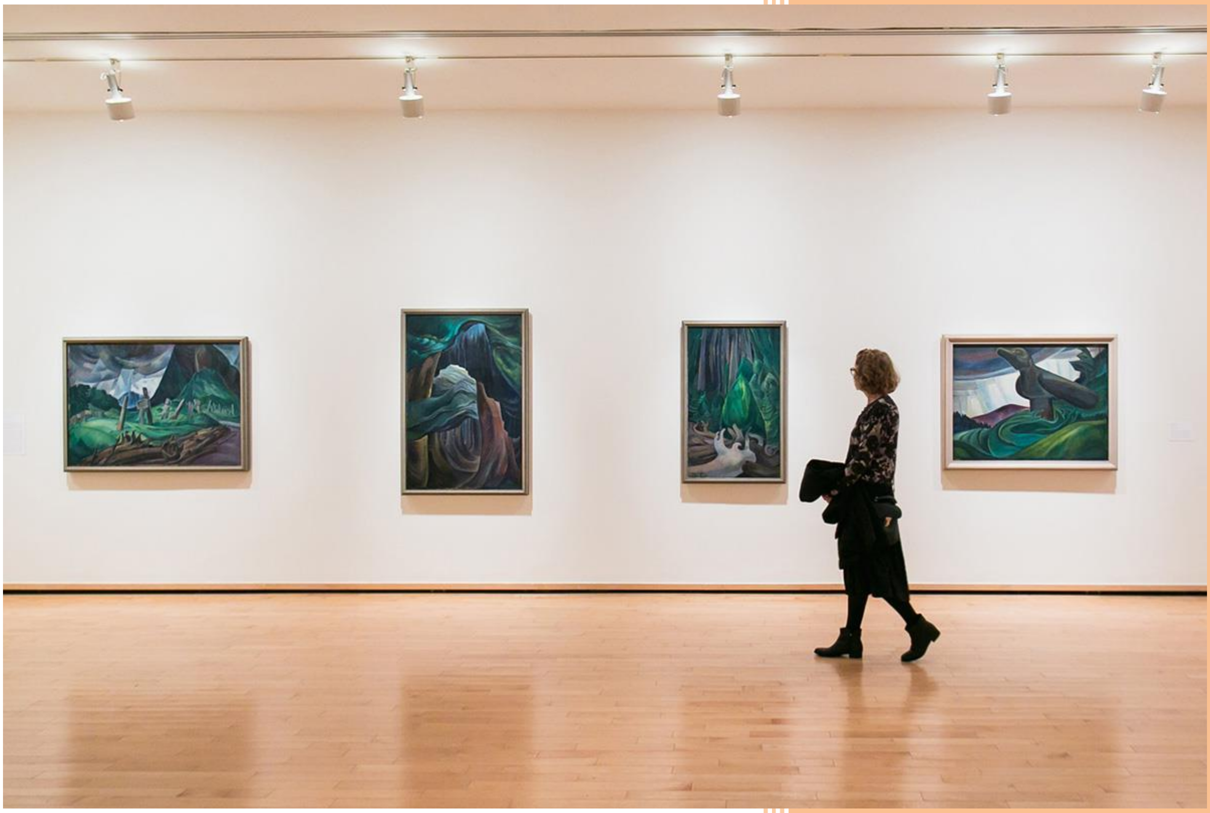
2023

Pokémon Gallery



CAB432 Assignment 1

Jacob Fallows

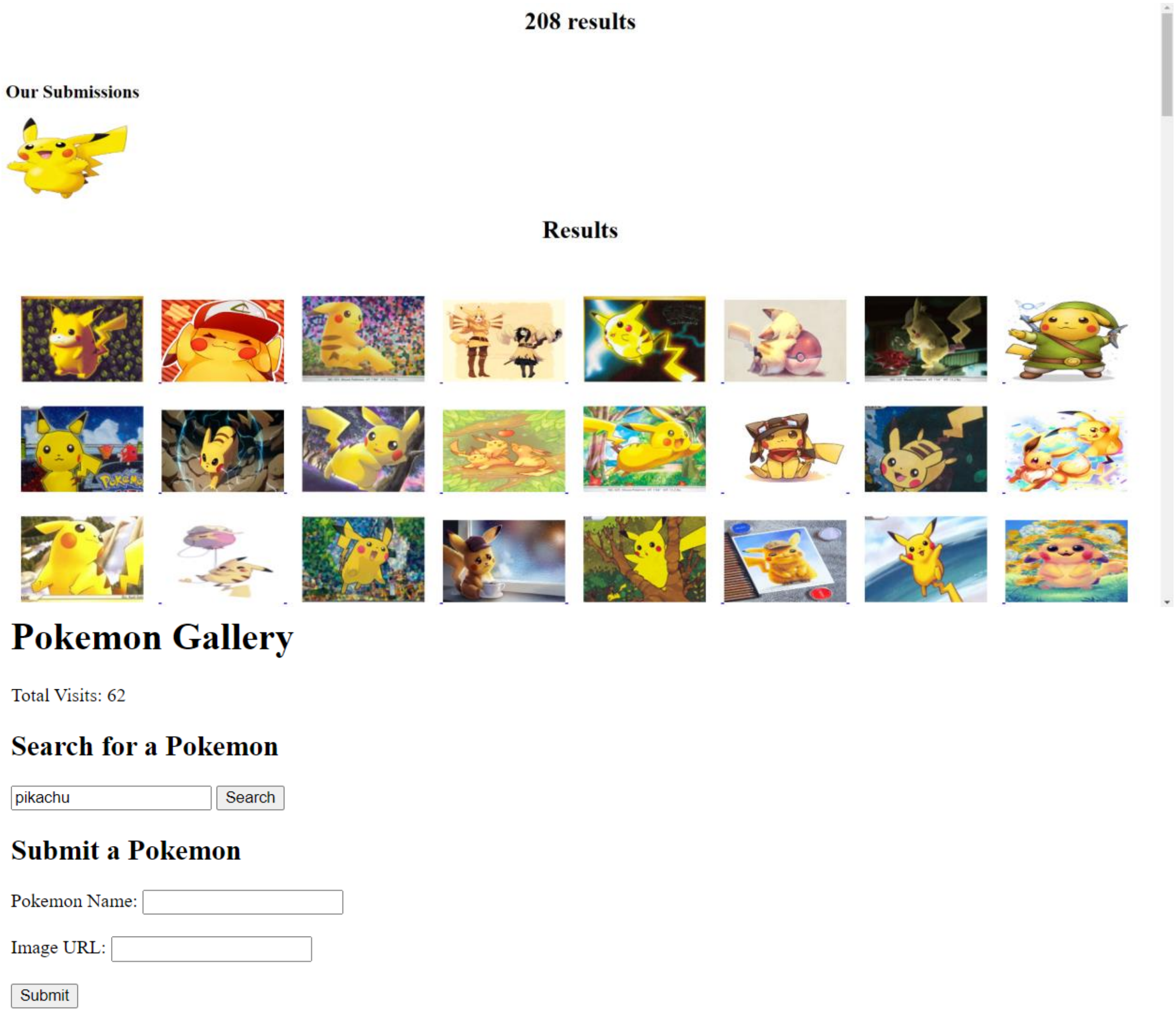
n10749527

9/17/2023

Contents

Executive Summary

The pokemon gallery website has 2 main endpoints and a home page to access those endpoints. It is hosted using amazon services and docker and takes advantage of 2 apis to function. The app can be run on multiple amazon containers so it can load balance to accommodate for changes in traffic. It has a stateless design where any instance can update the artwork data and the page views. The site takes advantage of various node modules for debugging, body parsing environment variables for security and more. There are still some bugs in specific edge cases, however the average user won't come across these.



208 results

Our Submissions

Results

Pokemon Gallery

Total Visits: 62

Search for a Pokemon

pikachu [Search]

Submit a Pokemon

Pokemon Name:

Image URL:

[Submit]

Introduction

Description

The pokemon gallery is a site that compiles the artworks parts of pokemon cards with already established fan art to create a massive gallery of pokemon. Users can search for their favorite pokemons and can even upload their own works.

System Requirements

User Story 1:

As a user I want the system to let me search art of different Pokémon of my choosing so that I can view and download my favourite images.

User Story 2:

As a user I want the system to let me upload my own artworks/images so that I can share them with others.

Resources

API 1: Pokémon TCG: https://dev.pokemontcg.io

This API provides a comprehensive list of Pokémon cards from the trading card game. The main data source is the cards endpoint which has the image files from every version of the game.

https://docs.pokemontcg.io/api-reference/cards/card-object/

API 2: Deviant Art: https://www.deviantart.com/developers/

This API provides another data source which has unofficial fan art on various topics. It will be used to complement the artwork from the Pokémon TCG official artworks.

https://www.deviantart.com/search/deviations?q=pikachu

Persistence service

Persistence Service: Amazon S3 Bucket

https://aws.amazon.com/s3/

Cloud Services:

Docker

https://hub.docker.com/repository/docker/n10749527/pokegallery

Technical Breakdown

Architecture and Data Flow

The application uses the node module “express-generator” for the main architecture of the program. I have separated the docker stuff from the main application. There are two main folders which take care of the javascript files which are helper/ and routes/.

See appendix: file structure

For the data flow I will explain for each use case. It starts in the app.js file which serves the index.js route specified in package.json. The user enters a pokemon name into the html form which will redirect them to the search route in search.js.

See appendix: search route

This is the main section for search route. It takes the name from the query parameter and gives that to the 3 api functions which I will not go into detail here. It pulls from the 2 apis and then an amazon s3 bucket which has a text file with names and urls. Ideally this would be implemented with DynamoDB or another EC2 instance hosting an SQL server.

There is a create page function which will generate the html based on the responses from the 3 data points. I have not used html files in this case as they are very basic pages.

If there are any errors with the query, such as too short or invalid characters like !@#$%^&

It will give a 400 bad request error and the user will have to try again.

The second use case is similar but requires uploading data to the s3 bucket and has to check the name is a valid pokemon name and also checks the fan art server deviantart is online with the placebo endpoint.

Deployment and the Use of Docker

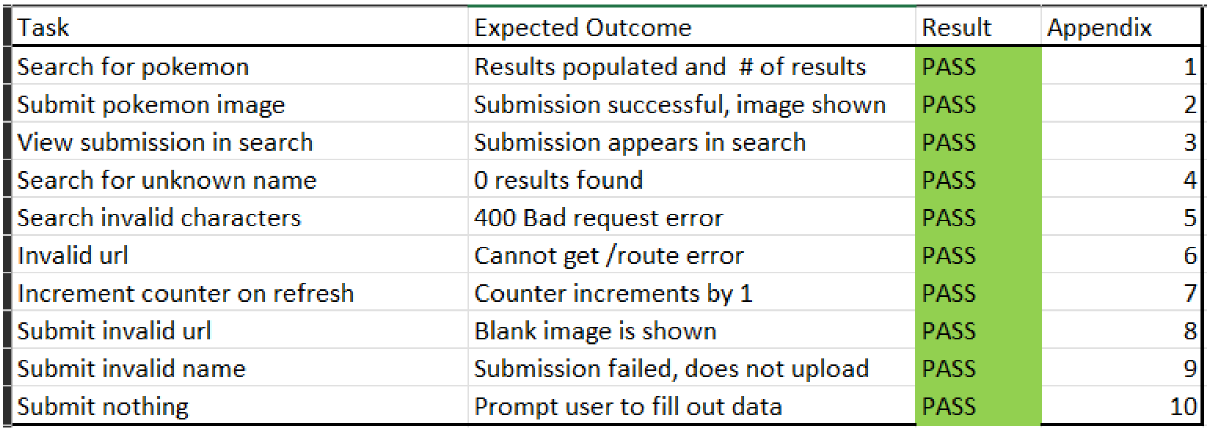
Fresh Ubuntu box setup

```
sudo curl -fsSL https://get.docker.com/ | sh

sudo sh -eux

apt-get install -y uidmap

sudo apt-get install -y uidmap

dockerd-rootless-setuptool.sh install

docker login

docker run -p 8000:3000 -i -t n10749527/pokegallery
```

The express app uses port 3000 and docker forwards that to the instance ip + the 8000 port specified in the command. Once I developed the node application I added the dockerfile which will include all of the src directories under node version 20. Then it installs the required node modules and runs the server automatically. If you want to stop the server you will need to open another terminal and run docker ps then docker stop <id> to close the server.

See appendix: dockerfile

Test plan



Task	Expected Outcome	Result	Appendix
Search for pokemon	Results populated and # of results	PASS	1
Submit pokemon image	Submission successful, image shown	PASS	2
View submission in search	Submission appears in search	PASS	3
Search for unknown name	0 results found	PASS	4
Search invalid characters	400 Bad request error	PASS	5
Invalid url	Cannot get /route error	PASS	6
Increment counter on refresh	Counter increments by 1	PASS	7
Submit invalid url	Blank image is shown	PASS	8
Submit invalid name	Submission failed, does not upload	PASS	9
Submit nothing	Prompt user to fill out data	PASS	10

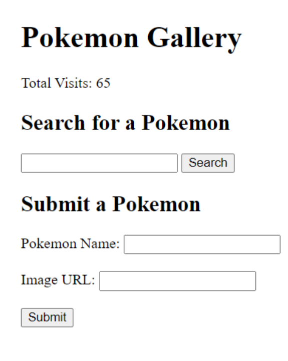
Difficulties / Exclusions / unresolved & persistent errors /

There is some functionality that can be improved especially when the user does the wrong thing. Ideally my application would use a) credentials that don't expire, b) a proper database rather than a csv file which works at this small scale but at a large scale it can break easily, also if there are 2 reads at the same time from different users it will remove one of their submissions as they retrieved the old file version, c) a proper front end framework like react.

I had some difficulties with docker when I would make changes and try to push them there is a duplicate image which I could only delete with docker hub desktop for windows. Also I have the environment variables on a private docker image which is not good practice, they should be passed in with the command line or supply the dotenv file separately from the docker image.

There may still be bugs if you delete the s3 bucket and data files in it, sometimes it struggles to create and update these files right away, but I believe I fixed these bugs. Also I'm sure there are errors which crash the program, like invalid credentials which is "handled" with a log of the catch error.

User guide



Pokemon Gallery

Total Visits: 65

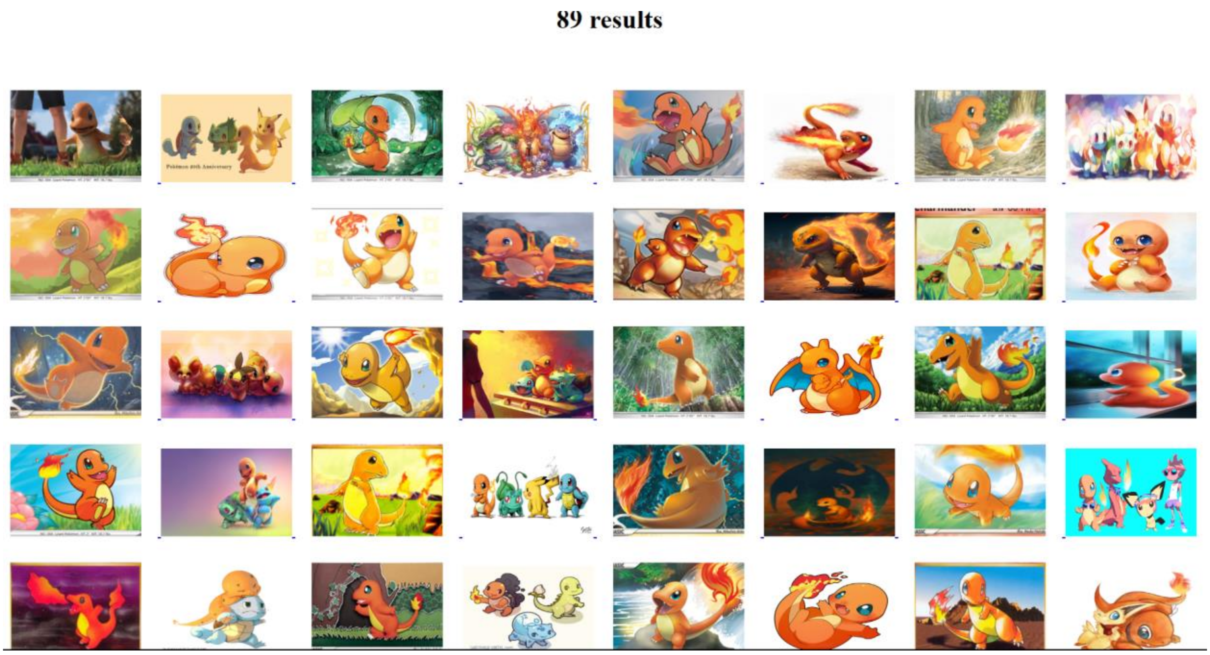
Search for a Pokemon

Submit a Pokemon

Pokemon Name:

Image URL:

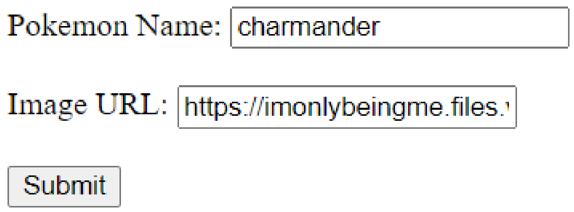
Welcome to the pokemon gallery, enter a pokemon name or submit your own favourite images. Lets say for this example you are looking for "Charmander". Type the name in the box and hit "search"!



89 results

Looks like there are a lot of results for this pokemon. Once you are done looking, head back to the home page and submit your own image.

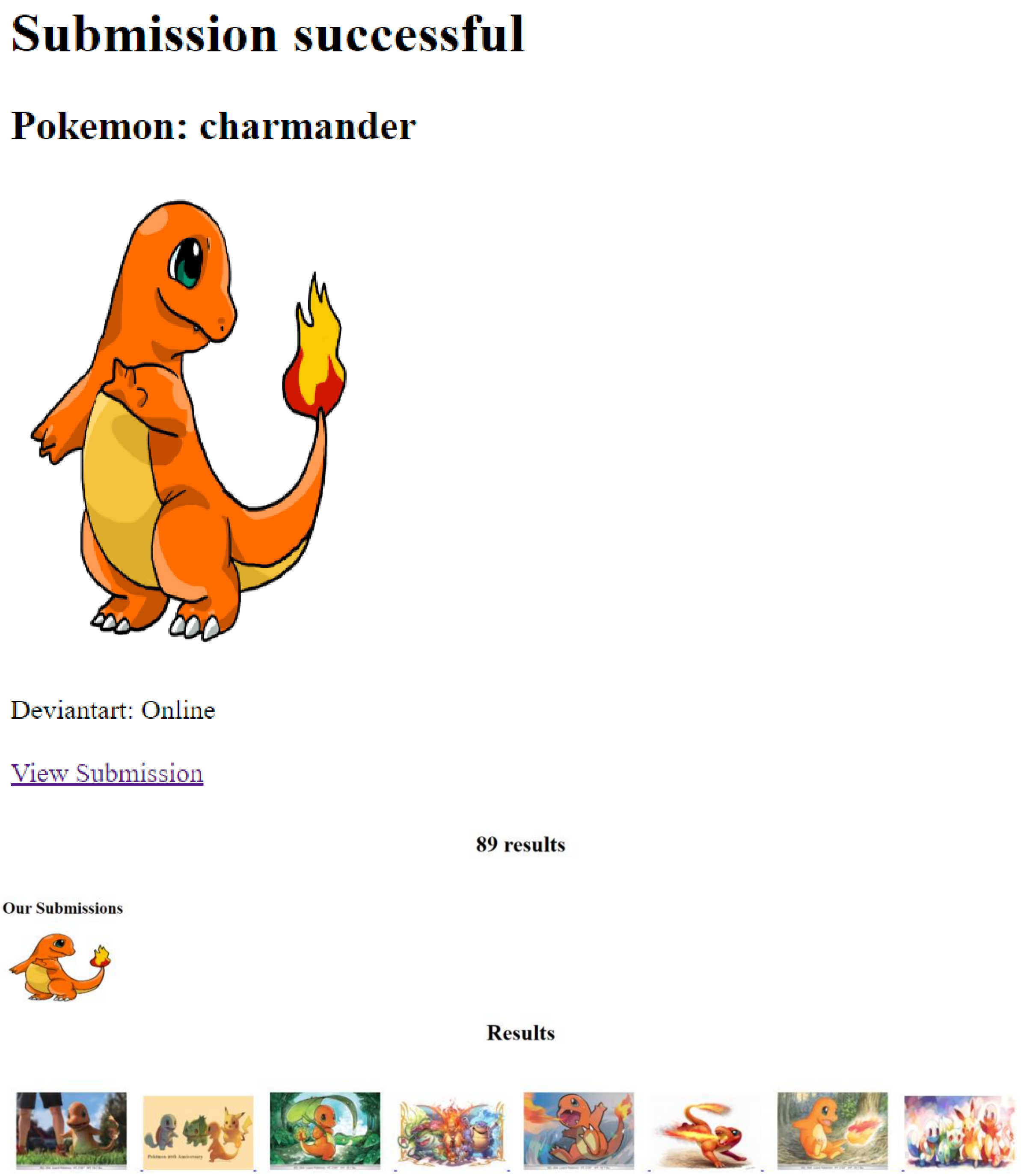
Type in the name of the pokemon the image is of, then find an image online and right click to get the image link.



Pokemon Name: charmander

Image URL: https://imonlybeingme.files.

Hit submit and you will see if it worked or not. If it was successful the image will be stored on our servers. To see your image among the results search the pokemon name again or click the handy link provided below your submission.



Submission successful

Pokemon: charmander

Deviantart: Online

View Submission

89 results

Our Submissions

Results

Analysis

> *In this section we ask you look at your application and to analyse it in response to a couple of prompts we supply below. The marking is based on the quality of the analysis and not on the length of your response. There are two questions. In each case, there is an overall question, and then a series of bullet points that help you respond. A good answer can be no more than a couple of sentences in response to each of them. Say more if you have more to say, but don't waffle. Say it quickly and get on with the next one. This exercise is*

> *comparatively straightforward but there is a corresponding task in Assignment 2 which will be far more difficult.*

Question 1: Vendor Lock-in

> *Looking at your mashup as it stands, how dependent are you on the people who provide the services that you use? In a commercial context, the APIs, the data and the cloud services all matter to you. How hard would it be to change?*
>
> *In your response, you should consider the following prompts:*
>
> *How hard would it be to replace the APIs that you are working with? Could you easily replace them if they were shut down suddenly or their terms of service changed? Consider each of them in turn and explore the domain of the API and tell us about obvious competitors or their absence. (4 marks)*

If the API's stopped working or the key became invalid my service would break. It should be relatively trivial to replace them with alternative ones.

The pokemontcg api is very handy and there are not many actually good alternatives. The closest is the Niantic API but you need private access for that from the pokemon company.

The closest thing I found was https://www.tcgdex.dev/ which is what I would use to replace the current one.

Deviantart was the best home for pokemon fan art, but there are lots of others like imgur, flickr, reddit etc.

> *How hard would it be to change the persistence service you have used to one supplied by another vendor? Identify equivalent alternatives and discuss briefly how that might affect your approach. (2 marks)*

The app is using an s3 bucket at the moment, to switch to another amazon service wouldn't be that hard. To use a sql server I would need to host another docker container and link the two with docker compose, this would compromise on statelessness but be the best alternative at this small scale. This would mean I would need to implement a connection between the replaceable container and the SQL server, like creating my own API.

Question 2: Container Deployment

> *The mashup you have created as part of this assignment is a very limited application by commercial standards. Without getting too carried away, I want you now to think of a more substantial application which has similar characteristics – drawing mainly on external services, perhaps extending to include some user accounts and security, adding in some additional persistence services like those discussed in the weeks leading up to the assignment submission.*
>
> *Working with this more substantial application, what are the advantages and disadvantages of the container-based deployment? This new context will in practice involve scaling and load balancing. You don't yet have direct experience of these services, so I want you to focus on the deployment of the application through software containers. You should refer to earlier notes about the trade-offs between containers and 'full' VMs and consider which might apply here, and which might not.*
>
> *In your response, you should consider the following prompts:*
>
> *Assuming that we have access to a service to manage container deployment and communication, are there any disadvantages to a container-based deployment for this*

> *application? Would we consider deployment of the application directly to a virtual machine i.e. one instance of the application for each instance of a EC2 VM? (2 marks)*

A more substantial application would be another established art site such as pixiv or artstation. These sites take advantage of many node modules and third-party vendors for things like account security, image hosting, website hosting, DDOS defense. If they were using amazon and a container based approach they could do a static allocation for load balancing which would create new containers when the site receives the most traffic, for sites like this it will be a peak hour during the day, usually around 6-9pm peak hours every day. You would not have a 1-1 container – client ratio, it would be more likely to have bigger containers such as 100 or 1000 per container.

> *Drawing upon the discussions of cloud architectures in the early lectures and the material on persistence from week 5 onwards, what persistence options would you consider if the application were to be deployed at scale using a collection of software containers? You may consider more than one level. (2 marks)*

For an image hosting site like mine I would not need a low latency approach. Also managing state would not be as important. There might be a comment section and that and the image uploading it would not matter if there were some delays. Right now on a small scale there are up to 3 seconds delays. In terms of the CAP theorem I would want a AP option with partition tolerance and availability because it is hosting large files the database would quickly be overrun without it. Having acid transactions is not as important as availability, in most cases you always pick availability and have to choose acid or flexible.

References

> https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/s3-node-examples.html

Appendices

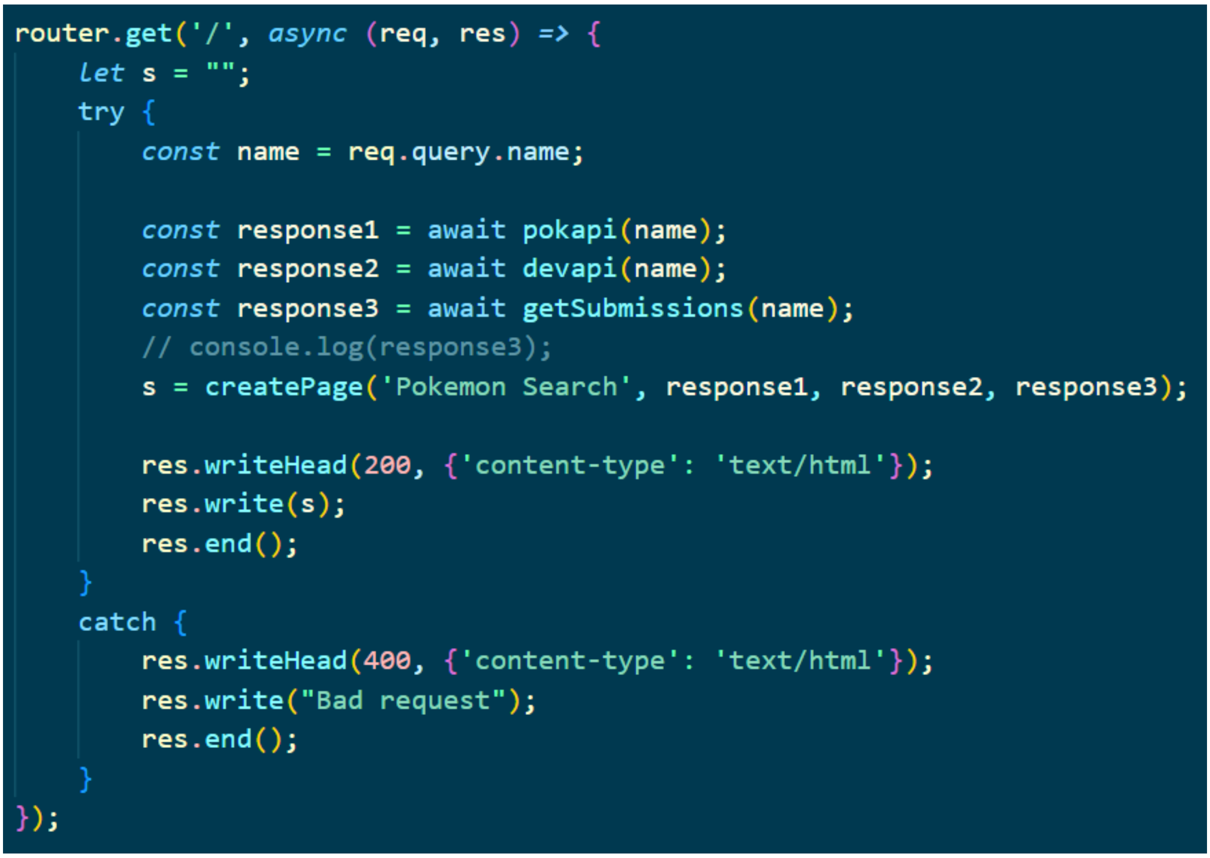
Node modules

```
├── aws-sdk@2.1455.0
├── axios@1.5.0
├── dotenv@16.3.1
├── express@4.18.2
└── morgan@1.10.0
```

Dockerfile

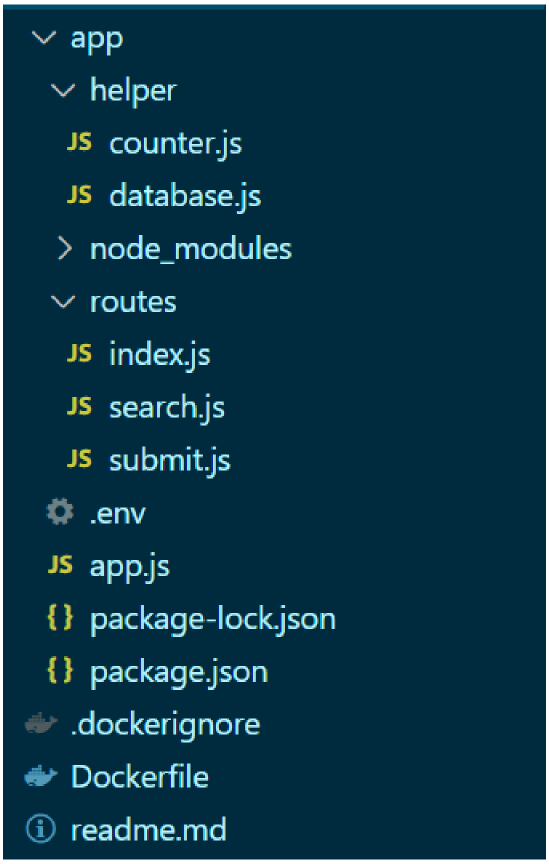
```
# Basic DockerFile to run a node application

# Node version 20
FROM node:20

# Path in the container where the application files will go
WORKDIR /usr/src/app

# Copy in the package files and install required packages
COPY app/package*.json ./
RUN npm install

# Copy our app source tree into the working directory
COPY app/ .

# Command to execute when the container starts
CMD [ "node", "app.js" ]
```

Search route



```
router.get('/', async (req, res) => {
    let s = "";
    try {
        const name = req.query.name;

        const response1 = await pokapi(name);
        const response2 = await devapi(name);
        const response3 = await getSubmissions(name);
        // console.log(response3);
        s = createPage('Pokemon Search', response1, response2, response3);

        res.writeHead(200, {'content-type': 'text/html'});
        res.write(s);
        res.end();
    }
    catch {
        res.writeHead(400, {'content-type': 'text/html'});
        res.write("Bad request");
        res.end();
    }
});
```

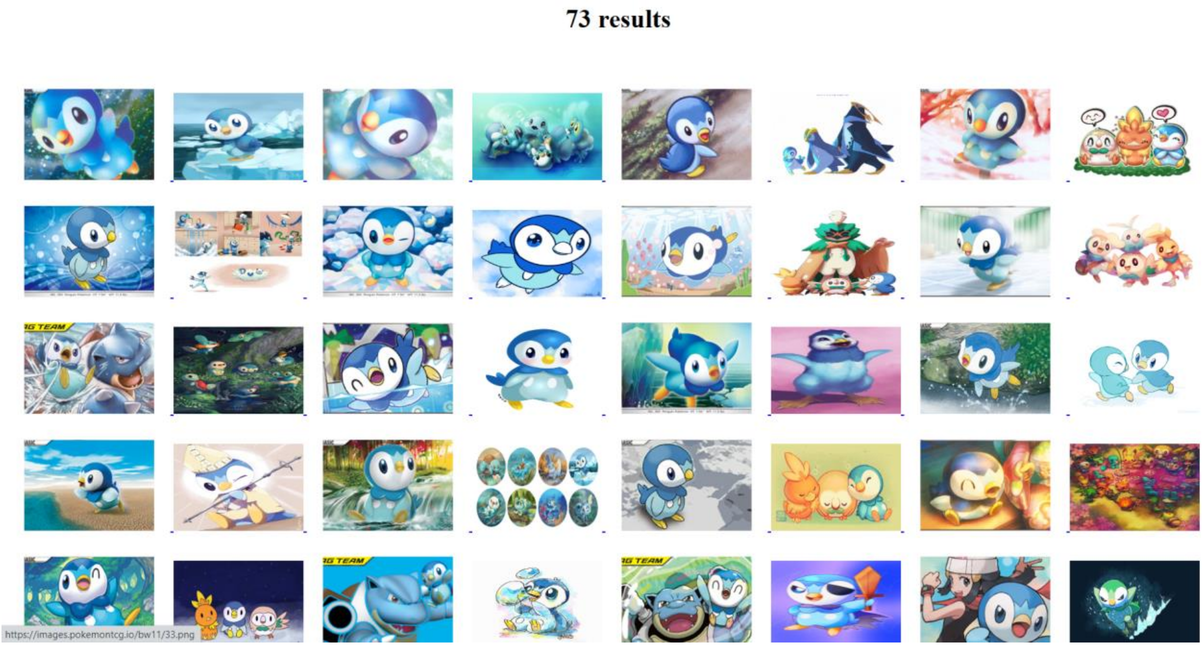
File structure



```
∨ app
  ∨ helper
    JS counter.js
    JS database.js
  > node_modules
  ∨ routes
    JS index.js
    JS search.js
    JS submit.js
  .env
  JS app.js
  {} package-lock.json
  {} package.json
.dockerignore
Dockerfile
readme.md
```

Test cases

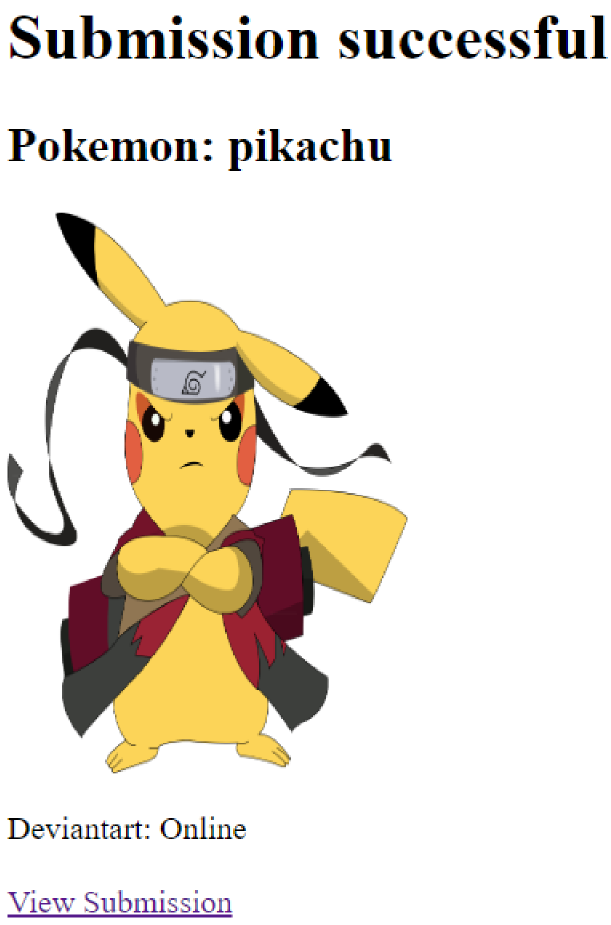
Case 1



73 results

https://images.pokemontcg.io/bw11/33.png

Case 2



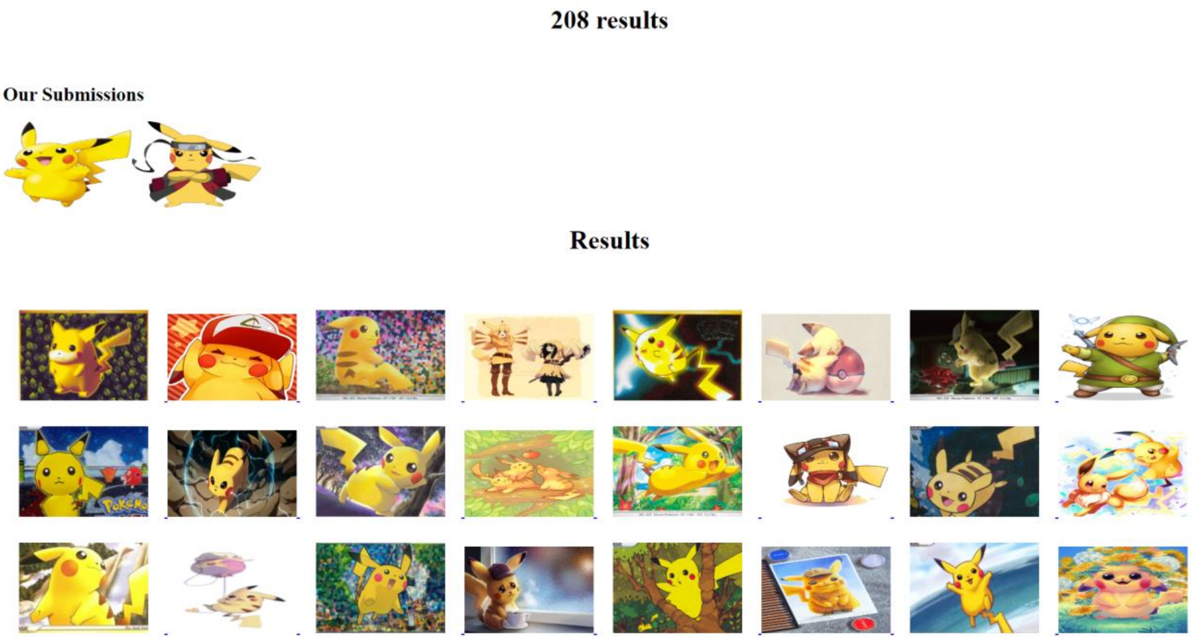
Submission successful

Pokemon: pikachu

Deviantart: Online

View Submission

Case 3

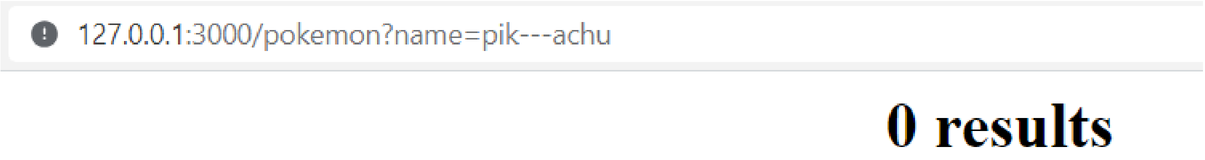


208 results

Our Submissions

Results

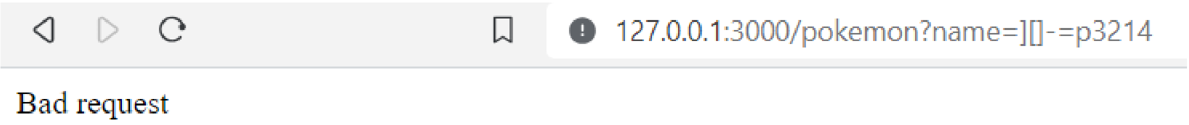
Case 4:



127.0.0.1:3000/pokemon?name=pik---achu

0 results

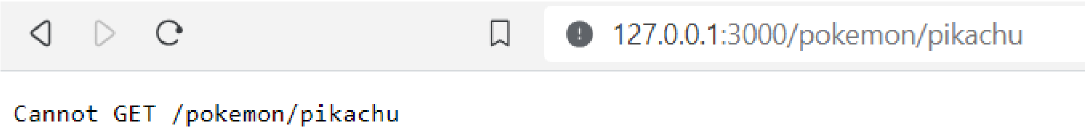
Case 5:



127.0.0.1:3000/pokemon?name=][]-=p3214

Bad request

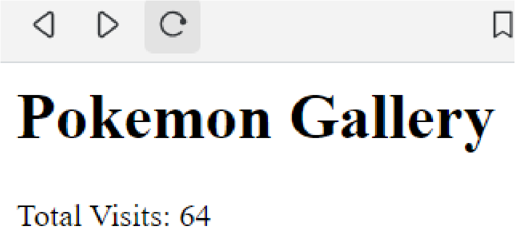
Case 6:



127.0.0.1:3000/pokemon/pikachu

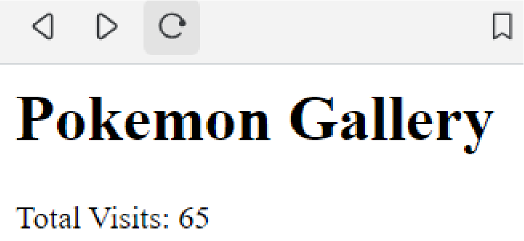
```
Cannot GET /pokemon/pikachu
```

Case 7:



Pokemon Gallery

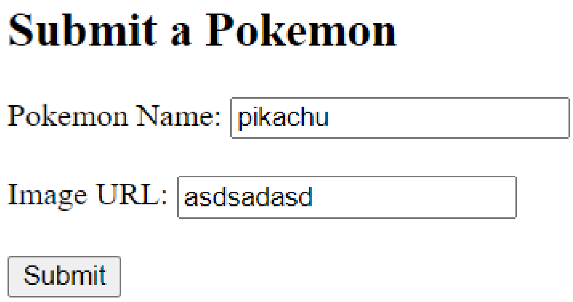
Total Visits: 64



Pokemon Gallery

Total Visits: 65

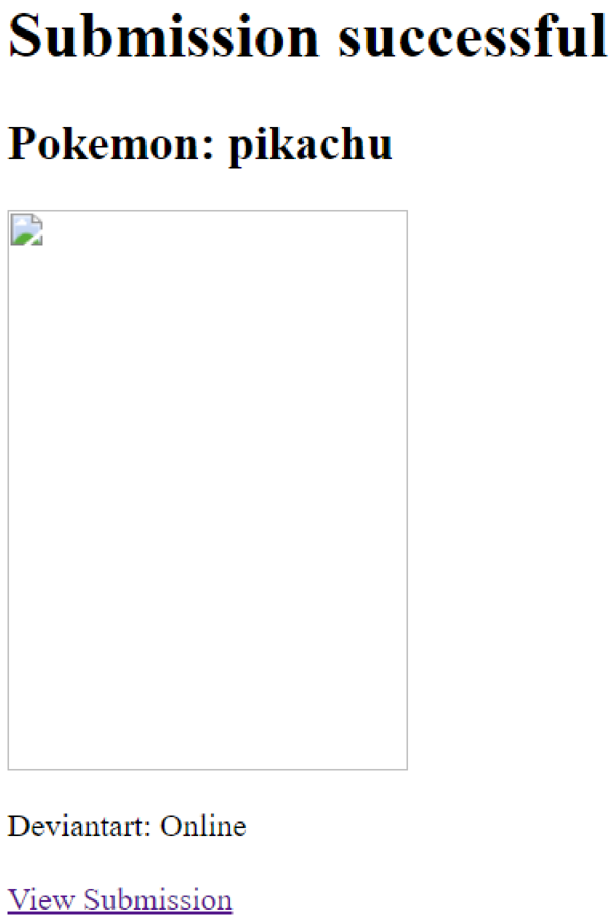
Case 8:



Submit a Pokemon

Pokemon Name: pikachu

Image URL: asdsadasd



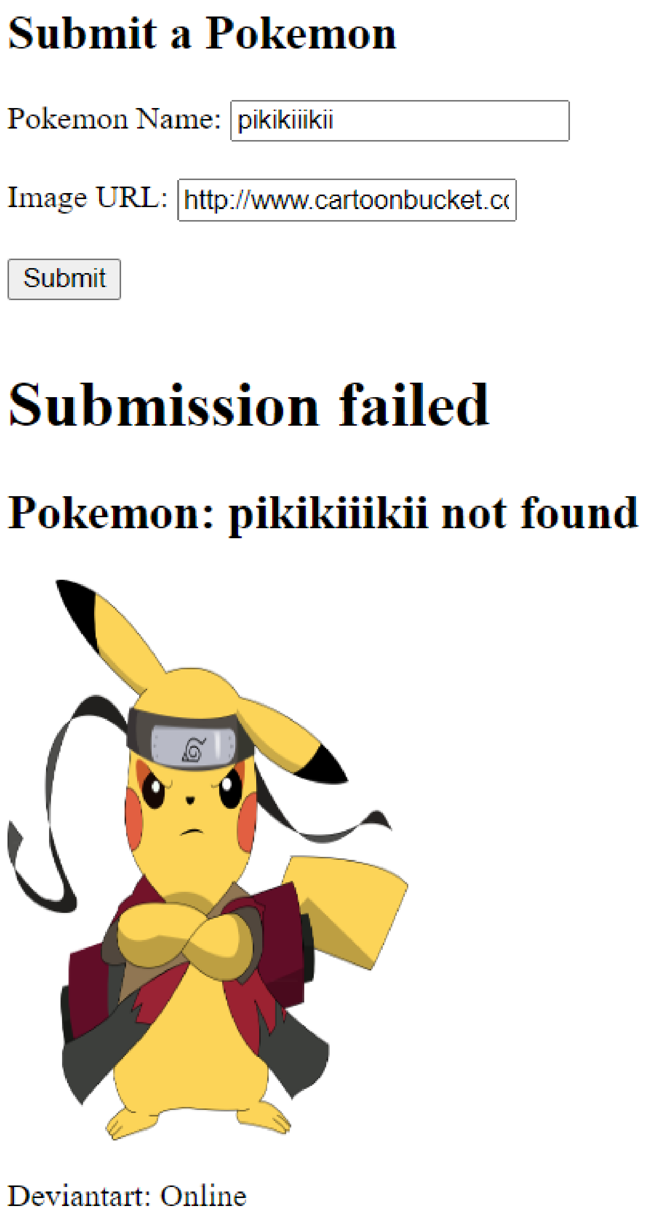
Submission successful

Pokemon: pikachu

Deviantart: Online

View Submission

Case 9



Submit a Pokemon

Pokemon Name: pikikiiikii

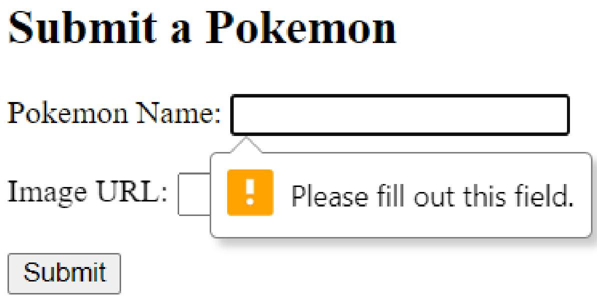
Image URL: http://www.cartoonbucket.c

Submission failed

Pokemon: pikikiiikii not found

Deviantart: Online

Case 10:



Submit a Pokemon

Pokemon Name:

Image URL:

Please fill out this field.