Software Requirements Specification
for
Cinema E-Booking System

Version 1.0 approved

Prepared by Team 8

Brogan Young, Gustavo Gomez, Chris Callahan, and Sergio Rossi

CSCI 4050 Class B

04/25/2018

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Brogan Young	03/24/18	Updating Hardware Interface	1.0

1. Introduction

1.1 Purpose

This SRS describes the functional and non-functional requirements for the software for the initial release of the Cinema E-Booking System. This document is intended to be used by team members implementing and correcting the system and its functions. All requirements specified here are of the highest priority unless otherwise specified, and will be implemented with the initial release of the software.

1.2 Document Conventions

1.2.1 Topics are Times New Roman formatted in 'Heading 1' style. The font size for headings is 18.
1.2.2 Subtopics are Times New Roman formatted in 'Heading 2' style. The font size for headings is 14.
1.2.3 Text is Times New Roman formatted in 'Normal text' style. The font size for text is 12.
1.2.4 Italics have been used for laying greater emphasis on certain information throughout the document.
1.2.5 All references to websites used are hyperlinked in Times New Roman with normal text and font 12.

1.3 Intended Audience and Reading Suggestions

The document is intended for customers, administrators, and managers to login successfully through the applications login page and access the Cinema E-Booking System. The rest of the SRS contains the systems functionality, requirements and general interface of the project. The sequence for reading the document would be to begin with the overview sections and proceed through the sections in order.

1.4 Product Scope

The software being specified here is an online cinema booking and ticket purchasing website. It allows users to go to a website, search for the movie they want to watch by one of many means (by movie title, movie genre, year released, actors/actresses, director/producers, rating, and showing time). Then, the customer is able to buy a ticket for the specified movie at any available hall, seat, and/or viewing time. Customers are also able to request refunds, create accounts, update their accounts, and change the date/time they wish to view the movie. System administrators will be able to suspend accounts, edit movie information (such as viewing date, hall number, viewing time, title, actors/actresses, directors/producers, rating, description, and

film rating), create and add promotional offers, and even delete movies from their viewing hall. This software is intended for use by anyone who wishes to book or view movies online, specifically for the CSCI 4050 Class B class.

1.5 References

1. Website draft idea: www.pvrcinemas.com
2. Database guide: Drkušić, Emil. "How to Design a Database Model for a Movie Theater Reservation System." *Data Design in Practical Examples*, Vertabelo , 3 Mar. 2016, 26 How to Design a Database Model for a Movie Theater Reservation …

2. Similar existing systems

Fandango is a system that is similar in concept to the Cinema E-Booking System we are putting in place. Fandango allows users to search for movies and book tickets for your local theater for any of the available showings. A major difference between the two is that our Cinema E-Booking System will allow users to also select the seat for the movie they wish to view, and only shows the showings of a single movie theater. Also, our Cinema E-Booking System will not have package deals for extra items, such as bundles that include a shirt or a cd in addition to your ticket.

3. Proposed System

3.1 Product Description

The product has a few different factors, including the customer and what they are searching for. Do they need a movie title, a genre of movies, a list of movies by producer? Does the website have all this information easily accessible to the customer? Can the customer purchase the tickets in advance? Can they choose an individual seat in the theater? Some of the requirements that need to be implemented include a search functionality, a ticket ordering functionality, a ticket date search functionality, and a payment and account functionality.

3.2 Functional Requirements

3.2.1 The System shall allow the system administrator to add/edit/delete movie information such as the title, genre, cast list, director(s), producer(s), ratings, reviews, and trailers

3.2.2 The System shall allow the system administrator to add/edit information for booking tickets such as seat location, seating availability, and viewing times and dates

3.2.3 The System shall allow the system administrator to add search functionality so users can search for movies

3.2.4 The System shall allow the system administrator to add/edit ticket information including ticket price

3.2.5 The System shall allow the system administrator to suspend or delete accounts based on the account's activity

3.2.6 The System shall allow the system administrator to send promotional offers to users through the email address the users provide when they create their account

3.2.7 The System shall verify user's login, so that system administrators can have different permissions allowed on their accounts

3.2.8 The System shall have a create account option, so users can create an account to book tickets

3.2.9 The System shall have a verification email sent to a user who is registering their account or editing their information on their account. This email will contain a code that will verify that the user associated with the account is requesting this action

3.2.10 The System shall have a login system, so users can login to their account

3.2.11 The System shall have an option to view the layout of a hall, so they can see which seats are available and which seats they wish to choose

3.2.12 The System shall allow a customer to book a ticket for a seat for a movie they choose. This includes the seat placement, how many tickets, what type of ticket, which viewing hall, what time the viewing is, what date the viewing is, and how much the ticket will cost

3.2.13 The System shall have a checkout page that will allow a customer to checkout and pay for the tickets they wish to book

3.2.14 The System shall send a confirmation email to confirm the purchase a customer has made and to give the customer a copy of the receipt sent to their email

3.2.15 The System shall allow customers to enter in a promotional offer to get special discounts for their tickets

3.2.16 The System shall allow the customer to sign up for a subscription for promotional offers and news to be sent to their email address, and has an option to unsubscribe from these emails

3.2.17 The System shall allow a customer to view/edit their account and will send a verification email to make sure the account owner is requesting the changes

3.2.18 The System shall allow a customer to request a new/temporary password in case they have forgotten their old password

3.2.19 The System shall allow a customer to request a refund if they no longer want to attend the showing they have paid for. All refund requests must be made 24 hours prior to the showing of the movie

3.2.20 The System shall have an option for a customer to view their order history on their account, and only the customer themselves can view this information

3.2.21 Use Cases

Use Case Name:	Add/Edit Movie Information
Scenario:	System administrator adds and/or edits movie information
Triggering Event:	System administrator wants to have movie information available to customers
Brief Description:	Admin can add or edit movie information such as title, genre(s), cast list, director(s), producer(s), ratings, reviews, and trailers
Actors:	System administrator
Stakeholders:	System administrators Customers
Preconditions:	Movie must have information available Admin must have access to the information
Postconditions:	Movie now has information on display

	Customers can search for any of these criteria Information is displayed on movie's home screen	
Flow of Activities:	Actor	System
	1. Admin finds movie 2. Admin chooses to update movie information 3. Admin enters fields for update 4. Admin saves new information	1. System displays movie 2. System allows admin to edit information 3. System tests entered fields to make sure they are correct and unique for the movie title, and correct for every other piece of information 4. System saves information for the movie, and now displays the movie with its information
Exception Conditions:	1. Admin enters movie title that already exists, informs admin a movie with that title already exists 2. Admin enters no text for update, informs admin they need to enter in at least one character. 3. System does not have the movie the admin is looking for, informs admin they need to search again	
Alternative flow	Movie has no information at 2, Allows information to be entered for first time and checks to make sure the movie is unique in its name	

Use Case Name:	Add Booking information
Scenario:	System admin adds information for booking
Triggering Event:	System administrator wants to have movie information available to customers and allow them to book their tickets in advance
Brief Description:	Admin adds the viewing times and dates for movies, seating availability and seating location

<table>
<tr><td>Actors:</td><td colspan="2">System admin</td></tr>
<tr><td>Stakeholders:</td><td colspan="2">System admin
Customers</td></tr>
<tr><td>Preconditions:</td><td colspan="2">Movie must have available viewing information
Movie must be currently showing or will be shown in the future
Seating must be available for viewing
Admin must choose a movie</td></tr>
<tr><td>Postconditions:</td><td colspan="2">Movie now has viewing information and dates
Movie now has seating information available
Movie now can have seats booked for the future
Customers can now pay for their tickets in advance</td></tr>
<tr><td>Flow of Activities:</td><td>Actor</td><td>System</td></tr>
<tr><td></td><td>1. Admin must find movie
2. Admin chooses to edit the information
3. Admin enters the information
4. Admin saves and updates the information</td><td>1. System displays movie
2. System allows admin to edit information
3. System allows admin to enter text
4.1. System saves information and displays it on the movie’s home screen, making it searchable by customers
4.2. System allows searching for movies by the viewing dates and times
4.3. System allows choosing a seat and pay for the ticket for that seat in advance
4.4. System can accept payment</td></tr>
<tr><td>Exception Conditions:</td><td colspan="2">1. Admin enters no text for update, informs admin they need to enter in at least one character.
2. System does not have the movie the admin is looking for, informs admin they need to search again
3. All available seats are booked, system informs customer
4. Credit card information is wrong/invalid/not authorized, tells customer that card is wrong/invalid/not authorized
5. Admin adds invalid viewing dates or times, informs admin that the time/date is not valid</td></tr>
<tr><td>Alternative flow</td><td colspan="2">None</td></tr>
</table>

Use Case Name:	Search for movie
Scenario:	System administrator adds search functionality
Triggering Event:	System administrator wants to enable customers to search for movies

<table>
<tr><td>Brief Description:</td><td colspan="2">System admin can create search functionality so customers can search for movies based on certain criteria</td></tr>
<tr><td>Actors:</td><td colspan="2">System administrator</td></tr>
<tr><td>Stakeholders:</td><td colspan="2">System administrators
Customers</td></tr>
<tr><td>Preconditions:</td><td colspan="2">Movie must have categorical data
Customers must know about this data
System needs to have categories for the data to search</td></tr>
<tr><td>Postconditions:</td><td colspan="2">Customer can now find movie
Customer can find multiple movies with the same category
Customer can view lists of movies within a specific category</td></tr>
<tr><td rowspan="2">Flow of Activities:</td><td>Actor</td><td>System</td></tr>
<tr><td>1. Admin finds movie categories
2. Admin adds movie categories to movie
3. Admin adds categories to the search engine
4. Customer searches for category
5. Customer now has list of movies under a category</td><td>1. System enables Admin to input data for movie categories
2. System saves and stores categories for a movie
3. System now saves categories to search engine database
4. System allows customer to search for categories
5. System displays a list of movies that are listed under the specified category</td></tr>
<tr><td>Exception Conditions:</td><td colspan="2">1. Admin enters no information for movie category, system informs admin they need to enter information
2. Movie already has the specified category attached, informs admin to use a new category
3. Customer does not input anything when searching, informs customer they need to input a category
4. Customer enters an invalid category, informs them to search based on a valid category</td></tr>
<tr><td>Alternative flow</td><td colspan="2">None</td></tr>
</table>

Use Case Name:	Ticket Prices

<table>
<tr><td>Scenario:</td><td colspan="2">System admin adds information for ticket prices</td></tr>
<tr><td>Triggering Event:</td><td colspan="2">System admin wants customers to be able to see ticket prices for the movies</td></tr>
<tr><td>Brief Description:</td><td colspan="2">System admin can add or edit ticket prices for a movie so customers can see this information when viewing a movie</td></tr>
<tr><td>Actors:</td><td colspan="2">System administrator</td></tr>
<tr><td>Stakeholders:</td><td colspan="2">System administrator
Customer</td></tr>
<tr><td>Preconditions:</td><td colspan="2">Movie must exist in database
Movie must have current or future show time</td></tr>
<tr><td>Postconditions:</td><td colspan="2">Movie now has available ticket pricing information
Customers can view this information</td></tr>
<tr><td rowspan="2">Flow of Activities:</td><td>Actor</td><td>System</td></tr>
<tr><td>Admin finds movie
Admin chooses to edit ticket pricing
Admin enters new ticket price
Admin saves ticket price</td><td>System displays movie
System allows admin to edit pricing
System allows pricing information to be entered, and makes sure the price is not out of range
System saves ticket price, and now displays it on the movie home page and the booking page for that movie</td></tr>
<tr><td>Exception Conditions:</td><td colspan="2">Ticket price entered is out of range, will tell admin that new pricing needs to be within range
No new price is entered, informs admin that a new price must be entered
System cannot find movie admin is searching for, informs admin they need to search again</td></tr>
<tr><td>Alternative flow</td><td colspan="2"></td></tr>
</table>

<table>
<tr><td>Use Case Name:</td><td colspan="2">Employee Accounts</td></tr>
<tr><td>Scenario:</td><td colspan="2">System admin changes permissions for accounts to become employee accounts</td></tr>
<tr><td>Triggering Event:</td><td colspan="2">System admin wants to keep track of their employee accounts</td></tr>
<tr><td>Brief Description:</td><td colspan="2">System admin can add or edit a new account and give it permissions to become an employee account</td></tr>
<tr><td>Actors:</td><td colspan="2">System administrator</td></tr>
<tr><td>Stakeholders:</td><td colspan="2">System administrator
Employee</td></tr>
<tr><td>Preconditions:</td><td colspan="2">Must have employee
Account must not be banned</td></tr>
<tr><td>Postconditions:</td><td colspan="2">New account added/updated
Account now has employee permissions</td></tr>
<tr><td rowspan="2">Flow of Activities:</td><td>Actor</td><td>System</td></tr>
<tr><td>1. Admin adds account
2. Employee enters information
3. Admin gives account employee permissions
4. Admin saves new permissions to account</td><td>1. System creates account
2. System allows employee to enter information for account
3. System accepts new permission level for account
4. System stores the new permission level for the account</td></tr>
<tr><td>Exception Conditions:</td><td colspan="2">1. Account already has employee permissions, informs admin that already has permissions
2. Account has been banned, informs admin that a new course of action must be taken</td></tr>
<tr><td>Alternative flow</td><td colspan="2">If new employee already has an account, skip straight to 3</td></tr>
</table>

<table>
<tr><td>Use Case Name:</td><td colspan="2">Suspend/Delete accounts</td></tr>
<tr><td>Scenario:</td><td colspan="2">System administrator suspends and/or deletes an account or its information</td></tr>
<tr><td>Triggering Event:</td><td colspan="2">An account has too many failed password attempts in a span of time, or has declined payment attempts in a span of time</td></tr>
<tr><td>Brief Description:</td><td colspan="2">Admin can suspend accounts if they feel it necessary, or delete member information if they choose to remove their information</td></tr>
<tr><td>Actors:</td><td colspan="2">System administrator</td></tr>
<tr><td>Stakeholders:</td><td colspan="2">System administrators
Customers</td></tr>
<tr><td>Preconditions:</td><td colspan="2">Account must exist
Account must have a history of non-compliance with features or is removing personal information
Account</td></tr>
<tr><td>Postconditions:</td><td colspan="2">Account is now suspended or information is now deleted
Customer is informed</td></tr>
<tr><td rowspan="2">Flow of Activities:</td><td>Actor</td><td>System</td></tr>
<tr><td>1. Admin finds the offending account
2. Admin chooses the suspend account option
3. Admin saves the flag for the account</td><td>1.1 System displays the account the admin is searching for
1.2 System displays a flag for the account that shows that the account is up for review
2. System denies customer access to the account and informs the customer to contact support
3. System stores the flag for the account and the reason why it was suspended</td></tr>
<tr><td>Exception Conditions:</td><td colspan="2">1. Account does not exist, informs admin to search again
2. Account is already suspended, informs admin no further action can be completed</td></tr>
<tr><td>Alternative flow</td><td colspan="2">Account is going from registered user to non-registered user
After step 1, then
2. Admin deletes personal information from account
3. Admin saves the changes made to the account
4. Admin flags the reason for the changes and saves the flag
System side:
2. System allows admin to delete data
3. System stores the changes to the account
4. System stores the flag</td></tr>
</table>

Use Case Name:	Promotional Offers	
Scenario:	System admin sends promotional offers to customers through email	
Triggering Event:	System admin wants customers to receive promo offers for movies	
Brief Description:	System admin can send promo offers to customers through email, and they can in turn use these offers when booking tickets	
Actors:	System administrator	
Stakeholders:	System administrator Customer	
Preconditions:	Customer must be registered user Customer must have email address set up Customer must have opted into receiving promo offers	
Postconditions:	Customer receives promo offer System now accepts that promo offer on booking	
Flow of Activities:	Actor	System
	1. Admin creates promo offer 2. Admin sends links of promo offer to customer emails	1. System adds promo offer in database 2. System emails customers with link to existing promo offer site
Exception Conditions:	1. Customer has invalid email saved, informs admin the email is not valid 2. Admin creates promo offer with fields that are out of range, informs admin to recreate fields	
Alternative flow	None	

<table>
<tr><td>Use Case Name:</td><td colspan="2">Login User</td></tr>
<tr><td>Scenario:</td><td colspan="2">Log user into the system</td></tr>
<tr><td>Triggering Event:</td><td colspan="2">A user submits login information to the system.</td></tr>
<tr><td>Brief Description:</td><td colspan="2">A user wants to log in to the system. They submit their login info.</td></tr>
<tr><td>Actors:</td><td colspan="2">User</td></tr>
<tr><td>Stakeholders:</td><td colspan="2">User
Login Database</td></tr>
<tr><td>Preconditions:</td><td colspan="2">User has submitted a username and password.
Login Database must exist.</td></tr>
<tr><td>Postconditions:</td><td colspan="2">User is logged into the system, or
User has received a log-in error message.</td></tr>
<tr><td rowspan="2">Flow of Activities:</td><td>Actor</td><td>System</td></tr>
<tr><td>1. User enters username.
2. User enters password.
3. User clicks the login button.</td><td>1. System compares data entered against login database.
2. System checks whether login information is in database.
3. Redirect user to homepage with logged in status</td></tr>
<tr><td>Exception Conditions:</td><td colspan="2">1. If log in information does not match database entry, display error message</td></tr>
<tr><td>Alternative flow</td><td colspan="2"></td></tr>
</table>

<table>
<tr><td>Use Case Name:</td><td colspan="2">Authenticate Administrator Login</td></tr>
<tr><td>Scenario:</td><td colspan="2">Verify administrator log in information</td></tr>
<tr><td>Triggering Event:</td><td colspan="2">An administrator submits login information to the system.</td></tr>
<tr><td>Brief Description:</td><td colspan="2">A n administrator logs into the system. In addition to normal log in procedures, the system should display special administrator actions for this user.</td></tr>
<tr><td>Actors:</td><td colspan="2">Administrator</td></tr>
<tr><td>Stakeholders:</td><td colspan="2">Administrator
Login Database</td></tr>
<tr><td>Preconditions:</td><td colspan="2">Administrator has submitted a username and password.
Login Database must exist.
Log in data is marked as administrator data in database.</td></tr>
<tr><td>Postconditions:</td><td colspan="2">Administrator is logged into the system, or
User has received a log-in error message.</td></tr>
<tr><td rowspan="2">Flow of Activities:</td><td>Actor</td><td>System</td></tr>
<tr><td>1. Administrator logs in.</td><td>1. System begins logging user in normally.
2. Login data is determined to be administrator data.
3. System finishes logging in user, but marks user as an administrator.</td></tr>
<tr><td>Exception Conditions:</td><td colspan="2">1. If log in information does not match database entry, display error message.</td></tr>
<tr><td>Alternative flow</td><td colspan="2"></td></tr>
</table>

<table>
<tr><td>Use Case Name:</td><td colspan="2">New User Login Data Storage Procedure</td></tr>
<tr><td>Scenario:</td><td colspan="2">Add new user login data to database</td></tr>
<tr><td>Triggering Event:</td><td colspan="2">A user registers an account to the system.</td></tr>
<tr><td>Brief Description:</td><td colspan="2">A new user registers an account to the system. This user’s data is stored in a database for future use in login verification.</td></tr>
<tr><td>Actors:</td><td colspan="2">User</td></tr>
<tr><td>Stakeholders:</td><td colspan="2">User
Login Database</td></tr>
<tr><td>Preconditions:</td><td colspan="2">User has submitted a non-null username and password.
Login Database must exist.</td></tr>
<tr><td>Postconditions:</td><td colspan="2">User login data is stored in database</td></tr>
<tr><td rowspan="2">Flow of Activities:</td><td>Actor</td><td>System</td></tr>
<tr><td>1. User clicks on “Register Account” in login page.
2. User enters a new username.
3. User enters a password.
4. User enters an optional email address.</td><td>1. System verifies username is not already in database.
2. System verifies prospective password meets requirements.
3. System creates new user account in database.</td></tr>
<tr><td>Exception Conditions:</td><td colspan="2">1. If username is already in database, display error message.
2. If password does not meet requirements, display error message.</td></tr>
<tr><td>Alternative flow</td><td colspan="2"></td></tr>
</table>

Use Case Name:	View cinema layout	
Scenario	Customer views the cinema layout	
Triggering Event:	Customer wants to select seats to reserve for a specific movie.	
Brief Description:	After customer chooses a showtime for a movie, then is prompted to select the seats they wish to reserve from a diagram of the cinema. The diagram highlights the available seats.	
Actors:	Customer	
Stakeholders:	Customer Website administrators	
Preconditions:	Customer has selected a movie. Customer has selected a showtime. Both movie and showtime must exist in database.	
Postconditions:	Seats are reserved for specific movie/showtime. Available seats are updated. Customer is prompted for payment.	
Flow of Activities:	Actor	System
	1. Customer sees diagram. 2. Customer selects seats to reserve. 3. Customer clicks reserve button. 4. Customer redirected to complete order page.	1.1 System displays seats based on current status of reservations for movie in database, highlighting non-reserved seats. 2.1 System changes color of selected seats. 3.1 System updates reservation database for this movie. 4.1 System opens the order completion page.
Exception Conditions:	1. If user cancels order in the completion stages, the reservation database gets updated for that movie.	

	NOTE: update database immediately when seats are selected to avoid two people trying to get the same seats. (Avoid someone else trying to reserve same seats before the original customer has completed order). 2. (POSSIBILITY) Timer runs out for order completion, the reservation database gets updated for that movie.
Alternative flow	None

Use Case Name:	Ticket booking	
Scenario	The customers ticket booking process	
Triggering Event:	The customer sees a movie they wish to see.	
Brief Description:	A customer wants to book tickets for a movie. After selecting a movie from the page of current movies, the customer selects a date, time then seats/type (age group) of seat and enters an email.	
Actors:	Customer Movie reservation database	
Stakeholders:	Customer Website Administrator	
Preconditions:	Movie has showtimes available. Movie is not sold out.	
Postconditions:	Reservations database gets updated for that movie. Customer gets redirected to order completion pages.	
Flow of Activities:	Actor	System
	1. The customer selects a movie from a list of current movies playing in the cinema. 2. The customer chooses from a list of dates that the movie is playing. 3. The customer then chooses from a list of showtimes for the movie on the desired date. 4. The customer is prompted to choose seats to reserve from a diagram of the cinema. 5. When seats are reserved, the customer is prompted to select	1.1 Display all movies currently being played. 2.1 System gets dates the movie is showing. 3.1 System gets the showtimes for the movie on the date chosen. 4.1 System runs the seat display/selection code to display the diagram.

	the type of ticket (age of viewer) for each seat reserved. 6. Customer is prompted to enter a valid email address. Then is redirected to the payment process.	4.2 Reserved table in database gets updated. 5.1 Update type of ticket for reserved database. 6.1 Add email to emails database. Redirect to payment.
Exception Conditions:	2.1/3.1 If movie is sold out during showtime and/or date customer wants, display apology and return to dates selection for movie.	
Alternative flow	None.	

Use Case Name:	Checkout	
Scenario	Customer has selected a showtime and seats and wishes to pay for order.	
Triggering Event:	Customer selects the type of seat for each desired seat and clicks confirm.	
Brief Description:	Customer securely enters billing address, payment type, card number, exp date, and security number. Customer may enter a promo code now as well. After info is entered, the customer sees the final price calculation and confirms payment.	
Actors:	Customer Website Administrator	
Stakeholders:	Customer Website Administrator	
Preconditions:	Customer has selected a movie/showtime/seats Movie exists in database.	
Postconditions:	New sale created in database. Customer acct(if exists) is updated. Emails database is updated (new receipt corresponding to email is added).	
Flow of Activities:	Actor	System
	1. Customer is prompted it enter billing address, payment type, card number, exp date, and security number (if non-registered user). 1ALT (if registered): select payment method from previously saved methods. 2. Customer enters promo-code. 3. Customer enters valid email (if non-registered user). 4. Total cost is displayed and customer confirms payment.	1.1 System checks validity of payment information. 2.1 System checks promo code database to see if it exists. 3.1 System updates email database. 4.1 System calculates final price including fees and promotions and displays this info to customer.
Exception Conditions:	1.1 If invalid payment info is entered, alert customer and ask to re-enter. 2.1 If invalid promo code, alert user and ask to re-enter. 2.2 If customer is not registered, politely deny promo code and explain reason, then ask if they wish to register. 4.4 Total cost is negative, display a cost of $0.00.	
Alternative flow	User is registered, just prompt to choose payment method, or add new one. Skip step 3 because an email is already associated with account.	

Use Case Name:	Confirmation	
Scenario	Customer receives a confirmation email and a confirmation page is displayed.	
Triggering Event:	Customer purchases tickets.	
Brief Description:	Customer confirms payment and completes the order. An email confirming the order is sent to the provided email and a screen is displayed confirming the order/specifying that a conf. email was sent.	
Actors:	Customer Website Administrator	
Stakeholders:	Customer Website Administrator	
Preconditions:	Customer successfully ordered tickets. Email that was entered is valid. Payment was not rejected.	
Postconditions:	Email sent to customer's email. Confirmation page is displayed.	
Flow of Activities:	Actor	System
	1. Customer clicks confirm payment button. 2. A page is displayed with an order summary and a message saying a confirmation email was sent to [customers_email@xxx.xxx]. 3. Email is sent to user.	1.1 System processes payment. 2.1 System displays confirmation page with order summary. 3.1 Email is grabbed from database, and code for emailing customer is executed.
Exception Conditions:	1.1 If payment failed for reasons other than incorrect input, apologize and explain customer then redirect to billing info input page. 3.1 If email is invalid (how to detect?), ask for new email if they wish to get emailed a confirmation.	
Alternative flow	None	

Use Case Name:	Order price	
Scenario	Order price calculation by system.	
Triggering Event:	Customer selects seats, seat types, enters promo code, email and valid payment options.	
Brief Description:	The system calculates the price of the order by adding ticket prices w/ promo codes applied, sales tax, online fees.	
Actors:	Customer Website Administrator	
Stakeholders:	Customer Website Administrator	
Preconditions:	Customer has selected a movie/showtime/seats, entered payment info and promo codes. Movie exists in database. Tickets are available. Promo code is valid	
Postconditions:	A final price is calculated.	
Flow of Activities:	Actor	System
	1. Customer inputs all necessary info. 2. Customer views price.	1.1 System applies valid promo codes to the current price of tickets. 1.2 System calculates tax and online fees then adds them to the overall total. NOTE: is tax calculated based on promo total or pre-promo total?
Exception Conditions:	1.1 If info is incorrect, prompt customer to re-enter.	
Alternative flow	None	

Use Case Name:	Promo Codes	
Scenario	Customer inputs promo code for a discount.	
Triggering Event:	Customer inputs promo code in the payment page.	
Brief Description:	When the customer applies a promo code, the system grabs the promo info from the database then uses that info to apply a discount to the price (pre-tax/fees).	
Actors:	Customer	
Stakeholders:	Customer Website Administrator Database – promo table	
Preconditions:	Customer is registered. Promo code exists.	
Postconditions:	Promo code is grabbed from DB. Apply discount to price (pre-tax/fees)	
Flow of Activities:	Actor	System
	1. Customer prompted to enter a promo code. 2. Customer receives discount.	1.2 System checks DB for code and discount info. 2.1 System applies discount to price (pre-tax/fees)
Exception Conditions:	1.1 If the promo code does not exist in the DB (or is expired), alert the user and ask for re-entry	
Alternative flow		

Use Case Name:	Registration	
Scenario	Customer registration	
Triggering Event:	Customer wishes to register an account with the web application.	
Brief Description:	A customer who wishes to register an account is prompted to enter a valid email, password, first and last name (optional: home address, phone #, and payment info for speedy checkout)	
Actors:	Customer	
Stakeholders:	Customer Website Administrator Registered User DB	
Preconditions:	Customer is not already registered.	
Postconditions:	Customer is registered. Register DB is updated. Email DB is updated.	
Flow of Activities:	Actor	System
	1. Customer wants to register an account. 2. Prompted to enter a valid email, a password, first name, last name. 3. Optional section displayed below advertising a quicker checkout: enter home address, phone # and payment info. 4. Click submit.	1.1 System pulls form for registration. 4.1 System updates the registered customer DB, email DB and sends initial "Thanks" email to the email on file.
Exception Conditions:	2.1 If email is already in the DB, alert customer prompt to enter new one. 2.2 (optional) if password strength isn't good enough, prompt to re-enter.	
Alternative flow	None	

<table>
<tr><td>Use Case Name:</td><td colspan="2">Customer Verification</td></tr>
<tr><td>Scenario:</td><td colspan="2">Verify that customers obtain a verification code</td></tr>
<tr><td>Triggering Event:</td><td colspan="2">customer wants to login to the system</td></tr>
<tr><td>Brief Description:</td><td colspan="2">customers want to register in the system and should receive a verification code through their email address. Then they are given a unique account ID to login into the system.</td></tr>
<tr><td>Actors:</td><td colspan="2">Customer</td></tr>
<tr><td>Stakeholders:</td><td colspan="2">System administrator
Manager
Customer</td></tr>
<tr><td>Preconditions:</td><td colspan="2">1. Each user should register before attempting to login into the system.
2. Each user should have one account ID that is linked to their email address and belongs to them only and is not shared.</td></tr>
<tr><td>Post conditions:</td><td colspan="2">1. Verification code is sent to the users address.
2. User is assigned a unique account ID.
3. Registered users can login using account ID or email address and password.</td></tr>
<tr><td rowspan="2">Flow of Activities:</td><td>Actor</td><td>System</td></tr>
<tr><td>1. Customers registers into the system.
2. Customers use one email address to register.
3. Customers receive verification code to their email address.
4. After verification, Customers are assigned a unique account ID
5. Registered customers can login using account ID or email address and password.
6. Customers email address is linked to their account ID and are only assigned the account ID once.</td><td>1.1 System stores customer’s email address and password.
2.1 System generates unique account ID and sends it to customers email address
3.1 System saves customers login information
4.1 System links customer’s unique account ID to their email address.</td></tr>
<tr><td>Exception Conditions:</td><td colspan="2">1. If customer fails to receive verification code, require customer to resend the verification code.</td></tr>
<tr><td>Alternative flow</td><td colspan="2">None</td></tr>
</table>

Use Case Name:	Customer subscription	
Scenario:	Verify that customers subscribe for promotions and latest news	
Triggering Event:	Customer subscribes for promotions	
Brief Description:	Registered customers want to subscribe or unsubscribe for promotions and latest news offered by the system administrator	
Actors:	Customer	
Stakeholders:	System administrator Manager Customer	
Preconditions:	1. Users must be registered in the system to subscribe/unsubscribe to promotions and latest news being offered.	
Post conditions:	1. Users can successfully subscribe/unsubscribe for promotions being offered in the system.	
Flow of Activities:	Actor	System
	1. Registered customers subscribe to promotions desired through the system. 2. Registered customers unsubscribe to promotion not desired through the system.	1.2 System checks if customer is registered and then allows access for promotional offers 2.1 System checks if customer is registered and then allows access to remove offers
Exception Conditions:	1. If customer fails to subscribe/unsubscribe to promotions being offered in the system, check if customer is registered in the system. Check if customer has access to promotions.	
Alternative flow	None	

Use Case Name:	Customer profile modification	
Scenario:	Customer wants to view his or her profile	
Triggering Event:	Customer views or updates profile	
Brief Description:	Customers want to view and modify their profile at any time when accessing the system.	
Actors:	Customer	
Stakeholders:	System administrator Manager Customer	
Preconditions:	1. Customers must be registered into the system to have a profile.	
Post conditions:	1. Customers can view and update profile.	
Flow of Activities:	Actor	System
	1. Customers login into system. 2. Customers access their profile. 3. Customers view their profile. 4. Customers update profile.	1.1 System gives customer access to his or her account 2.1 System allows customer to access profile 3.1 System saves any modifications made and updates the customers profile
Exception Conditions:	1. If customer cannot view their profile, check if customer has access to it 2. If customer cannot update their profile, check if customer has access to modify their profile at any time.	
Alternative flow	None	

<table>
<tr><td>Use Case Name:</td><td colspan="2">Secure forget-password facility</td></tr>
<tr><td>Scenario:</td><td colspan="2">Customer forgets their password</td></tr>
<tr><td>Triggering Event:</td><td colspan="2">Customer wants to update password</td></tr>
<tr><td>Brief Description:</td><td colspan="2">System should have an option to allow a customer to update or change his or her password in case they forget it</td></tr>
<tr><td>Actors:</td><td colspan="2">System Administrator</td></tr>
<tr><td>Stakeholders:</td><td colspan="2">System administrator
Manager
Customer</td></tr>
<tr><td>Preconditions:</td><td colspan="2">1. Administrator should provide the facility within the system so that users are able to utilize the forget-password option.</td></tr>
<tr><td>Post conditions:</td><td colspan="2">1. Customers make use of the forget-password facility and update their password to login into the system.</td></tr>
<tr><td rowspan="2">Flow of Activities:</td><td>Actor</td><td>System</td></tr>
<tr><td>1. User forgets his or her password.
2. User selects the forget-password option.
3. User updates their password.
4. User logins into the system</td><td>1.1 System allows customer to update password
2.1 System updates customer’s password and saves it</td></tr>
<tr><td>Exception Conditions:</td><td colspan="2">1. If customer fails to update password, check to see if they have access.
2. If system does not update password, check to see if passwords being saved correctly. Also, make sure password can always be updated if customer forgets it.</td></tr>
<tr><td>Alternative flow</td><td colspan="2">None</td></tr>
</table>

<table>
<tr><td>Use Case Name:</td><td colspan="2">Customer refund</td></tr>
<tr><td>Scenario:</td><td colspan="2">Customer wants to return ticket for refund</td></tr>
<tr><td>Triggering Event:</td><td colspan="2">Customer request for refund</td></tr>
<tr><td>Brief Description:</td><td colspan="2">Customer returns his or her ticket and receives a full refund up to 60 minutes before the show time.</td></tr>
<tr><td>Actors:</td><td colspan="2">Customer</td></tr>
<tr><td>Stakeholders:</td><td colspan="2">System administrator
Manager
Customer</td></tr>
<tr><td>Preconditions:</td><td colspan="2">1. Customer should return tickets before the show time of the movie.</td></tr>
<tr><td>Post conditions:</td><td colspan="2">1. Customer should receive full refund up to 60 minutes before the show time</td></tr>
<tr><td rowspan="2">Flow of Activities:</td><td>Actor</td><td>System</td></tr>
<tr><td>1. Customers decide to return tickets
2. Customers return their tickets before the show time of a movie
3. Customers receive refund up to 60 minutes before the show time</td><td>1.1 System checks if the ticket has not been expired and is past the show time.
2.1 System processes the ticket and calculates the customers refund
3.1 System processes the refund before the 60 minute period</td></tr>
<tr><td>Exception Conditions:</td><td colspan="2">1. If customer fails to return ticket before show time, reject refund
2. If customer returns ticket and doesn’t receive refund, check to see if system sent a refund, if not, modify system to send refund and promotion for inconvenience.</td></tr>
<tr><td>Alternative flow</td><td colspan="2">None</td></tr>
</table>

Use Case Name:	Order history	
Scenario:	Customer wants to view order history	
Triggering Event:	Customer requests to view order transaction history	
Brief Description:	Customer wants to view all order history that he or she has purchased through the system.	
Actors:	Customer	
Stakeholders:	System administrator Manager Customer	
Preconditions:	1. Customer should be registered in the system 2. Customer must make purchases first before viewing order history	
Post conditions:	1. Customer successfully views all order history since the first purchase	
Flow of Activities:	Actor	System
	1. Customer requests to view order history	1.1 System checks if customer is registered in system 2.1 System checks to see if customer made purchases 3.1 System provides all purchase transactions
Exception Conditions:	1. If customer cannot view order history, check to see if he or she has access to purchases. If not, give them access 2. If customer cannot find certain transaction, check to see if the transaction was made by checking the time and day of transaction	
Alternative flow	None	

3.3 Nonfunctional Requirements

3.3.1 Performance Requirements

PE-1: The Cinema E-Booking system shall accommodate 400 users during the peak hours of 5:00pm – 8:00pm when most users will be purchasing tickets.
PE-2: The Cinema E-Booking system shall use indexing and other similar tools of MySQL to keep query times under ten seconds for retrieving movies, account information, and promotional codes.
PE-3: The Cinema E-Booking system shall display confirmation messages within 5 seconds of the time the user submits their order to the system.

3.3.2 Safety Requirements

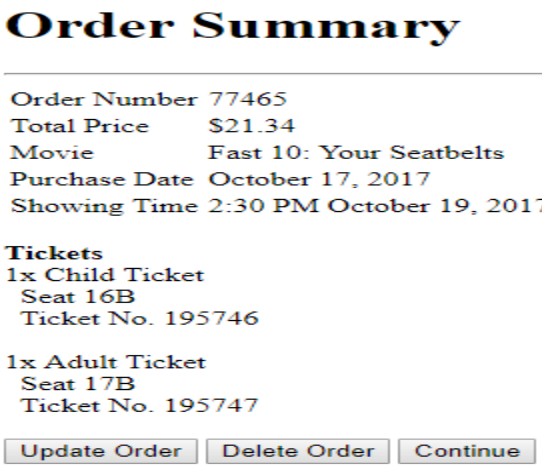
SR-1: All sensitive user account data shall be encrypted in the MySQL database using AES encryption provided by MySQL.

3.3.3 Security Requirements

SR1 - In order to authenticate a new account, a screen will appear after a user has provided their credentials to create a new account/edit your account information. This screen will prompt for a verification code that is sent to the user's email. After the user has entered the verification code, if the code is a correct, unique, unused code, the account will be created/changed.
SR2 - Also, a user is required to be logged in to their account when they are attempting to view or edit their account information.
SR3 – Users will only be able to access their accounts, and can only view information and order history from their individual account. Users shall not be permitted to look at order history or account information of a separate user.
SR4 – All transactions involving financial and personal information will be encrypted

3.3.4 User Interface

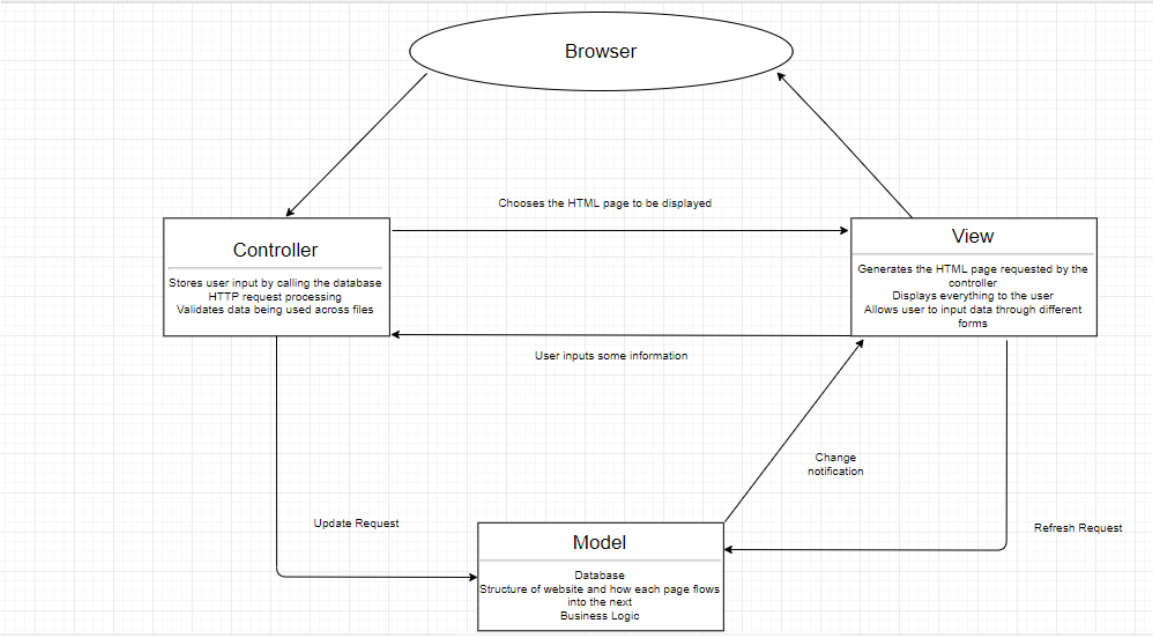
UI-1: There will be a help link from each displayed HTML page to provide information to the user so the user can understand how to navigate and utilize each page.
UI-2: The HTML pages utilize a keyboard and mouse combination to navigate each and every page
UI-3: An example of how the user interface will look for the order summary screen



3.4 System Architectural Design

The System Architecture Pattern is a Model-View-Controller Pattern. It has a controller that handles requests to the database by user input and gives this information to both the View and the Model. The Model component here is our Mysql database, and also uses the Apache Tomcat server to run the various programs and operations. This requires the programs and files to all be in the Tomcat folder in order to run properly. The View component here takes the form of the different HTML files that display everything to the user. Instead of just regular HTML files, however, we used .jsp files, in order to have the files be able to call upon each other themselves instead of working with a servlet. Also, CSS styles for the HTML is half and half, certain files with minimal CSS were deemed to be more concise if the CSS was done internally, while those with a large amount of CSS were deemed more practical to have it done externally.

3.4.1



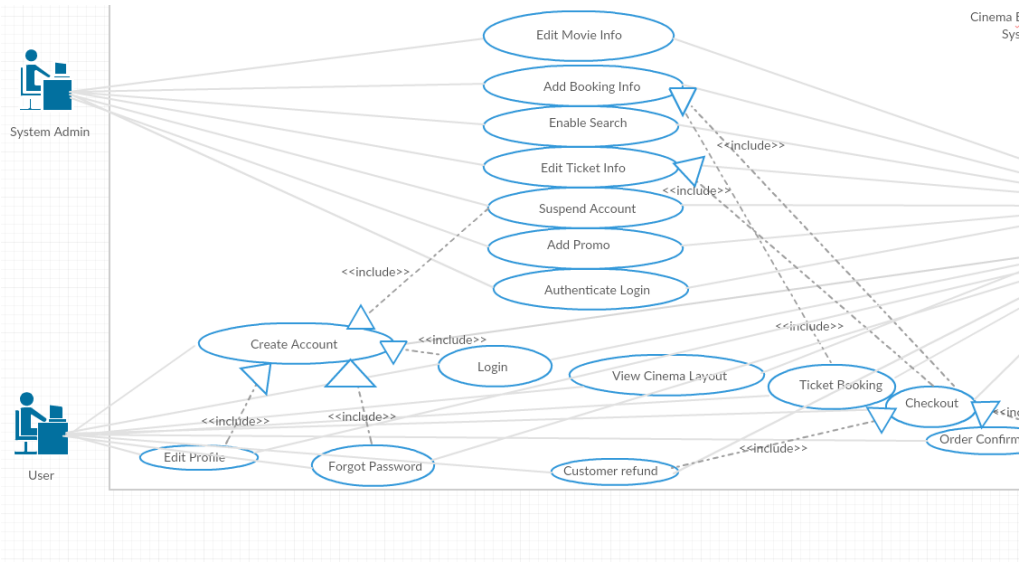
3.5 Constraints

1.3.1 Implementation Languages

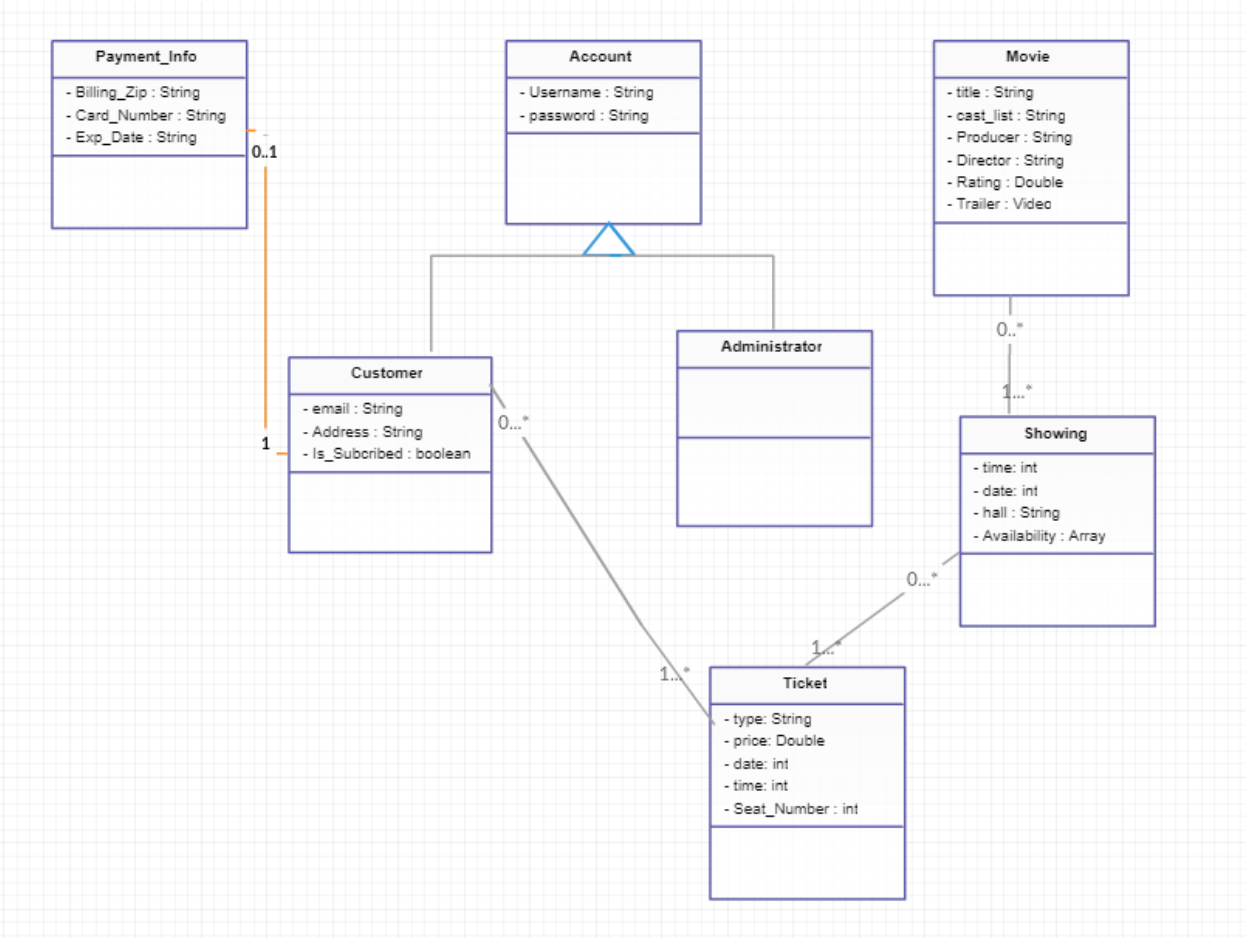
Portability-1: The Cinema E-Booking system shall be coded in Java to insure cross platform functionality to accommodate any changes to the system's location.
Portability-2: The Cinema E-Booking system's web interface shall be compatible and available to all web browsers, and shall perform at a consistent rate across browsers.

4 System Models

4.1 Use Case Model



4.2 Domain Object Model



4.3 Data Dictionary

4.3.1

<table>
<tr><td>Class</td><td colspan="2"></td></tr>
<tr><td rowspan="5">Customer</td><td colspan="2">The Customer can browse movies on the site and order tickets for future show times. When they do, the they go through a check-out process before the order is added to their order history. They can enter payment info at checkout or save the info for later purchases, which is stored in a PaymentInfo object. All information about the customer’s profile is stored in the Database.</td></tr>
<tr><td rowspan="3">Relationships</td><td>Associations: PaymentInfo, Order, Database</td></tr>
<tr><td>Aggregations: none</td></tr>
<tr><td>Generalizations: none</td></tr>
<tr><td colspan="2">Variables: customerID: int, orderHist: Ticket[], email: String, hasPromo:boolean, account:Account, streetAddress:String, phoneNo:String, city:String, state:String, payment:PaymentInfo</td></tr>
<tr><td colspan="2">Functions: none</td></tr>
</table>

4.3.2

<table>
<tr><td>Class</td><td colspan="2"></td></tr>
<tr><td rowspan="4">Movie</td><td colspan="2">Movies are categorized and stored in the database. They can be added, updated, or deleted by a manager or admin. Movies should have associated showtimes so that users can buy tickets from the movie page. The information for a movie should be stored in the Database.</td></tr>
<tr><td rowspan="3">Relationships</td><td>Associations: Showing, Order, Database</td></tr>
<tr><td>Aggregations: none</td></tr>
<tr><td>Generalizations: none</td></tr>
</table>

	Variables: movieID:int, title:String, castList:String[][], genre:String, producer:String, director:String, synopsis:String, reviews:String[], trailer:String, pictures:String, rating:int, showtimes:Showing[]
	Functions: none

4.3.3

<table>
<tr><td>Class</td><td colspan="2"></td></tr>
<tr><td rowspan="5">Showing</td><td colspan="2">A showing is a time slot where a movie is being screened. The showing should have an associated hall where it will be screened, movie that it will be showing, and a time when the screening will happen, as well as tickets for that showing that would be on sale.</td></tr>
<tr><td rowspan="3">Relationships</td><td>Associations: Movie, Database</td></tr>
<tr><td>Aggregations: none</td></tr>
<tr><td>Generalizations: none</td></tr>
<tr><td colspan="2">Variables: tichets:Ticket[], movie:Movie, hall:int, dateTime:Date</td></tr>
<tr><td colspan="2">Functions: none</td></tr>
</table>

4.3.4

<table>
<tr><td>Class</td><td colspan="2"></td></tr>
<tr><td rowspan="5">PaymentInfo</td><td colspan="2">An order is a collection of tickets that a customer is to purchase. The order can contain different tickets of different prices, but they must all be for the same movie. When the purchase is complete, the order is saved in the database.</td></tr>
<tr><td rowspan="3">Relationships</td><td>Associations: Customer, Database</td></tr>
<tr><td>Aggregations: none</td></tr>
<tr><td>Generalizations: none</td></tr>
<tr><td colspan="2">Variables: customer:Customer, cardNo:String, billingZip:String, cardType:String, expDate:Date</td></tr>
<tr><td colspan="2">Functions: verifyPaymentInfo():boolean</td></tr>
</table>

4.3.5

<table>
<tr><td>Class</td><td colspan="2"></td></tr>
<tr><td rowspan="5">Order</td><td colspan="2">An order is a collection of tickets that a customer is to purchase. The order can contain different tickets of different prices, but they must all be for the same movie. When the purchase is complete, the order is saved in the database.</td></tr>
<tr><td rowspan="3">Relationships</td><td>Associations: Movie, Customer, Database</td></tr>
<tr><td>Aggregations: none</td></tr>
<tr><td>Generalizations: none</td></tr>
<tr><td colspan="2">Variables: bookingID:int, date:Date, totalPrice:Currency, tickets:Ticket[]</td></tr>
<tr><td colspan="2">Functions: selectMovie(), selectShowTime(), selectSeat()</td></tr>
</table>

4.3.6

<table>
<tr><td>Class</td><td colspan="2"></td></tr>
<tr><td rowspan="5">Manager</td><td colspan="2">A manager is an employee who has the ability to add, delete, or edit the information of the movies on the site. A manager can access the movie database and edit the information anyway they wish.</td></tr>
<tr><td rowspan="3">Relationships</td><td>Associations: Admin, Database</td></tr>
<tr><td>Aggregations: none</td></tr>
<tr><td>Generalizations: none</td></tr>
<tr><td colspan="2">Variables: username:String, password:String</td></tr>
<tr><td colspan="2">Functions: addMovie(), updateMovie(), removeMovie()</td></tr>
</table>

4.3.7

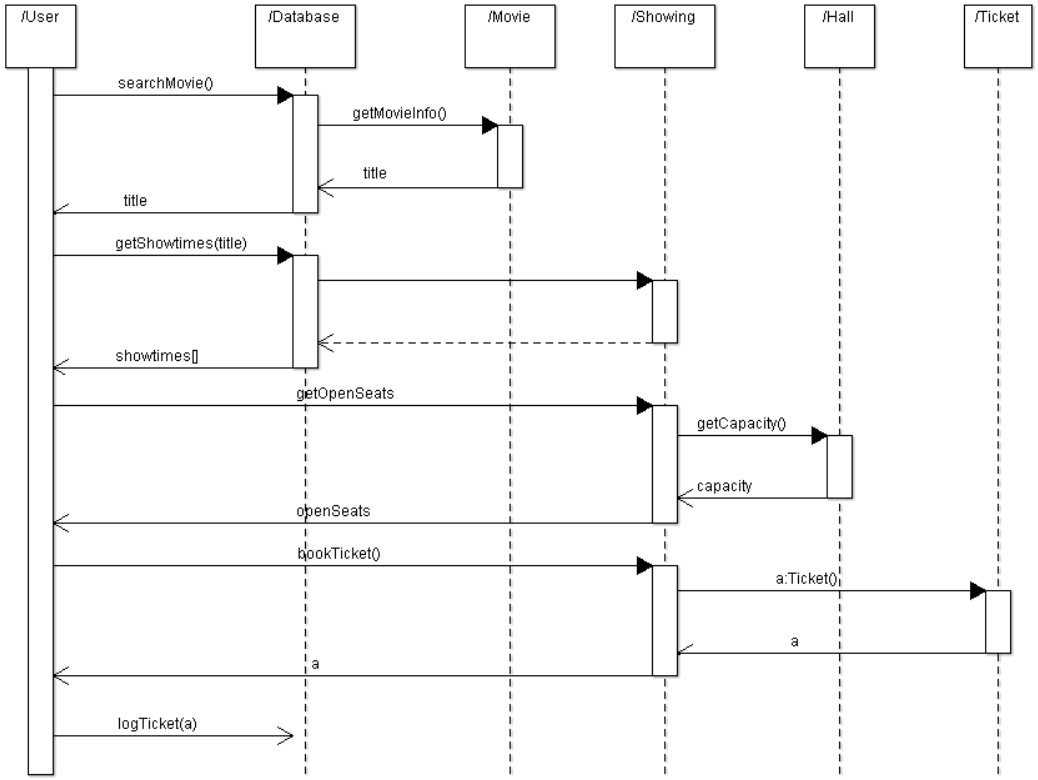
<table>
<tr><td>Class</td><td colspan="2"></td></tr>
<tr><td rowspan="5">Admin</td><td colspan="2">An admin is an advanced version of a manager, with all the same attributes and abilities, but with the added ability to promote user accounts to managers for new employees. There must always be only one admin.</td></tr>
<tr><td rowspan="3">Relationships</td><td>Associations: Manager, Database</td></tr>
<tr><td>Aggregations: none</td></tr>
<tr><td>Generalizations: none</td></tr>
<tr><td colspan="2">Variables: username:String, password:String, employees:Manager[]</td></tr>
<tr><td colspan="2">Functions: addMovie(), updateMovie(), removeMovie(), promoteUser(), demoteUser()</td></tr>
</table>

4.3.8

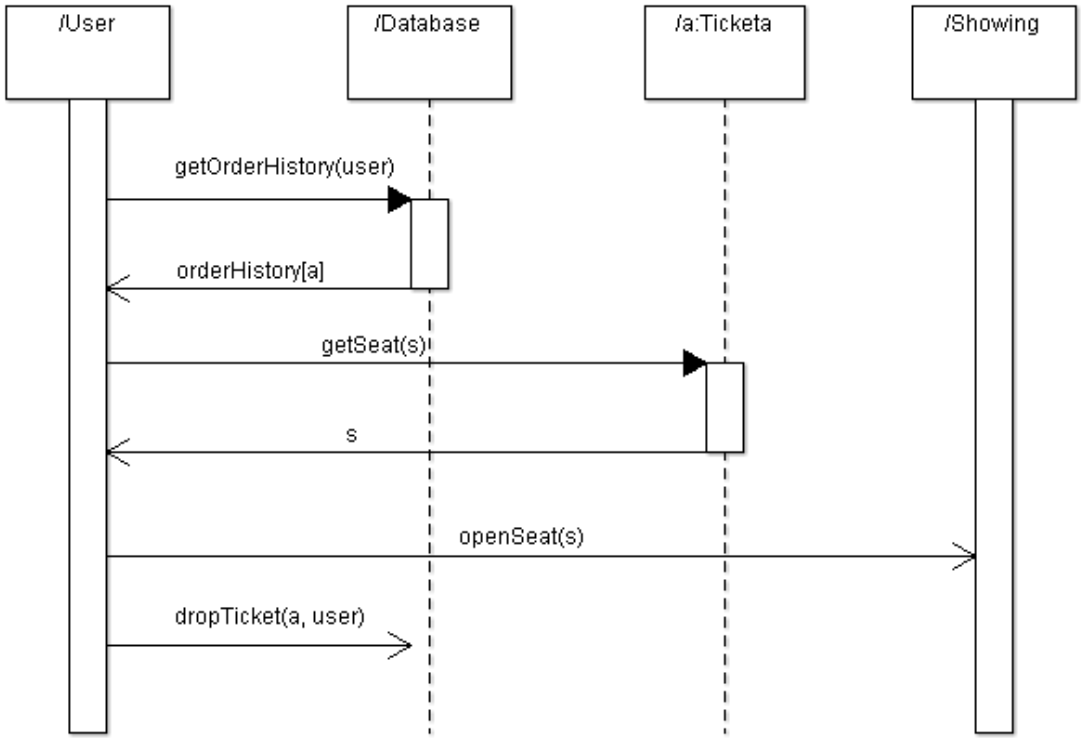
<table>
<tr><td>Class</td><td colspan="2"></td></tr>
<tr><td rowspan="5">Database</td><td colspan="2">The database will hold all recorded information on movies, under accounts, payment infos, and purchase histories. This class will interact with the SQL database whenever other classes need to save data.</td></tr>
<tr><td rowspan="3">Relationships</td><td>Associations: PaymentInfo, Customer, Manager, Admin, Customer, Order, Movie, Showing</td></tr>
<tr><td>Aggregations: none</td></tr>
<tr><td>Generalizations: none</td></tr>
<tr><td colspan="2">Variables: database:SQLDatabase</td></tr>
<tr><td colspan="2">Functions: addTable(), dropTable(), updateTable(), selectTable()</td></tr>
</table>

4.4 Sequence Diagram

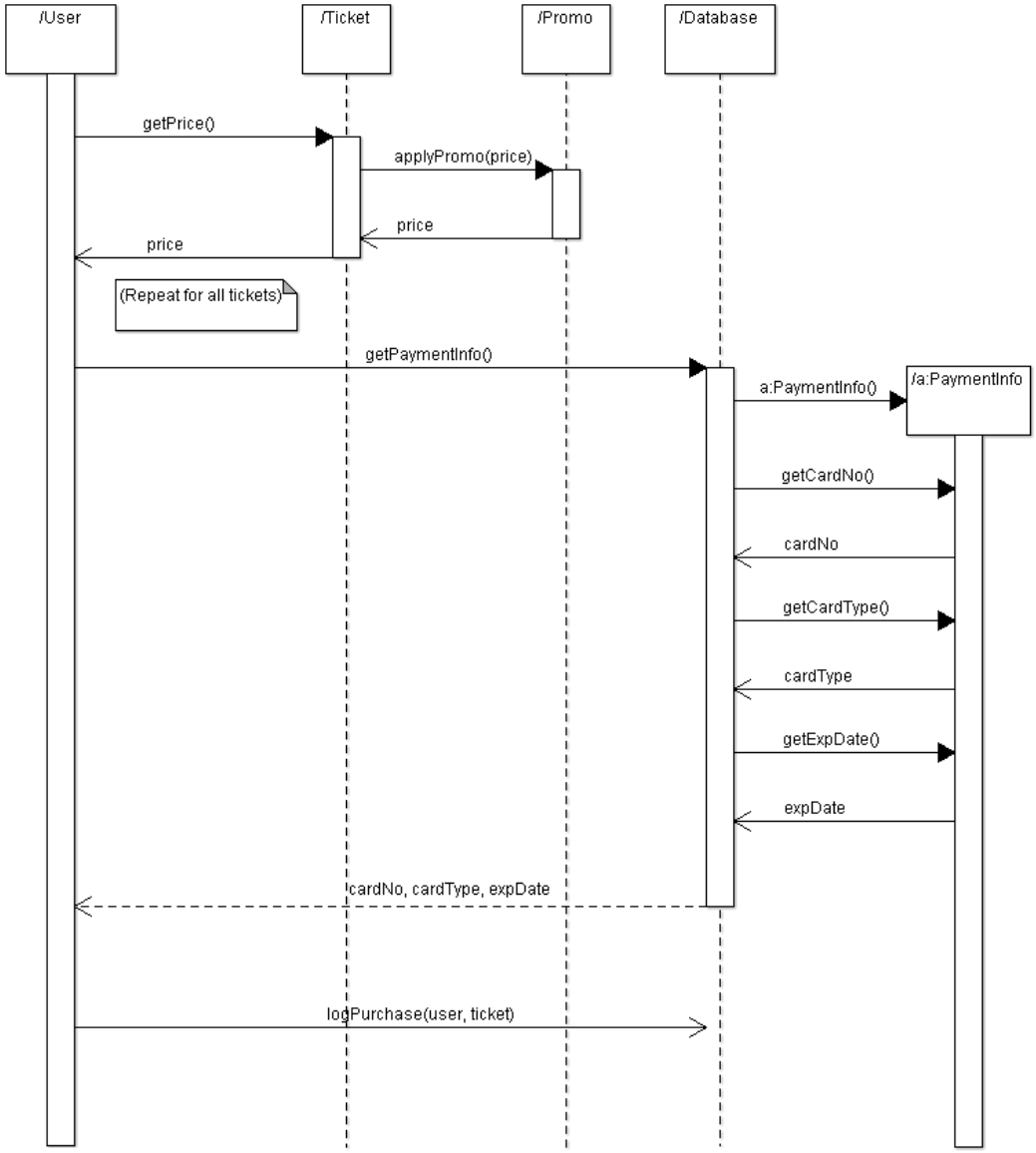
4.4.1 Booking a Ticket



4.4.2 Cancelling a Ticket

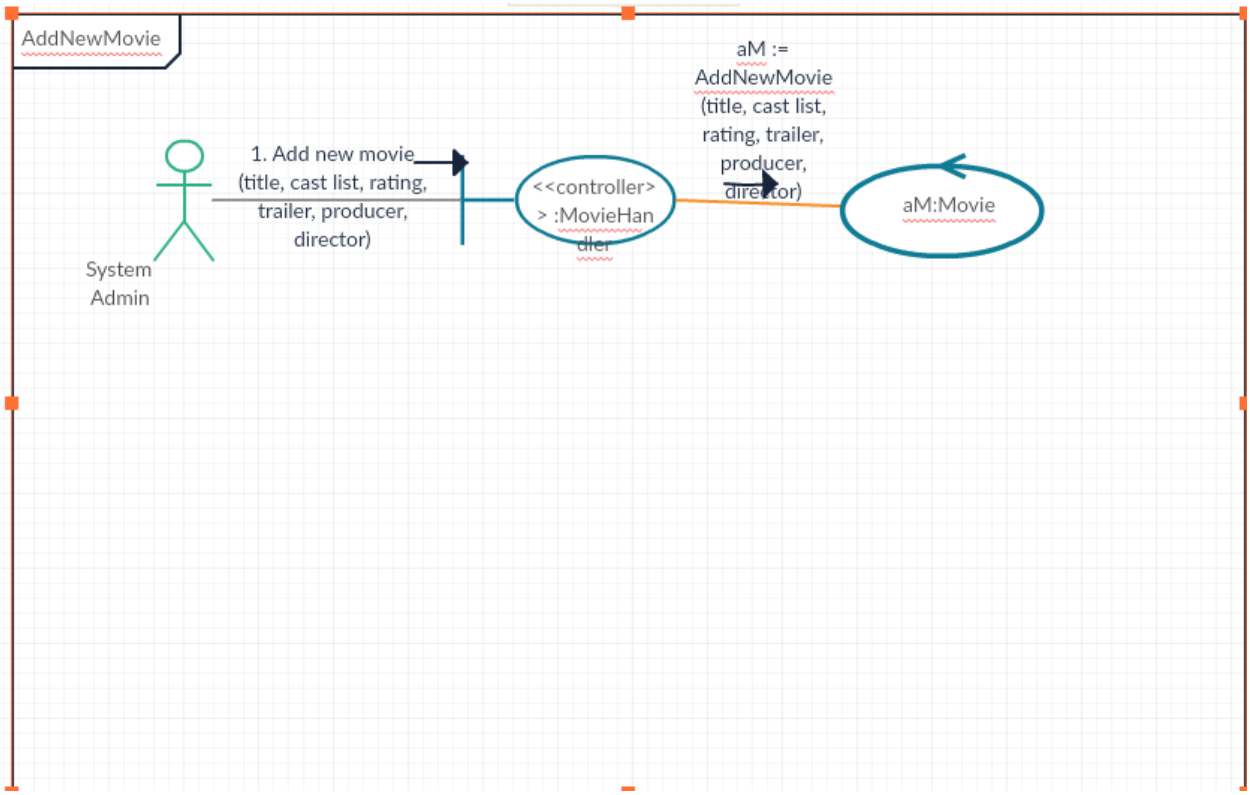


4.4.3 Checkout

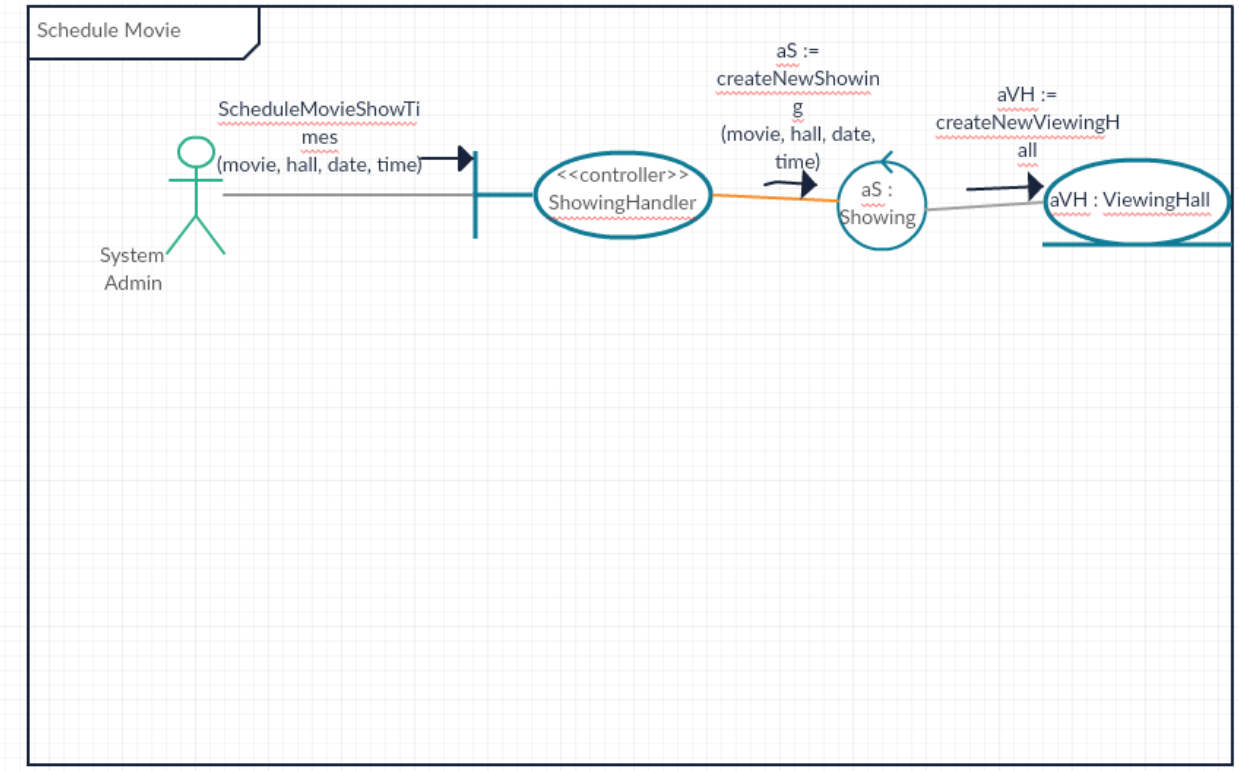


4.5 Communication Diagram

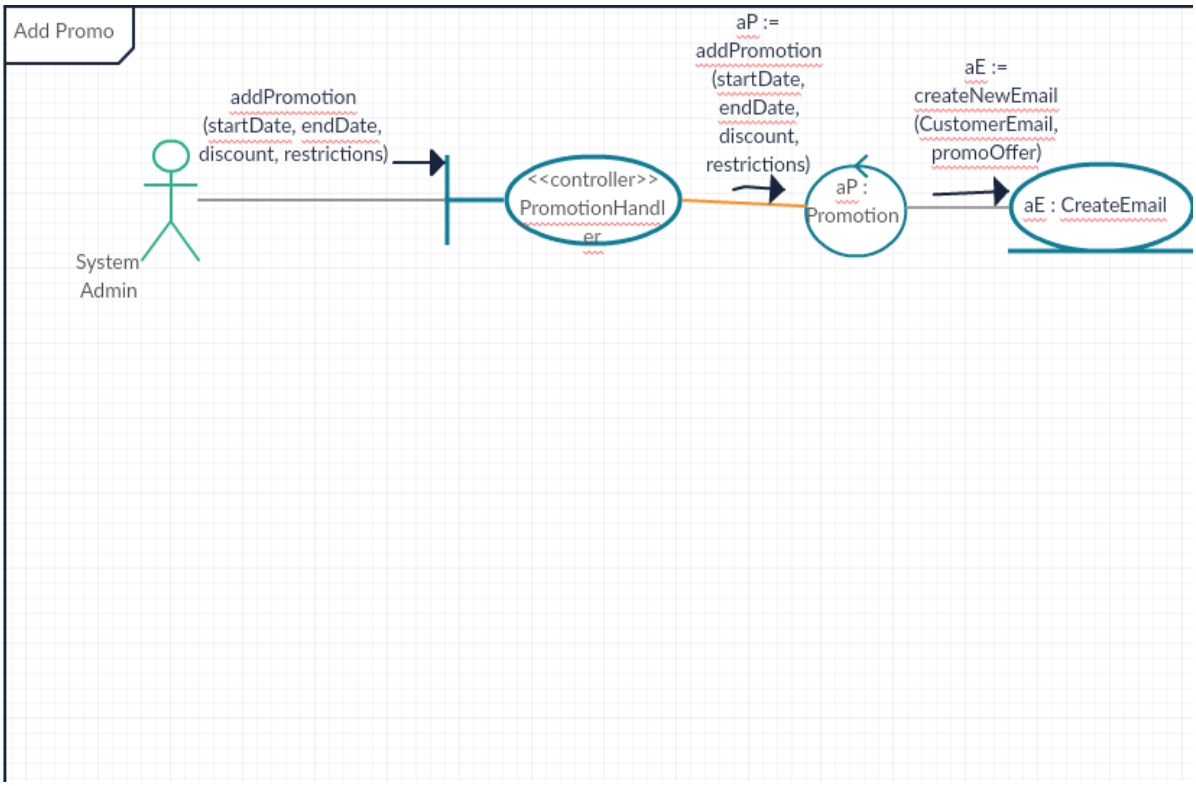
4.5.1 Add New Movie



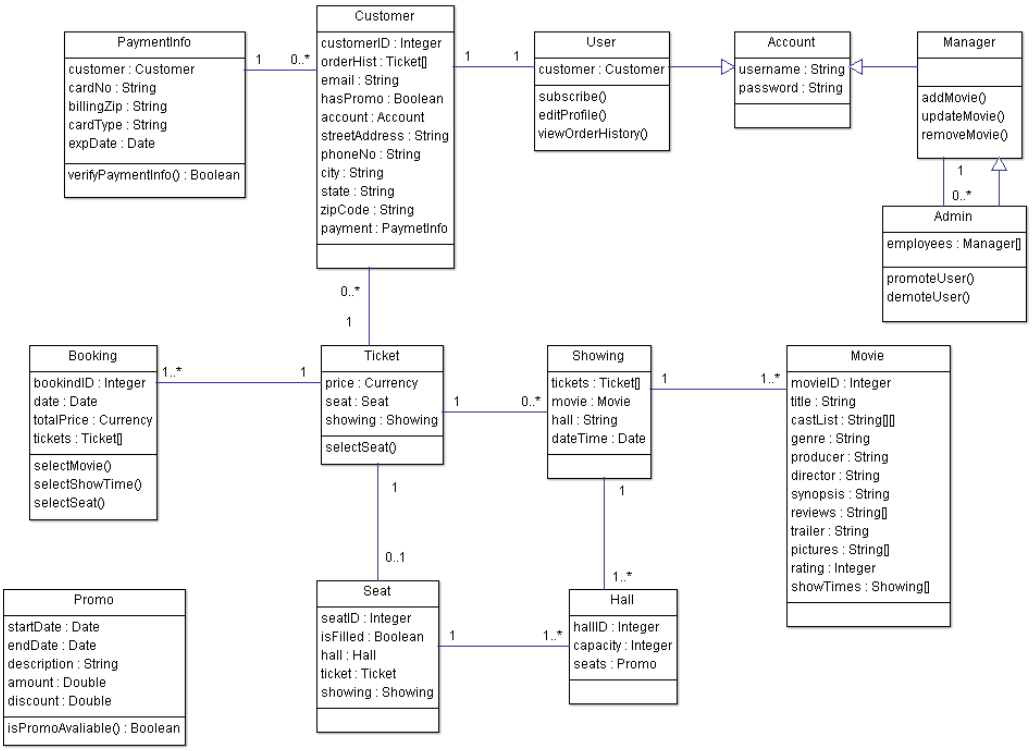
4.5.2 Schedule Movie Times



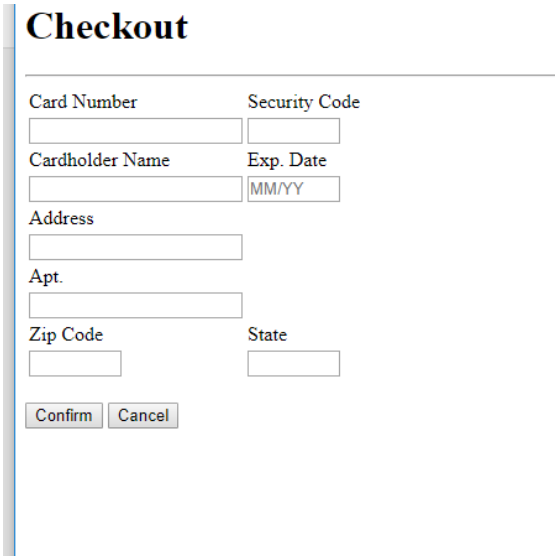
4.5.3 **Add Promo**

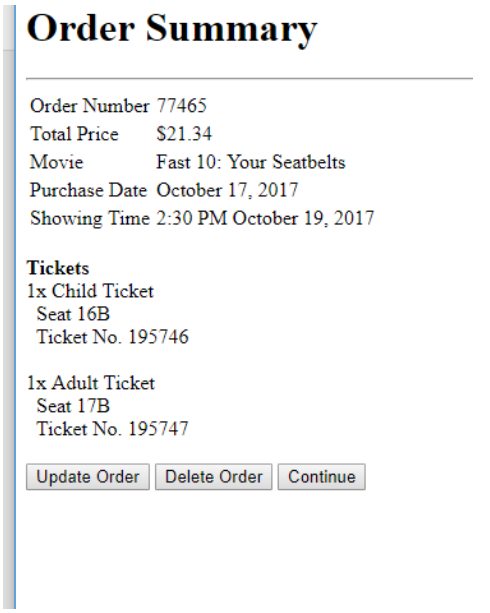


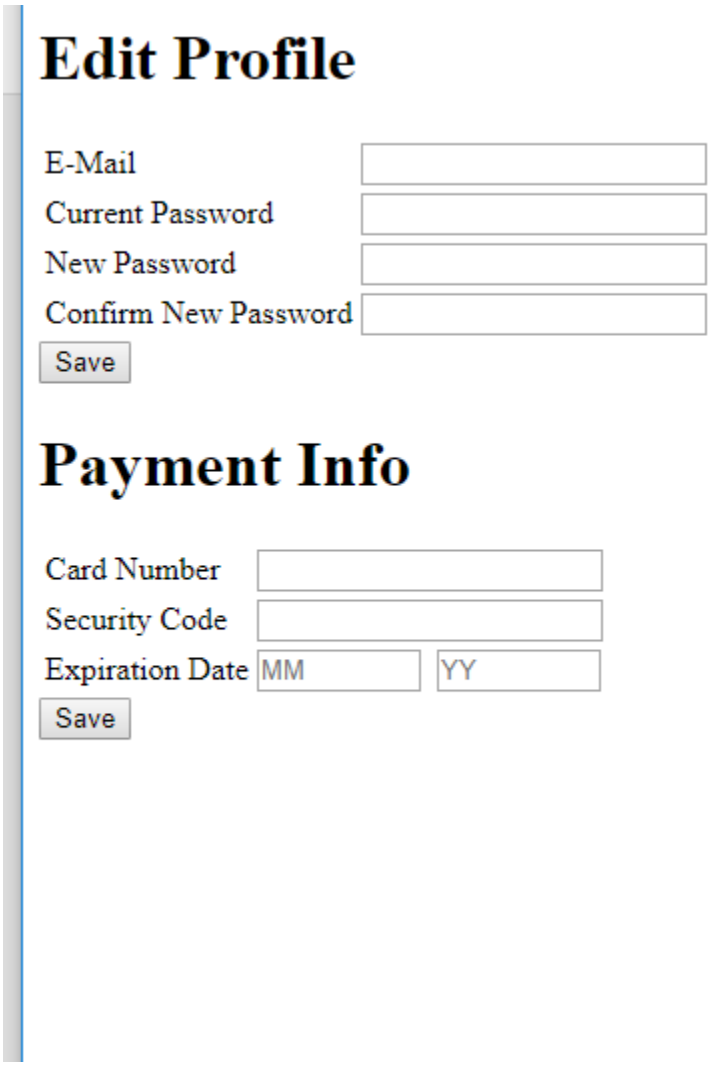
4.6 **Design Class Diagram**



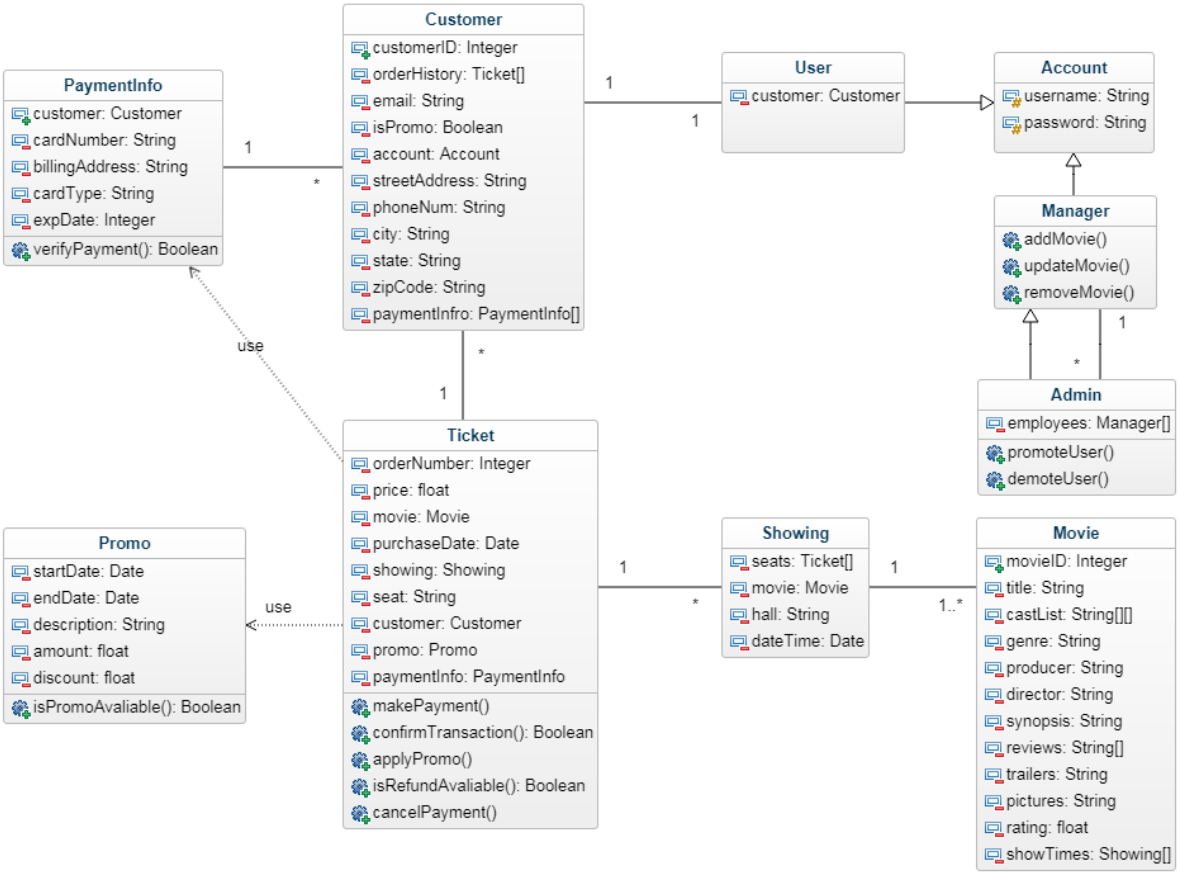
4.7 **User Interface**




Order Summary
Order Number 77465
Total Price $21.34
Movie Fast 10: Your Seatbelts
Purchase Date October 17, 2017
Showing Time 2:30 PM October 19, 2017
Tickets
1x Child Ticket
Seat 16B
Ticket No. 195746
1x Adult Ticket
Seat 17B
Ticket No. 195747
Update Order
Delete Order
Continue


Edit Profile
E-Mail
Current Password
New Password
Confirm New Password
Save
Payment Info
Card Number
Security Code
Expiration Date
MM
YY
Save

4.8 Class Diagram



4.9 Data Dictionary

booking

Table comments: This table contains a record of each "booking" or order made by the customer with 'customer_id'. To find info on each individual ticket in an order, view the tickets in the ticket table with the cooresponding booking_id.

Column	Type	Null	Default	Links to	Comments	MIME
booking_id *(Primary)*	int(11)	No			The ID number generated when a customer books a ticket, like an order number	
customer_id	int(11)	Yes	*NULL*	customer -> customer_id	The ID number generated for a customer, unique for each customer	

date	date	Yes	*NULL*		Date of the viewing of the movie	
num_tickets	int(11)	Yes	*NULL*		Number of tickets ordered	
total_price	double	Yes	*NULL*		Price of the tickets	
movie_id	int(11)	Yes	*NULL*		ID number of the movie, unique for each movie	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	booking_id	0	A	No	
customer_id_idx	BTREE	No	No	customer_id	0	A	Yes	

customer

Table comments: This contains all customer information for REGISTERED customers.

Column	Type	Null	Default	Links to	Comments	MIME
customer_id *(Primary)*	int(11)	No			The ID number generated for a customer, unique for each customer	
email	varchar(45)	Yes	*NULL*		Customer's email address	

has_promo	varchar(45)	Yes	*NULL*		A value storing whether or not the customer is using a promotional offer	
account	varchar(45)	Yes	*NULL*		The customer's account	
street_address	varchar(45)	Yes	*NULL*		The street address of the customer	
phone_number	varchar(45)	Yes	*NULL*		Customer's phone number	
city	varchar(45)	Yes	*NULL*		City where the customer lives	
state	varchar(45)	Yes	*NULL*		State where the customer lives	
zipcode	varchar(45)	Yes	*NULL*		Customer's zipcode	
first_name	varchar(45)	Yes	*NULL*		Customer's first name	
last_name	varchar(45)	Yes	*NULL*		Customer's last name	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	customer_id	0	A	No	

employees

Table comments: contains general employee info.

Column	Type	Null	Default	Links to	Comments	MIME
employee_id *(Primary)*	int(10)	No			ID number for an employee, unique for each employee	
employee_type	int(10)	Yes	*NULL*		Designates what type of employee an employee is	
first_name	varchar(45)	Yes	*NULL*		Employee's first name	

last_name	varchar(45)	Yes	*NULL*		Employee's last name	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	employee_id	0	A	No	

employee_types

Table comments: contains the different types of employees. employee_type will be an identifier (INT) used to find the employee's role. types are as follows: 1 - Admin 2 - Manager 3 - general staff

Column	Type	Null	Default	Links to	Comments	MIME
employee_type *(Primary)*	int(11)	No			Designates what type of employee an employee is	
employee_label	varchar(45)	Yes	*NULL*			

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	employee_type	0	A	No	

Halls

Table comments: The halls are the different rooms to watch the movies

Column	Type	Null	Default	Links to	Comments	MIME
hall_id *(Primary)*	int(11)	No			The ID number of a specific hall, each hall has a unique ID	
capacity	int(11)	Yes	*NULL*		Total number of people allowed in a hall	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	hall_id	0	A	No	

movie

Table comments: movie info stored in this table. Each movie has a unique ID. currently_showing is either yes or no.

Column	Type	Null	Default	Links to	Comments	MIME
movie_id *(Primary)*	int(11)	No			The ID number of a movie, each movie has a unique ID	
movie_name	varchar(45)	Yes	*NULL*		Name of the movie	
cast_list	varchar(500)	Yes	*NULL*		Cast list for a movie	
producer	varchar(45)	Yes	*NULL*		Producer(s) of a movie	
director	varchar(45)	Yes	*NULL*		Director(s) of a movie	
synopsis	varchar(500)	Yes	*NULL*		A brief summary of a movie	
picture	varchar(45)	Yes	*NULL*		A short clip or picture of a movie	
rating	int(11)	Yes	*NULL*		A movie’s IMDB rating and its ESRB rating	
currently_showing	varchar(10)	Yes	*NULL*		Whether a movie is currently showing or not	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	movie_id	0	A	No	

payment_info

Table comments: This is a customer’s payment info, which can be saved to their account

Column	Type	Null	Default	Links to	Comments	MIME
billing_zip	text	Yes	*NULL*		The zipcode for a customer's billing address	
card_type	text	Yes	*NULL*		Type of card a customer is using, such as Mastercard, American Express, etc.	
exp_date	text	Yes	*NULL*		Expiration date of a card a customer is using	
customer_id	int(11)	Yes	*NULL*		The ID number generated for a customer, unique for each customer	
card_number *(Primary)*	int(11)	No			The number on the front of a credit/debit card that designate that specific card	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	card_number	0	A	No	
customer_id_idx	BTREE	No	No	customer_id	0	A	Yes	

Promo

Table comments: This is the promotional offer table. These promotional offers are sent via email to registered users and can be applied when booking tickets for a movie online

Column	Type	Null	Default	Links to	Comments	MIME

description	text	Yes	*NULL*		A description of the promotional offer and any and all restrictions it may have	
promo_id *(Primary)*	int(11)	No			The ID number associated with a specific promotional offer, each offer has a unique ID	
amount_available	int(11)	Yes	*NULL*		The maximum amount the offer can reduce the price by i.e. max savings $20	
discount	int(11)	Yes	*NULL*		The percentage/amount the promotional offer will take off i.e. 20% off next ticket	
start_date	date	Yes	*NULL*		The date the promotional offer first starts	
end_date	date	Yes	*NULL*		The date when the promotional offer ends	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	promo_id	0	A	No	

seats

Table comments: This table contains the seat availability for a specific showing_id. The seats are named 1-50. If a seat is open its value is 0, if occupied it is a 1.

Column	Type	Null	Default	Links to	Comments	MIME
showing_id *(Primary)*	int(11)	No			A showing ID is a unique ID that designates the specific showing of a movie. This is different from a booking ID in that a booking ID is similar to an order number, while a showing ID is a specific number that designates the hall, the time, and the date of a specific showing	
1	int(11)	Yes	*NULL*		Seat numbers for each showing	
2	int(11)	Yes	*NULL*			
3	int(11)	Yes	*NULL*			
4	int(11)	Yes	*NULL*			
5	int(11)	Yes	*NULL*			
6	int(11)	Yes	*NULL*			
7	int(11)	Yes	*NULL*			
8	int(11)	Yes	*NULL*			
9	int(11)	Yes	*NULL*			
10	int(11)	Yes	*NULL*			
11	int(11)	Yes	*NULL*			
12	int(11)	Yes	*NULL*			
13	int(11)	Yes	*NULL*			
14	int(11)	Yes	*NULL*			
15	int(11)	Yes	*NULL*			
16	int(11)	Yes	*NULL*			
17	int(11)	Yes	*NULL*			
18	int(11)	Yes	*NULL*			
19	int(11)	Yes	*NULL*			
20	int(11)	Yes	*NULL*			
21	int(11)	Yes	*NULL*			
22	int(11)	Yes	*NULL*			
23	int(11)	Yes	*NULL*			
24	int(11)	Yes	*NULL*			
25	int(11)	Yes	*NULL*			

26	int(11)	Yes	*NULL*			
27	int(11)	Yes	*NULL*			
28	int(11)	Yes	*NULL*			
29	int(11)	Yes	*NULL*			
30	int(11)	Yes	*NULL*			
31	int(11)	Yes	*NULL*			
32	int(11)	Yes	*NULL*			
33	int(11)	Yes	*NULL*			
34	int(11)	Yes	*NULL*			
35	int(11)	Yes	*NULL*			
36	int(11)	Yes	*NULL*			
37	int(11)	Yes	*NULL*			
38	int(11)	Yes	*NULL*			
39	int(11)	Yes	*NULL*			
40	int(11)	Yes	*NULL*			
41	int(11)	Yes	*NULL*			
42	int(11)	Yes	*NULL*			
43	int(11)	Yes	*NULL*			
44	int(11)	Yes	*NULL*			
45	int(11)	Yes	*NULL*			
46	int(11)	Yes	*NULL*			
47	int(11)	Yes	*NULL*			
48	int(11)	Yes	*NULL*			
49	int(11)	Yes	*NULL*			
50	int(11)	Yes	*NULL*			

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	showing_id	0	A	No	

showings

Table comments: This table contains all showtimes for

Column	Type	Null	Default	Links to	Comments	MIME
showing_id *(Primary)*	int(11)	No			A showing ID is a unique ID that designates the specific showing of a movie. This is different from a booking ID in that a booking ID is similar to an order number, while a showing ID is a specific number that designates the hall, the time, and the date of a specific showing	
movie_id	int(11)	Yes	*NULL*	movie -> movie_id	ID number of the movie, unique for each movie	
hall_id	int(11)	Yes	*NULL*	halls -> hall_id	The ID number of a specific hall, each hall has a unique ID	
num_tickets_available	int(11)	Yes	*NULL*		The number of tickets available for a specific showing, i.e. if max capacity for a hall is 50 seats, and 20 seats have already been reserved, then num_tickets_available will be 30	
date	date	Yes	*NULL*		Date of a specific showing	
time	time	Yes	*NULL*		Time the showing will start and end	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	showing_id	0	A	No	
movie_id_idx	BTREE	No	No	movie_id	0	A	Yes	
hall_id_idx	BTREE	No	No	hall_id	0	A	Yes	

ticket

Table comments: types: 0 - child 1 - adult 2 - senior

Column	Type	Null	Default	Links to	Comments	MIME
booking_id *(Primary)*	int(11)	No			The ID number generated when a customer books a ticket, like an order number	
type	int(11)	Yes	*NULL*		Type of ticket, such as adult, child, or senior	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	booking_id	0	A	No	

ticket_type

Table comments: 0-child 1-adult 2-senior

Column	Type	Null	Default	Links to	Comments	MIME
ticket_id *(Primary)*	int(11)	No			ID number specific for each ticket.	
price	double	Yes	*NULL*		Price of a ticket before promotional offers, including tax	

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	ticket_id	0	A	No	

4.10 Test Cases

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
001	Check customer login	1. Go to site 2. Enter UserID 3. Enter password 4. Click Submit	Userid= cbc55695@uga.edu Password= wre	Successful login	Successful login	Pass
002	Check Registration	1. Go to site 2. Go to create account 3. Fill in information 4. Click register 5. Receive Verification email 6. Enter verification ID	Email= brogangrey@gmail.com Password = qazxswedc First name= John Last name= Smith Phone = 0000000000 Address= 123 Abcdefg Street City = Athens State = Georgia Zipcode = 30605 Promo = no	Registration successful	Registration Successful	Pass
003	Search Movie	1. Go to site 2. search for movie	Movie = Black Panther	Displays showing screen	Displays Showing Screen	Pass
004	Choose Seat	1. Go to site 2. Search for movie 3. Choose viewing time for movie 4. Choose seat for	Movie = Black Panther Viewing time = 1:00pm Seat = Seat 1	Shows payment screen with one seat selected	Shows payment screen with one seat selected	Pass

		viewing time				
005	Employee Login	1. Go to site 2. Go to employee login 3. Enter info 4. Hit Submit	Employee uname = Steve@steve.com Employee Password = uhyuhy	Successful Login	Successful Login	Pass