PROJECT REPORT

ON

CUSTOMER RELATIONSHIP MANAGEMENT USING AGILE METHODOLOGY

Submitted By:

Nikhil Harwani

1. ABSTRACT

In the contemporary business environment, maintaining robust relationships with customers is paramount for organizational success. This project report details the development of a Customer Relationship Management (CRM) system utilizing the Agile software development methodology. The primary objective of this system is to streamline customer interactions, sales processes, and data management into a centralized platform.

Unlike traditional Waterfall models, the adoption of Agile methodology allowed for iterative development, continuous feedback integration, and rapid adaptation to changing requirements throughout the project lifecycle. The system features modules for Lead Management, Contact Storage, Sales Pipeline Visualization, and Activity Tracking.

The resulting application provides a scalable, user-friendly interface that enhances the efficiency of sales and support teams. This report documents the entire software development lifecycle, from problem identification and requirement gathering to system design, implementation, and testing, highlighting the benefits of Agile practices in delivering a high-quality software product.

2. INTRODUCTION

2.1 Introduction

Customer Relationship Management (CRM) refers to the practices, strategies, and technologies that companies use to manage and analyze customer interactions and data throughout the customer lifecycle. The goal is to improve customer service relationships, assist in customer retention, and drive sales growth. This project focuses on building a web-based CRM tailored for small to medium enterprises (SMEs) using modern web technologies and the Agile Scrum framework.

2.2 Problem Identification

Many organizations still rely on decentralized methods for managing customer data, such as spreadsheets, physical files, or disconnected email threads. This leads to:

- **Data Redundancy:** Multiple copies of inconsistent data.
- **Communication Gaps:** Sales teams lack visibility into previous interactions.
- **Inefficiency:** Manual tracking of leads results in lost sales opportunities.
- **Lack of Analytics:** Inability to generate reports on sales performance.

2.3 Need of the Project

To address the identified problems, a centralized CRM system is required to:

1. Unify customer data into a single "source of truth."
2. Automate routine tasks such as follow-up emails.
3. Provide visual pipelines to track the status of potential deals.
4. Allow for the flexible addition of features via Agile sprints.

2.4 Project Scheduling (Agile Timeline)

The project was divided into 4 Sprints, each lasting two weeks:

Phase	Duration	Deliverable
Sprint 1	2 Weeks	Requirement Analysis, UI Prototyping, Database Design
Sprint 2	2 Weeks	User Authentication, Contact Management Module
Sprint 3	2 Weeks	Lead Tracking, Sales Pipeline, Activity Logs
Sprint 4	2 Weeks	Reporting, Testing, Deployment, Documentation

2.5 Objectives

- To design and develop a responsive web-based CRM application.
- To implement Agile Scrum practices for project management.
- To create a secure database architecture for sensitive customer data.
- To provide analytical tools for visualizing sales performance.

3. SOFTWARE REQUIREMENT SPECIFICATION

3.1 Purpose

The purpose of this SRS document is to outline the functional and non-functional requirements of the CRM system. It serves as a guide for developers and testers to ensure the final product meets the stakeholder's needs.

3.2 Scope

The system covers the following modules: **Admin Panel** (User management), **Sales Dashboard** (Lead tracking), and **Customer Database** (CRUD operations). It does not currently cover AI-based predictive analytics or third-party payment gateway integration.

3.3 System Requirements

Hardware Requirements (Minimum)

- **Processor:** Intel Core i3 or equivalent
- **RAM:** 4 GB
- **Hard Disk:** 500 MB free space
- **Monitor:** Standard 1366x768 resolution

Software Requirements

- **Operating System:** Windows 10/11, Linux, or macOS
- **Frontend:** HTML5, CSS3, JavaScript (React.js)
- **Backend:** Node.js / Express.js (or Java Spring Boot)
- **Database:** MySQL or MongoDB
- **Browser:** Chrome, Firefox, or Edge

3.4 Tools Used

- **VS Code:** Integrated Development Environment.
- **GitHub:** Version Control.

- **Jira/Trello:** Agile Project Management (Sprint tracking).
- **Postman:** API Testing.

3.5 Software Process Model

Agile Model: The project utilizes the Agile model. This model was chosen because it permits adaptive planning, evolutionary development, early delivery, and continuous improvement, and it encourages rapid and flexible response to change.

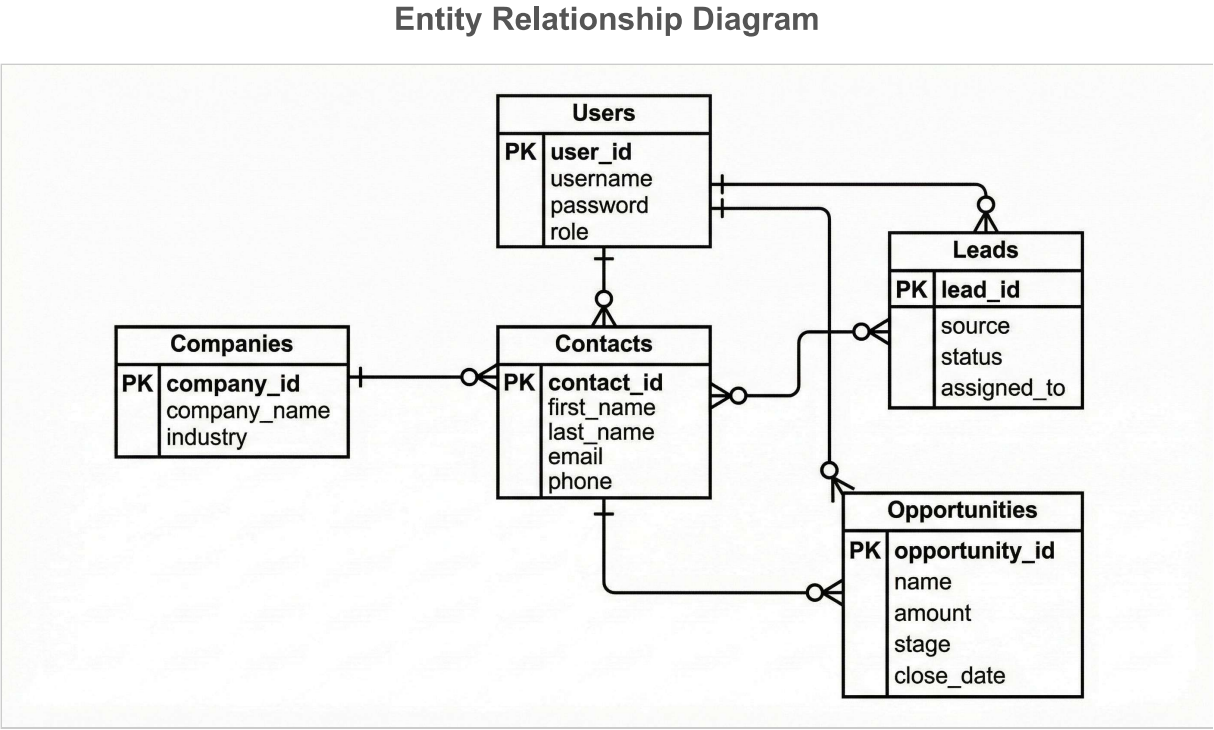
4. SYSTEM DESIGN

4.1 Data Dictionary

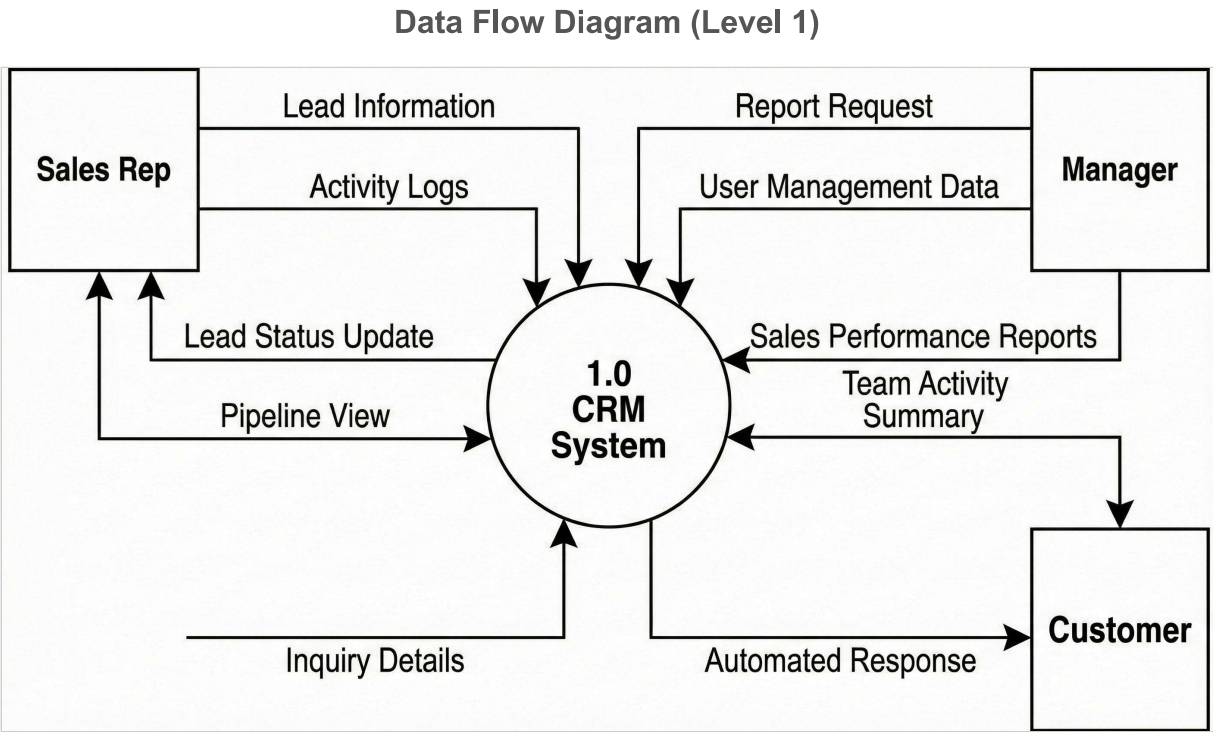
Table: Users

Field	Type	Description
user_id	INT (PK)	Unique Identifier
username	VARCHAR	Login Name
password	VARCHAR	Encrypted Password
role	ENUM	Admin/Sales/Viewer

4.2 ER Diagram

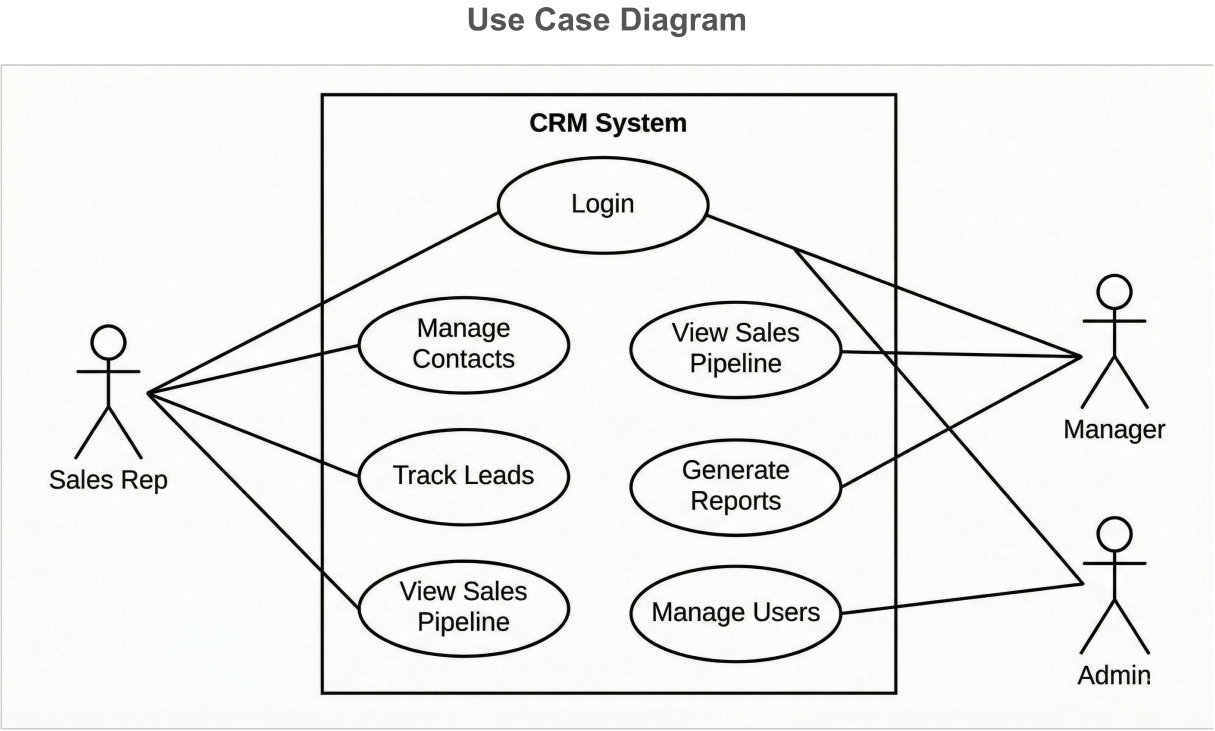


4.3 Data Flow Diagram (DFD)

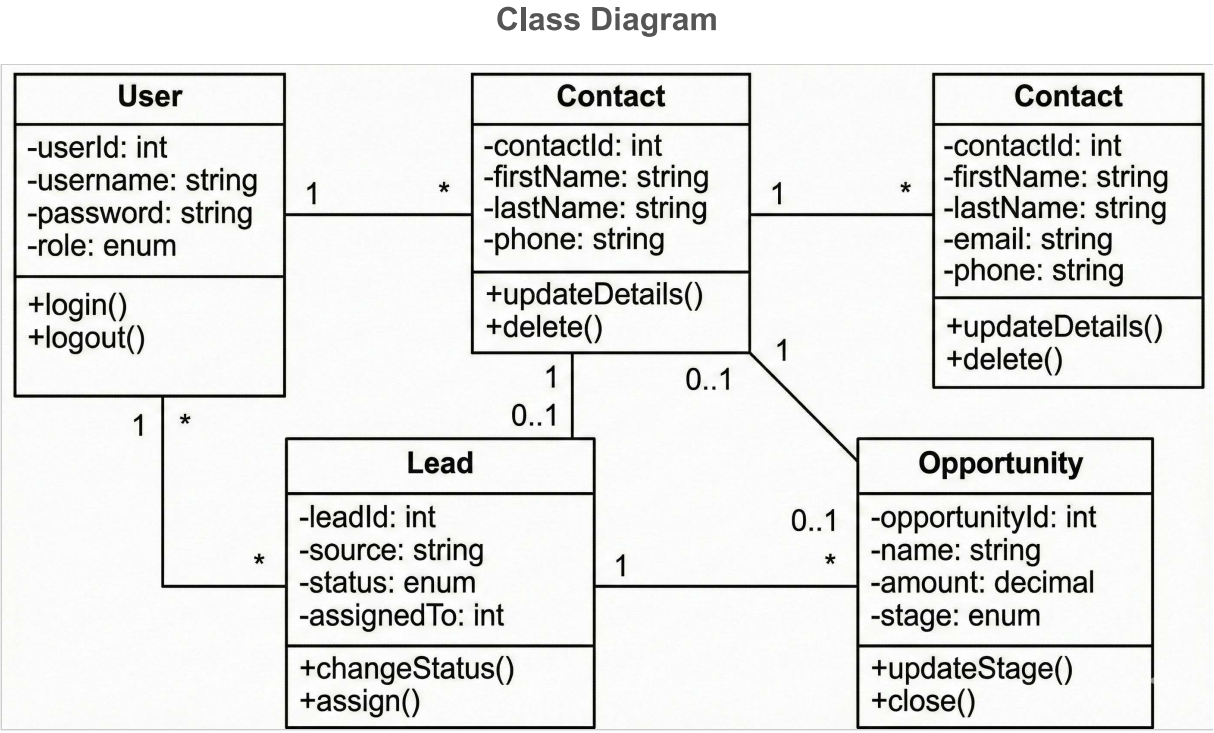


4.4 UML Diagrams

Use Case Diagram



Class Diagram



5. IMPLEMENTATION

5.1 Program Code

Below are selected core snippets of the application logic.

Backend Controller (Lead Management)

```
// LeadController.js
const Lead = require('../models/Lead');

exports.createLead = async (req, res) => {
    try {
        const newLead = new Lead({
            name: req.body.name,
            email: req.body.email,
            status: 'New',
            assignedTo: req.user.id
        });

        const savedLead = await newLead.save();
        res.status(201).json(savedLead);
    } catch (err) {
        res.status(500).json({ error: err.message });
    }
};
```

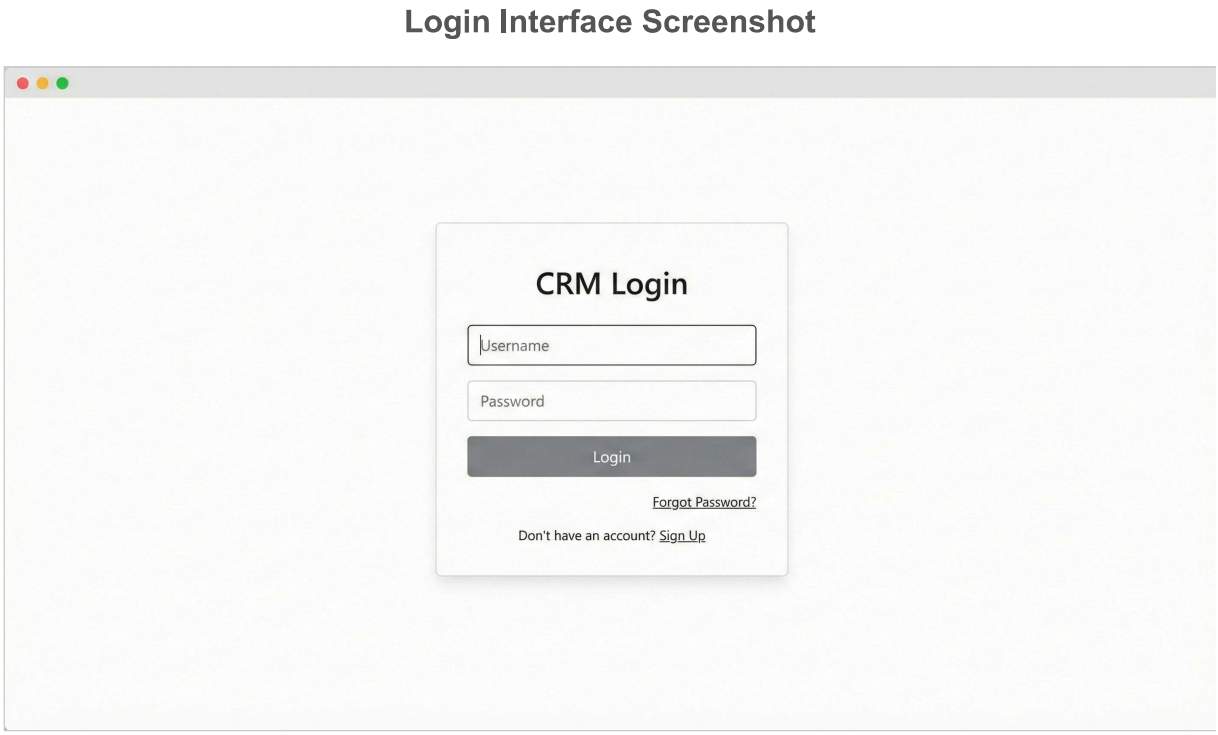
Database Connection

```
// db.js
const mongoose = require('mongoose');

const connectDB = async () => {
    try {
        await mongoose.connect(process.env.MONGO_URI, {
            useNewUrlParser: true,
            useUnifiedTopology: true,
        });
        console.log('MongoDB Connected...');
    } catch (err) {
        console.error(err.message);
        process.exit(1);
    }
};
module.exports = connectDB;
```

5.2 Output Screens

Login Screen:



Dashboard Overview:

Dashboard Screenshot

6. TESTING

6.1 Test Data & Strategy

Testing was conducted iteratively at the end of every sprint. The strategy included Unit Testing (using Jest) and Manual Integration Testing.

6.2 Test Results

Test Case ID	Description	Input Data	Expected Output	Status
TC_01	Login Validation	User: admin / Pass: 1234	Dashboard Access	PASS
TC_02	Invalid Login	User: admin / Pass: wrong	Error Message	PASS
TC_03	Create Lead	Name: John Doe	Lead Saved to DB	PASS
TC_04	Delete Lead	ID: 101	Record Removed	PASS

7. USER MANUAL

7.1 How to use project guidelines

1. **Installation:** Ensure Node.js and MongoDB are installed on the local machine.
2. **Setup:** Run `npm install` to install dependencies.
3. **Execution:** Run `npm start` to launch the localhost server.
4. **Access:** Open browser and navigate to `http://localhost:3000`.

7.2 Screen Layouts and Description

Navigation Bar: Located at the top, provides links to Leads, Contacts, and Settings.
Sidebar: Provides quick filters for viewing 'New', 'In Progress', and 'Closed' deals.
Action Buttons: 'Add New' buttons are consistently placed in the top-right corner of data tables.

8. PROJECT APPLICATIONS AND LIMITATIONS

Applications

- **SMEs:** Ideal for small businesses tracking local clients.
- **Real Estate:** Tracking property buyers and viewing schedules.
- **Education Consultancies:** Managing student leads and follow-ups.

Limitations

- The system currently relies on an active internet connection; offline mode is not supported.
- Mobile responsiveness is limited in the current version.
- No integrated VoIP calling feature.

9. CONCLUSION AND FUTURE ENHANCEMENT

The "Customer Relationship Management using Agile" project successfully demonstrated the efficiency of iterative development. By breaking the complex requirements of a CRM into manageable sprints, the team was able to deliver a functional product that meets core business needs. The system resolves the issues of data fragmentation and communication gaps identified in the problem statement.

Future Enhancements:

- **AI Integration:** Utilizing Machine Learning to score leads based on conversion probability.
- **Mobile App:** Developing a React Native application for field sales agents.
- **Third-Party API:** Integration with Gmail and Outlook for automatic email syncing.

10. BIBLIOGRAPHY & REFERENCES

Books:

- Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education.
- Schwaber, K., & Beedle, M. (2002). *Agile Software Development with Scrum*. Prentice Hall.

Websites:

- Agile Manifesto (http://agilemanifesto.org)
- Documentation for React.js (https://reactjs.org/)
- MongoDB Manual (https://docs.mongodb.com/)