WebSVF

Static Analysis Leveraging SVF and CodeGPT

UNSW | Software Engineering | T3 2025 | Christian Tolentino and Joshua Wills

Supervisor: Yulei Sui
Assessor: Yuekang Li

Overview

1. Recap
    a. Problem Statement
    b. General Context
    c. Previous Features
2. Joshua's Improvements
    a. C++ Support
    b. Custom Keybindings
    c. Default Code
3. Christian's Improvements
    a. User Onboarding Guide
    b. Terminal Feature Fix/Improvements
    c. Quality of Life Improvements
4. Looking to the Future

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.

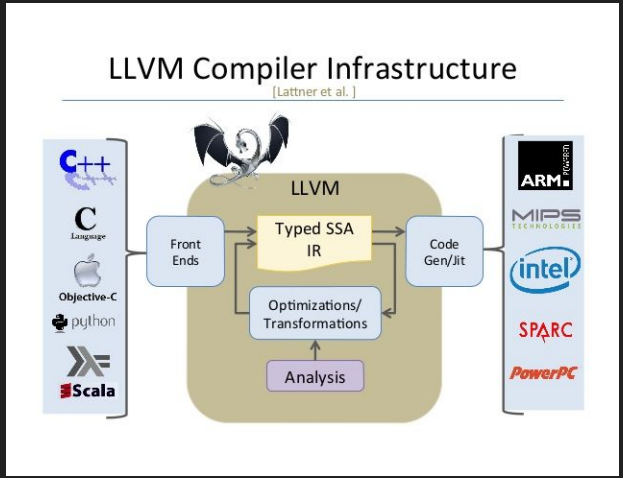
General Context | Static Analysis

- The abstract interpretation of programs to gain insight on the actual computations of a program - Cousot 1970s
- Gain an understanding of a software project's internal structure
- Identify common programming errors
  - Buffer overflows
  - Memory leaks
  - Null pointer dereference
  - etc.

Patrick Cousot, a father of Static Analysis techniques

General Context | LLVM

- Stands for ‘Low Level Virtual Machine’
  - Albeit a somewhat misleading name
  - First developed by Chris Lattner in the early 2000s
- In practice, a compiler framework, which features a low-level and platform-generic intermediate representation (IR)
  - IR is in a Single Static Assignment (SSA) form
- SVF operates on the bitcode of LLVM.
  - Multi language support!



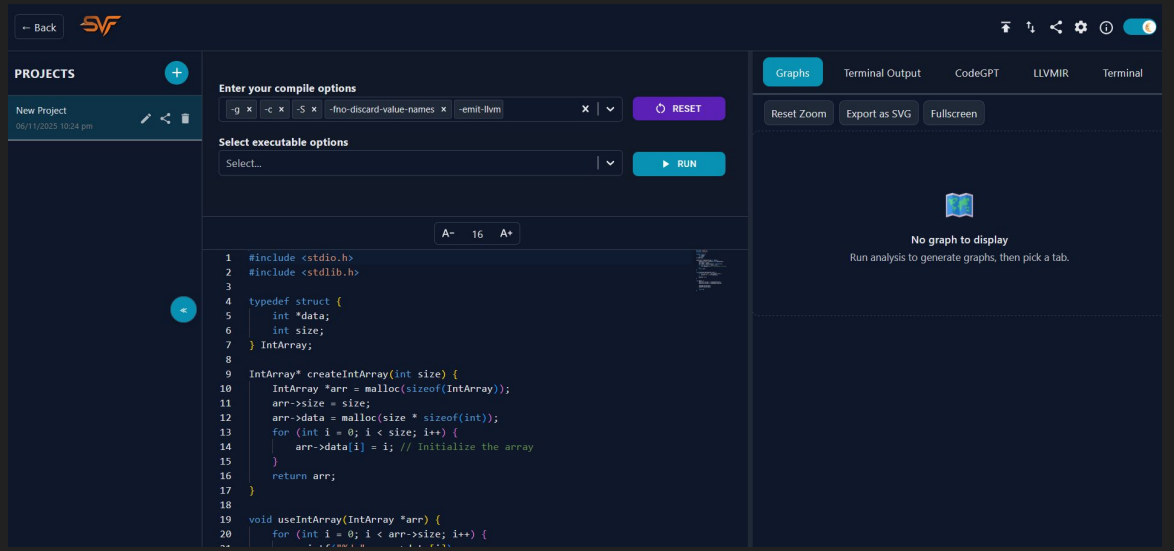
LLVM operating on multiple compiler frontends

Context | SVF

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

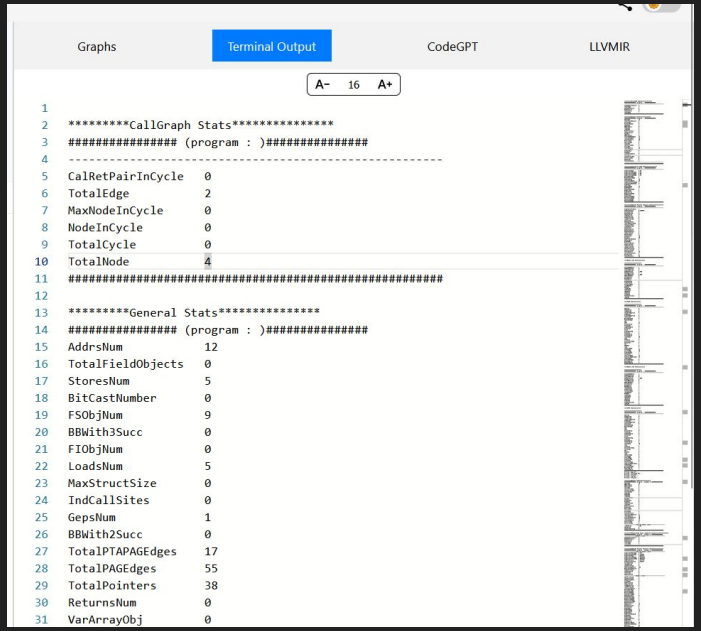
Context | WebSVF

- Web-based application for interfacing with SVF
- 6th major version
- Lowers barrier to entry
- Single paged

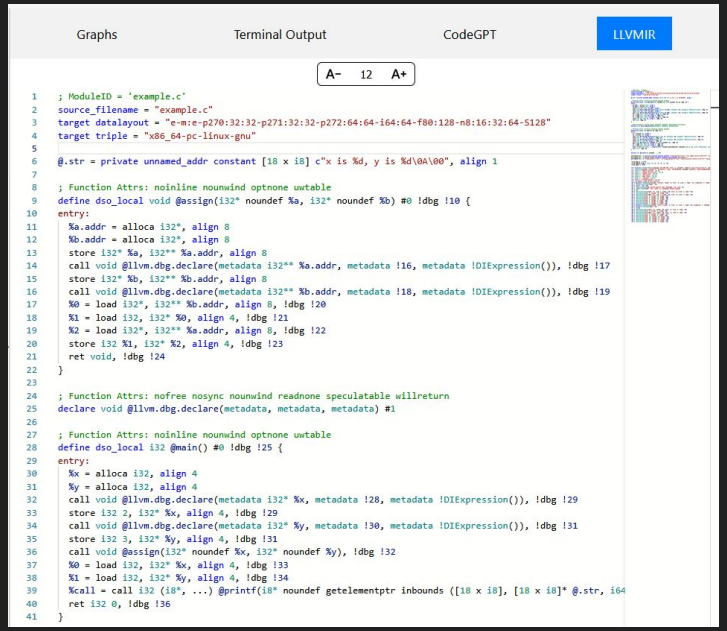


A snapshot of WebSVF in its current state

Context | WebSVF

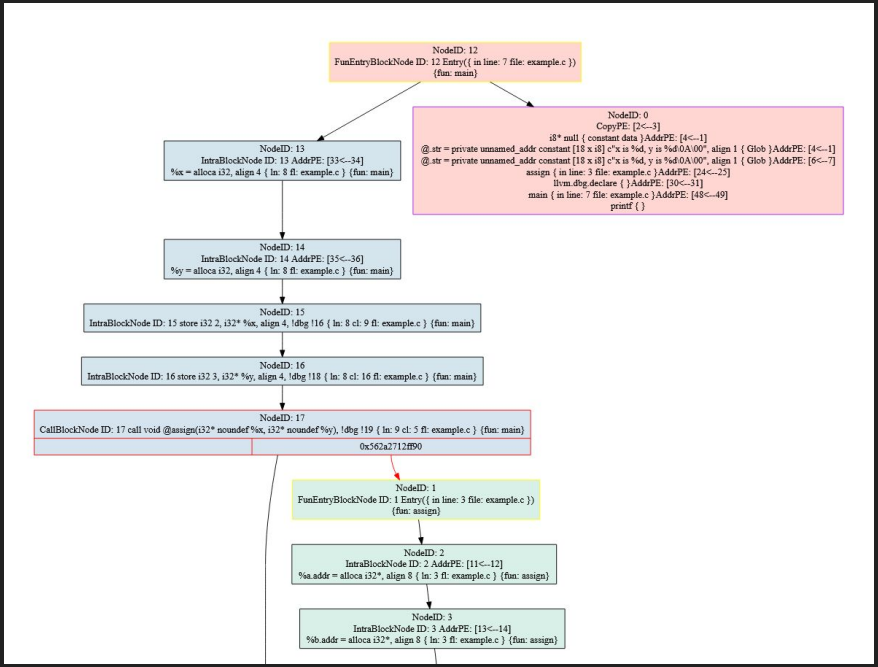


Example of the terminal output

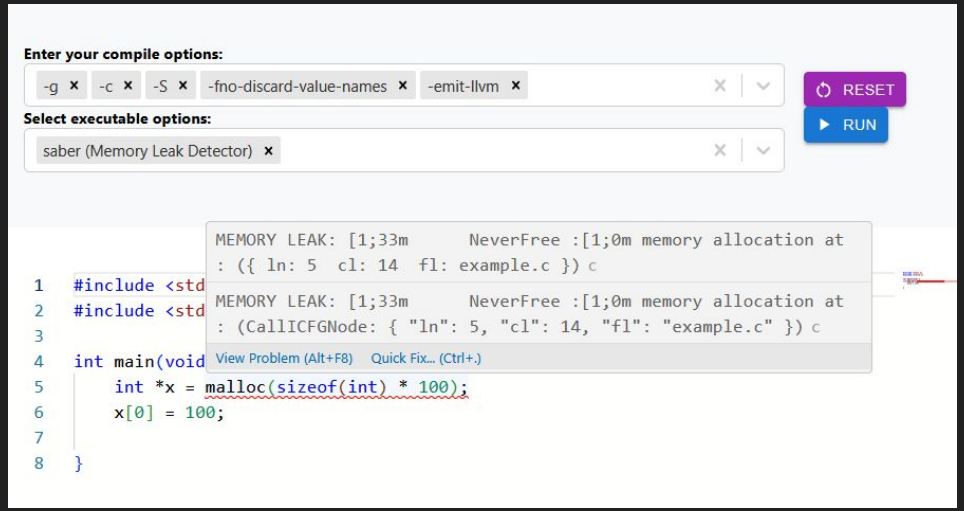


Example of the LLVM IR output

Context | WebSVF



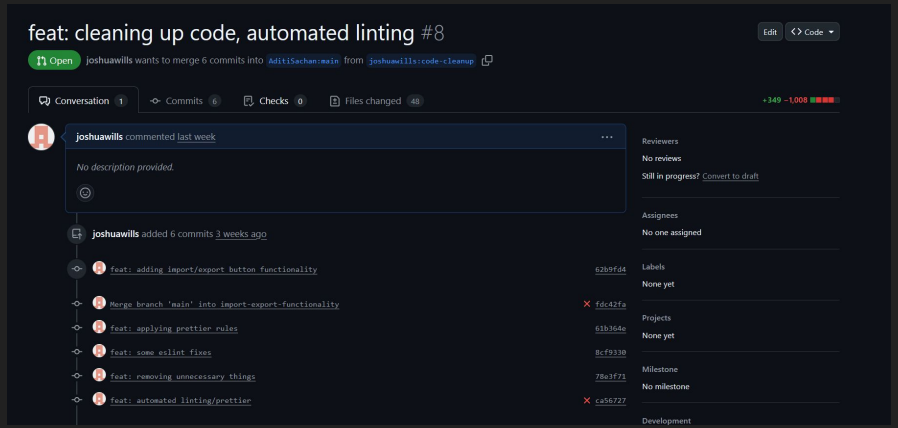
Example of an Interprocedural Control Flow Graph (ICFG)



SABER executable identifying a memory leak

Previous Features

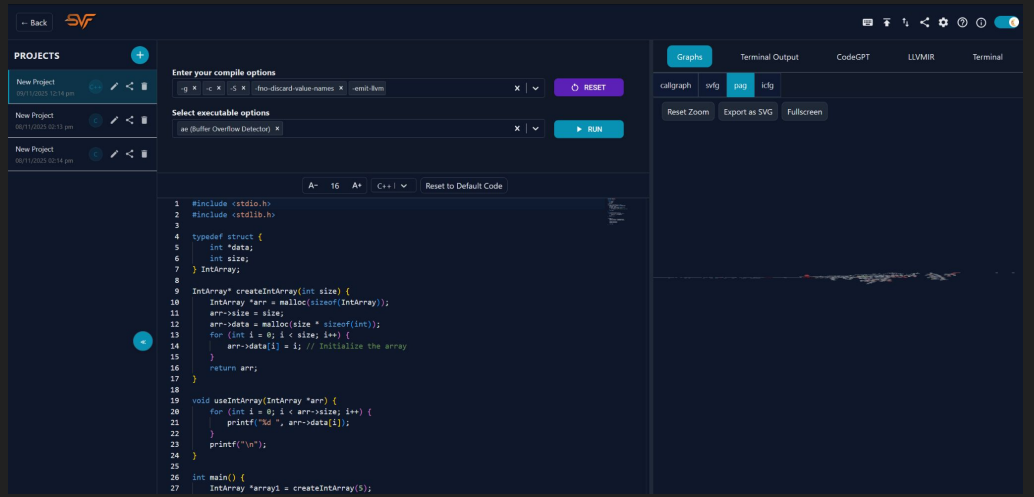
- **Less Obvious Alterations**
  - Software Clean Up
  - Python SVF Backend Conversion
  - Repository amalgamation
  - *All in the name of inviting contribution*



PR adding in cleaned up code

Previous Features

- **Import/Export Functionality**
  - Interface with local filesystem
  - Streamline integration with existing workflow
- **UI Enhancements**
  - Modernise w/ consistency
  - Greater clarity in utility
  - Dark mode enhancements



All the UI improvements in Thesis A and B

Josh’s Improvements

C++ Support

Why it was done:

- Broaden scope of usability
- Increased functionality and support
- Easier onboarding for existing projects

C++ Support

What was done:

- Compilation support
  - Full breadth of LLVM IR, graphs, terminal output
- Languages specified for projects
- Ability to easily switch between them

Custom Keybindings

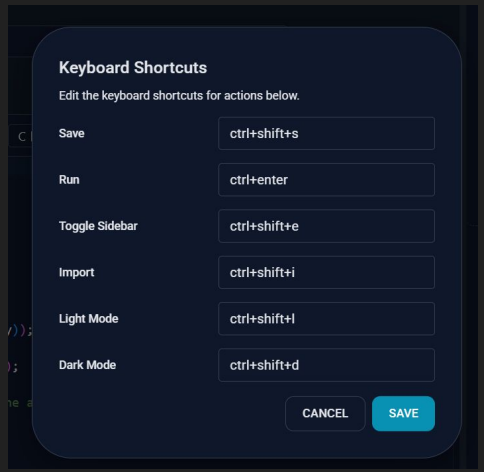
Why it was done:

- Enhance the IDE-like feel of WebSVF
- Give users more freedom
- Enhance productivity and development speed

Custom Keybindings

What was done:

- Default keybindings for:
  - Save file, run SVF, toggle sidebar, import code, light mode, dark mode
- Ability to alter key bindings as one pleases

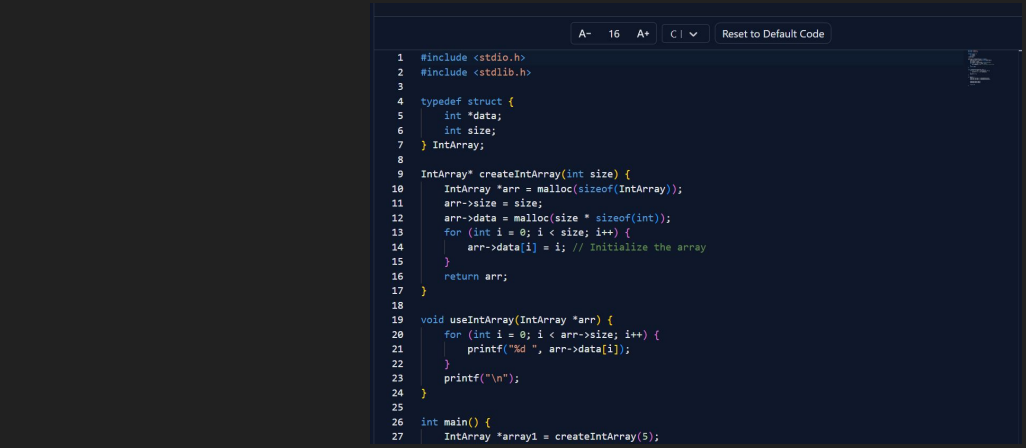


Modal to alter keyboard shortcuts on WebSVF

Default Code Button

Why it was done:

- Easy transition between C and C++ applications
- Ability to “revert” simply to an original state
- See an ideal example of a code snippet to test SVF with

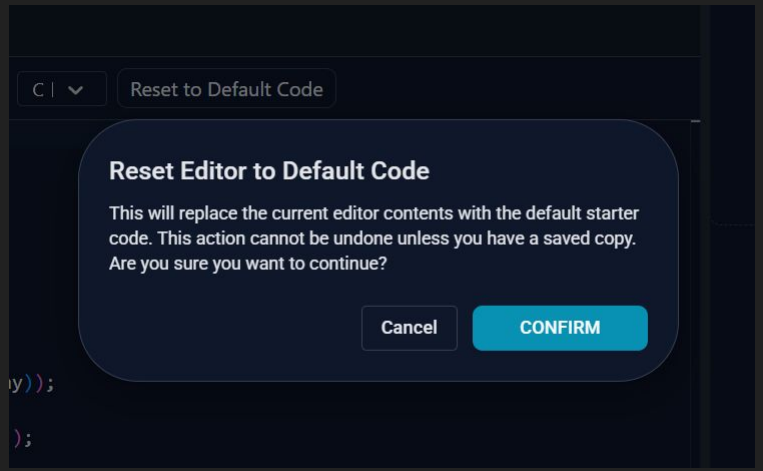


Part of the default C code

Default Code Button

What was done:

- Button added to the text editor component to reset default code
- Confirmation modal to ensure code not accidentally lost
- Language specific defaults



Popup modal when you reset the code

Christians' Improvements

New User Tutorial/Onboarding Guide

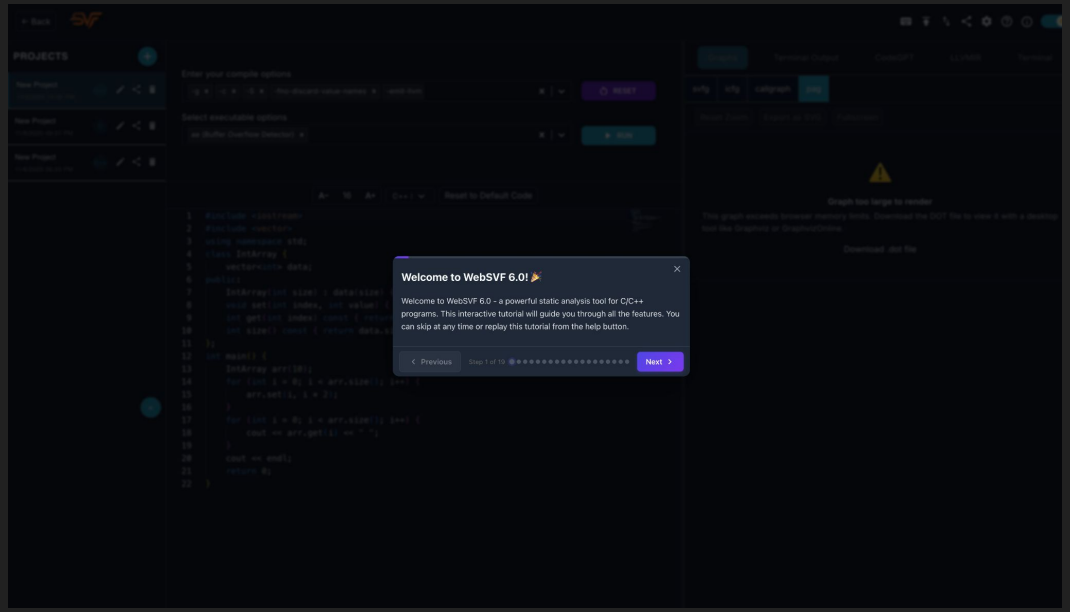
Why it was done:

- Eases the transition for users to try out SVF's capabilities or integrate SVF into their software development lifecycle
  - Explains basic tools and features of SVF and WebSVF and how to use them
  - Simplifies key terminology essential to static analysis
- Adds professionalism to the WebSVF product, enticing users to use it

New User Tutorial/Onboarding Guide

What was done:

- When users first enter WebSVF, they are greeted with a tutorial, highlighting key features of WebSVF and allowing users to try it out themselves
- Tutorial can be replayed through navigation bar



Output Tabs

After running, results appear in different tabs:

- Graph: Visual analysis results
- Terminal Output: Compilation logs
- LLVM IR: Intermediate representation
- CodeGPT: AI assistant

Previous

Step 7 of 19

Next

Graphs | Terminal Output | CodeGPT | LLVMIR | Terminal

Terminal Feature Fix + Improvements

Why it was done:

- Allows for greater use of Unix tools for extended analysis e.g. other analyses such as Valgrind
- Allows for greater integration with user’s development cycle e.g. git for collaboration
- Gives a greater IDE feel, users can code and compile straight on the web

Terminal Feature Fix + Improvements

What was done:

- Integrated terminal per session that comes pre-installed with Unix tools e.g. git and Python
- Users are able to send their code written in WebSVF to this terminal for further command-line usage
- Ability to clear terminal and disconnect from it
- Included as part of tutorial

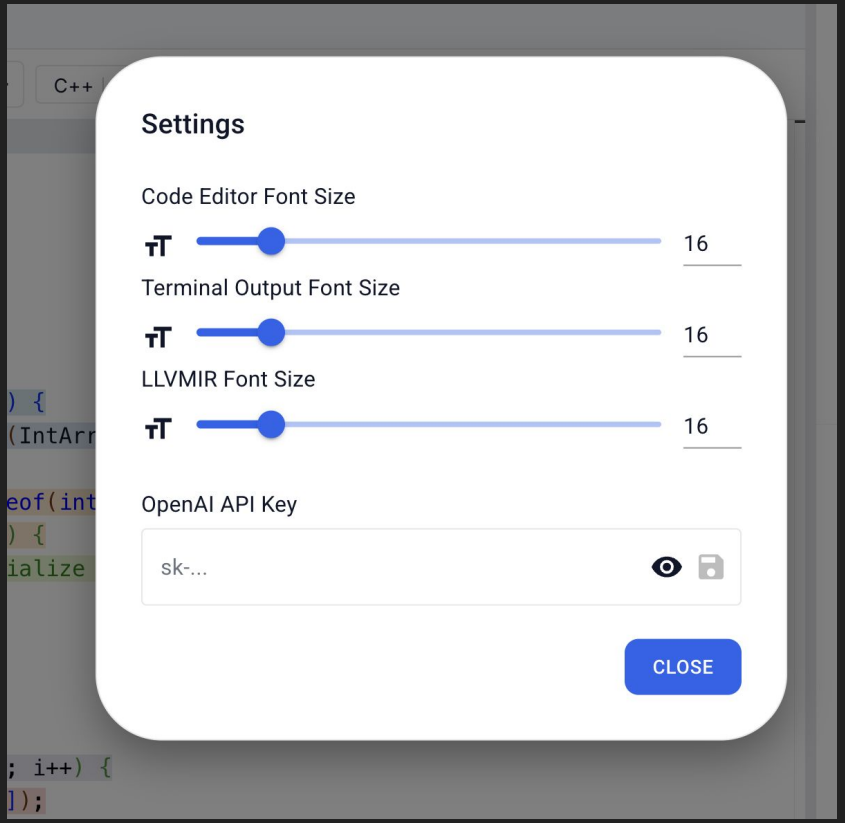
Graphs | Terminal Output | CodeGPT | LLVMIR | Terminal

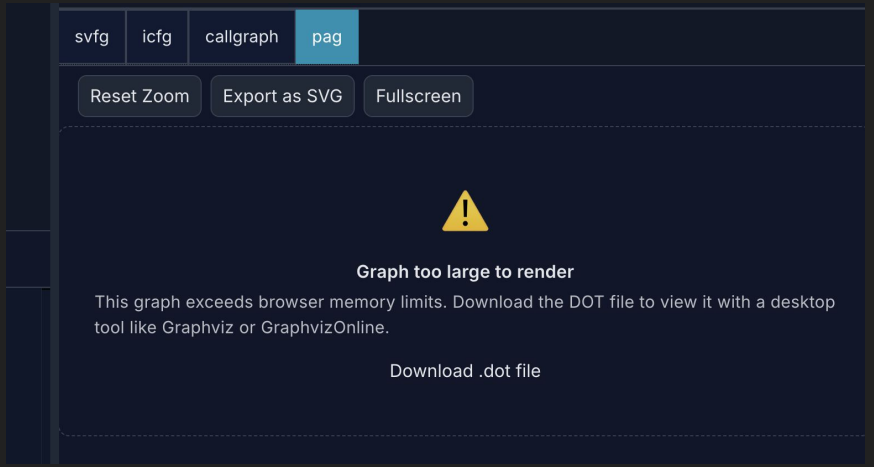
Terminal ● Connected | Clear | Reconnect | Paste Code | Write code to terminal

```
> #include <vector>
> using namespace std;
> class IntArray {
>     vector<int> data;
> public:
>     IntArray(int size) : data(size) {}
>     void set(int index, int value) { data[index] = value; }
>     int get(int index) const { return data[index]; }
>     int size() const { return data.size(); }
> };
> int main() {
>     IntArray arr(10);
>     for (int i = 0; i < arr.size(); i++) {
>         arr.set(i, i * 2);
>     }
>     for (int i = 0; i < arr.size(); i++) {
>         cout << arr.get(i) << " ";
>     }
>     cout << endl;
>     return 0;
> }
> EOF_7n5z0btw
root@48e2590f35e168:/app# ls
__pycache__  app.py  example.c  requirements.txt
root@48e2590f35e168:/app#
```

Miscellaneous Quality of Life Features

- Added new settings for users to enter their own OpenAI API key for extended usage
- Fixed multiple spacing, colouring and other styling issues
- Added labels to icon only buttons
- Error handling case when graphs generated exceed browser memory
- Bug fixes to sessions


C++
Settings
Code Editor Font Size
16
Terminal Output Font Size
16
LLVMIR Font Size
16
OpenAI API Key
sk-...
CLOSE


svfg
icfg
callgraph
pag
Reset Zoom
Export as SVG
Fullscreen
Graph too large to render
This graph exceeds browser memory limits. Download the DOT file to view it with a desktop tool like Graphviz or GraphvizOnline.
Download .dot file

The Future of WebSVF

Potential Improvements

- Ability to use different AI models e.g. Gemini, Grok, Claude
- Support more LLVM-IR compatible languages
- Support Multi-file projects and hybrid language projects
- GitHub/GitLab/Google Drive/OneDrive etc. integration
- Live collaboration e.g. Google Docs
- AI-assisted SVF command-line tool usage
- CI-like automated checks e.g. upload code to WebSVF, runs basic analysis as if it were a pipeline
- Code to Graph Node visual step-through mode
- Vim keybindings support in text-editor