**A Project Report On**

**An Android Application for keeping Up The Latest Headlines**

**BACHELOER OF COMPUTER SCIENCE**

**Devanger Arts college,**

**(Affiliated to Madurai Kamaraj University)**

**Aruppukottai-626101**

**Submitted**

**By**

**R.NANDHINI (Reg.No:C0S42003)**

**V.SUBALAKSHMI (Reg.No:C0S42004)**

**K.AJAYRAJ (Reg.No:C0S42007)**

**P.MUTHUPANDI (Reg.No:C0S42013)**

**1 INTRODUCTION**

**1.1 Overview**

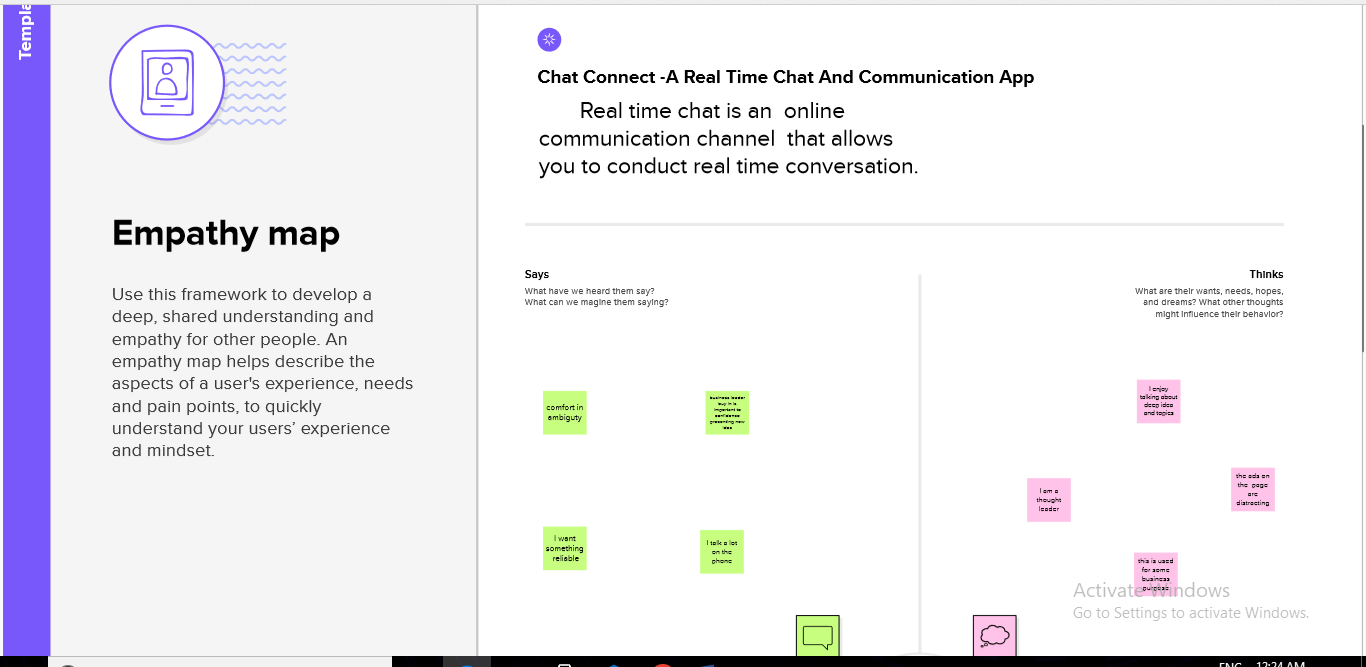
**Chat connect** allows you to interact with other ITIL staff in a productive way using a familiar chat tool similar to SMS on your smartphone. There are two ways to view Connect Chat, via the Sidebar or Workspace.

**1.2Purpose**

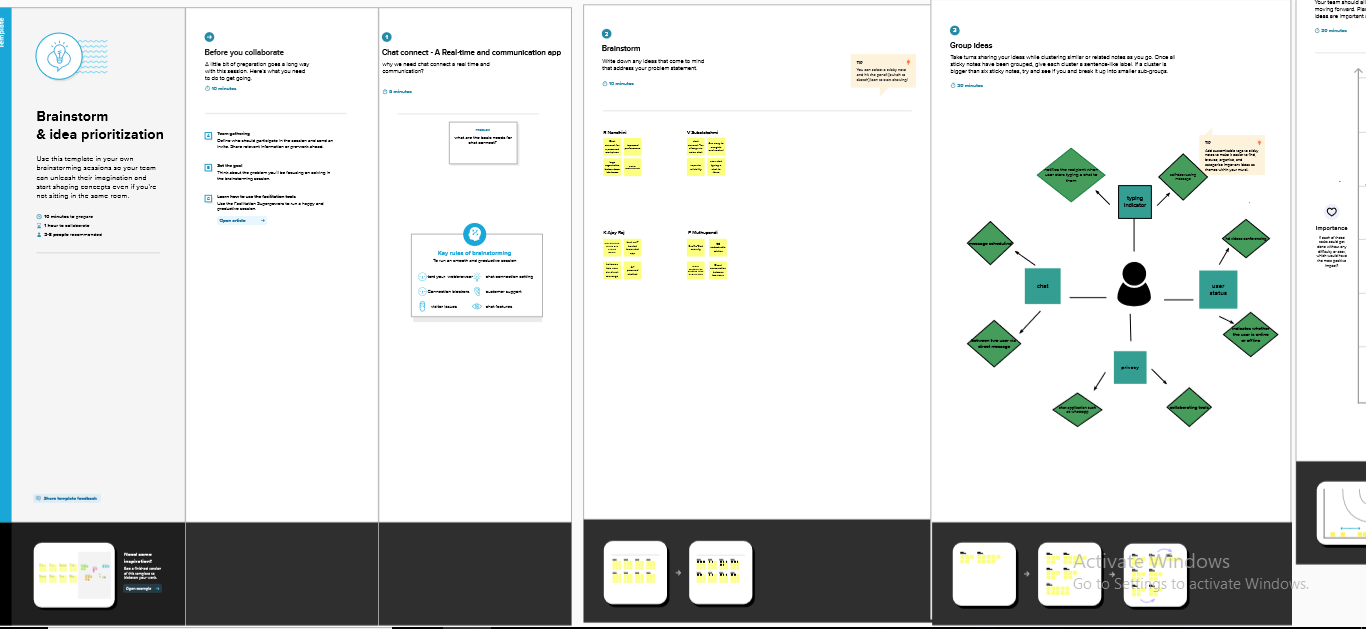
Our Mission. WhatsApp started as an alternative to SMS. Our product now supports sending and receiving a variety of media: text, photos, videos, documents, and location, as well as voice calls. Some of your most personal moments are shared with WhatsApp, which is why we built end-to-end encryption into our app.

**2 Problem Definition & Design Thinking**

**2.1 Empathy Map**

****

**2.2** **Ideation & Brainstorming Map**

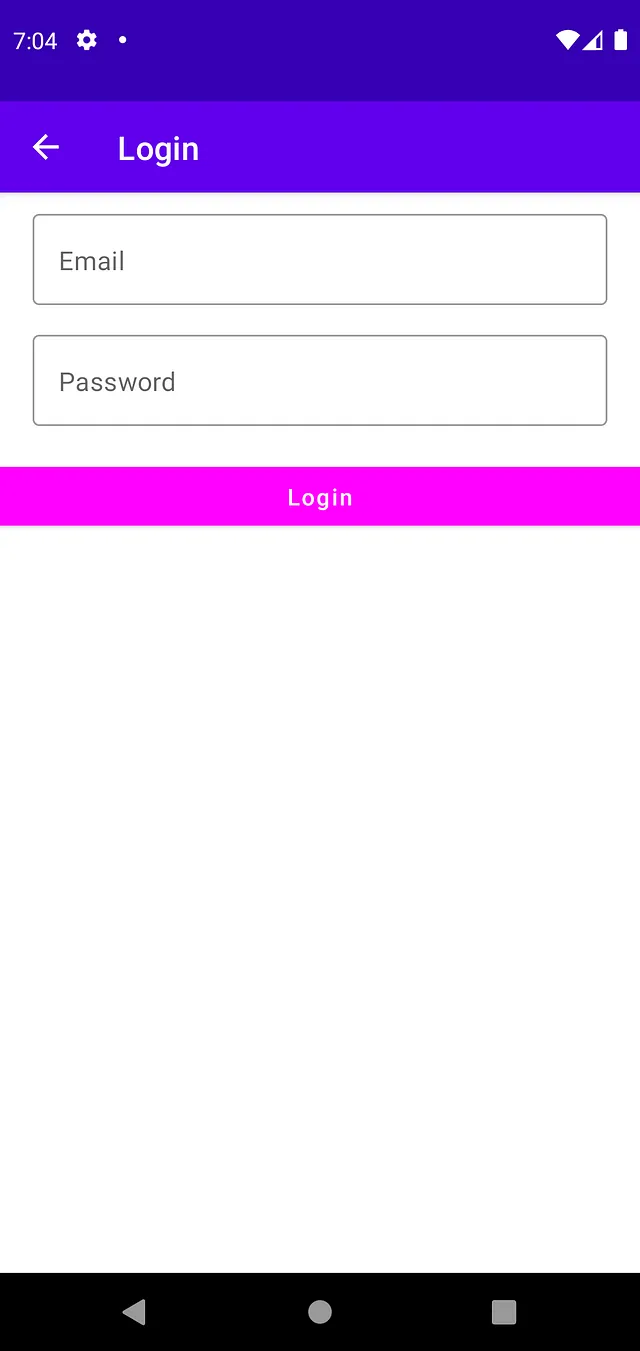
****

**3 RESULT**

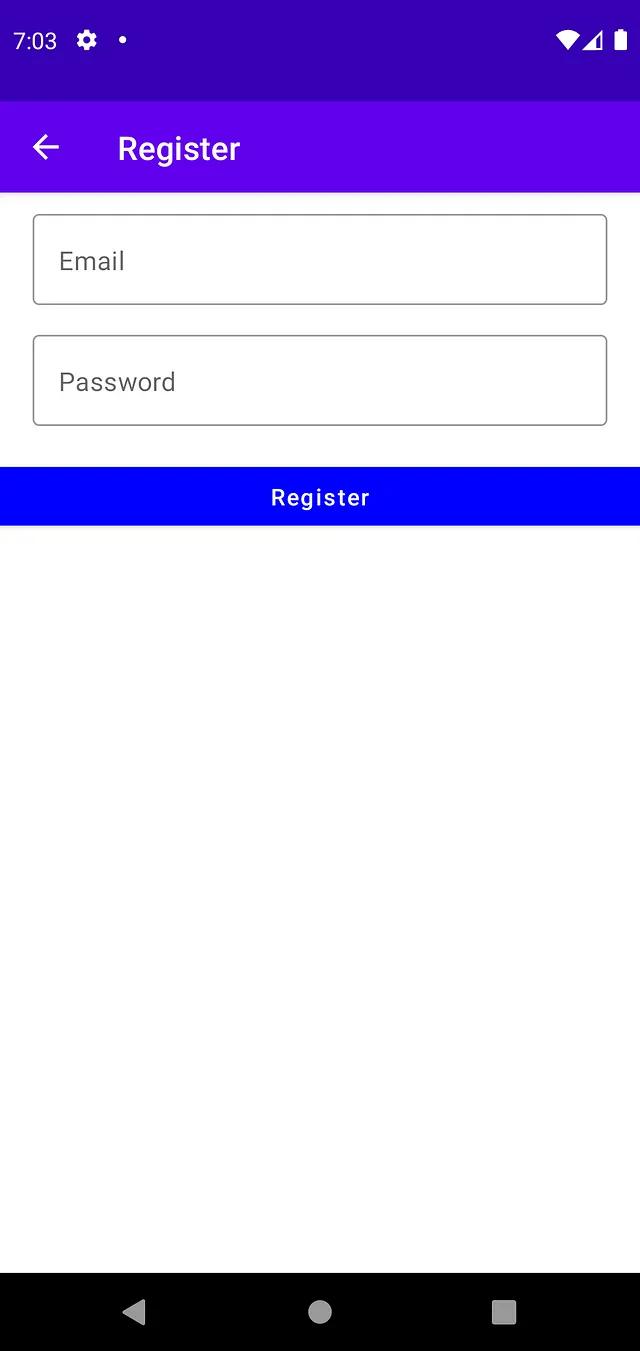
First Screen:



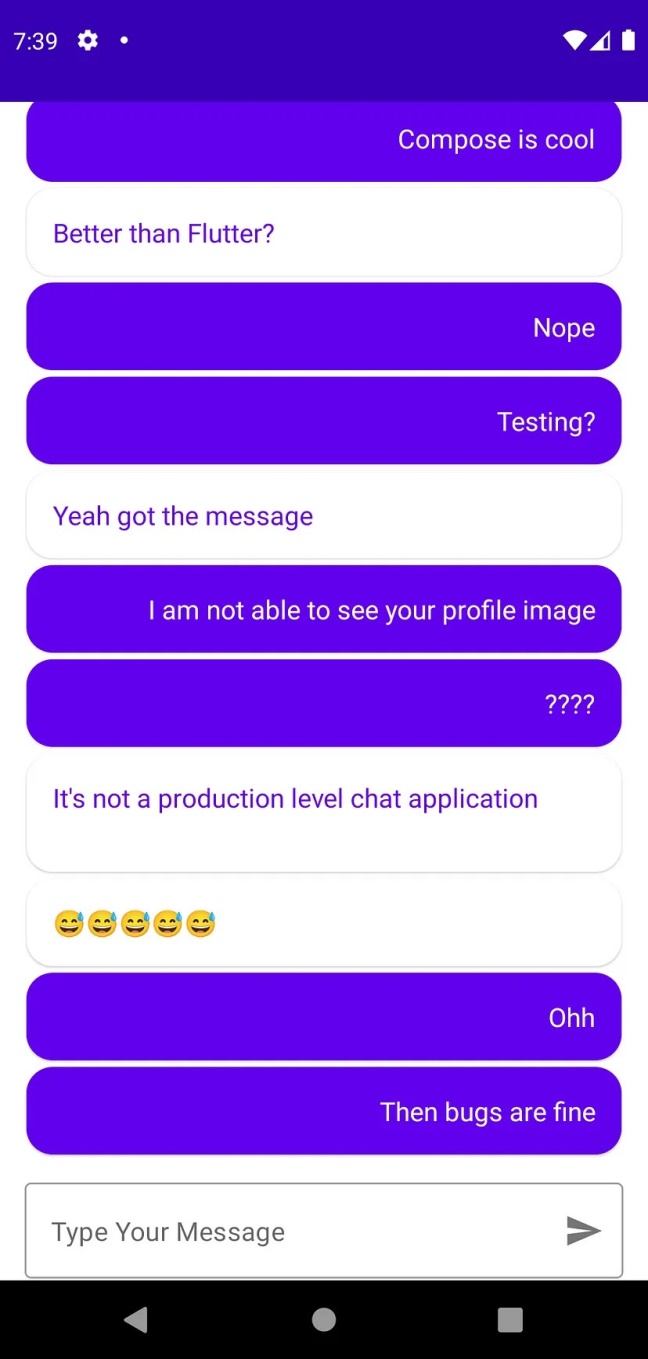
Login Page :



RegisterPage :



Home Screen:



**ADVANTAGES**

Sharing Information. The best platform for sharing and accessing information is chat connect.

Electronic Learning. Chat connect has recently evolved into a fictitious e-learning platform.

Protection and Confidentiality.

Facilitates Global Connection.

There Are No Ads.

Possibility for business.

**DISADVANTAGES**

A Means of Spreading False Information.

Leads to Issues with Mental Health.

Problems Associated with Health.

Sexual Stimulation.

No Content Censorship.

Financial Fraud.

**APPLICATIONS**

Team chat apps aren't just tools—they're the modern office. That's obviously true for remote teams, but the appeal goes well beyond that. Companies that have a shared physical workspace increasingly depend on workplace chat to get things done.

Use automation to support brainstorming, delegation, and communication

Learn how

Part of this is about efficiency. But business chat apps also mean that everyone can decide to participate in, or at least see, relevant conversations, while crucially also allowing people to ignore messages that aren't relevant to them. Slack is an excellent go-to choice, but there are plenty of Slack alternatives to choose from.

We reviewed and tested dozens of team communication apps, and here we'll present our picks for the best team chat apps currently on the market.

**CONCLUSION**

Overall we believe the chat reference service is a worthwhile endeavor. On the internal chat operator survey we asked for responses to "The chat service provides a much needed service and should definitely be continued." The results: Strongly Agree-2; Agree-8; Disagree-1; Strongly Disagree-0. Chat provides another reference option and is a good resource for remote users. Survey feedback suggests most users are satisfied with the service. There is some concern regarding the number of times technical difficulties hinder access. The Chat Reference Task Force will examine this issue more carefully over the summer. In consultation with all the librarians staffing chat reference, the Task Force also will explore various enhancement possibilities:

Extend the hours chat reference is available.

Include an "Ask A Librarian" link on all the Libraries’ web pages.

Upgrade or replace older computers to improve performance speed.

Cross training for librarians in other subject areas.

Evaluate available software using a more up-to-date set of criteria.

Whatever the outcome on these issues, the Libraries have moved from offering chat on a trial basis to making chat a regular feature of Carnegie Mellon's reference services.

**FUTURE SCOPE**

Chat connect is working on a bunch of features and it is expected to release them in the near future. The company is reportedly planning to add a new option for last seen, a new disappearing chats feature and a redesigned group info page. Chat connect could also soon allow you to send high-resolution videos or images

**APPENDIX**

1. Source Code

**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() |
|  | | } |
|  | | } |
|  | | } |
|  | | } |
|  |  | |

**Navigation.kt**

package com.project.pradyotprakash.flashchat.nav  
  
import androidx.navigation.NavHostController  
import com.project.pradyotprakash.flashchat.nav.Destination.Home  
import com.project.pradyotprakash.flashchat.nav.Destination.Login  
import com.project.pradyotprakash.flashchat.nav.Destination.Register  
  
/\*\*  
 \* A set of destination used in the whole application  
 \*/  
object Destination {  
    const val AuthenticationOption = "authenticationOption"  
    const val Register = "register"  
    const val Login = "login"  
    const val Home = "home"  
}  
  
/\*\*  
 \* Set of routes which will be passed to different composable so that  
 \* the routes which are required can be taken.  
 \*/  
class Action(navController: NavHostController) {  
    val home: () -> Unit = {  
        navController.navigate(Home) {  
            popUpTo(Login) {  
                inclusive = true  
            }  
            popUpTo(Register) {  
                inclusive = true  
            }  
        }  
    }  
    val login: () -> Unit = { navController.navigate(Login) }  
    val register: () -> Unit = { navController.navigate(Register) }  
    val navigateBack: () -> Unit = { navController.popBackStack() }  
}

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |