Ticket Sales Anywhere

Software Design Specification

v1.0.0

5/31/2024

Group 3

Lexa Rao, James Ardilla, Eli Lopez

System Description

This software system will be used as both a ticketing and customer service application for a movie theater. The client has requested that this software is a web-based application, that will accept both credit cards as well as bitcoin. The client has additionally requested that the customer side of the software features a queue system for crowd control as well as a customer reward system. Focusing on the queue system, it should manage in demand movies and block detected bots from purchasing tickets to these movies.

The customer-service side of the application should be able to accept customer feedback in the form of a 5-star style system. The clients' workers should be able to enter an “admin” mode that allows for them to perform customer service actions such as refunds or account support.

Since this software has two parts, we will be using a hybrid architecture pattern. This is because we must combine the event driven pattern and the microservice pattern. The event driven pattern will be used for the front-end website features like previews and the rating system. The microservice pattern will be used for the payment processing systems and well as anti-bot measures.

Software Architecture Overview

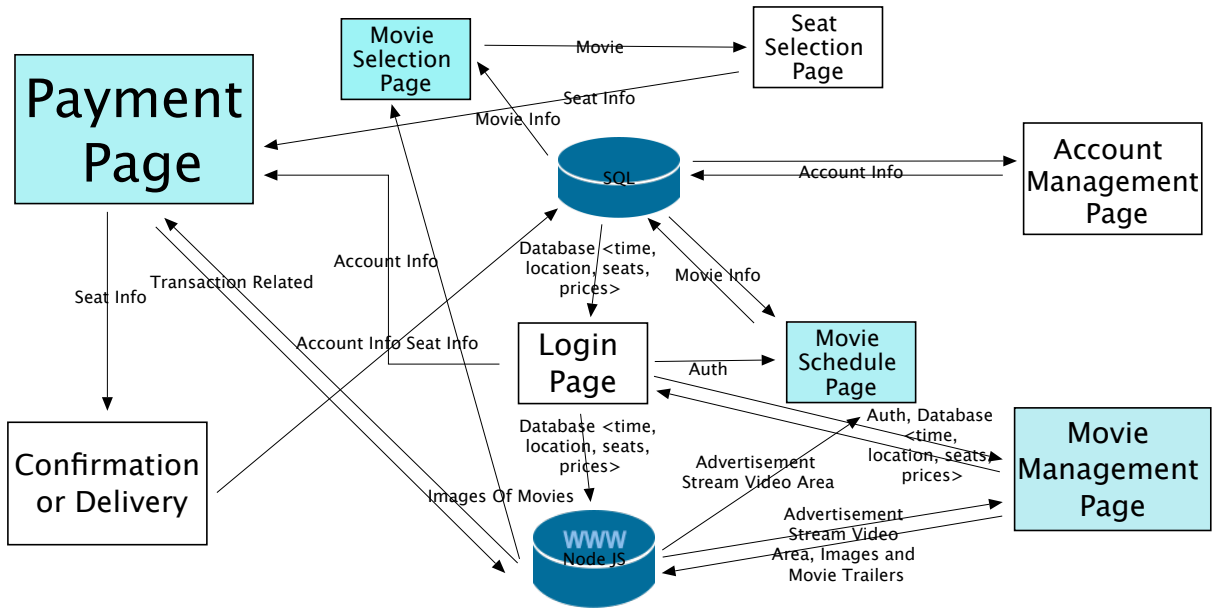


Diagram of the Architecture for Ticket Sales Anywhere and Ticket Sales Assistant

This is a diagram of what the Ticket Sales Anywhere and Ticket Sales Assistant client's architecture should look like. The diagram has the SQL servers which are responsible for storing information about the Time, location, seats, and movie prices. The Node JS server is responsible for serving movie images of movies currently playing to the movie selection screen, and clips to the user on the movie selection page which could be seen as the greeting or welcome screen that also displays a list of currently playing movies within it. The Payment Page has information being sent to it from the Node JS server because PayPal and Bitcoin transactions require a server to be used while making payments and keeping track of them. All three of these screens have blue boxes to show they are using information from the Node JS server that is being used in their code. The movie management page is only logged into when an administrator logs in and displays the Ticket Sales Assistant UI which is what allows administrators to manage movies. It also uses the Node JS backend so it can change data in it and manage available movies.

Login Page:

Displays login information for users when they get to the website.

Movie Schedule Page:

A page that displays options for current movies that are playing as well as to schedule the movie at a specific theater and date.

Account Management Page:

A separate page a user can get to when pressing a button somewhere on the top of the UI. It allows users to configure their accounts and manage their information.

Seat Selection Page:

A page that allows the user to select which seat they are currently sitting at. It displays the row of seats available.

Movie Selection Page:

Displays information about the current movies available and allows you to select the one you are choosing to watch within the theater.

Payment Page:

Allows you to select a payment option. It will include a PayPal payment and a Bitcoin Payment area.

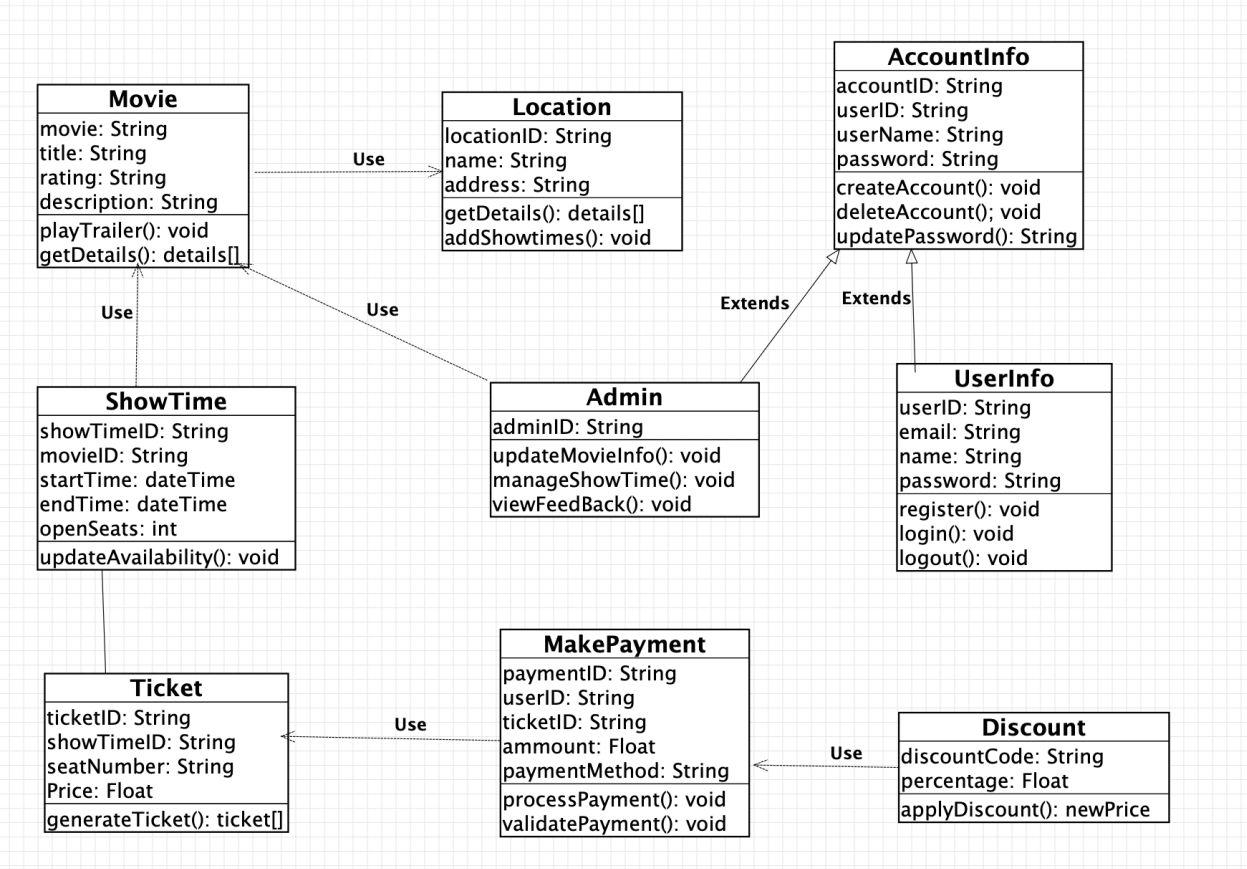
Confirmation and Delivery Page:

A ticket will be emailed to the user that will have more information on the ticket they just purchased.

Movie Management Page:

Uses the Ticket Sales Assistant UI and is responsible for managing movies currently playing. It allows them the upload new movies and delete old ones from the currently playing list when they are no longer playing. It also has a pop-up website that can be maximized and displays the Advertisement Stream Video Area.

UML CLASS DIAGRAM:



Explanations of UML Diagram: The UML Class Diagram represents a movie ticket booking system where movies are associated with locations and showtimes. Users can register and manage their accounts, purchase tickets, make payments, and apply discounts. Admins have special privileges to manage movie information and showtimes.

Overview of Classes: The classes in the UML Diagram are Movie, Location, AccountInfo, Admin, UserInfo, ShowTime, Ticket, MakePayment, and Discount.

Class Descriptions

Movie

Attributes:
movie: String
title: String
rating: String
description: String

Operations:
playTrailer(): void - Plays the trailer of the movie.
getDetails(): details[] - Retrieves the details of the movie.

Location

Attributes:
locationID: String
name: String
address: String

Operations:
getDetails(): details[] - Retrieves the details of the location.
addShowtimes(): void - Adds showtimes to the location.

AccountInfo

Attributes:
accountID: String
userID: String
userName: String
password: String

Operations:
createAccount(): void - Creates a new account.
deleteAccount(): void - Deletes an account.
updatePassword(): String - Updates the password for the account.

Admin (Extends AccountInfo)

Attributes:
adminID: String

Operations:
updateMovieInfo(): void - Updates information about movies.
manageShowTime(): void - Manages showtimes for movies.
viewFeedBack(): void - Views feedback from customers.

UserInfo (Extends AccountInfo)

Attributes:
userID: String
email: String
name: String
password: String

Operations:
register(): void - Registers a new user.
login(): void - Logs in an existing user.
logout(): void - Logs out the user.

ShowTime

Attributes:
showTimeID: String
movieID: String
startTime: dateTime
endTime: dateTime
openSeats: int

Operations:
updateAvailability(): void - Updates the availability of seats for the showtime.

Ticket

Attributes:
ticketID: String
showTimeID: String
seatNumber: String
Price: Float

Operations:
generateTicket(): ticket[] - Generates a ticket for the showtime.

MakePayment

Attributes:
paymentID: String
userID: String
ticketID: String
amount: Float
paymentMethod: String

Operations:
processPayment(): void - Processes the payment for the ticket.
validatePayment(): void - Validates the payment information.

Discount

Attributes:
discountCode: String
percentage: Float

Operations:
applyDiscount(): newPrice - Applies a discount to the ticket price.

Explanation of Relationships

Use Relationships:
Movie uses Location.
Movie uses ShowTime.
Ticket uses MakePayment.
MakePayment uses Discount.

Associations:
ShowTime associated with Ticket.

Generalization Relationships:
Admin extends AccountInfo.
UserInfo extends AccountInfo.

Development Plan and Timeline

Partitioning of Tasks:

Account Management
Payment System/Payment Processing
Ticketing
Movie Managment
Security/Queue System
Verification/Inspection

Estimated Timeline of Tasks:

Based on the client budget of 500k and our current team size of 3 people, we estimate a total development time of about 7 months for the project. This timeline includes the build time, the time for verification of the application, and post launch feedback.
Estimated time for each of the above tasks is approximately 1 month for each task with an additional month and a half for quality control and for client feedback.

Below is a high-level overview of the development timeline. For now, this timeline is monthly and can be changed based on team feedback.

- Month 1: Initial Planning and Requirements Gathering
- Month 2: Dev Team Design Period
- Month 3: Development – Front End
- Month 4: Development – Back End
- Month 5: Testing and QA
- Month 6: Bug Fixing and Product Launch
- Month 7: Post-Launch and Maintenance

Team Member Responsibilities:

James - Verification/Inspection, Security
Lexa - Author – Account Management, Payment Processing
Eli - Author – Security, Ticketing, Movie Management System