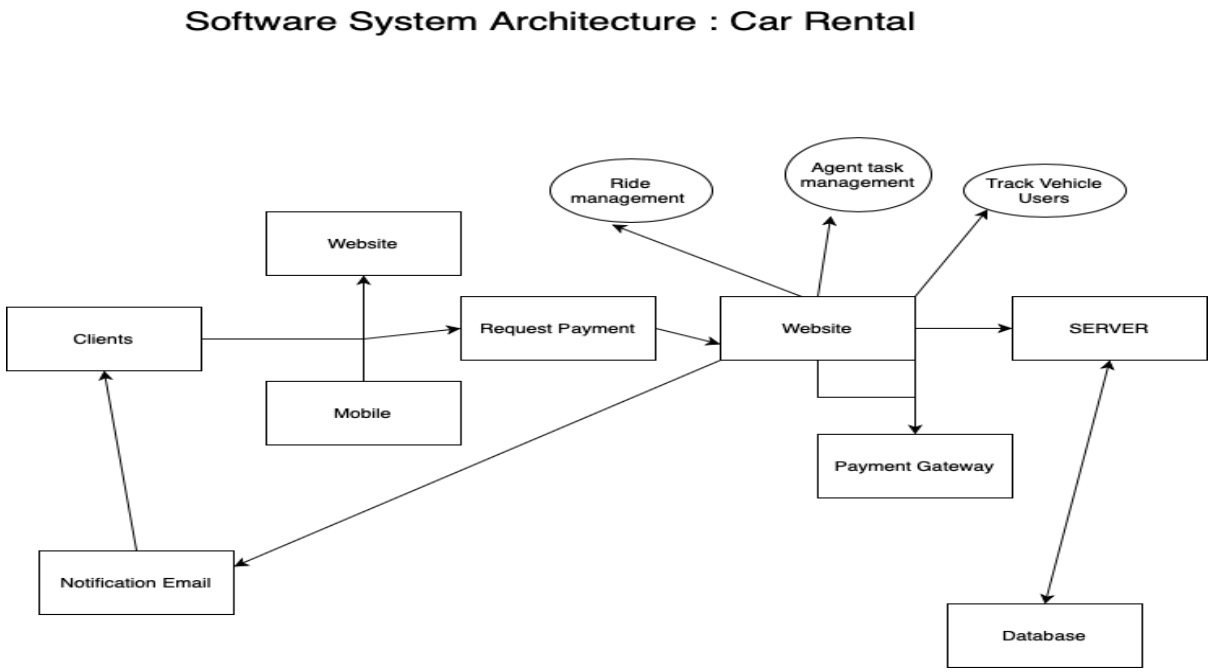
Software Design Specification

Software Title: CarRentalPro

Team Members:
1. Jesus Serna
2. Matthew Sprague

System Description:
Brief Overview of System: CarRentalPro is a comprehensive car rental management system designed to streamline the process of renting vehicles. It provides an intuitive user interface for customers to book and manage reservations while offering robust backend functionality for administrators.

Software Architecture Overview:
Architectural Diagram:

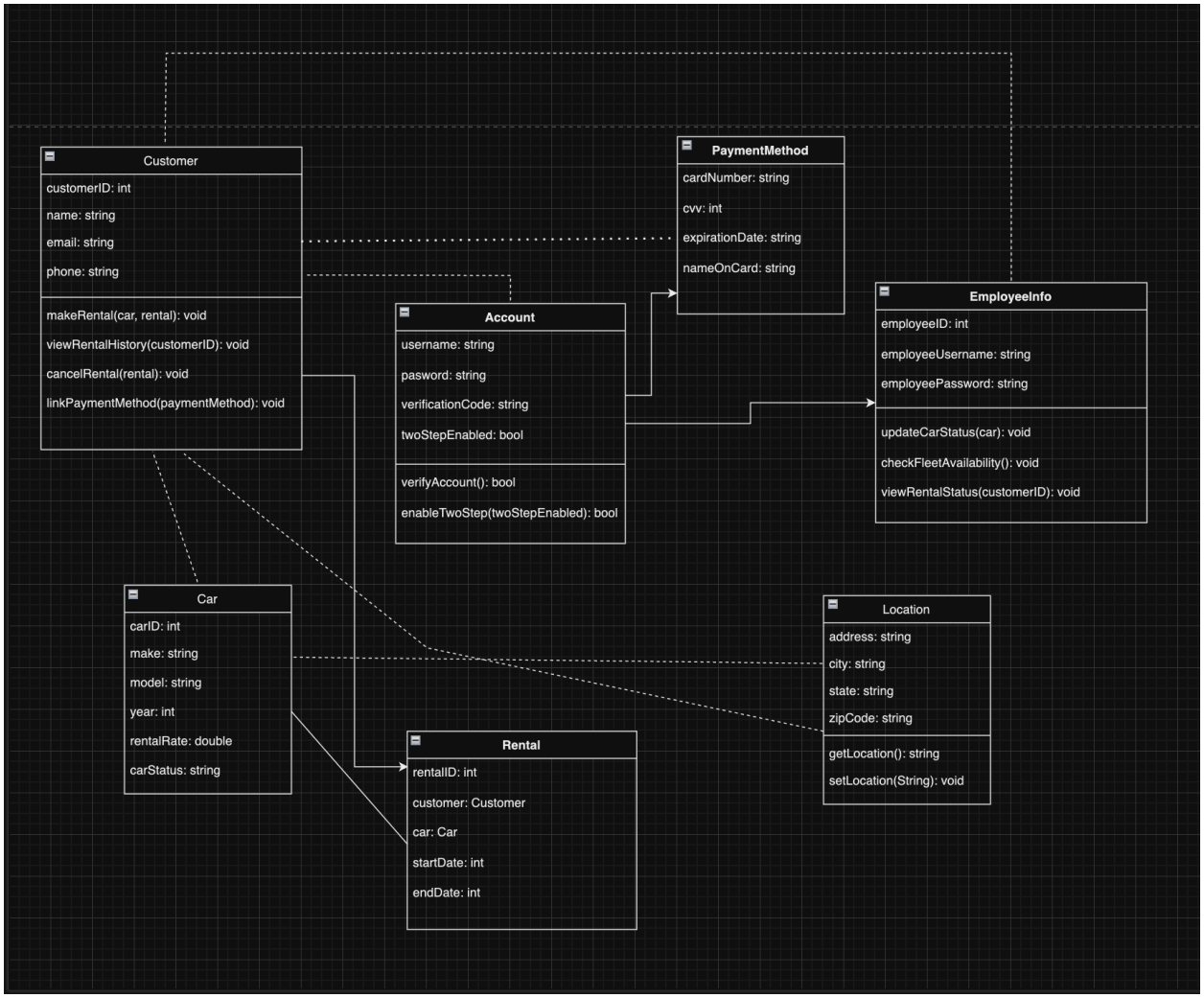


Software System Overview:

1) Mobile and Web Users: represent the interaction through the browser or mobile app which includes registration, vehicle booking and management.
2) Server: The functionality is to act as the central processing unit which handles business communication with a database
3) Database: The functionality is to store and manage data that is relevant related to users.

4) The Website system communicates via accessing specific URL and communicates with the server request, it utilizes HTTP/HTTPS protocols for secure data transfer and uses APIS for data exchange. To store data SQL is use so that it becomes efficient.

UML Class Diagram:



Description of Classes:

Customer: represents a customer of the car rental company
Attributes:

- customerID: int - unique ID the customer receives when account is created so info can be looked up

- name: string - the customer's name
- email: string - the customer's email address
- phone: string - the customer's phone number

Operations:

- makeRental(car, rental): void - initiates a rental for the customer by utilizing an object from the car class and an object of the rental class as parameters
- viewRentalHistory(customerID): void - using the customer's unique ID, past rentals by that customer are displayed
- cancelRental(rentalID): void - cancels a current rental by using the unique rental ID
- linkPaymentMethod(paymentMethod): void - links a payment method with the customer's account

Car: represents a car in the fleet

Attributes:

- carID: int - the unique ID number for the car so info can be looked up
- make: string - the brand/manufacturer of the car
- model: string - the model of the car
- year: string - the year the car was made
- rentalRate: double - the cost of renting the car per day
- carStatus: string - whether the car is available for rent, currently being rented, or under maintenance

Account: represents the account that is created for the customer

Attributes:

- username: string - the customer's created username
- password: string - the customer's created password
- verificationCode: string - a code sent to the customer when creating an account so the customer can verify their email address
- twoStepEnabled: bool - indicates if customer has enabled two step authentication for when they sign in to their account

Operations:

- verifyAccount(): bool - checks if the customer's account is verified
- enableTwoStep(twoStepEnabled): void - enables two step authentication if the customer indicated they wanted it enabled

Rental: represents the rental info for the car the customer rented

Attributes:

- rentalID: int - the unique ID number for the customer's rental so info can be looked up
- customer: Customer - uses an object of the customer class to indicate who is renting the car
- car: Car - uses an object of the Car class to indicate what car is being rented
- startDate: int - the start date of the rental period
- endDate: int - when the car must be returned

PaymentMethod: represents the customer's payment method for making rentals
Attributes:

- cardNumber: string - the customer's credit card number
- cvv: int - the card verification value of the customer's card
- expirationDate: int - the expiration date of the customer's card
- nameOnCard: string - the name on the card

Location: represents the physical location of where the car can be picked up for rental
Attributes:

- address: string - the address of where the car is located
- city: string - the city of the where the car is located
- state: string - the state of where the car is located
- zipCode: string - the zip code of where the car is located

Operations:

- getLocation(): string - returns the location of the car
- setLocation(String) - identifies the location of the car

EmployeeInfo: represents the employee account so they can access the system to view information or make changes
Attributes:

- employeeID: int - the ID number of the employee
- employeeUsername: string - unique employee username for their account
- employeePassword: string - unique employee password for their account

Operations:

- updateCarStatus(Car): void - takes an object of the car class so the status of the car can be updated to show if it is being rented, available, or under maintenance
- checkFleetAvaibility(): void - shows the cars available for rent
- viewRentalStatus(customerID): void - indicates the status of a customer's rental

Development plan and timeline

- Partitioning of tasks:
    - Define Project roles and responsibilities
    - Gather requirements by conducting interviews and documentation
    - System Design finalization including Software architecture, UML diagrams and database schema
    - Development by implementing the logic, setting up database and develop APIs, for frontend implement customer and admin interfaces.
    - Test by doing unit testing, integration testing, UAT and fix bugs for optimization and document.
    - Finalize product and deploy.

- Team member responsibilities: Team members will share all software system duties equality.