

# **Finite Automata Playground**

## **A PROJECT REPORT**

*Submitted by*

**BL.EN.U4AIE20031- Kunisetty Jaswanth**

**BL.EN.U4AIE20051 - Pranav Ramachandrula**

**BL.EN.U4AIE20055 - S Sruthi**

*for the course*

**19AIE301 FORMAL LANGUAGE AND AUTOMATA**

*Guided and Evaluated by*

**Dr. Supriya M**

***Vice-Chairperson,***

***Asst. Prof,***

***Dept, of CSE***



**AMRITA SCHOOL OF ENGINEERING, BANGALORE**

**AMRITA VISHWA VIDYAPEETHAM  
BANGALORE - 560035**

**DECEMBER 2022**

# Problem Statement - 1

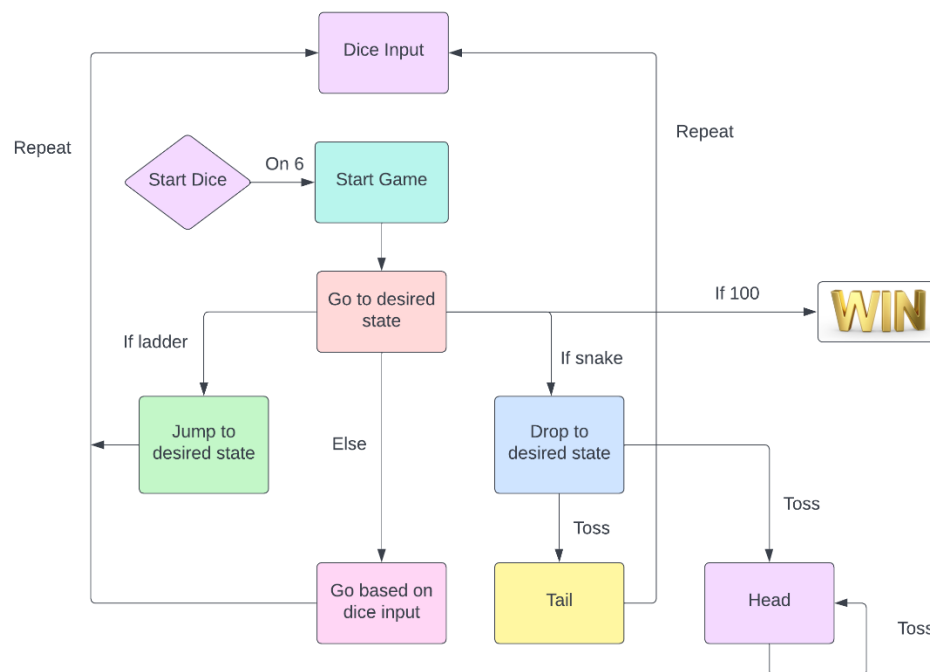
## Using JFLAP:

Creating a model for snake and ladders using JFLAP which defines the person who reaches 100 states with minimum transitions as winner.

## Snake and Ladders using JFLAP

**JFLAP:** JFLAP is a software for experimenting with formal languages topics such as finite automata, pushdown automata, mealy machine, etc. It allows users to create and simulate structures while experimenting with proofs.

## Logic Behind the Game:

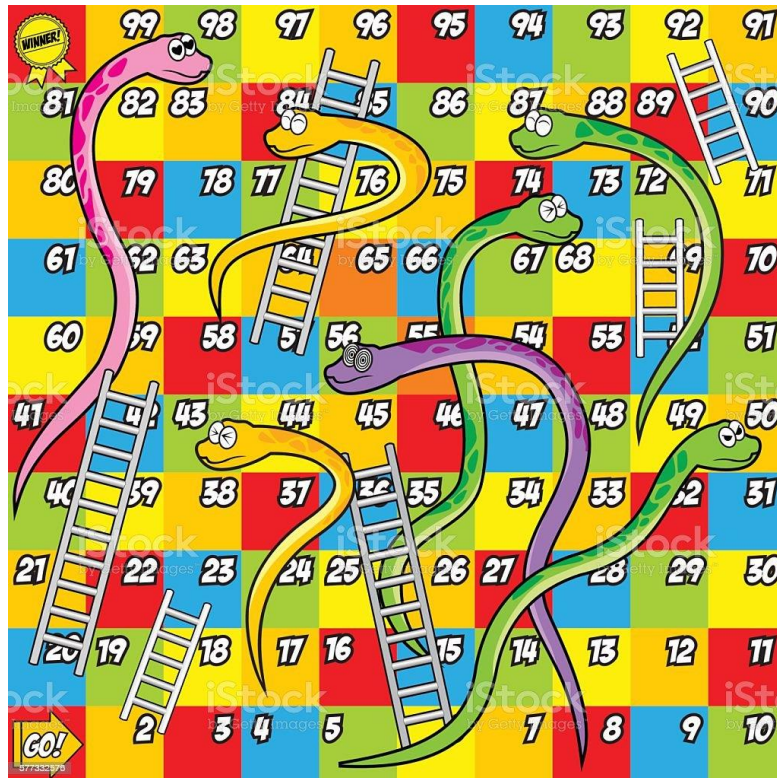


## Implementation:

The game Snake and Ladders is a finite state game which ends when the player reaches 100. To make the game interesting there are snakes and ladders in between which are completely luck factors. Unlike the traditional game there is an addition to the implementation of ours, i.e., when the player encounters a snake, he should drop to the assigned number and play the toss. He can't roll the die until the toss outcome is tail. If he gets a tail, he can freely continue to play. The addition of toss is like a task or penalty for the player which is also a luck-based outcome. Each die throw and each toss are considered as chances and the winner is who can reach 100 with minimum chances.

### Reference Snake and Ladder Board:

The number of checks on board remains same in all traditional Snake and Ladders boards but arrangement of snakes and ladders may differ. So, to fix to a layout a reference board has been chosen.



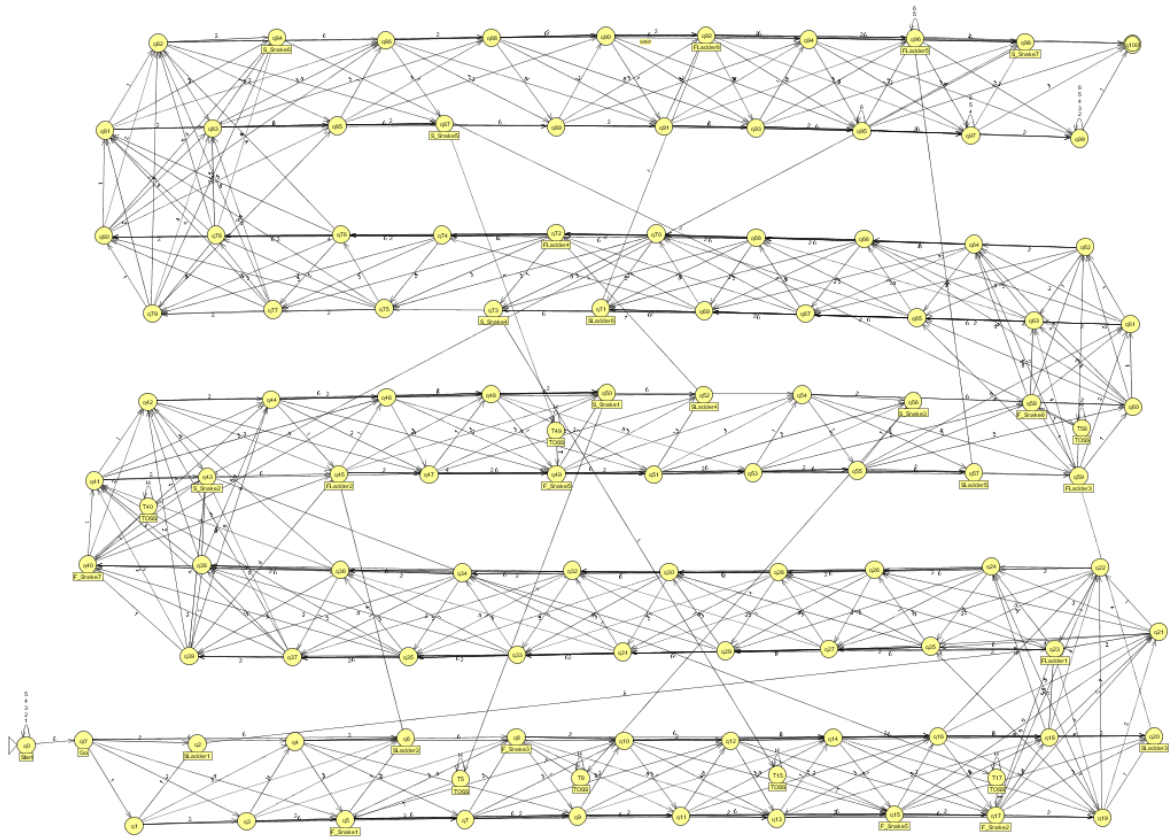
### Implementation using JFLAP:

Using the same logic, a Non-Deterministic Finite Automata has been designed in JFLAP.

1. **Start State:** Entire game depends on outcome of die. So the player must get a 6 to enter the "Go state" of the game. Until they get a 6 the player keeps rolling the die
2. **Go State:** This state basically represents the player being outside the board and ready to start the game. It can also be termed as 0 state.
3. **SLadder States:** These states are starting states for ladder which has only one  $\lambda$  transition to FLadder states.
4. **FLadder States:** For each SLadder State there will be a FLadder states, which transition works as a ladder.
5. **S\_Snake States:** These states function as a snake's head and it has only one  $\lambda$  transition defined to Toss states.
6. **Toss States:** These states can be termed as penalty states in which player has to get Tail while tossing the coin to proceed into game further.
7. **F\_Snake States:** After passing the Toss states the player lands up in these states to continue in the game further.
8. **Usual States:** States other than above labels have all the possible transitions defined for all outcomes on the die.

9. **Final State:** Last state that defines the win status of the game.

### Automata Design:



Above figure shows the automata design in JFLAP.

## RESULTS

Some of the accepting configurations in JFLAP are:

Input	Result
626666666653	Accept
66664	Accept
66662HHT6665665	Accept
665HHT666666666653	Accept

We can decide the winner based on the player having minimal number of turns took to reach 100. For example, in the above figure let's consider each input as one player. As player 2 took minimum turns and the inputs are accepted by automata. That signifies the player reached the final state 100. So, player 2 is the winner.

## Problem Statement - 2

### Using Pyformlang:

Developing a UI tool using Streamlit to visualize the different automata namely Deterministic Finite Automata, Non-Deterministic Finite Automata and Regular Expression.

## Finite Automata Playground

### Pyformlang:

Pyformlang is a Python library that is used to create and manipulate formal languages. Formal languages are sets of symbols that are used to represent different kinds of data. They can be used to describe the structure of a document, the syntax of a programming language, or the rules of a computer system.

Pyformlang is designed to make it easy for developers to work with formal languages. It provides a set of tools that can be used to create, modify, and analyze formal languages. Some of the features of Pyformlang include:

- A comprehensive set of data structures that can be used to represent different types of formal languages, including regular expressions, context-free grammars, and finite automata.
- An API that allows developers to easily manipulate these data structures and perform various operations on them, such as union, intersection, and complement.
- A set of algorithms that can be used to analyze and manipulate formal languages, such as minimization, determinization, and conversion between different representations.

Currently the version of Pyfromlang is 1.0.1. It started its release version as 0.0.1 in Jan 2019 and it is under development to add furthermore packages for visualization, etc.

### Streamlit:

Streamlit is a open source Python library that is helpful to create custom web apps for data science and machine learning. Streamlit is a powerful open-source framework that allows developers to build interactive web applications quickly and easily. With Streamlit, you can create dashboards, data visualizations, and other types of applications with just a few lines of code.

One of the key features of Streamlit is its simplicity. It uses a Python API to allow developers to build applications without having to worry about front-end development or web design. This means that you can focus on the core functionality of your application and let Streamlit handle the rest.

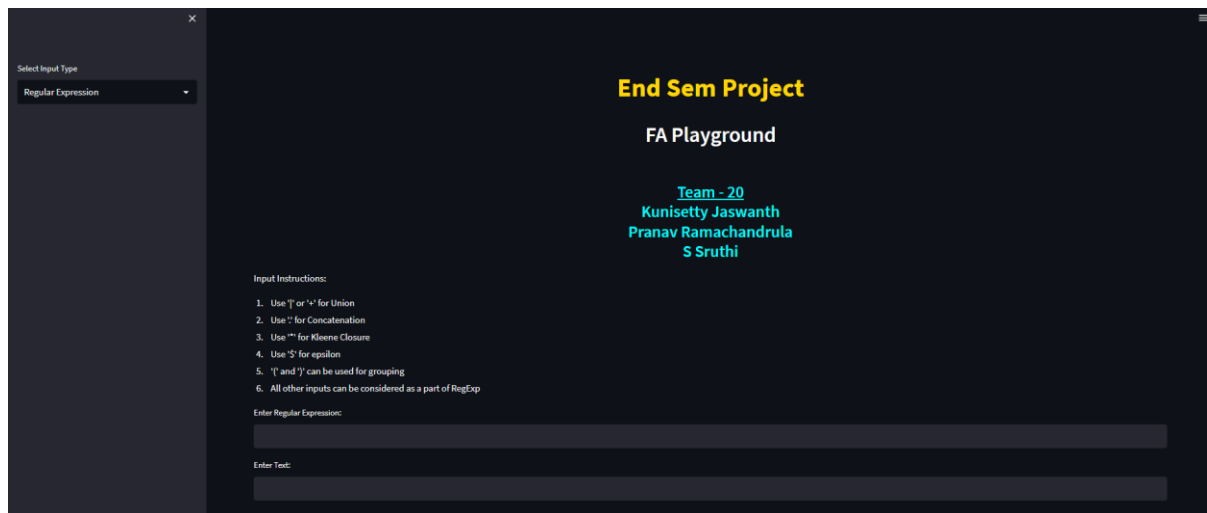
Another advantage of Streamlit is its flexibility. It can be used to build a wide range of applications, including data visualization tools, machine learning models, and even simple games. This means that you can use Streamlit to build whatever type of application you need, without having to worry about the underlying technology. Streamlit has a strong community

of developers and users. This means that you can get help and support whenever you need it, and you can also contribute to the development of the framework itself.

Overall, Streamlit is a powerful tool that allows developers to build interactive web applications quickly and easily. Its simplicity, flexibility, and strong community make it a great choice for anyone looking to build interactive applications with Python.

## RESULTS

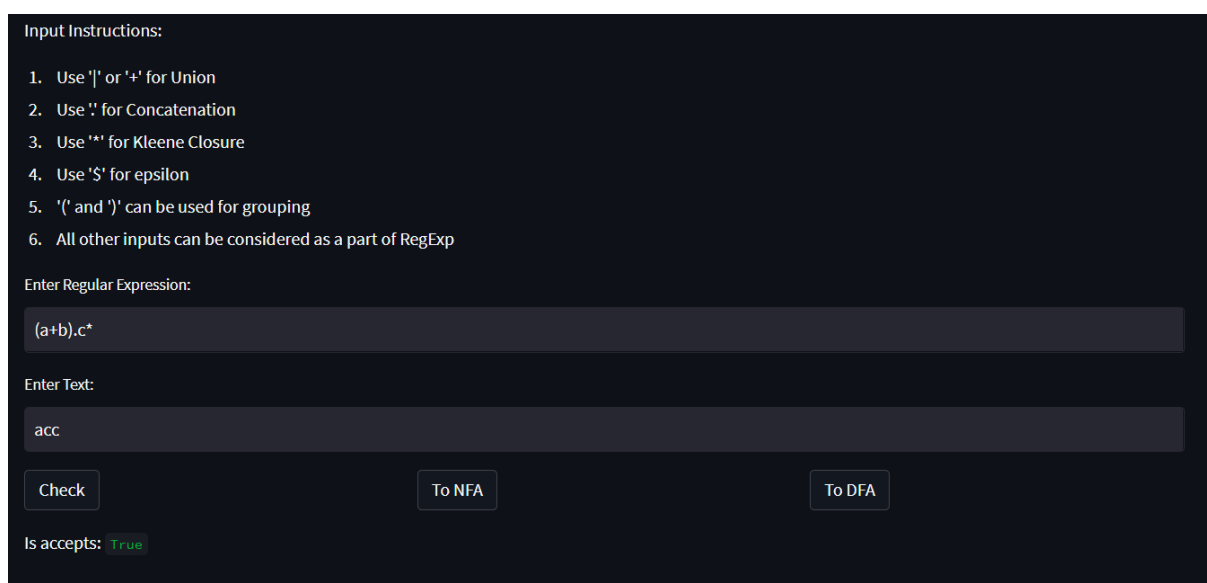
### Basic UI:



### GUI Features Implemented:

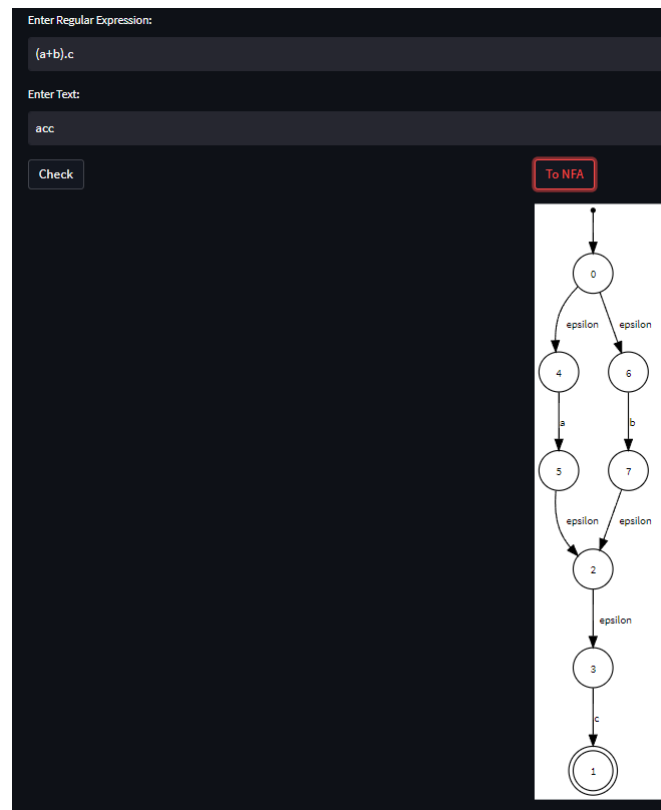
#### 1. Regular Expression evaluation

Instructions for entering the regular expression is displayed to avoid unnecessary errors and exceptions and make it user-friendly.



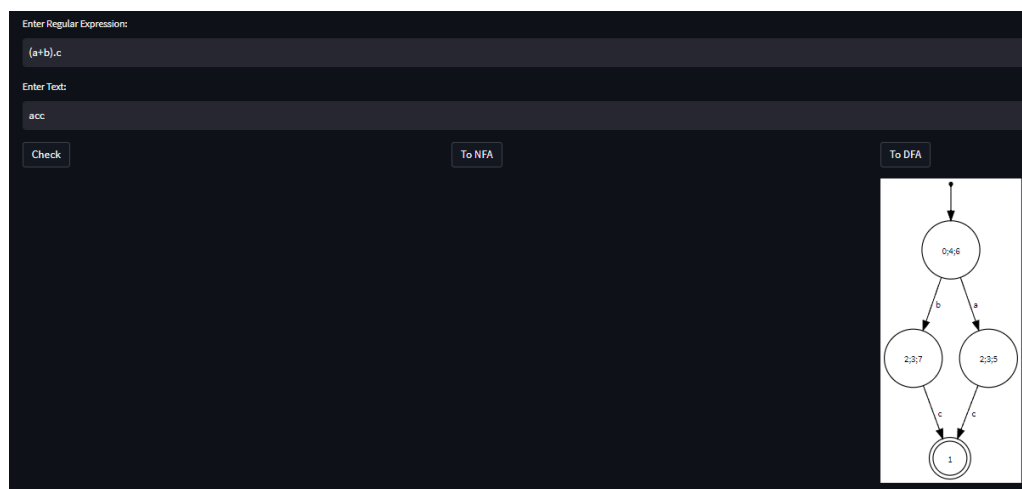
## 2. Regular Expression to NFA conversion

Using Pyfromlang package the input Regular Expression (RE) is converted into Non-Deterministic Finite Automata (NFA) by using Thompson's Rule. It also adds epsilon transitions in the output which are represented as 'epsilon' itself and the generated output will be initially in .dot format which is later rendered into .png format using graphviz package in python and displayed as output. The figure below shows an example for the conversion of RE  $(a+b).c$  into NFA when **To NFA** button is clicked.



## 3. Regular Expression to DFA conversion:

Using Pyformlang itself the RE is first converted to NFA and later it is converted into DFA and later is minimized into DFA and displayed when **To DFA** button is clicked. The below figure shows DFA for RE  $(a+b).c$



#### 4. DFA Visualization:

The DFA is given input to the UI in a text file that contains all the transitions written in each line as input in format (from\_state, to\_state, transition). Then the file is first evaluated if it represents DFA or not. If it doesn't then it will show to the user as **Invalid Input** and hides all the operations as shown below.

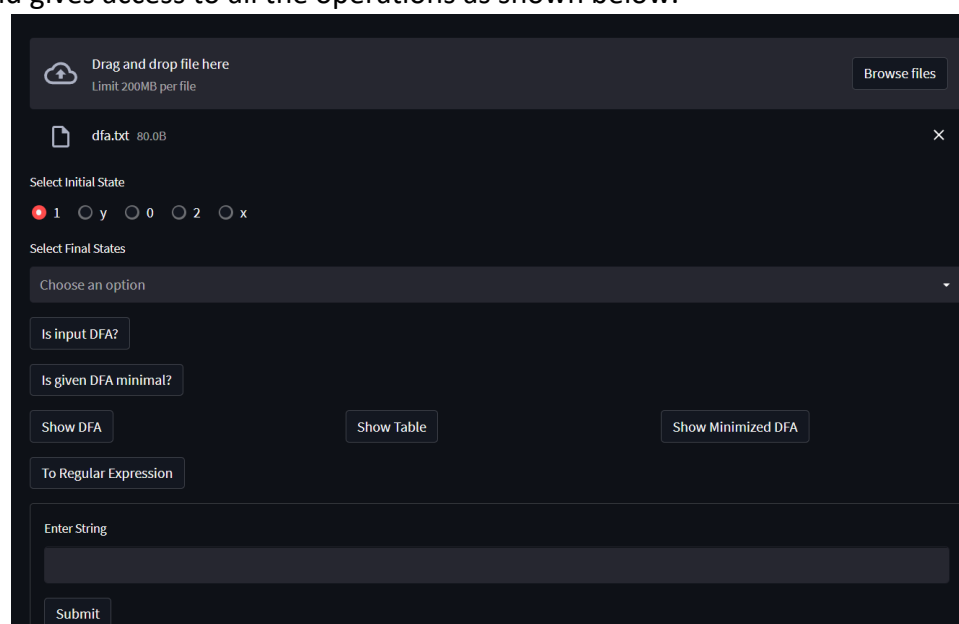


Given a text file containing the NFA transitions as input and it recognizes the transitions as not DFA and it throws an error. The contents of nfa.txt file is shown below.

The screenshot shows a Notepad window titled "nfa - Notepad". The text inside the window is as follows:

```
File Edit View
(0,0,a)
(0,1,a)
(0,1,b)
(1,2,a)
(1,2,b)
(2,2,b)
```

If the input has valid transitions for DFA then it will evaluate the file and shows the states to select and gives access to all the operations as shown below.





## 5. Checking if input is DFA

This button evaluates if the given input is DFA or not after selecting the initial and final states that are given to choose to extract from input file as shown below.

The screenshot shows a web application interface with a dark background. At the top, there is a section titled "Select Initial State" with four radio buttons labeled "1", "y", "0", "2", and "x". The "1" radio button is selected. Below this is a section titled "Select Final States" with a text input field containing "2" and a red "x" button to its right. Below the input field is a button labeled "Is input DFA?". At the bottom, the text "True" is displayed in green.

Contents of file dfa.txt is as follows

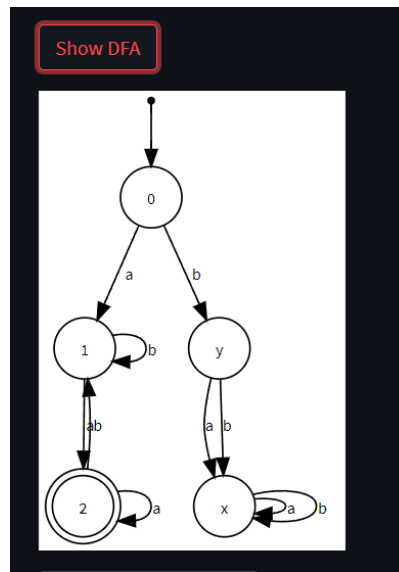
The screenshot shows a Notepad window titled "dfa - Notepad". The window has a menu bar with "File", "Edit", and "View". The text content of the file is as follows:

```
(0,1,a)
(1,1,b)
(1,2,a)
(2,1,b)
(2,2,a)
(0,y,b)
(y,x,a)
(y,x,b)
(x,x,a)
(x,x,b)
```

## 6. Checking if the input DFA is minimal or not

The screenshot shows a web application interface with a dark background. At the top, there is a section titled "Select Initial State" with four radio buttons labeled "1", "y", "0", "2", and "x". The "1" radio button is selected. Below this is a section titled "Select Final States" with a text input field containing "2" and a red "x" button to its right. Below the input field is a button labeled "Is input DFA?". Below that is a button labeled "Is given DFA minimal?". At the bottom, the text "No" is displayed.

7. Visualizing given DFA as it is

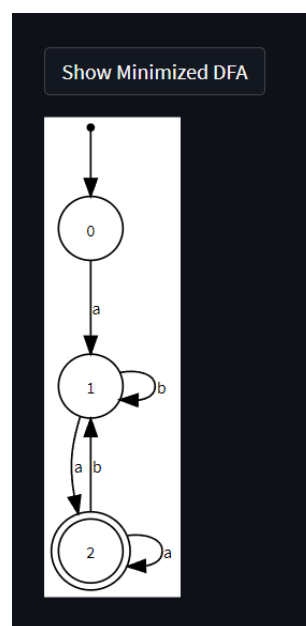


8. Visualizing transition table

Show Table

	a	b
-> 0	1	y
1	2	1
*2	2	1
y	x	x
x	x	x

9. Visualizing minimal DFA



## 10. Conversion of DFA to RE

To Regular Expression

$$(\$.((a.((b)^*.a)).((b.((b)^*.a))|a))^*)$$

## 11. Checking if a string is accepted by a DFA or not

Enter String

aaa

Submit

True

Enter String

aab

Submit

False

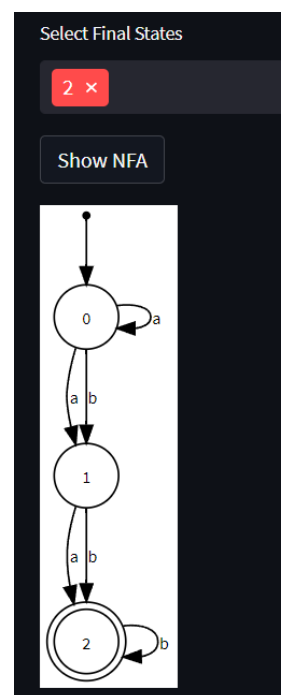
## 12. Selecting states and visualizing NFA

Select Initial State

☐ 1 ☒ 0 ☐ 2

Select Final States

2 x

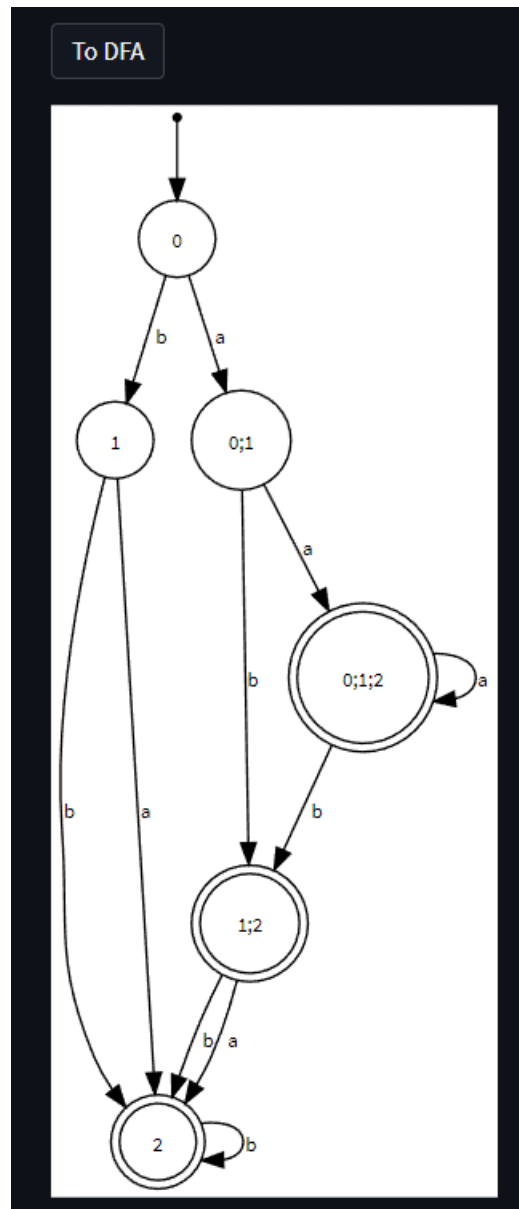


## 13. Visualizing transition table

Show Table

	a	b
-> 0	{1, 0}	{1}
1	{2}	{2}
*2	nan	{2}

#### 14. Converting NFA to DFA



#### 15. NFA to Regular Expression

To Regular Expression

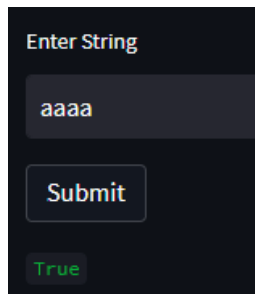
```
((a)*.(((b|a).(b|a)).(b)*))
```

#### 16. Checking if input file is DFA by chance

Is given NFA Deterministic?

```
False
```

## 17. Evaluating if a string is accepted by a NFA or not

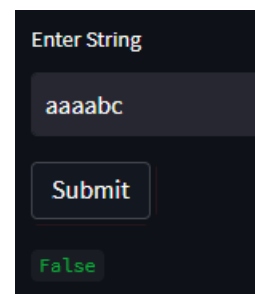


Enter String

aaaa

Submit

True



Enter String

aaaabc

Submit

False

## CONCLUSION

Through our project we extended the functionality of automata to games and shown how it can be useful in versatile scenarios.

The GUI which is developed using Streamlit can be useful to perform certain operations by visualizing them instead of using traditional pen and paper methods.

In conclusion, automata are a crucial part of the modern world. They have revolutionized the way we work, communicate, and interact with the world around us. From simple machines like vending machines and elevators to complex systems like self-driving cars and artificial intelligence, automata have the ability to perform tasks and make decisions without human intervention.

As automata continue to advance and become more integrated into our lives, it will be important to address these concerns and develop strategies to address them. This may involve the development of new laws and regulations, as well as education and training programs to ensure that individuals are prepared for the changing job market.

## FUTURE SCOPE

The Finite Automata Playground can be extended to develop a learning tool which can also display the steps to perform all the above operations so that the users can not only play with the software but also use it to learn about automata. It can also be extended to implement features such as CFGs, Push-Down Automata, etc.