2020

Trend/Bing

CAB432

Assignment 2

Mark Burton & Mitch Traynor

N9801154 & n10325808

23/10/2020

Contents

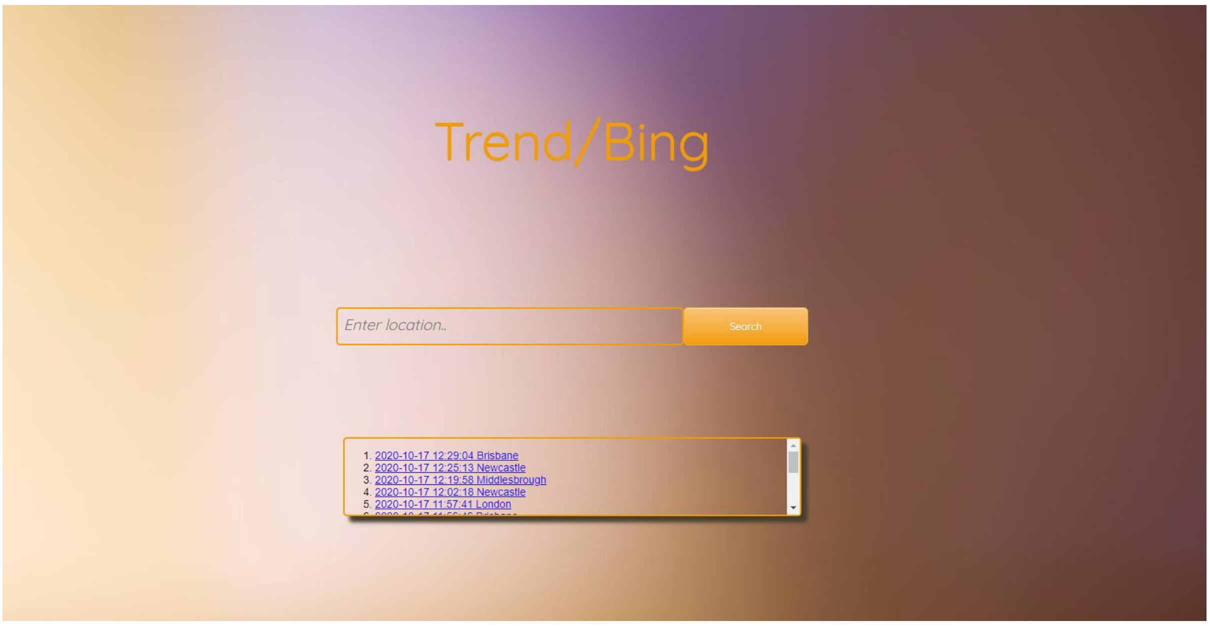
Introduction

Purpose & description

The purpose of this web application is to serve up news articles and images related to top trending topics for specific locations. The user is able to enter the name of a town, city, state or country and the app will communicate with the Twitter API, and direct the user to a page listing the top trending topics for that location. The user is then able to select a number of either news or images relating to each trend and the app communicates with either the Bing News Search API or the Bing Image Search API, respectively, to retrieve and display the requested news or image data to the user. This data is then kept int the web storage associated with the app.

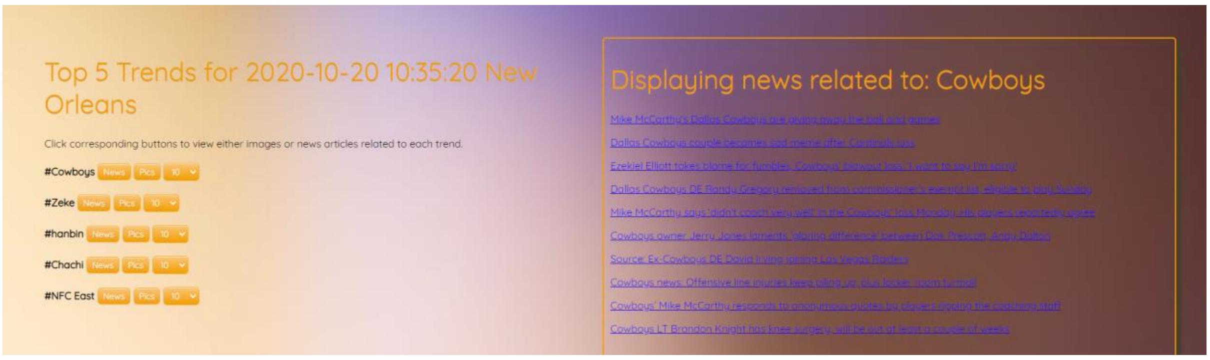
This application serves the purpose of providing news articles and images from a variety of sources to give users more information on trending topics for locations across the world. Social media has a reputation for providing unreliable information, however, Twitter is able to provide credible data regarding the trending topics of its users for certain locations. Combining these trends with news articles from more credible sources, provided by news aggregator Bing news, offers users far better information on trending topics than what may be provided in associated tweets. Similarly, the app's ability to provide a selection of images, from a variety of sources, relative to a trend gives the user an unbiased overview of trend topics that would not be available on Twitter.

The application has been designed to appear clean and uncluttered whilst still providing users with a smooth in retrieving informative information for any location quickly.



1. 2020-10-17 12:29:04 Brisbane
2. 2020-10-17 12:25:13 Newcastle
3. 2020-10-17 12:19:58 Middlesbrough
4. 2020-10-17 12:02:18 Newcastle
5. 2020-10-17 11:57:41 London

Figure 1 - The main page of the app appears clean and uncluttered.


Top 5 Trends for 2020-10-20 10:35:20 New Orleans

Click corresponding buttons to view either images or news articles related to each trend.

#Cowboys News Pics 10
#Zeke News Pics 10
#hanbin News Pics 10
#Chachi News Pics 10
#NFC East News Pics 10

Displaying news related to: Cowboys

Mike McCarthy's Dallas Cowboys are giving away the ball and games
Dallas Cowboys couple becomes sad meme after Cardinals loss
Ezekiel Elliott takes blame for fumbles, Cowboys' blowout loss: 'I want to say I'm sorry'
Dallas Cowboys DE Randy Gregory removed from commissioner's exempt list, eligible to play Sunday
Mike McCarthy says 'didn't coach very well' in the Cowboys' loss Monday. His players reportedly agree
Cowboys owner Jerry Jones laments 'glaring difference' between Dak Prescott, Andy Dalton
Source: Ex-Cowboys DE David Irving joining Las Vegas Raiders
Cowboys news: Offensive line injuries keep piling up, plus locker room turmoil
Cowboys' Mike McCarthy responds to anonymous quotes by players ripping the coaching staff
Cowboys LT Brandon Knight has knee surgery, will be out at least a couple of weeks

Figure 2 - Users are able to retrieve informative information quickly for any location.

Services used

The app uses the Twitter API, the Bing News Search API and the Bing Image Search API. Three separate endpoints of the Twitter API are required along with one endpoint for each of the Bing APIs, for a total of five endpoints. The three twitter APIs are used sequentially to return the trend data of the location closest to the user entered location that Twitter is able to provide trend data for. Each of the end points that are used in the app are detailed below.

For storage the two services that are used are Redis via the AWS service ElastiCache and the AWS service S3 Bucket storage. The Redis server is used when the news or the image button is pressed, when pressed the images or news link is stored in the redis for easy and fast response to the button being repressed. The S3 Bucket is used to allow users to see recent searches and in what area people are searching for.

Twitter Geo Search API (v.1.1)

Returns a list of valid locations matching the queried place name that are recognised by Twitter. An example response of this endpoint can be seen in Appendix A.

Endpoint: `https://api.twitter.com/1.1/geo/search.json`

Docs: https://developer.twitter.com/en/docs/twitter-api/v1/geo/places-near-location/api-reference/get-geo-search

Twitter Trends Closest API (v.1.1)

Returns the locations that Twitter has trending topic information for, closest to a specified location. An example response of this endpoint can be seen in Appendix B.

Endpoint: `https://api.twitter.com/1.1/trends/closest.json`

Docs: https://developer.twitter.com/en/docs/twitter-api/v1/trends/locations-with-trending-topics/api-reference/get-trends-closest

Twitter Trends Place API (v.1.1)

Returns the top 50 trending topics for a specific `WOEID`, if trending information is available for it. An example response of this endpoint can be seen in Appendix C.

Endpoint: `https://api.twitter.com/1.1/trends/place.json`

Docs: https://developer.twitter.com/en/docs/twitter-api/v1/trends/trends-for-location/api-reference/get-trends-place

Bing News Search API (v7)

Returns a list of relevant news articles matching a specified query.

Endpoint: `https://api.cognitive.microsoft.com/bing/v7.0/news/search`

Docs: https://docs.microsoft.com/en-us/rest/api/cognitiveservices-bingsearch/bing-news-api-v7-reference

Bing Image Search API (v7)

Returns a collection of relevant images matching a specified query.

Endpoint: `https://api.cognitive.microsoft.com/bing/v7.0/images/search`

Docs: https://docs.microsoft.com/en-us/rest/api/cognitiveservices-bingsearch/bing-images-api-v7-reference

Redis via “ElastiCache”

Stores the URL data for the news or image clicked to be used on reactivation of button.

Docs: https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/WhatIs.html

AWS S3

Stores long term data about recent searches.

Docs: https://docs.aws.amazon.com/s3/index.html

Use Cases and Services

To better illustrate the services that this app provides, the following use cases demonstrate how it may be used by different people.

News Articles for Local Trends

As an	Local resident
I want	To be able to read news articles of popular trends in my area
So that	I can keep up with current events relative to my community

In order to achieve this use case, the resident enters their home town into the text input field of the index page. The app then uses the Twitter API to retrieve the top trending topics for that area and redirects the resident to the "trending" route which displays the top 10 trends for the resident's home town. The resident would then press the "News" button of the trends that they wish to view news articles for. When a "News" button is pressed the app contacts the Bing News Search API and retrieves the top 10 news articles for that trend. It then updates the page to display the titles of the news articles which also serve as URL links to the news article pages themselves.

Image Viewer for Destination Trends

As a	Frequent traveler
I want	To be able to view images of trending topics for my next destination
So that	I can see what's currently hot at that location

In order to achieve this use case, the traveler enters their travel destination into the text input field of the index page. The app then uses the Twitter API to retrieve the top trending topics for the location and redirects the traveler to the "trending" route which displays the top 10 trends for the traveler's next destination. The traveler would then press the "Pics" button of the trends that they wish to view images for. When a "Pics" button is pressed the app contacts the Bing Image Search API and retrieves the top 20 Images for that trend. It then updates the page to display a collage of the images to the traveler.

Insight to Community Interests

As a	Traveling salesperson
I want	To get a quick overview of what's popular at each of the towns that I visit
So that	I can tailor my sales pitches to suite the current interest of my audience

In order to achieve this use case, the salesperson enters a town that they will visit into the text input field of the index page. The app then uses the Twitter API to retrieve the top trending topics for that town and redirects the salesperson to the "trending" route which displays the top 10 trends for the town. If the salesperson would like to view images for a particular trend then they may press the "Pics" button. The app contacts the Bing Image Search API and retrieves the top 20 Images for that trend. It then updates the page to display a collage of the images to the traveler. If the salesperson is intrigued by the images of a trend and wishes to view news articles for the same trend then they could press the "News" button. When the "News" button is pressed the app contacts the Bing News Search API and retrieves the top 10 news articles for that trend. It then updates the page to hide the currently displayed images and instead display the titles of the news articles which also serve as URL links to the news article pages themselves.

Overview of World Trends

As a	US President
I want	To view *and understand* what is trending on Twitter worldwide
So that	I can tweet about these trends myself and pretend I know what I am talking about

In order to achieve this use case, the president leaves the text input field of the index page blank. The app then uses the Twitter API to retrieve the top trending topics worldwide and redirects the president to the "trending" route which displays the top 10 trends in the world. If the president would like to view news articles for a particular trend then he may press the "News" button. The app then contacts the Bing News Search API and retrieves the top 10 news articles for that trend. It then updates the page to display the titles of the news articles which also serve as URL links to the news article pages themselves. If the president then decides that he prefers to look at pictures rather then read news articles then he could press the "Pics" button. When the "Pics" button is pressed the app contacts the Bing Image Search API and retrieves the top 20 Images for that trend. It then updates the page to hide the currently displayed news articles and instead display a collage of images.

Technical breakdown

When a new connection is made to the app it is directed to the appropriate route based upon the URL. The index route, is the home page of the app and is designed to have a clean and simple appearance. This page features a main heading, an item that comprises of an input text field and a button, and scroll box that contains previous searches. The user is prompted to either select a previous search via clicking on the hyperlink in the bottom box or enter a location into the input text field by a subtle placeholder. Pressing either the 'search' button or the 'enter' keyboard key will take the text entered in the input field as the search term to redirect the user to the trending route.

If a search query is received then the 'trending' route will make three separate calls to the Twitter API. The first request is made to the geo/search endpoint and uses the location entered by the user as the query parameter. This endpoint returns the valid locations recognized by twitter that match the queried location name. The coordinates of the top resulting location are used in the next sequential request to the trends/closest endpoint. This endpoint returns the closest location to the coordinates that Twitter is able to return trending topics for. The WOEID of the top resulting location is then used for the third request to the trends/place endpoint. This endpoint returns the top 50 trends for the WOEID location, of which, the top 10 are used to render the 'trending' page.
In the case that there is no search query for the trending route, then the first two endpoint requests are skipped and only the trends/place endpoint will be used. A WOEID of 1, which Twitter recognizes as worldwide, is used in this instance causing the endpoint to return the top trends worldwide. Again, the top 10 of these trends are used to render the 'trending' page. The search data is then stored in the S3 bucket, this data contains the key which is what is displayed in the scroll box, and the data that is received back from the API calls. This information will then be readily available for other users to use quickly.

The 'trending' page features; a heading which specifies the location of the top 10 trends along with a brief explanation of how to use the app. This page then displays 10 items containing the top 10 trends for the location along with 'News' and 'Pics' buttons, both of which are associated with a drop-down box of how many results the user wants displayed. When one of the buttons is pressed the app displays the requested data returned from either the bingNews or bingImages routes. In the case of the 'News' button, the bingNews route uses the Bing News Search API to retrieve the selected amount of news articles related to that trend and sends it to the client. Likewise, the bingImages route uses the Bing Images Search API to retrieve the selected amount of images related to that trend and sends it to the client. Once the data has been received on the client-side, it is displayed to the user in a separate box to the right of the screen. In order to keep this page uncluttered, it is not possible to display both news stories and images for the same trend at the same time. If news stories are currently been displayed and the Pics button is pressed then the news stories will be removed before the requested images are displayed to the user. Similarly, if the news button is pressed whilst images are currently displayed for the same trend then the images will be removed before the requested news articles are displayed. As the button gets pressed for the first time, the application will send the information to the Redis server, this data is then stored in the short term cache provided by AWS for the event that the user decides to look at the same field twice. Finally, if a button Is pressed whilst it's corresponding data is already been displayed then that data will simply be removed and no fetch request will be made to the server.

This application is built using the express application framework and utilizes pug engine support for generating the front-end HTML. The first step of building the application was to generate an empty express app using express-generator. Then, the server-side of the app was developed starting with implementation of the communication with the required API endpoints for Twitter and Bing. The first

Twitter API endpoint, geo/search, was reached first using NPM package 'twit', then, subsequently the other Twitter endpoints, trends/closest and trends/place were reached in the same way. The response data of each of these endpoints was then manipulated into the request parameters of the other endpoints until the correct trends for a given location could be received. Next, both the Bing News Search API and Bing Image Search API were reached using the NPM package 'node-bing-api'. Once all of the required API endpoints could be reached and the required data retrieved from each, they were used in the implementation of the app routes; twitter, bingImages and bingNews. To complete the server-side JavaScript component of the app an index route was built to serve the home page of the web app.

The next stage of the building process of the app was to start building the HTML component to be send to the client-side of the app and displayed to the user. For this the pug engine was used to generate the HTML documents, starting with the 'layout' view which defines the title and favicon along with a background, all inherited by the 'index' and 'trending' user views. The 'trending' view was then created featuring a main heading informing of the location that the displayed trends belong to, a brief explanation of how to view news or images related to the trends, and the trends themselves along with 'News' and 'Pics' buttons for each and their corresponding drop-down boxes. Once the 'trending' page was displaying correctly, a client-side JavaScript was built to handle the button click events. The client-side JavaScript retrieves data from either the bingImages or bingNews routes depending on which button is pressed and updates the 'trending' page to display the required data to the user and stores the data in the associated Redis server. Alternatively, if the data relating to the button that is pressed is already showing then it will be removed and, if data of a different type will be replaced with the most recently requested data type. The 'index' view was then created featuring a main heading, an input text box and search button to receive the location that the user wants to receive trending data for, below this a scroll box has been made to display the previous searches which can be selected to display previous data. A client-side JavaScript was attached to the 'index' view that would serve to retrieve the user input text from the text box and use it to redirect the user to the 'trending' page, or retrieve the data out of the S3 storage to display on the 'trending' page.

When implementing the twitter route, issues were encountered when sequentially making requests to the three Twitter endpoints. The issue was caused by stringing callback functions together, resulting in complex and confusing code commonly referred to as known as callback hell. In order to resolve this issue promises were utilized to make the code less confusing and easier to work with. Luckily, the 'twit' package get function supports the return of promises which are objects that represent the eventual completion (or failure) of an asynchronous operation and its resulting value.

Another challenge that was overcome as part of the implementation of this web application was an inexperience with HTML, CSS and PUG which made building the front end of the application difficult. This was overcome by reading lots of online material relative to the front-end development of web apps complimented by you tube videos. The result is an elegantly simple implementation that is able to achieve its purpose whilst remaining a clean, uncluttered appearance.

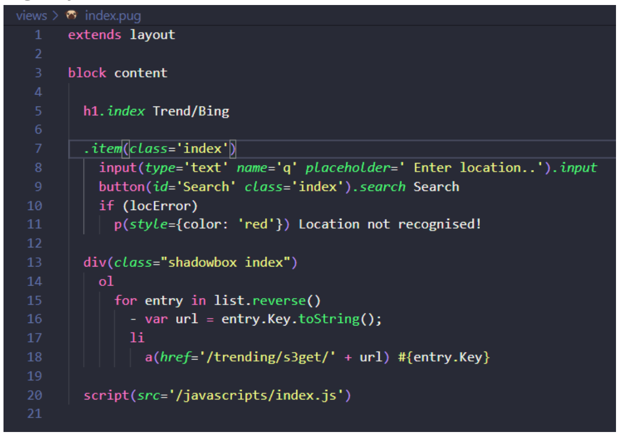
Architecture and Data Flow

NPM package Express-Generator was used to generate a template for the application. Additionally, Pug engine support was added to the app generation to assist in the development of the app's frontend. Part of the express-generated code, 'bin/www', sets out the network parameters of the app. It sets the port number that the app runs on to 3000 and uses it to create a http server.

When a new connection is made to the app, app.js receives the connection and either routes it to the appropriate route based upon the URL, or, handles any errors. The 4 routes that are defined in app.js are:

- '/' – This is the route to the user interface index page of the app and directs the app to the index router, 'routes/index.js'.
- '/trending?' - This is the route to the user interface trending page of the app and directs the app to the trending router, 'routes/twitter.js'.
- '/trending/news?' - This is the route used by the client side of the 'trending' page which fetches news articles from the Bing news router 'routes/bingNews.js'.
- '/trending/pics?' - This is the route used by the client side of the 'trending' page which fetches images from the Bing image router 'routes/bingImages.js'.

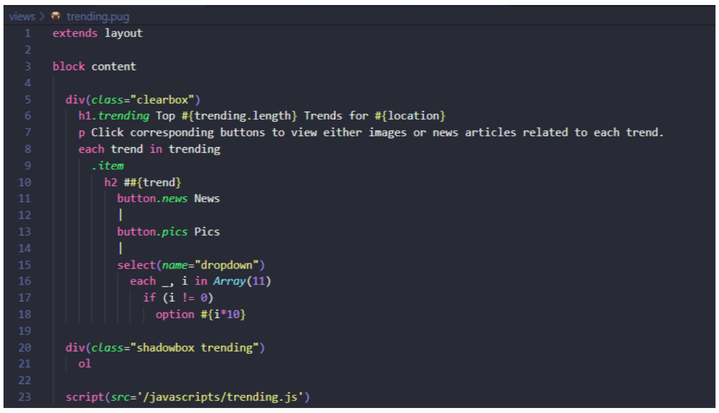
The index route, handled in the 'routes/index.js' file, is the home page of the app and is rendered using the 'index.pug' file, shown in figure 3. This page features; a main heading (h1), and an item that comprises of an input text field and a button. Pressing either the 'search' button or the 'enter' keyboard key will trigger a client-side action listener which takes the text entered in the input field as the search term to redirect the user to the trending route. The last div in the file is the scroll box that will display the recent searches, if these hyperlinks get clicked the user will be redirected to the 'trending' route and will be served with the previous data.



```
views > index.pug
extends layout

block content

  h1.index Trend/Bing

  .item(class='index')
    input(type='text' name='q' placeholder=' Enter location..').input
    button(id='Search' class='index').search Search
    if (locError)
      p(style={color: 'red'}) Location not recognised!

  div(class="shadowbox index")
    ol
      for entry in list.reverse()
        - var url = entry.Key.toString();
        li
          a(href='/trending/s3get/' + url) #{entry.Key}

  script(src='/javascripts/index.js')
```

Figure 3 - PUG engine file for the index.

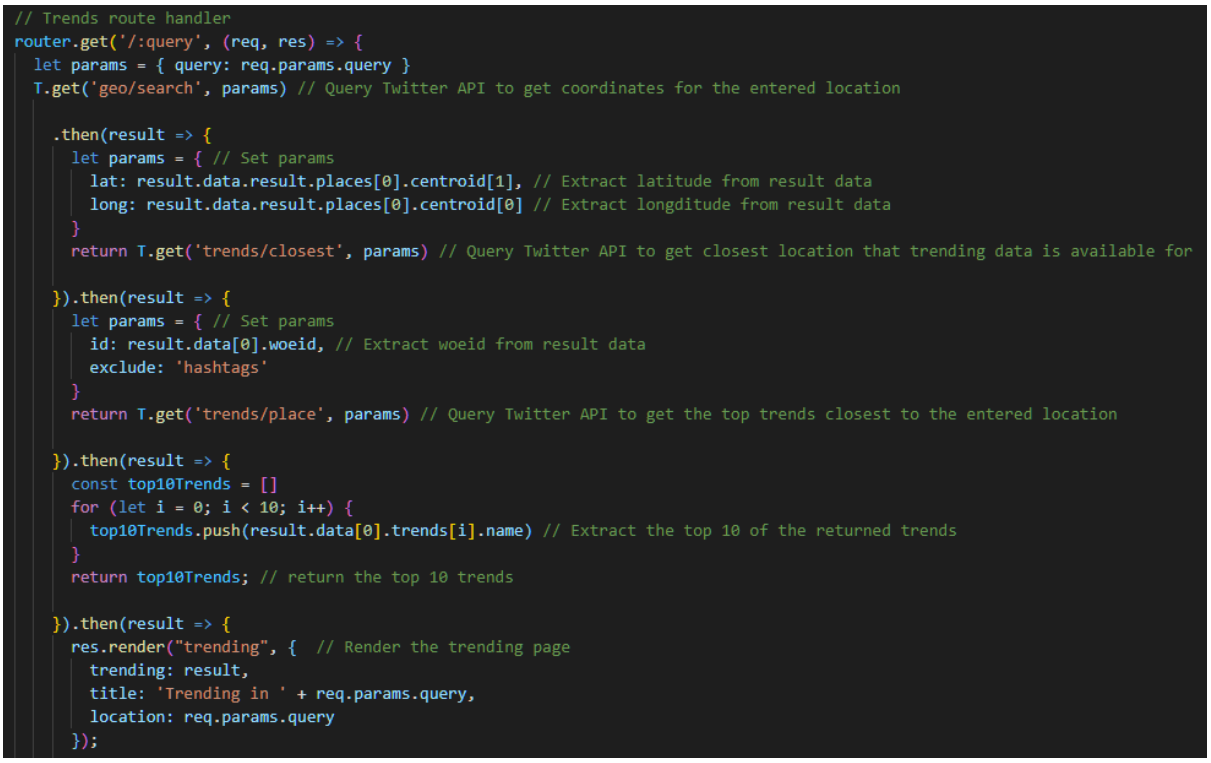
The trending route, handled in the 'routes/twitter.js' file, has two router.get functions, one is used to process URL requests containing a search query and another is used to handle empty requests that do not contain a search query.
If a search query is received then this route will sequentially make three separate calls to the Twitter API, as shown in figure 5. The first request is made to the geo/search endpoint and uses a location as the query parameter. The coordinates of the top resulting location are then extracted from the response and the latitude and longitude values are used as the parameters to the next request in the sequence, the trends/closest endpoint. The default scale of the locations that Twitter is able to provide trending topic data for is 'neighborhood'. The WOEID of the top resulting location is then used as the id parameter to the next request in the sequence, the trends/place endpoint. Additionally, 'hashtags' is included as the exclude parameter to exclude any hashtags from the result trends. This endpoint then returns the top 50 trends for the WOEID location, of which, the top 10 are extracted and added to a string array using a for loop.



```
views > trending.pug
extends layout

block content

  div(class="clearbox")
    h1.trending Top #{trending.length} Trends for #{location}
    p Click corresponding buttons to view either images or news articles related to each trend.
    each trend in trending
      .item
        h2 ##{trend}
          button.news News
          |
          button.pics Pics
          |
          select(name="dropdown")
            each _, i in Array(11)
              if (i != 0)
                option #{i*10}

  div(class="shadowbox trending")
    ol

  script(src='/javascripts/trending.js')
```

Figure 4 - PUG engine file for the trending page.

Once the top 10 trends for the entered location have been retrieved from the Twitter API, they are passed to the 'trending' page which is rendered using the 'trending.pug' file, shown in figure 4.
In the case that there is no search query for the trending route, then the first two endpoint requests are skipped and only the trends/place endpoint will be used. A default WOEID of 1, which Twitter recognizes as worldwide, is used in this instance causing the endpoint to return the top trends worldwide. Again, the top 10 of these trends are extracted into and array which is then used to render the 'trending' page using the 'trending.pug' file.



```
// Trends route handler
router.get('/:query', (req, res) => {
  let params = { query: req.params.query }
  T.get('geo/search', params) // Query Twitter API to get coordinates for the entered location

    .then(result => {
      let params = { // Set params
        lat: result.data.result.places[0].centroid[1], // Extract latitude from result data
        long: result.data.result.places[0].centroid[0] // Extract longditude from result data
      }
      return T.get('trends/closest', params) // Query Twitter API to get closest location that trending data is available for

    }).then(result => {
      let params = { // Set params
        id: result.data[0].woeid, // Extract woeid from result data
        exclude: 'hashtags'
      }
      return T.get('trends/place', params) // Query Twitter API to get the top trends closest to the entered location

    }).then(result => {
      const top10Trends = []
      for (let i = 0; i < 10; i++) {
        top10Trends.push(result.data[0].trends[i].name) // Extract the top 10 of the returned trends
      }
      return top10Trends; // return the top 10 trends

    }).then(result => {
      res.render("trending", {  // Render the trending page
        trending: result,
        title: 'Trending in ' + req.params.query,
        location: req.params.query
      });
```

Figure 5 - Part of the 'views/twitter' route showing the three sequential requests made to the Twitter API to return the top 10 trends for a user specified location.

The 'trending' page uses; a h1 element to specify the location that the app is showing the top 10 trends, and a p element which contains a brief explanation of how to use the app. This page then displays 10 items containing the top 10 trends for the location as an h2 element, along with 'News' and 'Pics' buttons with an associated drop-down box for the amount of results. Embedded in this page is also a client side trending.js file, which is used to handle button press events and update the page accordingly. When one of the buttons is pressed an event is triggered in the client-side JavaScript which uses the button type along with the trend of which it is associated with to make a fetch request to either the bingNews or bingImages routes. In the case of the 'News' button, the bingNews route uses the Bing News Search API to retrieve the selected news articles related to that trend and sends it to the client-side JavaScript. Likewise, the bingImages route uses the Bing Images Search API to retrieve the selected number of images related to that trend and sends it to the client-side JavaScript. Once the data has been received on the client-side, it is displayed to the user as part of the item of that trend. In order to keep this page uncluttered, it is not possible to display both news stories and images for the same trend at the same time. If news stories are currently been displayed and the Pics button is pressed then the news stories will be removed before the requested images are displayed to the user. Similarly, if the news button is pressed whilst images are currently displayed for the same trend then the images will be removed before the requested news articles are displayed. Finally, if a button Is pressed whilst it's corresponding data is already been displayed then that data will simply be removed and no fetch request will be made to the server.

The routes of the application, found in the 'routes' directory, are all handled on the server. These contain all the code required to interface with the API endpoints that are used. Additionally, all processing of the data that is received from the API endpoints is done in these files in the client-side. Also, on the server side are the PUG engine files, found in the 'views' directory, that are used to generate the HTML that is sent to the client. These receive data and are rendered from within the server route JavaScript files. The 'public' directory contains client-side data, including the favicon icon, the client-side JavaScript files and the CSS stylesheet. The client-side JavaScript files mostly handle client events such as button presses, however, 'public/javascripts/trending.js', is responsible for determining what data is currently being displayed in order to make the correct action based upon which button was pressed. One such action that may be required is to make fetch requests to either the bingNews or bingImage routes and add the corresponding data to the existing HTML. Additionally, this script may be required to remove existing data, display new data or both.

The storage for this app involves two stages. Once the user inserts the location, the app will run the java script to display the trending page, at the same time the app sends this data to the associated S3 bucket that has been set up for long term storage. This bucket will hold 10000 searches and the user will be able to select a search that happened and it will display the result of a location search at a specific time. The second style of storage comes in the form of a Redis server supplied by the 'ElastiCache' service from AWS. This storage is used when the trends of a location is displayed the associated image and news data for each is added to Redis cache to access to the results quickly and without further API calls.

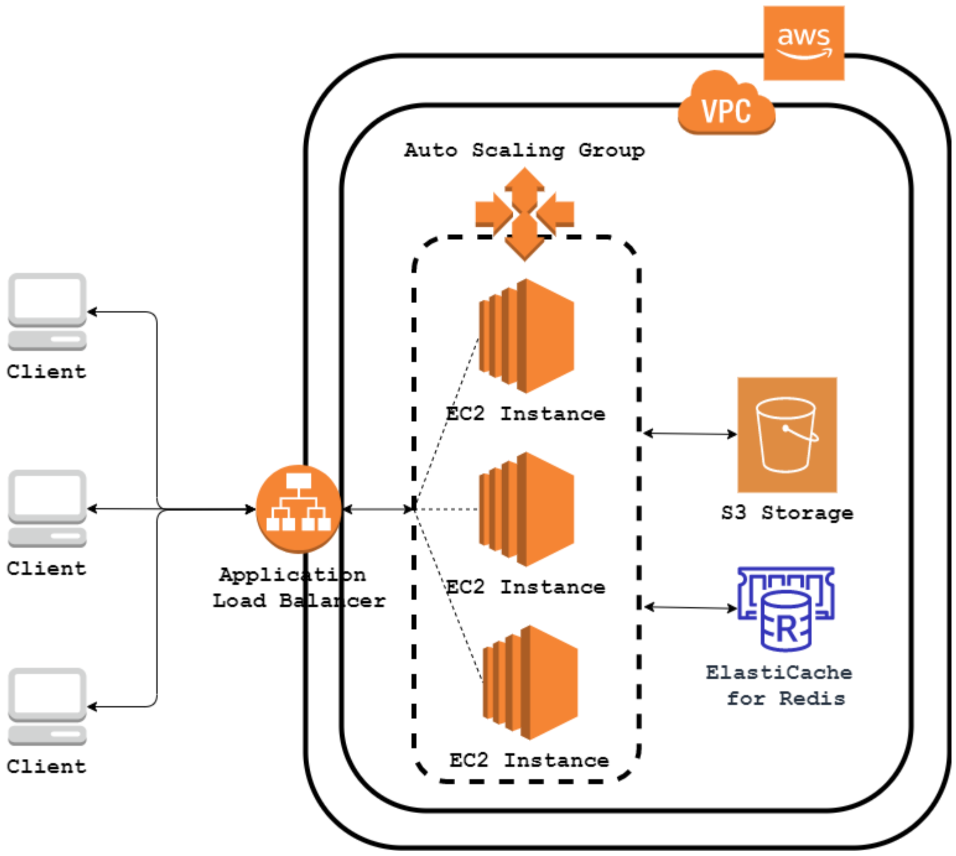


Figure 6 - Cloud architecture diagram of the application

As we expect this app to become a trend itself, we have implemented a small form of scaling with the use of autoscaling via AWS auto scaling group as shown in the architecture diagram in figure 6 In order to distribute the client requests evenly over instances in the scaling group we used an application load balancer as this AWS product is best suited to distributing HTTP requests. To ensure that we will be running the app over a large amount of instances and thus lower the risk of overworking a single instance, the instances will scale from one to five depending on the load but will scale early, at 5 percent of the full load, this will allow the instance to scale up quickly and then the auto load balancing will distribute requests on the app evenly.

Figure 7 shows a diagram of the data flow of the application.

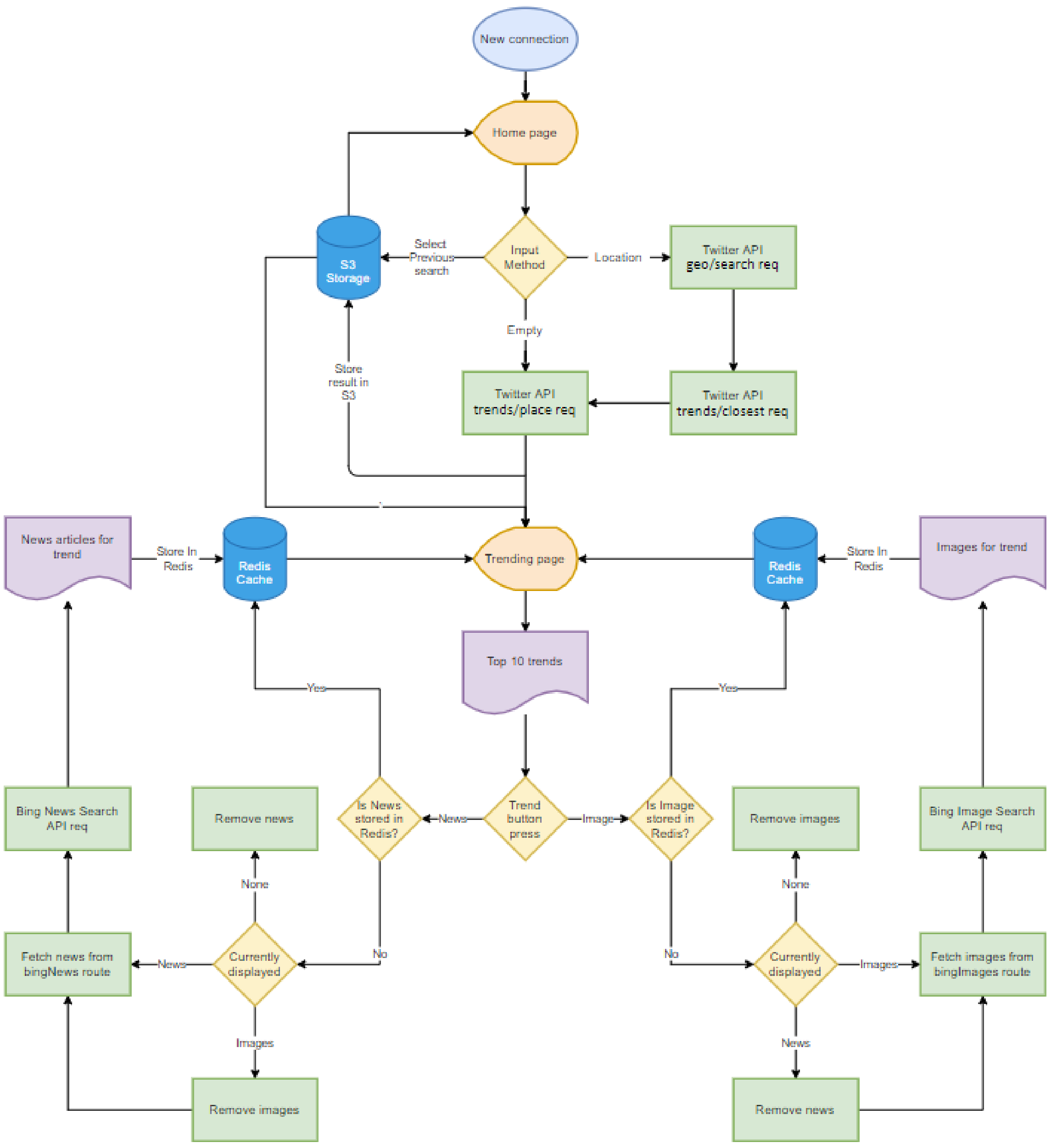


Figure 7 - Flow chart diagram of the data flow of the application.

Deployment

To deploy this application, the design team implemented git to do a git pull to load all files on a single instance. After the instance was able to run the code using the ‘npm start’ command, then pm2 was installed and set up to initialize and to set a boot and run command for when the instance turned on. This means that when the instance turns on it immediately will run an ‘npm start’ command and this will then in turn start the application and open port 3000.

Scaling and Performance.

As stated above, the design team have high hopes for this app. Because of this an automatic scaling group has been implemented to handle the expected load. To accomplish this, the instance was created and then once the instance was working bug free, we stored the image in the AWS database for later use as well as to implement scaling.

The goal of the scaling in this project is to have a high number of instances that will scale automatically. To accomplish this the CPU usage threshold was set to 5%, this figure is quite small but setting it low means that when the app receives a lot of requests the scaling group will increase quickly and dynamically to ensure maximum performance. To demonstrate the scaling in action A postman request collection was run, this collection was run 4 times concurrently with 1500 requests each. These requests were targeted at the IP address and the specific port. The results are displayed below in figure 8. As displayed below once the load leaves the instances the auto scale group will revert the minimum group and wait to rescale

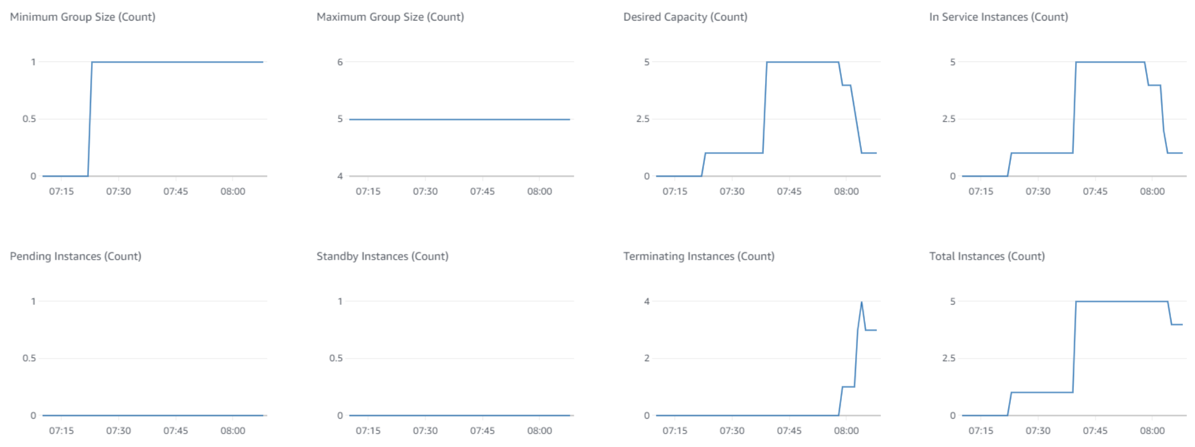


Figure 8 - Scaling monitoring AWS.

To achieve a high level of performance the two storage types mentioned above were instigated. The Redis is the short-term storage that will drastically improve the response times. The response for a not stored search is roughly 200ms, though when the application loads the results from the Redis the responses drop drastically to around 50ms. This change is extremely evident in practice where the first search we can watch load, where is the second search will look like it is already there.

Test plan

In order to test that the final release of the web app meets its design specifications a series of manual tests were carried out. The tests applied to the app along with their expected outcomes and results can be viewed in figure 9 and screenshots of the tests are included in the Appendix.

#	Test	Expected Outcome	Actual Outcome	Pass/Fail	Screenshot/s
1	The web app can be connected to when running on a local network	The web app home page is displayed at http://localhost:3000	The home page is displayed at http://localhost:3000	Pass	Appendix E
2	Entering a location	The web app should redirect to the 'trending' page and display top 10 trends for the entered location	User redirected to the 'trending' page and top 10 trends are displayed for the entered location	Pass	Appendix F
3	Leaving the input text field blank	The web app should redirect to the 'trending' page and display top 10 trends worldwide	User redirected to the 'trending' page and top 10 trends are displayed for the entered location	Pass	Appendix G
4	Pressing 'Pics' button for a trend with no other data showing	Images are displayed for the trend	Images are displayed for the trend	Pass	Appendix H
5	Pressing 'Pics' button for a trend with 'News' data showing	Currently displayed news articles are removed and replaced with images for the trend	Currently displayed news articles are removed and replaced with images for the trend	Pass	Appendix I
6	Pressing 'News' button for a trend with no other data showing	News articles are displayed for the trend	News articles are displayed for the trend	Pass	Appendix J
7	Pressing 'News' button for a trend with 'Pics' data showing	Currently displayed images are removed and replaced with news articles for the trend	Currently displayed images are removed and replaced with news articles for the trend	Pass	Appendix K
8	Images displayed for multiple trends	Images for multiple trends can be displayed	Images for multiple trends can be displayed at once	Pass	Appendix L
9	News displayed for multiple trends	News articles for multiple trends can be displayed	News articles for multiple trends can be displayed at once	Pass	Appendix M
10	Run app on instance	Instance will run app on IP address and on port 3000	Instance showing app on ip address and port 3000	Pass	Appendix E
11	Postman request to check scaling.	Send 4500 requests and instance should scale	Scales correctly	Pass	Appendix M
12	use the S3 Storage to get previous request	load previous request	loaded previous request	Pass	Appendix N
13	load images at 10 then clear then 50 to see redis response	loads first 10 fast then loads others slower	loads first 10 fast then loads others slower	Pass	Appendix O

Figure 9 - Table of tests carried out on the app.

Difficulties and Exclusions

Some difficulties were experienced when implementing the three sequential requests to the aforementioned Twitter API endpoints required when a location is entered. The problems were caused by staggering callback functions within one another. This caused messy and difficult to understand code that proved very difficult to work with, an issue commonly referred to as 'callback hell'. A solution was sought and a solution found in promises. promises are an object that represents the eventual value returned by an asynchronous function and were introduced into NodeJS in 2011 [3]. Implementing promises, which are supported in the 'twit' NPM package used, vastly improved the appearance of the code and made it much easier to work with.

Another difficulty experienced as part of the development of this application was the implementation of the front-end HTML that is displayed in the web browser to the user. This difficulty was due to a lack of experience working with this technology. This difficulty was overcome by learning some basic PUG engine commands to easily generate the required HTML. The main source of this learning came from an informative video provided by Michael Esteban [4]. Applying this method of generating HTML streamlined the front-end development of the application and provided a good platform to build upon.

A deviation from the initial design proposal of the application was the decision to apply the ElastiCache for Redis persistence method to the caching of news and image data for each trend. This change was made as the initial use case of the Redis cache was extremely close to that of the S3 storage persistence method and would not have made any meaningful impact on the application. This revised use of the Redis cache compliments the functionality of the application and offers users much decreased latency when displaying images of news articles from the cache rather than requesting them from the API each time.

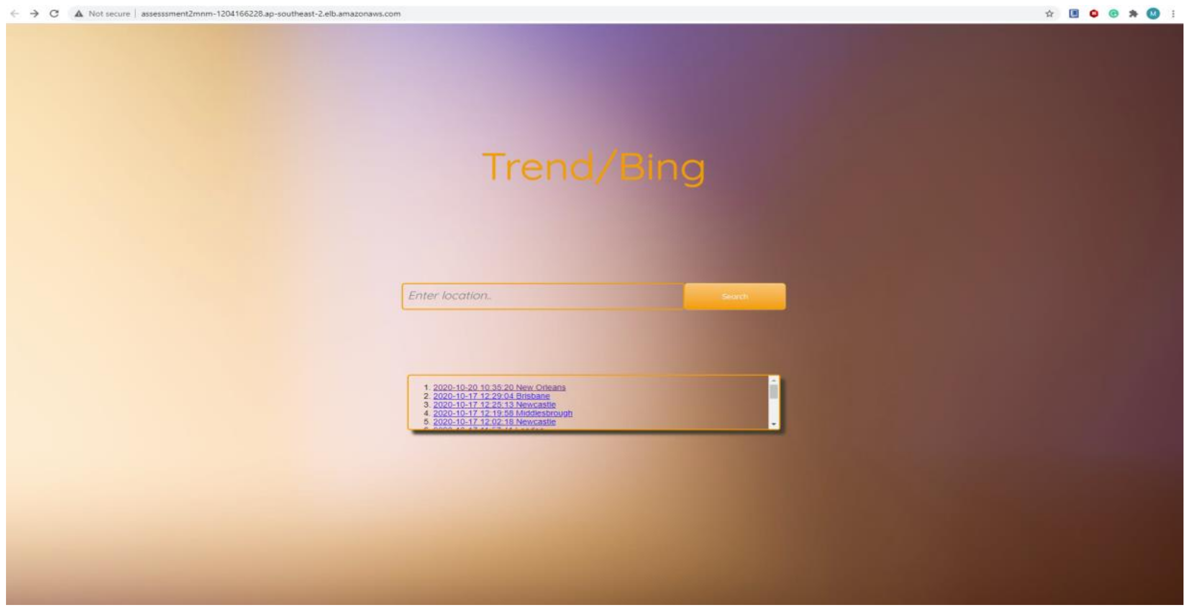
Extensions

There is much room for extending the application in its current form from both a user experience and web security view point. To improve user experience other Azure Cognitive Services such as Bing Video Search, which returns links to videos, could be added to offer users a more diverse selection of data. This was not included in this release of the application, as it did not fall within the design scope of the project. Routing products such as WAS Route 53 could be used to apply a more appropriate web address that users can use to access the application, such as www.trend-bing.com.

To improve web security the application could be extended to incorporate AWS protection products such as WAF and Shield which can be used to mitigate malicious attacks such as SQL-injection or distributed denial of service (DDoS).

User guide

When a connection is made to the web application, the user is presented with the home page shown in figure 10.


Not secure | assesssment2mnm-1204166228.ap-southeast-2.elb.amazonaws.com
Trend/Bing
Enter location..
Search

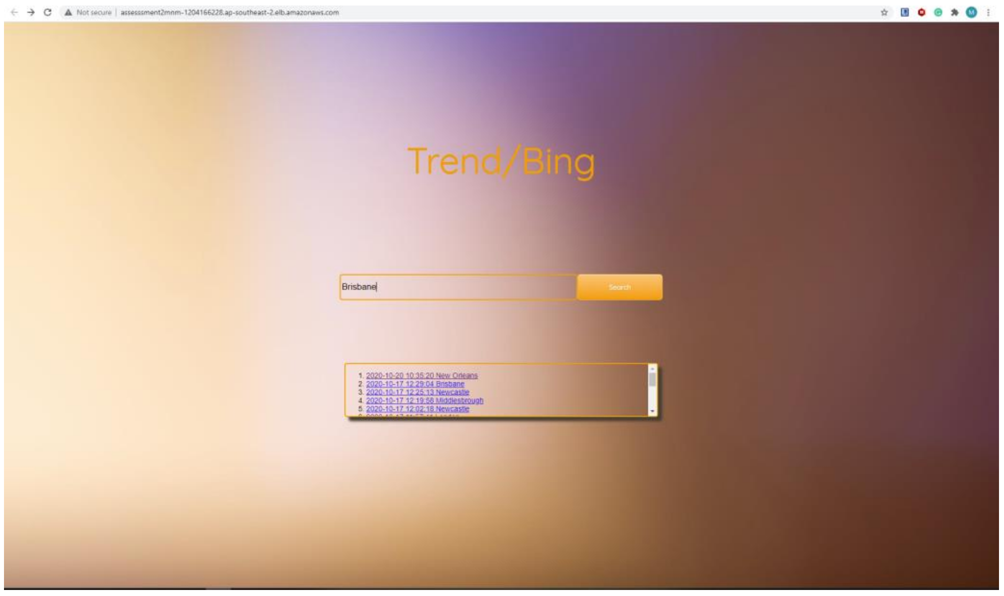
1. 2020-10-20 10:35:20 New Orleans
2. 2020-10-17 12:29:04 Brisbane
3. 2020-10-17 12:25:13 Newcastle
4. 2020-10-17 12:19:58 Middlesbrough
5. 2020-10-17 12:02:18 Newcastle

Figure 10 - Home page of the web application

The user is able to enter the name of a location that they wish to find information of the trending topics for. Alternatively, if the text input is left blank then the application will default to find to top worldwide trends. Another option is to use the previous searches in the scroll box below the search box.

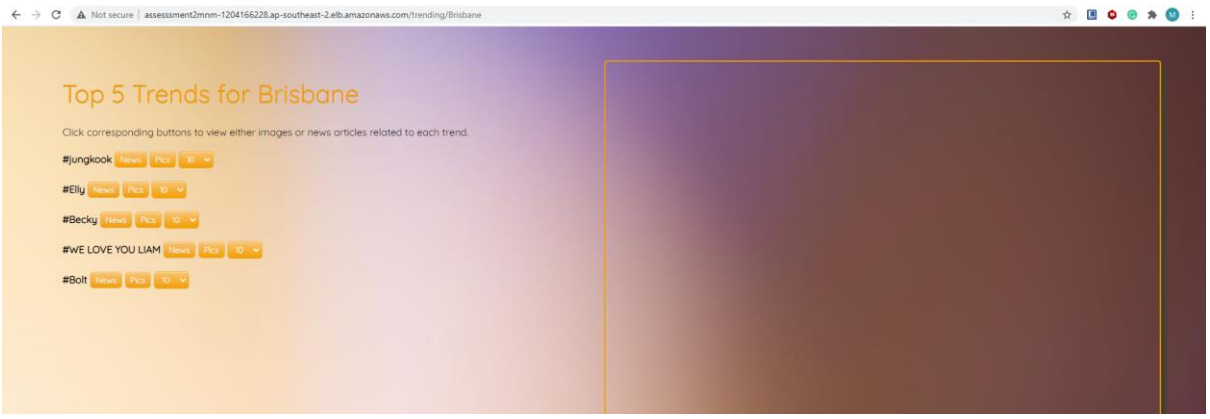
First, we will go over the text input approach then cover the historical search after.

Either the search button or the enter key on the keyboard will prompt the application to redirect the user to the 'trending' page.


Not secure | assesssment2mnm-1204166228.ap-southeast-2.elb.amazonaws.com
Trend/Bing
Brisbane
Search

1. 2020-10-20 10:35:20 New Orleans
2. 2020-10-17 12:29:04 Brisbane
3. 2020-10-17 12:25:13 Newcastle
4. 2020-10-17 12:19:58 Middlesbrough
5. 2020-10-17 12:02:18 Newcastle

Figure 11 - Home page with a location inputted as a query


Not secure | assesssment2mnm-1204166228.ap-southeast-2.elb.amazonaws.com/trending/Brisbane

Top 5 Trends for Brisbane

Click corresponding buttons to view either images or news articles related to each trend.

#jungkook News Pics 10

#Elly News Pics 10

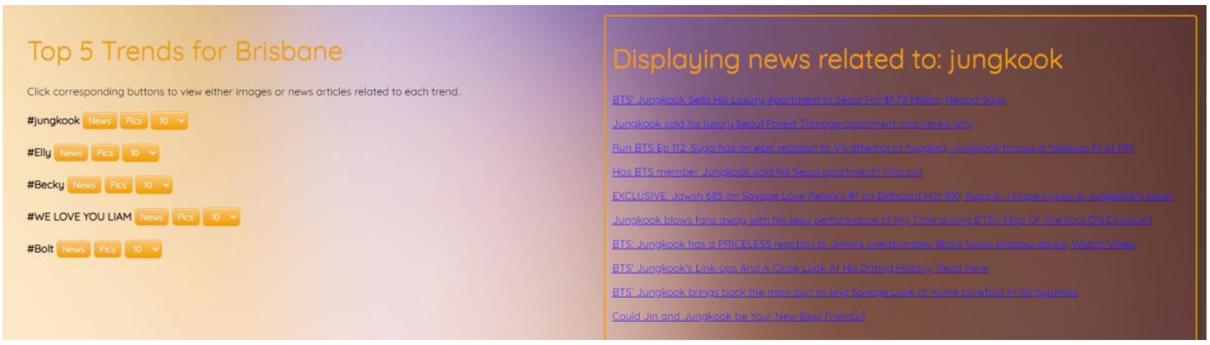
#Becky News Pics 10

#WE LOVE YOU LIAM News Pics 10

#Bolt News Pics 10

Figure 12 - Trending page displaying the top 10 trends for Brisbane

From this page the user is free to select to view either news articles or images related to each trend using the buttons beside it. Figures 12, 13 and 14 show some examples of news and images being displayed for different trends.


Top 5 Trends for Brisbane

Click corresponding buttons to view either images or news articles related to each trend.

#jungkook News Pics 10

#Elly News Pics 10

#Becky News Pics 10

#WE LOVE YOU LIAM News Pics 10

#Bolt News Pics 10

Displaying news related to: jungkook

BTS' Jungkook Sells His Luxury Apartment In Seoul For $1.79 Million, Report Says

Jungkook sold his luxury Seoul Forest Trimage apartment and here's why

Run BTS Ep 112: Suga has an epic reaction to V's attempt at hugging, Jungkook throws a hilarious fit at RM

Has BTS member Jungkook sold his Seoul apartment? Find out

EXCLUSIVE: Jawsh 685 on Savage Love Remix's #1 on Billboard Hot 100, Suga & J-Hope's lyrics & Jungkook's cover

Jungkook blows fans away with his sexy performance of My Time during BTS's Map Of The Soul ONE concert

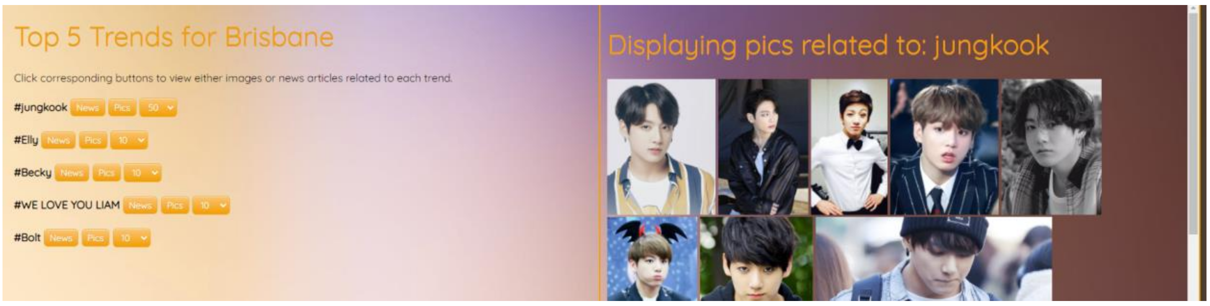
BTS: Jungkook has a PRICELESS reaction to Jimin's overdramatic Black Swan shadow dance; Watch Video

BTS' Jungkook's Link-ups And A Close Look At His Dating History; Read Here

BTS' Jungkook brings back the man-bun to sing Savage Love at home barefoot in his pyjamas

Could Jin and Jungkook be Your New Best Friends?

Figure 13 - News stories displayed for different trends


Top 5 Trends for Brisbane

Click corresponding buttons to view either images or news articles related to each trend.

#jungkook News Pics 50

#Elly News Pics 10

#Becky News Pics 10

#WE LOVE YOU LIAM News Pics 10

#Bolt News Pics 10

Displaying pics related to: jungkook

Figure 14 - Images displayed for different trends

The user is able to press buttons more than once and the data displayed will reflect that of the most recent button pressed, unless the data is removed by the most recent button press.

Note: to avoid overly cluttered pages, users are only able to display one of either news articles or images for each trend.

As for the use of the Historical search function the user is able to select from the hyperlinks in the scroll box below the text entry fields. This can be done by clicking the link wanted. A display of this is in figure 15 and then results in figure 16.

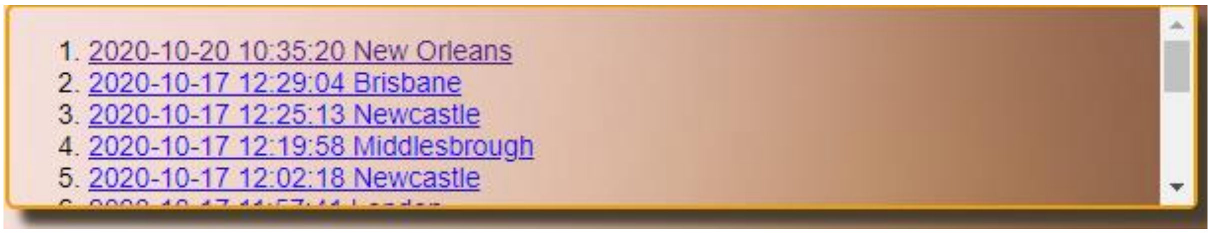
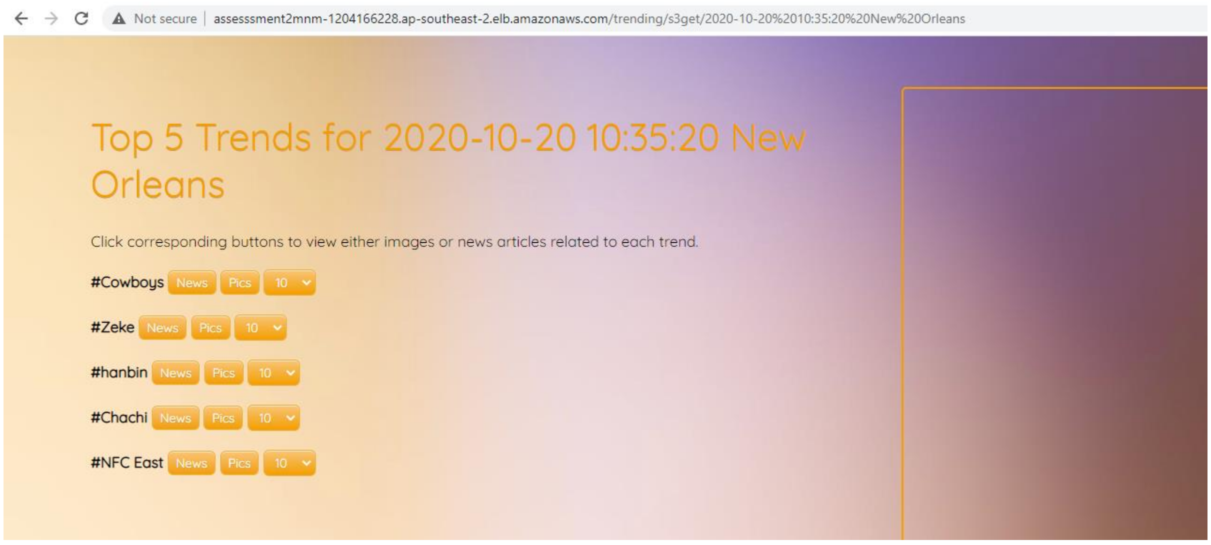

1. 2020-10-20 10:35:20 New Orleans
2. 2020-10-17 12:29:04 Brisbane
3. 2020-10-17 12:25:13 Newcastle
4. 2020-10-17 12:19:58 Middlesbrough
5. 2020-10-17 12:02:18 Newcastle

Figure 15 - scroll box containing Hyperlinks.


Not secure | assesssment2mnm-1204166228.ap-southeast-2.elb.amazonaws.com/trending/s3get/2020-10-20%2010:35:20%20New%20Orleans

Top 5 Trends for 2020-10-20 10:35:20 New Orleans

Click corresponding buttons to view either images or news articles related to each trend.

#Cowboys News Pics 10

#Zeke News Pics 10

#hanbin News Pics 10

#Chachi News Pics 10

#NFC East News Pics 10

Figure 16 - Results. Take note of URL saying it got info from S3.

References

[1] Docker Inc., “Dockerfile reference | Docker Documentation,” 2020. https://docs.docker.com/engine/reference/builder/ (accessed Sep. 12, 2020).

[2] Node.js “Releases | Node.js.” https://nodejs.org/en/about/releases/ (accessed Sep. 12, 2020).

[3] B. Diuguid “Asynchronous Adventures in JavaScript: Promises | by Benjamin Diuguid | DailyJS | Medium.” https://medium.com/dailyjs/asynchronous-adventures-in-javascript-promises-1e0da27a3b4 (accessed Sep. 13, 2020).

[4] M. Esteban, “*Wikipedia-Overview-Video.mp4” 2020*. https://blackboard.qut.edu.au/bbcswebdav/pid-8822856-dt-content-rid-33907933_1/courses/CAB432_20se2/Wikipedia-Overview-Video.mp4 (accessed Sep. 13, 2020).

Appendix A – Example response of the Twitter API geo/search endpoint

```
{
    "result": {
        "places": [{
            "id": "3797791ff9c0e4c6",
            "url": "https://api.twitter.com/1.1/geo/id/3797791ff9c0e4c6.json",
            "place_type": "city",
            "name": "Toronto",
            "full_name": "Toronto, Ontario",
            "country_code": "CA",
            "country": "Canada",
            "centroid": [-79.27828265214646, 43.629311],
            "bounding_box": {
                "type": "Polygon",
                "coordinates": [
                    [
                        [-79.639319, 43.403221],
                        [-79.639319, 43.855401],
                        [-78.90582, 43.855401],
                        [-78.90582, 43.403221],
                        [-79.639319, 43.403221]
                    ]
                ]
            }
        },
```

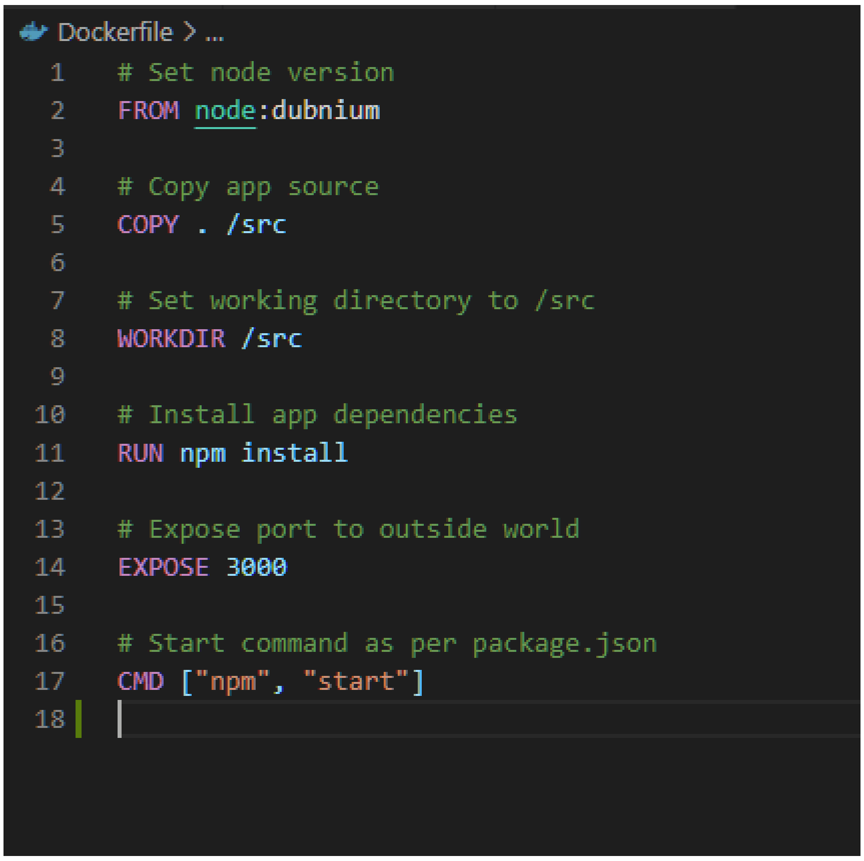
Appendix B – Example response of the Twitter API trends/closest endpoint

```
 [
  {
    "country": "Australia",
    "countryCode": "AU",
    "name": "Australia",
    "parentid": 1,
    "placeType": {
      "code": 12,
      "name": "Country"
    },
    "url": "http://where.yahooapis.com/v1/place/23424748",
    "woeid": 23424748
  }
]
```

Appendix C – Example response of the Twitter API trends/place endpoint

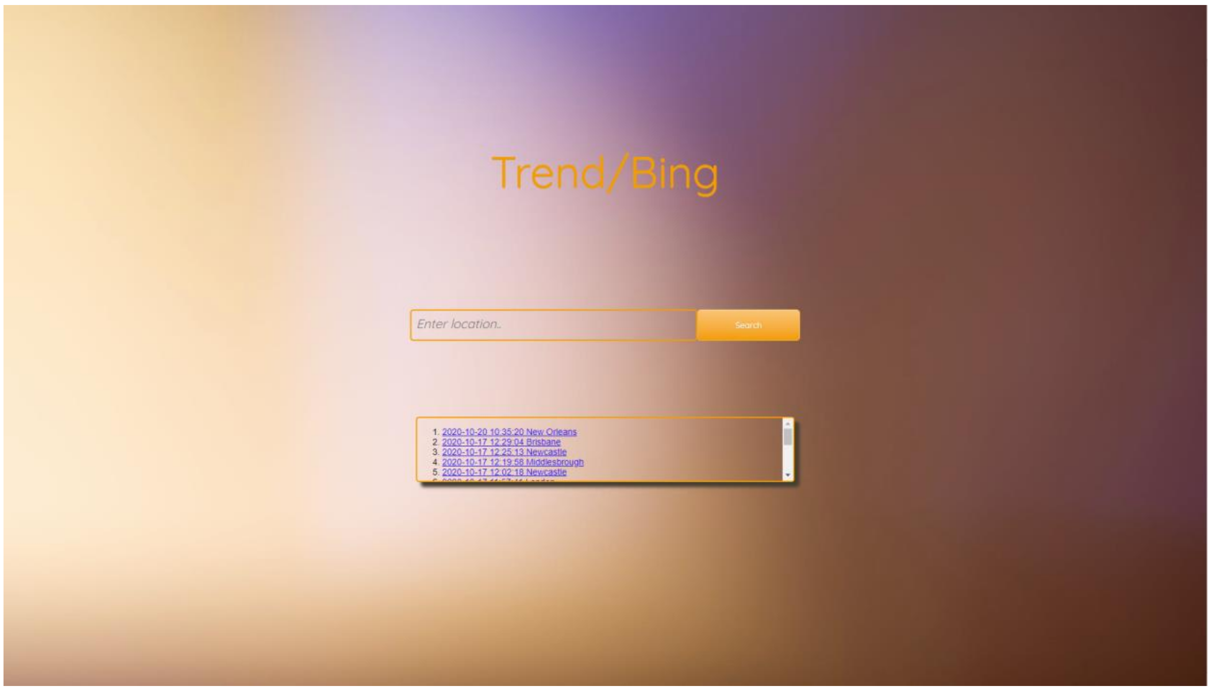
```
 [
  {
    "trends": [
      {
        "name": "#ChainedToTheRhythm",
        "url": "http://twitter.com/search?q=%23ChainedToTheRhythm",
        "promoted_content": null,
        "query": "%23ChainedToTheRhythm",
        "tweet_volume": 48857
      },
      {
        "name": "#اليوم_العالمي_للعتبان",
        "url": "http://twitter.com/search?
q=%23%D8%A7%D9%84%D9%8A%D9%88%D9%85_%D8%A7%D9%84%D8%B9%D8%A7%D9%84%D9%85%D9%8A_%D9%84%D9%8
4%D8%B9%D8%AA%D8%A8%D8%A7%D9%86",
        "promoted_content": null,
        "query":
"%23%D8%A7%D9%84%D9%8A%D9%88%D9%85_%D8%A7%D9%84%D8%B9%D8%A7%D9%84%D9%85%D9%8A_%D9%84%D9%84
%D8%B9%D8%AA%D8%A8%D8%A7%D9%86",
        "tweet_volume": null
      },
      {
        "name": "George Lopez",
        "url": "http://twitter.com/search?q=%22George+Lopez%22",
        "promoted_content": null,
        "query": "%22George+Lopez%22",
        "tweet_volume": 90590
      },
```

Appendix D – Dockerfile used to build the Docker image



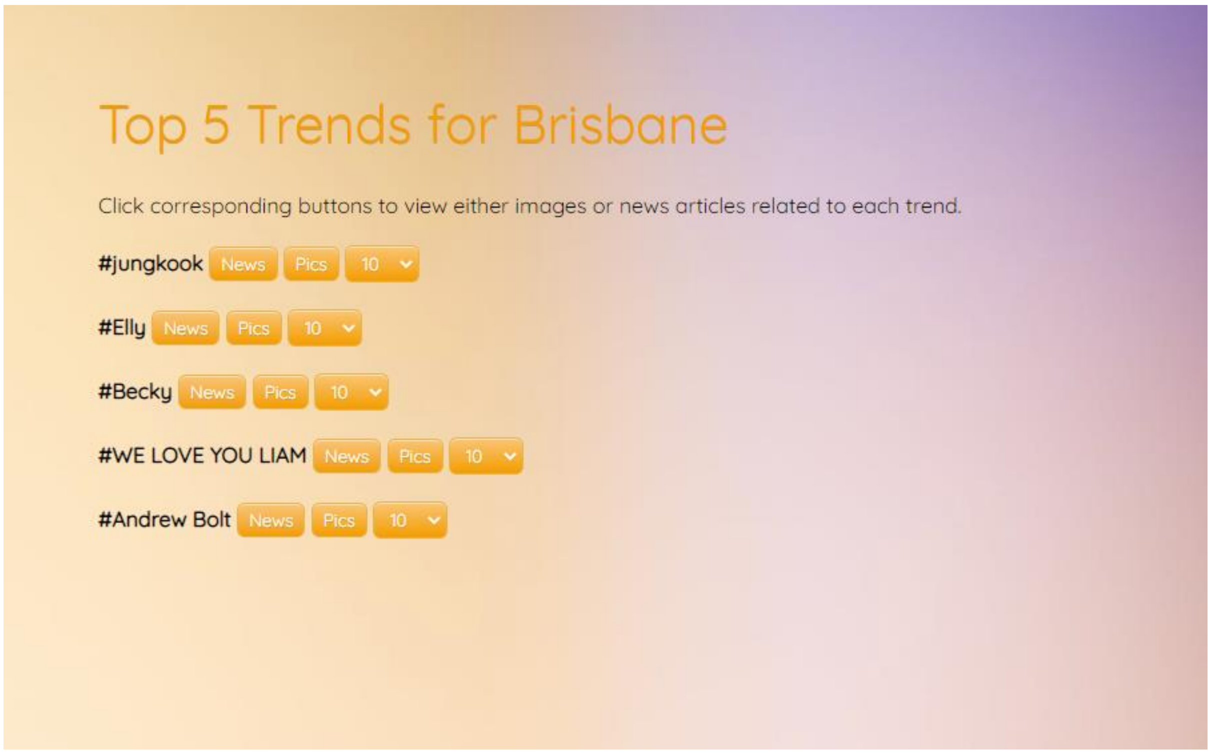
```dockerfile
# Set node version
FROM node:dubnium

# Copy app source
COPY . /src

# Set working directory to /src
WORKDIR /src

# Install app dependencies
RUN npm install

# Expose port to outside world
EXPOSE 3000

# Start command as per package.json
CMD ["npm", "start"]

```

Appendix E – Screenshot result/s of test #1



1. 2020-10-20 10:35:20 New Orleans
2. 2020-10-17 12:29:04 Brisbane
3. 2020-10-17 12:25:13 Newcastle
4. 2020-10-17 12:19:58 Middlesbrough
5. 2020-10-17 12:02:18 Newcastle

Appendix F – Screenshot result/s of test #2



Top 5 Trends for Brisbane

Click corresponding buttons to view either images or news articles related to each trend.

#jungkook News Pics 10

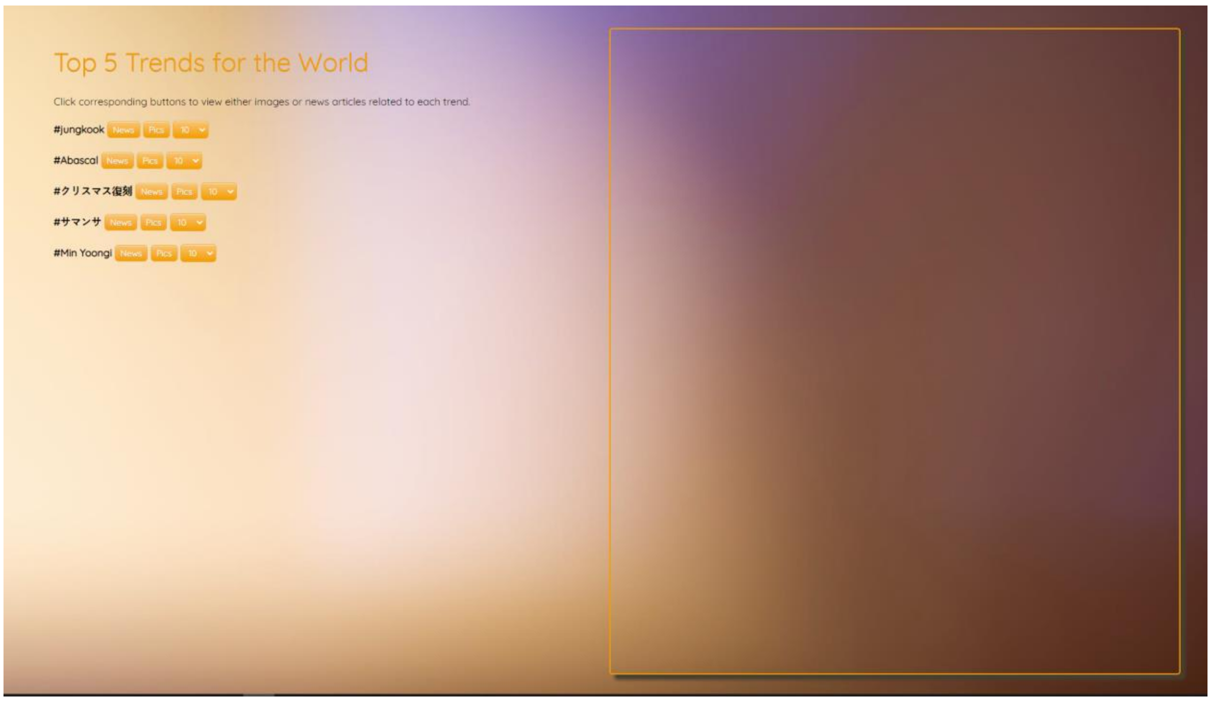
#Elly News Pics 10

#Becky News Pics 10

#WE LOVE YOU LIAM News Pics 10

#Andrew Bolt News Pics 10

Appendix G – Screenshot result/s of test #3



Top 5 Trends for the World

Click corresponding buttons to view either images or news articles related to each trend.

#jungkook News Pics 10

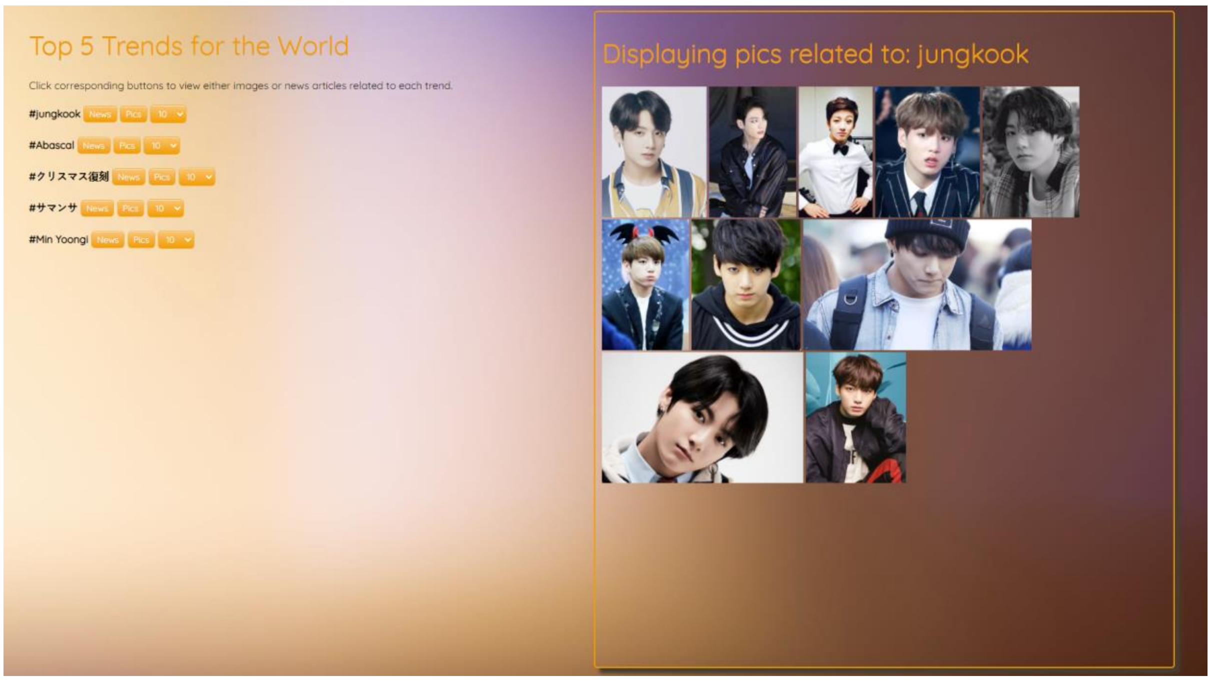
#Abascal News Pics 10

#クリスマス復刻 News Pics 10

#サマンサ News Pics 10

#Min Yoongi News Pics 10

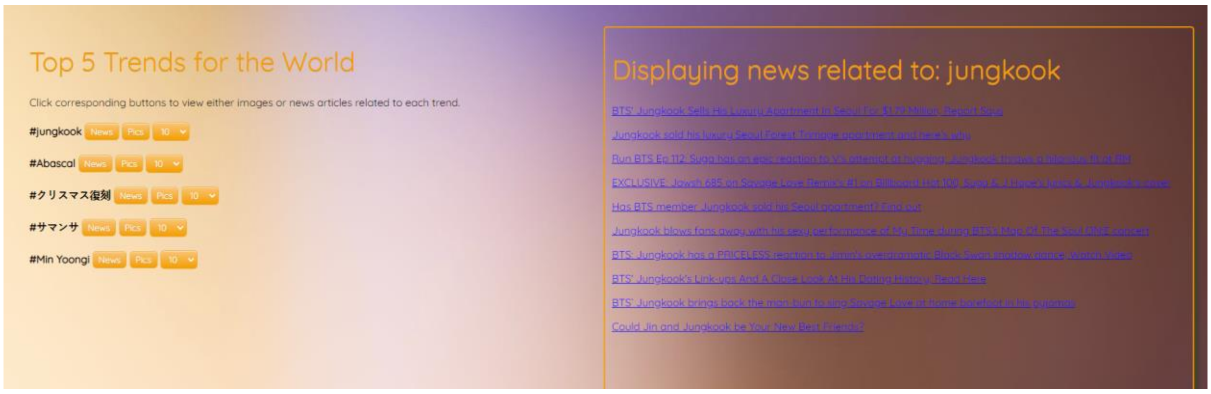
Appendix H – Screenshot result/s of test #4


Top 5 Trends for the World

Click corresponding buttons to view either images or news articles related to each trend.

Displaying pics related to: jungkook

Appendix I – Screenshot result/s of test #5



Top 5 Trends for the World

Click corresponding buttons to view either images or news articles related to each trend.

#jungkook News Pics 10

#Abascal News Pics 10

#クリスマス復刻 News Pics 10

#サマンサ News Pics 10

#Min Yoongi News Pics 10

Displaying news related to: jungkook

BTS' Jungkook Sells His Luxury Apartment In Seoul For $1.79 Million, Report Says

Jungkook sold his luxury Seoul Forest Trimage apartment and here's why

Run BTS Ep 112: Suga has an epic reaction to V's attempt at hugging; Jungkook throws a hilarious fit at RM

EXCLUSIVE: Jawsh 685 on Savage Love Remix's #1 on Billboard Hot 100, Suga & J-Hope's lyrics & Jungkook's cover

Has BTS member Jungkook sold his Seoul apartment? Find out

Jungkook blows fans away with his sexy performance of My Time during BTS's Map Of The Soul ONE concert

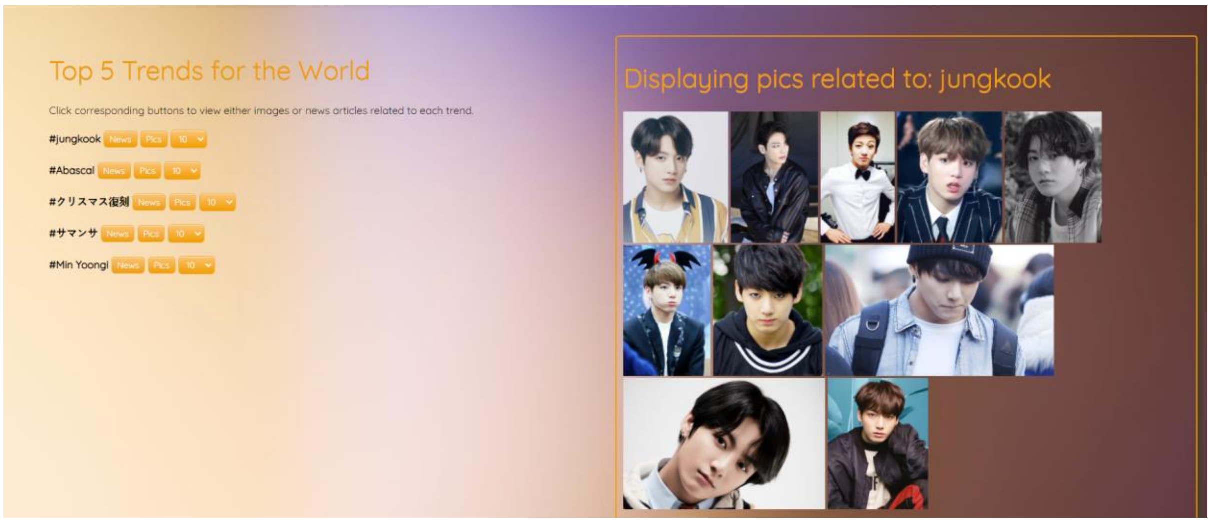
BTS: Jungkook has a PRICELESS reaction to Jimin's overdramatic Black Swan shadow dance; Watch Video

BTS' Jungkook's Link-ups And A Close Look At His Dating History; Read Here

BTS' Jungkook brings back the man-bun to sing Savage Love at home barefoot in his pyjamas

Could Jin and Jungkook be Your New Best Friends?

Pressing image button



Top 5 Trends for the World

Click corresponding buttons to view either images or news articles related to each trend.

#jungkook News Pics 10

#Abascal News Pics 10

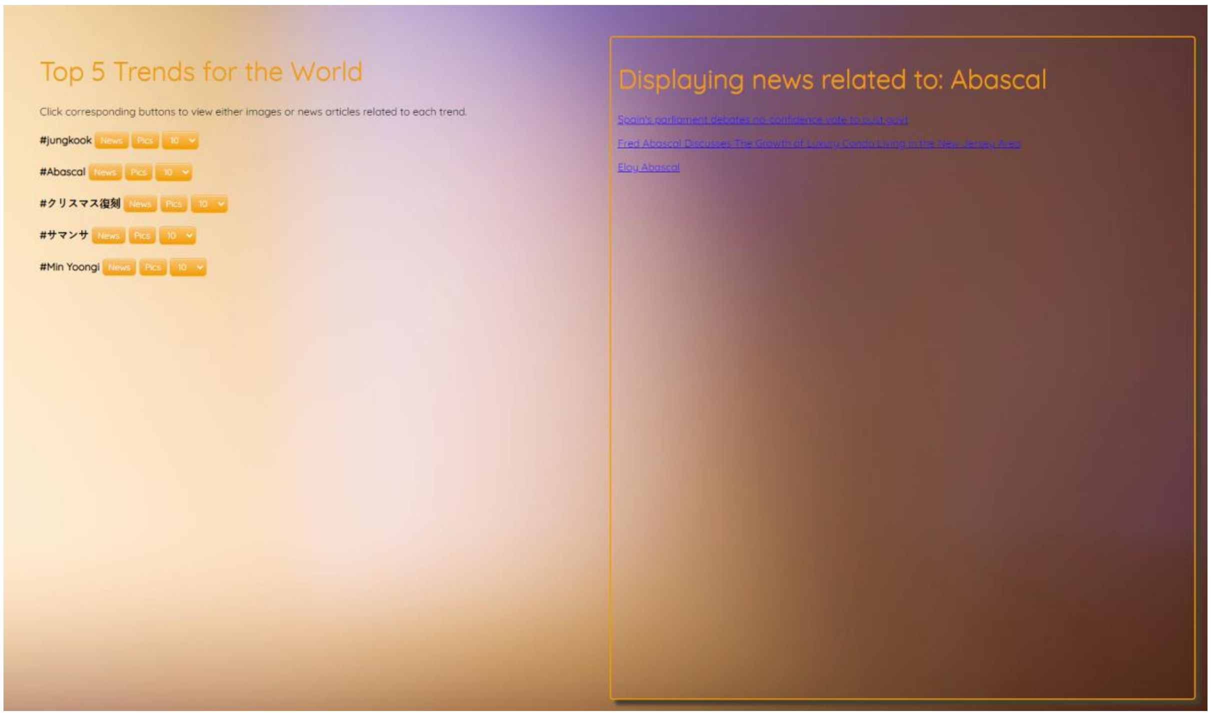
#クリスマス復刻 News Pics 10

#サマンサ News Pics 10

#Min Yoongi News Pics 10

Displaying pics related to: jungkook

Appendix J – Screenshot result/s of test #6



Top 5 Trends for the World

Click corresponding buttons to view either images or news articles related to each trend.

#jungkook
#Abascal
#クリスマス復刻
#サマンサ
#Min Yoongi

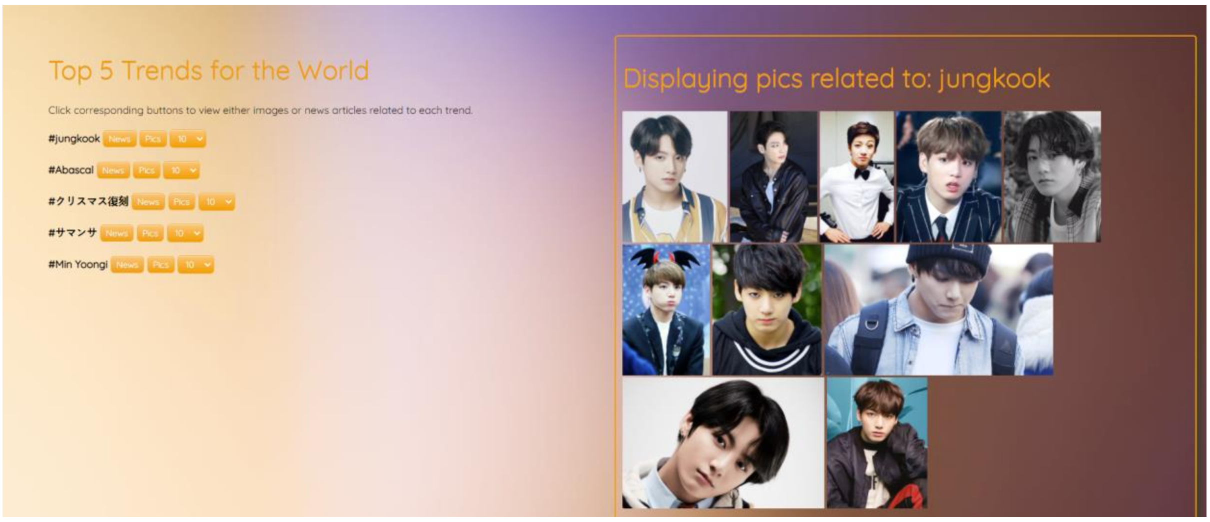
Displaying news related to: Abascal

Spain's parliament debates no-confidence vote to oust govt

Fred Abascal Discusses The Growth of Luxury Condo Living in the New Jersey Area

Eloy Abascal

Appendix K – Screenshot result/s of test #7



Top 5 Trends for the World

Click corresponding buttons to view either images or news articles related to each trend.

#jungkook News Pics 10

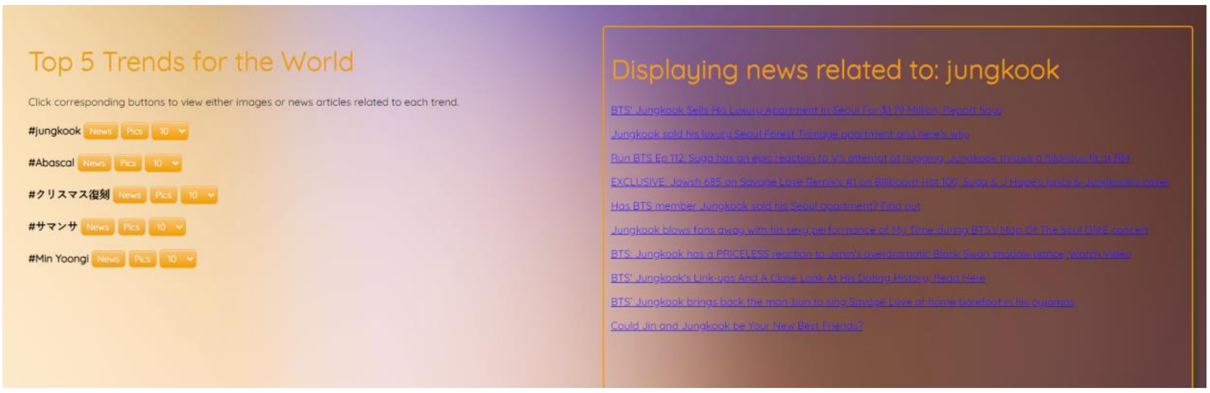
#Abascal News Pics 10

#クリスマス復刻 News Pics 10

#サマンサ News Pics 10

#Min Yoongi News Pics 10

Displaying pics related to: jungkook



Top 5 Trends for the World

Click corresponding buttons to view either images or news articles related to each trend.

#jungkook News Pics 10

#Abascal News Pics 10

#クリスマス復刻 News Pics 10

#サマンサ News Pics 10

#Min Yoongi News Pics 10

Displaying news related to: jungkook

BTS' Jungkook Sells His Luxury Apartment In Seoul For $1.79 Million: Report Says

Jungkook sold his luxury Seoul Forest Trimage apartment and here's why

Run BTS Ep 112: Suga has an epic reaction to V's attempt at hugging; Jungkook throws a hilarious fit at RM

EXCLUSIVE: Jawsh 685 on Savage Love Remix's #1 on Billboard Hot 100, Suga & J-Hope's lyrics & Jungkook's cover

Has BTS member Jungkook sold his Seoul apartment? Find out

Jungkook blows fans away with his sexy performance of My Time during BTS's Map Of The Soul ONE concert

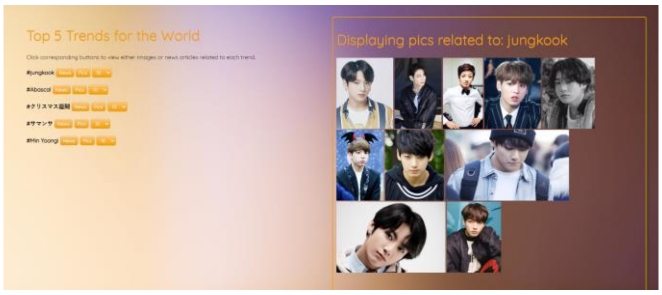
BTS: Jungkook has a PRICELESS reaction to Jimin's overdramatic Black Swan shadow dance; Watch Video

BTS' Jungkook's Link-ups And A Close Look At His Dating History, Read Here

BTS' Jungkook brings back the man-bun to sing Savage Love at home barefoot in his pyjamas

Could Jin and Jungkook be Your New Best Friends?

Appendix L – Screenshot result/s of test #8


Top 5 Trends for the World

Click corresponding buttons to view either images or news articles related to each trend.

#jungkook

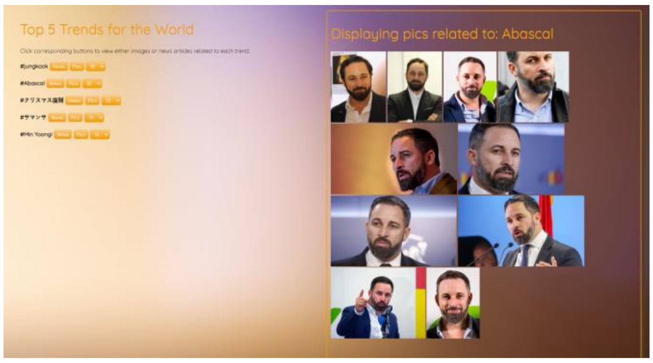
#Abascal

#クリスマス復刻

#サマンサ

#Min Yoongi

Displaying pics related to: jungkook


Top 5 Trends for the World

Click corresponding buttons to view either images or news articles related to each trend.

#jungkook

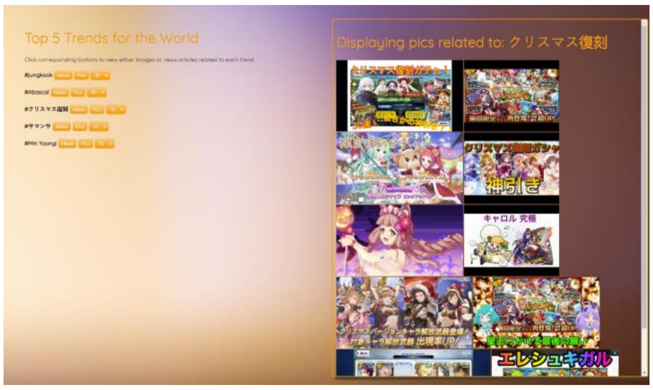
#Abascal

#クリスマス復刻

#サマンサ

#Min Yoongi

Displaying pics related to: Abascal


Top 5 Trends for the World

Click corresponding buttons to view either images or news articles related to each trend.

#jungkook

#Abascal

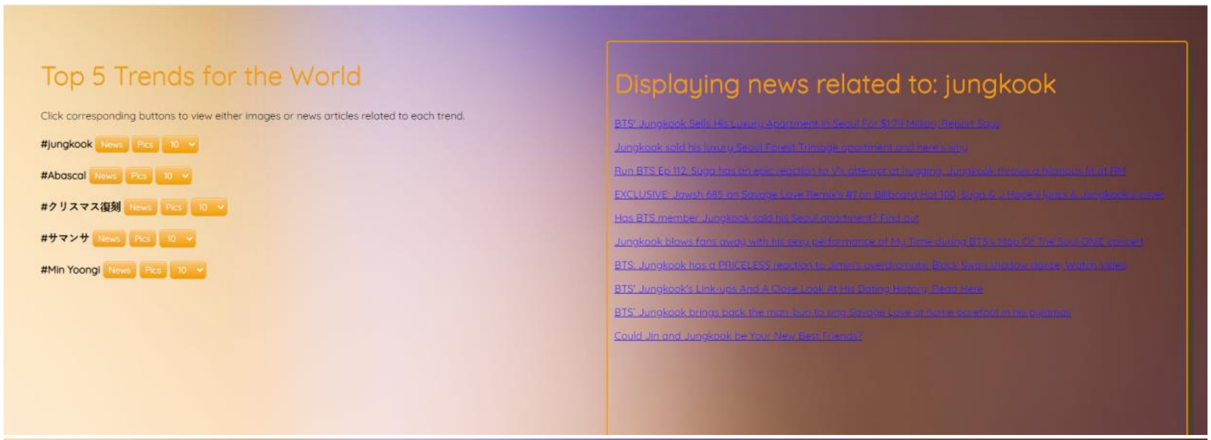
#クリスマス復刻

#サマンサ

#Min Yoongi

Displaying pics related to: クリスマス復刻

Appendix M – Screenshot result/s of test #9



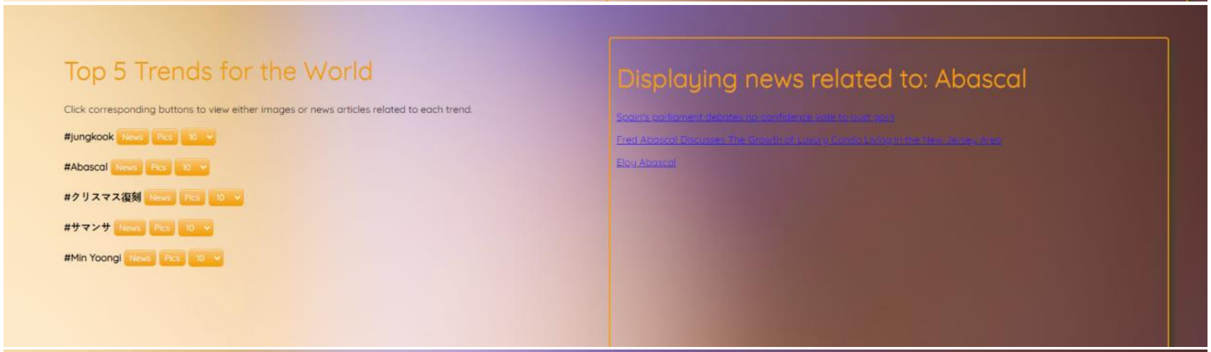
Top 5 Trends for the World

Click corresponding buttons to view either images or news articles related to each trend.

#jungkook
#Abascal
#クリスマス復刻
#サマンサ
#Min Yoongi

Displaying news related to: jungkook

BTS' Jungkook Sells His Luxury Apartment In Seoul For $1.79 Million: Report Says
Jungkook sold his luxury Seoul Forest Trimage apartment and here's why
Run BTS Ep 112: Suga has an epic reaction to V's attempt at hugging, Jungkook throws a hilarious fit at RM
EXCLUSIVE: Jawsh 685 on Savage Love Remix's #1 on Billboard Hot 100, Suga & J-Hope's lyrics & Jungkook's cover
Has BTS member Jungkook sold his Seoul apartment? Find out
Jungkook blows fans away with his sexy performance of My Time during BTS's Map Of The Soul ONE concert
BTS: Jungkook has a PRICELESS reaction to Jimin's overdramatic Black Swan shadow dance; Watch Video
BTS' Jungkook's Link-ups And A Close Look At His Dating History; Read Here
BTS' Jungkook brings back the man-bun to sing Savage Love at home barefoot in his pyjamas
Could Jin and Jungkook be Your New Best Friends?



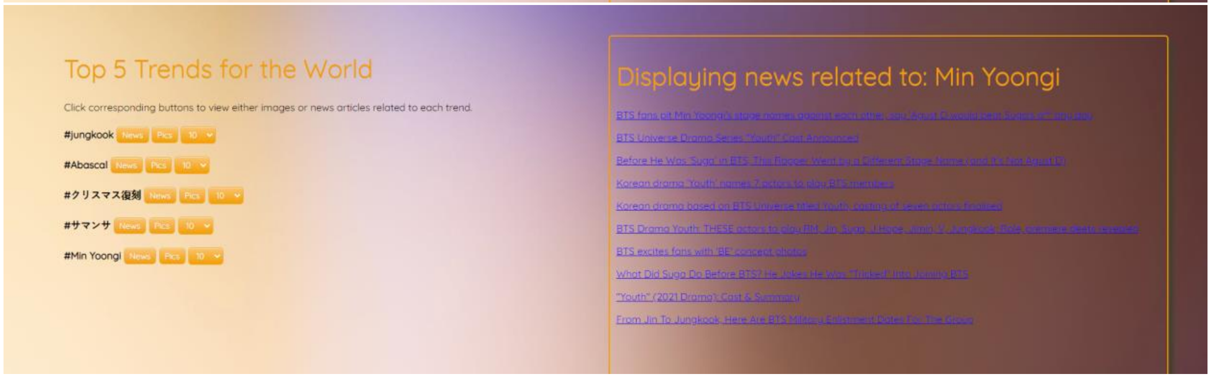
Top 5 Trends for the World

Click corresponding buttons to view either images or news articles related to each trend.

#jungkook
#Abascal
#クリスマス復刻
#サマンサ
#Min Yoongi

Displaying news related to: Abascal

Spain's parliament debates no-confidence vote to oust govt
Fred Abascal Discusses The Growth of Luxury Condo Living in the New Jersey Area
Eloy Abascal



Top 5 Trends for the World

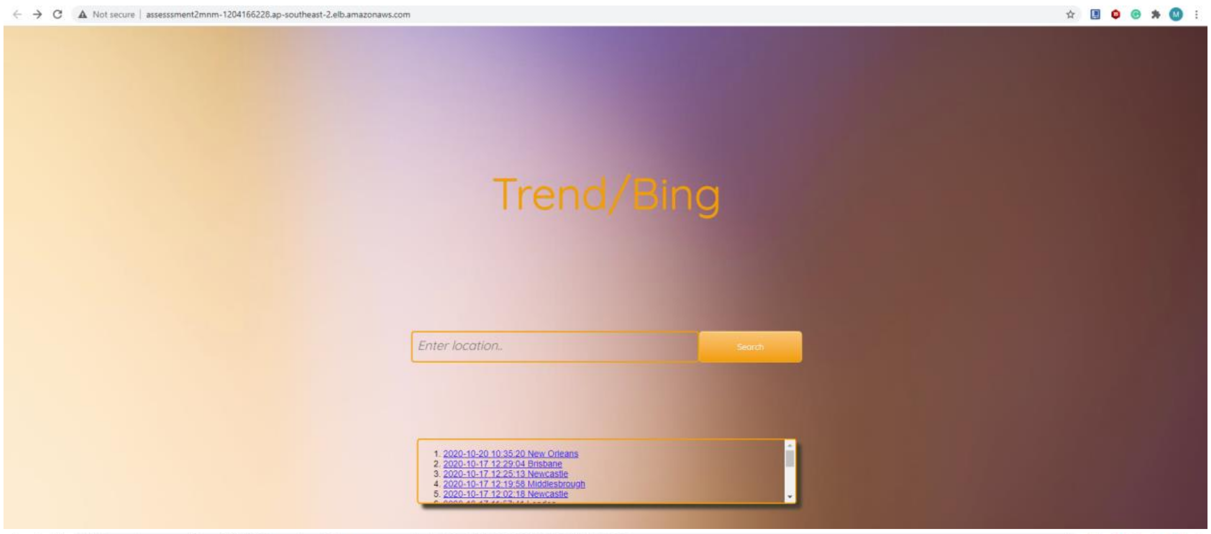
Click corresponding buttons to view either images or news articles related to each trend.

#jungkook
#Abascal
#クリスマス復刻
#サマンサ
#Min Yoongi

Displaying news related to: Min Yoongi

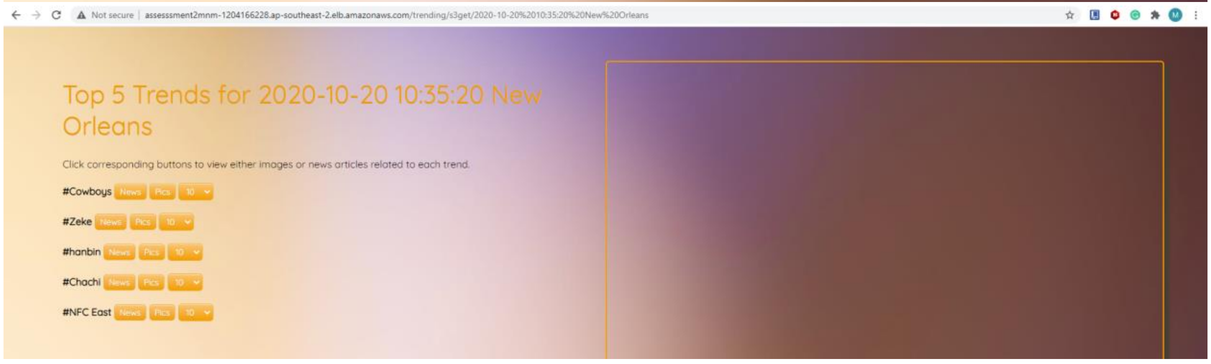
BTS fans pit Min Yoongi's stage names against each other, say 'Agust D would beat Suga's a** any day'
BTS Universe Drama Series "Youth" Cast Announced
Before He Was 'Suga' in BTS, This Rapper Went by a Different Stage Name (and It's Not Agust D)
Korean drama 'Youth' names 7 actors to play BTS members
Korean drama based on BTS Universe titled Youth, casting of seven actors finalised
BTS Drama Youth: THESE actors to play RM, Jin, Suga, J-Hope, Jimin, V, Jungkook; Role, premiere deets revealed
BTS excites fans with 'BE' concept photos
What Did Suga Do Before BTS? He Jokes He Was "Tricked" Into Joining BTS
"Youth" (2021 Drama): Cast & Summary
From Jin To Jungkook, Here Are BTS Military Enlistment Dates For The Group

Appendix N – Screenshot result/s of test #10



Not secure | assesssment2mnm-1204166228.ap-southeast-2.elb.amazonaws.com

1. 2020-10-20 10:35:20 New Orleans
2. 2020-10-17 12:29:04 Brisbane
3. 2020-10-17 12:25:13 Newcastle
4. 2020-10-17 12:19:58 Middlesbrough
5. 2020-10-17 12:02:18 Newcastle



Not secure | assesssment2mnm-1204166228.ap-southeast-2.elb.amazonaws.com/trending/s3get/2020-10-20%2010:35:20%20New%20Orleans

Top 5 Trends for 2020-10-20 10:35:20 New Orleans

Click corresponding buttons to view either images or news articles related to each trend.

#Cowboys News Pics 10

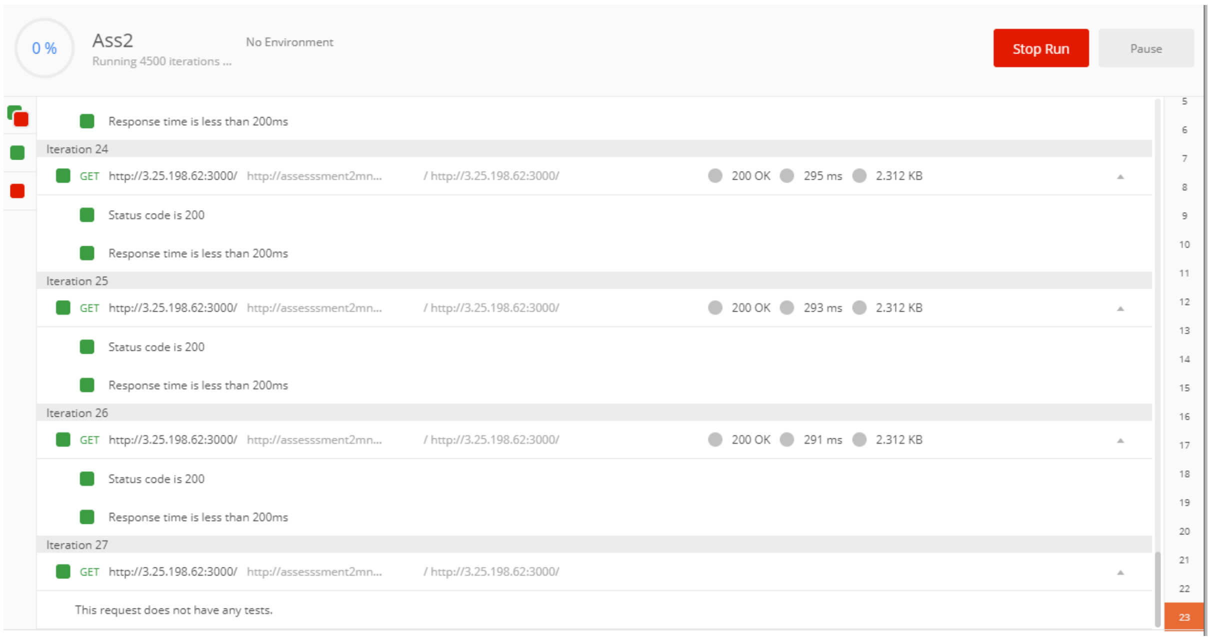
#Zeke News Pics 10

#hanbin News Pics 10

#Chachi News Pics 10

#NFC East News Pics 10

Appendix O – Screenshot result/s of test #11



0 %
Ass2
Running 4500 iterations ...
No Environment
Stop Run
Pause

🟩 Response time is less than 200ms

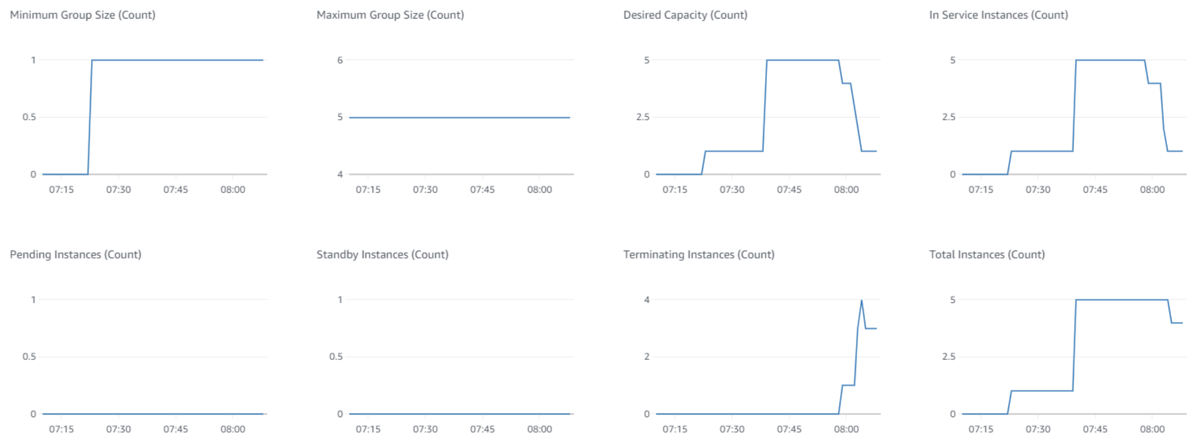
Iteration 24
🟩 GET http://3.25.198.62:3000/ http://assesssment2mn... / http://3.25.198.62:3000/ ● 200 OK ● 295 ms ● 2.312 KB
🟩 Status code is 200
🟩 Response time is less than 200ms

Iteration 25
🟩 GET http://3.25.198.62:3000/ http://assesssment2mn... / http://3.25.198.62:3000/ ● 200 OK ● 293 ms ● 2.312 KB
🟩 Status code is 200
🟩 Response time is less than 200ms

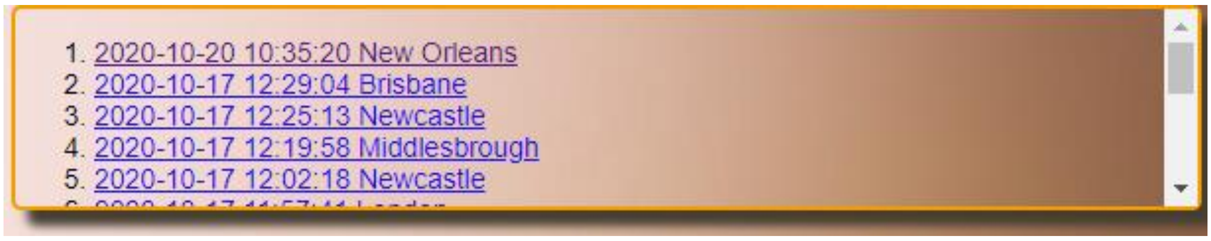
Iteration 26
🟩 GET http://3.25.198.62:3000/ http://assesssment2mn... / http://3.25.198.62:3000/ ● 200 OK ● 291 ms ● 2.312 KB
🟩 Status code is 200
🟩 Response time is less than 200ms

Iteration 27
🟩 GET http://3.25.198.62:3000/ http://assesssment2mn... / http://3.25.198.62:3000/
This request does not have any tests.

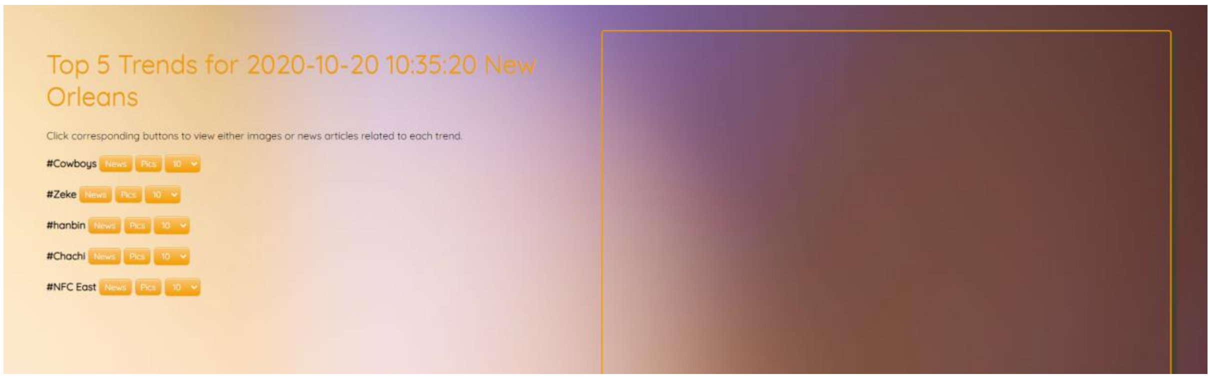
5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23



Appendix P – Screenshot result/s of test #12



1. 2020-10-20 10:35:20 New Orleans
2. 2020-10-17 12:29:04 Brisbane
3. 2020-10-17 12:25:13 Newcastle
4. 2020-10-17 12:19:58 Middlesbrough
5. 2020-10-17 12:02:18 Newcastle

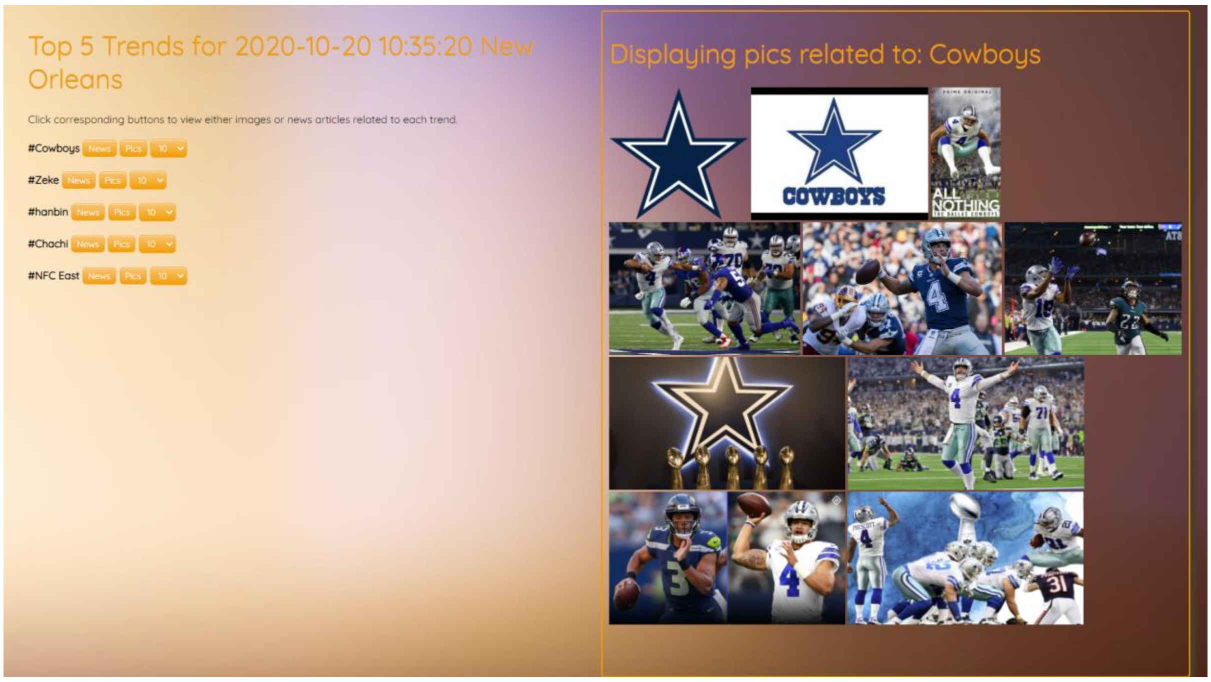


Top 5 Trends for 2020-10-20 10:35:20 New Orleans

Click corresponding buttons to view either images or news articles related to each trend.

#Cowboys
#Zeke
#hanbin
#Chachi
#NFC East

Appendix Q – Screenshot result/s of test #13



Top 5 Trends for 2020-10-20 10:35:20 New Orleans

Click corresponding buttons to view either images or news articles related to each trend.

#Cowboys News Pics 10

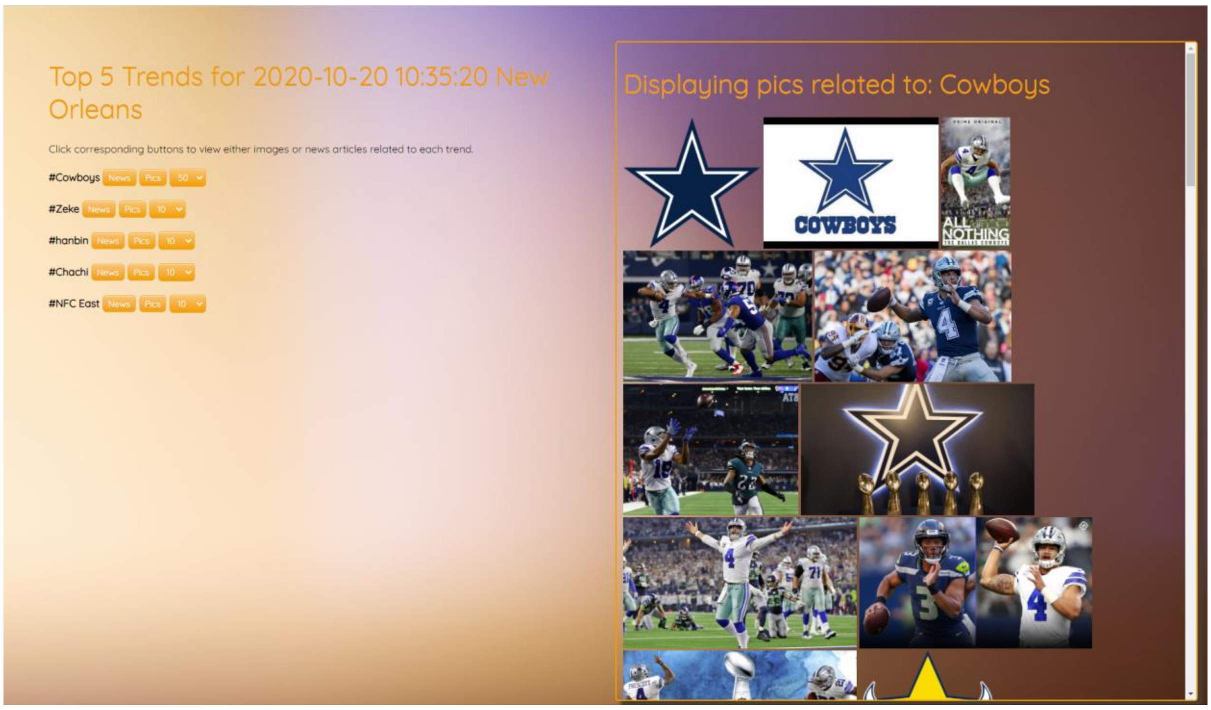
#Zeke News Pics 10

#hanbin News Pics 10

#Chachi News Pics 10

#NFC East News Pics 10

Displaying pics related to: Cowboys



Top 5 Trends for 2020-10-20 10:35:20 New Orleans

Click corresponding buttons to view either images or news articles related to each trend.

#Cowboys News Pics 50

#Zeke News Pics 10

#hanbin News Pics 10

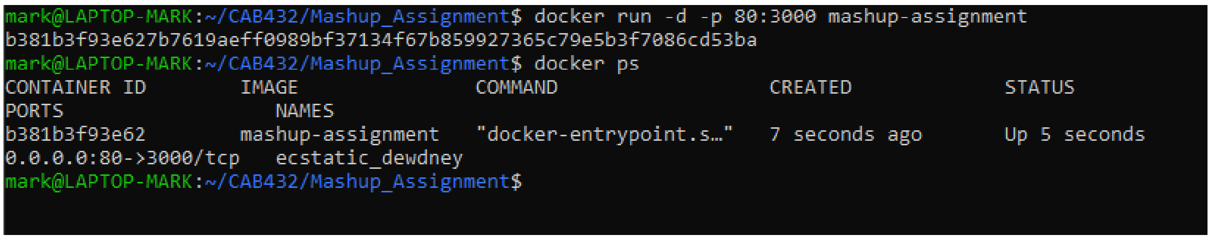
#Chachi News Pics 10

#NFC East News Pics 10

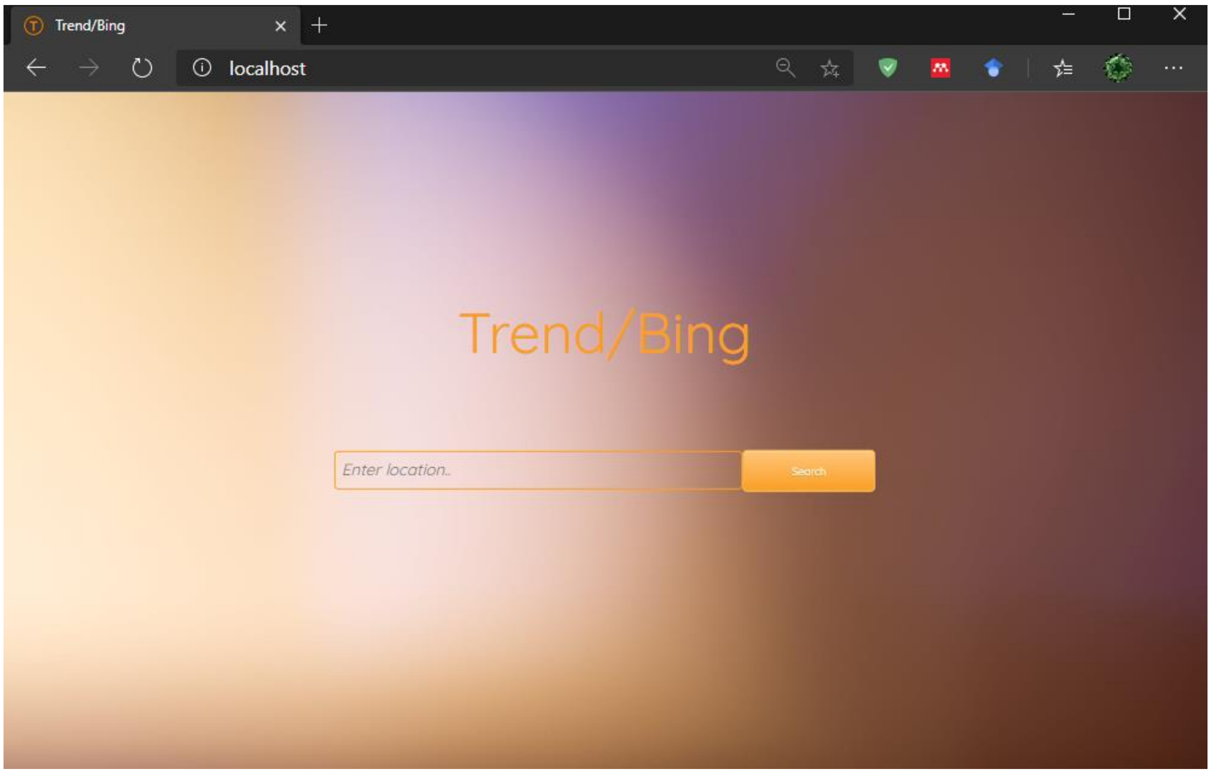
Displaying pics related to: Cowboys

Hard to see in photos but test passes

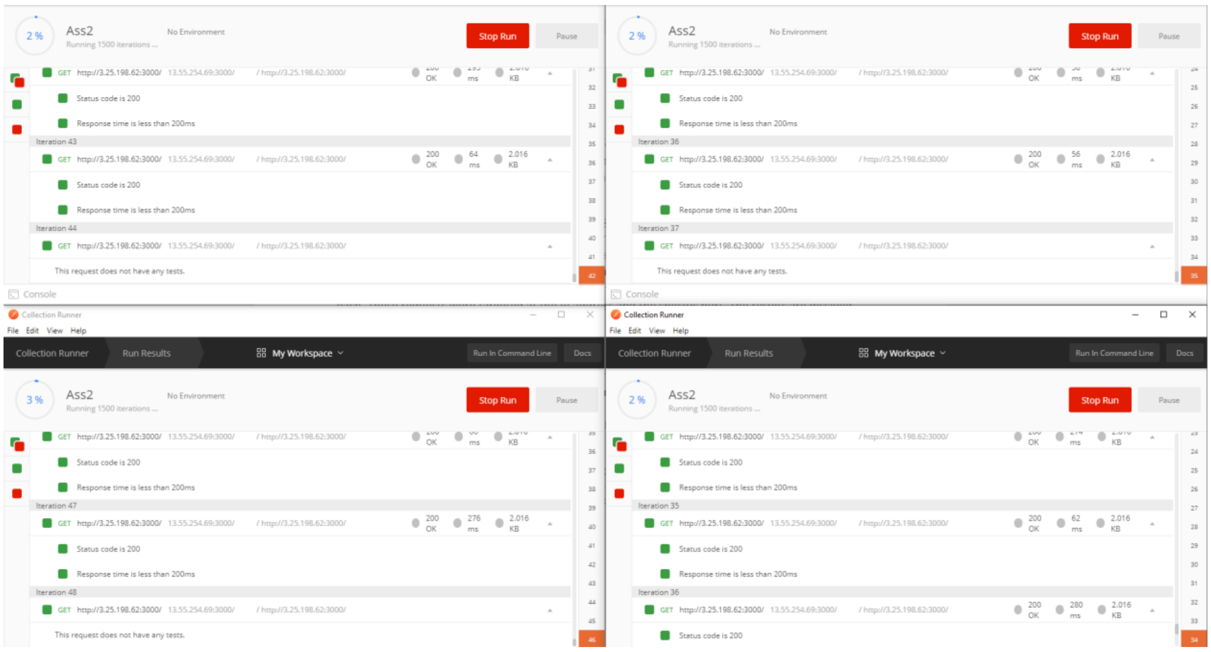
Appendix R – Screenshot result/s of test #14



```
mark@LAPTOP-MARK:~/CAB432/Mashup_Assignment$ docker run -d -p 80:3000 mashup-assignment
b381b3f93e627b7619aeff0989bf37134f67b859927365c79e5b3f7086cd53ba
mark@LAPTOP-MARK:~/CAB432/Mashup_Assignment$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
PORTS                  NAMES
b381b3f93e62        mashup-assignment   "docker-entrypoint.s…"   7 seconds ago       Up 5 seconds
0.0.0.0:80->3000/tcp   ecstatic_dewdney
mark@LAPTOP-MARK:~/CAB432/Mashup_Assignment$
```



Appendix S - Running postman to test scaling



```
2 %   Ass2                                   No Environment        Stop Run   Pause
      Running 1500 iterations ...
GET  http://3.25.198.62:3000/  13.55.254.69:3000/  / http://3.25.198.62:3000/   OK   ms   KB
     Status code is 200
     Response time is less than 200ms
Iteration 43
GET  http://3.25.198.62:3000/  13.55.254.69:3000/  / http://3.25.198.62:3000/   200 OK   64 ms   2.016 KB
     Status code is 200
     Response time is less than 200ms
Iteration 44
GET  http://3.25.198.62:3000/  13.55.254.69:3000/  / http://3.25.198.62:3000/
     This request does not have any tests.
Console
```

```
2 %   Ass2                                   No Environment        Stop Run   Pause
      Running 1500 iterations ...
GET  http://3.25.198.62:3000/  13.55.254.69:3000/  / http://3.25.198.62:3000/   OK   ms   KB
     Status code is 200
     Response time is less than 200ms
Iteration 36
GET  http://3.25.198.62:3000/  13.55.254.69:3000/  / http://3.25.198.62:3000/   200 OK   56 ms   2.016 KB
     Status code is 200
     Response time is less than 200ms
Iteration 37
GET  http://3.25.198.62:3000/  13.55.254.69:3000/  / http://3.25.198.62:3000/
     This request does not have any tests.
Console
```

```
Collection Runner
File  Edit  View  Help
Collection Runner   Run Results        My Workspace        Run In Command Line   Docs
3 %   Ass2                                   No Environment        Stop Run   Pause
      Running 1500 iterations ...
GET  http://3.25.198.62:3000/  13.55.254.69:3000/  / http://3.25.198.62:3000/   OK   ms   KB
     Status code is 200
     Response time is less than 200ms
Iteration 47
GET  http://3.25.198.62:3000/  13.55.254.69:3000/  / http://3.25.198.62:3000/   200 OK   276 ms   2.016 KB
     Status code is 200
     Response time is less than 200ms
Iteration 48
GET  http://3.25.198.62:3000/  13.55.254.69:3000/  / http://3.25.198.62:3000/
     This request does not have any tests.
```

```
Collection Runner
File  Edit  View  Help
Collection Runner   Run Results        My Workspace        Run In Command Line   Docs
2 %   Ass2                                   No Environment        Stop Run   Pause
      Running 1500 iterations ...
GET  http://3.25.198.62:3000/  13.55.254.69:3000/  / http://3.25.198.62:3000/   OK   ms   KB
     Status code is 200
     Response time is less than 200ms
Iteration 35
GET  http://3.25.198.62:3000/  13.55.254.69:3000/  / http://3.25.198.62:3000/   200 OK   62 ms   2.016 KB
     Status code is 200
     Response time is less than 200ms
Iteration 36
GET  http://3.25.198.62:3000/  13.55.254.69:3000/  / http://3.25.198.62:3000/   200 OK   280 ms   2.016 KB
     Status code is 200
```