THE LATEST HEADLINES

1. INTDRODUCTION

1.1Overview

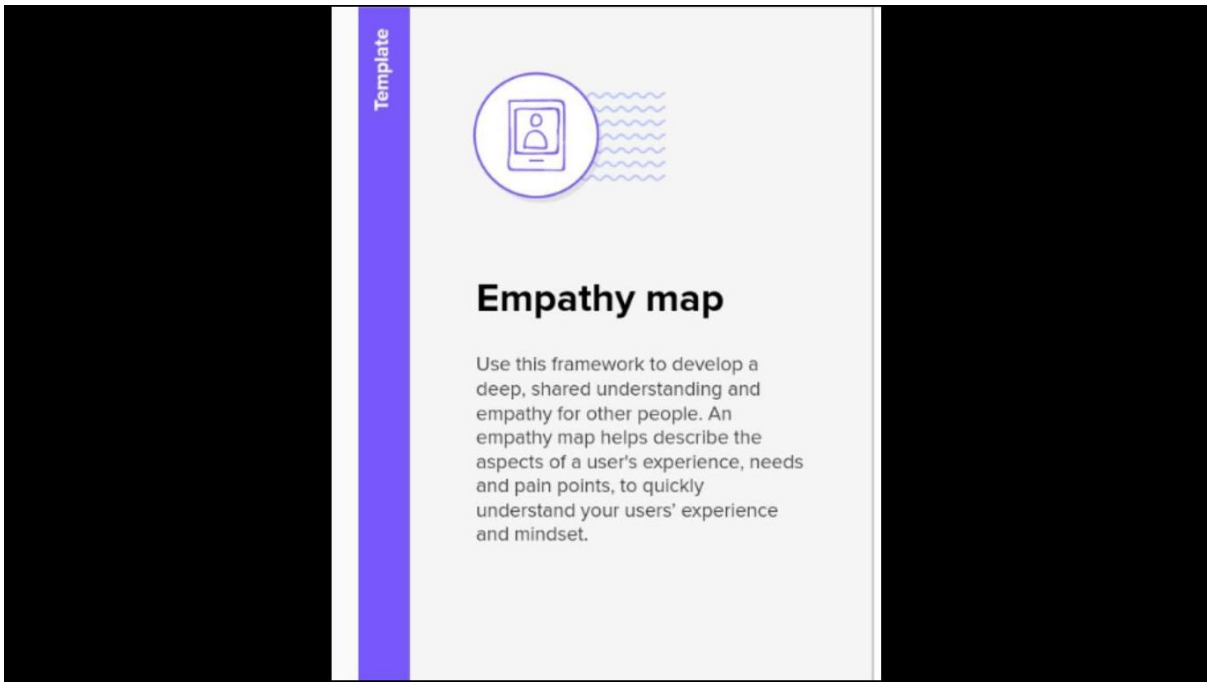
The app’s main feature is displaying a list of news articles, each with a title, image, and brief description. Users can scroll through the list of articles and tap on an article to view more details. The app uses the Jetpack Compose UI toolkit to build the UI and it users the coil library to load images. The app fetches data from a remote server using Retrofit library and demonstrates how to use the Jetpack Compose UI toolkit for Android development.

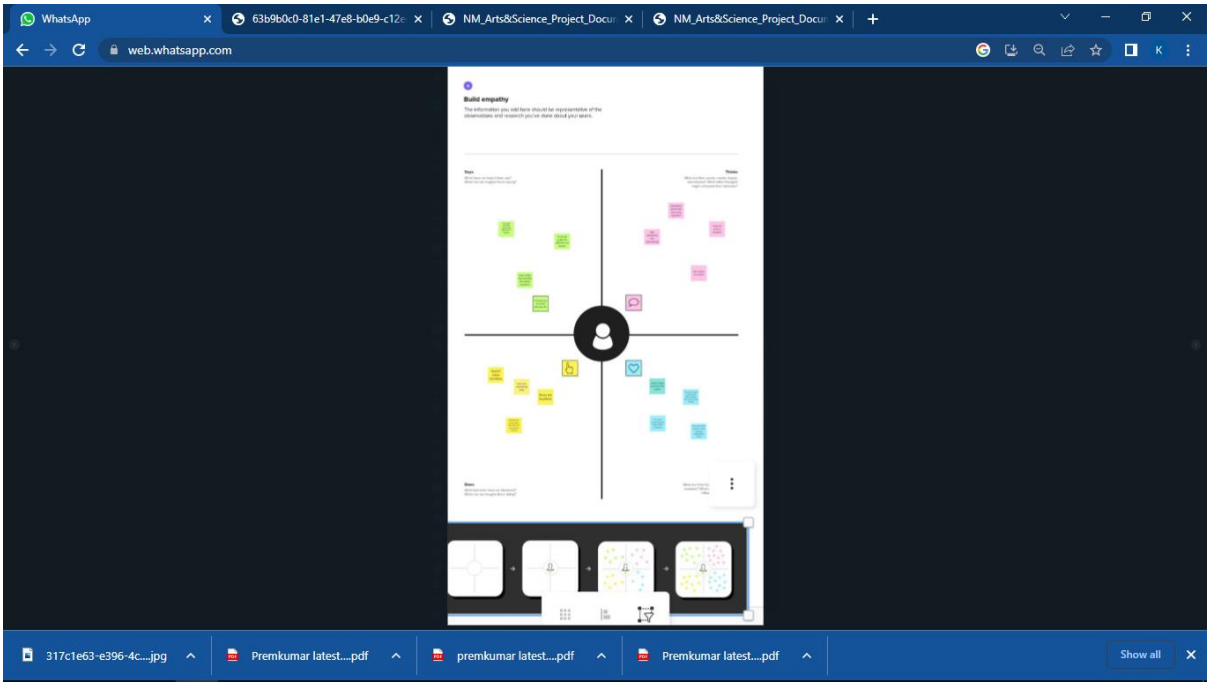
1.2Purpose

A headline’s purpose is to quickly and briefly draw attention to the story. It generally written by a copy editor, but may also be written by the writer, the page layout designer, or other editors.

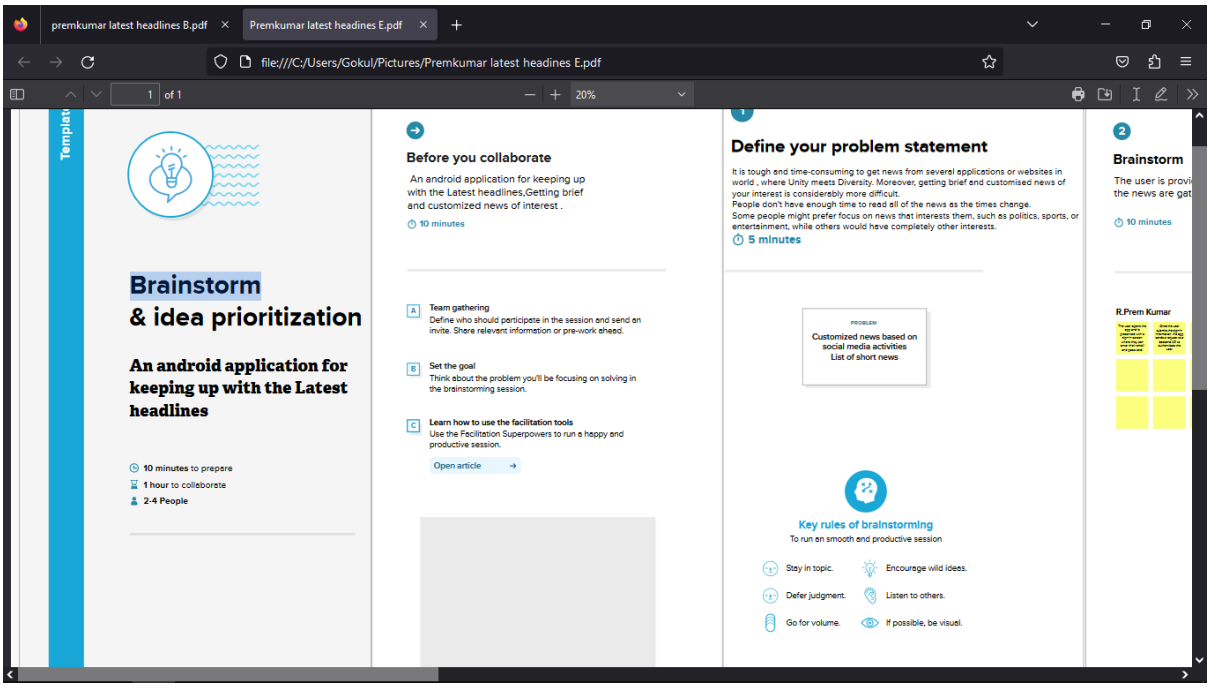
2. PROBLEM DEFINITION AND DESIGN THINKING

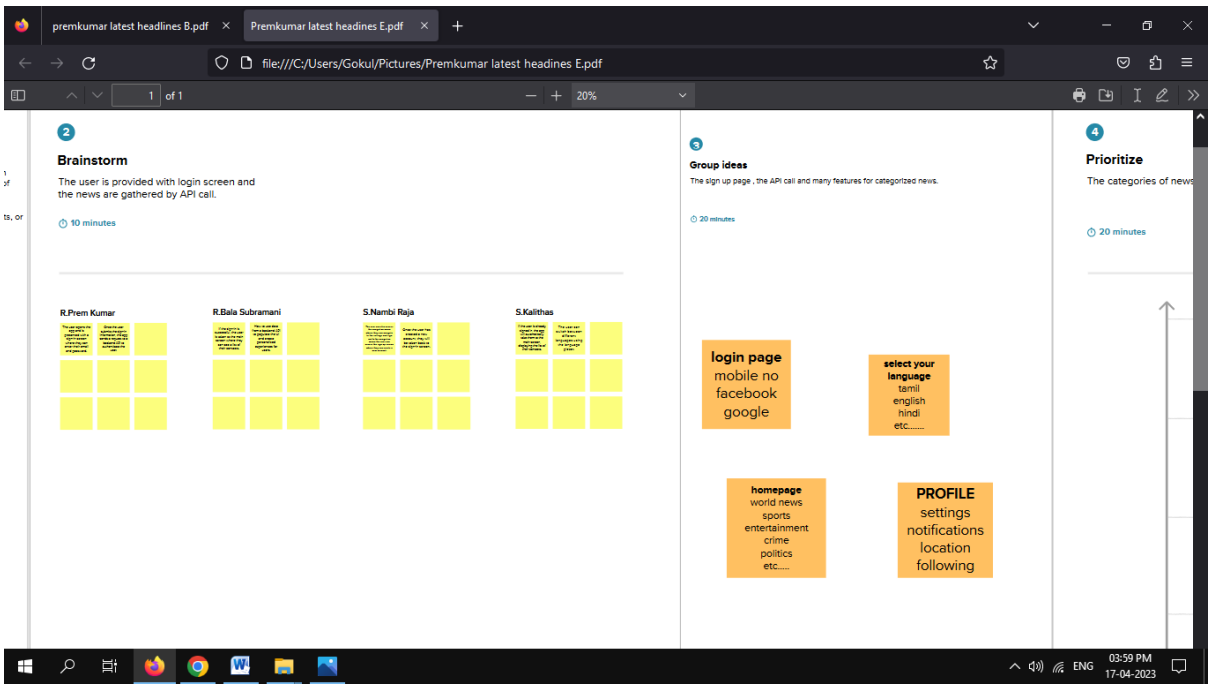
2.1 Empathy Map





Ideation& Brainstorming Map





premkumar latest headlines B.pdf

file:///C:/Users/Gokul/Pictures/premkumar latest headlines B.pdf

1 of 1 20%

3

Group ideas

The sign up page , the API call and many features for categorized news.

20 minutes

login page
mobile no
facebook
google

select your language
tamil
english
hindi
etc.....

homepage
world news
sports
entertainment
crime
politics
etc.....

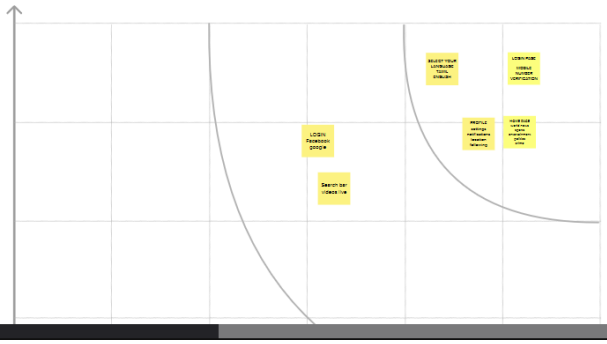
PROFILE
settings
notifications
location
following

4

Prioritize

The categories of news based on the people interest.

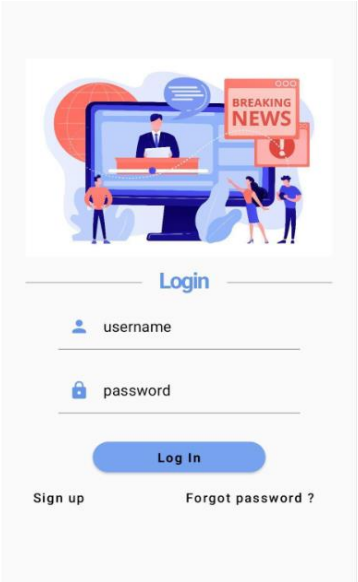
20 minutes



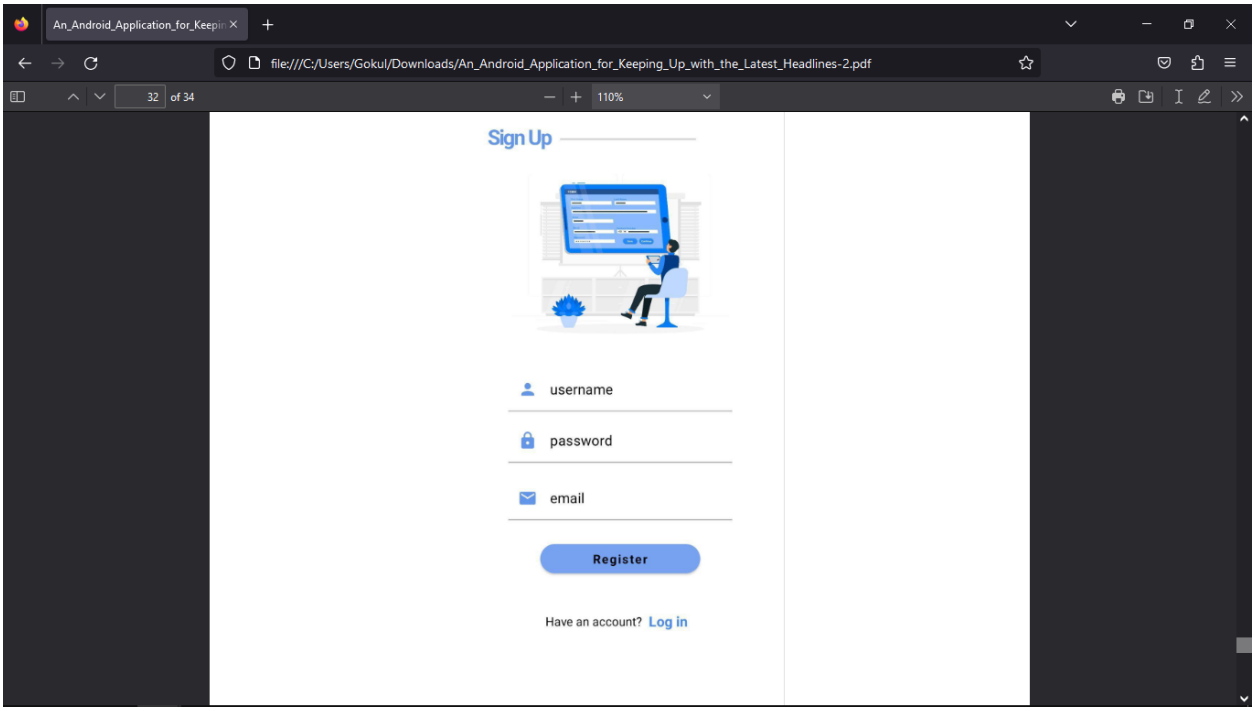
RESULT

Find Output of the Application:

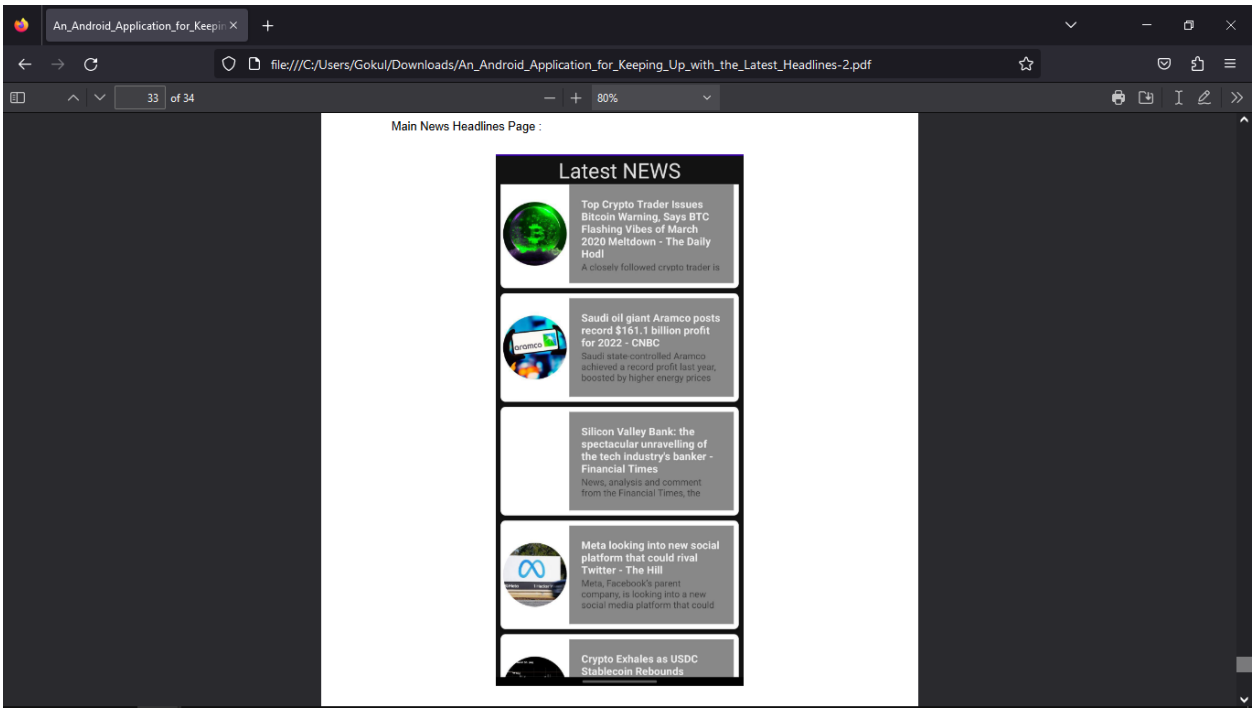
Login Page:



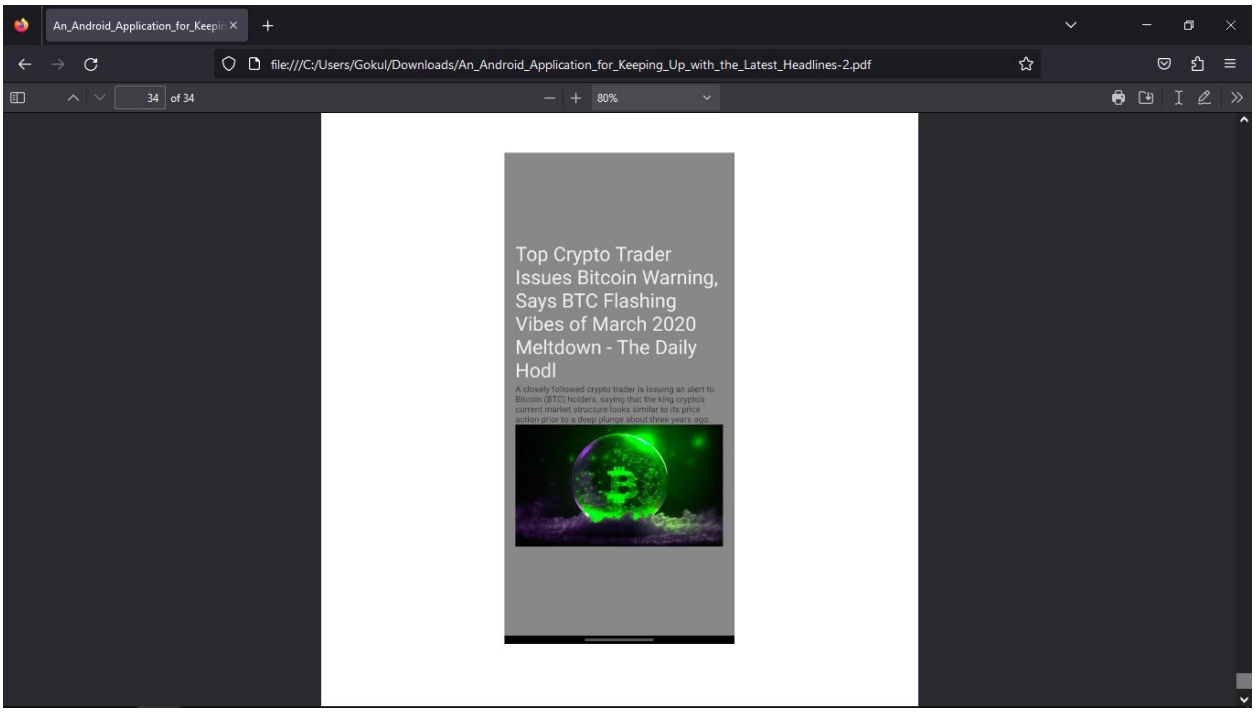
Register Page:



Main page:


An_Android_Application_for_Keepin
file:///C:/Users/Gokul/Downloads/An_Android_Application_for_Keeping_Up_with_the_Latest_Headlines-2.pdf
33 of 34
80%
Main News Headlines Page :
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

Location Page:



3. ADVANTAGES & DISADVANTAGES.

ADVANTAGES:

- Summarize
- Generate interest
- Satisfy immediacy needs
- Direct attention.

DISADVANTAGES:

- It CAN BE limited by time.
- It may rely too heavily on personalities, emotions, opinion .not facts.
- It can short change complex stories or avoid them altogether
- Lack of prestige.

5. APPLICATIONS

- Google News
- Smart News
- News Break
- Good News
- Ground News
- Global News
- AP News
- The Time of India
- News Headlines and weather live
- News Point
- World News Live
- Headlines

6. CONCLUSION

- We conclude news headlines as
- A short news broadcast briefly outlining the main news stories of the day
- Banner
- Summarize news stories, direct readers' attention to certain facts over others, and help news users decide
  On which stories to click

7. FUTURE SCOPE

In a democracy, the role of media has prime responsibility to educate, inform, guide and make society aware of the issues. The data has been troubling for a long time, and gets more convincing each day.

8. APPENDIX

User Class Code

```
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?,

)
```

User Dao

```
package com.example.newsheadlines


import androidx.room.*


@Dao

interface UserDao {


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

    suspend fun getUserByEmail(email: String): User?
```

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

    @Update
    suspend fun updateUser(user: User)

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

Database Class Code

```
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
            }
        }
    }
}
```

Database Helper Class Code

```
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
  }

}
```

API Service Interface

```
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 Class Code

```
package com.example.newsheadlines



data class Movie(val name: String,
                 val imageUrl: String,
                 val desc: String,
```

```
        val category: String)
```

News Class Code

```
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()
)
```

Model Class Code

```
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 Class Code

```
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,

  )
```

Main View Model Class Code

```
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()
         }
      }
   }
}
```

Complete Class Code

```
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)

}
```