Laporan Tugas Besar

Teori Bahasa dan Automata

Mata Kuliah

Teori Bahasa dan Automata

Anggota Kelompok:

Maulana Ihsan - 1301204200
Zalfaa Putri Ayudhia - 1301200301
Balqis Fa'izah Shofura Rasdi - 1301204462

FAKULTAS INFORMATIKA
TELKOM UNIVERSITY
2022

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Manusia adalah makhluk sosial yang membutuhkan manusia lain, untuk bisa berkomunikasi dengan manusia lain maka dibutuhkan suatu bahasa yang dapat dimengerti. Dan bahasa yang digunakan oleh manusia adalah bahasa alami. Arti dari bahasa alami itu sendiri adalah suatu bahasa yang dikemukakan, ditulis dan diisyaratkan oleh manusia untuk berkomunikasi, bahasa alami juga memiliki aturan dan tata bahasanya sendiri.

Terdapat suatu bahasa yang tidak terdapat pembatasan tata bahasa dalam hasil produksinya. Bahasa ini diciptakan oleh para ilmuwan yang terinspirasi dengan adanya bahasa alami, para ilmuwan ini mengembangkan bahasa pemrograman dengan memberikan grammar kedalam bahasa tersebut secara formal, grammar ini diciptakan secara bebas-konteks dan pada akhirnya disebut dengan *Context Free Grammar*.

BAB II

PENYELESAIAN

2.1 *Context Free Grammar*

Pada tugas kali ini, mahasiswa diminta untuk mendefinisikan suatu *Context Free Grammar* (CFG) yang dapat merepresentasikan bahasa alami atau bahasa yang digunakan oleh manusia ke dalam aturan bahasa sederhana. Dan kelompok kami memilih untuk menggunakan bahasa Aceh yang memiliki struktur S-V-O (*subject-verb-object*).

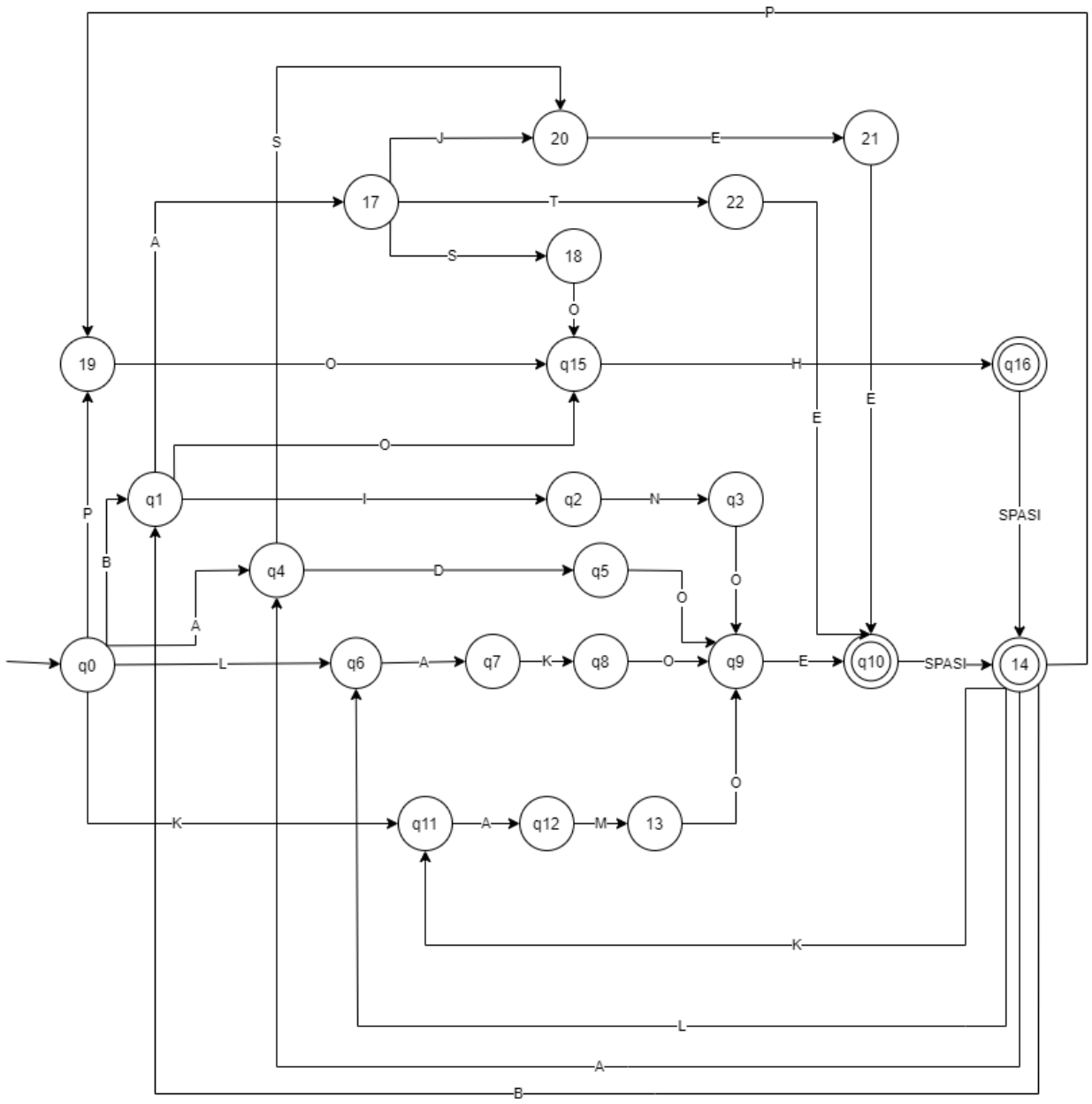
Berikut adalah *context free grammar* :
S → NN VB NN
NN → lakoe | kamoe | adoe | binoe | bajee | bate | asee
VB → boh | basoh | poh

Berikut adalah tabel pengelompokkan simbol terminal dan non-terminal, serta arti dari bahasa Aceh ke bahasa Indonesia.

Simbol Non-Terminal	Bahasa Aceh	Bahasa Indonesia
NN	Lakoe	Suami
NN	Kamoe	Saya
NN	Adoe	Adik
NN	Binoe	Istri
VB	Boh	Meletakkan
VB	Basoh	Mencuci
VB	Poh	Memukul
NN	Bajee	Baju
NN	Bate	Batu
NN	Asee	Anjing

2.2 Finite Automata

Pada rancangan finite automa ini terdapat 22 state dengan q0 sebagai state awal dan 3 accepted state yaitu q10,q14 dan q16. Finite automata ini digunakan sebagai rancangan untuk memvalidasi kata-kata dari tiap huruf dari kata yang sudah ditentukan. Berikut ini adalah rancangan finite automata yang sudah kami buat untuk setiap katanya.



Atau file bisa didownload melalui link berikut ini : disini

BAB III
PROGRAM

3.1 Lexical Analyzer

Lexical analyzer merupakan program yang digunakan untuk memvalidasi setiap kata yang telah ditentukan. Kami menggunakan bahasa python sebagai bahasa pemrograman yang akan digunakan untuk membuat program lexical analyzer. Berikut adalah program code python lexical analyzer berdasarkan Finite Automata yang telah kami buat sebelumnya :

```
import string

#objek  = lakoe|kamoe|adoe|binoe|bajee|bate|asee
#verb   = Boh|basoh|poh

#input example
print('=====================================================')
print('objek  = lakoe|kamoe|adoe|binoe|bajee|bate|bateueng')
print('verb   = Boh|basoh|poh')
print('=====================================================')
print('==========BUATLAH KALIMAT UNTUK DI UJIKAN============')
print('=====================================================')
print('KALIMAT :')
sentence =input()
print('=====================================================')
input_string = sentence.lower()+'#'

#initialization
alphabet_list = list(string.ascii_lowercase)
state_list =
['q0','ql','q2','q3','q4','q5','q6','q7','q8','q9','ql0','q11',

'q12','q13','q14','q15','q16','q17','q18','q19','q20','q21',
             'q22']

transition_table={}

for state in state_list:
    for alphabet in alphabet_list:
         transition_table[(state,alphabet)]='error'
    transition_table[(state,'#')]='error'
    transition_table[(state,' ')]='error'

#spaces sebelum imputan string
```

```
transition_table['q0',' ']='q0'

#--------------------INISIALISASI SUBJEK---------------------#
#inisialisasi token:binoe
transition_table[('q0','b')]='q1'
transition_table[('q1','i')]='q2'
transition_table[('q2','n')]='q3'
transition_table[('q3','o')]='q9'
transition_table[('q9','e')]='q10'
transition_table[('q10',' ')]='q14'
transition_table[('q10','#')]='accept'
#inisialisasi token:adoe
transition_table[('q0','a')]='q4'
transition_table[('q4','d')]='q5'
transition_table[('q5','o')]='q9'
transition_table[('q9','e')]='q10'
#inisialisasi token:lakoe
transition_table[('q0','l')]='q6'
transition_table[('q6','a')]='q7'
transition_table[('q7','k')]='q8'
transition_table[('q8','o')]='q9'
transition_table[('q9','e')]='q10'
#inisialisasi token:kamoe
transition_table[('q0','k')]='q11'
transition_table[('q11','a')]='q12'
transition_table[('q12','m')]='q13'
transition_table[('q13','o')]='q9'
transition_table[('q9','e')]='q10'
#--------------------INISIALISASI VERP-----------------------#
#inisialisasi token:poh
transition_table[('q0','p')]='q19'
transition_table[('q19','o')]='q15'
transition_table[('q15','h')]='q16'
transition_table[('q16',' ')]='q14'
transition_table[('q16','#')]='accept'
#inisialisasi token:boh
transition_table[('q0','b')]='q1'
transition_table[('q1','o')]='q15'
transition_table[('q15','h')]='q16'
#inisialisasi token:basoh
transition_table[('q0','b')]='q1'
transition_table[('q1','a')]='q17'
transition_table[('q17','s')]='q18'
transition_table[('q18','o')]='q15'
```

```
transition_table[('q15','h')]='q16'
#--------------------INISIALISASI objek-----------------------#
#inisialisasi token:bajee
transition_table[('q0','b')]='q1'
transition_table[('q1','a')]='q17'
transition_table[('q17','j')]='q20'
transition_table[('q20','e')]='q21'
transition_table[('q21','e')]='q10'
#inisialisasi token:bate
transition_table[('q0','b')]='q1'
transition_table[('q1','a')]='q17'
transition_table[('q17','t')]='q22'
transition_table[('q22','e')]='q10'
#inisialisasi token:asee
transition_table[('q0','a')]='q4'
transition_table[('q4','s')]='q20'
transition_table[('q20','e')]='q21'
transition_table[('q21','e')]='q10'
#--------------------INISIALISASI LOOPING--------------------#
transition_table[('q14','k')]='q11'
transition_table[('q14','l')]='q6'
transition_table[('q14','a')]='q4'
transition_table[('q14','b')]='q1'
transition_table[('q14','p')]='q19'

idx_char= 0
state ='q0'
current_token =' '
while state!='accept':
   current_char = input_string[idx_char]
   current_token += current_char
   state = transition_table[(state,current_char)]
   if (state == 'q10')or (state == 'q16'):
       print('current.Loken: ',current_token,', valid')
       current_token=' '
   if state =='error':
       print('KATA TIDAK TERDETEKSI')
       break;

   idx_char=idx_char+1

#conclusion.
if state =='accept':
    print('=====================================================')
```

```
    print('KALIMAT: ',sentence,'(VALID)')
    print('=====================================================')
```

Program code python dapat dilihat disini.

Berikut adalah program code implementasi pada web javascript lexical analyzer berdasarkan finite automata yang dibuat:

```
const alpha = Array.from(Array(26)).map((e, i) => i + 65);
const alphabetList = alpha.map((x) =>
String.fromCharCode(x).toLowerCase());
const stateList = [
  "q0",
  "q1",
  "q2",
  "q3",
  "q4",
  "q5",
  "q6",
  "q7",
  "q8",
  "q9",
  "q10",
  "q11",
  "q12",
  "q13",
  "q14",
  "q15",
  "q16",
  "q17",
  "q18",
  "q19",
  "q20",
  "q21",
  "q22",
];

let transitionTable = {};

stateList.forEach((state) => {
  alphabetList.forEach((alphabet) => {
    transitionTable[[state, alphabet]] = "ERROR";
  });
```

```
  transitionTable[[state, "#"]] = "ERROR";
  transitionTable[[state, " "]] = "ERROR";
});

// starting node space
transitionTable[["q0", " "]] = "q0";

// Finish State
transitionTable[["q10", "#"]] = "ACCEPT";
transitionTable[["q14", "#"]] = "ACCEPT";
transitionTable[["q16", "#"]] = "ACCEPT";


// binoe
transitionTable[["q0", "b"]] = "q1";
transitionTable[["q1", "i"]] = "q2";
transitionTable[["q2", "n"]] = "q3";
transitionTable[["q3", "o"]] = "q9";
transitionTable[["q9", "e"]] = "q10";
transitionTable[["q10", " "]] = "q14";


// adoe
transitionTable[["q0", "a"]] = "q4";
transitionTable[["q4", "d"]] = "q5";
transitionTable[["q5", "o"]] = "q9";
transitionTable[["q9", "e"]] = "q10";

// lakoe
transitionTable[["q0", "l"]] = "q6";
transitionTable[["q6", "a"]] = "q7";
transitionTable[["q7", "k"]] = "q8";
transitionTable[["q8", "o"]] = "q9";
transitionTable[["q9", "e"]] = "q10";

// kamoe
transitionTable[["q0", "k"]] = "q11";
transitionTable[["q11", "a"]] = "q12";
transitionTable[["q12", "m"]] = "q13";
transitionTable[["q13", "o"]] = "q9";
transitionTable[["q9", "e"]] = "q10";

// poh
transitionTable[["q0", "p"]]  = "q19";
```

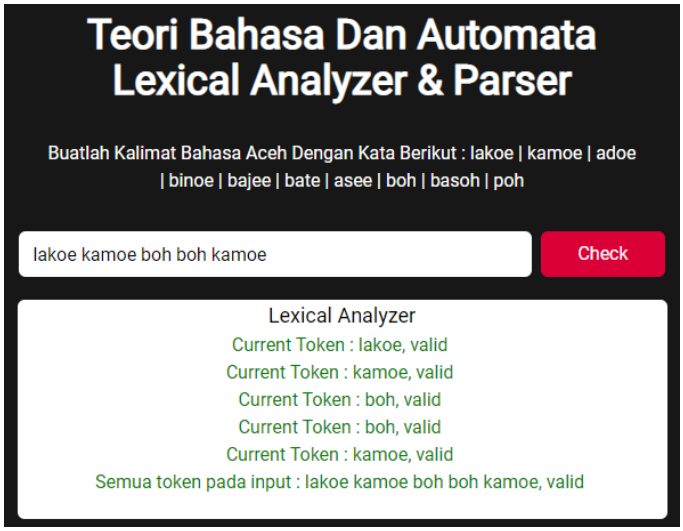
```
transitionTable[["q19", "o"]] = "q15";
transitionTable[["q15", "h"]] = "q16";
transitionTable[["q16", " "]] = "q14";
transitionTable[["q16", "#"]] = "ACCEPT";

// boh
transitionTable[["q0", "b"]] = "q1";
transitionTable[["q1", "o"]] = "q15";
transitionTable[["q15","h"]] = "q16";


// basoh
transitionTable[["q0", "b"]] = "q1";
transitionTable[["q1", "a"]] = "q17";
transitionTable[["q17", "s"]] = "q18";
transitionTable[["q18", "o"]] = "q15";
transitionTable[["q18", "h"]] = "q16";

// bajee
transitionTable[["q0", "b"]] = "q1";
transitionTable[["q1", "a"]] = "q17";
transitionTable[["q17", "j"]] = "q20";
transitionTable[["q20", "e"]] = "q21";
transitionTable[["q21", "e"]] = "q10";

// bate
transitionTable[["q0", "b"]] = "q1";
transitionTable[["q1", "a"]] = "q17";
transitionTable[["q17", "t"]] = "q22";
transitionTable[["q22", "e"]] = "q10";

// asee
transitionTable[["q0", "a"]] = "q4";
transitionTable[["q4", "s"]] = "q20";
transitionTable[["q20", "e"]] = "q21";
transitionTable[["q21", "e"]] = "q10";

//looping
transitionTable[["q14", "k"]] = "q11";
transitionTable[["q14", "l"]] = "q6";
transitionTable[["q14", "a"]] = "q4";
transitionTable[["q14", "b"]] = "q1";
transitionTable[["q14", "p"]] = "q19";
```

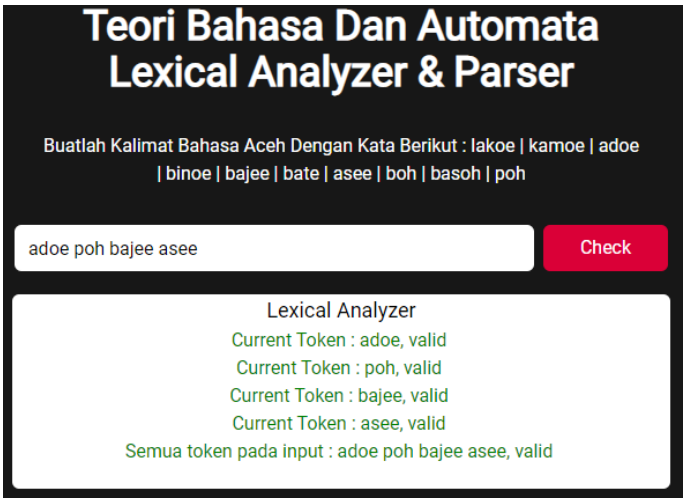
```
const checkSentence = (sentence) => {
  // lexical analysis
  let resultLa = document.getElementById("resultLa");
  let laTitle = document.getElementById("laTitle");
  let parserTitle = document.getElementById("parserTitle");
  let resultParser = document.getElementById("resultParser");
  let bgResult = document.getElementById("bgResult");
  bgResult.classList.remove("hidden");
  laTitle.className = "block font-medium text-lg";
  parserTitle.className = "hidden";
  resultLa.innerText = "";
  resultParser.innerText = "";

  let inputString = sentence.toLowerCase() + "#";
  let idxChar = 0;
  let state = "q0";
  let currentToken = "";
  let currentChar = "";
  while (state != "ACCEPT") {

    currentChar = inputString[idxChar];
    console.log(state,inputString[idxChar],currentChar !="#")
    // Error handling
    if (
      currentChar != " " &&
      currentChar != "#" &&
      !alphabetList.includes(currentChar)
    ) {
      console.log("error");
      resultLa.innerText += "ERROR";
      resultLa.style.color = "red";
      break;
    }

    currentToken += currentChar;
    state = transitionTable[[state, currentChar]];
    console.log(transitionTable[state])
    if (state == "q10" || state == "q16" ) {
      resultLa.innerText =
        resultLa.innerText + "Current Token : " + currentToken + ",
valid";
      resultLa.innerText += "\n";
      currentToken = "";
```

```
    }
    else if (state == "ERROR") {
      console.log("ERROR adwawd")
      resultLa.innerText += "ERROR";
      resultLa.style.color = "red";
      break;
    }
    idxChar++;
  }

  if (state == "ACCEPT") {
    resultLa.innerText += "Semua token pada input : " + sentence +
", valid";
    resultLa.style.color = "green";


let form = document.getElementById("form");

const handleSubmit = (e) => {
  let sentence = document.getElementById("sentence").value;
  checkSentence(sentence);
  e.preventDefault();
};

form.addEventListener("submit", (e) => handleSubmit(e));
```

3.2 Hasil Running Program dengan Token Valid

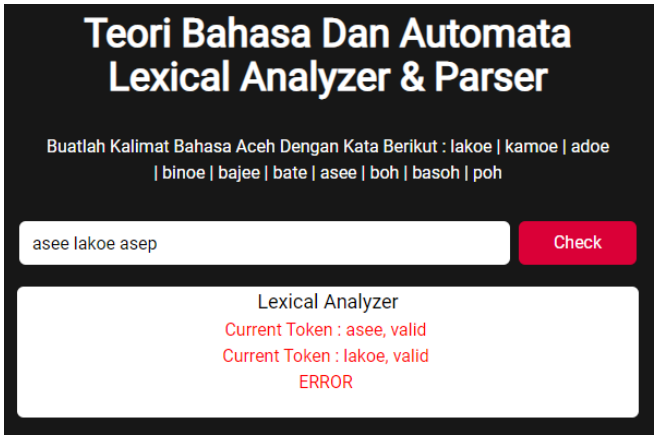
Pada tahap ini user akan diminta untuk memasukkan sebuah kalimat dimana setiap kata dari kalimat tersebut akan di lakukan pengujian apakah kata tersebut valid.

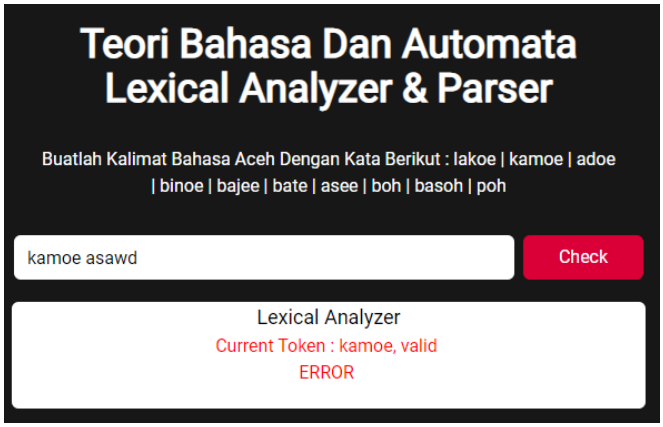


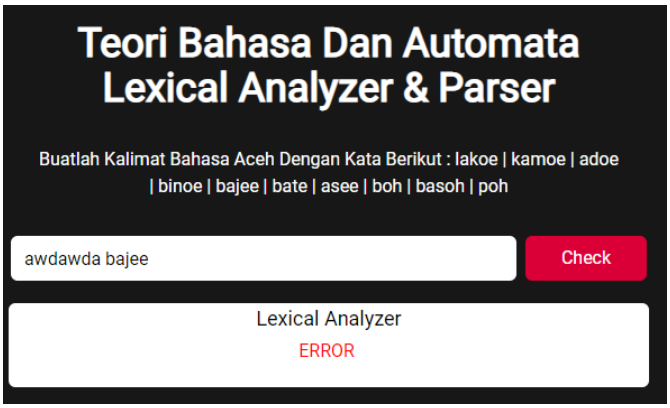


3.3 Hasil Running Program dengan Token Tidak Valid

Pada tahap ini user akan diminta untuk memasukkan sebuah kalimat dimana setiap kata dari kalimat tersebut akan di lakukan pengujian apakah kata tersebut tidak valid.







BAB IV
PARSE TABLE

4.1 Tabel Parser

Parse adalah serangkaian perintah pada program dan dipisahkan menjadi komponen yang lebih mudah diproses, yang dianalisis untuk sintaks yang benar dan kemudian dilampirkan ke tag yang menentukan setiap komponen.

Parsing adalah memecah sebuah kalimat atau kelompok kata menjadi komponen yang terpisah, termasuk definisi fungsi atau bentuk setiap bagian yang dapat lebih mudah untuk dimengerti. Pada tugas kali ini, kami akan melakukan parsing dengan menggunakan parse tabel.

Berikut adalah parse table dari kata-kata yang sudah ditentukan sebelumnya oleh kelompok kami :

	Lakoe	Kamoe	Adoe	Binoe	Bajee	Bate	Asee	Boh	Basoh	Poh	EOS
S	NN VB NN	NN VB NN	NN VB NN	NN VB NN	NN VB NN	NN VB NN	NN VB NN	Error	Error	Error	Error
NN	Lakoe	Kamoe	Adoe	Binoe	Bajee	Bate	Asee	Error	Error	Error	Error
VB	Error	Error	Error	Error	Error	Error	Error	Boh	Basoh	Poh	EOS

4.2 Program Parser

Parser adalah komponen compiler atau juru bahasa yang memecah data menjadi elemen yang lebih kecil untuk memudahkan dalam proses penerjemahan ke bahasa lain. Parser mengambil input dalam bentuk urutan token atau instruksi program dan biasanya akan membangun struktur data dalam bentuk pohon parse atau pohon sintaksis abstrak.

Pada program parser ini dilakukan untuk memvalidasi susunan kalimat yang sesuai dengan context free grammar yang sudah dibuat. Program ini akan memberikan keluaran valid dan grammar yang dihasilkan benar apabila susunan inputan kata adalah NN-VB-NN, dimana NN adalah kata benda dan VB adalah kata kerja. Apabila tidak sesuai dengan urutan tersebut program akan memberikan keluaran tidak valid dan grammar tidak diterima. Kami menggunakan Javascript untuk digunakan pada program parser kami. Berikut ini adalah code program untuk program parser.

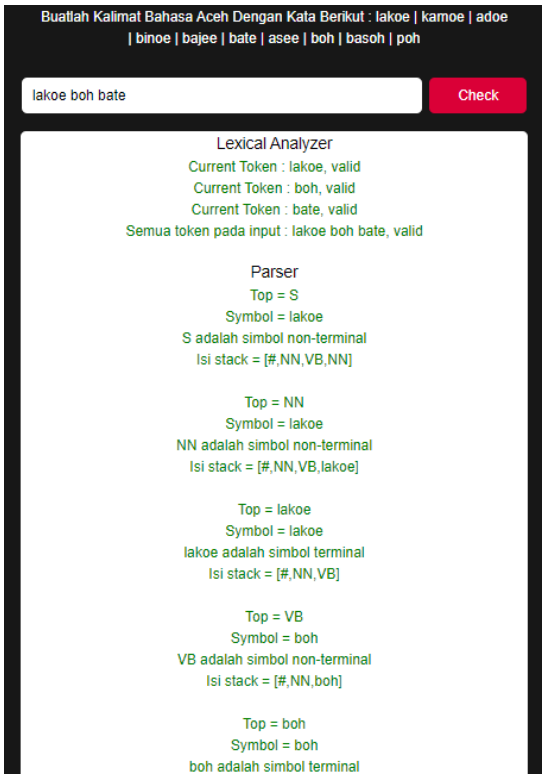
```
if (state == "ACCEPT") {
    resultLa.innerText += "Semua token pada input : " + sentence +
", valid";
```

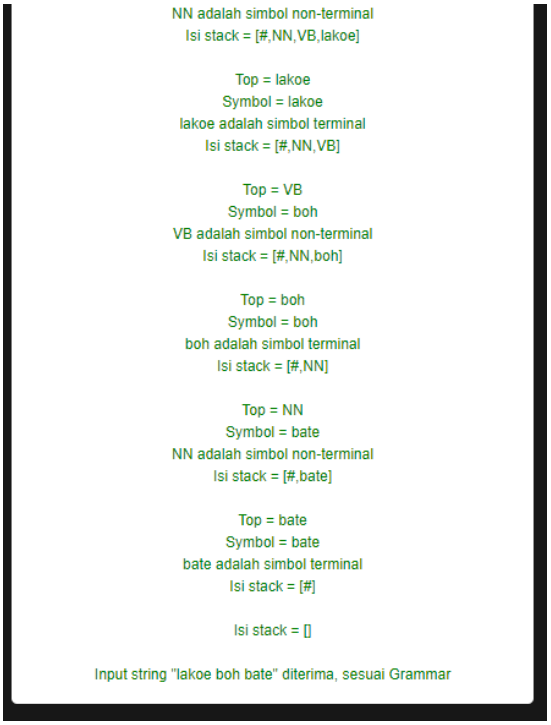
```
    resultLa.style.color = "green";

    // Parser
    parserTitle.className = "block font-medium text-lg";
    sentence = sentence.replace(/\s+/g, " ").trim();
    let tokens = sentence.toLowerCase().split(" ");
    tokens.push("EOS");

    // Symbol definition
    let nonTerminals = ["S", "NN", "VB"];
    let terminals = [
      "lakoe",
      "kamoe",
      "adoe",
      "binoe",
      "bajee",
      "bate",
      "asee",
      "boh",
      "basoh",
      "poh",
    ];

    // Parse Table
    let parseTable = {};

    //  kolom S
    parseTable[["S", "lakoe"]] = ["NN", "VB", "NN"];
    parseTable[["S", "kamoe"]] = ["NN", "VB", "NN"];
    parseTable[["S", "adoe"]] = ["NN", "VB", "NN"];
    parseTable[["S", "binoe"]] = ["NN", "VB", "NN"];
    parseTable[["S", "bajee"]] = ["NN", "VB", "NN"];
    parseTable[["S", "bate"]] = ["NN", "VB", "NN"];
    parseTable[["S", "asee"]] = ["NN", "VB", "NN"];
    parseTable[["S", "boh"]] = ["error"];
    parseTable[["S", "basoh"]] = ["error"];
    parseTable[["S", "poh"]] = ["error"];
    parseTable[["S", "EOS"]] = ["error"];

    // kolom NN
    parseTable[["NN", "lakoe"]] = ["lakoe"];
    parseTable[["NN", "kamoe"]] = ["kamoe"];
    parseTable[["NN", "adoe"]] = ["adoe"];
    parseTable[["NN", "binoe"]] = ["binoe"];
    parseTable[["NN", "bajee"]] = ["bajee"];
    parseTable[["NN", "bate"]] = ["bate"];
    parseTable[["NN", "asee"]] = ["asee"];
    parseTable[["NN", "boh"]] = ["error"];
    parseTable[["NN", "basoh"]] = ["error"];
    parseTable[["NN", "poh"]] = ["error"];
    parseTable[["NN", "EOS"]] = ["error"];
```

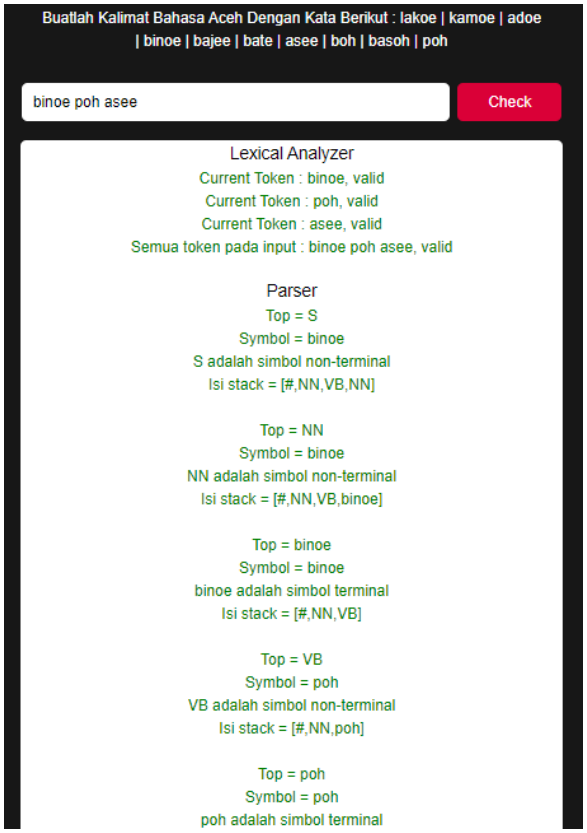
```
    // kolom VB
    parseTable[["VB", "lakoe"]] = ["error"];
    parseTable[["VB", "kamoe"]] = ["error"];
    parseTable[["VB", "adoe"]] = ["error"];
    parseTable[["VB", "binoe"]] = ["error"];
    parseTable[["VB", "bajee"]] = ["error"];
    parseTable[["VB", "bate"]] = ["error"];
    parseTable[["VB", "asee"]] = ["error"];
    parseTable[["VB", "boh"]] = ["boh"];
    parseTable[["VB", "basoh"]] = ["basoh"];
    parseTable[["VB", "poh"]] = ["poh"];
    parseTable[["VB", "EOS"]] = ["error"];


    // Inisialisasi stack
    let stack = [];
    stack.push("#");
    stack.push("S");

    // Input reading initialization
    let idxToken = 0;
    let symbol = tokens[idxToken];

    // parsing proses
    while (stack.length > 0) {
      let top = stack[stack.length - 1];
      resultParser.innerText = resultParser.innerText + "Top = " +
top + "\n";
      resultParser.innerText =
        resultParser.innerText + "Symbol = " + symbol + "\n";
      if (terminals.includes(top)) {
        resultParser.innerText =
          resultParser.innerText + top + " adalah simbol terminal
\n";
        if (top == symbol) {
          stack.pop();
          idxToken++;
          symbol = tokens[idxToken];
          if (symbol == "EOS") {
            resultParser.innerText =
              resultParser.innerText +
              "Isi stack = " +
              "[" +
              stack +
              "]" +
              "\n \n";
            stack.pop();
          }
        } else {
          resultParser.innerText = resultParser.innerText + "error
```

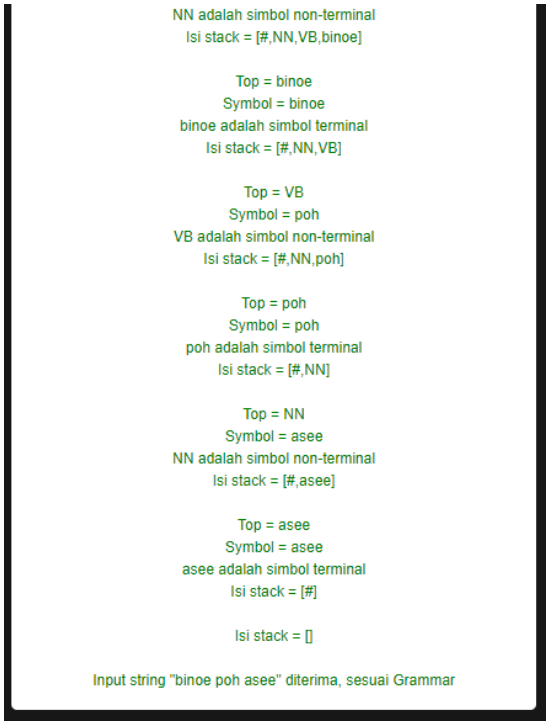
```
\n \n";
          break;
        }
      } else if (nonTerminals.includes(top)) {
        resultParser.innerText =
          resultParser.innerText + top + " adalah simbol
non-terminal \n";
        if (parseTable[[top, symbol]][0] != "error") {
          stack.pop();
          let symbolToBePushed = parseTable[[top, symbol]];
          for (let i = symbolToBePushed.length - 1; i > -1; i--) {
            stack.push(symbolToBePushed[i]);
          }
        } else {
          resultParser.innerText = resultParser.innerText + "error
\n \n";
          break;
        }
      } else {
        resultParser.innerText = resultParser.innerText + "error \n
\n";
        break;
      }
      resultParser.innerText =
        resultParser.innerText + "Isi stack = " + "[" + stack + "]"
+ "\n \n";
    }

    // Conclusion
    if (symbol == "EOS" && stack.length == 0) {
      resultParser.innerText =
        resultParser.innerText +
        'Input string "' +
        sentence +
        '" diterima, sesuai Grammar \n';
      resultParser.style.color = "green";
    } else {
      resultParser.innerText =
        resultParser.innerText +
        'Error, input string "' +
        sentence +
        '" tidak diterima, tidak sesuai Grammar \n';
      resultParser.style.color = "red";
    }
  }
```

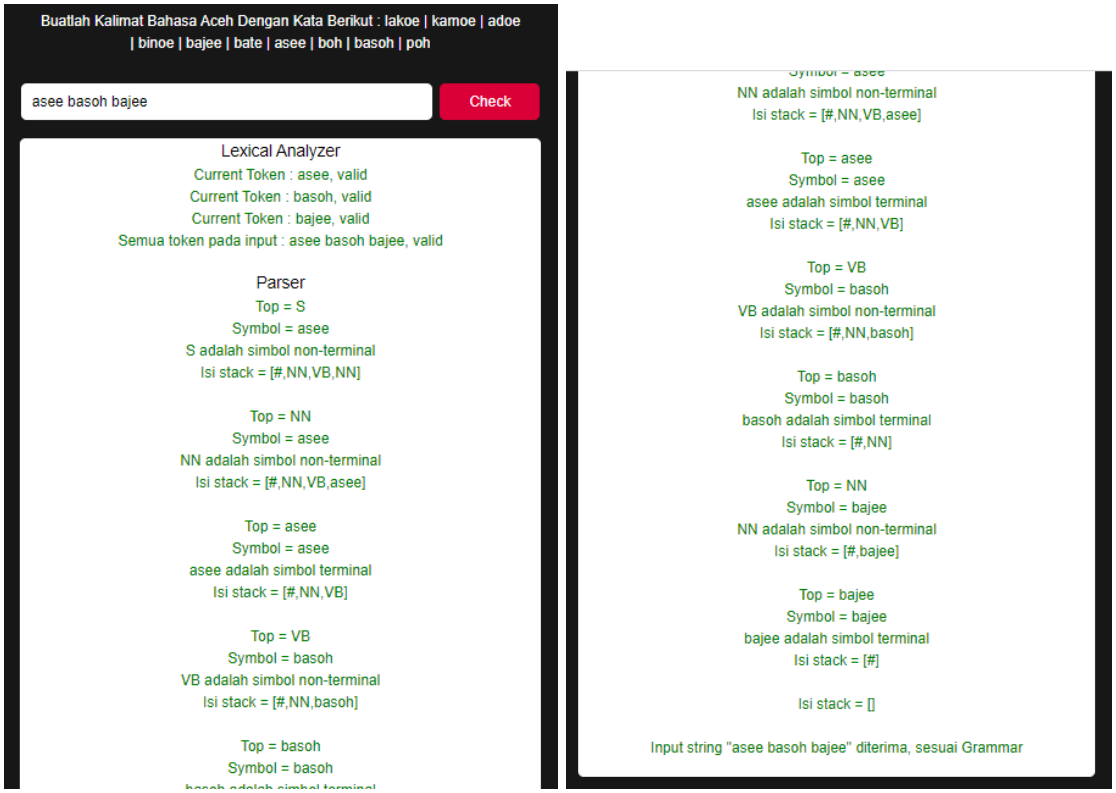
4.2.1 Hasil Running Program Dengan Hasil Token Valid



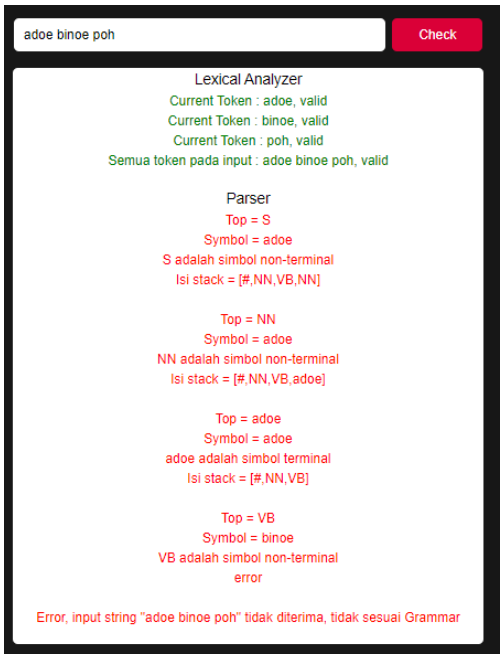


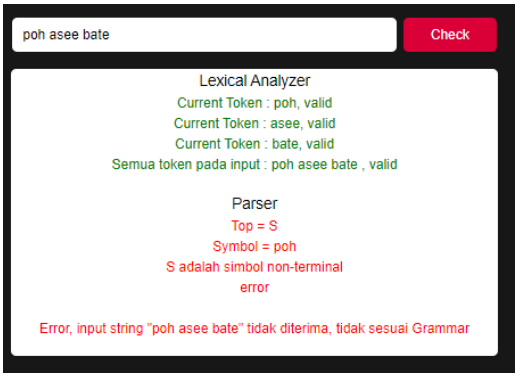


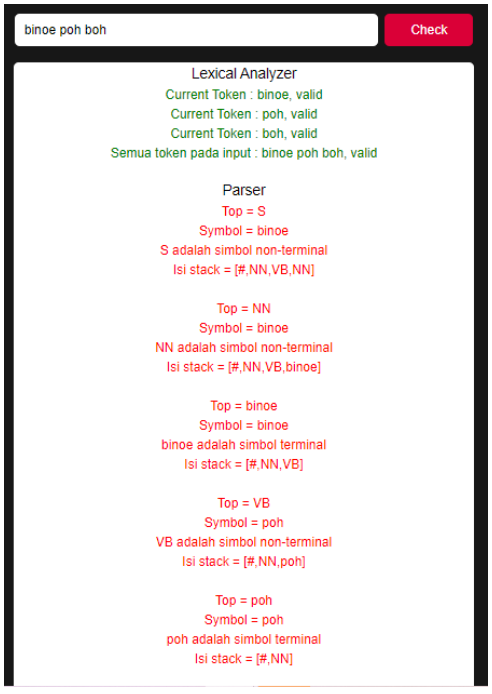


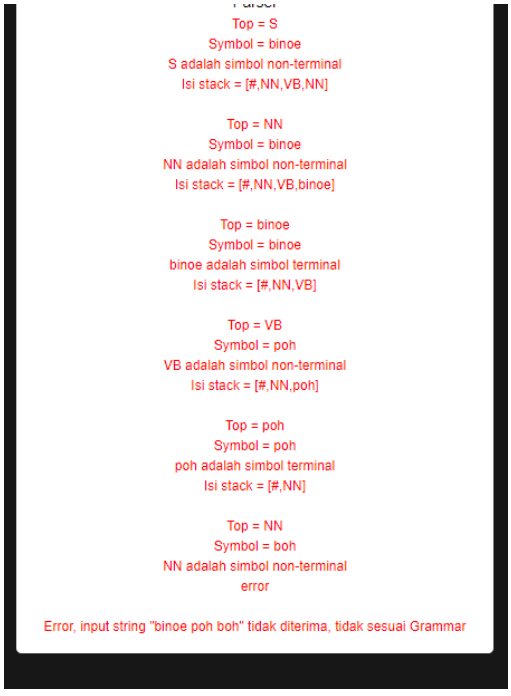

Buatlah Kalimat Bahasa Aceh Dengan Kata Berikut : lakoe | kamoe | adoe | binoe | bajee | bate | asee | boh | basoh | poh
asee basoh bajee
Check
Lexical Analyzer
Current Token : asee, valid
Current Token : basoh, valid
Current Token : bajee, valid
Semua token pada input : asee basoh bajee, valid
Parser
Top = S
Symbol = asee
S adalah simbol non-terminal
Isi stack = [#,NN,VB,NN]
Top = NN
Symbol = asee
NN adalah simbol non-terminal
Isi stack = [#,NN,VB,asee]
Top = asee
Symbol = asee
asee adalah simbol terminal
Isi stack = [#,NN,VB]
Top = VB
Symbol = basoh
VB adalah simbol non-terminal
Isi stack = [#,NN,basoh]
Top = basoh
Symbol = basoh
NN adalah simbol non-terminal
Isi stack = [#,NN,VB,asee]
Top = asee
Symbol = asee
asee adalah simbol terminal
Isi stack = [#,NN,VB]
Top = VB
Symbol = basoh
VB adalah simbol non-terminal
Isi stack = [#,NN,basoh]
Top = basoh
Symbol = basoh
basoh adalah simbol terminal
Isi stack = [#,NN]
Top = NN
Symbol = bajee
NN adalah simbol non-terminal
Isi stack = [#,bajee]
Top = bajee
Symbol = bajee
bajee adalah simbol terminal
Isi stack = [#]
Isi stack = []
Input string "asee basoh bajee" diterima, sesuai Grammar

4.2.2 Hasil Running Program Dengan Hasil Token Tidak Valid









Program Web dapat dilihat di sini.

BAB V
KESIMPULAN

Kesimpulan Pada penyelesaian dari tugas besar yang telah dibuat maka dapat disimpulkan bahwa, dengan bahasa Aceh yang memiliki struktur S-V-O (subject-verb-object) dapat dibuat context free grammar, finite automata, lexical analyzer, parse table, dan program parser. Dengan 5 hal tersebut dapat menunjukkan validasi dari susunan kata dalam bahasa aceh.