# Requirement Specification & Design Document for Vehicle Rental System

## I. Functional Requirements

Functional requirements describe the specific behaviors or functions that the system must perform. Based on the Use Case diagram you provided, the system has the following main functional requirements:

- Account Management
    + Registration: The Customer must be able to create a new account
    + Login: Both the Customer and the Admin must be able to log into the system
- Customer Functions
    + View Vehicles: The Customer can view the list of available vehicles for rent
    + View Vehicle Details: The Customer can view detailed information for a specific vehicle
    + Rent Vehicle: The Customer can perform the function of renting a vehicle process must include the payment step
    + Cancel Booking: The Customer can cancel a booking they have made
    + View Booking History: The Customer can review their history of vehicle bookings
- Administrator Functions
    + Manage Vehicles: The Admin has permissions to manage the vehicle list , including adding new vehicles, updating vehicle information, and removing vehicles from the system.
    + Manage Users: The Admin has permissions to manage user accounts , viewing the user list, locking or unlocking accounts.

## II. Non-Functional Requirements

- Usability:

    + The user interface must be friendly, clear, and easy to use.
    + Key functions (like renting a vehicle, searching for vehicles) must be easy to find.
- Performance:

    + Page load time and vehicle list display must be fast (e.g., under 3 seconds).
    + The system must process payment transactions quickly.

- Reliability:

    + The system must operate stably.
    + Booking data must be stored accurately, without loss or corruption.

- Security:

    + User passwords (both Customer and Admin) must be stored in a hashed format.
    + A Customer can only view their own booking history, not that of others.

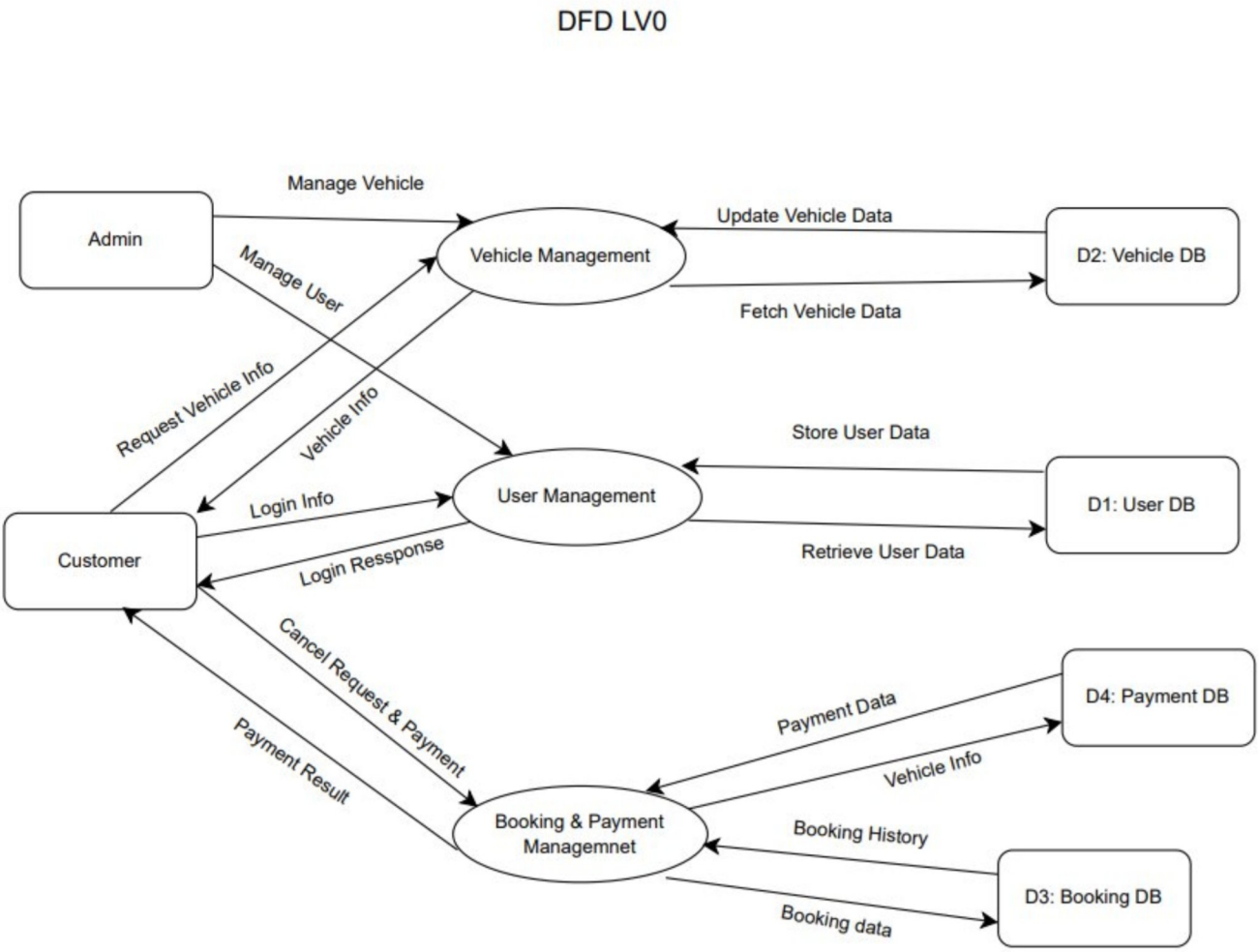
    + The Admin must be logged in to use management functions.

- Design Constraints :

    + Developed on Console Application platform. Programming language: Python. Source code management: GitHub. Packaging: Docker.

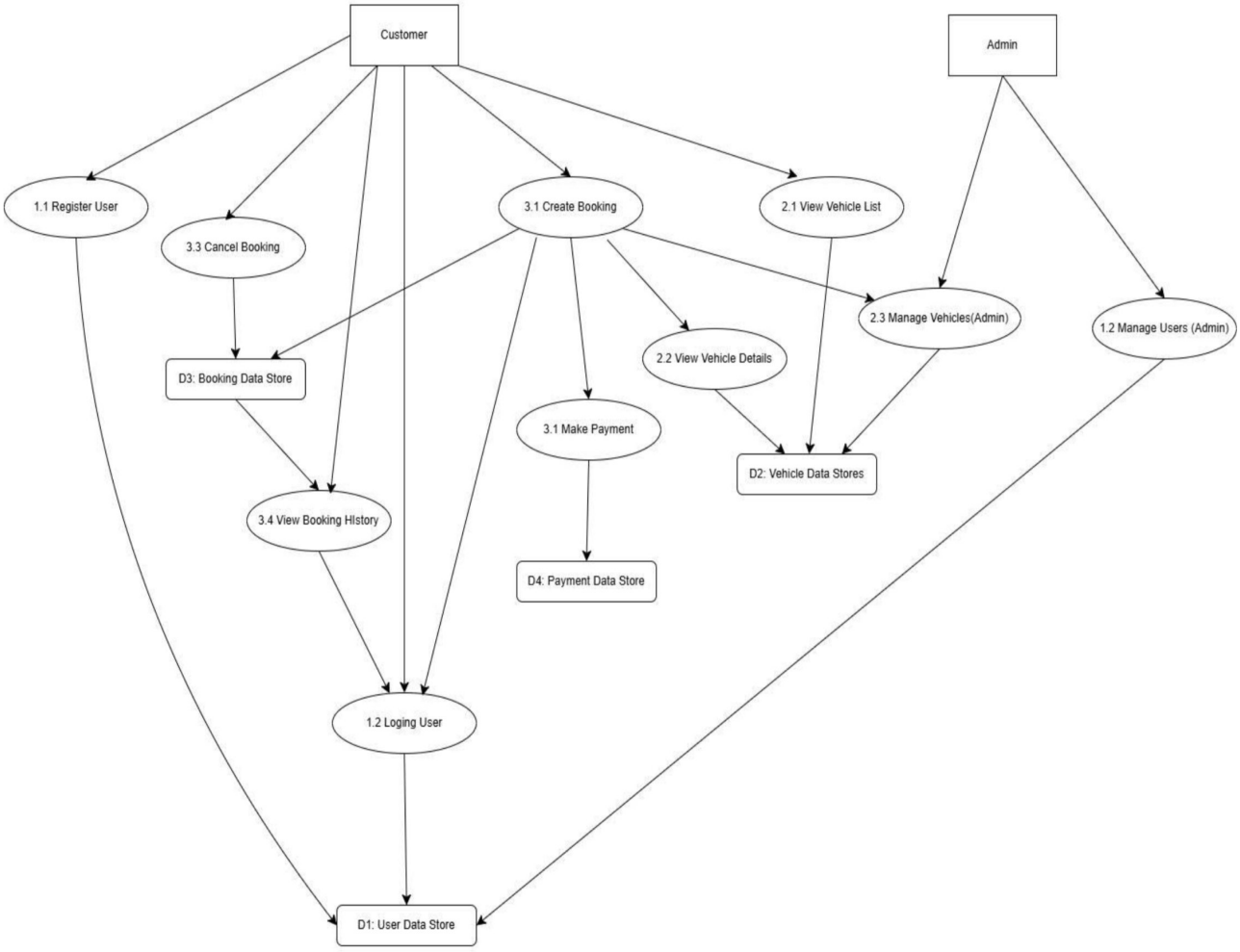
## III.Data Flow Diagram

DFD Level 0: Provides a high-level overview of the vehicle rental system, identifying the two external entities, Customer and Admin, interacting with three main business processes (User Management, Vehicle Management, Booking & Payment Management) and their four associated data stores.

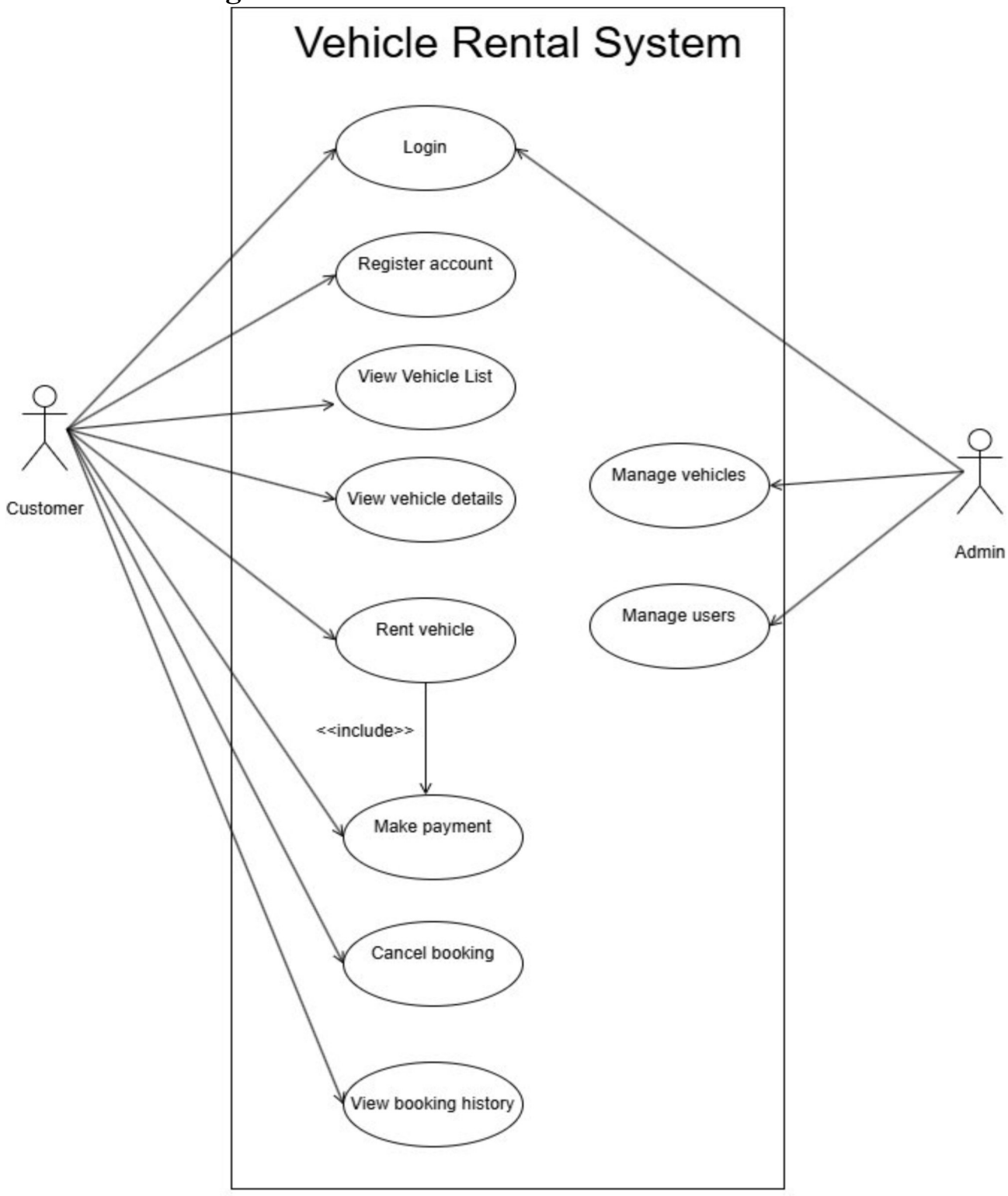



DFD Level 1: Decomposes the three main processes from Level 0 into detailed sub-functions (such as Register User, Create Booking, and Manage Vehicles (Admin)) and specifies the data flow for each function as it reads from or writes to the specific data stores (D1, D2, D3, D4).

DFD LV1


Customer
Admin
1.1 Register User
3.3 Cancel Booking
3.1 Create Booking
2.1 View Vehicle List
2.3 Manage Vehicles(Admin)
1.2 Manage Users (Admin)
D3: Booking Data Store
2.2 View Vehicle Details
3.1 Make Payment
D2: Vehicle Data Stores
3.4 View Booking HIstory
D4: Payment Data Store
1.2 Loging User
D1: User Data Store

## III.Use Case Diagram



# 1. System Overview Diagram

## 1.1. List of Actors

| No. | Actor | Meaning |
|---|---|---|
| 1 | Customer | A customer who wants to view and rent vehicles |
| 2 | Admin | An administrator responsible for system operation. |

## 1.2 List of Use Cases

| No. | Use-case | Meaning |
|---|---|---|
| 1 | Login | Log into the system |
| 2 | Register account | Register a new account |
| 3 | View Vehicle List | View vehicle list |
| 4 | View vehicle details | View vehicle details |

# 2. Use Case Register account

## 2.1. Summary

Allows a new customer to create an account.

## 2.2. Event Flow Actor

Customer

## 2.2.1. Main Flow

1. The Customer chooses the "Register" function.
2. The System asks the Customer to input: Full Name, Email, Phone, Password, and Re-enter Password.
3. The Customer enters the information.
4. The System checks if the Password and Re-enter Password match. 5. The System checks if the Email already exists in the user data file.
6. If everything is valid, The System saves the new Customer info to the file (password is hashed).

7. The System displays the message "Registration successful." 2.2.2. Alternative Flows Email exists: In step 5, if the Email is found, The System displays "Email is already in use." and ends. Password mismatch: In step 4, if passwords don't match, The System displays "Passwords do not match." and ends.

### 2.3. Special Requirements

The password must be hashed before being saved to the data file.

### 2.4. Pre-condition

None.

## 2.5. Post-condition

If successful, a new Customer account is added to the data file.

## 2.6. Extension Points None.

### 3. Use Case Login

## 3.1. Summary

Allows a Customer or Admin to log in.

## 3.2. Event Flow Actor: Customer, Admin 3.2.1. Main Flow

1. The user chooses the "Login" function.
2. The System asks for: Email and Password.
3. The user enters the information.
4. The System searches the user file for a matching Email and (hashed) Password.
5. If a match is found, The System records the user as logged in (noting their role as Customer or Admin) and shows the Main Menu. 3.2.2. Alternative Flows Incorrect info: In step 4, if no match is found, The System displays "Incorrect Email or Password." **3.3. Special Requirements**

None.

### 3.4. Pre-condition

User is not logged in.

### 3.5. Post-condition

If successful, the user is in a "logged in" state.

### 3.6. Extension Points

None.

## 4. Use Case View Vehicle List

## 4.1. Summary

Allows the Customer to see a list of vehicles.

## 4.2. Event Flow Actor

Customer

## 4.2.1. Main Flow

1. The Customer chooses the "View Vehicle List" function.
2. The System reads the vehicle data file.
3. The System displays a numbered list of vehicles with basic info (e.g., ID, Name, Price).
4. The System allows the Customer to input a vehicle ID to see details (triggers "View vehicle details" UC). 4.2.2. Alternative Flows No vehicles: In step 2, if the file is empty, The System displays "No vehicles are available for rent." **4.3. Special Requirements**

None.

### 4.4. Pre-condition

None.

### 4.5. Post-condition

None.

### 4.6. Extension Points

None.

## 5. Use Case View vehicle details

### 5.1. Summary

Allows the Customer to see detailed information for one vehicle.

### 5.2. Event Flow Actor

Customer

### 5.2.1. Main Flow

1. Starts when the Customer inputs a vehicle ID from the "View Vehicle List" UC.
2. The System finds the vehicle with that ID in the data file.
3. The System displays all info for that vehicle (Name, Description, License Plate, Price, Status...).
4. The System asks "Do you want to rent this vehicle? (Y/N)" (to trigger "Rent vehicle" UC). 5.2.2. Alternative Flows Not found: In step 2, if the ID does not exist, The System displays "Vehicle not found."

### 5.3. Special Requirements

None.

### 5.4. Pre-condition

None.

### 5.5. Post-condition

None.

### 5.6. Extension Points

None.

## 6. Use Case Rent Vehicle

### 6.1. Summary

Allows the Customer to book a vehicle rental.

## 6.2. Event Flow Actor

Customer

### 6.2.1. Main Flow

1. Starts when the Customer selects "Y" from the "View vehicle details" UC.
2. The System asks to input: "Pickup Date" and "Return Date".
3. The System checks the booking data file to see if the vehicle has a schedule conflict during that time.
4. If no conflict, The System calculates the total cost (e.g., Num of Days * Price).
5. The System displays "Total cost is [COST]. Confirm payment? (Y/N)".
6. The Customer selects "Y".
7. «include» Make payment: The System simulates the payment and displays "Payment successful."
8. The System saves a new "booking" record to the booking data file.
9. The System displays "Booking successful!" 6.2.2. Alternative Flows Schedule conflict: In step 3, if a conflict is found, The System displays "Vehicle is already booked for this time." Payment failed: In step 7, if payment fails, The System displays "Payment failed." **6.3. Special Requirements**

None.

## 6.4. Pre-condition

The Customer must be "logged in".

## 6.5. Post-condition

If successful, a new booking is saved.

## 6.6. Extension Points

None.

# 7. Use Case Cancel Booking

## 7.1. Summary

Allows the Customer to cancel a booking.

## 7.2. Event Flow Actor

Customer

## 7.2.1. Main Flow

1. Starts when the Customer selects "Cancel booking" from the "View booking history" UC.
2. The System asks "Are you sure you want to cancel this booking? (Y/N)".
3. The Customer selects "Y".
4. The System checks the cancellation policy (e.g., must be 1 day before pickup date, booking must be in "new" state).
5. If valid, The System updates the booking's status in the file to "Cancelled".
6. The System displays "Booking successfully cancelled." 7.2.2. Alternative Flows Not eligible: In step 4, if the policy is not met (e.g., too late), The System displays "Cannot cancel, deadline has passed." **7.3. Special Requirements**

None.

### 7.4. Pre-condition

The Customer must be "logged in".

### 7.5. Post-condition

The booking's status is updated to "Cancelled".

### 7.6. Extension Points

None.

## 8. Use Case View Booking History

## 8.1. Summary

Allows the Customer to see all their past bookings.

## 8.2. Event Flow Actor

Customer

### 8.2.1. Main Flow

1. The Customer chooses "View booking history".
2. The System filters the booking data file to find all bookings for this Customer.
3. The System displays a list of their bookings (Booking ID, Vehicle Name, Date, Status...).
4. The System allows the Customer to select a booking to "Cancel" (triggers "Cancel booking" UC). 8.2.2. Alternative Flows No bookings: In step 2, if no bookings are found, The System displays "You have no booking history." **8.3. Special Requirements**

None.

## 8.4. Pre-condition

The Customer must be "logged in".

## 8.5. Post-condition

None.

## 8.6. Extension Points Use-case Cancel Order:

The user can choose to cancel a booking.

# 9. Use Case Manage Vehicles

## 9.1. Summary

Allows the Admin to Add, Edit, or Delete vehicles.

## 9.2. Event Flow Actor

Admin

### 9.2.1. Main Flow

1. The Admin chooses "Manage vehicles".

2. The System displays a menu: "(1) Add vehicle, (2) Edit vehicle, (3) Delete vehicle, (4) View all vehicles".
3. The Admin chooses a function.
4. If (1) Add: The System asks for new vehicle info (Name, Price, Desc...), then saves it to the vehicle file.
5. If (2) Edit: The System asks for the ID of the vehicle to edit, then allows updating the info.
6. If (3) Delete: The System asks for the ID of the vehicle to delete, confirms, then removes it from the file.
7. If (4) View: The System displays all vehicles (similar to "View Vehicle List" UC). 9.2.2. Alternative Flows Input error: If the Admin enters invalid info (e.g., non-existent ID for Edit/Delete), The System will display an error. **9.3. Special Requirements**

None.

### 9.4. Pre-condition

The user must be "logged in" as an Admin.

### 9.5. Post-condition

The vehicle data file is updated (added, edited, or deleted).

### 9.6. Extension Points

None.

## 10.Use Case Manage Users

## 10.1. Summary

Allows the Admin to view and Lock/Unlock Customer accounts.

## 10.2. Event Flow Actor

Admin

## 10.2.1. Main Flow

1. The Admin chooses "Manage users".
2. The System displays a list of all Customer accounts (Name, Email, Status: "Active" / "Locked").

3. The System allows the Admin to select an account to "Lock" (equivalent to "Delete" or "Disable" in the sample) or "Unlock".
4. If the Admin chooses "Lock", The System updates that user's status in the file to "Locked".
5. The System displays "Account locked successfully." 10.2.2. Alternative Flows (Kept simple, none needed for this function). **10.3. Special Requirements**

None.

### 10.4. Pre-condition

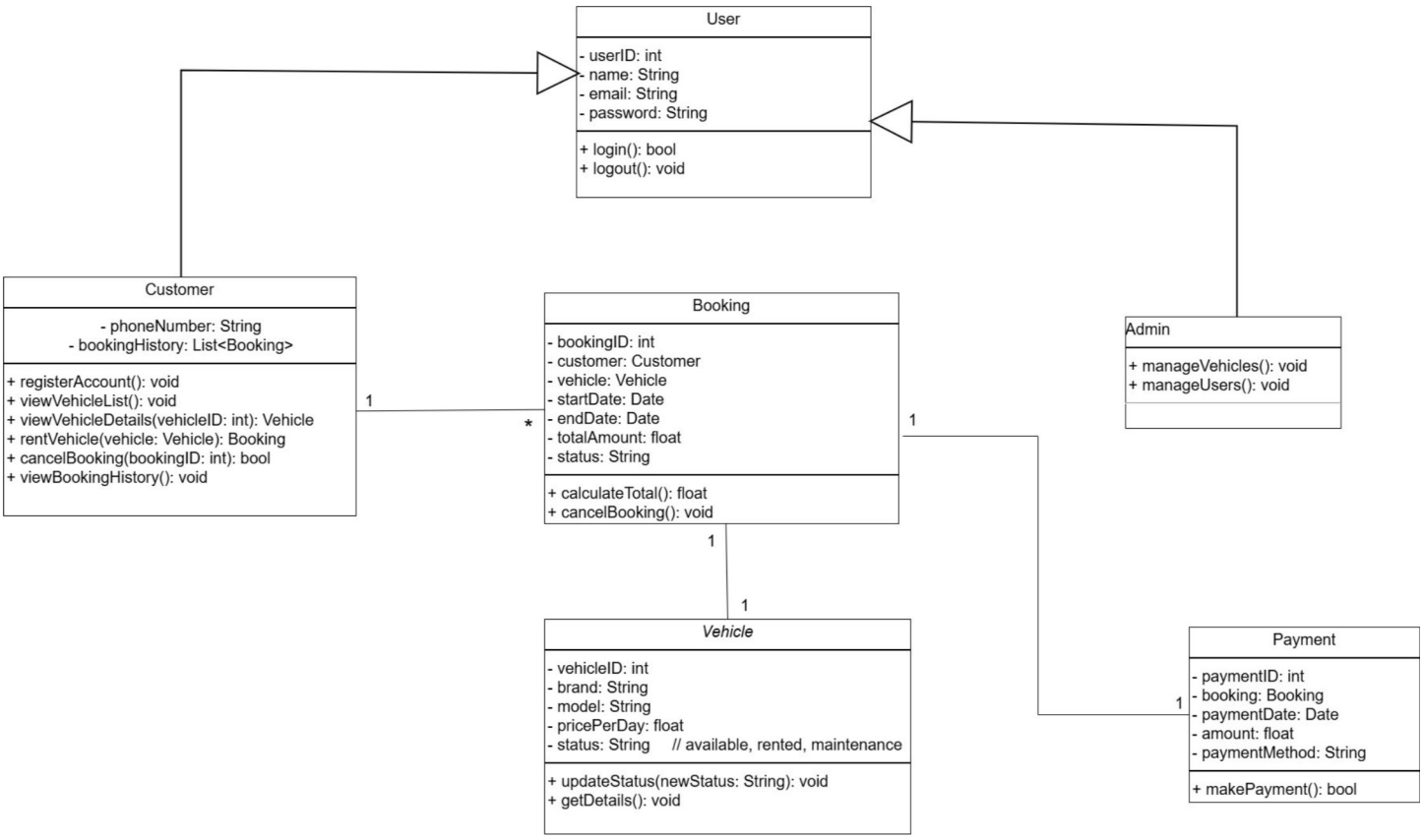
The user must be "logged in" as an Admin.

### 10.5. Post-condition

The status of a Customer account may be changed.

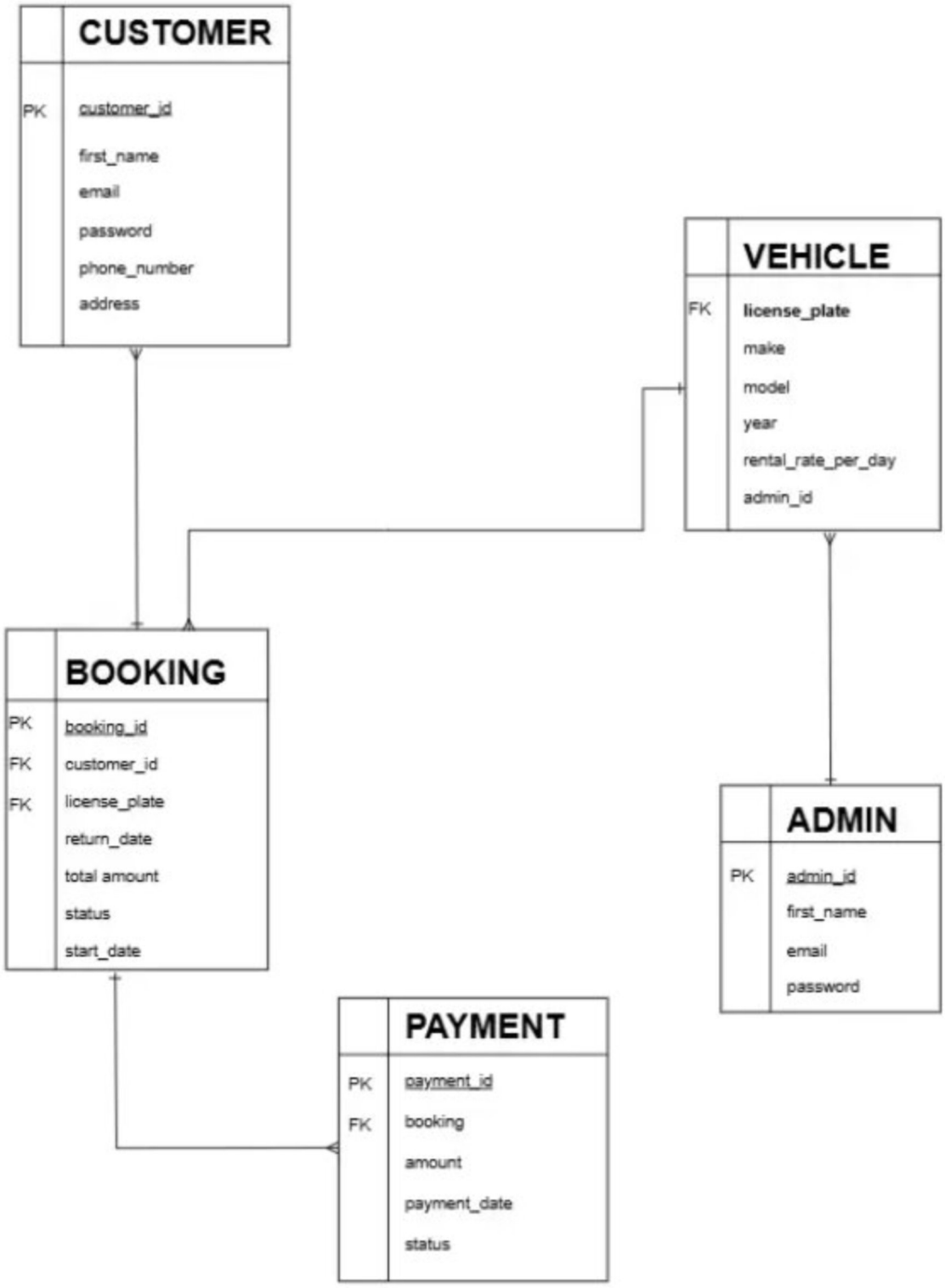
### 10.6. Extension Points None.

# IV. Class Diagram




## V. Data Model 1. Entity Relationship Diagram

The figure below shows the relationships between the main tables in the system.


CUSTOMER
PK
customer_id
first_name
email
password
phone_number
address
VEHICLE
FK
license_plate
make
model
year
rental_rate_per_day
admin_id
BOOKING
PK
booking_id
FK
customer_id
FK
license_plate
return_date
total amount
status
start_date
ADMIN
PK
admin_id
first_name
email
password
PAYMENT
PK
payment_id
FK
booking
amount
payment_date
status

## 2. Description

The system has five main tables:

Customer – stores customer information

Admin – stores admin informatio

Vehicle – stores vehicle details

Booking – stores rental orders

Payment – stores payment information

## 3. Table Summary

**Customer**

| Field | Type | Description |
|---|---|---|
| customer_id (PK) | INT | Customer ID |
| name | TEXT | Full name |
| email | TEXT | Login email |
| password | TEXT | Password (hashed) |
| phone | TEXT | Phone number |

**Admin**

| Field | Type | Description |
|---|---|---|
| admin_id (PK) | INT | Admin ID |
| name | TEXT | Admin name |
| email | TEXT | Login email |
| password | TEXT | Password (hashed) |

**Vehicle**

| | | |
|---|---|---|
| license_plate (PK) | TEXT | Vehicle plate number |
| make | TEXT | Brand |
| model | TEXT | Model name |
| price_per_day | FLOAT | Rental price per day |
| admin_id (FK) | INT | Admin who added this vehicle |

**Booking**

| Field | Type | Description |
|---|---|---|
| booking_id (PK) | INT | Booking ID |
| customer_id (FK) | INT | Customer ID |
| license_plate (FK) | TEXT | Vehicle ID |
| start_date | DATE | Rent start date |
| end_date | DATE | Rent end date |
| total | FLOAT | Total amount |
| status | TEXT | Status (Pending / Confirmed / Cancelled) |

**Payment**

| Field | Type | Description |
|---|---|---|
| payment_id (PK) | INT | Payment ID |
| booking_id (FK) | INT | Booking ID |
| amount | FLOAT | Payment amount |
| payment_date | DATE | Payment date |
| status | TEXT | Payment status |

**4. Relationships** One Customer can have many Bookings.

One Booking has one Payment.

One Admin can manage many Vehicles.

One Vehicle can appear in many Bookings.

**VI. Interface Design Description**

The program uses a simple text-based (console) interface written in Python. It is easy to use, clear, and suitable for students.

V.1. Main Menu
==== VEHICLE RENTAL SYSTEM ====
1. Login
2. Register
3. Exit
================================ Enter your choice:

V.2. Customer Menu
==== CUSTOMER MENU ====
1. View Vehicles
2. View Vehicle Details
3. Rent Vehicle
4. View Booking History
5. Cancel Booking
6. Logout
======================== Enter your choice:

V.3. Admin Menu
==== ADMIN MENU ====
1. Manage Vehicles
2. Manage Users
3. View All Bookings
4. Logout
==================== Enter your choice:

-