# Movie Theater Reservation System

## Software Requirements Specification

## 1.0.9

## May 27. 2024

Group 4
Nolan Lee, Akil Parrish, Grant Mauger

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

## Revision History

| Date | Description | Author | Comments |
|---|---|---|---|
| 5/24 | 1.0.0 | Nolan Lee | 1.1, 1.2, 1.3 |
| 5/26 | 1.0.1 | Akil Parrish | 3.3 |
| 5/27 | 1.0.2 | Nolan Lee | 3.5 |
| 5/27 | 1.0.3 | Grant Mauger | 3.2 |
| 5/27 | 1.0.4 | Akil Parrish | Revised 3.3, 1.5 |
| 5/27 | 1.0.5 | Grant Mauger | 3.1, 3.4 |
| 5/27 | 1.0.6 | Nolan Lee | 2.1, 2.2, 2.3, 2.4, 2.5 |
| 6/10 | 1.0.7 | Nolan Lee | 6.2 |
| 6/10 | 1.0.8 | Akil Parrish | 6.1 |
| 6/10 | 1.0.9 | Grant Mauger | 6.2 |
| | | | |
| | | | |

## Document Approval

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

| Signature | Printed Name | Title | Date |
|---|---|---|---|
| | Akil Parrish | Software Eng. | |
| | Nolan Lee | Software Eng. | |
| | Grant Mauger | Software Eng. | |
| | Dr. Gus Hanna | Instructor, CS 250 | |

# Table of Contents

# 1. Introduction

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

[ TODO: after majority of sections are completed ]

## 1.1 Purpose

This SRS is the primary tool for communication between business leaders, software developers, maintenance teams, and clients. It outlines the specific manner in which the application is to be designed, developed, and tested as it gets ready for deployment. It will see constant revisions, as the requirements for completion dynamically change with client interests and software/budget limitations.

## 1.2 Scope

The software products to be produced will include a database and its communication to the main system, a credit card payment API, a website with a user interface for both customers and managers, and any security measures taken to ensure information privacy.

The database will contain:

- User login info, such as first and last name, email address, phone number (optionally), and a 10-character-minimum password with at least one capital and one numerical.
    - Users can also choose to save payment information to the database, in which case 2FA will be prompted to set up.
- A list of supported theaters, entered in by site admins.
- A list of current movies with their available showings, entered in by theater managers.
- A 2D array of available seats for every showing, gradually filled by user entries.

The credit card API will gather information from the user and use that information to request a transaction to VISA / MasterCard / PayPal, and depending on the output, inform the user of the success or failure of the transaction.
The website will allow customers to browse movie titles using a search engine that, when granted access, will use the user's location to determine the closest possible showing for the chosen movie, in both location and time. Once selected, a grid of available seats will be shown and the user will choose their desired seating. Finally, the user will enter payment information (or select if previously entered and saved), and a confirmation message will appear with their receipt.

The website will also allow theater managers to fully customize which movies and showings appear on the website for customer selection. They can also blacklist specific users if they so choose, and they can also create special offers and sales that will be advertised to the user.

User info, passwords, 2FA codes, and credit card info will be encrypted at rest immediately after entry and before being transmitted to the database.

## 1.3 Definitions, Acronyms, and Abbreviations

API : Application Programming Interface - How our system will communicate with outside interfaces like credit card applications.
2FA: Two Factor Authentication - Extra security measures, like a secure code sent to email or phone number, to decrease hacking and phishing attempts.

## 1.4 References

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

## 1.5 Overview

1. Overall Description
    - Product perspective
    - Product functions
    - User characteristics
    - Constraints
    - Assumptions and dependencies
2. Specific Requirements
    - Functional requirements
    - Performance requirements
    - Design constraints
    - Software system attributes
    - External interface requirements
3. Supporting Information
    - Table of contents
    - Index
    - Appendices

The SRS is organized to flow logically from a general overview to specific details. It begins with an introduction that sets the stage for the document, followed by an overall description that provides context for the system. The specific requirements section is the core of the document,

detailing the requirements for the system. Supporting information includes any additional material that supports the understanding of the SRS.

# 2. General Description

## 2.1 Product Perspective

This product aims to replicate the experience of popular reservation applications such as Fandango, while stripping down the unnecessary junk and optimizing client and administrator needs. It will offer a simplistic and ergonomic UI to help guide users to complete their task as quickly and painlessly as possible.

## 2.2 Product Functions

This product will have an encrypted login system, allowing both normal users and movie theater administrators to enter a username and password to gain access to the system. Users will be able to search for movies by name or recency, and after selection, they can choose to either reserve single or group seating. A simple credit card interface will allow the customer to instantly reserve their desired seats. Administrators are also able to manage available movies and showings, and are able to determine which users are allowed to reserve.

## 2.3 User Characteristics

The users of the program will most likely have an interest in any of the most recent movies to hit theaters, from seeing them in advertisements to word of mouth from friends or family. Most movie-goers do not enjoy going alone, so a group reservation system will help ensure that their friends and family are seated adjacent to them for optimal enjoyment. Both users and administrators likely want to make reservations or adjustments on-the-go, so website support for mobile devices is crucial.

## 2.4 General Constraints

The searching of movies by title should first show the most popular movies (driven by data from recent purchases). Below the title and description, there should be a list of showings in the next twenty-four hours that are able to be clicked on to be taken to the reservation page. There, a grid of seats, constrained by theater size given by admins, are shown. The seats are numbered and lettered by row and column, and they are grayed-out if already reserved. After selection of one or multiple seats, the user is prompted to enter their credit card or PayPal information, and the reservation is completed.

## 2.5 Assumptions and Dependencies

We can safely assume that a large majority of users will either be using Windows or Mac for desktop use, or either iOS or Android for mobile use. We can also assume that the database will be operated with either Linux or Windows, and communications between database and client will be through the cloud.

# 3. Specific Requirements

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

The system will provide a web-based interface accessible through popular web browsers (e.g., Chrome, Firefox, Safari, Edge).

### 3.1.2 Hardware Interfaces

The system will interface with internet connectivity hardware to enable communication between the web-based application and the server. Examples include the modem, the router, and other hardware to facilitate an internet connection.

### 3.1.3 Software Interfaces

The system will integrate with third-party payment gateways (e.g., PayPal, Stripe) to process payments securely. It will handle payment confirmations or failures, including declined transactions and invalid payment details.

### 3.1.4 Communications Interfaces

The system will require internet connectivity for accessing the web-based interface and third-party services. Data transmission between the client (web browser) and the server will use HTTP/HTTPS protocols. Error messages will be displayed for connectivity issues, server timeouts, and failed data transmissions.

## 3.2 Functional Requirements

### 3.2.1 User Login and Authentication

3.2.1.1 Introduction

The system shall provide a secure login and authentication mechanism for users to access the reservation system.

3.2.1.2 Inputs

- Username
- Password

3.2.1.3 Processing

- Validate the username and password against the stored credentials.
- Establish a session for the authenticated user.

3.2.1.4 Outputs

- Successful login: User is granted access to the reservation system.
- Failed login: User is prompted to re-enter their credentials.

3.2.1.5 Error Handling

- Incorrect credentials: Display an error message indicating incorrect username or password.
- Account lockout: After a predefined number of unsuccessful login attempts, lock the account and prompt the user to contact support.

### 3.2.2 Movie Selection

3.2.2.1 Introduction

The system shall allow users to select a movie from a list of currently showing movies.

3.2.2.2 Inputs

- User selection of a movie (movie ID or title)

3.2.2.3 Processing

- Retrieve the list of currently showing movies from the database.
- Filter movies based on user preferences or search criteria.

3.2.2.4 Outputs

- Display a list of currently showing movies along with details such as showtimes, ratings, and genres.

3.2.2.5 Error Handling

- No movies available: Display a message indicating that there are no movies available for the selected criteria.

### 3.2.3 Show Time Selection

3.2.3.1 Introduction

The system shall allow users to select a showtime for the chosen movie.

3.2.3.2 Inputs

- User selection of a showtime (date and time)

3.2.3.3 Processing

- Retrieve available showtimes for the selected movie from the database.
- Check availability for the selected showtime.

3.2.3.4 Outputs

- Display available showtimes for the selected movie.
- Confirm the selected showtime.

3.2.3.5 Error Handling

- No showtimes available: Display a message indicating that there are no available showtimes for the selected movie.

## 3.3 Use Cases

### 3.3.1 Use Case #1

**Use Case: Online Seat Selection**

**Purpose**:

The Online Seat Selection use case allows customers to choose specific seats for a movie when making a reservation through the theater's website.

**Actors**:

1. - Customer: The person who wants to reserve seats online.
2. - System: Represents the movie reservation system.

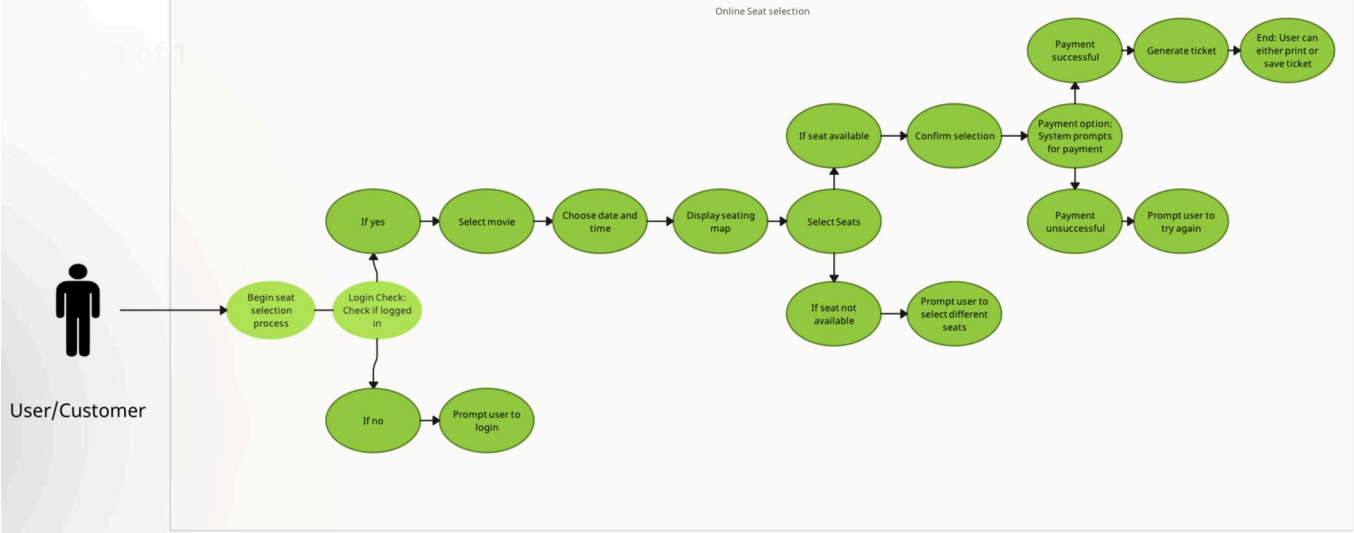
**Flow of Events**:

1. The Customer visits the theater's website and selects a movie they want to watch.
2. The System displays information about the selected movie, including showtimes, available seats, and ticket prices.
3. The Customer clicks on the "Select Seats" button.
4. The System presents a seating layout (usually in the form of a grid) with available seats marked as selectable.
5. The Customer clicks on specific seats to choose them. The selected seats are highlighted.

6. The System validates seat availability in real-time:
    - If the selected seats are available:
        - The System reserves the chosen seats for the Customer.
        - The System calculates the total cost based on the ticket price and the number of selected seats.
        - The System generates a reservation confirmation with a unique reservation ID.
    - If the selected seats are not available (already reserved or unavailable due to social distancing, etc.):
        - The System informs the Customer that the seats are no longer available.
        - The Customer can choose different seats or cancel the reservation.
7. The Customer provides payment information (credit card, etc.) to confirm the reservation.
8. The System processes the payment and sends a confirmation email or SMS to the Customer.

**Assumptions** (Entry Conditions):

- The Customer has an active internet connection.
- The movie schedule and seat availability are up-to-date.
- The payment gateway is functional.




### 3.3.2 Use Case #2

**Use case: Group Reservations and Special Accommodations**

**Purpose**:
The system can cater to groups, such as school trips or corporate events, allowing a bulk reservation of seats. It can also handle special requests for accommodations, such as wheelchair-accessible seating or assistive listening devices for the hearing impaired. This use case demonstrates the system's flexibility in managing diverse customer needs and enhancing the movie-going experience for all patrons.

**Actors:**

1. Customer: Represents individuals who want to reserve movie tickets.
2. Group Organizer: Represents someone organizing a group reservation (e.g., for a school trip or corporate event).
3. System: Represents the movie theater reservation system.

**Flow of Events:**

1. Search Movie:
    - Purpose: Allows Customers and Group Organizers to search for movies by title, genre, or date.
    - Flow of Events:
        - The Customer or Group Organizer enters search criteria (e.g., movie title, date).
        - The System retrieves matching movies.
        - The results are displayed to the user.
    - Assumptions:
        - The movie database is up-to-date.
        - The search functionality is available.
2. View Movie Details:
    - Purpose: Provides information about a selected movie, including cast, plot, and reviews.
    - Flow of Events:
        - The Customer or Group Organizer selects a movie from the search results.
        - The System displays detailed information about the movie.
    - Assumptions:
        - Movie details are available in the system.
3. Select Seats:
    - Purpose: Enables Customers and Group Organizers to choose specific seats for a movie.
    - Flow of Events:
        - The Customer or Group Organizer selects a movie and showtime.
        - The System presents an interactive seating layout.
        - The user clicks on specific seats to choose them.
    - Assumptions:
        - Seat availability is real-time.
        - The user is logged in.
4. Make Reservation:
    - Purpose:
        - Customer:
            - Reserves seats for an individual or a small group.
            - Selects the type of ticket (e.g., standard, VIP, child).
        - Group Organizer:
            - Reserves a block of seats for a larger group.

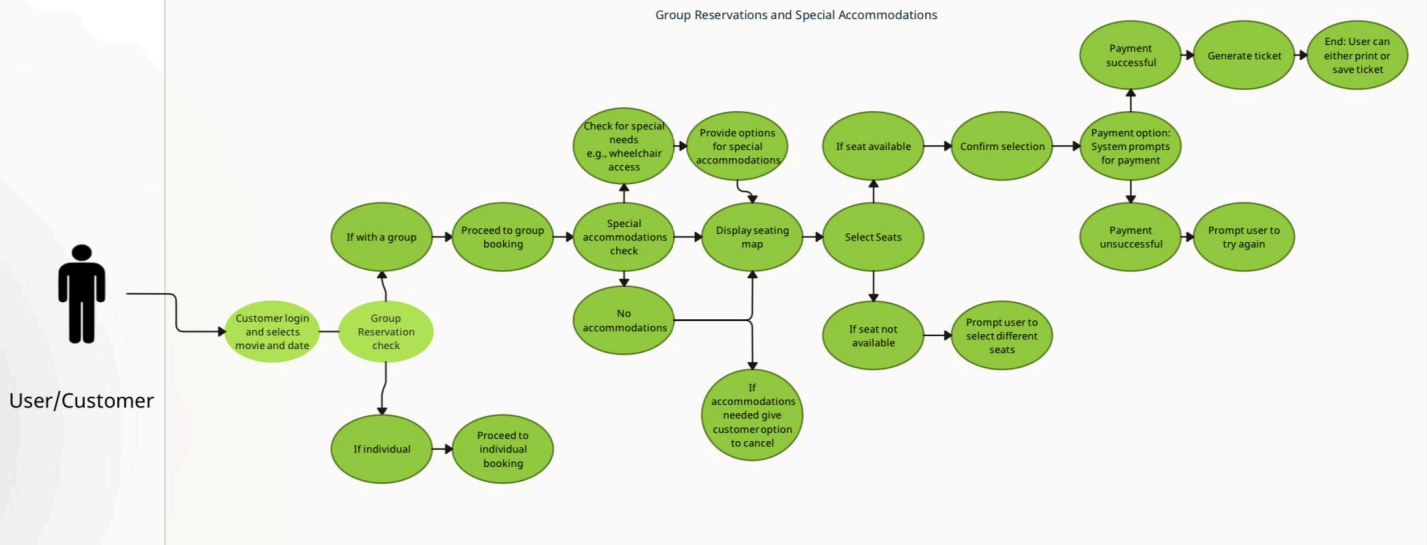
        - Specifies special accommodations (e.g., wheelchair-accessible seats, hearing-impaired assistance).
    - Flow of Events:
        - The user selects seats (individual or group).
        - The System validates seat availability.
        - If seats are available, the reservation is confirmed.
    - Assumptions:
        - Payment information is provided.
        - Special accommodations are handled.
5. Cancel Reservation:
    - Purpose: Allows users to cancel a booked reservation and receive a refund (subject to cancellation policy).
    - Flow of Events:
        - The user selects a reservation to cancel.
        - The System processes the cancellation.
    - Assumptions:
        - Cancellation policies are defined.
6. View Booking History:
    - Purpose: Displays the history of bookings made by a user.
    - Flow of Events:
        - The user requests their booking history.
        - The System retrieves and displays past reservations.
    - Assumptions:
        - User authentication is in place.
7. Manage Movie Collection (Admin):
    - Purpose:
        - Adds, updates, or removes movies from the system.
        - Generates sales reports to track movie performance.
    - Flow of Events:
        - The admin logs in.
        - The admin performs movie management tasks.
    - Assumptions:
        - Admin privileges are required.

**Relationships:**

- **Customer** and **Group Organizer** interact with the **System** for various use cases.
- **Group Organizer** interacts specifically with the **Make Reservation** use case for group bookings.

**Special Accommodations:**

- The **Make Reservation** use case allows specifying special accommodations during seat selection (e.g., accessible seats, companion seats).




### 3.3.3 Use Case #3

**Use Case Diagram: Movie Theater Reservation System (Admin)**

**Purpose:** The purpose of the admins in a movie theater reservation system is multifaceted and central to the operation of the theater. Admins are essential for the smooth running of the theater, from updating the movie schedule to ensuring customer satisfaction. Their role supports both the operational needs of the theater and the service quality experienced by the customers.

**Actors:**

1. Theater Administrator: Represents an admin responsible for managing the theater operations.
2. System: Represents the movie theater reservation system.

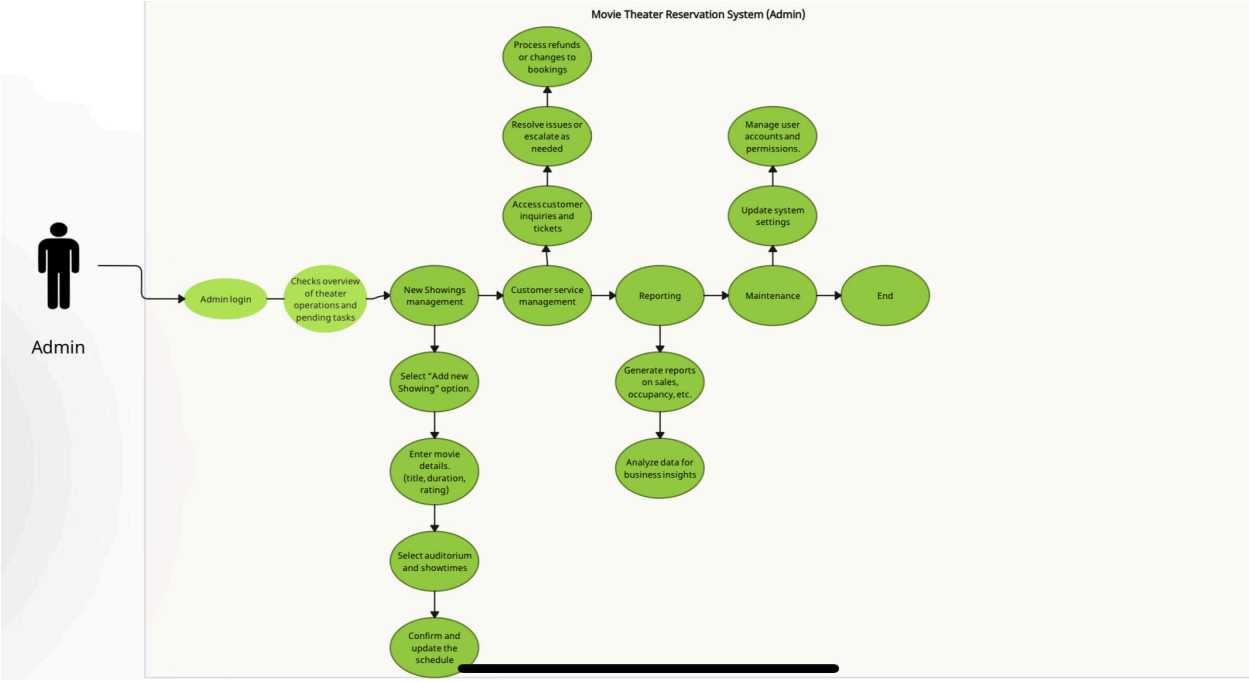
**Flow of Events:**

1. Add New Movie Showing:
    - Purpose: Allows administrators to add new movie showings to the system.
    - Flow of Events:
        1. The Theater Administrator logs in to the admin panel.
        2. The admin selects the option to add a new movie showing.
        3. The System prompts the admin to enter details:
            - Movie title
            - Showtime (date and time)
            - Theater hall or screen number
            - Ticket prices
        4. The admin submits the information.
        5. The System updates the movie schedule.
2. Update Movie Showing:

    - Purpose: Allows administrators to modify existing movie showings.
    - Flow of Events:
        1. The Theater Administrator selects an existing movie showing.
        2. The admin edits relevant details (e.g., showtime, ticket prices).
        3. The admin saves the changes.
        4. The System updates the movie schedule.
3. Manage Special Accommodations:
    - Purpose: Enables administrators to handle special accommodations for customers (e.g., wheelchair-accessible seats, hearing-impaired assistance).
    - Flow of Events:
        1. The Theater Administrator receives a request for special accommodations.
        2. The admin reviews the request (e.g., wheelchair seating, closed captioning).
        3. The admin assigns appropriate seats or services.
        4. The System records the accommodations.
4. Handle Customer Service Requests:
    - Purpose: Allows administrators to address customer inquiries, complaints, or issues.
    - Flow of Events:
        1. The Theater Administrator receives a customer service request (via phone, email, or in person).
        2. The admin listens to the customer's concern (e.g., ticket refund, lost item, technical issue).
        3. The admin provides assistance or escalates the issue to the relevant department (e.g., technical support, ticketing).
        4. The System logs the interaction for reference.

**Assumptions (Entry Conditions):**

- The Theater Administrator has appropriate permissions to access the admin panel.
- The movie schedule and seat availability are up-to-date.
- Customer service channels (phone, email, in-person) are operational.




## 3.4 Classes / Objects

### 3.4.1 User

3.4.1.1 Attributes

- userID (int): Unique identifier for the user.
- username (string): The user's chosen login name.
- password (string): The user's password for authentication (stored securely).
- email (string): The user's email address.
- phoneNumber (string): The user's contact phone number.
- userType (string): Type of user (e.g., customer, staff, administrator).

3.4.1.2 Functions

- register(): Allows a new user to create an account.
- login(): Authenticates the user and starts a session.
- updateProfile(): Enables the user to update their personal information.
- logout(): Ends the user's session.
- resetPassword(): Initiates the password reset process.

### 3.4.2 Movie

3.4.2.1 Attributes

- movieID (int): Unique identifier for the movie.

- title (string): Title of the movie.
- genre (string): Genre of the movie (e.g., action, comedy, drama).
- rating (string): Movie rating (e.g., PG, R).
- duration (int): Duration of the movie in minutes.
- description (string): A brief description of the movie.
- cast (string): List of main actors/actresses.
- director (string): Director of the movie.
- releaseDate (date): Release date of the movie.

3.4.2.2 Functions

- addMovie(): Adds a new movie to the system.
- updateMovie(): Updates details of an existing movie.
- deleteMovie(): Removes a movie from the system.
- getMovieDetails(): Retrieves detailed information about a specific movie.

### 3.4.3 Seat

3.4.3.1 Attributes

- seatID (int): Unique identifier for the seat.
- theaterID (int): The ID of the theater where the seat is located.
- rowNumber (string): The row in which the seat is located.
- seatNumber (string): The specific number of the seat.
- isAvailable (boolean): Indicates if the seat is available for booking.

3.4.3.2 Functions

- checkAvailability(): Checks if the seat is available for a specific showtime.
- reserveSeat(): Reserves the seat for a user.
- releaseSeat(): Releases the seat, making it available for other users.
- getSeatDetails(): Retrieves details about the seat.

### 3.4.4 Reservation

3.4.4.1 Attributes

- reservationID (int): Unique identifier for the reservation.
- userID (int): The ID of the user who made the reservation.
- showtimeID (int): The ID of the showtime being reserved.
- seatIDs (list<int>): List of IDs of seats reserved.
- paymentStatus (string): Status of the payment (e.g., pending, completed).
- reservationDate (date): The date the reservation was made.

3.4.4.2 Functions

- createReservation(): Creates a new reservation.
- updateReservation(): Updates details of an existing reservation.
- cancelReservation(): Cancels an existing reservation.
- getReservationDetails(): Retrieves details about a specific reservation.

## 3.5 Non-Functional Requirements

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

### 3.5.1 Performance

Communications between database and website instance must not exceed 5 seconds.
Load times for website pages must be less than 3 seconds on standard internet speed.

### 3.5.2 Reliability

System downtime must not exceed 5 minutes per day, unless by scheduled maintenance.
No glaring or debilitating bugs should be visible to the user or admin.
Navigation of the system must be intuitive and simple, with options included to accommodate certain guest or theater limitations.

### 3.5.3 Availability

System must be available for both user and admin all day, every day (unless for above maintenance).
System downtime must not exceed 2 hours at a time.

### 3.5.4 Security

Customers who save payment information, and administrators, must use 2 Factor Authentication.
All customer data must be encrypted at rest before sent to the database.

### 3.5.5 Maintainability

Customer support phone lines must always be available during business hours of the theater.
Customer support emails and messages will be answered within 1 business day.
If a critical failure occurs, a patch or update to fix the problem must be rolled out within 2 hours of notice.

#### 3.5.6 Portability

The web application is able to be used on both desktop and mobile devices, and all information is displayed appropriately.
The website system must be functional across different device manufacturers and operating systems, such as Windows, Mac, and Linux.
The system will also support many web browsers, including Edge, Chrome, and Firefox.

### 3.6 Inverse Requirements

*State any *useful* inverse requirements.*

### 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

*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. Testing

### 6.1 Test Plan

1. Feature Identification

First, identify all the features of the Movie Theater Reservation System, such as:

- User account creation and login
- Movie selection
- Seat selection
- Payment processing
- Reservation confirmation
- Error handling and system recovery

2. Test Set/Vector Creation

For each feature, create test sets/vectors that cover:

- Normal operation (happy path)
- Boundary conditions
- Error conditions
- Security vulnerabilities

3. Test Description and Execution

User Account Creation and Login

- Test: Attempt to create accounts with valid and invalid details.
- Vectors: Valid email/password, invalid email, weak password, SQL injection.
- Coverage: Ensures account creation is secure and rejects invalid inputs.

Movie Selection

- Test: Select various movies from different genres and times.
- Vectors: Popular movies, less-known movies, future dates.
- Coverage: Checks if the system can handle diverse selections without errors.

Seat Selection

- Test: Choose seats including edge cases like first/last row and column.
- Vectors: Single seat, adjacent seats, full row, seats near the screen.
- Coverage: Verifies the seat map is interactive and updates availability correctly.

Payment Processing

- Test: Process payments using different methods and simulate transaction failures.
- Vectors: Credit card, PayPal, insufficient funds, network error.
- Coverage: Confirms payment gateway integration and error handling.

Reservation Confirmation

- Test: Verify that the system sends a confirmation with correct details.
- Vectors: Email confirmation, SMS notification, incorrect email.
- Coverage: Ensures the user receives accurate reservation details.

Error Handling and System Recovery

- Test: Introduce faults and observe system behavior.
- Vectors: Database disconnection, server downtime, client-side errors.
- Coverage: Tests the system's resilience and recovery mechanisms.

4. Design Diagram Modification

Modify the system design diagram to highlight the components involved in each test. Use different colors or annotations to indicate the target and scope of the tests, and the types of failures covered.

5. Result Analysis

After executing the tests, analyze the results to identify any failures or weaknesses in the system. Document these findings and suggest improvements or fixes.

6. Security Testing

This involves performing security testing to ensure that sensitive data, like user information and payment details, are handled securely and in compliance with relevant standards. This could involve testing the system's resistance to common web attacks, such as Cross-Site Scripting (XSS) and SQL Injection, and ensuring that the system complies with data privacy regulations like GDPR and CCPA.

**GitHub**:

*https://github.com/grantmauger15/CS250_group_4*

## 6.2 Test Cases

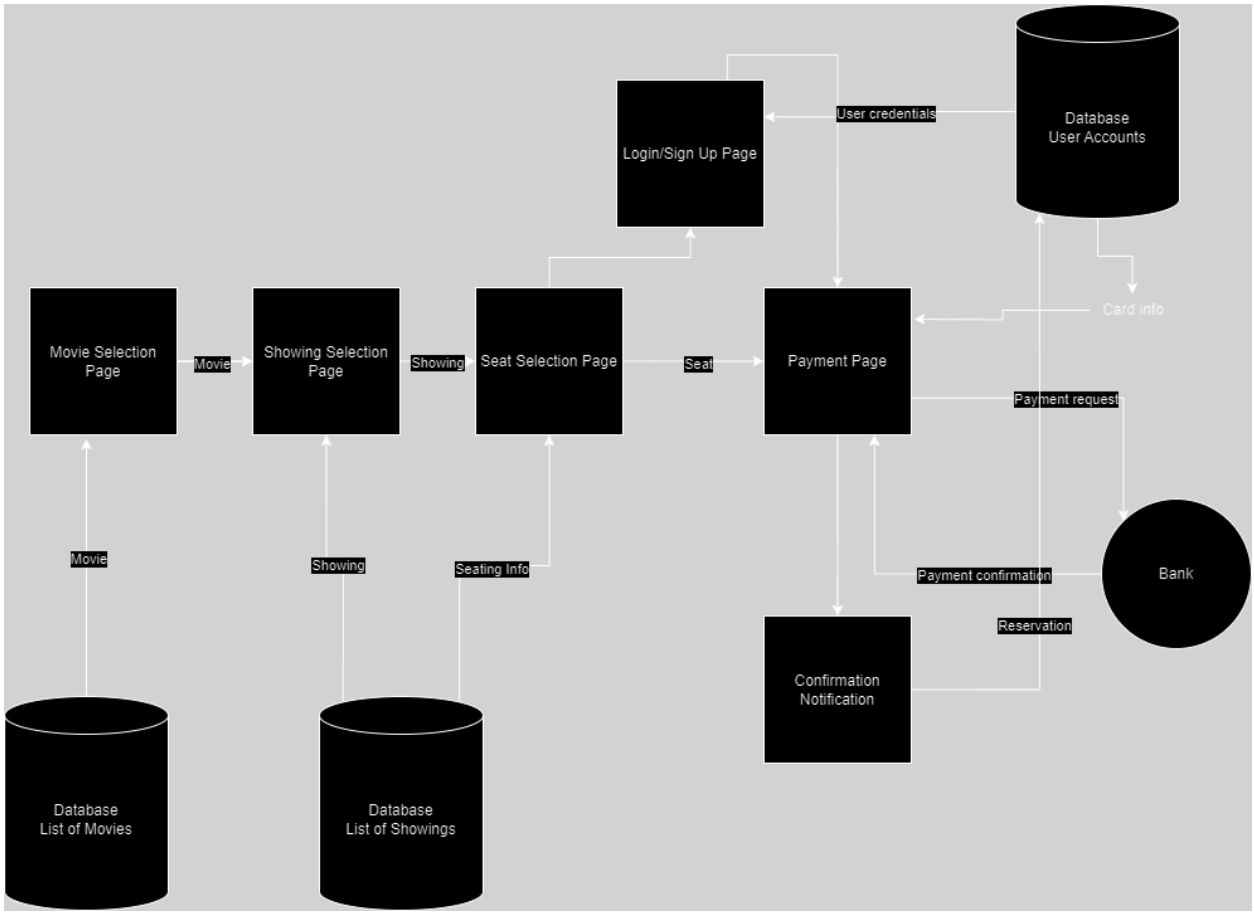
| Test Case ID | Component | Priority | Description | Prerequisites | Test Steps | Expected Result | Actual Result | Status |
|---|---|---|---|---|---|---|---|---|
| Search_1 | Search_Bar | P1<br>*Unit* | Verify that when the user types a keyword and submits, the results include movies related to that keyword | Website is launched and user clicks on the search bar or icon | 1. Launch the website from URL or google search<br>2. Select search bar and type 'Godzilla'<br>3. Press enter or click search icon | A list of theaters that support the movie display with showing times underneath | The list appears, and the theaters shown are sorted by distance from approximate location of user given their IP address. | Pass |
| Seating_1 | Seating_ Vector | P2<br>*Function* | Verify that once a seat is selected and reserved, it can no longer be selected by another user | Website is launched and user selects a showing and a seat | 1. Select a seat at a given showing<br>2. Complete the reservation<br>3. Under the same showing and a different user, attempt to reserve that same seat | The seat should display as grayed out and should give an error message when trying to select it | The seat is grayed out and an error message appears at the top of the page, instructing the user to select another seat | Pass |
| Payment_1 | Payment_ Gateway | P1<br>*Function* | Ensure that when a user reserves a seat using payment information, that the total | Seat is selected and user has entered in credit card information or PayPal | 1. Select a showing and a seat<br>2. Enter payment information or log into PayPal<br>3. Submit | A payment receipt generated by the credit card API or PayPal should show that the payment was | A receipt is sent by either service which confirms that the transaction was successful | Pass |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | is deducted from their account and added to the business's | | | complete and the money has entered the business account | | |
| Payment_2 | Payment_ Gateway | P2<br>*Unit* | Verify that the user must enter payment information in the correct manner | Seat is selected and user is now on the payment information page | 1. Seat selected<br>2. Enter a non-16 digit number for CCN<br>3. Enter a non-3 digit number for CVV<br>3. Enter a non-date for the expiry date | All three incorrectly filled fields should be highlighted red and the error type should be displayed above | All three boxes are highlighted red, and the correct error message was displayed | Pass |
| Signup_1 | Login_ Signup | P2<br>*Unit* | Verify that the user must enter accurate information upon signup and that the email is registered. | Website is launched with no prior logins saved | 1. Select sign up on main page<br>2. Enter special characters and numbers under Name<br>3. Enter an invalid email address<br>4. Enter an invalid phone number | Application should deny any attempts to submit with the invalid information, and error messages should display above the correct fields. | User is unable to sign up, and the invalid fields are outlined in red with an error message above explaining how to solve the issue | Pass |
| Login_1 | Login_ Signup | P1<br>*Unit* | Verify that the login information is stored on the database and the password is correct | Website is launched and the Login button is pressed on the main page | 1. Select log in on main page<br>2. Enter incorrect login information and attempt to log in<br>3. Enter correct information and attempt again | User should first be denied by the system with a displayed error message, and then the login should be successful with correct info | An error message displays at the top of the screen prompting a reset of password, and then the user logs in and is taken to the search page | Pass |
| Login_2 | Login_ Signup | P2<br>*Function* | Verify that the password reset function works properly in communicating with email services | Website is launched, and the Reset Password button is pressed on the Login page. | 1. Select log in on main page<br>2. Press the Reset Password button on login page<br>3. Look for reset password email | The user should be redirected to a page telling them to check their email for a link that will let them reset their password | The user is successfully redirected to the correct page and receives a reset password link in their email. | Pass |
| Showings_1 | Showings_ Page | P1<br>*Function* | Verify that once all seats for a showing are reserved, the showing is removed from the list of a movie's available showings. | Website is launched and the user has selected a movie and is now on the showings page. | 1. Reserve the last seat for a showing of a given movie<br>2. On a different user, select the same movie as in step 1<br>3. Check if the showing from step 1 is still available | The showing that has all seats reserved should be grayed out and users should be unable to reserve any of its seats. | The showing is grayed out and the user is unable to interact with it in any way, leaving them to choose a different showing. | Pass |
| Reservation_ | Full System | P1 | Verify the full | The website is | 1. Select a movie by | The user should be | The user was able | Pass |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| With_Card | | *System* | end-to-end process of selecting a movie, creating an account, reserving a showtime and seat, and using a credit card for payment. | launched and the user is on the movie selection page. | scrolling through a list and/or using the search bar.<br>2. Select an available showtime from a list of showtimes in your area.<br>3. Select a seat that is available and not grayed out.<br>4. Sign up for an account and provide the appropriate credentials.<br>5. Pay for your reservation using a valid credit card. | able to go through the full process of selecting a movie and a showtime, reserving a seat as well as creating their account and completing the payment with a valid credit card. | to complete the full process of creating a reservation and paying for it with a credit card. | |
| Reservation_ With_PayPal | Full System | P1<br>*System* | Verify the full end-to-end process of logging into an account, selecting a movie, reserving a showtime and seat, and using PayPal for payment. | The website is launched and the user is on the login page. | 1. Log into your account using valid user credentials.<br>2. Navigate to the movie selection page and select a movie.<br>3. Select a showtime from a list of showtimes in your area.<br>4. Select a seat that is available and not grayed out.<br>5. On the payment page, select PayPal and be redirected to PayPal to fulfill payment and complete the reservation. | The user should be able to go through the full system process of logging into their account, selecting a movie, selecting a showtime and seat, and finally using PayPal to fulfill their payment and complete the reservation. | The user was able to complete the process of creating a reservation and paying for it using PayPal's services. | Pass |

*All tests performed by TesterA and TesterB*

# 7. Data Management Strategy




- To handle large amounts of data flow properly, there should exist three different databases holding different categories of data. Two of the databases have no sensitive info, the list of Movies and the list of Showings, and thus little encryption or protection is needed. While these two categories could be merged into one database for ease of development, the Showings database will likely have orders of magnitude more pull and push requests than the Movies database. Movies also gives much more information per request (description, cast, ratings, etc.) whereas Showings has just the time and the boolean vector for seating arrangement. Splitting up the two allows for more streamlined fetching and higher availability for both databases.
- All databases will be built and maintained using SQL. This language is highly standardized and readable, helping the development team with potentially diverse backgrounds to collaborate efficiently. This database structure is also very secure, offering real-time monitoring of the database from admins, as well as maintaining regular backups and user authentications. SQL does have some scalability and monetary cost concerns, however the data we are dealing with will likely not come close to reaching the upper limits of SQL. The language also has some diversification problems, as it cannot handle complex data types, but for our purposes these issues can be ignored.

# 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