Android App Development

Contents

1 INTRODUCTION

1.1 Overview
1.2Purpose

2 Problem Definition & Design Thinking

2.1Empathy Map
2.2Ideation & Brainstorming Map

3 RESULT

4 ADVANTAGES & DISADVANTAGES

5 APPLICATIONS

6 CONCLUSION

7 FUTURE SCOPE

8 APPENDIX

A. Source Code

Overview:

Android Application for Keeping Up with the Latest Headlines :

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. This 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.

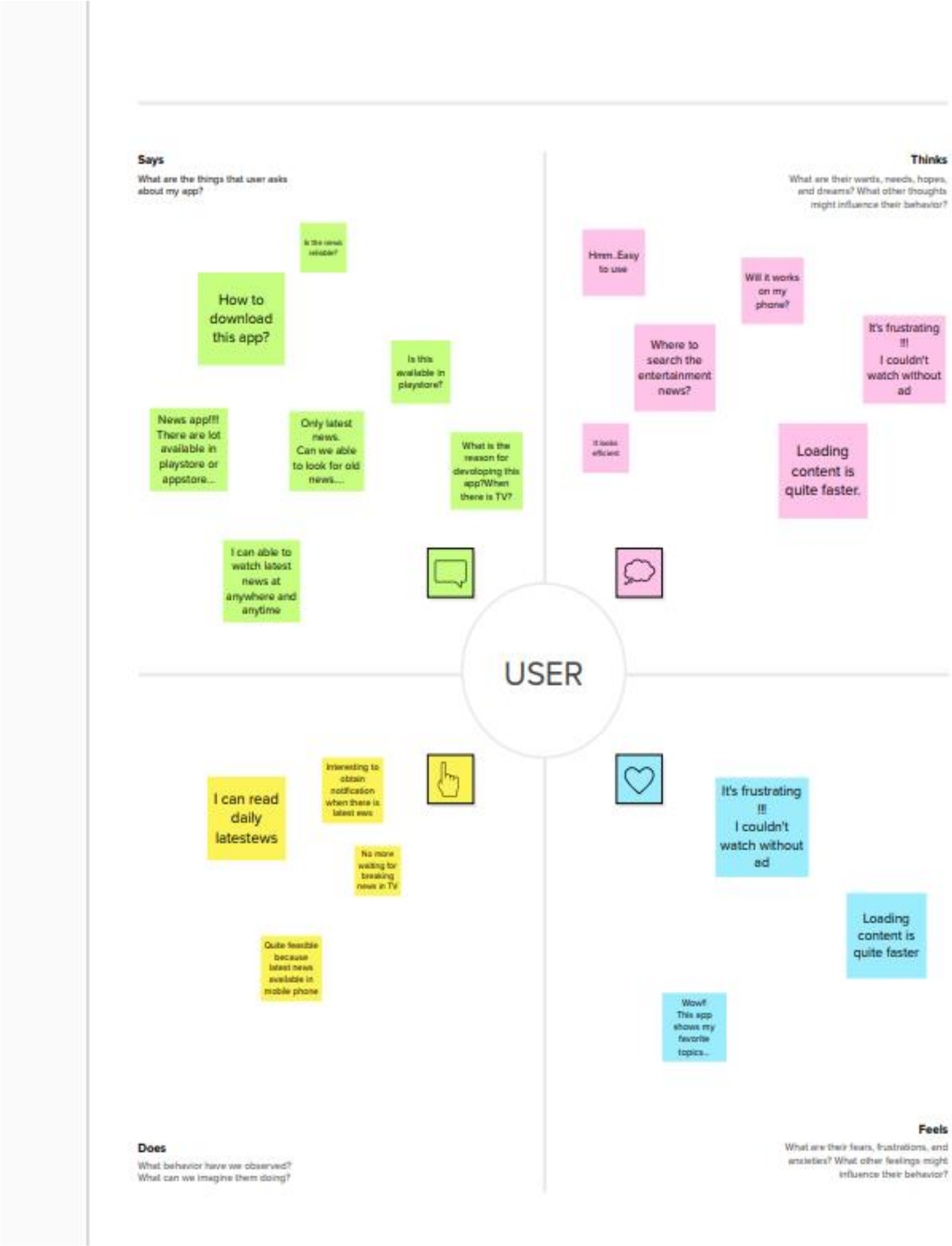
As world's technology is rapidly growing we has fast connection and network to instantly connect to other person. Day to day use in mobile, tablets and laptop is increasing, most of the people already have this facilities. In this fast and information oriented world we need to stay updated with every incidents and news too. This News app is android mobile application where user have access to latest news from 120+ newspapers from 50+ countries. The main focus of this application is to connect news articles from all around the world and deliver it to user as fast as possible in best visualize way.

Purpose:

The purpose is to develop an android application, which will eliminate the problems faced in the current scenario. This application will provide all the information and news related to cybersecurity, E-sport, Science, and Technology or that are in trend at one place. So, it will save time and effort of the users by making it more efficient. Using, this application will terminate the possibility of information redundancy

Problem Definition & Design Thinking

Empathy map:



Says

What are the things that user asks about my app?

Is the news reliable?

How to download this app?

Is this available in playstore?

News app!!! There are lot available in playstore or appstore...

Only latest news. Can we able to look for old news....

What is the reason for developing this app?When there is TV?

I can able to watch latest news at anywhere and anytime

Thinks

What are their wants, needs, hopes, and dreams? What other thoughts might influence their behavior?

Hmm..Easy to use

Will it works on my phone?

Where to search the entertainment news?

It's frustrating !!! I couldn't watch without ad

It looks efficient

Loading content is quite faster.

USER

I can read daily latestews

Interesting to obtain notification when there is latest ews

No more waiting for breaking news in TV

Quite feasible because latest news available in mobile phone

It's frustrating !!! I couldn't watch without ad

Loading content is quite faster

Wow! This app shows my favorite topics...

Does

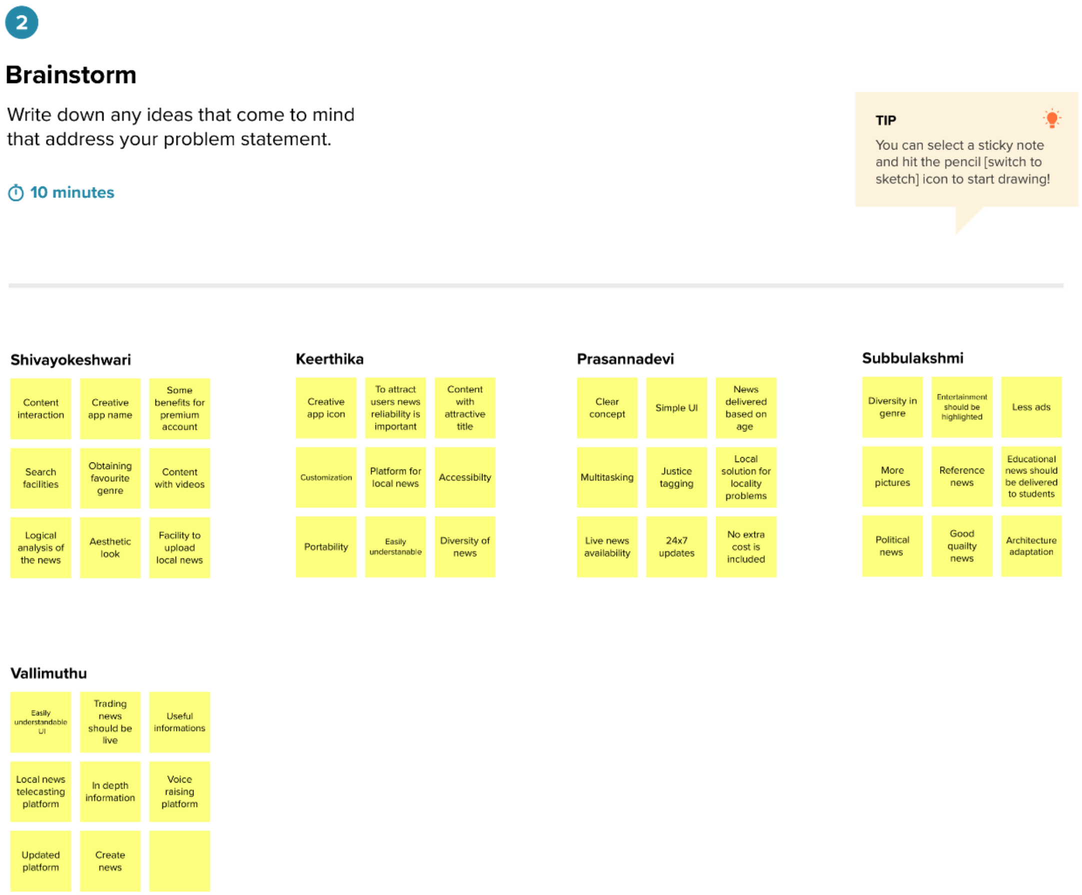
What behavior have we observed? What can we imagine them doing?

Feels

What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

Brainstorming and Ideation:

Ideas:



2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

⏱ 10 minutes

TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Shivayokeshwari

Content interaction	Creative app name	Some benefits for premium account
Search facilities	Obtaining favourite genre	Content with videos
Logical analysis of the news	Aesthetic look	Facility to upload local news

Keerthika

Creative app icon	To attract users news reliability is important	Content with attractive title
Customization	Platform for local news	Accessibilty
Portability	Easily understanable	Diversity of news

Prasannadevi

Clear concept	Simple UI	News delivered based on age
Multitasking	Justice tagging	Local solution for locality problems
Live news availability	24x7 updates	No extra cost is included

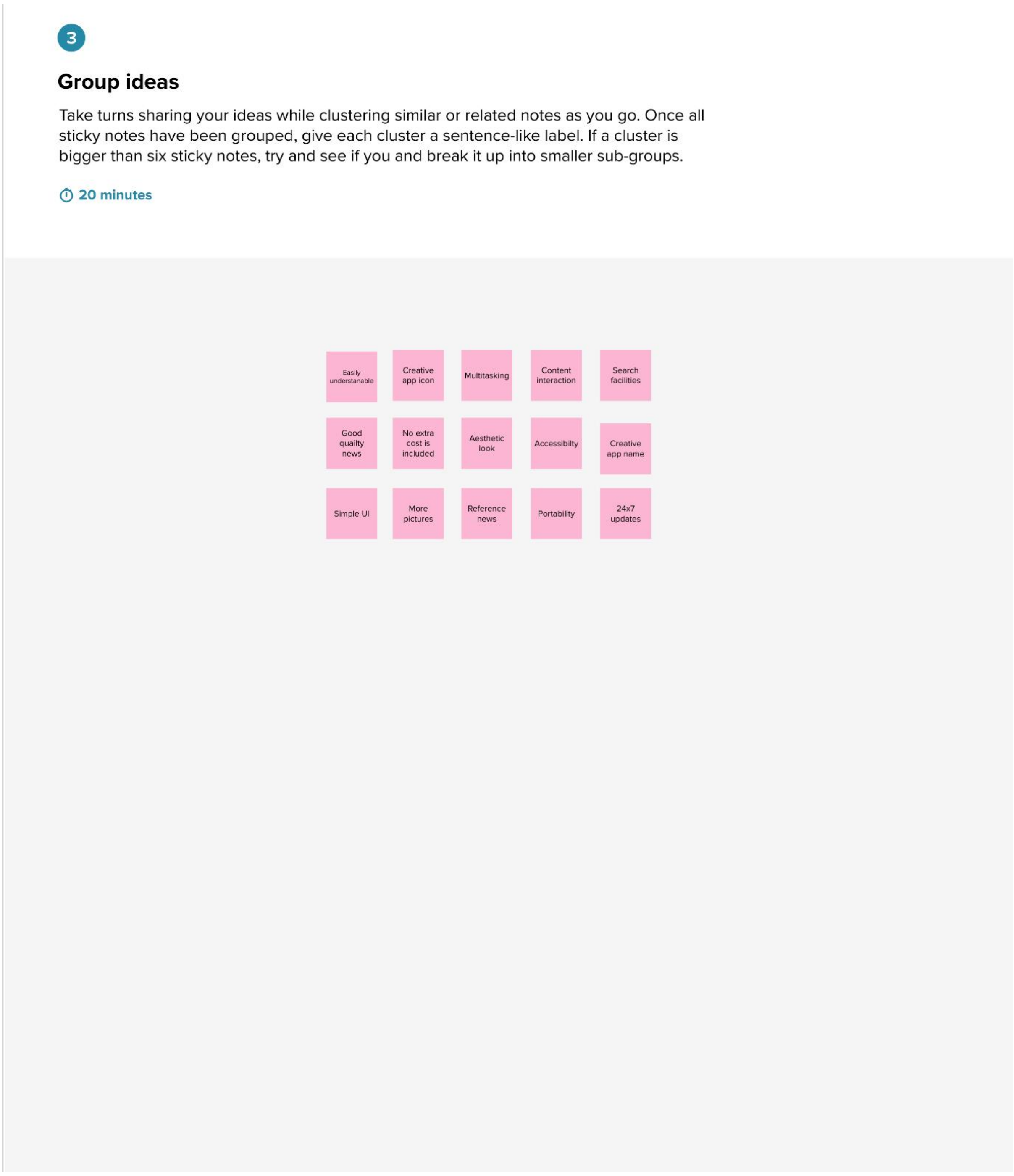
Subbulakshmi

Diversity in genre	Entertainment should be highlighted	Less ads
More pictures	Reference news	Educational news should be delivered to students
Political news	Good quailty news	Architecture adaptation

Vallimuthu

Easily understandable UI	Trading news should be live	Useful informations
Local news telecasting platform	In depth information	Voice raising platform
Updated platform	Create news	

Group Ideas:



3

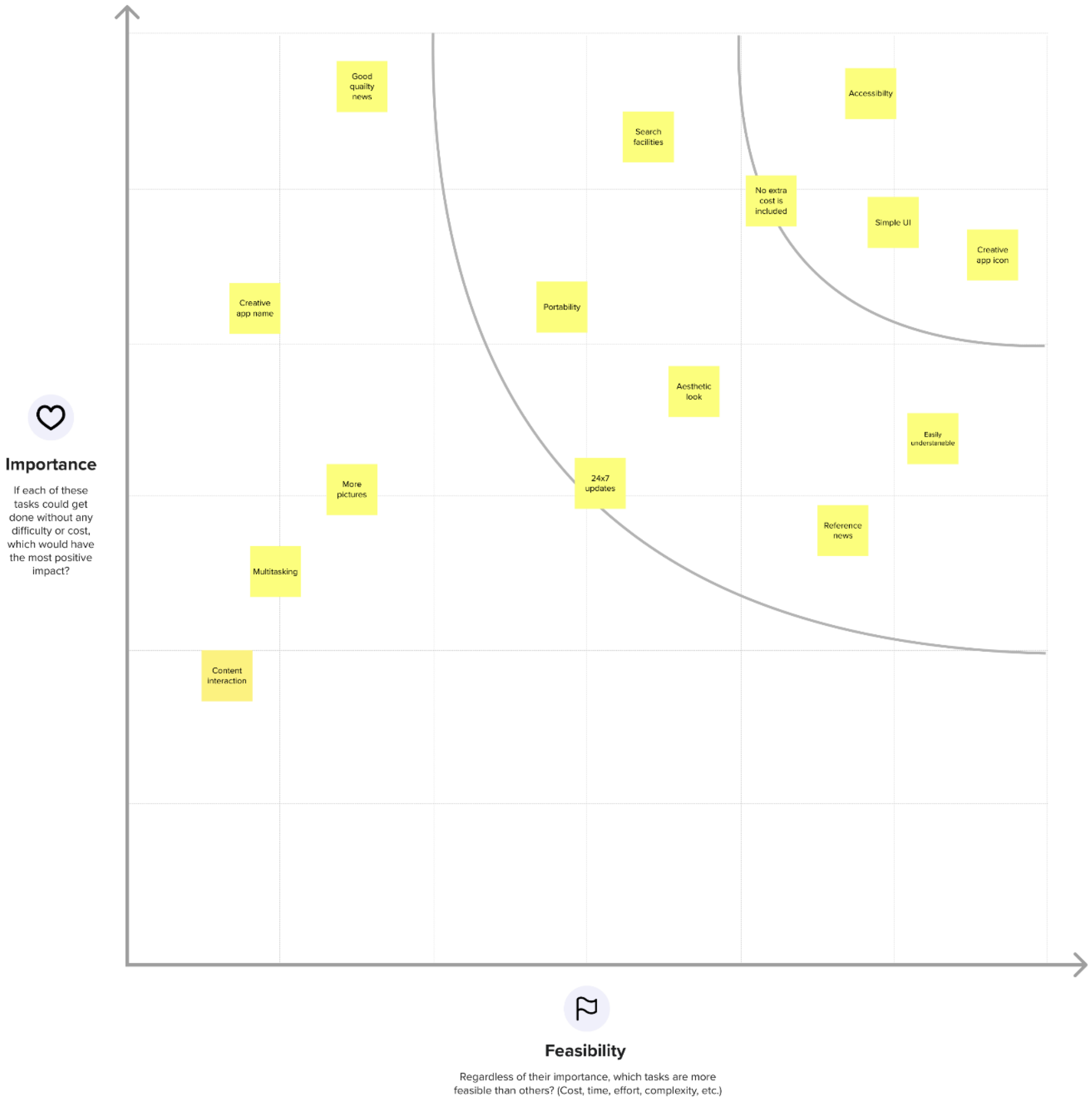
Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

⏱ 20 minutes

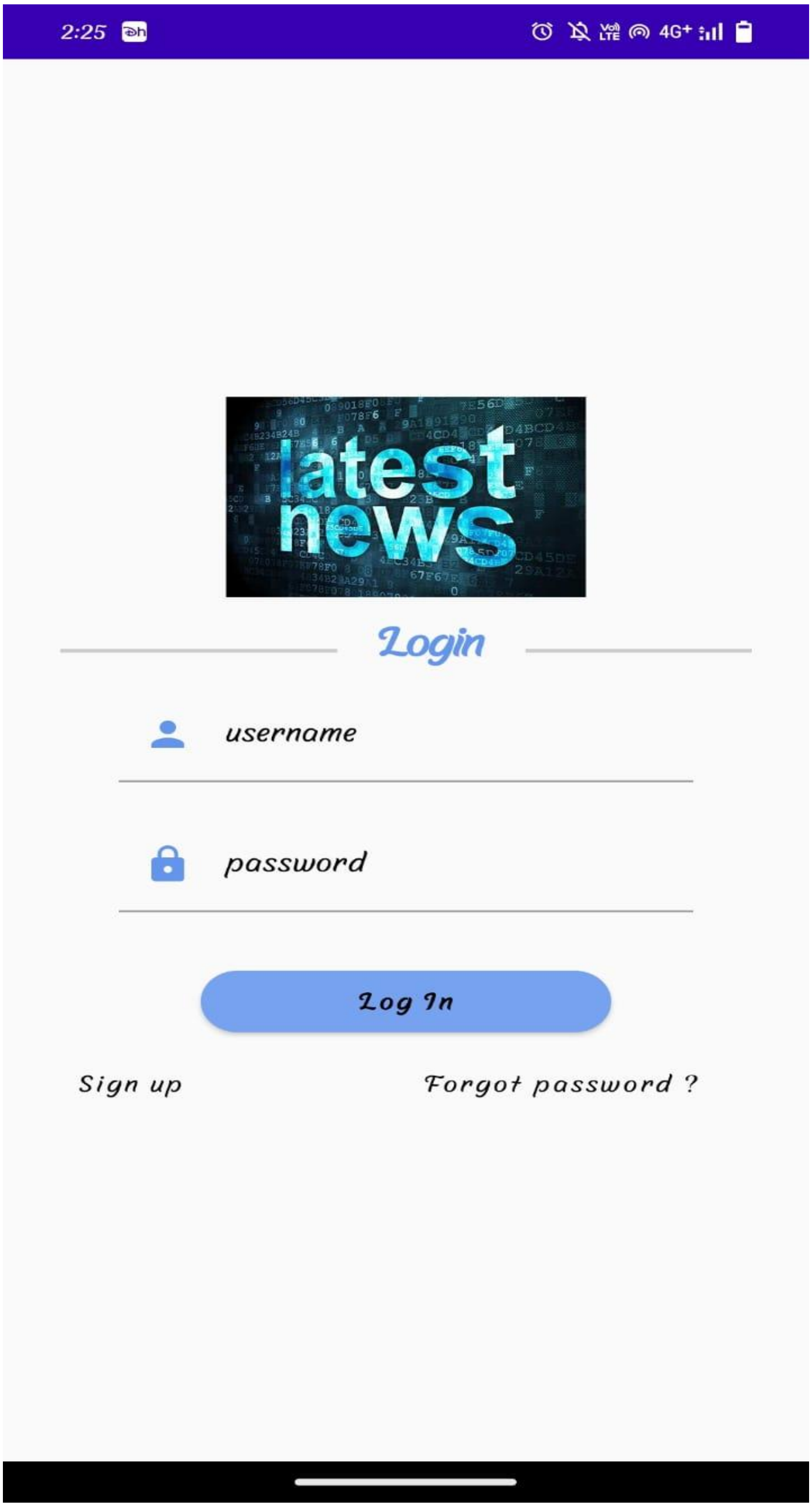
Easily understanable	Creative app icon	Multitasking	Content interaction	Search facilities
Good quailty news	No extra cost is included	Aesthetic look	Accessibilty	Creative app name
Simple UI	More pictures	Reference news	Portability	24x7 updates

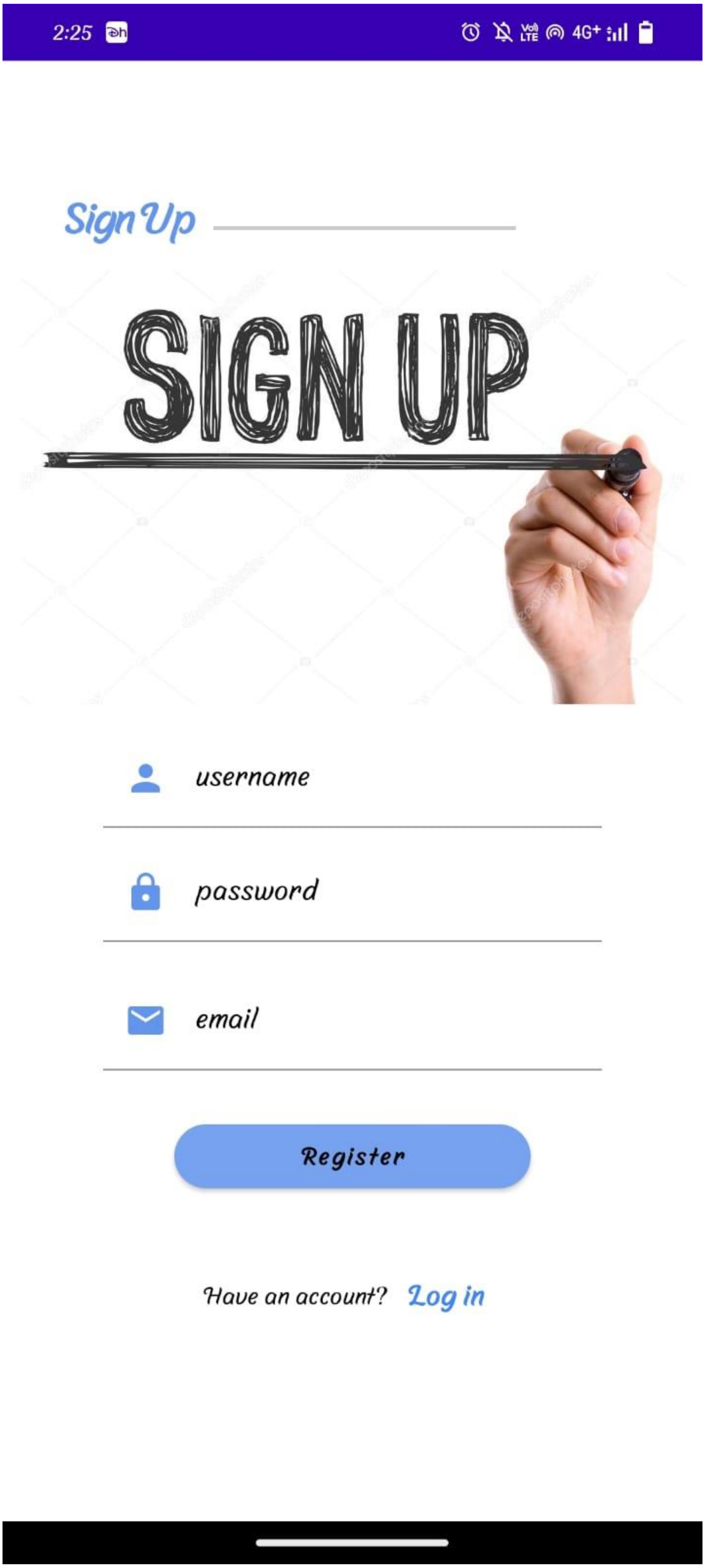
Prioritize:


Good quailty news
Search facilities
Accessibilty
No extra cost is included
Simple UI
Creative app icon
Creative app name
Portability
Aesthetic look
Easily understanable
More pictures
24x7 updates
Reference news
Multitasking
Content interaction
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

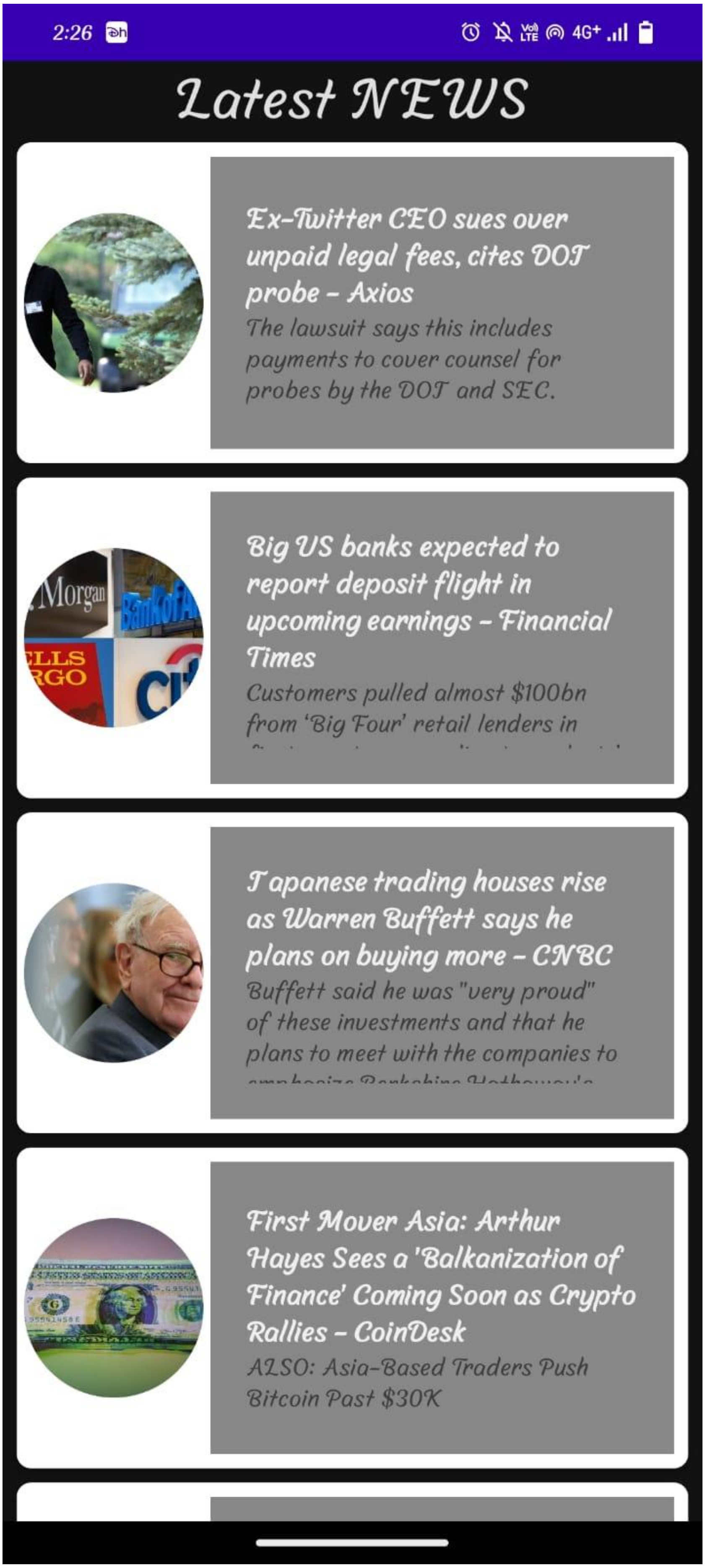
RESULT:

Screen Shots:





Have an account? Log in



Latest NEWS

Ex-Twitter CEO sues over unpaid legal fees, cites DOJ probe - Axios

The lawsuit says this includes payments to cover counsel for probes by the DOJ and SEC.

Big US banks expected to report deposit flight in upcoming earnings - Financial Times

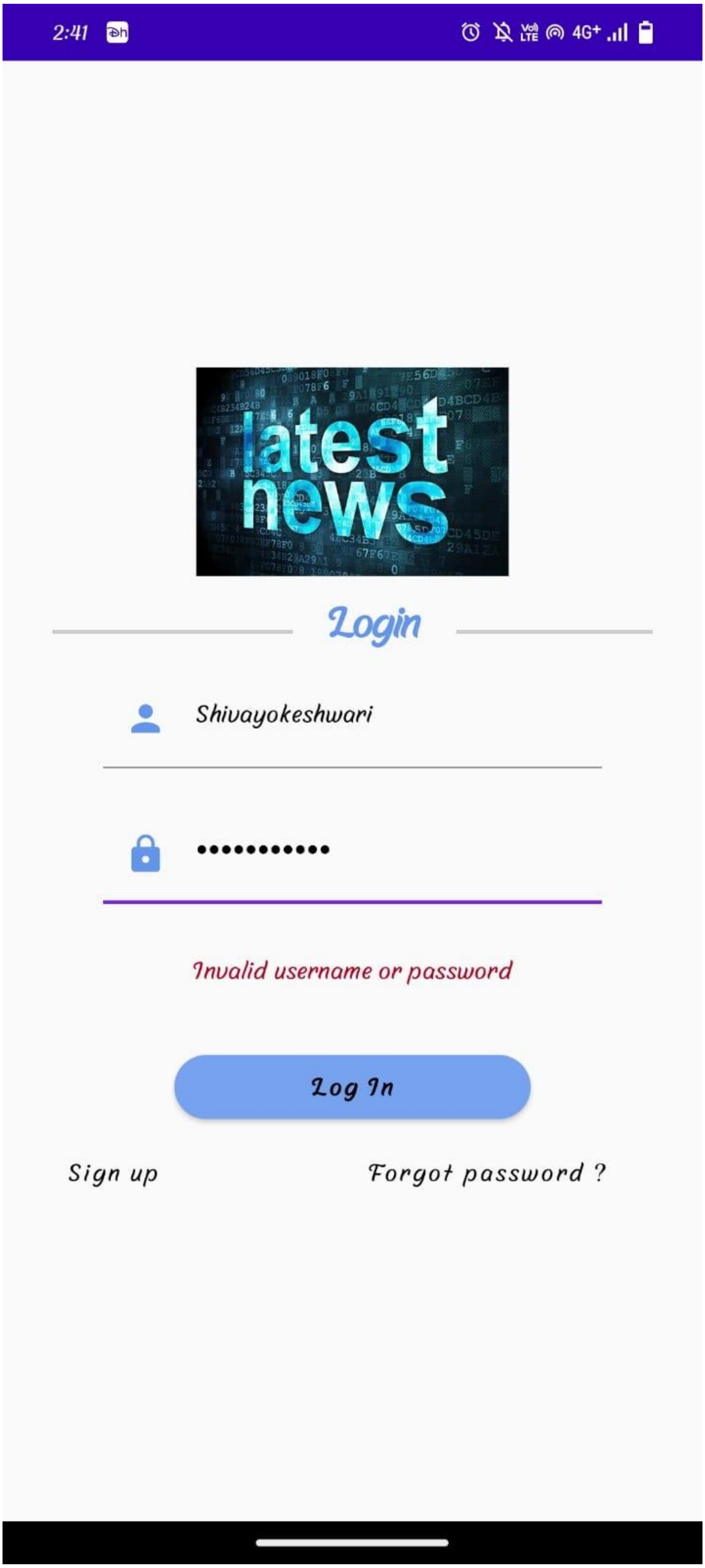
Customers pulled almost $100bn from 'Big Four' retail lenders in

Japanese trading houses rise as Warren Buffett says he plans on buying more - CNBC

Buffett said he was "very proud" of these investments and that he plans to meet with the companies to emphasize Berkshire Hathaway's

First Mover Asia: Arthur Hayes Sees a 'Balkanization of Finance' Coming Soon as Crypto Rallies - CoinDesk

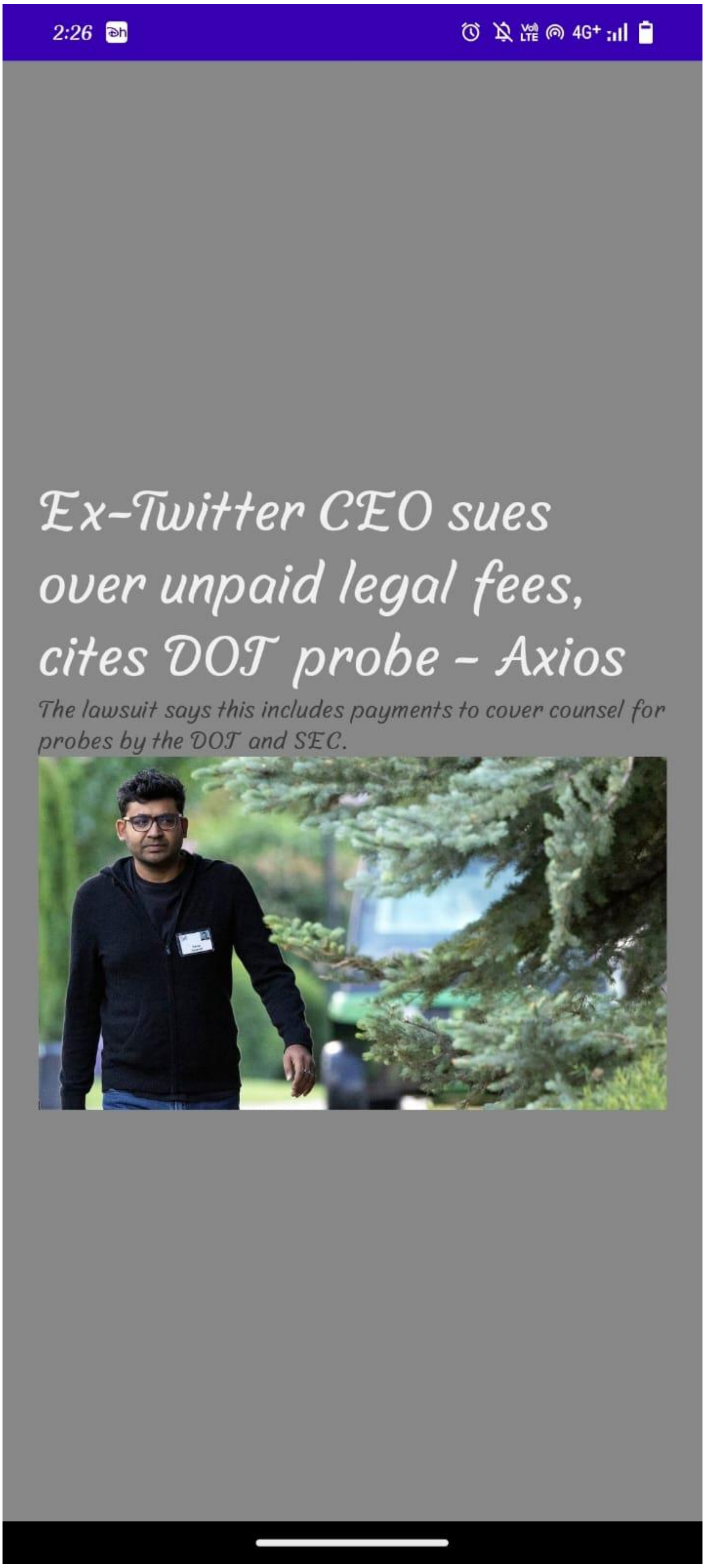
ALSO: Asia-Based Traders Push Bitcoin Past $30K



Shivayokeshwari

●●●●●●●●●●●

Invalid username or password



Ex-Twitter CEO sues over unpaid legal fees, cites DOJ probe - Axios

The lawsuit says this includes payments to cover counsel for probes by the DOJ and SEC.

Big US banks expected to report deposit flight in upcoming earnings - Financial Times

Customers pulled almost $100bn from 'Big Four' retail lenders in first quarter, according to analysts' forecasts

Advantages:

- News apps offer a great reading experience and simultaneously display the ads helpfully so they don't annoy users and distract them from the focus on the content they are reading, yet showing the ads simultaneously.

Disadvantages:

- Require data/wifi to get online.

- Companies not making as much money due to free reading for audiences.

Application:

Newspaper is one of the most popular and required assets of our daily lives. And, in today’s hectic world, reading newspaper has become one of the traditional ways of reading the news. With the news being created every minute and relayed through TV, radio and internet, the updated news is already old by the next morning. And, that’s why newspaper and magazine publishers are struggling to keep-up with the pace. Change is needed and publishers must embrace mobile.

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 is quite simple and rewarding. They run ads and generate a good amount of money.

Conclusion:

In this provided system we created a basic system of displaying news and headlines fetched from the api. In future enhancement we are eager to add our proposed ideas dicussed in the brainstorming session.

Future Enhancement:

Features to upload our own article will be added.

Location feature with automation can be implemented which means as user move from one city to other local news will change as per it. Offline Reading can be improve will more efficient way on full articles. Data quality check needed. If API can’t reach to certain article source it gives null value which can cause problem in JSON parsing.

Appendix:

Source code:

Build gradle

```
dependencies {


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

    // Room Database

    implementation 'androidx.room:room-common:2.5.0'
    // Retrofit

    implementation 'com.squareup.retrofit2:retrofit:2.9.0'

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

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

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">
    <!-- permissions -->
    <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=".MainPage"
            android:exported="false"
            android:label="@string/title_activity_main_page"
            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=".LoginActivity"
            android:exported="true"
            android:label="News Headlines"
            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.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=8a00c7a22fd14797b8c
2ba3f89b73c82")
    ///@GET("search?q=chatgpt")
    suspend fun getMovies() :News

    companion object {
        var apiService: ApiService? = null
        fun getInstance() : ApiService {
            if (apiService == null) {
                apiService = Retrofit.Builder()
                    // .baseUrl("https://howtodoandroid.com/apis/")
                    .baseUrl("https://newsapi.org/v2/")
                    //.baseUrl("https://podcast-episodes.p.rapidapi.com/")
                    .addConverterFactory(GsonConverterFactory.create())
                    .build().create(ApiService::class.java)
            }
            return apiService!!
        }
    }

}
```

Articles.kt

```
package com.example.example

import com.google.gson.annotations.SerializedName


data class Articles (

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

    )
```

DisplayNews.kt

```
package com.example.newsheadlines

import android.content.Intent
import android.os.Bundle
import android.util.Log
import android.widget.TextView
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.compose.ui.viewinterop.AndroidView
import androidx.core.text.HtmlCompat
import coil.compose.rememberImagePainter
import com.example.newsheadlines.ui.theme.NewsHeadlinesTheme

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

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

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

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

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

LoginActivity.kt

```
package com.example.newsheadlines

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

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

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

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

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

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


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

        }

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

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

        )


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

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



        Spacer(modifier = Modifier.height(12.dp))
        if (error.isNotEmpty()) {
```

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

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

    Row(modifier = Modifier.fillMaxWidth()) {
        TextButton(onClick = {
            context.startActivity(
                Intent(
                    context,
                    RegistrationActivity::class.java
                ))})
        { Text(text = "Sign up",
            color = Color.Black
        )}

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

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




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

MainPage.kt

```
package com.example.newsheadlines

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

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


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

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

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

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

}

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


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

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

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

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

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

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


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

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

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

MainViewModel

```
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

```
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
        ) {


          Text(text = "Have an account?")

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

        }
```

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

Source

```
package com.example.example

import com.google.gson.annotations.SerializedName


data class Source (

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

)
```

User

```
package com.example.newsheadlines

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

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

    )
```

UserDao

```
package com.example.newsheadlines

import androidx.room.*

@Dao
interface UserDao {

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

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

    @Update
    suspend fun updateUser(user: User)

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

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")
    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
    }

}
```