1INTRODUCTION

1.1 OVERVIEW

Latest news headlines app were the most familiarly used app by the massive amount of people to know about the facts & incidents throughout the world Andriod app plays a vital in this current technology world.

The main goal of this app is to be feed the latest news through their mobile itself.It is simple to access the app which is user friendly.

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. It Contains following pages such as

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

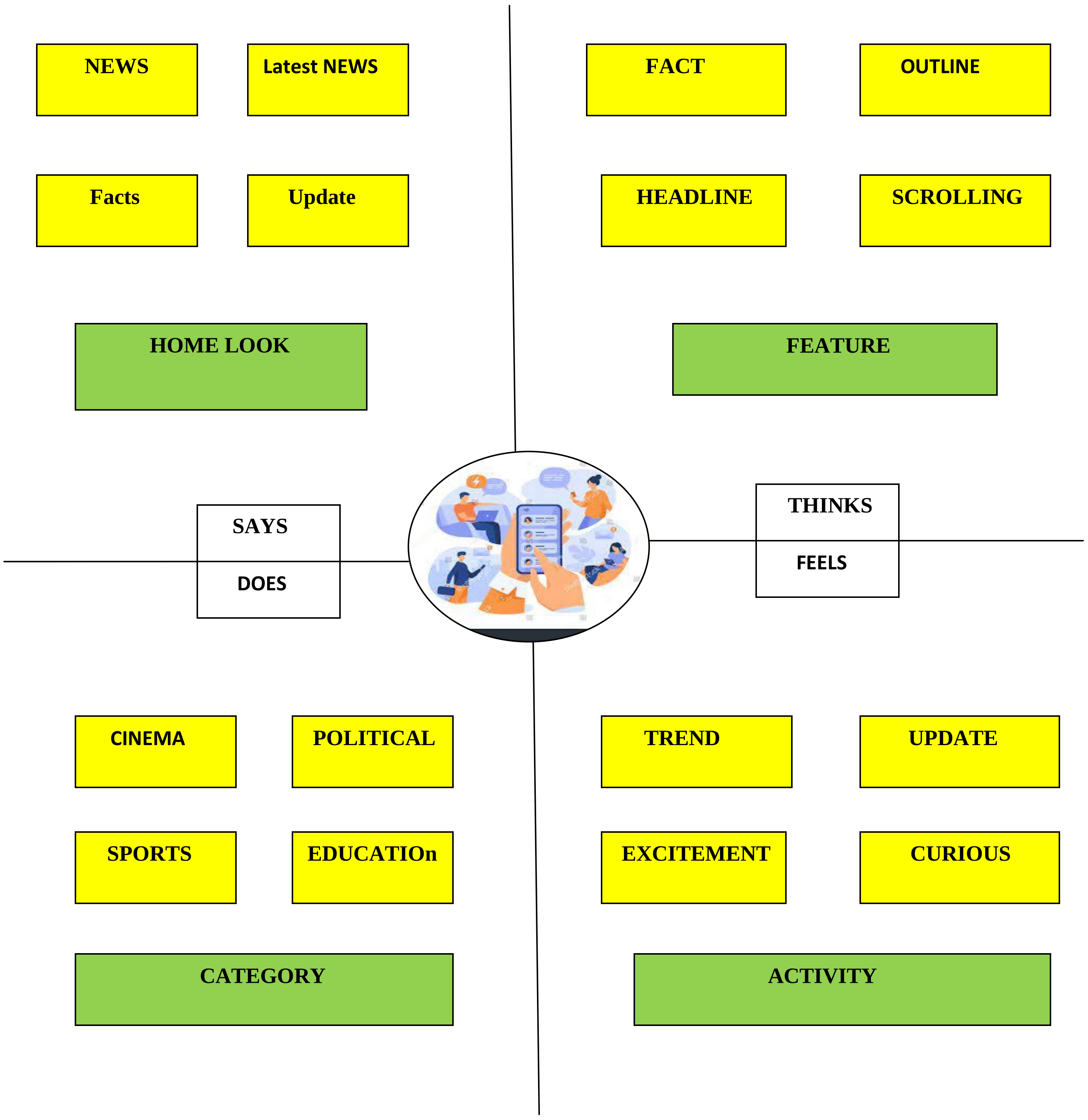
1.2 PURPOSE

This project is used to feed about the news instantly through a mobile app itself. It made the users to know about the brief news into simple headlines through the mobile app itself. It can be achieved through learning the process over this project were,

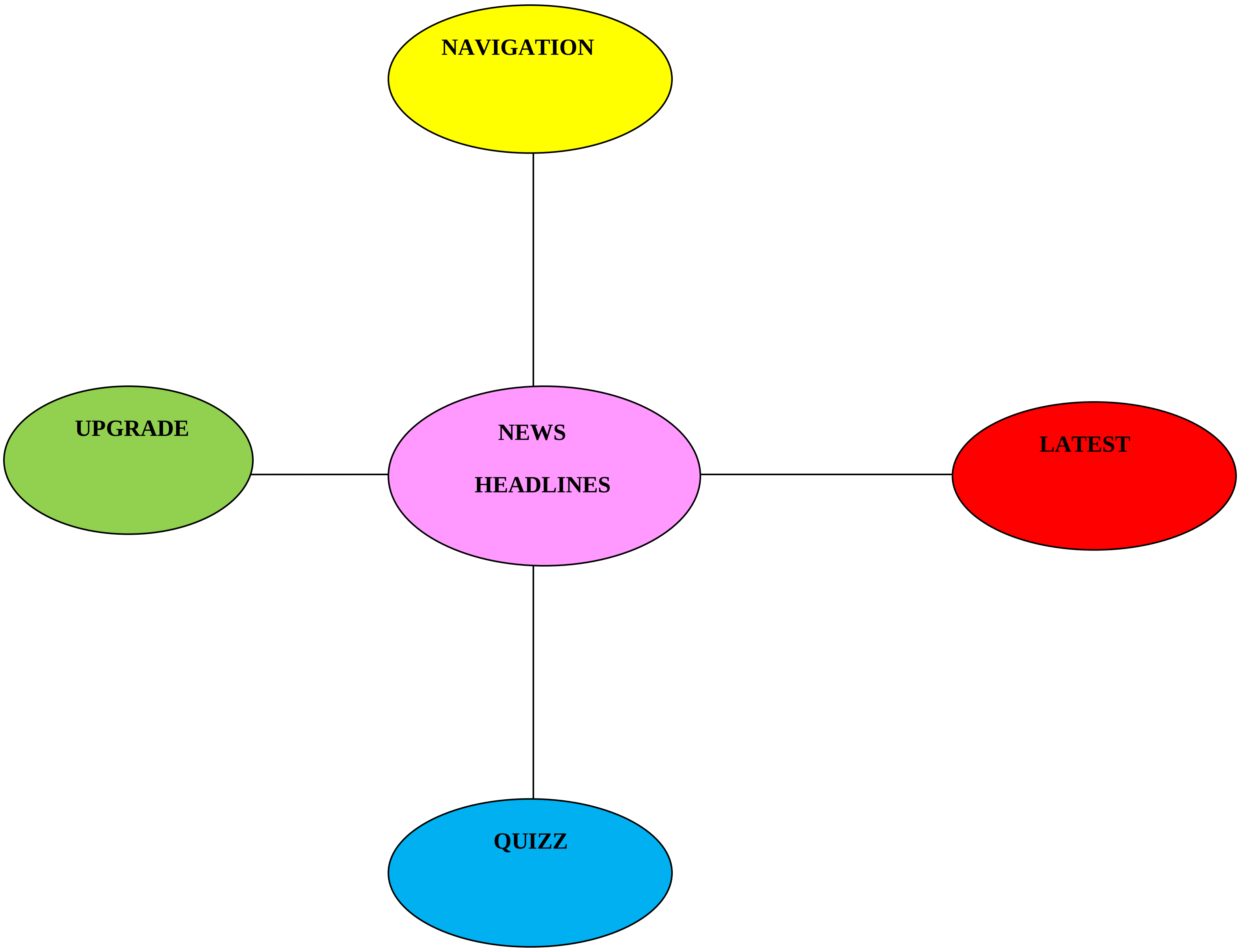
- It will be able to work on Android studio and build an app.
- It will be able to integrate the database accordingly.
- It will be able to integrate the API’s accordingly.

2.PROBLEM DEFINITION AND DESIGN THINKING

2.1 Empathy Map



2.2 IDEATION AND BRAINSTROMING MAP



3.RESULT

Login page

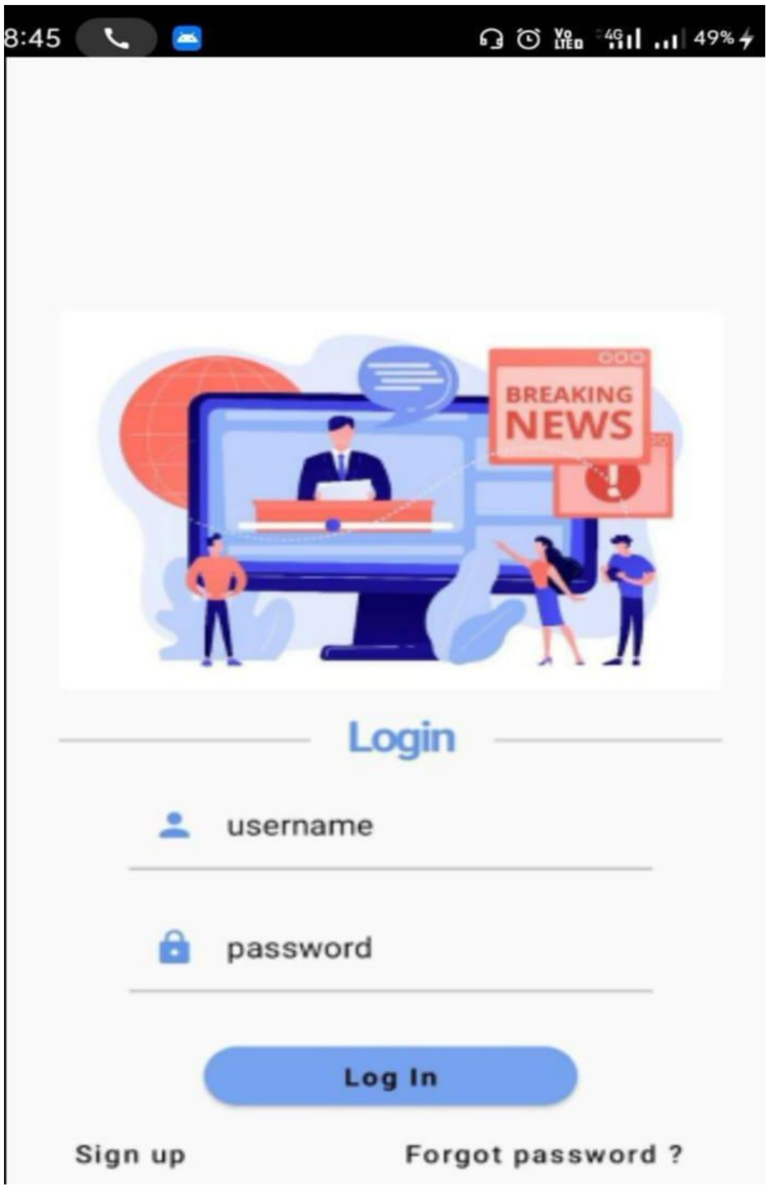


Figure 3.1

Register Page

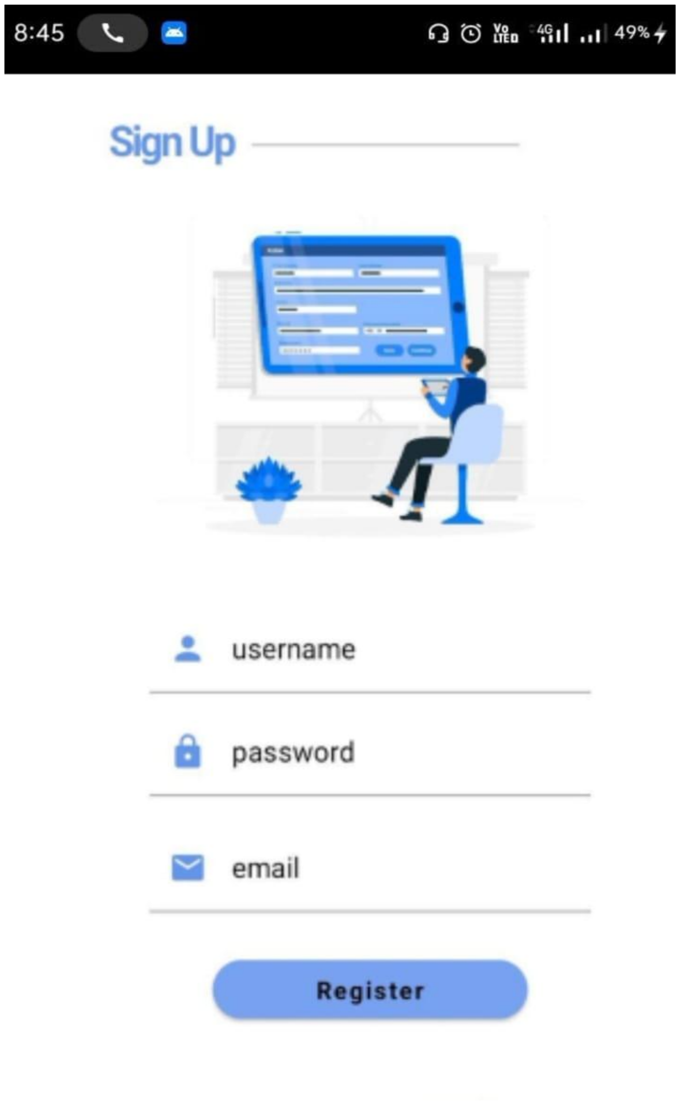
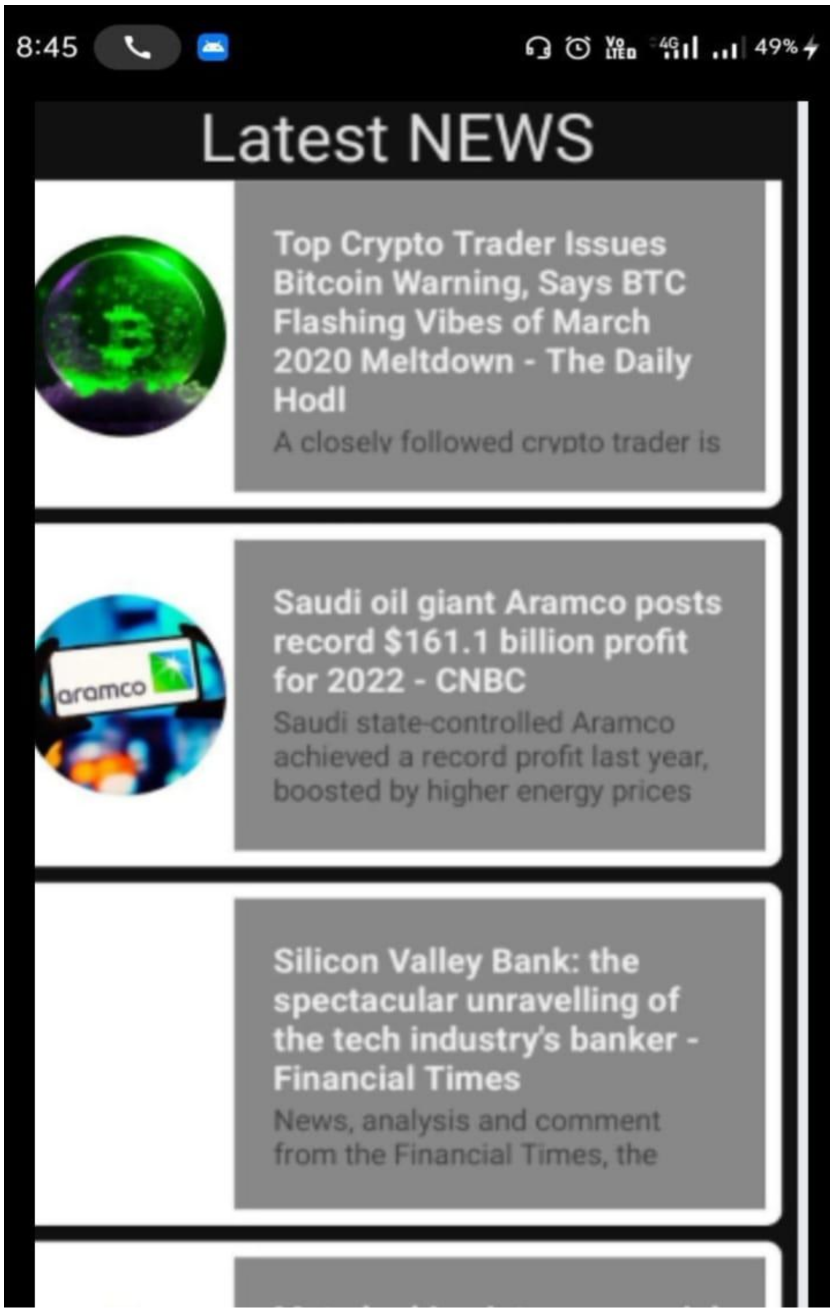


Figure 3.2

Main News HeadLines page



8:45 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

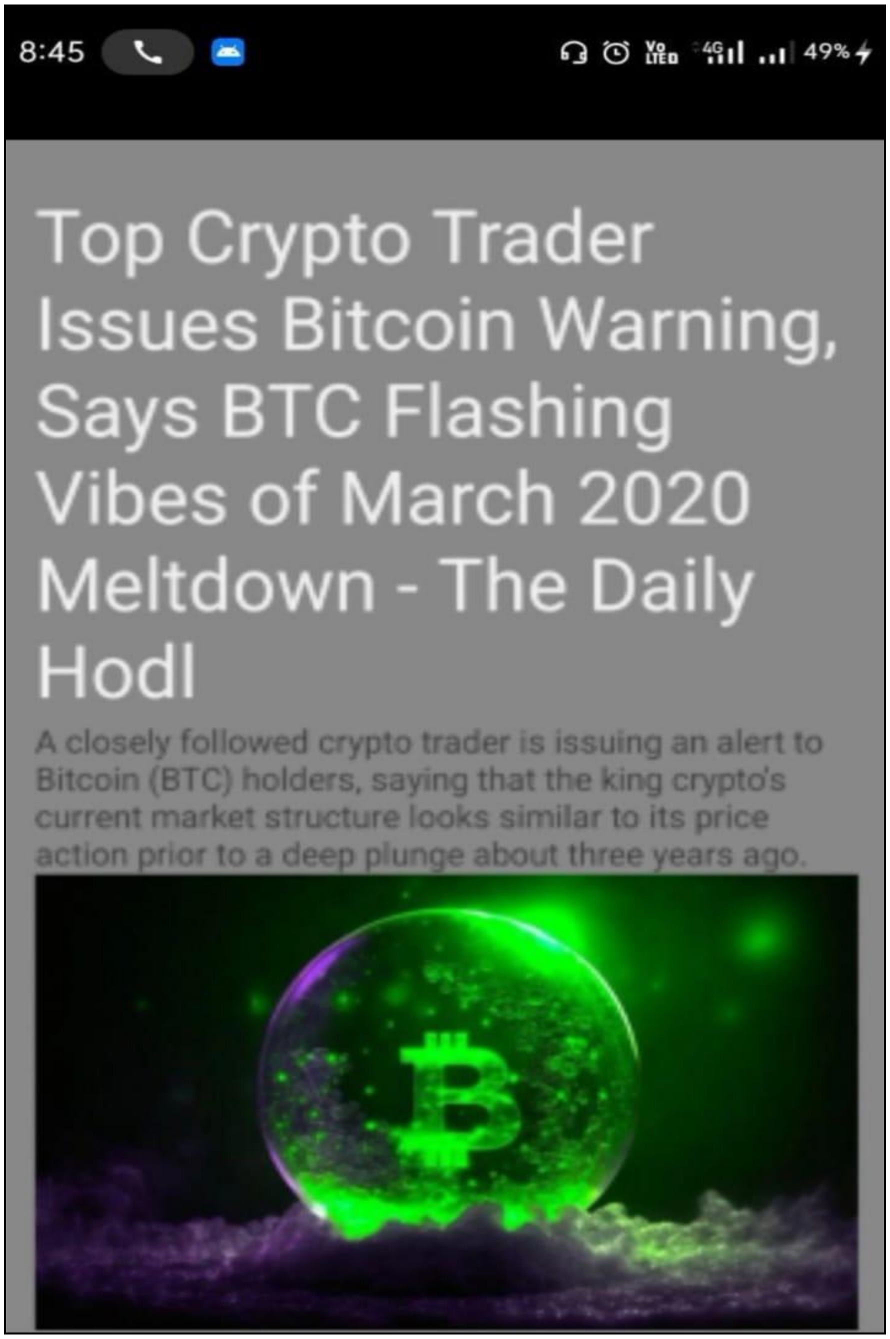
Saudi oil giant Aramco posts record $161.1 billion profit for 2022 - CNBC

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

Display news page



8:45 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. ADVANTAGE AND DISADVANTAGES

ADVANTAGES

Automatic updates: The user gets automatic updates of the latest news on the app and so does not have to refresh the app window again and again. The user can get regular news updates related to business, sports, technology, and others on their Smartphone with simple to use navigation and in the preferred language as well. So if you want cricket daily news in Hindi, then you can choose the default app language toEnglish and read in the language you are most comfortable with.

Video coverage: The mobile news apps provide the user with the latest news videos and coverage's on the mobile phone. Just by going to the mobile news app, the user can access the latest news videos in a single tap.

Home screen widgets: The user also gets the option to set home screen widgets for such mobile apps which in turn helps in quick access to the app and news stories. With just a single click, the user can easily read the news.

Have a wide range: Such mobile apps provide the latest and reliable news update from the world. The user gets access to information in every field such as politics, sports and many. The app covers all the top news stories, headlines, topics, events, videos, etc. from across the world.

Bookmarks: The users get the option to save their favorite news topics for later view by bookmarking it in the app.

Push notifications: The mobile news apps provide the users with push notifications as well. So when the user is opening any other window on the mobile phone or when he/she is offline, then push notifications display in the notification panel of the phone. This enables the user to access the news stories from the notification panel itself.

DISADVANTAGES

- The most crucial disadvantage of mobile apps is that they are constantly connected to the internet. This means that they can be used to collect personal data, track user movements, and spy on them. This is a huge issue, as it can hurt both users' privacy and the economy.

- The use of mobile apps is also impacting society in other ways. For example, it is causing a decline in the usage of traditional newspapers and magazines. People are instead turning to mobile apps to read news, which is hurting conventional news publishers' revenue.

5 APPLICATION

- 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.

- The main focus of this application is to connect news articles from all around the world and deliver it to user as fast as possible in best visualize way

- This will help the users to share news on various platforms such as Twitter and Facebook. This will not only give an amazing user experience and also will also increase the views.

- When a user is not online due to some reason he/she should have to access to the internet. Whenever the user is online the news content is downloaded in the cache memory of the app, this is how a user can access to the content offline

- It is an old saying “first impression is the last impression”. So the developer should make sure that the application should leave a mark on the users. This is where you need to focus on bringing interactive, visual and architectural designs as well. It means that the content should be distributed in the app such that the screen do not appear crowded with the content.

6 CONCLUSION

The most interesting and useful app that was a trend setting over the internet and also used by a various people over the world .The news headlines app provides the huge factor to know about the facts from the place where they itself.

It provide a great platform in the pandemic situations to know more about the disaster over the surrounding and buy the support of this app we can secure ourselves.

In Covid pandemic period the news app plays major role to discover the new cases, and delivery the facts to the people over internet itself which is very useful to the people who are Isolated can also knows the information over them and be safety from spread of the viruses

And also it is helpful to social medias in delivering news via internet itself .By this useful technology there is no means of spreading virus that the people can know the news buy the app not by the new paper.

It completely provide the huge amount of knowledge and information throughout the world itself.

7. FEATURES AND SCOPE

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.

Must Have :

Users register into the application.

After registration , user logins into the application.

User enters into the main page.

SCOPE

The scope of this educational project content is the better and best in future days to develop this android application which is very useful in knowing the information through the news itself in this busy world**.**

8.APPENDIX

Create 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?,


            )
```

Create user dao 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)
}
```

Create a user an user database 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()

    }

  @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
}
```

create an user database helper 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()
    }

    @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 service and required classes for Integrating API

Database for news integration into project

```
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!!

        }

    }}
```

Create model data class

```
package com.example.newsheadlines

data class Movie(val name: String,

                 val imageUrl: String,

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

Create News Data class

```
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()

)
```

Create source data class

```
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

)
```

Create Article Data Class

```
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,
 )
```

Create MainView Model 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.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()
            }
        }
    }
}
```

Creating Login Activity.Kt with Database

```
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)
}
```

Creating Register Activity.Kt with Database

```
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 Kt

```
package com.example.newsheadlines


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

class MainPage : ComponentActivity() {
    val mainViewModel by viewModels<MainViewModel>()
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            NewsHeadlinesTheme {
                // A surface container using the 'background' color from the theme
                Surface(color = MaterialTheme.colors.background) {
                    Column() {
                        Text(text = "Latest NEWS", fontSize = 32.sp, modifier = Modifier.fillMaxWidth(),
textAlign = TextAlign.Center)


                        MovieList(applicationContext, movieList = mainViewModel.movieListResponse)

                        mainViewModel.getMovieList()

                    }

                }

            }

        }

    }

}


@Composable

fun MovieList(context: Context, movieList: List<Articles>) {

    var selectedIndex by remember { mutableStateOf(-1) }

    LazyColumn {


        itemsIndexed(items = movieList) {

                index, item ->

            MovieItem(context,movie = item, index, selectedIndex) { i ->

                selectedIndex = i

            }

        }

    }


}
```

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



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

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

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

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

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

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



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

```
                    Text(
                        text = movie.title.toString(),
                        style = MaterialTheme.typography.subtitle1,
                        fontWeight = FontWeight.Bold
                    )
HtmlText(html = movie.description.toString())
                }
            }
        }
    }
    @Composable
    fun HtmlText(html: String, modifier: Modifier = Modifier) {
        AndroidView(
            modifier = modifier
                .fillMaxSize()
                .size(33.dp),
            factory = { context -> TextView(context) },
            update = { it.text = HtmlCompat.fromHtml(html,
HtmlCompat.FROM_HTML_MODE_COMPACT) }
        )
    }
}
```

Creating News Display Kt

```
package com.example.newsheadlines


import android.content.Context
import android.content.Intent

import android.content.Intent.FLAG_ACTIVITY_NEW_TASK

import android.os.Bundle

import android.util.Log

import android.widget.TextView

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.activity.viewModels

import androidx.compose.foundation.Image

import androidx.compose.foundation.background

import androidx.compose.foundation.clickable

import androidx.compose.foundation.layout.*

import androidx.compose.foundation.lazy.LazyColumn

import androidx.compose.foundation.lazy.itemsIndexed

import androidx.compose.foundation.selection.selectable

import androidx.compose.foundation.shape.RoundedCornerShape

import androidx.compose.material.Card

import androidx.compose.material.MaterialTheme

import androidx.compose.material.Surface

import androidx.compose.material.Text

import androidx.compose.runtime.*

import androidx.compose.ui.Modifier

import androidx.compose.ui.graphics.Color

import androidx.compose.ui.text.font.FontWeight

import androidx.compose.ui.text.style.TextAlign

import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp

import androidx.compose.ui.viewinterop.AndroidView

import androidx.core.text.HtmlCompat

import coil.compose.rememberImagePainter

import coil.size.Scale

import coil.transform.CircleCropTransformation

import com.example.example.Articles

import com.example.newsheadlines.ui.theme.NewsHeadlinesTheme


class MainPage : ComponentActivity() {

    val mainViewModel by viewModels<MainViewModel>()

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        setContent {

            NewsHeadlinesTheme {

                // A surface container using the 'background' color from the theme

                Surface(color = MaterialTheme.colors.background) {

                    Column() {



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


                        MovieList(applicationContext, movieList = mainViewModel.movieListResponse)

                        mainViewModel.getMovieList()

                    }

                }
            }
        }
    }
}

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

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

}

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



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

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

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

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

```
) {
    Surface(color = Color.White) {


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


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

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

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

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

Modifying Android Manifest.XML

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <application
        android:allowBackup="true"
```

```
android:dataExtractionRules="@xml/data_extraction_rules"
android:fullBackupContent="@xml/backup_rules"
android:icon="@drawable/news_app_icon"
android:label="@string/app_name"
android:supportsRtl="true"
android:theme="@style/Theme.NewsHeadlines"
tools:targetApi="31">
<activity
    android:name=".DisplayNews"
    android:exported="false"
    android:label="@string/title_activity_display_news"
    android:theme="@style/Theme.NewsHeadlines" />
<activity
    android:name=".RegistrationActivity"
    android:exported="false"
    android:label="@string/title_activity_registration"
    android:theme="@style/Theme.NewsHeadlines" />
<activity
    android:name=".MainPage"
    android:exported="false"
    android:label="@string/title_activity_main_page"
    android:theme="@style/Theme.NewsHeadlines" />
<activity
    android:name=".LoginActivity"
    android:exported="true"
    android:label="@string/app_name"
```

```
            android:theme="@style/Theme.NewsHeadlines">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```