An Android Application For Keeping Up With The Latest Headlines

Required Initial Steps

A PROJECT REPORT

Submitted by

MOHAMMED USMAN.J (812022104022)

MANIKANDAN.A. (812022104042)

BALAMURUGAN.S (812022104014)

MOHAMMED GANI.K(8120221040701)

BACHELOR OF ENGINNERING

IN

FIFTH SEMESTER

COMPUTER SCIENCE AND ENGINEERING

M.A.M. COLLEGE OF ENGINEERING AND TECHNOLOGY, TRICHY

ANNA UNIVERSITY : CHENNAI 600 025

NOVEMBER 2022

TABLE OF CONTENTS

Abstract:

This Android application is designed to keep users informed about the latest headlines and news in real-time. The app aggregates news from multiple reliable sources, presenting a curated feed tailored to the user's preferences. It features categories like politics, sports, technology, entertainment, and global events, allowing users to customize their news experience.

The app provides notifications for breaking news, offline reading capabilities, and a search function for exploring specific topics. It incorporates AI-based algorithms to recommend articles based on reading habits. A sleek, user-friendly interface ensure**s an engaging and seamless**

experience. By combining speed, relevance, and accessibility, this application serves as a comprehensive solution for staying updated with current events.

Introduction:

In today’s fast-paced world, staying informed about current events is essential. However, with the overwhelming amount of information available online, it can be challenging to find reliable and relevant news quickly. To address this need, an Android application for keeping up with the latest headlines offers a streamlined and efficient solution for users to access real-time news from various trusted sources.

This application provides a curated platform where users can explore breaking news, trending stories, and personalized content based on their interests. With features such as category-based browsing, push notifications for important updates, and offline reading options, the app ensures that users remain connected to the world, regardless of time or place.

By leveraging modern technology, including AI-driven recommendations and a user-friendly interface, this application aims to revolutionize the way people consume news, making it more accessible, personalized, and engaging for everyone.

CHAPTER -1

Required initial steps

1. **Requirement Analysis**

   - Define the target audience and their needs (e.g., personalized news, offline access).

   - Determine key features such as news categorization, notifications, offline reading, and search functionality.

   - Identify data sources (e.g., APIs like NewsAPI, RSS feeds).

2. **Conceptualization and Planning**

   - Create a project roadmap with milestones for development, testing, and deployment.

   - Design the app architecture (frontend, backend, database).

   - Decide on tools and technologies (e.g., Android Studio, Kotlin/Java).

3. **UI/UX Design**

   - Design wireframes and mockups for the app layout.

   - Focus on a clean, user-friendly interface with easy navigation.

   - Test designs for responsiveness across various devices and screen sizes.

4. **API Integration**

   - Research and select reliable news APIs or RSS feeds for aggregating headlines.

   - Plan for API usage limits and data handling (e.g., JSON parsing).

5. **Backend Development**

   - Set up a backend server for handling user data, preferences, and notifications.

   - Integrate a database for storing user preferences, bookmarked articles, etc.

6. **Frontend Development**

   - Develop the app's user interface using Android development tools.

   - Implement features like category browsing, search functionality, and push notifications.

7. **Personalization Features**

   - Integrate AI or ML algorithms to recommend articles based on user behavior.

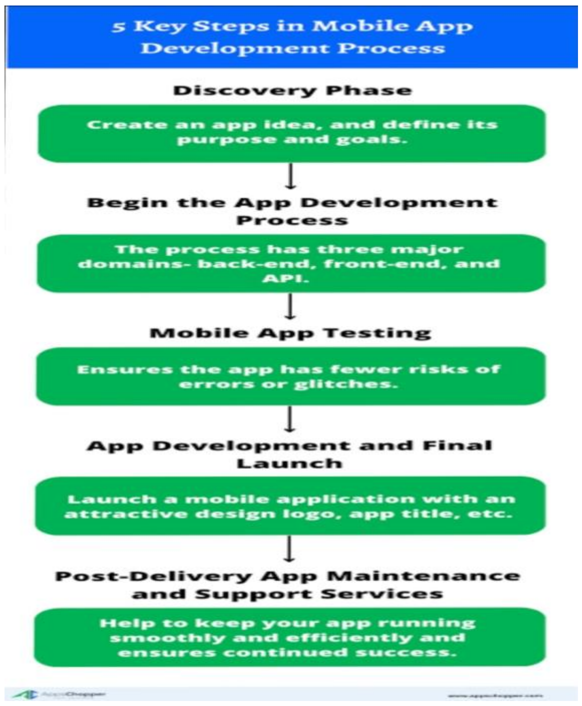
   - Allow users to customize their news feed by selecting preferred categories or topics.

8. **Testing**

- Perform unit testing, integration testing, and UI testing.

- Ensure compatibility with different Android versions and devices.

- Test for performance, security, and scalability.


5 Key Steps in Mobile App Development Process
Discovery Phase
Create an app idea, and define its purpose and goals.
Begin the App Development Process
The process has three major domains- back-end, front-end, and API.
Mobile App Testing
Ensures the app has fewer risks of errors or glitches.
App Development and Final Launch
Launch a mobile application with an attractive design logo, app title, etc.
Post-Delivery App Maintenance and Support Services
Help to keep your app running smoothly and efficiently and ensures continued success.

9. **Deployment**

- Publish the app on the Google Play Store.

- Ensure compliance with Google Play policies and guidelines.

- Prepare marketing materials, including screenshots and app descriptions.

10. **Maintenance and Updates**

- Monitor app performance and user feedback.

- Provide regular updates to fix bugs, improve features, and enhance security.

- Expand content sources and add new features based on user needs.

CHAPTER -2

This structured approach ensures a comprehensive and efficient development process, delivering a high-quality news application.

Creating a New Project for an Android Application for Keeping Up With the Latest Headlines

To begin developing the application, follow these steps:

1. Setting Up the Development Environment

- **Install Android Studio**: Download and install Android Studio, the official IDE for Android development.

- **Set Up the SDK**: Ensure the Android SDK and required libraries are installed within Android Studio.

- **Configure the Emulator**: Set up an Android Virtual Device (AVD) to test the app during development.

2. Starting a New Project

1. **Open Android Studio**: Launch the IDE and click on **"New Project"**.

2. **Select a Template**: Choose a starting template based on the app's design (e.g., "Empty Activity" or "Basic Activity").

3. **Enter Project Details**:

   - **Application Name**: Enter a descriptive name (e.g., *Headline Hub*).

   - **Package Name**: Specify a unique package identifier (e.g., `com.headlinehub.app`).

   - **Save Location**: Choose a directory to save the project.

   - **Language**: Select *Kotlin* (preferred) or *Java*.

- **Minimum SDK**: Set the minimum Android version (e.g., API level 21 for wide compatibility).

4. **Finish Setup**: Click **Finish** to create the project.

3. Structuring the Project

- **Organize Files and Folders**:

  - Use `res/layout` for UI XML files.

  - Use `res/drawable` for images and graphic resources.

  - Use `res/values` for styles, strings, and themes.

  - Use `java` or `kotlin` directories for app logic.

- **Configure Gradle**:

  - Open `build.gradle` files and add dependencies for libraries such as Retrofit (for API calls), Glide (for image loading), and Firebase (if notifications are needed).

4. Implementing Key Features

- **Create Main Activities/Fragments**: Design primary screens like the home feed, category list, and article details.

- **Set Up Navigation**: Use the Navigation Component for smooth navigation between screens.

- **Integrate News APIs**: Use Retrofit or OkHttp to fetch and display data from news APIs (e.g., NewsAPI).

5. Testing and Debugging

- **Run on Emulator**: Test the app on an AVD for functionality and layout.

- **Connect a Physical Device**: Debug on real Android devices to ensure compatibility.

6. Version Control

- **Set Up Git**: Initialize a Git repository for the project.

- **Push to Repository**: Host the project on platforms like GitHub for collaboration and backup.

By following these steps, you can establish a solid foundation for your Android application and proceed with feature development efficiently.

### **Adding Required Dependencies for an Android Application to Keep Up With the Latest Headlines**

CHAPTER -3

Dependencies are libraries and tools that simplify the development process and enhance functionality. Here's how to add the required dependencies for your news application:

1. Retrofit (API Integration)

Retrofit is a type-safe HTTP client for Android, widely used for making API requests.

```gradle

Implementation 'com.squareup.retrofit2:retrofit:2.9.0'

Implementation 'com.squareup.retrofit2:converter-gson:2.9.0'

**2. Glide (Image Loading)**

Glide is an efficient library for loading and caching images from the internet.

Implementation 'com.github.bumptech.glide:glide:4.15.1'
```

annotationProcessor ‘com.github.bumptech.glide:compiler:4.15.1’

3. RecyclerView (UI Component)

RecyclerView is used for displaying a list or grid of news articles.

gradle

Implementation ‘androidx.recyclerview:recyclerview:1.3.1’

4. Navigation Component (Screen Navigation)

The Navigation Component simplifies fragment transactions and navigation logic.

gradle

Implementation ‘androidx.navigation:navigation-fragment-ktx:2.7.3’

Implementation ‘androidx.navigation:navigation-ui-ktx:2.7.3’

5. Material Design (UI Design)

Material Design components provide modern UI elements.

~~~gradle
Implementation ‘com.google.android.material:material:1.11.0’

**6. Room Database (Local Storage)**

Room is used for caching articles locally for offline reading.

```gradle
Implementation ‘androidx.room:room-runtime:2.6.1’

annotationProcessor ‘androidx.room:room-compiler:2.6.1’

kapt‘androidx.room:room-compiler:2.6.1’  // If using Kotlin
~~~

7. Lifecycle Components (ViewModel and LiveData)

Lifecycle-aware components manage UI-related data in a lifecycle-conscious way.

Implementation ‘androidx.lifecycle:lifecycle-viewmodel-ktx:2.6.1’

Implementation ‘androidx.lifecycle:lifecycle-livedata-ktx:2.6.1’

8. Firebase Cloud Messaging (Push Notifications)

For sending push notifications about breaking news.

Implementation ‘com.google.firebase:firebase-messaging:23.3.0’

9. Paging Library (For Infinite Scrolling)

The Paging library supports loading data gradually, ideal for news feeds.

Implementation ‘androidx.paging:paging-runtime-ktx:3.2.1’

10. Gson (JSON Parsing)

Gson is used for parsing JSON responses from APIs.

Implementation ‘com.google.code.gson:gson:2.10.1’

11. Coroutines (For Asynchronous Programming)

Coroutines make asynchronous code cleaner and easier to manage.

****Steps to Add Dependencies****

1. Open the `build.gradle` file of the `app` module.

2. Add the above dependencies under the `dependencies` section.

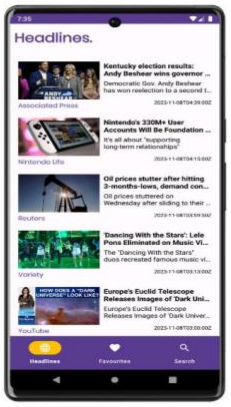
3. Click **Sync Now** in Android Studio to download and integrate the libraries.

Testing Dependencies

- Verify the integration of each library by implementing small test cases (e.g., fetch sample data using Retrofit or display an image with Glide).

- Regularly check for updates to ensure compatibility with the latest Android versions.

Adding Permissions for an Android Application to Keep Up With the Latest Headlines



CHAPTER -4

To ensure the app functions properly, certain permissions must be added to the Android Manifest file. Permissions enable the app to access resources or perform specific tasks on the user's device.

****Common Permissions Required****

1. **Internet Access**

   - To fetch news articles from APIs or RSS feeds.

   \`\`\`xml

   <uses-permission android:name="android.permission.INTERNET" />

2. **Access Network State**

   - To check the device's network connectivity status.

   \`\`\`xml

   <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

3. **Write to External Storage (Optional)**

   - If the app supports downloading articles or images for offline use.

   \`\`\`xml

   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

   <uses-permission android:name=”android.permission.READ_EXTERNAL_STORAGE” />

4. **Push Notifications (Optional)**

   - Required for Firebase Cloud Messaging to send breaking news alerts.

   ```xml

   <uses-permission android:name=”com.google.android.c2dm.permission.RECEIVE” />

   ```

Figure 1 illustrates the workflow and set of decisions associated with this process:

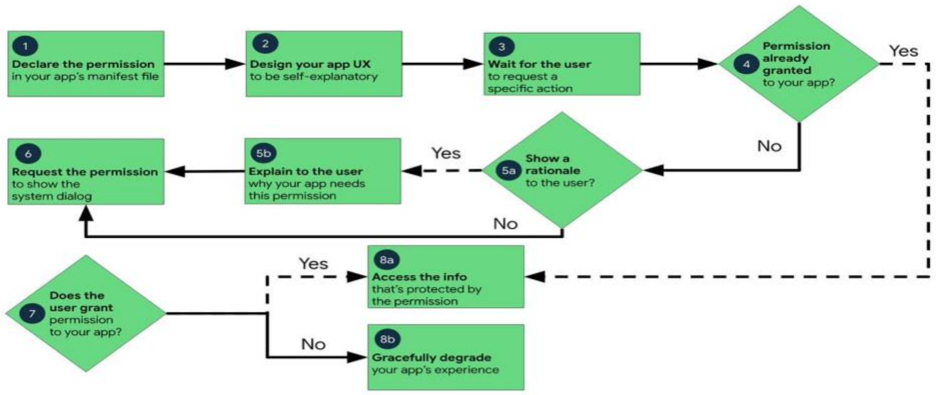

1 Declare the permission in your app's manifest file
2 Design your app UX to be self-explanatory
3 Wait for the user to request a specific action
4 Permission already granted to your app?
Yes
No
5a Show a rationale to the user?
Yes
5b Explain to the user why your app needs this permission
6 Request the permission to show the system dialog
No
7 Does the user grant permission to your app?
Yes
8a Access the info that's protected by the permission
No
8b Gracefully degrade your app's experience

Figure 1. Diagram that shows the workflow for declaring and requesting runtime permissions on Android.

Adding Permissions to `AndroidManifest.xml`

1. Open the `AndroidManifest.xml` file located in the `app/src/main` directory.

2. Add the required permissions above the `<application>` tag, like this:

   ```xml

   <manifest xmlns:android=http://schemas.android.com/apk/res/android

      Package=”com.example.headlineapp”>

      <!—Permissions →

      <uses-permission android:name=”android.permission.INTERNET” />

      <uses-permission android:name=”android.permission.ACCESS_NETWORK_STATE” />

      <uses-permission android:name=”android.permission.WRITE_EXTERNAL_STORAGE” />
   ```

```
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />

    <application
      Android:allowBackup="true"
      Android:label="Headline Hub"
      Android:icon="@mipmap/ic_launcher">
      …
    </application>
  </manifest>
```

Runtime Permissions

For permissions considered **dangerous** (e.g., accessing storage), runtime permission requests must be implemented for Android 6.0 (API level 23) and higher.

Example Code for Requesting Permissions:

```kotlin
If (ContextCompat.checkSelfPermission(this, Manifest.permission.WRITE_EXTERNAL_STORAGE)
   != PackageManager.PERMISSION_GRANTED) {
   ActivityCompat.requestPermissions(this,
     arrayOf(Manifest.permission.WRITE_EXTERNAL_STORAGE),
       REQUEST_CODE_STORAGE)
}
```

Handle Permission Result:

```kotlin

Override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<out String>, grantResults: IntArray) {
   If (requestCode == REQUEST_CODE_STORAGE) {
      If (grantResults.isNotEmpty() && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
         // Permission granted
      } else {
         // Permission denied
      }
   }
}
```

****Testing Permissions****

1. **Internet and Network State**: Test by fetching data from APIs and checking connectivity.

2. **Storage**: Verify downloading and accessing files.

3. **Push Notifications**: Ensure notification delivery via Firebase Cloud Messaging.

Adding the correct permissions ensures the app can access required resources while maintaining compliance with user privacy standards.

Creating Database Classes and API Service for a News Headlines Application

CHAPTER -5

To develop an Android application for fetching and displaying the latest headlines, you need to implement a local database for offline

support and create classes to integrate an API service for fetching news.

1. Creating Database Classes

Using Room for Local Database

Room is an abstraction layer over SQLite that makes it easier to work with databases.

Add Room Dependencies

Ensure the following dependencies are in your `build.gradle` file:

```gradle
Implementation 'androidx.room:room-runtime:2.6.1'
Kapt 'androidx.room:room-compiler:2.6.1'
```

Define the Entity Class

An entity represents a table in the database. For example, a `NewsArticle` entity:

```kotlin
Import androidx.room.Entity
Import androidx.room.PrimaryKey


@Entity(tableName = "news_articles")
Data class NewsArticle(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    Val title: String,
    Val description: String,
    Val url: String,
    Val imageUrl: String,
```

    Val publishedAt: String

)

Create the DAO (Data Access Object)

The DAO defines the database operations:

```kotlin

Import androidx.room.Dao

Import androidx.room.Insert

Import androidx.room.OnConflictStrategy

Import androidx.room.Query


@Dao

Interface NewsArticleDao {

    @Insert(onConflict = OnConflictStrategy.REPLACE)

    Suspend fun insertArticles(articles: List<NewsArticle>)


    @Query("SELECT * FROM news_articles")

    Suspend fun getAllArticles(): List<NewsArticle>


    @Query("DELETE FROM news_articles")

    Suspend fun clearArticles()

}

```

Set Up the Database

Define the database class that integrates the DAO:

```kotlin
Import androidx.room.Database
Import androidx.room.RoomDatabase


@Database(entities = [NewsArticle::class], version = 1)
Abstract class NewsDatabase : RoomDatabase() {
   Abstract fun newsArticleDao(): NewsArticleDao
}
```

Initialize the Database

In your `Application` class or a singleton:

```kotlin
Import android.content.Context
Import androidx.room.Room


Object DatabaseProvider {
   Private var INSTANCE: NewsDatabase? = null


   Fun getDatabase(context: Context): NewsDatabase {
      Return INSTANCE ?: synchronized(this) {
         Val instance = Room.databaseBuilder(
            Context.applicationContext,
            NewsDatabase::class.java,
                “news_database”
            ).build()
            INSTANCE = instance
            Instance
        }
    }
}
```
2. Creating API Service and Required Classes

API Integration Using Retrofit

Add Retrofit Dependencies

Ensure these dependencies are in your `build.gradle` file:

```gradle
Implementation ‘com.squareup.retrofit2:retrofit:2.9.0’
Implementation ‘com.squareup.retrofit2:converter-gson:2.9.0’
```
Define the API Interface

Create an interface to define API endpoints:

```kotlin
Import retrofit2.http.GET
Import retrofit2.http.Query
Import retrofit2.Response
Interface NewsApiService {
    @GET(“v2/top-headlines”)
```

    Suspend fun getTopHeadlines(
        @Query("country") country: String,
        @Query("apiKey") apiKey: String
    ): Response<NewsResponse>
}

Create the Data Model for API Response

Use data classes to model the API response. For example:

```kotlin
Data class NewsResponse(
    Val status: String,
    Val totalResults: Int,
    Val articles: List<Article>
)

Data class Article(
    Val title: String,
    Val description: String,
    Val url: String,
    Val urlToImage: String?,
    Val publishedAt: String
)
```

Set Up the Retrofit Instance

Create a singleton to initialize Retrofit:

```kotlin
Import retrofit2.Retrofit
Import retrofit2.converter.gson.GsonConverterFactory
Object RetrofitInstance {
  Private const val BASE_URL = https://newsapi.org/

  Val api: NewsApiService by lazy {
    Retrofit.Builder()
      .baseUrl(BASE_URL)
      .addConverterFactory(GsonConverterFactory.create())
      .build()
      .create(NewsApiService::class.java)
  }
}
```

3. Integrating API with Database

Use a **Repository** to connect the API service with the local database:

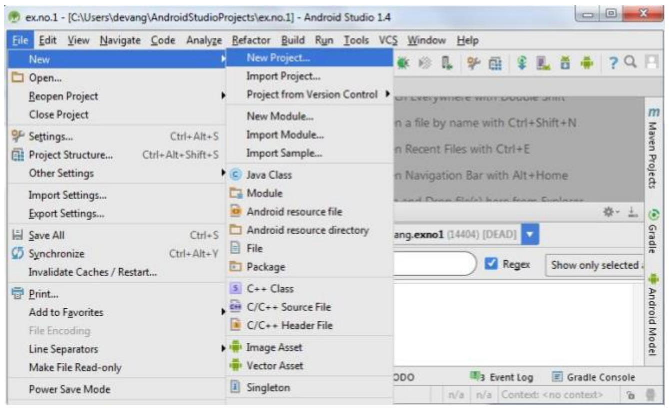
```kotlin
Class NewsRepository(private val database: NewsDatabase, private val apiService: NewsApiService) {
  Suspend fun fetchAndStoreHeadlines(country: String, apiKey: String) {
    Val response = apiService.getTopHeadlines(country, apiKey)
    If (response.isSuccessful) {
            Response.body()?.articles?.let { articles ->
                Val newsArticles = articles.map {
                    NewsArticle(
                        Title = it.title,
                        Description = it.description,
                        url = it.url,
                        imageUrl = it.urlToImage.orEmpty(),
                        publishedAt = it.publishedAt
                    )
                }
                Database.newsArticleDao().clearArticles()
                Database.newsArticleDao().insertArticles(newsArticles)
            }
        }
    }

    Suspend fun getOfflineHeadlines(): List<NewsArticle> {
        Return database.newsArticleDao().getAllArticles()
    }
}
```
4. ViewModel for Managing Data

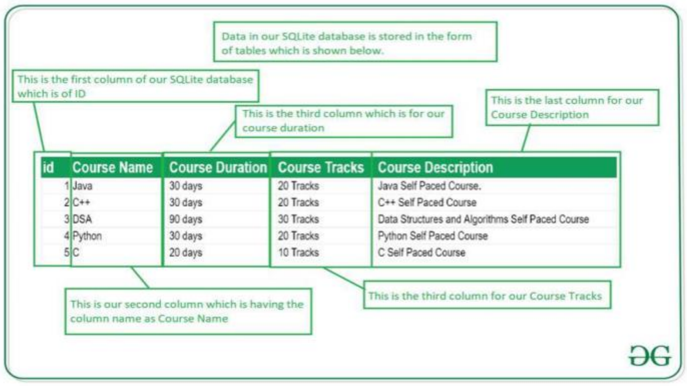
Use ViewModel to manage UI-related data:

```kotlin
```

```
Import androidx.lifecycle.ViewModel

Import androidx.lifecycle.viewModelScope

Import kotlinx.coroutines.launch

Class NewsViewModel(private val repository: NewsRepository) : ViewModel() {

   Val articles = mutableListOf<NewsArticle>()

   Fun loadHeadlines(country: String, apiKey: String) {

      viewModelScope.launch  repository.fetchAndStoreHeadlines(country, apiKey)

         articles.clear()

         articles.addAll(repository.getOfflineHeadlines())

      }

   }

}
```

By combining these components, you create a robust structure for fetching, storing, and displaying the latest headlines in your Android application.




Data in our SQLite database is stored in the form of tables which is shown below.
This is the first column of our SQLite database which is of ID
This is the third column which is for our course duration
This is the last column for our Course Description
This is our second column which is having the column name as Course Name
This is the third column for our Course Tracks

id	Course Name	Course Duration	Course Tracks	Course Description
1	Java	30 days	20 Tracks	Java Self Paced Course.
2	C++	30 days	20 Tracks	C++ Self Paced Course
3	DSA	90 days	30 Tracks	Data Structures and Algorithms Self Paced Course
4	Python	30 days	20 Tracks	Python Self Paced Course
5	C	20 days	10 Tracks	C Self Paced Course