Final Report

for

Theater Booking System

Version 1.0

Prepared by Noah Coley, Cameron Fisher, Jacob Ilas, John Pita

Florida Polytechnic

04/24/2022

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this system is to allow customers to book a specific time frame in which they would like to watch a movie offered by the theater.
The capabilities of the software allow the user to select preferred seats, determine their intended form of payment, change or cancel booking, and the addition of including a friend to see your booking. The entire booking software will be described in the document.

1.2 System Overview

The capabilities of the software allow the user to select preferred seats, determine their intended form of payment, edit or cancel booking, and the addition of including a friend to see your booking. The admin is able to perform the same actions as the users as well as edit listings and provide direct booking refunds/cancellations.
The entire booking software will be described in the document.

1.3 Intended Audience and Reading Suggestions

The intended audience of this document is for the developers and document writers as well as the professor who's grading the assignment. The sections in the introduction should be a general overview of the project without much technical detail and the general overview is a bit more specific in the relevant sections. Sections 3 through 5 pertain to technical specifics. The information following section 5 will be mostly use case diagrams to visualize functionalities.

1.4 Product Scope

The project will allow interested customers to purchase a ticket to see a show offered by the theater, offering the ability to choose a prefered time of viewing, a preferred seating arrangement and the ability to change or cancel a booking based on the individual's circumstances.
The benefits that arise from a booking system stem from the presale capabilities of individuals who are highly anticipating to see a movie, allowing the establishment to make a profit before the movie releases. Additionally, it streamlines the process of purchasing a ticket and going to the relevant theater without having to wait in lines. It also makes planning easier for both the theater and the customers which improves the customer experience for both the theater's customers and the theater themselves.

2. Overall Description

2.1 Product Perspective

The product is a self-contained new project and this document encapsulates the whole project. The origin is simply an assignment in our Software Engineering course though a more traditional origin would simply be a request by a company, or a proposal we sent, to create a booking system for them.

2.2 Product Functions

- Allows customers to purchase tickets and reserve specific seats for specific movies showing in the theater.
- Allows customers to change their seats before the designated movie time.
- Allows administrators to add or remove movies from the list of movies showing in the theater.
- Allows the administrators to add or remove time slots for a specific movie.
- Allows the administrators to add or remove seats from a specific room in the theater.

2.3 User Classes and Characteristics

Administrators:

Administrators will have access to the full functionality of the system. They will be able to add and remove movies, adjust the times of movies, handle tickets and view transaction history. There are fewer administrators though their usage of the system will be more consistent and involved. They may require some light training in how to use the system to manage the movies, tickets, etc, but there is no need for prior specialty education or skills and the prompts and fields should be easy to follow.

Customers:

Customers will be able to purchase tickets and reserve seats through the product. There are no prior requirements for a customer to use the system. Customers will have the lowest overall access to the system. They simply need a method to pay for their tickets and while an individual customer may infrequently visit the site, the vast majority of users will be customers and there will be many users using the system at any given point.

2.4 Operating Environment

For this project the software will be created using Java/JavaFx in the IntelliJ IDE ver. 2021.1.3. The compiler will be javac and version 16 of Java will be used. The product will be run on Windows 11 Home, and should be independent from any other software as there are no dependencies.
In the event, we have time to implement a database for the prototype, we will likely utilize an SQL Database, likely SQLite as that is the technology we're most familiar with.

2.5 Design and Implementation Constraints

There are no hardware requirements nor limitations for the project, as is, due to the nature of the assignment. If this were a more traditional project, data storage and servers would be created in a cloud environment for easy scalability and reliability. Account systems, if implemented, will require salted hashes to be utilized to store sensitive information like passwords.

2.6 User Documentation

Per the final deliverable there is no requirement to create a user manual. It would be simple to create one that can elaborate on the basic functionality of the system, highlighting the purposes of each menu as well as how to deliver basic inputs into each required field. However, each button and prompt should be self-explanatory in the prototype.

2.7 Assumptions and Dependencies

The project can be impacted if the IDE being used, IntelliJ, is updated or discontinued which is unlikely as an update was recently posted. The project will use new code and have no dependencies from former projects.

3. External Interface Requirements

3.1 Hardware Interfaces

The client apps will be isolated to desktop/laptop uses, hardware requirements should be very basic and lightweight RAM/CPU functionality that comes with most computers. The server's hardware systems will be coded to run on a desktop computer for the sake of us not spending money on the project.

3.2 Software Interfaces

If time allows, a SQLite database will be designed and created to contain all information about the movies, times and seats available. A separate database should be used to hold user information such as hashed passwords, and emails. However the simple implementation that we may default to for the prototype will be utilizing a simple class structure to create and hold data, or to use some file format (e.g., csv). All of this should be usable with the java standard library.

3.3 Communications Interfaces

Ideally, the system being developed would be a web application utilizing HTTPS and other encryption methods to communicate with the server. However, given limitations on our knowledge, time, and budget we're creating a simple local application that communicates with the server. There must be a general TCP connection to the server on an open port. All communications involving sensitive information should be encrypted.

4. System Features

4.1 Purchase Tickets

4.1.1 Description and Priority

A customer focused feature that is of high priority. As the title implies, it is simply the main feature of the whole booking system; the feature where you actually book the ticket. Without the ability to purchase a ticket and book a seat the system cannot function.

4.1.2 Stimulus/Response Sequences

In the customer application after signing in there will be a button to browse the available movies. Once at that menu there should be a list of movies, their times and available seat number. The customer selects a movie and then is prompted with how many tickets they are purchasing and what seats they would like for each ticket. After this process is done the customer confirms their selection and is then brought to a window to input payment information. Another confirmation screen confirms the transaction and the whole process is done.

4.1.3 Functional Requirements

There has to be some UI created for about 5 different screens (exact number TBD). Additionally, an account system has to be used or a transaction ID has to be generated in the absence of an account system to log the transaction. The client needs to be able to request the movie information from the server. The client also needs to be able to encrypt or hash (exact protocols TBD) the sensitive information they are sending to the server.

REQ-1: Account system or some other method of tracking transactions
REQ-2: Encryption and hashing of data
REQ-3: Log in screen
REQ-4: Movie browser screen
REQ-5: Ticket count & seat selection screen
REQ-6: Payment screen
REQ-7: Confirmation screen(s)
REQ-8: Customer home screen

4.2 Change Seats

4.2.1 Description and Priority

A customer focused functionality that allows for a customer to change their seat selection after purchasing the ticket. It is the lowest priority of all functionalities due to it being a quality of life feature at best and is not completely required to provide a functioning system.

4.2.2 Stimulus/Response Sequences

If the user is logged in, the record of ticket purchases will be saved. Clicking a button to access this will show all transactions completed by the user. If tickets have been purchased for an upcoming movie they can select those tickets and change their seats in a selection screen. Clicking the confirmation button will then lead to a confirmation screen

4.2.3 Functional Requirements

There will be at least 3 separate screens created for this feature (exact count TBD). It simply needs to request transaction information for a specific user account from the server then display them. It should check for if a ticket is valid (the showing hasn't already happened or is happening). If valid, and the seats are changed then the updated request should be sent to the server which will verify if the seats are still available. If so a confirmation is sent, if not then the customer is warned that the seats are no longer available.

REQ-1: Account system
REQ-3: Log in screen
REQ-5: Ticket count & seat selection screen
REQ-7: Confirmation screen(s)
REQ-8: Customer home screen

4.3 Add/Delete Movies

4.3.1 Description and Priority

An administrator focused feature with a high priority. It gives admins the ability to add and remove movies from the available listings and without implementing the feature as software, there would need to be some prior knowledge of how the data is stored to create files or database entries. Overall, it reduces the barrier to entry and makes training as simple and straightforward as reading the prompts and filling in the respective fields.

4.3.2 Stimulus/Response Sequences

After the admin logs in, the list of movies will be displayed alongside their timings. A button on the side will display a popup showing the Add/Delete movie screen. The relevant information is filled out and submitted and the data is populated. If a selected movie is already removed there will be a warning popup.

4.3.3 Functional Requirements

At least 2 screens are required for this feature (exact number TBD). When adding, the data entered for the new movie will be sent to the server and the data will be populated on the client side. When deleting, it should check if there is a movie under the specific name, if so the selected entry would be removed from the server and the information would reflect on the client-side, if there is no movie to be removed a warning is sent displaying there are no entries to be deleted.

REQ-9: Admin main screen
REQ-10: Add/Delete movie screen
REQ-11: Server-side add/delete functionality
REQ-13: Admin application login

4.4 Update Movies

4.4.1 Description and Priority

A medium priority admin feature. In the event of a rescheduling, a mistake in listing the movie, or an error in the software, it allows for the admin to update the movie listings to be accurate.

4.4.2 Stimulus/Response Sequences

After the admin logs in and clicks the update button, the available list of upcoming movies will be listed. The admin can then select the movie and, similar to the add page, can update the fields of the movie and click confirm. Once confirmed the data will be propagated and the new listings will be available for customer viewing.

4.4.3 Functional Requirements

There will be at least three screens (exact count TBD). The data that's been entered into the fields in the update screen should be sent directly to the server which will update all relevant data stored, the propagated information will then be viewable for the client side. The admin screen will require that the admin enter login information prior to making any changes to movie listings, once the changes have been uploaded then the server will be updated. All the viewable information will be updated on the movie browsing screen, allowing the users to see new listing times or changes made to current offerings of screenings.

REQ-4: Movie browser screen
REQ-9: Admin main screen
REQ-12: Update movie screen
REQ-13: Admin application login
REQ-14: Server-side update functionality

4.5 Manage Tickets

4.5.1 Description and Priority

A low priority admin feature. It gives the admins the capability of monitoring, refunding, and/or canceling tickets. The system is capable of functioning without it, it is a quality of life feature that keeps track of customer purchases and tickets so that records can be properly maintained and any anomalies or refund requests can be dealt with accordingly.

4.5.2 Stimulus/Response Sequences

After logging into the admin application there is a button to manage tickets. Clicking that brings up a list of all transactions. The admin can search each ticket by either the user or the transaction ID. Transactions can be deleted, updated, or refunded. Notes can be added alongside each transaction by clicking the note button to keep records of any changes or anomalies.

4.5.3 Functional Requirements

At least 2 screens are required for this feature (exact number TBD). The admin client has to request transaction history and display a set amount per page (number TBD) so that the history of all transactions aren't queried at once. Depending on the format for storing data, notes can go into a table in the database, simply be stored as a string within a class, or be a field in the text file storing data. The status of the ticket should be stored and saved accordingly (e.g., "refunded", "expired", "purchased").

REQ-9: Admin main screen
REQ-14: Server-side update functionality
REQ-15: Transaction history screen
REQ-16: Search by transaction ID functionality
REQ-17: Search by user functionality
REQ-18: Note taking functionality

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Exception cases should be in place to catch invalid inputs by the user for any one of the menus. There must be a restriction on user input sizes to reduce the likelihood of overflow of memory.

5.2 Safety Requirements

Given the small stakes of the project, there are no potential damages while using the product beyond a customer potentially purchasing the wrong tickets. At worst, there is a small loss of money in refunding transactions due to user error.

5.3 Security Requirements

There must be hashing of credit/debit card entries into the system as a form of payment as well as hashed passwords for logging into accounts. Encryption will be used for user related information into the system such as name, email, phone number, etc. Additionally the program must encrypt any network traffic as a means to reduce the likelihood of eavesdropping user transactions.

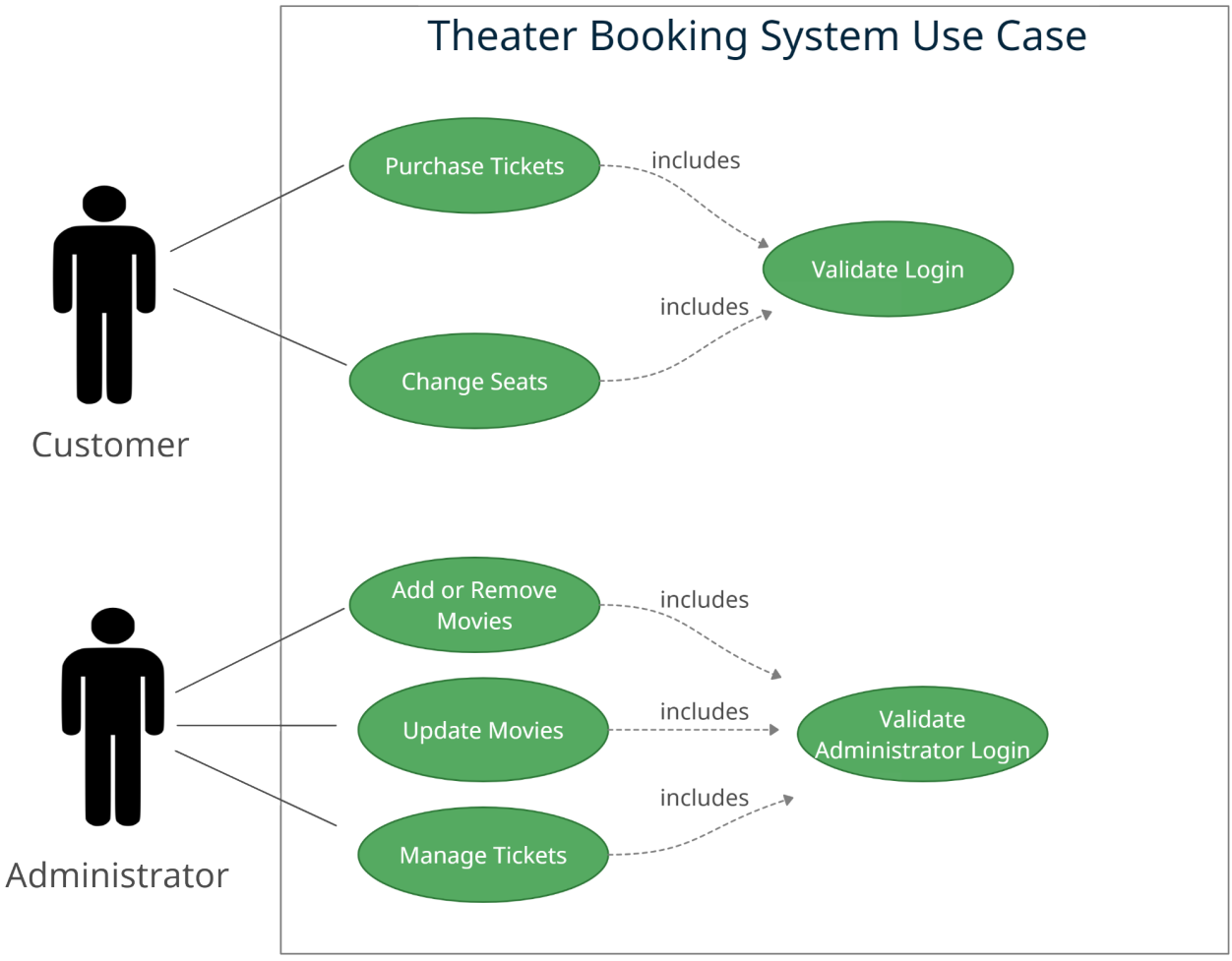
5.4 Software Quality Attributes

Ease of use and reliability are the foremost concern regarding quality attributes. Users should easily be able to follow prompts on each screen to book their movie and the administrator portion of the software should be just as easy to follow. Maintaining a reliable environment ensures that the theater's customers can access the booking system 24/7, maximizing the revenue for the customer and user satisfaction. Additionally guaranteeing the reliability and integrity of the data is vital for minimizing confusion for all parties involved.

6. Other Requirements

There are no other requirements outside what has been outlined in the document.

7. Analysis Models



Use Case Number	1
Use Case Name	Purchase Tickets
Brief Description	A customer purchasing tickets for a certain movie
Precondition(s)	A customer has a valid login.
Successful Post Condition	The tickets are added to the customer's account and the seats are reserved.
Actors	Customer
Priority	High
Related Use Cases	N/A
Flow of Events	
Main Flow	
8. The Customer selects the "Purchase Tickets" option on the home screen. 9. The Customer selects the movie they are purchasing tickets for 10. The Customer selects their time slot and seats. 11. The Customer confirms their choice and validates their login. 12. The choice is confirmed and saved.	
Alternate/Exceptional Flows	
4a. The login is invalid and the request is rejected until a valid login is used.	

Use Case Number	2
Use Case Name	Change Seats
Brief Description	A customer changing their seats for their previously purchased tickets
Precondition(s)	A customer has a valid login and a valid ticket(s).
Successful Post Condition	The new seats are reserved and the previous seats are unreserved.
Actors	Customer
Priority	Low
Related Use Cases	N/A
Flow of Events	
Main Flow	
1. The Customer selects an owned ticket. 2. The Customer selects the "Change Seats" option. 3. The Customer selects their new seats. 4. The Customer confirms their choice and validates their login. 5. The choice is confirmed and saved.	
Alternate/Exceptional Flows	
4a. The login is invalid and the request is rejected until a valid login is used.	

Use Case Number	3
Use Case Name	Add or Remove Movies
Brief Description	An administrator adding or removing a specific movie.
Precondition(s)	An administrator has a valid login.
Successful Post Condition	The specific movie is either added or removed and the administrator is returned to a home screen.
Actors	Administrator
Priority	High
Related Use Cases	N/A
Flow of Events	
Main Flow	
6. The Administrator selects the "Add or Remove Movies" option on the home screen. 7. The Administrator selects whether they are adding or removing a movie. 8. The Administrator types the name of the movie they are adding/removing. If the Administrator is removing a movie, the name is validated against the database of current movies. 9. The Administrator confirms their choice and validates their administrator login. 10. The choice is confirmed and saved.	
Alternate/Exceptional Flows	
3a. The movie the administrator attempts to remove is not in the database and the request is rejected. 4a. The login is invalid and the request is rejected until a valid login is used.	

Use Case Number	4
Use Case Name	Update Movies
Brief Description	An administrator adding, changing or removing time slots from a specific movie showing in the theater.
Precondition(s)	An administrator has a valid login.
Successful Post Condition	The specific time slot operation is carried out and the administrator is returned to a home screen.
Actors	Administrator
Priority	Medium
Related Use Cases	N/A
Flow of Events	
Main Flow	
11. The Administrator selects the "Manage Movies" option on the home screen. 12. The Administrator selects which movie they are editing time slots for. 13. The Administrator decides to either add a time slot, edit an existing time slot or remove an existing time slot 14. The Administrator adds the necessary information for the selected option. 15. The Administrator confirms their choice and validates their administrator login.	

16. The choice is confirmed and saved.
Alternate/Exceptional Flows
5a. The login is invalid and the request is rejected until a valid login is used.

Use Case Number	5
Use Case Name	Manage Tickets
Brief Description	An administrator managing tickets.
Precondition(s)	An administrator has a valid login.
Successful Post Condition	The specific ticket operation is carried out and the administrator is returned to a home screen.
Actors	Administrator
Priority	Low
Related Use Cases	N/A
Flow of Events	
Main Flow	
17. The Administrator selects the “Manage Tickets” option on the home screen. 18. The Administrator selects a specific ticket. 19. The Administrator selects whether they would like to refund, cancel or monitor the ticket. 20. The Administrator confirms their choice and validates their administrator login. 21. The choice is confirmed and saved.	
Alternate/Exceptional Flows	
4a. The login is invalid and the request is rejected until a valid login is used.	

8. Design Considerations

8.1 Assumptions

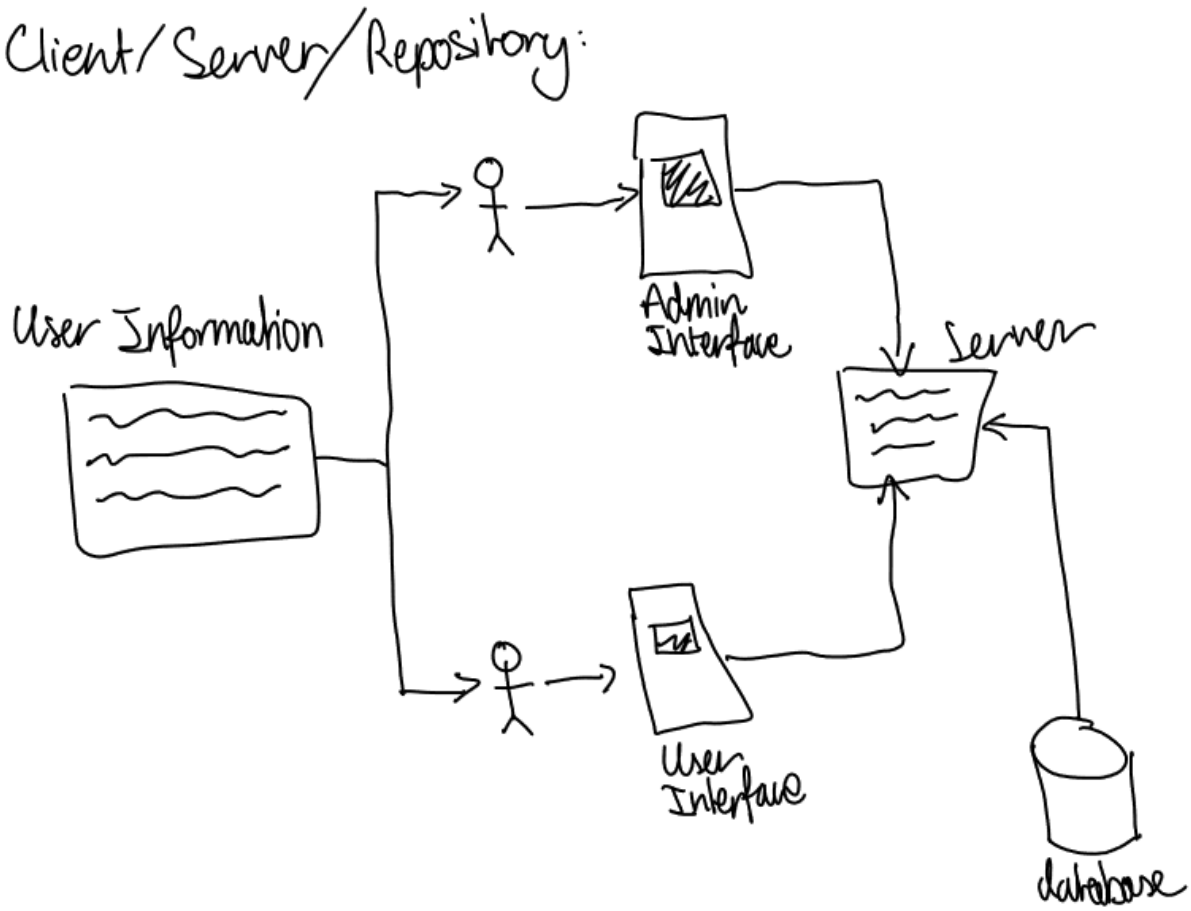
In a proper implementation of the project, given enough time, resources and knowledge, the only running assumption would be that end users have an internet connection and that their operating systems would be some relatively modern OS. Due to limits on time, resources and knowledge, though, we're creating local applications that do not require an internet connection and will run on one computer. This means the only assumption for utilizing and designing the prototype will be a modern OS, i.e, Windows so that the applications created can function.

8.2 General Constraints

For a complete project, end-users should be connected to the internet and can access the system with any modern device capable of using a browser. The customer's on-premises or cloud resources for the server should be able to handle all traffic from customers. A cloud server and database would be more preferred for flexibility and less concern for hardware constraints. The constraint then becomes a budgetary one in order to meet demand.

The database's required storage likely would not exceed a few terabytes unless the customer was a large chain with millions of customers; therefore there is little constraint in regards to data storage. Any personal information of a customer should be encrypted or hashed for security purposes. For the purposes of the prototype though, no internet connection is required and the end-users must be on a computer.

8.3 System Architecture



Client/Server & Repository

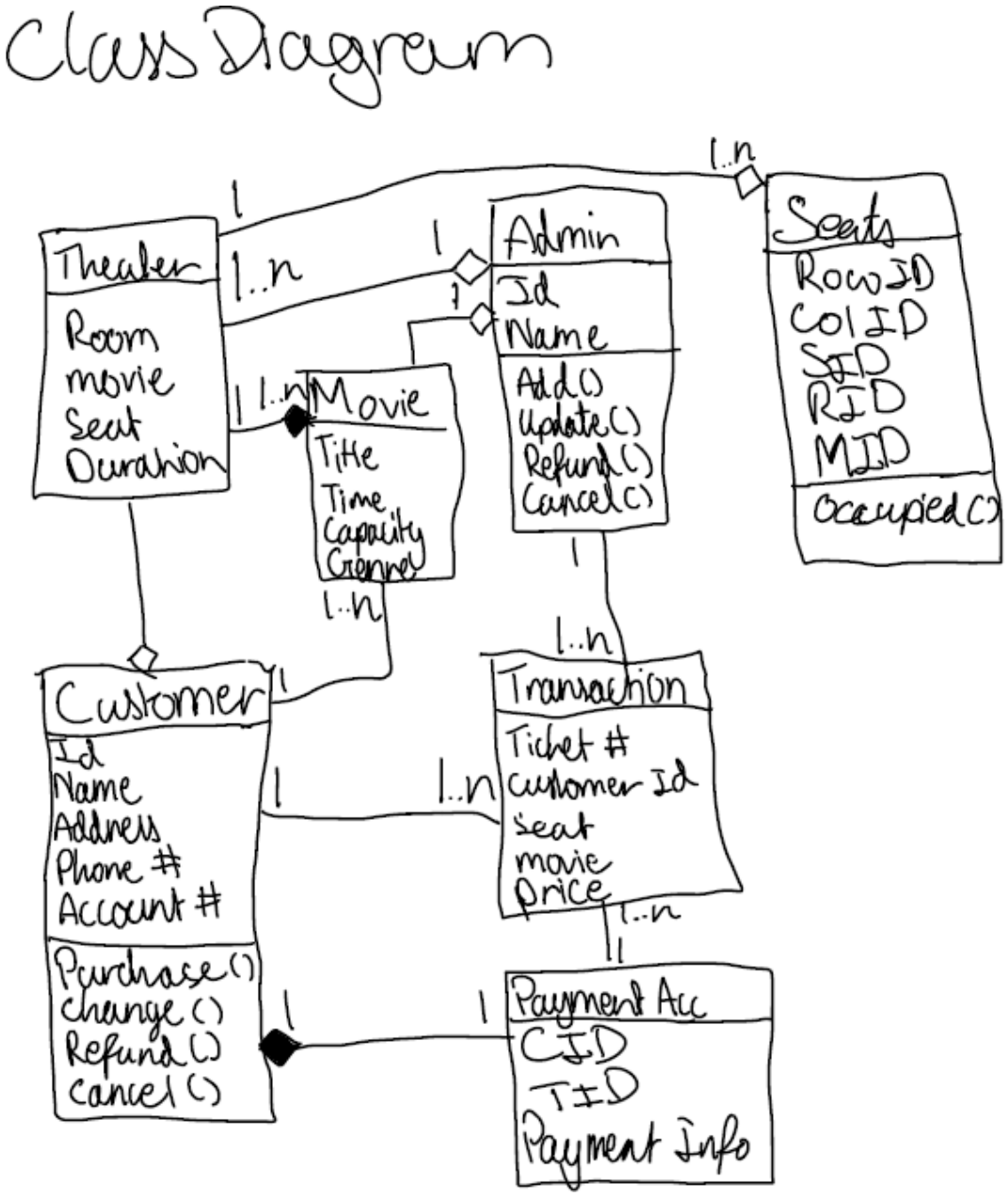
Repository houses information related to the movies being offered by the theater. Title of the movie, the duration of the movie, any additional lookup options on genre etc. Additionally, transactions where the customer purchases a ticket and books a seat will be stored in the database.

Repository will also store customer related information for their accounts, such as login information as well as any stored payment methods. The admin account information will similarly be stored so that admins can login and access the repository.

Client/Server will allow interested customers to connect to the application in order to book their tickets for the theater. Customers will be able to access the server through an interface that will allow them to make their proper seating accommodations for the show based off of information that they have provided. Admins will use their interface to update, add, and remove movies, and handle customer inquiries regarding purchases and refunds.

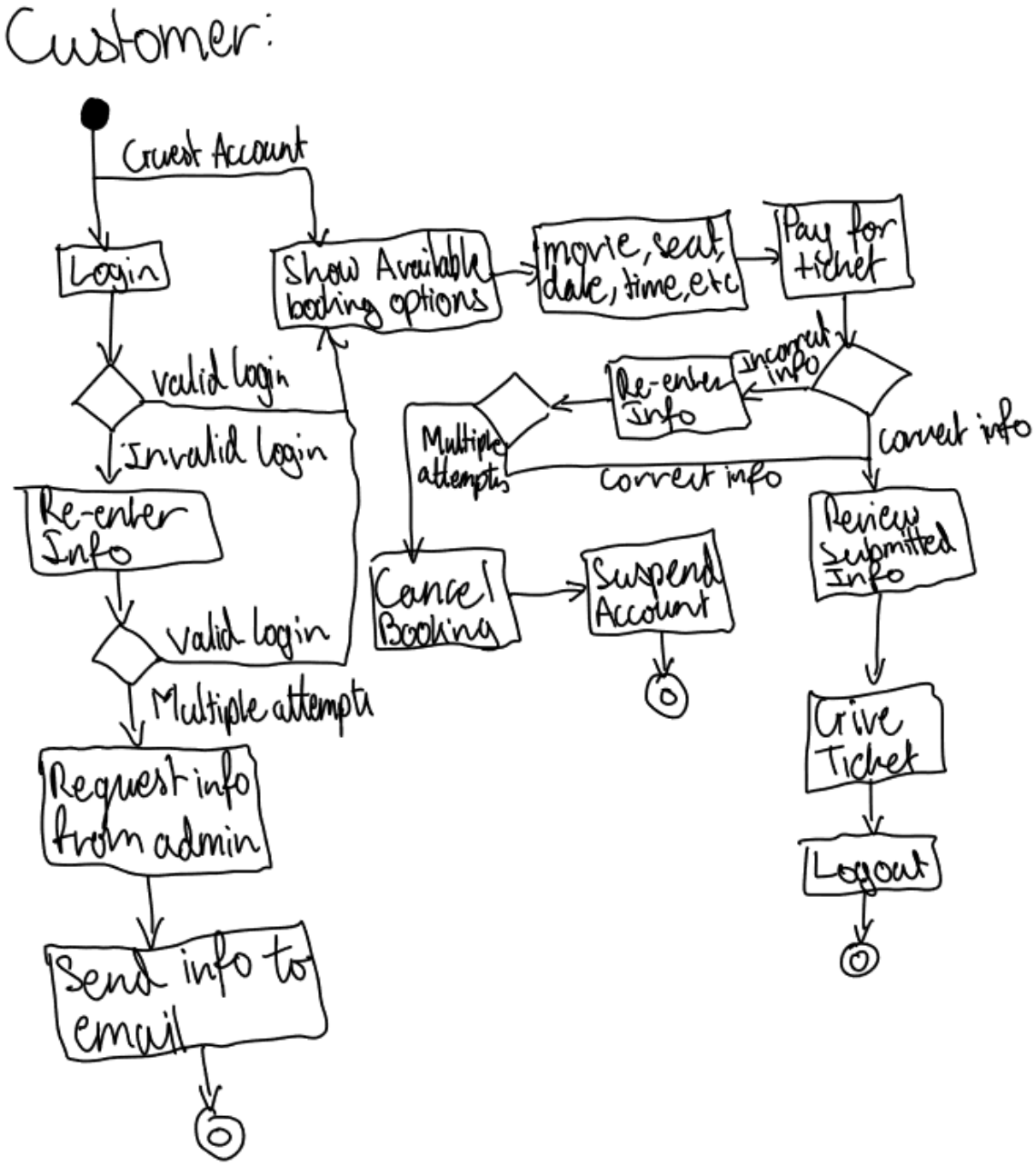
9. Detailed System Design

9.1 Class Diagram



Theater, Movie, Transaction, and Payment Account classes only hold data for their respective object. The Customer and Admin classes handle the majority of the functionality of the applications. This allows for easier isolation of admin and customer functionality when creating the respective applications.

9.2 Activity Diagram



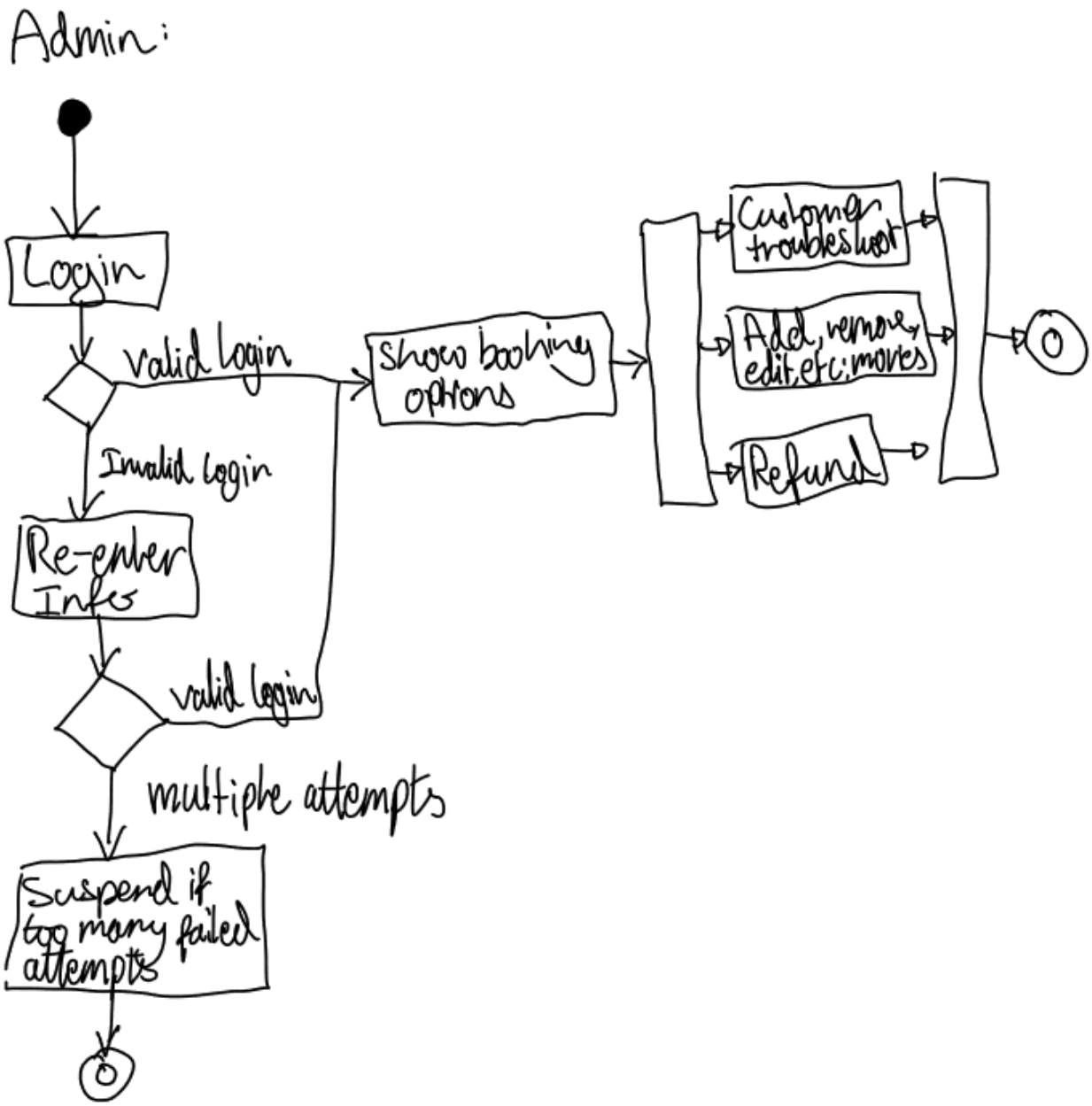
For the customer class, the user must first verify login information if they have an account with the service, if not they can use a guest login (loses the functionality of being able to store information to make the check out process faster).

If the input for the login is invalid, the user must re-enter information, if the user exceeds the incorrect login limit, their account will be disabled and the admin will have to re-enable their account.

Once logged in the user will be able to view booking options for the viewings that are being offered, from that stage they can choose a seating arrangement, the day and time that they wish to see the movie, etc.

If they wish to purchase a ticket, they must verify their banking information in order to complete the transaction.

If the input for the ticket purchase is incorrect, the user must re-enter their banking information, if the user exceeds the attempts limit, their account will be suspended regardless of account standing, admin must re-verify account if they have a registered account and if they are a guest they will need to try again at a later time.



For the admin class, the user must first verify login information.

If the input for the login information is invalid, the user must re-enter information, if the user exceeds the incorrect login limit, their account will be disabled, another admin will need to re-enable the account.

Once logged in the admin will be able to view the possible booking options being offered to customers.

From the booking options they will be able to add or remove showings, be able to assist customers that are having account issues, as well as issue refunds or ticket changes for customers upon request.

10. Algorithms for Components/Methods

```
public boolean validateCustomer(nCID, nPassword){
        for loop i through Customer table size{
                if nCID == Customer CID[i] AND nPassword == Customer password[i] {
                        return true
                }
        }
        return false
}

public void bookTickets(CID, MID){
        initialize  seatsW
        output “Which seat(s) would you like: ”
        seatsW = input
        for loop i in seatsW size{
                while(!validateSeat(seatsW[i], MID){
                        output “Seat “ + “seatW[i] + “ is invalid. Chose another”
                        seatsW[i] = input
                }
        }
        get and process payment information
        create a Transaction entry with the given CID, MID, the movie’s price from the MID, and
        the seat for each seat in seatW
        add the entries to the Transaction table if not already done in the previous step
}

public boolean validateSeat(seat, MID){
        for loop i through Admin table size{
                if MID == Seats MID[i] AND seat[0]==Movie rowID[i] AND seat[1] == Movie
                columnID[i] AND Movie occupied[i] == false {
                        return true
                }
        }
        return false
```

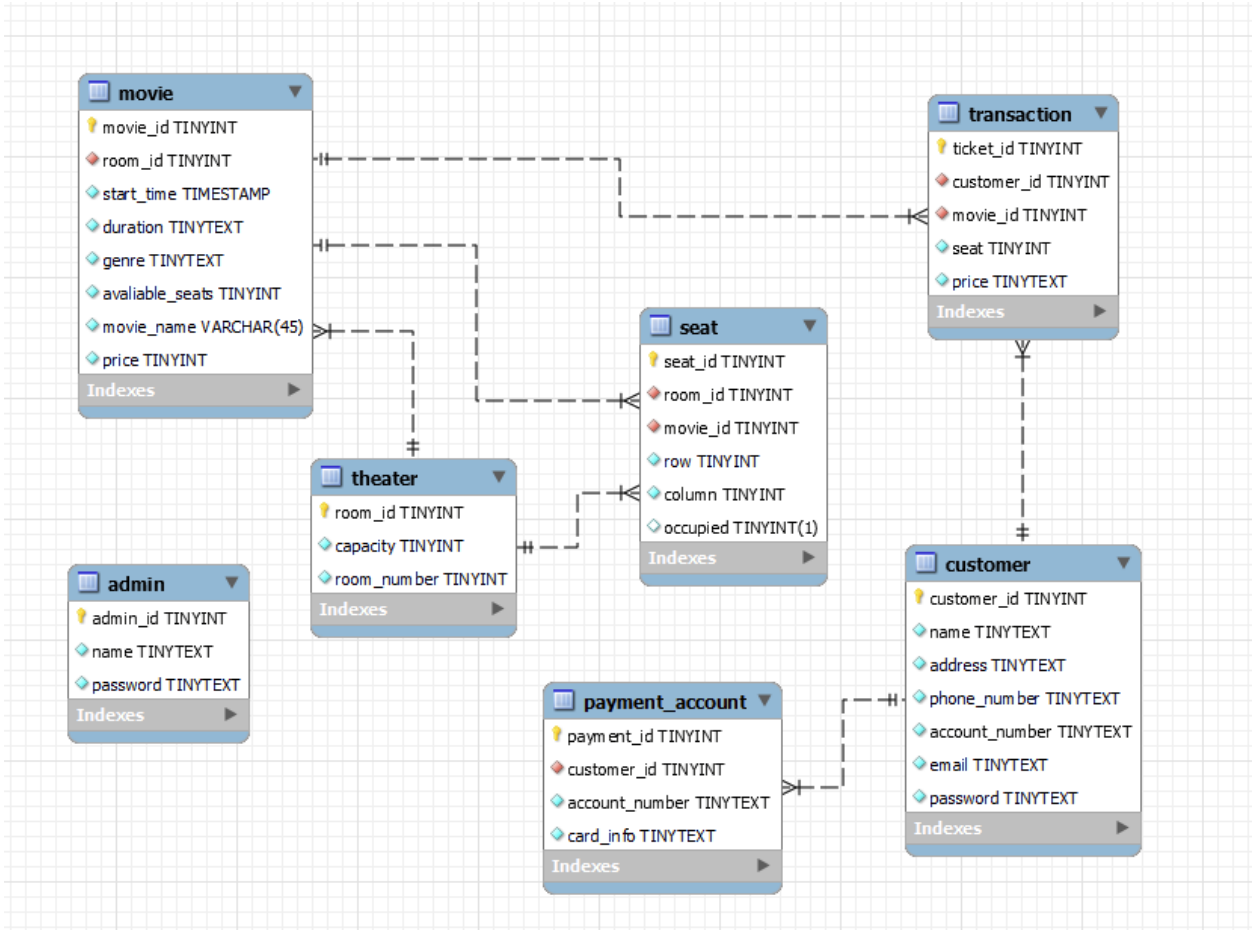
```
}

public boolean validateAdmin(nAID, nPassword){
        for loop i through Admin table size{
                if nAID == Admin AID[i] AND nPassword == Admin password[i] {
                        return true
                }
        }
        return false
}

public void addMovie(MID, time, duration, genre, seats, capacity){
        create a Movie table entry with the respective fields
        add the entry to the table (if it was not done in the previous step)
}
public void updateMovie(MID){
        for loop i through Movie table size{
                if MID == Movie MID[i]{
                        carry out the update
                        return
                }
        }
}
public void removeMovie(MID){
        for loop i through Movie table size{
                if MID == Movie MID[i]{
                        remove that entry from the Movie table
                        return
                }
        }
}

public void resetPassword(nCID, nPassword){
        for loop i through Customer table size {
                if nCID == Customer CID[i] {
                        password[i] = nPassword
                }
        }
}
```

11. Database Design



Customer Table

- customer_id
- name
- address
- phone number
- account number
- email
- password

Table for storing customer information. Personal information such as emails, and phone numbers should be encrypted while passwords should be hashed.

Admin Table

- admin_id
- name
- password

Basic account information for admin accounts is stored so that they can sign in.

Movie Table

- movie_id

- Time
- Duration
- Genre
- Capacity (available number)
- Price

The data regarding a movie. The seats entry links to a table storing information about various available seats. The capacity field states the available seat count.

Transaction Table

- ticket
- movie_id
- seat
- price
- customer_id

A table for storing information about customer transactions.

Seats Table

- seat
- movie_id
- room_id
- Row
- Column
- Occupied

A table for seats, both available and occupied for each movie, in their respective room.

Payment Acc

- payment_id
- customer_id
- Account #
- Card Info

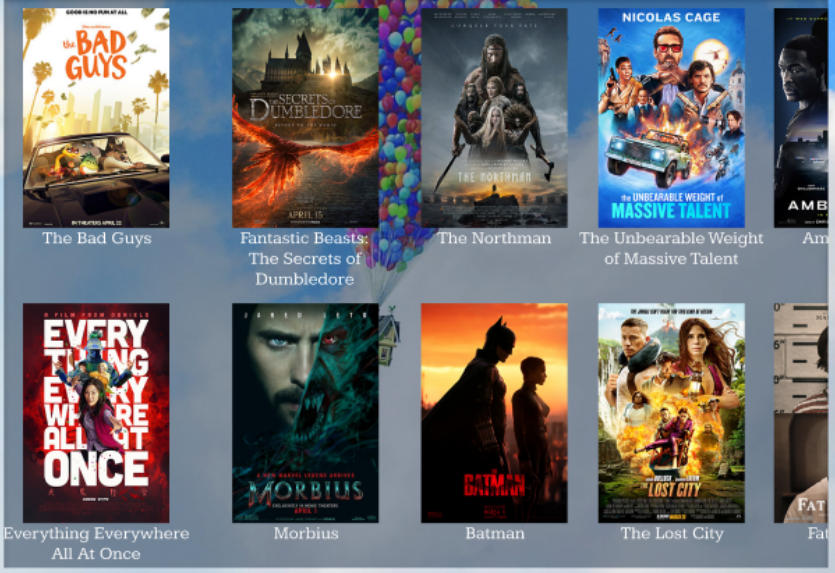
A table for storing payment information related to the customers. Largely useless in the prototype, but in a more fleshed out implementation would require frequent use.

12. User Interface Design

The two user applications open with a login screen that prompts the user to enter their credentials. The customer application allows for the user to click the guest sign in button to purchase tickets or browser available movies without needing to sign in. A guest account however lacks the ability to store purchase history or quickly access their payment information.

Account

Current Movies



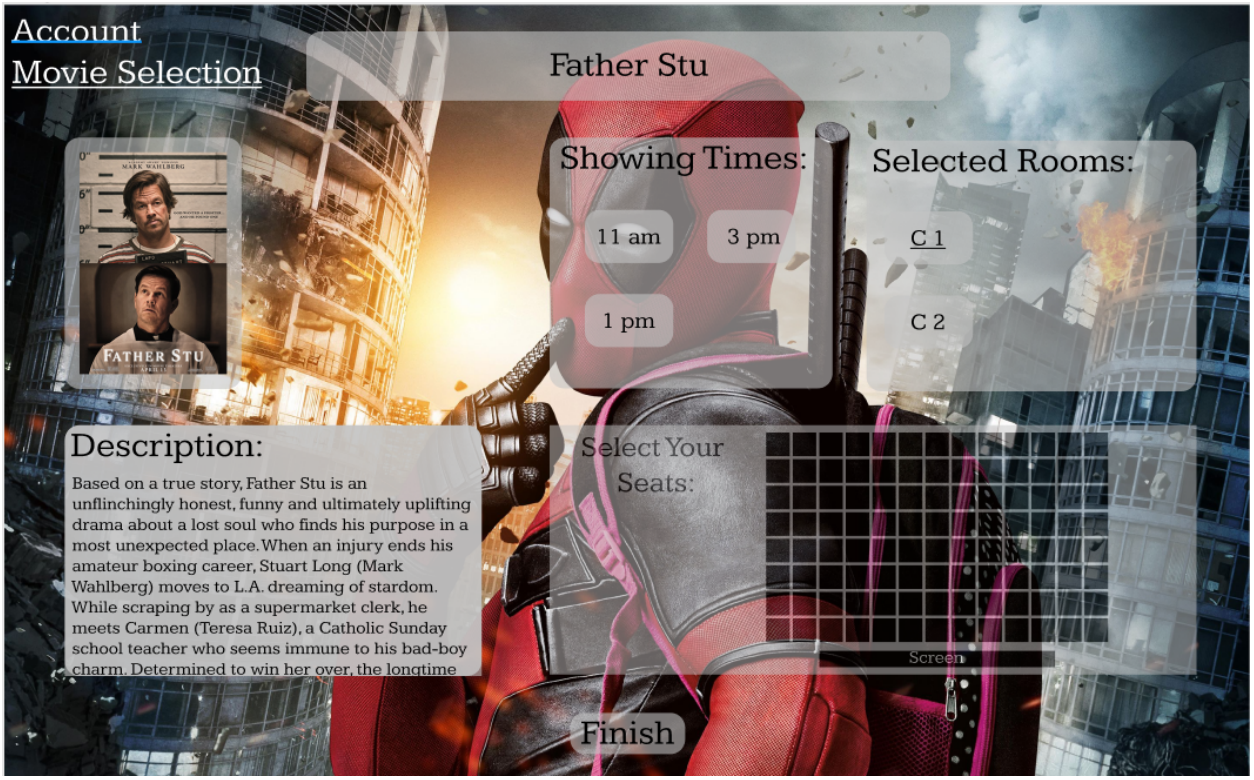
The Bad Guys | Fantastic Beasts: The Secrets of Dumbledore | The Northman | The Unbearable Weight of Massive Talent | Am

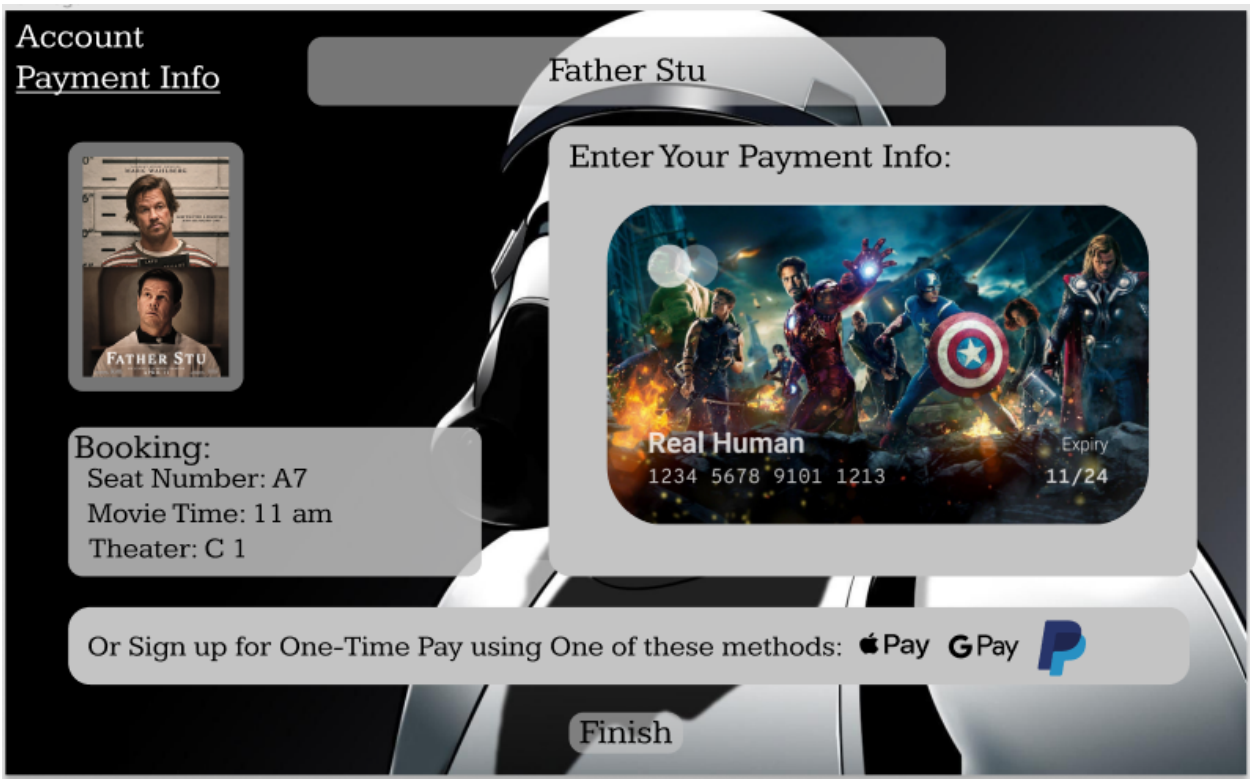
Everything Everywhere All At Once | Morbius | Batman | The Lost City | Fa

Featured:

- Most Popular
- New Releases
- Recently Added
- Leaving Soon
- Animated
- Action
- Anime
- Classics
- Comedy
- Drama
- Romance

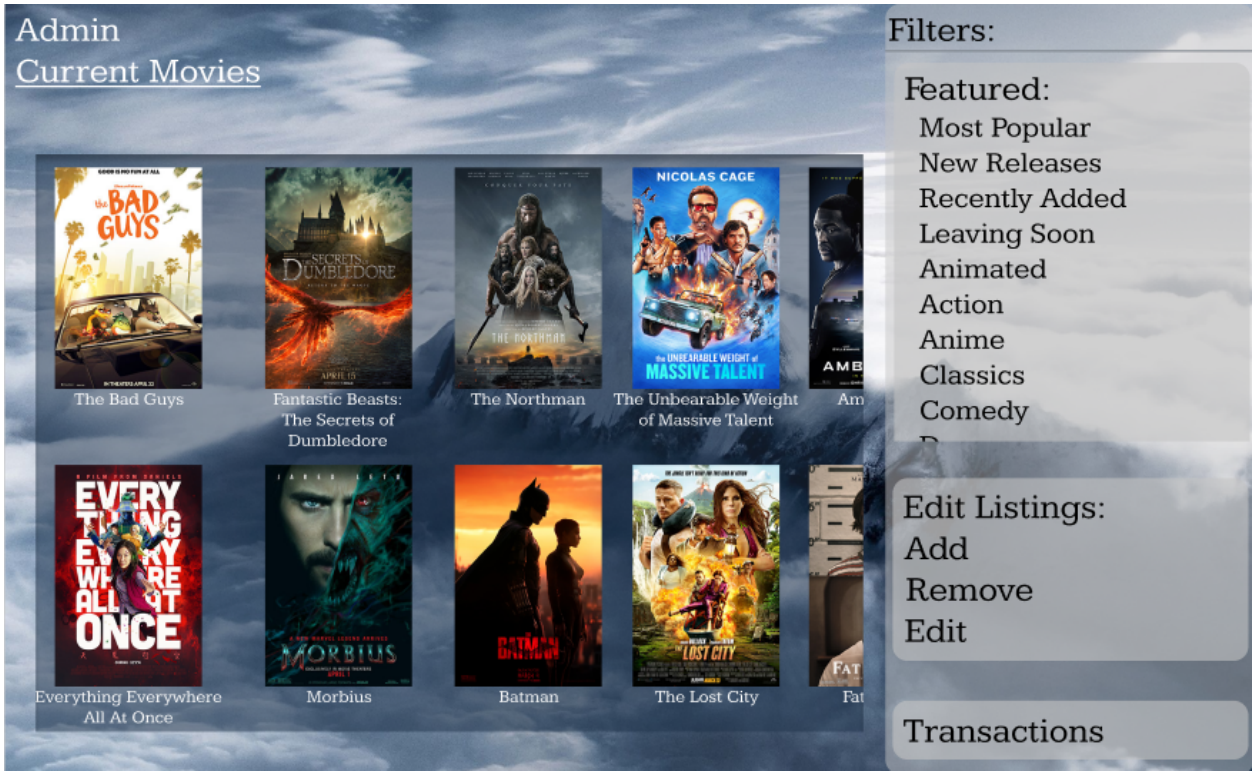
Start Booking!





Once signed in, the main screens pop up that allows the users to select which functionality they need to access. Customers can check their purchase history and request refunds prior to the movie's screening, or browse the list of upcoming movies. Browsing movies

brings up a list of available movies with an attached “book” button. Selecting the book button then brings up purchasing screens requesting payment information, what seats the customer wants, how many tickets, etc.



Admin

Add Movies

Insert Movie Title

Insert
+
Image

Showing Times:

Description:

Selected Rooms:

Finish

Admin

Remove Movies

Father Stu

Showing Times:

11 am ⊖ 3 pm ⊖

1 pm ⊖

Description:

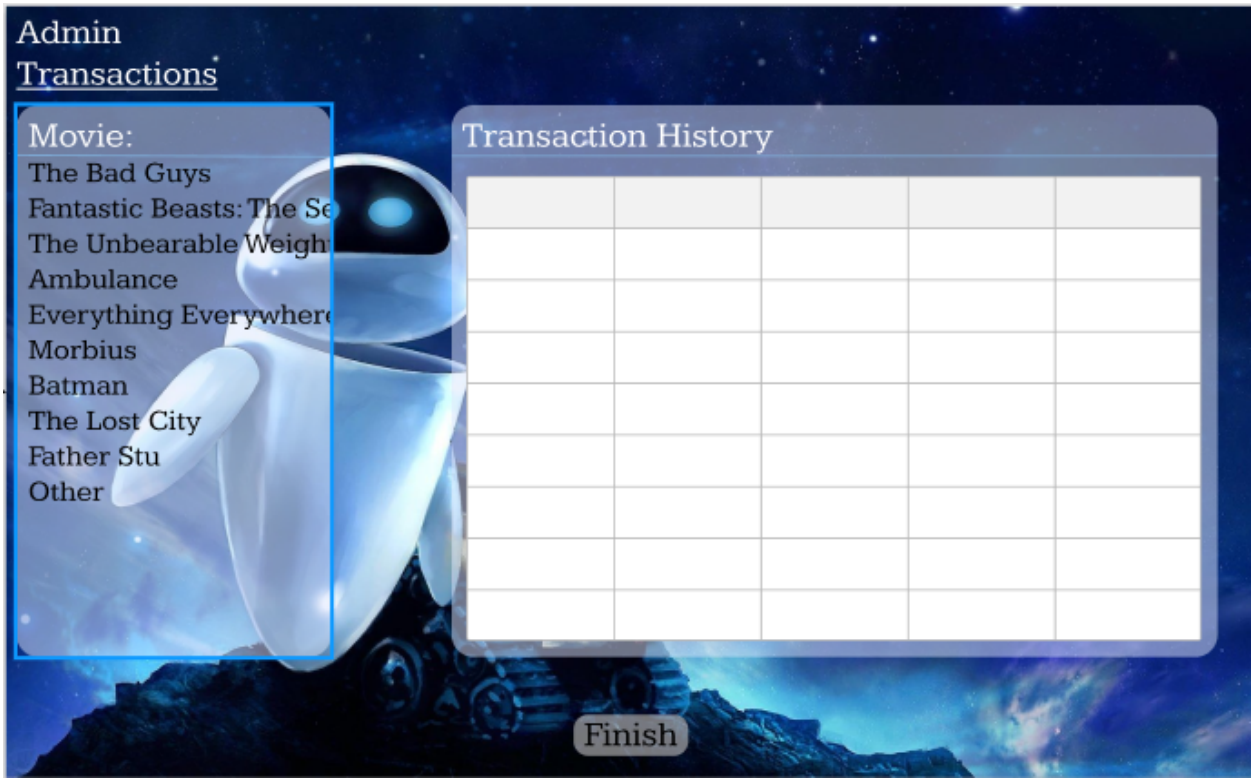
Based on a true story, Father Stu is an unflinchingly honest, funny and ultimately uplifting drama about a lost soul who finds his purpose in a most unexpected place. When an injury ends his amateur boxing career, Stuart Long (Mark Wahlberg) moves to L.A. dreaming of stardom. While scraping by as a supermarket clerk, he meets Carmen (Teresa Ruiz), a Catholic Sunday school teacher who seems immune to his bad-boy charm. Determined to win her over, the longtime

Selected Rooms:

C 1 ⊖

C 2 ⊖

Finish



Admins have multiple buttons to access different pieces of the data they need to manage. There's a button related to movie management for updating the list of movies, a button for accessing transaction history and a button for account management. Refunds, password resets and movie

management is done through buttons attached to each entry on their respective tables. Data filters and search bars can make parsing through the data easier.

12.1 General Outline

Customer Application

- Login Screen
    - Username field
    - Password field
    - Login button
    - Guest sign in button
- Main Screen
    - Movie List
        - Book seat button
            - Purchase screens
    - Purchase history button
        - Table of purchases (if signed in)

Admin Application

- Login Screen
    - Username field
    - Password field
    - Login button
- Main Screen
    - Update Movie List button
        - Movies table
        - Add button
        - Remove button
        - Update button
            - Update window with fields pop up
    - Transactions button
        - Transactions table
        - Search bar
        - Refund button
    - Customer Accounts button
        - Customer accounts table
        - Reset password

13. Testing

Database Unit Testing

The data inputted into the database has to follow a specific format in order to be properly processed by the software, which prevents users from inputting any out of ordinary values into areas they shouldn't go.

Data fields that require integer input have restrictions that are put on the data values from mysql require that the data inputs fall within a number between 0 and 255, cannot include special characters or the use of characters.

Data fields that require text input have restrictions that are put on the data values from mysql require that the data inputs fall within 255 characters, can include special characters and numbers.

Data fields that require a timestamp must follow the format of YYYY-MM-DD HH:MM:SS or else the data input will not be accepted by the software; throwing an exception.

Database Integration Testing

Once all of the foreign keys were situated between the tables, the system works as a whole. Each data field that is required for another table is properly formatted to match the intended value for the next table it is calling for, if any values are improperly inputted or misconfigured the system won't allow the generation of the data values into the listing.