SC4052 - Assignment 2 - Choice 2: Study basics of API in Github and Code Search.

Name: Lim Song Wei, Greg
Matriculation Number: U2120517G

GitHub Repository: https://github.com/Greg-Lim/SC4052-Assignment-2

Demo Video: https://youtu.be/Tx5FrBH0ZfU?si=XRqgAR_z_hyFEFTK

Table of Contents

1 Problem Statement

As a new developer, it may be daunting to work on existing codebases. They want to practice writing new pull requests to existing repositories, but are unsure of how to do so. Sometimes, they need a tool to help them practice writing PRs base on other people's existing successful PRs.

The saas aim to solve this problem by providing a simple search tool to find PRs that have already been merged and closed to practice on. Additionally, it provides steps to start working on the PR, hints on how to fix the issues based on your current commits and the original commits. When the user is satisfied with the iteration on the forked repo, they can get perform a check in which an AI will review the user code with the original code and provide suggestions on how to improve the code.

2 Architecture

As this project is a simple Saas that uses GitHub Search API and OpenAI API, and there is no persistent data, there is no backend.

The frontend is build on ReactJS. There are 2 pages in the front end; the search page that users can find completed merged PRs to practice on, and the Issue Page that users can check their code against the original code.

2.1 Search Page

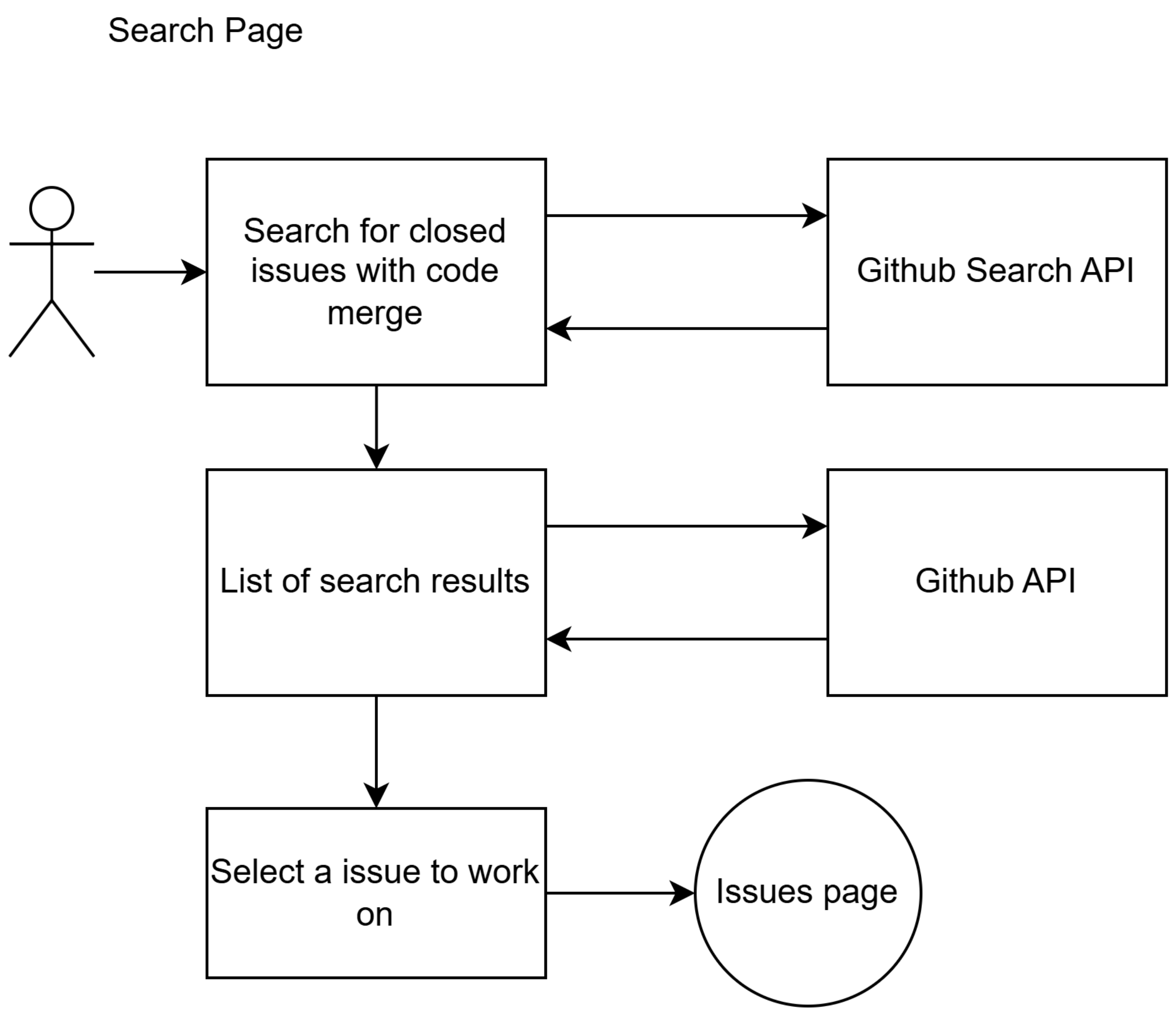


Figure 1: Search Page Flow

In the search page, users can search for PRs that have been merged and closed. The user can enter a keyword to search for PRs that are related to the keyword. The user can also select the language of the PRs to filter the results.

The Search Page, built as a React component, offers a streamlined interface where users can authenticate with their GitHub Personal Access Token, enabling access to the GitHub Search API with increased rate limits. Once authenticated, users can:

- **Search for Issues:**
  Enter keywords, select programming languages, or specify repositories to filter closed issues with merged pull requests. The search query dynamically incorporates these parameters along with a filter for relevant PRs.
- **View Results:**
  The interface lists issues with details, such as:
  - Title with a direct GitHub link
  - Issue number, repository name, and associated labels (color-coded)
  - A brief, truncated description
  - The count of associated pull requests
  - An estimated change complexity (e.g., "Trivial < 10 lines")
- **Practice Contributions:**
  A "Practice This Issue" button for each issue opens a dedicated page, allowing users to simulate contributing by solving issues based on real-world examples.

2.2 Issue Page

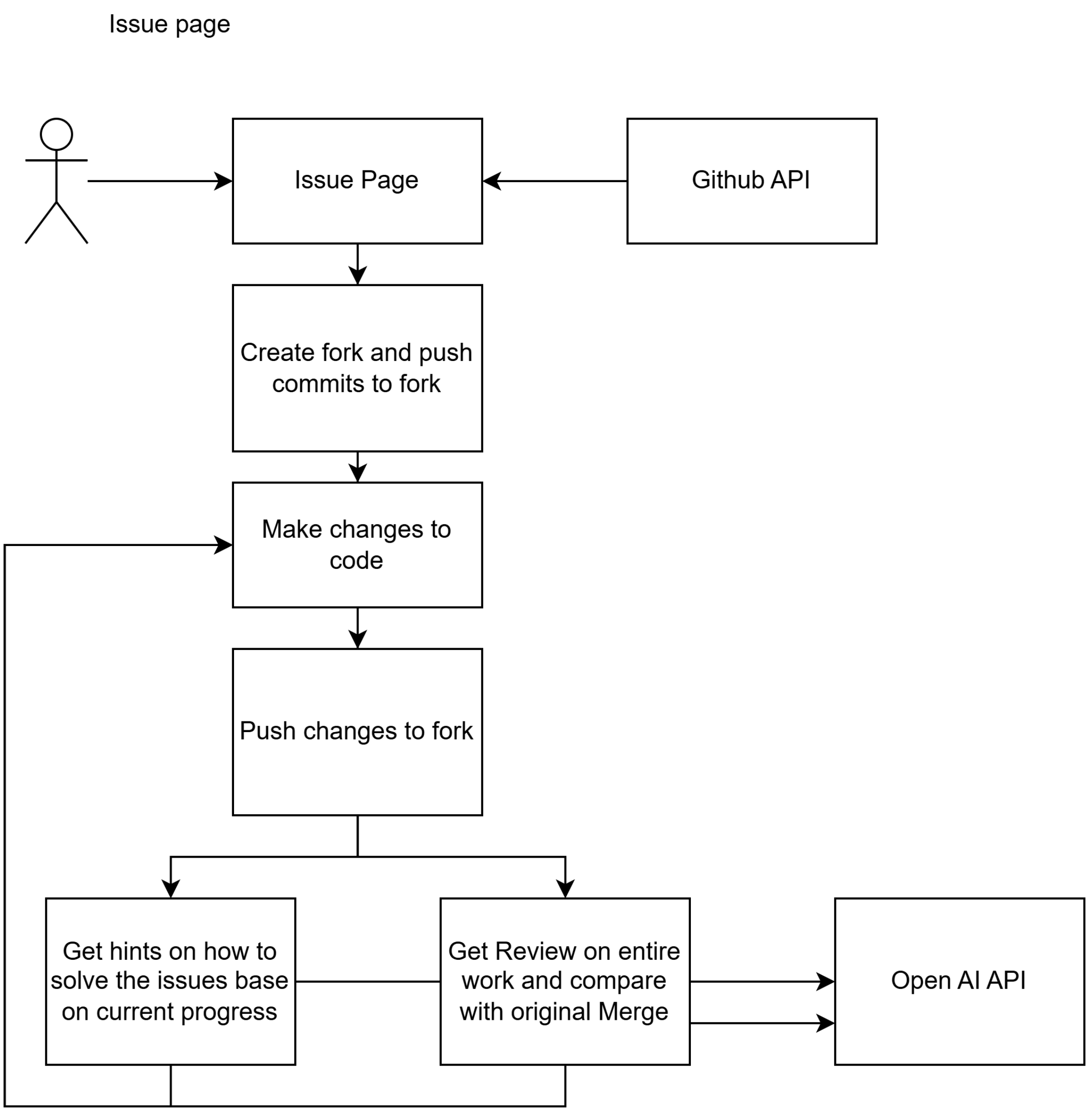

Issue page
Issue Page
Github API
Create fork and push commits to fork
Make changes to code
Push changes to fork
Get hints on how to solve the issues base on current progress
Get Review on entire work and compare with original Merge
Open AI API

Figure 2: Issue Page Flow

The Issue Page provides an interactive environment for users to practice solving real GitHub issues and receive feedback on their solutions. It is composed of two main panels: Practice Instructions and PR Review Panel.

The Issue Page is made up of two main panels: the Practice Instructions panel and the PR Review panel. This design allows users to practice solving real GitHub issues while receiving feedback on their solutions.

Practice Instructions Panel

The Practice Instructions panel guides users through the process of forking a repository, cloning it, and preparing their environment for coding. It includes the following features:

- **Step-by-step Workflow:**
  Users are guided through the process of forking the repository, cloning it, checking out the commit before the fix, and creating a new branch for their solution.
- **Files Changed & Hints:**
  The panel lists the files changed in the original PR, with the number of additions and deletions, helping users focus on relevant parts of the codebase. Users can also view hints, such as the number of commits and files changed in the original solution.
- **Additional infomation:**
  The panel provides additional information about the issue, including the title, description (rendered with markdown), labels, repository, and any associated comments. It also lists all pull requests that have been merged to resolve the issue, including their titles, numbers, and descriptions.

PR Review Panel

The PR Review panel allows users to compare their solution with the original pull request and receive feedback. It includes the following features:

- **Repository and Commit Comparison:**
  Users can input their forked repository and a commit SHA to compare their solution with the original PR using GitHub's compare API.
- **Automated Hints:**
  The panel can generate up to three AI-powered hints, each becoming more specific, to help users align their solution with the original PR.
- **AI Code Review:**
  After comparing their code, users can request an AI-generated review. The AI provides a concise assessment, specific feedback, and suggestions for improvement, referencing the original PR.
- **User Experience:**
  The Issue Page is designed to be informative and supportive, providing clear instructions, actionable hints, and constructive feedback. It encourages learning by allowing users to iteratively improve their solutions and compare them with real-world PRs.

Workflow Summary:

1. User selects an issue to practice.
2. The page loads the issue details, associated PRs, and comments.
3. The Practice Instructions panel guides the user through forking, cloning, and preparing the repo.
4. The user implements their solution and pushes it to their fork.
5. In the PR Review Panel, the user compares their solution with the original, receives hints, and can request an AI review.
6. The user iterates on their solution based on feedback and hints.

This approach provides a hands-on, feedback-driven learning experience, closely simulating real open-source contribution workflows.

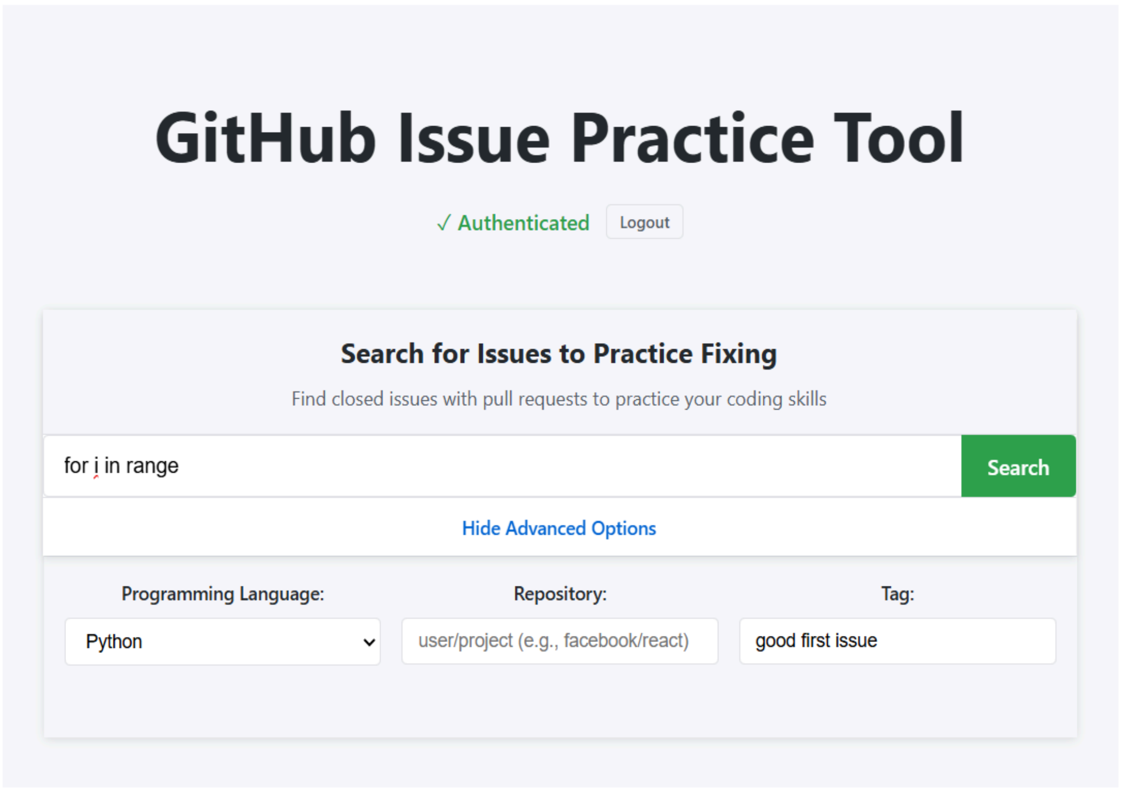
2.3 API Integration

The project leverages the GitHub REST API v3 via the Octokit JavaScript client and the OpenAI API to provide a rich, interactive learning experience:

- **GitHub API (via Octokit):**
    - Authenticates requests using a stored GitHub Personal Access Token.
    - Searches issues and pull requests with the `GET /search/issues` endpoint.
    - Retrieves detailed pull request information with `GET /repos/{owner}/{repo}/pulls/{pull_number}` to assess change complexity and gather file diffs.
    - Fetches associated comments, commit logs, and changed files for each PR to provide comprehensive context.
    - Ensures compatibility with the latest API features by setting the `X-GitHub-Api-Version` header.
- **OpenAI API:**
    - Integrates with the OpenAI API (using a user-provided API key) to generate AI-powered hints and code reviews.
    - In the PR Review Panel, users can request up to three progressively specific hints, each generated by prompting the OpenAI API with the context of the issue, the original PR diff, and the user's code changes.
    - After submitting their solution, users can request an AI-generated code review. The OpenAI API analyzes the differences between the user's implementation and the original PR, providing concise, constructive feedback and suggestions for improvement.

This dual integration ensures that practice material is always up-to-date, and that users benefit from both authentic code changes and personalized, AI-driven feedback to accelerate their learning.

3 User Interface


GitHub Issue Practice Tool

✓ Authenticated Logout

Search for Issues to Practice Fixing

Find closed issues with pull requests to practice your coding skills

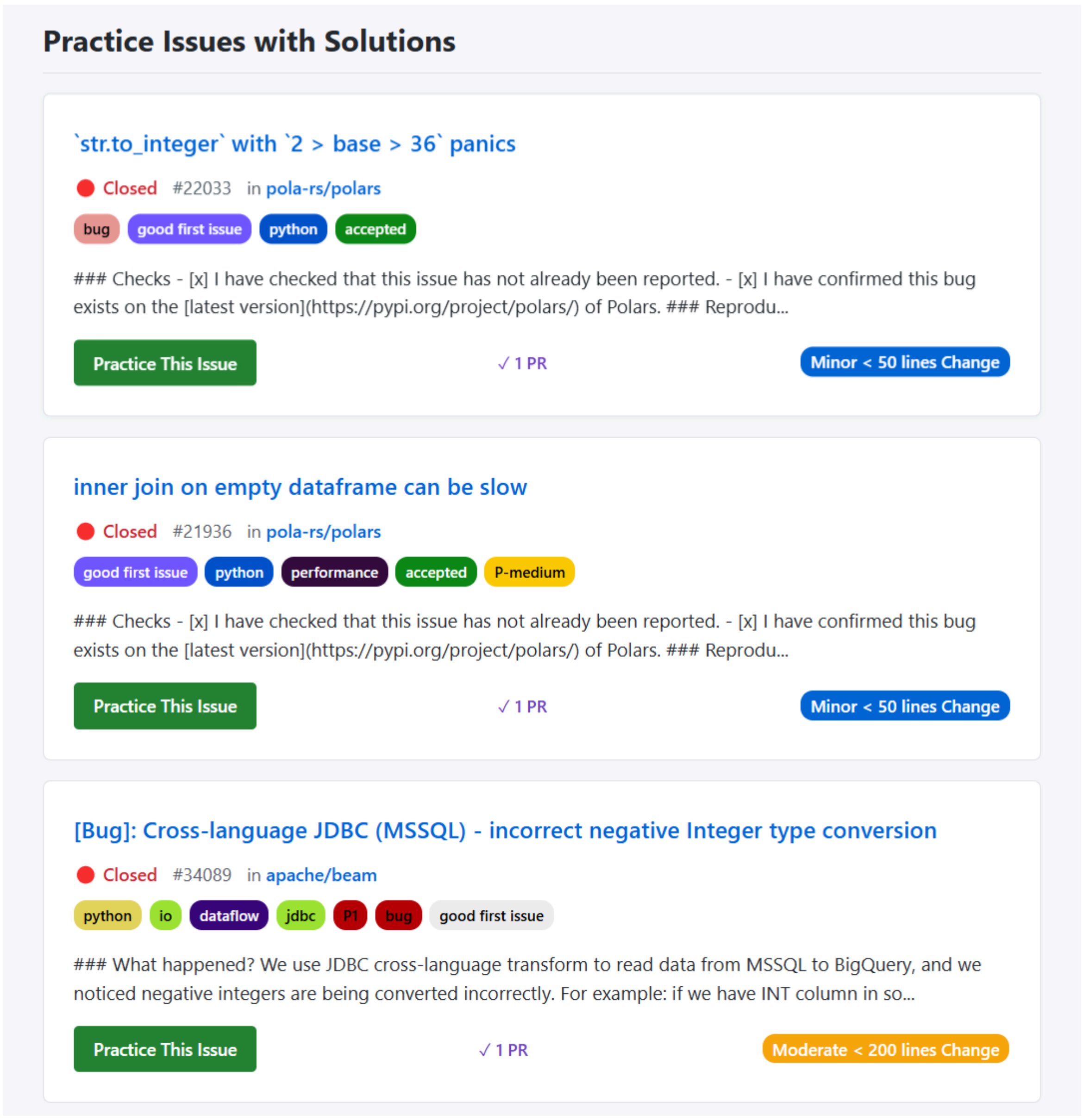
for i in range

Programming Language: Python

Repository: user/project (e.g., facebook/react)

Tag: good first issue

Figure 3: Search Bar UI



Practice Issues with Solutions

`str.to_integer` with `2 > base > 36` panics

● Closed #22033 in pola-rs/polars

bug | good first issue | python | accepted

Checks - [x] I have checked that this issue has not already been reported. - [x] I have confirmed this bug exists on the [latest version](https://pypi.org/project/polars/) of Polars. ### Reprodu...

inner join on empty dataframe can be slow

● Closed #21936 in pola-rs/polars

good first issue | python | performance | accepted | P-medium

Checks - [x] I have checked that this issue has not already been reported. - [x] I have confirmed this bug exists on the [latest version](https://pypi.org/project/polars/) of Polars. ### Reprodu...

[Bug]: Cross-language JDBC (MSSQL) - incorrect negative Integer type conversion

● Closed #34089 in apache/beam

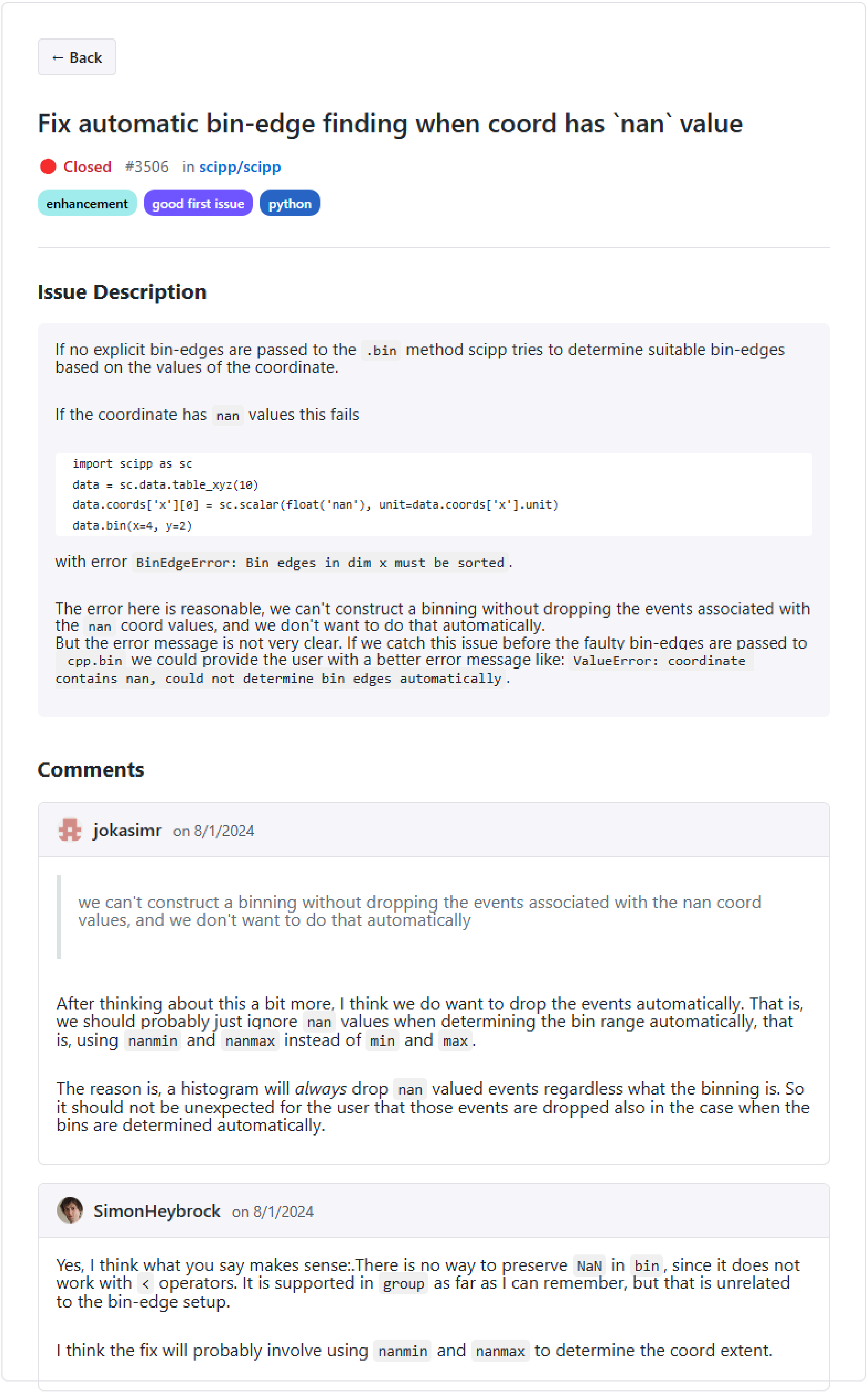
python | io | dataflow | jdbc | P1 | bug | good first issue

What happened? We use JDBC cross-language transform to read data from MSSQL to BigQuery, and we noticed negative integers are being converted incorrectly. For example: if we have INT column in so...

Figure 4: Search Results UI

Features of the Search Page UI include:

- **Search Bar:** Users can enter keywords and select programming languages to filter issues.
- **Search Filter:** Users can filter issues by programming language, repository, and Tag.
- **Search Results:** The results are displayed in a list format, showing the title, issue number, repository name, labels, a brief description, and the number of associated pull requests and the estimated change complexity.
- **Practice This Issue Button:** Each issue has a button that redirects users to the Issue Page for practice.



← Back

Fix automatic bin-edge finding when coord has `nan` value

● Closed #3506 in scipp/scipp

enhancement good first issue python

Issue Description

If no explicit bin-edges are passed to the `.bin` method scipp tries to determine suitable bin-edges based on the values of the coordinate.

If the coordinate has `nan` values this fails

```
import scipp as sc
data = sc.data.table_xyz(10)
data.coords['x'][0] = sc.scalar(float('nan'), unit=data.coords['x'].unit)
data.bin(x=4, y=2)
```

with error `BinEdgeError: Bin edges in dim x must be sorted`.

The error here is reasonable, we can't construct a binning without dropping the events associated with the `nan` coord values, and we don't want to do that automatically.
But the error message is not very clear. If we catch this issue before the faulty bin-edges are passed to `cpp.bin` we could provide the user with a better error message like: `ValueError: coordinate contains nan, could not determine bin edges automatically`.

Comments

jokasimr on 8/1/2024

> we can't construct a binning without dropping the events associated with the nan coord values, and we don't want to do that automatically

After thinking about this a bit more, I think we do want to drop the events automatically. That is, we should probably just ignore `nan` values when determining the bin range automatically, that is, using `nanmin` and `nanmax` instead of `min` and `max`.

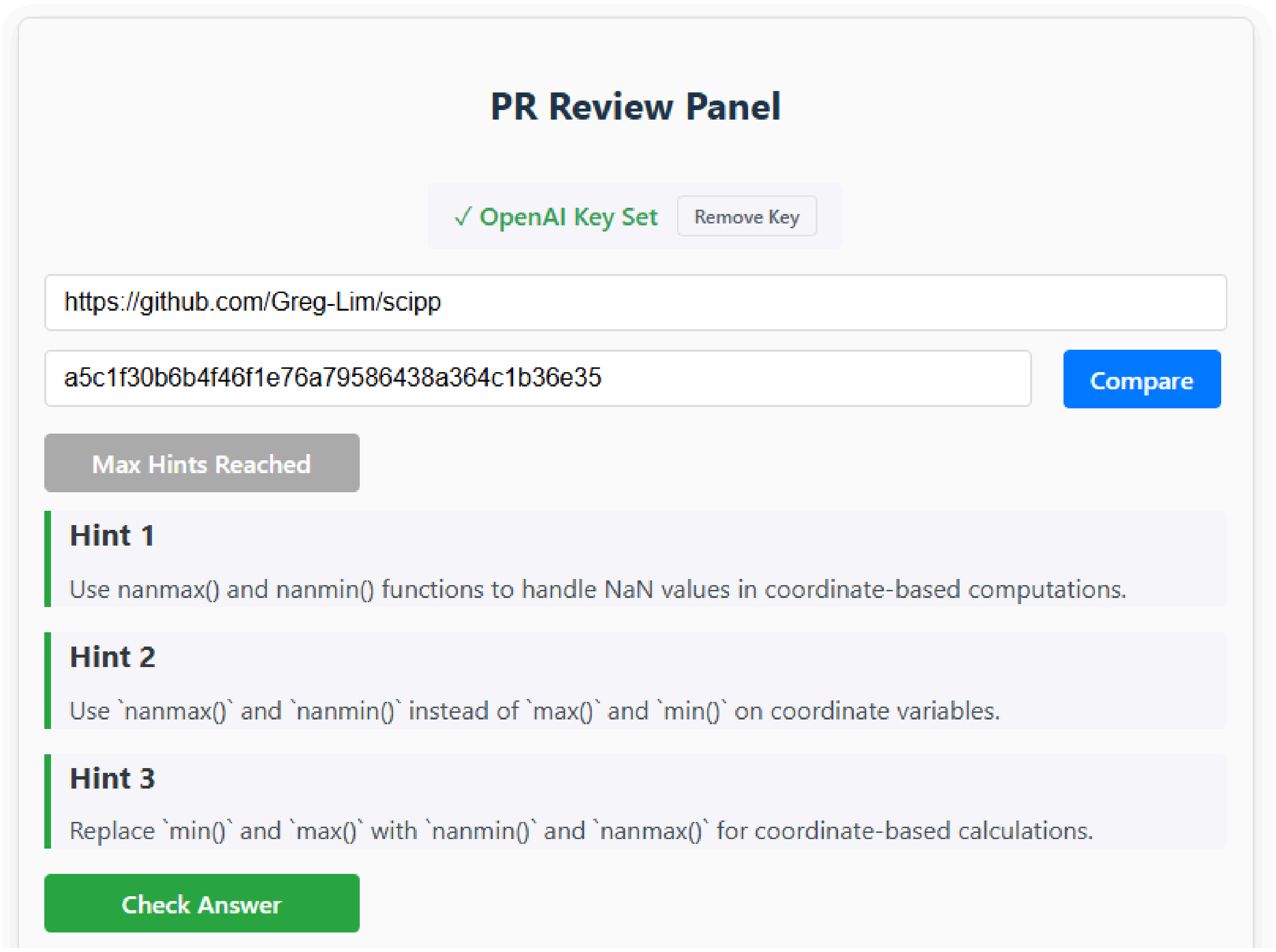
The reason is, a histogram will *always* drop `nan` valued events regardless what the binning is. So it should not be unexpected for the user that those events are dropped also in the case when the bins are determined automatically.

SimonHeybrock on 8/1/2024

Yes, I think what you say makes sense:.There is no way to preserve `NaN` in `bin`, since it does not work with `<` operators. It is supported in `group` as far as I can remember, but that is unrelated to the bin-edge setup.

I think the fix will probably involve using `nanmin` and `nanmax` to determine the coord extent.

Figure 5: Issue Description Page UI



PR Review Panel

✓ OpenAI Key Set Remove Key

https://github.com/Greg-Lim/scipp

a5c1f30b6b4f46f1e76a79586438a364c1b36e35 Compare

Max Hints Reached

Hint 1

Use nanmax() and nanmin() functions to handle NaN values in coordinate-based computations.

Hint 2

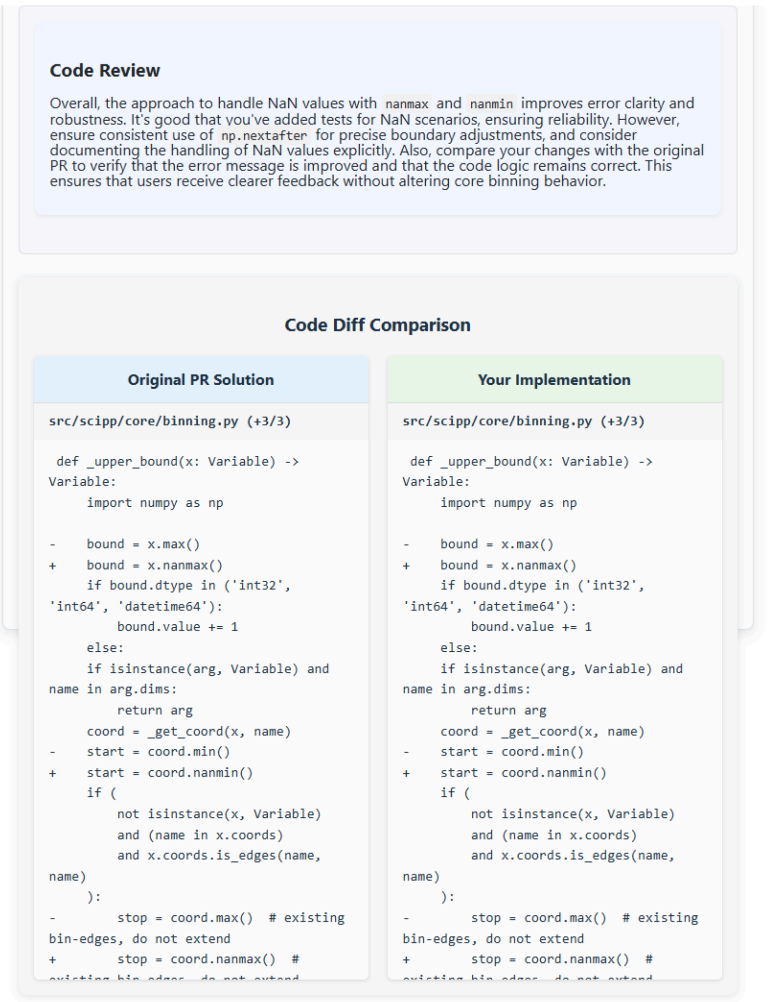
Use `nanmax()` and `nanmin()` instead of `max()` and `min()` on coordinate variables.

Hint 3

Replace `min()` and `max()` with `nanmin()` and `nanmax()` for coordinate-based calculations.

Check Answer

Figure 6: Hints UI



Code Review

Overall, the approach to handle NaN values with `nanmax` and `nanmin` improves error clarity and robustness. It's good that you've added tests for NaN scenarios, ensuring reliability. However, ensure consistent use of `np.nextafter` for precise boundary adjustments, and consider documenting the handling of NaN values explicitly. Also, compare your changes with the original PR to verify that the error message is improved and that the code logic remains correct. This ensures that users receive clearer feedback without altering core binning behavior.

Code Diff Comparison

Original PR Solution

```
src/scipp/core/binning.py (+3/3)

 def _upper_bound(x: Variable) ->
Variable:
     import numpy as np

-    bound = x.max()
+    bound = x.nanmax()
     if bound.dtype in ('int32',
'int64', 'datetime64'):
         bound.value += 1
     else:
     if isinstance(arg, Variable) and
name in arg.dims:
         return arg
     coord = _get_coord(x, name)
-    start = coord.min()
+    start = coord.nanmin()
     if (
         not isinstance(x, Variable)
         and (name in x.coords)
         and x.coords.is_edges(name,
name)
     ):
-        stop = coord.max()  # existing
bin-edges, do not extend
+        stop = coord.nanmax()  #
```

Your Implementation

```
src/scipp/core/binning.py (+3/3)

 def _upper_bound(x: Variable) ->
Variable:
     import numpy as np

-    bound = x.max()
+    bound = x.nanmax()
     if bound.dtype in ('int32',
'int64', 'datetime64'):
         bound.value += 1
     else:
     if isinstance(arg, Variable) and
name in arg.dims:
         return arg
     coord = _get_coord(x, name)
-    start = coord.min()
+    start = coord.nanmin()
     if (
         not isinstance(x, Variable)
         and (name in x.coords)
         and x.coords.is_edges(name,
name)
     ):
-        stop = coord.max()  # existing
bin-edges, do not extend
+        stop = coord.nanmax()  #
```

Figure 7: Code Review UI

The Issue Page UI consists of two main panels: the Practice Instructions panel and the PR Review panel.

The Practice Instructions panel includes:

- **Repository Information:** Displays the repository name, issue number, and title.
- **Description:** Renders the issue description using markdown.
- **Labels:** Displays the labels associated with the issue, color-coded for easy identification.
- **Comments:** Lists all comments associated with the issue, including the comment author and date.
- **Files Changed:** Lists the files changed in the original PR, with the number of additions and deletions.
- **Hints:** Provides hints on how to fix the issue based on the original PR.
- **Instructions:** Provides step-by-step instructions on how to fork the repository, clone it, and prepare the environment for coding.

The PR Review panel includes:

- **Repository and Commit Comparison:** Users can input their forked repository and a commit SHA to compare their solution with the original PR using GitHub's compare API.
- **Automated Hints:** Users can generate up to three AI-powered hints, each becoming more specific, to help them align their solution with the original PR.
- **AI Code Review:** Users can request an AI-generated review, which provides a concise assessment, specific feedback, and suggestions for improvement, referencing the original PR.
- **Checking Answer:** Users can check their answer by comparing their solution with the original PR and receiving hints and feedback.

4 Prompt Engineering and LLM Functions

A core feature of this project is the use of prompt engineering to generate context-aware hints and code reviews using the OpenAI API. The PR Review Panel is designed to guide users through improving their code by providing AI-generated feedback at different stages.

4.1 Progressive Hint Generation

The system generates up to three hints for each user submission. Each hint is created by sending a tailored prompt to the OpenAI API, which includes:

- The context of the GitHub issue and pull request
- A summary of the original PR's code changes
- The user's current code changes
- All previously generated hints

The prompt is dynamically adjusted based on the number of hints already provided:

- The first hint is vague and simple
- The second hint is more specific
- The third hint is very specific and actionable

This progressive approach helps users iteratively improve their solution without immediately revealing the full answer, simulating a real-world mentoring process.

4.2 AI Code Review

After the user submits their solution, the system sends another prompt to the OpenAI API to request a code review. This prompt includes the issue context, the original PR's code changes, and the user's code changes. The AI is instructed to provide a concise review, including an overall assessment, specific feedback, and suggestions for improvement, The prompt ensures the feedback is constructive and educational.

4.3 Experimentation and Results

Through experimentation with prompt phrasing and structure, the system achieves a balance between helpfulness and challenge. By referencing both the original and user code, the AI delivers targeted feedback that is relevant to the user’s progress. This demonstrates the effectiveness of prompt engineering in creating an interactive, scalable, and supportive learning environment for open-source contributions.

5 Conclusion

This project demonstrates how the GitHub REST API and OpenAI API can be combined to create an interactive SaaS platform for practicing open-source contributions. By enabling users to search for real, merged pull requests and practice replicating or improving them, the platform addresses challenges faced by new developers when contributing to existing codebases.

The implementation uses prompt engineering and LLM functions to provide context-aware, AI-generated hints and code reviews. The PR Review Panel generates up to three progressively specific hints and offers concise, constructive feedback after users submit their solutions. This ensures feedback is relevant and actionable, enhancing the learning experience.

Experimentation with prompt design and API integration shows that combining real GitHub data with AI-powered feedback creates a scalable and engaging learning environment. The ReactJS frontend guides users from searching for issues to receiving AI-driven reviews.

Overall, the project meets the assignment requirements by demonstrating technical novelty, sound experimentation, and clear presentation. It provides a practical tool for new developers to build confidence and skills in open-source contribution, while showcasing the power of modern APIs and prompt engineering in educational SaaS applications.

6 Appendix

Sample Hint Prompt

> You are generating the third hint, be very specific and helpful.
>
> Here are the previous hints you have generated:
> Hint 1: Use nanmax() and nanmin() functions to handle NaN values in coordinate-based computations.
>
> Ensure that the new hint is different and more specific than the previous ones.
>
> Here is the context of the PR:
> {Original PR Context}
>
> Here is the solution given by the original PR:
> {Original Commit Diff}
>
> Here are the code changes made by the user
> {Your Code Changes}
>
> Generate a 1 short sentance hint for the user to help the user code resemble the original PR solution.
> keep the solution to less than 20 words.

Sample Code Review Prompt

> You are a code reviewer analyzing the code diff of the actual PR solution and a solution I am working on. Please review the following code changes:
>
> Here is the context of the PR:
> {Original PR Context}
>
> Here is the solution given by the original PR:
> {Original Commit Diff}
>
> Here are the code changes made by the user
> {Your Code Changes}
>
> Provide a detailed review with:

> 1. An overall assessment (2-3 sentences)
> 2. Specific feedback on what's good and what could be improved
> 3. Suggestions for improvement referencing the original code
>
> Keep your response focused on constructive feedback that helps the user learn.
> Keep the response short and concise, less than 100 words.