APPLICATION DEVELOPMENT WITH ANDROID STUDIO

(MINI PROJECT)

Roll. No.: B30	Name: Bhatt Pranjal
Class: TE B COPMS	Batch: B2
Date of Experiment:	Date of Submission:
Grade:	

NEWS APP

Aim: To develop a **News App** that provides users with real-time, categorized, and news updates by integrating external news APIs. The application will feature a user-friendly interface, efficient search functionality, and customizable news feeds to enhance the reading experience.

Objectives:

 To Introduce Android App Development Concepts

- Familiarize students with **Android Studio**, **Kotlin**, and **Jetpack Compose** for app development.
- Understand **project structure, Gradle build system, and manifest configurations**.

 To Implement a Real-Time News Fetching System

- Integrate **NewsAPI** (or similar sources) using **Retrofit** to fetch the latest news dynamically.
- Use **JSON parsing** with Gson or Moshi to process and display news articles efficiently.

 To Develop an Intuitive & Engaging User Interface

- Design an easy-to-navigate UI using **Jetpack Compose / XML layouts**.
- Implement **RecyclerView** for displaying a list of news articles smoothly.
- Provide a **dark mode** feature for enhanced readability.

 To Implement News Filtering & Personalization

- Enable users to filter news by **categories** (e.g., Sports, Technology, Business).
- Implement a **search bar** for quick access to specific news articles.
- Allow users to **bookmark** articles for offline reading.

 To Enhance User Experience with Push Notifications

- Integrate **Firebase Cloud Messaging (FCM)** to notify users about breaking news.
- Ensure notifications are **timely and relevant** to avoid user fatigue.

 To Implement Efficient Data Storage & Performance Optimization

- Use **Room Database** for offline caching and smooth content retrieval.
- Optimize app performance with **background processing using WorkManager**.
- Reduce data usage by implementing **lazy loading and pagination**.

 To Introduce Deployment & Testing Practices

- Debug and test the app using **Android Emulator and real devices**.
- Learn how to generate a **signed APK** for deployment on the **Google Play Store**.

Outcome: After the successful completion of this project, students will be able to:

1. **Develop and Deploy an Android News Application**
   - Build a fully functional **News App** using **Android Studio** and deploy it on real devices.
2. **Integrate and Handle APIs Efficiently**
   - Fetch and display real-time news using **NewsAPI** with **Retrofit/Volley** for seamless data retrieval.
3. **Design and Implement a User-Friendly Interface**
   - Develop an intuitive UI using **Jetpack Compose/XML layouts** with smooth navigation and category-based filtering.
4. **Enhance User Experience with Key Features**
   - Implement **search functionality, real-time filtering, bookmarking for offline reading, and dark mode**.
   - Use **Firebase Cloud Messaging (FCM)** to enable push notifications for breaking news.
5. **Implement Local Data Storage and Performance Optimization**
   - Utilize **Room Database** for offline caching, ensuring users can access saved articles without an internet connection.
   - Optimize app performance using **background processing with WorkManager and pagination techniques**.
6. **Test and Debug Applications Effectively**
   - Use **Android Emulator, Logcat, and Profiler** for debugging and performance optimization.

7. **Deploy the Application Successfully**
    - Learn to generate a **signed APK** and prepare the app for **Google Play Store deployment**.

Theory:

1. Introduction to News Apps

A **News App** is a mobile application that provides users with real-time access to the latest news and updates from various sources. With the increasing reliance on mobile devices for information consumption, news apps have become a crucial medium for delivering content in an efficient, user-friendly manner.

This project involves building an Android-based **News App** that fetches real-time news articles using **NewsAPI** and presents them in a structured, user-friendly format. The app will allow users to browse news articles, filter them based on categories, search for specific topics, and even save articles for offline reading.

---

2. Technologies Used in News App

a) Android Studio

Android Studio is the official **Integrated Development Environment (IDE)** for Android development. It provides tools for designing, coding, testing, and debugging Android applications. Features like **Gradle Build System, Layout Editor, and Emulator** make it a powerful tool for app development.

b) Programming Language: Kotlin

Kotlin is the preferred language for modern Android development due to its **concise syntax, enhanced safety features, and compatibility** with Java. It simplifies UI development and improves performance compared to traditional Java-based apps.

c) Jetpack Compose & UI Components

- **Jetpack Compose** is a modern UI toolkit that simplifies UI design by using a declarative approach.
- The app includes essential UI components like **RecyclerView, CardView, and ViewPager** for displaying news articles in a visually appealing way.

d) API Integration with Retrofit

- **Retrofit** is a powerful HTTP client for handling API requests.
- The app fetches news data from **NewsAPI** in **JSON format**, which is parsed using **Gson or Moshi**.

e) Local Database with Room

- **Room Database** is used for offline storage of bookmarked news articles.
- This ensures users can read saved articles even without an internet connection.

f) Firebase Cloud Messaging (FCM)

- Push notifications are implemented using **Firebase Cloud Messaging (FCM)** to keep users informed about breaking news.

---

3. Architecture of the News App

a) Model-View-ViewModel (MVVM) Architecture

The project follows the **MVVM architecture** to ensure better code organization and maintainability:

- **Model:** Handles data processing and API integration.
- **View:** Represents the UI components and displays data to users.
- **ViewModel:** Acts as an intermediary, managing UI-related data and logic efficiently.

b) Data Flow and API Calls

- The app fetches data from **NewsAPI** and displays it using **RecyclerView**.
- API requests are handled by **Retrofit**, which retrieves data in JSON format.
- The data is then parsed and displayed on the UI.

SOFTWARE/HARDWARE Requirements:

Software Requirements

1. **Development Platform:**
   - **IDE:** Android Studio (latest stable version)
   - **Build System:** Gradle
   - **Version Control:** Git/GitHub
2. **Programming Languages:**
   - **Primary:** Kotlin
   - **Secondary:** Java (for legacy support)
3. **Architecture & Libraries:**
   - **Architecture:** MVVM (Model-View-ViewModel)
   - **Android Jetpack Components:**
     - Room (local SQLite database for offline articles)
     - Retrofit (API calls to news sources)
     - Kotlin Coroutines (asynchronous tasks)
     - Jetpack Compose (modern UI design)
     - WorkManager (background sync for real-time updates)

   - **Dependency Injection:** Hilt or Koin
4. **APIs & Data Handling:**
   - **News API Integration:** RESTful APIs (e.g., NewsAPI, GNews)
   - **JSON Parsing:** Moshi/Gson
   - **Authentication:** Firebase Auth (if user accounts are added later)
5. **Additional Tools:**
   - **Push Notifications:** Firebase Cloud Messaging (FCM)
   - **Analytics:** Firebase Analytics (user engagement tracking)
   - **Testing Frameworks:**
     - JUnit, Espresso (unit/UI testing)
     - MockK (mocking for Coroutines)

Hardware Requirements

1. **Development Hardware:**
   - **Processor:** Intel Core i5 (2GHz+) or equivalent AMD
   - **RAM:** 8GB DDR4 (16GB recommended for smoother performance)
   - **Storage:** 1TB HDD/256GB SSD (for IDE, emulators, and tools)
2. **Target User Devices:**
   - **OS:** Android 5.0 (Lollipop) and above
   - **Processor:** Quad-core ARM or higher (for smooth real-time updates)
   - **RAM:** 2GB minimum (4GB recommended for multitasking)
   - **Storage:** 100MB+ free space (for app + cached articles)
   - **Network:** Wi-Fi/4G/5G (for real-time news fetching)

CODE Implementation: ***(Paste your Code script related to your case study completed)***

a) MAINACTIVITY

```
package com.example.newsroom

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.viewModels
import androidx.compose.foundation.background
import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.foundation.layout.Box
import androidx.compose.material3.MaterialTheme
import androidx.compose.runtime.SideEffect
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.core.splashscreen.SplashScreen.Companion.installSplashScreen
import androidx.core.view.WindowCompat
import androidx.lifecycle.lifecycleScope
import com.example.newsroom.data.local.NewsDao
import com.example.newsroom.domain.model.Article
import com.example.newsroom.domain.model.Source
import com.example.newsroom.presentation.navgraph.NavGraph
import com.example.newsroom.ui.theme.NewsRoomTheme
import com.google.accompanist.systemuicontroller.rememberSystemUiController
import dagger.hilt.android.AndroidEntryPoint
import kotlinx.coroutines.launch
import javax.inject.Inject

@AndroidEntryPoint
class MainActivity : ComponentActivity() {

    val viewModel by viewModels<MainViewModel>()
    @Inject
    lateinit var dao: NewsDao
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        WindowCompat.setDecorFitsSystemWindows(window, false)

        lifecycleScope.launch {
            dao.upsert(
                Article(
                    author = "",
                    title = "Coinbase says Apple blocked its last app release on NFTs in Wallet ...
- CryptoSaurus",
                    description = "Coinbase says Apple blocked its last app release on NFTs in
Wallet ... - CryptoSaurus",
                    content = "We use cookies and data to Deliver and maintain Google services
Track outages and protect against spam, fraud, and abuse Measure audience engagement
and site statistics to unde… [+1131 chars]",
                    publishedAt = "2023-06-16T22:24:33Z",
                    source = Source(
                        id = "", name = "bbc"
                    ),
                    url =
"https://consent.google.com/ml?continue=https://news.google.com/rss/articles/CBMiaWh
0dHBzOi8vY3J5cHRvc2F1cnVzLnRlY2gvY29pbmJhc2Utc2F5cy1hcHBsZS1ibG9ja2V
kLWl0cy1sYXN0LWFwcC1yZWxlYXNlLW9uLW5mdHMtaW4td2FsbGV0LXJldXRlc
nMtY29tL9IBAA?oc%3D5&gl=FR&hl=en-US&cm=2&pc=n&src=1",
                urlToImage =
"https://media.wired.com/photos/6495d5e893ba5cd8bbdc95af/191:100/w_1280,c_limit/T
he-EU-Rules-Phone-Batteries-Must-Be-Replaceable-Gear-2BE6PRN.jpg"

            )
        )
    }

    installSplashScreen().apply {
        setKeepOnScreenCondition{
            viewModel._splashCondition
        }
    }

    setContent {
        NewsRoomTheme {
            val isSystemInDarkMode = isSystemInDarkTheme()
            val systemController = rememberSystemUiController()

            SideEffect {
                systemController.setSystemBarsColor(
                    color = Color.Transparent,
                    darkIcons = !isSystemInDarkMode
                )
            }

            Box (modifier = Modifier.background(color =
MaterialTheme.colorScheme.background)){
                val startDestination = viewModel.startDestination
                NavGraph(startDestination = startDestination)

            }
        }
    }
  }
}
```

b) Article card

```
package com.example.newsroom.presentation.common

import android.content.res.Configuration.UI_MODE_NIGHT_YES
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.layout.width
import androidx.compose.material3.Icon
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.colorResource
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextOverflow
import androidx.compose.ui.tooling.preview.Preview
import coil.compose.AsyncImage
import coil.request.ImageRequest
import com.example.newsroom.R
import com.example.newsroom.domain.model.Article
import com.example.newsroom.domain.model.Source
import com.example.newsroom.presentation.Dimension.ArticleCardSize
import com.example.newsroom.presentation.Dimension.ExtraSmallPadding
import com.example.newsroom.presentation.Dimension.ExtraSmallPadding2
import com.example.newsroom.presentation.Dimension.SmallIconSize
import com.example.newsroom.ui.theme.NewsRoomTheme

@Composable
fun ArticleCard(
    modifier: Modifier = Modifier,
    article: Article,
    onClick:() -> Unit
){
    val context = LocalContext.current
```

```
Row(modifier = Modifier.clickable { onClick() }){

    AsyncImage(
        modifier = Modifier
            .size(ArticleCardSize)
            .clip(MaterialTheme.shapes.medium),
        model = ImageRequest.Builder(context).data(article.urlToImage).build(),
        contentDescription = null,
        contentScale = ContentScale.Crop
    )

    Column(
        verticalArrangement = Arrangement.SpaceAround,
        modifier = Modifier
            .padding(horizontal = ExtraSmallPadding)
            .height(ArticleCardSize)
    ) {
        Text(
            text = article.title,
            style = MaterialTheme
                .typography.bodyMedium,
            color = colorResource(
                id = R.color.text_title
            ),
            maxLines = 2,
            overflow = TextOverflow.Ellipsis
        )

        Row (
            verticalAlignment = Alignment.CenterVertically){
            article.source.name?.let {
                Text(
                    text = it,
                    style = MaterialTheme
                        .typography.labelMedium.copy(fontWeight = FontWeight.Bold),
                    color = colorResource(
                        id = R.color.body
                    )
                )
            }
            Spacer(modifier = Modifier.width(ExtraSmallPadding2))
            Icon(
                painter = painterResource(id = R.drawable.ic_time),
                contentDescription = null,
                modifier = Modifier.size(SmallIconSize),
                tint = colorResource(id = R.color.body)
            )
            Spacer(modifier = Modifier.width(ExtraSmallPadding2))

            article.publishedAt?.let {
                Text(
                    text = it,
                    style = MaterialTheme
                        .typography.labelMedium.copy(fontWeight = FontWeight.Bold),
                    color = colorResource(
                        id = R.color.body
                    )
                )
            }
        }
    }
}
}

@Preview(showBackground = true)
@Composable
fun ArticleCardPreview(){
    NewsRoomTheme {
        ArticleCard(article = Article(
            author = "",
            content = "",
            description = "",
            publishedAt = "2 hours",
            source = Source(id = "", name = "BBC"),
            title = "Her traint broke down. Her phone died. And the she met her saver in a",
            url = "",
            urlToImage = ""

        )) {

        }
    }
}
```

c) Article list

```
package com.example.newsroom.presentation.common

import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.PaddingValues
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import androidx.paging.LoadState
import androidx.paging.compose.LazyPagingItems
import com.example.newsroom.domain.model.Article
import com.example.newsroom.presentation.Dimension.MediumPadding1
import com.loc.newsapp.presentation.common.EmptyScreen

@Composable
fun ArticleList(
    modifier: Modifier = Modifier,
    article: LazyPagingItems<Article>,
    onClick:(Article) -> Unit
){
    val handelPagingResult = handelPagingResult(article = article)
    if(handelPagingResult){
        LazyColumn(
            modifier = modifier.fillMaxSize(),
            verticalArrangement = Arrangement.spacedBy(MediumPadding1),
            contentPadding = PaddingValues(all = 4.dp)
        ){
            items(count = article.itemCount){
                article[it].let {
                    if (it != null) {
                        ArticleCard(modifier = modifier, article = it, onClick = {onClick(it)})
                    }
                }
            }
        }
    }
}

@Composable
fun ArticleList(
    modifier: Modifier = Modifier,
    article: List<Article>,
    onClick:(Article) -> Unit
){


    LazyColumn(
            modifier = modifier.fillMaxSize(),
            verticalArrangement = Arrangement.spacedBy(MediumPadding1),
            contentPadding = PaddingValues(all = 4.dp)
    ){
        items(count = article.size){
            val article = article[it]
            if (it != null) {
                ArticleCard(modifier = modifier, article = article, onClick = {onClick(article)})
            }
        }
    }
}



@Composable
fun handelPagingResult(
    article: LazyPagingItems<Article>,
):Boolean{

    val loadState = article.loadState
    val error = when{
        loadState.refresh is LoadState.Error -> loadState.refresh as LoadState.Error
        loadState.prepend is LoadState.Error -> loadState.prepend as LoadState.Error
        loadState.append is LoadState.Error -> loadState.append as LoadState.Error
        else -> null
    }

    return when{
        loadState.refresh is LoadState.Loading -> {
            ShimmerEffect()
            false
        }
        error != null ->{
            EmptyScreen()
            false
        }
        else -> {
            true
        }
    }
}

@Composable
private fun ShimmerEffect(){
    Column(verticalArrangement = Arrangement.spacedBy(MediumPadding1)) {
        repeat(10){
            ArticleCardShimmerEffect(
                modifier = Modifier.padding(horizontal = MediumPadding1)
            )
        }

    }
}
```

d) Empty Screen

```
package com.loc.newsapp.presentation.common

import android.content.res.Configuration.UI_MODE_NIGHT_YES
import androidx.compose.animation.core.animateFloatAsState
import androidx.compose.animation.core.tween
import androidx.compose.foundation.isSystemInDarkTheme
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.foundation.layout.size
import androidx.compose.material3.Icon
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.LaunchedEffect
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.alpha
import androidx.compose.ui.graphics.Color.Companion.DarkGray
import androidx.compose.ui.graphics.Color.Companion.LightGray
import androidx.compose.ui.graphics.DefaultAlpha
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.paging.LoadState
import com.example.newsroom.R

//import com.loc.newsapp.R
import java.net.ConnectException
import java.net.SocketTimeoutException

@Composable
fun EmptyScreen(error: LoadState.Error? = null) {

    var message by remember {
        mutableStateOf(parseErrorMessage(error = error))
    }

    var icon by remember {
        mutableStateOf(R.drawable.ic_network_error)
    }

    if (error == null){
        message = "You have not saved news so far !"
        icon = R.drawable.ic_search_document
    }

    var startAnimation by remember {
        mutableStateOf(false)
    }

    val alphaAnimation by animateFloatAsState(
        targetValue = if (startAnimation) 0.3f else 0f,
        animationSpec = tween(durationMillis = 1000)
    )

    LaunchedEffect(key1 = true) {
        startAnimation = true
    }

    EmptyContent(alphaAnim = alphaAnimation, message = message, iconId = icon)

}

@Composable
fun EmptyContent(alphaAnim: Float, message: String, iconId: Int) {
    Column(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment = Alignment.CenterHorizontally,
        verticalArrangement = Arrangement.Center
    ) {
        Icon(
            painter = painterResource(id = iconId),
            contentDescription = null,
            tint = if (isSystemInDarkTheme()) LightGray else DarkGray,
            modifier = Modifier
                .size(120.dp)
                .alpha(alphaAnim)
        )
        Text(
            modifier = Modifier
            .padding(10.dp)
            .alpha(alphaAnim),
          text = message,
          style = MaterialTheme.typography.bodyMedium,
          color = if (isSystemInDarkTheme()) LightGray else DarkGray,
        )
    }
}


fun parseErrorMessage(error: LoadState.Error?): String {
    return when (error?.error) {
        is SocketTimeoutException -> {
            "Server Unavailable."
        }

        is ConnectException -> {
            "Internet Unavailable."
        }

        else -> {
            "Unknown Error."
        }
    }
}
@Preview(showBackground = true)
@Preview(showBackground = true, uiMode = UI_MODE_NIGHT_YES)
@Composable
fun EmptyScreenPreview() {
    EmptyContent(alphaAnim = 0.3f, message = "Internet
Unavailable.",R.drawable.ic_network_error)
}
```

e) News button

```
package com.example.newsroom.presentation.common

import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.Button
import androidx.compose.material3.ButtonDefaults
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.material3.TextButton
import androidx.compose.runtime.Composable
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
```

```
@Composable
fun NewsButton(
    text: String,
    onClick: () -> Unit
){
    Button(
        onClick = onClick, colors = ButtonDefaults.buttonColors(
            containerColor = MaterialTheme.colorScheme.primary,
            contentColor = Color.White
        ),
        shape = RoundedCornerShape(size = 6.dp)
    ) {
        Text(
            text = text,
            style = MaterialTheme
                .typography
                .labelMedium
                .copy(fontWeight = FontWeight.SemiBold)
        )
    }
}

@Composable
fun NewsTextButton(
    text: String,
    onClick: () -> Unit
){
    TextButton(onClick = onClick) {
        Text(
            text = text,
            style = MaterialTheme.typography.labelMedium.copy(fontWeight =
FontWeight.SemiBold),
            color = Color.White
        )
    }
}
```

f) Seach Button

```
package com.example.newsroom.presentation.common

import android.content.res.Configuration.UI_MODE_NIGHT_YES
import android.util.Log
import androidx.compose.foundation.border
import androidx.compose.foundation.clickable
import androidx.compose.foundation.interaction.MutableInteractionSource
import androidx.compose.foundation.interaction.collectIsPressedAsState
import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.text.KeyboardActions
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material3.ExperimentalMaterial3Api
import androidx.compose.material3.Icon
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.material3.TextField
import androidx.compose.material3.TextFieldDefaults
import androidx.compose.runtime.Composable
import androidx.compose.runtime.LaunchedEffect
import androidx.compose.runtime.remember
import androidx.compose.runtime.rememberCoroutineScope
import androidx.compose.ui.Modifier
import androidx.compose.ui.composed
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.res.colorResource
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.input.ImeAction
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import com.example.newsroom.R
import com.example.newsroom.presentation.Dimension.IconSize
import com.example.newsroom.ui.theme.NewsRoomTheme


@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun SearchBar(
    modifier: Modifier = Modifier,
    text: String,
    readOnly: Boolean,
    onClick: (() -> Unit)? = null,
    onValueChange: (String) -> Unit,
    onSearch: () -> Unit
) {

    val interactionSource = remember {
        MutableInteractionSource()
    }
    val isClicked = interactionSource.collectIsPressedAsState().value
    LaunchedEffect(key1 = isClicked){
        if(isClicked){
            onClick?.invoke()
        }
    }

    Box(modifier = modifier) {
        TextField(
            modifier = Modifier
                .fillMaxWidth()
                .searchBar(),
            value = text,
            onValueChange = onValueChange,
            readOnly = readOnly,
            leadingIcon = {
                Icon(
                    painter = painterResource(id = R.drawable.ic_search),
                    contentDescription = null,
                    modifier = Modifier.size(IconSize),
                    tint = colorResource(id = R.color.body)
                )
            },
            placeholder = {
                Text(
                    text = "Search",
                    style = MaterialTheme.typography.bodySmall,
                    color = colorResource(id = R.color.placeholder)
                )
            },
            shape = MaterialTheme.shapes.medium,
            colors = TextFieldDefaults.textFieldColors(
                containerColor = colorResource(id = R.color.input_background),
                focusedTextColor = if (isSystemInDarkTheme()) Color.White else
Color.Black,
                unfocusedTextColor = if (isSystemInDarkTheme()) Color.White else
Color.Black,
                cursorColor = if (isSystemInDarkTheme()) Color.White else Color.Black,
                disabledIndicatorColor = Color.Transparent,
                errorIndicatorColor = Color.Transparent,
                focusedIndicatorColor = Color.Transparent,
                unfocusedIndicatorColor = Color.Transparent
            ),
            singleLine = true,
            keyboardOptions = KeyboardOptions(imeAction = ImeAction.Search),
            keyboardActions = KeyboardActions(
                onSearch = {
                    onSearch()
                }
            ),
            textStyle = MaterialTheme.typography.bodySmall,
            interactionSource = interactionSource
        )
    }
}

fun Modifier.searchBar(): Modifier = composed {
    if (!isSystemInDarkTheme()) {
        border(
            width = 1.dp,
            color = Color.Black,
            shape = MaterialTheme.shapes.medium
        )
    } else {
        this
    }
}

@Preview(showBackground = true)
@Preview(showBackground = true, uiMode = UI_MODE_NIGHT_YES)
@Composable
fun SearchBarPreview() {
    NewsRoomTheme {
        SearchBar(text = "", onValueChange = {}, readOnly = false) {

        }
    }
}
```

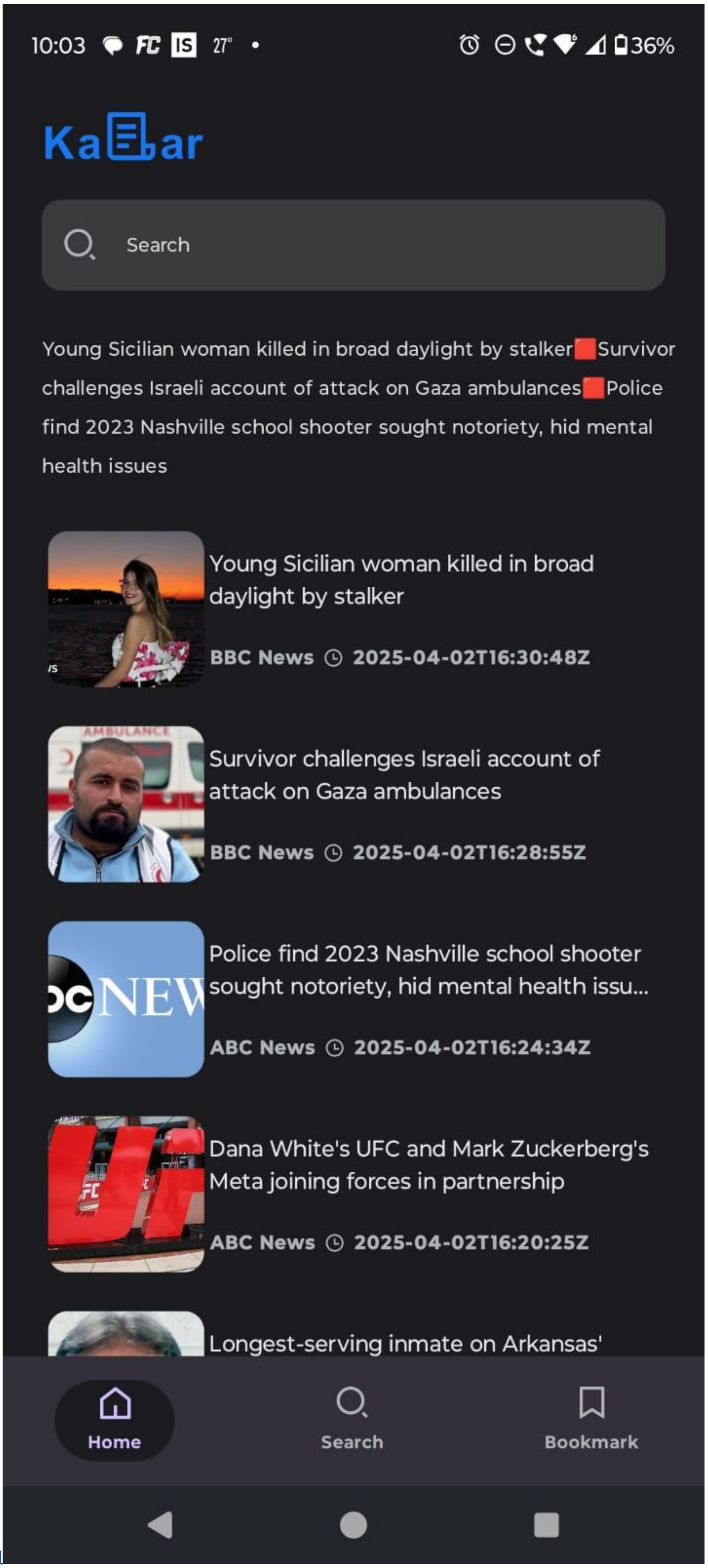
g) Homescreen.kt

```
package com.example.newsroom.presentation.home

import androidx.compose.foundation.Image
import androidx.compose.foundation.basicMarquee
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.statusBarsPadding
import androidx.compose.foundation.layout.width
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.derivedStateOf
import androidx.compose.runtime.remember
import androidx.paging.compose.LazyPagingItems
import com.example.newsroom.domain.model.Article
import androidx.compose.runtime.getValue
import androidx.compose.ui.Modifier
import androidx.compose.ui.res.colorResource
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.example.newsroom.R
import com.example.newsroom.presentation.Dimension.MediumPadding1
import com.example.newsroom.presentation.common.ArticleList
import com.example.newsroom.presentation.common.SearchBar
import com.example.newsroom.presentation.navgraph.Route

@Composable
fun HomeScreen(
    articles: LazyPagingItems<Article>,
    navigateToSearch: () -> Unit,
    navigateToDetails: (Article) -> Unit
){
    val titles by remember {
        derivedStateOf {
            if (articles.itemCount > 10){
                articles.itemSnapshotList.items
                    .slice(IntRange(start = 0, endInclusive = 2))
                    .joinToString(separator = "\uD83d\uDFE5"){it.title}
            }
            else{
                ""
            }
        }
    }

    Column(modifier = Modifier
        .fillMaxSize()
        .padding(top = MediumPadding1)
        .statusBarsPadding()
    ) {
        Image(
            painter = painterResource(id = R.drawable.ic_logo),
            contentDescription = null,
            modifier = Modifier
                .width(150.dp)
                .height(30.dp)
                .padding(horizontal = MediumPadding1)
        )

        Spacer(modifier = Modifier.height(MediumPadding1))


        SearchBar(
            modifier = Modifier.padding(horizontal = MediumPadding1),
            text = "",
            readOnly = true ,
            onValueChange = {},
            onClick = {
                    navigateToSearch()
            },
            onSearch = {}
        )

        Spacer(modifier = Modifier.height(MediumPadding1))

        Text(
           text = titles,
           modifier = Modifier
             .fillMaxWidth()
                .padding(start = MediumPadding1),
            fontSize = 12.sp,
            color = colorResource(id = R.color.placeholder)
        )

        Spacer(modifier = Modifier.height(MediumPadding1) )

        ArticleList(
            modifier = Modifier.padding(horizontal = MediumPadding1),
            article = articles,
            onClick = {
               navigateToDetails(it)
            }
        )
    }
}
```

h) Seach screen

```
package com.example.newsroom.presentation.search

import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.statusBarsPadding
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.paging.compose.collectAsLazyPagingItems
import com.example.newsroom.domain.model.Article
import com.example.newsroom.presentation.Dimension.MediumPadding1
import com.example.newsroom.presentation.common.ArticleList
import com.example.newsroom.presentation.common.SearchBar
import com.example.newsroom.presentation.navgraph.Route

@Composable
fun SearchScreen(
    state: SearchState,
    event: (SearchEvent) -> Unit,
    navigateToDetails: (Article) -> Unit
){

    Column(
        modifier = Modifier
            .padding(
                top = MediumPadding1,
                start = MediumPadding1,
                end = MediumPadding1
            )
            .statusBarsPadding()
            .fillMaxSize()
    ) {
        SearchBar(
            text = state.searchQuery,
            readOnly = false,
            onValueChange = {event(SearchEvent.UpdateSearchQuery(it))},
            onSearch = {event(SearchEvent.SearchNews)}
        )

        Spacer(modifier = Modifier.height(MediumPadding1))
        state.articles?.let {
            val article = it.collectAsLazyPagingItems()
            ArticleList(article = article, onClick = {navigateToDetails(it)})
        }

    }
}
```

Input and Output:



10:03 27° 36%

Search

Young Sicilian woman killed in broad daylight by stalker🟥Survivor challenges Israeli account of attack on Gaza ambulances🟥Police find 2023 Nashville school shooter sought notoriety, hid mental health issues

Young Sicilian woman killed in broad daylight by stalker
BBC News 2025-04-02T16:30:48Z

Survivor challenges Israeli account of attack on Gaza ambulances
BBC News 2025-04-02T16:28:55Z

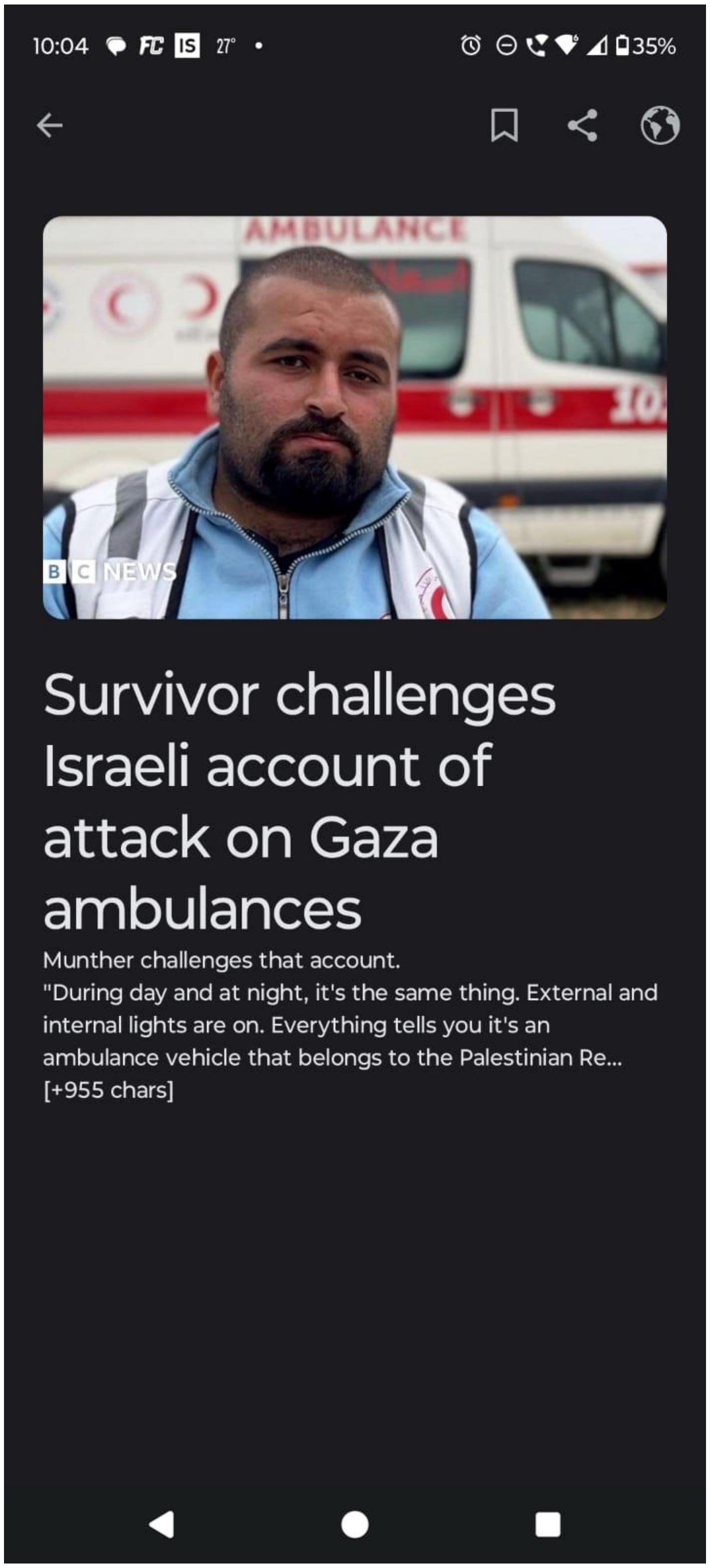
Police find 2023 Nashville school shooter sought notoriety, hid mental health issu...
ABC News 2025-04-02T16:24:34Z

Dana White's UFC and Mark Zuckerberg's Meta joining forces in partnership
ABC News 2025-04-02T16:20:25Z

Longest-serving inmate on Arkansas'

Home screen

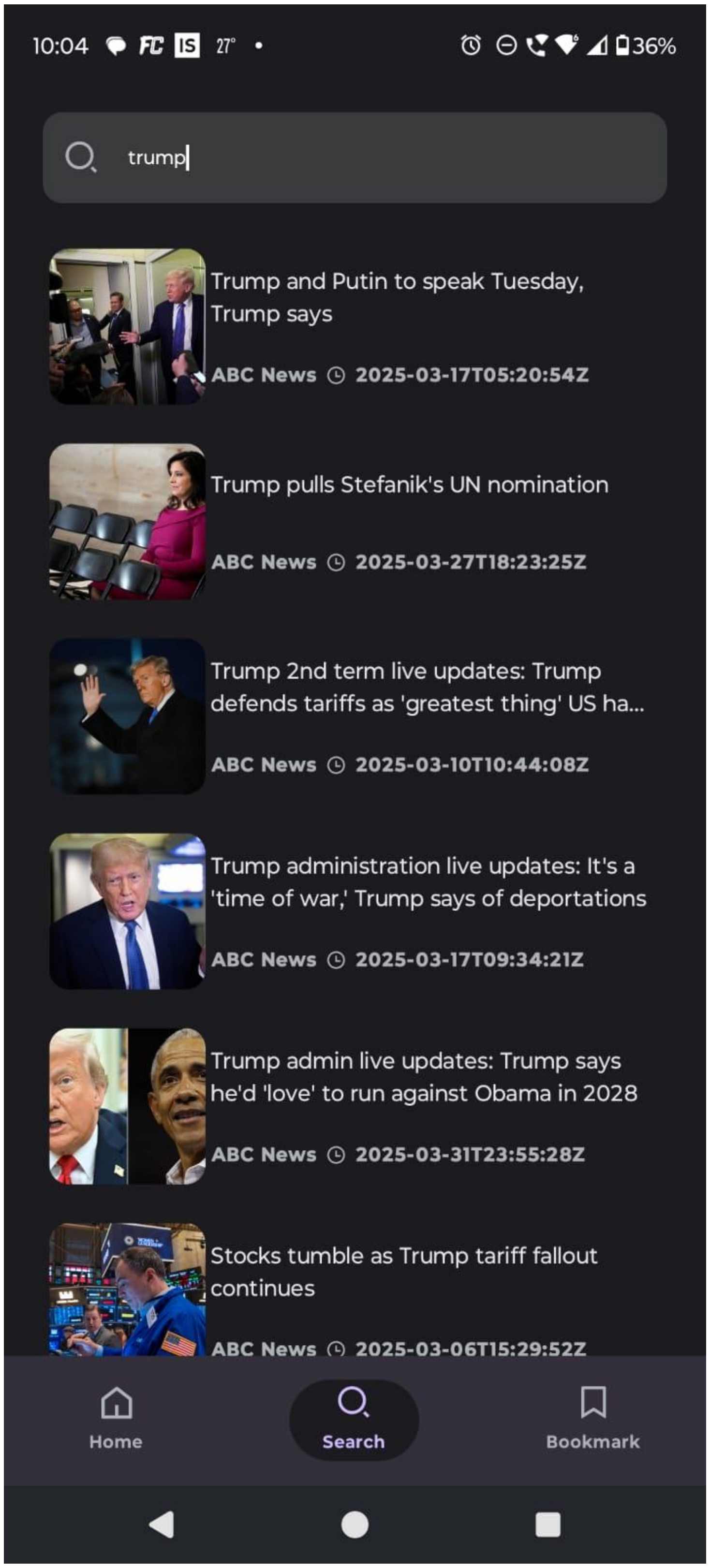
Detail Screen:


10:04 27° 35%

Survivor challenges Israeli account of attack on Gaza ambulances

Munther challenges that account.
"During day and at night, it's the same thing. External and internal lights are on. Everything tells you it's an ambulance vehicle that belongs to the Palestinian Re...
[+955 chars]

Search Screen:-



10:04 27° 36%

trump

Trump and Putin to speak Tuesday, Trump says

ABC News 2025-03-17T05:20:54Z

Trump pulls Stefanik's UN nomination

ABC News 2025-03-27T18:23:25Z

Trump 2nd term live updates: Trump defends tariffs as 'greatest thing' US ha...

ABC News 2025-03-10T10:44:08Z

Trump administration live updates: It's a 'time of war,' Trump says of deportations

ABC News 2025-03-17T09:34:21Z

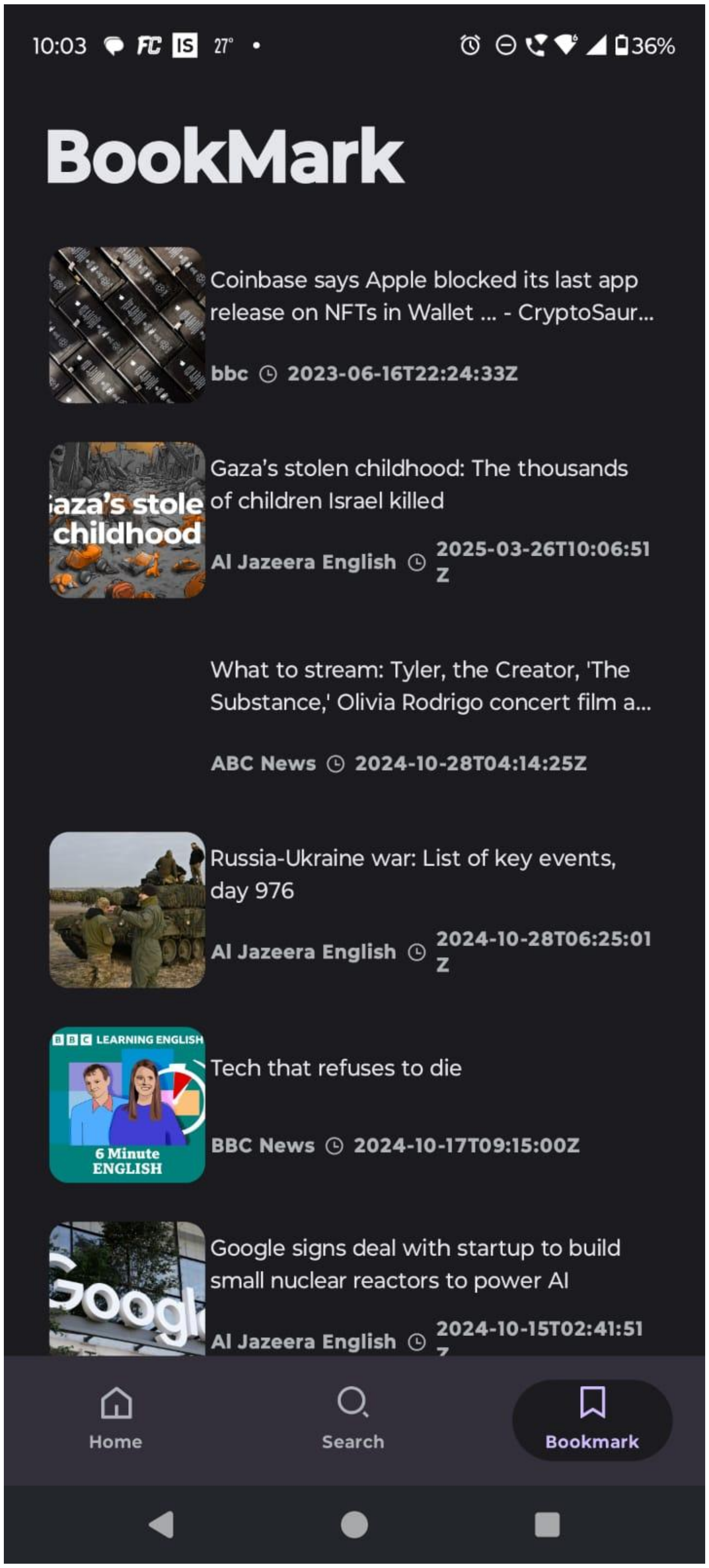
Trump admin live updates: Trump says he'd 'love' to run against Obama in 2028

ABC News 2025-03-31T23:55:28Z

Stocks tumble as Trump tariff fallout continues

ABC News 2025-03-06T15:29:52Z

Bookmark Screen



10:03 27° 36%

BookMark

Coinbase says Apple blocked its last app release on NFTs in Wallet ... - CryptoSaur...

bbc 2023-06-16T22:24:33Z

Gaza's stolen childhood: The thousands of children Israel killed

Al Jazeera English 2025-03-26T10:06:51Z

What to stream: Tyler, the Creator, 'The Substance,' Olivia Rodrigo concert film a...

ABC News 2024-10-28T04:14:25Z

Russia-Ukraine war: List of key events, day 976

Al Jazeera English 2024-10-28T06:25:01Z

Tech that refuses to die

BBC News 2024-10-17T09:15:00Z

Google signs deal with startup to build small nuclear reactors to power AI

Al Jazeera English 2024-10-15T02:41:51

Observations and learning:

Observations:

1. **Real-Time News Updates:**
   - Successfully fetched and displayed the latest news articles from external APIs.
   - Observed that API response time varies based on internet speed and server load.
   - Some APIs had rate limits, requiring efficient request handling.
2. **Category-Wise News Filtering:**
   - Users could filter news based on categories such as Sports, Technology, and Business.
   - Implementing filtering improved user experience by displaying only relevant news.
3. **Search Functionality:**
   - The search feature allowed users to find specific news articles quickly.
   - Observed that using indexing or caching improved search performance.
4. **Offline Reading:**
   - Successfully stored news articles in the local Room database for offline access.
   - Observed that large amounts of saved articles could slow down retrieval if not optimized.
5. **Push Notifications (FCM):**
   - Notifications were successfully sent and received for breaking news.
   - Observed that too many notifications led to user fatigue, requiring a balanced notification strategy.
6. **User Interface (UI) and Navigation:**
   - Implemented an intuitive UI using Jetpack Compose.
   - Observed that users preferred a minimalist design for easy navigation.

Learning Outcomes:

1. **Understanding API Integration:**
   - Learned how to fetch real-time data using Retrofit and handle API responses efficiently.
2. **Efficient State Management:**
   - Used ViewModel (MVVM architecture) to manage UI-related data effectively.
3. **Database Management:**
   - Learned how to store and retrieve news articles using Room Database.
   - Optimized queries to improve offline reading performance.
4. **Improving Performance:**
   - Implemented caching and pagination to improve app speed and efficiency.
5. **User Experience (UX) Best Practices:**
   - Understood the importance of clean UI design and smooth navigation.

    - Balanced push notifications to enhance engagement without overwhelming users.
6. **Cross-Platform Considerations:**
    - Explored cross-platform tools like Flutter for future app scalability.
7. **Deployment & Testing:**
    - Understood how to test and deploy Android applications on real devices.
    - Learned the importance of debugging and performance testing before release.

Conclusion:

The development of the **News Application** provided valuable insights into mobile application development, API integration, and user experience design. The successful implementation of real-time news updates, category-wise filtering, offline reading, and push notifications demonstrated a comprehensive approach to delivering an efficient and user-friendly news platform.

Through this project, we achieved the following key outcomes:

1. **Seamless API Integration:** We effectively fetched real-time news from external sources using **Retrofit**, ensuring up-to-date content for users.
2. **Enhanced User Experience:** By implementing **category filtering, search functionality, and a clean UI**, we improved usability and engagement.
3. **Efficient Offline Storage:** The integration of the **Room database** allowed users to save articles for offline access, improving accessibility.
4. **Optimized Performance:** Using **ViewModel, Coroutines, and caching techniques**, we enhanced app speed and resource management.
5. **Push Notification Management:** We balanced notifications to keep users informed without causing notification fatigue.