An Android Application For Keeping Up With The Latest Headlines

Introduction

1.1 Overview

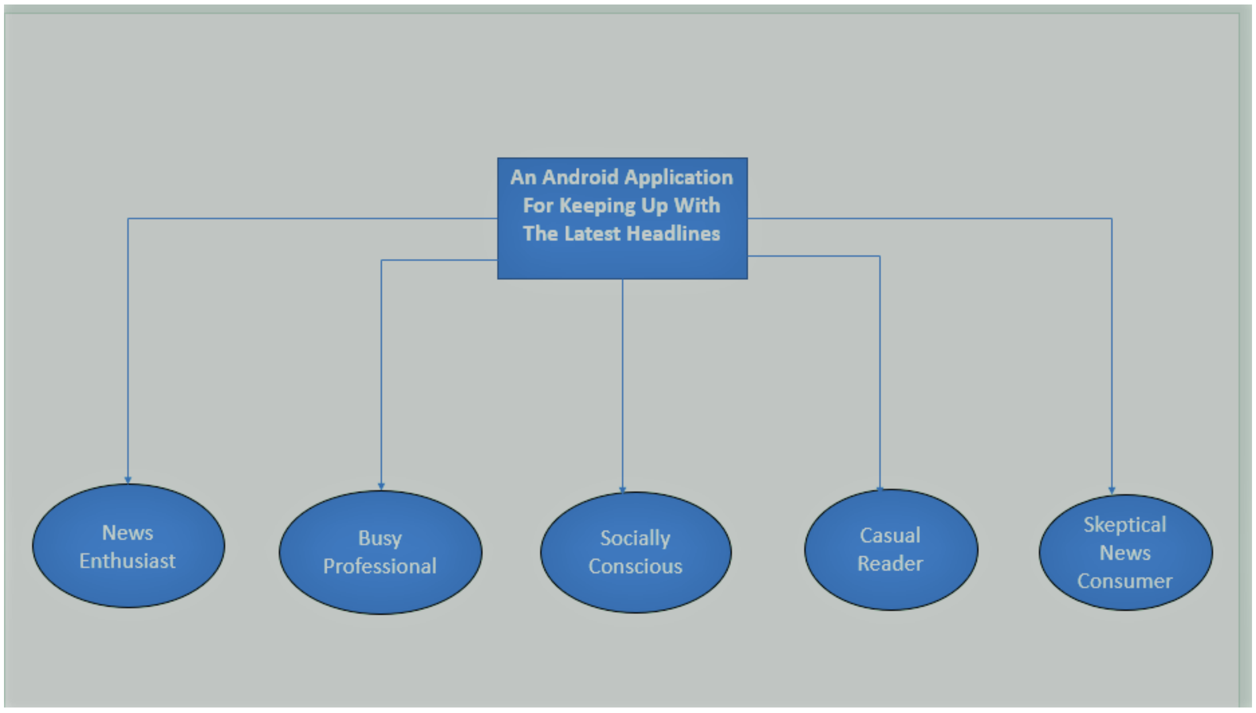
- The app's main feature is displaying a list of news articles, each with a title, image, and brief description. Users can scroll through the list of articles and tap on an article to view more details. The app uses the Jetpack Compose UI toolkit to build the UI and it uses the coil library to load images.
- The app fetches data from a remote server using Retrofit library and demonstrates how to use the Jetpack Compose UI toolkit for Android development.

1.2 Purpose

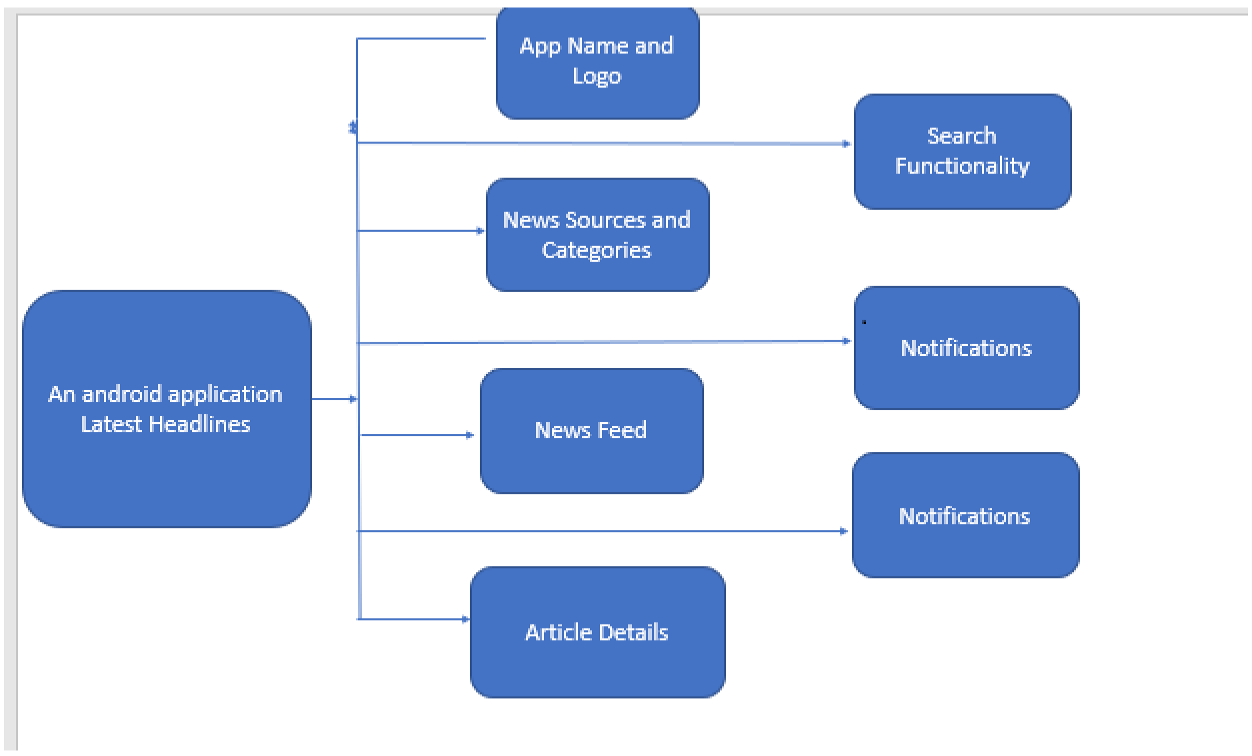
- The main focus of this application is to connect news articles from all around the world and deliver it to use as fast as possible in best visualize way.
- The goal is create a news feed app which gives a user regularly-updated news from the internet related to a particular topic,person,or location.

Problem definition & Design thinking

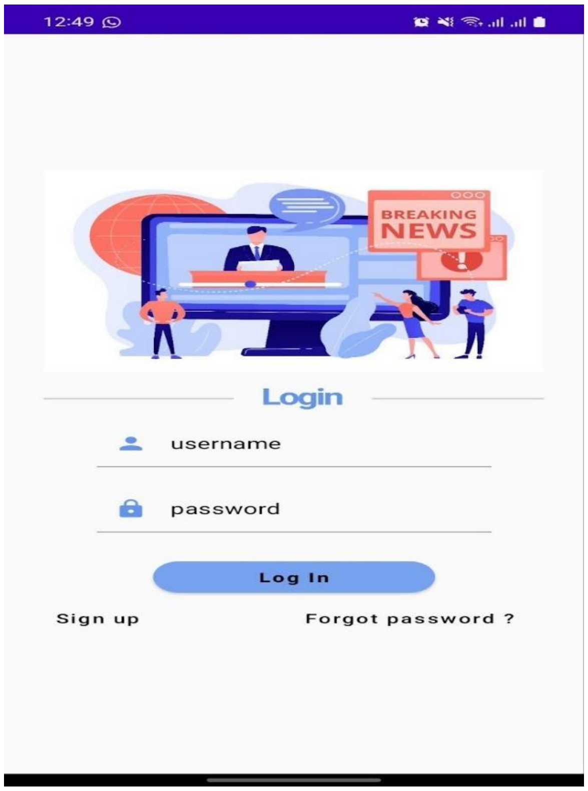
2.1 Empathy Map

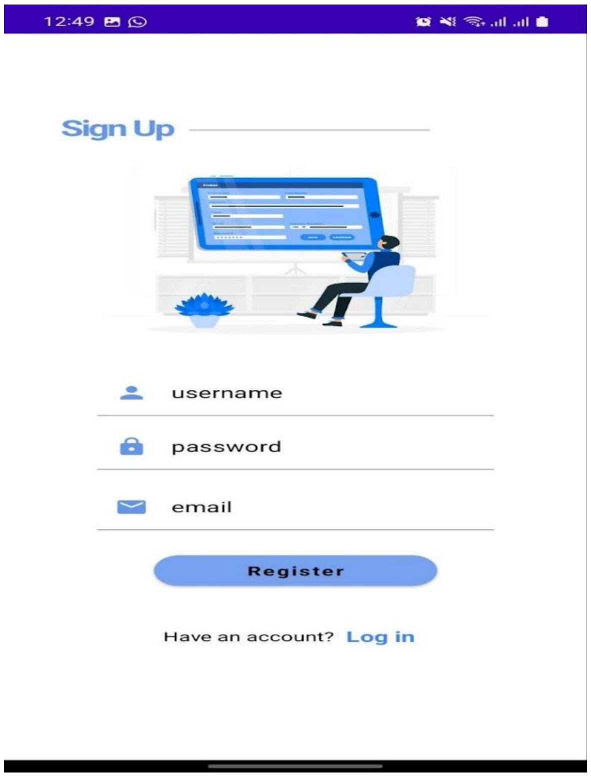


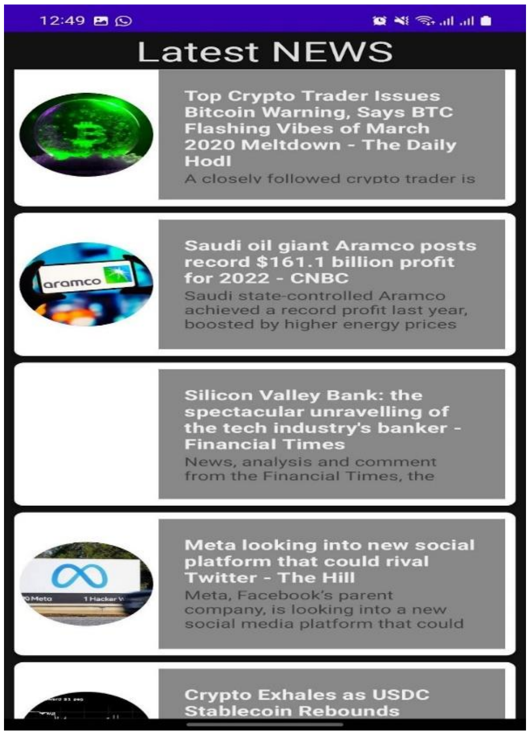
2.2 Ideation & Brainstroming Map


An android application Latest Headlines
App Name and Logo
Search Functionality
News Sources and Categories
Notifications
News Feed
Notifications
Article Details

Result







Latest NEWS

Top Crypto Trader Issues Bitcoin Warning, Says BTC Flashing Vibes of March 2020 Meltdown - The Daily Hodl
A closely followed crypto trader is

Saudi oil giant Aramco posts record $161.1 billion profit for 2022 - CNBC
Saudi state-controlled Aramco achieved a record profit last year, boosted by higher energy prices

Silicon Valley Bank: the spectacular unravelling of the tech industry's banker - Financial Times
News, analysis and comment from the Financial Times, the

Meta looking into new social platform that could rival Twitter - The Hill
Meta, Facebook's parent company, is looking into a new social media platform that could

Crypto Exhales as USDC Stablecoin Rebounds

Advantage and Disadvantages

Advantages:

- The user can get regular news updates related to business,sports,technology,and others on their smartphone with simple to use navigation and in the preferred language as well.
- The user gets automatic updates of the latest news on the app and so does not have to refresh the app window again and again.
- The users get the option to save their favourite news topics for later view by bookmarking in the app.
- The mobile news apps provide the users with push notification as well.

Disadvantage:

- Require data/wifi to get online.
- Companies not making as much money due to free reading for audiences.
- News spreads quicker online-people find out news before they should.
- Lose money-cann't get peopleto pay for digital.
- Older audiences may not access digital platforms.
- Costly to maintain.

Applications

- User was allowed to use this application in his smartphone and screenshorts were taken as a result for this study.First user need to sign up in order to access the application which provides security for this application.
- Also predicted user error handling with pop-up messaging was done before this experiment like entering invalid data in fields,not selecting a field before clicking on action button etc….
- one of the factors in successful news app development is visualization of news and its feature with user. For the development of an android app material design is very useful and provide smooth experience with custom layout,views and animations.
- For this news app user should be able to select from different categories,countries and newspaper.
- News API has been used for collecting different news soures at one spot. On sending request it will give response in JSON format which contains source id,title,description,image URL,article URL,auther,time etc.WE need to handle and parse this JSON into string format which is our reguired format.
- An Android application for keeping up with the latest headlines can be a powerful tool to stay informed and up-to-date with news and information

from around the world. Here are some key features that could be included in such an application:

- News Aggregation: The application can aggregate news from various reputable sources, such as news websites, newspapers, and news agencies, and provide a curated selection of headlines and articles covering different topics and categories.

- Customizable News Categories: Users should have the ability to customize their news categories based on their interests and preferences. They can choose from a wide range of categories such as politics, business, technology, sports, entertainment, health, and more, to receive news that is relevant to their interests.

- Push Notifications: The application can send push notifications for breaking news or important updates, allowing users to receive real-time news alerts even when the app is not actively being used. Users can customize their notification settings based on their preferences and priority of news updates.

- Offline Reading: The application can provide offline reading capabilities, allowing users to access news articles even without an internet connection. This can be useful for users who want to read news on the go or in areas with limited internet access.

- Bookmarking and Saving: Users can bookmark or save news articles for later reading or reference. The application can provide a feature to allow users to save articles, create reading lists, or bookmark articles for easy access and reference.

- Share and Social Media Integration: The application can allow users to share news articles through social media platforms, email, or other communication channels. Integration with popular social media platforms can enable users to easily share interesting articles with their friends and followers.

- Search and Advanced Filtering: The application can provide a search functionality that allows users to search for news articles based on keywords, sources, or categories. Advanced filtering options can help users refine their news feed and discover articles that are most relevant to their interests.

- User Profiles and Personalization: The application can offer user profiles where users can create accounts and customize their news preferences. Personalization features such as recommended articles based on browsing history, liked articles, or saved articles can provide a tailored news experience to individual users.

- Readability and Accessibility: The application can prioritize readability and accessibility by offering options for adjustable font sizes, text-to-speech capabilities, dark mode, and other accessibility features. This can make the news content more accessible to a wide range of users, including those with visual or hearing impairments.

- Trust and Authenticity: The application can prioritize news from reputable sources and provide fact-checking features to ensure the trustworthiness and authenticity of the news content. This can help users identify reliable news sources and combat misinformation.

- In conclusion, an Android application for keeping up with the latest headlines can offer a comprehensive and customizable news experience to users. By incorporating features such as news aggregation, customizable news categories, push notifications, offline reading, bookmarking, social media integration, search and filtering, user profiles and personalization, readability and accessibility, and trust and authenticity, the application can provide an engaging, reliable, and user-friendly platform for staying informed with the latest headlines.

Conclusion

- IIn conclusion, an Android application for keeping up with the latest headlines has a promising future scope, driven by advancements in technology and changing user preferences. With features such as personalized news, multimodal news delivery, real-time news alerts, social media integration, enhanced user interaction, trust and authenticity, localized news, voice and conversational interfaces, data-driven insights, and offline reading and accessibility, the application can offer an engaging, tailored, and immersive news experience to users. The ability to adapt to evolving user needs, incorporate emerging technologies, and provide reliable and credible news content will be crucial for the success of such an application in the future. As news consumption patterns continue to evolve, an Android news application with innovative features, user-centric design, and a focus on trustworthiness can have a significant impact in keeping users informed and engaged with the latest headlines..

Future scope

- The future scope of an Android application for keeping up with the latest headlines is promising, as the demand for news and information continues to grow in our fast-paced digital world. Here are some potential future developments and opportunities for such an application:

- Personalized News: The ability to provide personalized news based on user preferences, browsing history, and location can be a significant enhancement for a news application. Utilizing machine learning algorithms and user feedback, the application could offer tailored news content that matches individual interests, helping users stay informed on topics that matter to them the most.

- Multimodal News Delivery: With advancements in natural language processing and multimedia technologies, the future of news consumption could involve not just text but also images, videos, and audio. An Android application for keeping up with the latest headlines could incorporate multimedia news formats to offer a more immersive and engaging experience for users.

- Real-time News Alerts: Providing real-time news alerts and notifications can be a valuable feature for users who want to stay updated on breaking news or topics of interest. An Android application could leverage push notifications or other notification mechanisms to deliver timely news updates to users, keeping them informed even when they are not actively using the app.

- Social Media Integration: Integrating with social media platforms could allow users to share news articles, engage in discussions, and discover news content through their social networks. By incorporating social media features, an Android news application can foster user engagement and facilitate information sharing among users.

- Enhanced User Interaction: Future news applications could leverage emerging technologies such as augmented reality (AR) or virtual reality (VR) to offer unique and immersive news experiences. For example, users could virtually explore news events or historical locations, visualize data in 3D, or interact with news content in new and innovative ways.

- Trust and Authenticity: In the era of fake news and misinformation, ensuring the trustworthiness and authenticity of news content is critical. Future news applications could incorporate fact-checking features, utilize blockchain technology for content verification, or partner with reputable news organizations to provide credible and reliable news to users.

- Localized News: Providing localized news content based on user's location can be a valuable feature for an Android news application. The ability to deliver news that is relevant to a user's local community or region can enhance user engagement and cater to specific interests and preferences.

- Voice and Conversational Interfaces: With the increasing adoption of voice assistants and conversational interfaces, the future of news applications could involve voice-enabled features. Users could interact with the news application using voice commands, receive news updates through voice assistants, or engage in conversational interactions to discover news content.

Data-driven Insights: Utilizing data analytics and machine learning, an Android news application could provide data-driven insights and trends to users. For example, the application could generate personalized news recommendations, provide visualizations of news data, or offer insights into user's news consumption habits to enhance their news experience.

Offline Reading and Accessibility: Providing offline reading capabilities and accessibility features, such as text-to-speech or adjustable font sizes, could improve the usability and inclusiveness of a news application. These features can cater to users with limited internet access or special accessibility needs, making the application more inclusive and user-friendly.

In conclusion, the future scope of an Android application for keeping up with the latest headlines is vast, with potential advancements in personalization, multimodal news delivery, real-time news alerts, social media integration, enhanced user interaction, trust and authenticity, localized news, voice and conversational interfaces, data-driven insights, and offline reading and accessibility. Embracing these technological advancements and user-centric features can create a more engaging, personalized, and inclusive news experience for users.

APPEDIX

SOURCE CODE

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>
```

ApiService.kt

```
package
com.example.newsh
eadlines
```

```
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=684cb893c
af7425abeffad82ac1d0f4e")
    ///@GET("search?q=chatgpt")
    suspend fun getMovies() :News

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


.addConverterFactory(GsonConverterFactory.create())

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

}
```

Articles.kt

```
package
com.example.example

import com.google.gson.annotations.SerializedName


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

  )
```

DisplayNews.kt

```
package
com.example.newsheadlin
es
```

```
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.NewsHeadlinesT
heme

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

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


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

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

@Preview(showBackground = true)
@Composable
fun DefaultPreview() {
    NewsHeadlinesTheme {
        //   Greeting("Android")
    }
}
@Composable
fun HtmlText(html: String, modifier: Modifier =
Modifier) {
    AndroidView(
        modifier = modifier,
        factory = { context -> TextView(context)
},
        update = { it.text =
HtmlCompat.fromHtml(html,
HtmlCompat.FROM_HTML_MODE_COMPACT) }
```

```
        )
    }
```

LoginActivity.kt

```
package
com.example.newsheadl
ines

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.RoundedCornerShap
e
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.PasswordVisualTransf
ormation
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.NewsHeadlinesThe
me

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

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

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

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

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


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

        }

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

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

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

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



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

        Button(
            onClick = {
                if (username.isNotEmpty() &&
password.isNotEmpty()) {
                    val user =
databaseHelper.getUserByUsername(username)
```

```
                    if (user != null &&
user.password == password) {
                        error = "Successfully log
in"
                        context.startActivity(
                            Intent(
                                context,

MainPage::class.java
                            )
                        )
                        //onLoginSuccess()
                    } else {
                        error = "Invalid username
or password"
                    }
                } else {
                    error = "Please fill all
fields"
                }
            },
            shape = RoundedCornerShape(20.dp),
            colors =
ButtonDefaults.buttonColors(backgroundColor =
Color(0xFF77a2ef)),
            modifier = Modifier.width(200.dp)
            .padding(top = 16.dp)
        ) {
            Text(text = "Log In", fontWeight =
FontWeight.Bold)
        }

        Row(modifier = Modifier.fillMaxWidth()) {
            TextButton(onClick = {
                context.startActivity(
                    Intent(
                        context,

RegistrationActivity::class.java
                    ))})
            { Text(text = "Sign up",
                color = Color.Black
            )}
            Spacer(modifier =
Modifier.width(100.dp))

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




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

MainPage.kt

```
package
com.example.newsheadlin
es

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.RoundedCornerSh
ape
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.NewsHeadlinesT
heme

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

```
                Surface(color =
MaterialTheme.colors.background) {
                    Column() {


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


MovieList(applicationContext, movieList =
mainViewModel.movieListResponse)

mainViewModel.getMovieList()
                    }
                }
            }
        }
    }
}

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

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

}

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


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

@Composable
fun MovieItem(context: Context, movie: Articles,
index: Int, selectedIndex: Int,
              onClick: (Int) -> Unit)
{

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

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

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

            )
            {
                Image(
                    painter =
rememberImagePainter(
                        data = movie.urlToImage,
                        builder = {
                            scale(Scale.FILL)

placeholder(R.drawable.placeholder)

transformations(CircleCropTransformation())
                        }
                    ),
                    contentDescription =
movie.description,
                    modifier = Modifier
                        .fillMaxHeight()
                        .weight(0.3f)
                )


                Column(
                    verticalArrangement =
Arrangement.Center,
                    modifier = Modifier
                        .padding(4.dp)
                        .fillMaxHeight()
                        .weight(0.8f)
                        .background(Color.Gray)
                        .padding(20.dp)
                        .selectable(true, true,
null,
                            onClick = {

Log.i("test123abc", "MovieItem:
$index/n${movie.description}")

context.startActivity(

Intent(context, DisplayNews::class.java)

.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK)
.putExtra("desk", movie.description.toString())

.putExtra("urlToImage", movie.urlToImage)

.putExtra("title", movie.title)
                                )
                            })
                ) {

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

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

MainViewModel.kt

```
package
com.example.newsheadlines
```

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

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

Model.kt

```
package
com.example.newsheadlines
```

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

News.kt

```
package
com.example.newsheadlines

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


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

RegistrationActivity.kt

```
package
com.example.newsheadl
ines

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.RoundedCornerShap
e
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.PasswordVisualTransf
ormation
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.NewsHeadlinesThe
me

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



RegistrationScreen(this,databaseHelper)
                }
            }
        }
```

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

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

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

        }

        Image(
            painter = painterResource(id =
R.drawable.sign_up),
            contentDescription = "",
            modifier = Modifier.height(270.dp)
        )
        TextField(
            value = username,
            onValueChange = { username = it },
            leadingIcon = {
                Icon(
                    imageVector =
Icons.Default.Person,
                    contentDescription =
"personIcon",
                    tint = Color(0xFF6495ED)
                )
            },
            placeholder = {
                Text(
                    text = "username",
                    color = Color.Black
                )
            },
            colors =
TextFieldDefaults.textFieldColors(
                backgroundColor = Color.Transparent
            )

        )

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

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


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


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

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

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

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

LoginActivity::class.java
                        )
                    )

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

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


            Text(text = "Have an account?")

            TextButton(onClick = {
                context.startActivity(
                    Intent(
                        context,
                        LoginActivity::class.java
                    )
                )
            }) {
                Text(text = "Log in",
                    fontWeight = FontWeight.Bold,
                    style =
MaterialTheme.typography.subtitle1,
                    color = Color(0xFF4285F4)
                )}

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

Source.kt

```
package
com.example.example
```

```
import com.google.gson.annotations.SerializedName


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

)
```

User.kt

```
package
com.example.newsheadlines


import androidx.room.ColumnInfo

import androidx.room.Entity

import androidx.room.PrimaryKey


@Entity(tableName = "user_table")

data class User(

    @PrimaryKey(autoGenerate = true) val id:
Int?,

    @ColumnInfo(name = "first_name") val
firstName: String?,

    @ColumnInfo(name = "last_name") val
lastName: String?,

    @ColumnInfo(name = "email") val email:
String?,

    @ColumnInfo(name = "password") val
password: String?,


    )
```

UserDao.kt

```
package
com.example.newsheadlines

import androidx.room.*
```

```
@Dao
interface UserDao {

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

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

    @Update
    suspend fun updateUser(user: User)

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

UserDatabase.kt

```
package
com.example.newsheadlines


import android.content.Context

import androidx.room.Database

import androidx.room.Room

import androidx.room.RoomDatabase


@Database(entities = [User::class], version =
1)

abstract class UserDatabase : RoomDatabase() {


    abstract fun userDao(): UserDao
    companion object {


        @Volatile

        private var instance: UserDatabase? =
null


        fun getDatabase(context: Context):
UserDatabase {

            return instance ?:
synchronized(this) {

                val newInstance =
Room.databaseBuilder(

                    context.applicationContext,

                    UserDatabase::class.java,

                    "user_database"

                ).build()

                instance = newInstance

                newInstance

            }

        }

    }

}
```

Fo