**MENGENAL APLIKASI KALKULATOR**

**1.1. Defenisi aplikasi kalkulator**

Kebanyakan kalkulator saat ini memerlukan listrik untuk beroperasi atau kalkulator bertenaga baterai. Kalkulator bekerja dengan menjalankan fungsi terprogram berdasarkan masukan numerik.

Sebelum kalkulator elektronik (sekitar tahun 1970), kalkulator yang lebih primitif, yaitu mistar hitung, sudah umum digunakan. Ini terdiri dari bilah kayu yang disebut perosotan yang dapat dipindahkan masuk dan keluar dari sepasang bilah yang diperkuat. Slide dan sepasang bilah luar telah mengkalibrasi skala numerik.

**1.2. Manfaat dan Tujuan Aplikasi Kalkulator**

Manfaat dan Tujuan dibentuknya Kalkulator yaitu antara lain:

1. Untuk membantu manusia belajar terutama mahasiswa dan mahasiswi.
2. Membantu menunjang pekerjaan terutama pekerjaan yang ada di divisi keuangan.
3. Membantu perencanaan keuangan seseorang dalam [mengelola aset](https://ajaib.co.id/penilaian-tingkat-risiko-dan-alokasi-aset/" \t "_blank) atau harta Salah satunya adalah menabung dan investasi.

**1.3. Bahasa Pemrograman Yang Dipakai**

Bahasa pemrograman yang dipakai antara lain:

1. Bahasa pemrograman HTML.
2. Bahasa pemrograman CSS.
3. Bahasa Pemrograman JavaScript.

**KONSEP ALGORITMA & PEMODELAN APLIKASI**

**2.1. Flow Chart Perancangan Aplikasi**

## Pengertian flowchart

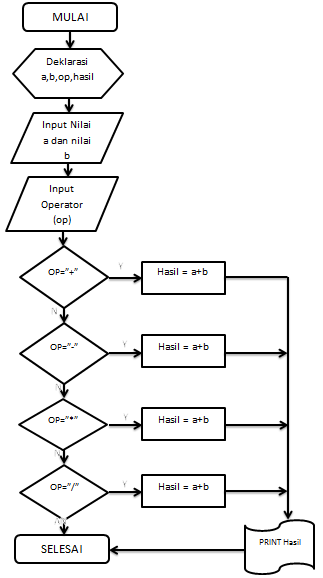
Flowchart atau bagan alur adalah diagram yang menampilkan langkah-langkah dan keputusan untuk melakukan sebuah proses dari suatu program. Setiap langkah di gambarkan dalam bentuk diagram dan dihubungkan dengan garis atau arah panah.

Flowchart berperan penting dalam memutuskan sebuah langkah atau fungsionalitas dari sebuah proyek pembuatan program yang melibatkan banyak orang sekaligus. Selain itu dengan menggunakan bagan alur proses dari sebuah program akan lebih jelas, ringkas, dan mengurangi kemungkinan untuk salah penafsiran. Penggunaan flowchart dalam dunia pemrograman juga merupakan cara yang bagus untuk menghubungkan antara kebutuhan teknis dan non-teknis.

## Simbol flowchart

Pada dasarnya simbol-simbol dalam flowchart memiliki arti yang berbeda-beda. Berikut adalah simbol-simbol yang sering digunakan dalam proses pembuatan flowchart.





**2.2. Use Case Diagram Aplikasi Kalkulator**

**Pengertian Use Case**

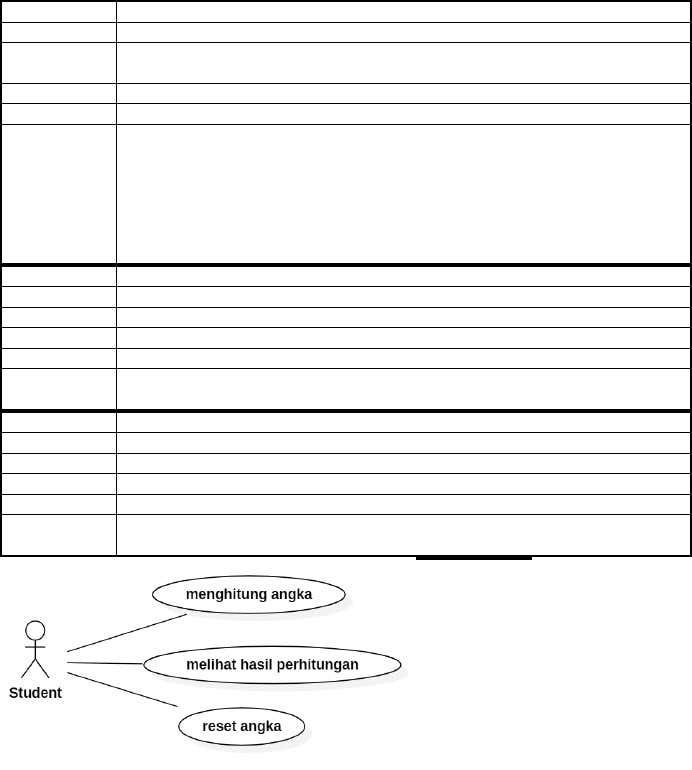
Use case diagram adalah satu dari berbagai jenis diagram UML (Unified Modelling Language) yang menggambarkan hubungan interaksi antara sistem dan aktor. Use Case dapat mendeskripsikan tipe interaksi antara si pengguna system dengan sistemnya.

Tentunya, use case diagram merupakan sesuatu yang mudah dipelajari. Langkah awal untuk melakukan pemodelan, tentu perlu nya suatu diagram yang mampu menjabarkan aksi actor dengan aksi system itu sendiri, seperti yang terdapat pada use case diagram.

## Simbol-Simbol pada Use Case Diagram



**Use Case Diagram Aplikasi Kalkulator**



**2.3. Activity Diagram Aplikasi**

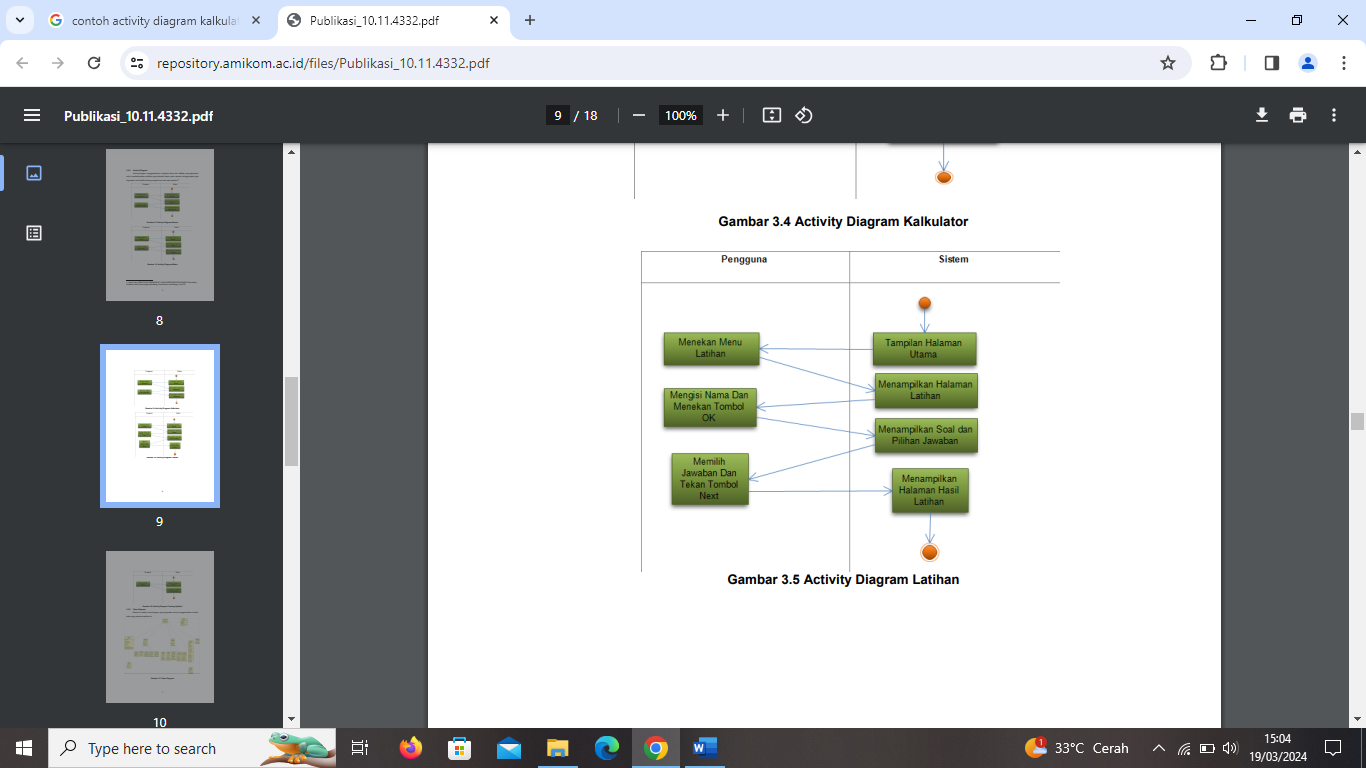
**Pengertian Activity Diagram**

Activity diagram, dalam bahasa Indonesia diagram aktivitas, yaitu diagram yang dapat memodelkan proses-proses yang terjadi pada sebuah sistem. Runtutan proses dari suatu system digambarkan secara vertikal. Activity diagram merupakan pengembangan dari Use Case yang memiliki alur aktivitas.

Alur atau aktivitas berupa runtutan menu-menu atau proses bisnis yang terdapat di dalam system tersebut. Dalam buku Rekayasa Perangkat Lunak karangan Rosa A.S mengatakan, “Diagram aktivitas tidak menjelaskan kelakuan aktor. Dapat diartikan bahwa dalam pembuatan activity diagram hanya dapat dipakai untuk menggambarkan alur kerja atau aktivitas system saja.”

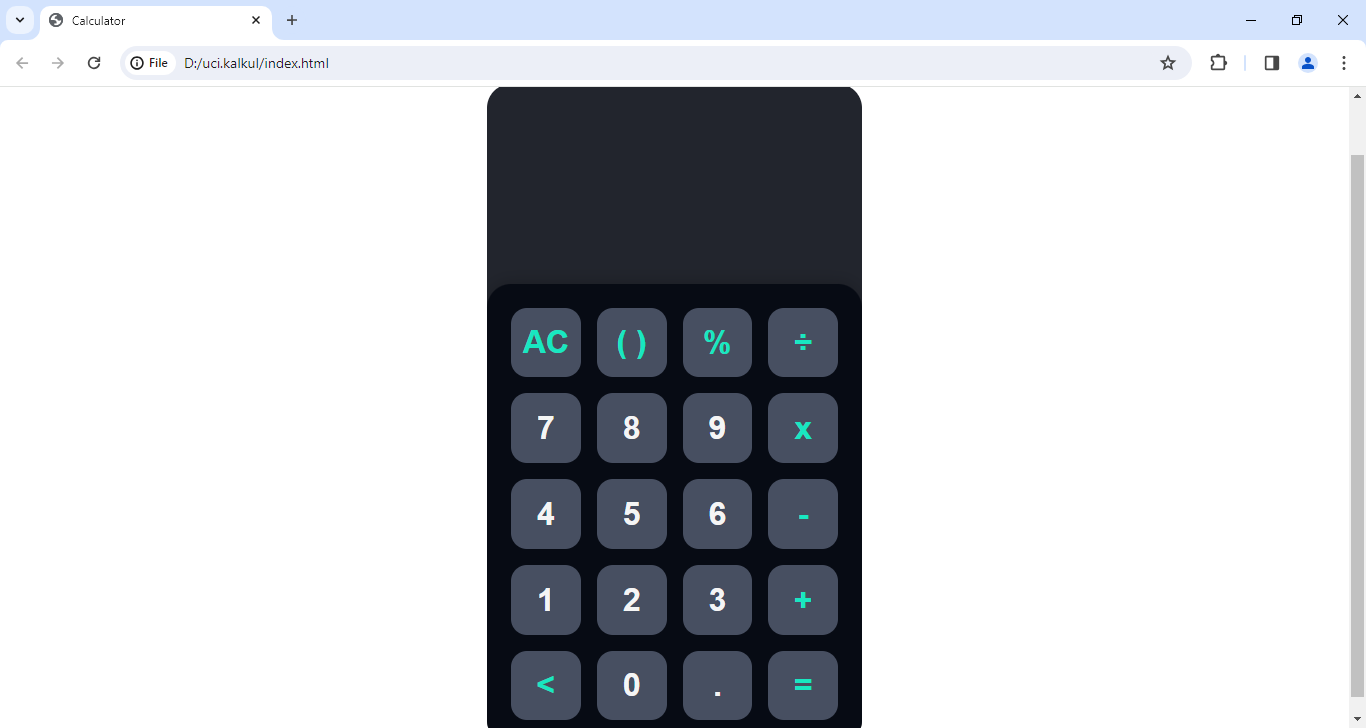
## ****Komponen Activity Diagram****





**DESAIN RANCANGAN USER INTERFACE/USER EXPERIENCE (UI/UX)**

**3.1. Aplikasi Kalkulator (Desain +Code)**



**3.2. Code AplikasiKalkulator**

**3.2.3. Code HTML**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Calculator</title>

    <link rel="stylesheet" href="main.css" />

</head>

<body>

    <div class="app">

        <div class="calculator">

            <div class="display">

                <div class="content">

                    <div class="input"></div>

                    <div class="output"></div>

                </div>

            </div>

            <div class="keys">

                <div data-key="clear" class="key action">

                    <span>AC</span>

                </div>

                <div data-key="brackets" class="key action">

                    <span>( )</span>

                </div>

                <div data-key="%" class="key action">

                    <span>%</span>

                </div>

                <div data-key="/" class="key operator">

                    <span>÷</span>

                </div>

                <div data-key="7" class="key">

                    <span>7</span>

                </div>

                <div data-key="8" class="key">

                    <span>8</span>

                </div>

                <div data-key="9" class="key">

                    <span>9</span>

                </div>

                <div data-key="\*" class="key operator">

                    <span>x</span>

                </div>

                <div data-key="4" class="key">

                    <span>4</span>

                </div>

                <div data-key="5" class="key">

                    <span>5</span>

                </div>

                <div data-key="6" class="key">

                    <span>6</span>

                </div>

                <div data-key="-" class="key operator">

                    <span>-</span>

                </div>

                <div data-key="1" class="key">

                    <span>1</span>

                </div>

                <div data-key="2" class="key">

                    <span>2</span>

                </div>

                <div data-key="3" class="key">

                    <span>3</span>

                </div>

                <div data-key="+" class="key operator">

                    <span>+</span>

                </div>

                <div data-key="backspace" class="key action">

                    <span>&lt;</span>

                </div>

                <div data-key="0" class="key">

                    <span>0</span>

                </div>

                <div data-key="." class="key">

                    <span>.</span>

                </div>

                <div data-key="=" class="key operator">

                    <span>=</span>

                </div>

            </div>

        </div>

    </div>

    <script src="main.js"></script>

</body>

</html>

**3.2.4. Code CSS**

\* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

    font-family: sans-serif;

}

.app {

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

}

.calculator {

    background-color: #22252D;

    width: 100%;

    max-width: 375px;

    min-height: 640px;

    display: flex;

    flex-direction: column;

    border-radius: 1.5rem;

    overflow: hidden;

    margin-top: 10%;

}

.display {

    min-height: 200px;

    padding: 1.5rem;

    display: flex;

    justify-content: flex-end;

    align-items: flex-end;

    color: white;

    text-align: right;

    flex: 1 1 0%;

}

.display .content {

    display: flex;

    flex-direction: column;

    justify-content: flex-end;

    width: 100%;

    max-width: 100%;

    overflow: auto;

}

.display .input {

    width: 100%;

    font-size: 1.25rem;

    margin-bottom: 0.5rem;

}

.display .output {

    font-size: 3rem;

    font-weight: 700;

    width: 100%;

    white-space: nowrap;

}

.display .operator {

    color: #18e7c1;

}

.display .brackets,

.display .percent {

    color: #18e7c1;

}

.keys {

    background-color: #070b14;

    padding: 1.5rem;

    border-radius: 1.5rem 1.5rem 0 0;

    display: grid;

    grid-template-columns: repeat(4, 1fr);

    grid-template-rows: repeat(5, 1fr);

    grid-gap: 1rem;

    box-shadow: 0px -2px 16px rgba(0, 0, 0, 0.2);

}

.keys .key {

    position: relative;

    cursor: pointer;

    display: block;

    height: 0;

    padding-top: 100%;

    background-color: #474f61;

    border-radius: 1rem;

    transition: 0.2s;

    user-select: none;

}

.keys .key span {

    position: absolute;

    top: 50%;

    left: 50%;

    transform: translate(-50%, -50%);

    font-size: 2rem;

    font-weight: 700;

    color: whitesmoke;

}

.keys .key:hover {

    box-shadow: inset 0px 0px 8px rgba(0, 0, 0, 0.2);

}

.keys .key.operator span {

    color: #18e7c1;

}

.keys .key.action span {

    color: #18e7c1;

}

**3.2.5. Code Script.js**

const keys = document.querySelectorAll('.key');

const display\_input = document.querySelector('.display .input');

const display\_output = document.querySelector('.display .output');

let input = "";

for (let key of keys) {

    const value = key.dataset.key;

    key.addEventListener('click', () => {

        if (value == "clear") {

            input = "";

            display\_input.innerHTML = "";

            display\_output.innerHTML = "";

        } else if (value == "backspace") {

            input = input.slice(0, -1);

            display\_input.innerHTML = CleanInput(input);

        } else if (value == "=") {

            let result = eval(PerpareInput(input));

            display\_output.innerHTML = CleanOutput(result);

        } else if (value == "brackets") {

            if (

                input.indexOf("(") == -1 ||

                input.indexOf("(") != -1 &&

                input.indexOf(")") != -1 &&

                input.lastIndexOf("(") < input.lastIndexOf(")")

            ) {

                input += "(";

            } else if (

                input.indexOf("(") != -1 &&

                input.indexOf(")") == -1 ||

                input.indexOf("(") != -1 &&

                input.indexOf(")") != -1 &&

                input.lastIndexOf("(") > input.lastIndexOf(")")

            ) {

                input += ")";

            }

            display\_input.innerHTML = CleanInput(input);

        } else {

            if (ValidateInput(value)) {

                input += value;

                display\_input.innerHTML = CleanInput(input);

            }

        }

    })

}

function CleanInput(input) {

    let input\_array = input.split("");

    let input\_array\_length = input\_array.length;

    for (let i = 0; i < input\_array\_length; i++) {

        if (input\_array[i] == "\*") {

            input\_array[i] = ` <span class="operator">x</span> `;

        } else if (input\_array[i] == "/") {

            input\_array[i] = ` <span class="operator">÷</span> `;

        } else if (input\_array[i] == "+") {

            input\_array[i] = ` <span class="operator">+</span> `;

        } else if (input\_array[i] == "-") {

            input\_array[i] = ` <span class="operator">-</span> `;

        } else if (input\_array[i] == "(") {

            input\_array[i] = `<span class="brackets">(</span>`;

        } else if (input\_array[i] == ")") {

            input\_array[i] = `<span class="brackets">)</span>`;

        } else if (input\_array[i] == "%") {

            input\_array[i] = `<span class="percent">%</span>`;

        }

    }

    return input\_array.join("");

}

function CleanOutput (output) {

    let output\_string = output.toString();

    let decimal = output\_string.split(".")[1];

    output\_string = output\_string.split(".")[0];

    let output\_array = output\_string.split("");

    if (output\_array.length > 3) {

        for (let i = output\_array.length - 3; i > 0; i -= 3) {

            output\_array.splice(i, 0, ",");

        }

    }

    if (decimal) {

        output\_array.push(".");

        output\_array.push(decimal);

    }

    return output\_array.join("");

}

function ValidateInput (value) {

    let last\_input = input.slice(-1);

    let operators = ["+", "-", "\*", "/"];

    if (value == "." && last\_input == ".") {

        return false;

    }

    if (operators.includes(value)) {

        if (operators.includes(last\_input)) {

            return false;

        } else {

            return true;

        }

    }

    return true;

}

function PerpareInput (input) {

    let input\_array = input.split("");

    for (let i = 0; i < input\_array.length; i++) {

        if (input\_array[i] == "%") {

            input\_array[i] = "/100";

        }

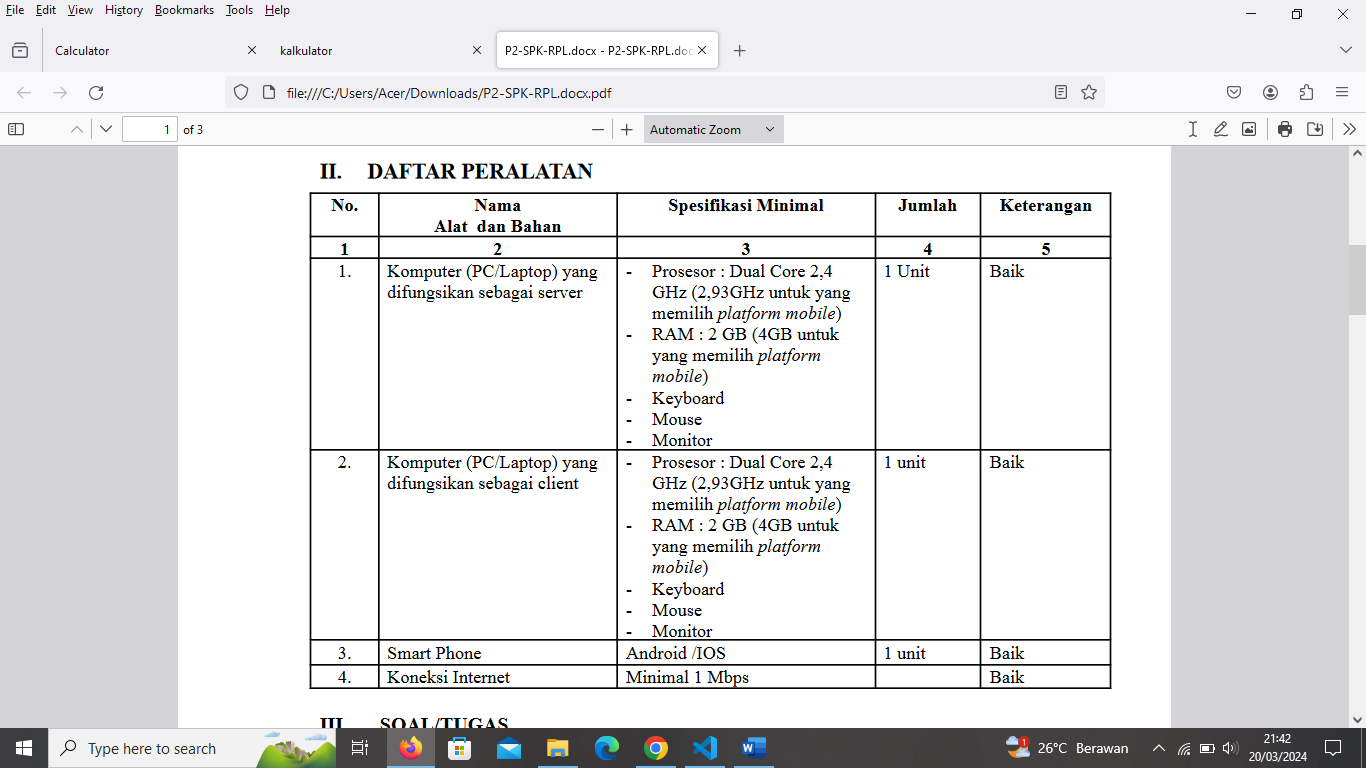
    }

    return input\_array.join("");

}

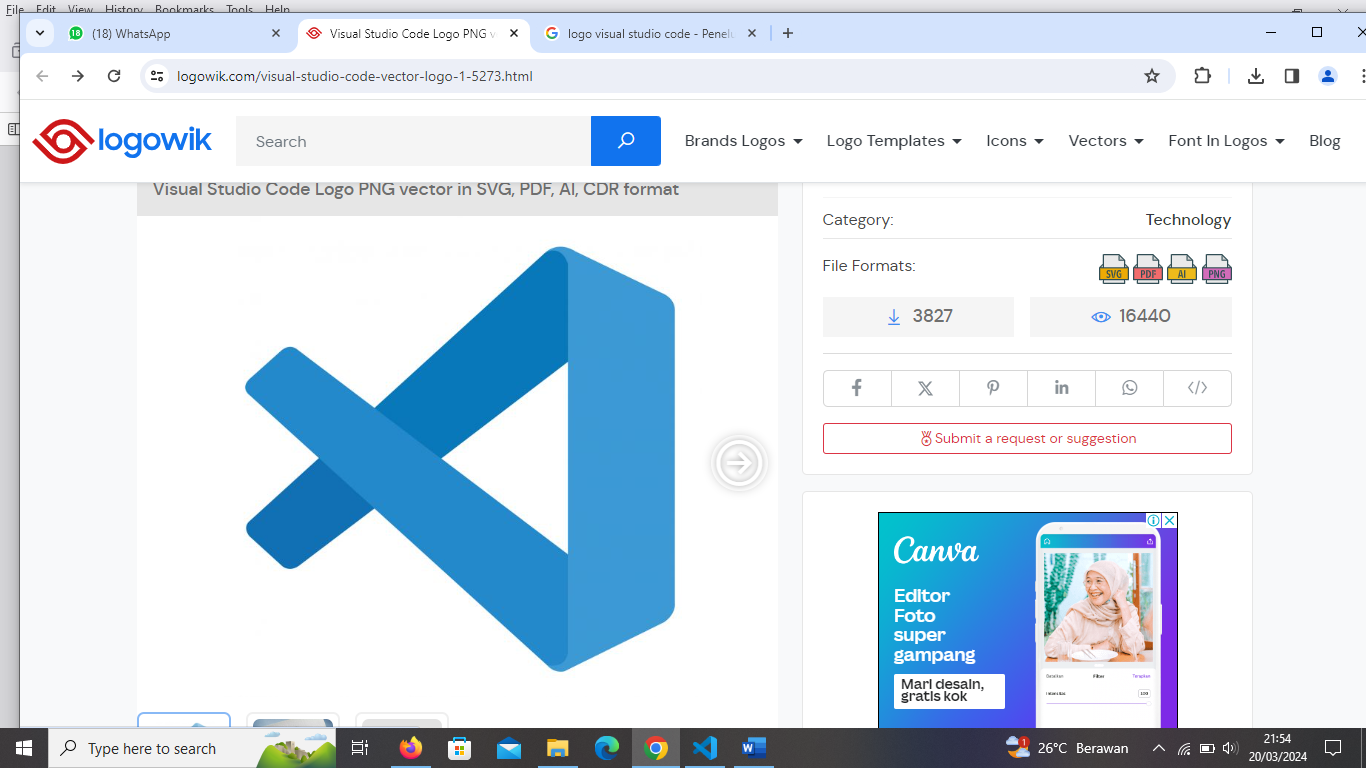
**MANUAL SINGKAT PENGGUNAAN APLIKASI**

**5.1. KebutuhanAlat dan Bahan**



**5.2. Instalasi Aplikasi Pendukung**

1. Visual Studio Code



**Visual Studio Code** (disingkat **VCode**) adalah perangkat lunak penyunting kode sumber buatan [Microsoft](https://id.wikipedia.org/wiki/Microsoft" \o "Microsoft) untuk [Linux](https://id.wikipedia.org/wiki/Linux), [macOS](https://id.wikipedia.org/wiki/MacOS), dan [Windows](https://id.wikipedia.org/wiki/Microsoft_Windows" \o "Microsoft Windows). Visual Studio Code menyediakan fitur seperti [penyorotan sintaks](https://id.wikipedia.org/wiki/Penyorotan_sintaksis" \o "Penyorotan sintaksis), penyelesaian kode, kutipan kode, merefaktor kode, [pengawakutuan](https://id.wikipedia.org/wiki/Pengawakutuan" \o "Pengawakutuan), dan [Git](https://id.wikipedia.org/wiki/Git" \o "Git). Microsoft merilis sumber kode Visual Studio Code di repositori [GitHub](https://id.wikipedia.org/wiki/GitHub" \o "GitHub) dengan [lisensi MIT](https://id.wikipedia.org/wiki/Lisensi_MIT) (**Code OSS**),  sedangkan biner yang dibangun oleh Microsoft tidak dirilis dengan lisensi MIT dan merupakan perangkat lunak berpemilik.

Visual Studio Code pertama kali diperkenalkan di tanggal 29 April 2015 oleh Microsoft di konferensi Build 2015. Versi pratinjau dirilis tidak lama setelah itu.

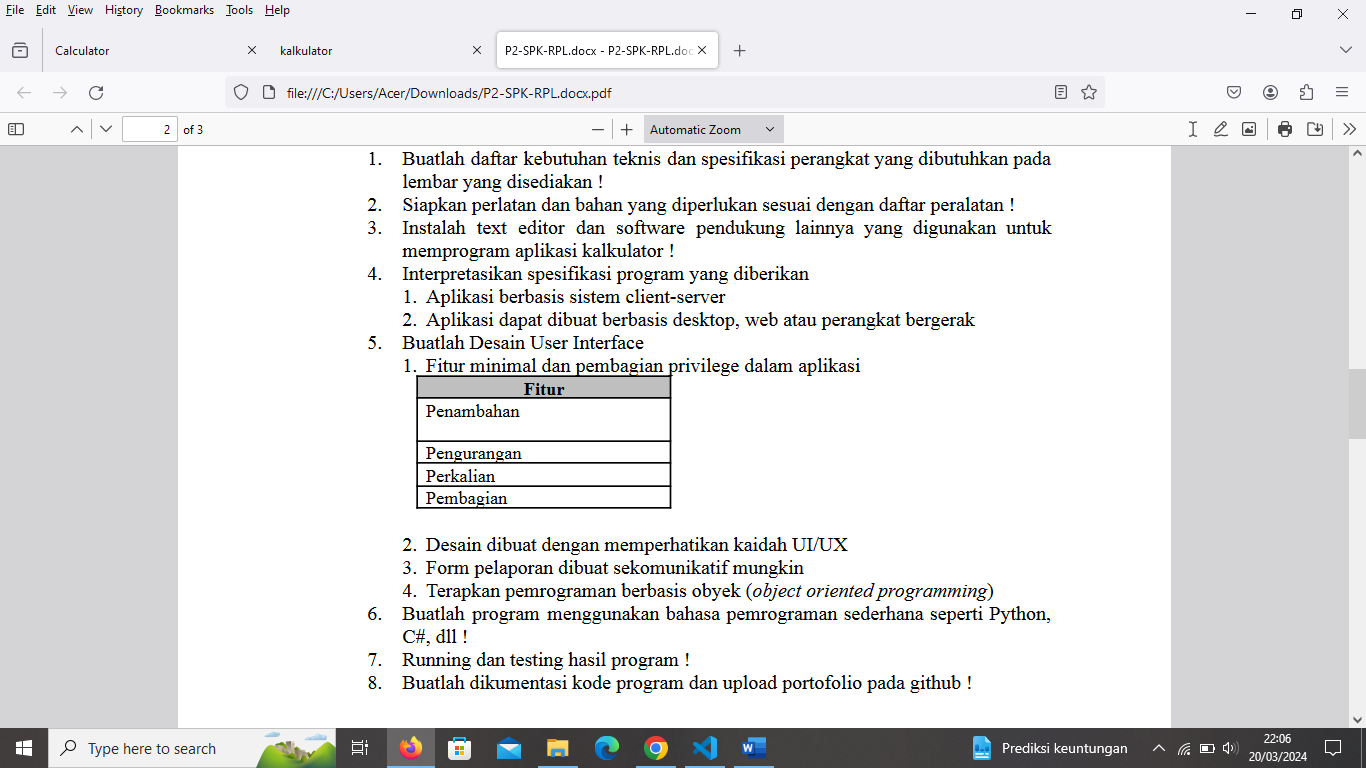
Visual Studio Code merupakan penyunting kode-sumber yang dapat digunakan untuk beragam bahasa pemrograman, termasuk C#, HTML, CSS,  [Java](https://id.wikipedia.org/wiki/Java" \o "Java),  [JavaScript](https://id.wikipedia.org/wiki/JavaScript" \o "JavaScript), [Go](https://id.wikipedia.org/wiki/Go_(bahasa_pemrograman)),  [Node.js](https://id.wikipedia.org/wiki/Node.js), [Python](https://id.wikipedia.org/wiki/Python_(bahasa_pemrograman)), [PHP](https://id.wikipedia.org/wiki/PHP) dan [C++](https://id.wikipedia.org/wiki/C%2B%2B). Visual Studio Code memperkerjakan komponen penyunting yang sama (namakode "Monaco") yang digunakan di [Azure DevOps](https://id.wikipedia.org/w/index.php?title=Azure_DevOps&action=edit&redlink=1) (awalnya dippangil Visual Studio Online dan Visual Studio Team Services).

Sebagai ganti dari sistem proyek, VSCode memungkinkan pengguna untuk membuka satu atau lebih banyak direktori, yang dapat disimpan di ruang kerja untuk digunakan ulang nanti. Ini memungkinkan untuk beroperasi sebagai penyunting kode *[languange-agnostic](https://id.wikipedia.org/w/index.php?title=Bahasa_agnostik&action=edit&redlink=1" \o "Bahasa agnostik (halaman belum tersedia))* untuk semua bahasa. VSCode mendukung banyak bahasa pemrograman dan sekumpulan fitur yang berbeda dari setiap bahasa. Berkas dan map yang tidak diingikan dapat di kecualikan dari pohon proyek melalui pengaturan proyek tersebut. Kebanyakan fitur VSCode tidak di terekspos melalui menu atau tampilan pengguna tetapi dapat diakses melalui *command palette.*

Visual Studio Code dapat diperluas melalui [ekstensi](https://id.wikipedia.org/wiki/Colok-masuk" \o "Colok-masuk), tersedia melalui repositori pusat VSCode. Ini mencakup penambahan ke penyunting dan dukungan bahasa. Salah satu fitur terkemuka dari VSCode adalah kemampuan untuk membuat ekstensi yang menambahkan dukungan untuk [bahasa](https://id.wikipedia.org/wiki/Bahasa_pemrograman" \o "Bahasa pemrograman) baru, [tema](https://id.wikipedia.org/w/index.php?title=Tema_(komputer)&action=edit&redlink=1), dan [pengawakutuan](https://id.wikipedia.org/wiki/Pengawakutuan), melalukan [analisiskodestatis](https://id.wikipedia.org/w/index.php?title=Analisis_kode_statis&action=edit&redlink=1),dan menambahkan [linter kode](https://id.wikipedia.org/w/index.php?title=Lint&action=edit&redlink=1) menggunakan [Protokol Server Bahasa](https://id.wikipedia.org/w/index.php?title=Protokol_Server_Bahasa&action=edit&redlink=1).

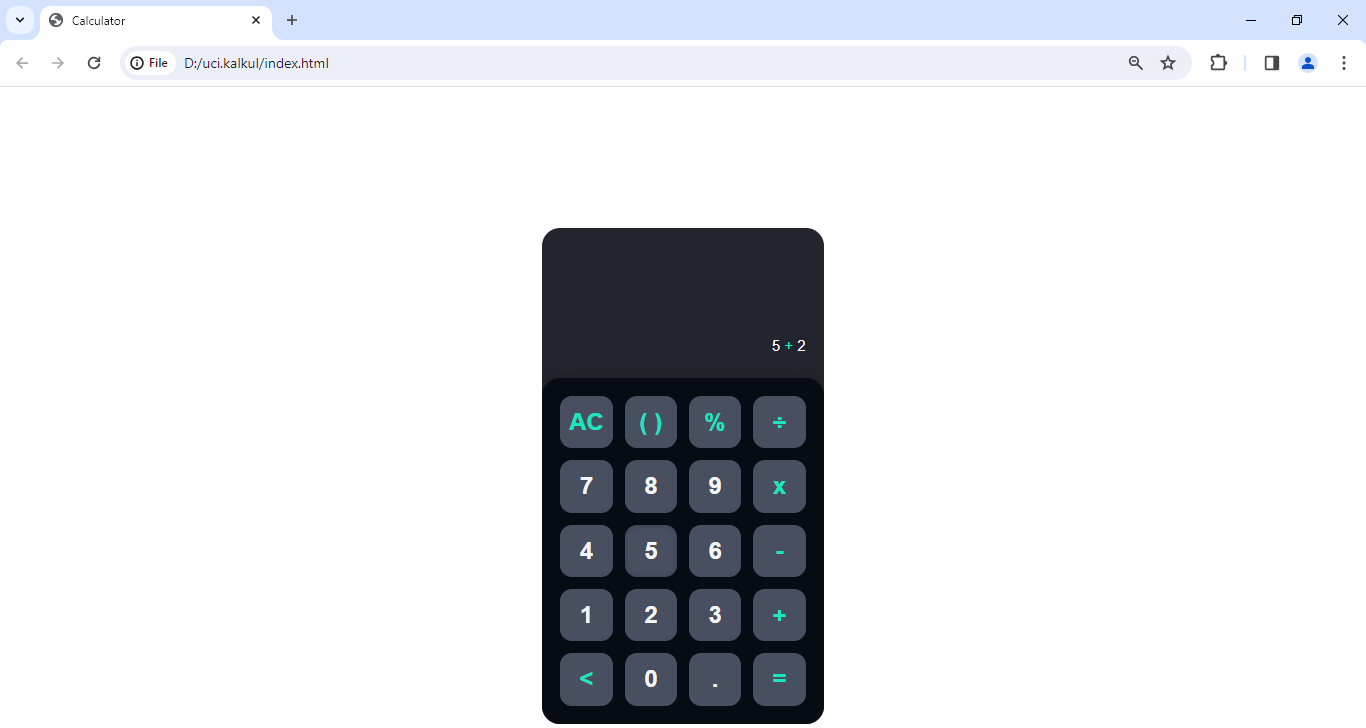
Visual Studio Code memilki beberapa ekstensi untuk [FTP](https://id.wikipedia.org/wiki/Protokol_Transfer_Berkas" \o "Protokol Transfer Berkas), memungkinkan VSCode untuk digunakan sebagai perangkat lunak gratis alternatif untuk pengembangan web. Kode dapat disinkronkan antara penyunting dan server, tanpa harus mengunduh perangkat lunak tambahan.

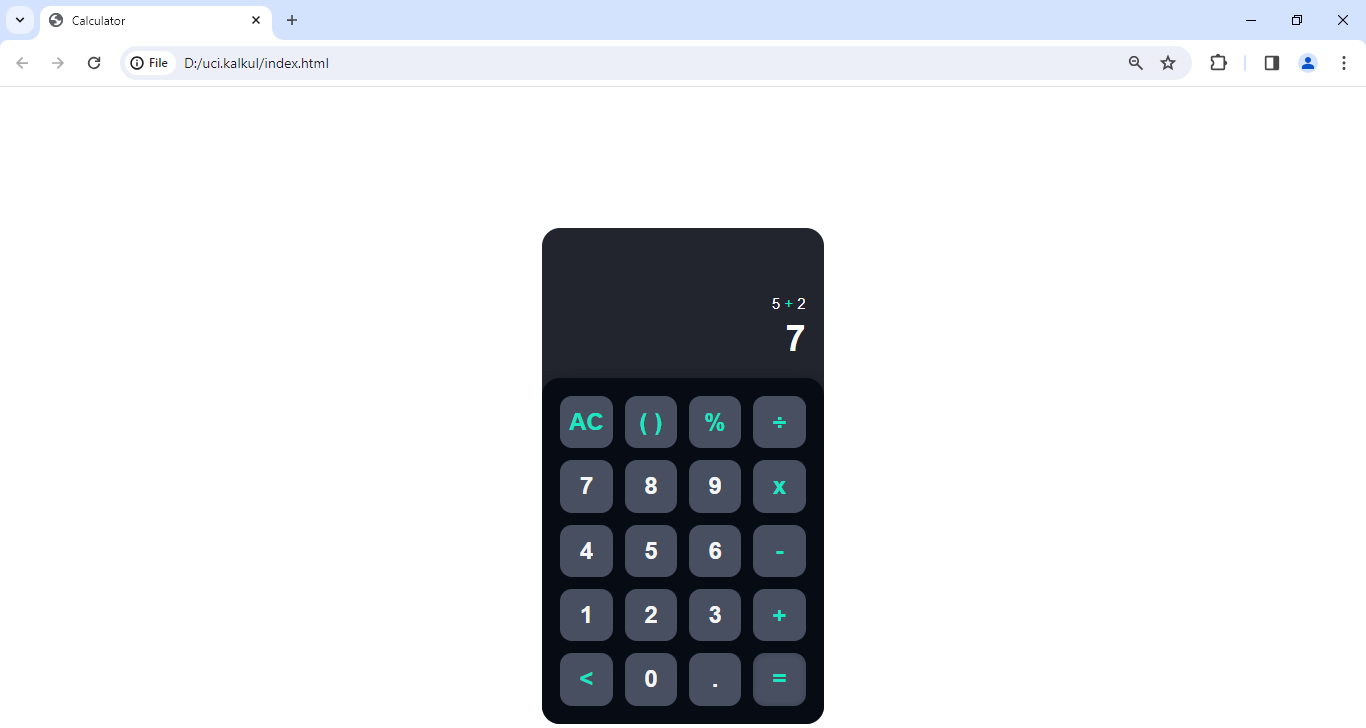
**5.3. Fitur-Fitur Minimal dalamAplikasi**



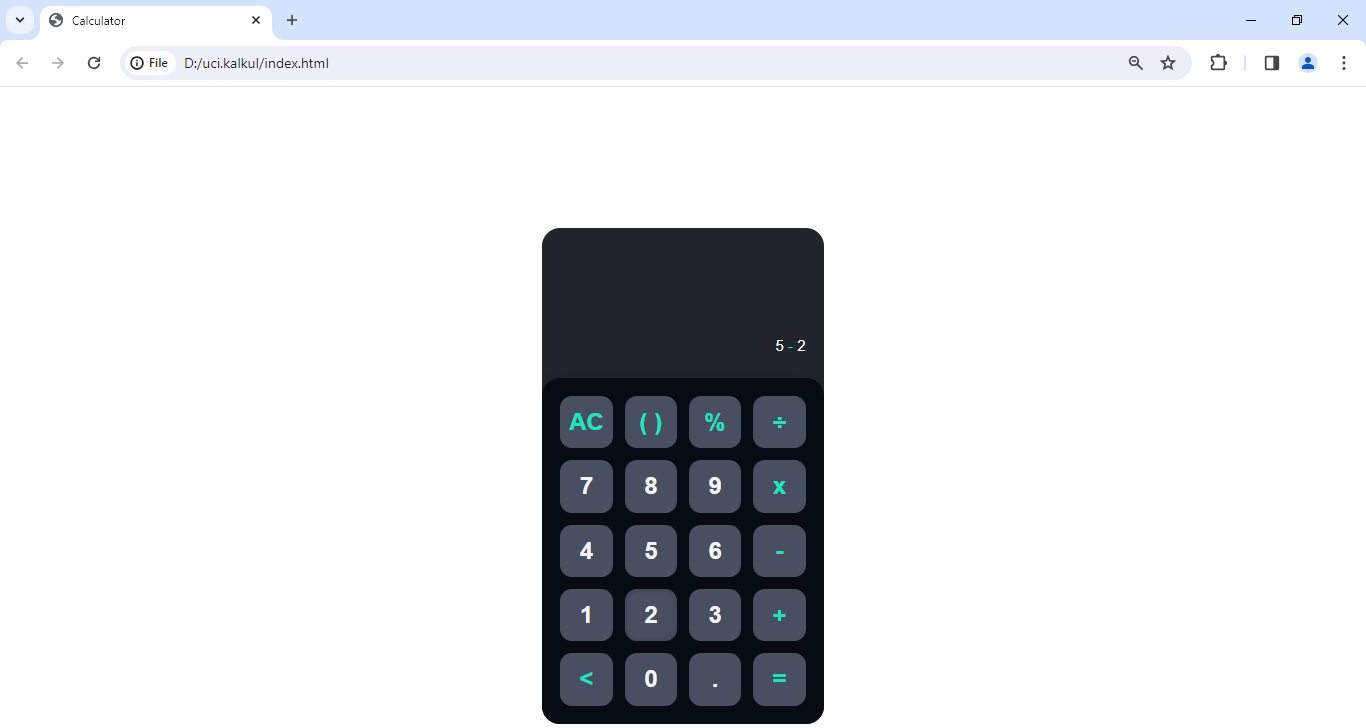
**5.4. Cara Kerja Penggunaan Aplikasi**

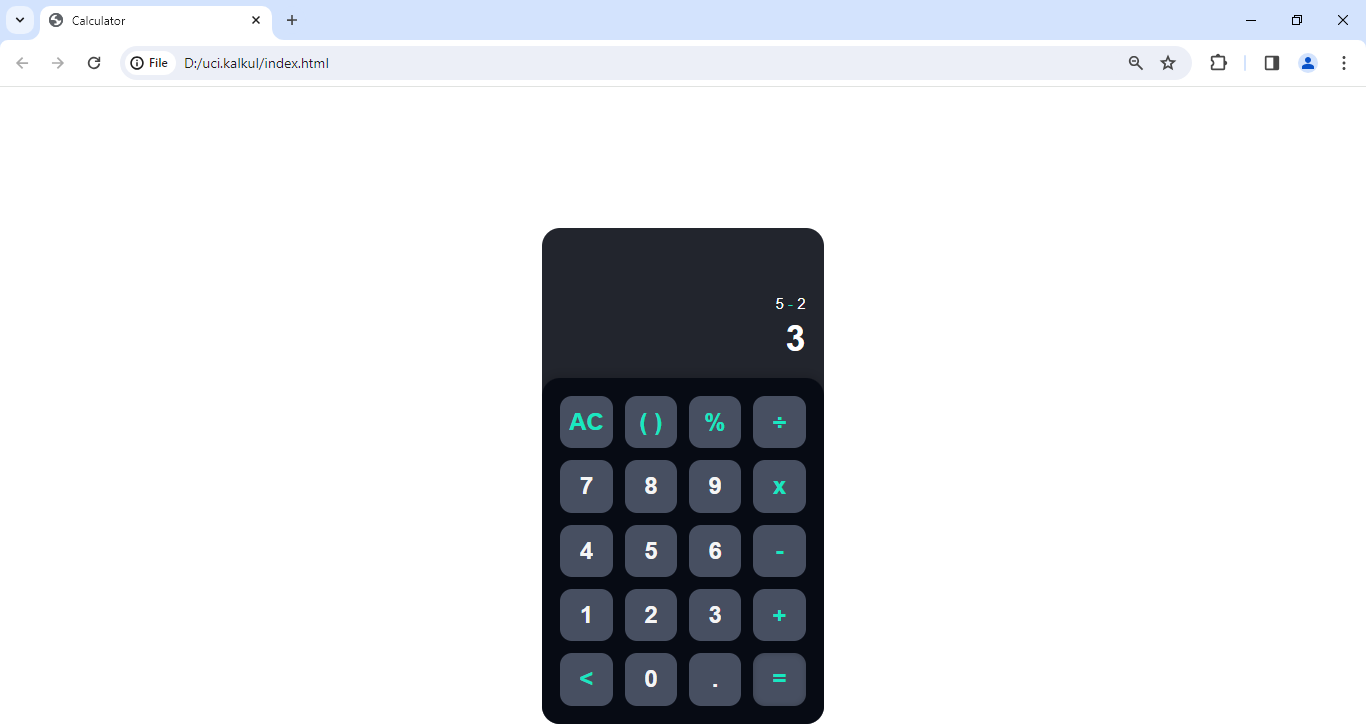
* Jika ingin melakukan penambahan bisa dilakukan dengan mengklik tanda tambah



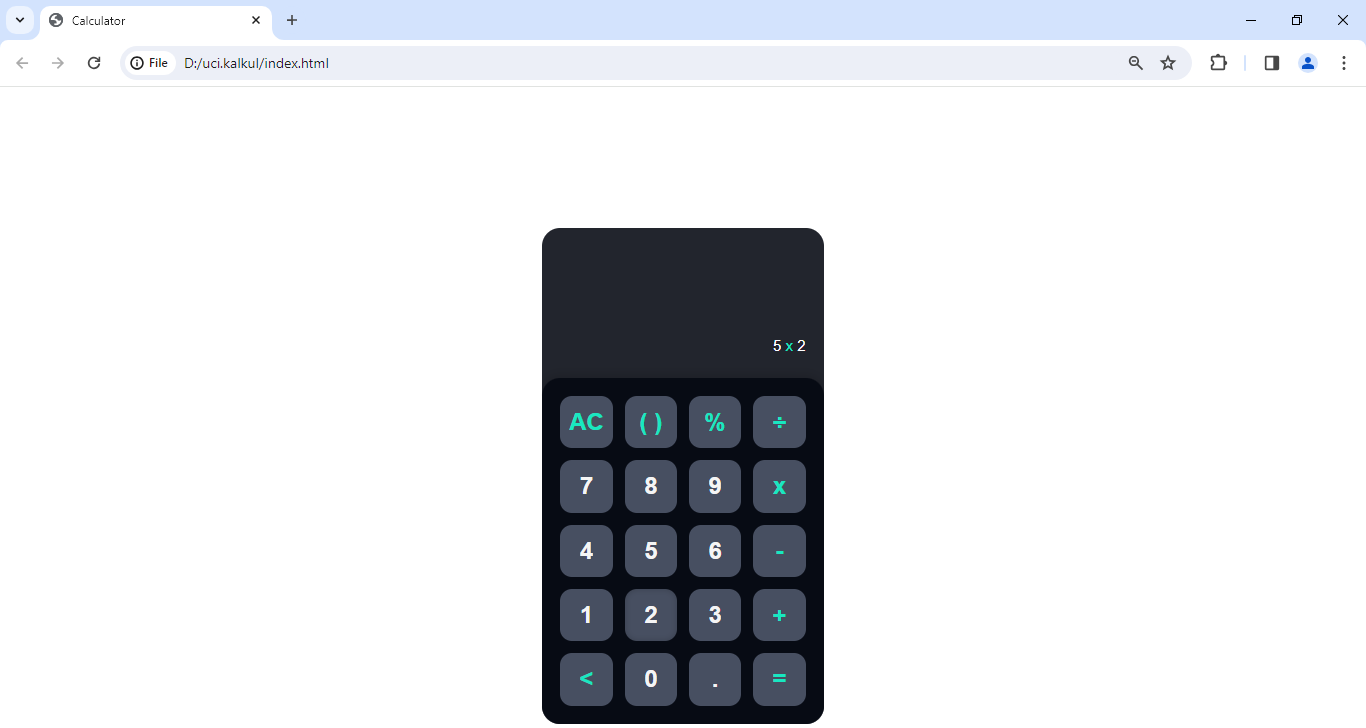


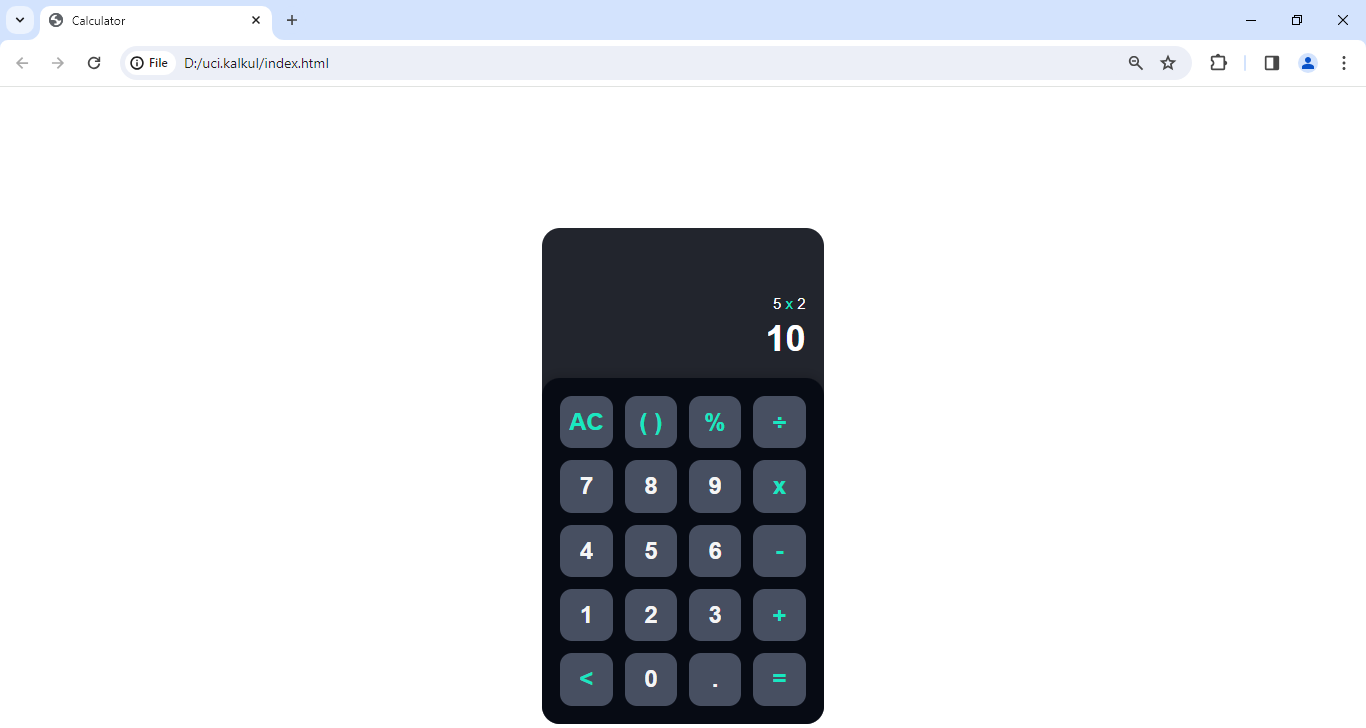
* Jika ingin melakukan pengurangan bisa dilakukan dengan mengklik tanda kurang



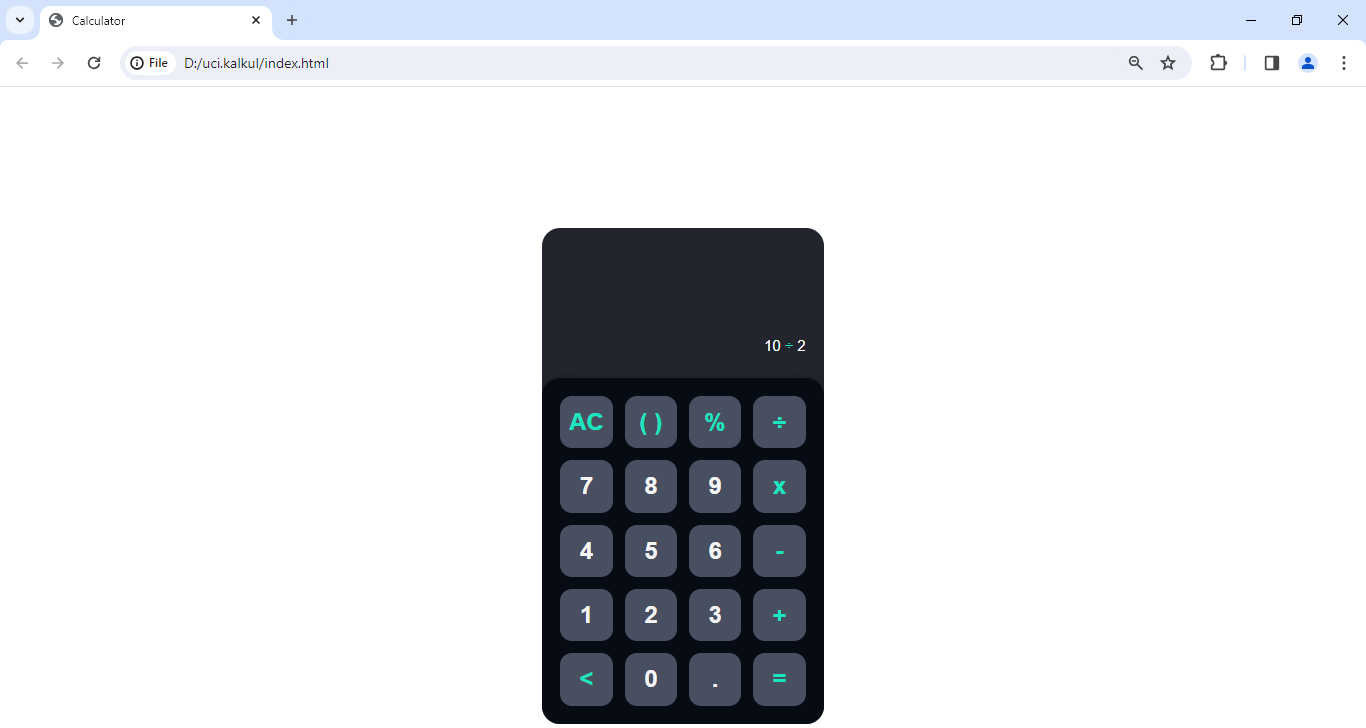


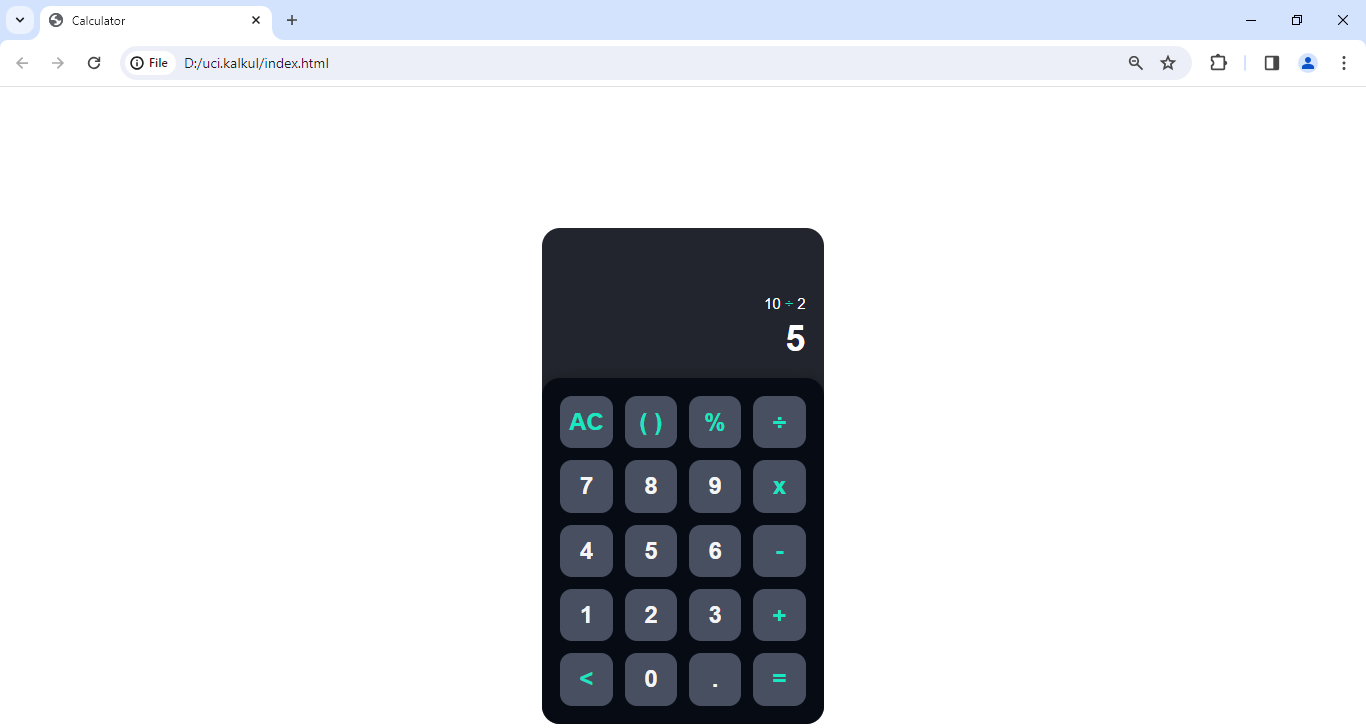
* Jika ingin melakukan perkalian bisa dilakukan dengan mengklik tanda kali





* Jika ingin melakukan pembagian bisa dilakukan dengan mengklik tanda bagi





**LINK DOKUMENTASI GITHUB**

[**https://sucifadilahh.github.io/project-UKK/**](https://sucifadilahh.github.io/project-UKK/)