NEWS HEADLINES

INTRODUCTION

1.1 OVERVIEW

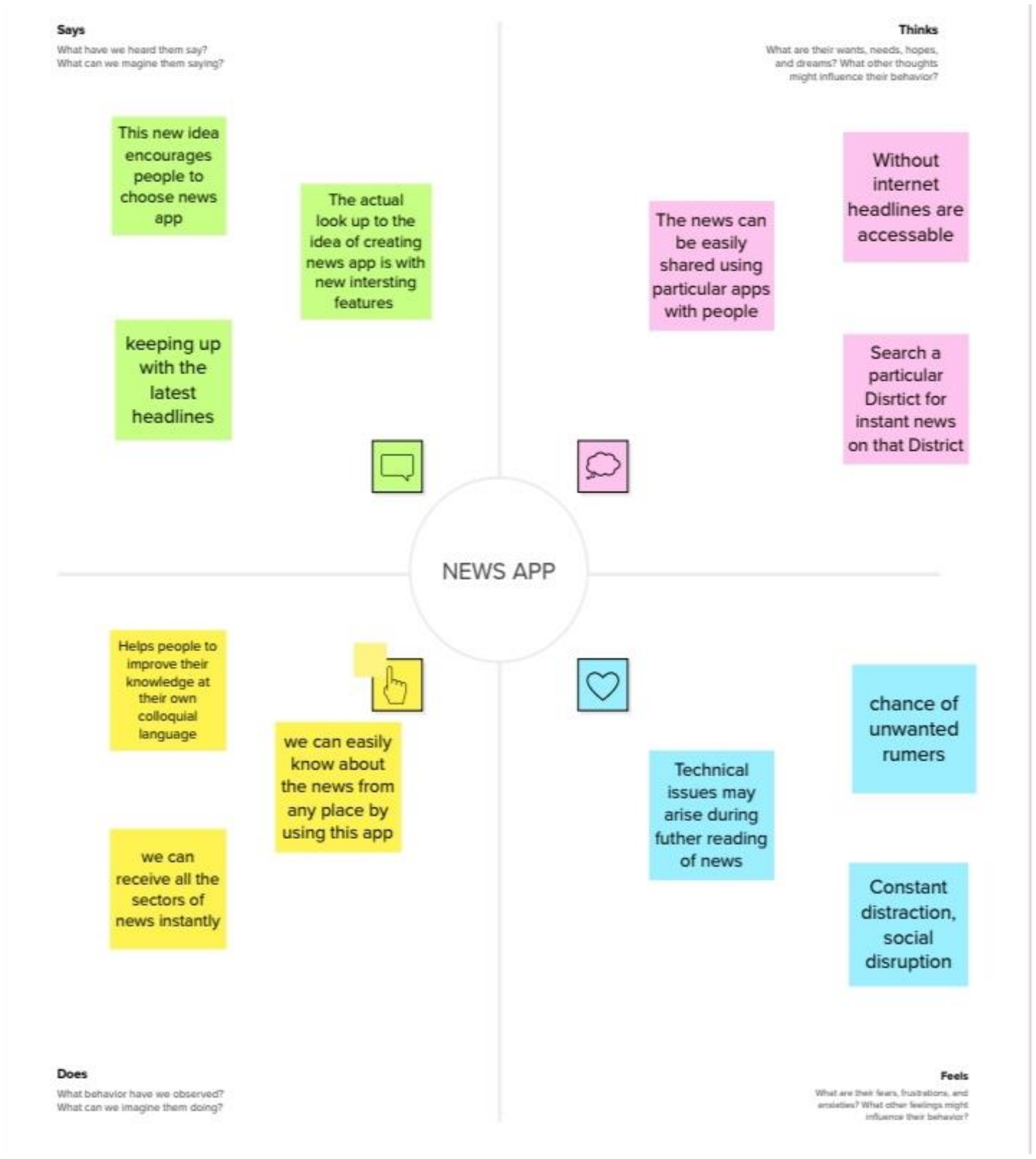
The app's main feature is displaying a list of news articles, each with a title, image, and brief description. Users can scroll through the list of articles and tap on an article to view more details. The app uses the Jetpack Compose UI toolkit to build the UI and it uses the coil library to load images. 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

A news application is a big interactive database that tells a news story. Think of it like you would any other piece of journalism. It just uses software instead of words and pictures.

PROBLEM DEFINITION & DESIGN THINKING

2.1 EMPATHY MAP



Says

What have we heard them say?
What can we magine them saying?

This new idea encourages people to choose news app

The actual look up to the idea of creating news app is with new intersting features

keeping up with the latest headlines

Thinks

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

Without internet headlines are accessable

The news can be easily shared using particular apps with people

Search a particular Disrtict for instant news on that District

NEWS APP

Does

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

Helps people to improve their knowledge at their own colloquial language

we can easily know about the news from any place by using this app

we can receive all the sectors of news instantly

Feels

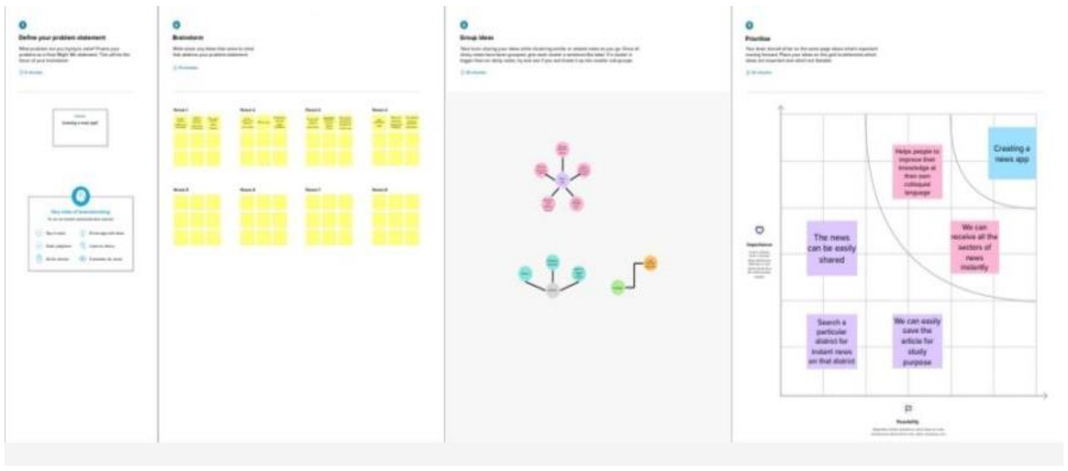
What are their fears, frustrations, and anxieties? What other feelings might influence their behavior?

chance of unwanted rumers

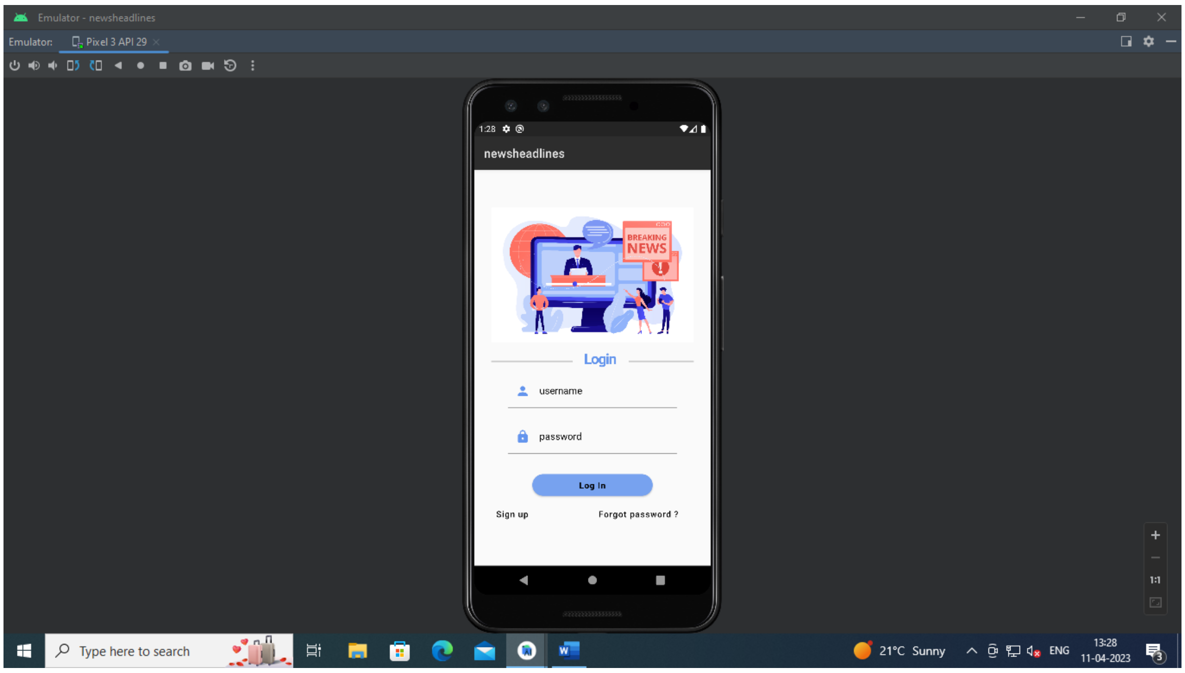
Technical issues may arise during futher reading of news

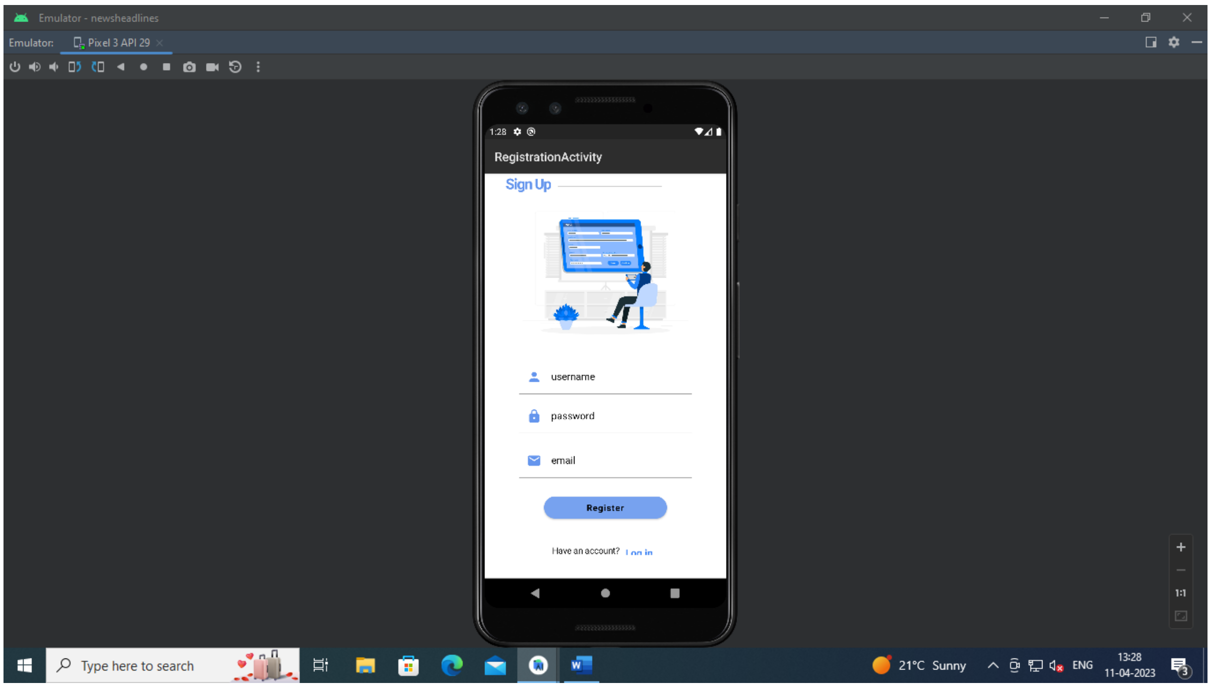
Constant distraction, social disruption

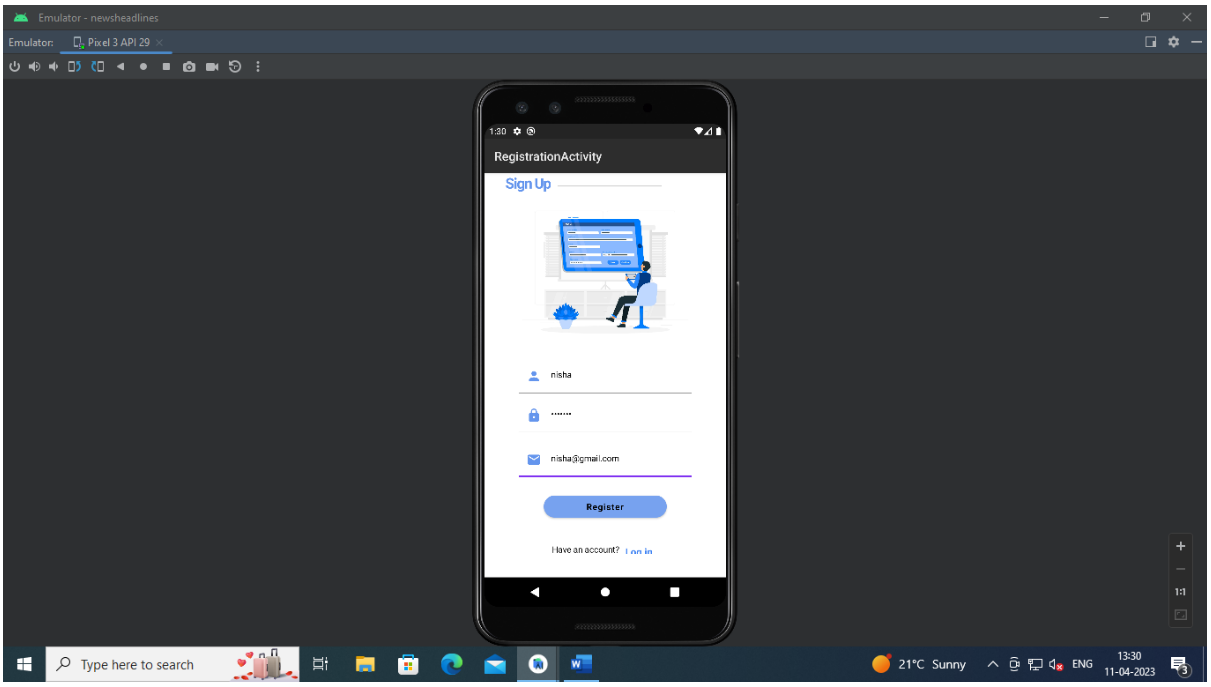
2.2 IDEATION & BRAINSTROMING MAP


Define your problem statement
Brainstorm
Group ideas
Prioritize
Helps people to improve their knowledge at their own colloquial language
Creating a news app
The news can be easily shared
We can receive all the sectors of news instantly
Search a particular district for instant news on that district
We can easily save the article for study purpose
Importance
Feasibility

RESULT


Emulator - newsheadlines
Emulator: Pixel 3 API 29
newsheadlines
BREAKING NEWS
Login
username
password
Log In
Sign up
Forgot password ?
Type here to search
21°C Sunny
ENG
13:28
11-04-2023


Emulator - newsheadlines
Emulator: Pixel 3 API 29
RegistrationActivity
Sign Up
username
password
email
Register
Have an account? Log in
Type here to search
21°C Sunny
ENG
13:28
11-04-2023



Emulator - newsheadlines

Emulator: Pixel 3 API 29

RegistrationActivity

Sign Up

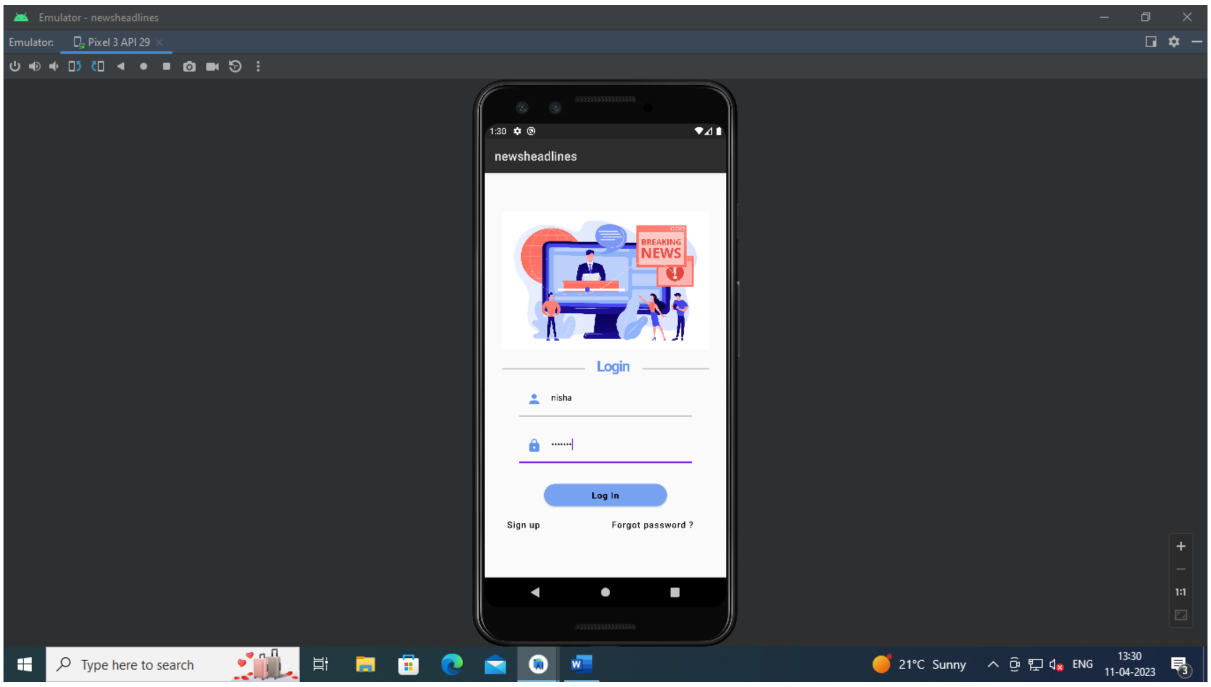
nisha

•••••••

nisha@gmail.com

Register

Have an account? Log in



Emulator - newsheadlines

Emulator: Pixel 3 API 29

newsheadlines

Login

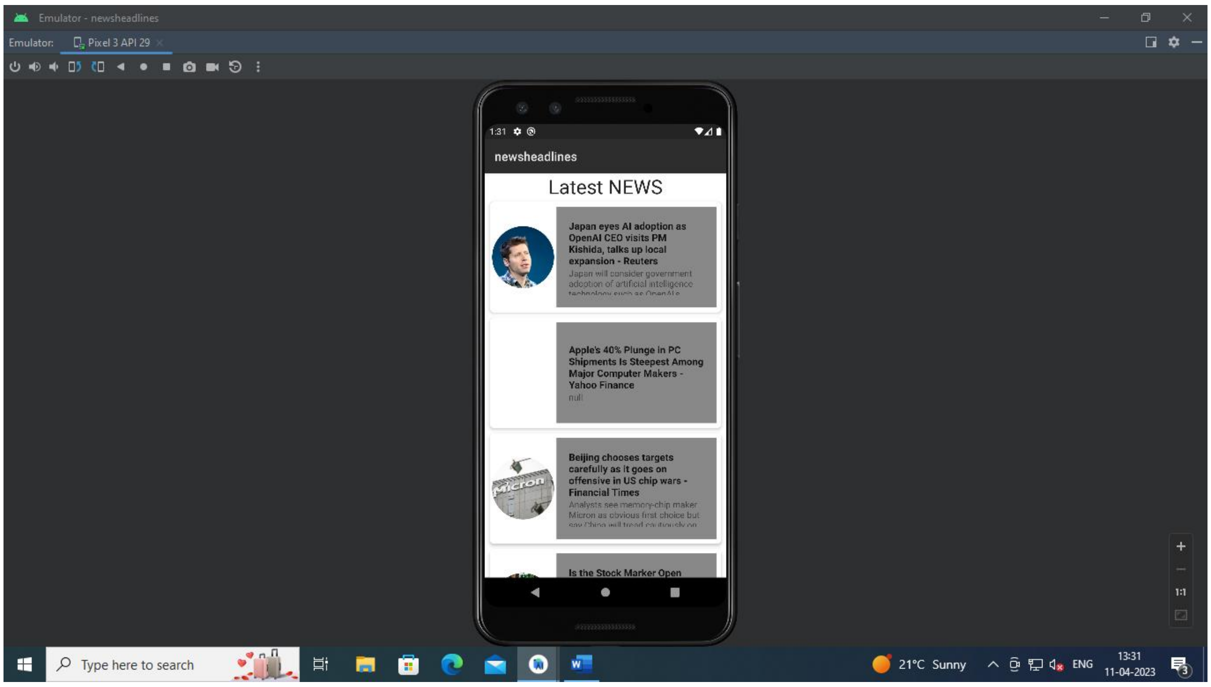
nisha

•••••••

Log In

Sign up

Forgot password ?



newsheadlines

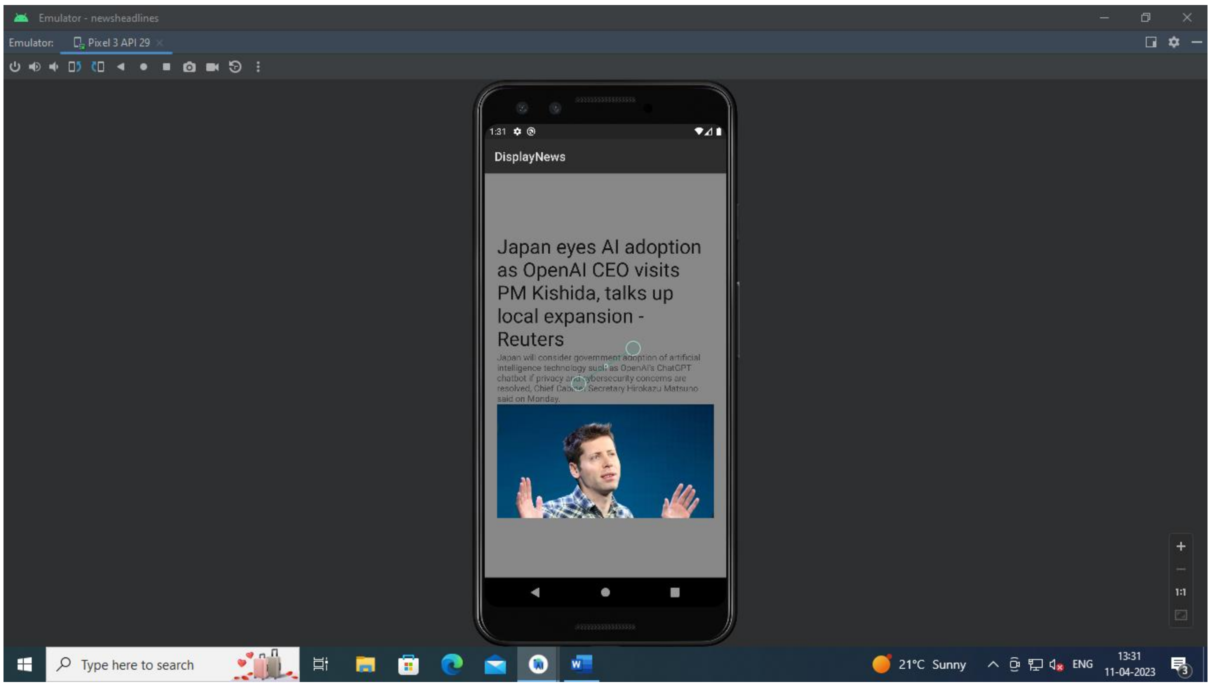
Latest NEWS

Japan eyes AI adoption as OpenAI CEO visits PM Kishida, talks up local expansion - Reuters
Japan will consider government adoption of artificial intelligence technology such as OpenAI's

Apple's 40% Plunge in PC Shipments Is Steepest Among Major Computer Makers - Yahoo Finance
null

Beijing chooses targets carefully as it goes on offensive in US chip wars - Financial Times
Analysts see memory-chip maker Micron as obvious first choice but say China will tread cautiously on

Is the Stock Marker Open



DisplayNews

Japan eyes AI adoption as OpenAI CEO visits PM Kishida, talks up local expansion - Reuters

Japan will consider government adoption of artificial intelligence technology such as OpenAI's ChatGPT chatbot if privacy and cybersecurity concerns are resolved, Chief Cabinet Secretary Hirokazu Matsuno said on Monday.

ADVANTAGES & DISADVANTAGES

ADVANTAGES

1. we can easily know about the news for many places by using this app
2. Helps people to improve their knowledge at their own colloquial language
3. we can receive all the sectors of news instantly
4. Without internet headlines are accessible
5. Search a particular district for instant news on that district

DISADVANTAGES

1. chance of unwanted roomers
2. Constant distraction, social disruption
3. Technical issues may arise during further reading of news constant

APPLICATION

1. General public
2. Marketing team

CONCLUSION

The headline tells them what to expect, the introduction eases them into the content, the subheadings help fill out their understanding of the content, and the conclusion wraps up the piece. In many cases, they can learn most of what they want to know from the introduction and conclusion alone.

FUTURE SCOPE

A news app needs to be steered with precision. Today’s digital marketing tools allow publishers to select multiple parameters monitoring the use of an application: They can measure how long the app is used, when, for how long, why and where people tend to drop it, what kind of news they like, if they hit a paywall and give up, and why they do so

APPENDIX

SOURCE CODE

Gradle Scripts

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

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

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

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

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

dependencies {

    implementation 'androidx.core:core-ktx:1.7.0'
    implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.3.1'
    implementation 'androidx.activity:activity-compose:1.3.1'
    implementation "androidx.compose.ui:ui:$compose_ui_version"
    implementation "androidx.compose.ui:ui-tooling-
preview:$compose_ui_version"
    implementation 'androidx.compose.material:material:1.2.0'
    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"
    implementation 'androidx.room:room-common:2.5.0'

    implementation 'androidx.room:room-ktx:2.5.0'
    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")
}
```

Data base class
User Dao

```
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)
}
```

User Database

```
package com.example.newsheadlines

import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase
```

```kotlin
@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
            }
        }
    }
}
```

User Database helper

```kotlin
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
    }

}
```

Creating API services

```
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=684cb893caf7425abeffad82ac1d0
f4e")
    ///@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!!
        }
    }

}
```

Creating Main view model

```
package com.example.newsheadlines

import android.util.Log
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.setValue
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.newsheadlines.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()
            }
        }
    }
}
```

Building Application UI
Login Activity

```
package com.example.newsheadlines

import android.content.Context
import android.content.Intent
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material.*
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Lock
import androidx.compose.material.icons.filled.Person
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.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(
```

```kotlin
            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)
}
```

Registration Activity

```kotlin
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)
}
```

Main page

```
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.newsheadlines.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) }
        )
    }
}
```

Display news

```
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) }
    )
}
```

Android Manifest

```
<?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"

        tools:targetApi="31">
        <activity
            android:name=".DisplayNews"
            android:exported="false"
            android:label="@string/title_activity_display_news"
            />
        <activity
            android:name=".RegistrationActivity"
            android:exported="false"
            android:label="@string/title_activity_registration"
             />
        <activity
            android:name=".MainPage"
            android:exported="false"
             />
        <activity
            android:name=".LoginActivity"
            android:exported="true"
            android:label="@string/app_name"
            >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```