# Ticket System

## Software Requirements Specification

## Version 2

## 03/06/25

Group 2
Malaika Joiner
Julie Tong
Aditya Ujawane

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

## Revision History

| Date | Description | Author | Comments |
|---|---|---|---|
| <date> | <Version 1> | <Your Name> | <First Revision> |
| | | | |
| | | | |
| | | | |

## 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 purpose of this Software Requirements Specification (SRS) document is to provide a detailed description of the requirements for the Movie Ticketing App. This document will outline the application's functionalities, constraints, and dependencies to help software engineers design and implement the software efficiently. The document follows the IEEE Guide to SRS and serves as a reference for all project stakeholders, including developers, testers, and business analysts.

## 1.1 Purpose

The Movie Ticketing App aims to provide a seamless platform for users to browse, book, and manage movie tickets online. The intended audience includes:

- **End-users (Customers)**: Users who want to book movie tickets for themselves and others.
- **Cinema Administrators**: Responsible for managing movie schedules, seat availability, pricing, and promotions.
- **Developers and Testers**: Software engineers and QA testers are responsible for building, testing, and maintaining the app.
- **Business Stakeholders**: Decision-makers aiming to improve the efficiency and profitability of the movie ticketing business.

## 1.2 Scope

The Movie Ticketing App will:

- Allow users to **search** for movies, view showtimes, and book tickets.
- Enable secure **online payment** and provide digital tickets with QR codes.
- Offer **seat selection** based on real-time seat availability.
- Provide **user authentication** and profile management for personalized recommendations.
- Allow cinemas to **manage movie listings, schedules, pricing, and discounts.**
- Send **notifications and reminders** for upcoming bookings.
- Integrate a **loyalty and rewards system** to incentivize frequent bookings.

The app will **not** include:

- In-app streaming of movies.
- User-generated content such as reviews or forums in the initial release.

The application will provide:

- A **mobile-friendly interface** for both Android and iOS.
- **Fast and secure ticket booking** with minimal user effort.
- **Integration with third-party payment gateways** (e.g., Stripe, PayPal, Apple Pay, Google Pay).
- **Real-time updates** on seat availability and movie schedules.
- **Support for multiple languages and currencies.**

## 1.3 Definitions, Acronyms, and Abbreviations

- **SRS** – Software Requirements Specification
- **UI** – User Interface
- **API** – Application Programming Interface
- **OTP** – One-Time Password
- **DBMS** – Database Management System

- **Admin** – Cinema Administrator managing movie listings
- **PCI-DSS** – Payment Card Industry Data Security Standard
- **JWT** – JSON Web Token (used for secure authentication)

### 1.4 References

- IEEE Std 830-1998 – Recommended Practice for Software Requirements Specifications. (University of Alaska System)
- Payment Gateway API Documentation (Stripe Documentation).
- OAuth 2.0 and JWT authentication standards (Frontegg).
- Database design principles for high-performance applications (Red Gate Software).

### 1.5 Overview

- **Section 2** provides a general description of the product.
- **Section 3** details specific software requirements, including functional and non-functional requirements.
- **Section 4** lists external interfaces and system constraints.
- **Section 5** describes system features and use cases in more detail.

## 2. General Description

This section provides an overview of the product and its functionalities.

### 2.1 Product Perspective

The Movie Ticketing App is a standalone product but integrates with:

- **Third-party payment gateways** for processing transactions.
- **External movie databases** for automated scheduling updates.
- **Theater management systems** to synchronize movie listings and seat availability.
- **Push notification services** to send real-time updates to users.

### 2.2 Product Functions

The app will include the following functionalities:

- **User Registration/Login**: Secure authentication via email, phone number, or social media accounts.
- **Movie Browsing & Filtering**: Users can search for movies by title, genre, language, location, and time.
- **Seat Selection**: Interactive seat maps showing real-time availability and pricing tiers.
- **Online Payment Processing**: Secure transactions using credit/debit cards, digital wallets, and UPI.
- **E-Tickets and QR Code Generation**: Generate digital tickets for easy check-in and verification at theaters.
- **User Profiles**: Allow users to manage bookings, track purchase history, and set preferences.
- **Loyalty Rewards System**: Offer discounts, cashback, and reward points for frequent bookings.
- **Cinema Management Dashboard**: Admin panel for movie theater owners to manage schedules, ticket prices, and promotions.

- **Push Notifications & Email Alerts**: Send booking confirmations, reminders, and special offers.

## 2.3 User Characteristics

- **General Users**: Non-technical users who want a simple interface for booking movie tickets.
- **Cinema Administrators**: Users who need advanced features for managing schedules, pricing, and reports.
- **System Administrators**: IT personnel responsible for maintaining and securing the app's backend and database.

## 2.4 General Constraints

- The app must comply with local **data protection laws** (e.g., GDPR, CCPA).
- **Payment transactions** must be processed via PCI-DSS-compliant gateways.
- System downtime should not exceed **1% monthly** for reliability.
- Mobile app size should not exceed **50MB** to ensure faster downloads.
- The app should be **scalable** to support high traffic during peak hours (e.g., movie releases, and holidays).

## 2.5 Assumptions and Dependencies

- The app assumes **stable internet connectivity** for seamless operation.
- It relies on **third-party payment gateways** for transactions and refunds.
- It depends on **external APIs** to fetch real-time movie schedules and seat availability.
- The operating system requirements are **iOS 13+** and **Android 8+** for compatibility.
- Customer support will be available via **chatbots and live agents** during business hours.

# 3. Specific Requirements

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

- The UI needs to be easy to use
- Users should be able to access via a digital kiosk in the theater or online via a website

### 3.1.2 Hardware Interfaces

- UI needs to support both physical/printable tickets and digital tickets

### 3.1.3 Software Interfaces

- The system has to interface with the database of showtimes and tickets available
- It needs to be able to scrape online review websites to display reviews and critic quotes of movies

### 3.1.4 Communications Interfaces

- Tickets can be provided via email or in person at the box office

## 3.2 Functional Requirements

### 3.2.1 <Feature #1: Ticket Purchase and Management>

#### 3.2.1.1 Introduction

- This feature enables customers to purchase movie showing tickets and manage them. It ensures a seamless transition process while also adhering to security and business rules.

#### 3.2.1.2 Inputs

- User account information
- Showtime selection
- Seat selection
- Payment information—may take credit card, PayPal, or Bitcoin
- Number of Tickets—up to 20 per transaction
- Discount codes—student, veteran/military, and senior

#### 3.2.1.3 Processing

- Verify user account and login
- Check showtime and seat availability
- Apply discounts if applicable
- Process payment and currency conversion
- Generate unique and non-replicable tickets
- Update database with transaction details and remaining tickets
- Record transactions in daily logs

#### 3.2.1.4 Outputs

- Digital or printable tickets with unique identification
- Confirmation email with ticket details
- Updated user account information—purchase history, loyalty points

#### 3.2.1.5 Error Handling

- Invalid user account or login attempt
- Unavailable showtime or seats
- Invalid payment information
- Failed transactions
- Exceeding the maximum ticket limit
- Discount code errors—invalid or expired

### 3.2.2 <Feature #2: User Account Management>

#### 3.2.2.1 Introductions

- This feature allows customers to create and manage their accounts within the theater ticketing system
- Creating an account is optional but provides benefits such as storing personal and payment information, loyalty points, and purchase history
- Allows for ticket returns and exchanges

#### 3.2.2.2 Inputs

- **User Registration**
  - Personal information—name, contact information
  - Login credentials—email/username/phone number, password
- **Payment Information**
  - Credit card details
  - PayPal account information
  - Bitcoin wallet information
- **Account Login**
  - Email/username/phone number
  - Password
- **Account Updates**

    - Modified personal information
    - Updated payment information

#### 3.2.2.3 Processing

- Account creation: The system must validate the provided information
    - Checks for uniqueness of email/username
    - Encrypts password
    - Stores account information in the database
- Login Authentication: System compares and verifies that the provided credentials are the same as the stored credentials.

Only one account can concurrently log in to a user account.

- Payment Processing: System securely stores and manages payment information for a faster transaction
- Loyalty Point Calculation: System automatically awards loyalty points based on ticket purchases and updates the user's loyalty points balance
- Ticket Returns/Exchanges: System verifies if the ticket was purchased with a registered account and processes returns/exchanges according to the theater's policies

#### 3.2.2.4 Outputs

- Account Confirmation: A confirmation email/message is sent to the user given a successful account creation
- Login Success: User is granted access to their account dashboard
- Purchase History: A detailed record of past purchases is displayed
- Loyalty Points Balance: Current loyalty point balance is shown to user
- Updated Account Information: Confirmation that personal information or payment information has been successfully updated

#### 3.2.2.5 Error Handling

- Invalid Input—incorrect or missing information during registration or account updates
- Duplicate Account—email or username is already associated with an existing account
    - Incorrect Login Credentials
    - Payment Failure
- Session Management—if a user attempts to log in from a second device, the system terminates the previous session

## 3.3 Use Cases

### 3.3.1 Use Case #1

Buy Ticket

Actor(s): TicketBuyer

Flow of events:

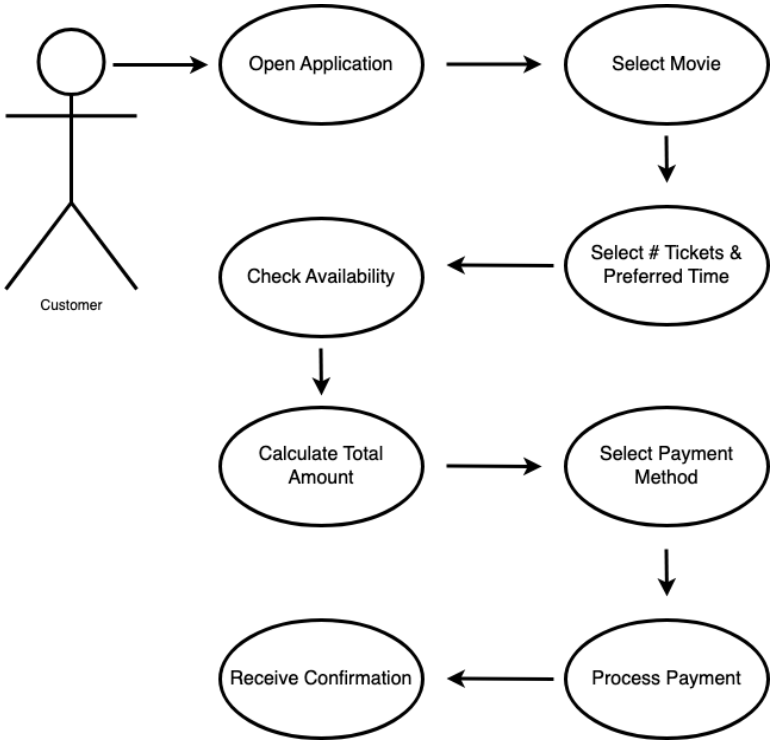
1. TicketBuyer opens the application or website on the device.
2. TicketBuyer chooses which movie they want to buy tickets for.
3. TicketBuyer selects how many tickets to buy and for which seats.
4. The amount of money owed is calculated, and the TicketBuyer selects their preferred payment option and gives the required amount.
5. The TicketBuyer receives a confirmation of purchase to show at the movie site.

Entry Conditions

1. TicketBuyer must be connected to the internet.

2. Computing requirements: supported browser if on the website

Use Case Diagram:




**3.3.2 Use Case #2**

Search for Movies

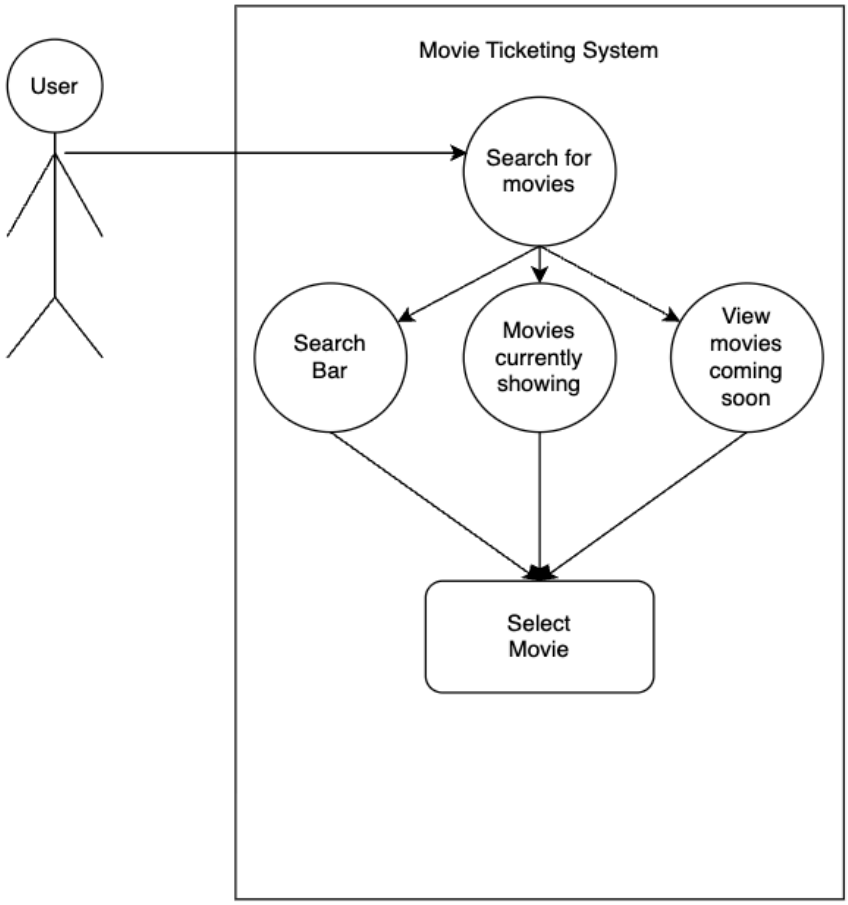
Actor(s): MovieSearcher

Flow of events:

1. MovieSearcher opens the application or website on the device.
2. MovieSearcher can click on the “Movies Currently Showing” or “Movies Coming Soon” page to redirect them to their desired movie list or choose to use the search button to filter movies with certain keywords. .
3. MovieSearcher scrolls through their options of movies until they decide to click on a specific movie or exit the page.
4. If a movie is chosen, MovieSearcher will be directed to ticket buying.

Entry Conditions

3. MovieSearcher must be connected to the internet.
4. Computing requirements: supported browser if on website



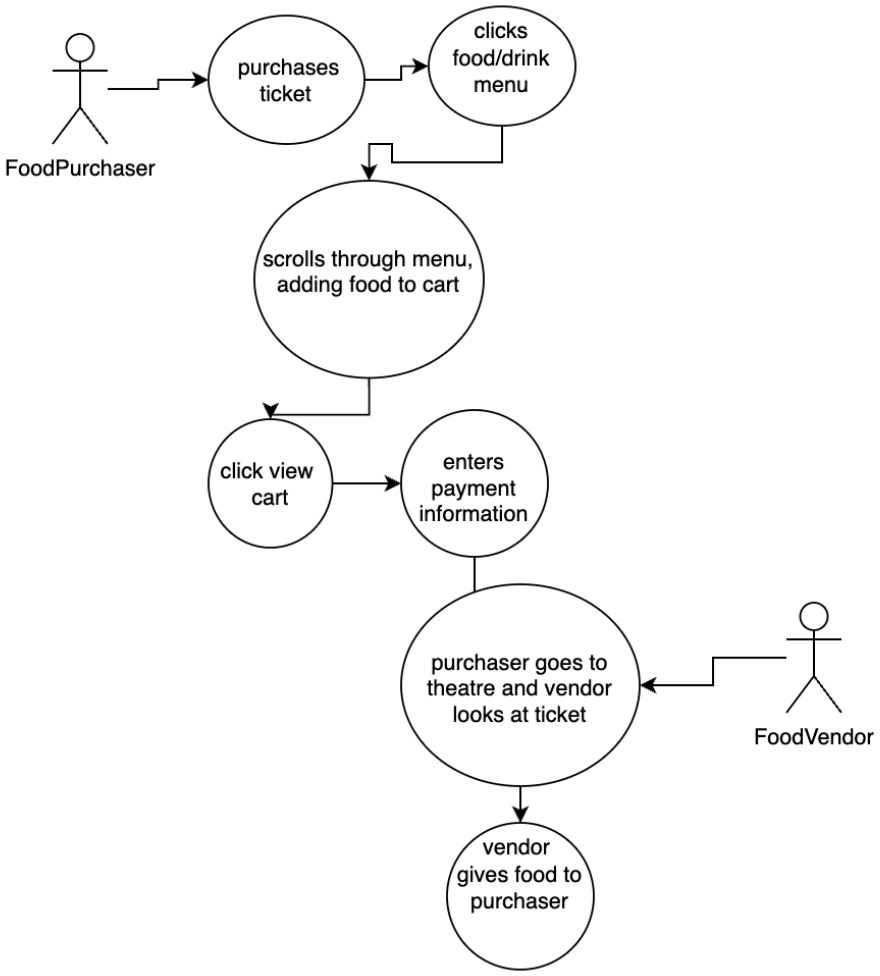

**3.3.3 Use Case #3**

Purchase Food and Drink

Actor(s): FoodPurchaser, FoodVendor

Flow of events:

1. FoodPurchaser successfully purchased a ticket to a movie.
2. FoodPurchaser has the option to click on the “Add Food or Drink” menu.
3. After clicking the menu, FoodPurchaser is redirected to the list of food and drinks they can add to their purchase.
4. FoodPurchaser looks through a list of options and may choose to add any items to cart.
5. FoodPurchaser clicks “View Cart” when finished ordering.
6. FoodPurchaser is directed to the payment page. When buying, FoodPurchaser is asked which movie ticket it is related to.
7. When FoodPurchasers arrive at the movie theatre, they show the confirmation of purchase to FoodVendor, and they receive their food quickly without needing to pay for it in the theatre.




## 3.4 Classes / Objects

### 3.4.1 <Class / Object #1-3>

| TicketBuyer |
| --- |
| Username: String<br>Payment: Payment Information<br>Login : Log-in Information |
| PurchaseTicket(ticket : movie)<br>PurchaseFood(food : String)<br>InputUsername(username: String)<br>InputPassword(password: String)<br>InputPaymentInfo(info : PaymentInformation) |

| Log-InInformation |
| --- |
| UserName: String<br>Password: String |
| ValidatePassword() |

| PaymentInformation |
| --- |
| Balance: Dollars = 0 |
| GiveMoney(amount : dollars) |

| MovieList |
| --- |
| movie1: movie<br>movie2: movie<br>movie3: movie |
| addMovie(Movie: movie)<br>deleteMovie(Movie: movie)<br>listMovies() |

| movie |
| --- |
| movieName: String<br>rating: String<br>ticketPrice: (amount : dollars = 0)<br>movieDesc: Sting |
| giveMovieInformation() |

## 3.5 Non-Functional Requirements

### 3.5.1 Performance

The product should be able to run on both a website browser and from a downloadable application for mobile devices. It will rely on a good connection to the internet to get access to information that is always updating. Application should be fully functioning with an internet speed at 2 Mbps.

### 3.5.2 Reliability
System will experience little to no crashes, even during times with increased usage and demand. When a purchase is stopped mid-transaction, it is guaranteed that the user will receive their money back and will be able to try again. There will be less than 6 minutes of downtime per year for the system.

### 3.5.3 Availability
The application is available to use and make purchases 24/7 at all times. It will still be functional even during the theatre's closing hours, allowing users to make purchases during the night or holidays.

### 3.5.4 Security
All data for accounts and purchases made through the application are secure and protected against data breaches. Payment information will never be given to unauthorized users and will be kept confidential. Security system follows the ISO 27001 standard.

### 3.5.5 Maintainability
General system failures will be addressed and fixed within 5 minutes with an 85% maintainability rate. Users are also able to personally submit help requests to customer service, which are closely monitored for 13 hours on weekdays from 8:00 AM to 9:00 PM. When emailed during customer service hours, the user will generally receive a reply within 10 minutes. However, if emailed outside of customer service hours, they will receive a reply promptly the next business day.

### 3.5.6 Portability
The application should be available to most devices released in the past 15 years. This includes but is not limited to a variety of smartphones, laptops, desktop computers, and tablets; regardless of branding. This does not include certain devices such as playstation consoles, flip-phones, and smartwatches.

## 3.6 Inverse Requirements
- Users cannot purchase a ticket without being already logged into an account.
- Users cannot purchase food or drink without being logged in and already buying a movie ticket.
- Users cannot purchase a movie ticket to an R-rated movie without verifying age as 17+ on their account.
- Users cannot purchase an alcoholic drink without verifying as 21+ on their account.
- Users cannot view or purchase tickets from a movie that has stopped showing.
- If payment is not successful, the purchase of the item does not go through.

## 3.7 Design Constraints
- The app will not be guaranteed to run on older devices. It will be constrained to mostly modern devices from the last 10 years.
- The devices needed to run the application need at least 1GB of RAM, an internet connection, and a 2.0 GHz clock speed.
- Movie tickets will only be available to participating theatres and the movies those theatres are offering.
- Age verification for alcoholic drinks and certain rated movies must fit with the local legal guidelines

- The application's interface must be kept simple to control the loading speeds (should all load within 2 seconds) for browsing and purchasing on mobile devices
- The overall budget for this project is minimal with only a few software developers hired, so the design should be restricted to necessary elements.
- The application must be compatible with screen readers, text magnification, captioned text and other accommodations for users with preferences and/or disabilities.
- Language support and currency exchange must be available for the regions where the application is available

### 3.8 Logical Database Requirements

- Uses an MySQL database to manage customer accounts and purchase details.
- The database must be able to store individual accounts that contain a username, password, payment information, and all their purchase history.
- Each purchase must be documented with a unique purchase ID and date.
- Each username created must be unique to one account.
- Database is backed up every 2 hours to prevent data loss.
- All data should be kept in an archive for 2 years.
- Properly documents and monitors audit trails.
- The system should be able to handle and smoothly operate a large amount of accounts stored in the database.

### 3.9 Other Requirements

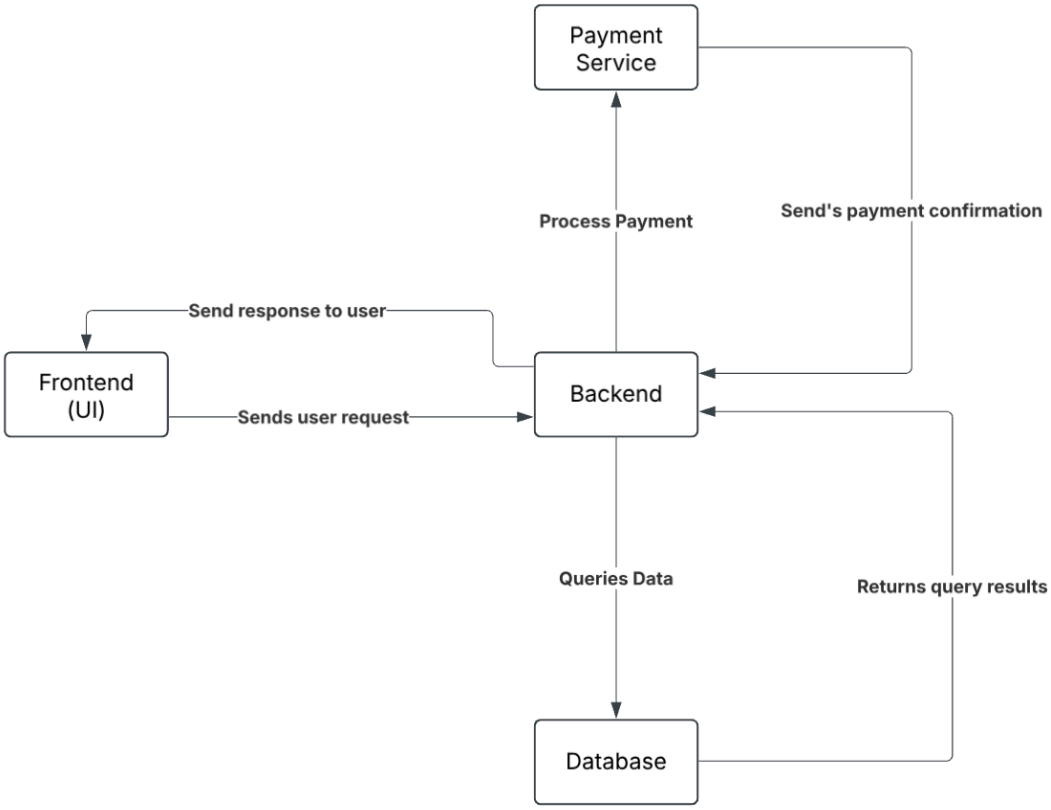
- The website's accessibility accommodations must comply with WCAG guidelines
- Both web-accessed and application user interfaces must be kept simple and user-friendly
- Application must well adapt to different screen sizes and formats
- Payments must comply with local taxes and other regulations
- The website must be well search engine optimized
- Purchases must follow proper PCI SSC security standards for card transactions and data.

## 4. Software Design Specification (SDS)

### 4.1 System Description

- The Movie Ticket Theater System is designed to facilitate online movie ticket and food purchases for customers. From a development perspective, the system follows a modular, object-oriented design to ensure maintainability and scalability.
- The user interface will communicate with the backend to access and change data in the database. The database will store user information such as accounts, purchase history, and other information. The payment services will receive requests with a payment gateway to complete and process transactions.

## 4.2 Software Architecture Overview




The movie ticketing system follows a client-server architecture with multiple components handling different responsibilities:

1. **Frontend (UI):**
   The primary interface through which users interact with the system.
   - Allows users to browse available movies, select showtimes, book tickets, and make payments.
   - Sends user requests to the backend and displays responses.
2. **Backend:**
   - Processes all incoming requests from the frontend.
   - Retrieves and updates data from the database.
   - Manages business logic such as seat availability, user authentication, and booking validation.
   - Sends booking confirmation and ticket details to the frontend.
3. **Database:**

   - Stores all relevant information, including user profiles, movie listings, showtimes, bookings, and payment transactions.

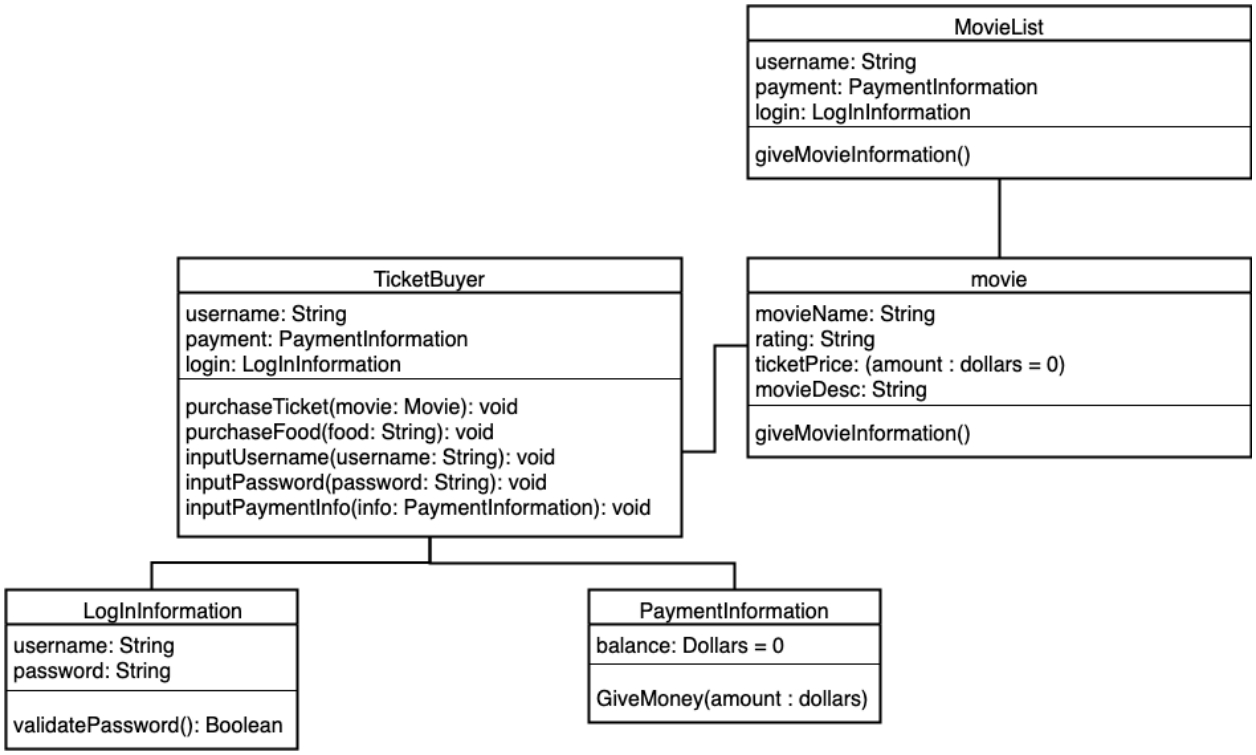
- Ensures data consistency and integrity.

4. **Payment Service:**
    - Handles secure transactions and payment processing.
    - Interfaces with third-party payment gateways if necessary.
    - Confirms successful or failed payments to the backend.

**Workflow Example:**

- The user selects a movie and requests available seats.
- The frontend sends the request to the backend.
- The backend queries the database for available seats and returns the data.
- The user selects a seat and proceeds with the booking.
- The backend initiates the payment process through the payment service.
- Once payment is confirmed, the backend updates the database and sends confirmation to the frontend.

## 4.3 UML Class Diagram & Descriptions




Class Descriptions:

- TicketBuyer: Class to represent the user who is using the application. Uses the classes Log-in Information and payment information objects as attributes. It has functions to put in information as string inputs as parameters (username, password, payment information) and also has a function to purchase items using a String.
- Log-in Information: Can be created as an instance object for TicketBuyer to store usernames and passwords as strings in the database. Its validate function is used to see if the imputed usernames and passwords strings as parameters are correct and match the strings already inside the database.

- Payment information: Can be created as an instance object for TicketBuyer to purchase items. It is able to transfer money from the linked payment account to the movie ticket site using the GiveMoney() function. It has Dollars as a double data type attribute.
- MovieList: Class to represent the entire list of movies currently available to view on the application. Its attributes are the movie objects it holds, and it has functions to add or remove the movies in the list, which both take a movie object as input parameters. There is also a function to display the entire list of movies available to the user.
- Movie: Can be created as objects for the MovieList class. Movie instances can contain the movie title, rating, and description as strings and price as a double data type. It has a function to list all its attributes to the user and its attributes can be accessed by the system.

# 5. Test Plan

## 5.1 Test Strategy Overview

The purpose of the test plan is to verify that the movie theater ticket software system is functional and to ensure that its components work as intended. The test plan will cover unit, functional, and system testing. The test cases are designed to validate the software's correctness, reliability, and usability.

## 5.2 Features to be Tested

## 5.3 Test Strategy

## 5.4 Test Cases

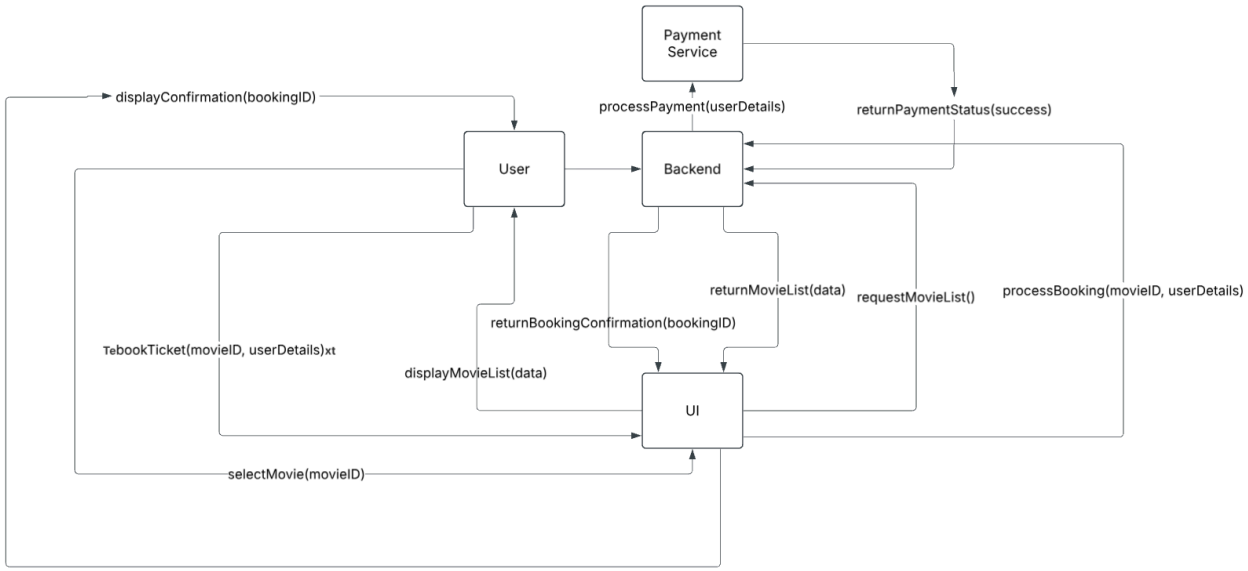
This can be found in this link

## 5.5 Coverage and Failure Scenarios

# 6. Development Plan & Timeline

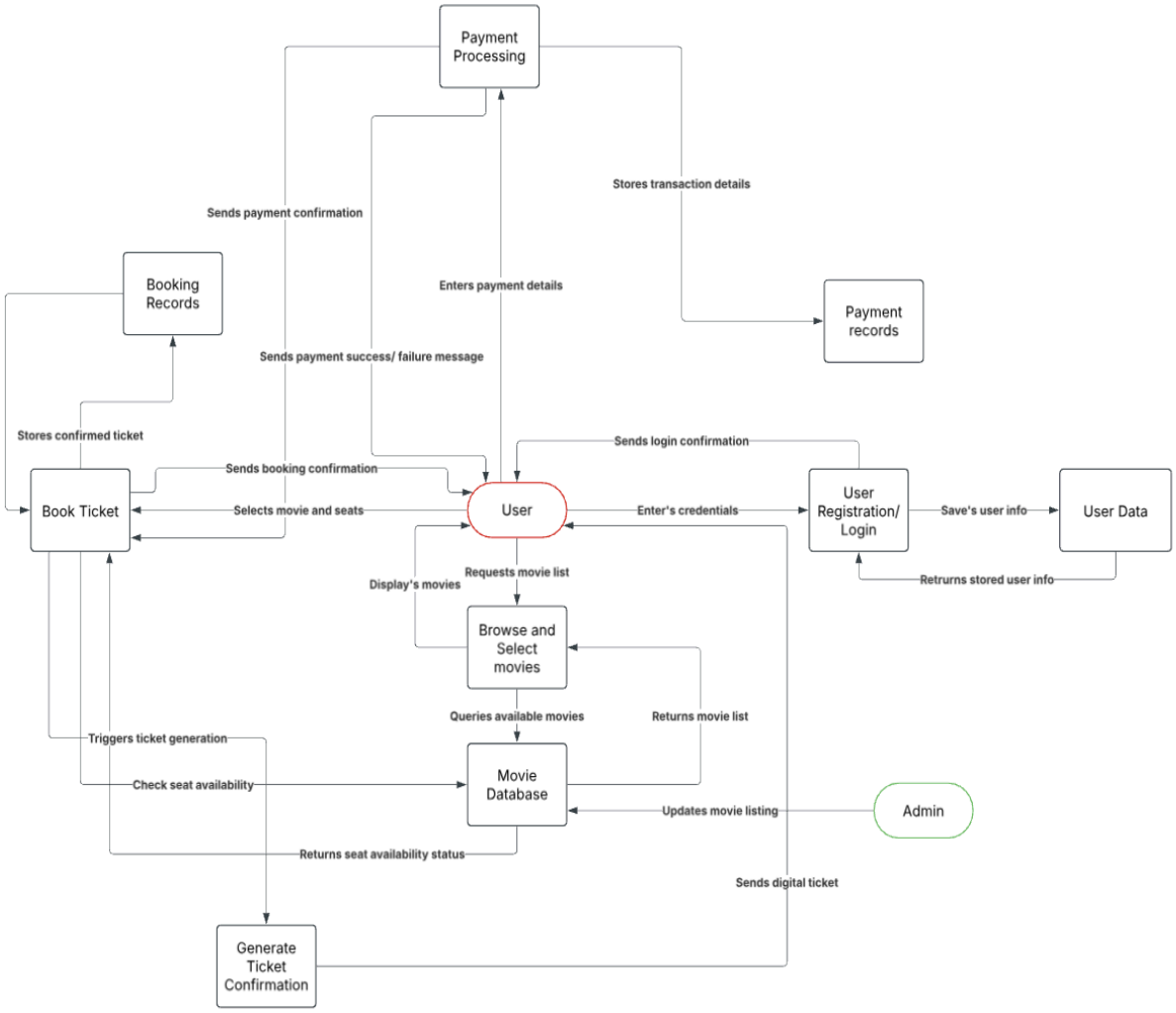
| Team Member | Task Breakdown | Estimated Time |
|---|---|---|
| Julie | UML Diagram<br>System Description | 1 hour |
| Aditya | Software Architecture overview<br>Data Flow Diagrams (DFD)<br>State Transition Diagrams (STD) | 2 hours |
| Malaika | UML Diagram Descriptions<br>Use Cases<br>Specific Requirements | 3 hours |

# 7. Analysis Models
## 7.1 Sequence Diagrams




The Sequence Diagram outlines the interactions between the User, UI, Backend, and Payment Service during the ticket booking process. The user selects a movie through the UI, which requests the movie list from the Backend. The Backend returns the movie list to the UI, which displays it to the user. Once the user chooses a movie and enters their details, the UI sends a booking request to the Backend. The Backend processes the booking and initiates payment processing with the Payment Service. After the payment is processed, the Payment Service sends the payment status back to the Backend, which then sends a booking confirmation to the UI. Finally, the UI displays the booking confirmation to the user. This diagram captures the flow of data and actions throughout the ticket booking process.

## 7.3 Data Flow Diagrams (DFD)




The Data Flow Diagram illustrates how data moves through the system, including user interactions and system processes.

**Entities, Processes, and Data Stores**

1. **User (External Entity):**
    - Provides input, such as registration details, movie selection, and payment information.
    - Receives output, such as available movies, booking confirmation, and payment status.
2. **Processes:**
   **User Registration:** Captures user details and stores them in the database.
    - **Movie Selection:** Fetches available movies and showtimes.
    - **Booking Ticket:** Handles seat selection and reservation.

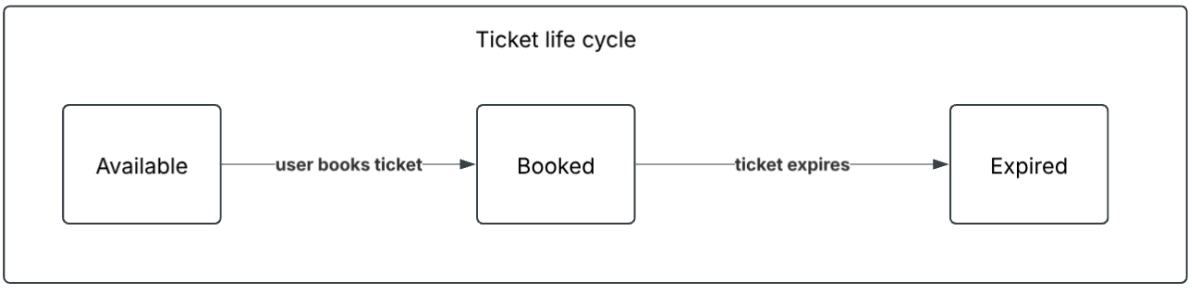
- **Payment Processing:** Validates and processes the payment.

3. **Data Stores:**
   **User Data:** Stores registered user details.
   - **Movie Data:** Stores movie listings, showtimes, and available seats.
   - **Ticket Data:** Maintains booking details and seat reservations.
   - **Payment Data:** Records payment transactions and statuses.

**Data Flow (Connections)**

- **User Registration → User Data:** Stores user details upon registration.
- **Movie Selection → Movie Data:** Retrieves and displays available movies.
- **Booking Ticket → Ticket Data:** Reserves the selected seat.
- **Payment Processing → Payment Data:** Stores transaction details and status.
- **Payment Data → Payment Processing:** Returns success/failure status to confirm or reject the booking.

## 7.2 State-Transition Diagrams (STD)




The State-Transition Diagram represents the different states a ticket can be in and the transitions between them.

**States:**

1. **Available:**
   - The default state where the ticket is not yet booked and can be reserved by any user.

2. **Booked:**
   - The ticket is reserved by a user after a successful booking.
   - Payment has been processed, and the booking details are recorded.
3. **Cancelled:**
   - The user cancels the booking before the showtime.
   - The seat returns to the available pool.
4. **Expired:**
   - The ticket automatically expires after the showtime has passed.
   - No longer available for use or modification.

**Transitions (Events That Change State):**

1. **Available → Booked:**
   - The user selects a seat and successfully books the ticket.
2. **Booked → Cancelled:**

    - The user cancels their booking before the showtime, making the seat available again.
3. **Booked → Expired:**
    - The movie showtime has passed, and the ticket is no longer valid.
4. **Expired → Available (Optional):**
    - If a system rule allows, expired tickets might be reset for rebooking.

# 8. Change Management Process

A team member can propose changes to the SRS by submitting a GitHub issue detailing the modification, its rationale, and its impact. Proposed changes will be reviewed through team discussions, and if approved by majority vote, they will be documented in the Change Log. Implementation will follow a structured process where changes are made in a separate Git branch, submitted via a pull request, and reviewed by at least one other team member before merging. The updated SRS will be pushed to the group's GitHub repository as a PDF or TXT file

# 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