Tradeoff Discussion

1. SQL or noSQL database?
   Justin Opinion: SQL since it is just a simple movie theater ticketing system
   Rose : Yep SQL
   Ruben: SQL since it is more appropriate for applications that conduct financial transactions

2. How many databases did you choose? And why?
   Justin Opinion: 1 SQL database
   Rose: 1 SQL database per the requirements
   Ruben: 1 SQL database per project requirements, also if needed to scale vertically by upgrading CPU, SSD, and/or RAM.

3. How will you split up the data logically?
   Data would be split into a tables containing columns and rows for movies, screening times, length of movies, theater locations and to manage seats that are taken for premium locations.

4. Possible alternatives you could have used (both in technology and organization of data).
   We could've used NoSQL databases.

5. What the tradeoffs are between your choice and alternatives?

   Since we will be using multiple tables instead of lists, relational SQL will be utilized. NoSQL would have been less expensive to manage, but would not be ideal in terms of consistency.

(Rose)Database ERD:

Entities:

User
Movie
Theater
Showtime
Booking
Ticket
Users can create bookings to purchase tickets for movies.
Movies have information like title, genre, and release date.
Theaters have details about their name, location, and seating capacity.
Relationships: Showtimes include information about the movie, the theater it's shown in, and the start and end times.

Bookings are associated with users and contain information about when they were created.
Tickets are part of bookings and are linked to specific showtimes, including seat numbers and prices.

(Rose) ENTERPRISE LEVEL RELATIONSHIP DATABASE: Supports many users at the SAME time.

- Provides data to dashboards used by managers who manage theaters.
- Provides data to users utilizing the website.
- Has multiple tables.

USER >> DATABASE APPLICATION >> DBMS >> DATABASE
USERS: CUSTOMERS
ADMINS
THEATERS
DATABASE APPLICATION: Web Browser HTML
DBMS: MySQL
DATABASE: Multiple Tables

- Movies
- Reservations
- Theaters
- Etc.

SECURITY of DB:

- Encryption
- Passwords
- Views

Periodic Backups

SQL vs NoSQL tradeoffs

SQL	**NoSQL**
SQL database not well-suited for hierarchical data storage.	NoSQL databases best suited for hierarchical data storage as it follows the key-value pair method for storing the data
From a commercial perspective, SQL databases are generally classified as open source or closed source.	They are classified on the basis of the way they store data as key-value store, document store, graph store, column store, and XML store.
SQL databases properly follow ACID properties (Atomicity, Consistency, Isolation & Durability)	NoSQL databases properly follow Brewers CAP theorem (Consistency, Availability, and Partition tolerance)
Adding new data in SQL database requires some changes to be made like backfilling data, altering schemas.	New data can be easily inserted in NoSQL databases as it does not require any prior steps.
Excellent vendor support and community Support is available for all SQL databases.	Only limited community support is available for NoSQL databases.
Best fit for a high transaction-based application.	You can use NoSQL for heavy transactional purposes. However, it is not the best fit for this.
Not suitable for hierarchical data storage.	Suitable for hierarchical data storage and storing large data sets (E.g. Big Data)