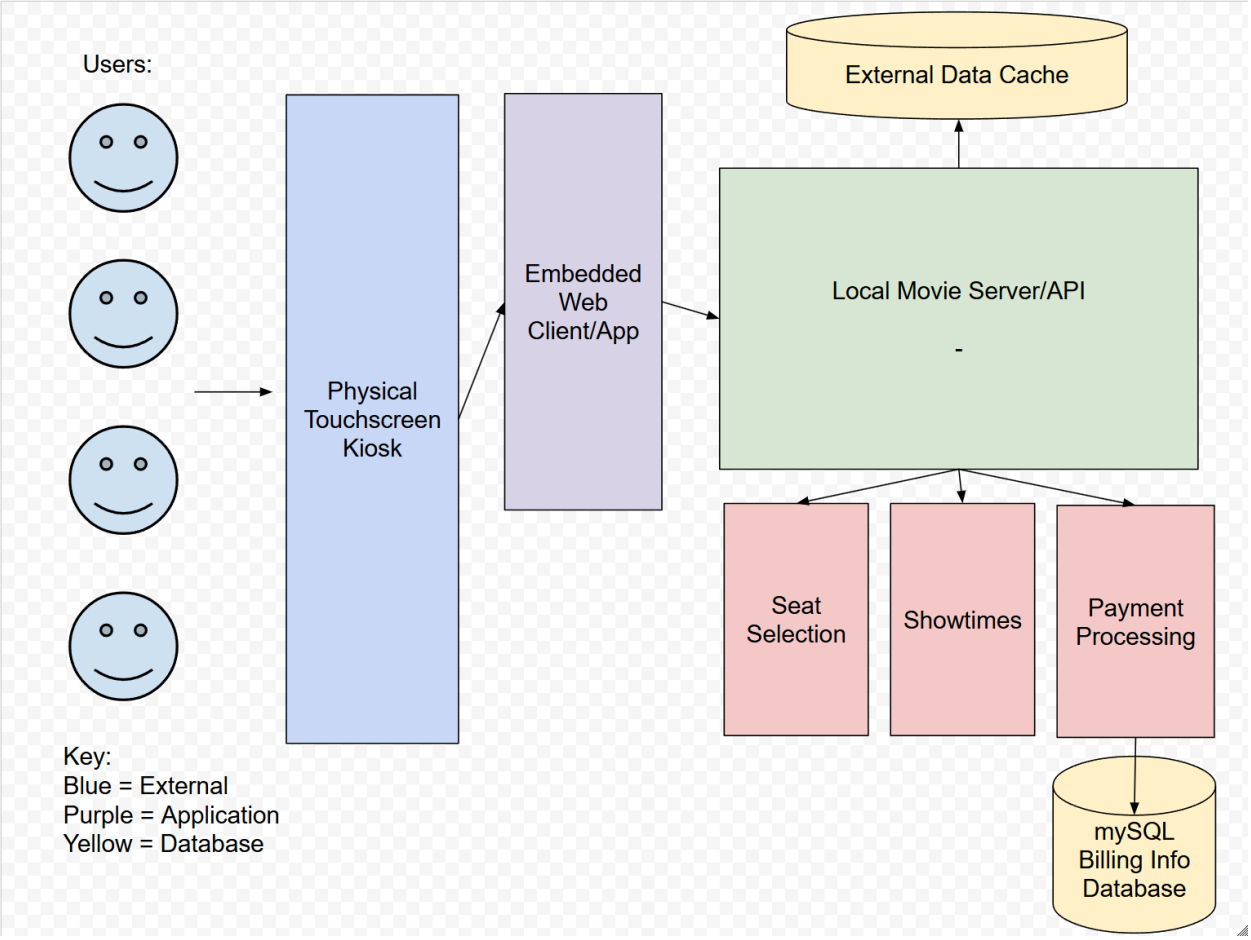
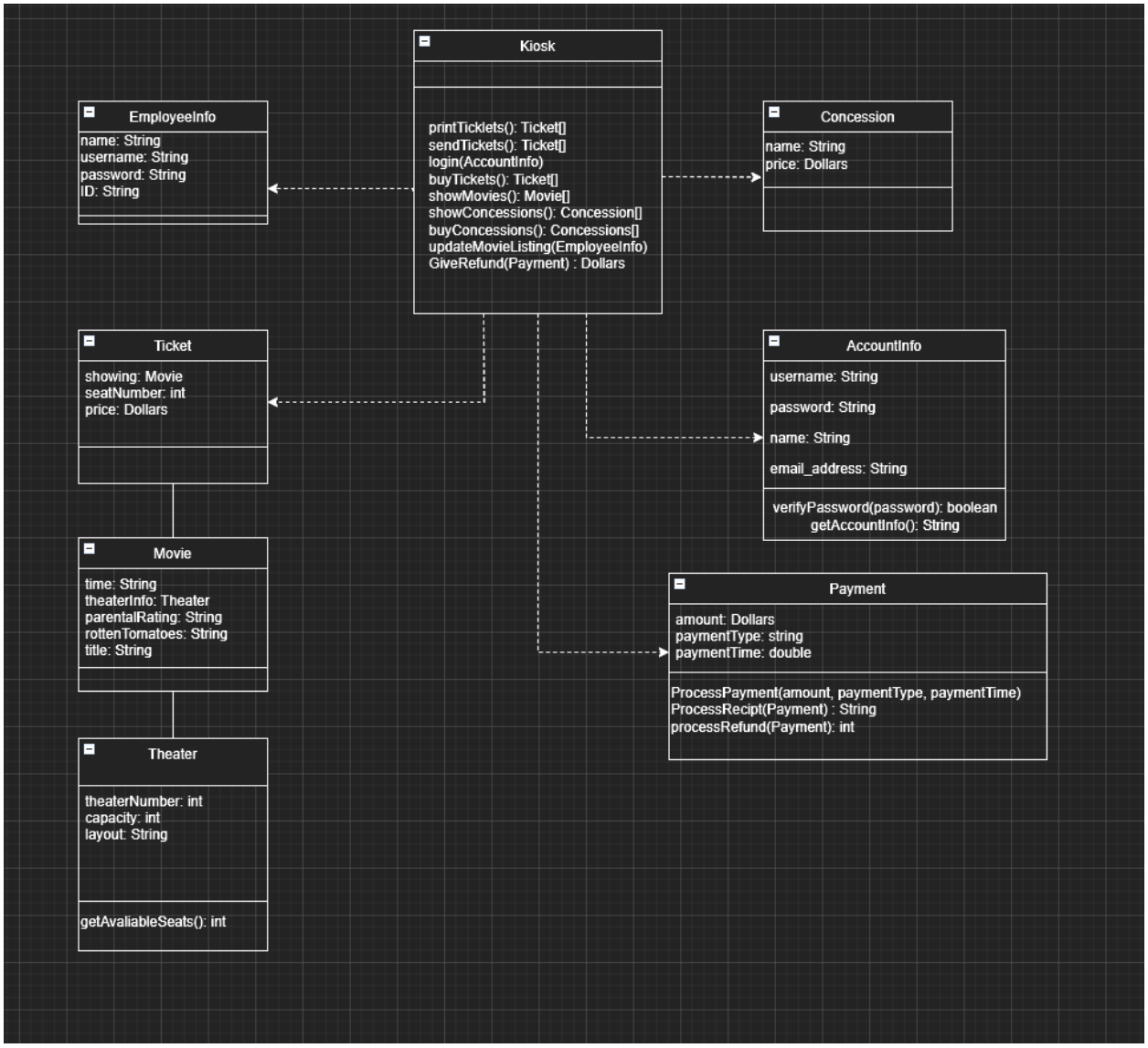

Users:
Physical Touchscreen Kiosk
Embedded Web Client/App
External Data Cache
Local Movie Server/API
-
Seat Selection
Showtimes
Payment Processing
mySQL Billing Info Database
Key:
Blue = External
Purple = Application
Yellow = Database



Data Management Strategy

- **Choice of Database:**
    - The document states the system needs a database to store movie information, screening times, ticket availability, and booking details.
    -

    - Given the relational nature of the data (movies have showtimes, showtimes are in theaters, tickets are for specific seats), a **Relational Database Management System (RDBMS)** like MySQL, PostgreSQL, or SQL Server is a suitable choice.
    - **Justification:** RDBMS excels at managing structured data with relationships, ensuring data integrity (preventing double bookings, etc.), and providing efficient querying.
- **Database Structure (Logical Design):**
        - **Movies:** MovieID, Title, Genre, Description, Poster, ParentalRating
        - **Theaters:** TheaterID, TheaterNumber, Capacity, Layout
        - **Showtimes:** ShowtimeID, MovieID (FK), TheaterID (FK), DateTime
        - **Seats:** SeatID, TheaterID (FK), SeatNumber, Row, Column
        - **Tickets:** TicketID, ShowtimeID (FK), SeatID (FK), Price, QRCode, TransactionID, Type
        - **Transactions:** TransactionID, PaymentMethod, Amount, TransactionTime, Status
    - **Relationships:**
        - A Movie can have multiple Showtimes.
        - Showtime is in one Theater.
        - A Theater has many Seats.
        - A Ticket is for one Showtime and one Seat.
        - A Transaction can include multiple Tickets.
- **Data Splitting:**

    - The tables are logically separated to reduce data redundancy and improve maintainability.
    - For example, movie information is stored separately from showtime information, and seat details are separated from ticket details.

- **Possible Alternatives:**
    - **NoSQL Databases:**
        - Alternatives like MongoDB or Cassandra (NoSQL databases) could be used, especially if the system needed to scale to handle massive amounts of data or if the data structure was less rigid.
        - **Tradeoffs:** NoSQL databases might offer better scalability but could sacrifice some of the data integrity features of RDBMS (like transactions and constraints). For this application, the strong consistency and relational features of an RDBMS are generally preferred to prevent issues like double booking.
    - **Data Warehousing:**
        - For analytics and reporting (e.g., sales trends, popular movies), a separate data warehouse could be used. Data from the operational database would be extracted, transformed, and loaded (ETL) into the data warehouse.
        - **Tradeoffs:** A data warehouse adds complexity but provides optimized storage and querying for analytical purposes, without impacting the performance of the main ticketing system.