Theater Ticketing System

03.01.2023

—

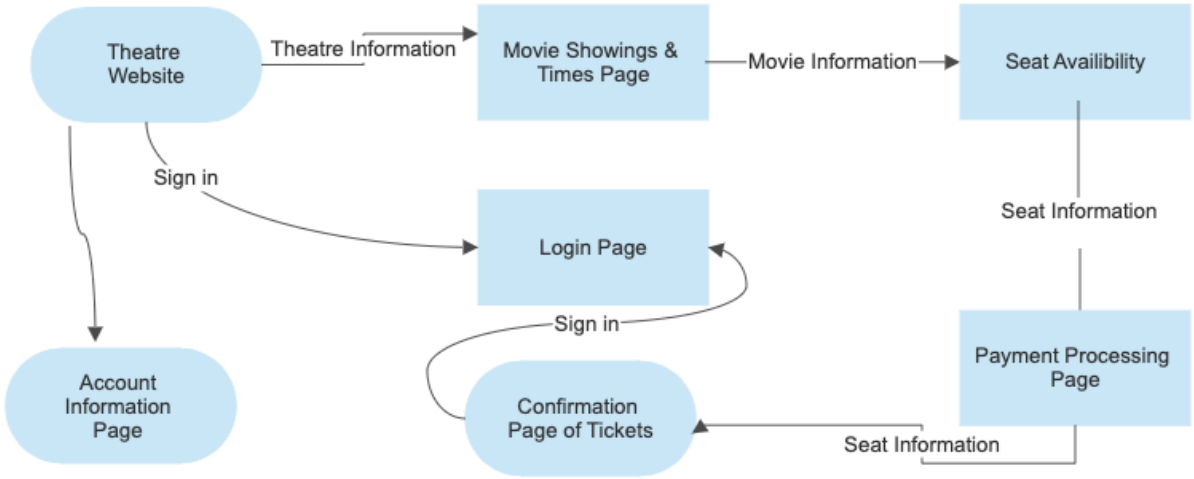
Steven Gervacio, Jason Lam, Tri Pham
CS 250
Dr. Hanna

System Description

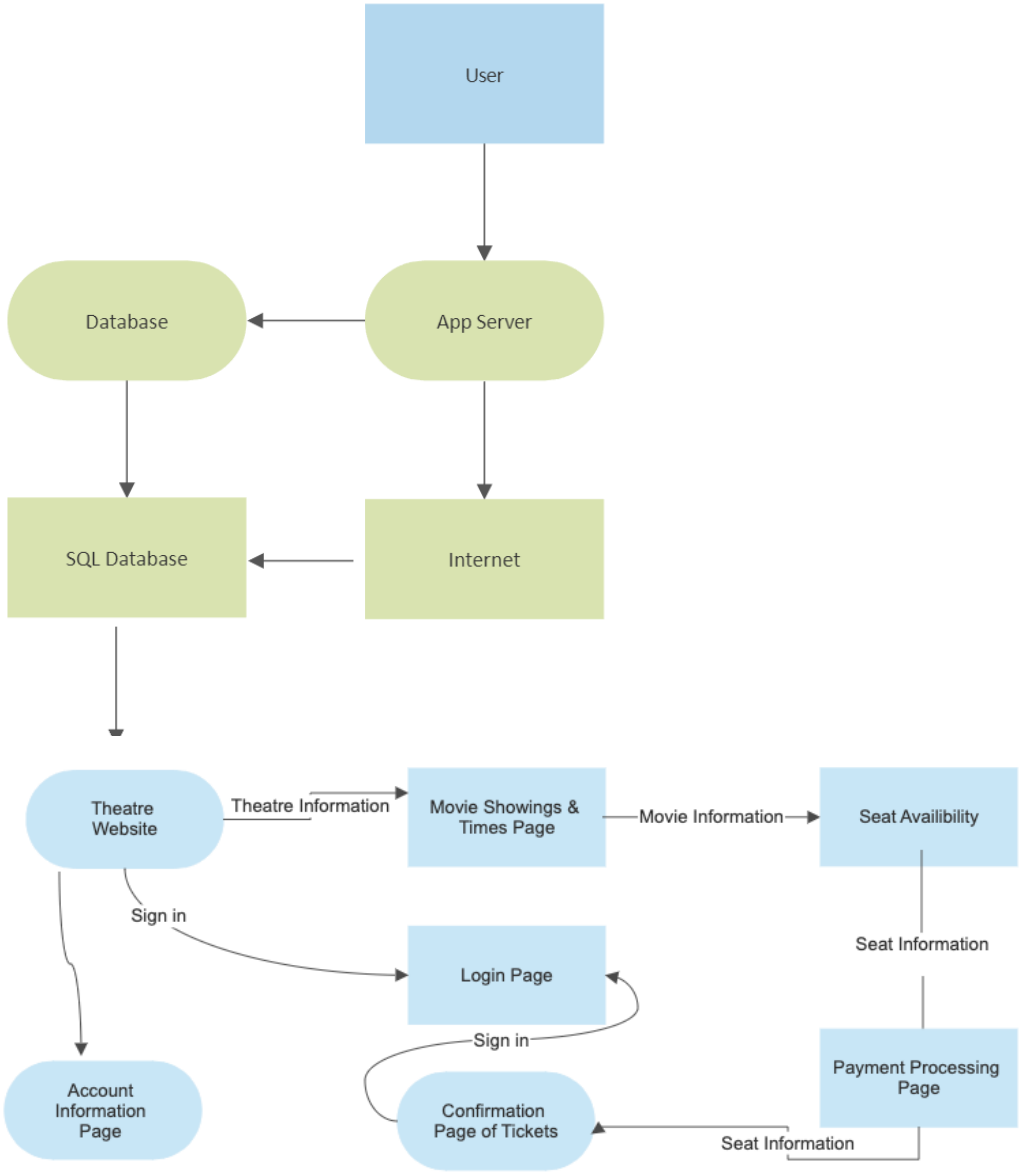
The theater ticketing system is a browser application that we can use while online. The intention for this system is to allow users to buy movie tickets. This system will have all of the necessary information of a movie theater like showings, times, availability, and prices. The website will allow users to choose what category they fall into like child, adult, senior, military, etc. Upon choosing this it will change the prices of the movie tickets. The user will be able to choose how many tickets they want to purchase, where the seats will be, and what movie they want to watch. After all of this information is selected, they will be able to pay by entering in their information and will lead them into a confirmation of their purchase. Users will be able to sign in or make an account in congruence with the website for rewards and points. Since this system will have to be able to take in a lot of information and has to be able to handle millions of users, the tickets must not repeat. The budget for this system should be thousands of dollars.

Software Architecture Overview

Architectural Diagram:



Architectural Diagram (Updated) :

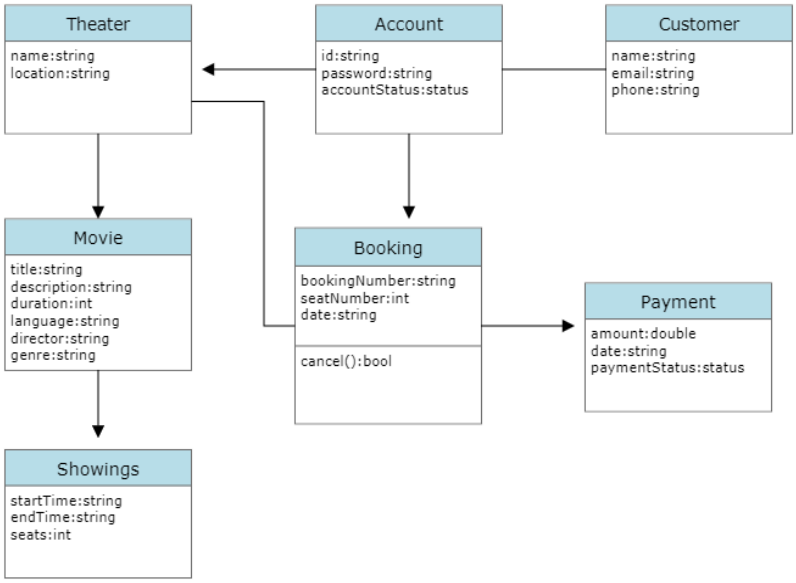


Architecture Diagram Updates:

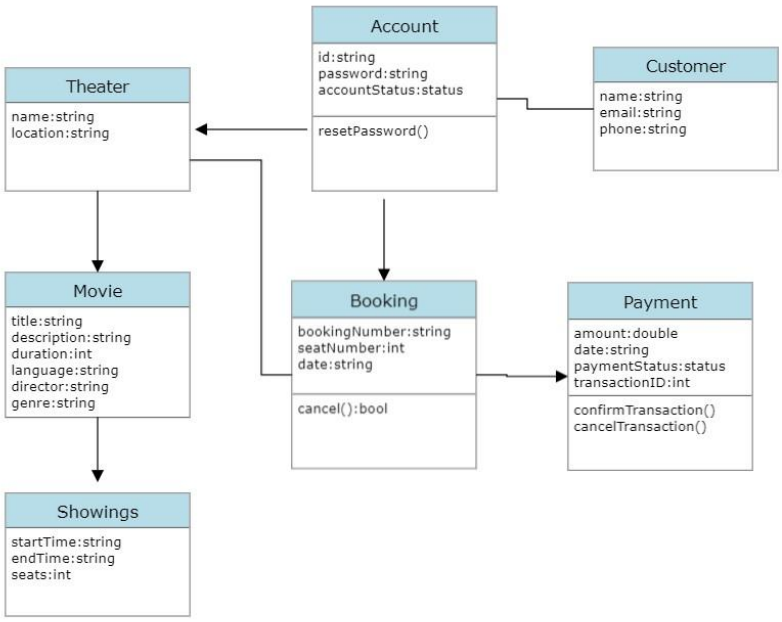
Added database to design to store all the necessary information and data that relate to the movies, showtimes, and such. Databases will also be used to store the information about customers also such as past purchases and account information. We chose to use a single SQL database as it can handle complex loads regarding tickets, customer data, and theater operations. It will also provide a reliable and secure way to store our data. So from the user device, they connect to the app server through the use of the internet and then they can

access the database which has all the information about the customer account login as well as information about movies and such.

UML Class Diagram



UML Class Diagram (Updated):



Data Management Strategy:

For this data management strategy we chose databases that will consist of one SQL database. The data that we will take in will be structured and is user input data. The databases we will use are for information taking. With this, the data will be structured meaning it is best to store the data specifically according to their category. An example of some data in this theater ticketing system is when a user logs into their account. This would require constant change and adding data which is one of the features that SQL provides in the database. The data will then split depending on the user input so that it can figure out each different data from each other. Since the data varies it will need to be reviewed before it can be stored so that the data can be grouped logically. In the theater ticketing system two different data that would need to be stored are customer data and movie tickets.

Trade-off Discussion:

Since we are using a single SQL database to handle complex loads of data there are advantages and disadvantages. The advantages of it being the data is centralized and is stored all in one centralized location. Another being scalability, the single database can handle large amounts of data and traffic. Finally, the use of a single database can be less expensive as opposed to one with multiple. The disadvantages when it comes to using a single SQL is it having a single point of failure, meaning that if the database fails then the applications will also fail. Another being security risks, since it is just a single database it leads to vulnerability because of security breaches and attacks all at one location when the data is held. Lastly, it can have limited flexibility due to not being able to accommodate different data structures and data.

Description of Diagrams

Architecture Diagram:

This architecture diagram that is provided is what we made for the Theatre Ticketing System. In this diagram we provided what it will look like when the user interacts with the website. The main part of this system is the website which controls all of the functions of buying a movie ticket. When clicking on the theater website, you will have the option of looking at all of the available movie showings and times. You will also have an option to sign in or make an account. When you have selected a movie and time you will get redirected to the seating page which will show the available seats and how many seats you

would like. After selecting your seats, you will be able to enter your information to pay for the tickets you have chosen. Lastly, you will go to the final page which is the confirmation of your tickets. You will have the option to sign into your account or create an account. There will also be an account information page where you have access to update or complete any changes to your account in the theater website.

Description of UML Classes

Theater Class: This class stores the name and location of the theater customers are looking for

- **Name:** Stores the name of the theater in a string
- **Location:** Stores the physical location of the theater in a string

Movie Class: This class stores attributes of the movie in order to find what the customer is looking for

- **Title:** Stores the name of the title of the movie in a string
- **Description:** Stores the description of all movies in a string
- **Duration:** Stores the length of the movie in minutes in an int
- **Language:** Stores what language the movie is in in a string
- **Director:** Stores who directed the movie in a string

Showings Class: This class stores attributes of the showtimes and seats

- **startTime:** Stores the start time of the movie in a string
- **endTime:** Stores the end time of the movie in a string
- **Seats:** Stores how many seats there are in a int value

Account Class: This class stores the attributes of the customer's account

- **ID:** Stores the ID of the user in a string
- **Password:** Stores the password of the user account in a string
- **AccountStatus:** Checks status of the account
- **resetPassword() :** allows user to reset password if need

Customer Class: This class stores the attributes of the customer

- **Name:** Stores the name of the customer in a string
- **Email:** Stores the email address of the user in a string
- **Phone:** Stores the phone number of the user in a string

Booking Class: This class stores the booking attributes that the users put in

- **BookingNumber:** Stores the code of the unique booking number for the customer in a string
- **SeatNumber:** Stores the seat number that customer has selected in a int value
- **Date:** Stores the date of booking in a string
- **cancel():bool:** Allows the user to cancel the booking if needed

Payment Class: This class stores the details of the payment made by the customer

- **Amount:** Stores the payment amount in a double value
- **Date:** Stores the date of the purchase in a string
- **PaymentStatus:** Stores the status of the payment to see if it is successful or not
- **ConfirmTransaction():** Allows user to confirm if the payment is correct
- **CancelPayment():** Allows user to cancel the payment

UML Diagram Updates

Here are the updates we made to our UML Diagram:

- Added resetPassword() method to the account class to allow users to reset their password if needed.
- Added a confirmTransaction() method to the payment class so that users can make sure that their payment is final before purchasing.
- Added a cancelPayment() method to the payment class so that users can cancel their payment if needed to.

These updated functions allow the user to have more options on choosing what to do like canceling their payment or confirming their transaction.

Description of Attributes and Operation

The theater class stores the name and location of the theater. Such classes include the movie class which stores attributes of the movie in order to find what the customer is looking for. The operation of the system will include many different user interactions depending on what the user wants. Functions like showings will be used throughout the program to see the available movie showtimes and seats. There will be functional requirements such as accounts class which will let the user choose to make any changes to their account. Throughout our software we will be implementing strings, booleans, integers and other data types that will be stored in the various classes.

Development Plan and Timeline

I. Partitioning of Tasks

Steven Gervacio- Github repository, Architectural Diagram, Description

Jason Lam- UML Class Diagram, Description

Tri Pham- Overview of System, Description

II. Team Member Responsibilities

Steven- Layout of System, Description of Attributes and Operations

Jason Lam- Description of Classes, Diagrams

Tri Pham- Description of Overview

Verification Test Plan

We start the test sets by testing the ticket pricing which displays the correct pricing of tickets. To start the testing process, we go to the ticket purchasing page, select which show and date in addition to ticket and quantity, then we verify the total price based on the quantity and ticket type. In that way, we can confirm the status of the transaction. We would then compare the prices displayed on the website to the actual prices provided by the theaters and make changes based off of that. The failures that we are looking for include wrong or not updated prices being shown to the user. We will also be testing to see if movie prices can be updated in real time. Another one we can test on is the login function which helps the user to login to their account that can be used to purchase tickets on the chosen theater. To test if this works, we start by clicking on the login page of the website, enter the valid username and password, and press the login button. In that way, users should be able to login into their respective account and buy tickets for the movie. The login process should also scan if the user has an account on the website or not so they would be able to sign up. Some failures that we would look out for include errors on logging in when using a valid account as well as testing the ability of the reset password method to allow users to troubleshoot their account. Users should be able to seamlessly log in and out of their accounts using their account credentials and be able to change the password if needed. Other tests that will be run include the selection of movies, seating, and payment which will ensure that users will have a smooth and error free experience when using our movie ticketing system.