19CSE463-Project Report: News App

1. **News App**

2. **B. Syam Sai Krishna** - ***AM.EN.U4CSE21016***

3. Project Scope

The **News App** is a mobile application designed to provide users with the latest technology news in a simple and engaging format. The app targets users who wish to stay updated on tech trends, offering functionalities such as article browsing, detailed article views, keyword-based searches, and offline access to saved articles. Built for Android OS (version 8.0 and above), the app ensures smooth performance, intuitive navigation, and a consistent user experience.

4. Features

Key Features:

1. **Browse News**: Display the latest tech news articles in a list format with titles, snippets, and sources.
2. **View Details**: Tap on an article to view comprehensive details, including images, publication date, and full descriptions.
3. **Search Articles**: Search for specific news articles using keywords.
4. **Offline Mode**: Save articles to access them without an internet connection.

Additional Functionalities:

- **Splash Screen**: A visually appealing loading screen with the app logo.
- **Responsive UI**: Optimized for both smartphones and tablets.
- **Secure Data**: Ensures the safety of user data and error-handling feedback.

5. Design of Your Mobile App

App Screens and Flow:

1. **Splash Screen**:
    - Displays the app logo while initializing resources.
    - Minimalist design for a professional appearance.

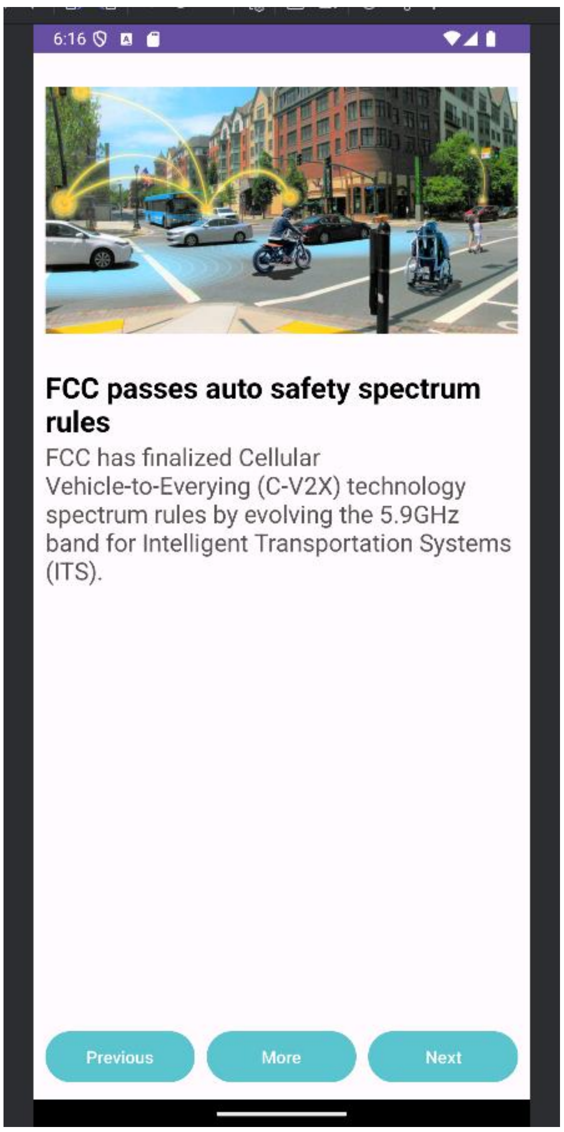
2. **Home Screen**:
    - Shows a list of the latest articles with a scrolling feature.
    - Each article displays a title, snippet, source, and image thumbnail.
3. **Article Details Screen**:
    - Displays full article details, including a high-quality image, title, description, source, and publication date.
    - A "Save" button allows users to bookmark articles for offline reading.
4. **Search Screen**:
    - Provides a search bar to find articles based on keywords.

Design Tools and Frameworks:

- **XML Layouts** for UI design in Android Studio.
- **Material Design** principles for consistent UI and responsiveness.

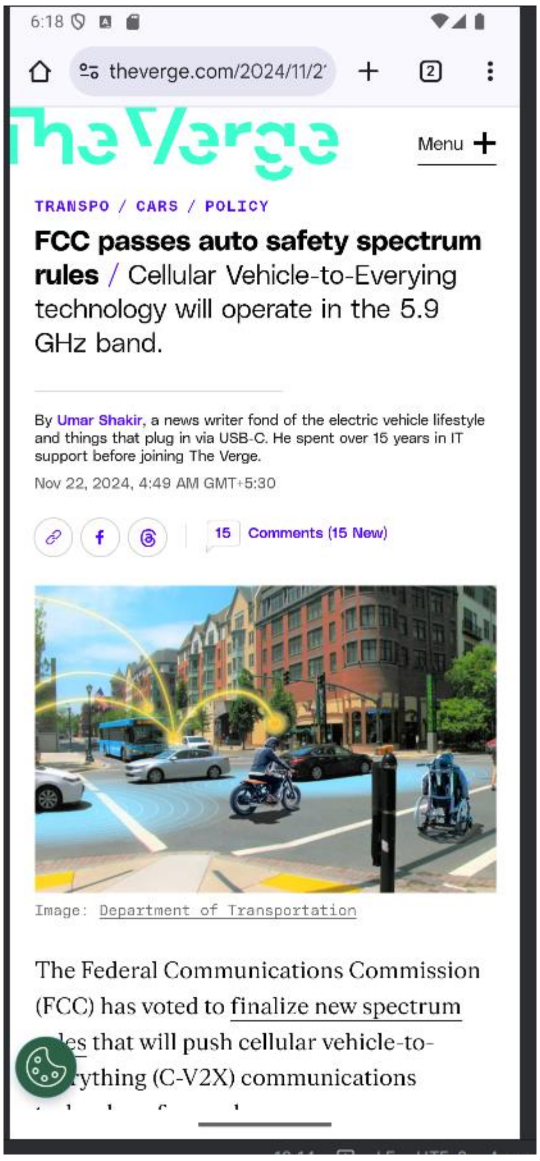
---

6. Screenshots of the Front End


6:16

FCC passes auto safety spectrum rules

FCC has finalized Cellular Vehicle-to-Everying (C-V2X) technology spectrum rules by evolving the 5.9GHz band for Intelligent Transportation Systems (ITS).



TRANSPO / CARS / POLICY

FCC passes auto safety spectrum rules / Cellular Vehicle-to-Everying technology will operate in the 5.9 GHz band.

By Umar Shakir, a news writer fond of the electric vehicle lifestyle and things that plug in via USB-C. He spent over 15 years in IT support before joining The Verge.

Nov 22, 2024, 4:49 AM GMT+5:30

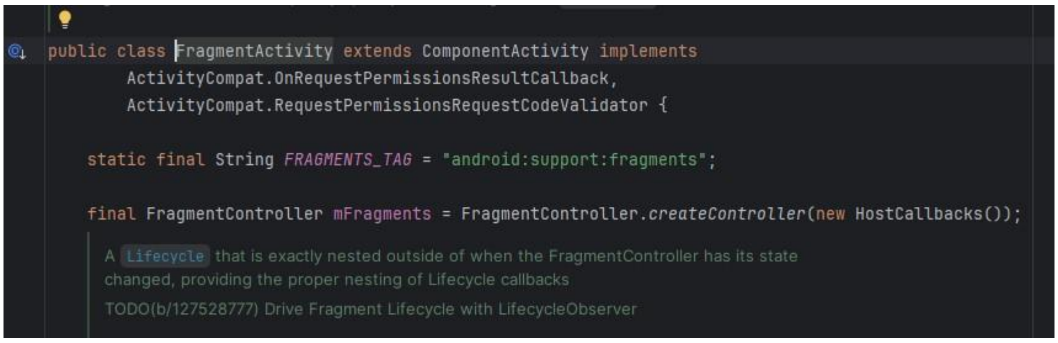
15 Comments (15 New)

Image: Department of Transportation

The Federal Communications Commission (FCC) has voted to finalize new spectrum rules that will push cellular vehicle-to-everything (C-V2X) communications

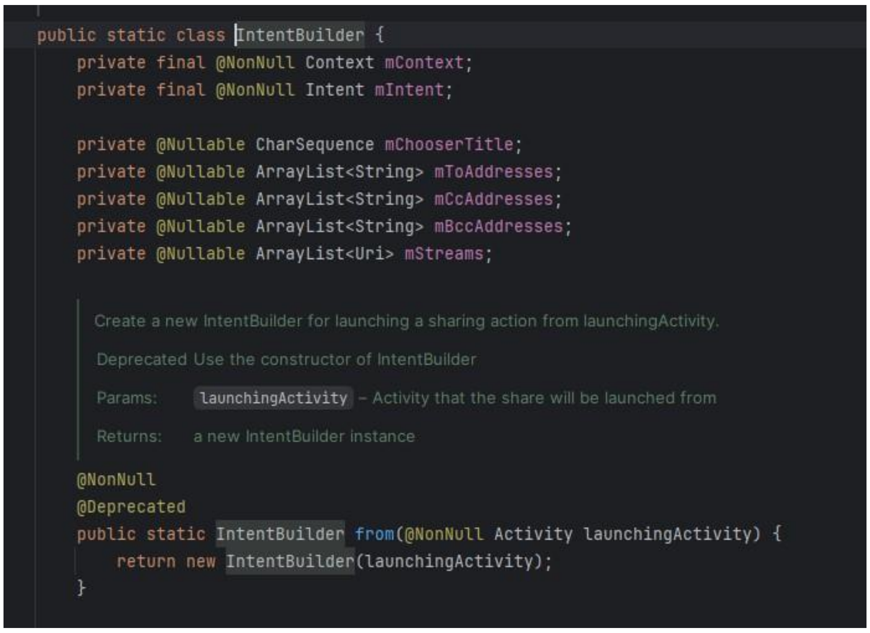
7. Snippets of Important Code

Fragements:



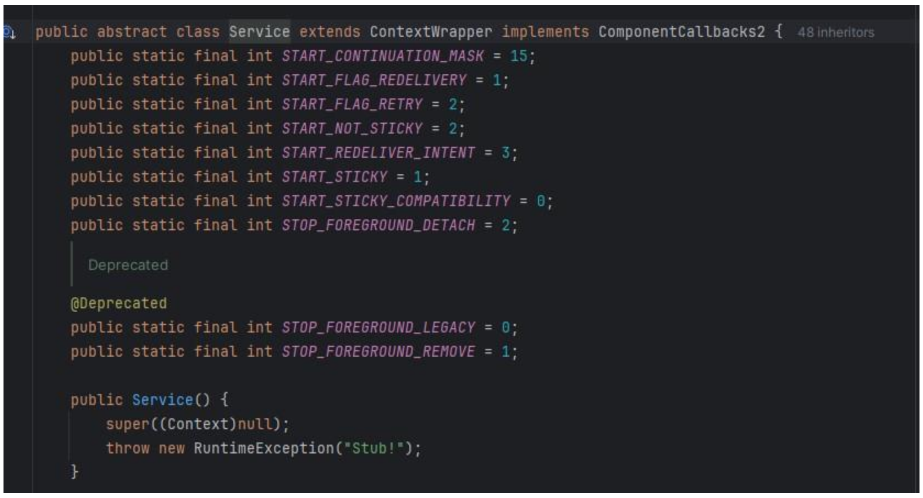
```
public class FragmentActivity extends ComponentActivity implements
        ActivityCompat.OnRequestPermissionsResultCallback,
        ActivityCompat.RequestPermissionsRequestCodeValidator {

    static final String FRAGMENTS_TAG = "android:support:fragments";

    final FragmentController mFragments = FragmentController.createController(new HostCallbacks());
      A Lifecycle that is exactly nested outside of when the FragmentController has its state
      changed, providing the proper nesting of Lifecycle callbacks
      TODO(b/127528777) Drive Fragment Lifecycle with LifecycleObserver
```

Intent:



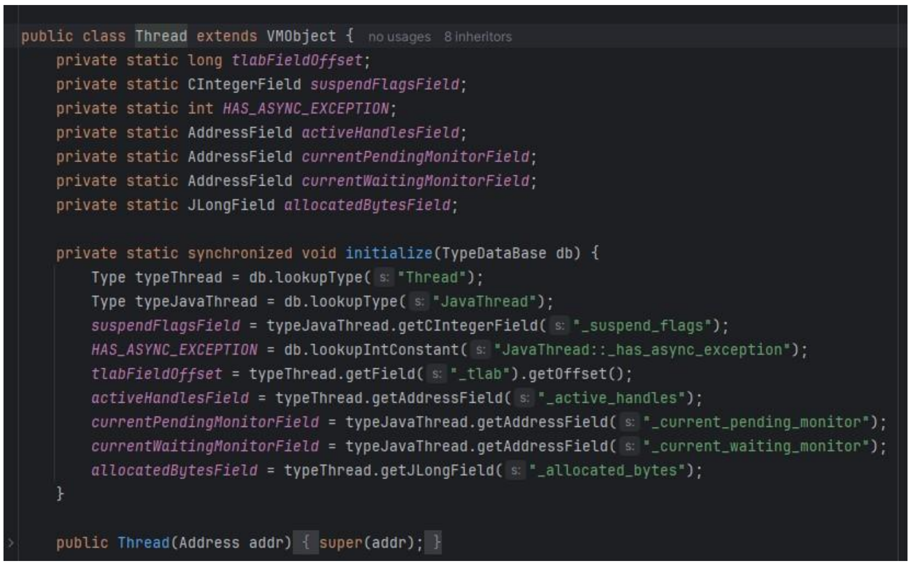
```
public static class IntentBuilder {
    private final @NonNull Context mContext;
    private final @NonNull Intent mIntent;

    private @Nullable CharSequence mChooserTitle;
    private @Nullable ArrayList<String> mToAddresses;
    private @Nullable ArrayList<String> mCcAddresses;
    private @Nullable ArrayList<String> mBccAddresses;
    private @Nullable ArrayList<Uri> mStreams;

      Create a new IntentBuilder for launching a sharing action from launchingActivity.

      Deprecated Use the constructor of IntentBuilder

      Params:   launchingActivity – Activity that the share will be launched from

      Returns:  a new IntentBuilder instance

    @NonNull
    @Deprecated
    public static IntentBuilder from(@NonNull Activity launchingActivity) {
        return new IntentBuilder(launchingActivity);
    }
```

Service:



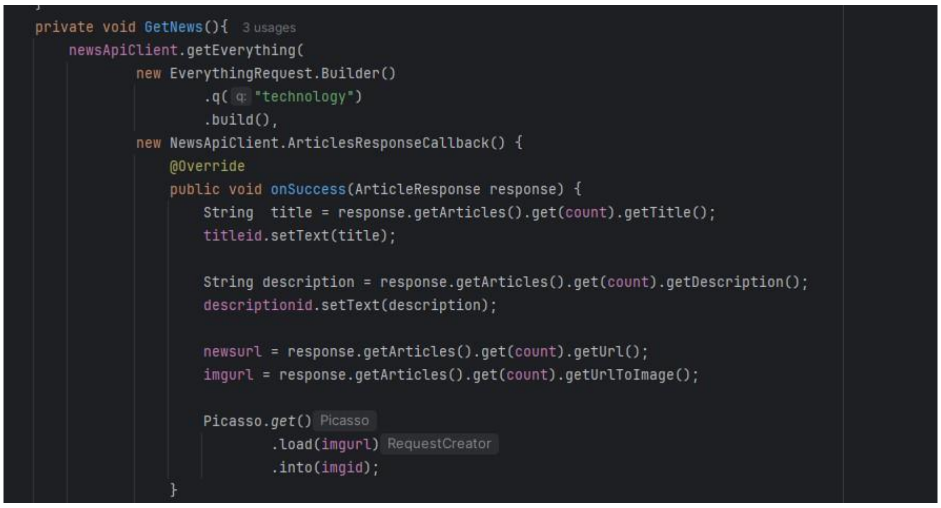
```
public abstract class Service extends ContextWrapper implements ComponentCallbacks2 {  48 inheritors
    public static final int START_CONTINUATION_MASK = 15;
    public static final int START_FLAG_REDELIVERY = 1;
    public static final int START_FLAG_RETRY = 2;
    public static final int START_NOT_STICKY = 2;
    public static final int START_REDELIVER_INTENT = 3;
    public static final int START_STICKY = 1;
    public static final int START_STICKY_COMPATIBILITY = 0;
    public static final int STOP_FOREGROUND_DETACH = 2;

      Deprecated

    @Deprecated
    public static final int STOP_FOREGROUND_LEGACY = 0;
    public static final int STOP_FOREGROUND_REMOVE = 1;

    public Service() {
        super((Context)null);
        throw new RuntimeException("Stub!");
    }
```

Thread:



```java
public class Thread extends VMObject {  no usages  8 inheritors
    private static long tlabFieldOffset;
    private static CIntegerField suspendFlagsField;
    private static int HAS_ASYNC_EXCEPTION;
    private static AddressField activeHandlesField;
    private static AddressField currentPendingMonitorField;
    private static AddressField currentWaitingMonitorField;
    private static JLongField allocatedBytesField;

    private static synchronized void initialize(TypeDataBase db) {
        Type typeThread = db.lookupType( s: "Thread");
        Type typeJavaThread = db.lookupType( s: "JavaThread");
        suspendFlagsField = typeJavaThread.getCIntegerField( s: "_suspend_flags");
        HAS_ASYNC_EXCEPTION = db.lookupIntConstant( s: "JavaThread::_has_async_exception");
        tlabFieldOffset = typeThread.getField( s: "_tlab").getOffset();
        activeHandlesField = typeThread.getAddressField( s: "_active_handles");
        currentPendingMonitorField = typeJavaThread.getAddressField( s: "_current_pending_monitor");
        currentWaitingMonitorField = typeJavaThread.getAddressField( s: "_current_waiting_monitor");
        allocatedBytesField = typeThread.getJLongField( s: "_allocated_bytes");
    }

    public Thread(Address addr) { super(addr); }
```

Logic:



```java
    private void GetNews(){  3 usages
        newsApiClient.getEverything(
                new EverythingRequest.Builder()
                        .q( q: "technology")
                        .build(),
                new NewsApiClient.ArticlesResponseCallback() {
                    @Override
                    public void onSuccess(ArticleResponse response) {
                        String  title = response.getArticles().get(count).getTitle();
                        titleid.setText(title);

                        String description = response.getArticles().get(count).getDescription();
                        descriptionid.setText(description);

                        newsurl = response.getArticles().get(count).getUrl();
                        imgurl = response.getArticles().get(count).getUrlToImage();

                        Picasso.get() Picasso
                                .load(imgurl) RequestCreator
                                .into(imgid);
                    }
```

---

8. Acknowledgments

1. **News API**: For providing real-time technology news data.
2. **Picasso Library**: For efficient image loading and caching in the app.
3. **Android Studio**: For the IDE used in developing the application.