**PASSWORD VALIDATION USING AUTOMATA**

A PROJECT REPORT

(15CSE303 – Theory of Computation)

***Submitted by***

BL.EN.U4CSE18023 CH. YUVA TEJA

BL.EN.U4CSE18136 V.MANOJ SAIPRAKASH

BL.EN.U4CSE18043 I.YASWANTH

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

IN

COMPUTER SCIENCE AND ENGINEERING



SCHOOL OF ENGINEERING, BENGALURU CAMPUS

AMRITA VISHWA VIDYAPEETHAM

**BENGALURU 560 035**

November-2020

**AMRITA VISHWA VIDYAPEETHAM**

**SCHOOL OF ENGINEERING, BENGALURU CAMPUS**



**BONAFIDE CERTIFICATE**

This is to certify that the project/case study report (15CSE303 – Theory of Computation) entitled “**PASSWORD VALIDATION”** submitted by

“ REG.1:CSE18023 NAME: CH.YUVATEJA

REG.2:CSE18136 NAME:V.MANOJ SAIPRAKASH

REG.3:CSE18043 NAME:I.YASWANTH ” in partial fulfillment of the requirements for the award of the **Degree Bachelor of Technology** in “**COMPUTER SCIENCE AND ENGINEERING**“ is a bonafide record of the work carried out under my(our) guidance and supervision at Amrita School of Engineering, Bengaluru.

This project report was evaluated by us on ………….

Faculty EXAMINER I Faculty EXAMINER II

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **TITLE**  1.ABSTRACT      2.CHAPTER Ⅰ    3.BIBLIOGRAPHY | **INDEX**  1  3-4      5 |

**Abstract**

**Introduction**: **(1)**

* A strong password provides essential protection from financial fraud and identity theft.
* One of the most common ways that hackers break into computers is by guessing passwords. Simple and commonly used passwords enable intruders to easily gain access and control of a computing device.

**Key points of Password Security**

There are key points of password security that users must know in order to reduce the likelihood of a hacker cracking their password and thus gaining access to their device.

* Most importantly, passwords must be long and complex.
* Long and complex passwords require more effort and time for a hacker to guess.
* Passwords should have a combination of characters such as percent signs as well as upper-case and lower-case letters and numbers.

In many applications (Facebook, Instagram etc..) we have to secure our password and make our password strong so that no one can crack our password. so, for making our password strong I have created a model that work with the password validation using Finite automata.

**Problem description:**

Suppose a password contains a small letter, a capital letter, a number and a special character. We have to check whether user enter all of these types or not. If user enter all these types then his/her password become valid. Else it should be invalid.

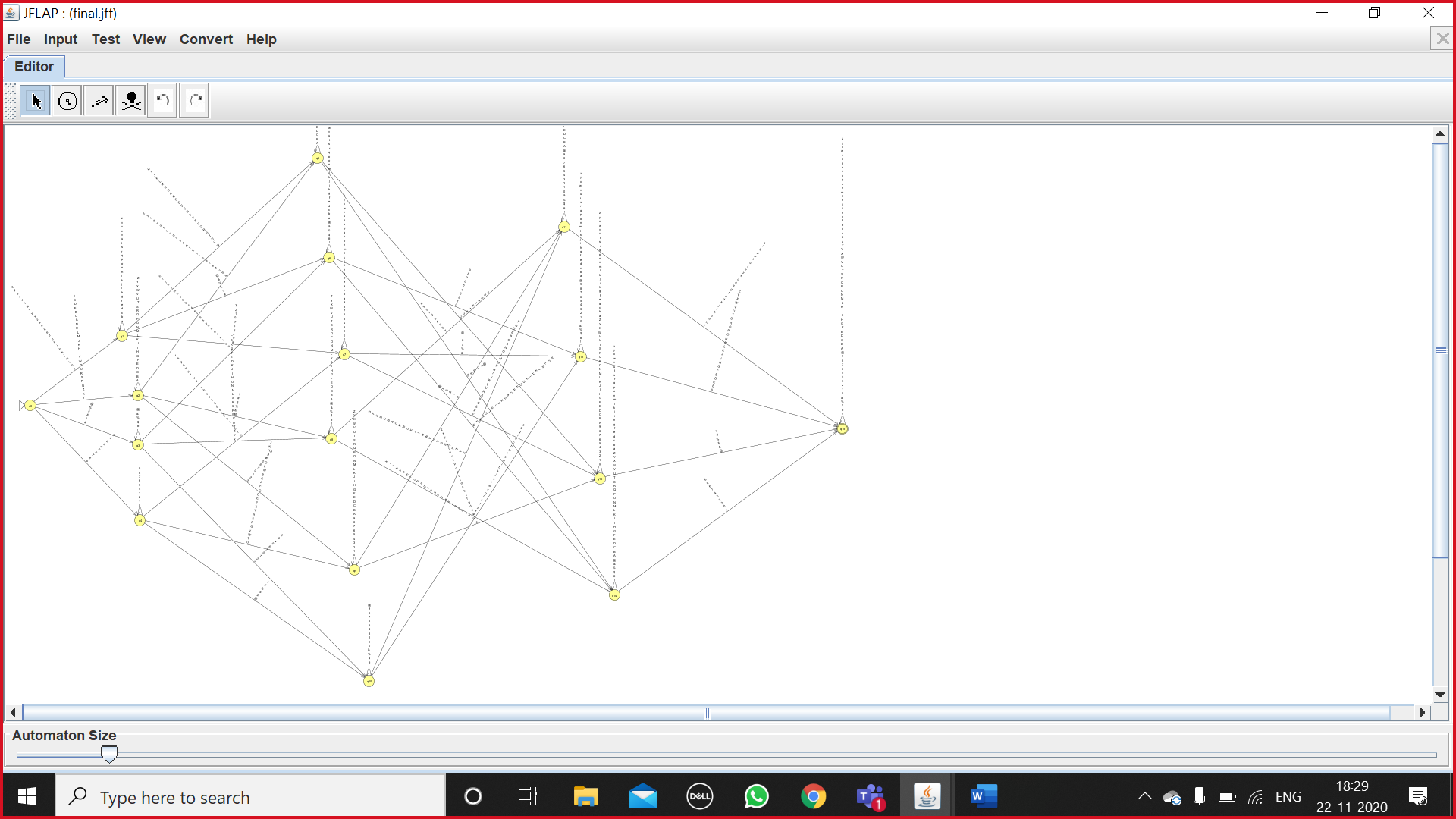
We have designed an automata like this.

Scope and limitations:

* For a password it will accept minimum string of length of 4.
* It will accept all uppercase, lowercase, numbers but for special characters it will only accept 6 (@, #, $, %, &, ! ) we can put all the special characters but we have designed automata limiting with this much only.

**Solution:**  **(2)**

Implementation diagram of automata



**CHAPTER 1 (3)**

There are total 16 states in this DFA, q0, q1, q2, q3, q4, q5, q6, q7, q8, q9, q10, q11, q12, q14, q15.

1. Q: finite set of states
2. ∑: finite set of the input symbol
3. q0: initial state
4. F:  final state
5. δ: Transition function

Q = {q0, q1, q2, q3, q4, q5, q6, q7, q8, q9, q10, q11, q12, q14, q15.}

∑ = {Uppercase, Lowercase, Numbers, some special characters(@, #, $, %, &, !)}

q0 = {q0}

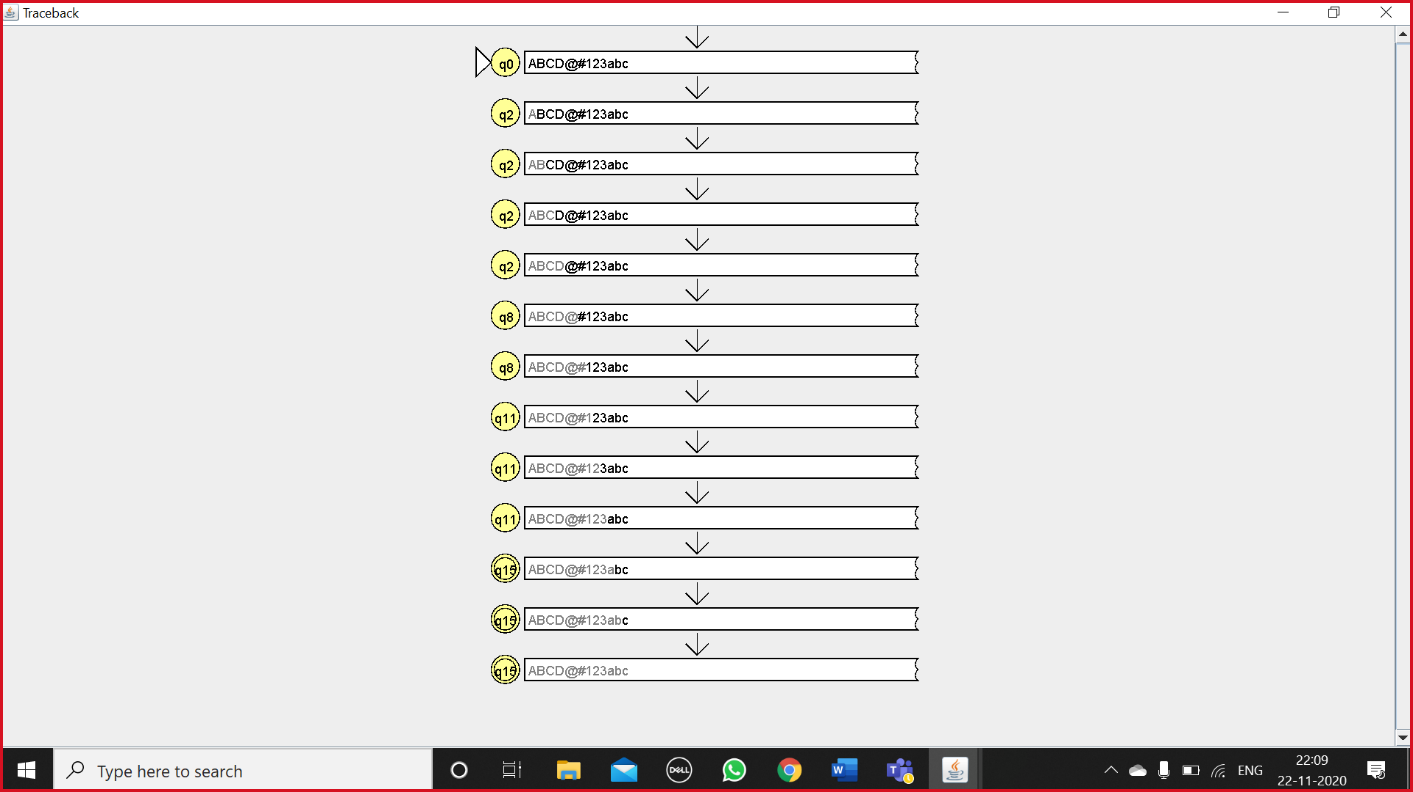
F = {q15}

Transition table**:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STATES | Nextstate for input upper case  (C) | Nextstate for input lowercase  (s) | Nextstate for input Number  (N) | nextstatefor input special characters(sc) |
| Q0 | Q2 | Q1 | Q4 | Q3 |
| Q1 | Q5 | Q1 | Q7 | Q6 |
| Q2 | Q2 | Q5 | Q9 | Q8 |
| Q3 | Q8 | Q6 | Q10 | Q3 |
| Q4 | Q9 | Q7 | Q4 | Q10 |
| Q5 | Q5 | Q5 | Q13 | Q14 |
| Q6 | Q14 | Q6 | Q12 | Q6 |
| Q7 | Q13 | Q7 | Q7 | Q12 |
| Q8 | Q11 | Q14 | Q8 | Q8 |
| Q9 | Q9 | Q13 | Q9 | Q11 |
| Q10 | Q11 | Q12 | Q10 | Q10 |
| Q11 | Q11 | Q15 | Q11 | Q11 |
| Q12 | Q15 | Q12 | Q12 | Q12 |
| Q13 | Q13 | Q13 | Q13 | Q15 |
| Q14 | Q14 | Q14 | Q15 | Q14 |
| \*Q15 | Q15 | Q15 | Q15 | Q15 |

**Sample test case: (4)**

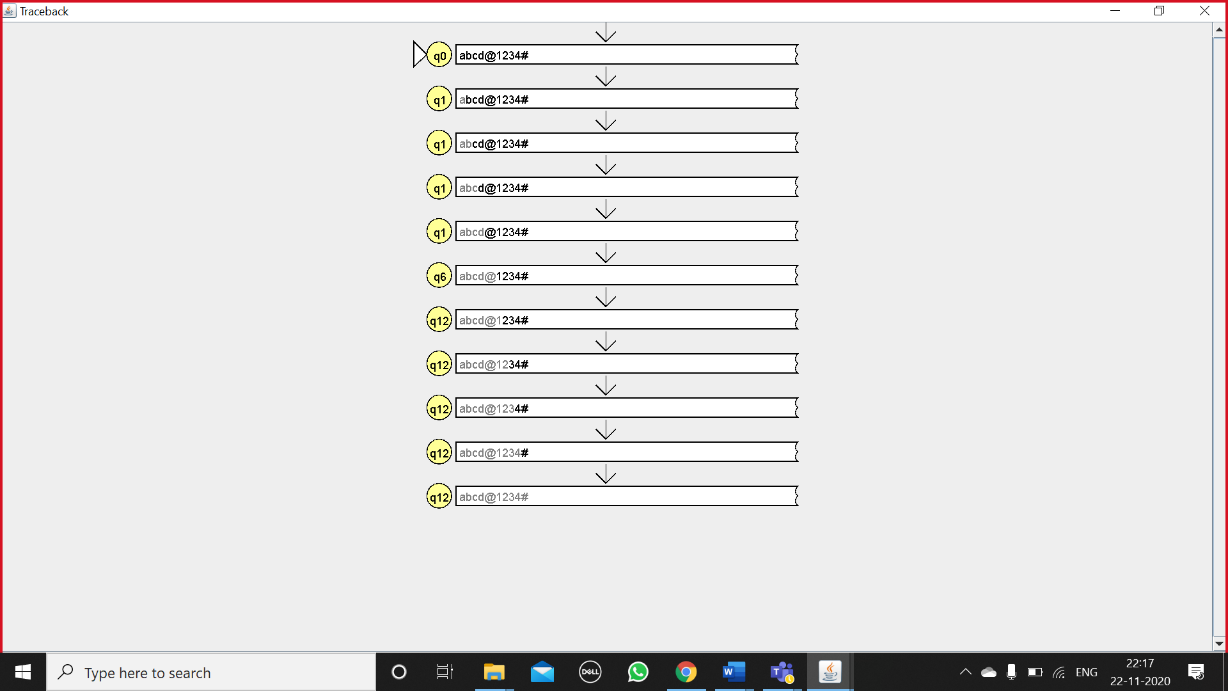
On input of **ABCD@#123abc**

****

It is accepted by automata because it contains uppercase, lowercase, numbers, special characters.

Sample test case 2:

On input abcd@1234# ( It is rejected because it doesn’t have uppercase letters.)



**BIBLIOGRAPHY: (5)**

1) https://www.geeksforgeeks.org/introduction-of-finite-automata/

2) https://www.javatpoint.com/deterministic-finite-automata