AN ANDROID APPLICATION FOR KEEPING UP WITH THE LATEST

HEADLINES

J.Boomika

S.Arockiya Jenifar

S.Mathanki

M.Nurul Hafiza

G.Brindha

Introduction

* 1. Overview
* The app’s main feature is displaying a list of news articles, each with a title, image, and brief description.

The app uses the Jetpack Composes UI toolkit

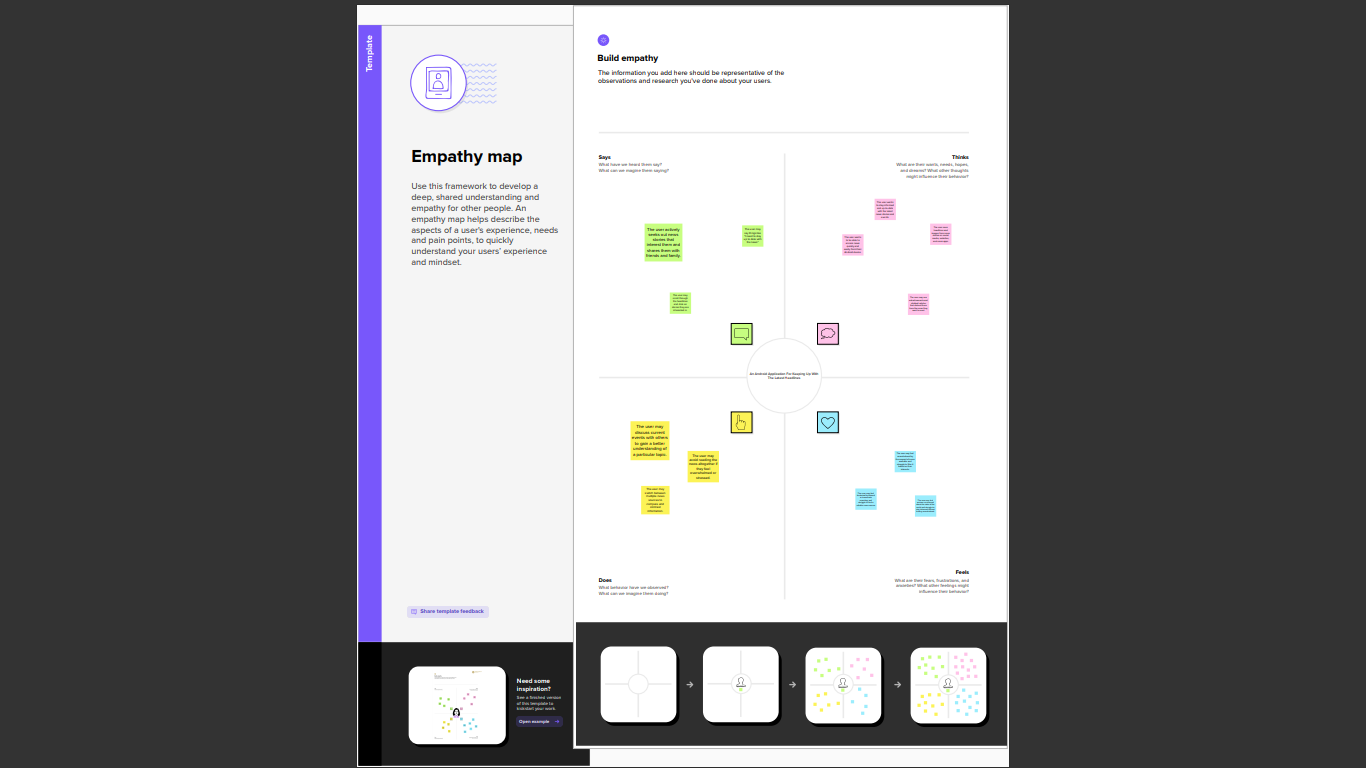
* To build the UI and it uses the coil library to load images.

Project workflow:

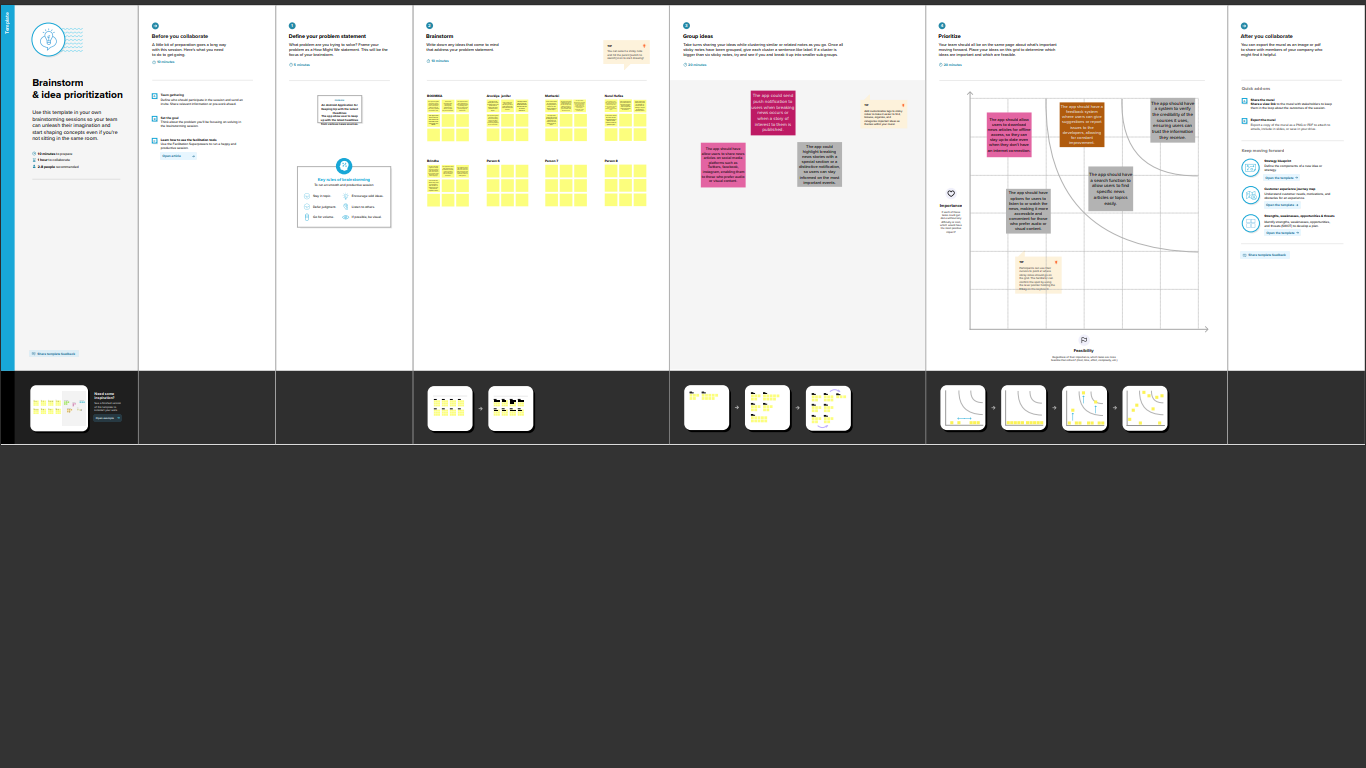
* Fast and easy experience that your readers expect from your mobile apps.
* Push notification, infinite scrolling, swipe action, content recommendations, native sharing, rich image galleries
* Offline mode that helps your readers to access content even at flow internet speeds.
  1. Purpose:
* Users can scroll through the list of articles and tap on an article to view more details.
* The app fetches data from a remote server using retrofit library and demonstrates how to use the jetpack compose UI toolkit for android development.
* Users register into application.
* After registration, user logins into the application.
* Users enter into the main page.

2. Problem definition & Design Thinking

2.1 Empathy map:

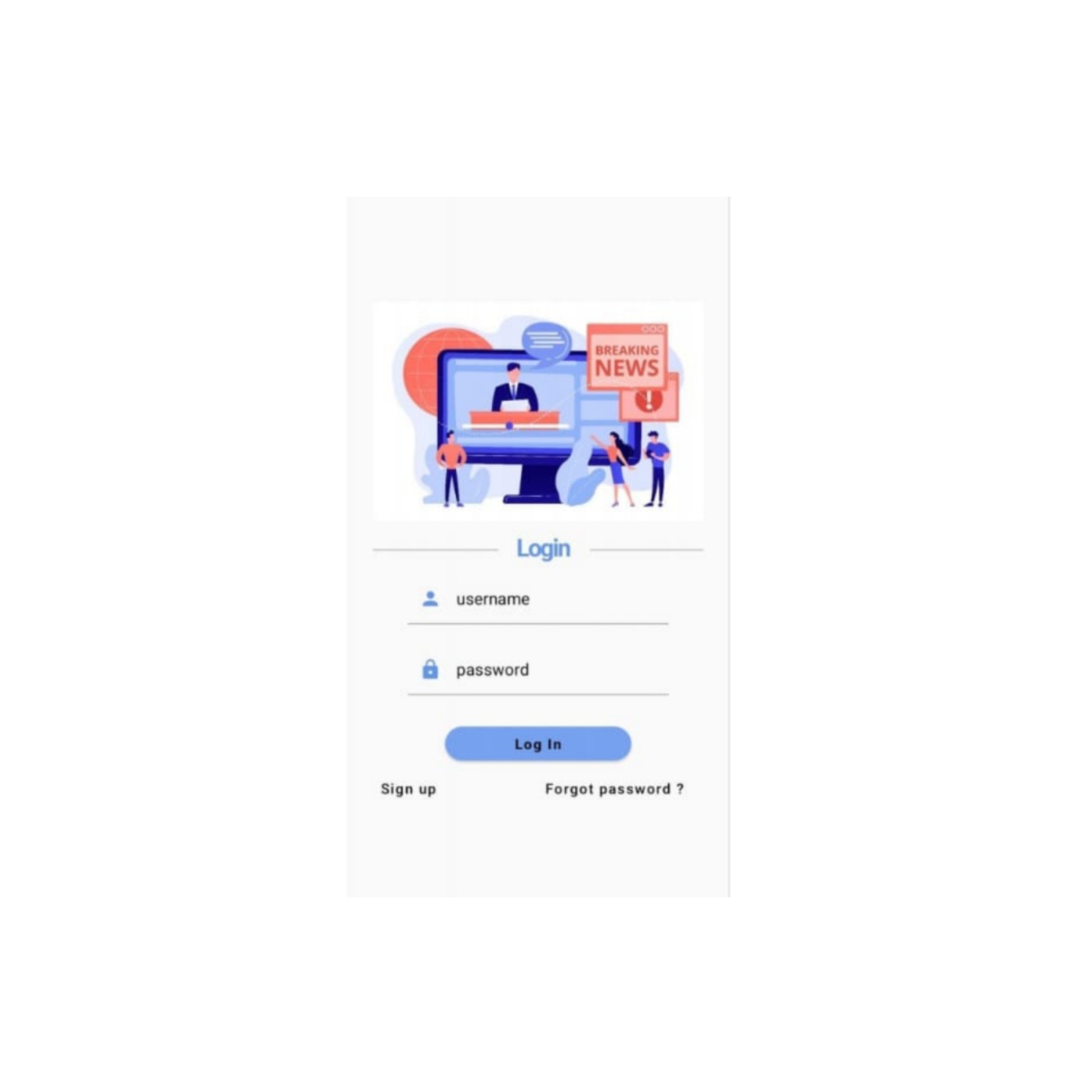


2.2 ideation & brainstorming:

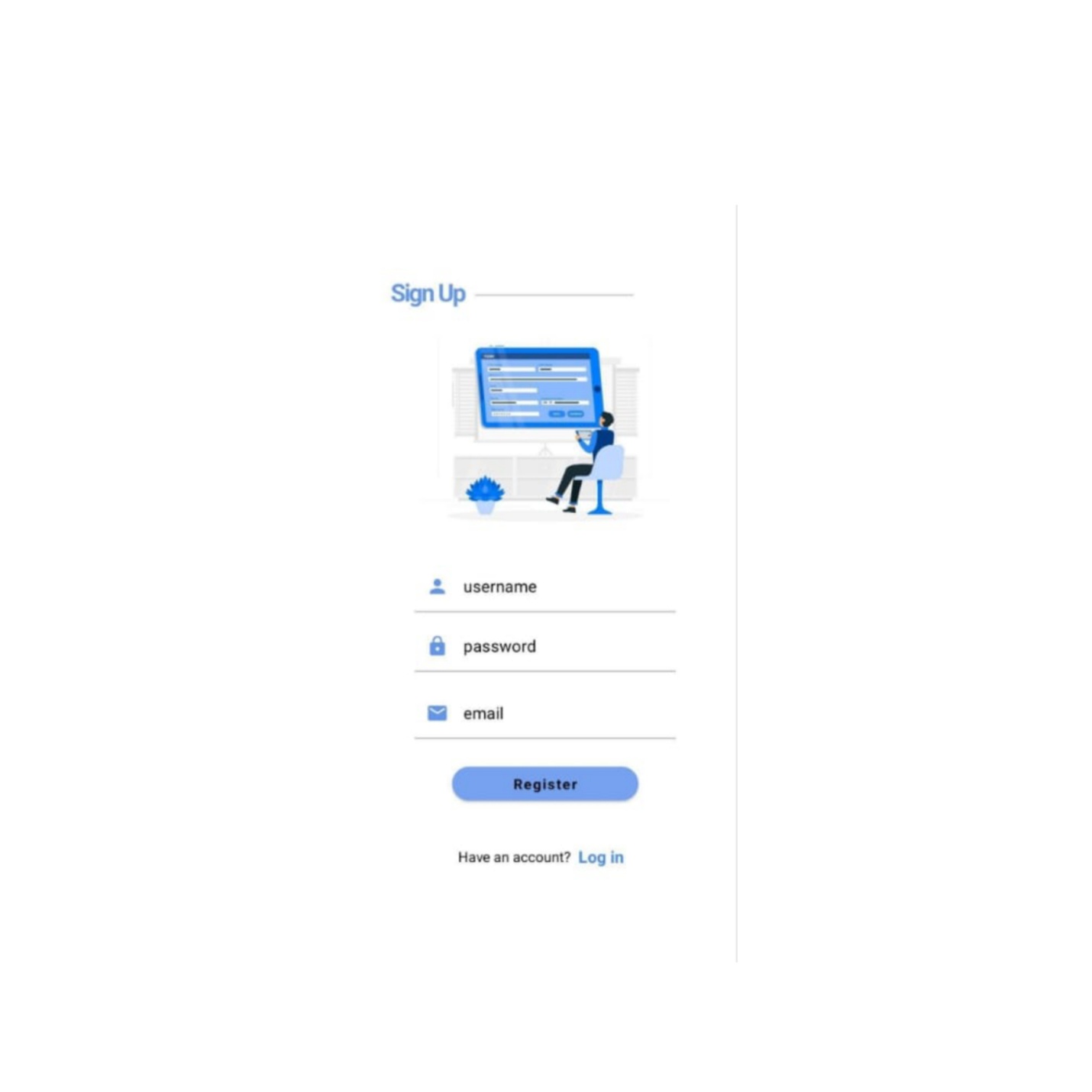


3.Result:

Login page:



Register Page:



Main News Headlines Page:



Display NEWS page:



4. Advantages and Disadvantages:

Advantages:

* Convenience: An Android application allows you to quickly and easily access the latest news headlines from your mobile or tablets no matter where you are. You can read news on the go without having to carry a physical newspaper or turning on your computer.
* Customization: Many Android News Application allow you to customize the news you see based on your interests. You can choose to receive news on specific topics or form specific sources, allowing you to focus on the news that matter most to you.
* Real-time updates: Android news applications are constantly updating their content to bring you the latest news as it happens. You can receive notifications about breaking news or important stories, ensuring that you never miss an important update.
* Social Sharing: Many Android news applications allow you to share news stories with your friends and family on social media platforms, making it easy to stay informed and share important news with others.

Disadvantages:

* Reliance on technology: using an Android application for news means you are reliant on technology to access and read the news. If there is a technical issue with the app or your device, you may not be able to access the news when you need it.
* Limited perspective: Many Android news application allow you to customize the news you see based on your interests, which can limit your exposure to diverse viewpoints and opinions. This can result in a narrow perspective on current events.
* Distractions: Android news applications can be distracting, with notifications and updates constantly popping up on your device. This can make it difficult to focus on other tasks or responsibilities.
* Privacy concerns: Using an Android application for news may require sharing personal information, such as your location or browsing history. This can raise privacy concerns and increase the risk of data breaches or identity theft.

5. Application:

I am interested in an Android application that provides me with the latest headlines from around the world. As a busy individual, I don’t always have the have time to read a complete news article, but I would like to stay informed of major news events.

I am looking for an application that offers a simple, user-friendly interface, which displays the latest headlines from a variety of news sources. The application should allow me to customize the list of sources according to my preferences, and allow me to select specific topics that I am interested in.

Moreover, I would like the application to provide brief summaries of the news stories, so that I can quickly and easily grasp the main points of the article. Additionally, it would be helpful if the application could provide links to the full articles in case I want to read more about a particular news item.

I also believe that a features that allows me to save news stories for later reading would be useful, as well as the ability to share news stories with my friends and family vis social media or email.

Overall, I believe that an Android application that provides me with the latest headlines in a quick and convenient way would be a valuable tool in keeping me informed of current events. I look forward to hearing more about your plans for such an application and would be happy to provide any further feedback or input that may be helpful.

6. Conclusion: In conclusion, an Android application for keeping up with the latest headlines would be a valuable tool for staying informed about current events. Whether it is for personal or professional reasons, having quick and easy access to the latest news can help individuals make better-informed decisions and stay up to date with what is happening in the world.

A good application should offer a variety of news source and allows users to customize their preferences, filter news by topics or location, provide brief summaries of news stories, and offer links to full articles for further reading. It should also include features like the ability to save news save news stories for later reading and share them with others via social media or email.

Overall, such an application would be a valuable addition to any android user’s device, providing a convenient and user-friendly way to keep up with the latest headlines and stay informed about the world around them.

7.Future Scope: The future Scope for an android application for keeping up with the latest headlines is quite promising. As the world becomes increasingly interconnected and information is shared faster than ever, the demand for real-time news updates is only set to grow.

Some potential future features that could be included in such an application include:

Personalization: The application could use artificial could use artificial intelligence(AI) to analyze a user’s reading habits and preferences, and recommend news articles that are most relevant to them.

Voice Assistant Integration: The application could integrate with popular voice Assistants or Amazon, allowing users to hear the latest news headlines by simply asking their devices.

Social Media Integration: The application could integrate with social media platforms to allow users to share news stories with their friends and followers.

Blockchain-based News Verification: The application could leverage blockchain technology to verify the authenticity of news sources and stories, ensuring that users receive accurate and reliable information.

There is a tremendous potential for an android application for keeping up with the latest headlines, and developers should continue to explore new ways to improve the user experience and provide valuable features to users.

8.Appendix:

A. Source code:

Step 1 : Create User data class

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?,

)

Step 2 : Create an UserDao interface

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)

}

Step 3 : Create an UserDatabase class

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

}

}

}

}

Step 4 : Create an UserDatabaseHelper class

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

}

}

Create ApiService interface

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!!

}

}

}

Create Model data class

package com.example.newsheadlines

data class Movie(val name: String,

val imageUrl: String,

val desc: String,

val category: String)

Create News data class

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()

)

Create Source data class

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

)

Create Articles data class

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,

)

Create MainViewModel class

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()

}

}

}

}

Creating LoginActivity.kt with database

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)

}

Creating RegistrationActivity.kt with database

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)

}

Creating MainActivity.kt file

Download the required drawable from the code

Download the nodpi images from drawablenodpi

Complete MainActivity.kt code

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) }

)

}

}

Complete DisplayNews.kt code

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) }

)

}

Modifying 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>