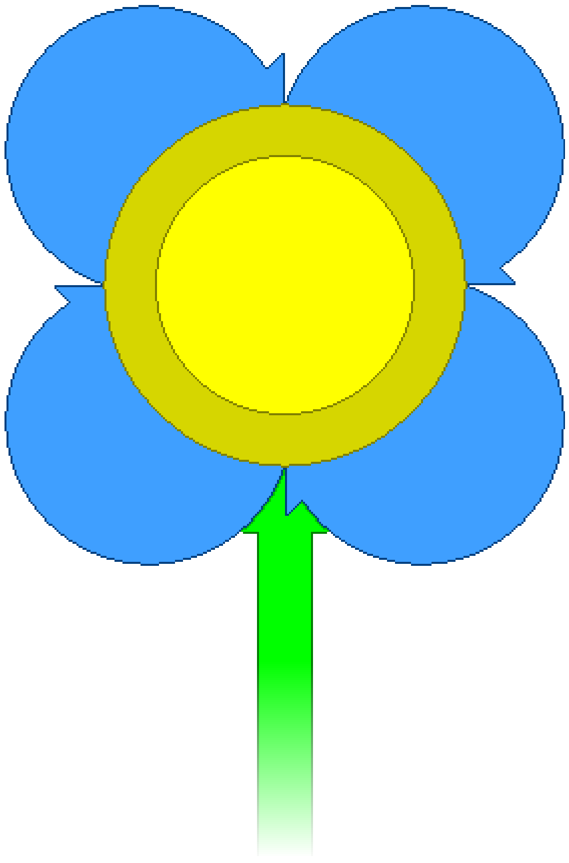
FSM Simulator



User’s Manual Document

Matt Bartyczak, Clayton Kinard, Nick Pierson, Jordan Tallent

April 30, 2015

USER'S MANUAL
TABLE OF CONTENTS

1.0 GENERAL INFORMATION

1.1 System Overview

The FSM Simulator allows the user to create and test a finite state machine (FSM) using a graphical interface.

The user may create a FSM by defining the Alphabet that the FSM will consider, creating states, and defining transitions between states. The user may also edit several aspects of the states, such as editing each state's name and setting whether each state should accept or reject an input string. The user may also set a single state as the FSM's initial state.

The FSM allows the user to test input strings by running them through the FSM. The user may simply choose to determine the acceptance or rejection of an input string. The user may also choose to run the input string through the FSM one character at a time, letting the FSM Simulator indicate which state the FSM is currently in after reading each character. In order to facilitate testing, the user may also add a number of input strings to a Test Set and test them all at once.

The FSM Simulator is accessible at http://www.radford.edu/npierson/fsm/fsmSimulator.html using any major web browser. Operating the user interface requires the use of a pointing device (such as a mouse) along with a device for entering text (such as a keyboard).

1.2 Points of Contact

1.2.1 Information

Dr. Edward Okie and Dr. Tracy Lewis-Williams of the Department of Information Technology at Radford University, Radford, Virginia, will maintain the FSM Simulator.

Dr. Edward G. Okie
Professor of Computer Science
Department of Information Technology
219 Davis Hall
Radford University
Radford VA, 24142-6933
www.radford.edu/nokie
nokie@radford.edu
540-831-5992 (Office)
540-831-6706 (FAX)

Dr. Tracy L. Lewis-Williams
Associate Professor of Computer Science
Department of Information Technology
229 Davis Hall
Radford University
Radford VA, 24142-6933
www.radford.edu/tlewis32
tlewis32@radford.edu
540-831-5358 (Office)
540-831-6706 (FAX)

1.3 Supporting Materials

1.3.1 External Software Used

A. Leray (2013) *PocketGrid* (Version 1.1.0) [Computer program]. Available: arnaudleray.github.io/pocketgrid/download/

JQuery UI (Version 1.11.4) [Computer program]. Available: jqueryui.com

1.3.2 References

Harris, *HTML, XHTML, and CSS All-in-One for Dummies*, 2nd ed. Hoboken, NJ: Wiley, 2011.

E. A. Rich, *Automata, Computability and Complexity: Theory and Applications*, 1st ed. Upper Saddle River, NJ: Pearson, 2008.

R. W. Sebesta, *Programming the World Wide Web*, 7th ed. Boston, MS: Pearson, 2013.

stackoverflow.com, 2015. [Online]. Available: stackoverflow.com

w3schools.com, 2015. [Online]. Available: www.w3schools.com

2.0 SYSTEM SUMMARY

2.1 System Configuration

The FSM Simulator is hosted on the Radford University servers, and is available at http://www.radford.edu/npierson/fsm/fsmSimulator.html. Accessing the system requires a web browser and access to the Radford University servers. The system requires an input device (such as a keyboard) and a point-and-click device (such as a mouse) for data input, and displays the user interface on a computer monitor (or other viewing device).

2.2 Data Flows

The FSM Simulator runs entirely on the user's web browser. As such, there is no need to store any data on a separate system. When the user enters information, such as creating an element of a FSM or testing an input string, that information is stored within the web browser and accessed as needed. The user interface is rendered using HTML and styled with CSS. The functionality is implemented in JavaScript and JQuery.

2.3 User Access Levels

Using the FSM Simulator does not require any special access on the part of the user, beyond access to the Radford University servers. No registration or login is required, and the system is free to use.

3.0 QUICK START GUIDE

3.1 System Menu

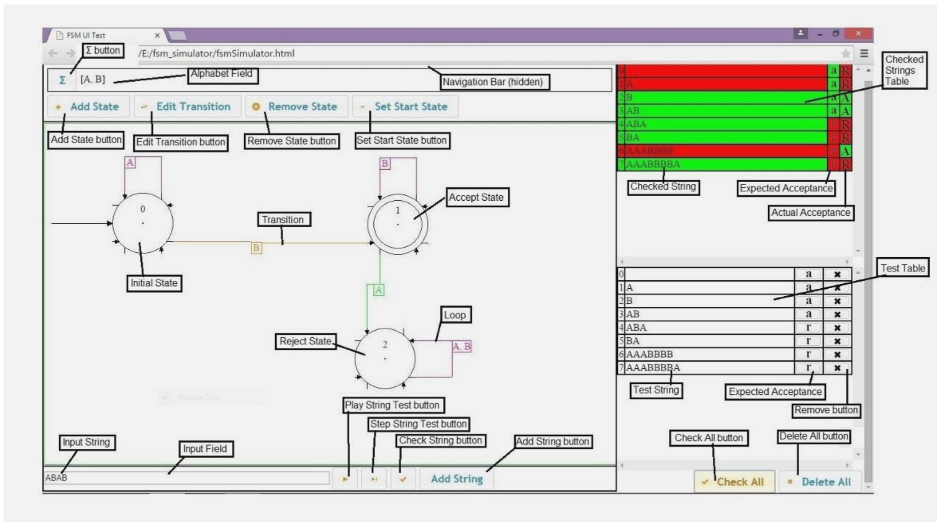


Figure 1. Elements of the User Interface

3.1.01 Adding Characters to the Alphabet

The user may enter the Alphabet, or set of characters, that the FSM will consider. To access the Edit Alphabet menu, the user should click on the Σ button in the upper-left corner of the screen.

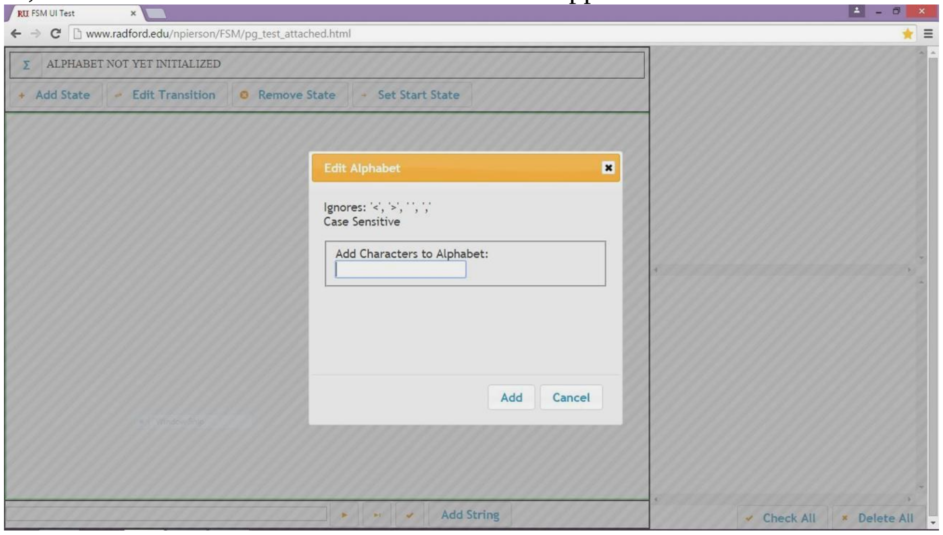

www.radford.edu/npierson/FSM/pg_test_attached.html
ALPHABET NOT YET INITIALIZED
Add State
Edit Transition
Remove State
Set Start State
Edit Alphabet
Ignores: '<', '>', ' ', ','
Case Sensitive
Add Characters to Alphabet:
Add
Cancel
Add String
Check All
Delete All

Figure 2. The Edit Alphabet menu

Once the Edit Alphabet menu is accessed (see Figure 2 above), the user may enter characters into the Add Characters to Alphabet field by first clicking on the text field and then entering characters via the keyboard. The user may enter any number of characters into the text field.

Note: Spaces (“ ”), commas (“,”), less-than signs (“<”), and greater-than signs (“>”) are not allowed as characters in the Alphabet.

Once all desired characters have been entered, the user may click the Add button to add all the input characters to the Alphabet. Clicking the Cancel button will cancel the action, and no characters will be added to the Alphabet. Once characters have been added to the Alphabet, the Alphabet will be visible in the Alphabet Field in the upper-left corner of the screen.

The Edit Alphabet menu should appear within 0.1 seconds after the user clicks the Σ button. The Edit Alphabet menu should disappear and any Alphabet characters should appear in the Alphabet Field within 0.1second after the user clicks the Add button or Cancel button.

Note: Characters may not be removed from the Alphabet once they are added. Doing so would invalidate the FSM by rendering any transitions triggered by the removed character(s) as invalid.

3.1.02 Adding a State

Along with transitions, states form the basis of the graphical depiction of the FSM. The user may add a state by clicking the Add State button, located just below the Σ Button.

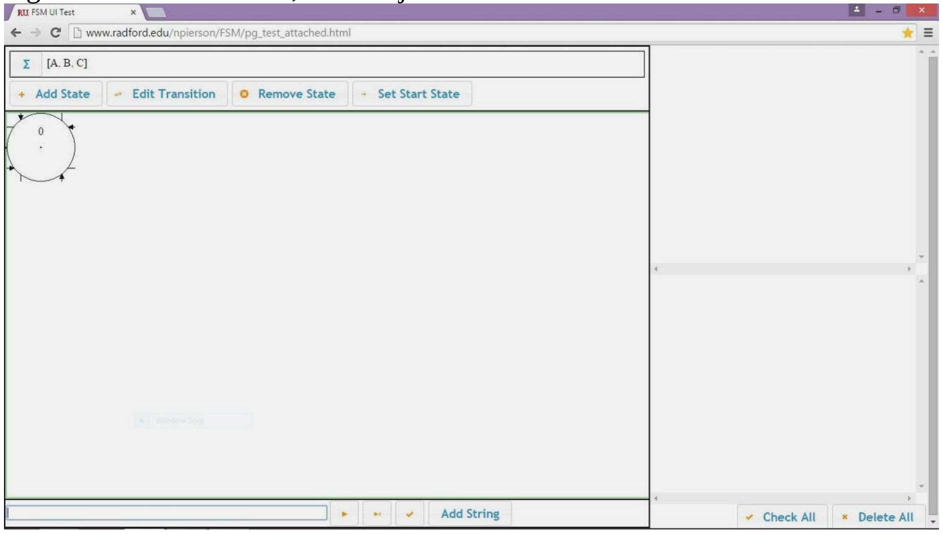


Figure 3. Adding a state

Once a state has been added, it will appear on the canvas. New states originally appear in the upper-left corner of the canvas (see Figure 3 above).

States appear as hollow circles with anchor-points around the perimeter. The name of the state appears in the within the state. By defaults, state names are numbers, indicating the order in which they were added to the FSM, starting at zero. In the figure above, the state is named “0”.

The new state should appear within 0.1 seconds after the user clicks the Add State button.

3.1.03 Moving a State

The user may move a state on the canvas by clicking-and-holding the cursor over the state, dragging the state to the desired location, and releasing the mouse button.

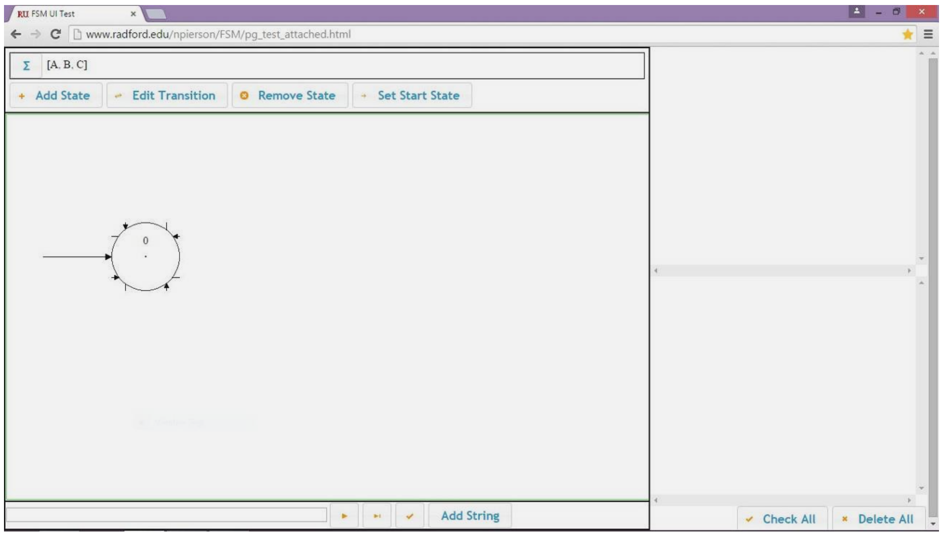


Figure 4. Moving a state

In the Figure 4 above, the state has been moved from the upper-left corner where it originated towards the center of the canvas.

There should be no more than a 0.1 seconds delay when moving a state.

3.1.04 Accessing the Edit State Menu

The user may edit a state via the state's Edit State menu (see Figure 5 below). To access a state's Edit State menu, double-click on the state.

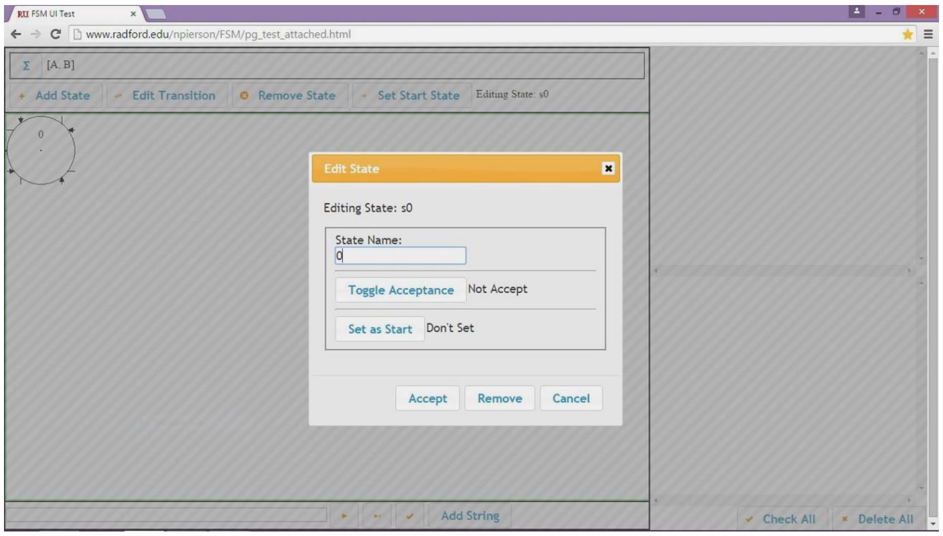


Figure 5. The Edit State menu

From the Edit State menu, the user may edit the state's name, toggle the state's acceptance, set a state as the FSM's start state, and remove the state from the FSM. These functions are described below.

The Edit State menu should appear within 0.1 seconds after the user double-clicks the state. The Edit State menu should be dismissed and any relevant changes should be visible within 0.1 seconds after the user clicks the Accept Button, the Remove Button, or the Cancel Button.

3.1.05 Setting a State’s Acceptance

The user may toggle whether a given state accepts or rejects an input string by clicking the Toggle Acceptance button on the Edit State menu (see Figure 5 above). The Edit State menu indicates the state’s current acceptance to the right of the Toggle Acceptance button by displaying “Not Accept” for reject states and Accept for accept states.

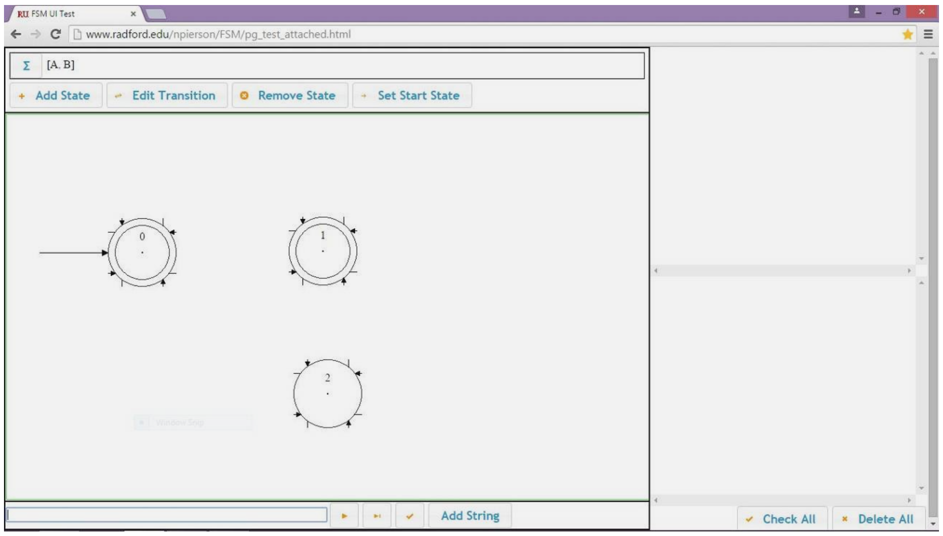


Figure 6. Accept and reject states

Single circles depict a reject state, while double circles depict an accept state. In Figure 6 above, states 0 and 1 are accept states, while state 2 is a reject state.

The state’s acceptance should be indicated (by either a single circle or a double circle) within 0.1 seconds after the Edit State menu is dismissed.

Note: The user may also toggle a state's acceptance by right-clicking on the state.

3.1.06 Changing a State’s Name

The state’s name is displayed in the center of the state. The user may change the name of a given state via the state’s Edit State menu (see Figure 5 above). The State Name field contains the state's name. The user may edit this name by clicking on the State Name field and editing the text via keyboard input. Once the desired name has been entered, the user may save the changes by clicking the Accept button, or discard the changes by clicking the Cancel button.

Note: The following characters are not allowed in a state name: Greater-Than (“>”), Less-Than (“<”).

3.1.07 Removing a State

The user may remove a state from the FSM via the state’s Edit State menu (see Figure 5 above). Clicking the Remove button on the Edit State menu will remove the state from the FSM, along with any transition leading to or from the removed state.
The state’s new name should appear inside the circle within 0.1 seconds after the Edit State menu is dismissed.

Note: States may also be removed from the main screen by clicking the Remove State button, located above the canvas (see Figure 1 above), and then clicking on the desired state.

3.1.08 Setting the FSM's Initial State

The initial state, or start state, is the state where the FSM will begin reading an input string. A black arrow pointing to a state indicates the initial state (see Figure 4 above).The user may define the initial state of the FSM via the state's Edit State menu (see Figure 5 above). Clicking the Set as Start button will set whether the selected state is the FSM's initial state.

The Edit State menu indicates whether the selected state will be set as the initial state by displaying a message to the right of the Set as Start button. The message "Set" indicates that the current state will be set as the initial state when the user clicks the Accept button, while the message "Don't Set" indicates that the current initial state will not be changed.

The arrow indicating the new initial state should be visible within 0.25seconds after the Edit State menu is dismissed.

A note about initial states: Every FSM should have exactly one initial state. FSMs without an initial state are invalid machines. Thus, the user may only assign another state as the initial state, and may not un-assign the current initial state. If the current initial state is removed from the FSM, the next state added will become the initial state. Attempting to run input strings on an invalid machine will result in an error message below the lower-right corner of the canvas. Attempting to test the Test Set on an invalid machine will have no effect.

Note: The initial state may also be set by clicking on the Set Start State button, located above the canvas (see Figure 1 above), and then clicking on the desired state.

3.1.09 Creating a Transition

Along with states, transitions form the basis of the graphical depiction of the FSM. The user may add a state by clicking the Edit Transition button, located above the canvas (see Figure 1 above). Once the Edit Transition button has been clicked, the transition is defined by clicking on the beginning state of the transition, then clicking on the ending state of the transition. Note that a transition may have the same state as its beginning and ending state, these transitions are known as "loops".

Once the beginning and ending states of a transition have been selected, the Edit Transition menu will appear (see Figure 7 below). From the Edit Transition menu, the user may define the characters that will trigger the transition by clicking on the Transition Characters field and entering the desired characters. Once the desired characters have been entered, the user may click the Accept Button to finish creating the transition, or click the Cancel button to discard the transition.

Transitions appear as arrows leading from one state to another, or back to the same state in the case of loops. A box connected to the transition contains the character(s) that trigger the transition. In Figure 8 below, a transition has been created from state 0 to state 1 that is triggered by the character B.

The new transition should be visible within 0.1 seconds after the Edit Transition menu is dismissed.

Some notes about entering Transition Characters: Only characters that are in the Alphabet may be assigned as Transition Characters. If any characters are entered into the Transition Characters field that are not in the Alphabet, the user will receive a prompt indicating that these characters were ignored.

Every transition should have at least one Transition Character; transitions with no Transition Characters will be discarded.

Some notes about states having multiple transitions for the same character: The FSM Simulator supports only *deterministic* FSMs. Thus, each state should only have one transition for each character. If a state is assigned multiple transitions for the same character, the FSM will only consider the most recently edited transition when reading an Input String. When a newly created transition conflicts with an existing transition, the user shall receive a prompt explaining the error and indicating that the FSM will only consider the most recently edited transition.

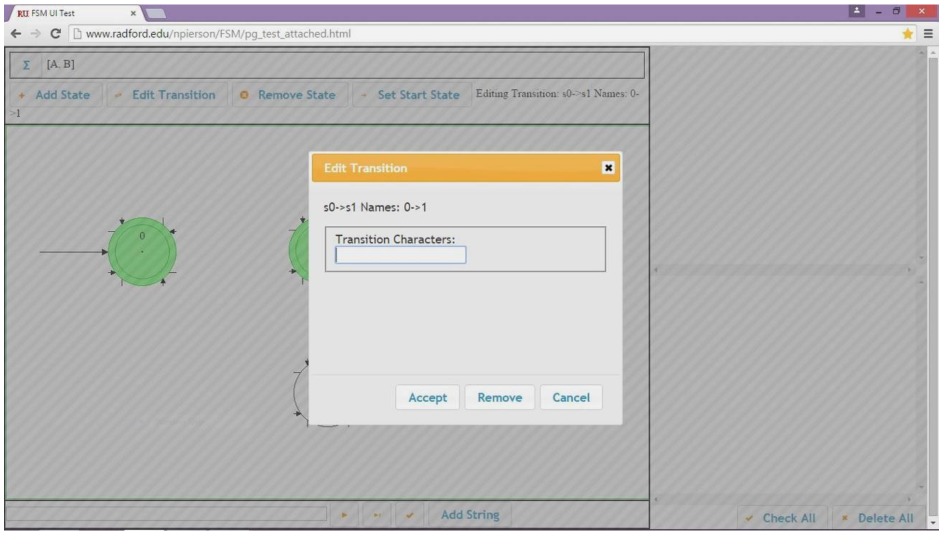


Figure 7. The Edit Transition menu

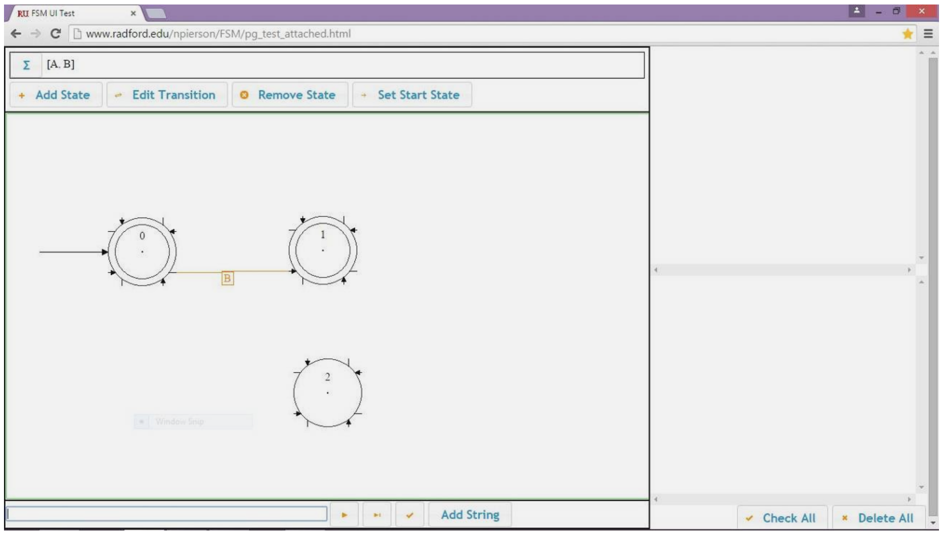


Figure 8. A Transition

3.1.10 Editing a Transition

The user may edit a transition via the state's Edit Transition menu (see Figure 7 above). To access a transition's Edit Transition menu, double-click on the character(s) that trigger the desired transition. This

Edit Transition menu functions identically to the one that is displayed when a transition is created (see 3.1.09 Creating a Transition) above.

The Edit Transition menu should appear within 0.1 seconds after the transition is double-clicked.

3.1.11 Removing a Transition

The user may remove a transition from the FSM via the Edit Transition menu (see Figure 7 above). Clicking the Remove button will remove the transition from the FSM. Deleting all the characters that trigger the transition and clicking the Accept button will also delete the transition.

The removed transition should no longer be visible within 0.1 seconds after the Edit Transition menu is dismissed.

3.1.12 Entering an Input String

The user may enter an input string that may be run through the FSM by clicking the Input Field below the canvas and entering the input string via the keyboard. In Figure 9 below, the input string "AAABBBB" has been entered into the Input Field.

The Input String should appear in the Input Field with no more than a 0.1 seconds delay from when the user entered it.

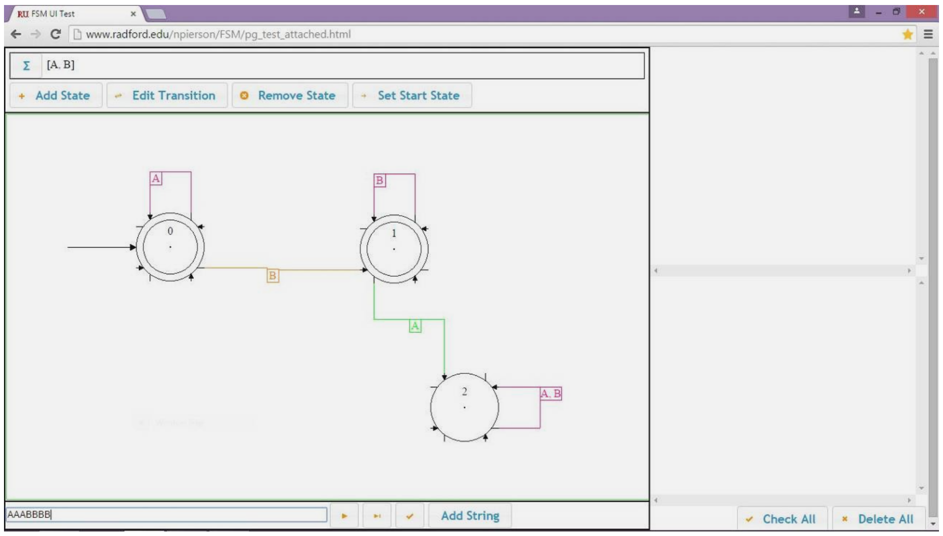


Figure 9. Entering an Input String

3.1.13 Iteratively Reading the Input String

Clicking on the Play String Test button, located immediately to the right of the Input Field, will cause the FSM to read the entire Input String, one character at a time. In order to depict the FSM reading the Input String, there will be a one-second delay between reading each character of the Input String.

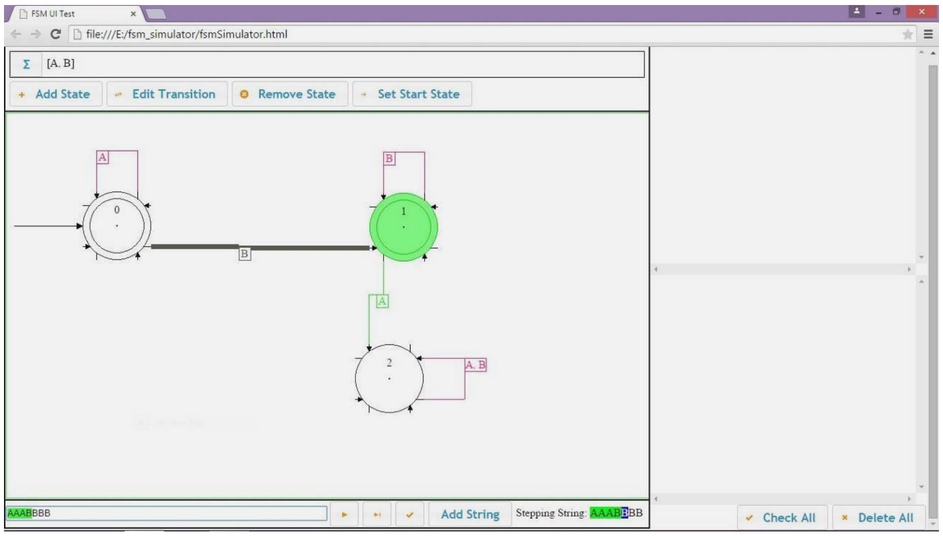


Figure 10. Reading an Input String

As the FSM reads the Input String, its current state will transition based on the most recently read character. The current state will be colored green, and a thick black line will indicate the transition that is taken (see Figure 10 above).

While the FSM is reading an Input String, a copy of the Input String is displayed just below the lower-right corner of the canvas. This copy indicates the progress of the FSM in reading the Input String. Characters that have already been read are highlighted in green, while the character currently being read is highlighted in blue.

After the FSM has read the final character of the Input String, the Input String copy will be replaced by text indicating whether the Input String was accepted or rejected. This text will either say "Accepted String: <*string*>" if the Input String was accepted, or else "Rejected String <*string*>" if the string was rejected.

In order to demonstrate the FSM's behavior as it reads the Input String, there is a 1-second delay between reading each input character.

A note on missing transitions: The FSM Simulator only simulates *deterministic* finite state machines. As such, each state should have exactly one transition for *every* character in the Alphabet. While reading an Input String, if the system encounters a character that the current state does not have a transition for, they system will stop reading the Input String, the current state will become red, and the system will indicate that the FSM rejected the Input String.

3.1.14 Iteratively Reading a Single Character of the Input String

The user may run a single character of the input string through the FSM by clicking the Step String Test button, located immediately to the right of the Play String Test button. Running a single character through the FSM behaves exactly as described in 3.1.13 Iteratively Reading the Input String above, except that only a single character is ran through the FSM. This feature may be used to read the entire Input String, one character at a time.

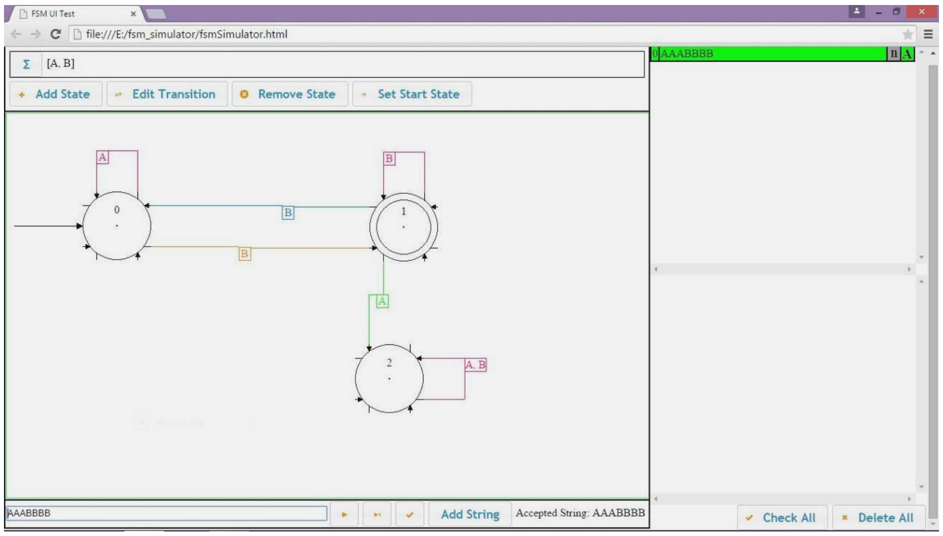
The new condition of the FSM should be rendered within 0.1 seconds after the user clicks the Step String Test button.

3.1.15 Checking an Input String

The user may check if the FSM accepts an input string by clicking the Check String button immediately to the right of the Step String Test button. This will indicate whether the Input String was accepted or rejected without displaying the FSM's progress as it reads the Input String. Text immediately to the right of the Add String button shall indicate whether the Input String was accepted or rejected. This text will either say "Accepted String: <*string*>" if the Input String was accepted, or else "Rejected String <*string*>" if the string was rejected. See Figure 11 below.

Also, note that checking the acceptance of an Input String in this manner will add the Input String to the Checked Strings table, described in 3.1.20 Testing the Test Strings below.

The acceptance of an Input String should be visible within 0.1 seconds after the user clicks the Check String button.



Accepted String: AAABBBB

Figure 11. Checking a String

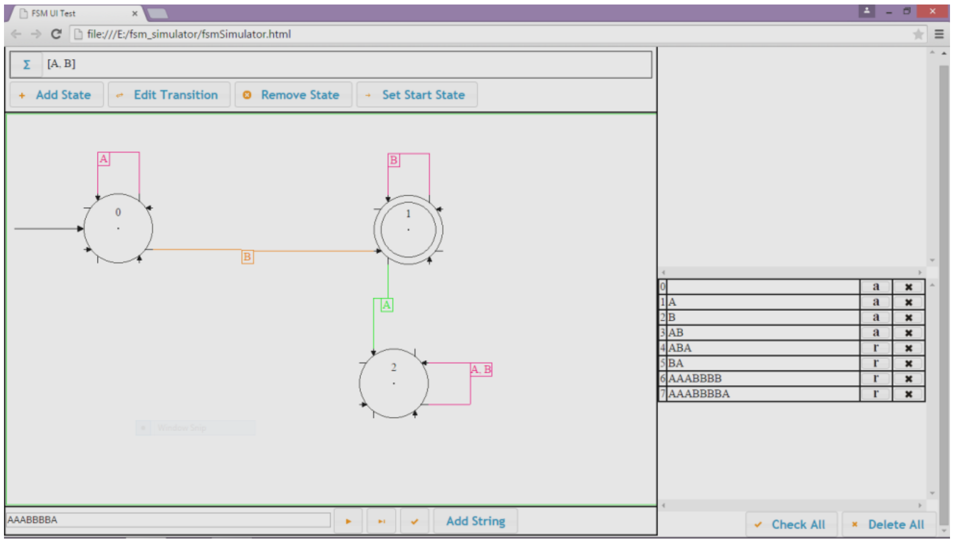
3.1.16 Adding an Input String to the Test Set

The FSM Simulator allows the user to test several Input Strings at once. The user may add the current Input String to the set of Test Strings by clicking the Add String button immediately to the right of the Check String button. Once a Test String has been added to the Test Set, the Test String will be displayed as a row in the Test Table located in the lower-right corner of the user interface.

The new Test String should appear in the Test Table within 0.1 seconds after the user clicks the Add String button.

Note that if the Test Set contains more Test Strings than can fit within the user interface, the FSM Simulator provides a vertical scroll bar immediately to the right of the Test Table. If any Test Strings are too long to fit within the user interface, there is a horizontal scroll bar immediately below the Test Table.

In Figure 12 below, several Test Strings have been added to the Test Set.



0		a	✖
1	A	a	✖
2	B	a	✖
3	AB	a	✖
4	ABA	r	✖
5	BA	r	✖
6	AAABBBB	r	✖
7	AAABBBBA	r	✖

Figure 12. The Test Set

3.1.17 Setting a Test String's Expected Acceptance

The user may set whether a Test String is expected to be accepted or rejected by the FSM. In the third column of the Test Table, immediately to the right of the Test String, is the Expected Acceptance cell of each Test String. This cell contains a single lower-case character that indicates whether the user expects the FSM to accept or reject the Test String. An "a" indicates expected acceptance, while an "r" indicates expected rejection. The user may toggle this value by clicking on the Expected Acceptance cell for a given Test String. In Figure 12 above, the top four Test Strings are expected to be accepted, while the bottom four Test Strings are expected to be rejected.

The character in the Expected Acceptance column should update within 0.1 seconds of when the user clicks it.

3.1.18 Removing a Test String

The user may remove a test string from the set of test strings by clicking the Test String's Remove button located in the right-most column of the Test Table. The Remove button is indicated by an "X".

The removed Test String should no longer be visible in the Test Table within 0.1 seconds after the user clicks the Remove button.

3.1.19 Removing All Test Strings

The user may remove all Test Strings from the Test Set by clicking the Delete All button located below the Test Table.

The Test Table should no longer be visible within 0.1 seconds after the user clicks the Delete All button.

3.1.20 Testing the Test Strings

The user may test the acceptance of all the Test Strings by clicking the Check All button located below the Test Table. This will check the Actual Acceptance of each Test String, as determined by the FSM, against the Test String's Expected Acceptance as set by the user in the Test Table. Each Test String will then be added to the Checked Strings table in the upper-right corner of the user interface.

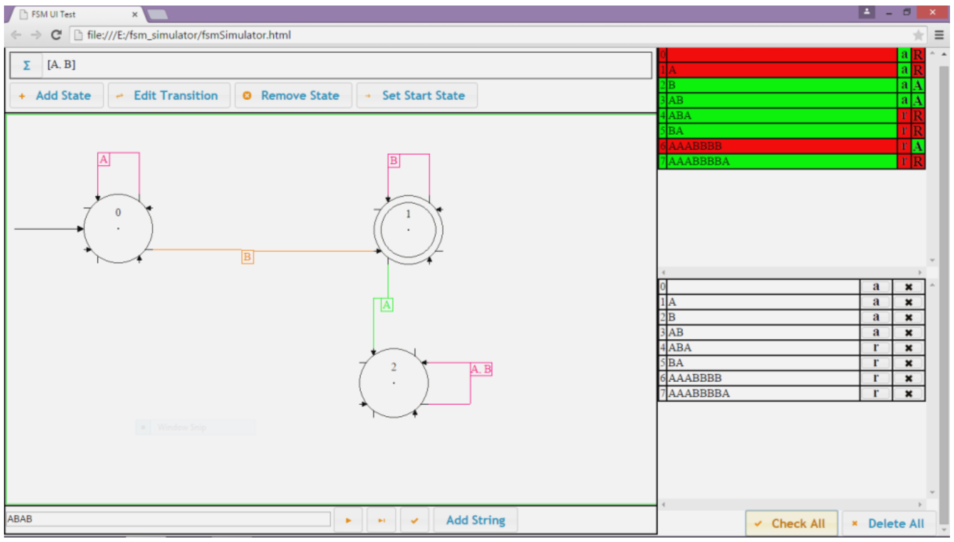
The right-most columns in the Checked Strings table indicate the Expected and Actual Acceptance of the Checked String. The right-most column indicates the Actual Acceptance, either with an upper-case “A” for acceptance or an upper-case “R” for rejection. The next column to the left indicates the Expected Acceptance, either with a lower-case “a” for acceptance or a lower-case “r” for rejection. In both columns, a green background also indicates acceptance while a red background also indicates rejection.

The background color of the Checked String itself indicates whether the Checked String’s Actual Acceptance matched its Expected Acceptance. A green background indicates that the Actual Acceptance matched the Expected Acceptance and that the Checked String passed the test. Meanwhile, a red background indicates that the Actual Acceptance did not match the Expected Acceptance and that the Checked String failed the test.

Note that Input Strings checked using the Check String button will also appear in the Checked Strings table, as mentioned in 3.1.15 Checking an Input String above. These Checked Strings will have a lower-case “n” with a grey background in their Expected Acceptance column, indicating that the user had not specified the string’s expected acceptance.

If the Test Strings are less than one-hundred characters long, the Checked Strings should begin appearing in the Checked Strings Table within 0.1 seconds after the user clicks the Check All button. There should be no more than a 0.1-second delay between Checked Strings appearing within the Checked Strings Table, again assuming that the strings being tested are less than one-hundred characters long.

Note that altering the FSM will cause all the cells in the Checked Strings Table to turn gray. This indicates that the results displayed in the Checked Strings Table are no longer valid.



0		a	R
1	A	a	R
2	B	a	A
3	AB	a	A
4	ABA	r	R
5	BA	r	R
6	AAABBBB	r	A
7	AAABBBBA	r	R

0		a	✖
1	A	a	✖
2	B	a	✖
3	AB	a	✖
4	ABA	r	✖
5	BA	r	✖
6	AAABBBB	r	✖
7	AAABBBBA	r	✖

Figure 13. The Checked Strings Table

3.1.21 Reloading a Previously-Tested String

The user may reload any Input String from the Checked Strings table back into the Input Field by double-clicking the Input String in the Checked Strings table.

The Checked String should appear within the Input Field within 0.1 second from when the user double-clicks it in the Checked Strings Table.

3.1.22 Accessing the Navigation Bar

The user may access the Navigation Bar by hovering the cursor over the top of the user interface (see Figure 14 below). The Navigation Bar allows access to the Tutorial page, the Future Features page, the Developers page, and the Client page.

The Navigation Bar should appear within 0.1 seconds after the user hovers the cursor over the top of the user interface.

Note: Navigating to any of the pages available on the Navigation Bar will navigate the browser to a new page. This will cause any information entered into the FSM Simulator (FSM, Alphabet characters, test strings, etc.) will be lost.

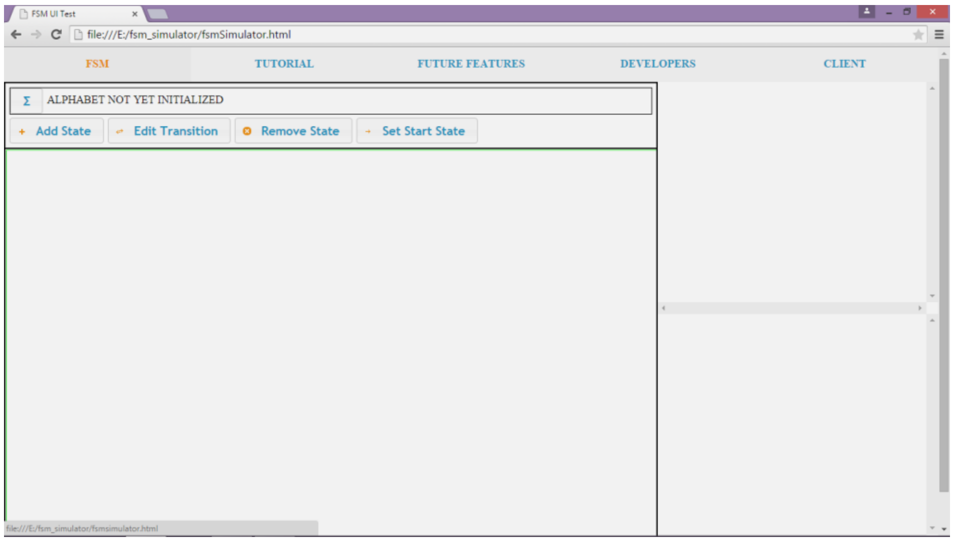


Figure 14. The Navigation Bar

3.1.23 Accessing the Tutorial Page

The user may access the Tutorial page by clicking the Tutorial Button on the Navigation Bar. This page features a video that the user may watch to become familiar with the basic features of the FSM Simulator. To view the video, click on the video image in the center of the Navigation page.

The Tutorial page should appear within 0.1 seconds after the user clicks the Tutorial button.

Note: Navigating to the Tutorial Page will navigate the browser away from the FSM Simulator. This will cause any information entered into the FSM Simulator (FSM, Alphabet characters, test strings, etc.) will be lost.

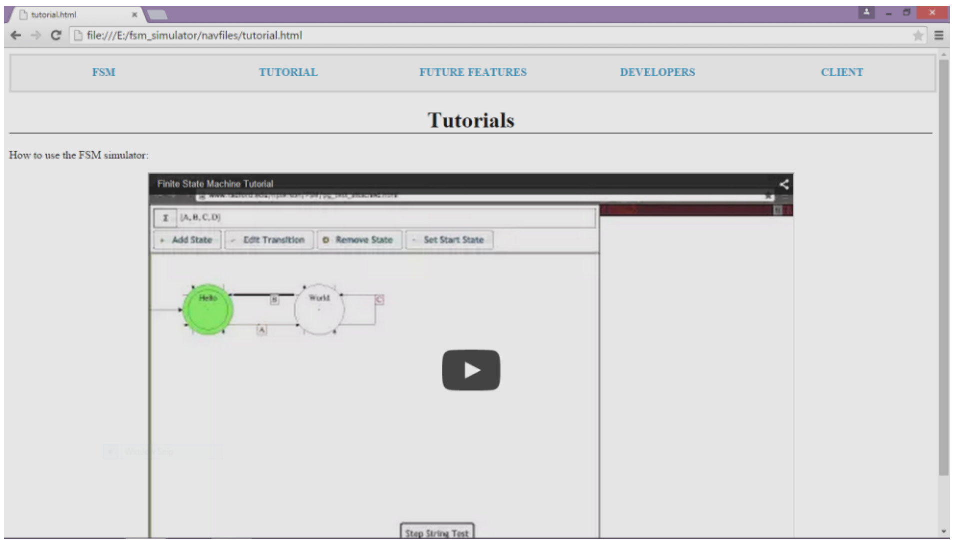


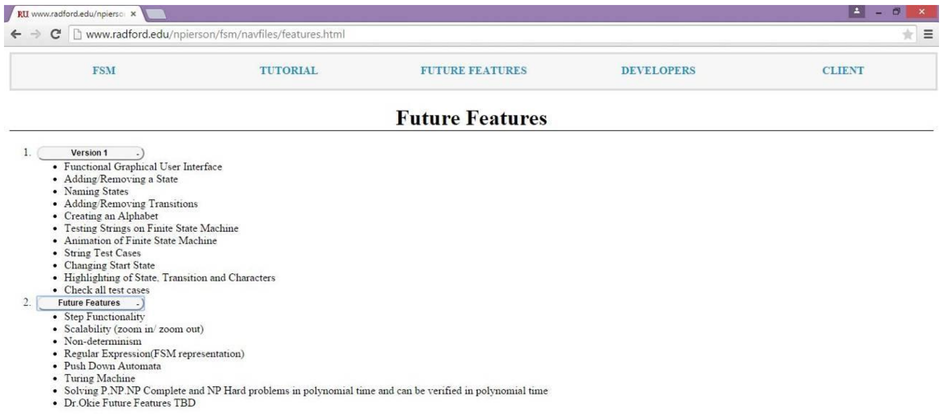
Figure 15. The Tutorial Page

3.1.24 Accessing the Future Features Page

The user may access the Future Features page by clicking the Future Features button on the Navigation Bar. This page features a list of the features added to each version of the FSM Simulator, along with features that the current developers would like to see added. The user may expand each list by clicking on the + symbol on each list's header.

The Future Features page should appear within 0.1 seconds after the user clicks the Future Features button.

Note: Navigating to the Future Features Page will navigate the browser away from the FSM Simulator. This will cause any information entered into the FSM Simulator (FSM, Alphabet characters, test strings, etc.) will be lost.



Future Features

1. Version 1
   - Functional Graphical User Interface
   - Adding/Removing a State
   - Naming States
   - Adding/Removing Transitions
   - Creating an Alphabet
   - Testing Strings on Finite State Machine
   - Animation of Finite State Machine
   - String Test Cases
   - Changing Start State
   - Highlighting of State, Transition and Characters
   - Check all test cases
2. Future Features
   - Step Functionality
   - Scalability (zoom in/ zoom out)
   - Non-determinism
   - Regular Expression(FSM representation)
   - Push Down Automata
   - Turing Machine
   - Solving P,NP,NP Complete and NP Hard problems in polynomial time and can be verified in polynomial time
   - Dr.Okie Future Features TBD

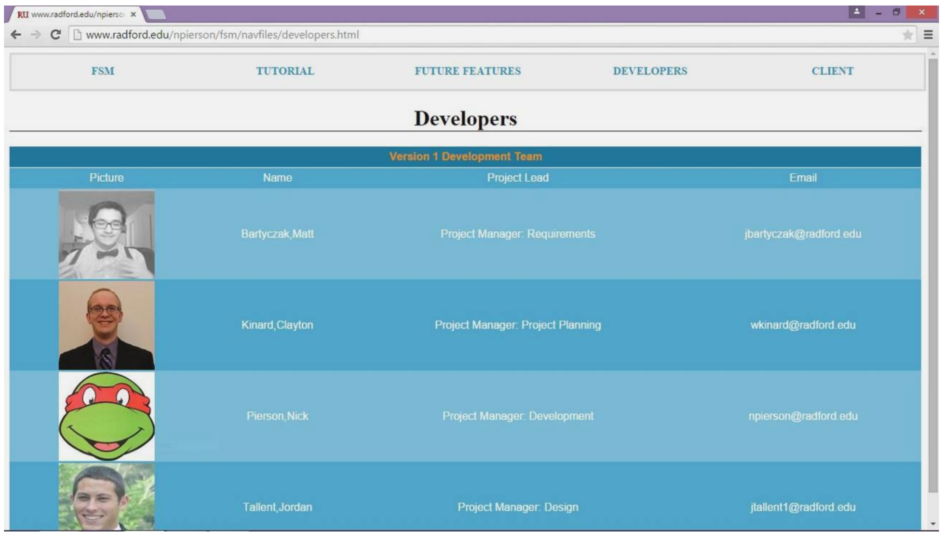
Figure 16. The Future Features Page

3.1.25 Accessing the Developers Page

The user may access the Developers page by clicking the Developers button on the Navigation Bar. This page has information on the current development team.

The Developers page should appear within 0.1 seconds after the user clicks the Developers button.

Note: Navigating to the Developers Page will navigate the browser away from the FSM Simulator. This will cause any information entered into the FSM Simulator (FSM, Alphabet characters, test strings, etc.) will be lost.



Developers

Version 1 Development Team

Picture	Name	Project Lead	Email
	Bartyczak,Matt	Project Manager: Requirements	jbartyczak@radford.edu
	Kinard,Clayton	Project Manager: Project Planning	wkinard@radford.edu
	Pierson,Nick	Project Manager: Development	npierson@radford.edu
	Tallent,Jordan	Project Manager: Design	jtallent1@radford.edu

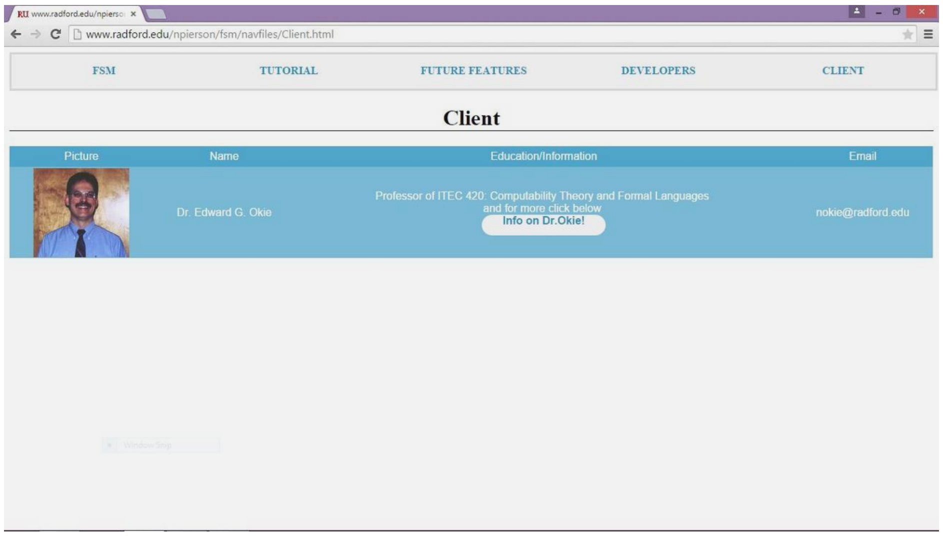
Figure 17. The Developers Page

3.1.26 Accessing the Client Page

The user may access the Client page by clicking the Client button on the Navigation Bar. This page has information on the client of the FSM Simulator project, Dr. Edward Okie. To access Dr. Okie’s webpage, click the Info on Dr. Okie! button.

The Client page should appear within 0.1 seconds after the user clicks the Client button.

Note: Navigating to the Client Page will navigate the browser away from the FSM Simulator. This will cause any information entered into the FSM Simulator (FSM, Alphabet characters, test strings, etc.) will be lost.



Client

Picture	Name	Education/Information	Email
	Dr. Edward G. Okie	Professor of ITEC 420: Computability Theory and Formal Languages and for more click below	nokie@radford.edu

Figure 18. The Client Page

3.1.27 Returning to the FSM Simulator

The user may return to the FSM Simulator by clicking the FSM Simulator button on the Navigation Bar.

The FSM Simulator page should appear within 0.1 seconds after the user clicks the FSM Simulator button.

Note: Navigating to the FSM Simulator Page will reload the FSM Simulator to its original state. This will cause any information entered into the FSM Simulator (FSM, Alphabet characters, test strings, etc.) will be lost.

3.2 Exiting the System

In order to exit the system, the user should simply navigate away from the webpage or close the web browser. Note that the FSM Simulator does not have any systematic means of saving the user's creations, and any information entered (FSM, Alphabet characters, test strings, etc.) will be lost upon exiting the system.

3.3 Special Instructions for Error Correction

There is no systematic way to remove a character from the Alphabet. This feature has been omitted because removing a character from the Alphabet could result in invalid transitions within the FSM and introduce computational errors to the system. Note that the Alphabet is the set of *available* characters that the FSM may consider. Thus, the user may ignore unwanted characters in the Alphabet by omitting those characters from Input Strings.

The FSM Simulator only simulates *deterministic* finite state machines. Thus, each state should have exactly one transition for *every* character in the Alphabet. While reading an Input String, if the system encounters a character that the current state does not have a transition for, they system will stop reading the Input String, the current state will become red, and the system will indicate that the FSM rejected the Input String. To avoid this situation, make sure that each state has one and only one transition for each character in the alphabet.

The FSM Simulator supports only *deterministic* FSMs. Thus, each state should only have one transition for each character. If multiple transitions exist on the same state for the same character, the FSM will consider only the most recently edited transition when reading Input Strings. When a newly created transition conflicts with an existing transition, the user shall receive a prompt explaining the error and indicating that the FSM will only consider the most recently edited transition when reading an Input String.

3.4 Caveats and Exceptions

There are some things to remember when using the FSM Simulator:

1. The FSM Simulator does not store data, neither on the user's computer nor on the RU servers. This means that all information entered by the user will be lost when the user navigates away from the FSM Simulator.
2. The FSM Simulator assumes "dead" states in the FSM. This means that attempting to transition from a state that does not have a transition defined on the current character will result in the FSM rejecting the input string. If this occurred while the FSM is iteratively reading the input string, the state that rejected the input string due to the missing transition will be highlighted in red.
3. Transitions require characters to trigger them in order to be meaningful within the FSM. Therefore, the user should define the Alphabet that the FSM will consider before creating any transitions.

4.0 FUTURE ENHANCEMENTS

4.1 Future Functionality

FSM's are the basis of several theoretical computing models, such as Push-down Automata and Turing Machines. As such, future developers could use the FSM Simulator as the basis for developing simulators for these and other types of theoretical computers.

4.2 Similar Systems – Additional Functionality

There are several features that the current developers wanted to include, but were unable to implement due to time constraints. These features include nondeterminism, the ability for a FSM to take multiple transitions at once. Another useful feature would have been the inclusion of a Step Back button, which could be used to un-read the most recently read character while iteratively reading the input string. Finally, the development team would have liked to include the ability to zoom in and out on the canvas, allowing for the creation of larger FSMs than are feasible with the current system.

The finite state machine simulator located at ivanzuzak.info is similar to the FSM Simulator. This system includes the ability to generate a FSM from a regular expression, as well as allowing for nondeterministic FSMs. The FSM Simulator would benefit from these features.

The Automaton Simulator available at www.cburch.com is very similar to the FSM Simulator, but with several additional features. This system allows for the creation of both deterministic and nondeterministic FSMs, along with Push-down Automata and Turing Machines. The Automaton Simulator draws transitions with much smoother lines that those found in the FSM Simulator. These features would make good additions to the FSM Simulator.

The SMCube simulator software available at erika.tuxfamily.org shares some similarities with the FSM Simulator. This system allows the user to save their FSMs in an XML file, which would be useful in the FSM Simulator.

4.3 Maintenance Capabilities

As new versions of HTML, CSS, and JavaScript are released, the FSM may need to be updated to account for any depreciated sections of the source code. The source code of the FSM Simulator may also need to be updated periodically to comply with the specification of future web browsers.