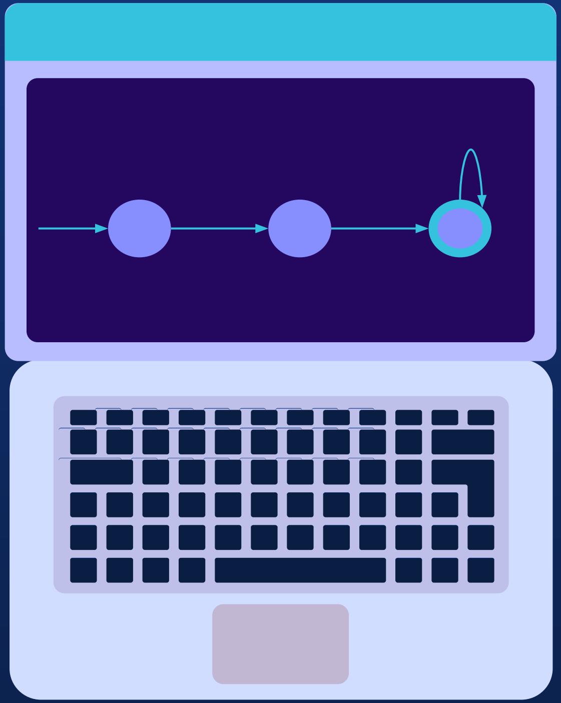
Improved Visualization for Formal Language

https://kmcnear2022.github.io/

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



Who is involved?

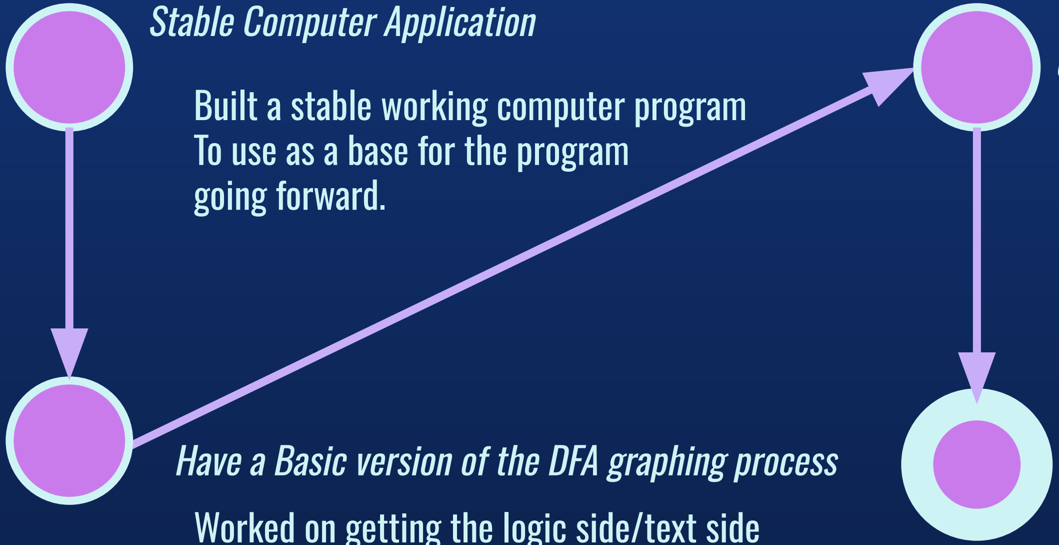
Faculty Advisor: Dr. Luginbuhl

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.

Milestone Two Deliverables



Stable Computer Application

Built a stable working computer program
To use as a base for the program
going forward.

Have a Basic version of the DFA graphing process

Worked on getting the logic side/text side implementation of early logic systems and node concepts working, not yet comprehensive though semi-stable internally.

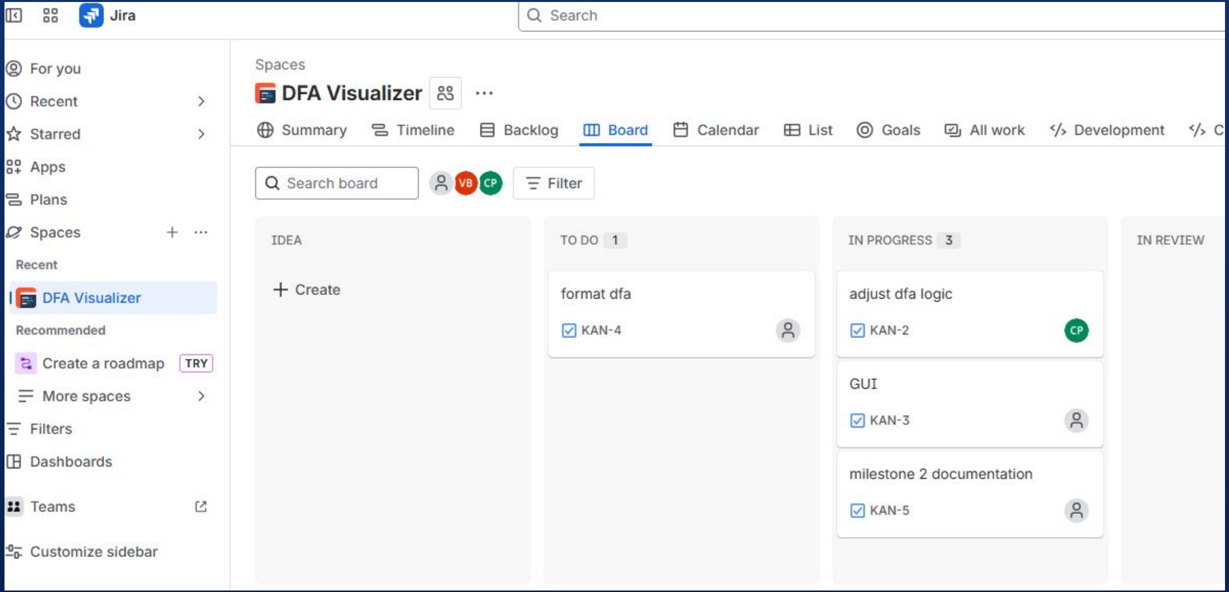
Onboard "Read Me" file

Started the writing process for the onboard "read me" file to be Implemented into the GUI of the program application.

Jira Organizational Project Planner

Elected to build out and create a Jira project environment as to help organize the project going forward.

Jira Work Environment



Spaces

DFA Visualizer

IDEA

+ Create

TO DO 1

format dfa
☑ KAN-4

IN PROGRESS 3

adjust dfa logic
☑ KAN-2 CP

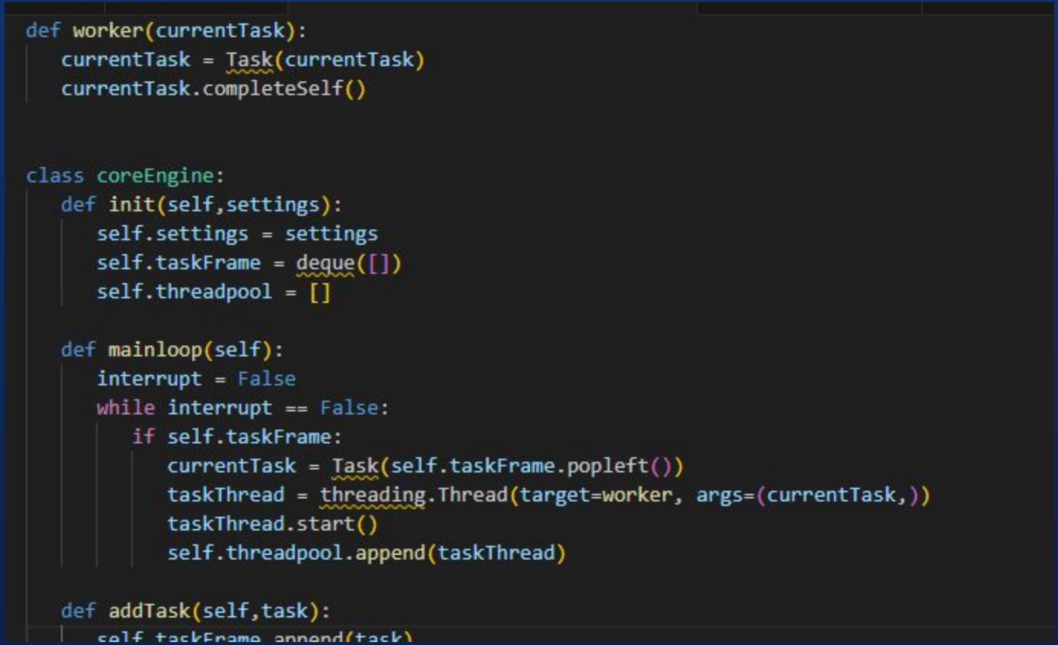
GUI
☑ KAN-3

milestone 2 documentation
☑ KAN-5

IN REVIEW

Early version of the Jira Work environment with set tasks for completion, has been further expanded since and will continue to be used updated as the project progresses.

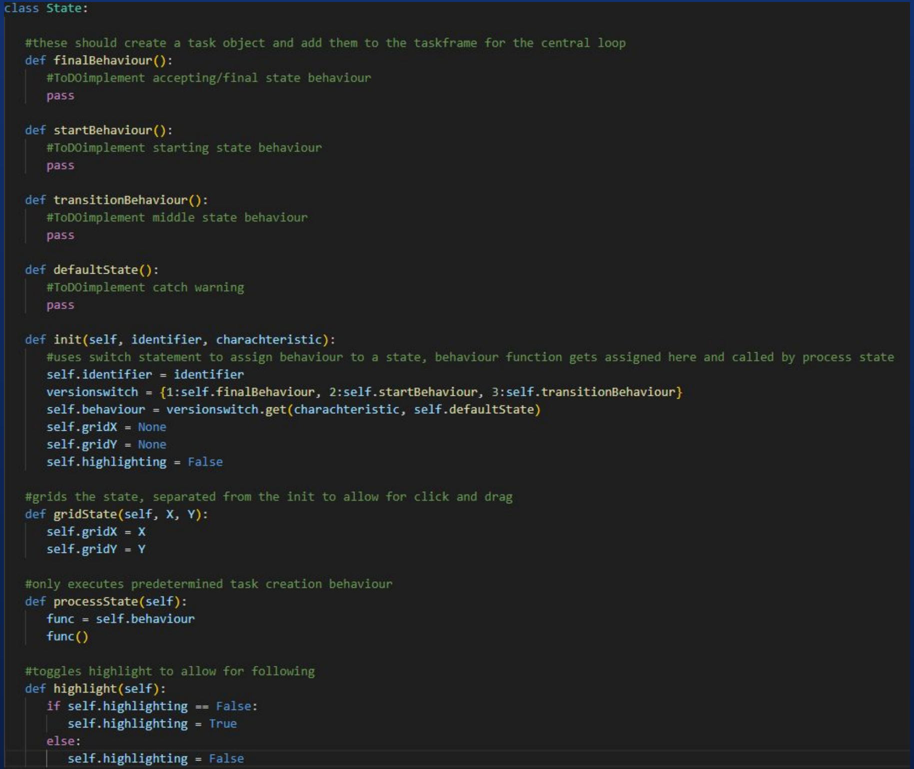
Internal Code Excerpts



```
def worker(currentTask):
    currentTask = Task(currentTask)
    currentTask.completeSelf()


class coreEngine:
    def init(self,settings):
        self.settings = settings
        self.taskFrame = deque([])
        self.threadpool = []

    def mainloop(self):
        interrupt = False
        while interrupt == False:
            if self.taskFrame:
                currentTask = Task(self.taskFrame.popleft())
                taskThread = threading.Thread(target=worker, args=(currentTask,))
                taskThread.start()
                self.threadpool.append(taskThread)

    def addTask(self,task):
        self.taskFrame.append(task)
```

Current Stable Engine Object Code

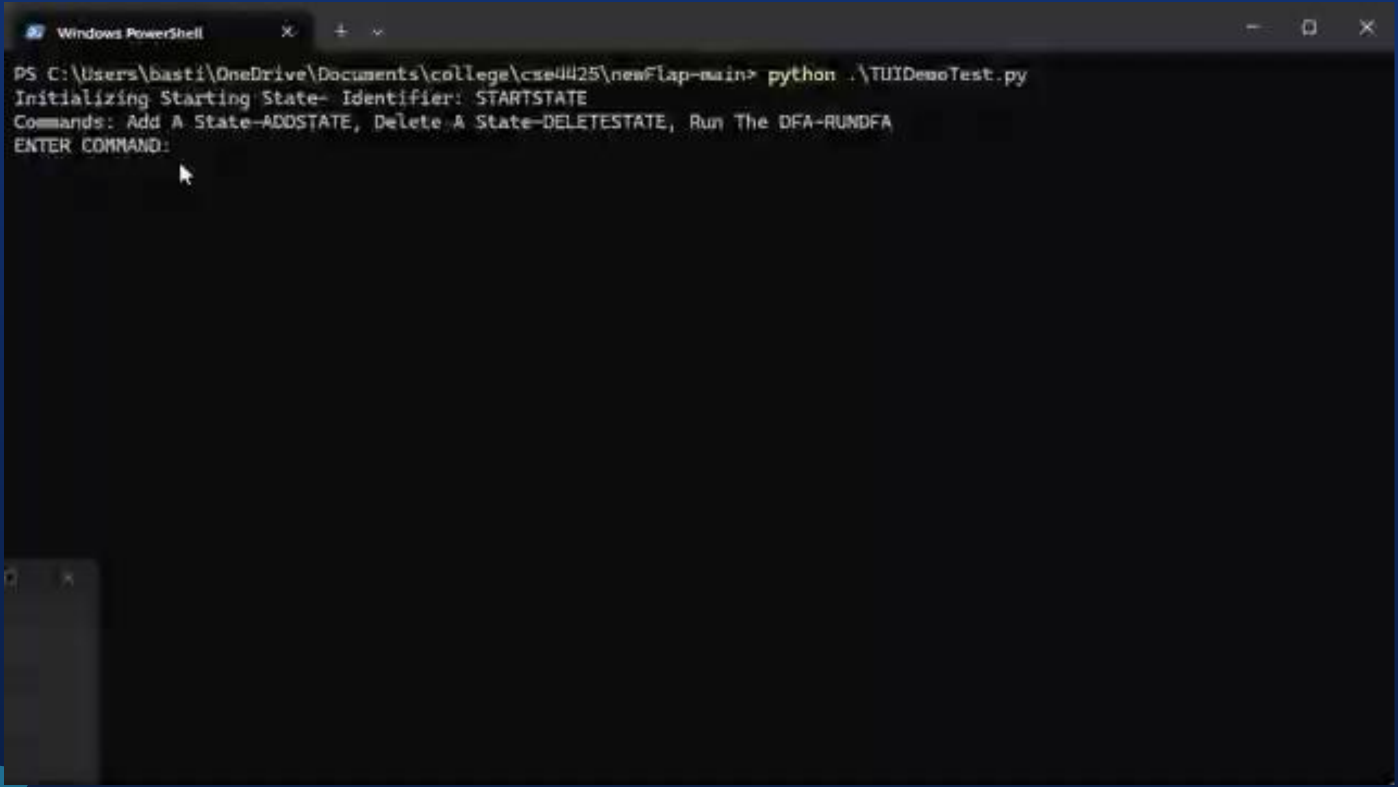
Internal Code Excerpts



```python
class State:

   #these should create a task object and add them to the taskframe for the central loop
   def finalBehaviour():
      #ToDOimplement accepting/final state behaviour
      pass

   def startBehaviour():
      #ToDOimplement starting state behaviour
      pass

   def transitionBehaviour():
      #ToDOimplement middle state behaviour
      pass

   def defaultState():
      #ToDOimplement catch warning
      pass

   def init(self, identifier, charachteristic):
      #uses switch statement to assign behaviour to a state, behaviour function gets assigned here and called by process state
      self.identifier = identifier
      versionswitch = {1:self.finalBehaviour, 2:self.startBehaviour, 3:self.transitionBehaviour}
      self.behaviour = versionswitch.get(charachteristic, self.defaultState)
      self.gridX = None
      self.gridY = None
      self.highlighting = False

   #grids the state, separated from the init to allow for click and drag
   def gridState(self, X, Y):
      self.gridX = X
      self.gridY = Y

   #only executes predetermined task creation behaviour
   def processState(self):
      func = self.behaviour
      func()

   #toggles highlight to allow for following
   def highlight(self):
      if self.highlighting == False:
         self.highlighting = True
      else:
         self.highlighting = False
```

Early State Class

Video Demonstration - Back End Text Based Version

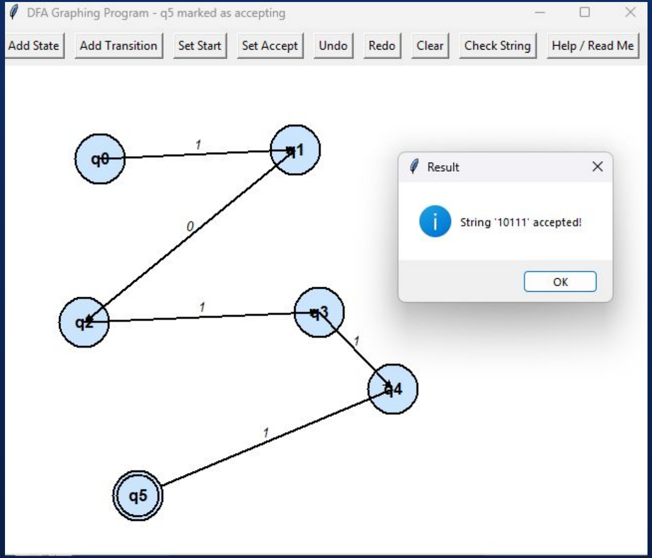


```
PS C:\Users\basti\OneDrive\Documents\college\cse4425\newFlap-main> python .\TUIDemoTest.py
Initializing Starting State- Identifier: STARTSTATE
Commands: Add A State-ADDSTATE, Delete A State-DELETESTATE, Run The DFA-RUNDFA
ENTER COMMAND:
```

Four State DFA

Video Demonstration - Front End GUI Aspect (WIP)

Uses an implementation of Tkinter and buttons to graph and connect DFAs



DFA Graphing Program - q5 marked as accepting

Result

String '10111' accepted!

Continued Challenges

New Modules

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

Connecting the internal workings to the GUI

Building out and connecting the internal DFA logic that exists in a more text based form has posed a challenge this milestone

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 2 Progress

Task	Completion %	Chris	Vincent	Andrew	Keegan	To do
Have running stable version of the computer application	100%	20%	20%	30%	30%	Further expand application functionalities in working program
Have a working basic version of the the DFA graphing process	70%	15%	15%	20%	20%	Need to produce a better, more visual form of the graphing process beyond its current text based internal state.
Implement a comprehensive onboard “read me” file for current application features	70%	20%	20%	15%	15%	Need to further expand onboard “read me” file as things are better fleshed out and expanded.
Implement internal logic to check DFA completeness and string validity	80%	15%`	15%	25%	25%	Further refine internal logic and checking systems

Milestone 3 Plan

Task	Chris	Vincent	Andrew	Keegan
Implement interactive canvas space for graphing	Bug Fixer/Code Contributor and designer	Bug Fixer/Code Contributor and designer	Co-Lead coder and development head	Co-Lead coder and development head
Implement basic animations in graphing space	Bug Fixer/Code Contributor and researcher	Bug Fixer/Code Contributor and researcher	Co-Lead coder and development head	Co-Lead coder and development head
Tie text based program version to visual version	Bug Fixer/Code Contributor and researcher	Bug Fixer/Code Contributor and researcher	Co-Lead coder and development head	Co-Lead coder and development head
Implement basic DFA reduction functionality	Logic writer, DFA logic consultant, and bug tester.	Logic writer, DFA logic consultant, and bug tester.	Co-Lead code side implementor	Co-Lead code side implementor
Updated and Accessible Program Side “Read Me” File	Co-Writer	Co-Writer	Code Side implementation	Code Side implementation

Questions?

Visit Our Site

https://kmcnear2022.github.io/