Project Title: Java OO Program Visualizer

Team Members:

- Darian Dean ddean2022@my.fit.edu
- Ashley McKim amckim2022@my.fit.edu
- Simon Gardling sgardling2023@my.fit.edu
- Josh Kalinsky jkalinsky2022@my.fit.edu

Faculty advisor: Dr. David Luginbuhl dluginbuhl@fit.edu

Client: David Luginbuhl, Professor at Florida Institute of Technology

Dates of Meetings:

1. September 26, 2025 @ 4:30 P.M.

2.

Task	Completion %	Darian	Ashley	Simon	Josh
1. Requirement Document	100%	30%	70%		
2. Test Plan Document	75%	5%	85%	5%	5%
3. Design Document	70%	30%		70%	
4. Investigate Tools	100%	25%	25%	25%	25%
5. Collaboration Tools	100%	25%	25%	25%	25%
6. Hello World Demos	100%		Graphviz	Tree-sitter	Rust/WASM/Javascript

3. Progress of current Milestone:
    - Task 1: Requirement Documentation
        i. We drafted a comprehensive requirement document that defined functional, performance, and security requirements of the system. This was a major accomplishment because it provided a clear foundation for later test planning and development. One challenge was ensuring requirements were specific and measurable.
    - Task 2: Test Plan Design
        i. We created a draft Test Plan that mapped requirements to specific test cases. The document established the overall test strategy, including

responsibilities, test requirements, and test case design. The key accomplishment was drafting concrete test cases for memory diagram generation, step-by-step execution, and user-drawn diagram comparison, which serve as the foundation for more detailed system and performance testing in later milestones.

- Task 3: Design Document
    i. The team produced the design document, which established the system architecture and outlined interactions between the backend, frontend, and visualization engine.
- Task 4: Technical Tool Selection
    i. We finalized our selection of technical tools for Java code parsing, backend programming language, and the frontend framework. After comparing options, we chose Tree-sitter for parsing Java code due to its incremental parsing capabilities. Rust for the backend to leverage performance and safety benefits, and Typescript with a modern frontend framework.
    ii. While GraphViz was initially selected as our visualization engine due to its simplicity and strong support for generating structured diagrams from graph descriptions, we have encountered a significant limitation: GraphViz only produces static images. This is problematic for our use case, since one of the core objectives of the Java Object-Oriented Visualizer is to provide dynamic, interactive memory diagrams that change as the code executes line by line.
- Task 5: Select Collaboration Tools
    i. Alongside technical development, we compared and selected collaboration tools for project management. For version control and software development, we chose Github; for documents and presentations, Google Workspace; for communication, Discord; and for scheduling, Google Calendar. These selected tools allow us to align our workflows effectively.
- Task 6: Demos
    i. To confirm our tool selections, we developed “hello world” demonstrations. Using Tree-sitter, we successfully parsed a simple Java snippet and visualized its parse tree. With Graphviz, we created a basic diagram to represent a variable reference in memory. These demos gave us confidence in the feasibility of our approach.

4. Discussion of contribution of each team member to the current Milestone:
    - Darian: Contributed to investigating tools and approaches. Worked on the definitions, assumptions, dependencies, and performance requirements in the Requirement document. Also worked on the overview (purpose and scope), system architecture (components and data flow), and contributed to the modules in the Design document.
    - Ashley: Worked on requirement document and test document, as well as milestone 1 report. Also worked on the “hello world” for Graphviz.

- Simon: Worked on investigating overall tools and approach. I also created a tree-sitter parser demo in order to demonstrate tree-sitter functionality in parsing Java code.
- Josh: Created demo connecting Rust and JavaScript by compiling to WASM. Helped with determining tools and features.

5. Plan for the next Milestone (task matrix)

Task	Darian	Ashley	Simon	Josh
Finalize Visualization engine (Graphviz vs dynamic alternative)	25%	25%	25%	25%
2. Implement, test & demo diagram generation	20%	10%	10%	60%
3. Implement, test & demo the parsing pipeline for a constrained Java subset	10%	10%	70%	10%
4. Implement, test & demo diagram equivalency	20%	50%	15%	15%
5. Implement, test & demo execution trace	60%	20%	10%	10%
6. Update Requirement, Design, and Test documents	25%	25%	25%	25%

6. Discussion of each planned task for the next Milestone
    - Task 1: Graphviz, while mature, may have issues integrating into a WASM environment due to FFI restrictions with Rust (i.e emscripten).
    - Task 2: Generation of dot language syntax from Java code, which is fed into Graphviz.

    - Task 3: Give feedback and error handling for unsupported Java features for the user.
    - Task 4: Compare two diagrams (start with comparing two “dot” language graphs).
    - Task 5: Step-by-step creation of the diagram in order to show the process of creating the diagram.
    - Task 6: Update documents given the experience in implementing them and changes we realized we need to make.
7. Date(s) of meeting(s) with Client during the current milestone:
    - 9/26/2026 (4:30 pm)
8. Client feedback on the current milestone
    - Discussed an alternative to students uploading hand-drawn documents.
        i. Proposed using a technique to draw diagrams directly on our platform.
9. Faculty Advisor feedback on each task for the current Milestone
    - Task 1:
    - Task 2:
    - Task 3:
    - Task 4:
    - Task 5:
    - Task 6:
10. Faculty Advisor Signature: _____ Date: _____