Watchalong

Software Design Specification

4.0

2/29/2024

Group 11

Manan Shukla

Dylan Rambo

Micah Miller

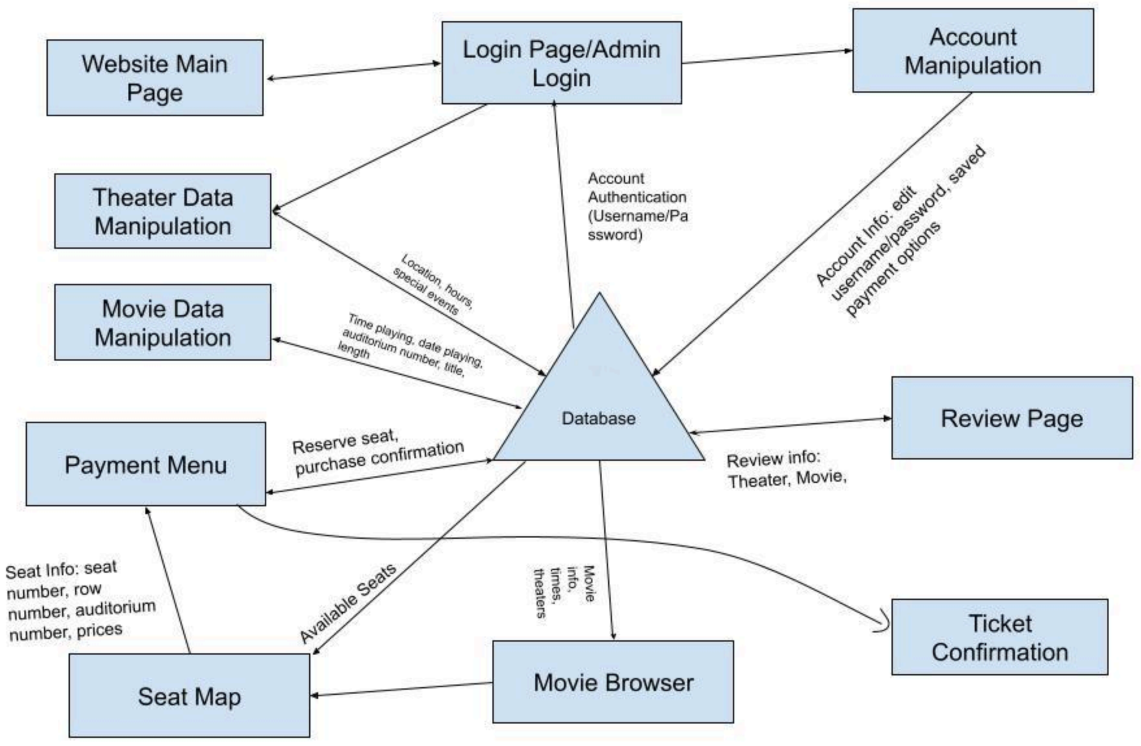
Prepared for
CS 250- Introduction to Software Systems
Instructor: Gus Hanna, Ph.D.
Spring 2024

1. System Description

The entire software system consists of multiple pages for the user to view and use, as well as a database that will be heavily utilized. These pages include a website main page, login page for regular customers and theater admins, account manipulation to edit account information, a movie browser, movie seat map, payment menu, ticket confirmation, review page, and theater/movie data manipulation for admins.

2. Software Architecture Overview

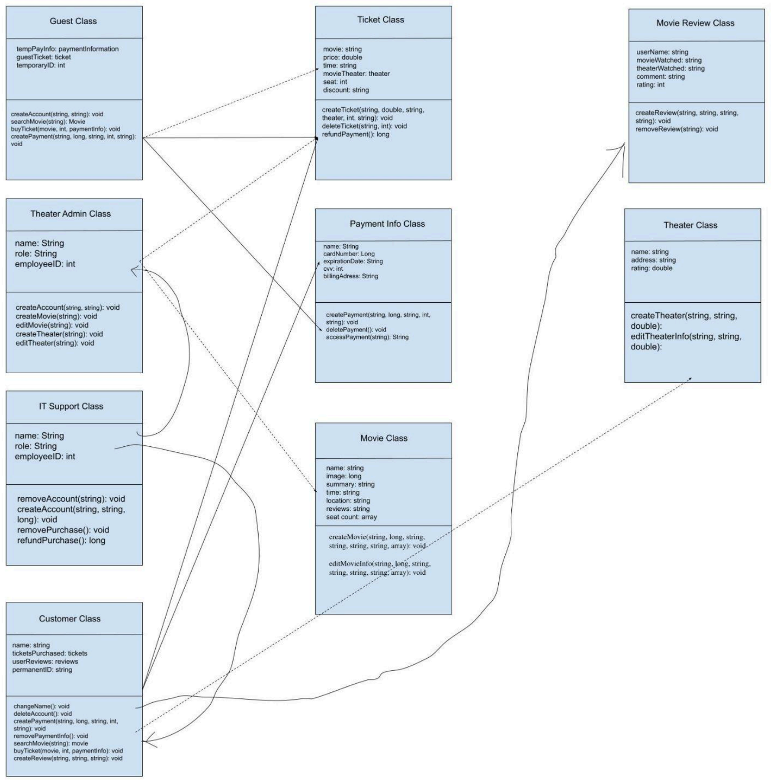
2.1 SWA Diagram


Website Main Page
Login Page/Admin Login
Account Manipulation
Theater Data Manipulation
Movie Data Manipulation
Account Authentication (Username/Password)
Account Info: edit username/password, saved payment options
Location, hours, special events
Time playing, date playing, auditorium number, title, length
Database
Review Page
Review info: Theater, Movie,
Payment Menu
Reserve seat, purchase confirmation
Seat Info: seat number, row number, auditorium number, prices
Available Seats
Movie info, times, theaters
Ticket Confirmation
Seat Map
Movie Browser

2.1.1 Description

The overall design of our software's architecture is heavily dependent on communicating with our database. The software will have 11 components that work together to bring a working theater ticketing system to a customer. The first component is the website's main page. This will mainly be a user interface meant to display to the user what the software actually does. From the main page the user will be able to navigate to the login page or admin login page depending on who is trying to access their login. Once the user enters their username and password the software will bring the associated fields from the database to compare and will authenticate the user if they match. Once successfully logged in, the user will be able to manipulate their existing account information such as their username, password, and saved payment options. This data will be updated in the database by the software. A user may wish to browse movies to watch at a local theater. In this case, the movie browser will take information from the database about available movies to watch, such as a synopsis, showing times, and theater locations. Once an option is clicked on a seat map will be displayed. The seat map will take information from the database about available seats such as seat number, row number, auditorium number, and prices for each seat. That information will be transferred to the payment menu once the user decides to buy a ticket(s). The payment menu will prompt the user for their payment information where the database will send information back to the software to check if the payment has been processed. Then the software will update the available seats in the database. Once the seats have been confirmed to be reserved and paid for, the software will send a confirmation message to the user. Separate from the other pages for purchasing tickets is the review page. Information about past reviews will be brought from the database to the review page. Reviews made by a user will be sent to the database so that it is added to the rest of the reviews. The last two components are the theater and movie manipulation pages for admins. Both will take existing data from the database in case an admin wants to edit existing theater or movie information. New movies or theaters can also be created by admins which will be updated in the database. The information to be transferred between these pages and the database are theater location, theater hours, theater special events, movie times/dates, auditorium number, length of movie, and title.

2.2 UML Class Diagram


Guest Class
tempPayInfo: paymentInformation
guestTicket: ticket
temporaryID: int
createAccount(string, string): void
searchMovie(string): Movie
buyTicket(movie, int, paymentInfo): void
createPayment(string, long, string, int, string): void
Ticket Class
movie: string
price: double
time: string
movieTheater: theater
seat: int
discount: string
createTicket(string, double, string, theater, int, string): void
deleteTicket(string, int): void
refundPayment(): long
Movie Review Class
userName: string
movieWatched: string
theaterWatched: string
comment: string
rating: int
createReview(string, string, string, string): void
removeReview(string): void
Theater Admin Class
name: String
role: String
employeeID: int
createAccount(string, string): void
createMovie(string): void
editMovie(string): void
createTheater(string): void
editTheater(string): void
Payment Info Class
name: String
cardNumber: Long
expirationDate: String
cvv: int
billingAdress: String
createPayment(string, long, string, int, string): void
deletePayment(): void
accessPayment(string): String
Theater Class
name: string
address: string
rating: double
createTheater(string, string, double):
editTheaterInfo(string, string, double):
IT Support Class
name: String
role: String
employeeID: int
removeAccount(string): void
createAccount(string, string, long): void
removePurchase(): void
refundPurchase(): long
Movie Class
name: string
image: long
summary: string
time: string
location: string
reviews: string
seat count: array
createMovie(string, long, string, string, string, string, array): void
editMovieInfo(string, long, string, string, string, string, array): void
Customer Class
name: string
ticketsPurchased: tickets
userReviews: reviews
permanentID: string
changeName(): void
deleteAccount(): void
createPayment(string, long, string, int, string): void
removePaymentInfo(): void
searchMovie(string): movie
buyTicket(movie, int, paymentInfo): void
createReview(string, string, string): void

2.2.1 Description of Classes/Attributes and Functions

2.2.1.1 Guest Class

This class will allow a user to perform operations on this website without an account

2.2.1.1.1 field variables:

- Holds a variable for tempPayInfo which holds a paymentInformation object. This field variable will be holding an object of the paymentInformation class. It will be deleted once the Guest session is ended along with the rest of the class.
- The guestTicket field variable holds a ticket object. This will be created upon purchase.
- The guestClass will have a temporaryID field variable to be used in the ticketBuying operation.

2.2.1.1.2 operations:

- The createAccount operation will take in strings for name and role. This exists to give the guest an option to create a permanent account.
- The searchMovie function will take a string for the movie name and search the database for the specified movie. Then if found it will return a Movie object.
- The buyTicket operation takes in a movie class, an integer for the userID, and a string for the payment information. This will return a ticket object and an email with ticket information when successful.
- The createPayment operation creates a payment information object. It will take in a string for the name on the card, a long for the card number, a string for the expirationDate, an int for the CVV, and a string for the billingAddress. This will create a payment information object.

2.2.1.1.3 dependencies:

- The guest uses the ticketClass because it will create ticket classes associated with the account
- The guest class uses the payment class to create an object to temporarily store payment info.

2.2.1.2 Theater Admin Class

This class will allow movie theater employees to have an account mainly used to perform operations that update movie information and movie theater information

2.2.1.2.1 field variables:

- The name variable is a string holding the name of the Admin user.
- The role variable is a string that will hold the status of the employee.
- The employee ID variable is an int that will hold a unique number to identify the employee.

2.2.1.2.2 operations:

The class will have several operations.

- The createAccount function will create a theater admin object.
- The createMovie function will take in a string and create a movie object in the database. The editMovie will be able to change the movie info.

- createTheater will take in a string and create a theater object.
- editTheater will update theater info.

2.2.1.2.3 dependencies:

- The theater admin class updates the theater and movie classes.

2.2.1.3 IT Support Class

This class allows IT support staff to fix problems such as adding and removing accounts and refunding purchases.

2.2.1.3.1 field variables:

- The name variable is a string that holds the name of the IT support user.
- The role variable is a string that holds the role of the IT user.
- The employee ID variable is an int that will hold a unique number to identify the IT user.

2.2.1.3.2 operations:

- The removeAccount function will take in a string of the account to remove and will delete the object.
- The createAccount function will take in the necessary info and create an account object.
- the removePurchase function will delete a ticket object.
- The refundPurchase function returns a long of the money being refunded.

2.2.1.3.3 dependencies:

- This class uses the Theater admin and customer classes because they have user rights to manipulate and create account information as needed.

This class also uses the ticketClass to refund purchases.

2.2.1.4 Payment Info Class

This class will take in the user payment information when they want to buy movie tickets.

2.2.1.4.1 field variables:

- Hold a string variable for name on the card
- Holds a long for the card number
- Holds a string for the expiration date
- Holds an int for CVV.
- Holds a string for a billing address.

2.2.1.4.2 operations:

- The createPayment operation is a constructor for this class that will take in a string, long, string, int, and a string for the field variables.
- deletePayment is an operation that will remove a paymentInfo object and all of its information.
- accessPayment will return a string with the requested information.

2.2.1.4.3 dependencies:

- This class is not really dependent on other classes. It is a core class that other classes rely on.

2.2.1.5 Customer Class

This class allows a general user a more permanent account to use that saves past movie information and allows the user to leave reviews.

2.2.1.5.1 field variables:

- The name variable is a string that holds the name of the customer.
- The tickets purchased variable is of type tickets.
- The user reviews variable is of type reviews.
- The permanent ID variable is a string that will identify the user.

2.2.1.5.2 operations:

- The changeName function will allow the user to update their username.
- The deleteAccount function will allow the user to delete their account.
- The createPayment function takes in the relevant banking info and creates an object.
- removePayment will delete the object.
- The searchMovie function will take in a string of the movie name and return a movie object.
- The buyTicket function takes in the move, cost, and payment info and creates a ticket object.
- The createReview function will take in strings of the review and create an object.

2.2.1.5.3 dependencies:

- This class is dependent on the Ticket class because it creates objects of the ticket class upon purpose.
- This class is dependent on the payment info class because it saves a field variable of the payment info class
- This class is dependent on the movie review class because this class creates objects of the movie review class.

2.2.1.6 Movie Class

This class is how movies will be stored in the database and will hold information about the movie

2.2.1.6.1 field variables:

- Holds a string for the name of the movie
- Holds a long to store the binary for the image of the movie cover
- Holds a string for the summary of the movie
- Holds a string for the time of the movie
- Holds a string for the location of the movie

- Holds a string for the reviews of the movie
- Holds a 2 dimensional array to represent the taken seats and the vacant seats

2.2.1.6.2 operations:

- The createMovie operation takes in a string, long, string, string, string, array to update the mentioned the variables above when creating an object
- The editMovieInfo takes in a string, long, string, string, string, array to update the mentioned variables above.

2.2.1.6.3 dependencies:

- This class is not dependent on other classes but many classes are dependent on this class.

2.2.1.7 Movie Review Class

This class will be associated with movie objects and be created by customer class objects.

2.2.1.7.1 field variables:

- The userName variable is a string that holds the name of the user leaving the review.
- The movieWatched variable is a string that holds the name of the movie being reviewed.
- The theaterWatched variable is a string that holds the theater the user saw the movie at.
- The comment variable is a string holding the review the user is leaving.
- The rating variable is an int holding the number rating.

2.2.1.7.2 operations:

- The createReview function takes in strings and creates an object of the review.
- The removeReview function deletes the instance of the review.

2.2.1.7.3 dependencies:

- This class isn’t dependent on any other classes.

2.2.1.8 Ticket Class

This class will be created every time a user purchases a ticket and stores relevant information about the purchase.

2.2.1.8.1 field variables:

- The movie variable is a string holding the name of the movie that the ticket is for.
- The price variable is a double that holds the price of the ticket.
- The time variable is a string variable that holds the time of the movies showing.
- The seat variable is an int holding the seat number.
- The discount variable is a string holding the discount code if used.

2.2.1.8.2 operations:

- The createTicket function takes in the relevant ticket info and creates a ticket object with it.
- The deleteTicket function deletes objects created by the createTicket.

- The refundPayment function returns a long of the ticket price being refunded.

2.2.1.8.3 dependencies:

- This class is not dependent on other classes but other classes are dependent on it.

2.2.1.9 Theater Class

The theater class will be manipulated by movie theater employees and will store relevant information about the theater.

2.2.1.9.1 field variables:

- Holds a string to holds the name of the theater
- Holds a string for the address of the theater
- Holds a double for the rating of the movie

2.2.1.9.2 operations:

- The createTheater operation takes in a string, string, and double to populate the variables mentioned above.
- The editTheaterInfo takes in a string, string, and double to update the variables mentioned above.

2.2.1.9.3 dependencies:

- The class is not dependent on other classes but is a core class that other classes.

3. Development Plan and Timeline

There will be three teams of developers on this project.
Group 1 will be led by Manan and will handle the guest, IT support, and customer classes.
Group 2 will be led by Micah and will handle the ticket, payment info, and movie classes.
Group 3 will be led by Dylan and will handle the theater admin, movie review class, and theater class.
Each group will take 2-3 weeks to complete their classes including the implementation of the attributes and functions.
After this period each group will complete testing on their classes and will have to implement new code if needed.
This period of testing will take 4 weeks.
All three groups will reconvene and test the entire system for a period of 2 weeks.