

An Android Application

- Keeping up with the Latest HeadLines

Mentor: Dr.T. Arul Raj M.Sc., M.Phil., Ph.D

Developed By:

Sri Paramakalyani College (Code-123), Alwarkurichi



DEPARTMENT OF COMPUTER SCIENCE

Team Leader:

Celciya. I (20201231506208)

Team Members:

Meera Jasmine. S (20201231506225)

Reshma. S (20201231506230)

Karthika Lakshmi. S (20201231506218)

Surya. P (20201231506243)

TABLE OF CONTENT

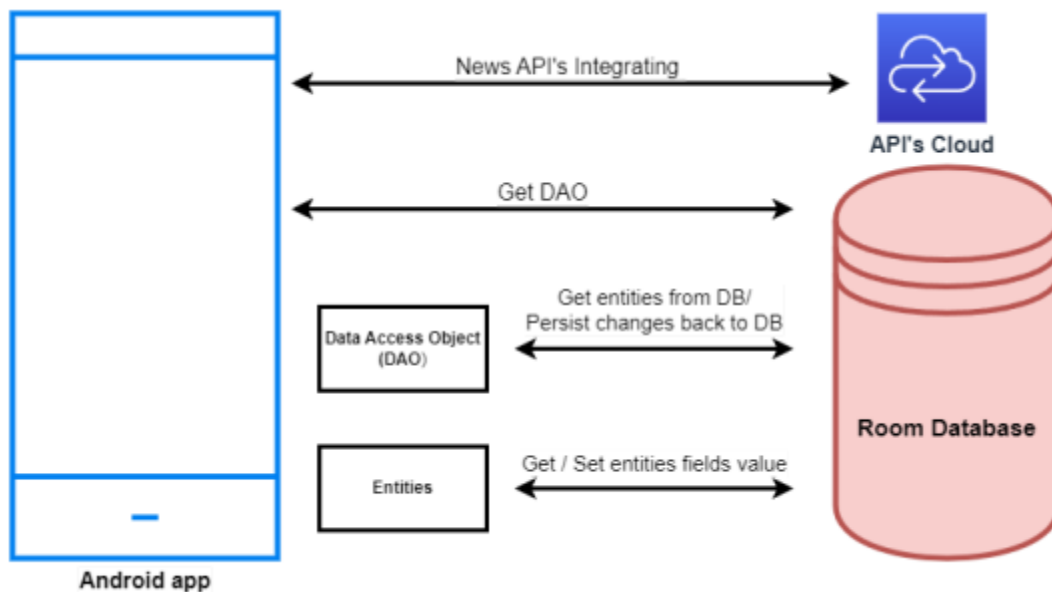
Chapter	Title	Page No
I	Introduction	03
	i)Overview	
	ii)Purpose	
II	Problem Definition & Thinking	05
	i) Empathy Map	
	ii)Ideation & Brainstorming Map	
III	Result	06
IV	Advantages & DisAdvantages	10
V	Applications	14
VI	Conclusion	19
VII	Future Scope	20
VIII	Appendix	22
	i)Source Code	

Chapter I

Introduction:

i) 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.



Project Workflow:

- Users register into the application.
- After registration , user logs into the application.
- User enters into the main page

ii) Purpose:**The use of this project:**

It helps to find the Latest top stories: **Top Headlines**. News Headlines App is the news aggregator service developed by our team, by simply sign in with your username and password. It updates the Latest News Headlines day to day

What can be achieved using this Project:

By end of this project:

1. You'll be able to work on Android studio and build an app.
2. You'll be able to integrate the database accordingly.
3. You'll be able to integrate the API's accordingly.

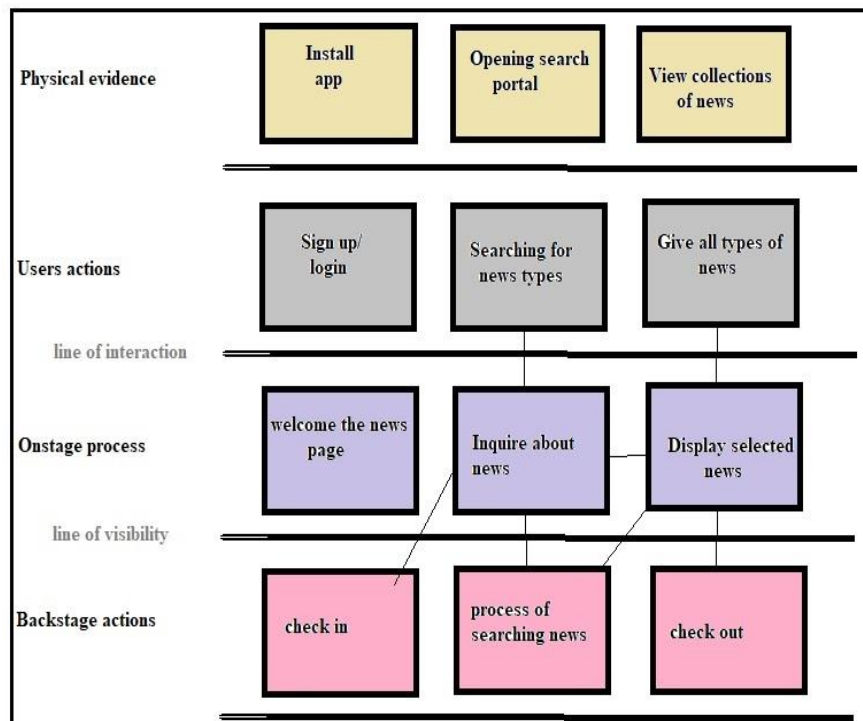
Chapter II

Problem Definition & Design Thinking:

i) Empathy Map:



ii) Ideation And Brainstorming Map:

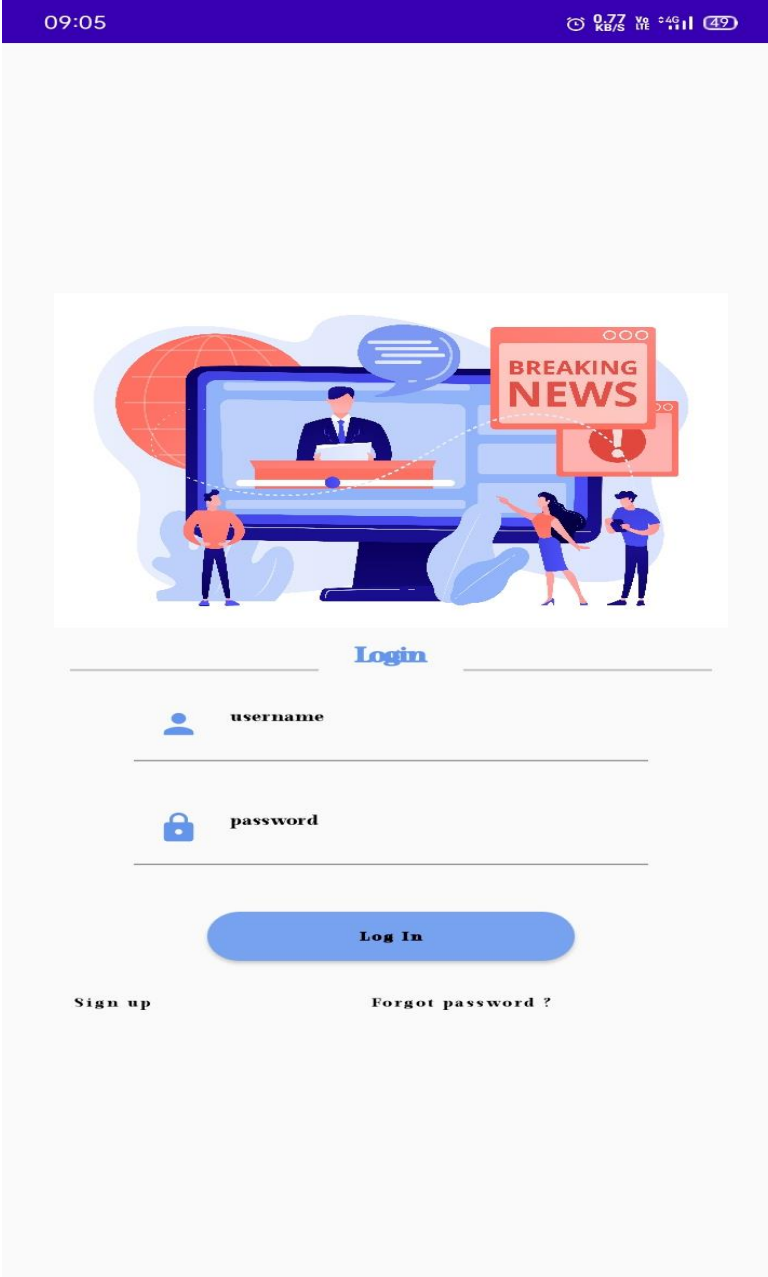


Chapter III


RESULT:

Final findings (Output):


Login Page:


A mobile application login screen with a purple header bar. The header contains the time '09:05' on the left and network status '9.77 KB/s 5G 49%' on the right. The main content area has a light gray background. At the top is a large illustration of a man in a suit sitting at a desk with a laptop, with a 'BREAKING NEWS' banner and a globe in the background. Below the illustration is the word 'Login' in blue. Underneath are two input fields: the first is labeled 'username' with a person icon, and the second is labeled 'password' with a lock icon. Below these fields is a blue 'Log In' button. At the bottom, there are two links: 'Sign up' on the left and 'Forgot password ?' on the right.

09:05 9.77 KB/s 5G 49%



Login

 username

 password

Log In

[Sign up](#) [Forgot password ?](#)

Register Page:

09:05

3.00 KB/S 5G 49

Sign Up



username



password



email


Register

Have an account? [Log in](#)


Main News Headlines Page :

09:06
0.04 KB/S
VoLTE
4G
49


Latest NEWS




JPMorgan Chase is set to report first-quarter earnings — here's what the Street expects - CNBC
The biggest U.S. bank by assets will be watched closely for clues on how the industry fared after the collapse of Silicon Valley Bank and Signature Bank.




First Mover Asia: Ether Climbs Over \$2.1K to Further Its Post Ethereum Shanghai Upgrade Surge - CoinDesk
Bitcoin has also continued its more moderately paced momentum, inching toward \$31K. ALSO: CoinDesk columnist considers Shapella's immediate and potential long term impact.



Warren Buffett's Berkshire Sells \$1.2 Billion of Yen Debt After Big Japan Bets - Yahoo Finance
(Bloomberg) — Warren Buffett's Berkshire Hathaway Inc. sold 164.4 billion yen (\$1.2 billion) of bonds on Friday, just days after the billionaire investor...

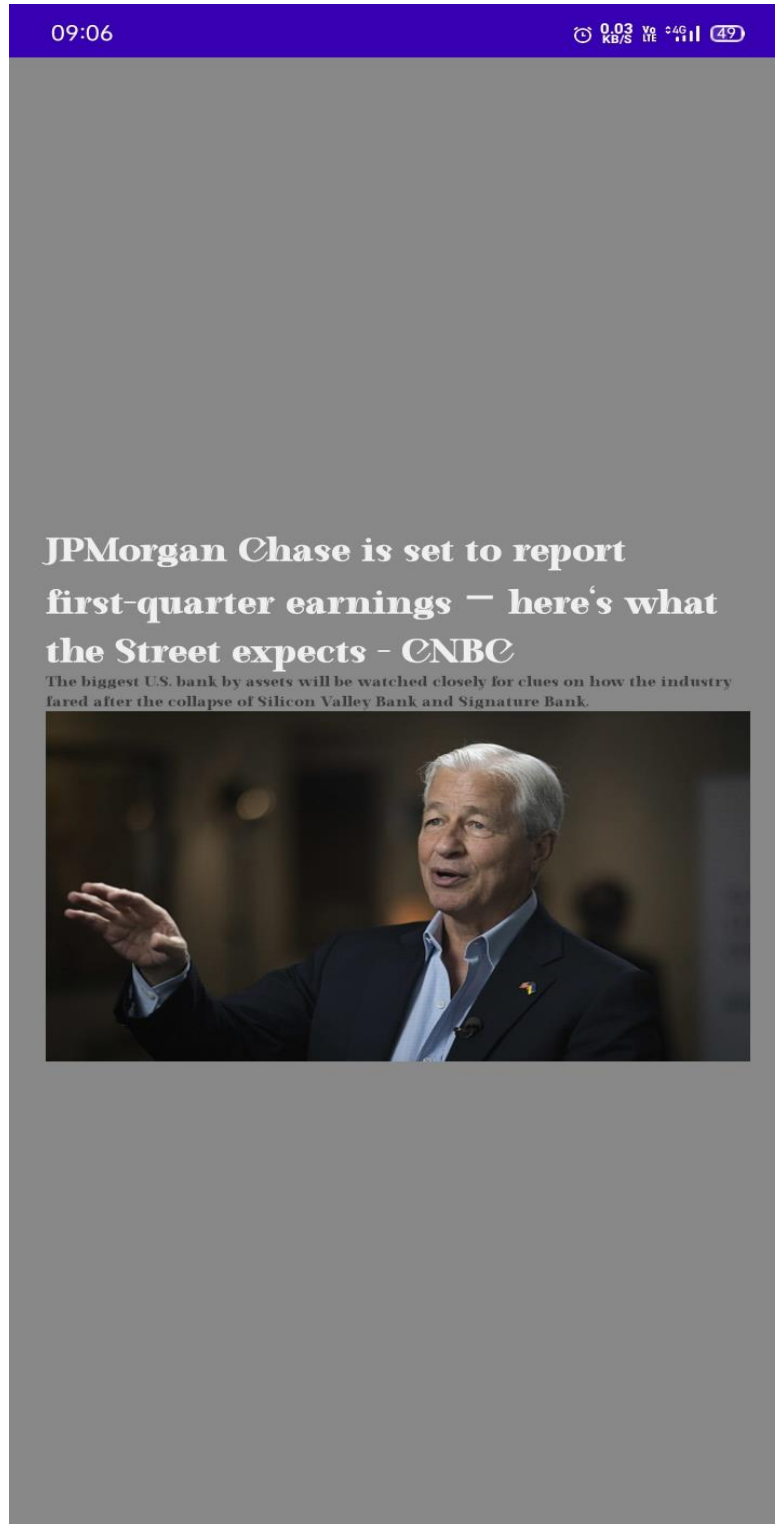


Deadly infant sleeper still being sold online even after recall - NBC News
The Consumer Product Safety Commission says a recalled infant sleeper that has been linked to the deaths of about 100 babies is still available on resale sit...



Bud Light sponsors NFL Draft as beer brand faces backlash over Dylan Mulvaney partnership - Fox News
The NFL Draft will commence in two weeks, and its main sponsor is Bud Light, which has received backlash for its partnership with trans influencer

Display News Page :



Chapter IV

ADVANTAGES & DISADVANTAGES:

Advantages:

1. Information Goes Paperless

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

2. Updates on the Go

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

3. Digital Media Keeps you Socially Active

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

4. Vast Employment Options

Digital News Media Benefits some professionals who are sound with the latest technological concepts. [Digital News Platforms](#) and forums are suitable career launch pads for such people who can join the revolutionizing world and influence their followers. Jobs like social media moderator, Social Media Manager, Social Media Strategist, and Chief Marketing Officer are some renowned profiles in the Digital Media sector.

5. Digital Journalism

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

6. Journalism Options for Freelancers

According to Upwork's "[Freelancing in America Report, 2017.](#)" online journalism jobs have boosted to 71%. Edstrom and Ladendorf also explained further that freelance journalism, worldwide, is growing as a result of shrinking newsrooms. Digital News Media benefits freelance journalists in terms of flexibility and reduced costs as they don't have to worry about the setup required for reporting. Freedom of expression is also a unique experience for freelance journalists.

7. Globalization

Thanks to the internet that enables News Media to bring coverage to the global audience. News channels are not just restricted to a particular now, as you can access content from anywhere. Digital News Media removes regional restrictions by presenting the regional events to the world expanding their viewership.

Disadvantages:

1. Fabricated Content/Hoax

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

In Brazil, 85% agree with a statement that they are worried about what is real and fake on the internet. The concern is also high in the UK (70%) and the US (67%) followed by Germany (38%) and the Netherlands (31%). Unfortunately, there are less provisions to hold the spread of fake news through social media apps like WhatsApp.

2. Negative Content

Accessing the internet is more convenient to everyone. It's easier for unscrupulous elements to spread negativity about a particular individual or group. The spread of

unfounded rumors has led to a spate of uncivilized acts in countries like India. Several countries have set up ‘tip lines’, appealing to the public to flag illegal or dangerous content.

The spread of negative or hateful news often happens via groups that are set up specifically to discuss sensitive issues.

3. Paid Subscription for Daily News

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

4. Advertising

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

5. News Overload

Apart from the leading international news channels, internet has helped many less popular news channels to evolve at a rapid rate. There is more news around the web circulating through different platforms. This is rather more confusing than to figure out what’s going on really.

Too much information partly reflects how constant news updates and different perspectives can complicate reality. A common complaint is that users are bombarded with multiple versions of the same story or the same alert. The perception of news overload is highest in the United States (40%) followed by Denmark and the Czech Republic.

Chapter V

APPLICATIONS:

1. Own Your Channels:

What's nice about building a news app is that you control the channel.

You do rely on Google and social media networks to decide when it makes the most sense to put it in front of your readers, but you've built a great and valuable asset you can count on, especially if users are registered, for the site or a newsletter.

You've probably seen the impact of Facebook's 2016 changes to the newsfeed. Social traffic is drying up. Building a business on top of social networks is dangerous territory - you have to take advantage of social traffic and encourage sharing, of course, but you know you're on rented space. And the rent is going up.

Organic traffic is down to a trickle and the networks are clearly pushing their pay to play strategies, after all, that's their business. If you publish your own mobile app, you're in full control. It's your own property, you control your listing in the app store, and you have a direct relationship with your audience.

Fostering this direct relationship is critical for audience development and ultimately revenue. It's like your website. But it comes with a subscription mechanism directly embedded in it, push notifications.

More on that later.

2. Better User Experience:

Like all of us, your readers have come to expect a native app experience from all the brands they trust to consume content from.

They want fast loading articles, offline use, no intrusive banners or popups, easy navigation designed for a thumb.

The average mobile site takes 15 seconds to load. Responsive sites are particularly terrible, as they're bloated with code trying to cater to desktop devices and mobile ones. Tracking and advertising code from the providers you use on your site are only making it worse.

Google's results are clear: 53% of visits are likely to be abandoned if pages take longer than three seconds to load, while 50% of people expect a page to load in under two seconds. AMP has made things better, but it applies to search results and Google properties, what's the experience for your most loyal readers, those that come back every day?

Progressive web apps can bridge the gap somewhat, but often don't do enough to create a truly mobile-first experience and lack push notifications and most of what makes them an "app" for iOS users, a part of your readership you can't ignore.

A native mobile app seems the only logical choice if User Experience is the main criteria in choosing how to serve your audience on their devices.

3. Higher Engagement:

There's something quite unique in how readers engage with a mobile app. [As the American Press Institute explained](#):

“Unlike the mobile website, the app is where you serve a loyal, familiar audience. They know you, and they've come here thinking, ‘Let's open up the app and see what you have for me today.’”

A mobile app presents a great opportunity to connect with loyal readers.

With a presence on the user's home screen and regular touch points with push notifications, you have the tools to grow traffic and engagement.

Mobile web users have to do a lot more work to remember that your publication exists and then get inside it. They have to load up their browser, type in your web address, and then locate the stories that interest them.

A [native application](#) holds dedicated space front and center. [Mobile websites generally tend to be much slower](#) than apps, with the average load time on the mobile web being 10.5 seconds. Mobile apps don't force users to wait, thanks to the ability to cache data locally.

4. Push notifications:

With [personalized push notifications](#) from your news mobile app, you have a more effective means to communicate with readers and get stories in front of them when it counts. According to [eMarketer research](#), only about 10% of news app users won't click on push notifications:

5. Revenue Opportunities

A mobile app gives you more opportunities to generate revenue for your publication, including:

- Displaying banner ads throughout the app - and you don't have to worry about the loss of revenue from ad blockers.
- Displaying app-only ad formats like full screen ads, offer walls, native ads that fit well within your content and promote relevant (mobile specific) offers.
- Selling in-app subscriptions to loyal readers who want to access premium content or pay to remove ads from the app and continue supporting your work.
- Selling sponsorship for the app – you can get creative here, but it won't be hard for you to show your sponsor's logo in the app's splash screen or above every article.
- You can use a paywall that offers a number of free articles and then requires users to buy a subscription or just require a subscription to all users. Purchasing access to content is so easy in a mobile app when using in-app purchase.

The Guardian released an ad-free app in 2020 that allowed subscribers (only) to scroll horizontally through paper sections. Premium tech news site The Information is launched a \$29 per year app called “Tech Top 10”. The Atlantic also launched a new app for subscribers only.

What do all these apps have in common? They all, to some degree, require readers to pay for access.

News apps are making a comeback as part of the broader industry pivot to reader revenue.

Reader revenue turns on audience loyalty, and although the apps will likely not achieve the same scale as websites – they help to identify the most loyal and valuable audience members.

Matt Skibinski, reader revenue advisor for Lenfest made the point that: “An app like this might help with conversion, but it’s really a retention play”

The habit forming qualities of apps are a key opportunity to develop the all-important engagement that can make a meaningful difference to the success of reader revenue plays.

Audiences often feel overwhelmed by a deluge of news that they can’t always trust completely – news apps might be the key to giving them a real home with a brand they trust and enjoy.

Chapter VI

CONCLUSION:

News headlines play a crucial role on social media. In a novel task to predict the social media popularity of news articles using headline-derived features, we improved significantly over several baselines. Features extracted from headline text were shown to have impact on the prediction performance when considered on their own.

This suggests that traditional editorial judgments about newsworthiness and insights from NLP research on style are applicable to predicting headline popularity on social media.

Our feature extraction methods are generic and can be repeated across different news outlets and genres. The results of the prediction model depend on the news source; further work can include performance comparison across different news outlets and online content. We are currently refining the prediction model taking into account user demographics and integrating world knowledge.

Firstly, we are considering user location (country of residence) to improve the Proximity feature. Secondly, to improve Prominence (our best-correlated feature) we are incorporating world knowledge from Wikidata to relate entity significance to the user's location.

Chapter VII

FUTURE SCOPE:

1. Online news portal design is so important as the future belongs to the online and not to the print. The explosive growth in the usage of smartphones for information and news during the past few years persistently boosted the newspaper online audience. According to surveys and studies, this trend is only going to increase in the future and there is no any sign of decrease.
2. The publishing industry is clearly at a crossroads. With the mass of free content of varying quality online, the industry is looking for new ways to cover the significant cost of high-quality journalism.
3. Trust is a key issue. Increasingly polarized politics and a high-profile backlash against ‘fake news’ make it a hard environment in which to build trust. Loyalty and direct communication with audiences will be most crucial – so we can expect a slow but steady build-up of more immersive experiences like live events, podcasts, and native apps.
4. With the younger generations turning to podcasts as a way to stay informed while on public transport, at the gym, or while relaxing in their homes; audio, in particular, looks promising.
5. The industry needs to think hard, at the same time, about how to counter increasing news avoidance in key markets – and the road ahead there is not yet clear.
6. The road ahead will be a thoroughly challenging one – with publishing one of the key industries disrupted by the digital revolution – but there is surely a pathway to

sustainable, quality, and profitable journalism within reach with so much talent working towards such a vital goal

Chapter VIII

Appendix:

i)Source Code:

Adding Required Dependencies:

```
package com.example.newsheadlines.ui.theme

import android.os.Bundle

import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview

import com.example.newsheadlines.ui.theme.ui.theme.NewsheadlinesTheme

class MainActivity : 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) {
    Greeting2(name: "Surya")
}
}
}
}
```

@Composable

```
fun Greeting2(name: String) {
    Text(text = "Hello $name!")
}
```

@Preview(showBackground = true)

@Composable

```
fun DefaultPreview2() {
    NewsheadlinesTheme {
        Greeting2(name: "Surya")
    }
}
```

Adding required dependencies:

```
plugins {
    id 'com.android.application'
```



```
id 'org.jetbrains.kotlin.android'

}

android {

    namespace 'com.example.newsheadlines'

    compileSdk 33

    defaultConfig {

        applicationId "com.example.newsheadlines"

        minSdk 24

        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:1.1.3'
androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
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'
implementation 'androidx.room:room-ktx: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")
}
```

User Data Class:

```
package com.example.newsheadlines

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

@Entity(tableName = "user_table")

data class User(

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

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

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

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

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

)
```

UserDao Interface:

```
package com.example.newsheadlines

import androidx.room.*

@Dao

interface UserDao {

    @Query("SELECT * FROM user_table WHERE email = :email")

    suspend fun getUserByEmail(email: String): User?

    @Insert(onConflict = OnConflictStrategy.REPLACE)

    suspend fun insertUser(user: User)

    @Update

    suspend fun updateUser(user: User)

    @Delete

    suspend fun deleteUser(user: User)

}
```

UserDataBase Class:

```
package com.example.newsheadlines

import android.content.Context

import androidx.room.Database

import androidx.room.Room

import androidx.room.RoomDatabase

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

abstract class UserDataBase : RoomDatabase() {

    abstract fun userDao(): UserDao

    companion object {

        @Volatile

        private var instance: UserDataBase? = null

        fun getDatabase(context: Context): UserDataBase {

            return instance ?: synchronized(this) {

                val newInstance = Room.databaseBuilder(

                    context.applicationContext,

                    UserDataBase::class.java,

                    "user_database"
```

```

        ).build()

        instance = newInstance

        newInstance

    }

}

}

}

```

UserDatabaseHelper Class:

```

package com.example.newsheadlines

import android.annotation.SuppressLint

import android.content.ContentValues

import android.content.Context

import android.database.Cursor

import android.database.sqlite.SQLiteDatabase

import android.database.sqlite.SQLiteOpenHelper

class UserDatabaseHelper(context: Context) :

    SQLiteOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION) {

```

```
companion object {  
  
    private const val DATABASE_VERSION = 1  
  
    private const val DATABASE_NAME = "UserDatabase.db"  
  
    private const val TABLE_NAME = "user_table"  
  
    private const val COLUMN_ID = "id"  
  
    private const val COLUMN_FIRST_NAME = "first_name"  
  
    private const val COLUMN_LAST_NAME = "last_name"  
  
    private const val COLUMN_EMAIL = "email"  
  
    private const val COLUMN_PASSWORD = "password"  
  
}  
  
override fun onCreate(db: SQLiteDatabase?) {  
  
    val createTable = "CREATE TABLE $TABLE_NAME (" +  
  
        "$COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +  
  
        "$COLUMN_FIRST_NAME TEXT, " +  
  
        "$COLUMN_LAST_NAME TEXT, " +  
  
        "$COLUMN_EMAIL TEXT, " +  
  
        "$COLUMN_PASSWORD TEXT" +  
  
        ")"
```



```

        db?.execSQL(createTable)

    }

    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {

        db?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")

        onCreate(db)

    }

    fun insertUser(user: User) {

        val db = writableDatabase

        val values = ContentValues()

        values.put(COLUMN_FIRST_NAME, user.firstName)

        values.put(COLUMN_LAST_NAME, user.lastName)

        values.put(COLUMN_EMAIL, user.email)

        values.put(COLUMN_PASSWORD, user.password)

        db.insert(TABLE_NAME, null, values)

        db.close()

    }

    @SuppressWarnings("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
    }
}

```

ApiService:

```

package com.example.newsheadlines

import retrofit2.Retrofit

import retrofit2.converter.gson.GsonConverterFactory

import retrofit2.http.GET

interface ApiService {

    //@GET("movielist.json")

    @GET("top-
headlines?country=us&category=business&apiKey=684cb893caf7425abeffad82ac1d0f4e")

    ///@GET("search?q=chatgpt")

    suspend fun getMovies():News

    companion object {

        var apiService: ApiService? = null

        fun getInstance() : ApiService {

```

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

Model Data Class:

```
package com.example.newsheadlines  
  
data class Model(  
  
    val name: String,  
  
    val imageUrl: String,
```

```
    val desc: String,  
  
    val category: String  
)
```

News Data Class:

```
package com.example.newsheadlines  
  
import com.example.newsheadlines.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()  
  
)
```

Source Data Class:

```
package com.example.newsheadlines

import com.google.gson.annotations.SerializedName

data class Source(

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

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

)
```

Articles Data Class:

```
package com.example.newsheadlines

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,

)
```


MainViewModel Class:

```
package com.example.newsheadlines

import android.util.Log

import androidx.compose.runtime.getValue

import androidx.compose.runtime.mutableStateOf

import androidx.compose.runtime.setValue

import androidx.lifecycle.ViewModel

import androidx.lifecycle.viewModelScope

import com.example.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()

    }

}

}

}
```

LoginActivity:

```
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.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.unit.dp

import androidx.compose.ui.unit.sp

import androidx.core.content.ContextCompat

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,
```

```
    onChange = { 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,

    onChange = { 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)

}
```

RegistrationActivity:

```
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,

    onChange = { 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)  
  
}
```

Mainpage:

```
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

import okhttp3.internal.threadName

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",

        "",

        " article"

    )

    MovieItem(context, movie = movie, 0, 0) { i ->

        Log.i("wertetest123abc", "MovieItem: "

            +i)

    }

}

fun Card(modifier: Modifier, shape: (String, Modifier) -> Unit) {

}

```

@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(selected = 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(selected=true, enabled = true, role = null

    ) {

        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) }

    )

}

}

```

DisplayNews:

```

package com.example.newsheadlines

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

                ) {

                    val desk = getIntent().getStringExtra("desk")

                    val title = getIntent().getStringExtra("title")

                    val 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) }    ) }
```