UNIVERSITI TEKNOLOGI MALAYSIA

SECP2523: DATABASE

SEM I 2025/ 2026

DATABASE DESIGN DESCRIPTIONS: LOGICAL DATABASE DESIGN

FOR Hasta Travel & Tours Car Rental System

Group Name: QueryCrew

Members:

No.	Name	Matrix Number
1.	KAVINESH REDDY A/L GOPALAKRISHNAN (TL)	A24CS0092
2.	MUHAMMAD AL-HAKIMI HAIKAL BIN ROMI SABIHIN	A24CS0271
3.	HARITZ HAYKAL BIN NAZRUL HISHAM	A24CS0250
4.	NIVEETHITA A/P PANDIA RAJAN	A24CS0148

Stakeholder: Hasta Travel & Tours

Representative Name	Date Interviewed
1. ALIF FIRDAUS BIN JALALUDIN	4th JANUARY 2026
2. ALIF FIRDAUS BIN JALALUDIN	8th JANUARY 2026

TABLE OF CONTENTS

1.0 INTRODUCTION

1.1 Overview about the company

Hasta Travel & Tours Sdn. Bhd., also known as Car Rental UTM, is a licensed travel and car rental company based at Student Mall, Universiti Teknologi Malaysia (UTM), Skudai, Johor. It offers a wide range of vehicles, from affordable models like Perodua Axia, Bezza, and Myvi to premium options such as Toyota Alphard and Hyundai Starex, catering to both short-term and long-term rental needs. Currently, the rental process is handled manually, where customers make inquiries via WhatsApp, phone calls, or in-person visits. Staff confirm bookings by checking availability through spreadsheets or written records, while customers submit personal details and payment proof manually. On the day of rental, vehicles are collected at the office or agreed location, and after return, deposits are refunded—often taking several days. This traditional process can lead to booking errors, record-keeping issues, and delays in customer service. As the demand for car rental services grows, there is a strong need to digitalize the system to improve operational efficiency, customer satisfaction, and business scalability.

1.2 The Existing System

1.2.1 Car Inquiry and Reservation

The process begins when customers submit rental inquiries through whatsapp and walk-in. Upon receiving the inquiry, staff manually checks vehicle availability using an internal tracking system. This depends entirely on staff accuracy and real-time updates. If a car is available, customers are required to make either a deposit or full payment before the booking can be confirmed.

1.2.2 Customer Registration and Verification

Customers provide personal details such as identification cards or passports and driving licenses via WhatsApp. Staff manually verifies these documents and stores them in their internal systems. Once full payment is completed, staff generates the rental agreement and sends it to the customer. Customers then digitally agree to the terms and conditions through a signature or checkbox with a timestamp. However, this still requires manual handling and verification by staff that risks the records consistency.

1.2.3 Payment Process

Payments are accepted through QR code, online bank transfer, or cash and proof of payment i sent in the form of a receipt screenshot. Staff manually verifies each transaction and updates the payment status in the system. A centralised system for this is necessary to log all the payment logs and document the verification to prevent error.

1.2.4 Car Collection and Return

On the rental day, customers collect the vehicle at the office location. Staff prepares a checklist to document the car's condition, fuel level, and other relevant details during handover. Upon return, staff inspects the vehicle for scratches, dents, cleanliness, fuel level, and overall condition. The return inspection details are then recorded in the system. There is no proper tracking system to calculate late return penalties or to notify customers in real time which increases the risk of delays and customer disputes.

1.2.5 Reporting

Rental records, payment details, vehicle usage, and return data are compiled using a combination of the Track system and Microsoft Excel. Management reports, such as monthly sales summaries, rental history, and car utilization reports, are prepared manually at the end of each month. Compiling data from multiple sources manually is time-consuming and it could lead to data inconsistencies.

1.3 The Proposed System

The proposed system is a web-based Car Rental Management System that supports both the customer side and the admin/staff side for Hasta Travels & Tours.

For customers, UTM students and staff will be able to register and verify their accounts, browse available cars, check the weekday promotional pricing, and make bookings using the deposit-first method. Customers can also upload before and after rental photos, sign the rental agreement online during pickup, make the full payment when they collect the car, and join the loyalty programme where they can earn vouchers based on the total rental hours. Besides that, customers can request refunds, reschedule bookings, and view any penalties such as late return fees or fuel charges.

For the admin and staff side, the system allows them to manage the car listings, update mileage and fuel level every day, record maintenance details, set the available drop-off locations, and monitor bookings without needing manual approval for each one. Staff can also carry out inspections with before/after photos, record penalties, and let the system automatically change the car status when the mileage passes the service limit. The system also provides data such as booking frequency by college, customer activity, and total revenue from rentals and penalties. There is also an activity calendar where the company can post updates or announcements.

The system is fully online and does not require walk-in customers. All bookings must be made through the website. All payments are uploaded manually through receipts, without any integration to external banking systems. Access to the system is controlled, where only verified UTM students and staff can rent cars, while blacklisted customers will be blocked from logging in until they settle their outstanding payments.

2.0 OVERVIEW OF THE PROJECT

The proposed system is a web-based Car Rental Management System that supports both the customer side and the admin/staff side for Hasta Travels & Tours.

For customers, UTM students and staff will be able to register and verify their accounts, browse available cars, check the weekday promotional pricing, and make bookings using the deposit-first method. Customers can also upload before and after rental photos, sign the rental agreement online during pickup, make the full payment when they collect the car, and join the loyalty programme where they can earn vouchers based on the total rental hours. Besides that, customers can request refunds, reschedule bookings, and view any penalties such as late return fees or fuel charges.

For the admin and staff side, the system allows them to manage the car listings, update mileage and fuel level every day, record maintenance details, set the available drop-off locations, and monitor bookings without needing manual approval for each one. Staff can also carry out inspections with before/after photos, record penalties, and let the system automatically change the car status when the mileage passes the service limit. The system also provides data such as booking frequency by college, customer activity, and total revenue from rentals and penalties. There is also an activity calendar where the company can post updates or announcements.

The system is fully online and does not require walk-in customers. All bookings must be made through the website. All payments are uploaded manually through receipts, without any integration to external banking systems. Access to the system is controlled, where only verified UTM students and staff can rent cars, while blacklisted customers will be blocked from logging in until they settle their outstanding payments.

3.0 DATABASE CONCEPTUAL DESIGN

3.1 Updated Business Rule

This section defines the business rules that determine how data is created, updated and managed in the proposed Hasta Travel & Tours Car Rental Management System. These rules ensure operations are consistent, prevent invalid transactions and automation on customer support.

3.1.1 Customer Registration and Verification

1. A customer must register an account before making a booking.
2. Each customer must have a unique Customer ID.
3. Customers must provide valid documents such as Identification Card or Passport, Matrics Card and Driving License.
4. Customer verification status will be recorded as either Pending, Approved or Rejected
5. Only customers with Approved verification status are allowed to make booking
6. Customers who are blacklisted are not allowed to create bookings until their blacklist status is removed by admin.

3.1.2 Vehicle Management

1. Each vehicle must have a unique Car ID and a unique Car Plate Number.
2. Vehicle status must be controlled by admin as Available, In Use and Maintenance
3. A vehicle can only be booked if its status is Available for the selected time.
4. A vehicle cannot be assignment to more than one booking in the same date and time
5. Vehicles that exceed the service mileage should be flagged by the system and set the status to Maintenance until resolved.

3.1.3 Booking

1. Each booking must have a unique Booking ID.
2. A booking must be created only by one verified customer.
3. A booking must be linked to only one vehicle
4. Booking must have a valid rental period.

5. The system should automatically change status to In Use once a customer has picked up the vehicle, and to Maintenance when the service mileage has exceeded.
6. A booking can only proceed to vehicle collection after payment is verified, signed agreement and necessary photo documents of the car are submitted.

3.1.4 Payment, Deposit and Refund

1. Payment types supported by the system are QR payment, Bank Transfer, and Cash
2. Payment receipt evidence must be recorded in the system
3. Deposit refunds should be processed after booking completion provided there are no outstanding penalties

3.1.5 Penalty Rules

1. Penalty records shall be listed to a booking and categorised as Late Return, Summon, Fuel Shortage, Damage or Cleanliness.
2. Customers with unpaid penalties may be blacklisted by the admin until the penalty is paid.

3.1.6 Car Collection, Return and Inspection

1. Customer should upload pictures of cars on all sides, signed agreement, fuel level before car collection
2. Customer should upload pictures of cars on all sides and fuel level after car collection
3. After car return and inspection done by the customer side, the system shall update vehicle status as Available unless maintenance is required.

3.1.7 Loyalty and Voucher

1. The system shall track loyalty stamps based on rental duration where 1 stamp is awarded for every 9 rental hours.
2. Loyalty stamps and voucher eligibility shall be stored under the customer's account
3. Vouchers generated by admin must have a unique voucher code and expiry date.

4. Customers may redeem vouchers during booking payment and the system should record voucher usage to prevent reuse

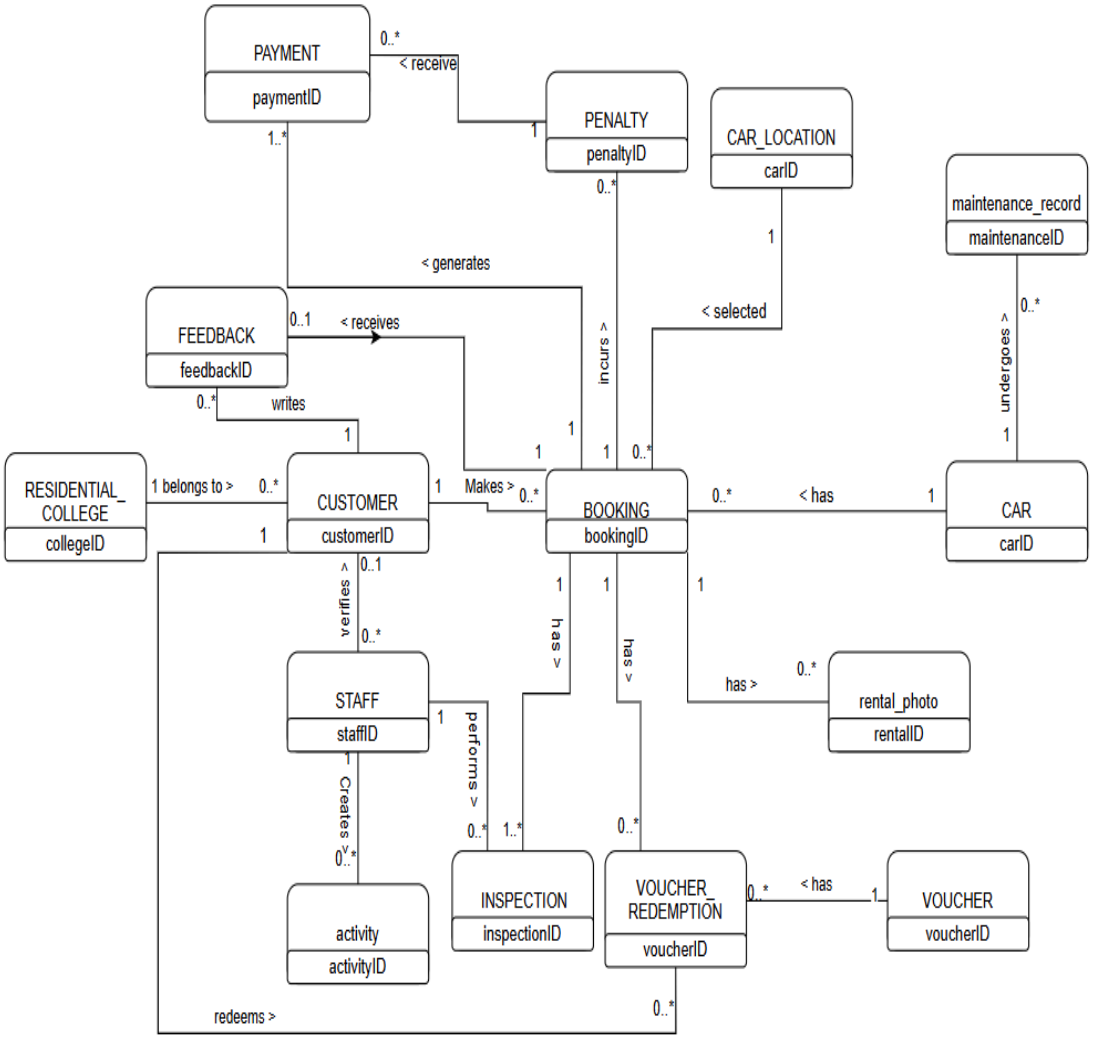
3.1.8 Feedback and Issue Handling

1. Customers may submit feedback only for bookings that have status Completed
2. Vehicle issues reported in feedback shall be linked to the respective vehicle for tracking
3. Admin actions taken on issues must be recorded to support accountability and follow-up.

3.1.9 Reporting

1. Only admin are allowed to manage vehicles, verify payments, process refunds, and view reports and analytics
2. The system shall generate reports on bookings, revenue, vehicle ulitisation and customer activity
3. Reports can be filtered by period to view the analytics.

3.2 Global Conceptual Database Design (Conceptual ERD)



3.3 Data Dictionary for the global conceptual ERD.

Entity Name	Description
Residential College	Represents a residential college within the university that customers (students) may belong to.
Staff	Represents staff members responsible for system operations such as verification, inspections, and activity management.
Customer	Represents users of the car rental system, including students and staff.
Car	Represents vehicles available for rental in the system.
Car Location	Represents locations where vehicles can be picked up and returned.
Booking	Represents a reservation made by a customer to rent a car for a specific period.
Payment	Represents financial transactions related to a booking, including deposits, rentals, refunds, and penalties.
Penalty	Represents charges imposed due to late return, damage, fuel issues, or other violations.
Inspection	Represents inspections conducted during vehicle pickup or return.
rental_photo	Represents photographic records related to a booking.
Maintenance_ Record	Represents maintenance activities performed on a vehicle.
Voucher	Represents discount vouchers available for use by customers.
Voucher_ Redemption	Represents the redemption of a voucher by a customer for a booking.
Activity	Represents activities or announcements created by staff.
Feedback	Represents feedback provided by customers regarding their booking experience.

4.0 DATABASE LOGICAL DESIGN

4.1 Logical ERD

4.2 Updated Data Dictionary

4.2.1.Customer

Attribute	Description
customer_id	Unique identifier for customer
full_name	Customer full name
email	Customer email address
phone	Contact number
password_hash	Encrypted password
ic_url	Identity card document
utmid_url	University ID document
license_url	Driving license document
utm_role	Customer role (student/staff)
college_id	Linked residential college
verification_status	Verification state
verified_by	Staff who verified
verified_at	Verification timestamp
is_blacklisted	Blacklist indicator
blacklist_reason	Reason for blacklist
blacklist_since	Blacklist date
deposit_balance	Current deposit balance
total_rental_hours	Total hours rented
total_stamps	Loyalty stamps
created_at	Account creation time

4.2.2. Staff

Attribute	Description
staff_id	Unique staff identifier
name	Staff name
email	Staff email
password_hash	Encrypted password
role	Staff role
created_at	Record creation time

4.2.3. Car

Attribute	Description
car_id	Unique car identifier
plate_number	Vehicle plate number
brand	Car brand
model	Car model
fuel_type	Fuel type
year	Manufacturing year
base_rate_per_hour	Rental rate
status	Availability status
current_mileage	Current mileage
service_mileage_limit	Service threshold
last_service_date	Last service date
image_url	Car image
gps_enabled	GPS availability
created_at	Record creation time

4.2.4. Booking

Attribute	Description
booking_id	Booking identifier
customer_id	Customer reference
car_id	Car reference
pickup_location_id	Pickup location
dropoff_location_id	Drop-off location
start_datetime	Rental start
end_datetime	Rental end
rental_hours	Total hours
base_price	Base price
promo_discount	Promotional discount
voucher_discount	Voucher discount
total_rental_amount	Rental amount
deposit_amount	Deposit paid
deposit_used_amount	Deposit used
deposit_refund_amount	Refund amount
deposit_decision	Deposit outcome
agreement_signed_at	Agreement time
final_amount	Final charge
status	Booking status
created_at	Booking creation time

4.2.5. Payment

Attribute	Description
payment_id	Payment identifier
booking_id	Related booking
penalty_id	Related penalty
amount	Payment amount
payment_type	Type of payment
payment_method	Payment method
receipt_url	Receipt document
payment_date	Payment date
status	Verification status
created_at	Record creation time

4.2.6. Penalty

Attribute	Description
penalty_id	Penalty identifier
booking_id	Related booking
penalty_type	Penalty category
description	Penalty details
amount	Penalty amount
is_installment	Installment flag
status	Penalty status
created_at	Record creation time

4.2.7. Inspection

Attribute	Description
inspection_id	Inspection identifier
booking_id	Related booking
inspection_type	Pickup/Return
datetime	Inspection time
fuel_level	Fuel level
odometer_reading	Mileage
notes	Remarks
inspected_by	Staff reference

4.2.8. Voucher

Attribute	Description
voucher_id	Voucher identifier
code	Voucher code
description	Voucher description
discount_type	Discount method
discount_value	Discount value
min_stamps_required	Required stamps
expiry_date	Expiry date
is_active	Voucher availability

4.2.9. Voucher_Redemption

Attribute	Description
redemption_id	Redemption identifier
voucher_id	Voucher reference

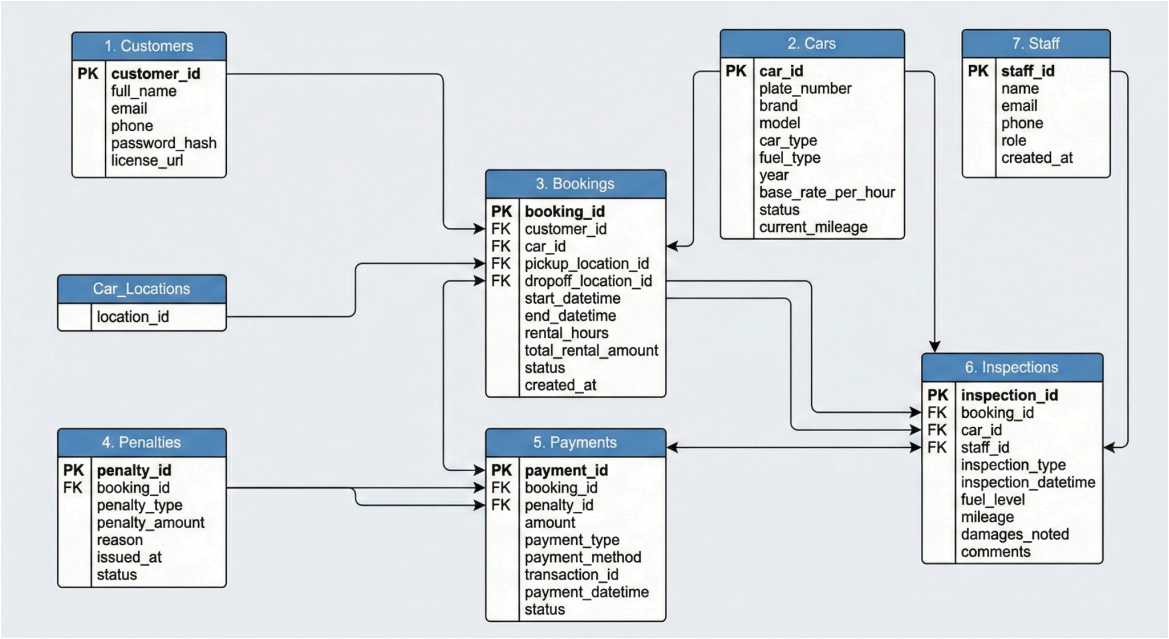
customer_id	Customer reference
booking_id	Booking reference
discount_amount	Applied discount
redeemed_at	Redemption time

4.2.10. Feedback

Attribute	Description
feedback_id	Feedback identifier
booking_id	Booking reference
customer_id	Customer reference
rating	Customer rating
comment	Feedback text
reported_issue	Issue indicator
created_at	Feedback time

4.3 Relational Database Schema

This diagram shows a relational database schema for a car rental system using seven interconnected tables. The central Bookings table links Customers to the specific Cars they rent. There are also other tables such as recording Payments and issuing Penalties. The schema also tracks vehicle condition through an Inspections table, which connects cars to the Staff members who check them.



4.4 Normalization (Up to Third Normal Form – 3NF)

This section presents a complete and systematic normalization process for the Hasta Travel & Tours Car Rental System. The normalization is clearly illustrated from Unnormalized Form (UNF) to First Normal Form (1NF), Second Normal Form (2NF), and finally Third Normal Form (3NF). Each step explicitly shows the transformation of relations, identification of dependencies, and justification of the resulting normal form.

4.4.1 Unnormalized Form (UNF)

For normalization purposes, it is assumed that all operational data is initially stored in one single unnormalized table. This table combines data related to customers, bookings, cars, locations, payments, penalties, inspections, vouchers, photos, and feedback.

CAR_RENTAL_UNF(

booking_id,

customer_id, full_name, email, phone, utm_role, college_name,

car_id, plate_number, brand, model, fuel_type, year, base_rate_per_hour,

pickup_location_name, dropoff_locati on_name,

start_datetime, end_datetime, rental_hours,

base_price, promo_discount, voucher_code, voucher_discount,

total_rental_amount,

deposit_amount, deposit_used_amount, deposit_refund_amount, deposit_decision,

payment_list { payment_id, payment_type, payment_method, receipt_url, payment_date, payment_status },

penalty_list { penalty_id, penalty_type, amount, status },

inspection_list { inspection_id, inspection_type, datetime, fuel_level, odometer_reading, notes, inspected_by },

photo_list { photo_id, uploaded_by_user_id, uploaded_by_role, photo_type, photo_url, taken_at },

feedback { feedback_id, rating, comment, reported_issue }

)

Problems in UNF:

- Presence of repeating groups (payment_list, penalty_list, inspection_list, photo_list)
- Attributes are not atomic
- High data redundancy
- Update, insert, and delete anomalies

Therefore, the relation is not in First Normal Form (1NF).

4.4.2 First Normal Form (1NF)

Rule (1NF):

- All attributes must contain atomic values
- No repeating groups are allowed

Transformation (UNF → 1NF):
To achieve 1NF, all repeating groups are removed and each row is restructured to contain only single-valued attributes. Repeated entities such as payments, penalties, inspections, photos, and feedback are flattened so that each row represents one booking with one related occurrence.

CAR_RENTAL_1NF(

booking_id,
customer_id, full_name, email, phone, utm_role, college_id,
car_id, plate_number, brand, model, fuel_type, year, base_rate_per_hour,
pickup_location_id, dropoff_location_id,
start_datetime, end_datetime, rental_hours,
base_price, promo_discount, voucher_discount, total_rental_amount,
deposit_amount, deposit_used_amount, deposit_refund_amount, deposit_decision,
payment_id, payment_type, payment_method, receipt_url, payment_date, payment_status,
penalty_id, penalty_type, penalty_amount, penalty_status,

inspection_id, inspection_type, inspection_datetime, fuel_level, odometer_reading,
photo_id, uploaded_by_user_id, uploaded_by_role, photo_type, photo_url, taken_at,
feedback_id, rating, comment, reported_issue

)

Candidate Key (1NF):

(booking_id, payment_id, penalty_id)

All attributes are now atomic and the relation satisfies First Normal Form (1NF).

4.4.3 Second Normal Form (2NF)

Rule (2NF):
A relation is in 2NF if:

1. It is already in 1NF, and
2. It contains no partial dependency on a composite primary key.

Identification of Partial Dependencies

In **CAR_RENTAL_1NF**, several non-key attributes depend only on part of the composite key:

- Customer attributes depend only on **customer_id**
- Car attributes depend only on **car_id**
- College attributes depend only on **college_id**
- Staff attributes depend only on **staff_id**
- Location attributes depend only on **location_id**
- Payment attributes depend only on **payment_id**
- Penalty attributes depend only on **penalty_id**

- Inspection attributes depend only on **inspection_id**
- Photo attributes depend only on **photo_id**
- Feedback attributes depend only on **feedback_id**

These partial dependencies violate 2NF.

Transformation (1NF → 2NF)

To remove partial dependencies, the 1NF relation is decomposed into separate relations where each non-key attribute depends on the whole primary key of its relation.

Relations after 2NF decomposition:

- residential_college(college_id PK, name, code)
- staff(staff_id PK, name, email, password_hash, role, created_at)
- customer(customer_id PK, full_name, email, phone, password_hash, ic_url, utmid_url, license_url, utm_role, college_id FK, verification_status, verified_by FK, verified_at, is_blacklisted, blacklist_reason, blacklist_since, deposit_balance, total_rental_hours, total_stamps, created_at)
- car(car_id PK, plate_number, brand, model, fuel_type, year, base_rate_per_hour, status, current_mileage, service_mileage_limit, last_service_date, image_url, gps_enabled, created_at)
- car_location(location_id PK, name, type)
- booking(booking_id PK, customer_id FK, car_id FK, pickup_location_id FK, dropoff_location_id FK, start_datetime, end_datetime, rental_hours, base_price, promo_discount, voucher_discount, total_rental_amount, deposit_amount, deposit_used_amount, deposit_refund_amount, deposit_decision, agreement_signed_at, final_amount, status, created_at)
- payment(payment_id PK, booking_id FK, penalty_id FK NULL, amount, payment_type, payment_method, receipt_url, payment_date, status, created_at)
- penalty(penalty_id PK, booking_id FK, penalty_type, description, amount, is_installment, status, created_at)

- inspection(inspection_id PK, booking_id FK, inspection_type, inspection_datetime, fuel_level, odometer_reading, notes, inspected_by FK)
- rental_photo(photo_id PK, booking_id FK, uploaded_by_user_id, uploaded_by_role, photo_type, photo_url, taken_at)
- maintenance_record(maintenance_id PK, car_id FK, service_date, description, mileage_at_service, cost)
- voucher(voucher_id PK, code, description, discount_type, discount_value, min_stamps_required, expiry_date, is_active)
- voucher_redemption(redemption_id PK, voucher_id FK, customer_id FK, booking_id FK UNIQUE, discount_amount, redeemed_at)
- activity(activity_id PK, title, description, start_date, end_date, created_by FK, created_at)
- feedback(feedback_id PK, booking_id FK UNIQUE, customer_id FK, rating, comment, reported_issue, created_at)

After decomposition, all partial dependencies are removed. Therefore, the relations satisfy Second Normal Form (2NF).

4.4.4 Third Normal Form (3NF)

Rule (3NF):
A relation is in 3NF if:

1. It is already in 2NF, and
2. There is no transitive dependency, where a non-key attribute depends on another non-key attribute.

Identification of Transitive Dependencies (from 2NF stage)

Although the relations produced in 2NF are mostly well-structured, there are conceptual transitive dependencies that must be addressed to clearly demonstrate the transition to 3NF.

Example 1: Customer and Residential College

In the 2NF stage, customer-related data conceptually contains college descriptive information:

customer(

customer_id PK,
full_name, email, phone, utm_role,
college_id,
college_name,
college_code

)

Here:

customer_id → college_id

college_id → college_name, college_code

This forms a transitive dependency, where college_name and college_code depend on customer_id through college_id.

Transformation (2NF → 3NF)

To remove this transitive dependency, college descriptive attributes are moved into a separate relation.

residential_college(college_id PK, name, code)

customer(customer_id PK, ..., college_id FK)

After this decomposition, all non-key attributes in **customer** depend directly on **customer_id**.

Example 2: Staff Verification Details

At the 2NF stage, verification-related attributes may conceptually appear as:

customer(

customer_id PK,
full_name, email,
verified_by,
staff_name, staff_role

)

Here:

customer_id → verified_by

verified_by → staff_name, staff_role

This is also a transitive dependency.

Transformation (2NF → 3NF)

Staff descriptive attributes are separated into their own relation:

staff(staff_id PK, name, email, role, created_at)

customer(customer_id PK, ..., verified_by FK)

Example 3: Voucher and Voucher Redemption

In the 2NF stage, voucher information may conceptually appear together with redemption details:

voucher_redemption(

redemption_id PK,
booking_id,
voucher_code,
discount_type,
discount_value,
discount_amount

)

Here:

redemption_id → voucher_code
voucher_code → discount_type, discount_value

This creates a transitive dependency through voucher information.

Transformation (2NF → 3NF)

Voucher descriptive attributes are separated into a voucher relation, voucher_redemption only keeps the foreign key reference:

voucher(voucher_id PK, code, description, discount_type, discount_value, min_stamps_required, expiry_date, is_active)

voucher_redemption(redemption_id PK, voucher_id FK, customer_id FK, booking_id FK, discount_amount, redeemed_at)

Final Normalized Relations (3NF)

After removing all identified transitive dependencies, the final relations are:

- residential_college(college_id PK, name, code)
- staff(staff_id PK, name, email, password_hash, role, created_at)
- customer(customer_id PK, full_name, email, phone, password_hash, ic_url, utmid_url, license_url, utm_role, college_id FK, verification_status, verified_by FK, verified_at, is_blacklisted, blacklist_reason, blacklist_since, deposit_balance, total_rental_hours, total_stamps, created_at)
- car(car_id PK, plate_number, brand, model, fuel_type, year, base_rate_per_hour, status, current_mileage, service_mileage_limit, last_service_date, image_url, gps_enabled, created_at)
- car_location(location_id PK, name, type)
- booking(booking_id PK, customer_id FK, car_id FK, pickup_location_id FK, dropoff_location_id FK, start_datetime, end_datetime, rental_hours, base_price, promo_discount, voucher_discount, total_rental_amount, deposit_amount, deposit_used_amount, deposit_refund_amount, deposit_decision, agreement_signed_at, final_amount, status, created_at)
- payment(payment_id PK, booking_id FK, penalty_id FK NULL, amount, payment_type, payment_method, receipt_url, payment_date, status, created_at)
- penalty(penalty_id PK, booking_id FK, penalty_type, description, amount, is_installment, status, created_at)
- inspection(inspection_id PK, booking_id FK, inspection_type, datetime, fuel_level, odometer_reading, notes, inspected_by FK)
- rental_photo(photo_id PK, booking_id FK, uploaded_by_user_id, uploaded_by_role, photo_type, photo_url, taken_at)
- maintenance_record(maintenance_id PK, car_id FK, service_date, description, mileage_at_service, cost)

- voucher(voucher_id PK, code, description, discount_type, discount_value, min_stamps_required, expiry_date, is_active)
- voucher_redemption(redemption_id PK, voucher_id FK, customer_id FK, booking_id FK UNIQUE, discount_amount, redeemed_at)
- activity(activity_id PK, title, description, start_date, end_date, created_by FK, created_at)
- feedback(feedback_id PK, booking_id FK UNIQUE, customer_id FK, rating, comment, reported_issue, created_at)

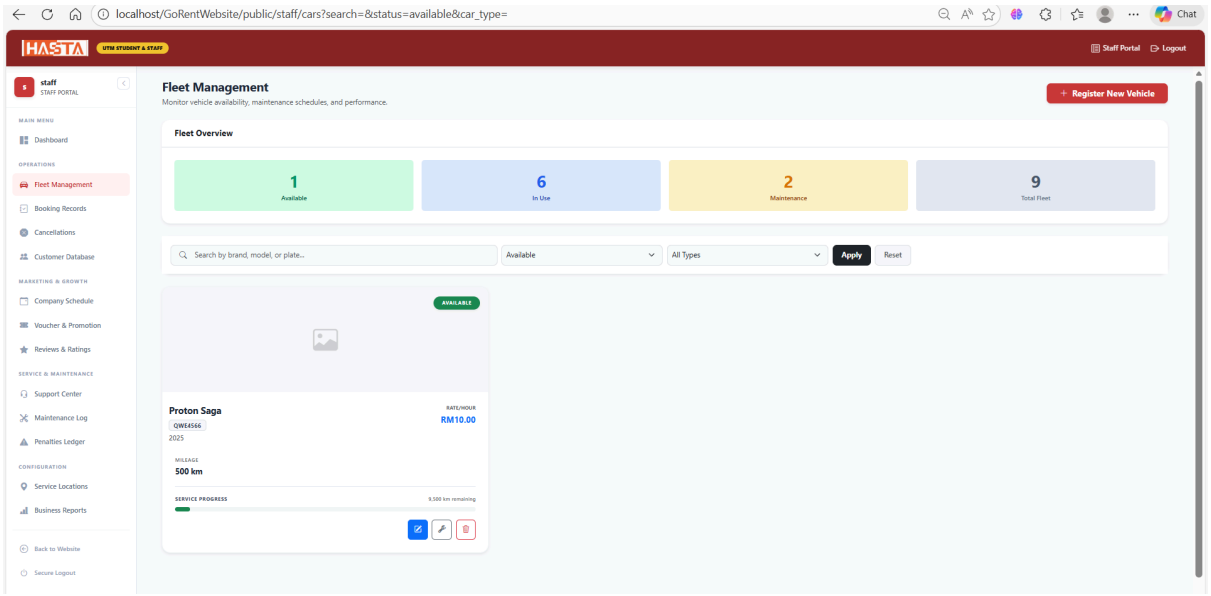
All relations now satisfy the conditions of Third Normal Form (3NF).

4.4.5 Summary

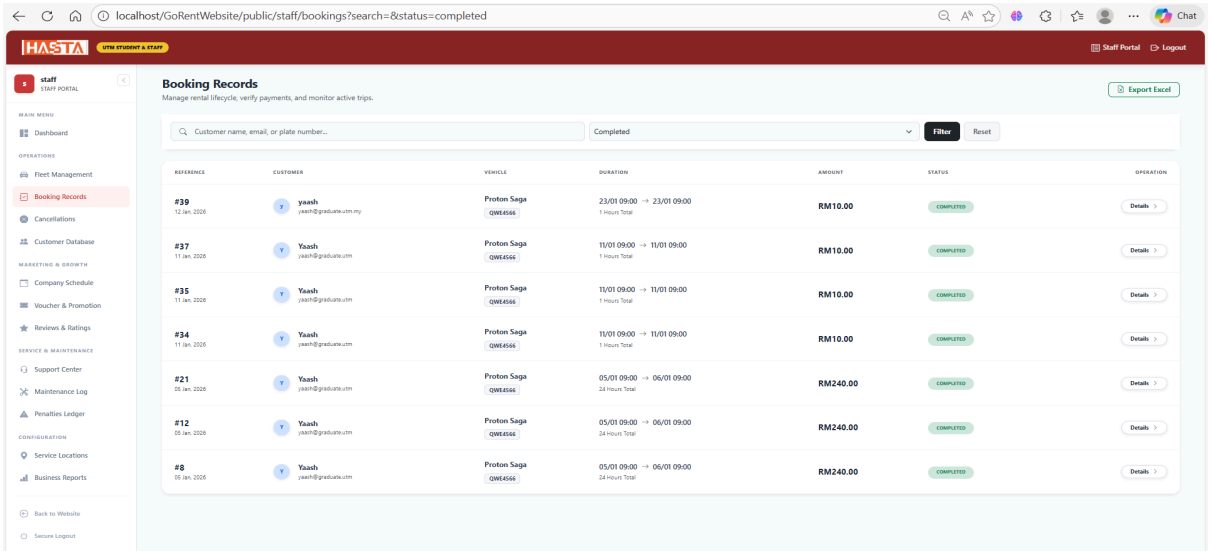
The normalization process was carried out from UNF → 1NF → 2NF → 3NF. Repeating groups were eliminated, partial dependencies were removed, and transitive dependencies were avoided. The resulting set of relations is consistent with the logical ERD and implemented SQL schema, ensuring minimal redundancy, improved data integrity, and compliance with normalization principles.

5.0 INTERFACE DESIGN & SQL IMPLEMENTATION

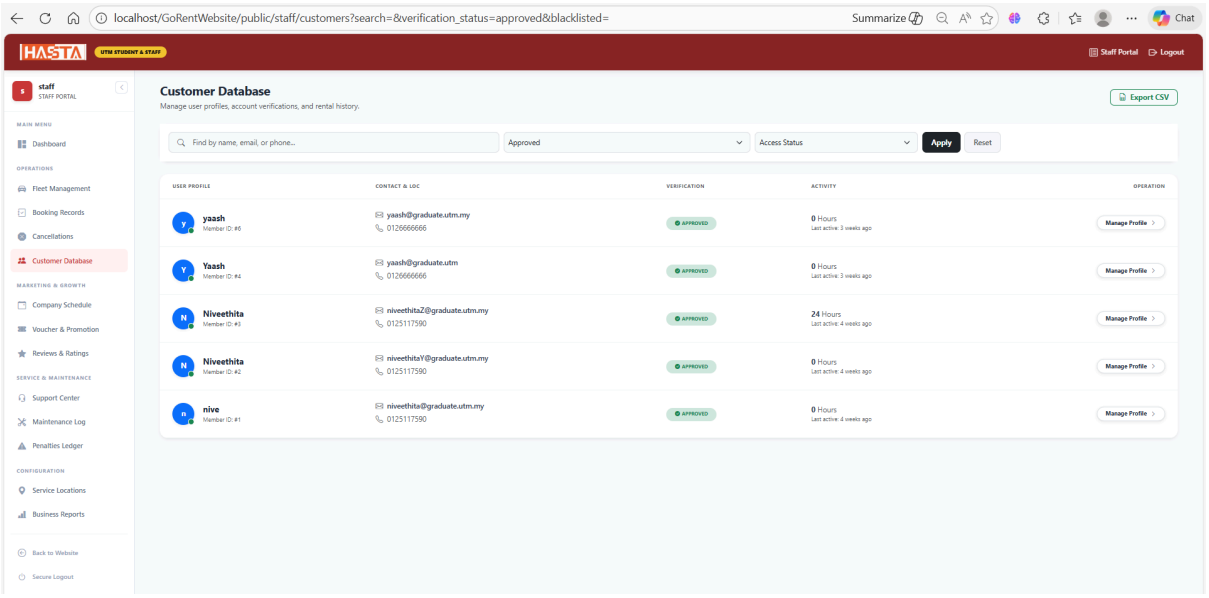
5.1 Interface Design



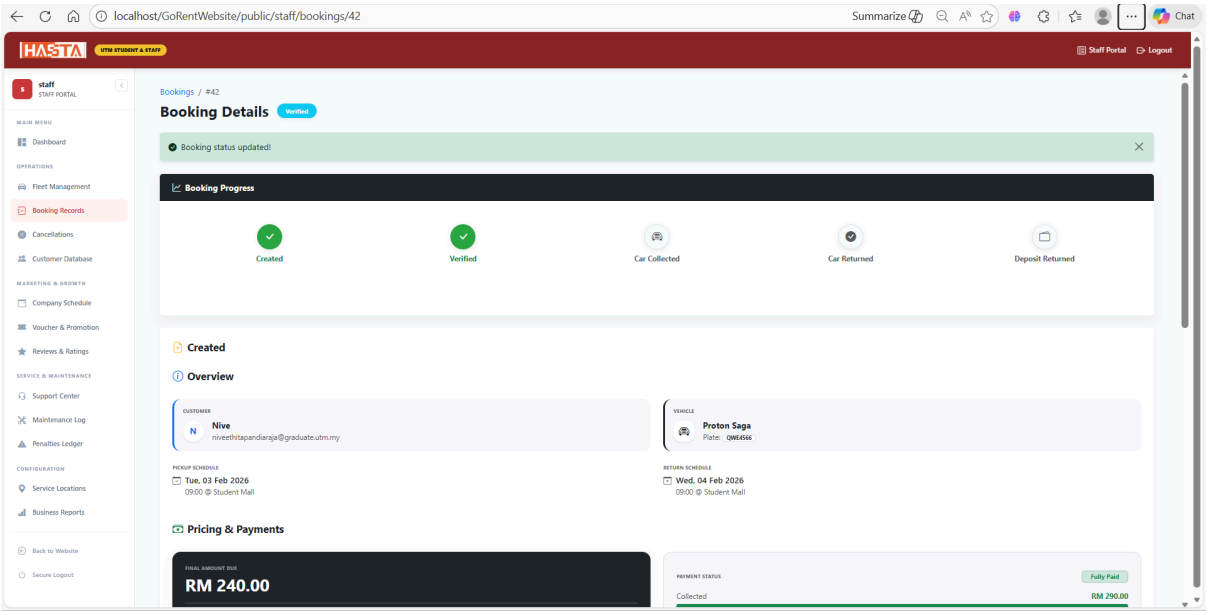
The diagram illustrates the SQL statement (Basic SELECT) involved in selecting all cars that in the status = “available”



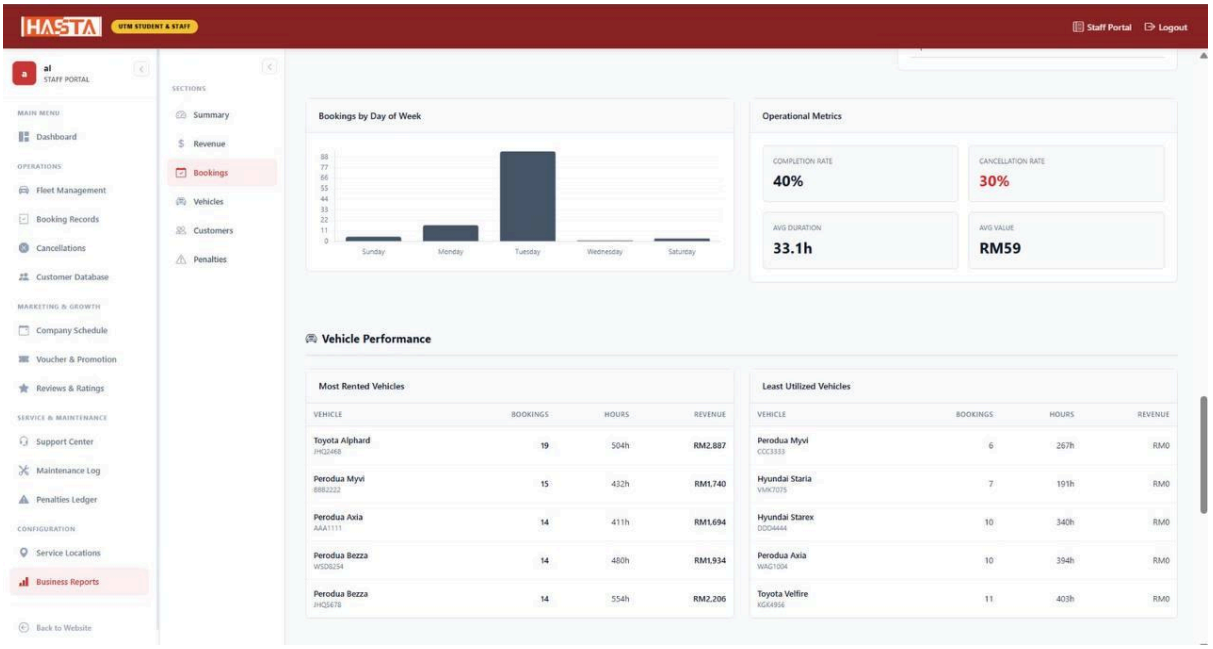
The diagram illustrates the SQL statement (INNER JOIN) involved in displaying data from three tables joined together which are bookings, customer and cars in one view



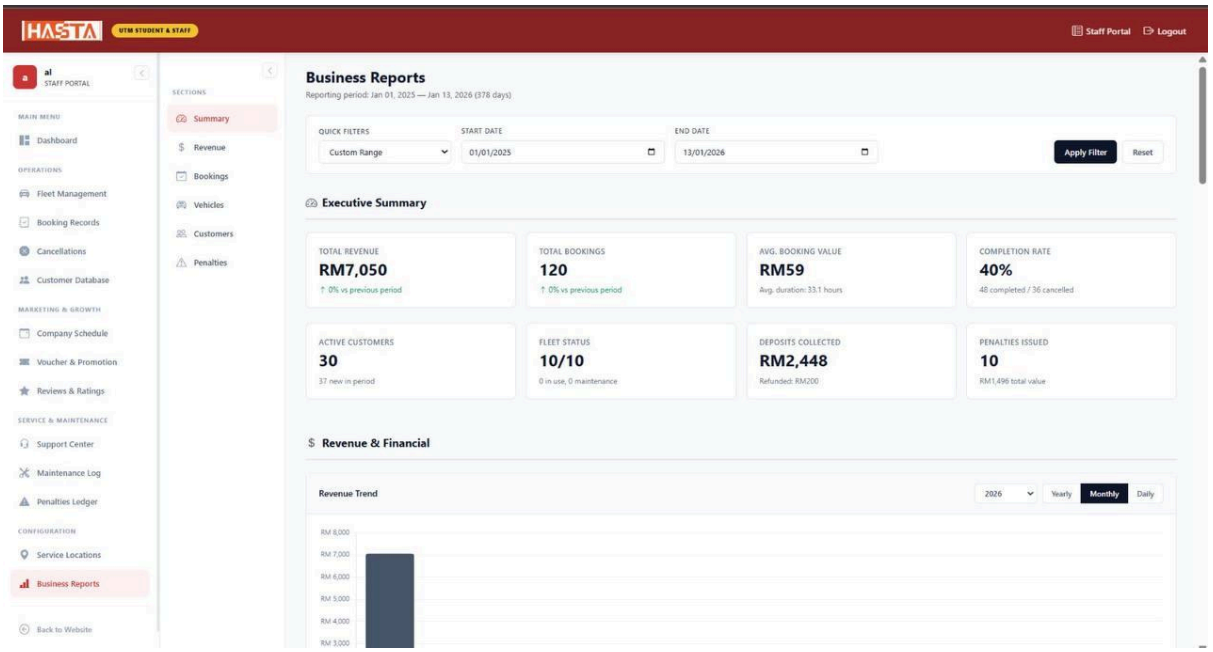
The diagram shows the SQL statement (LEFT JOIN) that displays all approved customer including those with no booking (NULL)



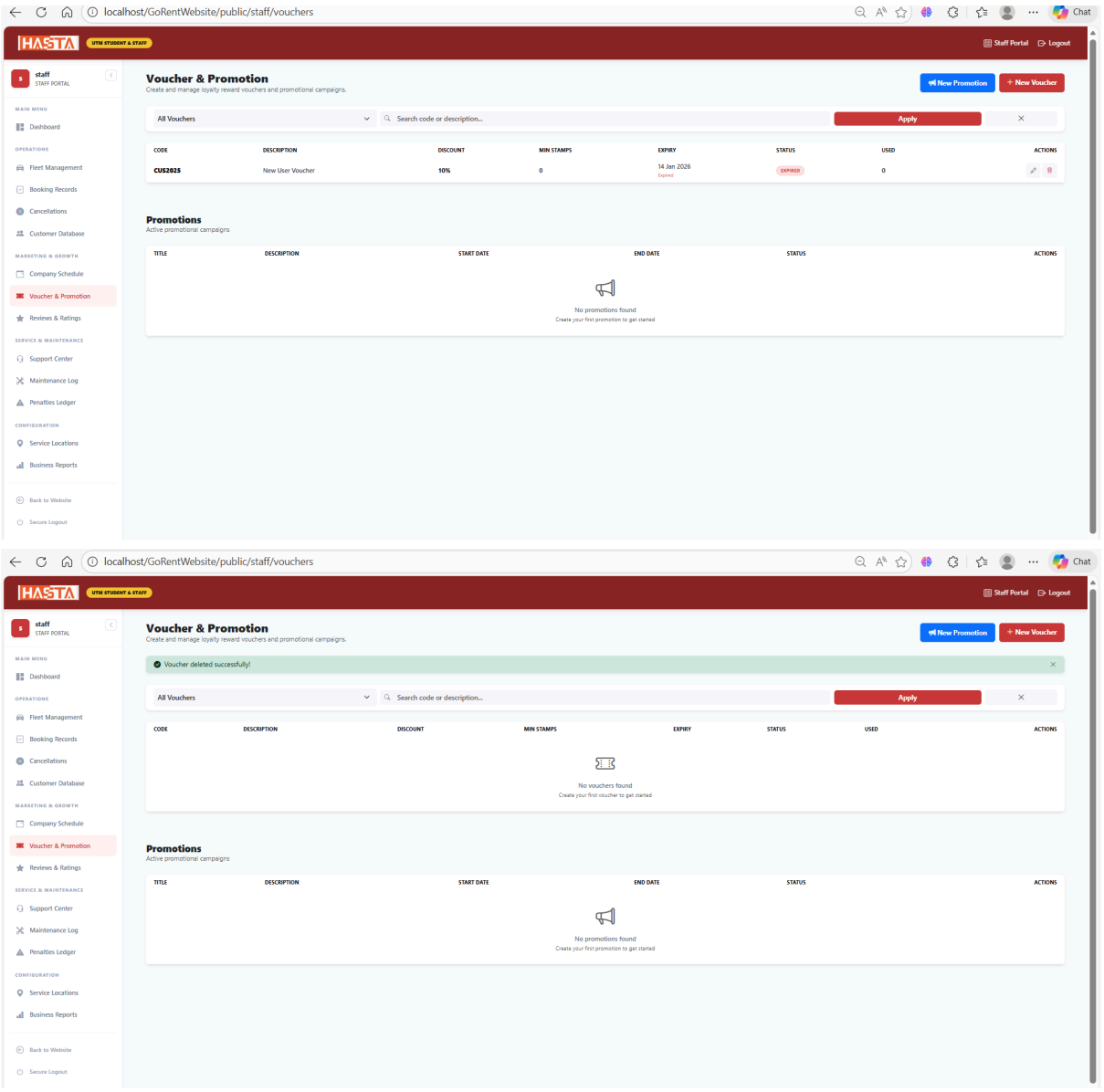
The diagram shows the implementation of SQL statement (UPDATE) involved changing status from pending to verified upon clicking on “Verify” button



The diagram shows the SQL statement (Aggregate Functions & GROUP BY) involved grouping the cars with same brand and model and the sum of bookings for each car model.



The diagram illustrates the SQL statement (CASE) involved in conditional categorization for counting different booking statues



The diagrams illustrates the SQL Statement (DELETE) involved in deleting a voucher that has expired upon clicking the delete button.

5.2 SQL statement

5.2.1 Data Definition Language (DDL)

```
CREATE TABLE residential_colleges (
    college_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    code VARCHAR(10) NOT NULL,
    created_at TIMESTAMP,
    updated_at TIMESTAMP,
    UNIQUE (code)
);

CREATE TABLE staff (
    staff_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL,
    role VARCHAR(20) DEFAULT 'staff',
    created_at TIMESTAMP,
    updated_at TIMESTAMP
);

CREATE TABLE customers (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    full_name VARCHAR(100),
    email VARCHAR(100) NOT NULL UNIQUE,
    phone VARCHAR(20),
    password_hash VARCHAR(255) NOT NULL,
    ic_url VARCHAR(255),
    utmid_url VARCHAR(255),
    license_url VARCHAR(255),
    utm_role VARCHAR(20),
    college_id INT,
    verification_status VARCHAR(20) DEFAULT 'pending',
    verified_by INT,
    verified_at TIMESTAMP,
    is_blacklisted BOOLEAN DEFAULT FALSE,
    blacklist_reason TEXT,
    blacklist_since TIMESTAMP,
    deposit_balance DECIMAL(10, 2) DEFAULT 0.00,
    total_rental_hours INT DEFAULT 0,
    total_stamps INT DEFAULT 0,
    created_at TIMESTAMP,
    updated_at TIMESTAMP,
    FOREIGN KEY (college_id) REFERENCES residential_colleges(college_id),
    FOREIGN KEY (verified_by) REFERENCES staff(staff_id)
);

CREATE TABLE car_locations (
```

```
    location_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    type VARCHAR(20) NOT NULL,
    created_at TIMESTAMP,
    updated_at TIMESTAMP
);

CREATE TABLE cars (
    id INT AUTO_INCREMENT PRIMARY KEY,
    plate_number VARCHAR(20) NOT NULL UNIQUE,
    brand VARCHAR(50) NOT NULL,
    model VARCHAR(50) NOT NULL,
    car_type VARCHAR(20),
    fuel_type VARCHAR(20) NOT NULL,
    year INT NOT NULL,
    base_rate_per_hour DECIMAL(10, 2) NOT NULL,
    status VARCHAR(20) DEFAULT 'available',
    initial_mileage INT DEFAULT 0,
    current_mileage INT DEFAULT 0,
    service_mileage_limit INT NOT NULL,
    last_service_date DATE,
    image_url VARCHAR(255),
    gps_enabled BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP,
    updated_at TIMESTAMP
);

CREATE TABLE bookings (
    booking_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT NOT NULL,
    car_id INT NOT NULL,
    pickup_location_id INT NOT NULL,
    dropoff_location_id INT NOT NULL,
    start_datetime DATETIME NOT NULL,
    end_datetime DATETIME NOT NULL,
    rental_hours INT NOT NULL,
    base_price DECIMAL(10, 2) NOT NULL,
    promo_discount DECIMAL(10, 2) DEFAULT 0.00,
    voucher_discount DECIMAL(10, 2) DEFAULT 0.00,
    total_rental_amount DECIMAL(10, 2) NOT NULL,
    deposit_amount DECIMAL(10, 2) NOT NULL,
    deposit_used_amount DECIMAL(10, 2) DEFAULT 0.00,
    deposit_refund_amount DECIMAL(10, 2) DEFAULT 0.00,
    deposit_decision VARCHAR(20),
    agreement_signed_at TIMESTAMP,
    final_amount DECIMAL(10, 2) NOT NULL,
    status VARCHAR(20) DEFAULT 'created',
    created_at TIMESTAMP,
    updated_at TIMESTAMP,
```

```
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
    FOREIGN KEY (car_id) REFERENCES cars(id),
    FOREIGN KEY (pickup_location_id) REFERENCES car_locations(location_id),
    FOREIGN KEY (dropoff_location_id) REFERENCES car_locations(location_id)
);

CREATE TABLE penalties (
    penalty_id INT AUTO_INCREMENT PRIMARY KEY,
    booking_id INT NOT NULL,
    penalty_type VARCHAR(20) NOT NULL,
    description TEXT,
    amount DECIMAL(10, 2) NOT NULL,
    is_installment BOOLEAN DEFAULT FALSE,
    status VARCHAR(20) DEFAULT 'pending',
    created_at TIMESTAMP,
    updated_at TIMESTAMP,
    FOREIGN KEY (booking_id) REFERENCES bookings(booking_id)
);

CREATE TABLE payments (
    payment_id INT AUTO_INCREMENT PRIMARY KEY,
    booking_id INT NOT NULL,
    penalty_id INT,
    amount DECIMAL(10, 2) NOT NULL,
    payment_type VARCHAR(20) NOT NULL,
    payment_method VARCHAR(20) NOT NULL,
    bank_name VARCHAR(100),
    account_holder_name VARCHAR(100),
    account_number VARCHAR(50),
    receipt_url VARCHAR(255) NOT NULL,
    payment_date DATETIME NOT NULL,
    status VARCHAR(20) DEFAULT 'pending',
    created_at TIMESTAMP,
    updated_at TIMESTAMP,
    FOREIGN KEY (booking_id) REFERENCES bookings(booking_id),
    FOREIGN KEY (penalty_id) REFERENCES penalties(penalty_id)
);

CREATE TABLE inspections (
    inspection_id INT AUTO_INCREMENT PRIMARY KEY,
    booking_id INT NOT NULL,
    car_id INT,
    inspection_type VARCHAR(20) NOT NULL,
    type VARCHAR(20) DEFAULT 'before',
    status VARCHAR(20) DEFAULT 'pending',
    datetime DATETIME NOT NULL,
    fuel_level INT NOT NULL,
    odometer_reading INT,
    mileage_reading INT,
```

```
    exterior_condition TEXT,
    interior_condition TEXT,
    engine_condition TEXT,
    damages_found TEXT,
    notes TEXT,
    photos TEXT,
    inspected_by INT NOT NULL,
    inspected_at TIMESTAMP,
    created_at TIMESTAMP,
    updated_at TIMESTAMP,
    FOREIGN KEY (booking_id) REFERENCES bookings(booking_id),
    FOREIGN KEY (car_id) REFERENCES cars(id),
    FOREIGN KEY (inspected_by) REFERENCES staff(staff_id)
);

CREATE TABLE vouchers (
    voucher_id INT AUTO_INCREMENT PRIMARY KEY,
    code VARCHAR(50) NOT NULL UNIQUE,
    description TEXT,
    discount_type VARCHAR(20) NOT NULL,
    discount_value DECIMAL(10, 2) NOT NULL,
    min_stamps_required INT DEFAULT 0,
    expiry_date DATE NOT NULL,
    is_active BOOLEAN DEFAULT TRUE,
    created_at TIMESTAMP,
    updated_at TIMESTAMP
);

CREATE TABLE voucher_redemptions (
    redemption_id INT AUTO_INCREMENT PRIMARY KEY,
    voucher_id INT NOT NULL,
    customer_id INT NOT NULL,
    booking_id INT NOT NULL,
    discount_amount DECIMAL(10, 2) NOT NULL,
    redeemed_at TIMESTAMP NOT NULL,
    created_at TIMESTAMP,
    updated_at TIMESTAMP,
    UNIQUE (booking_id),
    FOREIGN KEY (voucher_id) REFERENCES vouchers(voucher_id),
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
    FOREIGN KEY (booking_id) REFERENCES bookings(booking_id)
);

CREATE TABLE feedbacks (
    feedback_id INT AUTO_INCREMENT PRIMARY KEY,
    booking_id INT NOT NULL,
    customer_id INT NOT NULL,
    rating INT,
    comment TEXT,
```

```
    reported_issue BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP,
    updated_at TIMESTAMP,
    UNIQUE (booking_id),
    FOREIGN KEY (booking_id) REFERENCES bookings(booking_id),
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);

CREATE TABLE maintenance_records (
    maintenance_id INT AUTO_INCREMENT PRIMARY KEY,
    car_id INT NOT NULL,
    service_date DATE NOT NULL,
    description TEXT,
    mileage_at_service INT NOT NULL,
    cost DECIMAL(10, 2),
    created_at TIMESTAMP,
    updated_at TIMESTAMP,
    FOREIGN KEY (car_id) REFERENCES cars(id)
);

CREATE TABLE activities (
    activity_id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    description TEXT,
    type VARCHAR(20) DEFAULT 'schedule',
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    image_url VARCHAR(255),
    created_by INT NOT NULL,
    created_at TIMESTAMP,
    updated_at TIMESTAMP,
    FOREIGN KEY (created_by) REFERENCES staff(staff_id)
);

CREATE TABLE support_tickets (
    ticket_id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT NOT NULL,
    booking_id INT,
    car_id INT,
    maintenance_record_id INT,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    phone VARCHAR(20),
    category VARCHAR(20) DEFAULT 'other',
    subject VARCHAR(255) NOT NULL,
    description TEXT NOT NULL,
    status VARCHAR(20) DEFAULT 'open',
    staff_response TEXT,
    assigned_to INT,
```

```
    resolved_at TIMESTAMP,
    closed_at TIMESTAMP,
    flagged_for_maintenance BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP,
    updated_at TIMESTAMP,
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id),
    FOREIGN KEY (booking_id) REFERENCES bookings(booking_id),
    FOREIGN KEY (car_id) REFERENCES cars(id),
    FOREIGN KEY (assigned_to) REFERENCES staff(staff_id)
);

CREATE TABLE cancellation_requests (
    request_id INT AUTO_INCREMENT PRIMARY KEY,
    booking_id INT NOT NULL,
    customer_id INT NOT NULL,
    reason_type VARCHAR(50) NOT NULL,
    reason_details TEXT,
    bank_name VARCHAR(100) NOT NULL,
    bank_account_number VARCHAR(50) NOT NULL,
    bank_account_holder VARCHAR(100) NOT NULL,
    proof_document_url VARCHAR(255),
    status VARCHAR(20) DEFAULT 'pending',
    processed_by_staff_id INT,
    processed_at TIMESTAMP,
    staff_notes TEXT,
    refund_amount DECIMAL(10, 2),
    refund_reference VARCHAR(255),
    refunded_at TIMESTAMP,
    created_at TIMESTAMP,
    updated_at TIMESTAMP,
    FOREIGN KEY (booking_id) REFERENCES bookings(booking_id),
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);
```

5.2.2 Data Manipulation Language (DML)

5.2.2.1 INSERT INTO

```
INSERT INTO residential_colleges (name, code, created_at, updated_at) VALUES
('Kolej Tun Dr Ismail', 'KTDI', NOW(), NOW()),
('Kolej Tun Fatimah', 'KTF', NOW(), NOW());

INSERT INTO staff (name, email, password_hash, role, created_at, updated_at) VALUES
('Ahmad bin Hassan', 'ahmad@hastatravel.com',
'$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', 'admin', NOW(),
NOW()),
('Siti Nurhaliza', 'siti@hastatravel.com',
'$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', 'manager',
NOW(), NOW()),
```

```
('Mohd Faizal', 'faizal@hastatravel.com',
'$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', 'staff', NOW(),
NOW());

INSERT INTO car_locations (name, type, created_at, updated_at) VALUES
('UTM Main Gate', 'both', NOW(), NOW()),
('KTDI Parking Lot', 'pickup', NOW(), NOW());

INSERT INTO cars (plate_number, brand, model, car_type, fuel_type, year,
base_rate_per_hour, status, initial_mileage, current_mileage, service_mileage_limit,
last_service_date, image_url, gps_enabled, created_at, updated_at) VALUES
('JKL1234', 'Perodua', 'Myvi', 'hatchback', 'petrol', 2022, 15.00, 'available', 0, 15000, 10000,
'2024-01-15', '/images/cars/myvi.jpg', TRUE, NOW(), NOW()),
('JKL5678', 'Proton', 'Saga', 'sedan', 'petrol', 2021, 12.00, 'available', 0, 20000, 10000,
'2024-02-20', '/images/cars/saga.jpg', FALSE, NOW(), NOW()),
('JKL9012', 'Honda', 'City', 'sedan', 'petrol', 2023, 25.00, 'in_use', 0, 5000, 10000,
'2024-03-10', '/images/cars/city.jpg', TRUE, NOW(), NOW()),
('JKL3456', 'Toyota', 'Vios', 'sedan', 'petrol', 2022, 22.00, 'available', 0, 18000, 10000,
'2024-01-25', '/images/cars/vios.jpg', TRUE, NOW(), NOW());

INSERT INTO customers (full_name, email, phone, password_hash, ic_url, utmid_url,
license_url, utm_role, college_id, verification_status, verified_by, verified_at,
deposit_balance, total_rental_hours, total_stamps, created_at, updated_at) VALUES
('Ali bin Ahmad', 'ali.ahmad@student.utm.my', '0123456789',
'$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', '/docs/ali_ic.jpg',
'/docs/ali_utmid.jpg', '/docs/ali_license.jpg', 'student', 1, 'approved', 1, NOW(), 100.00, 24, 2,
NOW(), NOW()),
('Sarah binti Mohd', 'sarah.mohd@student.utm.my', '0123456790',
'$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi',
'/docs/sarah_ic.jpg', '/docs/sarah_utmid.jpg', '/docs/sarah_license.jpg', 'student', 2, 'approved',
1, NOW(), 150.00, 48, 4, NOW(), NOW()),
('Lim Wei Jie', 'lim.weijie@student.utm.my', '0123456791',
'$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi', '/docs/lim_ic.jpg',
'/docs/lim_utmid.jpg', '/docs/lim_license.jpg', 'student', 1, 'approved', 2, NOW(), 200.00, 72,
7, NOW(), NOW());

INSERT INTO bookings (customer_id, car_id, pickup_location_id, dropoff_location_id,
start_datetime, end_datetime, rental_hours, base_price, promo_discount, voucher_discount,
total_rental_amount, deposit_amount, deposit_used_amount, deposit_refund_amount,
deposit_decision, agreement_signed_at, final_amount, status, created_at, updated_at)
VALUES
(1, 1, 1, 1, '2024-04-01 08:00:00', '2024-04-01 20:00:00', 12, 180.00, 0.00, 0.00, 180.00,
100.00, 100.00, 0.00, 'carry_forward', '2024-03-30 10:00:00', 80.00, 'completed', NOW(),
NOW()),
(2, 2, 2, 1, '2024-04-05 09:00:00', '2024-04-06 09:00:00', 24, 288.00, 20.00, 0.00, 268.00,
150.00, 150.00, 0.00, 'refund', '2024-04-03 14:00:00', 118.00, 'completed', NOW(), NOW()),
(3, 3, 1, 1, '2024-04-10 10:00:00', '2024-04-10 18:00:00', 8, 200.00, 0.00, 25.00, 175.00,
200.00, 175.00, 25.00, 'refund', '2024-04-08 11:00:00', 0.00, 'active', NOW(), NOW()),
```

```
(1, 4, 1, 1, '2024-04-15 07:00:00', '2024-04-15 19:00:00', 12, 264.00, 0.00, 0.00, 264.00,
100.00, 0.00, 0.00, NULL, '2024-04-13 09:00:00', 264.00, 'confirmed', NOW(), NOW());

INSERT INTO payments (booking_id, penalty_id, amount, payment_type, payment_method,
bank_name, account_holder_name, account_number, receipt_url, payment_date, status,
created_at, updated_at) VALUES
(1, NULL, 100.00, 'deposit', 'bank_transfer', 'Maybank', 'Ali bin Ahmad', '1234567890',
'/receipts/payment_001.jpg', '2024-03-28 10:00:00', 'verified', NOW(), NOW()),
(1, NULL, 80.00, 'rental', 'bank_transfer', 'Maybank', 'Ali bin Ahmad', '1234567890',
'/receipts/payment_002.jpg', '2024-03-30 11:00:00', 'verified', NOW(), NOW()),
(2, NULL, 150.00, 'deposit', 'bank_transfer', 'CIMB Bank', 'Sarah binti Mohd', '9876543210',
'/receipts/payment_003.jpg', '2024-04-01 09:00:00', 'verified', NOW(), NOW()),
(2, NULL, 118.00, 'rental', 'bank_transfer', 'CIMB Bank', 'Sarah binti Mohd', '9876543210',
'/receipts/payment_004.jpg', '2024-04-03 15:00:00', 'verified', NOW(), NOW()),
(3, NULL, 200.00, 'deposit', 'bank_transfer', 'Public Bank', 'Lim Wei Jie', '5555555555',
'/receipts/payment_005.jpg', '2024-04-06 14:00:00', 'verified', NOW(), NOW());

INSERT INTO penalties (booking_id, penalty_type, description, amount, is_installment,
status, created_at, updated_at) VALUES
(1, 'late', 'Returned 2 hours late', 30.00, FALSE, 'settled', NOW(), NOW()),
(2, 'fuel', 'Returned with less fuel than pickup', 20.00, FALSE, 'settled', NOW(), NOW());

INSERT INTO inspections (booking_id, car_id, inspection_type, type, status, datetime,
fuel_level, odometer_reading, mileage_reading, notes, inspected_by, inspected_at,
created_at, updated_at) VALUES
(1, 1, 'pickup', 'before', 'completed', '2024-04-01 08:00:00', 8, 15000, 15000, 'Vehicle in good
condition', 3, '2024-04-01 08:15:00', NOW(), NOW()),
(1, 1, 'return', 'after', 'completed', '2024-04-01 22:00:00', 6, 15120, 15120, 'Returned 2 hours
late, minor fuel reduction', 3, '2024-04-01 22:10:00', NOW(), NOW());

INSERT INTO vouchers (code, description, discount_type, discount_value,
min_stamps_required, expiry_date, is_active, created_at, updated_at) VALUES
('WELCOME10', 'Welcome discount for new customers', 'percent', 10.00, 0, '2024-12-31',
TRUE, NOW(), NOW()),
('STAMP25', '25% discount for loyal customers', 'percent', 25.00, 5, '2024-12-31', TRUE,
NOW(), NOW());

INSERT INTO voucher_redemptions (voucher_id, customer_id, booking_id,
discount_amount, redeemed_at, created_at, updated_at) VALUES
(2, 3, 3, 25.00, '2024-04-08 11:00:00', NOW(), NOW());

INSERT INTO feedbacks (booking_id, customer_id, rating, comment, reported_issue,
created_at, updated_at) VALUES
(1, 1, 4, 'Good service, but returned late due to traffic', FALSE, NOW(), NOW());

INSERT INTO maintenance_records (car_id, service_date, description, mileage_at_service,
cost, created_at, updated_at) VALUES
(1, '2024-01-15', 'Annual service', 10000, 300.00, NOW(), NOW());
```

```
INSERT INTO activities (title, description, type, start_date, end_date, image_url, created_by, created_at, updated_at) VALUES
('Semester Break Promotion', 'Get 20% off on all rentals during semester break', 'promotion', '2024-05-01', '2024-05-31', '/images/promos/semester_break.jpg', 2, NOW(), NOW());

INSERT INTO support_tickets (customer_id, booking_id, car_id, name, email, phone, category, subject, description, status, assigned_to, created_at, updated_at) VALUES
(1, 1, NULL, 'Ali bin Ahmad', 'ali.ahmad@student.utm.my', '0123456789', 'booking', 'Late return penalty query', 'I was charged for late return but there was heavy traffic', 'resolved', 3, NOW(), NOW());
```

5.2.2.2 BASIC SELECT

```
SELECT plate_number, brand, model, car_type, base_rate_per_hour, status
FROM cars
WHERE status = 'available'
ORDER BY created_at DESC
LIMIT 12;
```

5.2.2.3 INNER JOIN SELECT

```
SELECT b.booking_id, c.full_name AS customer_name, c.email AS customer_email, car.plate_number, car.brand, car.model, b.start_datetime, b.end_datetime, b.final_amount, b.status
FROM bookings b
JOIN customers c ON b.customer_id = c.customer_id
JOIN cars car ON b.car_id = car.id
WHERE b.status = 'completed'
ORDER BY b.start_datetime DESC;
```

5.2.2.4 LEFT JOIN SELECT

```
SELECT c.customer_id, c.full_name, c.email, c.utm_role, COUNT(b.booking_id) AS total_bookings, IFNULL(SUM(b.final_amount), 0) AS total_revenue
FROM customers c
LEFT JOIN bookings b ON c.customer_id = b.customer_id
GROUP BY c.customer_id, c.full_name, c.email, c.utm_role
ORDER BY total_revenue DESC;
```

5.2.2.5 SELECT WITH AGGREGATE FUNCTIONS

```
SELECT car.plate_number, car.brand, car.model, COUNT(b.booking_id) AS booking_count, IFNULL(SUM(b.final_amount), 0) AS total_revenue, IFNULL(SUM(b.rental_hours), 0) AS total_hours
FROM cars car
LEFT JOIN bookings b ON car.id = b.car_id
GROUP BY car.id, car.plate_number, car.brand, car.model
ORDER BY booking_count DESC;
```

5.2.2.6 SELECT WITH SUBQUERY

```
SELECT c.customer_id, c.full_name, c.email
FROM customers c
```

```
WHERE c.customer_id IN (
    SELECT DISTINCT customer_id
    FROM bookings
    WHERE status = 'completed'
)
ORDER BY c.full_name;
```

5.2.2.7 SELECT USING CASE STATEMENT

```
SELECT
    b.booking_id,
    c.full_name AS customer_name,
    car.plate_number,
    b.rental_hours,
    CASE
        WHEN b.rental_hours <= 8 THEN 'Short Rental'
        WHEN b.rental_hours <= 24 THEN 'Day Rental'
        WHEN b.rental_hours <= 72 THEN 'Extended Rental'
        ELSE 'Long Term Rental'
    END AS rental_category,
    CASE
        WHEN b.status = 'created' THEN 'Booking Created - Awaiting Payment'
        WHEN b.status = 'confirmed' THEN 'Payment Confirmed - Ready for Pickup'
        WHEN b.status = 'active' THEN 'Currently Active Rental'
        WHEN b.status = 'completed' THEN 'Rental Completed Successfully'
        WHEN b.status = 'cancelled' THEN 'Booking Cancelled'
        ELSE 'Unknown Status'
    END AS status_description,
    b.final_amount  -- Changed from total_rental_amount
FROM bookings b
INNER JOIN customers c ON b.customer_id = c.customer_id
INNER JOIN cars car ON b.car_id = car.id
ORDER BY b.rental_hours DESC;
```

5.2.2.8 UPDATE

```
UPDATE customers
SET deposit_balance = deposit_balance + 25.00, updated_at = NOW()
WHERE customer_id = 3;

UPDATE bookings
SET status = 'completed', updated_at = NOW()
WHERE booking_id = 3 AND status = 'active';

UPDATE payments
SET status = 'verified', updated_at = NOW()
WHERE payment_id = 4 AND status = 'pending';
```

5.2.2.9 DELETE

```
DELETE FROM bookings
WHERE status = 'cancelled' AND created_at < DATE_SUB(NOW(), INTERVAL 1 YEAR);

DELETE FROM vouchers
WHERE expiry_date < CURDATE() AND is_active = FALSE;
```

7.0 SUMMARY

Phase 3 of this project focuses on the logical database design for the Hasta Travel & Tours Car Rental System. This phase translates the conceptual and enhanced ERD into a logical structure that is suitable for a relational database. It defines how data is organized, connected, and prepared for implementation.

In this phase, a Logical ERD is developed to show all entities, attributes, primary keys, and foreign keys. Relationships between entities are clearly defined, and many-to-many relationships are resolved using associative tables. Important constraints such as uniqueness and optional relationships are also considered to reflect system rules.

Normalization up to Third Normal Form (3NF) is applied to reduce data redundancy and improve data integrity. The outcome of Phase 3 is a complete and well-structured logical database design that is ready to be implemented using SQL in the next phase.

8.0 APPENDICES

8.1 Use Case Diagram

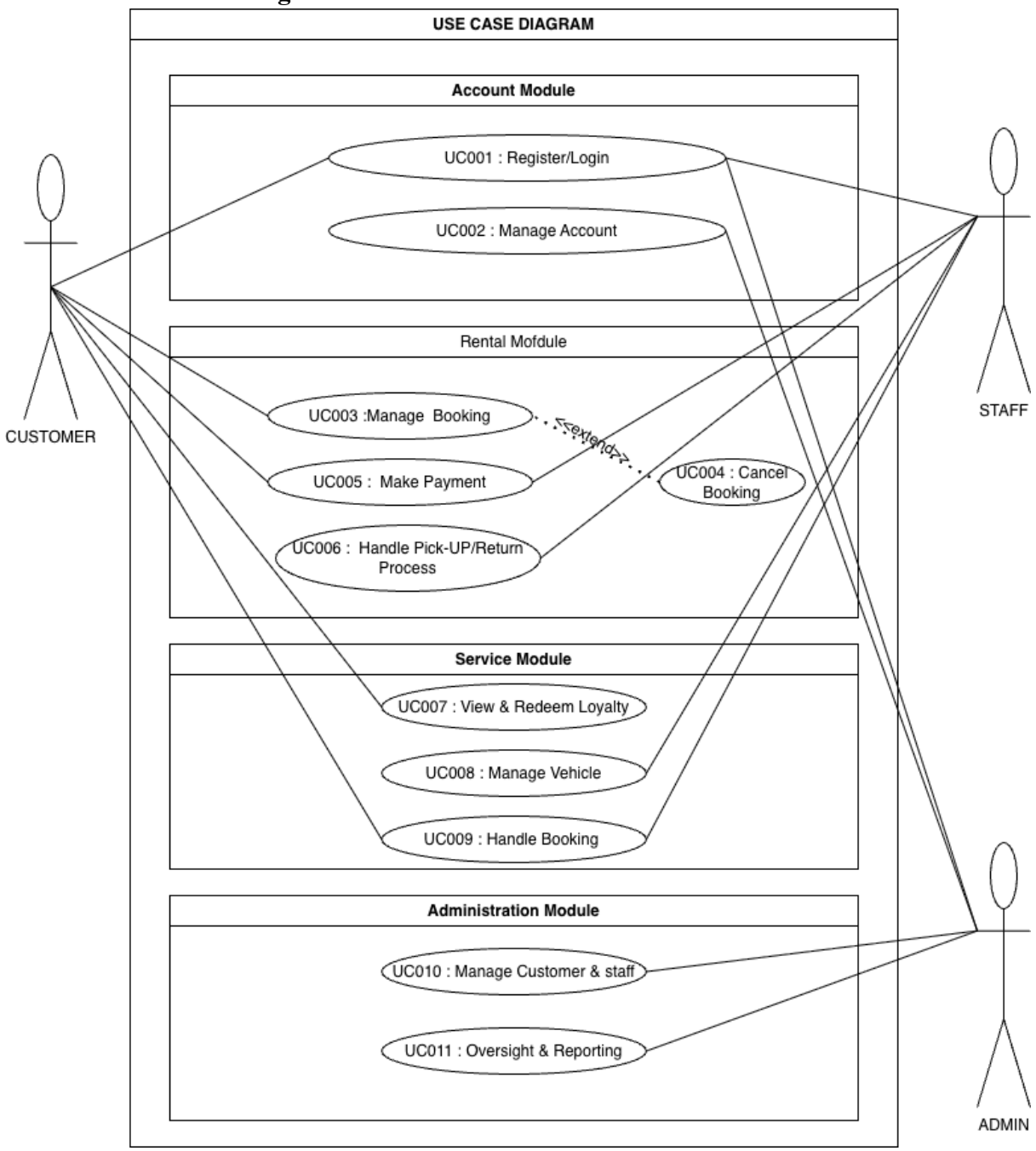


Figure 8.1.1 Use Case Diagram for Hasta Travel & Tours Car Rental System.

8.2 Activity Diagram

8.2.1 U001 : User Story <Register/Login>

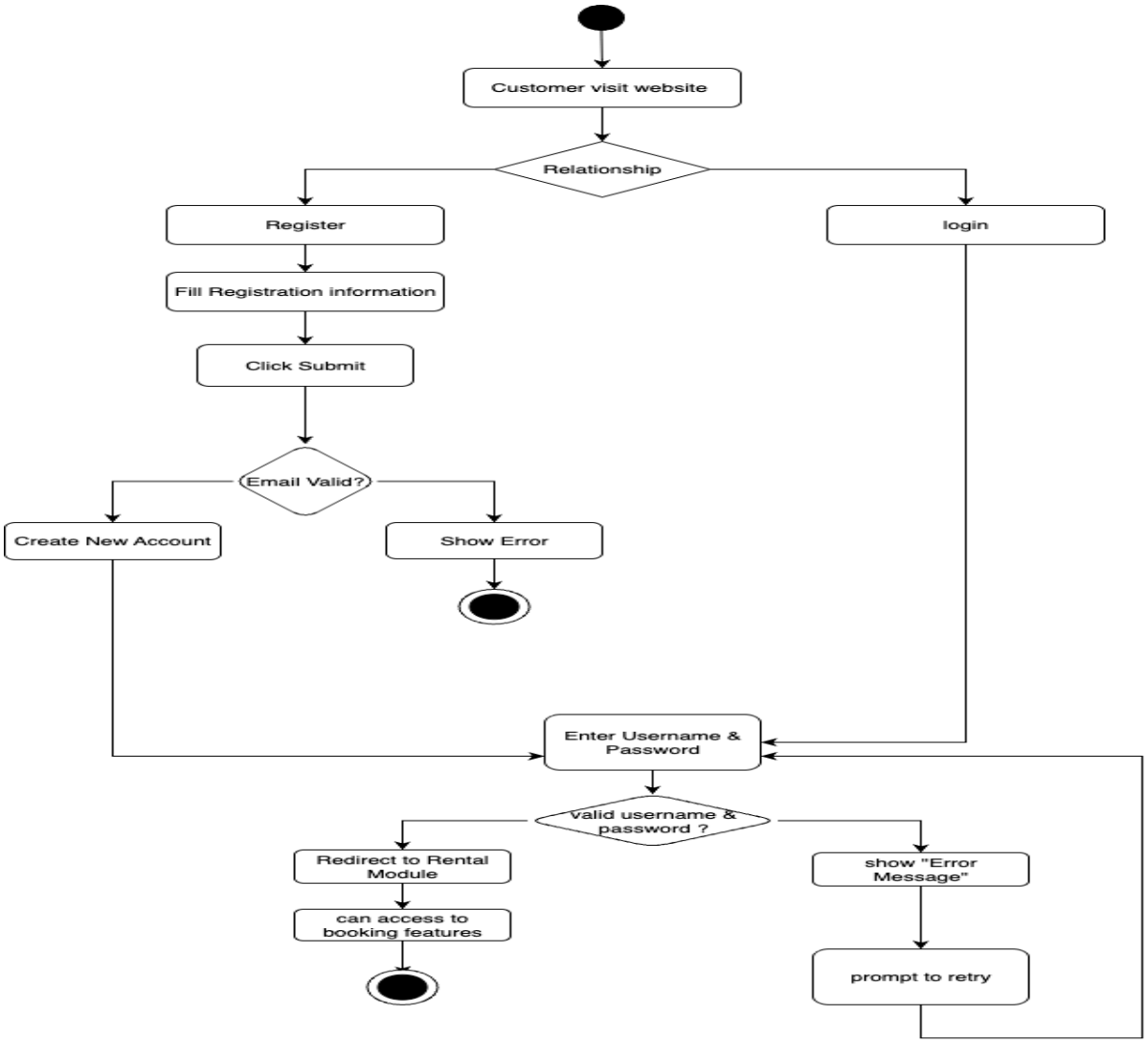


Figure 8.2.1: Activity Diagram for <Register/Login>

8.2.2 U002 : User Story <Manage Account>

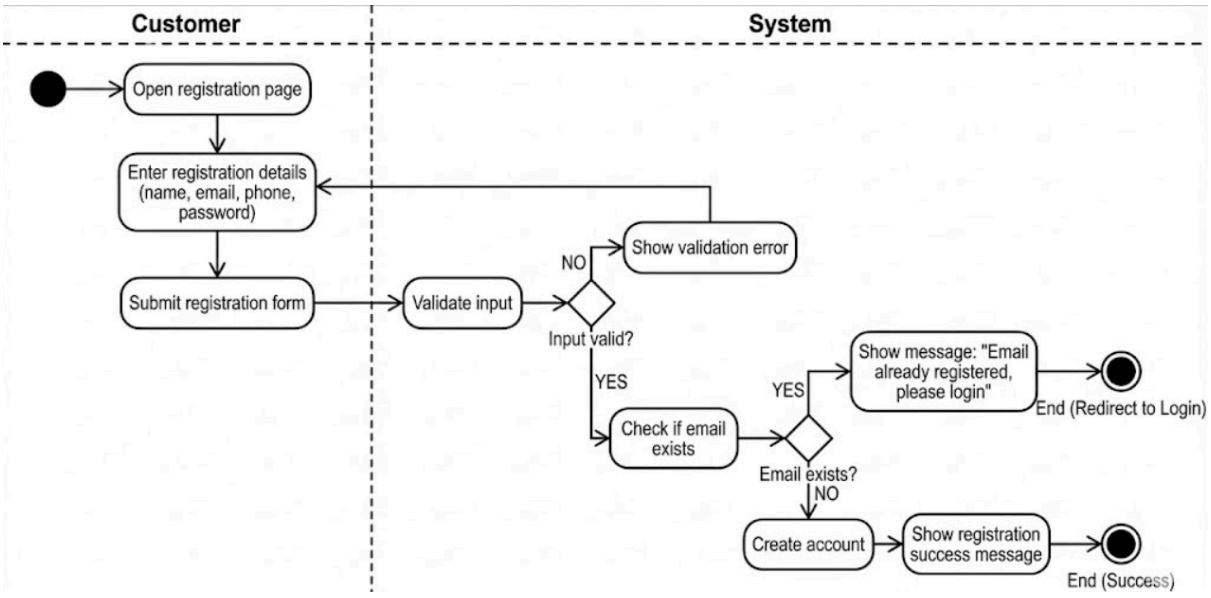


Figure 8.2.2: Activity Diagram for <Manage Vehicle>

8.2.3 U003 : User Story <Manage Booking>

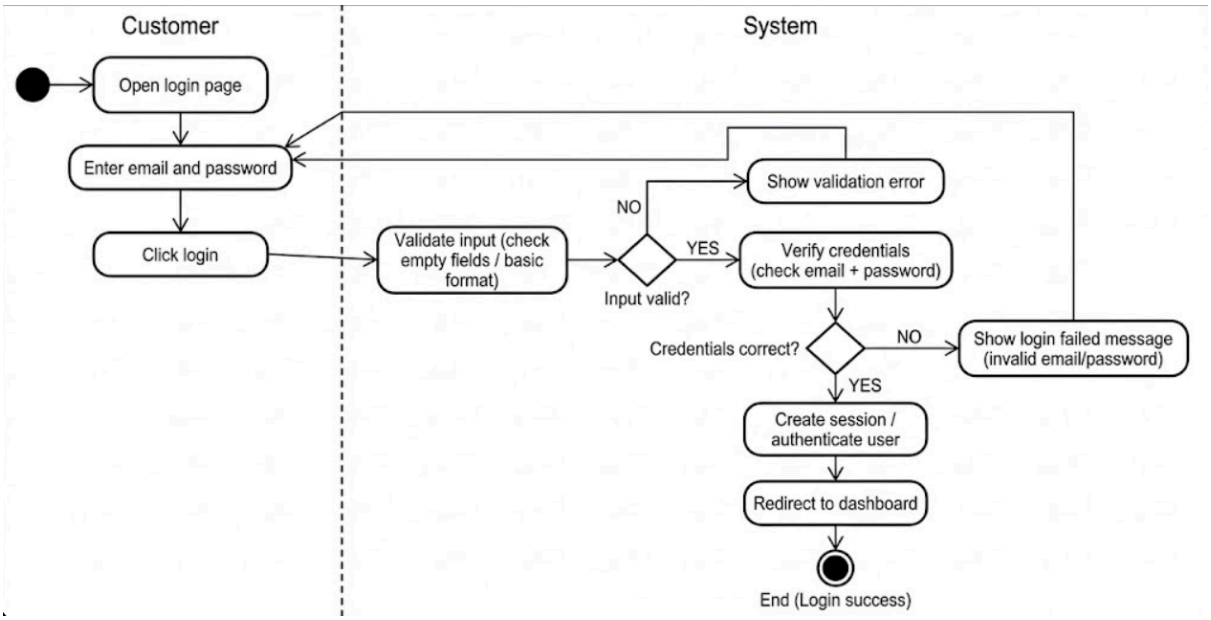


Figure 8.2.3: Activity Diagram for <Manage Booking>

8.2.2 U004 : User Story <Cancel Booking>

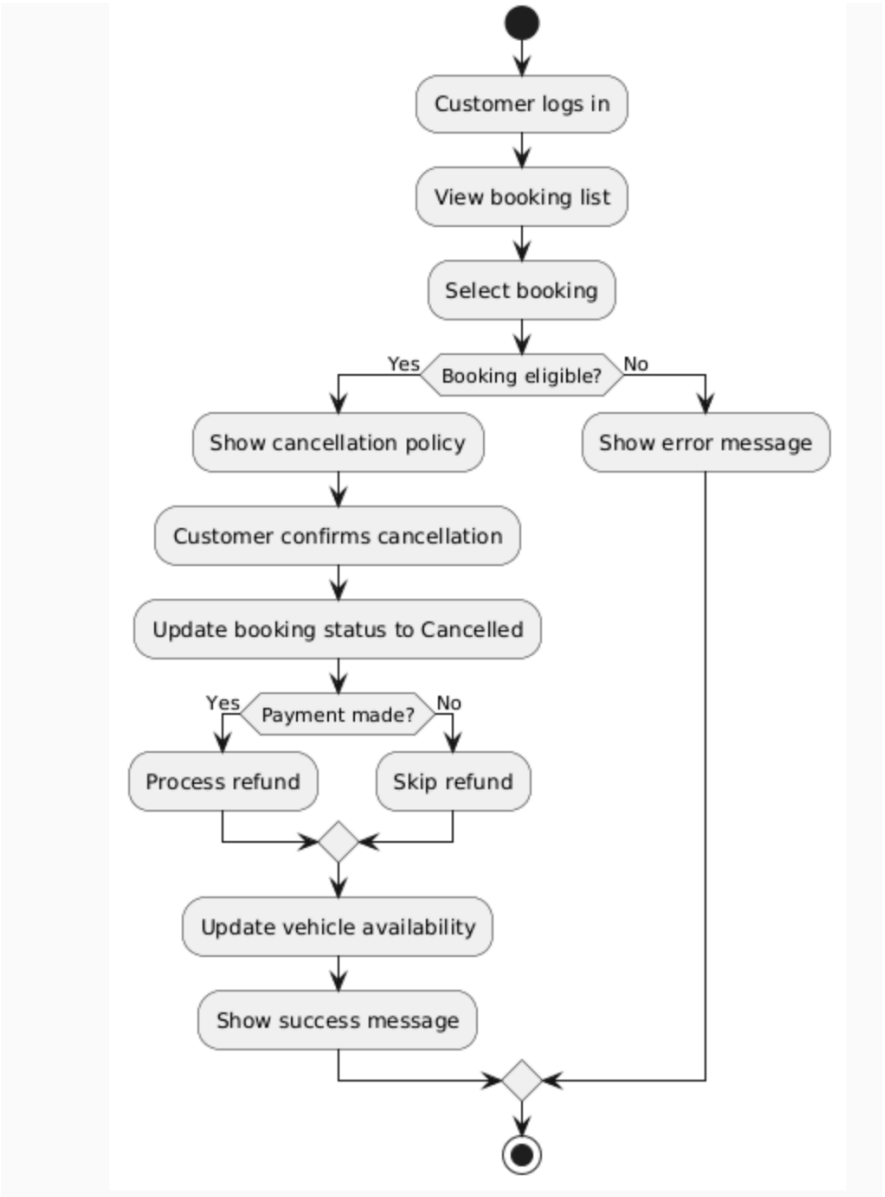


Figure 8.2.4: Activity Diagram for <Cancel Booking>

8.2.3 U005 : User Story <Make Payment>

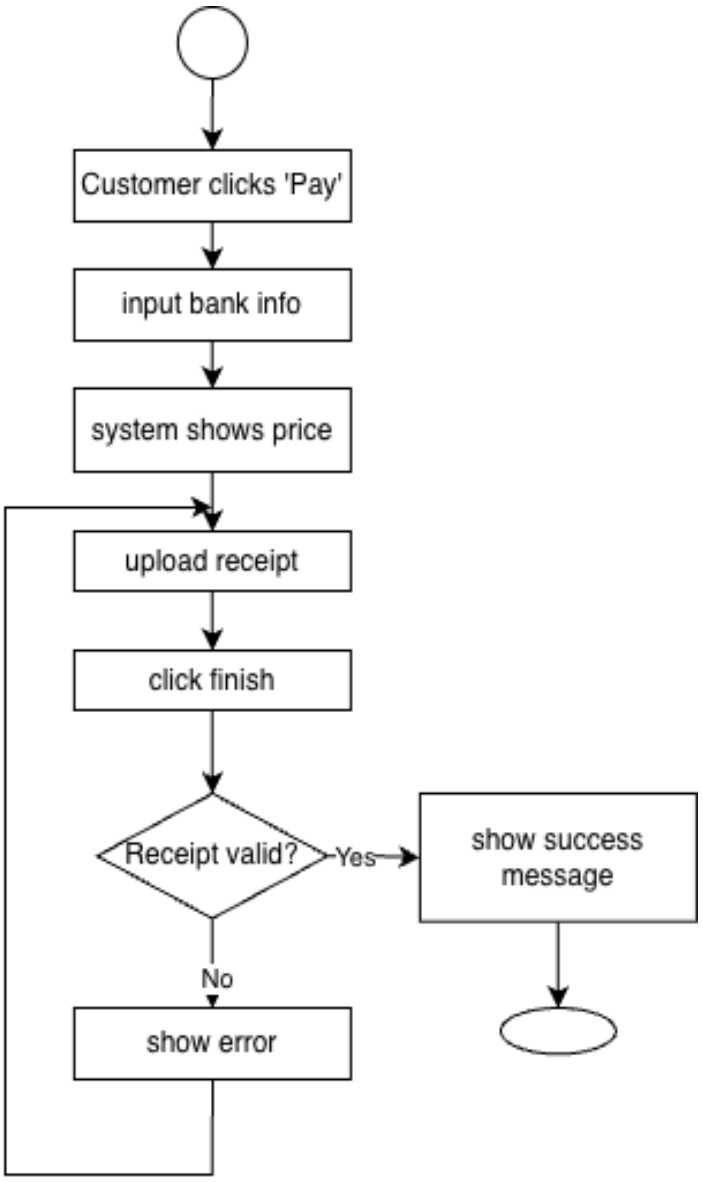


Figure 8.2.5: Activity Diagram for <Make Payment>

8.2.6 U006 : User Story <Handle Pick-Up/Return Process>

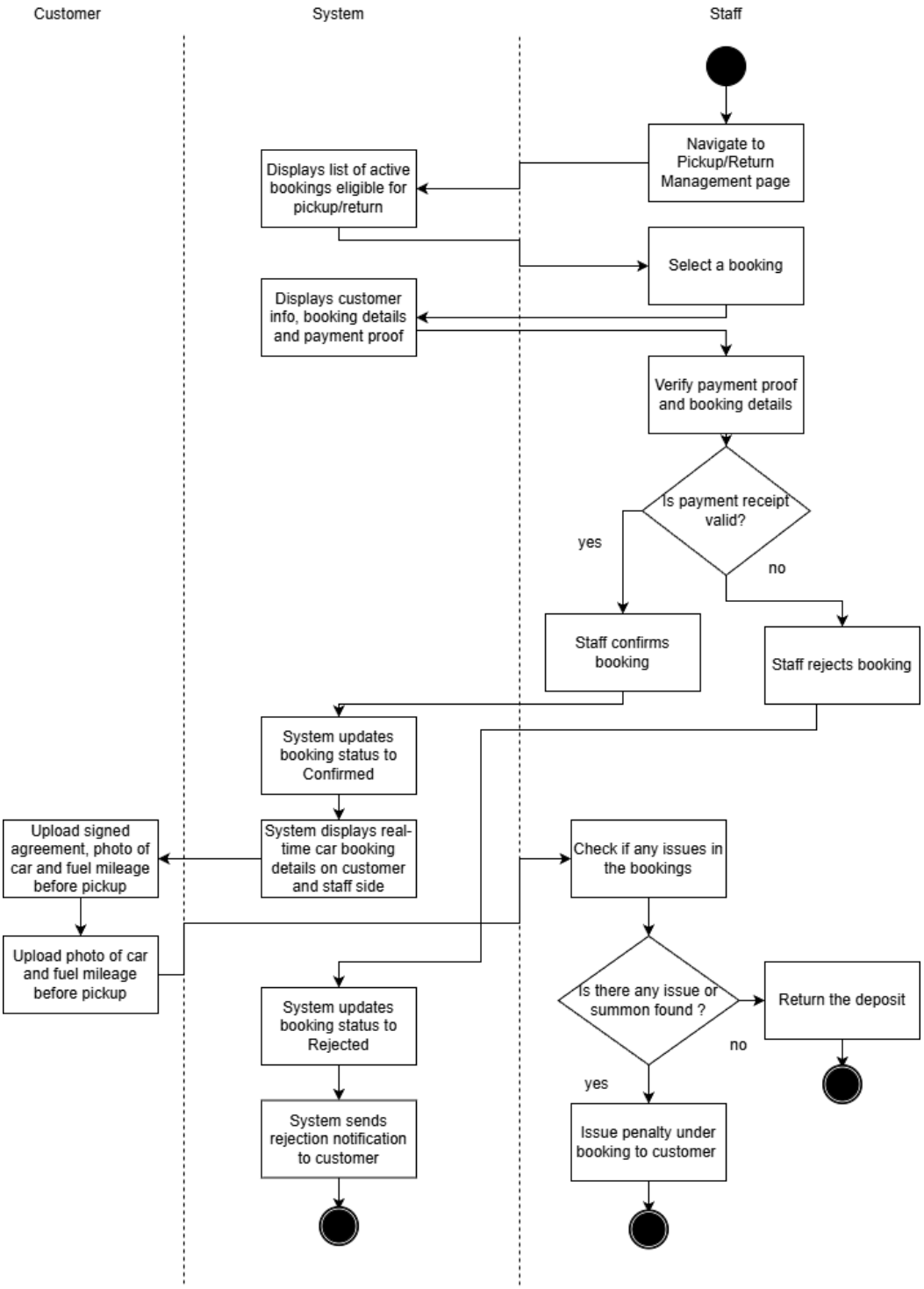


Figure 8.2.6: Activity Diagram for <Handle Pick-Up/Return Process>

8.2.7 U007 : User Story <View and Redeem Loyalty>

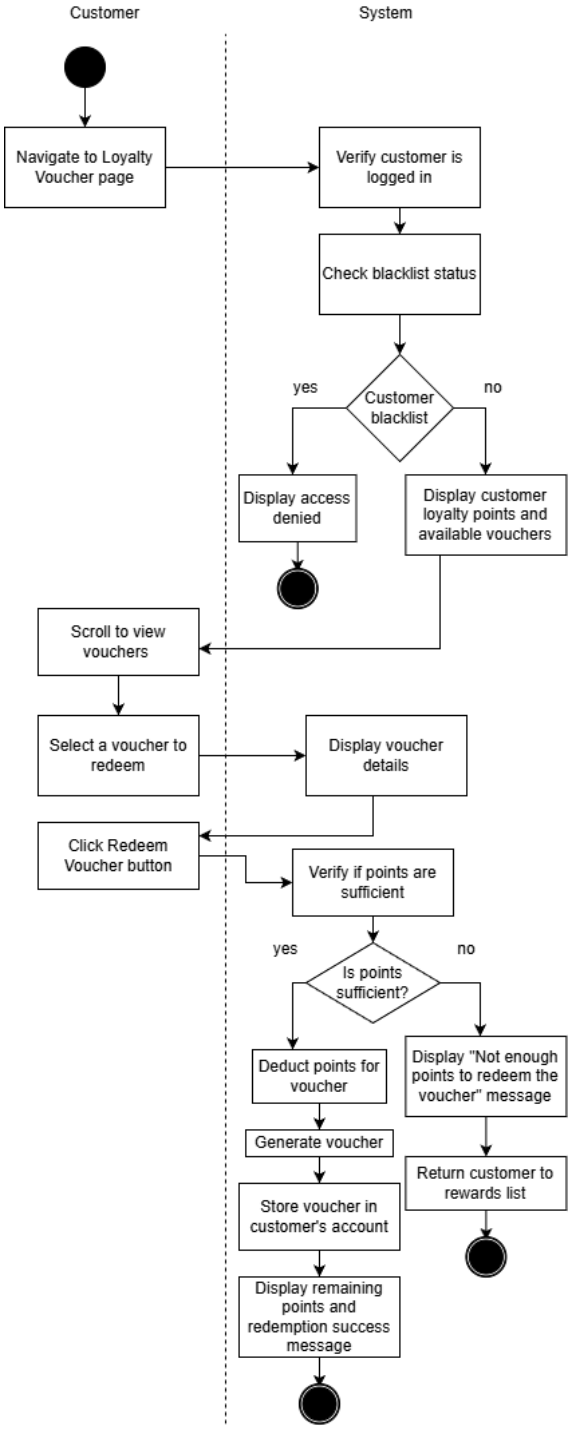


Figure 8.2.7: Activity Diagram for <View and Redeem Loyalty>

8.2.8 U008 : User Story <Manage Vehicle>

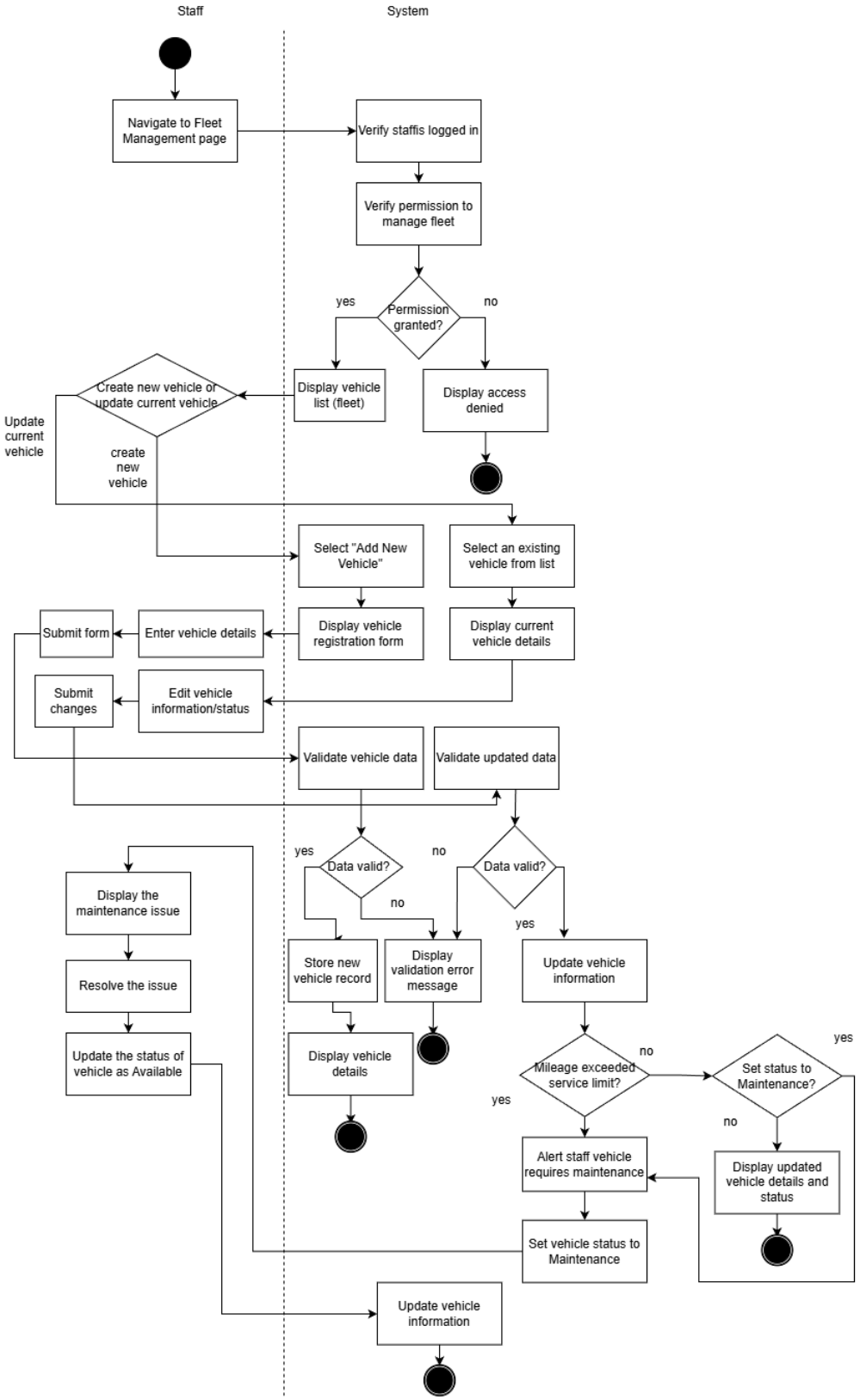


Figure 8.2.8: Activity Diagram for <Manage Vehicle>

8.2.9 U009 : User Story <Handle Customer Issues>

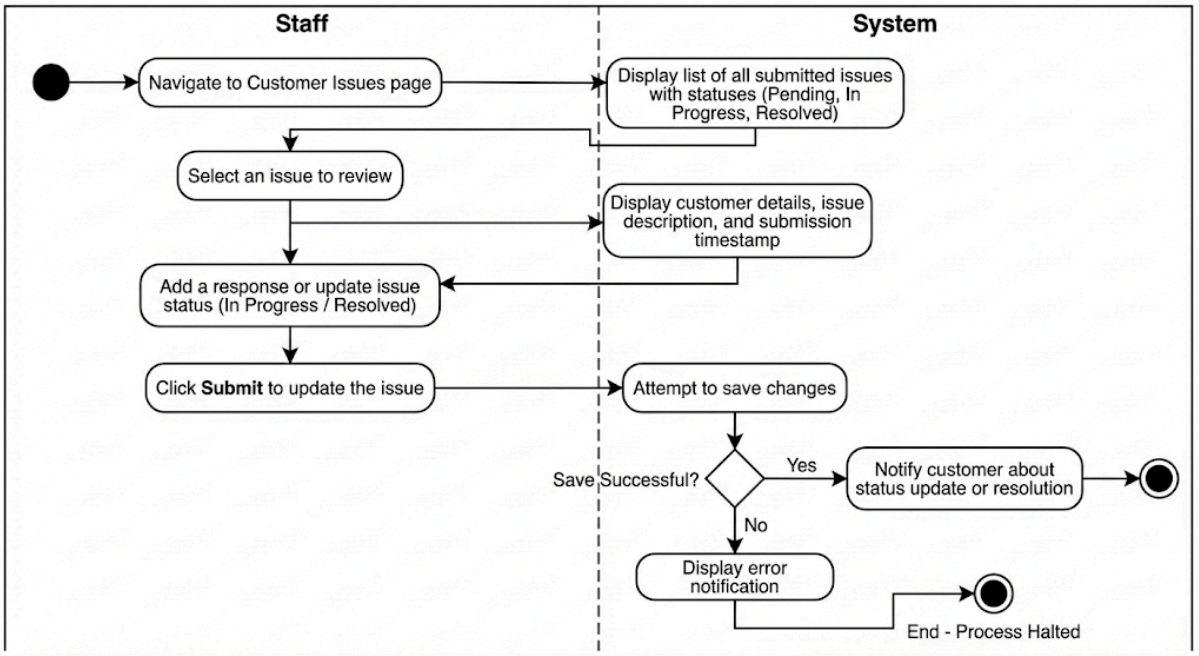


Figure 8.2.9: Activity Diagram for <Handle Customer Issues>

8.2.10 U010 : User Story <Manage Customer & Staff>

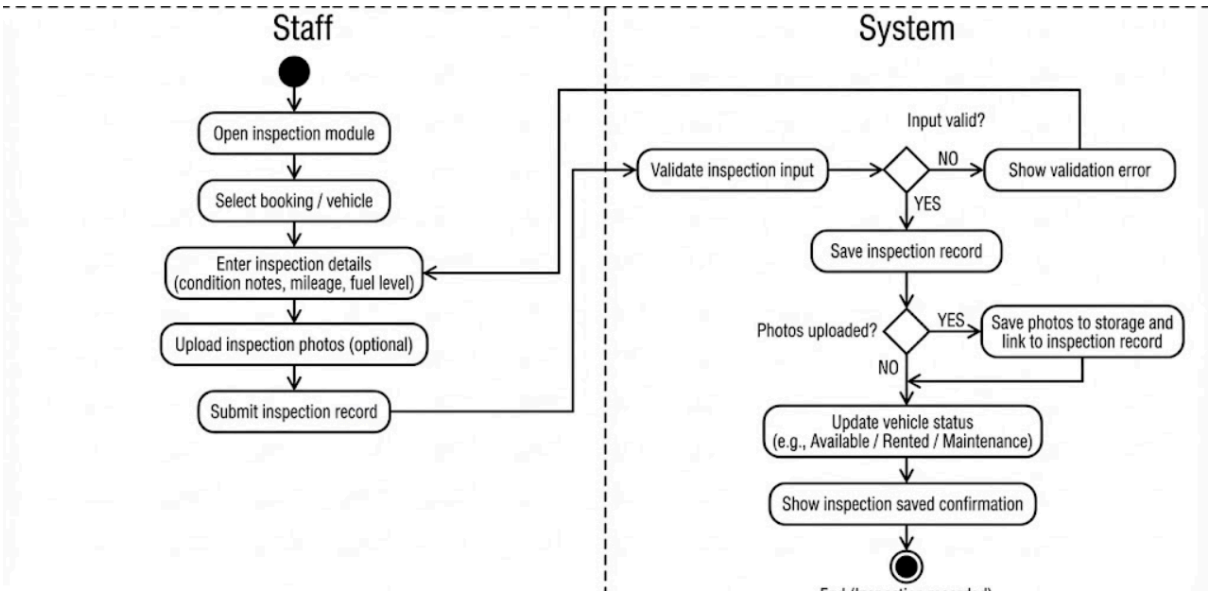


Figure 8.2.10: Activity Diagram for <Manage Customer & Staff>

8.2.11 U0011 : User Story <Oversight & Reporting>

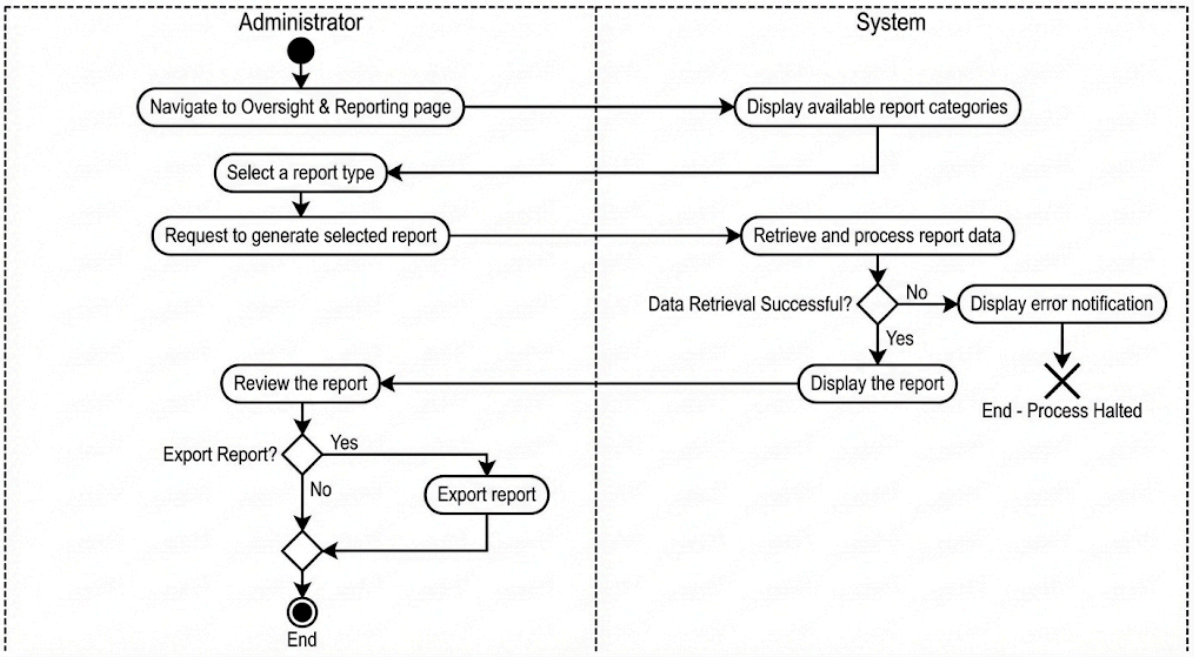


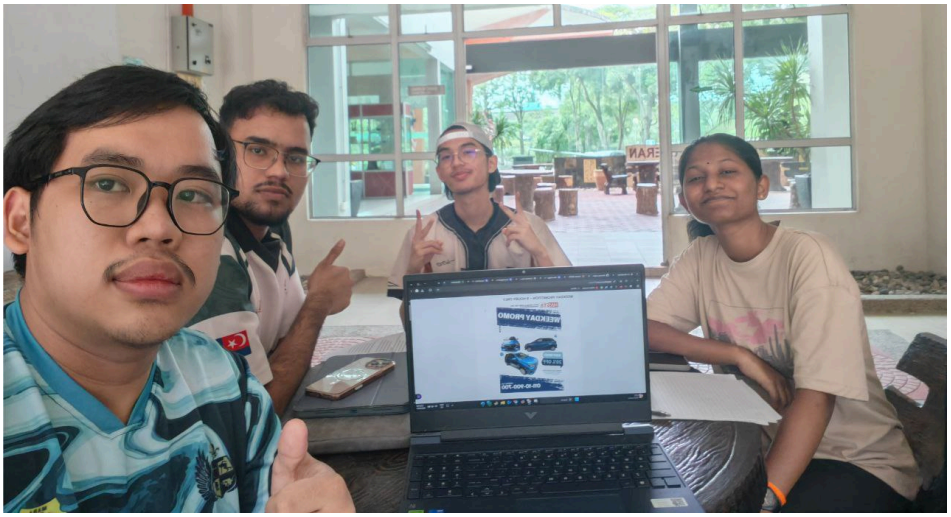
Figure 8.2.11: Activity Diagram for <Oversight & Reporting>

6.3 Meeting Log

Meeting Discussion #1 (13/10/2025)

Our team discussed on the system that we are intended to do and brainstorming on what features to add. We also researched on similar rental system to get inspiration and idea to improve the our proposed system

Meeting Discussion #2 (06/11/2025)

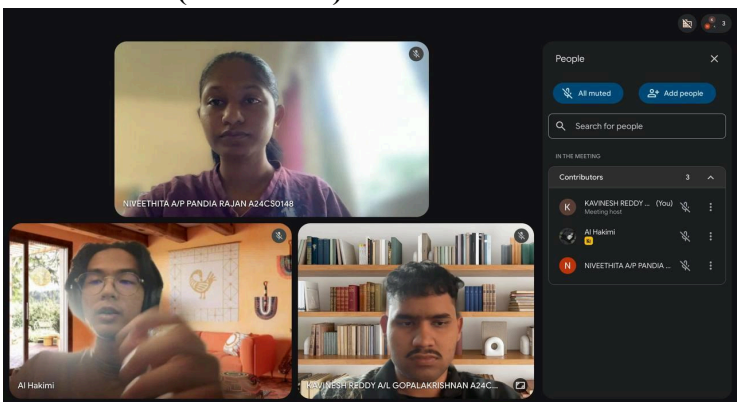


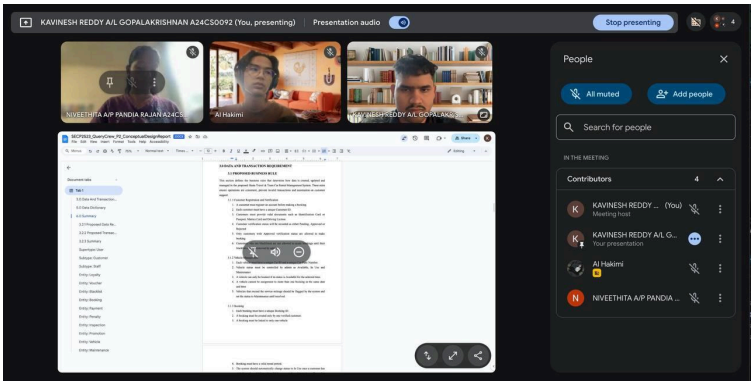
We finalised on any improvements and features needed on our proposed system and divided the task among members. We discussed and clarified on our respective tasks and set internal deadline for accountability checking among the team members.

Meeting Discussion #3 (16/01/2026)

The team discussed the system requirements gathered earlier and how they should be represented in the conceptual model. Members ensured the ERD to be designed should be aligned to the requirements and not out of it. Tasks are divided among members equally to complete the report. The team clarified on the confusion regarding the conceptual model, the current system and the task allocated to ensure no errors made later on.

Meeting Discussion #4 (31/01/2026)





The team reviewed conceptual ERD and discussed how to improvise it before creating the logical ERD. The task for the project is divided among members equally and clarified on doubts regarding their respective tasks. Members aligned on the updated version of the system to ensure everyone is on the same page.