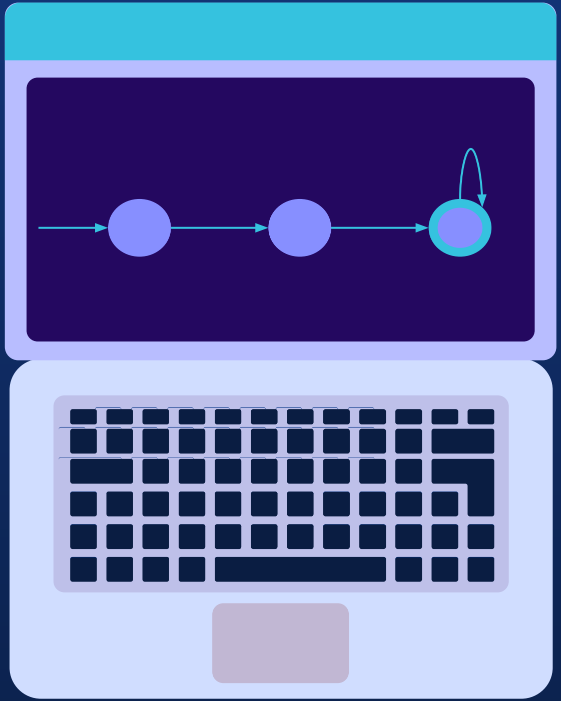
Improved Visualization for Formal Language

https://kmcnear2022.github.io/

Group Members: Chris Pinto-Font, Vincent Borrelli, Andrew Bastien, Keegan McNear



Who is involved?

Faculty Advisor: Dr. Luginbuhl

Serves the role of academic advisor for the project; overseeing product needs and design goals. Providing guidance in the progression of our project while keeping us on track and focused on our goals.

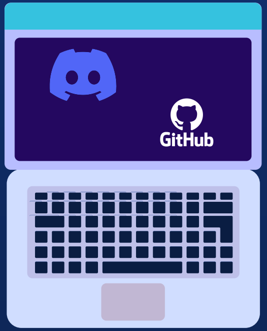
Client: Dr. Luginbuhl

The genesis for project was based on the needs and preferences of Dr. Luginbuhl, specifically his experiences with other graphing software. His close involvement with this project will allow us to quickly address his user needs as he tests our program regularly.

Technical/Collaboration Tools

Main Technical Tool

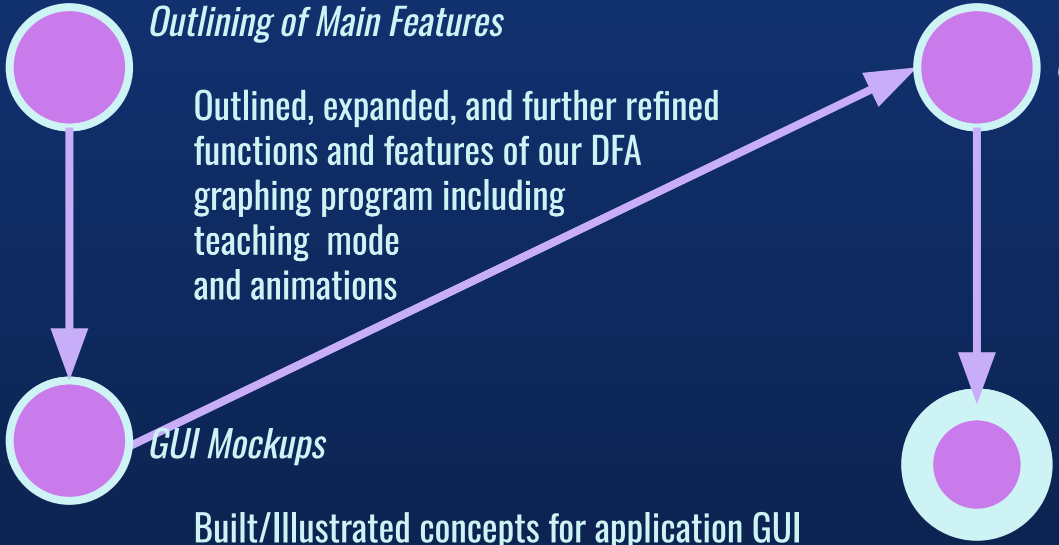
- Visual Studio Code
  - We opted to utilize the application Visual Studio Code as our basis for project development and code construction due to its ease of use and project team familiarity. Said code construction and team contributions are linked together and collected via github.



Collaborative Software

- Github
  - Selected for its ease of use, industry standard status, and overall team familiarly, Github is the basis for code collaboration and project integrity
- Discord
  - We opted to utilize Discord for group correspondence team familiarity and simplicity of use, as well the team's preexisting use of the app means keeping up with team discussion is far easier.

Milestone One Deliverables



Outlining of Main Features

Outlined, expanded, and further refined functions and features of our DFA graphing program including teaching mode and animations

GUI Mockups

Built/Illustrated concepts for application GUI as to help plan our program's graphical output going forward. They too serve to give our client a better idea of what said finished program will likely look like.

UML/Class Diagrams

Illustrated diagrams for conceptualized classes as to plan class interactions and object use within the finished program..

Test Cases

Conceptualized and planned out a list of tests to ensure app reliability, functionality, a operational quickness in terms of animation progression.

Requirements - Examples

Refined Scope requirements:

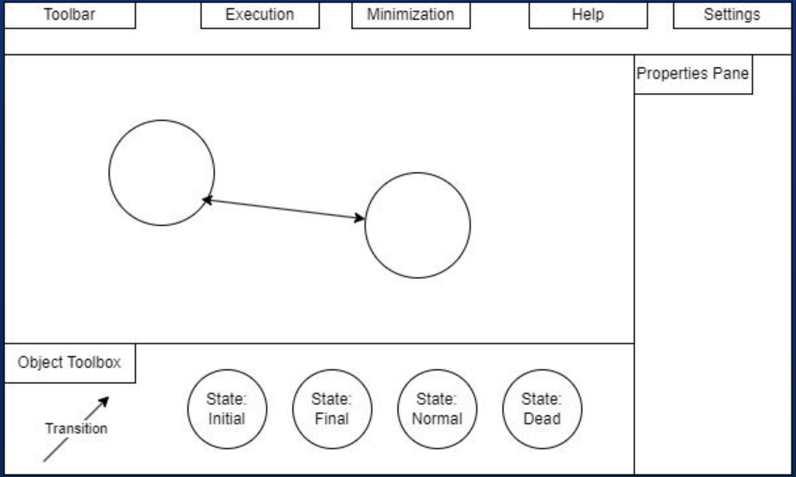
- Allow users to manually or automatically create DFAs.
- Plot DFAs graphically, designate initial/final/dead states, and track execution for a given input string.
- Animate the construction and execution of DFAs step-by-step.
- Automatically complete or minimize DFAs using internal algorithms.
- Provide an intuitive, “toolbox”-style GUI with hover-over labeling.
- Offer multiple-symbol transitions (including lambda), readable animations, and built-in user help.
- Format DFA graphs by aligning nodes to improve readability.
- Allow the user to better understand and learn DFAs through step by step animated construction and a “teaching mode”

Requirements - Examples, cont.

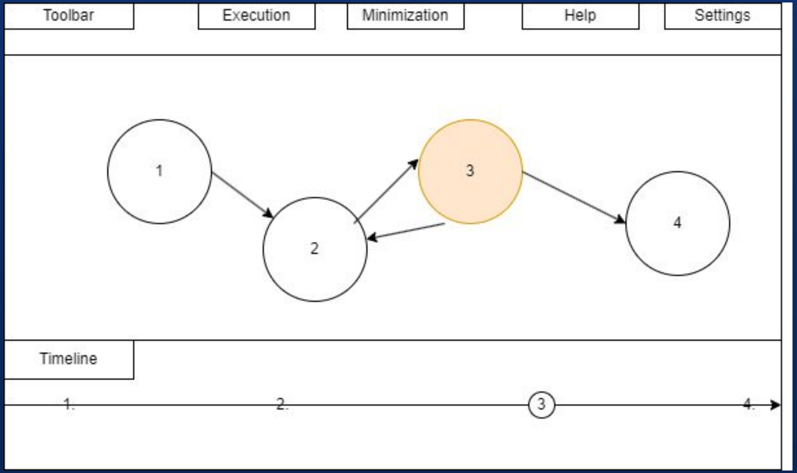
Performance Requirements:

- The system shall load and render DFA diagrams with ≤ 2 seconds delay for small DFAs (<50 states).
- The animations for construction, reduction, and teaching mode will occur at a consistent rate for even large DFAs
- All animations will occur and be able to be rendered quickly through an adherence to algorithmic streamlining to avoid long calculations slowing them down.
- The application should be able to run on pretty much any computer as to make its teaching purpose more widely accessible.

Design Concepts

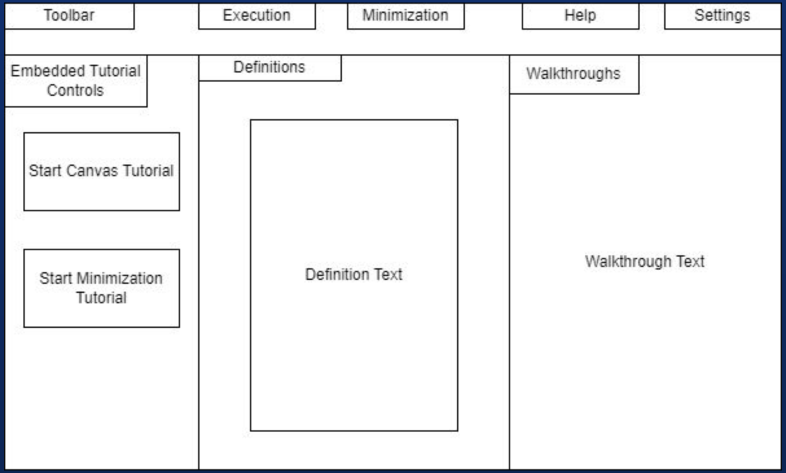


Simplistic DFA created in GUI

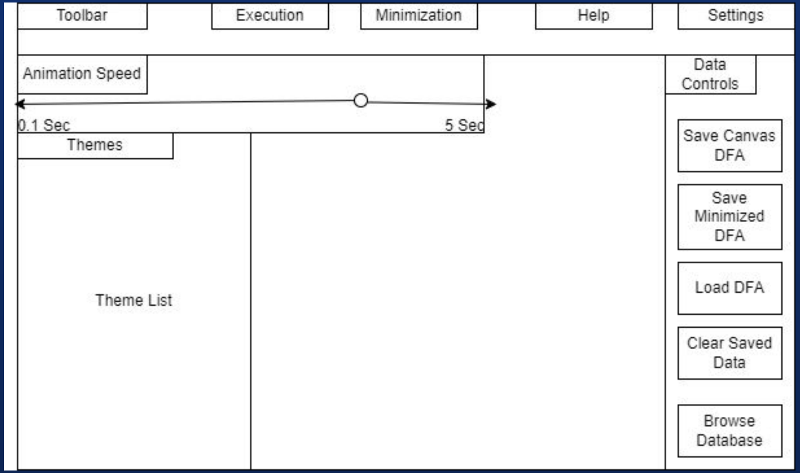


Simplistic DFA created in GUI
With animated timeline

Design Concepts

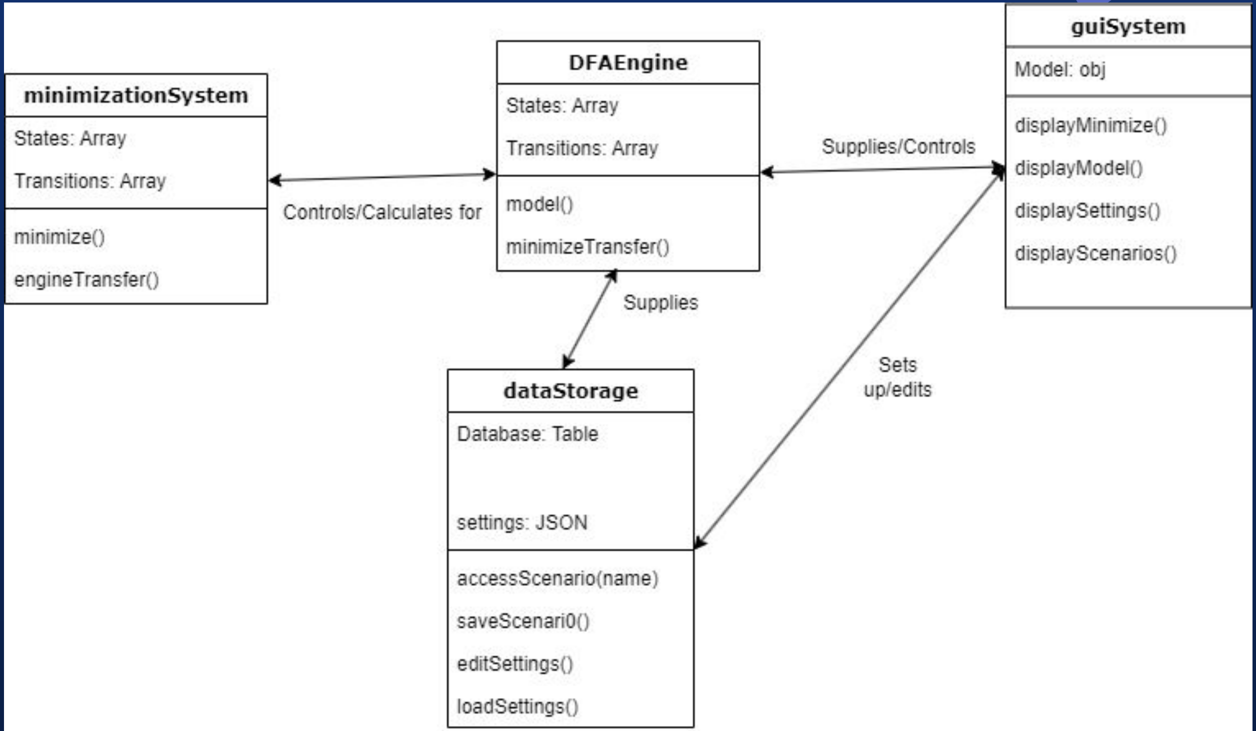


Onboard documentation



Animations settings speed

UML Class Diagram Mockup



minimizationSystem

States: Array

Transitions: Array

minimize()

engineTransfer()

DFAEngine

States: Array

Transitions: Array

model()

minimizeTransfer()

guiSystem

Model: obj

displayMinimize()

displayModel()

displaySettings()

displayScenarios()

dataStorage

Database: Table

settings: JSON

accessScenario(name)

saveScenari0()

editSettings()

loadSettings()

Testing Plans Excerpt

6.1 Functional Requirements

- **[FR-1] Test Case – DFA Creation**
  Input: User creates a DFA graphically in the editor
  Method: Black-box testing of GUI drawing and backend storage
  Usual Output: States and transitions appear correctly and can be saved
  Unusual Output: Invalid state names trigger error messages

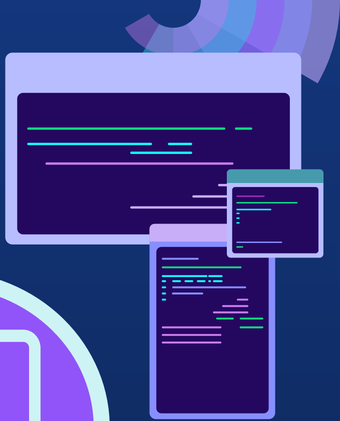
- **[FR-2] Test Case – DFA Execution Animation**
  Input: Enter string to run on a DFA
  Method: Observe animation step-by-step
  Usual Output: Animation highlights current state and transitions correctly
  Unusual Output: If string invalid, system stops and shows error

- **[FR-3] Test Case – State Designation**
  Input: Mark states as initial/final/dead
  Method: GUI toggle test
  Usual Output: States visually update (icons/labels)
  Unusual Output: Conflicting designations prevented by warning

6.2 Performance Requirements

- **[PR-1] Test Case – Large DFA Load**
  Input: Load DFA with 100+ states
  Method: Time loading and rendering
  Usual Output: Diagram appears in ≤3 seconds
  Unusual Output: If slower, system displays “loading” indicator
- **[PR-2] Test Case – Animation Speed**
  Input: Run DFA execution with adjustable speed
  Method: Observe FPS and smoothness
  Usual Output: Animation runs smoothly at 30 FPS or more
  Unusual Output: If lag detected, logs performance metrics

Initial Challenges



New Modules

Project requires researching new tools and python libraries to help render out our required interactive GUI and canvas like graphing process.

Animations

Implementing dynamic and bespoke animations for DFA traversal, reduction, and constructions means building a rendering process into the code.

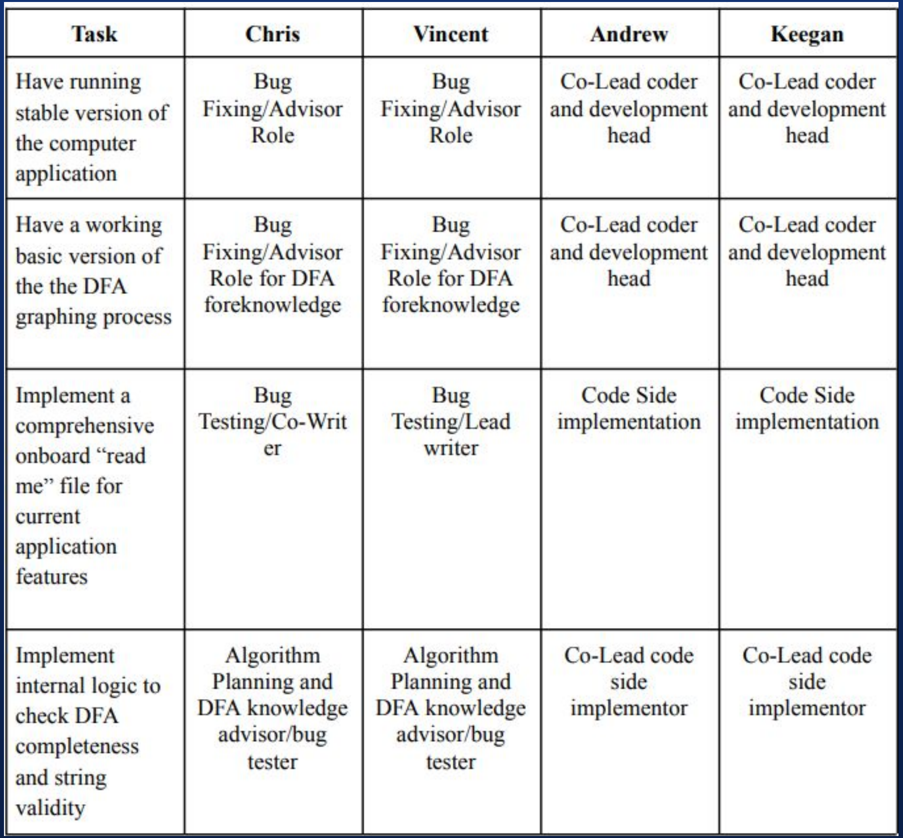
Algorithmic Complexity

The complexity of application features for animation, DFA building, and things like reduction means our application will require complex and reliable algorithms.

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
"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

Milestone 2 Plan



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

Questions?

Visit Our Site

https://kmcnear2022.github.io/