

Project Title: SLAIT – Secure Language Assembly Inspector Tool

Team Members:

- Maria Linkins-Nielsen – mlinkinsniel2022@my.fit.edu
- Michael Bratcher – mbratcher2021@my.fit.edu

Faculty Advisor: Dr. Marius Silaghi – msilaghi@fit.edu

Client: Dr. Marius Silaghi – CSE Professor & Faculty Advisor

Progress of Current Milestone:

Task	Completion%	Michael	Maria	Reasoning/To-Do
Tool Decisions	90%	50%	50%	Still have to select a visualization library for visualizing returned inspection elements
“Hello World” Demos	50%	50%	50%	Backend Demo facing roadblocks, Frontend demo completed
Technical Challenge Resolution	60%	25%	75%	Configuring docker and sandbox permissions - 50% Ensuring flagged code sections translate correctly after compilation - 0% Establishing angular build/test pipeline - 100% Configuring backend compilation for one language - 50%
Frontend-backend communication demo	0%	0%	0%	We didn’t have access to the backend server at the beginning of our milestone, so this was postponed.
Requirements Document	100%	80%	20%	
Design Document	100%	80%	20%	

Test Plan	100%	20%	80%	n/a - Rough Draft Complete
-----------	------	-----	-----	----------------------------

Discussion of Milestone Tasks:

1.

A majority of the tool decisions were made by Dr. Silaghi before the research even began. While these were “suggestions”, we wanted to attempt to utilize the tools he requested before moving onto other options. With the frontend, Angular was fine for the job. Neither of us have much experience with frontend development languages, but after a bit of time and building myself (Michael) an Angular cheat sheet document, progress moved forward and Angular proved to be useful. For the backend, we faced different issues with using the Windows container option with Docker and are currently looking at other options.

2.

There were supposed to be two “Hello world” demos for the milestone, but only one was achieved. The frontend hello world demo consists of a basic webpage that contains two angular components, a component for the file selection and a component for the inspection manager, which is dependent on the file selection. The inspection manager is a widget that lets you create “inspections” which for the meantime, are small tabs that allow the user to select which line, and mark if they'd like the registers and/or flags inspected. It can most certainly use some more fleshing out, such as restricting the number of inspections viewable before they must scroll within a given space, one change I plan to make. GUI wise, its also boring and can use a bit more character to it, but that'll be a much further milestone. For now, the focus was functionality and make the understanding of the purpose of the software easier to outside users. For the backend, we were able to setup and test a docker for the Linux container, executing x86 assembly code with NASM. However, testing the use of the Windows container using MASM proved to be much more difficult.

3.

As mentioned previously, there were issues with configuring the docker container for the option of using NASM, so the first technical issue is halfway done. We were not able to begin researching the capabilities of ensuring that the flag code translates correctly after compilation, so this task will be postponed into the next phase. For our third task, we were able to perform build test methodologies, so that task is complete. Finally, the fourth task of configuring backend compilation for one language is halfway done - also due to the docker issue mentioned before. We will continue to work on these issues in the next phase.

4.

At the beginning of the milestone, this task was unachievable as the server was inaccessible due to an error causing failure on login. Even though the server was brought back online, the

biggest issue with this milestone was the dependency on having both the frontend and the backend in a communicable state. This meant that we had to have beyond a “hello world” demo with increased functionality for both the frontend and the backend, which with all of the deliverable requirements and the limited number of people in the group, meant this milestone was a bit out of scope for us this time around to begin with.

5.

The Requirements Specifications define the functional, interface, and performance requirements for the software. It specifies the features needed for the application to be fully functional, which includes file uploads, viewing files, inspection creation and management, containerized executions, register/flag state captures, data display, and more. The document defines target users, system assumptions, performance criteria, and design necessities. It is written such that the scope for the development of the application is clear, concise, approachable, and specific.

6.

The Design document outlines both the architecture and early designs of SLAIT. It details the specifics of the client-server relationship the application will possess, with an Angular frontend to procure user files and instructions, a backend API that compiles and creates readable instructions for execution, and the containerized environment used for secure isolated executions. The document discusses security focuses, along with ease of use, core data flows, Angular interface components, and the primary user interaction loop the software is intended to perform.

7. The SLAIT Test Plan defines how the Secure Language Assembly Inspector Tool will be tested to ensure it meets its functional, usability, security, and performance requirements. Testing covers code upload and inspection, code execution and state capture, visualization of results, interface behaviors, and system performance. The strategy includes unit, integration, system, and stress testing, validating both normal and unusual cases such as invalid inputs, infinite loops, and heavy resource usage. By confirming SLAIT’s ability to handle errors gracefully, visualize execution states, and remain responsive, the plan ensures the tool will be reliable, secure, and practical for student and faculty use.

Contribution to Milestone Tasks:

Michael:

I was given the responsibility of frontend development, which for this milestone, involved designing the frontend “hello world” demo, which included researching and implementing

Angular, setting up the frontend dev environment, creating the webpage components for the file selector, file viewer, inspector creator tool, and the “test” button demo feature. Furthermore, I also headed and wrote the requirements and the design documentation.

Maria:

I was given the tasks of comparing technical tools for the backend execution, completing initial testing with Docker, configuring Docker, and compilation testing for x86 code snippets, and establishing the backend API. I was able to complete the tool comparison, part of the initial testing, and compilation testing for Docker, but then I hit a roadblock. I plan to finish the Docker configuration and compilation, as well as begin establishing the backend API in the next phase.

1. Plan for the next Milestone (task matrix) or
[skip if this is for Milestone 6]

Task	Michael	Maria
1. Establishing backend access	50%	50%
2. Frontend Visualization of returned information	100%	0%
3. Finalize container environment	0%	100%
4. Enhance Frontend GUI	100%	0%
5. Confirm container compatibility with API and other tools	0%	100%

--	--	--

2. Discussion of each planned task for the next Milestone

- Task 1: Both Michael and Maria will contribute to ensuring reliable backend access. This will involve researching server authentication issues and setting up a consistent development environment that connects frontend and backend services. Achieving this will allow progress on integration tasks that were previously blocked.
- Task 2: Michael will implement the display of execution sample results captured by the backend. This includes designing clear visual representations (e.g., tables, timelines, or highlighted differences) that align with SLAIT's usability goals. The visualization will support comparing multiple runs and emphasize readability for both students and faculty.
- Task 3: Maria will focus on completing Docker container setup and testing for consistent compilation and secure execution of MASM/NASM code. This task ensures SLAIT runs in an isolated environment with proper timeout handling and resource management, critical for security and performance.
- Task 4: Michael will improve the interface by refining the layout, adding visual polish, and implementing constraints (e.g., limiting inspection tabs before scrolling). The goal is to make SLAIT more intuitive and visually appealing, balancing functionality with user experience.
- Task 5: Maria will validate that the finalized container communicates properly with the backend API and integrates smoothly with toolchains chosen for SLAIT. This step will ensure end-to-end workflow consistency, from file upload to inspection creation, execution, and result visualization.

3. Date(s) of meeting(s) with Client during the current milestone: n/a

4. Client feedback on the current milestone

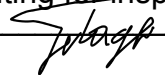
- see Faculty Advisor Feedback below

5. Date(s) of meeting(s) with Faculty Advisor during the current milestone:

- Twice via email

6. Faculty Advisor feedback on each task for the current Milestone

- Task 1: I'd say that the current requirements/Test/Design document
- Task 2: are rather simpler than the customer's request of a tool
- Task 3: to evaluate code snippets by completing them automatically
- Task 4: into full programs using AI or other tools, and instrumenting for inspection.

7. Faculty Advisor Signature:  Date: 09/26/25

8. Evaluation by Faculty Advisor

- Faculty Advisor: detach and return this page to Dr. Chan (HC 209) or email the scores to pkc@cs.fit.edu
- Score (0-10) for each member: circle a score (or circle two adjacent scores for .25 or write down a real number between 0 and 10)

Michael	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
Maria	0	1	2	3	4	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10

- Faculty Advisor Signature: _____ Date: _____