Online Theater Ticketing Software Design Specification

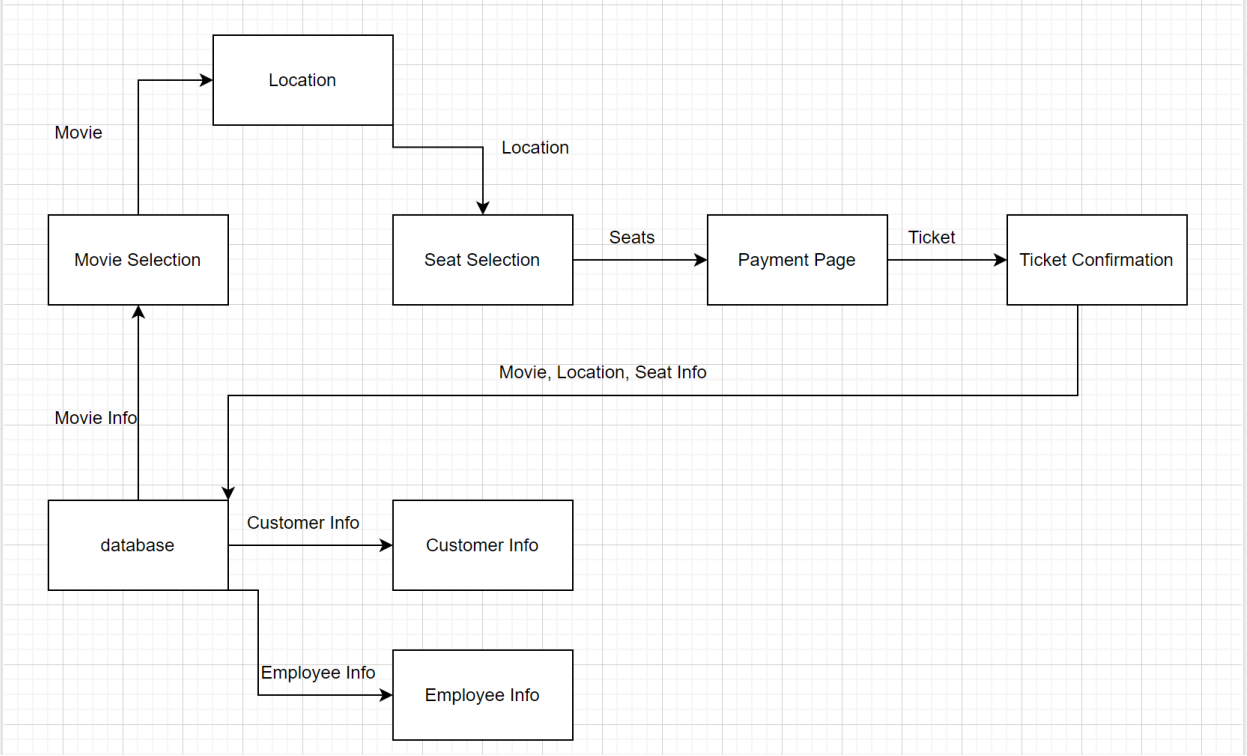
By: Angelina Mom, Brandon Slusser, Vincent Huynh, and Conor Murphy

https://github.com/amom1053/Software-Design-Specification

System Description (Brief overview of system)

The purpose of this system will be to choose a movie, then pick out a seat, and to pay for the ticket. Then, the user will be sent the ticket to their email. The system will be intended for movie theaters to have the customers go through the entire process of getting their tickets online. In this system we will be building classes with attributes along with operations in order for a customer to pick out and pay for a ticket in an efficient manner. The steps for the entire process will follow a straight line with the selection of the movie directly moving to the seat choice which will go to the payment process which will finish with a ticket confirmation.

Architectural diagram

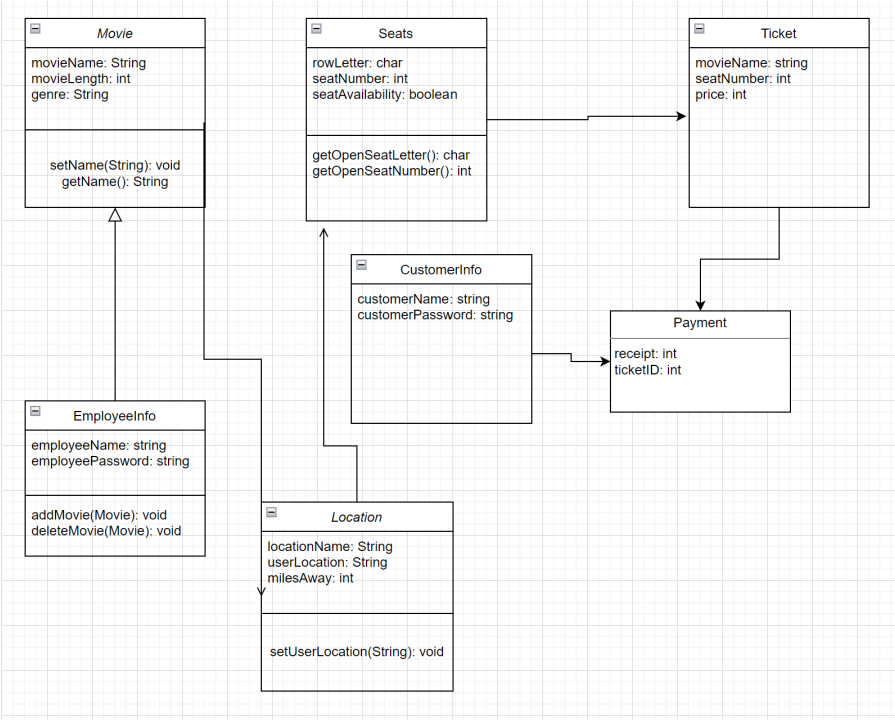


https://drive.google.com/file/d/1GczHMnxtBrQLPHaztREFe_aUGopSOfZG/view?usp=sharing

Architectural Diagram Description

The architectural diagram shows how a normal customer would view the system. All customers would start at the movie selection page. After they choose a movie it would move on to the location page to choose the location of the theater. Then to the seat selection page so they can choose their seat. After choosing a seat it would show the payment page. After purchasing the ticket, they will receive a ticket confirmation page. A customer cannot access the location, seat, payment, and ticket confirmation page without first clicking on the movie selection page. After the ticket is confirmed, that information goes back into the database and is updated so another customer cannot buy the same seat. The customer gets access to the movie selection page through the database which also has access to the customer and employee information.

UML Class Diagram



https://drive.google.com/file/d/14T24Y2ZmfYA9v__EwmtIC1d0nvIHJule/view?usp=sharing

Description of classes:

Movie Class:

-Parent Class to Employee Info class, derived from Seats class

-Has three variables: movieName(String), movieLength(int), and genre(String)

-Has two functions setName(String) and getName()

Seats Class:

-Parent class to Movie class, derived from Ticket class

-Has three variables: rowLetter(char), seatNumber(int), and seatAvailabiltiy(boolean)

-Has two functions getOpenSeatLetter() and getOpenSeatNumber()

Ticket Class:

-Parent class to Seats class, derived from Payment class

-Has three variable: movieName(String), seatNumber(int), price(int)

Payment Class:

-Parent class to Ticket Class and Customer Info Class

-Has two variables: receipt(int) and ticketID(int)

Customer Info Class

-Derived from Payment class

-Has two variables: customerName(String) and customerPassword(String)

Employee Info Class:

-Derived from Movie Class

-Has two variables: employeeName(String) and employeePassword(String)

-Has two functions: addMovie() and deleteMovie()

Location Class:

-Has three variables: locationName(String), userLocation(String), milesAway(int)

-Has one function: setUserLocation(String)

Description of attributes:

Movie Class:

- movieName this is what the name of the movie is and it is a String data type.
- movieLength this is how long the movie is and it is represented as an integer.
- genre this is what genre the movie belongs to and it is a String.

Seats Class:

- rowLetter is a character data type that is used to hold the row letter that a given seat is in.
- seatNumber this is the seat number for a given seat and it is represented as an integer.
- seatAvailability this attribute determines whether a seat is available or not and it is a boolean data type.

Ticket Class:

- movieName from the movie class this attribute says what the name of the movie is and is a string data type.
- seatNumber from the seats class this attribute says what the seatNumber is for a given seat and is an integer.

- price this is an integer data type that says what the price is.

Payment Class:

- receipt this attribute is an integer and has the receipt for the payment.
- ticketID is an integer data type that holds the ticket identification.

Customer Info Class

- customerName this attribute is a string that holds the name of the customer.
- customerPasswords holds the customer's password and is represented as a string data type.

Employee Info Class:

- employeeName this attribute holds the name of the employee and is a string data type.
- employeePassword is a string that holds the password for an employee.

Location Class:

- locationName this is the location of the nearest theater to the user and is a string data type.
- userLocation this is the name of the user's location and is a string data type.
- milesAway is the distance from the user's location and the nearest theater and it is an int data type.

Description of operations

Movie Class:

setName(String): void – Sets the variable MovieName based on the user input from getName()

getName(): String –Gets the name of the movie from the user.

Seats Class:

getOpenSeatLetter(): char -- Returns an open seat letter from the user.

getOpenSeatNumber(): int -- Returns an open seat number from the user

Employee Info Class:

addMovie(Movie): void – Takes movie as a parameter, and adds it

deleteMovie(Movie): void – Takes movie as a parameter, and deletes it

Location Class:

setUserLocation(String): void - Sets the user locations as a string

Development plan and timeline

When it came to distributing the work, we first decided to each do a small section of it. And if anyone wanted to take on more work they could. The estimated time it took for each of us to do all the work was a few hours. Then we all committed the work to github before the deadline.

Angelina Mom: Architectural Diagram, UML Class Diagram, Architectural Diagram Description, Test Cases and Verification Test Plan

Brandon Slusser: Description of Attributes and Verification Test Plan

Vincent Huynh: System Description and Test Cases

Conor Murphy: Class Description, Attributes of Operations, and Test Cases

Verification Test Plan

Ticket Purchase Management

Tests Related: Ticket_Purchase_Module, Ticket_Removal_Module, Credit_Card_Invalid

The Ticket_Purchase_Module tests that when a user purchases a ticket that they receive it in their email. The Ticket_Removal_Module tests whether the user is able to cancel a ticket in their cart before they purchase the ticket. The Credit_Card_Invalid tests that when a user purchases a ticket with an invalid credit card, they are informed of error. All these modules test whether or not the website is able to properly manage the ticket when it comes to its purchase.

Ticket Cannot be Purchased Management

Tests Related: Age_Requirement_Movie_Module, Movie_Sold_Out_Module, Movie_Started_Module

The Age_Requirement_Movie_Module tests that when a user is under 17 years old, they are informed that they cannot purchase a ticket for a NC-17 movie. The Movie_Sold_Out_Module tests that when a customer attempts to buy a movie ticket for a full theater, they are informed that no tickets are available. The Movie_Started_Module tests that a customer is not able to buy a movie ticket of a movie that has already started. All these modules test whether or not the website is able to properly function when it comes to situations of not being able to purchase the ticket.

Design Specification Diagram and Description Update:

The architectural diagram was updated to include the database where all the websites' information is stored and updated. I also added location, employee info, and customer info to the diagram. The UML class diagram was updated to include a location class, which has the variables of locationName(String), userLocation(String), milesAway(int) and the function of setUserLocation(String).