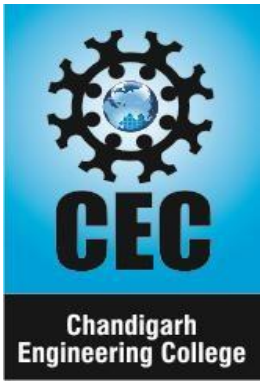
Lexical Analyzer

PROJECT REPORT

OF MAJOR PROJECT

BACHELOR OF ENGINEERING

COMPUTER SCIENCE & ENGINEERING



Submitted By:
Akshat Chaudhary

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
CHANDIGARH ENGINEERING COLLEGE
LANDRAN, MOHALI-140307

TABLE OF CONTENT

CERTIFICATE

This is to certify that the work embodied in this Project Report entitled **" Lexical Analyzer "** being submitted by **" Akshat Chaudhary " -** UID **" 2003190 ", 6**th Semester for partial fulfillment of the requirement for the degree of **" Bachelor of Engineering in Computer Science & Engineering "** discipline in " **Chandigarh Engineering College** " during the academic session Jan-Jun 2023 is a record of bonafide piece of work, carried out by student under my supervision and guidance in the **" Department of Computer Science & Engineering ", ChandigarhEngineering College.**

APPROVED & GUIDED BY:

Ms. Maninder

DECLARATION

I, student of **Bachelor of Engineering** in **Computer Science & Engineering, 6th Semester** , **session: Jan – June 2022, Chandigarh Engineering College,** hereby declare that the work presented in this Project Report entitled **"Lexical Analyzer "** is the outcome of my own work, is bonafide and correct to the best of my knowledge and this work has been carried out taking care of Engineering Ethics. The work presented does not infringe any patented work and has not been submitted to any other university or anywhere else for the award of any degree or any professional diploma.

Student details:

Akshat Chaudhary

APPROVED & GUIDED BY:

Ms. Maninder

To our parents, teachers and all the well wishers out there . .

ABSTRACT

The compiler is software that converts a program written in a high-level language (Source Language) to a low-level language (Object/Target/Machine Language/0, 1's).

A translator or language processor is a program that translates an input program written in a programming language into an equivalent program in another language. The compiler is a type of translator, which takes a program written in a high-level programming language as input and translates it into an equivalent program in low-level languages such as machine language or assembly language.

The program written in a high-level language is known as a source program, and the program converted into a low-level language is known as an object (or target) program. Without compilation, no program written in a high-level language can be executed. For every programming language, we have a different compiler; however, the basic tasks performed by every compiler are the same. The process of translating the source code into machine code involves several stages, including lexical analysis, syntax analysis, semantic analysis, code generation, and optimization.

High-Level Programming Language :-

A high-level programming language is a language that has an abstraction of attributes of the computer. High-level programming is more convenient to the user in writing a program.

Low-Level Programming Language :-

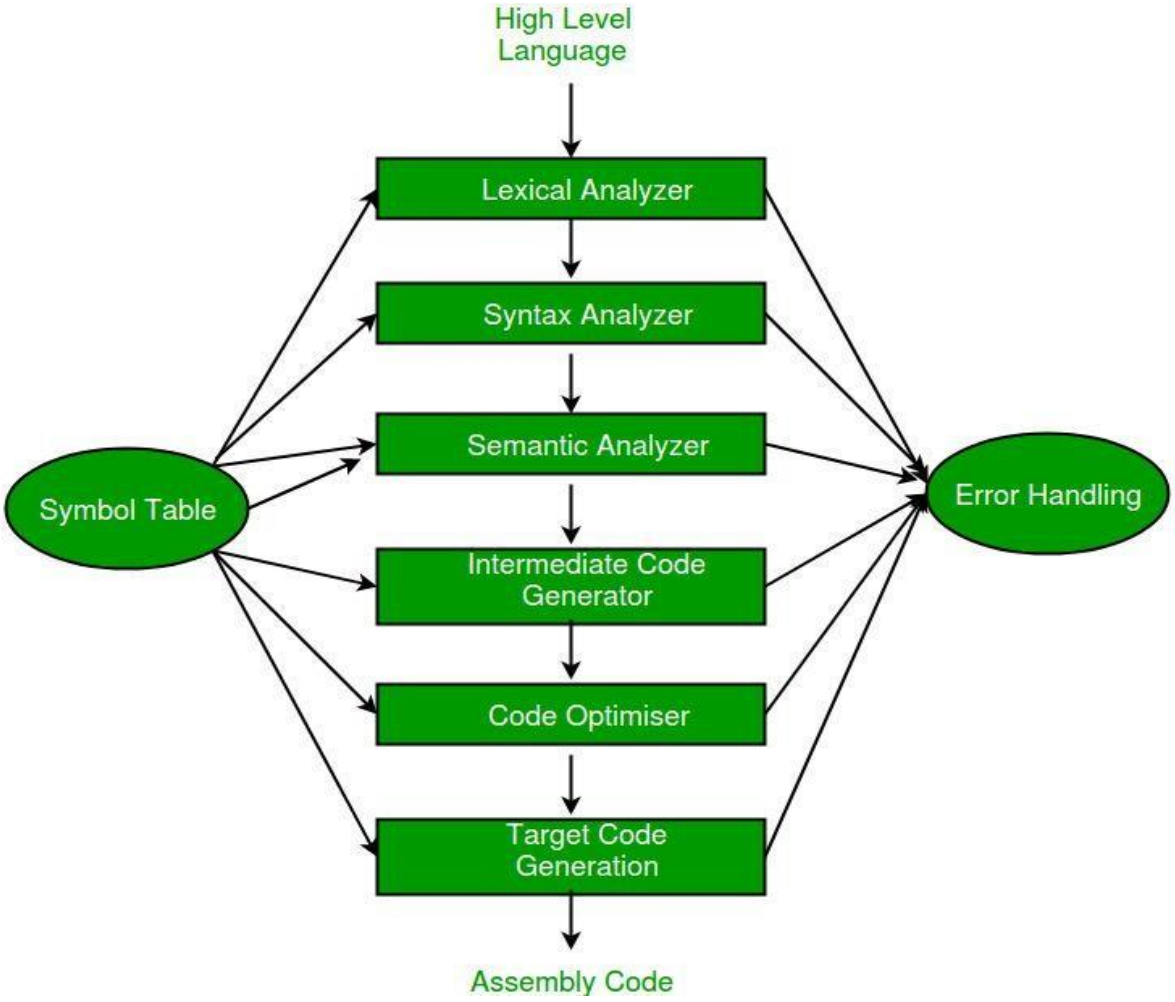
A low-Level Programming language is a language that doesn't require programming ideas and concepts.

Stages of Compiler Design :-

- **Lexical Analysis:** The first stage of compiler design is lexical analysis, also known as scanning. In this stage, the compiler reads the source code character by character and breaks it down into a series of tokens, such as keywords, identifiers, and operators. These tokens are then passed on to the next stage of the compilation process.
- **Syntax Analysis:** The second stage of compiler design is syntax analysis, also known as parsing. In this stage, the compiler checks the syntax of the source code to ensure that it conforms to the rules of the programming language. The compiler builds a parse tree, which is a hierarchical

representation of the program's structure, and uses it to check for syntax errors.

- **Semantic Analysis:** The third stage of compiler design is semantic analysis. In this stage, the compiler checks the meaning of the source code to ensure that it makes sense. The compiler performs type checking, which ensures that variables are used correctly and that operations are performed on compatible data types. The compiler also checks for other semantic errors, such as undeclared variables and incorrect function calls.
- **Code Generation:** The fourth stage of compiler design is code generation. In this stage, the compiler translates the parse tree into machine code that can be executed by the computer. The code generated by the compiler must be efficient and optimized for the target platform.
- **Optimization:** The final stage of compiler design is optimization. In this stage, the compiler analyzes the generated code and makes optimizations to improve its performance. The compiler may perform optimizations such as constant folding, loop unrolling, and function inlining.



TITLE

What is Lexical Analysis ?

Lexical analysis is the starting phase of the compiler. It gathers modified source code that is written in the form of sentences from the language preprocessor. The lexical analyzer is responsible for breaking these syntaxes into a series of tokens, by removing whitespace in the source code. If the lexical analyzer gets any invalid token, it generates an error. The stream of character is read by it and it seeks the legal tokens, and then the data is passed to the syntax analyzer, when it is asked for.

Terminologies :-

There are three terminologies-

- Token
- Pattern
- Lexeme

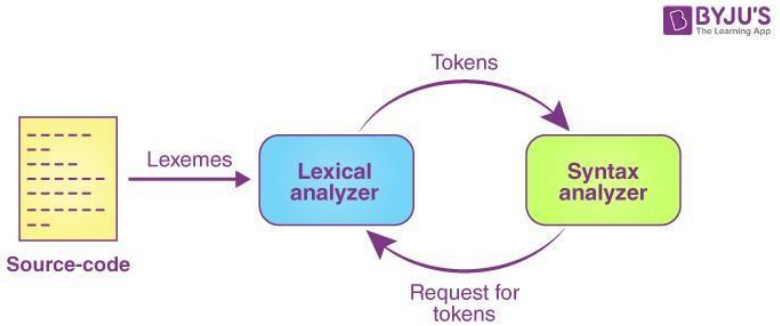
Token: It is a sequence of characters that represents a unit of information in the source code.

Pattern: The description used by the token is known as a pattern.

Lexeme: A sequence of characters in the source code, as per the matching pattern of a token, is known as lexeme. It is also called the instance of a token.

The Architecture of Lexical Analyzer :-

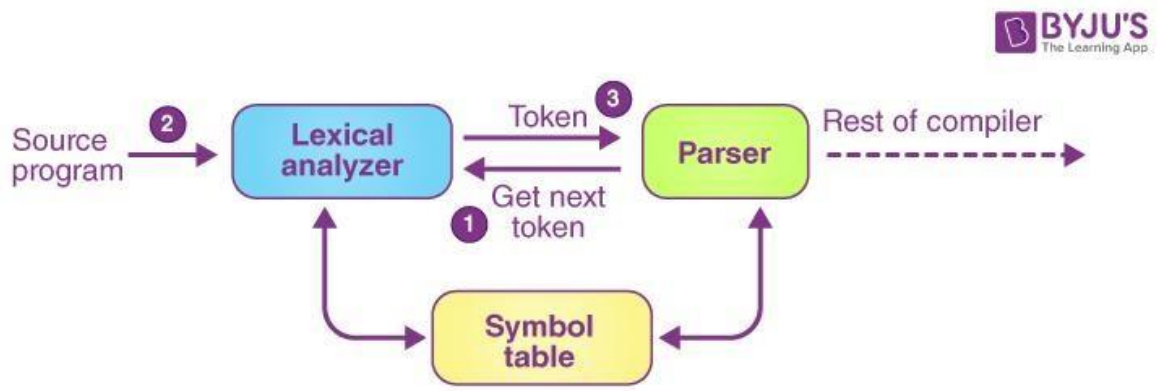
To read the input character in the source code and produce a token is the most important task of a lexical analyzer. The lexical analyzer goes through with the entire source code and identifies each token one by one. The scanner is responsible to produce tokens when it is requested by the parser. The lexical analyzer avoids the whitespace and comments while creating these tokens. If any error occurs, the analyzer correlates these errors with the source file and line number.



Roles and Responsibility of Lexical Analyzer :-

The lexical analyzer performs the following tasks-

- The lexical analyzer is responsible for removing the white spaces and comments from the source program.
- It corresponds to the error messages with the source program.
- It helps to identify the tokens.
- The input characters are read by the lexical analyzer from the source code.



Advantages of Lexical Analysis :-

- Lexical analysis helps the browsers to format and display a web page with the help of parsed data.
- It is responsible to create a compiled binary executable code.
- It helps to create a more efficient and specialized processor for the task.

Disadvantages of Lexical Analysis :-

- It requires additional runtime overhead to generate the lexer table and construct the tokens.
- It requires much effort to debug and develop the lexer and its token description.
- Much significant time is required to read the source code and partition it into tokens.

SYSTEM ANALYSIS

INTRODUCTION: Analysis can be defined by breaking up of any whole so as to find out their nature, working functionalities etc. It defines design as to make preliminary sketches of; to sketch a pattern or outline for planning. To plan and carry out especially by artistic arrangement or in a skilful way. System analysis and design can be characterized as a set of techniques and processes, a community of interests, a culture and intellectual orientation.
The various tasks in system analysis phase including the following
Understanding Application Project Planning
Project Scheduling
Performing Trade Studies
Performing Cost Benefit Analysis
Recommending Alternative Solutions
Supervising, Installing, Maintaining the system This system allows the user to feel the application of the software. First design the class which will keeps track various functions like UI structure, Data and file input, grid selection, Solution analysis and displaying output etc. And according to the input given by user this project will display the result in a very
simplified format.

2.1 Existing System:

The Existing system is a simple lexical analyzer without a proper UI that makes it very difficult to operate and understand. Conventional lexical analyzer is not very efficient and its maintenance can be complicated. Regular expressions and the finite state machines are not capable of handling recursive patterns, such as n opening parentheses, followed by a statement, followed by n closing parentheses. This application requires correct feed on input into the respective field. Suppose the wrong inputs are entered, then the whole process is to be done again. So, the users find it difficult to use.

2.2 Proposed System:

To overcome the drawbacks of the existing system, the proposed system has been evolved. This project aims to reduce the complexity and saving time to generate accurate results
from the user's perspective. The system provides with the best Graphical User Interface.
The efficient reports can be generated by using this proposed system.

Advantages of Proposed System:

- The machine has been made user friendly with proper use of graphical interface.
- The user can input any code without any interpretation.
- It is highly reliable, approximate result from user.

- The game has been made as a thorough expert system.
- The result is accurate and fast
- It is a good for students to understand the process and function of lexical analyzer.

2.3 Feasibility Study:

Feasibility study begins once the goals are defined. It starts by generating board possible
solutions, which are possible to give an indication of what is new system should look like. That is where creativity and imagination are used. Analysts must think up the new ways of doing things generating new ideas. There is no need to go into the detailed system operation yet. The solution should provide enough information to make reasonable estimates about
project cost and give user an indication of how the new system will fit into the organization. Feasibility of a new system means ensuring that the new system, which we're going to implement is efficient and affordable. There are various types of feasibility that should be
taken into consideration:

2.3.1 Economical Feasibility:

Development of this application is highly economically feasible. The only thing to be done is making an environment with effective supervision. It is time effective in sense that it will eliminate paper work completely. The system that is being developed is also cost effective.

2.3.2 Technical Feasibility:

The technical requirement for the system is economic and it doesn't use any other hardware or software. Technical evaluation must also assess whether the Existing System can be paraded to use the new technology and whether the organization has the expertise to use it. Install all the upgrades frameworks into the node package manager supported Windows based application. This application mostly depends on React JS, bootstrapped by Create react app by Node package manager.

2.3.3 OPERATIONAL FEASIBILITY:

The system working is quite easy to use and learn due to its simple but attractive interface.
User requires no prerequisites for operating the product Technical performance includes issues such as determining whether the system can sense the proper click of the mouse or after sensing the click it places the symbol in the desired cell and whether the system is organized in such a way that it always displays the correct result according to the moves
made by the players.

SYSTEM SPECIFICATION

3.1 Hardware Requirement:

Minimum RAM: - 1GB
Minimum Hard Disk: - 128GB
Processor: - Intel Pentium 4(1.50 GHz) or above

3.2 Software Requirement:

Operating System: - Support for both LINUX and WINDOWS users
Back End: - JavaScript
Front End Language: - React JS (Framework of JavaScript)

SOFTWARE DESCRIPTION

4.1 Node Package Manager (npm)

Node Package Manager (NPM) is a command line tool that installs, updates or uninstalls Node.js packages in your application. It is also an online repository for open-source Node.js packages. The node community around the world creates useful modules and publishes them as packages in this repository.
NPM – or "Node Package Manager" – is the default package manager for JavaScript's runtime Node.js.
It's also known as "Ninja Pumpkin Mutants", "Nonprofit Pizza Makers", and a host of other random names that you can explore and probably contribute to over at npm-expansions.
NPM consists of two main parts:

- a CLI (command-line interface) tool for publishing and downloading packages, and
- an online repository that hosts JavaScript packages

4.2 Development Tools and Technologies:

4.2.1 React JS:

The React.js framework is an open-source JavaScript framework and library developed by Facebook. It’s used for building interactive user interfaces and

web applications quickly and efficiently with significantly less code than you would with vanilla JavaScript.
In React, you develop your applications by creating reusable components that you can think of as independent Lego blocks. These components are individual pieces of a final interface, which, when assembled, form the application's entire user interface.
React's primary role in an application is to handle the view layer of that application just like the V in a model-view-controller (MVC) pattern by providing the best and most efficient rendering execution. Rather than dealing with the whole user interface as a single unit, React.js encourages developers to separate these complex UIs into individual reusable components that form the building blocks of the whole UI. In doing so, the ReactJS framework combines the speed and efficiency of JavaScript with a more efficient method of manipulating the DOM to render web pages faster and create highly dynamic and responsive web applications.

4.2.2 Create React App:

The create-react-app is an excellent tool for beginners, which allows you to create and run React project very quickly. It does not take any configuration manually. This tool is wrapping all of the required dependencies like Webpack, Babel for React project itself and then you need to focus on writing React code only. This tool sets up the development environment, provides an excellent developer experience, and optimizes the app for production.

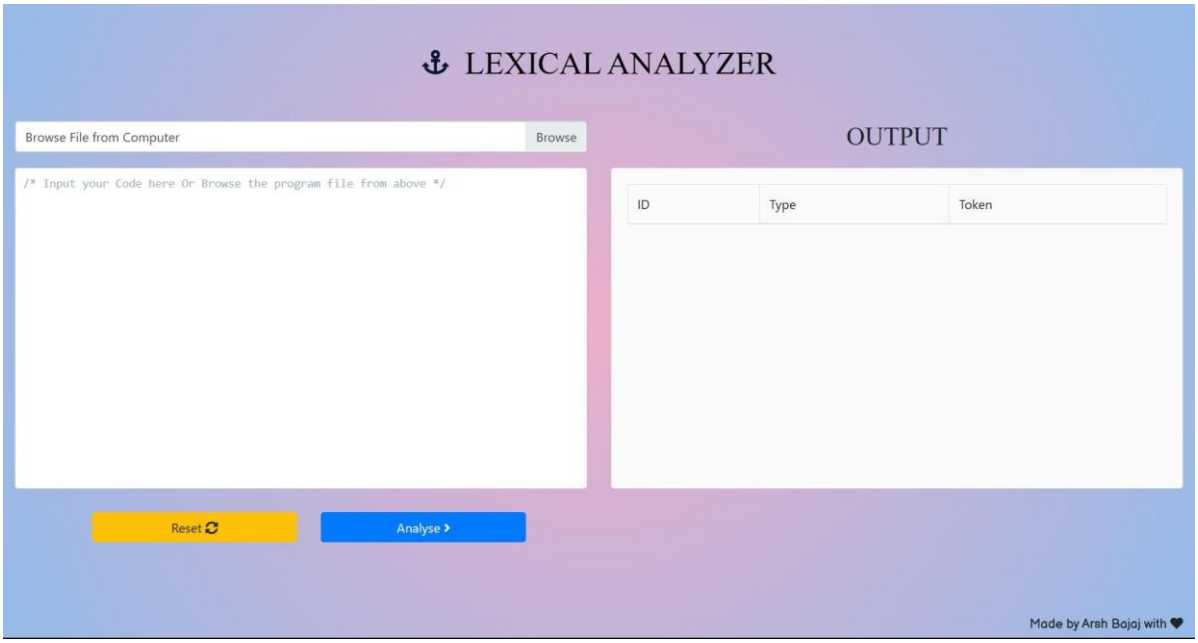
4.2.3 CSS:

CSS stands for Cascading Style Sheets. It is a language used for describing the presentation of a document written in HTML or XML, including the layout, colors, fonts, and other visual elements. With CSS, web developers can separate the content of a webpage from its presentation, allowing for more flexibility and control over the appearance of a website.

CSS works by selecting HTML elements and applying rules to them that define how they should be styled. These rules can be defined within the HTML document itself, or in a separate CSS file that is linked to the HTML document. CSS rules can target specific elements based on their type, class, or ID, as well as their position in the document tree.

CSS has evolved over the years, with new features and specifications being added regularly. Some of the latest features include support for responsive design, which allows web pages to adjust their layout and styling based on the size and orientation of the device they are being viewed on. CSS also includes support for animations and transitions, which can be used to create dynamic and engaging user interfaces.

PROJECT DESCRIPTION

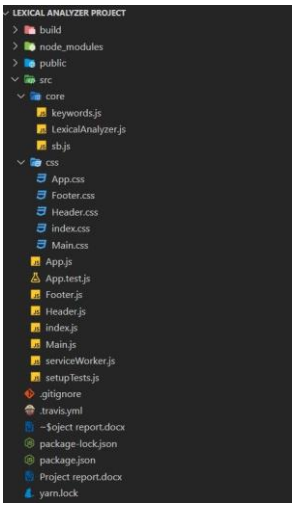


5.1 Problem Definition:

The project consists of developing and implementing a computer program that analyses a code and identify tokens from the code. This developed system will act as visualization for a phase of compiler design for study and research purposes. The system can able to provide solutions to hundreds of lines of code.

5.2 Overview of the Project:

This project is divided into several modules and all the modules are appearing in one class These modules are called from the main module located at the root of the project. All the modules have different as well as vital functionalities in order to develop the right product. These modules are designed in such a way that they reflect a highly cohesiveness.



5.3 Module Description:

5.3.1 Initialization:

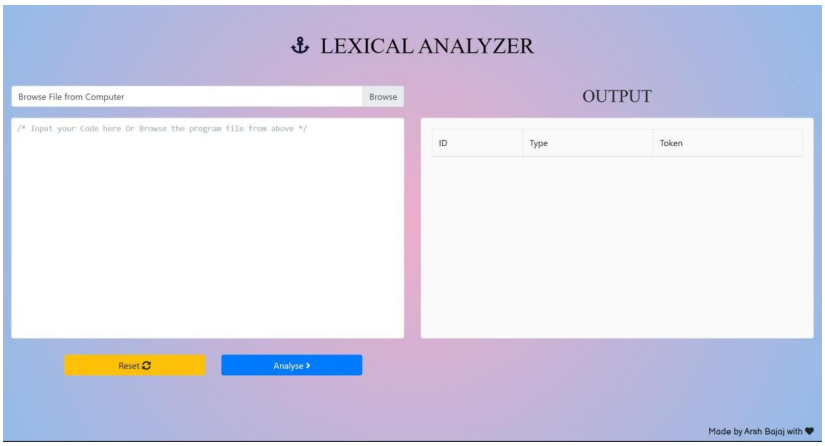
With the start argument, NPM will begin the process to make a development server available for your React application.
Here's a list of tasks for this script:

- Set the build environment into development for Node and Babel
- Ensure environment variables are read for the build process
- Verify the packages installed in your project are not outdated
- Check whether the code is in TypeScript or not
- Import required packages for the process, mostly Webpack-related modules
- Check for available port and host IP, defaults to 0.0.0.0:3000
- Run the compiler and listen for any messages from Webpack. Webpack will take care of using Babel, ESLint, and any other tools to prepare your code
- While Webpack is running, the script will open your browser and start the development server
- The development server created by WebpackDevServer will also create a listener for changes in your JavaScript file. When you make changes and save your JavaScript file, the development server will recompile your code and quickly refresh the browser.

5.3.2 Title Screen:

The title screen of the project has all the features a user may need to analyse their code lexically and obtain tokens out of it.

- Code input manually
- Code input through file
- Reset input button
- Code analyse button
- Output window



5.3.3 Analyse button:

Analyze button starts the process and runs the machine which then analyzes the code and returns an output which is displayed in the output section of the window

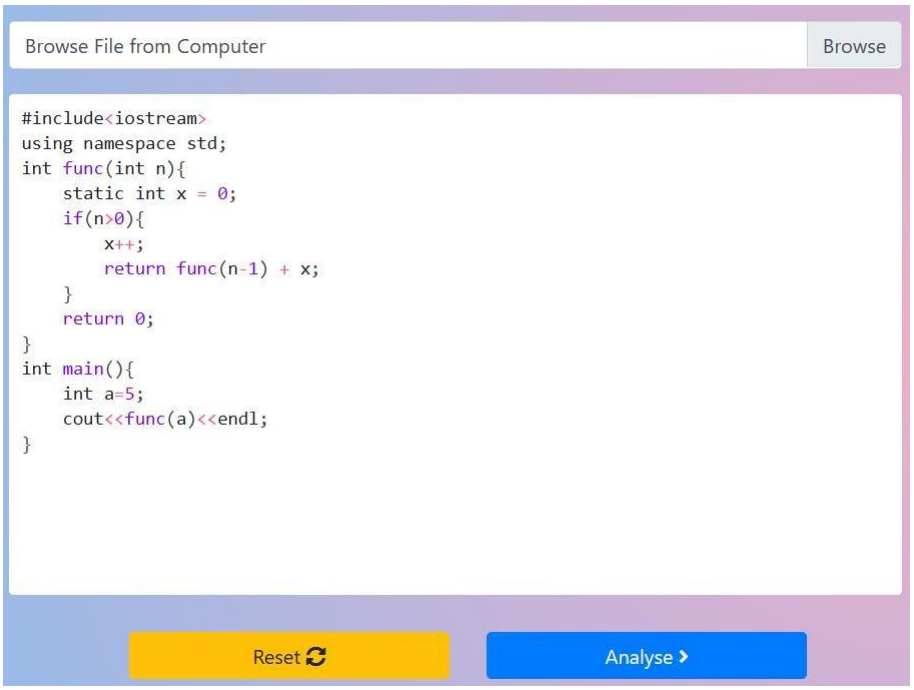
5.3.4 Reset button:

Reset button resets the code and deletes the input from the input window. To start a fresh new process.

5.3.5 Input Design:

There are two methods to input the code into the lexical analyzer -

1. Input code manually
2. Browsing the code file locally and selecting it from the computer



5.3.6 Output Design:

The output is displayed in the form of a table which as further columns -

1. ID
2. Type of token
3. Token

OUTPUT

ID	Type	Token
1	Identifier	include
2	Operator	<
3	Identifier	iostream
4	Operator	>
5	Identifier	using
6	Identifier	namespace
7	Identifier	std

CODE ANALYSIS

6.1 FRONT-END -

6.1.1 App.JS

```
import React from "react";
import Header from "./Header";
import Main from "./Main";
import Footer from "./Footer";

import "./css/App.css"
import "bootstrap/dist/css/bootstrap.min.css";

function App() {
  return (
    <div className="App container-fluid">
      <Header />
      <Main />
      <Footer />
    </div>
  );
}
export default App;
```

6.1.2 Index.JS

```
import React  from  'react';
import ReactDOM from 'react-dom';
import App from './App';
import * as serviceWorker from './serviceWorker';
```

```
ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
serviceWorker.unregister();
```

6.1.3 main.JS

```
import React from "react";
import analyzeString from "./core/LexicalAnalyzer";
import { CodeFlaskReact } from "react-codeflask";

import "./css/Main.css";
import "bootstrap/dist/css/bootstrap.min.css";
```

```
class Main extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      text: "/* Input your Code here Or Browse the program file from above */",
      result: [],
    };
    this.handleFileChange = this.handleFileChange.bind(this);
    this.handleTextChange = this.handleTextChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);
    this.handleReset = this.handleReset.bind(this);
  }
```

```
  handleFileChange() {
    let fileInput = document.getElementById("file-input");
    let file = fileInput.files[0];
    let reader = new FileReader();
    reader.addEventListener("loadend", () => {
      this.setState({
        text: reader.result,
      });
      fileInput.value = "";
    });
    reader.readAsText(file);
  }
```

```
  handleTextChange(code) {
    this.setState({
      text: code,
    });
  }
```

```
  handleSubmit() {
    let string = this.state.text;
    let result = analyzeString(string);
    this.setState({ result });
  }
```

```
  handleReset() {
    this.setState({ text: "/* Input your Code here Or Browse the program file from above */", result:
[] });
  }
```

```
  render() {
    return (
      <div className="main row">
```

```
        <div className="left-box col-lg-6 col-md-6 col-sm-6 col-xs-1">
          <div className="custom-file">
            <input
              className="custom-file-input"
              type="file"
              id="file-input"
              onChange={this.handleFileChange}
            />
            <label className="custom-file-label" data-browse="Browse">
              Browse File from Computer
            </label>
          </div>
```

```
          <CodeFlaskReact
            code={this.state.text}
            onChange={this.handleTextChange}
            id="code-editor"
            language="clike"
            fontSize={25}
            defaultTheme={false}
          />
```

```
          <div className="buttons row">
            <div className="col-lg-6 col-xs-1">
              <button className="btn btn-warning" onClick={this.handleReset}>
                Reset <i className="fas fa-sync-alt"></i>
              </button>
            </div>
            <div className="col-lg-6 col-xs-1">
              <button className="btn btn-primary" onClick={this.handleSubmit}>
                Analyse <i className="fas fa-angle-right"></i>
              </button>
            </div>
          </div>
        </div>
```

```
        <div className="right-box col-lg-6 col-md-6 col-sm-6 col-xs-1">
          <h2>OUTPUT</h2>
          <div className="token-table-container">
            <TokenTable tokens={this.state.result} />
          </div>
        </div>
      </div>
    );
  }
}
```

```
function TokenTable(props) {
  let typeStringsCHN = [
    "Error", //error
    "Keyword", //Keyword
    "Identifier", // Identifier
    "Operator", //Operator
    "Seperator", //separator
    "Constant",// Constant
    "Comment", // Comment
  ];
  const tokens = props.tokens.map((token) => {
    return (
      <tr key={token.key}>
        <td>{token.key + 1}</td>
        <td>{typeStringsCHN[token.type]}</td>
        <td>{token.content}</td>
```

```
      </tr>
    );
  });
```

```
  return (
    <table className="table table-bordered">
      <thead>
        <tr>
          <td>ID</td>
          <td>Type</td>
          <td>Token</td>
        </tr>
      </thead>
      <tbody>{tokens}</tbody>
    </table>
  );
}
export default Main;
```

6.1.3 Header.JS

```
import React from "react";
import "@fortawesome/fontawesome-free/css/all.css";
import "./css/Header.css";

export default function Header() {
  return (
    <div className="header">
      <i className="fas fa-anchor"></i>
      <h1>LEXICAL ANALYZER</h1>
    </div>
  );
}
```

6.1.4 Footer.js

index.html ● JS Footer.js M ✕ {} package.json

tokenator_lexical_analyzer_reactjs > src > JS Footer.js > Footer

```
1   import React from 'react'
2   import "./css/Footer.css"
3
4   export default function Footer() {
5     return (
6       <div className="footer">
7         <h6>
8           Made by Akshat Chaudhary with <i class="fas fa-heart fa-beat"></i>
9         </h6>
10      </div>
11    );
12  }
```

6.1.5 App.css

```
@import url('https://fonts.googleapis.com/css2?family=Balsamiq+Sans&family=ZCOOL+KuaiLe&display=swap');
.App {
  /* background-image: linear-gradient(120deg, #a1c4fd 0%, #c2e9fb 100%); */
  background: rgb(238,174,202);
  background: radial-gradient(circle, rgba(238,174,202,1) 0%, rgba(148,187,233,1) 100%);
  display: flex;
  width: 100%;
  height: 100vh;
  min-height: 750px;
  flex-direction: column;
}
```

6.1.6 Main css file

```
.main {
  flex: 4;
  height: 500px;
  display: flex;
  flex-direction: row;
}

.left-box {
  display: flex;
  flex-direction: column;
  align-content: center;
  align-items: center;
}
```

```
#code-editor {
  font-family: "Roboto Mono";
  background-color: #fafafa;
  width: 100%;
  height: 400px;
  margin: 20px 0;
  border-radius: .25rem;
  position: relative;
}
```

```
.codeflask {
  border-radius: .25rem;
  transition: all .05s ease-in;
  font-size: 20px !important;
}
```

```
.codeflask:focus-within {
  box-shadow: 0 0 0 5px rgb(129, 187, 253);
  -moz-outline-radius: .25rem;
}
```

```
.codeflask__flatten {
  font-size: 15px !important;
}
```

```
.buttons {
  width: 80%;
}
```

```
.buttons button {
  width: 100%;
  margin: 10px
}
```

```
.right-box h2 {
  user-select: none;
  text-align: center;
  margin-bottom: 20px;
  font-family: Abel;
}
```

```
.token-table-container {
  width: 100%;
  background-color: #fafafa;
  height: 400px;
  padding: 20px;
  border-radius: .25rem;
  overflow-y: scroll;
```

```
  scrollbar-width: none;
}
```

```
.token-box::-webkit-scrollbar {
  display: none;
}
```

```
::-webkit-scrollbar {
  display: none;
}
```

```
.token-table-container table {
  user-select: none;
}
```

6.1.6 Index css file

```
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}
```

6.1.7 Header css file

```
.header {
  flex: 1;
  width: 100%;
  display: flex;
  flex-direction: row;
  justify-content: center;
  align-items: center;
  color: black;
}
.header i {
  top: 50%;
  display: inline;
  font-size: 30px;
  margin: 10px;
  color: #0f1338;
}

.header h1 {
  top: 50%;
  display: inline;
  font-size: 40px;
  margin: 10px;
  user-select: none;
  font-family: 'Abel';
}
```

6.1.8 Footer css file

```
.footer {
  height: 60px;
  display: flex;
  flex-direction: row-reverse;
  justify-content: flex-start;
  align-items: flex-end;
  font-family: 'Balsamiq Sans';
  bottom: 15px;
  right: 15px;
  user-select: none;
}
```

6.2 BACKEND-END -

```
import "./keywords";
import { KEYWORDS } from "./keywords";
import FakeAutomata from "./sb";

function isKeyword(string) {
  if (KEYWORDS.indexOf(string) !== -1) return true;
  else return false;
}
```

```
function isIdentifier(string) {
  let statusTable = [
    [0, 0, 0],
    [0, 2, 0],
    [0, 2, 2],
  ];
  let finalStatus = [2];
  let transitions = [
    "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz_",
    "0123456789",
  ];
```

```
  return FakeAutomata({
    statusTable,
    finalStatus,
    transitions,
    string,
  });
}
```

```
function isArithmeticOperator(string) {
  let statusTable = [
    [0, 0, 0, 0],
    [0, 2, 3, 4],
    [0, 5, 0, 0],
    [0, 0, 6, 0],
    [0, 0, 0, 0],
    [0, 0, 0, 0],
    [0, 0, 0, 0],
  ];
  let finalStatus = [2, 3, 4, 5, 6];
  let transitions = [["+"], ["-"], ["*", "/", "%", "="]];
```

```
  return FakeAutomata({
    statusTable,
    finalStatus,
    transitions,
    string,
```

```
  });
}
```

```
function isLogicalOperator(string) {
  let statusTable = [
    [0, 0, 0],
    [0, 2, 3],
    [0, 0, 4],
    [0, 0, 4],
    [0, 0, 0],
  ];
  let finalStatus = [2, 4];
  let transitions = [["!", ">", "<"], ["="]];
```

```
  return FakeAutomata({
    statusTable,
    finalStatus,
    transitions,
    string,
  });
}
```

```
function isOperator(string) {
  return isArithmeticOperator(string) || isLogicalOperator(string);
}
```

```
function isUnsigned(string) {
  let statusTable = [
    [0, 0, 0, 0, 0],
    [0, 2, 0, 0, 0],
    [0, 2, 3, 5, 0],
    [0, 4, 0, 0, 0],
    [0, 4, 0, 5, 0],
    [0, 7, 0, 0, 6],
    [0, 7, 0, 0, 0],
    [0, 7, 0, 0, 0],

  ];
  let finalStatus = [2, 4, 7];
  let transitions = ["0123456789", ".", "Ee", "+-"];
```

```
  return FakeAutomata({
    statusTable,
    finalStatus,
    transitions,
    string,
  });
}
```

```
function isIntenger(string) {
  let statusTable = [
    [0, 0, 0, 0, 0, 0],
    [0, 3, 2, 0, 0, 0],
    [0, 2, 2, 0, 0, 0],
    [0, 0, 0, 4, 5, 0],
    [0, 0, 6, 0, 0, 6],
    [0, 5, 0, 0, 5, 0],
    [0, 6, 6, 0, 0, 6],

  ];
  let finalStatus = [2, 3, 5, 6];
  let transitions = ["0", "123456789", "x", "1234567", "abcdefABCDEF"];
```

```
  return FakeAutomata({
    statusTable,
```

```
    finalStatus,
    transitions,
    string,
  });
}
```

```
function isConstant(string) {
  return isUnsigned(string) || isIntenger(string);
}
```

```
function isDelimiter(string) {
  if (string.length > 1) return false;
  let delimiters = ",{}[]();";
  return delimiters.indexOf(string) !== -1 ? true : false;
}
```

```
function isComment(string) {
  let statusTable = [
    [0, 0, 0, 0],
    [0, 2, 0, 0],
    [0, 0, 3, 0],
    [0, 0, 4, 3],
    [0, 5, 4, 3],
    [0, 0, 0, 0],
  ];
  let finalStatus = [5];
  let transitions = ["/", "*", "_others"];
```

```
  return FakeAutomata({
    statusTable,
    finalStatus,
    transitions,
    string,
  });
}
```

```
function analyzeWord(word) {
  if (word.length === 0) {
    return 0;
  } else if (isKeyword(word)) {
    return 1;
  } else if (isIdentifier(word)) {
    return 2;
  } else if (isOperator(word)) {
    return 3;
  } else if (isDelimiter(word)) {
    return 4;
  } else if (isConstant(word)) {
    return 5;
  } else if (isComment(word)) {
    return 6;
  } else {
    return 0;
  }
}
```

```
/**
 * 读取字符串，返回一个json 数组
 * @param 字符串
 * @returns 一个json 数组
 */
function analyzeString(string) {
  let content = string.replace(/[\r\n]/g, "");
```

```
  let result = [];

  while (content.length > 0) {
    let buffer = content.slice();
    let index = content.length - 1;

    for (; analyzeWord(buffer) === 0 && index > 0; index--) {
      buffer = buffer.substring(0, buffer.length - 1);
    }

    if (index === 0 && analyzeWord(buffer) === 0) {
      content = content.substring(1);
    } else {
      result.push({
        type: analyzeWord(buffer),
        content: buffer,
        key: result.length
      });
      content = content.substring(index + 1);
    }
  }

  return result;
}

export default analyzeString;
```

REFERENCES -

7.1 Websites referred -

- GeeksForGeeks
- ResearchGate
- TutorialsPoint
- Hubspot blog
- FreeCodeCamp
- MDN Documentation
- Github