**An Android Application for Keeping Up with the**

**Latest Headlines**

**1.INTRODUCTION**

**Overview**

In this project, we aim to build an application that will allow users to get the latest headlines from various news sources. The application will provide a user-friendly interface that can be accessed through a web browser or a mobile device.

The application will be designed to aggregate news from various sources such as major news outlets, local news sources, and independent journalists. The latest headlines will be presented in a concise and easy-to-read format, allowing users to quickly browse through the news and read the articles that interest them.

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.

To build the application, we will use a combination of programming languages and technologies, including HTML, CSS, JavaScript, and APIs. We will also use machine learning algorithms to analyze user preferences and provide personalized recommendations.

**Purpose**

The purpose of this application is to provide users with a centralized platform to access the latest news and stay informed about current events. In today's fast-paced world, staying up-to-date with news can be challenging, as there are so many different sources and platforms to follow.

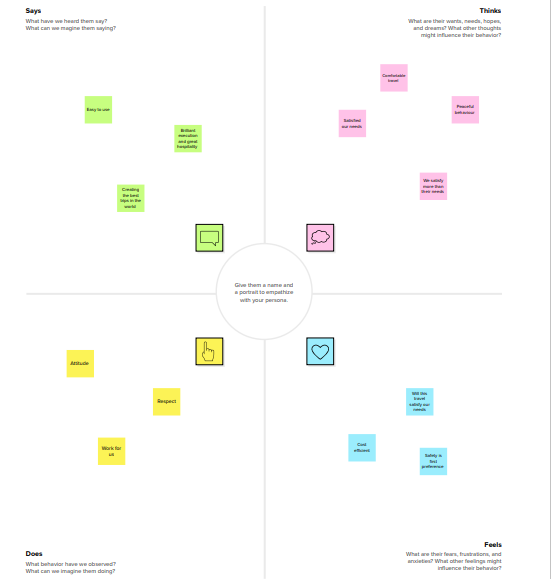
This application aims to solve this problem by aggregating news from various sources and presenting it in a user-friendly format. Users can quickly browse through the latest headlines, read articles that interest them, and share news with their friends and family.

In addition to providing quick access to news, the application also aims to personalize the news experience for users. By analyzing user preferences and behavior, the application can provide personalized recommendations and help users discover new sources of information.

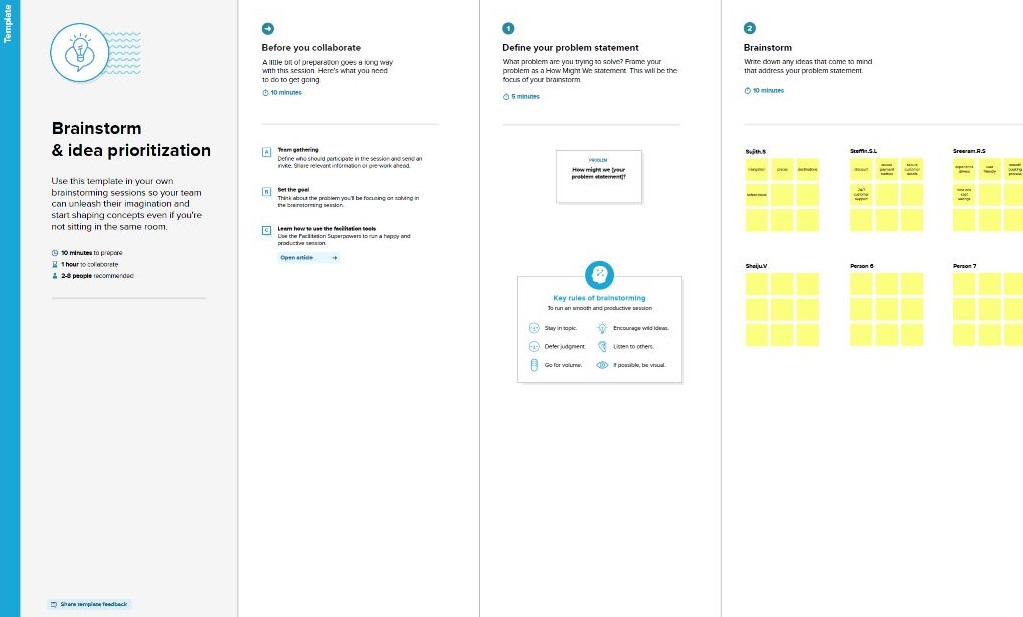
Overall, the purpose of this application is to make it easier for people to stay informed and engaged with the world around them.

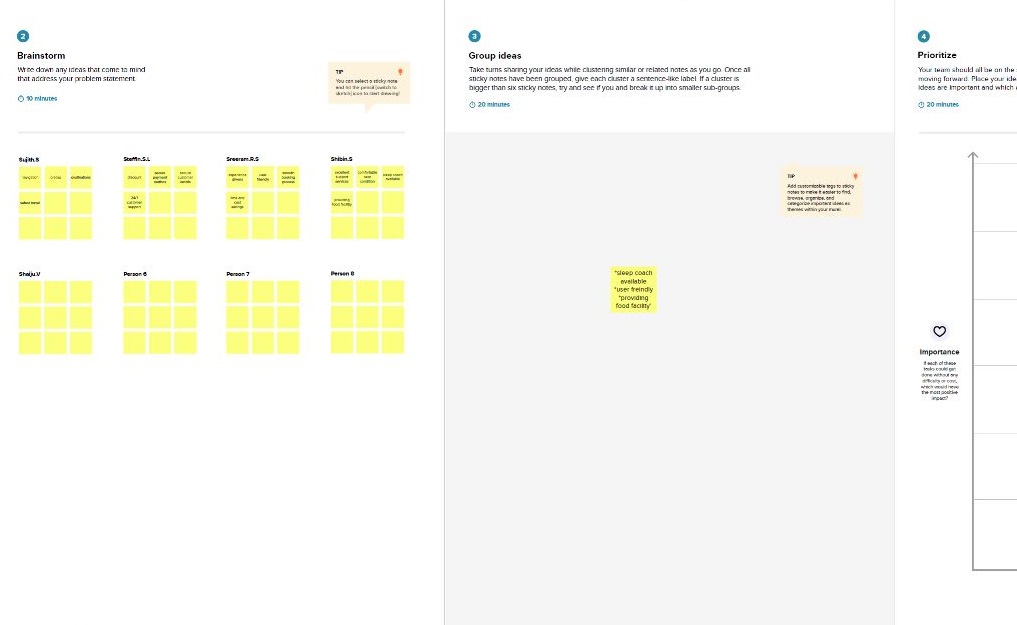
**2.Problem Definition & Design Thinking**

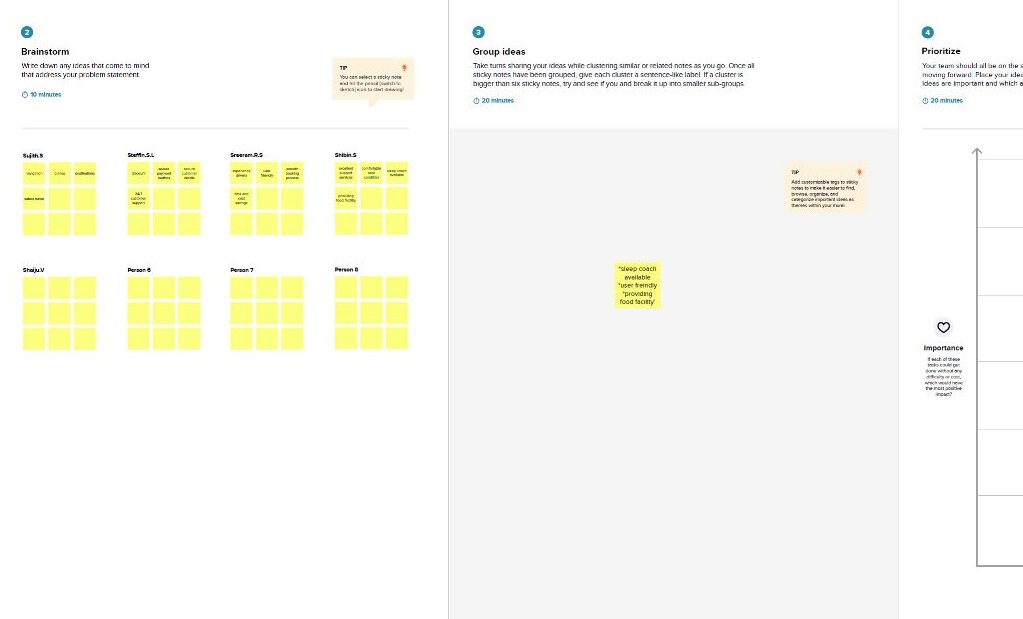
**2.1 Empathy Map**

****

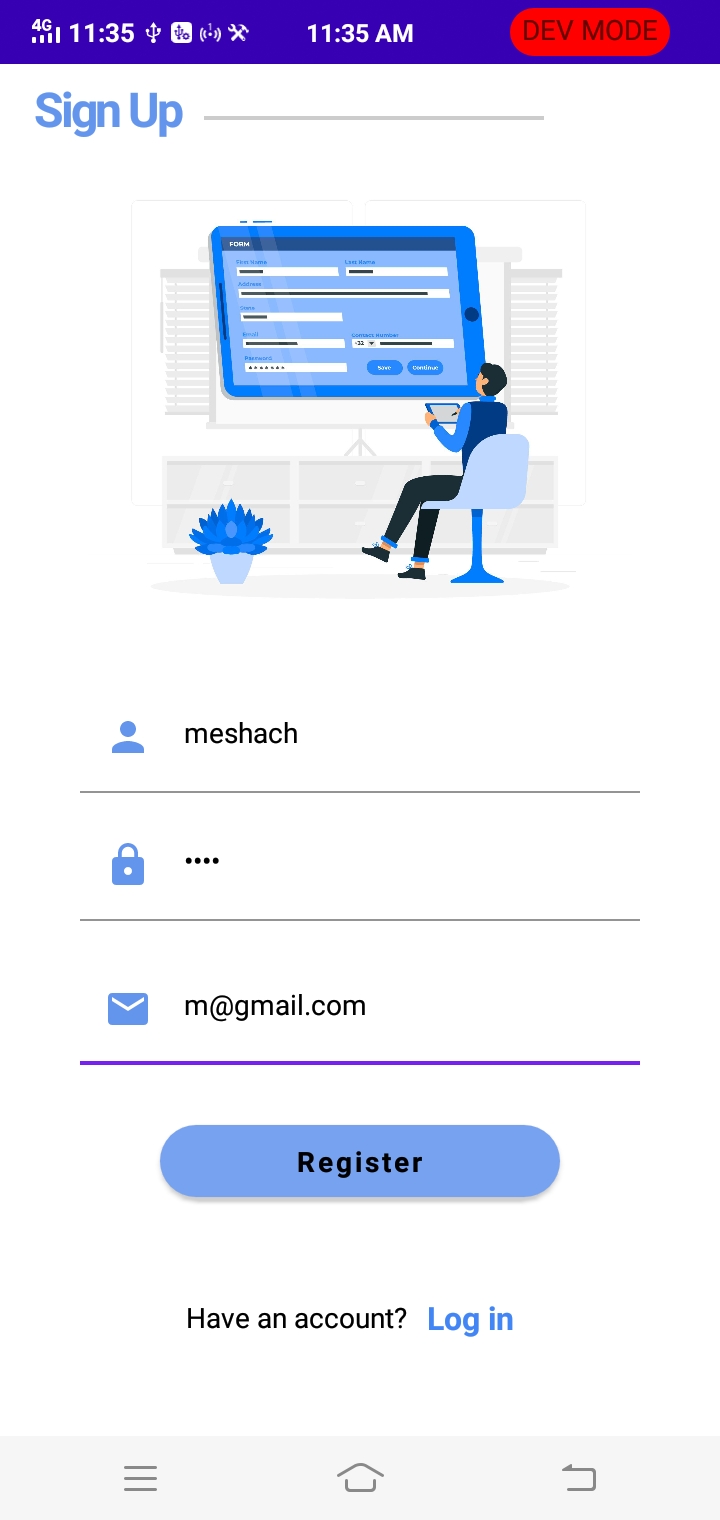
**2.2 Ideation and Brainstorming Map**

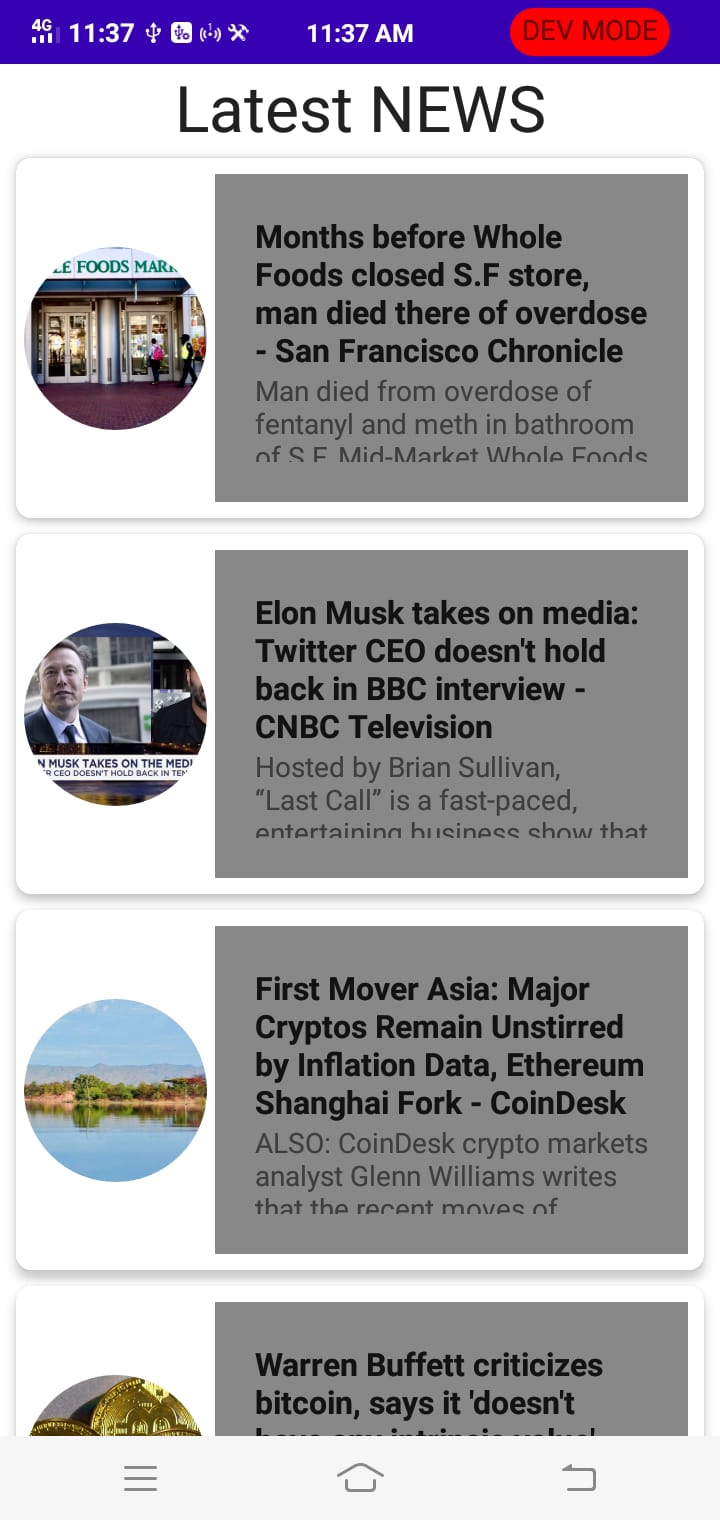
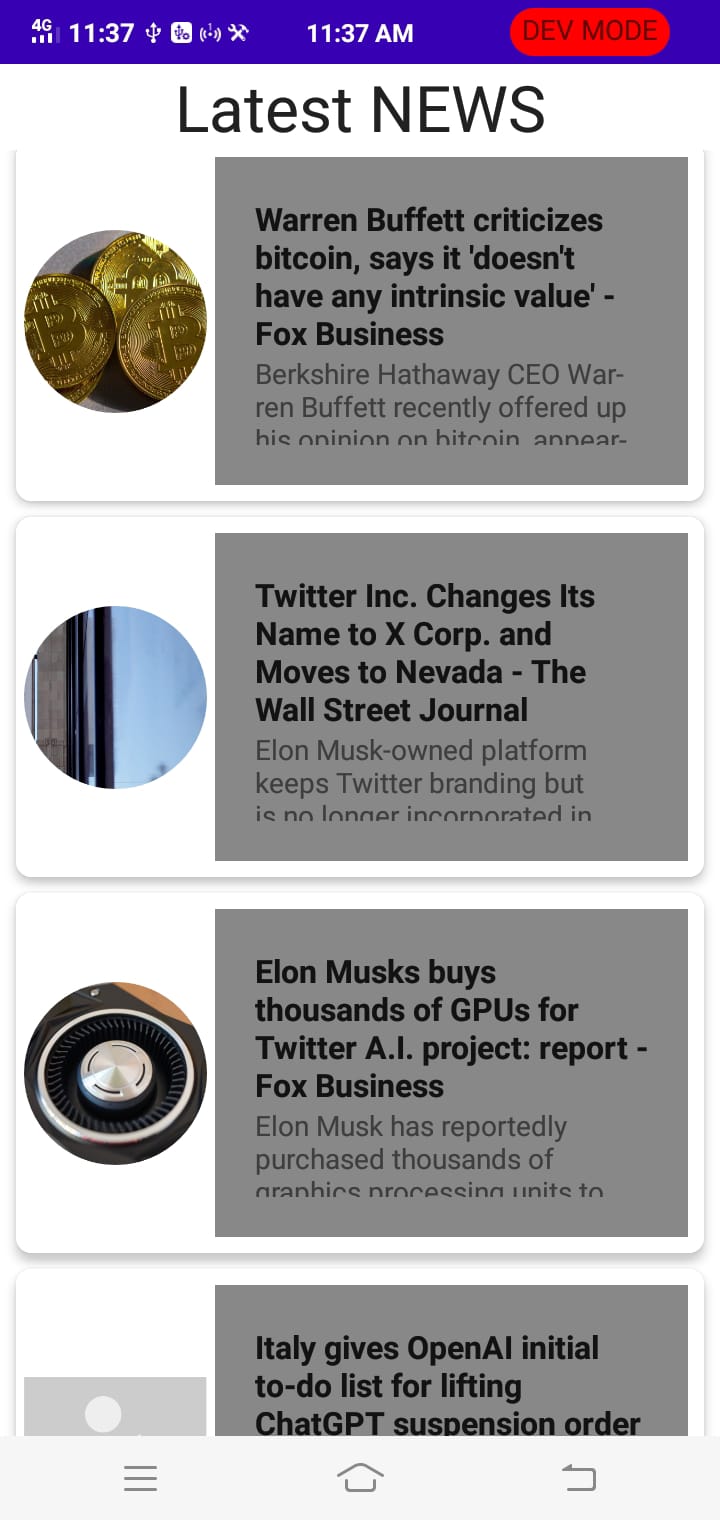
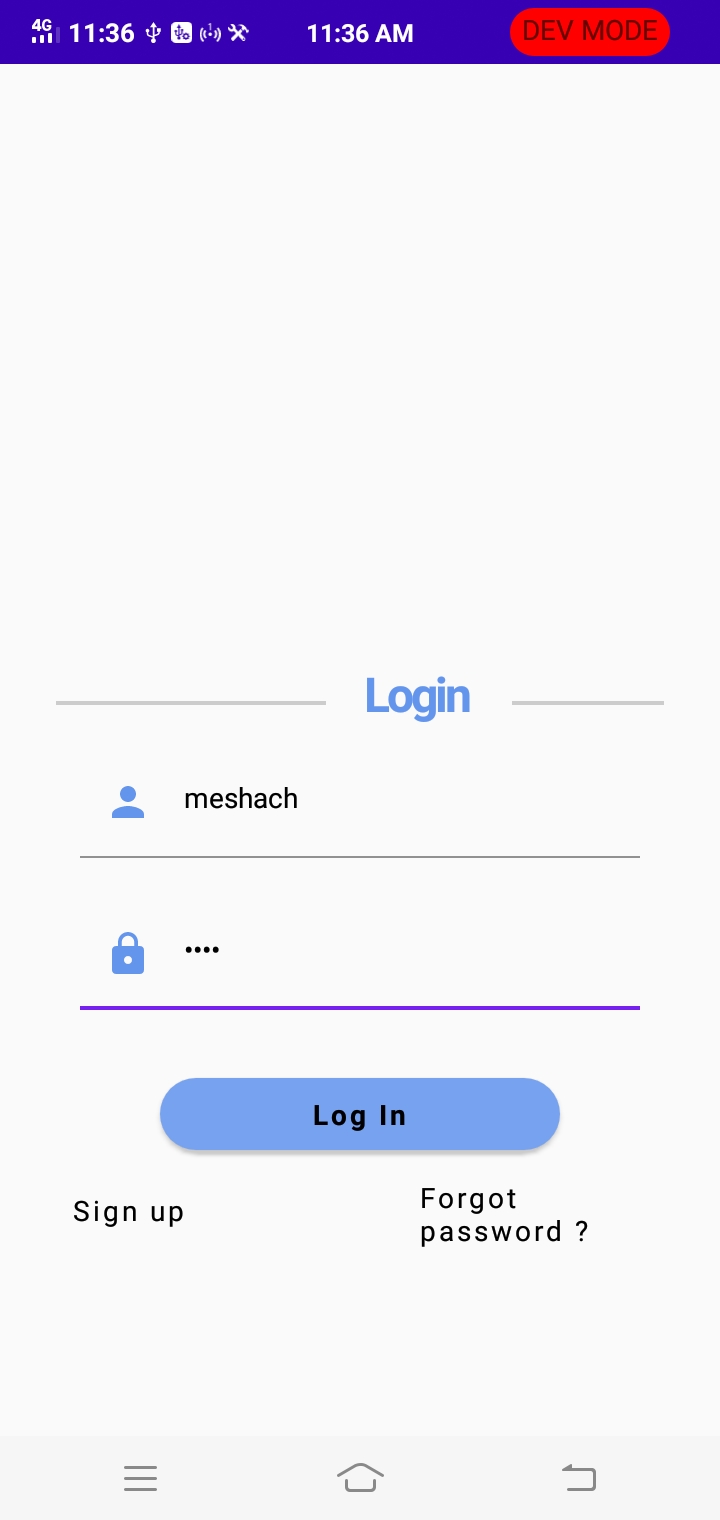






**RESULT**

****

****

**ADVANTAGES & DISADVANTAGES**

**Advantages**

* Increased user engagement: By providing a personalized experience and easy access to the latest news, the application is likely to increase user engagement. Users are more likely to return to the application frequently, which can help increase traffic and user retention.
* Improved user retention: The application's personalized experience and user-friendly interface can help improve user retention. Users are more likely to continue using the application if they have a positive experience and find the content useful and relevant.
* Potential for monetization: The application can potentially generate revenue through advertising and sponsored content. As the application gains popularity and attracts more users, advertisers may be willing to pay for ad space or sponsor certain articles or features.
* Valuable data insights: The application's machine learning algorithms can provide valuable data insights into user behavior and preferences. This data can be used to improve the application's functionality and user experience, as well as inform business decisions.
* Competitive advantage: In a crowded market of news applications and websites, the personalized experience and easy access to the latest news provided by this application can help it stand out and gain a competitive advantage.

**Disadvantages**

* Dependence on external sources: The application depends on external news sources for content, which means that any issues with those sources (such as downtime or changes in their APIs) could negatively impact the application's functionality.
* Competition: The market for news applications and websites is highly competitive, and it can be difficult to attract and retain users in such a crowded space.
* Ad-blocking software: Users may choose to use ad-blocking software, which can reduce or eliminate the application's revenue from advertising.
* Privacy concerns: As with any application that collects user data, there may be privacy concerns from users who are uncomfortable with their data being used for machine learning algorithms or targeted advertising.
* Technical issues: The application may encounter technical issues such as bugs, errors, or server downtime, which could negatively impact the user experience.

**APPLICATIONS**

* Personalization: The application could use machine learning algorithms to personalize the news experience for each user. By analyzing user behavior and preferences, the application could recommend articles and sources that are most relevant to the user's interests.
* Search functionality: Users could search for news articles based on keywords, topics, or sources. The application could also include advanced search options, such as filtering by date range or by type of news (e.g., sports, politics, entertainment).
* Sharing: Users could share articles with their friends and family through social media, email, or messaging apps.
* Bookmarking: Users could bookmark articles for later reading or reference.
* Notifications: The application could send push notifications to users for breaking news or for articles that match their interests.
* Multiple sources: The application could aggregate news from various sources, such as major news outlets, local news sources, and specialty publications.
* Accessibility: The application could be accessible on multiple devices, such as desktops, laptops, tablets, and mobile devices.
* Advertisement: The application could generate revenue through advertising, which could be targeted based on user preferences and behavior.

**CONCLUSION**

In conclusion, the application for accessing the latest headlines offers several advantages, including a centralized platform for news, real-time updates, personalized recommendations, a user-friendly interface, accessibility across devices, and potential revenue through advertising. While there are some potential disadvantages such as competition, dependence on external sources, and privacy concerns, these can be managed with proper planning and execution. Overall, the application offers a valuable solution for users seeking to stay informed and up-to-date on the latest news, while also providing opportunities for revenue and business growth.

**FUTURE SCOPE**

* Integration with emerging technologies: The application could integrate with emerging technologies such as voice assistants, augmented reality, or virtual reality to provide users with new and innovative ways to consume news content.
* Expansion of content formats: The application could expand beyond traditional news articles and incorporate other formats such as podcasts, videos, or interactive content.
* Increased personalization: The application could continue to improve its personalization algorithms and offer even more tailored recommendations based on user behavior, preferences, and location.
* Partnerships with news organizations: The application could establish partnerships with news organizations to provide exclusive content, access to experts, or other value-add services for users.
* Internationalization: The application could expand its reach beyond a single country or language and offer news content from around the world in multiple languages.
* Data analytics: The application could use its data analytics capabilities to provide valuable insights to news organizations, advertisers, or other stakeholders in the news industry.

Overall, the future scope for the application is vast and exciting, with potential opportunities to innovate, expand, and improve the user experience. By staying on top of emerging trends and technologies, partnering with key players in the news industry, and continuing to personalize the news experience for users, the application can remain a valuable and relevant platform for accessing the latest headlines.

**Appendix**

1. **Source code**

AndroidManifest.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>

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=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!!

}

}

}

Articles.kt

@@ -0,0 +1,12 @@

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,

)

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) }

)

}

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)

}

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

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) }

)

}

}

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()

}

}

}

}

Model.kt

package com.example.newsheadlines

data class Movie(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()

)

RegistrationActivity.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)

}

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

)

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

@@ -0,0 +1,19 @@

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

}

}

Color.kt

package com.example.newsheadlines.ui.theme

import androidx.compose.ui.graphics.Color

val Purple200 = Color(0xFFBB86FC)

val Purple500 = Color(0xFF6200EE)

val Purple700 = Color(0xFF3700B3)

val Teal200 = Color(0xFF03DAC5)

Shape.kt

package com.example.newsheadlines.ui.theme

import androidx.compose.foundation.shape.RoundedCornerShape

import androidx.compose.material.Shapes

import androidx.compose.ui.unit.dp

val Shapes = Shapes(

small = RoundedCornerShape(4.dp),

medium = RoundedCornerShape(4.dp),

large = RoundedCornerShape(0.dp)

)

Theme.kt

@@ -0,0 +1,47 @@

package com.example.newsheadlines.ui.theme

import androidx.compose.foundation.isSystemInDarkTheme

import androidx.compose.material.MaterialTheme

import androidx.compose.material.darkColors

import androidx.compose.material.lightColors

import androidx.compose.runtime.Composable

private val DarkColorPalette = darkColors(

primary = Purple200,

primaryVariant = Purple700,

secondary = Teal200

)

private val LightColorPalette = lightColors(

primary = Purple500,

primaryVariant = Purple700,

secondary = Teal200

/\* Other default colors to override

background = Color.White,

surface = Color.White,

onPrimary = Color.White,

onSecondary = Color.Black,

onBackground = Color.Black,

onSurface = Color.Black,

\*/

)

@Composable

fun NewsHeadlinesTheme(

darkTheme: Boolean = isSystemInDarkTheme(),

content: @Composable () -> Unit

) {

val colors = if (darkTheme) {

DarkColorPalette

} else {

LightColorPalette

}

MaterialTheme(

colors = colors,

typography = Typography,

shapes = Shapes,

content = content

)

}

Type.kt

package com.example.newsheadlines.ui.theme

import androidx.compose.material.Typography

import androidx.compose.ui.text.TextStyle

import androidx.compose.ui.text.font.FontFamily

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

import androidx.compose.ui.unit.sp

// Set of Material typography styles to start with

val Typography = Typography(

body1 = TextStyle(

fontFamily = FontFamily.Default,

fontWeight = FontWeight.Normal,

fontSize = 16.sp

)

/\* Other default text styles to override

button = TextStyle(

fontFamily = FontFamily.Default,

fontWeight = FontWeight.W500,

fontSize = 14.sp

),

caption = TextStyle(

fontFamily = FontFamily.Default,

fontWeight = FontWeight.Normal,

fontSize = 12.sp

)

\*/

)