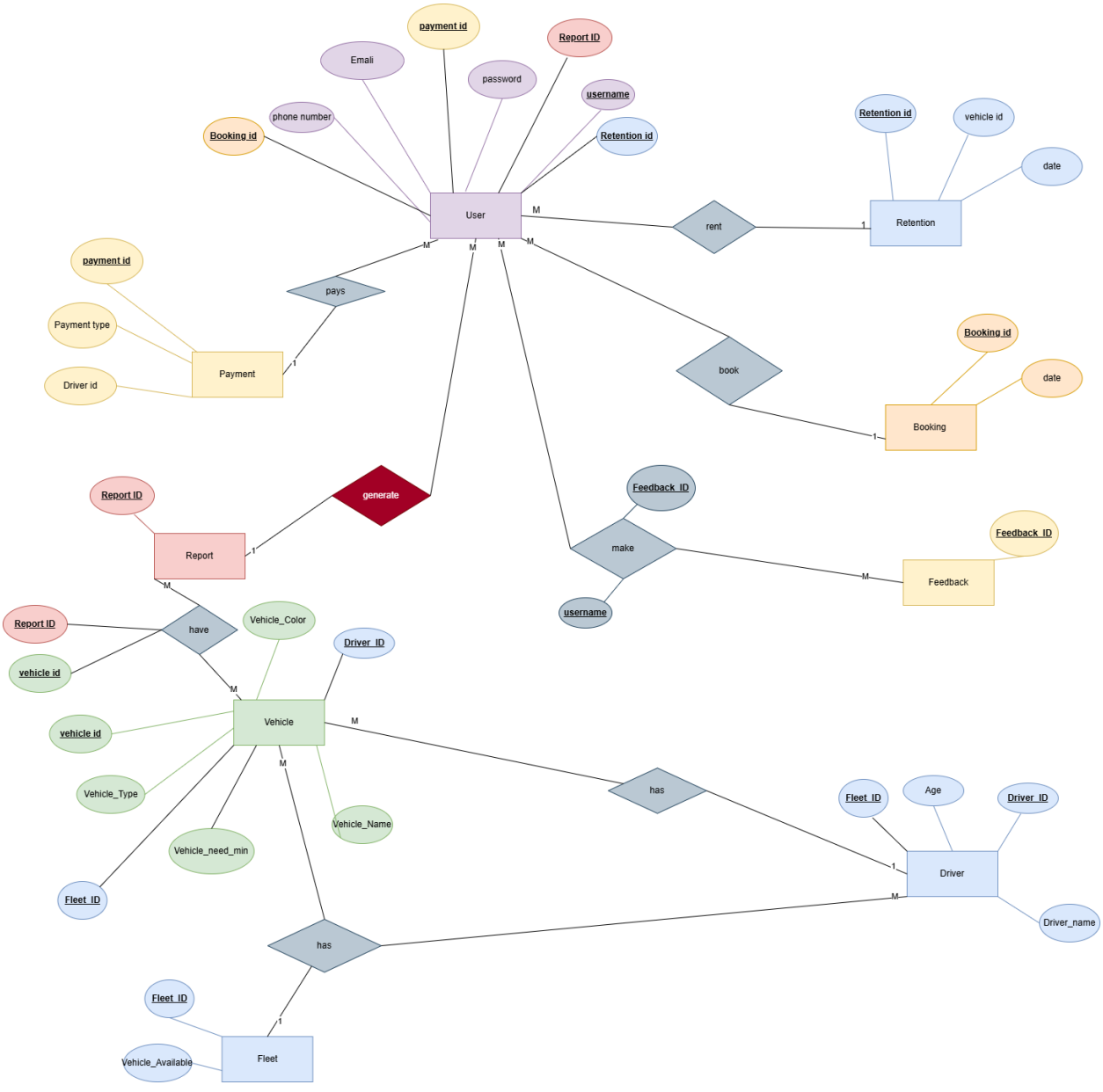
AIU.
ALAMEIN
INTERNATIONAL UNIVERSITY
جامعة العلمين الدولية

Project Design and Planning

Software Engineering CSE251

ERD Diagram


payment id
Report ID
Emali
password
username
phone number
Booking id
Retention id
Retention id
vehicle id
date
User
M
rent
1
Retention
M
M
M
M
payment id
pays
Payment type
Payment
1
Driver id
book
Booking id
date
Booking
1
Report ID
generate
Feedback_ID
Report
1
make
Feedback_ID
M
Feedback
M
username
Report ID
have
Vehicle_Color
Driver_ID
vehicle id
M
Vehicle
vehicle id
M
M
has
Fleet_ID
Age
Driver_ID
Vehicle_Type
Vehicle_Name
Vehicle_need_min
1
Driver
Fleet_ID
M
Driver_name
has
Fleet_ID
1
Vehicle_Available
Fleet

Relational Schema

User(**user_id**, email, password, phone_number, retention_id, report_id, username)

Driver(**driver_id**, driver_name, age, fleet_id)

Fleet(**fleet_id**, vehicle_available)

Vehicle(**vehicle_id**, vehicle_name, vehicle_type, vehicle_color, vehicle_need_min, fleet_id, driver_id)

Report(**report_id**, vehicle_id)

Booking(**booking_id**, date, user_id)

Payment(**payment_id**, payment_type, driver_id, user_id)

Feedback(**feedback_id**, username, user_id)

Retention(**retention_id,** vehicle_id, date)

make(**Feedback_ID, username**)

Have(**Report ID, vehicle id**)

Users

- user_id (PRIMARY KEY)
- email
- password

- phone_number
- retention_id (FOREIGN KEY references Retention.retention_id)
- report_id (FOREIGN KEY references Reports.report_id)
- username

Drivers

- driver_id (PRIMARY KEY)
- driver_name
- age
- fleet_id (FOREIGN KEY references Fleet.fleet_id)

Fleet

- fleet_id (PRIMARY KEY)
- vehicle_available

Vehicles

- vehicle_id (PRIMARY KEY)
- vehicle_name
- vehicle_type
- vehicle_color
- vehicle_need_min
- fleet_id (FOREIGN KEY references Fleet.fleet_id)
- driver_id (FOREIGN KEY references Drivers.driver_id)

Reports

- report_id (PRIMARY KEY)
- vehicle_id (FOREIGN KEY references Vehicles.vehicle_id)

Bookings

- booking_id (PRIMARY KEY)
- date
- user_id (FOREIGN KEY references Users.user_id)

Payments

- payment_id (PRIMARY KEY)
- payment_type
- driver_id (FOREIGN KEY references Drivers.driver_id)
- user_id (FOREIGN KEY references Users.user_id)

Feedback

- feedback_id (PRIMARY KEY)
- username
- user_id (FOREIGN KEY references Users.user_id)

Retention

- retention_id (PRIMARY KEY)

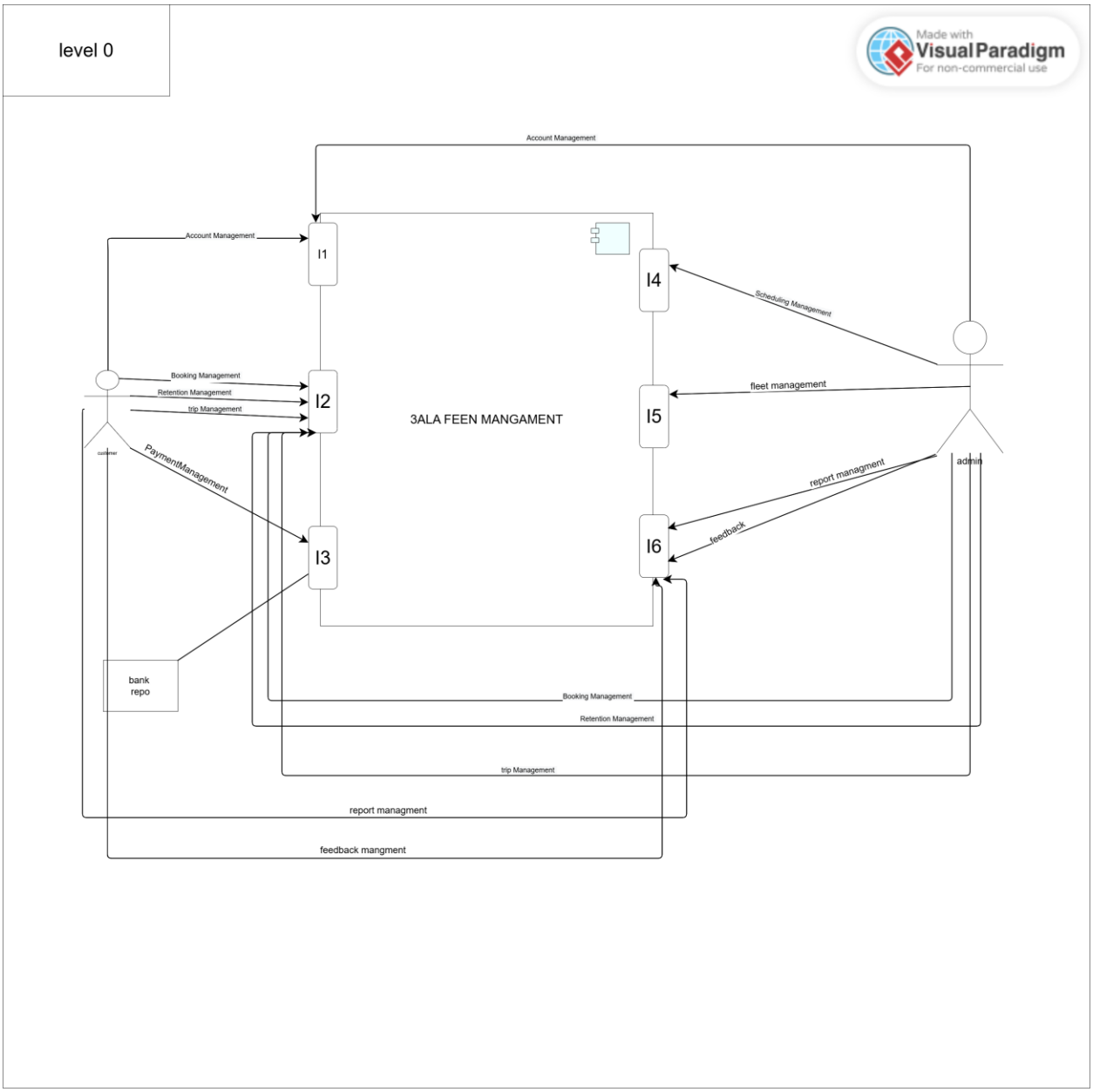
- vehicle_id (FOREIGN KEY references Vehicles.vehicle_id)

- date

Relationships:

- **User (One)** "Pays" -> **Payment (Many)**
- **User (One)** "Generates" -> **Report (Many)**
- **User (One)** "Makes" -> **Feedback (Many)**
- **User (One)** "Rents" -> **Retention (Many)**
- **User (One)** "Books" -> **Booking (Many)**
- **Payment (One)** "Includes" -> **Payment Type (One)**
- **Payment (One)** "Associates with" -> **Driver (One)**
- **Retention (Many)** "Relates to" -> **Vehicle (One)**
- **Booking (Many)** "Includes" -> **Booking Date (One)**
- **Feedback (Many)** "Refers to" -> **Feedback ID (One)**
- **Report (Many)** "Has" -> **Vehicle (Many)**
- **Report (Many)** "Associates with" -> **Report ID (One)**
- **Vehicle (One)** "Belongs to" -> **Fleet (Many)**
- **Vehicle (One)** "Relates to" -> **Driver (One)**
- **Vehicle (One)** "Includes Details of" -> **Vehicle Type, Name, Color, and Needs Minimum (One each)**
- **Fleet (Many)** "Includes" -> **Fleet ID (One)**
- **Fleet (Many)** "Tracks Availability of" -> **Vehicle (Many)**
- **Driver (Many)** "Has" -> **Driver Name and Age (One each)**
- **Driver (Many)** "Belongs to" -> **Fleet (Many)**

Architecture Model level 0



Interface Definitions:

1- Interface I1: Account Management

- boolean login(String email, String password);

- boolean SignUp(String email, String password, String UserName, String ConfirmationPassword);
- String getAccountDetails(String userId);
- boolean checkCredentials(String email, String password);
- boolean validateCredentials(String email, String password);
- boolean invalidCredentials(String email, String password);
- String errorMessage();
- boolean updateDatabase(String email, String password);
- boolean createAccount(String email, String password, String userType);
- boolean updateAccount(String userId, String newDetails);
- boolean deleteAccount(String userId);
- boolean getAccount(String userId);

2- Interface I2 : Booking Management

- boolean createBooking(String userId, String tripId, String date);
- boolean cancelBooking(String bookingId);
- boolean confirmBooking(String bookingId);
- String getBookingDetails(String bookingId);
- boolean offerRetentionPlan(String userId);
- boolean acceptRetentionPlan(String userId);
- String getRetentionDetails(String userId);
- boolean createTrip(String route, String date, int seatsAvailable);
- boolean updateTrip(String tripId, String newDetails);
- boolean cancelTrip(String tripId);
- String getTripDetails(String tripId);

3- Interface I3: Payment Management

- boolean processPayment(String bookingId, double amount);
- boolean confirmPayment(String paymentId);
- String sendInvoice(String email, String bookingId);
- boolean refundPayment(String paymentId);

4- Interface I4: Scheduling Management

- boolean scheduleTrip(String tripId, String date, String time);

- boolean updateSchedule(String scheduleId, String newDetails);
- String getScheduleDetails(String scheduleId);
- boolean cancelSchedule(String scheduleId);

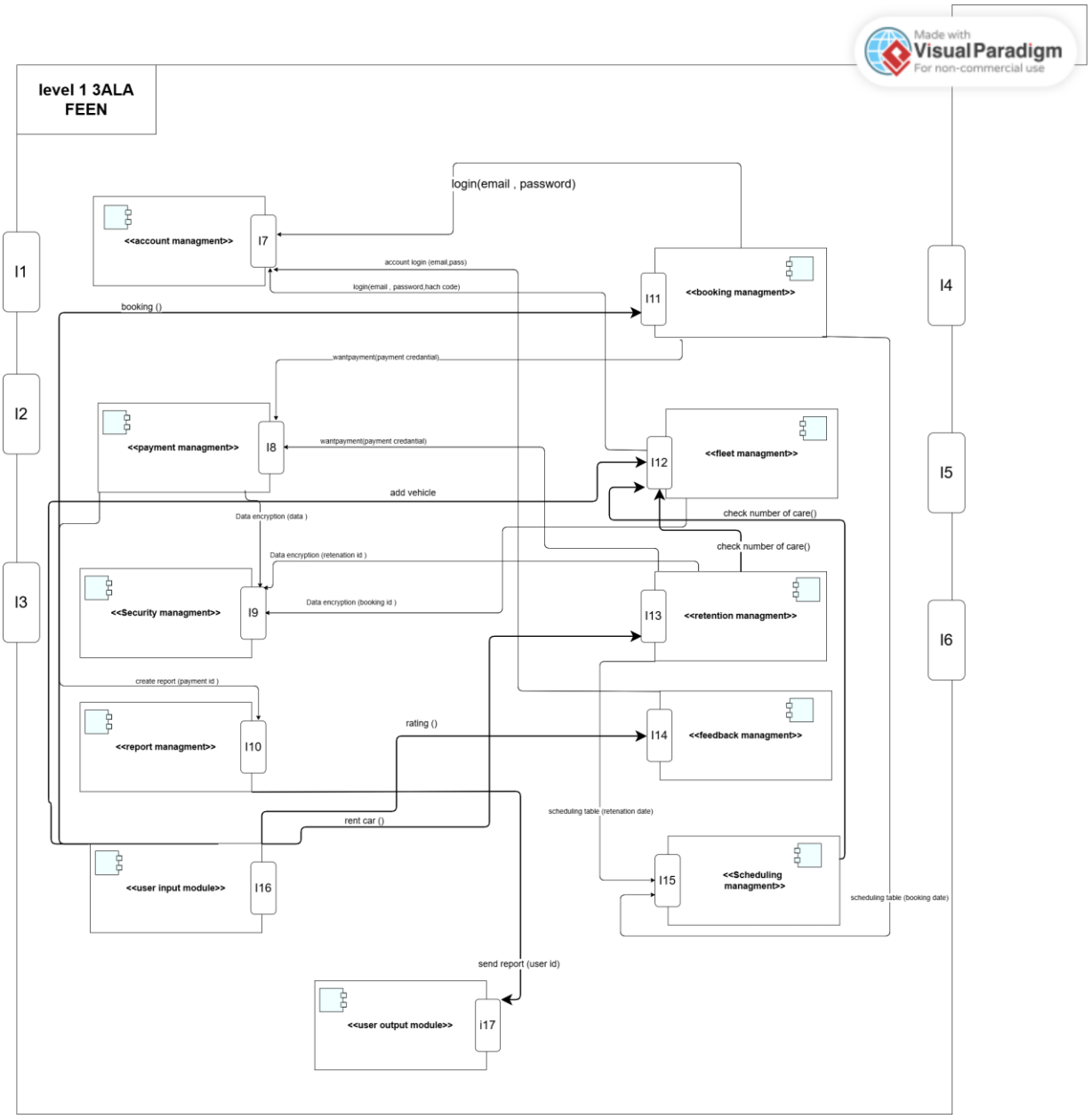
Interface I5: Fleet Management

- boolean removeVehicle(String vehicleId);
- String getVehicleDetails(String vehicleId);
- boolean updateVehicleDetails(String vehicleId, String newDetails);
- boolean scheduleMaintenance(String vehicleId, String date);
- boolean addVehicle(String vehicleDetails);

Interface I6: Reports Management

- String generateReport(String reportType, String dateRange);
- String getReportDetails(String reportId);
- boolean sendReport(String email, String reportId);
- boolean archiveReport(String reportId);

Architecture Model level 1



Interface Definitions:

Interface I7: Account Management

- boolean userLogin(String username, String password);
- boolean adminLogin(String username, String password, String hashcode);

- String getAccountInfo(String accountId);
- boolean changePassword(String accountId, String newPassword);
- boolean createAccount(String email, String password);
- boolean updateAccount(String accountId, String details);
- boolean deleteAccount(String accountId);

Interface I8: Payment Management

- boolean makePayment(String bookingId, double amount);
- String getPaymentStatus(String paymentId);
- String generateReceipt(String paymentId);
- boolean refundPayment(String paymentId);
- boolean applyDiscount(String bookingId, double discount);

Interface I9: Security Management

- String encrypt(String data);
- String decrypt(String encryptedData);
- boolean validateAccess(String userId);
- boolean logEvent(String event);

Interface I10: Report Management

- String createReport(String reportType, String dateRange);
- String getReport(String reportId);
- boolean sendReport(String reportId, String email);
- boolean archiveReport(String reportId);

Interface I11: Booking Management

- boolean createBooking(String userId, String tripId);
- boolean cancelBooking(String bookingId);
- String getBookingInfo(String bookingId);
- boolean updateBooking(String bookingId, String details);

Interface I12: Fleet Management

- boolean addVehicle(String vehicleDetails);
- boolean removeVehicle(String vehicleId);
- String getVehicleInfo(String vehicleId);
- boolean updateVehicle(String vehicleId, String details);
- boolean scheduleMaintenance(String vehicleId, String date);

Interface I13: Retention Management

- boolean createRetention(String userId);
- boolean cancelRetention(String userId);
- String getRetentionDetails(String userId);
- boolean offerRetention(String userId);

Interface I14: Feedback Management

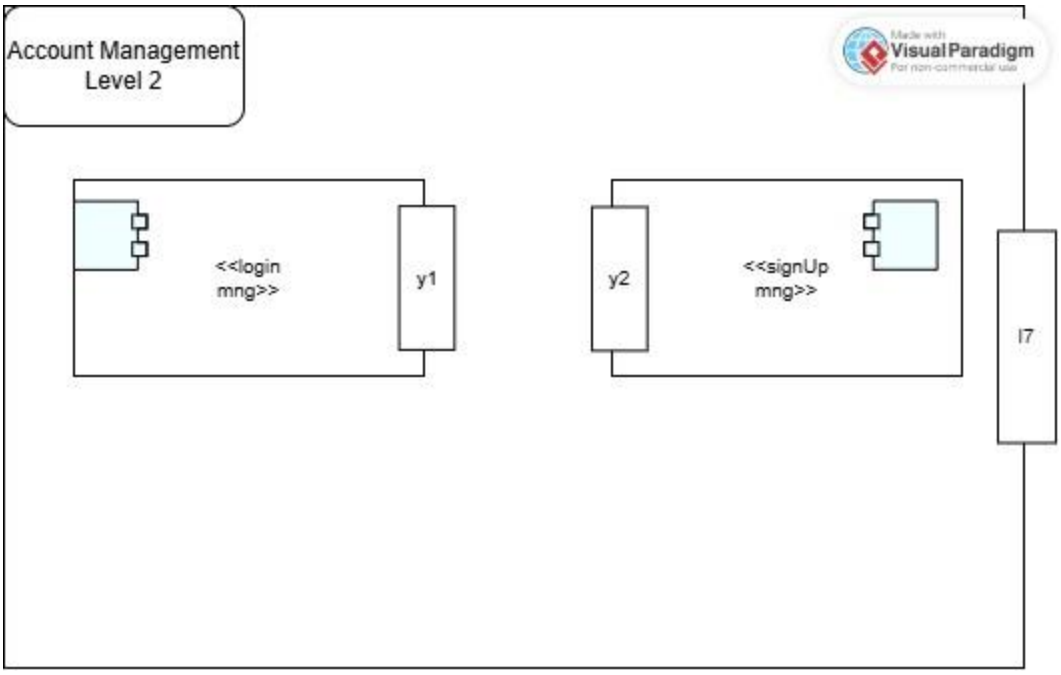
- boolean submitFeedback(String userId, String feedbackText);
- String getFeedback(String feedbackId);
- boolean updateFeedback(String feedbackId, String updatedText);
- boolean deleteFeedback(String feedbackId);
- String analyzeFeedback(String tripId);

Interface I15: Scheduling Management

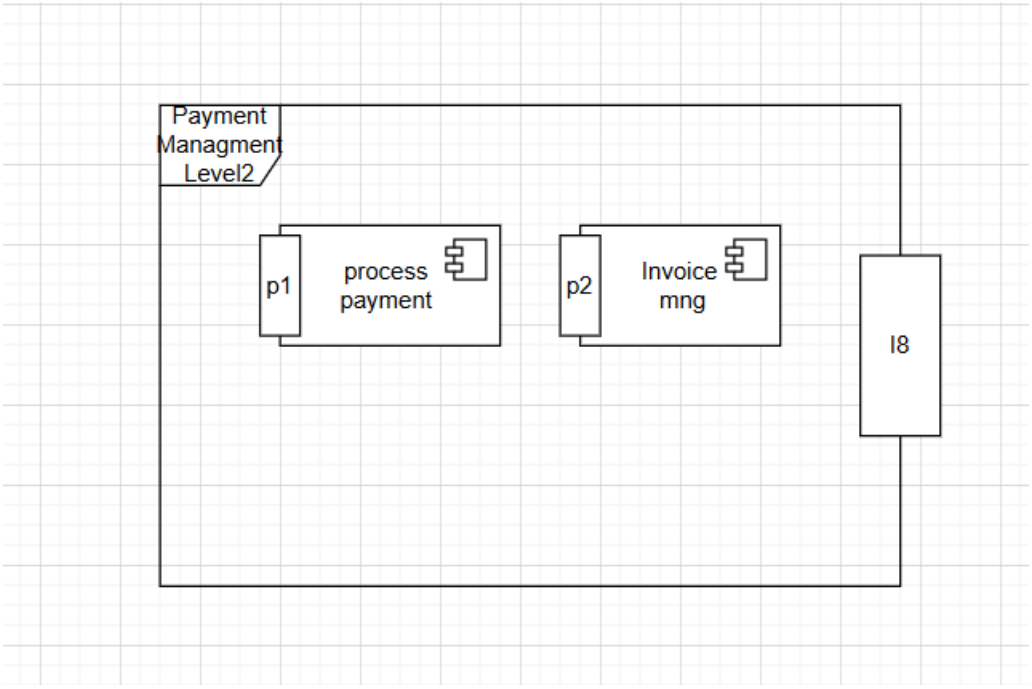
- boolean scheduleTrip(String tripId, String date);
- boolean updateSchedule(String scheduleId, String newDetails);
- String getSchedule(String scheduleId);
- boolean cancelSchedule(String scheduleId);

Architecture Model level 2

Account Management level 2



Payment Management level 2



Interface Definitions:

1. Interface y1: loginManagement Interface:

- public boolean userLogin(String email, String password);

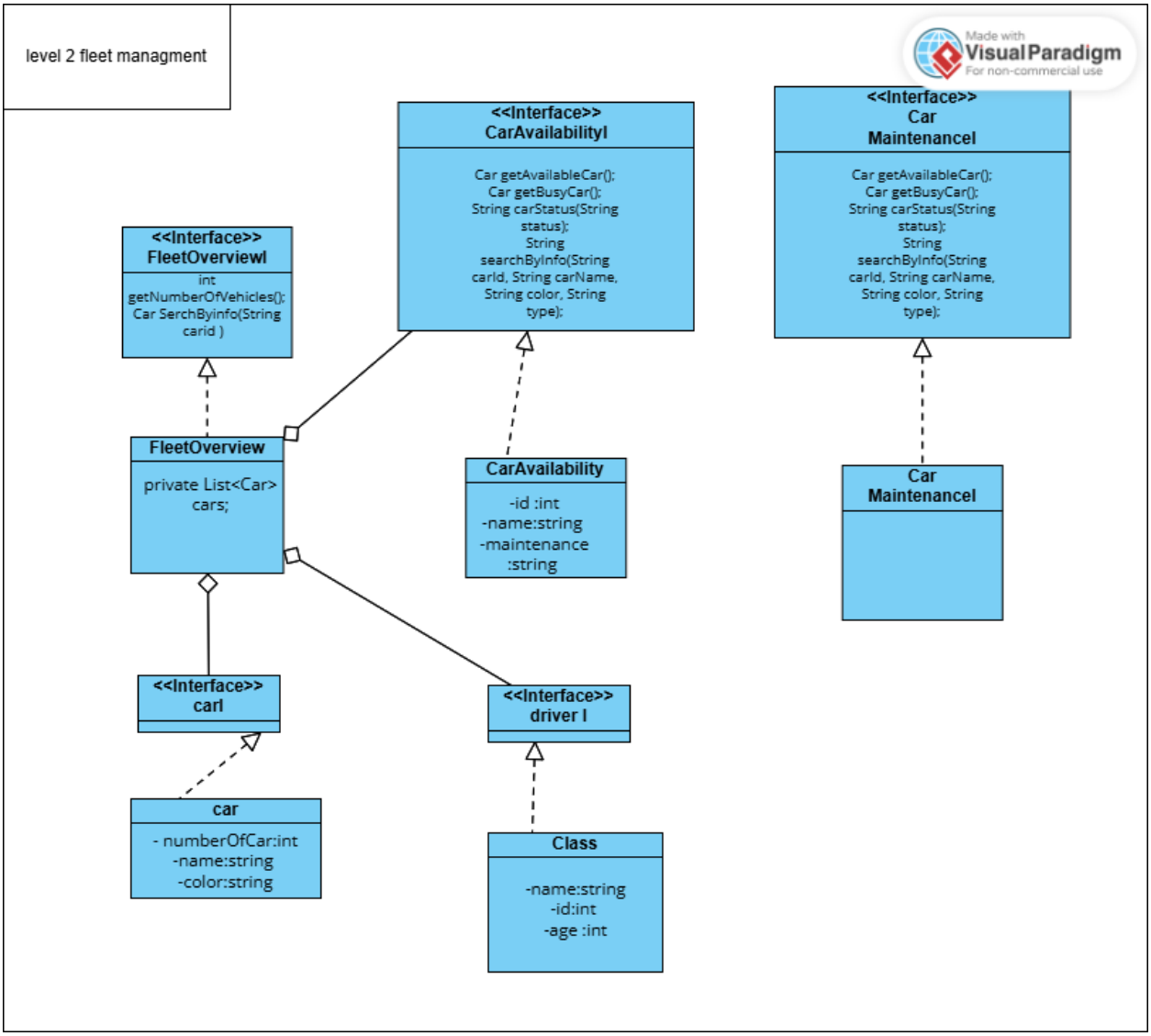
- boolean adminLogin(String username, String password, String hashcode);
- public boolean validateCredentials(String email, String password);
- public String getUserDetails(String email);

2. Interface y2: signUpManagement Interface:

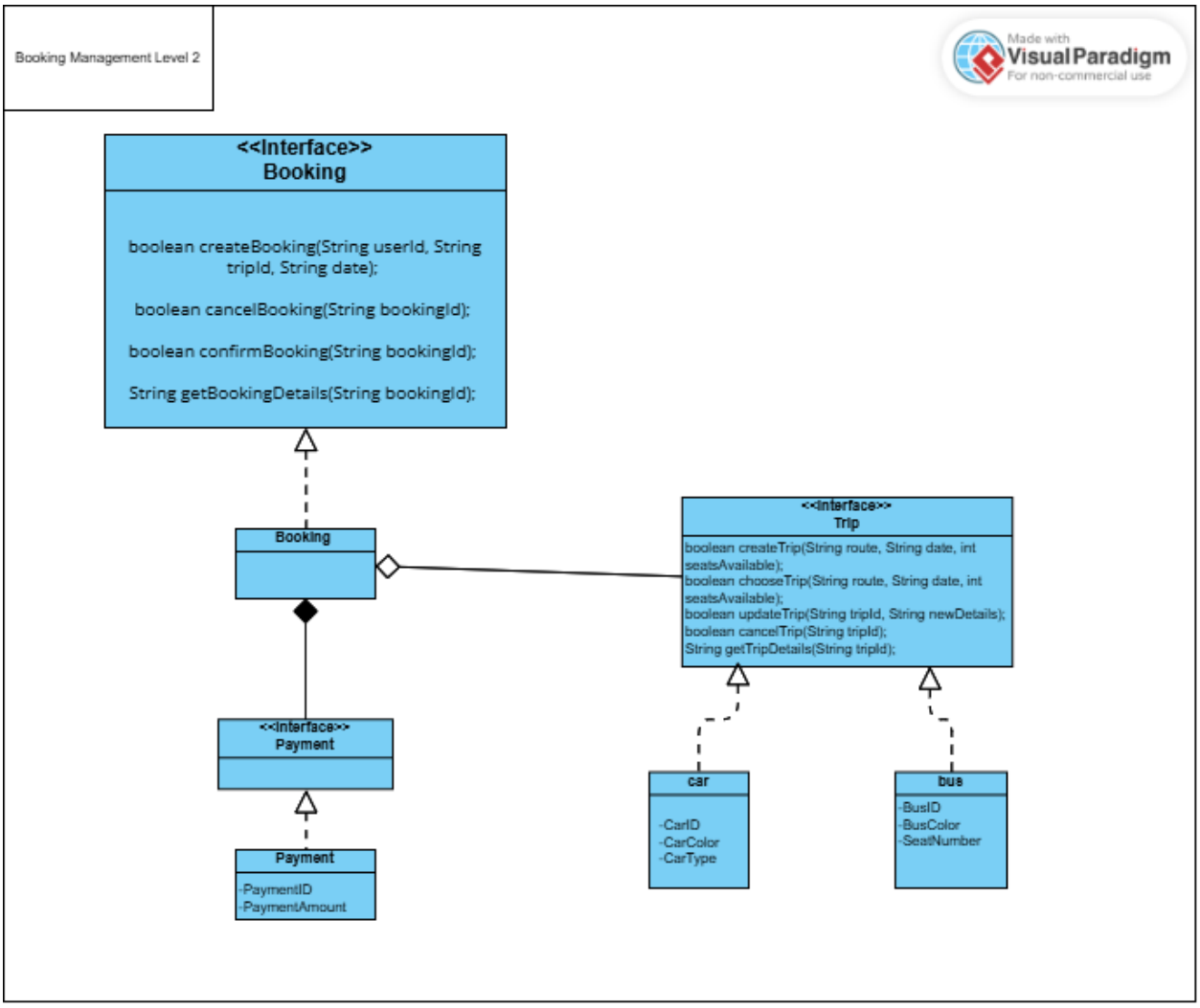
- public boolean signUp(String email, String password, String username);
- public boolean checkUserExists(String email);
- public String generateUserId();

_____-

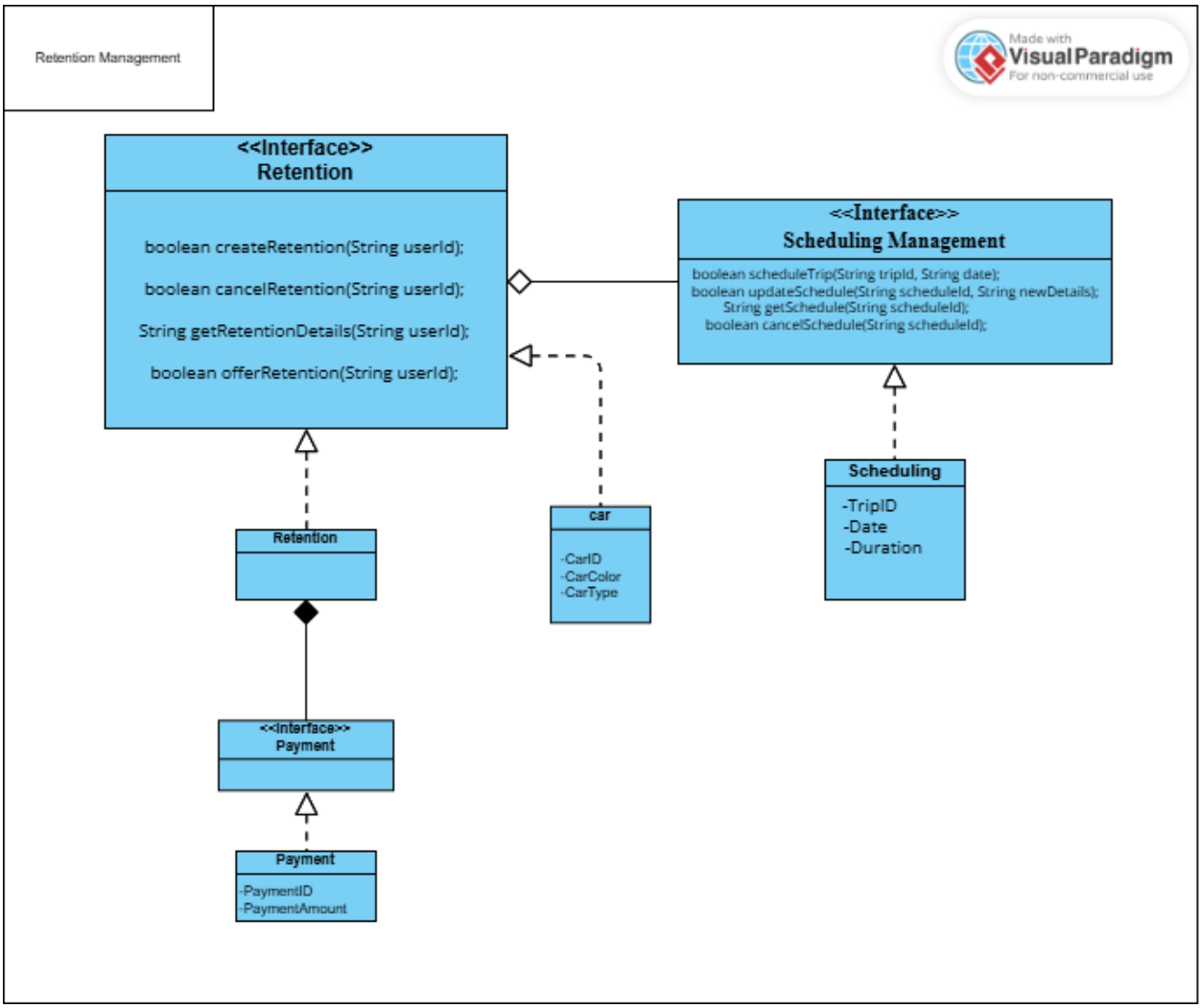
Fleet Management level 2



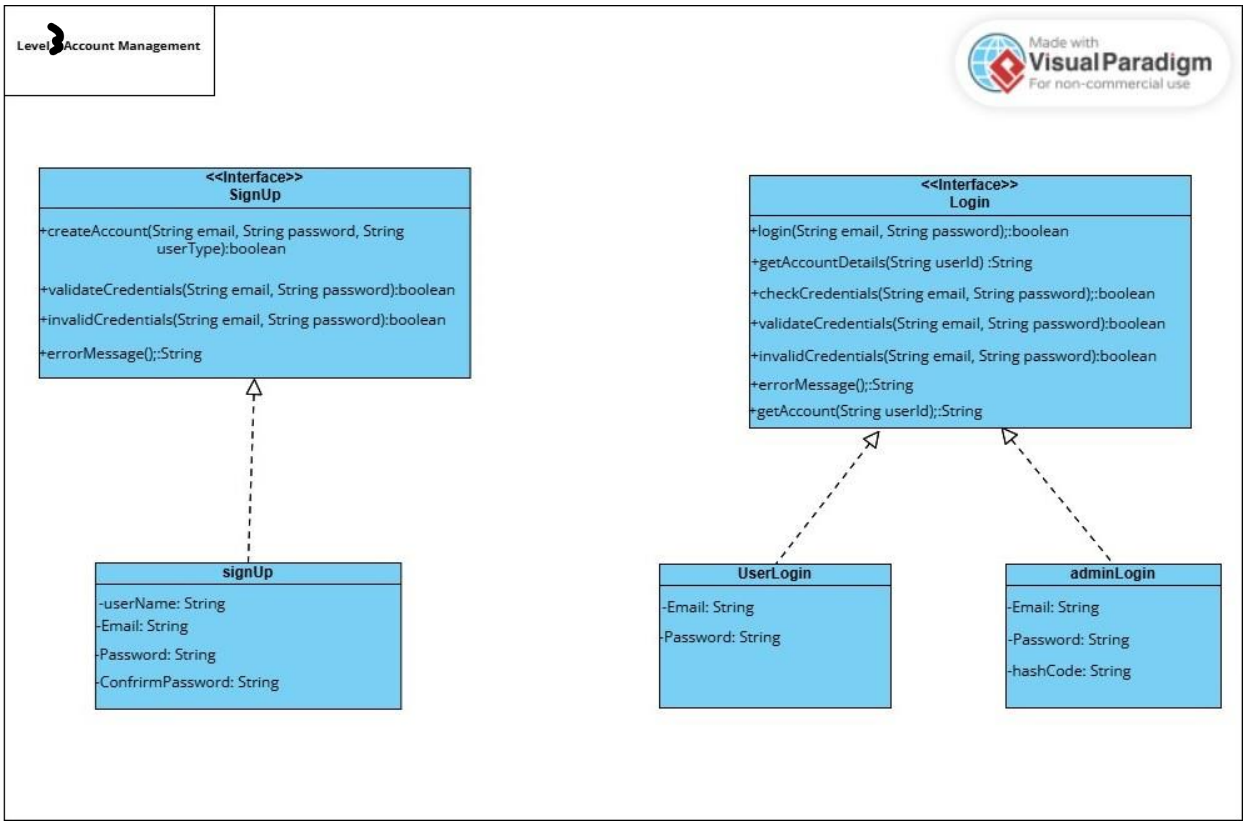
Booking Management level 2



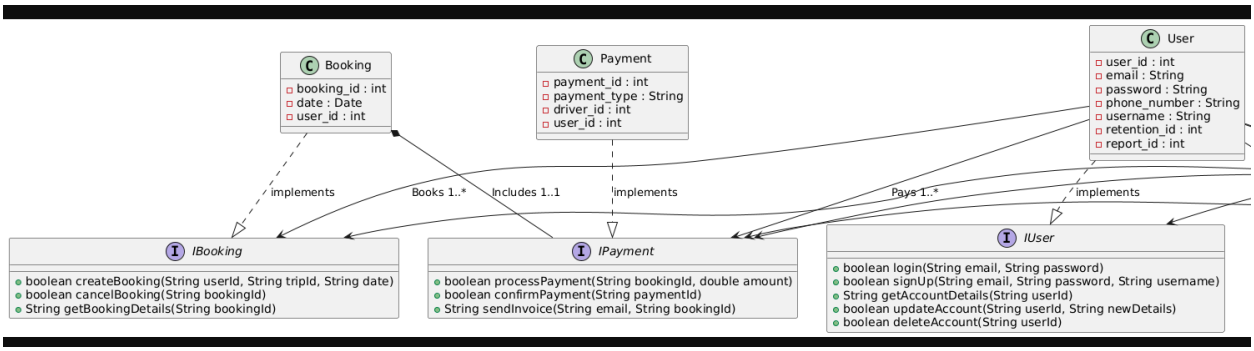
Retention Management level 2

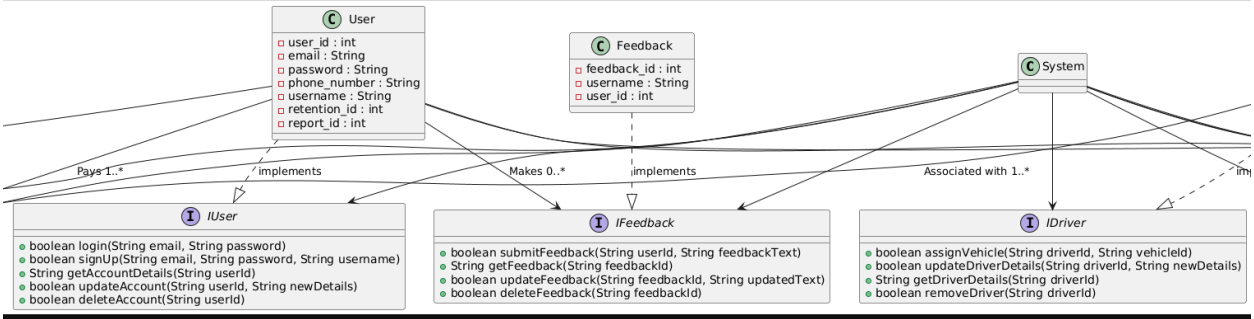


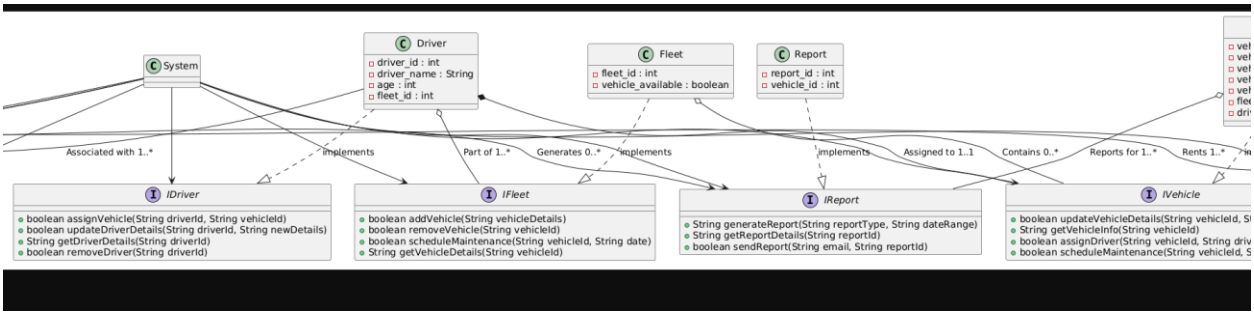
Account Management level 3

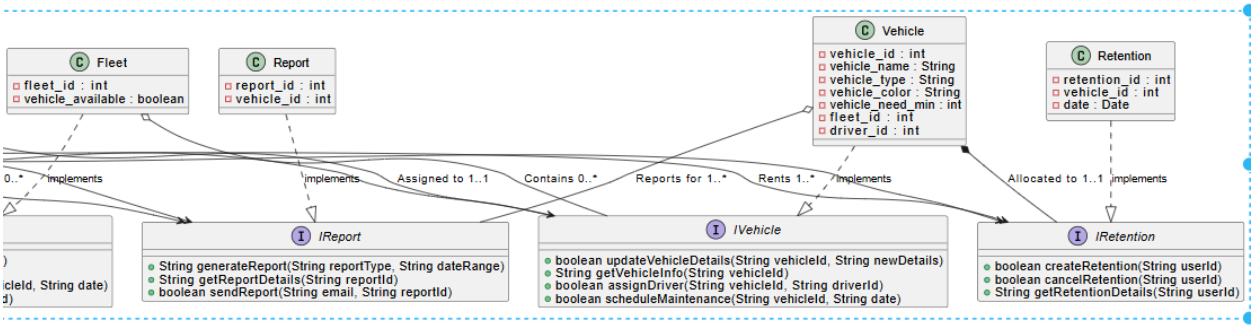


Class Diagram


Booking
booking_id : int
date : Date
user_id : int
Payment
payment_id : int
payment_type : String
driver_id : int
user_id : int
User
user_id : int
email : String
password : String
phone_number : String
username : String
retention_id : int
report_id : int
implements
Books 1..*
Includes 1..1
implements
Pays 1..*
implements
IBooking
boolean createBooking(String userId, String tripId, String date)
boolean cancelBooking(String bookingId)
String getBookingDetails(String bookingId)
IPayment
boolean processPayment(String bookingId, double amount)
boolean confirmPayment(String paymentId)
String sendInvoice(String email, String bookingId)
IUser
boolean login(String email, String password)
boolean signUp(String email, String password, String username)
String getAccountDetails(String userId)
boolean updateAccount(String userId, String newDetails)
boolean deleteAccount(String userId)


User
user_id : int
email : String
password : String
phone_number : String
username : String
retention_id : int
report_id : int
Feedback
feedback_id : int
username : String
user_id : int
System
Pays 1..*
implements
Makes 0..*
implements
Associated with 1..*
IUser
boolean login(String email, String password)
boolean signUp(String email, String password, String username)
String getAccountDetails(String userId)
boolean updateAccount(String userId, String newDetails)
boolean deleteAccount(String userId)
IFeedback
boolean submitFeedback(String userId, String feedbackText)
String getFeedback(String feedbackId)
boolean updateFeedback(String feedbackId, String updatedText)
boolean deleteFeedback(String feedbackId)
IDriver
boolean assignVehicle(String driverId, String vehicleId)
boolean updateDriverDetails(String driverId, String newDetails)
String getDriverDetails(String driverId)
boolean removeDriver(String driverId)

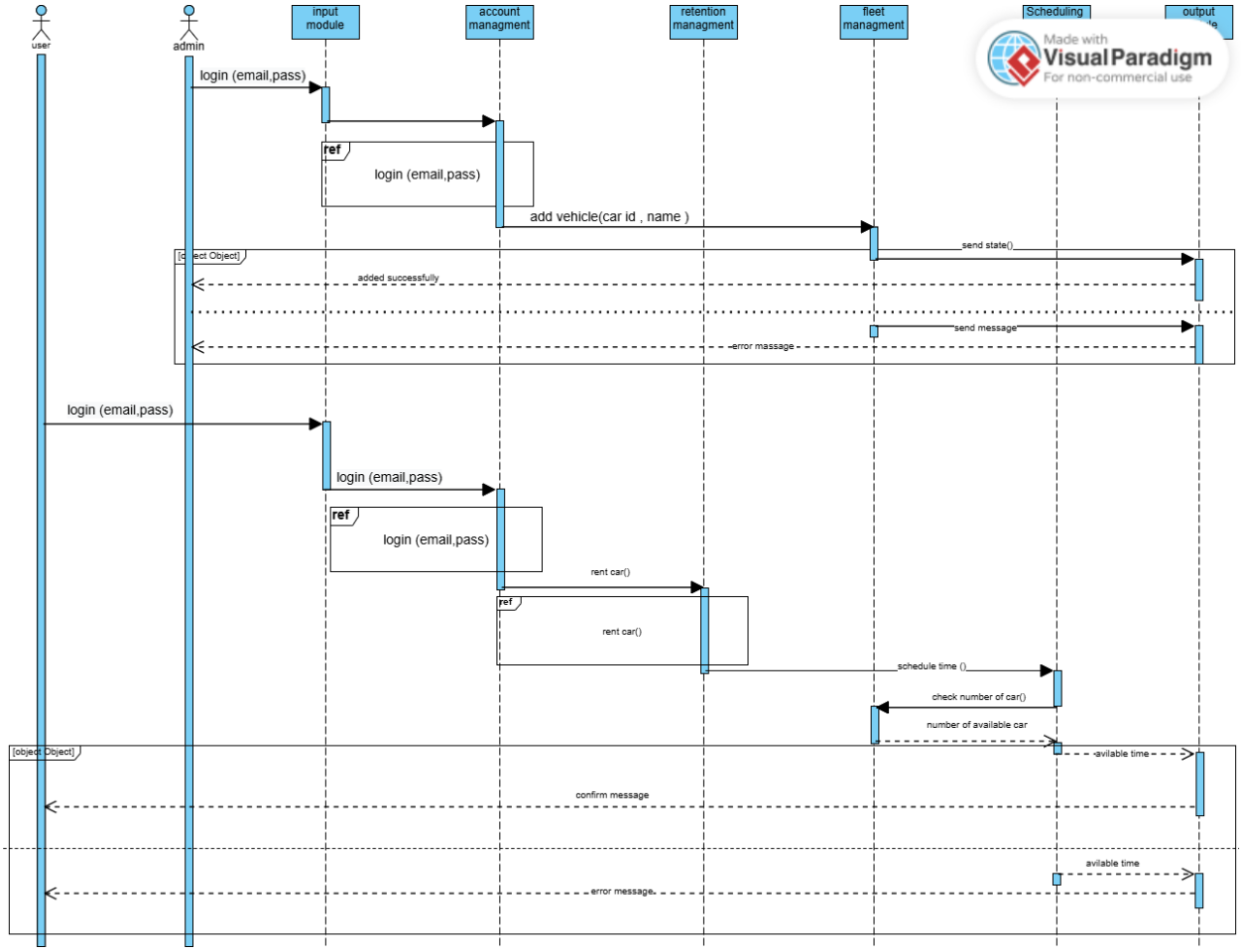

System
Driver
driver_id : int
driver_name : String
age : int
fleet_id : int
Fleet
fleet_id : int
vehicle_available : boolean
Report
report_id : int
vehicle_id : int
Associated with 1..*
implements
Part of 1..*
Generates 0..*
implements
implements
Assigned to 1..1
Contains 0..*
Reports for 1..*
Rents 1..*
IDriver
boolean assignVehicle(String driverId, String vehicleId)
boolean updateDriverDetails(String driverId, String newDetails)
String getDriverDetails(String driverId)
boolean removeDriver(String driverId)
IFleet
boolean addVehicle(String vehicleDetails)
boolean removeVehicle(String vehicleId)
boolean scheduleMaintenance(String vehicleId, String date)
String getVehicleDetails(String vehicleId)
IReport
String generateReport(String reportType, String dateRange)
String getReportDetails(String reportId)
boolean sendReport(String email, String reportId)
IVehicle
boolean updateVehicleDetails(String vehicleId,
String getVehicleInfo(String vehicleId)
boolean assignDriver(String vehicleId, String driv
boolean scheduleMaintenance(String vehicleId,


Fleet
fleet_id : int
vehicle_available : boolean
Report
report_id : int
vehicle_id : int
Vehicle
vehicle_id : int
vehicle_name : String
vehicle_type : String
vehicle_color : String
vehicle_need_min : int
fleet_id : int
driver_id : int
Retention
retention_id : int
vehicle_id : int
date : Date
implements
implements
Assigned to 1..1
Contains 0..*
Reports for 1..*
Rents 1..*
implements
Allocated to 1..1
implements
IReport
String generateReport(String reportType, String dateRange)
String getReportDetails(String reportId)
boolean sendReport(String email, String reportId)
IVehicle
boolean updateVehicleDetails(String vehicleId, String newDetails)
String getVehicleInfo(String vehicleId)
boolean assignDriver(String vehicleId, String driverId)
boolean scheduleMaintenance(String vehicleId, String date)
IRetention
boolean createRetention(String userId)
boolean cancelRetention(String userId)
String getRetentionDetails(String userId)

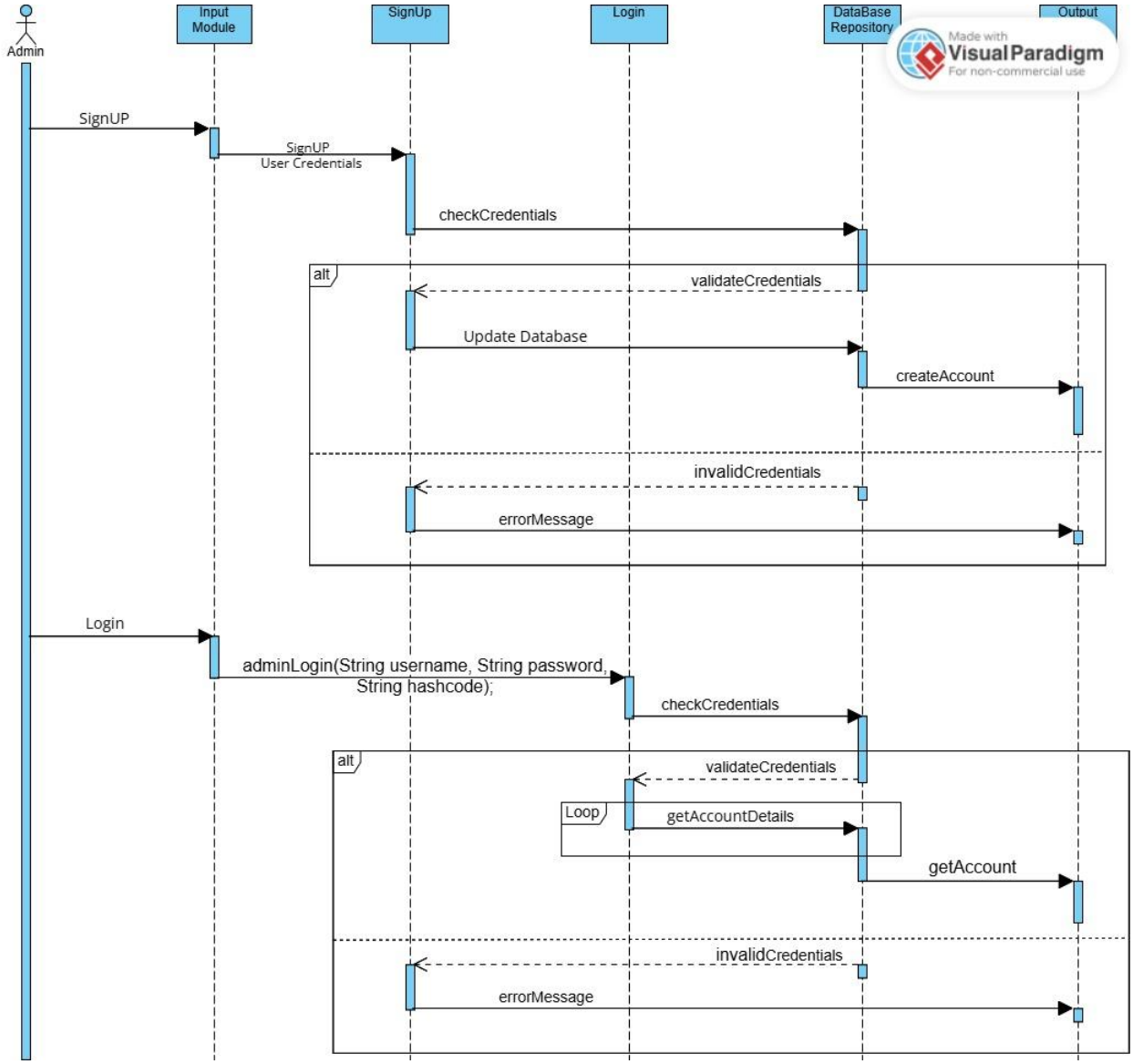
Traceability matrix

Use cases **Designed Class**	Account Management	Booking Management	Payment Management	Fleet Managment	Retention Management	Security Management	Report Management	Scheduling Management	Feedback Management
Booking		✓	✓					✓	
Payment			✓						
User	✓	✓	✓		✓	✓			✓
Feedback									✓
System	✓	✓	✓	✓	✓	✓	✓	✓	✓
Driver		✓						✓	
Fleet				✓					
Report							✓		
Vehicle				✓				✓	
Retention					✓				

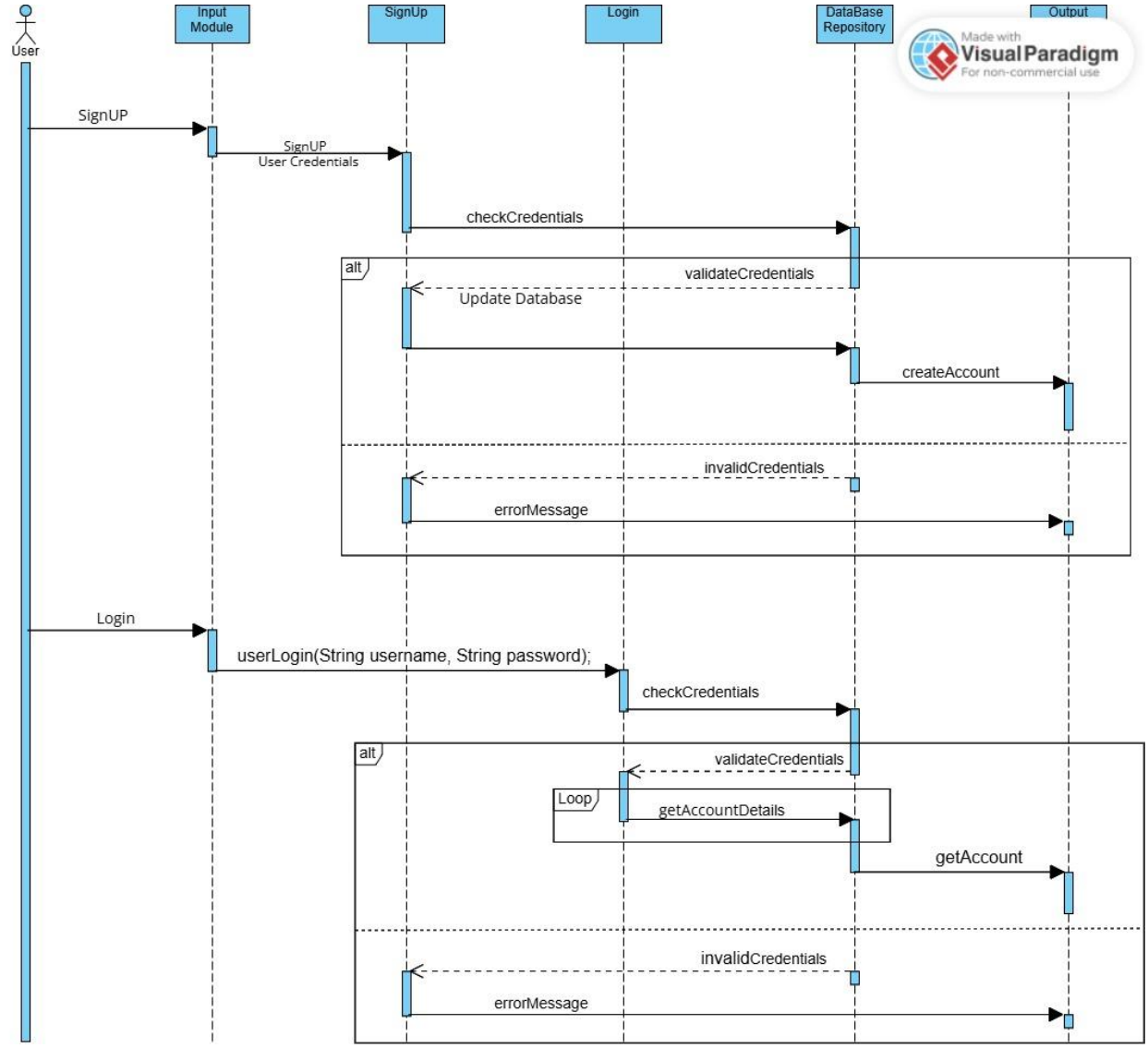
Sequence diagram for Fleet Management:



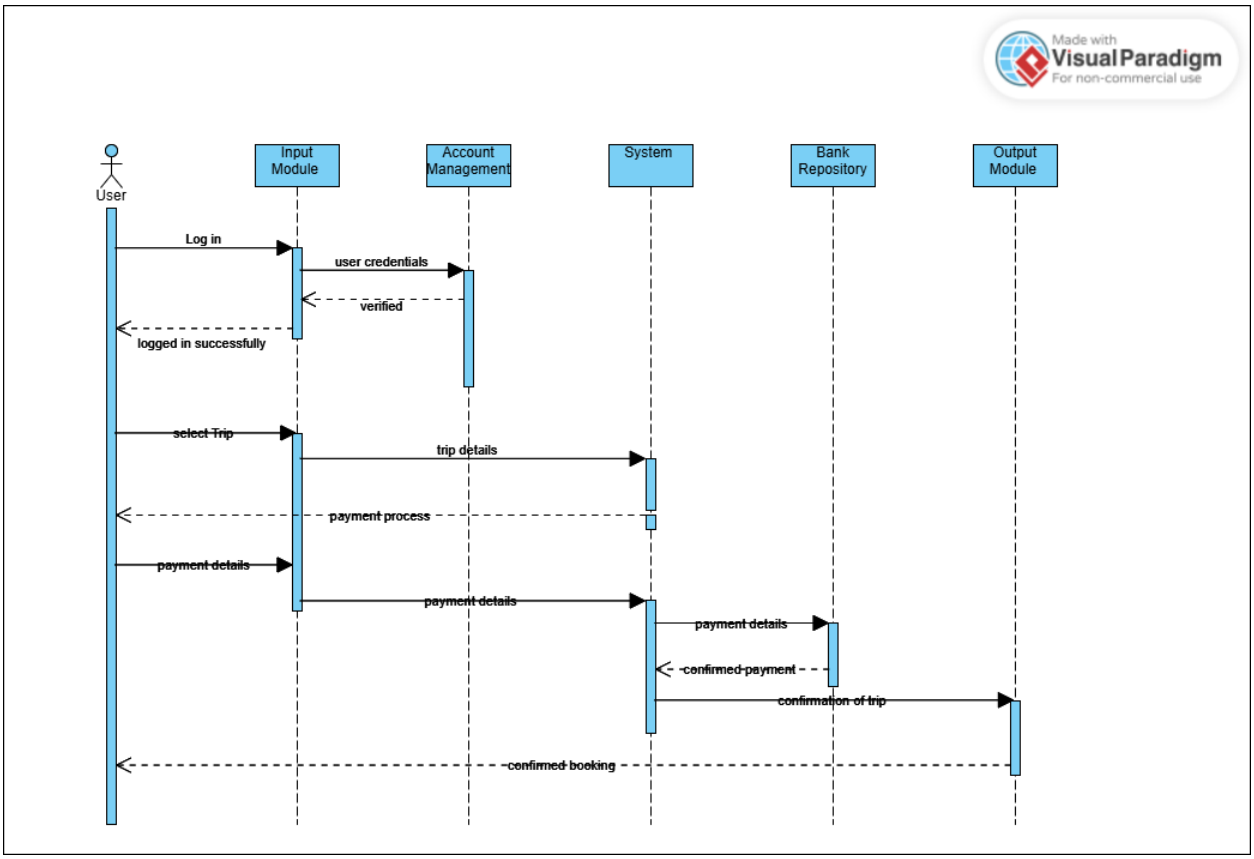
Sequence Diagram Account Management for Admin:



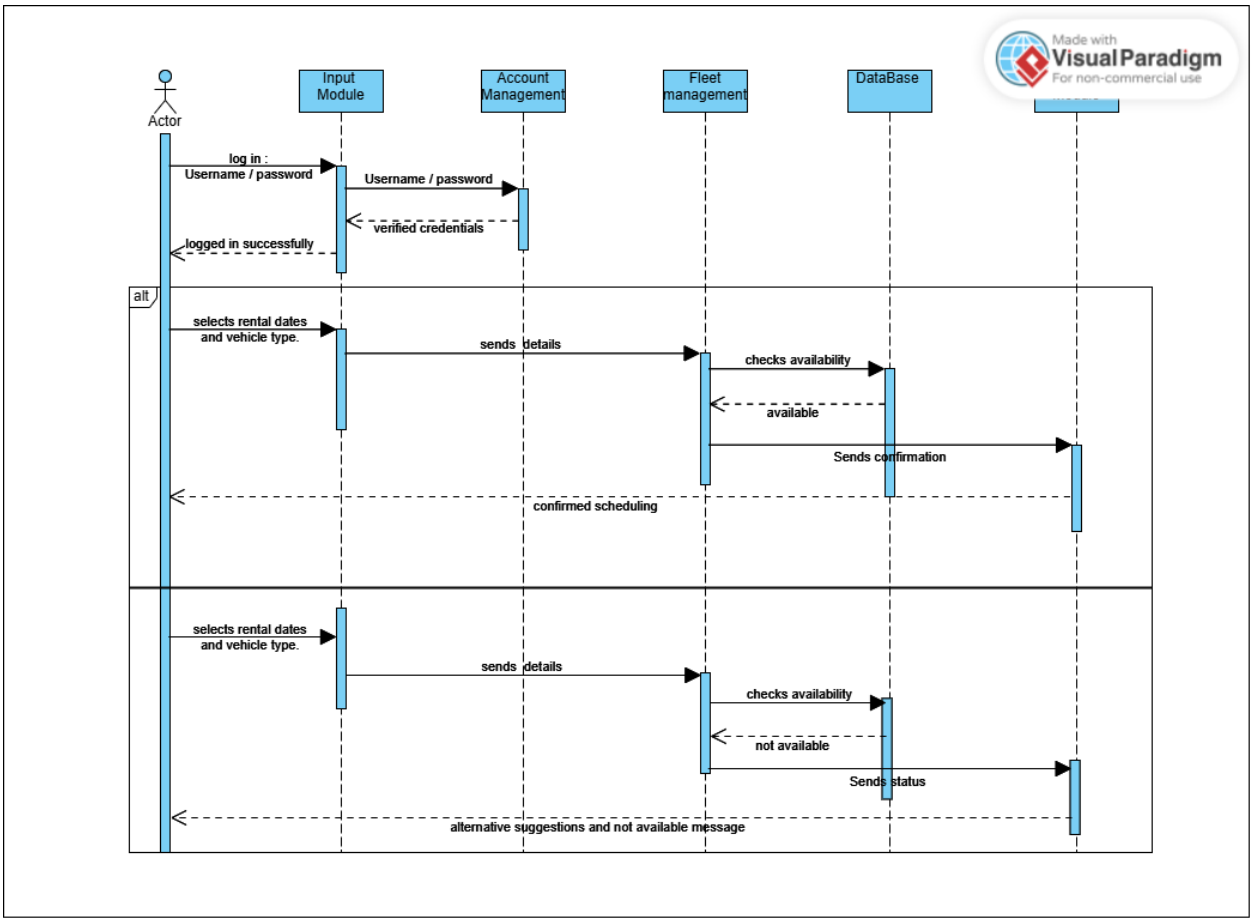
Sequence Diagram Account Management for User:



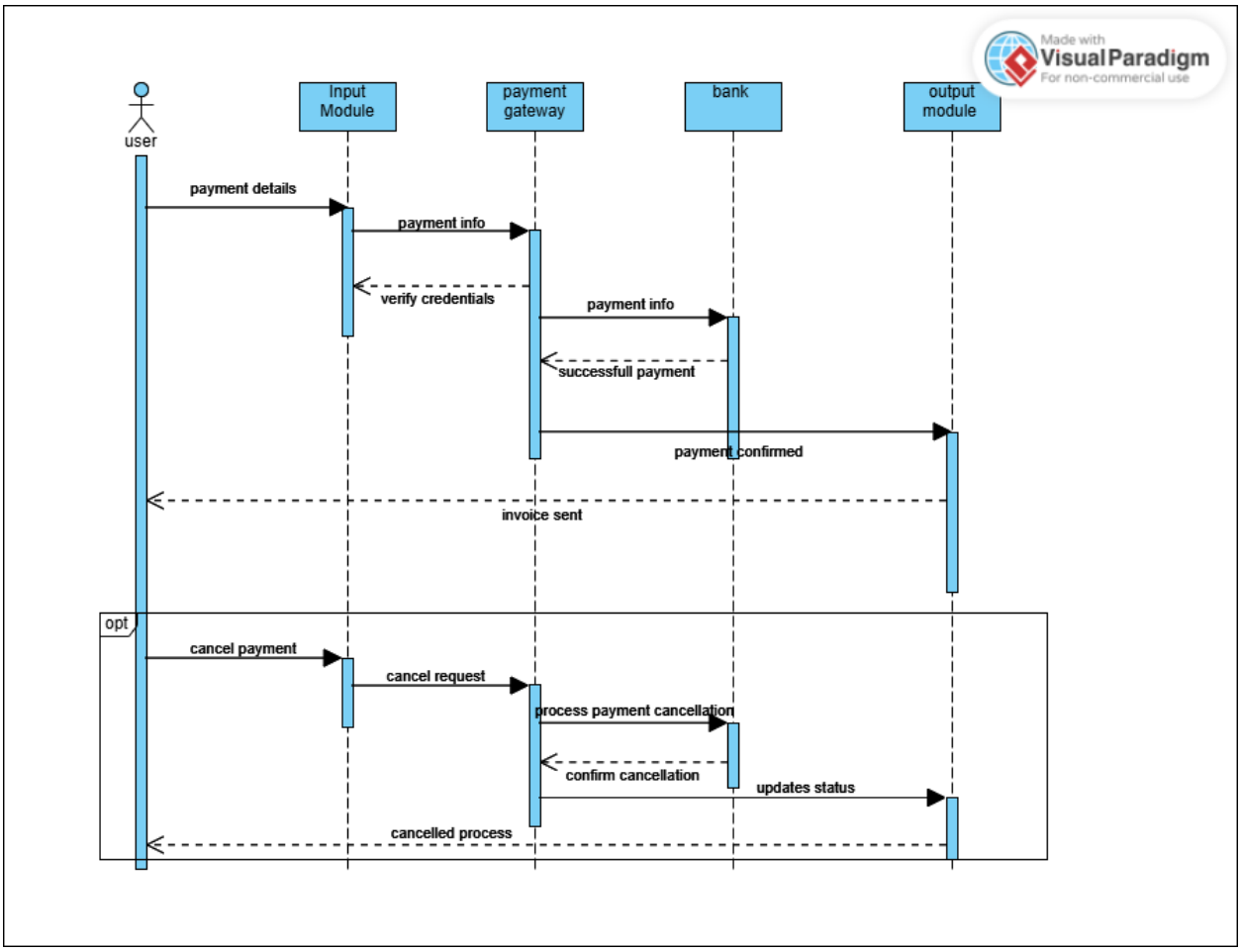
Sequence Diagram Booking Management:



Sequence Diagram Renting Management :

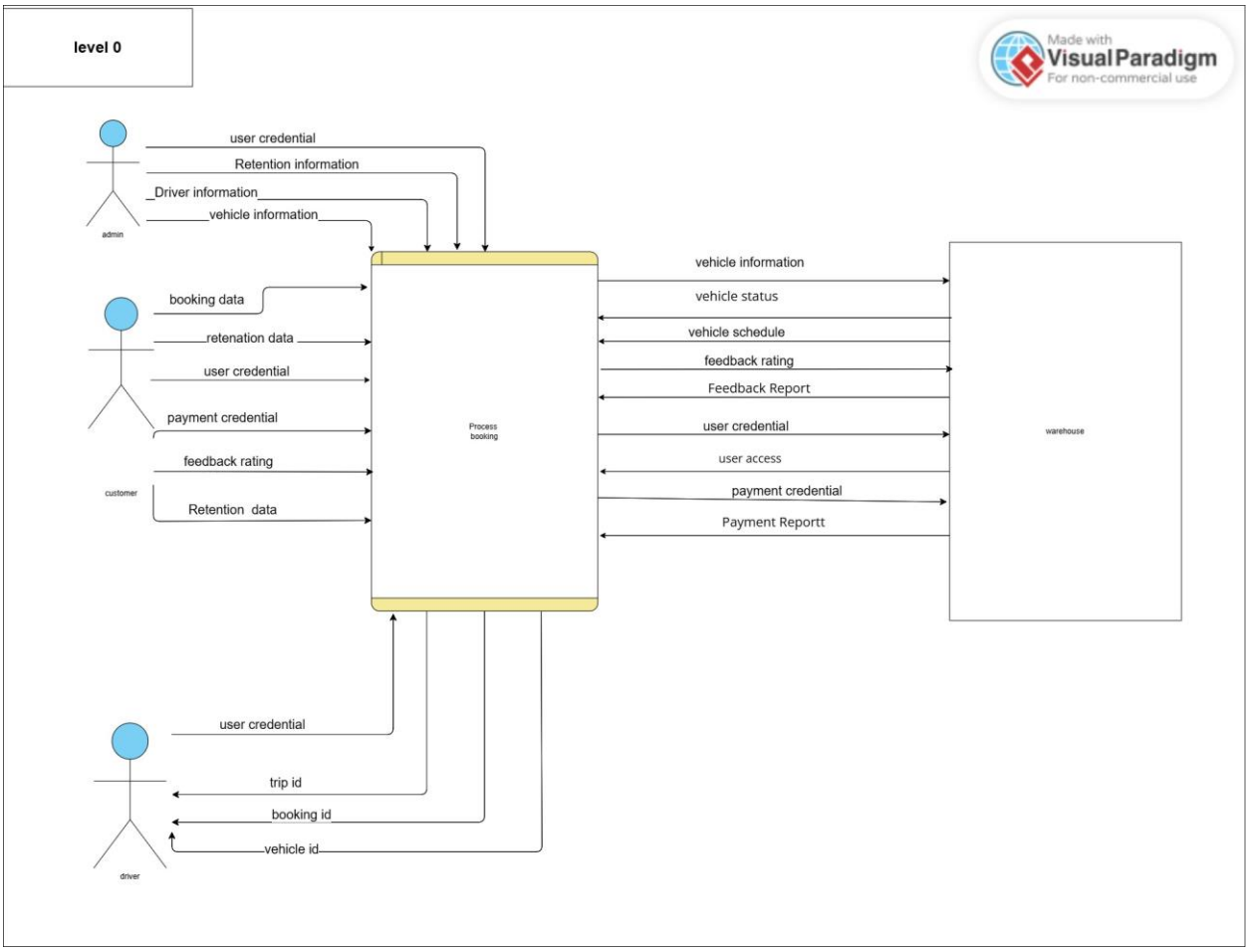


Sequence Diagram Payment Management :

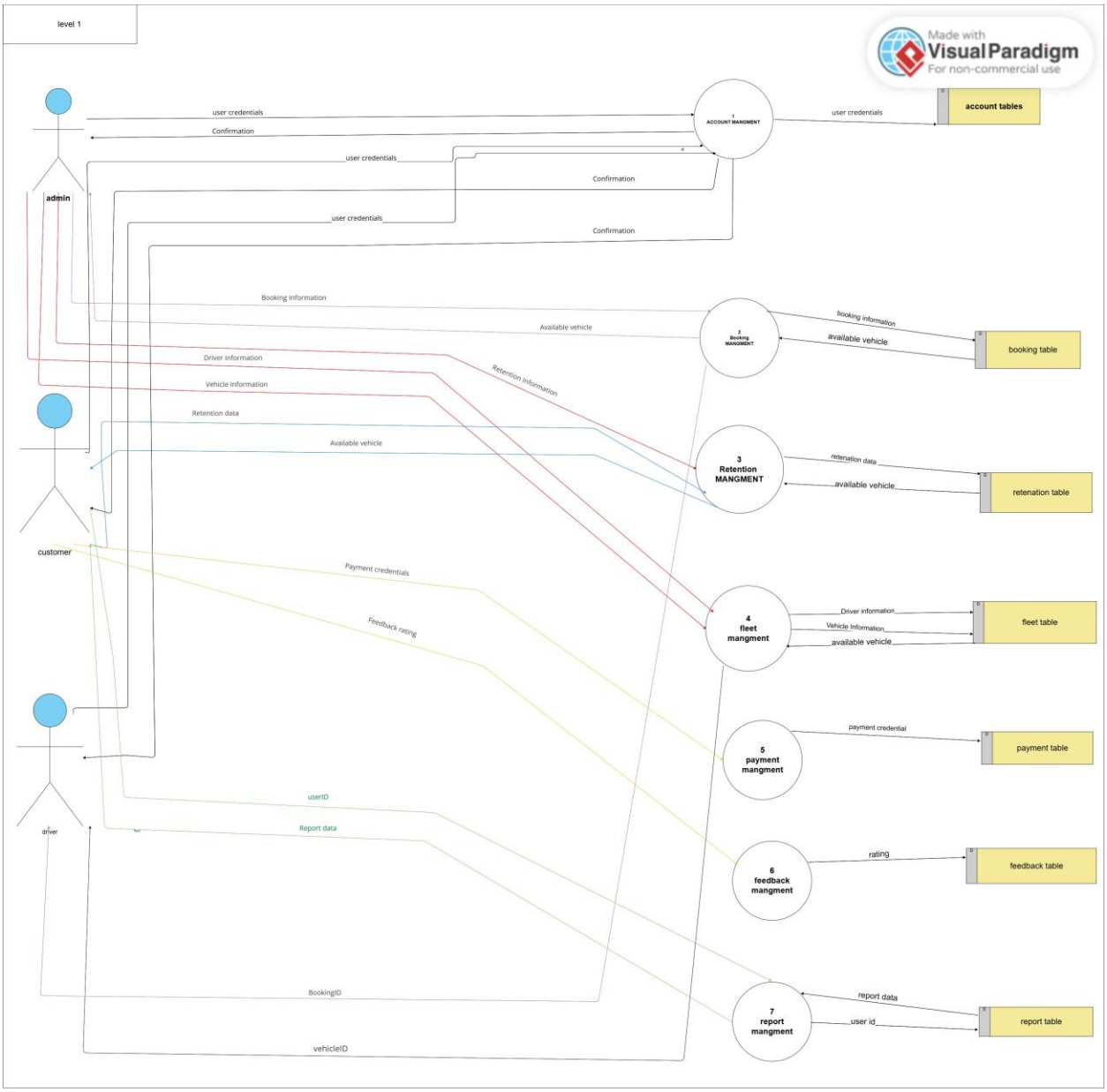


Data Flows:

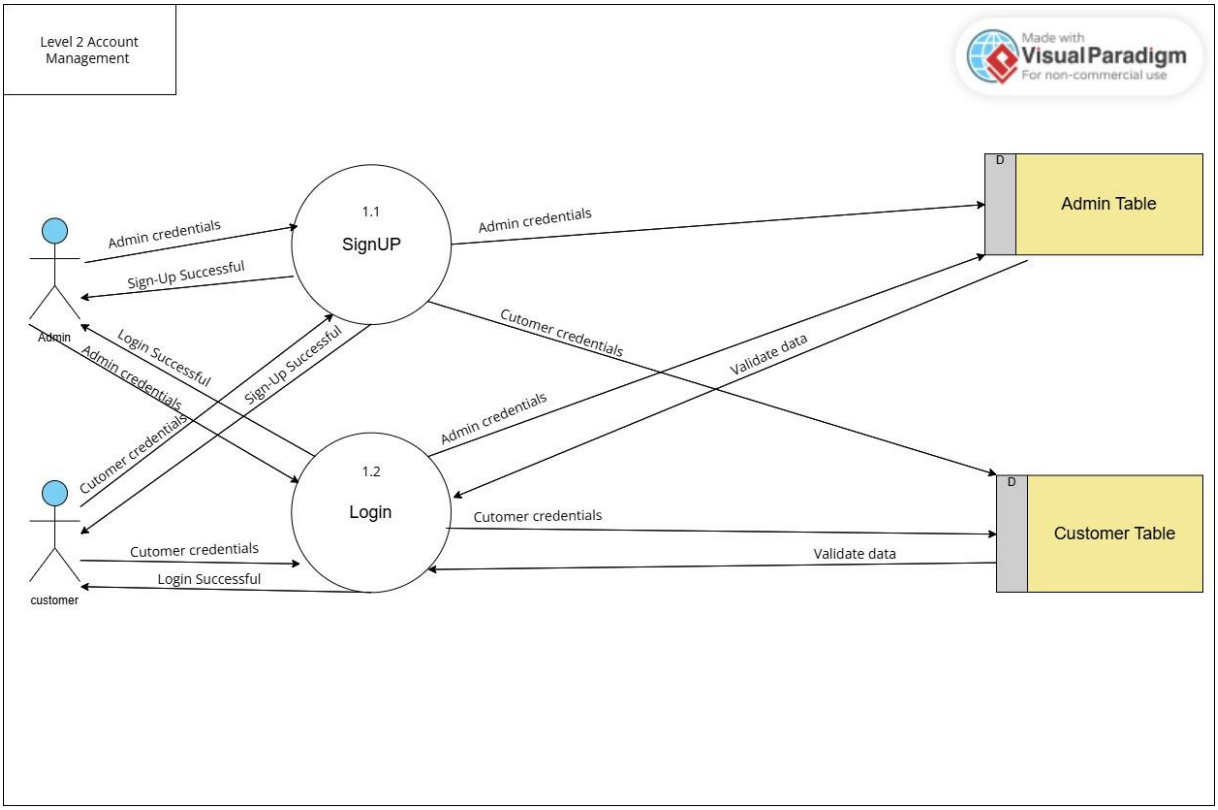
DFD LEVEL 0


level 0
Made with Visual Paradigm
For non-commercial use
admin
user credential
Retention information
Driver information
vehicle information
customer
booking data
retenation data
user credential
payment credential
feedback rating
Retention data
Process booking
vehicle information
vehicle status
vehicle schedule
feedback rating
Feedback Report
user credential
user access
payment credential
Payment Reportt
warehouse
driver
user credential
trip id
booking id
vehicle id

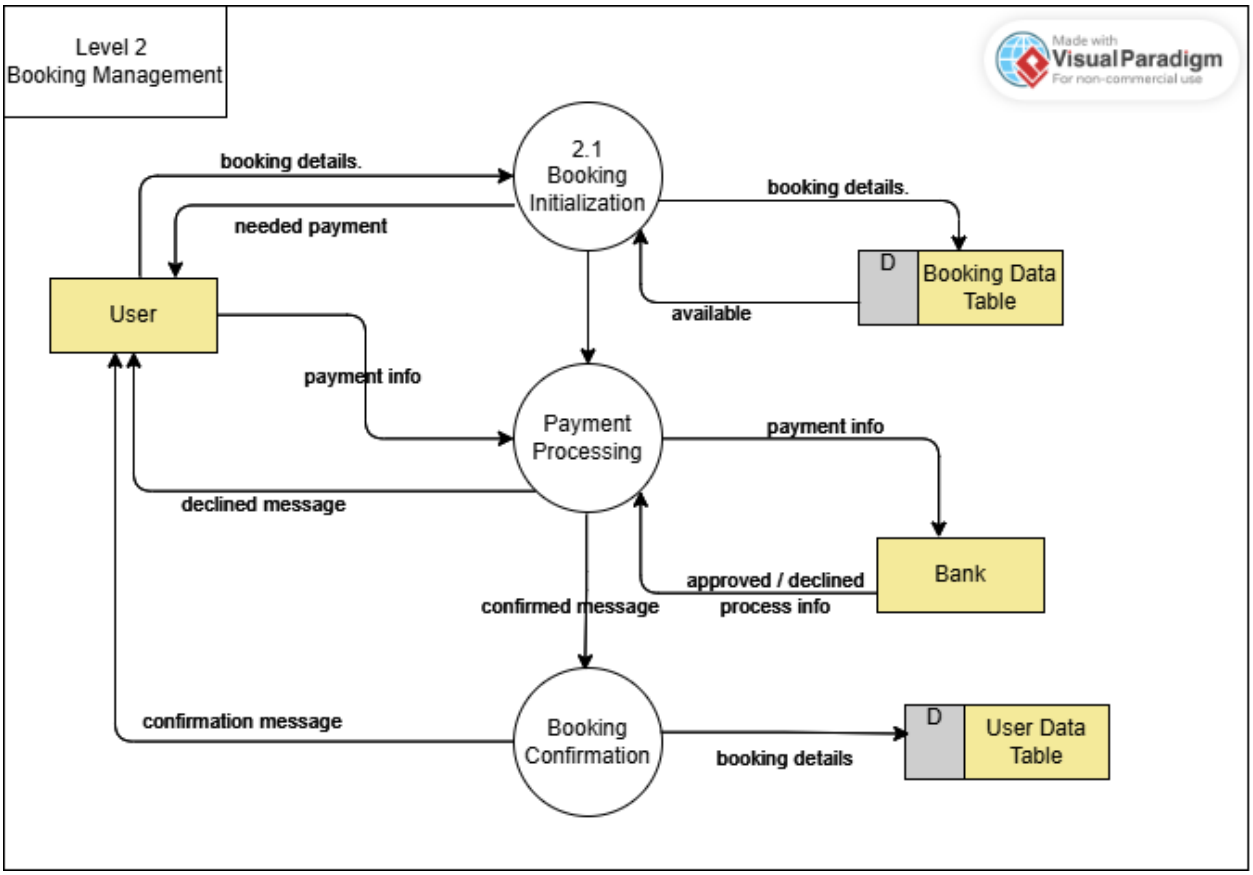
DFD Level 1



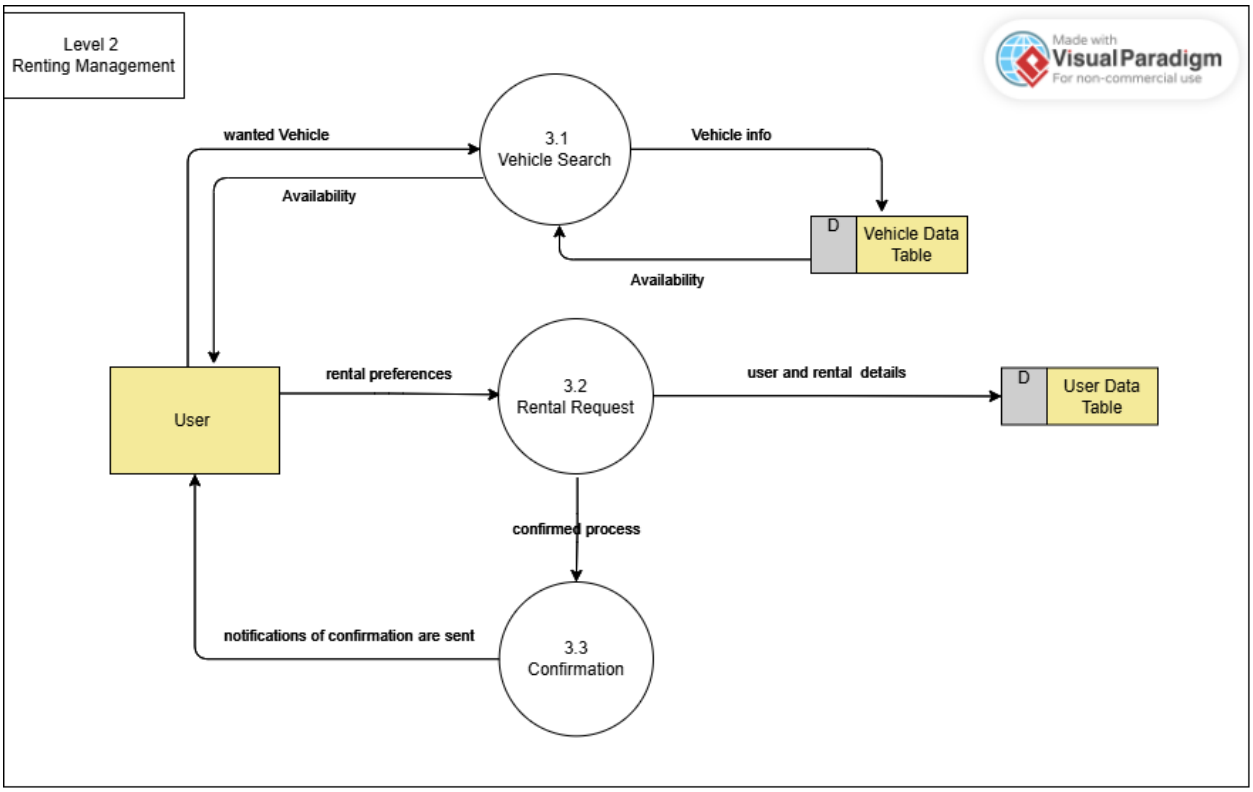
DFD Account Management Level 2



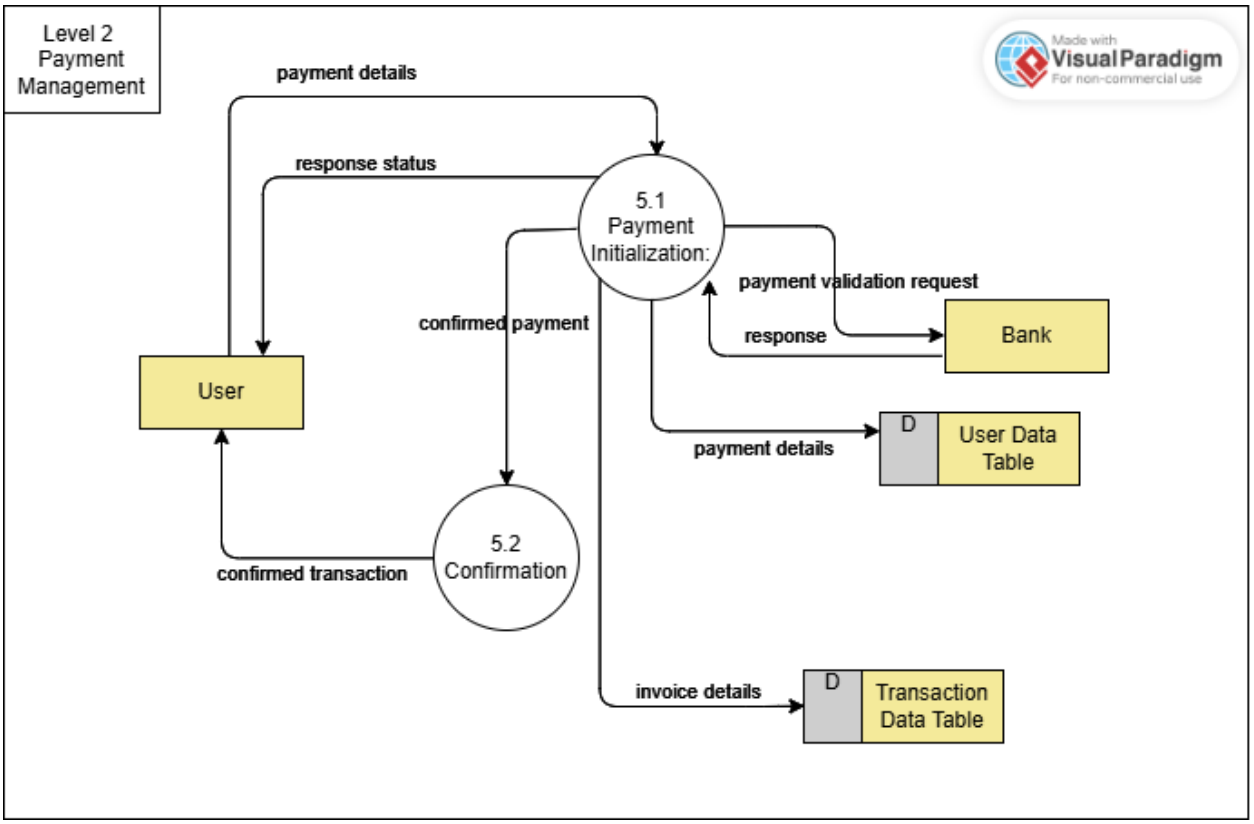
DFD Booking Management Level 2



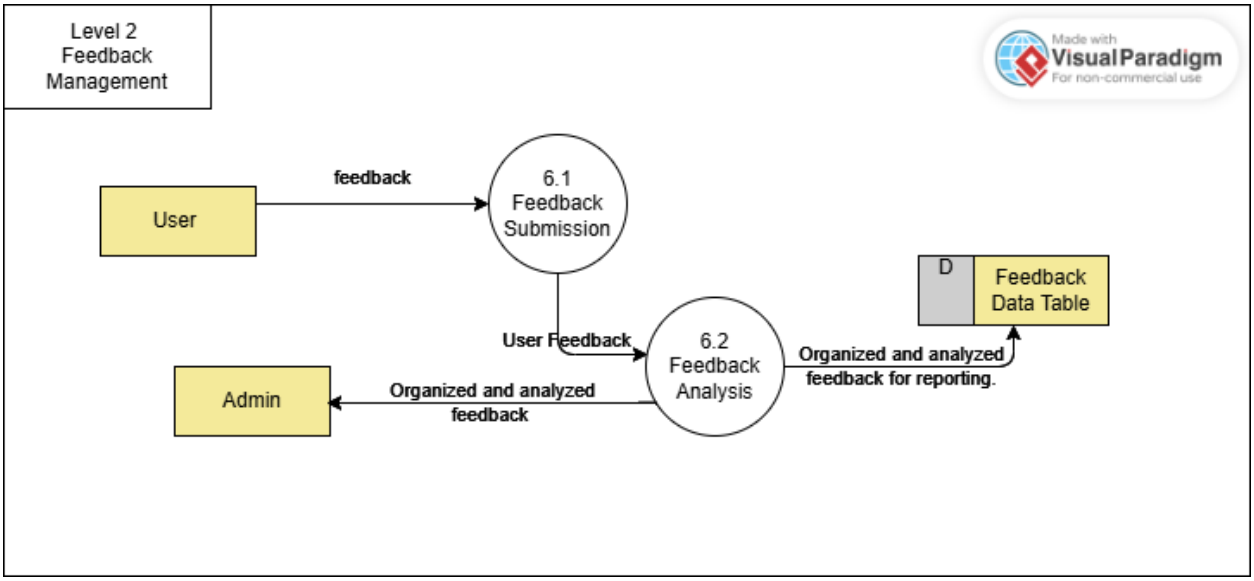
DFD Renting Management Level 2


Level 2
Renting Management
Made with
Visual Paradigm
For non-commercial use
wanted Vehicle
3.1
Vehicle Search
Vehicle info
Availability
D
Vehicle Data
Table
Availability
User
rental preferences
3.2
Rental Request
user and rental details
D
User Data
Table
confirmed process
notifications of confirmation are sent
3.3
Confirmation

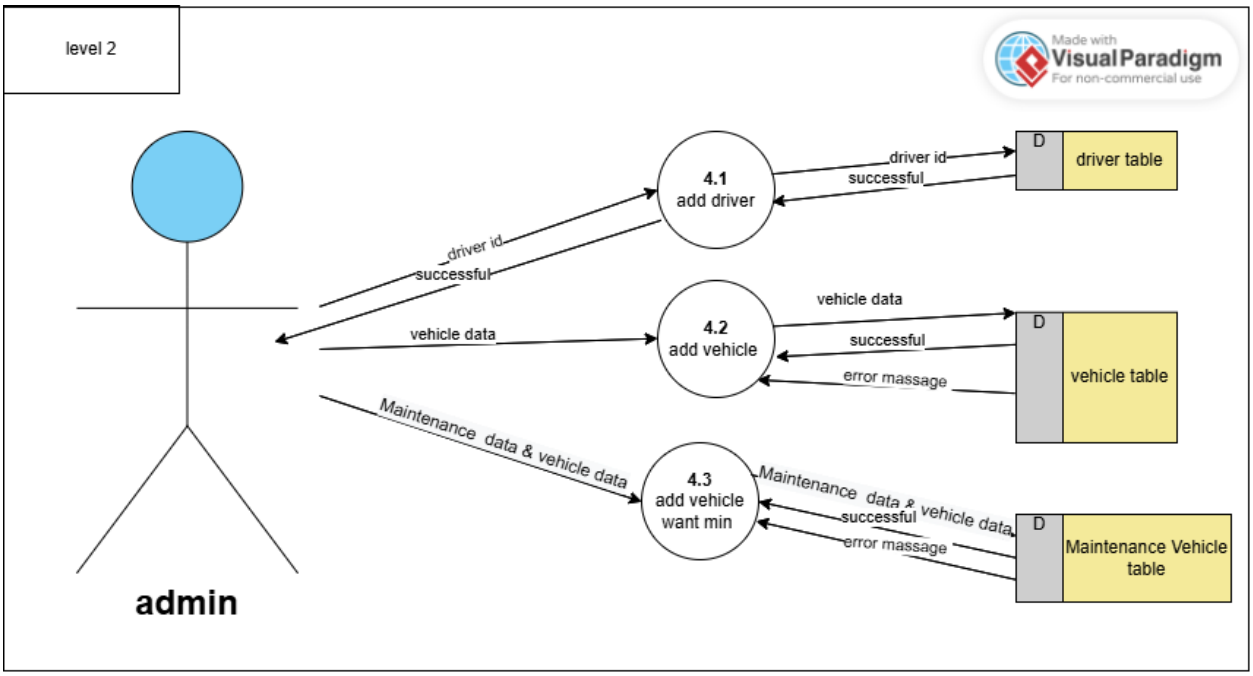
DFD Payment Management Level 2



DFD Feedback Management Level 2



DFD Fleet Management Level 2



Fibonacci-Based Rough Use Case Estimations

Using the Fibonacci Method for relative complexity estimates:

1. **Account Management**: **3** (Low complexity).
2. **Fleet Management**: **8** (Moderate complexity with dependencies).
3. **Renting Management**: **13** (High complexity due to user interaction and availability checks).
4. **Booking Management**: **5** (Low complexity; focuses on interactions and payment).
5. **Scheduling Management**: **8** (Moderate complexity due to conflict resolution).
6. **Payment Management**: **5** (Medium complexity; requires secure integrations).
7. **Feedback Management**: **3** (Low complexity).

1. Account Management: 3

- **Explanation**:
  - Account management typically involves basic CRUD (Create, Read, Update, Delete) operations for user accounts.

- Complexity is considered low as these are well-defined, common tasks with minimal dependencies.

2. Fleet Management: 8

- **Explanation**:
  - Fleet management includes handling vehicle inventories, tracking availability, scheduling maintenance, and managing operational statuses.
  - It is moderately complex due to dependencies on external data sources (e.g., maintenance schedules) and the need for synchronization between different entities.

3. Renting Management: 13

- **Explanation**:
  - Renting management requires managing user interactions (e.g., selecting rental items, durations) and performing availability checks against a dynamic inventory.
  - High complexity arises due to real-time updates, multiple user interactions, and potential edge cases (e.g., double bookings).

4. Booking Management: 5

- **Explanation**:
  - Booking management focuses on reserving resources or services, managing interactions, and processing payments.
  - It has a lower complexity because it primarily deals with straightforward user input, resource allocation, and transactional workflows.

5. Scheduling Management: 8

- **Explanation**:
  - Scheduling management involves resolving conflicts between overlapping bookings or schedules, optimizing resource allocation, and ensuring time-sensitive operations.
  - This adds moderate complexity due to the need for algorithms to handle scheduling conflicts and edge cases.

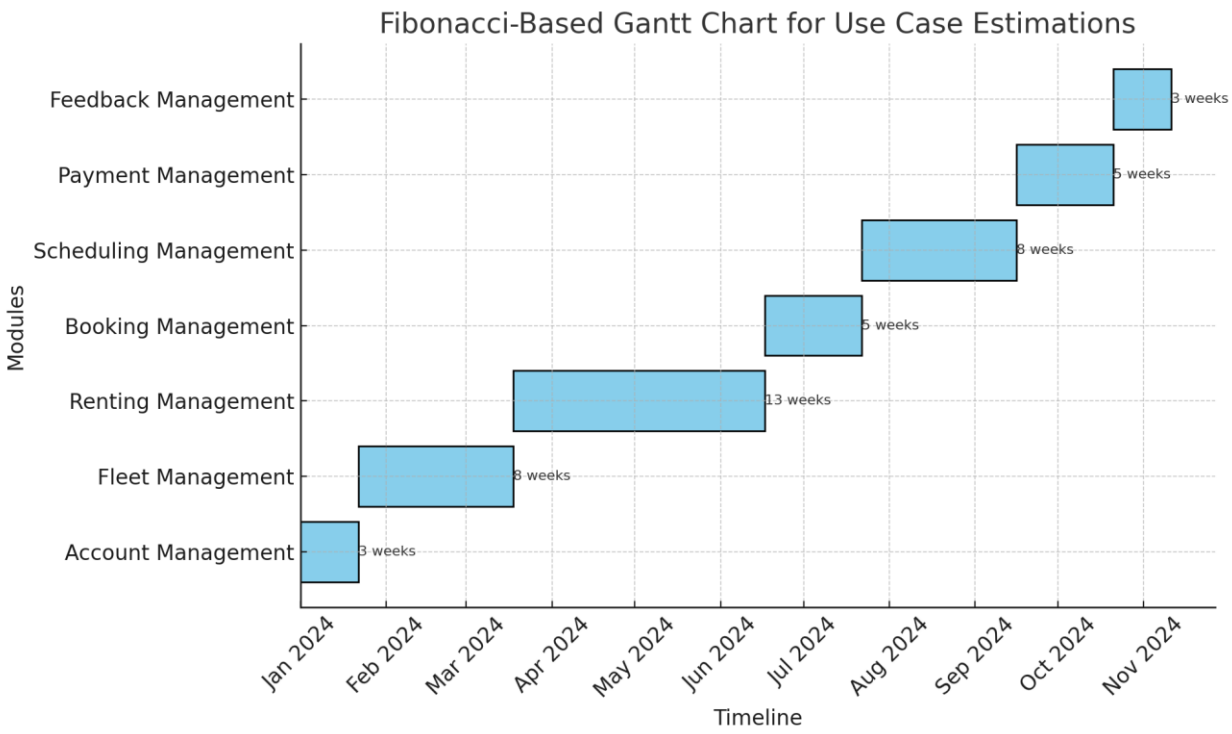
6. Payment Management: 5

- **Explanation**:
  - Payment management includes handling secure payment gateways, processing transactions, and integrating with external payment systems.
  - While slightly more complex than basic CRUD operations, it's manageable due to the availability of standardized APIs and libraries for secure payment processing.

7. Feedback Management: 3

- **Explanation**:
  - Feedback management generally involves collecting user feedback, storing it, and displaying it.
  - Low complexity as it primarily involves straightforward form handling and database operations with minimal dependencies.

Gantt Chart:



Test Plan:

1. UT(Account Management)
2. UT(Route Optimization)
3. UT(Fleet Management)
4. UT(Payment Management)
5. UT(Scheduling Management)
6. UT(Customer Engagement)
7. IT(Account Management, Route Optimization)
8. RT(Account Management)
9. RT(Account Management, Route Optimization)

10. IT(Account Management, Route Optimization, Fleet Management)
11. RT(Account Management)
12. RT(Route Optimization)
13. RT(Account Management, Route Optimization)
14. RT(Fleet Management)
15. RT(Account Management, Route Optimization, Fleet Management)
16. IT(Account Management, Route Optimization, Fleet Management, Payment Management)
17. RT(Account Management)
18. RT(Route Optimization)
19. RT(Fleet Management)
20. RT(Payment Management)
21. RT(Account Management, Route Optimization)
22. RT(Account Management, Route Optimization, Fleet Management)
23. RT(Account Management, Route Optimization, Fleet Management, Payment Management)
24. IT(Account Management, Route Optimization, Fleet Management, Payment Management, Scheduling Management)
25. RT(Account Management)
26. RT(Route Optimization)
27. RT(Fleet Management)
28. RT(Payment Management)
29. RT(Scheduling Management)
30. RT(Account Management, Route Optimization, Fleet Management)
31. RT(Account Management, Route Optimization, Fleet Management, Payment Management)
32. RT(Account Management, Route Optimization, Fleet Management, Payment Management, Scheduling Management)
33. IT(Account Management, Route Optimization, Fleet Management, Payment Management, Scheduling Management, Customer Engagement)
34. RT(Account Management)
35. RT(Route Optimization)
36. RT(Fleet Management)
37. RT(Payment Management)
38. RT(Scheduling Management)
39. RT(Customer Engagement)
40. RT(Account Management, Route Optimization)
41. RT(Account Management, Route Optimization, Fleet Management)
42. RT(Account Management, Route Optimization, Fleet Management, Payment Management)

43. RT(Account Management, Route Optimization, Fleet Management, Payment Management, Scheduling Management)
44. RT(Account Management, Route Optimization, Fleet Management, Payment Management, Scheduling Management, Customer Engagement)

45. Release Testing

46. Performance Testing

47. Stress Testing

48. Installation Testing

49. Acceptance Testing

50. Alpha

51. Beta

Explanation:

1) Test the functionality of account creation, deletion, and updates.
2) Test the efficiency and accuracy of route planning.
3) Test the system's ability to manage vehicle assignments and status.
4) Test the payment processing, including transactions and validations.
5) Test the scheduling of resources, vehicles, and tasks.
6) Test customer interaction features such as feedback and communication.
7) Test the integration of account management with route optimization functionality.
8) Re-test account management to ensure it still works after integration.
9) Ensure the combined functionality of account management and route optimization remains intact.
10) Test the interaction between account management, route optimization, and fleet management.
11) Verify account management after further integrations.
12) Re-check route optimization functionality post-integration.
13) Ensure that account management and route optimization work well together.
14) Test fleet management to confirm no regressions after integration.
15) Verify the combined functionality of all three modules.

16)Test the integration of payment management with the other three features.
17)Ensure account management remains functional post-payment integration.
18)Verify route optimization after the addition of payment features.
19)Check fleet management functionality after integration.
20)Test payment management for regressions.
21)Revalidate account and route optimization interaction.
22)Ensure the functionality of account, route, and fleet features together.
23)Verify the integration of all four features.
24)Test the interaction of scheduling management with the other features.
25)Confirm account management functionality post-scheduling integration.
26)Ensure route optimization is unaffected by scheduling features.
27)Test fleet management for regressions.
28)Verify payment management functionality post-integration.
29)Test scheduling management to confirm no issues.
30)Ensure these three features work seamlessly together.
31)Verify the interaction of four key features.
32)Confirm the functionality of all five modules.
33)Test customer engagement integration with all other features.
34)Ensure account management functionality remains intact.
35)Verify no issues with route optimization post-integration.
36)Confirm fleet management works as expected.
37)Re-test payment management for consistency.
38)Verify scheduling management functionality.
39)Test customer engagement for regressions.
40)Revalidate the integration of account management and route optimization.
41)Ensure these three modules work correctly together.
42)Verify no issues across these four features.
43)Confirm seamless functionality of all five key features.
44)Verify the comprehensive integration of all features.
45)Verifies the readiness of the software for deployment by testing it in a production-like environment.
46)Measures how the system performs under normal and peak conditions, focusing on speed, scalability, and reliability.
47)Assesses the system's stability and robustness by pushing it beyond its normal operational limits.
48)Ensures the software installs correctly, configures as expected, and operates after installation.
49)Validates that the system meets the business requirements and is ready for end-user approval.
50)A pre-release testing phase where the software is tested internally by the development team or in a controlled environment.

51)A pre-release testing phase where the software is tested by a limited number of external users to identify any remaining issues before final release.

_____