BeAvis Car Rental System

By: Brianna Garcia, Jose Arroyo Redondo, Nathan Moreno, Gabriel Magana

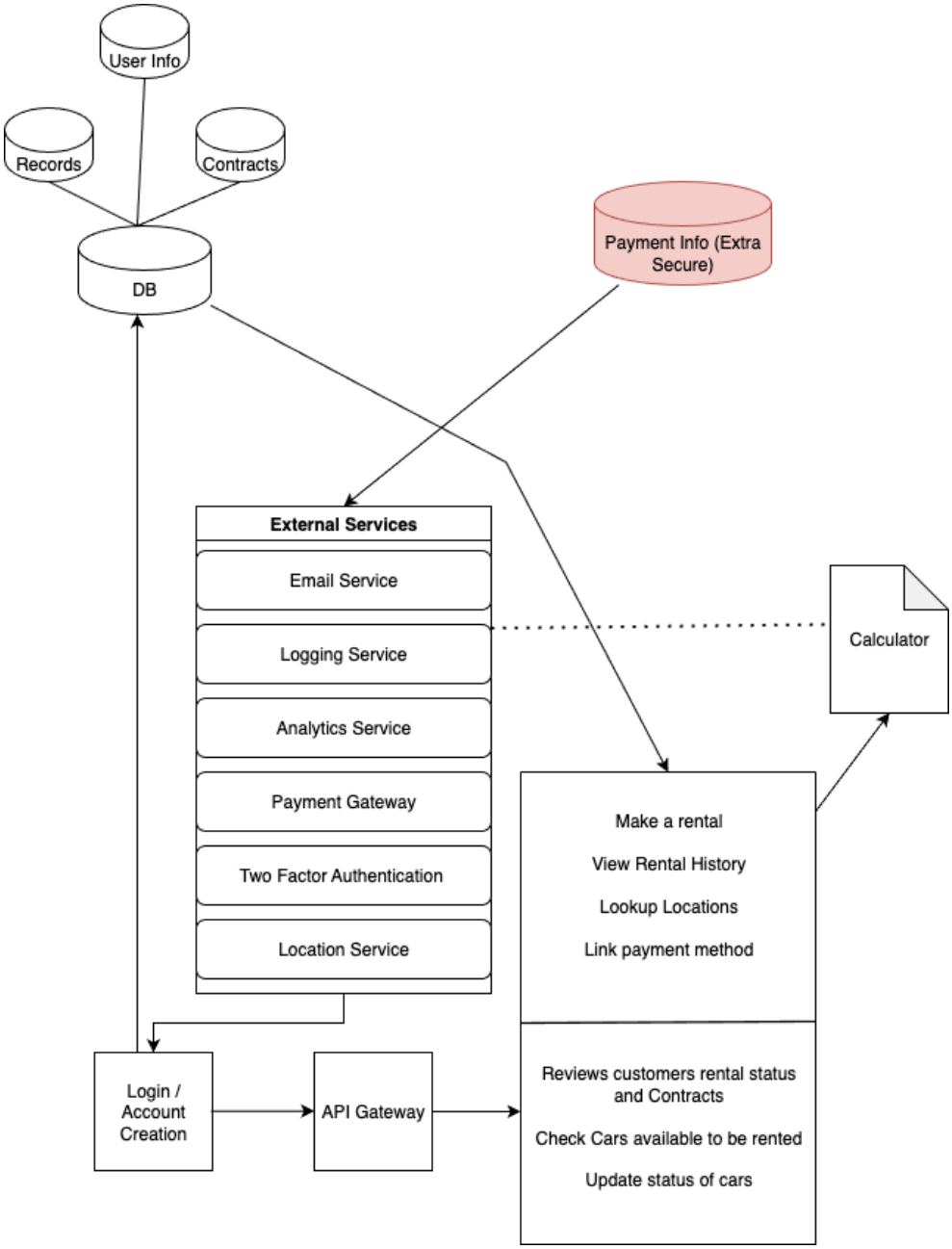
System Description

The car rental system software is aimed to be a substitute for the traditional car rental process that is made manually on paper and requires both people (contractor and client) to be present physically at the car rental office. This software will be efficient as contracts can be signed from anywhere with a connection to the network, with no need to visit the rental place.

The software is accessible either from a mobile application or a website. This means it has to be compatible with the newest version of the host OS. In addition, location services are required. This software will also require credentials from the user to rent a car, and will be as intuitive as possible to avoid users reaching a deadlock. For instance, it will keep a basic login/register process with an option of two-step authentication activation. Users can enter their payment information, which will be kept secure from other information. Employees can check rental history, view car status, and update car status if needed. Security is key, which is why the software will be enforced with security methods that assure the protection of all the client's data. Finally, employees will also have access to the software to monitor the requests and manage the system.

Software Architecture Overview

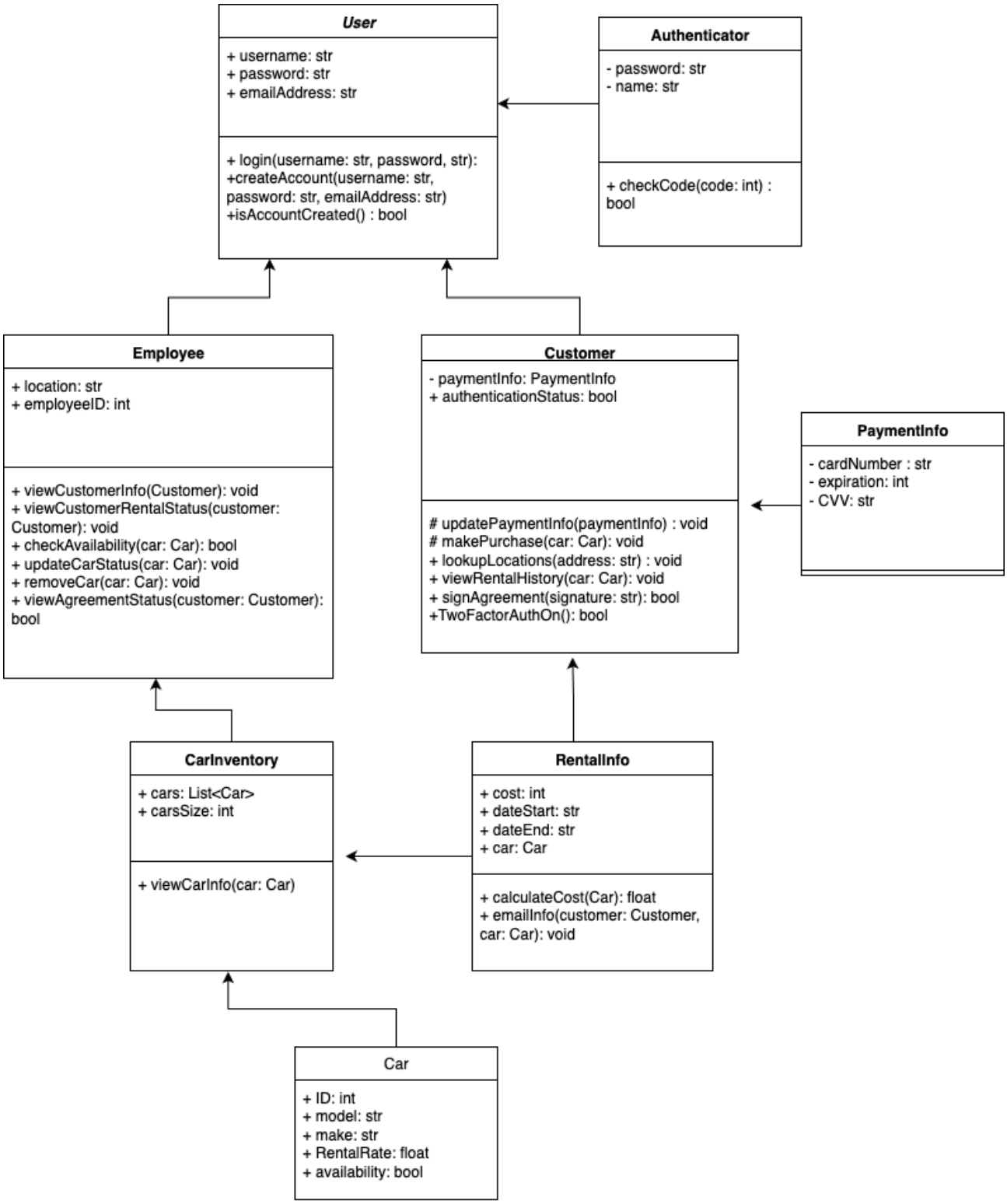
- Architectural Diagram of all major components



- SWA Description

The architectural diagram illustrates how the major components are interconnected. Starting from the Login/Account Creation, it has access to the database, which includes customer records, user information, and contracts. Such privileges and access are granted to those who are employees of BeAvis. The Login also connects to the API Gateway, which is a service that verifies users' authorization to access the software. The API Gateway connects to the services the software provides: Make a rental, view rental history, look up locations, and link payment methods. The listed services are those accessible to customers. The other services shown are only accessible by employees, such as reviewing customers' rental status and contracts, checking car availability to be rented, and updating the status of cars. All these services are then connected to the calculator, which calculates the total amount the customer will be charged for the rental car. The external services are: email service, logging service, analytics service, payment gateway, two-factor authentication, and location service. The external services are connected to Log-in/Account Creation because they'll be catered to the particular users' settings and previous activity. The payment information database is extra secured, and has access to the external services to access the payment gateway. The external services should have a calculator to determine the total cost the customer will be charged for the car rental. The database is only accessible to employees and is in a separate database to prevent the chances of a data breach. The other database, which holds three different databases, connects to the services that are accessible to customers and/or employees because it retrieves customer information from the services they use, and it is stored in the appropriate database.

- UML Diagram


User
+ username: str
+ password: str
+ emailAddress: str
+ login(username: str, password, str):
+createAccount(username: str,
password: str, emailAddress: str)
+isAccountCreated() : bool
Authenticator
- password: str
- name: str
+ checkCode(code: int) :
bool
Employee
+ location: str
+ employeeID: int
+ viewCustomerInfo(Customer): void
+ viewCustomerRentalStatus(customer:
Customer): void
+ checkAvailability(car: Car): bool
+ updateCarStatus(car: Car): void
+ removeCar(car: Car): void
+ viewAgreementStatus(customer: Customer):
bool
Customer
- paymentInfo: PaymentInfo
+ authenticationStatus: bool
# updatePaymentInfo(paymentInfo) : void
# makePurchase(car: Car): void
+ lookupLocations(address: str) : void
+ viewRentalHistory(car: Car): void
+ signAgreement(signature: str): bool
+TwoFactorAuthOn(): bool
PaymentInfo
- cardNumber : str
- expiration: int
- CVV: str
CarInventory
+ cars: List<Car>
+ carsSize: int
+ viewCarInfo(car: Car)
RentalInfo
+ cost: int
+ dateStart: str
+ dateEnd: str
+ car: Car
+ calculateCost(Car): float
+ emailInfo(customer: Customer,
car: Car): void
Car
+ ID: int
+ model: str
+ make: str
+ RentalRate: float
+ availability: bool

- **Description of classes**

- **User**: Is the main base class for all users in the system. It stores general information such as username, password, and email address. This class also manages the basic account functions, including creating an account and logging into the system.
- **Authenticator:** a class that handles the security part of the system. It is responsible for checking authentication codes and managing two-factor authentication when users choose to enable it. This class helps make the login process more secure.
- **Employee**: It is a class that represents the workers of the car rental company. It inherits from the User class, meaning it has the same basic account details but also adds employee-specific information, like their location and employee ID. Employees can view customer information, check car availability, and update car or rental statuses.
- **Customer**: represents people who rent cars from the company. It also inherits from the User class and adds extra features such as storing payment information and managing authentication status. Customers can make rentals, view their rental history, and update their payment details.
- **PaymentInfo**: stores all the customers' payment details. This includes the card number, expiration date, and CVV code. It ensures that sensitive payment data is handled and stored securely.
- **CarInventory**: class keeps track of all cars available in the system. It stores a list of car objects and the total number of cars. Employees and customers can use this class to check which cars are available for rent.

- **Car**: describes each vehicle in the system. It includes details such as the car's ID, model, make, rental rate, and whether it is available. This class helps the system identify and manage each car.
- **RentalInfo** is the last class, and it stores all the details related to a specific rental transaction. It keeps the rental cost, the start and end dates, and the car being rented. It also includes methods for calculating rental costs and emailing rental details to the customer.

● Description of attributes

The User class has three main attributes: "username", "password", and "emailAddress". These store the basic login information for any person using the system, whether they are a customer or an employee. They make it possible for users to create an account and access their personal data securely.

The Authenticator class includes attributes like "password" and "name". These are used to identify the authentication method and to verify codes during the login process. This helps improve the security of the system by allowing two-step verification.

The Employee class adds two extra attributes: "location" and "employeeID". The location represents the branch or office where the employee works, while "employeeID" is a unique number that identifies each staff member. These attributes help organize and track employees within the system.

The Customer class includes "paymentInfo" and "authenticationStatus". The "paymentInfo" attribute links to the customer's stored payment details, while "authenticationStatus" shows whether two-factor authentication is turned on. These help ensure smooth and secure transactions.

The PaymentInfo class contains the attributes “cardNumber”, expiration, and “CVV”. These store the customer’s card details safely. The system keeps this information separate from other user data to reduce the risk of security issues.

The CarInventory class has two attributes: “cars” and “carsSize”. The “cars” attribute is a list that stores all the cars in the system, and “carsSize” shows how many cars are in the inventory. These attributes make it easy to manage and update the list of vehicles.

The Car class includes several important attributes such as “ID, model, make, rentalRate, and availability”. These describe each car in detail — what brand and model it is, how much it costs per day, and whether it is currently available for rent.

Finally, the RentalInfo class has attributes like “cost”, “dateStart”, “dateEnd”, and car. These keep track of how much the rental costs, when it starts and ends, and which car is being rented. They help record and manage all rental transactions in the system.

- **Description of operations**

In the User class, there is the function login, which takes the username and password, both strings, as parameters. This function is used for employees and the manager to log into the system, and if their logins are recognized, they can access the system. The create Account function takes the parameters username, password, and email address, all strings, that allow a new user to use their information to create an account with the system. Their login information will then be stored in the system so that they do not have to create an account again. The function isAccountCreated() is used to demonstrate whether the account has been created successfully or not. All

functions in this class are public.

In the Authenticator class, there is the function called check code, which takes in an integer code as a parameter and ensures that the 2-factor authentication code that the user inputs matches the one that the system generated. This adds another layer of security to the user's account. This function is public since the Authenticator class is used by the User class.

The Employee class has a function called viewCustomerInfo(), which takes a Customer as a parameter to retrieve information about the customer, which includes their payment details and authentication status. The viewCustomerRentalStatus() that takes in Customer as a parameter is used to check whether they are renting a car or returning a car. The checkAvailability() function takes in a car as a parameter, which checks to see what cars are available to rent to a customer, ones that are not already rented. The updateCarStatus() takes in a car as a parameter and is used to mark the car as either being rented or available. The removeCar() function takes in a car as a parameter and is used to remove the car from the list of available cars to be rented out to customers. The viewAgreementStatus() function takes in the customer as a parameter, which is used to view the documents signed by the customer before renting a car. All functions are made public.

The Customer class has a protected function called updatePaymentInfo() which takes in a paymentinfo instance, which allows a customer to update their card number, expiration date, and ccv. The function makePurchase() is a protected class and takes in a car instance that the customer wishes to buy and uses their stored information to allow the customer to complete the process. lookupLocations() is a public function takes in the user's location using the

external location service and shows the user a map of all nearby car rental locations. viewRentalHistory() is a public function that takes in a car instance that the user wishes to see past uses from other customers and reviews. signAgreement() is a public function that takes in a string as the customers signature and uses it as a signature to agree to the terms and conditions of the purchase. TwoFactorAuthOn is a public function that checks if the customer has 2-factor authentication enabled before allowing them to be logged in from just their username and password.

The Car Inventory class has a public function called viewCar() that takes in a car as a parameter. This is used to check the attributes of the car, such as the ID number given to the car, model, make, the rental rate, and availability.

The RentalInfo class has a public function calculateCost() which takes in a car instance and uses that data to show the user the cost of renting that car. emailInfo() is a public function that takes in a customer instance and an instance of a car that the customer is purchasing, and emails them their confirmation details and receipt.

Development plan and timeline

Overall, the Car Rental System is used to provide an efficient and faster way for customers to rent a car, and employees to keep track of records, along with other services.

Tasks:

- Write system overview: **Jose**
    - Estimated timeline to complete: 1 day
- Create Software Architecture Diagram (SAW): **Gabe/Brianna**
    - Estimated timeline to complete: 1 day
- Write an overview of the SAW Diagram: **Brianna**
    - Estimated timeline to complete: 1 day
- Create UML Diagram: **Nathan**
    - Estimated timeline to complete: 1 day
- Write a description of classes, attributes: **Jose**
    - Estimated timeline to complete: 2 days
- Write a description of operations: **Brianna/Gabe**
    - Estimated timeline to complete: 2 days