Assignment 3: New York Times

Due Tuesday, October 16th, at 11:59 PM.

Overview

For this assignment you will be building an article browser for New York Times articles. You want something simple that will provide users with a straight-forward browsing experience. You believe that the key to a successful (lightweight) NYT article browser lies on three main principles: users should be able to browse top stories by category, they should be able to search for content across the entire NYT database, and they should be able to see just the relevant snippets of an article at first glance. To accomplish this, you will be tapping into the NYT API and populating articles in a `List`. On top of that you will be building search and category-browsing functionality using a `TextInput` box.

your final product will look something like this:

Design Details

You've seen the screenshots above. You probably already started thinking about how to layout most of the particular components. For this assignment, you don't have to worry about mimicking the exact design — play around with sizing, margins, spacing, colors, and whatever you see fit. Get creative. However, your app must contain all of the components that are pictured (more details below), and should look polished regardless of the device and platform that it is running on.

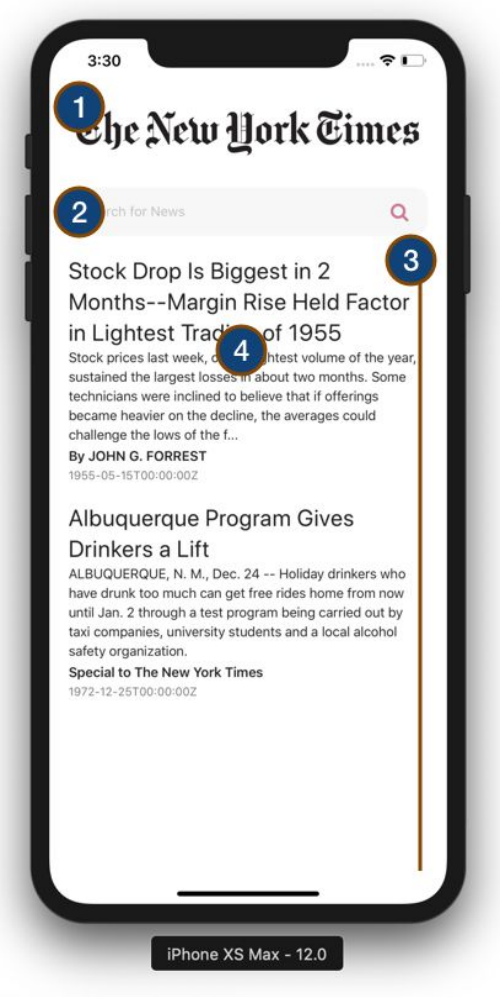
The concepts you applied in assignment 2 will be very helpful when creating this design. *However, don't spend too much time on design.* We know you can do it. Here's a breakdown of what we want to see:

Feature	Description
Visual Requirements	• The New York Times logo must take up the width of the screen and look reasonably sized on multiple devices. • Search bar must have a search icon right next to it. This icon should be a vector icon (this will actually make your life easier).
Search Bar	• The search bar should be made up of a `TextInput` and a `Button` (you can also use a pressable vector icon). • When the button is pressed, an API request will be triggered to search for articles that contain the text on the input box. • Keyboard should be dismissed when clicking outside of the input box.
Article List	• You must implement an article viewing list using either a `FlatList` or a `SectionList`. • For your render function, you should create `Text` components for the following article items: *title of the article, its snippet summary, its author, and the date at which it was posted.* You can choose to cover different elements of an article, as long as you cover at least four. • The title should be a bigger font then the other categories. See the screenshot for styling suggestions. To achieve this, you can use this Typography styling library (recommended; covered in class) or create your own styles.

Loading Indicator	● When articles are being downloaded, you must use an `ActivityIndicator` to show that something is loading. ● You will have to conditionally render items to achieve this. If an API request is being fetched, render the indicator in place of a list. Once the request returns a value, render the list in place of the indicator.
Tap to View	● Tapping on an article should take the user to the website URL of the article itself. See the implementation details section below if you need help achieving this.

Implementation Details

Here are some additional details that will help with planning. For context, here's a diagram of the relevant features outlined above:



(1) The title; NYT logo
(2) Search Bar
(3) `FlatList` or `SectionList`
(4) The article's content; the product of the list's render function.

1. The title; NYT Logo

This is just an image. You will find it in your starter files. This `Image` must take up the width of the screen. To calculate the screen width you can use the `Dimensions` API. You can also access the screen width directly from the `Metrics` file that we've provided in past live demos.

2. Search Bar

The `TextInput` component has a property called `onChangeText` that you can use to retrieve the text of the input box as you are typing on it. We encourage you to look at each Component's API in order to find other cool properties you can use. For instance, `TextInput` also has a property called `onSubmitEditing` that runs a function whenever you press the "submit," or "done," button in your keyboard. This will let you run search queries instantly. After you run a search query, clear the input box.

One thing that you can do to dismiss the keyboard, depending on your use case, is use the `Keyboard` API from the `react-native` package:

```
import { Keyboard } from 'react-native';
```

You will then be able to call the function `Keyboard.dismiss();`

To make queries using the NYT API, you will need to track what the text that is entered in your `TextInput` is.

3. `FlatList` or `SectionList`

   Your list implementation should be consistent with what we learned in class. You will also be considering how to render an `ActivityIndicator` instead of a `List` when elements are loading.

   There's multiple ways to do conditional rendering. We've covered some in class. Here's an interesting approach based on a function that you haven't seen yet:

```
getArticleContent = () => {
  const {articles, loading} = this.state;

  let contentDisplated = null;

  if (loading) {
    contentDisplated = <ActivityIndicator
                            style={styles.activityIndicator}
                            size="large" color="black"/>;
  } else {
    contentDisplated = <News articles={articles}/>;
  }

  return (
    <View style={{flex: 1}}>
      {contentDisplated}
    </View>
  )
}
```

```
...   (later in the return value of the main render function)
```

```
{this.getArticleContent()}
```

   This approach specifically creates an object which references the JSX tags. This is syntactically allowed. We invite you to explore.

   If you implement "swipe down to refresh," you might need to render both elements at once in some circumstances.

4. **The article's content; the product of the list's render function.**
   You've seen how to render `Text` tags in a render function and how to use the Typography library. In order to implement the "tap to view" functionality, take a look at the `Linking` component from the package `react-native`. Articles have a `url` property that you can use with `Linking`.

5. **(bonus) Making Network Requests**
   We've included some code for you to handle network requests with the NYT API. Checkout the `APIRequest` file. You can modify this however you want. The result of all queries is saved to the component's `State`.

Grading

Grading is based on the specifications above. Like last time, we will be rewarding bonus points for novel solutions and extensions.

Visual Requirements	2 points total
Search Bar	3 points
Article List (with tap to view)	4 points
Loading Indicator	1 point
Total	**10 points**

Extensions

Give these a shot! Try at least one. You might love what you end up with.

- Implement swipe down to refresh on your `List`! (+1 point)
- Some articles contain images. Pull these and render them in your list. You might have to conditionally render them since not all articles will contain an image (+2 points).
- Implement an additional third party library that modifies a component visually. You can use any compatible library from npm! (+2 point)
- Add more ways to filter articles. You can use any new components that you want. To achieve this, you might want to take a look at the NYT API in-depth: https://developer.nytimes.com/ (+3 points)
- Make your `FlatList` rows swipeable (see https://github.com/jemise111/react-native-swipe-list-view). Bonus points if you add a

"delete" icon that removes the given article from the `State`. It's okay if the article gets added again on the next refresh. Later on we'll learn how to persist this. (+3 points)

Resources

You can optional starter files for this project in the GitHub repository here: https://github.com/CS47-Stanford/Assignment3-Starter

Git, or source control in general, is out of the scope of this class. You will be able to access the entire starter project by pressing the "Clone or download" button and then clicking on "Download ZIP."

Submission

When you have completed your assignment, please remove the `node_modules` folder and package the entire project that you worked on in a .zip file. Please email this file to reactnative@cs.stanford.edu with the subject line CS47SI Assignment 3 Submission (SUNetID). Please include your actual SUNetID (from your email, not the number).

NOTE!!

- While working on the assignment, you will make a lot of requests to the NYT API. Since this is a free API, it will limit the number of calls per hour and per day that you can make, which could make the development process not work as expected. If the API stops working, please
    - Head to https://developer.nytimes.com/signup, and create a new key for "Article Search API".
    - Feel free to put "cs47.stanford.edu" as the website.
    - An API key will be emailed to you soon after.
    - Copy the new key over to `App/Config/AppConfig.js`, and replace the current `articleSearch` value with your new key.