كلية الحاسبات والذكاء الإصطناعي
Faculty of Computers & Artificial Intelligence

SWE 2 Project

Car management

Car management system using spring boot -microservices.

CONTENTS

0- Introduction

Project overview

The Car Rental Management System is designed to streamline and optimize the processes involved in managing a car rental business. By leveraging microservices architecture, the system aims to provide scalability, flexibility, and robustness, catering to the dynamic needs of the rental industry.

Project purpose

The primary purpose of this project is to develop a comprehensive solution that addresses the challenges faced by car rental companies in managing their operations efficiently. By adopting microservices architecture, the system aims to achieve the following objectives:

1. **Enhanced Scalability:** The modular nature of microservices allows for independent scaling of different components based on demand, ensuring optimal resource utilization, and accommodating fluctuations in rental activity without compromising performance.
2. **Flexibility and Customization:** By breaking down the application into smaller, loosely coupled services, the system offers greater flexibility in terms of feature enhancements, updates, and customization according to specific business requirements and evolving market trends.
3. **Fault Isolation and Resilience:** Microservices promote fault isolation, meaning that a failure in one service does not cascade to others, thereby enhancing the overall resilience of the system. This ensures uninterrupted service availability and minimizes the impact of potential failures on the business operations.
4. **Improved Performance:** With each service dedicated to a specific function, the system can be optimized for performance, allowing for faster response times and efficient resource utilization, ultimately leading to a seamless user experience for both rental administrators and customers.
5. **Scalable Architecture:** The use of microservices architecture lays the foundation for a scalable and future-proof solution that can easily accommodate growth in the volume of rental transactions, expansion into new markets, and integration with emerging technologies or third-party services.

Project Scope

1. **Core Functionality:**
   - **User Management:**
     - Authentication and authorization mechanisms for administrators, employees, and customers.
     - User profile management including registration, login, and password management.
   - **Vehicle Management:**
     - Adding, updating, and removing vehicle information including make, model, year, mileage, and condition.
     - Tracking availability, status, and location of vehicles within the fleet.
   - **Reservation and Booking:**
     - Facilitating reservation requests, confirmation, modification, and cancellation processes.
     - Real-time availability updates to prevent double bookings and optimize fleet utilization.
   - **Billing and Payment:**
     - Integration with payment gateways for secure and automated payment processing.
     - Generating invoices, receipts, and financial reports for transactions.
   - **Reporting and Analytics:**
     - Generating reports and analytics dashboards to track key performance indicators (KPIs) such as revenue, utilization rates, and customer satisfaction metrics.
     - Customizable reporting features to cater to specific business requirements.
2. **Additional Features (Optional):**
   - **Customer Relationship Management (CRM):**
     - Managing customer interactions, feedback, and support tickets.
     - Implementing loyalty programs and promotional offers.

- **Fleet Maintenance:**
    - Tracking vehicle maintenance schedules, service history, and alerts.
    - Automating maintenance reminders and scheduling service appointments.
- **Integration with Third-Party Services:**
    - Integrating with external systems such as GPS tracking, insurance providers, and roadside assistance services.
    - Ensuring compatibility with industry standards and protocols for seamless data exchange.
- **Mobile Application:**
    - Developing a mobile app for customers to browse inventory, make reservations, and manage bookings on-the-go.
    - Implementing push notifications and location-based services for enhanced user experience.

3. **Exclusions:**
    - **Vehicle Procurement:** The system will not handle the procurement or acquisition of new vehicles into the fleet.
    - **Regulatory Compliance:** Compliance with specific regional or industry regulations will be considered but not implemented as part of this scope.
    - **Hardware Infrastructure:** The project scope does not include provisioning or configuring hardware infrastructure such as servers or networking equipment.
4. **Constraints:**
    - **Technology Stack:** The system will be developed using specific technologies and frameworks as outlined in the project requirements.
    - **Timeline:** The project will adhere to the agreed-upon timeline and milestones as defined in the project plan.
    - **Budget:** Development costs and resource allocations will be within the approved project budget.
5. **Assumptions:**
    - **Data Security:** It is assumed that appropriate security measures will be implemented to protect sensitive data and prevent unauthorized access.

- **Stakeholder Involvement:** Stakeholders will be actively involved in requirements gathering, feedback sessions, and user acceptance testing (UAT) to ensure alignment with business objectives.

6. **Risks:**
   - **Technical Challenges:** Potential challenges related to integration, scalability, and performance optimization will be addressed through proactive risk management strategies.
   - **Change Management:** Any changes to project scope, requirements, or timelines will be assessed and managed through proper change control procedures.
   - **Resource Constraints:** Adequate resources including personnel, budget, and technology infrastructure will be allocated to mitigate risks associated with resource constraints.
7. **Deliverables:**
   - **Functional Requirements Document:** Detailed documentation outlining the functional specifications and features of the Car Rental Management System.
   - **Software Prototype/Minimum Viable Product (MVP):** A working prototype or MVP demonstrating key functionalities of the system for stakeholder review and validation.
   - **User Manuals and Training Materials:** Comprehensive user manuals and training materials for administrators, employees, and customers to facilitate system adoption and usage.
8. **Acceptance Criteria:**
   - **User Acceptance Testing (UAT):** The system will undergo UAT to validate that it meets the specified requirements and achieves the desired outcomes.
   - **Stakeholder Approval:** Final acceptance and approval of the system will be contingent upon stakeholder satisfaction with the delivered product.

1. System requirements

Non-functional requirements

1. **Performance**
    - The system should respond to user input within 2 seconds
    - The system should handle a minimum of 50 concurrent users
2. **Security**
    - The system should enforce password complexity and expiration policies
    - The system should encrypt sensitive data (credit card information, etc.)
3. **Usability**
    - The system should provide an intuitive user interface for easy navigation
    - The system should provide error messages and feedback for invalid user input
4. **Scalability**
    - The system should be able to handle an increase in rental agreements by 20% within 3 months
    - The system should be able to integrate with third-party services (e.g. payment gateways)

Functional requirements

Admin:

- Manage listings: Add, edit, and remove car listings.
- Access and modify system settings and configurations.
- View analytics and reports.
- Access all features and functionalities of the system.

Customer:

- Browse and search car listings.

- View detailed information about listed cars, including specifications, pricing, and photos.
- Book cars for rental or test drives.
- Leave reviews and ratings for cars they have interacted with.
- Save favorite cars and create wish-lists.
- View their booking history and manage bookings.
- Contact car owners for inquiries and negotiations.
- Make payments for bookings or purchases.
- Receive notifications about booking confirmations, updates, and promotions

Guest/User without Account:

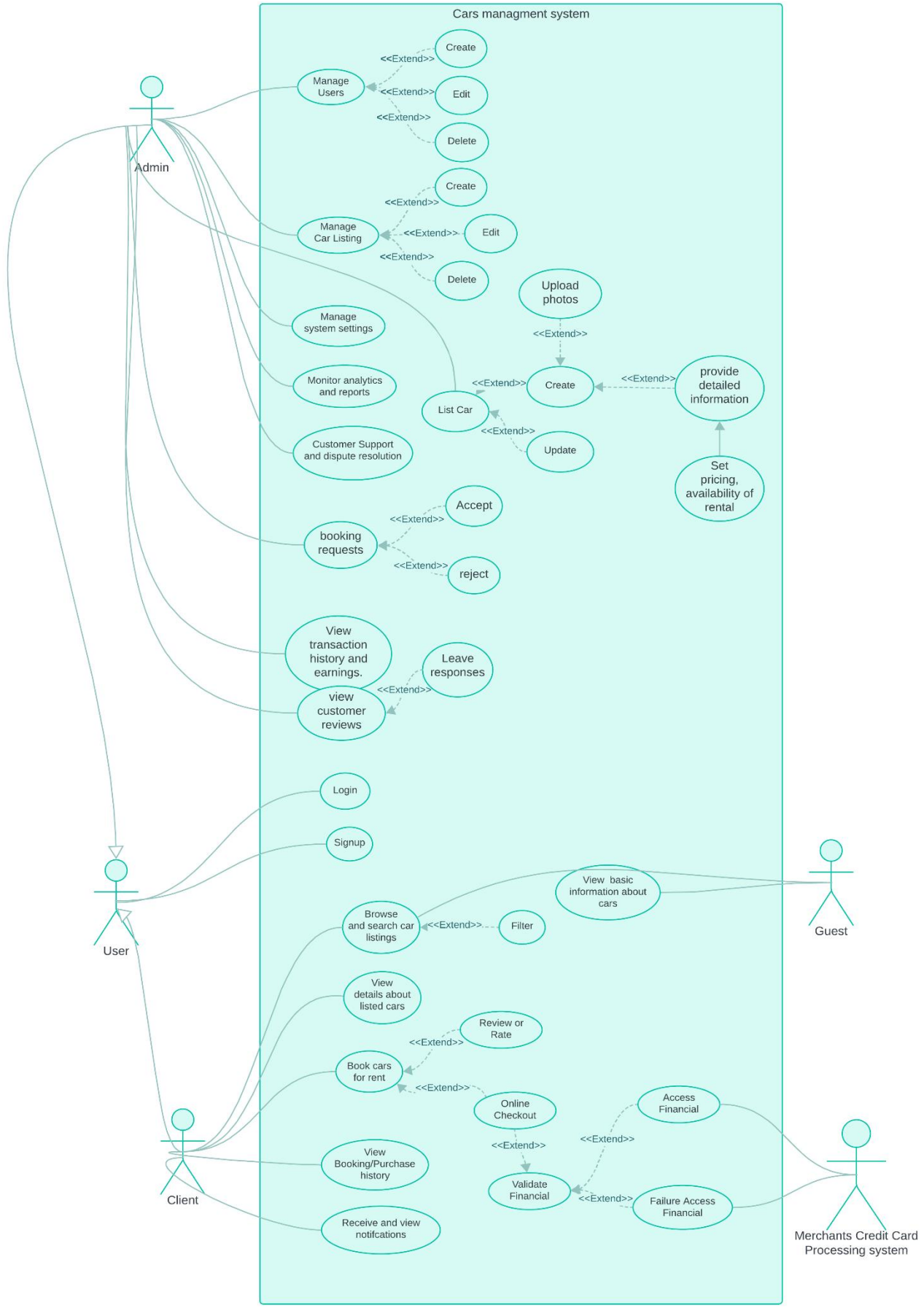
- Browse and search car listings.
- View basic information about listed cars.

Technical Architecture

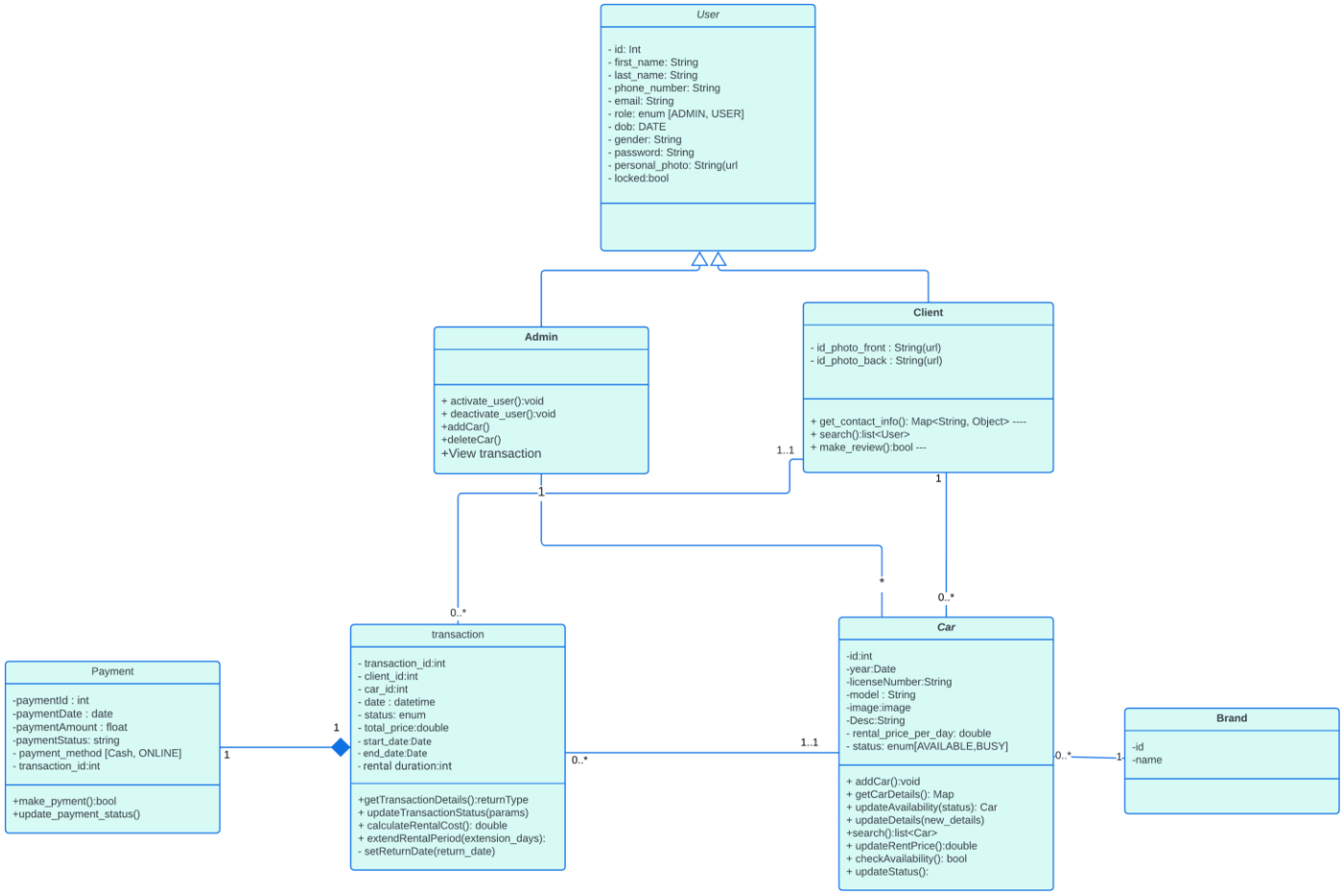
- Spring Boot Framework (JAVA)
- Micro services
- Database pgSQL/MySQL
- Ajax

Diagrams

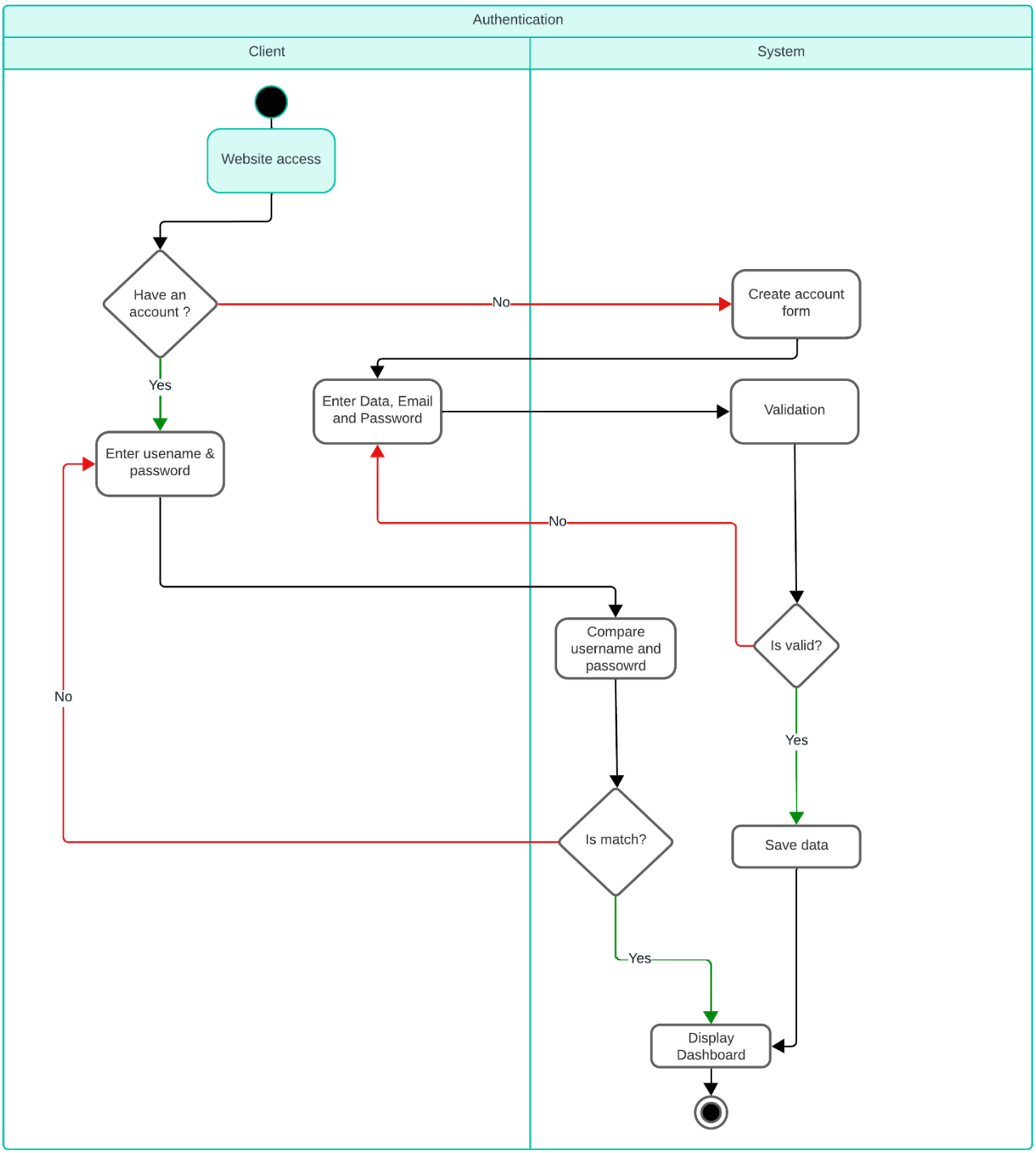
1-use case

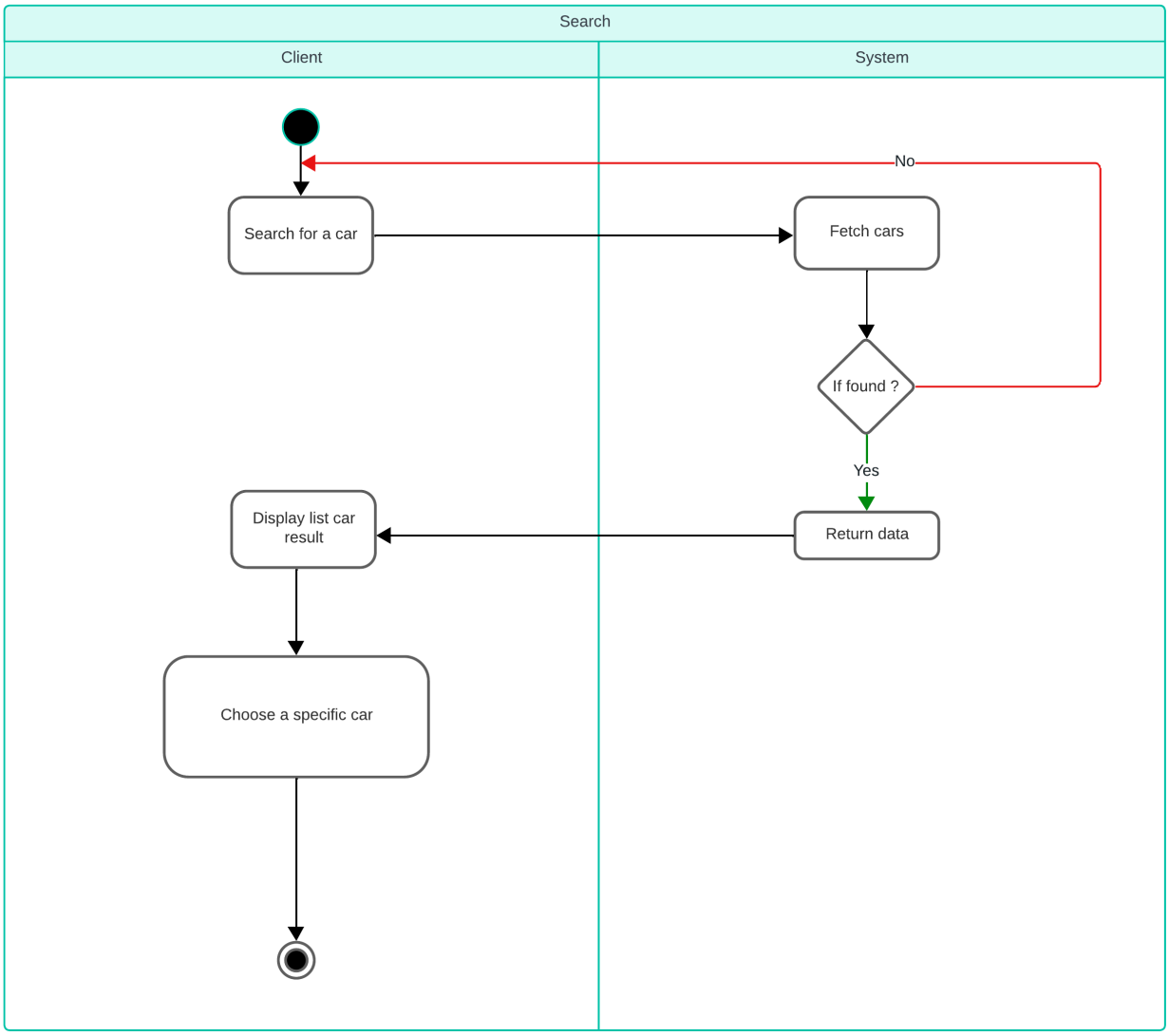

Cars managment system
Admin
Manage Users
Create
Edit
Delete
Manage Car Listing
Create
Edit
Delete
Manage system settings
Monitor analytics and reports
Customer Support and dispute resolution
List Car
Create
Upload photos
provide detailed information
Update
Set pricing, availability of rental
booking requests
Accept
reject
View transaction history and earnings.
view customer reviews
Leave responses
Login
Signup
User
Guest
View basic information about cars
Browse and search car listings
Filter
View details about listed cars
Review or Rate
Book cars for rent
Online Checkout
Client
View Booking/Purchase history
Receive and view notifcations
Access Financial
Validate Financial
Failure Access Financial
Merchants Credit Card Processing system
<<Extend>>

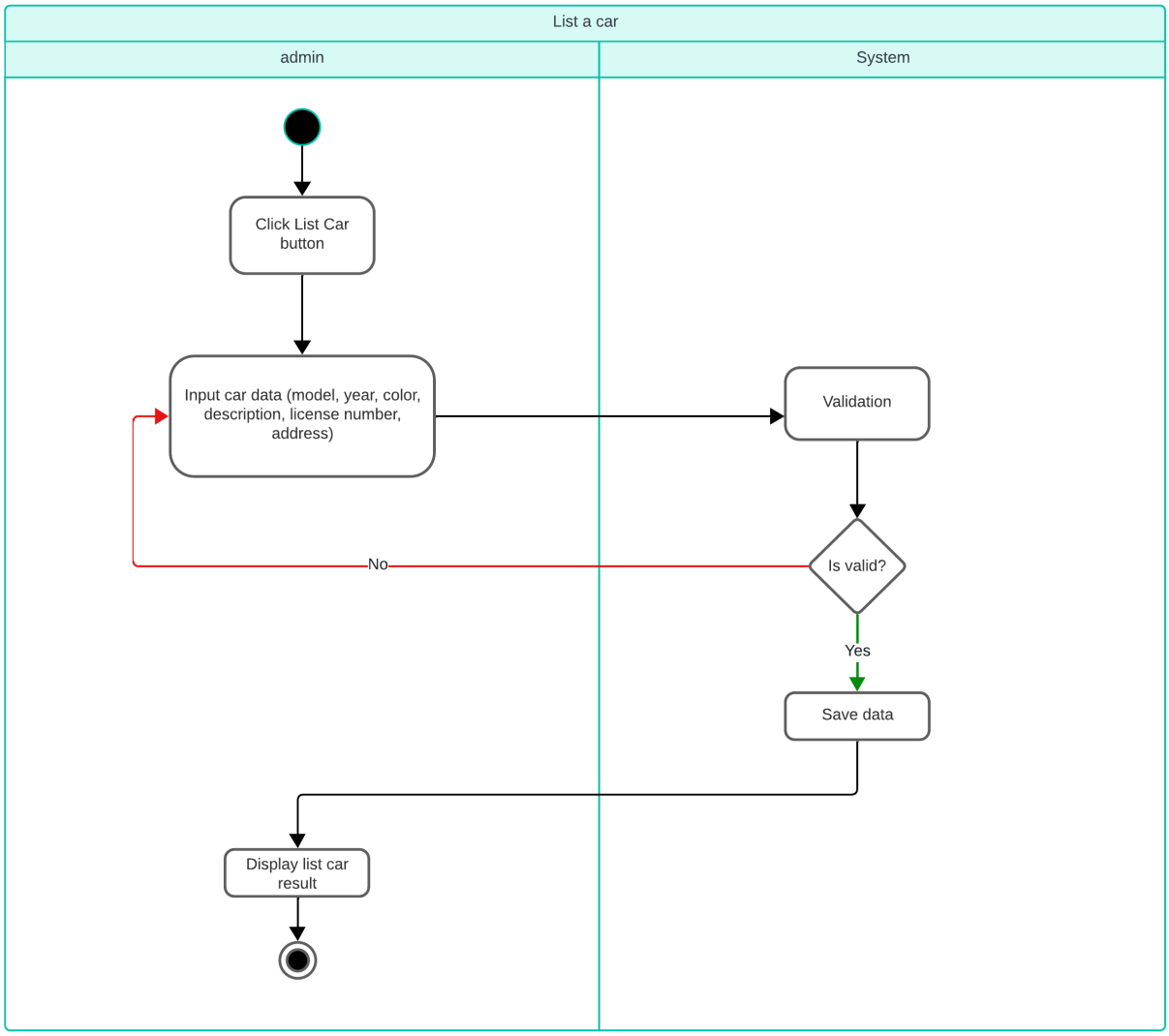
2-Class

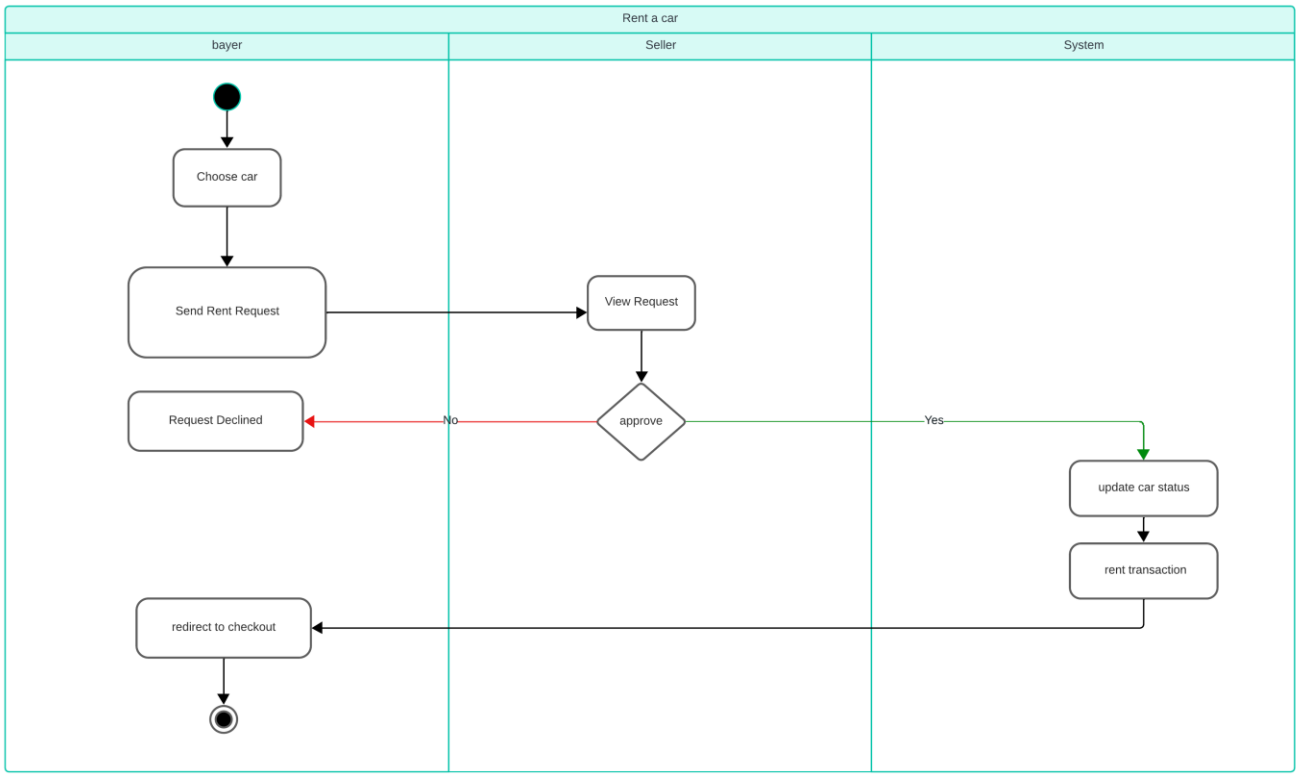

User
- id: Int
- first_name: String
- last_name: String
- phone_number: String
- email: String
- role: enum [ADMIN, USER]
- dob: DATE
- gender: String
- password: String
- personal_photo: String(url
- locked:bool
Admin
+ activate_user():void
+ deactivate_user():void
+addCar()
+deleteCar()
+View transaction
Client
- id_photo_front : String(url)
- id_photo_back : String(url)
+ get_contact_info(): Map<String, Object> ----
+ search():list<User>
+ make_review():bool ---
1..1
1
1
0..*
*
0..*
Payment
-paymentId : int
-paymentDate : date
-paymentAmount : float
-paymentStatus: string
- payment_method [Cash, ONLINE]
- transaction_id:int
+make_pyment():bool
+update_payment_status()
1
1
transaction
- transaction_id:int
- client_id:int
- car_id:int
- date : datetime
- status: enum
- total_price:double
- start_date:Date
- end_date:Date
- rental duration:int
+getTransactionDetails():returnType
+ updateTransactionStatus(params)
+ calculateRentalCost(): double
+ extendRentalPeriod(extension_days):
- setReturnDate(return_date)
0..*
1..1
Car
-id:int
-year:Date
-licenseNumber:String
-model : String
-image:image
-Desc:String
- rental_price_per_day: double
- status: enum[AVAILABLE,BUSY]
+ addCar():void
+ getCarDetails(): Map
+ updateAvailability(status): Car
+ updateDetails(new_details)
+search():list<Car>
+ updateRentPrice():double
+ checkAvailability(): bool
+ updateStatus():
0..*
1
Brand
-id
-name

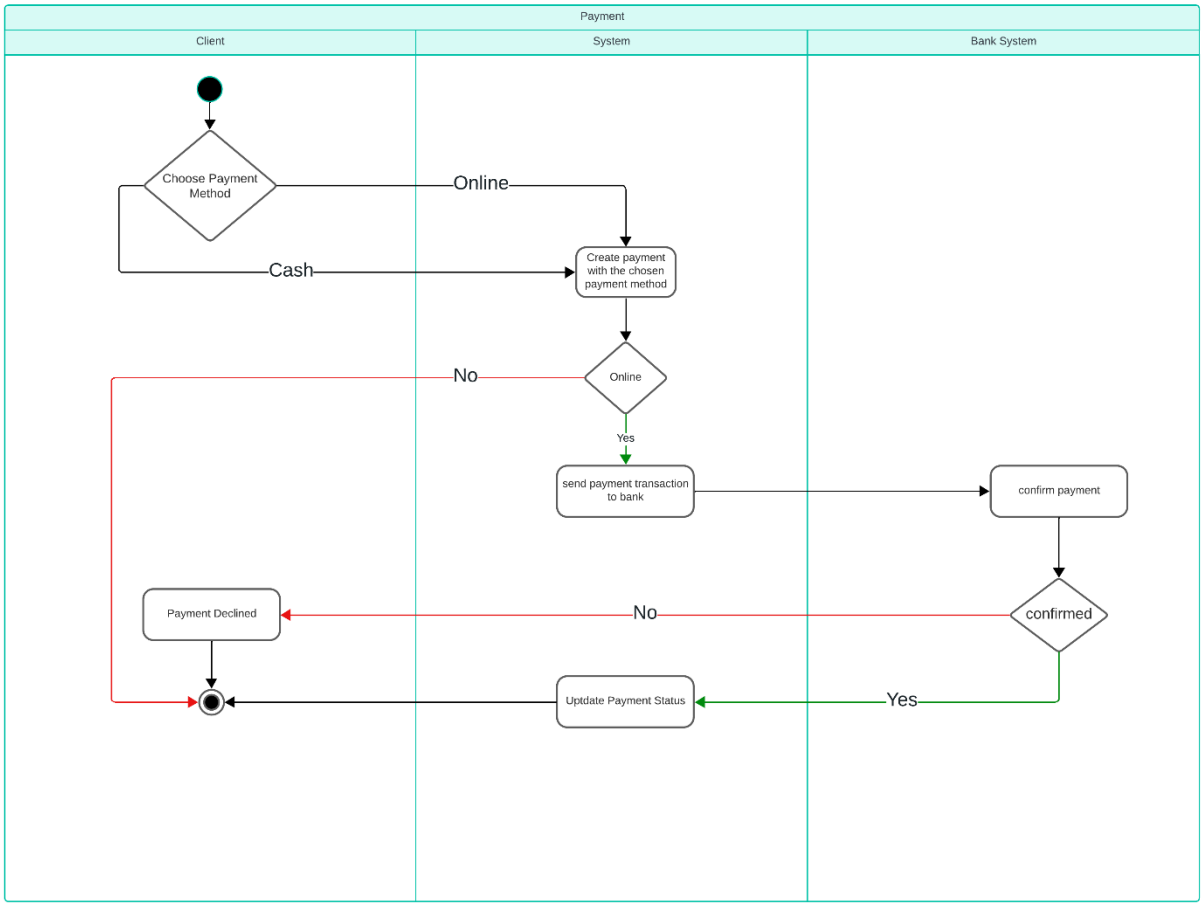
3-Activity


Authentication
Client
System
Website access
Have an account ?
No
Create account form
Yes
Enter Data, Email and Password
Validation
Enter usename & password
No
Compare username and passowrd
Is valid?
No
Yes
Is match?
Save data
Yes
Display Dashboard

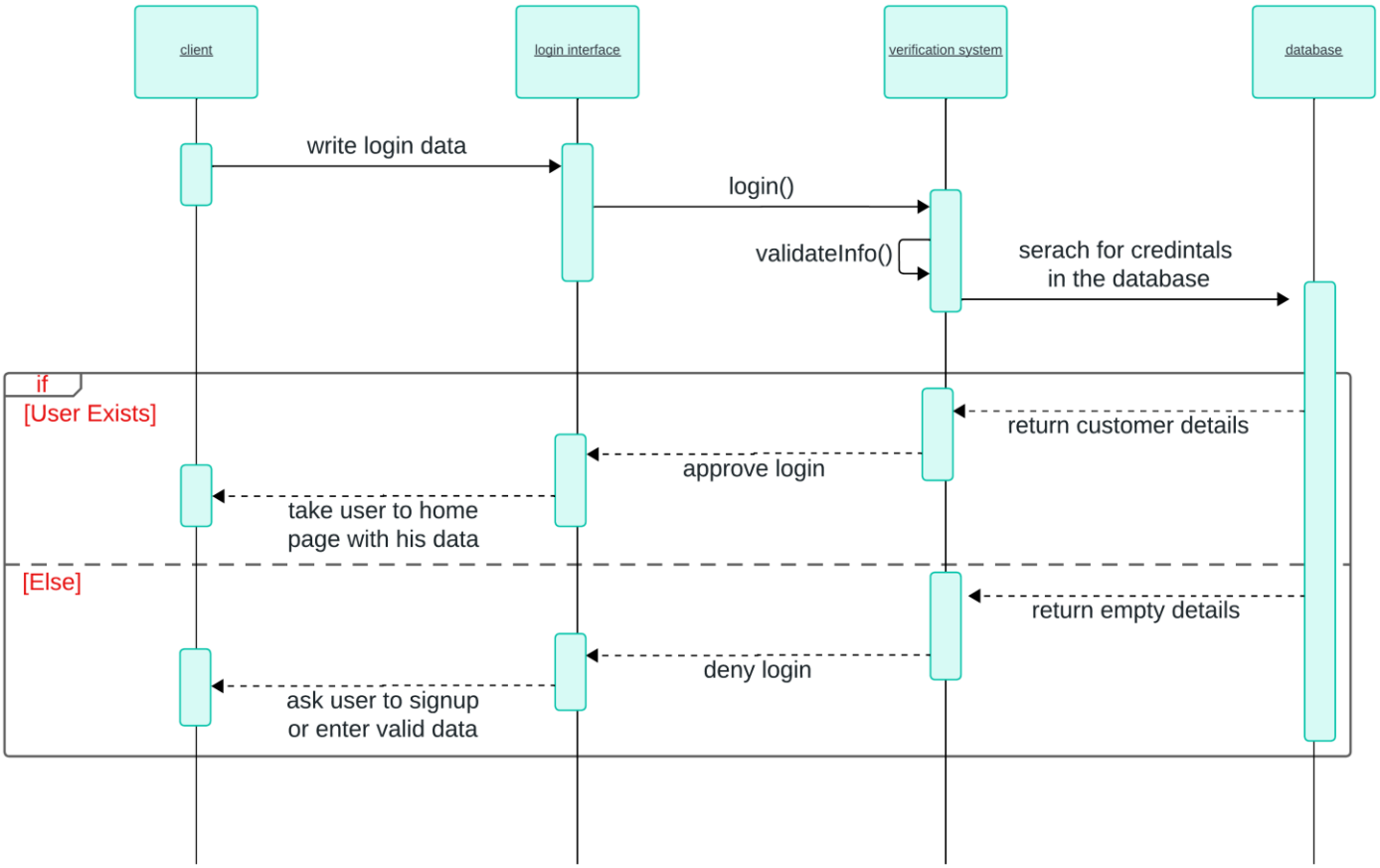

Search
Client
System
Search for a car
Fetch cars
No
If found ?
Yes
Return data
Display list car result
Choose a specific car

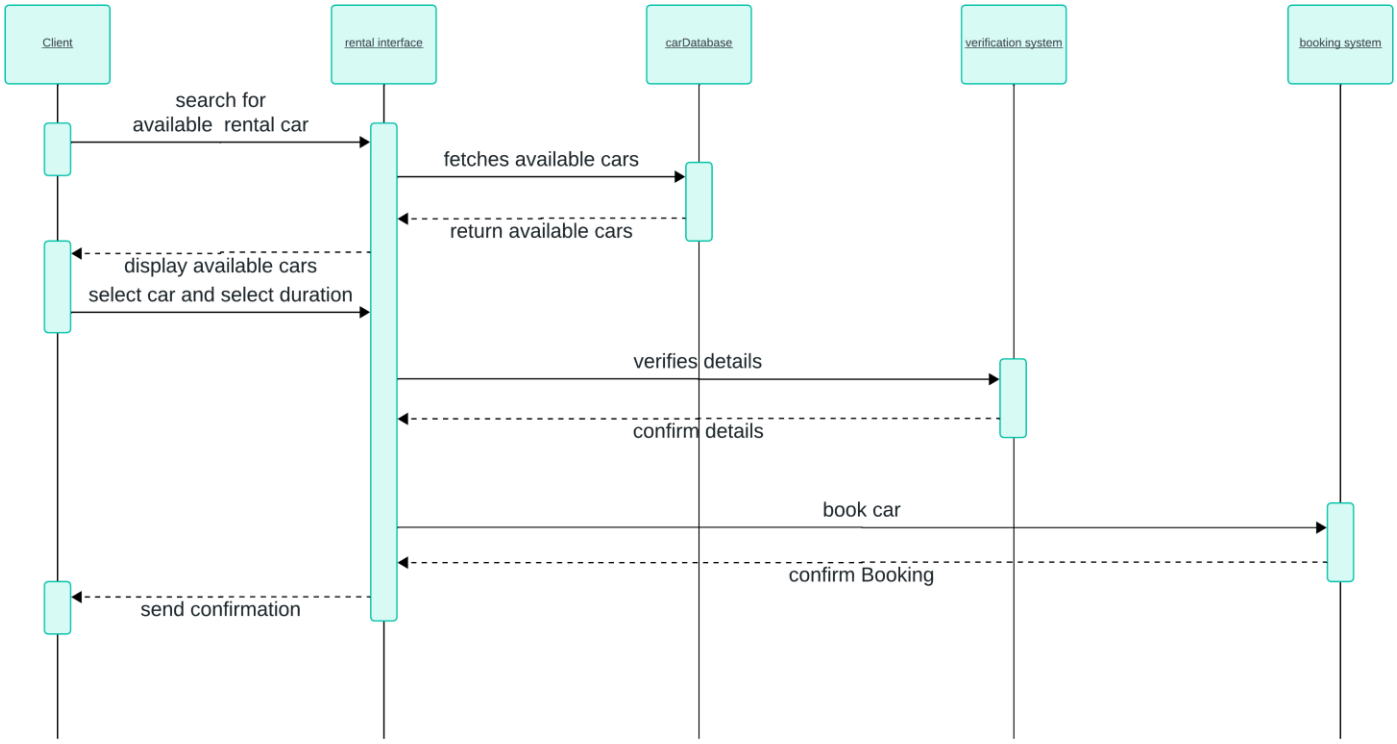

List a car
admin
System
Click List Car button
Input car data (model, year, color, description, license number, address)
Validation
Is valid?
No
Yes
Save data
Display list car result


Rent a car
bayer
Seller
System
Choose car
Send Rent Request
View Request
Request Declined
No
approve
Yes
update car status
rent transaction
redirect to checkout

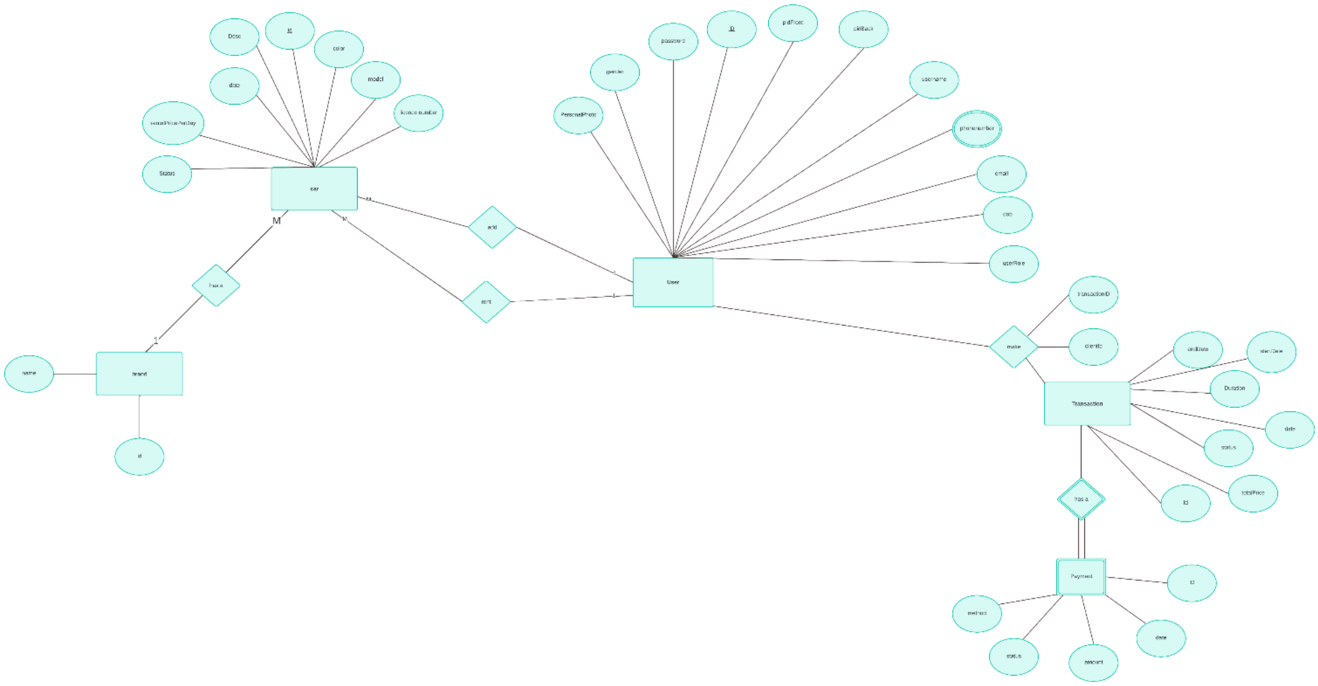

Payment
Client
System
Bank System
Choose Payment Method
Online
Cash
Create payment with the chosen payment method
Online
No
Yes
send payment transaction to bank
confirm payment
confirmed
Payment Declined
No
Yes
Uptdate Payment Status

4- sequence


client
login interface
verification system
database
write login data
login()
validateInfo()
serach for credintals
in the database
if
[User Exists]
return customer details
approve login
take user to home
page with his data
[Else]
return empty details
deny login
ask user to signup
or enter valid data

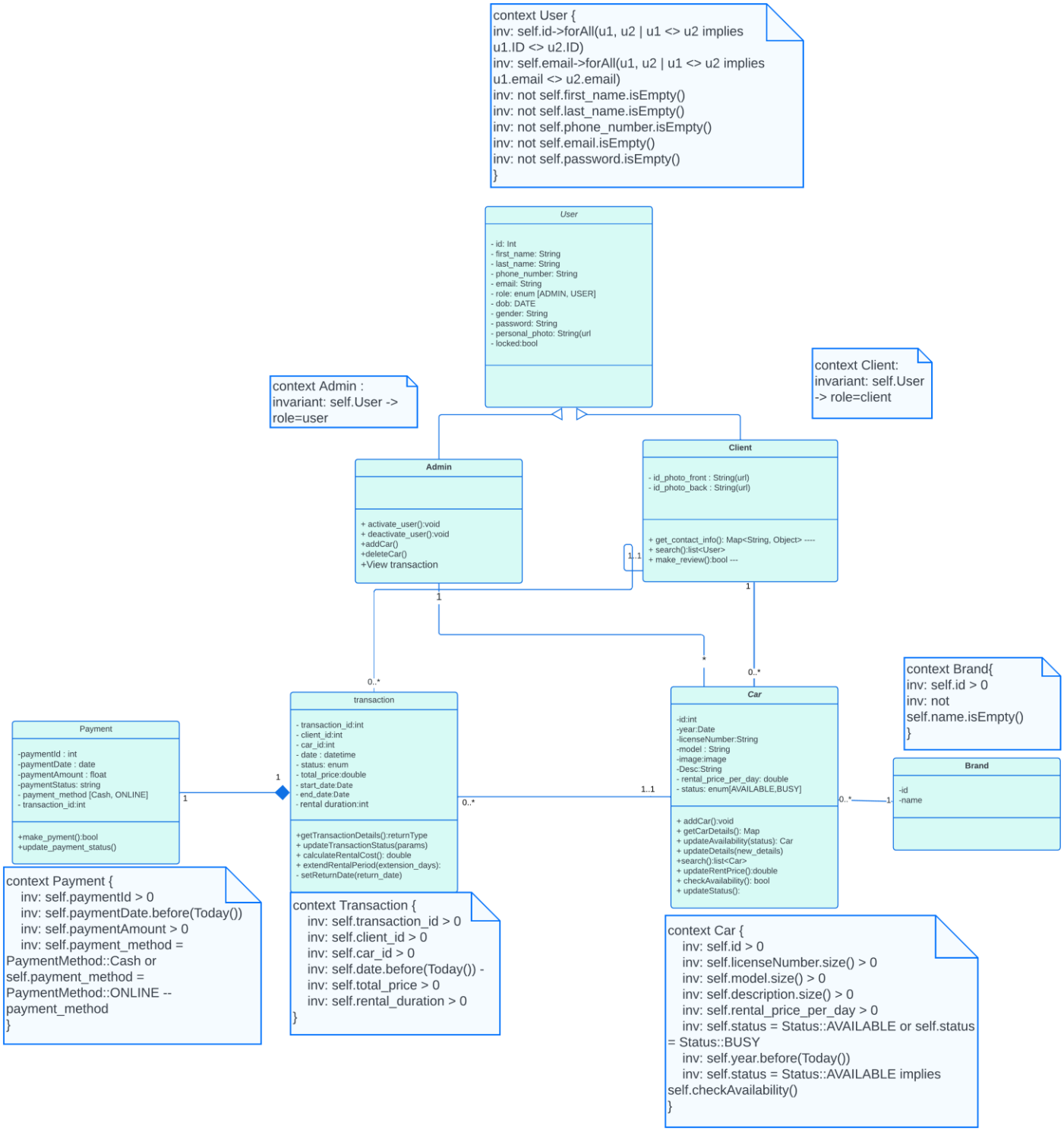

Client
rental interface
carDatabase
verification system
booking system
search for
available rental car
fetches available cars
return available cars
display available cars
select car and select duration
verifies details
confirm details
book car
confirm Booking
send confirmation

5-Erd



Desc
id
color
date
model
rentalPricePerDay
licence number
Status
car
M
M
M
have
1
name
brand
id
add
rent
1
1
PersonalPhoto
gender
password
ID
pidFront
pidBack
username
phonenumber
email
dob
userRole
User
make
transactionID
clientId
endDate
Duration
Transaction
date
status
totalPrice
Id
has a
Payment
id
method
status
amount
date

OCL


context User {
inv: self.id->forAll(u1, u2 | u1 <> u2 implies u1.ID <> u2.ID)
inv: self.email->forAll(u1, u2 | u1 <> u2 implies u1.email <> u2.email)
inv: not self.first_name.isEmpty()
inv: not self.last_name.isEmpty()
inv: not self.phone_number.isEmpty()
inv: not self.email.isEmpty()
inv: not self.password.isEmpty()
}
User
- id: Int
- first_name: String
- last_name: String
- phone_number: String
- email: String
- role: enum [ADMIN, USER]
- dob: DATE
- gender: String
- password: String
- personal_photo: String(url
- locked:bool
context Admin : invariant: self.User -> role=user
context Client: invariant: self.User -> role=client
Admin
+ activate_user():void
+ deactivate_user():void
+addCar()
+deleteCar()
+View transaction
Client
- id_photo_front : String(url)
- id_photo_back : String(url)
+ get_contact_info(): Map<String, Object> ----
+ search():list<User>
+ make_review():bool ---
1..1
1
1
*
0..*
0..*
context Brand{
inv: self.id > 0
inv: not self.name.isEmpty()
}
Payment
-paymentId : int
-paymentDate : date
-paymentAmount : float
-paymentStatus: string
- payment_method [Cash, ONLINE]
- transaction_id:int
+make_pyment():bool
+update_payment_status()
1
1
transaction
- transaction_id:int
- client_id:int
- car_id:int
- date : datetime
- status: enum
- total_price:double
- start_date:Date
- end_date:Date
- rental duration:int
+getTransactionDetails():returnType
+ updateTransactionStatus(params)
+ calculateRentalCost(): double
+ extendRentalPeriod(extension_days):
- setReturnDate(return_date)
0..*
1..1
Car
-id:int
-year:Date
-licenseNumber:String
-model : String
-image:image
-Desc:String
- rental_price_per_day: double
- status: enum[AVAILABLE,BUSY]
+ addCar():void
+ getCarDetails(): Map
+ updateAvailability(status): Car
+ updateDetails(new_details)
+search():list<Car>
+ updateRentPrice():double
+ checkAvailability(): bool
+ updateStatus():
0..*
1
Brand
-id
-name
context Payment {
inv: self.paymentId > 0
inv: self.paymentDate.before(Today())
inv: self.paymentAmount > 0
inv: self.payment_method = PaymentMethod::Cash or self.payment_method = PaymentMethod::ONLINE -- payment_method
}
context Transaction {
inv: self.transaction_id > 0
inv: self.client_id > 0
inv: self.car_id > 0
inv: self.date.before(Today()) -
inv: self.total_price > 0
inv: self.rental_duration > 0
}
context Car {
inv: self.id > 0
inv: self.licenseNumber.size() > 0
inv: self.model.size() > 0
inv: self.description.size() > 0
inv: self.rental_price_per_day > 0
inv: self.status = Status::AVAILABLE or self.status = Status::BUSY
inv: self.year.before(Today())
inv: self.status = Status::AVAILABLE implies self.checkAvailability()
}

Team Members:

ID	NAME
20210715	مازن اسلام صبحى
20210694	كيرلس سامى عطا الله
20210695	كيرلس سليمان وهيب
20210712	مارينا عطاس
20210669	فام فايز
20210710	مارينا ايمن حنا
20210679	فيلوباتير عزت رزق الله

Code Repository:

https://github.com/kerolus77/car_module