Movie Ticketing System

Software Requirements Specification

Version: 4

Date: 03/13/2024

Group # 2
Anirudh Nadella
Rithish Sivaraj
Riley Potter

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

Revision History

Date	Description	Author	Comments
2/14/24	Version 1	Anirudh Nadella, Rithish Sivaraj, Riley Potter	Initial version of SRS document, up to 3.6
2/28/24	Version 2	Anirudh Nadella, Rithish Sivaraj, Riley Potter	Updated SRS document, upto Development Plan and Timeline
3/13/24	Version 3	Anirudh Nadella, Rithish Sivaraj, Riley Potter	Update 3: Includes the SDS Test plan.and the test cases
3/27/24	version 4	Anirudh Nadella, Rithish Sivaraj, Riley Potter	update $: added software design and included tradeoff discussions

Document Approval

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

Signature	Printed Name	Title	Date
	Riley Potter	Software Design	2/14/24
	Dr. Gus Hanna	Instructor, CS 250	

Table of Contents

1. Introduction

This document will discuss the design and implementation of the {Name} Theater Ticketing System. This theater ticketing system will consist of an interactive interface for users for viewing upcoming show dates, open seats in different theaters, and allow them to purchase tickets and snacking options. It will also consist of many different security and user features to ensure a smooth operation to maximize user convenience in finding and booking movie tickets.

1.1 Purpose

The purpose of this SRS document is to highlight the requirements and processes that go into producing this Theater Ticketing System. Audiences for this document are developers and interested users who wish to view the functionality of our system.

1.2 Scope

This subsection should:

(1) In this project, we will be using MongoDB, Report generator, Paypal, DreamHost, Payment Gateway.

(2) MongoDB will help in storing various databases, such as clientele information, open and closed seats, booking information, movies and run times in their respective theater rooms, and even information such as pricing.

(3) This system will primarily be used only for ticketing customers for movies and for snacks in a convenient manner:

(a) The primary goal for this product is to increase convenience in booking movie tickets for clientele. We aim to have the process of booking be much more efficient and speedy than existing ticketing systems. We aim to do this by increasing processing, and server speeds, as well as include various techniques and methods like double booking prevention, and include a clean user interface for clear viewing of all the options in the theater.

1.3 Definitions, Acronyms, and Abbreviations

The following list will contain all the definitions, acronyms and abbreviations:
Sys - System
SRS - System Requirement Specification
DB - Database
RM - Report Manager
UI - User interface
{More to be added}

1.4 References

(1) This document was written with guidelines taken from the IEEE document and the SRS4.0 document, and the “software specifications with an example” document as a template.

1.5 Overview

(1) The rest of the SRS will contain more information on the details of the project. Such as special features, characteristics, use cases, and so on.
(2) This SRS will begin with a general description of the product and its functions, features and characteristics, and move on to the topic of system specific requirements. Then it will describe the analysis model and even discuss how to manage change, and version control.

2. General Description

This theater ticketing system will allow users and clients to easily view an understandable UI, and book movie ticket seats in different theaters. It will also allow clients to pre-book snack and meal options to be delivered to their seats within the theater. Aside from this, the business using this can track business analytics and statistics as well, which will allow them to make changes to their business which would in return help enhance their methods and techniques.

2.1 Product Perspective

This design will focus completely on the theaters in San diego. The ticketing web page application prevents bots from buying tickets in bulk and also has features of customer support. This design has one database integrated to all the theaters which is easy and has centralized management. This design also supports administrator modes and can extract the online reviews and display it in the webpage.

2.2 Product Functions

The online theater ticketing system sells movie tickets online to 20 theaters and has a feature to block the bots who are trying to buy more than 20 tickets per show. It shows the online reviews and critic quotes for the movie. There are two types of seating available: deluxe and the regular seating, and it shows the available shows and the seats available. This design also provides special discounts for students, military, and senior citizens. It also has a customer support system and an administrator mode. This design has a single database which is integrated to all the theaters to update the shows and ticket availability.

2.3 User Characteristics

The target audience for this online theater ticketing system design is the general public who wants to watch a movie, so an easy-to-navigate user interface is required as per the characteristics, which can be easily manageable to access for different age groups, from teens to senior citizens. There would also be users who are active moviegoers who want to prefer fast and easy bookings, and there would be users who would like to watch the reviews before watching a movie, so they would prefer a user interface that would make it easy for them to look at the reviews conveniently and book the show.

2.4 General Constraints

The main constraint will be the integrations constraint since it is only a web browser based design as per the clients requirements. The design has to heavily rely on the other integration

softwares for online transactions and online reviews, which limits the developers options for designing the system because of the collaboration of external providers. Since we are designing a webpage model, the designs should be compatible with all the web browsers that exist in the market, and their security requirements are slightly different from each other, which would be another minor constraint.

2.5 Assumptions and Dependencies

This design assumes that the users have a good and stable internet connection and that they have a device that has internet access to it. The design also assumes that refunds are in-person, as per the client's idea. This design is dependent on the external online movie review sites for displaying the movie ratings of the movies, and it is dependent on third party applications for the payments and for customer support.

3. Specific Requirements

This system will have many functional and non-functional requirements, as well as requirements for each category of the interfaces. This section mentions all the capabilities of the system and will explain each role of the functions in a detailed description. Each requirement is unique and has an identifier to easily identify it and segregate it. The detailed description of the requirements follows below:

3.1 External Interface Requirements

3.1.1 User Interfaces

The product will have different sets of User Interfaces for different users. General “buying” users will have user interfaces necessary to find and purchase tickets. Administrative users will have options for updating movie lists. Theater users will be able to update their databases of existing movies and their theater conditions. Customer Service users will have access to current customer service requests.

3.1.2 Hardware Interfaces

The product should behave in two distinct ways, mainly different in their spacing, layout, and visual style. One way should be used if the system detects it is being used on a mobile device such as a phone or tablet. The other should be used if the system detects it is being used on a laptop, desktop, or other device. It must detect various possible forms of user input from these types of devices, including touchscreen and mobile keyboard input on mobile devices, or mouse and keyboard input from desktop devices.

3.1.3 Software Interfaces

The product will be operating on a website, so it may use HTML and CSS for the general structure of the website, Javascript for the more complex scripting aspects, and PHP and SQL for databases of theaters and movies. It will need to interact with whatever browser is communicating with the website, and should support common browsers such as Chrome, Firefox, Safari, and Edge.

3.1.4 Communications Interfaces

The product should interface with electronic mail systems for the purposes of sending tickets to buyers. It must also interface with a Payment gateway service in order to accept payments for tickets. It also must be able to interface with popular movie review websites to display ratings for movies. In order to find nearby theaters to given locations, the system should interface and take information from a Map app such as Google Maps.

3.2 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

3.2.1 Displaying Trending Movies

3.2.1.1 Introduction

- This feature is the first thing a User will see upon entering the home page of the site. A list of trending movies will be displayed, available for selection (taking the user selecting them to the movie's page on the website).

3.2.1.2 Inputs

- The list of movies will be automatically and manually updated over time: It will include both movies which have had the most purchases within the week, and it can be added to manually with new anticipated titles by administrative users.

3.2.1.3 Processing

- The list will be its own small dataset, stored as a subset of all available movies

3.2.1.4 Outputs

- The list's posters will be displayed to users upon entering the website. The movies will be available for the user to navigate to (bringing them to movie pages described in 3.2.5)

3.2.1.5 Error Handling

- If the list is empty and movies cannot be displayed, empty spaces where posters would exist can be displayed instead.
- If there are too many movies in the list to fit in the display space, a random selection of movies in the list which will fit the display space will be displayed.

3.2.2 Creating and Logging Into Accounts

3.2.2.1 Introduction

- This feature allows for user convenience by storing things such as a user's previous transactions, and their payment information.

3.2.2.2 Inputs

- A User can create an account by selecting a unique username and password. Whenever they make a purchase (if they have not taken this option already) they can be prompted to save their payment information for future use. Their account will be created with a linked email address, for verification and ticket-sending purposes.

3.2.2.3 Processing

- Users and all of their information will be stored in a secure database.

3.2.2.4 Outputs

- Users will be able to use their stored information for viewing nearby theaters and entering payment information more easily, as well as seeing their previous transactions and tickets.

3.2.2.5 Error Handling

- No account can have the same username, password, or linked email address as another account.
- If two accounts are simultaneously created with the same username, password, or linked email address, the users attempting to create the accounts will be informed of the issue and sent back to pick new usernames, passwords, or linked email addresses.

3.2.3 Browsing and Searching Movies

3.2.3.1 Introduction

- This feature will allow users to browse the database of currently or eventually screening movies, and use a search function to find desirable ones. Navigating to a movie from the database will lead the user to the movie's details as described in 3.2.5.

3.2.3.2 Inputs

- The database of movies will be maintained by administrators. Each administrator will have its own log-in credentials, allowing them to add to the movie database with new titles, release dates, posters, and trailers. Customers will be able to see this database in a browning menu, and may input the name of a specific movie, which will display relevant movies.

3.2.3.3 Processing

- The database will be stored and maintained by administrators.

3.2.3.4 Outputs

- The database will be visible to customer users and theaters. If a customer chooses a movie from the database, they will be brought to its page (described in 3.2.5).

3.2.3.5 Error Handling

- when no matches in the database to a given movie title are found, this will be indicated to the user.
- To prevent confusion with movies of the same name, any movie which shares a name with another movie will have its release year displayed next to its name in the browsing menu.

3.2.4 Browsing and Searching Theaters

3.2.4.1 Introduction

- This feature will allow users to browse the database of affiliated movie theaters, and use a search function to find nearby ones. Picking a theater will display its currently-showing movies and their showtimes. Selecting a showtime will allow the user to proceed to transaction, as described in 3.2.7.

3.2.4.2 Inputs

- The database of theaters and their showtimes will be added to by Administrators and maintained by participating theaters. Each theater will have its own log-in credentials allowing it to modify current showtimes, conditions, and screening movies. Customers will be able to see this database in a browsing menu, and may input a specific theater, or a location, which will display nearby theaters to that location. The system will take this location information from a mapping software such as Google maps.

3.2.4.3 Processing
- New theaters and their associated accounts will be added to the database by Administrative accounts. Theaters will manage their own movie schedulings using their associated accounts.

3.2.4.4 Outputs
- The database will be viewable by customer users, and used to find currently-showing movies at any given theater.

3.2.4.5 Error Handling
- Any missing theater data (primarily name and locational data) will cause theaters to not be displayed in the database, and notify both the theater user and the administrator to the issue so it may be resolved.

3.2.5 Viewing Ticket Options and Information for Any Movie

3.2.5.1 Introduction
- This feature specifies how users will be able to see information about and interact with any particular currently-showing movie. Each relevant movie (part of the database) will have its own page which will display current theaters showing the movie and their showtimes (which will allow a User to navigate to specific tickets to purchase), the poster, a trailer, and relevant information about the movie including ratings from popular review sites.

3.2.5.2 Inputs
- Each movie will have its information (including name, poster, trailer, and release date) added from an Administrator and its showtimes and available theaters will be provided by the theater personnel. Data from popular review sites will be automatically viewed for the overall rating of the movie.

3.2.5.3 Processing
- Reviews will be averaged and displayed, If the user has an account their location will be used to find nearby theaters showing the movie.

3.2.5.4 Outputs
- The User, viewing the page for the movie, will see the name, poster, trailer, overall rating, and theaters showing the movie (popular ones if the User has no location on file, nearby ones if the User does have a location on file). Theaters will provide showtimes which will allow users to navigate to the purchasing page for that showtime at that theater.

3.2.5.5 Error Handling
- If any necessary displayed data is missing from the movie, the page for the movie will still be displayed with a message explaining the unavailability of the information (for example, if the poster is not added by an administrator when putting the movie into the database, it may display as the text “Poster not available.” Similarly, this will display if information from review sites could not be found).
- No movie can be entered with no title, to prevent issues with the browsing system.

3.2.6 Transaction: Paying for and Receiving Tickets

3.2.6.1 Introduction

- This feature involves the primary transaction between the User and the System: buying a ticket. Once a user is viewing the type of ticket they want (the movie, time, theater, and specific viewing room), they will have the option to purchase it.

3.2.6.2 Inputs

- User payment information must be collected once the option to purchase is selected. This must include an email address if the User is not using a registered account. The User, having entered their information, will be prompted one more time, with a clear statement of the ticket they are purchasing and its cost, to ensure they are certain of their purchase.

3.2.6.3 Processing

- User payment information must be used with a payment portal service. Once the payment is verified, the appropriate ticket will be added to the user's transactions (if they have an account), and sent through the provided / user's email. The relevant seat in the relevant theater's showroom will be changed to occupied for that showing.

3.2.6.4 Outputs

- An image of the appropriate ticket will be displayed for the user, the transaction and ticket will be stored in the user's profile (if applicable), and the ticket will be sent to the provided email address.

3.2.6.5 Error Handling

- To prevent double-booking, while the user has selected a ticket to buy, it will be unavailable for all other users. If somehow two of the same ticket are selected at the same time, this should be detected by the system and the users should be made aware of the error and sent back to the showtime/ticket selection page.

3.2.7 Customer Service

3.2.7.1 Introduction

- This feature involves two types of users interacting: Customers and customer service personnel. Customers

3.2.7.2 Inputs

- Customers can select an option to open a service request. They will be shown an empty chatroom, which will remain empty until a customer service account chooses to join them. Customer service personnel will be shown an updating list of active customer service requests, and may select one to join and converse with the customer. Both will input their text to the chat interface.

3.2.7.3 Processing

- Active conversations will use a real-time chat system, requiring a system to update each in real time with the inputs of the other.

3.2.7.4 Outputs

- Once the conversation is complete, a log will be generated and saved associated with the customer service account involved in the interaction.

3.2.7.5 Error Handling

- Once a customer service personnel has opened a conversation with a customer, the request must immediately be removed from the list of active requests, to prevent multiple attempts at once to interact with the customer.

3.2.8 Anti-accidental-transaction system

3.2.8.1 Introduction

- This system is in place to protect users who may accidentally, due to technical or user error, attempt to purchase multiple tickets where they mean to purchase one. It will track when an account or machine purchases two of the same ticket at the same location within an hour or one after the other.

3.2.8.2 Inputs

- This feature will require tracking recent purchases from any device or user for one hour, and checking new purchases (detailed in 3.2.7) against the most recent purchase from that account.

3.2.8.3 Processing

- Each new purchase must be checked against the machine and account performing the purchase and its list of recent (and within-one-hour) purchases. This list should be purged when an hour passes after any purchase.

3.2.8.4 Outputs

- Simply, the system will display a message to the user in the event of this feature's activation, informing them of two very similar consecutive transactions. They may choose to go on with the transaction, or not.

3.2.8.5 Error Handling

- If this system detects many purchases in a row, it may flag the series of similar transactions as anomalous to an administrative account

3.3 Use Cases

3.3.1 Use Case #1

Case Title: Purchase Ticket via Searching By Movie
Actor(s): Buyer
Flow of Events:

1. Buyer opens Firefox.
2. Buyer enters the URL for our Ticket System's website into their browser.
3. The website loads, displaying featured popular movies to Buyer.
4. The system checks whether Buyer's computer has recently been logged into an account on the website. It has, so the Buyer is automatically logged into their account.
5. Buyer opens the Browsing menu and types "Spiderman" into the search bar.
6. The browsing menu displays a list of currently-showing movies with "Spiderman" in the title, prioritizing by the number of tickets bought for each movie within the last week.
7. Buyer clicks the option at the top of the list, "Spiderman: Beyond the Spider-verse."
8. The system displays the page for the movie, including ratings, the poster, a trailer, and the list of theaters currently screening the movie alongside available showtimes at those theaters. The list is ordered by proximity to Buyer's account's stored location (which they have provided previously).
9. Buyer selects a screening of the movie the next day at the theater closest to them.
10. This brings them to the transaction page, where they are prompted to enter their payment information or to automatically use information on file. They select automatic entering, and their payment details are filled out using information they have provided previously.

11. Buyer clicks “purchase,” and is prompted once more with a description of the purchase including the theater, showtime, movie, and cost, to fully accept the transaction.
12. They accept, and the purchase (alongside the appropriate ticket) is listed in their account’s transaction list. An email with the ticket is sent to their address as well.

Assumptions / Entry requirements:

- Buyer has a computer and browser (firefox) compatible with the System
- Buyer already has a registered account with payment and location information stored
- Buyer is looking for a movie which exists in the database

3.3.2 Use Case #2

Case Title: Add New Movie and Showtimes to Theater
Actor(s): Theater
Flow of Events:

1. Theater opens Chrome.
2. Theater enters the URL for our Ticket System’s website into their browser.
3. Theater selects “log into account” and enters the login credentials associated with the movie theater they are affiliated with.
4. They navigate to a new available page through clicking a new available option in their account, “edit screenings”
5. A list of all currently-scheduled movies to be shown in their movie theater is displayed, alongside the option to add another or remove or edit any movie’s scheduling.
6. They choose to add a screening schedule.
7. The browsing database of all movies is shown to the Theater. They enter into the search bar, “Frozen 2.”
8. A list of relevant movies matching or similar to that title appears.
9. They click the option at the top of the list, Frozen 2.
10. They are prompted to add scheduled screenings. They add a screening at 4PM every day of the following week in their showrooms numbered 3, 4, 5, and 6.
11. They press a button labeled “add schedule.”
12. A message explaining conflicts with that schedule appears: Shrek 4 is already scheduled for a screening in their 5th showroom at 4PM the next week.
13. They close the message and remove the showroom 5 showing from the prospective schedule. They again press “add schedule.”
14. The page navigates back to the list of currently-scheduled movies, now including the showings of Frozen 2.
15. Theater closes the window.

Assumptions / Entry requirements:

- Theater has a computer and browser (chrome) compatible with our software
- Theater has a registered theater-type account and has an existing schedule of screenings

3.4 Classes / Objects

Different types of user: Customers, Administrators, Theater personnel, Customer service personnel

3.4.1 Customer

3.4.1.1 Attributes

- Basic user data (unique username, password, and linked email address)
- List of all previous transactions made by the account
- Optional stored payment information
- Optional stored location

3.4.1.2 Functions

- Can browse movies and theaters and purchase tickets
- Can view previous purchases
- Can request customer service

3.4.2 Theater

3.4.2.1 Attributes

- Basic user credentials (unique username, password, and linked email address)
- Number of showrooms and seats in those showrooms
- List of all currently and scheduled screening films, the showrooms and showtimes for which they are screening
- Physical address of the associated movie theater
- Optional list of current conditions (closed showrooms, etc)

3.4.2.2 Functions

- Can schedule movies to be shown at certain times in certain showrooms of their associated theater, allowing Customer users to purchase tickets for those scheduled screenings.

3.4.3 Customer Service

3.4.3.1 Attributes

- Basic user credentials (unique username, password, and linked email address specifically for receiving customer service requests)
- Log of recent conversations

3.4.3.2 Functions

- Can connect to current customer service requests and converse with users requiring customer service

3.4.3 Administrator

3.4.4.1 Attributes

- Basic user credentials (unique username, password, and linked email address)
- Log of recent interactions with the system (changes to user data or the movie database)

3.4.4.2 Functions

- Can edit user data, view transactions, conversations, and movie schedules of other users
- Can add to the movie database, adding new titles with information, posters, and trailers.

3.5 Non-Functional Requirements

This system will carry a number of non-functional requirements. Transactions shall be completed in a total of 5 minutes, while the payment transaction itself will be completed in a second. Live

chats will be responded to with an automated AI chat bot in a maximum of 5 seconds. More such requirements are described below:.

3.5.1 Performance

- Can perform at high speeds even when user count exceeds 1000 users per hour.

3.5.2 Reliability

- 98% Success rate in transactions assuming average - high speed internet connection.

3.5.3 Availability

- Will be available on any browser on any device, so long as the device carries a recent version of the OS.

3.5.4 Security

- High security and confidentiality in transaction and customer information, also maintains the security protocols followed by the web browser

3.5.5 Maintainability

- Easy to maintain by an administrator so that if something falters, admins will have full access to the source code and have reduced downtimes.The admins have a chance to make adjustments to override.

3.5.6 Portability

- Can be accessed on any device on any web browser, assuming decently strong internet connection is established.

3.6 Inverse Requirements

This theater booking webpage is designed in a manner where refunds are not directly given to the customers through the webpage; they are handed through in person. There is no option to book the shows 5 minutes after the show start date, and the system only allows users a 5-minute window to book the tickets and complete the payments. The system won't allow the users to buy more than 20 tickets at a time.

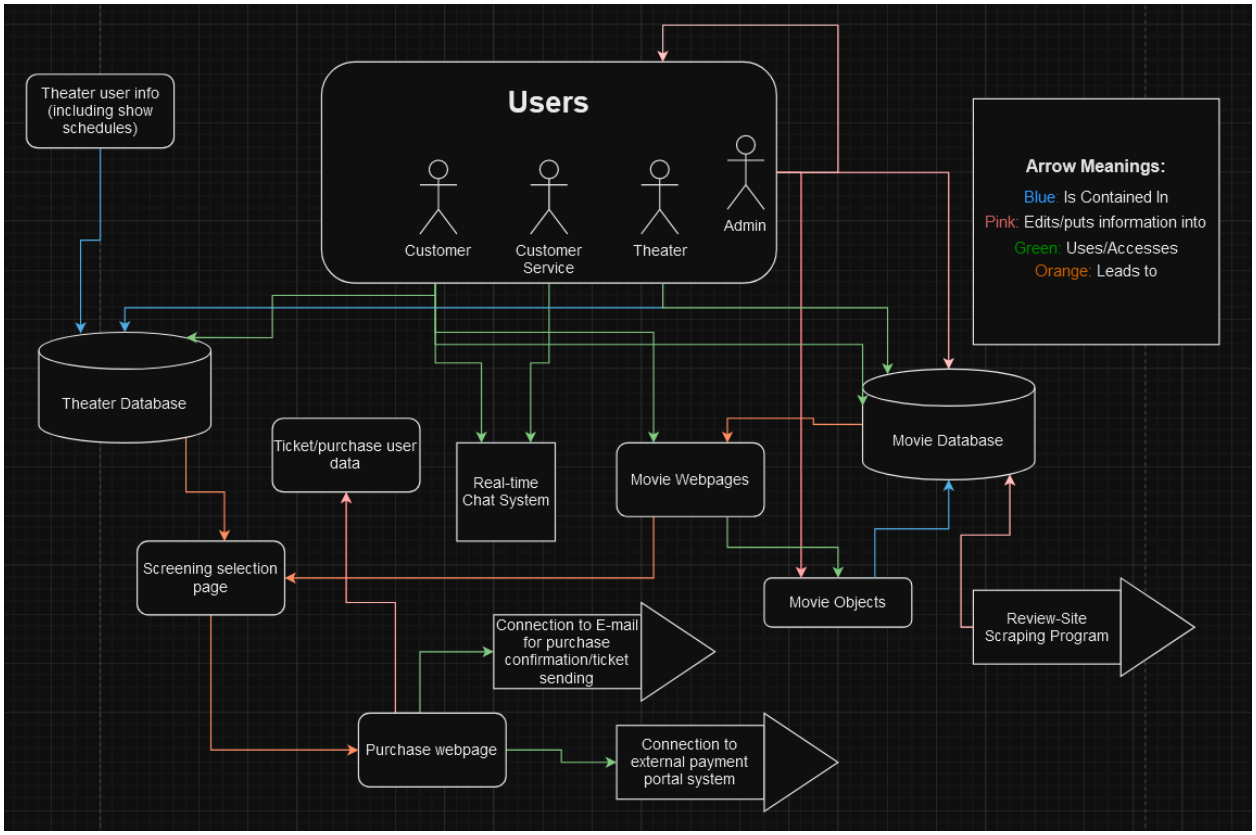
4. Software Design Specification

4.1 System Description

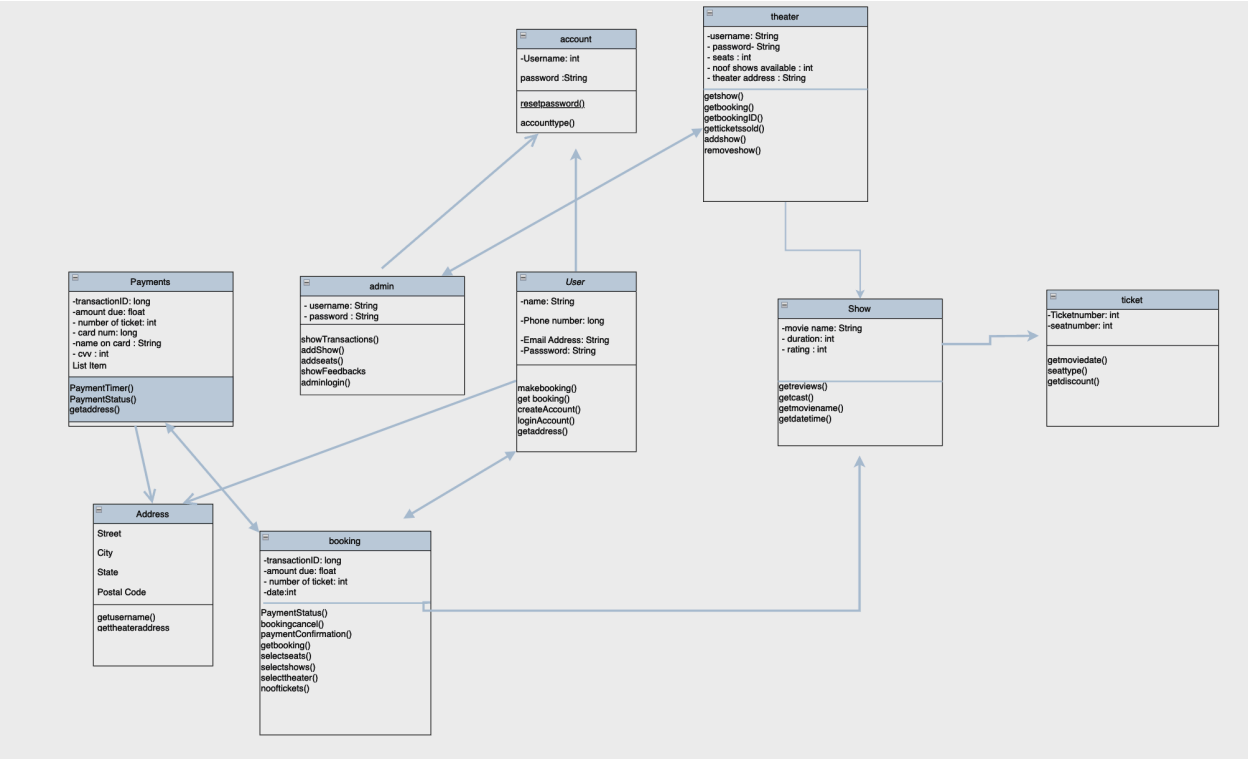
This is a system for theaters to sell tickets to users. Participating theaters may schedule showtimes for any one of a set of movies stored in a database, and make tickets available for purchase for those showtimes. Users who wish to buy tickets may search for certain theaters or certain movies, and find available locations and showtimes for those movies, at those theaters. They may then use their virtual payment information to purchase access to those tickets, and reserve spots for those showtimes at the theater they bought a ticket for. Also included is a

customer-support chat system, and administrative accounts which add new movies and theaters to the relevant databases.

4.2 Architectural Diagram



4.2 UML Diagram



4.3 Description of classes:

1. Account:
    - uses: this class is used for the users and the admin to login
    - operations:
        - resetpassword() operation , helps to reset passwords
        - accounttype() operation, helps to change the account type from admin , user and theater owners
2. Theater:
    - uses: This class is used by the theater owners to modify the shows add,remove and look at the tickets sold
    - operations:
        - getshow: this operation retrieves the current shows that are displayed on the website
        - getbooking: this operation get any past booking that the theater owners wants to refer to
        - getticketshow(): this operation displays the current available seats on a current show

- removeshow(): this operation removes a show that are to displayed on the website
- addshow(): this operation adds a new show that are to be displayed on the website

3. Payments
    - uses: This class is used for the customers payments, it has customers payment details and stores it in database which they can later access them
    - operations:
        - paymentstatus() this operation tells the user whether their payment is successful or not
        - paymenttimer() this operation sets the payment time for 5 mins can decided by the client
        - getaddress() this operation gets the users payment address
4. Admin
    - uses: This class is for the admin login where he can manage all the theaters and check the number of tickets sold in each theater, the admin can also add more shows and seats and can check and write feedbacks for the customers about the website
    - operations:
        - showtransactions(): this operation shows all the transactions made by the customers
        - addshow(): this operation adds the required shows to a certain theater
        - removeshow():this operation removes the required shows to a certain theater
        - removeseats():this operation removes seats for a required shows to a certain theater
        - addseats(): this operation adds seats for a required shows to a certain theater
        - showfeedbacks(): this operation shows feedback given by the customers and from the theater owners
5. Users
    - uses: this class is for the users to book tickets and view their past booking and modify their bookings
    - operations:
        - makebooking(): this operation creates a new booking for the user
        - getbooking(): this operation gets a past booking for the users and they can modify it
        - createaccount(): this operation is to create a new account for the new users

- loginaccount(): this operation is to login for the existing users
- getaddress(): this operation is to get and modify the users saved address

6. Show:
    - uses:this class is used to show all the current available Shows that are in the partnered theaters
    - operations:
        - getmoviename(): this operation is used to show the movie name which are available at certain show times
        - getdatetime(): this operation shows at what time and date the show and the movie is available
        - getcast(): this operation displays the cast info of the show
        - getreviews(): this operation shows all the reviews from the external sources
7. Ticket:
    - uses: this class is used for the users to show their ticket number and the type of ticket they bought
    - operations:
        - getmoviedate(): this operation displays the movie date and time for which the user bought the ticket for
        - seattype(): this operation is used to show the type of seat regular or premium seat that has been purchased
        - getdiscount(): this operation is used to apply discounts if the users have any promo codes or any AMC passes
8. Address:
    - uses: this class is helps the users to modify their address or look up the theater address of certain theater or customer care address
    - operations:
        - getusername(): this operation helps to identify the users if they want to change their address
        - gettheateraddress(): this operation is used to help the users to get the theaters address
        - getcustomercare(): this operation helps the users to find their nearest customer care center/ partner
9. Booking:
    - uses: this operation helps for the users to book manage their tickets
    - operations:
        - paymentstatus(): this operation checks the payments status of the users booking, weather its successful or not

- bookingcancel(): this operation helps the users to cancel their booking
- paymentconfimation(): this operation is used to send the payment confirmation to the users after the payment is successful
- getbooking(): this operation is used to get the booking information of past and current shows for the users
- selectseats(): this operation is helps the users to select their desired seats and type of seats
- selectshows(): this operation is helps the users to select their desired shows
- selecttheater(): this operation is helps the users to select their desired theater
- nooftickets(): this operation helps the users to select the number of tickets and if it's more than 20 its pops an error message

4.4 Description of Attributes:

1) Basic User Data:
    - This attribute describes all the basic information regarding a user when they make use of the theater ticketing system. This includes pieces of information such as, name, last name, unique username, password, and linked email address.
    - The data type being used for this will be VARCHAR
    - Name in code: BUData
2) Previous Transaction:
    - This attribute will keep a record of all previous transactions made, whether it is for food and snacks, or for movie tickets. It will also possess the capability to link purchases to the respective clients, whilst also not openly saving card and financial information to ensure security.
    - The datatype used for this will also be VARCHAR.
    - Name of variable: PrevTransact
3) Stored Payment Methods:
    - This will allow users to save their primary methods of payment within the site. Storing their payment information will make it quicker and easier to buy tickets in the future. However, this choice will be optional to the user.
    - The data type used for this is STRING.
    - Name of variable: StPayMethods
4) Number of showrooms and seats
    - This attribute will show the total number of showrooms which are there along with the number of seats situated in each showroom.
    - The data type used for this will be INT.

- Name of variable: NumRoomsSeats

5) Movies
    - This will display the movies that are running at the moment. It will do so in alphabetical order, or if chosen by the user, in order from most recent to latest release date. The movies will be displayed with their names.
    - The data type we will be using for this attribute will be VARCHAR (SQL), and dateTime.
    - Name of variable: MovieInfo
6) Addresses:
    - This will store all the addresses of each theater in which a movie is played at and from where tickets can be booked. This will assist in transport and in commute as the address of each theater will be automatically included in the text messages / emails for the movie tickets.
    - The data type we will be using for this attribute is VARCHAR (SQL)
    - Name of variable: TheatreAddr
7) Ticket Pricing:
    - This will hold information of the varying prices for tickets, and contain different pricing for each category of tickets based on age. Seniors will have to pay $15.55, Adults $18.00 and Children $12.50. Also, aside from these regular pricings, every Thursday, tickets will only be $5.00 and on special holidays such as Thanksgiving and Christmas, prices will also be lowered.
    - The data type used for this information is FLOAT.
    - Name of variable: TicPrice
8) Food Pricing:
    - This will carry the information regarding the various options for food which are held at each theater location, for pre-booking and delivery to your seat.
    - The data type used here would be FLOAT
    - Name of variable: FoPrice
9) Count:
    - Refers to the available count of purchasable things within the movie theater. This could be tickets, snack counts, merchandise counts. This would be helpful for employees and owners in seeing their stock count, and placing new stock orders as soon as possible.
    - The Data type for this is INT
    - Name of variable: Count
10) Inventory Names:
    - This attribute will keep track of all items in inventory and their brand names. Can be linked with the Count attribute to keep track of inventory stock and place reorders of supplies if necessary.
    - The data type used here is STRING

- Name of variable: Inv_Names

11) Employee Information:
    - Will contain a list of employee names (first and last), email, phone number, position, and pay rate. This will be useful when contact to various employees is needed within the business, and also, will provide employees with discounts when ordering through site, with employee user ID.
    - Data type used here will be: EmpInf (this is a data type made by us which consists of STRING, Float and Long.)
    - Name of variable: Emp_Info
12) Employee Logins
    - Will contain a list of all employees user ID, Passwords, and work email. When linked with the site, it will allow employees to enjoy personal and family and friend discounts when booking movie tickets, or purchasing concessions within movie theaters.
    - Data type used here will be STRING.
    - Name of variable: Emp_Log

4.5 Description of Operations:

1) Sale:
    - This operation will allow tickets and snack sales to be completed. It will process the searched up ticket/food, find the price from the database, and display the price to users so that they may complete the purchase. Once the purchase is completed, it will return a receipt to the customers email / phone number (Upon customers preference of either).
    - Operation name: SaleProcessing
    - Parameters: Price(Float), email/num(STRING/LONG), Receipt(STRING)
2) Returning Tickets / Food options:
    - This operation allows customers to return tickets within 24 hours of purchase, assuming the movie is still at least 48 hours away. Once the time falls within 48 hours of the movie's showtime, returns will not be accepted. Once a return is processed, the operation will return the customers money to their payment method / banks within 12 business days. If the time is within 48 hours of the movie showtime, the return option will automatically shut down and vanish from the user's side. With food options, returns will be accepted up until 2 hours of the film start time, as users may change their mind about their food and snack options.
    - Operation name: ReturnTicks
    - Parameters: MovieName(STRING), Price(FLOAT), email(STRING)
3) Movie Search / Display:

- This operation will allow users to search for specific movies they are interested in watching, and display all information regarding the movie to the users. It will display information such as, full movie title, cast, description of movie, title image, a trailer video, and the various pricing options too.
- Operation name: MovDisp
- Parameters: MovieName(STRING),Director(STRING)

4) Customer Service
    - This operation allows clients to ask questions regarding anything to a little box and it will return the String val “Thank you for your question! A member of our team will respond to your inquiry shortly!”. This will assist in gathering questions that clients may have in a simpler manner as opposed to having to constantly receive calls.
    - Operation name: CustServ
    - Parameters: Name(STRING), email(STRING), question(VARCHAR)
5) Selection of Seats:
    - This will allow users to select seats from a display of all the available seats. Once seats are clicked on, it will block the seats for that particular user for the next 15 minutes to allow them to complete their purchase. Once the seats are confirmed, the user will be able to view their selected seat numbers and rows, as well as the seat pricing in FLOAT.
    - Operation name: SeatSelect
    - Parameters: Seat(AlphaNum), price(Float)

4.6 Development plan and timeline

This Movie Ticketing System, should be a relatively simple project, completed within 4-5 months of work.While the first three months will be spent in perfecting the UI and ensuring smooth running of all the operations with minimal error, the remaining two months will be spent in gathering data from potential partnership theaters, and adding information and data to the site to be ready for user usage. These are the following roles of the people involved in this project and their expected responsibilities.

1) Senior Project Manager (Rithish):
    - This project manager is responsible for the proper and efficient work of all the various teams involved in completing this project.
    - This role is responsible for keeping close contact with all the Team Managers, and ensuring all tasks are completed as follows, and will act as the main point of contact to find assistance for teams struggling to complete a particular task. The
    - The Senior Project Manager will also be responsible for leading team meetings to ensure communication between the teams.

- Will ensure that their team's particular set of tasks and responsibilities are getting done.
- Will also work alongside their team in all technical aspects of the project.
- Will be responsible in creating reports of any improvements/digressions within their teams tasks and the overall build of the project.

2) Team Members (Programmers)(anirudh):
    - Will take up tasks as requested by their Project Managers, and attend team meetings to communicate any concerns or ideas.
3) Security Team(riley):
    - Will work separately from the other teams.
    - Focuses on ensuring security for the system.
    - Must also ensure that data leaks and private information is not revealed.
4) Payment Team:
    - Takes care of all design in relation to the payment aspect of the project.
5) UI Team:
    - Will design the basic outline and the detailed user interface for this project.
6) AI Team:
    - Will work in making use of AI for Customer Service Chatbots and more.
7) Food Purchase Team:
    - Will work in designing the food menu options based on information provided by the theaters.
8) Business Partnerships Team:
    - Will make connections and tie-ups with various theaters to work with the system in order to improve their and our business.
9) Testing Team:
    - Will make use of test cases in order to ensure each operation and functions are running without issues and report to the Junior Manager in order to get bugs and defects resolved.

Timeline:

Month 1: Basic outline of all programmed tasks must be completed.

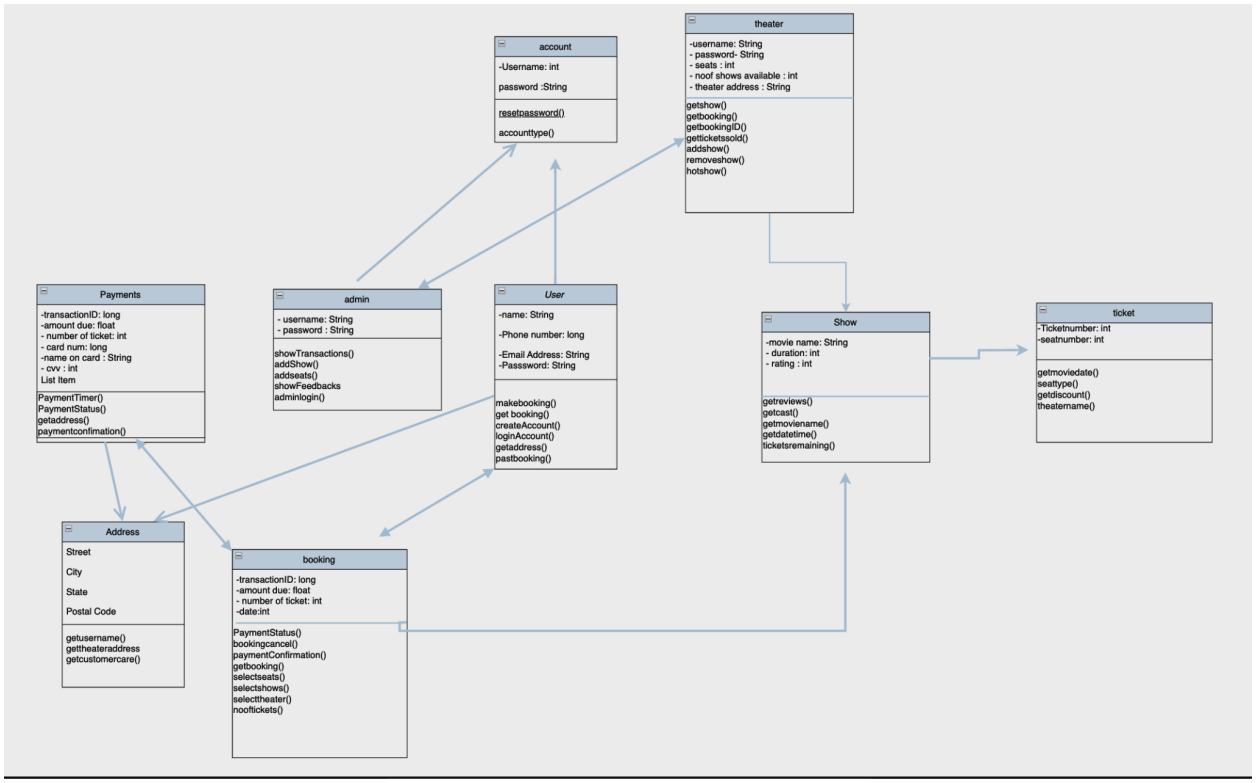
Month 2: Any design suggestions and fixes must be implemented and approved.

Month 3: Final UI and UX must be completed.

Month 4: Testing

Month 5: Complete all initial partnerships and open to the general public.

5. Software Design Specification:



We have updated the UML diagram with few the new functions that increases which helps users to navigate through the website we have made the following changes to the UML

a. User class, has been added with a method of pastbookings , where the users can retrieve the past bookings on their profile instead of scrolling through the booking section to access it
b. The show class now has a tickets remaining method with shows the ticketing remaining for the shows the user have selected
c. The ticket class shows the theater name function on it which makes it easy for the user to remind which theater they have booked the show for
d. The theater class has a hot show function which displays the top movies rated for the week
e. Payments class has now has a payment confirmation function for which when a user makes a payments it sends an email to the user about the payment confirmation

6. Verification Test Plan:

A. Unit Testing for different classes :

1. Testing for the user class :
    a. Description of Test: To check if the user can be able to create a new account or login to their existing account
    b. Test Set/Vector(s): the user details, address along with their username and password
    c. Output to be Expected: the output expected is that the user can create a new account or login to the existing account without any bugs
    d. Test fails if: the test fails if the user tried to create a new account but it failed to create the account or if the user is trying to login and fails to login

2. Test for admin class:
    a. Description of Test: to check if the admin can be able to login and use all the methods that are required
    b. Test Set/Vector(s): need the admin username and password in order to login
    c. Output to be Expected: the expected output would be that the admin login is successful and can use all the functionalities in the class like showing the recent transactions and reacting to the feedbacks
    d. Test Fails if: the test is considered fail if the admin login is not successful and if any of the functions failed to work

3. Test for booking class:
    a. Description of Test: this test is to check and validate if the booking class can be able to do all the specific functionalities
    b. Test Set/Vector(s): the test set are the transaction id and the date
    c. Output to be Expected: the output expected is that the class displays the correct functionalities like the number of tickets purchased and the theater name when the unique booking id and the date is entered by the users
    d. Test Fails if: the test is considered as fail when the class shows wrong functionalities and the wrong results or doesnt even show result when a certain functionality is used

B. Integration Testing of different classes/ function:

4. Test for SaleProcessing function:

a. Description of Test: Will test to see if sale is processed correctly for different versions of sale such as through online, mobile, and if in person purchases are updated on site.
b. Test Set/Vector(s): An online site/mobile phone connected to sufficient wifi, and in person sale device connected to network.
c. Output to be Expected: Sales should go through instantly, and transactions should be confirmed by the associated bank for all devices. Instantly, the booked seats should be marked as booked on all sites, and for the front register as well.
d. Test Fails if: Test is considered to be a fail if all expected outcome requirements are not met.

5. Test for address class:

a. Description of Test: to check and validate that the address has integration to the payment class and the users class and the users can change their existing address
b. Test Set/Vector(s): need an user and have a valid payment card information to check and validate
c. Output to be Expected: the expected output is that the user can change their existing address and they change see the changes they made and moreover when the user is making payments the class has to get the address from the address for the billing details
d. Test Fails if: the test is considered fail if the class fails to update the user address or the payment class failed to get the address from the class

6. Test for ReturnTicks Function:
a. Description of Test: Will check to make sure if requested returns fall before 48 hours of movie showtime, and run branch statements based off of this. If yes, the return will be processed and the customer will be sent a message stating, “Return was processed. Transaction amount should return to your account within the next 14 business days. Please contact, accounts@moviesys.com for any other questions.” Once this message has been sent, return should be processed no matter what device return is requested. If no, the customer should be sent a message, “Sorry, time for return is passed. Please contact accounts@moviesys.com for any further assistance. Thank you.”
b. Test Set/Vector(s): Devices associated with software, and a bank account which has completed a previous transaction.
c. Output to be Expected: Time should be compared to with correctly, and the appropriate message should be sent even down to the minute milliseconds in which time is being compared to, and the return should be processed as soon as possible.

d. Test Fails if: Test is considered to fail if there are errors in comparing timings, as even international time should be considered and wrong messages are sent. Aside from this, the test is also considered to have failed if return processing was not completed correctly or the wrong amount is returned to the customer.

7. Test for SeatSelect Function:
    a. Description of Test: This test will ensure that while booking transaction is awaited, selected seats will remain booked to the visibility of others so as to prevent double booking, or errors in booking for people booking currently. The selected seats will remain booked for around 5 minutes and a timer will be displayed to customers attempting to complete the transaction to indicate when the seats will be lost and put up for booking again.
    b. Test Set/Vector(s): Working accurate digital timer, device connected to wifi.
    c. Output to be Expected: In this test, it will be expected for the device to start a 5 minute timer once customer clicks checkout and is redirected to payment info page. The selected seats should be shaded darker in color to all systems to indicate currently booked or under booking for other users attempting to book the same seats. Once the 5 minute timer is completed, the customer should be redirected towards the opening page and a message, “Sorry, transaction has timed out, please try again.” should be displayed on their device (PC, Laptop, or mobile device). Transaction page should be reset.
    d. Test Fails if: Timer starts before customer is redirected towards the payment information/confirmation page, or if timer is inaccurate and experiencing irregular changes, and customer is allowed more than 5 mins. Also, it is considered to have failed if the payment still goes through, but seats are not booked for the customer. Another fail category is if the seat is not marked as booked for other systems during the transaction process, and there occurs booking errors where two systems have the same tickets, or if the process is not completed for one of the users.

B. Testing for User Interface Functions:

8. Test for MovDisp Function:
    a. Description of Test: Test should provide a clean and understandable display of trending movies, and should also show all the closest showtimes as a preview. Also, this function should allow the user to search for existing movies and see where the shows are playing and what the showtimes are. Along with this, if the user searches for a movie which is not currently playing, a message should be displayed redirecting the user.
    b. Test Set/Vectors: Buttons and search bar, device connected to sufficient wifi and a display screen.
    c. Output to be Expected: All movies should be displayed clearly along with their showtimes upon request from the user depending on their search. Also, all

buttons and clickers should lead to corresponding pages related to them, such as if the checkout button is clicked, then it should lead the user to the checkout page. If the user searches for a movie which is not currently playing or does not exist, it should show them a blank page with the message, “Sorry, the movie you searched for is currently unavailable or does not exist.”

d. Test Fails if: Test is considered to be a fail if the search does not show the correct results upon searching for an existing movie, even if spelling is slightly off/casing is off, and if the above mentioned message is not displayed when a movie that is not available or does not exist is requested. Test is also considered to be a fail if the buttons do not click or lead to the correct corresponding pages when necessary.

9. Test for CustServ Function:
    a. Description of Test: This will ensure the chatbot works properly, although not giving a perfect answer to each specific question, using a list of keywords, it should redirect customers to the necessary page and or contacts in order to get their issue resolved.
    b. Test Set/Vectors: List of keywords, contacts, and pages to redirect to.
    c. Output to be Expected: When asked a variety of questions, some vague and some specific, the chatbot should pick up on keywords, and try to find assistance without the help of a person first. Once this is passed, or if chatbot is unable to trace keyword/synonym of keyword, it should display a message which simply redirects users to a link where they can submit their question, awaiting a response from a human assistant.
    d. Test Fails if: Test is considered to have failed if the chatbot fails to pick up on all the keywords and their synonyms and does not provide the correct assistance. It is considered to have failed if wrong responses are given to an unrelated question as well.

10. Security testing

    a. Description of Test: to check if the passwords and the username of the different users are protected and only the users who have the correct password and username can login
    b. Test Set/Vectors: need the usernames and passwords for the three different users admin, customer and the theater
    c. Output to be Expected: the expected output is that the password by using the password hashing and the user can only login when their username and their password is correct
    d. Test Fails if: this test fails if the user could login with a different password or username other than their saved password or username

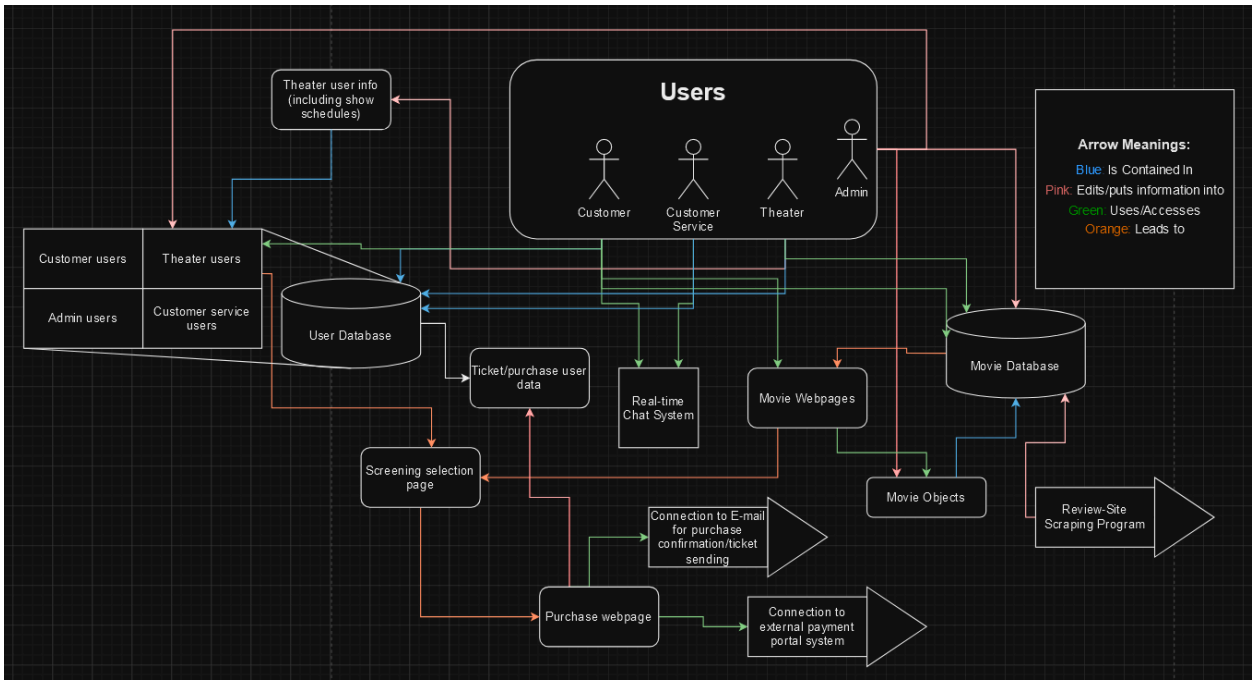
7. Test cases

Link to Test Case Excel Document:
SDSTestCases.xlsx

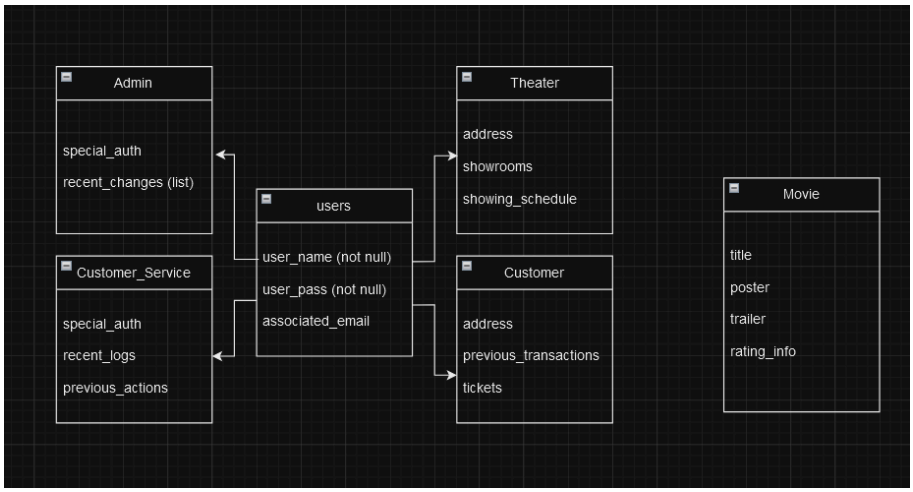
TEST CASES									
TestCaseId	**Component**	**Priority**	**Description/Test Summary**	**Pre-requisites**	**Test Steps**	**Expected Result**	**Actual Result**	**Status**	**Test Executed By:**
MovieSearch_1	Movie_Search_Module	P1	Verify that when a movie is searched for, the correct m	Site is already open	1. Search bar is clicked on site.	Movies relating to "titanic are displayed	Results with "Titanic are displaye	Pass	TesterRithish
AddMovie_1	Admin_Database_Edit	P1	Verify that Administrative users can add to movie data	Admin account logged in	1. Navigate to user page. 2. Click "Edit Movie List." 3. Click "Add." 4. In the provided fields, add a sample movie name, link to a trailer, poster image, and review site URL. 5. Click "submit." 6. Search in the site's movie search bar for the added movie's name.	Newly-added movie listing appears, and has an associated page with added information.	Newly-added movie listing appears, and has an associated page with added information.	Pass	TesterPotter
SeatSelect_1	Seat_Select_Booking	P4	Verify that seats can be selected without double booki	Movie should be chosen and should be on t	1. Movie and showtime must be selected. 2. 3 seats must be chosen at different places across the theater. 3. Another device to then attempt to pick the same seats.	The second device should not be able to pick those seats.	Seats were not able to have been	Pass	TesterRithish
AntiBot_1	Anti_botting_limitation	P4	Verify that per-user ticket sales are limited to prevent b	Generic customer account logged in, on ar	1. Click "Purchase ticket," select the first available location. 2. Proceed to payment, enter valid payment information. 3. Complete purchase. 4. Navigate back to same movie's page, attempt to repeat steps 1-4 at least 10 times.	User account is temporarily banned from purchasing before the 10 purchases can be completed.	User account is temporarily banned from purchasing before the 10 purchases can be completed.	Pass	TesterPotter
HotShow_1	Trending_Show_Display	P7	Verify that most trending shows are displayed on bar.	Site is already open.	1. Create 5 trending shows on a database not in ordered view of their ratings. 2. Click trending shows link on site 3. View the order of the trending shows on the moving bar and check if ratings are ordered.	Movies should descend on moving bar from order of most to le	Movies are displayed in correct o	Pass	TesterRithish
AcctCreate_1	Account_Creation_Doub	P4	Verify that account creation system behaves properly when one attempts to create an account with a non-unique username.	Site is open, no account currently logged in. Account exists named "newUser."	1. Click "sign up." 2. Enter new user information, using the name "newUser" and any password. 3. Submit and attempt to create account.	New account is not added to database, creating user gets message informing them of their already-existing username.	New account is not added to database, creating user gets message informing them of their already-existing username.	Pass	TesterPotter
AddMovie_2	Admin_Database_Edit	P4	Verify that admin database editing reacts appropriately	Site is open, admin account currently logged in. Movie exists in the database by the name of "newMovie."	1. Navigate to user page. 2. Click "Edit Movie List." 3. Click "Add." 4. In the provided fields, add a new movie with the name "newMovie." 5. Click "submit."	New movie is not added to database, editing user is notified of the existing movie and prompted to either cease editing or add a year to each movie's name to distinguish.	New movie is not added to database, editing user is notified of the existing movie and prompted to either cease editing or add a year to each movie's name to distinguish.	Pass	TesterPotter
CustServ_1	Customer_Service_Live_	P4	Verify that two customer service personnel cannot con	Two customer service accounts (A and B) and one customer account (C) are logged in on different devices.	1. User C selects "Chat with customer service" 2. A sees the open request and clicks to join immediately. 3. B also sees the open request and clicks to join after a brief delay.	C's device indicates that they are waiting for a representative. A successfully opens a chat connection, B is notified that the request is already being answered and does not.	C's device indicates that they are waiting for a representative. A successfully opens a chat connection, B is notified that the request is already being answered and does not.	Pass	TesterPotter
TheaterScheduling_1	Theater_Schedule_Edit	P3	Verify that schedule conflicts when theaters add movie showings are handled	Theater-type account is logged in and on it	1. Click "Edit Schedule" 2. Click "Add Showing." 3. Select any movie. Schedule it for any specific date. 4. Repeat steps 1-3 with a different movie and the same date.	Scheduling fails, user is notified of schedule conflict (specifically indicating the time and movie from the first schedule addition).	Scheduling fails, user is notified of schedule conflict (specifically indicating the time and movie from the first schedule addition).	Pass	TesterPotter

8. Architecture Design.

Architectural diagram:

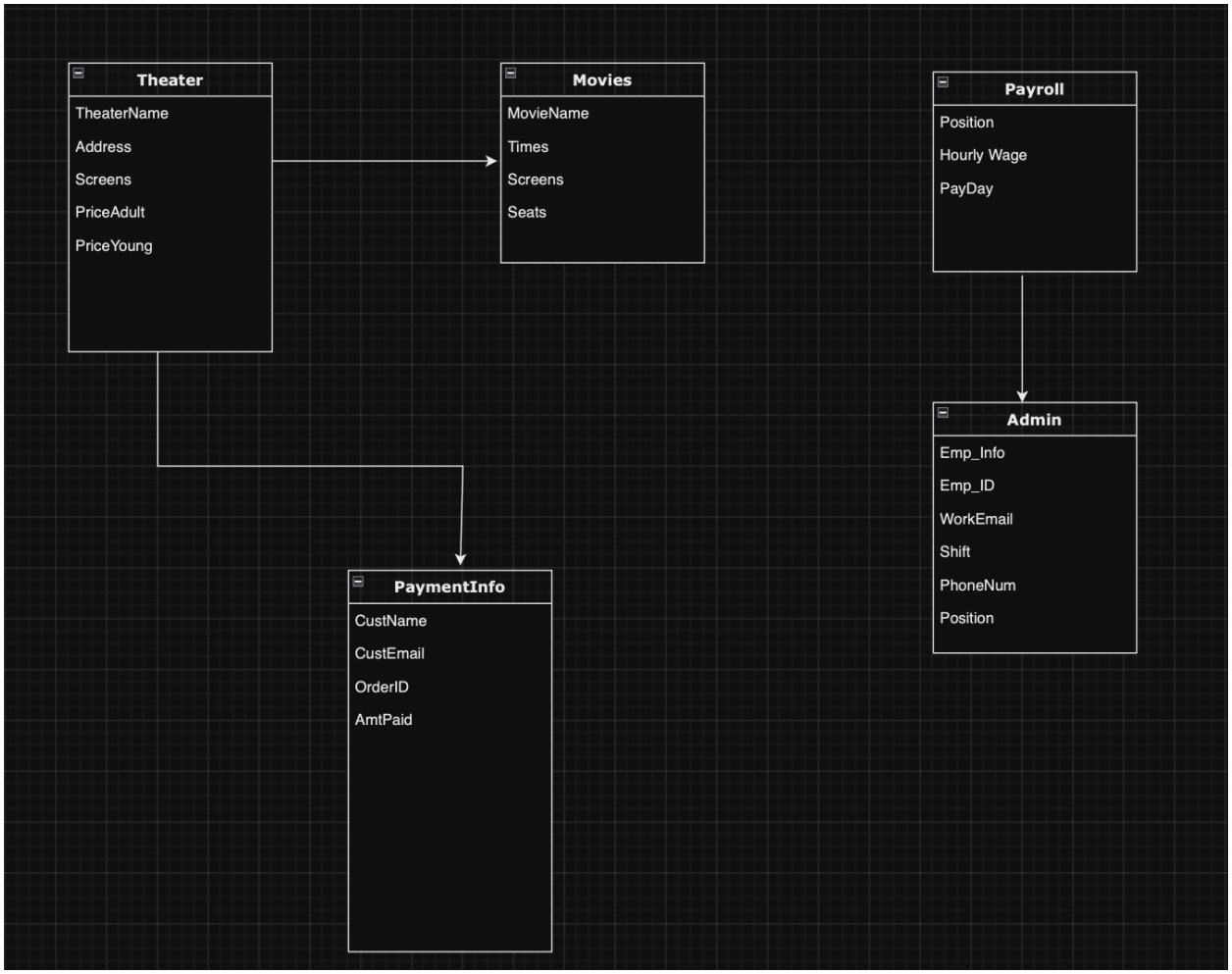


The original architectural diagram with the database section slightly updated. Changes reflect the various types of user stored by the database and specific accesses each type of user has to the database: customers can interact with the theater and movie databases through webpages to see showtimes and buy tickets, theaters alter their information and schedules by accessing the movie database, administrative users can access and alter data in both databases including viewing customer service logs, adding new theater users, and adding new movies or editing existing ones in the movie database.



A diagram of the relationship between various user types stored in the user database vs. the movie database

8.1 Data Management Strategy:



In our theater ticketing system, we have made use of an SQL Database. The reason for this is because SQL provides easy and powerful database management and manipulation techniques and has a method for relating databases together in order to have smoother operational

1. **Relational Database(SQL):** In our system, we make use of relational databases. This allows us to store multiple new fields from changing just one database due to the foreign key function. With relational databases we save time and resources in committing new information to multiple tables.

2. **Multiple Databases:** In our system, we are using a total of two different databases. One database will consist of all user end things which allow users to see and book movies, as well as payment related information. The other database will consist of Administration and management information to easier organize payment and work shifts between employees.

3. **Table Organization:** In order to make it easier to organize and understand various tables, each table will only consist of important fields which assist in the understanding of each table clearly. Aside from this, each table will consist of one field which is the foreign key to allow data linking.

4. **Foreign Key Linking:** We use multiple foreign keys in linking of each table in the database. Some tables are not fully linked within the entire database as well, to serve their functions better.

8.2 Tradeoff discussions:

Technology choice(SQL): For our data model we chose SQL for its ease of use and it has the table base database which has a predefined schema and more over SQL supports transactions that helps us in combining groups to interact with the database and SQL has the best community and the vendor support in the market

Single Database vs Multiple Database: our data model, We chose to use multiple databases as using it can lead to faster query times and with having different theaters which can have different price for a certain ticket or a show having multiple database is easier to maintain and update and it also acts as a backup servers if one failed in any situation

Table organization: The table organization includes all the data from the theaters , shows , movies and payments. Since we wanted to make our data easily readable we only wanted to include the main information which gets all the essential information

Possible alternatives: if the client wanted to scale their business we could considered using NOSQl option since it allows easy accessibility and also we could have opted for single database for the simplification of the data.