Name: Subhraneel Sil

VIT CHENNAI

20BEC1307

An Android Application for keeping up with the latest headlines

1 INTRODUCTION

1.1 Overview

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

1.2 Purpose

The purpose is to develop an android application, which will eliminate the problems faced in the current scenario. This application will provide all the information and news at one place. So, it will save time and efforts of the users by making it more efficient. Using this application will terminate the possibility of information redundancy.

2 LITERATURE SURVEY

2.1 Existing problem

There are several existing approaches you can take to build an Android application for keeping up with the latest headlines. Here are a few common approaches:

- RSS Feeds Integration: Many news websites provide RSS feeds that allow you to fetch their latest headlines and articles programmatically. You can integrate these feeds into your Android application using libraries like Rome or SimpleRSS to parse the XML data and display the headlines to the users.
- News API Integration: There are several news APIs available that provide easy access to the latest headlines from various sources. You can use popular APIs like NewsAPI, Google News API, or RSS2JSON to fetch the news data and display it in your Android application. These APIs often provide additional features like filtering by category, language, or country.

- Content Aggregator APIs: Some platforms provide aggregated news content from various sources. For example, Flipboard and Feedly have APIs that allow you to fetch curated news articles. By integrating with these APIs, you can provide a wide range of news topics and sources to your users.
- Custom Backend: If you have control over the news sources and want more flexibility, you can build a custom backend that aggregates news data from different sources. Your backend can periodically fetch the latest headlines using any of the above methods and expose an API that your Android application can consume.

2.2 Proposed solution

In our project we are using a blend of **MVVM, Dagger Hilt** and **Retrofit**.

MVVM separates the UI (View) from the business logic (ViewModel) and data (Model), making the codebase more modular and maintainable. With a clear separation of responsibilities, it becomes easier to write unit tests for the ViewModel and ensure the correctness of the business logic. MVVM enables reactive UI updates through data binding, allowing the UI to automatically reflect changes in the ViewModel without manual intervention. Dagger Hilt simplifies the management of dependencies by automating much of the setup and boilerplate code, resulting in a more efficient and maintainable codebase. By leveraging dependency injection, Dagger Hilt promotes modularity and testability, allowing for easier testing and reusability of components. Dagger Hilt offers built-in support for scoping dependencies, ensuring that objects are created and managed based on specific lifecycles, improving resource utilization and memory management. Retrofit provides a high-level, declarative interface for making network requests, handling HTTP methods, request headers, and response serialization, simplifying the integration with RESTful APIs. Retrofit optimizes network communication by leveraging features like connection pooling, request compression, and response caching, leading to improved performance and reduced bandwidth consumption. Retrofit supports asynchronous execution of network requests using callbacks, coroutines, or Kotlin Flow, enabling responsive and non-blocking network interactions in the app.

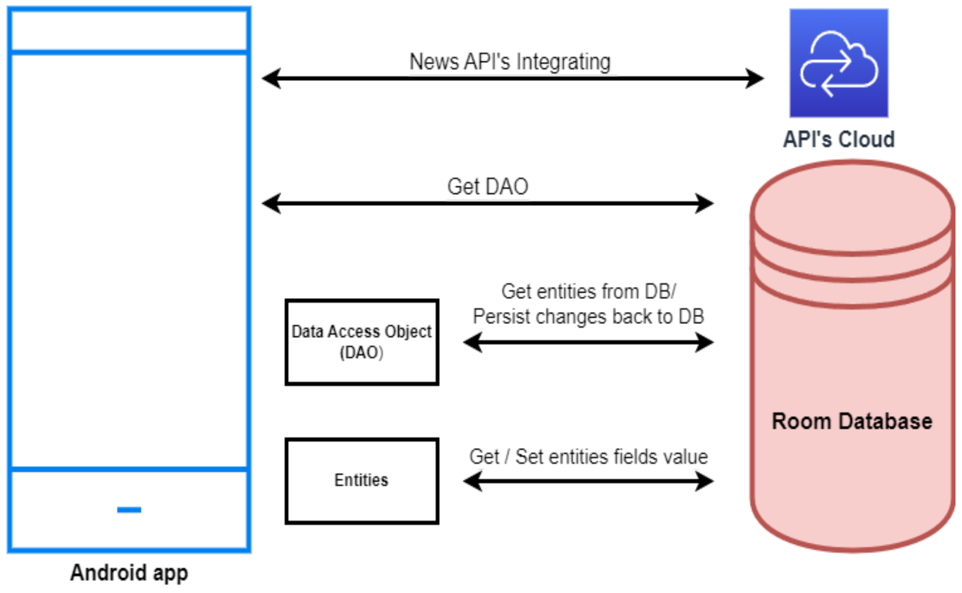
By combining MVVM Architecture, Dagger Hilt, and Retrofit code maintainability, modularity, testability, simplified dependency management, efficient network communication, and a streamlined development process became seamless. These technologies work synergistically to enhance the overall architecture, scalability, and performance of the application.

However, Implementing MVVM architecture and Dagger Hilt can introduce additional complexity to the codebase. The introduction of ViewModels and data binding requires careful management of data flows and communication between components, which may be more complex compared to simpler architectures. The additional layers and components in MVVM architecture can introduce some overhead in terms of code size and complexity. This can potentially impact app performance and increase the overall memory usage.

When dealing with complex APIs or specialized protocols, Retrofit may require additional effort to understand and implement properly. Some APIs may have specific requirements or authentication mechanisms that need to be handled carefully.

3 THEORETICAL ANALYSIS

3.1 Block diagram



3.2 Hardware / Software designing

Hardware requirements:

Processor:	A multi-core processor with a minimum clock speed of 1.2 GHz
RAM:	2 GB or higher
Storage:	At least 100 MB of free storage space
Display:	A display with a minimum resolution of 480 x 800 pixels
GPU:	An Adreno GPU or a similar GPU is recommended for optimal graphics performance

Software requirements:

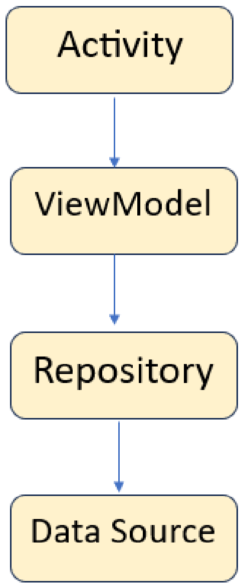
Operating system:	Android 4.1 Jelly Bean or higher
Coding Language:	Java, XML, kotlin.
Java Platform:	JRE1.8.0 amd64 and JDK 64-bit Server VM by Jetbrains.
IDE:	Android Studio Flamingo | 2022.2.1.5.2

4 EXPERIMENTAL INVESTIGATIONS

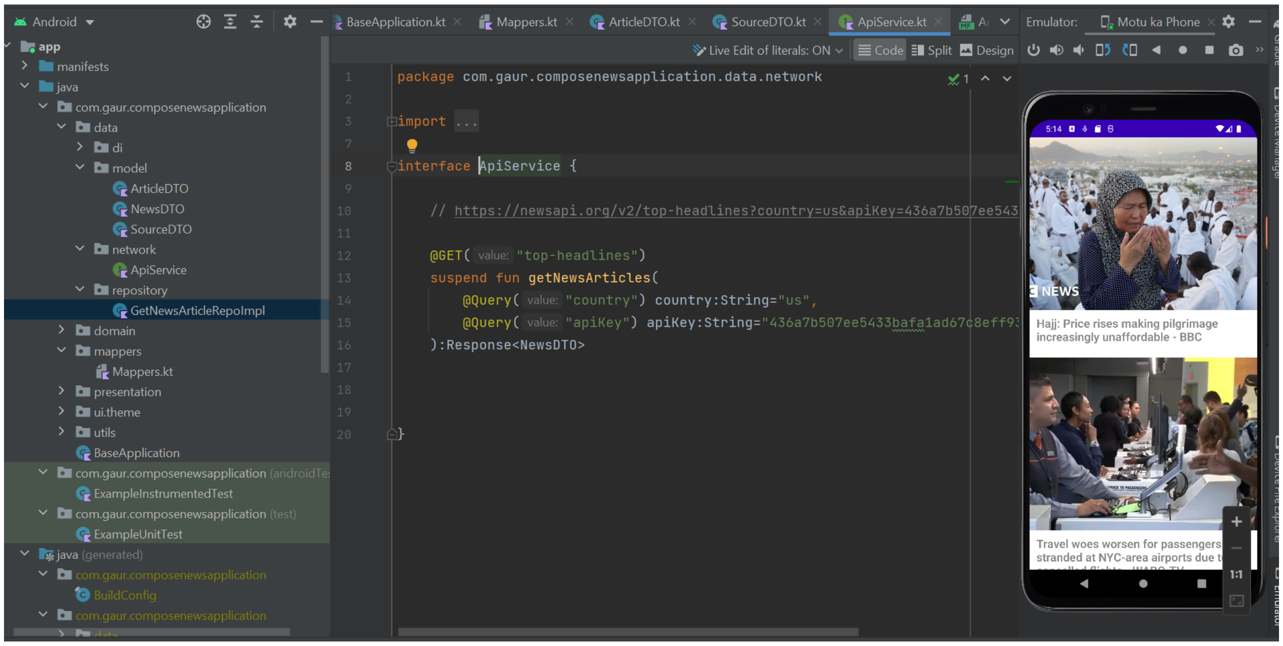
During the development of the Android application for keeping up with the latest headlines, several investigations were conducted to ensure the solution met the desired requirements and provided a seamless user experience. Here are some key areas that were explored during the investigation process:

- News Sources: A thorough investigation was carried out to identify reliable news sources and APIs that could provide up-to-date headlines. The aim was to select diverse and reputable sources to ensure the application provided a comprehensive range of news topics.
- API Integration: Different news APIs were evaluated to determine the most suitable option for retrieving headlines. Factors considered included ease of integration, data quality, response time, and any usage limitations or costs associated with the API.
- User Interface (UI) Design: Extensive research was conducted to understand user preferences and expectations regarding news consumption. The investigation involved exploring various UI design patterns, layouts, and navigation structures that would allow users to effortlessly browse and explore news articles.
- Performance and Responsiveness: Extensive testing and investigation were conducted to ensure the application performed well, even with a large number of news articles or active users.

5 FLOWCHART



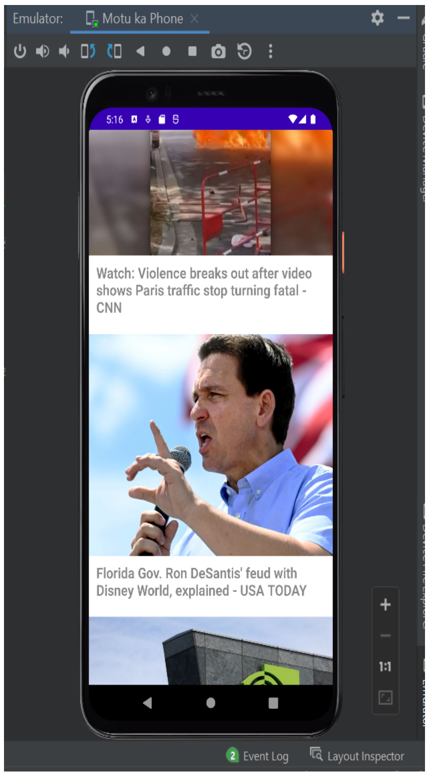
6 RESULT



```
package com.gaur.composenewsapplication.data.network

import ...

interface ApiService {

    // https://newsapi.org/v2/top-headlines?country=us&apiKey=436a7b507ee543

    @GET("top-headlines")
    suspend fun getNewsArticles(
        @Query("country") country:String="us",
        @Query("apiKey") apiKey:String="436a7b507ee5433bafa1ad67c8eff93
    ):Response<NewsDTO>


}
```

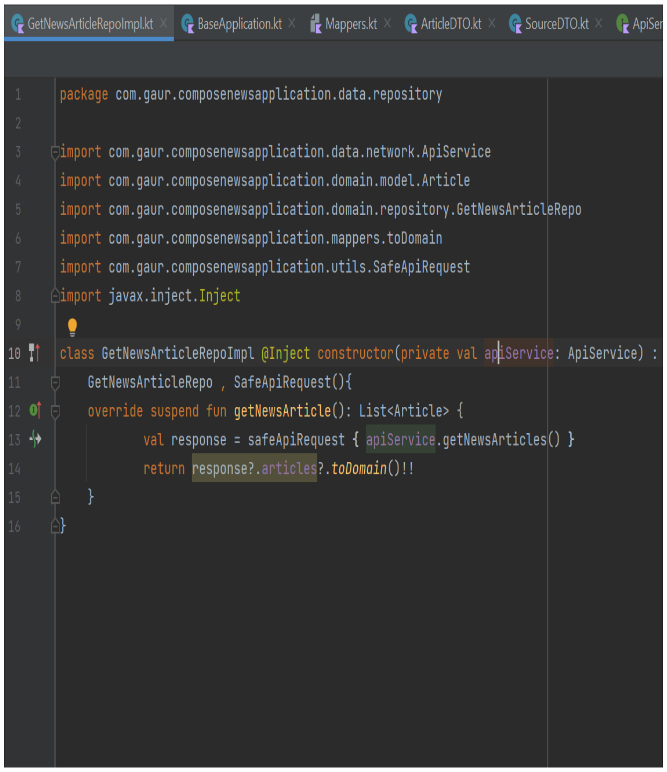
Hajj: Price rises making pilgrimage increasingly unaffordable - BBC

Travel woes worsen for passengers stranded at NYC-area airports due t



Watch: Violence breaks out after video shows Paris traffic stop turning fatal - CNN

Florida Gov. Ron DeSantis' feud with Disney World, explained - USA TODAY



```
package com.gaur.composenewsapplication.data.repository

import com.gaur.composenewsapplication.data.network.ApiService
import com.gaur.composenewsapplication.domain.model.Article
import com.gaur.composenewsapplication.domain.repository.GetNewsArticleRepo
import com.gaur.composenewsapplication.mappers.toDomain
import com.gaur.composenewsapplication.utils.SafeApiRequest
import javax.inject.Inject

class GetNewsArticleRepoImpl @Inject constructor(private val apiService: ApiService) :
    GetNewsArticleRepo , SafeApiRequest(){
    override suspend fun getNewsArticle(): List<Article> {
            val response = safeApiRequest { apiService.getNewsArticles() }
            return response?.articles?.toDomain()!!
    }
}
```

7 ADVANTAGES & DISADVANTAGES

- A news headliner app provides users with quick and convenient access to the latest news headlines from various sources.

- Users can stay updated on current events without the need to browse multiple news websites or sources individually. Many news headliner apps offer customization options, allowing users to personalize their news feed based on their interests, preferred topics, or specific news sources.
- Also, it is very much time efficient and at a few blinks of an eye we stay updated to current trends of the world.
- However, News headliner apps typically provide only a brief summary or headline, which may lack the full context or details of a news story. Depending on the app's news sources or algorithms, there is a possibility of inherent bias in the selection or presentation of news headlines. Users should be aware of this and consider diversifying their news sources to get a well-rounded perspective.

8 APPLICATIONS

News reading Android apps have become increasingly popular due to their convenience and ability to deliver up-to-date news from various sources. These apps offer a range of features and benefits to enhance the user's news consumption experience. Here are some applications of news reading Android app that we look forward to adding to our application in the very near future:

- Aggregating News from Multiple Sources: News apps gather news articles and updates from multiple sources, such as newspapers, magazines, blogs, and news websites. Users can access a wide range of news content in one place, eliminating the need to visit multiple websites or apps.
- Customized News Feed: Many news reading apps allow users to personalize their news feed based on their interests and preferences. By selecting specific topics, categories, or sources, users can receive relevant news tailored to their liking. This customization helps users stay informed about the subjects they care about the most.
- Breaking News Alerts: News apps often provide breaking news alerts, ensuring that users are promptly notified of significant events or developments. These alerts can be customized based on the user's interests or can be set to cover general breaking news across various categories.
- Social Sharing: News reading apps frequently include social sharing options, allowing users to share news articles or snippets with their friends and followers on social media platforms. This feature facilitates the dissemination of news and encourages discussions around current events.

9 CONCLUSION

In conclusion, news reading Android apps have revolutionized the way people consume news and stay informed about current events. These apps offer a plethora of features and benefits that enhance the user experience and provide a convenient platform for accessing news content from various sources. From aggregating news from multiple sources to personalized news feeds, breaking news alerts, offline reading capabilities, and social sharing options, these apps cater to the diverse needs and preferences of users.

The ability to customize news feeds based on interests and preferences ensures that users receive relevant and engaging content. Moreover, the inclusion of multimedia elements like images, videos, and audio clips adds depth and immersion to the news reading experience. Additionally, features such as bookmarking, saving articles, personalized recommendations, and access to archives enhance the usability and accessibility of news content.

News reading apps also cater to the global audience by providing multiple language support, enabling users to access news in their preferred language and stay updated on global events. The search functionality and archival access further facilitate easy navigation and retrieval of specific articles or topics.

Overall, news reading Android apps have transformed the way news is consumed, making it more convenient, personalized, and interactive. These apps serve as a centralized hub for news consumption, allowing users to access a wide range of news sources, topics, and multimedia content in one place. By leveraging the power of technology, news apps have made staying informed a seamless and enriching experience for users.

10 FUTURE SCOPE

The future scope of a news app is promising, as technological advancements and evolving user needs continue to shape the landscape of news consumption. Here are some potential areas of growth and development for news apps in the future:

- Artificial Intelligence and Personalization: News apps can further leverage artificial intelligence (AI) to enhance personalization. AI algorithms can analyze user preferences, behavior, and reading habits to deliver even more tailored and relevant news content. This can include more accurate topic recommendations, smarter news curation, and real-time content suggestions based on user interests.
- Augmented Reality (AR) and Virtual Reality (VR): News apps could integrate AR and VR technologies to create immersive news experiences. Users might be able to virtually visit locations, events, or historical moments through AR/VR, bringing news stories to life in a more engaging and interactive manner.
- Voice Assistants and Natural Language Processing: With the growing popularity of voice assistants, news apps could incorporate voice-based interactions. Users might be able to ask their virtual assistants for news briefings, customized news updates, or specific information from news articles using natural language processing techniques.
- Enhanced Multimedia Integration: News apps can further enhance the integration of multimedia elements. This may include interactive infographics, 360-degree videos, live streaming, and augmented visuals to provide users with a more immersive and visually appealing news experience.

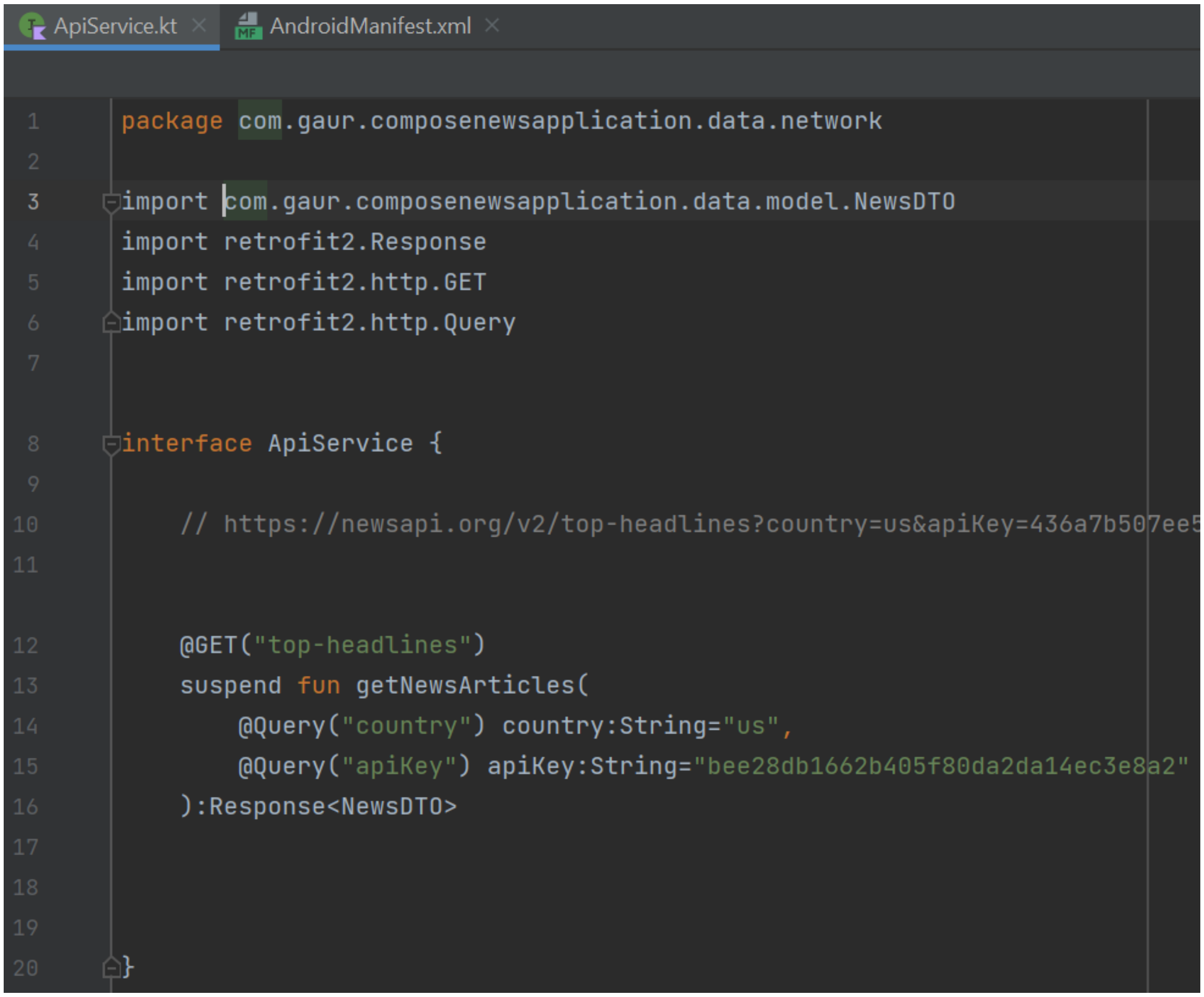
11 BIBLIOGRAPHY

- RibeiroandA.R.D.Silva,“SurveyonCross-PlatformsandLanguagesforMobile Apps,”EighthInternationalConferenceontheQualityofInformationand CommunicationsTechnology,2012.

- A.KathuriaandA.Gupta,“ChallengesinAndroidApplicationDevelopment:ACase Study”InternationalJournalofComputerScienceandMobileComputing,vol.4,no.5, pp.294–299,May2015.
- https://naanmudhalvan.smartinternz.com/saas-guided-project/1/an-android-application-for-keeping-up-with-the-latest-headlines
- https://www.youtube.com/watch?v=eIpkxJtZP2g

APPENDIX

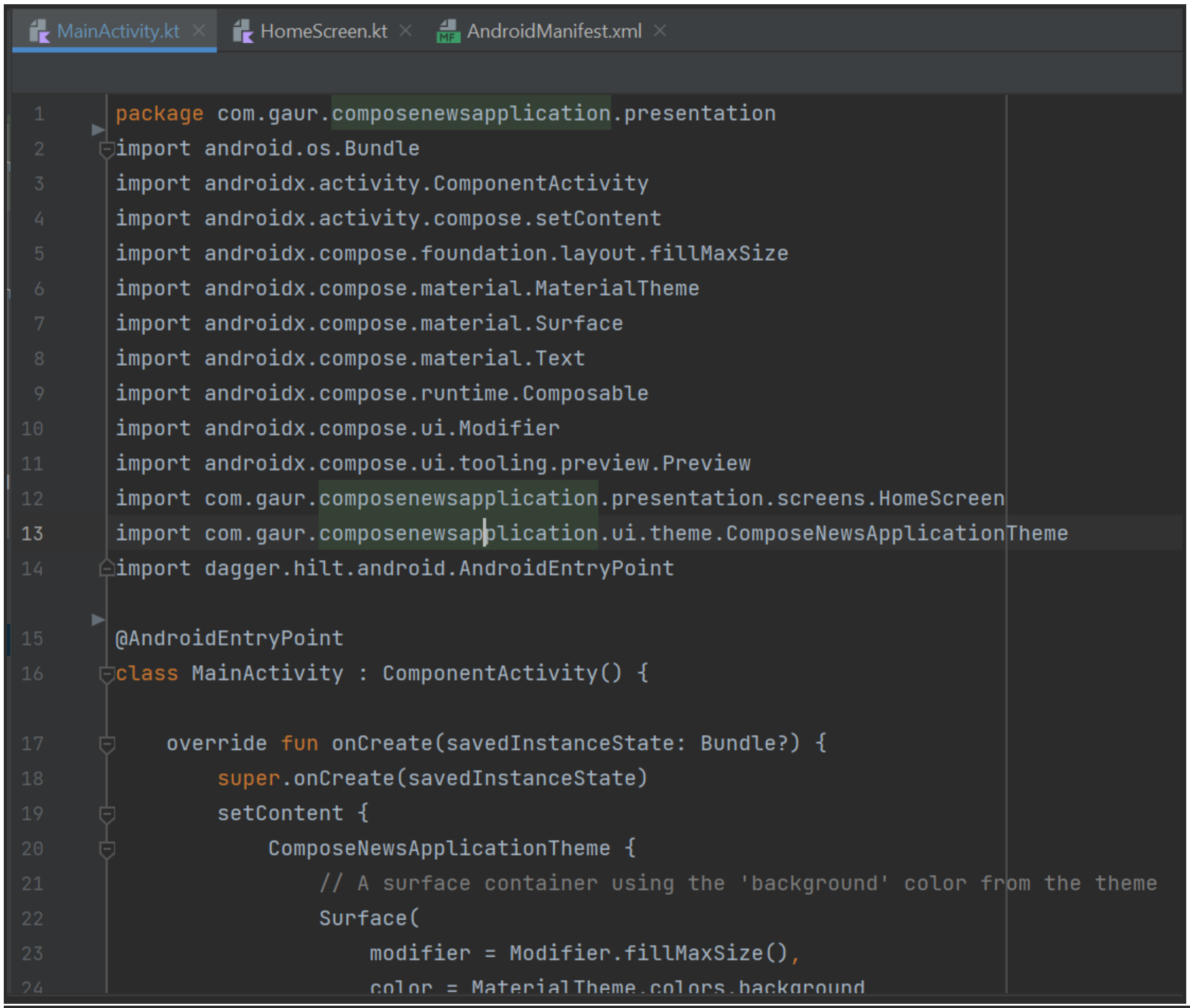
Api-



ApiService.kt AndroidManifest.xml

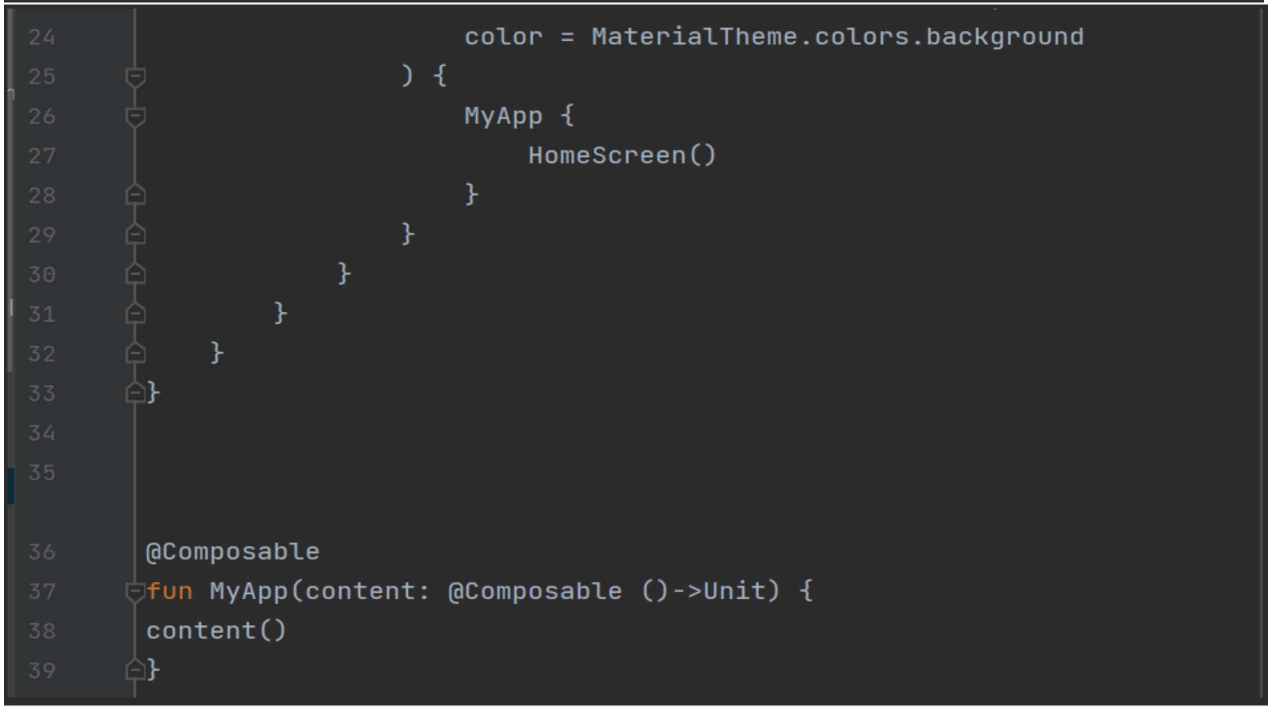
```
package com.gaur.composenewsapplication.data.network

import com.gaur.composenewsapplication.data.model.NewsDTO
import retrofit2.Response
import retrofit2.http.GET
import retrofit2.http.Query


interface ApiService {

    // https://newsapi.org/v2/top-headlines?country=us&apiKey=436a7b507ee5


    @GET("top-headlines")
    suspend fun getNewsArticles(
        @Query("country") country:String="us",
        @Query("apiKey") apiKey:String="bee28db1662b405f80da2da14ec3e8a2"
    ):Response<NewsDTO>



}
```

MainActivity-



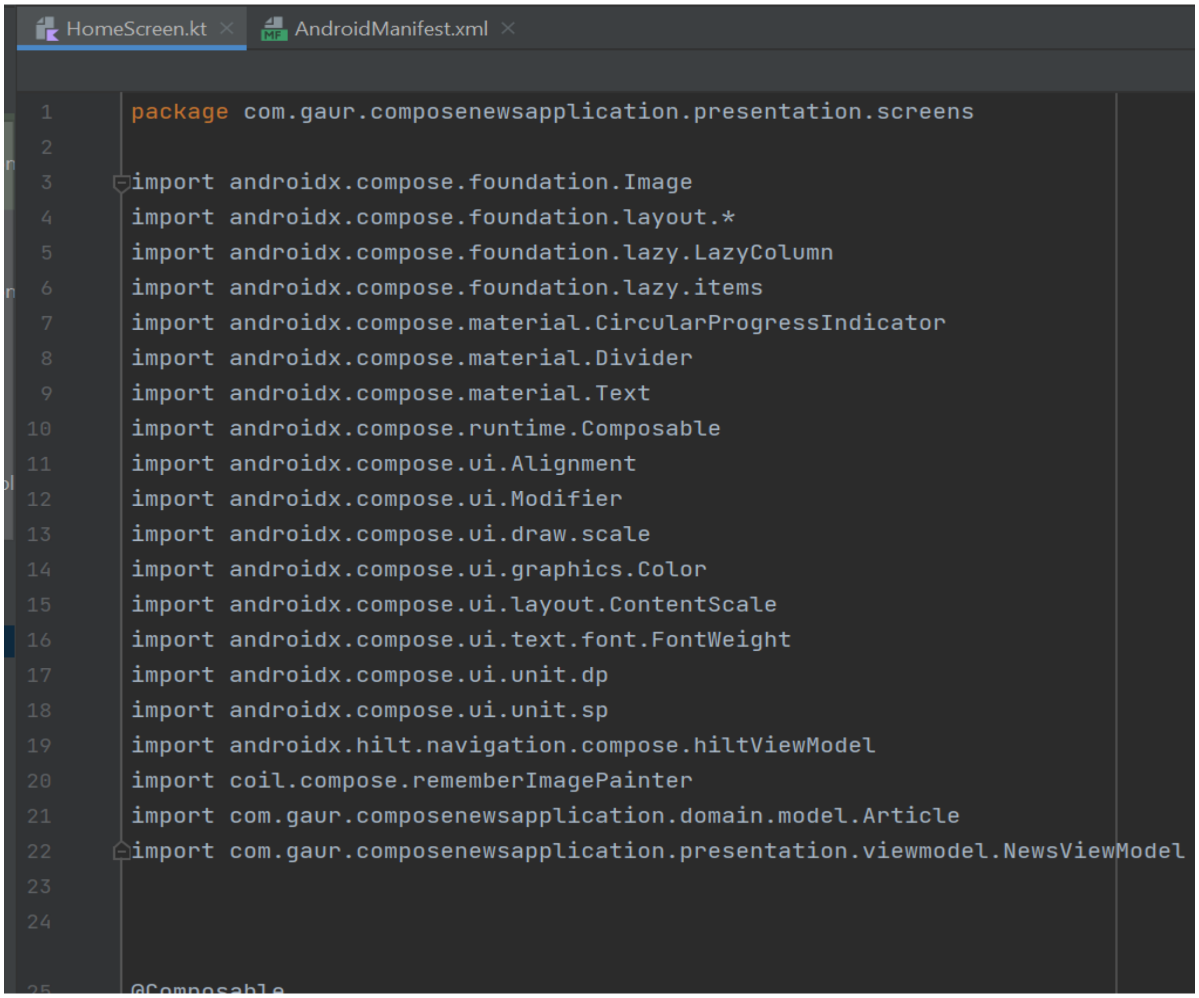
MainActivity.kt × HomeScreen.kt × AndroidManifest.xml ×

```kotlin
package com.gaur.composenewsapplication.presentation
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material.MaterialTheme
import androidx.compose.material.Surface
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.tooling.preview.Preview
import com.gaur.composenewsapplication.presentation.screens.HomeScreen
import com.gaur.composenewsapplication.ui.theme.ComposeNewsApplicationTheme
import dagger.hilt.android.AndroidEntryPoint

@AndroidEntryPoint
class MainActivity : ComponentActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            ComposeNewsApplicationTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
```

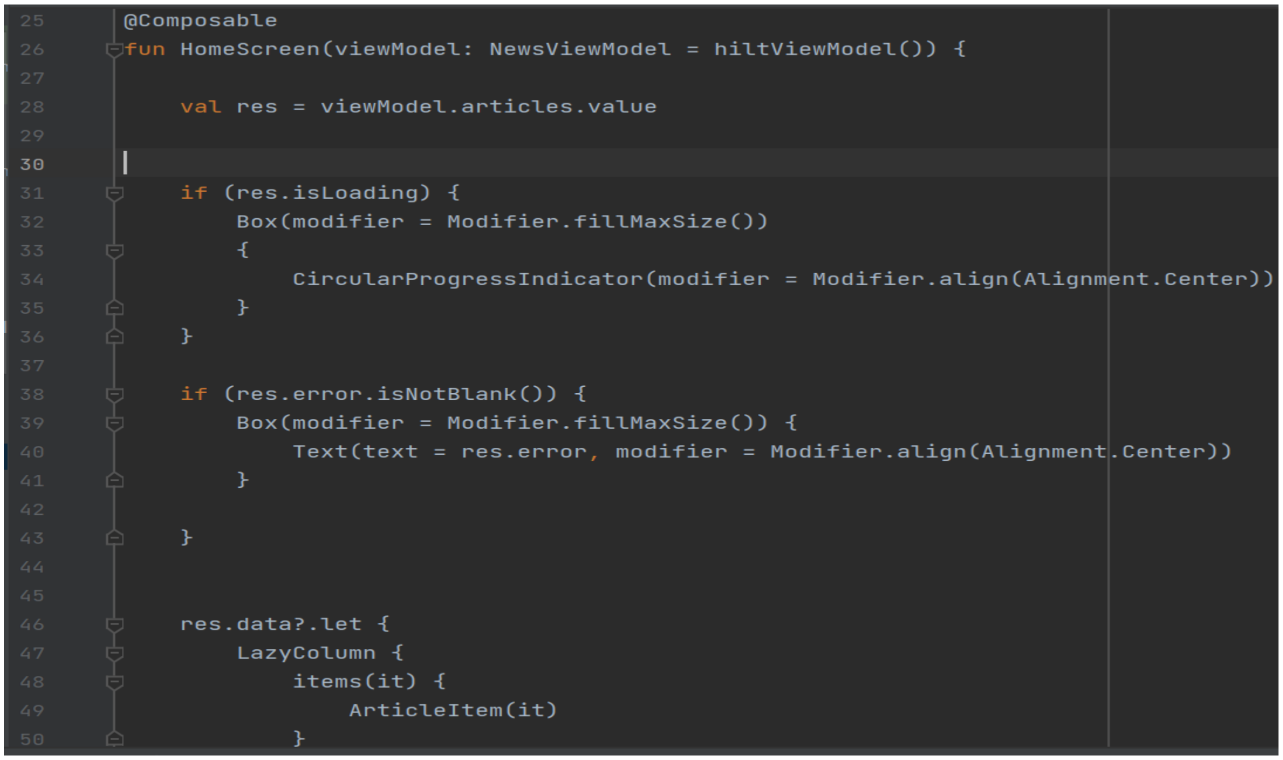


```kotlin
                    color = MaterialTheme.colors.background
                ) {
                    MyApp {
                        HomeScreen()
                    }
                }
            }
        }
    }
}



@Composable
fun MyApp(content: @Composable ()->Unit) {
content()
}
```

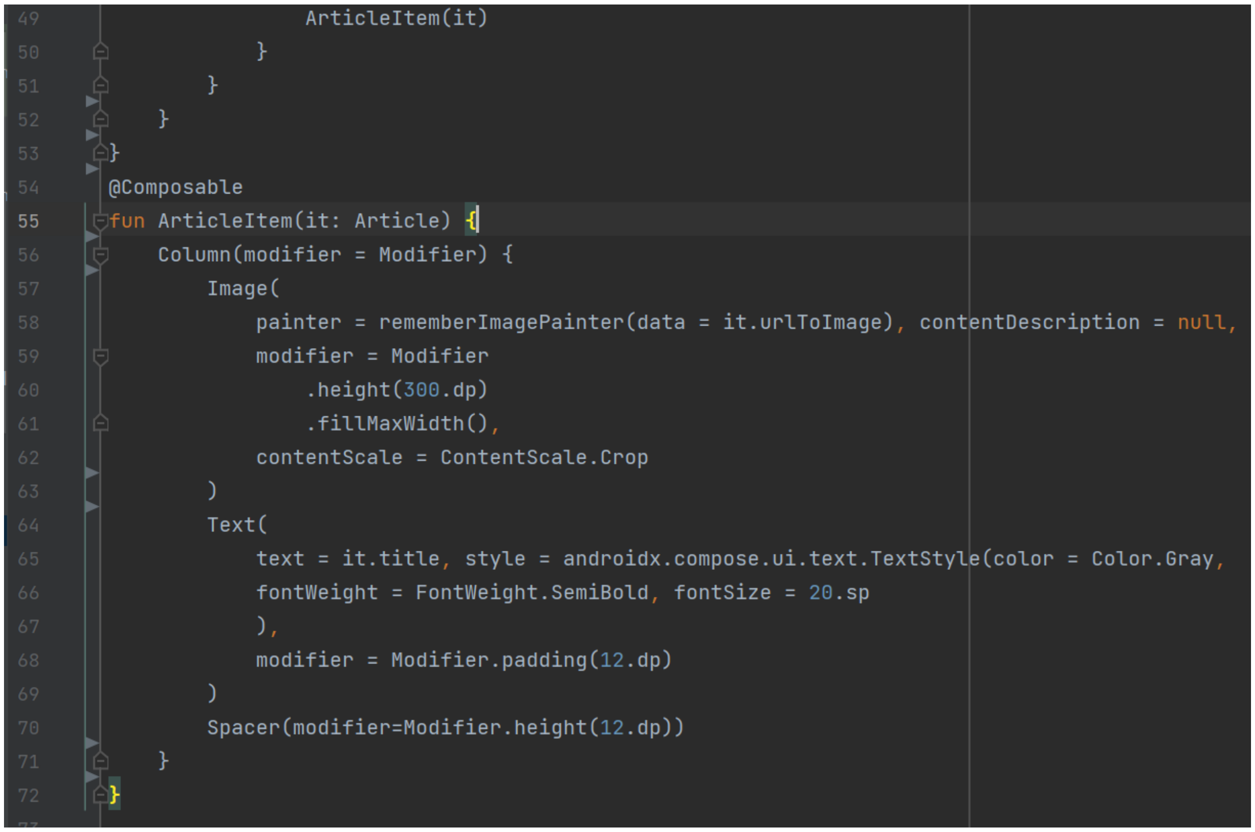
Home Screen Cod


HomeScreen.kt AndroidManifest.xml

```
package com.gaur.composenewsapplication.presentation.screens

import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material.CircularProgressIndicator
import androidx.compose.material.Divider
import androidx.compose.material.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.scale
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.hilt.navigation.compose.hiltViewModel
import coil.compose.rememberImagePainter
import com.gaur.composenewsapplication.domain.model.Article
import com.gaur.composenewsapplication.presentation.viewmodel.NewsViewModel



@Composable
```



```kotlin
@Composable
fun HomeScreen(viewModel: NewsViewModel = hiltViewModel()) {

    val res = viewModel.articles.value


    if (res.isLoading) {
        Box(modifier = Modifier.fillMaxSize())
        {
            CircularProgressIndicator(modifier = Modifier.align(Alignment.Center))
        }
    }

    if (res.error.isNotBlank()) {
        Box(modifier = Modifier.fillMaxSize()) {
            Text(text = res.error, modifier = Modifier.align(Alignment.Center))
        }

    }


    res.data?.let {
        LazyColumn {
            items(it) {
                ArticleItem(it)
            }
```



```kotlin
                ArticleItem(it)
            }
        }
    }
}
@Composable
fun ArticleItem(it: Article) {
    Column(modifier = Modifier) {
        Image(
            painter = rememberImagePainter(data = it.urlToImage), contentDescription = null,
            modifier = Modifier
                .height(300.dp)
                .fillMaxWidth(),
            contentScale = ContentScale.Crop
        )
        Text(
            text = it.title, style = androidx.compose.ui.text.TextStyle(color = Color.Gray,
            fontWeight = FontWeight.SemiBold, fontSize = 20.sp
            ),
            modifier = Modifier.padding(12.dp)
        )
        Spacer(modifier=Modifier.height(12.dp))
    }
}
```