IBM Innovation Center for Education

PHEMESOFT Software That Matters
TRUSTED DELIVERY PARTNER for IBM ICE

# UPES

# Internship - Final Report

# on

# Cloud-based Customer Relationship Management

**BACHELOR OF TECHNOLOGY**

CLOUD COMPUTING & VIRTUALIZATION TECHNOLOGY(CCVT)

SUBMITTED BY:


| Specialization | Roll Number | Name |
|---|---|---|
| CCVT | R2142211066 | Rohan Bakshi |
| CCVT | R2142211034 | Vilish Kumar |
| CCVT | R2142211041 | Harsh Choudhary |
| CCVT | R2142211045 | Lavanya Raj |
| CCVT | R2142211073 | Sidharth Bansal |


GUIDED BY:

Dr. Lalit Kumar

# Table of Contents

## Executive Summary

This report outlines the development of a Customer Relationship Management (CRM) system, focusing on its background, requirements, design, implementation, testing, and evaluation. The project leverages advanced technologies to enhance business processes and improve customer interactions.

# 1. Background

## 1.1. Aim

The primary aim of the CRM project is to create a robust system that facilitates effective management of customer relationships. This involves streamlining interactions, automating repetitive tasks, and providing tools for in-depth analysis of customer data. The CRM system aims to:

- Improve customer satisfaction through personalized interactions and timely responses.
- Enhance sales processes by tracking customer interactions and managing sales pipelines.
- Optimize marketing efforts by analyzing customer behavior and engagement metrics.
- Provide a unified platform for managing customer data and communication channels.

## 1.2.Technologies

### Backend

The backend of the application is built to handle data management, business logic, and real-time synchronization. The technologies used are:

● PHP: PHP processes server-side logic, handles form submissions, interacts with the MySQL database for CRUD operations, and dynamically generates HTML content for the CRM system, ensuring seamless functionality and data management.

● MySQL: CRM system, PHP interacts with MySQL to perform essential operations such as storing and retrieving user data, tickets, and quotations. It processes SQL queries to create, read, update, and delete records, ensuring efficient data management and providing dynamic content to users.

### Frontend

The frontend of the application is designed to provide an engaging and responsive user experience. The technologies used are:

● HTML: Structures the web pages, providing the skeleton for content. It is used to create the essential

elements of the web pages, ensuring that the content is well-organized and accessible.
● CSS: Utilizes custom stylesheets to style the user interface, ensuring a visually appealing and cohesive design.

● JavaScript: Adds dynamic functionality and interactivity to the web pages through scripts. JavaScript frameworks and libraries. These scripts handle tasks such as form validation, data fetching, and DOM manipulation, providing a smooth user experience.

● BootStrap: Bootstrap is used in this CRM system to ensure a responsive and visually appealing user interface. It provides pre-designed components and a grid system, simplifying the design process and enhancing user experience across various devices.

## 1.3 Hardware Architecture

The hardware architecture is based on cloud computing to ensure scalability and high availability:

- **Cloud Servers:** Virtual servers will host the application, database, and backend services.
- **Load Balancers:** To distribute incoming traffic and enhance performance.
- **Secure Storage:** For reliable data storage with redundancy and backup solutions to safeguard against data loss.

## 1.4 Software Architecture

The software architecture consists of:

- **Frontend:** Developed using HTML, JavaScript, and CSS to provide a responsive and user-friendly interface.
- **Backend:** Built with PHP to handle business logic, data processing, and interactions with the database.
- **Database:** A SQL or NoSQL database stores customer data, interactions, and other relevant information. This architecture supports efficient data retrieval and management.

# 2. System

### 2.1 Requirements

#### 2.1.1 Functional Requirements

- **Customer Data Management:** Storing and managing customer profiles, contact details, interaction history, and preferences.
- **Sales Tracking:** Monitoring sales activities, managing sales pipelines, and tracking leads and opportunities.
- **Marketing Automation:** Automating email campaigns, managing social media interactions, and tracking marketing metrics.

- **Reporting and Analytics:** Generating reports and dashboards to analyze customer data and sales performance.

### 2.1.2 User Requirements

These describe the needs of end-users, including:

- **User-Friendly Interface:** An intuitive and easy-to-navigate interface for various user roles (e.g., sales reps, managers).
- **Role-Based Access Control:** Ensuring that users can only access data and features relevant to their role.
- **Customizable Dashboards:** Allowing users to personalize their view and access key metrics easily.
- **Device Compatibility:** Ensuring the system works seamlessly on desktops, tablets, and mobile devices.

### 2.1.3 Environmental Requirements

Considerations for the operating environment include:

- **Regulatory Compliance:** Adhering to data protection laws like GDPR to ensure customer data privacy.
- **High Availability:** Implementing measures like failover systems and backups to ensure continuous service availability.
- **Cross-Platform Support:** Ensuring compatibility with different operating systems and web browsers.

## 2.2 Design and Architecture

- **Frontend Design:** Creating a visually appealing and responsive interface using HTML, JavaScript, and CSS.
- **Backend Design:** Developing PHP scripts to handle business logic, process user requests, and manage data interactions.
- **Database Design:** Structuring the database schema to efficiently store and retrieve data, with considerations for normalization and indexing.

## 2.3 Implementation

- **Coding:** Writing PHP scripts for backend functionality and HTML/CSS/JavaScript for frontend features.
- **Integration:** Connecting frontend components with backend services and integrating with external APIs if needed.
- **Deployment:** Deploying the application to the cloud servers and configuring the environment for production use.

## 2.4 Testing

### 2.4.1 Test Plan Objectives

- **Verifying Functionality:** Ensuring that all features work as intended.
- **Assessing Performance:** Evaluating system responsiveness and efficiency.
- **Ensuring Security:** Identifying and addressing security vulnerabilities.

### 2.4.2 Data Entry

- **Validate Input:** Ensure data is correctly validated and stored.
- **Prevent Errors:** Catch and handle incorrect or incomplete data entries

### 2.4.3 Security

Security testing involves:

- **Authentication:** Verifying that user login and access control mechanisms are secure.
- **Authorization:** Ensuring users have appropriate permissions.
- **Data Encryption:** Protecting data in transit and at rest.

### 2.4.4 Test Strategy.

The strategy includes:

- **Manual Testing:** Hands-on testing of features and user interactions.
- **Automated Testing:** Using scripts and tools to automate repetitive test cases.

### 2.4.5 System Test

Testing the integrated system to:

- **Ensure Components Work Together:** Verify that all parts of the system interact correctly.
- **Check Overall Functionality:** Confirm that the system meets all functional requirements.

### 2.4.6 Performance Test

Evaluating performance involves:

- **Load Testing:** Assessing system performance under expected user loads.
- **Stress Testing:** Testing system behavior under extreme conditions

### 2.4.7 Security Test

Conducting security tests to:

- **Identify Vulnerabilities:** Use tools and techniques to find potential security issues.
- **Test Security Measures:** Verify that security features are effective

### 2.4.8 Basic Test

Initial testing to:

- **Check Core Features:** Verify that essential functionalities are working.
- **Test User Interface:** Ensure the interface is user-friendly and responsive.

### 2.4.9 Stress and Volume Test

Testing involves:

- **Stress Testing:** Determining how the system performs under high loads.
- **Volume Testing:** Checking system performance with large volumes of data.

### 2.4.10 Recovery Test

Testing recovery processes to:

- **Ensure Data Integrity:** Verify that data is not lost and can be restored after failures.
- **Validate Backup Systems:** Ensure backups are reliable and can be used for recovery.

### 2.4.11 Documentation Test

Reviewing documentation to:

- **Ensure Accuracy:** Verify that technical and user documentation accurately reflects the system.
- **Check Completeness:** Confirm that all necessary information is included.

### 2.4.12 User Acceptance Test

Involves:

- **User Feedback:** Collecting feedback from end-users to ensure the system meets their needs.

- **Final Validation:** Confirming that the system is ready for production use.

### 2.4.13 System

Overall assessment of system performance, functionality, and reliability.

## 2.5 Graphical User Interface (GUI) Layout

Designing the GUI with:

- **Responsive Design:** Ensuring compatibility with various devices and screen sizes.
- **User Experience (UX):** Creating an intuitive layout with easy navigation.

### 2.6 Customer Testing

Involves:

- **Pilot Testing:** Conducting trials with selected customers to gather feedback.
- **Refinement:** Making adjustments based on customer input.

### 2.7 Evaluation

### 2.7.1 Table 1: Performance

Compiling performance metrics such as response times and system load handling to evaluate system efficiency.

### 2.7.2 Static Code Analysis

Reviewing the code for:

- **Code Quality:** Ensuring code is clean, well-organized, and maintainable.
- **Best Practices:** Verifying adherence to coding standards.

### 2.7.3 Wireshark

Using Wireshark to:

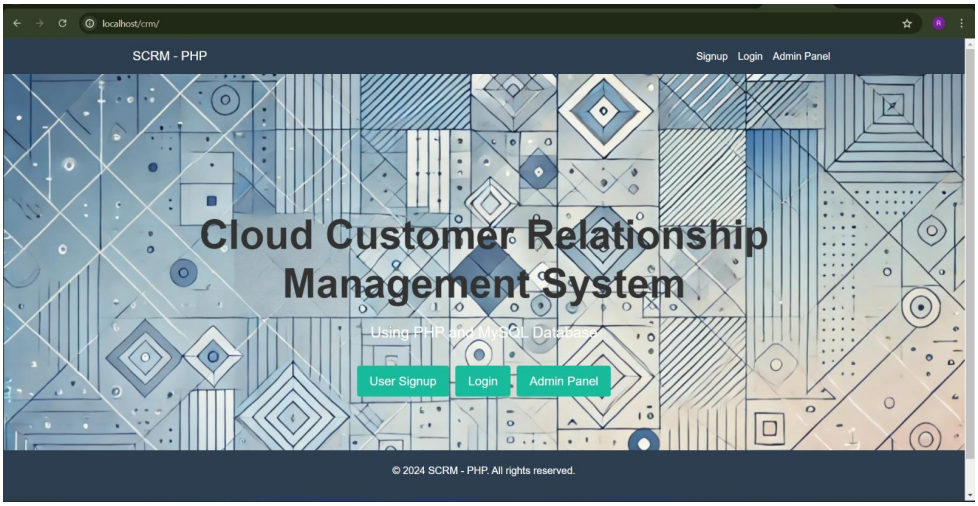
- **Analyze Network Traffic:** Monitor and analyze data packets to detect any issues or inefficiencies.

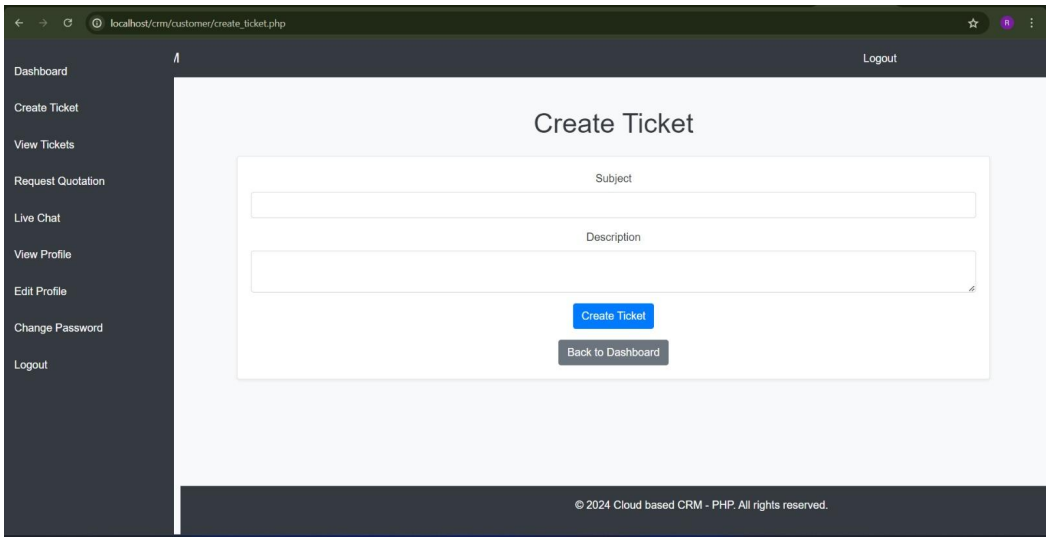
### 2.7.4 Test of Main Function

Ensuring that all core functionalities work correctly and meet the requirements.

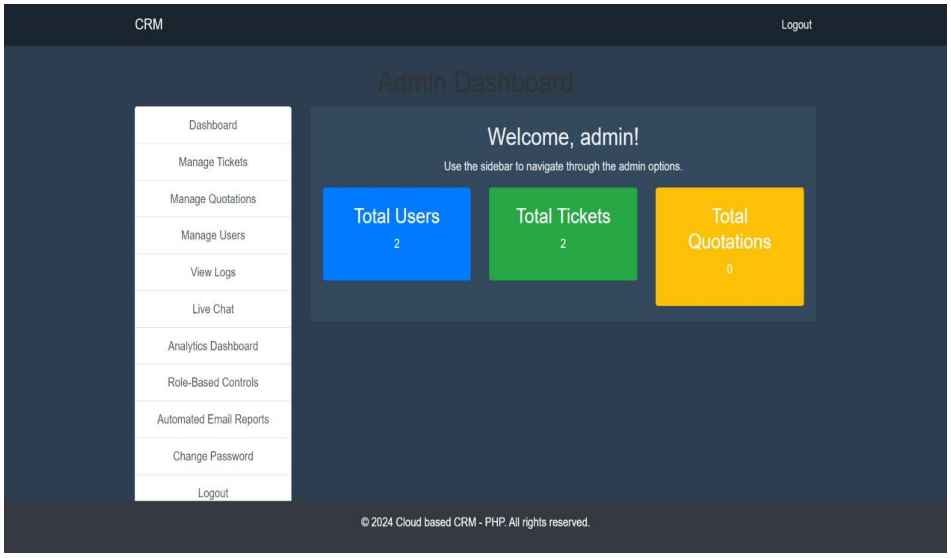
## 3. Snapshots of the Project

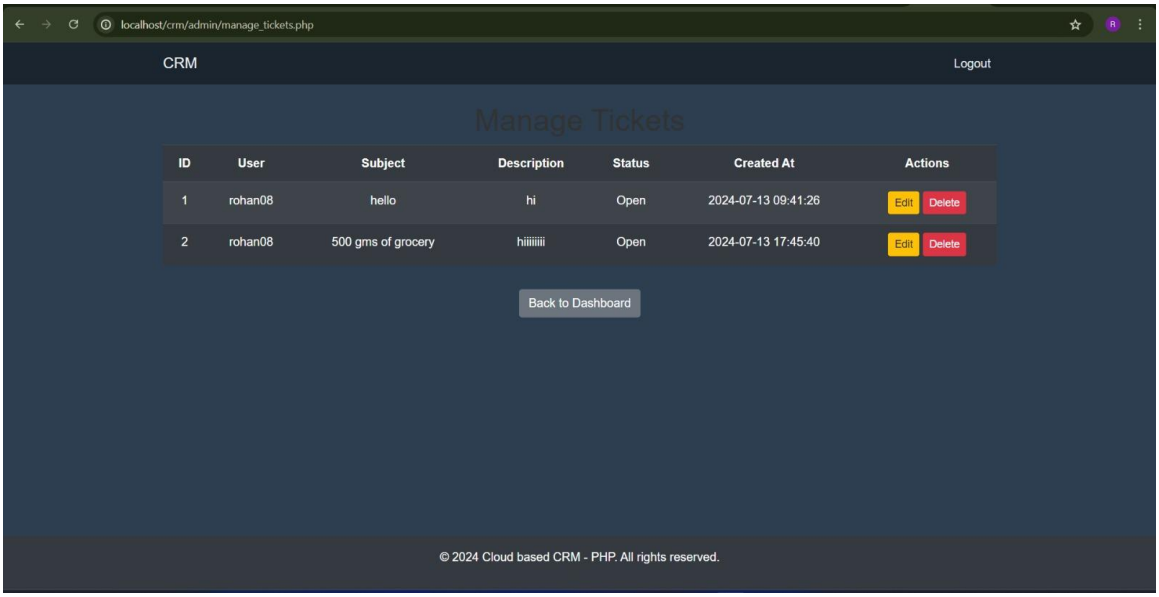
### Customer



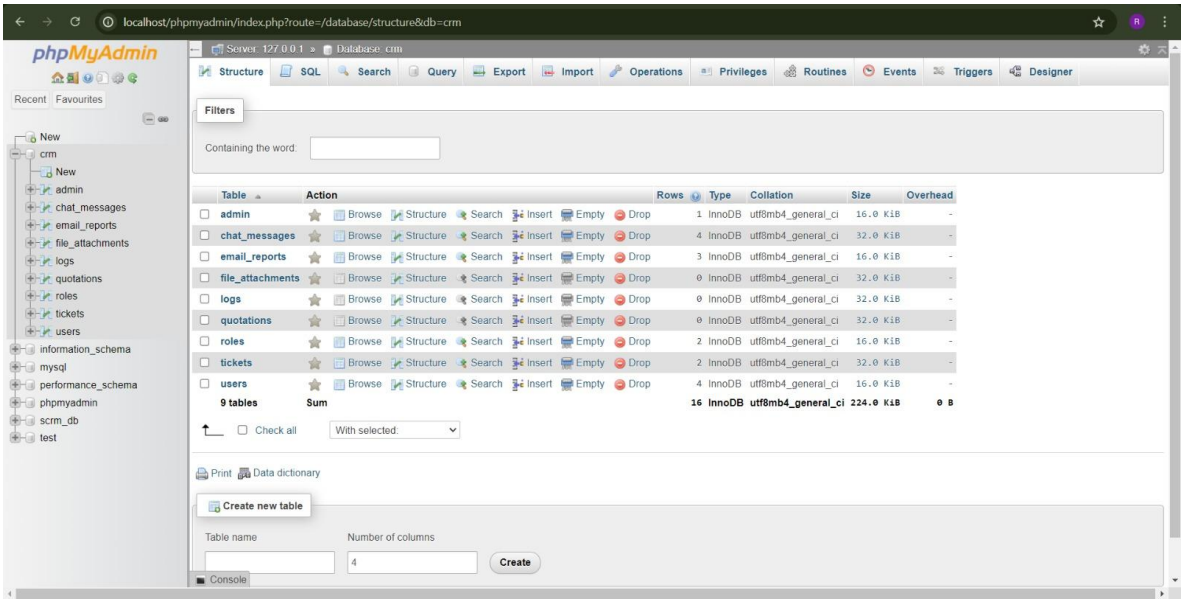



# Admin


CRM
Logout
Admin Dashboard
Dashboard
Manage Tickets
Manage Quotations
Manage Users
View Logs
Live Chat
Analytics Dashboard
Role-Based Controls
Automated Email Reports
Change Password
Logout
Welcome, admin!
Use the sidebar to navigate through the admin options.
Total Users
2
Total Tickets
2
Total Quotations
0
© 2024 Cloud based CRM - PHP. All rights reserved.


localhost/crm/admin/manage_tickets.php
CRM
Logout
Manage Tickets
ID
User
Subject
Description
Status
Created At
Actions
1
rohan08
hello
hi
Open
2024-07-13 09:41:26
Edit
Delete
2
rohan08
500 gms of grocery
hiiiiiiii
Open
2024-07-13 17:45:40
Edit
Delete
Back to Dashboard
© 2024 Cloud based CRM - PHP. All rights reserved.

## Database




## 4. Conclusions

The CRM project has successfully achieved its primary objective of developing a comprehensive customer relationship management system designed to enhance business operations and customer interactions. Key conclusions from the project include:

### 4.1. Effective Implementation of Technologies:

The integration of PHP and MySQL for backend development, along with HTML, JavaScript, CSS, and Bootstrap for the frontend, has resulted in a well-rounded and functional system. PHP's flexibility and MySQL's robustness have ensured reliable data management and server-side processing, while HTML, CSS, JavaScript, and Bootstrap have contributed to a responsive and user-friendly interface.

### 4.2. Enhanced Customer Management:

The CRM system provides businesses with a centralized platform for managing customer data, interactions, and sales processes. Features such as customer profiles, sales tracking, and automated marketing campaigns have been effectively implemented, allowing businesses to streamline operations and improve customer engagement.

### 4.3.User-Centric Design:

The use of Bootstrap in conjunction with HTML, CSS, and JavaScript has resulted in a modern and intuitive user interface. The system's design focuses on ease of use, accessibility, and responsiveness, ensuring that users can navigate and utilize the system efficiently across various devices and screen sizes.

4. **4. Robust Testing and Quality Assurance:**

A comprehensive testing strategy has been employed to ensure the system meets high standards of functionality, performance, and security. Tests conducted include system, performance, security, and user acceptance testing. The findings from these tests have been used to refine and optimize the system, addressing any issues and ensuring reliability.

### 4.5. Scalability and Future-Proofing:

The system's architecture is designed to be scalable, supporting future growth and additional features. The use of cloud-based infrastructure ensures that the system can handle increasing data volumes and user loads. Future enhancements and updates can be integrated seamlessly into the existing framework.

### 4.6.Positive Impact on Business Operations:

The CRM system is expected to significantly improve business operations by providing actionable insights, automating repetitive tasks, and enhancing customer relationship management. The system's capabilities are aligned with industry best practices and standards, positioning it as a valuable tool for businesses aiming to optimize their customer management processes.

## 5. Further development or research

Future development of the CRM system will focus on integrating advanced analytics and AI for deeper insights, developing cross-platform mobile applications, and enhancing interoperability with third-party services. Performance optimization efforts will include improving scalability with cloud solutions, implementing caching mechanisms, and refactoring code for better efficiency. Security will be bolstered with advanced measures like WAF and IDS, regular audits, and compliance updates for regulations such as GDPR and CCPA. User experience will be enhanced through continuous UI/UX improvements and customization options. Research into emerging technologies such as IoT and blockchain will be pursued to explore new capabilities. Additionally, fostering community contributions through open-source releases and building partnerships with technology providers and academic institutions will drive innovation and continuous improvement.

## 6. References

[1] **"Cloud Computing: Concepts, Technology & Architecture"** by Thomas Erl, Zaigham Mahmood, and Ricardo Puttini

[2] **"Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems"** by Martin Kleppmann

[3] **"Firebase Essentials: A Guide to Firebase and Firestore"** by Martin T. Hargreaves

[6] "Cloud-Based CRM Systems: A Review" by International Journal of Scientific and Research Publications.

[7] "Evaluating the Benefits of Cloud-Based CRM Systems for Small and Medium Enterprises" by IEEE Xplore

[8] "Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation" by Jez Humble and David Farley

[9] "Clean Code: A Handbook of Agile Software Craftsmanship" by Robert C. Martin

[10] "Test Driven Development: By Example" by Kent Beck

[11] "Designing Data-Intensive Applications" by Martin Kleppmann

## 7. Appendix

The appendix section of your report should include supplementary materials that provide additional context or details relevant to the CRM project. This may include technical documentation, additional diagrams, code snippets, data samples, or other supporting materials. Below is a structured example of what you might include in the appendix:

### 7. 1 Technical Documentation

#### 7.1.1 System Architecture Diagram

- **Description:** Visual representation of the overall system architecture, including frontend, backend, and database components.

#### 7.1.2 Database Schema

- **Description:** Detailed schema of the MySQL database, including tables, relationships, and key constraints.

### 7.1.3 API Documentation

- **Description:** Documentation for any APIs used or developed, including endpoints, request/response formats, and authentication methods.

## 7.2 Code Samples

### 7.2.1 Backend Code Snippets

- **Description:** Examples of PHP code for key functionalities such as data retrieval, business logic, and API integration.

### 7.2.2 Frontend Code Snippets

- **Description:** Examples of HTML, JavaScript, and CSS used for the user interface, including custom components and interactions.

## 7.3 Data Samples

### 7.3.1 Sample Customer Data

- **Description:** Example data used for testing and validation, including customer profiles, interactions, and transaction records.

### 7.3.2 Test Data Sets

- **Description:** Data sets used for performance and stress testing, including volume and load test scenarios.

## 7.4 Testing Materials

### 7.4.1 Test Plan and Cases

- **Description:** Detailed test plan and individual test cases used for validating the CRM system.

### 7.4.2 Test Results

- **Description:** Summary of testing results, including performance metrics, security findings, and user acceptance feedback.

## 7.5 User Manuals and Guides

### 7.5.1 User Manual

- **Description:** Comprehensive guide for end-users, including instructions on system usage, navigation, and troubleshooting.

### 7.5.2 Administrator Guide

- **Description:** Guide for system administrators, covering setup, configuration, and maintenance procedures.

## 7.6 Additional Resources

### 7.6.1 Reference Materials

- **Description:** List of books, articles, and online resources referenced during the project.

### 7.6.2 Glossary

- **Description:** Definitions of key terms and concepts related to the CRM system.