Software Requirements Specification (SRS)

COURSE TITLE : Software Engineering and System Analysis Lab
COURSE CODE : CSE 356

Date: 27/05/2025

Prepared By:
Group Number : 06
Group Members :

1. NAME : Fahim Sahariar Alvi	**2. NAME : Mehedi Hasan**
ID : 0432220005101026	**ID : 0432220005101033**
SECTION : 6A1	**SECTION : 6A1**
3. NAME :Taposhi Rabia Fardin	**4. NAME : Tonny Talukder**
ID : 0432220005101049	**ID : 0432220005101050**
SECTION : 6A1	**SECTION : 6A1**

Institution:
University of Information Technology & Sciences (UITS)

Reviewed By:
Engineer Md. Safaet Hossain
Lecturer,
Department of Computer Science and Engineering,
University of Information Technology & Sciences (UITS)

Software Requirements Specification (SRS) for Advanced Customer Relationship Management (CRM)-based Live Chat System

1. Introduction

1.1 Purpose

This document presents a detailed Software Requirements Specification (SRS) for a Customer Relationship Management (CRM) System. The primary goal of this system is to enable administrators to manage customer data efficiently through a simple, user-friendly web interface.

1.2 Scope

The CRM system is a web-based application developed using React.js and Next.js for the frontend, with json-server providing mock backend functionality. It allows administrators to perform essential CRUD operations—Create, Read, Update, and Delete—on client data. The system is designed for academic use and runs locally on a development server.

1.3 Definitions, Acronyms, and Abbreviations

- **CRM**: Customer Relationship Management
- **CRUD**: Create, Read, Update, Delete
- **UI**: User Interface
- **API**: Application Programming Interface
- **JSON**: JavaScript Object Notation
- **React.js**: A JavaScript library for building user interfaces.
- **Redux**: A state management library for predictable application behavior.

2. Overall Description

2.1 Product Perspective

This CRM system is a self-contained academic project. It simulates the basic features of a commercial CRM by utilizing a frontend interface and a mock backend (json-server) that reads and writes to a local JSON file (db.json).

2.2 Product Functions

The system offers the following features:

- **Secure admin login**
- **View all registered clients**
- **Add a new client with contact details**
- **Edit an existing client's information**
- **Delete clients from the system**
- **Search and filter client records**

2.3 User Classes and Characteristics

- **Administrator**: Has full access to all system functionalities including managing client records.

2.4 Operating Environment

- **Local machine with Node.js installed**
- **Browser-based interface**
- **Backend runs using json-server on port 4000**

2.5 Design and Implementation Constraints

- **System is intended strictly for academic demonstration**
- **No persistent database; data is stored in a local JSON file**
- **Admin login is hardcoded for testing purposes**

3. Specific Requirements

3.1 Functional Requirements

- **FR1**: The system shall authenticate the admin using predefined credentials.
- **FR2**: The system shall display a list of all clients upon successful login.
- **FR3**: The system shall allow the admin to add a new client.
- **FR4**: The system shall allow the admin to edit existing client details.
- **FR5**: The system shall allow the admin to delete any client.
- **FR6**: The system shall support searching/filtering clients by name or email.

3.2 Non-Functional Requirements

- **NFR1**: The UI shall be intuitive and responsive across devices.
- **NFR2**: The system shall handle input validation and display relevant error messages.
- **NFR3**: The system shall use visual alerts (e.g., SweetAlert2) for confirmation and notifications.

4. External Interface Requirements

4.1 User Interfaces

- **Login Page**: For admin authentication
- **Dashboard**: Displays client list and actions (add, edit, delete)
- **Form Pages**: For entering and editing client data

4.2 Software Interfaces

- **json-server**: Serves as a mock RESTful API
- **Axios**: Used for making HTTP requests to the backend

4.3 Hardware Interfaces

- **No special hardware required; runs on any modern personal computer with internet browser**

5. Appendices

5.1 Tools and Technologies Used

- **React.js**
- **Next.js**
- **Redux**
- **Axios**
- **Formik + Yup**
- **SweetAlert2**
- **json-server**

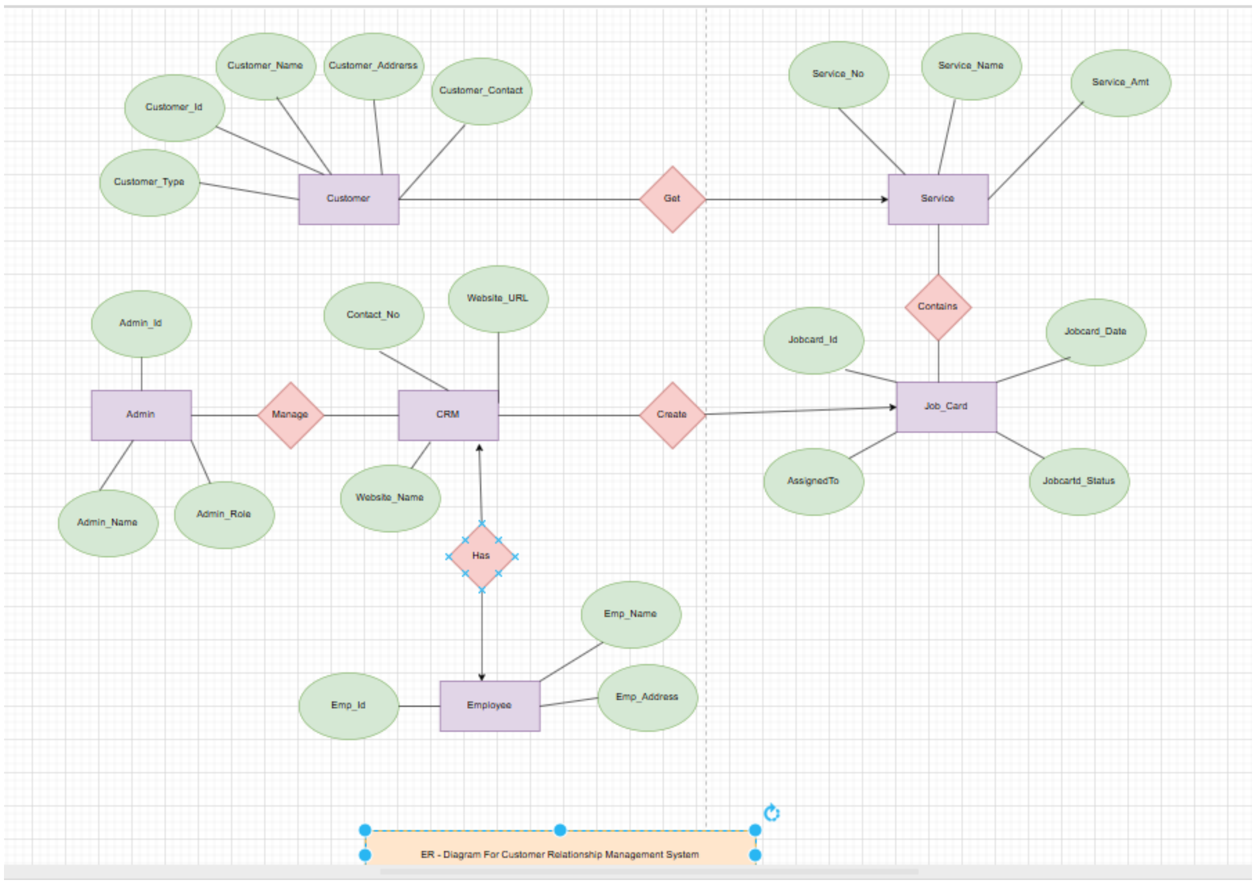
5.2 Admin Credentials for Testing

- **Email**: **admin@admin.com**
- **Password: 1234**
  Note: These credentials are hardcoded for academic testing purposes only and should not be used in a production environment.

6. Conclusion

This CRM system successfully demonstrates the core functionalities required for managing customer data in a simplified and structured way. Built entirely with modern web technologies and designed for local use, it serves as a strong academic prototype. The project achieves its objective by focusing solely on essential CRM features—making it lightweight, efficient, and ideal for student learning and presentation purposes.

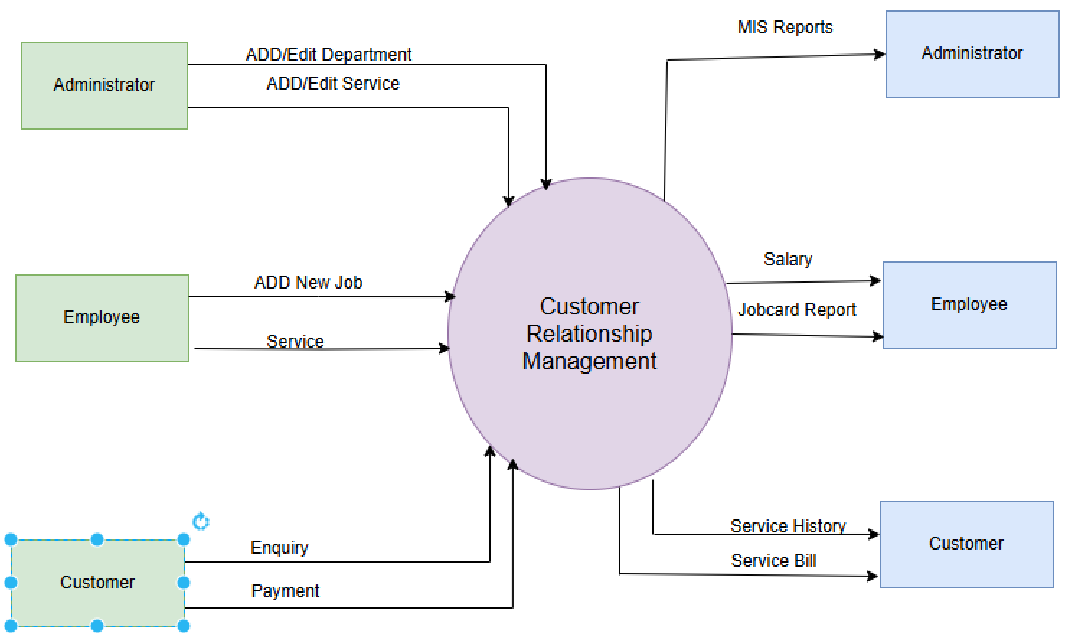
1. ER-DIAGRAM :



Description :

This is an Entity-Relationship (ER) Diagram for a Customer Relationship Management (CRM) System. It illustrates the relationships between various entities involved in managing customer service and internal operations. Key Components: Entities (rectangles): Customer, Admin, CRM, Service, Job_Card, Employee Attributes (ellipses): Describe each entity, e.g., Customer_Id, Service_Name, Jobcard_Status, etc. Relationships (diamonds): Describe interactions: Customer “Gets” Service CRM “Manages” by Admin CRM “Creates” Job_Card Job_Card “Contains” Service CRM “Has” Employee This diagram helps visualize how different parts of a CRM system interact to manage customers, services, job assignments, and employees.

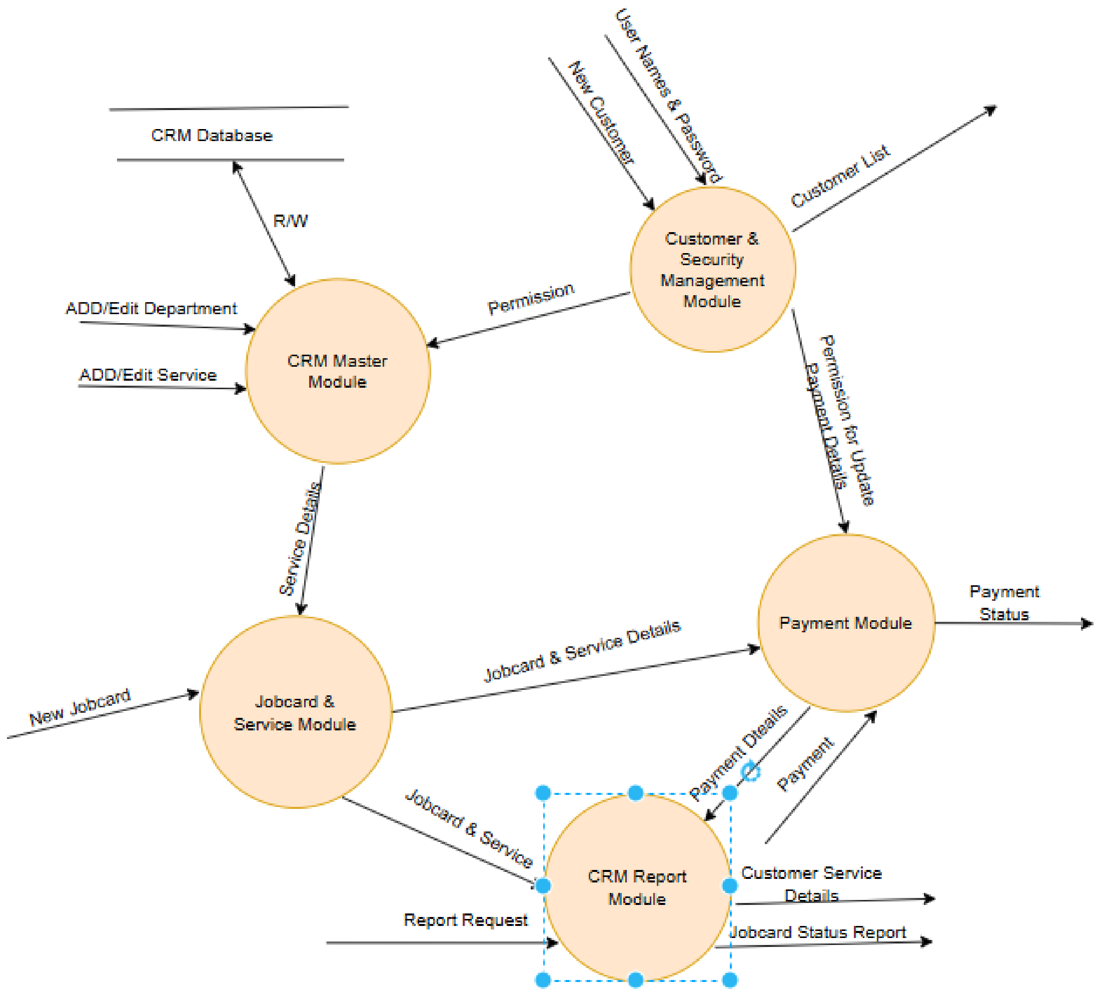
2.a) DFD0 :



Description :

This diagram represents a Customer Relationship Management (CRM) system showing interactions between different user roles (Administrator, Employee, Customer) and the CRM system. Administrators can add/edit departments and services, and receive MIS reports. Employees can add new jobs and services, and receive salary and job card reports. Customers can make enquiries and payments, and receive service history and service bills. The CRM system serves as the central hub for managing tasks, data, and communication between these entities.

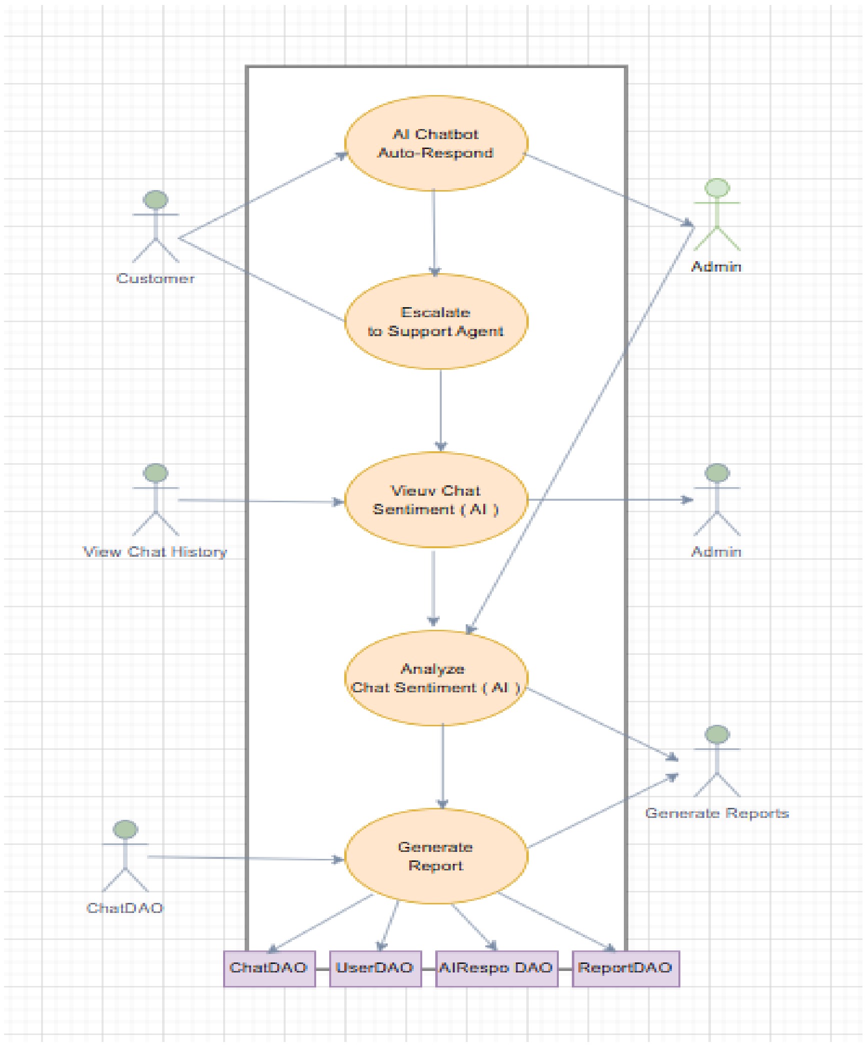
2.b) DFD 1 :



Description :

This is a CRM System Architecture Diagram, showing the interaction between different modules within a CRM application. Here's a short description: The diagram outlines the modular structure of a CRM system, with each module handling specific business functions: Customer & Security Management Module: Manages customer data, user credentials, and access permissions. CRM Master Module: Central module for managing departments and services; communicates with the CRM database. Jobcard & Service Module: Handles jobcard creation and service details. Payment Module: Manages payments, updates status, and ensures secure transactions. CRM Report Module: Generates reports based on jobcard, service, customer, and payment details. Modules interact through data exchange (e.g., jobcard details, service info, permissions), with a central database facilitating read/write operations. This design supports a structured, integrated CRM workflow.

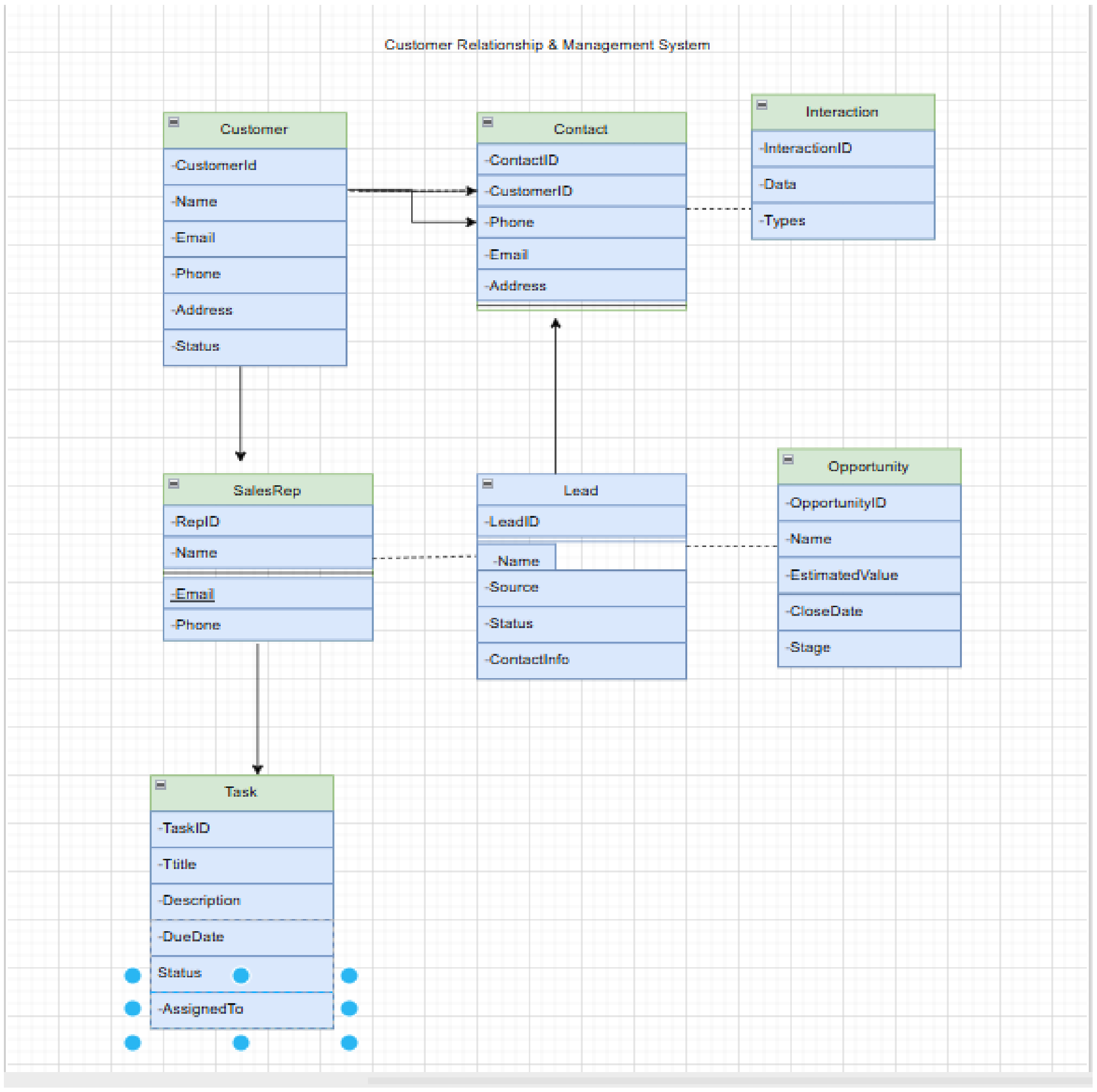
3. Use Case :



Description :

This is an Activity Diagram for an AI-based Chat Support System. Key Highlights: Actors: Customer interacts with the chatbot. Admin monitors and generates reports. Process Flow: 1. AI Chatbot Auto-Responds to customer queries. 2. If unresolved, it escalates to a support agent. 3. The system allows users to view chat sentiment using AI. 4. Admin can analyze chat sentiment for deeper insights. 5. Finally, the system generates reports based on chat data. Data Access Objects (DAOs): Used for database operations: ChatDAO, UserDAO, AIRespDAO, ReportDAO. This diagram visualizes how AI and admin roles work together to enhance customer service and generate analytical reports.

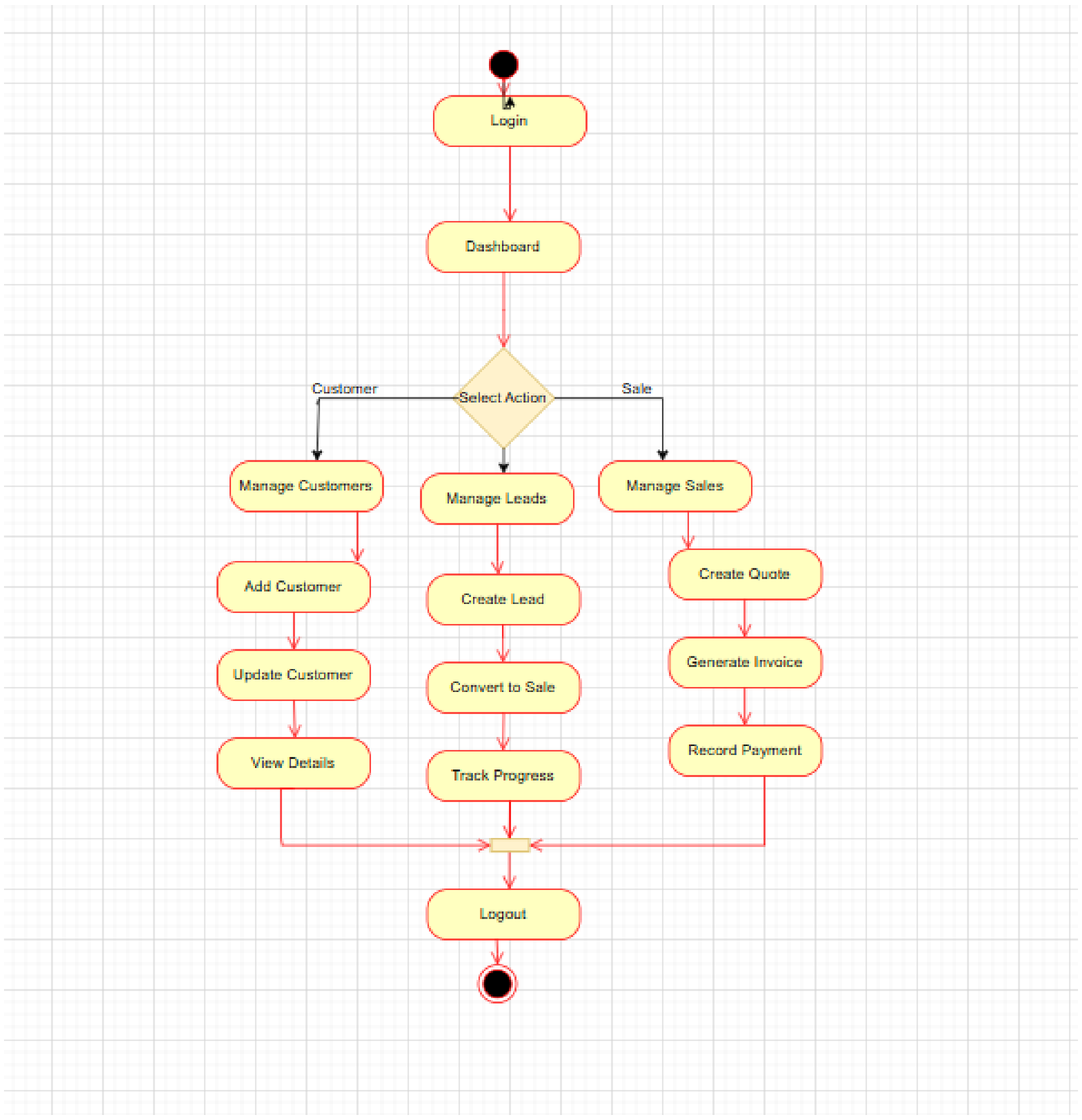
4. Class Diagram :



Description :

This is an Entity-Relationship Diagram (ERD) for a Customer Relationship & Management System (CRM). Here's a short description: The CRM system manages customer data, interactions, leads, sales opportunities, tasks, and sales representatives. Key entities include: Customer: Holds customer information. Contact: Linked to a customer, stores individual contact details. Interaction: Logs communications with contacts. Lead: Tracks potential customers, linked to sales reps. Opportunity: Represents potential sales tied to leads. SalesRep: Stores information about sales representatives. Task: Tracks tasks assigned to sales reps, related to customer activities. Relationships between entities enable comprehensive tracking of customer engagement and sales workflow.

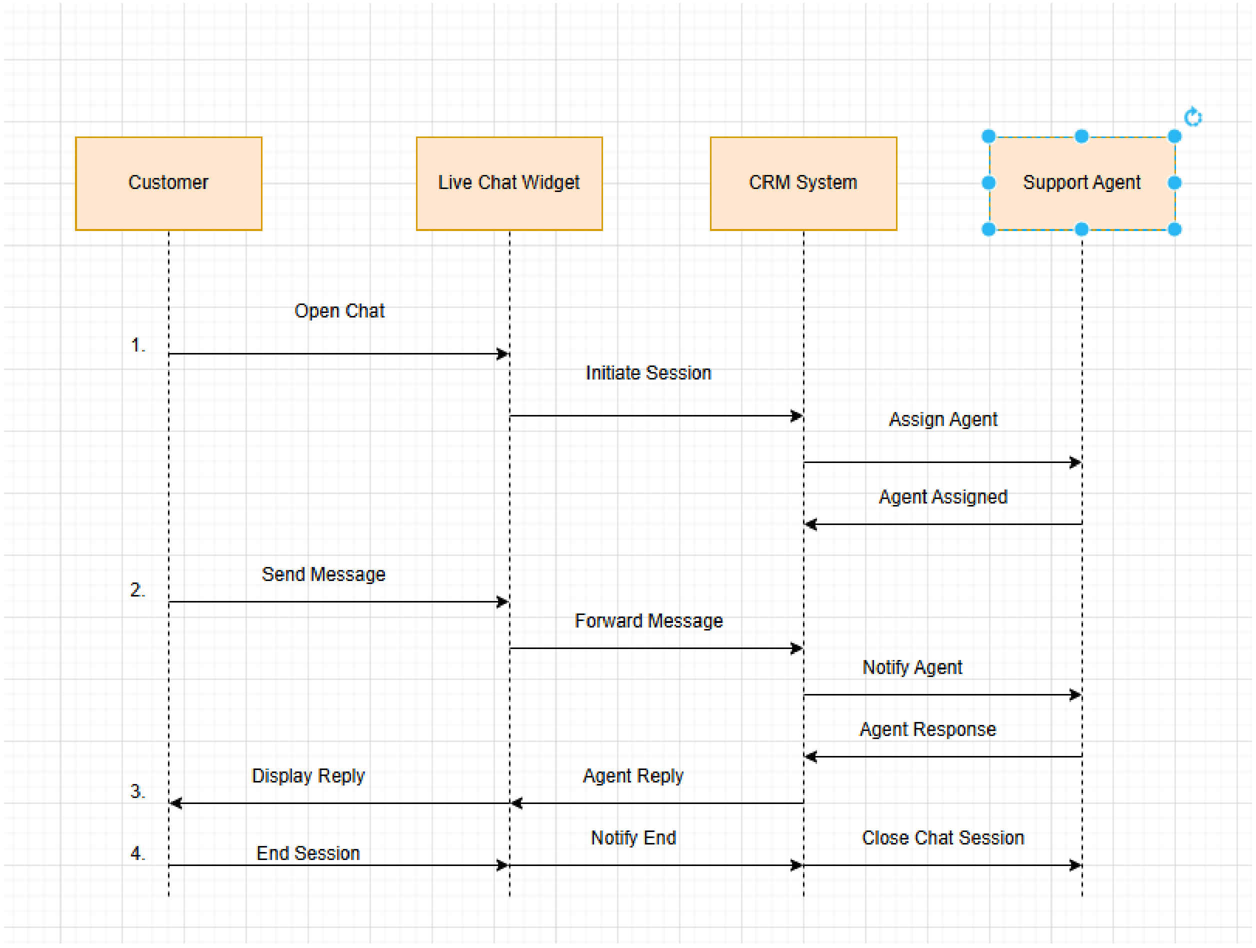
5. Activity Diagram :



Description :

This flowchart illustrates the CRM system workflow from login to logout, covering customer, lead, and sales management. After login, the user accesses the dashboard and selects an action. Customer management allows adding, updating, and viewing customer details. Lead management involves creating leads, converting them to sales, and tracking progress. Sales management includes creating quotes, generating invoices, and recording payments. All workflows lead to the logout process, completing the session. The flow provides a structured and user-friendly approach to handling CRM operations efficiently.

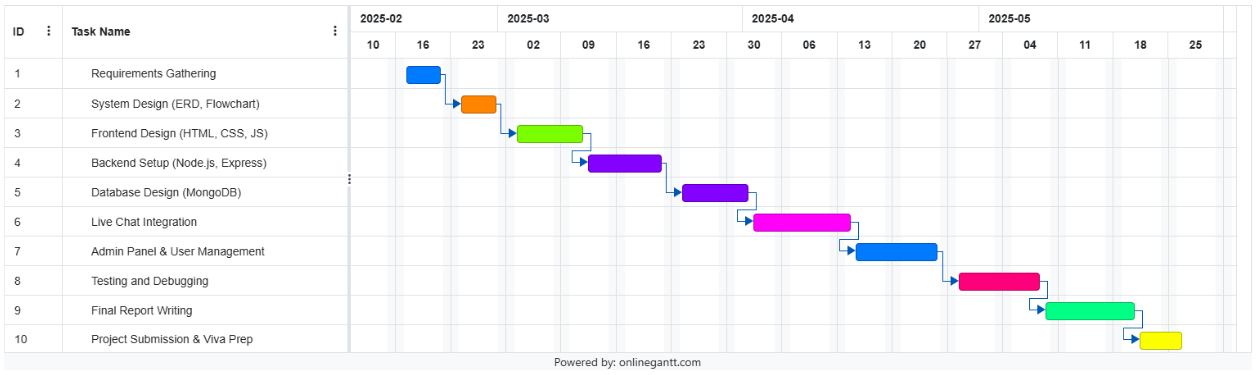
6. Sequence Diagram :



Description :

This sequence diagram illustrates the live chat support process in a CRM system. 1. The customer opens a chat via the live chat widget, which initiates a session with the CRM system. 2. The CRM system assigns a support agent, who is notified of the incoming chat. 3. Messages sent by the customer are forwarded to the agent, and the agent's response is relayed back to the customer. 4. When the conversation ends, the session is closed by the CRM, and all parties are notified. The diagram shows real-time communication flow between customer, system, and support agent.

7. Scheduling :

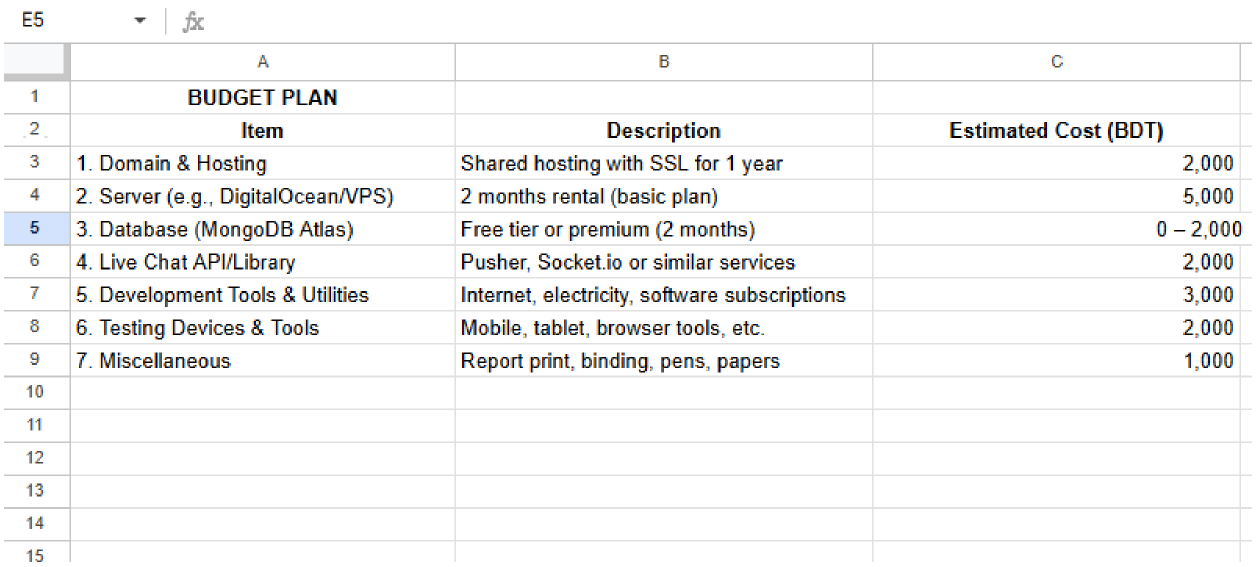


ID	Task Name
1	Requirements Gathering
2	System Design (ERD, Flowchart)
3	Frontend Design (HTML, CSS, JS)
4	Backend Setup (Node.js, Express)
5	Database Design (MongoDB)
6	Live Chat Integration
7	Admin Panel & User Management
8	Testing and Debugging
9	Final Report Writing
10	Project Submission & Viva Prep

Description :

This is a Gantt Chart illustrating the project timeline for a software development project, likely a Customer Relationship Management (CRM) system. Key Highlights: Timeline: February to May 2025 Tasks: 10 sequential tasks including: 1. Requirements Gathering 2. System Design (ERD, Flowchart) 3. Frontend & Backend Development 4. Database Design (MongoDB) 5. Live Chat Integration 6. Admin Panel & User Management 7. Testing & Debugging 8. Final Report Writing 9. Project Submission & Viva Preparation Each task is visually represented with colored bars showing duration and dependencies, helping track progress and manage deadlines efficiently.

8. Budget :



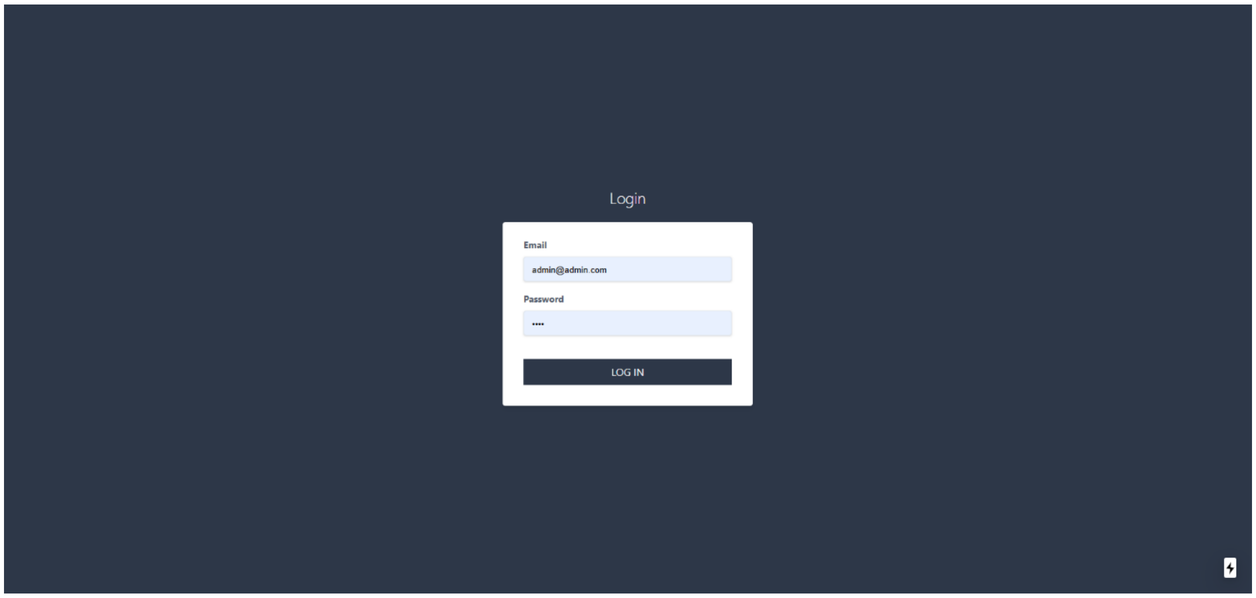
BUDGET PLAN

Item	Description	Estimated Cost (BDT)
1. Domain & Hosting	Shared hosting with SSL for 1 year	2,000
2. Server (e.g., DigitalOcean/VPS)	2 months rental (basic plan)	5,000
3. Database (MongoDB Atlas)	Free tier or premium (2 months)	0 – 2,000
4. Live Chat API/Library	Pusher, Socket.io or similar services	2,000
5. Development Tools & Utilities	Internet, electricity, software subscriptions	3,000
6. Testing Devices & Tools	Mobile, tablet, browser tools, etc.	2,000
7. Miscellaneous	Report print, binding, pens, papers	1,000

Description :

This appears to be a sample budget plan for a web or software development project. It outlines estimated costs for essential components such as hosting, servers, databases, live chat services, development utilities, testing tools, and miscellaneous items. The plan is likely a placeholder or mock-up used for planning or presentation purposes, rather than reflecting a specific, real project.

9. Software Testing :



Login

Email: admin@admin.com

Password: ····