Case Study

Chat App

Cross-Platform Real-Time Messaging Application

Mark Francis

Overview

The React Native Chat App is a cross-platform real-time messaging application built using React Native, Expo, and Firebase. The app supports text messaging, image sharing, location sharing, and offline caching.
The application runs on iOS, Android, and Web platforms through Expo and integrates Firebase Firestore and Firebase Storage for real-time communication and media handling.

https://www.creativemarkus.com/chat-app/

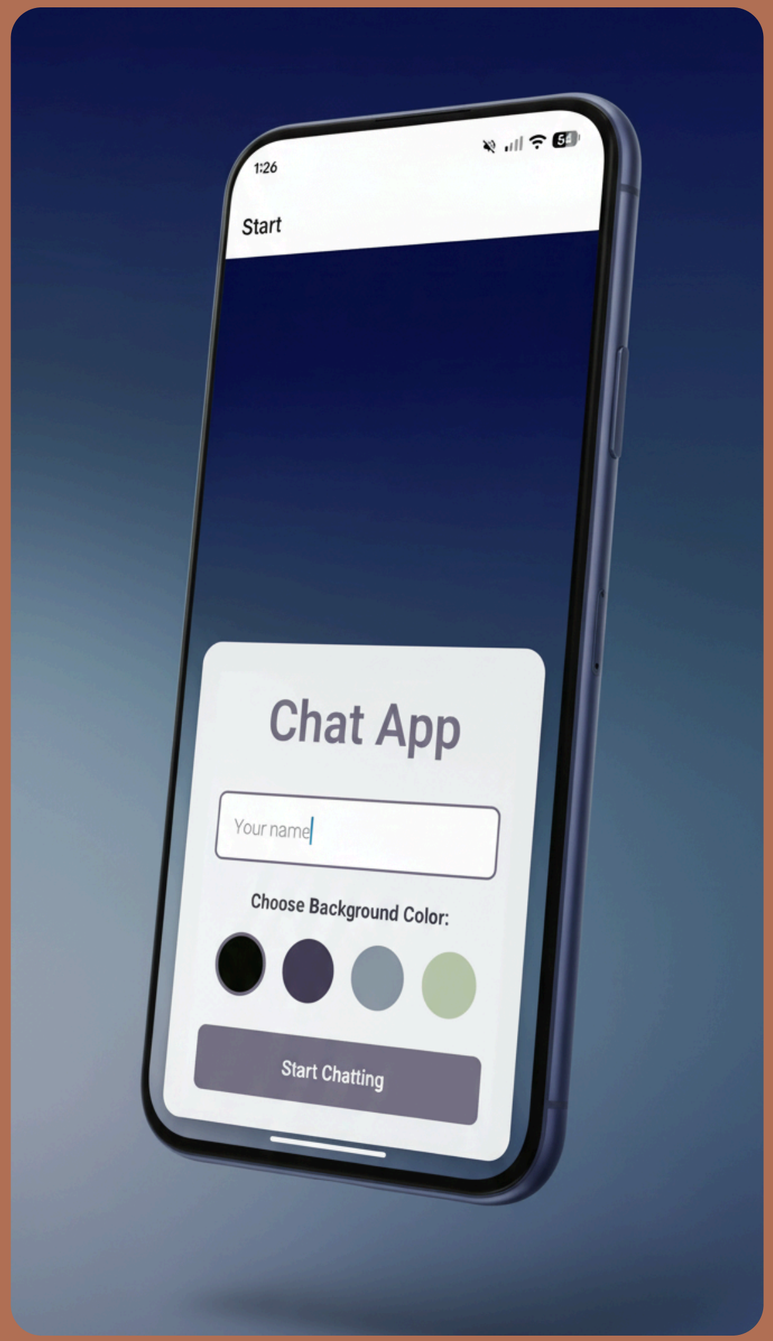
Purpose

The purpose of this project was to develop a feature-rich mobile application using React Native while implementing real-time communication, offline support, and native device capabilities.

Objective

The primary objective of this project was to:

- Build a cross-platform React Native application
- Integrate Firebase Firestore for real-time messaging
- Implement media upload using Firebase Storage
- Enable location sharing with interactive maps
- Provide offline caching using AsyncStorage
- Deploy using Expo Application Services



Duration & Credits

Role: Sole Developer

Responsibilities: UI design, Firebase integration, state management, testing, deployment

Methodologies & Tools

- React Native
- Expo
- Firebase Firestore
- Firebase Storage
- AsyncStorage
- React Native Maps
- Expo Location
- Expo Image Picker
- NetInfo (Network Monitoring)
- EAS (Expo Application Services)

Chat App

Server-Side Integration

The app integrates with Firebase services:

- **Firestore for real-time messaging**
- **Firebase Storage for image uploads**
- **Authentication handling**
- **Network state management**

Client-Side Architecture

Messages are stored in Firebase Firestore and synchronized instantly across connected devices.

Core Components

- Chat Screen
- Start Screen
- Custom Actions Component

Services

- Firebase configuration
- AsyncStorage caching
- Network monitoring

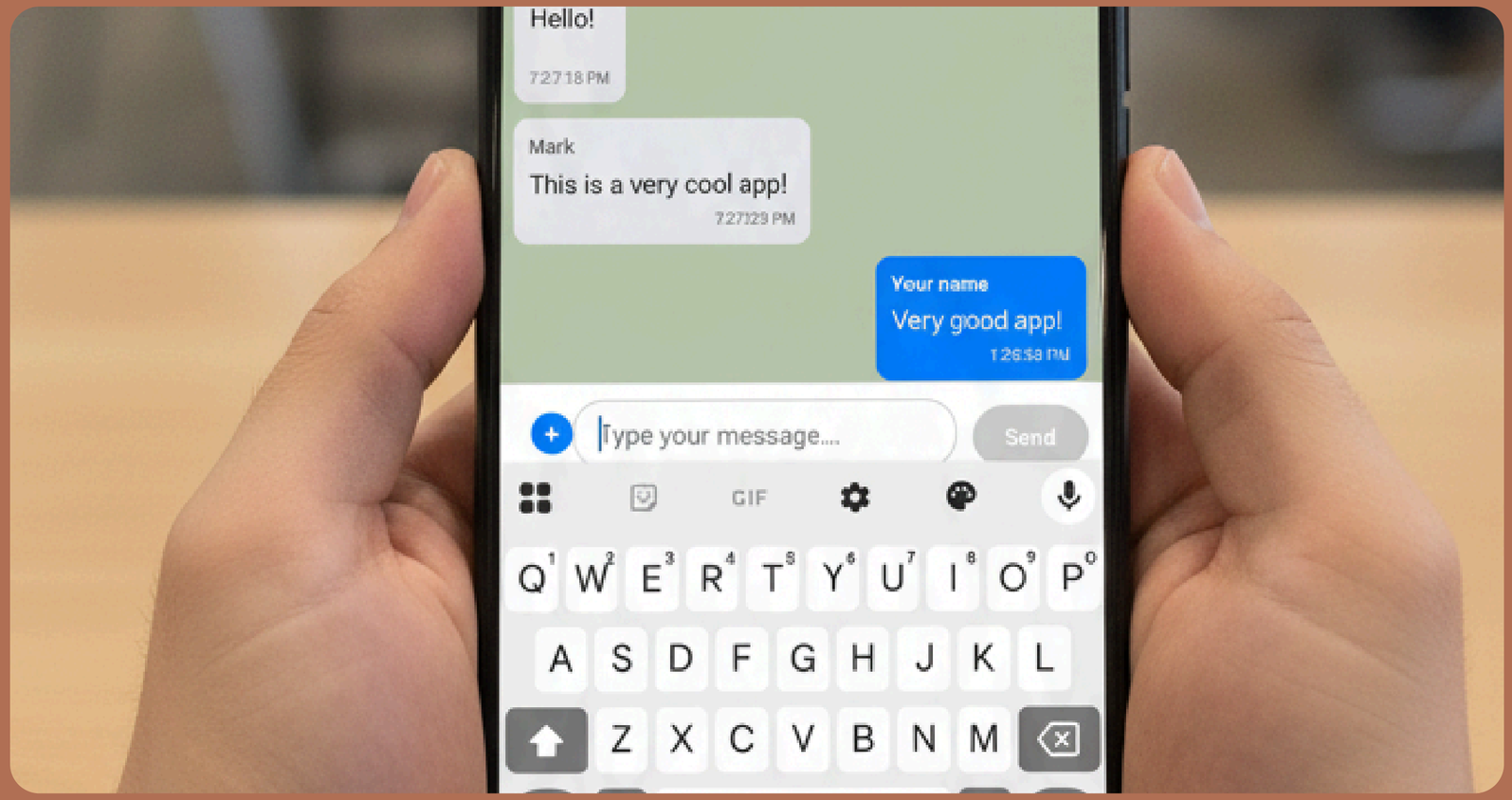
The UI was built using custom message bubbles rendered via FlatList for performance optimization.

Real-Time Messaging

Messages are stored in Firebase Firestore and synchronized instantly across connected devices.

Features include:

- Real-time message updates
- Custom styled message bubbles
- Sender distinction
- Timestamp support



Media & Location Sharing

Image Sharing:

- Camera capture
- Photo library selection
- Firebase Storage upload
- Persistent image URLs

Location Sharing:

- GPS coordinate retrieval
- Interactive MapView rendering
- Persistent location storage

Offline Support

The app detects network status changes and:

- **Caches messages locally**
- **Displays cached data when offline**
- **Syncs messages when connection is restored**

Retrospective

What Went Well

Real-time messaging integration with Firebase was highly effective. Cross-device synchronization worked seamlessly. Offline caching improved user reliability.

Challenges

Handling blob conversion for Firebase Storage required additional debugging. Managing network state transitions and ensuring smooth reconnection behavior required careful logic implementation.

Future Steps

- Add typing indicators
- Implement push notifications
- Add message reactions
- Improve UI animations
- Expand automated testing

Final Thoughts

This project strengthened my understanding of mobile-first development, cross-platform architecture, and real-time backend integration. Working with Firebase and Expo provided practical experience in building scalable, production-ready mobile applications.