VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JNANA SANGAMA”, BELAGAVI – 590018

Automata Theory and Computability (18CS54)

Assignment III

Submitted by

DEEKSHA PK	*4SF20IS029*
KARTHIK J	*4SF20IS041*
SUHAS SHETTIGAR K	*4SF20IS103*

In partial fulfillment of the requirements for V Semester of

BACHELOR OF ENGINEERING
IN
INFORMATION SCIENCE & ENGINEERING

Under the Guidance of

Mr. Rithesh Pakkala P.

Assistant Professor

Department of Information Science & Engineering

SAHYADRI

College of Engineering & Management

Mangaluru – 575 007

SAHYADRI

College of Engineering & Management

Mangaluru – 575 007

Department of Information Science & Engineering

CERTIFICATE

This is to certify that the **Assignment III** on “**Automata Theory and Computability (18CS54)**" has been completed by **DEEKSHA PK (4SF20IS029)**, **KARTHIK J (4SF20IS041)** and **SUHAS SHETTIGAR K (4SF20IS103)**, the bonafide students of **Sahyadri College of Engineering & Management** in partial fulfillment for the V semester of Bachelor of Engineering in **Information Science & Engineering** of Visvesvaraya Technological University, Belagavi during the Academic Year 2022 - 23.

Faculty Incharge

Mr. Rithesh Pakkala P.

Q. No.	Design and Implementation of GUI (5 Marks)	Output Analysis with Different Test Cases (5 Marks)	Total Marks (10)
1			
2			
3			
4			
5			
Total Marks			

SAHYADRI
COLLEGE OF ENGINEERING & MANAGEMENT
An Autonomous Institution
MANGALURU

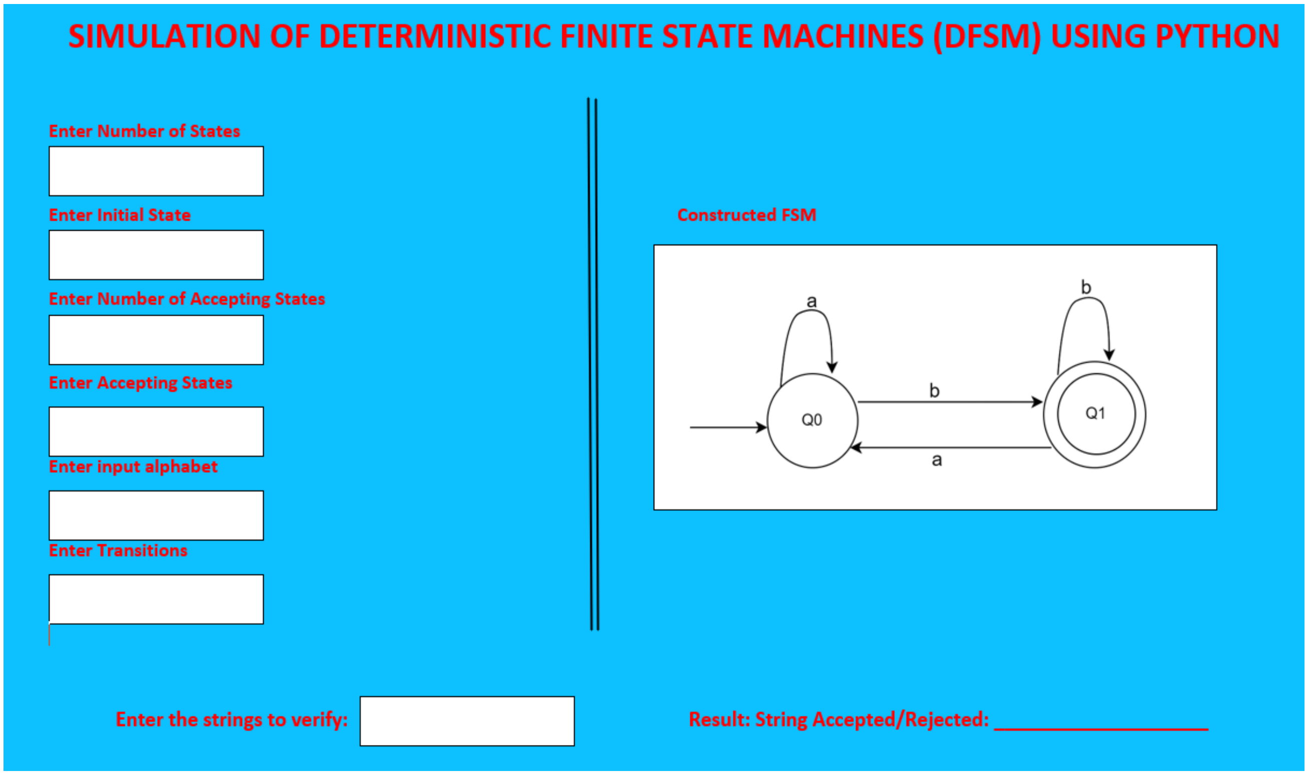
Department of Information Science & Engineering

Assignment III - ODD Semester 2022 - 23

Course Title : Automata Theory and Computability		Course Code: 18CS54
Sem / Section: V 'A'	Faculty: Mr. R Pakkala	Max. Marks: 50
Date of Announcement: 17/01/2023	Date of Submission: 25/01/2023	

Note:

- **Answer All the Five Questions**
- **The Assignment document must contain**
    - ❖ **Cover Page**
    - ❖ **Problem statement**
    - ❖ **Programming Code**
    - ❖ **Screenshots of Execution with different test cases.**
- **Student can write a program in any language (C/C++/Java/Python/C#/PHP)**
- **The GUI, may be simple frontend to accept inputs for designing FSM transition diagram and accept or reject the input string given. The sample frontend is given below:**



SIMULATION OF DETERMINISTIC FINITE STATE MACHINES (DFSM) USING PYTHON

Enter Number of States

Enter Initial State

Enter Number of Accepting States

Enter Accepting States

Enter input alphabet

Enter Transitions

Constructed FSM

Enter the strings to verify:

Result: String Accepted/Rejected: ____________________

Q. No.	Questions	Marks	Blooms Level	CO No.
1	Design and Implement a GUI (Graphical user Interface) for simulating a deterministic finite state machine (DFSM) which accept the language L = {$a^n b^m$ | n mod 2=0, m ≥1}. Analyze the output with different test cases.	10	CL3	CO1
2.	Design and Implement a GUI (Graphical user Interface) for simulating a DFSM which accept the language L = {w | w ∈ {a, b}* and Na (w) mod 3 = Nb (w) mod 3}. Analyze the output with different test cases.	10	CL3	CO1
3	Design and Implement a GUI (Graphical user Interface) for simulating a DFSM which accept strings that start and end with same character. Analyze the output with different test cases.	10	CL3	CO1
4	Design and Implement a GUI (Graphical user Interface) for simulating a DFSM which accept Binary strings that starts or ends with "01". Analyze the output with different test cases.	10	CL3	CO1
5	Design and Implement a GUI (Graphical user Interface) for simulating a DFSM which accept the language having all 'a' before all 'b'. Analyze the output with different test cases.	10	CL3	CO1

Cognitive Levels of Bloom's Taxonomy

No.	CL1	CL2	CL3	CL4	CL5	CL6
Level	Remember	Understand	Apply	Analyze	Evaluate	Create

Course Outcomes

CO1	Solve the finite state machine problems for different formal languages by discussing the central concepts of Automata Theory.	CL3
CO2	Solve the regular expression and regular grammar problems. Also discuss the proofs of regular languages.	CL3
CO3	Solve the context free grammar and pushdown automata problems for the different formal languages.	CL3
CO4	Discuss the algorithms and decision procedures for context free languages. Also solve the turing machine problems for the different formal languages.	CL3
CO5	Discuss the concepts of decidability and complexity related to computational problems.	CL2

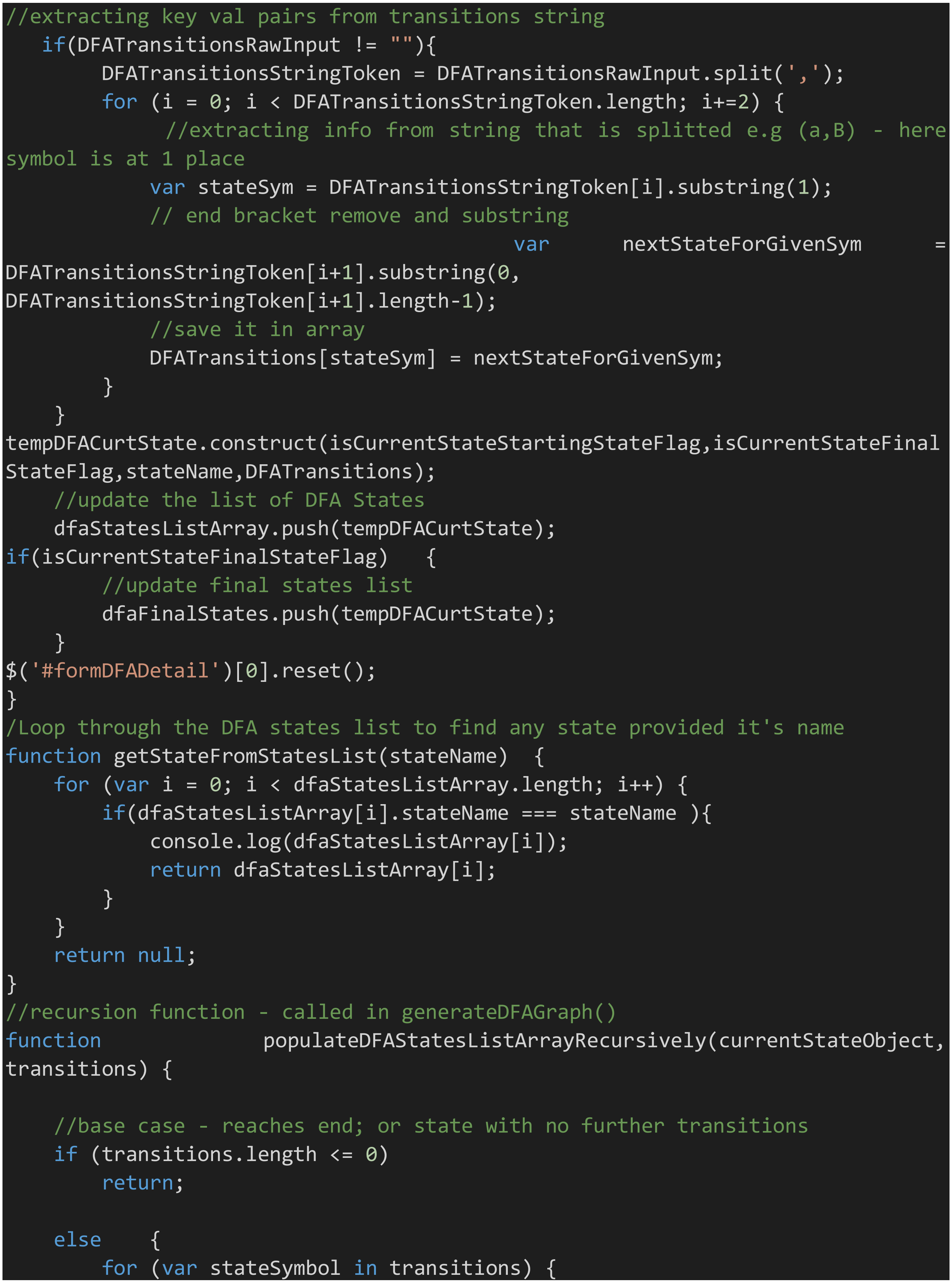
Assessment Method		
Sl. No.	**Assessment Component**	**Marks Allotted**
1.	Design and Implementation of GUI	5
2.	Output Analysis with Different Test Cases	5

Design and Implement a GUI (Graphical user Interface) for the coding done in Assignment-1.

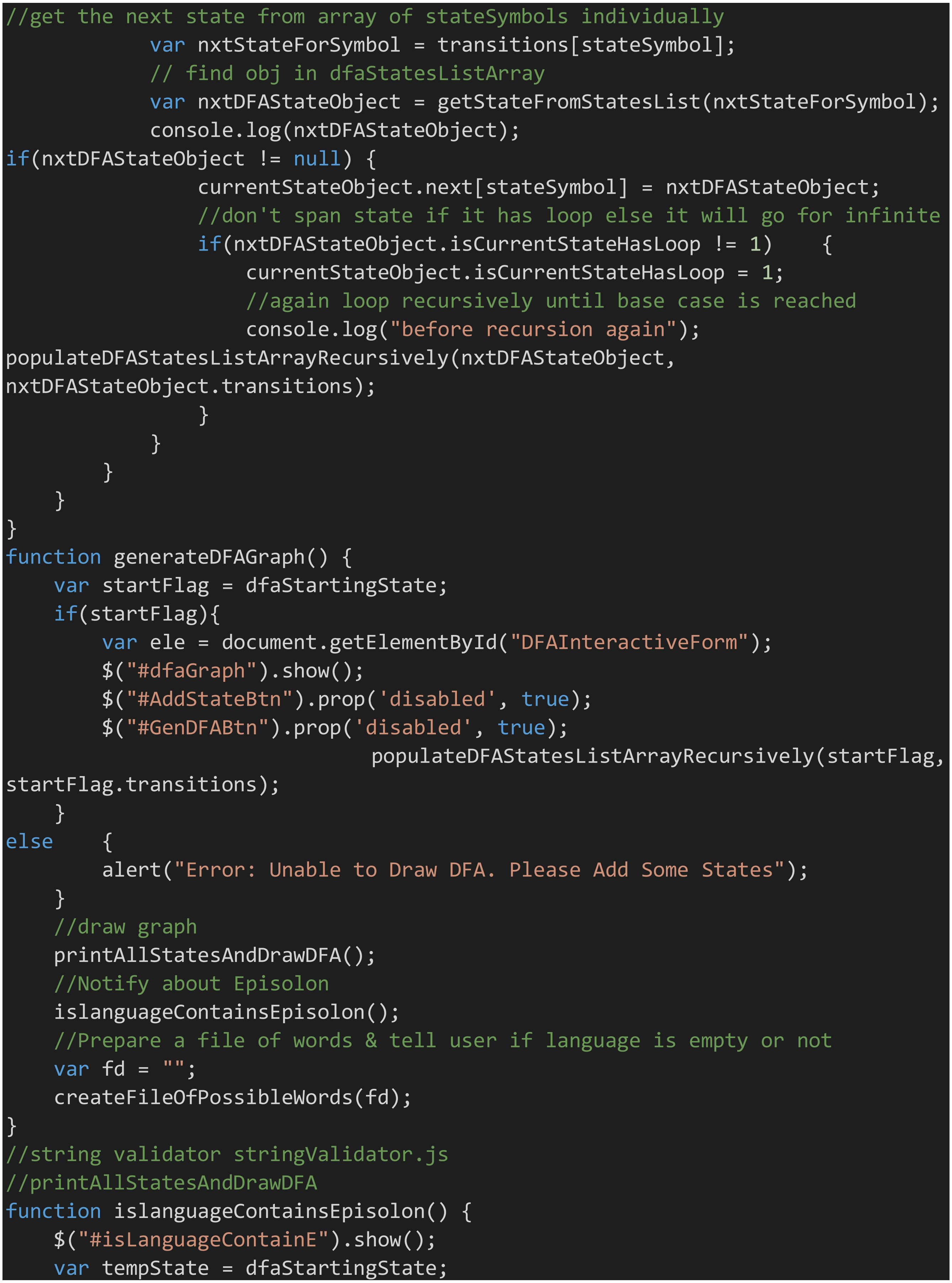
Program:

dfa.js:

```
//header.js
//user enters data in form for each state and on adding, this function handles
form data
function addStateFromFormToDFAStatesList(form) {
    //set initially values of flags = 0
    var isCurrentStateStartingStateFlag = 0;
    var isCurrentStateFinalStateFlag = 0;

    var stateName = form.stateName.value;
    var DFATransitionsRawInput = form.transitions.value;

    if($("#DFAStartStateCheckBox").is(':checked'))  {
        isCurrentStateStartingStateFlag = 1;
    }
    if($("#DFAFinalStateCheckBox").is(':checked'))  {
        isCurrentStateFinalStateFlag = 1;

        //adding state to the final states array for graph construction
        if($.inArray(stateName, graphVizFinalStates) == -1) {
            graphVizFinalStates.push(stateName);
            console.log("final states: " + graphVizFinalStates);
        }
    }
$("#dfaTransitions").append("State:  "  +  stateName  +  "  |  Transitions:  "  +
DFATransitionsRawInput + "<br/>");

    //New instance of DFAStateObject - like object of class with data about
the state
    var tempDFACurtState = new dfaStateObject();
    if(!FLAG_isStartStateExists && isCurrentStateStartingStateFlag == 1)    {
   //hide the checkbox
        $("#startingStateLabel").remove();
        FLAG_isStartStateExists = true;
        dfaStartingState = tempDFACurtState;
        graphVizStartState = stateName;
    }
 //cleaning the user input
    DFATransitionsRawInput = DFATransitionsRawInput.replace(/\s+/g, '');
    var DFATransitions = {};
```



```
//extracting key val pairs from transitions string
   if(DFATransitionsRawInput != ""){
        DFATransitionsStringToken = DFATransitionsRawInput.split(',');
        for (i = 0; i < DFATransitionsStringToken.length; i+=2) {
             //extracting info from string that is splitted e.g (a,B) - here
symbol is at 1 place
            var stateSym = DFATransitionsStringToken[i].substring(1);
            // end bracket remove and substring
                                          var      nextStateForGivenSym      =
DFATransitionsStringToken[i+1].substring(0,
DFATransitionsStringToken[i+1].length-1);
            //save it in array
            DFATransitions[stateSym] = nextStateForGivenSym;
        }
    }
tempDFACurtState.construct(isCurrentStateStartingStateFlag,isCurrentStateFinal
StateFlag,stateName,DFATransitions);
    //update the list of DFA States
    dfaStatesListArray.push(tempDFACurtState);
if(isCurrentStateFinalStateFlag)   {
        //update final states list
        dfaFinalStates.push(tempDFACurtState);
    }
$('#formDFADetail')[0].reset();
}
/Loop through the DFA states list to find any state provided it's name
function getStateFromStatesList(stateName)  {
    for (var i = 0; i < dfaStatesListArray.length; i++) {
        if(dfaStatesListArray[i].stateName === stateName ){
            console.log(dfaStatesListArray[i]);
            return dfaStatesListArray[i];
        }
    }
    return null;
}
//recursion function - called in generateDFAGraph()
function             populateDFAStatesListArrayRecursively(currentStateObject,
transitions) {

    //base case - reaches end; or state with no further transitions
    if (transitions.length <= 0)
        return;

    else    {
        for (var stateSymbol in transitions) {
```



```
//get the next state from array of stateSymbols individually
            var nxtStateForSymbol = transitions[stateSymbol];
            // find obj in dfaStatesListArray
            var nxtDFAStateObject = getStateFromStatesList(nxtStateForSymbol);
            console.log(nxtDFAStateObject);
if(nxtDFAStateObject != null) {
                currentStateObject.next[stateSymbol] = nxtDFAStateObject;
                //don't span state if it has loop else it will go for infinite
                if(nxtDFAStateObject.isCurrentStateHasLoop != 1)    {
                    currentStateObject.isCurrentStateHasLoop = 1;
                    //again loop recursively until base case is reached
                    console.log("before recursion again");
populateDFAStatesListArrayRecursively(nxtDFAStateObject,
nxtDFAStateObject.transitions);
                }
            }
        }
    }
}
function generateDFAGraph() {
    var startFlag = dfaStartingState;
    if(startFlag){
        var ele = document.getElementById("DFAInteractiveForm");
        $("#dfaGraph").show();
        $("#AddStateBtn").prop('disabled', true);
        $("#GenDFABtn").prop('disabled', true);
                               populateDFAStatesListArrayRecursively(startFlag,
startFlag.transitions);
    }
else    {
        alert("Error: Unable to Draw DFA. Please Add Some States");
    }
    //draw graph
    printAllStatesAndDrawDFA();
    //Notify about Episolon
    islanguageContainsEpisolon();
    //Prepare a file of words & tell user if language is empty or not
    var fd = "";
    createFileOfPossibleWords(fd);
}
//string validator stringValidator.js
//printAllStatesAndDrawDFA
function islanguageContainsEpisolon() {
    $("#isLanguageContainE").show();
    var tempState = dfaStartingState;
```

```
if(tempState.isCurrentStateFinalStateFlag == 1) {
            $("#isLanguageContainE").html("<b>Language Contains ε </b>");
    } else {
         $("#isLanguageContainE").html("<b>Language doesn't Contains ε </b>");

    }
}
//dfaLanguageWords.js
```

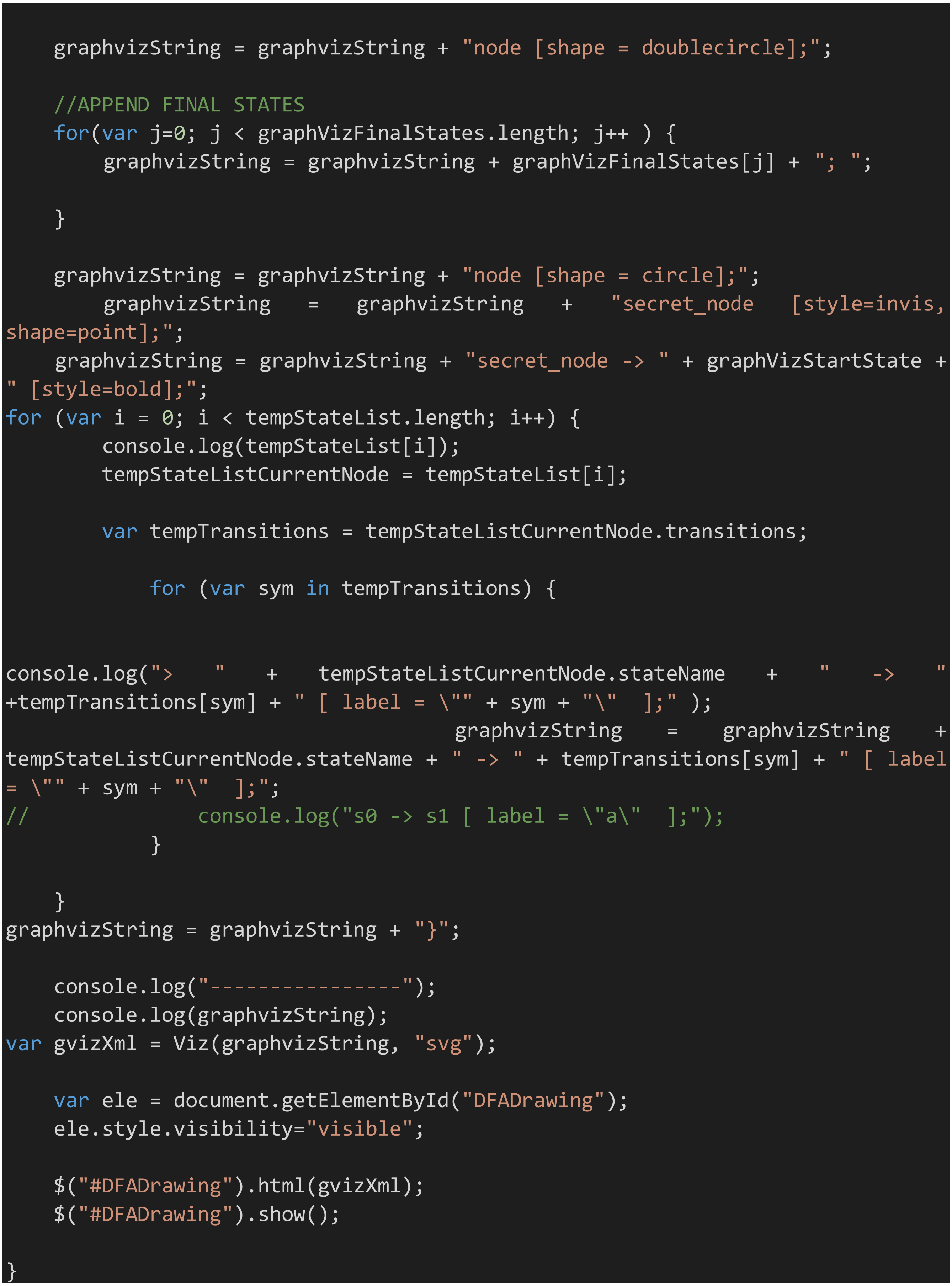
DfaLanguageWords.js:

```
function createFileOfPossibleWords(fileDescriptor) {

    dfaLanguageWordsArray = [];

    var wordString = "";
    var currWordLength = 0;

    console.log("In prepareAcceptedWordRecursively\n");
              prepareAcceptedWordRecursively(dfaStartingState, wordString,
currWordLength);
    console.log(dfaLanguageWordsArray);

    //update the link of download file
    var link = document.getElementById('fileDownloadLink');
    link.href = makeTextFile(dfaLanguageWordsArray);
    link.style.display = 'block';
    $("#isLanguageEmptyAlert").show();
if(dfaLanguageWordsArray.length == 0) {
        console.log("Empty Language");
        $("#isLanguageEmptyAlert").html("<b>Language is Empty </b>");

    } else {
        console.log("non empty language");
        $("#isLanguageEmptyAlert").html("<b>Language is not Empty </b>");
    }

    return dfaLanguageWordsArray;

}
function prepareAcceptedWordRecursively(currentStateObject, wordString,
currWordLength) {
    //incrementing the length as we need upto 10
    currWordLength++;
    if(currentStateObject.isCurrentStateFinalStateFlag == 1){
        if(wordString==""){
```

```
dfaLanguageWordsArray.push("E");
        } else {
            dfaLanguageWordsArray.push(wordString);
        }
    }
if(Object.getOwnPropertyNames(currentStateObject.next).length === 0||
currWordLength > 10){
        return;
    }
    else    {
        for (var DFAStateSymbol in currentStateObject.next) {
            var nextDFAState = currentStateObject.next[DFAStateSymbol];
            //concatenate and extend word
            concatenatedWord = wordString + DFAStateSymbol;
            if(nextDFAState != null){
                //recursively span and loop through it.
                prepareAcceptedWordRecursively(nextDFAState, concatenatedWord,
currWordLength);
            }
        }
    }

}
var textFile = null,
      makeTextFile = function (text) {
        var data = new Blob([text], {type: 'text/plain'});

        // If we are replacing a previously generated file we need to
        // manually revoke the object URL to avoid memory leaks.
        if (textFile !== null) {
          window.URL.revokeObjectURL(textFile);
        }

        textFile = window.URL.createObjectURL(data);

        // returns a URL you can use as a href
        return textFile;
  };
```

drawDFA.js:

```
function printAllStatesAndDrawDFA() {

    var tempStateList = dfaStatesListArray;

    var graphvizString = "digraph finite_state_machine {";
    graphvizString = graphvizString + "rankdir=LR;";
```



```
    graphvizString = graphvizString + "node [shape = doublecircle];";

    //APPEND FINAL STATES
    for(var j=0; j < graphVizFinalStates.length; j++ ) {
        graphvizString = graphvizString + graphVizFinalStates[j] + "; ";

    }

    graphvizString = graphvizString + "node [shape = circle];";
        graphvizString = graphvizString + "secret_node [style=invis,
shape=point];";
    graphvizString = graphvizString + "secret_node -> " + graphVizStartState +
" [style=bold];";
for (var i = 0; i < tempStateList.length; i++) {
        console.log(tempStateList[i]);
        tempStateListCurrentNode = tempStateList[i];

        var tempTransitions = tempStateListCurrentNode.transitions;

            for (var sym in tempTransitions) {


console.log(">  " + tempStateListCurrentNode.stateName + " -> "
+tempTransitions[sym] + " [ label = \"" + sym + "\"  ];" );
                                     graphvizString = graphvizString +
tempStateListCurrentNode.stateName + " -> " + tempTransitions[sym] + " [ label
= \"" + sym + "\"  ];";
//              console.log("s0 -> s1 [ label = \"a\"  ];");
            }

    }
graphvizString = graphvizString + "}";

    console.log("----------------");
    console.log(graphvizString);
var gvizXml = Viz(graphvizString, "svg");

    var ele = document.getElementById("DFADrawing");
    ele.style.visibility="visible";

    $("#DFADrawing").html(gvizXml);
    $("#DFADrawing").show();

}
```

Header.js:

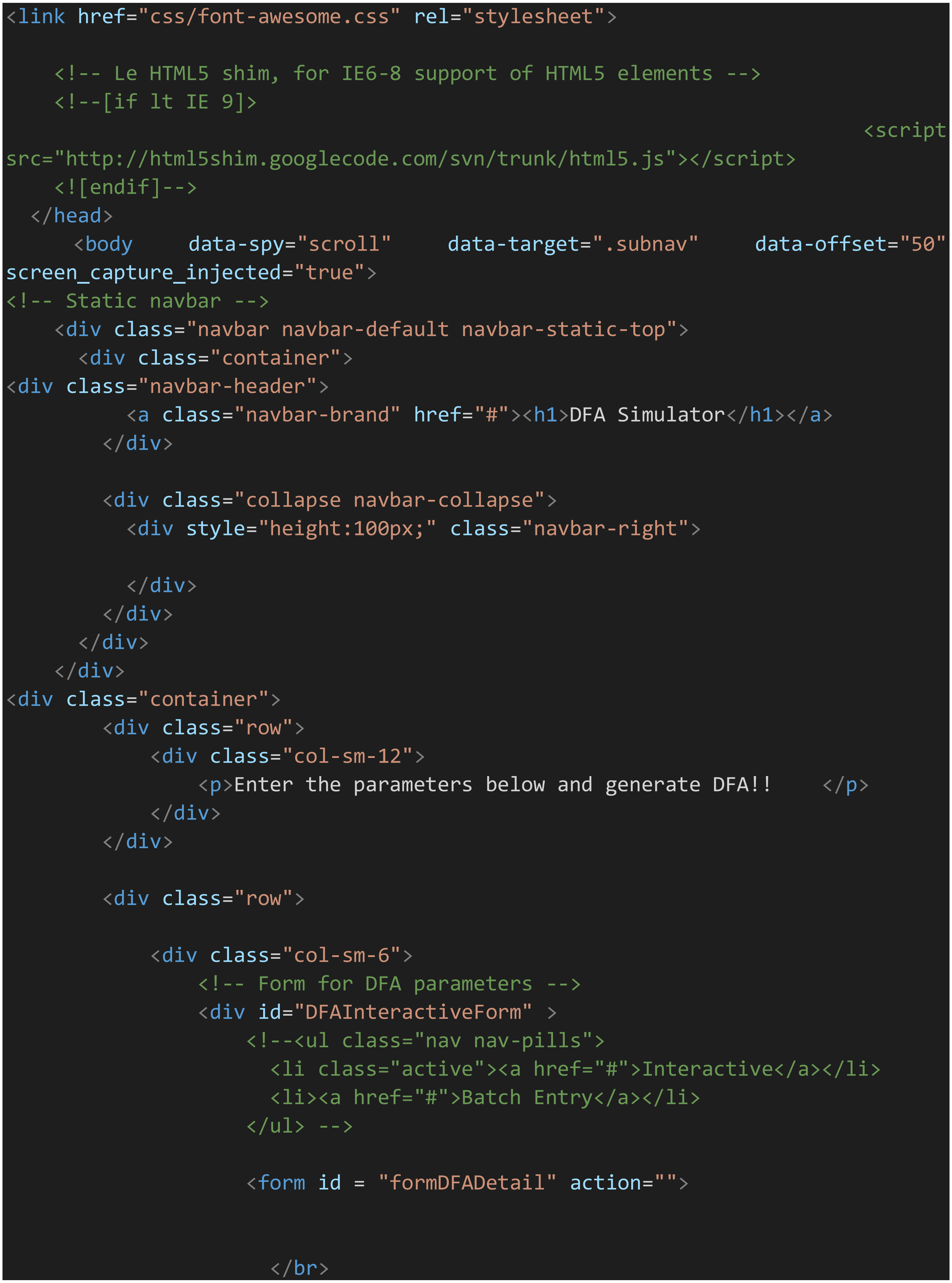
```
var FLAG_isStartStateExists = false;    //flag to check if user has
set starting state, initially set false
var dfaStatesListArray = [];    // holds the states of DFA
var dfaStartingState = null;    // reference to the starting state of FSM
var graphVizFinalStates = [];   //holds final states of DFA - graphviz helper
var graphVizStartState = "";    //holds Start states of DFA - graphgiz helper

var dfaFinalStates = [];    //holds final states of DFA
var dfaLanguageWordsArray = [];      //helper for createFileOfPossibleWords
function. stores the words of language

//helper function - class Alike! Stores information of a DFA state

 function dfaStateObject(){
    this.stateName = "";
    this.isCurrentStateStartingStateFlag = 0;
    this.isCurrentStateFinalStateFlag = 0;
     this.transitions = {};
     this.next = {}; //KeyVal pairs
    this.isCurrentStateHasLoop = 0;
this.construct = function(isCurrentStateStartingStateFlag,
isCurrentStateFinalStateFlag, stateName, transitions)   {
                                  this.isCurrentStateStartingStateFlag =
isCurrentStateStartingStateFlag;
        this.isCurrentStateFinalStateFlag = isCurrentStateFinalStateFlag;
        this.stateName = stateName;
        this.transitions = transitions;
    }
}
```

stringValidator.js:

```
// test the user string for validity
function testUserString(form)   {

    var userInputString = form.inputTestString.value;
    console.log("String to Check: " + userInputString );

    //get an instance of starting state as we need to start checking from that
    var tempState = dfaStartingState;
    console.log(tempState);
            console.log("here: val of final state: " +
tempState.isCurrentStateFinalStateFlag);
    //language accepts Episolon
if(tempState.isCurrentStateFinalStateFlag==1 && userInputString == "")  {
        $("#StringValidationAlert").removeClass('alert-danger');
        $("#StringValidationAlert").addClass('alert-success');
                      $("#stringValidationText").html('<font face="verdana"
color="white">Valid!</font>');
        return 1;
    }
// none found
    for (var i = 0; i < userInputString.length; i++) {
        console.log(tempState);
        tempState = tempState.next[userInputString.charAt(i)];
        console.log(tempState);
        if(!tempState)  {
            $("#StringValidationAlert").addClass('alert-danger');
            $("#stringValidationText").html('<font face="verdana"
color="white">Invalid!</font>');
            return 0;
        }
    }
//if it is accepting - final state
    if(tempState.isCurrentStateFinalStateFlag == 1) {
        $("#StringValidationAlert").removeClass('alert-danger');
        $("#StringValidationAlert").addClass('alert-success');
                      $("#stringValidationText").html('<font face="verdana"
color="white">Valid!</font>');
        return 1;
    }
$("#StringValidationAlert").addClass('alert-danger');
                  $("#stringValidationText").html('<font face="verdana"
color="white">Invalid!</font>');
    return 0;
}
```

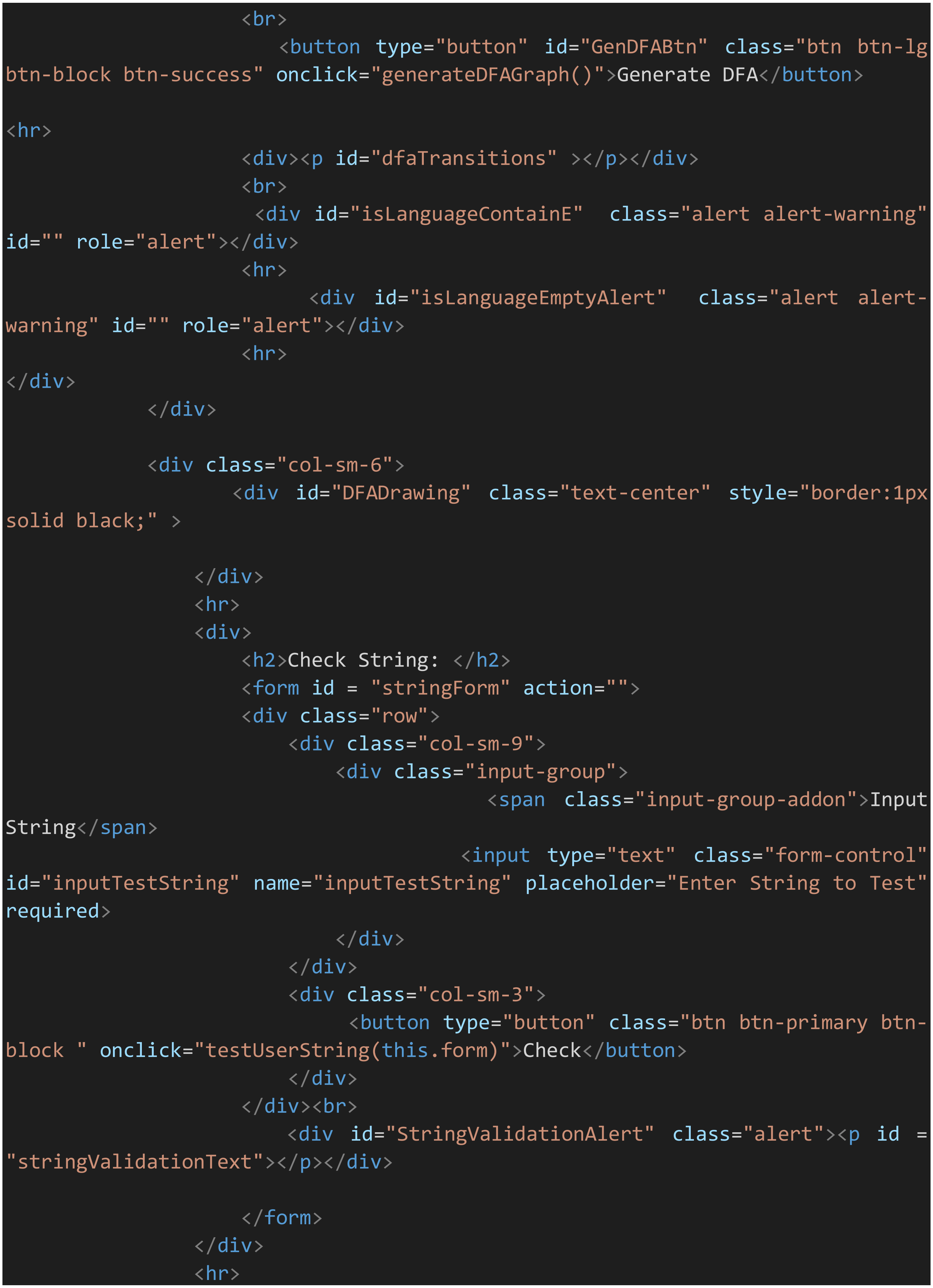
Index.html:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <meta charset="utf-8">
    <title>DFA Simulator</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta name="description" content="">
    <meta name="author" content="Shumail Mohy-ud-Din">
    <!-- Le styles -->
    <link href="css/metro-bootstrap.css" rel="stylesheet">
```

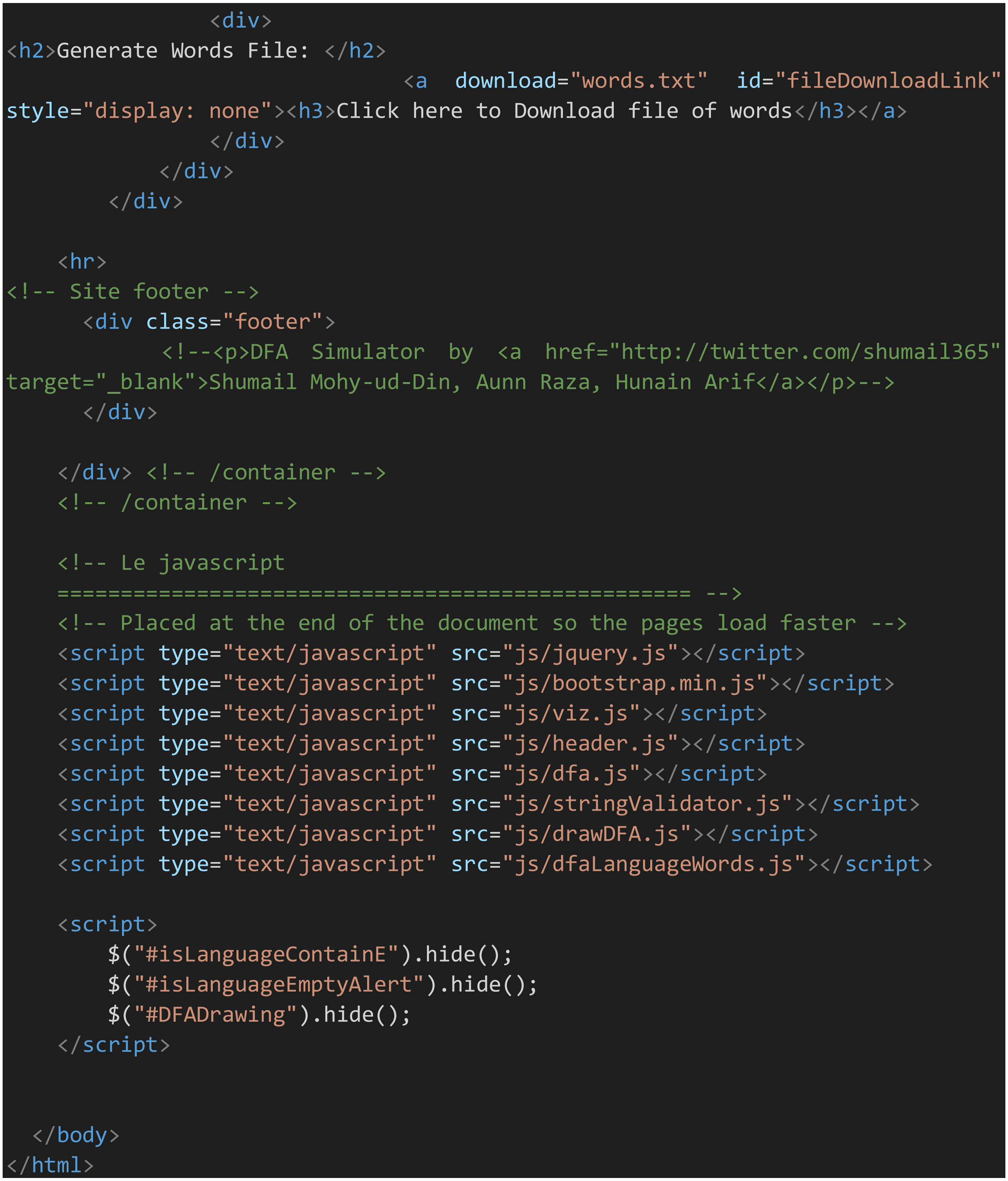


```
<link href="css/font-awesome.css" rel="stylesheet">

    <!-- Le HTML5 shim, for IE6-8 support of HTML5 elements -->
    <!--[if lt IE 9]>
                                                                         <script
src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]-->
  </head>
      <body     data-spy="scroll"     data-target=".subnav"     data-offset="50"
screen_capture_injected="true">
<!-- Static navbar -->
    <div class="navbar navbar-default navbar-static-top">
      <div class="container">
<div class="navbar-header">
          <a class="navbar-brand" href="#"><h1>DFA Simulator</h1></a>
        </div>

        <div class="collapse navbar-collapse">
          <div style="height:100px;" class="navbar-right">

          </div>
        </div>
      </div>
    </div>
<div class="container">
        <div class="row">
            <div class="col-sm-12">
                <p>Enter the parameters below and generate DFA!!    </p>
            </div>
        </div>

        <div class="row">

            <div class="col-sm-6">
                <!-- Form for DFA parameters -->
                <div id="DFAInteractiveForm" >
                    <!--<ul class="nav nav-pills">
                      <li class="active"><a href="#">Interactive</a></li>
                      <li><a href="#">Batch Entry</a></li>
                    </ul> -->

                    <form id = "formDFADetail" action="">


                      </br>
```

```
<div class="form-group">

                        <div class="row">
                            <div class="col-sm-6">
                                    <div class="input-group">
                                                     <span class="input-group-
addon">State</span>
                                       <input type="text" class="form-control"
id="stateName" name="stateName" placeholder="State Name" required>
                                    </div>
</div>
<div class="col-sm-3">
                                              <label class="control checkbox"
id="startingStateLabel">
                                                       <input type="checkbox"
id="DFAStartStateCheckBox" name="DFAStartStateCheckBox" value="1" >
                                        <span class="checkbox-label"> Starting
State</span>
                                <label>
                            </div>
<div class="col-sm-3">
                                              <label class="control checkbox"
id="startingStateLabel">
                                                       <input type="checkbox"
id="DFAFinalStateCheckBox" name="DFAFinalStateCheckBox" value="1" >
                                           <span class="checkbox-label"> Final
State</span>
                                </label>
                            </div>
                        </div>

                        <br>
                        <div class="input-group">
                          <span class="input-group-addon">Transitions</span>
                                      <input type="text" class="form-control"
id="transitions" name="transitions" placeholder="(Symbol, Next-State),(Symbol,
Next-State) "required>
                        </div>

                    </div>
                      </br>
                      <!-- !-->
                      <button type="button" id="AddStateBtn" class="btn btn-lg
btn-block                                                          btn-primary"
onclick="addStateFromFormToDFAStatesList(this.form)">Add State</button>
                    </form>
```



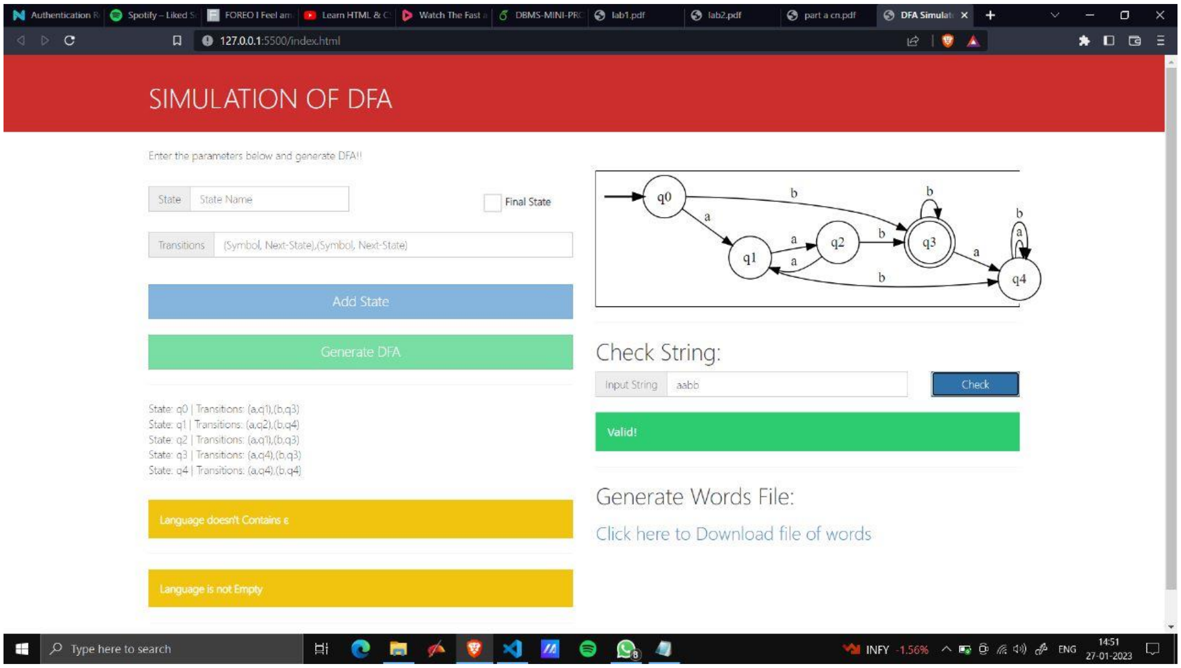
```
                    <br>
                       <button type="button" id="GenDFABtn" class="btn btn-lg
btn-block btn-success" onclick="generateDFAGraph()">Generate DFA</button>

<hr>
                    <div><p id="dfaTransitions" ></p></div>
                    <br>
                     <div id="isLanguageContainE" class="alert alert-warning"
id="" role="alert"></div>
                    <hr>
                          <div id="isLanguageEmptyAlert" class="alert alert-
warning" id="" role="alert"></div>
                    <hr>
</div>
            </div>

            <div class="col-sm-6">
                   <div id="DFADrawing" class="text-center" style="border:1px
solid black;" >

                </div>
                <hr>
                <div>
                    <h2>Check String: </h2>
                    <form id = "stringForm" action="">
                    <div class="row">
                        <div class="col-sm-9">
                            <div class="input-group">
                                         <span class="input-group-addon">Input
String</span>
                                       <input type="text" class="form-control"
id="inputTestString" name="inputTestString" placeholder="Enter String to Test"
required>
                            </div>
                        </div>
                        <div class="col-sm-3">
                             <button type="button" class="btn btn-primary btn-
block " onclick="testUserString(this.form)">Check</button>
                        </div>
                    </div><br>
                        <div id="StringValidationAlert" class="alert"><p id =
"stringValidationText"></p></div>

                    </form>
                </div>
                <hr>
```



```
                <div>
<h2>Generate Words File: </h2>
                                <a  download="words.txt"  id="fileDownloadLink"
style="display: none"><h3>Click here to Download file of words</h3></a>
                </div>
            </div>
        </div>

    <hr>
<!-- Site footer -->
      <div class="footer">
            <!--<p>DFA  Simulator  by  <a  href="http://twitter.com/shumail365"
target="_blank">Shumail Mohy-ud-Din, Aunn Raza, Hunain Arif</a></p>-->
      </div>

    </div> <!-- /container -->
    <!-- /container -->

    <!-- Le javascript
    ================================================== -->
    <!-- Placed at the end of the document so the pages load faster -->
    <script type="text/javascript" src="js/jquery.js"></script>
    <script type="text/javascript" src="js/bootstrap.min.js"></script>
    <script type="text/javascript" src="js/viz.js"></script>
    <script type="text/javascript" src="js/header.js"></script>
    <script type="text/javascript" src="js/dfa.js"></script>
    <script type="text/javascript" src="js/stringValidator.js"></script>
    <script type="text/javascript" src="js/drawDFA.js"></script>
    <script type="text/javascript" src="js/dfaLanguageWords.js"></script>

    <script>
        $("#isLanguageContainE").hide();
        $("#isLanguageEmptyAlert").hide();
        $("#DFADrawing").hide();
    </script>


  </body>
</html>
```

1. Illustrate the design in step by step approach and write a program to simulate deterministic finite state machine (DFSM) for accepting the language L = {$a^n b^m$ | n mod 2=0, m ≥1}. Analyze the output with different test cases.

   Output :



Enter the parameters below and generate DFA!!

State: State Name □ Final State

Transitions: (Symbol, Next-State),(Symbol, Next-State)

State: q0 | Transitions: (a,q1),(b,q3)
State: q1 | Transitions: (a,q2),(b,q4)
State: q2 | Transitions: (a,q1),(b,q3)
State: q3 | Transitions: (a,q4),(b,q3)
State: q4 | Transitions: (a,q4),(b,q4)

Language doesn't Contains ε

Language is not Empty

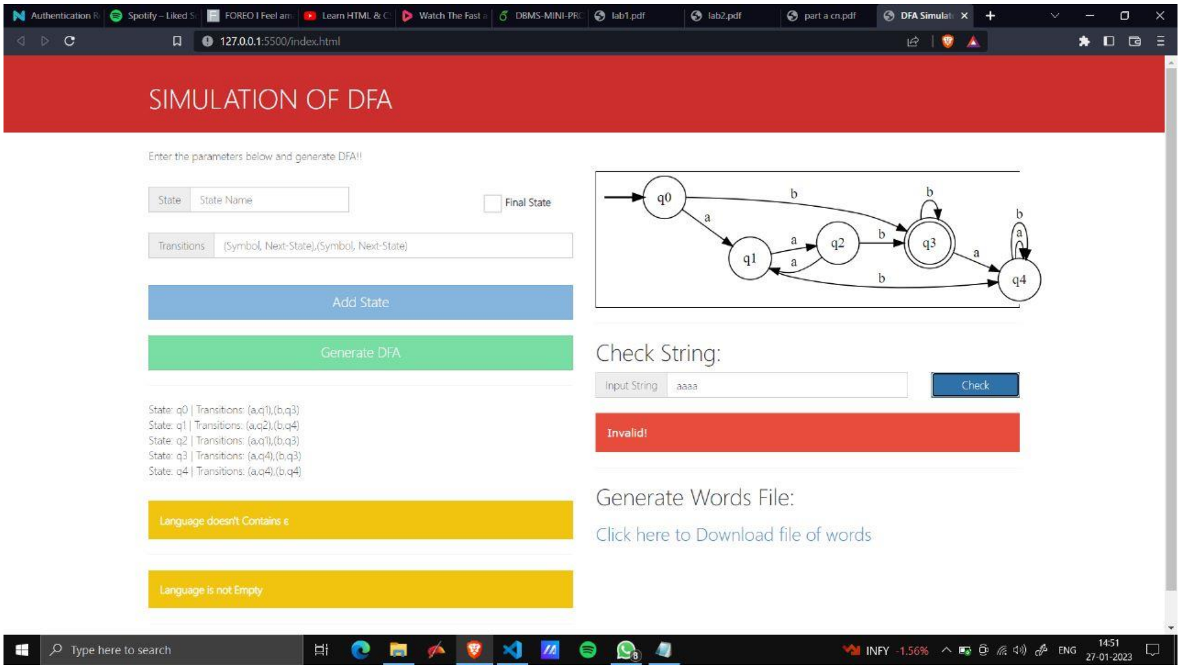
Check String:

Input String: aabb

Valid!

Generate Words File:

Click here to Download file of words



Enter the parameters below and generate DFA!!

State: State Name □ Final State

Transitions: (Symbol, Next-State),(Symbol, Next-State)

State: q0 | Transitions: (a,q1),(b,q3)
State: q1 | Transitions: (a,q2),(b,q4)
State: q2 | Transitions: (a,q1),(b,q3)
State: q3 | Transitions: (a,q4),(b,q3)
State: q4 | Transitions: (a,q4),(b,q4)

Language doesn't Contains ε

Language is not Empty

Check String:

Input String: aaaa

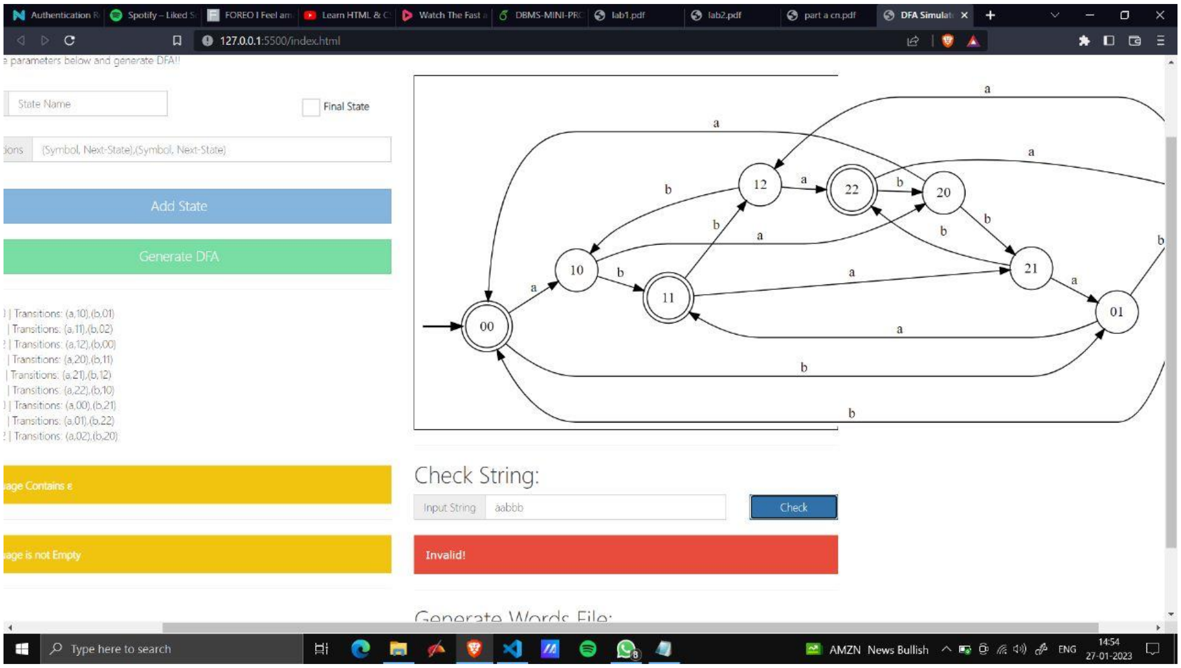
Invalid!

Generate Words File:

Click here to Download file of words

2. Illustrate the design in step by step approach and write a program to simulate a DFSM which accept the language L = {w | w ∈ {a, b}* and $N_a(w)$ mod 3 = $N_b(w)$ mod 3}. Analyze the output with different test cases.

   Output :



e parameters below and generate DFA!!

State Name □ Final State

ions (Symbol, Next-State),(Symbol, Next-State)

Add State

Generate DFA

| Transitions: (a,10),(b,01)
| Transitions: (a,11),(b,02)
| Transitions: (a,12),(b,00)
| Transitions: (a,20),(b,11)
| Transitions: (a,21),(b,12)
| Transitions: (a,22),(b,10)
| Transitions: (a,00),(b,21)
| Transitions: (a,01),(b,22)
| Transitions: (a,02),(b,20)

uage Contains ε

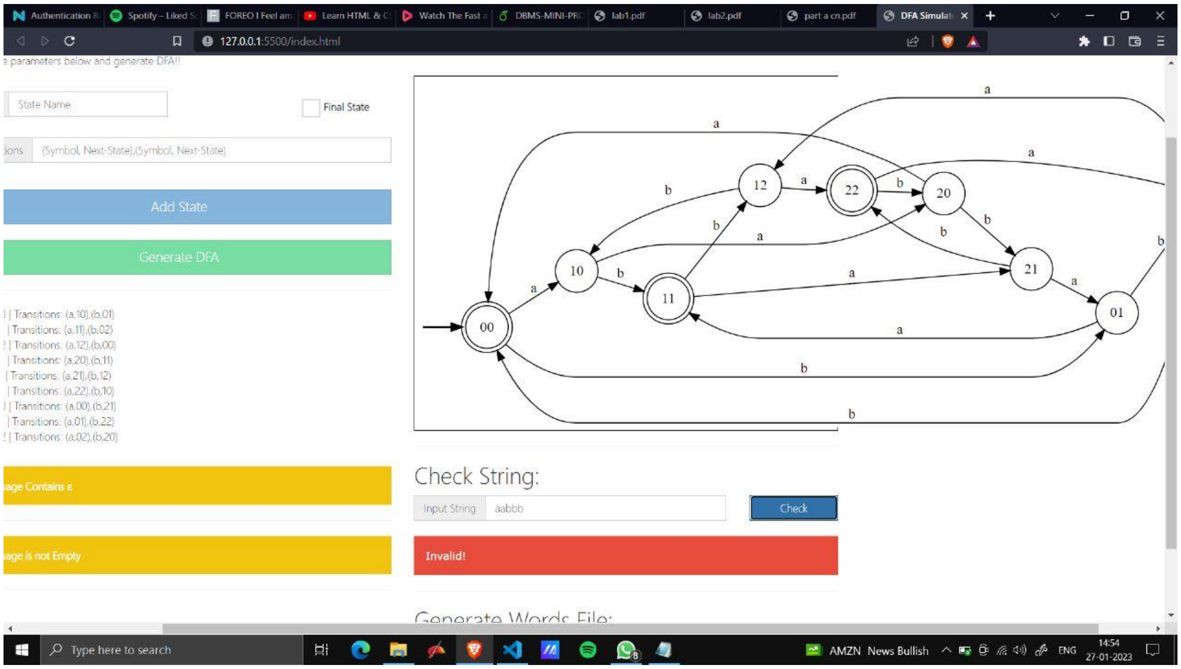
uage is not Empty

Check String:

Input String aabbb Check

Invalid!

Generate Words File:



e parameters below and generate DFA!!

State Name □ Final State

ions (Symbol, Next-State),(Symbol, Next-State)

Add State

Generate DFA

| Transitions: (a,10),(b,01)
| Transitions: (a,11),(b,02)
| Transitions: (a,12),(b,00)
| Transitions: (a,20),(b,11)
| Transitions: (a,21),(b,12)
| Transitions: (a,22),(b,10)
| Transitions: (a,00),(b,21)
| Transitions: (a,01),(b,22)
| Transitions: (a,02),(b,20)

uage Contains ε

uage is not Empty

Check String:

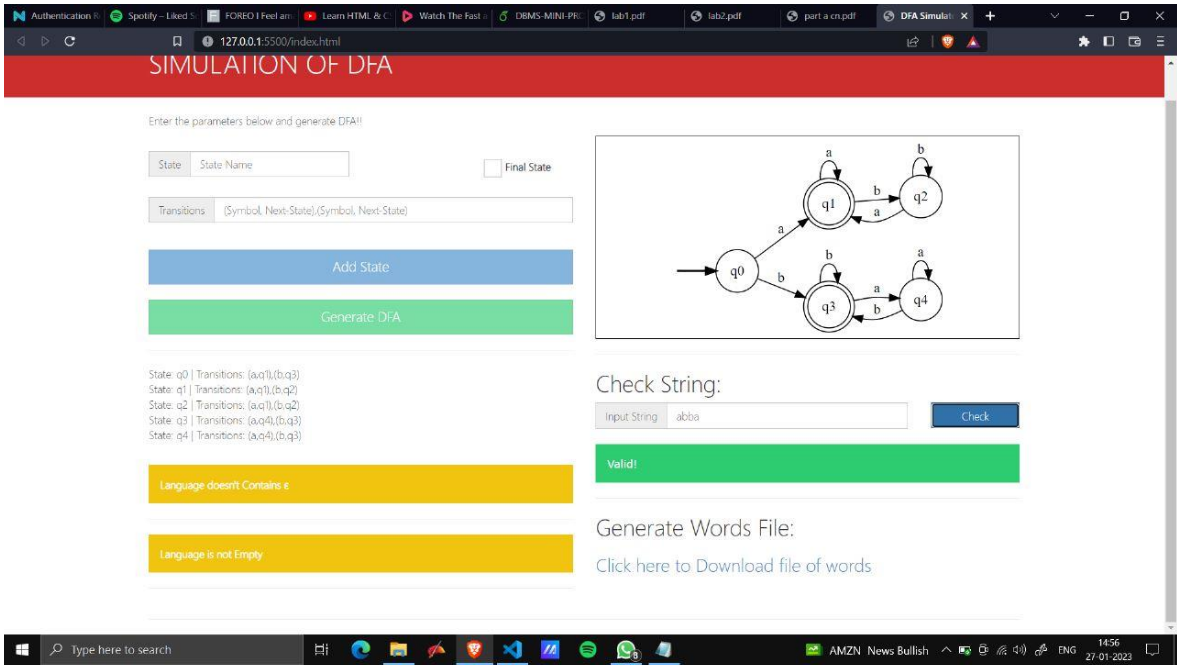
Input String aabbb Check

Invalid!

Generate Words File:

3. Illustrate the design in step by step approach and write a program to simulate a DFSM which accept strings that start and end with same character. Analyze the output with different test cases.

   Output :



SIMULATION OF DFA

Enter the parameters below and generate DFA!!

State: q0 | Transitions: (a,q1),(b,q3)
State: q1 | Transitions: (a,q1),(b,q2)
State: q2 | Transitions: (a,q1),(b,q2)
State: q3 | Transitions: (a,q4),(b,q3)
State: q4 | Transitions: (a,q4),(b,q3)

Language doesn't Contains ε

Language is not Empty

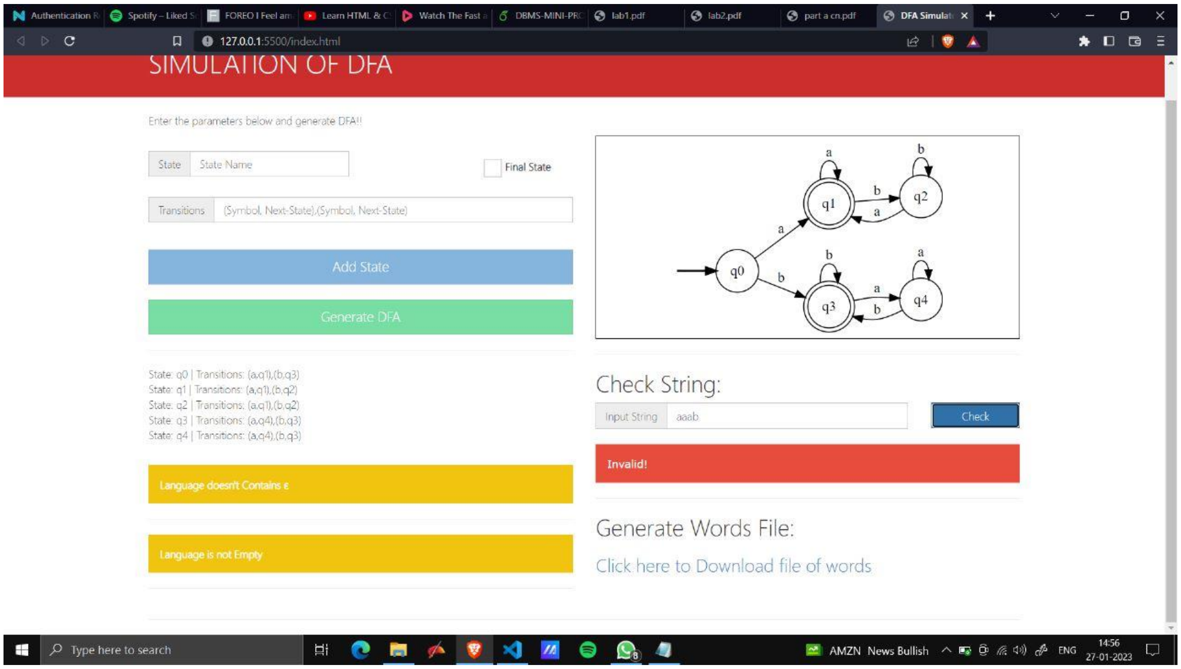
Check String:

Input String: abba

Valid!

Generate Words File:

Click here to Download file of words



SIMULATION OF DFA

Enter the parameters below and generate DFA!!

State: q0 | Transitions: (a,q1),(b,q3)
State: q1 | Transitions: (a,q1),(b,q2)
State: q2 | Transitions: (a,q1),(b,q2)
State: q3 | Transitions: (a,q4),(b,q3)
State: q4 | Transitions: (a,q4),(b,q3)

Language doesn't Contains ε

Language is not Empty

Check String:

Input String: aaab

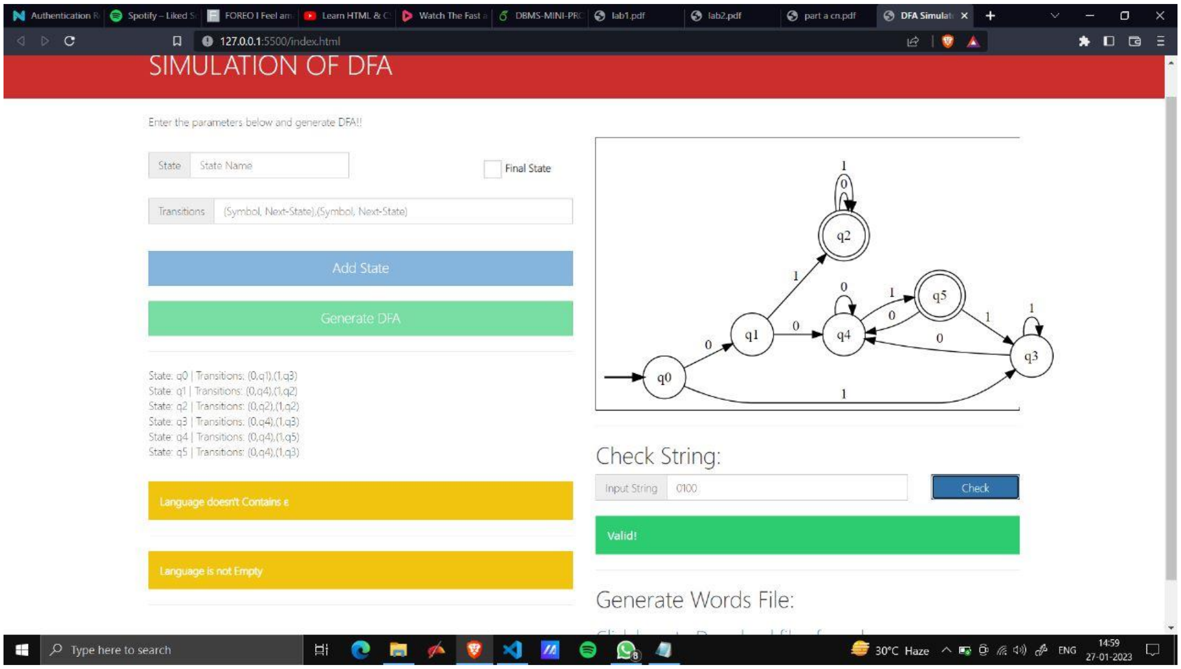
Invalid!

Generate Words File:

Click here to Download file of words

4. Illustrate the design in step by step approach and write a program to simulate a DFSM which accept Binary strings that starts or ends with “01”. Analyze the output with different test cases.

   Output :



SIMULATION OF DFA

Enter the parameters below and generate DFA!!

State: State Name □ Final State

Transitions: (Symbol, Next-State),(Symbol, Next-State)

State: q0 | Transitions: (0,q1),(1,q3)
State: q1 | Transitions: (0,q4),(1,q2)
State: q2 | Transitions: (0,q2),(1,q2)
State: q3 | Transitions: (0,q4),(1,q3)
State: q4 | Transitions: (0,q4),(1,q5)
State: q5 | Transitions: (0,q4),(1,q3)

Language doesn't Contains ε

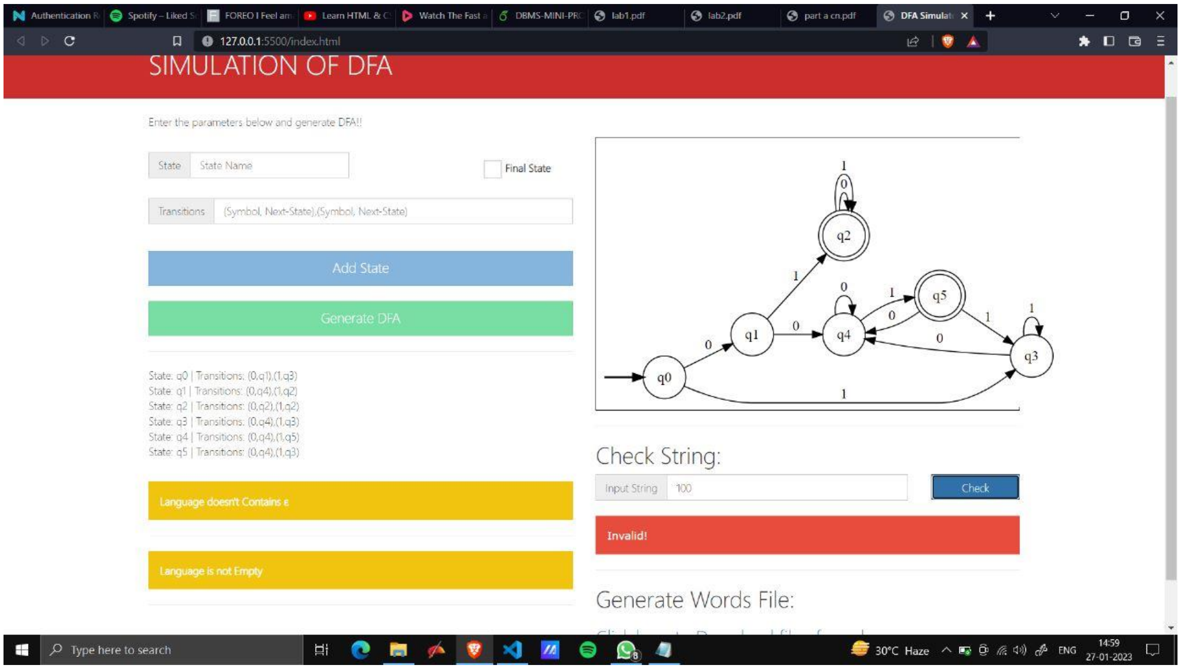
Language is not Empty

Check String:

Input String: 0100

Valid!

Generate Words File:



SIMULATION OF DFA

Enter the parameters below and generate DFA!!

State: State Name □ Final State

Transitions: (Symbol, Next-State),(Symbol, Next-State)

State: q0 | Transitions: (0,q1),(1,q3)
State: q1 | Transitions: (0,q4),(1,q2)
State: q2 | Transitions: (0,q2),(1,q2)
State: q3 | Transitions: (0,q4),(1,q3)
State: q4 | Transitions: (0,q4),(1,q5)
State: q5 | Transitions: (0,q4),(1,q3)

Language doesn't Contains ε

Language is not Empty

Check String:

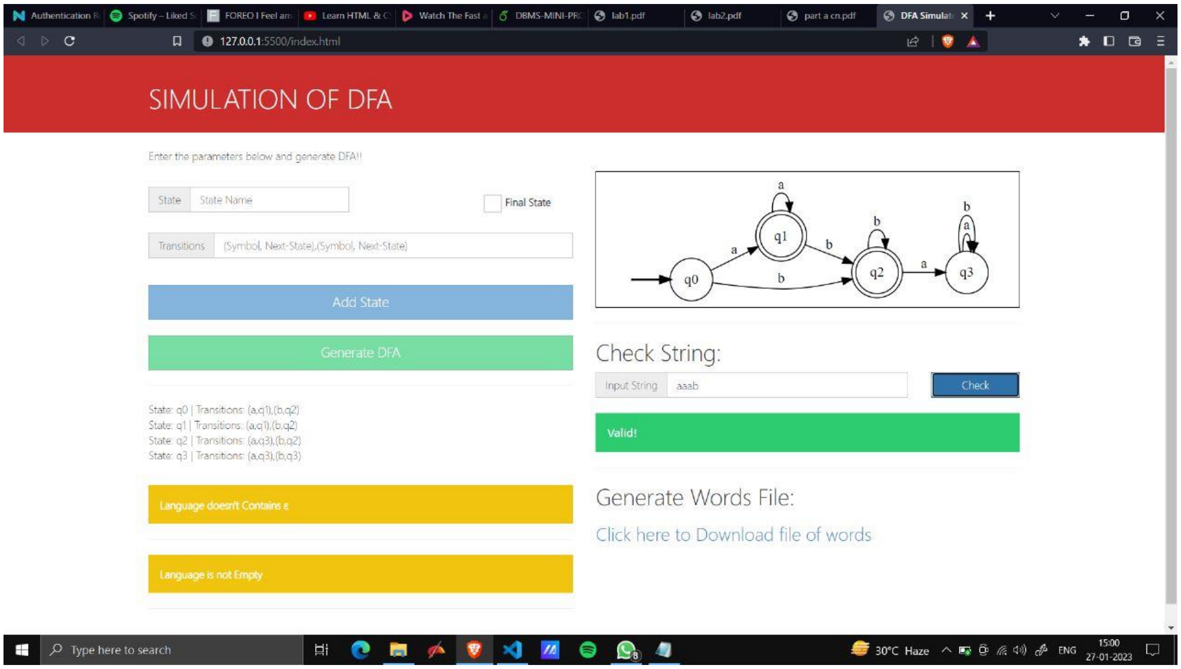
Input String: 100

Invalid!

Generate Words File:

5. Illustrate the design in step by step approach and write a program to simulate a DFSM which accept the language having all ‘a’ before all ‘b’. Analyze the output with different test cases.

   Output :



SIMULATION OF DFA

Enter the parameters below and generate DFA!!

State: q0 | Transitions: (a,q1),(b,q2)
State: q1 | Transitions: (a,q1),(b,q2)
State: q2 | Transitions: (a,q3),(b,q2)
State: q3 | Transitions: (a,q3),(b,q3)

Language doesn't Contains ε

Language is not Empty

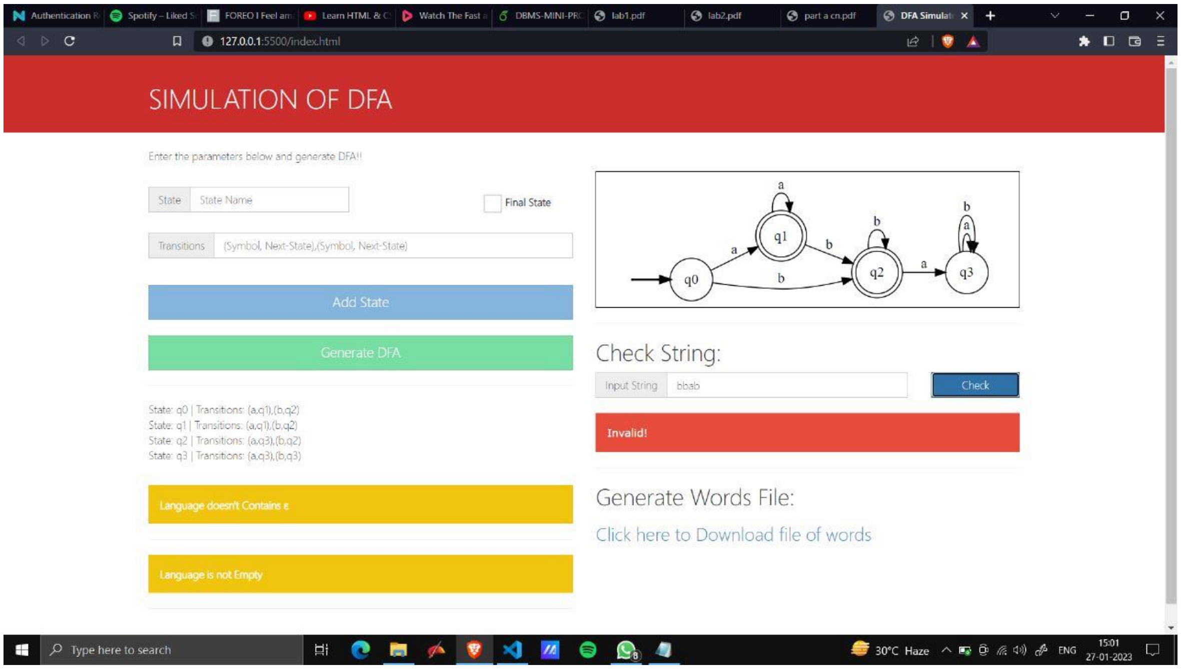
Check String:

Input String: aaab

Valid!

Generate Words File:

Click here to Download file of words



SIMULATION OF DFA

Enter the parameters below and generate DFA!!

State: q0 | Transitions: (a,q1),(b,q2)
State: q1 | Transitions: (a,q1),(b,q2)
State: q2 | Transitions: (a,q3),(b,q2)
State: q3 | Transitions: (a,q3),(b,q3)

Language doesn't Contains ε

Language is not Empty

Check String:

Input String: bbab

Invalid!

Generate Words File:

Click here to Download file of words