Software Design Specification

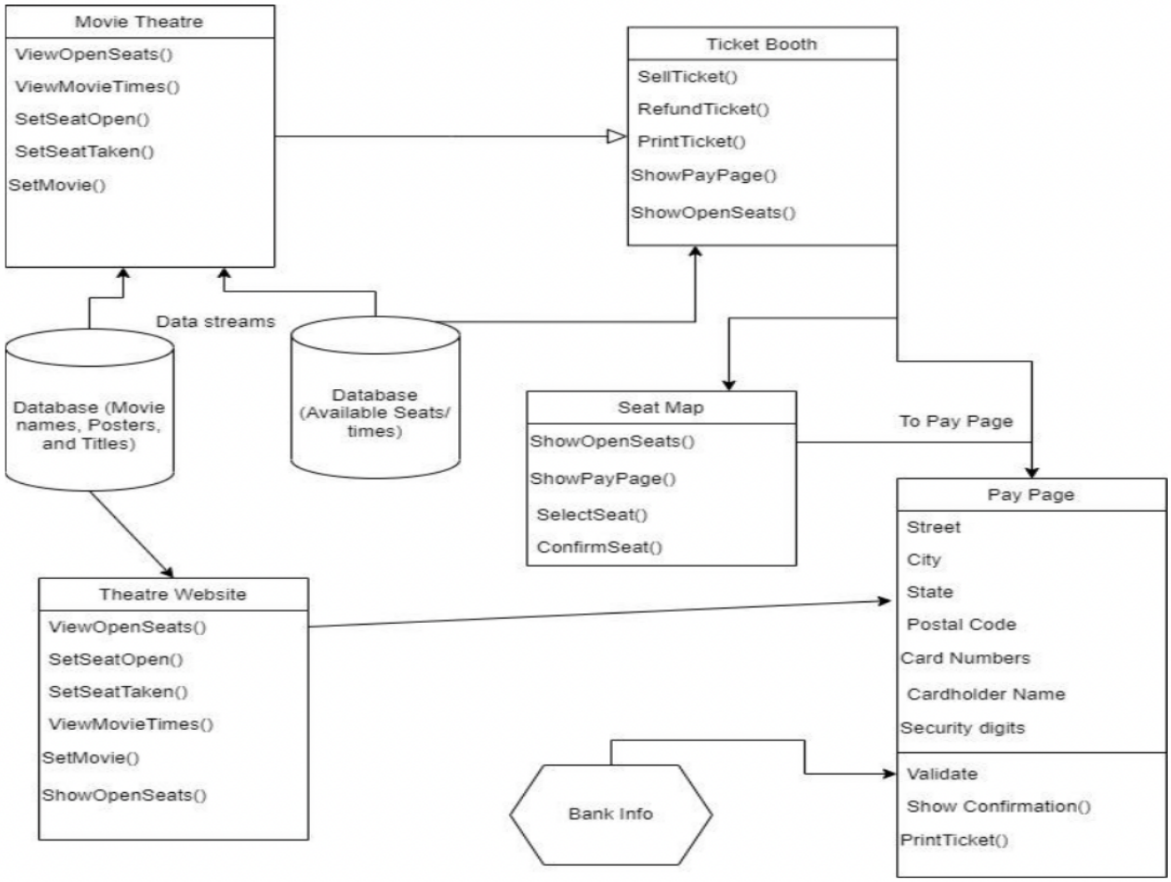
Movie Theater Ticketing System

Wyatt Mason
Mohamed Sulevani
Paul Nusser
Bryce Zimmerman

Software Description:

The software designed will present an interactive program where users and team members can access the ticketing system for the movie theater. It will contain various requirements as specified below. The program will allow customers to book tickets to movies and will allow team members to access the database. It gives many options such as the ability to choose the seat and a full payment portal.

UML Diagram



UML Diagram Function

Private Functions:

● SetSeatOpen(seat number): sets a seat as open for customers to select ● SetSeatTaken(seat number): Sets a seat as taken by a customer

● SetMovie(name, time): Adds a movie and time to the database if the time slot is available ● SellTicket(): Sets ticket status to sold to keep track of sales ● PrintTicket(): Send you to a print page for the ticket

● RefundTicket(sale number): Starts the refund process for incorrectly sold ticket ● ShowPayPage(): Sends the user to the payment confirmation page

● ShowOpenSeats(): Shows the open seats on the screen

● ShowConfirmation(): Shows a confirmation prompt on the screen

Public Functions(for the user)"

● ViewOpenSeats(): Shows the user the available and taken seats

● ViewMovieTimes(): Shows the available movies and times to the user ● SelectSeat(): Enables the user to select a seat for their movie

● ConfirmSeat(): The user confirms the seat they selected is the right one

Private Data:

● Street

● City

● State

● Postal Code

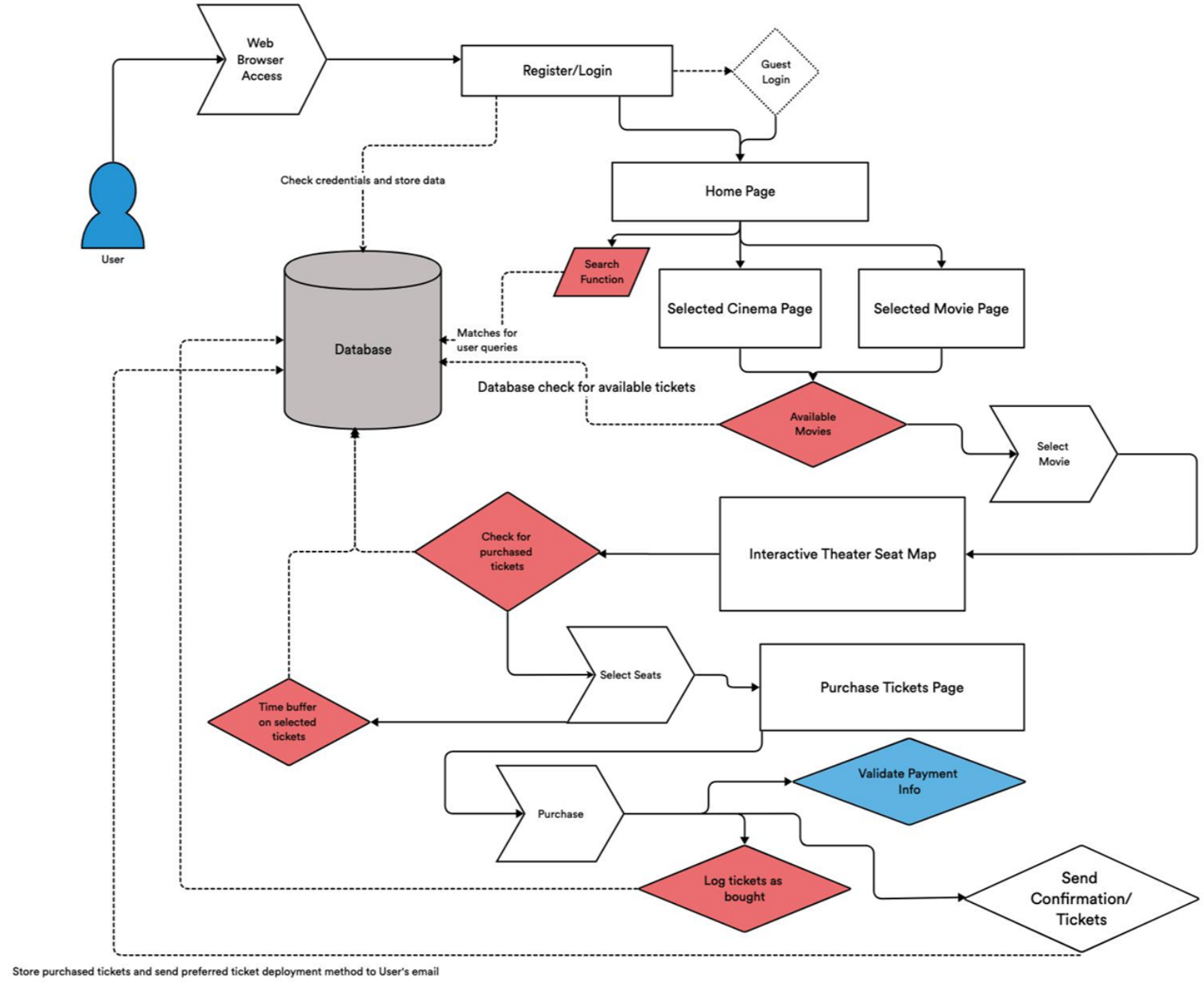
● Card numbers

● Cardholder Name

● Security digits

● Bank Info: Contains information used to verify card and cardholder information

Software Architecture Diagram



Software Design Overview:

Account and Access:

The software architecture shows the design of the system and how it is used in practice. The software system is built to be used in any web browser. Provided the user has access to a web browser, then the user can use our system. Upon accessing our webpage, the user will be prompted for their login information or an opportunity to register with our site. While this is encouraged, the user will always have the ability to login and operate all core features as a guest. User accounts will allow our system to store critical information about the user and to streamline their experience.

Database:

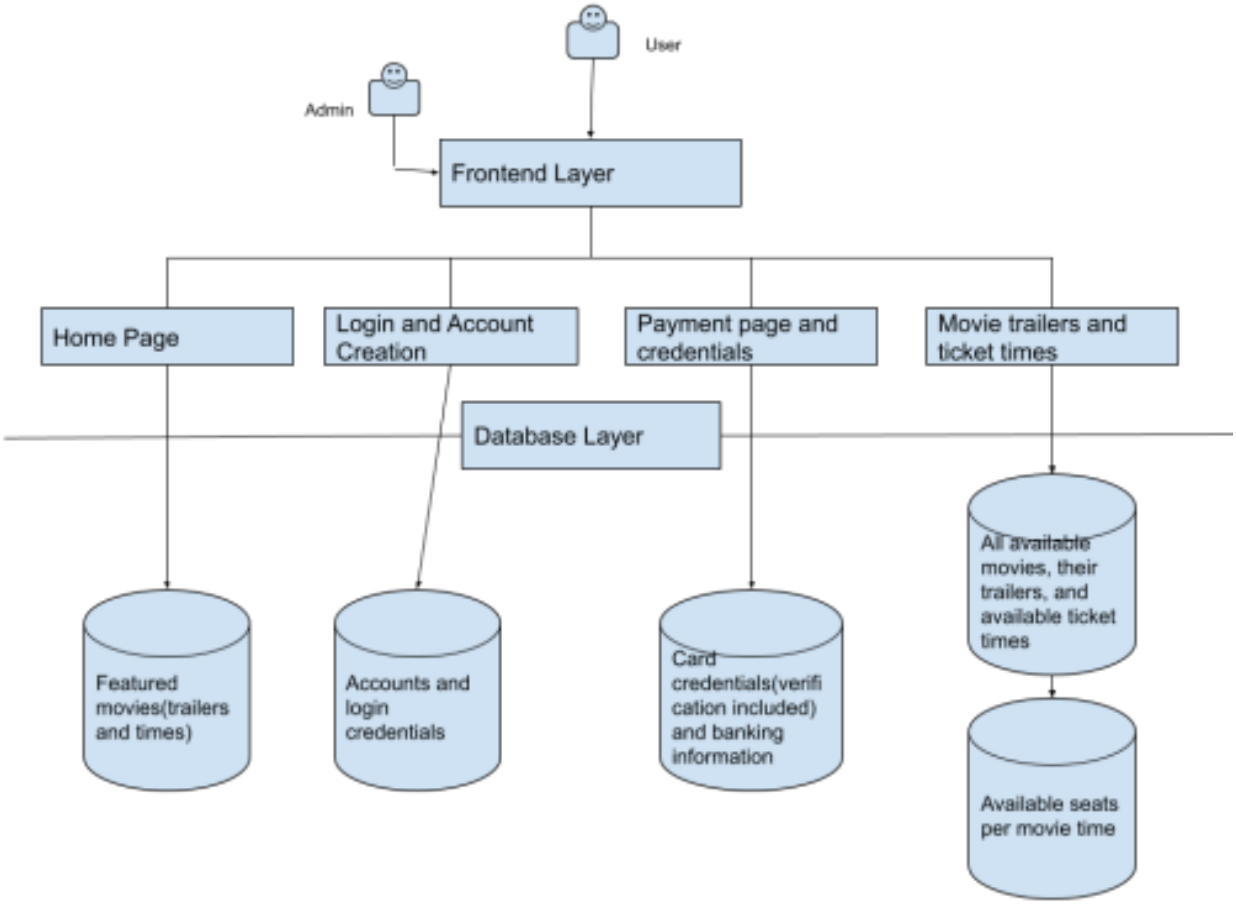
The database will be storing movie/cinema data, dates, ticket status, and user data. Each movie stored will have its images, details, reviews, showtimes, and tickets stored. These can be readily accessed by any page hosting that data. Each user account will have its own ID, email, phone number, preferences, home theater, and purchase history stored. Ticket status is stored for each showtime and must be able to be modified indirectly through user inputs. Tickets must be purchased, available, or temporarily put on hold for functionality reasons.

Webpage and Usage:

After logging in or continuing as a guest, the user will be met with our home page. This will have featured movies, nearby cinemas, and a search function posted on the top of the screen. The user can navigate through the selected movies and cinemas or use the search function to refine what they are looking for. Users can search for specific movie titles, cinemas, and filter by date. A user can proceed towards the page of a specific movie title, seeing showtimes and cinemas. Or they can proceed to a cinema's page, seeing movies and their showtimes. Once a user has selected a movie/showtime, the database will be checked for available tickets.

Ticket selection is done through the interactive seat map. A map of the theater is shown with each seat showing its availability. A user is then able to select desired seats. Selected seats, purchased or not yet will be deemed unavailable for a small time buffer. This will inhibit problems of users purchasing tickets while another user is attempting to purchase the tickets. If the purchase tickets page is left or the user times out, those tickets will be set to available again. Once the user wishes to purchase, they will be taken to the purchase page where they will be prompted for their payment info. Once that payment info is validated then those tickets will be changed to purchased and the user can receive their tickets. Tickets will be deployed to the user's email or if a guest, they will be prompted with how they would like to receive their tickets.

Architecture Diagram:



Data Management Strategy:

Diagram Description:

Our Diagram depicts the distribution of the website's data in our system. It shows that the website can be accessed by either a user or an admin on the frontend. It then shows what data is stored under each page. The home page stores the featured movies with trailers and times. The login and account creation page has a database storing the credentials of the accounts. The payment page stored payment credentials and banking information. Finally the movie trailers and ticket time has a database storing the information on movie seats, times, availability, and trailers. Through this data management system we store all of our data and can access it all when needed.

Tradeoff and Justification Discussion:

SQL: We will use SQL to manage our online movie theater ticketing system. This decision is based on several factors. One of the benefits of SQL databases is that they provide advanced query capabilities, enabling us to quickly and easily search and analyze large datasets. This means that we can ask complex questions of our data and get meaningful answers quickly. Additionally, SQL is a mature and well-understood technology, making it easier to find skilled developers to maintain and improve our system over time. SQL databases offer strong ACID properties (Atomicity, Consistency, Isolation, and Durability), which guarantee data consistency, reliability, and durability. It will also easily handle multiple databases on the same system.

Our SQL database can be organized into multiple tables, each storing different types of information. For example, we can have a separate table for our users. Their credentials, personal information, and order history. This will enable us to store and retrieve user information easily and securely. We could also include tables for genre classifications, as well as for reviews and ratings from other users. Being able to modularize data in such a way allows for better performance and maintenance.

The use of SQL also has some potential tradeoffs. For example, SQL databases can be less flexible than other databases, which can be a disadvantage if we need to make frequent changes to our data model. Additionally, SQL databases can be slower than other databases when handling large amounts of unstructured data. However, we feel the benefits of SQL outweigh these potential tradeoffs, as our online movie theater ticketing system relies heavily on consistency, reliability, and security.

Design: We will be using multiple databases on our system to house different types of data. We have divided these into four groups. Movie data, user accounts, payment, and seating. These 4 databases will store their respective data. We decided to use these databases in order to provide better security and reliability. The movie data database will have all necessary information about specific movies, their showtimes, and theater data. The user account database is going to store all login credentials and user account details. The payment database is going to store banking information, card credentials, and payment processing/verification. Finally, our seating database will store seat availability status and the seat selection map data.

We feel that separating into four databases will give us enhanced security, improved performance, and easier scalability. While we could have more or less than 4 databases, we feel this will be a good amount. Any more and we feel it may become too diluted. Any less and we feel there is too much crossover.

Other tech: We plan to use a modern web development framework such as Node.js, which will handle the web server, and HTML, CSS, and JavaScript for the front-end. To handle user authentication and authorization, we will use a secure and widely-used technology like OAuth2. To handle payment processing, we will integrate with a payment gateway such as PayPal, which can handle credit card processing and ensure compliance with relevant regulations.