LAPORAN

TUGAS BESAR

Disusun untuk memenuhi tugas

Mata Kuliah: Teori Bahasa dan Automata
Dosen Pengampu: DEDE ROHIDIN, Drs., M.T.

Oleh:

MUHAMMAD RAFI ANDEO PRAJA (1301200278)
M NAUFAL RIFQI RAMDHANI (1301201572)
RIZKY FERDIAN PRASETYO (1301204229)

KELAS IF-44-01
JURUSAN S1 INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM

DAFTAR ISI

BAB I

PENDAHULUAN

1. Deskripsi Tugas Besar

   Dalam tugas besar ini kami memilih bahasa spanyol sebagai bahasa yang akan kami implementasikan kedalam tugas lexical analyzer dan parser kami. Kami menggunakan bahasa pemrograman JavaScript dan HTML yang dapat membantu kami membuat sebuah website sebagai salah satu tujuan utama tugas ini. Kata yang kami pilih dari bahasa spanyol tertera di tabel berikut ini.

Simbol Non-Terminal	Bahasa Spanyol	Bahasa Indonesia
VB	planta	menanam
NN	flor	bunga
VB	comer	makan
NN	arroz	nasi
VB	comprar	membeli
NN	zapato	sepatu
VB	usar	memakai
NN	vestir	baju
VB	buscar	mencari
NN	libro	buku
SB	yo	saya
SB	madre	ibu
SB	padre	ayah
SB	ellos	mereka
SB	nosotros	kami

BAB II

DASAR TEORI

1. Pengertian Context Free Grammar

   Kelas tata bahasa yang disebut Context Free (CF) oleh Chomsky telah digunakan dalam berbagai teori linguistik, baik sebagai satu - satunya komponen dan hanya sebagai salah satu dari beberapa komponen tata bahasa alami. CF grammar memainkan peran dominan dalam spesifikasi dan terjemahan bahasa pemrograman. Tata bahasa CF di representasikan oleh (I, T, S, P) yang dimana I ∪ T adalah sebuah alfabet dan P mengandung aturan untuk menulis kembali simbol dari I sebagai sebuah string dari simbol I ∪ T.

2. Pengertian Finite Automata

   Istilah dari finite automata menggambarkan kelas model komputasi yang dicirikan dengan memiliki jumlah state yang terbatas. Finite automata membentuk kelas bahasa pemrograman dengan set operasi yang sangat terbatas. Secara kasar, program dimana ada tidak ada rekursi atau perulangan.

3. Pengertian Lexical Analyzer

   Lexical Analyzer (juga dikenal sebagai lexer) adalah sebuah mesin pengenalan pola yang mengambil serangkaian huruf individual sebagai inputnya dan membaginya menjadi token. Selain itu, lexer juga menyaring apa pun (biasanya spasi, baris baru, komentar, dan lainnya) dan memisahkan token. Tujuan utama dari fase ini adalah untuk membuat fase berikutnya lebih mudah. Token yang dimaksud merupakan sebuah unit leksikal atau bahasa dasar yang tidak dapat dipisahkan elemennya. Token juga disebut adalah simbol terminal dalam tata bahasa, konstanta, operator, tanda baca, kata kunci, kelas rangkaian karakter dengan makna kolektif, nilai integer arbitrer, dan lainnya yang mewakili leksem. Leksem adalah sebuah string asli (urutan karakter) terdiri (cocok) dengan instance token.

4. Pengertian Parser

   Parser dimaksudkan untuk menunjukkan bahwa bagaimana kata - kata diekstraksi dari ucapan yang akan diimplementasikan dalam simulasi komputer. Parser juga dapat dibuat dalam bentuk tabel parser dan program parser yang dimana menggunakan salah satu algoritma yaitu LL Parser.

BAB III

PENYELESAIAN

1. Rancangan Context Free Grammar

   Context Free Grammar yang kami buat berstruktur S-V-O (Subject-Verb-Object) dengan bahasa spanyol. Aturan Context Free Grammar yang kami buat tertera dibawah ini:

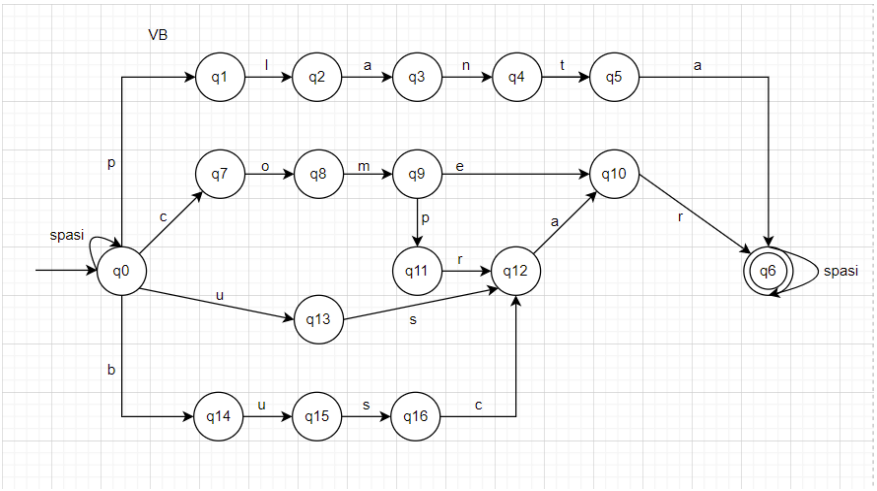
   S (Starting Symbol) → <SB (Subject)> <VB (Verb)> <NN (Noun)>
   SB (Subject) → <yo (aku)> <madre (ibu)> <padre (ayah)> <ellos (mereka)> <nosotros (kami)>
   VB (Verb) → <planta (Menanam)> <comer (Makan)> <comprar (Membeli)> <usar(Memakai)> <buscar (Mencari)>
   NN (Noun) → <flor (Bunga)> <arroz (Nasi)> <zapato (Sepatu)> <vestir (baju)> <libro (buku)>

   Simbol Non-Terminal terdiri dari: S (Starting Symbol), SB (Subject), VB (Verb), NN (Noun)
   Simbol Terminal terdiri dari: yo, madre, padre, ellos, nosotros, planta, comer, comprar, usar, buscar, flor, arroz, zapato, vestir, libro

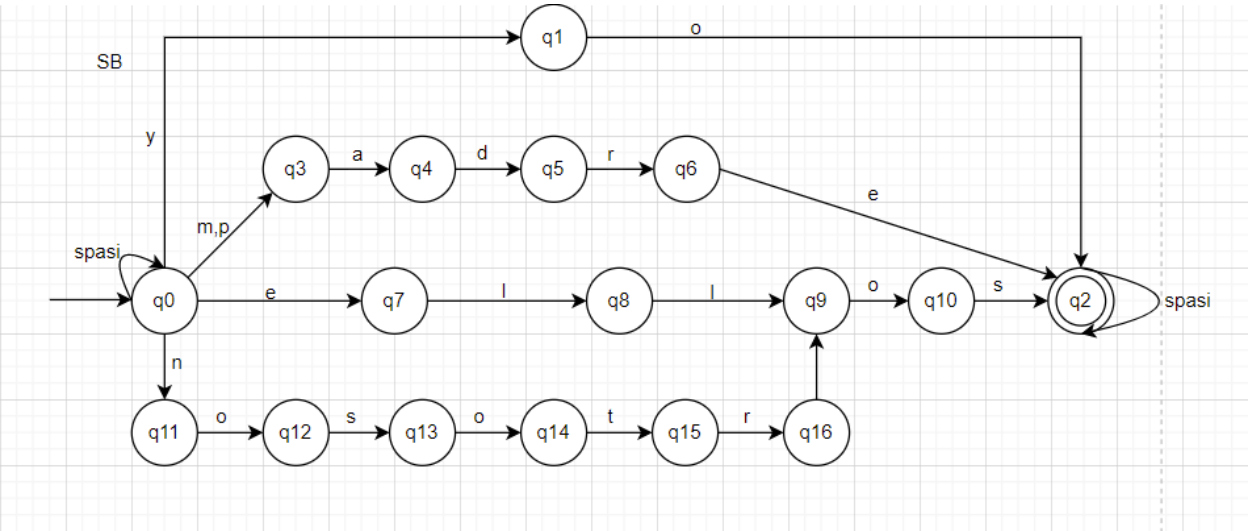
2. Rancangan Finite Automata

   Finite Automata yang kami rancang berjumlah 4 mesin yang dimana 3 mesin masing - masing finite automata berstruktur S-V-O (Subject-Verb-Object) dan finite automata terakhir adalah gabungan dari ketiga finite automata tersebut. Model finite automata berikut akan digunakan untuk membuat program lexical analyzer kami.

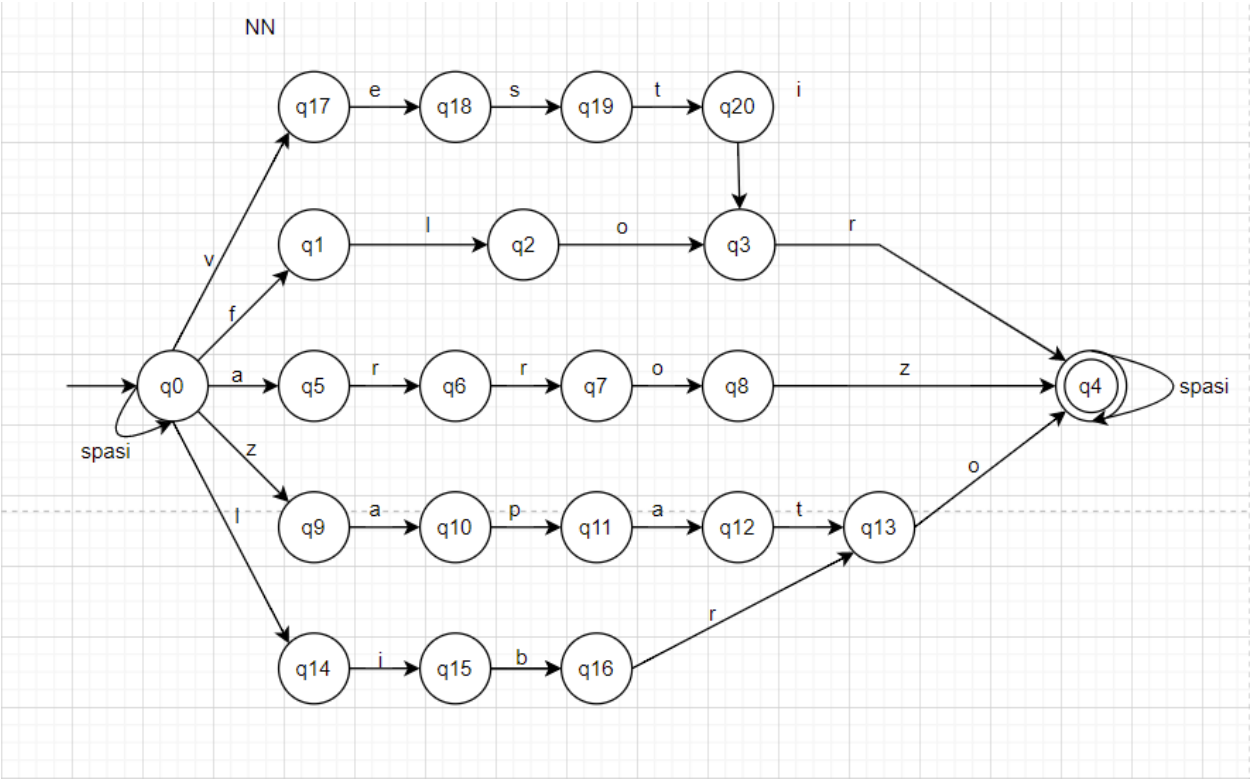
   Finite Automata Verb (VB):



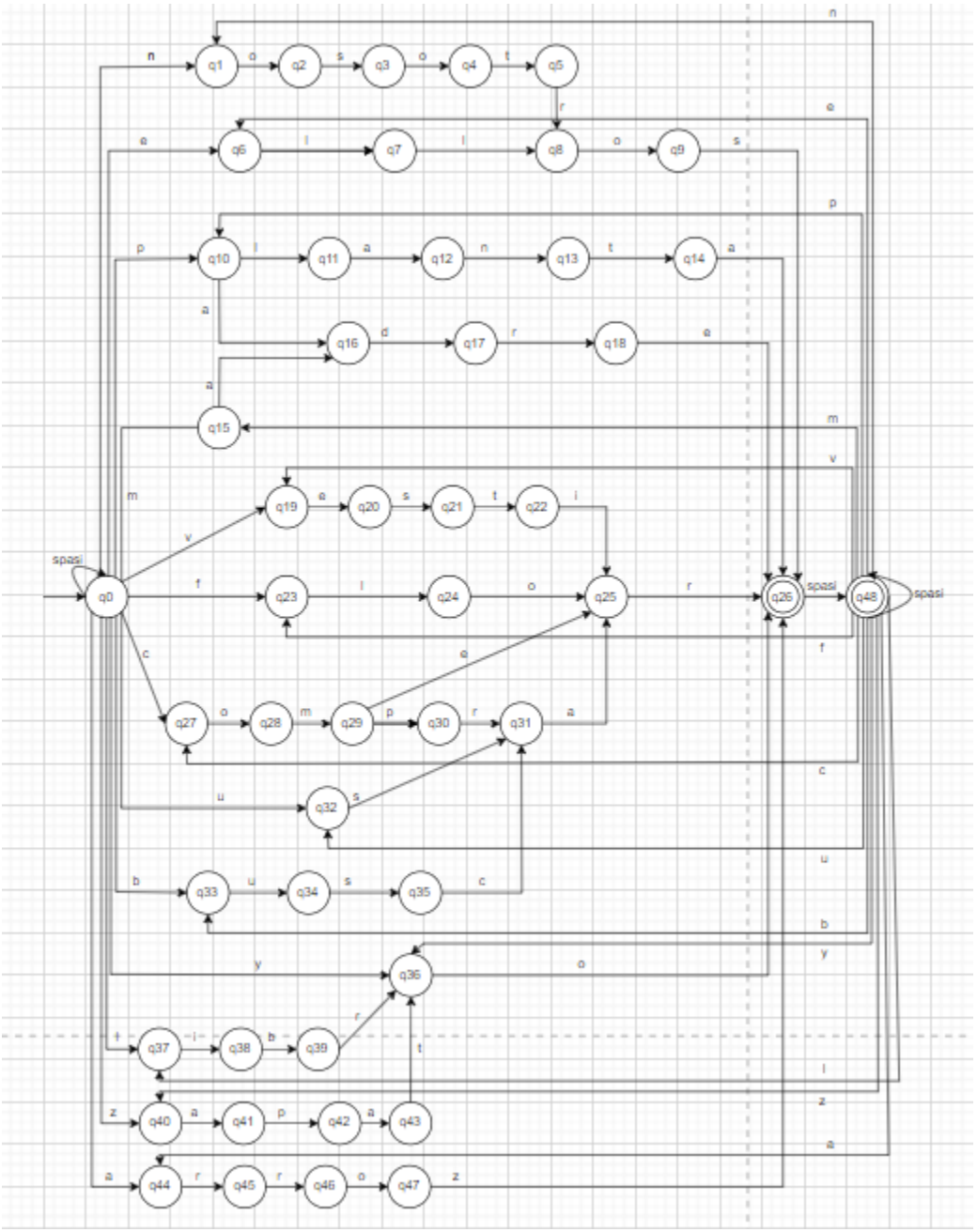
Finite Automata Subject (SB):



Finite Automata Noun (NN):



Finite Automata Gabungan (SB-VB-NN):



3. Rancangan Lexical Analyzer

Lexical Analyzer yang kami rancang berdasarkan finite automata gabungan yang sudah dibuat. Kami menggunakan library string yang dimana membantu proses pembuatan program agar dapat berjalan sesuai dengan tujuan tugas besar berikut. Kode program lexical analyzer beserta inputnya tertera di bagian bawah berikut:

```
// Proses Inisialisai
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', 'q23', 'q24', 'q25', 'q26',
'q27', 'q28', 'q29', 'q30', 'q31', 'q32', 'q33', 'q34', 'q35',
'q36', 'q37', 'q38', 'q39', 'q40',
    'q41', 'q42', 'q43', 'q44', 'q45', 'q46', 'q47', 'q48']

let transitionTable = {}

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

// Start
transitionTable[['q0', ' ']] = 'q0'
transitionTable[['q26', ' ']] = 'q48'
transitionTable[['q26', '#']] = 'ACCEPT'
transitionTable[['q48', ' ']] = 'q48'
transitionTable[['q48', '#']] = 'ACCEPT'

//Bahasa Spanyol
// Nosotros
transitionTable[['q0', 'n']] = 'q1'
transitionTable[['q1', 'o']] = 'q2'
transitionTable[['q2', 's']] = 'q3'
transitionTable[['q3', 'o']] = 'q4'
```

```
transitionTable[['q4', 't']] = 'q5'
transitionTable[['q5', 'r']] = 'q8'
transitionTable[['q8', 'o']] = 'q9'
transitionTable[['q9', 's']] = 'q26'
transitionTable[['q48', 'n']] = 'q1'

// Ellos
transitionTable[['q0', 'e']] = 'q6'
transitionTable[['q6', 'l']] = 'q7'
transitionTable[['q7', 'l']] = 'q8'
transitionTable[['q48', 'e']] = 'q6'

// Planta
transitionTable[['q0', 'p']] = 'q10'
transitionTable[['q10', 'l']] = 'q11'
transitionTable[['q11', 'a']] = 'q12'
transitionTable[['q12', 'n']] = 'q13'
transitionTable[['q13', 't']] = 'q14'
transitionTable[['q14', 'a']] = 'q26'
transitionTable[['q48', 'p']] = 'q10'

// Padre
transitionTable[['q10', 'a']] = 'q16'
transitionTable[['q16', 'd']] = 'q17'
transitionTable[['q17', 'r']] = 'q18'
transitionTable[['q18', 'e']] = 'q26'

// Madre
transitionTable[['q0', 'm']] = 'q15'
transitionTable[['q15', 'a']] = 'q16'
transitionTable[['q48', 'm']] = 'q15'

// vestir
transitionTable[['q0', 'v']] = 'q19'
transitionTable[['q19', 'e']] = 'q20'
transitionTable[['q20', 's']] = 'q21'
transitionTable[['q21', 't']] = 'q22'
transitionTable[['q22', 'i']] = 'q25'
transitionTable[['q25', 'r']] = 'q26'
transitionTable[['q48', 'v']] = 'q19'
```

```
// flor
transitionTable[['q0', 'f']] = 'q23'
transitionTable[['q23', 'l']] = 'q24'
transitionTable[['q24', 'o']] = 'q25'
transitionTable[['q48', 'f']] = 'q23'

// comer
transitionTable[['q0', 'c']] = 'q27'
transitionTable[['q27', 'o']] = 'q28'
transitionTable[['q28', 'm']] = 'q29'
transitionTable[['q29', 'e']] = 'q25'
transitionTable[['q48', 'c']] = 'q27'

// Comprar
transitionTable[['q29', 'p']] = 'q30'
transitionTable[['q30', 'r']] = 'q31'
transitionTable[['q31', 'a']] = 'q25'


// usar
transitionTable[['q0', 'u']] = 'q32'
transitionTable[['q32', 's']] = 'q31'
transitionTable[['q48', 'u']] = 'q32'

// Buscar
transitionTable[['q0', 'b']] = 'q33'
transitionTable[['q33', 'u']] = 'q34'
transitionTable[['q34', 's']] = 'q35'
transitionTable[['q35', 'c']] = 'q31'
transitionTable[['q48', 'b']] = 'q33'

// yo
transitionTable[['q0', 'y']] = 'q36'
transitionTable[['q36', 'o']] = 'q26'
transitionTable[['q48', 'y']] = 'q36'

// libro
transitionTable[['q0', 'l']] = 'q37'
transitionTable[['q37', 'i']] = 'q38'
```

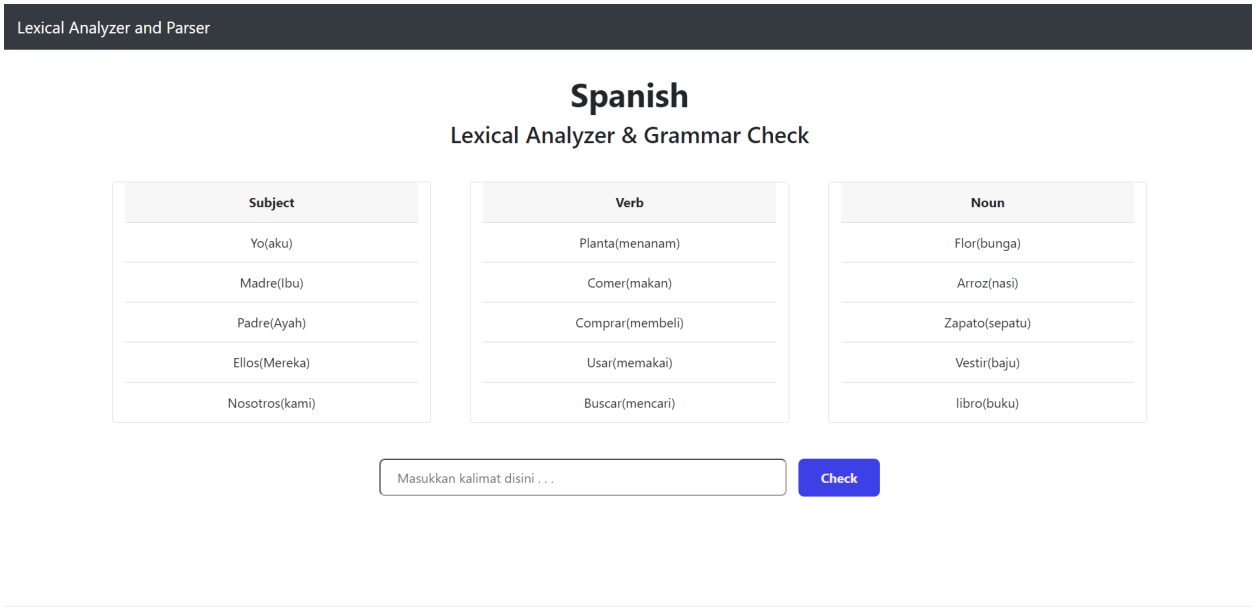
```
transitionTable[['q38', 'b']] = 'q39'
transitionTable[['q39', 'r']] = 'q36'
transitionTable[['q48', 'l']] = 'q37'

// zapato
transitionTable[['q0', 'z']] = 'q40'
transitionTable[['q40', 'a']] = 'q41'
transitionTable[['q41', 'p']] = 'q42'
transitionTable[['q42', 'a']] = 'q43'
transitionTable[['q43', 't']] = 'q36'
transitionTable[['q48', 'z']] = 'q40'

// arroz
transitionTable[['q0', 'a']] = 'q44'
transitionTable[['q44', 'r']] = 'q45'
transitionTable[['q45', 'r']] = 'q46'
transitionTable[['q46', 'o']] = 'q47'
transitionTable[['q47', 'z']] = 'q26'
transitionTable[['q48', 'a']] = 'q44'

//////////////// LEXICAL ANALYSIS ////////////////
const checkSentence = (sentence) => {
    let resultLa = document.getElementById('resultLa')
    let laTitle = document.getElementById('laTitle')
    let parserTitle = document.getElementById('parserTitle')
    let resultParser = document.getElementById('resultParser')
    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]
        if (currentChar != ' ' && currentChar != '#' &&
!alphabetList.includes(currentChar)) {
```

```
            console.log('error');
            resultLa.innerText += 'ERROR'
            resultLa.style.color = 'red'
            break
        }

        currentToken += currentChar
        state = transitionTable[[state, currentChar]]
        if (state == 'q26') {
            resultLa.innerText = resultLa.innerText + 'Current Token
: ' + currentToken + ', valid'
            resultLa.innerText += '\n'
            currentToken = ''
        }
        if (state == 'ERROR') {
            resultLa.innerText += 'ERROR'
            resultLa.style.color = 'red'
            break
        }
        idxChar++
    }

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

4. Rancangan dan Cara Menggunakan Website

Website yang kami rancang dibuat menggunakan bahasa pemrograman HTML dan JavaScript. Website akan meminta user untuk menginputkan sebuah string atau token yang nantinya akan dimasukkan ke dalam program dan akan mengoutputkan output yang sesuai. Berikut adalah tampilan antarmuka website:

Tampilan antarmuka website:



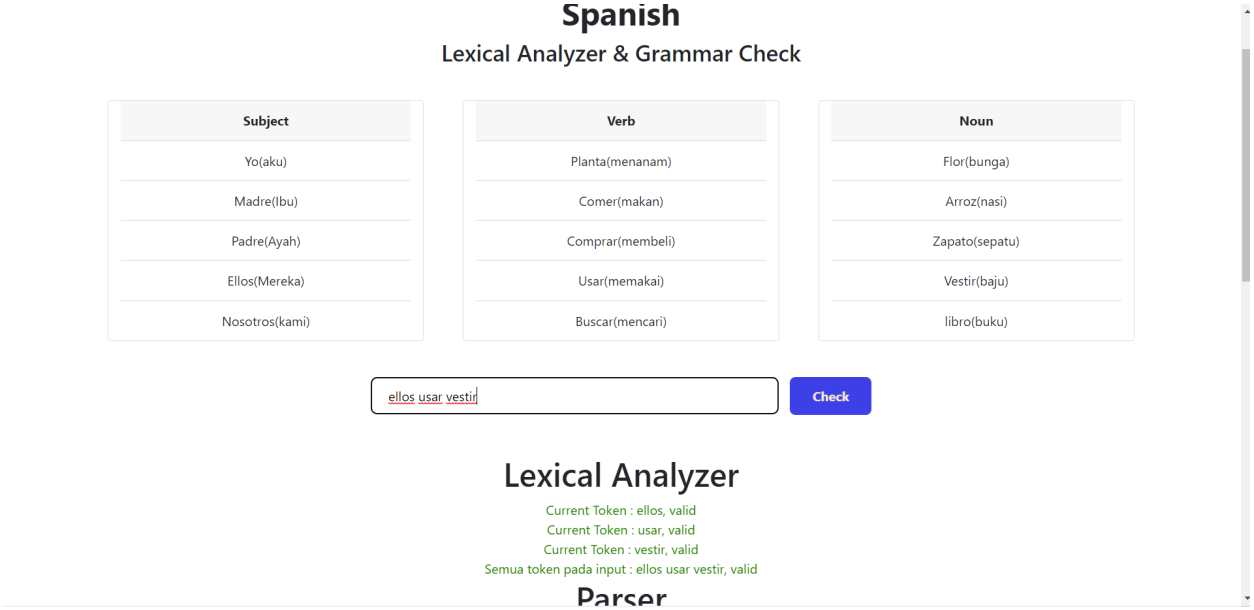
Cara menggunakan website:

1) Akses link berikut: https://tubes-tba.vercel.app/
2) Ketik string atau token yang ingin dicoba
3) Website akan mengeluarkan output valid jika token atau string yang diinput user berada pada tabel yang tersedia
4) Website akan mengeluarkan output tidak valid jika token atau string yang diinput user berada pada tabel yang tidak tersedia

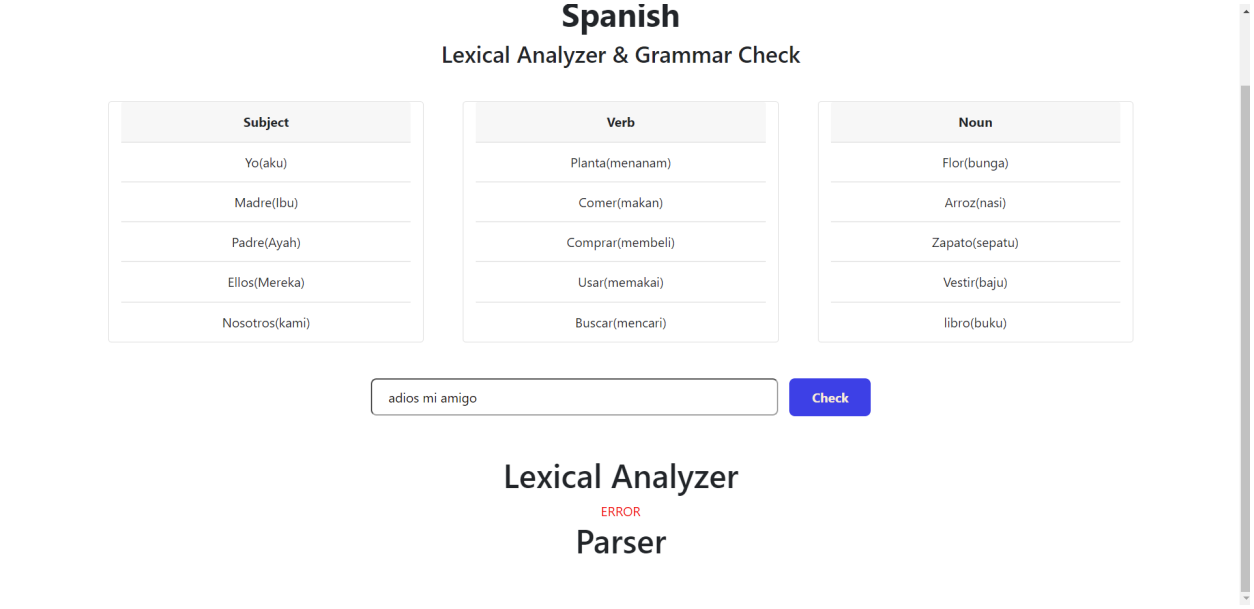
5. Hasil Output Lexical Analyzer

Kami mengelompokkan hasil output menjadi 2 bagian yang dimana mencoba 3 kata valid dan 3 kata tidak valid kedalam program. Lexical Analyzer yang dibuat memuat 15 non-terminal yang dibagi masing - masing 5 kata untuk Subject, Verb, dan Noun. Berikut adalah hasil output Lexical Analyzer:

Hasil output valid:



Hasil output tidak valid



6. Rancangan Parser

Selesai menyelesaikan finite automata dan lexical analyzer langkah selanjutnya yang harus dikerjakan adalah membuat parser table yang akan digunakan untuk membangun program parser kalimat dalam menentukan susunan aturan grammar yang telah ditentukan yaitu <subject>, <verb> , <noun>. Adapun susunan parser table dijelaskan sebagai berikut:

	yo	madre	padre	ellos	nosotros	planta	comer	comprar	usar	buscar	flor	arroz	zapato	vestir	libro	EOS
S	SB VB NN	SB VB NN	SB VB NN	SB VB NN	SB VB NN	error	error	error	error	error	error	error	error	error	error	error
SB	yo	madre	padre	ellos	nosotros	error	error	error	error	error	error	error	error	error	error	error
VB	error	error	error	error	error	planta	comer	comprar	usar	buscar	error	error	error	error	error	error
NN	error	error	error	error	error	error	error	error	error	error	flor	arroz	zapato	vestir	libro	error

```
        ///////////////// 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', 'SB', 'NN', 'VB']
              let terminals = ['yo', 'madre', 'padre', 'ellos',
'nosotros', 'flor', 'arroz', 'zapato', 'vestir', 'libro', 'planta',
'comer', 'comprar', 'buscar', 'usar']

        // Parse Table Definition
        let parseTable = {}
        parseTable[['S', 'yo']] = ['SB', 'VB', 'NN']
        parseTable[['S', 'madre']] = ['SB', 'VB', 'NN']
        parseTable[['S', 'padre']] = ['SB', 'VB', 'NN']
        parseTable[['S', 'ellos']] = ['SB', 'VB', 'NN']
        parseTable[['S', 'nosotros']] = ['SB', 'VB', 'NN']
        parseTable[['S', 'flor']] = ['error']
        parseTable[['S', 'arroz']] = ['error']
        parseTable[['S', 'zapato']] = ['error']
        parseTable[['S', 'vestir']] = ['error']
        parseTable[['S', 'libro']] = ['error']
        parseTable[['S', 'planta']] = ['error']
        parseTable[['S', 'comer']] = ['error']
        parseTable[['S', 'comprar']] = ['error']
        parseTable[['S', 'buscar']] = ['error']
```

```
        parseTable[['S', 'usar']] = ['error']
        parseTable[['S', 'EOS']] = ['error']

        parseTable[['SB', 'yo']] = ['yo']
        parseTable[['SB', 'madre']] = ['madre']
        parseTable[['SB', 'padre']] = ['padre']
        parseTable[['SB', 'ellos']] = ['ellos']
        parseTable[['SB', 'nosotros']] = ['nosotros']
        parseTable[['SB', 'flor']] = ['error']
        parseTable[['SB', 'arroz']] = ['error']
        parseTable[['SB', 'zapato']] = ['error']
        parseTable[['SB', 'vestir']] = ['error']
        parseTable[['SB', 'libro']] = ['error']
        parseTable[['SB', 'planta']] = ['error']
        parseTable[['SB', 'comer']] = ['error']
        parseTable[['SB', 'comprar']] = ['error']
        parseTable[['SB', 'buscar']] = ['error']
        parseTable[['SB', 'usar']] = ['error']
        parseTable[['SB', 'EOS']] = ['error']

        parseTable[['NN', 'yo']] = ['error']
        parseTable[['NN', 'madre']] = ['error']
        parseTable[['NN', 'padre']] = ['error']
        parseTable[['NN', 'ellos']] = ['eerror']
        parseTable[['NN', 'nosotros']] = ['error']
        parseTable[['NN', 'flor']] = ['flor']
        parseTable[['NN', 'arroz']] = ['arroz']
        parseTable[['NN', 'zapato']] = ['zapato']
        parseTable[['NN', 'vestir']] = ['vestir']
        parseTable[['NN', 'libro']] = ['libro']
        parseTable[['NN', 'planta']] = ['error']
        parseTable[['NN', 'comer']] = ['error']
        parseTable[['NN', 'comprar']] = ['error']
        parseTable[['NN', 'buscar']] = ['error']
        parseTable[['NN', 'usar']] = ['error']
        parseTable[['NN', 'EOS']] = ['error']


        parseTable[['VB', 'planta']] = ['planta']
        parseTable[['VB', 'comer']] = ['comer']
```

```
        parseTable[['VB', 'comprar']] = ['comprar']
        parseTable[['VB', 'buscar']] = ['buscar']
        parseTable[['VB', 'usar']] = ['usar']
        parseTable[['VB', 'yo']] = ['error']
        parseTable[['VB', 'madre']] = ['error']
        parseTable[['VB', 'padre']] = ['error']
        parseTable[['VB', 'ellos']] = ['ellos']
        parseTable[['VB', 'nosotros']] = ['error']
        parseTable[['VB', 'flor']] = ['error']
        parseTable[['VB', 'arroz']] = ['error']
        parseTable[['VB', 'zapato']] = ['error']
        parseTable[['VB', 'vestir']] = ['error']
        parseTable[['VB', 'libro']] = ['error']
        parseTable[['VB', 'EOS']] = ['error']

        // Stack Initialization
        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') {
```

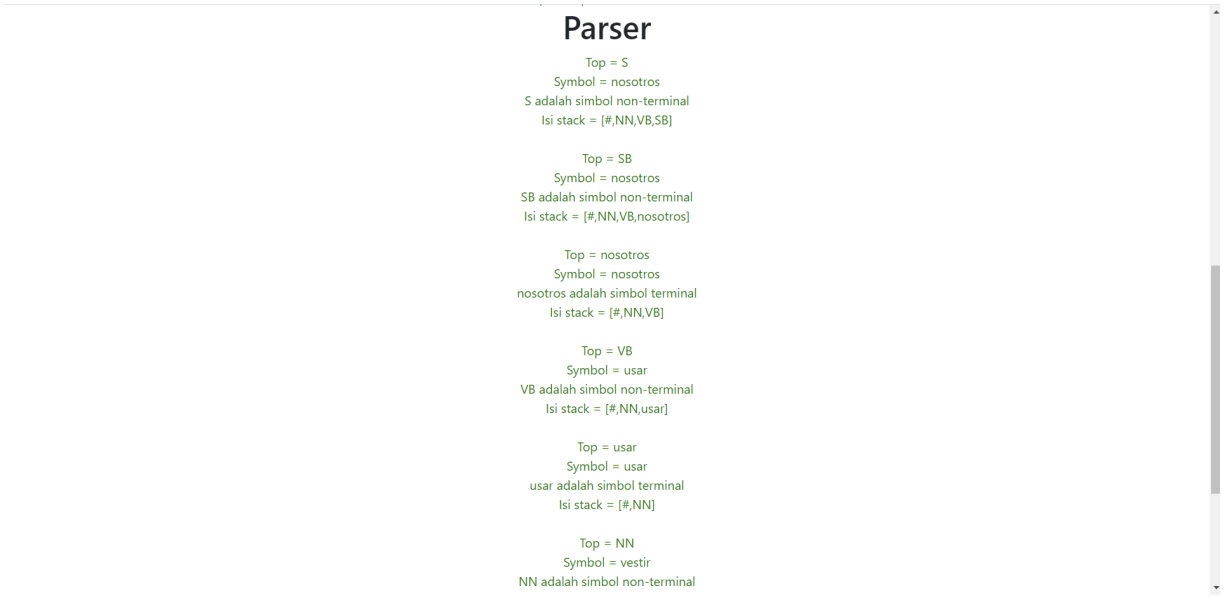
```
                                        resultParser.innerText =
resultParser.innerText + 'Isi stack = ' + '[' + stack + ']' + '\n
\n'
                        stack.pop()
                    }
                } else {
                      resultParser.innerText = resultParser.innerText
+ 'error \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'
        }
    }
}

let form = document.getElementById('form')

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

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

7. Hasil Output Parser

Kami mengelompokkan hasil output menjadi 2 bagian yang dimana mencoba 3 kalimat dengan panjang 2 - 5 kata yang diterima grammar dan tidak diterima grammar. Berikut adalah hasil output dari Parser:

Hasil output valid 1 (nosotros usar vestir):



Top = nosotros
Symbol = nosotros
nosotros adalah simbol terminal
Isi stack = [#,NN,VB]

Top = VB
Symbol = usar
VB adalah simbol non-terminal
Isi stack = [#,NN,usar]

Top = usar
Symbol = usar
usar adalah simbol terminal
Isi stack = [#,NN]

Top = NN
Symbol = vestir
NN adalah simbol non-terminal
Isi stack = [#,vestir]

Top = vestir
Symbol = vestir
vestir adalah simbol terminal
Isi stack = [#]

Isi stack = []

Input string "nosotros usar vestir" diterima, sesuai Grammar

Hasil output valid 2 (ellos comer libro):

Parser

Top = S
Symbol = ellos
S adalah simbol non-terminal
Isi stack = [#,NN,VB,SB]

Top = SB
Symbol = ellos
SB adalah simbol non-terminal
Isi stack = [#,NN,VB,ellos]

Top = ellos
Symbol = ellos
ellos adalah simbol terminal
Isi stack = [#,NN,VB]

Top = VB
Symbol = comer
VB adalah simbol non-terminal
Isi stack = [#,NN,comer]

Top = comer
Symbol = comer
comer adalah simbol terminal
Isi stack = [#,NN]

Top = NN
Symbol = libro
NN adalah simbol non-terminal

Top = ellos
Symbol = ellos
ellos adalah simbol terminal
Isi stack = [#,NN,VB]

Top = VB
Symbol = comer
VB adalah simbol non-terminal
Isi stack = [#,NN,comer]

Top = comer
Symbol = comer
comer adalah simbol terminal
Isi stack = [#,NN]

Top = NN
Symbol = libro
NN adalah simbol non-terminal
Isi stack = [#,libro]

Top = libro
Symbol = libro
libro adalah simbol terminal
Isi stack = [#]

Isi stack = []

Input string "ellos comer libro" diterima, sesuai Grammar

Hasil output valid 3 (yo planta flor):

Parser

Top = S
Symbol = yo
S adalah simbol non-terminal
Isi stack = [#,NN,VB,SB]

Top = SB
Symbol = yo
SB adalah simbol non-terminal
Isi stack = [#,NN,VB,yo]

Top = yo
Symbol = yo
yo adalah simbol terminal
Isi stack = [#,NN,VB]

Top = VB
Symbol = planta
VB adalah simbol non-terminal
Isi stack = [#,NN,planta]

Top = planta
Symbol = planta
planta adalah simbol terminal
Isi stack = [#,NN]

Top = NN
Symbol = flor
NN adalah simbol non-terminal
Isi stack = [#,flor]

Top = yo
Symbol = yo
yo adalah simbol terminal
Isi stack = [#,NN,VB]

Top = VB
Symbol = planta
VB adalah simbol non-terminal
Isi stack = [#,NN,planta]

Top = planta
Symbol = planta
planta adalah simbol terminal
Isi stack = [#,NN]

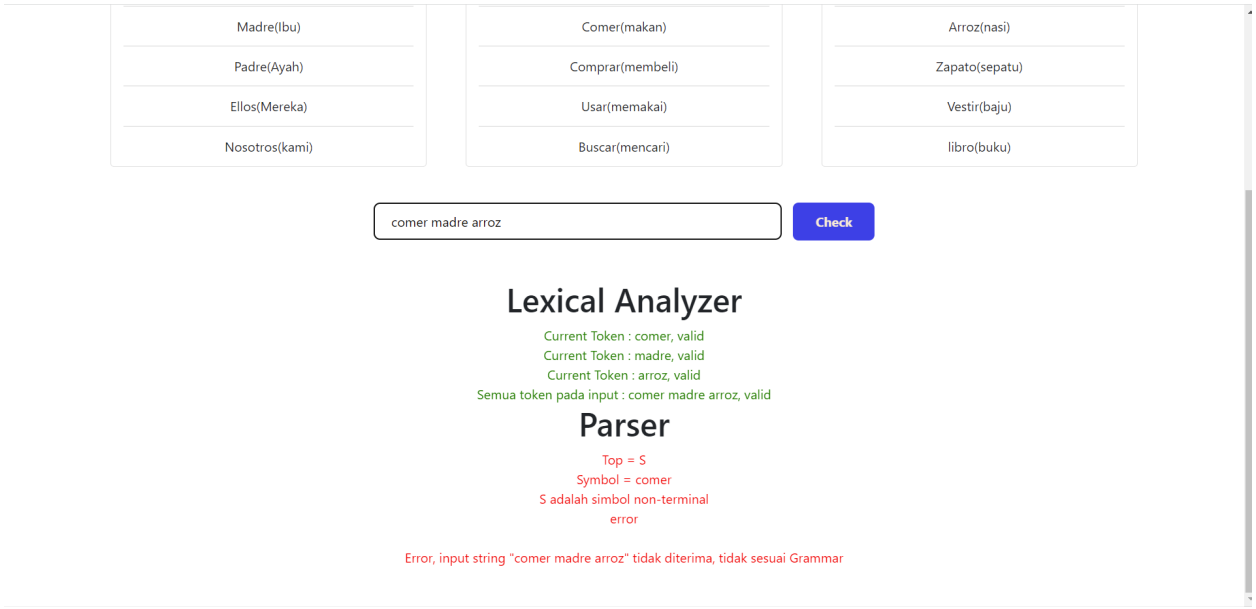
Top = NN
Symbol = flor
NN adalah simbol non-terminal
Isi stack = [#,flor]

Top = flor
Symbol = flor
flor adalah simbol terminal
Isi stack = [#]

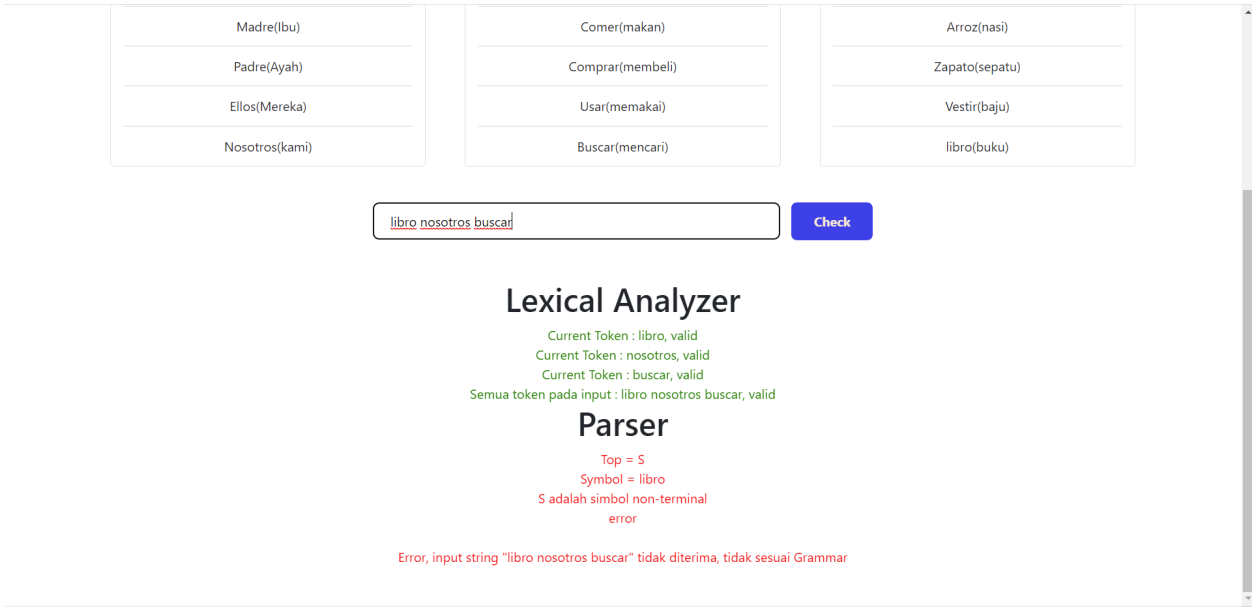
Isi stack = []

Input string "yo planta flor" diterima, sesuai Grammar

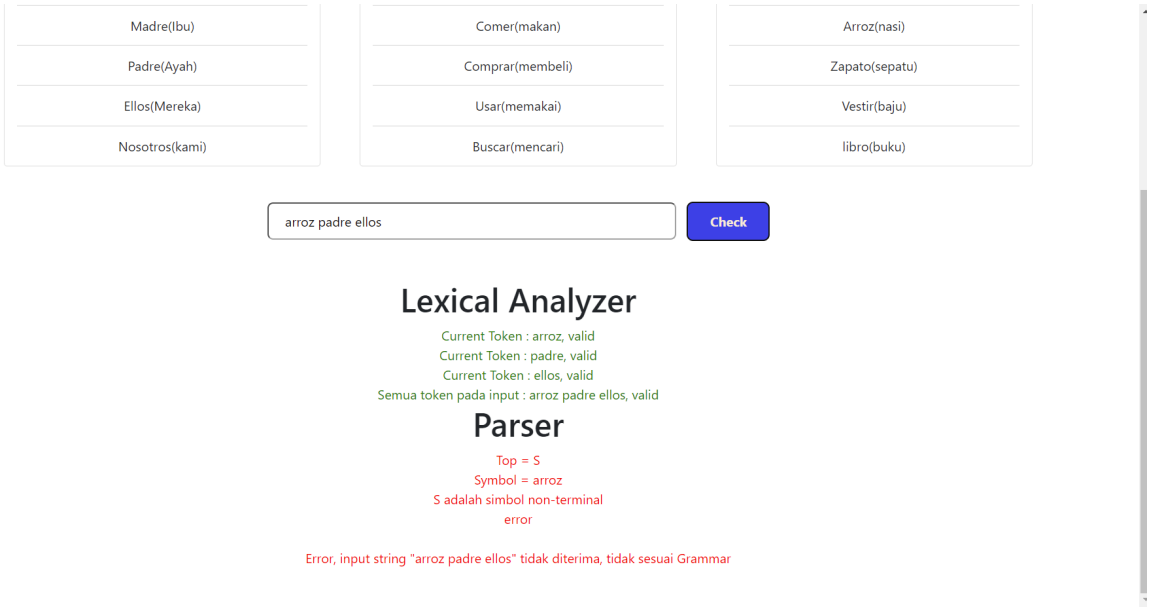
Hasil output tidak valid 1 (comer madre arroz):



Hasil output tidak valid 2 (libro nosotros buscar):



Hasil output tidak valid 3 (arroz padre ellos):



BAB IV

PENUTUP

1. Kesimpulan

   Kesimpulan yang dapat kami buat adalah, rancangan finite automata dapat diimplementasikan kedalam sebuah program yang dinamakan lexical analyzer. Sebelum membuat finite automata diperlukan membuat sebuah context free grammar yang nantinya akan digunakan untuk membuat sebuah parse table. Parse table juga dapat dirancang di dalam sebuah program. Lexical analyzer adalah sebuah program yang mendeteksi kebenaran dari sebuah string dan state di finite automata, sedangkan parse table adalah sebuah program yang mendeteksi kebenaran dari grammar sebuah string dan state di context free grammar. Kami mengimplementasikan lexical analyzer dan parser yang kami buat kedalam sebuah program berbasis website.

2. Saran

   Kami menyadari bahwa laporan, program, dan website yang kami buat masih jauh dari kata sempurna, maka dari itu kami berharap dapat lebih baik dan lebih teliti dalam pembuatan program, website, dan penulisan laporan untuk tugas mata kuliah lainnya. Kami mengucapkan terima kasih banyak kepada bapak DEDE ROHIDIN, Drs., M.T. sebagai dosen pengampu mata kuliah teori bahasa dan automata yang telah memberikan kami kesempatan sebagai mahasiswa untuk belajar dan mencari ilmu sebanyak banyaknya mengenai context free grammar, finite automata, lexical analyzer, parser, dan dasar pemrograman website. Kami berharap bahwa kami bisa mendapatkan kritik dan saran dari siapapun yang membaca laporan dan mencoba program serta website kami, kami menerima segala kritik dan saran yang dapat memperbaiki hasil kinerja kami untuk tugas kedepannya. Semoga laporan, program, dan website kami bisa bermanfaat bagi siapapun yang tertarik mengenai tugas ini.

DAFTAR PUSTAKA

M. V. Lawson. (2003). Finite Automata. Diambil 5 Juni 2022, dari https://www.macs.hw.ac.uk/~markl/preprints/Lawson.pdf

T. V. Griffiths., S. R. Petrick. (1965). On The Relative Efficiencies of Context-Free Grammar Recognizers. Diambil 5 Juni 2022, dari https://dl.acm.org/doi/pdf/10.1145/364914.364943

Biswajit. Bhowmik., Abhishek. Kumar., Abhishek. Kumar. Jha., Rajesh. Kumar. Agrawal. (2010). A New Approach of Compiler Design in Context of Lexical Analyzer and Parser Generation for NextGen Languages. Diambil 5 Juni 2022, dari https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.206.4342&rep=rep1&type=pdf

Pierre. Perruchet., Annie. Vinter. (1998). PARSER: A Model for Word Segmentation. Diambil 5 Juni 2022, dari http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.379.7326&rep=rep1&type=pdf