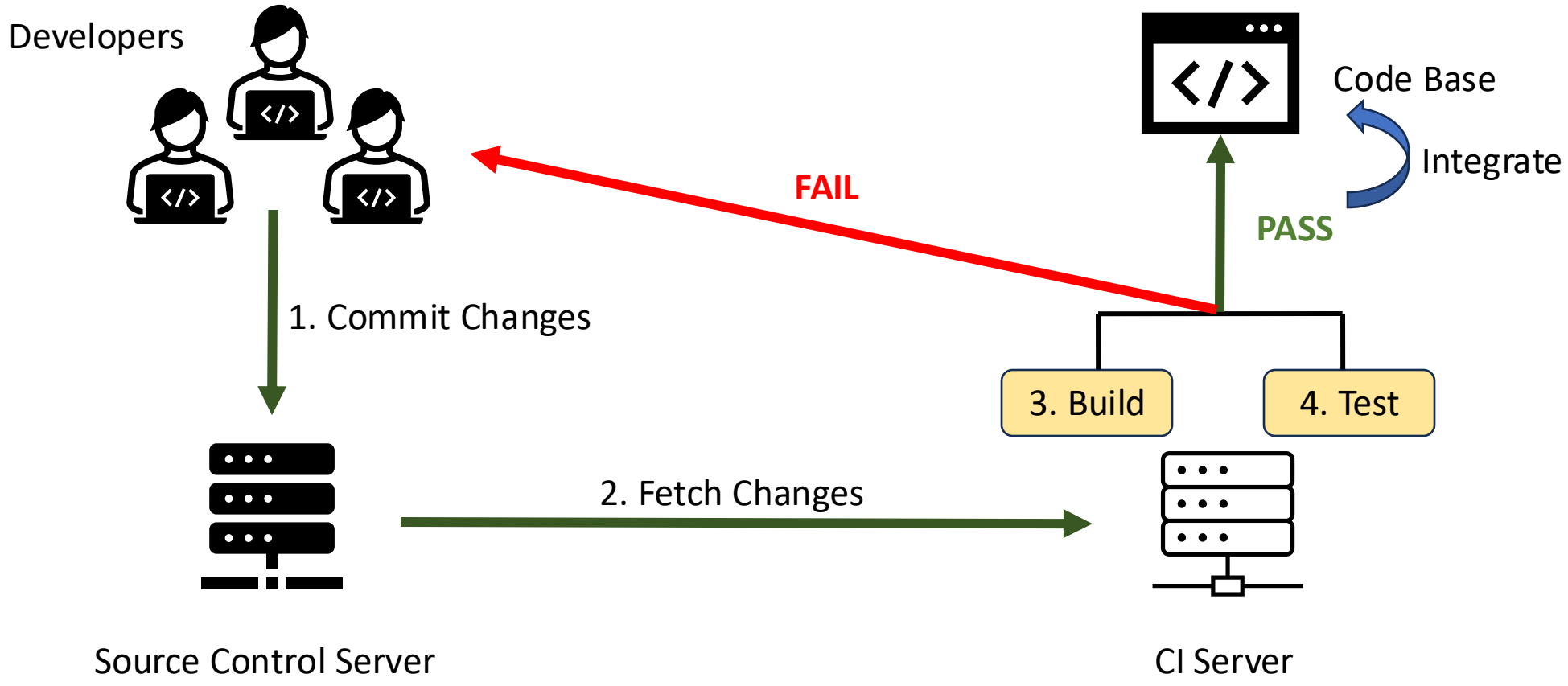
CS202: Software Tools and Techniques for CSE

Lecture 2

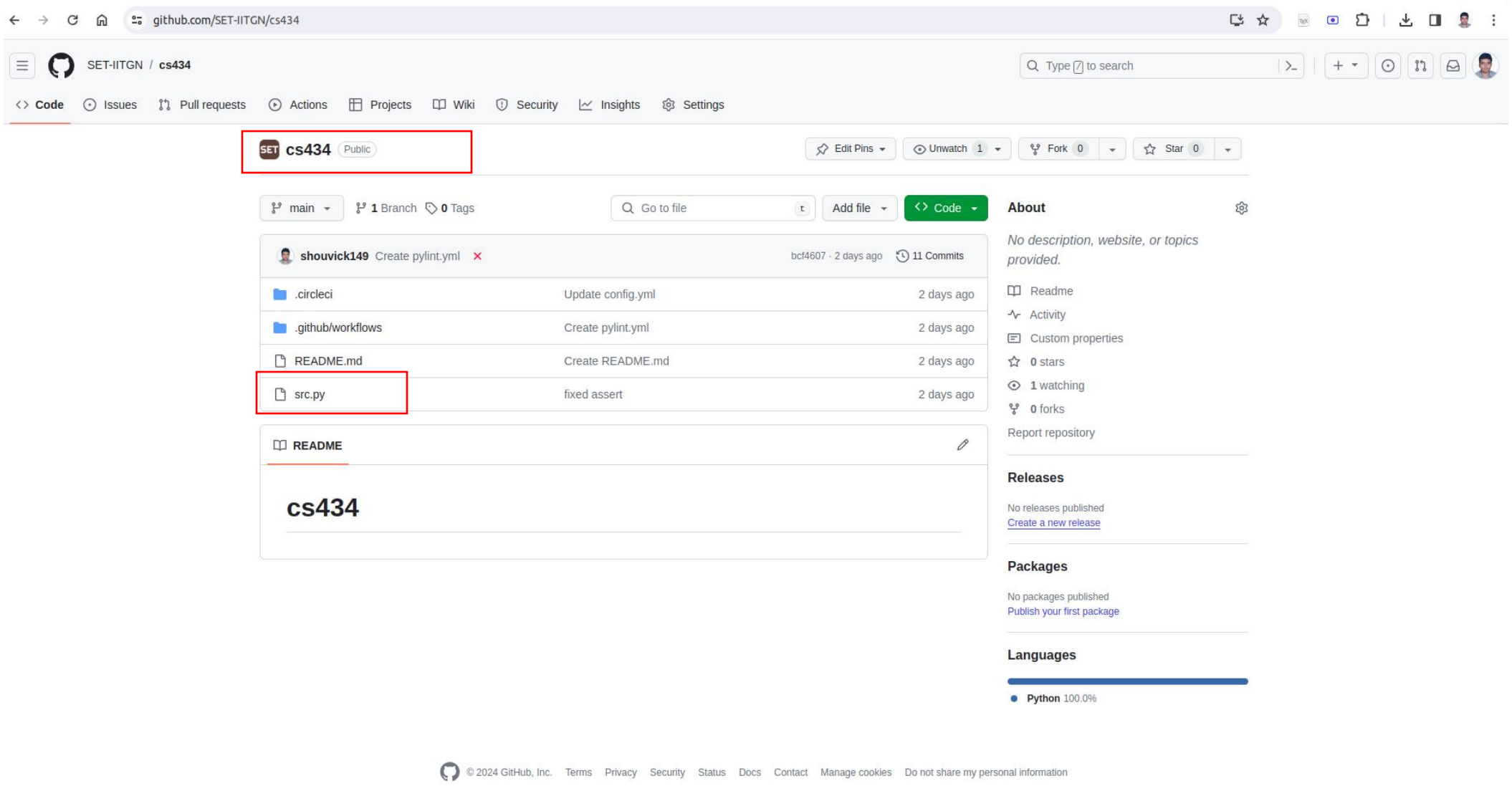
Shouvick Mondal

shouvick.mondal@iitgn.ac.in
August 2025

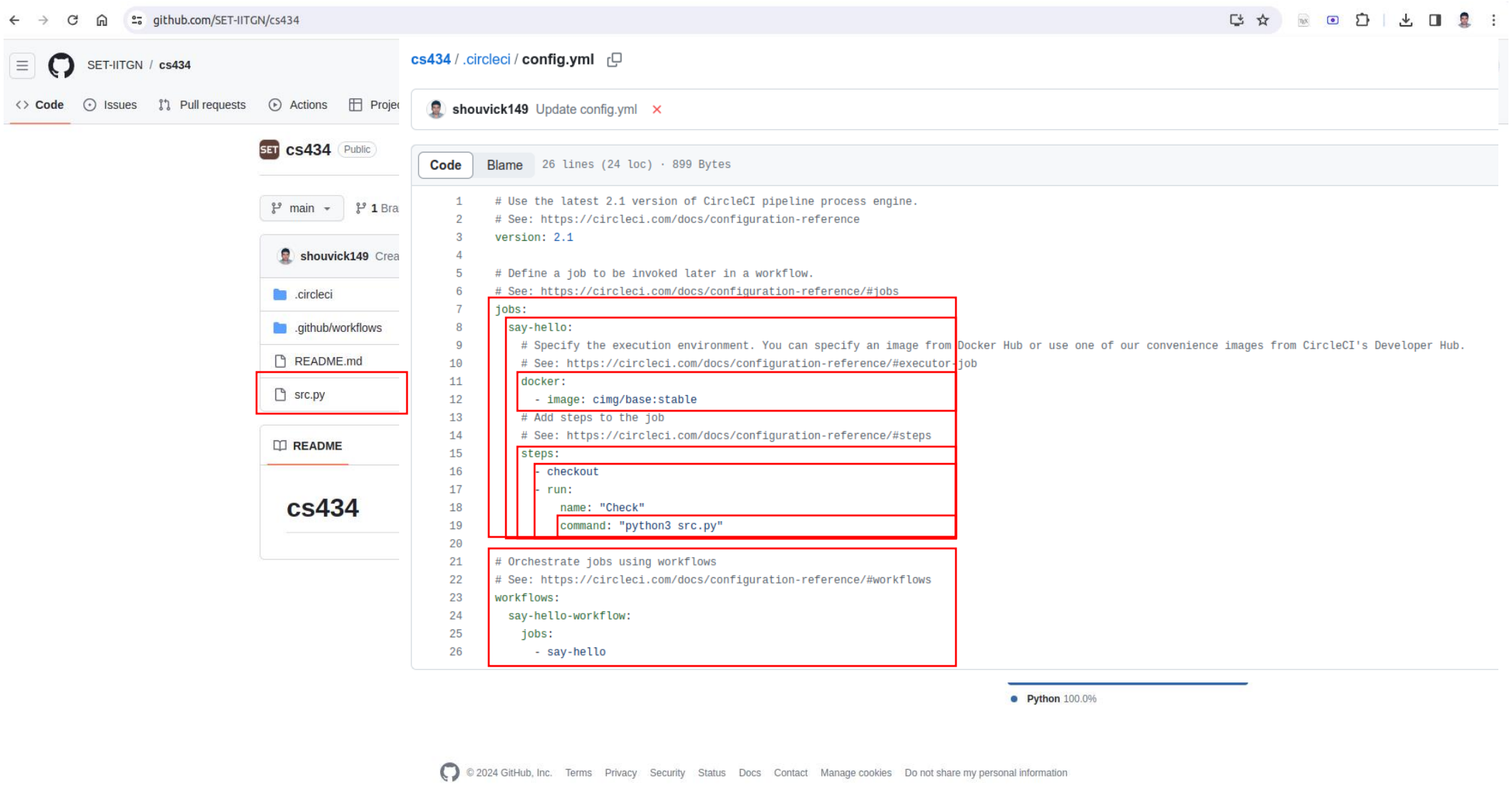
Continuous Integration (CI)



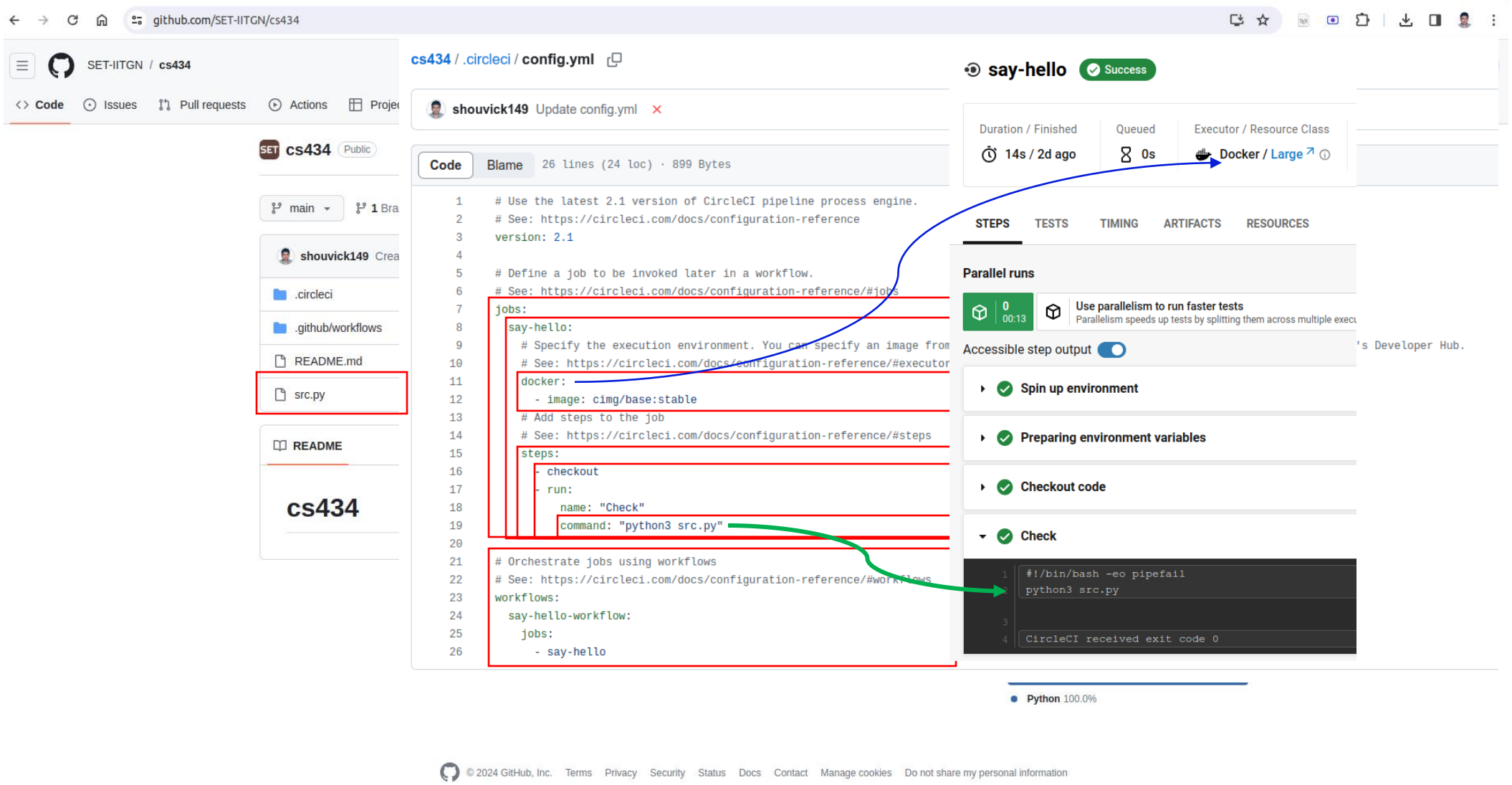
Setting up CI for a GitHub project (cs434)



Specify CI workflows/jobs in YAML (`config.yml`)



Specify CI workflows/jobs in YAML (`config.yml`)



Source code *diff* (unified)

github.com/SET-IITGN/cs202/commit/45a77e06f0b1b13907fcca80e7debd614fe0ba3f?diff=unified#diff-5cc5f480c15364f8b27b8941ae1e9498d3170d3a6e13421e7d527e7e1b5e5410

Filter files…

- src.c
- vuln.c

2 files changed +1 -2 lines changed

Search within code

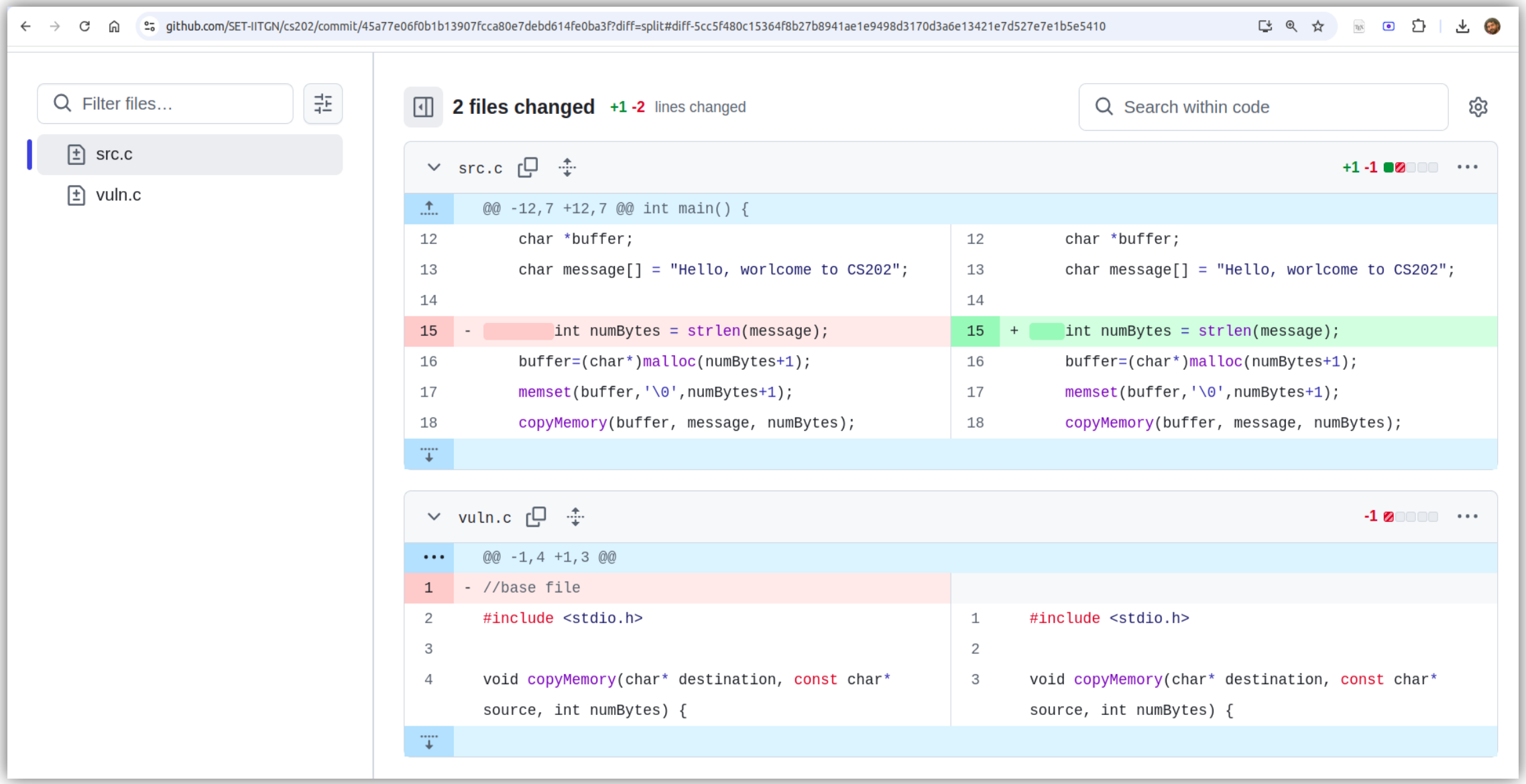
src.c +1 -1

```
@@ -12,7 +12,7 @@ int main() {
12 12      char *buffer;
13 13      char message[] = "Hello, worlcome to CS202";
14 14
15    -    int numBytes = strlen(message);
   15 +    int numBytes = strlen(message);
16 16      buffer=(char*)malloc(numBytes+1);
17 17      memset(buffer,'\0',numBytes+1);
18 18      copyMemory(buffer, message, numBytes);
```

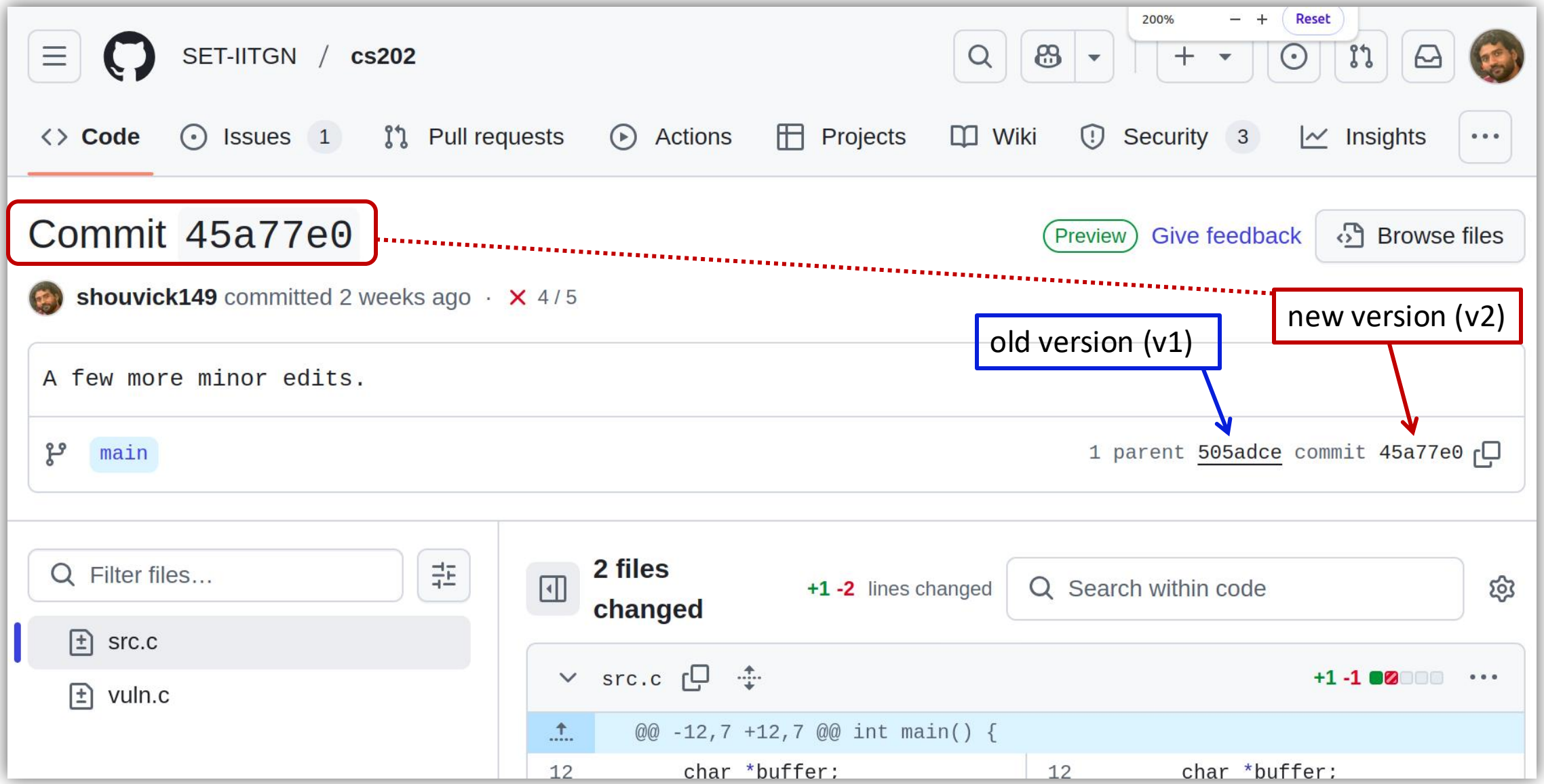
vuln.c -1

```
@@ -1,4 +1,3 @@
1    - //base file
2  1   #include <stdio.h>
3  2
4  3   void copyMemory(char* destination, const char* source, int numBytes) {
```

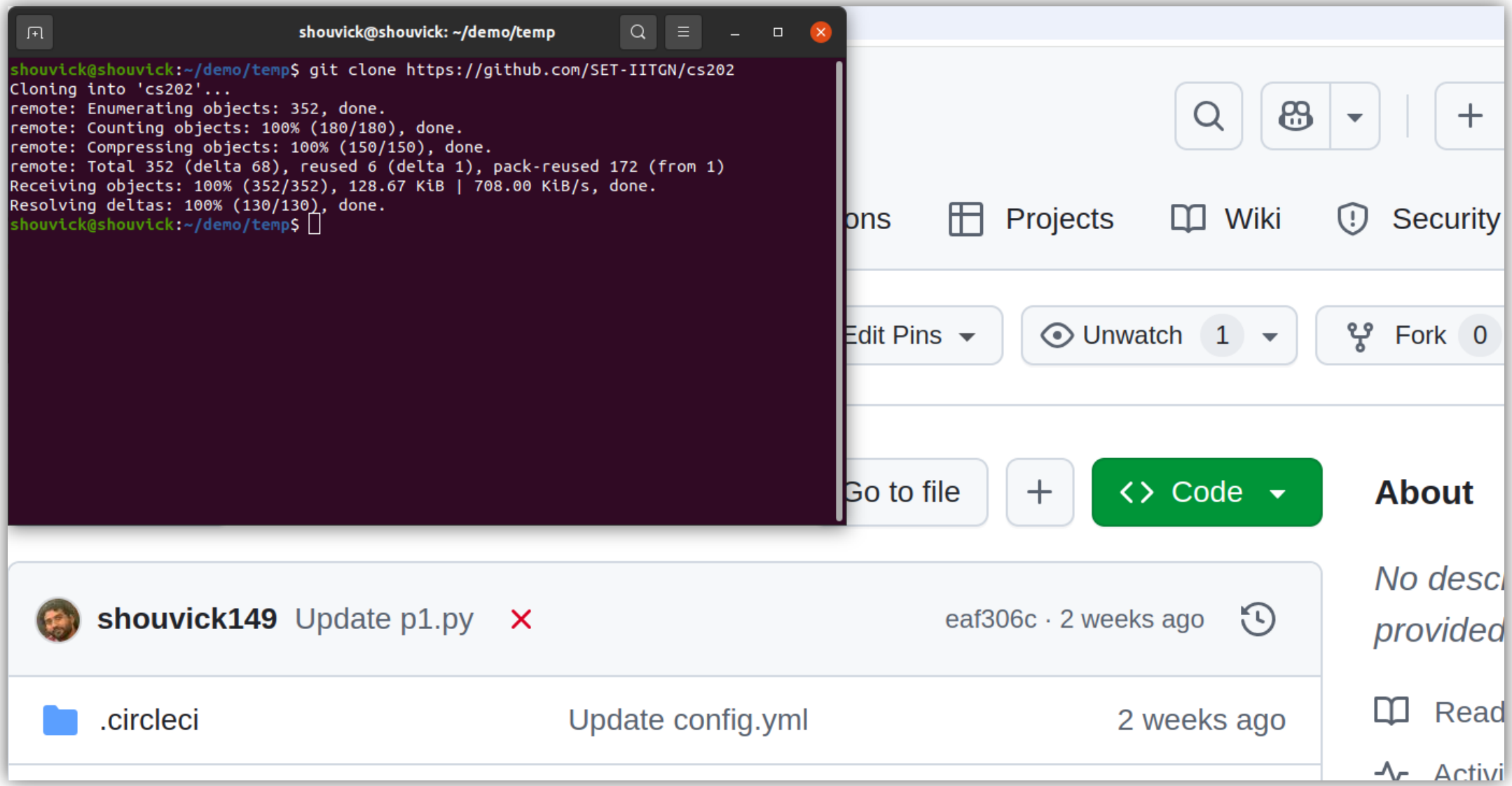
Source code *diff* (split)



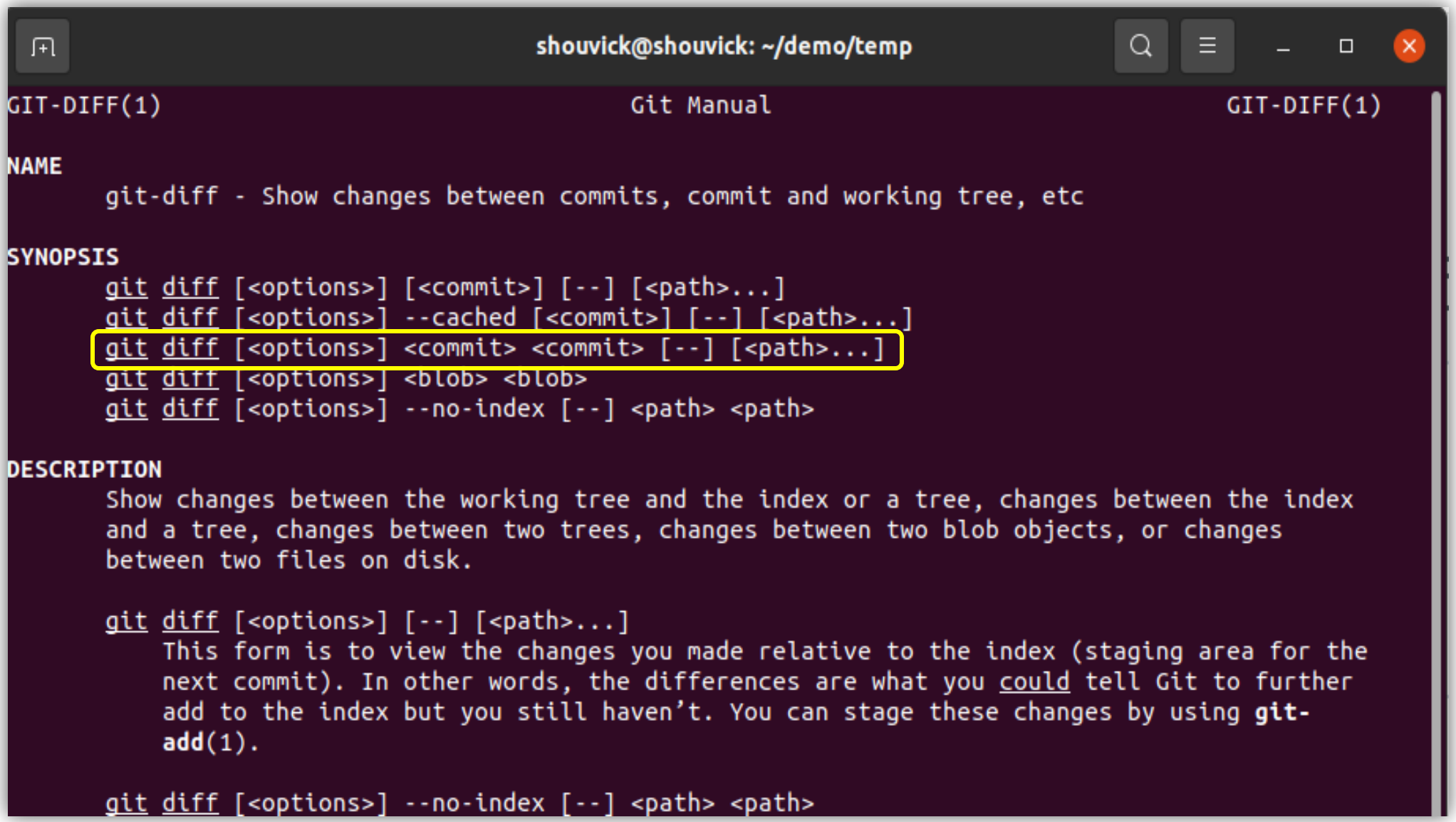
Browsing a commit on GitHub



git clone https://github.com/SET-IITGN/cs202



cd cs202; git diff --help



Source code *diff* (unified)

github.com/SET-IITGN/cs202/commit/45a77e06f0b1b13907fcca80e7debd614fe0ba3f?diff=unified#diff-5cc5f480c15364f8b27b8941ae1e9498d3170d3a6e13421e7d527e7e1b5e5410

Filter files…

- src.c
- vuln.c

2 files changed +1 -2 lines changed

Search within code

src.c +1 -1

```
@@ -12,7 +12,7 @@ int main() {
12 12      char *buffer;
13 13      char message[] = "Hello, worlcome to CS202";
14 14
15    -    int numBytes = strlen(message);
   15 +    int numBytes = strlen(message);
16 16      buffer=(char*)malloc(numBytes+1);
17 17      memset(buffer,'\0',numBytes+1);
18 18      copyMemory(buffer, message, numBytes);
```

vuln.c -1

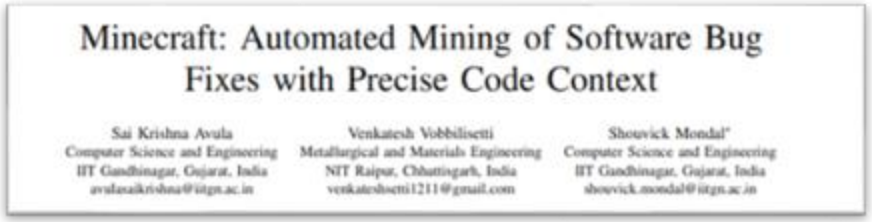
```
@@ -1,4 +1,3 @@
1    - //base file
2  1   #include <stdio.h>
3  2
4  3   void copyMemory(char* destination, const char* source, int numBytes) {
```

git diff <parent_commit> <curr_commit>

```
diff --git a/src.c b/src.c
index 0d6a029..85bb4fb 100644
--- a/src.c
+++ b/src.c
@@ -12,7 +12,7 @@ int main() {
     char *buffer;
     char message[] = "Hello, worlcome to CS202";

-       int numBytes = strlen(message);
+    int numBytes = strlen(message);
     buffer=(char*)malloc(numBytes+1);
     memset(buffer,'\0',numBytes+1);
     copyMemory(buffer, message, numBytes);
diff --git a/vuln.c b/vuln.c
index 838cba5..c1a0e33 100644
--- a/vuln.c
+++ b/vuln.c
@@ -1,4 +1,3 @@
-//base file
 #include <stdio.h>
```

My Group's Contribution to the OSS community



Minecraft: Automated Mining of Software Bug Fixes with Precise Code Context

Sai Krishna Avula
Computer Science and Engineering
IIT Gandhinagar, Gujarat, India
avulasaikrishna@iitgn.ac.in

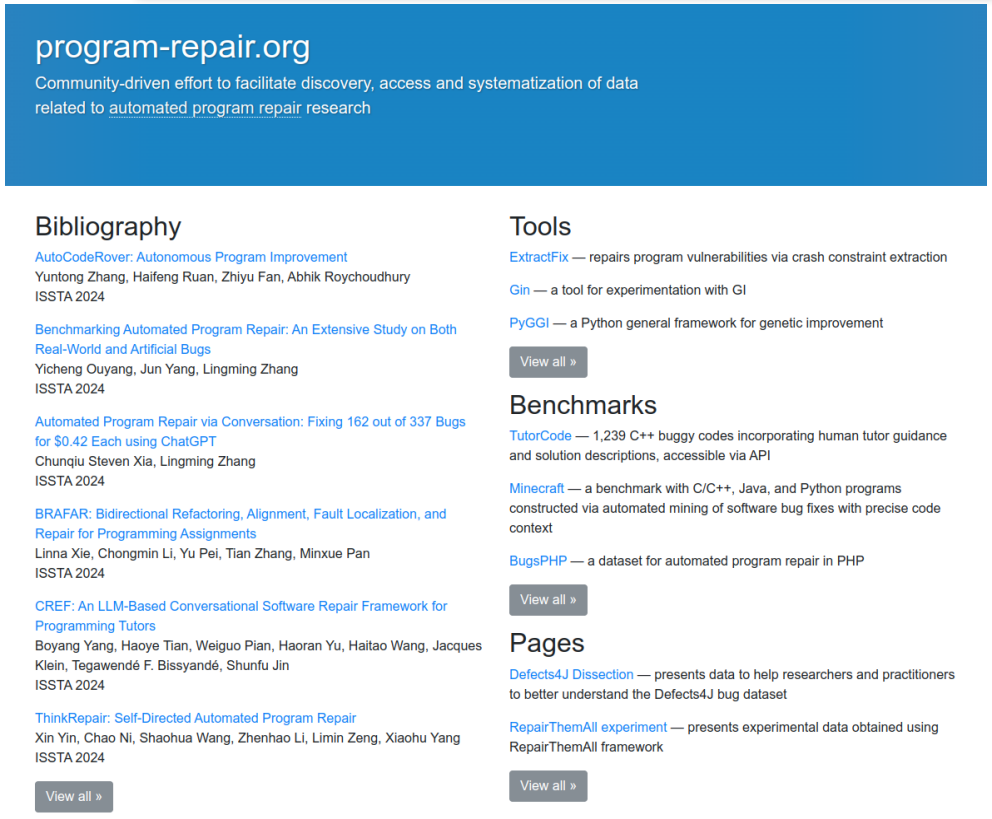
Venkatesh Vobbilisetti
Metallurgical and Materials Engineering
NIT Raipur, Chhattisgarh, India
venkateshsetti1211@gmail.com

Shouvick Mondal*
Computer Science and Engineering
IIT Gandhinagar, Gujarat, India
shouvick.mondal@iitgn.ac.in

MineCPP: Mining Bug Fix Pairs and Their Structures

Sai Krishna Avula
IIT Gandhinagar
Gandhinagar, India

Shouvick Mondal
IIT Gandhinagar
Gandhinagar, India



program-repair.org

Community-driven effort to facilitate discovery, access and systematization of data related to automated program repair research

Bibliography

AutoCodeRover: Autonomous Program Improvement
Yuntong Zhang, Haifeng Ruan, Zhiyu Fan, Abhik Roychoudhury
ISSTA 2024

Benchmarking Automated Program Repair: An Extensive Study on Both Real-World and Artificial Bugs
Yicheng Ouyang, Jun Yang, Lingming Zhang
ISSTA 2024

Automated Program Repair via Conversation: Fixing 162 out of 337 Bugs for $0.42 Each using ChatGPT
Chunqiu Steven Xia, Lingming Zhang
ISSTA 2024

BRAFAR: Bidirectional Refactoring, Alignment, Fault Localization, and Repair for Programming Assignments
Linna Xie, Chongmin Li, Yu Pei, Tian Zhang, Minxue Pan
ISSTA 2024

CREF: An LLM-Based Conversational Software Repair Framework for Programming Tutors
Boyang Yang, Haoye Tian, Weiguo Pian, Haoran Yu, Haitao Wang, Jacques Klein, Tegawendé F. Bissyandé, Shunfu Jin
ISSTA 2024

ThinkRepair: Self-Directed Automated Program Repair
Xin Yin, Chao Ni, Shaohua Wang, Zhenhao Li, Limin Zeng, Xiaohu Yang
ISSTA 2024

View all »

Tools

ExtractFix — repairs program vulnerabilities via crash constraint extraction

Gin — a tool for experimentation with GI

PyGGI — a Python general framework for genetic improvement

View all »

Benchmarks

TutorCode — 1,239 C++ buggy codes incorporating human tutor guidance and solution descriptions, accessible via API

Minecraft — a benchmark with C/C++, Java, and Python programs constructed via automated mining of software bug fixes with precise code context

BugsPHP — a dataset for automated program repair in PHP

View all »

Pages

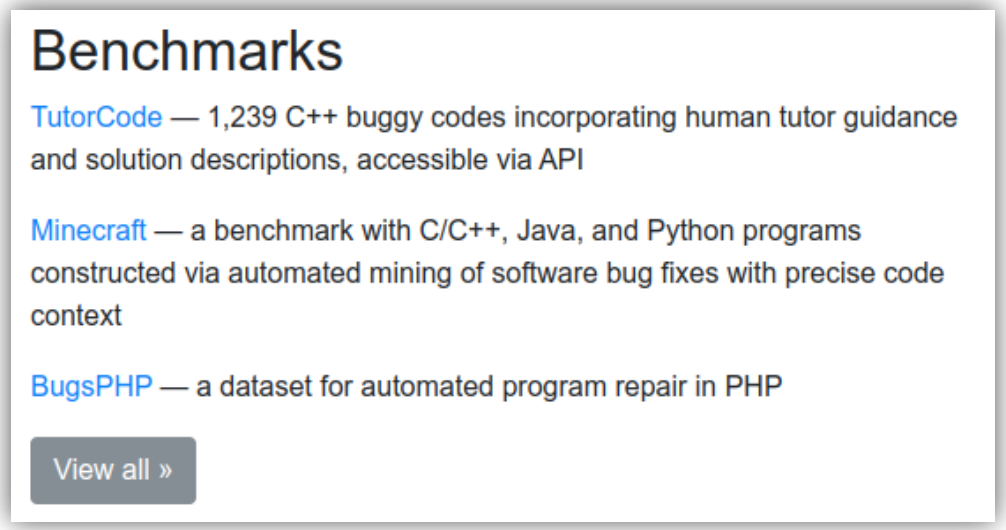
Defects4J Dissection — presents data to help researchers and practitioners to better understand the Defects4J bug dataset

RepairThemAll experiment — presents experimental data obtained using RepairThemAll framework

View all »

The data on this website is released under the Open Data Commons Attribution License (ODC-BY 1.0). The bibliographic records are extracted from dblp.

Our dataset has been included as a multilingual program repair benchmark in the area of Automated Program Repair (APR).



Benchmarks

TutorCode — 1,239 C++ buggy codes incorporating human tutor guidance and solution descriptions, accessible via API

Minecraft — a benchmark with C/C++, Java, and Python programs constructed via automated mining of software bug fixes with precise code context

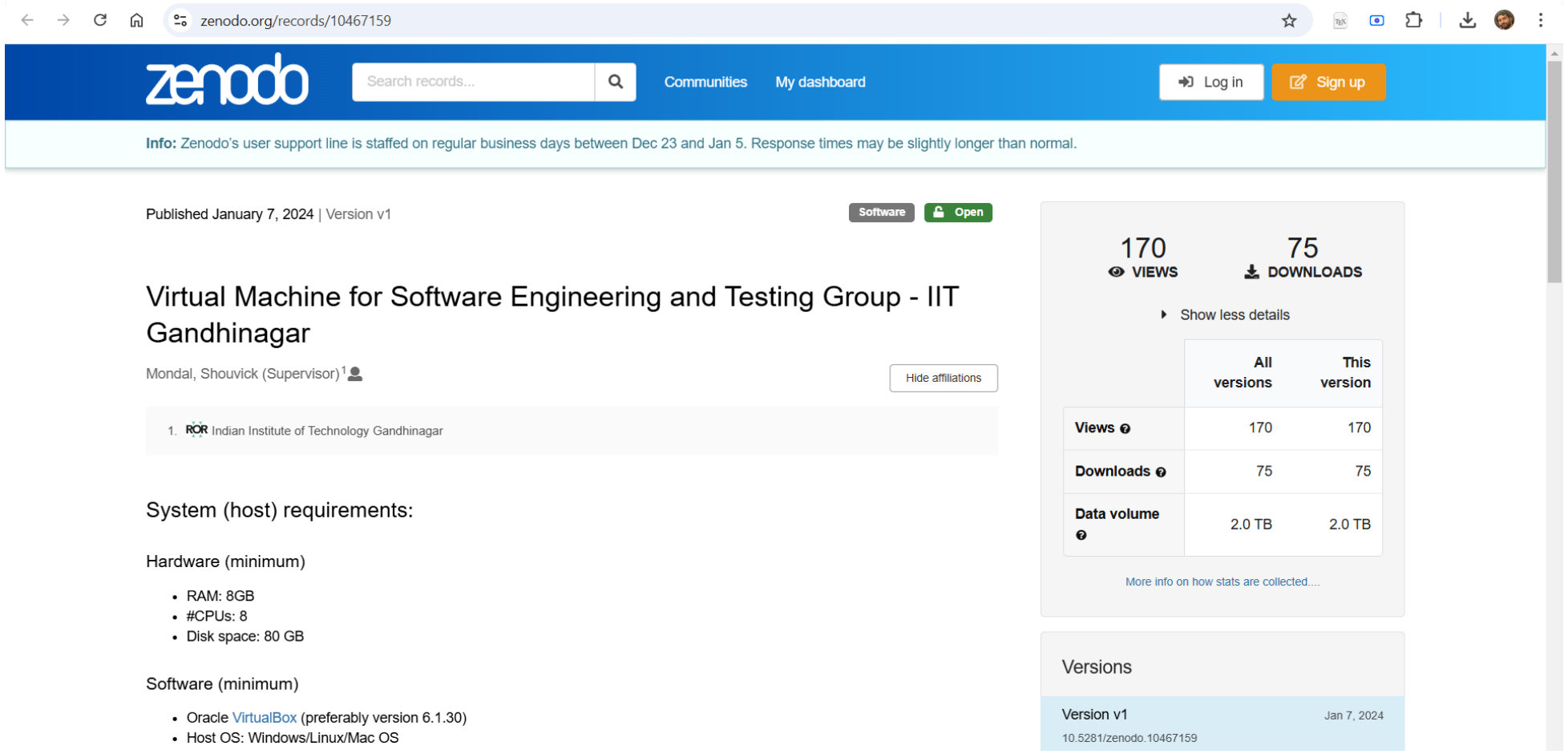
BugsPHP — a dataset for automated program repair in PHP

View all »

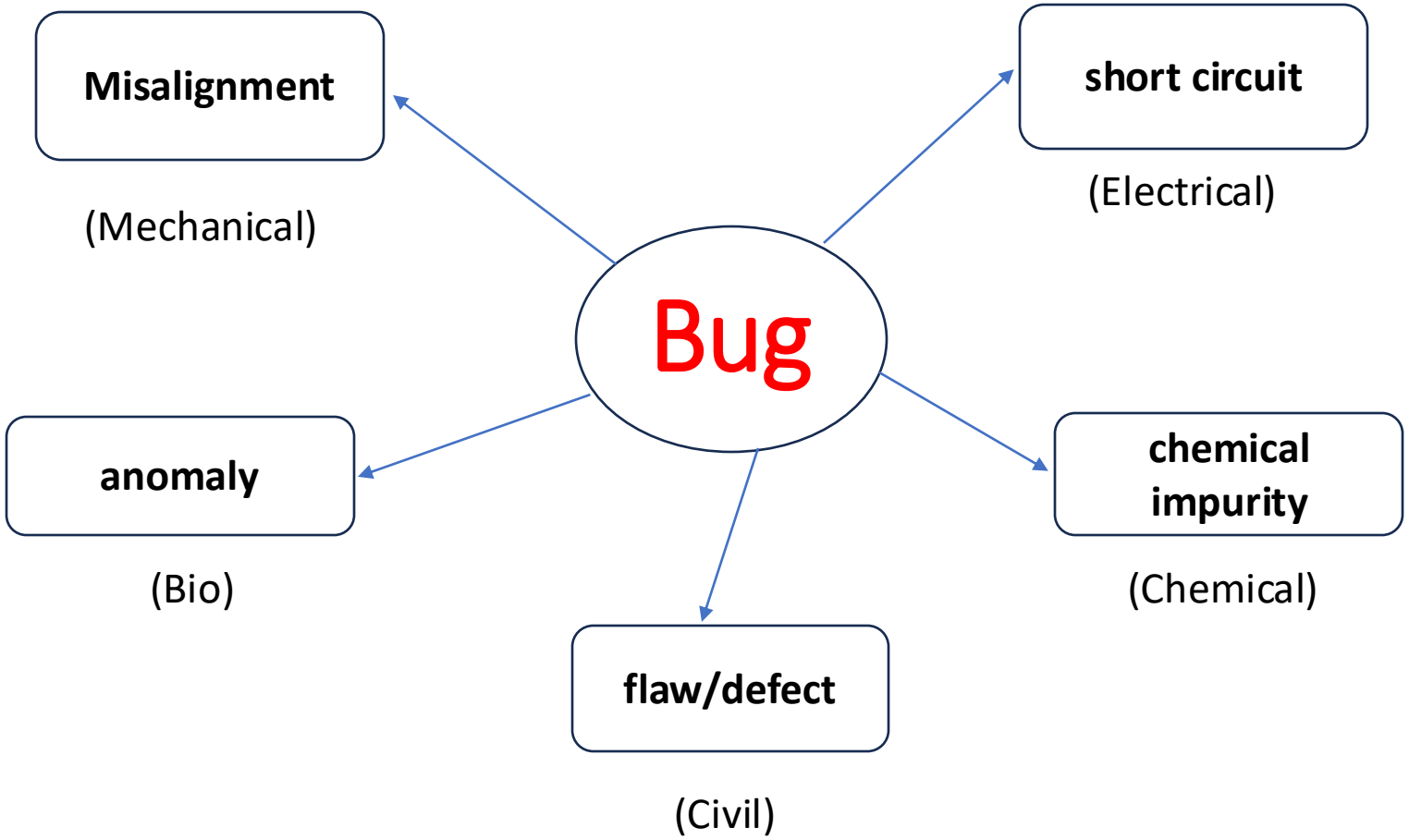
My Group's Contribution to the OSS community

Publicly Available Virtual Machine (sandbox) from SET Group – IITGN.

Provides a playground without worrying about the consequence if anything breaks!



Analogy of Bug



Bug Mining

- Bug mining (bug extraction) – identifying, **extracting code snippets** with **bugs** along with their **corresponding fixes** and **contextual information** from software repositories.
- **Identification** and **resolution** of bugs - critical for **software reliability**, **performance**, and **security**.
- Essential for improving software quality, enhancing developer productivity, and advancing our understanding of software defects

The challenges of extracting bug and its contextual information

- **Granularity** of repository-mined bug-fixing datasets/bug-mining techniques
  - **Function level**, **without** **precise bug location** and **bug types**.
- **No** language agnostic algorithm to extract **code snippets**
- *Major issue*: bug-fix **context** has never been explored...

- **No** **unified** tools - bug **contextual information**
- **Unavailability** of GUI to **visualise** and **analyse** the bugs

Minecraft: Automated Mining of Software Bug Fixes with Precise Code Context

Published in ASE 2023 (CORE A*) Industry challenge track

https://www.doi.org/10.1109/ASE56229.2023.00116

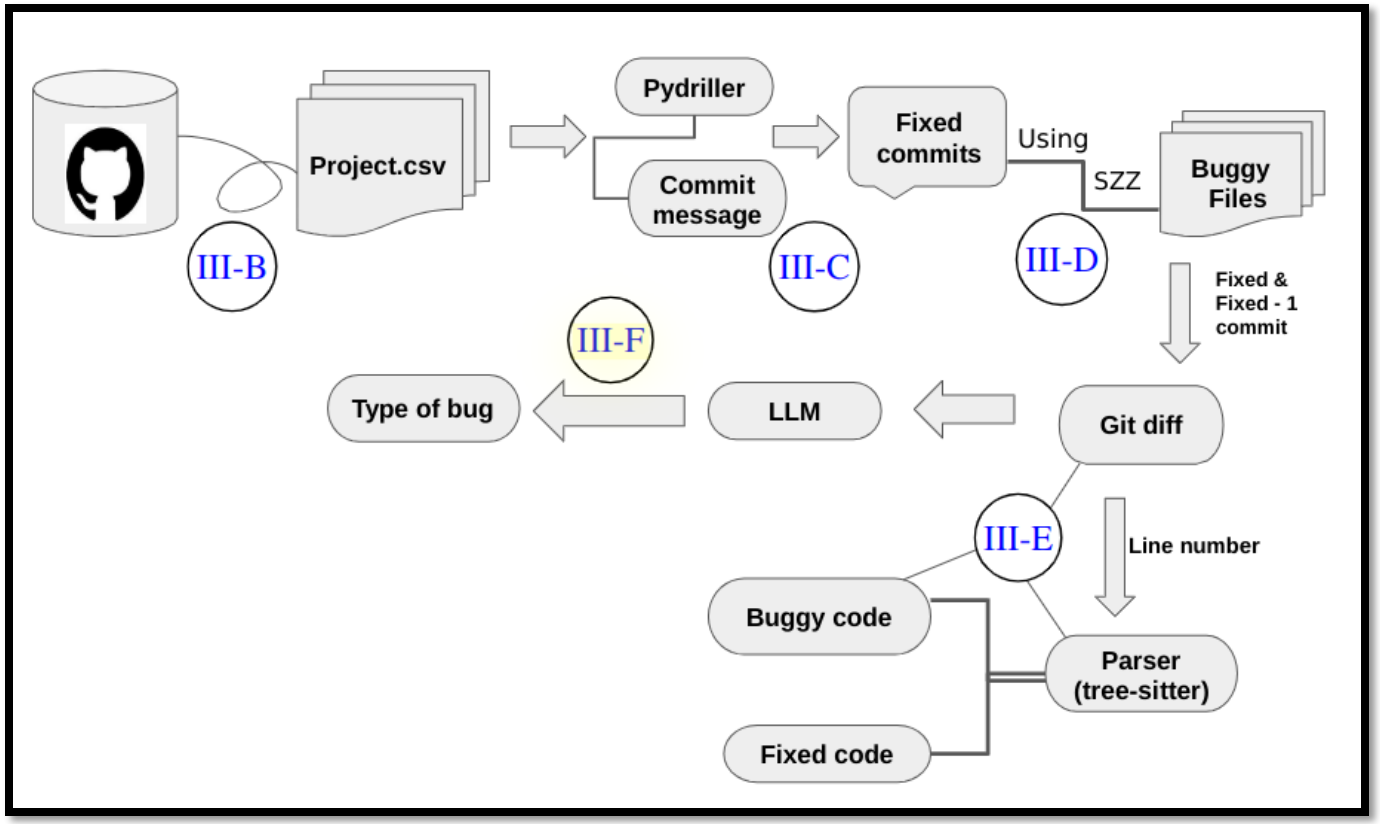
Terminology

- **GitHub** is a **web-based platform** commonly used for **version control** and **collaboration** in software development projects.
- A **commit** refers to a specific **revision** or **snapshot of a project's files** and directories. When a developer makes changes to the codebase, they create a commit to record those changes. A commit typically includes a **concise message** that **describes the purpose or nature of the changes made - Commit Message**

Bug Fix Dataset Acquisition Flowchart with Precise Context

Data acquisition is done in following ways:

- **Project Selection**
- Bug fix commits
- Extracting Bug and fixed code Snippets
- Type of the bug



Projects Selection Criteria (Visual Representation)

In our project selection process, we implemented a rigorous and stringent set of filtering criteria. These criteria were designed to sift through a wide array of options and zero in on real-world projects. This approach ensured that the ***projects we selected were not only substantial but also highly relevant to practical applications*** ***in the fields of C, C++, and Python.***

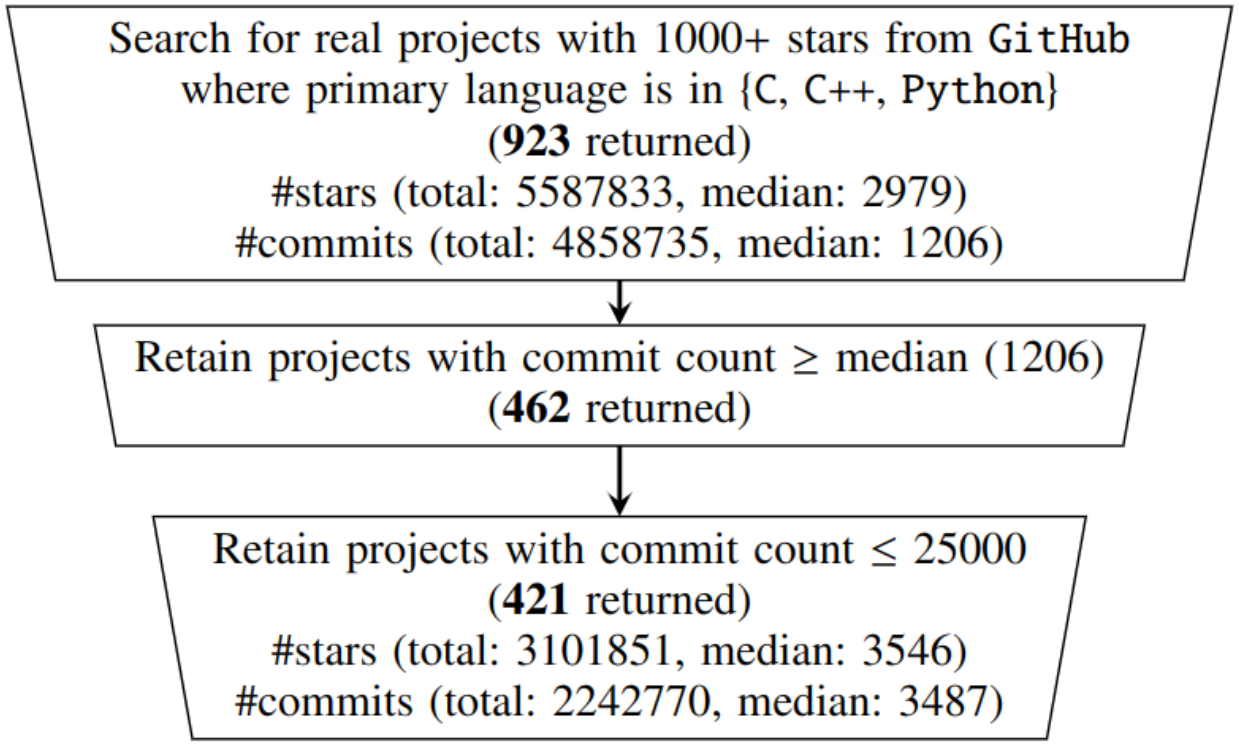
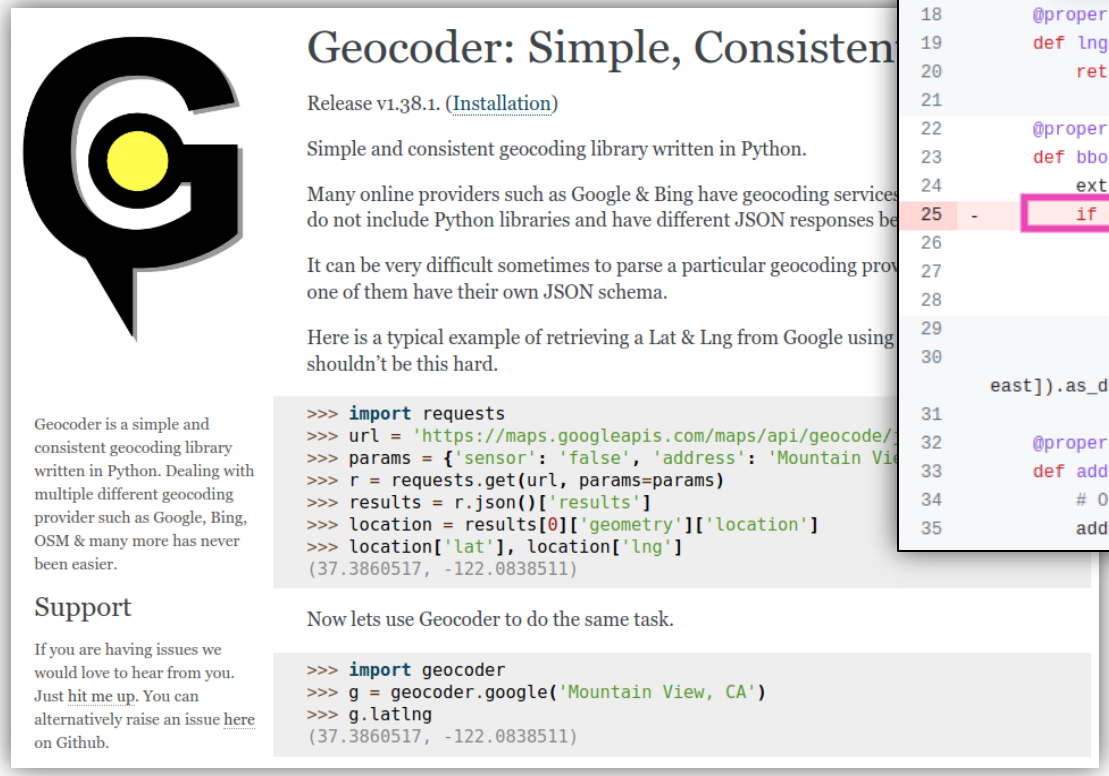


Fig. 2. Our project selection process.

Example workflow...

To demonstrate with examples, we chose a project named **Geocoder** which is written in **Python**



Geocoder: Simple, Consisten

Release v1.38.1. (Installation)

Simple and consistent geocoding library written in Python.

Many online providers such as Google & Bing have geocoding services do not include Python libraries and have different JSON responses be

It can be very difficult sometimes to parse a particular geocoding prov one of them have their own JSON schema.

Here is a typical example of retrieving a Lat & Lng from Google using shouldn't be this hard.

```
>>> import requests
>>> url = 'https://maps.googleapis.com/maps/api/geocode/
>>> params = {'sensor': 'false', 'address': 'Mountain Vi
>>> r = requests.get(url, params=params)
>>> results = r.json()['results']
>>> location = results[0]['geometry']['location']
>>> location['lat'], location['lng']
(37.3860517, -122.0838511)
```

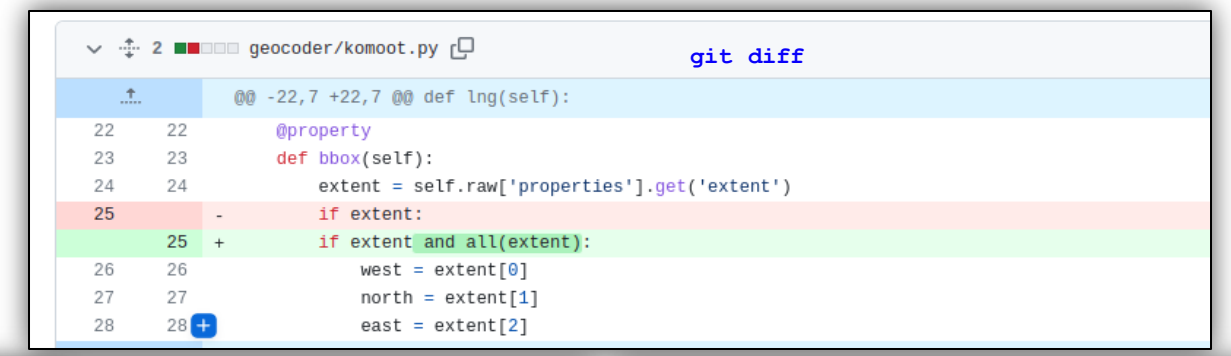
Now lets use Geocoder to do the same task.

```
>>> import geocoder
>>> g = geocoder.google('Mountain View, CA')
>>> g.latlng
(37.3860517, -122.0838511)
```

Geocoder is a simple and consistent geocoding library written in Python. Dealing with multiple different geocoding provider such as Google, Bing, OSM & many more has never been easier.

Support

If you are having issues we would love to hear from you. Just hit me up. You can alternatively raise an issue here on Github.

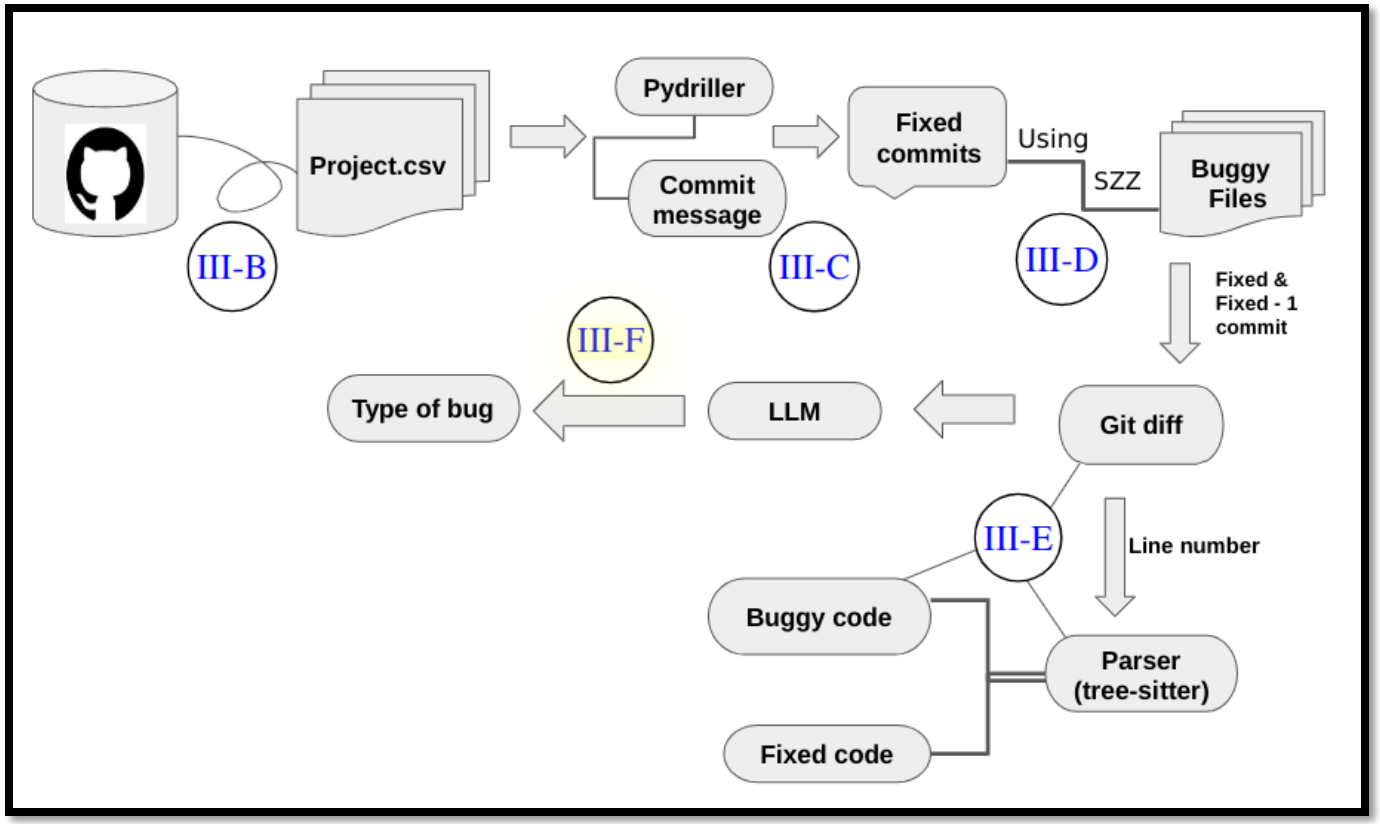


Loc: [before: 25, after: 25]
Bug Type: fix komoot.py when extent is none
Commit Message: fix calls to BBox with invalid input

Bug Fix Dataset Acquisition Flowchart with Precise Context

Data acquisition is done in following ways:

- Project Selection
- **Bug fix commits**
- Extracting Bug and fixed code Snippets
- Type of the bug



Collection of Bug-fix Commits

The collection of bug-fixing commits involved multiple strategies:

- Commit messages were analyzed to identify bug fix commits, leveraging keywords related to bug fixes. A total of **52 keywords** are used.
- Also, **Regular expressions** were utilized to perform precise searches for bug fix commit patterns
- The **SZZ algorithm** was applied to trace back and identify the buggy files associated with bug fix commits

"fixed ", " bug", "fixes ", "fix ", " fix", " fixed", " fixes", "crash", "solves", " resolves", "resolves ", " issue", "issue ", "regression", "fall back", "assertion", "coverity", "reproducible", "stack-wanted", "steps-wanted", "testcase", "failur", "fail", "npe ", " npe", "except", "broken", "differential testing", "error", "hang ", " hang", "test fix", "steps to reproduce", "crash", "assertion", "failure", "leak", "stack trace", "heap overflow", "freez", "problem ", " problem", " overflow", "overflow ", "avoid ", " avoid", "workaround ", " workaround", "break ", " break", " stop", "stop "

Commit messages with
52 keywords [6] (*c.f.* Sec. III-C)

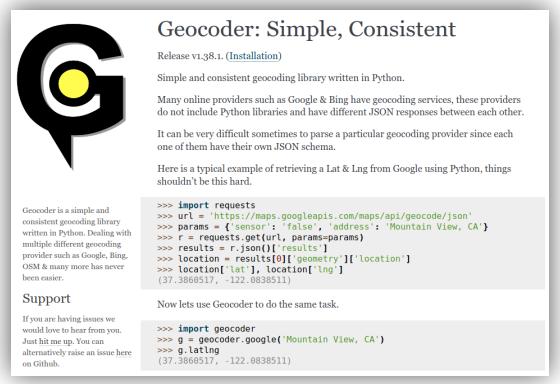
+ RegEx. [7]: `'.((solv(ed|es|e|ing))|(fix(s|es|ing|ed)?)'`
`|((error|bug|issue)(s)?)).'`

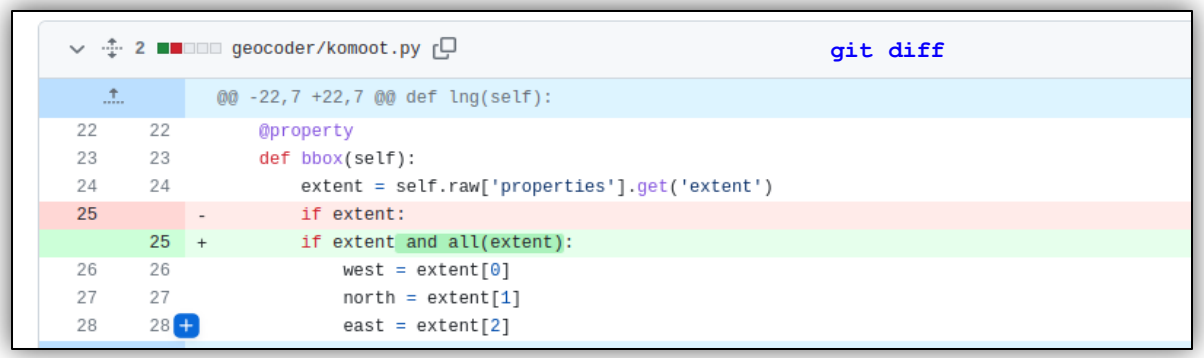
→SZZ Algorithm [8]

[J. Śliwerski, T. Zimmermann, and A. Zeller; (2005)] SZZ takes a commit (***bug-fixing***) as the input and returns a list of commits (***bug-introducing***) that last added the deleted lines in the input commit.

Example workflow...

To demonstrate with examples, we chose a project named **Geocoder** which is written in **Python**



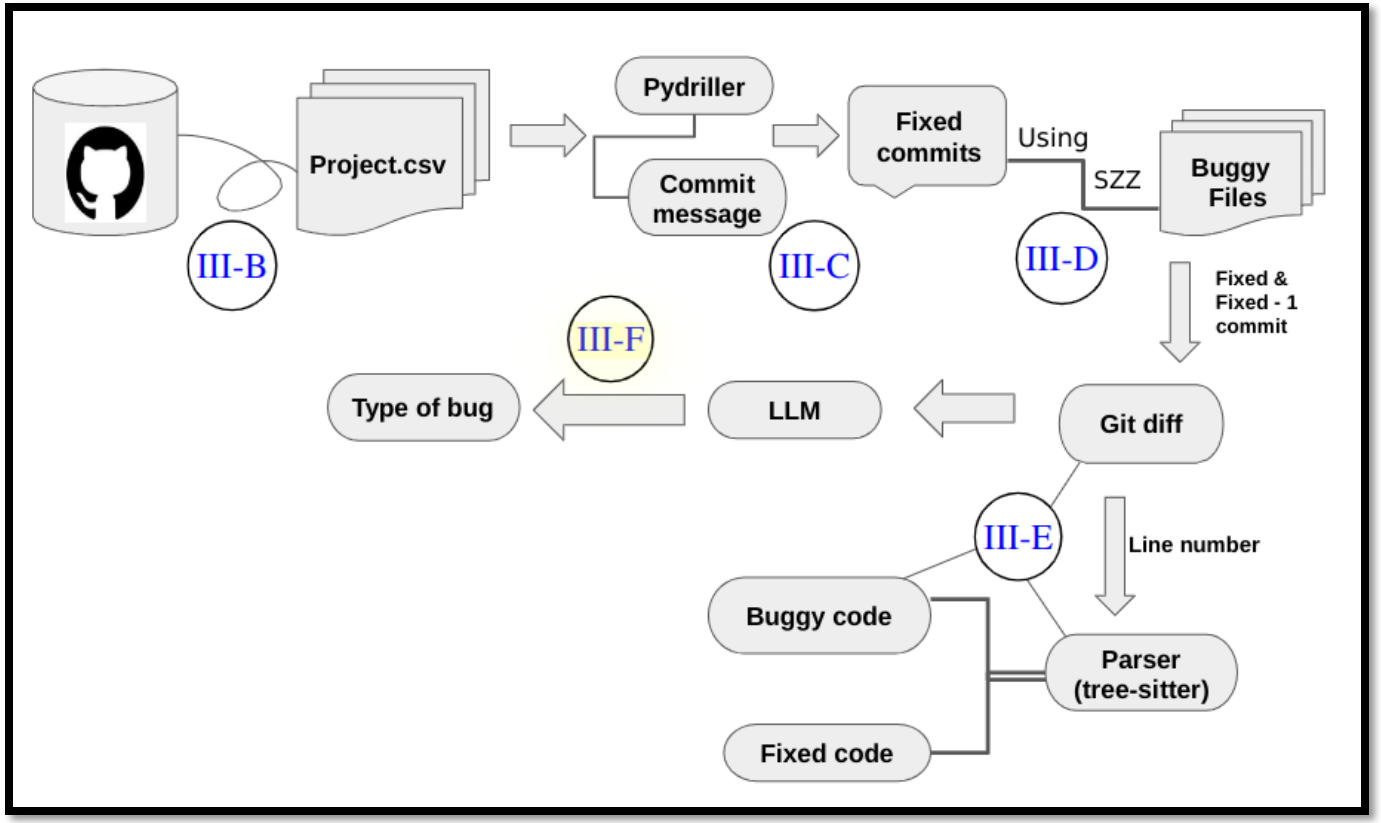


Loc: [before: 25, after: 25]
Bug Type: fix komoot.py when extent is none
Commit Message: fix calls to BBox with invalid input

Bug Fix Dataset Acquisition Flowchart with Precise Context

Data acquisition is done in following ways:

- Project Selection
- Bug fix commits
- **Extracting Bug and fixed code Snippets**
- Type of the bug

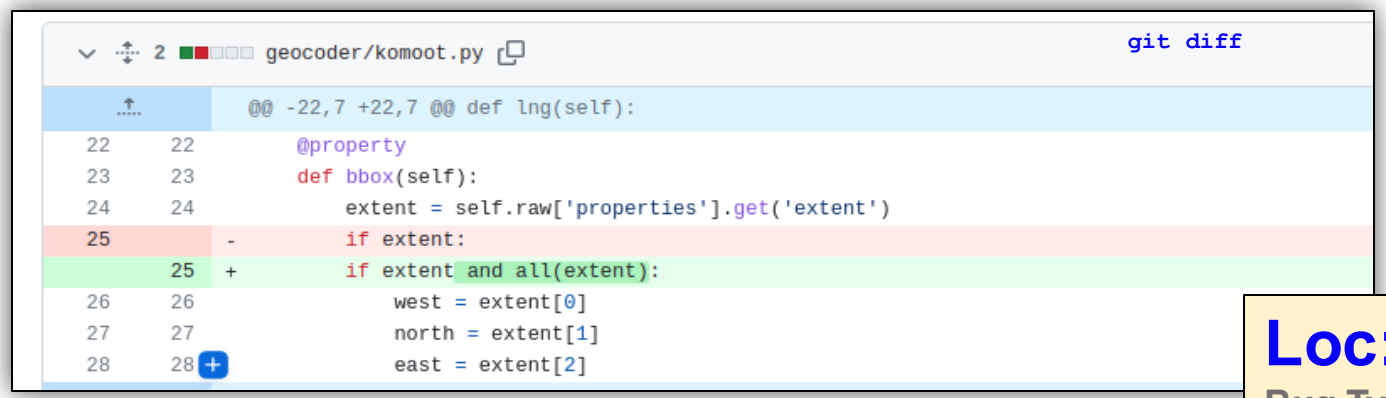


Extracting Code Snippets

The extraction of code snippets is done as follows:

Step 1:

- **Location:** The location of bug is found using git diff between fixed - 1 and fixed commit. The location here indicates the line number in which the bug is detected and fixed.



a **modification** typically involves an **addition** after a **deletion**

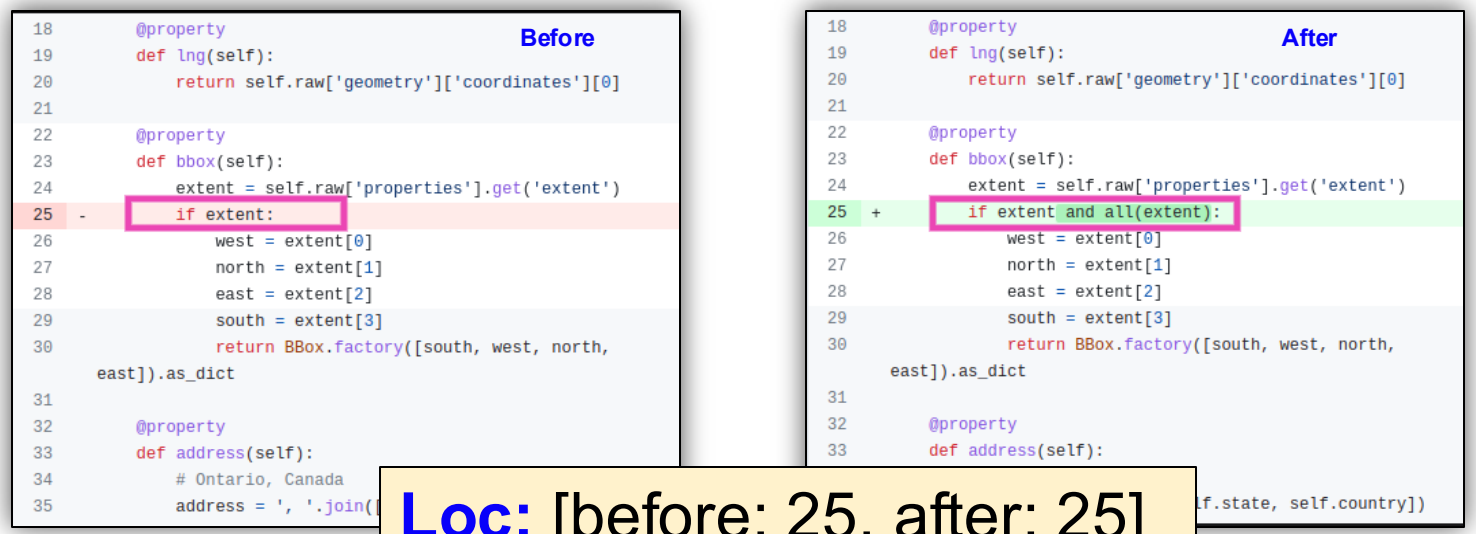
Loc: [before: 25, after: 25]
Bug Type: fix komoot.py when extent is none
Commit Message: fix calls to BBox with invalid input

[G. Canfora, L. Cerulo, and M. Di Penta, "Identifying changed source code lines from version repositories," in Fourth International Workshop on Mining Software Repositories (MSR'07:ICSE Workshops 2007), 2007, pp. 14–14.]

Extracting Code Snippets

Step 2:

- **Buggy Code:** The location[before] obtained is used to extract the code snippet using Tree-Sitter parser.
- **Fixed Code:** The location[after] obtained is used to extract the code snippet using Tree-Sitter parser.



Loc: [before: 25, after: 25]

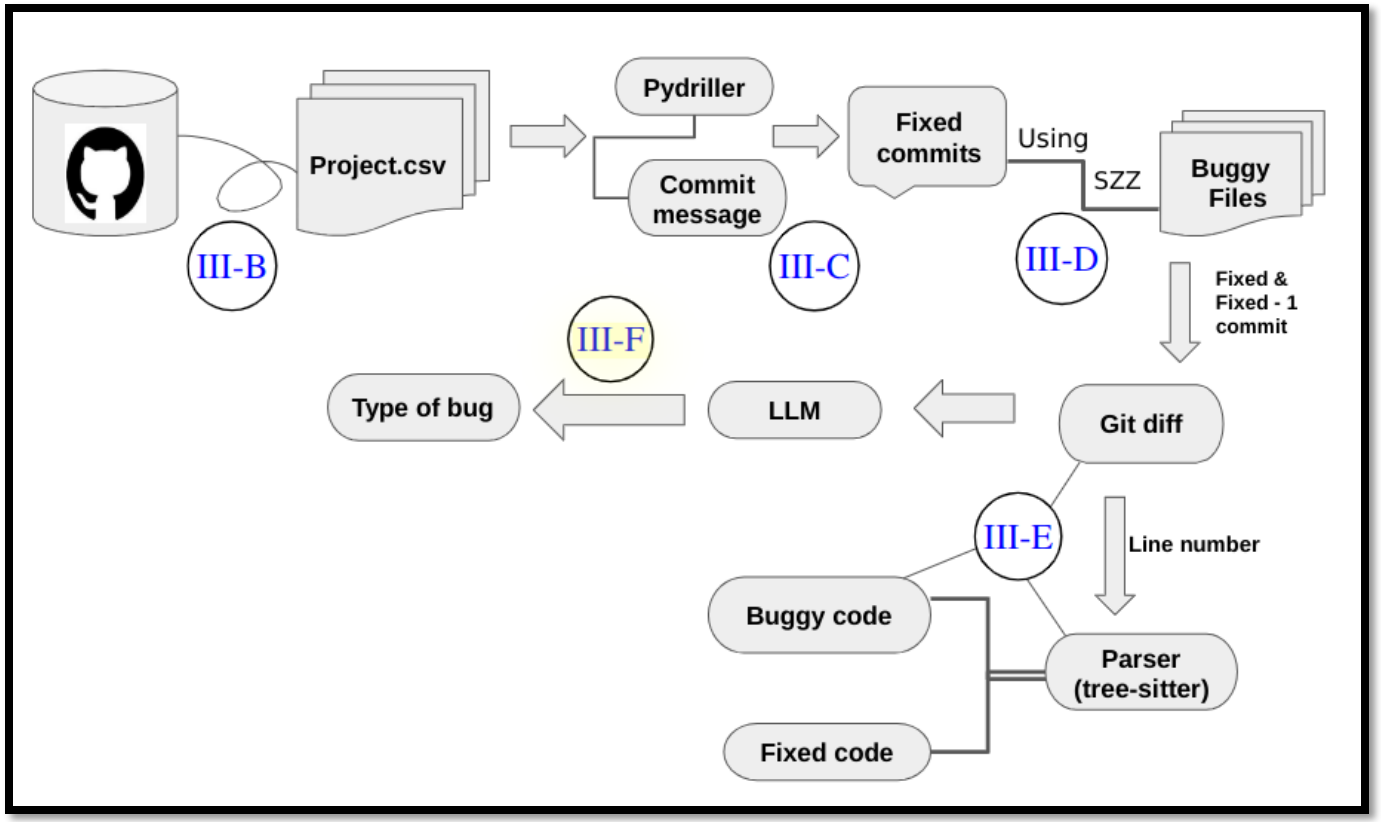
Bug Type: fix komoot.py when extent is none

Commit Message: fix calls to BBox with invalid input

Bug Fix Dataset Acquisition Flowchart with Precise Context

Data acquisition is done in following ways:

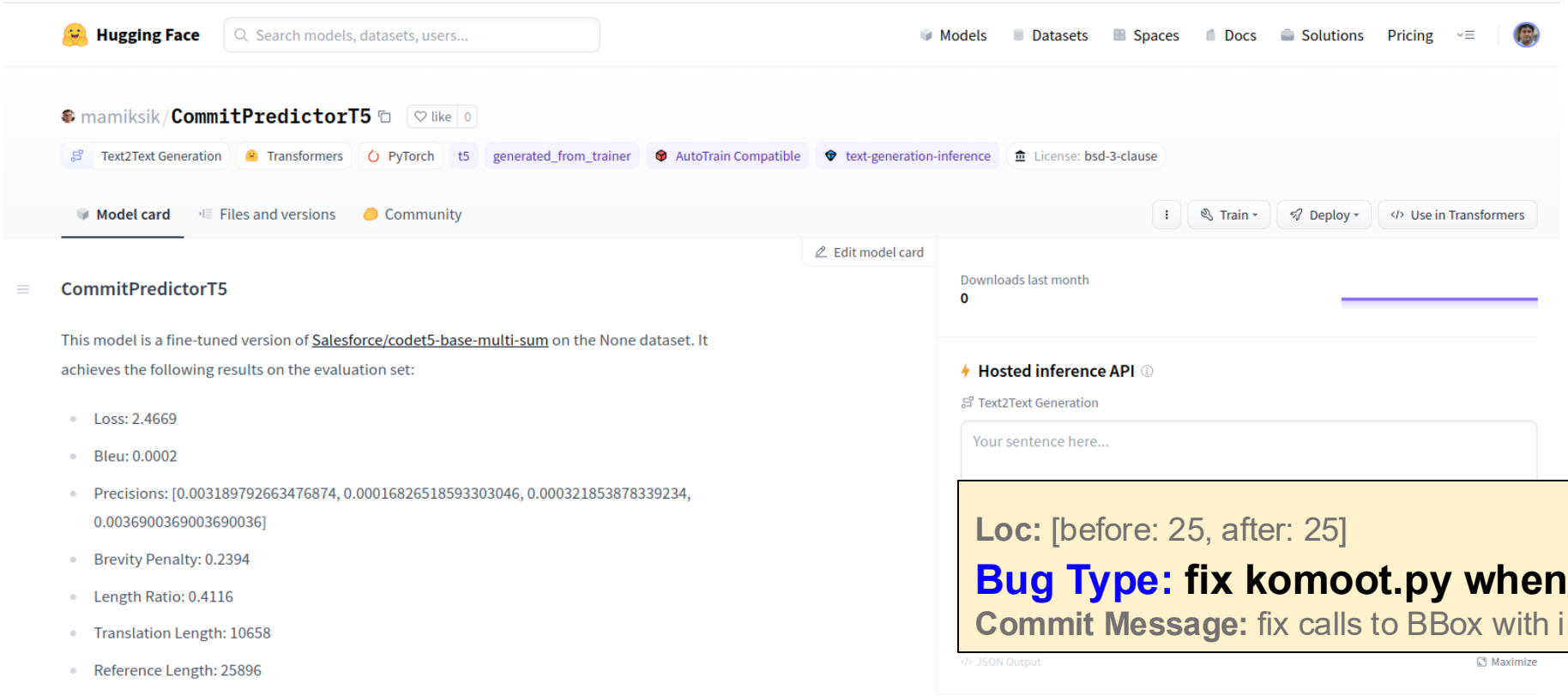
- Project Selection
- Bug fix commits
- Extracting Bug and fixed code Snippets
- **Type of the bug**



Inferencing the type of Bug

The type of bug is inferenced using a commit message generator Large Language Model (LLM).
URL: https://huggingface.co/mamiksik/CommitPredictorT5

The LLM takes `git-diff` of fixed - 1 and fixed commits as input and generates the summary of `diff` which is nothing but a fix and thus inferred as bug type.



Loc: [before: 25, after: 25]

Bug Type: fix komoot.py when extent is none

Commit Message: fix calls to BBox with invalid input

Texts, References, and Acknowledgements

Online:

- Continuous Integration and Delivery (**CircleCI**: https://circleci.com)

Textbook:

- Sharp, J. (2022). *Microsoft Visual C# Step by Step*, 10th edition, Microsoft Press.
- Watson, K., Nagel, C., Pedersen, J. H., Reid, J. D., & Skinner, M. (2008). *Beginning Microsoft Visual C# 2008*. John Wiley & Sons.
- Mark J. Price (2024). *C# 13 and .NET 9 – Modern Cross-Platform Development Fundamentals*, 9th edition, Packt Publishing Ltd.

Reference:

- Soni, M. (2016). *DevOps for Web Development*. Packt Publishing Ltd.
- Yusuf Sulistyo Nugroho, Hideaki Hata, and Kenichi Matsumoto. 2020. *How different are different diff algorithms in Git? Use* `--histogram` *for code changes*. Empirical Softw. Engg. 25, 1 (Jan 2020), 790–823.
- Sai Krishna Avula (2024), *Mining Software Bug Fixes and SAT Analysis: Dataset Creation and Tool Development for Improved Software Quality Assurance* - Awarded Gold Medal for the outstanding research (M.Tech.) at 13th Convocation IITGN.