Advanced Programming:

|  |  |  |  |
| --- | --- | --- | --- |
| **Lab No** | 11 | **Reg. No** | 237885 |
| **Student Name** | Ershad Hussain | **Section** | A |

App.js:

import React, { Component } from 'react';

import {View,Text,ToastAndroid, ScrollView} from 'react-native';

//Styles

import styles from './styles';

//Custom Components

import NumberButtons from './components/NumberButtons';

import HistoryView from './components/HistoryView'

//constants

const buttons = [

  ['CLEAR', 'DEL'],

  ['7', '8', '9', '÷'],

  ['4', '5', '6', 'x'],

  ['1', '2', '3', '+'],

  ['.', '0', '=','-']

]

const initialOutput = '0';

const maxLength = 57;

//Serves as the Container Class

class App extends Component {

  //Initialization

  constructor(props){

      super(props);

      this.state = {

          \_output: initialOutput,

          \_mathExpression: '',

          \_history: [],

      }

      this.\_handleEvent = this.\_handleEvent.bind(this);

      this.\_clearHistory = this.\_clearHistory.bind(this);

  }

  //Handles actions on button press

  \_handleEvent = (value) => {

    if(!isNaN(value) || value=='.'){

      this.\_concatToOutput(value);

    }

    else{

      switch(value) {

        case buttons[0][0]:

          this.\_initOutput();

          break;

        case buttons[0][1]:

          if (this.state.\_output.length === 1){

            this.\_initOutput();

          }

          else {

            this.\_replaceLastIndex('');

          }

          break;

        case buttons[4][2]:

          this.\_evaluate();

          break;

        default:

          let strLastChar = this.state.\_output.slice(-1);

          if(isNaN(strLastChar)){

            this.\_replaceLastIndex(value)

          }

          else{

            this.\_concatToOutput(value);

          }

          break;

      }

    }

  }

  //Function to concat user input to output screen

  \_concatToOutput = (value) => {

    if(this.state.\_output.length>=maxLength){

      this.\_showToast('Maximum Expression Length of ' + maxLength + ' is reached.');

    }

    else{

      if(this.state.\_output !== initialOutput){

        this.setState({\_output: this.state.\_output + '' + value + ''})

      }

      else{

        this.setState({\_output: value + ''})

      }

    }

  }

  //Function to replace the last index of the output

  \_replaceLastIndex = (value) => {

    var str1 = this.state.\_output.replace(/.$/,value)

    this.setState({

      \_output: str1

    })

  }

  //Validate and Calculate the output state as a Mathematical expression

  \_evaluate = () => {

    try{

      let strCurOutput = this.state.\_output;

      if(isNaN(strCurOutput)){

        let dEval = eval(this.\_convertToMathExpression(this.state.\_output));

        let aHistory = [...this.state.\_history];

        aHistory.push([strCurOutput, dEval])

        this.setState({

          \_output: ''+dEval,

          \_history: aHistory

        })

      }

    }

    catch(exception){

      /\* console.log('exception: ' + exception); \*/

      this.\_showToast('Invalid format used.');

    }

  }

  //Function to convert the output state into a valid mathematical expression

  \_convertToMathExpression = (value) => {

     let strTemp = value.replace(new RegExp(this.\_escapeRegExp(buttons[1][3]), 'g'), '/');

     strTemp = strTemp.replace(new RegExp(this.\_escapeRegExp(buttons[2][3]), 'g'), '\*');

    return strTemp;

  }

  \_escapeRegExp = (str) => {

    return str.replace(/([.\*+?^=!:${}()|\[\]\/\\])/g, "\\$1");

  }

  //Function to initialize output state

  \_initOutput = () => {

    this.setState({

      \_output: initialOutput

    })

  }

  //Function to clear the history

  \_clearHistory = () => {

    console.log('inside \_clearHistory function');

    const emptyArray = [];

    this.setState({

      \_history: emptyArray

    })

  }

  //Function to display an android toast

  \_showToast = (value) => {

    ToastAndroid.show(value, ToastAndroid.SHORT);

  }

  render() {

    return (

      <View style={styles.container}>

        <View style={styles.contHistory}>

          <HistoryView data={this.state.\_history} onClear={this.\_clearHistory}/>

        </View>

        <View style={styles.contOutput}>

          <View style={styles.placeHolderOutput}>

            <Text style={styles.txtDefault}>{this.state.\_output}</Text>

          </View>

        </View>

        <View style={styles.contButtons}>

          <NumberButtons onBtnPress={this.\_handleEvent} buttons={buttons}/>

        </View>

      </View>

    );

  }

}

export default App

Output:

