Detailed System Design

SOFTWARE DEVELOPMENT LIFE CYCLE

Car Rental System

Prepared By :

Deepasree Meena Padmanabhan, Pujan Bhuva,

Simranjeet Kaur & OgheneRukevwe Esegba

Algoma UNIVERSITY

Department of Computer Science

Software engineering (COSC3506 001XE)

1. Table of Contents

2. Revision History

Version	Date	Author	Description
1	2/19/2025	Deepasree Meena Padmanabhan	Defined system objectives, scope, and assumptions, ensuring alignment with project goals.
1.1	2/21/2025	Simranjeet Kaur	Designed system architecture, detailing the three-tier structure, component interactions, and UML diagrams.
1.2	2/22/2025	OgheneRukevwe Esegba	Documented module functionalities, covering responsibilities, data flows, dependencies, and reporting.
1.3	2/24/2025	Pujan Bhuva	Finalized security, implementation, testing, and deployment strategies for system reliability.
1.4	3/10/2025	Deepasree Meena Padmanabhan	Final editing and formatting done

3. Introduction

3.1 Objective

This is a thorough system design of the Car Rental System. It helps users search cars, book them, process payments, and manage stock in an effective way. The system facilitates automated core activities like tracking cars, reservation, and the maintenance schedule. A systematic process enhances user experience, reduces errors, and maximizes usage. This report is a blueprint for designing, deploying, and maintaining the system.

3.2 Scope

The Car Rental System comprises user login, car inventory management, booking, payment gateway, vehicle return validation, and reporting. The system offers safe login, effective vehicle tracking, and smooth reservation management. The system is both front-end and back-end enabled, offering convenience of user interaction. But the system does not include third-party payment integration and AI-based recommendations. The report clearly says the features and limitations of the system for systematic development.

3.3 Assumptions

- The system is desktop-based with JavaFX for GUI, Java for backend, and MySQL for storage.
- Role-Based Access Control (RBAC) facilitates safe user access.
- All users log in using their own credentials.
- Admins manage transactions and vehicle availability within a managed environment.
- Customers must enter accurate data to prevent inconsistency with booking.

4. System Architecture Design

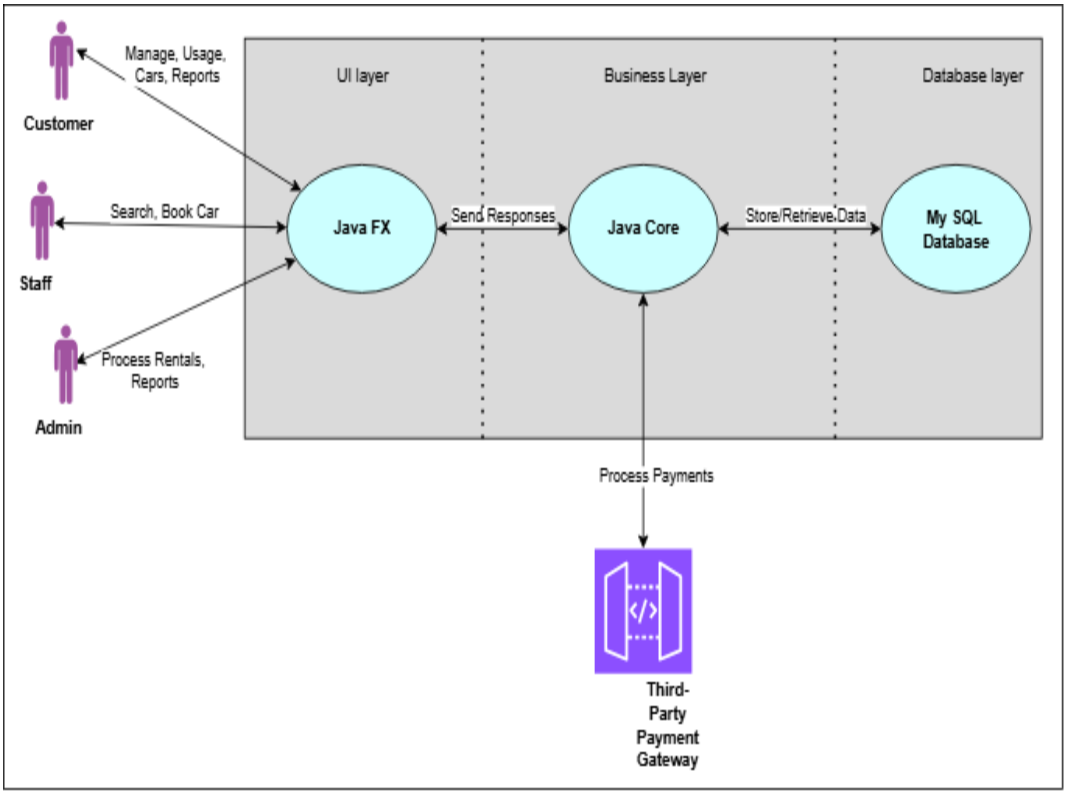
4.1 Architectural Overview

The Car Rental System follows three-tier architecture, ensuring separation of concerns and better system management:

- **Presentation Layer (UI):** Handles user interactions and displays relevant information. Built using JavaFX.

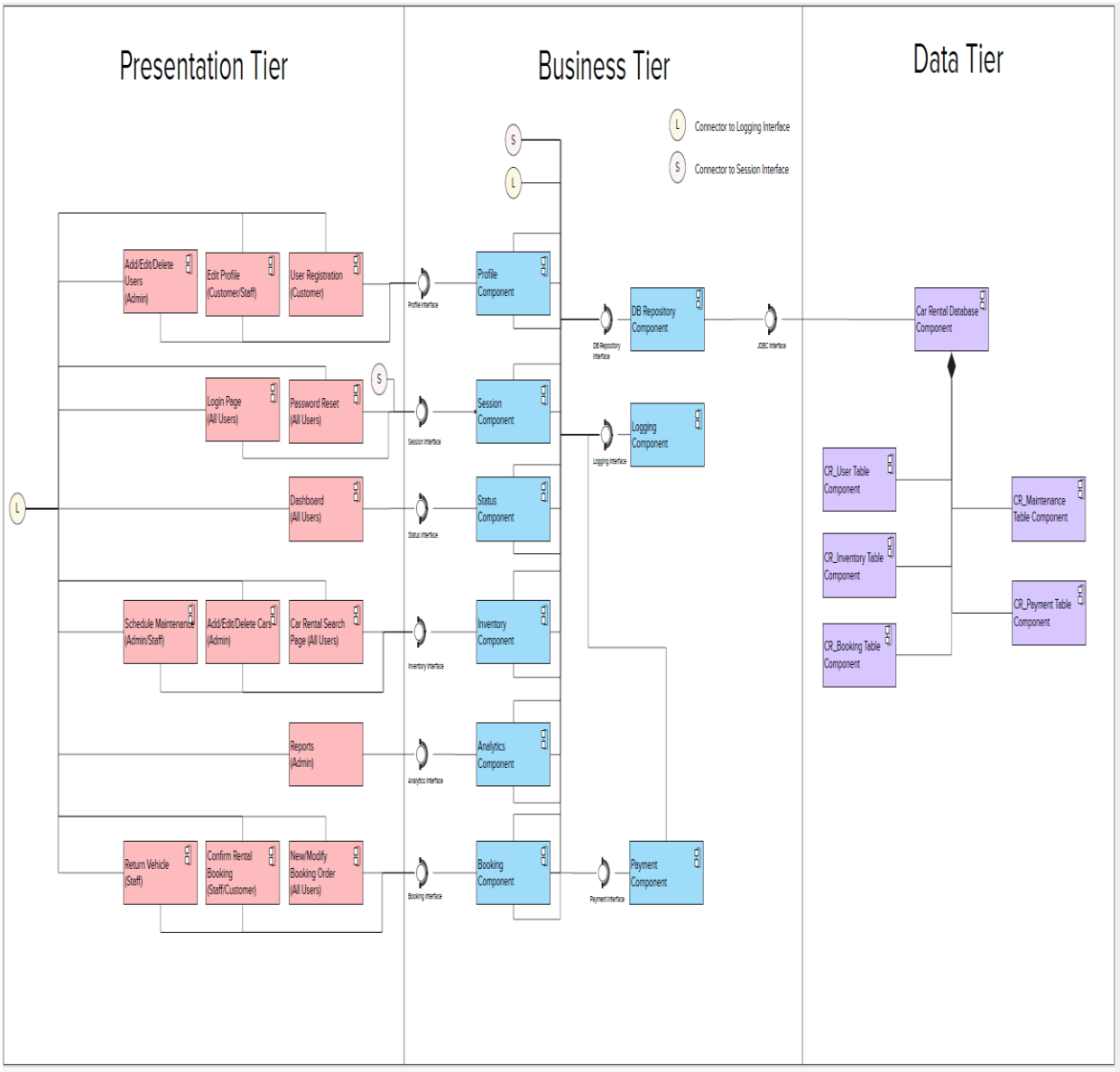
- **Business Logic Layer:** Manages the core functionalities like authentication, booking, and reporting using Java classes.

- **Data Layer:** Stores user, vehicle, and booking details in a MySQL database.



4.2 Component Diagram

A Component Diagram is a diagram that illustrates the interaction between different system components, such as modules, databases, interfaces, or services. It represents the system's structure, dependencies, and communication, aiding in system design, architecture planning, and understanding system parts interconnection.



The component diagram illustrates the Car Rental System's architecture, divided into three primary tiers:

- Presentation Tier (User Interface)
- Business Tier (Application Logic)
- Data Tier (Database & Storage)

Each tier contains distinct components responsible for specific functionalities, ensuring modularity and maintainability.

Presentation Tier

The Presentation Tier consists of UI components that allow users (Admins, Customers, and Staff) to interact with the system. These components handle user input and communicate with the Business Tier for processing.

Admin Functionalities:

- Add/Edit/Delete Users
- Schedule Maintenance
- Add/Edit/Delete Cars
- Reports

General User Functionalities:

- Login Page
- Password Reset
- Dashboard
- Car Rental Search
- Confirm Rental Booking
- New/Modify Booking Order

Staff Functionalities:

- Return Vehicle

Connectors:

These UI components interact with the Business Tier through APIs or service calls.

Business Tier

The Business Tier serves as the application's processing layer. It contains several service components that handle business logic and interactions between the Presentation and Data Tiers.

Key Components:

- Profile Component: Manages user data and profiles.
- Session Component: Maintains active sessions.
- Status Component: Tracks vehicle and booking statuses.
- Inventory Component: Manages available car inventory.
- Analytics Component: Processes system usage and rental data.
- Booking Component: Handles rental booking requests.
- Payment Component: Manages transactions.
- Logging Component: Records system logs.
- DB Repository Component: Facilitates database queries.

Connectors:

Session Interface (S: Manages user authentication.

Logging Interface (L): Sends logs to the logging component.

Other Interfaces: Facilitate interactions between business components.

Data Tier

The Data Tier consists of the Car Rental Database and its tables, which store all system-related information.

Database Tables:

- CR_User Table: Stores customer and staff data.
- CR_Inventory Table: Maintains car details and availability.
- CR_Booking Table: Stores rental booking records.
- CR_Maintenance Table: Tracks vehicle maintenance schedules.
- CR_Pyment Table: Records payment transactions.

Connectors :

The DB Repository Component in the Business Tier interacts with the database via JDBC Interface to perform CRUD operations.

This modular approach allows better scalability, maintainability, and flexibility in system development.

5. Detailed Module Descriptions

5.1 Authentication & Profile Management Module

Module Overview

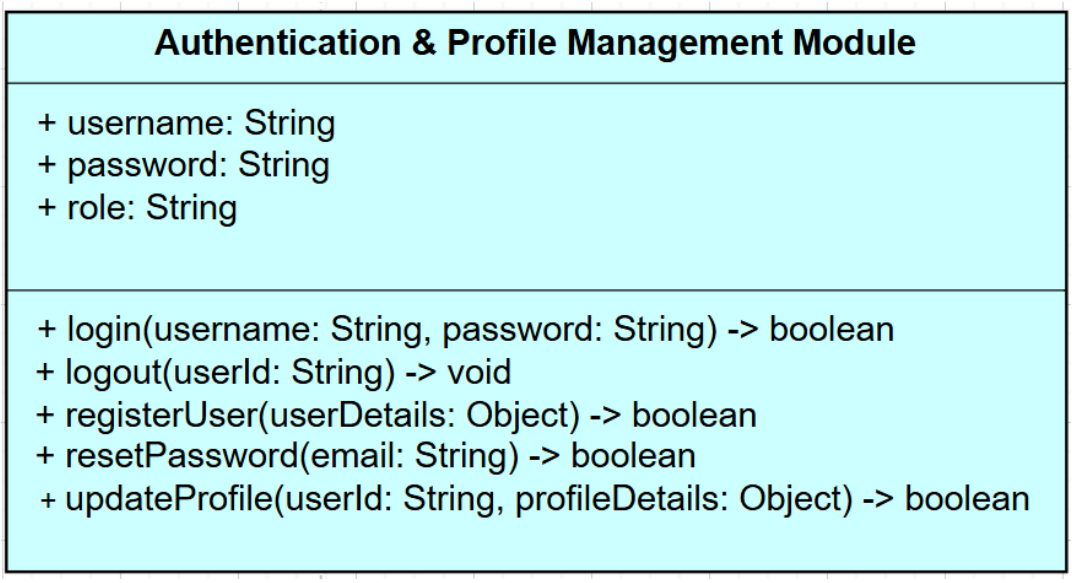
Responsibilities: "This module is responsible for user authentication, role-based access control and profile update, giving secure system access to Administrators, Staff, and Customers. It is responsible for login, logout, password reset, and user registration and imposes security policy such as encryption and multi-factor authentication. The module verifies user identity and denies unauthorized access based on specified roles. It also updates user profile data, allowing users to update their information while ensuring data integrity. With the integration of session tracking, it gives security through auditing authentication attempts and monitoring login activities."

Dependencies: "This module interacts with the Session Management Module to securely manage user sessions and prevent unauthorized access. It also relies on the Database Module to store and retrieve user profiles and credentials. It also interacts with the Logging Module to log in and track failed login attempts for security reasons."

Input : "The module accepts login details (password and username), registration details for the user, requests for updating a profile, and password reset requests."

Output: "The module creates authentication tokens, user role assignments, password reset links, and update profile success messages.

UML Diagram – Class Diagram



Methods	Descriptions
login(username: String, password: String) -> boolean	This approach takes a username and password and returns true if the login is successful.
logout(userId: String) -> void	Logs out the user session and renders invalid the authentication tokens.
registerUser(userDetails: Object) -> boolean	Creates a new user and allocates a role (Admin, Staff, or Customer).
resetPassword(email: String) -> boolean	Send an OTP or a password reset link to the registered email.
updateProfile(userId: String, profileDetails: Object) -> boolean	Respects users' modification of their own data and enforces security validations.

5.2 Session Management Module

Module Overview

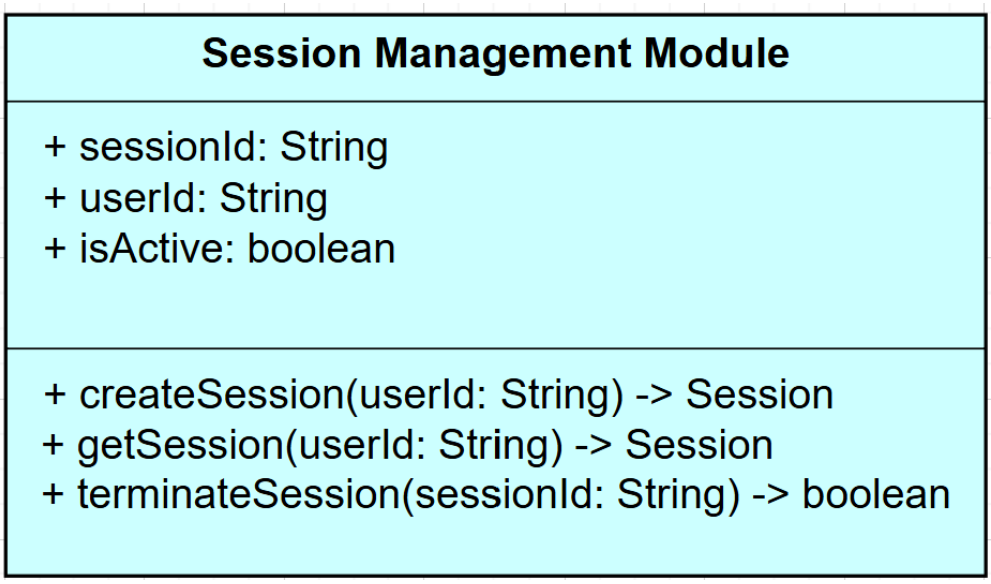
- **Responsibilities:** "This module enables user session management, secure login, and system use protection from malicious users. It generates session tokens on successful login and enables generation of user sessions to avoid sequential authentications. The user activity managing module enables automatic session expiration after inactivity and provides administrators with the option of manually invalidating suspicious-looking sessions. It has permitted active sessions to grant access and protection of system resources from attacks such as session hijacking, and it also provides auditing and tracing of security intrusions by allowing session log persistence."

- **Dependencies:** "This module will send messages to the Authentication Module to authenticate logins and initiate user sessions. It will exchange messages with the Logging Module to record session activity and monitor security intrusions. It also relies on the Database Module to cache session tokens and record user activity for authentication and auditing."

- **Input:** "User authentication data input, session initiation requests, authentication requests for current sessions, logout requests."

- **Output:** "The module generates session tokens, verifies valid sessions, closes stale sessions, and audits session usage for security."

UML Diagram – Class Diagram



Methods	Descriptions
createSession(userId: String) -> Session	Generates a session token upon successful login.
getSession(userId: String) -> Session	Retrieves the current session details for a user.
terminateSession(sessionId: String) -> boolean	Ends a session manually or after a timeout.

5.3 Car Inventory Management Module

Module Overview

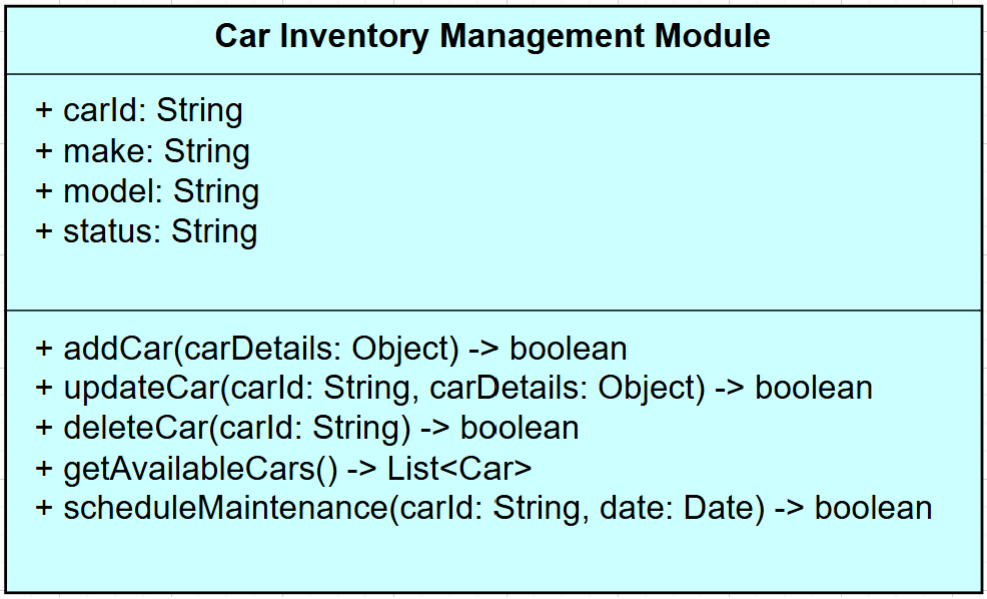
- **Responsibilities:** "This module is responsible for maintaining a current rental fleet by virtue of the function to add, update, and delete vehicle records. It tracks the availability of every car and does not pre-book only cars for hire. The module offers scheduling for maintenance in such a way that broken-down or out-of-service cars will not be pre-booked. It assists in managing fleet use efficiently, so cars are serviced from time to time and withdrawn from stock when needed. Through integration with booking operations, it provides real availability status to clients."

- **Dependencies:** "This module interacts with the Car Booking and Rental Management Module for confirming real-time car availability, thus preventing double booking. It also interacts with the Rental Return and Maintenance Scheduling Module for scheduling car maintenance and ascertaining their safety for driving." It also relies on the Database Module for the storage of vehicle details, maintenance records, and status updates of the fleet."

- **Input:** "The module receives vehicle details (make, model, registration number, price, availability status), update requests, deletion requests, and maintenance scheduling information."

- **Output:** "The module updates the car inventory, modifies vehicle availability, schedules maintenance tasks, and prevents booking of unavailable vehicles."

UML Diagram – Class Diagram



Methods	Descriptions
addCar(carDetails: Object) -> boolean	Adds a new vehicle to the rental fleet.
updateCar(carId: String, carDetails: Object) -> boolean	Modifies car information (e.g., price, availability).
deleteCar(carId: String) -> boolean	Removes a vehicle from the system (if not currently booked).
getAvailableCars() -> List<Car>	Retrieves a list of available vehicles for booking.
scheduleMaintenance(carId: String, date: Date) -> boolean	Marks a car for maintenance and prevents booking until resolved.

5.4 Car Booking & Rental Management Module

Module Overview

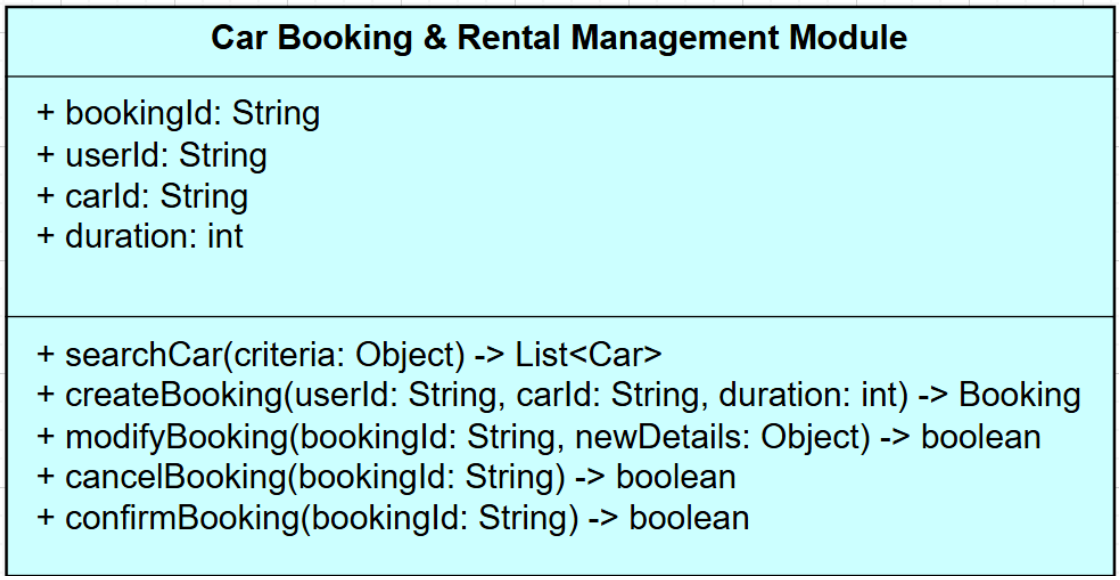
- **Responsibilities:** "This module deals with car rental process, including car search, booking, modification of the booking, and cancellation. It enables customers to make bookings according to availability and prevents overlapping with existing bookings. Furthermore, the module provides real-time updating of rental status, only completing rental transactions upon successful payment. It enables users to modify bookings, for instance, rental duration extension or car change, within system policy parameters. With integration with notifications, it reminds users about their rental dates and return due dates."

- **Dependencies:** "This module is shared with the Car Inventory Management Module to obtain available cars and car status updates on confirmation of booking. It also depends on the Payment Processing Module to receive payment before confirming booking. It also shares information with the Reporting & Analytics Module for tracking booking trends and rental performance."

- **Input:** "Customer rental requests, vehicle selection, rental period, change request, and cancellation requests are handled by the module."

- **Output:** "The module sends rental confirmations, tracks changes to vehicle availability, tracks rental status, and emails customers rental data and reminders.

UML Diagram – Class Diagram



Methods	Descriptions
searchCar(criteria: Object) -> List<Car>	Allow customers to filter available cars based on preferences.
createBooking(userId: String, carId: String, duration: int) -> Booking	Reserves a car for a specified period.
modifyBooking(bookingId: String, newDetails: Object) -> boolean	Adjusts existing reservations (e.g., changing rental dates).
cancelBooking(bookingId: String) -> boolean	Cancels a booking and updates inventory.
confirmBooking(bookingId: String) -> boolean	Finalizes a booking after payment completion.

5.5 Payment Processing Module

Module Overview

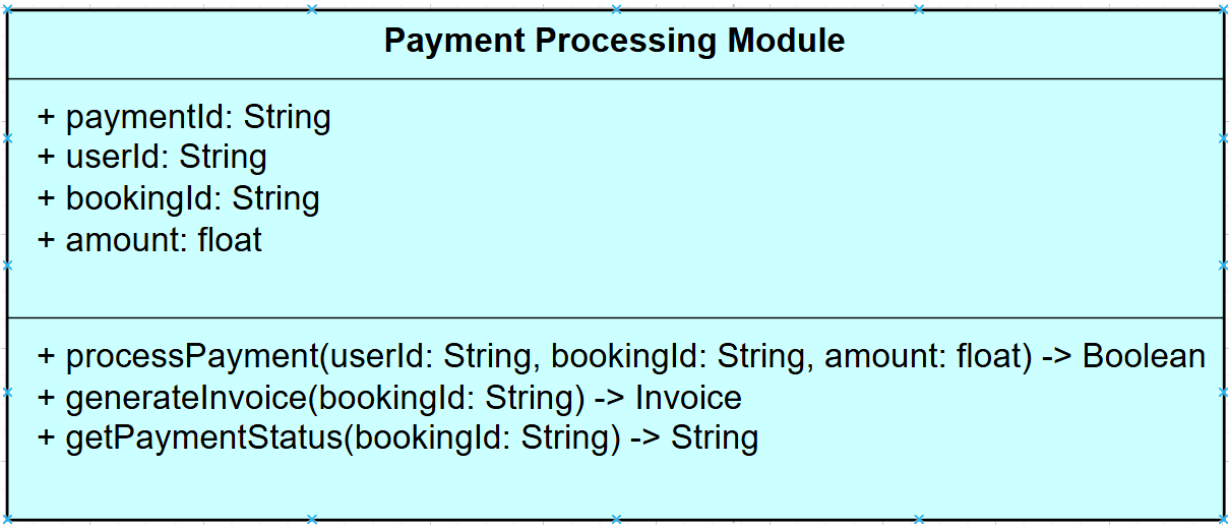
- **Responsibilities:** "This "This module processes rent payments securely, asking customers to pay before reserving. It checks payment information, calculates rental charges by length of stay, and charges taxes, discounts, or penalties as needed. The module integrates into payment gateways for credit card, digital wallet, or online banking processing. It produces invoices for successful payment and maintains books of accounts for auditing and compliance. With failed payment monitoring, it allows customers to retry payments and wards off scammers."

- **Dependencies:** "This module interacts with the Car Booking & Rental Management Module in an attempt to verify rental charges before payment. It interacts with the Database Module to make payment and create invoices. It also relies on the Reporting & Analytics Module in an attempt to create revenue tracking and finance reports."

- **Input:** "The module accepts user ID, payment method booking ID, print invoice requests, and transaction authorizations requests."

- **Output:** "The module remits, reports successful remittance, issues bill, displays status of booking and manages failed attempts at remittance."

UML Diagram – Class Diagram



Methods	Descriptions
processPayment(userId: String, bookingId: String, amount: float) -> boolean	Validates and processes rental payments.
generateInvoice(bookingId: String) -> Invoice	Generates a rental invoice after successful payment.
getPaymentStatus(bookingId: String) -> String	Retrieves payment status (Paid, Pending, Failed).

5.6 Rental Return & Maintenance Scheduling Module

Module Overview

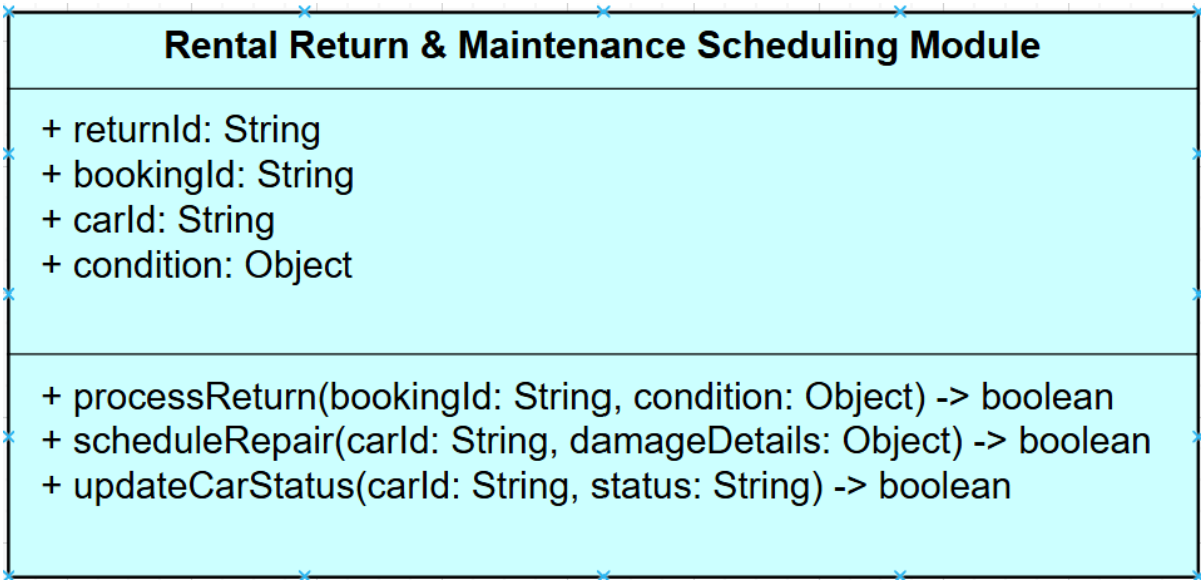
- **Responsibilities:** "This module is responsible for accepting automobiles in, tracking when rentals are due and scheduling maintenance be done when needed before dispatching cars for the next rental. Thus, it checks the car for damage, mileage, and level of gas when returned. The tool lets you plan when maintenance is done, so customers can't rent cars that are being fixed. It also changes the status of the fleet to show whether a vehicle is accessible, being serviced, or not working. Keeping track of how the cars are maintained helps fleet managers make smart choices about what to replace and fix."

- **Dependencies:** "It communicates with the Car Inventory Management Module to inform vehicle availability for updates upon maintenance and repairs. It also relies on the Car Booking & Rental Management Module for tracking which cars are due back. It also communicates with the Database Module for maintaining maintenance records, rental return history, and service schedules."

- **Input:** "The "The module takes in vehicle return data (booking ID, mileage, fuel level, reported damages), maintenance requests, and repair updates."

- **Output:** "The module logs vehicle return status, schedules required repairs, updates car availability, and prints maintenance reports."

UML Diagram – Class Diagram



Methods	Descriptions
processReturn(bookingId: String, condition: Object) -> boolean	Logs vehicle return details and inspects condition.
scheduleRepair(carId: String, damageDetails: Object) -> boolean	Assigns cars for repair based on damage reports.
updateCarStatus(carId: String, status: String) -> boolean	Updates car availability post-maintenance.

5.7 Reporting & Analytics Module

Module Overview

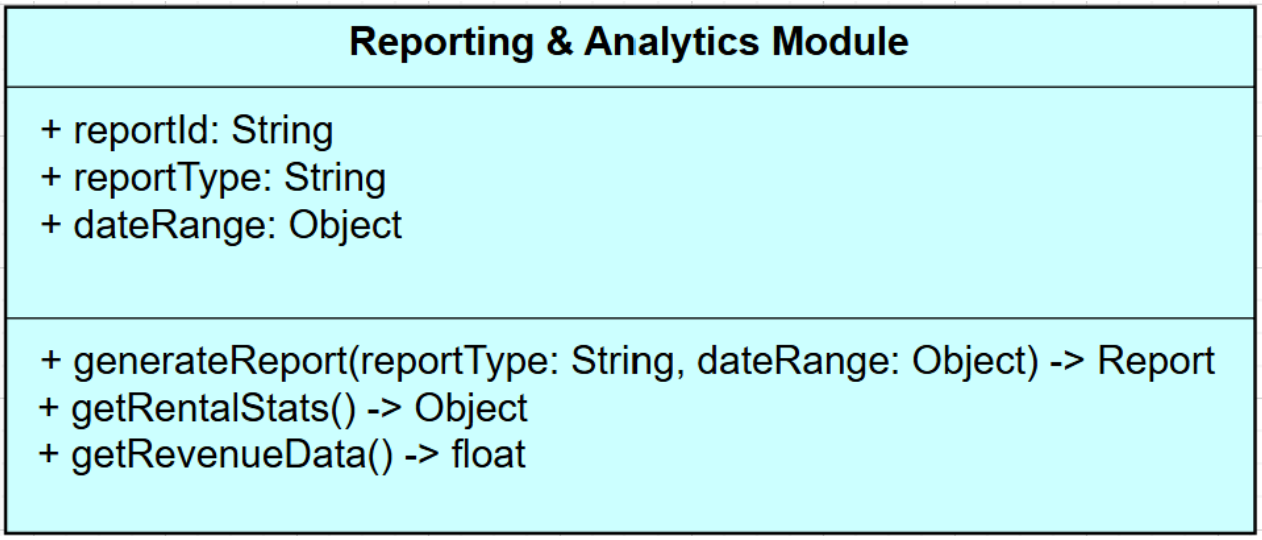
- **Responsibilities:** "The module is responsible for analysis of rental patterns, preparation of business reports, and tracking key performance indicators such as revenue, customer taste, and utilization of the fleet. It combines data from other system modules to prepare insights upon which the administrators can fine-tune price and fleet management. The module allows users to produce reports in various formats, such as PDFs, charts, and Excel spreadsheets, for business review. It also provides real-time analysis of peak rental time, most requested vehicles, and finance. Through data analysis, it helps businesses to make optimal decisions and maximize operation."

- **Dependencies:** "This module interacts with the Car Booking & Rental Management Module to track booking statistics and rental patterns. It also relies on the Payment Processing Module to obtain revenue information and analyze financial transactions. Furthermore, it is interfaced with the Database Module to save historical data and generate comprehensive business reports."

- **Input:** "The module accepts booking statistics, revenue analysis, fleet utilization reports, and customer rental trends requests."

- **Output:** "The module produces sophisticated reports in reports (PDF, Excel, graphs), provides rental trends and assists administrators in business decision-making."

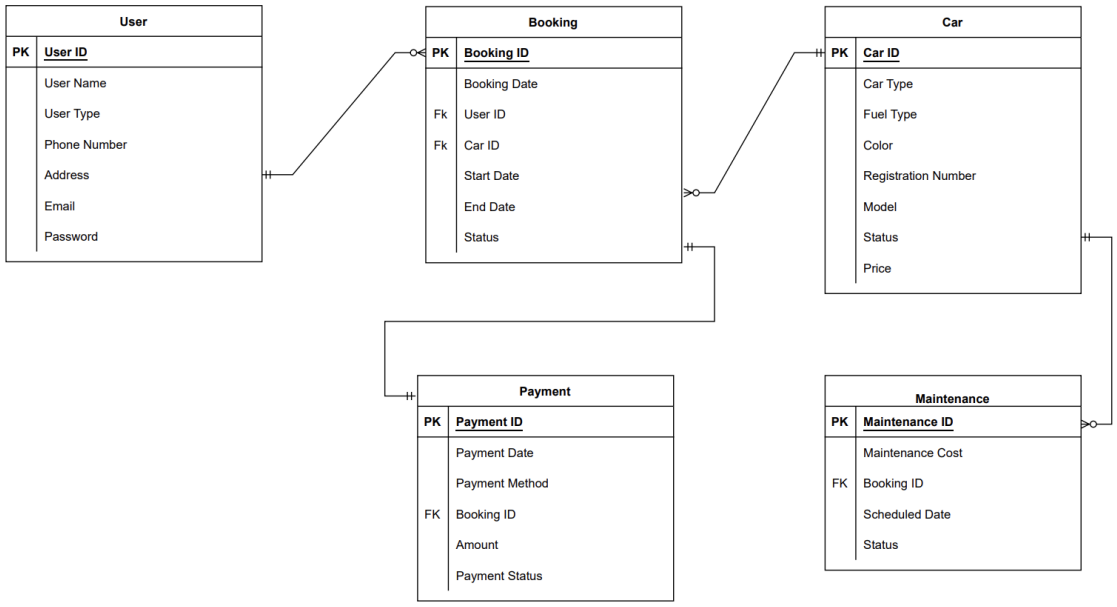
UML Diagram – Class Diagram



Methods	Descriptions
generateReport(reportType: String, dateRange: Object) -> Report	Creates customized business reports.
getRentalStats() -> Object	Provides insights into popular cars and peak booking times.
getRevenueData() -> float	Calculates total revenue for a selected period.

6 Database Design

6.1 Entity-Relationship (ER)Diagram



6.2 Database Normalization

To ensure data integrity and eliminate redundancy, the Car Rental System database is normalized up to Third Normal Form (3NF). Below is a step-by-step breakdown of the normalization process:

1. First Normal Form (1NF)

A database is in 1NF if:

- All columns contain atomic (indivisible) values.
- Each row has a unique identifier (Primary Key).
- There are no repeating groups in any table.

Applying 1NF to Our Database

- The User, Car, Booking, Payment, and Maintenance tables already satisfy 1NF since:
    - Each attribute contains atomic values.
    - There are no repeating groups (e.g., multiple phone numbers for a single user).
    - Each table has a Primary Key (User ID, Car ID, Booking ID, etc.)

2. Second Normal Form (2NF)

A database is in 2NF if:

- It meets all 1NF requirements.
- No partial dependencies exist (All non-key attributes must depend entirely on the primary key, not just part of it).

Fixing Partial Dependencies

- In the Booking table, User ID and Car ID are Foreign Keys but do not introduce partial dependencies.
- In the Maintenance table, Booking ID is a Foreign Key, ensuring maintenance records are linked to a booking without duplicating data.
- The Payment table relates to Booking ID correctly, preventing partial dependencies.

Since no attributes are partially dependent on only part of a composite key, the database already meets 2NF.

3. Third Normal Form (3NF)

A database is in 3NF if:

- It meets all 2NF requirements.
- There are no transitive dependencies (i.e., non-key attributes should only depend on the primary key).

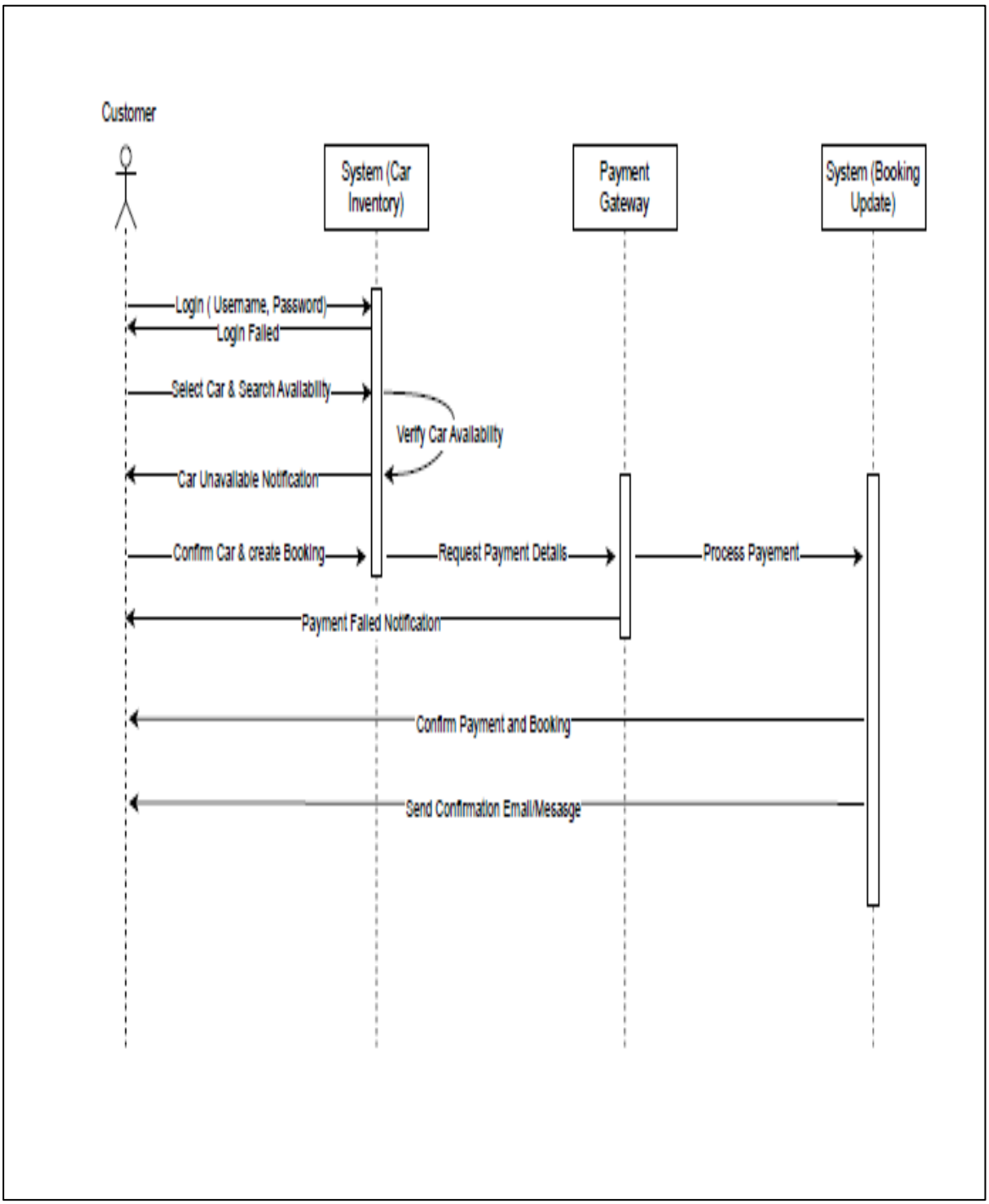
Fixing Transitive Dependencies

- User Table:
  - Username, User Type, Phone Number, Address, Email, and Password all depend only on User ID → No issue.
- Booking Table:
  - Booking Date, Start Date, End Date, and Status only depend on Booking ID → No issue.
- Car Table:
  - Car Type, Fuel Type, Color, Registration Number, Model, Status, and Price only depend on Car ID → No issue.
- Payment Table:
  - Payment Date, Payment Method, Amount, and Payment Status only depend on Payment ID → No issue.
- Maintenance Table:
  - Maintenance Cost, Scheduled Date, and Status only depend on Maintenance ID → No issue.

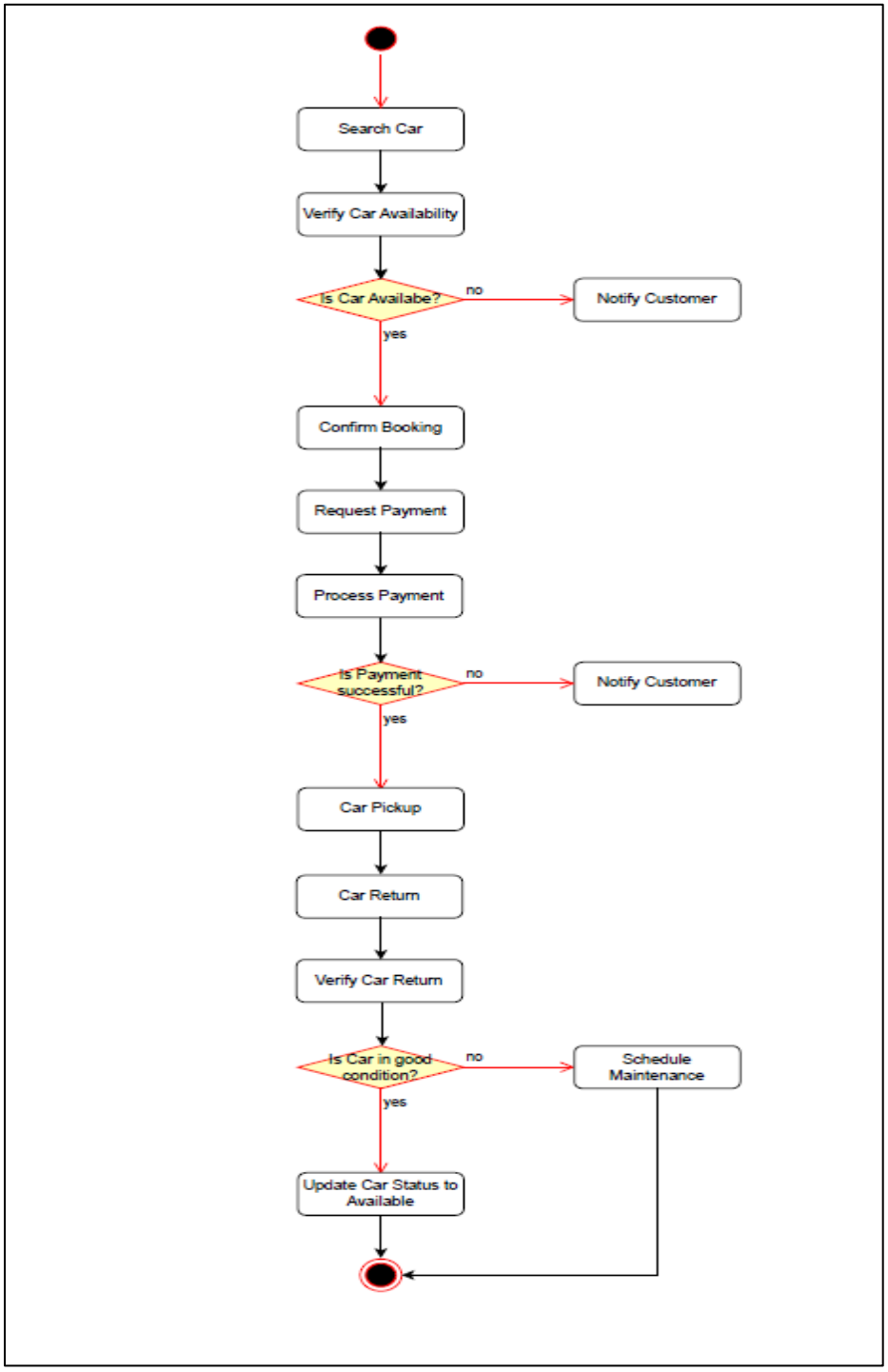
Since all attributes in each table only depend on their respective primary keys, the database meets 3NF.

7 System Flow Design

7.1 Sequence Diagram



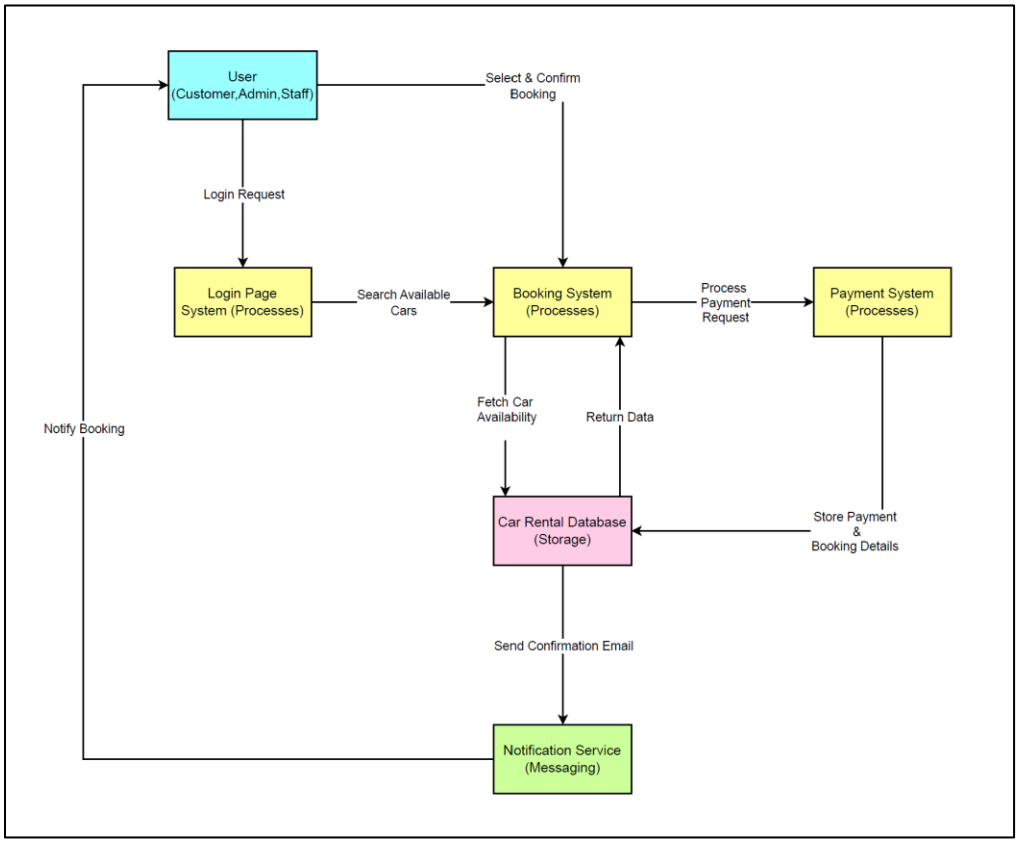
7.2 Activity Diagram



8 Interaction Between Components

8.1 Collaboration Diagram

The collaboration diagram indicates how various components of a car rental system cooperate.



- The **user (customer, admin, or staff)** logs in by sending a request, which is handled by the **Login Page** System.

- After logging in, the user finds available cars via the **Booking System**. The collaboration diagram indicates how various parts of the car rental system interact.

- The **customer, staff, or admin** first logs in by sending a request, which is then executed by the **Login Page System**.

- The logged-in customer searches for available cars using the **Booking System**, which then asks the **Car Rental Database** about availability and reports back.

- Once a car has been selected and booked, the **Booking System** requests payment from **the Payment System**.

- The **Payment System** makes the payment and updates the booking to the **Car Rental Database**.

- The **Notification Service** sends the user an email confirming the booking and notifies them of the booking.

- This diagram shows how all the components work together in harmony with processing a car rental booking.

- After a vehicle is selected and confirmed, the **Booking System** sends out a payment request to **Payment System**.

- The **Payment System** processes the payment and records the booking details to the **Car Rental Database**.

- Finally, the **Notification Service** generates an email notification to the customer and notifies him of the reservation.

This flowchart illustrates the way all parts function in conjunction to complete a car rental booking.

9 Non-Functional Requirements

9.1 Performance Considerations

Through optimized search queries and UI components the system provides efficient rental car searching and booking functionality. JavaFX UI components facilitate smooth and responsive user experience throughout all system interactions. Loading indicators called spinners display system status when executing background tasks like car searches and payment processing to confirm bookings.

9.2 Security

- **User Authentication & Role-Based Access:** Customers, Staff members and Admin personnel receive distinct access rights and permission levels. The system requires users to authenticate with username and password inputs which generate error messages when credentials are incorrect.

- **Password Policies:** Passwords must meet secure rules which include a minimum of 8 characters and at least one uppercase letter, lowercase letter and special character. OTP verification for password reset via email.

- **Payment Security:** The system validates payment details including credit card information as well as expiry date and CVV before it proceeds with processing the payment.

- **Data Privacy and Compliance:** The system securely stores user data and booking details behind restricted admin access controls. The terms and policies establish rules for eligibility requirements as well as payment procedures alongside privacy, liability, and refund guidelines.

9.3 Scalability

The modular system design enables teams to incorporate new features while maintaining current system operations. The system provides powerful search and filtering capabilities which efficiently manage extensive car inventory and booking datasets. Vehicle listings require pagination and sorting mechanisms to keep the system responsive.

9.4 Reliability

- **Error handling mechanisms:** Real-time inline red alert messages guide users through form input validation. Critical system errors trigger popup notifications to alert users.

- **Version Control:** The GitHub repository with version control capabilities supports smooth updates while maintaining system stability. The system uses the Gitflow strategy to manage design and code modifications.

- **Success and Error Feedback:** The interface displays green banners when users successfully complete actions such as booking confirmations and password resets. Validation errors prompt red messages when the system detects invalid email addresses or incorrect payment information.

10 System Interfaces

Multiple internal and external interfaces combine to support the Car Rental System's efficient operation.

10.1 External Interfaces

- **Payment Gateway:** The system guarantees safe payment transactions when users enter credit or debit card information. Validates billing information before confirming a booking.
- **Email Services:** The Email Services module delivers account verification emails as well as password reset OTPs and booking confirmation receipts to users. Send notifications for unsuccessful login attempts along with critical alerts.

10.2 Internal Interfaces

- **Database Interface:** The Database Interface manages user authentication as well as storage for rental car records and payment information. Provides live data updates for current bookings while keeping track of available vehicles and returned ones.
- **Admin Dashboard & Reports Page:** The Admin Dashboard & Reports Page provides data on total revenue figures and booking quantities along with system user statistics for customers, staff, and admins as well as various reports. Administrators can monitor business performance metrics along with car rental operations.

- **Car Management Interface:** Administrators can add new vehicles to the system as well as update existing ones and delete vehicles when necessary. The system shows vehicle details including name, transmission type, mileage, and rental price.

- **Booking Management System:** Users can search for cars, and both view their booking details and change or cancel their reservations. The staff is responsible for booking management for clients and monitoring both rented and returned vehicles.

11 Design Patterns

Key software design patterns implemented in the Car Rental System ensure enhanced scalability along with maintainability and flexibility.

11.1 Patterns Used: MVC (Model-View-Controller) Pattern

- **Model:** Manages data (Car details, bookings, users).
- **View:** The user interface of JavaFX consists of Forms, Tables, Buttons, and Booking pages.
- **Controller:** Handles user interactions (Search, Booking, Login, Payments).

➢ Justification:

- **Separation of concerns:** The system maintains distinct boundaries between business logic and UI components which streamline development processes.
- **Scalability:** New features can be implemented without needing to change the entire system.

11.2 Factory Pattern

The Factory Pattern creates user roles (Customer, Staff, Admin) programmatically.

➢ Justification:

- **Separation of concerns:** The system provides straightforward management options for user roles and permissions.
- **Scalability:** The account creation process is streamlined by unifying the user initialization logic.

11.3 Singleton Pattern

Database connection management utilizes the Singleton Pattern to maintain one instance across the system.

➢ Justification:

- **Separation of concerns:** Prevents multiple unnecessary database connections, improving performance.
- **Scalability:** Maintains uniform database access throughout the entire system.

11.4 Observer Pattern

This pattern serves notification purposes and update functions (e.g., real-time booking status changes).

➢ Justification:

- **Separation of concerns:** Implement a system where users get instant notifications about booking confirmations and payment successes.
- **Scalability:** By eliminating manual page refresh requirements users experience improved interaction with the system.

12 Detailed Algorithm Descriptions

Below are detailed descriptions and pseudocode for key algorithms used in the Car Rental System, focusing on core business logic functionalities.

12.1 User Authentication Algorithm

Description

This algorithm verifies user credentials during login.

Pseudocode

```
FUNCTION authenticateUser (username, password):
    RETRIEVE stored credentials from database USING username
    IF credentials found:
        IF stored password matches input password:
            RETURN successful login, redirect user based on role
    ELSE:
        DISPLAY "Invalid Username or Incorrect Password."
```

12.2 User Management (CRUD) Algorithm

Description

This algorithm allows creating, reading, updating, and deleting user records in the system.

Pseudocode

```
FUNCTION createUser (username, password, role):
    IF username NOT IN database:
        INSERT new user INTO database (username, password, role)
        RETURN "User created successfully"
    ELSE:
        RETURN "Username already exists"

FUNCTION readUser(username):
    RETRIEVE user details FROM database USING username
    IF user exists:
        RETURN user details
    ELSE:
        RETURN "User not found"

FUNCTION updateUser (username, newPassword, newRole):
    IF username IN database:
        UPDATE user record SET password = newPassword, role = newRole
        RETURN "User updated successfully"
```

```
    ELSE:
        RETURN "User not found"

FUNCTION deleteUser(username):
    IF username IN database:
        DELETE user record FROM database
        RETURN "User deleted successfully"
    ELSE:
        RETURN "User not found"
```

12.3 Car Availability Checking Algorithm

Description

Ensures the requested vehicle is available during the desired rental period.

Pseudocode

```
FUNCTION checkCarAvailability (carId, requestedStartDate, requestedEndDate):
    FETCH bookings for carId FROM database
    FOR each booking IN bookings:
        IF requested dates overlap WITH booking dates:
            RETURN FALSE
    RETURN TRUE
```

12.4 Car Booking Algorithm

Description

Enables customers to book cars based on their search criteria and availability.

Pseudocode

```
FUNCTION bookCar (customerId, carId, startDate, endDate):
    IF checkCarAvailability (carId, startDate, endDate):
        CREATE booking record IN database
        RETURN booking confirmation details
    ELSE:
        RETURN message "Car unavailable for selected dates"
```

12.5 Car Return and Condition Verification Algorithm

Description

Staff verifies the vehicle condition upon return and updates the system.

Pseudocode

```
FUNCTION processCarReturn (bookingId, conditionReport):
    RETRIEVE booking details FROM database
    VALIDATE return details (fuel level, mileage, damages)
    IF damage OR maintenance needed:
        UPDATE vehicle status TO 'Under Maintenance'
        SCHEDULE required maintenance
        NOTIFY customer OF additional charges (if applicable)
    ELSE:
        UPDATE vehicle status TO 'Available'
    UPDATE booking record WITH return details
```

12.6 Maintenance Scheduling Algorithm

Description

Schedules necessary maintenance based on vehicle condition after return.

Pseudocode

```
FUNCTION scheduleMaintenance (carId, maintenanceDetails):
    INSERT maintenance details INTO database
    UPDATE car status TO 'Under Maintenance'
```

12.7 Report Generation Algorithm

Description

Generates various reports such as revenue, vehicle usage, and rental history.

Pseudocode

```
FUNCTION generateReport (reportType, criteria):
    SWITCH reportType:
        CASE 'RentalHistory':
            FETCH rental history FROM database USING criteria
        CASE 'Revenue':
            FETCH revenue data FROM database USING criteria
        CASE 'VehicleUsage':
            FETCH vehicle usage data FROM database USING criteria
    FORMAT fetched data INTO report
    DISPLAY OR EXPORT report IN selected format (PDF, Excel)
```

12.8 Rental Order Validation Algorithm

Description

Validates rental orders by checking vehicle availability, customer eligibility, and payment verification before finalizing the booking.

Pseudocode

```
BEGIN OrderValidation (customerID, vehicleID, rentalPeriod, paymentInfo):
    IF VehicleAvailable (vehicleID, rentalPeriod) THEN
        IF CustomerEligible(customerID) IS TRUE:
            IF PaymentValidation (customerID, rentalCost) == SUCCESS:
                Confirm Rental
                Mark vehicle as rented for specified period
                Send confirmation to customer
            ELSE:
                Notify customer of payment failure
            ENDIF
        ELSE:
            Notify customer of vehicle unavailability
        ENDIF
END
```

12.9 Vehicle Search Algorithm

Description

Allows users to find vehicles based on criteria like make, model, availability, and price range.

Pseudocode

```
BEGIN VehicleSearch(criteria):
    INITIALIZE resultsList AS empty
    FOR EACH vehicle IN vehiclesDatabase DO
        IF vehicle matches criteria (e.g., type, availability, price range) THEN
            ADD vehicle to resultList
        ENDIF
    ENDFOR
    SORT resultList by user's chosen preference (price, rating, or availability)
    RETURN resultList
```

12.1 Payment Processing Algorithm

Description

Ensures secure and accurate payment processing by validating credit or debit card transactions.

Pseudocode

```
BEGIN PaymentProcessing (customerID, paymentDetails, amount)
    VERIFY paymentDetails (credit card validity, expiry, CVV)
    IF details valid:
        PROCESS payment through gateway
        IF transaction successful:
            RETURN payment confirmation
        ELSE:
            RETURN transaction failure message
        ENDIF
    ELSE:
        RETURN invalid payment details message
    ENDIF
END
```

13 Hardware & Software Requirements

13.1 Hardware Requirements

The Car Rental System is developed with minimal hardware complexity and is optimized for use on desktop or laptop computers. The hardware should meet the following minimum specifications:

- **Processor:** Intel Core i5 or equivalent
- **RAM:** 8 GB recommended (minimum of 4 GB)
- **Storage:** At least 256 GB HDD or SSD for efficient data handling
- **Display:** Minimum resolution of 1366x768 pixels
- **Peripheral Devices:** Keyboard and mouse for efficient user interaction

Specialized hardware components are not required for the operation of this system.

13.2 Software Requirements

The system employs a structured three-tier architecture that comprises Presentation, Business Logic, and Data layers. Below are the software specifications required to operate the Car Rental System efficiently:

Operating Environment

- **Operating System:** Windows 10/11 or compatible Linux distribution (e.g., Ubuntu 20.04+)

Frontend Development

- **Framework:** JavaFX for intuitive and responsive desktop user interfaces

Backend Development

- **Programming Language:** Java (JDK 17 or higher recommended)
- **Development Tools**
  - **IDE:** Eclipse
  - **Version Control:** GitHub
  - **Project Management:** Trello

Database Management

- **Database:** MySQL (Version 8.0+)
- Data stored includes user profiles, car inventory details, booking records, and maintenance schedules.

Communication and Data Handling

- Internal system communication through Java function calls.
- SQL queries for interaction between the backend and the database.

Security and Access Control

- **Authentication & Authorization:** Role-Based Access Control (RBAC)
- **Encryption:** Base64 encoding for user credentials and sensitive data encryption.

Additional Dependencies

- **Java Libraries:** Additional libraries for encryption, reporting, and efficient user interface components.

Performance and Constraints

- System designed to handle up to 100 simultaneous bookings without performance degradation.
- Database queries optimized response times of under 5 seconds for 95% of interactions.
- User interfaces and interactions maintain a response time under 30 seconds under normal operating conditions

14 Appendices

14.1 Glossary

MVC (Model-View-Controller)	Software design pattern splitting the application into three interconnected parts.
OTP (One-Time Password)	One-time password sent to users for the reason of identity authentication.
API (Application Programming Interface)	A collection of principles that allow computer programs to speak to one another.
JDBC (Java Database Connectivity)	Java database connection API.
Component Diagram	UML diagram that displays system components and their interactions.
Class Diagram	UML diagram showing system classes, attributes, methods, and interactions.
Sequence Diagram	Shows the order of interaction among system components.
Activity Diagram	Shows activity flows and activities in the system.
ER Diagram (Entity-Relationship Diagram)	Data entity representation and relationships in the database using a diagram.
Collaboration Diagram	A collaboration diagram is a graphical chart that shows how different components or objects in a system interact with each other in order to carry out a task or process.

14.2 References

- General Guidelines for preparing UI Documentation – provided by professor

Other References

Version	Date	Author	Description
1	2/7/2025	Pujan Bhuva, Deepasree Meena Padmanabhan, OgheneRukevwe Esegba, Simranjeet Kaur	COSC3506 001_Final Project Software Requirement Specification Car Rental System.pdf
2	2/17/2025	Pujan Bhuva, Deepasree Meena Padmanabhan, OgheneRukevwe Esegba, Simranjeet Kaur	COSC3506_001_Architecture &GUI Specification_CarRentalSystem.pdf