ANDROID APPLICATION DEVELOPMENT

An Android Application Development For Keeping The latest Headlines

TEAM ID : NM2024TMID05378

Submitted By

Rohan Mohana Devi T(Team Leader)	**-**	**0A2B110E6AD1A489C1B48F29BBB1CB49**
Preethi S(Team Member)	**-**	**FAAD1B9F7E9567E8D554F8590D035EB0**
Rakshana Fathima J(Team Member)	**-**	**75BE4B315634C2A93A44EF6F744BDF23**
Kavya A(Team Member)	**-**	**EA2CFDF7596E8F3B9FE41D794BF2D7A1**

SEMESTER - V

B.E COMPUTER SCIENCE AND ENGINEERING

ACADEMIC YEAR - 2024 - 2025

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ANNA UNIVERSITY REGIONAL CAMPUS COIMBATORE

COIMBATORE – 641046

NOVEMBER 2024

TABLE OF CONTENT:

ABSTRACT:

Android application that uses Jetpack Compose, a contemporary UI toolkit for creating native Android user interfaces, to display news articles in a scrollable list style. It features an intuitive user interface. Because each article entry has a title, an image, and a synopsis, readers can browse and choose content that interests them with ease. Users can see all of the details on a new screen when they tap an article. Through the Retrofit library, the application dynamically retrieves data from a distant server, guaranteeing effective and instantaneous data access. The Coil library, which is renowned for its lightweight, quick-loading characteristics tailored for Android, loads images with ease. This application demonstrates how to use Coil for image handling in conjunction with Jetpack Compose's declarative UI paradigm.

OBJECTIVES:

The primary objective of this application is to:

1. **Demonstrate Modern Android Development Practices**: By using **Jetpack Compose** for UI design, the application highlights the benefits of declarative programming in creating intuitive and adaptable user interfaces.

2. **Enable Dynamic and Efficient Data Handling**: Leveraging **Retrofit**, the app dynamically fetches news articles from a remote server, ensuring up-to-date content delivery and optimal performance.

3. **Showcase Advanced Image Loading Techniques**: Through the integration of **Coil**, the application illustrates how to manage image rendering efficiently within a Compose-based UI.

4. **Provide a Seamless User Experience**: The intuitive design allows users to browse, select, and explore news articles effortlessly, making the application highly user-centric.

This project serves as a foundation for developers to explore modern Android development paradigms while building scalable and visually engaging applications.

PROPOSED SOLUTION:

The project Android news reader app developed in Android Studio using Jetpack Compose, Retrofit, and Coil to provide users with a streamlined way to browse and read news articles. The app's main feature is a list of articles, each displaying a title, image, and short description. Users can tap on an article to view its full content. Built on the MVVM architecture, the project maintains a clean separation between data, UI, and business logic, ensuring a maintainable codebase. Retrofit handles network requests to fetch articles from a remote server, while Coil enables efficient asynchronous loading and display of images. Jetpack Compose facilitates the creation of a modern and reactive UI, with composable functions defining each part of the screen in a structured way. The core data model includes a

NewsArticle class, with a repository responsible for data retrieval and a ViewModel that manages UI state and data flow using StateFlow for reactive updates. Android Studio's development tools simplify the building, testing, and refining processes for this Compose-based UI. Planned enhancements include features like article search, offline capabilities, and push notifications for new articles. This project highlights the effective use of Android Studio with Jetpack Compose, Retrofit, and Coil to deliver a dynamic and modular Android application.

REQUIRED TOOLS:

SOFTWARE REQUIRMENTS:

1. Android Studio
2. Android SDK
3. Java Development Kit
4. Libraries : Retrofit, Gson, Coil, Koltin Coroutines.

HARDWARE REQUIEMENTS:

1. CPU: intel i5.
2. RAM: Minimum 10 GB free disk space.
3. Android Iphone (or) Physical Android device.
4. Network: Stable internet connection.

TOOLS AND VERSIONS:

1. Android-studio-2024.2.1.11-windows.exe
2. JDK version 11
3. Android SDK: level 21
4. Gradle version 8.0
5. Android Emulator : latest version
6. Kotlin 1.6.0
7. Jetpack compose version 1.2.0

TESTING:

Username: It should be in the form of an email address. This means it needs to include the special character and a valid domain.

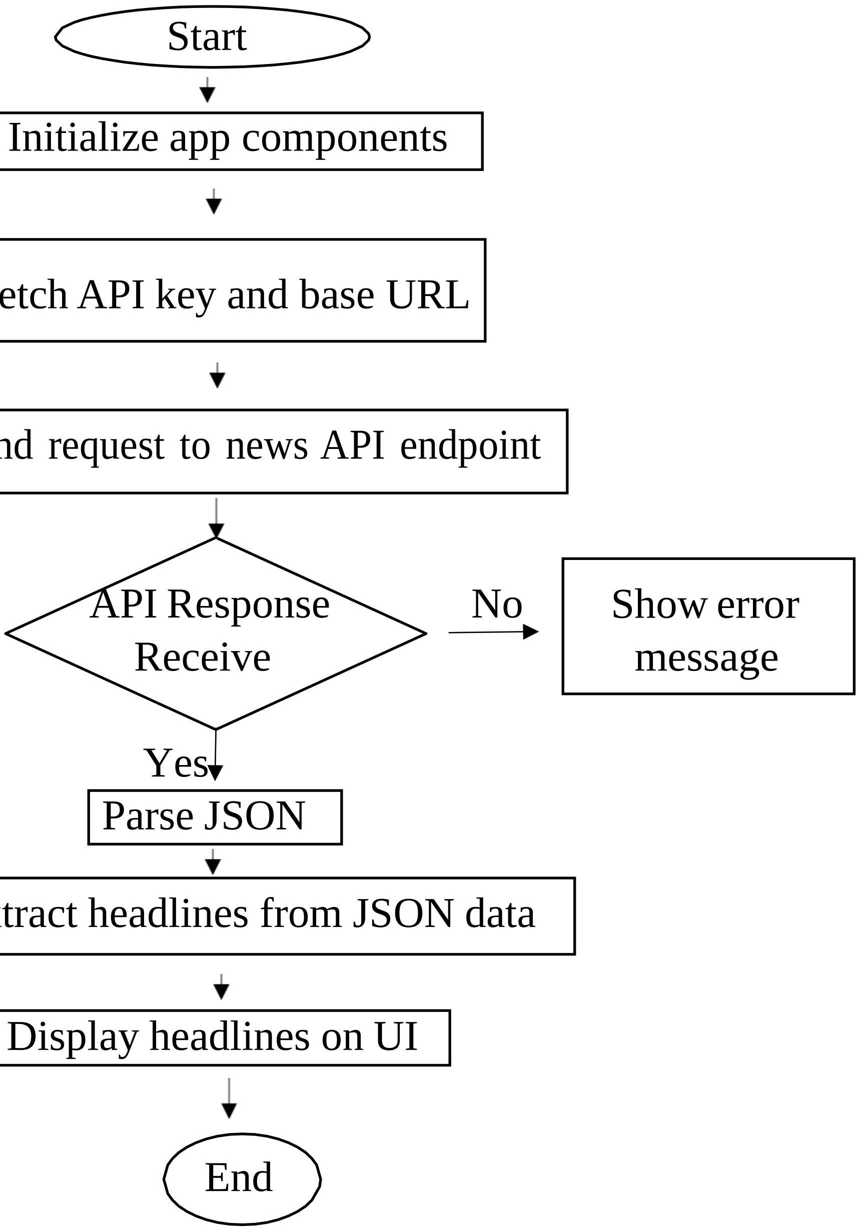
Password: It should be at least a certain number of characters long, like 8 characters, to make it strong enough.

Invalid login details: When users enter the wrong username or password, the app shows a "Incorrect username or password." This message should let them know there was an error without giving away any extra information.

Device Compatibility: Test the login page on various screen sizes and Android OS versions (if supporting multiple versions).

Response Time: Measure the time between submitting credentials and receiving a response to ensure a smooth, responsive user experience.

FUNCTIONALITY:


Start
Initialize app components
Fetch API key and base URL
Send request to news API endpoint
API Response Receive
No
Show error message
Yes
Parse JSON
Extract headlines from JSON data
Display headlines on UI
End

VIDEO LINK:

https://drive.google.com/file/d/146h7YAcRtz5UjDGvDCGz4ulNAg_lPATb/view?usp=drive_link

APPENDIX:

```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"
```

```
<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>
```

API SERVICE:

```
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=684cb893caf7425abeffad82ac1d0f4e")
    ///@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:

```
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,


  )
```

NEWS DISPLAY:

```
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) }
    )
}
```

LOGIN ACTIVITY:

```
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.unit.dp
import androidx.compose.ui.unit.sp
import androidx.core.content.ContextCompat
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 {
            NewsHeadlinesTheme {
                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 = "News Icon"
        )
```

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

LoginHeader()

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

UsernameField(username) { username = it }

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

PasswordField(password) { password = it }

Spacer(modifier = Modifier.height(12.dp))

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

LoginButton {
    if (username.isNotEmpty() && password.isNotEmpty()) {
        val user = databaseHelper.getUserByUsername(username)
        if (user != null && user.password == password) {
            error = "Successfully logged in"
            startMainPage(context)
        } else {
            error = "Invalid username or password"
```

```
                }
            } else {
                error = "Please fill all fields"
            }
        }

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

        ActionsRow(context)
    }
}

@Composable
fun LoginHeader() {
    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)
        Divider(color = Color.LightGray, thickness = 2.dp, modifier = Modifier.width(155.dp).padding(top = 20.dp, start = 20.dp))
    }
}

@Composable
fun UsernameField(username: String, onUsernameChange: (String) -> Unit) {
    TextField(
        value = username,
        onValueChange = onUsernameChange,
        leadingIcon = {
            Icon(
                imageVector = Icons.Default.Person,
                contentDescription = "Person Icon",
                tint = Color(0xFF6495ED)
            )
        },
        placeholder = { Text(text = "Username", color = Color.Black) },
        colors = TextFieldDefaults.textFieldColors(backgroundColor = Color.Transparent)
    )
}

@Composable
fun PasswordField(password: String, onPasswordChange: (String) -> Unit) {
    TextField(
        value = password,
        onValueChange = onPasswordChange,
        leadingIcon = {
            Icon(
                imageVector = Icons.Default.Lock,
                contentDescription = "Lock Icon",
                tint = Color(0xFF6495ED)
            )
        },
        placeholder = { Text(text = "Password", color = Color.Black) },
        visualTransformation = PasswordVisualTransformation(),
        colors = TextFieldDefaults.textFieldColors(backgroundColor = Color.Transparent)
    )
}

@Composable
fun LoginButton(onClick: () -> Unit) {
    Button(
        onClick = onClick,
        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)
    }
}

@Composable
fun ActionsRow(context: Context) {
    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 = { /* Implement Forgot Password */ }) {
            Text(text = "Forgot password?", color = Color.Black)
        }
    }
}

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

MAINACTIVITYKT:

```
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) }
        )
    }
}
```

MAINVIEW MODEL:

```
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()
            }
        }
    }
}
```

MAIN PAGE:

```
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()
)
```

REGISTRATION ACTIVITY:

```
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:

```
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:

```
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:

```
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)
}
```

STRING XML:

```
<resources>
    <string name="app_name">News Headlines</string>
    <string name="title_activity_main2">MainActivity2</string>
    <string name="title_activity_main_page">MainPage</string>
    <string name="title_activity_login">LoginActivity</string>
    <string name="title_activity_registration">RegistrationActivity</string>
    <string name="title_activity_display_news">DisplayNews</string>
</resources>
```

THEME XML:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>


    <style name="Theme.NewsHeadlines"
parent="android:Theme.Material.Light.NoActionBar">
        <item name="android:statusBarColor">@color/purple_700</item>
    </style>
</resources>
```

COLOUR XML:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFFFC107</color> <!-- Amber -->
    <color name="purple_500">#FFFF9800</color> <!-- Deep Orange -->
    <color name="purple_700">#FFF57C00</color> <!-- Darker Orange -->
    <color name="teal_200">#FF4CAF50</color> <!-- Green -->
    <color name="teal_700">#FF2E7D32</color> <!-- Dark Green -->
    <color name="black">#FF37474F</color> <!-- Blue Grey -->
    <color name="white">#FFFAFAFA</color> <!-- Light Grey -->
</resources>
```

BUILD GRADLE:

```
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
}

android {
    namespace 'com.example.newsheadlines'
    compileSdk 33

    defaultConfig {
        applicationId "com.example.newsheadlines"
        minSdk 21
        targetSdk 33
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
        vectorDrawables {
            useSupportLibrary true
        }
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = '1.8'
    }
    buildFeatures {
        compose true
    }
    composeOptions {
        kotlinCompilerExtensionVersion '1.2.0'
    }
    packagingOptions {
        resources {
            excludes += '/META-INF/{AL2.0,LGPL2.1}'
        }
    }
}

dependencies {


    implementation 'androidx.core:core-ktx:1.7.0'
    implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.3.1'
    implementation 'androidx.activity:activity-compose:1.3.1'
    implementation "androidx.compose.ui:ui:$compose_ui_version"
    implementation "androidx.compose.ui:ui-tooling-preview:$compose_ui_version"
    implementation 'androidx.compose.material:material:1.2.0'
    implementation 'androidx.room:room-common:2.5.0'
    implementation 'androidx.room:room-ktx:2.5.0'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
    androidTestImplementation "androidx.compose.ui:ui-test-junit4:$compose_ui_version"
    debugImplementation "androidx.compose.ui:ui-tooling:$compose_ui_version"
    debugImplementation "androidx.compose.ui:ui-test-manifest:$compose_ui_version"


    // Retrofit
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation "com.squareup.okhttp3:okhttp:5.0.0-alpha.2"
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
    implementation("io.coil-kt:coil-compose:1.4.0")
}
```

BACKGROUD XML:

```
<?xml version="1.0" encoding="utf-8"?>
<vector
    android:height="108dp"
    android:width="108dp"
    android:viewportHeight="108"
    android:viewportWidth="108"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <path android:fillColor="#3DDC84"
          android:pathData="M0,0h108v108h-108z"/>
    <path android:fillColor="#00000000" android:pathData="M9,0L9,108"
          android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,0L19,108"
          android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M29,0L29,108"
          android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
```

```
<path android:fillColor="#00000000" android:pathData="M39,0L39,108"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M49,0L49,108"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M59,0L59,108"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M69,0L69,108"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M79,0L79,108"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M89,0L89,108"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M99,0L99,108"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,9L108,9"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,19L108,19"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,29L108,29"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,39L108,39"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,49L108,49"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,59L108,59"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,69L108,69"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,79L108,79"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M0,89L108,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M0,99L108,99"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,29L89,29"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,39L89,39"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,49L89,49"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,59L89,59"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,69L89,69"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,79L89,79"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M29,19L29,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M39,19L39,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M49,19L49,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M59,19L59,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M69,19L69,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M79,19L79,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
</vector>
```

FOREGROUND XML:

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:aapt="http://schemas.android.com/aapt"
    android:width="108dp"
    android:height="108dp"
    android:viewportWidth="108"
    android:viewportHeight="108">
    <path android:pathData="M31,63.928c0,0 6.4,-11 12.1,-13.1c7.2,-2.6 26,-1.4 26,-
1.4l38.1,38.1L107,108.928l-32,-1L31,63.928z">
        <aapt:attr name="android:fillColor">
            <gradient
                android:endX="85.84757"
                android:endY="92.4963"
                android:startX="42.9492"
                android:startY="49.59793"
                android:type="linear">
                <item
                    android:color="#44000000"
                    android:offset="0.0" />
                <item
                    android:color="#00000000"
                    android:offset="1.0" />
            </gradient>
        </aapt:attr>
    </path>
    <path
        android:fillColor="#FFFFFF"
        android:fillType="nonZero"
        android:pathData="M65.3,45.828l3.8,-6.6c0.2,-0.4 0.1,-0.9 -0.3,-1.1c-0.4,-0.2 -0.9,-0.1
-1.1,0.3l-3.9,6.7c-6.3,-2.8 -13.4,-2.8 -19.7,0l-3.9,-6.7c-0.2,-0.4 -0.7,-0.5 -1.1,-
0.3C38.8,38.328 38.7,38.828 38.9,39.228l3.8,6.6C36.2,49.428 31.7,56.028
```

```
31,63.928h46C76.3,56.028 71.8,49.428 65.3,45.828zM43.4,57.328c-0.8,0 -1.5,-0.5 -1.8,-
1.2c-0.3,-0.7 -0.1,-1.5 0.4,-2.1c0.5,-0.5 1.4,-0.7 2.1,-0.4c0.7,0.3 1.2,1 1.2,1.8C45.3,56.528
44.5,57.328 43.4,57.328L43.4,57.328zM64.6,57.328c-0.8,0 -1.5,-0.5 -1.8,-1.2s-0.1,-1.5 0.4,-
2.1c0.5,-0.5 1.4,-0.7 2.1,-0.4c0.7,0.3 1.2,1 1.2,1.8C66.5,56.528 65.6,57.328
64.6,57.328L64.6,57.328z"
        android:strokeWidth="1"
        android:strokeColor="#00000000" />
</vector>
```

NEWS BACKGROUND XML:

```
<?xml version="1.0" encoding="utf-8"?>
<vector
    android:height="108dp"
    android:width="108dp"
    android:viewportHeight="108"
    android:viewportWidth="108"
    xmlns:android="http://schemas.android.com/apk/res/android">
    <path android:fillColor="#3DDC84"
          android:pathData="M0,0h108v108h-108z"/>
    <path android:fillColor="#00000000" android:pathData="M9,0L9,108"
          android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,0L19,108"
          android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M29,0L29,108"
          android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M39,0L39,108"
          android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M49,0L49,108"
          android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M59,0L59,108"
          android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
```

```
<path android:fillColor="#00000000" android:pathData="M69,0L69,108"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M79,0L79,108"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M89,0L89,108"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M99,0L99,108"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,9L108,9"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,19L108,19"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,29L108,29"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,39L108,39"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,49L108,49"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,59L108,59"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,69L108,69"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,79L108,79"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,89L108,89"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M0,99L108,99"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
<path android:fillColor="#00000000" android:pathData="M19,29L89,29"
    android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,39L89,39"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,49L89,49"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,59L89,59"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,69L89,69"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,79L89,79"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M29,19L29,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M39,19L39,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M49,19L49,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M59,19L59,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M69,19L69,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M79,19L79,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
</vector>
```

TESTING:

[7:19 am, 15/11/2024] Preethisethuraman: 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)
    }
}
[7:19 am, 15/11/2024] Preethisethuraman: /**
 * Automatically generated file. DO NOT MODIFY
 */
package com.example.newsheadlines;

public final class BuildConfig {
  public static final boolean DEBUG = Boolean.parseBoolean("true");
  public static final String APPLICATION_ID = "com.example.newsheadlines";
  public static final String BUILD_TYPE = "debug";
  public static final int VERSION_CODE = 1;
  public static final String VERSION_NAME = "1.0";
}
```

BUILD GRADLE:

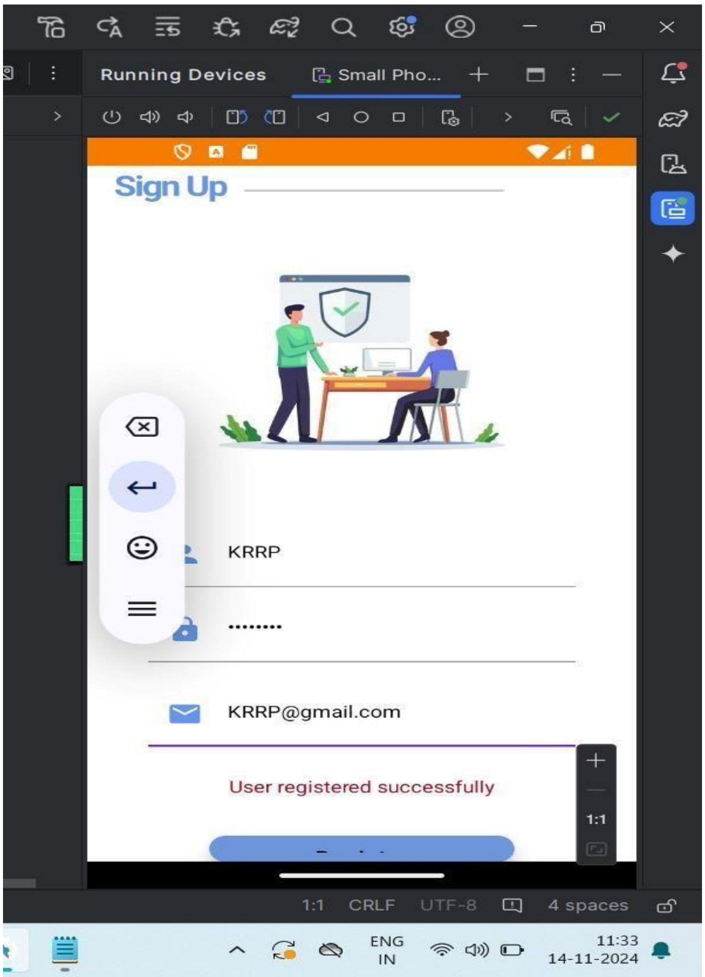
```
plugins {
    id 'com.android.application'
    id 'org.jetbrains.kotlin.android'
}

android {
    namespace 'com.example.newsheadlines'
    compileSdk 33

    defaultConfig {
        applicationId "com.example.newsheadlines"
        minSdk 21
        targetSdk 33
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
        vectorDrawables {
            useSupportLibrary true
        }
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-
rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = '1.8'
    }
    buildFeatures {
        compose true
    }
    composeOptions {
        kotlinCompilerExtensionVersion '1.2.0'
    }
    packagingOptions {
        resources {
            excludes += '/META-INF/{AL2.0,LGPL2.1}'
        }
    }
}

dependencies {

    implementation 'androidx.core:core-ktx:1.7.0'
    implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.3.1'
    implementation 'androidx.activity:activity-compose:1.3.1'
    implementation "androidx.compose.ui:ui:$compose_ui_version"
    implementation "androidx.compose.ui:ui-tooling-preview:$compose_ui_version"
    implementation 'androidx.compose.material:material:1.2.0'
    implementation 'androidx.room:room-common:2.5.0'
    implementation 'androidx.room:room-ktx:2.5.0'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
    androidTestImplementation "androidx.compose.ui:ui-test-junit4:$compose_ui_version"
    debugImplementation "androidx.compose.ui:ui-tooling:$compose_ui_version"
    debugImplementation "androidx.compose.ui:ui-test-manifest:$compose_ui_version"


    // Retrofit
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation "com.squareup.okhttp3:okhttp:5.0.0-alpha.2"
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'


    implementation("io.coil-kt:coil-compose:1.4.0")



}
```

```
<?xml version="1.0" encoding="utf-8"?><!--
   Sample backup rules file; uncomment and customize as necessary.
   See https://developer.android.com/guide/topics/data/autobackup
   for details.
   Note: This file is ignored for devices older that API 31
   See https://developer.android.com/about/versions/12/backup-restore
-->
<full-backup-content>
    <!--
   <include domain="sharedpref" path="."/>
   <exclude domain="sharedpref" path="device.xml"/>
-->
</full-backup-content>
```

```
<?xml version="1.0" encoding="utf-8"?><!--
   Sample data extraction rules file; uncomment and customize as necessary.
   See https://developer.android.com/about/versions/12/backup-restore#xml-changes
   for details.
-->
<data-extraction-rules>
```

```
    <cloud-backup>
        <!-- TODO: Use <include> and <exclude> to control what is backed up.
        <include .../>
        <exclude .../>
        -->
    </cloud-backup>
    <!--
    <device-transfer>
        <include .../>
        <exclude .../>
    </device-transfer>
    -->
</data-extraction-rules>
```

OUTPUT:



Running Devices

Small Pho...

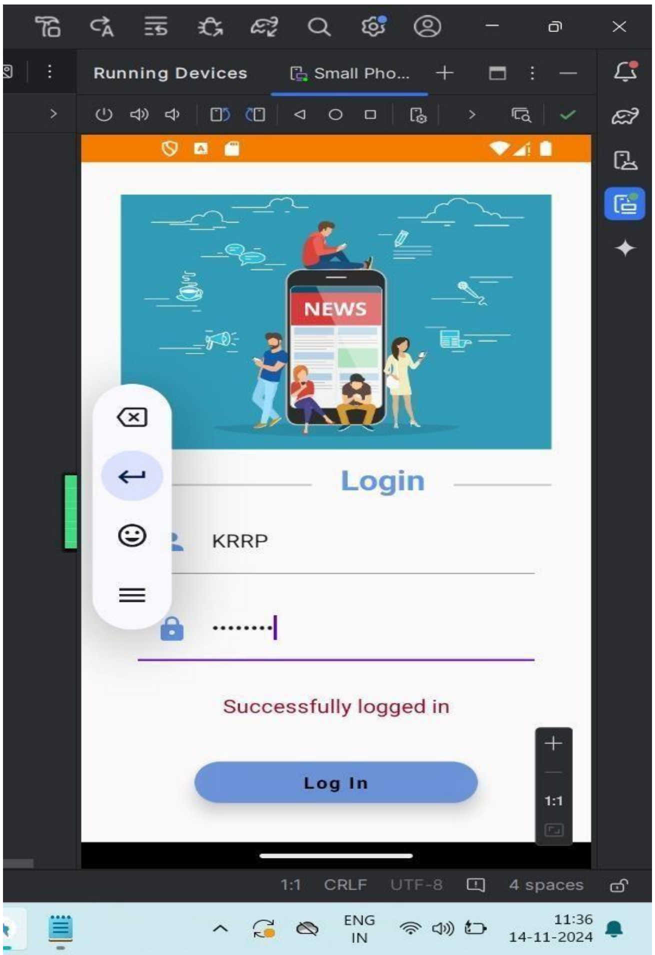
Sign Up

KRRP

••••••••

KRRP@gmail.com

User registered successfully



Running Devices

Small Pho...

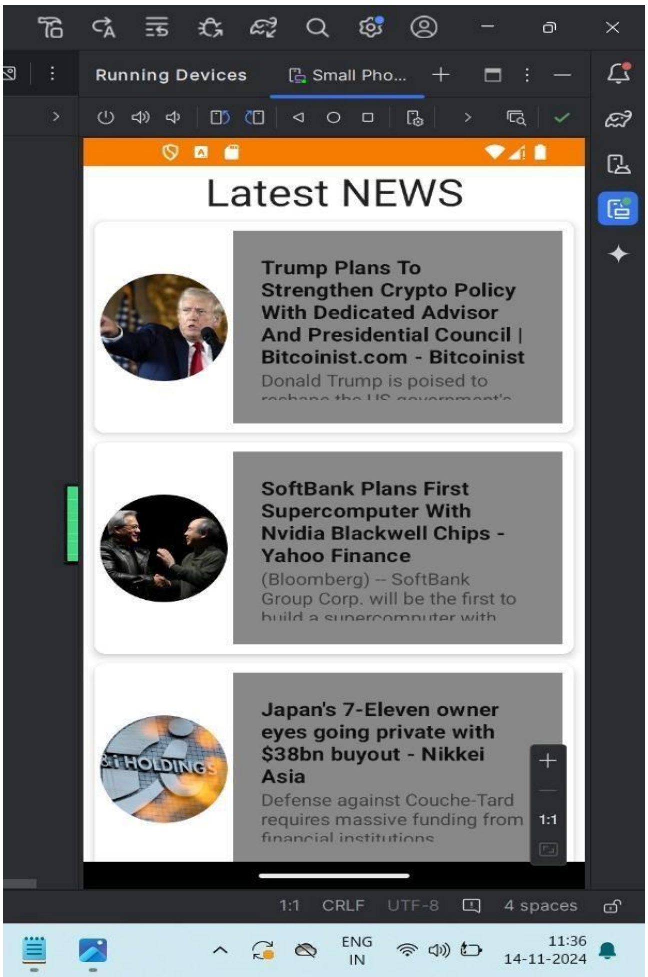
Login

KRRP

••••••••

Successfully logged in

Log In

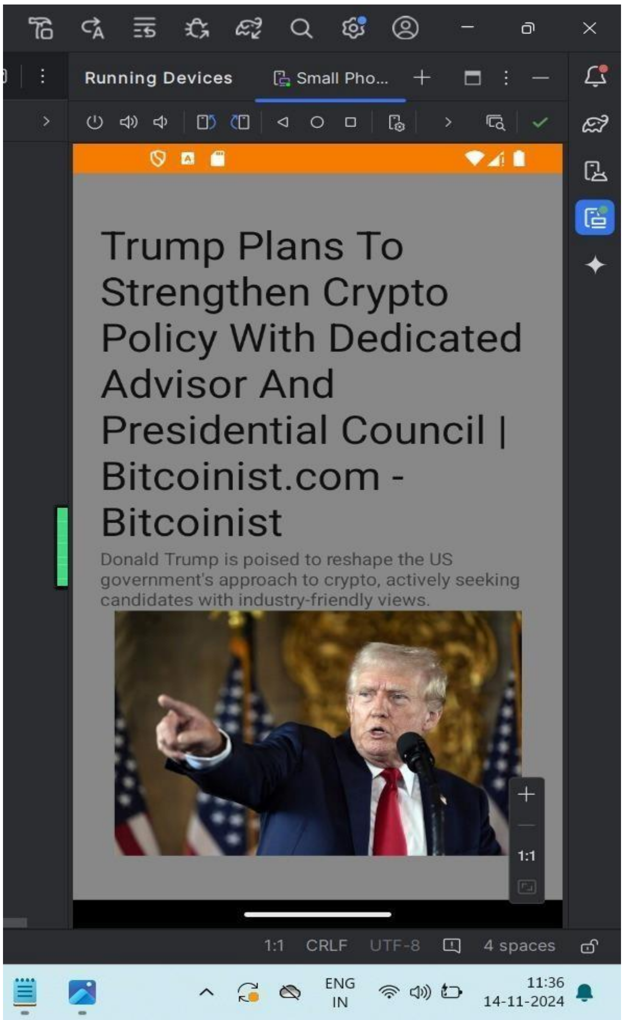


Latest NEWS

Trump Plans To Strengthen Crypto Policy With Dedicated Advisor And Presidential Council | Bitcoinist.com - Bitcoinist
Donald Trump is poised to reshape the US government's

SoftBank Plans First Supercomputer With Nvidia Blackwell Chips - Yahoo Finance
(Bloomberg) -- SoftBank Group Corp. will be the first to build a supercomputer with

Japan's 7-Eleven owner eyes going private with $38bn buyout - Nikkei Asia
Defense against Couche-Tard requires massive funding from financial institutions



Trump Plans To Strengthen Crypto Policy With Dedicated Advisor And Presidential Council | Bitcoinist.com - Bitcoinist

Donald Trump is poised to reshape the US government's approach to crypto, actively seeking candidates with industry-friendly views.

References:

News APIs: Using APIs like News API, The Guardian API, or NY Times API allows you to fetch the latest news articles. This is essential for keeping the app's content up-to-date.

Reference Documentation: Ensure you reference the official documentation for these APIs. For example, the News API documentation provides guidelines on authentication, querying, and usage limits.

Retrofit: Retrofit is a popular HTTP client for Android that simplifies API requests. Retrofit's documentation on GitHub is an excellent resource for understanding how to make network requests in Android.

Firebase Cloud Messaging (FCM): For real-time news updates, FCM allows you to send notifications about breaking news. Firebase provides a comprehensive guide for integrating FCM into Android apps.

GitHub/GitLab: NewsheadlinesApp for source control. Platforms like GitHub or GitLab are excellent for version control and code collaboration.

Project Hurdles:

1. API Integration: Integrating a news API can present challenges, particularly in handling API keys securely and parsing JSON responses accurately.

2. Error Handling: Managing network errors and API call failures gracefully to ensure a smooth user experience.

3. UI Consistency: Designing an intuitive UI that displays news headlines clearly while accommodating various screen sizes.

Conclusion:

This project successfully demonstrates how to fetch and display real-time news headlines using a Kotlin-based app. By integrating a news API and handling network responses, the app provides a foundation for scalable news applications. The project overcomes challenges related to API integration, error handling, and UI design, resulting in a streamlined news-viewing experience for users.