Ticket Hub – Software Design Specification

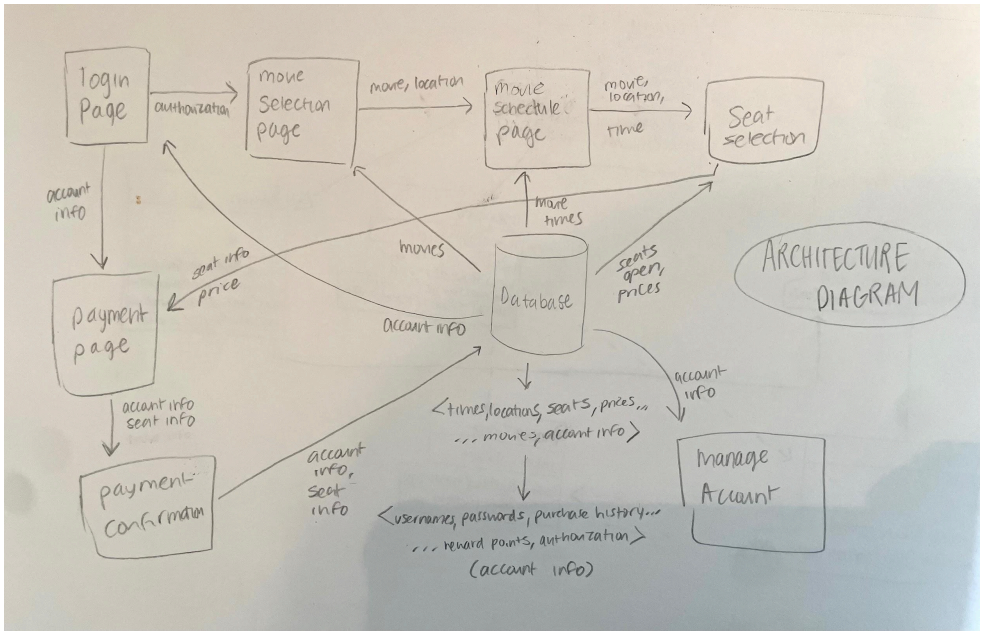
Hannah Ferdows, Amelia Grevin, Elliot Gambale

Brief Description of System:

This Software Design System has been established in order to regulate and establish architectural, visual and working components of our application, TicketHub. Below you will see and read about the data components involved, as well classes, attributes, functions and surrounding specifications needed in order to help us efficiently design and bring to life our software surrounding movie ticket purchasing.

Software Architecture Overview

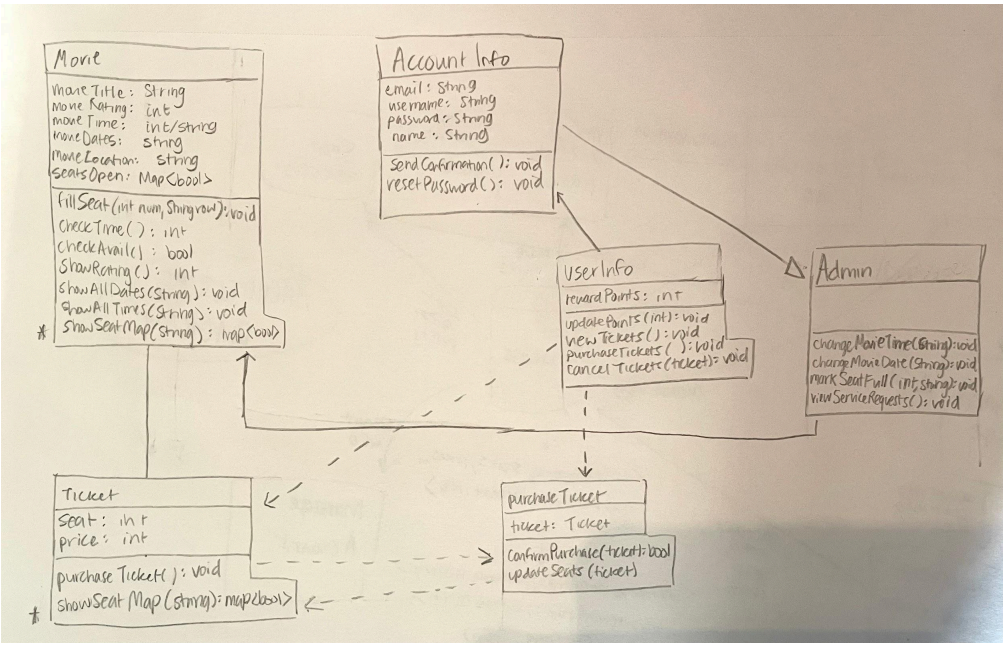
Software Architecture Diagram:



For the architecture of our software, the user will first be met with a login page, where an authorization will be necessary in order for them to move onto the movie selection page. From there, they can choose a movie and a location and therefore choose a time on that specific movie's schedule page and with the information of movie, location and time the seat selection map will be displayed. From there, selected seats will lead the user to the payment page, where the account info from the login page will be sent over in order to execute buying the seat for the price extracted from the seat selection. Finally, the last page will be a payment confirmation where the account info will be used to send that account its confirmation, as well as the

information regarding the seats they purchased. There is also a manage account page, separate to the cycle of the rest of them the account info will be used for the user to do certain functions specific to the details of their account (more on that below within the classes). The database will be the central means of which the different components will receive certain information, such as times, locations, seats, prices, movies and account information. Account information will more specifically have the usernames, passwords, purchase history, rewards points and account authorization.

UML Class Diagram:



Classes:

The classes consist of *Movie, Ticket, PurchaseTicket, UserInfo, Admin*, and *AccountInfo*

Attributes:

Movie – movie consists of movieTitle, movieRating, movieTime, movieDates, movieLocation, and seatsOpen. The title, date and location and time will all be held and stored as Strings within the class, while the rating will be an integer, and finally the seats open will be stored within a map where boolean values will determine each individual seats availability.

Ticket – ticket consists of an assigned seat value, and price for the ticket, which will be two attributes stored as integers within the class.

PurchaseTicket – the class purchaseTicket has the attribute of ticket, which is an object (ticket) of the Ticket class.

UserInfo – userInfo will have rewardPoints, where the number of points gained on their account will be stored, each being unique to the individual account, stored as an integer.

Admin – since the Admin class inherits the attributes of the class AccountInfo, it does not have any other components outside of those it inherits. (listed below)

AccountInfo – this class consists of the attributes of the user's email, the accounts username and password – as well as the name of the accounts owner – which will all be stored as Strings.

Operations:

Movie – within the movie class, you will find the operations fillSeat, checkTime, checkAvail, showRating, showAllDates, showAllTimes, showSeatMap. With fillSeat, between the ticket class, and purchaseTicket, when a customer is to purchase a ticket the information will update seatAvailability() (function of purchaseTicket), which will update the ticket class' showSeatMap, finally sent over to the movie class where it will specifically then mark the seat for that movie at the customer's specific time and location - receiving the values of seat number (int) and row (String) in order to do so. Next, there is checkTime() and checkAvail() which when called will print all available times for the chosen movie, and whether the showtime is still available for more viewers. For showRating allDates and allTimes, the method will print the extracted rating of the movie, the available dates for the movie to be shown, and the times at which it will be shown. Finally, showSeatMap() will allow the customer to view which seats are available visually rather than by numbers or a listing, in order to see if the movie that they have chosen has their ideal seating arrangement for purchase. To see dates, times and the seat map however it must be mentioned that the method will need to receive the movie title to provide the appropriate information.

Ticket – for the ticket class, there will be two operations, being purchaseTicket(), and the inherited method of showSeatMap() which will be used to validate whether purchasing a ticket for the client's specified seat is allowed. purchaseTicket will first call showSeatMap() where the parameter of which movie to show the seat map for is accepted. Once viewing the seat map, the user will be able to choose a seat to then have purchaseTicket() called on.

PurchaseTicket – confirmPurchase() will receive the parameter of the object ticket, which will hold the seat number and price (inherited from Ticket) and will return a confirmation if successfully executed (bool), and if else (false) will let the customer know the purchase did not go through. For updateSeats(), the same ticket will then be passed to this method if the purchase confirmation is successful, in order to update the seat map which is essential to the functionality of both Movie and Ticket.

AccountInfo – account info will need to send a confirmation that the account has been logged into, leading to sendConfirmation() operation, as well as the potential need for a password reset, which is why resetPassword() function will clear the password and allow the user to enter and save a new one to their specific account.

UserInfo – for userInfo, since it already holds all the inherited information from account info regarding the general necessities of an account, this class instead has the specific details of the customer, leading for the need of the functions updatePoints(), viewTickets(), cancelTicket(), and purchaseTickets(). When a ticket purchase goes through, the user's points will be updated, leading to update points adding the value to the existing amount. View tickets will allow the user to view any tickets they've already purchased, and cancel tickets will cancel any, given that it is given the parameter of which ticket to cancel. Finally, the user will have the option available to purchase more tickets if needed, where the function purchaseTickets() will send them to purchaseTicket in order to confirm the sale.

Admin – the Admin will need the ability to override certain things, update information – yet will also have an account, leading it to inherit the attributes of an account from AccountInfo. The admin will be able to change the available movie times with changeMovieTime(), with the parameter of the movie to change being received (as a String) and it will also be able to do the same with the movie date. Next, the admin will be able to manually mark a seat as full in case a ticket is purchased at the theater in person, or any other need to do so. This leads the method markSeatFull() to need to receive the seat number (int) and the row (String). Lastly, the administrator will be able to view any submitted service requests about the website/application using the parameterless method viewServicerequests().

Development plan and timeline

- Partitioning of tasks:

  To assign tasks to the different teams working on this project we must first divide the work and assign tasks to the proper teams. We want to ensure that the work is divided evenly enough that no single team member is overloaded with responsibilities. Find the list of responsibilities for each team and their member counts in order for an efficient product production

- Team member responsibilities:

  -Web developers(30 members)-The web developers are the engineers responsible for the coding and main construction of the website. After receiving instructions from the visual design teams and the analysts, they do the primary block of work. This team is the largest because the work is the most time consuming, the developers will be working on a software level and will implement code to ensure the quality of the new site. They are also in charge of debugging and fixing any errors during the testing process. This team

can reference the srs, specifically section 3 as a guide for the construction of the site. Following the srs should ensure all requirements/specifications are met and properly executed throughout the software development side of the project.

-Security(5 members)- The security team should also be well versed in coding languages as well as software security. The security team is responsible for the part of the site that deals with payment methods(such as a direct payment to certain banking systems). They should be able to ensure the safety of a user's card information, as well as the safety of a user's device. This team should be in charge of the firewalls that prevent viruses from taking over the site on any administration or users device. The security team's main purpose is to ensure reliability of the site.

-Visuals Team(4 members)-The visuals team are just what it sounds like, they are in charge of designing the website's graphics. They create templates/layouts of the site that should be displayed on the users devices, they should take into account the sizing modifications for different devices. The work done here should be handed off to the web developers to execute.

-Analyst(1 member)-It is the analyst's job to ensure that both client and company needs are met. The analyst should work closely on the front end of the project, ensuring that the guidelines for the site will satisfy the customer and follow the company policies.

3.7 Design Constraints

Ticket hub will run into a few constraints imposed by other standards, these standards include; company policies and hardware limitations. Below are constraints listed for each of these standards.

Company policies:

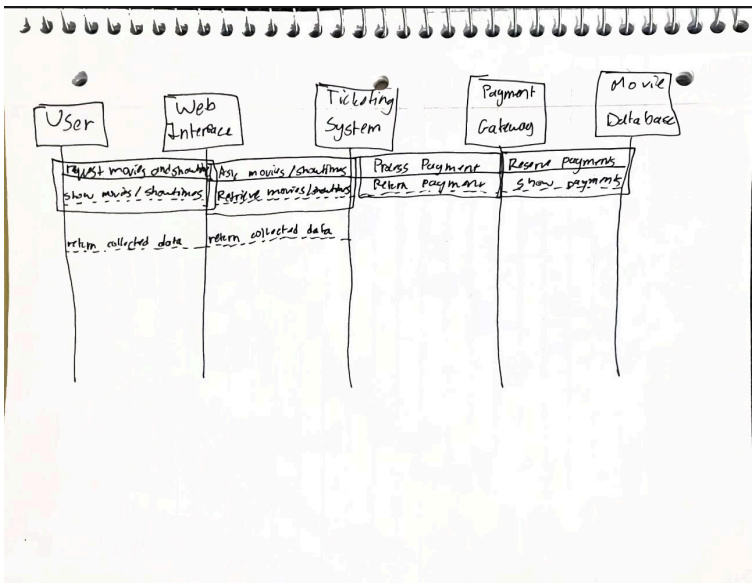
- Privacy Policy: this policy ensures that the privacy of the user remains intact, that no credit card information is at risk of obstruction and personal information should remain private.
- Terms of use: When users access the site, it should be noted that their information should be truthful and accurate. The password information should not be shared and is in the hands of the user, it is not the responsibility of the admin. Ensure good user conduct, there should be no abuse of the site. In any of these cases there should be a user ban for a predetermined time.
- Ticket policy: return policy for tickets should be declared and rules should be strictly followed by users and enforced by administration. Simple ticket rules should ensure there is only one ticket given out for each seat and a maximum number of tickets should be given

Hardware Limitations:

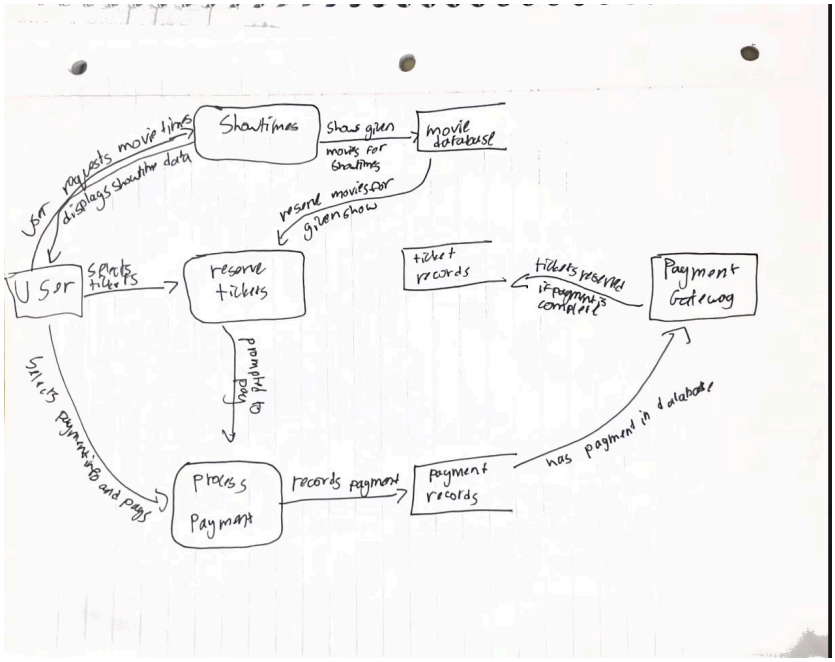
Hardware limitations often depend on the user and admin devices. The site should not overwhelm a user's device/cpu and should be scaled to adjust to any different model/brand/company's device. Software should be formatted to fit any screen size/ be available in different formats for phone vs computers. Hardware constraints during production period/admin accessing, may include; memory storage space, internet connection, battery life, and heat management. One major hardware restriction for both admin and users is wifi/connectivity. The user should be connected to the internet to access the site.

4. Analysis Models

4.1 Sequence Diagrams



4.3 Data Flow Diagrams (DFD)



4.2 State-Transition Diagrams (STD)