Patrick Simpauco

Jenny Tran

Johnny Rosas

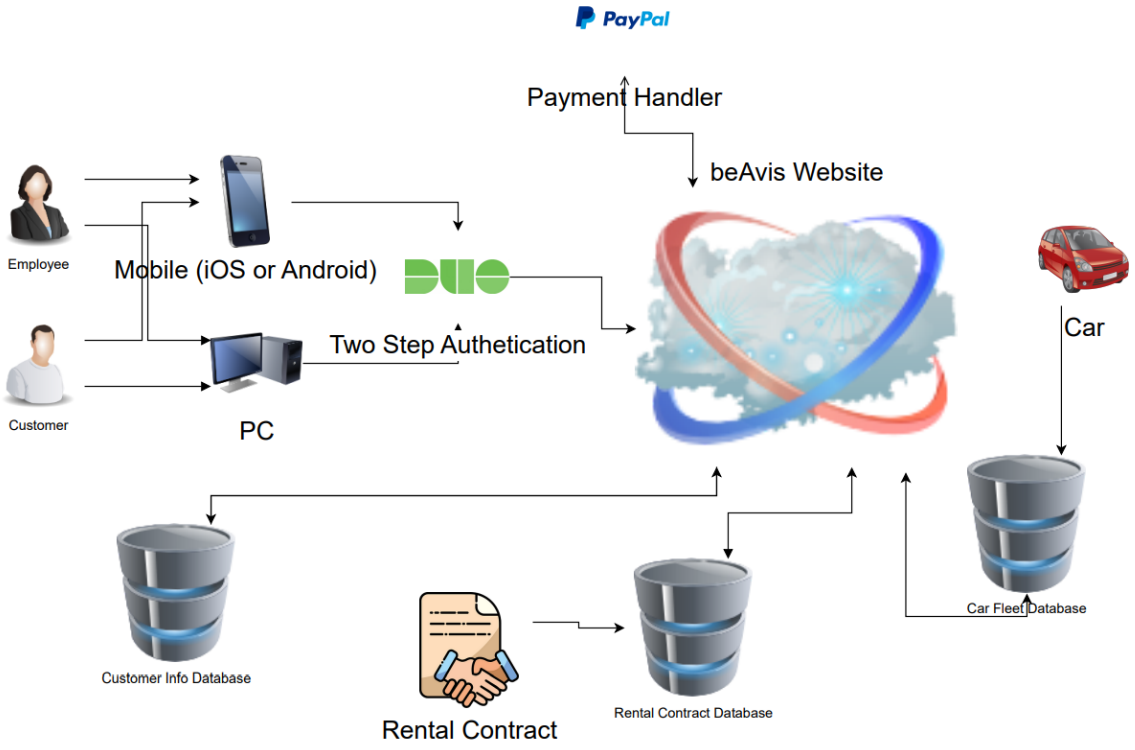
CS 250

Professor Donyanavard

November 10, 2023

Assignment 4

1. **Software Architecture Diagram:**



Updates: To track rental contracts, cars, and customers as individual entities, a third customer info database is added.

2. **Data Management Strategy**

Data management strategy:

- Database structure choice:
    - **Relational Database Management System (RDBMS)**: like MySQL or PostgreSQL for structured data management
    - Justification:
        - Making sure that the car rental system's critical data, such as user information and rental transactions, maintains accuracy and integrity
        - All data stored in each database is specifically structured and can be represented within a table. All points of data should be directed towards one another. Ex: A carID within a contract.
        - Entity Relationship Diagrams can help visualize the relationships and cardinalities between different data entities (e.g users, cars, contracts,...)
- Data security and access control:
    - Implement security measures to safeguard sensitive data
    - Encrypting data to protect:
        - User credentials
        - Payment information
        - Other personally identifiable information (PII)
    - Define user roles and access permissions
        - Ensure authorized prevent unauthorized data

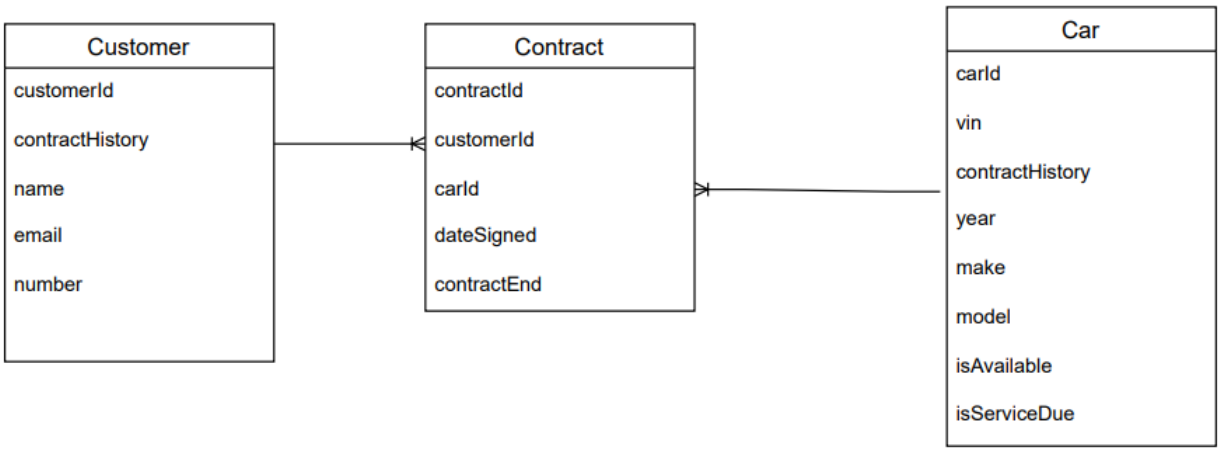
- Scalability and performance:
    - Monitor & optimize database performance for efficient processing and response time
    - Consider scalability options (such as sharding or partitioning) if the database grows in size or experiencing increased traffic
- Existing Vendor Support:
    - There is a larger community of developers and vendors that support MySQL database

Design decision:

- Data division and structure: 3 databases:
    - User Data: Store user profiles and preferences.
    - Rental Contract Data: Record rental transactions, contract details, cars involved and customer interactions.
    - Car Fleet Management: Manage car inventory, availability, and maintenance status.
    - Justification: 3 separate databases with 3 different entities allow each object to be viewed individually. Each entity is individually indexed to track the history of objects easily and to relate objects to each other.
- Logical division:

- Consider table structures or entities for each data type. For instance, 'Users' table to store user profiles, 'Rentals' for rental records, 'Cars' for vehicle details, and 'Contracts' for contract information.

Entity Relationship Diagram



Cardinalities:

One to many: Car to Contract, Customer to Contract

- Reasoning: A single customer can have multiple contracts (expired or open) within their contract history. A single contract is specific to a single customer. To track the rental history of a single car, contracts can be listed within its contract history.

Many to one: Contract to Car, Contract to Customer

- Reasoning: Since contracts have similar data but different field values, multiple contracts can be stored in the contract history of a Car or Customer entity. A car could have been rented multiple times in the past and a customer could have signed multiple contracts.

- Trade offs/alternatives:
    - Database structures:
        - NoSQL databases: For certain data types like logs or semi-structured data, NoSQL databases might offer more flexibility
        - Structured vs. semi-structured:
            - Using a single SQL database keeps the data structured but might limit flexibility.
            - Integrating NoSQL for specific data types could offer more flexibility but complicates the overall management. Ex: Document organization for the Contract database
        - Single Database vs. Multiple Databases:
            - Single database: simplifies data management and maintains data integrity but might limit scalability and representation of multiple objects.
            - Multiple databases: can enhance performance and scalability but may introduce complexities in data consistency