General

I built an web app, which searches imgur.com for random top 100 images corresponding to the query you type and shows a random one. It is hosted at http://2id60.win.tue.nl/~s141426. I have tested the website on `Chromium 62.0.3202.94` and `Mozilla Firefox 57.0.2`.

Design

The app is built mostly using React.I used Bootstrap library for React to design the website and also FontAwesome for icons at the footer. Bootstrap helped make the website responsive and better-looking. I some custom CSS for the app, i.e. I made `<main>` element respect size of header and footer, added colors to divide sections and impove the looks of the website. I have added an outline button, which changes color, once it's hovered upon. See screenshots at the end of the document.

Interaction

Once the user types in the search bar, requests is sent to the API to query for the user specified input. There is a small(50ms) delay between last search bar content change and executing the request to avoid sending requests while user is still typing the queries. The app initally searches for 'hello AND world', to avoid blank image on page load. If no results are found, a "image-not-found" placeholder is shown. To workaround the issue with CORS not being supported by Imgur API I have developed a small backend in Go, which is also running on university server. It is available on port `8042` and provides one REST endpoint `/data`, which takes a `q` parameter and passes that to Imgur API. This is then called from front-end using the `Fetch` API. `jQuery` was not required given the capabilities of `React` and `Fetch`.

Backend

I used Go for the backend. No framework is required, given a very good standard library. I used InfluxDB database for analytics(storing the queries) and embedded BoltDB for storing user data.

Databases

I used InfluxDB database for analytics(storing the queries) and embedded BoltDB for storing user data. The user can register/login and specify his favorite query, for this data I use BoltDB(login,password,query). InfluxDB stores the interaction log(time,duration of query,endpint,query,user,address)

API

- `/image` returns a JSON array of JSON-encoded images as defined by Imgur API. Queries to this endpoint must be of "GET" method and should include a `q` parameter, which specifies the search query. Once a request to this endpoint is received, the specified search query is sent to Imgur and the result(if successful) is sent to the client. More info can be found at https://apidocs.imgur.com/#
- `/register` returns a JSON-encoded user object. Queries to this endpoint must be of "POST" method and should include JSON-encoded user object in the request body. This endpoint is used to register a user. Length of password and login should be at least 2. input :

```
{
'login': ...,
'password': ...,
'query': ...
}
```

output :

```
{
    'login': ...,
    'password': ...,
    'query': ...
}
```

- `/login` returns a JSON-encoded user object. Queries to this endpoint must be of "POST" method and should include JSON-encoded user object in the request body.

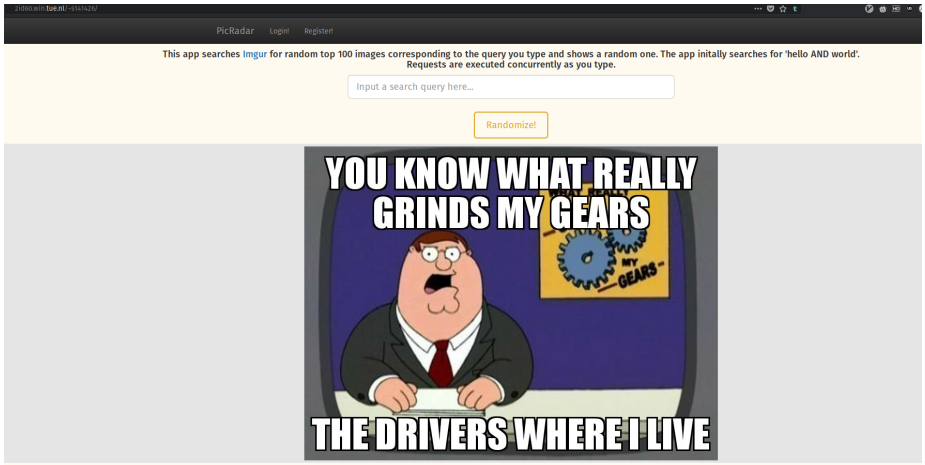
input :

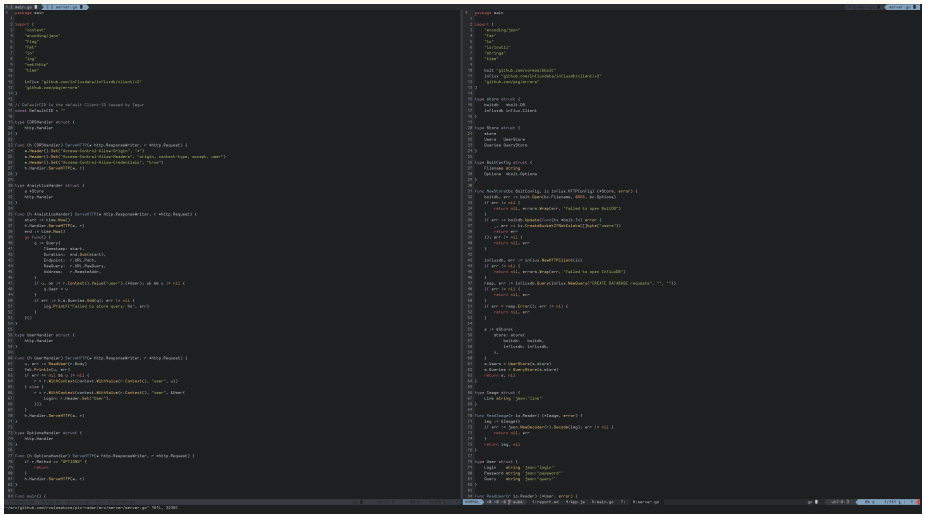
```
{
    'login': ...,
    'password': ...,
    'query': ...
}
```

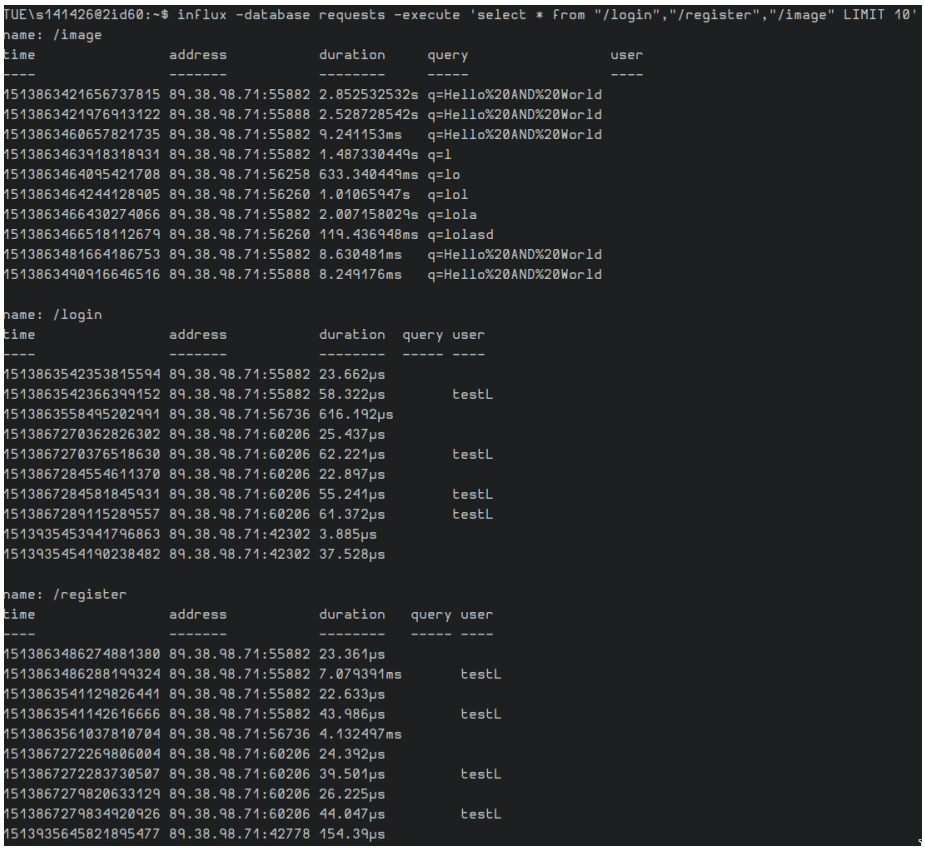
output :

```
{
    'login': ...,
    'password': ...,
    'query': ...
}
```

Screenshots







```
TUE\s141426@2id60:~$ influx -database requests -execute 'select * from "/login","/register","/image" LIMIT 10'
name: /image
time                address           duration     query                 user
----                -------           --------     -----                 ----
1513863421656737815 89.38.98.71:55882 2.852532532s q=Hello%20AND%20World
1513863421976913122 89.38.98.71:55888 2.528728542s q=Hello%20AND%20World
1513863460657821735 89.38.98.71:55882 9.241153ms   q=Hello%20AND%20World
1513863463918318931 89.38.98.71:55882 1.487330449s q=l
1513863464095421708 89.38.98.71:56258 633.340449ms q=lo
1513863464244128905 89.38.98.71:56260 1.01065947s  q=lol
1513863466430274066 89.38.98.71:55882 2.007158029s q=lola
1513863466518112679 89.38.98.71:56260 119.436948ms q=lolasd
1513863481664186753 89.38.98.71:55882 8.630481ms   q=Hello%20AND%20World
1513863490916646516 89.38.98.71:55888 8.249176ms   q=Hello%20AND%20World

name: /login
time                address           duration  query user
----                -------           --------  ----- ----
1513863542353815594 89.38.98.71:55882 23.662µs
1513863542366399152 89.38.98.71:55882 58.322µs        testL
1513863558495202991 89.38.98.71:56736 616.192µs
1513867270362826302 89.38.98.71:60206 25.437µs
1513867270376518630 89.38.98.71:60206 62.221µs        testL
1513867284554611370 89.38.98.71:60206 22.897µs
1513867284581845931 89.38.98.71:60206 55.241µs        testL
1513867289115289557 89.38.98.71:60206 61.372µs        testL
1513935453941796863 89.38.98.71:42302 3.885µs
1513935454190238482 89.38.98.71:42302 37.528µs

name: /register
time                address           duration   query user
----                -------           --------   ----- ----
1513863486274881380 89.38.98.71:55882 23.361µs
1513863486288199324 89.38.98.71:55882 7.079391ms       testL
1513863541129826441 89.38.98.71:55882 22.633µs
1513863541142616666 89.38.98.71:55882 43.986µs         testL
1513863561037810704 89.38.98.71:56736 4.132497ms
1513867272269806004 89.38.98.71:60206 24.392µs
1513867272283730507 89.38.98.71:60206 39.501µs         testL
1513867279820633129 89.38.98.71:60206 26.225µs
1513867279834920926 89.38.98.71:60206 44.047µs         testL
1513935645821895477 89.38.98.71:42778 154.39µs
```