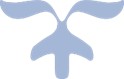


news AGGREGATOR APP

NEWSX





B.TECH CSE (B) 3RD YEAR

**SYNOPSIS**

SUBMITTED BY:

VAIBHAV VERMA 41918002718

ANSHUL RAJ 44518002718

**What is a News Aggregator?**

It is a web application which aggregates data (news articles) from multiple websites. Then presents the data in one location.

News aggregator service is a very important start of the day.

There are various publications and news sites online. They publish their content on multiple platforms. Now, imagine when you open 10-20 news sites every day. The time you waste to gain information. Information gain is everything in today’s world.

It can give you leverage over those who don’t have it. Now, is there a way we can make it easier? Yes!!

A news aggregator makes this task easier. In a news aggregator, you can select the websites you want to follow. Then the news aggregator collects the articles for you. And, you are just a click away to get information from various websites.

This task otherwise takes too much time on our schedule.

**Why use news aggregators?**

The flow of information never stops. Every day, more and more data is generated on the internet, and some of that information affects your organization. Business leaders need to keep as up to date as much as possible with the relevant information in their respective industries, but that’s not always easy to do. News aggregators give you the ability to read through the latest news stories that have an impact on you and your business, all from one convenient location. There’s no need to comb through dozens of websites hoping to find the news that you want. News aggregators take care of the busy work and put that information at your fingertips.

Staying on top of the news like this helps you to remain competitive. It helps you to know what the latest trends are and what your closest competitors are up to. As the saying goes, “knowledge is power,” and news aggregators are an excellent way to obtain knowledge and use it to further your business goals. They help you see things from a fresh perspective, in turn giving you the information you need to keep your business at the forefront of your industry.

**How does a news aggregator work?**

In the early days of news aggregation, individual curators would search the web to place stories on their own news aggregator websites. Some still operate manually, but the vast majority of news aggregator software and applications operate through systems and processes that have been coded in. Most follow a similar workflow to handle all of the new web stories that are added to the internet every day.

The first part of that workflow deals with gathering data from previously determined websites. This involves crawling a web page, sometimes checking on RSS feeds to collect that data. The next part extracts articles from that gathered data. This is the part of the workflow where the relevancy of the articles is determined. Next comes the step where articles are grouped together based upon the topics they discuss. There are many different methods that can be used for this. Some of the most common approaches include keyword-based techniques and topic modelling.

After that, an article is chosen that best represents a group of articles based on a topic. This is usually the article that the aggregator shows first. Finally, the topics of the articles are then visualized so the user can access them easily. Most of the time, the user can determine which topics he or she is most interested in so the news aggregator can focus on them.

**Prerequisite**

We are using the following libraries and programming languages:

* **Back End:** Java, News API, Retrofit, Android X browser library
* **Front End:** XML
* **Database:** SQL lite

**Software Requirements:**

* Windows 7, Windows 10
* Android Studio
* Android Emulator

**Hardware Components:**

* Android device
* Android Version 4.0 and above
* Minimum RAM required: 1GB
* Processor:
  + - Qualcomm:- Minimum Snapdragon 615
    - MediaTek: - Minimum MT6753

**The importance of news aggregation**

Using news aggregation is one of the best ways to stay on top of the news and topics you want. They offer convenience and time-saving features. At a time when data is more important than ever, they’re a handy way to compile relevant data and present it into a form that’s easy to consume and process. Businesses that use news aggregation will find they stay ahead of their competitors and ready to handle any future challenges.

**SOFTWARE REQUIREMENT SPECIFICATION**

**Introduction:**

It is a web application which aggregates data (news articles) from multiple websites. Then presents the data in one location.

There are hundreds of news websites, they do cover news on several broad topics, out of which only a few of them are of our interest. A news aggregator can be a tool to save a lot of time and with some modifications and filtration we can fine tune it to show only news of our interest.

A news aggregator can be a useful tool to get information within short time.

**Purpose & Scope:**

Visiting many separate websites frequently to find out if content on the site has been updated can take a long time.

* Aggregation technology helps to consolidate many websites into one page that can show only the new or updated information from many sites.
* Aggregators reduce the time and effort needed to regularly check websites for updates, creating a unique information space or *personal newspaper*.
* Once subscribed to a feed, an aggregator is able to check for new content at user-determined intervals and retrieve the update.

**THE OVERALL DESCRIPTION:**

* **Product perspective**
* **Hardware interfaces**
* The database connectivity requires a hardware configuration of Android version 4.0, minimum of 1GB ram and a decent Internet connection
* **Software interfaces:**
* **Back End:** Java, News API, Retrofit, Android X browser library
* **Front End:** XML
* **Database:** SQL lite

**Operations:**

In a news aggregator, you can select the websites you want to follow. Then the news aggregator collects the articles for you. And, you are just a click away to get information from various websites.

**Product Functions:**

News aggregator fundamentally follows the following news aggregation workflow in order to aggregate the vast amount of news articles to few, important topics:

1. **Data gathering**, i.e., crawl articles from news websites. Simple approaches regularly check RSS feeds, while others, such as Google News, visit any link on a news website and crawl whole HTML pages.
2. **Article extraction** from raw website data, such as HTML. The goal of this phase is to extract only the relevant parts of the website. In news aggregation, this is typically the title, the lead paragraph, the remainder of the text, and further components, such as the main image and author.
3. **Clustering related articles**, i.e., find and group related articles about the same topic or event. Commonly applied techniques to find related articles are topic modelling.
4. **Summarization** of related articles. The goal is to find the “most representative” article in a group of related articles, so that later in the visualization not just any article is shown, but a “good one”.
5. **Visualization**, e.g., present the most important topics to users. This is what is visible to the users. Most commercial news aggregators show a list of topics, where each topic is then represented by a list of related articles.

**User characteristics**

* The intended users of this software need not have specific knowledge as to what is the internal operation of the system. Thus, the end user is at a high level of abstraction that allows easier, faster operation and reduces the knowledge requirement of end user.
* The Product is absolutely user friendly, so the intended users can be the naïve users.
* The product does not expect the user to possess any technical background.

**SPECIFIC REQUIREMENTS**

* **FRONT – END DESCRIPTION**

XML stands for **E**xtensible **M**arkup **L**anguage. It is a text-based markup language derived from Standard Generalized Markup Language (SGML).

XML tags identify the data and are used to store and organize the data, rather than specifying how to display it like HTML tags, which are used to display the data. XML is not going to replace HTML in the near future, but it introduces new possibilities by adopting many successful features of HTML.

There are three important characteristics of XML that make it useful in a variety of systems and solutions −

* **XML is extensible** − XML allows you to create your own self-descriptive tags, or language, that suits your application.
* **XML carries the data, does not present it** − XML allows you to store the data irrespective of how it will be presented.
* **XML is a public standard** − XML was developed by an organization called the World Wide Web Consortium (W3C) and is available as an open standard.
* **BACK – END DESCRIPTION**

**JAVA**

Java is the native language used by Android, applications that communicate with the operating system and directly use the hardware uses Java. This language allows to create any program and supports almost all types of machines, and OS X be it Android, Windows, or Linux.  Java was developed by Sun Microsystems (now the property of Oracle) and one can use [Microservices with Java](https://www.xenonstack.com/insights/microservices-java-go/).

**Java is an object-oriented programming language, which is one of the most beloved languages by developers.**

**Pros**

1. Java is multiplatform and works on practically any device, server or operating system
2. Java coding is robust, and it is impossible for Java instruction to corrupt memory or compromise data from other applications of OS X
3. Java is object-oriented and makes it easy to create modular applications and reuse the parts that contribute to the robustness
4. It is ready to use, and with Java, you will get a lot of third-part code ready to be used.
5. While comparing Kotlin vs Java performance, it to other languages, Java is easy to use, compiling and makes debugging and deploying even simpler
6. Java is an open-source language that ensures safety since many of its libraries are managed by trusted companies like Google, Apache, and others.

**NEWS API**

News API is a simple HTTP REST API for searching and retrieving live articles from all over the web. It can help you answer questions like:

* What top stories is TechCrunch running right now?
* What new articles were published about the next iPhone today?
* Has my company or product been mentioned or reviewed by any blogs recently?

You can search for articles with any combination of the following criteria:

* **Keyword or phrase**. Eg: find all articles containing the word 'Microsoft'.
* **Date published**. Eg: find all articles published yesterday.
* **Source name**. Eg: find all articles by 'TechCrunch'.
* **Source domain name**. Eg: find all articles published on thenextweb.com.
* **Language**. Eg: find all articles written in English.

You can sort the results in the following orders:

* Date published
* Relevancy to search keyword
* Popularity of source

You need an API key to use the API - this is a unique key that identifies your requests. They're free for development, open-source, and non-commercial use.

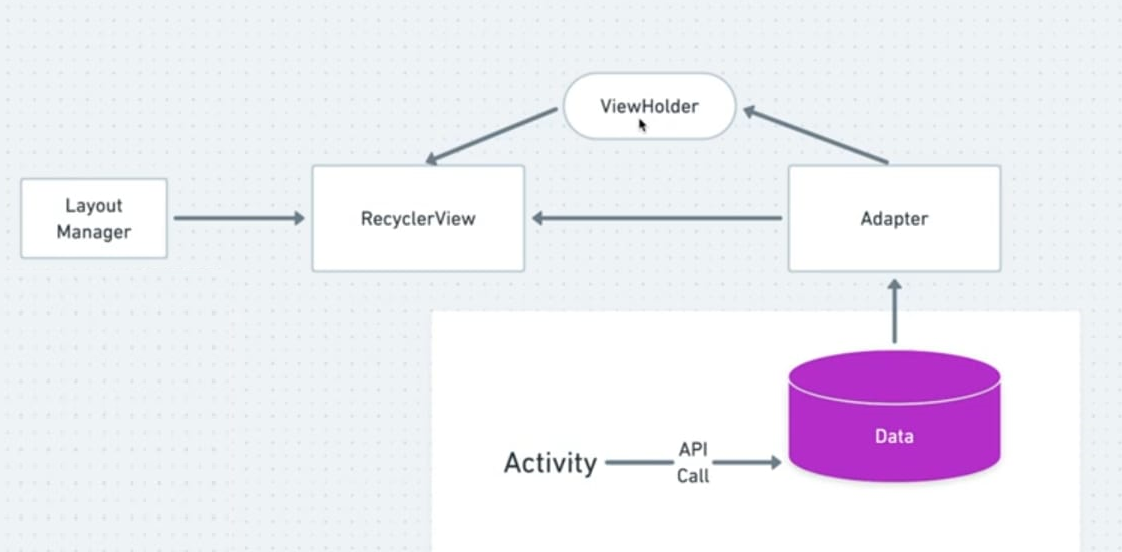
**RETROFIT**

Retrofit is a REST Client for Java and Android. It makes it relatively easy to retrieve and upload JSON (or other structured data) via a REST based webservice. In Retrofit you configure which converter is used for the data serialization. Typically for JSON you use GSon, but you can add custom converters to process XML or other protocols. Retrofit uses the OkHttp library for HTTP requests.

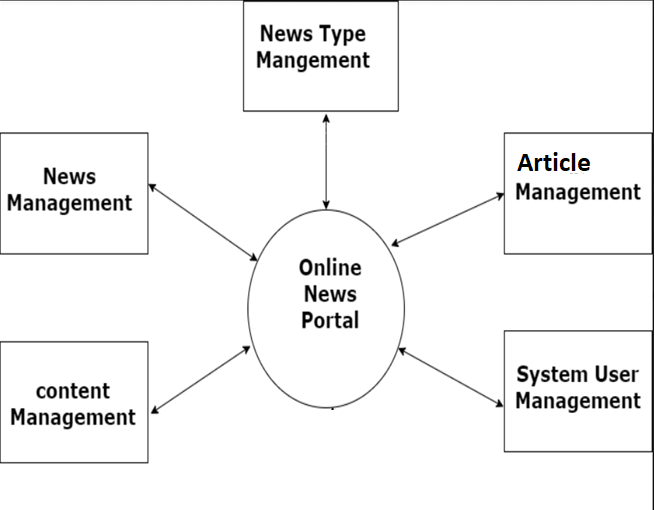
**Android X Browser Library**

We want our app not to open in Chrome instead we will create custom tab using this library

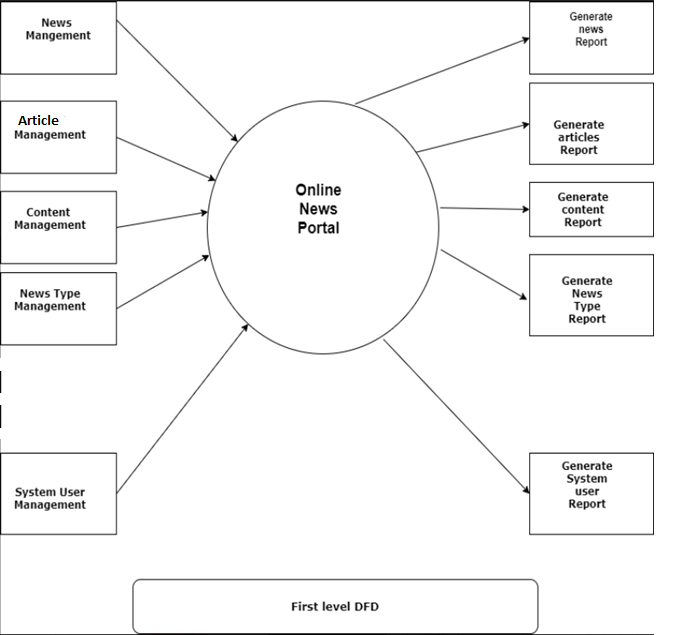
**View of App**



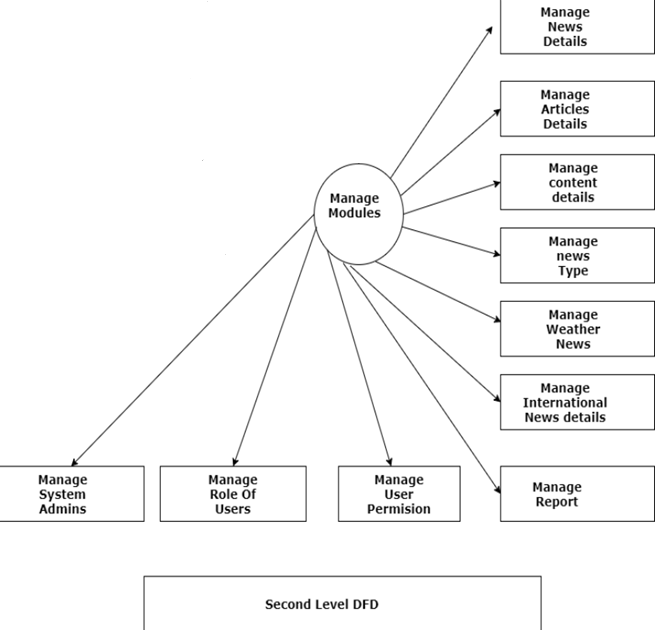
1. **Zero Level Data Flow Diagram (0 Level DFD)**

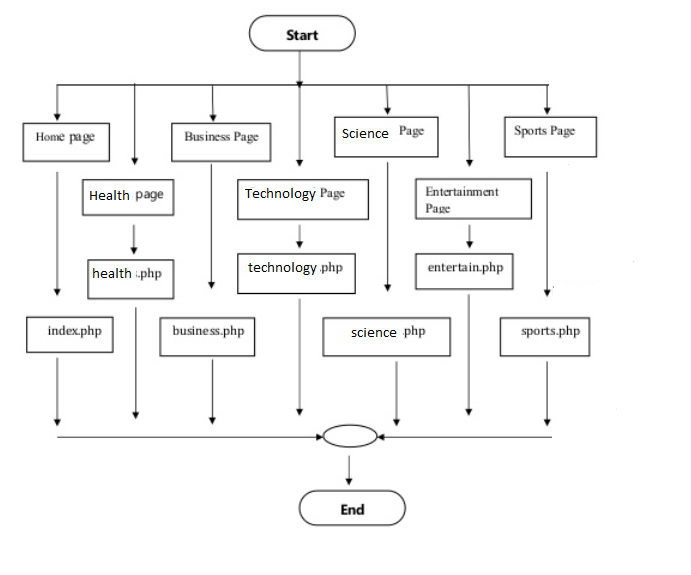


1. **First Level Flow Diagram (1st Level DFD)**

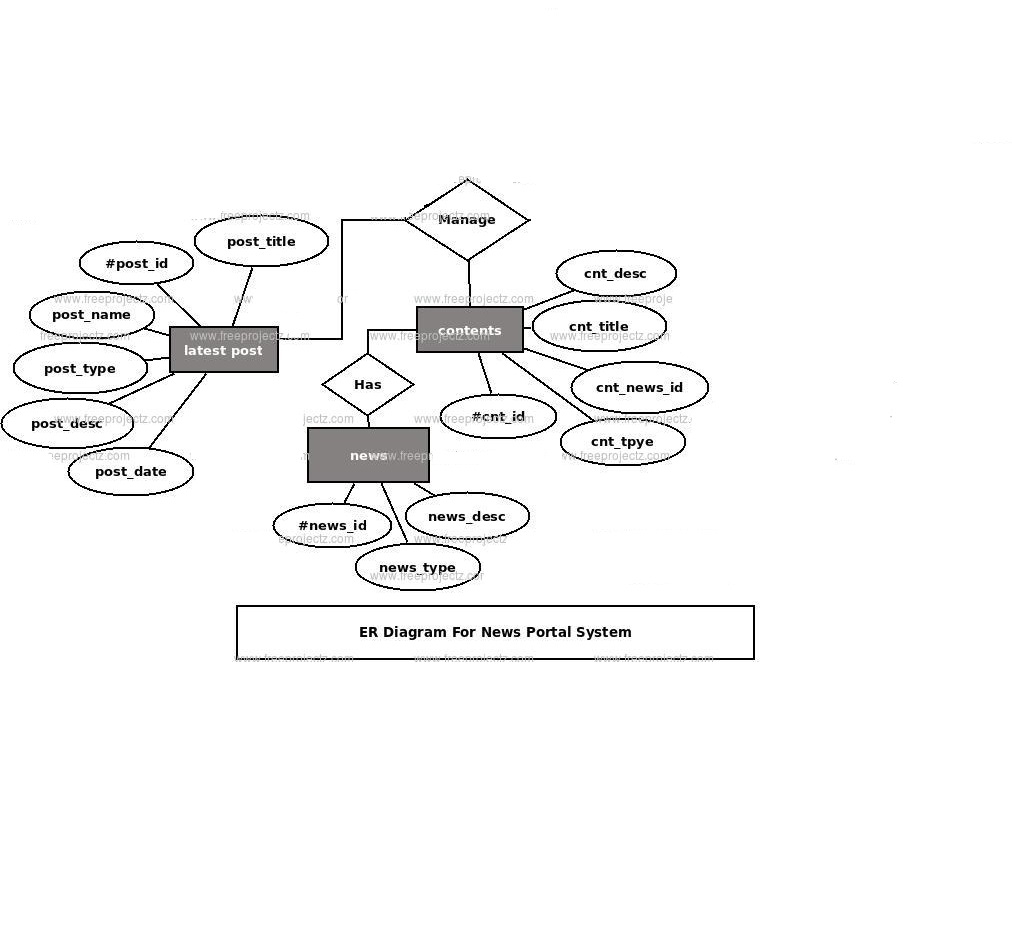


1. **Second Level Data Flow Diagram (2nd Level DFD)**





**ENTITY RELATIONSHIP DIAGRAM**



**SOURCE CODE**

**Front end**

<?xml version="1.0" encoding="utf-8"?>  
<layout xmlns:bind="http://schemas.android.com/apk/res-auto">  
  
 <data>  
  
 <import type="com.example.vaibansh.newsapp.utils.BindingUtils" />  
  
 <variable  
 name="article"  
 type="com.example.vaibansh.newsapp.models.Article" />  
 </data>  
  
 <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <android.support.constraint.ConstraintLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:paddingBottom="16dp"  
 tools:context=".ui.news.DetailActivity">  
  
 <View  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 android:background="@color/recyclerViewDivider"  
 app:layout\_constraintBottom\_toBottomOf="@id/tv\_news\_source"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/iv\_news\_image" />  
  
 <ImageView  
 android:id="@+id/iv\_news\_image"  
 android:layout\_width="match\_parent"  
 android:layout\_height="0dp"  
 android:contentDescription="@string/image\_desc"  
 android:foreground="@drawable/news\_image\_gradient"  
 app:layout\_constraintDimensionRatio="h,16:9"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 bind:articleUrl="@{article.url}"  
 bind:urlToImage="@{article.urlToImage}" />  
  
 <ImageButton  
 android:id="@+id/iv\_save"  
 android:layout\_width="48dp"  
 android:layout\_height="48dp"  
 android:layout\_marginEnd="4dp"  
 android:background="?selectableItemBackgroundBorderless"  
 android:contentDescription="@string/image\_desc\_save\_button"  
 app:layout\_constraintEnd\_toStartOf="@id/iv\_share"  
 app:layout\_constraintTop\_toTopOf="@id/tv\_time"  
 app:srcCompat="@drawable/ic\_save" />  
  
 <ImageButton  
 android:id="@+id/iv\_share"  
 android:layout\_width="48dp"  
 android:layout\_height="48dp"  
 android:background="?selectableItemBackgroundBorderless"  
 android:contentDescription="@string/image\_desc\_share\_button"  
 app:layout\_constraintEnd\_toStartOf="@id/guideline\_right"  
 app:layout\_constraintTop\_toTopOf="@id/tv\_time"  
 app:srcCompat="@drawable/ic\_share" />  
  
 <TextView  
 android:id="@+id/tv\_time"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:text="@{BindingUtils.formatDateForDetails(article.publishedAt)}"  
 android:textSize="16sp"  
 android:textStyle="bold"  
 app:layout\_constraintStart\_toStartOf="@id/guideline\_left"  
 app:layout\_constraintTop\_toBottomOf="@+id/tv\_news\_title"  
 tools:text="@tools:sample/date/ddmmyy" />  
  
 <TextView  
 android:id="@+id/tv\_news\_source"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="4dp"  
 android:paddingBottom="18dp"  
 android:text="@{article.source.name}"  
 android:textSize="16sp"  
 android:textStyle="bold"  
 app:layout\_constraintStart\_toStartOf="@id/guideline\_left"  
 app:layout\_constraintTop\_toBottomOf="@id/tv\_time"  
 tools:text="The Washington Post" />  
  
 <TextView  
 android:id="@+id/tv\_news\_title"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="16dp"  
 android:lineSpacingMultiplier="1.1"  
 android:text="@{article.title}"  
 android:textColor="@color/colorTextSecondary"  
 android:textSize="24sp"  
 android:textStyle="bold"  
 app:layout\_constraintEnd\_toEndOf="@id/guideline\_right"  
 app:layout\_constraintStart\_toStartOf="@id/guideline\_left"  
 app:layout\_constraintTop\_toBottomOf="@id/iv\_news\_image"  
 tools:text="@string/sample\_title" />  
  
 <TextView  
 android:id="@+id/tv\_news\_desc"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="32dp"  
 android:alpha="0.6"  
 android:lineSpacingMultiplier="1.3"  
 android:text="@{article.description}"  
 android:textColor="@color/colorTextSecondary"  
 android:textSize="18sp"  
 app:layout\_constraintEnd\_toEndOf="@id/guideline\_right"  
 app:layout\_constraintStart\_toStartOf="@id/guideline\_left"  
 app:layout\_constraintTop\_toBottomOf="@id/tv\_news\_source"  
 tools:layout\_editor\_absoluteY="365dp"  
 tools:text="@string/sample\_description" />  
  
 <TextView  
 android:id="@+id/tv\_news\_content"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="32dp"  
 android:alpha="0.6"  
 android:lineSpacingMultiplier="1.3"  
 android:text="@{BindingUtils.truncateExtra(article.content)}"  
 android:textColor="@color/colorTextSecondary"  
 android:textSize="18sp"  
 app:layout\_constraintEnd\_toEndOf="@id/guideline\_right"  
 app:layout\_constraintStart\_toStartOf="@id/guideline\_left"  
 app:layout\_constraintTop\_toBottomOf="@id/tv\_news\_desc"  
 tools:text="@string/sample\_description" />  
  
 <Button  
 android:id="@+id/btn\_read\_full"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="32dp"  
 android:background="@drawable/button\_selector"  
 android:paddingStart="16dp"  
 android:paddingEnd="16dp"  
 android:text="@string/button\_read\_full\_story"  
 android:textSize="16sp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@id/tv\_news\_content" />  
  
 <android.support.v7.widget.Toolbar  
 android:id="@+id/toolbar"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:background="?actionBarItemBackground"  
 android:fitsSystemWindows="true"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:navigationIcon="@drawable/ic\_close" />  
  
 <android.support.constraint.Guideline  
 android:id="@+id/guideline\_left"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 app:layout\_constraintGuide\_begin="12dp" />  
  
 <android.support.constraint.Guideline  
 android:id="@+id/guideline\_right"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical"  
 app:layout\_constraintGuide\_end="12dp" />  
  
 <android.support.constraint.Barrier  
 android:id="@+id/bottom\_barrier"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:barrierDirection="bottom"  
 app:constraint\_referenced\_ids="tv\_news\_desc,tv\_news\_content,tv\_news\_title" />  
  
  
  
 </android.support.constraint.ConstraintLayout>  
  
 </ScrollView>  
</layout>

**Code for News Api**

package com.example.vaibansh.newsapp.network;  
  
import …  
  
import retrofit2.Call;  
import retrofit2.http.GET;  
import retrofit2.http.Headers;  
import retrofit2.http.Query;  
  
*/\*\*  
 \* An Api interface to send network requests  
 \* Includes Category enum that provides category names for requests  
 \*/*public interface NewsApi {  
 String *API\_KEY* = BuildConfig.*NewsApiKey*;  
  
 @Headers("X-Api-Key:" + *API\_KEY*)  
 @GET("/v2/top-headlines")  
 Call<ArticleResponseWrapper> getHeadlines(  
 @Query("category") String category,  
 @Query("country") String country  
 );  
  
 @Headers("X-Api-Key:" + *API\_KEY*)  
 @GET("/v2/top-headlines")  
 Call<ArticleResponseWrapper> getHeadlinesBySource(  
 @Query("sources") String source  
 );  
  
 @Headers("X-Api-Key:" + *API\_KEY*)  
 @GET("/v2/sources")  
 Call<SourceResponseWrapper> getSources(  
 @Query("category") String category,  
 @Query("country") String country,  
 @Query("language") String language  
 );  
  
 enum Category {  
 *business*("Business"),  
 *entertainment*("Entertainment"),  
 *general*("General"),  
 *health*("Health"),  
 *science*("Science"),  
 *sports*("Sports"),  
 *technology*("Technology");  
  
 public final String title;  
  
 Category(String title) {  
 this.title = title;  
 }  
 }  
}

**Main Activity**

package com.example.vaibansh.newsapp.ui;  
  
import ….  
  
  
  
public class MainActivity extends AppCompatActivity implements OptionsBottomSheet.OptionsBottomSheetListener {  
 private final FragmentManager fragmentManager = getSupportFragmentManager();  
 private ActivityMainBinding binding;  
 private HeadlinesFragment headlinesFragment;  
 private SourceFragment sourceFragment;  
 private NewsFragment newsFragment;  
 private FirebaseAnalytics mFirebaseAnalytics;  
 private final BottomNavigationView.OnNavigationItemSelectedListener mOnNavigationItemSelectedListener  
 = new BottomNavigationView.OnNavigationItemSelectedListener() {  
  
 @Override  
 public boolean onNavigationItemSelected(@NonNull MenuItem item) {  
 Bundle bundle = new Bundle();  
 switch (item.getItemId()) {  
 case R.id.*navigation\_headlines*:  
 fragmentManager.beginTransaction()  
 .replace(R.id.*fragment\_container*, headlinesFragment)  
 .commit();  
 bundle.putString(  
 FirebaseAnalytics.Param.*ITEM\_CATEGORY*,  
 getString(R.string.*title\_headlines*)  
 );  
 return true;  
 case R.id.*navigation\_saved*:  
 if (newsFragment == null) {  
 newsFragment = NewsFragment.*newInstance*(null);  
 }  
 getSupportFragmentManager().beginTransaction()  
 .replace(R.id.*fragment\_container*, newsFragment)  
 .commit();  
 bundle.putString(  
 FirebaseAnalytics.Param.*ITEM\_CATEGORY*,  
 getString(R.string.*title\_saved*)  
 );  
 return true;  
   
 }  
 mFirebaseAnalytics.logEvent(FirebaseAnalytics.Event.*SELECT\_CONTENT*, bundle);  
 return false;  
 }  
 };  
 private Snackbar snackbar;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 if (BuildConfig.*DEBUG*) {  
 Timber.*plant*(new Timber.DebugTree());  
 }  
  
 // Bind data using DataBinding  
 binding = DataBindingUtil.*setContentView*(this, R.layout.*activity\_main*);  
 binding.navigation.setOnNavigationItemSelectedListener(mOnNavigationItemSelectedListener);  
  
 // Obtain the FirebaseAnalytics instance.  
 mFirebaseAnalytics = FirebaseAnalytics.*getInstance*(this);  
  
 if (savedInstanceState == null) {  
 // Add a default fragment  
 headlinesFragment = HeadlinesFragment.*newInstance*();  
 fragmentManager.beginTransaction()  
 .add(R.id.*fragment\_container*, headlinesFragment)  
 .commit();  
 }  
  
 setupToolbar();  
  
 final AppWidgetManager appWidgetManager = AppWidgetManager.*getInstance*(this);  
  
 final LiveData<List<Article>> saved = NewsRepository.*getInstance*(this).getSaved();  
 saved.observe(this, new Observer<List<Article>>() {  
 @Override  
 public void onChanged(@Nullable List<Article> articles) {  
 if (articles != null) {  
 int[] appWidgetIds = appWidgetManager.getAppWidgetIds(new ComponentName(getApplicationContext(), SavedNewsWidget.class));  
 if (articles.size() == 0) {  
 SavedNewsWidget.*updateNewsWidgets*(getApplicationContext(), appWidgetManager, articles, -1, appWidgetIds);  
 } else {  
 SavedNewsWidget.*updateNewsWidgets*(getApplicationContext(), appWidgetManager, articles, 0, appWidgetIds);  
 }  
 }  
 }  
 });  
 }  
  
 private void setupToolbar() {  
 setSupportActionBar(binding.toolbar);  
 ActionBar actionBar = getSupportActionBar();  
 if (actionBar != null) {  
 actionBar.setTitle(getString(R.string.*app\_name*));  
 //Remove trailing space from toolbar  
 binding.toolbar.setContentInsetsAbsolute(10, 10);  
 }  
 }  
  
 @Override  
 public void onSaveToggle(String text) {  
 if (snackbar == null) {  
 snackbar = Snackbar.*make*(binding.coordinator, "Hello", Snackbar.*LENGTH\_SHORT*);  
 final CoordinatorLayout.LayoutParams params = (CoordinatorLayout.LayoutParams) snackbar.getView().getLayoutParams();  
 params.setMargins(  
 (int) getResources().getDimension(R.dimen.*snackbar\_margin\_vertical*),  
 0,  
 (int) getResources().getDimension(R.dimen.*snackbar\_margin\_vertical*),  
 (int) getResources().getDimension(R.dimen.*snackbar\_margin\_horizontal*)  
 );  
 snackbar.getView().setLayoutParams(params);  
 snackbar.getView().setPadding(  
 (int) getResources().getDimension(R.dimen.*snackbar\_padding*),  
 (int) getResources().getDimension(R.dimen.*snackbar\_padding*),  
 (int) getResources().getDimension(R.dimen.*snackbar\_padding*),  
 (int) getResources().getDimension(R.dimen.*snackbar\_padding*)  
 );  
 }  
 if (snackbar.isShown()) {  
 snackbar.dismiss();  
 }  
 snackbar.setText(text);  
 snackbar.show();  
 }  
}

**TESTING**

**UNIT TEST**

package com.example.vaibansh.newsapp;  
  
import org.junit.Test;  
  
import static org.junit.Assert.*assertEquals*;  
  
*/\*\*  
 \* Example local unit test, which will execute on the development machine (host).  
 \*  
 \** ***@see*** *<a href="http://d.android.com/tools/testing">Testing documentation</a>  
 \*/*public class ExampleUnitTest {  
 @Test  
 public void addition\_isCorrect() {  
 *assertEquals*(4, 2 + 2);  
 }  
}

**INSTRUMENTED TEST**

package com.example.vaibansh.newsapp;  
  
import android.content.Context;  
import android.support.test.InstrumentationRegistry;  
import android.support.test.runner.AndroidJUnit4;  
  
import org.junit.Test;  
import org.junit.runner.RunWith;  
  
import static org.junit.Assert.*assertEquals*;  
  
*/\*\*  
 \* Instrumented test, which will execute on an Android device.  
 \*  
 \** ***@see*** *<a href="http://d.android.com/tools/testing">Testing documentation</a>  
 \*/*@RunWith(AndroidJUnit4.class)  
public class ExampleInstrumentedTest {  
 @Test  
 public void useAppContext() {  
 // Context of the app under test.  
 Context appContext = InstrumentationRegistry.*getTargetContext*();  
  
 *assertEquals*("com.example.abhishek.newsapp", appContext.getPackageName());  
 }  
}

**TEST OUTPUT :**

16:32 \* daemon started successfully

**//UNIT TEST**

17:31 Executing tasks: [:app:assembleDebug] in project A:\NewsApp

17:31 Gradle build finished in 45 s 980 ms

17:31 Install successfully finished in 974 ms.: App restart successful without requiring a re-install.

17:33 Executing tasks: [:app:generateDebugSources, :app:compileDebugSources, :app:createMockableJar, :app:compileDebugUnitTestSources] in project A:\NewsApp

17:33 Gradle build finished in 16 s 313 ms

17:33 Tests passed: 1

**//INSTRUMENTED TEST**

17:34 Executing tasks: [:app:assembleDebug, :app:assembleDebugAndroidTest] in project A:\NewsApp

17:34 Gradle build finished in 14 s 301 ms

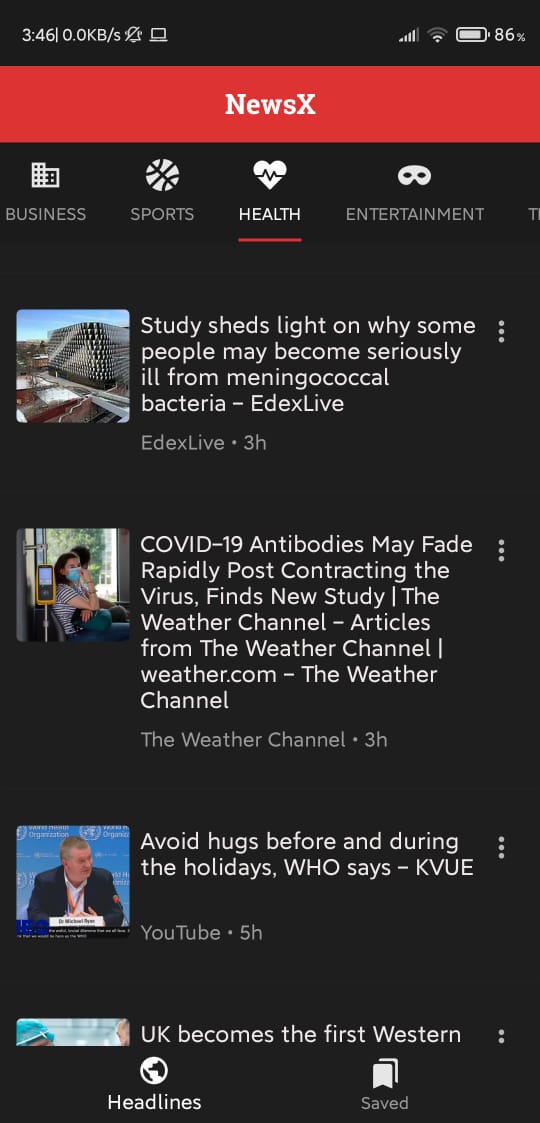
17:34 Install successfully finished in 1 s 711 ms.: App restart successful without re-installing the following APK(s):com.example.vaibansh.newsapp

17:34 Tests passed: 1

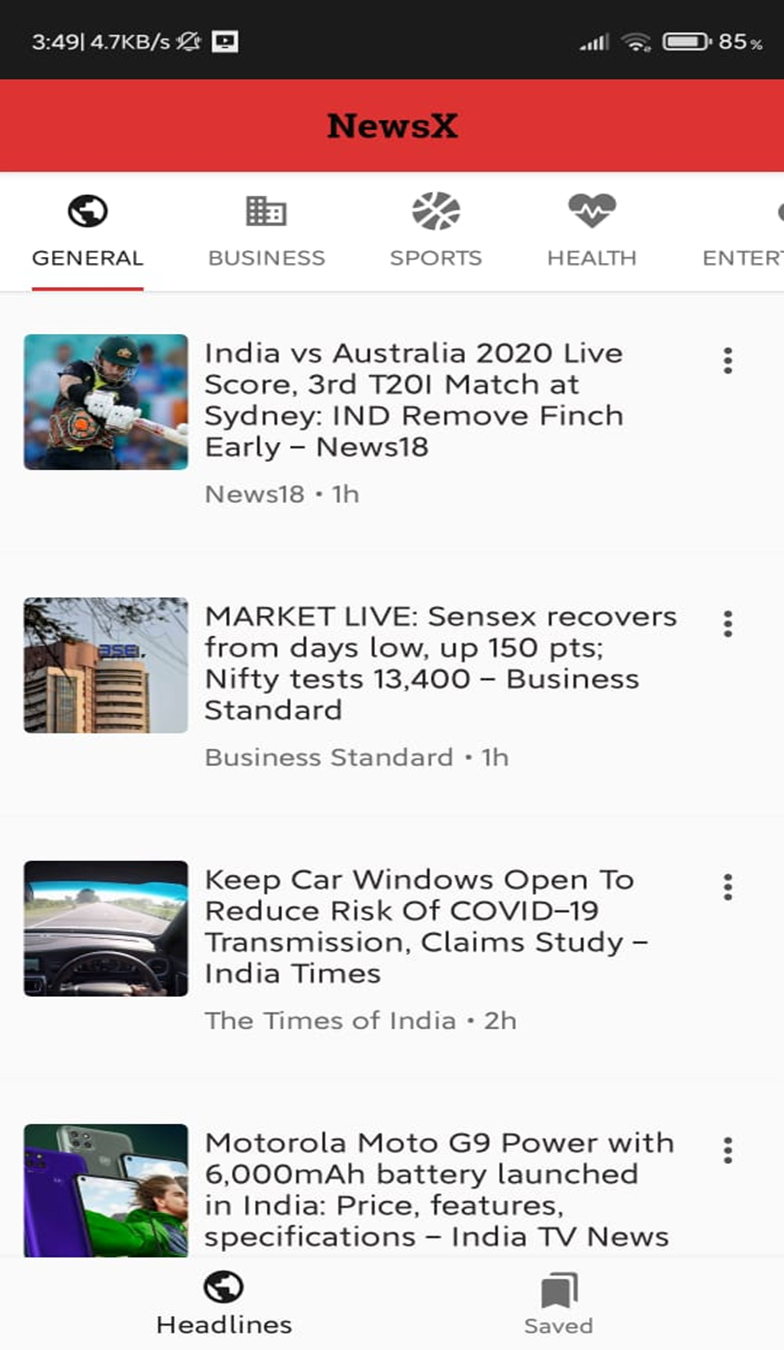
**IMAGES OF THE FINAL PRODUCT**

**1) Different categories of news**

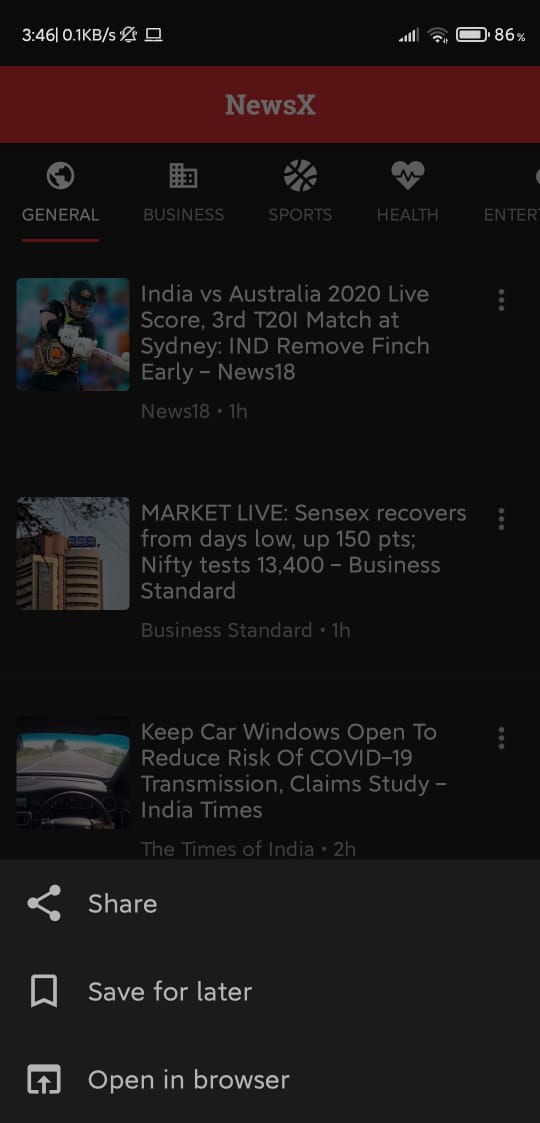
**In Dark mode**



**In Normal mode**



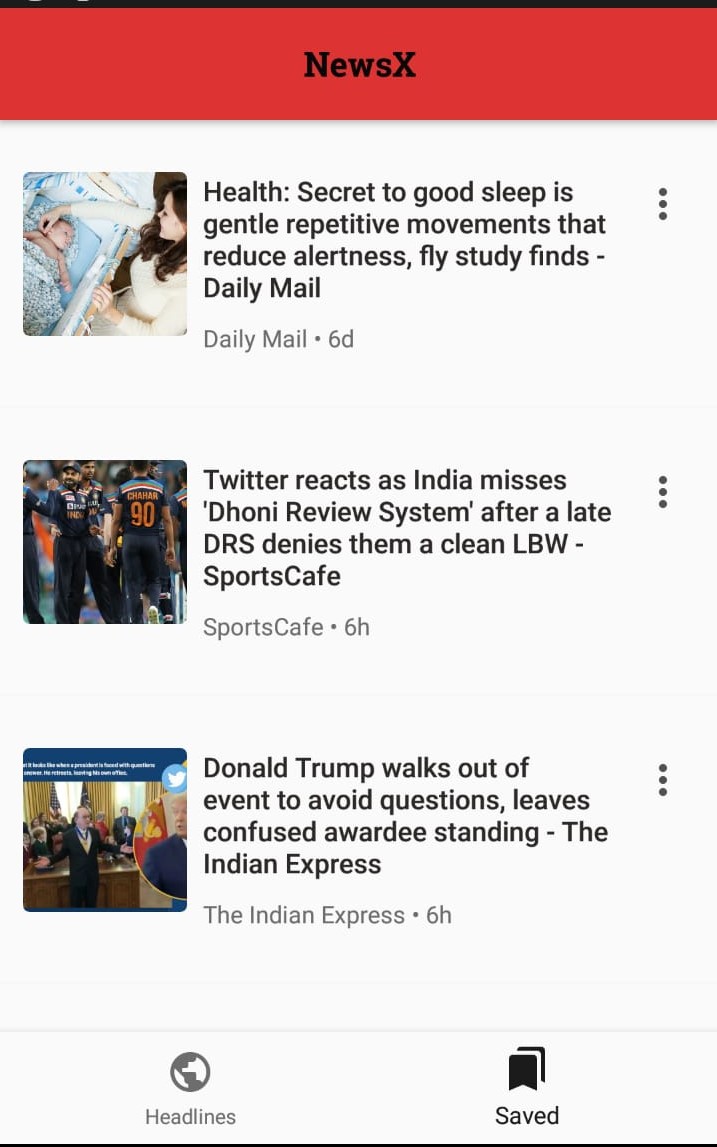
**2) Sharing ,Save for later and open in Browser options**



**3) The look of the news article after clicking on the news with save for later and share option**



**4) After saving the news**



**REFERENCES**

* Google
* YouTube
* developer.android.com