AN ANDROID APPLICATION FOR KEEPING UP WITH THE LATEST HEADLINES

A PROJECT REPORT

Submitted By

ARSHA.A.S

ASHA.R

ASLIN SHENIGHA.S

AYARLIN BABISHA.A

BACHELOR OF SCIENCE

In

COMPUTER SCIENCE

1. INTRODUCTION

1.1 Overview:

In today's modern world, everyone will search for one of the simplest and the easiest way of approach. Yes, it seems difficult to watch news apart from TV when we travel or at office etc…Hence the news app will bring you to the brightest future.

The news app allows the users to browse through news headlines for different categories such as Top stories, General, Business, Technology, Entertainment, and so on. It helps you to get notified with the push up notifications.

The app was designed with a clean and modern interface, intuitive icons and buttons that make it easy for users to navigate the app. It was developed using the latest Android development technologies. It helps you to organize and understand the news.

1.2 Purpose:

The main focus of this application is to connect news articles from all over the world and deliver it to the user as fast as possible in best visualize way.

Today, the publishing industry is facing such a threat when it comes to newspaper publishing and sales. So, magazine and newspaper lovers are moving towards reading news on mobiles and tablets. The revenue model of the online apps are quite simple and rewarding. They run ads and generate a good amount of money.

News app ascertains that its users can have a great experience and at the same time show the ads in such a way that they don’t irritate the users and divert them from the content they are reading ,yet the same time display the ads.

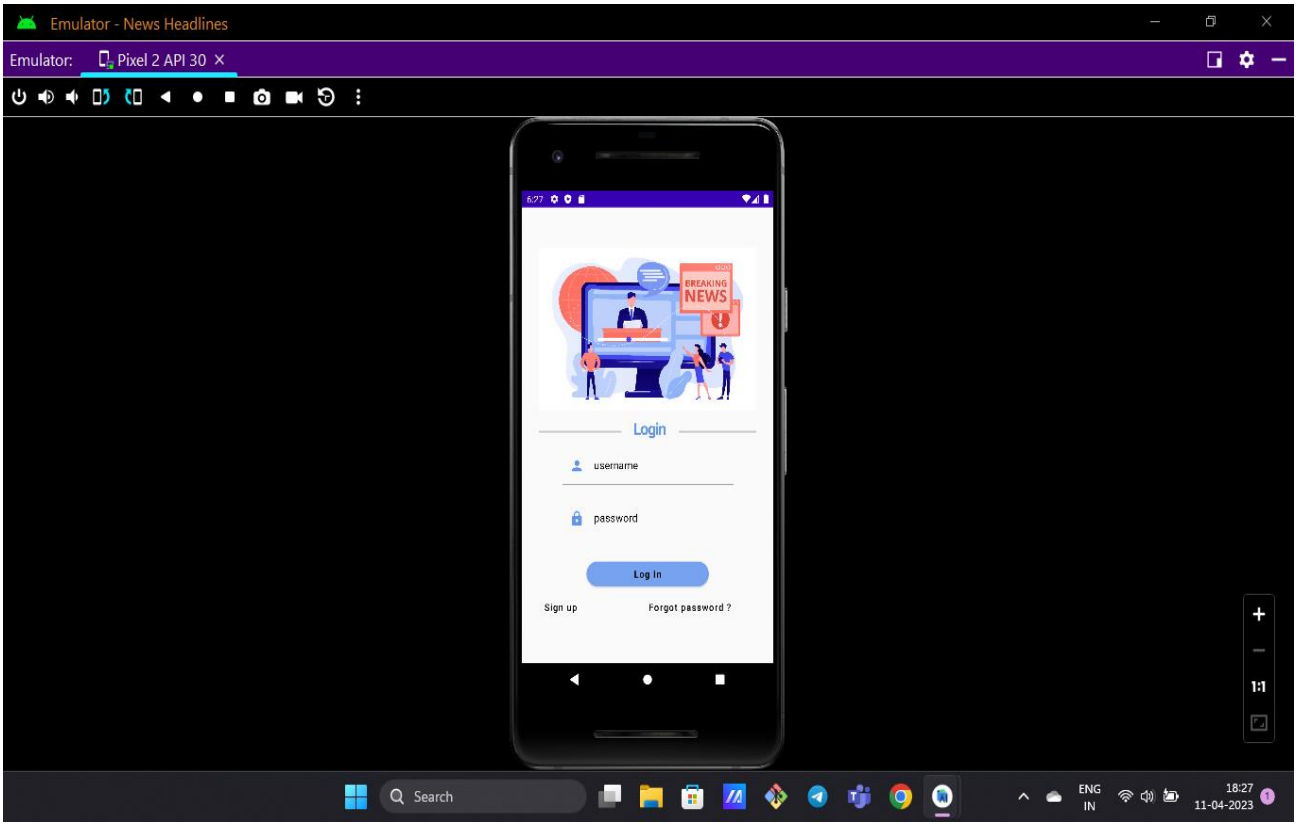
2. PROBLEM DEFINITION & DESIGN THINKING

2.1 Empathy map:

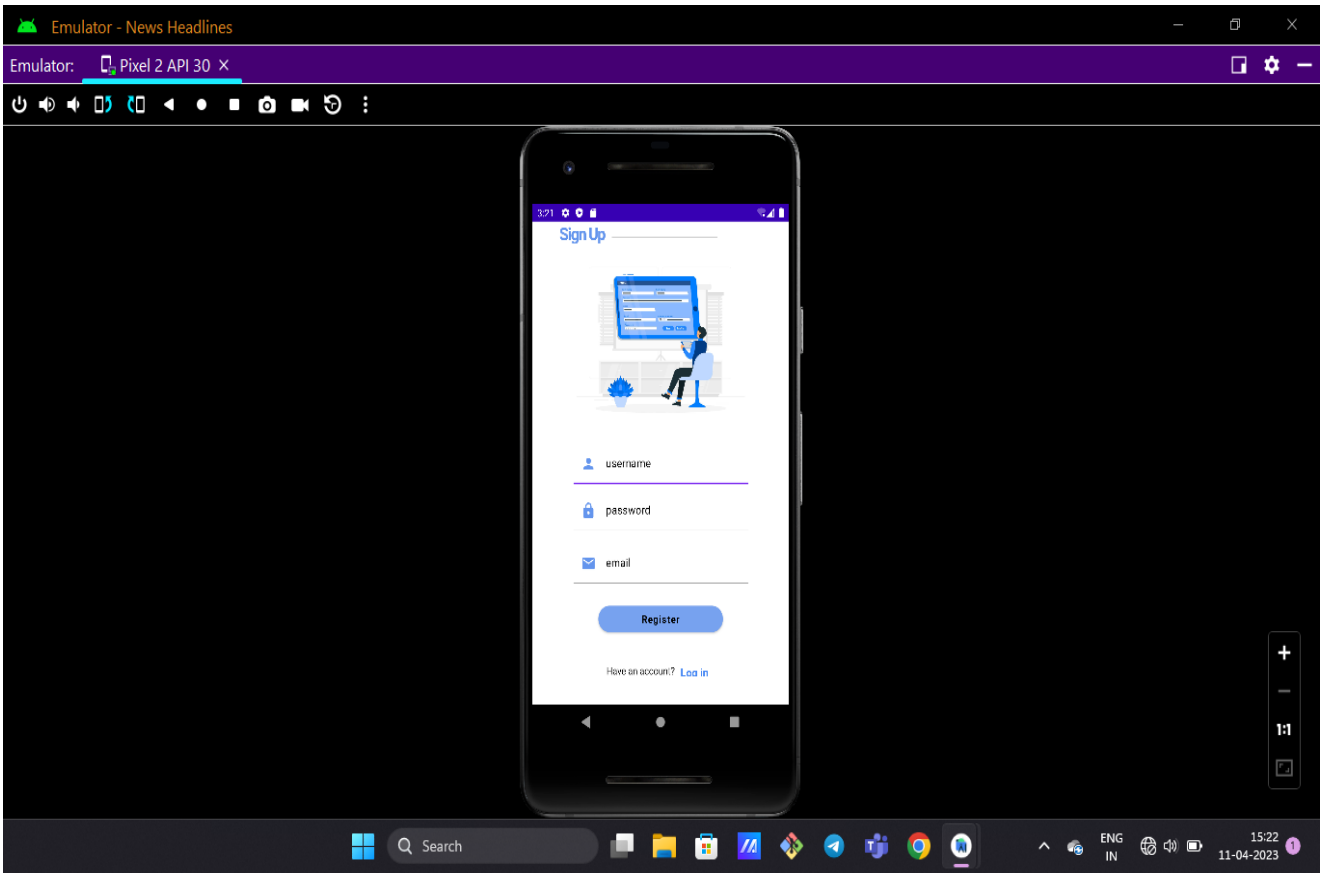
2.2 Ideation & Brainstorming Map:

3. RESULT

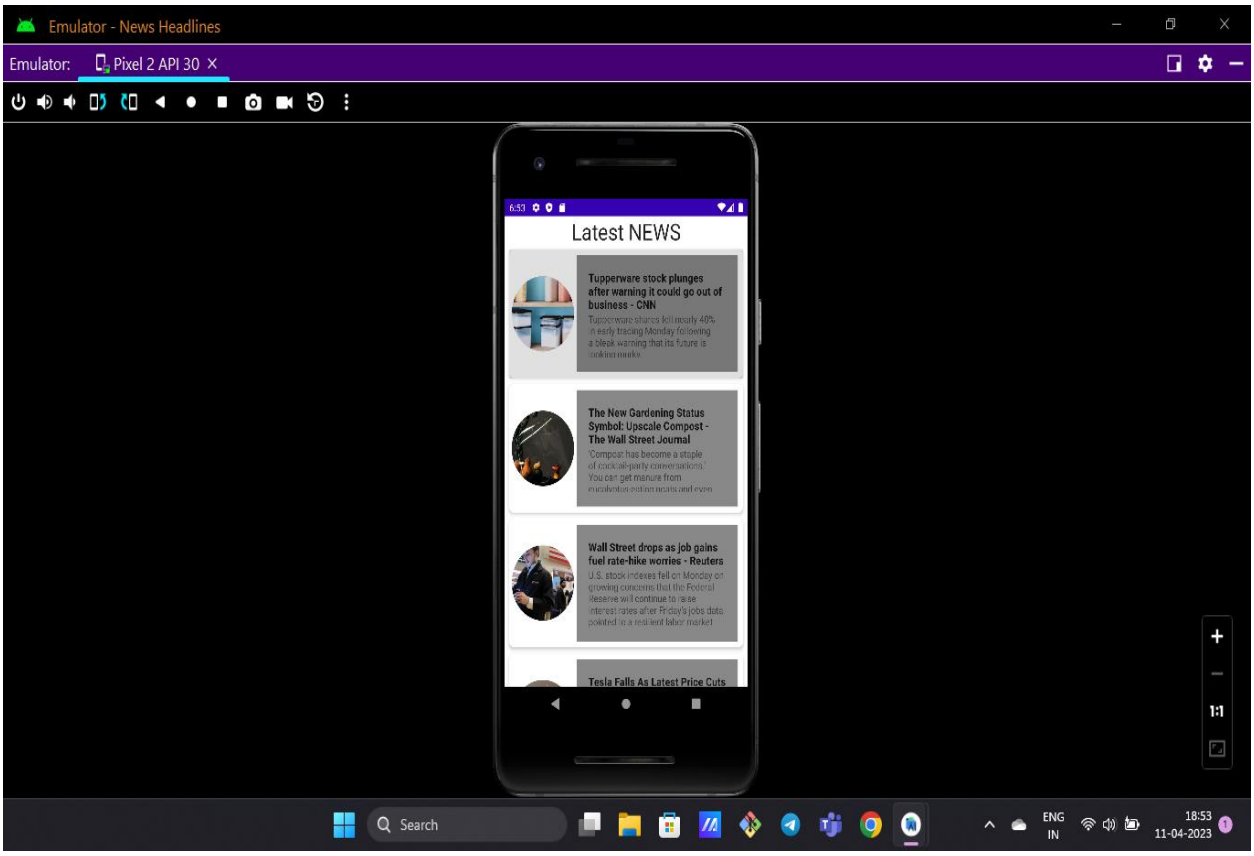
Login Page:



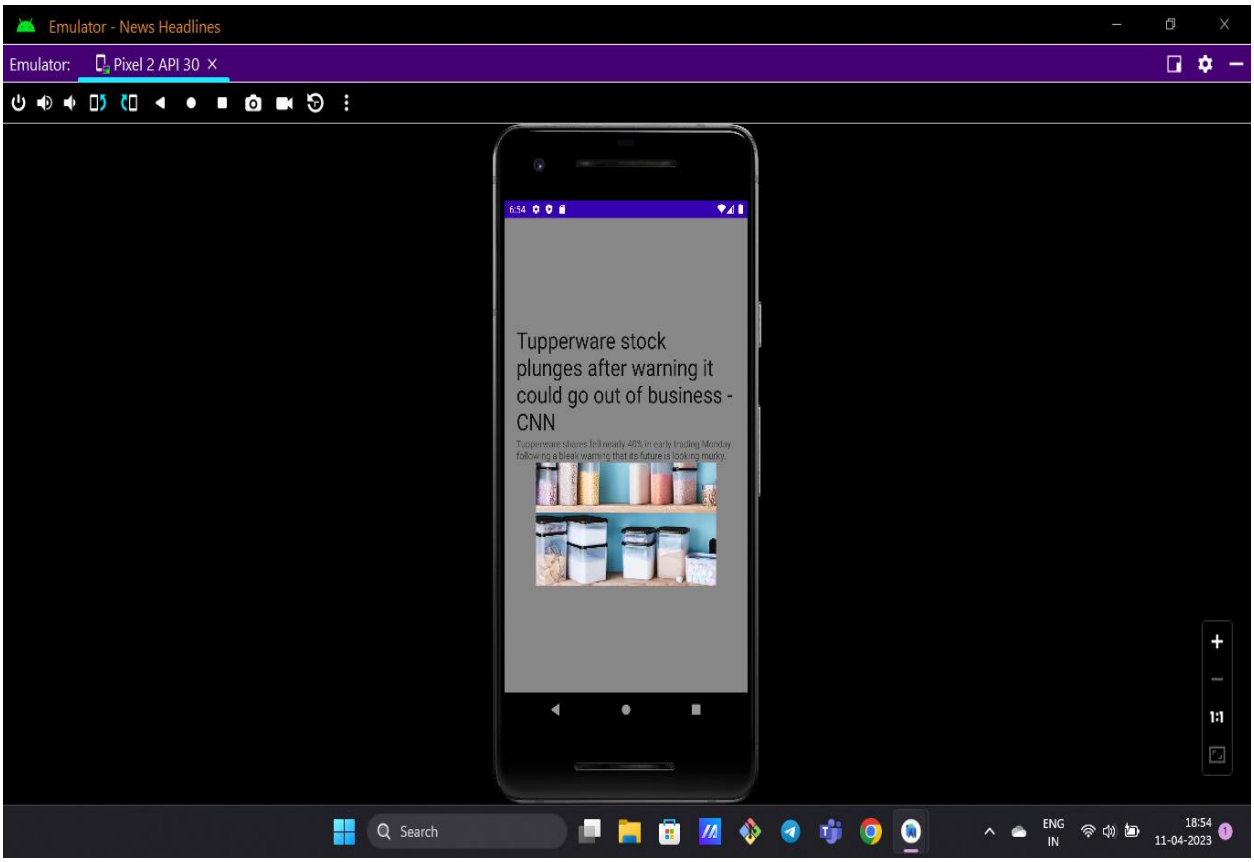
Register Page:



Main Page:


Emulator - News Headlines
Emulator:
Pixel 2 API 30
Latest NEWS
Tupperware stock plunges after warning it could go out of business - CNN
The New Gardening Status Symbol: Upscale Compost - The Wall Street Journal
Wall Street drops as job gains fuel rate-hike worries - Reuters
Tesla Falls As Latest Price Cuts
Search
ENG
IN
18:53
11-04-2023

Display News Page:



4. ADVANTAGES & DISADVANTAGES

Advantages:

Information Goes Paperless:

Every piece of update you receive is through electronic media hence saving the environment. A tree flourishes for every paper saved. With apps available for every newspaper, Media companies can contribute to utilizing natural resources sustainably.

Updates on the Go:

Digital News Media allows you to select the type of news you are into, and these updates are just a tap away. News Apps provide you relevant content as per your preference that you can access with ease anywhere and at any time of the day. So, don't worry if your newspaper guy does not show up at times.

Digital Media keeps you Socially Active:

More and more users receive updates through social platforms like Facebook and Twitter. As celebrities and leading politicians use such platforms to deliver their concerns, many people are influenced by what appears on their profiles. You can share your general views about any of the content that seems sparking.

Vast Employment Options:

Digital News Media Benefits some professionals who are sound with the latest technological concepts. Digital News Platforms and forums are suitable career launch pads for such people who

can join the revolutionizing world and influence their followers. Jobs like social media moderator, Social Media Manager, Social Media Strategist, and Chief Marketing Officer are some renowned profiles in the Digital Media sector.

Digital Journalism:

With everything moved to the digital platforms, digital reporting or digital journalism has turned into a much easier choice of career for many. Many Journalism Institutes have launched digital programs to get the youngsters going with their skills to reporting and spreading an occurrence swiftly as it happens.

Disadvantages:

Fabricated Content/Hoax:

A sudden wave of fake news hits a wider part of global population when circulated through digital platforms like Facebook, Twitter, and WhatsApp. Concern about misinformation and disinformation remains high despite efforts by leading online journalism platforms to build public confidence.

Negative Content:

Accessing the internet is more convenient to everyone. It's easier for unscrupulous elements to spread negativity about a particular individual or group. The spread of unfounded rumors has led to a spate of uncivilized acts in countries like India. Several countries have set up 'tip lines', appealing to the public to flag illegal or dangerous content. The spread of negative or hateful news often

happens via groups that are set up specifically to discuss sensitive issues.

Paid Subscription for Daily News:

Many popular news media allow access to readers only after a paid subscription. Paid subscriptions for daily news is practiced majorly to contribute to revenue. Although it's a must-do for news media these days, many readers would disagree with paying extra money given that there are plenty of other news websites to get the recent news.

Advertising:

Whether you watch your news segment online, on TV, or through an app, you are more likely to spend time on advertisements. Mobile Apps ask you to pay for an ad-free version, which comes as one of the most troubling disadvantages of digital new media these days.

News Overload:

Apart from the leading international news channels, internet has helped many less popular news channels to evolve at a rapid rate. There is more news around the web circulating through different platforms. This is rather more confusing than to figure out what's going on really. Too much information partly reflects how constant news updates and different perspectives can complicate reality. A common complaint is that users are bombarded with multiple versions of the same story or the same alert. The perception of news overload is highest in the United States (40%) followed by Denmark and the Chech Republic.

5. APPLICATIONS

*keeping people informed: The primary purpose of news headlines is to keep people informed about current events, both locally and globally.

*Decision-making: News headlines can be used by individuals, business, and governments to make informed decisions about investments, policy changes, and other important matters.

*Education: News headlines can be used as educational tools to teach people about history, culture and other important topics.

*Entertainment: News headlines can be entertaining, particularly when they involve celebrity gossip, sports, or other popular topics.

*Social commentary: News headlines can be used as a means of social commentary, to highlight issues of social injustice, political corruption, or other important topics.

*Advertising: News headlines can be used as a form of advertising, particularly in the context of sponsored content or native advertising.

6. CONCLUSION

Online newspapers provide a challenge for the theories we employ to analyse and understand texts, for the critical analysis of such texts, and for the development of pedagogical approaches by which knowledge and understanding of such texts can be taught and learned. These challenges are significant, and meeting them will require a range of approaches, and sustained effort. The current research has implications for meeting these challenges, and is intended as a contribution towards understanding of online newspapers and the roles they play in a range of social contexts.

Newspapers have always depended for their survival on being at the edge of information construction and distribution, and the institutionalised power and adaptability of newspapers remain as strengths. The speed of the evolution of the macro-genre of the online newspaper is remarkable. New genres such as newsbites, newsbits, 'videobites', slide shows, blogs, and image galleries are commonplace. Not only this, but story pages often allow readers to link these pages to the very fora that may collectively threaten the newspapers' existence (e.g. del.icio.us, Digg, Facebook, twitter), fora that represent methods of communication that literally did not exist when this research project began. In this way, newspapers become integrated in readers' web-mediated social networks.

7. FUTURE SCOPE

Personalization: One of the biggest trends in news apps is personalization, where the app delivers news tailored to the user's interests and preferences. This can be achieved through machine learning and natural language processing, which can analyze the user's behavior and provide personalized recommendations.

Multimedia: As mobile devices become more powerful, news apps can incorporate more multimedia content, such as videos, podcasts, and interactive graphics, to make the news more engaging and informative.

Augmented Reality (AR) and Virtual Reality (VR): With the increasing popularity of AR and VR technologies, news apps can provide immersive experiences that bring the news to life. For example, users can explore 3D models of news events or attend virtual press conferences.

Blockchain: Blockchain technology can be used to enhance the security and transparency of news apps, making it easier to verify the authenticity of news stories and prevent fake news from spreading.

Collaborative journalism: News apps can facilitate collaboration among journalists from different news organizations, allowing them to work together on investigative stories and share resources.

Social media integration: Many news apps already integrate social media feeds into their platforms, but this trend is likely to continue as social media becomes an increasingly important source of news for many people.

Voice recognition: Voice recognition technology is becoming more sophisticated, and news apps can use this to allow users to interact with the app using voice commands.

8. APPENDIX

A. Source code

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"
```

```
        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)
}
```

ApiService:

```
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,

 )
```

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 urlImage = 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(urlImage),
                contentDescription = "My content description",
              )
            }
            //  Greeting(desk.toString())
          }
        }
      }
    }
}

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

@Preview(showBackground = true)
@Composable
```

```
@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) }
    )
}
```

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:

```
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:

```
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
```

```
    }

    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:

```
ackage 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:

```
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:

```
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")
{
```

```
    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
}
```

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>
```