PROJECT REPORT

1. INTRODUCTION

1.1 Overview

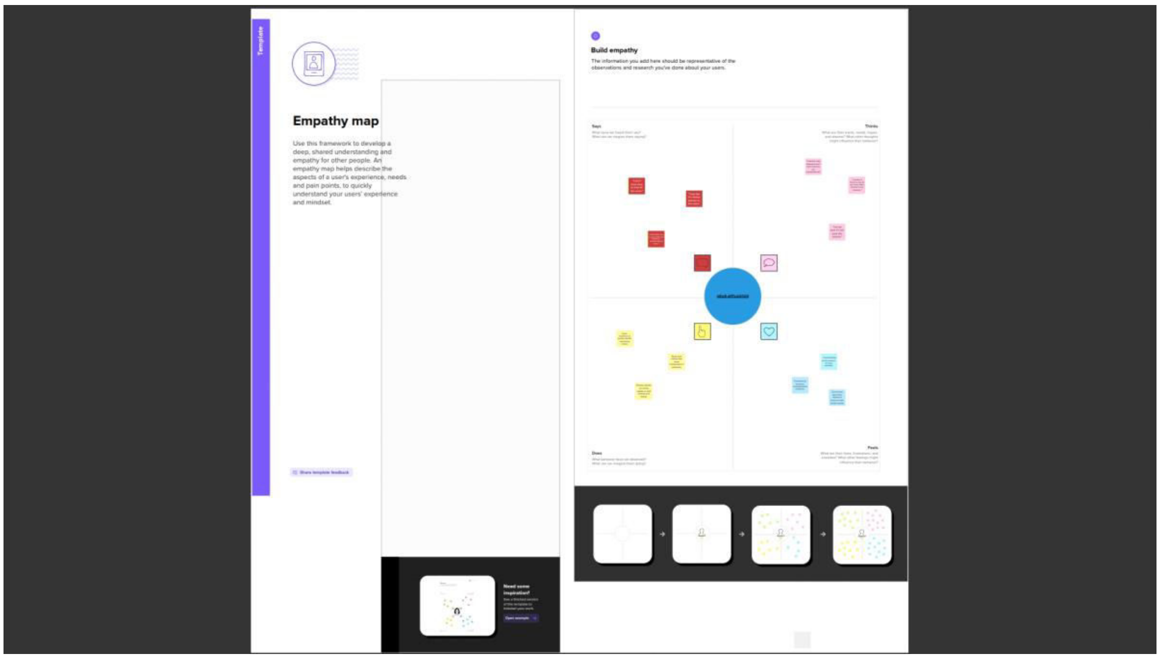
The app's main feature is displaying a list of news articles, each with a title, image, and brief description. Users can scroll through the list of articles and tap on an article to view more details. The app uses the Jetpack Compose UI toolkit to build the UI and it uses the coil library to load images.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

This project is used to display the latest headlines of the day. It provides the list of news articles. Each article has the title, the photo and a small description on the news item

2. PROBLEM DEFINITION & DESIGN THINKING

2.1 Empathy Map



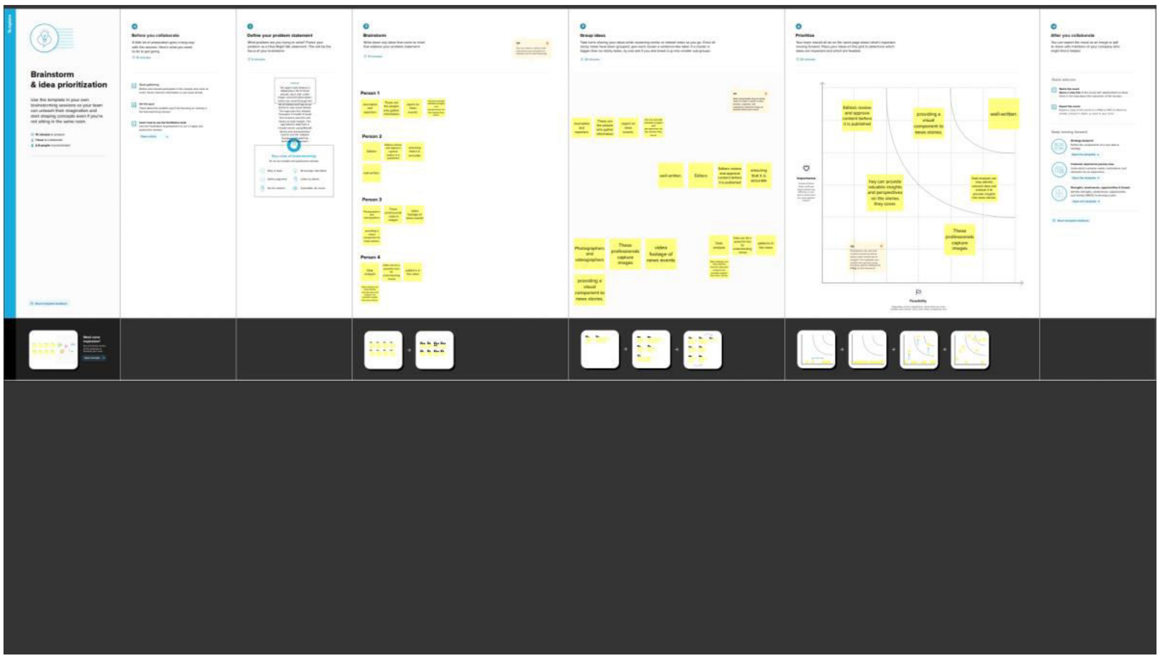
Empathy map

Use this framework to develop a deep, shared understanding and empathy for other people. An empathy map helps describe the aspects of a user's experience, needs and pain points, to quickly understand your users' experience and mindset.

Build empathy

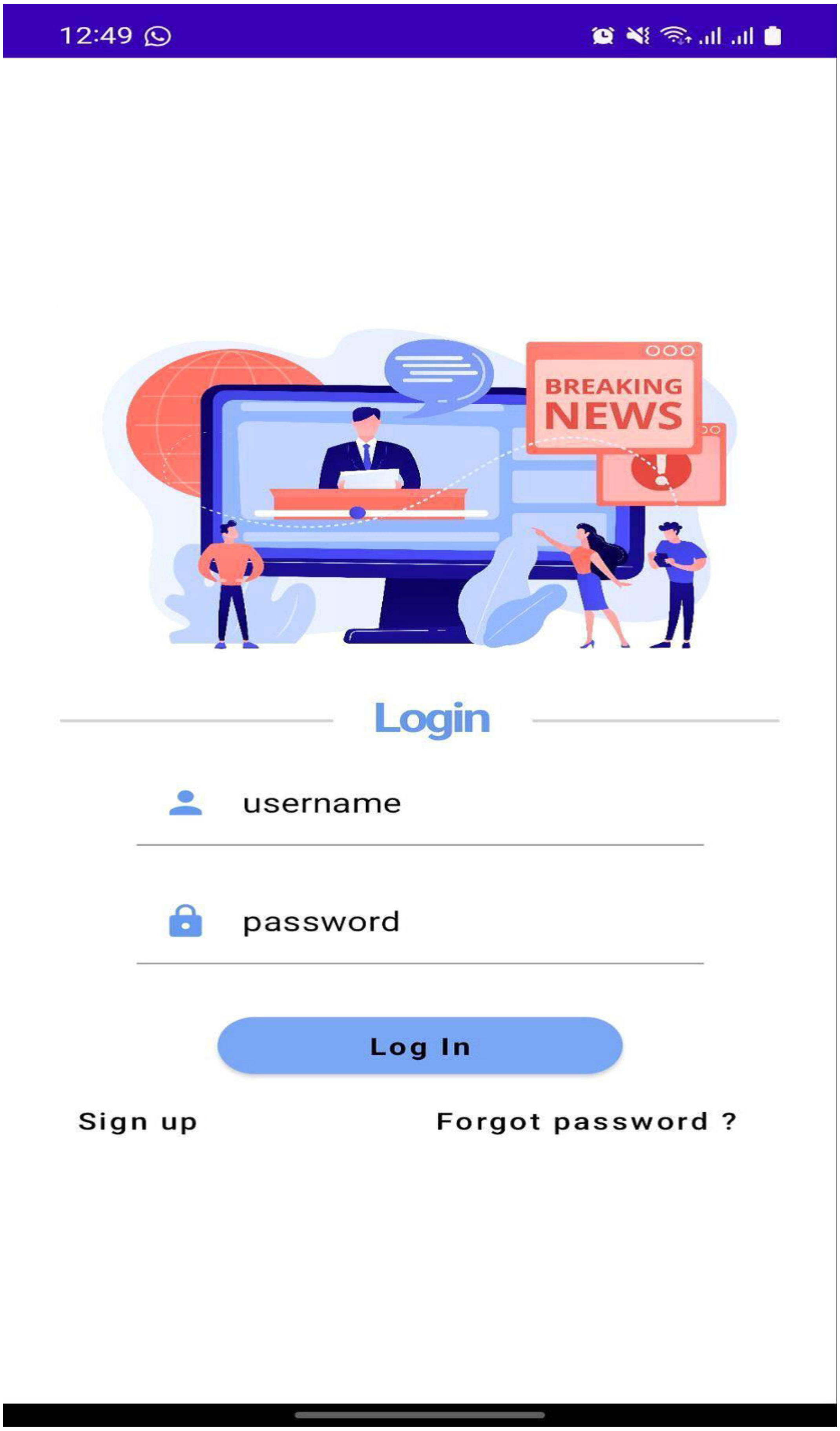
The information you add here should be representative of the observations and research you've done about your users.

2.2 Ideation & Brainstorming Map

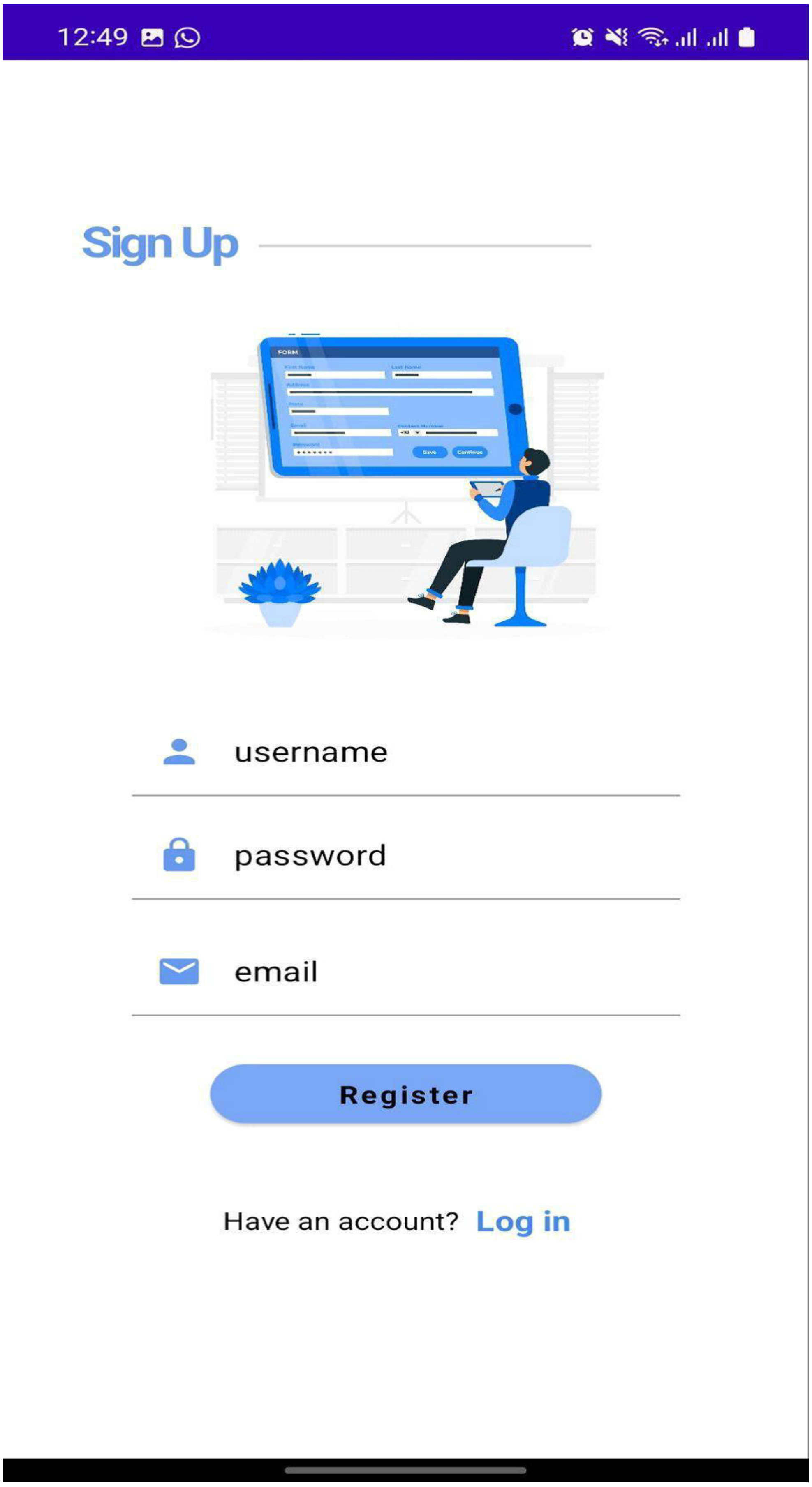

Brainstorm & idea prioritization
Before you collaborate
Define your problem statement
Brainstorm
Person 1
Person 2
Person 3
Person 4
Group ideas
Prioritize
After you collaborate
Editors
well-written
Editors review and approve content before it is published
ensuring that it is accurate
Photographers and videographers
These professionals capture images
video footage of news events
providing a visual component to news stories.
hey can provide valuable insights and perspectives on the stories they cover.
Importance
Feasibility

3. RESULT

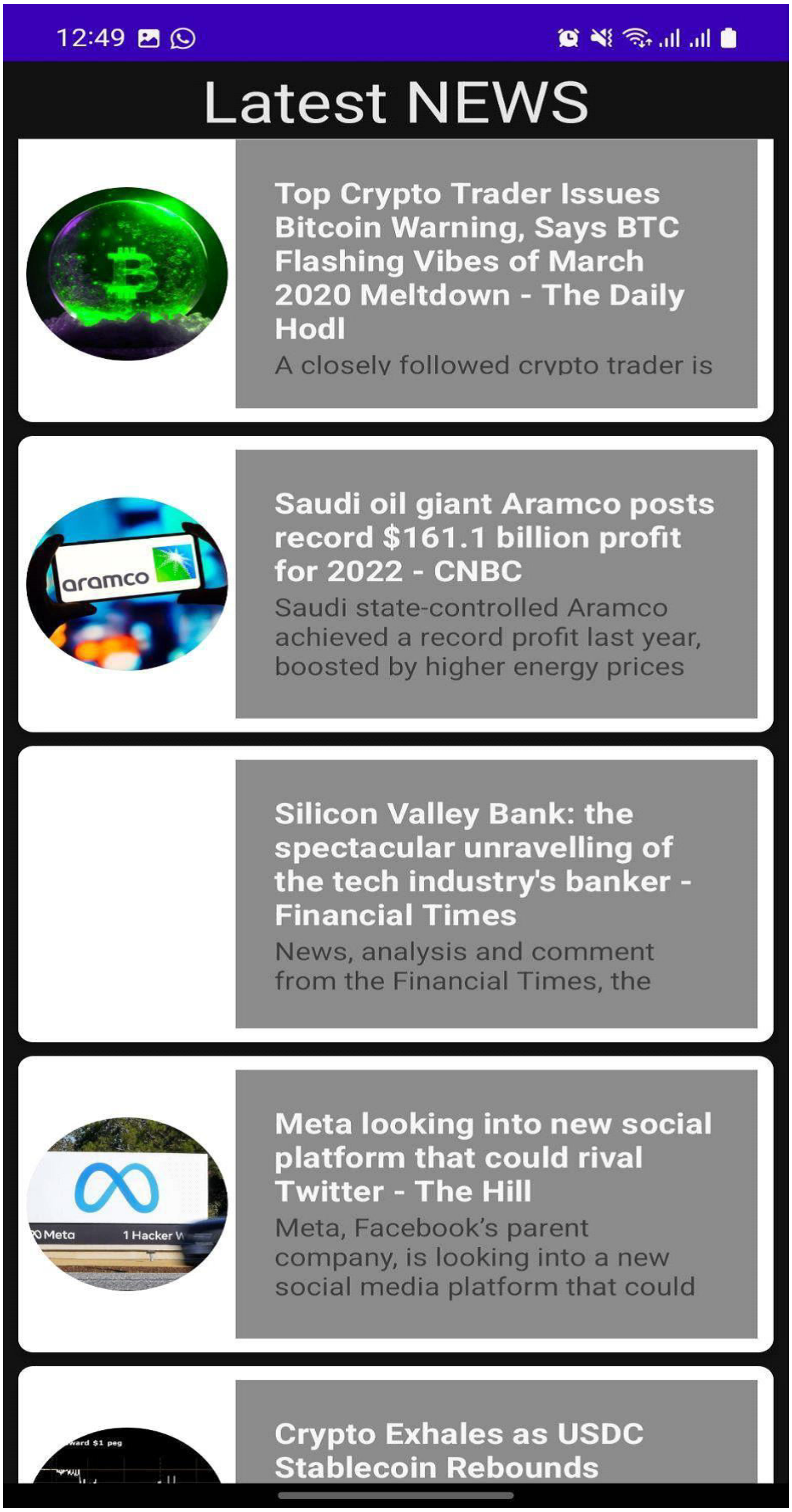
Login Page


12:49
BREAKING
NEWS
Login
username
password
Log In
Sign up
Forgot password ?

Register Page



Main News Headlines Page


12:49

Latest NEWS

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

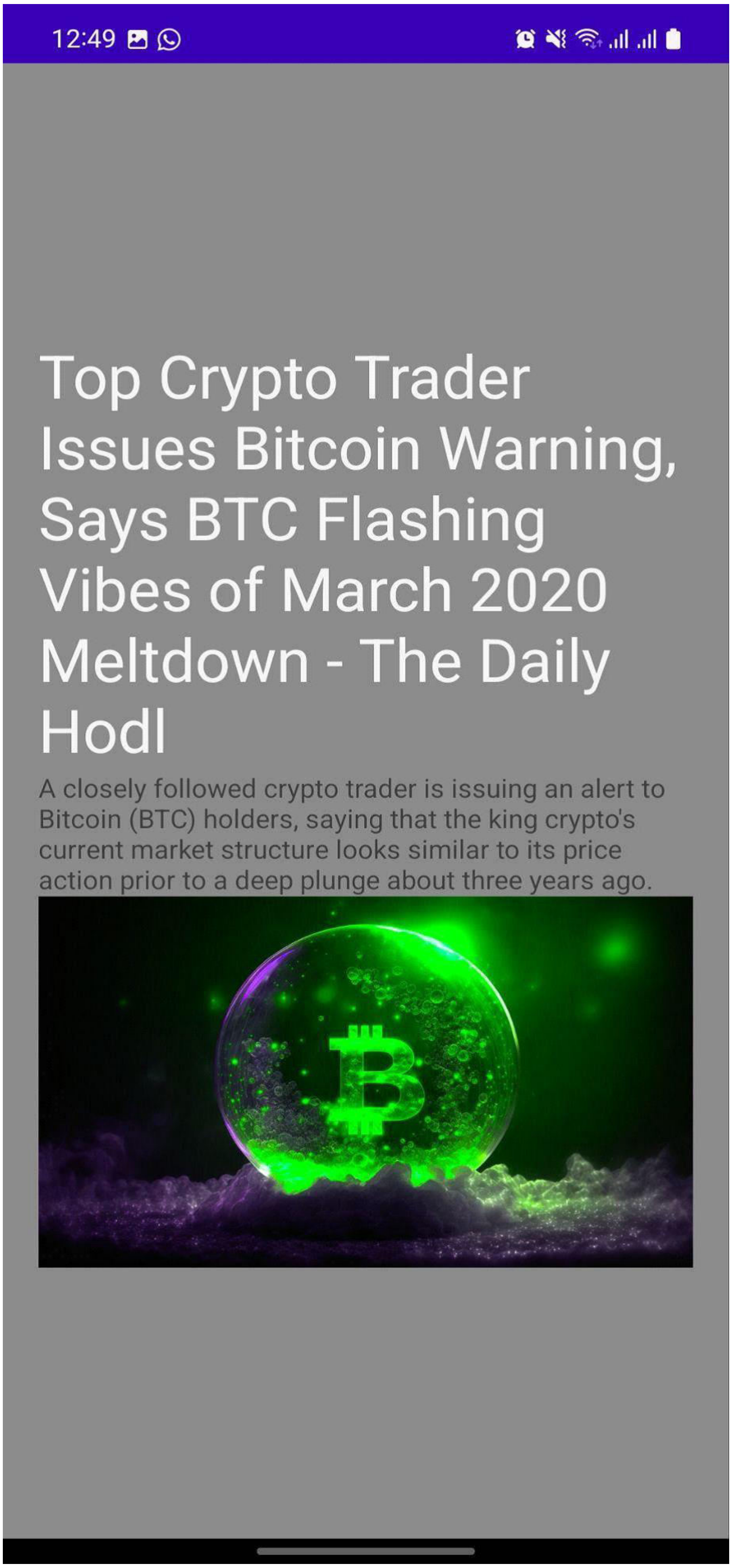
Saudi oil giant Aramco posts record $161.1 billion profit for 2022 - CNBC
Saudi state-controlled Aramco achieved a record profit last year, boosted by higher energy prices

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

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

Crypto Exhales as USDC Stablecoin Rebounds

Display News Page



12:49

Top Crypto Trader Issues Bitcoin Warning, Says BTC Flashing Vibes of March 2020 Meltdown - The Daily Hodl

A closely followed crypto trader is issuing an alert to Bitcoin (BTC) holders, saying that the king crypto's current market structure looks similar to its price action prior to a deep plunge about three years ago.

4. ADVANTAGES & DISADVANTAGES

- **Advantages**
    - Easy to use
    - Secured
    - Latest news headlines
- **Disadvantages**
    - Depending on the External website
    - Only one News API

5. APPLICATIONS

The Project provides only the latest news headlines. The App can be used as personal app for individuals. The App will increase the General knowledge of the users and keep them updated at the time. The project will be useful to all the sections of the society.

6. CONCLUSION

The feature of this app to provides a list of news articles. Each news item contains the tile, image, and explanation. The user is allowed to scroll through the list of articles and tap on an article to view more details. Here, the news items is gathered from an external API.

7. FUTURE SCOPE

The present App can be enhanced by adding more features. User may be allowed to customize the app based on their requirements. More categories can be included the future edition of the App.

8. APPENDIX

// User.kt

```
package com.example.newsheadlines

import androidx.room.ColumnInfo

import androidx.room.Entity

import androidx.room.PrimaryKey


@Entity(tableName = "user_table")

data class User(

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

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

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

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

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


    )
```

// UserDao.kt

```
package com.example.newsheadlines

import androidx.room.*

@Dao

interface UserDao {

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

    suspend fun getUserByEmail(email: String): User?
    @Insert(onConflict = OnConflictStrategy.REPLACE)

    suspend fun insertUser(user: User)

    @Update

    suspend fun updateUser(user: User)

    @Delete

    suspend fun deleteUser(user: User)

}
```

// UserDatabase.kt

```
package com.example.newsheadlines

import android.content.Context

import androidx.room.Database

import androidx.room.Room

import androidx.room.RoomDatabase

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

abstract class UserDatabase : RoomDatabase() {

    abstract fun userDao(): UserDao

    companion object {

        @Volatile

        private var instance: UserDatabase? = null

        fun getDatabase(context: Context): UserDatabase {

            return instance ?: synchronized(this) {
                val newInstance = Room.databaseBuilder(
                    context.applicationContext,
                    UserDatabase::class.java,
                    "user_database"
                ).build()
                instance = newInstance
                newInstance
            }
        }
    }
}
```

// UserDatabaseHelper.kt

```
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

    }

}
```

// 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=684cb893caf7425ab
effad82ac1d0f4e")

    ///@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.kt

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

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


// News.kt
package com.example.newsheadlines
import com.example.example.Articles
import com.google.gson.annotations.SerializedName
data class News (
  @SerializedName("status") var status:String?= null,
  @SerializedName("totalResults") var totalResults : Int?
= null,
  @SerializedName("articles") var articles     : ArrayList<Articles>
= arrayListOf()
)


// Source.kt
package com.example.example
import com.google.gson.annotations.SerializedName
data class Source (
  @SerializedName("id"   ) var id   : String? = null,
  @SerializedName("name" ) var name : String? = null
      )
```

// 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,

  )
```

// MainViewModel.kt

```
package com.example.newsheadlines

import android.util.Log

import androidx.compose.runtime.getValue

import androidx.compose.runtime.mutableStateOf

import androidx.compose.runtime.setValue

import androidx.lifecycle.ViewModel

import androidx.lifecycle.viewModelScope

import com.example.example.Articles

import kotlinx.coroutines.launch

class MainViewModel : ViewModel() {

    var movieListResponse:List<Articles> by mutableStateOf(listOf())

    var errorMessage: String by mutableStateOf("")

    fun getMovieList() {

        viewModelScope.launch {

            val apiService = ApiService.getInstance()

            try {
                val movieList = apiService.getMovies()

                movieListResponse = movieList.articles

            }

            catch (e: Exception) {

                errorMessage = e.message.toString()

            }

        }

    }

}
```

// 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)
}
```

// RegisterActivity.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)
}
// 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
```

```kotlin
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) }

        )

    }

}
```

// 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) }
    )
}
```

- **INTRODUCTION**