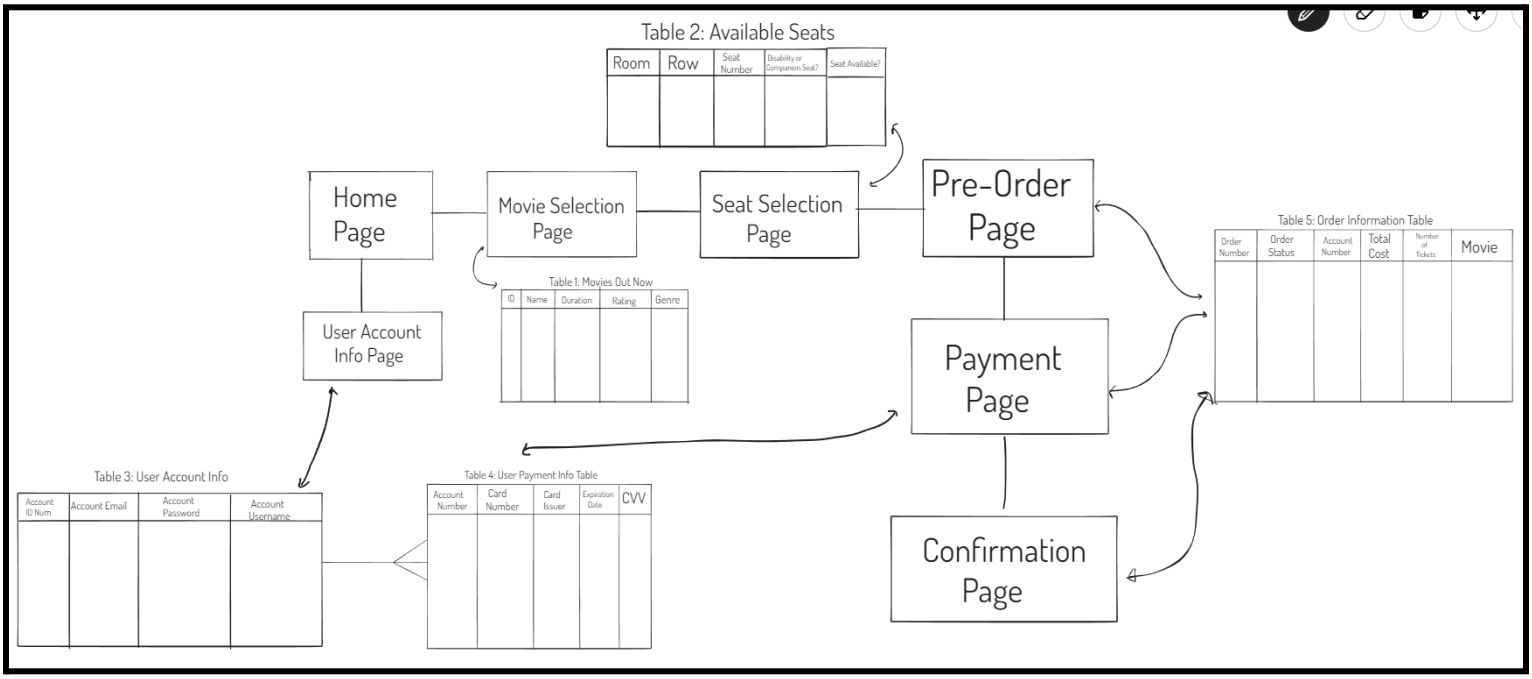
Felix Marecaux

Erasto Astudillo

Carlos Anguiano

Aidan Shotland

Software Architecture Design w/Data Management



1. Software Architecture Diagram:

Description:

- Our updated system architecture diagram includes some illustrations of the 5 database tables that will be used to manage user data and order data. Our system begins at the home page. The user may elect to navigate to their account information page, which draws its information from a database table that contains their information. They may also navigate to the movie selection page, which is connected to a database table containing various information on movies that the user may buy tickets for.
- Here, they can select a movie to purchase tickets for, and once they do they are taken to select their seats on the seat selection page. Information on available and reserved seats is drawn from another database table.
- Once they have selected their seats, they can navigate to the pre-order page to see an overview of their order. Once confirmed, they are taken to the newly added payment page. The user's

payment information is drawn from a separate database table, where if the user has previously ordered and opted to save their payment information, the information is autofilled for them. If they have not made any previous orders, they are prompted to enter their payment information. Lastly, once purchasing their order, they are sent to a confirmation page which displays their order number for verification. All of the pre-order, payment, and confirmation pages draw from another database table. They may navigate back to the home page from here if they would like.

Our software system contains various data tables.

1. First, there is a data table for what movies are currently showing. This table contains five fields, which are ID, Name, Duration, Rating, and Genre.
    - *ID*: a unique integer number that specifies one movie from another
    - *Name*: a string that will hold the name of the movie
    - *Duration*: an integer number that specifies how long the movie is, in minutes.
    - *Rating*: a type double number that provides a rating for the movie
    - *Genre*: a string that provides context for what category the movie falls under.

Table 1: Movies Out Now

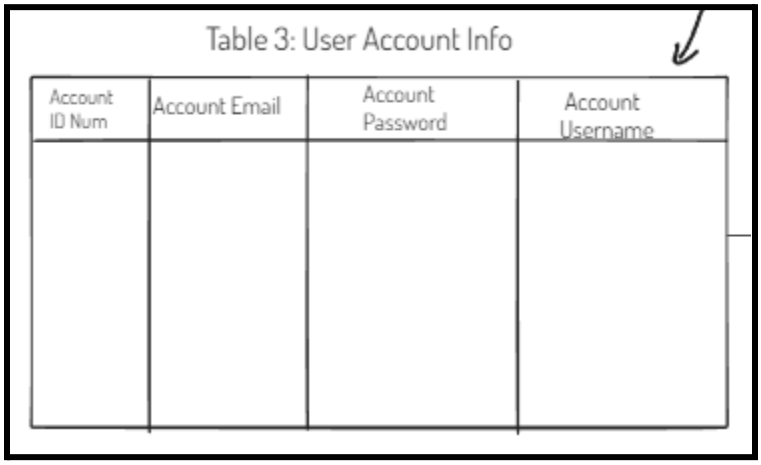
ID	Name	Duration	Rating	Genre

2. Second there is also a data table for available seats.This table includes five fields, which are the room number, row, seat num, availability, and disability.
    - *Room Num*: specifies which room in what theater the seats are
    - *Row Num*: specifies what row the seat is in
    - *Seat Num*: specifies the specific seat number
    - *Availability*: boolean value that specifies whether seat is available
    - *Disability*: indicates whether seat is reserved for people w/ disabilities

Table 2: Available Seats

Room	Row	Seat Number	Disability or Companion Seat?	Seat Available?

3. Third, there is an account info data table. This has four fields, which are account ID number, account email address, account password, account username, payment info.
    - *ID*: this will be a integer value that is unique to each user, used for internal processes
    - *Email*: string that holds email address of user, used for payments, notifications, and confirmations
    - *Password*: string that will hold the user's login password
    - *Username*: user's display name on the software system



4. Fourth, there is the payment information data table. This table is linked to the account info datatable, and has 5 fields. The fields being, account number, card number, issuer, expiration date, and security code.
    - *Account Num*: account ID for the card that is going to be used
    - *Card Num*: 16 digit value that holds the card number
    - *Issuer*: String value holding the card company
    - *Expiration*: value that holds the expiration date of the card
    - *Security*: integer value that holds the three digit security number

Table 4: User Payment Info Table

Account Number	Card Number	Card Issuer	Expiration Date	CVV

5. Lastly, there will be a data table that correlates to the pre-order page, payment page, and confirmation page. This table has 6 fields, being order number, status, account number, price, number of tickets, and movie.
    - *Order Num*: This will be a unique integer value that is specific to the current order that is the user is about to complete
    - *Status*: This will be a string value which will show whether the movie is ongoing, hasn't started, or is finished
    - *Account Num*: Specific ID to the user that is completing the order
    - *Price*: double type value that specifies the price for the number of tickets ordered
    - *Num Tickets*: integer value that holds the number of tickets being purchased
    - *Movie*: string value that holds the name of the movie being chosen

Table 5: Order Information Table

Order Number	Order Status	Account Number	Total Cost	Number of Tickets	Movie

2. Data Management Strategy:

For our software system, we decided to use an SQL approach to our data management strategy. There are a couple of reasons for this. Some of the reasons are:

1. Non-SQL can not guarantee ACID properties. This is an issue when our software system needs to be able to make financial transactions. ACID stands for atomicity, consistency, isolation, and durability.
    - *Atomicity*: the entire transaction must go through at once, or not at all. When a user pays for their selected movie/seats, the transaction must occur entirely, when prompted.
    - *Consistency*: the database must remain the same before and after, and it can not change or transform. The movie selection will not be constantly changing, and neither will the prices.
    - *Isolation*: Multiple transactions will be occurring simultaneously, as there will be many users using the software at the same time. This means that all the transactions must be doable at the same time, and remain consistent from user to user.
    - *Durability*: Even if an error occurs within the system, the transaction will still go through and be successful.
2. Since consistency is a priority, and the database will not be getting any large-scale changes in terms of data volume, SQL is better. The number of movie showings will not increase or decrease, since there is a set number of screens and theaters. Thus, data volume will not receive any large-scale changes.
3. Using the SQL approach, we can take advantage of SQL database normalization. This means it will be easier to avoid redundancy and inconsistency, while promoting efficient storage, updating, and maintenance.
4. SQL has a wide range of different commands that are available for use, and also includes a lot of documentation for developers to use. Additionally, operations such as insertion, deletion, querying, manipulation, and calculations within the database are a lot quicker in SQL thus making a more efficient experience for us and the users.