HOMEWORK 9: MOBILE APP DEVELOPMENT- ANDROID

Prof. Marco Papa

Developed and Designed by Yash Doshi

CSCI 571: Web Technologies
Homework 9: Android News App

Table of Contents

1. Objectives

- Become familiar with Java, JSON, Android Lifecycle and Android Studio for Android app development.
- Build a good-looking Android app.
- Learn the essentials of Google's Material design rules for designing Android apps
- Learn to use the Guardian, Google Trends and Open Weather APIs and Android SDK.
- Get familiar with third party libraries like Picasso, Glide and Volley.

The objective is to create an Android application as specified in the document below and in the video here.

2. Background

2.1 Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android application development, based on IntelliJ IDEA - a powerful Java IDE. On top of the capabilities you expect from IntelliJ, Android Studio offers:

- Flexible Gradle - based build system.
- Build variants and multiple apk file generation.
- Code templates to help you build common app features.
- Rich layout editor with support for drag and drop theme editing.
- Lint tools to catch performance, usability, version compatibility, and other problems.
- ProGuard and app-signing capabilities.
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine.

More information about Android Studio can be found at:
http://developer.android.com/tools/studio/index.html

2.2 Android

Android is a mobile operating system initially developed by Android Inc. a firm purchased by Google in 2005. Android is based upon a modified version of the Linux kernel. As of Nov 2018, Android was the number 1 mobile OS, in unit sales, surpassing iOS, while iOS was still the most profitable platform.

The Official Android home page is located at:
http://www.android.com/
The Official Android Developer home page is located at:
http://developer.android.com/

3. Prerequisites

This homework requires the use of the following components:

- Download and install Android Studio. Technically, you may use any IDE other than Android Studio such as Eclipse, but the latest SDKs may not be supported with Eclipse. We will not be providing any help on problems arising due to your choice of alternate IDEs.
- You must use the **emulator**. Everything should just work out of the box.
- If you are new to Android Development, Hints are going to be your best friends!

4. High Level Design

This homework is a mobile app version of Homework 8.

In this exercise, you will develop an Android application, which allows users to search for latest news, look at detailed information about them, bookmark those news and post on Twitter about them.

All API calls except the call in Home Tab and Trending Tab are same as in HW8. So, you can use the same Node.js backend as HW8 with these 2 calls added as new endpoints. In case you need to change something in Node, make sure you do not break your React assignment (or deploy a separate copy) as the grading will not be finished at least until 2 weeks later.
You must use Java strictly. Kotlin is not allowed.

PS: This app has been designed and implemented in a Pixel 2XL emulator. It is highly recommended that you use the same virtual device to ensure consistency.

Demo will be on an emulator, no personal devices allowed, see the rules:

https://csci571.com/courseinfo.html#homeworks

5. Implementation

5.1 App Icon and Splash Screen

In order to get the app icon/image, please see the hints.

The app begins with a welcome screen (Figure 2) which displays the icon provided in the hint above.
This screen is called Splash Screen and can be implemented using many different methods. The simplest is to create a resource file for launcher screen and adding it as a style to AppTheme.Launcher (see hints).

This image is also the app icon as shown in Figure 1

Figure 1: App Icon

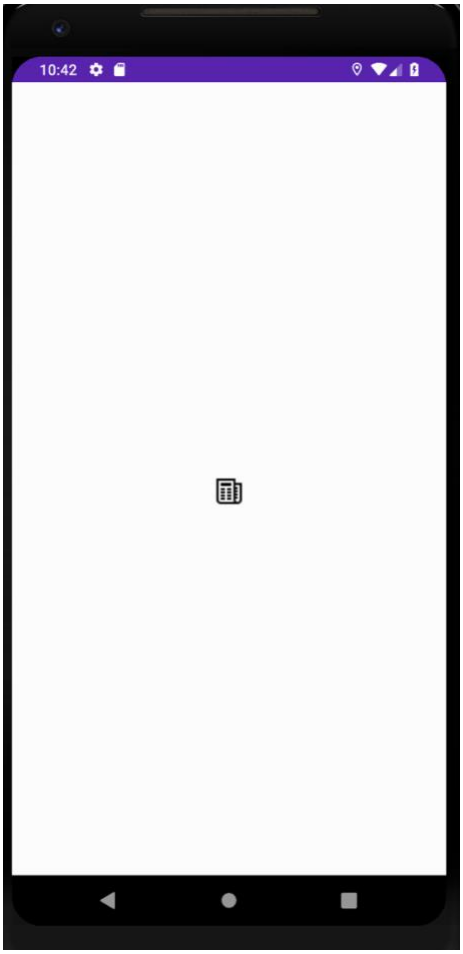

Figure 2: Splash Screen

5.2 Home Tab / Weather Summary Card

When you open the app, there is a Toolbar at the top with the search icon and a Navigation bar at the bottom. Both of these persists for all the 4 tabs.
As soon as you open the app, the weather summary card of the **current location of the user** is displayed and below the weather card there is a list of 10 **latest** news as shown in Figure 3.
The current location is to be fetched from the **emulator** itself, which will be used to call the **Openweathermap** API to get the weather data.
The list of news is to be fetched from the **Guardians** API.

Note: For getting the weather information, the Openweathermap API has be made directly from Android mobile client to the Web Service (Not from Node.js backend).

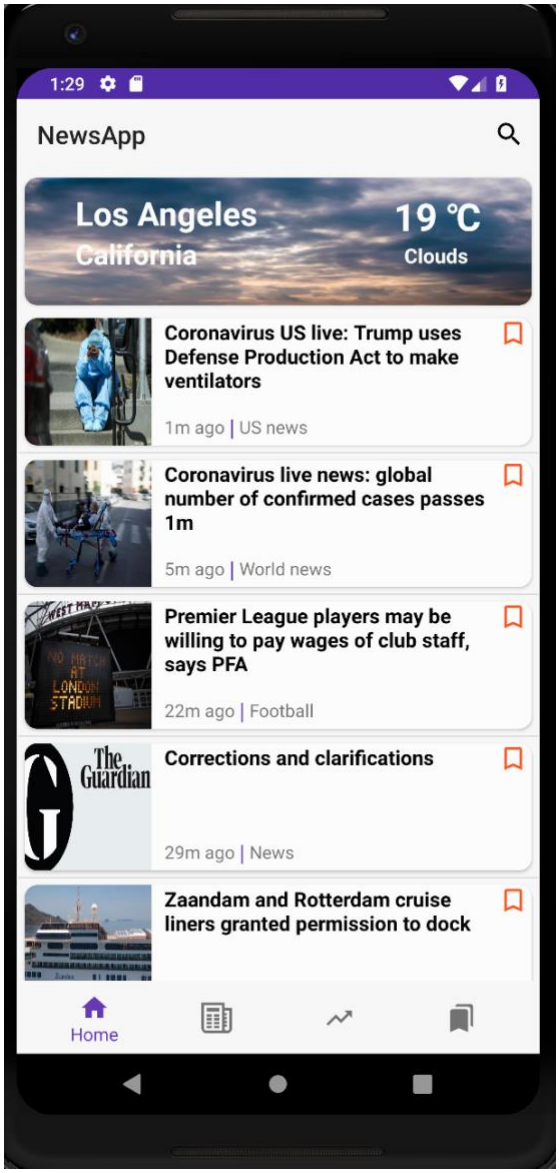

Figure 3: The Home page for the app

This tab has information which might be most useful for a user. It has the following information:

Weather Card:

To get the current location, get the city and state from the device emulator using Location Manager and Geocoder:

Please allow the app to access the device location. Check out section 6.21 for hints.

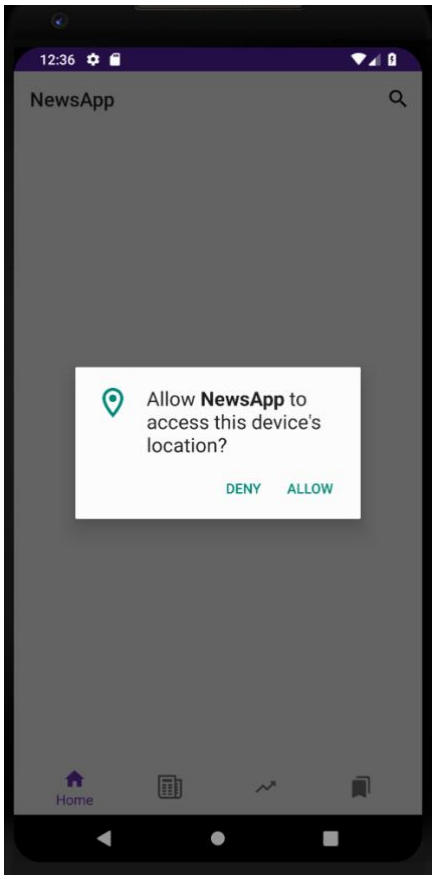

Figure 4: Request Permission from User

Once the device location is available, get the following properties:

- **City**: Get the value of city from the Geocoder.
- **State**: Get the value of state from the Geocoder.

To get the weather summary call the Openweathermap API (refer section 6.19 for registration steps) with the City parameter:
https://api.openweathermap.org/data/2.5/weather?q=[CITY]&units=metric&appid=[YOUR_APP_ID]

The response from the openweathermap API is shown in Figure 5:

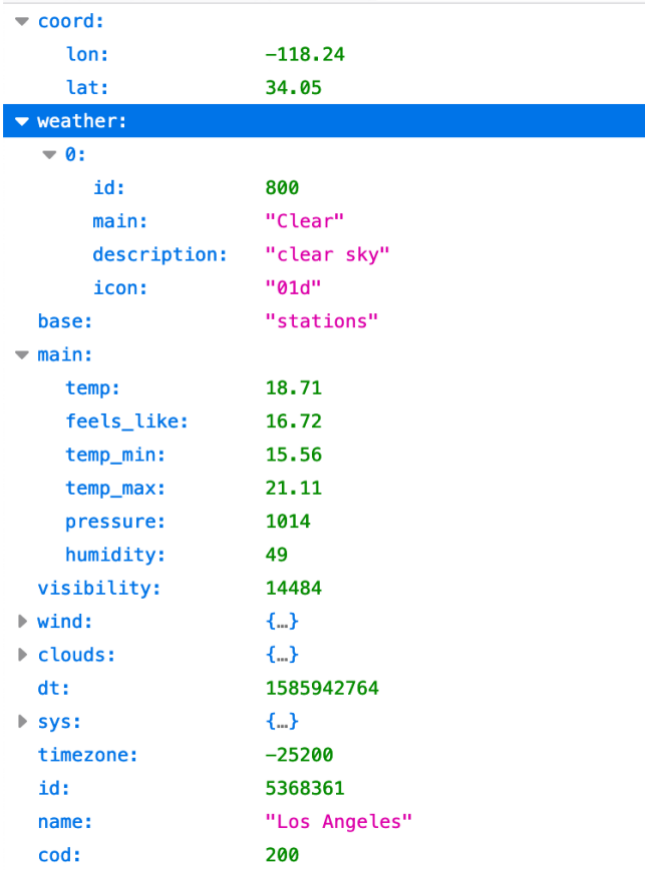

Figure 5: openweathermap API response

The following keys are to be used from the openweathermap API response:

- **Temperature**: **"temp"** property in **"main"** object. Make sure that the temperature is rounded off to nearest integer.
- **Summary**: **"main"** property in the **0th index** of the **"weather"** array.
- **Background Image**: This image is based on the Summary of the weather. The mapping of the image to the type is given in Table 1.

Summary Value	Image
Clouds	https://csci571.com/hw/hw9/images/android/cloudy_weather.jpg
Clear	https://csci571.com/hw/hw9/images/android/clear_weather.jpg
Snow	https://csci571.com/hw/hw9/images/android/snowy_weather.jpeg

Rain / Drizzle	https://csci571.com/hw/hw9/images/android/rainy_weather.jpg
Thunderstorm	https://csci571.com/hw/hw9/images/android/thunder_weather.jpg
Default (For all other values)	https://csci571.com/hw/hw9/images/android/sunny_weather.jpg

Table 1: Mapping Summary value to images

Latest News:

This provides a list of 10 latest news from the Guardians API to be displayed using a recycler view. To get the list of news call the Guardians API: https://content.guardianapis.com/search?order-by=newest&show-fields=starRating,headline,thumbnail,short-url&api-key=[YOUR_API_KEY]

The response from the Guardians API call is shown in Figure 6:

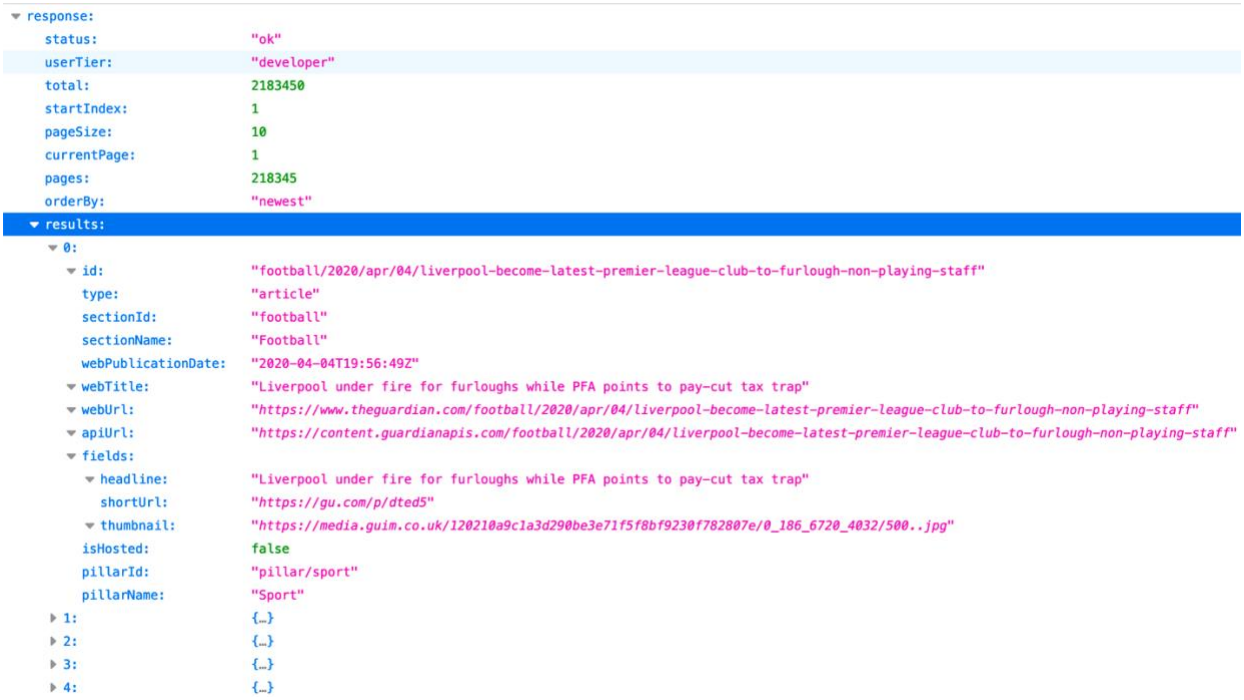

Figure 6: Guardian API response

Each News card has the following properties:

- **Image: "thumbnail"** property of the **"fields"** in the **"results"** array of the **"response"** json.
- **Title**: **"webTitle"** property in the **"results"** array of the **"response"** json. **Limit to 3 lines and show ellipsis.**
- **Time: "webPublicationDate"** property in the **"results"** array of the **"response"** json.
- **Section: "sectionName"** property in the **"results"** array of the **"response"** json.

- **Bookmark Icon:** On clicking the bookmark icon, that particular news will be bookmarked, and the bookmark icon will change. Similarly, if the news is already bookmarked, it will be removed from bookmarks and the bookmark icon will change back. The icon can be downloaded from the reference provided in section 6.1
- **Article ID: "id"** property in the **"results"** array of the **"response"** json. This will be used for the API call to fetch details of each article (Refer section 5.6)

For the **Time**, convert the **webPublicationDate** to a **ZonedDateTime** format of "**America/Los_Angeles"** and get the difference in time from the Current date and display them as:
"h ago" if the difference is greater than 1 hour. (e.g. 12h ago).
"m ago" if the difference is greater than 1 minute but less than 1 hour. (e.g. 12m ago).
"s ago" if the difference is less than a minute. (e.g. 12s ago)
Please refer section **6.6.**

On **long click** on a news card, a dialog will pop up with the image, title and the option to Bookmark and share it on Twitter as shown in Figure 7.

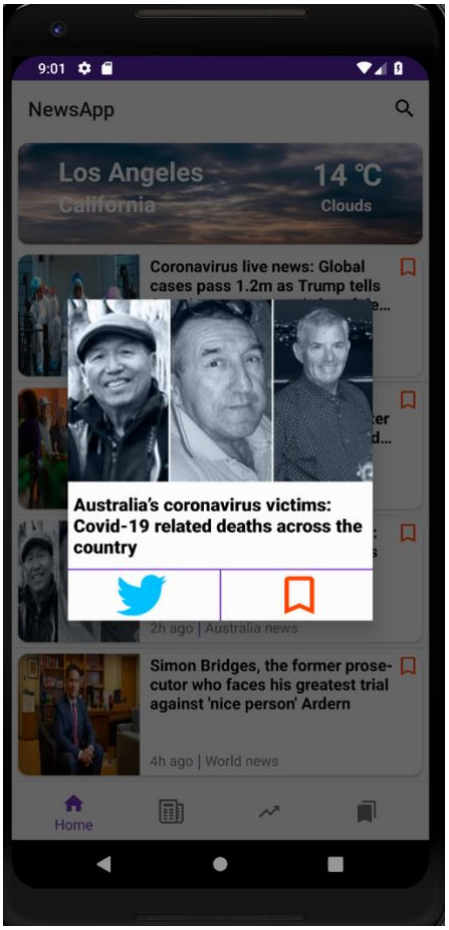

Figure 7: Dialog

Notes:

- **Every news card is clickable (image not required) to go to detailed news activity. Refer section 5.6.**
- **Every news card has a long click feature which will open a Dialog.**
- **If the word breaks in the Title before ellipsis, it is fine.**
- **Please refer Section 5.10 for the Tweet functionality.**
- **If any image is not available, use the default Guardians image provided in Homework 8.**

5.3 Headlines Tab

In this tab there are multiple fragments of different categories as shown in Figures 8, 9 and 10:

The API response for the below API Endpoints: **(same as HW8)**

- **World**: This fragment shows the latest news related to the World News.
  The API call to get data for this fragment:
  http://content.guardianapis.com/world?api-key=[YOUR_API_KEY]&show-blocks=all

- **Business**: This fragment shows the latest news related to the Business News.
  The API call to get data for this fragment:
  http://content.guardianapis.com/business?api-key=[YOUR_API_KEY]&show-blocks=all

- **Politics**: This fragment shows the latest news related to the Political News.
  The API call to get data for this fragment:
  http://content.guardianapis.com/politics?api-key=[YOUR_API_KEY]&show-blocks=all

- **Sports**: This fragment shows the latest news related to the Sport News.
  The API call to get data for this fragment:
  http://content.guardianapis.com/sport?api-key=[YOUR_API_KEY]&show-blocks=all

- **Technology**: This fragment shows the latest news related to the Technology News.
  The API call to get data for this fragment:
  http://content.guardianapis.com/technology?api-key=[YOUR_API_KEY]&show-blocks=all

- **Science**: This fragment shows the latest news related to the Science News.
  The API call to get data for this fragment:
  http://content.guardianapis.com/science?api-key=[YOUR_API_KEY]&show-blocks=all

The properties in the news card in these tabs are as follows:

- **Image: "file"** property of the **"assets"** array in the **"elements"** array in the **"main"** object in the "blocks" object in the **"results"** array of the **"response"** json.

- **Title**: **“webTitle”** property in the **“results”** array of the **“response”** json. **Limit to 3 lines and show ellipsis.**
- **Time: “webPublicationDate”** property in the **“results”** array of the **“response”** json.
- **Section: “sectionName”** property in the **“results”** array of the **“response”** json.
- **Bookmark Icon:** On clicking the bookmark icon, that particular news will be bookmarked, and the bookmark icon will change. Similarly, if the news is already bookmarked, it will be removed from bookmarks and the bookmark icon will change back. The icon can be downloaded from the reference provided in section 6.1
- **Article ID: “id”** property in the **“results”** array of the **“response”** json. This will be used for the API call to fetch details of each article (Refer section 5.6)

Notes:

- **All the News card in these tabs have the same properties as the Home Tab.**
- If you still have to change Node backend, make sure you don’t break the React app (if grading is still in progress) or deploy a newer instance of Node.
- **Every news card is clickable (image not required) to go to detailed news activity. Refer section 5.6.**
- **Every news card has a long click feature which will open a Dialog (Refer Figure 7).**
- **Please refer Section 5.10 for the Tweet functionality on long click of article.**
- **If any image is not available, use the default Guardians image provided in Homework 8.**

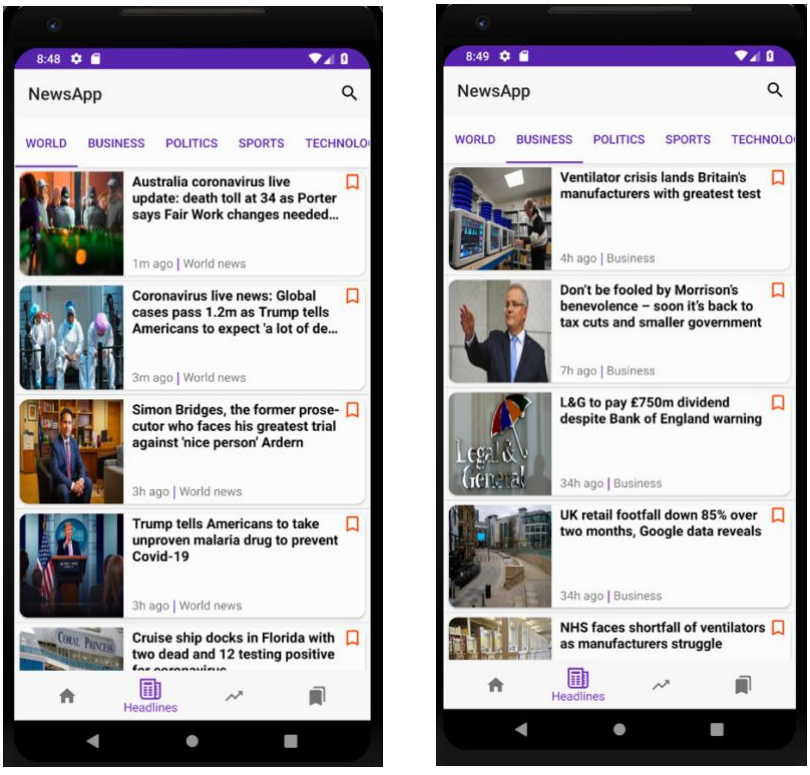

Figure 8: Word & Business Fragment

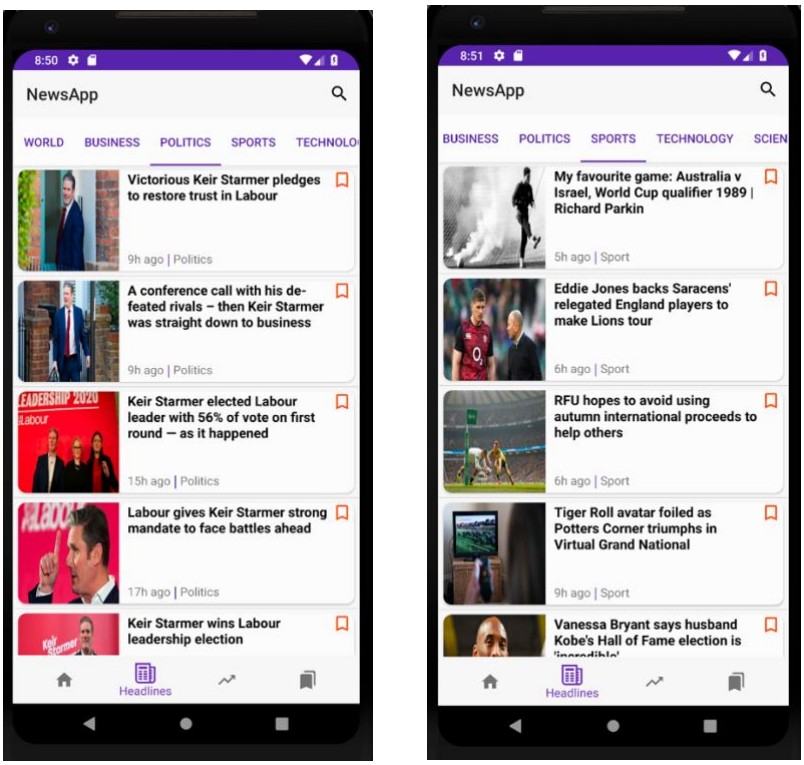

Figure 9: Politics & Sports Fragment

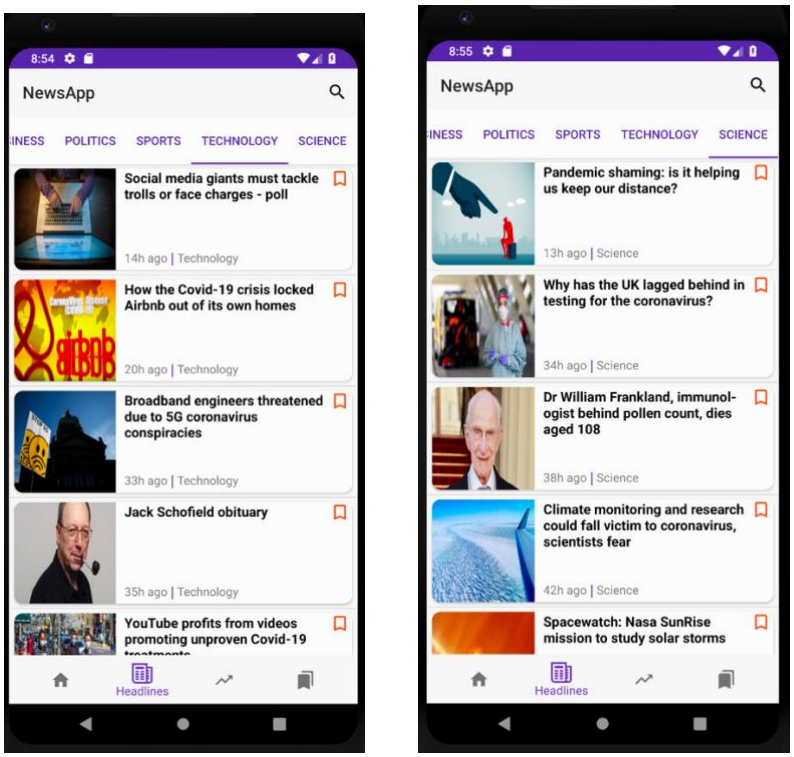

Figure 10: Technology & Science Fragment

5.4 Trending Tab

- This tab uses **MPAndroidChart** 3rd party library. You must add this as a dependency.
- This chart contains one LineChart as shown in Figure 11.
- To get the latest trending keyword over time, you have to use Google Trends API.

Google Trends API:

1. Please follow this link:
https://www.npmjs.com/package/google-trends-api#interestovertime
2. Please specify the startTime to be – 2019-06-01, when calling the Google trends API.
3. Extract all the **"value"** property from the "**timelineData**" array in the "**default**" object from the response (Refer the link provided above for a sample JSON response structure).
4. The **"value"** array extracted is to be set to the LineChart.
5. The LineChart accepts a List of type<**Entry**>. The <**Entry**> is a pair of (**x,y**) coordinate where x is the counter and y is the value.

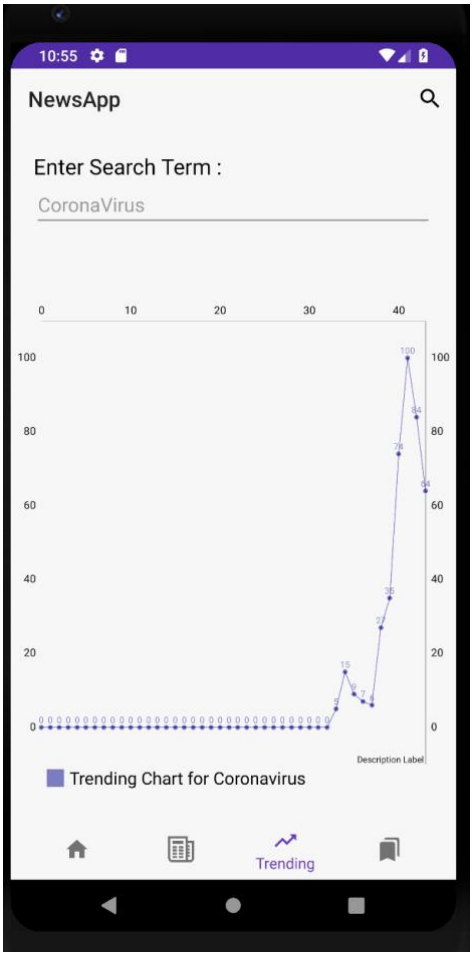

Figure 11: Default LineChart

When the user enters a new search term and presses the **actionSend** button or the **enter** key, the graph should render with the new search term. Refer the video for understanding.

Notes:

- **Make sure you change the Legend, font and line colors and remove the gridlines.**
- **There are some hints which should be enough to implement this graph view.**
- **Call the Google Trends API from Node.js backend.**
- **The default search term to render the graph is "Coronavirus". Please provide the Placeholder for it.**

5.5 Bookmark Tab

The idea is to give users an ability to add a news article to Bookmark. The bookmarked news **must** persist even after the user has closed the app. The Recycler view should be displayed using GridLayoutManager with CardLayout. Figure 12 demonstrates a sample bookmark page.

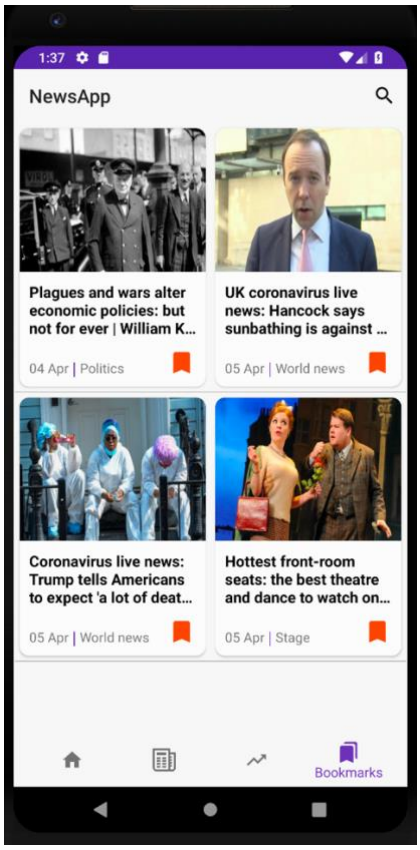

Figure 12: Bookmark Tab

Every news card is clickable **(Not the image)** to go to detailed news activity. Refer section 5.6. Limit the title in the bookmark tab articles to 3 lines with an ellipsis at the end. The **"sectionName"** on each card should be truncated if it overlaps with bookmark icon with an **ellipsis ('…')** at the end. If there is no overlap, there should be **NO** truncation. Long press on

each article opens a dialog as shown in Figure 7. The dialog displays the full title along with a twitter and bookmark button to remove the article from bookmarks. The entire bookmark tab must persist even when the application is closed and restarted.

5.5.1 Adding to Bookmark

The user can add a news article to bookmark from the **Home Tab**, **Headlines Tab** and also when the user **searches** from a particular keyword. The news can be also bookmarked from the **dialog on long press**.

5.5.2 Removing from Bookmark

The user can remove a news article from the **Home Tab**, **Headlines Tab**, **Search Activity** and also from the **Bookmark tab**. The news can be also removed from bookmark using the **dialog on long press**.

Figure 13 and 14 show the bookmark feature along with the Toast messages described in the notes below.

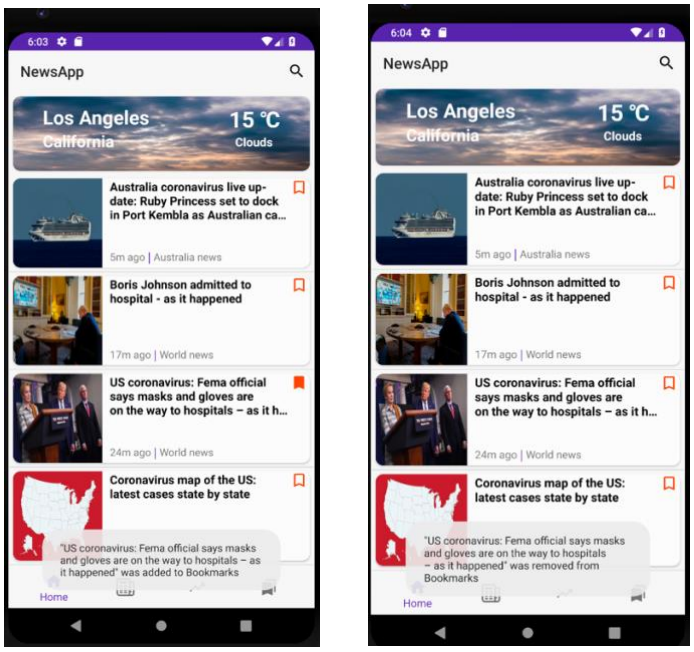

Figure 13: Toast messages on adding/removing a news article from bookmark in the Home Tab (to be replicated in the Headlines tab, Bookmark tab and Search Activity)

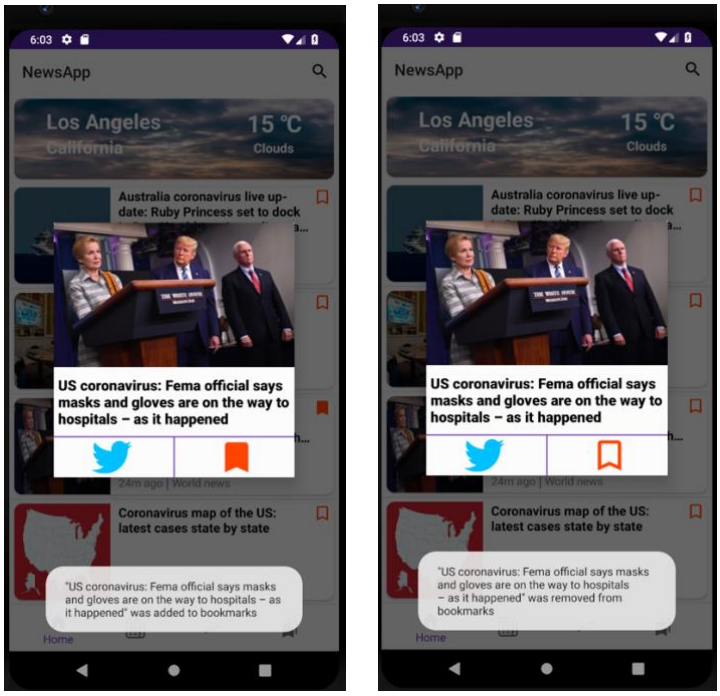

Figure 14: Toast messages on adding/removing a news article from bookmark from the dialog

5.5.3 Empty Bookmark Tab

Figure 15 shows the bookmark tab without any article saved.

Figure 15: No Bookmarked Articles

Notes:

- Display "**"[NEWS TITLE]"** was added/removed to/from bookmarks" using a Toast anytime any article is either added or removed from bookmarks anywhere throughout the app. Hints provided for Toasts. Note the quotes around the title.
- Make sure that the add/remove bookmark icon toggles correctly.
- The bookmarked news must persist across sessions i.e. Opening and closing should retain the view. You can use SharedPreferences to do this like LocalStorage in hw8.
- The date for each article in the **Bookmark tab** is to be formatted in **"dd MMM"** to match the example given above as in Figure 12.
- The bottom line in Figure 12 should always span only 1 line.

5.6 Detailed Article Tab

On clicking the articles from any of the Home Tab, Headlines Tab, Search Results or the Bookmark tab, the detailed view of an article should open up as shown in Figure 16.

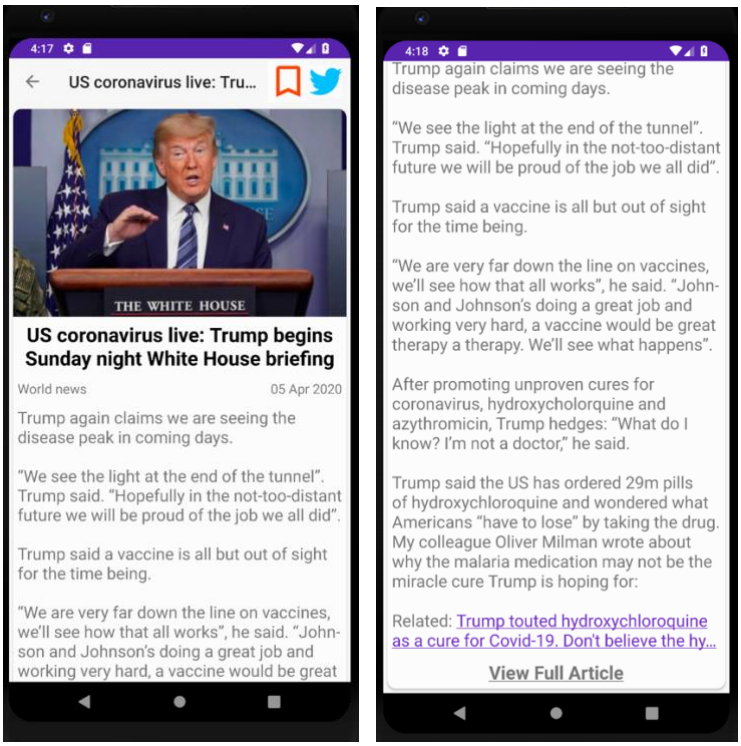

Figure 16: Detailed Article Page

To get the details of the article, call the following API: **(Same as HW8)**
https://content.guardianapis.com/[ARTICLE_ID]?api-key=[YOUR_API_KEY]&show-blocks=all

The **ARTICLE_ID** is fetched from the JSON response shown in **Figure 6** by accessing the **“id”** property in the **“results”** array of the **“response”** json.

The response from the API is shown in Figure 17:

Figure 17: Detailed Article Response JSON

Each detailed article page (shown in Figure 16) has the following properties:

- **Image: “file”** property of the **“assets”** array in the **“elements”** array in the **“main”** object in the “blocks” object in the **“results”** array of the **“response”** json.
- **Title**: **“webTitle”** property in the **“content”** object of the **“response”** json.
- **Date: “webPublicationDate”** property in the **“content”** object of the **“response”** json.
- **Section: “sectionName”** property in the **“content”** object of the **“response”** json.
- **Description:** Append every **“bodyHtml”** property in the **“body”** array of the **“blocks”** object in the **“content”** object of the **“response”** json.
- **Article URL (for View Full Article at the bottom): “webUrl”** property in the **“content”** object of the **“response”** json.

For the **Date**, convert the **webPublicationDate** to a **ZonedDateTime** of “**America/Los_Angeles”** in the format **“dd MMM yyyy”**.

The toolbar in the detailed article page has the following properties:

- **Back Button:** The back button takes the user back to the location where the article was clicked on. See hints for help.
- **Title: “webTitle”** property in the **“content”** object of the **“response”** json.
- **Bookmark Icon:** On clicking the bookmark icon, that particular news will be bookmarked, and the bookmark icon will change. Similarly, if the news is already bookmarked, it will be

removed from bookmarks and the bookmark icon will change back. The icon can be downloaded from the reference provided in section 6.1.

- **Twitter Share Button:** Refer section 5.10 for twitter button functionality

Notes:

- **Limit the Description to max 30 lines and add an ellipsis at the end. If the word breaks it is fine.**
- **If any image is not available, use the default Guardian image provided in HW 8.**
- **The "View Full Article" is clickable and redirects to the original article on Guardian news website.**

5.7 Search Functionality

- On top right side, there will be a search button which opens a textbox where the user can enter keyword to search for news. See hints for icon.
- The user is provided with suggestions of keywords using the Bing Autosuggest API.
- When the user taps on a suggestion, it is filled inside the search box and clicking enter/next takes the user to the next page.
- Before you get the data from your backend server, a progress bar should display on the screen as indicated in Figure 22.
- On the next page, the user will then be redirected to a new page/activity which will show the recycler view with the list of news related to the keyword.
- This flow is demonstrated in Figure 18.

This component involves 2 things:

- Implementing a searchable – see hints
- Implementing autocomplete – see hints

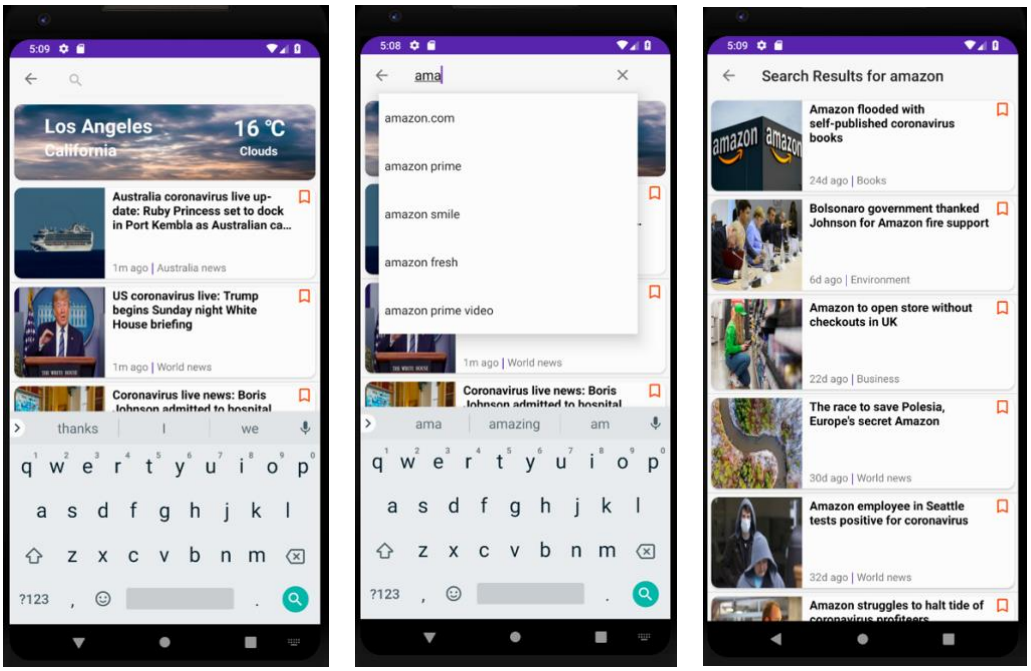

Figure 18: Flow for searching using a keyword

For the autosuggest API, call the following API directly from your Android App (No need to use Node.js):
https://api.cognitive.microsoft.com/bing/v7.0/suggestions?q=[KEYWORD]

Similar to that in Homework 8, you will have to pass the **'Ocp-Apim-Subscription-Key'** to make the API request. The response from the API is shown in Figure 19:


```
{
"_type": "Suggestions",
"queryContext": {
    "originalQuery": "ama"
},
"suggestionGroups": [
    {
        "name": "Web",
        "searchSuggestions": [
            {
                "url": "https://www.bing.com/search?q=amazon.com&FORM=USBAPI",
                "displayText": "amazon.com",
                "query": "amazon.com",
                "searchKind": "WebSearch"
            },
            {
                "url": "https://www.bing.com/search?q=amazon+prime&FORM=USBAPI",
                "displayText": "amazon prime",
                "query": "amazon prime",
                "searchKind": "WebSearch"
            },
            {
```


Figure 19: Bing Autosuggest API Response

For the suggestions, use the **“displayText”** property of the **“searchSuggestions”** array in the **0th index** object of the **“suggestionGroups”** array. Restrict your suggestions to **5** values.

To fetch the articles based on keyword, call the following API **(same as HW8)**:

https://content.guardianapis.com/search?q=[KEYWORD]&api-key=[YOUR_API_KEY]&show-blocks=all

The response from the API is shown in Figure 20:

Figure 20: JSON Response for Search using keyword

Notes:

- **All the News card in these tabs have the same properties as the Headlines Tab.**
- On **long click** on a news card, a dialog will pop up with the image, title and the option to Bookmark and share it on Twitter as shown in Figure 7.

- If you still have to change Node backend, make sure you don't break the React app (if grading is still in progress) or deploy a newer instance of Node.
- **Reusing the recycler view and the corresponding logic to dynamically set the values will make it much faster to implement this.**
- Do not forget the static **"Search Results for [KEYWORD]"** label on top.
- **Back Button:** The back button in the toolbar takes the user back to the location where the article was clicked on. See hints for help.
- In the Autosuggest, only make an API call to the Bing Autosuggest after the user enters **3 characters**. **Example:** "am" should not display any suggestions but "ama" should have suggestions. Refer video for better understanding.

5.8 Swipe Refresh

The **Home tab**, **Headlines tab** and **Search Results** page are the three places where pull to refresh functionality is to be implemented as shown in Figure 21:

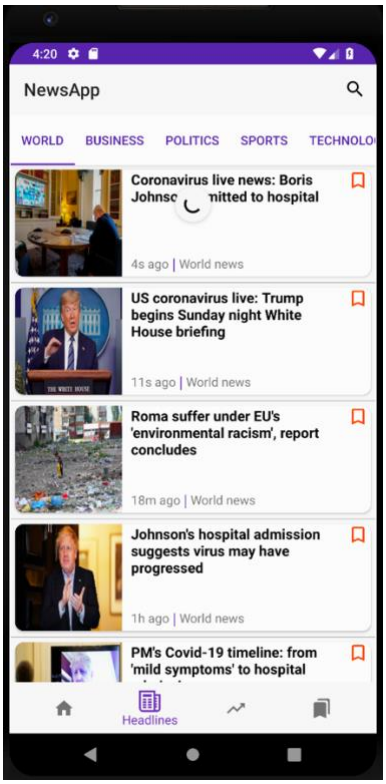

Figure 21: Pull to refresh

The idea is to **re-fetch articles** and update the list of news in the Recycler View on the current tab whenever the pull to refresh is used. Refer video for a better understanding. [See hints]

5.9 Progress bar

Every time the user has to wait before he can see the data, you must display a progress bar as shown in Figure 22. The progress bar is to be present across **Home Tab**, **Headlines Tab**, **Search Page** navigation and **Detailed Article** navigation and just says “Fetching News”. One idea would be to display the progress bar by default (where needed) and then hide it as soon as the data is received/view is prepared.
See hints for progress bar related ideas and styling.

Note:
Based on your implementation, you might need to put progress bars on different places. During demo, there should NOT be any screen with default/dummy/placeholder values or an empty screen.

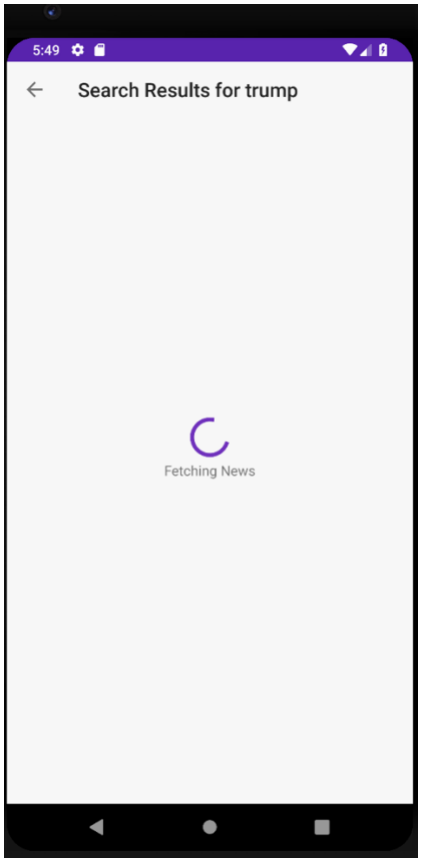

Figure 22: Progress bar

5.10 Twitter Button

From every twitter button, on clicking the button, the article should be shared by opening a browser with Twitter Intent. Use the following link to implement the Twitter Intent:
https://developer.twitter.com/en/docs/twitter-for-websites/tweet-button/guides/web-intent

The icon is provided in section 6.1.

5.11 Summary of detailing and error handling

1. Make sure there are no conditions under which the app crashes
2. Make sure all icons and texts are correctly positioned/centered as in the video/screenshots
3. Make sure the LineChart legends and axes are correctly displayed.
4. Make sure there is a progress bar every time there is nothing to show
5. The bookmark button works as expected.
6. All API calls except Bing Autosuggest and Openweathermap are to be made using Node.js backend.

5.12 Additional Info

For things not specified in the document, grading guideline, or the video, you can make your own decisions. But keep in mind about the following points:

- Always display a proper message and don't crash if an error happens.
- You can only make HTTP requests to your backend Node.js on AWS/GAE/Azure and use the Google Map SDK for Android.
- All HTTP requests should be asynchronous and should not block the main UI thread. You can use third party libraries like Volley to achieve this in a simple manner.

6. Implementation Hints

6.1 Icons

6.1.1 Icons

The images used in this homework are available in the table below.
You can choose to work with xml/png/jpg versions. We recommend using xml as it is easy to modify colors by setting the Fill Colors.

Icon Name	**Usage**
App Icon/ Splash Screen/ Headlines Tab Icon	https://csci571.com/hw/hw9/images/android/icon_news.png
Home Tab Icon	https://material.io/resources/icons/?search=home&icon=trending_up&style=baseline
Trending Tab Icon	https://material.io/resources/icons/?search=trending_up&icon=trending_up&style=baseline
Bookmark Icon	https://material.io/resources/icons/?search=bookmark&icon=trending_up&style=baseline
Search Icon	https://material.io/resources/icons/?search=search&icon=bookmark&style=baseline

Guardian News Default App	https://assets.guim.co.uk/images/eada8aa27c12fe2d5afa3a89d3fbae0d/fallback-logo.png
twitter	https://csci571.com/hw/hw9/images/android/bluetwitter.png

6.3 Third party libraries

Sometimes using 3rd party libraries can make your implementation much easier and quicker. Some libraries you may have to use are:

6.3.1 Volley HTTP requests

Volley can be helpful with asynchronously http request to load data. You can also use Volley network ImageView to load photos in Google tab. You can learn more about them here:

https://developer.android.com/training/volley/index.html

6.3.2 Picasso

Picasso is a powerful image downloading and caching library for Android.

http://square.github.io/picasso/

If you decide to use RecycleView to display the photos with Picasso Please use version 2.5.2 since latest version does not support RecycleView well.
https://github.com/codepath/android_guides/wiki/Displaying-Images-with-the-Picasso-Library

6.3.3 Glide

Glide is also powerful image downloading and caching library for Android. It is similar to Picasso. You can also use Glide to load photos in Google tab.

https://bumptech.github.io/glide/

6.3.4 MPAndroidChart

In order to create graphs we will use the MPAndroidChart library. Full code and documentation can be found here -https://github.com/PhilJay/MPAndroidChart

For our purposes following these links should suffice:
Creating a LineChart: https://weeklycoding.com/mpandroidchart-documentation/setting-data/
Styling the Legend: https://github.com/PhilJay/MPAndroidChart/wiki/Legend
Styling the label: https://stackoverflow.com/questions/28632489/mpandroidchart-how-to-set-label-color

6.5 Working with action bars and menus

https://developer.android.com/training/appbar/setting-up
https://stackoverflow.com/questions/38195522/what-is-oncreateoptionsmenumenu-menu

6.6 Converting number to date in Java

https://howtodoinjava.com/java/date-time/localdatetime-to-zoneddatetime/

6.7 Displaying Progress Bars

https://stackoverflow.com/a/28561589
https://stackoverflow.com/questions/5337613/how-to-change-color-in-circular-progress-bar

6.8 SearchBar and AutoCompleteTextView

To implement the search functionality, these pages will help:
https://www.youtube.com/watch?v=9OWmnYPX1uc
https://developer.android.com/guide/topics/search/search-dialog

Working with the AutoCompleteTextView to show the suggestions might be a little challenging. This tutorial goes over how it is done so that you get an idea of how to go about it.
https://www.truiton.com/2018/06/android-autocompletetextview-suggestions-from-webservice-call/

In order to link your Search Bar with autocomplete suggestions, these links might help:
https://www.dev2qa.com/android-actionbar-searchview-autocomplete-example/

6.9 Implementing Splash Screen

There are many ways to implement a splash screen. This blog highlights almost all of them with examples:
https://android.jlelse.eu/the-complete-android-splash-screen-guide-c7db82bce565

6.10 Adding the App Icon

https://dev.to/sfarias051/how-to-create-adaptive-icons-for-android-using-android-studio-459h

6.11 Adding ellipsis to long strings

https://stackoverflow.com/questions/6393487/how-can-i-show-ellipses-on-my-textview-if-it-is-greater-than-the-1-line

6.12 Adding tabs in Android

https://www.spaceotechnologies.com/create-multiple-tabs-using-android-tab-layout/

6.13 Adding a button to ActionBar

https://developer.android.com/training/appbar/actions
https://stackoverflow.com/questions/12070744/add-back-button-to-action-bar

6.14 Implementing a Recycler view in android

https://www.learningsomethingnew.com/how-to-use-a-recycler-view-to-show-images-from-storage
https://developer.android.com/guide/topics/ui/layout/recyclerview

6.15 Adding Toasts

https://stackoverflow.com/questions/3500197/how-to-display-toast-in-android

6.16 To implement pull to refresh feature

https://medium.com/@houdayer.corentin/how-to-use-swipe-to-refresh-swiperefreshlayout-99abd674c907

6.17 To implement bookmark

https://developer.android.com/reference/android/content/SharedPreferences
https://www.journaldev.com/9412/android-shared-preferences-example-tutorial

6.18 Passing variables to intent

https://stackoverflow.com/questions/2405120/how-to-start-an-intent-by-passing-some-parameters-to-it

6.19 Openweathermap API Registration

To get the API key (appid):

- Sign up and Log in to https://home.openweathermap.org/
- In the pop-up that asks for Company (Optional) and Purpose – select purpose as Mobile apps developments
- Visit the API keys tab to get your API key
- **NOTE:** There is a wait time of approximately 2 hours before your API key activates and can be used

6.20 Implementing Dialog

https://mkyong.com/android/android-custom-dialog-example/
https://androidexample.com/Custom_Dialog_-_Android_Example/index.php?view=article_discription&aid=88&aaid=111

6.21 User Location using Emulator

Use the following emulator settings to get the Current location of the device.

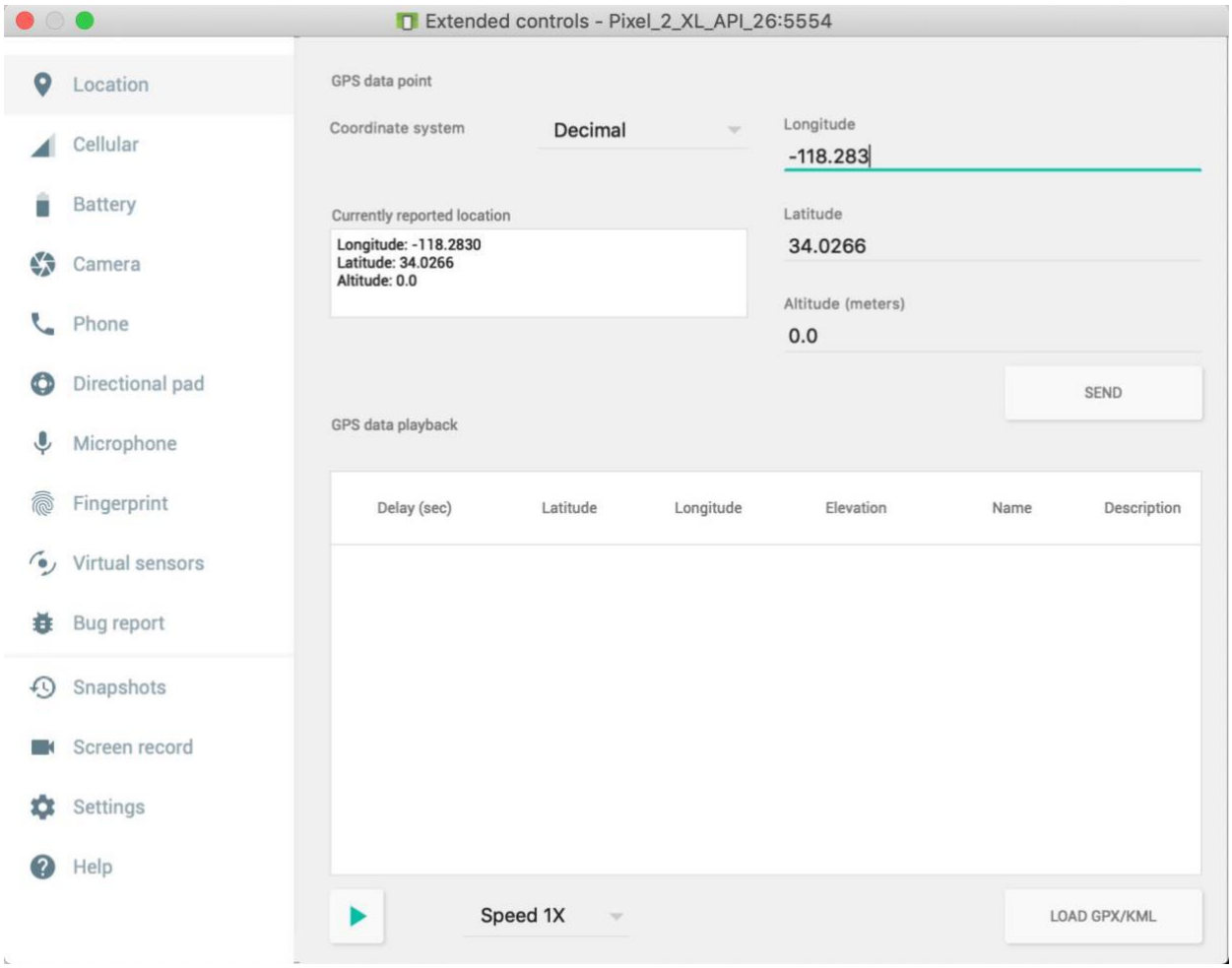

Figure 23: Location setting of the Emulator

https://stackoverflow.com/questions/40142331/how-to-request-location-permission-at-runtime
https://stackoverflow.com/questions/22323974/how-to-get-city-name-by-latitude-longitude-in-android

7. What to Upload to GitHub Classroom

You should also ZIP all your source code (the java/ and res/ directories excluding the vector drawables that were downloaded) and submit the resulting ZIP file by the end of the demo day.

Unlike other exercises, you will have to demo your submission using **Zoom Remote Control** during a special grading session. Details and logistics for the demo will be provided in class, on the Announcement page and on Piazza. **Demo is done a laptop/ notebook/MacBook or Windows PC using the emulator, and not a physical mobile device.**

****IMPORTANT****

All videos are part of the homework description. All discussions and explanations on Piazza related to this homework are part of the homework description and will be accounted into grading. So please review all Piazza threads before finishing the assignment.