Dr. Babasaheb Ambedkar Marathwada University, chhatrapati sambhajinagar,

A Major Project Synopsis on

CHAT APPLICATION

Submitted By:

Mr. Omkar Gorde

Mr. Rajesh Gayke

Mr. Sagar kale

Guided By:

Ms. Priyanka Hiwrale.

Ms. Namrata kulkarni.

In partial Fulfilment for the Degree Of B.Sc. (Computer Science)

in Faculty Of Science

Submitted To:

Vivekanand Shikshan Santha's Vivekanand Arts, Sardar Dalipsingh Commerce & Science College, Aurangabad

Academic Year 2024-2025

Introduction to the Project:

Introduction to a Chat Application Project

In today's hyper-connected world, real-time communication is essential. Chat applications have become ubiquitous, facilitating instant messaging, file sharing, and even voice/video calls. This project aims to develop a robust and user-friendly chat application that enables seamless communication among individuals or groups.

Key Objectives:

Efficient Communication: Enable users to engage in instant messaging with friends, colleagues, or groups, fostering efficient and effective communication.

User-Friendly Interface: Design an intuitive and easy-to-navigate interface that enhances user experience and encourages engagement.

Real-time Updates: Ensure messages are delivered and displayed immediately as they are sent, providing a responsive and interactive experience.

Cross-Platform Compatibility: Support a wide range of devices and operating systems, allowing users to access the application from anywhere.

Security and Privacy: Implement robust security measures to protect user data and ensure privacy.

Introduction to Technology and Database for Chat Applications:

Chat applications have become an integral part of our daily lives, facilitating communication and connection across the globe. Understanding the underlying technologies and databases that power these applications is crucial for developers and enthusiasts alike.

Core Technologies

Frontend:

HTML, CSS, and JavaScript: These are the foundational languages for building the user interface of a chat application. HTML provides the structure, CSS handles the styling, and JavaScript adds interactivity and dynamic behavior.

* Frontend Frameworks: Popular frameworks like React, Angular, and Vue.js provide reusable components, efficient data binding, and streamlined development processes.

Backend:

Programming Languages: Languages like Python, Node.js, Java, and Ruby are commonly used for backend development. They handle server-side logic, database interactions, and communication with the frontend.

Real-time Communication: Technologies such as WebSockets, Socket.IO, and Firebase enable real-time, bidirectional communication between the server and clients, ensuring messages are delivered instantly.

API Development: RESTful APIs are used to define how different components of the application interact with each other, enabling modularity and scalability.

Visibility Study:

A visibility study for a chat application aims to understand user needs and preferences to ensure the application's success. Key aspects to consider include:

Target Audience:

Demographics (age, gender, location)

Interests and hobbies

Technological proficiency

Communication preferences (text, voice, video)

User Needs:

Primary use cases (e.g., personal

messaging, group chats, professional communication)

Desired features (e.g., file sharing, voice/ video calls, group video calls, encryption)

Accessibility requirements (e.g., screen reader compatibility)

Competitive Analysis:

Identify existing chat applications and their strengths/weaknesses

Analyze user reviews and feedback

Platform Considerations:

Mobile (iOS, Android)

Desktop (Windows, macOS, Linux)

Web-based

Hardware Requirements

Server:

CPU: Multi-core processor with high clock

Speed
RAM: At least 8GB, depending on expected

user load

Storage: Sufficient SSD storage for database and application files

Network: High-speed, reliable internet connection

Client Devices:

CPU: Minimum 1.5 GHz dual-core processor

RAM: At least 2GB

Storage: Sufficient space for application

installation

Network: Stable internet connection

Software Requirements

Server-Side:

Programming Language:PHP, Python, Node.js, Java, or other suitable languages

Framework: Express.js (Node.js), (Python), (Java)
Database: Relational (MySQL,SQL)

Real-time Communication: WebSockets, Socket.IO, or other technologies

Client-Side:

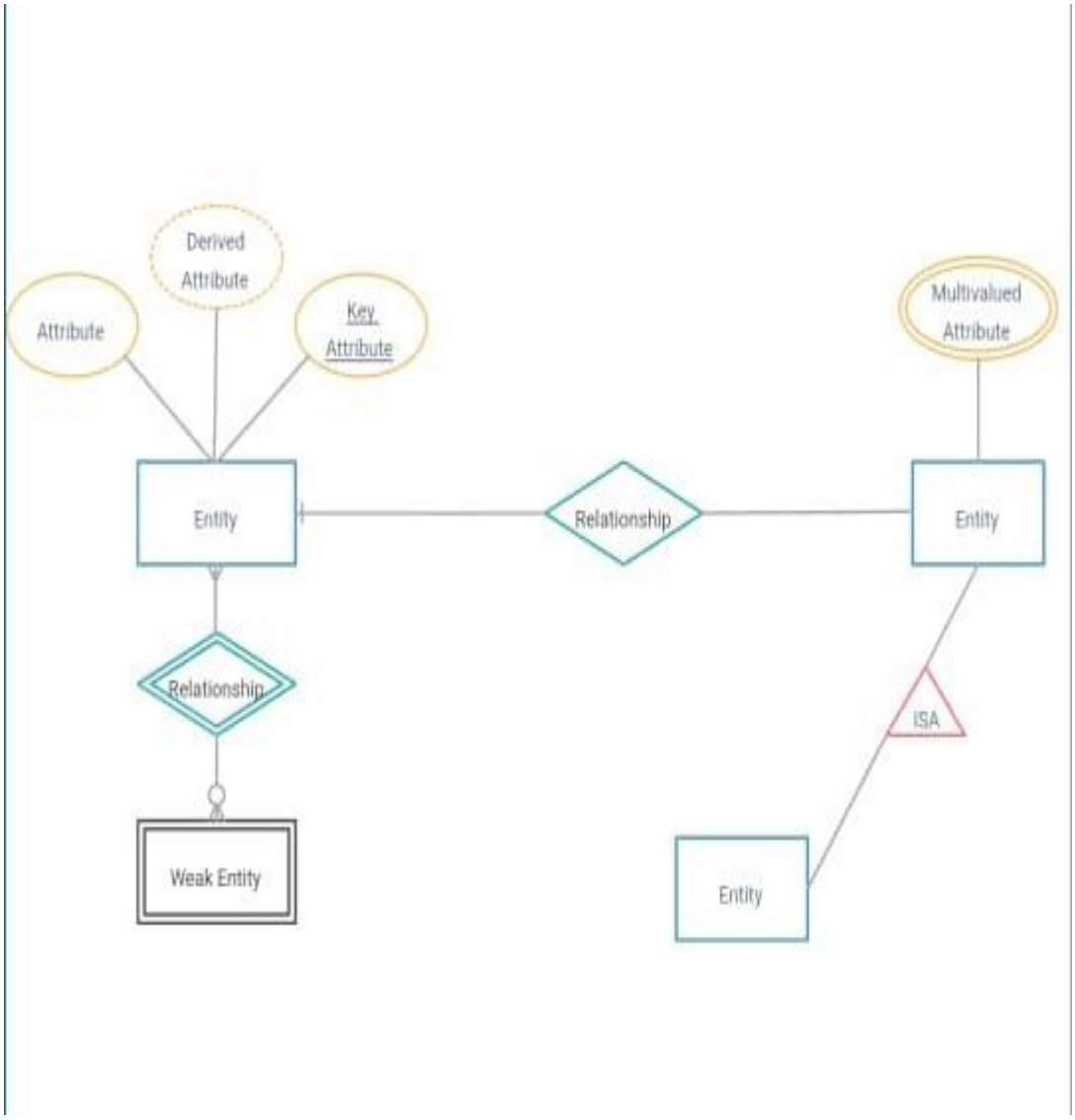
Programming Languages: HTML, CSS, JavaScript

Additional Tools:

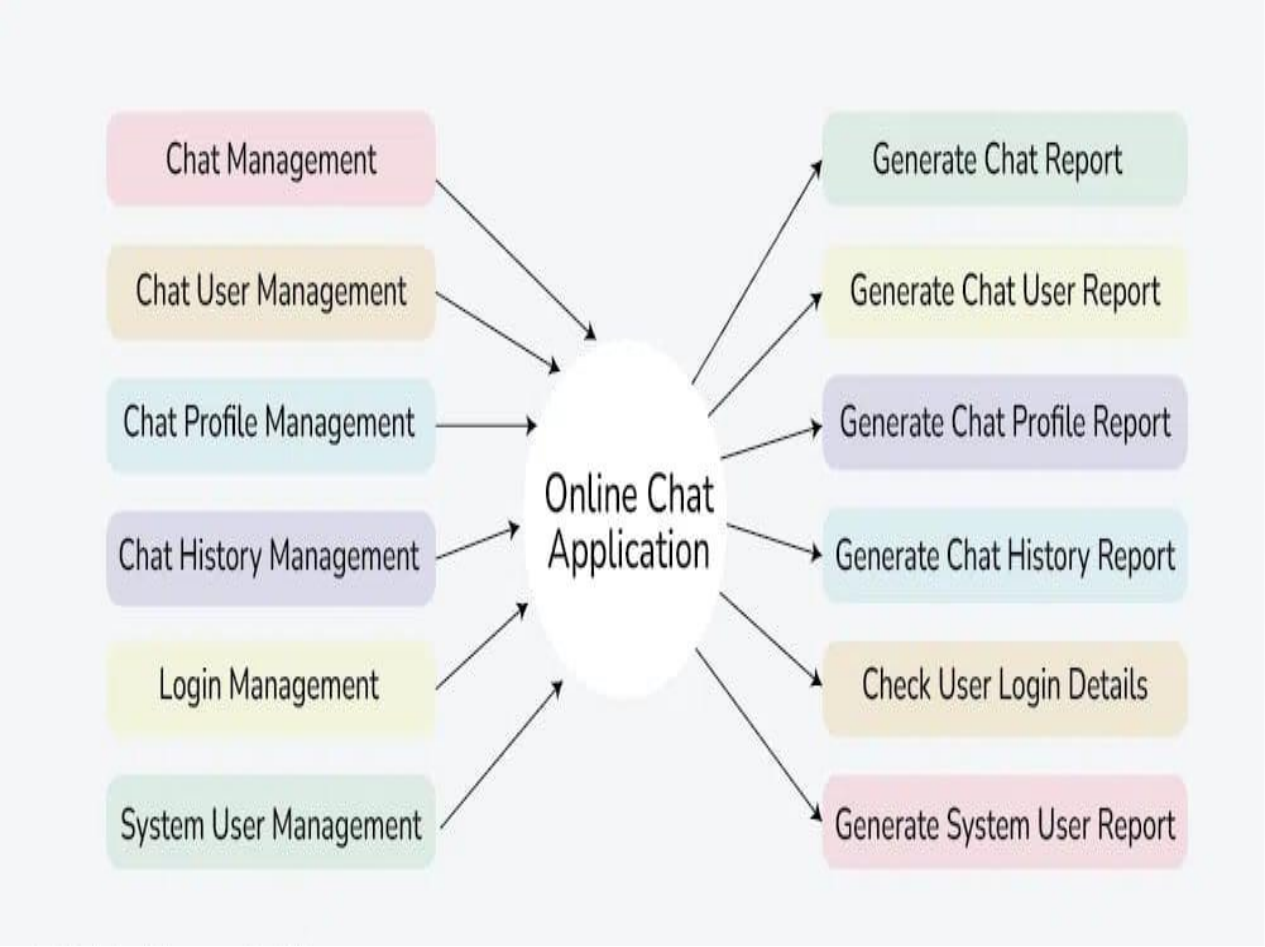
Cloud Services: AWS, Azure, Google Cloud

Platform (optional)

ER DIAGRAM:


Derived Attribute
Attribute
Key Attribute
Multivalued Attribute
Entity
Relationship
Entity
Relationship
ISA
Weak Entity
Entity

DFD DIAGRAM:


Chat Management
Chat User Management
Chat Profile Management
Chat History Management
Login Management
System User Management
Online Chat Application
Generate Chat Report
Generate Chat User Report
Generate Chat Profile Report
Generate Chat History Report
Check User Login Details
Generate System User Report

Data Dictionary Chat Application: A Conceptual Overview

Core Concept

A data dictionary chat application is a specialized communication tool designed to facilitate the creation, management, and discussion of data dictionaries within a collaborative environment. It combines the interactive nature of chat with the structured organization of a data dictionary.

Key Features

1. Data Dictionary Editor:

A visual interface for creating and modifying data dictionary entries.

Supports various data types (text, numbers, dates, etc.) and metadata fields (name, description, data type, length, format, etc.).

Drag-and-drop functionality for organizing and linking data elements.

Version control to track changes and revert to previous versions.

2. Chat Functionality:

Real-time messaging for communication between team members.

Ability to attach data dictionary entries or excerpts to chat messages.

Contextual chat threads tied to specific data elements for focused discussions.

Integration with external communication platforms (optional).

3. Collaboration Tools:

Shared workspace for team members to collaborate on the data dictionary.

Access control and permissions to manage user roles and access levels.

Activity feed to track changes and updates made to the data dictionary.

Commenting and annotation features for providing feedback and suggestions.

4. Search and Filtering:Powerful search capabilities to quickly find specific data elements.

Core Layout:

Conversation List:

A scrollable list displaying all active conversations.

Each conversation item should include:

Contact Info: Name, profile picture (if available), last message snippet.

Timestamps: Time of last message or "Last seen" status.

Unread Message Indicator: A badge or visual cue for unread messages.

Chat Window:

Takes up the majority of the screen when a conversation is selected.

Message History:

Scrollable list of messages sent and received in chronological order.

Clear message timestamps and sender differentiation (e.g., name/initials, color-coding).

Support for various message types: text, images, videos, files, location, etc.

Input Bar:

Text input field for typing messages.

Send button or "Enter" key functionality.