Part A
Project Overview

Group 06

Team Members:
G M Shahariar
Md Taukir Azam Chowdhury
Md. Olid Bhuiyan
Samiha Khan
Zabir Al Nazi

Date of Submission:
February 14th, 2025

Table of Content

Project Overview

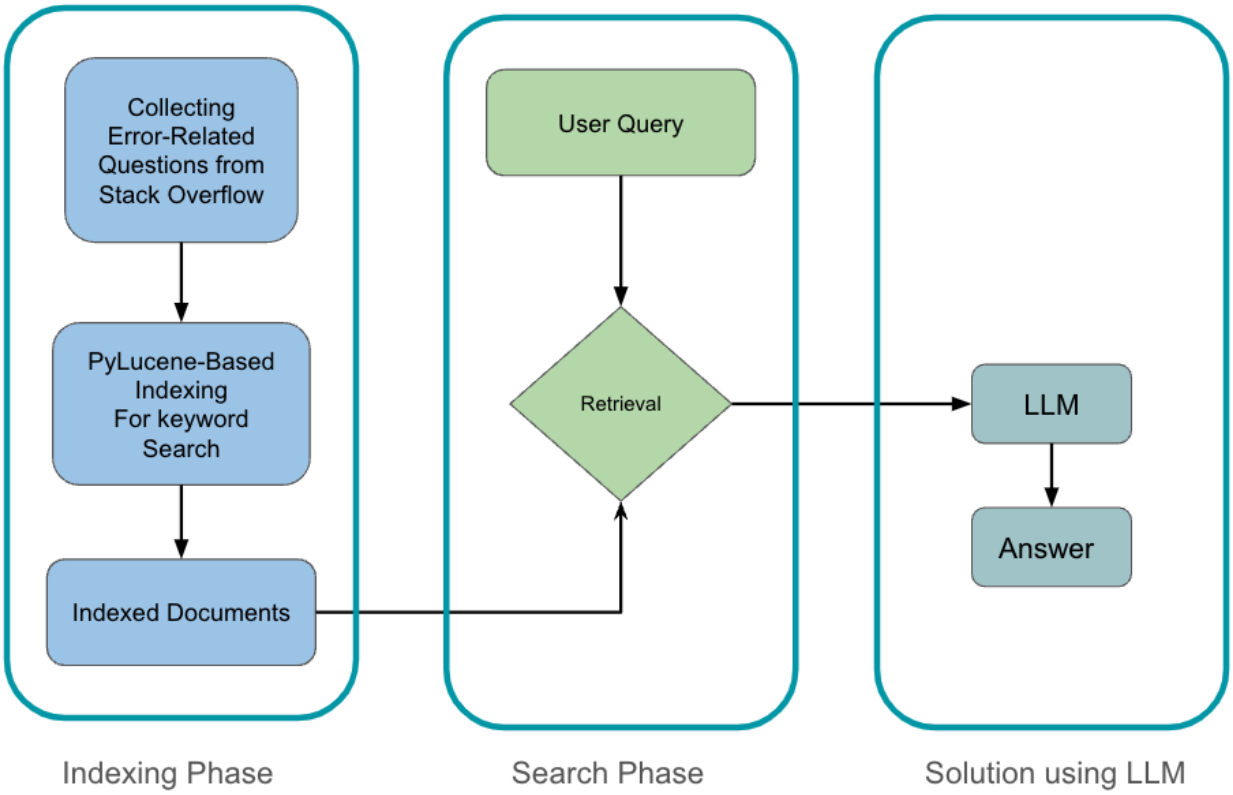
Short Description

We aim to assist developers in resolving programming errors efficiently using AI and data from StackOverflow without manually testing multiple solutions.

Project Phases
(1) Indexing Phase:
We will collect error-related questions and their associated comments (containing solutions) from the StackOverflow website. The collected questions will be indexed using a traditional PyLucene-based index for keyword search. This method will enable efficient retrieval of questions relevant to the user query.

(2) Search Phase: A user interface will allow users to input error descriptions (queries). Using indexing, we will retrieve relevant questions. The index of the retrieved questions will easily provide the corresponding comments (candidate solutions).

(3) Solution using LLM: we will give the comments to an LLM. The LLM will summarize the comments and return a final solution. Finally, we will evaluate the performance of the proposed pipeline by using the accepted solutions (ground truth).



Crawler

Data Collected

For data collection, we utilized Scrapy, a powerful and flexible web scraping framework, to extract error-related data from StackOverflow. Our scraping strategy focused on questions tagged with the most frequently occurring error-related tags such as [*"error-handling", "compiler-errors","runtime-error","syntax-error","typeerror","linker-errors","importerror" etc*]. This approach ensured that we captured a diverse and representative dataset of programming errors commonly encountered by developers.

In total, we collected **777.798 mb** of data, which includes:

- 224046 unique error-related entries,
- Their associated tags, and
- Corresponding user-provided solutions (extracted from comments and answers).

We also conducted a duplicate-data analysis for our data. The result is given below:

```
cs242@class-046:~/cs242$ python3 duplicate_analysis.py

📊 Duplicate Analysis Results:
🔹 Total Entries: 226340
🔹 Unique Entries: 224046
🔹 Duplicate Entries: 2294
🔹 Duplicate Percentage: 1.01%
🔹 Entries to be Removed (if deduplication is applied): 2294
```

Architecture & Crawling Strategy

Our goal was to extract programming error questions along with their solutions (user comments and answers), focusing on questions tagged with common error-related keywords.

Tag-Based Crawling Strategy

We began by identifying a list of the most frequently used error-related tags on StackOverflow (e.g., database errors, compiler errors, recursion issues). These tags served as entry points for the crawler, allowing us to target specific categories of programming errors.
For each error tag:

- The crawler generated a URL corresponding to StackOverflow's tag-specific pages (e.g., https://stackoverflow.com/questions/tagged/mysql-error-1292).
- It systematically visited each page under that tag to extract links to individual questions.

Pagination Handling

Since StackOverflow organizes questions across multiple pages for each tag, we implemented a pagination mechanism. This ensured that the crawler didn't stop at the first page but continued to follow "Next" page links until all questions under each tag were collected. This approach maximized data coverage.

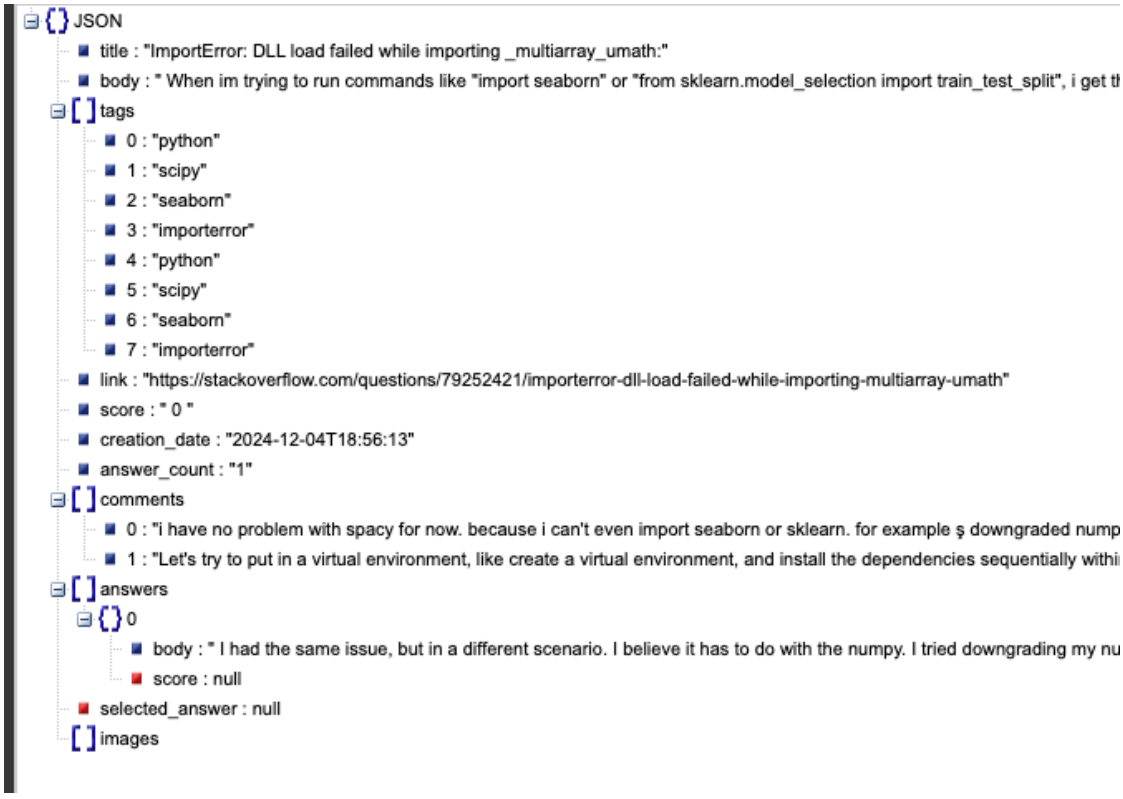
Data Extraction Process

Once a question page was reached, the crawler extracted several key data points:

- Question Details: Title, detailed description (body), associated tags, and the date it was posted.
- User Interactions: All answers provided, including their vote scores, and user comments that often contain quick fixes or clarifications.
- Accepted Solutions: If the question had an accepted answer, the crawler flagged it as a reliable solution.
- Additional Metadata: Information such as the question's score (upvotes), number of answers, and any embedded images related to the error.

This structured extraction ensured we captured both the problem description and potential solutions for each error.

Sample Data:



Difficulties, Solutions, Limitations

Limitation 01: Scraping 500 mb of data is time consuming.

Solution: Since we have 5 members in our team, we divided our error-tags in 5 different sets, and ran the code separately in our own machine. Since we were using different seeds, we collected different data based on those seeds. It made the whole process 5 times faster.

Limitation 02: Crawling Stack Overflow too aggressively could trigger rate limiting or blocking from the server.

Solution: In the **settings.py** file of the scrapy project, we added the following modifications to avoid blocking from the server -

DOWNLOAD_DELAY = 1

CONCURRENT_REQUESTS = 8

CONCURRENT_REQUESTS_PER_DOMAIN = 4

Limitation 02: If any CSS selector fails or the expected data is missing, it could cause issues or data loss or any error that may stop the scrapping process.

Solution: In the **settings.py** file of the scrapy project, we added the following modifications to continue scrapping even if we encounter any error -

RETRY_ENABLED = True

RETRY_TIMES = 5

RETRY_HTTP_CODES = [418, 429, 500, 503]

Instructions for Usage

To run the StackOverflow data crawler,

- Clone the project repository to your local machine.
- Navigate to the stackoverflow_spider.py file located in the CS242_Project-master/stackoverflow_scraper/spiders folder and update the error_tags list with the specific error-related tags you want to scrape.
- After updating the error_tags, go to CS242_Project-master/stackoverflow_scraper folder
- Run scrapy crawl stackoverflow_spider --nolog

PyLucene

Index Fields & Justification:

In our PyLucene indexer, we chose fields that are most useful for helping developers quickly find relevant answers. Here's what we included:

TextFields:

- **Title:** The main point of a question, making it crucial for search accuracy.
- **Body:** The detailed description, providing depth for full-text search.
- **Tags:** Useful for narrowing results to specific topics or languages.
- **Comments:** Adds valuable context from user discussions.
- **Answers:** Stores solutions from the community for better query results.

StoredFields:

Link, Score, Creation Date: Stored for display purposes, not search.

We picked these fields because they capture everything a developer might look for when searching for solutions.

Text Analyzer Choices:

We used `StandardAnalyzer` because it's a reliable choice for handling text. It breaks down the text into searchable tokens, removes common stop words, and ensures everything is lowercase for consistent results. It's simple but effective for technical queries.

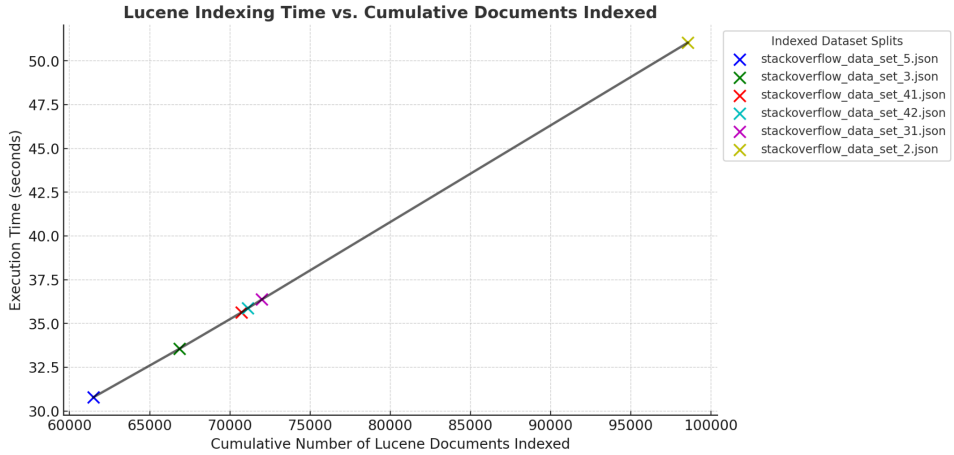
Run-time:

The index construction process involves reading documents, tokenizing text, and storing indexed data. The biggest time factor was building fields for answers and comments since they can be long. Despite that, PyLucene handled the load well, thanks to its efficient indexing system. We ran a timer to calculate the total time required to construct the indexing. The report is shared below:

```
cs242@class-046:~/cs242$ python3 lucene_indexer.py
indexing /home/cs242/group_6/stackoverflow_data_set_5.json (61485 entries)...
finished indexing /home/cs242/group_6/stackoverflow_data_set_5.json! (skipped 1 invalid entries)
Execution time: 30.78350089804735 seconds
indexing /home/cs242/group_6/stackoverflow_data_set_3.json (5362 entries)...
finished indexing /home/cs242/group_6/stackoverflow_data_set_3.json! (skipped 0 invalid entries)
Execution time: 33.546601317997556 seconds
indexing /home/cs242/group_6/stackoverflow_data_set_41.json (3875 entries)...
finished indexing /home/cs242/group_6/stackoverflow_data_set_41.json! (skipped 0 invalid entries)
Execution time: 35.637637828011066 seconds
indexing /home/cs242/group_6/stackoverflow_data_set_42.json (403 entries)...
finished indexing /home/cs242/group_6/stackoverflow_data_set_42.json! (skipped 0 invalid entries)
Execution time: 35.871503165049944 seconds
indexing /home/cs242/group_6/stackoverflow_data_set_31.json (869 entries)...
finished indexing /home/cs242/group_6/stackoverflow_data_set_31.json! (skipped 0 invalid entries)
Execution time: 36.376664867042564 seconds
indexing /home/cs242/group_6/stackoverflow_data_set_2.json (26560 entries)...
finished indexing /home/cs242/group_6/stackoverflow_data_set_2.json! (skipped 0 invalid entries)
Execution time: 51.04684152599657 seconds
```

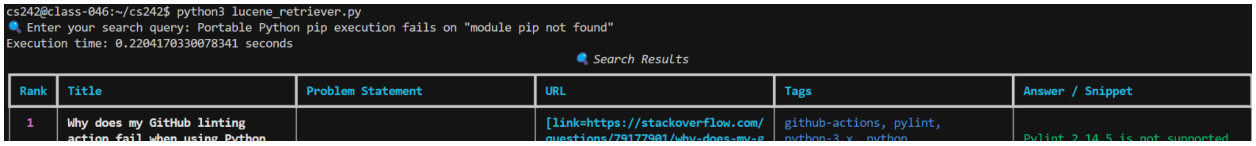
So, the total execution time is approximately **223.26 seconds.**

Here's another graph to demonstrate the data:



It shows the indexing time for each data file depending on the documents it contained.

We also tested how much time it takes for our retriever to fetch the data for any query.



As shown in the screenshot taken from our search query, it takes nearly 0.22 seconds to fetch the data.

Limitations:

- **Skipped Entries:** We logged documents without titles as skipped_entries to keep track of data issues.

- **Memory Usage:** The index size grew quickly because we stored large fields like answers.
- **Search Limitations:** The StandardAnalyzer struggled with code snippets and special characters, which are common in technical discussions.But we have tried to solve this limitation as much as we can by using different input sanitization techniques.

Instructions for Usage:

To Create the Index: *python lucene_indexer.py*
Make sure PyLucene is installed and the JSON file is ready.

LLM-based Answers (Bonus)

We employ the **Qwen 2.5 3B Instruct**, an instruction-following model. It is hosted on Hugging Face's API inference platform (https://huggingface.co/Qwen/Qwen2.5-3B-Instruct), allowing seamless integration with our retrieval pipeline. The model is leveraged to process and generate meaningful answers from the retrieved documents (comments containing solutions). Our intuition is that integrating a Large Language Model (LLM) in the error tag retrieval system will enhance the accuracy and relevance of responses provided to users based on their queries.

To generate relevant responses, the system follows a three-step workflow:

Step 1: Take the user query and documents retrieved by PyLucene as input

Step 2: The query and the retrieved documents are formatted into a structured prompt. We used the following prompt template -

“””Using the information from the following documents, answer the question concisely:

Document 1: {text_of_retrieved_document_1}
Document 2: {text_of_retrieved_document_2}
Document 3: {text_of_retrieved_document_3}
Document 4: {text_of_retrieved_document_4}
Document 5: {text_of_retrieved_document_5}

Question: {user_query}
Answer:
“””

Step 3: The system interacts with the Hugging Face API using the InferenceClient. The model call involves the following steps:

- The prompt is converted to a chat-style conversation template.

- A POST request containing the model name and chat template is sent to the API endpoint.
- The LLM generates a response, leveraging its pre-trained knowledge and the provided context. For generation, the system uses a max token of 1024 tokens.
- The response is streamed in chunks. The system iterates over the output chunks to collect the response gradually. Each chunk is extracted, cleaned (removing None values), and concatenated to form the full response. The final response is presented to the user in a coherent format

A few example test queries are:

1. *INotifyDataErrorInfo ArgumentOutOfRangeException when raising ErrorsChanged event*
2. *I have a table of tags and want to get the highest count tags from the list.*
   Sample data looks like this\n\nid (1) tag ('night')\nid (2) tag ('awesome')\nid (3) tag ('night')\n\n\nusing\n\nSELECT COUNT(*), `Tag` from `images-tags`\nGROUP BY `Tag`\n\n\ngets me back the data I'm looking for perfectly. However, I would like to organize it, so that the highest tag counts are first, and limit it to only send me the first 20 or so.\n\nI tried this...\n\nSELECT COUNT(id), `Tag` from `images-tags`\nGROUP BY `Tag`\nORDER BY COUNT(id) DESC\nLIMIT 20\n\n\nand I keep getting an \"Invalid use of group function - ErrNr 1111\"\n\nWhat am I doing wrong?\n\nI'm using MySQL 4.1.25-Debian\n
3. *I'm using Toga to develop an Android app and I've set an icon for the application, but it's not displaying correctly. Here's what I've done:\nPlaced the Icon File: I have a PNG icon file (F.png) located in the resources directory of my project.\nUpdated the main Function: I specified the icon path in the main function as icon='resources/F.png'.\nBuilt the App: I ran briefcase build android and confirmed that the build was successful.\nDespite these steps, the app icon is not showing up in the Android emulator or device.\n*

A few example queries, and corresponding retrieved documents by PyLucene with LLM generated responses are attached below.

Example 01

```
cs242@class-046:~/cs242$ python3 LLM.py
None of PyTorch, TensorFlow >= 2.0, or Flax have been found. Models won't be available and only tokenizers, configuration and file/data utilities can be used.
Note: Environment variable`HF_TOKEN` is set and is the current active token independently from the token you've just configured.

🔍 Enter your search query (Press Ctrl+D when done):
I have the following table in MySQL (version 5):

id      int(10)        UNSIGNED             No     auto_increment
year    varchar(4)     latin1_swedish_ci    No
title   varchar(250)   latin1_swedish_ci    Yes    NULL
body    text           latin1_swedish_ci    Yes    NULL
And I want the db to auto add the current year on insert, I've tried the following SQL statement:

ALTER TABLE `tips` CHANGE `year` `year` VARCHAR(4) NOT NULL DEFAULT year(now())
But it gives the following error:

1067 - Invalid default value for 'year'
What can I do to get this functionality? Thanks in advance
```

```
🤖 AI-Generated Answer:

You can't use a function as a default value in MySQL. You can use a trigger to set the default value at the time of insert. Here's an example:

CREATE TRIGGER set_year_default BEFORE INSERT ON tips FOR EACH ROW SET NEW.year = YEAR(NOW());


This trigger will set the year column to the current year whenever a new row is inserted into the tips table.
```

```
🔍 Related Stack Overflow Discussions:

Page 1 of 1

                                                    🔍 Search Results
```

Rank	Title	Problem Statement	URL	Tags	Answer / Snippet	Score
1	Database Schema, Default Value is NOW()	I have database schema for users. It looks like... CREATE TABLE `users` (`id` int(8) unsigned AUTO_INCREMENT, `username` varcha...	[link=https://stackoverflow.com/questions/5322651/database-schema-default-value-is-now]🔗 Open[/link]	datetime, time, mysql-error-1067, mysql	Use CURRENT_TIMESTAMP, and change the column to timestamp. It does the same, but works (as long as you can live with timestamp) - it is a limitation...	49.92
2	Auto increment column	I want to make an AUTO_INCREMENT column in a database table,here is the syntax i write: create table comments (name varchar(2...	[link=https://stackoverflow.com/questions/3949064/auto-increment-column]🔗 Open[/link]	mysql-error-1075, sql, mysql	It is PRIMARY KEY without the underscore. create table comments (name varchar(20), mail varchar(30), comment varchar(100), com_no i...	45.19
3	I don't understand how to choose a database in MySQL.(Error 1056)	CREATE TABLE User(id INT PRIMARY KEY, names VARCHAR(100), aling VARCHAR(30), eye VARCHAR(30), hair VARCHAR(30),...	[link=https://stackoverflow.com/questions/74875613/i-dont-understand-how-to-choose-a-database-in-mysql-error-1056]🔗 Open[/link]	mysql-error-1046, mysql	No answer available	48.61
4	Snowflake Problem: Populate Table with Dates and its Attributes	I have a problem statement: Write a Stored Procedure that populates a table with certain date attributes. The data would be populate...	[link=https://stackoverflow.com/questions/76272327/snowflake-problem-populate-table-with-dates-and-its-attributes]🔗 Open[/link]	sql-server, javascript, sql, snowflake-cloud-data-platform, stored-procedures	CREATE OR REPLACE PROCEDURE uspPopulateDateTimeTable("startDate" TIMESTAMP) RETURNS STRING LANGUAGE JAVASCRIPT AS $$ { var currentDate = startDat...	45.49

Example 02

```
cs242@class-046:~/cs242$ python3 LLM.py
None of PyTorch, TensorFlow >= 2.0, or Flax have been found. Models won't be available and only tokenizers, configuration and file/data utilities can be used.
Note: Environment variable`HF_TOKEN` is set and is the current active token independently from the token you've just configured.

🔍 Enter your search query (Press Ctrl+D when done):
What do internal compiler error messages mean, and what can I do?
```

```
🤖 AI-Generated Answer:

An internal compiler error (ICE) is a bug in the compiler itself, not in your code. It means there's a problem with the compiler, not with your program. You can't fix an ICE in your code, but you can report it to the compiler vendor.
```

🔍 Related Stack Overflow Discussions:

Page 1 of 2

🔍 Search Results

Rank	Title	Problem Statement	URL	Tags	Answer / Snippet	Score
1	What do internal compiler error messages mean, and what can I do?	I was trying to compile my C++ program, which uses MPICH and NAG C library (I use NAG to generate random numbers), with a pgCC compi...	[link=https://stackoverflow.com/questions/12405606/what-do-internal-compiler-error-messages-mean-and-what-can-i-do]🔗 Open[/link]	compilation, internal-compiler-error, c++, compiler-errors	An internal compiler error is a bug in the compiler. There's not much you can do short of raising the problem with the compiler vendor. Usually, ICE...	14.59
2	how to handle server connection error and internet connection error and show different messages for each one in volley android?	what I want is to show different messages for connection errors, what I mean when there is a server connection error show to user th...	[link=https://stackoverflow.com/questions/56374093/how-to-handle-server-connection-error-and-internet-connection-error-and-show-dif]🔗 Open[/link]	android, error-handling, android-volley, general-network-error	'Here is the solution' @Override public void onErrorResponse(VolleyError error) { String json = null; ...	9.51
3	Bison C++ multiple error recovery with missing semi colon	I'm developing my own compiler, and I have a problem with error recovery design in panic mode for java grammar. I thought about mul...	[link=https://stackoverflow.com/questions/14689849/bison-c-multiple-error-recovery-with-missing-semi-colon]🔗 Open[/link]	c++, error-recovery, panic, bison, mode	You won't mind to solve that problem in the C++ OOP way, not in the bison way, would you? Consider you have these kinds of AST node defined struct ...	9.30
4	Working Around a Visual Studio Internal Compiler Error With Nested Templated Variables	I'm trying to write code that will let me index into the parameter types of a function: template <typename R, typename... ARGS> R f...	[link=https://stackoverflow.com/questions/56379486/working-around-a-visual-studio-internal-compiler-error-with-nested-templated-var]🔗 Open[/link]	visual-studio-2017, c++, internal-compiler-error, template-variables, function-parameter	It probably depends on the exact (sub)version of VS2017, as mine does not produce an ICE on the code. However, the code is still problematic, as it c...	9.27
5	jslint vim errorformat	I have the jslint installed with npm on my system. It produces error messages in the following format: 1 1,9: Missing name in fu...	[link=https://stackoverflow.com/questions/5870579/jslint-vim-errorformat]🔗 Open[/link]	errorformat, compiler-construction, vim, jslint	The problem was with the \,, the skip comma, the correct format is: CompilerSet errorformat= \%*[\ ]%n\ %l%.%c:\ %m, \%-G%.%# | The c...	9.49

Instructions for Usage:

To run the LLM system, use the command -

Python3 LLM.py

Limitations

- We did not handle user queries efficiently (i.e. some user queries might be too short or contain garbage input or abstract error terms)

Pending Work & Future Scope

- UI development
- Handle user queries efficiently
- Performance evaluation

Contribution:

1. Crawler Development (Data Collection):

a) Research and selection of error-related tags for scraping: Zabir Al Nazi, G M Shahariar
b) Developing the Scrapy crawler to extract StackOverflow data: Md Taukir Azam Chowdhury, Md. Olid Bhuiyan, Samiha Khan
c) Adding pagination support to scrape multiple pages per tag: Zabir Al Nazi
d) Handling duplicate data and validating collected entries: G M Shahariar
e) Data collection on multiple machines using different seeds: Samiha Khan, Zabir Al Nazi, G M Shahariar, Md Taukir Azam Chowdhury, Md. Olid Bhuiyan

2. PyLucene Indexing

a) *lucene_indexer.py*: Md. Olid Bhuiyan, Zabir Al Nazi, Samiha Khan
b) *lucene_retriever.py:* G M Shahariar, Md Taukir Azam Chowdhury
c) Runtime Analysis: Zabir Al Nazi

3. LLM-Based Answers (Bonus): G M Shahariar, Md Olid Bhuiyan

4. Testing and Debugging: Samiha Khan, Zabir Al Nazi, G M Shahariar, Md Taukir Azam Chowdhury, Md. Olid Bhuiyan

4. Reporting & Documentation: Md Taukir Azam Chowdhury, Samiha Khan, Zabir Al Nazi