Movie Theatre Ticketing System (MTTS)

Software Requirements Specification

Version 1

9-19-2024

Group #7
Matthew Kloth
Doan Quoc Tien Nguyen
Jingyi Chen
Julio Nevarez

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

Revision History

Date	Description	Author(s)	Comments
9/19/24	Version 1	Matthew Kloth Doan Quoc Tien Nguyen Jingyi Chen Julio Nevarez	First Revision
10/10/24	Included Design section: https://github.com/1400Pika/MTTS/blob/main/MTTS%20Software%20Design%20Specification.pdf	Matthew Kloth Doan Quoc Tien Nguyen Jingyi Chen Julio Nevarez	
10/24/24	Included Test Plan document and Excel Test Case Samples	Matthew Kloth Doan Quoc Tien Nguyen	Test Plan document lays out test plan for verification and validation. Also includes an

	https://github.com/1400Pika/MTTS	Jingyi Chen Julio Nevarez	excel test case document with test case samples
11/7/24	Added Data Mgmnt. Strategy	Matthew Kloth Doan Quoc Tien Nguyen Jingyi Chen Julio Nevarez	

Table Of Contents

1. Introduction

The purpose of this Software Requirements Specification (SRS) is to provide a detailed description of the Movie Theater Ticketing System (MTTS), outlining the system's intended functionalities, interfaces, and interactions. The MTTS allows users to browse, select, and purchase movie tickets, order concessions, and manage subscriptions to newsletters containing information on upcoming movies and promotional deals. This document covers all features and constraints of the system and serves as a reference guide throughout the software development lifecycle. The introduction section provides an overview of the entire SRS document, ensuring a shared understanding of the system's purpose, scope, definitions, and key references, which are elaborated on in subsequent sections. The remainder of the document is organized to describe specific software functionalities, performance metrics, and interaction guidelines in detail.

1.1 Purpose

The purpose of this document is to define the software requirements for the Movie Theater Ticketing System (MTTS). The system will allow users to purchase movie tickets, order concessions such as food and drinks, receive promotional deals and subscribe to newsletters for upcoming movies and special deals. This document outlines the functionality, constraints, and objectives of the system, ensuring a shared understanding among the project stakeholders.

1.2 Scope

The software product to be developed is the Movie Theater Ticketing System (MTTS). This system will allow users to perform the following tasks:

1. Purchase Movie Tickets: Users can browse available movies, view showtimes, select preferred seating, and complete ticket purchases online or via a mobile device.
2. Order Concessions: The system will enable users to pre-order food and drinks, including access to promotional deals, to be picked up at the concession stand upon arrival.
3. Newsletter Subscriptions: Users can sign up for email newsletters to receive updates on upcoming movies, exclusive deals, and concession offers. This feature will also support targeted marketing efforts by the theater.
4. User Account Management: Users can create accounts to store preferences, view past transactions, and manage newsletter subscriptions.

   **The system will not handle third-party payment processing directly but integrate with an external payment gateway for secure transactions.

Application of the Software:

(a) Benefits, Objectives, and Goals:

- Efficient Ticket Purchase: The MTTS will allow users to quickly browse, select, and purchase tickets with a seamless, user-friendly interface, with an expected average completion time of 2-3 minutes per transaction.

- Pre-order Concessions: Users can add concession items to their orders in a single transaction, minimizing wait times upon arrival at the theater.
- Targeted Email Campaigns: By subscribing to newsletters, users will receive personalized emails based on their movie preferences and previous purchases, with options for exclusive deals on food and drinks.
- Real-Time Seat Selection: The system will provide real-time seat availability, ensuring accurate seat reservations for users.

(b) Consistency with Higher-Level Specifications:
This software product aligns with the theater's broader goals of enhancing the customer experience through digital engagement and increasing revenue through pre-sales of tickets and concessions. These goals reflect the theater's strategic objectives of improving customer retention, streamlining operations, and increasing sales through targeted promotions.

1.3 Definitions, Acronyms, and Abbreviations

MTTS: Movie Theater Ticketing System – The software being developed for managing movie ticket purchases, concessions, and email subscriptions.
UI: User Interface – The part of the software that users interact with directly.
UX: User Experience – The overall experience of a person using the MTTS, especially in terms of how easy or pleasant it is to use.
API: Application Programming Interface – A set of routines, protocols, and tools for building the software and allowing it to communicate with other services, such as payment gateways.
Newsletter: A recurring email sent to users who subscribe, providing information about upcoming movies, promotions, and deals on concessions.
Payment Gateway: A third-party service that securely processes payments for the purchase of movie tickets and concessions.
Concessions: Food and drinks are available for pre-order or purchase within the theater.
User Account: A profile created by the user that stores personal information, preferences, and purchase history/order history.
Real-Time: The immediate processing and display of data, such as seat availability and ticket confirmation, as the system receives updates.
SRS: Software Requirements Specification – This document, outlines the requirements and expectations for the MTTS.

1.4 References

1. IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications
2. ISO/IEC 25010:2011, Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models
3. Payment Gateway API Documentation (e.g., Stripe, PayPal)
4. Movie Theater Concession Management Best Practices

1.5 Overview

The Movie Theater Ticketing System (MTTS) aims to revolutionize the user experience by offering an all-in-one platform for purchasing movie tickets, ordering concessions, and receiving personalized email newsletters. By simplifying the process of browsing available showtimes, selecting seats, and pre-ordering food and drinks, the system reduces the hassle often associated with traditional ticket buying. Additionally, users can create accounts to store their preferences, manage transactions, and subscribe to targeted newsletters featuring exclusive deals and movie updates. The MTTS is designed to streamline operations for theaters while enhancing customer engagement through real-time ticketing and targeted marketing efforts.

2. General Description

The Movie Ticketing System is designed to provide users with an efficient and intuitive platform to purchase movie tickets, select seats, and explore additional offerings such as food, drinks, and exclusive deals. This system will also include a feature for users to create accounts, allowing them to receive personalized newsletters and notifications about upcoming movies, special events, and promotions.

2.1 Product Perspective

Standalone vs Integrated: This system could be a standalone system or part of a more extensive system that handles cinema management, including inventory, staffing, and promotions.
Interfaces
Online payment systems (e.g., credit card or PayPal).
Movie databases (for movie information, showtimes).
User interfaces: Web app, mobile app, or kiosk-based interaction.

2.2 Product Functions

User Account Creation: The User Account Creation feature allows users to sign up for personalized experiences, track their past purchases, and access exclusive offers.
Browse Movies: Users can view movie listings with details like showtimes, genre, rating, and available seats.
Select Movie and Showtime: Users can choose a specific movie and time for booking.
Seat Selection: Users can choose their preferred seats from a real-time seating chart.
Ticket Purchase: Users can purchase tickets using various payment options (credit card, debit card, digital wallets).
Generate Ticket: A digital or printable ticket is created, with an option to send it via email or mobile app.
Cancel or Modify Booking: Users should be able to cancel or reschedule their booking (based on availability and policy).
Add-on Purchases： (Food, Drinks, and Deals) The Add-on Purchases feature allows users to enhance their moviegoing experience by pre-ordering concessions and taking advantage of special deals or combos.

Newsletter and Notifications: The Newsletter and Notifications feature allows the cinema to communicate directly with users, providing them with personalized updates, offers, and reminders.

2.3 User Characteristics

General Users/Customers:
Demographics: All age groups, familiar with basic web and mobile app usage.
Tech Skills: Minimal tech skills, should be able to navigate the system easily.
Goals: They want a fast, reliable, and simple interface to buy movie tickets.

Administrative Users (Cinema Employees):
Demographics: Cinema staff members, familiar with more complex system functions.
Tech Skills: Moderate skills, able to manage showtimes, movies, and bookings.
Goals: Efficiently manage cinema operations related to movie listings and ticket sales.

2.4 General Constraints

System performance: The system must provide a fast and smooth user experience and be able to handle a large amount of users interacting with it simultaneously without performance degradation.
Scalability: Cloud-based infrastructure (e.g., AWS, Google Cloud, Azure) requires that the database can scale as the number of users and transactions grows.
Compatibility: The system should be compatible with multiple platforms (web, mobile) and different browsers.
Security: Compliance with payment regulations and protect user information are crucial.
Data Storage: The system needs to store user information, payment history, and bookings securely.
Legal Requirements: The system must comply with local regulations regarding ticket refunds, cancellations, age restrictions for certain movies (e.g., enforcing age verification for restricted films), and privacy policies.
Integration with External Systems: The system depends on reliable integration with third-party services such as payment gateways, movie databases, and email marketing platforms.
Time Constraints: The system must be developed within a set timeframe to meet cinema business requirements, such as the launch of new features in time for peak movie seasons.

2.5 Assumptions and Dependencies

Assumptions
Stable Internet Connection:
It is assumed that users will have a stable and reliable internet connection when accessing the system via a web browser, mobile app, or cinema kiosk.

Impact if Invalid: Without stable internet, users may face difficulties completing transactions or loading movie information, leading to a poor user experience.

Third-Party Payment Gateway Availability:
The system assumes that third-party payment gateways (e.g. PayPal) will be available and functioning properly. These gateways handle the financial transactions securely and ensure payment processing is completed in a timely manner.
Impact if Invalid: If the payment gateways are down or experiencing latency issues, users may be unable to complete purchases, resulting in lost revenue and customer dissatisfaction.

Availability of Movie Databases:
The system assumes that third-party movie databases (e.g., those providing showtimes, movie descriptions, and trailers) will supply updated information about upcoming and current movies. These databases ensure that users can browse accurate and up-to-date movie listings.
Impact if Invalid: If movie data is outdated or unavailable, users may not see the correct showtimes, causing confusion or incorrect bookings.

Browser and Operating System Compatibility:
It is assumed that users will access the system through modern web browsers (e.g., Chrome, Firefox, Safari) and common operating systems (e.g., Windows, iOS, Android). The system is designed to function optimally on these platforms.
Impact if Invalid: If users access the system on outdated browsers or less common operating systems, they may experience compatibility issues.

User Willingness to Create Accounts:
It is assumed that a significant number of users will choose to create accounts to receive benefits like personalized offers, booking history, and newsletters. This will help increase customer engagement and retention.
Impact if Invalid: If users are unwilling to create accounts, the system’s marketing and loyalty features may not reach their full potential, limiting personalized offers and repeat customer interaction.

Customer Email Accuracy:
It is assumed that users will provide accurate and valid email addresses during registration for receiving important communications such as booking confirmations, newsletters, and special offers.
Impact if Invalid: If users enter incorrect email addresses, they may not receive their booking confirmations or marketing materials.

Cinema’s Operational Policies Remain Consistent:
The system assumes that the cinema’s operational policies (e.g., refund/cancellation policies, age verification for restricted movies) will remain consistent over time. These policies are embedded in the system's workflows.
Impact if Invalid: If cinema policies change (e.g., extending refund periods or modifying age restrictions), the system would need updates to reflect these changes, which could delay functionality or confuse users.

Dependencies
Payment Gateways: The Movie Ticketing System depends on external payment processors (e.g., PayPal) for secure and reliable payment handling. These services are responsible for processing payments and ensuring transaction security.
Dependency Risks: Downtime: If the payment gateway experiences downtime, users won't be able to complete transactions, resulting in failed bookings.
Fees: Changes in the fees or policies of these gateways can impact the system's pricing model.

Movie Database and APIs: The system depends on external APIs for pulling movie data, including showtimes, movie details, cast information, trailers, and reviews. These services ensure that users are presented with accurate and up-to-date movie information.
Dependency Risks: Data Availability: If the movie database or API becomes unavailable, outdated, or slow, users may not see correct or current movie listings and showtimes.

Email and Notification Services: The system relies on third-party email services for sending booking confirmations, newsletters, and promotional offers. Push notifications and SMS alerts may also depend on external communication platforms.
Dependency Risks: Delivery Issues: If the email service experiences delays or spam issues, users may not receive critical information.

Web Hosting and Infrastructure Services: The system depends on web hosting and infrastructure providers for server uptime, data storage, and performance. These services are responsible for ensuring that the website and mobile app remain accessible.
Dependency Risks: Server Downtime: If the hosting service goes down or experiences performance issues, the movie ticketing platform will be unavailable to users.
Data Security: Any breaches or failures in the infrastructure provider's security could compromise user data, requiring swift action from the cinema's side to protect customer information.

Mobile App Stores: If the system includes a mobile app, it depends on distribution through the Google Play Store and Apple's App Store. These platforms govern how users download and update the app.
Dependency Risks: Approval Delays: Updates to the mobile app may be delayed due to the app store review process.
Store Policies: Changes to app store policies could force modifications to the app or business model.

Legal and Regulatory Frameworks: The system must comply with various laws and regulations, including data protection laws and payment security standards. Additionally, age restrictions for certain movies must be enforced according to local regulations.
Dependency Risks: Legal Changes: If there are changes to privacy laws or ticketing regulations, the system may require updates, which could lead to downtime or rework.
Non-Compliance Penalties: Failing to comply with legal requirements could result in fines or penalties, potentially impacting the cinema's operations.

3. Specific Requirements

This section outlines the detailed requirements for the Movie Theater Ticketing System (MTTS). It includes external interface requirements, functional requirements, and use cases to demonstrate the interaction between users and the system.

3.1 External Interface Requirements

3.1.1 User Interfaces

The Movie Theater Ticketing System (MTTS) will provide the following user interfaces:

- Web-based Interface: Users can access the MTTS through a web browser. The interface will display available movies, showtimes, a real-time seat selection chart, payment options, and order history. The design will be responsive to work on both desktop and mobile browsers.
- Mobile Interface: The MTTS will be optimized for mobile web browsing, providing similar functionality to the desktop version, ensuring smooth navigation and interaction.
- Real-Time Seat Selection Interface: A graphical interface that allows users to view and select available seats in a movie theater. The interface will update in real-time to show seats that have been taken.
- Payment and Checkout Interface: A secure checkout interface where users enter their payment details and confirm their purchases. This screen will also allow users to review their movie choices and concession orders.

3.1.2 Hardware Interfaces

The system will interact with the following hardware:

- Theater Kiosks: Kiosks in the theater will allow users to print tickets or check-in by scanning their QR codes.
- Personal Devices: Users will access the MTTS via personal devices such as desktop computers, tablets, or mobile phones.
- POS Systems: The system will interface with point-of-sale systems at the theater to handle concession sales, ticket scanning, and in-person payments.

3.1.3 Software Interfaces

The system will integrate with the following external software systems:

- Payment Gateway (e.g., PayPal, Stripe): To process online payments securely.
- Movie Database API: To pull movie showtimes, descriptions, ratings, and other details.
- Email Services (e.g., SendGrid): To send confirmation emails for ticket purchases, newsletters, and special promotions.

3.1.4 Communications Interfaces

The system will handle communication through:

- Real-time Updates: The system will provide real-time updates for seat availability and showtime changes.
- Email Notifications: The system will send users confirmation emails, promotional newsletters, and updates about upcoming movies or special offers.

3.2 Functional Requirements

The functional requirements of the MTTS are listed below. These define the key features and behavior of the system:

3.2.1 Account Management

3.2.1.1 Introduction
Feature: The system will allow users to create and manage their accounts.
3.2.1.2 Inputs
User details (name, email, password).
3.2.1.3 Processing
The system will store user information securely, validate login credentials, and allow users to update their profile.
3.2.1.4 Outputs
Successful account creation, profile updates, and error messages for failed login attempts.
3.2.1.5 Error Handling
Error messages will be displayed for invalid input or existing accounts.

3.2.2 Movie Selection and Showtimes

3.2.2.1 Introduction
Feature: Users will be able to browse available movies and showtimes.
3.2.2.2 Inputs
Search criteria (movie name, genre, date, or theater).
3.2.2.3 Processing
The system will query the movie database and display relevant movies and showtimes.
3.2.2.4 Outputs
List of movies, showtimes, and theater details.
3.2.2.5 Error Handling
If no results are found, an appropriate message will be displayed.

3.2.3 Real-Time Seat Selection

3.2.3.1 Introduction
Feature: Users will be able to view and select available seats from a real-time seating chart.
3.2.3.2 Inputs

Movie selection, theater, showtime.
3.2.3.3 Processing
The system will check for seat availability and reserve the selected seats. Once seats have been selected, the selected seats will be reserved for 5 minutes till expiration.
3.2.3.4 Outputs
Confirmation of selected seats.
3.2.3.5 Error Handling
If a seat is unavailable or selected by another user, an error message will be displayed.

3.2.4 Ticket Purchase

3.2.4.1 Introduction
Feature: Users will purchase tickets using a secure payment gateway.
3.2.4.2 Inputs
Selected seats, payment details (credit card, PayPal, etc.).
3.2.4.3 Processing
The system will process the payment and confirm the purchase.
3.2.4.4 Outputs
A confirmation email and SMS text with a QR code for ticket access.
3.2.4.5 Error Handling
Failed payment transactions will result in an error message and a prompt to retry.

3.2.5 Concessions Pre-Order

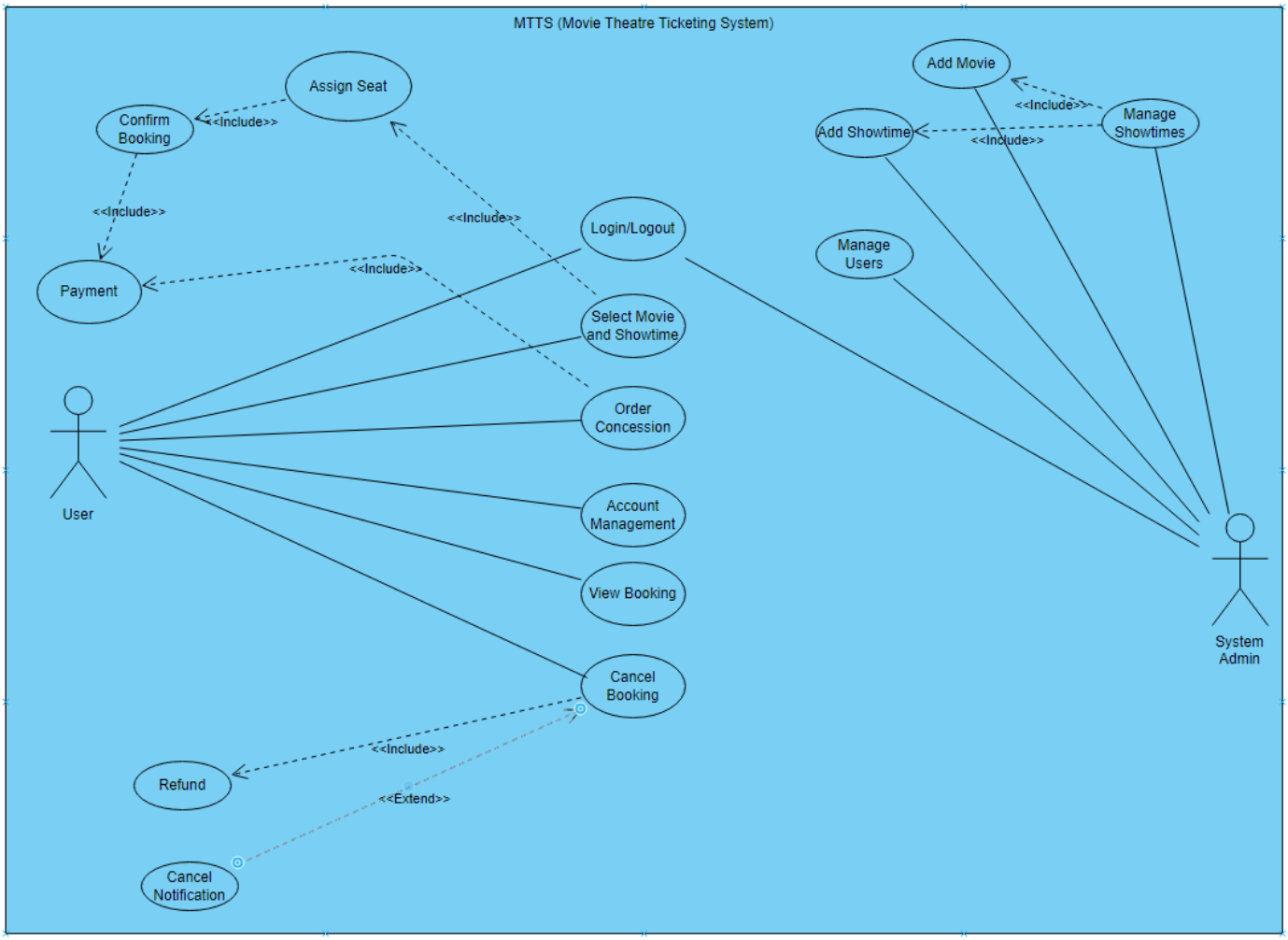
3.2.5.1 Introduction
Feature: Users can pre-order food and drinks through the system, either when purchasing a movie ticket or independently.
3.2.5.2 Inputs
Selected concessions (popcorn, drinks, combos).
3.2.5.3 Processing
The system will add the chosen concessions to the user’s order, process the payment, and notify the theater concession stand of the order.
3.2.5.4 Outputs
Confirmation of the order is provided to the user, the user's order history is updated to reflect confirmation of the order, and an order ticket is sent to the concessions staff for preparation.
3.2.5.5 Error Handling
If an item is unavailable, the system will notify the user and suggest alternative options.
Additional Details:
If the user orders concessions without purchasing a ticket, they can select a pickup time during the order process. Users will receive an email or mobile notification with the details of their concession order and the scheduled pickup time.

3.2.6 Queueing System for High Traffic

3.2.6.1 Introduction
Feature: During high-demand periods, the system will implement a queue to manage user load.
3.2.6.2 Inputs
User requests during peak traffic times.
3.2.6.3 Processing
The system will place users in a virtual queue and allow them to enter the system in order.
3.2.6.4 Outputs
User position in queue and estimated wait time.
3.2.6.5 Error Handling
If the system is overloaded, users will be shown an error message and prompted to try again later.

3.3 Use Cases

3.3.1 UML Use-Case Diagram



3.3.2 Use Case #1: Select Movie and Showtime

- **Actors**: Registered User

Flow of Events:

1. User logs into their account.
2. User selects a movie from the available list.
3. User chooses a showtime and seats from the real-time seating chart.
4. **Include: Assign Seat** – User selects a seat from available options.
5. **Include: Payment** – User proceeds to checkout and enters payment information.
6. The system processes the payment and generates a QR code for the ticket.
7. The ticket is emailed to the user, and if the user chose SMS delivery, a text message is sent as well.

Preconditions:

- User must be logged in; seats must be available.

Postconditions:

- The ticket is confirmed, and a QR code is generated.

Error Handling:

- If payment fails, the user will be prompted to retry.

3.3.3 Use Case #2: Order Concessions

- **Actors**: Registered User

Flow of Events:

1. User logs into their account or selects the option to continue as a guest.
2. User navigates to the concessions menu.
3. User selects food and drinks from the available items.
4. User chooses a pickup time if not purchasing a movie ticket.
5. **Include: Payment** – User proceeds to payment and completes the order.
6. The system processes the payment and sends a confirmation email with the order and pickup details.
7. The order is sent to the concessions staff for preparation.

Preconditions:

- Concession items must be available.

Postconditions:

- The order is confirmed, and an email or mobile notification is sent with the details and pickup time.

3.3.4 Use Case #3: Cancel or Modify Booking

- **Actors**: Registered User

Flow of Events:

1. User logs into their account.
2. User navigates to their booking history.
3. User selects the booking they wish to cancel or modify.
4. If modifying, the user selects a new showtime or seat (if available).
5. If canceling, the user confirms the cancellation.
6. **Include: Refund** – The system processes a refund (if applicable) and updates the user's booking history.
7. **Extend: Cancel Notification** – The system sends a cancellation notification to the user via email or SMS.

Preconditions:

- User must be logged in, and cancellation/modification must occur within the allowable time frame.

Postconditions:

- The booking is canceled or modified, and the user is notified via email or SMS.

Error Handling:

- If the cancellation or modification is not allowed, the system shows an error message.

3.3.5 Use Case #4: Login/Logout

- **Actors**: Registered User, Admin

Flow of Events:

1. User/Admin navigates to the login page.
2. User/Admin enters their credentials (username and password).
3. The system verifies the credentials and logs the user/admin into the system.
4. User/Admin logs out by clicking the "Logout" button.

Preconditions:

- User/Admin must have valid credentials.

Postconditions:

- User/Admin is logged into or out of the system.

Error Handling:

- If the credentials are invalid, the system displays an error message.

3.3.6 Use Case #5: View Booking

- **Actors**: Registered User

Flow of Events:

1. User logs into their account.
2. User navigates to the "Booking History" section.
3. User selects a booking to view.
4. The system displays the details of the booking, including movie name, showtime, seat, and purchase date.

Preconditions:

- User must be logged in.

Postconditions:

- The user views their past or current bookings.

3.3.7 Use Case #6: Manage Users

- **Actors**: Admin

Flow of Events:

1. Admin logs into the system.
2. Admin navigates to the "Manage Users" section.
3. Admin can view, add, delete, or modify user accounts.
4. The system updates the user information as requested by the admin.

Preconditions:

- Admin must be logged in with appropriate permissions.

Postconditions:

- User information is updated successfully.

Error Handling:

- If the update fails, the system displays an error message.

3.4 Classes / Objects

3.4.1 Class / Object #1: USER

3.4.1.1 Attributes:
All this information will be encrypted to protect the user's identity.

1. UserID: A unique identifier for identifying each individual user in the database
2. Name: First and Last name of the user
3. Email: User's email address
4. Phone Number: Optional for the user to receive notifications about new events/sales the movie theater is putting on. Can also be optional for 2 factor authentication.
5. Password: Password for the user's account
6. Account Status: Whether the account is registered as active, inactive, or suspended if they have violated the user agreement (for example charging back on a ticket after they received it).
7. Purchase History: Past purchases on tickets and or concession orders

3.4.1.2 Functions
1. createAccount(): Allows users to create an account. Will ask for things like Name, Email, Password, and Phone Number(optional).
2. login(): Allows users to login to our website with the account they have previously created. Will ask for the previous Email and Password to verify ownership of the account.
3. updateProfile(): Allows the user to change the Email, Password, Name, and Phone Number on their account.
4. viewPurchaseHistory(): Allows the user to view the history of their purchases for both tickets and concession items.
5. placeBookingOrder(): Allows the user to purchase a ticket for a movie.
6. placeConcessionOrder(): Allows the user to place an order for items at the concession stands in the theater.
7. cancelBooking(): Allows the user to cancel a booking at a respective theater as long as it's before the movie has started to play and they have not already used the ticket to get into the theater.
8. cancelConcessionOrder(): Allows the user to cancel an order on items at the concession stand as long as they have not already picked up the items.

3.4.2 Class / Object #2: MOVIE

3.4.2.1 Attributes:

1. MovieID: Specific ID for a respective movie, will also include an ending unique ID that will correspond with the time. For example 2369-1230 where 2369 is the ID for the specific movie and 1230 is the ID for the time.
2. Title: Title of the movie.
3. Genre: Genre of the movie.
4. Description: A short summary of what the movie will be about.
5. Rating: Rating given by the audience on a scale of 0-5 with the ability to have numbers up to 2 decimal places of accuracy.
6. Movie Duration: How long the movie is in minutes.
7. Showing Times: Available showing times for the movie.

3.4.2.2 Functions

1. getMovieTitle(): Gets the title of the movie.
2. getMovieGenre(): Gets the genre of the respective movie.
3. getMovieDescription(): Gets the description of the movie.
4. getMovieRating(): Gets the audience rating on the movie.
5. getMovieDuration(): Gets the duration of the movie in minutes.
6. getMovieShowTimes(): Gets the available showtimes for the movie.
7. getSeatsAvailability(): Checks the seating availability for the respective showtime.

3.4.3 Class / Object #3 BOOKING

3.4.3.1 Attributes:

1. BookingID: Unique identifier for each booking.
2. UserID: The user's unique identifier that is making the booking.
3. MovieID: The movie's unique identifier that will be booked.
4. Seats: Seats that have been selected to be booked by the user.
5. TotalAmount: Total price for the booking.
6. PaymentStatus: True or False value that will show if the payment has gone through or has been declined.

3.4.3.2 Functions

1. createBooking(): Creates a new booking for the user.
2. cancelBooking(): Allows the user to cancel the booking they have already made and it will also refund the user in full.
3. updateBooking(): Allows the user to change seats or showtimes for their booking.

3.4.4 Class / Object #4 PAYMENT

3.4.4.1 Attributes:

1. PaymentID: Unique identifier for the user's specific purchase.
2. BookingID: Unique identifier for the booking being made.
3. PaymentMethod: The method of payment (ex. credit card, paypal, mastercard, visa).
4. Amount: Total amount paid.
5. PaymentStatus: True or False value on if the payment has gone through or been declined.

3.4.4.2 Functions

1. processPayment(): Processes the user's payment through their respective payment method. Returns true if the payment has gone through and false if it has been declined.
2. refundPayment(): Returns the amount paid by the user back to their respective method of payment.

3.5 Non-Functional Requirements

3.5.1 Performance

Payments must be processed in under 2 seconds 98% of the time. Pages should fully load in at least 2-3 seconds on average. Top websites aim for 2-3 second average page load times. The available seats should refresh every time a user enters the screen to pick their seats. The system should be able to handle a minimum of 5,000 concurrent users during high traffic events like weekends, holidays, or new film releases.

3.5.2 Reliability

System should be up at least 99.95% of the time which accounts for 4 hours of downtime a year. This downtime should be scheduled during off times like 2am-5am when no one is really ordering a movie ticket. All payments and price amounts should be 100% accurate with no leniency in accuracy due to the possibility of being sued or losing reputation. The system should have systems in place in case a server crashes. In this case a backup server should be able to come online immediately and should not lose any of the data from the previous server.

3.5.3 Availability

The system must be available at least 99.95% of the time (allowing for the 4 hours a year downtime for maintenance). The system should be accessible by all platforms, for example mobile (IOS and Android), over the web (supporting the top 10 popular web browsers), in theater kiosks.

3.5.4 Security

All user data (Name, Email, Phone Number, Payment info, Password) must be properly encrypted using AES-546 encryption with TLS 1.3 for data transit following best practices implemented by companies like Paypal and Stripe. Must follow PCI DSS (Payment Card Industry Data Security Standard) Compliance level 1 for secure handling of card transactions which would follow payment gateways like Paypal and Stripe. Must comply with data privacy regulations like California Consumer Privacy Act (CCPA) for all U.S users. No need to comply with General Data Protection Regulation (GDPR) from the EU since all of our theaters are in the U.S. Must use multi-factor authentication for staff to avoid unauthorized access. This feature is used by other industry leaders to help avoid breaches. Lastly we should use fraud detection to

flag suspicious transactions like purchasing an entire theater worth of tickets to avoid things like ticket scalping.

3.5.5 Maintainability

Everything in the system should be built with modularity in mind. This is to reduce the cost of developing new features and to reduce the likelihood of a system-wide crash. Should also be automatically monitored for things like errors and alerts and these issues should be reported to staff immediately to again avoid a possible system crash or fault. Should allow for hotfixes to be deployed to the whole system in a matter of at most 1 hour. Everything in the system should be properly documented so that new developers can easily figure out what things do what and can properly maintain the system without much headache.

3.5.6 Portability

Should allow for mobile access on web browsers like Safari for IOS and Chrome/Firefox for Android. Should also be able to run on the in theater movie kiosks. Should be an app available on both IOS and Android that will be on the respective app store that will allow users on mobile to more easily purchase tickets and concession items. This app will be developed in parallel with the website, just with different code obviously.

Temp Cutoff for Assignment #1 Due 9-26-2024

3.6 Inverse Requirements

State any *useful* inverse requirements.

3.7 Design Constraints

Specify design constrains imposed by other standards, company policies, hardware limitation, etc. that will impact this software project.

3.8 Logical Database Requirements

Will a database be used? If so, what logical requirements exist for data formats, storage capabilities, data retention, data integrity, etc.

3.9 Other Requirements

Catchall section for any additional requirements.

3.10 Design

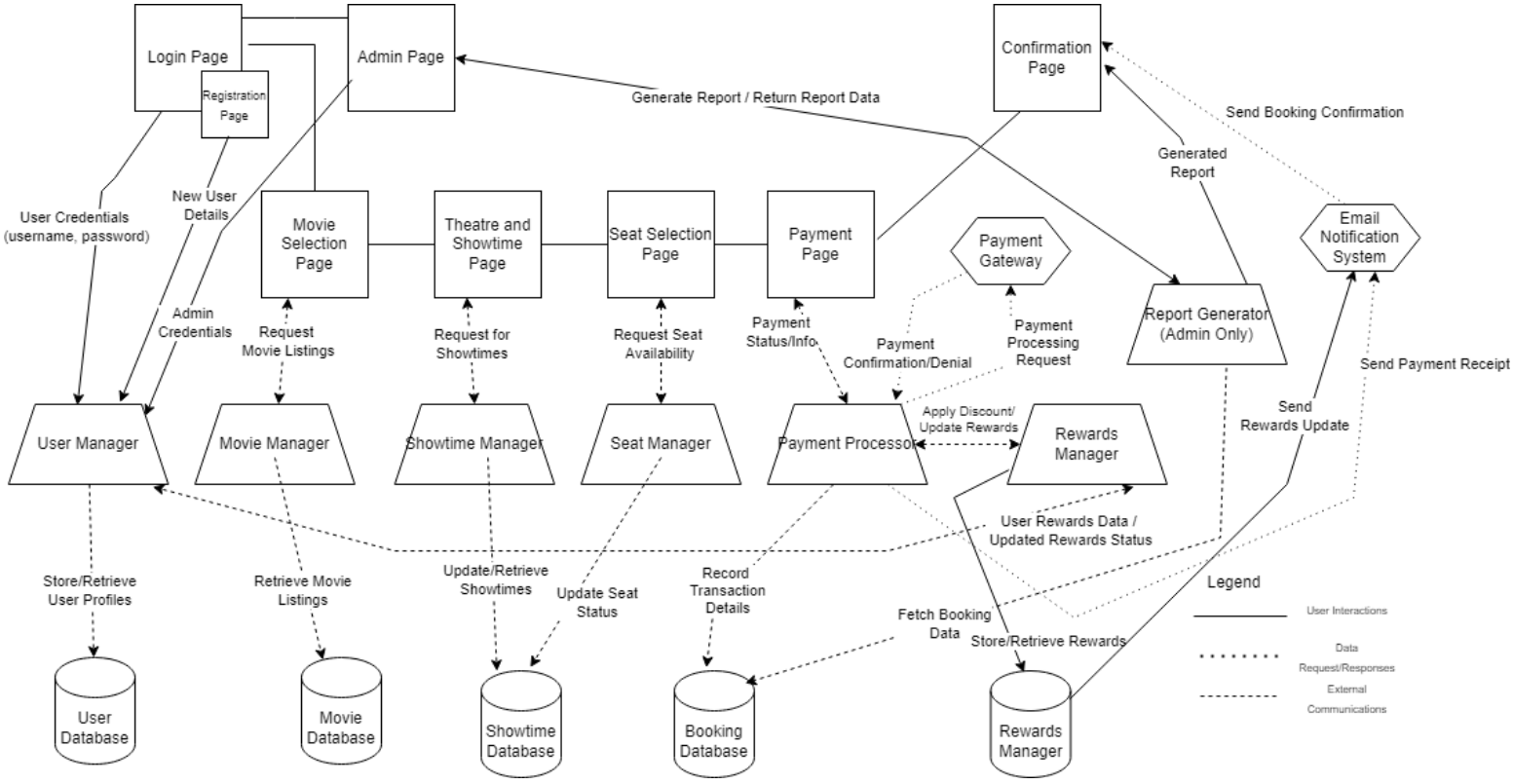
System Description

The Movie Theatre Ticketing System (MTTS) is designed to optimize the management and sale of movie tickets, offer an intuitive interface for customers to acquire tickets, and incorporate

administrative features for theatre personnel to oversee showtimes, access transaction history, and produce sales reports. The system seeks to provide a seamless experience for customers and theatre personnel, incorporating features like real-time seat availability, adaptable payment methods, and comprehensive show details.

Software Architecture Overview

SWA Diagram



Software Architecture Diagram Description

The software architecture for the **Movie Theatre Ticketing System (MTTS)** is structured into distinct layers that manage different functionalities, interactions, and data flow. The architecture can be divided into the following main components:

1. User Interface Layer

The User Interface Layer consists of various pages that provide an interactive experience for users and administrators. These pages capture user input, display system information, and initiate requests to the backend for data and operations. Pages include:

- **Login Page**: Handles user authentication, allowing existing users to log in and new users to navigate to the Registration Page.
- **Registration Page**: Manages new user registration by capturing details such as username, password, and contact information.

- **Movie Selection Page**: Displays available movies for users to choose from and requests movie listings from the backend.
- **Theatre and Showtime Page**: Shows detailed information about showtimes and theatre locations for selected movies.
- **Seat Selection Page**: Allows users to select available seats for a chosen showtime and sends seat selection data to the backend.
- **Payment Page**: Collects payment details, applies rewards or discounts, and initiates payment processing.
- **Confirmation Page**: Displays a summary of the completed booking and sends confirmation details to the user.

The Admin Page serves as a specialized interface for administrative users, providing access to generate reports, update showtimes, and manage the theatre's backend data.

2. Backend Application Layer

The Backend Application Layer is responsible for processing the logic, managing data transactions, and coordinating operations across different modules. This layer includes various **managers** that handle specific parts of the system:

- **User Manager**: The User Manager communicates the User Database for various operations:
    - **Login and Registration**: The User Manager sends a request to the User Database to verify login credentials or store new user details. If the credentials are valid, the database returns the user's profile, including rewards status.
    - **Profile Updates**: For any user profile changes (e.g., updating contact info or password), the User Manager updates the corresponding record in the User Database and retrieves the updated information for display.
- **Movie Manager**: The Movie Manager retrieves movie data from the Movie Database:
    - **Movie Listings**: When the Movie Selection Page requests movie details, the Movie Manager queries the Movie Database to fetch the latest movie titles, descriptions, genres, and ratings. The movie data is then sent back to the frontend for display.
    - **Movie Updates**: If new movies are added or existing ones are modified (by an admin, for example), the Movie Manager updates the Movie Database accordingly and ensures that all movie listings are current.
- **Showtime Manager**: The Showtime Manager interacts with the Showtime Database to manage showtime data:

    - **Retrieve Showtimes**: For a given movie, the Showtime Manager requests available showtimes and theatre locations from the Showtime Database and provides the results to the Theatre and Showtime Page.

    - **Update Showtimes**: In case of changes (like added showtimes or changes in availability), the Showtime Manager sends updated showtime data to the Showtime Database.

- **Seat Manager**: The Seat Manager is responsible for maintaining seat availability data in the Showtime Database:

    - **Seat Availability**: The Seat Manager queries the Showtime Database to check seat availability for a given showtime and updates the UI accordingly.

    - **Seat Reservation**: When a user reserves a seat, the Seat Manager updates the seat status in the Showtime Database, marking the seat as taken.

- **Payment Processor**: The Payment Processor interacts with both the Booking Database and the Rewards Manager during a transaction:

    - **Transaction Recording**: After a successful payment, the Payment Processor stores the transaction details (e.g., ticket information, payment method) in the Booking Database.

    - **Rewards Application**: If the user is eligible for rewards or discounts, the Payment Processor checks with the Rewards Manager to retrieve the user's rewards status and apply any applicable points or discounts before storing the final transaction data in the Booking Database.

- **Rewards Manager**: The Rewards Manager maintains and updates rewards data in the Rewards Database:

    - **Rewards Retrieval**: The Rewards Manager queries the Rewards Database to retrieve the user's current rewards balance and eligibility for discounts during login or checkout.

    - **Rewards Update**: After a successful transaction, the Rewards Manager updates the user's rewards points in the Rewards Database, adding any newly earned points based on the transaction value.

- **Report Generator (Admin Only)**: The Report Generator fetches data from the Booking Database:

    - **Report Generation**: When an admin requests a report, the Report Generator queries the Booking Database for relevant transaction and booking data. It then compiles this information into a report that is returned to the admin.

The **Email Notification System** is integrated into this layer as a notification handler, responsible for sending automated emails triggered by various backend managers, such as the Payment Processor and Rewards Manager.

3. Database Layer

The Database Layer stores all persistent data and ensures the integrity of the system's operations. Each database interacts with the backend managers to perform data-related operations. The following databases are integral to the system:

- **User Database**: Stores user profiles, credentials, purchase history, and rewards status. The User Manager performs Create, Read, Update, Delete operations on this database during login, registration, profile updates, and rewards retrieval.
- **Movie Database**: Contains movie listings, genres, and ratings. The Movie Manager retrieves this data to display available movies, and admins can update or add new movies via the Movie Manager.
- **Showtime Database**: Manages showtime schedules and seat availability. The Showtime Manager retrieves and updates showtime data, while the Seat Manager retrieves seat availability and updates the status of reserved or released seats.
- **Booking Database**: Keeps track of completed bookings, seat assignments, and transaction details. The Payment Processor stores each confirmed transaction, and the Report Generator retrieves booking data for report creation.
- **Rewards Database**: Stores user-specific rewards data, including points earned and discounts available. The Rewards Manager handles the retrieval and updating of this data, allowing users to accumulate and apply rewards during checkout.

4. External Systems Layer

The External Systems Layer integrates the MTTS with third-party services to enable additional functionality:

- **Payment Gateway**: An external service that validates and processes user payments. The Payment Processor communicates with the gateway to send payment requests and receive confirmations.
- **Email Notification System**: An external component responsible for sending transactional emails, such as booking confirmations, payment receipts, and rewards updates. The Payment Processor, Rewards Manager, and Confirmation Page send triggers to the Email Notification System to notify users of completed transactions, rewards status changes, and booking confirmations.

Each external system is represented as a separate component and uses secure communication channels to ensure data integrity and privacy.

5. Data Flow and Communication

The data flow in the system is facilitated using a combination of user interactions, data requests/responses, and external communications:

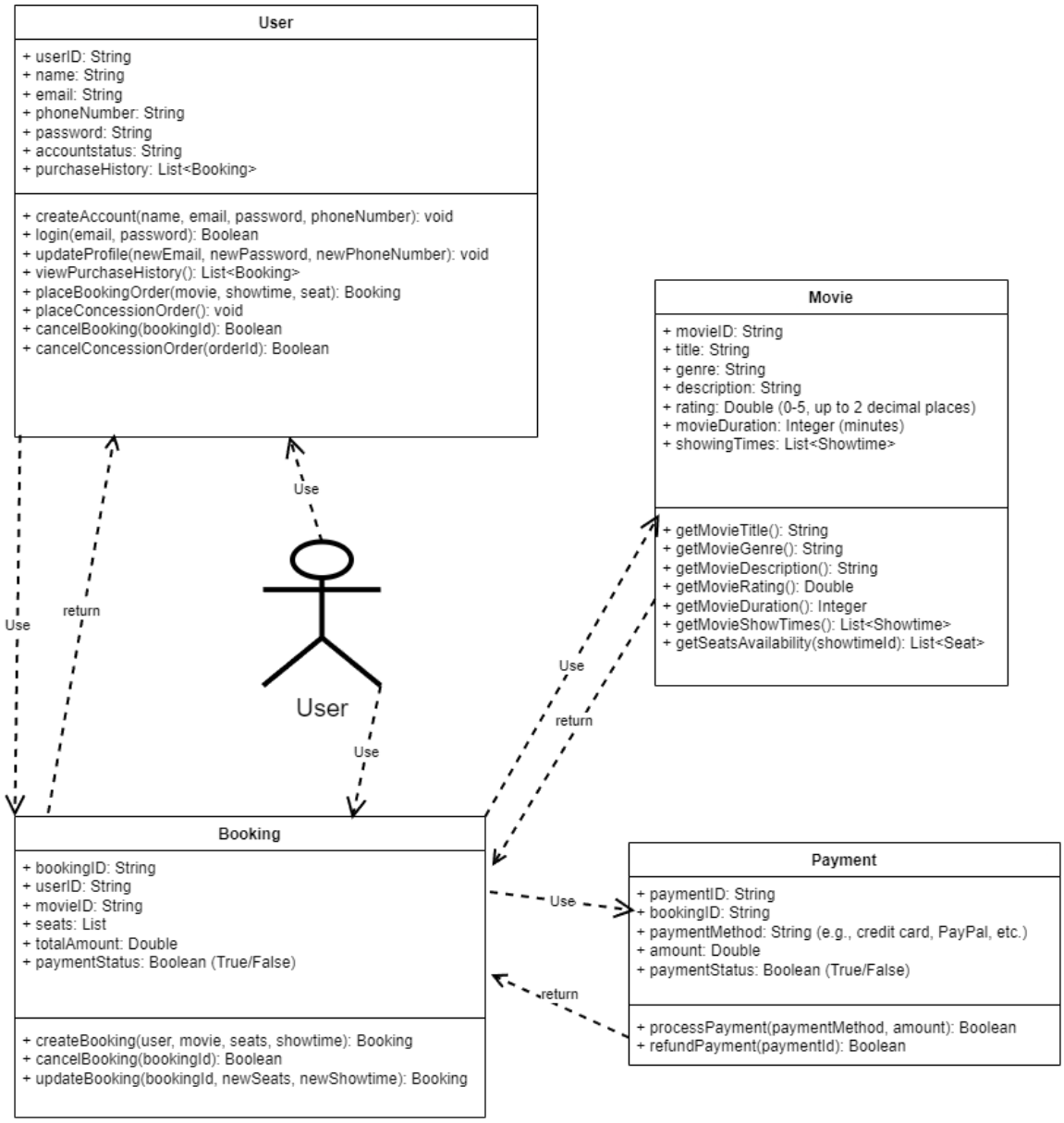
1. **User Interactions (Solid Lines)**: These represent direct interactions between the user and the UI pages as they navigate through the system.
2. **Data Requests/Responses (Dashed Lines)**: Backend managers communicate with databases and the UI components using these lines to handle data operations, including rewards management and seat availability checks.
3. **External Communications (Dotted Lines)**: Represent interactions with external systems, such as the Payment Gateway and Email Notification System, for secure payment processing and email notifications.

6. Architectural Considerations

- **Modularity**: Each component operates independently, allowing for easy updates or replacements.
- **Scalability**: The architecture can accommodate additional features or modifications without disrupting existing functionalities.
- **Security**: Sensitive data like user credentials and payment information is handled through secure communications with the User Manager and Payment Gateway.
- **Maintainability**: The clear separation of responsibilities between managers and UI components simplifies debugging and future extensions.

This architecture provides a robust foundation for the Movie Theatre Ticketing System, ensuring a seamless experience for both users and administrators while supporting advanced features like rewards and email notifications.

UML Class Diagram



Class Descriptions

- Class User: Represents a customer that uses the system to book movie tickets and manage their account. It lets the user manage their profile, interact with booking and rewards, and place concession orders.

- Attributes:

- userID: String - A unique identifier for each user.
- name: String - The user's full name (first and last name).
- email: String - The user's email address, used for account login and notifications.

- phoneNumber: String (Optional) - User's phone number for notifications and optional two-factor authentication.
- password: String - Password for the user’s account.
- accountStatus: String - Account status, which could be "active," "inactive," or "suspended."
- purchaseHistory: List<Booking> - List of bookings made by the user, including past ticket purchases and concessions.

- Operations:

createAccount(name: String, email: String, password: String, phoneNumber: String?): void

- Creates a new account with an optional phone number.

login(email: String, password: String): Boolean

- Authenticates the user by checking the email and password.

updateProfile(newEmail: String, newPassword: String, newPhoneNumber: String?): void

- Allows the user to update their account details such as email, password, and phone number.

viewPurchaseHistory(): List<Booking>

- Retrieves the list of the user's past bookings and purchases.

placeBookingOrder(movie: Movie, showtime: Showtime, seat: Seat): Booking

- Allows the user to place an order for a movie ticket with a selected showtime and seat.

placeConcessionOrder(orderItems: List<ConcessionItem>): void

- Allows the user to order items from the concession stand.

cancelBooking(bookingID: String): Boolean

- Cancels a booking if the movie has not started and the ticket has not been used.

cancelConcessionOrder(orderID: String): Boolean

- Cancels a concession order if the items have not been picked up.

- Class Movie: Represents the details of each individual movie. It contains information like the movie's title, genre, description, and ratings. It is linked to the showtimes and seat availability for the movie.

- Attributes:

- movieID: String - A unique identifier for each movie.
- title: String - The title of the movie.
- genre: String - The genre of the movie (e.g., Action, Comedy, Drama).
- description: String - A brief summary of the movie.
- rating: Double - Audience rating on a scale from 0 to 5 with two decimal places.
- movieDuration: Integer - The duration of the movie in minutes.
- showingTimes: List<Showtime> - A list of available showtimes for the movie.

- Operations:

getMovieTitle(): String

- Returns the title of the movie.

getMovieGenre(): String

- Returns the genre of the movie.

getMovieDescription(): String

- Returns the description of the movie.

getMovieRating(): Double

- Returns the audience rating for the movie.

getMovieDuration(): Integer

- Returns the duration of the movie in minutes.

getMovieShowTimes(): List<Showtime>

- Returns the list of available showtimes for the movie.

getSeatsAvailability(showtimeID: String): List<Seat>

- Checks and returns the availability of seats for the given showtime.

- Class Booking: Manages the ticket booking process for users. It records the user's selected movie, showtime, seats, and payment status. It also allows users to cancel or update their bookings.

- Attributes:

- bookingID: String - A unique identifier for each booking.
- userID: String - The unique identifier of the user who made the booking.
- movieID: String - The unique identifier of the movie being booked.
- seats: List<Seat> - The list of seats that the user has selected for the booking.
- totalAmount: Double - The total amount to be paid for the booking.
- paymentStatus: Boolean - Indicates whether the payment has been confirmed (True) or declined (False).

- Operations:

createBooking(user: User, movie: Movie, seats: List<Seat>, showtime: Showtime): Booking

- Creates a new booking for the user with the selected movie, seats, and showtime.

cancelBooking(bookingID: String): Boolean

- Cancels an existing booking and refunds the user if applicable.

updateBooking(bookingID: String, newSeats: List<Seat>, newShowtime: Showtime): Booking

- Allows the user to update the seats or showtime for an existing booking.

- Class Payment: Handles the payment process for bookings. It processes the user's payment and stores payment information, including the payment method and status on if the payment has been processed or not.

- Attributes:

- paymentID: String - A unique identifier for the payment.

- bookingID: String - The unique identifier for the booking associated with the payment.
- paymentMethod: String - The method used for payment (e.g., Credit Card, PayPal).
- amount: Double - The total amount paid for the booking.
- paymentStatus: Boolean - Indicates whether the payment has been processed successfully (True) or failed (False).

- Operations:

processPayment(paymentMethod: String, amount: Double): Boolean

Processes the payment and returns True if successful or False if declined.

refundPayment(paymentID: String): Boolean

Issues a refund to the user based on the provided paymentID and returns True if the refund is successful.

Development Plan and Timeline

Project Phases

The development of the Movie Theatre Ticketing System will be divided into several key phases.

Phases Overview

Phase 1: Requirements Gathering and Analysis

Duration: 1 week

Tasks:

Gather detailed requirements from stakeholders (users, administrators).

Define functional and non-functional requirements.

Document use cases for user interactions and administrative tasks.

Phase 2: System Design

Duration: 2 weeks

Tasks:

Design the overall architecture and create class diagrams.

Create wireframes for the user interface (UI) and administrative pages.

Define the database schema and data flow between components.

Review design with stakeholders for feedback and approval.

Phase 3: Development

Duration: 4 weeks

Tasks:

Week 1: Develop the User Interface Layer (Login, Registration, Movie Selection pages).

Week 2: Develop the Seat Selection, Payment, and Confirmation pages.

Week 3: Implement the Backend Application Layer (User Manager, Movie Manager, Showtime Manager).

Week 4: Implement Payment Processor and Rewards Manager; integrate external systems (Payment Gateway, Email Notification System).

Phase 4: Testing

Duration: 2 weeks

Tasks:

Unit testing for individual components and classes.

Integration testing to ensure different layers work together seamlessly.

User Acceptance Testing (UAT) with stakeholders to validate functionality.

Phase 5: Deployment

Duration: 1 week

Tasks:

Deploy the system to a staging environment for final review.

Address any issues found during the staging review.

Launch the system to production.

Phase 6: Documentation and Training

Duration: 1 week

Tasks:

Prepare user documentation and admin manuals.

Conduct training sessions for users and administrators.

Gather feedback for future improvements.

Team Member Responsibilities

1. **Julio Nevarez**
    a. **Backend Application Layer**:
        i. Responsible for developing the User Manager, Movie Manager, and Showtime Manager components.
        ii. Implements the logic for user authentication, movie listings, and showtime management.
        iii. Works on the integration with the Rewards Manager.
2. **Matthew Kloth**
    a. **Database Layer:**
        i. Oversee the design and management of the User Database, Movie Database, Showtime Database, and Booking Database.
        ii. Ensures data consistency, integrity, and proper communication between the backend and the databases.
        iii. Implements the data models and storage for user profiles, rewards, and booking transactions.
3. **Doan Quoc Tien Nguyen**
    a. **User Interface Layer:**
        i. Focuses on developing the Login Page, Registration Page, Movie Selection Page, Theatre and Showtime Page, and Seat Selection Page.
        ii. Ensures the UI is user-friendly, interactive, and properly connected to backend services.

iii. Collaborate with Julio to ensure proper data flow between the UI and the backend.

4. **Jingyi Chen**
    a. **External Systems Layer:**
        i. Responsible for integrating the Payment Gateway and Email Notification System into the backend.
        ii. Implements the Payment Processor and ensures secure transaction handling and notification triggers.
        iii. Works on automating confirmation emails and
        iv. other user notifications.

Shared Responsibilities

Report Generator: Both Julio Nevarez and Matthew Kloth will collaborate on developing the admin-specific report generation functionality, ensuring it pulls the correct data from the backend and databases.

Testing and Debugging: All team members will contribute to the testing phase, ensuring that their respective layers communicate properly with one another and handle edge cases effectively

Data Management Strategy

(1) **Database Type Choice:**

(i) **Primary Choice:** SQL Database

- **a. Structured Data:** Given the structured nature of MTTS's data (e.g., user profiles, bookings, showtimes), an SQL relational database is optimal.
- **b. ACID Compliance:** SQL databases offer ACID properties (Atomicity, Consistency, Isolation, Durability), which are crucial for the MTTS, ensuring each booking or payment transaction is processed accurately without risk of data corruption.
- **c. Consistency Across Modules:** SQL enables consistent data management across multiple modules, essential for handling tasks such as seat selection, which requires real-time updates and availability checks.

(ii) **Alternative Consideration:** NoSQL Database

- **a. Scalability and Flexibility:** NoSQL databases, like document or key-value stores, could offer advantages in handling unstructured data or scaling with fluctuating loads.
- **b. Drawbacks:** However, NoSQL lacks the strong consistency and relational structure SQL offers, which is critical for transactional systems like MTTS. This makes SQL the preferred choice, given MTTS's current functional requirements.

(2) Data Segmentation and Organization:

(i) Logical Separation:

Data is segmented into specific databases based on functionality:

- **a. User Database:** Stores user credentials and profile data, accessible only by the User Manager.
- **b. Movie Database:** Holds movie listings, genres, and details, managed by the Movie Manager.

c. **Showtime Database:** Manages showtimes for each movie, organized by date and time for easy retrieval by the Showtime Manager.
d. **Booking Database:** Tracks seat selection, booking transactions, and records payment statuses, critical for Seat Manager and Payment Processor.
e. **Rewards Database:** Stores rewards data, tracking user points and discounts.

(ii) Normalization and Indexing:

a. Each database is normalized to minimize redundancy, improving storage and efficiency
b. Indexing is applied to frequently queried fields, like user ID, showtime ID, and booking ID, which enhances query performance for high-traffic operations.

(3) **Security, Backup and Recovery**:

(i) **Data Security**:

a. Sensitive data fields (e.g., payment details) are encrypted to protect user information.
b. Role-based access controls restrict data access to only authorized users, with different access levels for admins, regular users, and system managers.

(ii) **Backup and Recovery**:

a. Regular database backups ensure data is not lost in case of a system failure. Backups are automated and stored securely offsite.
b. The recovery protocol includes both manual and automated processes to restore data up to the last backup, minimizing downtime.

(4) **Concurrency and Transaction Management**:

(i) **Transaction Management**:

a. SQL's transaction handling ensures that seat selection and booking remain atomic, where each transaction completes fully or rolls back in case of errors.
b. **Two-Phase Locking**: Implemented to prevent issues like double-booking during concurrent seat reservations. The growing phase locks all necessary records, and the shrinking phase releases locks only after the transaction completes.

(ii) **Deadlock Detection**:

a. A deadlock detection mechanism monitors for potential deadlocks, such as two users trying to book the same seat, and releases locks if necessary to resolve the conflict.

(5) **Tradeoff Discussion**:

(i) **SQL vs. NoSQL**:

a. SQL databases are chosen due to their structured format and reliability for transactional data, which is essential for MTTS's booking and payment processes.
b. NoSQL could be advantageous if MTTS expands to handle high volumes of non-transactional data or needs more flexible schema

management. For now, SQL meets the system's needs with minimal complexity and maximum consistency.

(ii) **Multiple vs. Single Database**:

- **a.** Using multiple specialized databases enhances modularity and performance by allowing each module to access only its relevant data.
- **b.** A single database could reduce management overhead but could introduce performance bottlenecks, especially in high-traffic scenarios.

4. Test Plan

Test Plan documentation:
https://github.com/1400Pika/MTTS/blob/main/MTTS%20Test%20Plan.pdf
Excel Test Case Samples:
https://github.com/1400Pika/MTTS/blob/main/CS250%20Test%20Cases.xlsx

5. Analysis Models

List all analysis models used in developing specific requirements previously given in this SRS. Each model should include an introduction and a narrative description. Furthermore, each model should be traceable the SRS's requirements.

5.1 Sequence Diagrams

5.3 Data Flow Diagrams (DFD)

5.2 State-Transition Diagrams (STD)

6. 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