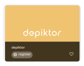
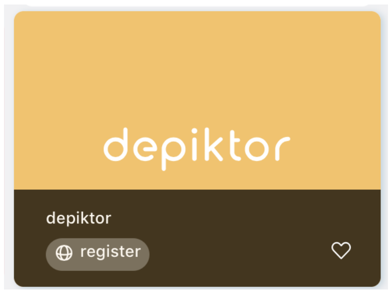
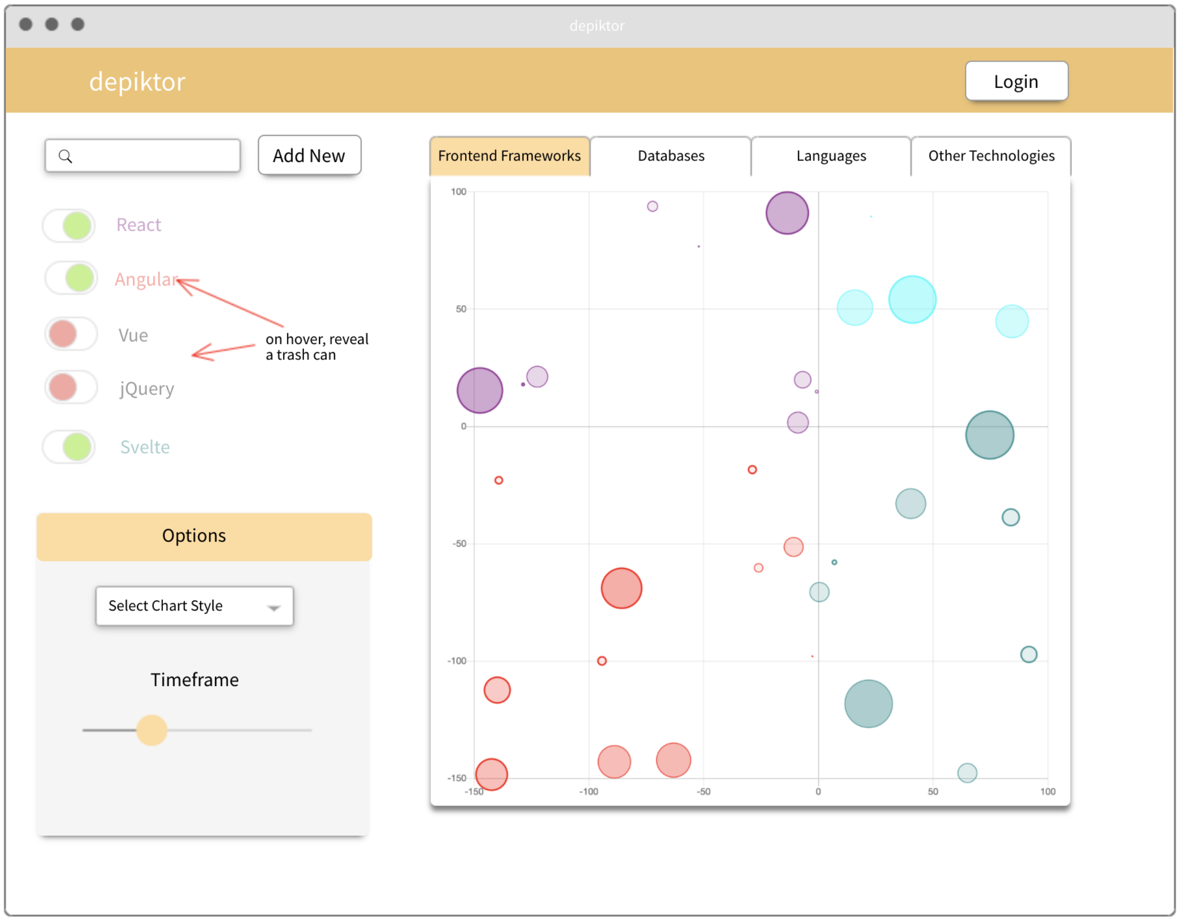


# Depiktor

## Branding



## UI ⇒ desktop view



Nav color #f5b64e

Lighter orange #fed68c

Light grey #f1f1f1

Text color #4b4c4c

Box outline color #aaa

Box shadow:hover rgba(66, 97, 255, 0.5)

### UI ⇒ mobile screen view

TODO: make mobile screen view

### Front End

Components

- Nav ⇒ presentational component that holds the login button
- List
  - ListItem ⇒ when the modal is toggled on/off the current chart us updated with the data added or removed
  - no API calls for this
- Charts ⇒ holds a list of charts and the content for the active tab
  - Chart ⇒ displays the name of the chart and adds a additional class if chart is active, when clicked fires a handler which lets Charts know which tab is active
- Options ⇒ holds the Chart Style & Timeframe component
  - Chart Style ⇒ when new chart is selected, data is transformed to new format and passed to current chart to update
  - Timeframe ⇒ update the data to add or remove data points
  - no API calls for this

When logged in

- AddNew ⇒ allows to add a new item to the list
  - Make a GET request to get a new dataset
  - background worker collects the data and puts it in the db, db sends it over to client to display
- Chart ⇒ allows a new chart to be added to create a new category

### TODO:

- [ ] change animation on tabs list
- [ ] add colors to the charts and corresponding colors to the toggle labels
- [ ] change the highlight on the dropdown to match color scheme
- [ ] add search bar and button

From API getting an array of data points

- depending on the chart style, the data array needs to be in a different format

- [ ] make a new GET request per chart style

-

### Reactive Components

#### Initial State

- [ ] GET req should pass data to
    - [x] ~~Tabs ⇒ create a TabList from mockData.datasets.label~~
    - [x] ~~TabPanel ⇒ create a list of tab panels~~
        - [x] ~~create one chart per data object~~
    - [x] ~~ToggleList (make component) ⇒ create a list of Toggle components with mockData.datasets.label~~
        - [ ] display list of Toggle components based on current TabPanel
    - [x] ~~Select ⇒ pass data labels to select~~

- [ ] Toggle ⇒ off on current tab should update data in TabPanel ⇒ Graph
- [ ] Select ⇒ select should update the chart style in the current tab

## Backend

### Data

- I want to continuously sample the Twitter API every 15 mins ⇒ can sample about 50 times per min, and provides approximately 1% of all tweets
- for every tweet text see if they contain mention of one of the libraries, languages or frameworks on the StackOverflow 2020 developer survey
    - don't use raw data, compile a CSV of popular things to query
- if a tweet contains a mention, add it to the counter
- therefore, data will just be a table
- if a user is logged in, they should be able to add or remove their own technology
    - that means, if a relational db, each user will have a foreign key to technologies
- for one table of technologies with their count and technology type (framework, language etc)
- how can I make it into a time frame?
    - do I show info collected over the last seven days
        - automatically discard info from db that's over 7 days old ⇒ or move to a different storage if you want to expand your dataset later
        - this means I have to keep track of days ⇒ on the table have mentions per day
    - do I show only info for
- sanitize data with NLP ⇒ how can I distinguish real mentions of the language or framework vs other things?
- use setTimeout to schedule adding data to the db

### Requirements

- [x] ~~Get Twitter realtime data~~
    - [x] ~~figure out how API works~~
    - [x] ~~write script to get API data~~
    - [x] 
- [x] ~~Parse data to search for technology mentions~~
- [x] ~~Store data in DB~~
    - [x] ~~make data available through controller to front end ⇒ tie two together~~
    - [ ] use ENV variables for the db connection
        - [ ] make a dotenv example for when you push to github if anyone wants to use the app
    - [ ] data stored are regular intervals, days hours etc. continuous background fetching

- [x] ~~Display data in graph format at '/'~~
    - [ ] load in the beginning
- [x] ~~Able to toggle technologies on and off from graph in same view~~
- [x] ~~change chart style ⇒ bubble vs bar graph etc.~~
- [x] ~~toggle to different technology categories in the different view~~

BM Comments:

- [ ] Add details to Requirements, potentially as sub-points
    - [ ] Break down into very small steps, e.g. "count results for a single term"
    - [ ] If doing it as a background job, describe how going to split the work between executions
- [ ] Write up DB structure - tables, columns, associations, data types
- [ ] If want to make the data parsing a background job, add that as a separate task

## Stack

- server ⇒ express
    - db ⇒ postgreSQL
    - orm ⇒ sequelize
- background process ⇒ twitter recent search vv 2 https://developer.twitter.com/en/docs/labs/recent-search/api-reference/get-recent-search with setTimeout
    - this needs to be async process
    - what will sanitize the data first?
    - then write data to db
- client ⇒ react
    - graph ⇒ react-chartjs-2 https://www.npmjs.com/package/react-chartjs-2
        - based on chart-js https://github.com/chartjs/awesome
    - CSS ⇒ vanilla or bootstrap
    - tabs ⇒ https://github.com/reactjs/react-tabs

- dropdown ⇒ https://react-select.com/styles
- toggle ⇒ https://aaronshaf.github.io/react-toggle/

## Description

I want to take the terms in the StackOverflow 2020 Developer Survey under the Technologies section (https://insights.stackoverflow.com/survey/2020#technology) and see how many of those terms show up in a random sample of tweets over the period of a week. I want to create a database where I have for example

**Technologies**

| id | tech | category |
|---|---|---|
| 2 | React | frontend |
| 5 | Angular | frontend |
| 7 | JavaScript | frontend |
| | | Untitled |

**Counts**

| id | total | technologies foreign key |
|---|---|---|
| 1 | 1873 | 2 |
| 2 | 8353 | 5 |
| 3 | 7263 | 7 |

Then, in the client I would like to show a graph that will show the table above with the ability for the user to toggle, from a list, the datasets added to the graph.

## Approach

Given that there is a set of known terms that you are starting from, there are two main approaches to consider:

1. Stream tweets:
    1. then filter them yourself (sampled stream)
    2. that have been filtered by Twitter (filtered stream)
2. Search tweets directly (recent search)

https://developer.twitter.com/en/docs/labs/recent-search/api-reference/get-recent-search

## Using the Twitter Recent Search API

- make an array of search terms that I would like to query with
- the endpoint receives a single search query and
    - build a search query using operators that match Tweet attributes (basically filter them down more) ⇒
- use the "get historical" default behavior that returns all matching tweets for the last 7 days
- use the below code to make requests, modify so that

- load credential tokens from another sources (env file?)
- how to use pagination to collect all matching tweets? Use next_token and loop until no next_token in response
- make a request & update the data when
  - user loads '/'
  - user adds new search term ⇒ make a fetch request and display the data

### Request

```
const https = require('https');
const request = require('request');
const util = require('util');

const get = util.promisify(request.get);
const post = util.promisify(request.post);

const consumer_key = ''; // Add your API key here
const consumer_secret = ''; // Add your API secret key here

const bearerTokenURL = new URL('https://api.twitter.com/oauth2/token');
const searchURL = new URL('https://api.twitter.com/labs/2/tweets/search');

async function bearerToken (auth) {
  const requestConfig = {
    url: bearerTokenURL,
    auth: {
      user: consumer_key,
      pass: consumer_secret,
    },
    form: {
      grant_type: 'client_credentials',
    },
  };

  const response = await post(requestConfig);
  return JSON.parse(response.body).access_token;
}

(async () => {
  let token;
  const query = 'from:twitterdev has:media'; //add the query parameters here
  const maxResults = 10;

  try {
    // Exchange your credentials for a Bearer token
    token = await bearerToken({consumer_key, consumer_secret});
  } catch (e) {
    console.error(`Could not generate a Bearer token. Please check that your credentials are correct and that the Filtered Stream pr
    process.exit(-1);
  }

  const requestConfig = {
    url: searchURL,
    qs: {
      query: query,
      max_results: maxResults,
    },
    auth: {
      bearer: token,
    },
    headers: {
      'User-Agent': 'LabsRecentSearchQuickStartJS',
    },
    json: true,
  };

  try {
    const res = await get(requestConfig);
    console.log(res.statusCode);
    console.log(res);
    if (res.statusCode !== 200) {
      throw new Error(res.json);
      return;
    }
    console.log(res.json);
  } catch (e) {
    console.error(`Could not get search results. An error occurred: ${e}`);
    process.exit(-1);
  }
})();
```

### Response

```
{
  "data": [
    {
      "author_id": "2244994945",
      "created_at": "2020-02-12T17:09:56.000Z",
      "id": "1227640996038684673",
      "lang": "en",
      "text": "Doctors: Googling stuff online does not make you a doctor\n\nDevelopers: https://t.co/mrju5ypPkb"
    },
    {
      "author_id": "2244994945",
      "created_at": "2020-02-07T22:48:32.000Z",
      "id": "1225914265946730497",
      "lang": "en",
      "text": "RT @usman4all: Join us tomorrow for #TapIntoTwitterLafia Q1 Meetup of the year tomorrow. #TapIntoTwitter\nCc: @Twitte
    }
  ],
  "meta": {
    "newest_id": "1227640996038684673",
    "oldest_id": "1225914265946730497",
    "result_count": 2
  }
}
```

- NB: use this example to build the front-end while waiting for the authentication

### Search Query

```
const query = 'from:twitterdev has:media'; //look for this in the query example above
```

- make this query dynamic with variables from the technology variables array I will create
- I get 225 requests per 15-minute window ⇒ does that include "pagination" or not? NO
  - means I have to have the data stored before the user hits '/' otherwise I won't be able to get all of the data when they come to the route, since all this will have to run async
- With the search parameters, just needs to store the result_count in my db for every query I make since I already know it's been filtered

### Database connection

### Background Process

- refresh data once a day at 12:01 am ⇒ continue making the requests until all the data has been updated
- Create with Bull package which uses Redis https://github.com/OptimalBits/bull/

- get data every hour since numbers too large for every day ⇒ schedule job to run every hour

- make scale for the last 24 hrs
- make timescale 1 hour ⇒ update data accordingly

## Resources

### Authentication

use https://medium.com/@robince885/how-to-do-twitter-authentication-with-react-and-restful-api-e525f30c62bb

with https://www.npmjs.com/package/react-twitter-auth component

### Tabs Styling

https://codepen.io/tutsplus/pen/VLeXqy

### Slider Styling

https://codesandbox.io/s/nn6vb?file=/src/index.js:1935-1941

### Generate Random Pastel Colors

https://stackoverflow.com/questions/43193341/how-to-generate-random-pastel-or-brighter-color-in-javascript#:~:text=An example (using JQuery)%3A,%24("body").

### Example Charts

https://tobiasahlin.com/blog/chartjs-charts-to-get-you-started/

https://embed.plnkr.co/JOI1fpgWIS0lvTeLUxUp/

### Pg-hstore

https://www.npmjs.com/package/pg-hstore

## APIs

### Twitter Recent Search v2

https://developer.twitter.com/en/docs/labs/recent-search/overview

### Search Tweets

https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets