WebSVF

Static Analysis Leveraging SVF and CodeGPT

UNSW | Software Engineering | T2 2025 | Christian Tolentino and Joshua Wills

Supervisor: Yulei Sui
Assessor: Yuekang Li

Overview

1. Project Summary
    a. Problem Statement
    b. General Context
    c. Revised Timeline
2. Joshua’s Improvements
    a. Import/Export Functionality
    b. Software Clean Up
    c. Multi File Support
3. Christian’s Improvements
    a. SVF Python Backend Conversion
    b. General UI Improvements
4. Future Work

Problem Statement

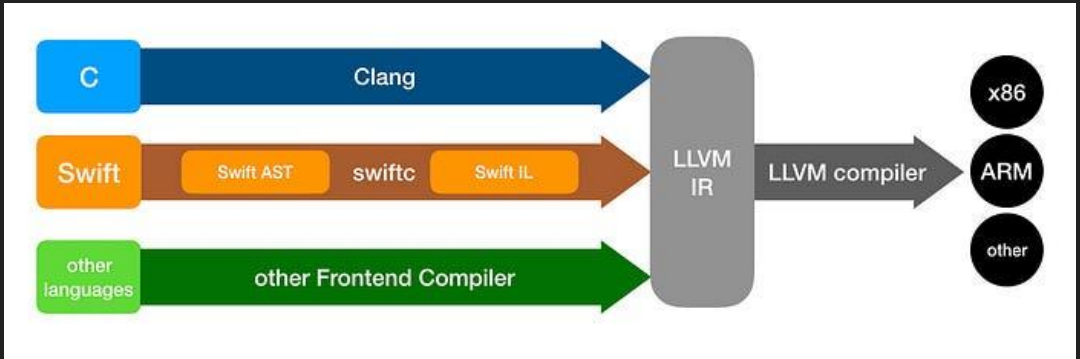
There is a significant problem in modern software development relating to the identification of bugs and security risks in large, complicated applications. As the complexity of such applications grows, the difficulty of identifying such issues worsens. The goal of WebSVF, including CodeGPT, is to provide a simple interface for users to easily identify such bugs and security risks. This is achieved by leveraging SVF, the Static Value-Flow Analysis Framework, and modern LLMs to assist in the detection of vulnerabilities, generate graphs of code structure and explain blocks of software.

General Context | Static Analysis

- Compiling and abstractly executing a program without running it
- Gain understand of software project’s internal structure
- Identify common programming errors
  - Memory leaks
  - Buffer overflows
  - Null pointer dereference
  - etc.

General Context | LLVM

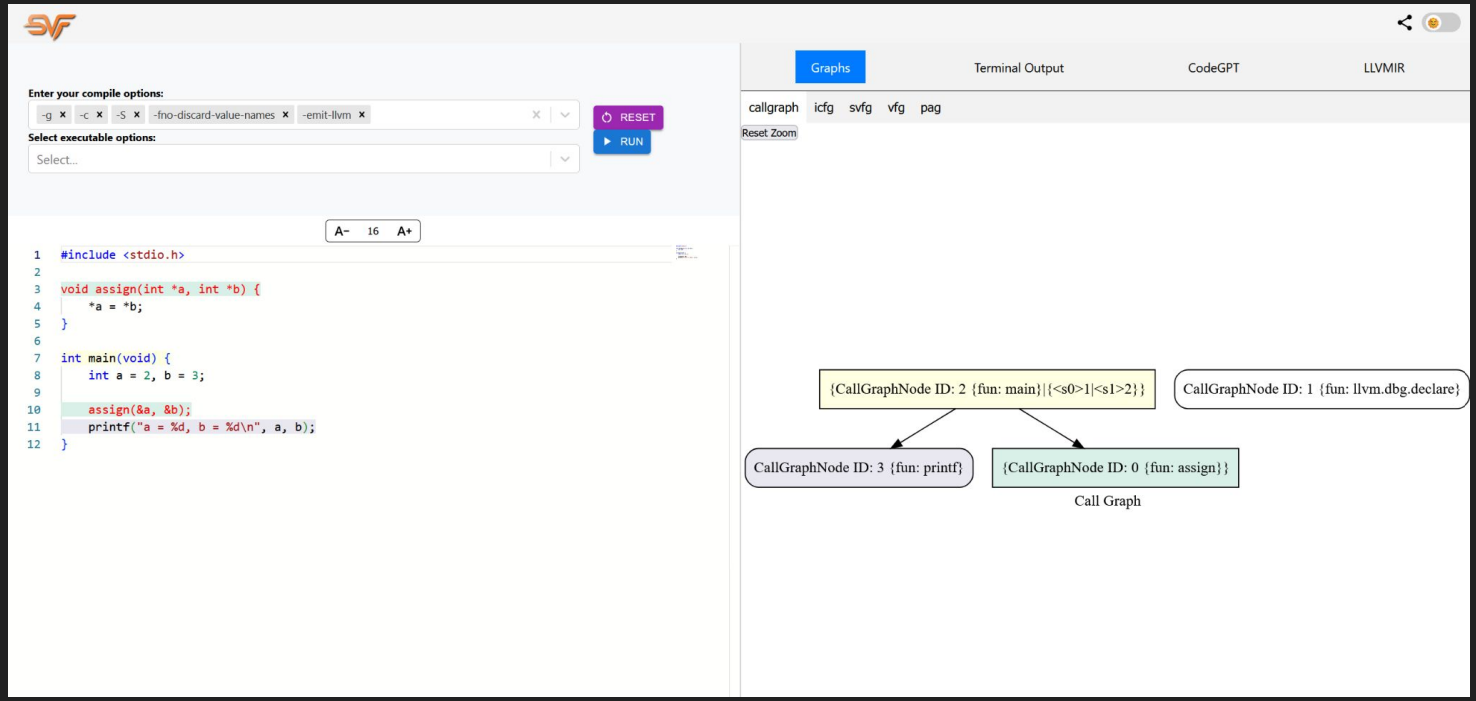
- Short for ‘Low Level Virtual Machine’
- A compiler framework, establishing a low-level, platform-generic IR
- IR written in Single Static Assignment (SSA) form.
- SVF operates on LLVM bitcode, enabling multi language support.



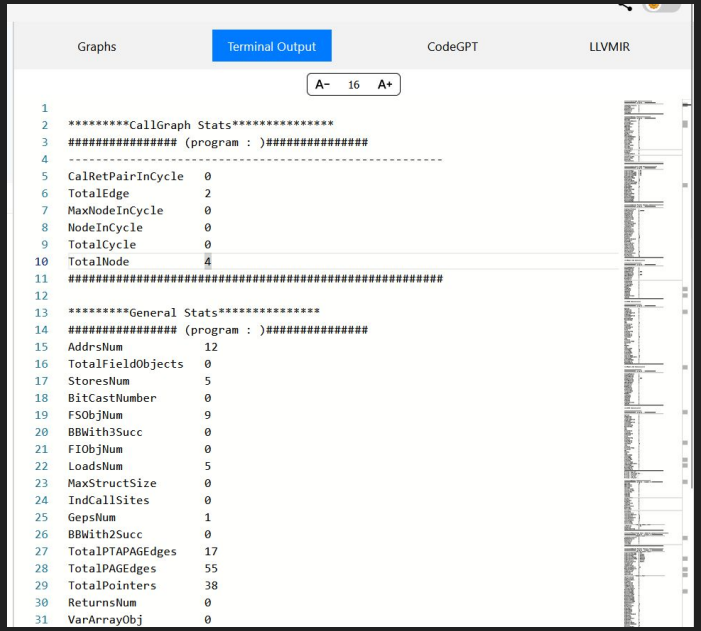
General Context | SVF

- Static Value-Flow Analysis Framework
- “Source code analysis tool that enables interprocedural dependence analysis for LLVM-based languages.”

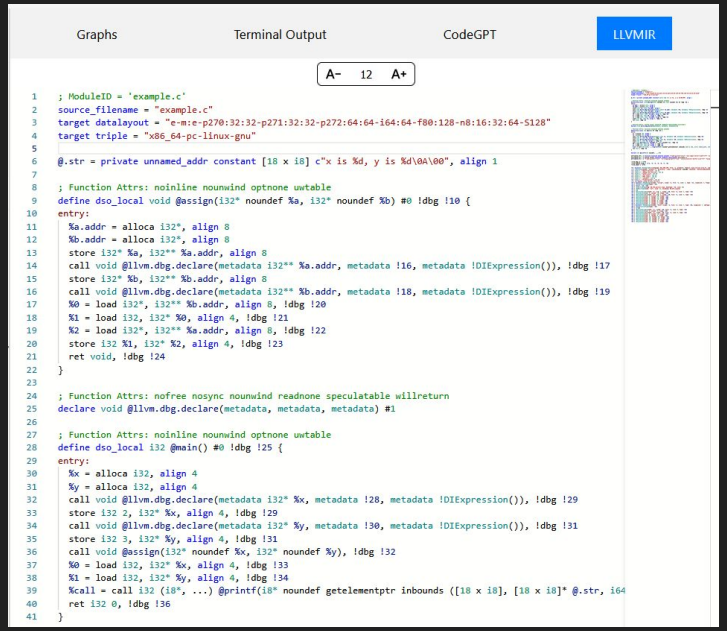
General Context | WebSVF



Context | WebSVF

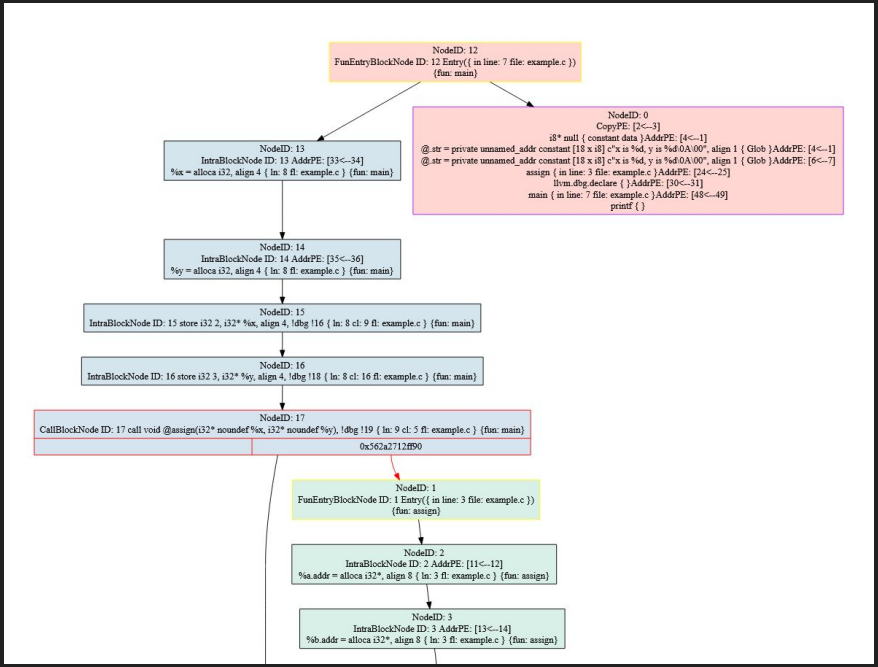


Example of the terminal output

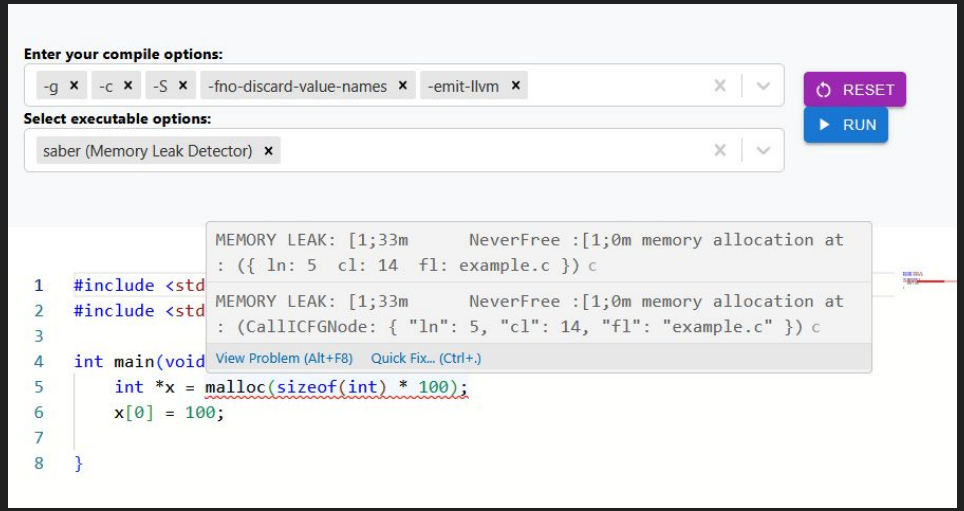


Example of the LLVM IR output

Context | WebSVF



Example of an Interprocedural Control Flow Graph (ICFG)



SABER executable identifying a memory leak

Revised Timeline | Josh

Thesis B

- Continuous cycle of development, reaffirming ideas with stakeholders
- Import/Export functionality
- Code clean up and automated linting
- Adding IDE-like support

Thesis C

- Further work on the IDE-like experience
- Adding support for Rust/C++
- Integrating more executables/graph options
- Various small UI improvements

Revised Timeline | Christian

Thesis B

- Convert WebSVF Backend to use SVF-Python
- Revamp and redesign UI, fix bugs and performance, improve dark-mode and enhance user accessibility

Thesis C

- Onboarding guide for new users
- Support keybindings for an IDE-like experience
- Provide more clarity on terminal output, LLVM through tooltips and informative modals
- Analyse effectiveness of new features on user engagement, understanding and retention

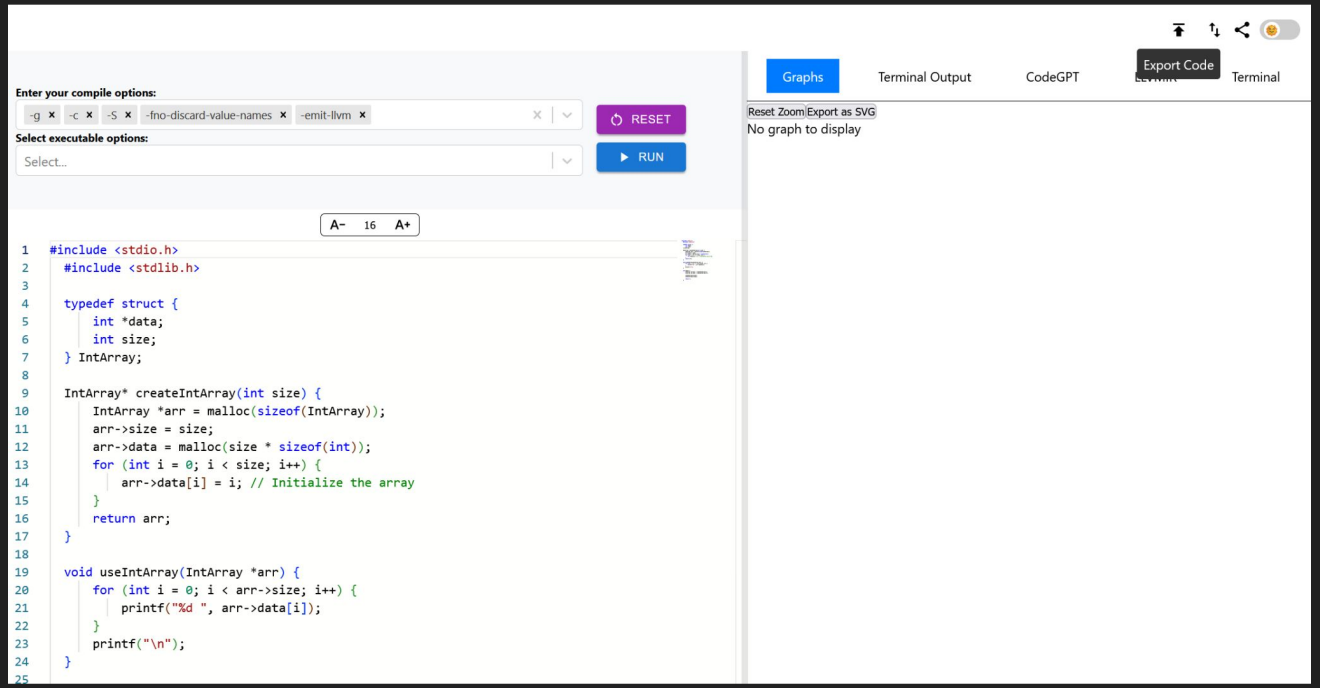
Josh’s Improvements

Import/Export Code

- **Why it was done:**
  - Easy ability to load/store coding projects between a user's browser and their local file system
  - Easy integration into a user's existing workflow
  - Able to apply local file versioning/memory redundancy not possible in the browser.
  - Make WebSVF more like an IDE
    - Increase user-base => greater UX and more OSS contributions

Import/Export Code

- **What was done:**



Import/Export buttons added to the WebSVF navbar

Import/Export Code

Browser compatibility

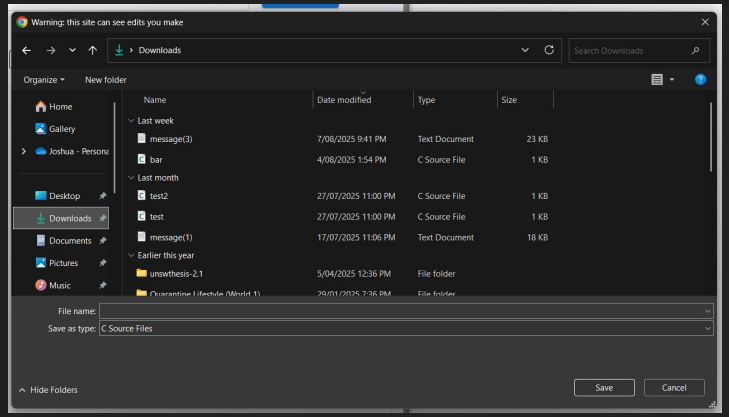
Report problems with this compatibility data • View data on GitHub

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android	Safari on iOS	Samsung Internet	WebView Android	WebView on iOS	Deno
showSaveFilePicker	✓ 86	✓ 86	✗ No	✓ 72	✗ No	✓ 132	✗ No	✓ 87	✗ No	✗ No	✓ 132	✗ No	✗ No

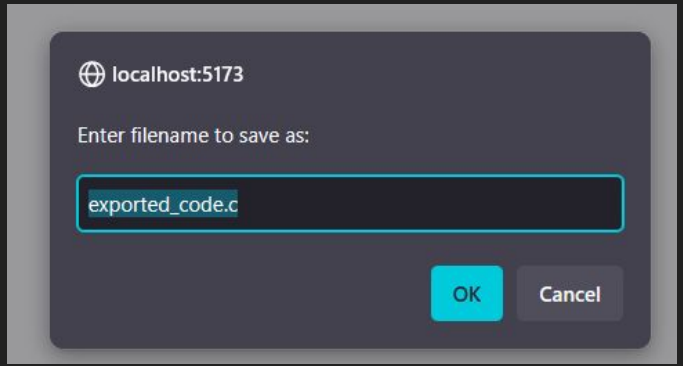
Tip: you can click/tap on a cell for more information.

✓ Full support ✗ No support Experimental. Expect behavior to change in the future.

MDN Web Docs re. showSaveFilePicker() (Aug 2025)

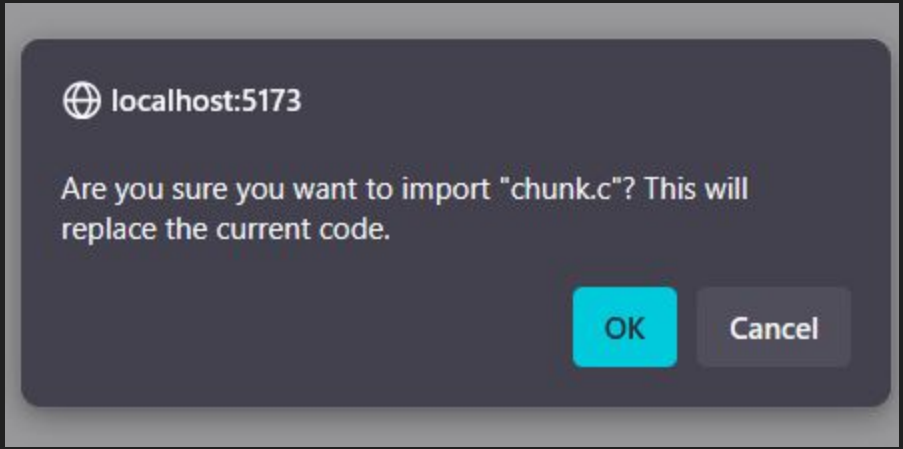


Supported browser (Chrome)



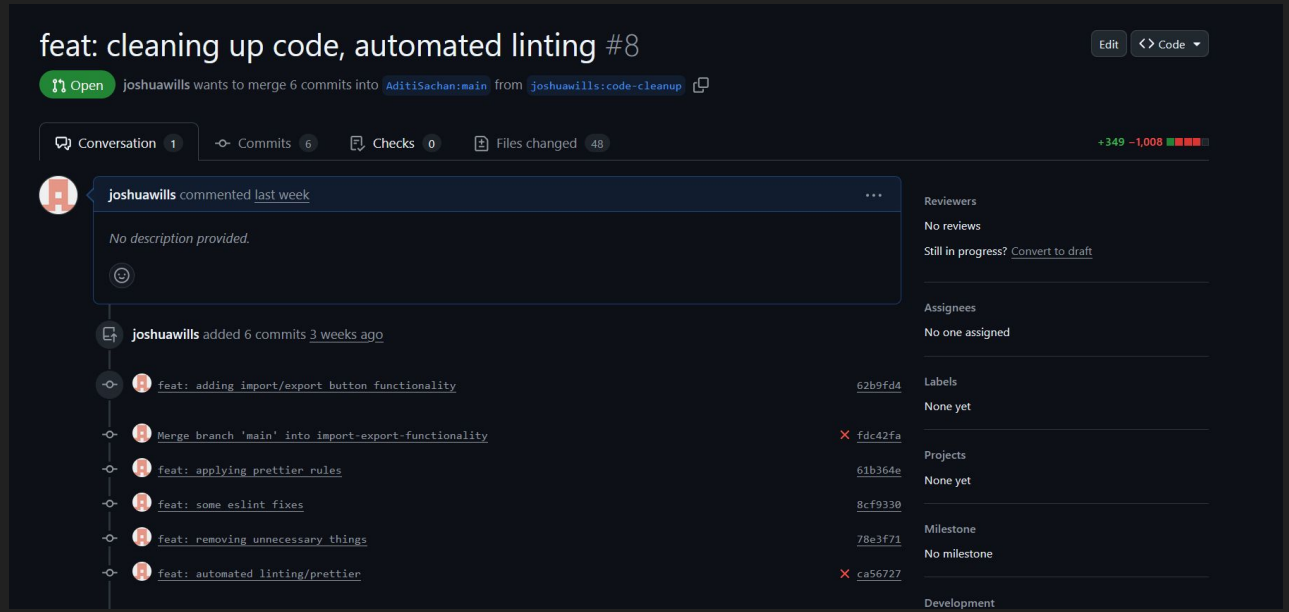
Unsupported browser (Firefox)

Import/Export Code



Browser warning before import

Software Clean Up



PR adding in cleaned up code

Software Clean Up

Code | Blame | 35 lines (27 loc) · 711 Bytes

```yaml
name: Lint and Format Check

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  lint-format:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Set up Node.js
        uses: actions/setup-node@v4
        with:
          node-version: 18
          cache: npm
          cache-dependency-path: frontend/package-lock.json

      - name: Install dependencies
        run: npm ci
        working-directory: frontend

      - name: Run Prettier check
        run: npm run format:check
        working-directory: frontend

      - name: Run ESLint check
        run: npm run lint
        working-directory: frontend
```

Automated linting and formatting script

Multi File Support

- **Why it was done:**
  - Make WebSVF more feasible for real-world projects
  - Create an IDE-like environment
  - Easy onboarding from/to existing projects
  - Later On:
    - Support custom compilation
    - More languages supported (Rust/C++)

Christian’s Improvements

Python SVF Backend Conversion

Why it was done:

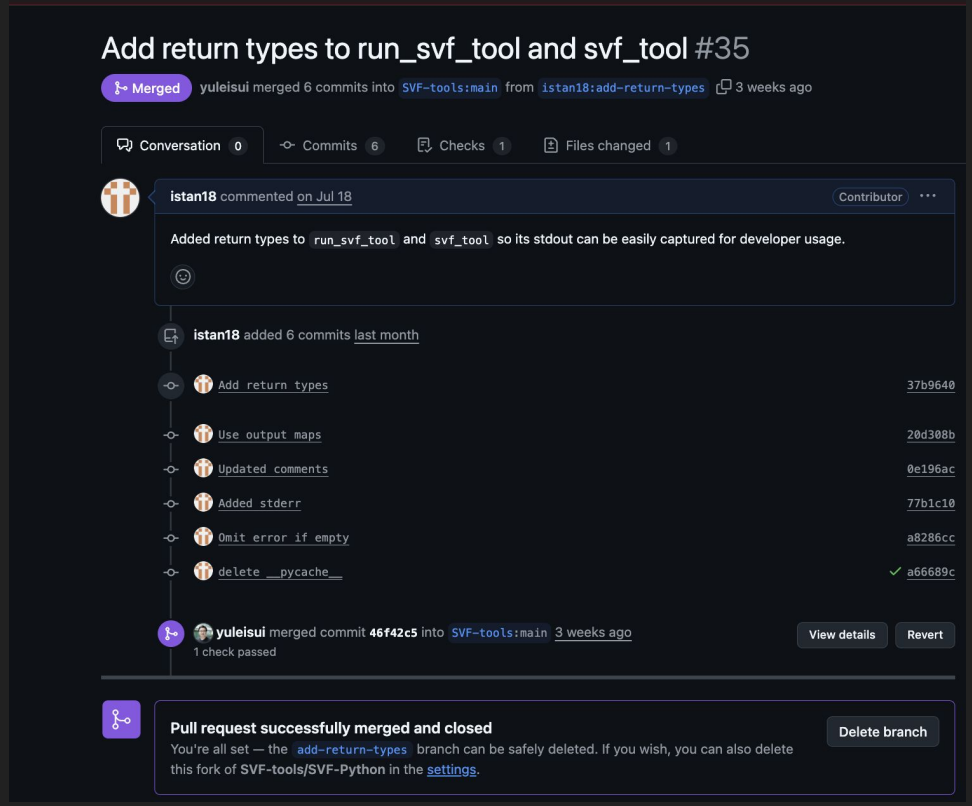
- Easier open-source contributions as Python is a more commonly used language than C++/C#.
- Provides examples for users to use Python-SVF in their own projects
- Allows for more native support and extensibility since there is a Python API.
- Improves debugging, readability and dependency management

Python SVF Backend Conversion

What was done:

- Updated the main Python-SVF repository to capture stdout and stderr from tool calls
- Converted the WebSVF Backend to use Python FastAPI instead of C# .NET
  - Instead of running SVF on the command line via subprocesses, tool calls are done via `pysvf.run_svf_tool(args)`.
- Updated Dockerfile, README.md documentation and setup guides

Python SVF Backend Conversion



Add return types to run_svf_tool and svf_tool #35

Merged yuleisui merged 6 commits into SVF-tools:main from istan18:add-return-types 3 weeks ago

Conversation 0 Commits 6 Checks 1 Files changed 1

istan18 commented on Jul 18 — Contributor

Added return types to `run_svf_tool` and `svf_tool` so its stdout can be easily captured for developer usage.

istan18 added 6 commits last month

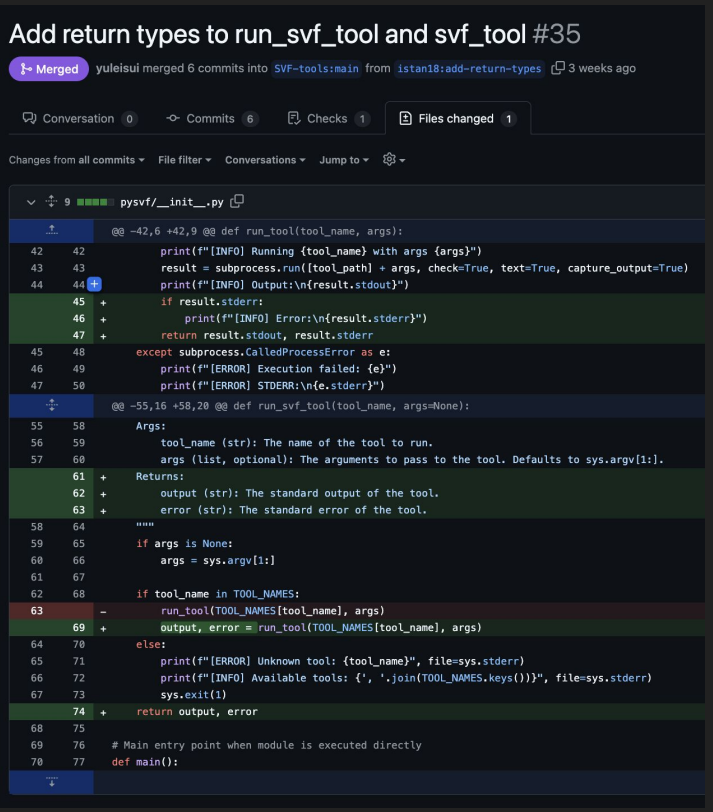
- Add return types — 37b9640
- Use output maps — 20d308b
- Updated comments — 0e196ac
- Added stderr — 77b1c10
- Omit error if empty — a8286cc
- delete __pycache__ — a66689c

yuleisui merged commit **46f42c5** into SVF-tools:main 3 weeks ago — View details — Revert

1 check passed

Pull request successfully merged and closed

You're all set — the `add-return-types` branch can be safely deleted. If you wish, you can also delete this fork of **SVF-tools/SVF-Python** in the settings. — Delete branch



Add return types to run_svf_tool and svf_tool #35

Merged yuleisui merged 6 commits into SVF-tools:main from istan18:add-return-types 3 weeks ago

Conversation 0 Commits 6 Checks 1 Files changed 1

Changes from all commits — File filter — Conversations — Jump to

pysvf/__init__.py

```
@@ -42,6 +42,9 @@ def run_tool(tool_name, args):
42 42        print(f"[INFO] Running {tool_name} with args {args}")
43 43        result = subprocess.run([tool_path] + args, check=True, text=True, capture_output=True)
44 44        print(f"[INFO] Output:\n{result.stdout}")
   45 +      if result.stderr:
   46 +          print(f"[INFO] Error:\n{result.stderr}")
   47 +      return result.stdout, result.stderr
45 48    except subprocess.CalledProcessError as e:
46 49        print(f"[ERROR] Execution failed: {e}")
47 50        print(f"[ERROR] STDERR:\n{e.stderr}")
@@ -55,16 +58,20 @@ def run_svf_tool(tool_name, args=None):
55 58    Args:
56 59        tool_name (str): The name of the tool to run.
57 60        args (list, optional): The arguments to pass to the tool. Defaults to sys.argv[1:].
   61 +  Returns:
   62 +      output (str): The standard output of the tool.
   63 +      error (str): The standard error of the tool.
58 64    """
59 65    if args is None:
60 66        args = sys.argv[1:]
61 67
62 68    if tool_name in TOOL_NAMES:
63    -      run_tool(TOOL_NAMES[tool_name], args)
   69 +      output, error = run_tool(TOOL_NAMES[tool_name], args)
64 70    else:
65 71        print(f"[ERROR] Unknown tool: {tool_name}", file=sys.stderr)
66 72        print(f"[INFO] Available tools: {', '.join(TOOL_NAMES.keys())}", file=sys.stderr)
67 73        sys.exit(1)
   74 +  return output, error
68 75
69 76  # Main entry point when module is executed directly
70 77  def main():
```

UI Improvements

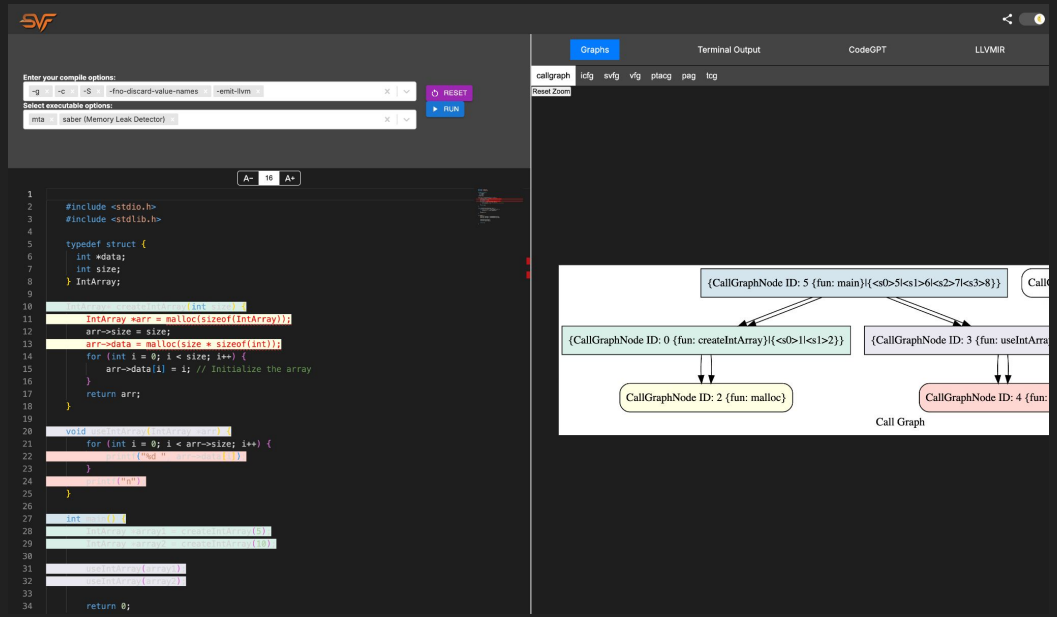
Why it was done:

- Increases user productivity and efficiency by making WebSVF easier to understand
- More responsive and interactive UI -> improves user engagement, retention and conversion
- 88% of users won’t return after a bad UI/UX experience (HubSpot)

UI/UX Improvements - Dark Mode

- Multiple spacing issues
- Poor colour contrast
- Inconsistent button design
- Laggy, delayed responsiveness
- Edges are too rough
- Some components not fully converted to dark mode equivalent

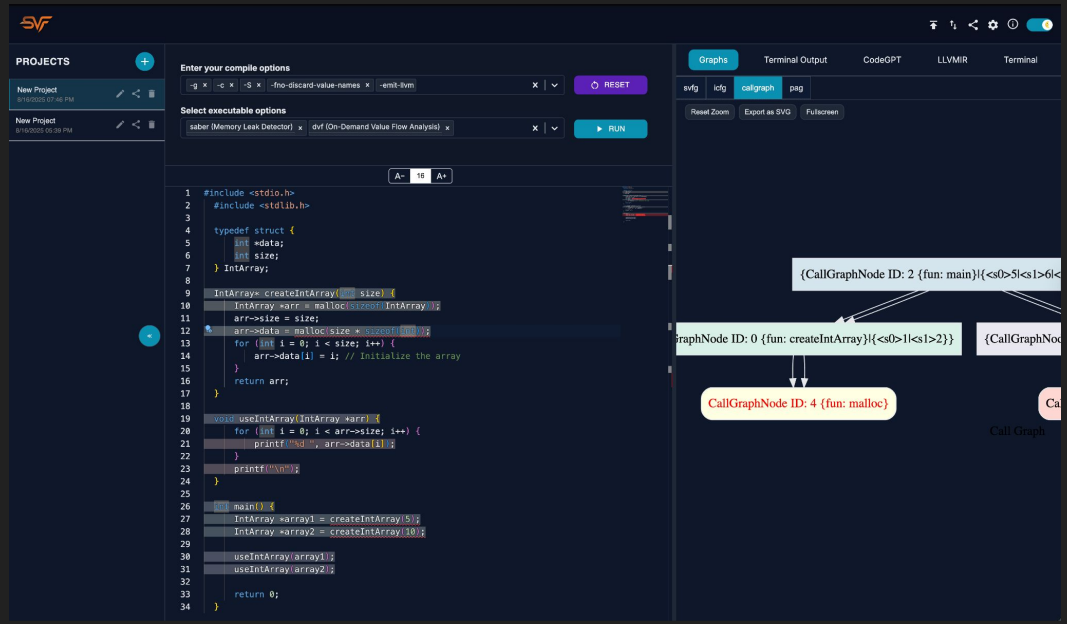
Old Version



UI/UX Improvements - Dark Mode

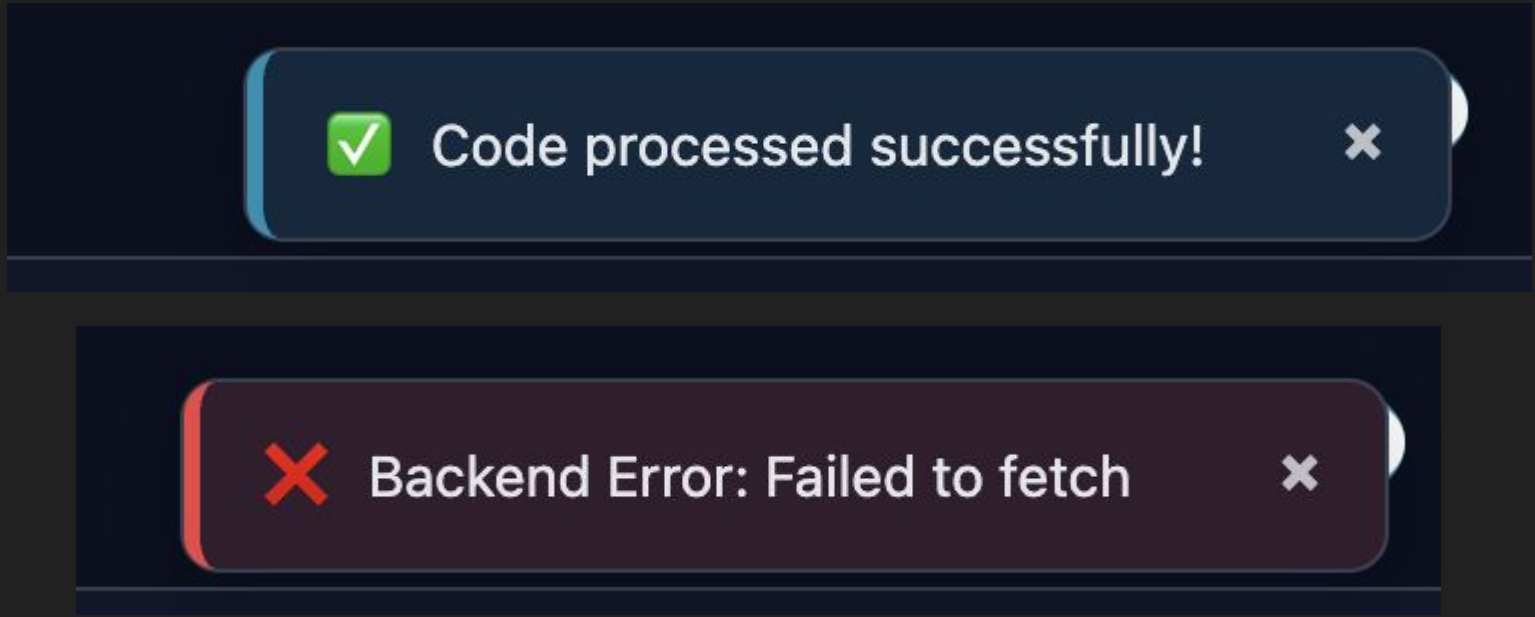
- Consistent colour design
- Consistent spacing
- Rounded and consistent buttons
- Clear colour contrast
- Rounded corners
- Consistent with light mode design

New Version



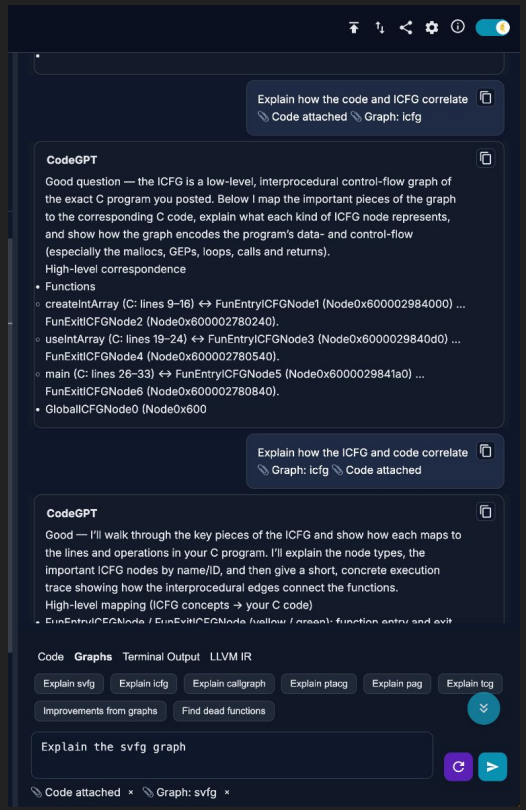
UI/UX Improvements - Fullscreen

UI/UX Improvements - Toast



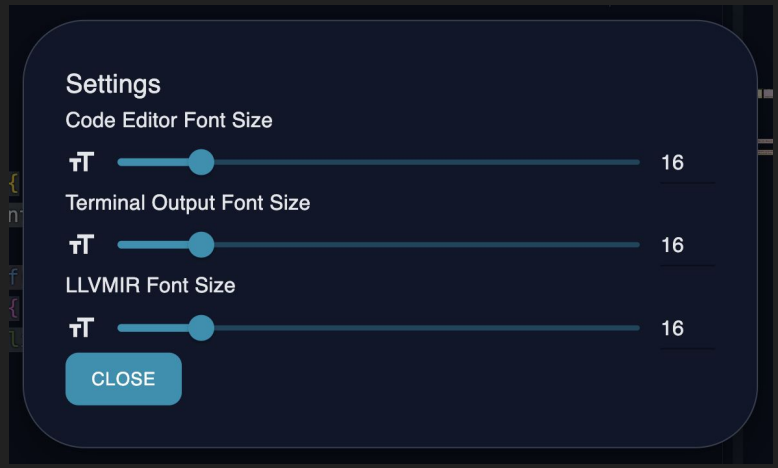
UI/UX Improvements - CodeGPT

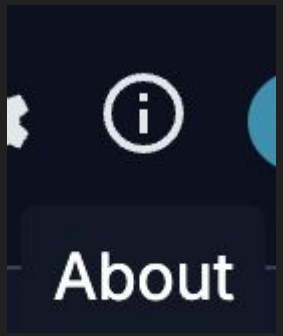
- Updated CodeGPT to use the latest GPT-5 model with web search capabilities
- Added copy feature
- Ability to add and delete multiple attachments (code and multiple graphs)
- Reduce chatbox bloat when adding attachments
- General chat formatting and UI improvements

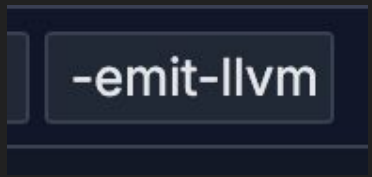


UI/UX Improvements - General

- Added about page -> allows users to gather context about SVF and WebSVF
- Added settings page
- Fixed bugs and error handling







Questions/Feedback