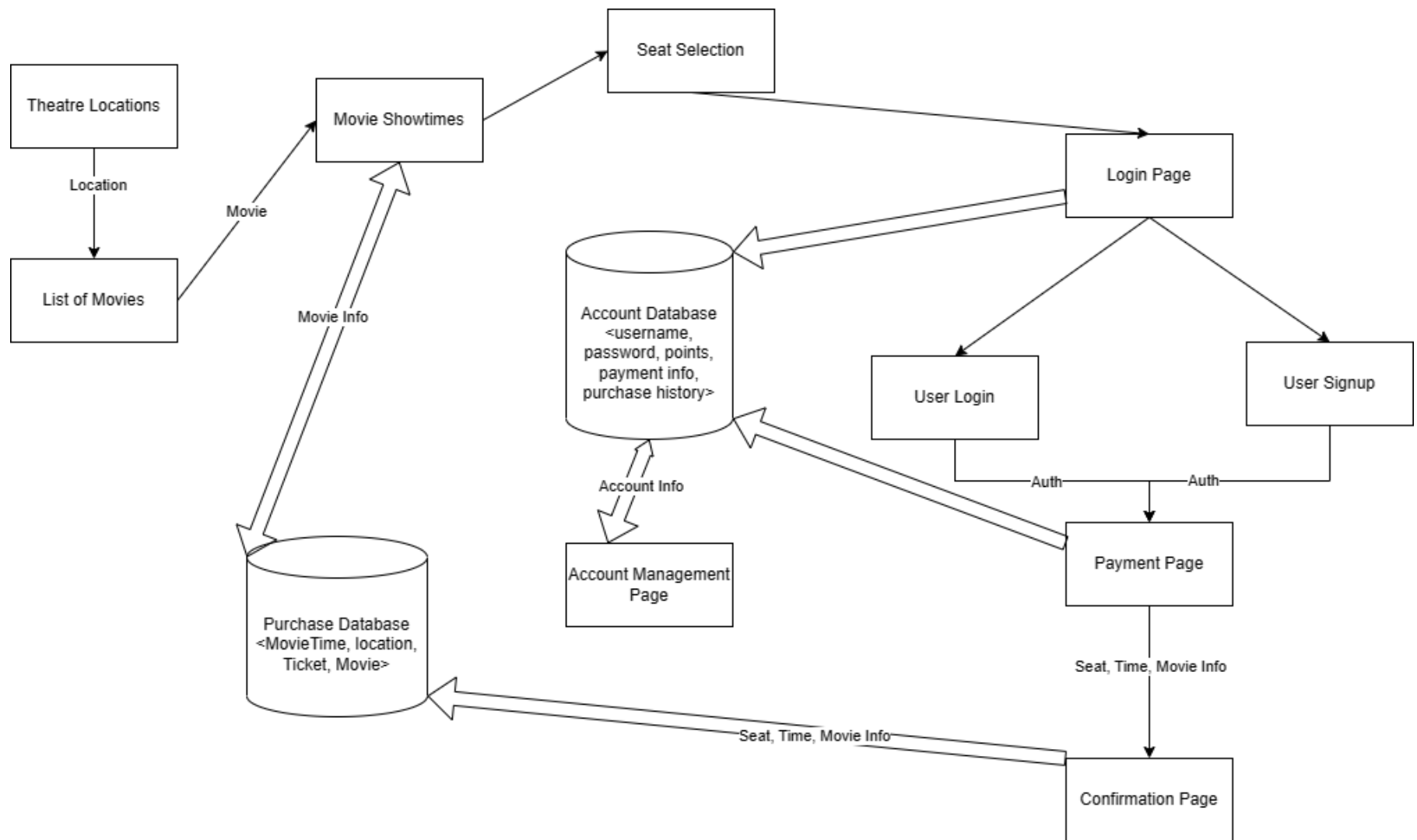


# Movie Theater Architecture Design w/ Data Management

Prepared By: Ahmed Hageb, Aaron Maryfield, Michael Zalameda,  
Trenton Taylor, and James Yang

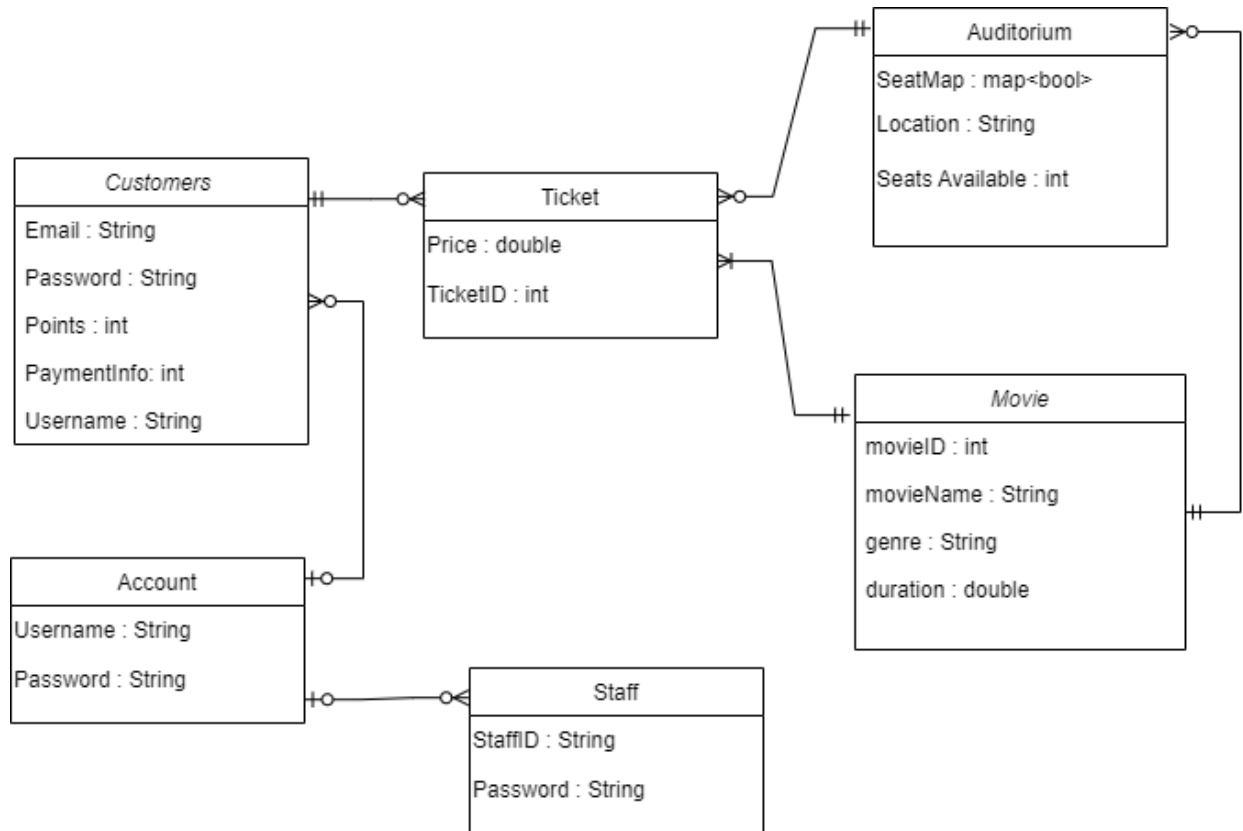
April 16, 2023



### Adjustments:

- Separated the databases into Account and Purchase Databases for better data management

## Data Management Strategy :



This ERDiagram features the 2 databases:

### Account Database:

- Every account can either be of type customer account or a staff account. You can have many 0 or many customer/staff accounts in your database.

### Purchase Database:

- Every customer account can purchase 0 or many tickets. Each ticket is specified to one and only one customer.
- Every ticket can only be specified to one movie. You can have one or many tickets. A movie can be shown in one or more auditoriums.
- Every ticket is specified to one and only one auditorium. You can have 0 or many tickets for one auditorium. Each auditorium is specified to one and only one movie at a time.

**Tradeoff Discussion :** You should also discuss your design decisions: how many databases you chose and why, how you split up the data logically, possible alternatives you could have used (both in technology and organization of data), and what the tradeoffs are between your choice and alternatives.

When deciding on the database management system for our movie theater architecture design, we considered both SQL and NoSQL options. If we were to use SQL databases our decided options were MySQL, Oracle, or MicrosoftSQL. And for NoSQL, our options were MongoDB, BigTable, or Redis. After evaluating the requirements and constraints of the project, we ultimately chose MySQL because of SQL's ability to handle complex relationships between data and its strong consistency guarantees. One of the primary advantages of SQL is its ability to handle complex relationships between data, which is essential for our project. For example, each ticket must be linked to a specific customer account, movie, and auditorium. SQL databases allow us to define these relationships using foreign keys and other constraints, ensuring data integrity and making it easier to query and analyze the data. Another advantage of SQL is its strong consistency guarantees. This means that any changes made to the database are immediately reflected in all queries, ensuring that everyone accessing the database is seeing the most up-to-date information. In a high-volume environment like a movie theater, this is critical for ensuring that customers are able to purchase tickets for the movies they want to see. While NoSQL databases can also handle complex relationships between data, they do not provide the same level of consistency guarantees as SQL. This can lead to issues with data integrity and reliability in high-volume environments, which is why we ultimately chose SQL for our project. However, there are tradeoffs to using SQL. For example, it can be more difficult to scale SQL databases horizontally as the amount of data grows. Additionally, SQL databases can be more complex to set up and maintain than non-SQL databases. However, for our specific use case, the benefits of SQL outweigh these potential tradeoffs.