New App

I. Introduction.

Purpose and Problem: This news app provides users with easy access to the latest news articles. It aims to solve the problem of information overload by offering a curated and user-friendly experience for consuming news.

Target Audience: This app targets individuals who want to stay informed about current events in their location but may not have the time or desire to browse through multiple news sources.

II. Project Overview

Key Features:

- Onboarding screens: Guide new users through the app's features and functionalities.
- News article display: Presents news articles with title, description, and potentially other details like source or image.
- Location-based personalization : Ability to personalize news content based on the user's location.

Technologies:

- Programming Language: Kotlin
- Framework: Jetpack Compose (for UI)
- Libraries: Retrofit (for API calls), Room (for local data storage)

- Design Pattern: Model-View-ViewModel

III. System Design

Model-View-ViewModel Architecture:

- **Model:** Represents the data layer, including data structures like `Article` and `Page` classes. It may also interact with the news API and local database (Room).
- **View:** Represents the UI components and layouts defined in composable functions (`OnStartPage`, buttons, etc.) using Jetpack Compose.
- **ViewModel:** Acts as the intermediary between Model and View, handling data preparation, state management, and communication between them.

Component Interaction:

1. User interacts with UI elements (buttons, swiping) in the `onstartScreen` composable.
2. Then the app requests location permission.
3. The composable triggers actions based on user interaction and current state (e.g., navigate between onboarding screens, handle button clicks).
4. The composable might interact with the ViewModel to update data or fetch information from the Model layer (data classes, API calls, database access).
5. The ViewModel updates the data and communicates changes back to the composable, triggering UI updates.

IV. Implementation Details

1. MainActivity:

- This is the entry point of the application.
- It sets up the splash screen and the app's theme.
- It delegates the navigation to the `HomeScreenActivity` using an `Intent`.

2. HomeScreenActivity:

- This activity displays the main news feed and allows users to browse different news categories.
- It fetches news data from an API using Retrofit networking library.
- It uses location services to personalize news based on user location (commented out in the provided code).
- It displays news articles in a RecyclerView using a custom `NewsAdapter`.
- It handles clicks on news items, opening them in a WebView activity (`NewsUI`).
- It provides buttons to navigate to different sections of the app:
    - Category buttons (General, Business, etc.) fetch news based on the selected category.
    - Saved news button opens the `SavedNews` activity.

3. NewsUI:

- This activity displays the full content of a news article loaded in a WebView.
- It provides buttons to:
    - Save the article to the device's storage using a Room database (implementation not shown in the provided code).
    - Share the news article using an Intent.

4. SavedNews:

- This activity displays a list of saved news articles retrieved from the Room database.
- It uses a `SavedNewsAdapter` to populate the RecyclerView.
- Clicking on a saved news article opens it in the `NewsUI` activity.
- It provides a back button that navigates to the `HomeScreenActivity`.

Overall, this code demonstrates the core functionalities of a news app, including fetching news data from an API, displaying it in a user-friendly interface, and allowing users to save and share articles.

5. NewsAdapter:

- This class is a `RecyclerView.Adapter` responsible for displaying news articles in a list format.
- It takes an interface `NewsItemClickListener` as a constructor argument, which allows the adapter to communicate click events on news items to the activity (`HomeScreenActivity`).
- It maintains a list of `Article` objects representing the news articles to be displayed.
- The `onCreateViewHolder` method inflates the layout for each news item using `ItemNewsArticleBinding`.
- The `onBindViewHolder` method binds data from an `Article` object to the corresponding ViewHolder, setting the title, description, and potentially loading an image (not implemented in the provided code).
- The `getItemCount` method returns the number of articles in the list.
- The `setData` method allows updating the list of articles and notifying the adapter of the change.

6. NewsAdapter.NewsViewHolder:

- This inner class represents a ViewHolder for each news item in the RecyclerView.
- It holds references to the UI elements (title, description, image view - not used here) using `ItemNewsArticleBinding`.
- It implements `View.OnClickListener` to handle clicks on the news item view.
- The `onClick` method retrieves the clicked article's position and URL, then calls the `onItemClick` method of the `NewsItemClickListener` (which is the `HomeScreenActivity`) to handle the click event (likely to open the article in a WebView).
- The `bind` method sets the title, description, and potentially loads an image for the corresponding article.

7. SavedNewsAdapter:

- This class is similar to `NewsAdapter` but is used in the `SavedNews` activity to display saved news articles retrieved from the database.
- It also takes an interface `NewsItemClickListener` as a constructor argument, in this case, for the `SavedNews` activity.
- It takes a list of `NewsItem` objects representing the saved news articles.
- The `onCreateViewHolder` method inflates the layout for each saved news item using a custom layout resource (`saved_news_layout.xml`).
- The `onBindViewHolder` method binds data from a `NewsItem` object to the corresponding ViewHolder, setting the title and description.
- It sets an onClickListener on the ViewHolder's itemView to handle clicks on saved news items.

- The `onItemClick` method of the `NewsItemClickListener` (which is the `SavedNews` activity) is called to handle the click event, likely to open the article in a WebView.

8. Data Models:

- **Article:** This class represents a single news article. It holds properties like author, content, description, publication date, source information (source object), title, URL, and image URL.
- **NewsApi:** This class represents the response structure from the news API. It contains a list of articles (`articles`), the API status (`status`), and the total number of results (`totalResults`).
- **Source:** This class represents the source information of a news article, including its ID and name.

9. Networking with Retrofit:

- **NewsAppInterface:** This interface defines the API endpoint for fetching news data using Retrofit.
    - The `getNewsData` method takes optional query parameters for category, API key, and country to filter the news results. It returns a `Call` object representing the asynchronous network request.
- **newsService:** This singleton object creates a Retrofit instance with the base URL ("https://newsapi.org/v2/").
    - It uses `GsonConverterFactory` to convert JSON responses from the API to corresponding data objects.
    - It creates an instance of the `NewsAppInterface` using the Retrofit instance.

10 . Database Access and Instance:

- **Instance:** This singleton object provides a central point to access the Room database instance.
    - It uses a `Volatile` variable `database` to ensure thread-safety when accessing the database.
    - The `getDatabase` method creates the database instance if it doesn't exist yet using `Room.databaseBuilder`. It also includes logic for database migrations (explained later).
    - The `giveInstance` method (commented out) was a potential alternative approach to provide the database instance, but `getDatabase` is generally preferred.

11. Database Schema (NewsDatabase and NewsItem):

- **NewsDatabase:** This abstract class represents the database itself.
    - It defines the entities (tables) it holds, which is currently just `NewsItem`.
    - It specifies the database version (`version = 2`), which is crucial for handling schema changes (migrations).
    - It defines an abstract method `newsItemDao()` to access the DAO (data access object) for interacting with the `NewsItem` table.
- **NewsItem:** This class represents a row (record) in the "news" table of the database.
    - It defines properties for each column:
        - `id` (primary key, auto-generated)
        - `title` (text)

        - `description` (text)
        - `url` (text) - This column was likely added in a later version (version 2)

12. Database Migration (MIGRATION_1_2):

- The provided code includes a `Migration` object named `MIGRATION_1_2`.
    - This migration is used to handle changes in the database schema between versions 1 and 2.
    - The `migrate` method defines the logic to update the existing table structure. In this case, it adds a new column named "url" to the existing "news" table using SQL (`ALTER TABLE`).

13. NewsItemDao:

- This interface defines methods for interacting with the "news" table in the database.
    - It uses annotations to specify the database operations for each method.
    - `deleteAllNewsItems`: Deletes all rows from the "news" table.
    - `insertNewsItem`: Inserts a new `NewsItem` object into the table (replacing any existing row with the same primary key value due to `OnConflictStrategy.REPLACE`).
    - `getAllNewsItems`: Retrieves all rows from the "news" table as a list of `NewsItem` objects.

14. GetLocation:

- This class facilitates retrieving the user's country code using the device's location.

- It checks for the necessary permission (`ACCESS_FINE_LOCATION`) before accessing location data.
- If permission is granted, it retrieves the last known location using the `LocationManager` service.
- It uses the `Geocoder` class to convert the latitude and longitude coordinates to a country code based on the device's locale.

- **Key Methods:**
    - `getLocation`: This method initiates the process of getting the user's location.
        - It checks for location permission.
            - If permission is not granted, it requests permission and returns null.
            - If permission is granted, it proceeds to call `requestLocation` to get the country code.
    - `requestLocation`: This method fetches the last known location from the `LocationManager`.
        - It retrieves the last known location from providers with best accuracy (considering permissions).
        - If a valid location is found, it extracts the country code using `getCountryCode` and returns it.
        - If no location is found, it returns an empty string.
    - `getLastKnownLocation`: This method retrieves the last known location from available providers considering permission checks.

- `getCountryCode`: This method uses the `Geocoder` to get the country code for a given latitude and longitude.

IV. Conclusion

Wrapping Up

This report served as a deep dive into the code powering a news application's onboarding screens. We explored how Jetpack Compose crafts a visually engaging and informative introduction for new users. Additionally, the `GetLocation` class unveiled functionalities for retrieving the user's country code, hinting at the potential for location-based personalization in the future.

Challenges and Learnings

Based on our analysis, building a news app might involve hurdles like:

- **Data Wrangling:** Efficiently fetching and managing the vast amount of news data an API throws our way.
- **UI Performance Under Pressure:** Maintaining a smooth and responsive user interface even when dealing with potentially extensive news content and images.
- **Location Personalization:** Finding the sweet spot between personalization and user privacy concerns regarding location data.

Looking Ahead: Enhancements Galore

Building upon the functionalities we identified, here are some exciting areas for improvement:

- **News API Integration:** Let's bring the real news in! Integrating a news API to retrieve and display actual articles.
- **User Preferences:** Taking personalization a step further by allowing users to log in and set preferences based on their interests, not just location.
- **Offline Functionality:** Empowering users to access previously viewed articles even without an internet connection (potentially using a Room database).
- **Testing, Testing, 1, 2, 3:** Implementing comprehensive testing strategies to ensure the app is rock-solid and user-friendly.