



# 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

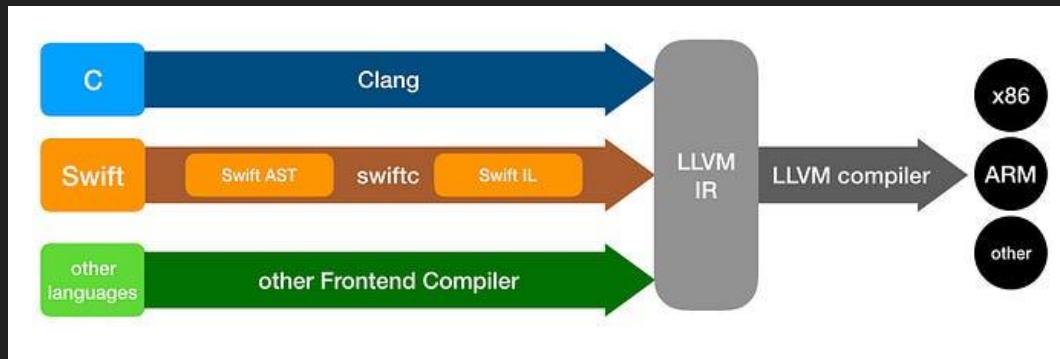
There is a significant problem in modern software development relating to the identification of bugs and security risks in large, complicated applications. As the complexity of such applications grows, the difficulty of identifying such issues worsens. The goal of WebSVF, including CodeGPT, is to provide 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

The screenshot shows the WebSVF interface with the following components:

- Code Editor:** Displays the following C code:

```
1 #include <stdio.h>
2
3 void assign(int *a, int *b) {
4     *a = *b;
5 }
6
7 int main(void) {
8     int a = 2, b = 3;
9
10    assign(&a, &b);
11    printf("a = %d, b = %d\n", a, b);
12 }
```
- Compile Options:** Shows command-line options: `-g x -c x -S x -fno-discard-value-names x -emit-llvm x`. Buttons for **RESET** and **RUN** are present.
- Graphs Tab:** Active tab. Sub-tabs include: callgraph, icfg, svfg, vfg, pag. A "Reset Zoom" button is available.
- Call Graph:** Visual representation of the control flow between functions.

```
graph TD
    N2[CallGraphNode ID: 2 {fun: main}|{<s0>1|<s1>2|}] --> N3[CallGraphNode ID: 3 {fun: printf}]
    N2 --> N0[CallGraphNode ID: 0 {fun: assign}]
    N1[CallGraphNode ID: 1 {fun: llvm.dbg.declare}]
```

The call graph shows three nodes:

  - CallGraphNode ID: 2 (fun: main) has two outgoing edges: one to CallGraphNode ID: 3 (fun: printf) and one to CallGraphNode ID: 0 (fun: assign).
  - CallGraphNode ID: 1 (fun: llvm.dbg.declare) has no outgoing edges.
  - CallGraphNode ID: 3 (fun: printf) has no outgoing edges.

# Context | WebSVF

Graphs      Terminal Output      CodeGPT      LLVMIR

A- 16 A+

```
1 ****CallGraph Stats*****
2 ###### (program : )#####
3
4 -----
5 CalRetPairInCycle 0
6 TotalFedge 2
7 MaxNodeInCycle 0
8 NodeInCycle 0
9 TotalCycle 0
10 TotalNode 4
11 #####
12 ****General Stats*****
13 ###### (program : )#####
14
15 AddrNum 12
16 TotalFieldObjects 0
17 StoresNum 5
18 SbitCastNumber 0
19 FSObjNum 9
20 BBWith3Succ 0
21 FIObjNum 0
22 LoadsNum 5
23 MaxStructSize 0
24 IndCallsites 0
25 GepsNum 1
26 BBWith2Succ 0
27 TotalIPTAPAGEdges 17
28 TotalPAGEdges 55
29 TotalPointers 38
30 ReturnsNum 0
31 VarArrayObj 0
```

Example of the terminal output

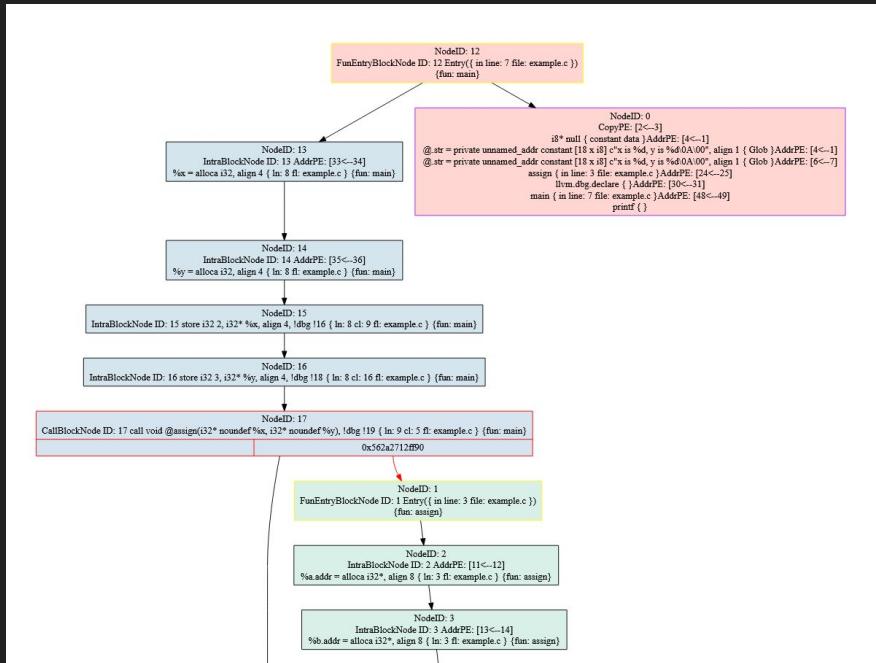
Graphs      Terminal Output      CodeGPT      LLVMIR

A- 12 A+

```
1 ; ModuleID = 'example.c'
2 source_filename = "example.c"
3 target datalayout = "e-m:e-p270:32:32-p271:32:32-p272:64:64-164:64-f80:128-n8:16:32:64-S128"
4 target triple = "x86_64-pc-linux-gnu"
5
6 @.str = private unnamed_addr constant [18 x i8] c"x is %d, y is %d\bA\b0", align 1
7
8 ; Function Attrs: noinline nounwind optnone untable
9 define dso_local void @ssign(i32* noundef %a, i32* noundef %b) #0 !dbg !10 {
10   entries:
11   .L1:
12     %a.addr = alloca i32*, align 8
13     %b.addr = alloca i32*, align 8
14     store i32 %a, i32** %a.addr, align 8
15     call void @_lvm.debug.declare(metadata i32** %a.addr, metadata !16, metadata !DIExpression()), !dbg !17
16     store i32 %b, i32** %b.addr, align 8
17     call void @_lvm.debug.declare(metadata i32** %b.addr, metadata !18, metadata !DIExpression()), !dbg !19
18     %0 = load i32*, i32** %b.addr, align 8, !dbg !20
19     %1 = load i32, i32* %0, align 4, !dbg !21
20     %2 = load i32*, i32** %a.addr, align 8, !dbg !22
21     store i32 %1, i32* %2, align 4, !dbg !23
22   ret void, !dbg !24
23 }
24
25 ; Function Attrs: nofree nosync nounwind readonly speculatable willreturn
26 declare void @_lvm.debug.declare(metadata, metadata, metadata) #1
27
28 ; Function Attrs: noinline nounwind optnone untable
29 define dso_local i32 @main() #0 !dbg !25 {
30   entries:
31   .L1:
32     %x = alloca i32, align 4
33     %y = alloca i32, align 4
34     call void @_lvm.debug.declare(metadata i32* %x, metadata !28, metadata !DIExpression()), !dbg !29
35     store i32 2, i32* %x, align 4, !dbg !29
36     call void @_lvm.debug.declare(metadata i32* %y, metadata !30, metadata !DIExpression()), !dbg !31
37     store i32 3, i32* %y, align 4, !dbg !31
38     call void @ssign(i32* noundef %x, i32* noundef %y), !dbg !32
39     %0 = load i32, i32* %x, align 4, !dbg !33
40     %1 = load i32, i32* %y, align 4, !dbg !34
41     %call = call i32 (i8*, ...) @printf(i8* noundef getelementptr inbounds ([18 x i8], [18 x i8]* @.str, i64
42     ret i32 0, !dbg !36
43 }
```

Example of the LLVM IR output

# Context | WebSVF



Example of an Interprocedural Control Flow Graph (ICFG)

Enter your compile options:  
 -g x -c x -S x -fno-discard-value-names x -emit-llvm x

Select executable options:  
 saber (Memory Leak Detector) x

RESTART RUN

MEMORY LEAK: [1;33m NeverFree :[1;0m memory allocation at : ({ ln: 5 cl: 14 fl: example.c }) c

```

1 #include <std
2 #include <std
3
4 int main(void
5     int *x = malloc(sizeof(int) * 100);
6     x[0] = 100;
7 }
8 
```

View Problem (Alt+F8) Quick Fix... (Ctrl+.)

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:

The screenshot shows the WebSVF interface with a code editor containing C code for creating and using an integer array. The code includes #include directives for stdio.h and stdlib.h, a struct definition for IntArray, a function createIntArray that allocates memory and initializes the array, and a function useIntArray that prints the array elements. Above the code editor are sections for 'Enter your compile options:' and 'Select executable options:', both currently empty. To the right of the code editor is a toolbar with buttons for 'Graphs' (highlighted in blue), 'Terminal Output', 'CodeGPT', and 'Export Code' (with sub-options 'Export as SVG' and 'Terminal').

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 typedef struct {
5     int *data;
6     int size;
7 } IntArray;
8
9 IntArray* createIntArray(int size) {
10    IntArray *arr = malloc(sizeof(IntArray));
11    arr->size = size;
12    arr->data = malloc(size * sizeof(int));
13    for (int i = 0; i < size; i++) {
14        arr->data[i] = i; // Initialize the array
15    }
16    return arr;
17 }
18
19 void useIntArray(IntArray *arr) {
20    for (int i = 0; i < arr->size; i++) {
21        printf("%d ", arr->data[i]);
22    }
23    printf("\n");
24 }
25
```

Import/Export buttons added to the WebSVF navbar

# Import/Export Code

## Browser compatibility

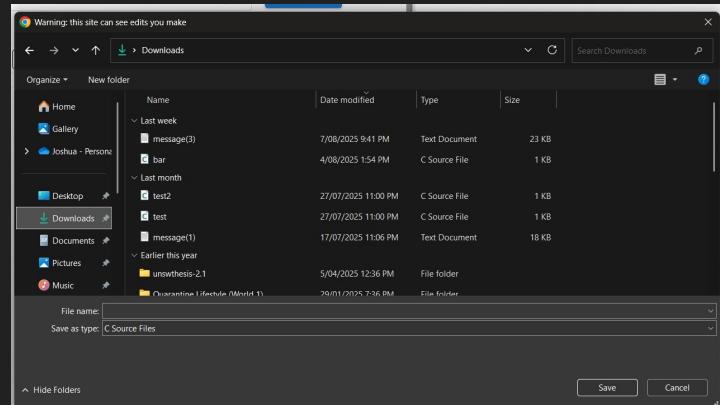
Report problems with this compatibility data • View data on GitHub

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

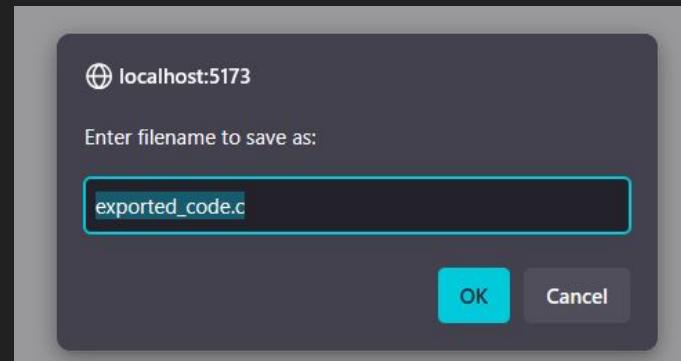
Tip: you can click/tap on a cell for more information.

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

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

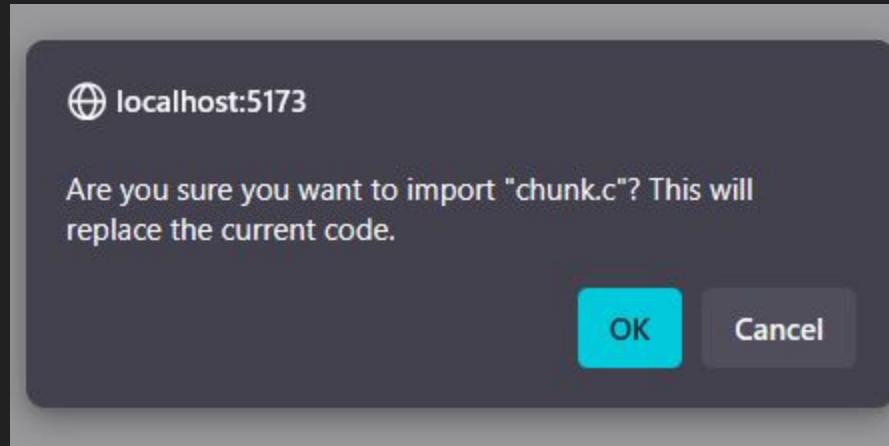


Supported browser (Chrome)



Unsupported browser (Firefox)

# Import/Export Code



Browser warning before import

# Software Clean Up

feat: cleaning up code, automated linting #8

**Open** joshuawills wants to merge 6 commits into [AditiSachan:main](#) from [joshuawills:code-cleanup](#)

Conversation 1 Commits 6 Checks 0 Files changed 48 +349 -1,008

joshuawills commented last week  
No description provided.

joshuawills added 6 commits 3 weeks ago

- feat: adding import/export button functionality 62b9fd4
- Merge branch 'main' into import-export-functionality fdc42fa
- feat: applying prettier rules 61b364e
- feat: some eslint fixes 8cf9330
- feat: removing unnecessary things 78e3f71
- feat: automated linting/prettier ca56727

Reviewers  
No reviews  
Still in progress? [Convert to draft](#)

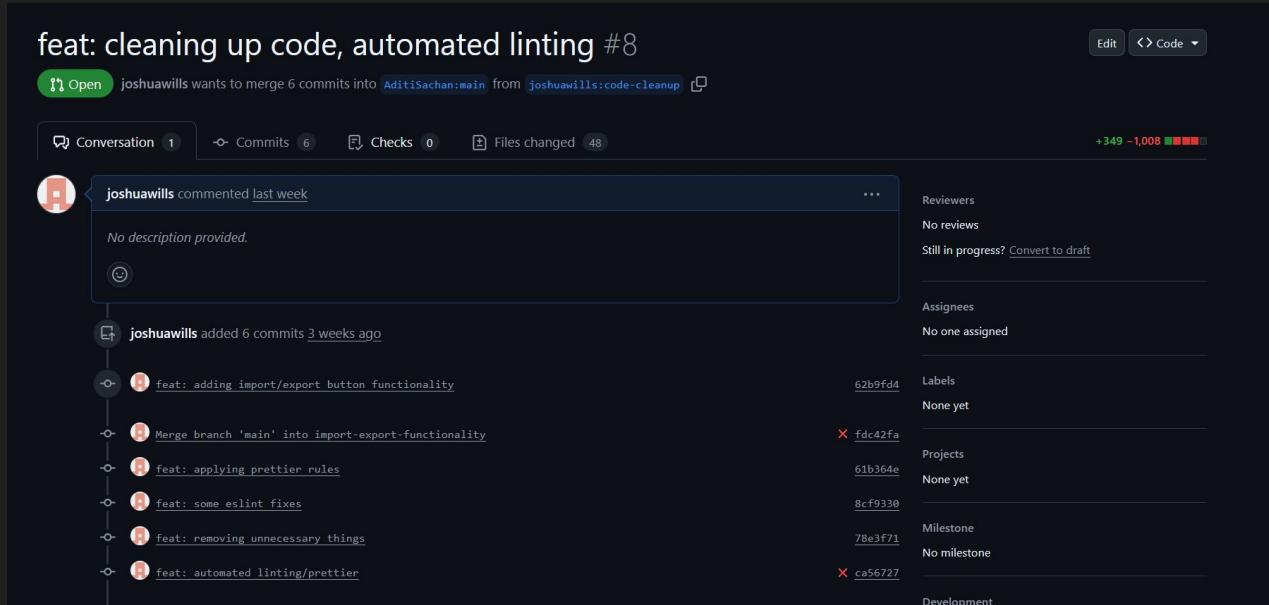
Assignees  
No one assigned

Labels  
None yet

Projects  
None yet

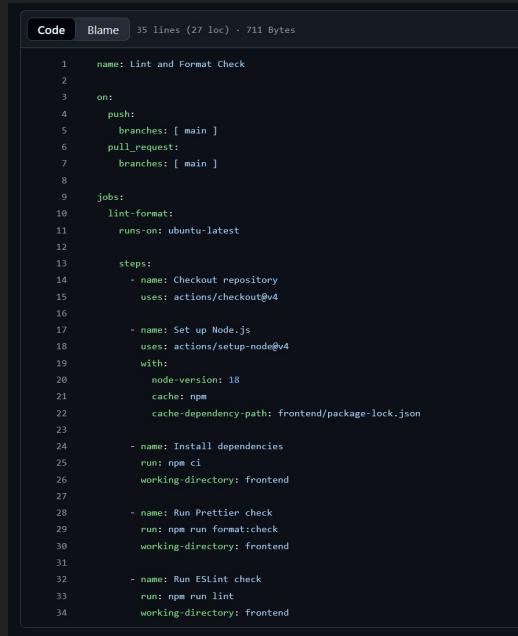
Milestone  
No milestone

Development



PR adding in cleaned up code

# Software Clean Up



A screenshot of a code editor displaying a GitHub Actions workflow file. The file is titled "name: Lint and Format Check". It defines an "on" section for pushes to the main branch and pull requests to the main branch. The "jobs" section contains a single job named "lint-format" which runs on an Ubuntu-latest runner. This job has several steps: 1. Checkout repository using actions/checkout@v4. 2. Set up Node.js using actions/setup-node@v4 with node-version: 18 and cache: npm, setting cache-dependency-path to frontend/package-lock.json. 3. Install dependencies using npm ci in the frontend directory. 4. Run Prettier check using npm run format:check in the frontend directory. 5. Run ESLint check using npm run lint in the frontend directory.

```
1  name: Lint and Format Check
2
3  on:
4    push:
5      branches: [ main ]
6    pull_request:
7      branches: [ main ]
8
9  jobs:
10   lint-format:
11     runs-on: ubuntu-latest
12
13   steps:
14     - name: Checkout repository
15       uses: actions/checkout@v4
16
17     - name: Set up Node.js
18       uses: actions/setup-node@v4
19       with:
20         node-version: 18
21         cache: npm
22         cache-dependency-path: frontend/package-lock.json
23
24     - name: Install dependencies
25       run: npm ci
26       working-directory: frontend
27
28     - name: Run Prettier check
29       run: npm run format:check
30       working-directory: frontend
31
32     - name: Run ESLint check
33       run: npm run lint
34       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

a6286cc

delete \_\_pycache\_\_

a66689c

yuleisui merged commit 46f42c5 into SVF-tools:main 3 weeks ago

1 check passed

View details

Revert



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 ▾ ⚙

```
diff --git a/pysvf/_init_.py b/pysvf/_init_.py
@@ -42,6 +42,9 @@ def run_tool(tool_name, args):
    print("[INFO] Running {tool_name} with args {args}")
    result = subprocess.run([tool_path] + args, check=True, text=True, capture_output=True)
    print("[INFO] Output:\n{result.stdout}")
+   if result.stderr:
+       print("[INFO] Error:{result.stderr}")
+   return result.stdout, result.stderr
except subprocess.CalledProcessError as e:
    print("[ERROR] Execution failed: {e}")
    print("[ERROR] STDERR:{e.stderr}")
@@ -55,16 +58,20 @@ def run_svf_tool(tool_name, args=None):
    Args:
        tool_name (str): The name of the tool to run.
        args (list, optional): The arguments to pass to the tool. Defaults to sys.argv[1].
+       Returns:
+           output (str): The standard output of the tool.
+           error (str): The standard error of the tool.
+
+       """
+       if args is None:
+           args = sys.argv[1:]
+
+       if tool_name in TOOL_NAMES:
+           run_tool(TOOL_NAMES[tool_name], args)
+           output, error = run_tool(TOOL_NAMES[tool_name], args)
+       else:
+           print("[ERROR] Unknown tool: {tool_name}", file=sys.stderr)
+           print("[INFO] Available tools: {', '.join(TOOL_NAMES.keys())}", file=sys.stderr)
+           sys.exit(1)
+
+       return output, error
@@ -75,76 +76,76 @@ def main():
    # Main entry point when module is executed directly
    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

The screenshot shows the old version of the SVF interface. On the left is a code editor with the following C code:

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
    int *data;
    int size;
} IntArray;

IntArray* createIntArray(int size) {
    IntArray *arr = malloc(sizeof(IntArray));
    arr->size = size;
    arr->data = malloc(size * sizeof(int));
    for (int i = 0; i < size; i++) {
        arr->data[i] = i; // Initialize the array
    }
    return arr;
}

void printIntArray(IntArray arr) {
    for (int i = 0; i < arr.size; i++) {
        printf("%d ", arr.data[i]);
    }
    printf("\n");
}

int main() {
    IntArray arr = createIntArray(10);
    printIntArray(arr);
    free(arr.data);
    return 0;
}
```

On the right is a call graph window titled "Call Graph". It displays a hierarchy of call graph nodes:

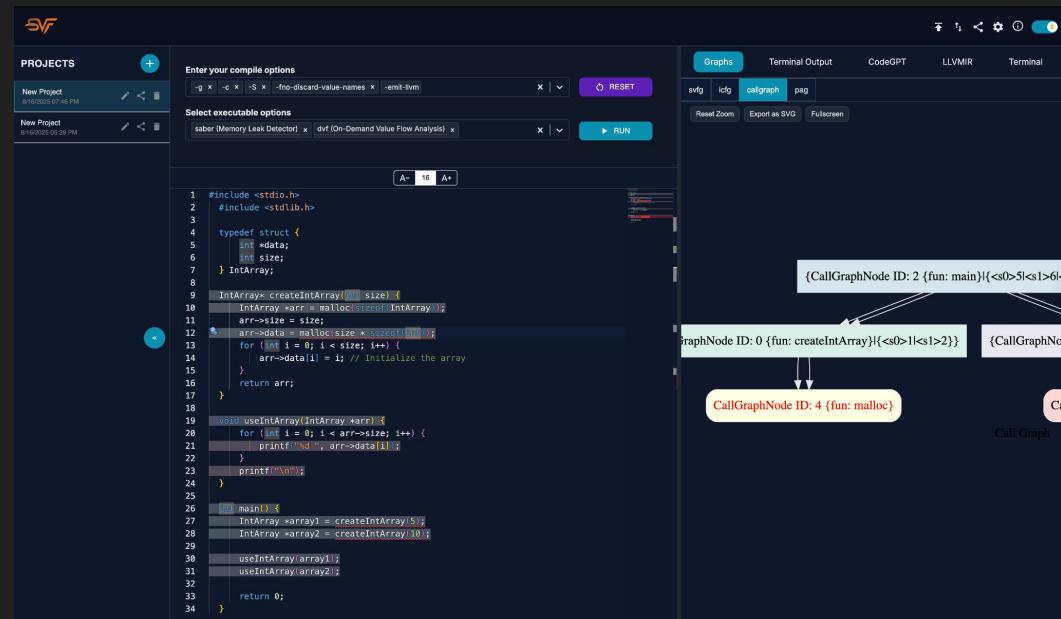
- CallGraphNode ID: 5 {fun: main} (Root Node)
- CallGraphNode ID: 0 {fun: createIntArray}
- CallGraphNode ID: 3 {fun: useIntArray}
- CallGraphNode ID: 2 {fun: malloc}
- CallGraphNode ID: 4 {fun: free}

Arrows indicate the flow of calls from the main function down to the creation and use of the array, and finally to the deallocation of memory.

# 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



The screenshot shows a dark-themed user interface for a developer tool. At the top, there are input fields for 'Enter your compile options' and 'Select executable options'. Below these are tabs for 'Graphs', 'Terminal Output', 'CodeGPT', 'LLVMIR', and 'Terminal'. A code editor displays the following C code:

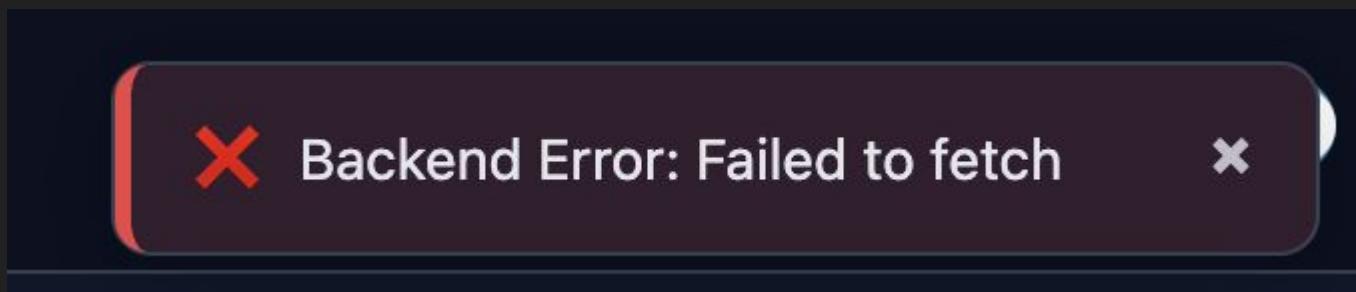
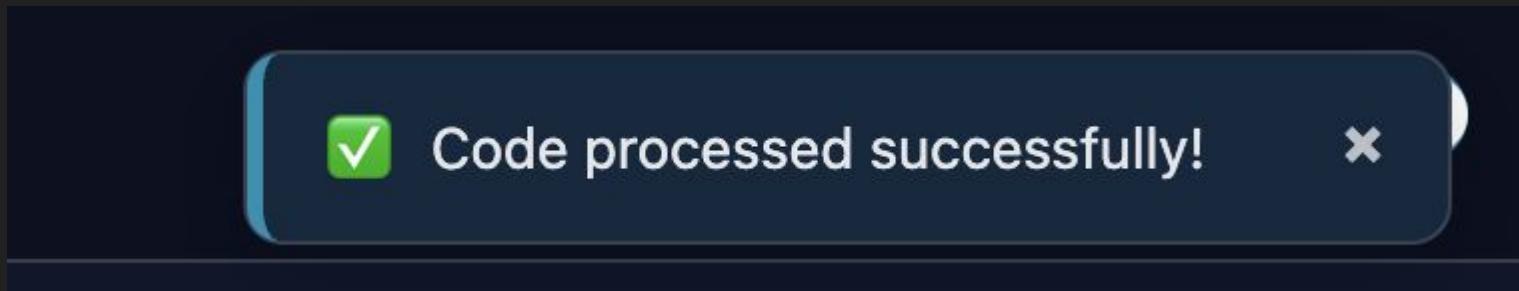
```
#include <stdio.h>
#include <stdlib.h>
typedef struct {
    int *data;
    int size;
} IntArray;
IntArray* createIntArray(int size) {
    IntArray* arr = malloc(sizeof(IntArray));
    arr->size = size;
    arr->data = malloc(size * sizeof(int));
    for (int i = 0; i < size; i++) {
        arr->data[i] = i; // Initialize the array
    }
    return arr;
}
void useIntArray(IntArray arr) {
    for (int i = 0; i < arr->size; i++) {
        printf("%d ", arr->data[i]);
    }
    printf("\n");
}
int main() {
    IntArray* array1 = createIntArray(5);
    IntArray* array2 = createIntArray(10);
    useIntArray(array1);
    useIntArray(array2);
    return 0;
}
```

On the right side, a 'Call Graph' visualization is shown with nodes labeled by their CallGraphNode ID. The graph consists of several nodes connected by arrows, illustrating the flow of function calls in the program.

# UI/UX Improvements - Fullscreen

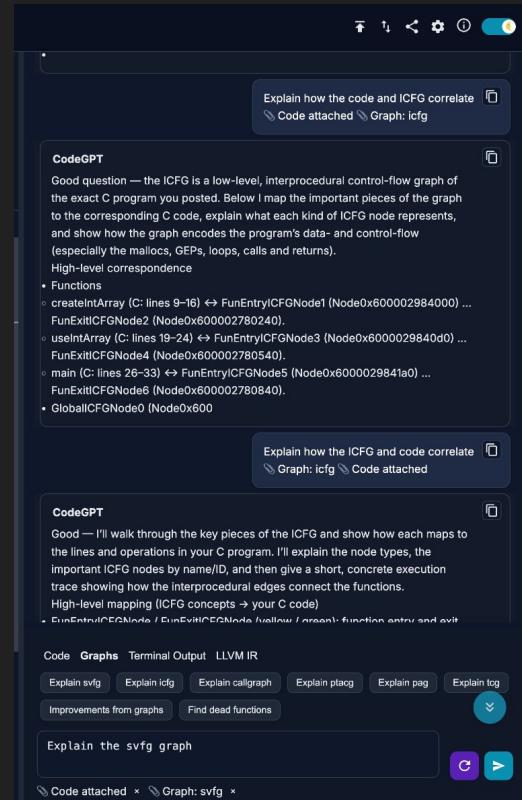


# UI/UX Improvements - Toast



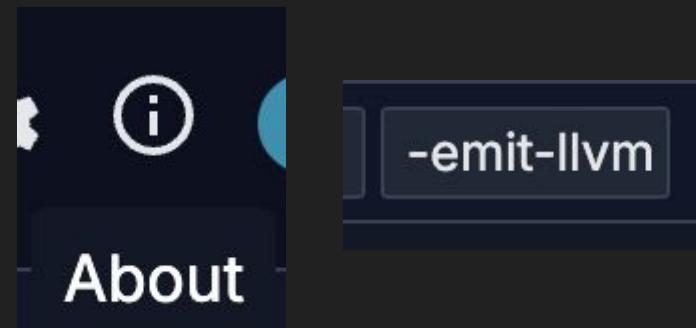
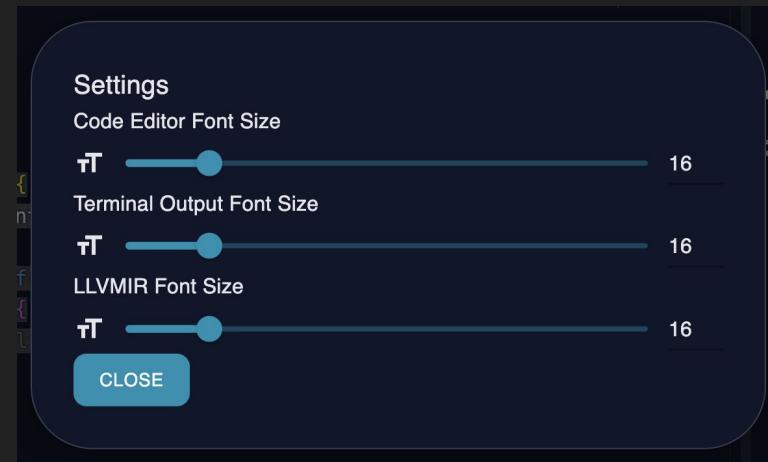
# UI/UX Improvements - CodeGPT

- 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