TEMPORARY PAGE

ReelEasy (Movie Theater Ticketing System)

Software Requirements Specification

Version 1.0

7/12/2026

Group 2
Bobby Bavongkhoun
Caleb Wolf
Gleb Rodin

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

Revision History

Date	Description	Author	Comments
7/15/2025	Version 1.0	Bavongkhoun, Wold, Rodin	Created to include sec. 3

Document Approval

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

Signature	Printed Name	Title	Date
	Bobby Bavongkhoun	Software Eng.	
	Caleb Wolf	Software Eng.	
	Gleb Rodin	Software Eng.	
	Dr. Gus Hanna	Instructor, CS 250	

Table of Contents

1. Introduction

This section provides a broad overview of the software requirements by detailing its purpose, intended audience, overall scope, and key terminology pertaining to the system.

1.1 Purpose

The purpose of this SRS is to define the functions and non-functional requirements for ReelEasy. It is an online ticketing system designed to support SDSU's School of Theatre, Television, and Film's movie theater operations. The system aims to provide a secure, easy-to-use and accessible platform for students, faculty, and community patrons to browse movie showtimes and select seats. This document is a guide for developers, designers, quality assurance teams, and project stakeholders who are responsible through the software development life cycle.

1.2 Scope

Demands for the use of SDSU's School of Theatre, Television, and Film facilities have created the need for a local ticketing system that is simple, secure, and accessible across multiple platforms. The software product to be developed is named ReelEasy. ReelEasy will enable students, faculty, staff, and community patrons to: (1) Manage digital ticket reservations and sales for movie screenings hosted by SDSU's School of Theatre, Television, and Film. (2) Enable customers purchase tickets and allow seat selection through web and mobile devices. (3) Provide electronic tickets for online purchases and support physical ticket printing for box sales. The ReelEasy system will NOT: (1) Process payments unrelated to ticketing or seating reservations to movie screenings. (2) Manage or process any food or beverage orders or payments. (3) Manage live performances or non-movie theater events.

ReelEasy system is designed to provide a secure, ease-of-use, cross-platform system for patrons to reserve and purchase tickets for movie screenings. Accommodate the unique dine-in movie theater environment by enabling seat reservations for dining sections. Efficiency of handling wait times through online ticket purchases only. Provide staffers with the tools needed to manage schedules, seating layouts, ticket availability, and generate attendance and sale reports. The strategic objective is to support SDSU's School of Theatre, Television, and Film's mission to deliver accessible, high-quality movie experiences in a modern and flexible facility. The system's scope is strictly limited to ticketing and seat reservations for movie screenings and excludes food and beverage services.

1.3 Definitions, Acronyms, and Abbreviations

This subsection provides the definitions of all terms, acronyms, and abbreviations utilized throughout this document.

ReelEasy	Online ticketing system developed for SDSU's School of Theatre, Television, and Film movie theater operations.
SDSU	San Diego State University
Dining Sections	Designated seating areas within the movie theater that provide dining services, coordinated separately by SDSU's Dining Services
Ticket	Electronic or physical ticket for proof of purchase to grant admission to a movie screening.
Patron	Any individual purchasing tickets for screenings through ReelEasy.

1.4 References

IEEE Recommended Practice for Software Requirements Specifications, IEEE Std 830-1998, 25 June 1998, IEEE-SA Standards Board
Your 2025 Guide to Writing a Software Requirements Specification – SRS Document, June 4, 2025, Relevant Software
Scenarios and Use Cases, CS 5150 Software Engineering, Cornell University Computing and Information Science

1.5 Overview

Section 2: Discusses everything that is taken into consideration before planning for the product itself and its design can begin. Section 2 includes relation to adjacent projects such as the dining services, functions it should be expected to perform, a description of the expected end user, constraints applicable to the development of this project, and conditions we can expect to be present for the product.

Section 3: Lists out specific requirements the product is meant to facilitate, use cases we expect the end user to have with the product, requirements for functionality, thresholds to maintain for the functions, and design constraints.

Section 4: Includes documentation of models and diagrams that were used to establish requirements and how we should expect the product to behave.

Section 5: Documents changes made to the SRS over time and how the SRS changing process should proceed.

2. General Description

Section 2 will not be going over requirements for the product, but instead will provide a broad look at the outside factors impacting how the product is being designed and a vague description of functionality.

2.1 Product Perspective

ReelEasy will include the usage of a third party database software for saving accounts, payment methods, schedules, and seating availability. ReelEasy handles the interface for customers to purchase tickets online for the SDSU's School of Theatre, Television, and Film's movie theater and its showings. Dining services are mentioned as a feature of the establishment, but are not handled by ReelEasy.

2.2 Product Functions

ReelEasy allows customers to purchase tickets for the shown available seats for that showing. Include a section of the website dedicated to providing information about what movies are being shown throughout the week, what specific times a movie is being played, and what section of the theater it is planned on being played in. Incorporate a section of the website that displays reviews for each movie that come from critics. And contain methods for theater staff to manually update the currently shown schedule as well as input the planned schedule for future showings.

2.3 User Characteristics

Customer:

This end user is expected to be either SDSU students or individuals adjacent to the SDSU institution. The end user will be interacting with ReelEasy via their web browser. Login information will not be required, but the customer can opt in to create an account. Customers are expected to be able to navigate web pages already.

Critic:

Audience members who have attended showings of a movie will be able to leave a review which will then be shown on the ReelEasy website. This critic will require an account to leave any reviews so that we can verify a ticket was bought for the movie being reviewed and to ensure safety of what is being shown on the website. A basic word processing textbox will be provided to the critic for them to write their review in.

Staff:

Staff members of the theater itself will have access to updating prices and scheduled showings. An account will be required for the staff member to make these changes. The staff member is expected to be comfortable with spreadsheets, as the scheduling interface will be similar to one.

2.4 General Constraints

ReelEasy is expected to operate on a personal computing platform with access to the internet and web-surfing applications such as Google Chrome, Microsoft Edge, Safari, and Mozilla Firefox. ReelEasy should use HTTPS, have secure account information storing methods, and role-based access for varying levels of authority to provide a sense of security. The system must also comply with SDSU IT security policies. The website should be responsive and allow for an ease of use for the end user. The website should be available at all times outside of scheduled maintenance times. User data should be stored according to data protection laws.

2.5 Assumptions and Dependencies

All users using the software will have access to internet-connected devices for online purchases. Furthermore, SDSU Dining Services will provide updated information on dining section availability to ensure seat reservation accuracy. The system will utilize and depend on SDSU's existing network infrastructure for hosting data security.

3. Specific Requirements

This section will highlight the specific requirements that ReelEasy will be required to meet.

3.1 External Interface Requirements

3.1.1 User Interfaces

3.1.1.1 The system interface is accessible from standard browsers.

3.1.1.2 The system has a responsive design that adjusts for mobile device screens.

3.1.1.3 The system requires account login to verify identity.

3.1.1.4 The system shall support different languages (English, Spanish, French, and Vietnamese)

3.1.1.5 The system permits users to search for movies using keywords, filters, and or categories.

3.1.1.6 The system allows for users to select preferred showtimes and available seats (including dining sections).

3.1.1.7 The system supports universal design and accessibility with seating arrangements for people with disabilities

3.1.1.8 The system has a box office interface for physical tickets.

3.1.1.9 The system allows for an administrative interface for management of the system.

3.1.1.10 The system admin interface shall add, update, and or remove movie listings and descriptions.

3.1.1.11 The system admin manages showtimes, seat maps, and dining reservations

3.1.1.12 The system permits the real time monitoring of ticket sales and seat availability.

3.1.2 Hardware Interfaces

3.1.2.1 The system is integrated with box office ticket printers for physical ticket printing.

3.1.2.2 The system is compatible with barcode or QR code scanners at the theater entrances.

3.1.3 Software Interfaces

3.1.3.1 The system is integrated with SDSU's CASHNET Payment portal for secure transaction processing.

3.1.3.1 The System receives dining section availability updates from SDSU's Dining Services

3.1.4 Communications Interfaces

3.1.4.1 The system utilizes HTTPS for secure data transmissions

3.1.4.2 The system uses email and SMS gateways for sending electronic tickets and confirmation of purchases.

3.2 Functional Requirements

This section defines the core operations and features that the system will support. These requirements will specify how the software will interact and behave in response to the user's input and system events to ensure it fulfills its intended purpose.

3.2.1 Search and Browse Movies

3.2.1.1 Introduction

This functionality shall allow patrons to search for available movie screenings and browse listings based on title, genre, and or keywords.

3.2.1.2 Inputs

The user can input keywords and or optional filters (i.e movie titles, genres, dates, time, rating)

3.2.1.3 Processing

The system shall validate input format (i.e no invalid characters).

Query the movie database for listings that match the search criteria.

Sort and filter results as requested.

3.2.1.4 Outputs

Display a list of matching movies with basic details (i.e title, showtimes, rating).

Allow the user to clock on a movie to view all the necessary information (description, duration, and available seats).

3.2.1.5 Error Handling

The system will display “No results found” when no results match the search.

The system shall display an input validation message if the input is invalid (unsupported characters).

3.2.2 Seat Reservation / Selection

3.2.2.1 Introduction

The system shall allow patrons to view available seats for a chosen movie and select preferred seats, including standard and dining section seats.

3.2.2.2 Inputs

The system shall take the user’s selected movie and showtime into consideration for selection

The app shall take the user’s desired seat into consideration for selection.

The system shall take into consideration the availability of the seat for selection.

3.2.2.3 Processing

The system shall retrieve a seat map for selected showtimes.

The system temporarily hold marked seats during checkout to prevent double booking or subsequent changes.

The system shall have a final reservation for the seat reservation upon successful payment.

3.2.2.4 Outputs

Display real-time seat mapping.

Highlight selected seats

Show final confirmation of selected seats

Show available seats after each confirmed reservation and transaction

3.2.2.5 Error Handling

System notify user of seat becomes unavailable during checkout, notify user to prompt for new selection.

The system shall output an error and retry option if data retrieval fails.

3.2.3 Payment Processing

3.2.3.1 Introduction

The system shall secure online payments for ticket purchases to ensure compliance with university and industry payment security standards.

3.2.3.2 Inputs

Payment details (credit/debit card, digital wallet, or other digital payment methods)

Ticket cost and user confirmation

3.2.3.3 Processing

Validate user payment information.

Communicate with SDSU CASHNET Payment portal

The system can authorize or decline transactions.

3.2.3.4 Outputs

Confirmation of successful payment and transaction ID.

Issue of electronic ticket

3.2.3.5 Error Handling

Notify the user of payment failures (i.e card declined)

System outputs clear instruction to retry to use an alternate payment method if payment fails.

3.2.4 Administrative Movie Management

3.2.4.1 Introduction

The system shall authorize staff to be able to add, update, or remove listings, showtimes, and seating configurations through a secure administrative interface.

3.2.4.2 Inputs

Movie details (i.e title, description, genre, duration)

Showtimes and theater building assignments.

Seat map configurations.

3.2.4.3 Processing

Validate data input for accuracy and completeness

Update movie listings database.

Sync seat maps with availability in real-time.

3.2.4.4 Outputs

Updated listings visible to patrons through search and browse feature.

Confirmation of successful changes

Administrative reports that summarizes movie schedules.

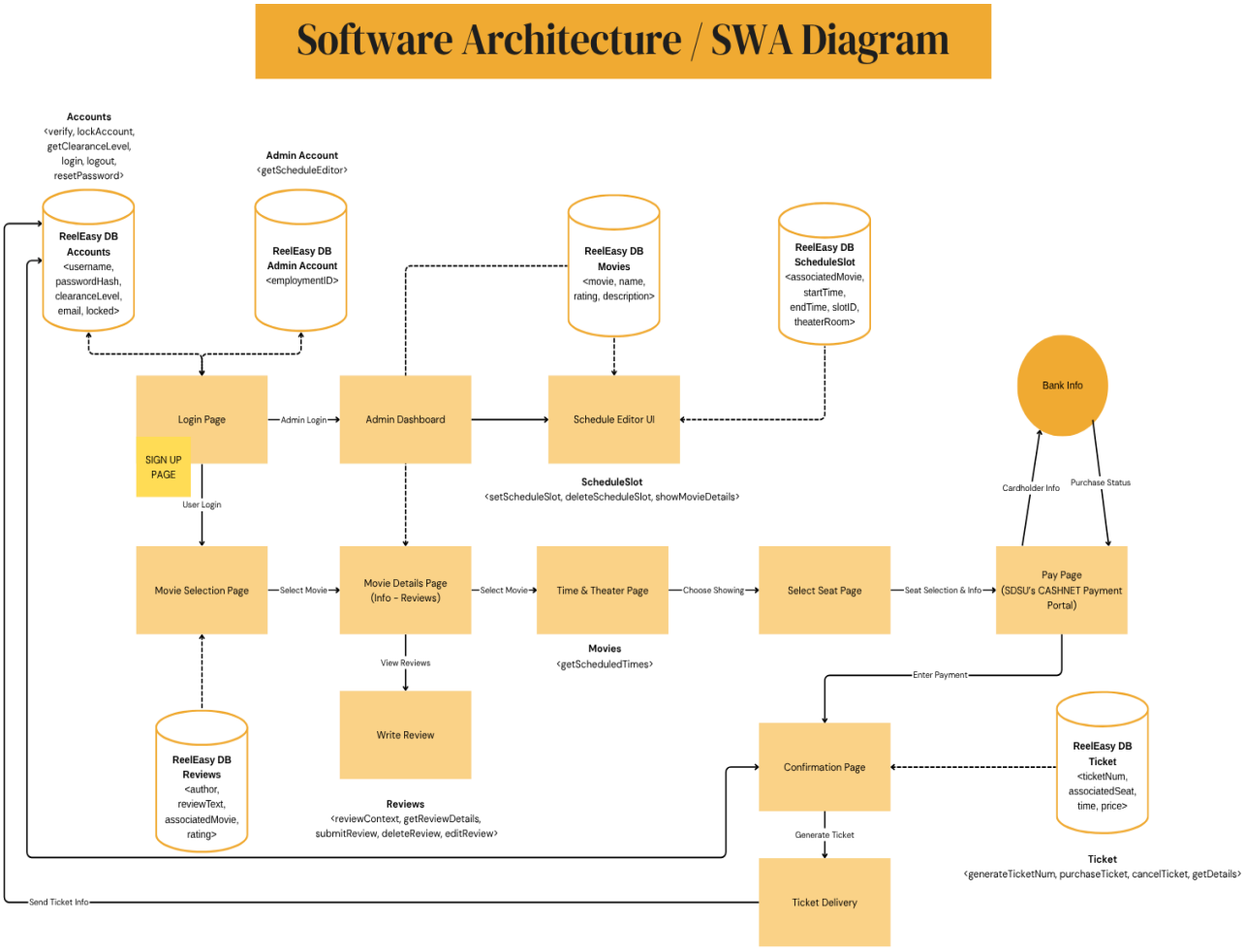
3.2.4.5 Error Handling

Alert admin of missing or invalid data.

Prevent scheduling conflicts that overlap showtimes in the same facilities

Log changes for audit tracking

3.2.5 ReelEasy Software Architecture



ReelEasy - Software Architecture Diagram
(Separate printed handout may be given for the full picture)

The ReelEasy Software Architecture Diagram (SWA) illustrates how the entire ticketing system's front-end user interfaces, admin tools, and back-end storage work together to deliver all required functionality for SDSU's School of Theatre, Television, and Film.

The diagram encompasses main components, attributes, and its functions in how they interact in tandem with one another through a single centralized database, which contains all the information needed to manage users, movies, show schedules, tickets, and reviews.

There is one centralized database called the “ReelEasy” database which has different subcategories for storing different information. This includes:

- Accounts: Contain the information and functions for a user to have a functional account in the app.
- Admin Account: Contains the information and functions for staffers to operate and edit the app.
- Movies: Contains the information and functions of movie listings and descriptions.
- ScheduleSlot: Contains the information and function for the location and times for the movie.
- Reviews: Contains the information and functions for users to access and write reviews on said movies.
- Ticket: Contains the information and functions for users to get their ticket and theater information.

3.2.5.1 Login / Sign Up

- Users start at the Login Page or Sign-Up Page.
- Credentials are verified with the ReelEasy Accounts DB, which stores usernames, hashed passwords, clearance levels (admin/user) and email addresses.
- Accounts can be locked automatically if multiple failed logins attempts occur

3.2.5.2 Movie Selection / Browsing

- Logged-in users can access the Movie Selection Page where they can browse all available movies.
- When a movie is selected, the system will then load the Movie Details Page, which shows the movie information retrieved from the Movie DB and also pull reviews from the Reviews DB.

3.2.5.3 Reviews

- On the Movie Details Page, users can view all reviews for their selected movie

(getReviewDetails()). If logged in, users can submit (submitReview()), edit (editReview()), or delete (deleteReview()) their own reviews.

- The reviewContent() allows the system to validate content to prevent inappropriate submissions.

3.2.5.4 Showtimes / Seat Selection

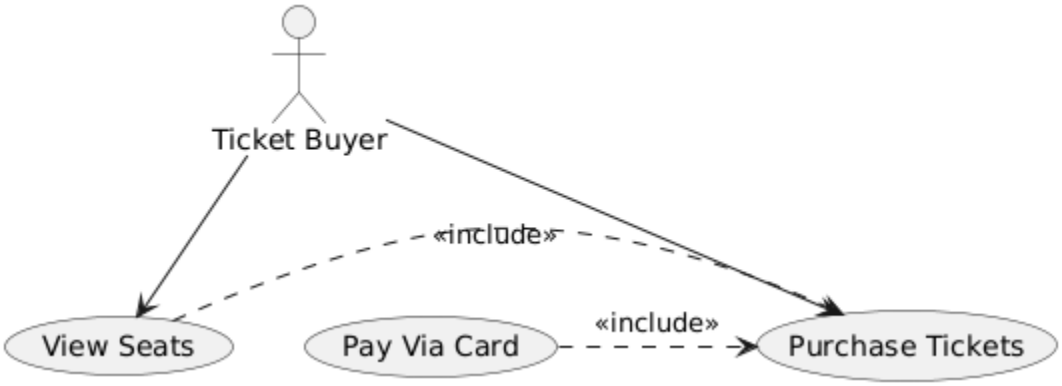
- After selecting a movie, users may proceed to the Time & Theater page where they will display available showtimes retrieved from the scheduleSlot DB.
- The select page allows for the user to choose an available seat for their chosen showtime.

3.2.5.5 Reviews

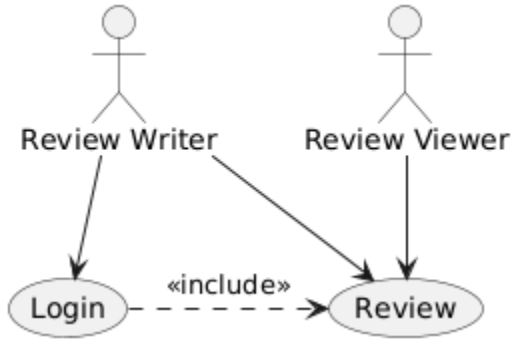
- Pay page verifies payment details and processes the ticket purchase (purchaseTicket())
- The system would generate a ticket number for the user (generateTicketNum()), store it in the tickets table, and create an electronic ticket with the seat, the time, and price information.
- At the confirmation page, it shows the completed transaction, and the Ticket Delivery module sends the ticket details to the user's email or phone.

3.3 Use Cases

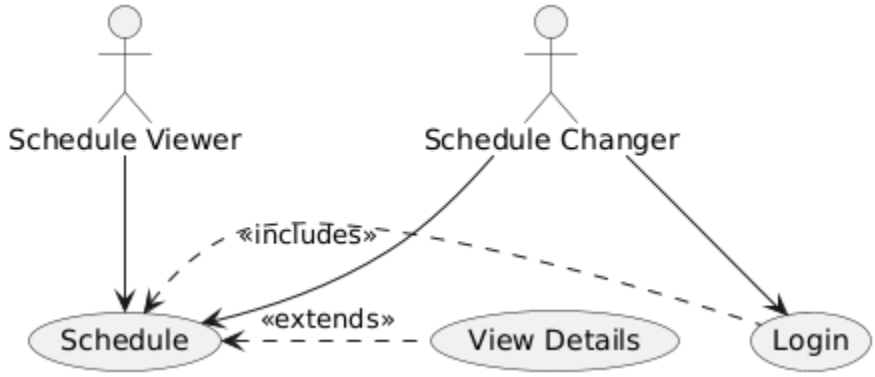
3.3.1 Use Case #1



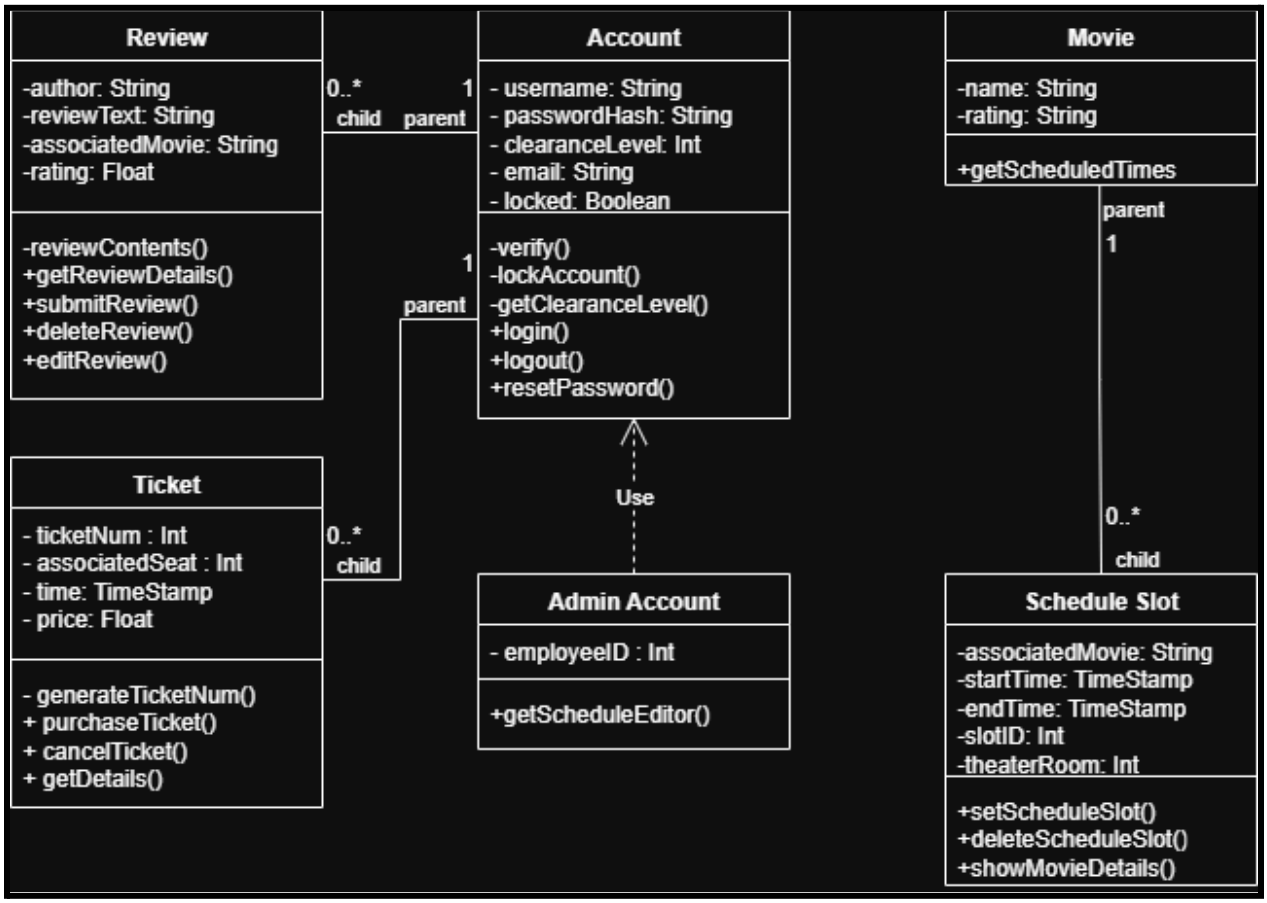
3.3.2 Use Case #2



3.3.2 Use Case #3



3.4 Classes / Objects



3.4.1 Ticket

3.4.1.1 Attributes

- ticketNum : Int
- associatedSeat : Int
- time: TimeStamp
- price: Float

3.4.1.2 Functions

- generateTicketNum()
  A helper function to create a unique ticket ID once the ticket has been purchased.
- purchaseTicket()
  Verifies the payment details and calls the helper function generateTicketNum().
- cancelTicket()
  Refunds the ticket and then deletes it.
- getDetails()
  Shows the ticket's seat, time, and ticketNum in the form of a barcode.

3.4.2 Account

3.4.2.1 Attributes

- username: String
- passwordHash: String
- clearanceLevel: Int
- email: String
- locked: boolean

3.4.2.2 Functions

- verify()
  A helper function that verifies the username and password.
- lockAccount()
  A helper function that locks the account if too many login attempts were made, and sends an email to the account holder's email address to inform them of the account being locked.
- getClearanceLevel()
  A helper function that updates the user interface if the account's clearance level is that of an admin.
- login()
  Gets the user's input username and password attempts, and calls the verify function.
- logout()
  Removes the current connection from the account, returns to the login page.
- resetPassword()
  Sends an email to the account's email address with a link to a page that lets them reset their password.

3.4.3 Admin Account

3.4.3.1 Attributes

- employeeID: int

3.4.3.2 Functions

- getScheduleEditor()
  Opens a user interface for an admin to update the movie schedule, and makes a note on which employee made the change.

3.4.4 Review

3.4.4.1 Attributes

- author: String
- reviewText: String
- associatedMovie: String
- rating: Float

3.4.4.2 Functions

- reviewContent()
  A helper function to ensure no inappropriate content was being submitted.
- getReviewDetails()
  Returns the text, author, movie, and rating of the review for an account to see.
- submitReview()
  Takes the user's review and submits it to the website after calling for the reviewContent() function to ensure it is safe to be posted.
- deleteReview()
  Allows an user that matches the author to delete the review.
- editReview()
  Allows a user that matches the author to edit the review. It reopens the user interface used to submit a review with the details previously submitted.

3.4.5 Movie

3.4.5.1 Attributes
- name: String
- rating: String

3.4.5.2 Functions
- getScheduledTimes()
  Returns a list of times the movie has been scheduled for

3.4.6 Schedule Slot

3.4.6.1 Attributes
- associatedMovie: String
- startTime: TimeStamp
- endTime: TimeStamp
- slotID: Int
- theaterRoom: Int

3.4.6.2 Functions
- setScheduleSlot()
  Allows an admin account to edit and submit the times, associated movie, and room for a movie. Generates a slotID so it can be located in a list of schedule slots.
- deleteScheduleSlot()
  Deletes the schedule slot.
- showMovieDetails()
  Allows for the movie class to get information about this particular schedule slot.

3.5 Non-Functional Requirements

3.5.1 Performance

- The system should complete 95% of ticket purchases in under 2 seconds.

- Pages like movie listings, seat selection, and payment should load in less than 2 seconds on a fast internet connection.
- The system should support up to 1,000 users at the same time without slowing down.

3.5.2 Reliability

- The system should work without major problems for at least 30 days at a time.
- If something goes wrong, users should see a helpful message. The system should try to fix the issue automatically or allow the user to send a support ticket with details of the error.
- An error monitoring tool like Sentry may be used to automatically report and track errors for the development team.
- Important actions like payments should have a retry option if they fail the first time.
- The system should prevent duplicate charges by using a temporary hold period on failed or incomplete payment attempts.

3.5.3 Availability

- The system should be up and running at least 99.5% of the time, except for planned maintenance.
- Maintenance times should be announced at least 48 hours in advance.
- Maintenance should be performed during the night time.
- The system should have alerts to let staff know if it goes down unexpectedly.

3.5.4 Security

- All data should be protected using HTTPS encryption.
- Passwords and sensitive information should be securely stored in a data database using encrypting procedures.
- Different users should have different access levels (admins, customers).
- The system should follow SDSU's security rules and privacy laws.

3.5.5 Maintainability

- The system should be built in a clean, organized and modular way to make future updates easier.
- Settings should be kept in the separate database or environmental files so they can be changed without touching the main code.
- There should be tools to track bugs (aka Sentry) and support/feedback systems to request new features.

3.5.6 Portability

- The website should work on major browsers like Chrome and Safari, others are not so important due to very low popularity among users..

- It should look good and work well on phones, tablets, and desktops. The main focus should be on the mobile version, because expected traffic from mobile should be over 60% versus 35% on the desktop.
- The system should be easy to move between different environments using tools like Docker and containerization technologies.

3.6 Inverse Requirements

*State any *useful* inverse requirements.*

3.7 Design Constraints

Specify design constraints imposed by other standards, company policies, hardware limitations, 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.

3.10 Development & Timeline

3.10.1 Team Responsibilities

The ReelEasy system is being built by four project teams, each responsible for a major area of the system:

- *Front-End Team: Designs and develops all user pages, including login, movie selection, seat selection, payment, and ticket display.*
- *Back-End Team: Builds server logic and APIs for login, reviews, schedule slots, ticket generation, and database interaction.*
- *QA Team: Tests each feature (login, movie browsing, payments, etc.), reports bugs, and verifies fixes.*
- *Project Management: Plans the workflow, monitors deadlines, manages the staging environment, and oversees deployment.*

All teams commit to GitHub and coordinate through weekly development meetings every Monday to review progress, plan the week's tasks, and address any issues.

Group 2 - Project Assignments			
Name	Role	Projects	Contributions
Bobby Bavongkhoun	Developer / Design	Planning	Project planning
Caleb Wolf	Developer / Design	Implementation	Specifications
Gleb Rodin	Developer / Design	Research	Specifications

3.10.2 Development Timeline

Phase: Planning
Dates: July 28 – August 2
Tasks: Define system goals, assign team roles, create GitHub repo, outline user and admin flows
Teams Involved: Project Management
Note: Weekly development meetings begin Monday, July 29
Phase: Design
Dates: August 3 – August 9
Tasks: Finalize database structure, UML/class diagrams, and wireframes for each page
Teams Involved: Front-End, Back-End

Phase: Development Part 1
Dates: August 10 – August 16
Tasks: Build login/signup pages, account verification, admin dashboard access, movie selection and review pages
Teams Involved: Back-End, Front-End

Phase: Development Part 2
Dates: August 17 – August 23
Tasks: Implement ScheduleSlot logic, Time & Theater page, and seat selection flow
Teams Involved: Back-End, Front-End

Phase: Development Part 3
Dates: August 24 – August 30
Tasks: Integrate payment flow (mock SDSU CASHNET), generate ticket, and implement confirmation and ticket delivery
Teams Involved: Back-End, Front-End

Phase: Staging
Dates: August 31 – September 2
Tasks: Push codebase to a staging server for full-system testing
Teams Involved: QA Team, All Teams

Phase: Testing
Dates: September 3 – September 4
Tasks: Conduct end-to-end testing, fix bugs, review performance and security
Teams Involved: QA Team, Back-End

Phase: Deployment
Dates: September 5 – September 6
Tasks: Final deployment to production environment, documentation finalized, and submission prepared
Teams Involved: Project Management, All

3.10.3 Tools and Workflow

- *GitHub: Used for version control and collaboration*
- *Figma or Canva: Used for design mockups and architecture diagrams*
- *VS Code: Used as the code editor*
- *Postman: Used for testing API endpoints*
- *Discord, Zoom and Slack: Used for communication and weekly check-ins*
- *Weekly Development Meeting: Held every Monday to review progress and assign tasks*
- *Staging Server: Used to test the full system before final deployment*
- *SDSU Mock CASHNET Portal: Simulated payment system used for the Pay Page*

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 to 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.

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