Theater Ticketing System

Software Requirements Specification

Version <4.0>

10/10/2024

Group 3
Dominic Griffith, Kevin Callahan, Amira Blount, Duy Dao

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

Revision History

Date	Description	Author	Comments
9/26/24	Version <1.0>	Group 3	First Revision
10/10/24	Version <2.0>	Group 3	Second Revision
10/24/24	Version <3.0>	Group 3	Third Revision
11/7/24	Version <4.0>	Group 3	Fourth Revision

Document Approval

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

Signature	Printed Name	Title	Date
Group 3	Group 3	Software Eng.	9/19/2024
	Dr. Gus Hanna	Instructor, CS 250	

Table of Contents

1. Introduction

This Software Requirement Specification (SRS) document provides an overview of the requirements for developing an online Theater Ticketing System. The goal of this document is to gather and analyze the necessary information to design a system that efficiently manages theater operations. It will outline the system's scope, purpose and key functionalities.

1.1 Purpose

The purpose of this document is to assemble a collection of ideas that can define the online theater ticketing system and its requirements to the benefit of all users. This document will also aid us in understanding the products functional and non-functional requirements and constraints. Concepts will be outlined that may be developed at a later time, may be adjusted over time, or may be extracted any time in accordance with how the product will evolve. The purpose is to provide an illustration of the software product and its goals in regards to many conditions, especially the software and hardware requirements and constraints. The document will describe the target audience of the product and include the user interface and its functionality for the client and audience. This document will also be beneficial for all designers and software developers involved in the software delivery lifecycle process.

1.2 Scope

The scope of the Theatre Ticketing System is to create a web-based system that a customer will be able to use in order to browse movie showing times and to be able to purchase tickets and select seats for that movie. Upon purchase the system will send the user a confirmation email and a payment receipt. The customer will also be able to log into an account in order to save email, payment information (sensitive information will be stored as hashes, card information will be stored in a secure dedicated system that meets all PCI standards) as well as favorite genres, closest theater, reservations, and purchase history in order to create a more customized experience. This account will also have a points program in order to incentivise return customers.

1.3 Definitions, Acronyms, and Abbreviations

(1) User/Customer: The terms "User" and "Customer" are used interchangeably throughout this document, referring to any individual who interacts with the Online Theater Ticketing System (OTTS) for purposes such as purchasing tickets, reserving seats, or ordering admission.
(2) Graph: An ideal data type used to represent relationships between objects (nodes) where pairs of objects are connected by links (edges).
(3) API: Application Programming Interface - Allows OTTS to communicate with other systems.
(4) DBMS: Database Management System - The software used to manage the databases that store and organize all the data related to OTTS, including user information, movie schedules, and transaction records.

(5) HTTPS: HyperText Transfer Protocol Secure - Provide secure communication over the internet.
(6) UI: User Interface - The display on which the customer will interact with

1.4 References

- IEEE Std 830-1998: “IEEE Recommended Practice for Software Requirements Specifications”

1.5 Overview

This document is organized into several sections that cover the system’s scope, functionalities, and technical specifications. Section 1 introduced the purpose and scope of the system. Section 2 will provide a general description of the system, including its product perspective, main functions, user characteristics, constraints and assumptions. Section 3 focuses on specific requirements, including external interfaces, functional requirements, use cases, object models, and non-functional requirements. This section will also outline design constraints and database requirements. Section 4 presents analysis models, including sequence diagrams, data flow diagrams, and state-transition diagrams. Section 5 outlines the change management process.

2. General Description

The following section will go over the general requirements for the Movie Theatre Ticketing software. This will aid in overviewing and understanding the functions as well as other broad aspects the software requires.

2.1 Product Perspective

This product begins the movie theater process for all movie goers by having them select the movie, type of tickets (i.e. child, adult, senior, etc), type of theater (Standard, IMAX, Dolby) and seating reservation in a set sequential order ending with payment and confirmation of payment through number or email. The ticketing system also includes the option for ordering food and drink ahead of time. Users will be given multiple payment options be it card through many banking institutions, cash at the box office, PayPal, ApplePay.

2.2 Product Functions

Our Online Movie Theater Ticketing software is designed to heighten the movie-going experience by offering a range of user-friendly features. Customers can pre-order tickets and deal, allowing them to bypass long lines and head straight to their seats. The software provides a complete movie listings, complete with ticket prices, maturity ratings, trailers, and showtimes, enabling customers to make well-informed decisions. For added benefit, the system offers a loyalty program with a premium subscription monthly or yearly memberships that include exclusive perks such as discounts, early access to new releases, and special events. Additionally, the software combines a dynamic seating map that allows customers select and reserve their preferred seats in real-time. In the event of any assistance, provide a smooth and enjoyable experience for all users.

2.3 User Characteristics

Users of the system should be able to retrieve showing information with given date and time from the database. The system will support two types of user privileges, customers and employees. Customers will have access to customer functions and employees will have access to administrative functions.

- Customer Functions:
    - Browse available shows
    - View seating arrangements and select seats
    - Make reservations and purchase tickets
    - Cancel or modify existing reservations
    - Receive booking confirmation via email
    - Access and manage history and account information
- Administrative Functions:
    - Manage show schedules
        - Add or Remove shows
    - Update ticket pricing
    - Oversee seating arrangements
    - View and manage customer reservations
    - Generate sales reports

2.4 General Constraints

- Account Data Security - All sensitive data must be encrypted using a hash system and credit card information must be stored according to PCI standards
- Quality - It is expected that Software Quality Assurance is implemented during each stage of development and that the end product is usable with minimal errors
- User Support - It is expected that the website is able to run on modern smartphones (Apple and Android) web browsers as well as desktop browsers (Apple and Windows)
- Development - There are no specific requirements upon which language the website itself is built on, although the preferred language for the account and movie database is SQL.
- Legal - To avoid fines from the federal government all credit card payments taken must be PCI compliant. All trailer plays must also be counted in order to pay royalties to production companies.

2.5 Assumptions and Dependencies

There are many possibilities that can affect the customer experience due to external circumstances such as:

1. Users' payment type is not being supported on the system.
    1.1. How would we expand our payment system to not exclude our customer base?

2. User buys a drink, food, or merchandise item that is sold out when they arrive for their movie theater reservation
3. User arriving at the box office to pay for their reservation in cash but the line is too long and their reservation time is soon
    3.1. Can the theater accommodate a line just for customers reserving their tickets and paying in cash before the movie?
4. The website for the product saves customers information, including payment information, but isn't secure and has their customers data leaked
    4.1. How would we secure the website and make sure it stays secure to protect customers?
    4.2. How would we keep the company data secure? How would we recover?
5. Users should need no more than 15 minutes to complete their entire reservation
    5.1. Would the seat selection mark the users' selected seats as reserved to other users purchasing tickets simultaneously for the 15 minutes of the reservation or would the completed payment mark the seats as reserved?
    5.2. If there is a high volume of users purchasing similar seat reservations for a specific movie, how do we tell the second user the seats they selected are no longer available?

3. Specific Requirements

3.1 External Interface Requirements

The Theater Ticketing System interacts with various external components through user, software, hardware, and communication interfaces. These interfaces are designed to ensure smooth interaction between users and the system while maintaining compatibility with commonly used platforms.

3.1.1 User Interfaces

The website the ticketing system will be displayed on will be developed so that it is appealing to the eyes and easy to navigate on both mobile devices and desktop web browsers. There is no current plan for an application to be developed so all focus will be on ease of use and beautification of the webpage and so that customers can quickly choose which movie they want to watch. The user interfaces (UI) required will be as follows

- Initial Website UI:
  This page will show a list of currently showing movies. These movies will be scrolling down in an effect similar to old style film roll to bring attention and raise excitement in the customer. Theater selected should be near the scrolling list of movies. Current offers will be displayed near the top, if the customer is logged in their name will be displayed on the top left corner, if not a button that says "Login/Create Account" will be displayed instead.
- Account Login UI:
  This page will initially have one prompt, one for the customer's email. If the email given doesn't exist, the customer is prompted to create an account. If it does, the customer is

asked to enter a password. If the password does not match, an error is given and an email is sent to assist in resetting the password.

- Account Creation UI:
  This page will have boxes for the customer to input all information required (as listed in 3.3.1)
- Account UI:
  This page will display all account information. There will also be an option to modify the account information or update payment settings. On the bottom of the page you will be able to view all your points, reservations, and purchase history from this page.
- Movie Page UI
  When a movie is clicked on from the initial website, a new page is shown with that movie's poster, trailer, showtimes, and reviews. Showtimes can be clicked on to be brought to seat selection
- Seat Selection UI
  Customers are allowed to choose a seat from available seats. When a customer has selected a seat, they are automatically brought to checkout
- Checkout UI
  If logged in, the customer is prompted to use saved credit card information to confirm purchase. If not logged in, a secure field is displayed where customers can enter card information to confirm the purchase. After purchase is completed, an email is sent to the customer confirming the purchase and the receipt with the ticket.

3.1.2 Hardware Interfaces

The online ticketing system prioritizes convenience and accessibility for the customer base to make their movie going experience as easy and fun as possible. The convenience also allows the theater itself to work smoothly with very minimal human error. The hardware needed for both the user and the theater are as follows:

- ➔ *Device/Mobile Device* - A user can access the website through their smartphone or computer from any location and either browse movies or movie showings, purchase tickets and merchandise, sign up for the membership service, and find other information about the theater. A device is also necessary to display the ticket (barcode) the user is sent when approaching the theater attendant.

3.1.3 Software Interfaces

The Online Theater Ticketing System (OTTS) will interface with a specialized database system known as a Graph Database Management System (Graph DBMS). Unlike traditional databases, this system uses a graph structure where data is organized into nodes and edges. Each node represents an individual, such as a movie, and stores complete details including the title, user reviews, descriptions, weekly showtimes, and seating availability.

This graph-based approach is beneficial for OTTS, as it allows for quick and efficient statement of connections between different data points, like linking showtimes to limited theaters or finding related movie recommendations. The use of JSON format of data storage within the nodes ensures flexibility and comfort of access, supporting the dynamic nature of the ticketing

system. This setup is crucial for delivering a smooth and responsive user experience, enabling customers to access all necessary information quickly as they navigate the ticketing process.

3.1.4 Communications Interfaces

The primary communication protocol between the user and the system will be HTTP for secure data transmission. Data exchanged between the user and the server will be in JSON format to ensure efficient communication.

3.2 Functional Requirements

- Movie/Seat selection System
- Ticket Payment System
- Ticket Cancellation/Modification System
- Movie Review System
- Account Login/Creation/Modification System

3.2.1 Ticket Payment

3.2.1.1 Introduction

The product relies on maintaining a reliable and secure payment system that allows users to purchase tickets without error and the company to manage the payment through the company's processes.

3.2.1.2 Inputs

When the user goes through the movie reservation process on the website, the users last prompt will ask them for their contact information, that may be bypassed if user has a paid membership for the theater, including:

- Full name
- Email address
- Ticket Voucher (if applicable)

After the user completes the contact information they must complete the following billing information depending on their chosen payment method:

- Billing address
- Debit/Credit Card Number
- Debit/Credit Card Name
- Debit/Credit Card CVV/CVC

  Or:
- Gift Card Number
- Apple Pay/E-Wallet
- Cash

3.2.1.3 Processing

Once the user fills out all of the necessary fields, they can hit the "purchase" button that will be provided at the bottom. After the button is clicked the system must make sure the payment information is all valid and then securely charge the customer through the chosen form of payment.

3.2.1.4 Outputs

Once the payment has been confirmed and deemed valid by the banking system the user will be moved to a confirmation screen on the website that will tell the user their payment has been made with a confirmation ID and that they will receive a confirmation email and an email that will provide a link to the tickets and a barcode to provide the theater attendant. The page will also provide a receipt to the customer with the last four digits and card type they used to pay. It will also provide instructions on how to pick up food/drink/merchandise items if they purchased them through their reservation. If the user wants to pay in cash prior to the movie reservation it will also provide instructions on how to pay at the box office.

3.2.1.5 Error Handling

- If the user has incorrectly filled out the necessary fields and hits the purchase button the website will invalidate the attempt and automatically move the users display to where the incorrect/lack of information has been entered. The website will simultaneously prompt the user to fix or fill out the information before purchasing again.
- If the user attempts to complete the purchase but the purchase does not go through or declines, the website will refresh the page and tell the user that their original payment method declined and failed to go through. This will prompt the user to retry when they have fixed the issue on their end.
- If the error is an issue with the website or the company then the user will be provided the proper error code on the website and given the contact information for the theater's customer service.

3.2.2 Reviews and Ratings

3.2.2.1 Introduction

The Reviews and Ratings features are designed to enhance user engagement by allowing customers to express their opinions about movies they have watched or employee services . Users will have the ability to rate films on a scale of 1 to 5 stars and provide detailed written reviews. This not only aids other users in making informed decisions but also serves a sense of community within the platform. Additionally, it serves as a feedback mechanism for the theater to check customers' reception and preferences.

3.2.2.2 Inputs

- Movie Selection: Users will select a movie from the listing on the OTTS platform to review.
- Rating Submission: Users will provide a rating from 1 to 5 stars, with a visual interface (e.g., star icons) to help easy selection.
- Critic Information: The interface displays critic quotes and necessary movie details (such as cast, genre, and recap) to guide the user’s review process.

3.2.2.3 Processing

- Data Storage: Upon submission, the user’s rating and review will be linked to their user ID and associated with the selected movie in the database. This ensures that each review is traceable to the user who submitted it.
- Content Moderation: The system will automatically filter for inappropriate language or content using a combination of keyword checks and user reporting mechanisms. Reviews flagged as inappropriate will be temporarily held for review by moderators.

- Real-time Updates: Once processed, the new rating and review will be reflected in the movie's overall rating and feedback summary immediately, allowing other users to see the most current feedback.

3.2.2.4 Outputs

- Review Display: The system will present user reviews and ratings openly on the movie's detail page, with the option to sort reviews by most helpful, highest rating, or most recent.
- User Profile Integration: Users will be able to view their submitted reviews in their profiles, with an option to edit or delete them.
- Collected Ratings: The average rating of the movie will be updated to reflect new submissions, with a breakdown of ratings (e.g., percentage of 1-star to 5-star ratings) displayed to users for quick reference.

3.2.2.5 Error Handling

- Resubmission of Reviews: Users will be notified if their submitted review violates content guidelines due to inappropriate language. The system will allow them to revise and resubmit their review after addressing the highlighted issues.
- Retry Prompt: If there's an issue processing a review or rating, the system will prompt users to attempt resubmission. This message will provide insights into common errors and guide users through the resubmission process.
- Support Contact Options: Should users continue to experience issues with their reviews or ratings not being displayed, the system will offer easy access to customer support. This may include a dedicated support ticket system or a live chat feature for prompt assistance.

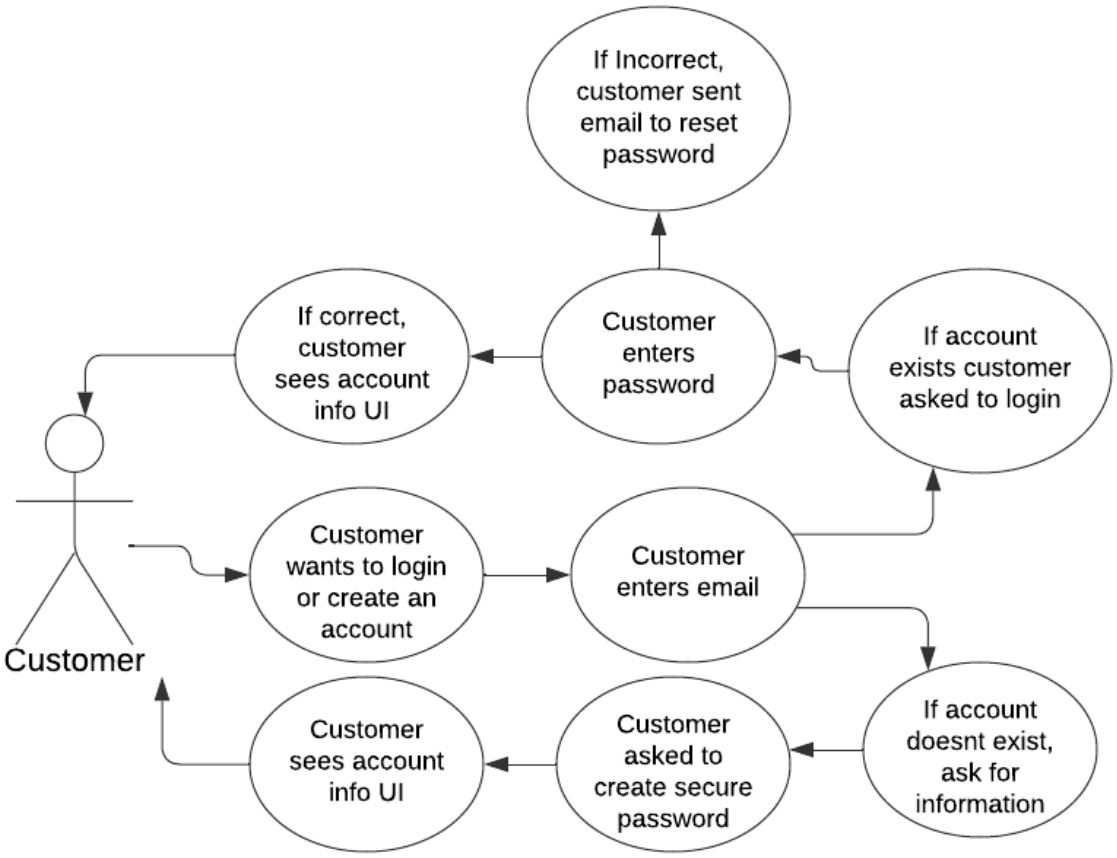
3.3 Use Cases

This section outlines the primary use cases for the Theater Ticketing System, describing how users interact with the system to perform key tasks.

3.3.1 Use Case #1 Account Creation/Login

- Actors: Customer
- Description: A customer creates an account or wants to log into a previously created account

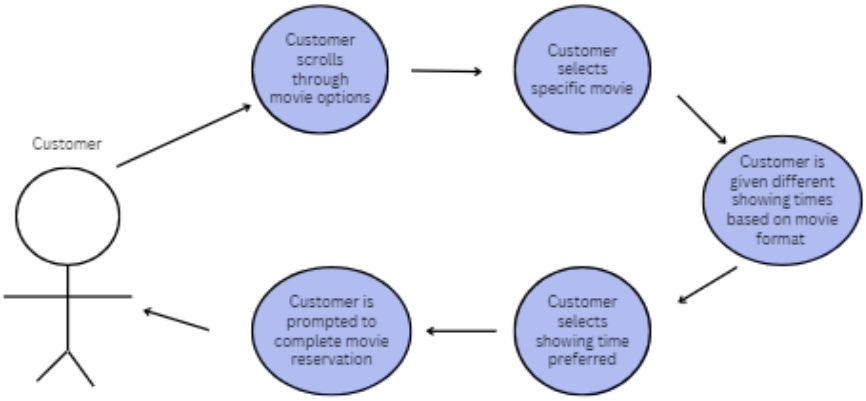
- Diagram:



- Pre-Conditions:
    - Customer must be on account creation/login page
    - If creating account, email must not exist in DB
    - If logging into account, account must already exist
- Post-Conditions:
    - Customers must see account information UI, with personal info, reservations, history, and favorites.
    - If an account is created, information updated and securely stored into db.

3.3.2 Use Case #2: Movie Selection

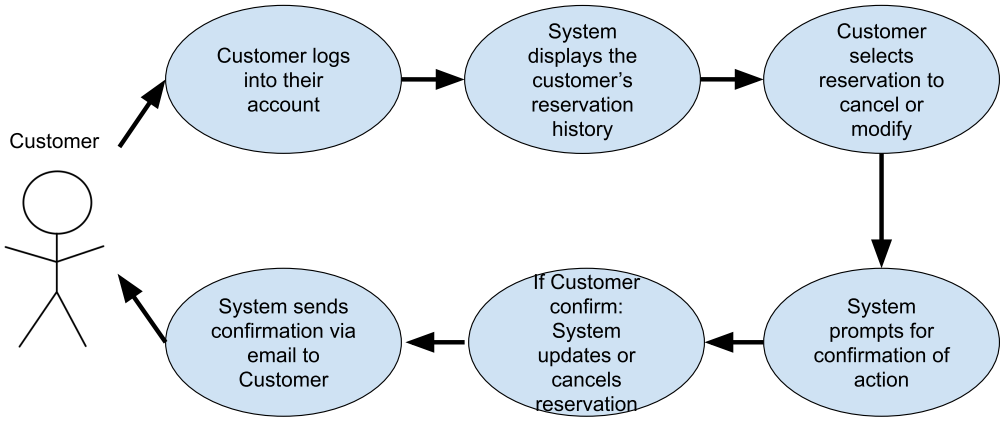
- Actors: Customer
- Description: A customer begins the movie selection process
- Diagram:



- Pre-Conditions:
  - Customer must be on main page of theater website
- Post-Conditions:
  - Movie showing and movie format is selected, enter reservation page
  - Customer simply was browsing for information, exits page

3.3.3 Use Case #3: Cancel or Modify Reservation

- Actors: Customer
- Description: A customer wants to cancel or modify an existing reservation
- Diagram:



- Pre-Conditions:
  - Customer has an account
  - Customer has an existing reservation
- Post-Conditions:
  - Customer's reservation is updated or canceled

…

3.4 Classes / Objects

3.4.1 Reservation

3.4.1.1 Attributes

- Ticket ID: unique identifier for the reservation
- Customer ID: identifies the linking customer who made the reservation
- Movie Title: title of the movie
- Rating: rating of the movie
- Price: cost of the ticket
- Seat Number: identifies the seat location
- Reservation Date: the date and time when the reservation was made

3.4.1.2 Functions

- Confirmation: Confirms and purchases ticket. Function relates to Use Case 3.3.2
- Modification: Allows the user to change reservation details, such as selected movie, seat number, and reservation date. Function relates to Use Case 3.3.3
- Cancelation: Processes a cancellation of the selected reservation. Function relates to Use Case 3.3.3

3.4.2 Account

3.4.2.1 Attributes

- Email
- Name
- Address - Two addresses stored if billing address is not the same
- Favorite Location - Automatically stored as closest location to customer, but can be modified if customer wants
- Favorite Genres
- Reservations/Purchase History
- Points Counter - Amount of points accrued by customer
- Payment Information - Secured meeting PCI standards

3.4.2.2 Functions

- Create Account
- Login
- View/Modify Account Information
- View/Modify Reservations/Purchase History
- View Points

3.5 Non-Functional Requirements

The following non-functional requirements of the online theater ticketing system will provide more clarity on the product's performance, reliability, availability, security, maintainability, and portability. The non-functional requirements will describe the systems capabilities and constraints necessary for the product to function at the system wide level. For added detail, each specification will be expressed in the proper measurable terms.

3.5.1 Performance

The Online Theater Ticketing System (OTTS) is designed for high performance and user satisfaction. It features an emotional user interface based on industry standards, ensuring efficient navigation and quick action completion. The system aims for load times under three seconds for key interfaces and supports flexibility to handle increased user traffic during peak periods. The responsive design guarantees consistent performance across different devices. Continuous performance monitoring will identify and address bottlenecks proactively. Additionally, the OTTS observe industry standard security protocols for data storage and transfer, enhancing user trust and system reliability.

3.5.2 Reliability

The system needs to demonstrate a high level of reliability to ensure smooth operation. This is especially true for peak hours. To deliver this, the Theater Ticketing System will maintain consistent uptime, with minimal service interruptions. Uptime will be maintained by deploying the system on a cloud infrastructure with auto-scaling capabilities. Additionally, regular backups and stress testing will be essential to ensure that the system can sustain peak demand and also be able to recover from failures without significant downtime or data loss.

3.5.3 Availability

The website will be available to anyone able to connect to the internet. Account creation will require an email and will be limited to those with a United States address. There will be no charge to use the service, as to encourage customers to use the online system.

3.5.4 Security

The security of the online theater ticketing system is of paramount importance to not only protect the company and its reputability but also its largely projected customer database. The system has to store and transmit highly sensitive and confidential information with respect to the specific security standards required.

- The sensitive information shall not be provided by guest customers until the final page where it is expected for users to spend most of their time filling out the information fields. When the credit/debit card number and CVV/CVC field is fully filled out by the user, the website will immediately hide the information except for the last four digits of the card with the special character “*” to visually secure customers from external factors that could view the information. When users then confirm the purchase the guest checkout information will be highly encrypted during the transaction.
- With users who have the theater's free account or paid membership, their data must already be stored within the system. Upon logging in with the correct username and password all of the customers previous information will appear at the checkout page or when the customer wants to view their account information. The only payment information that will be viewable is the card type (VISA, MASTERCARD, etc) and the last four digits of the card so that it may be recognizable to the user.
- If the user attempts to log in to their account and fills out the username or password information incorrectly then the website will prompt the user to try again with the correct username or password but also log the number of failed attempts. If the user has the correct username but incorrect password and has attempted more than 3 times, the

website will prompt the user to click the “forgot password” option and take the user through the steps on how to securely reset their password.

3.5.5 Maintainability

- Updates and Maintenance: The OTTS will support regular updates and maintenance to fix bugs, optimize performance, and adapt to new technologies. These updates will be implemented proactively to prevent potential issues from impacting users.
- Mean Preventative Maintenance Time: Scheduled maintenance will occur monthly, with a maximum duration of 30 minutes. Users will be notified of upcoming maintenance through on-site alerts, ensuring they are aware of any expected downtime.
- Mean Time to Restore the System (MTTRS): In the event of a system failure or technical issue, the target resolution time will be no longer than 10 minutes. Users will receive timely notifications regarding any delays and estimated time frames for resolution.
- Version Control: The development team will implement a Git version control system to manage collaborative efforts, ensuring consistency and reliability across the software. This will allow for multiple copies of the codebase, enhancing workflow efficiency without risking damage to the original source code.
- Documentation: Clear and complete documentation will be required for all code components, detailing the APIs, configuration settings, and system architecture. This ensures that all team members can easily understand and contribute to the object.
- Modularity: The software will be developed using a modular approach, enabling easy modifications during maintenance and updates without delicate the integrity of the source code. This design strategy will help efficient upgrades and enhancements in the future.

3.5.6 Portability

- Web Browser Support: The system will be fully functional across popular web browsers, including but not limited to Chrome, Firefox, Safari, and Edge. This ensures accessibility and usability for all users irrespective of preferred browsers.
- Minimal Host Dependencies: The system will use external libraries and frameworks will be chosen to ensure cross-platform performance and easy migration. This ensures that the software remains adaptable and reduces the time and effort required for future updates.
- Cross-platform compatibility: The system will be developed to ensure that it can run across different operating systems, including Windows, Linux, and macOS.
- Mobile devices: The system will also be optimized for various mobile devices. The application will feature a responsive design that adjusts to different screen sizes and orientations, providing a consistent and high-quality user experience. This ensure that the user can interact with the system effectively, regardless of the device they are using.

3.6 Inverse Requirements

*State any *useful* inverse requirements.*

3.7 Design Constraints

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

3.8 Logical Database Requirements

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

3.9 Other Requirements

Catchall section for any additional requirements.

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 the SRS's requirements.

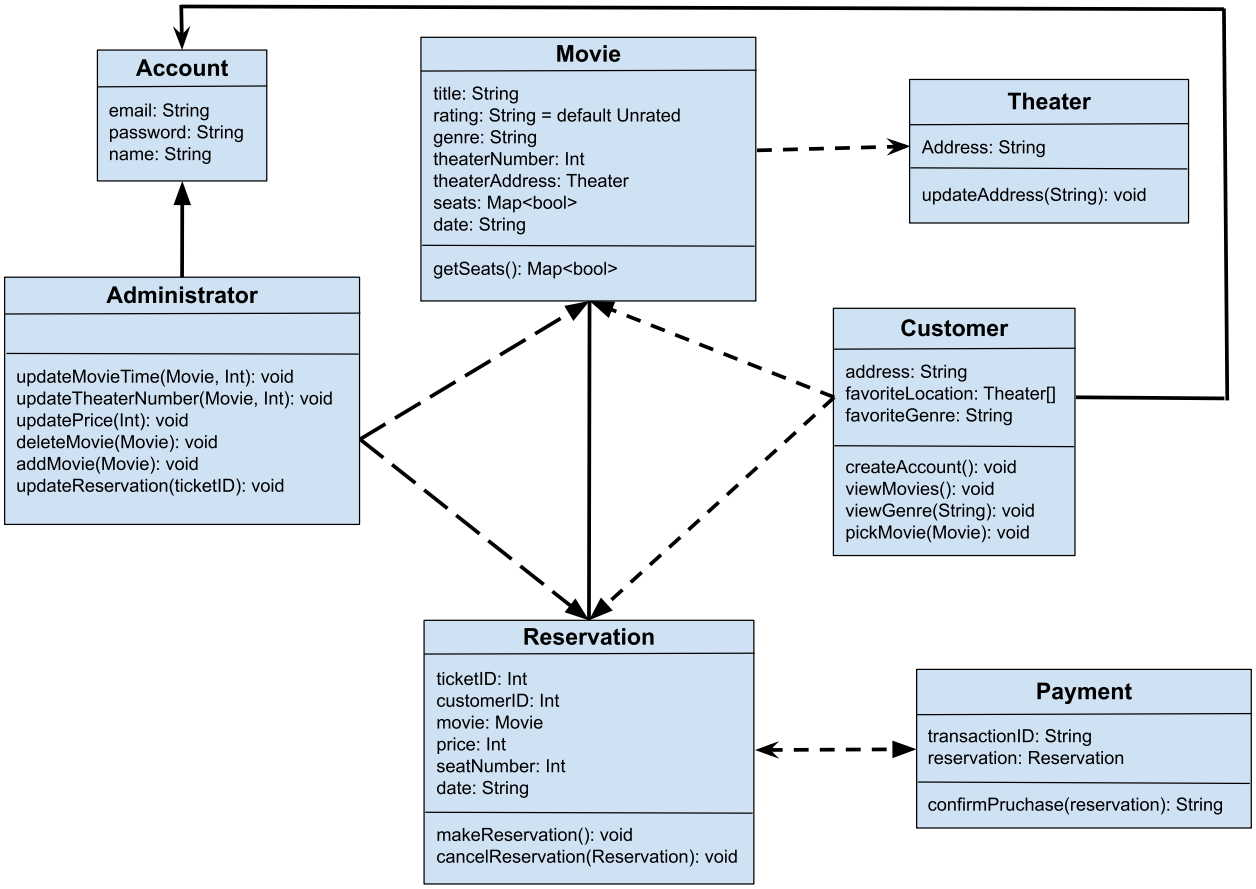
4.1 Sequence Diagrams

4.2 Data Flow Diagrams (DFD)

4.3 State-Transition Diagrams (STD)

4.4 UML

4.4.1 Class Diagram



4.4.2 Description

1) *Movie*
   a) *Class* - The Movie class represents the movies that are showing in the theater that the user can select from.
   b) *Attributes* - The attributes of the movie are very important when a user is selecting a movie so that they may gain more insight about movies they may have never heard of before. The user will be presented with the title (String) of the movie, the rating (String) of the movie which is automatically initialized to unrated until the movie is officially labeled by a rating, and the genre (String) of the movie which can help the user understand what the film is about. There's also the theater number (Int) which will tell the user where the movie is being shown within the movie theater. In case a customer online

is confused as to which theater they are trying to reserve a ticket for the theater address (Theater) is also listed for more clarification. When beginning the seat reservation process the user will be presented with a map of seats (Map<bool>), depending on the theater room, with gray seats (seat available = false) being seats that are already reserved and non-gray seats (seat available = true) being available. The last attribute is the date (String) of the movie the customer would like to reserve for.

c) *Operations* - The main operation of the movie class is to retrieve the map of seats (map of booleans) for the movie showing selected. When the request goes through the user will then be presented the seating options for their reservation. When the seat is selected it will swap the boolean values of the seat for 10 minutes; or if purchased indefinitely unless the reservation is canceled.

2) *Theater*

a) *Class* - The purpose of the Theater class is to store all theater locations.

b) *Attributes* - The only attribute within the Theater class is the address (String) of all of the theater locations.

c) *Operations* - The only operation is to update the theater address in case of an external circumstance such as the theater closing or being moved to a different location. This takes a string and returns nothing.

3) *Customer*

a) *Class* - The Customer class is about the user on the website and their details that are necessary for the website.

b) *Attributes* - The address (String) of the customer is needed to provide recommendations on the closest theater location in relation to the customer and their address. If the user is a frequent customer their favorite theater (Theater[]) will appear at the top of the suggested locations to watch a particular movie. If the customer has adjusted their settings they can also select a favorite genre (String) of film to help the website recommend movies catered towards the user.

c) *Operations* - The customer can trigger many operations that we hope will benefit the user as well as benefit the company. The most important operation is to allow a user to create an account for the company where they can manage their reservations and membership services. Along with creating an account the user can view and select movies out of all of the options on the website which will then provide them with more information. The user also has the capability of viewing a specific genre where the website can provide a selection of movies based upon that filter. The create account takes an address(string), favorite location(instance of theater class), and favorite genre(string) and returns nothing. The view movies takes nothing and displays a list of movie instances. The view genre takes a string and displays a list of movies in that genre. The pick movie takes an instance of a movie.

4) *Account*

a) *Class* - The Account class is connected to the Customer class because if the user has an account for the movie theater or wants to create a new account for the movie theater then this is where all of that information will be securely stored.

b) *Attributes* - The three attributes that we need from the user are the user's valid email address (String) that ends with "@[domain].[top-level domain]", the user's password (String) that must contain more than eight characters, a special character out of the

approved list of special characters, and at least one number 0-9, finally the user's name is important to validate the purchases made. The user must provide a first and last name that should match their ID so the movie attendant can verify who bought the ticket.

5) *Reservation*
   a) *Class* - The reservation class is important for the user and the website itself so that the customer may be able to select and reserve tickets for a movie without any complications.
   b) *Attributes* - The user and the company needs to have a ticket ID (Int) and customer ID (Int) that are unique to the customer and the ticket itself so that the website can manage the reservation. The reservation also provides the movie (Movie) the customer selected for their reservation as well as the price (Int) and the seat number (Int) for the movie the customer has booked. There is also the date (String) of the movie that the customer needs to know so they may show up at their scheduled reservation time.
   c) *Operations* - The main operations of this class is to allow the user to make the initial reservation, which can be altered within this same operation, and to cancel the reservation so the website can know that new seats have been made available due to the cancellation. Both make and cancel reservation functions take a user's input and updates the database, but returns nothing.

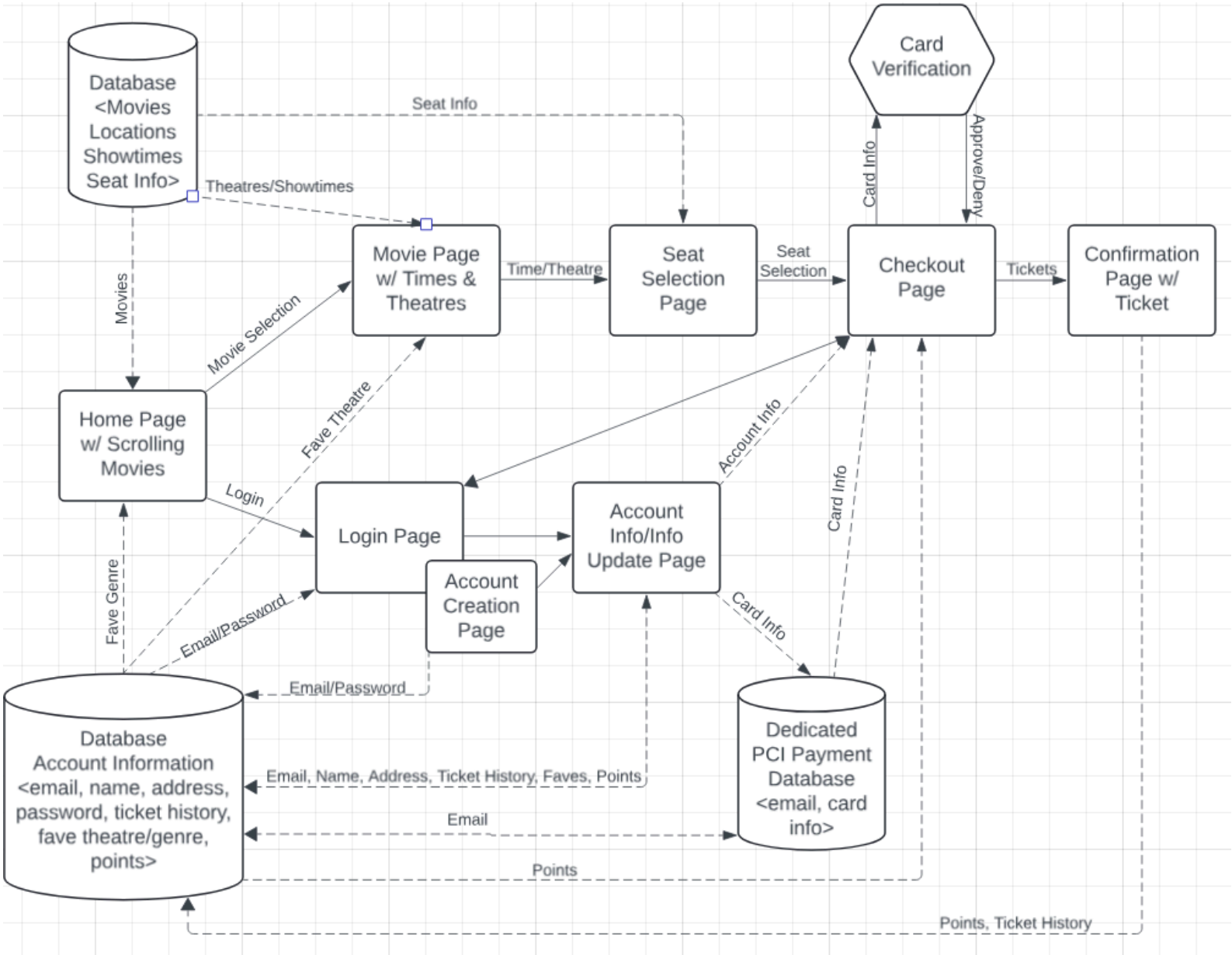
6) *Payment*
   a) *Class* - The Payment class is very important for the transactions and the financials of the customer and the company.
   b) *Attributes* - Each payment needs a transaction ID (String) it will appear to be a random collection of numbers and capitalized letters to the user but they are very important to keep track of in the system which will also be linked to a reservation (Reservation).
   c) *Operations* - After a valid payment is made by the customer and the system can confirm that, then the user will receive a confirmation of the purchase with the ticket ID and reservation that they have made. This takes an instance of a reservation and returns a string.

7) *Administrator*
   a) *Class* - The Administrator class is a very important operational class for the company and workers who need to make certain changes on the spot for a customer. Likely to be used by customer service attendants at the theater or employees with the authorization to access and change movies.
   b) *Operations* - An administrator would need to update a movies time or update a movies theater number if there were an external case where the movie would need to be moved to a different theater room such as high or low demand for a movie from users or there could be an issue in theater with a projector or sound system not working. These operations just need the administrator to provide the specific movie and integer to make their changes. The administrator can also update the price of certain movies or ticket types at their own discretion so long as they are authorized to do so. The administrator may need to add or delete different movies due to the movie ending their theatrical run or adding a new movie that will release within the next two weeks. Adding a movie will add all of the movie's information from the Movie class. If there is an employee providing customer service then they are authorized to update a customer's reservation in accordance to the situation they are dealing with so long as the customer can provide their ticket ID information. updateMovieTime takes a movie instance and an int, and returns nothing.

updateTheatreNumber takes a movie instance and an int, and returns nothing. updatePrice takes an int and returns nothing. deleteMovie takes a movie instance and returns nothing. addMovie takes a movie instance and returns nothing. updateReservation takes a ticketID and returns nothing.

4.5 Software Architecture

4.5.1 Diagram



4.5.2 Description

The Software Architecture Diagram of the Movie Theater Ticketing System provides a complete overview of the system's components and their interactions. It outlines the main modules of the

system, how they communicate, and the flow of data between them to ensure efficient booking, verification, and payment processing for customers.

1) Home page with Scrolling Movies
    a) *Component Overview* - This is the main landing page of the application. It displays a list of movies currently available at the theater in a scrolling format. Users can browse through movies(class) to make a selection.
    b) *Connector 1* - It connects to the Movie Database, which provides the list of available movies, showtimes(string), and theater information(class).
    c) *Connector 2* - When a user selects a movie, it connects to the Movie Page with Times & Theatres, allowing users to view showtimes and choose a theater.
2) Movie Page with Times and Theaters
    a) *Component Overview* - Once the user selects a movie from the Home Page, this component displays the available theaters and showtimes for that movie.
    b) *Connector 1* - It retrieves data from the Movie Database, including the theaters showing the movie and the corresponding showtimes.
    c) *Connector 2* - Once the user selects a showtime and theater, it passes this information to the Seat Selection Page.
3) Login Page
    a) *Component Overview* - The login page allows registered users to enter their credentials (email (string) and password(hashed string)) to log into their accounts.
    b) *Connector 1* - Connects to the Account Information Database, where user diplomas are stored. It verifies whether the entered character matches the stored data.
    c) *Connector 2* - Upon successful login, it may redirect to the Home Page, or to specific components like the Account Info Page.
    d) *Connector 3* - If the user does not have an account, it connects to the Account Creation Page.
4) Account Creation Page
    a) *Component Overview* - This component allows new users to create an account by providing personal information such as email, name, and password.
    b) *Connector* - Connects to the Account Information Database to store the newly created account details, ensuring the user can log in later.
5) Account Info/Info Update Page
    a) *Component Overview* - This page allows users to view or update their account details (e.g., name, email, and password).
    b) *Connector* - Connects to the Account Information Database to retrieve current user information, and allows updating details when necessary.
6) Seat Selection Page
    a) *Component Overview* - After selecting a showtime and theater, this page allows the user to choose their preferred seats.
    b) *Connector 1* - Connects to the Movie Database to retrieve seat availability for the selected showtime.
    c) *Connector 2* - Once seats are selected, it forwards this information to the Checkout Page for processing.

7) Checkout Page
    a) *Component Overview* - This is where the user enters their payment information to complete the ticket purchase after selecting seats.
    b) *Connector 1* - Connects to the Seat Selection Page to receive seat details.
    c) *Connector 2* - Connects to the Card Verification Module to verify the payment information.
    d) *Connector 3* - Once payment is confirmed, it passes ticket details to the Confirmation Page.
8) Card Verification Module
    a) *Component Overview* - This module is responsible for verifying the user's payment information. It ensures that the provided card details are valid.
    b) *Connector 1* - It communicates with the Dedicated PCI Payment Database, which securely stores and verifies card information.
    c) *Connector 2* - It sends payment confirmation or denial back to the Checkout Page, determining whether the transaction is approved.
9) Confirmation Page with Ticket
    a) *Component Overview* - After successful payment, this page provides the user with a confirmation of their ticket purchase. It displays the ticket details, including movie, time, theater, and seat information.
    b) *Connector* - Connects to the Checkout Page to retrieve and display the final ticket information.
10) Databases
    a) *Component Overview* - Stores all data related to movies, including movie titles, locations (theaters), showtimes, and seat availability.
    b) *Connector 1* - The Home Page retrieves movie list and theaters.
    c) *Connector 2* - The Movie Page with Timers & Theatres collects showtimes and theater details.
    d) *Connector 3* - The Seat Selection Page retrieves seat availability for the selected showtime.

4.6 Development Plan

4.6.1 Estimated Timeline

Date	Description	Author	Tasks
9/26/24	Version <1.0>	Dominic Griffith	Introduction, Overview, User Characteristics, External Interface Requirement (Description, Communication Interfaces), Use Case #3, Classes/Objects (Reservation), Non-Functional Requirements (Reliability, Portability)
9/26/24	Version <1.0>	Duy Dao	Definitions, Acronyms, and Abbreviations, Product Functions, Software Interface, Review and Ratings, Performance,

			Non-Functional Requirements (Maintainability)
9/26/24	Version <1.0>	Amira Blount	Purpose, Product Perspective, Assumptions and Dependencies, Hardware Interfaces, Ticket Payment, Use Case #2 Movie Selection, Non-Functional Requirements (Security)
9/26/24	Version <1.0>	Kevin Callahan	Scope, General Description, General Constraints, User Interfaces, Functional Requirements, Use Case #1, Account (Attributes/Functions), Availability
10/10/24	Version <2.0>	Dominic Griffith	UML Class Diagram
10/10/24	Version <2.0>	Duy Dao	Software Architecture (Description)
10/10/24	Version <2.0>	Amira Blount	UML Class Description
10/10/24	Version <2.0>	Kevin Callahan	Software Architecture Diagram
10/24/24	Version <3.0>	Dominic Griffith	Test Case Samples 1-3
10/24/24	Version <3.0>	Amira Blount	Test Case Samples 4-6
10/24/24	Version <3.0>	Duy Dao	Test Case Samples 9-10
10/24/24	Version <3.0>	Kevin Callahan	Test Case Samples 7-8
11/7/24	Version <4.0>	Dominic Griffith	Data Management (Diagram), Design Justification
11/7/24	Version <4.0>	Amira Blount	Data Management Strategy
11/7/24	Version <4.0>	Duy Dao	Data Management Design Decisions
11/7/24	Version <4.0>	Kevin Callahan	Software Architecture Diagram

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.

6. Test Plan

6.1 Test Case Samples

Test Case Samples									
TestCaseId	**Component**	**Priority**	**Description/Test Summary**	**Pre-requisites**	**Test Steps**	**Expected Result**	**Actual Result**	**Status**	**Test Executed By**
ReservationCreation	Reservation_Class	P1	Verify that a user can successfully create a reservation with valid ticket details, including ticket ID, customer ID, movie name, seat number, price and date.	The Movie class is implemented and a valid Movie object exist in database. Seats have been selected.	1. Instantiate a Reservation object with the follwing details: ticketID: 123, customerID: 456, movieName: "The Batman", Price: 15, seatNumber = 12, date: "2024-11-01". 2. Verify the Reservation Attribute.	The reservation is created successfully, and all attributes should match the input values.	Reservation attributes matched the input values.	Pass	Tester - Dominic
GenerateUniqueTicketID	TicketID_Generator	P2	Verify that the system generates a uique ticket ID for each reservation made.	The TicketIDGenerator class is implemeneted.	1. Initialize TicketID Generator. 2. Generate Multiple Ticket IDs by calling the method multiple times, and store the results in a list. 3. Verify that each subsequent ticket ID is exactly greater than the previous one in the list.	The ticket ID should be unique and incremented by 1 for each subsequent call.	All ticket IDs are unique and correctly increment by 1.	Pass	Tester - Dominic
ConfirmationEmail	Email_Notification_Module	P3	Verify that the system sends a confirmation email to the user upon a success ticket reservation.	A user has a valid account with a valid email address. User has successfully created a reservation. The email notification system is implemented and configured to send emails.	1. Initialize the Email Notification System. 2. Simulate a ticket reservation for a movie using valid user credentials. 3. Use mock function to simulate sending the confirmation email. 4. Verify the emails content: ticketID, customerName, movieName, date, and seatNumber. 5. Verify the email is sent to the correct user email address.	A confirmation email is sent with the correct details and recipient address.	The email was successfully sent with the correct details and address.	Pass	Tester - Dominic
SemanticSearch	Search_Bar	P4	Verify that when a user writes a search term into the Search Bar, semantic search results pulled from the updated movie database.	Home Page is launched.	1. Open the Theater Ticketing Systems Homepage. 2. After the homepage has been opened via browser, the search bar is displayed within the Navigation Bar at the top. 3. Type a current movie into the search bar. 4. Press enter on keyboard or click the search icon on the webpage.	Search results related to the User's searched term should be displayed even if the term is not a lexical match to the database.	Search results related to the User's search term was displayed.	Pass	Tester - Amira
AccountCreation	Account_Management	P5	Verify that the system will create and store a new and unique account created by the user that can be accessed thereafter.	Login/Sign-up page is launched on the Theater website.	1. Open the Theater Ticketing System's Homepage. 2. Click "Login" or "Sign-Up" icons in top right corner 3. User fills out all prompts with their unique and specific information. 4. Agree to the terms and conditions then click "join now"	System should create and securely store a unique account ID for the user to access given the information provided.	A unique account ID was created for the user and stored all of users important data with the ability to log on to the account from any device.	Pass	Tester - Amira
TransactionVerification	Transaction_ID	P6	Verify that the system can determine if a payment has gone through on both the user's and system's end so that a ticket reservation can be completed financially.	Payment Page must have all required prompts filled out.	1. User must select a movie showing must select desired seats and click "continue". 2. User must fill out all payment information that is required. 3. User must click "purchase" at the bottom right corner.	System should verify the purchase with the user's banking institution and relay the verification to the reservation class to complete the purchase.	System verified the purchase by the user and completed the transaction, adding a unique transaction ID to the purchase.	Pass	Tester - Amira
LoginVerification	Login_System	P7	Verify that the login page will accept a matching username and password and will deny login if password is entered incorrectly.	Login page is launched on theater website.	1. Open the Theater Ticketing System's Homepage. 2. Click "Login" icon in top right corner. 3. Fill out fields with a correct user and incorrect password, attempt login. 4. If allowed into account with incorrect password, test fails. If entry is denied, continue test 5. Fill out fields with a correct user and correct password, attempt login. 6. Verify correct user is logged in.	Expected to be denied entry on first login (incorrect password) attempt. After second login attempt (correct password), the webpage is now the account info/update page with the correct user entered.	Denied entry on login with incorrect password, account page displayed on login attempt with correct password.	Pass	Tester - Kevin
SeatAvailability	Seat_Availability_Module	P8	Verify that a system correctly makes a seat unavailable when selected by a customer on the seat selection page. Also verify that if the ticket was not purchased, the seat will become available again after 10 minutes of user inactivity, and that it will not become available if purchased.	Location, movie and showtime have been selected, and the seat selection page is launched.	1. On seat selection page, select a seat and close this browser. 2. On a new browser, select the same location, movie, and time as first browser. 3. Verify seat previously selected is unavailable. 4. Wait 10 minutes. 5. Navigate again to the same seat selection page. 6. Verify seat is now available. 7. Select seat, and complete checkout. 8. Wait 10 minutes. 9. Navigate again to the same seat selection page. 10. Verify seat is still unavailable.	After selecting a seat on the first browser, seat will become unavailable on a second browser, until 10 minutes have passed, then the seat will be available again. After checkout, seat will stay unavailable.	Seat becomes unavailable upon selection, 10 minutes later seat becomes available again. Seat stays unavailable after checkout.	Pass	Tester - Kevin
SystemLoad	System_Performance_Module	P9	Verify that the system can handle a high user load of 1000+ concurrent users without performance reduction.	System monitoring tools and user simulation software are available.	1. Launch the system and ensure all components are working. 2. Replicate 1000+ users booking tickets at the same time. 3. Monitor the system's response time and look for crashes.	The system maintains a response time of under 5 seconds and does not crash.	The system successfully handled 1000+ concurrent users with an average response time of 4.5 seconds and no crashes.	Pass	Tester - Duy
SystemBackup	Data_Backup_Module	P10	Verify the principle of data after a system backup and restoration.	Backup system is operational, and backup procedures are in place.	1. Perform a full backup of the system's database. 2. Simulate a system failure by shutting down the database. 3. Restore the system from the backup. 4. Verify that all user data (e.g., bookings, transactions) is restored precisely.	All data is restored without loss or corruption.	The system backup and restoration were completed successfully with all data restored perfect and no corruption.	Pass	Tester - Duy

6.2 Test Case Verification

1) **TestCaseID: ReservationCreation**
    a) Description: This test case ensures the theater ticketing system successfully creates reservations with valid ticket details. It will verify that the user can create a reservation for a movie by using details such as movie name, seat number, date, price, customer ID, and ticket ID. The system should store all the reservation details and allow users to retrieve or modify them if needed.
    b) Test Sets/Vectors: This test is designed to execute the reservation process. One test set includes creating a reservation for a single movie with valid ticket details. Another test set involves making a reservation for multiple movies. These tests also include validating that all details are stored correctly. Different scenarios will be covered, such as reserving when a showing is fully booked or selecting seats that are already reserved.
    c) Targeted Features: This test covers multiple targeted features of the reservation system, most importantly the ability to create and manage reservations. The reservation system will serve as a mediator between the seat selection page and the checkout page. It will ensure that valid reservations are passed along to the checkout process. The test also verifies that the system handles invalid inputs and edge cases appropriately.
2) **TestCaseID: GenerateUniqueTicketID**
    a) Description: This test ensures that the system generates unique and correctly incremented ticket IDs for each reservation made. It verifies the functionality of the TicketIDGenerator class, ensuring that ticket IDs are generated sequentially and incremented by 1 for each subsequent reservation. The test will confirm that each ticket ID is unique, correctly ordered, and no duplicates are created.
    b) Test Sets/Vectors: This test is designed to validate the uniqueness and sequential order of ticket IDs generated by the system. One test set involves generating ticket IDs for multiple reservations in sequence, ensuring that each ticket ID is unique and incremented by 1. Another test set includes stress testing by generating a large number of ticket IDs to confirm the system's ability to handle high-volume requests. Additional scenarios include simulating concurrent reservations to verify that no duplicate ticket IDs are produced under simultaneous load conditions.
    c) Targeted Features: This unit test focuses on ensuring the core functionality of the ticket generation process works as intended. It targets the essential feature of the system by guaranteeing that every reservation is assigned a unique, incremented ticket ID. This is crucial for tracking and identifying reservations. By focusing on the class in isolation, this test ensures the integrity and reliability of this fundamental function.
3) **TestCaseID: ConfirmationEmail**
    a) Description: This test verifies the functionality of of the email notification module by confirming that a confirmation email is sent to the user after successfully reserving a ticket. It ensures that the system correctly processes the reservation,

triggering the email notification process and delivering the email to the user's valid email address.

b) Test Sets/Vectors: This test is designed to ensure that the module functions correctly. One test set involves simulating a reservation for a movie using valid user credentials and verifying that the confirmation email is sent to the user's registered email address. Another test set includes checking the content of the email to ensure it accurately matches the reservation details, such as ticket ID, customer name, movie name, date, and seat number. Additional scenarios may include testing the system's response to invalid user accounts or invalid email configurations to confirm that no email is sent in such cases.

c) Targeted Features: The test covers multiple targeted features of the email notification module. It ensures that confirmation emails are sent accurately upon successful reservations, containing essential details. Additionally, it verifies that the emails are sent to the correct recipient address and that the system handles any errors appropriately.

4) TestCaseID: SemanticSearch

a) Description: This test case is meant to verify one of the important functionalities for the website when a user needs to find a specific movie on the theaters website. The test ensures the system returns search results related to the user's input as best as possible even if the user's prompt has human error. The results returned should be the most updated version of the movie database within the system.

b) Test Sets/Vectors: One of the main test sets is using the search function in a way that includes human error such as spelling mistakes, adding extra characters like spaces or commas, and using synonyms for certain words that a user may not have remembered. This test set ensures that the search function is able to still recognize what the user may have been looking for with as much accuracy given the user's prompt. Another test set is the using the filtered search function where a user can search by either genre, MPA ratings, specific date for showings, and many more filters.

c) Targeted Features: The search module targeted feature is the movie theater's website. Semantic search is an important function for a user's ability to browse a website and find any specific information they may be looking for. The search works with the movie module and being able to produce movie results that are related to the search, the genre the user may prefer, along with which theater the movie results are showing.

5) TestCaseID: AccountCreation

a) Description: This test ensures that all users are allowed to not only log in to their account with the theater but also sign up to the benefit of both the user and theater. The test should result in unique accounts being created and stored securely in the system to help promote more traffic through the theater itself. This test case is necessary because account creation is a required function for the theater website.

b) Test Sets/Vectors: The first approach taken to testing this test case is signing up for an account with the theater and filling out the required prompts with "John

Doe" information to then click on the "sign up" button and make sure the system creates and stores the tester accounts. After creating tester accounts to verify the creation of the accounts we then had to test the uniqueness of the account. The next test set was attempting to create accounts with the exact same "John Doe" information and clicking "sign up" with the expectation being the system rejecting the account being created again.

c) Targeted Features: This test allows users to create accounts for the theater to save all personal information and preferences securely. The main feature provides security of all information provided for the benefit of easy access to users when utilizing the website. Another feature that is prioritized is a unique account is created for every new user and stored into the systems account module. This test verifies correct integration between the customer and account modules.

6) TestCaseID: TransactionVerification

a) Description: This test must verify that a user has sufficient funds and has the funds to pay for the reservation. When the user clicks "purchase" the system must begin the transaction with the user's banking information and move the funds into the theater's financials. Once the transaction is verified, purchase goes through, the system must then create a unique transaction ID for the purchase.

b) Test Sets/Vectors: The first test set with this test case was to ensure the system was able to pull the correct amount of money for each purchase out of the users banking institutions. After verifying the first test set it is imperative to test the security of each transaction. Once both of those very important test sets are verified then we must test the transaction IDs provided by the system and determine if the transaction IDs are unique to every transaction and can allow any user as well as the administrators to follow any transaction that has been verified.

c) Targeted Features: This test focuses on the payment feature within the system to complete transactions on the users end and also on the systems end. Following the verification of payment the system needs to produce a transaction ID along with the reservation in order for each unique transaction to include the ability to track each payment. This test works closely with the reservation module and those features along with the module to finalize the user's reservation process.

7) TestCaseID: LoginVerification

a) Description: This test will assess the login systems ability to allow or deny a user access to an account. It will ensure that when given an existing username with an incorrect password access is denied, and when given a correct username and password access is granted.

b) Test Sets/Vectors: The tester will first test to see if giving a username (that exists) with an incorrect password will deny entry into the account. If the tester is allowed entry with an incorrect password, the test fails at this point. If denied entry, the test continues. The tester will then see if giving a correct password with the username allows them into the account. If denied entry, the test fails. If allowed access to the correct account's page, the test passes.

c) Targeted Features: The main targeted feature of the LoginVerification test is security. We must ensure that someone who gives the incorrect password to a username is not granted access to that account. A second feature is access to the account when the correct password is given.

8) TestCaseID: SeatAvailability

a) Description: This functional test will ensure that the system correctly marks a seat as unavailable once selected by a user. If the user has not checked out 10 minutes after selecting a seat, the seat should become available again. If the user has checked out, the seat should stay unavailable.

b) Test Sets/Vectors: On the first browser, after selecting a location, movie and showtime, the tester will select a seat on the seat selection page. The tester will exit out of the first browser. On a second browser, the tester will navigate to the seat selection page of the same showtime, and check to ensure that the previously selected seat is unavailable. The tester will then wait 10 minutes, and check again to ensure the seat is now available. The tester will now checkout with the selected seat, wait 10 more minutes, and ensure the seat stays unavailable.

c) Targeted Features: Through this test, we are ensuring that no two users can select the same seat when purchasing a ticket. We are also ensuring that the seat becomes available again after the user does not checkout, and that it stays unavailable after the ticket is purchased.

9) TestCaseID: SystemLoad

a) Description: This test evaluates the system's ability to handle 1000+ concurrent users without performance reduction. It determines how well the system manages high traffic while maintaining core functions like browsing, ticket booking, and payment processing. The test monitors response times, system resource usage, and overall stability during peak load scenarios. The goal is to ensure that the system can support heavy user demand without crashing or slowing down.

b) Test Sets/Vectors: This will affect 1000+ concurrent users accessing the system that would build a realistic load environment. The system is closely monitored and the users will perform a variety of operations. CPU usage, memory usage and network bandwidth will be monitored and everywhere look for any bottleneck that could cause slow down. The system will also be evaluated under steady state conditions as well as sudden increase in activity to evaluate the system's flexibility and scalability.

c) Targeted Features: The purpose of this test is to measure the system's dependency on the system to perform high user loads in a stable manner, fast response times, and reliable transaction processing. The exercise tests the system's ability to be scaled and sturdy at peak loads, reviewing performance of key modules, such as ticket booking, payment processing and seat availability.

10) TestCaseID: SystemBackup

a) Description: This test ensures that critical data can be backed up and restored successfully during unexpected failure, server crash, for example. It is like simulating a full system failing and to see how the restoration process will be

valid, all users data, bookings and transactions should be there. The system is also tested because the test verifies that the system can proceed in normal mode from a restore without data corruption.

b) Test Sets/Vectors: The test begins by backing up the entire system's data, capturing every users' bookings and transactions. Once that is done, a simulated system failure – server crash, or data corruption – will be introduced. Once the previously created backup is used, the process of restoration will start and we will make sure that data restored is in good shape. This means comparing the restored data with the original backup, to see if no information was lost or changed.

c) Targeted Features: This test is going to be aimed at data flexibility after restoration. Evaluation of data integrity handling after restoration as well as being able to deal with corrupted or incomplete data recovery is a key feature. To this end, we aim to focus on these areas, so that user information is secured in a manner that lets it be restored free of error. The thing that this thorough examination will do is will increase the whole system's data management functions' reliability.

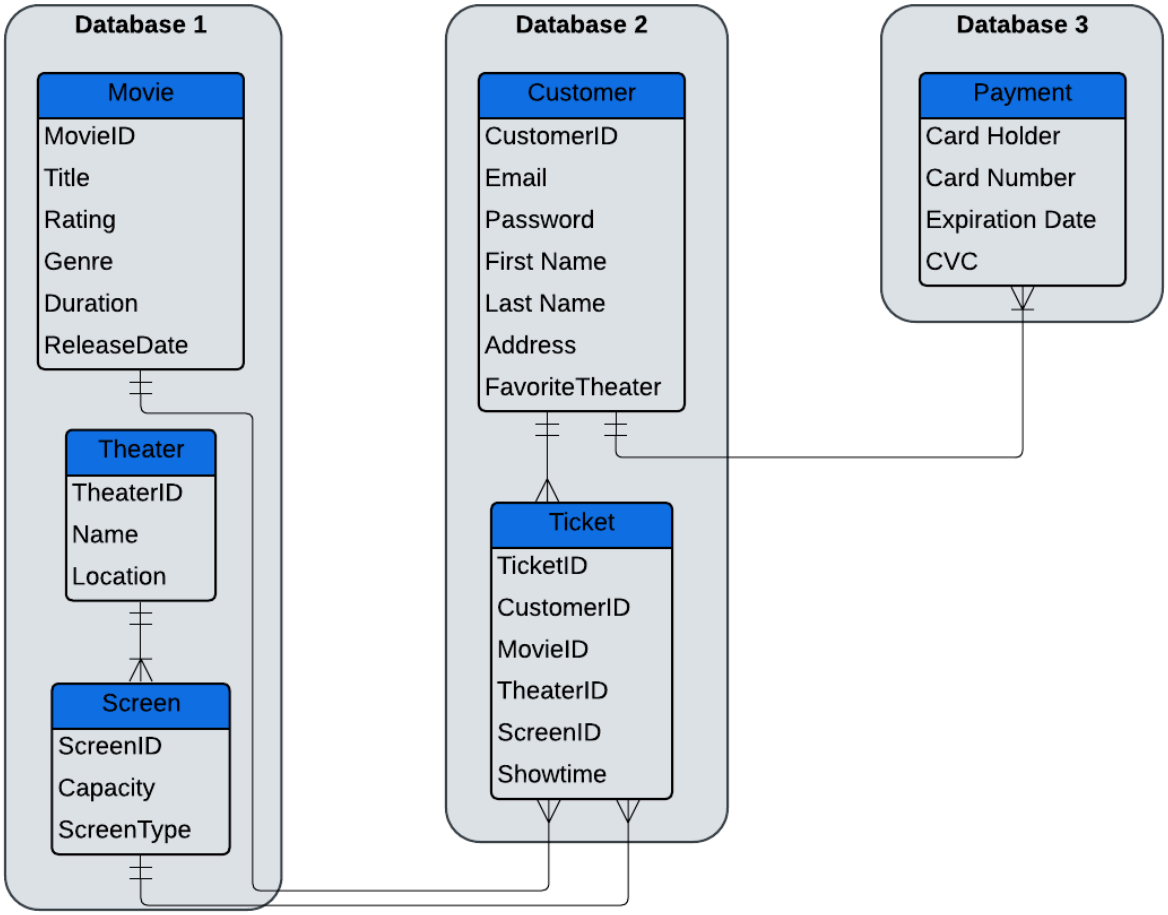
6.3 Github Link

https://github.com/dominic-griffith/CS250_Group_3

7. Data Management Strategy

Here you should specify your data management strategy, SQL or non. (Hint: diagrams always help) You should also discuss your design decisions: how many databases you chose and why, how you split up the data logically, possible alternatives you could have used (both in technology and organization of data), and what the tradeoffs are between your choice and alternatives.

7.1 Diagram



Data Dictionary
TicketID: unique identifier for ticket
CustomerID: unique identifier for customer
MovieID: unique identifier for movie
TheaterID: unique identifier for theater
ScreenID: unique identifier for screen
Title: name of move
Rating: based on MPPA ratings
Genre: type of movie
Duration: length of movie
Release Date: day movie came out
Name: Name of the theater
Location: Address of the Theater
Capacity: Amount of avaible seats
ScreenType: regular or deluxe
Showtime: time movie will be played

7.2 Description

The Data Management Diagram helps visualize the online theater systems data management strategy involving SQL. The diagram displays three separate databases labeled "*Database 1*", "*Database 2*", and "*Database 3*". The following descriptions explain the diagram's databases and the relationships between all of the classes and the three databases.

7.2.1 Database 1:

Database 1 is the database with the least sensitive information so the security of the following classes is not the first priority in terms of protection. The first class within Database 1 is the Movie class which stores data relating to movies playing in the theater. The data attributes of this class are MovieID, Title, Rating, Genre, Duration, and ReleaseDate. The Movie class has a relationship with the Ticket class within Database 2. The second class is Theater and stores data attributes TheaterID, Name, and Location. This class has a direct relationship with the Screen class also within Database 1. The Screen class holds the ScreenID, Capacity, and ScreenType within the specific theaters. Along with the Movie class the Screen class has to also be linked to the Ticket class within Database 1 as the Movie, Theater, and Screen are all important to the Ticket type with their own special attributes within the Ticket class.

7.2.2 Database 2:

Database 2 has a higher priority in terms of security because the data it stores more sensitive information regarding the customer and their personal information. The first class is the Customer class which holds all of the relevant customer information needed to use the online theater ticket system. The attributes are the CustomerID, Email, Password, First Name, Last Name, Address, and FavoriteTheater. This class has two direct links to the Ticket class within the same database and the Payment class within Database 3. The link between both classes is very similar to the other classes link to Ticket. One of the data attributes both classes share is the CustomerID. The Ticket class is the second class within Database 2 which also has three connections with other classes. The data attributes include TicketID, CustomerID, MovieID, TheaterID, ScreenID, and Showtime. All of these attributes are very important to the entire transaction between customers and the theater ticketing system.

7.2.3 Database 3:

Database 3 is the most important database with the highest security priority because it holds the most sensitive information regarding the payment between the customer and the payment system for the theater. The Payment Class is a standalone class with the Customer class having a connection to this class. The reason for this is because the customer's information extends to their payment information. The data attributes for the Payment class are Card Holder, Card Number, Expiration Date, and CVC. All attributes are fields for the customer's card used for payment.

7.3 Discussion and Justification

7.3.1 Justification

Database Justification:

In our ticketing system, we went with three separate databases: One for account (customer) information, one for payment information, and one for our theater and movie information. The reason this route was taken was two main reasons: security and optimization. For these reasons

as well, all of our databases will be SQL. For security reasons, we want our payment information stored with PCI standards, which requires strict proof of proper SQL security.

As for optimization, we expect the movie and theater database to be accessed much more frequently than account information, so we will expect a more robust server behind the database optimized for speed over security. The account database will have sensitive information, but still be accessed often, so it will need to be optimized for a good balance of speed and security. The justification for storing payment in a separate database from the customer's account information is to avoid any extra processing time, and since our payment information will be stored behind much tighter security, accessing the information will require more resources, but we will not require it to be prioritized for speed, as we expect it to be accessed rarely.

Databases:
-Movie, Theatre, and Screen: These classes all share a database. None of them contain any sensitive information so a breach would not affect the company as a whole, and these will be accessed the most frequently out of all our other databases, so this database will be optimized for speed over security.
-Customer/Ticket: These two classes make up our account database. While needing to be secure as this information is still considered sensitive, we still expect this information to be accessed often, so we will not be storing it behind as strict of a security wall as the payment information. We expect the database containing these classes to have a good balance of speed and security.
-Payment: Our payment information will be our most secure database, stored alone and to exact PCI compliance, which will require the database to adhere to the harsh SQL security standards. This is because any payment information that is compromised will cause a major issue for any customer affected and the company, possibly tarnishing the reputation as a whole, and resulting in a lawsuit. This information is expected to be accessed rarely, only once during every completed transaction, so optimization of speed is of low priority. An additional security justification for having payment stored separately would be that we would not want any individual management employee (who will have access to the customer information) be able to access the payment information, only a high level admin.

7.3.2 Discussion

Class:

Theater: For the Theater class, we included TheaterID, Name, and Location in a single table. An alternative approach could be to separate Location into a distinct Location table to reduce data redundancy if multiple theaters shared the same location details. This separation would ensure that updates to a single location could propagate across all relevant theaters, improving data integrity and making it easier to manage common locations. The trade-off, however, is that it introduces additional joins in queries involving theater data, which could slow down performance. Given the likely one-to-one relationship between each theater and its unique location in this system, keeping Location in the same table was considered the simpler and more efficient choice.
Screen: In the Screen class, we included ScreenID, Capacity, and ScreenType as the core attributes. An alternative design could add more specific details, such as Audio Quality or 3D Capability, to provide a richer description of each screen’s features. While this approach would

improve the detail available to customers and make it easier to identify between screens, the trade-off is increased data redundancy, as many screens may share similar characteristics. Additionally, the database structure would become more complex, making it harder to update data when screen specifications change. To keep the system lightweight, only needed types were included, as additional details were supposedly unnecessary for the current scope of the ticketing system.

Movie: The Movie class holds many types such as Title, Rating, Genre, Duration, and ReleaseDate. An alternative design could involve splitting Genre and Rating into separate tables, allowing each movie to link to multiple genres or ratings, thereby supporting more flexible categorization and filtering. While this approach would enable a richer user experience and make it easier to manage commonly used genres and ratings, the trade-off is the added complexity in database management and increased query times due to joins. For the ticketing system's purposes, where each movie has a straightforward genre and rating, keeping this information within the Movie class ensures faster querying and simpler database structure.

Customer: The Customer class includes CustomerID, Email, Password, First Name, Last Name, Address, and FavoriteTheater. An alternative approach could be to separate sensitive fields (like Password and Email) into a separate Authentication table, improving security by isolating personal information. Another option would be to implement encryption for sensitive fields, adding an additional layer of security. The trade-off here involves performance, as encrypting data or using separate tables increases renewal time and complicates access to customer information. Given the scope of the system, storing customer details in a single table was supposedly acceptable, with the understanding that further security measures (like hashing passwords) would be applied as needed.

Ticket: The Ticket class was designed to contain TicketID, CustomerID, MovieID, TheaterID, ScreenID, and Showtime. A possible alternative would be to separate Showtime into a distinct table, linking it to the Ticket class. This approach would make updating showtimes more efficient, especially for tickets associated with the same movie and screen. The trade-off is that additional joins would be required to retrieve complete ticket information, potentially impacting performance. Given that each ticket is linked to a unique showtime, installing Showtime directly in the Ticket class keeps the system efficient and simple for this context.

Payment: In the Payment class, we included fields for Card Holder, Card Number, Expiration Date, and CVC. An alternative design would be to express the Card Number or outsource the payment processing to a third-party provider to enhance security and reduce the risk of data breaches. Encryption replaces sensitive card details with a unique identifier, and outsourcing shifts the responsibility of data protection to a specialized provider. The trade-off for encryption or outsourcing is the added complexity of integration with an external system, potential costs, and possible limitations in data accessibility for internal analysis. For a smaller-scale system, retaining basic payment details in the database, along with encryption and security best practices, was determined to be acceptable.

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