Car Rental Design 2.0

Prepared by: Andressa Wu (824360167), Sadia Abdirahim (826998101), Trevor Thayer (826194012)

1. **SYSTEM DESCRIPTION:** The purpose of this document is to specify the requirements for a simple software application for a Rental Car System. This system is to be used by both employees and customers to keep inventory, pricing, and payments up to date.

    1.1. **Customer/Employee Management:** The system should allow all users (both employees and customers) to create and manage personal accounts. For employees, it should consider different information authorization levels based on employee level. Customers should be able to view their rental history and update personal/payment information.

    1.2. **Car Management:** The system should allow employees with proper authorization to add and remove car models. Each model should have specifications of year, make, model, color, and gas mileage. Customers should be able to search for cars based on filters relevant to car specifications.

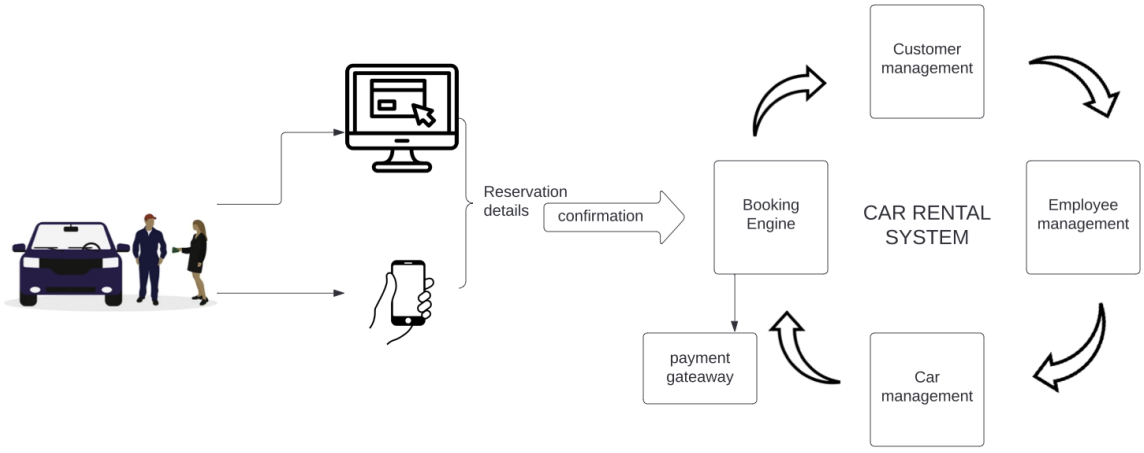
    1.3. **Booking Management:** The system should allow for reservations to be made in advance with specific dates, as well as in person. It should compare these dates with availability for the specific vehicle requested.

    1.4. **Rental Management:** Each employee should be able to check out vehicles to customers. They should also be able to take vehicles out of circulation based on damages. During each check in and check out, the system should require an update of date and time, gas level and any damages. The system should then calculate any additional fees based on the time the car was utilized and any damages.

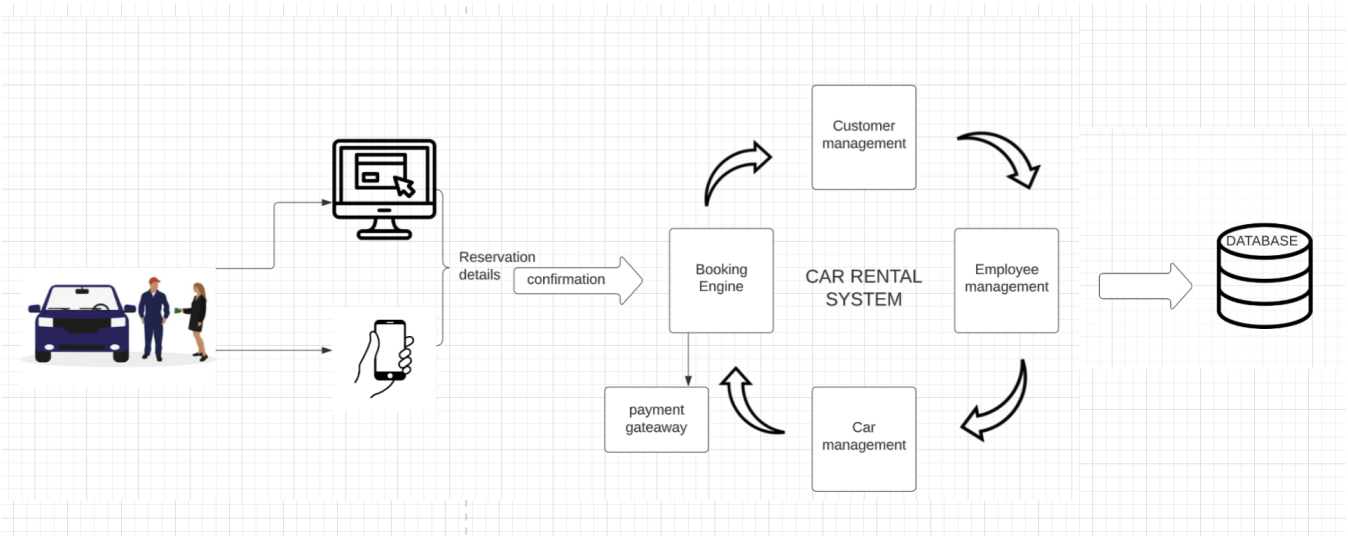
    1.5. **Payment Management:** The system should allow for payments in advance, as well as in person payments. Cash and card should be options for in person payments. Processing refunds should be handled as well. Each payment should be added to the account of the customer, as well as a record of all payments received.

2. SOFTWARE ARCHITECTURE OVERVIEW

2.1. Architectural Diagram of All Major Components:



2.2. New Architectural Diagram of All Major Components:



2.3. Description of Architecture Diagram:

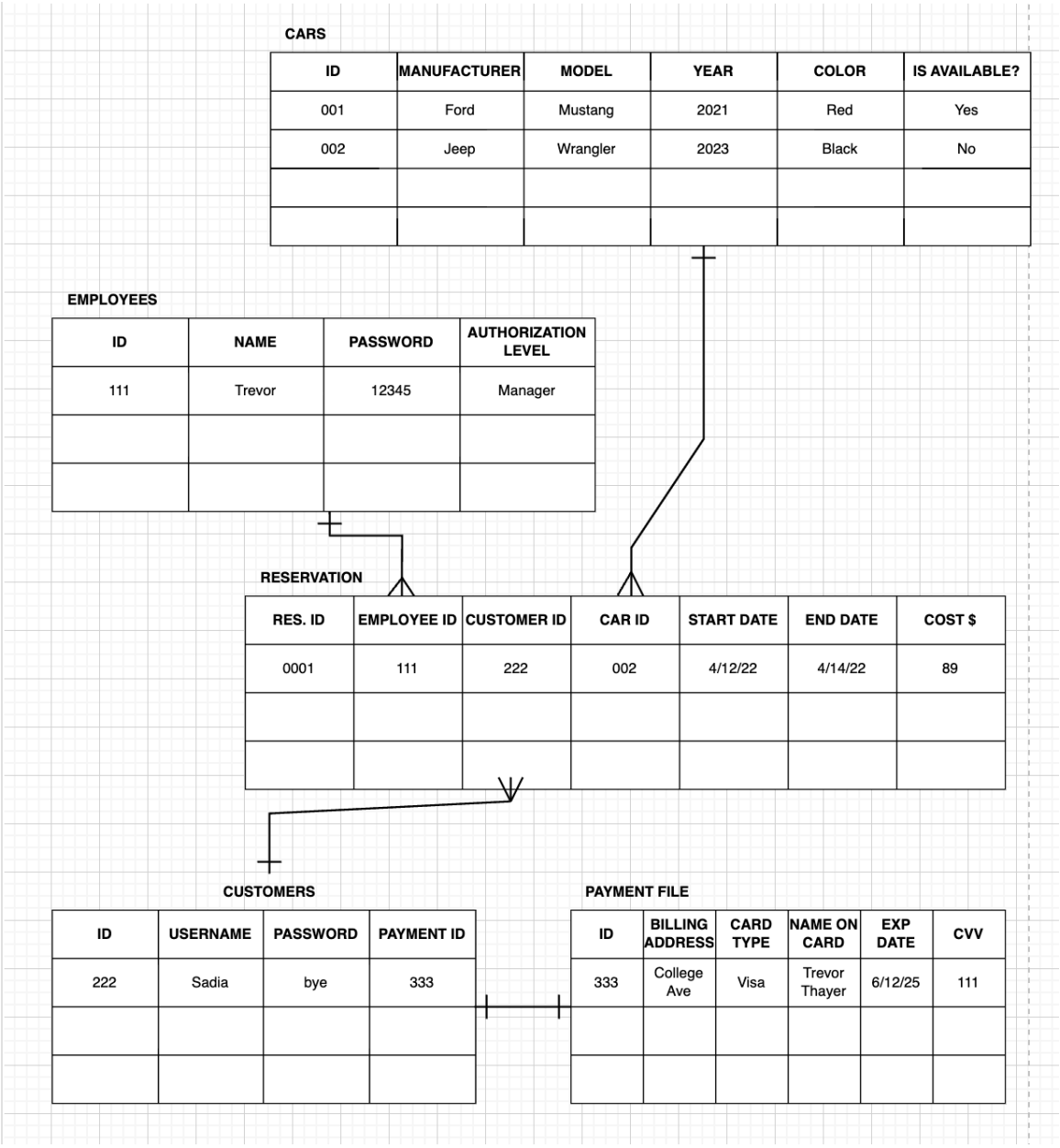
The diagram above represents the architecture diagram of a car rental system. The system is to be used by the customers and the employees of the company. In order to access the system, they can either use it through a mobile application or the website version. The reservation details are linked to the booking engine which is at the forefront of the system. Through the booking engine, customers and employees can access the cars, the costs, availability, maintenance, payments and among other things.

2.4. Explanation of the Software Architecture Diagram Change:

A database was added to the software architecture diagram because it is an important part of the system. The database is a collection of data that is organized so the information within can be easily accessed whenever it is needed. This database layer contains the SQL database which stores all the data related to cars, payments, employees, clients, reservations, and other relevant information.

3. DATA MANAGEMENT STRATEGY

3.1. Diagram



3.2. Explanation

In a reservation for a car rental, there are various types of information that are necessary to be accessed. This is clearly depicted in our SQL Database Design. This system is structured such that the Reservation Class holds all the relevant information for each reservation in the system. This consists mostly of ID's, which can be found in the other classes and relate to more information. Firstly, each Customer has 1 PaymentFile, which holds their general payment information. This is a one-to-one relationship because for each customer, there is only one payment file.

In this system, the databases will keep track of all current and past reservations. This means that it is possible that each employee is present in multiple reservations. Hence, there can

be many different reservations that have the same customer, employee, or car stored in the database. Therefore, between the employee and the reservation, there is a one-to-many relationship, as there are many reservations, but there are unique employees. This same logic applies to the relationships between reservations and cars, as well as reservations and customers.

As shown in the diagram and explained in the logic above, Customers and PaymentFile have a one to one relationship. Customers, Employees, and Cars in relation to Reservations all have a one to many relationship.

3.3. Trade Off Discussion

For our design, we decided that an SQL diagram would best accommodate and reflect our design choices. Unpacking this from a data management perspective, using an SQL diagram proved to be an exceptional approach in generating our diagram because it illustrates correlation between our provided data in an efficient manner. Since our Car Rental system is always subject to change (i.e customer changing reservation dates, cancellation, etc.) an SQL diagram would best be suitable for our design decision because its data can easily be manipulated. Although noSQL diagrams do not depict relationships between entities, it can be utilized as an alternative because it provides flexibility in terms of inputting and managing data.