Picnews - DASS Project

Team Number - 15

Team Members:

Trinadh - 2019101043

Varul - 2019111015

Shivanshu - 2019101001

Srijith - 2019114002

About

Our project aims at showing news to its users via a mobile/web application. It is less dependent on text and leans towards a higher use of image/visual content, to provide the user with a better and smoother experience.This is incredibly helpful as the average user’s attention span is far less than what is required to read an entire news article. The project also makes use of Machine Learning Algorithms to predict the user’s preferred news-category,their favoured language,their user interests etc. and provide customized news feeds. The app provides content from a wide variety of publishers like Bloomberg, Reddit, NY Times, Hindustan Times, Times of India etc. and the content can also be availed in multiple different languages as per the user’s demand.

Backend

The backend in our project is comprised of 2 main files:

* index.js -

const functions = require("firebase-functions");

const admin = require("firebase-admin")

const NewsAPI = require("newsapi");

const newsapi = new NewsAPI("KEY\_HERE");

This code snippet is used to obtain all of the libraries that are required for the file to perform its required function and store them in variables as per our need. The libraries are obtained from 2 sources, namely ‘Firebase’ and ‘NewsAPI’. The last part is used to fetch news from the API and requires entering a valid key obtained from the NewsAPI website.

exports.fetchData = functions.pubsub.schedule("55 23 \* \* \*").timeZone("Asia/Kolkata").onRun((*context*)=>{

The ‘exports’ variable is used along with the function ‘fetchData’ in order to get the data from the NewsAPI for our service. The data is obtained by using the function ‘pubsub’ from the library obtained by us in the variable ‘functions’ at the beginning. It is further given the arguments of ‘schedule’ which gives the pages to be obtained and the articles per page to be obtained. Another argument is ‘timeZone’ which is given to get news belonging to that area and at that time. The last argument is ‘onRun’ which when provided with the input ‘context’ displays the articles obtained from the api.

const tags = ["sports",

"politics",

"entertainment",

"technology",

"business"];

The snippet describes the different categories of the articles that will be made available to the user by us. The number of these can be changed as per the service provider’s need and the demand that arises henceforth.

for (let x=0; x<tags.length; x++) {

newsapi.v2.topHeadlines({

category: tags[x],

country: "in",

pageSize: 100,

}).then((*response*)=>{

if (*response*.status == "ok") {

*// add to db here*

console.log("Fetched articles on Category ",tags[i]," number of articles : ",*response*.articles.length);

*// let batch = db.batch();*

} else {

console.log("Some error", JSON.stringify(*response*));

}

return true;

}).catch((*error*)=>{

console.log("Error", *error*);

});

}

The snippet above is a ‘for’ loop. The number of iterations in it are equal to the total number of articles available to us. For displaying the headlines, we take the articles available to us and then process them by their category, the country and decide the page size for the service. The ‘if-else’ condition next is used to make sure that the articles are fetched correctly and the process of segregating them into different categories is completed accurately. In case it is not, a message is displayed on the console that informs the dev about the same.

The code is finished working.

* searchQuery.js -

const functions = require("firebase-functions");

const admin = require("firebase-admin")

const NewsAPI = require("newsapi");

const newsapi = new NewsAPI("KEY\_HERE"); *// ADD valid key here*

Just like above, here too we import the required libraries and store them in variables as per our need.

admin.initializeApp()

const db = admin.firestore()

The above snippet contains 2 lines. The 1st one is used to initialize the application in firebase so that it can be readied for further processing. The next line is used to start the database that we use in firestore so that the articles obtained can be stored in it.

exports.searchQuery = functions.https.onRequest((*request*,*response*)=>{

The snippet requires a ‘request’ from the user so that the app can ‘respond’ appropriately. All this is done by using the variable ‘exports’ and then using an argument ‘searchQuery’ with it. This is carried out with the help of the library imported earlier named ‘functions’, then giving it an argument ‘onRequest’ that has its input in the form of a ‘request’ and the output is provided in the form of the ‘response’.

*/\*\**

*\* REQUEST FORMAT*

*\* {*

*\* query : String query*

*\* country : all for no filter, country code otherwise*

*\* language : all for no filter, name of language otherwise*

*\* category : all for no filter, category name otherwise*

*\* date\_upper : same*

*\* date\_lower : same*

*\* }*

*\*/*

Even though the above part is a comment, it is very important since it describes the way the ‘request’ has to be structured so that the output can be provided by the application in a correct way.

if(*request*.query == undefined || *request*.query == null || *request*.query.length == 0){

*response*.send(JSON.stringify({status : "Error", message : "Invalid Query"}));

}

The code is used to prevent abnormal/wrong types of queries from being provided as input to the application. The query is checked by several filters to see if it fits the correct type of input or not and if it does not, a response is sent prematurely saying that the query does not adhere to the standard provided.

di = {}

di["q"] = query

if(!(*request*.country == "all")){

di["country"] = *request*.country; *// assuming country code will be valid*

}

if(!(*request*.category == "all")){

di["category"] = *request*.category;

}

if(!(*request*.language == "all")){

di["language"] = *request*.language

}

if(!(*request*.date\_upper == "all")){

di["from"] = *request*.date\_upper

}

if(!(*request*.date\_lower == "all")){

di["to"] = *request*.date\_lower

}

di["sortBy"] = "relevancy";

newsapi.v2.everything(di).then(*resp* =>{

*response*.send(JSON.stringify(*resp*));

return true;

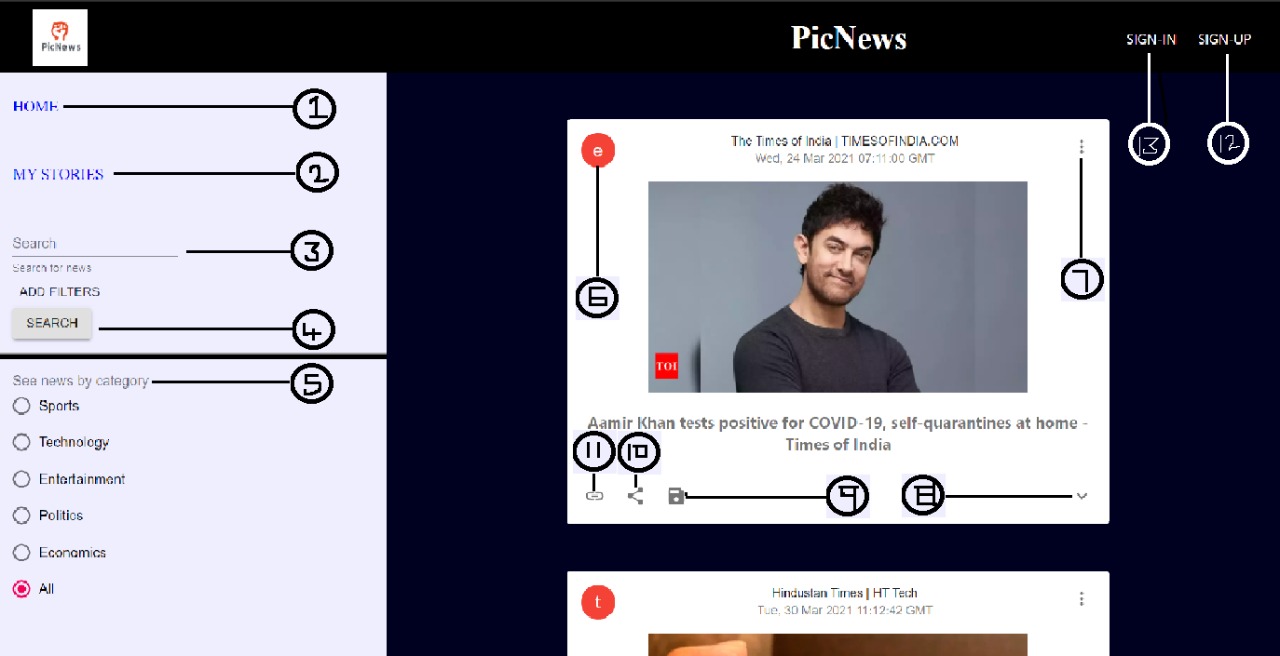
}).catch(*err*=>{

*response*.send(JSON.stringify(*err*));

})

The snippet starts with us declaring an empty array. The next few lines are filtering the code for our help. All the filters work the same way, ex; the first one checks if the news is requested from all countries and sends it, but in case only news from 1 country is desired, the news is sent depending on if the country code is valid. The rest of them work the same way. Then we have a line of code that is used to sort the news by its ‘Relevance’ so as to make the experience a better one. The leftover part is used to send the news articles to the user after the request has been processed.

UI



Attached is the screenshot of the Picnews App that is available to the user.

The item number 1 is the ‘HOME’ button. When the user clicks on an article, they are taken to it. After the user finishes reading it, the button can be clicked upon to bring the user back to the home page so that they may continue browsing the articles.

The item number 2 is the ‘MY STORIES’ button. The app provides the users to store the stories that they liked in their accounts’ storage so that important stories are never a touch away. This can be used for articles that are of utmost importance or produce fondness in the user.

The item number 3 is the Search bar. This is where the user will type the article/topic that they want to know more about. Using search algorithms, the articles that fit the bill are presented to the user.

The item number 4 is the ‘ADD FILTERS’ button. This button is used to incorporate filters that are used to help the search for articles be more directed and hence, fast.

The item number 5 is the categories menu. The menu contains several topics. Clicking on one will present all articles on that topic to the user.

The item number 6 is the topic descriptor. It is a single letter that is used to denote what category the article belongs to. In this case, the letter ‘e’ represents that the article belongs to the Entertainment category.

The item number 7 is the ‘Options’ dropdown. This presents the activities that the user can do with the article like sharing it, saving it to the ‘My Stories’ service etc.

The item number 8 is the summary dropdown. It presents a very short summary of what the article contains in case the user can’t afford to spend lots of time going through the whole article.

The item number 9 is the ‘Save’ button. This facility allows the user to save the article they liked to a location of their desire. It can be either in the ‘My Stories’ or even on their local device storage so that they can read it as per their desire.

The item number 10 is the ‘Share’ button. This allows the user to share the article they found interesting on a platform of their choice for no cost. This allows for a more permanent form of storage of the article.

The item number 11 is the ‘Link’ button. This provides the user with a link of the article in case they want to share it in a way not provided in the ‘Share’ button or in case the user likes to keep the links of articles he found interesting.

The item number 12 is the ‘SIGN-UP’ button. This allows non-existing users to create an account so that they can avail the facilities offered there.

The item number 13 is the ‘SIGN-IN’ button. The app can be used in 2 ways: either the user can read the articles anonymously or they can sign-in so that certain facilities which are available to account owners can be availed by them.