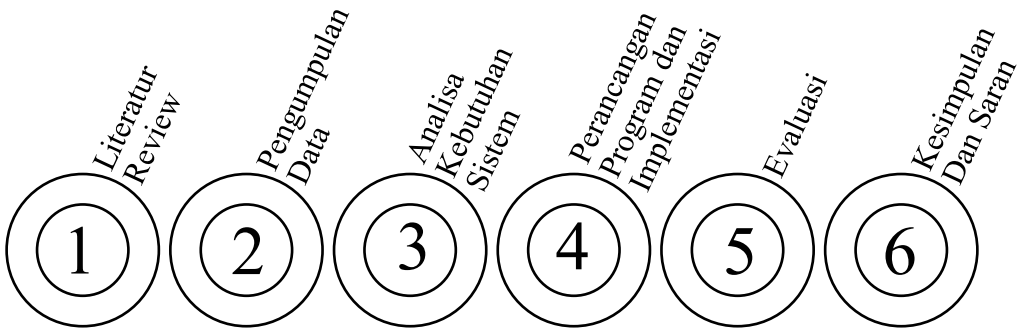
BAB III

METODOLOGI PENELITIAN

3.1 Alur Penelitian



Gambar 3. 1 Alur Penelitian

Merujuk gambar 3.1 ditunjukan alur penelitian sebagai gambaran yang akan dilakukan tahap demi tahap dalam penelitian yang dilakukan.

1. Eksplorasi Jurnal
   Eksplorasi terhadap penelitian sebelumnya dan identifikasi masalah untuk menentukan arah penelitian.
2. Pengumpulan Data

   Data yang dikumpulkan dalam penelitian ini merupakan data sekunder yang didapatkan dari sumber :

   https://gist.github.com/qti3e/6341245314bf3513abb080677cd1c93b. Data-data tersebut meliputi ekstensi *file*, offset *file* dan *hex file* dari setiap jenis *file*.

3. Tahap Analisa Kebutuhan Sistem
   Mendeteksi sebuah arsitektur aplikasi *File Signature Analyzer* 2.0 terkait kebutuhan *software* dan *hardware* yang digunakan. Tabel 3.1 menggambarkan spesifikasi *software* dan *hardware* yang digunakan *File Signature Analyzer 2.0*.

Tabel 3.1 Analisis Kebutuhan *Software* dan *Hardware*

No	Analisa Kebutuhan Software	Analisa Kebutuhan Hardware
1	Sistem Operasi: Windows 10 Home 64-Bit	Device: Asus Aspire 5
2	Database: PostgreSQL 15.3	Processor: Intel Core I3-7020u
3	Bahasa Pemrograman: Javascript	RAM: 4 Gb
4	Backend Framework: Express.Js	HDD: 1000 Gb

4. Tahap Implementasi
   Berdasarkan eksplorasi jurnal yang telah ditempuh, implementasi dan pengujian akan berfokus meningkatkan fitur aplikasi.
5. Tahap Evaluasi
   Pengumpulan informasi mengenai kinerja yang bertujuan memperbaiki kekurangan yang telah dilakukan pengujian sebelumnya.
6. Kesimpulan dan Saran

Kesimpulan dari penelitian yang telah dibuat dan saran bagi penelitian selanjutnya.

1.2 *Pseudocode* Aplikasi

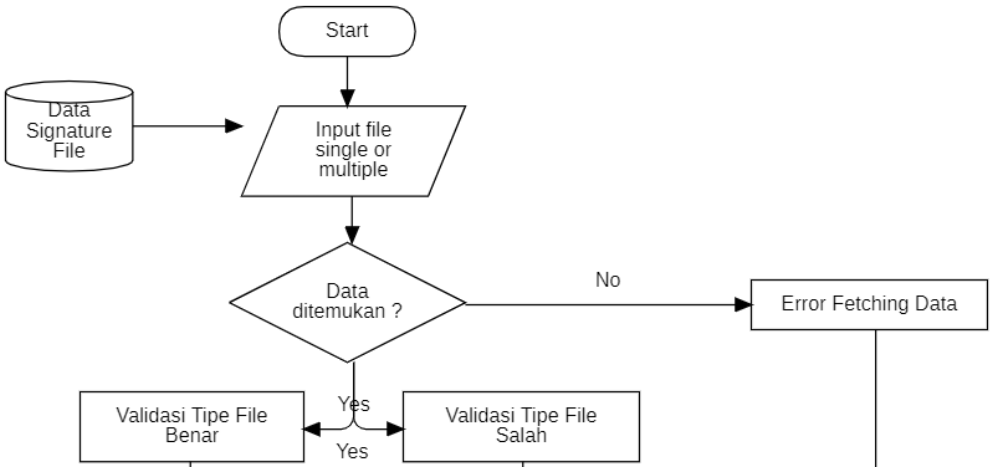
```
function mulai() {

    let daftarFile = ambilDaftarFile();
    let hasilAnalisis = [];

    for (let file of daftarFile) {
        let ekstensiFile = bacaEkstensiFile(file);
        let sigantureFile = bacaSignatureFile(file);

        if (cocokkanSignatureFile(sigantureFile)) {
            hasilAnalisis.push({ file: file, status: 'orisinal' });
        } else {
            hasilAnalisis.push({ file: file, status: 'modifikasi' });
        }
    }

    simpanHasilAnalisis(hasilAnalisis);

    tampilkanLaporan(hasilAnalisis);

    selesai();
}

function ambilDaftarFile() {
    return ['file1.txt', 'file2.pdf', 'file3.jpg'];
}

function bacaEkstensiFile(file) {
    return file.split('.').pop();
}

function bacaSignatureFile(file) {
    return 'signature_sample';
}

function cocokkanSignature(signature) {
    let basisDataSignature = ['signature_sample'];
    return basisDataSignature.includes(signature);
}

function simpanHasilAnalisis(hasilAnalisis) {
    let hasil = '';
    hasilAnalisis.forEach(hasilItem => {
```

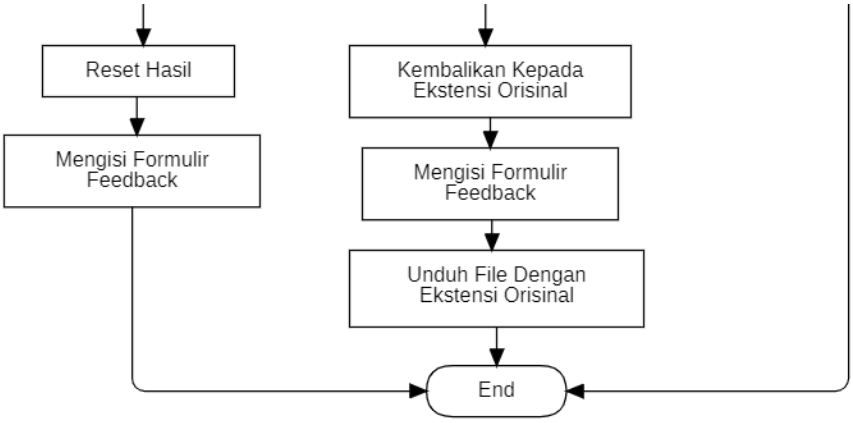
```
44          hasil += `File: ${hasilItem.file}, Status: ${hasilItem.status}\n`;
45      });
46      console.log(hasil);
47  }
48
49  function tampilkanLaporan(hasilAnalisis) {
50      hasilAnalisis.forEach(hasilItem => {
51          console.log(`File: ${hasilItem.file}, Status: ${hasilItem.status}`);
52      });
53  }
54
55  function selesai() {
56      console.log("Analisis selesai.");
57  }
```

Gambar 3.2 Algoritma Aplikasi

Merujuk gambar 3.2 ditunjukan *pseudocode* yang diterapkan kepada aplikasi. Algoritma ini menggunakan metode pengulangan sebagai inti dari proses yang berlangsung ketika aplikasi melakukan *check document*. Cara kerja algoritma pengulangan ini sistem memproses *file input* dengan cara memeriksa keseluruhan *magic number* dan berhenti ketika mendapatkan *magic number* yang sesuai antara *file input* dengan *magic number* yang tersimpan di dalam *database* aplikasi.

1.2.1 Flowchart Aplikasi





Gambar 3. 3 *Flowchart* Aplikasi

Merujuk gambar 3.3 ditunjukan *flowchart* aplikasi yang bisa diakses oleh pengguna yang sudah memiliki akun. Aplikasi *File Signature Analyzer* 2.0 hanya memiliki aktor utama yaitu user.

3.2.2 Cara Kerja Algoritma

1. *Record* data dalam *database* memiliki 4 atribut:
   - *Offset*
   - *Hex*
   - Ekstensi
   - *Mimetype*
2. Contoh mendapatkan *file input* = .jpeg.
3. Mengambil semua *record* dengan data ekstensi = .jpeg.
4. Melakukan *looping* terhadap data tersebut dengan tujuan untuk membandingkan dengan data *signature* yang tersimpan di *database*, setiap melakukan *looping* ambil data paling penting yaitu *offset* dan *hex*.

5. *Hex* dibaca sepanjang 2 *byte* untuk memberikan *output*. Jika iya *hex* = FFD8 *return true*, jika tidak *return false*.
6. Data yang ditampilkan jika *signature* sama adalah data yang pertama kali melalui proses *looping* (sesuai urutan *database*).