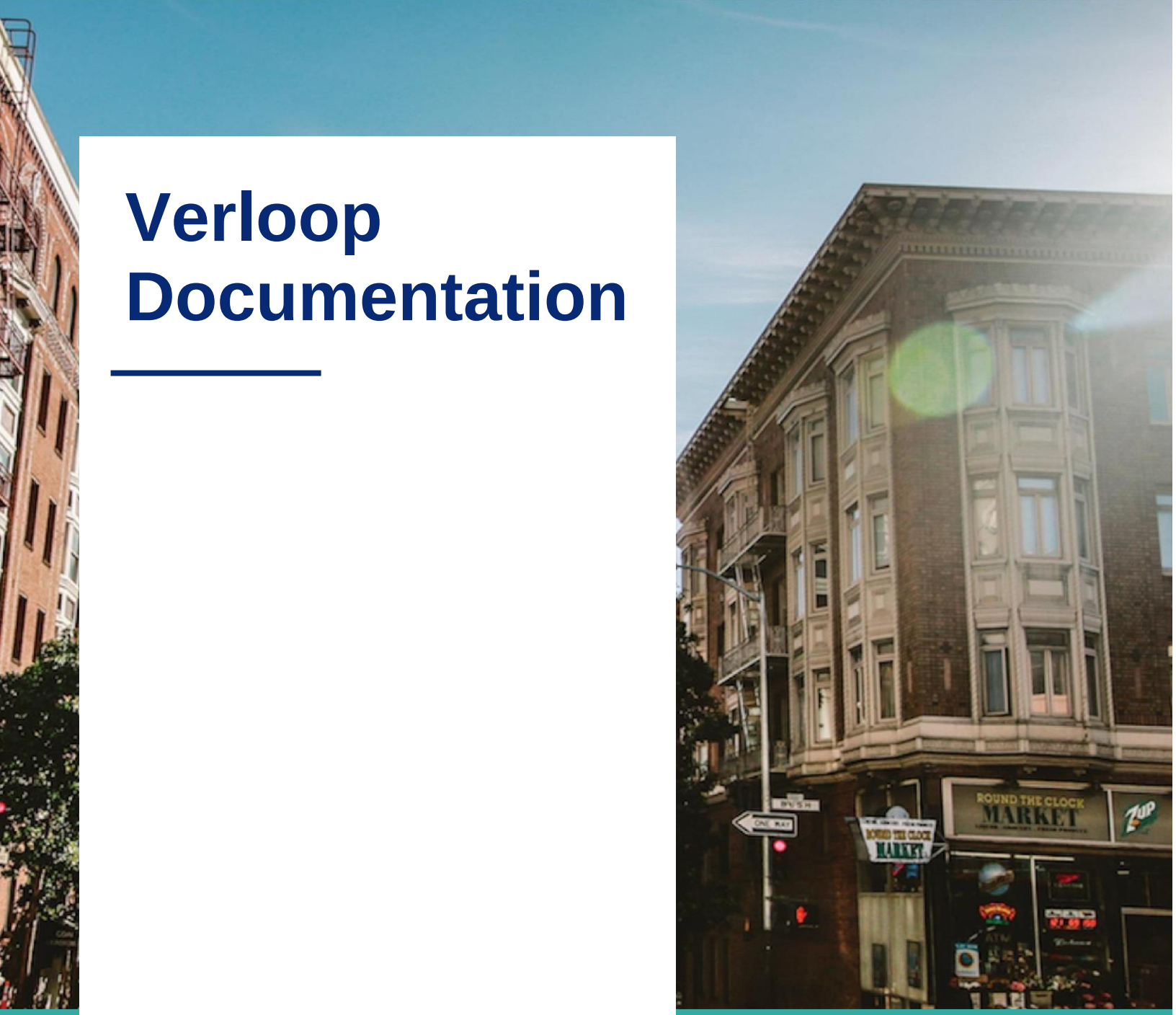
Verloop Documentation



DECEMBER 8

Harshit Singhai
https://harshitsinghai77.github.io/

Task: A simple API which returns top 3 repositories of an organization in GitHub by stars.

Frameworks: React and Node.
Deployed: Heroku, Netlify

Node (Backend): https://harshit-verloop.herokuapp.com/repos
React (Frontend): https://verloop-harshit.netlify.com/
Github: https://github.com/harshitsinghai77/verloop

GitHub API Endpoint: https://api.github.com/orgs/:orgsName/repos
GitHub API Endpoint Parameters: per_page, page,

Note: Free GitHub API limit was reached. Thus, authorization was used to get more API calls. As the application needed to make unauthenticated calls with a higher rate limit, app's client ID and secret were passed as part of the query string.

By default, GitHub API fetches repositories available in the first page. Hence, to get all the repositories spread across multiple pages pagination was used.

Traversing with pagination was used to list all the repositories and get top 3 repositories based on stars.

Getting Started

1) Extract Harshit-hiring-challenge.rar
2) Navigate to the folder
3) Run “npm install”
4) Make .env file containing following values
   GITHUB_CLIENT_ID="#YOUR_GITHUB_CLIENT_ID"
   GITHUB_CLIENT_SECRET="#YOUR_GITHUB_CLIENT_SECRET"
5) Run “node app.js”
6) To test run “npm test”

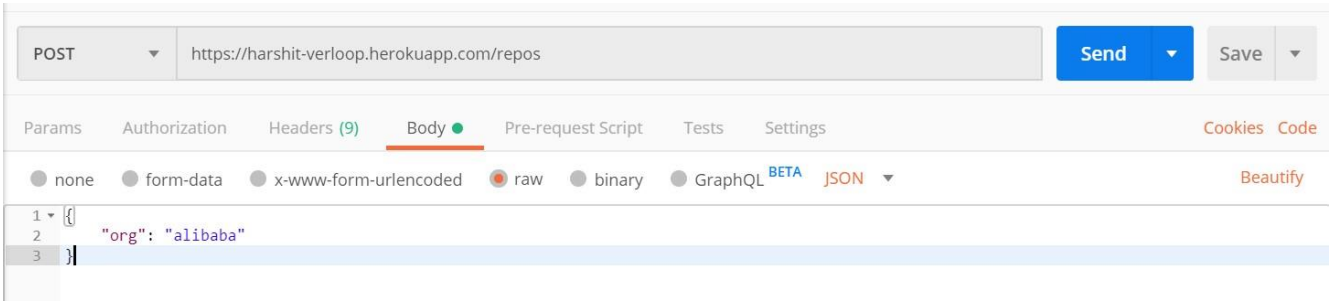
Node Dependencies

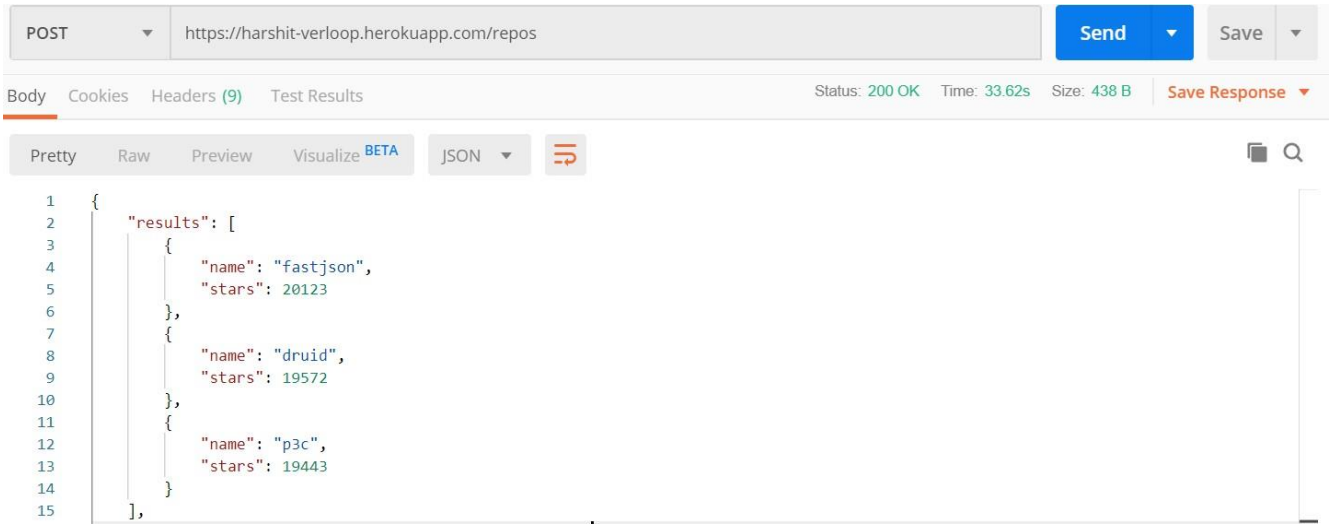
```
"dependencies": {
    "axios": "^0.19.0",
    "body-parser": "^1.19.0",
    "cors": "^2.8.5",
    "dotenv": "^8.2.0",
    "express": "^4.17.1",
    "express-pino-logger": "^4.0.0",
    "pino": "^5.14.0"
  }
```

React Dependencies

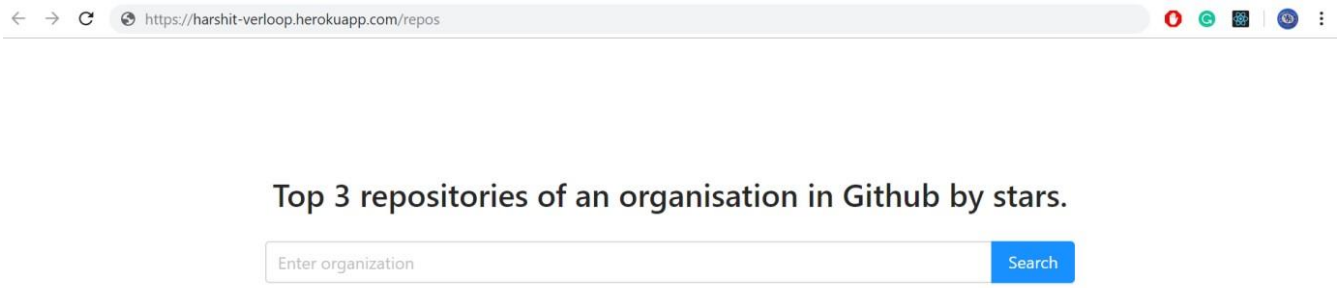
```
"dependencies": {
    "antd": "^3.19.1",
    "axios": "^0.19.0",
    "node-sass": "^4.12.0",     "react": "^16.8.6",
    "react-dom": "^16.8.6",
    "react-scripts": "3.0.1"
  }
```

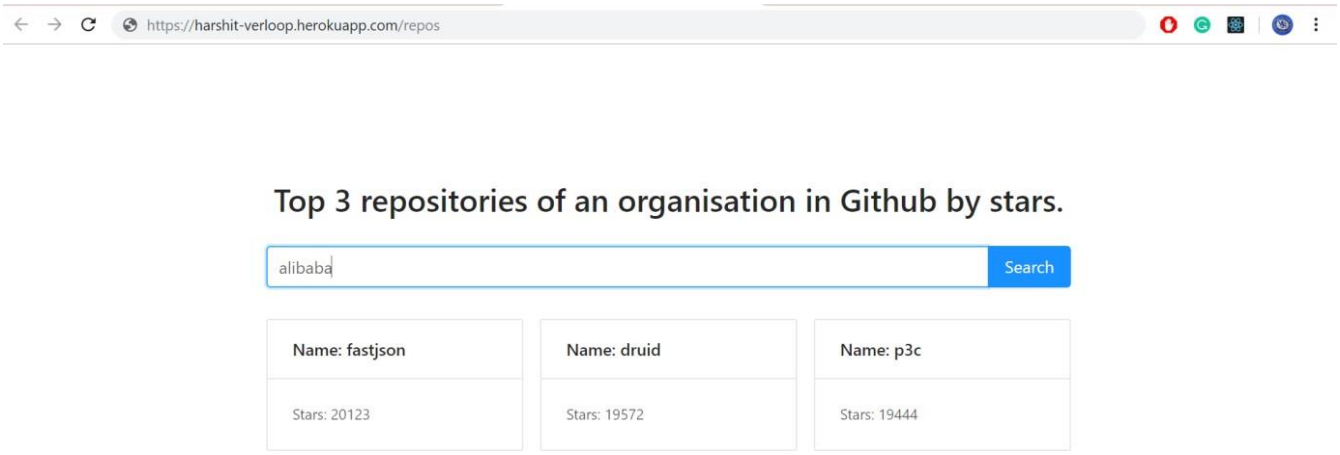
Working Demo (Postman)





Working Demo (Web)





Deploy this as a microservice on a cloud and share link

https://harshit-verloop.herokuapp.com/repos

Functional and Unit Testing

```
"devDependencies": {
    "chai": "^4.2.0",
    "mocha": "^6.2.2",
    "supertest": "^4.0.2"
}
```

Mocha, a JavaScript test framework to unit test express routes was used. For testing HTTP calls a SuperTest was used and Chai for asserting the result.

All the test files reside in test/ folder. To test simply run “npm run test”

```
npm run test
```

I’ve used following tests cases for unit testing the route.

1) The route should return status 200.
2) Results on passing correct values
3) Empty values passed to the route {“org”: “”}
4) Organization repository name does not exist {“org”: “kdsfbksdfbadfadsbiaD”}

```
> github-repositories-star@1.0.0 test C:\Users\shadow\Documents\Projects\verloop\node
> mocha --timeout 10000



Server running on port 3000
  Functional testing the /repos route
    √ should return OK status (2405ms)
    √ should return results on passing correct values (2700ms)
    √ should return Oganization name not found on rendering with empty values
    √ should return No organization found on rendering with invalid organization name (768ms)


  4 passing (6s)
```

Logging

```
"dependencies": {
    "express-pino-logger": "^4.0.0",
    "pino": "^5.14.0"
  }
```

All the logs can be found in the directory logs/
Info.txt file is created in the logs/ directory containing useful information such as hostname, timestamp etc.

Each request is automatically logged.

info - Notepad

File Edit Format View Help


```
{"level":30,"time":1575714438809,"pid":32004,"hostname":"LAPTOP-609D4RP1","req":
{"id":1,"method":"POST","url":"/repos","headers":{"content-type":"application/json","user-
agent":"PostmanRuntime/7.19.0","accept":"*/*","cache-control":"no-cache","postman-token":"b2e12f59-e54c-44cc-add3-
98522d08cdd4","host":"localhost:3000","accept-encoding":"gzip, deflate","content-length":"22","connection":"keep-
alive"},"remoteAddress":"::1","remotePort":51325},"res":{"statusCode":200,"headers":{"x-powered-
by":"Express","access-control-allow-origin":"*","content-type":"application/json; charset=utf-8","content-
length":"238","etag":"W/\"ee-8UvwEiHqU3M/ltBqjDaYAW/W3F4\""}},"responseTime":1206,"msg":"request completed","v":1}
{"level":30,"time":1575733938628,"pid":13660,"hostname":"LAPTOP-609D4RP1","req":
{"id":2,"method":"POST","url":"/repos","headers":{"host":"127.0.0.1:3000","accept-encoding":"gzip, deflate","user-
agent":"node-superagent/3.8.3","content-type":"application/json","content-
length":"17","connection":"close"},"remoteAddress":"::ffff:127.0.0.1","remotePort":60517},"res":
{"statusCode":200,"headers":{"x-powered-by":"Express","access-control-allow-origin":"*","content-
type":"application/json; charset=utf-8","content-length":"41","etag":"W/\"29-kFQDiOevgACkOufbcXXSLV1ynAM
\""}},"responseTime":7302,"msg":"request completed","v":1}
{"level":30,"time":1575733938666,"pid":13660,"hostname":"LAPTOP-609D4RP1","req":
{"id":3,"method":"POST","url":"/repos","headers":{"host":"127.0.0.1:3000","accept-encoding":"gzip, deflate","user-
agent":"node-superagent/3.8.3","connection":"close","content-
length":"0"},"remoteAddress":"::ffff:127.0.0.1","remotePort":60532},"res":{"statusCode":200,"headers":{"x-powered-
by":"Express","access-control-allow-origin":"*","content-type":"application/json; charset=utf-8","content-
length":"85","etag":"W/\"55-CaRH6N2TAonoAfaVjaYYdr5B1ec\""}},"responseTime":2,"msg":"request completed","v":1}
{"level":30,"time":1575733942868,"pid":13660,"hostname":"LAPTOP-609D4RP1","req":
{"id":1,"method":"POST","url":"/repos","headers":{"host":"127.0.0.1:3000","accept-encoding":"gzip, deflate","user-
agent":"node-superagent/3.8.3","content-type":"application/json","content-
length":"17","connection":"close"},"remoteAddress":"::ffff:127.0.0.1","remotePort":60479},"res":
{"statusCode":200,"headers":{"x-powered-by":"Express","access-control-allow-origin":"*","content-
```


Response time

Response time is simply calculated based on time it requires. Even though this is not good metric to measure the response time but for simplicity I've calculate the response time based on end time – start time.

As soon as the route is call, I calculate the start time

```
let start_time = new Date().getTime();
```

Before sending the response, I calculate the total time using

```
new Date().getTime() - start_time
```

Request/Seconds

Npm package to measure the performance of the endpoint

Loadtest: https://www.npmjs.com/package/loadtest

```
Command: loadtest -c 10 --rps 100 http://localhost:3000/repos -P‘{“org”:”alibaba”}’
```

Docstrings and comments

Docstrings and comments are included in the code.

Conclusion

Thank you for giving me this opportunity. I would love if you could provide a feedback. Looking forward to hearing from you soon.

Follow me at

https://github.com/harshitsinghai77
https://harshitsinghai77.github.io/
https://medium.com/@harshitsinghai77
harshitsinghai77@gmail.com