Database Management System

Semester-III(Batch-2024)

Vehicle Service Management System

Submitted To:
Sachin Garg

Submitted By:
Anshika (2410990846)
Anshul Lal (2410990850)
Abhinav Thakur (2410990815)

Department of Computer Science and Engineering
Chitkara University Institute of Engineering & Technology, Chitkara University, Punjab.

Project Title: Vehicle Service Management System

1. Introduction:

The Vehicle Service Management System is a database-based solution designed to manage customer, vehicle, service, employee, and payment records efficiently. It uses an ER model to represent key entities such as Employee, Department, Customer, Vehicle, Service, and Payment, along with their relationships.

This system ensures smooth handling of operations—linking customers to vehicles, tracking services and tasks performed by employees, managing departmental work, and recording payments. By organizing data in a structured way, it improves accuracy, accessibility, and efficiency in vehicle service centers.

2. Problem Statement:

Vehicle service centers deal with multiple interconnected entities such as Customers, Vehicles, Services,Employees, Departments, and Payments.In the absence of a well-structured management system, the following challenges arise:

Manual processing delays in registering customers, assigning services, and tracking progress.Data redundancy and inconsistency due to poorly maintained records across multiple registers or spreadsheets.Difficulty in tracking relationships between customers, vehicles, services, and payments.Lack of centralized access to information across departments (Mechanics, Cosmetics, Repair, etc.).Inefficient task allocation leading to poor employee utilization and delays in service completion.Security risks due to unprotected or misplaced service and payment records.Limited reporting and analytic, making it hard for management to assess performance, revenue, or customer service history.A well-designed Vehicle Service Management System eliminates these challenges by providing a centralized relational database with properly defined entities, relationships, and constraints. It ensures faster processing, better task allocation, secure record-keeping, and powerful reporting features to enhance customer satisfaction and workshop efficiency.

3. Scope of the Project:

The scope of this project is to design and implement a relational database system that will:Store all details of customers, vehicles, services, tasks, employees, departments, and payments in a structured way.Represent complex relationships between these entities (e.g., one customer owns many vehicles, one service involves many tasks, one employee may work on multiple services).Enable fast retrieval and updating of records through structured queries, minimizing manual delays.Provide centralized access to customer, vehicle, service, and payment data across all departments of the service center.Support task allocation by linking employees and departments with specific service tasks.Ensure data consistency and integrity by applying relational constraints and avoiding redundancy.Allow secure storage and controlled access to sensitive data such as payments, customer contacts, and employee salaries.Generate reports and analytics on service history, revenue, employee workload, and customer trends for better decision-making.

4. Objectives:

The primary objectives of the Vehicle Service Management System (VSMS) are: Centralized Data Management, Efficient Data Retrieval, Relationship Mapping, Data Integrity and Consistency, Support for Reporting and Analytics.

5. Significance of the Project:

The proposed Vehicle Service Management System (VSMS) will significantly improve the efficiency of vehicle service centers by:Reducing operational delays through automation of service booking, task allocation, and payment processing.Improving accuracy and reliability of records by maintaining a centralized database for customers, vehicles, services, employees, and payments.Enhancing customer satisfaction by providing faster service updates, clear billing, and transparent service tracking.Optimizing resource utilization by effectively assigning employees to tasks based on their department and expertise.Increasing security of data by protecting sensitive customer, employee, and payment records with proper access controls.Facilitating decision-making by generating reports on customer trends, service performance, employee workload, and revenue.Supporting scalability by ensuring that the system can easily adapt to growing customer bases, new vehicle types, and additional services in the future.

Core Functional Requirements (Entities and Relationships):

1. **Employee Entity**
   - Attributes:
     - employee_id (PK) → Primary key, unique identifier for each employee.
     - name → Employee's full name.
     - phone_no → Contact number (unique).
     - salary → Monthly or yearly salary of the employee.
     - Department_id(FK) → Department in which the employee works.
2. **Department Entity**
   - Attributes:
     - department_id (PK) → Primary key, unique identifier for each department.
     - department_name → Name of the department (Cosmetics, Mechanics, Repair, Manager).
3. **Service Entity**
   - Attributes:
     - service_id (PK) → Primary key, unique identifier for each service.
     - customer_id (FK) → References the customer who requested the service.
     - vehicle_id (FK) → References the vehicle being serviced.
     - Dept_id(FK) → References to the department.
     - Description →Type of service performed.
4. **Customer Entity**
   - Attributes:
     - customer_id (PK) → Primary key, unique identifier for each customer.
     - name → Customer's full name.
     - phone_no → Contact number (unique).
     - address → Customer's address.
     - email → Customer's email (unique).
5. **Vehicle Entity**
   - Attributes:
     - vehicle_id (PK) → Primary key, unique identifier for each vehicle.
     - model → Vehicle model.
     - brand → Vehicle brand.
     - registration_no (Unique) → Vehicle registration number.
     - Customer_id(FK) →References to the customer who owns vehicle.
6. **Payment Entity**
   - Attributes:
     - payment_id (PK) → Primary key, unique identifier for each payment.
     - service_id (FK) → References the service for which the payment is made.
     - Customer_id(FK) →References the Customer who paid.
     - amount → Amount paid by the customer.
     - payment_method → Mode of payment (Cash, Online, Check).

7. **Employee_Service (Junction table for M:N)**
    - Attributes:
        - Emp_id(FK) →References Employee.
        - Service_id(FK) →References Service.

Relationships:

Entity 1	Entity 2	Relationship	Cardinality	Explanation
Employee	Department	Works_In	M:1	Many employees work in one department.
Employee	Service	Assigned_To	M:N	One employee can be assigned to many services, and one service may need many employees.
Department	Service	Handles	1:M	One department handles multiple services.
Service	Customer	Requested_By	M:1	Many services can be requested by one customer.
Service	Vehicle	Has	M:1	Many services can be performed on one vehicle.
Customer	Vehicle	Owns	1:M	One customer can own multiple vehicles.
Service	Payment	Paid_By	1:1	One service generates exactly one payment, and one payment belongs to one service.
Customer	Payment	Makes	1:M	One customer can make many payments over time.

Relational Database Schema:

1.Customer: (Cust_ID (PK), Cust_Name, Cust_Email (Unique), Cust_Phone (Unique), Cust_Address (Street, City, Pincode))

2. Vehicle: (Veh_ID (PK), Cust_ID (FK), Model, Brand, Registration_No (Unique))

3. Service: (Serv_ID (PK), Veh_ID (FK), Cust_ID (FK), Dept_ID (FK), Description)

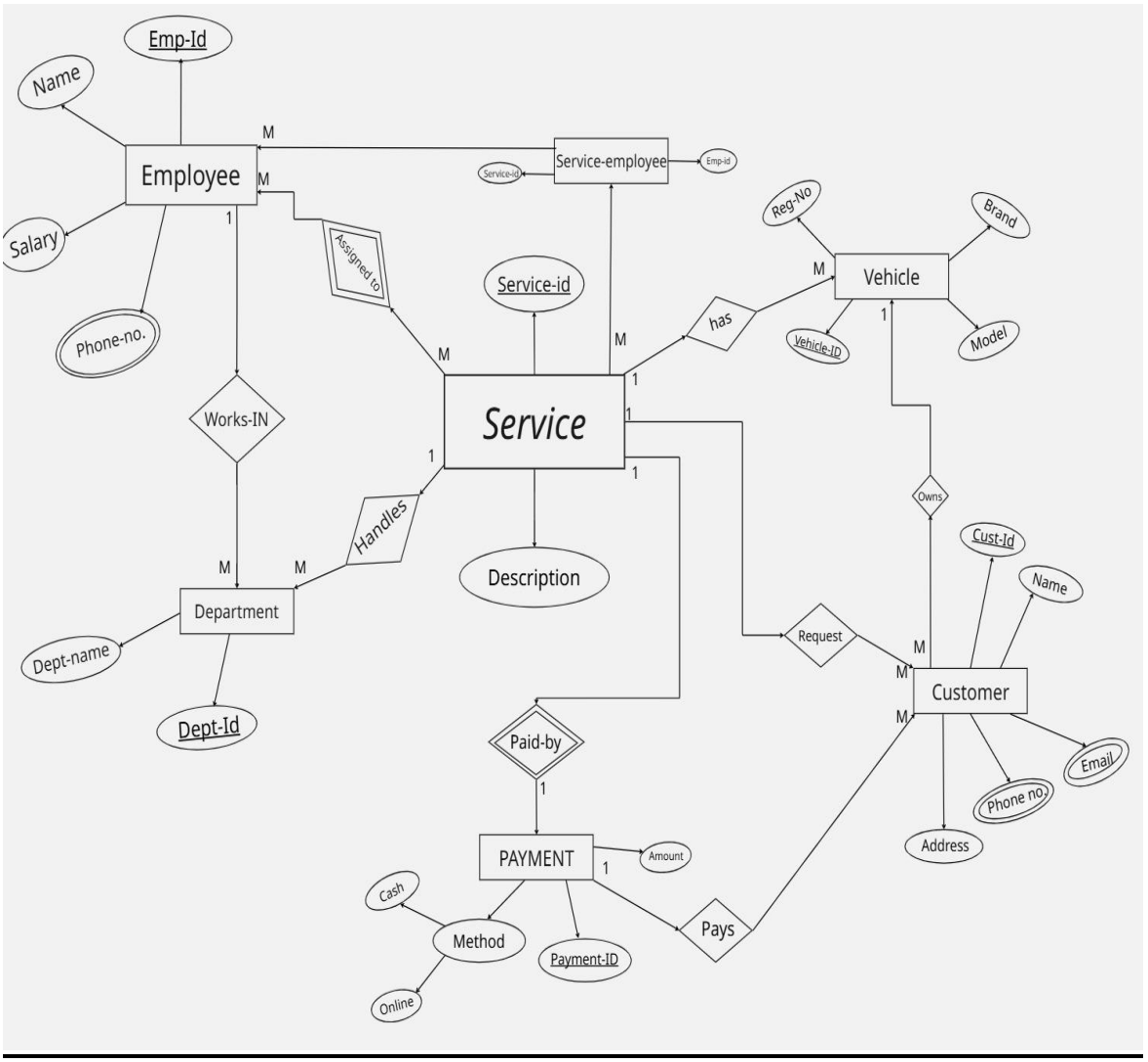
4. Employee: (Emp_ID (PK), Emp_Name (F_Name, L_Name), Phone (Unique), Salary, Dept_ID (FK))

5. Department: (Dept_ID (PK), Dept_Name)

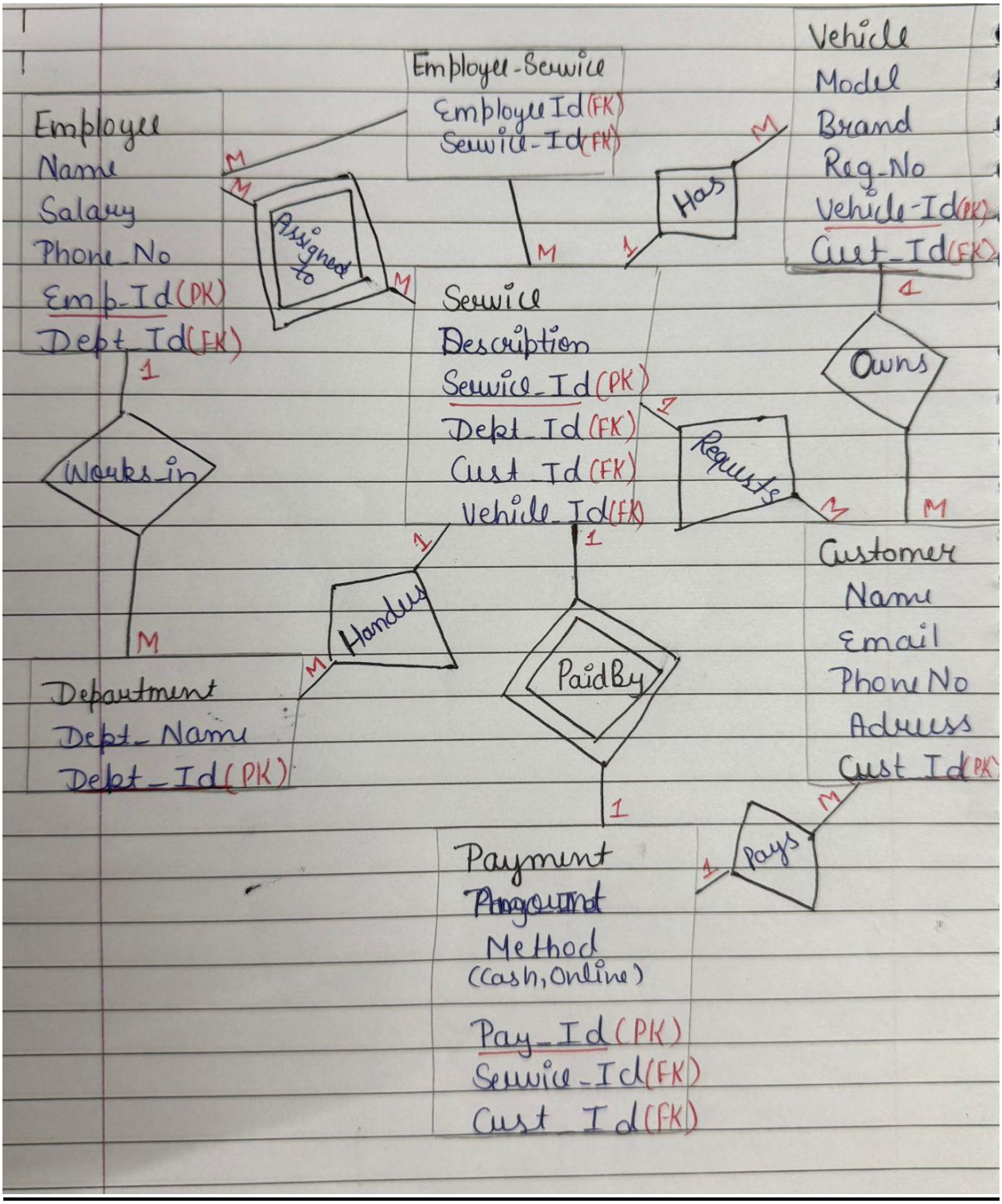
6. Payment: (Pay_ID (PK), Serv_ID (FK, UNIQUE), Cust_ID (FK), Amount, Payment_Method (Cash/Online))

7. Employee_Service: (Emp_ID (FK), Serv_ID (FK), PRIMARY KEY(Emp_ID, Serv_ID)) ← for M:N

ER Diagram


Emp-Id
Name
Employee
Salary
Phone-no.
Service-employee
Service-id
Emp-id
Assigned to
Service-id
Reg-NO
Brand
Vehicle
Vehicle-ID
Model
has
Works-IN
Service
Handles
Description
Owns
Cust-Id
Name
Department
Dept-name
Dept-Id
Request
Customer
Email
Phone no.
Address
Paid-by
PAYMENT
Amount
Cash
Method
Online
Payment-ID
Pays
M
1

ER Model


Employee
Name
Salary
Phone_No
Emp_Id(PK)
Dept_Id(FK)
Employee-Service
Employee Id(FK)
Service-Id(FK)
Assigned to
Service
Description
Service_Id(PK)
Dept_Id(FK)
Cust_Id(FK)
Vehicle_Id(FK)
Has
Vehicle
Model
Brand
Reg_No
Vehicle_Id(PK)
Cust_Id(FK)
Owns
Requests
Works_in
Handles
Department
Dept_Name
Dept_Id(PK)
PaidBy
Payment
Amount
Method
(Cash, Online)
Pay_Id(PK)
Service_Id(FK)
Cust_Id(FK)
Pays
Customer
Name
Email
Phone No
Adress
Cust_Id(PK)

Queries

1. Find employees with salary > 40000
   → σ_salary>40000(Employee)
2. Find employees in Dept_ID = 3
   → σ_dept_id=3(Employee)
3. Get customers from 'Delhi'
   → σ_city='Delhi'(Customer)
4. Find vehicles of brand 'Honda'
   → σ_brand='Honda'(Vehicle)
5. Get payments greater than 10000
   → σ_amount>10000(Payment)
6. Get all employee names
   → π_name(Employee)
7. List all employee phone numbers
   → π_phone_no(Employee)
8. Show all customer emails
   → π_email(Customer)
9. Get all vehicle registration numbers
   → π_registration_no(Vehicle)
10. List all department names
    → π_department_name(Department)
11. Get all names of employees and customers
    → π_name(Employee) ∪ π_name(Customer)
12. List all phone numbers of employees and customers
    → π_phone_no(Employee) ∪ π_phone_no(Customer)
13. Show all IDs (employee_id and customer_id)
    → π_employee_id(Employee) ∪ π_customer_id(Customer)
14. Get all service IDs from Service and Payment
    → π_service_id(Service) ∪ π_service_id(Payment)
15. List all department names and vehicle brands
    → π_department_name(Department) ∪ π_brand(Vehicle)
16. Find vehicles that have not been serviced
    → π_vehicle_id(Vehicle) − π_vehicle_id(Service)
17. Find customers who haven't made any payments
    → π_customer_id(Customer) − π_customer_id(Payment)
18. Get service IDs that exist in both Service and Payment
    → π_service_id(Service) ∩ π_service_id(Payment)
19. Get employee IDs that appear in both Employee and Employee_Service
    → π_emp_id(Employee) ∩ π_emp_id(Employee_Service)
20. Get all possible pairs of employees and departments
    → Employee × Department
21. Get all possible pairs of customers and vehicles
    → Customer × Vehicle
22. Get employee names with their department names
    → π_name,department_name(Employee ⋈ Department)

23. List customer names with their vehicle registration numbers
    → π_name,registration_no(Customer ⋈ Vehicle)
24. Find customer names who made payments
    → π_name(Customer ⋈ Payment)
25. Find services performed on vehicles with their customer names
    → π_description,registration_no,name(Service ⋈ Vehicle ⋈ Customer)
26. Get employee names and salaries with their department names
    →π_name,salary,department_name(Employee⋈Employee.dept_id=Department.dept_id Department)
27. Left Outer Join → List all customers and their payments (include customers without payments)
    → Customer ⟕ Payment
28. Right Outer Join → List all services and their payments (include services without payments)
    → Service ⟖ Payment
29. Full Outer Join → Get all customers and payments (include unmatched)
    → Customer ⟗ Payment
30. Rename Customer as C and Employee as E
    → ρ C(Customer), ρ E(Employee)