Design and implement a production-ready crawling application to efficiently access review data from below given business profile sites using available data access options on the site such as API, HTML content or GraphQL etc.

1. https://www.glassdoor.com/Reviews/Employee-Review-BAYADA-Home-Health-Care-RVW51285938.htm
2. https://www.facebook.com/workingwithnaturenotagainstit/reviews
3. https://www.yelp.com/biz/mathis-brothers-furniture-tulsa

The application should exhibit the following behavior:

- Successfully retrieve **all reviews in sorted order of review date [recent to old]** from the provided URLs.
- **Optionally retrieve reviews** up to a specific review date.
- Return an **error response in case of any network issues**.

The application should expose an API with the following request and response attributes:

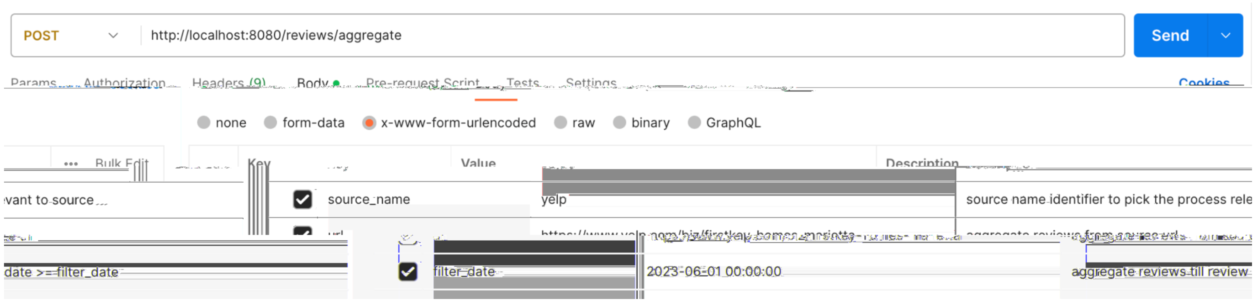
API Endpoint to be Exposed:

Example : http://localhost:8080/reviews/aggregate

API Request Query Params:

- Source URL
- Source Name
- Filter Date (**optional field**: if provided, aggregate reviews till review date >= filter date)

Example :



POST http://localhost:8080/reviews/aggregate

Body: x-www-form-urlencoded

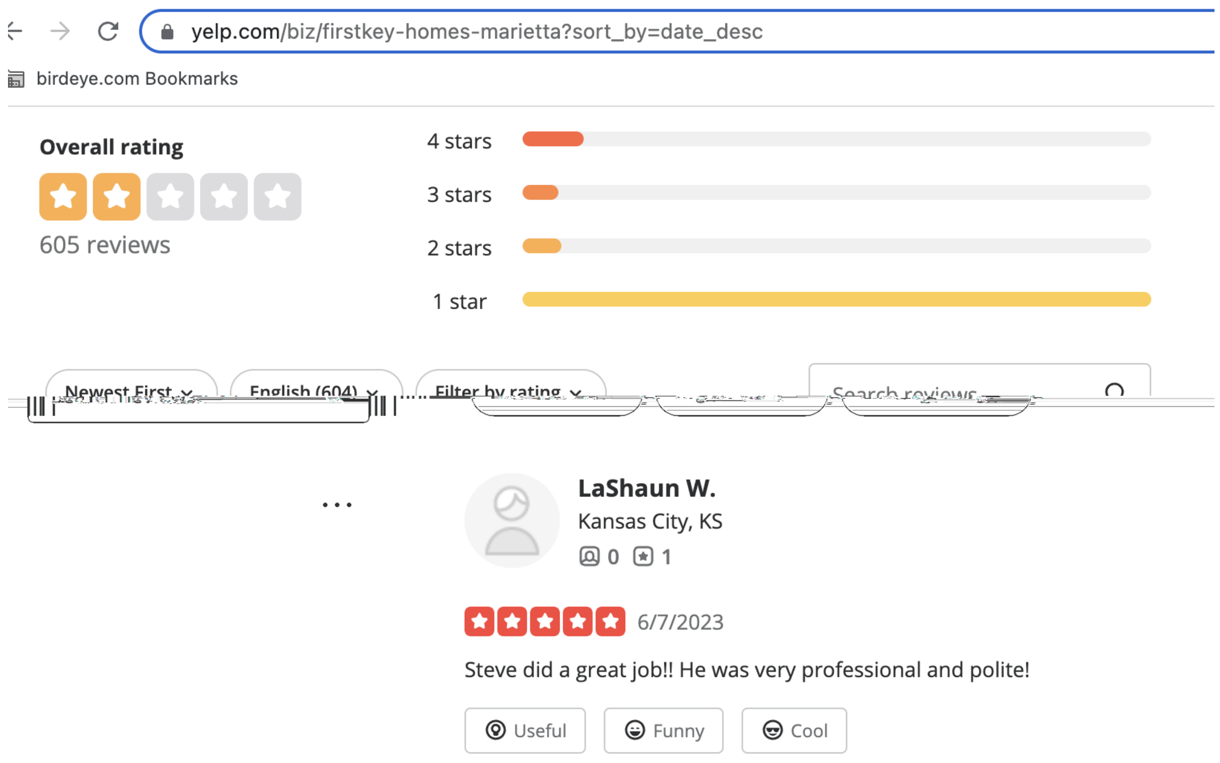
	Key	Value	Description
✓	source_name	yelp	source name identifier to pick the process rele
✓	url	[illegible]	aggregate reviews from source url
✓	filter_date	2023-06-01 00:00:00	aggregate reviews till review date >= filter_date

API Response Object fields:

- Total count of reviews
- List of all reviews (**or reviews up to a certain review date if the filter_date param input is provided**).
- Each review data should include below given attributes:
  - Comment
  - Date

        - Reviewer's Name
        - Review Rating
    - Error response in case of an issue

Example :


yelp.com/biz/firstkey-homes-marietta?sort_by=date_desc

birdeye.com Bookmarks

Overall rating

605 reviews

4 stars
3 stars
2 stars
1 star

Newest First
English (604)
Filter by rating
Search reviews

LaShaun W.
Kansas City, KS
0 1

6/7/2023

Steve did a great job!! He was very professional and polite!

<u>Success Response :</u>

```
{
    “review_count” : 605,
    “aggregated_reviews” : [
        {
            “rating” : 5, “review_date”: “2023-06-07”, “reviewer_name”:”LaShaun W.”,
            “comment”: “Steve did a great job!! He was very professional and polite!”
        }
        …
  ] ,
 //Sorted Order Of Review Date From Recent To Older
```

```
    “review_aggregated_count” : 605

    “response_code”: 200

}
```

Failure Response :

```
{

    “response_code”: 400,

   “error” : {

        “message”: “Network Issue”

    }

}
```

Key areas of focus:

- Design the application in a modular manner to enhance flexibility.
- Strive for code efficiency.
- Handle errors and exceptions effectively.
- Prioritize code readability and maintainability.
- Implement fail-safe measures for robustness.
- Handle scenarios involving site blocking.

Frameworks to consider:

- Node.js
- Puppeteer (as a headless browser plugin for crawling) or any other suitable framework.

Submission Guidelines:

- Provide the code for the crawling application in a GitHub repository. Make sure your repo is publicly accessible.
- Include clear instructions on how to run the application in the README file of the GitHub repository. These instructions should outline the necessary setup steps, dependencies, and any specific commands or configurations required to successfully execute the crawling application