TrustyTicket

Software Requirements Specification

Version 0.9.8

September 26, 2024

Group 2
Justin Pelak, Lucas Fredricks, Andy Boyd,
Gilbert Venegas

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

Fall 2023

Revision History

Date	Description	Author	Comments
9/25/24	Version 0.8.1	Justin Pelak	Initial SRS section writing
10/8/24	Version 0.9.8	Justin Pelak	Diagrams/descriptions

Document Approval

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

Signature	Printed Name	Title	Date
	Justin Pelak	Software Eng.	
	Dr. Gus Hanna	Instructor, CS 250	

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this SRS is to provide our clients the ability to check that the system meets their needs, management a plan a bid for the system and the development process, give system engineers the ability to understand what is being developed, test engineers the ability to create validation tests, and maintenance engineers the ability to understand the system and relationships between parts. This document is based on IEEE/EIA 12207.1-1997 standard.

1.2 Scope

This document specifies requirements for a movie ticketing system for (our client).

This application will:

- Handle the payments of tickets
- Allow native language support for English, Spanish, and Swedish
- Localize ticket prices
- Keep prices consistent across the system
- Accept payment through PayPal, Bitcoin, and credit card
- Limit transactions to 20 tickets
- Only allow a single device concurrently with a single user account
- Keep daily logs of ticket purchases
- Allow users to register for loyalty accounts which stores personal information, purchase history, and loyalty points
- Allow tickets to be returned or exchanged prior to showing, only if purchased with user account
- Allow 10 million concurrent users
- Not allow replicable tickets

The system will be hosted using Amazon Web Services (AWS) and will store information in an encrypted database using Amazon Relational Database Service (RDS). The front-end will run in web browsers on Windows, Linux, and MacOS.

1.3 Definitions, Acronyms, and Abbreviations

SRS: Software Requirements Specification
NFT: Non-fungible Token
AWS: Amazon Web Services
RDS: Relational Database Service
GUI: Graphical user interface; visual interface for user interaction
HTTPS: HyperText Transfer Protocol Secure; used for securely transmitting data over the web

Loyalty Account: A user account system that tracks purchase history and points that can be redeemed for rewards
PCI DSS: Payment Card Industry Data Security Standard

1.4 References

This document is being used in conjunction with the following publications

- IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications
- ISO/IEC 25010
- PCI DSS Standards
- AWS Documentation
- Tableau Documentation
- ActiveMQ Documentation
- World Currency API Documentation

1.5 Overview

This document contains a description of the system's purpose, scope, the specifications, and use cases involved. It is organized as follows:

- Description of the system and its environment
- Requirements, both functional and non-functional
- Analysis models including sequence diagrams, data flow diagrams, and state-transition diagrams
- Process for managing changes to the system

2. General Description

2.1 Product Perspective

The theater ticketing system will replace the currently existing slow and buggy system used by the theater chain. The new system will handle ticket sales, manage user accounts, and provide features like seat reservation, ticket exchange, and feedback collection. It will be web-based and capable of handling high traffic.

2.2 Product Functions

The system will perform the following functions:

- Sell tickets online and in person, syncing data in real-time
- Display movie showtimes, seat availability, and ticket prices
- Allow users to select seats and purchase tickets for a specific showing
- Support loyalty accounts, discounts (student, military, senior), and different pricing tiers (regular v. deluxe).

- Handle payments and generate NFT-style tickets
- Offer administrator features to resolve ticketing issues and modify showtimes

2.3 User Characteristics

The users of the system include:

- Customers (General Users). They can purchase tickets, create user accounts, or cancel/exchange tickets through the web or in-theater kiosks. They can edit personal information, delete personal information, TBD through the web.
- Theater administrators and staff (Administrative Users). They can purchase tickets, create user accounts, edit personal information, delete personal information of/for General Users.

2.4 General Constraints

The system MUST be secure (due to handling highly sensitive information), reliable, and handle up to 10 million concurrent users. The system must be fully compatible with Windows, Linux, and macOS. It must be accessible in web browsers (Chrome and Firefox) and support real-time updates without requiring frequent downtime for system updates. The system will allow on device logged into a user account concurrently, and users will only be able to purchase 20 tickets. Tickets must be totally unique NFTs that are securely issued and tracked. Internet access should be required to purchase and reserve tickets, but users will still have access to their purchased tickets and showtimes without a reliable internet connection.

2.5 Assumptions and Dependencies

The system assumes that the necessary hardware infrastructure is available and Amazon's web services are operational at all times. Additionally, the system does depend on third-party payment gateways, such as PayPal, Bitcoin, and credit card providers for handling transactions.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The system will have a user-friendly GUI accessible via modern web browsers, with a focus on Chrome and Firefox. The interface will support English, Spanish, and Swedish based on the user's preference. Users will be able to browse showtimes, select specific seats, register loyalty accounts, and complete transactions. The interface will be optimized for accessibility, including keyboard navigation, screen reader compatibility, and color contrast options.

3.1.2 Hardware Interfaces

The system will not require any specific hardware interfaces. Users must have access to their own devices (PCs, laptops, mobile phones, etc.) which support modern web browsers with

internet access. In-theater kiosks will run the newest version of Windows 10 to ensure compatibility with the system.

3.1.3 Software Interfaces

The system will connect with AWS for hosting, storage, and data encryption. Ticket logs will be visualized and analyzed using Tableau software. integrate external payment gateways to process ticket purchases. Currency conversion services will be integrated to adjust prices based on the user's local currency.

3.1.4 Communications Interfaces

Communication between the client's web interface and the server will be conducted via HTTPS. The system will use Apache ActiveMQ for message handling and communication between the database and payment gateway.

3.2 Functional Requirements

3.2.1 Ticket Purchase and Reservation

3.2.1.1 Introduction: Users will be able to purchase tickets and reserve seats for any available screening.
3.2.1.2 Inputs: User login details, showtime selection, number of tickets, selected seats, payment information.
3.2.1.3 Processing: Verify ticket availability, process payment, and issue NFT tickets.
3.2.1.4 Outputs: Confirmation of ticket purchase, transaction ID, and access to the purchased tickets (including offline access).
3.2.1.5 Error Handling: If tickets are unavailable or payment fails, the system will alert the user and allow them to retry the transaction or select a different showing.

3.2.2 Loyalty Accounts

3.2.2.1 Introduction: Users can create accounts to store personal information, payment methods, and purchase history. Loyalty points will be accrued with each purchase.
3.2.2.2 Inputs: User registration details, payment method, loyalty preferences.
3.2.2.3 Processing: Store user information accordingly, securely. Calculate loyalty points per transaction and track purchase history.
3.2.2.4 Outputs: Account creation confirmation, access to loyalty points and purchase history.
3.2.2.5 Error Handling: If the registration process fails or the user inputs incorrect information, the system will prompt for corrections accordingly.

3.2.3 Currency Conversion

3.2.3.1 Introduction: The system will automatically convert ticket prices based on the user's local currency.
3.2.3.2 Inputs: User's location (via browser or account settings).
3.2.3.3 Processing: The system will retrieve real-time currency conversion rates from a third-party service and display ticket prices accordingly.

3.2.3.4 Outputs: Ticket prices display in the local currency.
3.2.3.5 Error Handling: If conversion services are unavailable, the system will default to displaying prices in USD with a notification to the user.

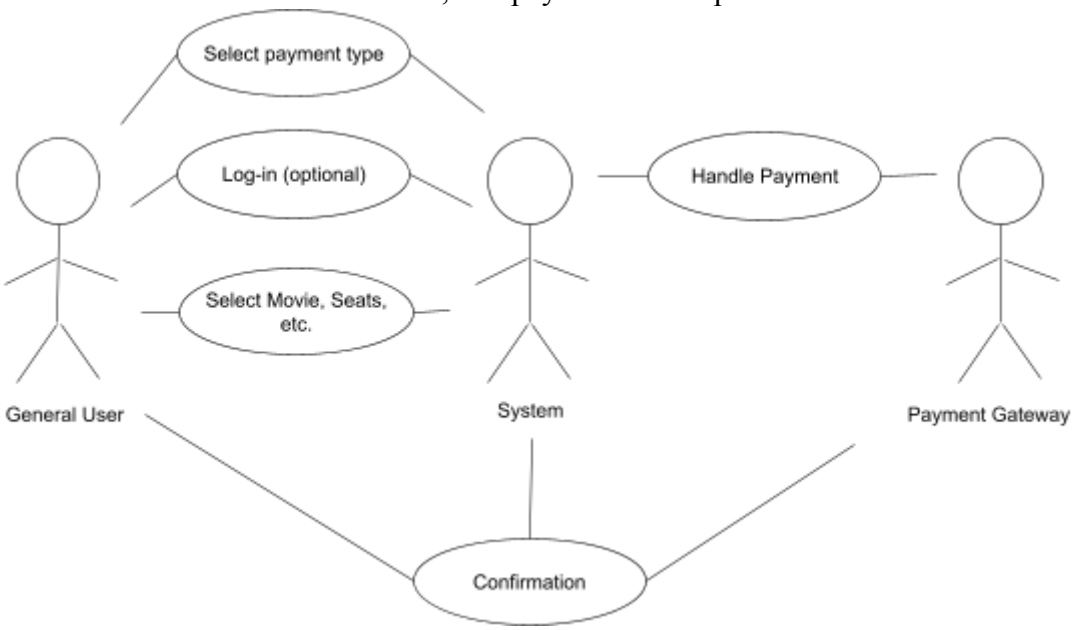
3.3 Use Cases

3.3.1 Use Case #1: Purchasing Tickets

Actors: User, System, Payment Gateway
Precondition: The user must be logged in and a ticket must be available for the desired showing.
Scenario:

- User selects a movie showing, number of tickets and seats
- User provides payment info
- System processes and generates NFT tickets
- User receives confirmation.

Postcondition: Tickets are reserved, and payment is completed.

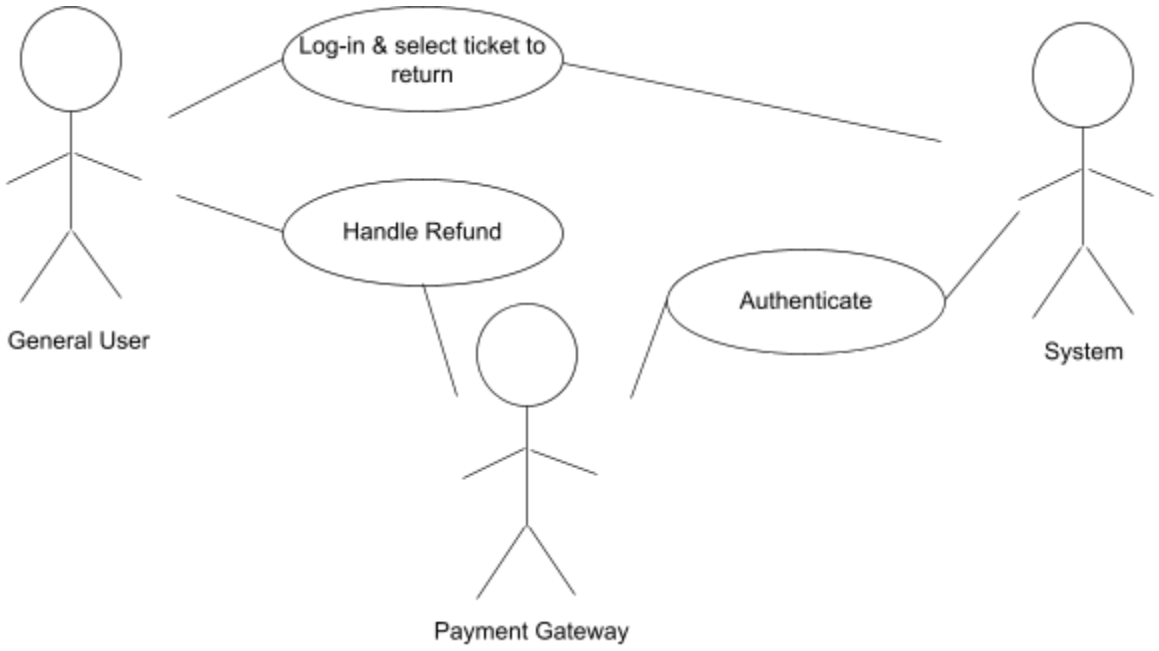


3.3.2 Use Case #2: Returning Tickets

Actors: User, System, Payment Gateway
Precondition: Ticket must have been purchased with a registered user account.
Scenario:

- User logs into their account
- User selects the ticket they wish to return
- System verifies the return policy (i.e., ticket is returned before showing)
- If eligible, the system processes the return.
- Users receive a refund to their original payment method.

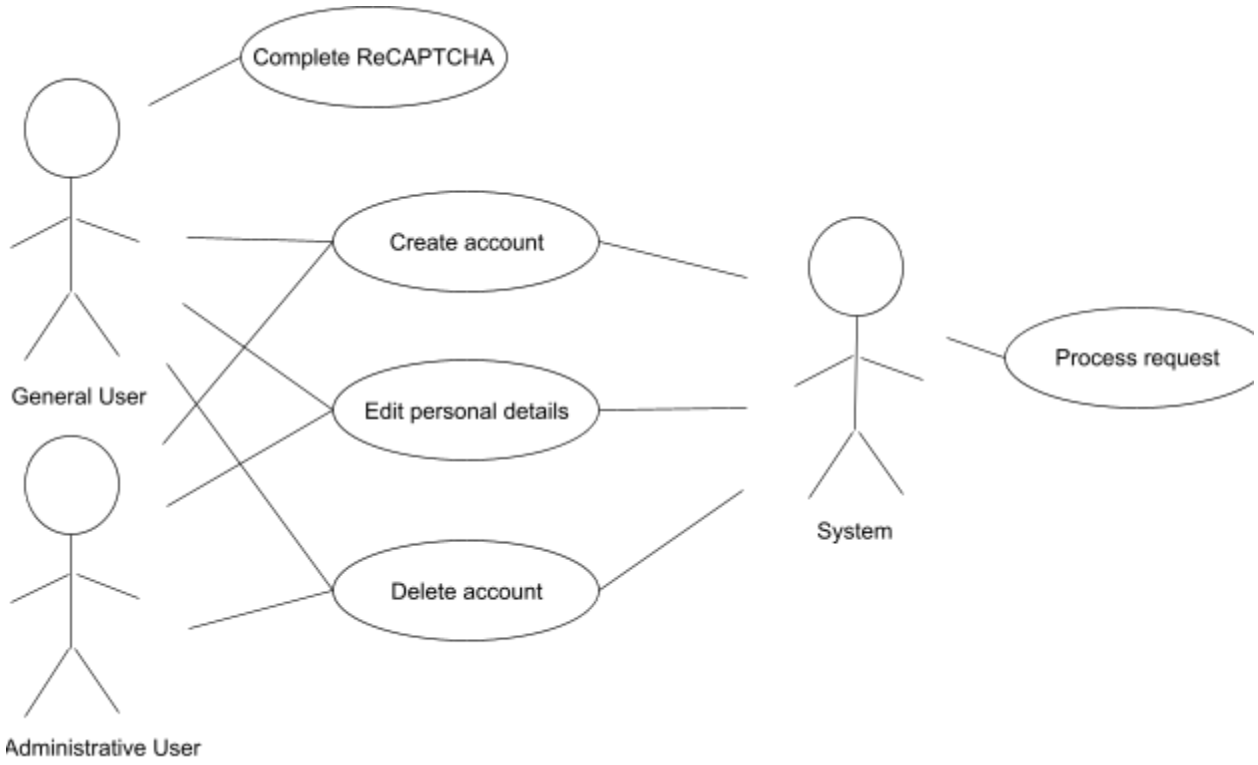
Postcondition: Tickets are returned, user's account is updated.



3.3.3 Use Case #3: User Account Management

Actors: User, System Administrator, System
Preconditions: User must be registered or have an account to access account management features
Scenario:

- User accesses Account Management
- User selects Account Management options (edit information/delete account)
- User passes ReCAPTCHA and enters valid login details (if applicable)
- Option 1: User selects to create a new account
    - Prompted to enter required information
    - System validates information and creates account if all criteria are met
    - Confirmation is sent to the user
- Option 2: If the user selects to edit information, they can modify personal information, payment methods, or preferences.
    - System validates new information
    - User saves changes and the system updates the details accordingly.
- Option 3: User selects delete account
    - Prompt the user to confirm their choice
    - System processes deletion request, removing the user's account and data
    - A confirmation message is displayed
- User logs out
- All previous actions are also accessible by System Administrators for other users (i.e., System Administrators can create new accounts for others, edit information, and delete accounts)



3.4 Classes / Objects

3.4.1 Ticket

3.4.1.1 Attributes: Ticket ID (NFT), showtime, seat number, price, user account (if applicable)
3.4.1.2 Functions: Generate unique ticket ID, reserve seat, validate ticket

3.4.2 User Account

3.4.1.1 Attributes: User ID, name, email, loyalty points, purchase history
3.4.1.2 Functions: Register account, store payment information, track loyalty points, edit personal information, delete account

3.4.2 Movies

3.4.1.1 Attributes: Film name, showtimes, genres, new releases
3.4.1.2 Functions: Sorting movies by new releases, sort based on availability, filter by genre

3.5 Non-Functional Requirements

3.5.1 Performance

- The system must handle up to 10 million concurrent users without significant performance degradation.
- 95% of transactions shall be processed within 1 second.

3.5.2 Reliability

- The system shall have an uptime of 99.9%, with regular backups of transaction data.

3.5.3 Availability

- Users must be able to access their tickets and showtimes offline after purchasing.

3.5.4 Security

- Tickets must be unique NFTs to prevent fraud.
- User data, including personal and payment information, must be encrypted at rest and in transit.

3.5.5 Maintainability

- The system's architecture will support modular updates to individual components (e.g., user interface).

3.5.6 Portability

- The system will be compatible with multiple platforms and run on modern web browsers.

3.6 Inverse Requirements

- The system shall not allow tickets to be returned after the showtime has passed.
- Tickets purchased without a user account cannot be returned or exchanged.

3.7 Design Constraints

- The system must fully comply with security standards such as PCI DSS for handling credit card payments.
- The use of NFTs must comply with blockchain and cryptocurrency regulations.

3.8 Logical Database Requirements

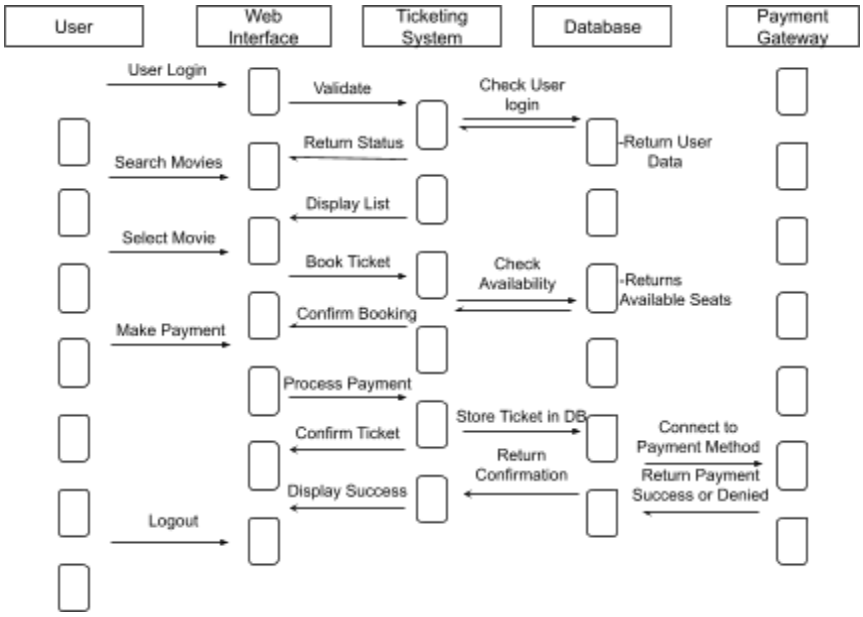
- The system will use Amazon's RDS for storing user data, ticket information, and transaction logs.
- Data must be encrypted and regularly backed up.

3.9 Other Requirements

- The system should support integration with third-party marketing services to offer promotions to users based on their purchase history.

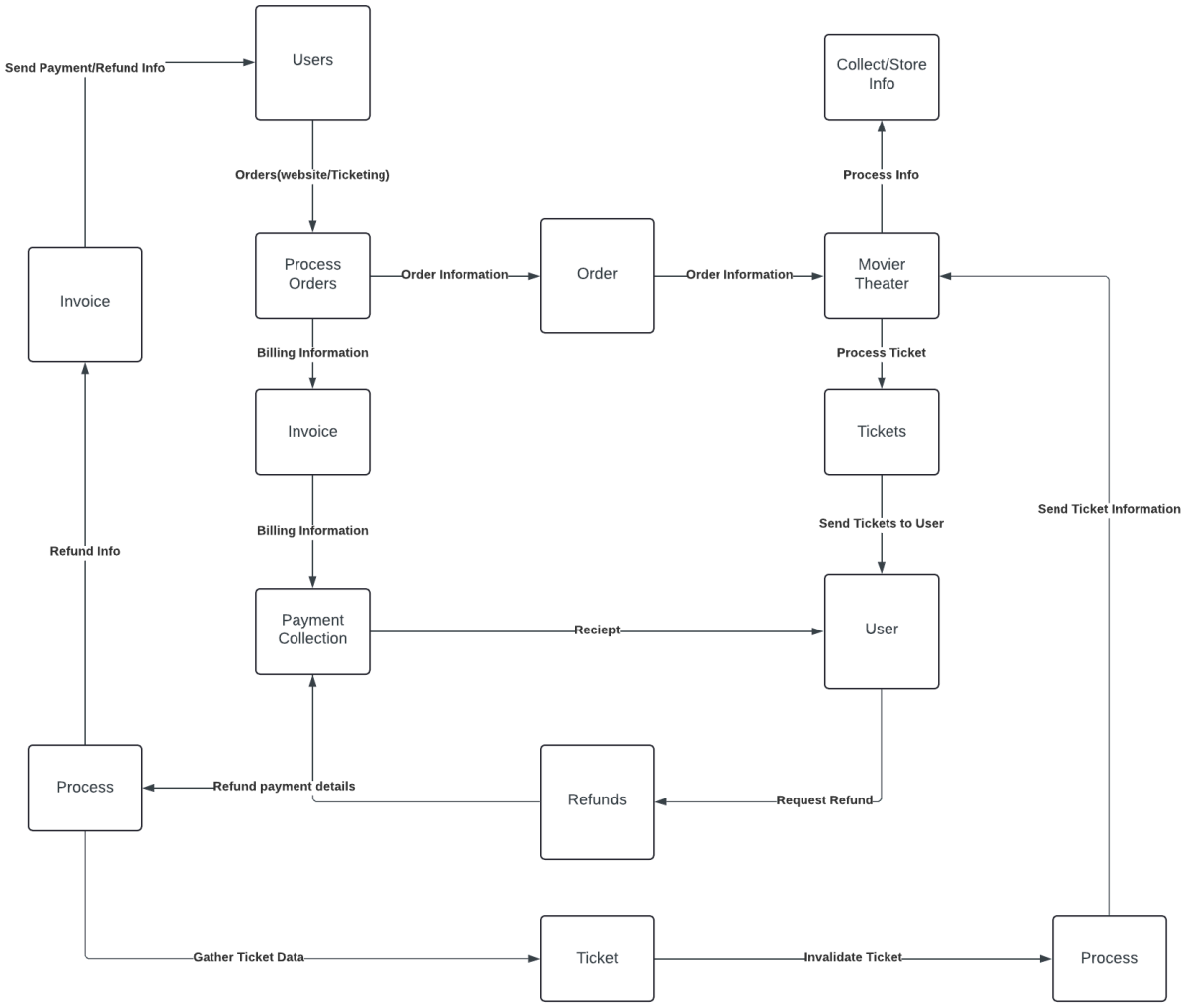
4. Analysis Models

4.1 Sequence Diagrams



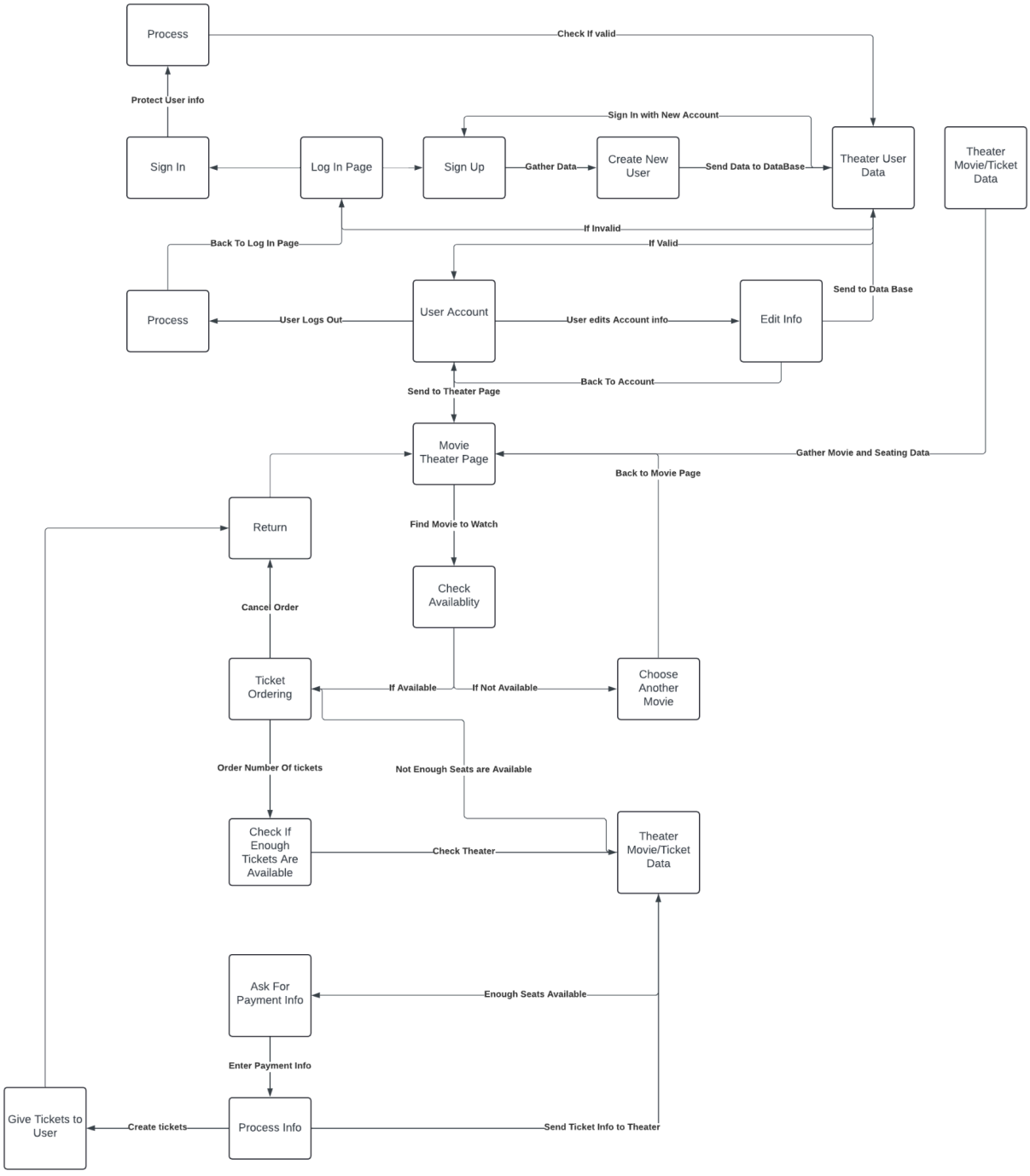
1. User Interaction
    - User logs in, searches for movies, selects a movie, and proceeds to make a payment
    - The user can log out once the process is completed
2. Web Interface
    - Validates user login credentials and communicates with the ticketing system to return user status and movie list availability.
    - Processes the movie selection and confirms bookings with the ticketing system
    - Checks seat availability and returns it to the user.
    - Displays success or failure messages back to the user
3. Ticketing System
    - System checks the login against the database and verifies availability during the booking
    - Confirms the booking before proceeding to payment processing
    - Once payment is done, it stores the ticket in the database and provides confirmation back to the interface
4. Database
    - Retrieves user data and checks ticket availability
    - Processes users payment and stores tickets in the database.
    - Stores the confirmed ticket after booking
5. Payment Gateway
    - Handles payment processing, sending confirmation or failure messages back to the ticketing system

4.2 Data Flow Diagrams (DFD)



1. Users
    - Users send orders for movie tickets, which are processed through the system.
    - They interact with payment collection and invoice systems for billing and can also request refunds.
2. Movie Theater and Tickets
    - In case of refunds, tickets can be invalidated or updated.
    - The theater processes orders and tickets, which are eventually sent to the user.
3. Payment Collection and Invoice
    - Payment collection systems generate invoices and receipts.
    - They interact with refund processes to handle payment adjustments if needed.
4. Refund Process
    - Refunds are initiated upon user request, involving the processing of both ticket and payment details.
    - This is a loopback where refund information is fed into invoices and payment systems to ensure the user receives their payment

4.3 State-Transition Diagrams (STD)



1. User Sign-In Process
    - User signs in through the log-in page or creates a new account via the sign-up option
    - In the sign-up flow, system gathers user data and sends it to the database. If successful sign-up, user is redirected to their account page
2. Account Management
    - If signed in, the user accesses their user account, where they can log out or edit their account information

- Edits to the account are also sent back to the database to ensure data accuracy

3. Movie Selection
    - After logging in or managing the account, the user is redirected to the movie theater page, where they can browse available movies
    - User can select a movie and check its availability. If the selected movie is unavailable, user can choose another movie
4. Ticket Ordering
    - If movie is available, the user proceeds to the “Ticket Ordering” phase.
    - System checks the availability of seats. If there are not enough seats, user can either choose another movie or cancel the order
    - If seats are available, system prompts user for payment information
5. Payment and Ticket Issuance
    - User enters payment details, which are processed by the system. Ifsuccessful payment, the system generates the tickets
    - Ticket information is sent to the theater, and the tickets are provided to the user
6. Cancellation
    - Throughout, user is able to return to the previous step or cancel the order entirely

4.4 Test Plan

The Test Plan for TrustyTicket can be found at the following GitHub link:
https://github.com/BoostedJ/CS250/blob/main/SDS/TrustyTicketTestPlan.pdf

The Test cases for TrustyTicket can be found at the following GitHub links:
https://github.com/BoostedJ/CS250/blob/main/SDS/SDSTestCases.pdf (PDF)
https://github.com/BoostedJ/CS250/blob/main/SDS/SDSTestCases.xlsx (Spreadsheet)

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