.4.10 <Notification class>
Attributes: userId, message, timestamp, datetime
Objects: Notification messages for booking cancellations, confirmations, or reminders.
Functions: sendNotification(userId, message), viewNotifications(userId)

3.4.11 <Logging class>
Attributes: Inherits username, Room number, Capacity, Start time, Movie name, #ofavailable Seats, Seat number, Seat status
Objects: User object, Showroom object, Showtime object, Seat object, all used to verify and book the user.
Functions: BookUser(Username, Room number, Start time, Movie name, seat number)
ViewTicketStatus(Username, Room number, Start time, Movie name, seat number)

3.4.12 <PaymentGateway class>
Attributes: gatewayName, apiEndpoint, status
Objects: Depicts the external payment gateway service integrated with the system.
Functions: startTransaction(orderId, amount), verifyTransaction(paymentId)

3.5 Non-Functional Requirements

3.5.1 Performance

- The product shall be web based and must be run from a web server.
- The product shall have an initial load time that will vary depending on internet strength and the browser in which the user loads the system.
- The performance of the product shall depend on the client/customers internet strength, speed, and their hardware components.
- The product shall have an uptime of 95% or higher per day
- The product shall have a response time of less than 400 MS at any given point in the day.

3.5.2 Reliability

- The product shall support up to 1000 users at once
- The product shall be processed in less than a second 95% of the time
- The product shall not crash upon too many instructions • The product shall maintain its speed given many active users
- The product shall be accessible anywhere in the U.S.

3.5.3 Availability

Any and all information requested should be quickly available from any computer or phone's web browser to the (authorized) registered user.

- Due to demand of seat availability, the system will not save progress for any user who logs out or closes the window in middle of the seat selection and / or payment transaction process.

o Forms of logging out include but not limited to: closure of J-Cinema browser page, exceeding the given time frame to finalize ticket bookings, or lost connection to server through any means. o If no follow up activity from user after a set period of time of closing connection and not finalizing ticket booking, a periodic email will be sent to user's email address on file in user's account, suggesting to re-book the tickets or to have another look at other showtimes.

3.5.4 Security

- The movie ticketing system is 100% accessible to only a secure authorized registered user account. This requires a username and password in order to be registered into the user database as an authorized user/member.
- The system will implement management of user sessions including session timeouts and reauthentication for operations involving sensitive information.
- Utilizes a secure payment processing system that handles sensitive payment information securely through the use of a payment gateway.
- The website also has implemented measures in place to ensure that only authorized users (registered accounts) can access specific functions and data.

3.5.5 Maintainability

- The website is maintained such that any updates necessary to the movie ticketing system is managed through a backend system that is supported by the Admins, which controls and directs the information between the movie studios and the local theater.
  - o The updates for the system processes periodic changes including updated movie showtimes, new upcoming movies, and enables Admin-access accounts to adjust specific details.

3.5.6 Portability

- The system shall be created to be run on any version of windows 10 or later.
- The movie ticketing web portal must be fully functional and render consistently across the latest stable versions of major web browsers, including Chrome, Firefox, Safari, and Edge.
- The system's database plan and data must be compatible with SQL database systems to allow for migration.
- System backups must be restorable on a different hardware or software setup than the one in which they were created on.

3.6 Inverse Requirements

State any *useful* inverse requirements.

3.7 Design Constraints

Specify design constraints 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

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 the SRS's requirements.

4.1 Sequence Diagrams

4.2 State-Transition Diagrams (STD)

4.3 Data Flow Diagrams (DFD)

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.

6. Software Design Specification

6.1 Software Architecture Overview

- Our software is designed to simplify and remove the chore of having to buy a movie ticket when you arrive at the theater. This way, customers will be able to access our ticketing system online, and have everything prepared beginning with them booking, going through the transaction process, and then securely paying via an external gateway. It is supposed to provide convenience and efficiency when purchasing movie tickets.

- Archtectural diagram of all major components – Jonah -- 1W
- UML Class Diagram -- Justin -- 1W
- Description of classes – Jonah/Justin -- 4D
- Description of attributes -- Jonah/Jordan -- 4D
- Description of operations -- Jordan -- 4D

6.1.1 Architectural Diagram of All Major Components

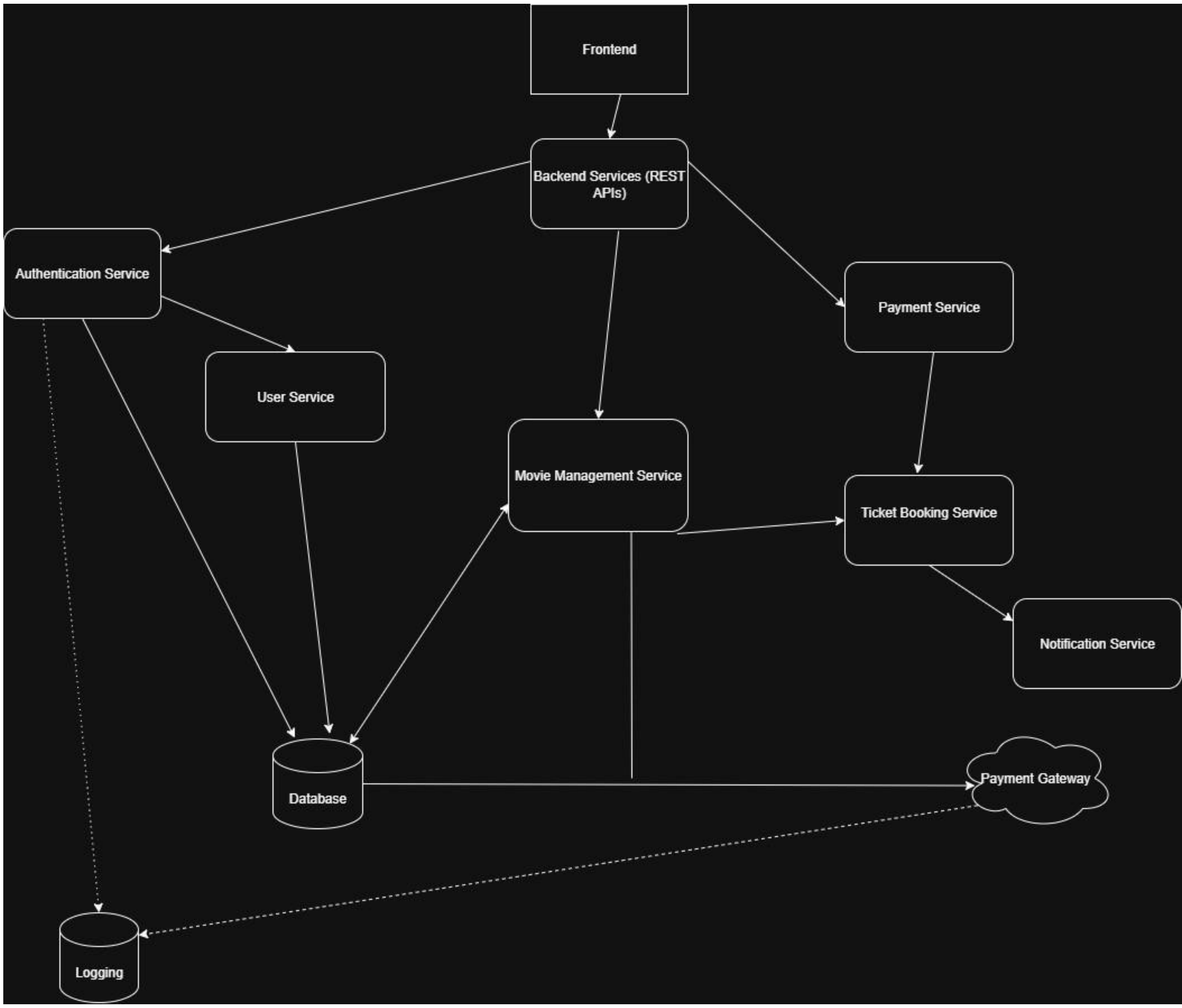


Figure 6.1.1 shows the high-level architecture of the J – Cinemas Movie Ticketing System, which depicts the relationship between the frontend, backend services, and components that support those things. The system follows a multi-tier microservices architecture, which spreads the system out into independent services that handle specific tasks. Each component will communicate through REST APIs whereas the main components are described below:

Frontend – The user interface that allows customers to brows movies, select seats, and purchase tickets. It communicates with REST APIs to get and send data.

Backend Services – The main communication between the frontend and backend microservices. It routes client requests, manages API responses, and handles data consistency across services.

User Service – Handles user-related data like registration details. Works with Authentication Service and the Database.

Movie Management Service – Manages movie listings, theater details, and showtimes. Will perform CRUD operations for movie data and interacts with the Database for storage.

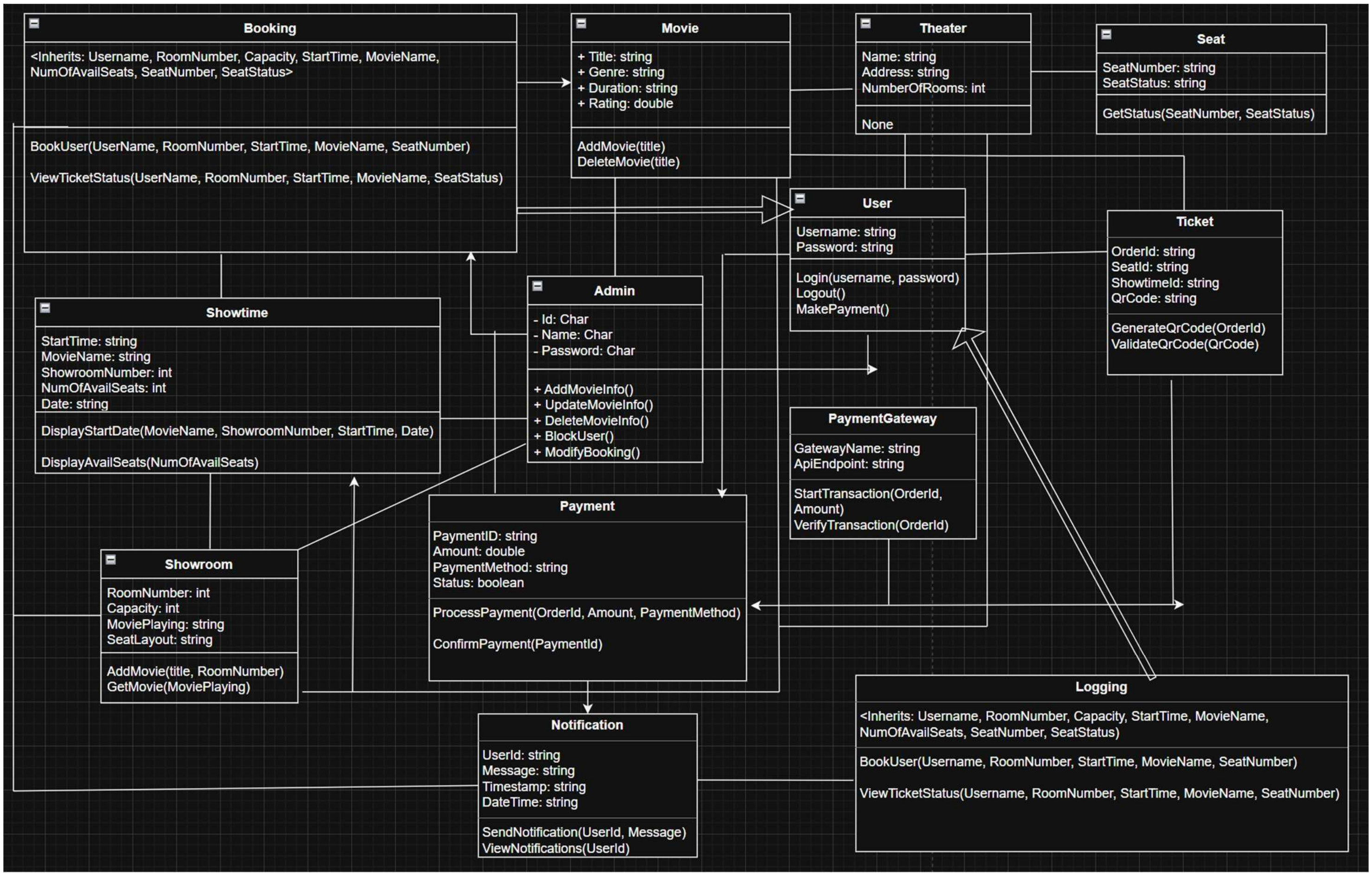
Ticket Booking Service – Navigates the ticket reservation, checks available seats, confirms bookings, and communicates with the Payment and Notification service.

Payment service – Processes transactions and is linked with the external payment gateway to handle payment authorization.

Notification Service – Sends notifications about ticket details through a specified communication method.

Database – The main place where all system components and information storage will live. Payment Gateway – Third party payment processor that will handle transactions and confirm successful payments to the payment service.

6.1.2 UML Class Diagram



6.1.3 Description of Classes:

6.1.3.1 <User Class> - Represents a customer that is registered in the system. It will handle authentication, folding profile info, and starting bookings and payments. It will interact with the Booking and Notification class and will be verified by the authentication service before those interactions begin.

6.1.3.2 <Movie class> Represents a film that is currently available. Provides metadate used by the front end and by scheduling.

6.1.3.3 <Theater Class> Represents a venue that will have multiple rooms. Owns a collection of showrooms and provides location detail in terms of schedules and bookings.

6.1.3.4 <Showroom class> Represents an individual venue within a theater. Hosts showtimes for selected movies and contains many seats.

6.1.3.5 <Seat class> Represents a seating position within a showroom. Tracks availability/reservation status and is linked to a showtime when it is held or sold.

6.1.3.6 <Booking class> Represents the user’s purchase for one or more seats for a showtime. Will handle seat holds, price calculation, payment delivery, and creation of tickets if successful.

6.1.3.7 <Payment class> - Represents an internal record of the user’s transaction for a booking. Manages payment status and coordinate with the payment gateway for authorization.

6.1.3.8 <Ticket class> - Represents proof of purchase for a seat at showtime. Generate a successful payment and later validated.

6.1.3.9 <Notification class> - Represents messages that are sent to a user. It is triggered by events such as booking or payment.

6.1.3.10 < PaymentGateway class> Represents the payment gateway provider. Handles API calls to authorize, confirm, and refund payments and will provide details to the payment process.

6.1.4 Description of Attributes –

Username – The unique identifier assigned to a user for login.
Password- The confidential key that user uses to access their account. Title – The name of the movie.
Genre – The category of the movie.
Duration – The overall running time of the movie.
Rating – The quality score of the movie.
Name – The title of the theater location. Address – The location of the theater.
Number of rooms – The number of auditoriums/showrooms within the theater.
Room number – The label that pertains to a showroom.
Capacity – The overall number of available seats.
Movie playing – The movie currently being played in the showroom.
Seat-layout – The scheduled arrangement in the showroom.
Start time – The scheduled time for when the movie starts.

Showroom number – Identifies the showroom where the movie is being played
Numofavailableseats – The number of remaining seats available for booking.   Date –
The date the movie is being shown.

Seat number – The specific seat assigned for a viewer.
Seat status = The seat that is available, reserved, or occupied.
PaymentId – A unique code that depicts a payment transaction.
Amount – The total cost of the booking or order.
Payment method – The method of payment being used.
Status – The state of the transaction.
OrderId – The reference ID that pertains to a ticket that also pertains to a particular booking.   SeatId –
The unique code that a seat is assigned to in the database.
ShowtimeId – Links a ticket or booking to a showtime.
QrCode – A digital code created for scanning and validating a ticket.
UserId – Identifies the user associated with a notification.
Message – The content of the notification.
Timestamp – The time a notification is created.
Datetime – A record of the date and time of an event.
GatewayName – The name of the external payment system that is used.
ApiEndpoint – The path that is used to sent data to a payment service.   Status – Indicates the
status of the payment gateway.

6.1.5 Description of Operations

BookUser(Username, RoomNumber, StartTime, MovieName, SeatNumber) – will use these variables to book the user, changing the SeatNumbers status to booked, as well as displaying information about where the movie will be, when it will play, and the Users name for confirmation.

AddMovie(Title) – Will use the Title to add a movie to the movie class.

DeleteMovie(Title) – Will use the Title to potentially delete an accidentally or canceled movie from the movie class.

DisplayStartDate(Theater, MovieName, ShowRoomNumber, StartTime, Date) – Used to display the start date of a particular movie, including the theater and rooms they will be playing in.

DisplayAvailableSeats(NumOfAvailableSeats) – Displays the number of available seats for a particular movie at a certain theater.

Login(Username, Password) – Used to log the user into the system, storing the unique username and password combination in a database.

ViewTicketStatus(DisplayStartDate(), Username) Used to see the start place and date of the given ticket that's attached to a certain user.

GenerateQRCode(TicketID) – Generates the QR code for a given ticket ID for booking. ValidateQRCode(qrCode) – User

must scan the QR code to validate their booking.

ProcessPayment(orderID, amount, paymentMethod) Uses an external system to process payment the user to the system and generate a payment ID

confirmPayment(PaymentID) Processes the generated payment ID to ensure that the payment is valid, and ties it to the users account.

SendNotification(UserID, Message) Allows the system to send a given message to the user.

viewNotifications(UserID) Allows the user to view the notifications they have received that are tied to their account.

startTransaction(orderID, amount) Uses the order ID that pertains to a certain booking to begin a transaction between the user and the desired ticket.

VerifyTransaction(PaymentID) Uses the previously generated payment ID to verify the user and their payment, and that the payment has gone smoothly.

7. Test Plan

7.1.1 Introduction

Purpose: This test plan describes the scope, objectives, and methods for validating our online movie ticketing system, JCinemas', software design. It ensures that the software system follows the specifications in the SRS.

Audience: This section is intended towards the developers, testers, and project managers.

References: Test Cases Excel Spreadsheet

Software Requirements Specifications for J-Cinemas

Software Design Specifications for J-Cinemas

7.1.2 Scope

In Scope:

- User registration and login
- User profile account modifications/changes.
- Admin privileges that allow for management/modifications of movies, showtimes, theater rooms, and changes to user accounts.
- Browsing and searching for movies and showtimes
- Selecting seats from an interactive mapped layout
- Processing payments through a secure gateway
- Delivering confirmations of bookings through email

Out of Scope:

- User rewards / loyalty system integration
- Analytics and data reporting for theater management
- Mobile application testing (our version is strictly for web-based purposes)

7.1.3 Strategy & Approach

- Test Types:

o Unit Testing: Individual unit modules, such as seat selection and payment processing.

o Integration Testing: Ensures that individual modules, such as booking a ticket and payment system, work smoothly together.

o System Testing: Verifies the whole, combined system against their functional and non-functional requirements. o User Acceptance Testing: End-users will test the system to verify and confirm it meets the needs of the business and consumers. o Performance Testing: Checks to determine the stability and responsiveness of the system under various load conditions.

o Security Testing: Evaluates the integrity of the system to assess potential vulnerabilities, prioritizing more around users' data and payment processing.

- Test Data: A diverse data set will be prepared, including valid and invalid inputs, as well as boundary value limits.

(Maximum amount of seats able to purchased at a time, etc) o

Valid Test Data

- Login with valid(correct) inputs (email/pass)
- Movie Selections: Movies currently being shown

    - Seat Selection: Row + Number of Seat in specific showroom.
    - Payment: Credit card with all valid information
    - Dates & Times: Correct present and future date and times for movies and their respective showtimes.
    - Quantity: Buying 1 ticket vs buying 4 tickets o Invalid Test Data
    - Login: Incorrect email or password or non-existent email
    - Seat Selection: An already purchased seat still ‘available’
    - Payment: Expired credit card, or incorrect formatting card number, or incorrect CVV.
    - Date & Times: Showtime of a movie that was shown in the past, or a movie that was shown in the past is still active in the system for end-users.
    - Quantity: Being able to purchase 0 tickets or buying more than maximum amount of purchasable tickets limit.
  - Boundary Value Limits
    - Buying the maximum number of tickets allowed per transaction.
    - Buying the minimum number of tickets.
    - Choosing a showtime for a movie that is already sold out.
    - Choosing a seat for a showtime that is considered the last available seat / last available seat in the row.
- Functional Testing o User Management: Test registration, login with valid / invalid credentials, password recovery o Booking: Test booking a single and multiple tickets, attempting to book tickets for a seat already purchased. o Seat Selection: Ensure seat display map displays correct, updated arrangements, unavailable seats are blocked, and concurrency issues so that multiple people can’t book the same seat(s).
  - Payment Integration: Test successful payment processing and handling of failed transaction attempts through the integrated gateway.
- Non-Functional Testing o Performance Testing: Simulate heavy user load to ensure the system remains responsive and stable during peak booking time frames.
  - Security Testing: Check for vulnerabilities specifically aimed towards user data information and payment details. o Usability Testing: Assess and evaluate the ease of use and effectiveness of the user interface across different platforms.

7.1.4 Roles & Responsibilities

Test Manager: Oversees the entire testing operation, from start (planning) to end (execution).

Test Engineers: High-level strategy and broader testing that includes: designing, creating, and executing the test cases.

Developers: Low-level, technical unit testing to ensure code runs efficiently and to help support bug resolution.

End-Users: Participates in User Acceptance Testing (UAT) in the final phase of the software development process that validates whether the software works as intended from a customer’s(user) point of view, confirms if it’s ready for release, and provides feedback.

7.1.5 Entry and Exit Criteria

- Entry Criteria: (When to start Testing) o The Software Design Specification (SDS) document is approved. o Test cases are well-written and reviewed.
    - o The test environment is finalized in setting up and considered stable. (must reflect true real conditions as if software system was live/up and running)
- Exit Criteria: (When to stop testing) o All planned test cases have been executed.

 A certain percentage of those test cases have passed. (example: 96%) o All critical and high priority bugs and issues have been fixed AND re-tested.

- o Test report has been reviewed and signed off by test manager.

7.1.6 Test Schedule & Deliverables

Outlines the timeline for all testing activities, specifying when each phase and major task will occur.

Start Date: 10/09/2025
End Date: 10/23/2025

Deliverables:

- Test Plan document
- Test cases and Scenarios.
- Test Data
- Bug Report Logs
- Test Execution Reports
- User Acceptance Test Reports

7.1.7 Test Cases (List)

- Login
- Change Password
- Invalid Credential Login
- Change Password Email Doesn’t Exist
- Movie Selection
- Movie Search
- Account Registration
- Email Validation
- QR Code Generator
- Payment Process
- Successful Booking

- Prevent Double Booking
- System Test 01
- System Test 02

7.1.8 Risk Management

The systematic process of identifying, assessing, and mitigating potential risks and threats that could negatively impact a company or organization’s objectives.

Risk 1: Integration of the payment gateway fails, stopping all transactions from going through.

Assessment 1: High impact priority Potential loss in revenue

Mitigation 1: Develop and test against a mock version of the payment service first. Also, can ensure that an alternative payment option is available.

Risk 2: The system experiences a drop in performance or crashes under a heavy load.

Assessment 2: High impact priority Potential Loss in revenue and damage against company reputation.

Mitigation 2: Implement heavy integration as well as conditions for racing (time-based) testing for seat selection module.

Risk 3: Customer data including payment information, is stolen or compromised due to vulnerabilities found in security.

Assessment 3: Highest impact priority Potential legal issues and severe damage against credibility of company in maintaining security of user accounts and payment information.

Mitigation 3: Perform comprehensive security testing (example: penetration testing/vulnerability assessments) to help uncover weaknesses that can be reinforced.

Risk 4 (Extension of 1): Payment transactions fail on system side, but the user is still charged.

Assessment 4: High impact priority Refunds, customer disapproval ratings increase

Mitigation 4: Test the integration of the payment gateway thoroughly, aiming to include both failed and successful transaction attempts that can also account for network timeout issues or potential bugs or errors that can arise on the payment processing side.

Risk 5: Movie ticketing system, J-Cinemas, doesn’t function correctly on all major devices, operating systems, and browsers.

Assessment 5: High impact priority Can potentially eliminate/alienate a huge section of user bases.

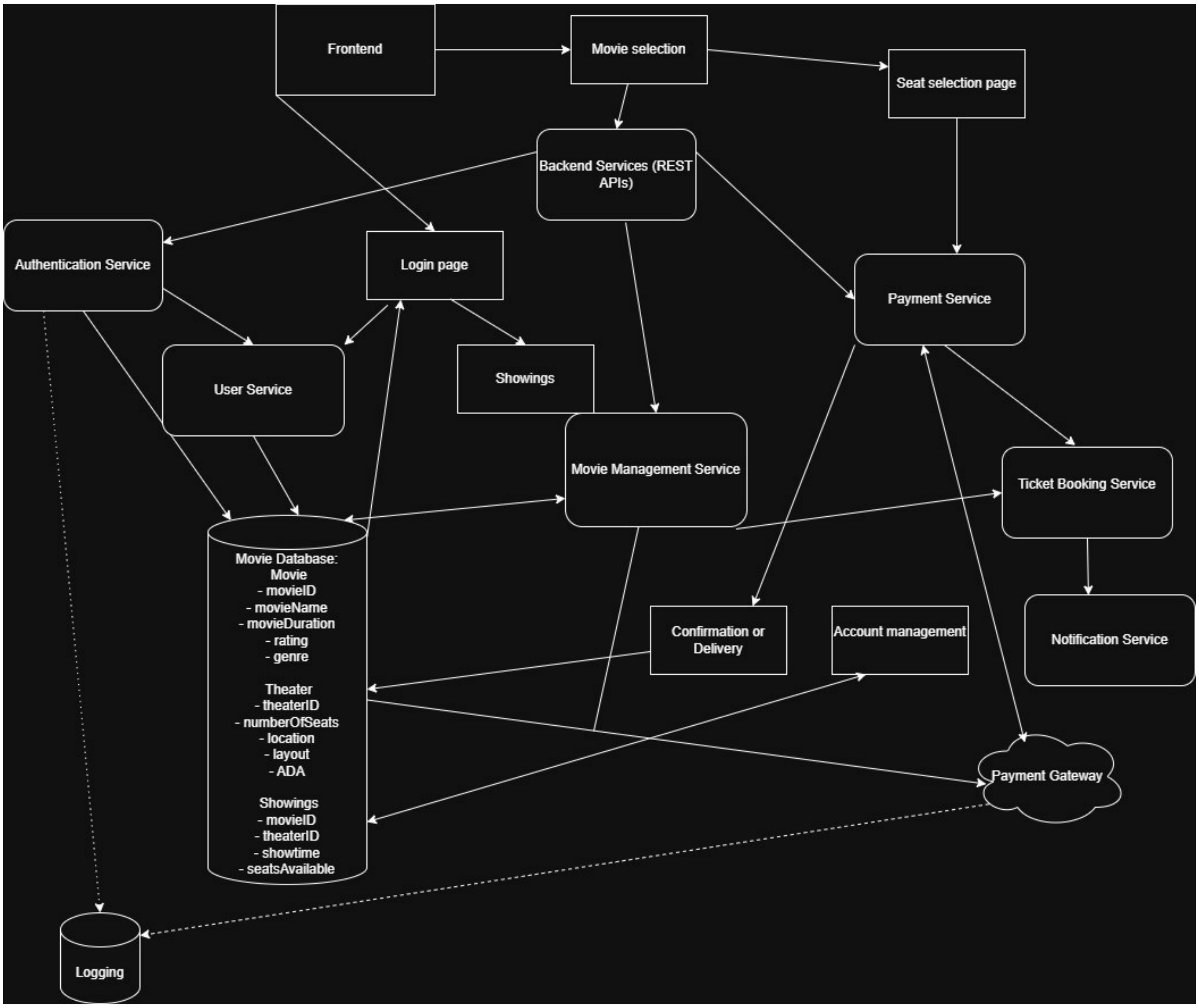
Mitigation 5: Perform compatibility testing for a range of all major browsers (Chrome, Safari, Firefox, Edge), devices, and operating systems (Android, iPhone, Computer Desktop Windows, Linux)
Risk 6: The system allows multiple users to book the same seat, resulting in overbooking and potential conflict with multiple customers.

Assessment 6: High impact priority Potential loss in revenue, dissatisfaction amongst customers.

Mitigation 6: Create test cases specifically to test concurrent user access in which multiple users attempt to book the same seat(s) at the same time. Test reservation of seats for showtime and verifies that the system correctly handles the process of temporarily locking the selected seat when the first (original) user chooses the seat based on their time frame window and either 1) making the seat available again if the original user does not complete their purchase or 2) locks the selected seat out once the purchase of the original user is complete and finalized.

8. Architecture Design With Data Management

8.1 Software Architecture Diagram



8.2 Data Management Strategy

Our movie ticketing system is going to be implemented using the SQL relational database as our primary management strategy. The system must manage data that doesn't change, including user accounts, movies present in the catalog, theaters and their locations, etc. We believe a relational database to be the best strategy since we can directly link booking data to accounts, theaters to showtimes and their number of seats, number of seats to a certain showroom, etc. There are many relationships among data in this system which is why a relational DB like SQL is a great choice.

8.2.1 Design and Structure

1. Use a single instance of a SQL DB first to handle data. When we acquire more customers, scalability will be simple.
    - **Database Sharding**
        - **Requirements:** The system must support a growing number of users and concurrent bookings without performance degradation, which is critically important during peak times.
        - **Scalability:** Implement database sharding (dividing a large database into smaller, more manageable pieces called "shards", with each shard stored on a separate database server) across multiple databases.
            - **Ex:** User / customer information [data] can be sharded based on their user ID ranges. (UserID)
            - **Ex:** Booking data can be sharded based on theater location. (TheaterID)
        - These can help distribute the load and allow horizontal scaling.
    - **Read Replicas for Read-Heavy Operations**
        - **Requirements:** The system needs to efficiently handle constant and recurrent read operations, such as displaying movie listings, their showtimes, and seat availability for each showroom, without impacting the overall performance of the write operations (booking tickets).
        - **Scalability:** To optimize the read performance and guarantee responsiveness during peak times' demand, the system shall utilize read replicas for the main SQL database. This offloads read queries to separate instances, which allows the primary database to focus on the write operations. Read-heavy operations, including retrieving movie information, their showtimes, and seat availability, will be directed to the respective replicas. This will help minimize the load on the primary database responsible for transactional operations such as booking tickets.
        - By distributing the extensive amount of read requests across multiple replicas, this can help free the primary database to focus on critical, time-sensitive transactions such as latency issues, booking tickets, and prevent bottlenecking, when one component of a system hinders and even limits the overall processing capacity or speed of the system that can result in delays or even crashes during peak demand.
    - **Strategic Indexing and Query Optimization**
        - **Requirements:** Any query related to searching for movies, their respective showtimes in multiple showrooms, finding available seats, and retrieving user history.
        - **Scalability:** The system shall implement strategic indexing on columns frequently accessed in search and retrieval operations (Ex: MovieName, Showtime, UserID, BookingID). In addition, all

SQL queries will be optimized for performance, utilizing tools such as proper JOIN conditions, without needing to deal with unnecessary subqueries and leveraging covering indexes where appropriate to guarantee efficient data retrieval at scale.

- These all help to ensure that common user actions, searching for movies and showtimes, seat selection, and booking tickets, happen quickly even if/when the number of concurrent users increase.
- Strategic indexing helps the system quickly locate specific records without needing to scan the entire table, while the SQL query optimization helps ensure that when the system is combining data from multiple table sources, such as connecting a user's movie booking with a specific movie and respective showtime, that the process is efficient, resulting in minimizing the use of resources.
- The main goal in regard to faster searching and retrieval is to make them almost instantaneous. In doing so, the system can find and process the information immediately which can help in reducing the potential issue of seat overbooking/double-booking. When users try to book tickets for a newly released movie, the system needs to quickly determine which seats are available (temporary seat locking being established efficiently and quickly) and which seats are already purchased/taken.

Optimized queries and indexing help guarantee that these checks happen fast, which can prevent any delays or conflicts between concurrent users.

- **Connection Pooling**
  - **Requirement:** The system needs to efficiently manage database connections to handle a large number of multiple user requests without potentially overwhelming the database server.
  - **Scalability:** Instead of a slow and resource-heavy dependency process of establishing a new connecting for each ticket booking or search query, the system will implement connecting pooling to reuse existing database connections, reducing the overhead of establishing new connections for each request.
    - For an incoming request, the system will grab an available connection from the pool, utilize it, and then return it back to the collective pool for the next request.
    - This allows the system to effectively manage and reuse database connections that help conserve database resources without needing to drastically increase the database server's capacity.

2. The database will have main tables for some classes that are used in the software system including:
   - User table to store the user's information, encrypted password, payment
   - Movie table stores such as name, genre, rating, duration, price
   - Theater table to store the name of the theater, address, and possible email address or phone number
   - Showroom table which has room number, linked with the correct movie and theater table
   - Seats table which is linked to the showroom, shows the seat number and status
   - Ticket table which uses information from all the above tables, as well as price, transaction ID
   - Payment table, which is linked to all the above tables, has all the information needed to display a complete transaction to a user.

3. We will be using 3NF (Third Normal Form), to ensure that data is stored only once. This helps prevent the same movie from being stored in the Showroom, Ticket, and Payment table. Instead, only one instance will be stored on the first table and can be accessed through relationships between data.
4. Third normal form will keep a neat and modular storage dynamic, as if something is changed it won't affect other parts of the system or will be dynamically included.
5. We will implement security systems to help protect and encrypt user data such as
    - Password hashing to ensure users' passwords are safe.
    - SQL injection protection such as input validation
    - Data encryption for personal information such as payment, billing address, name, etc.
        - This must be maintained for both:
            - In transit (Being transmitted across networks)
            - At rest (stored in a database or server)
    - We will implement quick daily backups, as well as full weekly backups to ensure the least amount of data is lost
        - **Back Up & Recovery Strategies (In case of data loss or system failure)**
            - **Full Backups**
            - **Differential Backups**
            - **Transactional Log Backups**
            - **Disaster Recovery Plan**
                - Step by step plan that documents and outlines each process for restoring the database and associated systems.
        - **Data Retention Policies**
            - **Financial Records**
            - **Movie Listings**
            - **Customer Data (Non-Financial)**
            - **System & Audit Logs**
            - **Marketing & Promotional Data**
        - Data that is not actively used but needs to be kept for future reference should be moved to a separate, cost-effective archive storage system.
        - Data that is no longer needed for any business or legal reason should be securely and properly deleted.
        - **Regulatory Compliance:** Must adhere to regulations and standards for data protection such as CCPA (California state law) and PCI DSS (Global Security standard for organizations handling credit card data) when defining retention periods for customer data.
        - **Policy Enforcement:** Must create a clear schedule incorporating assigned responsibilities for data retention and deletion activities.
    - **Access Control:** Role-based access control upon the status of the user, (Admin, Manager, Developer, User)
        - Admin can manage movie listings, showtimes, theater room updates, monitor reports, modify user accounts/profiles.
        - Users can only manage their own ticket bookings and account profile/settings.

    - **Monitoring and Tuning**
        - **Performance Monitoring**
            - Continuously monitor database performance metrics (CPU Usage, Input & Output, Query Execution times) to identify and address potential bottlenecks.

8.2.2 Alternative Database Approaches

- We could have used a non-relational DB, such as MongoDB, which would've had some upsides and downsides compared to SQL
    - Easy to add information to using nested documents in MongoDB
    - Faster than SQL when data is embedded and optimized properly
    - Handles high amounts of traffic, as well as easy horizontal scaling
    - Not a relational DB, so normalization must be implemented by the user, so relationships can be much harder to manage
    - Not as many ways to handle data as SQL, as the language isn't as mature or extensive
- Another possibility is to use multiple Databases, one to handle relationships and the other to handle data. We felt as if this wasn't necessary because SQL can process data, normalize data, and create relationships between data much more efficiently.
- SQL ensures relationships and integrity but is harder to scale compared to a DB such as Mongo. By choosing SQL we prioritize consistency, efficiency, and relationships between data rather than scalability, which is great for our developing system.

A. Appendices

Appendices may be used to provide additional (and hopefully helpful) information. If present, the Software Requirements
Specification
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.