WBB
Development

Software Requirements Specification

V. 1.0.0.1

November 6, 2025

Group 19
Brian Hebert, Will Jordan, Blake Harper
(https://github.com/wjordan9794/CS-250-Group-19)

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

Revision History

Date	Description	Author	Comments
9/25/25	V. 1.0.0.1	William Jordan	First Revision
9/25/25	V. 1.0.0.1	Brian Hebert	First Revision
9/25/25	V. 1.0.0.1	Blake Harper	First Revision

Document Approval

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

Signature	Printed Name	Title	Date
WILLIAM J	William Jordan	Software Eng.	11/06/2025
	Dr. Gus Hanna	Instructor, CS 250	
Brian H	Brian Hebert	Software Eng.	11/06/2025
Blake Harper	Blake Harper	Software Eng.	11/06/2025

Table of Contents

1. Introduction

The purpose of this document is to develop an in-depth overview and insight of the complete details regarding the WBB Theater Ticket application by defining all of its aspects in detail. It also expands on its expected benefits to both WBB Theaters and the users of the WBB Theater Ticket application. The detailed requirements of the WBB Theater Ticket application are provided in this document.

1.1 Purpose

The purpose of this document is to build and outline the bounds of a movie ticketing application. This document will also develop the application's intended use scenario, as well as the necessary software and hardware requirements, user interface, and expected users.

1.2 Scope

The software product produced will be titled as WBB Theater Ticket application. This software is intended to allow customers of WBB Theater Ticket application to purchase tickets to movies from their personal device(s). The primary goal of the WBB Theater Ticket application is to mitigate lines and wait times in the lobby of the WBB Theater and develop a convenient and easy-to-use application for customers looking to purchase movie tickets. The app will have access to a database updated daily with all currently playing movies and showtimes.

1.3 Definitions, Acronyms, and Abbreviations

NO CURRENT ACRONYMS or ABBREVIATIONS

1.4 References

- Movie Theater Client Questionnaire: Theater Ticketing System Qs.docx
- IEEE SRS Standard (1998)
- Use Cases Lecture 4

1.5 Overview

The rest of this document contains an overall description of the application, including the expected users of this project, the necessary hardware for running the application, and the functional and data requirements of the product. A general description of the project can be found in Section 2 of this document, while section 3 will go into detail on the specific requirements of the GENERIC MOVIE TICKETING APP. Other parameters, such as functional requirements, data requirements, application perspective from the user, and constraints and assumptions made while designing the GENERIC MOVIE TICKETING APP, can also be found in Section 3.

2. General Description

This section provides an overall description of our movie ticketing system. This section describes the context in which it operates, who will use it, as well as broad constraints and assumptions that shape the detailed requirements that follow.

2.1 Product Perspective

The movie ticketing system is a new, customer-facing web application that sits within the theater's existing ticketing ecosystem. It acts as a client to a theater backend that exposes movie listings, showtimes, seat maps, pricing, and order processing.

- ***System context:***
    - ***Client layer:*** Responsive web browser
    - ***Application layer (server):*** RESTful services for authentication, catalog (movies/showtimes), seat selection, orders, and payments.
    - ***Data layer:*** Theater operational database with daily updates to movie schedules and pricing; session data for carts and reservations.
    - ***External services (planned):*** Payment gateway (tokenized card processing) and email/SMS provider for confirmations
- ***Relationship to other systems:*** Integrates with the theater's POS/box office to keep seat inventory synchronized and to honor redemptions at on-premise scanners.

2.2 Product Functions

At a highly functioned level, this system will enable customers to:

- ***Browse & search*** currently playing and upcoming titles by date, time, rating, and popularity.
- ***View details*** for a selected movie (synopsis, runtime, rating) and **view showtimes** by date
- ***Select seats*** on an interactive seat map with real-time availability and pricing tiers.
- ***Create and manage an order/cart*** including ticket quantities, concessions, taxes, fees, and promotional codes.

- ***Checkout & pay*** via secure payment (card wallet/tokenized card on file when available).
- ***Receive confirmation*** (on-screen and via email/SMS) with QR/Barcode for entry.
- ***Manage account*** basics: sign up/in, view order history/receipts, re-send confirmations.
- ***Error & recovery paths:*** handle timeouts, seat conflicts, and payment declines with clear messaging and retry options.

2.3 User Characteristics

Primary user groups and their relevant characteristics:

- ***Moviegoers (General Audience):***
    - Ages 13+; broad range of technical proficiency.
    - Expect fast, intuitive flows; minimal required inputs; clear pricing; accessible design that is aligned with Web Content Accessibility Guidelines (WCAG).
- ***Registered Members (Returning Users):***
    - Comfortable with saving preferences (favorite movie genre, payment method).
    - Expect faster checkout, access to order history, promotions, and loyalty programs.
- ***Theater Staff (Indirect Users):***
    - Interact with the system via POS and QR/Barcode scanners; require reliable, scannable confirmations and real-time seat status.

2.4 General Constraints

Key constraints that influence design and implementation:

- ***Platforms & browsers:*** Must function on web browsers such as Chrome, Safari, and Firefox
- ***Performance:*** Pages should load in ≤2 seconds on typical web networks; seat selection and availability checks should respond in ≤1 second under normal load.

- ***Security & compliance:*** PCI-DSS for payment processing (via vetted gateway)
- ***Data freshness:*** Movie listings and showtimes updated at least daily; near real-time seat inventory during the purchase flow.
- ***Availability:*** Target 99.5% uptime during theater operating hours; graceful degradation if external services are unavailable.
- ***Accessibility:*** WCAG 2.1 Level AA to meet technical standard
- ***Scalability:*** Handle peak traffic spikes around prime showtimes and opening weekends.

2.5 Assumptions and Dependencies

Assumptions:

- A reliable ***payment gateway*** is available with tokenization and fraud checks.
- The theater provides an ***API or data feed*** for movies, showtimes, pricing, and seat maps; service-level expectations meet the performance targets above.
- Users have ***network connectivity*** sufficient for secure transactions and receipt retrieval.
- Email/SMS channels can deliver confirmations within 1–2 minutes of purchase.

Dependencies:

- ***Third-party services*** (payment, messaging) and their SLAs.
- ***Theater backend/POS integration*** for seat inventory synchronization and redemption.
- ***Content assets*** (posters, ratings, synopses) supplied by the theater or licensed feeds.
- ***Compliance policies*** (security, privacy, refunds) defined by the theater and applicable regulations.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces (gui)

There will be a few different pages on this website. There will be a main home page which displays popular showings. At the top right of the home page there will be a sign in/up button. At the top center of the home page there will be a search bar to search for specific movies playing. Upon searching for a movie, the titles being displayed on the home page will switch to relevant results. Each page will display the top 10 most relevant movie titles. After selecting a movie, the user will be brought to a page with the movie cover on the left and the showtimes on the right. After selecting a showtime, the user will be brought to a payment screen.

3.1.2 Hardware Interfaces (which device)

Users will be able to access the website from Windows and Mac, as well as iPhone and Android devices.

3.1.3 Software Interfaces (browser)

Supported browsers will be Chrome, Safari and Firefox.

3.1.4 Communications Interfaces (communication between database, client and server, which browsers does this support)

Upon loading the webpage, the client will send a request to the server. The server will access the database to gather necessary information (i.e. top movies currently playing). This information will be sent back to the client and displayed. Once the user selects a movie title, another request will be sent to the server. This request will return showtimes and prices for the movie. When the user selects a showtime, a request will be sent to the server. This time it will be to reserve the seat that the user chose. The server will return the request with the information for the payment screen. When the user inputs their payment information, it will be encrypted and sent to the correct bank to verify the payment. After verifying the purchase, the server sends a copy of the ticket and order confirmation to the client.

3.2 Functional Requirements

3.2.1 Return Currently Playing Movies, Showtimes, and Ticket Pricing to the User

3.2.1.1 Introduction: Upon providing input for what theater the user intends to visit, return and display currently playing movie showtimes pertinent to the user's desired theater.
3.2.1.2 Inputs: The user will be prompted to enter their zip code and choose from a list of theaters sorted by proximity to the user. (cc info, name, address, theater location, movie name and showtime)
3.2.1.3 Processing: The application will send a request to the server's database for the currently playing movies specifically at the user's chosen theater. The server will respond and send the requested movies, showtimes, and ticket pricing to the application for display to the user.
3.2.1.4 Outputs: Once the application has received the requested information, it will display the currently playing movies in a list format, with pertinent showtimes and ticket pricing displayed next to each movie entry.

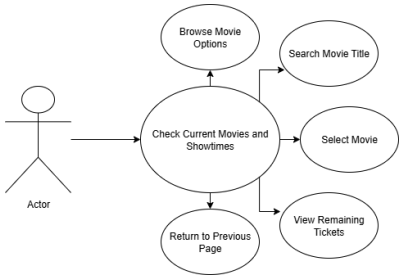
3.2.1.5 Error Handling: If the application is able to reach the server, it will return an error message to the user and provide contact information for customer support.

3.2.2 User Purchase and Acquisition of Movie Tickets

3.2.2.1 Introduction: Upon selecting a movie and associated ticket(s), prompt the user to pay for the movie ticket(s) and send them the ticket(s) and a receipt.
3.2.2.2 Inputs: The user will have already entered their location to see the currently playing movies in their proximity. Once they have selected a showtime and ticket type, they will be prompted to enter a payment method to pay for the tickets. (cc info, name, address, theater location, movie name and showtime)
3.2.2.3 Processing: The application will send a request to a third party API to process the payment (Visa, Mastercard, Paypal, Discover, Amex). Once the API confirms that the payment was successful, the application will send a request to update the remaining tickets on the database, and then display a QR code with an order number. A receipt will also be generated and sent to an email address, if it was provided.
3.2.2.4 Outputs: The QR code and order number will be displayed on the user's application. They will have an option to print or save their order to their digital wallet. They will also receive a receipt sent to their provided email address.
3.2.2.5 Error Handling: If the payment is unsuccessful, the application will NOT prompt the server to update the remaining tickets and will instead ask the user to try the purchase again. If the application is able to reach the server, or the payment fails a second time, it will return an error message to the user and provide contact information for customer support.

3.3 Use Cases

3.3.1 Check Current movies and showtimes



A. Name: Check Current movies and showtimes

B. Brief description: This use case will allow the actor to check what titles are currently being shown at the theater.

C. Actors: Actors will be anyone who wants to see what movies are playing at the theater. Restriction on who can and cannot access this use case.

D. Basic flow
   I. user either scrolls on home page to view suggested titles or searches for a title using The search bar.
   II. Users can then click on the movie title of their choice.
   III. The website displays the showtimes and prices for the title selected.

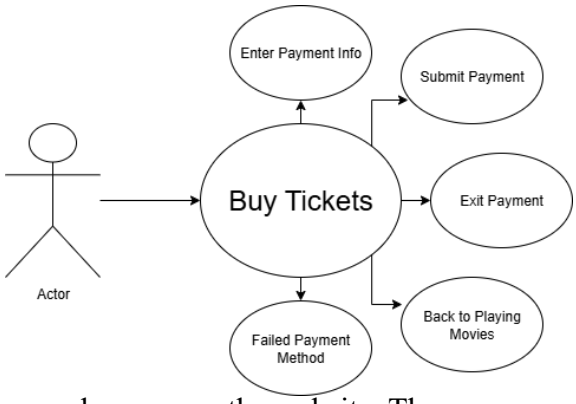
E. Alternate flows
   I. The user goes back to the homepage before selecting a title to check showtimes.

F. Pre-conditions
   I. The movie title is correctly entered.

G. Post-conditions
   I. Showtimes have been displayed or the user returned to the homepage.

H. Special requirements
   I. The website should display how many more tickets are available for the selected showtime.

3.3.2 Purchase Tickets



A. Name: buy tickets

B. Brief description: This use case will allow the actor to purchase or refund a ticket for any open showtime.

C. Actors: The actors of this use case will be anyone who goes on the website. There are no requirements whatsoever to be able to use this. Anyone who wants to buy movie tickets can use this use case.

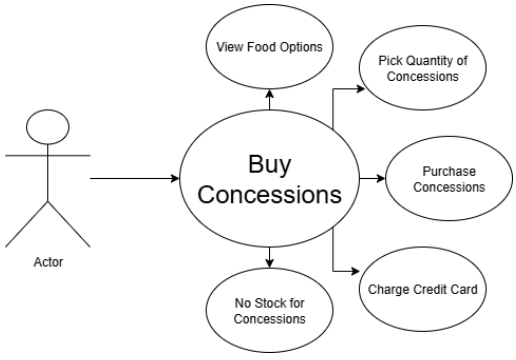
D. Basic flow
   I. After selecting a showtime for a movie, the user is brought to a purchase page.
   II. The user is asked for credit card information.
   III. The website sends a request with that information to the correct bank to withdraw the money from that account.
   IV. After verifying the validity of the purchase, a receipt and ticket are sent to the user

E. Alternate flows
   I. Payment information is incorrect, the user will be brought back to the payment info page
   II. The user exits the payment screen, which will bring them back to the showtime page.

F. Pre-conditions
   I. There are empty seats for that showing.

G. Post-conditions
   I. The user cancels or completes the purchase.

3.3.3 Buy Concessions



A. Name: buy concessions

B. Brief description: This use case will allow the actor to buy concessions for the movie

they are going to watch.

C. Actors: The actors for this use case will be limited to those who have already purchased tickets.

D. Basic flow
   I. Displays all choices for snacks and candies to the user
   II. User can change the number from a zero to something else below each snack to indicate the number of that item they want.
   III. The same card that was used to purchase the ticket will be charged for the concessions.

E. Alternate flows
   I. If an item is out of stock, it will display a message reading “We’re sorry but this item is currently out of stock.”

F. Pre-conditions
   I. Item is in stock.

G. Post-conditions
   I. Concession is purchased.
   II. Transaction is cancelled.

H. Special requirements
   I. deals on concessions are displayed
   II. limited time concessions are also displayed

3.4 Classes / Objects

3.4.1 Theater

The first class will be a class that represents individual theater rooms.

3.4.1.1 Attributes

Attributes for this class will be ticket prices, theater room number and number of seats for theater.

3.4.1.2 Functions

- Buy tickets
- Refund tickets
- Check number of open seats for theater

3.4.2 Concessions

The second class will represent the concession stand.

3.4.2.1 Attributes

Attributes for this class will be a list of available concessions and a list of their quantities.

3.4.2.2 Functions

I. List available concessions
II. Buy concessions
III. Request new concession item

3.5 Non-Functional Requirements

3.5.1 Performance

Users should not be left waiting for more than 30 seconds on any page.

3.5.2 Reliability

There should not be more than one issue every month with the reliability of ticket and concession purchasing.

3.5.3 Availability

This website should be available to anyone who would like to use it and will be available on major devices and browsers (Apple, Android, PC, Mac and Linux)

3.5.4 Security

Credit card information will be encrypted before it is transferred from client to server to protect user information. Credit card information will not be stored on the website. Instead we will rely on browser technology to save and fill in credit card information.

3.5.5 Maintainability

Developers will work to maintain this website weekly and make the changes necessary to keep it running.

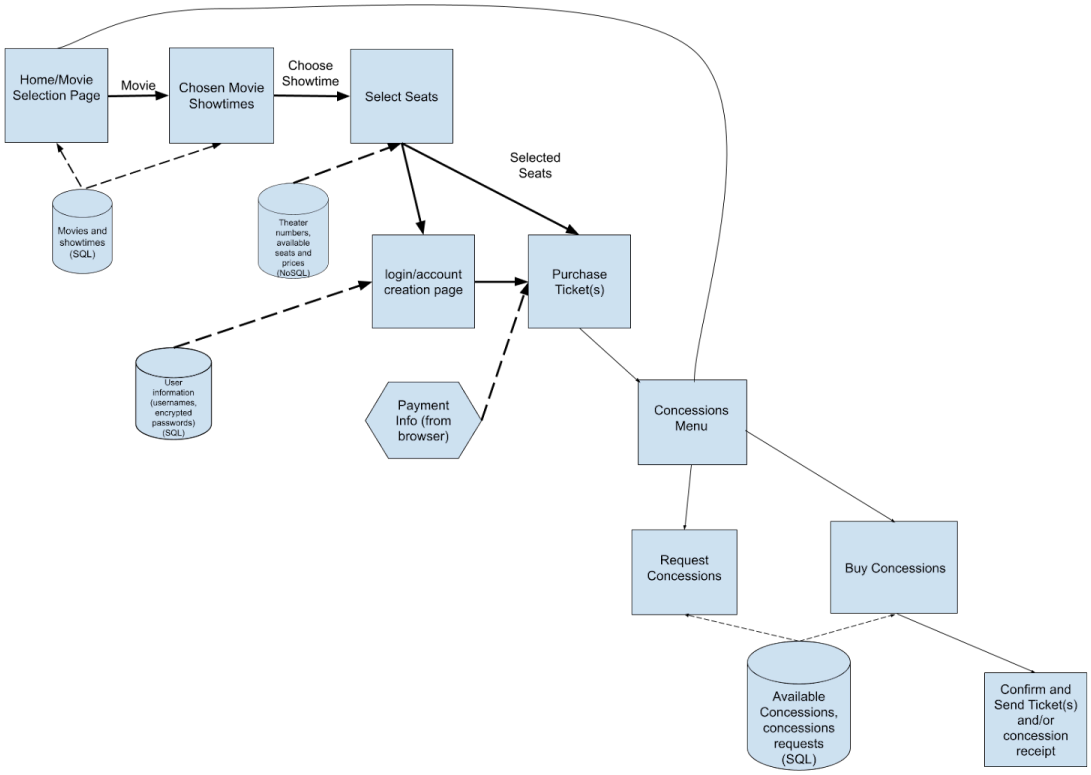
3.5.6 Portability

If the device can run any of the specified browsers, it will be able to run this website.

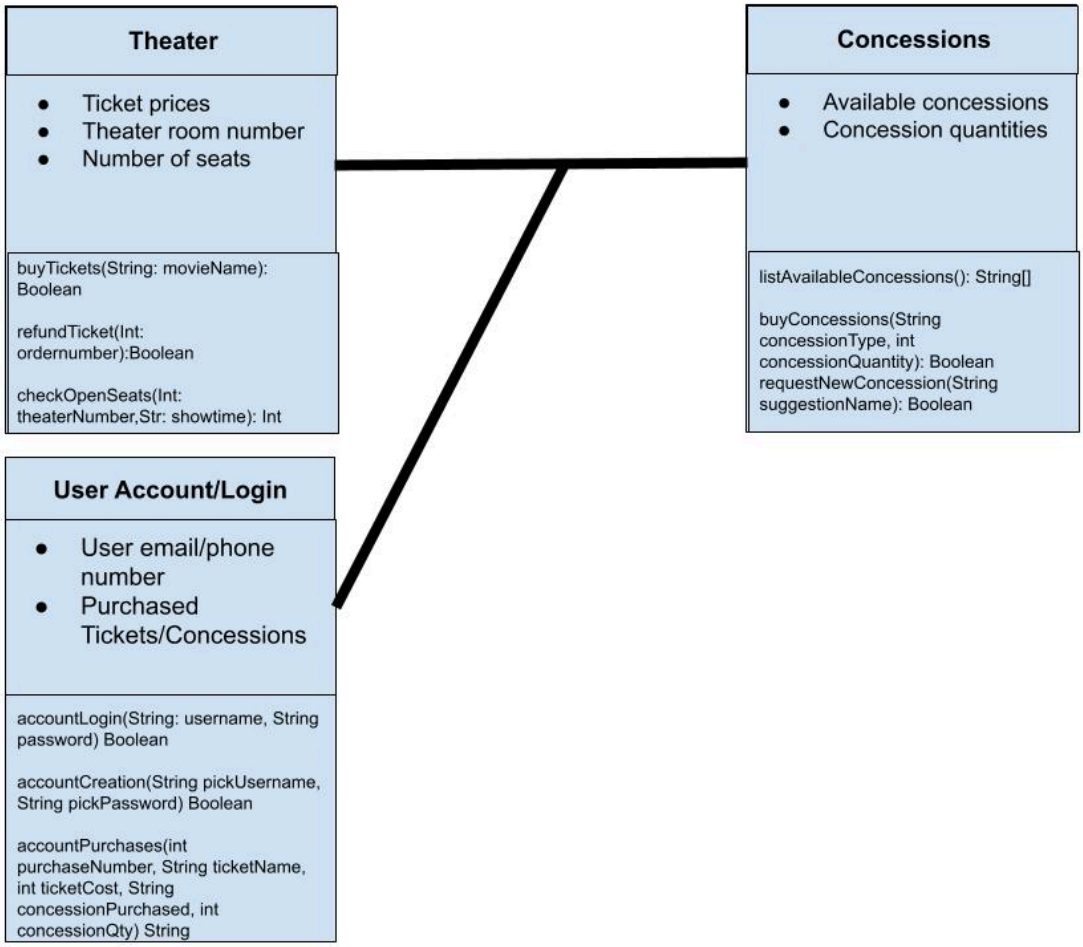
3.6 Inverse Requirements

*State any *useful* inverse requirements.*

3.7 Design Constraints



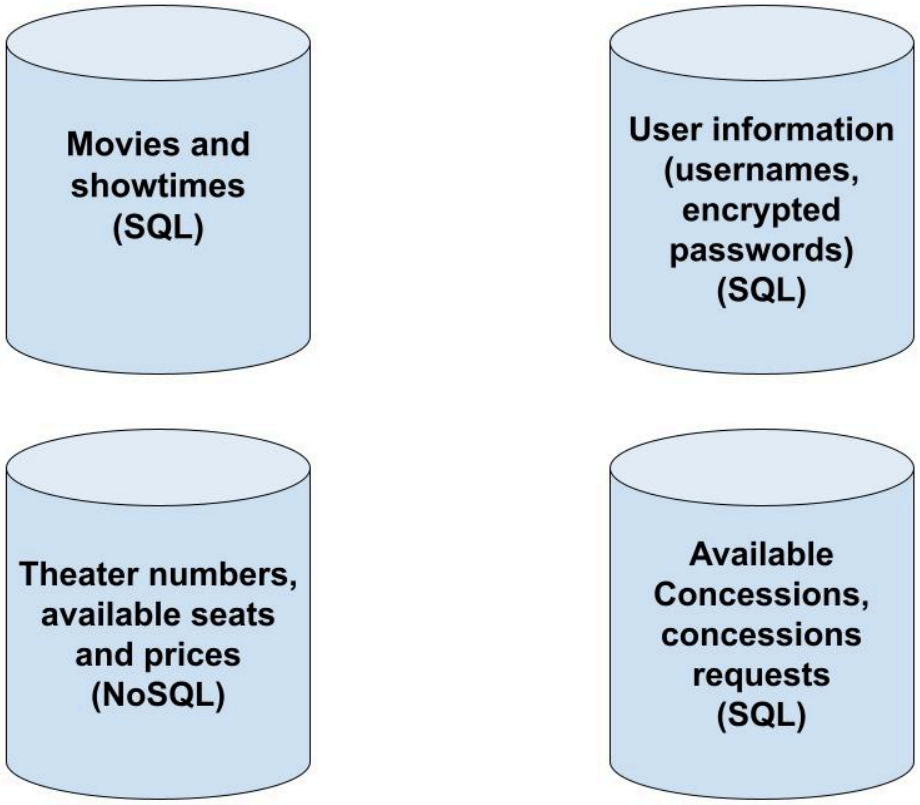
SWA Diagram:
This diagram illustrates the end-to-end user flow for the WBB Theater ticketing app and how the UI pages interact with core data stores. A user begins at the Home/Movie Selection Page, which queries the Movies & Showtimes database to display titles. After choosing a title, they move to Chosen Movie Showtimes, then select a specific showtime and proceed to Select Seats. The seat-selection page performs real-time lookups against the Theater/Seats/Pricing store to show availability and prices. From there, the user either signs in on the Login Page (validated against the User Information store of usernames and encrypted passwords) or continues if already authenticated. The selected seats are passed into Purchase Ticket(s), where Payment Info (from the browser) is submitted and processed. On success, the flow ends at Confirm and Send Ticket(s), which issues the receipt/QR ticket to the user.



UML Diagram:
This diagram shows three classes: Theater, Concessions, and User Account and Login. They interact and operate within the same ticketing system. Theater encapsulates operational details (ticket prices, theater room number, and number of seats) and exposes methods to buyTickets(movieName): Boolean, refundTicket(orderNumber): Boolean, and checkOpenSeats(theaterNumber, showtime): Int. Next, concessions models the concessions operations, with attributes for available items and their quantities, and methods to listAvailableConcessions(): String[], buyConcessions(concessionType, concessionQuantity): Boolean, and requestNewConcession(suggestionName): Boolean. Finally, User Account contains functions to store account login parameters, account creation parameters, and recent and previous ticket and concession purchases. Together, the classes capture core behaviors for purchasing tickets, checking seat availability, managing concession sales, and retaining user purchases.

3.8 Logical Database Requirements

Primary Database Diagram:



Data Management Strategy:

It makes the most sense to use a SQL database for our user account and payment database in order to maintain data integrity and database consistency with transparent data encryption (TDE). A SQL database would also be a realistic choice for our database to store our currently playing movies and showtimes, since we wouldn't need to store dynamic data.

Similarly, a SQL database would also be utilized for our concessions database, since it wouldn't need to be updated frequently. We would then use a noSQL database for our available theaters and seats. This is because noSQL databases can scale horizontally to better maintain consistent application reaction times during high traffic hours, when more people are purchasing tickets. This will allow customer satisfaction with our application to be satisfactory as well.

These 4 separate databases allow for us to update our databases on a singular basis; we don't need to consider how changes to one database affect the others. Similarly, it allows us to ensure that we don't put undue load on one database during peak or busy periods of time.

Another option that would help when we are in the testing phase of our application is using SQLite. This allows us to run it locally without needing to set up a server. Doing this would let us test if our system works as intended. Of course, the tradeoff is our databases' inability to handle our application during high traffic times, until we upsize or outsource to dedicated servers. For initial testing and preliminary use as well, it would be permissible to use noSQL databases for all of our necessary databases, since they are more flexible in nature and don't require relational database consistency.

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

4.3 Data Flow Diagrams (DFD)

4.2 State-Transition Diagrams (STD)

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. Test Plan

1. Objective
   Verify that the movie ticketing system works as intended.
2. Scope
   - Movie browsing and search
   - Showtime and seat selection
   - User account creation and login
   - Ticket purchase (including payments and concessions)
   - Email ticket and receipt delivery
   - UI behavior and layout across devices
   - Performance and security basics
3. Functional Tests
   - Search movie by title, genre, or time
   - Select available showtime and seat(s)
   - Prevent purchasing tickets for already sold seats

- Add tickets and concessions to cart
- Complete checkout with valid card
- Handle payment declines gracefully
- Confirm purchase and send ticket email
- Allow login/logout and session persistence

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

Preliminary Development Plan

Will Jordan, Brian Hebert, and Blake Harper committed to this project.

Section 1: Will Jordan

Section 2: Blake Harper

Section 3: Brian Hebert and Will Jordan

UML Diagram, SWA Diagram, and writeup: All 3 Members

A.2 Appendix 2

Database Outside Sources/References

https://www.mongodb.com/resources/basics/databases/nosql-explained/nosql-vs-sql
https://www.geeksforgeeks.org/sql/sql-data-encryption/
https://www.integrate.io/blog/the-sql-vs-nosql-difference/

A.3 Appendix 3

Test Cases: **Will Brian Blake 10 Test Cases**