**CODE:-**

**File Name: - App.css**

\*, \*::before, \*::after {

box-sizing:border-box;

}

body{

margin: 0;

background: rgb(25,0,36);

background: linear-gradient(90deg, rgba(25,0,36,1) 0%, rgba(41,9,121,1) 30%, rgba(210,0,255,1) 100%);

max-width: 500px;

margin: auto;

font-family: 'Inconsolata', monospace;

}

h1{

width: 400px;

color: white;

height: 50px;

font-family: 'Inconsolata', monospace;

margin: 0%;

text-align: center;

}

.calculator-grid{

display: grid;

margin-top: 2rem;

grid-template-columns: repeat(4,6rem);

grid-template-rows: minmax(7rem,auto) repeat(5,6rem);

justify-content: center;

}

.calculator-grid > button {

opacity: 0.5;

cursor: pointer;

font-size: 2rem;

border: 1px solid white;

outline: none;

background-color: rgb(252, 252, 252);

}

.calculator-grid >button:hover,

.calculator-grid >button:focus{

background-color: rgba(255,255,255,.9);

}

.span-two{

grid-column: span 2;

}

.output{

opacity: 0.9;

grid-column: 1 / -1;

background-color: rgba(255, 255, 255, 0.75);

display:flex ;

flex-direction: column;

align-items: flex-end;

justify-content: space-around;

padding: .75rem;

word-wrap: break-word;

word-break: break-all;

}

.output .previous-operand {

color: rgba(0, 0, 0, 0.75);

font-size: 1.5rem;

}

.output .current-operand {

color: rgb(0, 0, 0);

font-size: 2.5rem;

}

**File Name: - App.js**

import "./App.css";

import React, { useReducer } from "react";

import DigitButton from "./DigitButton";

import OperationButton from "./OperationButton";

export const ACTIONS = {

ADD\_DIGIT: "add-digit",

CHOOSE\_OPERATION: "choose-operation",

CLEAR: "clear",

DELETE\_DIGIT: "delete-digit",

EVALUATE: "evaluate",

};

function reducer(state, { type, payload }) {

switch (type) {

case ACTIONS.ADD\_DIGIT:

if (state.overwrite) {

return {

...state,

currentOperand:payload.digit,

overwrite: false,

}

}

if (payload.digit === "0" && state.currentOperand === "0") {

return state;

}

if (payload.digit === "." && state.currentOperand.includes(".")) {

return state;

}

return {

...state,

currentOperand: `${state.currentOperand || ""}${payload.digit}`,

};

case ACTIONS.CHOOSE\_OPERATION:

if (state.currentOperand == null && state.previousOperand == null) {

return state;

}

if (state.currentOperand == null) {

return {

...state,

operation: payload.operation,

}

}

if (state.previousOperand == null) {

return {

...state,

operation: payload.operation,

previousOperand: state.currentOperand,

currentOperand: null,

};

}

return {

...state,

previousOperand: evaluate(state),

operation: payload.operation,

currentOperand: null,

};

case ACTIONS.CLEAR:

return {};

case ACTIONS.DELETE\_DIGIT:

if (state.overwrite) {

return {

...state,

overwrite: false,

currentOperand: null

}

}

if (state.currentOperand == null) return state

if (state.currentOperand.length === 1){

return { ...state, currentOperand: null}

}

return {

...state,

currentOperand: state.currentOperand.slice(0, -1)

}

case ACTIONS.EVALUATE:

if (

state.operation == null ||

state.currentOperand == null ||

state.previousOperand == null

) {

return state

}

return {

...state,

overwrite: true,

previousOperand: null,

operation: null,

currentOperand: evaluate(state),

}

default:

return {}

}

}

function evaluate({ currentOperand, previousOperand, operation }) {

const prev = parseFloat(previousOperand);

const current = parseFloat(currentOperand);

if (isNaN(prev) || isNaN(current)) return "";

let computation = "";

switch (operation) {

case "+":

computation = prev + current;

break;

case "-":

computation = prev - current;

break;

case "\*":

computation = prev \* current;

break;

case "÷":

computation = prev / current;

break;

default:

return {}

}

return computation.toString()

}

const INTEGER\_FORMATTER = new Intl.NumberFormat("en-us", {

maximumFractionDigits: 0,

})

function formatOperand(operand) {

if(operand == null) return

const [integer,decimal] = operand.split('.')

if(decimal == null) return INTEGER\_FORMATTER.format(integer)

return `${INTEGER\_FORMATTER.format(integer)}.${decimal}`

}

function App() {

const [{ currentOperand, previousOperand, operation }, dispatch] = useReducer(

reducer,

{}

);

return (

<div className="calculator-grid">

<h1>Calculator App</h1>

<div className="output">

<div className="previous-operand">

{formatOperand(previousOperand)} {operation}

</div>

<div className="current-operand">{formatOperand(currentOperand)}</div>

</div>

<button

className="span-two"

onClick={() => {

dispatch({ type: ACTIONS.CLEAR });

}}

>

AC

</button>

<button onClick={() => dispatch({type: ACTIONS.DELETE\_DIGIT})}>DEL</button>

<OperationButton operation="÷" dispatch={dispatch} />

<DigitButton digit="1" dispatch={dispatch} />

<DigitButton digit="2" dispatch={dispatch} />

<DigitButton digit="3" dispatch={dispatch} />

<OperationButton operation="\*" dispatch={dispatch} />

<DigitButton digit="4" dispatch={dispatch} />

<DigitButton digit="5" dispatch={dispatch} />

<DigitButton digit="6" dispatch={dispatch} />

<OperationButton operation="+" dispatch={dispatch} />

<DigitButton digit="7" dispatch={dispatch} />

<DigitButton digit="8" dispatch={dispatch} />

<DigitButton digit="9" dispatch={dispatch} />

<OperationButton operation="-" dispatch={dispatch} />

<DigitButton digit="." dispatch={dispatch} />

<DigitButton digit="0" dispatch={dispatch} />

<button className="span-two" onClick={() => dispatch({type: ACTIONS.EVALUATE})}>=</button>

</div>

);

}

export default App;

**File Name: - DigitButton.js**

import { ACTIONS } from "./App"

export default function DigitButton({ dispatch, digit }) {

return(

<button onClick={() => dispatch({ type: ACTIONS.ADD\_DIGIT, payload: {digit}})}>{digit}</button>

)}

**File Name: - index.css**

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;

}

**File Name: - index.js**

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

ReactDOM.render(

<React.StrictMode>

<App />

</React.StrictMode>,

document.getElementById('root')

);

**File Name: - OperationButton.js**

import { ACTIONS } from "./App"

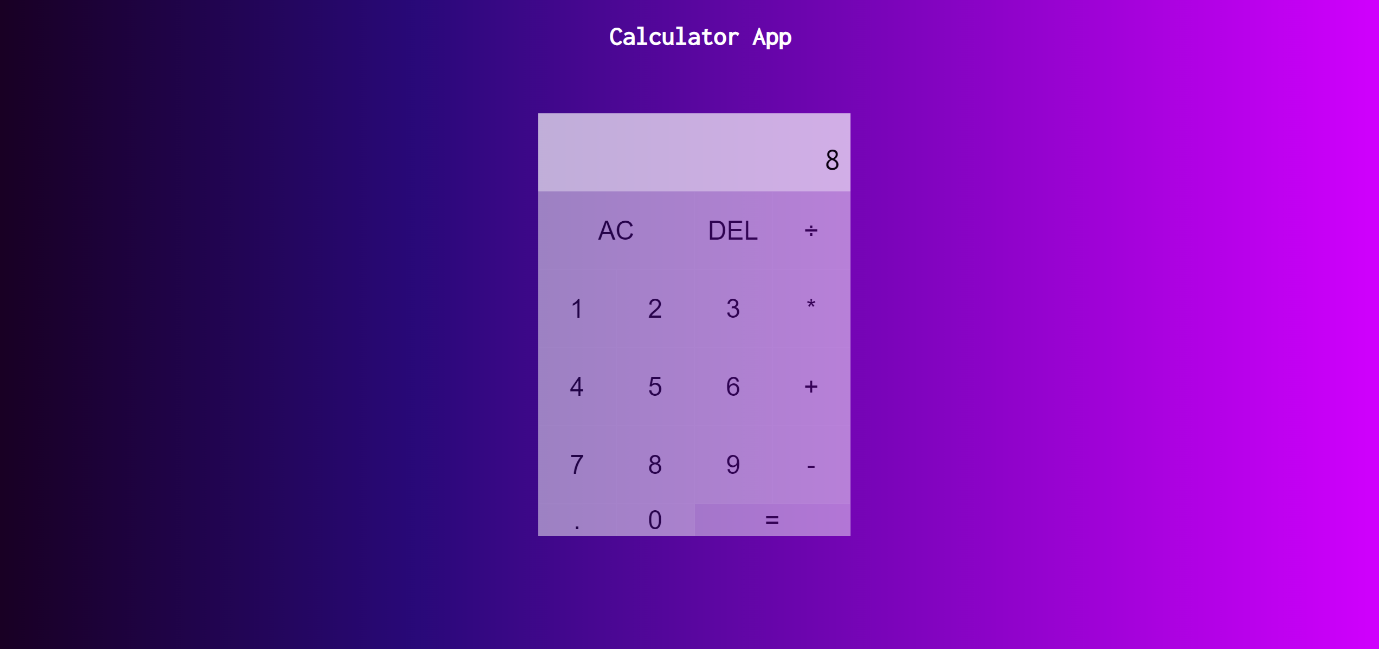
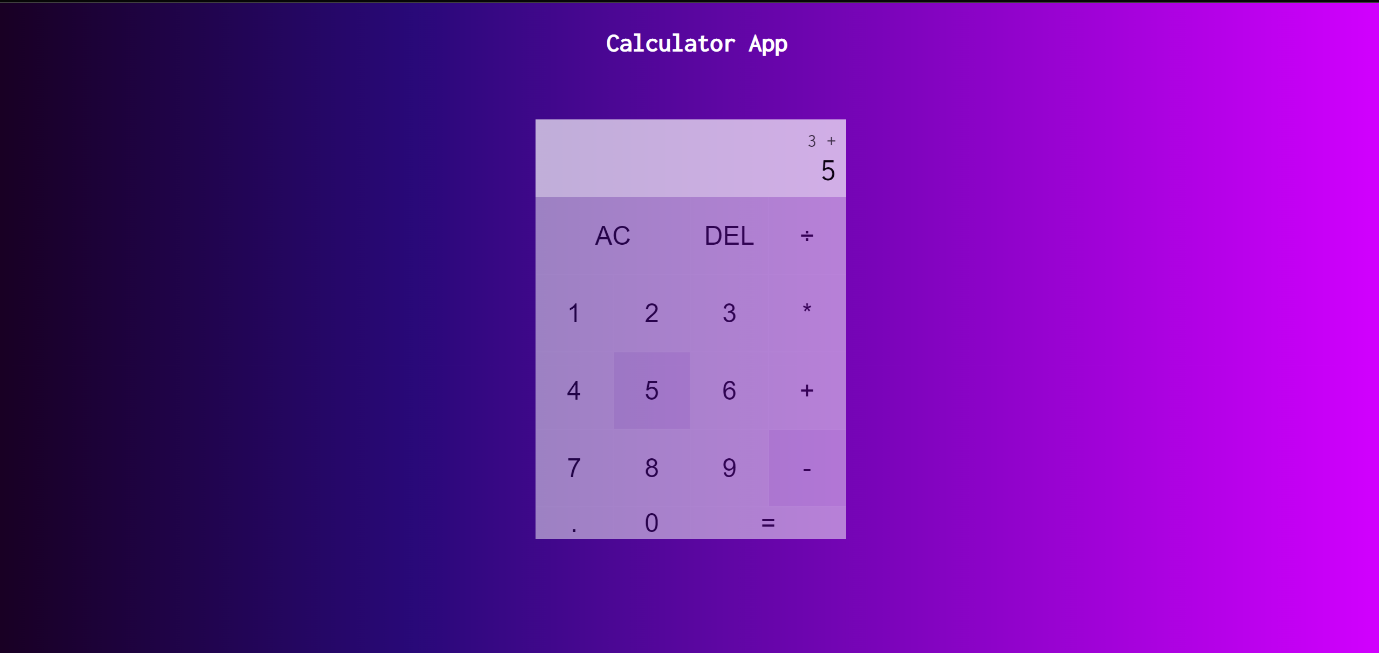
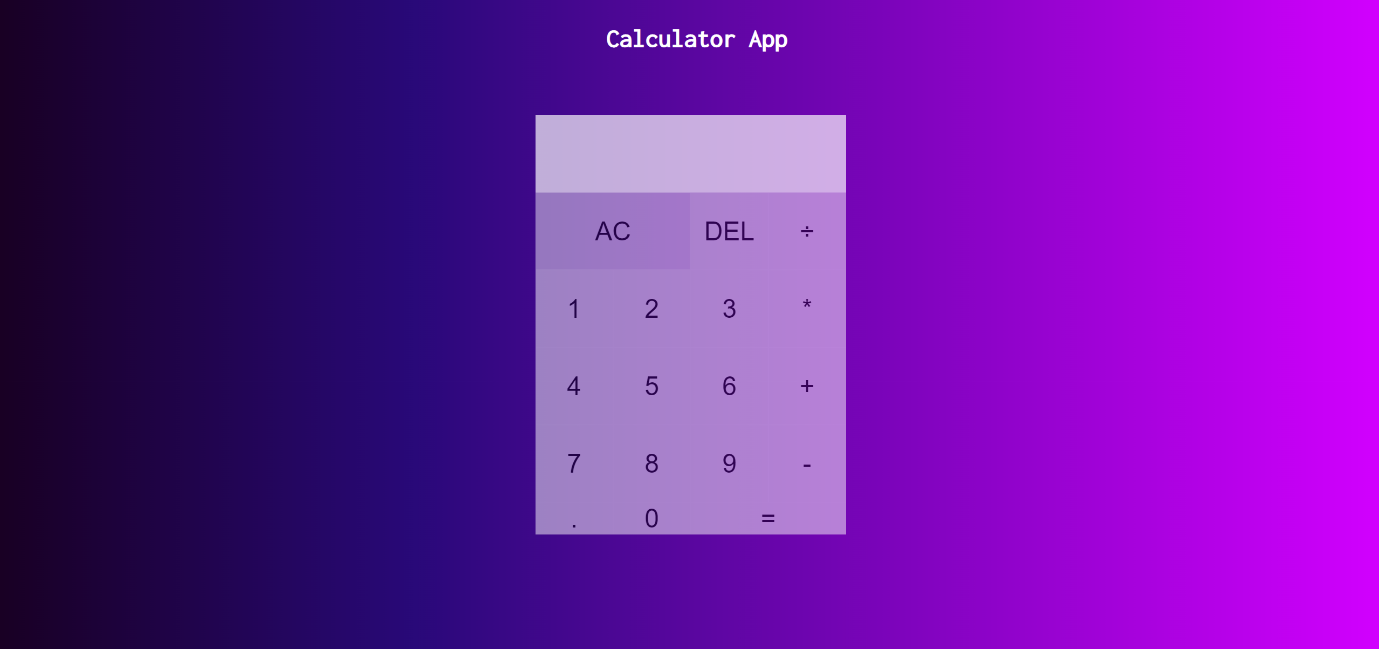
export default function OperationButton({ dispatch, operation }) {

return(

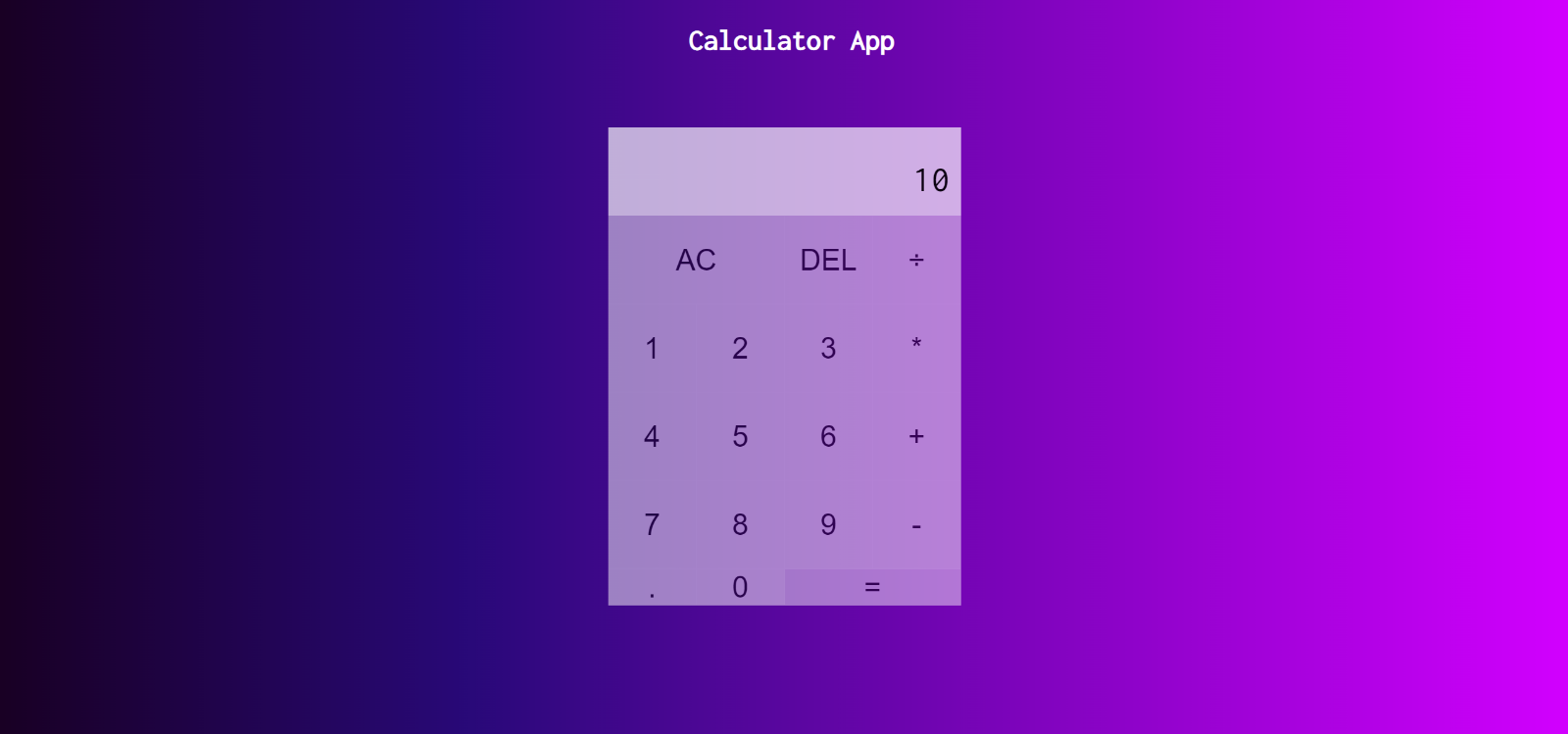
<button onClick={() => dispatch({ type: ACTIONS.CHOOSE\_OPERATION, payload: {operation}})}>

{operation}</button>)}

**Output:-**



Calendar

Description automatically generated

A screenshot of a computer

Description automatically generated

