HACKATHON PHASE-I

COLLEGE CODE: 9528

COLLEGE NAME: SCAD College of Engineering and Technology

DEPARTMENT: Computer Science and Engineering

STUDENT NM-ID: 65CC4DB6536037BBAA9487A966AA7BE9

C90AAB851852669AF3C40AA5C0072C14

5BA78A6BB207CF6931040AE332BD179E

ROLL NO: 952823104179

952823104149

952823104163

DATE: 10.10.25

TECHNOLOGY: Front End/Node JS

PROJECT NAME : E-Commerce Product Page

SUBMITTED BY : YOGA KARTHIKA G

Name : SUGANYA M
SHALINI I

Mobile Number : 7823917752

Real-Time Chat with Firebase

1.Project Overview&Objectives

Problem Statement

In the digital communication era, users expect instant messaging systems that are fast, secure, and available anytime. Many chat apps require complex backend setup and maintenance.
This project, **Real-Time Chat with Firebase**, aims to build a **lightweight, responsive, and real-time communication platform** using Firebase services — eliminating the need for a traditional backend server.

Key Features

- **Instant Messaging:** Real-time message sending and receiving using Firebase Realtime Database / Firestore.
- **User Authentication:** Secure login and signup using Firebase Authentication (Email/Password or Google Auth).
- **Timestamps & Read Receipts:** Each message includes time and delivery status.
- **Online/Offline Status:** Display active users in real time.
- **Responsive UI:** Works seamlessly on both web and mobile view.
- **Notifications (Optional):** Real-time message alerts using Firebase Cloud Messaging (FCM).

Expected Outcomes

- A fully functional, real-time chat application.
- Instant synchronization of messages between multiple users.
- Secure user management with Firebase Authentication.
- A modern and interactive interface suitable for demo or deployment.

2. Technology Stack & Environment Setup

Backend

- **Firebase Realtime Database / Firestore** – For storing and retrieving chat messages instantly.
- **Firebase Authentication** – For secure user login and identity management.
- **Firebase Cloud Messaging (optional)** – For push notifications.

Frontend Framework

- **React.js** (or **Flutter** / **Android Studio** — depending on platform preference).
- **HTML5, CSS3, JavaScript (ES6)** – For responsive design and styling.

Database

- **Firebase Firestore (NoSQL)** – Real-time synchronization of messages, users, and chat rooms.

Tools

- **VS Code** – Code editor
- **Firebase Console** – Project setup and configuration
- **Git & GitHub** – Version control and team collaboration
- **Node.js (optional)** – For development environment and npm dependency management

Environment Setup

1. Install **Node.js** and **npm**.
2. Create a **Firebase project** and enable Authentication + Firestore.
3. Initialize Firebase in the project using firebase init.
4. Clone the Git repository and run npm install.
5. Start the app with npm start or deploy with Firebase Hosting.

3. API Design & Data Model

Planned Endpoints (Handled via Firebase SDK)

Function	**Endpoint / Collection**	**Description**
User Authentication	Firebase Auth	Handles login, signup, and logout
Messages	/messages	Stores and retrieves real-time chat messages
Users	/users	Stores user profiles and online status
Chat Rooms	/chats/{roomId}	Handles room-specific messages

Firebase replaces manual REST endpoints with SDK methods such as onSnapshot(), set(), and update() for real-time updates.

Request/Response Format (Sample)

Send Message (Firestore Write):

```
{
  "senderId": "user123",
 "receiverId": "user456",
"message": "Hey there!",
"timestamp": "2025-10-11T08:45:00Z"
}
```

Response (Auto-updated in client app):

```
{
  "status": "success",
 "message": "Message delivered successfully"
}
```

Database Schema (Firestore Example)

```
/users
userId
    name: "Arjun"
    email: "arjun@gmail.com"
    onlineStatus: true


/messages
messageId
    senderId: "u101"
    receiverId: "u102"
    text: "Hi da macha!"
    timestamp: 2025-10-11T10:25:00Z

    seen: false



/chatRooms

roomId

    users: ["u101", "u102"]

    lastMessage: "Hi da macha!"

    updatedAt: 2025-10-11T10:25:00Z
```

4. Front-End UI/UX Plan

Wireframes (Concept)

1. **Login / Register Page** – Simple authentication screen using Firebase Auth.
2. **Home / Chat List Page** – Displays all recent chats and online users.
3. **Chat Room Page** – Real-time conversation with message bubbles, timestamps, and input box.
4. **Profile / Settings Page** – User info and logout option.

Wireframe

A wireframe is a two-dimensional skeletal blueprint of a website or application. It's used to illustrate the basic structure, content, and functionality on a page.

The primary purpose is to focus on the layout and user flow without being distracted by visual design elements like colors, images, or typography.

They are often referred to as "low-fidelity" prototypes. They use simple shapes and lines (usually in grayscale) to represent elements such as buttons, images, and text areas.

Wireframes are a critical step in the design process. They help stakeholders and the design team agree on the application's information architecture and user experience (UX) before any visual design or development begins.

Navigation Flow

Login → Chat List → Chat Room → Send/Receive Messages

State Management Approach

- Using **React Hooks (useState, useEffect)** for local states.
- **Context API or Redux** for global app state (current user, chat data).
- Real-time data handled by **Firebase listeners (onSnapshot / onValue)**.

UI Design Focus

- Minimal and responsive layout. Chat bubble UI
- for sent/received messages. Real-time message
- scroll and smooth transitions.

5. Development & Deployment Plan

Team Roles

Role	**Responsibility**
Frontend Developer	UI design, Firebase integration, and state management
Backend / Cloud Engineer	Firebase configuration, database setup
Tester	App functionality, UI responsiveness, and bug tracking
Project Lead	Documentation, deployment, and code review

Git Workflow

- **Main Branch:** Stable version
- **Feature Branches:** For new features (e.g., auth, chat room, profile)
- **Pull Requests:** Code review before merging
- **Commit Convention:** Clear messages like feat: add message send feature

Testing Approach

- **Unit Testing:** Test Firebase functions and components individually.

- **Integration Testing:** Check message flow and user authentication end-to-end.
- **UI Testing:** Ensure responsiveness across screen sizes.

Hosting & Deployment Strategy

- **Firebase Hosting:** Deploy frontend and connect with Firestore backend.
- **GitHub Actions:** For continuous integration and deployment.
- **Backup:** Firestore data export for safety and version control.