IMT 542
GitHub User Analysis Project

Name: Xinyi (Eve) Wang

UNIVERSITY of WASHINGTON

Overview

Info story or who is the user and why is it important to them

Existing structure and FAIR assessment

How you decided to improve the structure

What your new structure is

How would quality be identified and addressed

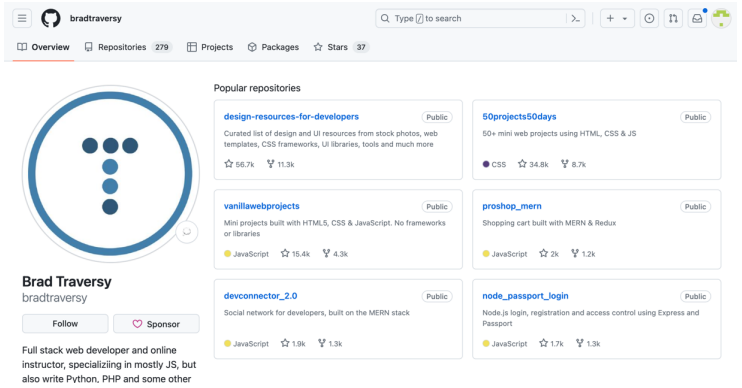
Info story

Recruiters - > Candidate’s GitHub

- > in 2 minutes, determine: Are you a proficient Java developer? How good are you?

Problem:

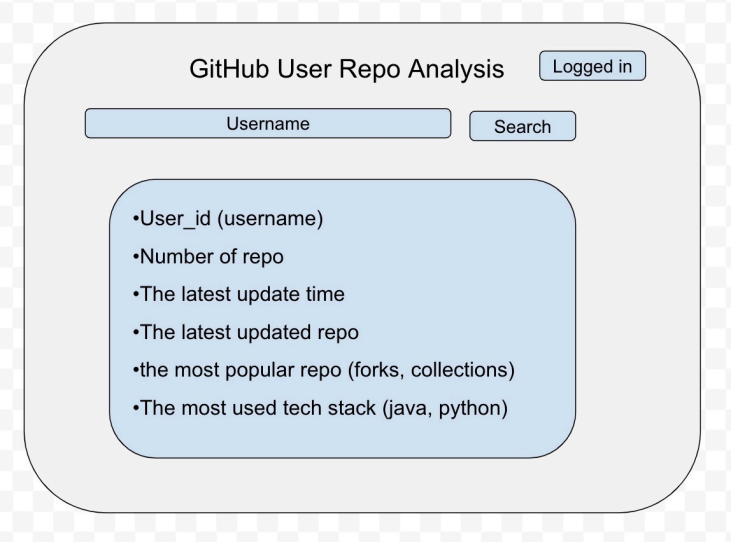
- Too much information
- What is their most frequently used language?
- Difficult to generate insights on candidates
- in a quick and uniform manner



Info story

Solution: Automated GitHub user analysis

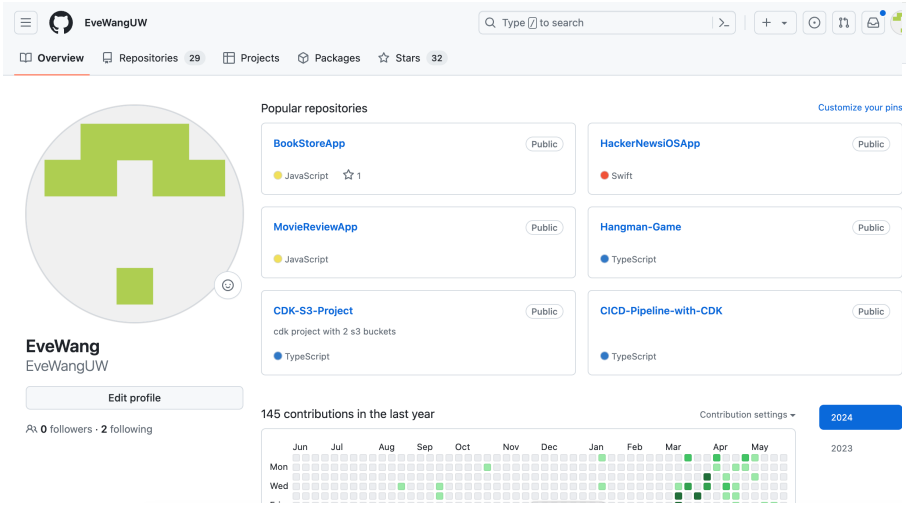
- ➔ Retrieving data available on the GitHub webpage
- ➔ Filtering useful information for recruiters
- ➔ Analyze and add new fields with additional insights
- ➔ Summarize the user and repo insights
- ➔ Help recruiters find candidates with the right skillsets

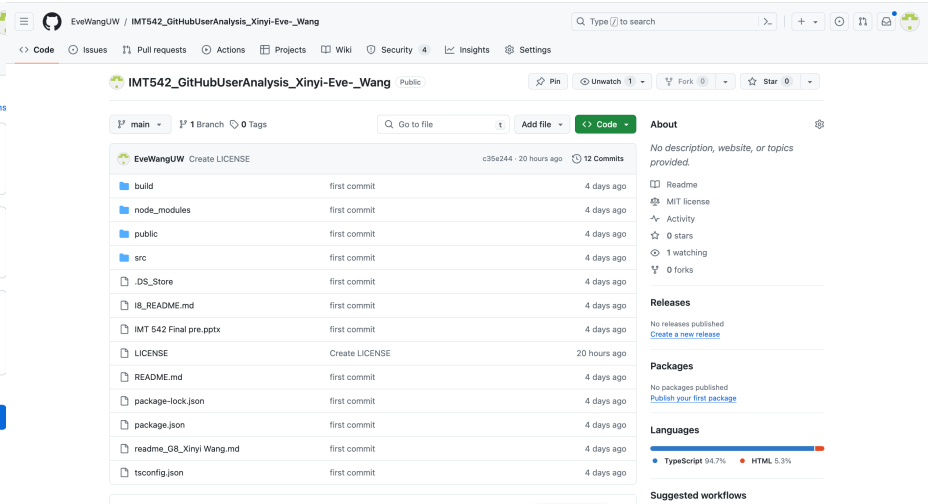


Information wireframe

Existing structure and FAIR assessment

Existing structure from GitHub web page





Existing structure and FAIR assessment

Existing structure from

GitHub REST API endpoints

1. User information

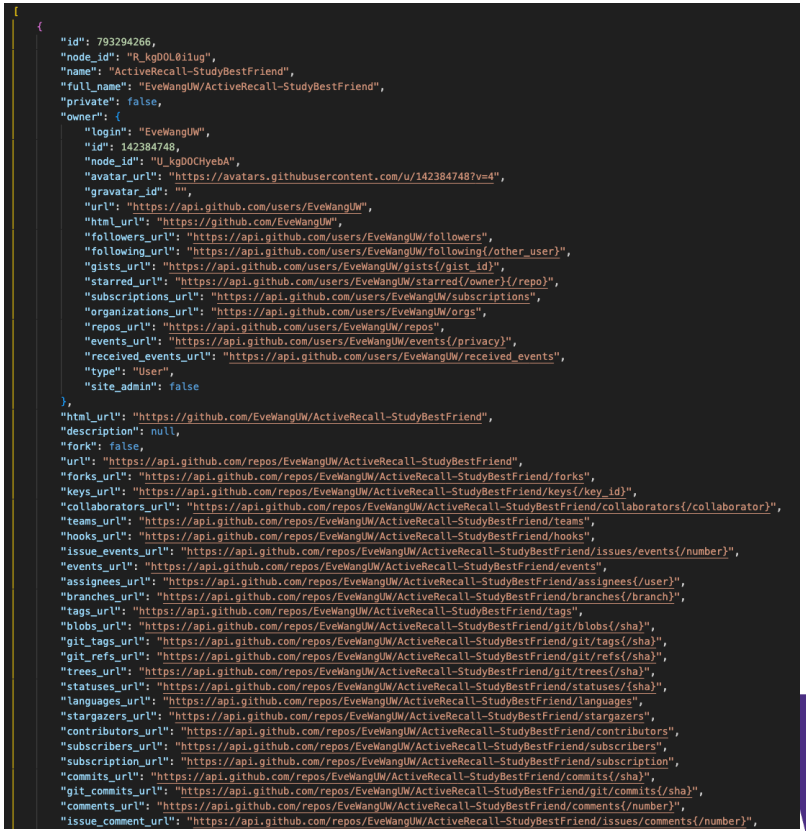

```
{
  "login": "EveWangUW",
  "id": 142384748,
  "node_id": "U_kgDOCHyebA",
  "avatar_url": "https://avatars.githubusercontent.com/u/142384748?v=4",
  "gravatar_id": "",
  "url": "https://api.github.com/users/EveWangUW",
  "html_url": "https://github.com/EveWangUW",
  "followers_url": "https://api.github.com/users/EveWangUW/followers",
  "following_url": "https://api.github.com/users/EveWangUW/following{/other_user}",
  "gists_url": "https://api.github.com/users/EveWangUW/gists{/gist_id}",
  "starred_url": "https://api.github.com/users/EveWangUW/starred{/owner}{/repo}",
  "subscriptions_url": "https://api.github.com/users/EveWangUW/subscriptions",
  "organizations_url": "https://api.github.com/users/EveWangUW/orgs",
  "repos_url": "https://api.github.com/users/EveWangUW/repos",
  "events_url": "https://api.github.com/users/EveWangUW/events{/privacy}",
  "received_events_url": "https://api.github.com/users/EveWangUW/received_events",
  "type": "User",
  "site_admin": false,
  "name": "EveWang",
  "company": null,
  "blog": "",
  "location": null,
  "email": null,
  "hireable": null,
  "bio": null,
  "twitter_username": null,
  "public_repos": 25,
  "public_gists": 0,
  "followers": 0,
  "following": 2,
  "created_at": "2023-08-16T07:12:54Z",
  "updated_at": "2024-05-03T01:02:37Z"
}
```

Existing structure and FAIR assessment

Existing structure from

GitHub REST API endpoints

2. User repo information

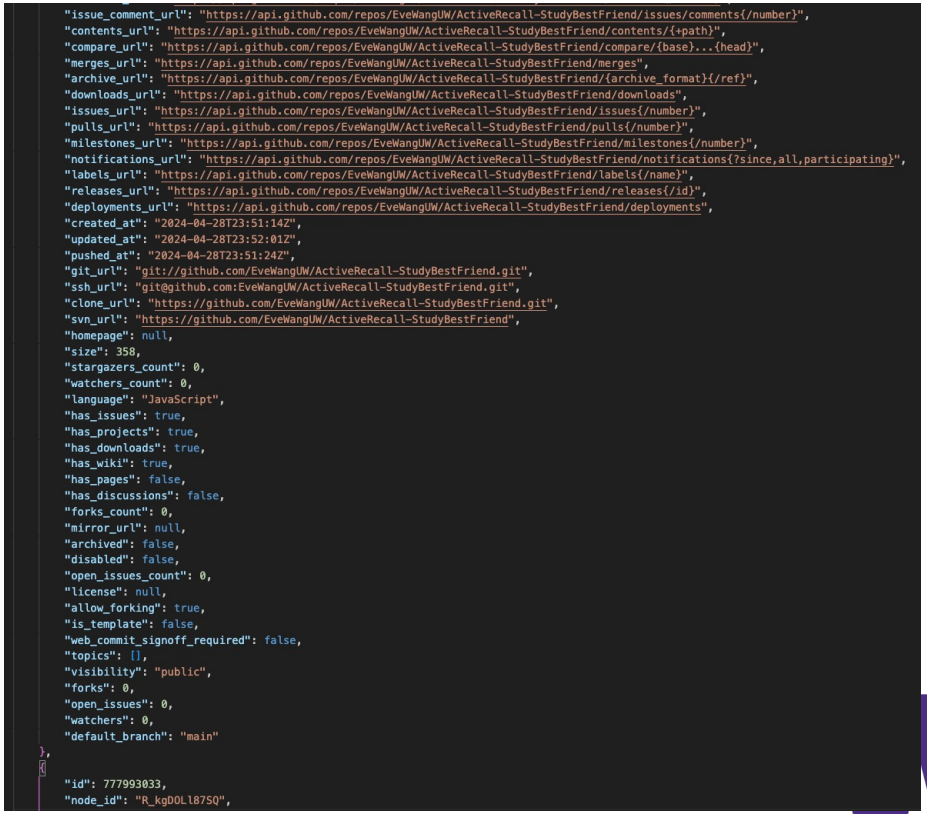


Existing structure and FAIR assessment

Existing structure from

GitHub REST API endpoints

2. User repo information



Existing structure and FAIR assessment

FAIR assessment - GOOD:

F1: Both the user data and the user repo data have globally unique identifiers such as login and id.

F3: The metadata clearly and explicitly include the identifier of the data they describe, for example, the repo data includes the repo owner's information

F2: There is also rich metadata on the user and repository details, such as 'updated_at', 'created_at', profile URLs, and owner details

A1, A1.1: The data are retrievable via standardized, open, and free APIs provided by GitHub.

I1: These data use the JSON language format, which is a formal, accessible, shared, and broadly applicable language for knowledge representation

I3: the various URLs included in the user and the repo data direct the users to related

resources, for example, the fields 'followers' and 'commits' in the user and repo data include

URLs to various resources such as 'followers_url' and 'commits_url'.

Existing structure and FAIR assessment

FAIR assessment – deficiencies that need to be improved

A2: if the user account or the repo is deleted, the metadata might not be accessible anymore, which may not conform to the A2 principle.

R1: In terms of reusability, although rich attributes are provided, some of the attributes are not relevant to our recruiters, such as ‘avater_url’, ‘is_template’, not conforming to R1

R1.1: In the user and user repo, most of the data lack license information, not conforming to the R1.1.

R1.2: Finally, although time related attributes such as ‘updated_at’, ‘created_at’ provide some provenance information, but more details could be included to achieve the R1.2 principle.

Existing structure and portability assessment

Portability assessment – deficiencies that needs to be improved

Clean meaning: raw JSON with too much info and lack of summary

Accessibility: look at the page directly or make API calls

Transparency: difficulty understanding the field

Interoperability: JSON format only, may not work well on mobile devices.

Usability: difficult to use directly for candidate assessment

Structure: lack of summary and structure

How you decided to improve the structure

To improve accessibility (A),

1. implement a backup system to retain metadata even if the user account or repo is deleted, which can be accomplished by allowing the user to **download the data** from the web application

To improve reusability (R),

1. **Remove unnecessary attributes** like ‘avator_url’ and focus on displaying information related to recruiters.

2. Add relevant **license** for this application

3. **Add more analytics details** and provenance information to provide recruiters with useful insights into candidate profiles.

How you decided to improve the structure

Portability improvement:

Clean meaning:

raw JSON with too much info and lack of summary -> summary of insights

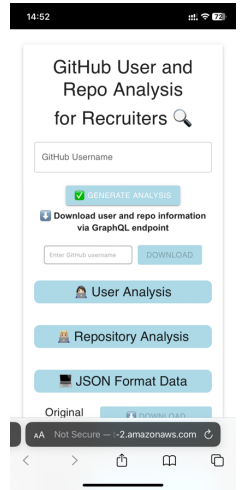
Accessibility:

look at the page directly or make API calls -> interactive frontend to present the analysis result + provide downloadable data

Transparency:

difficulty understanding the field -> detailed explanation documentation

How you decided to improve the structure



Portability improvement:
Interoperability:

JSON format only, may not work well on mobile devices

-> typescript for typing and data schema,

-> React for interoperability on web and mobile, more data format such as CSV

Usability:

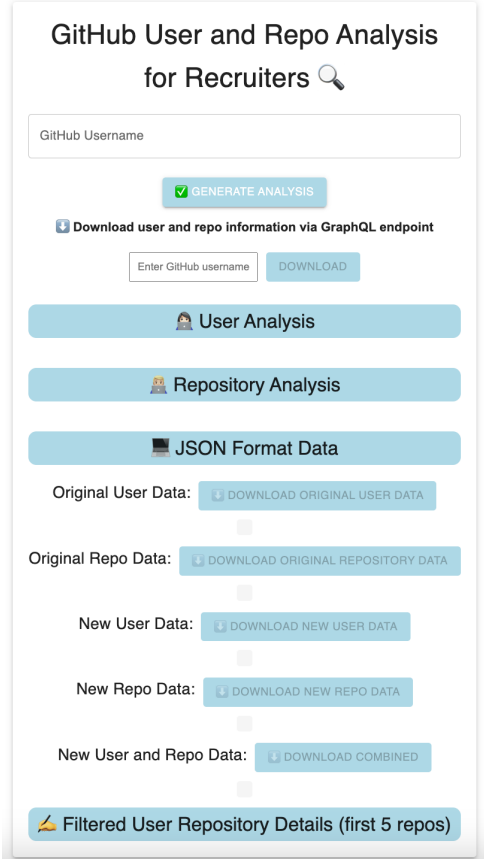
difficult to use directly for candidate assessment -> analyze based on raw data and generate deeper insights on the user and their repos

Structure:

lack of summary and structure -> combine user and repo insights in one JSON file

What your new structure is

- **GitHub User and Repo Analysis for Recruiters (hosted on Amazon S3)**
- http://githubanalysiseve.s3-website-ap-southeast-2.amazonaws.com/



What your new structure is

- **Data manipulation and transformation**
- See code!

What your new structure is

Information itself:

Existing information structure:

too much information

New information structure:

1. Information has been trimmed

-> retain information that is useful to recruiters in analyzing a candidate's programming skill level.

2. Added new fields: e.g.: the time of the user on GitHub

-> analyzed and summarized information from the user account and all the repositories' information.

What your new structure is

Structure/format:

Existing information structure:

raw JSON data directly retrieved from the GitHub API, includes overall details fields that may not be relevant to the recruiters (e.g. node IDs or avatar URLs)

New information structure:

filtered and summarized data that includes only relevant fields (e.g. user name, repository name)

and additional computed fields (e.g. total time the user has been on GitHub and fields about their programming skill levels)

-> providing more relevant information to the recruiters.

What your new structure is

Access methodology:

Existing information structure:

- Directly access the GitHub webpage
- Manual API calls

New information structure:

- Web application hosted on Amazon S3
- GraphQL API calls
- REST API calls

How would quality be identified and addressed

Lasting change:

- Provide **downloadable** original and new information structure

Durable and Robust Application:

- Apply rigorous **unit-testing**, including:

  - Sending requests - > Hitting the GitHub API endpoint
  - Getting responses -> Rendering components on the frontend based on responses

- Use TypeScript unit-testing frameworks with a goal of achieving 99%+ test coverage.

Application and data security

- Use an **MIT license** for the code to ensure open-source accessibility.
- The application is hosted on **Amazon S3 bucket** for security

Future improvements

- Storage of past user and repo performance (a detailed timeline of the GitHub user's performance)
- More personalized GraphQL endpoint usage, allowing users to extract the data they want
- Data security

Q & A

Thank you! :)