SISTEM PENDUKUNG KEPUTUSAN PRIORITAS PENAGIHAN ANGSURAN DALAM BENTUK *MICROSERVICE*

Muhammad Faiz Azam, Neny Rosmawarni
Program Studi Sistem Informasi
Fakultas Sains dan Teknolologi Informasi
Institut Sains dan Teknologi Nasional
Email : shouxrail@gmail.com, neny@istn.ac.id

ABSTRAK

Menurunnya perekonomian akibat pandemi COVID-19 menyebabkan pembayaran angsuran pembiayaan koperasi tersendat dan mengalami kemacetan. Untuk mengatasi hal tersebut maka diperlukan untuk melakukan penagihan yang tepat sasaran sesuai prioritas. Untuk memudahkan dan menambah akurasi prioritas penagihan, akan lebih baik jika penentuan prioritas diotomatiskan menggunakan Sistem Pendukung Keputusan (SPK). SPK tersebut perlu dapat memproses variabel penentu yang berubah-ubah, karena kondisi prioritas penagihan bisa saja berubah tergantung situasi. SPK tersebut juga perlu dapat diimplementasikan ke semua sistem aplikasi koperasi atau sistem informasi koperasi yang dapat berbeda-beda pada tiap koperasi. Pembuatan aplikasi pada penelitian ini menggunakan metode RAD (*Rapid Application Development*), dan metode SPK menggunakan metode SAW (*Simple Additive Weighting*). Dengan menggunakan Node JS penelitian ini menghasilkan aplikasi yang merupakan *microservice* dengan memanfaatkan teknologi REST API.

Kata kunci : Sistem pendukung keputusan, *simple additive weighting*, *microservice*, *application programming interface*, Node JS

ABSTRACT

Economic downturn due to the COVID-19 *pandemic has caused delays in payment of union's credits installments and has experienced bad credits. To overcome this, it is necessary to conduct collection that is right on target according to priority. To simplify and increase the accuracy of priorities of the collection, it would be better if the priority determination was automated using a Decision Support System* (DSS). *The* DSS *needs to be able to process variable determinants, because invoice collection priority conditions may change depends on the situation. The* DSS *also needs to can be implemented in all of union's application systems or union's information systems, which can be different with each union. The making of applications in this study using the* RAD *(Rapid Application Development) method, and for the* DSS *method this study using* SAW *(Simple Additive Weighting) for the method. By using* Node JS *this research produce application that is microservice by utilizing* REST API *technology.*

Keywords:
Decision support system, *simple additive weighting*, *microservice*, *application programming interface*, Node JS

I. PENDAHULUAN

Sudah hampir dua tahun pandemi COVID (*Coronavirus Disease*)-19 melanda dan telah mengguncang dunia. Pandemi COVID-19 mempengaruhi berbagai aspek dalam kehidupan, dan salah satunya adalah aspek ekonomi. Mulai dari karyawan PHK (Pemutusan Hubungan Kerja), bisnis gagal, perusahaan bangkrut, dan lain sebagainya. Begitu juga dengan Koperasi.

Koperasi yang memiliki sumber pemasukan utama dari simpan pinjam, tentu juga terdampak besar bersama dengan masyarakat yang merupakan nasabah atau anggota dari koperasi tersebut yang juga terdampak yang mengakibatkan kredit macet pada koperasi, sehingga tersendatnya perputaran modal pada koperasi.

Untuk mengatasi kredit macet tersebut, koperasi perlu melakukan penagihan angsuran piutang yang macet tersebut. Untuk melakukan penagihan angsuran piutang dengan sebaik-baik nya, penagihan perlu dilakukan sesuai kriteria prioritas yang dapat ditentukan. Penentuan prioritas penagihan akan sangat terbantu bila dapat menggunakan Sistem Pendukung Keputusan (SPK) yang dapat menentukan skala prioritas kredit macet yang perlu ditagih terlebih dahulu sesuai faktor-faktor yang dapat ditentukan sendiri.

Selain itu, dengan ditentukannya prioritas penagihan dan pengelompokan yang dihasilkan dari SPK. Pihak koperasi dapat terbantu untuk menentukan bagaimana cara penanganan yang tepat dan diperlukan untuk nasabah dengan kredit macet tersebut.

SPK ini dirancang sebagai *microservice* dari Sistem Informasi Koperasi (SIK) yang ada pada koperasi. Jadi koperasi tidak perlu menambahkan program ini kedalam SIK secara utuh.

Dengan menggunakan teknologi REST API, SIK hanya perlu mengirim data yang diperlukan untuk dianalisa pada SPK. Kemudian hasilnya akan dikirimkan kembali ke SIK dan langsung diolah pada SIK.

Penelitian dilakukan menggunakan metode *Simple Additive Weighting* (SAW). Metode SAW digunakan karena memiliki konsep dasar yang mudah diterapkan dalam berbagai kasus, terutama pada permasalahan kredit bank. Selain itu, metode SAW juga dapat digunakan untuk membandingkan hasil semua alternatif yang ada. Dengan begitu pihak koperasi dapat memilih bagaimana cara penanganan yang tepat untuk setiap kasus kredit macet.

II. TINJAUAN PUSTAKA

Sistem Pendukung Keputusan (SPK)

Menurut Kusrini dalam (Hendry Mandala Putra, 2013), Sistem Pendukung Keputusan (SPK) secara umum didefinisikan sebagai sebuah sistem yang memberikan pemecahan atau penanganan masalah dengan kondisi semi terstruktur maupun tidak terstruktur, pada keadaan dimana sulit untuk membuat keputusan (Devi Damayanti & Gafrun, 2021).

Simple Additive Weighting (SAW)

Menurut Kusumadewi dalam (Hendry Mandala Putra, 2013), metode *Simple Additive Wighting* (SAW) merupakan salah satu metode dari *Multi-Attribute Decision Making*. Metode ini juga sering dikenal dengan istilah metode penjumlahan terbobot (Devi Damayanti & Gafrun, 2021).

Rumus (1) berikut merupakan rumus normalisasi SAW :

$$r_{ij} = \begin{cases} \dfrac{X_{ij}}{\underset{i}{\text{Max}}\, X_{ij}} \\ \dfrac{\underset{i}{\text{Min}}\, X_{ix}}{i\quad Xij} \end{cases} \qquad (1)$$

Keterangan :

rij	= Nilai *rating* kinerja ternormalisasi
Xij	= Nilai atribut/*value* yang dimiliki setiap kriteria yang ada pada setiap alternatif
$\underset{i}{\text{Max}}\, X_{ij}$	= Nilai terbesar dari setiap kriteria
$\underset{i}{\text{Min}}\, X_{ix}$	= Nilai terkecil dari setiap kriteria
benefit	= Jika nilai terbesar adalah terbaik
cost	= Jika nilai terkecil adalah terbaik

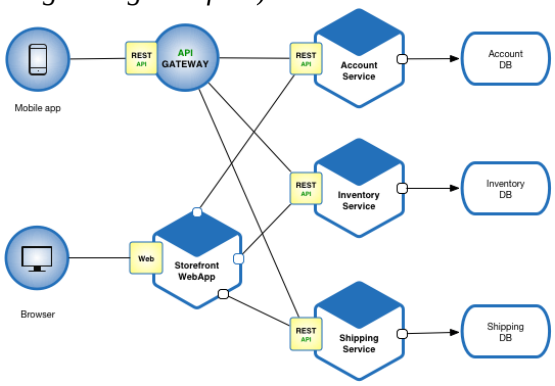
Rumus (2) berikut rumus penjumlahan perkalian matrikx ternormalisasi SAW :

$$V_i = \sum_{j=1}^{n} W_j r_{ij} \qquad (2)$$

Keterangan :
Vi = Rangking untuk setiap alternatif
Wj = Nilai bobot dari setiap kriteria
Ri = Nilai *rating* kinerja ternormalisasi

Microservices

Microservices adalah suatu *framework architecture* yang dipakai dalam pengembangan aplikasi *cloud* modern. Dengan *microservices* setiap aplikasi dibangun sebagai sekumpulan *service* dan setiap layanan atau sistem aplikasi *microservice* berjalan secara independen. Masing-masing *microservice* saling berkomunikasi melalu API (*Application Programing Interface*).



Gambar 1 Arsitektur *Microservices*

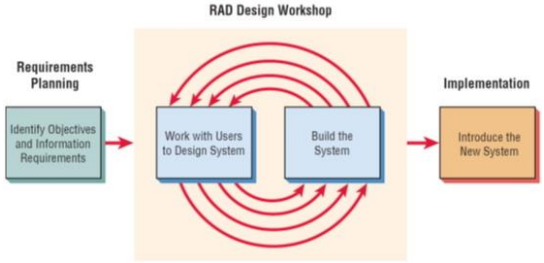
Gambar 1, menunjukkan setiap *microservice* memiliki fungsi masing-masing yang berbeda-beda dan saling terhubung menggunakan API.

REST API

RESTful API atau REST API merupakan penerapan dari API (*Application Programming Interface*). Sedangkan REST (*Representational State Transfer*) adalah sebuah arsitektur metode komunikasi yang menggunakan protokol HTTP (*Hypertext Transfer Protocol*) untuk pertukaran data dimana metode ini sering diterapkan dalam pengembangan aplikasi yang memerlukan pertukaran data yang intens. REST API digunakan untuk menjadikan performa sistem yang baik, cepat dan mudah dikembangkan terutama dalam komunikasi dan pertukaran data.

III. METODOLOGI PENELITAN

Metode Pengembangan Sistem



Gambar 2 *Rapid Application Depvelopment* (RAD) (Syabania & Rosmawarni, 2021)

a) Rencana Kebutuhan (*Requirement Planning*)

Berikut poin-poin yang digunakan sebagai dasar rencana kebutuhan :

- Prioritas penagihan tidak selalu sama kondisinya.
- Kondisi untuk prioritas penagihan pada tiap koperasi berbeda-beda
- Tiap koperasi menggunakan sistem SIK yang berbeda-beda.
- Data serta struktur data dari tiap koperasi juga berbeda-beda.

Kesimpulan kebutuhan yang diperlukan yaitu sistem SPK yang dapat fleksibel terhadap data yang dimasukkan. Dalam kasus ini berarti kriteria dan bobot nya dapat berubah sesuai keperluan. Serta perlu dibuat sebagai microservice yang berbasis API agar dapat terhubung dengan lebih dari satu jenis SIK.

b) Proses Desain Sistem (*Design Workshop*)

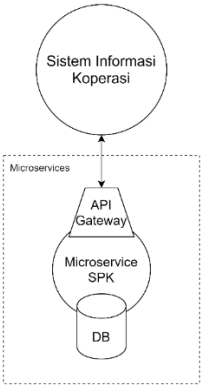
Untuk memenuhin kebutuhan pada tahap rencana kebutuhan, maka diperlukan sistem yang dapat menerima data yang berbeda-beda sesuai data yang dimasukkan. Untuk itu diperlukan *format* data yang sudah ditentukan agar sistem SPK dapat membaca data dari *format* tersebut.

JSON memiliki *support* untuk mengirim data dengan *format* yang cukup fleksibel dan dapat mendukung kebutuhan yang diperlukan sistem SPK prioritas penagihan.

c) Implementasi (*Implemantation*)

Implementasi desain sistem ke dalam bentuk program, pada penelitian ini menggunakan *text editor* Visual Studio Code dan *platform environment* JavaScript Node JS.

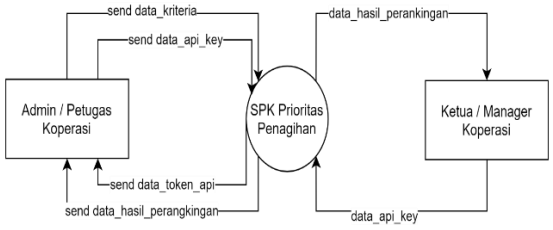
Arsitektur *Microservice* Sistem



Gambar 3 Arsitektur Sistem *Microservice* SPK Prioritas Penagihan – Hasil rancangan 2022

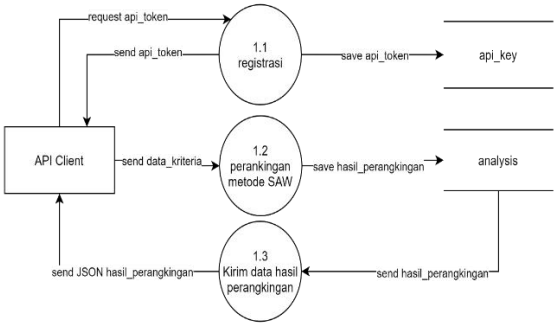
Rancangan DFD

Diagram Konteks



Gambar 4 Diagram Konteks SPK Prioritas Penagihan – Hasil rancangan 2021

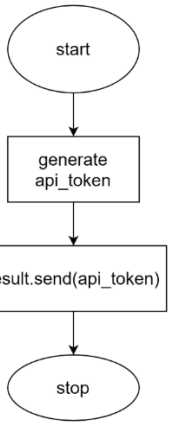
DFD *Level* 1



Gambar 5 DFD *Level* 1 SPK Prioritas Penagihan – Hasil rancangan 2021

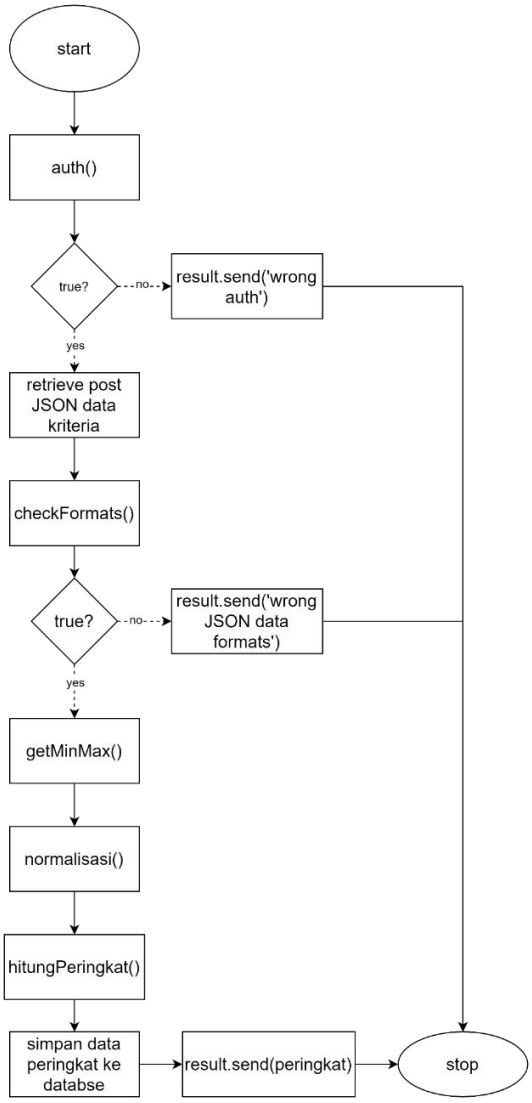
Rancangan *Flowchart*

Flowchart Registrasi



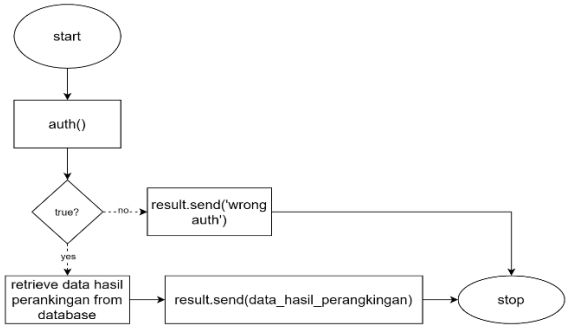
Gambar 6 *Flowchart* Registrasi SPK Prioritas Penagihan – Hasil rancangan 2021

Flowchart Perankingan



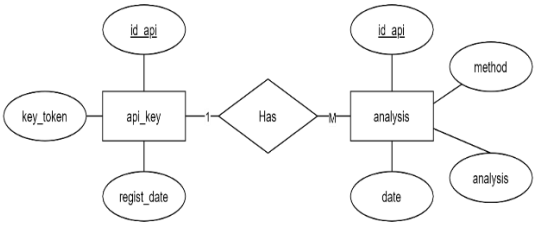
Gambar 7 *Flowchart* Perankingan SPK Prioritas Penagihan – Hasil rancangan 2021

Flowchart Request Data Hasil Perankingan



Gambar 8 *Flowchart Request* Data Hasil Perankingan SPK Prioritas Penagihan – Hasil rancangan 2021

Rancangan ERD



Gambar 9 Rancangan ERD SPK Prioritas Penagihan – Hasil rancangan 2021

Tabel 1 Keterangan detail atribut entitas api_*key* pada rancangan ERD

Nama Kolom	**Tipe Data**	**Keterangan**
*id*_api	*int* (10)	*Primary Key* *Not null*
key_token	*varchar* (100)	*Not null*
regist_date	*Datetime*	*Not null* *CURRENT_TIMESTAMP*

Tabel 2 Keterangan detail atribut entitas *analysis* pada rancangan ERD

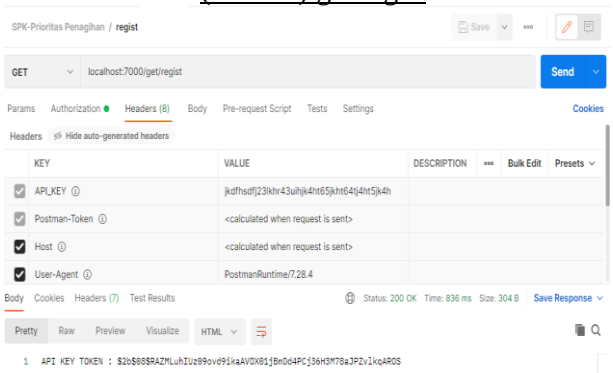
Nama Kolom	**Tipe Data**	**Keterangan**
*id*_api	*int* (10)	*Foreign Key* *Not null*
method	*varchar* (10)	*Not null*
analysis	json	*Not null*
Date	*datetime*	*Not null* *CURRENT_TIMESTAMP*

IV. HASIL DAN PEMBAHASAN

API *Gateway* Registrasi

API registrasi digunakan untuk mendaftarkan akun SPK prioritas penagihan untuk pemisahan data *user*/SIK. Hal ini dilakukan agar satu *microservice* SPK prioritas penagihan dapat digunakan oleh banyak aplikasi sekaligus.

Registrasi dapat dilakukan dengan cara melakukan *request* dengan metode *GET* dan *header* API_*Key* kode "jkdfhsdfj23lkhr43uihjk4ht65jkht64tj4ht5jk4h" ke alamat *url* "(domain)/get/regist".



Gambar 10 *Screenshot request* registrasi menggunakan Postman

Setelah itu akan muncul *feedback* dari SPK prioritas penagihan berupa teks yang berisi API_*Token*. Gambar 10 memperlihatkan contoh *request* registrasi menggunakan Postman.

API *Gateway* Perankingan

Setelah mendapatkan API_*Token* dari *request* registrasi, baru dapat melakukan *request* perankingan dengan cara menambahkan *header token* lalu isi dengan API_*Token* yang didapat dari *request* registrasi pada tahap sebelumnya.

Request perankingan membutuhkan data untuk diproses, agar dapat meng-*output* hasil perankingan dari data yang dikirim tersebut. Data yang dikirim berupa JSON dengan *format* seperti pada Gambar 11 berikut :

```
{
  "bobot_kriteria": {
    "penghasilan": 0.05,
    "pengeluaran": 0.05,
    ...
  },
  "jenis_kriteria": {
    "penghasilan": "benefit",
    "pengeluaran": "cost",
    ...
  },
  "data": [
    {
      "name": "BPY0000001",
      "penghasilan": 5000000,
      "pengeluaran": 20000000,
      ...
    },
    {
      "name": "BPY0000002",
      "penghasilan": 3000000,
      "pengeluaran": 2500000,
      ...
    },
    ...
  ]
}
```


Gambar 11 *Format* data JSON yang diminta *request* perankingan

Dapat dilihat pada Gambar 11 di atas. Data JSON yang dikirim memiliki *format* dengan data bobot kriteria, jenis kriteria, dan data piutang/pembiayaan yang ingin ditentukan prioritas nya. Penjelasan dari masing-masing data tersebut sebagai berikut :

- *Index* Bobot Kriteria

Index bobot kriteria berisi nilai bobot dari masing-masing kriteria yang dimasukkan untuk dijadikan kriteria perankingan.

- *Index* Jenis Kriteria

Index jenis kriteria berisi *parameter* untuk menentukan jenis bobot dari masing-masing kriteria. Data ini digunakan untuk menentukan nilai maksimum atau minimum dari tiap kriteria, tergantung dari apakah kriteria tersebut "*benefit*" atau "*cost*".

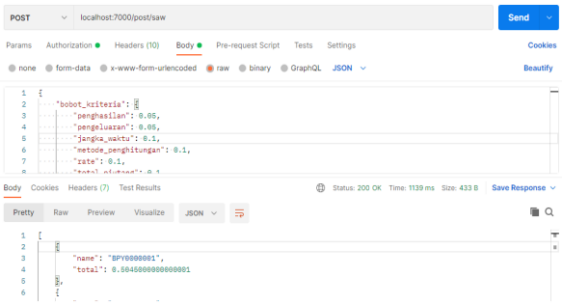
- *Index* Data

Index Data berisi data-data piutang yang ingin ditentukan *ranking* prioritas nya. Pada metode SAW disebut atribut data kriteria setiap alternatif. Data ini berisi nilai-nilai kriteria yang digunakan untuk perhitungan normalisasi. Selain itu data ini berisi *index* "*name*" yang digunakan untuk membedakan data piutang satu dengan yang lain.

Setelah menentukan data, kirim *request* perankingan ke *url* "(domain)/post/saw". Setelah itu SPK prioritas penagihan akan mengirim *feedback* berupa data JSON kembali yang berisi hasil perankingan dengan format seperti pada Gambar 12 berikut :

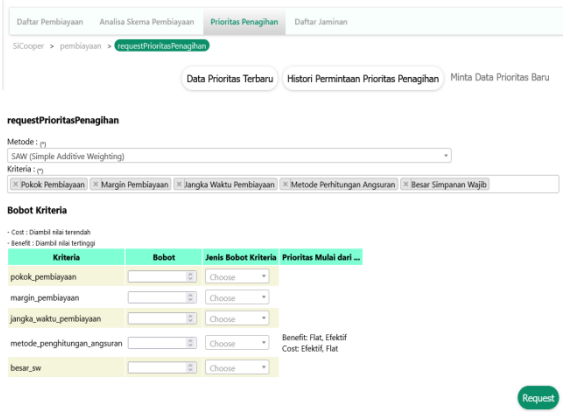

```
[
  {
    "name": "BPY0000001",
    "total": 0.5045000000000001
  },
  {
    "name": "BPY0000002",
    "total": 0.6460000000000001
  },
  ...
]
```


Gambar 12 *Format* data JSON hasil dari *request* perankingan



Gambar 13 *Screenshot request* perankingan menggunakan Postman

Gambar 13 di atas memperlihatkan contoh *request* perankingan menggunakan Postman.



Gambar 14 Tampilan SIK Halaman *Form Request* Prioritas Penagihan

Pada Gambar 14 ditampilkan tampilan SIK untuk halaman *form request* prioritas penagihan sebagai contoh simulasi *request* perankingan. Pada *field* kriteria dibuat dengan *tag* HTML *<select multiple>* agar pengguna dapat memilih kriteria yang ingin digunakan. Setelah pemilihan kriteria *field* bobot dan jenis bobot kriteria yang dipilih akan otomatis muncul.

Pada kriteria-kriteria tertentu ada *field* tambahan pada kolom "Prioritas Mulai Dari …" untuk menentukan nilai angka pada prioritas dengan *value* selain angka. *Field* tersebut menggunakan *tag* HTML *<select multiple>* yang mengambil *value* sesuai urutan *item* yang dipilih.

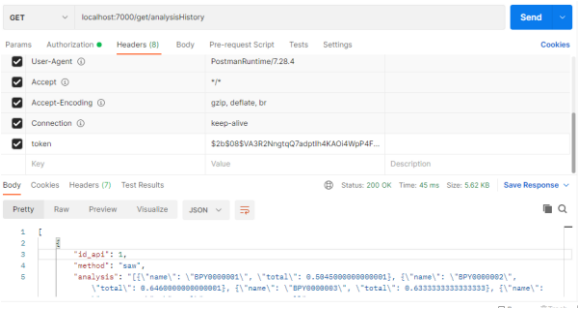
API *Gateway Request* Histori Perankingan

Request histori perankingan digunakan untuk me-*request* perankingan yang sebelum-sebelum nya pernah dilakukan. Dengan menggunakan API_Token yang didapat dari *request* registrasi, suatu SIK dapat me-*request* histori perankingan khusus SIK tersebut.

Request perankingan dilakukan dengan cara mengirim *request* dengan metode *GET* ke *url* "(domain)/get/analysisHistory". Lalu SPK prioritas penagihan akan mengirimkan *feedback* berupa data JSON dengan *format* seperti pada Gambar 15 berikut :

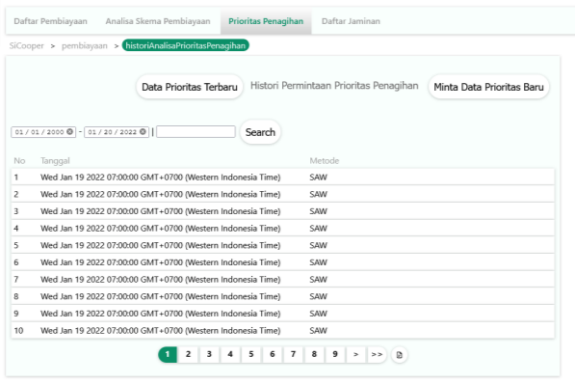

```
[
  {
    "id_api": 1,
    "method": "saw",
    "analysis": "[{\"name\": \"BPY0000001\", \"total\": 0.5045000000000001}, {\"name\": \"BPY0000002\", \"total\": 0.6460000000000001}, {\"name\": \"BPY0000003\", \"total\": 0.6333333333333333}, {\"name\": \"BPY0000004\", \"total\": 0.6333333333333333}]",
    "date": "2022-01-13T22:14:46.000Z"
  },
  {
    "id_api": 1,
    "method": "saw",
    "analysis": "[{\"name\": \"BPY0000001\", \"total\": null}, {\"name\": \"BPY0000002\", \"total\": null}, {\"name\": \"BPY0000003\", \"total\": null}, {\"name\": \"BPY0000004\", \"total\": null}]",
    "date": "2022-01-13T22:13:26.000Z"
  },
  ...
]
```


Gambar 15 *Format* data JSON yang dikirim dari *request* data histori perankingan



Gambar 16 *Screenshot request* histori perankingan menggunakan Postman

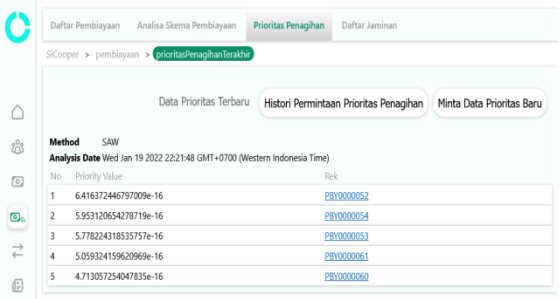
Untuk mendapatkan *feedback* berupa satu perankingan yang terakhir dilakukan, lakukan *request* ke *url* "(domain)/get/analysisHistory/last". Gambar 16 di satas memperlihatkan contoh *request* histori perankingan Postman.



Gambar 17 Tampilan SIK Halaman *List* Histori Prioritas Penagihan

Pada Gambar 17 ditampilkan tampilan SIK untuk halaman *list* histori prioritas penagihan sebagai contoh simulasi *request* histori perankingan. Daftar histori perangkingan dibuat per halaman 10 *item* serta pencarian yang dapat mencari *range* tanggal dan metode.

Jika di-*double click* pada salah satu list data, akan membuka halaman *ranking* prioritas penagihan dari data yang di-*double click* tersebut. Tampilah halaman tersebut dapat dilihat pada bagian selanjutnya.



Gambar 18 Tampilan SIK Halaman *Ranking* Prioritas Penagihan

Pada Gambar 18 ditampilkan halaman *ranking* prioritas penagihan dari SIK. Halaman ini menampilkan *ranking* prioritas penagihan dari *request* histori prioritas penagihan. Daftar *ranking* prioritas penagihan menampilkan *priority value* yang didapat dari hasil perankingan pada SPK Prioritas Penagihan.

Ditampilkan juga kode rekening pembiayaan dari pembiayaan yang ingin ditagih. kode rekening pembiayaan tersebut terhubung ke detail data pembiayaan dengan rekening tersebut.

Metode Pengujian Sistem

Metode pengujian yang digunakan pada penelitian ini menggunakan metode *White-Box Testing*. Metode *White-Box Testing* digunakan karena metode ini menguji dengan cara

menganalisis kode program tanpa memperdulikan UI atau tampilan dari aplikasi tersebut. Hal ini dirasa cocok pada kasus perancangan aplikasi seperti ini yang merupakan *microservice* dari aplikasi SIK.
Di bawah ini tahapan white box testing yang dilakukan pada penelitian ini. Mengikuti tahapan yang dilakukan pada penelitian (Pratala et al., 2020), yaitu :

a) Membuat *Flow Graph*

Control Flow Graph atau *Flow Graph* adalah grafik alir yang mewakili struktur kontrol dari suatu program atau modul.
Flow Graph memiliki *nodes* (V) dan *edges* (E) yang merepresentasikan proses dan jalur pada *source code* program di dalamnya.
Node-node pada *Flow Graph* memiliki sifat-sifar berikut untuk mendukung representasi *source code* tersebut :

- *Junction Node* - sebuah *node* dengan lebih dari satu panah yang memasukinya.
- *Decision Node* - sebuah *node* dengan lebih dari satu panah yang meninggalkannya.
- *Region* - area yang dibatasi oleh *edge* dan *node* (area di luar grafik juga dihitung sebagai *region*).
-

b)* Menentukan *Independent Path

Independent path adalah setiap jalur yang dilalui program saat melakukan eksekusi yang menunjukkan suatu *set* baru dari pemrosesan *statement* atau dari sebuah kondisi jalur baru.
Independent path pada *flow graph* harus melewati sedikitnya satu *edge* yang belum pernah dilewati oleh *path* sebelumnya. *Independent path* selalu dimulai dari *node* awal hingga ke *node* terakhir.

c) Menghitung *Cyclometic Complexity* (CC)

Cyclometic Complexity (CC) adalah metode pengukuran kompleksitas suatu program secara kuantitatif yang perkenalkan oleh Thomas J. McCabe, Sr. pada tahun 1976. CC mengukur jumlah jalur *linier-independent* melalui modul program yang dapat dilihat dari *flow graph*.
Nilai *Cyclometic Complexity* (CC) yang lebih rendah berarti program lebih mudah dipahami dan lebih kecil resiko untuk dimodifikasi. Untuk menghitung nilai CC, digunakan rumus sebegai berikut :

$$CC = E - N + 2 \times P \qquad (3)$$

Keterangan :
CC = *Cyclometic Complexity*
E = Jumlah *edges* pada *flow graph*
N = Jumlah *nodes* pada *flow graph*
P = Jumlah *predicates nodes* atau *node* penutup pada *flow graph*

Thomas J. McCabe, Sr. menjelaskan standarisasi *range* kompleksitas dari nilai CC. Yaitu sebagai berikut :

- *Range* 1 - 10 yang berarti prosedur program simpel dengan resiko rendah.
- *Range* 11 - 20 yang berarti prosedur lebih kompleks dengan resiko moderat atau sedang.
- *Range* 21 - 50 yang berarti prosedur kompleks dengan resiko tinggi.
- *Range* > 50 yang berarti kode tidak stabil dengan resiko sangat tinggi.

Pengujian

Tabel 4.1 Tabel rankuman hasil pengujian *white box*

PROSES YANG DIUJI	HASIL	KESIMPULAN
Registrasi	*Independent path* = 4 *Cyclometic Complexity* = 4	Nilai CC 4 yang masuk kedalam *Range* 1 - 10 yang berarti prosedur program simpel dengan resiko rendah.
Perankingan	*Independent path* = 26 *Cyclometic Complexity* = 11	Nilai CC 11 yang masuk ke dalam *Range 11 - 20* yang berarti prosedur lebih kompleks dari *range* 1 - 10, namun dengan resiko sedang atau menengah.
Request Histori Data Perankingan	*Independent path* = 4 *Cyclometic Complexity* = 4	Nilai CC 4 yang masuk kedalam *Range* 1 - 10 yang berarti prosedur program simpel dengan resiko rendah.

Pada Tabel 4.1 di atas dirangkum hasil dari pengujian *white box* yang dilakukan pada poin 4.2 BAB IV di atas.

V. PENUTUP

Kesimpulan

1. Aplikasi SPK Prioritas Penagihan pada penelitian ini dibuat sebagai *microservice* dari aplikasi SIK dan terhubung dengan SIK menggunakan API. Sehingga aplikasi ini dapat digunakan dengan lebih dari satu SIK dan dapat pula diintegrasikan dengan aplikasi lainnya yang membutuhkan perankingan prioritas penagihan.
2. Sebagai sistem pendukung keputusan yang dapat diintegrasikan dengan banyak aplikasi. Aplikasi ini juga dilengkapi dengan API_*Token* yang bertindak seperti *user account* untuk memisahkan data perankingan yang dilakukan tiap aplikasi.
3. Dilihat dari hasil pengujian *white box*, aplikasi ini dapat disebut berjalan dengan baik dengan skor *cyclometic complexity* terburuk nya adalah 11 yang masuk kedalam *range 11 - 20* yang berarti prosedur lebih kompleks dari *range* 1 - 10, namun dengan resiko masih moderat atau sedang. Dengan ini pembuatan prioritas penagihan dapat dilakukan dengan otomatis.

Saran

1. Penambahan metode perankingan. Karena sebagai apikasi berbasis *microservice* dengan REST API akan lebih baik lagi dengan banyak nya fasilitas yang dapat menambah fungsi kinerja SPK Prioritas Penagihan.
2. Pemantapan keamanan sistem. Karena metode yang digunakan saat ini masih menggunakan satu API_*Key* yang dirahasiakan dan *generated hash* dengan satu metode yaitu *hashing* bcrypt untuk API_*Token* sebagai *user account*. Akan lebih baik lagi jika metode keamanan baik akun atau API_*Token* maupun API_*Key* ditingkatkan lagi

DAFTAR PUSTAKA

Cahyadi, D., & Pratiwi, D. (2018). Sistem Informasi Pendukung Keputusan Pengangkatan Karyawan Tetap Berbasis Web Pada PT Dasa Windu Agung Di Bekasi. *Jurnal Rekayasa Informasi*, *7*(1), 31–42.

Devi Damayanti, S., & Gafrun. (2021). SISTEM PENDUKUNG KEPUTUSAN PENENTUAN KARYAWAN TELADAN DENGAN MENGGUNAKAN METODE SIMPLE ADDITIVE WEIGHTING (SAW). *JURNAL SISTEM INFORMASI DAN TEKNIK KOMPUTER*, *6*(2), 114–121. http://ejournal.catursakti.ac.id/index.php/simtek/article/view/103/118

Pratala, C. T., Asyer, E. M., Prayudi, I., & Saifudin, A. (2020). Pengujian White Box pada Aplikasi Cash Flow Berbasis Android Menggunakan Teknik Basis Path. *Jurnal Informatika Universitas Pamulang*, *5*(2), 111. https://doi.org/10.32493/informatika.v5i2.4713

Syabania, R., & Rosmawarni, N. (2021). PERANCANGAN APLIKASI CUSTOMER RELATIONSHIP MANAGEMENT (CRM) PADA PENJUALAN BARANG PRE-ORDER BERBASIS WEBSITE. In *Jurnal Rekayasa Informasi* (Vol. 10, Issue 1).

Nurmiati, S., & Natasurya, A. (2018). Sistem Penunjang Keputusan Penerimaan Calon Karyawan PT. Asa Foodenesia Abadi Bogor. *Jurnal Rekayasa Informasi*, *7*(1), 23–30. https://www.istn.ac.id

Istianto, W., & Baroqah Pohan, A. (2019). Sistem Pendukung Keputusan Pemberian Pinjaman Pada KOPWALI Tangerang Dengan Metode AHP dan SAW. In *IJCIT (Indonesian Journal on Computer and Information Technology)* (Vol. 5, Issue 1).

Sumitro, E., Marhaeni, & Yuyu, A. (2016). Sistem Pendukung Keputusan Monitoring Kedisiplinan Siswa SMK Berbasis Android. *Jurnal Rekayasa Informasi*, *5*(1), 1–13.