ASIAN
COLLEGE OF ENGINEERING AND TECHNOLOGY

Approved by AICTE New Delhi and Affliated to Anna University, Chennai

Promoted and Run by a team of eminent Professors from Reputed Academic Institutions

Coimbatore - 641 110

Department of

Computer Science and Engineering

LABORATORY RECORD

Name : _____ **University Reg.No:** _____

Class :_____ **Branch/Sem :** _____

Certified bonafide Record of work done by _____ in _____ Laboratory.

Place* :** ***Staff Incharge ***Head of the department***

***Date* :**

Submitted for the University Practical Examination held on

_____

Internal Examiner **External Examiner**

Index

ChatConnect - A Real-Time Chat and Communication App

Abstract:

ChatConnect is a real-time chat and communication application designed for Android devices. The app leverages modern technologies such as Firebase for real-time data synchronization and Kotlin for robust and efficient code development. The primary objective of ChatConnect is to provide users with a seamless and intuitive messaging experience, enabling instant communication with friends, family, and colleagues.

Key features of ChatConnect include:

- Real-Time Messaging: Instant message delivery and receipt notifications.
- User Authentication: Secure login and registration using Firebase Authentication.

Introduction:

In today's fast-paced digital world, effective communication is paramount. ChatConnect is an innovative Android application designed to facilitate seamless and instant communication among users. Built with cutting-edge technologies, ChatConnect aims to provide a robust platform for real-time messaging, ensuring that users stay connected with their friends, family, and colleagues effortlessly.

The application leverages Firebase for real-time data synchronization, ensuring that messages are delivered instantly and reliably. With a focus on user experience, ChatConnect offers a clean and intuitive interface, making it easy for users to navigate and engage in conversations. Security is a top priority, and the app incorporates end-to-end encryption to protect user privacy and data integrity.

Key features of ChatConnect include real-time messaging, secure user authentication, customizable user profiles, group chat capabilities, media sharing, and push notifications. These features are designed to enhance user interaction and connectivity, providing a comprehensive communication solution.

ChatConnect is not just a messaging app; it is a platform that brings people closer, fostering meaningful connections and interactions in a digital age. Whether for personal or professional use, ChatConnect is the ideal tool for anyone looking to stay connected and communicate effectively.

Project Description:

ChatConnect is a sample project built using the Android Compose UI toolkit. It demonstrates how to create a simple chat app using the Compose libraries. The app allows users to send and receive text messages. The project showcases the use of Compose's declarative UI and state management capabilities. It also includes examples of how to handle input and navigation using composable functions and how to use data from a firebase to populate the UI.

Computer System Requirements:

Operating System: Microsoft Windows : 8/10/11(64-bit)

macOS:10.14 Mojave (or) newer

Linux: Any 64 bit Linux distribution that supports Gnome,KDE

Ram:8Gb (Minimum)

16Gb (Recommended)

CPU:x86_64 CPU architecture; 2nd generation Intel Core or newer (or) AMD CPU with support for a Windows Hypervisor Framework

Disk Type : SSD

Disk Space :8Gb (Minimum) 16Gb or more(Recommended)

Screen Resolution: 1280*800 (Minimum) 1920*1080 (Recommended)

Android System Requirements:

Operating System: Android-7 or later

Processor: Quadcore Processor (or)Higher

Ram:1Gb (Minimum) 4Gb (recommended)

Rom:50Mb (Minimum)

Internet connection

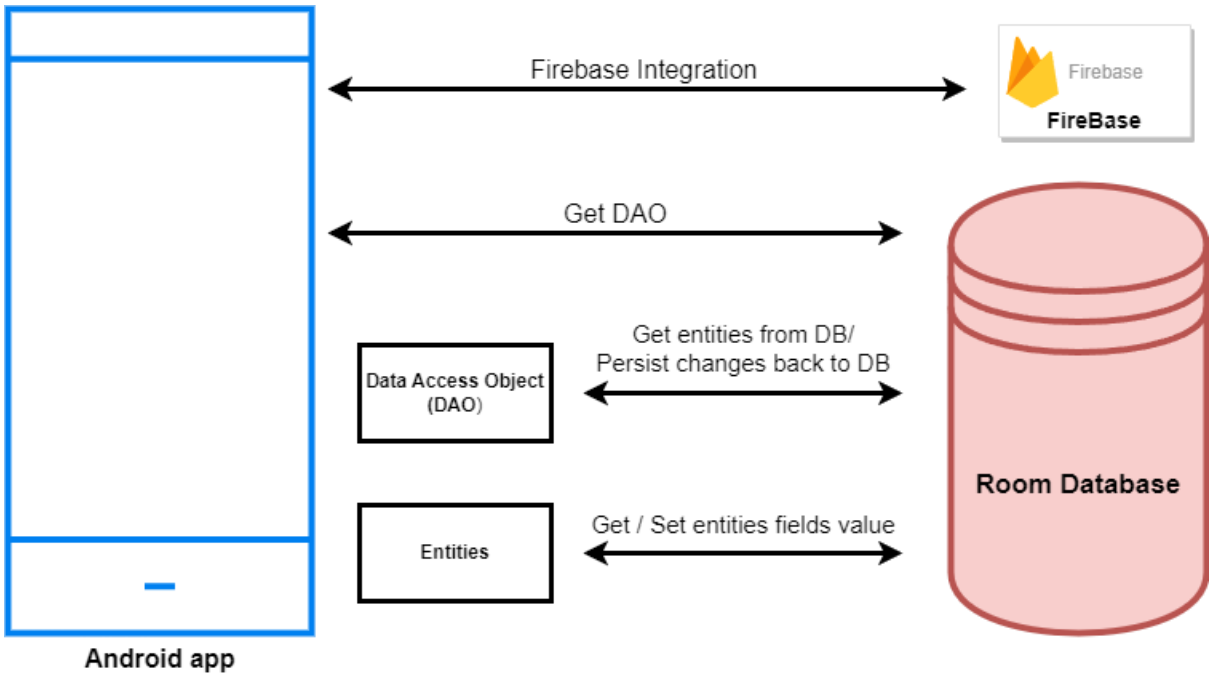
Tools Used:

Android Studio (Ladybug 2024.2.1)

Firebase (CLI v13.27.0)

Kotlin (2.0.0)

Architecture:



Tasks:

1.Required initial steps

2.Creating a new project

3.Integrating Firebase and Authentication

4.Creating UI files

5.Running the application.

Program:

MainActivity.kt

```
package com.project.pradyotprakash.flashchat

import android.os.Bundle
import androidx.activity.ComponentActivity import
androidx.activity.compose.setContent
import com.google.firebase.FirebaseApp

/**
 * The initial point of the application from where it gets started.
 *
 * Here we do all the initialization and other things which will be required  * thought
   out the application.
 */
class MainActivity : ComponentActivity() {     override
fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
FirebaseApp.initializeApp(this)         setContent {
            NavComposeApp()
        }
    } }
```

NavComposeApp.kt

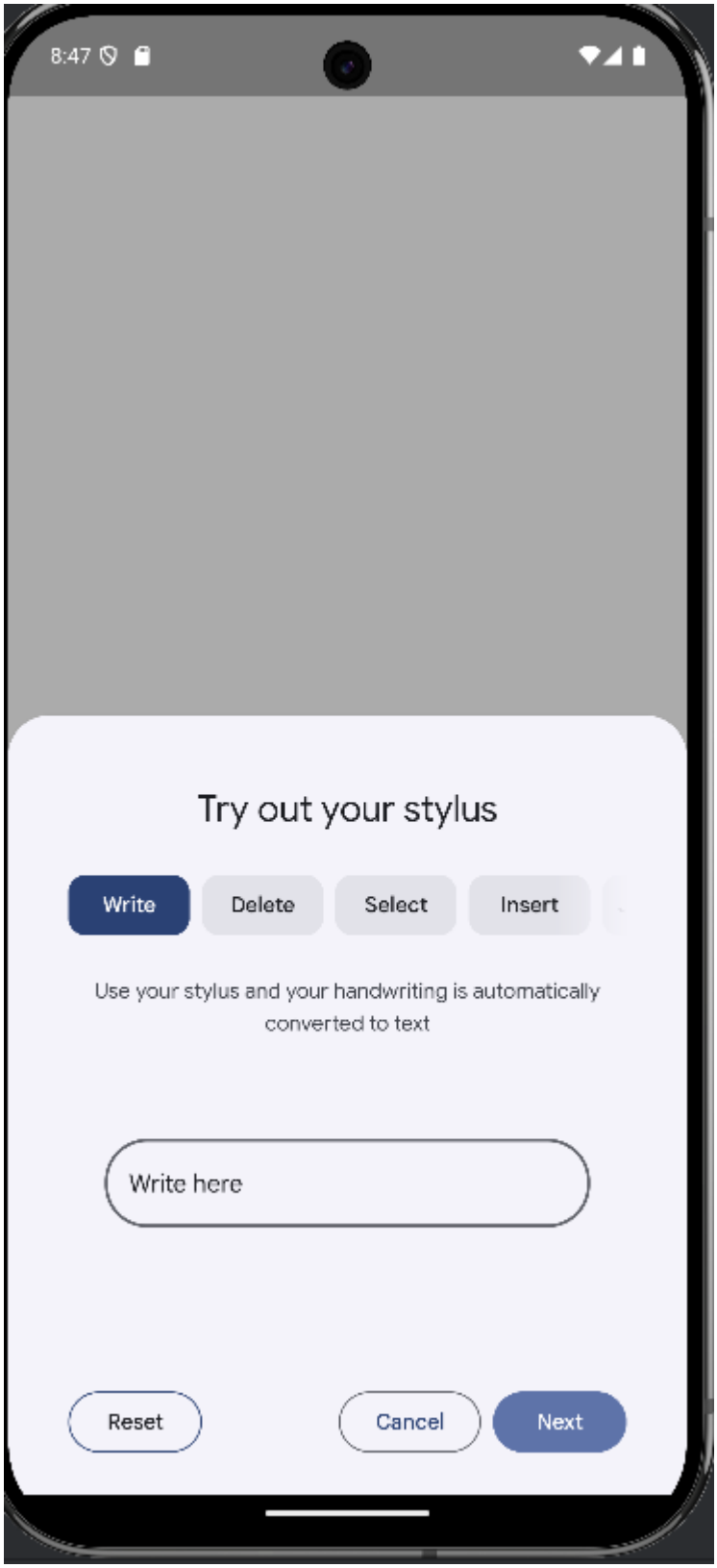
```
package com.project.pradyotprakash.flashchat

import androidx.compose.runtime.Composable import
androidx.compose.runtime.remember import
androidx.navigation.compose.NavHost import
androidx.navigation.compose.composable import
androidx.navigation.compose.rememberNavController import
com.google.firebase.auth.FirebaseAuth import
com.project.pradyotprakash.flashchat.nav.Action
import com.project.pradyotprakash.flashchat.nav.Destination.AuthenticationOption
import com.project.pradyotprakash.flashchat.nav.Destination.Home import
com.project.pradyotprakash.flashchat.nav.Destination.Login import
com.project.pradyotprakash.flashchat.nav.Destination.Register import
com.project.pradyotprakash.flashchat.ui.theme.FlashChatTheme import
com.project.pradyotprakash.flashchat.view.AuthenticationView import
com.project.pradyotprakash.flashchat.view.home.HomeView import
```

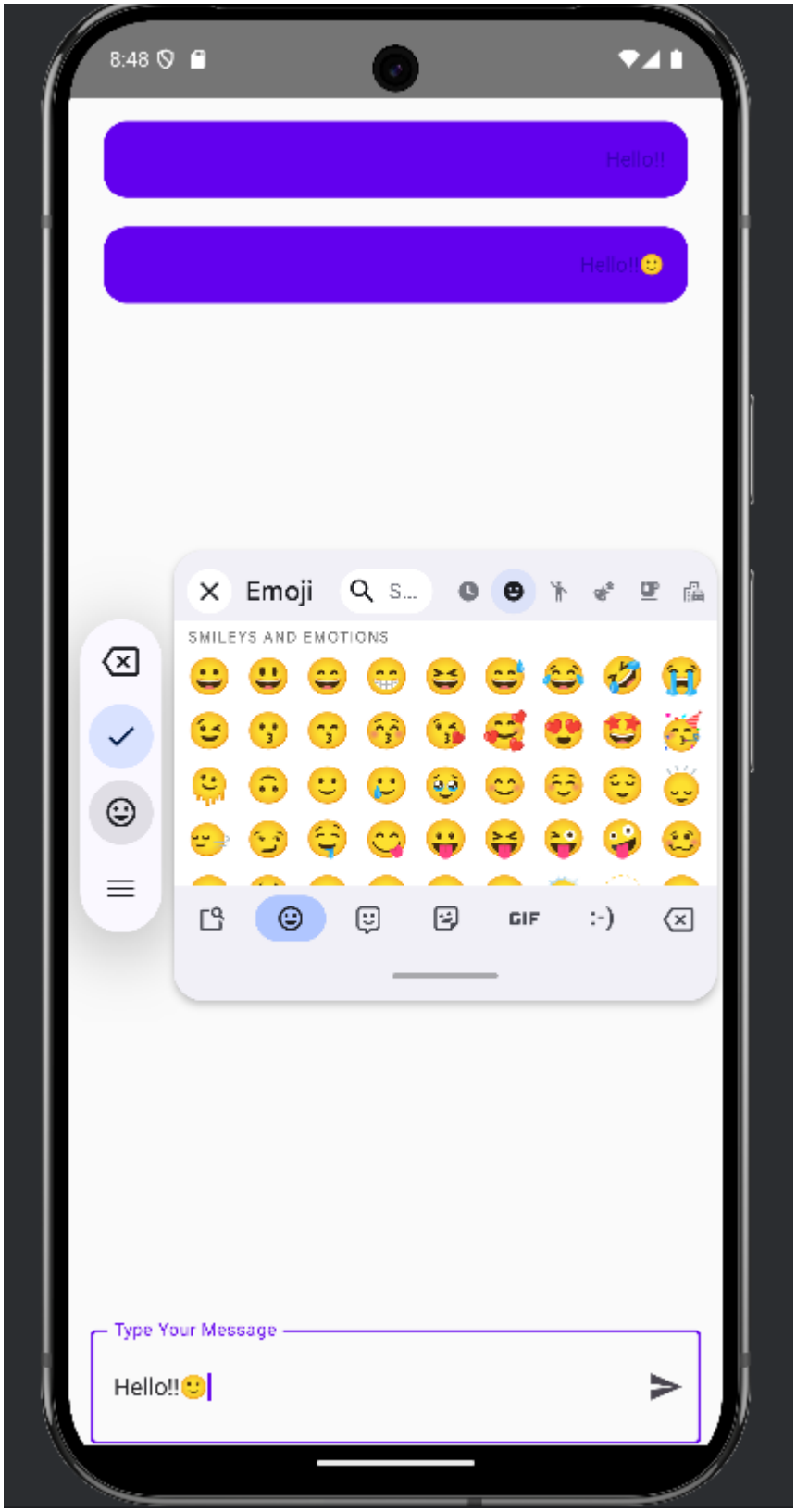
```
com.project.pradyotprakash.flashchat.view.login.LoginView import
com.project.pradyotprakash.flashchat.view.register.RegisterView

/**
 * The main Navigation composable which will handle all the navigation stack.
 */

@Composable fun
NavComposeApp() {
   val navController = rememberNavController()
   val actions = remember(navController) { Action(navController) }
   FlashChatTheme {  NavHost(
navController = navController,
startDestination =
      if (FirebaseAuth.getInstance().currentUser != null)
Home  else
        AuthenticationOption
    ) {
      composable(AuthenticationOption) {
AuthenticationView(  register =
actions.register,  login =
actions.login
        )
      }
      composable(Register) {
RegisterView(  home =
actions.home,  back =
actions.navigateBack
        )
      }
      composable(Login) {
LoginView(  home =
actions.home,  back =
actions.navigateBack
        )
      }
      composable(Home) {
        HomeView()
      }
    }
  }
```

}

Output:


8:46
Register
Email
abc@gmail.com
Password
Register


8:47
Try out your stylus
Write
Delete
Select
Insert
Use your stylus and your handwriting is automatically converted to text
Write here
Reset
Cancel
Next


8:48
Hello!!
Hello!!
Emoji
S...
SMILEYS AND EMOTIONS
GIF
:-)
Type Your Message
Hello!!

Conclusion:

The development of the Chat Connect App using Kotlin and Firebase has been a significant achievement. The app successfully integrates real-time messaging capabilities with a robust backend, ensuring a seamless user experience.

Key Achievements:

- Efficient Development: Leveraging Kotlin's concise syntax and Firebase's real-time database capabilities allowed for rapid development and deployment.
- Real-Time Communication: The app provides instant messaging features, ensuring users can communicate without delays.
- Scalability: Firebase's scalable infrastructure ensures the app can handle a growing number of users without compromising performance.
- Security: Implementing Firebase Authentication and Firestore security rules has ensured that user data is protected and privacy is maintained.

User Feedback and Future Enhancements: User feedback has been overwhelmingly positive, highlighting the app's ease of use and reliability. Future updates will focus on adding more features such as multimedia messaging, enhanced user profiles, and AI-driven chatbots to further enhance user engagement.

In conclusion, the Chat Connect App stands as a testament to the effective use of modern development tools and practices. It not only meets the initial project goals but also lays a strong foundation for future enhancements and scalability.

Future Scope:

1. **Multimedia Messaging**: Adding support for sending images, videos, voice messages, and documents can significantly enhance user engagement and make conversations more dynamic.
2. **AI-Driven Features**: Integrating AI for features like chatbots, smart replies, and sentiment analysis can improve user experience by providing quick responses and personalized interactions.
3. **Enhanced User Profiles**: Allowing users to customize their profiles with more details, status updates, and profile pictures can make the app more engaging and social.
4. **Group Chats and Channels**: Introducing group chat functionality and public/private channels can facilitate community building and allow users to connect over shared interests.
5. **End-to-End Encryption**: Implementing end-to-end encryption for all messages can enhance security and privacy, making the app more trustworthy for users.
6. **Offline Messaging**: Enabling offline messaging capabilities so users can send messages even without an internet connection, which will be delivered once they are back online.
7. **Localization and Internationalization**: Adding support for multiple languages and adapting the app for different regions can expand its user base globally.
8. **Integration with Other Services**: Integrating with other popular services and platforms (e.g., social media, cloud storage) can provide users with a more seamless and connected experience.
9. **Analytics and Insights**: Providing users with insights into their messaging habits and app usage can help them better understand their communication patterns.
10. **Monetization Strategies**: Exploring monetization options such as in-app purchases, premium features, and advertisements can help sustain the app financially.