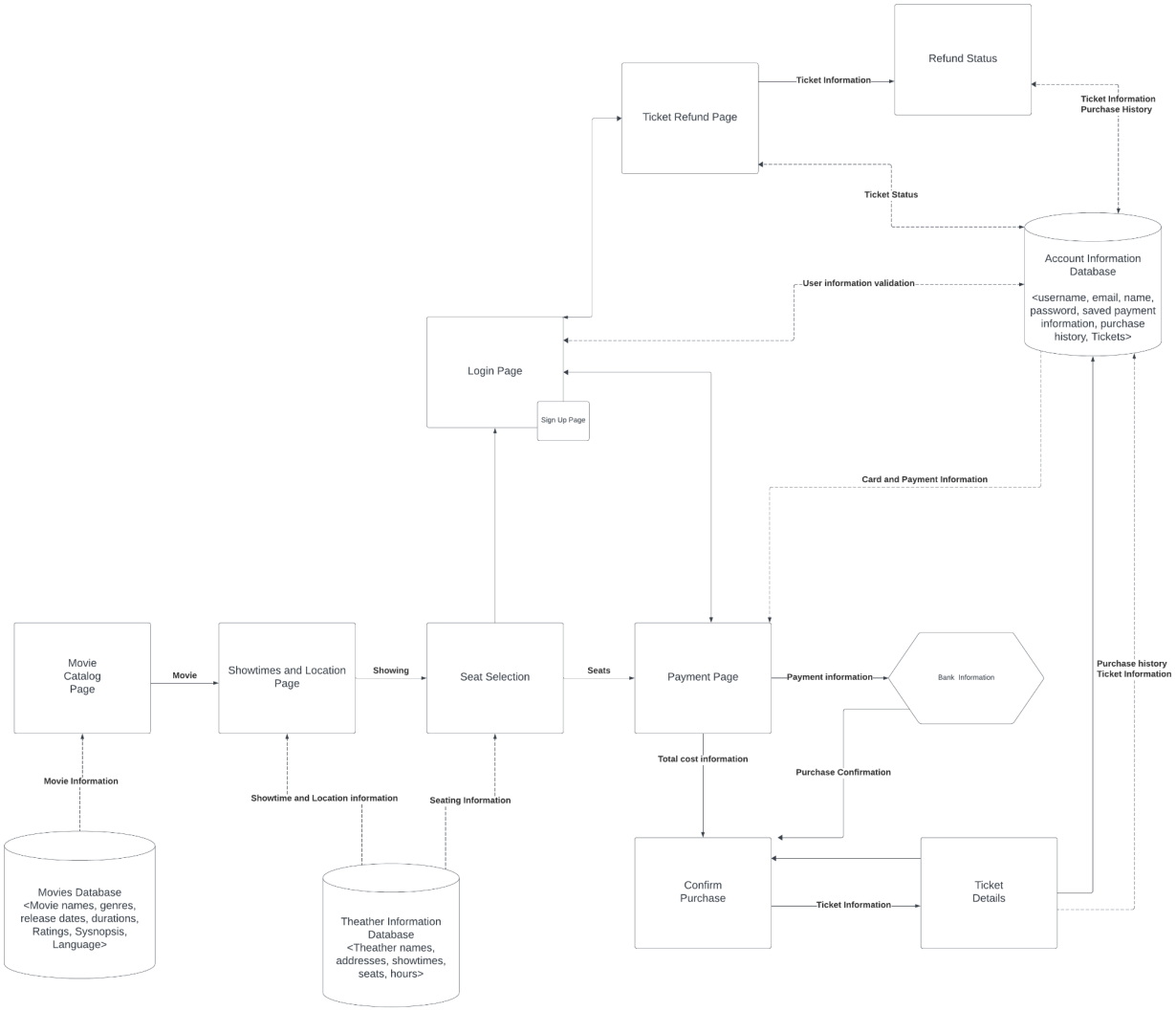
8.0 Data Management Strategy
8.1 Architecture Diagram & SQL Tables



Movie Database						
Movie Name	Genres	Release Date	Duration	Rating	Sysnopsis	Language
Saw X	Horror	9/29/23	118 minutes	6.7/10	Hoping for a miraculous cure, John Kramer	English
Five Nights at Freddy's	Horror	10/27/23	110 minutes	5.6/10	A troubled security guard begins working at Freddy Fazbear's Pizza...	English
The Exorcist: Believer	Horror	10/6/23	111 minutes	4.9/10	Exactly 50 years ago this fall, the most terrifying horror film in history landed on screens...	English
PAW Patrol: The Mighty Movie	Kids	9/29/23	95 minutes	6/10	When a magical meteor crash lands in Adventure City, it gives the PAW Patrol pups superpowers, transforming them into The MIGHTY PUPS...	English

Theater Database					
Theater Name	Address	Showtimes	Seats	Hours	Theater ID
AMC Mission Valley 20	1640 Camino Del Rio, North, San Diego, California 92108	3:50pm, 6:40pm, 9:35pm	100	12:00pm-12:00 pm	HH5231
Regal Edwards Mira Mesa	10733 Westview Pkwy, San Diego, CA 92126	2:40pm, 5:45pm, 8:45pm, 9:50pm	180	11:00 Am - 12:00 Am	J32T10

8.2.1 Data Management Strategy Outline/Plan:

The three main things of the system's data management approach are performance, security, and data integrity. Because of the organized form of the data and the requirement for transactions (atomicity, consistency, isolation, durability), a relational database system such as SQL is perfect for our system. SQL makes querying simple and guarantees data integrity. A strong system for handling such data is essential given the sensitive nature of user data, which includes financial information and personal details.

8.2.2 Database Type: SQL

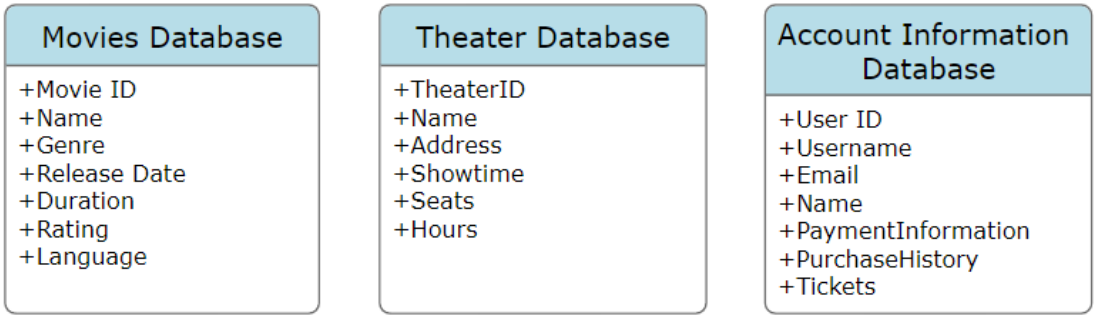
We will use an SQL database system for the following reasons:

- We selected a SQL database for the SwiftWave system because it needs relational and structured data, which is necessary given the structured nature of the data in our system.
- Data reliability is ensured via ACID transactions, particularly when processing payments and booking tickets. Strong querying capabilities are provided by this,

guaranteeing quick and effective data retrieval. An essential part of managing user accounts and checking seat availability in real-time for users attempting to buy tickets or alter any information.

- SQL's complex querying capabilities can be very good when sorting through movies, showtimes, etc.

8.2.3 Database Utilized:



- In order to guarantee organized storage and minimize the complexity of the system, data has been divided into three distinct databases. Data is segmented according to its usage and how people interact with it; these three databases also help with scalability. For example, when the number of movies or theaters increases, their individual databases can be increased without affecting the specifics of user accounts. This manages and speeds up data change and retrieval overall making the system modular and easier to manage. Also provides better security since user account data is separate from the other databases.
    1. Movies Database
        a. Includes all of the movie's information, including the title, runtime, reviews, and more.
        b. Contains movie names, genres, release dates, durations, ratings, synopses, and languages. This is essential information for users when browsing or selecting movies.
    2. Theater Information Database:
        a. To provide users with information about available theaters, showtimes, and seating.
        b. This database houses data such as theater names, addresses, showtimes, available seats, and operating hours. It is crucial for users to choose their desired location and time.
    3. Account Information Database:
        a. User account management and personalized experiences.
        b. Stores user details such as username, email, name, saved card payment information, purchase history, and tickets. This database

is vital for personalization, tracking, and ensuring secure transactions.

8.2.4 Data Storage and Retrieval

- Because SQL databases have a solid consistency model and transaction support, we'll utilize them to store structured data such as user account information profiles, movie information, and payments. This manages data storage and retrieval while taking performance, scalability, security, and compliance into account. Data integrity is ensured by ACID transactions. Payment and reservation transaction integrity is guaranteed via SQL databases.

8.2.5 Security Measures:

- Protection against unauthorized access: either intentional or accidental.
    1. Encryption: All sensitive data (like payment details, user information, etc) should be encrypted both at rest and in transit.
    2. Password Hashing: Use encryption algorithms for safe password storage according to industry standards. Allows only authorized users to access the database

8.2.6 Backup and Recovery:

- Periodic recovery tests, off-site storage, and daily automated backups are provided for disaster recovery situations that may arise from power outages, disk crashes, floods, or human error.
- All data can be restored from a catalog of all database backups that have been recorded, or from the most recent backup.

8.3 Justifications for SQL:

- The data is divided into three databases (Movies, Theater, and Account) in order to provide rapid data retrieval, modularity, and security. For instance, while the Account Information may be accessed less frequently but still needs to be protected, the Movies Database is visited more regularly.
- We have selected SQL for our databases since the data in our system is structured. Strong querying features in SQL enable quick and effective data retrieval, which is essential for user account management and real-time seat availability checks.
- SQL is the greatest option because our data is relational. Relational data is essential to the ticketing system's primary functionality. For example, it is simple to set up and manipulate when using SQL to combine data for showtimes between the Movies Database and the Theater Information Database.
- ACID properties:
    - Atomicity: Atomicity: Take the "Payment Page" and "Confirm Purchase" actions into account. A user's attempt to book a ticket initiates the following three processes: money processing, seat reservation, and ticket

generation. Atomicity makes sure that every single one of these processes such as reserving a seat, collecting payment, and generating a ticket is completed successfully, or none of them happens at all. For example, the reservation for a seat would be canceled if the payment failed.
    - Consistency: Following the "Confirm Purchase" step, the number of available seats in the "Theater Information Database" must decrease by one when a ticket is purchased. Maintaining consistency ensures that the database displays this update appropriately, avoiding situations in which a seat is booked twice or remains available after it has been reserved.
    - Isolation: Imagine that two individuals are attending the "Seat Selection" simultaneously. When two people attempt to reserve the same seat at the same time, isolation ensures that the system handles their requests so that only one person can take the seat. A notification stating that the seat is taken will be sent to the other individual.
    - Durability: Following confirmation of the purchase, the user's ticket information and payment history ought to be permanently preserved in the "Account Information Database" and "Purchase History." The durability ensures that, even in the event of a system failure shortly after a ticket is booked, the booking data remains intact and is retrieved when the system restarts.
    - Adding ACID qualities will make the ticket booking and managing system trustworthy, strong, and without mistakes or problems.
- SQL databases, such as MySql, Oracle, Sqlite, Postgres and MS-SQL are picked because they have the right features for our SwiftWave. SQL databases are a suitable fit because of the relationships between organizing data and defining and managing it. This configuration ensures it can scale, remain safe, and function well. Plus, having a database setup that's modular makes it simpler to handle, backup, and expand parts of the system without impacting the whole thing.

8.4 Tradeoff Discussion:

- We are able to handle data in an organized and efficient manner by using SQL, but we may run into issues with growth when this system expands into new projects and begins to sell tickets for events other than movies. Our flexibility and speed are increased when we have multiple databases, but maintaining them requires us to ensure that the data is consistent across all of them.
    - Why choose SQL?:
        - Structured data, powerful querying capabilities, and established ACID properties.

        - Not a lot of huge volume of structured, semi-structured, and unstructured data to handle at the moment.
        - NoSQL also lacks the ability to perform dynamic operations so it can't guarantee ACID properties.
        - SQL also offers consistent transactions, crucial for financial activities.
        - Maximum security is needed for storing sensitive user data, such as payment information. Data encryption should always be used, especially for sensitive fields like passwords.
        - SQL databases may not grow as effectively horizontally, but they do prioritize data quality and durability.
        - Simplifies the architecture but increases risk and may not scale well with high demand, unlike NoSQL.
        - Monthly costs may be very high with NoSQL compared to SQL
    - Why choose NoSQL instead of SQL?:
        - It might not be as scalable as NoSQL for very large datasets and can have performance issues if not optimized correctly.
        - Enhance security and performance but add complexity to the database system. Because we have three databases, some procedures may become more complicated because they may need to access and merge data from different databases.
        - Ideal for data that is not structured. Not suited for our extremely transaction-based present system for SwiftWave.
        - Although NoSQL databases are renowned for their excellent performance, there are situations in NoSQL that are very inconsistent.

Even though there are alternative approaches to organizing and storing data for our SwiftWave system, we have chosen to use SQL for our database because it provides a balance between organization, security, and scalability. While SQL databases offer a reliable framework for relational maintenance, separate databases ensure enhanced security and modularization.