**Practical No-04**

**Aim: Modeling UML Use Case Diagrams and Capturing Use Case Scenarios**

**Aim of the Experiment:**

In this experiment, we will learn how use cases and actors can be captured and how different use cases are related in a system.

**Introduction:**

A **Use Case Diagram** is a platform that provides a common understanding for the end users, stakeholders, and developers about how the system interacts with external entities. It captures the **functional requirements** of a system, focusing on actors (users) and their interactions with the system.

For the **Garage Management System**, the use case diagram will define the functionalities available for different users like **garage owners, mechanics, and customers**, and how they interact with the system.

**Objectives:**

After completing this experiment, you will be able to:  
✅ Identify different **actors** and **use cases** for the **Garage Management System**.  
✅ Establish different **relationships** among use cases (such as **include** and **extend**).  
✅ Draw a **Use Case Diagram** for the system.

**Theory:**

A Use Case Diagram consists of the following elements:

1. **Actors:**
   * **Primary Actor:** The main users of the system.
   * **Supporting Actor:** External systems or entities interacting with the system.
2. **Use Cases:**
   * Represent the functionalities provided by the system.
3. **Graphical Representation – Diagram:**
   * A visual representation showing the relationship between actors and use cases.
4. **Association between Actors and Use Cases:**
   * Shows who interacts with what functionality.
5. **Use Case Relationships:**
   * **Include Relationship:** A use case **always** invokes another.
   * **Extend Relationship:** A use case is **optional** or occurs under **specific conditions**.

**Use Case Diagram for Garage Management System (GMS)**

**1️⃣ Actors in the System:**

**🔹 Primary Actors (Directly Interact with the System)**

1. **Customer** – Requests vehicle services, views invoices, and makes payments.
2. **Mechanic** – Performs repairs and updates service status.
3. **Garage Owner/Admin** – Manages customers, mechanics, and services.

**🔹 Supporting Actors (External Systems or Entities)**

1. **Payment System** – Handles online transactions.
2. **Spare Parts Supplier** – Provides required vehicle parts.

**2️⃣ Use Cases in the System:**

**🔹 Customer Functionalities:**

**Register/Login** – Customers sign up or log in to their accounts.  
 **Book Service** – Customers schedule vehicle servicing or repairs.  
 **Track Service Status** – Customers can check the status of their vehicle repair.  
 **Make Payment** – Customers complete payment online or in cash.  
 **View Invoice** – Customers get an invoice after the service is completed.

**🔹 Mechanic Functionalities:**

**View Assigned Jobs** – Mechanics check which services they need to perform.  
 **Update Service Status** – Mechanics update the progress of vehicle repairs.

**🔹 Garage Owner/Admin Functionalities:**

**Manage Customers** – Add, update, or remove customer information.  
 **Assign Mechanics** – Allocate mechanics for booked services.  
 **Manage Inventory** – Track spare parts availability and order new parts.  
 **Generate Reports** – View analytics related to services, earnings, and repairs.

**🔹 External System Interactions:**

**Process Payment** – The system connects with a payment gateway to handle online payments.  
 **Order Spare Parts** – If parts are unavailable, the system requests new parts from suppliers.

**3️⃣ Use Case Relationships:**

1. **Includes (🔄 Mandatory Dependency)**
   * **"Book Service" ⬅ includes ⬅ "Make Payment"** (Booking cannot be confirmed without payment).
   * **"Generate Invoice" ⬅ includes ⬅ "View Invoice"** (Once an invoice is generated, customers must be able to view it).
2. **Extends (⚡ Optional Condition-Based Functionality)**
   * **"Update Service Status" ➝ extends ➝ "Notify Customer"** (Customers are notified only when major updates occur).
   * **"Manage Inventory" ➝ extends ➝ "Order Spare Parts"** (Only when parts are unavailable).

**Case Study: Modeling UML Use Case Diagrams for a Garage Management System**

**Introduction**

In software engineering, **Use Case Diagrams** are essential for capturing system functionalities and their interactions with users. The **Garage Management System (GMS)** requires a structured approach to define user roles, their actions, and system behavior. This case study focuses on modeling **UML Use Case Diagrams** for the **Garage Management System** to ensure efficient requirement gathering and system design.

**Problem Statement**

A garage service center faced operational inefficiencies due to the lack of a structured system for managing appointments, tracking repairs, and handling customer interactions. There was no clear **user role identification** and no systematic **workflow representation**, leading to miscommunication and service delays.

**Objectives**

* Identify different **actors** (users) interacting with the system.
* Define various **use cases** representing system functionalities.
* Establish relationships between **actors and use cases** to understand system flow.
* Design a UML **Use Case Diagram** for effective software requirement analysis.

**Methodology**

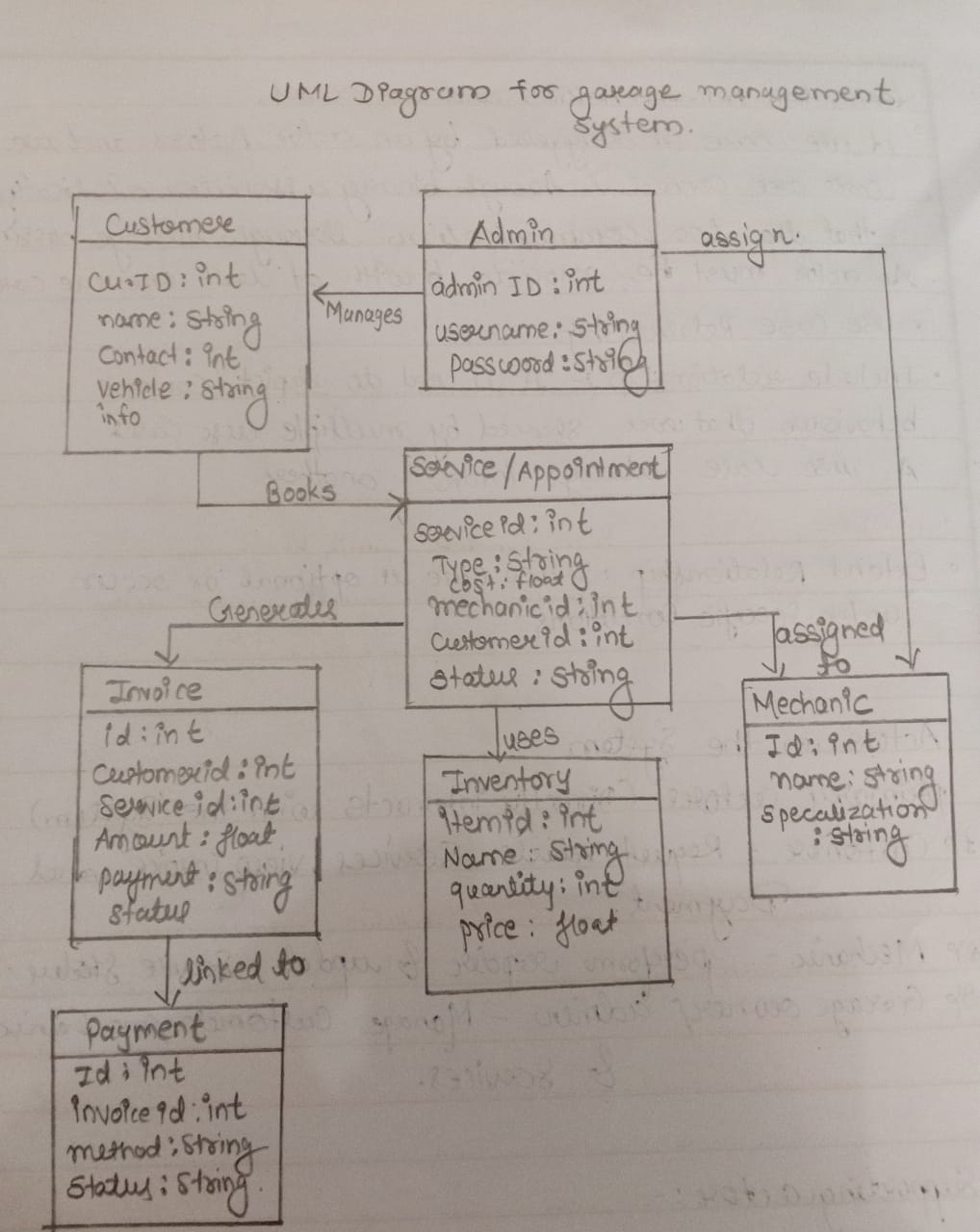
To address the problem, a **Use Case Diagram** was created using **UML modeling techniques**. The system was analyzed to define:

1. **Actors**
   * **Customer** – Books service appointments, checks vehicle status, makes payments.
   * **Mechanic** – Updates repair status, checks assigned tasks.
   * **Admin** – Manages appointments, assigns mechanics, tracks inventory.
2. **Use Cases**
   * Appointment Scheduling
   * Vehicle Repair Management
   * Spare Parts Inventory Management
   * Billing and Payment Processing
   * Report Generation
3. **Relationships**
   * **Customers interact** with appointment scheduling and payment.
   * **Mechanics interact** with vehicle repair and status updates.
   * **Admins control** all major functionalities.

**Results and Benefits**

* **Clear System Understanding**: The **Use Case Diagram** helped visualize the system structure and interactions.
* **Better Requirement Analysis**: It simplified communication between stakeholders (developers, managers, and end-users).
* **Efficient Workflow Design**: Defining use cases led to an optimized **service request and fulfillment cycle**.
* **Error Reduction**: Well-defined roles minimized **miscommunication and service delays**.

**Conclusion**

****The **Use Case Diagram** provided a clear representation of the **Garage Management System**, enabling effective **requirement analysis and system design**. This approach ensured that all stakeholders had a shared understanding of the functionalities before development, leading to **a well-structured, efficient, and scalable system**.