COLLEGE CODE : 9623
COLLEGE NAME : Amrita College Of Engineering And Technology
DEPARTMENT : Computer Science and Engineering
STUDENT NM-ID : F58A45CD199F65582904B377E24880E9
ROLL NO : 962323104078
DATE : 11-09-2025
Completed the project named as Phase_02_ ChatApplicationUI
NAME : Chat Application UI
SUBMITTED BY,
NAME : Raffrin Narmadh V M
MOBILE NO : 7845411725

Chat Application UI

Objective

Build an intuitive, responsive, and scalable chat application interface.
Focus Areas: Tech Stack, UI Components, APIs, Data Handling, and Architecture.
Goal: Deliver a smooth, real-time communication experience for end-users.

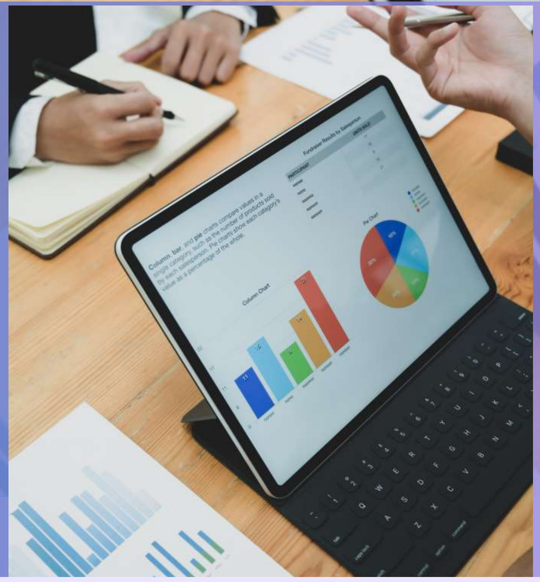
Tech Stack – Overview

- **Frontend:** React Native / Flutter → Cross-platform, responsive
  Backend: Node.js with WebSocket / Socket.IO → Real-time communication.

- **Database:** Firebase Firestore / MongoDB → Stores chats, users, media
  Authentication: Firebase Auth / JWT → Secure & simple login
  Hosting: Cloud platforms (AWS / Firebase Hosting / Render).

Why This Tech Stack?

1. React Native / Flutter → Fast UI, reusable components
2. Node.js + WebSocket → Handles real-time communication efficiently
3. Firebase / MongoDB → Flexible storage for messages, images, files
4. JWT / Firebase Auth → Secure login & session handling
5. Cloud Hosting → Scalable, always available



UI Structure

1. Login / Sign-Up
2. Chat List (Recent Conversations)
3. Individual Chat Screen Contacts / Search
4. Settings / Profile

Design Principles

Minimal, clean, and clutter-free designMobile-first responsive UIReal-time message updates with smooth animationsEasy navigation between conversationsAccessibility-friendly (font size, contrast, etc.)

API Schema Design

GET /chats → Fetch all chats of a user
POST /messages → Send a new message
GET /messages/:chatId → Fetch chat history
PUT /profile/:id → Update profile info
DELETE /messages/:id → Delete a message

Basic Flow

1. User opens app → Login/Register

2. App fetches recent chats → Displayed instantly.

3. User sends a message → Sent via WebSocket → Stored in DB

4. Recipient gets real-time notification → Message appears instantly

5. Chat history → Synced and stored for future access

Conclusion

1. Simple, scalable, and user-friendly chat interface

2. Real-time updates ensure engaging communication

3. Clear separation of concerns between UI, APIs and Database

4. Built for both performance and scalability

Thank You.