Movie Ticketing System

Software Requirements Specification

Version <3.0>

2/15/2025

Group #10
Aadi Bery, Dariel Gutierrez, Kyan Santiago-Calling

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

Revision History

Date	Description	Author	Comments
2/20/25	Version 1		<Requirements Specification>
3/4/25	Version 2		<Design Specification>
3/20	Version 3		<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

1. Introduction

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

1.1 Purpose

The purpose of this SRS document is to provide all necessary information needed by a software engineer to adequately design and implement the **Movie Ticketing System**, which is an online system for users to find and purchase movie tickets at theaters throughout San Diego County for upcoming showings. Our specified audience for our application is the residents of San Diego County and this document is intended to be read by software engineers.

1.2 Scope

The scope pertains to the features and implementation of the Movie Ticketing System, which is intended to provide a user-friendly interface for San Diego County residents to easily browse and purchase movie theater tickets with a variety of different seats, showtimes, movies, and prices. Additionally, the interface will allow users to create an account where they can gain loyalty points for each purchase for being a member. Alternatively, they will also have the option to access the interface as a guest, in the case that the customer is not a frequent user of the system. The software will not, however, contain data from movie theatres and the corresponding tickets from movie theatres outside San Diego County.

1.3 Definitions, Acronyms, and Abbreviations

This subsection should provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the SRS. This information may be provided by reference to one or more appendixes in the SRS or by reference to other documents.

Term	*Definition*
SRS	*Software Requirements Specification*
UI/UX	*User Interface/ User Experience*
API	*Application Programming Interface*
DBMS	*Database Management System*
MTBF	*Mean Time Between Failure*

1.4 References

AMC Movie Theatre Website
IEEE Guide to Software Requirements Specification (IEEE 830-1998)

1.5 Overview

The remaining sections of this document provide general descriptions, characteristics, and the functions of the projects/users, the specific requirements, which include functional and non-functional requirements. It will also include analysis models to assist in the complete development and implementation of this product.

2. General Description

The movie ticketing system is designed for cinemas and users to facilitate online booking and payment transactions. Users can browse movies, choose seats, make payments online, as well choose add-ons like refreshments and food items to improve their viewing experience. Our database will contain a variety of different movie theatres throughout San Diego County with a variety of ticketing options and pricing. Additionally, a loyalty program will be offered for those users who choose to create an account.

2.1 Product Perspective

The system connects theaters and customers, enabling real-time ticket reservations. Unlike traditional box office ticketing, it provides seat selection, digital ticketing, refund options, and a customer rewards program. This movie ticketing system is specifically designed and tailored towards San Diego County residents.

2.2 Product Functions

The Movie Ticketing System provides users with a seamless platform to browse movies, select showtimes, purchase tickets, and manage their bookings. The system ensures secure transactions, reliable performance, and an intuitive user experience.

2.3 User Characteristics

General Users: Browse and purchase movie tickets and receive booking details
Registered Users: Earn rewards and save booking details.
Guest Users: Purchase tickets without signing up.
Admin Users: Oversee the system operations and provide customer support

2.4 General Constraints

Seat availability updates must be real-time.
The system must be available 24/7 except during maintenance.
The system must be desktop-friendly
High server uptime is required during peak hours
Secure transactions and GDPR compliance
The system should support multiple theatre chains

2.5 Assumptions and Dependencies

User is assumed to have a modern browser,
User is assumed to have stable wifi connection
User is assumed to be familiar with basic online booking processes and can navigate through the UI without guidance
User must utilize a valid payment method and the system must provide secure payment processes
Theatre must apply accurate showtimes and showdates.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

This system shall have a responsive web interface supporting desktop and mobile applications
The interface shall allow multiple language options

3.1.2 Hardware Interfaces

The system shall be compatible with a barcode scanner for ticket scanning

3.1.3 Software Interfaces

The system shall integrate with different payment gateways
The system shall connect to the cinema scheduling system

3.1.4 Communications Interfaces

The system shall have email notifications for booking confirmations

3.2 Functional Requirements

3.2.1 <Provide Ticket Details>

3.2.1.1 The system shall display the selected movie, its watchtime, and its location.
3.2.1.2 The system shall allow users to view pricing information for the given ticket
3.2.1.3 The system shall allow users to view up-to-date and available seating and layout information with an inventory check.
3.2.1.3 The system shall allow users to view ticket category (Base or Premium)
3.2.1.4 The system shall allow users to add up to 10 tickets for a given order
3.2.1.4 The system shall return a viewable ticket on the website

3.2.1.5 The system shall allow users to view unique ticket ID
3.2.1.6 The system shall allow users to update their configuration to handle errors

3.2.2 <Provide Search and Filter Bar>

3.2.2.1 The system shall allow users to search for theatre locations and specific movies
3.2.2.2 The system shall allow users to sort and filter search results
3.2.2.3 In this sort and filter panel, the system shall allow users to enter in zip-code for location detection

3.2.3 <Provide Shopping Cart Facility>

3.2.3.1 The system shall allow users to add up to ten tickets to the shopping cart.
3.2.3.2 The system shall display a confirmation message when successfully adding tickets to cart.
3.2.3.3 The system shall allow users to remove selected ticks from the cart.
3.2.3.4 The system shall allow users to “View Cart”, displaying movie name, which theatre, showtime, seat selection, quantity, and price.
3.2.3.5 The system shall continuously display and update the total cost of tickets in the cart.
3.2.3.6 The system shall allow users to “proceed to checkout” on the shopping cart.

3.2.4 <Provide Purchasing Interface>

3.2.4.1 The system shall redirect users to a secure purchasing interface
3.2.4.2 The system shall allow users to enter email address for purchase confirmation and invoice
    3.2.4.2.1 The system shall prefill this information for registered users
    3.2.4.2.2 The system shall not prefill and allow guest to enter preferred email address
3.2.4.3 The system shall contain a 10 minute timer upon redirecting for user to complete payment
3.2.4.4 The system shall allow users to use multiple payment options for purchase (Credit/Debit, Apple Pay, Paypal) through payment gateways and APIs
    3.2.4.4.1 For Credit/Debit purchase, the system shall permit Visa/Mastercard
3.2.4.5 The system shall allow users to input discount for Military, Student, or Senior
3.2.4.6 The System shall email confirmation and customer invoice after purchase processed and authorized (detailed in **3.2.6)**

3.2.5 <Provide and Maintain Customer Profile>

3.2.5.1 The system shall allow users to create an account with the ticketing system
3.2.5.2 The system shall display users account information, purchases made, and point occurred
3.2.5.3 The system shall allow users to change personal information
3.2.5.4 The system shall allow users to save a payment method for future purchases
3.2.5.5 The system shall maintain a user email database to send discount notifications

3.2.5 <Provide Guest Access Alternative>

3.2.5.1 The system shall permit website visitors to purchase tickets with no account
3.2.5.2 The system shall display a second option for guest checkout
3.2.5.3 The system shall allow users to

3.2.6 <Email confirmation and customer invoice>

3.2.6.1 The system shall send an email confirmation to the user's registered email upon payment.
3.2.6.2 The email shall contain every detail about the user's ticket, including location, movie, showtime, seat, price, ED, payment method, quantity, and contact information.
3.2.6.3 The system shall generate a digital invoice for the transaction and attach it to the email.

3.2.7 <Offer rewards and loyalty program for registered customers>

3.2.7.1 The system shall implement a reward system that awards loyalty points for every dollar spent by every customer with an account.
3.2.7.2 The system shall allow users to redeem loyalty points by allowing them to use them at checkout to discount future ticket prices.
3.2.7.3 The system shall set and notify users that their loyalty points expire 12 months after earning them.

3.2.8 <Offer Refund Policy for Purchases>

3.2.8.1 The system will allow for returns until the day before the movie

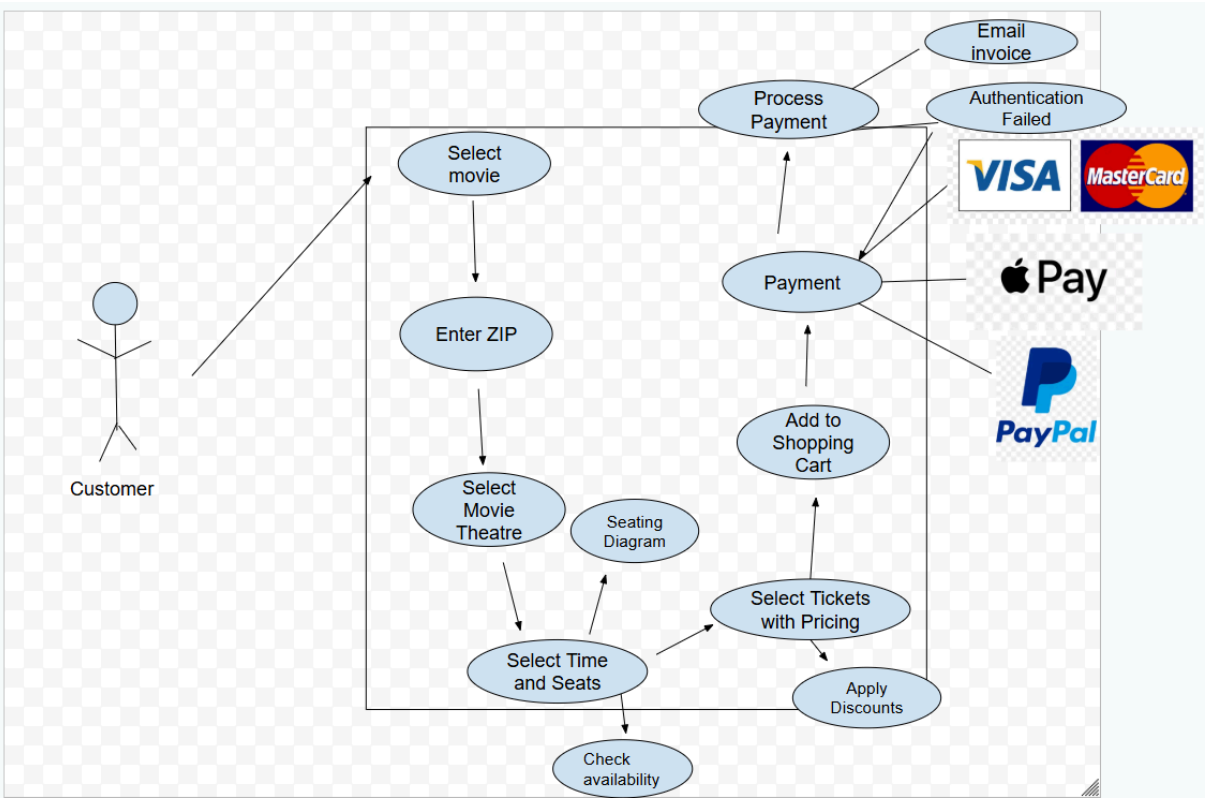
3.2.9 <Provide Customer Feedback Option>

3.2.9.1 The system shall allow users to reflect on their web experience after purchasing tickets
3.2.9.2 The system shall display a star ranking meter from 1-5 and an optional textbox for any concerns or comments

3.3 Use Cases

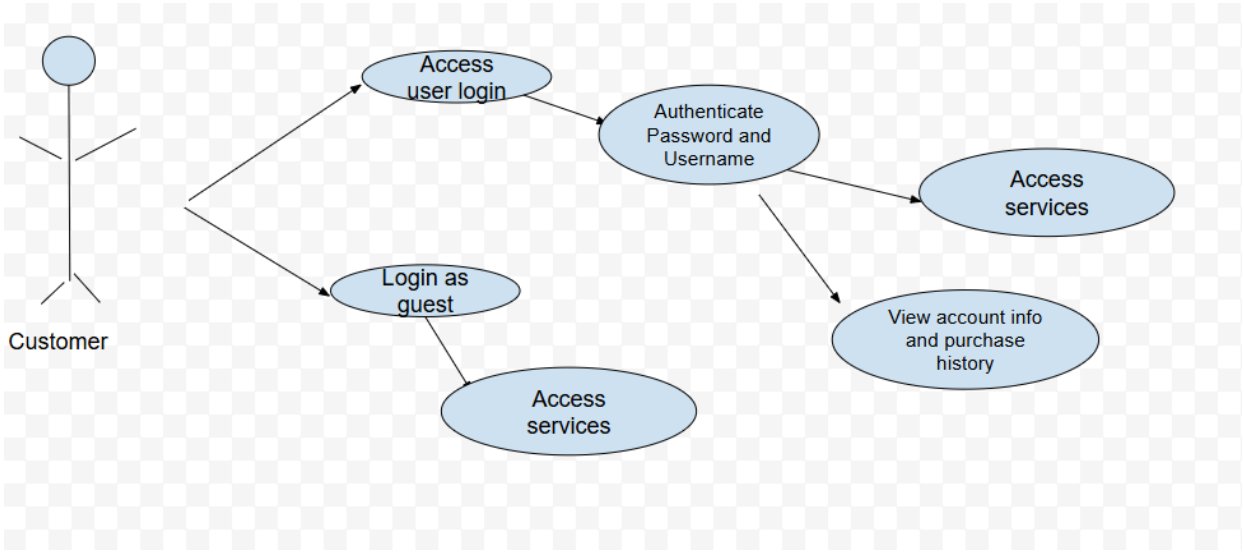
3.3.1 Use Case #1

Purchasing a Ticket
Actors: Customer, Payment gateway



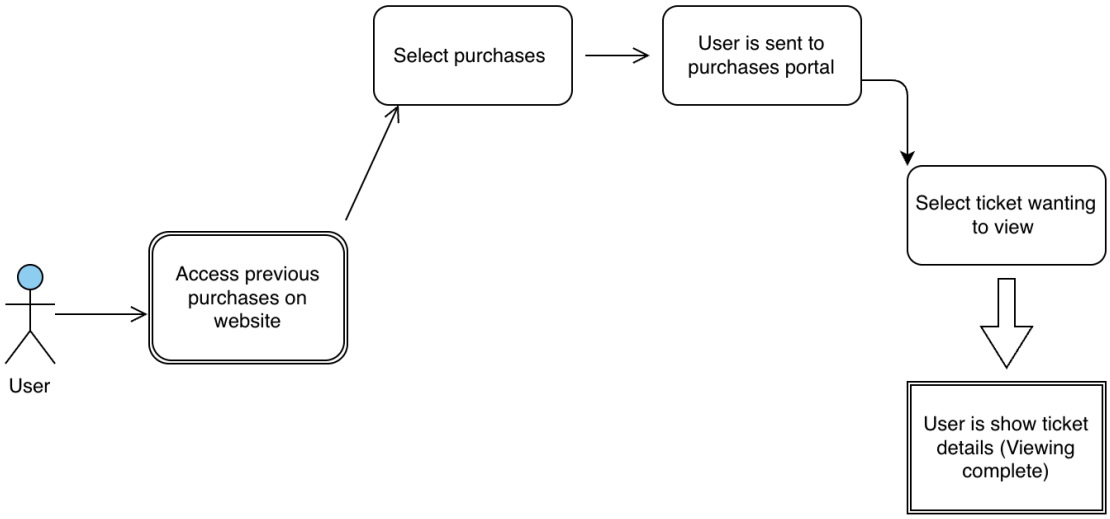
3.3.2 Use Case #2

Logging into System or Accessing As Guest



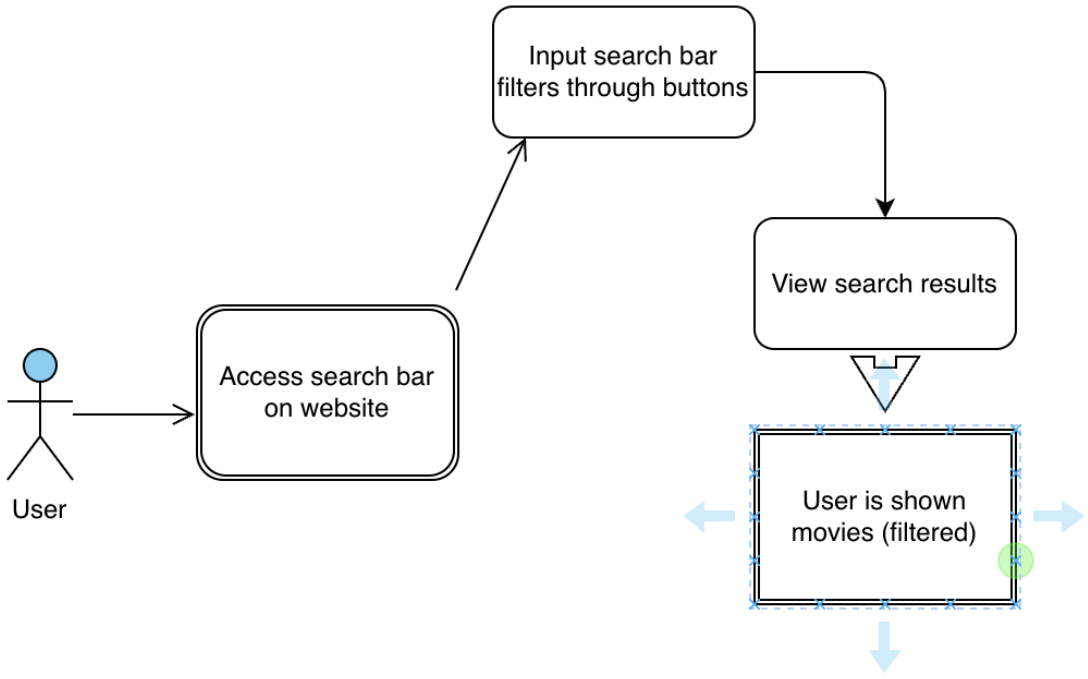
3.3.3 Use Case #3

Previewing ticket
Actors: User



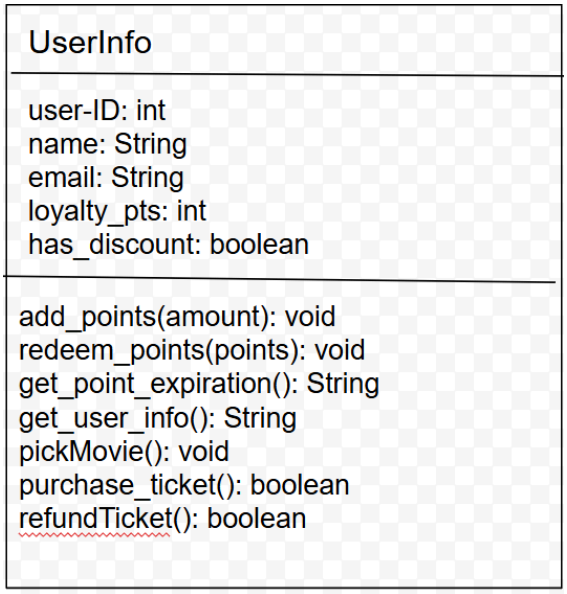
3.3.4 Use Case #4

Searching for Tickets or Theaters
Actors: User

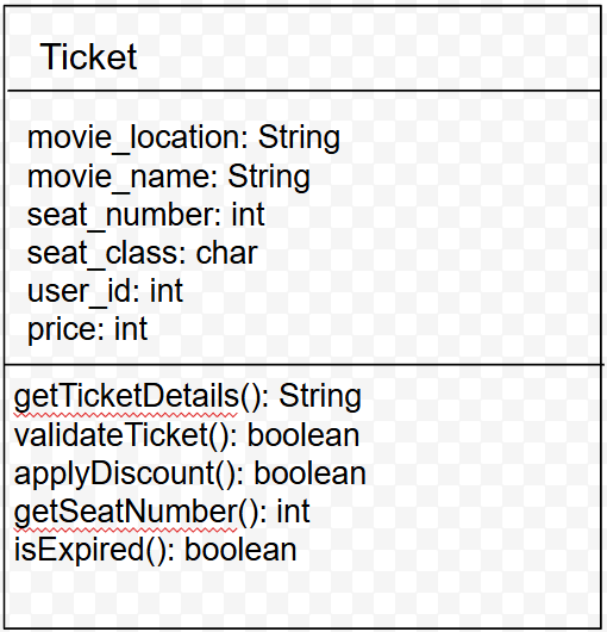


3.4 Classes / Objects

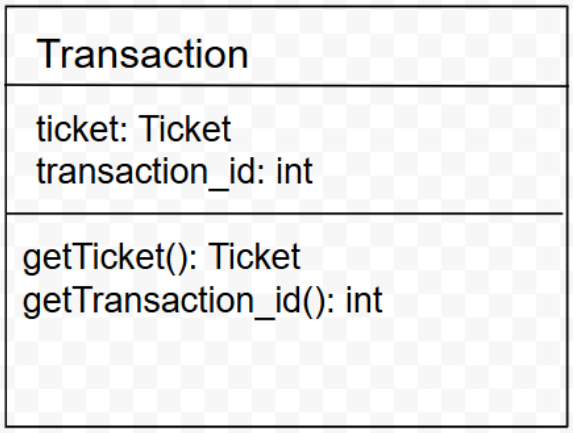
3.4.1 <UserInfo class>



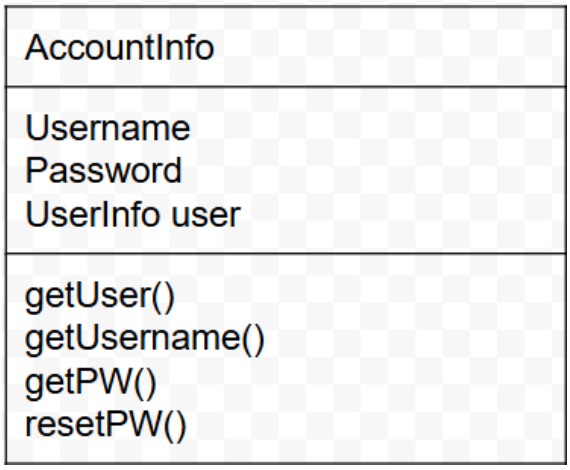
3.4.2 <Ticket Class>



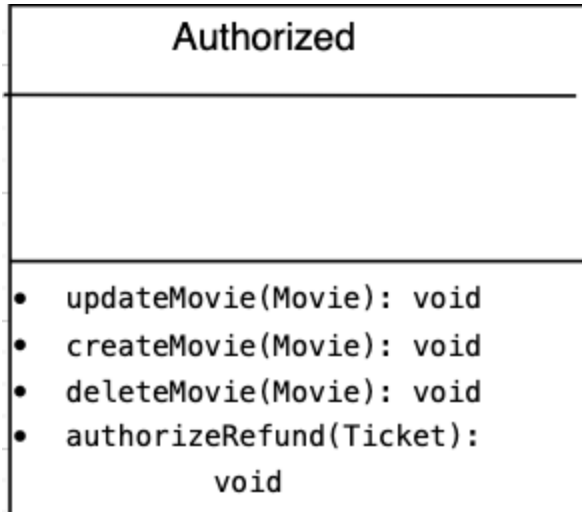
3.4.3 <Transaction Class>



3.4.4 <AccountInfo Class>



3.4.5 <Authorized Class>



3.5 Non-Functional Requirements

3.5.1 Performance

3.5.1.1 The system shall process 95% of transactions in under 2 seconds.
3.5.1.2 The system shall retrieve movie listings and showtimes in under 1 second.
3.5.1.3 The system shall handle 50,000 users at once and still maintain peak performance
3.5.1.4 The system shall redirect users to a secure payment portal in under 2 seconds
3.5.1.5 The system shall load the homepage within 1.5 seconds.
3.5.1.6 The system shall respond to all API calls in under 1 second for critical operations like ticket booking, user authentication, and seat selection

3.5.2 Reliability

3.5.2.1 The system shall have a MTBF of at least 30 days
3.5.2.2 The system shall have a 99% uptime with automated failure mechanisms.
3.5.2.3 Error recovery should take less than 5 minutes for non-critical issues
3.5.2.4 In the case of critical failure, the system shall contain a backup recovery plan that ensures data integrity and recovery within 15 minutes

3.5.3 Availability

3.5.3.1 The system shall be accessible 24/7 with a devoted 2 hours of downtime per month
3.5.3.2 The system's scheduled maintenance time shall be limited to 2AM-4AM and announced 24hrs in advance.

3.5.4 Security

3.5.4.1 User passwords must be hashed and crypted
3.5.4.2 User Payment and Personal Information must be non-visible in payment confirmation and invoice

3.5.5 Maintainability

3.5.5.1 The system shall allow for updates during downtime
3.5.5.2 The system shall be deployed at least once per quarter to ensure security and performance improvements

3.5.5.2 Logs of website errors should be maintained for 6 months

3.5.6 Portability

3.5.6.1 The system shall be compatible with the latest versions of Chrome, Firefox, Microsoft
3.5.6.2 Edge, Safari.
3.5.6.3 The system shall work on devices with screens from 4 inches to 30 inches.
3.5.6.4 The system shall function smoothly on windows and MacOS.

3.6 Inverse Requirements

3.6.1 The system shall not allow users to book tickets after the showtime has started.
3.6.2 The system shall not allow refund requests after the showtime unless the movie was canceled.

3.7 Design Constraints

The system shall comply with the standards for handling credit/debit card transactions securely.

The system shall follow the regulations for protecting users data protection.
The system shall follow OpenID for secure login processes.
Sensitive user data shall be encrypted.

3.8 Logical Database Requirements

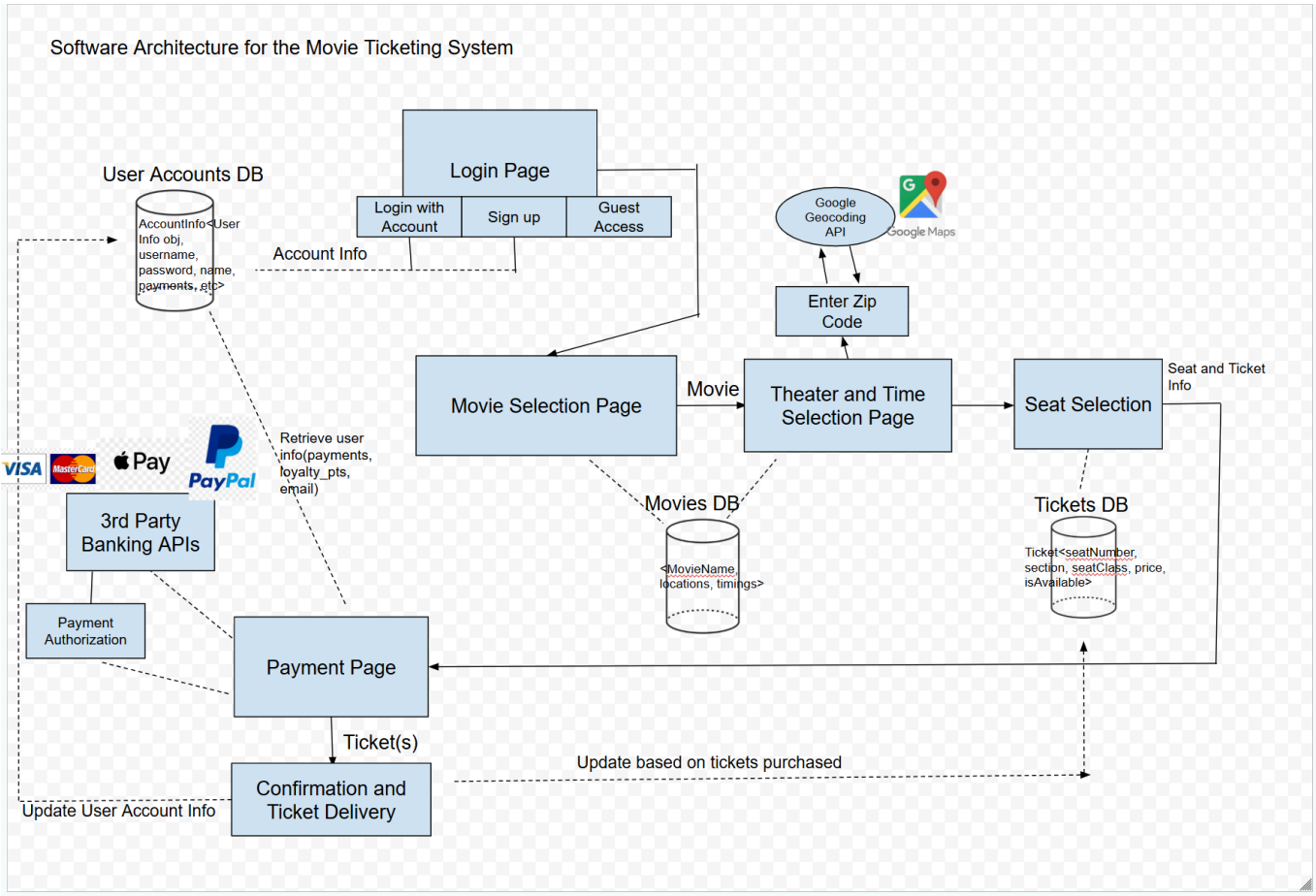
The system should use an SQL relational database with tables for **Users** (UserID, Name, Email, Password, Loyalty Points), **Movies** (MovieID, Title, Genre, Rating, Duration, Showtimes), **Theaters** (TheaterID, Location, Seating Layout), **Tickets** (TicketID, UserID, MovieID, TheaterID, SeatNumber, Price, Status), **Transactions** (TransactionID, UserID, TicketID, PaymentStatus, Date)

3.9 Other Requirements

The system shall integrate third-party APIs for payment processing, seat availability, and movie database updates while ensuring automated daily backups.

4. Software Design Specification

4.1 Architectural Diagram



4.1.1 Architectural Diagram Description

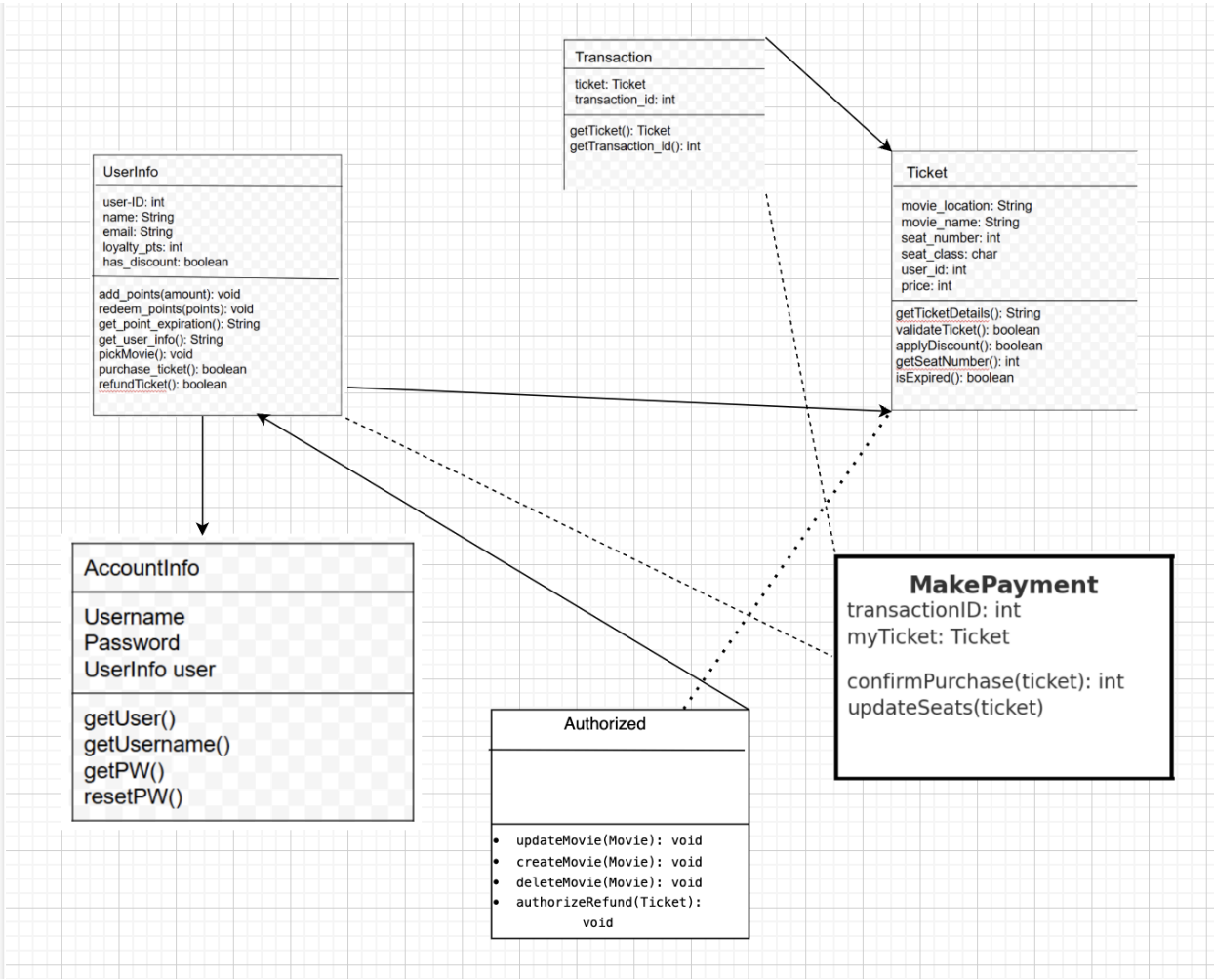
The software architecture diagram provides an overview of the system's structure, depicting the various components, their interactions, and data flow. It illustrates how the movie ticketing system is designed, incorporating key modules such as user authentication, movie selection, theater and seat booking, and payment processing. It includes multiple, interconnected components of the system.

These include:

- User Accounts DB, which stores the account information for each user in the form of a UserInfo object, username and password(encrypted), and as well as saved payments, among other info like purchase history.
- The login page, which allows for users to create an account, login with an existing account, or access services as a guest
- The movie selection page which fetches data from the Movies DB, containing info on all movie showings with their locations and showtimes.

- Theater and Time selection page which utilizes the Geocoding API to provide location-based results and enables users to select a theatre and showtime for their requested movie.
- Seat Selection page retrieves seating availability and information from the Tickets DB, enabling the user to select their desired seat based on seat class, section, price, and availability.
- Payment Page, which processes transactions for the desired tickets(s) through 3rd party Banking APIs, and it also retrieves user information from the User Accounts DB to prefill payment info and add discounts.
- Confirmation and Ticket Delivery, which is shown once there has been a successful transaction. The system generates a digital ticket and invoice based on the purchased ticket, which is sent to the email on file or provided by the guest. Additionally, the tickets and User Accounts databases are updated accordingly.

4.2 UML Class Diagram



4.2.1 UML CLASS DESCRIPTION

The UML class diagram for the Movie Ticketing System provides a structured representation of key system components, their attributes, and interactions. The UserInfo class stores customer details, including loyalty points and discount eligibility, while the AccountInfo class manages login credentials. Users can purchase movie tickets, represented by the Ticket class, which links to both transactions and payment processing. The MakePayment class facilitates secure payment confirmation and seat updates, ensuring a smooth booking experience. Additionally, the Transaction class tracks each ticket purchase or refund request. Administrative privileges are handled by the Authorized class, which allows authorized personnel to update, create, and delete movie listings, as well as approve ticket refunds. The relationships between these classes define how users interact with the system, from ticket selection to payment finalization and potential refund processing, ensuring a seamless and secure movie booking experience.

4.3 Individual Classes of System

4.2.1 UserInfo

The UserInfo class represents a user in the movie ticketing system. It stores information about registered users, including personal details, loyalty points, and discount eligibility. This class interacts with AccountInfo to manage login credentials and with Ticket to facilitate ticket purchases.

- userID: int – A unique identifier for each user.
- name: String – Stores the user's full name.
- email: String – Contact email for notifications and account-related communication.
- loyalty_pts: int – Tracks the number of loyalty points a user has accumulated.
- has_discount: boolean – Indicates whether the user qualifies for discounts.

4.2.1.2.1 add_points(amount: int): void

- Function: Adds loyalty points to the user’s account.
- The user must have an account in the system.
- The specified amount of points is added to the user’s loyalty balance.

4.2.1.2.2 redeem_points(points: int): void

- Function: Deducts loyalty points and applies a discount on ticket purchases.
- The user must have a sufficient number of loyalty points.
- Points are deducted, and the discount is applied.

4.2.1.2.3 purchase_ticket(): boolean

- Function: Confirms and completes a ticket purchase for the user.
- The selected ticket must be available and payment must be valid.

- The ticket is assigned to the user and marked as purchased.

4.2.1.2.4 refundTicket(): boolean

- Function: Initiates a refund request for a purchased ticket.
- The ticket must be eligible for a refund based on system policies.
- If approved, the ticket is refunded, and the payment is reversed.

4.2.2 AccountInfo

The AccountInfo class manages user authentication and security-related operations. It links directly to a UserInfo object and stores login credentials securely.

4.2.2.1 Attributes of AccountInfo Class

- username: String – Stores the unique username for the user's account.
- password: String – Stores the hashed password for authentication.
- user: UserInfo – Establishes a one-to-one relationship with a UserInfo object.

4.2.2.2 Functions Available in AccountInfo Class

4.2.2.2.1 getUser(): UserInfo

- Function: Retrieves the user associated with this account.
- The account must be registered in the system.
- Returns the UserInfo object linked to the account.

4.2.2.2.2 resetPW(): void

- Function: Allows the user to reset their password.
- The user must verify their identity (e.g., email verification).
- The password is updated securely.

4.2.3 Ticket

The Ticket class represents a movie ticket purchased by a user.

4.2.3.1 Attributes of Ticket Class

- movie_location: String – The theater where the movie is being played.
- movie_name: String – The title of the movie for which the ticket was purchased.
- seat_number: int – The specific seat assigned to the ticket.
- seat_class: char – The seating category (e.g., standard, premium).

- price: int – The price of the ticket.

4.2.3.2 Functions Available in Ticket Class

4.2.3.2.1 getTicketDetails(): String

- Function: Retrieves and displays the details of the ticket.
- The ticket must be associated with a valid purchase.
- The ticket details are returned to the user.

4.2.3.2.2 validateTicket(): boolean

- Function: Checks whether the ticket is valid for entry.
- The ticket must not be expired or used.
- Returns true if valid, otherwise false.

4.2.4 MakePayment

The MakePayment class is responsible for handling payments related to ticket purchases.

4.2.4.1 Attributes of MakePayment Class

- transactionID: int – A unique identifier for the payment transaction.
- myTicket: Ticket – The ticket associated with the payment.

4.2.4.2.1 confirmPurchase(ticket: Ticket): int

- Function: Confirms and processes a ticket payment.
- The user must provide valid payment details.
- The ticket is marked as paid, and the transaction ID is generated.

4.2.4.2.2 updateSeats(ticket: Ticket): void

- Function: Updates the seating chart after a ticket is purchased.
- The ticket purchase must be successful.
- The seat is marked as occupied.

4.2.5 Transaction

The Transaction class tracks ticket purchases and refund requests.

4.2.5.1 Attributes of Transaction Class

- ticket: Ticket – The ticket associated with this transaction.

- transaction_id: int – A unique ID for identifying the transaction.

4.2.5.2.1 getTicket(): Ticket

- Function: Retrieves the ticket associated with the transaction.
- The transaction must be valid.
- Returns the Ticket object linked to the transaction.

4.2.5.2.2 getTransaction_id(): int

- Function: Returns the unique transaction ID.
- The transaction must exist in the system.
- The transaction ID is returned.

4.2.6 Authorized

The Authorized class handles administrative functions such as movie management and refund approvals.

4.2.6.1 Functions Available in Authorized Class

4.2.6.1.1 updateMovie(Movie): void

- Function: Updates an existing movie's details.
- The movie must exist in the system.
- The movie details are modified successfully.

4.2.6.1.2 createMovie(Movie): void

- Function: Creates a new movie entry in the system.
- The administrator must have the required permissions.
- A new movie is added to the database.

4.2.6.1.3 deleteMovie(Movie): void

- Function: Deletes a movie from the system.
- The movie must exist in the system.
- The movie was removed successfully.

4.2.6.1.4 authorizeRefund(Ticket): void

- Function: Grants refund authorization for a ticket.
- The ticket must be eligible for a refund.
- The refund is processed, and the payment is reversed.

4.4. Other Information

The Movie Ticketing System is designed to facilitate seamless online movie ticket purchases, user account management, and administrative control over movie listings and transactions. The system includes essential functionalities such as user authentication, ticket selection, secure payment processing, refund management, and loyalty rewards. Additionally, an Authorized module is included to allow administrative users to create, update, and remove movie listings while managing refund approvals. The system is structured to provide an intuitive user experience, secure data handling, and efficient transaction processing.

Task	Description	Team Member(s) Responsible	Estimated Completion
System Design & Architecture	Design UML diagrams, database schema, and system flow.	Aadi, Dariel	Week 1 - Week 2
User Authentication	Develop login, registration, password reset features.	Dariel	Week 2 - Week 3
Ticket Booking Module	Implement movie selection, seat booking, and pricing.	Kyan	Week 3 - Week 5
Payment Processing	Integrate payment gateways and ensure secure transactions.	Aadi, Dariel	Week 4 - Week 6
Transaction & Refund System	Implement transaction tracking and refund approvals.	Kyan	Week 5 - Week 7

Loyalty & Rewards Program	Develop loyalty points tracking and redemption system.	Dariel	Week 6 - Week 8
Admin & Authorization Module	Implement movie management and refund authorization.	Aadi, Kyan	Week 7 - Week 9
Testing & Debugging	Perform unit, integration, and system testing.	Entire Team	Week 8 - Week 10
Deployment & Documentation	Deploy the system and finalize technical documentation.	Entire Team	Week 10 - Week 12

5. Test Plan

GitHub Link to Test Cases Document: here (view raw)

5.1 Unit Test Cases

5.1.1 Test Case #1 Payment_Processing

5.1.1.1 Test Description

Verify that when a user attempts payment, the system processes the payment and transitions the user to a "Transaction Successful" page.

5.1.1.2 Test Steps

1. Launch website
2. Navigate to the payment page
3. Enter valid payment information
4. Click “Submit” button
5. Confirm the transaction

5.1.1.3 Expected Result

Transaction valid, transitions user to a "Transaction completed" popup, and gives the user their information for product.

5.1.2 Test Case #2 Check_Credentials

5.1.2.1 Test Description
Verify that when a user enters email and password and selects login, credentials (email and password) are valid for said account.

5.1.2.2 Test Steps
1. Launch website for application and visit login page
2. Enter valid email address
3. Enter password on file and click sign in button

5.1.2.3 Expected Result
User credentials are correct, logs them into the homepage. Else, display an error message explaining with wrong information.

5.1.3 Test Case #3 Verify_Ticket_Status
5.1.3.1 Test Description
Verify that when user selects a movie and corresponding theater, seat and ticket is up to date

5.1.3.2 Test Steps
1. Launch website
2. Login to account or access as guest
3. Navigate to and select movie and theater location based off zip code
4. Load live seating and ticket data from Tickets Database

5.1.3.3 Expected Result
Correct and timely data is displayed for selected ticket.

5.1.4 Test Case #4 Verify_Movie_List
5.1.4.1 Test Description
Verify that when a user searches for a movie, the correct and available movies are displayed.

5.1.4.2 Test Steps
1. Launch website
2. Login, register a user account or access as guest
3. Navigate to Movie Search bar
4. Enter some relevant words or movie
5. Load most relevant result from movie list

5.1.4.3 Expected Result
Search results related to user's query should be listed, and sold out movies or previous movies should not be listed.

5.2 Functional / Integration Test Cases
5.2.1 Test Case #5 Verify_Ticket_Purchase

5.1.1.1 Test Description

Check if choosing a movie ticket based on location and selected movie correctly flows into payment/processing page, updates relevant databases and ticket status, and generates transaction

5.1.1.2 Test Steps

1. Launch Website
2. Login, register a user account, or access as guest
3. Select desired movie and theater location based on zip code
4. Select desired seat and get ticket information
5. Add to shopping cart and proceed to Payment Page
6. Input, verify, and system processes paymentTicket Purchase Flow
7. Ticket Status Updated to Booked: check if selecting a movie ticket correctly leads to payment processing, updates ticket status, and generates a transaction.
8. Transaction record is created a logged, user gets confirmation and ticket delivery

5.1.1.3 Expected Result

User is able to successfully select seats and movie details
Payment processing is successful and user receives confirmation and ticket.
Ticket and UserAccount Databases and status are updated
Transaction is created and stored

5.2 Functional / Integration Test Cases

5.2.2 Test Case #6 Verify Seat Reservation

5.2.2.1 Test Description

Confirm as soon as a ticket is added to cart, it is reserved on a timer, and cannot be booked by other users while it is reserved.

5.2.2.2 Test Steps

1. User A and User B launch website from different devices
2. User A and B both go to select the same seat
3. User A adds ticket to cart before User B
4. Ticket becomes unavailable for User B to add to cart
5. 15 minute timer begins for User A to checkout with ticket
6. If payment not completed within timer, ticket becomes available again.

5.2.2.3 Expected Result

User B should receive a popup/message that the ticket is no longer available.

5.2.3 Test Case #7 Verify Loyalty Points

5.2.3.1 Test Description

Verify that users can redeem loyalty points for discounts on ticket prices.

5.2.3.2 Test Steps

1. User logs into account and selects movie ticket

2. User continues to payment page
3. Clicks "redeem points" option at checkout page
4. System applies a discount based on available points
5. Completes discounted payment

5.2.3.3 Expected Result

Discount is directly applied before payment processes, with correct amount of user points and discount.

5.3 System Test Cases

5.3.1 Test Case #8 Admin Management

5.3.1.1 Test Description

Verify administrator access and management interface through ability to perform admin tasks successfully to maintain the website accordingly and fix bugs

5.3.1.2 Test Steps

1. User logs in with credentials and admin access
2. Updates a movie schedule or adds additional movies and seats
3. Confirms changes and validates that appear correctly for end-users

5.3.1.3 Expected Result

Administrator is successfully able to login with authorized privileges

Successfully able to create, update, delete movies or update schedule and seating

The updated changes are viewable by current and future users

5.3.2 Test Case #9 Full Ticket Booking Process

5.3.2.1 Test Description

Ensure full ticket booking pipeline from user authentication or guest access to ticket selection to payment and ticket confirmation/delivery

5.3.2.2 Test Steps

1. Launch website successfully
2. Login, create an account, or access as guest
3. Search for movies

4. Select seats and add tickets to shopping cart

5. Complete payment through 3rd party banking APIs

6. Receive ticket confirmation through email

5.3.2.3 Expected Result

The user successfully books a ticket, receives confirmation, and can use it for movie entry.

5.3.3 Test Case #10 System Performance

5.3.3.1 Test Description

Verify system stability under heavy traffic (many users at once), multiple users can simultaneously book tickets

5.3.3.2 Test Steps

1. Simulate 5000 users simultaneously booking tickets

2. Monitor response and loading times across all users

3. Ensure to note slowdowns and log crashes

4. Verify users, guests or registered users with accounts, can book tickets under a heavy load

5.3.3.3 Expected Result

System remains stable, response times all take less than 5 seconds, no crashes, no downtime.

6. 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.

6.1 Sequence Diagrams

6.2 Data Flow Diagrams (DFD)

6.3 State-Transition Diagrams (STD)

7. 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