

# Rental Car Software System

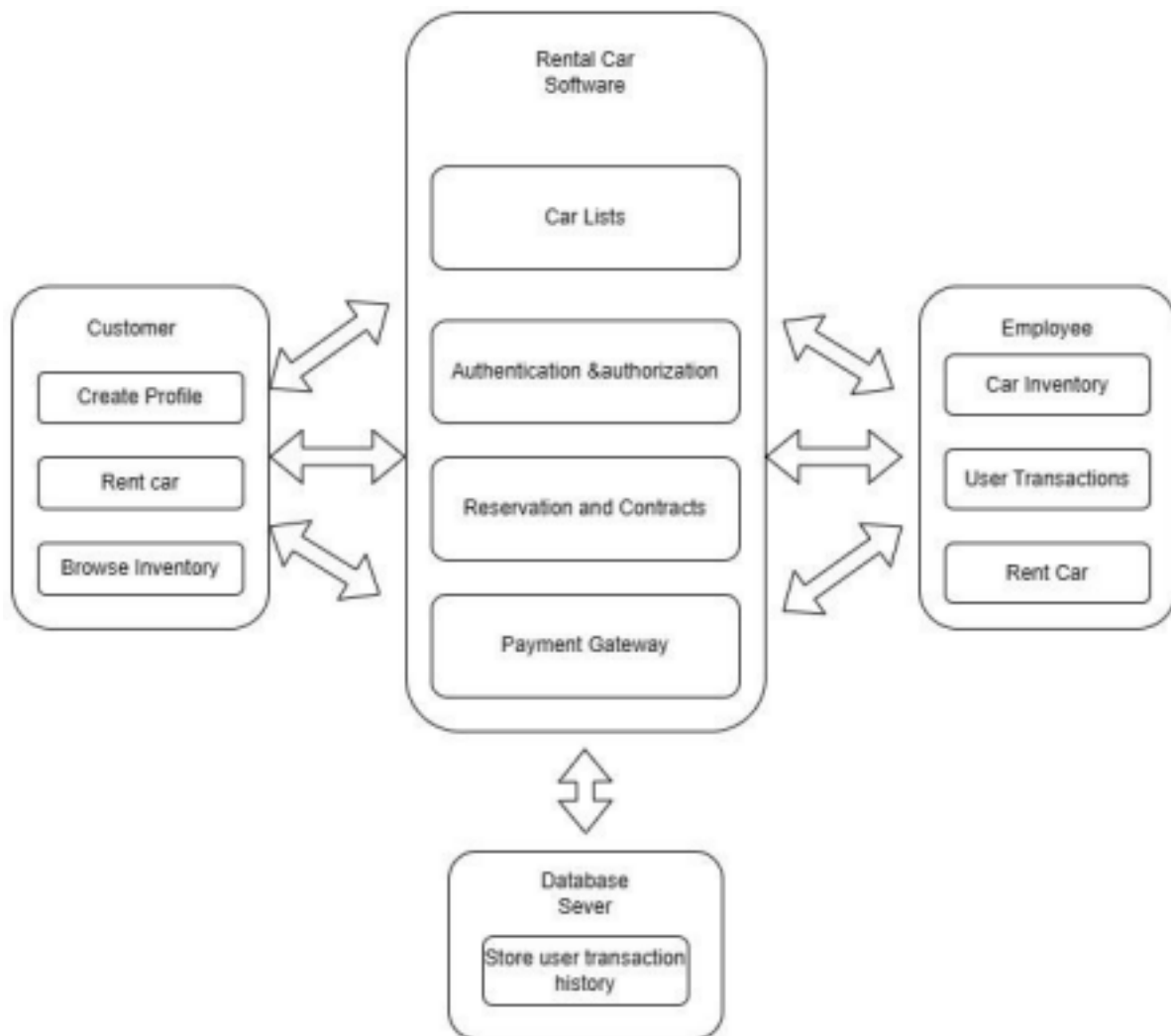
By:

Victor Lopez  
Hooman Manesh  
Patrick Ho

### Brief System Overview:

The car rental system aims to make the business more efficient by digitalizing the current system. The company requesting this system uses paper and pen to track available cars, the car's condition, and guests that rent the cars. This procedure is extremely slow and can be made efficient with a system that is easy to understand. Customers are first required to create accounts with verified emails to start renting. Then, they add a payment method depending on if they rent on the website or in-person. Finally, they choose which car they want and the process is finished. Employees can view customer information and update the cars' info. This includes car availability, orders, conditions, and insurance. They are also able to scan required documents into the system.

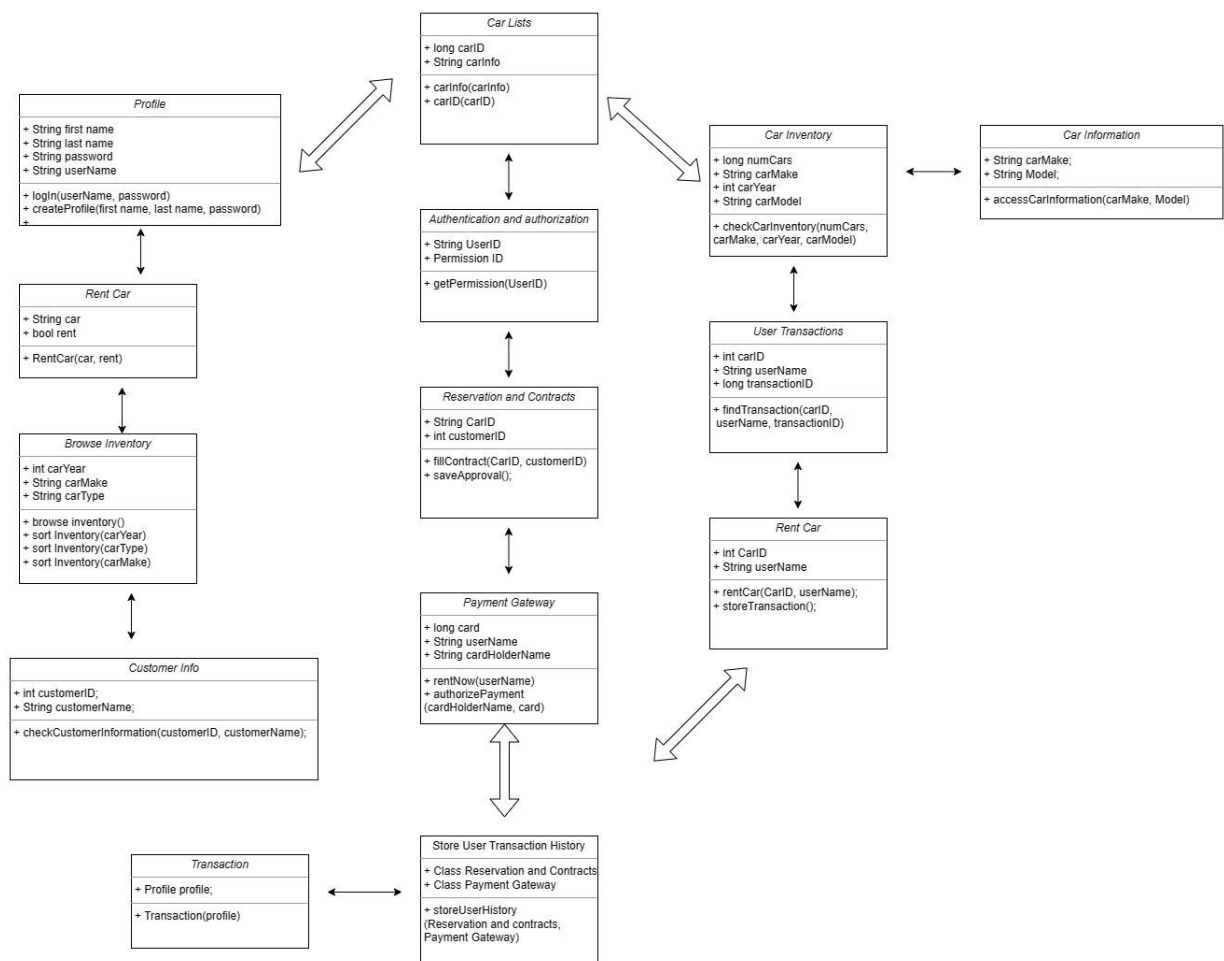
### SWA diagram:



## SWA Description:

Our diagram for the software architecture for the Rental Car Software is separated for different main users in the front end being customers (left), and employees on the right. Their general objectives and functions are connected to the central pillar which hosts essential back end functionality for the application to run. Everything either user would require, they can gather. The central hub being connected to the database server for data management and storage. Overall, the SWA was kept minimal with more UML diagrams and tests added below.

## UML Class diagrams:



### **Class In-Depth Descriptions:**

Class: Profile

Functions: login and create profile

Variables: 4 strings for first name, last name, password, and user name.

Description: Profile embodies all functions regarding profile creation or log in for the user to access the site on their own time and when needed.

Class: Rent Car

Functions: rent car

Variables: string car, boolean rent

Description: Basic function for user to choose if they want to rent a car.

Class: Browse Inventory

Functions: browse inventory, sort inventory (by year), sort inventory (by type), sort inventory (by make)

Variables: car year, car make, car type

Description: Class is meant to allow customer to browse the inventory however they would like by whatever parameters they prefer.

Class: Car Lists

Functions: car info, car ID

Variables: car info, and long carID

Description: based on the general variables regarding the car, the software will be able to save and list available cars for both user and employee.

Class: Authentication and authorization

Functions: fill contract, save approval

Variables: carID, customer id

Description: functions to fill the contract and save the approval. SaveApproval will generate new variables within function to return and save under transaction history.

Class: Reservation and Contracts

Functions: fill contract

Variables: car id, customer id

Description: class is meant to reserve and generate contracts for the customer and store them in the database.

Class: Payment Gateway

Functions: rent now, authorize payment

Variables: card, user name, card holder name

Description: payment gateway is meant to process payments when vehicles are rented.

Class: Store User Transaction History

Functions: store user history

Variables: classes for (reservation and contracts) and (payment gateways)

Description: Functions to store all the transactions regarding the rental of vehicles.

Class: Car Inventory

Functions: check car inventory

Variables: num cars, car make, car year, car model

Description: meant to show employee the car inventory available for customers to choose from.

Class: User Transactions

Functions: find transaction

Variables: carID, username, transaction id

Description: meant for employees to be able to find customer transactions if desired.

Class: Rent Car

Functions: rent car, store transaction

Variables: car id, user name

Description: Functions to allow employee access to rent cars for customer and streamline the process.

### **Development Plan:**

We plan to have an SWA diagram, UML diagram, and an overview of the system by October 13, 2023. Hooman is responsible for creating the SWA diagram and description. It describes the architecture of the system and guides us during the building process. Victor will create the UML diagram that goes into further detail about the system. It covers the classes and objects that need to be coded to create the system.

## **Verification Test Plan:**

For verification that our software will work as intended by the user and the company, the following tests will be implemented.

### **1 Test Set**

This test set is going to cover the plans to verify the system's ability to create an account that can be repeatedly used for renting cars. This feature will be tested with the partition method because there are far too many inputs for brute force and random is not guaranteed to showcase inputs not in email format.

#### **1.1 Unit**

Input:

- Email: cs250@gmail.com
- Password: cs250

Method:

- Profile x = new Profile(cs250@gmail.com, cs250)

Output:

- New Profile Object
  - Email: cs250@gmail.com
  - Password: cs250
  - Payment: None

Input:

- Email: awesomesauce
- Password: cs250

Method:

- Profile x = new Profile(awesomesauce, cs250)

Output:

- No Profile Object is created
- Field warns user to enter in email format

#### **1.2 Integration**

The account creation feature is not inside of any other methods, but adding a payment method is required for renting online.

Input:

- Card Number: 0000 0000 0000 0000

Method:

- x.addPayment(0000000000000000)

Output:

- Profile x's payment attribute is updated from None to 0000000000000000.

### 1.3 System

Having a profile is an important part of the system because every customer needs one to rent a car online and in-person. Transactions will interact with the accounts by acquiring the payment information. Car info can also be updated to show which profile is currently renting the car.

Input:

- Profile: x

Method:

- Transaction y = new Transaction(x)

Output:

- A new Transaction object that has account info

## 2 Test Set

In our car rental application, we want to ensure that employees can easily use the system for their regular tasks smoothly. Features such as checking the availability of the lot, customer information and car information, will help employees provide accurate information to customers and manage the inventory efficiently.

### 2.1 Inventory

This scenario is about requesting to check the car lot availability. It can be initiated by an employee selecting a specific date and possibly other filters like car type, location, or availability status.

Input:

- Date (the date for which the availability is to be checked)
- Car Type (optional filter for a specific car type, e.g., compact, SUV, luxury)
- Location (optional filter for a specific rental location)

**Method:**

Create a method called `checkCarAvailability` in your application. This method takes the input parameters mentioned above.

**Output:**

- A list of available cars that match the specified criteria.
- List of Car objects, where each Car object includes details like the car's make, model, year, rental price, and availability status.

**2.2 Customer Info:**

In this section, we want to make sure that the employees can easily access customer information for account management, inquiries, or assistance. This feature allows employees to retrieve and view customer details efficiently.

**Input:**

- The input for this scenario is a request to check customer information. It can be initiated by an employee searching for a specific customer's account. The input parameters may include:
  - Customer ID (or other unique identifier)
  - Customer Name (optional for searching by name)

**Method:**

We create a method called “`checkCustomerInformation`” in our application. This method takes the input parameters mentioned above.

**Output:**

The output is a Customer object that includes customer details such as name, contact information, rental history, and other relevant account information.

**2.3 Car Information:**



In this test case, we want to see if employees can access detailed information about the cars in our rental fleet. This feature will help employees provide accurate and informed answers to customer inquiries about the available cars.

**Input:**

The input for this scenario is a request to access car information. It can be initiated by an employee searching for a specific car's details, including its make, model, year, rental price, and availability status. The input parameters may include:

- Car Make (optional for searching by car make)
- Car Model (optional for searching by car model)

**Method:**

We will use a method called “accessCarInformation” in our application.

**Output:**

The output of the accessCarInformation method is a Car object that includes detailed information about the car, including its make, model, year, rental price, and availability status.