Project Proposal

On

Real-Time Chat Application (MERN + Socket.io)

Guided By :-

Anuj Kumar

Created By: -

- Himanshu Shukla
- Mohit Singh
- Pandey Vivek

AFid : -

- AF04979357,
- AF04979355,
- AF04991238

Batch Code: ANP-D2404

Course Code: WDR1

Table of Contents

1. Title of the Project
2. Introduction
3. Objectives
4. Project Category
5. Analysis
    - Modules & Description
    - Database Design
    - ER Diagram
    - Data Flow Diagram
6. Complete Structure
    - Process Logical Diagram
7. Platform Used
    - Hardware Requirement
    - Software Requirement
8. Future Scope
9. Bibliography

1. Title of the Project

Real-Time Chat Application

2. Introduction

Communication is an essential part of modern digital interaction, and real-time messaging applications have become a core necessity in social platforms, businesses, and collaborative environments.

Traditional chat systems suffer from delays, poor synchronization, and limited usability. To overcome these issues, the proposed system uses **Socket.io** for real-time data transmission combined with the **MERN Stack (MongoDB, Express.js, React.js, Node.js)** for a scalable and modern web architecture.

The application supports **JWT authentication**, personalized messaging between users, online/offline user status, and instant message delivery without page reload.

This platform delivers a smooth, fast, and secure chatting experience optimized for both personal and professional usage.

3. Objectives

Primary Objectives

- To build a **real-time chat system** using WebSocket technology (Socket.io).
- To allow users to send and receive messages instantly without refresh.
- To implement fast and secure JWT-based authentication.

Secondary Objectives

- To enable personalized **one-to-one chatting** between registered users.
- To track and display **user online/offline status** dynamically.
- **To store chat history securely in a NoSQL database (MongoDB).**

Technical Objectives

- To develop a full-stack MERN application supporting multi-user communication.
- To ensure secure data transmission using token-based authentication.
- To create a scalable backend architecture using Node.js and Express.js.
- To build a responsive and interactive frontend using React.js.

Long-Term / Business Objectives

- To scale the project into a SaaS-based messaging platform.
- To support team chats, group creation, and enterprise communication in future.
- To integrate additional features like file sharing and end-to-end encryption.

4. Project Category

Full Stack Web Application — Real-Time Communication System

This project falls under the category of **Real-Time Communication Platform** using WebSockets.

Why this is a Real-Time Application:

- Messages are delivered instantly.
- Online users are tracked live.
- No page reload is required for interactions.
- Socket.io maintains a persistent two-way communication channel.

The system is suitable for:

- Individuals
- Friends & teams
- Companies needing internal communication
- Support/chat-based service providers

5. Analysis

5.1 Modules & Description

The system is divided into multiple interconnected modules to ensure scalability, maintainability and modular development.

Module Name	Description
User Authentication Module	Login, logout, and registration using secure JWT authentication.
Chat Module	Real-time messaging between users using Socket.io.
User Status Module	Tracks online/offline presence of each user.
Message Module	Stores and retrieves chat history from MongoDB.
Frontend Module	React UI for messaging, chat windows, user list, and status view.
Admin Module (Future Scope)	To manage users, conversations, and platform analytics.

5.2 Database Design

The database is designed using MongoDB for flexible and scalable data storage.

Main Collections

- Users
- Messages
- Conversations
- Tokens (optional for refresh token)

Database Features

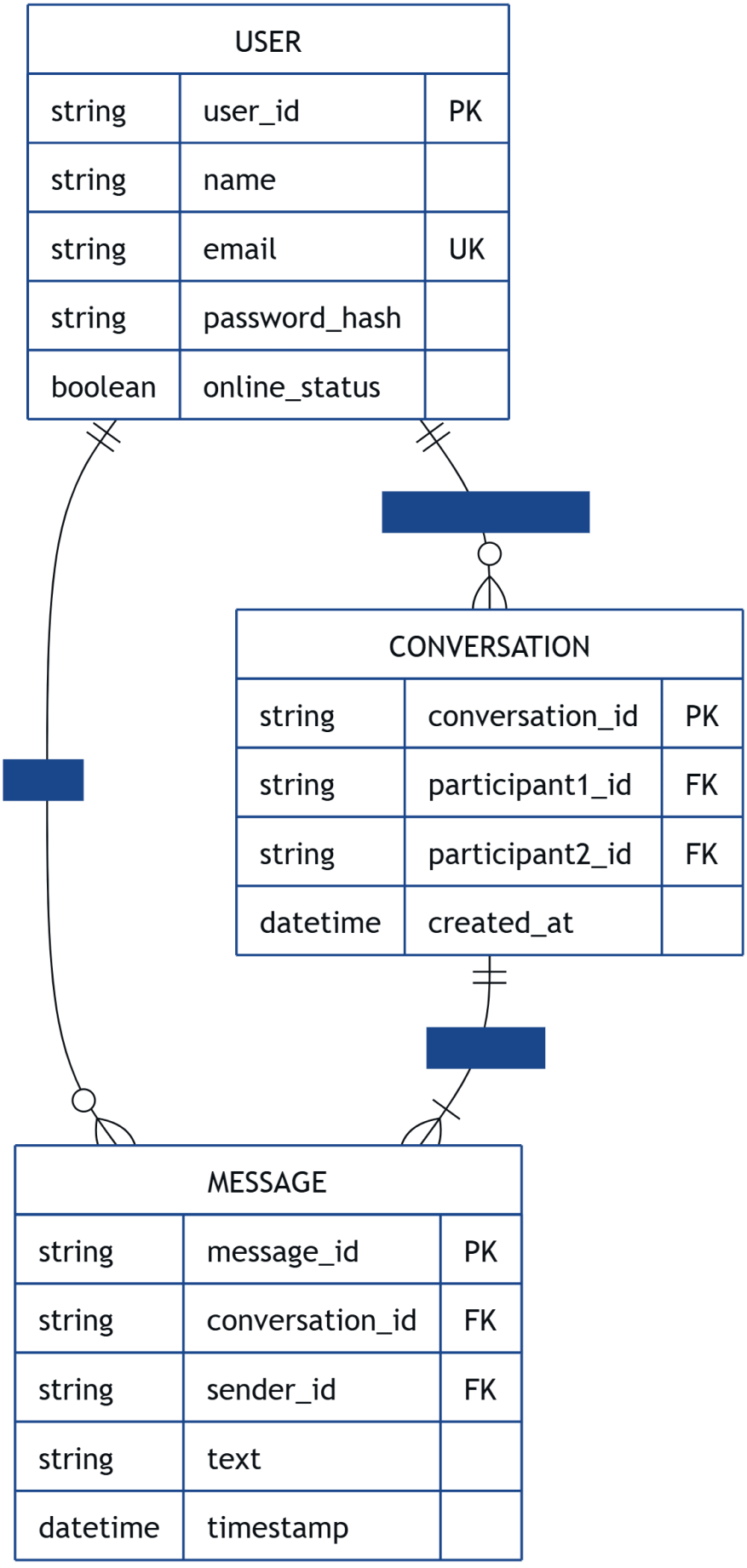
- Optimized for real-time read/write operations.
- Secure JWT token storage.
- Indexed collections for fast message retrieval.
- One-to-one chat mapping via conversation IDs.

5.3 ER Diagram (Entity Relationship Model)

The conceptual ER model visualizes the relationship between actors and database entities.

Explanation of relationships:

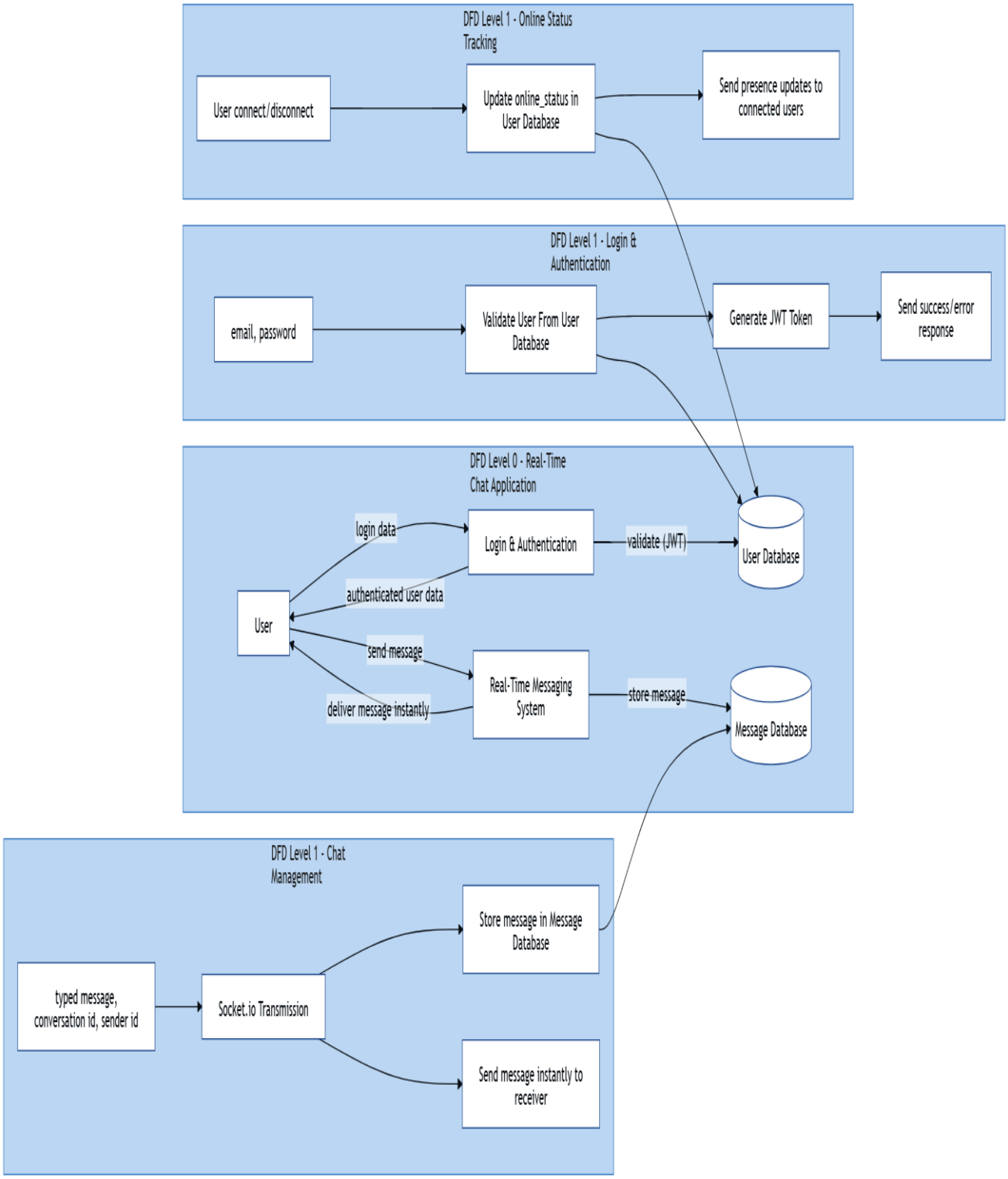
- One **User** can have many **Conversations**.
- One **Conversation** can have many **Messages**.
- Each message belongs to one sender and one receiver.

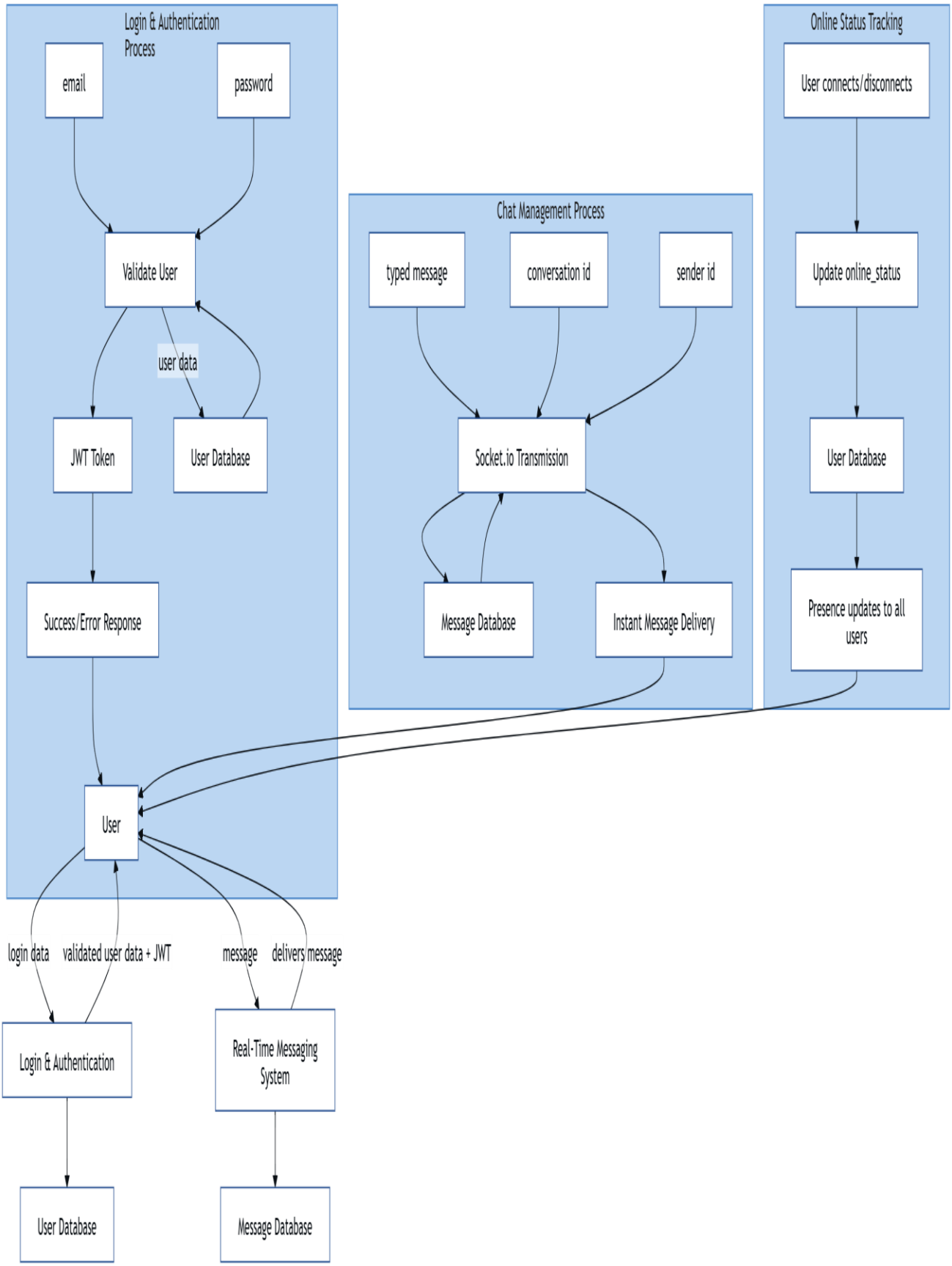

USER
string
user_id
PK
string
name
string
email
UK
string
password_hash
boolean
online_status
CONVERSATION
string
conversation_id
PK
string
participant1_id
FK
string
participant2_id
FK
datetime
created_at
MESSAGE
string
message_id
PK
string
conversation_id
FK
string
sender_id
FK
string
text
datetime
timestamp

5.4 Data Flow Diagram (DFD)

The Data Flow Diagram represents how data moves through the system from input to output.

DFD –




Login & Authentication Process
email
password
Validate User
user data
JWT Token
User Database
Success/Error Response
Chat Management Process
typed message
conversation id
sender id
Socket.io Transmission
Message Database
Instant Message Delivery
Online Status Tracking
User connects/disconnects
Update online_status
User Database
Presence updates to all users
User
login data
validated user data + JWT
message
delivers message
Login & Authentication
Real-Time Messaging System
User Database
Message Database

6. Complete Structure

Process Logical Diagram (Workflow)

User Login → JWT Verification → Enter Chat Dashboard

↓

Select User → Start Conversation → Real-Time Message Exchange

↓

Server (Socket.io) Broadcasts → Receiver Gets Message Instantly

System automatically performs:

- Real-time message syncing
- Online/offline status update
- Conversation management
- Secure token validation

7. Platform Used

Hardware Requirement

Component	**Minimum**
Processor	Intel Core i3
RAM	4 GB
Storage	2 GB
Operating System	Windows / Mac / Linux

Software Requirement

Component	**Technology**
Frontend	React.js, Tailwind CSS
Backend	Node.js, Express.js
Database	MongoDB
Real-Time Engine	Socket.io
Authentication	JWT
IDE	Visual Studio Code
Version Control	Git & GitHub

8. Future Scope

The platform can be expanded significantly:

- **Group Chat / Community Chat:** Support for creating public or private groups.
- **Media Sharing:** Send images, files, videos in chat.
- **End-to-End Encryption:** Improve privacy and security like WhatsApp.
- **Voice/Video Calling:** Integrate WebRTC for real-time communication.
- **Chatbot Integration:** AI-powered chatbot for customer support.
- **Mobile App:** React Native/Flutter app for Android and iOS.

These enhancements aim to make the platform more interactive, efficient, and data-driven, catering to evolving user needs and industry trends.

9. Bibliography

1. MongoDB Documentation – https://mongodb.com
2. React.js Documentation – https://react.dev
3. Express.js Documentation – https://expressjs.com
4. Node.js Documentation – https://nodejs.org
5. Socket.io Documentation – https://socket.io
6. JWT Authentication Docs – https://jwt.io
7. GitHub Guides – https://docs.github.com