Software Requirements Specification Template

Movie Ticketing System

Software Requirements Specification

4.00
5/26/2024

Group 1
Brandon Mahdavi
Justin Nam
Riley Mathews

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

Revision History

Date	Description	Author	Comments
05/26/24	1.00	Brandon M. Justin N. Riley M.	Initial Revision
06/03/24	2.00	Brandon M. Justin N. Riley M.	Added UML Diagram and Software Architectural Diagram.
06/10/24	3.00	Brandon M. Justin N. Riley M.	Adjusted UML Diagram and added Test Cases
06/17/24	4.00	Brandon M. Justin N. Riley M.	Added Database Structure and Adjusted SWA Diagram to fit the Database Structure

Document Approval

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

Signature	Printed Name	Title	Date
	Dr. Gus Hanna	Instructor, CS 250	05/27/24
Riley Mathews	Riley Mathews	Software Eng.	05/27/24
Brandon Mahdavi	Brandon Mahdavi	Software Eng.	05/27/24
Justin Nam	Justin Nam	Software Eng.	05/27/24

Table of Contents

1. Introduction

This Software Requirement Specification (SRS) document will be going over the requirements that are necessary for developing a Movie Theater Ticketing System.

1.1 Purpose

The purpose of this document is to outline the requirements for the development of a movie theater ticketing system. This system aims to provide an efficient and user-friendly platform for customers to browse movie listings, purchase tickets, and reserve seats for shows at a movie theater. It will help the software and project management teams understand the functional and non-functional requirements, various use cases, and the overall system architecture. The document aims to ensure the system meets the needs of theater administrators and customers, enhancing the efficiency of theater operations and improving the customer experience.

1.2 Scope

The software is designed for ROKU Theater to create a streamlined process for selling movie theater tickets. The developers of the system, (The Numba One Developers), are designing the next innovative Movie Ticketing System designed to allow users to purchase movie tickets both online and in person. The application will be web-based and accessible through a browser, with an initial deployment focused on theaters in the San Diego area. The system will also include administrative functions for managing showtimes and resolving customer issues.

The movie theater ticketing system will encompass both customer-facing and administrative functionalities. It will allow users to view available movies, showtimes, and seating options, as well as enable theater staff to manage movie schedules, ticket sales, and seating arrangements.

1.3 Definitions, Acronyms, and Abbreviations

SRS: (Software Requirement Specification)- is a detailed document that outlines what a software system should do, how it should behave, and its constraints and interfaces. It serves as a blueprint for software development, guiding the design, implementation, and testing processes.

API: (Application Programming Interface)-is a set of rules, protocols, and tools that allow different software applications to communicate and interact with each other.

UI: (User Interface)- user interacts with a computer system or software application, encompassing graphical elements, input controls, and navigational components.

DBMS: (Database Management System)- software that enables users to efficiently store, retrieve, and manage data in a structured format, providing functionalities for data manipulation, querying, and security.

CLI: (Command Line interface)- a text-based user interface that allows users to interact with a computer program by entering commands into a terminal or console, typically used for system administration, automation, and development tasks.

GUI: (Graphical user interface)- a visual interface that enables users to interact with electronic devices or software applications through graphical elements such as icons, windows, menus, and buttons, facilitating intuitive and user-friendly interactions.

1.4 References

References Included:

1. IEEE Computer Society. "IEEE Recommended Practice for Software Requirements Specifications." Approved 25 June 1998, reaffirmed 9 December 2009, Software Engineering Standards Committee, IEEE-SA Standards Board.
2. "Theater Ticketing System Qs.rtf" Canvas, San Diego State University.
3. "Theater Ticketing Requirements.rtf" Canvas, San Diego State University.
4. "Theater Ticketing Questions.rtf" Canvas, San Diego State University.

1.5 Overview

The movie theater ticketing system will consist of the following key components:

- User Interface (UI): A web-based/mobile application interface for customers to interact with the system.
- Database Management System (DBMS): Storage of movies, their showtimes, customer information, and transaction records.
- Backend Server: Process user requests, implement logic towards business standards, and third-party services (payment portals)
- Admin Control: An interface for the theater manager or owner to manage movie showings, seating arrangements, and to view sales reports for analytical data.

2. General Description

The theater ticketing system is a web-based consumer-friendly software that will allow consumers to purchase tickets online or in person at the theater using a digital kiosk. The web browser will be designed to incorporate a user-friendly interface, which will allow for easy access for both the consumer, and administrator mode.

2.1 Product Perspective

The ticketing system will be a standalone application designed to seamlessly integrate with the theater's existing software ecosystem. This integration encompasses various systems, including but not limited to, Inventory Management, Point of Sale (PoS), and Media Player software. By interfacing with these systems, the ticketing application ensures real-time synchronization of data such as inventory levels, sales transactions, and media scheduling. This interconnected approach enhances operational efficiency, providing a cohesive experience for theater administrators and staff. The integration allows for streamlined processes, such as automatic inventory updates when tickets are sold, synchronized media playback with showtimes, and consolidated financial reporting through the PoS system. Overall, this integrated ticketing solution aims to optimize theater operations and deliver a superior customer experience.

2.2 Product Functions

The product will allow customers to view showtimes for movies, select and purchase tickets based on seating. Seating arrangements will be live to reflect current availability. The product will support both online and in person sales. Discounts for supported groups will be available at checkout. There will also be administrative functions to manage showtimes, ticket prices, and customer feedback.

2.3 User Characteristics

System users will be movie-watchers, employees, and system administrators. Movie-watchers will use the system to purchase tickets and provide feedback. Employees will use the system to assist in purchases and customer service related activities. System administrators will handle management of listings, issues, and other related tasks.

2.4 General Constraints

- System must be able to handle 10 million concurrent users
- Login information must be secure
- Feedback system must be bot proof
- Discounts must be verified in person to be used online

2.5 Assumptions and Dependencies

The system will be able to integrate with existing management systems for data synchronization. The ticketing system will be deployed on a server. Users will have access to web browsers to use the software. In person software will be deployed on a touch screen kiosk. System will be able to handle concurrent users attempting to check out for the same ticket.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

Customer UI: Provides an intuitive interface for customers to browse movies, select seats, make payments, and receive tickets.
Administrator UI: Offers theater administrators tools to manage movies, showtimes, seating arrangements, and monitor sales and system performance.

3.1.2 Hardware Interfaces

Touch Screen:Facilitates user interaction with the ticketing system through an intuitive and responsive interface.

POS systems: Integrate with existing PoS systems to process ticket sales and generate receipts.
Printer: Facilitate the printing of physical tickets and receipts.

3.1.3 Software Interfaces

POS software: Synchronize sales data and financial transactions, providing comprehensive reporting and analytics.
Windows Operating System: Ensures robust security, compatibility, and ease of use, integrating seamlessly with existing IT infrastructure.
Ticketing Distribution Program: Manages the allocation and distribution of tickets across various sales channels, updating availability in real-time.
Firewall Software: Monitors and controls network traffic to protect against cyber threats and unauthorized access.

3.1.4 Communications Interfaces

LAN Card: Enables wired network connectivity for stable and high-speed communication between the ticketing system and other network resources.
WiFi Card: Provides wireless network connectivity, allowing flexible and convenient access to the ticketing system across the theater premises.
Bluetooth Card: Facilitates short-range wireless communication with compatible devices, supporting functionalities like ticket validation and mobile payments.

3.2 Functional Requirements

3.2.1 Customer Functionality

3.2.1.1 Customer UI

3.2.1.2 Inputs

User Registration and Authentication:
  Users can create accounts and log in securely.
  Password reset functionality is available.

Browse movies and showtimes:
  Display a list of currently playing movies and upcoming movies.
  Show available showtimes for each movie.

Seat Selection and Reservations:
  Allow users to select seats for their desired showtimes.
  Reserve selected seats for a limited time during the checkout process.

3.2.1.3 Processing
Ticket Purchase:
  Enable users to purchase tickets securely.
  Provide various payment options (credit or debit, apple/samsung/google pay, cryptocurrency)

3.2.1.4 Outputs
Ticket Confirmation:
  Issue electronic tickets with QR codes for validation at the theater.

3.2.1.5 Error Handling
View and Modify Bookings:
  Users can view past and upcoming bookings.
  Option to modify or cancel bookings within a specific time frame.

3.2.2 Administrator Functionality
3.2.2.1 Administrator UI

3.2.2.2 Input
Movie Management:
  Add, edit, or remove movie listings.
  Set showtimes and allocate screening rooms.
Seating Arrangement:
  Define seating layout for each screening room.
  Manage seat availability and reservations.

3.2.2.3 Output
Sales and Reporting:
  View ticket sale reports by movie, date, and time.
  Export sales data for accounting purposes.

3.3 Use Cases

3.3.1 Use Case #1

Use Case 1: Purchase Movie Tickets

Actor: Customer

Description: This use case describes the process of a customer purchasing movie tickets through the ticketing system.

Preconditions:

- Customers are logged into their account.
- Customer has selected a movie and showtime.

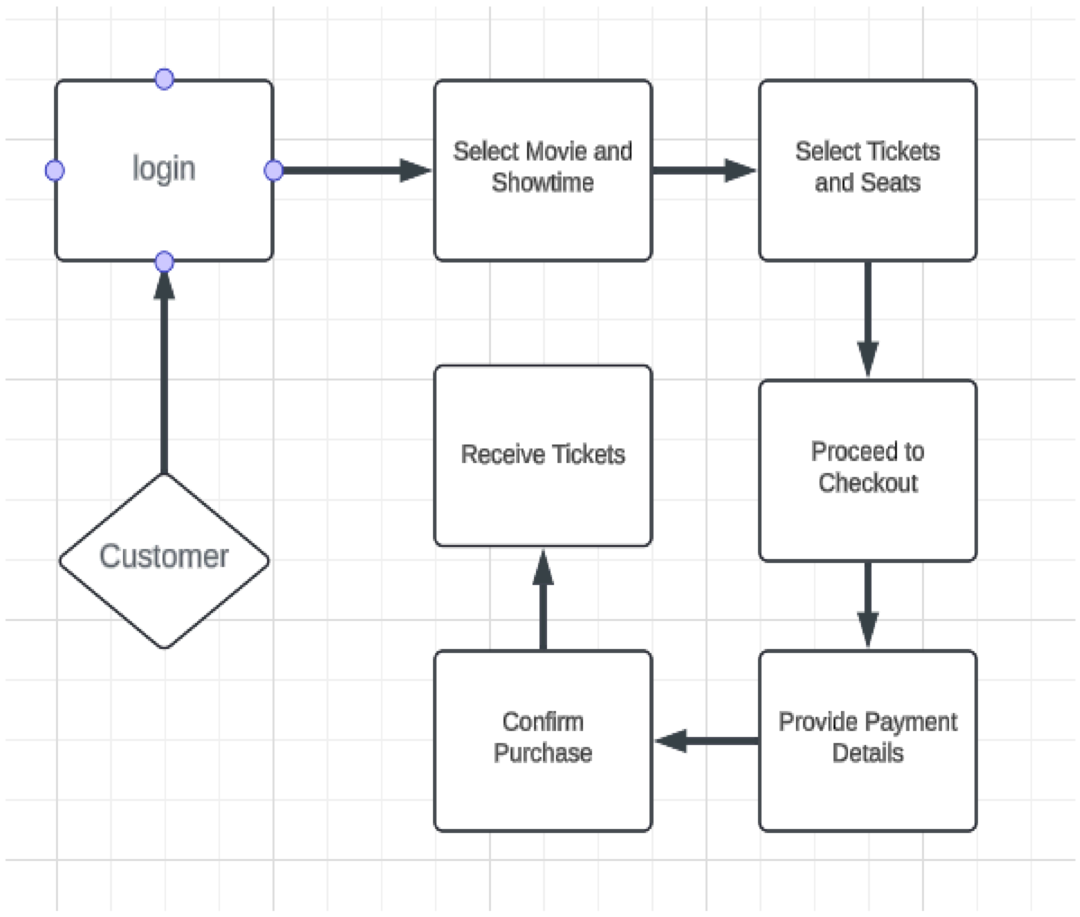
Main Flow:

1. Customer selects the desired movie and showtime from the list of available options.

2. Customers select the number of tickets and choose their preferred seats.
3. Customer proceeds to the checkout page.
4. Customer provides payment details and confirms the purchase.
5. System generates electronic tickets with QR codes.
6. Customers receive the tickets via email or within the app.

Postconditions:

- Tickets are reserved for the selected showtime.
- Payment is processed successfully.
- Customer receives confirmation of the purchase.



3.3.2 Use Case #2

Use Case #2: Manage Movie Listings

Actor: Administrator

Description: This use case describes the process of an administrator managing movie listings in the ticketing system.

Preconditions:

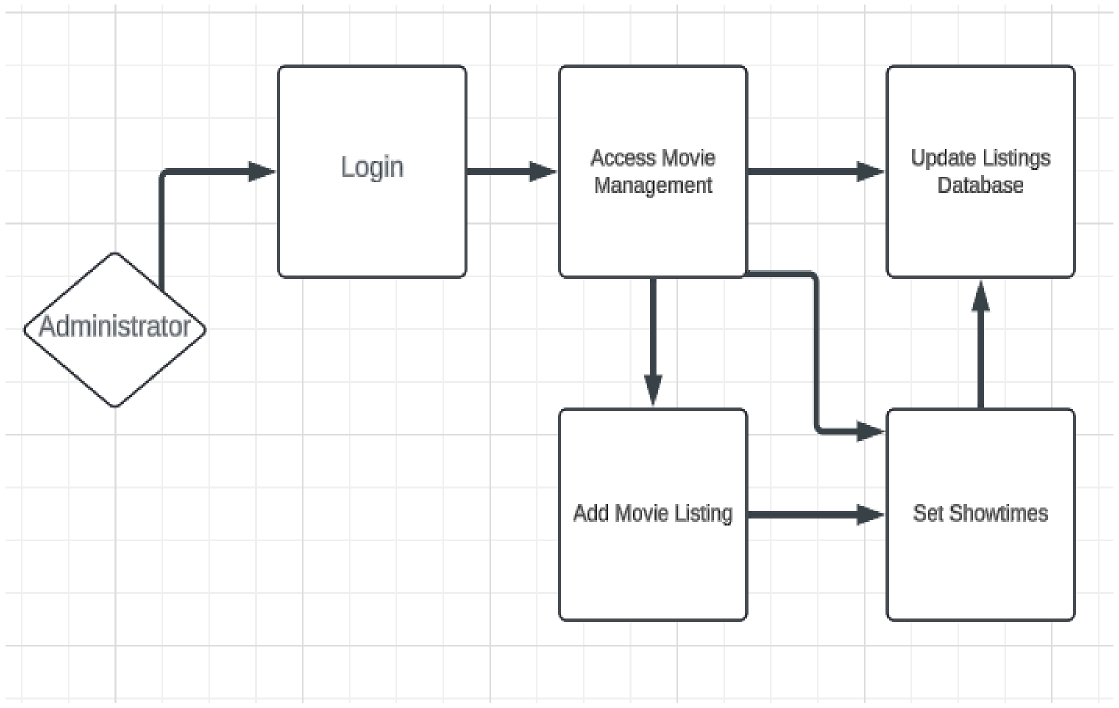
- Administrator is logged into the admin panel.

Main Flow:

1. Administrator accesses the movie management section of the admin panel.
2. Administrator adds a new movie listing by providing details such as title, description, genre, and poster image.
3. Administrator sets showtimes for the movie, specifying date, time, and screening room.
4. System updates the movie listings database with the new entry.

Postconditions:

- New movie listing is added to the system.
- Showtimes for the movie are scheduled and visible to customers.



3.3.3 Use Case #3

Use Case 3: View Sales Report

Actor: Administrator

Description: This use case describes the process of an administrator viewing sales reports in the ticketing system.

Preconditions:

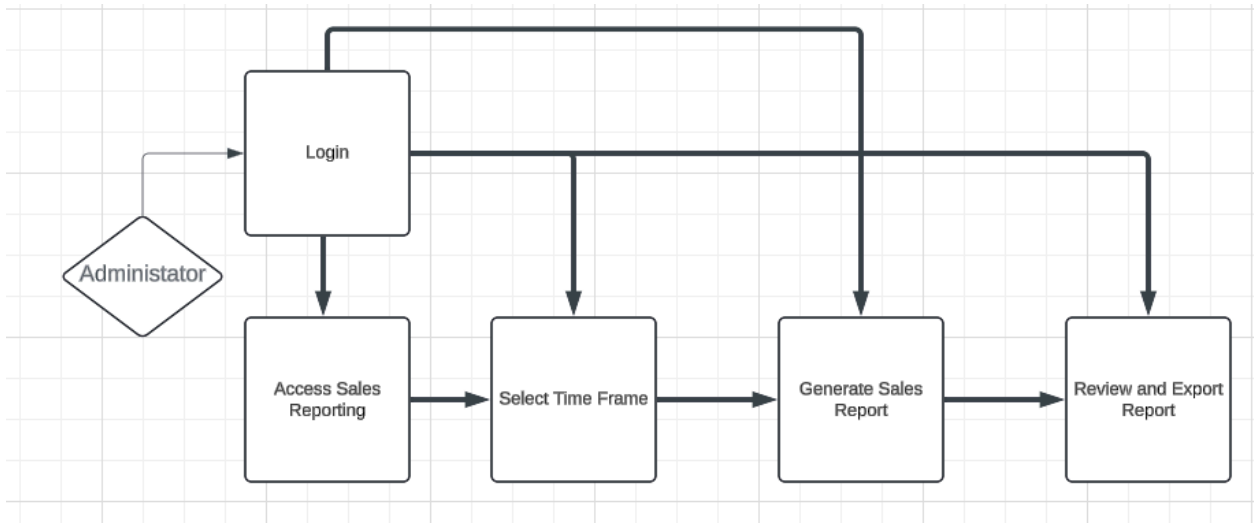
- Administrator is logged into the admin panel.

Main Flow:

1. Administrator navigates to the sales reporting section of the admin panel.
2. Administrator selects the desired time frame for the report (e.g., daily, weekly, monthly).
3. System generates a sales report, including total revenue, number of tickets sold, and breakdown by movie and showtime.
4. Administrator reviews the report and can export it for further analysis if needed.

Postconditions:

- Administrator gains insights into ticket sales performance within the selected time frame.
- Sales report data is available for analysis and decision-making purposes.



3.4 Classes / Objects

3.4.1 Get Available Tickets

3.4.1.1 Attributes
user, showtime, seats, booking date/time

3.4.1.2 Functions
Get which theaters are showing that movie and at what times.
confirmBooking(), cancelBooking()

3.4.2 Purchase Ticket

3.4.2.1 Attributes
user, showtime, seat, booking date/time, QR code

3.4.2.2 Functions
Purchase Selected ticket with movie, seat assignment, theater number, and time slot information.
generateQRCode(), validateTicket()

3.4.3 Get Sales

3.4.3.1 Attributes
username, password (encrypted)

3.4.3.2 Functions
Retrieve Sales Data from Database when called by the Administrator
login(), addMovie(), updateShowtime(), viewSalesReport()

3.5 Non-Functional Requirements

3.5.1 Performance

System should handle concurrent user traffic efficiently.
Response time for user interactions should be minimal.

3.5.2 Reliability

Error handling and feedback should be informative for users. The system will be backed up frequently so as to not lose any reservations or accounting information.

3.5.3 Availability

UI should be intuitive and accessible across different devices.

3.5.4 Security

Use encryption for sensitive data transmission.
Implement secure authentication and authorization mechanisms.

3.5.5 Maintainability

The systems will be maintained by IT techs on site that are not part of the regular movie theater staff.

3.5.6 Portability

The system will be able to be used and perform on anyone’s phone, computer, and kiosks
Install script that lets you install the software on any model of ticketing system

3.6 Inverse Requirements

*State any *useful* inverse requirements.*

3.7 Design Constraints

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

3.8 Logical Database Requirements

A database will be used to store what movies are showing at what times as well as how many seats are available in that theater.

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

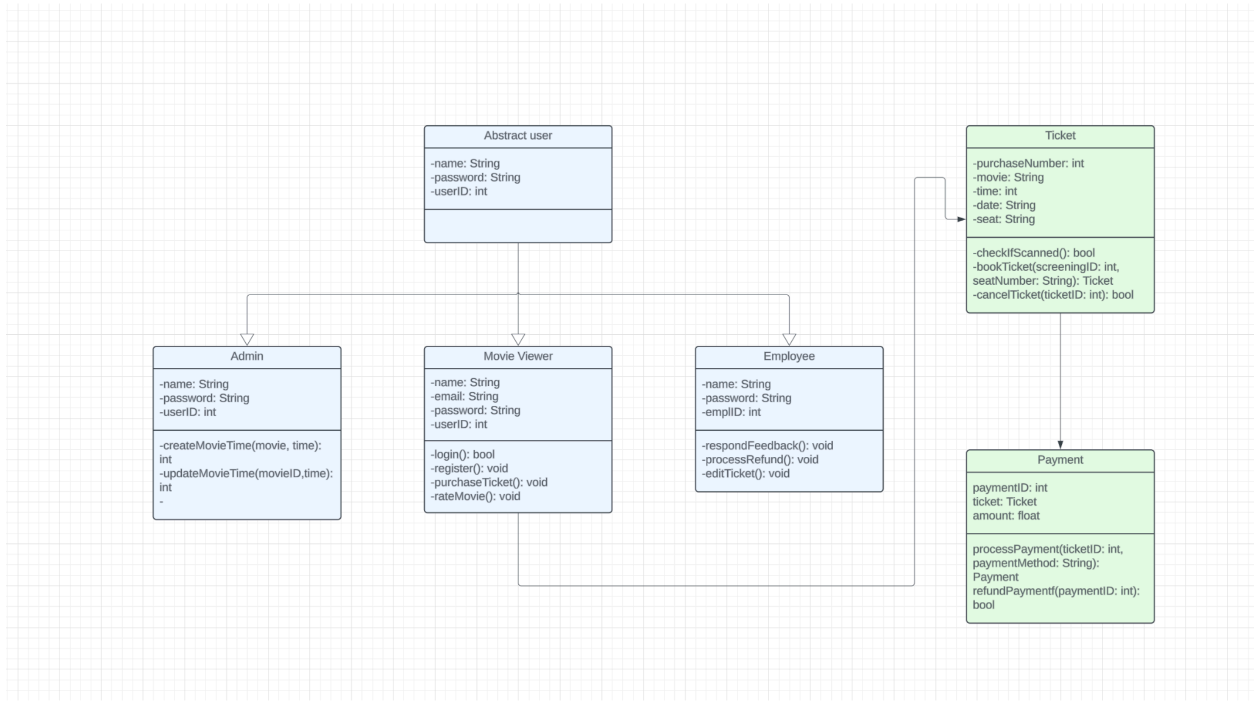
3.9 Other Requirements

Catchall section for any additional requirements.

4. Analysis Models

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

4.1 UML Diagram



4.1.1 UML Class Diagram Diagram

Class name:
Abstract User
Purpose:
Serves as the base class for different types of users within the system, defining common attributes and methods tha tall users share
Attributes:

- name: String
- password: String
- userID: int

Methods/Operations:
None specified, but might include basic user operations
Description:
Abstract user provides a blueprint for other user types. it contains common attributes like name, password, id, which are useful for identifying and authentication.

Class name:
Admin
Purpose:
Manage administrative tasks such as creating and updating movie times.
Attributes:

- name: String
- password: String
- userID: int

Methods/Operations:

- createMovieTime(movie, time): int
- updateMovieTime(movieID, time): int

Description:
The admin class extends AbstractUser and includes additional functionality such as managing movies.

Class name:
Movie Viewer
Purpose:
Represents a user who can view movies, register, login, and purchase tickets.
Attributes:

- name: String
- email: String
- password: String
- userID: int

Methods/Operations:

- login(): bool
- register(): void
- rateMovie(): void
- purchaseTicket(): void

Description:
The class extends the AbstractUser, and extends functionality by allowing Viewers to purchase tickets, login, and rate movies.

Class name:
Employee
Purpose:
Represents an employee within the theater who can perform specific tasks related to ticket management.
Attributes:
- name: String
- empID: int
- password: String

Methods/Operations:
- respondFeedback(): void
- processRefund(): void
- editTicket(): void

Description:
The class extends the AbstractUser, and provides additional functionality related to employees such as responding to feedback left by users, processing refunds, and editing tickets to assist viewers.

Class name: Ticket
Purpose:
To have a ticket object that can be purchased by a user.
Attributes:
- purchaseNumber: int
- movie: String
- time: int
- date: String
- seat: String

Methods/Operations:
- checkIfScanned(): bool
- bookTicket(screeningID: int, seatNumber: String): Ticket
- cancelTicket(ticketID: int): bool

Description:
The ticket class contains details about a movie ticket, including movie, time, date, and seat. It provides methods for checking if a ticket is scanned, to book a ticket, and to cancel a ticket (making it available for purchase again).

Class name: Payment
Purpose:
Handles payment transactions for purchasing tickets
Attributes:

- paymentID: int
- ticket: Ticket
- amount: float

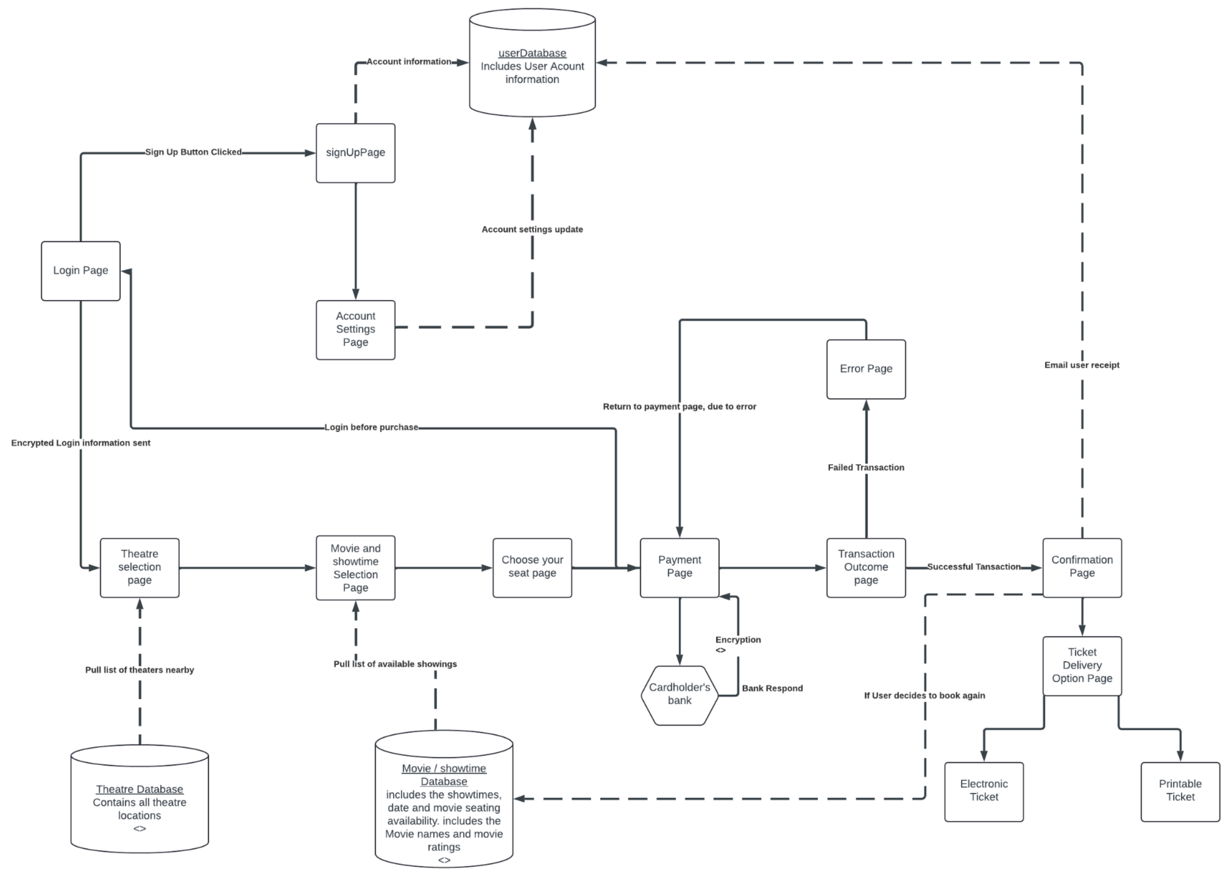
Methods/Operations:

- processPayment(ticketID: int, paymentMethod: String): Payment
- refundPayment(paymentID: int): bool

Description:

The Payment class manages payment transactions associated with ticket purchases. It includes attributes for payment ID, ticket, and amount, and methods for processing refunding payments.

4.2 Software Architecture Diagram


userDatabase
Includes User Acount information
Account information
Sign Up Button Clicked
signUpPage
Account settings update
Login Page
Account Settings Page
Email user receipt
Error Page
Return to payment page, due to error
Login before purchase
Encrypted Login information sent
Failed Transaction
Theatre selection page
Movie and showtime Selection Page
Choose your seat page
Payment Page
Transaction Outcome page
Successful Tansaction
Confirmation Page
Encryption
<>
Pull list of theaters nearby
Pull list of available showings
Cardholder's bank
Bank Respond
If User decides to book again
Ticket Delivery Option Page
Theatre Database
Contains all theatre locations
<>
Movie / showtime Database
includes the showtimes, date and movie seating availability. includes the Movie names and movie ratings
<>
Electronic Ticket
Printable Ticket

4.2.1 Architecture Diagram Description

The Architecture Diagram is a detailed representation outlining the Theater Ticketing System's main components and their interconnections. Each step is illustrated in this diagram, providing a visual representation of the operational flow that this website will incorporate. Additionally, the diagram includes the databases that will be utilized and describes their responsibilities.

The system begins with the "Login Page," where users can log in or navigate to the "Sign Up Page" to create a new account. From the "Sign Up Page," users can proceed to the "Account Settings Page" to update their account information. All account-related information is managed by the "User Database."

Once logged in, users are directed to the "Theatre Selection Page," where they can select a theater location. The information about theater locations is managed by the "Theatre Database." After selecting a theater, users proceed to the "Movie and Showtime Selection Page," where they can choose a movie and showtime. This page is connected to the "Movie/Showtime Database," which contains information about movie showtimes, dates, seating availability, movie names, and ratings.

Next, users navigate to the "Choose Your Seat Page" to select an available seat for the chosen showtime. After selecting a seat, users proceed to the "Payment Page" to make the ticket purchase. The system communicates with the "Cardholder's Bank" to process the payment. Depending on the bank's response, users are directed to the "Transaction Outcome Page," where they can see whether the transaction was successful or unsuccessful. If an error occurs, users are redirected to the "Error Page" and then given the option to return to the "Payment Page" to retry the transaction.

If the transaction is successful, users proceed to the "Confirmation Page" and then to the "Ticket Delivery Option Page," where they can choose between receiving an electronic ticket or a printable ticket. Upon successful transaction completion, an email receipt is sent to the user.

4.3 Development Plan

Brandon Mahdavi: Screening and Ticket Management Services

- Description: Develop services for managing screenings, booking tickets, and processing payments.

Riley Matthews: User and Movie Management Services

- Description: Implement services for user registration, login, and movie details retrieval.

Justin Nam: User Interface for Browsing Movies/Booking and Payment

- Description: Design and implement the interface for users to browse movies and view details, as well as, for users to select seats, book tickets, and make payments.

4.4 Timeline:

- Week 1-2**:**

  - Frontend: Initial design and setup of user interfaces (.
  - Backend: Setup of database and basic user and movie management services.
- Week 3-4**:**
  - Frontend: Development of ticket booking and payment interfaces.
  - Backend: Implementation of screening management and ticket booking services.
- Week 5-6**:**
  - Frontend: Integration testing of user interfaces.
  - Backend: Testing and refinement of backend services.
- Week 7-8**:**
  - Final integration and system testing.
  - Deployment and preparation for presentation.

5. Test Plans

5.1 Test Plan Introduction

This test plan ensures the movie theater ticketing system meets all specified requirements through comprehensive verification and validation. Each test is detailed to identify the features being tested, the test sets/vectors used, and how these tests cover the targeted features. The plan covers various scenarios, including normal operations, edge cases, and potential failure points, to ensure the system handles real-world usage effectively. This ensures that the system is functional, secure, and user-friendly before deployment.

5.2 Test Scope

The test scope encompasses functional requirements, unit tests, and system tests for the movie theater ticketing system. It includes thorough testing of user interfaces, ensuring that login, sign-up, movie browsing, seat selection, payment pages, and ticket confirmation interfaces are user-friendly and functional. The ticket booking process is rigorously tested to ensure that the entire workflow, from movie selection to seat reservation and ticket issuance, operates seamlessly. Payment processing is verified to ensure the secure handling of payment information, proper integration with the external payment service (cardholderBank), and accurate transaction validation. Backend functionalities are tested to ensure correct database interactions for user accounts, movie details, seat availability, and sales reporting, maintaining data accuracy and integrity. Sales reporting is tested by generating and validating reports for various time frames, ensuring data accuracy and comprehensive financial reporting. Additionally, non-functional aspects such as performance (handling peak loads), security (protecting sensitive data), usability (providing user-friendly interfaces), reliability (maintaining system uptime), and scalability (supporting growth) are thoroughly tested to ensure the system is robust, secure, and user-friendly.

5.3 Test Objectives and Strategy

Test Objectives

1. **Ensure all functional requirements are met:**
    - Validate that all specified functionalities of the system operate as intended.
    - Confirm that users can log in, sign up, browse movies, select seats, make payments, and receive ticket confirmations successfully.
2. **Verify the integration of different modules:**
    - Ensure smooth interaction between system components and external services.
    - Test the integration between the user interface, backend services, and the external payment service (cardholderBank).
3. **Validate the user experience:**
    - Assess the usability and overall user experience of the system.
    - Conduct user acceptance testing (UAT) to gather feedback from potential end-users and make necessary improvements.
4. **Ensure data integrity and security:**
    - Verify that user data, including personal information and payment details, is securely handled and stored.
    - Test for vulnerabilities and ensure compliance with data protection regulations.
    - Ensure accurate database interactions for user accounts, movie details, seat availability, and sales reporting.
5. **Ensure the system's performance and scalability:**
    - Test the system's ability to handle peak loads and high user traffic without performance issues.
    - Assess the system's reliability and uptime, ensuring it remains operational under various conditions.
    - Evaluate the system's scalability to support future growth and increased demand.

Test Strategy

The test strategy includes both verification and validation processes to ensure comprehensive testing of the movie theater ticketing system.

Verification involves:

- **Reviews and Inspections:**
    - Conduct reviews and inspections of requirements, design, and code to identify and resolve issues early in the development process.
- **Static Analysis:**
    - Perform static analysis to examine the code for potential errors and ensure adherence to coding standards.
- **Walkthroughs:**

- Conduct walkthroughs to review the logic and functionality of the system components with the development team.

Validation involves:

- **Dynamic Testing:**
    - **Functional Testing:** Validate that the system performs its intended functions correctly.
    - **Unit Testing:** Test individual components or units of the system to ensure they function correctly in isolation.
    - **Integration Testing:** Verify that different modules or components of the system work together as expected.
    - **System Testing:** Test the entire system to ensure it meets the specified requirements and performs as expected.
    - **Acceptance Testing:** Conduct tests to determine if the system meets the acceptance criteria and is ready for deployment.
- **Performance and Load Testing:**
    - Evaluate the system's performance under various load conditions to ensure it can handle peak usage without degradation.

5.4 Test Environment

The test environment is designed to simulate real-world conditions to ensure comprehensive testing of the movie theater ticketing system.

Hardware:

- Standard user devices, including PCs, smartphones, and tablets, to test the user interface and user experience.
- Servers for backend processing to test the system's performance, reliability, and scalability.

Software:

- **Operating Systems:** Windows, macOS, iOS, and Android to test compatibility across different platforms.
- **Web Browsers:** Chrome, Firefox, Safari to ensure the system works correctly on various browsers.
- **Testing Tools:**
    - **JUnit:** For unit testing to ensure individual components function correctly.
    - **Selenium:** For automated functional testing of the web interface.
    - **LoadRunner:** For performance and load testing to evaluate system behavior under peak loads.

By following this comprehensive test plan, we ensure the movie theater ticketing system is thoroughly tested to meet all specified requirements, ensuring it is functional, secure, user-friendly, and capable of handling real-world usage effectively.

5.5 Functional Requirements

Functional requirements to be tested include:

1. **User Registration and Login:**
    - **User Registration:** Validate the account creation process, ensuring users can register with necessary details, and verify their email. Test handling of duplicate emails and password strength requirements.
    - **User Login:** Ensure registered users can log in with valid credentials and handle incorrect login attempts appropriately. Verify successful logout and session data clearance.
    - **Password Recovery:** Test the password reset process via email, ensuring secure password reset and compliance with strength criteria.
2. **Movie Listing:**
    - **Browse Movies:** Verify that users can view a list of available movies with essential details and that the list updates in real-time. Test pagination and loading times for a smooth user experience.
    - **Search and Filter:** Ensure users can search for movies by title and apply filters based on genre, rating, and showtimes. Validate the accuracy of search results and filter application.
    - **Movie Details:** Confirm that users can access detailed information about movies, including synopsis, cast, showtimes, trailers, and user reviews, ensuring multimedia elements function correctly.
3. **Seat Selection:**
    - **Seat Availability:** Ensure users can view real-time seating layouts showing available, booked, and reserved seats.
    - **Seat Selection:** Validate that users can select preferred seats and the system accurately reflects their selection, handling errors for unavailable seats.
    - **Seat Confirmation:** Verify that selected seats are temporarily reserved for users during payment processing and update availability appropriately.
4. **Ticket Booking:**
    - **Booking Process:** Ensure a smooth flow from movie selection to seat reservation and ticket confirmation. Allow users to review and modify booking details before payment.
    - **Review Booking:** Validate that the review page accurately displays selections, additional charges, and fees, and allows for modifications.
    - **Booking Confirmation:** Confirm that the system updates seat availability and provides a booking reference number upon successful payment.
5. **Payment Processing:**
    - **Payment Options:** Ensure users can choose from various payment methods and handle them securely.
    - **Payment Gateway Integration:** Validate secure integration with the external payment service (cardholderBank) and appropriate handling of payment responses.

    - **Transaction Validation:** Verify accurate transaction processing, user notifications for successful and failed payments, and proper handling of edge cases like network interruptions.
6. **Confirmation and Notification:**
    - **Booking Confirmation:** Ensure users receive immediate booking confirmation on the website with all relevant details.
    - **Email Notification:** Validate prompt email confirmation with ticket details and instructions, ensuring accurate and well-formatted content.
    - **Notification Handling:** Test the system's ability to send accurate notifications for booking status updates, ensuring timely and helpful information for users.

By thoroughly testing these functional requirements, we ensure that the movie theater ticketing system is fully functional, secure, and user-friendly, providing a seamless experience from registration to ticket booking and confirmation.

5.6 Test Cases

5.6.1 Functional Test Cases

User Registration and Login

- TC01: Verify user registration with valid data.
- TC02: Verify user registration with invalid data.

Movie Listing

- TC03: Verify displaying of all available movies.

Seat Selection

- TC04: Verify availability of seats.

Ticket Booking

- TC05: Verify booking of selected seats.

Payment Processing

- TC06: Verify successful payment processing.

5.6.2 Unit Test Cases

Payment Gateway Response

- TC07: Verify function for handling payment gateway responses

Booking Reference Number

- TC08: Verify function for generating booking reference number.

5.6.3 System Test Cases

System Stress Testing

- TC09: Verify handling of simultaneous ticket bookings.

- TC10: Verify load handling for concurrent users.

5.7 Entry and Exit Criteria

Entry Criteria:

- All functional and non-functional requirements must be thoroughly documented, clarified, and approved by key stakeholders, including business analysts and project managers
    - **Test Plan Document Deliverable:** A detailed document outlining the test strategy, objectives, scope, and approach, including entry and exit criteria and risk management strategies (Section 5.9).
        - **Test Planning:** Develop and approve the test plan and strategy. Expected deadline is 1 week.
- The test environment is set up. Configure Hardware and Software environments to mimic the production setup, ensuring all the necessary tools, applications, and network configurations are in place and operational.
        - **Test Environment Setup:** Configure and validate the test environment and prepare test data. Expected deadline is 1 week.
- Prepare comprehensive test data covering normal, boundary, and edge test cases, ensuring compliance with data protection regulation.
        - **Test Case Development:** Create, review, and approve detailed test cases. Expected deadline is 2 weeks.
- Conduct peer reviews of test cases to ensure they are comprehensive and aligned with requirements, followed by formal approval from test managers and business analysts (Section 5.8).
    - **Test Cases Deliverable:** Comprehensive test cases covering all the requirements, with clear steps and expected results.

Exit Criteria:

- All planned test cases must be executed, with accurate recording of results. Any deviations must be documented and analyzed.
    - **Test Script Deliverable:** Automated scripts for functional and regression testing to ensure consistency and efficiency.
        - **Test Execution:** Execute test cases, track progress, and report defects. Expected deadline is 4 weeks.
    - **Test Defect Report Deliverable:** Comprehensive reports on identified defects, including severity, impact, and resolution status.
- Prioritize and resolve all critical and high priority defects, verifying fixes through retesting and ensuring no major issues remain.
        - **Test Defect Fixing and Retesting:** Expected Deadline is 2 weeks.

- Ensure all functional and non-functional requirements are met and the system performs as expected in all scenarios.
    - **Test Summary Report Deliverable:** A summary of the testing process, key findings, overall results, and recommendations for future improvements.
- Compile a comprehensive test summary report, including test execution status, defect metrics, and key findings. Review and approval by stakeholders are necessary (Section 5.10).
        - **Test Closure:** Compile and review the test summary report, obtain formal sign-off. Expected deadline is 1 week.

5.8 Testing Responsibilities

Clear roles and responsibilities are essential for an efficient testing process. Below are the key roles and their detailed responsibilities:

- **Test Manager**
    - Planning and Coordination:
        - Develop and manage the test strategy and plan.
        - Allocate resources and ensure schedule adherence.
        - Oversee test environment setup.
        - Monitor testing progress and communicate with stakeholders.
- **Test Engineers**
    - Test Case Design and Execution:
        - Design detailed test cases covering all scenarios.
        - Execute test cases, record results, and report defects.
        - Automate test scripts for regression testing.
        - Participate in test reviews and provide feedback.
- **Developers**
    - Defect Resolution:
        - Analyze and fix reported defects.
        - Conduct root cause analysis and implement fixes.
        - Collaborate with the test engineers for accurate issue reproduction.
        - Participate in code reviews to maintain quality.
- **Business Analysts**
    - Requirement Validation:
        - Review and validate test cases against requirements.
        - Provide requirement clarifications and additional details.
        - Participate in user acceptance testing.
        - Ensure requirements traceability throughout testing.

5.9 Risk Management

Effective risk management is essential for a smooth testing process and timely project delivery. Here's a detailed look at the potential risks, their implications, and mitigation strategies:

Identifying Risks

1. **Delays in Environment Setup:**
   - **Implications:**
     - Delays can push back the testing schedule, affecting the overall project timeline.
     - A poorly configured test environment can lead to inadequate testing and undetected defects.
2. **Incomplete Requirements:**
   - **Implications:**
     - Missing or unclear requirements can result in gaps in test cases, leading to undetected defects.
     - Constant requirement changes can cause rework and delays in testing.
3. **High Defect Rates:**
   - **Implications:**
     - A high number of defects, especially critical ones, can delay testing and development as time is diverted to fixing issues.
     - High defect rates can affect team morale and stakeholder confidence.

Mitigation Strategies

1. **Delays in Environment Setup:**
   - **Advance Planning:**
     - Plan the environment setup well in advance, identifying necessary resources and configurations.
   - **Buffer Time:**
     - Allocate buffer time in the schedule to accommodate potential delays.
   - **Regular Monitoring:**
     - Monitor setup progress closely and address issues promptly.
   - **Backup Environments:**
     - Ensure backup environments are available to avoid disruptions.
2. **Incomplete Requirements:**
   - **Continuous Communication:**
     - Maintain open communication with stakeholders to clarify and finalize requirements.
   - **Requirements Traceability Matrix:**
     - Use a traceability matrix to ensure all requirements are covered in test cases and tracked.
   - **Regular Review Sessions:**
     - Conduct regular review sessions to address gaps and ambiguities promptly.

- **Documentation:**
  - Document all clarifications and changes, ensuring they are communicated to the team.

3. **High Defect Rates:**
   - **Regular Code Reviews:**
     - Conduct regular code reviews to identify and address potential issues early.
   - **Early Testing:**
     - Implement early testing phases like unit and integration testing to catch defects early.
   - **Developer Training:**
     - Provide training to developers to improve code quality and reduce defects.
   - **Defect Management Process:**
     - Implement a robust defect management process to prioritize and resolve issues efficiently.

Risk Monitoring and Reporting

1. **Regular Risk Assessments:**
   - Conduct regular assessments to identify new risks and reassess existing ones.
   - Use risk assessment meetings and reports to keep the team informed.
2. **Risk Log:**
   - Maintain a risk log to document identified risks, mitigation strategies, and current status.
   - Update and share the risk log regularly with stakeholders.
3. **Stakeholder Communication:**
   - Keep stakeholders informed about risk status and mitigation efforts through regular updates and meetings.
   - Escalate significant risks to decision-makers promptly.

By identifying potential risks early and implementing detailed mitigation strategies, we can minimize their impact on the testing process. Regular monitoring and transparent communication ensure that all stakeholders are aware of the risks and the steps being taken to address them, leading to a smoother and more successful project delivery.

5.10 Approval

The approval process ensures that all stakeholders are aligned and committed to the testing plan. Here's how we handle it:

Approval Process

1. **Initial Review:**

- **Draft Review:** The test manager drafts the test plan, incorporating input from the team.
- **Internal Feedback:** Circulate the draft for internal feedback to ensure it covers all aspects.

2. **Stakeholder Presentation:**
    - **Meeting Setup:** Schedule a meeting with project managers, test managers, business analysts, and senior developers.
    - **Presentation:** The test manager presents the test plan, covering strategy, objectives, scope, deliverables, entry and exit criteria, schedule, and risk management.
3. **Stakeholder Review and Feedback:**
    - **Detailed Review:** Stakeholders review the test plan in detail, ensuring alignment with project goals.
    - **Feedback Collection:** Collect feedback through meetings, emails, or collaborative tools.
4. **Incorporation of Feedback:**
    - **Revisions:** The test manager incorporates feedback, refining test scenarios and timelines.
    - **Final Draft:** Prepare a final draft addressing all feedback.
5. **Formal Approval:**
    - **Sign-off Document:** Prepare a formal sign-off document summarizing key elements and confirming stakeholder agreement.
    - **Approval Signatures:** Obtain signatures from:
        - **Project Manager:** Brandon Ensures alignment with project goals and resource readiness.
        - **Test Manager:** Justin Confirms test strategy and resource preparation.
        - **Business Analysts:** Riley Validates coverage of all business requirements and clarifications.

Importance of Approval

- **Alignment:** Ensures all stakeholders understand and agree on the testing strategy and objectives.
- **Commitment:** Confirms stakeholder support for testing activities and issue resolution.
- **Accountability:** Establishes clear accountability for the execution and defect resolution in the test plan.

This structured approval process ensures our test plan is comprehensive and aligned with project goals. It fosters collaboration and accountability, setting the stage for a successful testing phase and a high-quality movie theater ticketing system ready for deployment.

6. Data Management Plan

6.1 Database Selection

For the ticketing system, we will use SQL databases to capitalize on the related characteristics that the data has. Users, movies, and theaters will have their own individual databases to keep data organized and to optimize performance and security.

6.2 Design Decisions

We will have 5 tables, and they will be: users, movies, tickets, and payments.

6.3 Data Storage

Data will be stored using SQL databases. Data normalization will be used to minimize redundancy. To increase speed of access to information like show times and seat availability indexing will be used.

6.4 Security

Sensitive information such as user data and passwords will be encrypted using industry standard techniques. Credit card information will be encrypted using ACID-compliant processes during transactions, but will not be stored for future use to avoid potential costs associated with added security. Access control will be implemented to prevent certain users from accessing data. Periodic audits will occur in order to check the security of the system along with annual pen testing to identify potential weaknesses.

6.5 Backups

Frequent backups will occur to attempt to prevent data loss in the event that a system fails. Due to the smaller size of the information being handled the full database will be backed up daily. Partial backups can be manually performed by administrators to save crucial data. Backups will be stored off site.

6.6 Scalability

The management system will be designed to handle horizontal scaling in the future. For the time being with the limited users vertical scaling will be used. Resources will be poured into maximizing the servers being used instead of adding them initially. Once capacity is being approached the system will be expanded horizontally to avoid potential bottlenecks.

6.7 Design Tradeoffs and Justification

Alternatives considered:

- NoSQL
- Single Database

- Single table

Multiple databases and SQL were used because they allow for scalability and the ACID guarantees of SQL. Multiple databases are more complex than single databases, and given the size that the initial system will be handling a single database might make more sense. However, we are looking forward to scalability, performance and security. The isolation of data that multiple databases provide aligns with our vision for long term security. While maintaining data across multiple databases may result in inconsistencies that will require more resources to fix, it allows for only the necessary data to be handled which can help with security and speed.

To continue, NoSQL offers great scalability, but given the large amount of transactions that will be occurring in our system we wanted to use a relational database that offered the ACID guarantees like SQL. The increased security is something that we value for our users.

To conclude some of the downfalls of the systems chosen are increased complexity, data inconsistency across multiple databases, operational overhead, development complexity. Some of the justifications for why we chose it are performance, security, scalability, and isolation of data.

7.1 Github Repository:
Github

4.3 Data Flow Diagrams (DFD)

4.2 State-Transition Diagrams (STD)

5. Change Management Process

Identify and describe the process that will be used to update the SRS, as needed, when project scope or requirements change. Who can submit changes and by what means, and how will these changes be approved.

A. Appendices

Appendices may be used to provide additional (and hopefully helpful) information. If present, the SRS should explicitly state whether the information contained within an appendix is to be considered as a part of the SRS's overall set of requirements.

Example Appendices could include (initial) conceptual documents for the software project, marketing materials, minutes of meetings with the customer(s), etc.

A.1 Appendix 1

A.2 Appendix 2