Data Management Strategy

For our Movie Ticketing System, we decided to use one main SQL database (PostgreSQL) to store all our data. SQL makes sense for our project because it’s good at keeping data consistent, which is really important for things like making sure no two people can book the same seat at the same time.

How We Organized the Data
We split our data into several tables so everything stays neat and easy to work with:

- Users — account info like name, email, hashed password, and payment method.
- Movies — title, genre, duration, rating.
- Showtimes — which movie is playing, when, and where.
- Tickets — which user booked which seat for which showtime.
- Payments — payment amount, status, and timestamp.

The tables are linked together:

- Each ticket points to a specific showtime.
- Each showtime points to a specific movie.
- Each payment points to a specific ticket.

Keeping Data Safe and Backed Up

- Backups — the database saves a backup every day so we can recover data if something goes wrong.
- Replication — we keep a copy of the database to handle more users and in case the main one fails.
- Security — passwords are hashed using bcrypt, and payment info is encrypted.
- Access Control — only the parts of the app that need the data can access it.

Why We Picked SQL
We looked at NoSQL (like MongoDB), but SQL was better for us because:

- We need strong relationships between our data (users, tickets, movies, etc.).
- We want transactions so booking is always accurate.
- It’s easy to back up and secure.

NoSQL could be more flexible and easier to scale for certain projects, but our system benefits more from SQL’s structure and rules.