Milestone 1 Progress Evaluation

Project Title: Improved Visualization for Formal Languages

Group Members: Chris Pinto-Font (cpintofont2021@my.fit.edu), Vincent Borrelli (vborrelli2022@my.fit.edu), Andrew Bastien (abastien2021@my.fit.edu), Keegan McNear (kmcnear2022@my.fit.edu)

Faculty Advisor: Dr. Luginbuhl (dluginbuhl@fit.edu)

Client: Dr. Luginbuhl

Milestone #1 Progress:

Task	Completion %	Chris	Vincent	Andrew	Keegan	To do
Compare and select Technical Tools	90%	15%	15%	35%	35%	Team is willing to add additional libraries and logic tools upon them making themselves necessary further into development via things like animations and complex logic
"hello world" demos	100%	20%	20%	30%	30%	None
Resolve Technical Challenges	40%	10%	10%	10%	10%	Learning and refining the animation process for our app is an ongoing endeavor
Compare and select Collaboration Tools	100%	25%	25%	25%	25%	None

Requirement Document	100%	15%	55%	15%	15%	None
Design Document	100%	10%	25%	35%	30%	None
Test Plan	80%	10%	50%	10%	10%	Will likely expand scope of testing following more concrete feature additions though current testing list is reasonably comprehensive for current imagined project scope.

Milestone #1 Discussion (Task Details):

- Compare and select Technical Tools
  - We looked into several tools and languages to see which would serve us best in terms of both internal logic and user interactive experience. This process was a teamwide effort though Keegan and Andrew were the deciding minds in the selection of which tools would best serve the project, thus we reached consensus and utilized python primarily for its ease of use, extensive libraries, and team familiarity. .
- "hello world" demos for technical tools.
  - We developed a simplistic “hello world” styled demo to showcase the graphical output and interactivity of python. This was based on the previous work of group members in previous classes which were focused on python outputs. The group was able to build and test a python program which outputs a graphical popup and even had a usable “button” feature to test user input via the mouse.
- Resolve Technical Challenges
  - The team collectively researched DFA logic implementation into similar programs such as JFLAP, as well as creation of GUI and animation features via python modules. The process is ongoing as the depths of complex implementation will likely produce further complications and necessary additional research. Such planned features as the teaching mode and automatic reduction of created DFAs will require complex internal work and further exploration of the selected animation modules.
- Compare and select Collaboration Tools

    - The group early on devised the collaborative environment for the project via selecting Discord for correspondence, a group google drive folder for documents, and a github repository for code submissions. Keegan lead the creation of the github repository as well as our public facing website.
- Requirement Document, Design Document, Testing document

Milestone #1 Discussion (Team Contribution)

- **Chris:** Chris contributed to technical tool research and selection process through independent research into tools and languages which would best fit our needs. He as well took charge in advisor correspondence and meetings via regular interaction via email. He also assisted in the GUI mockup process. He also helped with the bug testing process for the “hello world” demo.
- **Vincent:** Vincent contributed to technical tool research and selection process through independent research into tools and languages which would best fit our needs. He too made the primary contributions to the testing document, requirements document, and even the progress evaluation document. He assisted in the creation of GUI planning and mock ups via previous foreknowledge of DFA structures. He also helped with the bug testing process for the “hello world” demo.
- **Andrew:** Andrew suggested starting tools/libraries (Python, PyQT, optionally Cython, etc), drafting some data model and execution data model logic in the design document. He led early efforts for the “hello world” testing programs based upon tool selections and planned out and modeled early coding efforts via pseudocode.
- **Keegan:** Keegan began the decomposition of JFLAP and the analysis of its classes as well as conversion of JFLAP objects and classes to python. He also set up the Github and the initial collaboration for the milestone documents, and contributed to the documents, including diagrams and charts. He too was the lead force between setting up the website and formatting it properly.

Plan for Milestone #2:

Task	**Chris**	**Vincent**	**Andrew**	**Keegan**

Have running stable version of the computer application	Bug Fixing/Advisor Role	Bug Fixing/Advisor Role	Co-Lead coder and development head	Co-Lead coder and development head
Have a working basic version of the the DFA graphing process	Bug Fixing/Advisor Role for DFA foreknowledge	Bug Fixing/Advisor Role for DFA foreknowledge	Co-Lead coder and development head	Co-Lead coder and development head
Implement a comprehensive onboard “read me” file for current application features	Bug Testing/Co-Writ er	Bug Testing/Lead writer	Code Side implementation	Code Side implementation
Implement internal logic to check DFA completeness and string validity	Algorithm Planning and DFA knowledge advisor/bug tester	Algorithm Planning and DFA knowledge advisor/bug tester	Co-Lead code side implementor	Co-Lead code side implementor

Milestone #2 Plan Discussion (Task Details):

- Have running stable version of the computer application
    -
- Have a working basic version of the the DFA graphing process
    -
- Implement a comprehensive onboard “read me” file for current application features
    -
- Implement internal logic to check DFA completeness and string validity.
    -

Date(s) of Meeting/Client Feedback:
August 27:

- Focus is on DFAs
- User should be able to create a DFA intuitively (we'd need to nail down what that meant)
- The system should be able to check for completeness (e.g., are there transitions missing?) or correctness (is it truly a DFA)
- Maybe the system should be able to complete given an incomplete DFA (e.g., fill in missing transitions – but how?)
- Eventually it would be nice to format/prettify the DFA graphs (align nodes, improve readability)
- Be able to designate initial and final states and maybe even designate a state as a dead state and let the system do its magic
- Multiple symbols on a single transition between arcs in a way that is natural and readable
- Animate processing of strings on a DFA – step-by-step, play, pause, etc.
- Minimization
- The vision is, given a set of strings that are accepted and not accepted, to let the system create a minimal DFA (in terms of states) that accepts/rejects as specified

September 10:

- Involve in the program is lambda transitions.
- The key things needed is when given strings it would create the DFA, when given strings it would have to determine whether it accepts or rejects and being able to create these states whenever

Date(s) of Meeting/Advisor Feedback:

- ***Same as above***

Faculty Advisor Signature: ______________________ Date: September 29th, 2025