Customer Relationship Management System
Version 1.0 Approved

Prepared by: Shrasti Jadia

HCL Training

Date: - 01/01/2026

Software Engineering(Agile Methodology)

Table of Contents

1 . **Abstract**

A Customer Relationship Management (CRM) System using Agile Methodology is designed to manage customer information, improve communication, and enhance customer satisfaction. This system helps organizations store and access customer data, track leads, record interactions, and manage follow-ups effectively. Agile methodology is used to develop the CRM in iterative sprints, allowing continuous improvement, faster delivery, and flexibility in changing requirements. Each sprint delivers a working module such as login, customer management, lead tracking, task reminders, and reporting. The Agile approach ensures better collaboration among team members and stakeholders, resulting in a reliable, user-friendly, and scalable CRM solution.

2. Introduction

2.1 Introduction

A **Customer Relationship Management (CRM) system** is a software-based solution that helps organizations manage and strengthen relationships with both existing and potential customers. In today's highly competitive business world, maintaining strong customer relationships is essential for achieving business growth, increasing customer satisfaction, and ensuring long-term profitability.

The CRM system acts as a **centralized platform** where all customer-related information is stored in one place. This includes customer personal details, contact information, purchase history, interaction records, feedback, and preferences. By maintaining a single database, the organization can minimize data duplication, reduce manual errors, and ensure timely communication and follow-ups with customers.

This system is specially designed to improve the efficiency of various departments such as **sales, marketing, and customer support**. Sales teams can manage leads, track opportunities, and monitor customer engagement. Marketing teams can plan campaigns, analyze customer responses, and target the right audience. Customer support staff can instantly access customer history, enabling faster issue resolution and improved service quality.

Additionally, CRM systems provide valuable **analytics and reporting tools** that support management in making data-driven decisions.

2.2 Problem Identification

In many organizations, managing customer relationships and interactions efficiently remains a major challenge. Most businesses still depend on traditional methods such as maintaining customer data in **spreadsheets, paper records, or multiple disconnected applications**. These outdated approaches often create confusion, delays, and operational inefficiencies, making it difficult to provide timely and quality service to customers.

Some of the major issues faced are as follows:

a. Data Redundancy and Inaccuracy:

Customer details are frequently repeated across different files or systems. This leads to duplicate records, data inconsistency, and incorrect customer information.

b. Poor Lead and Opportunity Management:

Without a centralized platform, the sales team finds it difficult to track leads, schedule follow-ups, and monitor opportunities, which may result in missed conversions.

c. Delayed Customer Support and Communication:

When customer inquiries or complaints are not handled quickly, it affects customer satisfaction and reduces trust in the organization.

Due to these challenges, organizations may experience **loss of potential sales**, reduced productivity, and poor customer experience.

2.3 Need of the Project

The need for a **Customer Relationship Management (CRM) system** arises due to the difficulties many organizations face in managing customer data and interactions efficiently. In a fast-growing and competitive business environment, building strong customer relationships is essential for increasing customer retention, improving sales performance, and achieving long-term profitability.

The CRM system is necessary for the following reasons:

a. Centralization of Customer Information:

To store and maintain all customer details in a single centralized platform, reducing duplication, improving accuracy, and ensuring easy access to data.

b. Improvement in Sales Management:

To systematically manage leads, opportunities, and follow-ups so that sales teams can monitor progress and convert prospects into customers more effectively.

c. Better Customer Support and Service:

To provide quick access to customer history, queries, and complaint records, allowing support teams to resolve issues faster and deliver better service.

d. Support for Data-Driven Decision Making:

To generate accurate reports and analytics that help management understand trends, forecast sales, evaluate team performance, and make strategic business decisions.

2.4 Project Scheduling

The **Customer Relationship Management (CRM) System** project will be developed in multiple phases over a total period of **9 weeks**. Each phase includes specific tasks, deliverables, and milestones to ensure systematic progress and successful completion of the project. Proper scheduling helps in timely development, effective monitoring, and smooth coordination between the development team and stakeholders.

Project Phases and Key Activities

1) Requirement Analysis (1 Week)

Conduct meetings and discussions with stakeholders to gather **functional** and **non-functional** requirements.

Study the existing manual/legacy system and identify the current issues and limitations.

Prepare and finalize the **Software Requirements Specification (SRS)** document.

- **Milestone:** Completion of the SRS document by the end of **Week 1**.

2) System Design (2 Weeks)

Design the overall system architecture and prepare **database schema**.

Create diagrams such as **Data Flow Diagrams (DFDs)**, **ER Diagram**, and **Class Diagram**.

Design user interface layouts and define module interactions.

Review the system design with stakeholders and incorporate feedback.

- **Milestone:** Completion of system design by the end of **Week 3**.

3) Implementation / Coding (3 Weeks)

Develop major modules such as:

Customer Management

Lead & Opportunity Tracking
Sales and Marketing Module
Reporting and Analytics

Integrate all modules with the database.

Perform **unit testing** along with development to detect and resolve issues early.

- **Milestone:** Completion of implementation and coding by the end of **Week 6**.

4) Testing (2 Weeks)

Perform testing phases including:

Unit Testing
Integration Testing
System Testing

Conduct **performance testing** and **security testing**.

Fix identified bugs and ensure the system meets all requirements.

- **Milestone:** Completion of complete testing by the end of **Week 8**.

5) Deployment & Documentation (1 Week)

Deploy the CRM system on a web server/host environment.

Prepare final documentation such as:

User Manual
System Documentation
Project Report

Conduct final review, acceptance testing, and finalize submission.

- **Milestone:** Final deployment and project submission by the end of **Week 9**.

2.5 Objectives

The **Customer Relationship Management (CRM) System** is designed to simplify customer handling, improve business operations, and deliver meaningful insights for effective decision-making. The objectives of this project are categorized into **Functional, Business, Technical, Operational, and User-Oriented objectives**, as explained below:

A) Functional Objectives

Maintain a **centralized and well-structured customer database** that stores customer details such as personal information, contact details, purchase history, and communication records.

Enable effective **lead and opportunity tracking** to manage sales pipelines and improve conversion.

Support **marketing automation** by planning campaigns and monitoring customer engagement for better targeting.

B) Business Objectives

Improve **customer satisfaction and loyalty** through timely service, personalized interactions, and continuous communication.

Increase **sales productivity and conversion rate** by systematically managing leads and opportunities.

Reduce manual effort, data duplication, and human errors, leading to improved workflow efficiency.

Support management in **strategic decision-making** using analytics, trends, and performance insights.

C) Technical Objectives

Develop a **secure CRM system** with password-protected login, encryption for sensitive data, and access control policies.

Ensure system stability through a **robust architecture** and reliable database management.

Make the system **web-based and responsive**, so it can be accessed on desktops, laptops, and smartphones.

D) Operational Objectives

Reduce the time spent on repetitive tasks such as data entry, follow-ups, and report creation using automation features.

Provide a system that is easy to maintain and upgrade with minimal interruptions in services.

Enable quick adoption by designing an **interactive, simple, and user-friendly interface** for smooth onboarding.

E) User-Oriented Objectives

Ensure easy navigation for users to access customer records, sales progress, and marketing information.

Provide interactive and visual dashboards to help users understand performance quickly.

Support team collaboration through shared access to customer notes, updates, and interaction history.

Allow customization in reports, notifications, and user preferences according to individual roles and responsibilities.

3. Software Requirement Specification (SRS)

3.1 Purpose

The purpose of this **Software Requirement Specification (SRS)** document is to define the complete requirements for the **Customer Relationship Management (CRM) System**. This document provides a clear understanding of what the system will do, how it will perform, and what features it must include.

The SRS serves as a guideline for:

a. Developers to design and implement the CRM system

b. Testers to validate functionalities

c. Stakeholders to verify that the system meets business needs

d. Users to understand the expected outputs and system capabilities.

3.2 Scope

The **CRM System** is a web-based application designed to manage customer-related data and business interactions efficiently.

Scope of the CRM System includes:

Customer data management (Add, Update, Delete, View customers)

Lead and opportunity tracking

Follow-up reminders and task management

Logging customer interactions (calls, emails, meetings)

Marketing campaign planning and tracking.

3.3 Hardware Requirements / Software Requirements
--Hardware Requirements

Component	Requirement
Processor	Intel i3 / AMD equivalent or above
RAM	4 GB (Recommended: 8 GB)
Storage	200 GB minimum free space
Monitor	14-inch or above
Internet	Stable internet connection for web access

-- Software Requirements

Software	Requirement
Operating System	Windows 10/11 / Linux
Web Browser	Google Chrome / Mozilla Firefox
Backend	Java (Servlet/JSP) / any server-side tech
Database	MySQL / PostgreSQL
Server	Apache Tomcat
Frontend	HTML, CSS, JavaScript
IDE	Eclipse / IntelliJ / NetBeans

3.4 Tools

The tools used for developing and testing the CRM system are:

A. Development Tools

IDE: Eclipse / IntelliJ IDEA / NetBeans

Database Tool: MySQL Workbench / phpMyAdmin

Server: Apache Tomcat (for deploying the project)

Frontend Tools: HTML, CSS, JavaScript

Backend Technology: Java, Servlet, JSP, JDBC

B. Testing Tools

JUnit / Manual Testing

Postman (for testing APIs if applicable)

C. Documentation Tools

MS Word / Google Docs (Project report documentation)

MS PowerPoint (Presentation preparation)

D. Version Control Tools

Git (source code management)

GitHub (repository hosting, collaboration)

3.5 Software Process Model

For this project, the **Agile Software Development Model** is used. Agile is an iterative and incremental development approach that allows continuous improvement, flexibility, and fast delivery of working software.

Agile is chosen because it allows iterative development, continuous testing, and frequent feedback from stakeholders, which ensures that the system meets user requirements effectively and efficiently. In this approach, the development is divided into multiple sprints, with each sprint focusing on specific modules such as Customer Management, Lead Tracking, Sales, Marketing, and Reporting. Each sprint includes design, coding, and testing, allowing early detection of errors and providing the flexibility to make changes based on stakeholder feedback. Agile also emphasizes collaboration among developers, testers, and users, ensuring that functional and business objectives are aligned throughout the project lifecycle.

Key features of the Agile model for this CRM project include:

a. Iterative and incremental development to deliver functional modules progressively. Regular stakeholder reviews and feedback to refine requirements and design.
b. Flexibility to accommodate changing requirements or new features during development.
c. Focus on delivering a usable system after each sprint to enable early adoption and testing.

4. System Design

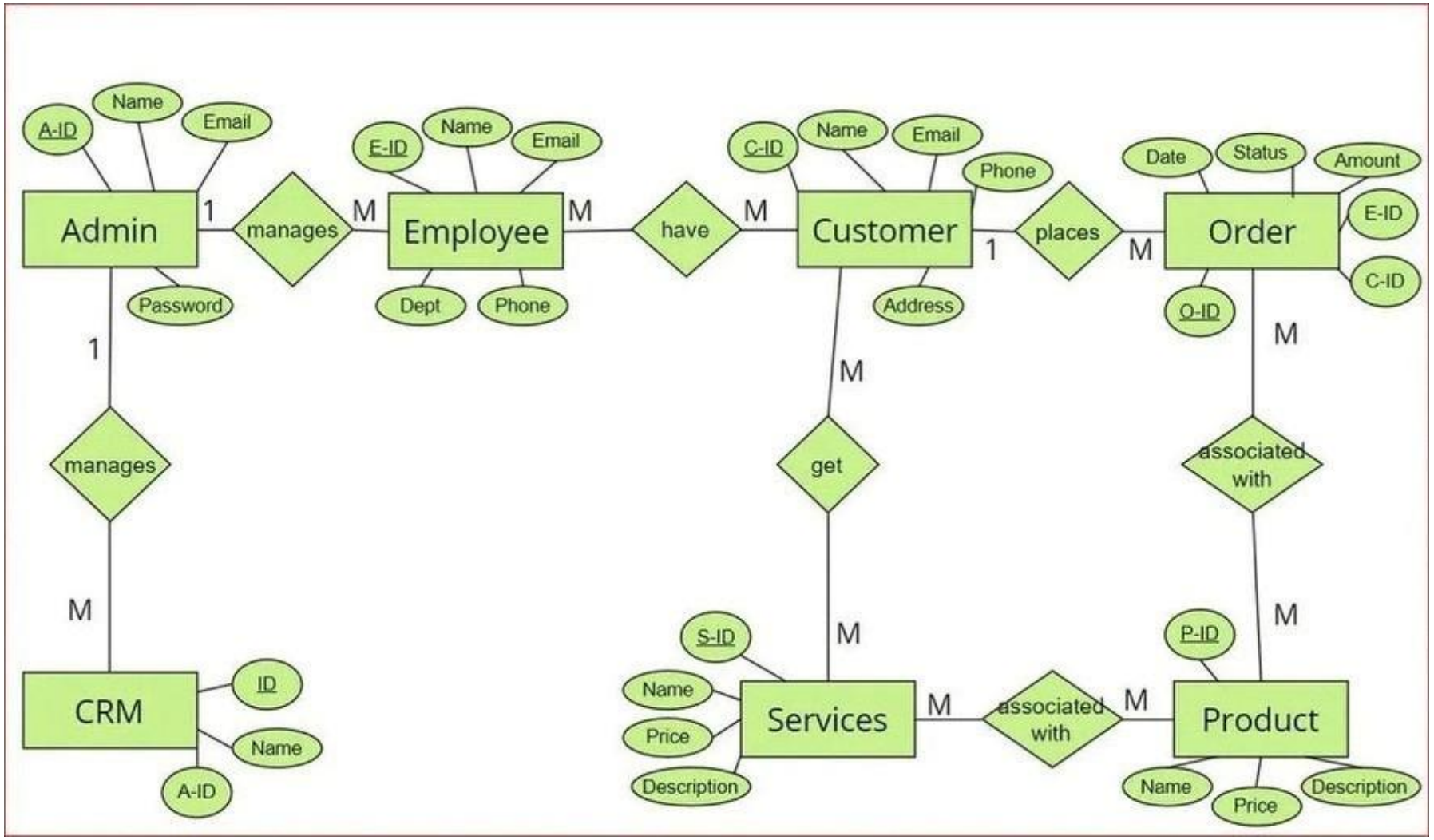
System Design is an important stage of software development where the overall structure of the CRM system is planned. It defines how the system will work internally, how data will flow, what database tables are required, and how different modules interact with each other. The design ensures the CRM system is scalable, secure, easy to use, and well-organized.

4.1 Data Dictionary

Field Name	Data Type	Size	Description	Constraint
user_id	INT	-	Unique ID of user	Primary Key, Auto Increment
name	VARCHAR	50	User full name	NOT NULL
email	VARCHAR	100	User email	UNIQUE, NOT NULL
password	VARCHAR	100	Encrypted password	NOT NULL
role	VARCHAR	20	User role (Admin/Sales/ Marketing)	NOT NULL
phone	VARCHAR	15	Contact number	Optional

4.2 ER Diagram (Entity-Relationship Diagram

The ER diagram visually shows these entities as rectangles, attributes as ovals, and relationships as diamonds, with proper cardinality, making the system structure clear for development and implementation.



Relationships

User → Customer

One user can add multiple customers

Relationship: **1 to Many**

Customer → Lead

One customer can have multiple leads/opportunities

Relationship: **1 to Many**

Customer → Interaction

One customer can have multiple interaction records

Relationship: **1 to Many**

User → Lead

One sales user can manage multiple leads

Relationship: **1 to Many**

User → Task

One user can have multiple assigned tasks

Relationship: **1 to Many**

An **ER Diagram** represents the database structure and relationships between entities in the CRM system.

Entities in CRM System- User, **Customer**,**Lead**, **Interaction**, **Task/Reminder** and **Campaign**.

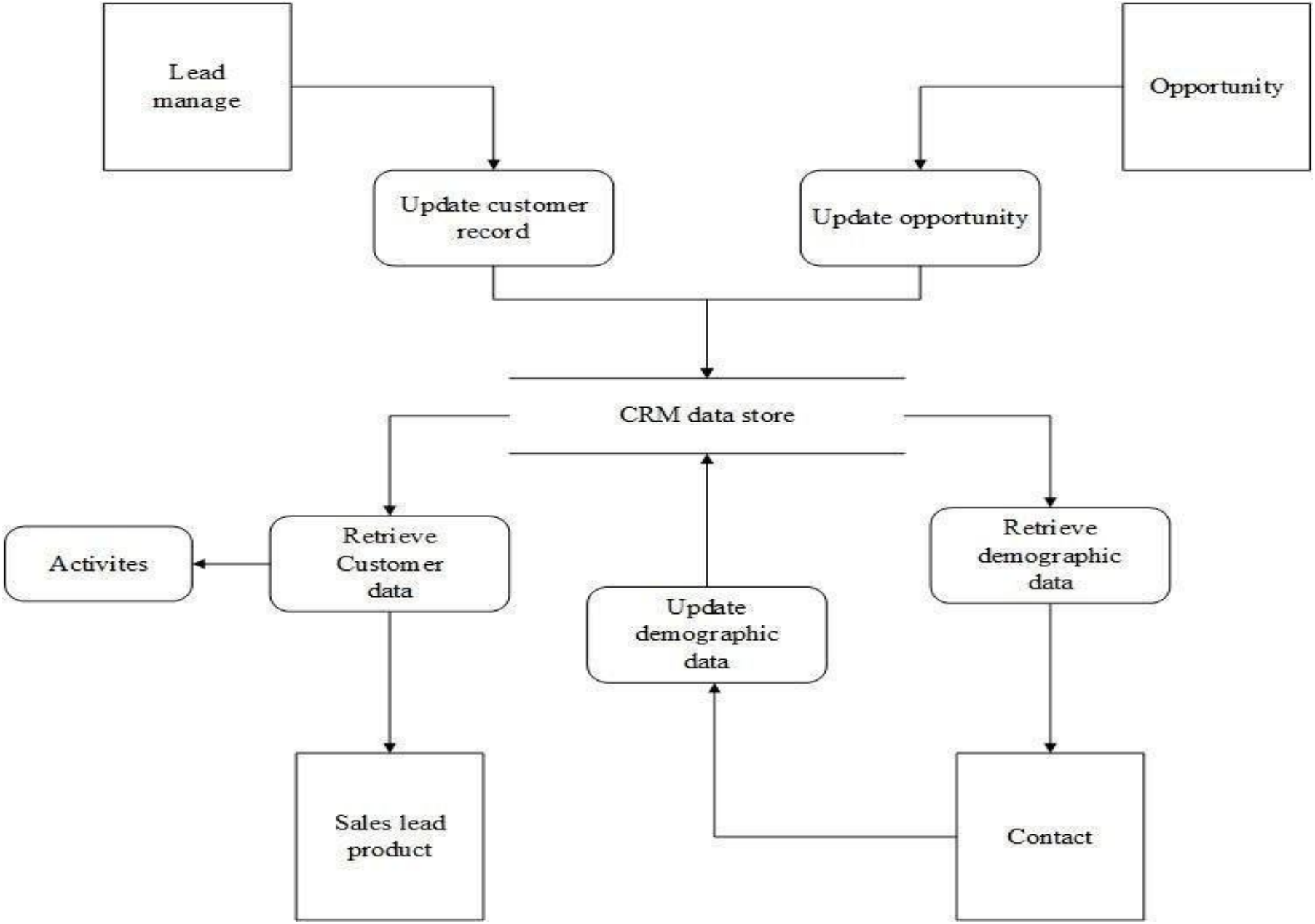
4.3 DFD (Data Flow Diagram)

A **Data Flow Diagram (DFD)** shows how data moves through the system. It includes processes, data stores, input/output flows, and entities.

The **Data Flow Diagram (DFD)** represents how data is entered, processed, stored, and transferred within the **Customer Relationship Management (CRM) system** to generate the required outputs. It provides a clear picture of the system's workflow and ensures proper understanding of how information moves through different modules.

At **DFD Level 0 (Context Diagram)**, the CRM system is shown as a single process interacting with external entities such as **Users (Admin, Sales Executive, Marketing Executive)** and supporting external services like **email systems or reporting tools**. The system receives inputs such as customer details, lead information, campaign updates, and task records, and produces outputs including **reports, dashboards, and notifications**.

At **DFD Level 1**, the CRM system is further divided into major functional processes such as **Customer Management, Lead Tracking, Sales Management, Campaign Management, and Reporting & Analytics**. Data flows between these processes and relevant data stores including **Customer Data, Lead Data, Sales Data, Campaign Data, and User Data**, ensuring structured storage and quick retrieval.


Lead manage
Opportunity
Update customer record
Update opportunity
CRM data store
Activites
Retrieve Customer data
Retrieve demographic data
Update demographic data
Sales lead product
Contact

5. Implementation

The **implementation phase** of the Customer Relationship Management (CRM) System focuses on converting the designed architecture into a fully working software application. In this stage, all planned modules are developed by writing source code, integrating the database, and applying the required business logic. Each functionality is implemented according to the system requirements to ensure smooth operation and accurate results.

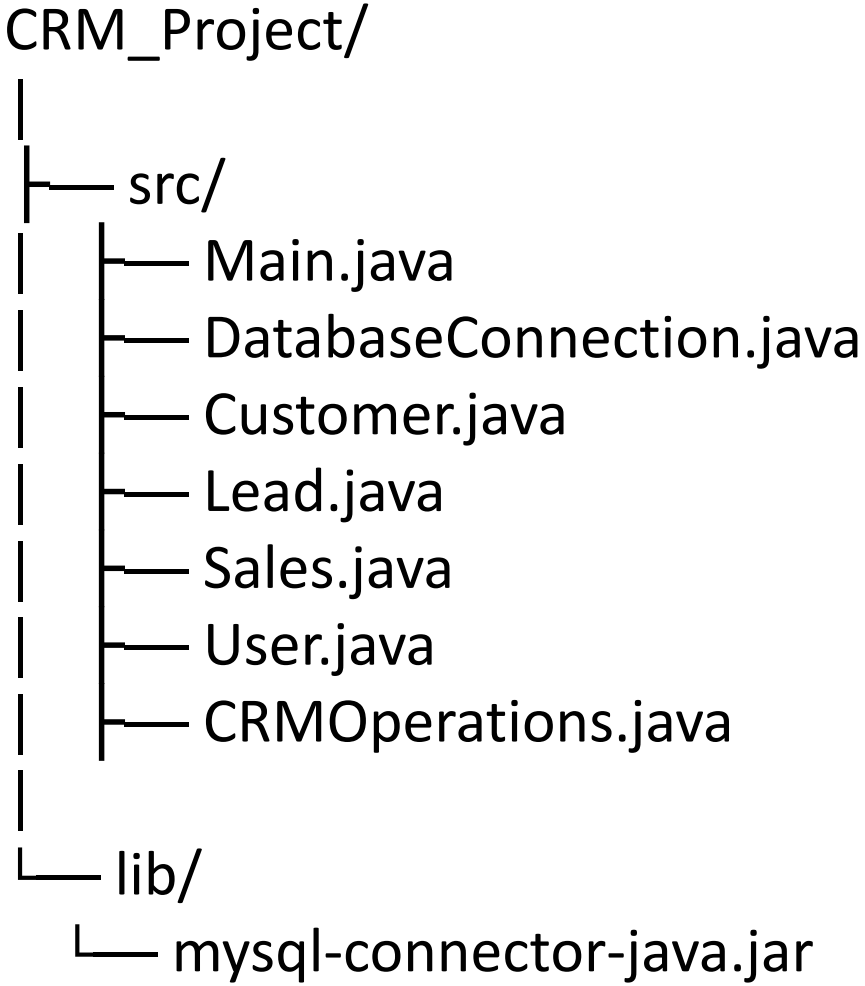
The CRM system is developed using **Java** as the backend programming language, ensuring robust performance and secure processing. **JDBC (Java Database Connectivity)** is used for connecting the application with the **MySQL database**, enabling real-time data storage, retrieval, and updates. The user interface is created using **HTML, CSS, and JavaScript**, providing a clean and interactive frontend for users.

Additionally, testing is performed during development to verify each module's functionality, reduce errors, and ensure that the system works correctly under different conditions.

The CRM system uses:

Java for backend logic and core processing
JDBC for database communication
MySQL for data storage
HTML/CSS/JavaScript for frontend interface

Project Structure



Description of Key Files

Main.java – Entry point of the CRM application and menu/dashboard control.
DatabaseConnection.java – Handles database connection using JDBC.
Customer.java – Contains customer attributes and customer-related methods.
Lead.java – Defines lead structure, lead status management, and follow-up features.
Sales.java – Handles sales activities, opportunity management, and conversions.
User.java – Stores user details and manages role-based authentication.
CRMOperations.java – Contains major CRUD operations, business logic, and system functions.
mysql-connector-java.jar – External MySQL JDBC driver required for database connectivity.

1. Database Setup (MySQL)

```
CREATE DATABASE crm_system;
USE crm_system;
CREATE TABLE users (
userID INT PRIMARY KEY AUTO_INCREMENT,
username VARCHAR(50) NOT NULL,
password VARCHAR(50) NOT NULL,
role VARCHAR(20) NOT NULL
);
```

```
CREATE TABLE customers (
customerID INT PRIMARY KEY AUTO_INCREMENT,
firstName VARCHAR(50),
lastName VARCHAR(50),
email VARCHAR(100),
phone VARCHAR(15)
);
CREATE TABLE leads (
leadID INT PRIMARY KEY AUTO_INCREMENT,
customerID INT,
status VARCHAR(20),
assignedTo INT,
FOREIGN KEY (customerID) REFERENCES customers(customerID),
FOREIGN KEY (assignedTo) REFERENCES users(userID)
);
CREATE TABLE sales (
salesID INT PRIMARY KEY AUTO_INCREMENT,
customerID INT,
leadID INT,
amount DOUBLE,
salesDate DATE,
FOREIGN KEY (customerID) REFERENCES customers(customerID),
FOREIGN KEY (leadID) REFERENCES leads(leadID)
);
```

2. **Database Connection** (**DatabaseConnection.java**)

```
import java.sql.*;
 public class DatabaseConnection
{
 private static final String URL = "jdbc:mysql://localhost:3306/crm_system";
private static final String USER = "root"; // your DB username
private static final String PASSWORD = ""; // your DB password
public static Connection getConnection()
{
 try
 {
 Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);
 return conn;
 }
 catch (SQLException e)
 {
System.out.println("Connection Failed: " + e.getMessage());
return null;
 }
 }
 }
```

3) Customer Module (Customer.java)

```
public class Customer {
    private int customerID;
    private String firstName;
    private String lastName;
    private String email;
    private String phone;

    // Constructor with ID (for fetching from DB)
    public Customer(int customerID, String firstName, String lastName, String email,
String phone) {
        this.customerID = customerID;
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
        this.phone = phone;
    }

    // Constructor without ID (for inserting new customer)
    public Customer(String firstName, String lastName, String email, String phone) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
```

```
 this.phone = phone;
    }

    public int getCustomerID()
{
 return customerID;
}
    public String getFirstName()
{
 return firstName;
}
    public String getLastName()
{
 return lastName;
 }
    public String getEmail()
{
 return email;
 }
    public String getPhone()
{
 return phone;
 }
```

```
@Override
   public String toString()
 {
       return customerID + " | " + firstName + " |
" + lastName + " | " + email + " | " + phone;
   }
}
```

4) User Module (User.java)

```
public class User {
   private int userID;
   private String username;
   private String password;
   private String role;

   public User(int userID, String username, String password, String role) {
      this.userID = userID;
      this.username = username;
      this.password = password;
      this.role = role;
   }
```

```
 public int getUserID()
{
return userID;
 }
    public String getUsername()
 {
return username;
 }
    public String getPassword() {
return password;
}
    public String getRole()
{
return role;
 }
}
```

5) Lead Module (Lead.java)

```
public class Lead {
    private int leadID;
    private int customerID;
    private int assignedTo;
    private String status;

    public Lead(int leadID, int customerID, String
status, int assignedTo) {
        this.leadID = leadID;
        this.customerID = customerID;
        this.status = status;
        this.assignedTo = assignedTo;
    }

    public int getLeadID() { return leadID; }
    public int getCustomerID() { return
customerID; }
    public int getAssignedTo() { return
assignedTo; }
    public String getStatus() { return status; }
```

```
 @Override
    public String toString() {
        return leadID + " | " + customerID + " | " +
status + " | " + assignedTo;
    }
}
```

6) Sales Module (Sales.java) –

```
import java.sql.Date;

public class Sales {
    private int salesID;
    private int customerID;
    private int leadID;
    private double amount;
    private Date salesDate;

    public Sales(int salesID, int customerID, int
leadID, double amount, Date salesDate) {
        this.salesID = salesID;
        this.customerID = customerID;
        this.leadID = leadID;
```

```
 this.amount = amount;
        this.salesDate = salesDate;
    }

    public int getSalesID()
{
return salesID;
}
    public int getCustomerID() {
return customerID; }
    public int getLeadID() {
return leadID; }
    public double getAmount() { return amount;
}
    public Date getSalesDate() { return
salesDate; }

    @Override
    public String toString() {
        return salesID + " | " + customerID + " | "
+ leadID + " | " + amount + " | " + salesDate;
    }
}
```

7) CRM Operations (CRMOperations.java) –

```
import java.sql.*;
import java.util.Scanner;

public class CRMOperations {
    private Scanner sc = new
Scanner(System.in);

    // ?   Add Customer
    public void addCustomer() {
        System.out.print("First Name: ");
        String fname = sc.nextLine().trim();

        System.out.print("Last Name: ");
        String lname = sc.nextLine().trim();

        System.out.print("Email: ");
        String email = sc.nextLine().trim();

        System.out.print("Phone: ");
        String phone = sc.nextLine().trim();
```

```
 // Basic Validation
       if (fname.isEmpty() || lname.isEmpty() ||
phone.isEmpty()) {
           System.out.println("Error: First Name,
Last Name and Phone cannot be empty!");
           return;
       }

       String query = "INSERT INTO
customers(firstName,lastName,email,phone)
VALUES (?,?,?,?)";

       try (Connection conn =
DatabaseConnection.getConnection();
            PreparedStatement ps =
conn.prepareStatement(query)) {

           ps.setString(1, fname);
           ps.setString(2, lname);
           ps.setString(3, email);
           ps.setString(4, phone);

           int rows = ps.executeUpdate();
```

```
 if (rows > 0) {
            System.out.println("✅ Customer
Added Successfully!");
        } else {
            System.out.println("❌ Customer Not
Added!");
        }

    } catch (SQLException e) {
        System.out.println("Database Error: " +
e.getMessage());
    }
  }

  // 🔍 View Customers
  public void viewCustomers() {
    String query = "SELECT * FROM
customers";

    try (Connection conn =
DatabaseConnection.getConnection();
         Statement stmt =
conn.createStatement();
```

```
ResultSet rs = stmt.executeQuery(query)) {

        System.out.println("\nCustomerID | FirstName | LastName | Email | Phone");
        System.out.println("-----------------------------------------------");

        boolean found = false;
        while (rs.next()) {
           found = true;
           System.out.println(
                 rs.getInt("customerID") + " | " +
                 rs.getString("firstName") + " | " +
                 rs.getString("lastName") + " | " +
                 rs.getString("email") + " | " +
                 rs.getString("phone")
           );
        }
   if (!found) {
           System.out.println("No customers found.");
        }
} catch (SQLException e) {
        System.out.println("Database Error: " + e.getMessage());
      }
   }
}
```

8) Main Class (Main.java) –

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        CRMOperations crm = new CRMOperations();
        Scanner sc = new Scanner(System.in);
      int choice;
        do {
            System.out.println("\n========== CRM SYSTEM ==========");
            System.out.println("1. Add Customer");
            System.out.println("2. View Customers");
            System.out.println("0. Exit");
            System.out.print("Enter choice: ");

            while (!sc.hasNextInt()) {
                System.out.println("Invalid input! Enter number only.");
                sc.next();
            }
            choice = sc.nextInt();
            sc.nextLine();
```

```
 switch (choice) {
                case 1 -> crm.addCustomer();
                case 2 -> crm.viewCustomers();
                case 0 -> System.out.println("
Exiting CRM System...");
                default -> System.out.println("
Invalid Choice!");
            }

        } while (choice != 0);

        sc.close();
    }
}
```

5.2 Output Screens

The **output screens** of the Customer Relationship Management (CRM) System are designed to provide a simple and user-friendly interaction for managing **customers, leads, sales, campaigns, and reports**. Although the current implementation is **console-based**, the outputs are well-structured, easy to understand, and enable smooth navigation across different modules.

1. Main Menu / Dashboard

When the CRM application starts, the system displays a dashboard menu that allows users to access different modules by entering the required option number.

--- CRM SYSTEM ---
1. Add Customer
2. View Customers
3. Add Lead
4. View Leads
5. Add Sales
6. View Sales
7. Add Campaign
8. View Campaigns
9. Generate Reports
0. Exit
Enter choice:

2. Customer Management Screens

a) Add Customer

This screen allows the user to enter customer details and store them in the database.

Enter Customer First Name: Deepak
Enter Customer Last Name: Jaiswal
Enter Customer Email: deepak@gmail.com
Enter Customer Phone: 9876543210

Customer Added Successfully!

b) View Customers

This screen displays all available customer records stored in the database in a tabular format.

CustomerID	FirstName	LastName	Email	Phone
1	Deepak	Jaiswal	deepak@gmail.com	9876543210
2	Rahul	Mehta	rahul@gmail.com	9123456789
3	Devansh	Dixit	devansh@gmail.com	9988776655

Highlights:

Customer information is displayed clearly.
Helps in quick identification and retrieval of customer details.

3. Lead Management Screens

a) Add Lead

This module allows the user to assign leads to sales executives and update the lead status.

Enter Customer ID: 1
Enter Lead Status (New/Contacted/Converted): New
Assign to UserID: 2
Lead Added Successfully!

b) View Leads

Displays all leads along with their customer linkage, current status, and assigned user.

LeadID	CustomerID	Status	AssignedTo
1	1	New	SalesExec1
2	2	Contacted	SalesExec2

Highlights:

Tracks the progress of leads effectively.
Helps sales team manage follow-ups systematically.

4. Sales Management Screens

a) Add Sales

Sales transactions can be recorded once a lead is converted into a successful deal.

Enter Customer ID: 1
Enter Lead ID: 1
Enter Sale Amount: 5000
Sale Recorded Successfully!

b) View Sales

Displays all sales records along with customer ID, lead ID, amount, and date.

SalesID	CustomerID	LeadID	Amount	SalesDate
1	1	1	5000.0	2026-01-08
2	2	2	7500.0	2026-01-08

Highlights:

Maintains sales history for analysis.
Links sales records with corresponding customers and leads.

5. Campaign Management Screens

a) Add Campaign

Allows marketing team to create marketing campaigns and set campaign duration.

Enter Campaign Name: New Year Promo
Enter Start Date (YYYY-MM-DD): 2026-01-01
Enter End Date (YYYY-MM-DD): 2026-01-31
Campaign Added Successfully!

b) View Campaigns

Displays the list of campaigns along with key details.

CampaignID	Name	StartDate	EndDate	TargetAudience
1	New Year Promo	2026-01-01	2026-01-31	All Customers

Highlights:

Helps monitor marketing activities.
Supports better customer engagement tracking.

6. Reports Screen

The report module generates summarized information helpful for management decision-making.

--- SALES REPORT ---

CustomerID | CustomerName | Total Leads | Total Sales

1 | Deepak Jaiswal | 2 | 5000.0

2 | Rahul Mehta | 1 | 7500.0

Highlights:

Provides clear performance summary.

Supports data-driven decision-making.

Can be further enhanced into graphical dashboards in future web/GUI versions.

6. **Testing**

The testing phase ensures that the CRM system works correctly, meets all functional requirements, and is free from errors. Both unit testing (testing individual modules) and integration testing (testing combined modules) are performed. The system is tested using sample test data to verify that customer management, lead tracking, sales processing, campaign management, and reporting work as expected.

6.1 Test Data The table below shows sample test data used to validate the system:

7. User Manual

The **User Manual** provides step-by-step guidance for users to operate the **Customer Relationship Management (CRM) System** efficiently. It explains the system installation, setup, running procedure, module navigation, and screen outputs. Since the current CRM system is **console-based**, all interactions are performed using menu options and input prompts.

7.1 How to Use Project (Guidelines)

1) Installation & Setup

a. To run the CRM system successfully, the following setup is required:
b. Install **Java JDK** on your system.
c. Install **MySQL Server** and MySQL Workbench (optional but recommended).
d. Create a database named:

Create required tables such as:
•users
•customers
•leads
•sales
•campaigns

5. Add the JDBC driver:
   Include **mysql-connector-java.jar** in the project classpath.
6. Open the project in an IDE such as:
   Eclipse / IntelliJ IDEA / NetBeans.

2) Running the System

Run the file:
Main.java

The system displays the **Main Menu / Dashboard**.
Users can select the required module using menu options.

3) Login (Optional / Advanced Version)

(If login module is implemented in the enhanced version)
User enters valid **username and password**.
System provides role-based access:
   Admin: access to all modules
   Sales Executive: customer, lead, and sales modules
   Marketing Executive: customer and campaign modules

4) Navigating Modules

Select the module by entering its corresponding number.

Follow on-screen instructions to:

- Add records
- View records
- Update data (if implemented)
- Generate reports

5) Exiting the System

Choose option:

0. Exit

The system safely terminates the program.

7.2 Screen Layouts and Description

This section describes the available screens and how to use each one.

1. Main Menu / Dashboard Screen

Screen Output:

--- CRM SYSTEM ---
1. Add Customer
2. View Customers
3. Add Lead
4. View Leads
5. Add Sales
6. View Sales
7. Add Campaign
8. View Campaigns
9. Generate Reports
0. Exit
Enter choice:

Description:
This dashboard acts as the navigation screen for the entire CRM system. Users can select a module by entering the required option number.

2. Customer Management Screens

a) Add Customer Screen
Screen Output:

Enter Customer First Name: Deepak
Enter Customer Last Name: Jaiswal
Enter Customer Email: deepak@gmail.com
Enter Customer Phone: 9876543210
Customer Added Successfully!

Description:
Allows the user to add a new customer into the CRM database with basic details such as name, email, and phone number.

b) View Customers Screen
Screen Output:

```
CustomerID | FirstName | LastName | Email            | Phone
1          | Deepak    | Jaiswal  | deepak@gmail.com | 9876543210
```

Description:
Displays all customer records stored in the database in a structured tabular format for easy viewing and verification.

3. Lead Management Screens

a) Add Lead Screen
Screen Output:

Enter Customer ID: 1
Enter Lead Status (New/Contacted/Converted): New
Assign to UserID: 2
Lead Added Successfully!

Description:
Captures lead details for a specific customer and assigns the lead to a sales executive for follow-up.

b) View Leads Screen
Screen Output:

LeadID | CustomerID | Status | AssignedTo
1 | 1 | New | SalesExec1

Description:
Displays all lead records, their current status, and the assigned sales executive.

4. Sales Management Screens

a) Add Sale Screen
Screen Output:

Enter Customer ID: 1
Enter Lead ID: 1
Enter Sale Amount: 5000
Sale Recorded Successfully!

Description:
Records a customer sale and links the sale with the corresponding lead and customer record.

b) View Sales Screen

Screen Output:

SalesID	CustomerID	LeadID	Amount	SalesDate
1	1	1	5000.0	2026-01-08

Description:
Displays all completed sales transactions for business analysis and revenue tracking.

5. Campaign Management Screens

a) Add Campaign Screen

Screen Output:

Enter Campaign Name: New Year Promo
Enter Start Date (YYYY-MM-DD): 2026-01-01
Enter End Date (YYYY-MM-DD): 2026-01-31
Campaign Added Successfully!

Description:
Stores marketing campaign details including campaign name and duration. Campaign details help marketing teams track promotional activities.

8. Project Applications and Limitations

8.1 Applications of the CRM System

The **Customer Relationship Management (CRM) System** is developed to help organizations efficiently manage customer relationships, track business activities, and improve overall productivity. It provides a centralized platform for handling customer information, sales, leads, and marketing campaigns.
The major applications of the CRM system are:

Customer Management

Stores all customer details in a centralized database.
Helps maintain complete customer history and interaction records for better service delivery.

Lead Tracking

Manages potential customers (leads) and tracks their progress through different stages.

Enables assignment of leads to sales executives to ensure timely follow-ups and higher conversion.

Sales Management

Records sales transactions and links them with customers and lead details.

Helps monitor sales performance and improve revenue tracking.

Campaign Management

Supports planning and managing marketing campaigns.

Tracks campaign duration, target audience, and customer engagement.

Reporting and Analytics

Generates reports related to leads, sales, and campaigns.

Helps management make **data-driven decisions** and identify business growth opportunities.

User Management

Provides role-based access control for Admin, Sales Executive, and Marketing Executive.

Ensures secure access to the system and prevents unauthorized actions.

Integration and Expansion

Can be enhanced into a web-based or GUI-based system.

Supports future integration with tools like email systems, mobile applications, and analytics dashboards.

8.2 Limitations of the CRM System

Although the CRM system provides essential features for customer and sales management, some limitations exist in the current version:

Console-based Interface

The system is text-based and lacks the visual interactivity of web-based or GUI applications.

Limited Scalability

Suitable for small to medium businesses; large organizations may require advanced enterprise-level CRM features.

No Real-time Notifications

The current system does not provide real-time reminders or alerts for lead follow-ups, tasks, or sales updates.

No Mobile Access

The system is not accessible through mobile devices in the current implementation.

Manual Data Entry

Data must be entered manually; there is no automatic data import or synchronization from external systems.

Basic Reporting

Reports are displayed in text format only; advanced dashboards, charts, and graphical analytics are not available.

9. Conclusion and Future Enhancements

9.1 Conclusion

The **Customer Relationship Management (CRM) System** is developed to efficiently manage customer interactions, lead handling, sales tracking, and marketing campaigns in a centralized and structured manner. The system provides quick access to customer data, supports lead monitoring, records sales transactions, and generates reports that assist businesses in making informed decisions.
This CRM project improves workflow efficiency, supports better customer service, and enhances productivity. Even though the current version is console-based, it successfully demonstrates the core CRM functionalities, modular programming structure, and database integration. It also serves as a strong foundation for future upgrades and real-world implementation.

9.2 Future Enhancements

Although the current CRM system meets basic requirements, it can be further enhanced to make it more advanced, scalable, and user-friendly:

GUI / Web-based Interface

Upgrade from console-based interface to GUI using **Java Swing/JavaFX**, or web technologies like **HTML, CSS, JavaScript**, and frameworks.

Mobile Application Development

Develop an Android/iOS application so sales and marketing executives can update and access data anytime from mobile devices.

Real-time Notifications & Alerts

Add notifications for follow-ups, pending tasks, sales updates, and campaign reminders using email/SMS/push notifications.

Advanced Analytics and Dashboards

Include interactive charts and dashboards to visualize lead status, sales performance, campaign success rate, etc.

Data Import / Export Feature

Allow importing customer details from CSV/Excel files and exporting reports for presentation and documentation.

AI Integration

Integrate AI-based features such as:

- lead scoring
- predictive sales analysis
- customer behavior tracking
- recommendation system

Security and Role Enhancements

Implement stronger security measures like encryption, activity logs, audit reports, and advanced permissions for enterprise use.

10. Bibliography & References

The following books, websites, and online resources were referred to during the design, development, and documentation of the **Customer Relationship Management (CRM) System** project:

Books

Pressman, Roger S. *Software Engineering: A Practitioner's Approach*, 9th Edition, McGraw-Hill Education, 2020.

Sommerville, Ian. *Software Engineering*, 10th Edition, Pearson Education, 2016.

Awad, Elias M. *Management Information Systems: Managing the Digital Firm*, 6th Edition, Pearson, 2012.

Laudon, Kenneth C., and Laudon, Jane P. *Management Information Systems: Managing the Digital Firm*, 15th Edition, Pearson, 2021.

Hoffer, Joseph A., Ramesh, Venkataraman, and Topi, Heikki. *Modern Database Management*, 13th Edition, Pearson, 2020.

Notes

All diagrams, tables, database structures, and program code examples included in this project report were prepared and implemented by the **project team**.

The references listed above were used to understand and apply concepts related to:

- Software Engineering and Agile methodology
- CRM system design principles
- Database management concepts
- JDBC-based database connectivity and implementation