BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Kegiatan magang ini dilaksanakan di bawah naungan divisi *Community Engagement Apps Development*, yang merupakan bagian dari *Research, Innovation, and Sustainability* (RIS) di Universitas Multimedia Nusantara (UMN). Divisi ini berfokus pada pengembangan sistem dan aplikasi berbasis web yang mendukung kegiatan pengabdian masyarakat universitas.

Dalam pelaksanaan magang ini, penulis menempati posisi sebagai *Back-End Developer* dengan sistem kerja *Work From Home* (WFH). Penulis melanjutkan pengembangan proyek Sistem Informasi Koperasi Simpan Pinjam Wiyata Mandala, yang sebelumnya telah dikembangkan oleh tim magang periode sebelumnya. Proyek ini berfungsi sebagai sistem digital koperasi yang mencakup pengelolaan simpanan, pinjaman, serta pembagian Sisa Hasil Usaha (SHU) anggota koperasi.

Koordinasi kegiatan dilakukan bersama rekan satu tim dan pembimbing lapangan dari pihak RIS UMN. Komunikasi dan pembagian tugas dilaksanakan secara daring melalui *Zoom Meeting* untuk rapat, *WhatsApp Group* untuk komunikasi harian, serta *GitHub* untuk kolaborasi dalam pengembangan kode sumber proyek. Pembimbing lapangan berperan dalam memberikan arahan teknis dan memantau perkembangan kegiatan magang agar selaras dengan tujuan pengembangan sistem yang sedang berjalan di divisi.

3.2 Tugas yang Dilakukan

Pada periode magang ini, tugas penulis difokuskan sepenuhnya pada penyelesaian pekerjaan utama yang berpusat pada perbaikan modul Sisa Hasil Usaha (SHU) di sisi backend. Pekerjaan ini dikelompokkan menjadi dua tugas inti yang saling terkait: perbaikan formula perhitungan dan penerapan automasi data masukan.

3.2.1 Perbaikan Perhitungan SHU Berdasarkan Indeks

Logika SHU yang diimplementasikan oleh tim sebelumnya masih menggunakan metode kalkulasi sederhana (Total Simpanan - Total Pinjaman), yang

tidak memenuhi ketentuan koperasi.

1. Tujuan: Mengganti logika perhitungan dalam fungsi `getAnggotaSHUData` dengan rumus perhitungan yang resmi, berdasarkan buku indeks perhitungan yang disediakan oleh pihak koperasi.

2. Cakupan: Implementasi fungsi `calculateShuIndexes` untuk menghitung indeks berdasarkan agregasi total masukan anggota (Simpanan Pokok + Wajib) terhadap total dana SHU yang dialokasikan.

3.2.2 Automasi Akumulasi Nilai SHU Real-Time

Kebutuhan untuk mencatat kontribusi SHU anggota secara real-time. Sebelumnya, kontribusi SHU hanya dapat dihitung melalui proses manual periodik.

1. Tujuan: Menerapkan sistem automasi yang menjamin bahwa nilai transaksi yang berkontribusi pada SHU terakumulasi secara langsung ke tabel MS_SHU_ANGGOTA setiap kali transaksi berhasil diproses.

2. Cakupan: Pengembangan fungsi `autoAccumulateShu` yang dipicu oleh `addManualTransaction` untuk melakukan *atomic increment* pada kolom akumulator di database.

3.3 Uraian Pelaksanaan Magang

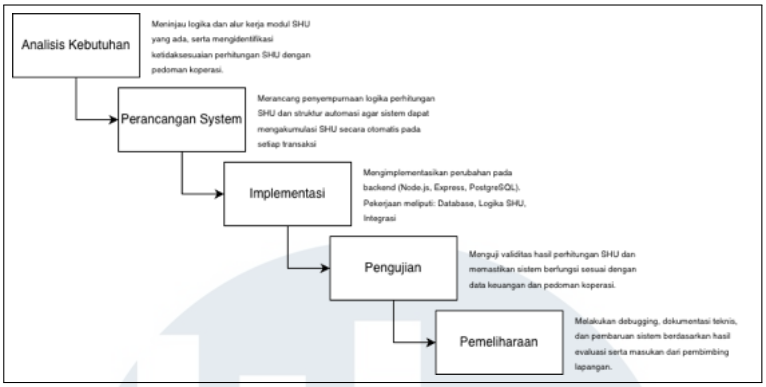
Pelaksanaan kegiatan magang dilakukan dengan menggunakan teknologi dan alat bantu pengembangan yang telah disiapkan dan digunakan oleh tim sebelumnya, di antaranya:

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke-	Pekerjaan yang Dilakukan
1	Bertemu dengan pengawas proyek dan tim pendahulu, perkenalan proyek.
2–4	Mempelajari struktur proyek dan memahami alur sistem koperasi, khususnya modul SHU yang dikembangkan oleh tim magang sebelumnya.
5–6	Meninjau dan menganalisis logika perhitungan SHU berdasarkan pedoman koperasi serta mempersiapkan rancangan penyempurnaan perhitungan.
7–8	Mengembangkan sistem automasi perhitungan SHU agar nilai SHU anggota terakumulasi secara otomatis setelah transaksi dilakukan.
9–10	Melakukan pengujian dan debugging modul SHU untuk memastikan fungsionalitas berjalan dengan baik dan hasil perhitungan sesuai data koperasi.
11–15	Menyusun dokumentasi pengembangan dan laporan akhir kegiatan magang, termasuk evaluasi hasil serta rekomendasi pengembangan lanjutan.

Metode pengembangan sistem yang digunakan dalam kegiatan magang ini adalah model Waterfall [1]. Model ini juga telah digunakan oleh tim magang periode sebelumnya dalam pengembangan Sistem Informasi Koperasi Simpan Pinjam.

Model Waterfall memiliki lima tahapan utama yaitu:



Gambar 3.1. *Waterfall Diagram*

1. Analisis Kebutuhan (*Requirement Analysis*): Meninjau logika dan alur kerja modul SHU yang dikembangkan oleh tim sebelumnya serta mengidentifikasi ketidaksesuaian pada perhitungan SHU dengan pedoman koperasi.

2. Perancangan Sistem (*System Design*): Merancang penyempurnaan logika perhitungan SHU dan struktur automasi agar sistem dapat mengakumulasi SHU secara otomatis setiap transaksi terjadi.

3. Implementasi (*Implementation*): Mengimplementasikan perubahan pada backend sistem menggunakan Node.js, Express, dan PostgreSQL [2]. Pekerjaan ini meliputi:

   (a) Database: Menambahkan UUID_MS_USER pada `TR_MANUAL_TRANSACTIONS` dan kolom akumulator baru pada `MS_SHU_ANGGOTA` (melalui ALTER TABLE) [3].

   (b) Logika SHU: Implementasi fungsi calculateShuIndexes dan autoAccumulateShu dalam `ShuController.js`.

   (c) *Trigger Point*: Integrasi panggilan autoAccumulateShu ke dalam fungsi addManualTransaction di `KeuanganController.js` untuk mencapai automasi real-time [4].

4. Pengujian (*Testing*): Menguji validitas hasil perhitungan SHU dan memastikan sistem berjalan sesuai dengan data keuangan koperasi [5].

5. Pemeliharaan (*Maintenance*): Melakukan debugging, dokumentasi teknis, dan pembaruan sistem berdasarkan hasil evaluasi dan masukan dari pembimbing lapangan.

3.3.1 Teknologi dan Lingkungan Kerja

Pengembangan sistem ini dilaksanakan dengan fokus pada sisi backend dan basis data, menggunakan alat-alat yang sudah ada pada proyek.

1. Lingkungan Server: Node.js dan Express.js sebagai framework minimalis untuk mengelola routing dan API REST[6][7].
2. Basis Data: PostgreSQL menggunakan aplikasi Postgres sebagai sistem manajemen basis data relasional [8].
3. ORM (Object-Relational Mapping): Sequelize.js digunakan untuk memetakan model JavaScript ke skema basis data, memfasilitasi operasi data yang kompleks, termasuk fungsi khusus seperti `Sequelize.literal` yang vital untuk automasi akumulasi[9].
4. *Debugging* dan Pengujian: Menggunakan *Visual Studio Code REST Client* (dengan file `.rest`) untuk mengirim permintaan HTTP ke endpoint API (localhost:5000) dan *pgAdmin4* untuk memverifikasi integritas data secara langsung di tabel [10].

3.3.2 Protokol Pengujian Fungsional

Untuk memverifikasi keberhasilan implementasi, protokol pengujian fungsional yang dilakukan meliputi:

1. Konfirmasi Data Master: Memastikan data penting (MS_TYPE_SIMPANAN untuk Simpanan Pokok dan Wajib, dan MS_JOB untuk role) telah dimuat dengan benar (dummy.rest).
2. Verifikasi Autentikasi: Mengatasi kendala 401 Unauthorized dengan melakukan login melalui API dan menyertakan Access Token di header Authorization: Bearer untuk setiap pengujian transaksi.
3. Pengujian Akumulasi: Mengirim transaksi POST ke `/addManualTransaction` dan memverifikasi di pgAdmin4 bahwa kolom `total_transaksi_jasa` di `MS_SHU_ANGGOTA` bertambah sesuai nominal transaksi.

4. Pengujian Kalkulasi: Mengirim permintaan GET ke `/shu/anggota` untuk mengkonfirmasi bahwa sistem menjalankan logika formula indeks (nilai SHU > 0 atau SHU $= 0$ jika indeks denominator 0).

3.4 Modul Sisa Hasil Usaha (SHU)

Pengembangan modul SHU berfokus pada perbaikan mendasar terhadap logika perhitungan dan implementasi automasi data masukan.

3.4.1 Modifikasi Struktur Model Kritis

Untuk menopang fungsionalitas SHU yang baru, dilakukan perubahan struktur pada tiga model utama:

1. Model `MS_USER`: Berfungsi sebagai *Primary Key* yang menautkan data anggota ke seluruh tabel transaksi dan akumulator melalui `UUID_MS_USER`.

2. Model `TR_MANUAL_TRANSACTIONS`: Dimodifikasi dengan penambahan *Foreign Key* `UUID_MS_USER` untuk memungkinkan pelacakan kontribusi ekonomi setiap anggota secara spesifik.

3. Model `MS_SHU_ANGGOTA`: Mengalami transformasi signifikan dengan penambahan kolom akumulator seperti `total_transaksi_jasa` dan `total_transaksi_alfa`. Kolom ini menggunakan tipe data *double precision* untuk menyimpan nilai kontribusi yang terakumulasi secara otomatis melalui *atomic increment*.

3.4.2 Definisi Variabel dan Fungsi Utama

Sebelum masuk ke tahap implementasi kode program, diperlukan pemahaman mengenai variabel dan fungsi kunci yang digunakan dalam pengembangan modul SHU. Hal ini bertujuan untuk memperjelas alur logika pada sistem yang dibangun menggunakan Node.js dan Sequelize. Definisi tersebut dirangkum dalam Tabel 3.2 berikut:

Tabel 3.2. Definisi Variabel dan Fungsi Modul SHU

Nama Variabel / Fungsi	Deskripsi	Sumber File
`autoAccumulateShu`	Fungsi mesin inti yang menjamin akumulasi nilai transaksi ke dalam tabel SHU secara otomatis dan *real-time*.	`ShuController.js`
`fieldToUpdate`	Variabel penentu kolom akumulator target (seperti *total_transaksi_jasa*) berdasarkan tipe transaksi.	`ShuController.js`
`Sequelize.literal`	Perintah untuk menjalankan *Atomic Increment* (*field* = *field* + *nominal*) guna menjaga integritas data.	`ShuController.js`
`calculateShuIndexes`	Fungsi untuk menghitung rasio pembagian SHU global berdasarkan total masukan anggota (*denominator*).	`ShuController.js`
`totalSimpanan Mandatory`	Variabel agregat yang menjumlahkan seluruh Simpanan Pokok (SP) dan Wajib (SW) anggota sebagai pembagi indeks.	`ShuController.js`
`addManualTransaction`	Fungsi pemicu (*trigger*) utama yang memanggil proses akumulasi SHU segera setelah transaksi disimpan.	`Keuangan Controller.js`
`UUID_MS_USER`	*Foreign Key* baru pada tabel transaksi untuk menghubungkan nominal dengan identitas spesifik anggota.	`TR_MANUAL_ TRANSACTIONS.js`
`total_transaksi_jasa`	Kolom akumulator baru pada tabel SHU yang menyimpan total kontribusi transaksi jasa sepanjang tahun.	`MS_SHU_ANGGOTA.js`

3.4.3 Implementasi Automasi Akumulasi

Automasi ini diimplementasikan dengan membuat fungsi pemicu yang dipanggil oleh kontroler transaksi.

A Pemicu di `KeuanganController.js`

Fungsi `addManualTransaction` diubah untuk memanggil mesin akumulasi (`autoAccumulateShu`) setelah transaksi berhasil disimpan, menjadikannya titik pemicu (*trigger point*) utama:

```
// KeuanganController.js: Pemicu Automasi
const newTransaction = await TrManualTransactions.create({
    // ... data transaksi
    UUID_MS_USER: req.body.UUID_MS_USER, // Data masukan
    // ...
});

// Panggil fungsi akumulasi SHU real-time
await autoAccumulateShu(
    newTransaction.UUID_MS_USER,
    newTransaction.YEAR,
    req.body.TRANSACTION_SHU_TYPE,
    newTransaction.AMOUNT
);
```

Kode 3.1: Pemicu Automasi pada KeuanganController.js

B Mesin Akumulasi di `ShuController.js`

Fungsi `autoAccumulateShu` adalah mesin inti yang menjamin integritas data saat penambahan nilai.

```
// ShuController.js: Mesin Akumulasi Real-Time
export const autoAccumulateShu = async (userId, year, type,
   nominal) => {
    let fieldToUpdate = mapTransactionType(type); // Menentukan
   kolom SHU

    if (fieldToUpdate) {
        // FindOrCreate: Mencegah duplikasi record SHU per tahun
        const [shuRecord] = await SHU.findOrCreate({
            where: { user_id: userId, tahun: year },
            // ... defaults: set semua kolom akumulator ke 0 ...
        });

        // Atomic Increment: Menjamin penambahan aman di database
        const updateObject = {};
```

```
14          updateObject[fieldToUpdate] = Sequelize.literal(`${
    fieldToUpdate} + ${nominal}`);
15
16          await shuRecord.update(updateObject);
17      }
18  };
```

Kode 3.2: Mesin Akumulasi Real-Time pada ShuController.js

3.4.4 Implementasi Perhitungan Indeks

Logika inti ini sepenuhnya menggantikan formula lama (Simpanan − Pinjaman). Rumus yang digunakan diubah berdasarkan buku indeks koperasi yang disediakan kepada penulis, yang dapat dilihat sebagai berikut.

$$Indeks_{Simpanan} = \frac{\text{Total Alokasi SHU Simpanan}}{\sum(\text{Simpanan Pokok} + \text{Simpanan Wajib})} \times 100\% \tag{3.1}$$

$$Indeks_{Transaksi} = \frac{\text{Total Alokasi SHU Jasa Transaksi}}{\sum \text{Total Transaksi Anggota}} \times 100\% \tag{3.2}$$

$$Indeks_{Alfamart} = \frac{\text{Total Alokasi Laba Alfamart}}{\sum \text{Total Transaksi Anggota}} \times 100\% \tag{3.3}$$

A Logika Indeks Global di `calculateShuIndexes`

Fungsi ini (di `ShuController.js`) menghitung rasio SHU global dengan menjumlahkan seluruh masukan anggota (*denominator*) dari tabel transaksi.

```
1  // ShuController.js: Menghitung Rasio Global (Denominator)
2  const totalSimpananMandatory = await TrPengajuanSimpanan.sum('
       NOMINAL', {
3      where: {
4          UUID_TYPE_SIMPANAN: { [Op.in]: [ID_SP, ID_SW] },
5          STATUS_CODE: 'APPROVED'
6      }
```

```
});

const totalTransaksiJasa = await TrManualTransactions.sum('AMOUNT', {
    where: { TRANSACTION_SHU_TYPE: 'JasaTransaksi', IS_DELETED: false }
});

// Indeks dihitung: Index = (Dana Dialokasikan / Total Masukan)
const indexSimpanan = totalSimpananAllocated / totalSimpananMandatory;
const indexTransaksi = totalTransaksiAllocated / totalTransaksiJasa;
// ... returns { indexSimpanan, indexTransaksi, ... }
```

Kode 3.3: Penghitungan Rasio Global SHU

B Penerapan Formula Indeks di `getAnggotaSHUData`

Fungsi ini menerapkan formula per anggota menggunakan input yang telah diakumulasi.

```
// ShuController.js: Penerapan Formula Indeks Per Anggota
// 1. Ambil indeks global
const { indexSimpanan, indexTransaksi } = await calculateShuIndexes(year);

// 2. Ambil input akumulator anggota
const shuInputs = await SHU.findOne({
    where: { user_id: user.UUID_MS_USER, tahun: year }
});

const totalSimpananMandatory = Number(shuInputs?.total_simpanan_mandatory || 0);
const totalTransaksiJasa = Number(shuInputs?.total_transaksi_jasa || 0);

// 3. Hitung SHU Komponen (Input * Index):
const shuSimpanan = totalSimpananMandatory * indexSimpanan;
const shuTransaksi = totalTransaksiJasa * indexTransaksi;

const totalSHUAnggota = shuSimpanan + shuTransaksi + shuAlfa + jasaSR;
```

```
18 // ... return data
```

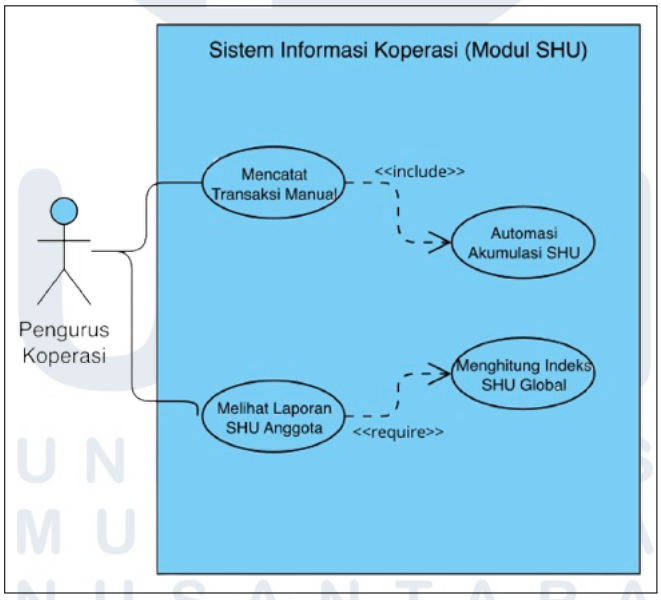
Kode 3.4: Penerapan Formula Indeks Per Anggota

3.5 Pemodelan UML

3.5.1 Use Case Diagram: Interaksi Fungsional SHU

Use Case Diagram (Gambar 3.3) memvisualisasikan hubungan antara *Pengurus Koperasi* sebagai aktor utama dengan fungsionalitas di dalam modul SHU. Diagram ini menekankan pada:

1. Pencatatan Transaksi: Pengurus melakukan input transaksi manual yang secara sistematis memicu (*include*) fungsi automasi akumulasi data anggota.
2. Pelaporan SHU: Sistem melakukan perhitungan indeks global berdasarkan data agregat sebelum menyajikan laporan SHU akhir kepada pengurus.



Gambar 3.2. *Use Case Diagram* Modul SHU *Real-Time*

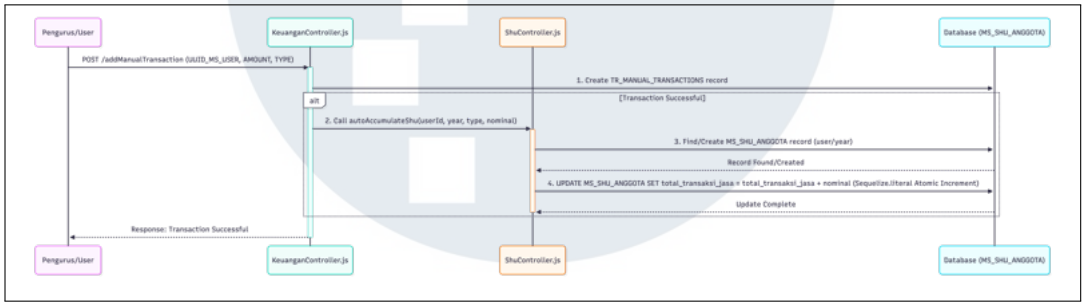
3.5.2 *Sequence Diagram*: Automasi Akumulasi SHU

Sequence Diagram memvisualisasikan alur kerja automasi real-time. Diagram ini secara eksplisit menunjukkan urutan panggilan fungsi yang terjadi di sisi server, mulai dari pemicu transaksi hingga pembaruan data di database.

1. Trigger: User (Pengurus) mengirimkan POST request ke `KeuanganController.js` untuk mencatat transaksi manual yang mengandung kontribusi SHU.

2. Call Accumulation Engine: Setelah transaksi dicatat di `TR_MANUAL_TRANSACTIONS`, KeuanganController memanggil fungsi `autoAccumulateShu` yang berada di dalam ShuController.js.

3. Atomic Update: ShuController berinteraksi dengan model MS_SHU_ANGGOTA di database menggunakan perintah yang mengandung `Sequelize.literal` untuk menjamin penambahan nominal secara aman dan terhindar dari race condition pada kolom akumulator.



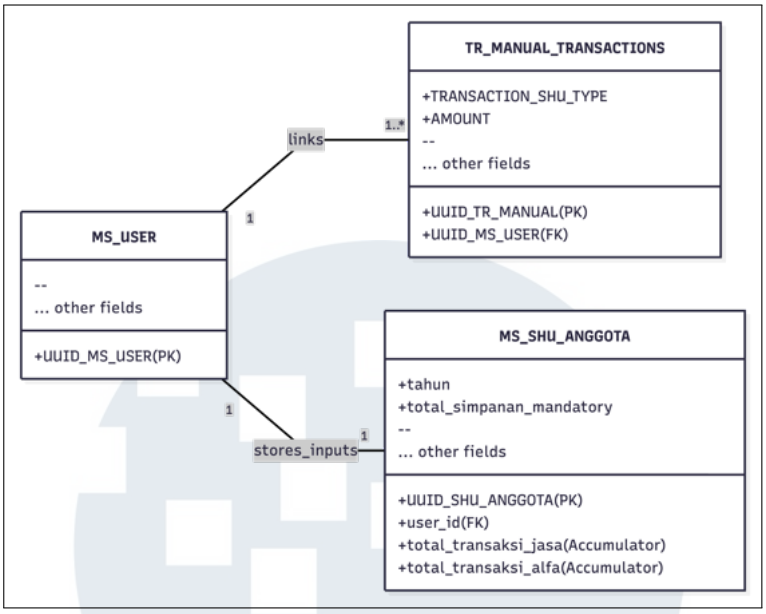
Gambar 3.3. *Sequence Diagram* Automasi Akumulasi SHU *Real-Time*

3.5.3 *Class Diagram*: Struktur Data SHU yang Diperbarui

Class Diagram menunjukkan struktur statis dari tiga entitas model utama yang menjadi fokus tugas. Diagram ini memvalidasi perubahan yang dilakukan pada skema database, yaitu:

1. Model `MS_SHU_ANGGOTA`: Menampilkan atribut-atribut akumulator baru (`total_transaksi_jasa`, `total_transaksi_alfa`, dan `total_simpanan_mandatory`) yang digunakan untuk basis perhitungan indeks.

2. Relasi: Menegaskan hubungan 1-to-N (one-to-many) antara MS_USER dengan TR_MANUAL_TRANSACTIONS melalui UUID_MS_USER, yang merupakan kunci utama berfungsinya automasi akumulasi.



Gambar 3.4. *Class Diagram* Model Kritis untuk Modul SHU

3.6 Kendala dan Solusi yang Ditemukan

3.6.1 Kendala yang Dihadapi

Selama pelaksanaan pengembangan modul Sisa Hasil Usaha (SHU), beberapa kendala teknis dan logistik yang dihadapi meliputi:

1. Kesalahan Dasar Perhitungan: Logika yang diwariskan dari tim sebelumnya masih menggunakan perhitungan sederhana (`Simpanan - Pinjaman`), yang bertentangan dengan kebutuhan formula indeks proporsional Koperasi.

2. Isu Data Masukan: Ketidaktersediaan kolom spesifik di model `TR_MANUAL_TRANSACTIONS` untuk menautkan transaksi langsung ke `UUID_MS_USER`. Hal ini menjadi kendala kritis untuk implementasi automasi SHU per anggota.

3. Konsistensi Skema Model: Model `MS_SHU_ANGGOTA` masih menggunakan nama kolom statis (`total_simpanan`, `total_pinjaman`, `selisih`) yang tidak relevan untuk menyimpan data masukan terakumulasi yang dibutuhkan oleh perhitungan indeks.

4. Isu Integritas Data Real-Time (*Race Condition*): Tantangan dalam memastikan bahwa proses akumulasi (`akumulasi = nilai lama +`

`nominal baru`) berjalan aman dan akurat di database saat terjadi permintaan simultan dari banyak pengguna.

5. Kendala Autentikasi Saat Pengujian: Server backend yang memerlukan validasi token (JWT) untuk endpoint `addManualTransaction`, yang mengharuskan proses *debugging* dan pengujian dilakukan setelah mendapatkan validasi sesi login.

3.6.2 Solusi yang Diterapkan

Untuk mengatasi kendala-kendala teknis yang dihadapi selama pengembangan, solusi-solusi berikut telah diimplementasikan:

1. Implementasi Formula Indeks: Logika pada `getAnggotaSHUData` sepenuhnya diubah, didukung oleh fungsi `calculateShuIndexes` yang menghitung rasio SHU berdasarkan data agregat (Total SP + SW) dan mengaplikasikan hasilnya ke input terakumulasi anggota.
2. Modifikasi Skema Kritis:
   (a) Penambahan Foreign Key: Menambahkan kolom `UUID_MS_USER` pada `TR_MANUAL_TRANSACTIONS` melalui migrasi SQL untuk memungkinkan pelacakan transaksi per anggota.
   (b) Model Akumulator: Mengubah model `MS_SHU_ANGGOTA` menjadi model akumulator dinamis dengan kolom `total_transaksi_jasa`, dsb., untuk menyimpan masukan SHU yang siap digunakan dalam kalkulasi.
3. Automasi Aman Menggunakan Sequelize.literal: Dibuat fungsi `autoAccumulateShu` yang dipicu oleh `KeuanganController.js`. Fungsi ini menggunakan Sequelize.literal untuk melakukan atomic increment (`field = field + nominal`) langsung di tingkat database, menjamin integritas data dan mencegah kesalahan pada proses akumulasi real-time.
4. Integrasi Token pada REST Client: Proses pengujian API disesuaikan dengan skema autentikasi, mengharuskan pengambilan Access Token melalui endpoint login dan menyertakannya di header Authorization: Bearer untuk setiap pengujian transaksi.