An Android Application For Keepling up the latest Headlines

Project Presented by

Team ID :NM2023TMID35013

Team Size :04

Team Leader : K.Kaviya

Team Member :Gayathri. R

Team Member: Parvathi.P

Team Member: Nishalini.K

An Android Application For Keepling up the latest Headlines

In this tutorial, you will understand how to build a basic public news app using Android Studio.

What is a News App?

This app is a news application. This application will show the news about the world. This is better than old conventional newspapers as you can see news anytime and anywhere now. This will also have live updates. This app will have some other features like adding your own news as an editor and people can see your published news too.

Flow of the Application

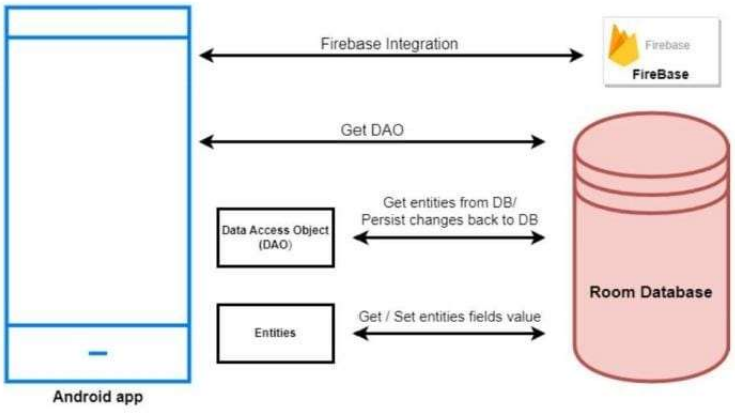
There is a login signup screen. On the dashboard of the app, there will be 3 buttons. There will be a button for publishing your own news, for seeing the general news, and for seeing the news published in the app. In general news, this will fetch news from API and will show it in the app. This app will show the news category-wise. There are a lot of categories.

- The users can publish their own news or articles in the app itself very easily.
- He can also add photos to their articles to make them look more attractive.
- He can also see the articles of other people.

Features of the Public News app

1. There is the main screen in this app.
2. Login and sign up are required in the app.
3. Users can see the real news with real-time updates.
4. The app will show category-wise news.
5. Auto update news because it fetch news from API
6. Users can see the news according to their interests.
7. Users can publish their own articles.
8. Users can add images to their articles.
9. Users can see the articles published by other users.

Architecture:-



Project Prerequisites

The project requires the following technologies/tools.

- **Java** – Java is used for writing the coding part.
- **XML** – XML is used for designing parts and design the screens
- **Firebase** – Firebase will be used for storing the published articles and publishing the news

Developing the Android Public News app

Let's look at the files that we created for the public news app before we implement it. Let us see them one by one-

1. **Activity_main**.xml is the dashboard file. This screen will contain three buttons.

2. There is a **MainActivity**.java to write the logic part.

3. We have to create different XML files for all three different screens and also their corresponding files for the writing of the logic part.

4. The following files are also used :

- **Colors.xml** : This file is used for defining the colors of our app.

An Android Application for Keeping Up with the Latest Headlines

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@drawable/news_app_icon"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.NewsHeadlines"
        tools:targetApi="31">
        <activity
            android:name=".DisplayNews"
            android:exported="false"
            android:label="@string/title_activity_display_news"
            android:theme="@style/Theme.NewsHeadlines" />
        <activity
            android:name=".RegistrationActivity"
            android:exported="false"
            android:label="@string/title_activity_registration"
            android:theme="@style/Theme.NewsHeadlines" />
        <activity
            android:name=".MainPage"
            android:exported="false"
            android:label="@string/title_activity_main_page"
            android:theme="@style/Theme.NewsHeadlines" />
        <activity
            android:name=".LoginActivity"
            android:exported="true"
            android:label="@string/app_name"
            android:theme="@style/Theme.NewsHeadlines">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Color.kt

```
package com.example.newsheadlines.ui.theme

import androidx.compose.ui.graphics.Color

val Purple200 = Color(0xFFBB86FC)
val Purple500 = Color(0xFF6200EE)
val Purple700 = Color(0xFF3700B3)
val Teal200 = Color(0xFF03DAC5)
```

Shape.kt

```
package com.example.newsheadlines.ui.theme

import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.Shapes
import androidx.compose.ui.unit.dp

val Shapes = Shapes(
    small = RoundedCornerShape(4.dp),
    medium = RoundedCornerShape(4.dp),
    large = RoundedCornerShape(0.dp)
)
```

Theme.kt

```
package com.example.newsheadlines.ui.theme

import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.material.MaterialTheme
import androidx.compose.material.darkColors
import androidx.compose.material.lightColors
import androidx.compose.runtime.Composable

private val DarkColorPalette = darkColors(
    primary = Purple200,
    primaryVariant = Purple700,
    secondary = Teal200
)

private val LightColorPalette = lightColors(
    primary = Purple500,
    primaryVariant = Purple700,
    secondary = Teal200

    /* Other default colors to override
    background = Color.White,
    surface = Color.White,
    onPrimary = Color.White,
    onSecondary = Color.Black,
    onBackground = Color.Black,
    onSurface = Color.Black,
    */
)
```

```
@Composable
fun NewsHeadlinesTheme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    content: @Composable () -> Unit
) {
    val colors = if (darkTheme) {
        DarkColorPalette
    } else {
        LightColorPalette
    }

    MaterialTheme(
        colors = colors,
        typography = Typography,
        shapes = Shapes,
        content = content
    )
}
```

Type.kt

```
package com.example.newsheadlines.ui.theme

import androidx.compose.material.Typography
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontFamily
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.sp

// Set of Material typography styles to start with
val Typography = Typography(
    body1 = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 16.sp
    )
    /* Other default text styles to override
    button = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.W500,
        fontSize = 14.sp
    ),
    caption = TextStyle(
        fontFamily = FontFamily.Default,
        fontWeight = FontWeight.Normal,
        fontSize = 12.sp
    )
    */
)
```

ApiService.kt

```
package com.example.newsheadlines

import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory
import retrofit2.http.GET
```

```
interface ApiService {

    //@GET("movielist.json")
    @GET("top-
headlines?country=us&category=business&apiKey=684cb893caf7425abeffad82ac1d0
f4e")
    ///@GET("search?q=chatgpt")
    suspend fun getMovies() :News

    companion object {
        var apiService: ApiService? = null
        fun getInstance() : ApiService {
            if (apiService == null) {
                apiService = Retrofit.Builder()
                    // .baseUrl("https://howtodoandroid.com/apis/")
                    .baseUrl("https://newsapi.org/v2/")
                    //.baseUrl("https://podcast-episodes.p.rapidapi.com/")

                    .addConverterFactory(GsonConverterFactory.create())
                    .build().create(ApiService::class.java)
            }
            return apiService!!
        }
    }

}
```

Articles.kt

```
package com.example.example

import com.google.gson.annotations.SerializedName


data class Articles (

  @SerializedName("title"       ) var title       : String? = null,
  @SerializedName("description" ) var description : String? = null,
  @SerializedName("urlToImage"  ) var urlToImage  : String? = null,

  )
```

DisplayNews.kt

```
package com.example.newsheadlines

import android.content.Intent
import android.os.Bundle
import android.util.Log
import android.widget.TextView
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
import androidx.compose.material.Text
```

```
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.compose.ui.viewinterop.AndroidView
import androidx.core.text.HtmlCompat
import coil.compose.rememberImagePainter
import com.example.newsheadlines.ui.theme.NewsHeadlinesTheme

class DisplayNews : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            NewsHeadlinesTheme {
                // A surface container using the 'background' color from
the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {

                    var desk = getIntent().getStringExtra("desk")
                    var title = getIntent().getStringExtra("title")
                    var uriImage = getIntent().getStringExtra("urlToImage")
                    Log.i("test123abc", "MovieItem: $desk")

                    Column(Modifier.background(Color.Gray).padding(20.dp),
horizontalAlignment = Alignment.CenterHorizontally, verticalArrangement =
Arrangement.Center) {
                        Text(text = ""+title, fontSize = 32.sp)
                        HtmlText(html = desk.toString())
                        /*  AsyncImage(
                              model = "https://example.com/image.jpg",
                              contentDescription = "Translated description
of what the image contains"
                            )*/
                        Image(
                            painter = rememberImagePainter(uriImage),
                            contentDescription = "My content description",
                        )
                    }
                    //   Greeting(desk.toString())
                }
            }
        }
    }
}

@Composable
fun Greeting(name: String) {
    // Text(text = "Hello $name!")
}

@Preview(showBackground = true)
@Composable
fun DefaultPreview() {
    NewsHeadlinesTheme {
        //   Greeting("Android")
```

```
    }
}
@Composable
fun HtmlText(html: String, modifier: Modifier = Modifier) {
    AndroidView(
        modifier = modifier,
        factory = { context -> TextView(context) },
        update = { it.text = HtmlCompat.fromHtml(html,
HtmlCompat.FROM_HTML_MODE_COMPACT) }
    )
}
```

LoginActivity.kt

```
package com.example.newsheadlines

import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.*
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Lock
import androidx.compose.material.icons.filled.Person
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import androidx.core.content.ContextCompat.startActivity
import com.example.newsheadlines.ui.theme.NewsHeadlinesTheme

class LoginActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {

            LoginScreen(this, databaseHelper)
        }
    }
}
@Composable
fun LoginScreen(context: Context, databaseHelper: UserDatabaseHelper) {
    var username by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var error by remember { mutableStateOf("") }
```

```
    Column(
        Modifier
            .fillMaxHeight()
            .fillMaxWidth()
            .padding(28.dp),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center)

    {
        Image(
            painter = painterResource(id = R.drawable.news),
            contentDescription = "")

        Spacer(modifier = Modifier.height(10.dp))


        Row {
            Divider(color = Color.LightGray, thickness = 2.dp, modifier =
Modifier
                .width(155.dp)
                .padding(top = 20.dp, end = 20.dp))
            Text(text = "Login",
                color = Color(0xFF6495ED),
                fontWeight = FontWeight.Bold,
                fontSize = 24.sp,style = MaterialTheme.typography.h1)
            Divider(color = Color.LightGray, thickness = 2.dp, modifier =
Modifier
                .width(155.dp)
                .padding(top = 20.dp, start = 20.dp))

        }

        Spacer(modifier = Modifier.height(10.dp))

        TextField(
            value = username,
            onValueChange = { username = it },
            leadingIcon = {
                Icon(
                    imageVector = Icons.Default.Person,
                    contentDescription = "personIcon",
                    tint = Color(0xFF6495ED)
                )
            },
            placeholder = {
                Text(
                    text = "username",
                    color = Color.Black
                )
            },
            colors = TextFieldDefaults.textFieldColors(
                backgroundColor = Color.Transparent
            )

        )


        Spacer(modifier = Modifier.height(20.dp))

        TextField(
```

```
            value = password,
            onValueChange = { password = it },
            leadingIcon = {
                Icon(
                    imageVector = Icons.Default.Lock,
                    contentDescription = "lockIcon",
                    tint = Color(0xFF6495ED)
                )
            },
            placeholder = { Text(text = "password", color = Color.Black) },
            visualTransformation = PasswordVisualTransformation(),
            colors = TextFieldDefaults.textFieldColors(backgroundColor =
Color.Transparent)
        )



        Spacer(modifier = Modifier.height(12.dp))
        if (error.isNotEmpty()) {
            Text(
                text = error,
                color = MaterialTheme.colors.error,
                modifier = Modifier.padding(vertical = 16.dp)
            )
        }

        Button(
            onClick = {
                if (username.isNotEmpty() && password.isNotEmpty()) {
                    val user = databaseHelper.getUserByUsername(username)
                    if (user != null && user.password == password) {
                        error = "Successfully log in"
                        context.startActivity(
                            Intent(
                                context,
                                MainPage::class.java
                            )
                        )
                        //onLoginSuccess()
                    } else {
                        error = "Invalid username or password"
                    }
                } else {
                    error = "Please fill all fields"
                }
            },
            shape = RoundedCornerShape(20.dp),
            colors = ButtonDefaults.buttonColors(backgroundColor =
Color(0xFF77a2ef)),
            modifier = Modifier.width(200.dp)
            .padding(top = 16.dp)
        ) {
            Text(text = "Log In", fontWeight = FontWeight.Bold)
        }

        Row(modifier = Modifier.fillMaxWidth()) {
            TextButton(onClick = {
                context.startActivity(
                    Intent(
                        context,
                        RegistrationActivity::class.java
```

```
                    ))})
                { Text(text = "Sign up",
                    color = Color.Black
                )}

                Spacer(modifier = Modifier.width(100.dp))

                TextButton(onClick = { /* Do something! */ })
                { Text(text = "Forgot password ?",
                    color = Color.Black
                )}
            }



    }
}
private fun startMainPage(context: Context) {
    val intent = Intent(context, MainPage::class.java)
    ContextCompat.startActivity(context, intent, null)
}
```

MainPage.kt

```
package com.example.newsheadlines

import android.content.Context
import android.content.Intent
import android.content.Intent.FLAG_ACTIVITY_NEW_TASK
import android.os.Bundle
import android.util.Log
import android.widget.TextView
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.viewModels
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.itemsIndexed
import androidx.compose.foundation.selection.selectable
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.Card
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
import androidx.compose.material.Text
import androidx.compose.runtime.*
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.compose.ui.viewinterop.AndroidView
import androidx.core.text.HtmlCompat
import coil.compose.rememberImagePainter
import coil.size.Scale
```

```
import coil.transform.CircleCropTransformation
import com.example.example.Articles
import com.example.newsheadlines.ui.theme.NewsHeadlinesTheme

class MainPage : ComponentActivity() {
    val mainViewModel by viewModels<MainViewModel>()
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            NewsHeadlinesTheme {
                // A surface container using the 'background' color from
the theme
                Surface(color = MaterialTheme.colors.background) {
                    Column() {


                        Text(text = "Latest NEWS", fontSize = 32.sp,
modifier = Modifier.fillMaxWidth(), textAlign = TextAlign.Center)

                        MovieList(applicationContext, movieList =
mainViewModel.movieListResponse)
                        mainViewModel.getMovieList()
                    }
                }
            }
        }
    }
}

@Composable
fun MovieList(context: Context, movieList: List<Articles>) {
    var selectedIndex by remember { mutableStateOf(-1) }
    LazyColumn {

        itemsIndexed(items = movieList) {
                index, item ->
            MovieItem(context,movie = item, index, selectedIndex) { i ->
                selectedIndex = i
            }
        }
    }

}

@Composable
fun MovieItem(context: Context) {
    val movie = Articles(
        "Coco",
        "",
        " articl"
    )


    MovieItem(context,movie = movie, 0, 0) { i ->
        Log.i("wertytest123abc", "MovieItem: "
                +i)
    }
}

@Composable
fun MovieItem(context: Context, movie: Articles, index: Int, selectedIndex:
```

```
Int,
                onClick: (Int) -> Unit)
{

    val backgroundColor = if (index == selectedIndex)
MaterialTheme.colors.primary else MaterialTheme.colors.background

    Card(
        modifier = Modifier
            .padding(8.dp, 4.dp)
            .fillMaxSize()
            .selectable(true, true, null,
                onClick = {
                    Log.i("test123abc", "MovieItem:
$index/n$selectedIndex")
                })
            .clickable { onClick(index) }
            .height(180.dp), shape = RoundedCornerShape(8.dp), elevation =
4.dp
    ) {
        Surface(color = Color.White) {

            Row(
                Modifier
                    .padding(4.dp)
                    .fillMaxSize()

            )
            {
                Image(
                    painter = rememberImagePainter(
                        data = movie.urlToImage,
                        builder = {
                            scale(Scale.FILL)
                            placeholder(R.drawable.placeholder)
                            transformations(CircleCropTransformation())
                        }
                    ),
                    contentDescription = movie.description,
                    modifier = Modifier
                        .fillMaxHeight()
                        .weight(0.3f)
                )


                Column(
                    verticalArrangement = Arrangement.Center,
                    modifier = Modifier
                        .padding(4.dp)
                        .fillMaxHeight()
                        .weight(0.8f)
                        .background(Color.Gray)
                        .padding(20.dp)
                        .selectable(true, true, null,
                            onClick = {
                                Log.i("test123abc", "MovieItem:
$index/n${movie.description}")
                                context.startActivity(
                                    Intent(context,
DisplayNews::class.java)
```

```
.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK)
                                        .putExtra("desk",
movie.description.toString())
                                        .putExtra("urlToImage",
movie.urlToImage)
                                        .putExtra("title", movie.title)
                                    )
                                })
                    ) {

                        Text(
                            text = movie.title.toString(),
                            style = MaterialTheme.typography.subtitle1,
                            fontWeight = FontWeight.Bold
                        )

                        HtmlText(html = movie.description.toString())
                    }
                }
            }
        }
        @Composable
        fun HtmlText(html: String, modifier: Modifier = Modifier) {
            AndroidView(
                modifier = modifier
                    .fillMaxSize()
                    .size(33.dp),
                factory = { context -> TextView(context) },
                update = { it.text = HtmlCompat.fromHtml(html,
HtmlCompat.FROM_HTML_MODE_COMPACT) }
            )
        }
}
```

MainViewModel.kt

```
package com.example.newsheadlines

import android.util.Log
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.setValue
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.example.Articles
import kotlinx.coroutines.launch

class MainViewModel : ViewModel() {
    var movieListResponse:List<Articles> by mutableStateOf(listOf())
    var errorMessage: String by mutableStateOf("")
    fun getMovieList() {
        viewModelScope.launch {
            val apiService = ApiService.getInstance()
            try {
                val movieList = apiService.getMovies()
                movieListResponse = movieList.articles
            }
            catch (e: Exception) {
```

```
                errorMessage = e.message.toString()
            }
        }
    }
}
```

Model.kt

```
package com.example.newsheadlines


data class Movie(val name: String,
                 val imageUrl: String,
                 val desc: String,
                 val category: String)
```

News.kt

```
package com.example.newsheadlines

import com.example.example.Articles
import com.google.gson.annotations.SerializedName


data class News (
  @SerializedName("status") var status:String?= null,
  @SerializedName("totalResults") var totalResults : Int?                =
null,
  @SerializedName("articles") var articles     : ArrayList<Articles> =
arrayListOf()
)
```

RegistrationActivity.kt

```
package com.example.newsheadlines

import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.*
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Email
import androidx.compose.material.icons.filled.Lock
import androidx.compose.material.icons.filled.Person
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.PasswordVisualTransformation
```

```
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
import com.example.newsheadlines.ui.theme.NewsHeadlinesTheme

class RegistrationActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper(this)
        setContent {


                    RegistrationScreen(this,databaseHelper)
                }
            }
        }



@Composable
fun RegistrationScreen(context: Context, databaseHelper:
UserDatabaseHelper) {
    var username by remember { mutableStateOf("") }
    var password by remember { mutableStateOf("") }
    var email by remember { mutableStateOf("") }
    var error by remember { mutableStateOf("") }

    Column(
        Modifier
            .background(Color.White)
            .fillMaxHeight()
            .fillMaxWidth(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center)

    {
        Row {
            Text(
                text = "Sign Up",
                color = Color(0xFF6495ED),
                fontWeight = FontWeight.Bold,
                fontSize = 24.sp, style = MaterialTheme.typography.h1
            )
            Divider(
                color = Color.LightGray, thickness = 2.dp, modifier =
Modifier
                    .width(250.dp)
                    .padding(top = 20.dp, start = 10.dp, end = 70.dp)
            )

        }

        Image(
            painter = painterResource(id = R.drawable.sign_up),
            contentDescription = "",
            modifier = Modifier.height(270.dp)
        )

        TextField(
```

```
            value = username,
            onValueChange = { username = it },
            leadingIcon = {
                Icon(
                    imageVector = Icons.Default.Person,
                    contentDescription = "personIcon",
                    tint = Color(0xFF6495ED)
                )
            },
            placeholder = {
                Text(
                    text = "username",
                    color = Color.Black
                )
            },
            colors = TextFieldDefaults.textFieldColors(
                backgroundColor = Color.Transparent
            )

        )

        Spacer(modifier = Modifier.height(8.dp))

        TextField(
            value = password,
            onValueChange = { password = it },
            leadingIcon = {
                Icon(
                    imageVector = Icons.Default.Lock,
                    contentDescription = "lockIcon",
                    tint = Color(0xFF6495ED)
                )
            },
            placeholder = { Text(text = "password", color = Color.Black) },
            visualTransformation = PasswordVisualTransformation(),
            colors = TextFieldDefaults.textFieldColors(backgroundColor =
Color.Transparent)
        )


        Spacer(modifier = Modifier.height(16.dp))


        TextField(
            value = email,
            onValueChange = { email = it },
            leadingIcon = {
                Icon(
                    imageVector = Icons.Default.Email,
                    contentDescription = "emailIcon",
                    tint = Color(0xFF6495ED)
                )
            },
            placeholder = { Text(text = "email", color = Color.Black) },
            colors = TextFieldDefaults.textFieldColors(backgroundColor =
Color.Transparent)
        )

        Spacer(modifier = Modifier.height(8.dp))

        if (error.isNotEmpty()) {
```

```
            Text(
                text = error,
                color = MaterialTheme.colors.error,
                modifier = Modifier.padding(vertical = 16.dp)
            )
        }

        Button(
            onClick = {
                if (username.isNotEmpty() && password.isNotEmpty() &&
email.isNotEmpty()) {
                    val user = User(
                        id = null,
                        firstName = username,
                        lastName = null,
                        email = email,
                        password = password
                    )
                    databaseHelper.insertUser(user)
                    error = "User registered successfully"
                    // Start LoginActivity using the current context
                    context.startActivity(
                        Intent(
                            context,
                            LoginActivity::class.java
                        )
                    )

                } else {
                    error = "Please fill all fields"
                }
            },
            shape = RoundedCornerShape(20.dp),
            colors = ButtonDefaults.buttonColors(backgroundColor =
Color(0xFF77a2ef)),
            modifier = Modifier.width(200.dp)
                .padding(top = 16.dp)
        ) {
            Text(text = "Register", fontWeight = FontWeight.Bold)
        }

        Row(
            modifier = Modifier.padding(30.dp),
            verticalAlignment = Alignment.CenterVertically,
            horizontalArrangement = Arrangement.Center
        ) {


            Text(text = "Have an account?")

            TextButton(onClick = {
                context.startActivity(
                    Intent(
                        context,
                        LoginActivity::class.java
                    )
                )
            }) {
                Text(text = "Log in",
                    fontWeight = FontWeight.Bold,
                    style = MaterialTheme.typography.subtitle1,
```

```
                    color = Color(0xFF4285F4)
                )}

            }
        }
    }
    private fun startLoginActivity(context: Context) {
        val intent = Intent(context, LoginActivity::class.java)
        ContextCompat.startActivity(context, intent, null)
    }
```

Source.kt

```
package com.example.example

import com.google.gson.annotations.SerializedName


data class Source (

  @SerializedName("id"   ) var id   : String? = null,
  @SerializedName("name" ) var name : String? = null

)
```

User.kt

```
package com.example.newsheadlines

import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "user_table")
data class User(
    @PrimaryKey(autoGenerate = true) val id: Int?,
    @ColumnInfo(name = "first_name") val firstName: String?,
    @ColumnInfo(name = "last_name") val lastName: String?,
    @ColumnInfo(name = "email") val email: String?,
    @ColumnInfo(name = "password") val password: String?,

    )
```

UserDao.kt

```
package com.example.newsheadlines

import androidx.room.*

@Dao
interface UserDao {
```

```
    @Query("SELECT * FROM user_table WHERE email = :email")
    suspend fun getUserByEmail(email: String): User?

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertUser(user: User)

    @Update
    suspend fun updateUser(user: User)

    @Delete
    suspend fun deleteUser(user: User)
}
```

UserDatabase.kt

```
package com.example.newsheadlines

import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase

@Database(entities = [User::class], version = 1)
abstract class UserDatabase : RoomDatabase() {

    abstract fun userDao(): UserDao

    companion object {

        @Volatile
        private var instance: UserDatabase? = null

        fun getDatabase(context: Context): UserDatabase {
            return instance ?: synchronized(this) {
                val newInstance = Room.databaseBuilder(
                    context.applicationContext,
                    UserDatabase::class.java,
                    "user_database"
                ).build()
                instance = newInstance
                newInstance
            }
        }
    }
}
```

UserDatabaseHelper.kt

```
package com.example.newsheadlines

import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
```

```
class UserDatabaseHelper(context: Context) :
    SQLiteOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION) {

    companion object {
        private const val DATABASE_VERSION = 1
        private const val DATABASE_NAME = "UserDatabase.db"

        private const val TABLE_NAME = "user_table"
        private const val COLUMN_ID = "id"
        private const val COLUMN_FIRST_NAME = "first_name"
        private const val COLUMN_LAST_NAME = "last_name"
        private const val COLUMN_EMAIL = "email"
        private const val COLUMN_PASSWORD = "password"
    }

    override fun onCreate(db: SQLiteDatabase?) {
        val createTable = "CREATE TABLE $TABLE_NAME (" +
                "$COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "$COLUMN_FIRST_NAME TEXT, " +
                "$COLUMN_LAST_NAME TEXT, " +
                "$COLUMN_EMAIL TEXT, " +
                "$COLUMN_PASSWORD TEXT" +
                ")"

        db?.execSQL(createTable)
    }

    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int,
newVersion: Int) {
        db?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db)
    }

    fun insertUser(user: User) {
        val db = writableDatabase
        val values = ContentValues()
        values.put(COLUMN_FIRST_NAME, user.firstName)
        values.put(COLUMN_LAST_NAME, user.lastName)
        values.put(COLUMN_EMAIL, user.email)
        values.put(COLUMN_PASSWORD, user.password)
        db.insert(TABLE_NAME, null, values)
        db.close()
    }

    @SuppressLint("Range")
    fun getUserByUsername(username: String): User? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE
$COLUMN_FIRST_NAME = ?", arrayOf(username))
        var user: User? = null
        if (cursor.moveToFirst()) {
            user = User(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
```

```
            )
        }
        cursor.close()
        db.close()
        return user
    }
    @SuppressLint("Range")
    fun getUserById(id: Int): User? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE
$COLUMN_ID = ?", arrayOf(id.toString()))
        var user: User? = null
        if (cursor.moveToFirst()) {
            user = User(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
            )
        }
        cursor.close()
        db.close()
        return user
    }

    @SuppressLint("Range")
    fun getAllUsers(): List<User> {
        val users = mutableListOf<User>()
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME", null)
        if (cursor.moveToFirst()) {
            do {
                val user = User(
                    id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
                    firstName =
cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
                    lastName =
cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
                    email =
cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
                    password =
cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)),
                )
                users.add(user)
            } while (cursor.moveToNext())
        }
        cursor.close()
        db.close()
        return users
    }

}
```

ExampleInstrumentedTest.kt

```kotlin
package com.example.newsheadlines

import androidx.test.platform.app.InstrumentationRegistry
import androidx.test.ext.junit.runners.AndroidJUnit4

import org.junit.Test
import org.junit.runner.RunWith

import org.junit.Assert.*

/**
 * Instrumented test, which will execute on an Android device.
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
@RunWith(AndroidJUnit4::class)
class ExampleInstrumentedTest {
    @Test
    fun useAppContext() {
        // Context of the app under test.
        val appContext =
InstrumentationRegistry.getInstrumentation().targetContext
        assertEquals("com.example.newsheadlines", appContext.packageName)
    }
}
```

ExampleUnitTest.kt

```kotlin
package com.example.newsheadlines

import org.junit.Test

import org.junit.Assert.*

/**
 * Example local unit test, which will execute on the development machine
(host).
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
class ExampleUnitTest {
    @Test
    fun addition_isCorrect() {
        assertEquals(4, 2 + 2)
    }
}
```

ScreenShot

Four Army personnel killed in firing incident inside Bathinda Military Station

A statement by HQ South Western Command of the Army said that the area was cordoned off and sealed and search operations are in progress.

How pep talk with Hardik Pandya changed Axar Patel's batting minset