Software Requirements Specification Template

Software Engineering

The following annotated template shall be used to complete the Software Requirements Specification (SRS) assignment.

Template Usage:
Text contained within angle brackets ('<', '>') shall be replaced by your project-specific information and/or details. For example, <Project Name> will be replaced with either 'Smart Home' or 'Sensor Network'.

Italicized text is included to briefly annotate the purpose of each section within this template. This text should not appear in the final version of your submitted SRS.

This cover page is not a part of the final template and should be removed before your SRS is submitted.

Movie Ticketing System

Software Requirements Specification

V-0.3

10/24/24

Group 08

Teague Sangster,
Victor Allen,
Chad Childers,
Trevor Plott.

Prepared for
CS 250 - Introduction to Software Systems
Instructor: Gus Hanna, Ph.D.
Fall 2024

Revision History

Date	Description	Author	Comments
<date>	<Version 1>	<Your Name>	<First Revision>
09/26/24	Version 0.1	Group 8	First Draft
10/10/24	Version 0.2	Group 8	Second Draft w/ Design
10/24/24	Version 0.3	Group 8	Third Draft w/ Test Plan

Document Approval

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

Signature	Printed Name	Title	Date
	<Your Name>	Software Eng.	
	Dr. Gus Hanna	Instructor, CS 250	

Table of Contents

PAGE INTENTIONALLY LEFT BLANK

1. Introduction

- The introduction to the Software Requirement Specification (SRS) document should provide an overview of the entire SRS document, including its scope, purpose, definitions, acronyms, abbreviations, and references. The aim of this document is to provide a complete and clear overview of the Movie Ticketing System by clearly defining the problem statement. It also focuses on the requirements of stakeholders and their needs while outlining high-level product features. The requirements of the Movie Ticketing System are provided in this document.

1.1 Purpose

- The purpose of this document is to provide a detailed description of the Movie Ticketing System. This system will allow consumers to browse movies and shows, select seats, purchase tickets, and manage their reservations. It will also provide administrative tools for managing shows, pricing and seating.
- Further, this document will describe the project's target audience, user interface, and hardware and software requirements. It will define how the client team and audience view the product and its features. It will also help any developer in maintenance or the software delivery lifecycle (SDLC) process.

1.2 Scope

- The Movie Ticketing System will be used by both users (consumers) and theater management or administration. It will provide a web-based user interface that users can use to book tickets to movies, and where administrators can manage bookings, pricing,movie showings, and seat availability. The primary functionalities of the System will include user registration, show browsing, ticket booking, and payment processing. The ticketing system will be web based and be compatible with common web browsers, like chrome, edge, safari, and firefox.

1.3 Definitions, Acronyms, and Abbreviations

API	App Protocol Interphase
REDIS	Remote Dictionary Server
MTS	Movie Ticketing System
Mac	Macintosh
MTBF	Mean Time Between Failures

SRS	Software Requirements Specification
PII	Personally Identifiable Information
SMS	Short Message Service
GCS	Google Cloud Services
SDLC	Software Delivery Lifecycle
2FA	Multi-Factor Authentication
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
// Add as they become important (right-click + add row above/below)	

1.4 References

- IEEE Recommended Practice for Software Requirements Specifications, 1998 Software Engineering Standards Committee
- How to write a good SRS for your Report, 3/17/23, geeksforgeeks.org, A.1

1.5 Overview

- The further sections of this document provide a general description, including general project user characteristics, the product's hardware, as well as the product's functional and data requirements. A general description of the project can be found in section 2 of the SRS. Section 3 gives functional and data requirements, and also gives the specific requirements of the product, describes the external interface requirements, and gives a description of functional requirements. Section 4 provides analysis models in the form of charts and graphs to provide additional detail to the information in this document. Section 5 describes the process to be used when this document needs to be updated. The Appendices provide useful context or additional information on the details of the project.

2. General Description

2.1 Product Perspective

- The MTS will improve on other online ticketing systems and is intended to replace physical and manual tickets. It (the MTS) will give users the ability to view show schedules, browse movie offerings, and book tickets.

2.2 Product Functions

- User Registration and Authentication: Users can register, log in, and manage their accounts.
- Show Listings: Users can view upcoming shows with details like timing, available seats, and pricing.
- Seat Selection: Users can view and choose available seats.
- Ticket Booking: Users can reserve or purchase tickets.
- Payment Gateway Integration: Users can pay for tickets through various payment options (credit/debit cards, online wallets, etc.).
- Booking Confirmation: Users receive booking confirmations via email or SMS.
- Ticket Cancellation: Users can cancel their tickets under specified conditions.
- Admin Panel: Admins can add/edit show details, manage bookings, and view reports.

2.3 User Characteristics

The users of the *Movie Ticketing System* (MTS) can be broadly categorized into three types: **Consumers**, **Administrators**, and **Support Staff**. Each user type will have specific roles, needs, and access privileges within the system.

1. Consumers (End Users):

- **Description**: Consumers are the primary users of the system. They will interact with the system through a web-based user interface to browse movie listings, book tickets, and manage their reservations.
- **Technical Expertise**: Most consumers will have basic to intermediate computer skills, with the expectation that the interface is intuitive and user-friendly.
- **Devices Used**: Consumers will typically access the MTS via desktop or mobile browsers (Chrome, Safari, Firefox, etc.), on personal devices such as laptops, tablets, or smartphones.
- **Primary Tasks**:
    - Register and log into the system.
    - Browse available movies, showtimes, and seat availability.
    - Purchase tickets using various payment methods.
    - Manage account details (update profile, payment methods, etc.).
    - Cancel bookings (subject to system constraints).
    - Receive notifications about bookings, cancellations, or promotional offers.
- **Additional Considerations**: Consumers may prefer personalized recommendations based on past purchases and may require support for accessibility features (e.g., text size adjustment, screen reader compatibility).

2. Administrators:

- **Description**: Administrators are responsible for managing the backend of the MTS, ensuring the system runs smoothly. They oversee the management of showtimes, pricing, seat availability, and general system maintenance.

- **Technical Expertise**: Administrators will typically have intermediate to advanced computer skills, capable of using administrative panels, managing databases, and responding to user requests.
- **Devices Used**: Administrators will generally use desktop computers, accessing the admin panel through standard web browsers.
- **Primary Tasks**:
    - Log into the admin dashboard with elevated privileges.
    - Add, update, or remove movie listings and showtimes.
    - Manage seating availability and pricing.
    - Handle customer complaints or special requests (e.g., refunds).
    - Monitor ticket sales, performance metrics, and system security.
    - Generate reports for business analysis (sales data, popular movies, etc.).
- **Additional Considerations**: Administrators may need to respond quickly to high-traffic events, such as major movie releases. They may also have access to user data for administrative purposes and will need to follow data privacy regulations.

3. Support Staff:

- **Description**: Support staff will interact with both consumers and administrators to provide assistance in case of technical or transactional issues.
- **Technical Expertise**: Support staff will have intermediate technical knowledge, especially in troubleshooting user issues, handling payments, and solving booking-related problems.
- **Devices Used**: Typically, support staff will use desktop systems to access customer information and troubleshooting tools.
- **Primary Tasks**:
    - Assist users with registration, login issues, or booking errors.
    - Resolve payment-related queries and process refunds.
    - Guide users on how to use the system’s features.
    - Report system bugs to the development team.
    - Handle urgent system issues such as duplicate bookings, misallocated seats, or payment failures.
- **Additional Considerations**: Support staff must be proficient in using the ticketing and payment system, ensuring users' issues are resolved quickly to prevent revenue loss or user dissatisfaction.

User Privileges

Each user type will have different access levels to the system:

- **Consumers**: Can access movie listings, book tickets, and manage their accounts.
- **Administrators**: Have access to all consumer-facing features as well as administrative functions like adding/editing movie showings, seat management, and generating reports.
- **Support Staff**: Have access to both consumer and administrative sections but are focused on troubleshooting and resolving issues.

Usability Considerations:

- **Consumers** need a straightforward, accessible interface with minimal technical barriers to ensure ease of use.
- **Administrators** require a more complex interface with tools for management but should still have an intuitive layout to reduce training time and maximize efficiency.
- **Support Staff** need efficient access to both user-facing tools and administrative controls, with features enabling quick response to consumer issues.

User Experience (UX) Expectations

- **Consumers**:
  - Fast and easy booking experience.
  - Responsive design to support both desktop and mobile platforms.
  - Notifications and reminders for upcoming bookings, seat confirmation, or promotional offers.
- **Administrators**:
  - Clear, easily navigable dashboards.
  - Robust security features (multi-factor authentication, role-based access control).
  - System alerts for critical actions (e.g., double bookings, system outages).
- **Support Staff**:
  - Access to detailed user logs for troubleshooting.
  - A simple and fast system to refund payments or modify bookings.

2.4 General Constraints

- The constraints of the MTS include a user-friendly interface compatible with common web browsers such as Chrome, Edge, and Firefox. While the system does not support mobile platforms, it shall remain compatible with smaller devices and touch screens.
- The system will require storage, security, and role-based management for consumer, admin, and support credentials. Monitoring and alerts for system performance and security issues will also be required.
- The MTS will comply with local, state, and federal online transactions and data privacy regulations. This will include maintaining and integrating third-party payment and API systems such as, but not limited to, Visa, MasterCard, and PayPal. Logging of all transactions for accounting and auditing purposes will also be required.
- The MTS will be capable of handling peak loads during high-demand periods(big movie releases) and support concurrent movie ticket purchases with a limit of 25 tickets per movie per transaction.
- Programming languages used will include, but are not limited to, Java, Python, Node.js, MySQL, JavaScript, and HTML/CSS.
- Prioritization of system performance and reliability is crucial for sales revenue and customer satisfaction.

- The MTS will implement robust security measures with a heavy emphasis on protecting user data and preventing unauthorized access using encryption and multi-factor authentication.

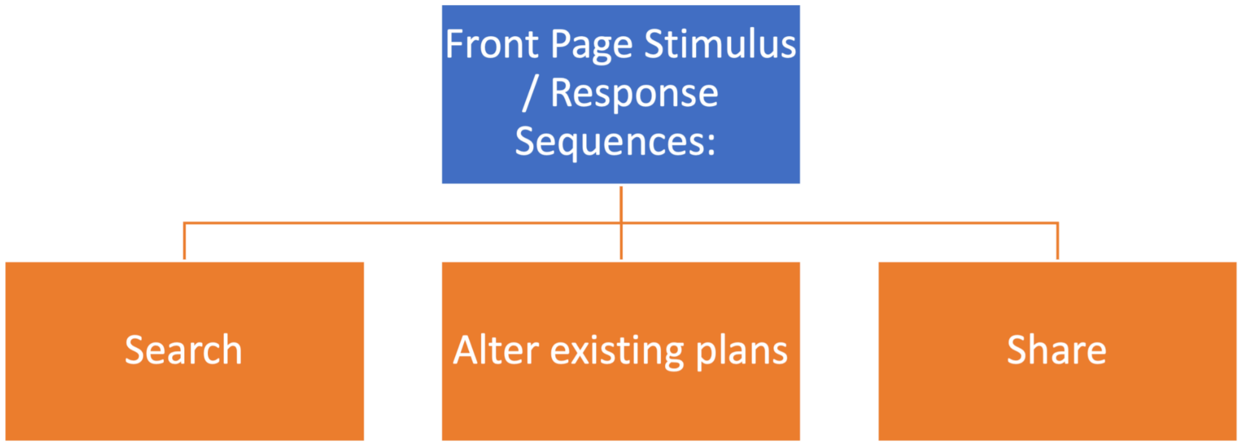
2.5 Assumptions and Dependencies

- This system assumes that the user has a stable internet connection and is using a supported browser on a supported operating system.
- The MTS system will depend on its ability to access third-party payment systems. Therefore, it is assumed that access to this will be available and reliable. Any changes to payment systems could require changes to the system and subsequent documentation.
- The system is dependent on its compliance with local, state, and federal regulations concerning online transactions and data privacy. Any change in regulation may require system updates.
- The system will be dependent on the specific hosting environment. Any change in this environment can affect system integrity and availability.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces



Front Page Stimulus / Response Sequences:

Search:
- Search for a theater in their location, provided with a way to compare their current location with possible locations around them. As well as organize locations by the times they offer for a certain movie.
- Search for Movies, users must be able to search for the movie they wish to see, as well as suggestive completion should the user not fully understand or remember the title of the film. (We don't want to lose a potential sale because they don't know the exact name.)

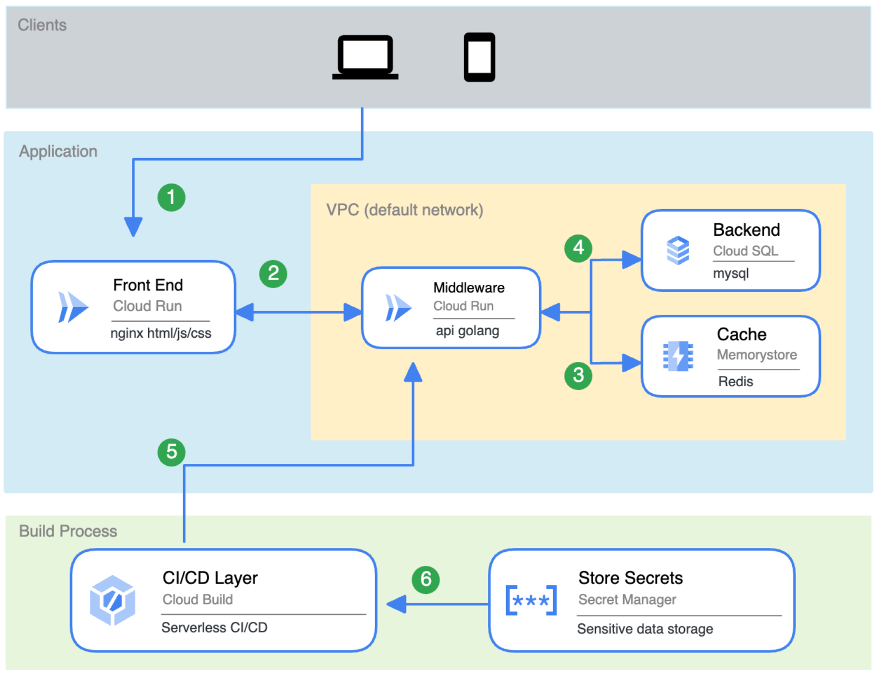
Alter existing plans:
- Users must be able to navigate from the front page to items they have purchased / upcoming. This should be really available next to potential purchases so users can understand and edit any previous purchased items.

Share:
- Provide a way to not only send a move they are interested in, but also once a ticket is purchased send additional details such as which location they're attending, when they're attending, and what seat they purchased.

3.1.2 Hardware Interfaces

Backend:

Hosted on Google Cloud Services and built on FireBase for easy scalability and implementation.



Front End:

- Held within Cloud Run as a Docker image scaled to interact with Pub/ Sub docker images to handle API requests.
    - Node JS with Google Firebase Authentication for handling user sign in authorization. Node JS acts as the framework for our front (user-facing website) with any interfaces being separated into the following buttons.
        - Client Rendered
            - Any non-critical or non-sensitive information, such as movie posters, navigation UI, and Movie times.
        - Server Rendered
            - Any location data, checkout sequences/credit card transitions, personal account details, and authorization.

Middleware APIs (App Protocol Interface):

- Docker Image, built using Spring Boot, Maven, Java, and Feign to handle API requests.
- Built to be quickly deployable and scalable as needed while interacting with the Front End / Back End Requests

Back End:

- Built on SQL and uses GCS (Google Cloud Storage) Firebase buckets to handle non-critical data for long-term storage.

- Deployable REDIS (Remote Dictionary Server) in times of High Demand (Such as when tickets first go on sale). It acts as a high-availability (in RAM) database for handling ticket checkout and seat assignments. Setup in raid 1 so that if a server goes down, the entire “Lightning” checkout window’s history isn’t lost.

3.1.3 Software Interfaces

Front End:

- The front-end of the movie ticketing website is the user-facing interface designed to provide a seamless and intuitive experience for customers purchasing movie tickets. Built with modern web technologies such as HTML, CSS, and JavaScript, it is optimized for accessibility and performance across various devices, including desktops, tablets, and smartphones. The user interface is responsive and adjusts dynamically based on the screen size to ensure a consistent experience.
- **Home Page**:The website will consist of a home page, where we will feature new and popular movie releases.
  Movie Listing Section:A movie listing section where the user can find all movies that are being shown in theaters.
  Ticket Booking interface:A ticket booking interface where the user will choose which theater they would like to go to as well as what seat they would like to sit in.
  We will also include a section where the user can make an account, or access their account if they already have one. In this section they can see things like payment information, past purchases, personal details, as well as a section where they would cancel tickets if needed. Their account and information will be used to recommend movies that they would enjoy seeing based on the movies they have already seen.
  Loyalty Rewards Program: There will also be a loyalty rewards program which would incite the user into spending money for a reward.
  Accessibility section:The website will also feature an accessibility section which would allow the user to adjust certain options to fit their needs, like adjustable text sizes, and keyboard navigation.

Software Services:

- The software services of the movie ticketing website are responsible for managing the core business logic, user interactions, and data flow between the front end and the back end. These services ensure seamless functionality for users while interacting with the platform and ensure secure and efficient operation in real time. These services would include the following:
- **Movie information service**: This service would be used to manage and find movie data. This data includes movie titles, descriptions, cast, showtimes, trailers, and ratings. It makes sure that the latest and most accurate data is represented.

Showtime and seat availability service: This service would track data from the theaters and provide the user with accurate information about showtimes for certain movies, as well as the seats available for those movies.
The booking and registration service: This service will ensure that when the user purchases a ticket, that their seat is reserved, and it will not allow any other customers to purchase a seat that has already been taken. The booking will be confirmed when the user's payment information has been confirmed and the website will generate a digital ticket with a QR code, as well as send one to the user's email, and will also send a text message confirming the purchase with all the details.
Payment integration service: There will need to be a way to successfully integrate external payment types for the user to use. These can include paypal, stripe, or a credit/debit card processor. Security will be the number one priority in this section to ensure that the users financial data is secure and will not be leaked. The user will also have the option to store their payment information for future use.
User Authentication Service: The user will have to be authenticated, this can be done through things such as a 2 factor authentication that the user sets up when making an account. This can be done using the user's email or phone number. All of the users data will have to be stored securely and encrypted to ensure that the data is never leaked.
Security and Fraud Detection Service: This will be a service to monitor any sort of fraudulent activity. This type of activity can include multiple attempts to log in, and unusual purchases. This would have to be integrated so that it could alert the user of the possible fraud either via email or text message.

Back End:

- The back end of the movie ticketing website forms the backbone of the system, responsible for handling all server-side operations, database management, business logic, and integrations with external services. It ensures the smooth functioning of all processes related to user interaction, data storage, and communication between the front end and other third-party services. Built using modern frameworks and technologies, the back end provides scalability, security, and performance optimization for the platform.
- Key Components:
  Server Side Logic: The core business logic of the system, including user authentication, ticket booking, seat allocation, and payment processing, is handled by the back end. This logic ensures that user requests are processed in real time, including ticket reservations, seat locking, and other transactional operations.
  DataBase Management: The back end manages all persistent data related to movies, showtimes, theaters, users, bookings, payments, and rewards. It interacts with a relational database (such as MySQL, PostgreSQL) or a NoSQL database (such as MongoDB) to store and retrieve structured and unstructured data efficiently. There are many key entities in the database. These include can be broken down into a couple different categories.
    - **Users**: All user data, login credentials, history, payment methods.
    - **Movies**: Information on movies, genres, cast, trailers.
    - **Showtimes**: Different showtimes and different theaters.
    - **Seats**: Seats available at different theaters and which ones are already booked.

- **Bookings**: A database of all tickets bought, correlating with which theaters, time, and which seat was bought.

Payment Integration: There needs to be a way to securely and efficiently integrate payment methods that the user trusts to store all of their sensitive data. The payment integration would ensure encryption of all users data and this can be done using SSL/TLS protocols and adhering to PCI-DSS compliance for handling sensitive user data.
Seat Locking Mechanic: To prevent double booking the back end would ensure that no two customers book the same seat, with a seat locking mechanic that is initiated once the user starts the booking process.
Third Party Integration: There would need to be integration of many third party resources, these include, movie databases, of course payment gateway, messaging services, and analytics to gather user data in order to form a more user based product.

3.1.4 Communications Interfaces

- **Protocols:** HTTPS for secure communication between users and the web server and possibly with other protocols for communication with external services.
- **External Systems:** Payment gateways for processing payments, email servers for sending relevant information, and SMS gateways for sending text notifications.
- **Data Formats**: JSON for data exchange between front and back ends.
- **Error Handling:** Mechanisms will be in place to properly handle communication failures, timeouts, and other issues.

3.2 Functional Requirements

3.2.1 User Interface

3.2.1.1 Introduction:

- This system will be accessible via compatible web browsers on supported devices.

3.2.1.2 Inputs:

- Users on the website will be able to send requests for new info from the server.

3.2.1.3 Processing:

- Each request will be processed via the server over an internet connection and will be sent back to the end user's computer.

3.2.1.4 Outputs:

- The system will output updated info from the server that the user has requested.

3.2.1.5 Error Handling:

- Errors that occur on the server side will be handled with a 404 Page Not Found error. Any errors from the user, such as incompatible browsers, will be handled with an HTTPS compatibility error, and a message will be displayed advising the user that there is a compatibility issue.

3.2.2 Sign up, Log in, and Authentication

3.2.2.1 Introduction:

- This system will allow users and admins to log in or sign up for the website, as well as manage stored PII.

3.2.2.2 Inputs:

- Users and admins will enter their username and password, then be authenticated by 2FA and a CAPTCHA to prevent bot registration or brute force hacking.

3.2.2.3 Processing:

- The processing and authentication will all happen on the server side.

3.2.2.4 Outputs:

- After login, the user will be redirected to the page they were on previously.

3.2.2.5 Error Handling:

- Errors with logins will be displayed to the user, but the number of attempted logins will be limited to 5 per hour. Usernames and passwords that are already taken or not allowed will display a message with the requirements for that text field.

3.2.3 Movie Listings, Showtimes, and Availability

3.2.3.1 Introduction:

- The system will allow users to browse movies and the associated showtimes

3.2.3.2 Inputs:

- Users will be able to browse and search for movie listings by title, genre, actors, and keywords.

3.2.3.3 Processing:

- This system will process requests through the webpage and send/receive all requests from the server

.

3.2.3.4 Outputs:

- The output from a search will produce a webpage that displays all titles associated with the keyword used to search. If no keywords are defined, a list of the most recent movies that are now playing will be displayed.

3.2.2.5 Error Handling:

- Errors with the input will be handled by displaying a "No Results Found" page. Errors with the webpage will produce a 404 Page Not Found error.

3.2.4 Seat Selection and Reservations

3.2.4.1 Introduction:

- The system will display a webpage with available seats and allow the user to make reservations.

3.2.4.2 Inputs:

- The system will display a "mock" version of the theater with clickable seats for selecting the desired location.

3.2.4.3 Processing:

- The server will process requests for a reservation by blocking out the seats for 5 minutes while the user processes their payment method.

3.2.4.4 Outputs:

- The output of a reservation request will redirect the user to a payment page that displays items being purchased with a description and the seat location, as well as fillable boxes for all necessary credit/debit cards or bank accounts.

3.2.4.5 Error Handling:

- Errors in making a reservation will redirect the user to the page where the error occurred and highlight specific issues, such as seats being taken already. Failure to reach the server will result in a 404 Page Not Found error.

3.2.5 Payment Processing

3.2.5.1 Introduction:

- The user will be able to pay for their movie tickets via PayPal and major credit card companies, such as but not limited to Visa, MasterCard, and American Express. The option to use bank accounts directly will also be implemented.

3.2.5.2 Inputs:

- The user will input credit card information or bank account information into fillable text boxes and then submit with a clickable button. If the user has saved credit cards or bank info, they will be given the option to prefill these boxes.

3.2.5.3 Processing:

- This sensitive PII will be processed with the highest level of security. Requests to the server will be encrypted.

3.2.5.4 Outputs:

- The output of this page will result in a receipt page with purchase information, printable ticket links, and an option to send yourself an email copy of the purchase order.

3.2.5.5 Error Handling:

- Third-party payment systems will handle Errors with payment processing. If a server-side error occurs, no info will be sent to the payment system, and the user will be prompted to re-enter their info.

3.2.6 Cancellations and Refunds

3.2.6.1 Introduction:

- The system will allow for users to receive refunds for their purchase up until the start time of the movie they selected. Tickets will also be refundable at the front desk of each theater.

3.2.6.2 Inputs:

- There will be a link on the confirmation page and the emailed receipt to refund tickets. A login will be required.

3.2.6.3 Processing:

- Processing of the refund can be initiated by either the theater on behalf of the user or the user themselves, but the refund processing will come from the third-party payment system the user used.

3.2.6.4 Outputs:

- After a successful refund or cancellation, a confirmation page will appear, with an option to email a version of that page.

3.2.6.5 Error Handling:

- Front desk staff or online admins will handle errors canceling a ticket.

3.2.7 Notifications

3.2.7.1 Introduction:

- The system will alert users and theater management of any show time changes or cancellations. Users can also opt-in/out of notifications about deals and new releases.

3.2.7.2 Inputs:

- The system will accept new notifications from theater admins with access to push new notifications.

3.2.7.3 Processing:

- Processing of these notifications will happen automatically on the server side.

3.2.7.4 Outputs:

- These notifications can be sent either by receiving emails or by getting browser alerts.

3.2.7.5 Error Handling:

- Errors with notifications will be handled with a follow-up notification notifying recipients of the error.

3.2.8 Security and PII Handling

3.2.8.1 Introduction:

- The system will ensure secure storage and transmission of PII or other sensitive information.

3.2.8.2 Inputs:

- Required inputs for safe PII handling will be user logins, 2FA, and CAPTCHA checkboxes to ensure the appropriate user access the right information.

3.2.8.3 Processing:

- This will all be handled with HTTPS protocols and safe server practices. All processed info will be encrypted and handled automatically.

3.2.8.4 Outputs:

- Successful logins will output the appropriate following page. Unsuccessful logins will output an error message and advise that the username or password was incorrect, forcing a new login attempt.

3.2.8.5 Error Handling:

- Errors with security will be handled by not allowing the attempted login, and requesting the user to log in again. If errors persist, an admin will address the issues with the cloud service provider.

3.2.9 Admin Dashboard

3.2.9.1 Introduction:

- Admins will have special access to a dashboard with extra privileges.

3.2.9.2 Inputs:

- The admin will input their credentials in the regular login screen.

3.2.9.3 Processing:

- The server will process requests for administrative access.

3.2.9.4 Outputs:

- Successful logins by admins will output a dashboard with options for user view, admin tools, and third-party system support.

3.2.9.5 Error Handling:

- The hosting cloud service will handle errors with admin logins. If the problem lies with the website itself, the development team in charge of the administrator dashboard and logins will handle the error.

3.3 Use Cases

3.3.1 Use Case #1: Purchase Ticket

- **Actor:** User
- **Description:** A user wishes to purchase a movie ticket using the MTS.
- **Precondition:** User must be logged in or create an account.
- **Basic Flow:**
    1. User looks through available movies and showtimes
    2. User selects a movie and a showtime.
    3. User selects from open seats at that show
    4. User proceeds to checkout.
    5. User enters payment information
    6. System processes payment
    7. System confirms the purchase and sends a confirmation email or text.
- **Error:** If payment fails the user is prompted to try again or to cancel the transaction.

3.3.2 Use Case #2: Admin Login

- **Actor:** Administrator
- **Description**: A system admin logs into the system to manage movies, showtimes, users, and other tasks.
- **Precondition:** Admin must have valid credentials.
- **Basic Flow:**
    1. Admin enters username and password on the admin login page.
    2. System verifies the admin's credentials.
    3. If credentials are valid, the system grants access to the admin dashboard.
    4. Admin is allowed to perform necessary tasks.
- **Error:** If Credentials are invalid, the system displays an error message and prompts to re-enter password.

3.4 Classes / Objects

3.4.1 User

3.4.1.1 Attributes

- email: the users email
- discount: whether or not the user has a discount.

- userID: a unique identifier for each user
- name: the user’s name
- password: the user’s account password
- phoneNumber (optional): the user’s phone number
- paymentInformation (encrypted): the saved payment information for the user
- bookings: a list of tickets/bookings the user has made.

3.4.1.2 Functions

- Register for an account
- Log into the system
- Purchase Tickets
- Manage Account information
- Manage and view booking information

3.4.2 Ticket

3.4.2.1 Attributes

- price: The price of the ticket.
- ticketID: a unique identifier for the ticket.
- showtimeID: References the specific showtime (date, time screen) for the ticket.
- userID: References the user who purchased the ticket.
- QRCode: A machine readable code used for entry
- used: Indicates if the ticket has been used for admission or not.

3.4.2.2 Functions

- purchase a ticket
- Generate a unique Ticket Id
- Assign a seat number based on availability
- Show ticket price
- Update ticket status
- Send confirmation
- Store ticket information in the database.

3.4.3 Movie

3.4.2.1 Attributes

- movieID: a unique id for the movie
- name : The name of the movie
- date: The date of the movie
- time: The time of the movie
- seats: A boolean array indicating which seats are taken at each movie. Will prevent double booking.

3.4.2.1 Functions

- getters and setters for the attributes.

3.4.4 Theater

3.4.4.1 Attributes

- movies: an array of the movies playing at that theater

3.4.4.2 Functions

- updateMovie : allows an admin to add or remove movies being shown.
- displayMovies: displays all the movies playing in that theater.

3.4.5 Admin

3.4.5.1 Attributes

- adminID: A unique identifier for each admin.
- username: The admin’s username for logging into the system.
- password: A hashed version of the admin’s password.
- permissionsLevel: Specifies the level of access and permissions the admin has (e.g., full access, restricted access to certain operations).
- email: The admin’s contact email.
- theatersManaged: A list of theaters that the admin is responsible for managing (if applicable).

3.4.5.2 Functions

- Getters and setters

3.4.6 Account

3.4.6.1 Attributes

- username: the name of the account holder
- password: hashed password for the account
- userID: references the user associated with the account

3.4.6.2 Functions

- Log in the user
- Reset the user’s password.

3.4.7 Payment

3.4.7.1 Attributes

- transactionID: an identifier for the transaction.
- ticket: The ticket object corresponding to the transaction

3.4.7.2 Attributes

- confirmPurchase: confirms the purchase of a ticket, returns a boolean
- updateSeats: Allows the user to update their reservation and change their seat.

3.5 Non-Functional Requirements

3.5.1 Performance

- The system should be able to handle a large number of concurrent users, especially during peak hours or popular movie releases.
- The response time for ticket booking and other transactions will be fast and efficient, ideally within a few seconds.
- Page load times will be minimal.
- Quick search and filtering of movie showings and seat availability will be optimized.

3.5.2 Reliability

- The reliability of the system will be based on GCS target reliability of 99.99%
- The system's reliability will depend on third-party payment system uptimes, which boast high levels of service reliability.

3.5.3 Availability

- The availability of this system relies on the GCS availability target of 99.99%
- The availability of third-party payment systems will depend on the uptime of each system. These systems promise near 24/7 availability.

3.5.4 Security

- The MTS will employ industry-standard security measures to protect PII, payment details, and browning history.
- Strong password policies and 2FA will be used to prevent unauthorized access.
- Sensitive data will be encrypted, both at rest and in transit using industry standard encryption algorithms to protect against data breaches.
- The MTS will comply with all relevant data protection regulations and industry best practices.

3.5.5 Maintainability

- The system shall be designed in a modular way to make it easy to update and maintain.
- Code shall be well-documented and follow established coding standards to enhance readability and maintainability.
- A comprehensive set of automated tests shall be developed and maintained to facilitate regression testing and ensure the stability of the system during maintenance activities.
- Maintenance will be performed with a clear plan and will be scheduled to ensure minimal downtime for users.

3.5.6 Portability

- The website is intended to be portable and accessible to a wide variety of people and will be available on all common web browsers, including chrome, edge, firefox, ect.

3.6 Inverse Requirements

3.6.1

- The System will allow the purchase of tickets up to 10 minutes after a show has started.

3.6.2

- Sensitive user information will be encrypted in order to protect privacy.

3.7 Design

Chad - SRS/Movie Theatere Platform Group 8.docx (1).pdf at main · Xymos1/SRS (github.com)

Trevorhttps://github.com/Tplott/Movie-Theatere-Platform-Group-8/blob/main/Copy%20of%20Movie%20Theatere%20Platform%20Group%208.docx.pdf
Teague - https://github.com/TeagueSS/CS250.git

Victor - https://github.com/victoralien/CS250/tree/main/Assignment02

3.7.1 Architecture Design

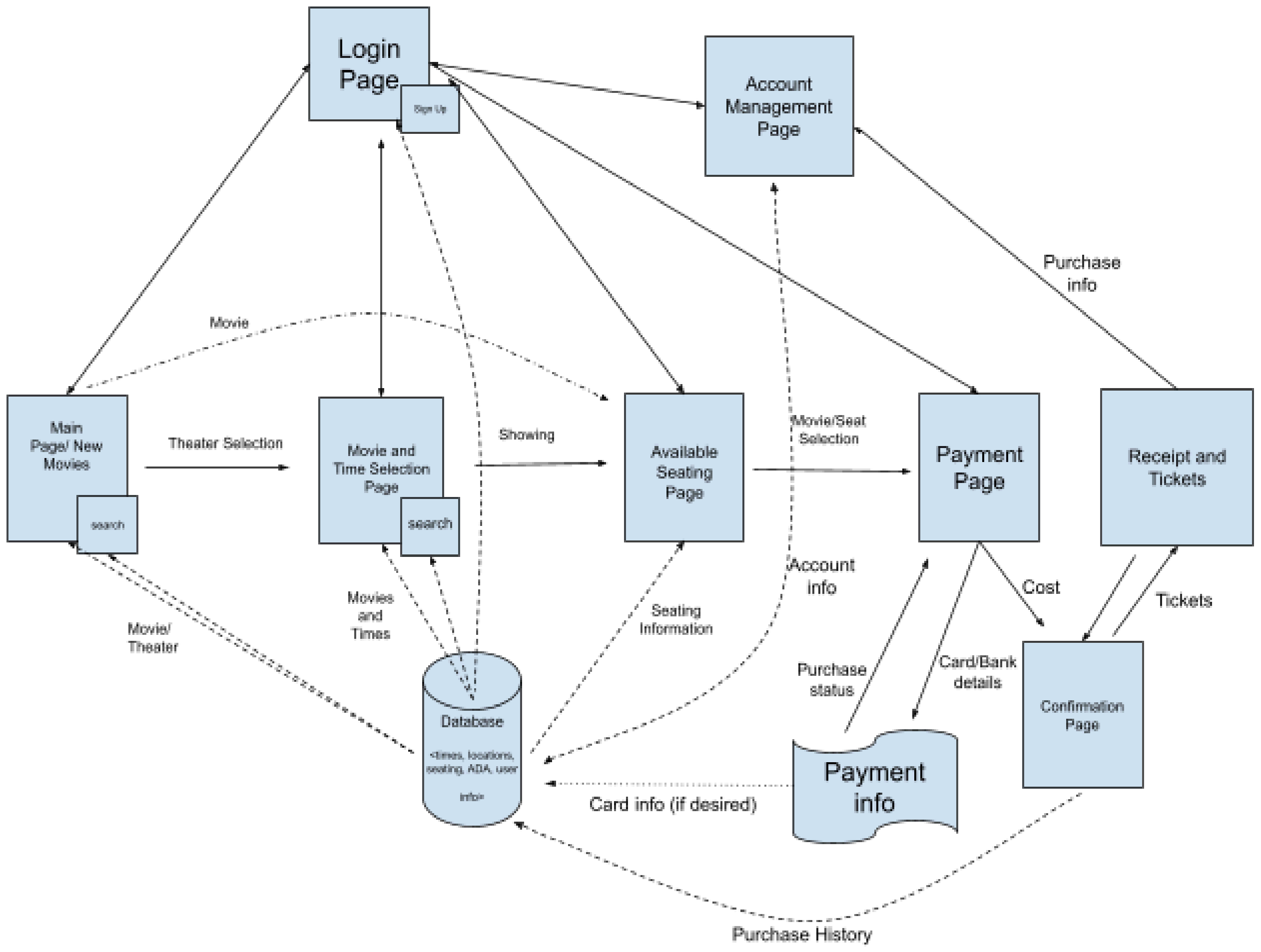


Figure 3: Movie Ticketing System Software Architecture Design Diagram

Description

- Figure 3 shows how the requirements of the ticketing system will be implemented and what components will be necessary. The primary components are the main page, login page, account management page, movie and time selection page, available seating page, payment page, and the confirmation page.
- The figure is used to describe the software architecture of the ticketing system, as well as visually shows how all of the main aspects of the website work together. In the figure we

can see components such as login page, movie and time selection page and a payment page, this diagram shows how they interact, communicate and function.

- This figure shows the flow of information as the customer traverses the website and makes choices. The customer will start on the main page and be given several options, such as searching, selecting a popular movie, or choosing a theater. From here, they will make appropriate decisions based on the page they are on, like movie, time, and seat selection. Finally, they move to the payment page and then the confirmation.
- The figure shows how information moves to and from the centralized database. Dotted lines represent potential data flow, whereas solid lines are selections that need to be made and data that will be transferred.
- Figure 3 also shows how customer information will be transmitted to a third party for payment, then displayed on the confirmation page and stored in the account management page.

3.7.2 UML Classes and Functions

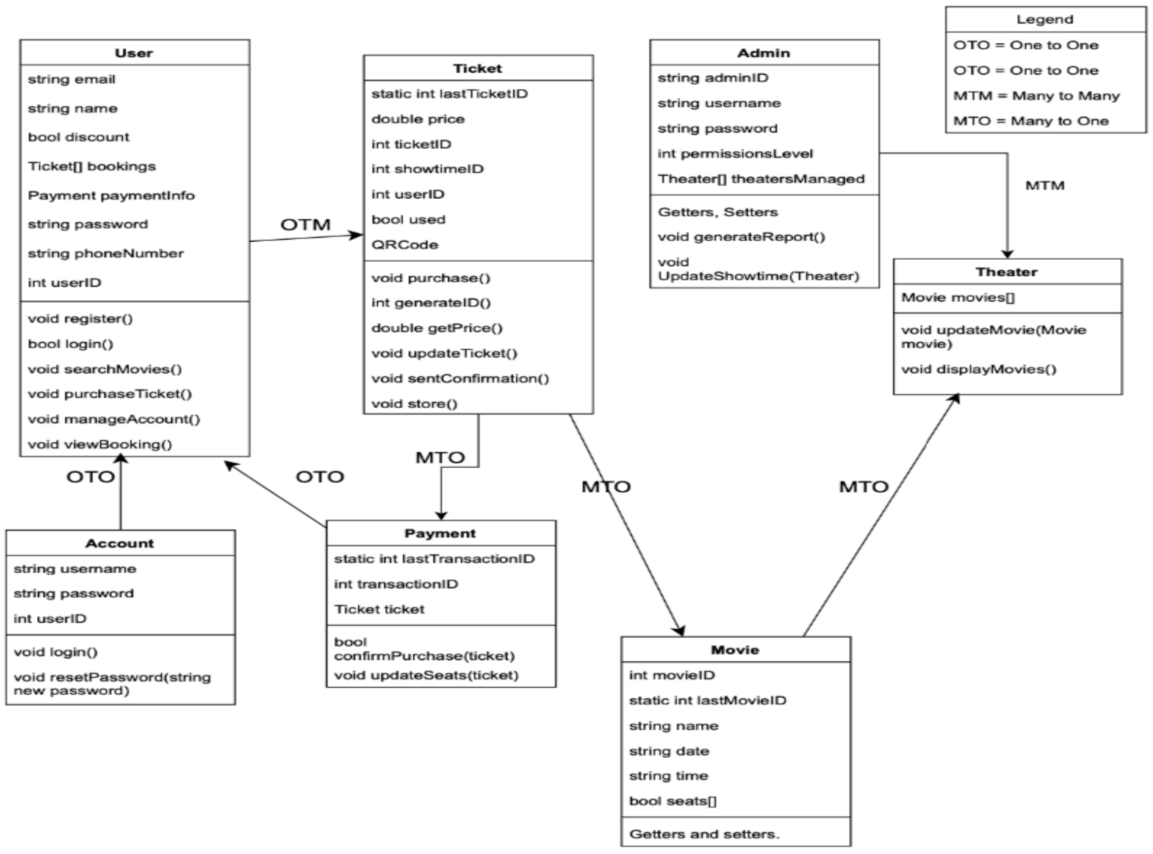


Figure 4: Movie Ticketing System UML Classes Diagram

Description

- Figure 4 shows how the different classes and functions will be implemented to accomplish customer-defined requirements.
- Class descriptions represent the structure and behavior of an object-oriented system.
- The core classes for our movie ticketing system are seen in the diagram above.
- **User:** Creating a class for the user to help us build a portfolio for them; we hold information such as payment method, passwords userId, etc.
- **Ticket:** The ticket class will hold everything from the movie to the movie theater to the time of the showing.

- **Account:** The account class holds the user's ID and password of said user.
- **Payment:** Payment class is used to describe the type of payment the user is using, as well as the method of payment.
- **Movie:** The movie class is where we will get information such as the name of the movie, the movie ID, as well as things like the time of the movie and the seats that are available for the movie.
- **Theater:** In the theater class, we will find the movies that are playing at the theater.

3.7.3 Attribute Description

- The names and data types of these attributes are clear and precise, for example, string email, int userId, bool discount.
- Data Flow and usage arrows specify if it is one client to one client (OTO) or if multiple clients worth of data is being handled at the same time (OTM)

3.7.4 Operation Description

- PurchaseTicket: This operation handles the entire ticket purchasing process, including seat selection, payment processing, and ticket generation. It then moves to payment processing, ensuring that all transactions are securely handled via supported payment methods (credit/debit cards, e-wallets, etc.). After successful payment, the system generates a digital ticket, which is sent to the user via email or available for download within the system. This is also where any gift cards or limited-time deals are applied.
- CancelTicket: This operation allows users to cancel their tickets and initiates the refund process. This process also highlights whether or not a user is eligible for a refund given the criteria set out by each individual theater. (I.E. If organizations want to accept unused tickets after a movie showing has already passed.)
- UpdateShowtime: This operation enables administrators to add, modify, or remove movie showtimes and seat availability. They can also reschedule showtimes, cancel sessions, or update pricing tiers based on factors like peak hours. Additionally, this function supports notifications to users who have already purchased tickets for any affected showtimes.
- GenerateReport: This operation generates various reports for administrators, such as sales reports, popular movie reports, and user demographics. Reports include sales reports (daily, weekly, monthly, or custom ranges), showing revenue from ticket sales, popular movie reports, ultimately indicating which films are attracting the most viewers.
- Login: This operation authenticates users and administrators, granting them access to the system. It also allows users to reset their password and redirect users to register if it is their first time seeing a movie.
- Register: This operation allows new users to create accounts in the system. It also allows users the functionality to apply previously bought tickets to future purchases on their account thereafter.
- SearchMovies: This operation enables users to search for movies based on title, genre, actors, and keywords.

3.7.5 Tasks, Responsibilities, and Timeline

- Frontend Development: Victor Allen - 2 weeks
- Backend Development: Teague Sangster - 4 weeks
- Database Design: Chad Childers - 2 weeks
- Integration: Trevor Plott - 3 weeks from completion of the module
- Testing: All - 6 weeks

3.8 Logical Database Requirements

The Movie Ticket System will use a database to store and manage persistent data. The database will handle a large amount of structured and unstructured data, requiring a robust and scalable database solution.

- Data Formats: The database will primarily store data in formats including text, numbers, dates, and boolean values.
- Storage Capabilities: The database must be able to store a large volume of data, including movie information, showtimes, user profiles, bookings, and payment records.
- Data Retention: Data retention policies will be implemented to ensure compliance with legal and business requirements.
- Data Integrity: Data integrity is crucial for the system's accuracy and reliability. Mechanisms like data validation, constraints, and backups will be implemented to maintain it.
- Accessibility and Performance: The database must provide fast and efficient data access to support real-time transactions and queries from the application.
- Security: The database will implement robust security measures to protect sensitive data, including user information and payment details. Access controls, encryption, and regular security audits will be employed.

4. Test Cases

4.1 Test Case Description

- This will cover and outline the testing approach and define individual test cases that are designed to demonstrate the functionality, performance, and usability of the MTS. It will cover different scenarios, including interactions with users, system responses, and the handling of errors. This will ensure that the system meets the requirements and delivers a proper user experience.

4.2 Testing Approach

- **Functional Testing:** this will verify that all features and promised functionality of the MTS work as intended.
- **Usability Testing:** This will evaluate the usability of the system from a user perspective, looking for ease of navigation and ease of use.
- **Performance Testing:** This will assess the system’s response time, ability to handle heavy loads, and average performance over different user loads.
- **Security Testing:** This will identify and address vulnerabilities to ensure PII is kept safe.
- **Compatibility Testing:** This will ensure the system functions correctly across different browsers and devices.

4.3 Test Case Categories

- **User Registration and Authentication:**
    - Successful registration and login
    - Invalid login attempts.
    - Password reset functionality
    - 2FA authentication verification
- **Movie Browsing and Selection:**
    - Search for movies by title, genre, actors, ect.
    - Filtering and sorting movie listings
    - Viewing movie details.
    - Checking Showtimes and availability.
- **Ticket Booking and Payment**:
    - Selecting seats from the map.
    - Handling seat availability conflicts
    - Processing payments
    - Generating and delivering digital tickets.
- **Account Management**:
    - Updating user profile information
    - Managing payment methods
    - Viewing history and ticket details
    - Canceling bookings and refund requests.
- **Admin Functionality**:
    - Adding, modifying and removing movie listings.
    - Managing showtimes and seat availability.
    - Handling user inquiries and support requests.
    - Generating sales reports and analytics.
- **Error Handling and Edge Cases:**
    - Handling invalid reports and analytics.
    - Graceful degradation during system failures or outages
    - Testing for security vulnerabilities
    - Verifying data integrity and consistency.

4.4 Test Case Execution and Reporting

- Test cases will be executed manually and be automated wherever possible.
- Test results will be documented and analyzed to fix any issues.
- Regular testing will occur to ensure new changes or updates do not introduce new bugs.

4.5 Test Case Scenarios

4.5.1 Test #1 - Successful Movie Search (Unit)

- This test is designed to test the search functionality of the MTS system. The function responsible for returning correct results, called searchMovies(), will be used to test if the function is working as intended. The tester will need to ensure that the movie "Minions" and any other movies related to the keyword “Minions”, such as “Despicable Me”, is in a predefined list the search function has access to. The tester will then call the searchMovies(“Minions”) function and analyze the results. If the predefined list is returned, the test is a pass.

TestCaseID	**Component**	**Priority**	**Description/Test Summary**	**Pre- requisite**	**Test Steps**	**Expected**	**Actual Result**	**Status**	**Test Executed By**
Movie Search	Search Module	P0	Verify that the searchMovies() function returns the correct movie results when a user searches using specific titles or keywords.	1. The database that the search function looks through has the movies “Minions” and “Despicable Me” as well as the sequels.	1. Call searchMovies(“Minions”) function 2. Check to see if a result is returned	Function returns a list of movies matching or has the keyword associated with "Minions"	A list with the "Minions" movie and "Despicable Me" as well as the sequels.	🟩 Pass	Trevor Plott

4.5.2 Test #2 - Failed Movie Search (Functional)

- This test case is designed to test if the search function properly shows the closest “matching results” when a spelling error occurs. The tester will navigate to the MTS page on a compatible browser, locate the search bar, type “Mnionz,” and then press enter. The search should result in movies like “Minions” and “Despicable Me.” If this does not occur, the test will fail.

TestCaseID	**Component**	**Priority**	**Description/Test Summary**	**Pre-requisite**	**Test Steps**	**Expected**	**Actual Result**	**Status**	**Test Executed By**
Movie Search	Search Module	P0	Verify that when a user types in a movie or related field and presses enter, search results populate.	1. MTS is opened in a browser	1. Type web address for MTS into approved browser 2. Once web page loads, type "Mnionz" into search bar 3. Press enter	Search that yields the "Minions" movie as well any movies tagged with the word "Minions"	Search with "Minions" movie and "Despicable Me" as well as the sequels.	🟥 Fail	Chad Childers

4.5.3 Test #3 - Successful Authentication (Unit)

- This test is designed to check if the MTS system is correctly authenticating users as they log in. The tester of this unit will need to verify that the authenitcateUser(username, password) method is returning “true” when the system is given a correct username and password. This test should begin with the teaser placing a known set of username and password into the database, such as username = “testUser” and password =

"password123." The tester will then call the function authenticateUser("testUser", "password123"). The authentication function will return "true" if the credentials exist.

TestCaseID	Component	Priority	Description/Test Summary	Pre- requisite	Test Steps	Expected	Actual Result	Status	Test Executed By
Login_01	Authentication_Module	P0	Verify that authenticateUser() function successfully and accurately authenticate a user with valid credentials.	1. The database contains username "testUser" and password "password123"	1. Call the authenticateUser("testUser", "password123") function	The function should return "true"	The function returned "true" indicating successful authentication.	🟩 Pass	Victor Allen

4.5.4 Test #4 - Failed Login with wrong password (Functional)

- This test aims to show that if the wrong password is provided, a failed login will occur, and access to the account will be denied. The tester will open a compatible browser, navigate to the MTS website, click on login, and then enter the correct username with the wrong password. The failed login should redirect to the login page again with an error message indicating that a failure has occurred with the username or password but not tell where the problem lies.

TestCaseID	Component	Priority	Description/Test Summary	Pre-requisite	Test Steps	Expected	Actual Result	Status	Test Executed By
Login_02	Login_Module	P1	Verify that the user can not log in to manage their account and purchase tickets.	1. MTS is opened to the home screen in a browser. 2. A valid user account has been created	1. Navigate to the login page from the home page. 2. Enter the user's username in the username field 3. Enter an incorrect password in the password field 4. Press the login button.	If the user provides incorrect password information but the proper username, an error that states something is wrong with the username or password has occurred.	An error occurs, redirecting the user back to the login screen with an error stating an issue with the username or password occurred. Please try again.	🟩 Pass	Trevor Plott

4.5.5 Test #5 - Failed Login with wrong username (Functional)

- This test aims to show that a failed login will occur if the wrong username is provided, and access to the account will be denied. The tester will open a compatible browser, navigate to the MTS website, click on login, and then enter the incorrect username with the correct password. The failed login should redirect to the login page again with an error message indicating a failure with the username or password but not telling where the problem lies.

TestCaseID	Component	Priority	Description/Test Summary	Pre-requisite	Test Steps	Expected	Actual Result	Status	Test Executed By
Login_03	Login_Module	P1	Verify that the user can not log in to manage their account and purchase tickets.	1. MTS is opened to the home screen in a browser. 2. A valid user account has been created	1. Navigate to the login page from the home page. 2. Enter an incorrect username in the username field 3. Enter the user's password in the password field 4. Press the login button.	If the user provides incorrect username information but the proper password, an error that states something is wrong with the username or password has occurred will be displayed.	An error occurs, redirecting the user back to the login screen with an error stating an issue with the username or password occurred. Please try again.	🟩 Pass	Chad Childers

4.5.6 Test #6 - Failed Login with wrong username and password (Functional)

- This test aims to show that a failed login will occur if the wrong username and password are provided, and access to the account will be denied. The tester will open a compatible browser, navigate to the MTS website, click on login, and then enter an incorrect username and password. The failed login should redirect to the login page again with an error message indicating a failure with the username or password but not telling where the problem lies.

TestCaseID	Component	Priority	Description/Test Summary	Pre-requisite	Test Steps	Expected	Actual Result	Status	Test Executed By
Login_04	Login_Module	P1	Verify that the user can not log in to manage their account and purchase tickets.	1. MTS is opened to the home screen in a browser. 2. A valid user account has been created	1. Navigate to the login page from the home page. 2. Enter an incorrect username in the username field 3. Enter an incorrect password in the password field 4. Press the login button.	If the user provides incorrect username and password information, an error that states something is wrong with the username or password has occurred will be displayed.	An error occurs, redirecting the user back to the login screen with an error stating an issue with the username or password occurred. Please try again.	🟩 Pass	Victor Allen

4.5.7 Test #7 - Password Reset (Functional)

- This test will demonstrate the integration of password management into the login module. The tester will already be logged into the account management page.

When successfully logged in, they will navigate to user settings, click on the password reset option, and then follow the prompts to reset their password. The tester should be prompted to enter the old password, followed by two correct inputs for the new password. After a successful password reset, the user will be logged out and redirected to the login screen.

TestCaseID	**Component**	**Priority**	**Description/Test Summary**	**Pre-requisite**	**Test Steps**	**Expected**	**Actual Result**	**Status**	**Test Executed By**
Password Reset	User_Home	P3	Verify that once logged in, the user is able to reset their password	1. A valid user account has been created 2. The user is logged in.	1. In the user home menu, navigate to the user settings sub-menu and select reset password. 2. Follow the prompts to reset the user's password.	The user will be prompted to enter their password and then their new password twice. After this, they will be asked if they are sure, and their stored password will become the new password.	The user followed the prompts and their password was successfully reset to a new password of their selection.	Pass	Victor Allen

4.5.8 Test #8 - Refund Test (System)

- This test will demonstrate the integration of the refund system with both the bank's systems and the MTS. The tester will already have a ticket reservation for a movie. They will navigate to their account page, then to the ticket confirmation page, where they will select the refund ticket option. The user will be prompted to enter their payment information for the refund, then will be asked to confirm their refund, provided a refund is allowed to be issued. After a successful refund the user's reservation will be canceled, they will be directed back to the home page, and the bank will issue a refund.

Refund Guest Ticket	Integrated Bank and Payment System	P0	Verify that Refund email is received after a user refunds a movie ticket. Email should contain a link to cancellation/ refund details.	1. A ticket has been purchased by a user. 2. The user has not yet seen the movie. 3. The date of the movie has not passed	1. Log into previous purchase either from the account page, or ticket confirmation link. 2. Verify you are human. 3. Click submit. 4. Click the refund ticket. 5. Select refund type (Payment on purchasing card, or account credit) 6. Press confirm	The user will get an Email confirming that their guest ticket has been refunded.	Movies were canceled and payment was refunded.	PASS	Teague Sangster

4.5.9 Test #9 - (Functional)

- This test ensures that users can share their movie ticket details, including the seat number, time, location, without revealing sensitive information like the ticket holder's name that could be used to steal the ticket. After purchasing a ticket, the customer can log into their account and/or use the provided confirmation link, then complete human verification, select the specific ticket and finally how they want to share. The site then generates and copies to the user's clipboard a shareable link that allows others to view the details of that showing. The test verifies that first the sharing function is secure and second that the link provided has the appropriate information. It passed successfully, confirming that tickets can be shared as expected.

Share Ticket	Movie Details	P3	Verify that a user can send a link of their ticket's seat #, time, and theater location. Should not give out the ticket holder's name, ensuring tickets cannot be stolen from sharing.	1. A ticket has been purchase d by a user	1. Log into previous purchase either from the account page, or ticket confirmation link. 2. Verify you are human. 3. Select the specific ticket to be shared. 4. Click the share button. 5. copy the link	A link should be provided that will navigate to the share showing	A link to the showing was shared	PASS	Teague Sangster

4.5.10 Test #10 - (System)

- This test ensures that a user can purchase a ticket for the same movie showing as the one received in a shared ticket link. Upon clicking the shared link, the user can see the details of the showing, especially the seat their friend has selected. The test then ensures they can proceed to select their own seat and the number of tickets they wish to purchase. (The test also ensures that the seat the share was from is visible but disabled on the same purchase map). After selecting ticket types (e.g., Child, Adult, Senior) and confirming the seat visibility, the user continues through the purchase process. The test ensures that the shared ticket functionality enables users to make their own purchase with visibility of their friend's seat, and that the transaction can be successfully completed.

Purchase from shared ticket	Movie Details	P3	Verify that given a shared ticket link, a user can Navigate to the same showing, and see their "Friends" seat while also being able to select their own seat for purchase.	1. A link to a purchased ticket has been shared	1. Click the linked shared. 2. See the showing details 3. Select "Purchase for this showing". 4. Select your seat purchase amount (I.E. Child, Adult, Senior; as well as amount of tickets to purchase) 5. Press Continue 6. Confirm ability to see the seat that was shared (The seat the "Friend" has) 7. call the "Ticket Purchase test" to continue from here	The link should take the user to a showing, and show the seat their "Friend" has. Tickets should be able to be purchased while seeing their seat.	Shared seat was visible, Ticket was purchased successfully	PASS	Teague Sangster

4.5.11 Test #11 - End-to-End System Test (System)

- This test is designed to show the functionality of the entire MTS system, including third-party payment platforms. The tester will begin by navigating to the MTS website and selecting a movie or theater. Once those selections have been made, the tester can then pick a time and seat. They will then move to the purchase page, where they will enter payment information and confirm they are not a robot. After purchase, confirmation should be sent to both the account management page and to the email associated with the user's account.

TestCaseID	**Component**	**Priority**	**Description/Test Summary**	**Pre- requisite**	**Test Steps**	**Expected**	**Actual Result**	**Status**	**Test Executed By**
System Test	System	P0	Verifies that a user can connect to the website, choose a movie, movie time, and seat selection, as well as test the purchase functionality and integration with third-party payment platforms.	1. A valid user account has been created 2. The user is logged in	1. Navigate to MTS website 2. Select a movie 3. Select a showing time 4. Select a seat 5. Enter payment info 6. Confirm payment info 7. Complete CAPTCHA to ensure human user 8. Navigate to account management to check for movie information and check email for movie details and confirmation	Users will be able to select a movie, a desired time, a seat, and pay for it.	Movie, time, and seats were all easily selected. Payment went through without issue and confirmation was sent to both the account and the email associated with the logged in user.	Pass	Chad Childers

4.6 Test Case Spreadsheet.

- See section A.2

4.7 Assignment 3 Git Repository Links.

Chad -
https://github.com/Xymos1/SRS/blob/main/Movie%20Theatere%20Platform%20Group%208.docx%20(1).pdf
Trevor-
https://github.com/Tplott/MTTS-CS_250-Test-Cases/blob/869a6ae8f43ff4f57b590084cd9ce9fbd8300ddd/Copy%20of%20Movie%20Theatere%20Platform%20Group%208.docx-2.pdf
Teague - https://github.com/TeagueSS/CS250
Victor - https://github.com/victoralien/CS250/tree/main/Assignment03

5. Analysis Models

5.1 Architecture Diagram

- See section 3.7.1

5.2 Data Management Strategy

5.2.1 General Description

The data management strategy for the Movie Ticketing System (MTS) will prioritize security, scalability, and real-time accessibility. The system will utilize a combination of database technologies to effectively manage the diverse data requirements of the application.

- Primary Database (Relational Database): A relational database (e.g., MySQL, PostgreSQL) will serve as the primary data store for structured data such as:
    - User information (account details, purchase history)
    - Movie information (titles, descriptions, genres, cast)
    - Theater information (locations, seating layouts)
    - Showtime schedules
    - Booking records
    - Payment transactions
- Caching Layer (Redis): A high-performance in-memory data store like Redis will be employed as a caching layer to improve system performance and responsiveness. Redis will store frequently accessed data, such as:
    - Seat availability for active showtimes
    - Popular movie listings
    - Real-time pricing information
- Cloud Storage (Google Cloud Storage): For storing large files like movie posters, trailers, and other media assets, Google Cloud Storage will be utilized. This will offload the storage burden from the primary database and provide efficient access to these resources.

Key Considerations:

- Data Security: Robust security measures will be implemented to protect user data, including encryption, access controls, and regular security audits.
- Scalability: The database architecture will be designed to accommodate a growing user base and increasing data volumes.

- Real-time Updates: The system will ensure real-time updates to critical data, such as seat availability and booking status, to provide accurate information to users.
- Data Consistency: Mechanisms will be in place to maintain data consistency across the different data stores.
- Backup and Recovery: Regular backups and disaster recovery procedures will be implemented to safeguard against data loss.

5.2.2 Relational Database (SQL)

- The SQL database is controlled using a structured query language that will arrange the data to create tables in a predefined pattern. Using SQL will ensure ACID compliance for transaction integrity.

5.2.3 Single Database

- Using a single database offers the advantage of centralized data management, simplifying maintenance, backups, and data consistency. In this approach, all tables required for the applications are created within one database. This setup allows easy querying across tables and promotes data integrity. A single database structure is ideal when dealing with a limited dataset or applications where real-time performance or extensive scalability isn't a primary concern. Using a single database provides easy querying and data integrity,

ensur

5.2.4 Table (Database) Structure

- The table structure should be designed based on the application's specific requirements, ensuring normalization to avoid redundancy. Below is a basic outline for structuring the tables, assuming an application that involves managing entities like "Users," "Bookings," and "Showtimes":

Users Table

- user_id (Primary Key)
- username
- email

Movie Table

- movie_id (Primary Key)
- title
- description

Ticket Table

- Ticket_id (Primary Key)
- user_id (Foreign Key referencing Users)
- total_amount

5.2.5 Foreign Key Relationships

- Foreign keys help maintain data consistency across related tables by enforcing a relationship constraint between columns. Here's how foreign key relationships work in this structure:

  Orders Table and **Orders Table**: The `user_id` column in the `Orders` table is a foreign key that references the `user_id` primary key in the `Users` table, establishing a relationship between orders and the user who placed them.

5.3 Trade Off Discussion

5.3.1 Technology Choice

- Because SQL databases have strong, well defined relationships, and incorporate ordered data, they allow for effective advanced searches and robust data security. NoSQL may not be more stable, but its flexibility, while useful, does not make it better than using SQL.

5.3.2 Single Database vs Multiple Databases

- A single database is easy to maintain, and support. It also allows for easier synchronization across the MTS, but it may become inefficient with scaling. Having multiple databases makes the system better with workload distributions, the difficulty in maintenance makes it not as appealing as a single database.

5.3.3 Table Organization

- If the table was organized by entities and their relationships to other entities, there would be improved data accuracy and improved query efficiency. To achieve this however, careful preparation and attention to detail is required to prevent the addition of components that slow the process down. A NoSQL database is useful for system components that need to be scaled. However since the movie theater has a limited number of screens, scalability is not a large concern.

6. Change Management Process

- The CMP (Change Management Process) will ensure that any changes to the MTS will be implemented in a controlled and professional manner. The process will maintain the system's integrity, minimize any disruptions, and ensure that changes are aligned with the goals of the project, as well as user needs.

6.1 Steps in the CMP

- **Change Request:** Proposed changes to the MTS, will be documented in a formal Change request that will include a description of the change, its rationale, and possible impacts and risks.
- **Change Assessment:** A designated Change Control Board (CCB) will review the request. They will evaluate its feasibility, necessity, and its alignment with the project. The CCB will assess all possible implications of the change.
- **Impact Analysis:** An impact analysis will be conducted in order to understand the consequences of the change(s) on the system's functionality, performance, user

experience, ect. This analysis will help to identify any disturbances or conflicts with additional features.

- **Implementation:** The development team will implement the approved change. They will follow established, standard coding methods and best practices. Testing will be conducted to ensure that the changes are done correctly and they do not incur new issues.
- **Documentation:** All changes will be documented including updates to the SRS, the design documents, and test cases.
- **Deployment:** The changes system will be deployed to the production environment following a controlled release process. User acceptance testing will be conducted to validate the change(s) in a real world setting.
- **Post Implementation Review:** After Deployment the CCP will conduct a post-implementation review to assess the effectiveness of the change and its impact. Lessons will be documented for future reference.

6.3 Change Control Board (CCB)

- The CCB will be responsible for overseeing the Change Management Process. It will consist of representatives from different stakeholders, including project managers, developers, testers, and user representatives. The CCB will ensure that changes are evaluated objectively and implemented in a controlled manner.

7. Assignment 4 Git Repository Links.

Chad -
Trevor-
Teague -
Victor -

5.x Sequence Diagrams - unused?

5.x Data Flow Diagrams (DFD) - unused?

5.x State-Transition Diagrams (STD) - unused?

A. Appendices

A.1 Appendix 1

- https://www.geeksforgeeks.org/how-to-write-a-good-srs-for-your-project/

A.2 Appendix 2

- https://docs.google.com/spreadsheets/d/1l3kgVSHLP8WCGsDr1JW-pziX6GAZNXAAJYWlcTXFxq0/edit?gid=0#gid=0