OTOMASI KONFIGURASI SISTEM INFORMASI BERDASARKAN ALIRAN DATA

SKRIPSI

Oleh :
PANDU SETIAWAN
NIM. 15650074

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2021

OTOMASI KONFIGURASI SISTEM INFORMASI BERDASARKAN ALIRAN DATA

SKRIPSI

Diajukan kepada:

Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk Memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh:
PANDU SETIAWAN
NIM. 15650074

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2021

LEMBAR PERSETUJUAN

OTOMASI KONFIGURASI SISTEM INFORMASI BERDASARKAN ALIRAN DATA

SKRIPSI

Oleh :
PANDU SETIAWAN
NIM. 15650074

Telah Diperiksa dan Disetujui untuk Diuji
Tanggal: 27 Mei 2021

Pembimbing I	Pembimbing II
Muhammad Ainul Yaqin, M.Kom	Ajib Hanani, M.T
NIP. 19761013 200604 1 004	NIDT.19840731201608011076

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Cahyo Crysdian

NIP.19740424 200901 1 008

LEMBAR PENGESAHAN

OTOMASI KONFIGURASI SISTEM INFORMASI BERDASARKAN ALIRAN DATA

SKRIPSI

Oleh :

PANDU SETIAWAN
NIM : 15650074

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 27 Mei 2021

Susunan Dewan Penguji:		**Tanda Tangan**
Penguji Utama	: Agung Teguh Wibowo Almais, M.T NIP. 19860301201802011235	( )
Ketua Penguji	: Syahiduz Zaman, M.Kom NIP. 197005022005011005	( )
Sekertaris Penguji	: M. Ainul Yaqin, M.Kom NIP. 197610132006041004	( )
Anggota Penguji	: Ajib Hanani, M.T NIDT. 19840731201608011076	( )

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

HALAMAN PERSEMBAHAN

Alhamdulillah puji syukur kehadirat Allah SWT yang memberikan kekuatan kepada saya hingga bisa sampai menyelesaikan kuliah S1 di kampus hijau tercinta. Sholawat serta salam kepada Nabi Muhammad SAW, yang membawa petunjuk terbaik kepada seluruh umat manusia.

Terima kasih kepada kedua orang tua saya, Ayah saya, Bapak Amin yang mendidik saya dari kecil hingga sekarang bisa menyelesaikan kuliah saya, Ibu saya tercinta, Sri Rumanti yang tiap hari mendo'akan saya, mendukung saya dalam melangkah, menemani saya setiap saat, mendidik saya dari lahir hingga mampu menyelesaikan segala kewajiban saya dibangku pendidikan.

Terima kasih kepada dosen-dosen yang telah sabar dan ikhlas dalam mendidik saya hingga mampu melewati seluruh ujian dari semua mata kuliah yang saya tempuh, terutama kepada Bapak M. Ainul Yaqin, M.Kom dan Bapak Ajib Hanani, M.T, semoga ilmu yang beliau amalkan berguna bagi seluruh mahasiswa dan semoga beliau diberikan kekuatan oleh Allah dalam berijtihad di dunia pendidikan hingga melahirkan anak didik yang mampu mengamalkan segala ilmu yang telah diberikan.

PERNYATAAN KEASLIAN TULISAN

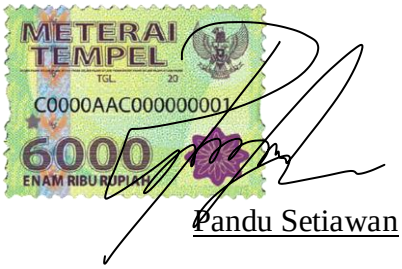
Saya yang bertanda tangan di bawah ini :

Nama	: Pandu Setiawan
NIM	: 15650074
Fakultas/Jurusan	: Sains dan Teknologi/Teknik Informatika
Judul Skripsi	: Otomasi Konfigurasi Sistem Informasi Bedasarkan Aliran Data

Menyatakan dengan benar bahwa Skripsi yang saya tulis ini benar-benar merupakan hasil karya sendiri, bukan merupakan pengambilan data, tulisan atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka. Apabila dikemudian hari terbukti atau dapat dibuktikan Skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 27 Mei 2021

Yang membuat Pernyataan



Pandu Setiawan

NIM. 15650074

MOTO

QS. Ali 'Imran Ayat 139

وَلَا تَهِنُوْا وَلَا تَحْزَنُوْا وَاَنْتُمُ الْاَعْلَوْنَ اِنْ كُنْتُمْ مُّؤْمِنِيْنَ

Artinya : “Dan janganlah kamu (merasa) lemah, dan jangan (pula) bersedih hati, sebab kamu paling tinggi (derajatnya), jika kamu orang beriman.”

(QS. Ali 'Imran Ayat 139)

Rosulullah SAW bersabda:

وَمَنْ سَلَكَ طَرِيقًا يَلْتَمِسُ فِيهِ عِلْمًا سَهَّلَ اللَّهُ لَهُ بِهِ طَرِيقًا إِلَى الْجَنَّةِ

Barangsiapa menempuh jalan untuk mendapatkan ilmu, Allah akan memudahkan baginya jalan menuju surga.

(HR. Muslim.)

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Syukur alhamdulillah penulis haturkan kehadirat Allah SWT yang telah melimpahkan Rahmat dan Hidayah-Nya, sehingga penulis dapat menyelesaikan studi di Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang sekaligus menyelesaikan Skripsi ini dengan baik. Selanjutnya penulis haturkan ucapan terima kasih seiring do'a dan harapan jazakumullah ahsanal jaza' kepada semua pihak yang telah membantu terselesaikannya Skripsi ini. Ucapan terima kasih ini penulis sampaikan kepada:

1. Prof. Dr. Abd. Haris, M.Ag selaku Rektor Universitas Islam NegeriMaulana Malik Ibrahim Malang
2. Dr. Cahyo Crysdian selaku ketua jurusan Teknik Informatika
3. Bapak M. Ainul Yaqin, M.Kom dan Bapak Ajib Hanani, MT selaku dosen pembimbing Skripsi, yang telah banyak memberikan pengarahan dan pengalaman yang berharga.
4. Segenap sivitas akademika Jurusan Teknik Informatika, terutama seluruh dosen, terima kasih atas segenap ilmu dan bimbingannya.
5. Ayahanda dan Ibunda tercinta yang senantiasa memberikan doa dan restunya kepada penulis dalam menuntut ilmu.
6. Semua pihak yang ikut membantu dalam menyelesaikan Skripsi ini baik berupa materiil maupun moril.

Penulis menyadari bahwa dalam penyusunan Skripsi ini masih terdapat kekurangan dan penulis berharap semoga Skripsi ini bias memberikan manfaat kepada para pembaca khususnya bagi penulis secara pribadi. Amin Ya Rabbal Alamin.

Wassalamu'alaikum Wr. Wb.

Malang, 27 Mei 2021

Penulis

DAFTAR ISI

DAFTAR GAMBAR

BAB 2

BAB 3

BAB 4

DAFTAR TABEL

DAFTAR KODE SUMBER

ABSTRAK

Setiawan, Pandu. 2021. **Otomasi Konfigurasi Sistem Informasi Berdasarkan Aliran Data.** Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
Pembimbing: (I) M. Ainul Yaqin, M.Kom. (II) Ajib Hanani, M.T

Kata Kunci : (*Data Flow Diagram*) *DFD, Extensible Markup Language (XML), Data Flow Execution Languange (DFEL), Web Service, Data Flow Management System (DFMS).*

Data Flow Diagram (DFD) merupakan notasi desain dari analisis terstruktur yang menggambarkan suatu sistem sebagai jaringan transformator data. Agar DFD dapat diimplementasikan pada sebuah sistem, maka perlu mengubah diagram DFD ke dalam format bahasa XML. XML merupakan sebuah platform bahasa independen yang mudah dimasukkan ke dalam lapisan persistensi, tidak bergantung pada platform, berisi informasi tekstual, merupakan standar terbuka, tidak bergantung pada bahasa, dapat diperluas sepenuhnya, mendukung struktur yang dapat dibagikan dan memungkinkan interoperabilitas. XML juga dapat digunakan dalam mendefinisikan komposisi fungsi fungsi *web service*. Proses pendefinisian komposisi *web service* ini salah satu penerapanya ada pada DFEL. DFEL merupakan sebuah bahasa yang mendeskripsikan proses eksekusi aliran data melalui *web service*. Format atau struktur DFEL ini terinspirasi dari Business Process Execution Language (BPEL). Proses pengimplementasian DFEL ini salah satunya ada pada DFMS. DFMS merupakan sebuah sistem yang digunakan untuk mengolah dan memanajemen file DFD. Format atau struktur dari DFMS ini terinspirasi dari *Bussiness Process Management System* (BPMS). Solusi yang ditawarkan dengan adanya DFMS ini antara lain, yang pertama dapat mengeksekusi file DFD, yang kedua mampu memetakan komponen-komponen yang ada pada DFD dengan *web service*, dan yang ketiga mampu meng-*generate* DFEL. Solusi yang ditawarkan tersebut dibangun berdasarkan alur pengembangan sistem metode *waterfall*. Hasil dari penelitian ini yaitu yang pertama kami berhasil menemukan atau berhasil mem-*parsing* komponen-komponen DFD yang disimpan dalam file yang berformat XML dan mengidentifikasi komponen-komponen yang ada dalam file DFD, yang kedua berhasil memetakan proses-proses atau komponen-komponen yang ada di dalam DFD dengan *web service*. Yang ketiga berhasil meng-*generate* DFEL yang dapat dieksekusi. Hasil dari pemetaannnya itu menunjukan bahwa dari 4 skenario itu semua dapat dieksekusi dan bisa jalan sesuai dengan kebutuhannya

ABSTRACT

Setiawan, Pandu. 2021. **Automation Configuration Information System Based on Data Flow.** Undergraduate Thesis. Informatic Department Enginering, Faculty of Science and Technology, State Islamic University of Maulana Malik Ibrahim Malang.
Supervisor: (I) M. Ainul Yaqin, M.Kom. (II) Ajib Hanani, M.T

Keywords : (Data Flow Diagram) DFD, Extensible Markup Language (XML), Data Flow Execution Languange (DFEL), Web Service, Data Flow Management System (DFMS).

Data Flow Diagrams (DFD) are design notations of structured analysis that describe a system as transformer network data. So that DFD can be implemented on a system, it is necessary to change / generate visual DFD into XML language format. XML is a platform independent language that is easy to incorporate into the persistence layer, is platform independent, contains textual information, is an open standard, is language-independent, fully extensible, supports shareable structures and enables interoperability. XML can also be used in defining the composition of the web service functions. One of the processes of defining the composition of this web service is DFEL. DFEL is a language that describes the process of executing data flow through the web service. This DFEL format is inspired by the Business Process Execution Language (BPEL). One of the DFEL implementation processes is in the DFMS. DFMS is a system used to process and manage Data Flow Diagram (DFD) files. The format of the DFMS is inspired by the Business Process Management System (BPMS). The solutions offered by the DFMS include, the first is able to execute DFD files, second is able to map existing components on the DFD with the web service, and third is able to generate DFEL. The solutions offered are built on the flow of the waterfall method system development. The results of this research are the first we have succeeded in finding / successfully parsing DFD components stored in an XML format file and identifying components in the DFD file, the second succeeded in mapping the process processes or components in the DFD with the web service. The third has succeeded in generating a DFEL that can be executed. The results of the mapping show that all of the 4 scenarios can be executed and can run according to their needs.

المستخلص

سيتياوان ,باندو.2021. **أتمتة تكوين نظام المعلومات على أساس تدفق البيانات**

مقال.قسم تخنية المعلوماتية ,كلية العلوم والتكنولوجي ,جامعة الدولة الإسالمية مولنا مالك إبراهيم مالنج

المشرف(1) محمد عين اليقين الما جستير(2) عجيب حناني, الما جستير

الكلمات المفتاحية : *DFMS, DFEL ,XML, DFD*

مخطط تدفق البيانات (DFD) هو ترميز تصميم للتحليل المنظم الذي يصف النظام على أنه شبكة محول بيانات. من أجل تنفيذ DFD في نظام ما ، من الضروري تغيير / جعل DFD المرئية إلى تنسيق لغة XML. XML هي لغة مستقلة عن النظام الأساسي يتم دمجها بسهولة في طبقة الاستمرارية ، وهي مستقلة عن النظام الأساسي ، وتحتوي على معلومات نصية ، وهي معيار مفتوح ، ومستقلة عن اللغة ، وقابلة للتوسيع بالكامل ، وتدعم بنية قابلة للمشاركة وتتيح إمكانية التشغيل البيني. يمكن أيضًا استخدام XML لتحديد تكوين وظائف خدمة الويب. DFEL هي عملية لتحديد تكوين خدمة الويب هذه. DFEL هي لغة تصف عملية تنفيذ تدفق البيانات من خلال خدمات الويب. نموذج أو هيكل DFEL مستوحى من لغة تنفيذ عمليات الأعمال (BPEL). أحد تطبيقات DFEL في DFMS. DFMS هو نظام يستخدم للتعامل مع وإدارة ملفات مخطط البيانات (DFD). شكل أو هيكل DFMS مستوحى من نظام إدارة عمليات الأعمال (BPMS). الحلول المقدمة من DFMS ، من بين أمور أخرى ، قادرة على إنشاء ملفات DFD ، والثاني قادر على تعيين المكونات على DFD مع خدمات الويب ، والثالث قادر على إنشاء DFELs. تعتمد الحلول المقدمة على تدفق التطوير لنظام نمط الشلال. نتائج هذه الدراسة هي الأولى التي نجحت في إيجاد / تحليل مكونات DFD المخزنة في ملف xml وتحديد المكونات في ملف DFD ، والثانية نجحت في تعيين عملية التشغيل أو مكونات المكون في DFD باستخدام خدمات الويب. نجح الثالث في إنشاء DFEL قابلة للتنفيذ. تظهر نتائج الخرائط أنه يمكن تنفيذ السيناريوهات الأربعة وتشغيلها وفقًا لاحتياجاتهم

BAB I
PENDAHULUAN

1.1. Latar Belakang

Data Flow Diagram merupakan notasi desain dari analisis terstruktur yang menggambarkan suatu sistem sebagai jaringan transformator data, baik itu transformator *input* maupun transformator *output*. *Data Flow Diagram* telah banyak digunakan dalam perancangan sistem informasi. Salah satu *library* yang menerapkan DFD adalah *Teensorflow*, yang dikembangkan oleh Google. Pengguna dapat memvisualkan operasi sistem informasi dan pengimplementasian sistem menggunakan DFD. Salah satu bentuk file simpanan DFD dalam format XML.

XML (*Extensible Markup Language*) merupakan sebuah platform bahasa independen yang mudah dimasukkan ke dalam lapisan persistensi, tidak bergantung pada platform, berisi informasi tekstual, merupakan standar terbuka, tidak bergantung pada bahasa, dapat diperluas sepenuhnya, mendukung struktur yang dapat dibagikan dan memungkinkan interoperabilitas [1]. XML yang terstruktur memiliki fungsi mentransfer informasi secara bersamaan menggunakan metadata melalui layanan jaringan atau *web service*. XML juga dapat digunakan untuk mentransfer file yang berkaitan dengan *Data Flow Diagram*, dengan menghilangkan masalah masalah yang berkaitan dengan pemahaman notasi diagram dan lebih berfokus pada arus informasi pada sistem. Selain itu, XML juga dapat digunakan dalam mendefinisikan komposisi fungsi-fungsi *web service* yang akan dieksekusi. Proses pendefinisian komposisi *web servic*e ini salah satu penerapanya ada pada DFEL (*Data Flow Execution Languange)*.

DFEL (*Data Flow Execution Languange*) merupakan sebuah istilah baru bahasa berbasis XML yang berfokus pada *Data Flow Diagram*. Setiap proses pada DFEL juga dapat diakses sebagai layanan *web service*. Format atau struktur DFEL ini terinspirasi dari BPEL (*Business Process Execution Language*). DFEL ini diterapkan untuk mengeksekusi DFD yang telah dipetakan dengan *web service*. Proses pengimplementasian DFEL ini salah satunya pada DFMS.

DFMS (*Data Flow Mangement System*) merupakan sebuah software digunakan untuk mengolah dan memanajement file *Data Flow Diagram* (DFD). Sebelum file DFD diolah pada sistem DFMS, file DFD diexport dalam bentuk XML. Pada file XML ini, informasi yang berkaitan dengan DFD dapat ditransfer secara bersamaan menggunakan metadata melalui *web service*. Pada Sistem DFMS ini terdapat dua *database*. Pertama yaitu *database* untuk ranah aplikasi DFMS, seperti *tenant* (penyewa), *web service*, *user* dan aplikasi. Kedua yaitu *database* untuk ranah DFD, seperti proses, *external entity* dan *data store*. Format atau struktur dari DFMS ini terinspirasi dari *Bussiness Process Management System* (BPMS).

BPMS (*Bussiness Process Management System*) merupakan sebuah platform untuk mengumpulkan, mengatur, menganalisis, mengoptimasi, dan meningkatkan proses bisnis. Banyak vendor BPMS yang digunakan sebagai manajemen proses bisnis, di antaranya Bonita BPM, Bizagi, Arabdox, Bonitad dan Joget [2]. Sistem BPMS saat ini telah mendukung dan mengimplementasikan konsep SOA. Konsep SOA ini berfungsi untuk pemodelan, desain, pengembangan, penyebaran aplikasi proses bisnis. Berbeda dengan BPMS yang befokus pada aliran proses, DFMS

sebagai *management sistem* yang tergolong baru berfokus pada aliran data. Aliran data ini didasarkan pada *Data Flow Diagram*.

Islam merupakan agama yang sempurna dengan berbagai kewajiban dan aturan yang telah diatur didalam Al-Qur'an dan Sunnah Rasulullah SAW. Segala aspek kehidupan manusia sudah diatur sedemikian rupa di dalam agama Islam. Sudah semestinya kita sebagai umat beragama Islam menjadi muslim yang taat dan beriman kepada Allah SWT serta ber-akhlak sesuai dengan aturan aturan dalam Al-Qur'an dan Sunnah. Al-Qur'an merupakan petunjuk, cahaya, sekaligus pedoman atau prinsip untuk kehidupan manusia dalam mengarungi/menjalankan kehidupan dunia agar selamat menuju kehidupan akhir yang sesungguhnya. Ada beberapa bentuk akhlak yang telah disebutkan dalam Al-Quran Surat Al Mu'minuun ayat 3 dan Al-Quran Surat Al Insyirah ayat 7 Allah berfirman:

وَٱلَّذِينَ هُمۡ عَنِ ٱللَّغۡوِ مُعۡرِضُونَ ٣

" Dan orang-orang yang menjauhkan diri dari (perbuatan dan perkataan) yang tiada berguna." (QS. Al Mu'minun : 3)

فَإِذَا فَرَغۡتَ فَٱنصَبۡ ٧

" Maka apabila kamu telah selesai (dari sesuatu urusan), kerjakan dengan sungguh-sungguh (urusan) yang lain." (QS. Al Insyirah : 7)

Dalam QS. Al Mu'minuun di atas menunjukkan bahwa sebagai seorang muslim yang beriman dan taat kepada Allah SWT, seharusnya menjauhi segala perbuatan yang tidak berguna atau sia-sia. Ayat tersebut mengarahkan agar manusia mampu berpikir mengenai perbuatan apa yang seharusnya dilakukan dan perbuatan

mana yang seharusnya ditinggalkan agar lebih efektif dan efisien dalam mempergunakan waktu. Sedangkan pada QS. Al Insyirah ayat 7 menunjukkan bahwa sebagai seorang muslim diperintahkan untuk beribadah sungguh-sungguh ketika telah menyelesaikan urusan atau pekerjaannya.

1.2. Masalah Penelitian

1. Bagaimana mengembangkan sistem manajemen *Data Flow Diagram* agar *executable* (dapat dieksekusi) ?
2. Bagaimana mengimplementasikan hasil *generate* DFD yang berformat XML ke *web service*?
3. Bagaimana menjalankan simulasi sistem berdasarkan *Data Flow Diagram*?

1.3. Tujuan Penelitian

1. Mengembangkan sistem manajemen *Data Flow Diagram* agar *executable* (dapat dieksekusi).
2. Mengimplementasikan hasil generate DFD yang berformat XML ke *web service*.
3. Menjalankan simulasi sistem berdasarkan *Data Flow Diagram*.

1.4. Batasan Masalah

Untuk menghindari meluasnya permasalahan yang ada serta keterbatasan ilmu dan kemampuan yang dimiliki peneliti, maka batasan masalah pada penelitian ini adalah:

1. Data uji yang digunakan adalah data skenario DFD yang tersimpan dalam format XML.

2. Proses *running* hanya bisa dilakukan pada satu DFD, yang artinya tidak bisa menjalankan dua atau lebih DFD secara bersamaan.
3. Proses *mapping* masih dilakukan secara manual yang artinya belum mendukung sistem kecerdasan.
4. Tidak mengakomodasi *split merge*.

1.5. Manfaat Penelitian

Adapun manfaat yang diharapkan dari penelitian ini adalah

1. Memberikan kontribusi baru terhadap layanan otomasi konfigurasi sistem informasi berdasarkan aliran data.
2. Otomasi konfigurasi sistem informasi ini diharapkan mampu memberikan kemudahan, yaitu kemudahan untuk memberikan informasi pada setiap alur data *(Flow)* dari yang sederhana sampai yang kompleks.

BAB II
STUDI LITERATUR

2.1 DFD (*Data Flow Diagram*)

Data Flow Diagram (DFD) merupakan notasi desain dari analisis terstruktur, yang menggambarkan suatu sistem sebagai jaringan transformator data, baik itu transformator *input* maupun transformator *output* [3]**.** DFD telah banyak digunakan dalam perancangan sistem informasi atau analisis sistem perangkat lunak. *TensorFlow*, sebuah *library* atau pustaka software pembelajaran *open-source* yang dikembangkan oleh Google juga didasarkan pada DFD [3]. Notasi pada DFD memiliki empat simbol utama yaitu

1. Proses

   Merupakan komponen yang menggambarkan bagian dari sistem yang mentransformasikan *input* menjadi *output*.

2. *Data Store*

   Komponen ini digunakan untuk membuat model sekumpulan paket data. *Data store* ini biasanya berkaiatan dengan penyimpanan penyimpanan, seperti *file* atau *database* yang berkaitan dengan penyimpanan secara komputerisasi.

3. *Data Flow*

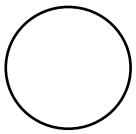
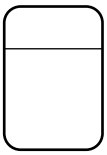
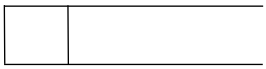
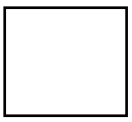
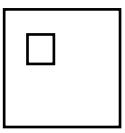
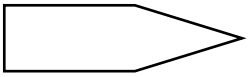
   Komponen *Data flow* digambarkan dengan anak panah, yang menunjukan arah menuju dan keluar dari suatu proses. Alur data ini digunakan untuk menerangkan perpindahan data atau paket data / informasi dari satu bagian *system* ke bagian lainnya.

4. Terminator / Entitas luar

   Terminator mewakili entitas eksternal yang berkomunikasi dengan sistem yang sedang dikembangkan. Istilah terminator juga dapat disebut sebagai (*external entity*). Terminator / *external entity* dapat berupa orang, sekelompok orang, organisasi, departemen di dalam organisasi, atau perusahaan.

Pada dasarnya ada beberapa notasi yang berbeda untuk menggambar diagram aliran data *(Yourdan & Coad dan Gane & Sarson)* ditunjukan seperti tabel 2.1 berikut.

Tabel 2.1 Perbedaan Notation Yourdan & Coad dan C.Gane & T.Sarson

	Yourdan & Coad	C.Gane & T.Sarson
Data Flow		
Proses		
Data Store		
External Entity		
Material Flow		

Dari perbedaan di atas tentunya menimbulkan sebuah pemahaman yg ambigu dari notasi kedua model DFD. Masalah lain terkait yaitu mengenai *Time Consuming*. DFD membawa proses yang cukup banyak untuk sampai ke *user*, sehingga hal ini menyebabkan proses yang dilakukan cinderung melambat. Untuk menangani hal tersebut agar berjalan lebih cepat dan efisien, diperlukan sebuah komunikasi melalui jaringan atau *network* [1]. Komunikasi melalui *network* telah berkembang dari waktu ke waktu. Untuk itu digunakanlah XML sebagai bahasa komunikasi data. Berikut merupakan contoh file XML sebagai hasil representasi DFD.

Gambar 2.1 Contoh Respresentasi DFD ke XML

Pada gambar 2.1, bahwa file XML berisi metadata. File metadata meliputi data-data yang terkait dengan diagram konteks atau diagram level 0. Konteks diagram terdiri dari satu proses, entitas eksternal, dan *data flow*. Selain konteks diagram, file metadata ini juga terdiri dari diagram level 1. Diagram level satu ini memiliki multi proses, entitas eksternal, dan datastore.

2.2 XML (*Extensible Markup Language*)

XML merupakan sebuah platform bahasa independen yang mudah dimasukkan ke dalam lapisan persistensi, tidak bergantung pada platform, berisi informasi tekstual, merupakan standar terbuka, tidak bergantung pada bahasa, dapat diperluas sepenuhnya, mendukung struktur yang dapat dibagikan dan memungkinkan interoperabilitas [1]. XML yang terstruktur memiliki fungsi mentransfer informasi secara bersamaan menggunakan metadata melalui *web service*. XML juga dapat digunakan untuk mengirim file yang berkaitan dengan DFD, dengan menghilangkan masalah masalah yang berkaitan dengan pemahaman notasi diagram dan lebih berfokus pada arus informasi pada sistem. Selain itu XML juga dapat digunakan dalam mendefinisikan komposisi fungsi-fungsi *web service* yang akan dieksekusi.

Blok penyusun dasar dokumen XML adalah elemen, yang ditentukan oleh tag. Sebuah elemen memiliki awal dan tag akhir. Semua elemen dalam dokumen XML terkandung dalam elemen terluar yang dikenal sebagai elemen root. XML juga dapat mendukung elemen bersarang, atau elemen di dalam elemen. Kemampuan ini memungkinkan XML mendukung struktur hierarki. Nama elemen mendeskripsikan konten elemen, dan struktur mendeskripsikan hubungan antar elemen.

Kemampuan XML untuk merepresentasikan atribut, turunan, dan data karakter semuanya memberikan cara yang lebih efektif dan deskriptif untuk merepresentasikan data [4]. Fitur yang sama ini menjadikan XML cara yang bagus untuk merepresentasikan informasi yang sangat detail, termasuk informasi jenis data. XML dapat mencakup informasi tentang jenis data dan tipe data khusus, dan setiap elemen dapat memiliki atribut untuk mendeskripsikan properti tertentu dari

suatu elemen. XML juga mendukung *namespace* yang terkadang digunakan dalam tipe data kompleks. XML sangat populer di antara banyak platform teknologi perusahaan seperti Java, Oracle, dan .NET, sehingga pengguna teknologi ini akan sering meminta XML sebagai media pilihan.

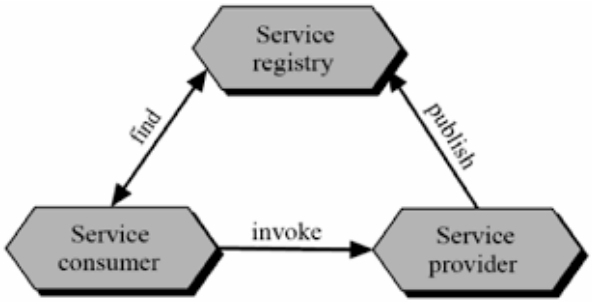
Ada banyak cara pengimplementasian XML di PHP, dan semuanya berguna dalam situasi yang berbeda. Ada tiga pendekatan utama untuk dipilih dan semuanya memiliki kelebihan dan kekurangan [4]:

1. *SimpleXML* adalah yang paling mudah digunakan. Pengimplementasiannya mudah digunakan dan dipahami, didokumentasikan dengan baik, dan menyediakan antarmuka yang sederhana (seperti namanya) untuk menyelesaikan pekerjaan. *SimpleXML* memang memiliki beberapa batasan, tetapi direkomendasikan untuk sebagian besar aplikasi.
2. DOM berguna saat sebuah proyek menemui beberapa batasan di *SimpleXML*. Pengimplementasiannya lebih rumit, tetapi ada sejumlah kecil operasi yang tidak dapat dilakukan dengan *SimpleXML*. Ada fungsi bawaan untuk memungkinkan konversi antara dua format ini, jadi sangat umum untuk menggunakan kombinasi keduanya dalam aplikasi,
3. *XMLReader*, *XMLWriter*, dan *XMLParser* adalah cara tingkat rendah untuk menangani XML. Secara umum, pengimplementasian ini rumit dan tidak intuitif tetapi memiliki keuntungan besar. Metode ini berbasis aliran dan oleh karena itu tidak memuat seluruh dokumen XML ke dalam memori sekaligus.

2.3 SOA (*Service Oriented Architecture*)

SOA merupakan pendekatan untuk membangun sistem yang berfokus pada pembuatan aplikasi dari kombinasi bisnis dan layanan lainnya serta pendekatan ini memberikan potensi untuk menciptakan perusahaan yang gesit yang memiliki fleksibilitas maksimum dan waktu reaksi minimum [5]. Dalam arti lain SOA dapat didefinisikan sebagai sebuah pendekatan arsitektur yang menggunakan service sebagai konstruksi dasar untuk mendukung pengembangan yang cepat dengan biaya yang rendah dan memiliki kemudahan dalam mengatur komposisi aplikasi yang terdistribusi dalam lingkungan yang heterogen sekalipun [6]

Dalam menggunakan pendekatan SOA, semua komponen aplikasi dimodelkan sebagai service. Terdapat tiga aktor yang terlibat dalam SOA [7]. Seperti terlihat pada Gambar 1.3 di bawah :



Gambar 2.2 Model desain (SOA) Service Oriented Architecture

1. *Service provider*; merupakan entitas yang menciptakan sebuah *service* dan membuatnya dapat digunakan oleh entitas yang lain
2. *Service consumer*; merupakan sebuah entitas atau sesorang yang menggunakan *service* yang diciptakan oleh *service provider*

3. *Service registry*; merupakan sebuah direktori terpusat yang mana *service provider* dapat menerapkan layanan yang telah diciptakan, dan tempat *service consumer* dapat menemukan layanan yang dibutuhkan

SOA merupakan sebuah pendekatan untuk memecahkan permasalahan yang besar dan memecahnya menjadi sekumpulan kecil *service* untuk meyelesaikan permasalahan yang lebih spesifik. SOA tidak terikat pada sebuah teknologi, tetapi lebih menekankan pada pendekatan untuk membangun sebuah sistem yang modular. Selain tiga aktor yang terlibat dalam model desain SOA, SOA juga terdiri dari empat komponen, antara lain :

1. *Message*, berupa data yang dibutuhkan untuk menyelesaikan sebuah pekerjaan yang dipertukarkan antara satu *service* dengan *service* yang lain
2. *Operation*, berupa fungsi yang dimiliki oleh *service* untuk memproses dan menghasilkan sesuatu. Fungsi ini akan berinteraksi dengan fungsi lainnya untuk menyelesaikan sebuah pekerjaan
3. *Service*, merepresentasikan sekumpulan operasi yang saling berkaitan utuk menyelesaikan sebuah pekerjaan
4. *Process*, merupakan *business rule* yang menentukan operasi mana yang akan digunakan untuk mencapai tujuan tertentu.

Dalam implementasi *Service Oriented Architecture*, salah satu teknologi yang paling sering digunakan adalah *web service*. Teknologi ini memberikan standar untuk pertukaran pesan dan deskripsi dari *service* itu sendiri. *Web service* didefinisikan oleh W3C sebagai sebuah sistem perangkat lunak yang didesain untuk

mendukung interoperabilitas antar perangkat dalam sebuah jaringan. Sebuah *web service* memiliki beberapa standar yang dibutuhkan dalam mengimplementasikan *Service Oriented Architecture* [7], antara lain :

1. *Simple Object Access Protocol* (SOAP), mendefinisikan struktur dari pesan dan protocol yang digunakan untuk standar dalam pertukaran informasi di jaringan yang heterogen. Dapat digunakan sebagai protokol pertukaran pesan dalam SOA.
2. *Web Service Description Language* (WSDL), sebuah bahasa yang digunakan oleh *service provider* untuk menggambarkan *service* yang diciptakannya dalam konteks interoperabilitas. Dengan menggunakan SOAP dan WSDL, operasi "*invoke*" pada model desain SOA (seperti yang ditunjukkan pada Gambar 2.2)
3. *Universal Description, Discovery and Integration* (UDDI) sebagai pilihan teknologi untuk mengimplementasikan *service broker*. Digunakan untuk mengimplementasikan operasi "*publis*h" dan "*invok*e" pada model desain SOA (seperti yang ditunjukkan pada Gambar No. 2.2)

Hubungan dari standar-standar *web service* tersebut dapat dilihat pada gambar 2.3.

Gambar 2.3 Standarisasi web service

2.4 BPM (*Business Process Management*)

BPM (*Business Processes Management*) merupakan bidang manajemen yang dapat didefinisikan sebagai paradigma yang mencakup metode, teknik, dan alat untuk mendukung desain, implementasi, manajemen dan analisis operasional proses bisnis [8]. Tujuan dari BPM adalah untuk menilai secara strategis proses yang dilakukan perusahaan dan untuk terus meningkatkan efektifitas dan efisiensi proses bisnis dalam organisasi, seperti:

1. Mencapai biaya yang lebih rendah
2. Meningkatkan kualitas
3. Meningkatkan produktifitas dan daya saing dalam kaitanya dengan organisasi lain dari area bisnis yang sama.

Seperti yang telah dibahas di berbagai sumber, bahwa BPM merupakan sebuah subjek dengan banyak pandangan, definisi dan persektif. Sifatnya yang multi-disiplin, seringkali mudah untuk menemukan bahan penelitian proses bisnis di berbagai aspek. Untuk memahami secara perspektif terminology dan fitur BPM, seorang individu atau organisasi harus mulai memahami terhadap siklus BPM. Metodologi siklus BPM secara umum dapat diimplementasikan dalam langkah langkah berikut:

1. Analisis

   Analisis komprehensif dilakukan untuk menemukan dan mengidentifikasi proses yang dapat dibuat atau dioptimalkan untuk memenuhi persyaratan bisnis atau meningkatkan kinerja. Spesifikasi untuk solusi desain dapat diturunkan dari analisis ini.

2. Desain

   Desain suatu proses melibatkan alur kerja yang mencakup interaksi manusia-ke-manusia, sistem ke sistem, atau interaksi manusia ke sistem. Desain harus bertujuan untuk mengurangi kesalahan dan mempertahankan prosedur operasi standar yang relevan atau perjanjian tingkat layanan.

3. Pemodelan

   Setelah desain proses siap, dapat dimodelkan dengan menggunakan berbagai nilai *input* untuk mengamati perilakunya. Jika perilaku yang tidak diinginkan diamati, perubahan desain dapat dilakukan secara iteratif. Alat perangkat lunak tersedia untuk memodelkan dan mengevaluasi proses secara efektif.

4. Eksekusi

   Model proses dapat dieksekusi menggunakan mesin aturan bisnis untuk mengatur eksekusi proses.

5. Pemantauan

   Selama pelaksanaan, proses dapat dipantau untuk mengumpulkan data untuk kinerja, kesalahan, dan kepatuhan. Pemantauan memungkinkan perusahaan untuk mengevaluasi solusi BPM yang dilakukan terhadap model desain yang sesuai dan relevan. Data yang dikumpulkan oleh pemantauan *real-time* juga dapat digunakan oleh perangkat lunak analitik prediktif untuk mengantisipasi masalah di masa depan.

6. Optimalisasi

   Data dari fase pemodelan dan pemantauan dapat digunakan untuk mengidentifikasi area solusi yang dapat ditingkatkan untuk memperoleh efisiensi yang lebih tinggi dan nilai yang lebih baik.

2.5 *Bussines Process Management Suite* (BPMS)

Saat ini, ada berbagai macam perangkat lunak (BPMS) yang memungkinkan pengelolaan siklus *Bussiness Process* untuk mempermudah manajemen proses dalam lingkungan bisnis. Secara definisi BPMS merupakan serangkaian perangkat lunak yang dibuat untuk mengumpulkan, mengatur, menganalisis, mengoptimasi, dan meningkatkan proses bisnis yang ada. BPMS dapat digunakan oleh staf TI maupun bukan TI, dan seringkali diperlukan koordinasi di antara keduanya. Selain itu BPMS adalah yang terbaik untuk perbaikan sistem yang sedang berlangsung dan dirancang untuk perbaikan berkelanjutan. Berikut merupakan beberapa keuntungan dari BPMS, di antaranya:

1. Meningkatkan efisiensi secara otomatis dan mengatur langkah langkah dari suatu proses.
2. Meningkatkan transparansi dengan memantau kinerja proses dan mengambil tindakan berdasarkan informasi yang diperoleh
3. Meningkatkan kualitas dengan menyederhanakan proses, menghindari kesalahan manual, memungkinkan aturan bisnis menjadi lebih fleksibel, serta membuat proses bisnis berubah lebih cepat.
4. Memiliki fleksibilitas untuk mengubah proses atau menambahkan langkah langkah baru, serta mampu beradaptasi dengan tuntutan bisnis perusahaan

Sederhananya, perangkat lunak BPM memungkinkan sebuah organisasi untuk menciptakan efisiensi lebih, dengan mengoptimasi dan menstandarisasi proses bisnis serta memastikan bahwa semua anggota organsasi menyelesaikan dan mencapai tujuan yang sama.

BPM *tool* atau BPMS juga tidak terlepas dari proses *upgrade*. Alasannya karena muncul berbagai kompleksitas yang harus diselesaikan untuk mengatasi berbagai solusi proses bisnis. Untuk Alasan itu, dan untuk memenuhi permintaan pelanggan suatu perusahaan selama beberapa tahun terakhir, banyak vendor BPMS yang digunakan sebagai *tool* manajemen proses bisnis, di antaranya Bonita BPM, Bizagi, Arabdox, Bonita, Joget, dll [2]. Sistem BPMS saat ini telah mendukung dan mengimplementasikan konsep SOA. Konsep SOA ini berfungsi untuk pemodelan, desain, pengembangan, penyebaran aplikasi proses bisnis.

2.6 Word Similarity

Mengukur kesamaan antara kata kata, kalimat dan dokumen adalah komponen penting dalam berbagai tugas seperti pencarian informasi, pengelompokan dokumen, disambiguasi kata, penilaian esai otomatis, jawaban singkat, terjemahan mesin dan untuk mempersingkat sebuah teks. Menemukan kesamaan antara kata-kata adalah bagian mendasar dari kesamaan teks yang kemudian digunakan sebagai tahap utama untuk persamaan kalimat, paragraf dan dokumen [9]. Kata-kata bisa serupa dalam dua cara, leksikal dan semantik. Kata-kata serupa secara leksikal jika mereka memiliki urutan karakter yang sama. Kata-kata serupa secara semantik jika mereka memiliki hal yang sama, saling berlawanan, digunakan dengan cara yang sama, digunakan dalam konteks yang sama.

Implementasi dalam penelitian ini lebih menekankan pada metode *semantic*. Metode *semantic* ini berperan pada proses *mapping* antara DFD dengan *web service*. Proses *mapping* menggunakan parameter yaitu banyaknya kata kunci yang sama antara salah satu metadata *web service* dengan kata kunci pencarian *query*

workflow yang akan di *mapping*. Hasil *mapping* pada setiap DFD akan disimpan pada file DFD tersebut dengan menambahkan id beserta data parameter dari proses *web service* yang sesuai dengannya.

2.7 DFEL (Data Flow Execution Languange)

Standarisasi manajemen proses bisnis dan teknologi *workflow* telah mengalami perkembangan yang cukup signifikan. BPEL sebagai salah satu bahasa berbasis XML yang memodelkan proses bisnis juga berperan terhadap serangkaian layanan berbasis web. BPEL diakui sebagai *service composition language* yang dapat dieksekusi dan berorientasi pada proses [10]. Berbeda dengan BPEL, DFEL (*Data Flow Execution Languange*) juga cukup berperan dalam ranah manajemen proses bisnis. Fokus utama dari DFEL adalah *dataflow*. Setiap proses dari DFEL juga dapat diakses sebagai layanan WEB. Spesifikasi DFEL juga hampir sama dengan BPEL. DFEL menggunakan Skema XML untuk definisi struktur data dan pengambilan elemen XML. BPEL sering dikaitkan dengan BPMN yang merupakan bahasa eksekusi untuk merepresentasikan proses bisnis secara grafis, sedangkan DFEL sering dikaitkan dengan DFD yang merupakan bahasa eksekusi untuk merepresentasikan *data flow* secara grafis.

2.8 Penelitian Terkait

Tabel 2.2 Penelitian terkait

Peneliti, Judul, Tahun	Uraian	Hasil & Kesimpulan
Gandhis Ulta, M. Ainul Yaqin. Implementasi Metode *Semantic Similarity* untuk Pengukuran Kemiripan Makna Antar Kalimat (2019)	Pada penelitian ini menggunakan perhitungan kemiripan berdasarkan contextual menggunakan WS4J. Pendekatan yang digunakan untuk menghitung nilai *word similarity* yaitu pendekatan Wu Palmer, path, dan lin. Setelah diketahui nilai *word similarity* dan nilai bobot kriterianya, kemudian menghitung matriks nilai kemiripan kata dengan nilai bobot masingmasing kriteria yang sesuai untuk menentukan nilai *sentence similarity*.	Jenis kriteria yang digunakan untuk pembobotan kalimat dalam menentukan nilai *sentence similarity* yaitu *noun* dan *verb* dengan bobot masing-masing adalah 0.25 dan 0.75.
Yuliani Ningsih, M.Ainul Yaqin**.** Komposisi *Web Service* Menggunakan *Cosine Similarity* Untuk Menyusun *Business Process Executing Languange* (BPEL) (2019)	Pada penelitian ini Menggunakan metode *cosine similarity*. Metode ini digunakan untuk untuk menghitung kemiripan antar *activity*, sehingga dapat menghasilkan komposisi dari rangkaian *web service* yang sesuai dengan proses bisnis. Hasil dari komposisi *web service* tersebut dapat secara otomatis disusun menjadi BPEL. Sehingga hasil akhir BPEL dapat dimanfaatkan sebagai acuan eksekusi dalam pemrograman.	Dalam melakukan komposisi *web service* menggunakan algoritma *cosine similarity*, diperoleh nilai *similarity* sebesar 1; 1,782; 2,3754. Sehingga nilai persamaan tertinggi tersebut dapat diambil sebagai patokan dalam komposisi *web service*
Usvir Kaur, Dheerendra Singh. BPEL – Business Process Execution Language Process Creation and Fault Localization Framework (2014)	BPEL telah diakui sebagai bahasa komposisi layanan yang dapat dieksekusi dan berorientasi pada proses. Bahasa berbasis format XML yang dikenal sebagai BPEL berbeda dari sintaks dan semantik dari bahasa program lain. Pada penelitian ini BPEL digunakan untuk menciptakan sebuah environtment untuk melokalisasi kesalahan dan untuk meminimalisir kesalahannya menggunakan Algoritma optimasi BEES..	Algoritma optimasi BEES digunakan untuk menghitung jumlah kesalahan yang terjadi. Hasil menunjukkan bahwa kombinasi berbeda dari tag dasar yang berbeda menghasilkan terjadinya kesalahan yang lebih rendah
Swapna Salil Kolhatkar. XML Based Representation of DFD (2011)	XML adalah platform independen, informasi tekstual dan benar-benar dapat dikembangkan. XML terstruktur dan merupakan cara terbaik untuk mentransfer informasi bersama dengan metadata melalui jaringan. XML dapat digunakan untuk mentransfer informasi yang terkait dengan DFD.	XML dapat digunakan untuk mentransfer informasi yang terkait dengan DFD, di sana dengan menghilangkan masalah yang berkaitan dengan memahami notasi diagram dan lebih berkonsentrasi pada aliran informasi dalam sistem

BAB III
ANALISIS DAN PERANCANGAN SISTEM

3.1 Gambaran Umum

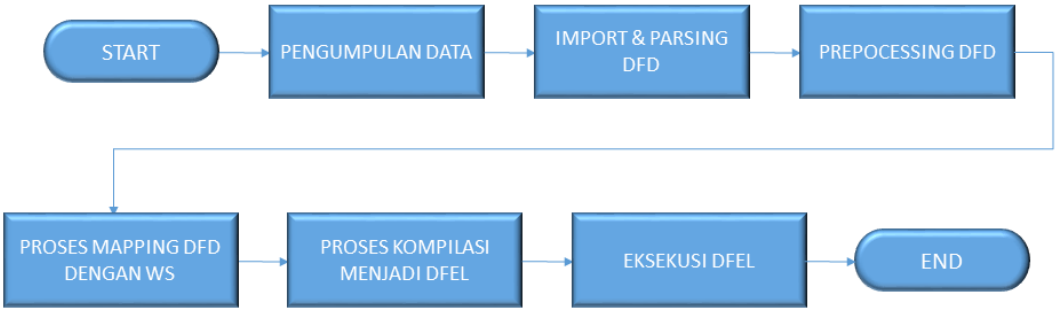
Sistem yang dibangun adalah otomasi konfigurasi sistem informasi. Sistem informasi disini bermodelkan platform *low-code* atau *no-code* yang membuat manajemen bisnis lebih cepat dan mudah. Fokus utamanya adalah aliran data. Sistem informasi ini memanfaatkan platform bahasa XML. Bahasa XML ini diperoleh dari representasi DFD. File XML yang merupakan hasil representasi DFD ini kemudian di *import* dan *parsing*. Tujuan dari *parsing* ini adalah untuk membedakan dan memilah proses, *flow, external entity* dari DFD yang dibuat. Setelah proses *parsing*, tahap selanjutnya yaitu melakukan *preprocessing* DFD. Tahap ini bertujuan untuk mendapatkan *key* atau kata kunci yang digunakan untuk proses *mapping* DFD dengan *web service*. Hasil proses *mapping* DFD dengan *web service* inilah yang kemudian dikompilasi menjadi DFEL.

3.2 Sumber Data

Sumber data yang digunakan dalam penelitian ini berupa sumber data primer dan sumber data sekunder. Sumber data primer pada penelitian ini mengacu pada file berekstensi XML. File berekstensi XML berisi metadata dari DFD. Sedangkan untuk data sekunder diperoleh dari literatur-literatur yang berkaitan.

3.3 Prosedur Penelitian

Prosedur penelitian menjelaskan tentang bagaimana alur penelitian akan dilaksanakan. Prosedur penelitian ini dimulai dari tahapan pengumpulan data sampai perancangan aplikasi.



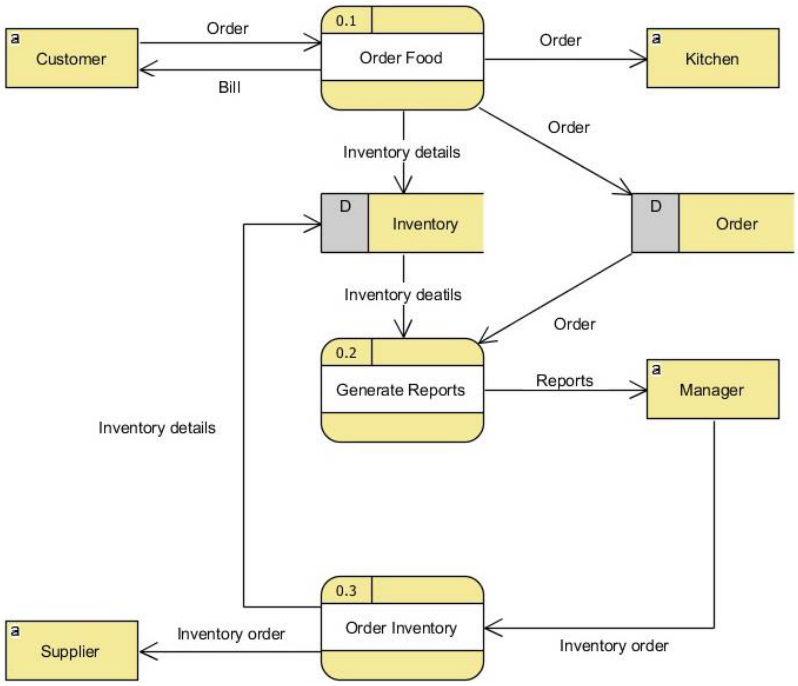
Gambar 3.1 Prosedur penelitian

3.3.1 Sumber Data

Sumber data yang digunakan dalam penelitian ini berupa sumber data primer dan sumber data sekunder. Sumber data primer pada penelitian ini mengacu pada file berekstensi XML. File berekstensi XML berisi metadata dari *Data Flow Diagram*. Sedangkan untuk data sekunder diperoleh dari literature-literature yang berkaitan.

3.3.2 *Import & Parsing* DFD

Pada proses Import dan Parsing data ini melibatkan file XML dari DFD yang dibuat. Tujuannya untuk memecah suatu rangkaian masukan XML. Berikut merupakan diagram atau DFD yang digunakan sebagai contoh untuk *parsing* data.



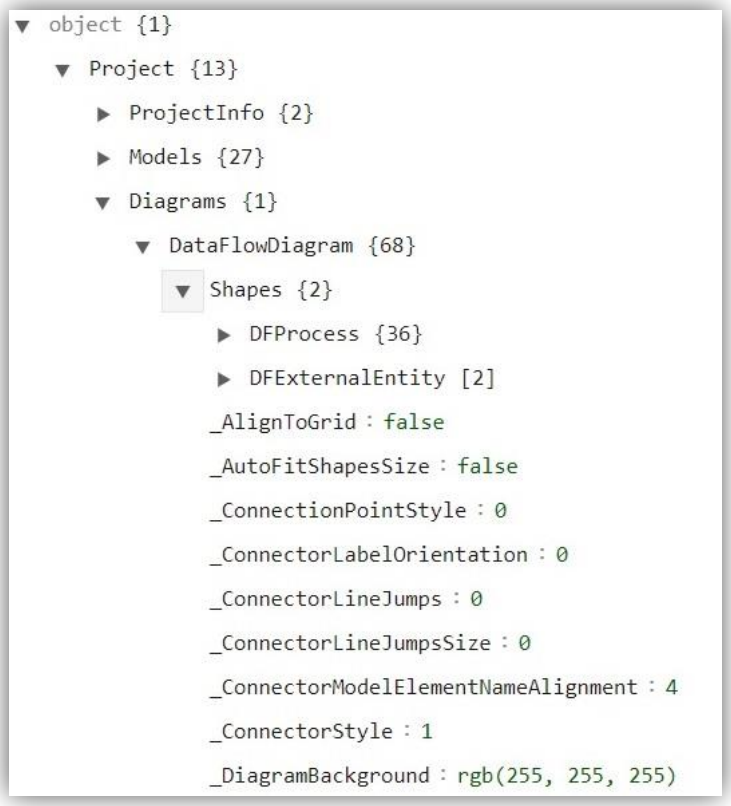
Gambar 3.2 Contoh konteks diagram

Contoh konteks diagram gambar 3.2 kemudian diexport menjadi XML. Berikut merupakan hasil *export* DFD menjadi XML.

```
<DFProcess Background="rgb(122, 207, 245)" ConnectToPoint="true" ConnectionPointType="2" CoverConnector="false" Foreground="rgb(0, 0, 0)"
Height="70" Id="IZpoHg6GAqBWqgsG" MetaModelElement="MppoHg6GAqBWqgrQ" Model="MppoHg6GAqBWqgrQ" ModelElementNameAlignment="9" Name="Generate
Reports" OverrideAppearanceWithStereotypeIcon="true" ParentConnectorDTheta="0.0" ParentConnectorHeaderLength="40" ParentConnectorLineLength="
10" PresentationOption="4" PrimitiveShapeType="0" RequestDefaultSize="false" RequestFitSize="false" RequestFitSizeFromCenter="false"
RequestResetCaption="false" RequestResetCaptionFitWidth="false" RequestResetCaptionSize="false" RequestSetSizeOption="0" Selectable="true"
ShowAllocatedFrom="0" ShowAllocatedTo="0" ShowStereotypeIconName="0" Width="110" X="338" Y="326" ZOrder="8">
    <ElementFont Color="rgb(0, 0, 0)" Name="Dialog" Size="11" Style="0"/>
    <Line Cap="0" Color="rgb(0, 0, 0)" Transparency="0" Weight="1.0">
        <Stroke/>
    </Line>
    <Caption Height="30" InternalHeight="-2147483648" InternalWidth="-2147483648" Side="FreeMove" Visible="true" Width="110" X="0" Y="20"/>
    <FillColor Color="rgb(244, 234, 154)" Style="1" Transparency="0" Type="1"/>
</DFProcess>
<DFProcess Background="rgb(122, 207, 245)" ConnectToPoint="true" ConnectionPointType="2" CoverConnector="false" Foreground="rgb(0, 0, 0)"
Height="70" Id="oZpoHg6GAqBWqgsI" MetaModelElement="0ppoHg6GAqBWqgrP" Model="0ppoHg6GAqBWqgrP" ModelElementNameAlignment="9" Name="Order·Food"
 OverrideAppearanceWithStereotypeIcon="true" ParentConnectorDTheta="0.0" ParentConnectorHeaderLength="40" ParentConnectorLineLength="10"
PresentationOption="4" PrimitiveShapeType="0" RequestDefaultSize="false" RequestFitSize="false" RequestFitSizeFromCenter="false"
RequestResetCaption="false" RequestResetCaptionFitWidth="false" RequestResetCaptionSize="false" RequestSetSizeOption="0" Selectable="true"
ShowAllocatedFrom="0" ShowAllocatedTo="0" ShowStereotypeIconName="0" Width="110" X="338" Y="101" ZOrder="10">
    <ElementFont Color="rgb(0, 0, 0)" Name="Dialog" Size="11" Style="0"/>
    <Line Cap="0" Color="rgb(0, 0, 0)" Transparency="0" Weight="1.0">
        <Stroke/>
    </Line>
    <Caption Height="15" InternalHeight="-2147483648" InternalWidth="-2147483648" Side="FreeMove" Visible="true" Width="110" X="0" Y="20"/>
    <FillColor Color="rgb(244, 234, 154)" Style="1" Transparency="0" Type="1"/>
</DFProcess>
```

Gambar 3.3 Contoh hasil export ke XML

Dari file XML gambar 3.3, kemudian di-*parsing*. Proses *parsing* menggunakan format JSON. Format JSON ini memungkinkan untuk pertukaran dan penyimpanan data. Berikut merupakan contoh hasil *parsing* dari file XML yang sudah di *export* dari DFD.



Gambar 3.4 Hasil parsing (format JSON)

3.3.3 Pre-Processing DFD

Preprocessing merupakan tahapan awal dalam *text mining* yang bertujuan untuk melakukan seleksi data yang akan diproses pada setiap dokumen [9]. Tahap *preprocessing* yang digunakan dalam penelitian ini antara lain [9]:

1. *Case Folding*

   Case folding merupakan tahapan yang mengubah huruf kapital dalam dokumen menjadi huruf kecil.

2. *Tokenzing*

   Tokenizing merupakan tahap memecah dokumen menjadi kumpulan kata. *Tokenizing* dapat dilakukan dengan menghilangkan tanda baca dan memisahkan per spasi.

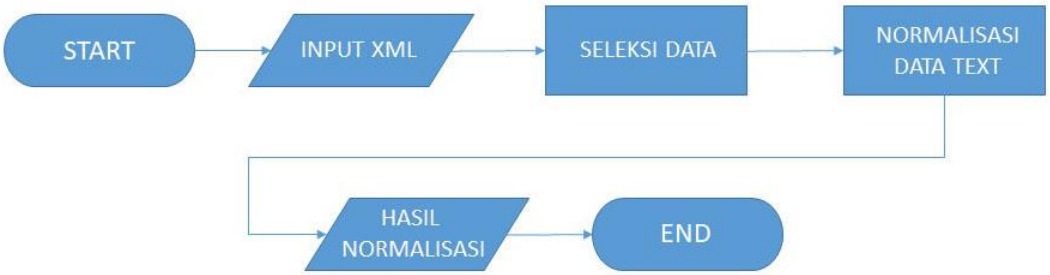
3. *Filtering*

   Filtering adalah proses penghilangan kata-kata tidak penting, jika termasuk dalam daftar *stoplist* (kata tidak penting) maka kata-kata tersebut akan dihilangkan sehingga yang tersisa dianggap sebagai kata penting.

4. *Stemming*

   Stemming merupakan proses merubah kata-kata menjadi kata dasar, sehingga lebih mudah untuk diolah.

Berikut merupakan tahapan penelitian dalam melakukan *preprocessing*.



Gambar 3.5 Flowchart pre-processing

Sesuai gambar 3.5, file XML hasil dari export DFD akan dilakukan seleksi data. Data dari hasil seleksi akan dinormalisasi sesuai kebutuhan. Tahap normalisasi ini semisal penghilangan simbol atau karakter yang tidak mendukung pada jenis bahasa pemrograman yang akan digunakan atau seperti menjadikan semua atribut

data menjadi *lowercase* (huruf kecil). Tujuan dari normalisasi *text* ini memudahkan analisis pada tahap selanjutnya serta mengurangi kerancuan pada tahap atau proses *mapping*.

3.3.4 Proses *Mapping* DFD dengan WS

Pada tahap ini setelah proses normalisasi *text* yaitu proses *mapping* ke *web service*. Tahap ini akan memetakan setiap proses alur datanya dengan proses pada *web service repository* yang sesuai dengan alur proses data tersebut. Sebelum dipetakan, setiap proses *web service* pada *repository* akan dibuatkan metadatanya terlebih dahulu. Pembuatan metadata ini dilakukan agar mempermudah pemetaan proses *mapping* proses alur data dan proses *web service*. Pembuatan metadata pada proses *web service* ini dengan memecah setiap id atau nama proses pada *web service* kedalam bentuk *array* dan menggolongkan metadata tersebut kedalam kategori-kategori yang sesuai.

Proses *mapping* antara DFD dengan WS ini menggunakan metode kemiripan antar kalimat / kata yang digunakan [9]. *Mapping* ini menggunakan paramaeter yaitu banyaknya kata kunci yang sama antara salah satu metadata *web service* dengan kata kunci pencarian *query workflow* yang akan di *mapping* [9]. Hasil *mapping* pada setiap DFD akan disimpan pada *file* DFD tersebut dengan menambahkan id beserta data parameter dari proses *web service* yang sesuai dengannya.

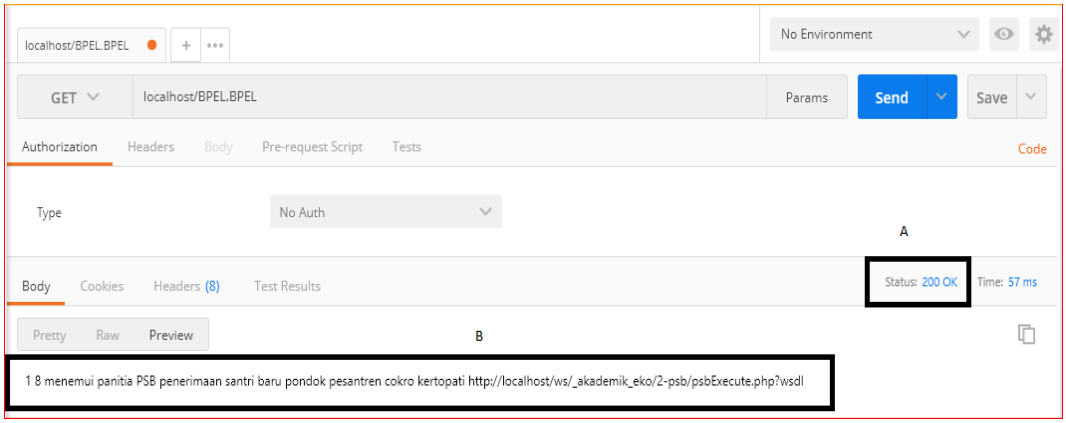
3.3.5 Proses Kompilasi Menjadi DFEL

Setelah proses *mapping* maka tahap selanjutnya adalah proses kompilasi menjadi DFEL. Skema DFEL dalam hal ini hampir sama dengan BPEL. Proses kompilasi merupakan proses penggabungan atau penyisipan *Uniform Resource Identifier* (URI) pada DFD. Penyisipan URI ini diletakan di bagian Proses DFD pada *web service*. Setelah penyisipan URI pada bagian proses DFD tahap selanjutnya adalah menjalankan DFEL. Dalam penelitian ini proses *running* DFEL menggunakan *environment* dengan standarisasi sendiri. Proses kompilasi tidak menggunakan *java library*. Hal ini dikarenakan perbedaan struktur DFEL dengan BPEL. DFEL menggambarkan informasi aliran data, sedangkan BPEL menggambarkan aliran proses, meskipun skema DFEL dengan BPEL hampir sama.

Setelah proses *running*, selanjutnya adalah pengujian DFEL. Pengujian DFEL dilakukan hampir sama dengan pengujian BPEL. Pada penelitian ini pengujian DFEL menggunakan *postman*. Pengujian ini dilakukan untuk menunjukan hasil DFEL dapat berjalan dengan baik dan dapat digunakan pada aplikasi lain. Berikut merupakan contoh hasil pengujian BPEL yang nantinya dijadikan landasan atau dasar dalam pengujian DFEL. Pada contoh gambar 3.6 ini menunjukan hasil bahwa pada aplikasi *postman* data menunjukan *response* berhasil saat melakukan *request.*

Gambar 3.6 Contoh hasil request file BPEL

Pada contoh gambar 3.6 menunjukkan bahwa *postman* melakukan *request* ke file BPEL untuk mendapatkan detail data dari BPEL [9].



Gambar 3.7 Contoh detail data dari BPEL

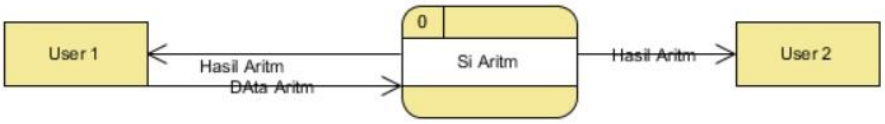
Pada Gambar 3.7 menunjukkan bahwa *response* yang diberikan oleh *postman* berhasil dikirim [9].

BAB IV
ANALISA HASIL DAN PEMBAHASAN

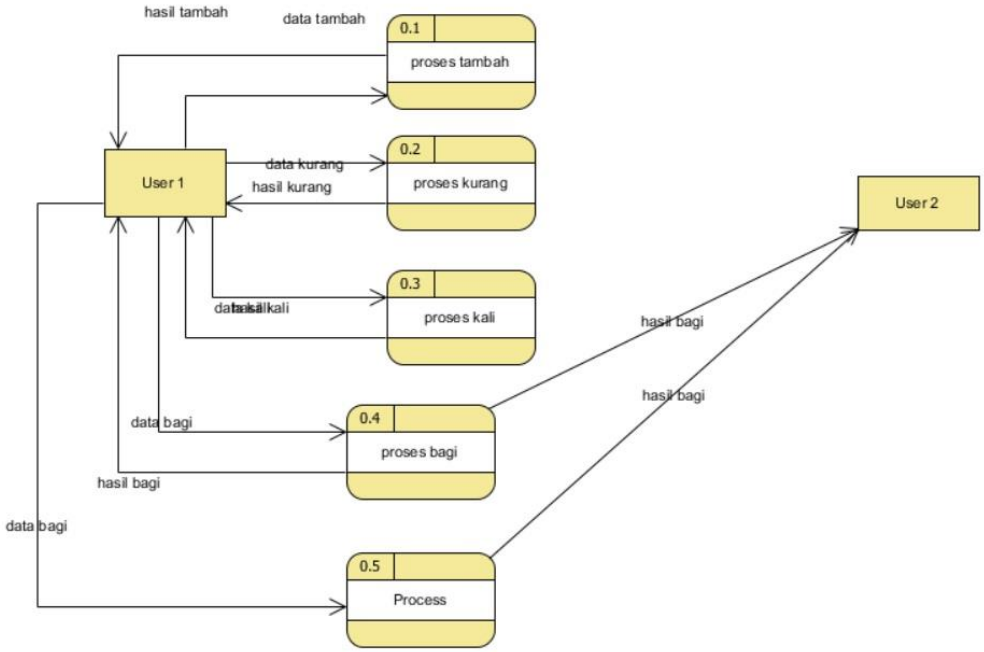
Bab ini menjelaskan tentang hasil serta pembahasan sistem yang telah dibuat. Oleh karena itu, dilakukanlah pengujian terhadap sistem dengan menjalankan program menggunakan contoh model *aritmatic*. Pengujian ini dilakukan untuk mengetahui apakah program sudah bekerja dengan baik atau belum. Dalam aplikasi ini terdapat dua *database* yang dibuat yaitu *database* untuk aplikasi dan *database* hasil *generate* dari ERD yang disesuaikan dengan model DFD. Dalam uji aplikasi, rancangan ERD dan DFD dibuat menggunakan *software* Visual Paradigm.

4.1 Model DFD Sebagai Uji Coba Aplikasi

Model DFD aritmatika ini dijadikan sebagai data uji untuk menjalankan sistem apakah sistem sudah berjalan atau belum. Model DFD ini dibuat menggunakan *software* Visual Paradigm. Sebelum model DFD diproses, model DFD harus di *export* dalam XML. Berikut merupakan model DFD yang digunakan sebagai data uji coba untuk menjalankan sistem.

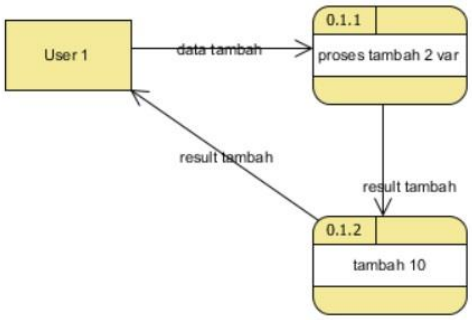


Gambar 4.1 Data uji DFD aritmatik Level 0



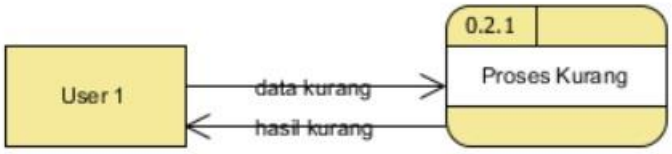
Gambar 4.2 Data uji DFD aritmatik Level 1

Konteks diagram pada gambar 4.2 memiliki berbagai proses aritmatika seperti proses tambah, proses kurang, proses bagi, dan proses kali, serta dua *external entity*. Berikut merupakan hasil *decompose* dari proses-proses dalam konteks diagram pada gambar 4.2.



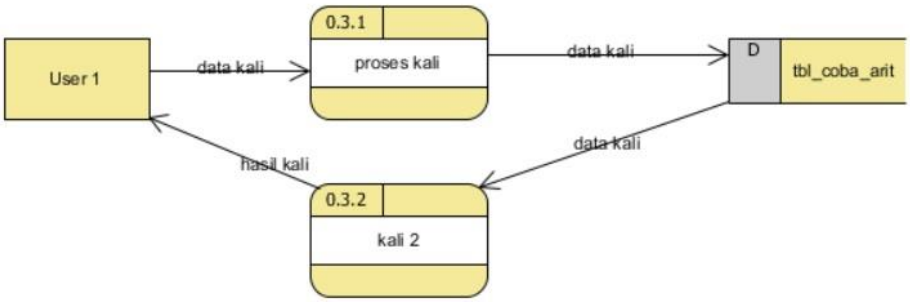
Gambar 4.3 DFD level 2 (proses tambah)

Pada diagram gambar 4.3 terdapat satu *external entity* dan dua diagram proses. Proses pertama (0.1.1) merupakan proses tambah dua variabel x & y, kemudian hasil proses pertama di teruskan ke diagram proses ke dua (0.1.2) yaitu menjumlahkan hasil (x+y) dengan 10, kemudian hasil akan ditampilkan ke *user* pertama.



Gambar 4.4 DFD level 2 (proses kurang)

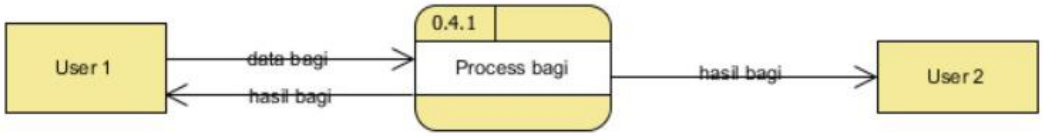
Diagram gambar 4.4 terdapat satu *external entity* dan satu diagram proses. Hasil proses pengurangan (0.2.1) langsung ditampilkan ke *user* pertama, tanpa harus melawati proses lain seperti proses tambah di atas.



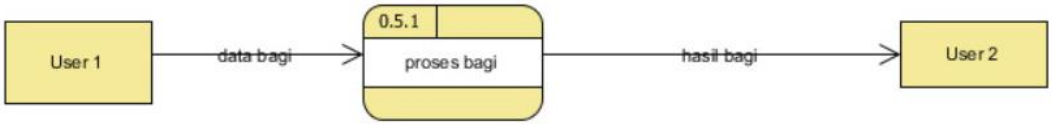
Gambar 4.5 DFD level 2 (proses kali)

Pada diagram di atas terdapat satu *external entity*, dua diagram proses dan satu *datastore*. Hasil perkalian dari proses pertama (0.3.1) disimpan pada *data store*, kemudian data dari data strore diproses kembali, dalam hal ini hasil kali dari proses

sebelumnya dikali 2, kemudian hasil perkalian pada proses kedua ditampilkan pada *user* pertama.



Gambar 4.6 DFD level 2 proses bagi Skenario pertama

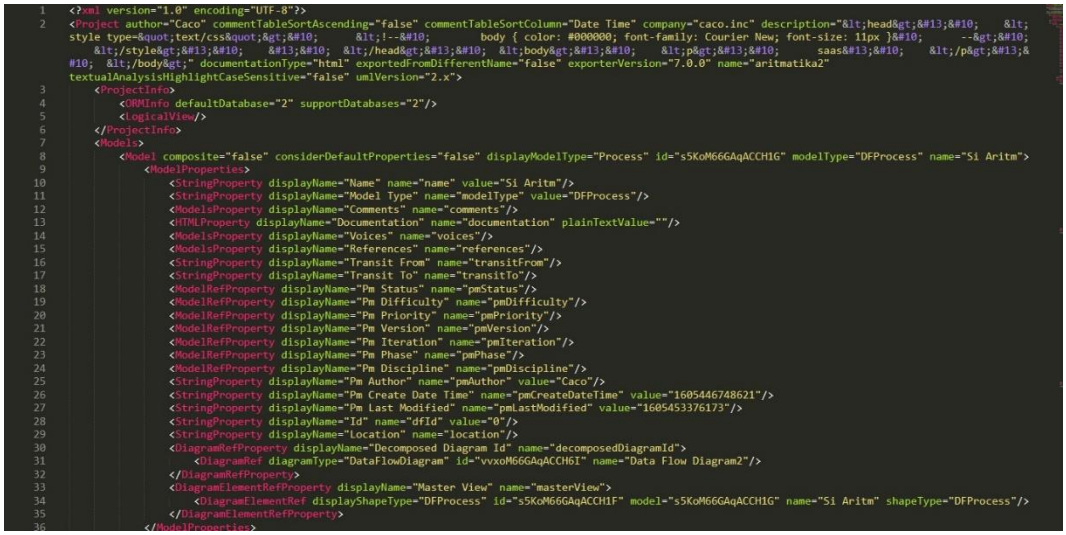


Gambar 4.7 DFD level 2 proses bagi skenario kedua

Pada diagram proses bagi di atas masing masing terdapat dua *external entity* dan satu proses bagi. Pada gambar 4.6 data diproses kemudian hasil dari proses bagi tersebut ditampilkan pada *user* pertama dan kedua. Sementara pada gambar 4.6 hasil dari proses bagi tersebut hanya ditampilkan pada *user* kedua tanpa dikembalikan / ditampilkan pada *user* pertama.

4.2. *Import, Parsing* dan *Preprocessing* DFD

Proses *import* dan *parsing* ini melibatkan file XML dari DFD. Tujuannya menganalisis setiap *syntac* yang benar kemudian dilampirkan ke *tag* yang menentukan setiap komponen. Gambaran atribut XML yang terdapat pada pemodelan DFD sebagaimana pada gambar 4.6



Gambar 4.8 Sebagian atribut XML pada pemodelan DFD aritmatik

Untuk proses *parsing* XML sendiri menggunakan PHP. Berikut merupakan kode sumber untuk melakukan *parsing*.

```
function parsing($id_dfd){
    $data = $this->get_dfd(null, null, $id_dfd);
    $url = $data[0]['data_dfd'][0]['url'];
    $file_name = basename($url);
    $url = $this->CI->mainlib->explode_uri($url);
    //Parsing
    $url = $this->CI->input->server('DOCUMENT_ROOT').'/'.$url[0].'/'.$url[1].'/'.$url[2].'/'.$file_name;
    $xml = simplexml_load_file($url);
    $xml = json_encode($xml);
    $xml = json_decode($xml);
    $level = 0;
    $end = false;
    $temp = $xml->Models->Model;
```

```
if($this->CI->mainlib->is_assoc($temp)){
            foreach($temp as $key=>$r){
              $r = (array) $r;
              if($r['@attributes']->modelType=='DFExternalEntity' or $r['@attributes']->modelType=='DFDataStore'
or $r['@attributes']->modelType=='DFProcess'){
                $queue;
                if($r['@attributes']->modelType=='DFProcess'){
                  foreach($r['ModelProperties']->StringProperty as $key2=>$v){
                    $v = (array) $v;
                    if($v['@attributes']->name=='dfId'){
                      $queue = $v['@attributes']->value;
                    }
                  }
                }
//attr
                $this->attr[count($this->attr)] = [
                  'id_dfd' => $id_dfd,
                  'id_attr' => $r['@attributes']->id,
                  'model_type' => $r['@attributes']->modelType,
                  'name_attr' => $r['@attributes']->name,
                  'dfd_level' => $level,
                  'data_attr' => serialize([
                              'FlowsOut' => (isset($r['FromSimpleRelationships']))? $r['FromSimpleRelationships'] :
null,
                              'FlowsIn' => (isset($r['ToSimpleRelationships']))? $r['ToSimpleRelationships'] : null
                              ]),
                  'queue' => ($r['@attributes']->modelType=='DFProcess')? $queue : null,
                  'ref' => null
                ];
//flow out
                if(isset($r['FromSimpleRelationships'])){
if($this->CI->mainlib->is_assoc($out)){
                    foreach($out as $key2=>$v){
                      $v = (array) $v;
$this->dtl_attr[count($this->dtl_attr)] = [
                        'id_dfd' => $id_dfd,
```

```
'id_attr1' => $v['@attributes']->from,
                                    'id_attr2' => $v['@attributes']->to,
                                    'flow' => $v['@attributes']->id,
                                    'vector' => 'OUT'
                                ];
                            }
                        }else{
                            $out = (array) $out;
                            $this->dtl_attr[count($this->dtl_attr)] = [
                                'id_dfd' => $id_dfd,
                                'id_attr1' => $out['@attributes']->from,
                                'id_attr2' => $out['@attributes']->to,
                                'flow' => $out['@attributes']->id,
                                'vector' => 'OUT'
                            ];
                        }
                    }
                     //flow in
                    if(isset($r['ToSimpleRelationships'])){
                        $in = $r['ToSimpleRelationships']->RelationshipRef;
                        if($this->CI->mainlib->is_assoc($in)){
                            foreach($in as $key2=>$v){
                                $v = (array) $v;
                                $this->dtl_attr[count($this->dtl_attr)] = [
                                    'id_dfd' => $id_dfd,
                                    'id_attr1' => $v['@attributes']->to,
                                    'id_attr2' => $v['@attributes']->from,
                                    'flow' => $v['@attributes']->id,
                                    'vector' => 'IN'
                                ]; }
}else{
                                $in = (array) $in;
                                $this->dtl_attr[count($this->dtl_attr)] = [
'id_dfd' => $id_dfd,
                                    'id_attr1' => $in['@attributes']->to,
```

```
'id_attr1' => $in['@attributes']->to,
                    'id_attr2' => $in['@attributes']->from,
                    'flow' => $in['@attributes']->id,
                    'vector' => 'IN'
                ];
            }
        }
```

Kode Sumber 4.1 Proses parsing XML Data Flow Diagram

Dalam file XML hasil *generate* DFD terdapat berbagai macam *tag* atribut, sehingga perlu untuk menganalisis mana *tag* yang diperlukan dan mana yang tidak. Fungsinya adalah untuk mempermudah dalam memproses program pada tahap selanjutnya. Sesuai kode PHP di atas *tag* atribut yang dibutuhkan adalah 'DFExternalEntity', 'DFDataStore' dan 'DFProcess'.

4.3. Proses *Mapping* DFD dengan WS

Pada tahap ini setelah proses *parsing* yaitu proses *mapping* ke *web service*. Tahap ini memetakan setiap proses alur datanya dengan proses pada *web service repository* yang sesuai dengan alur proses data tersebut. Sesuai dengan bahan uji sistem untuk memastikan apakah sistem sudah berjalan apa belum, maka dibuat *web service* aritmatika sesuai DFD yang yang dibuat sebelumnya. *Web service* yang dibuat sebagai ujicoba sistem yaitu *web service* proses tambah, *web service* proses kurang, *web service* proses kali dan *web service* proses bagi. Berikut merupakan

beberapa fungsi kode sumber yang menunjukan setiap proses dari *web service* aritmatik

4.3.1. *Web Service* Aritmatika Proses Tambah

```
function tambahtodb($user){
        $x = $this->CI->input->post('input_x');
        $y = $this->CI->input->post('input_y');
        $operation = '+';

        $hasil = $x+$y;

        $data = ['x'=>$x, 'y'=>$y, 'operation'=>$operation, 'hasil'=>$hasil];

        $this->db_tenant->insert('coba_arit', $data);

        return ['status'=>1];
    }

    function tambah10bydb($user){
        $data = $this->db_tenant->order_by('id_coba_arit', 'DESC')->get_where('coba_arit', array('operation'=>'+'))->row_array();

        $hasil = $data['hasil'] + 10;

        return ['result_tambah'=>$hasil];
    }
```

Kode Sumber 4.2 Web service aritmatika proses tambah

Kode sumber 4.2 merupakan kode sumber *web service* aritmatika proses tambah. Sesuai kode diatas bahwa terdapat dua fungsi proses tambah. Fungsi pertama yaitu penjumlahan antara variable x & y, sedangkan untuk fungsi kedua yaitu

penjumlahan hasil dari x, y dengan nilai 10, tanpa disimpan di *datastore*, kemudian dari fungsi kedua ditampilkan ke *user* pertama, sesuai proses *Data Flow Diagram* pada gambar 4.3. Untuk lebih jelasnya berikut detail *flow* level dua proses tambah

Gambar 4.9 Detail DFD level 2 artimatik proses tambah

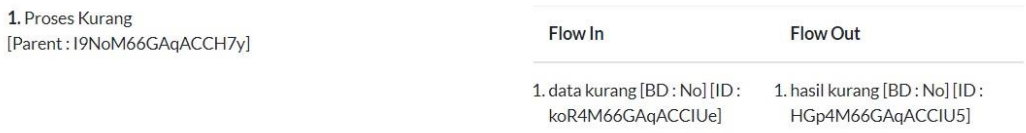
Dari gambar 4.9 dapat diambil kesimpulan bahwa terjadi dua aliran proses tambah. *Flow in* pada aliran proses pertama yaitu nilai *input* x + y, Sedangkan *flow out* pada aliran proses pertama merupakan hasil x +y. Sementara itu nilai *flow in* pada proses kedua itu sama dengan *flow out* pada aliran proses pertama. Sedangkan untuk nilai *flow out* pada aliran proses kedua merupakan hasil dari *flow out* pada proses pertama ditambah 10.

4.3.2. *Web Service* Aritmatika Proses Kurang

```
function kurang($user){
    $x = $this->CI->input->post('input_x');
    $y = $this->CI->input->post('input_y');

    $sum = $x - $y;

    return ['result_kurang'=>$sum];
```

Kode Sumber 4.3 Web service aritmatika proses kurang

Pada *web service* aritmatik proses kurang, sesuai DFD level 2 pada gambar 4.4 hasil langsung ditampilkan ke *user* 1, tanpa ada proses lain yang terlibat. sesuai detail *flow* gambar 4.10



Gambar 4.10 Detail DFD level 2 artimatik proses kurang

Dari gambar gambar 4.10 dapat diambil kesimpulan bahwa hanya terjadi satu aliran proses pengurangan. *Flow in* pada proses ini adalah x & y dan untuk *flow out* merupakan hasil dari x-y. Untuk *web service* selanjutnya yaitu proses kali. Berikut merupakan kode sumber untuk *web service* proses kali.

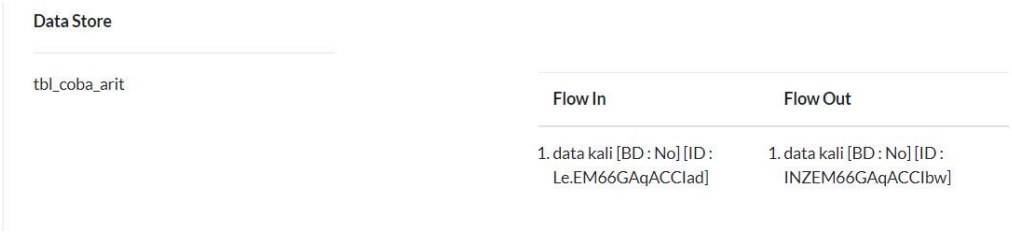
4.3.3. *Web Service* Aritmatika Proses Kali

```
function kalitodb($user){
        $x = $this->CI->input->post('input_x');
        $y = $this->CI->input->post('input_y');
        $operation = 'x';

        $hasil = $x*$y;

        $data = ['x'=>$x, 'y'=>$y, 'operation'=>$operation, 'hasil'=>$hasil];
        $this->db_tenant->insert('coba_arit', $data);

        return ['status'=>1];
    }

    function kali2bydb($user){
        $data = $this->db_tenant->order_by('id_coba_arit', 'DESC')->get_where('coba_arit', array('operation'=>'x'))->row_array();

        $hasil = $data['hasil'] * 2;

        return ['result_kali'=>$hasil];
    }   }
```

Kode Sumber 4.4 Web service aritmatika proses kali

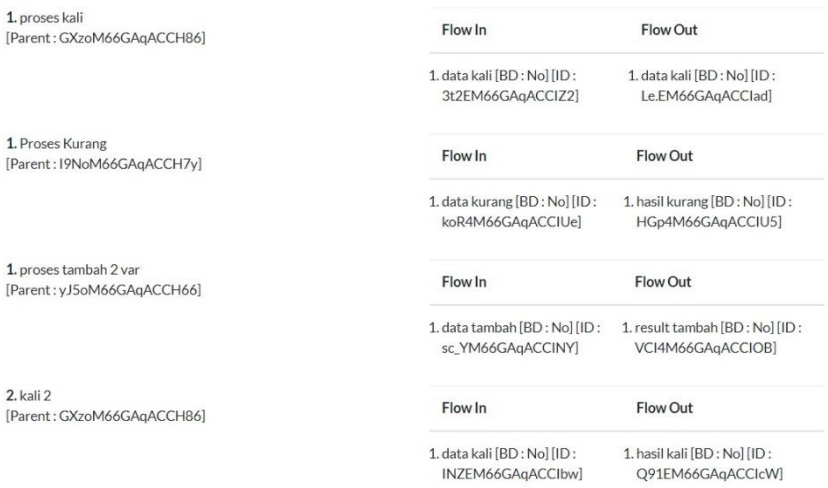
Pada *web service* aritmatika proses kali terdapat dua fungsi kali. Hal ini mengacu pada DFD level 2 pada gambar 4.5. *Flow in* pada proses pertama merupakan nilai *input* antara perkalian x & y. Kemudian data hasil kali disimpan di *data store*. Nilai *flow out* pada proses pertama bernilai sama dengan *flow in* yang masuk ke *data*

store. Hal ini dibuktikan dari gambar 4.11 yang merupakan hasil *generate* DFD arimatik.



Gambar 4.11 Detail DFD level 2 artimatik proses kali

Seperti gambar 4.11 bahwa proses kali pada *web service* melibatkan *data store* pada DFD level kedua. *Flow in* yang masuk ke *data store* merupakan *flow out* dari proses pertama. Setelah data disimpan di *data store* yang merupakan hasil kali x & y, kemudian diteruskan ke proses kedua. Dalam hal ini, *flow out* dari *data store* bernilai sama dengan *flow in* yang masuk ke proses kedua. Pada proses kedua, hasil perkalian pada proses pertama dikali dua, kemudian *flow out* pada proses kedua ditampilkan/dikembalikan pada *user* pertama. Untuk detail *flow* pada aritmatik proses kali dapat dilihat pada gambar berikut.



Gambar 4.12 Detail DFD level 2 artimatik proses kali

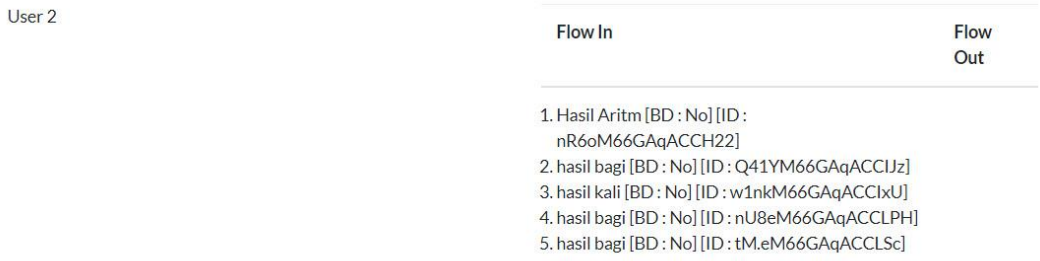
4.3.4. *Web Service* Aritmatika Proses Bagi

Pada uji aplikasi *web service* aritmatika proses bagi ini, terdapat dua skenario yang dibuat sesuai pada gambar 4.6 dan 4.7. Berikut merupakan kode sumber *web service* aritmatika proses bagi.

```
function bagi2user($user, $data){
    if($user=='User 1'){
        $x = $this->CI->input->post('input_x');
        $y = $this->CI->input->post('input_y');

        $hasil = $x/$y;
```

Kode Sumber 4.5 Web service aritmatika proses bagi

Pada proses ini terdapat dua skenario yang dibuat. Skenario pertama dengan menampilkan hasil bagi ke kedua *user* (*user* pertama dan *user* kedua), dan skenario kedua hanya menampilkannya ke *user* kedua. Pada skenario pertama *flow out* pada proses bagi bernilai sama dengan *flow in* pada *user* pertama dan kedua. Hal ini dibuktikan dengan gambar detail *flow* berikut.

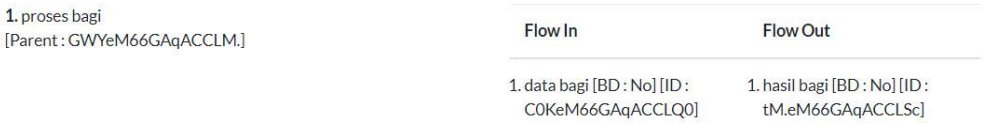


Gambar 4.13 Detail flow user 2

Gambar 4.14 Detail flow user 1

Gambar 4.15 Detail flow proses bagi (skenario 1)

Terbukti sesuai detail *flow* pada skenario pertama diatas, bahwa hasil *flow out* pada proses pertama sama dengan *flow in* pada *user* kedua dan *flow in user* pertama. Untuk selanjutnya yaitu skenario kedua. Skenario ini hampir sama dengan skenario pertama. Bedanya yaitu hasil dari proses bagi tidak ditampilkan atau dikembalikan ke *user* pertama, namun hanya di tampilkan ke *user* kedua. Berikut merupakan detail *flow* proses bagi skenario kedua.



Gambar 4.16 Detail low proses bagi (skenario 2)

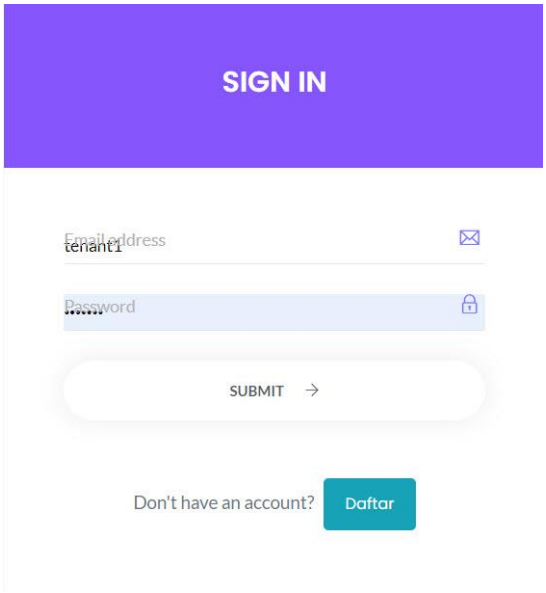
Sesuai detail *flow* pada gambar 4.14 bahwa *flow out* pada proses bagi x & y sama dengan *flow in* pada *user* kedua. Hal ini ditandai dengan ID yang sama pada kedua *flow* tersebut. Maksudnya (id : tM.eM66GAqACCLSc) terdapat pada *user* kedua dan proses bagi pada skenario kedua juga tidak terdapat pada *user* pertama, artinya hasil bagi tersebut tidak dikembalikan pada *user* pertama, sesuai DFD pada gambar 4.5.2.

4.4. Tampilan Aplikasi

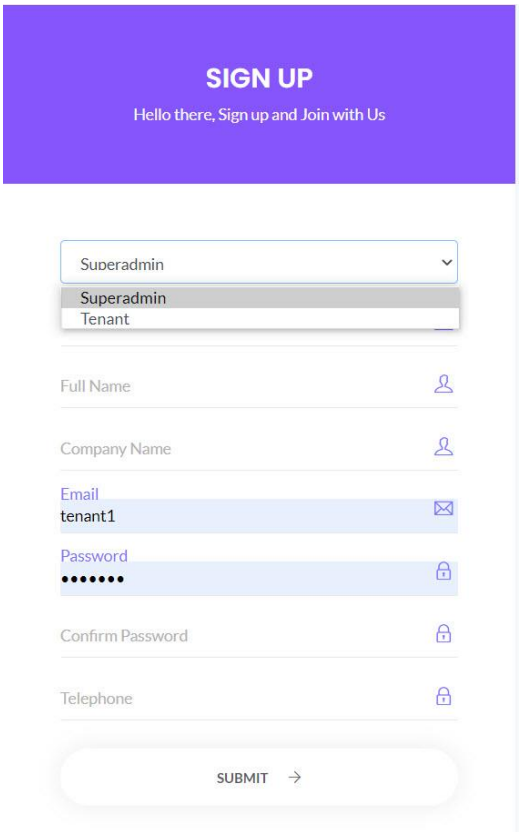
Gambaran umum aplikasi ini terdapat dua level akses yaitu *superadmin* dan penyewa (*tenant*) dan untuk level akses *workflow* terdapat dua yaitu *user* pertama dan *user* kedua. Level *tenant* untuk dapat menggunakan hak aksesnya harus mendapat persetujuan atau aktivasi dari *superadmin*. Tampilan aplikasi setiap level memiliki *interface* dan fungsi yang berbeda. Berikut merupakan penjelasan setiap tampilan aplikasi mulai dari *login, registration, list of tenant, list of web service, list application, list* DFD.

4.4.1. Halaman Login dan Registrasi

Halaman ini berfungsi agar *tenant* dan admin dapat masuk dan mengakses akun setelah dilakukan validasi dengan mencocokan *username* dan *password* yang telah dibuat saat mendaftar pada halaman registrasi. Berikut merupakan tampilan untuk Halaman *login* dan registrasi.



Gambar 4.17 Tampilan login ke aplikasi

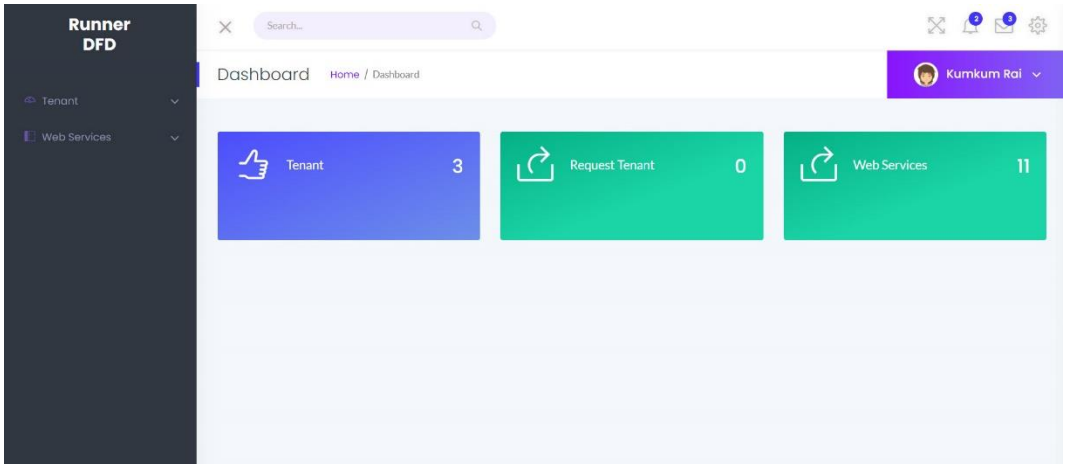


Gambar 4.18 Tampilan registrasi

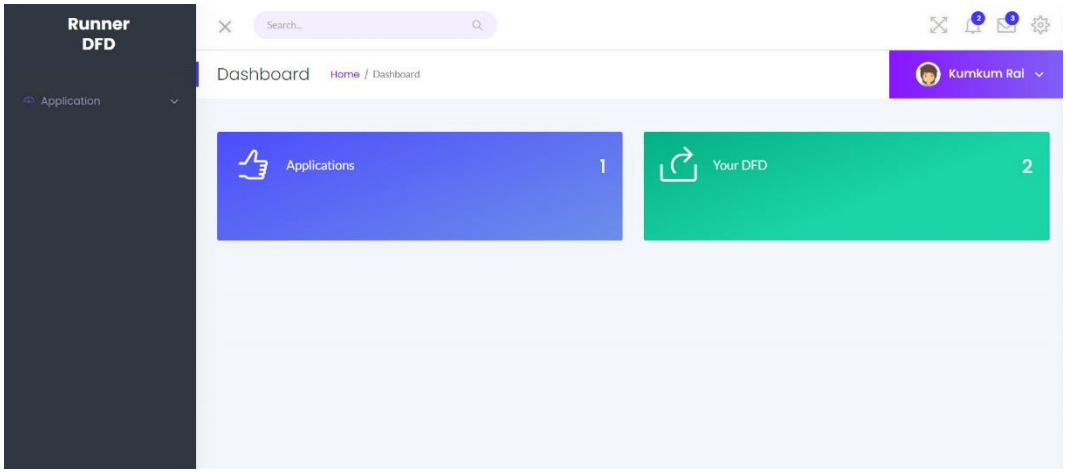
Sesuai Gambar 4.18 bahwa level dalam aplikasi ini terdapat dua level akses *superadmin* dan *tenant.*

4.4.2. Halaman *Dasboard*

Halaman ini merupakan menu utama sebelum masuk ke menu yang akan diakses. Untuk Halaman dashboard ini terdapat dua *interface*, yaitu *interface* untuk level *superadmin* dan *interface* untuk level *tenant*. Berikut merupakan tampilan *dasboard* untuk level keduanya.



Gambar 4.19 Dasboard superadmin

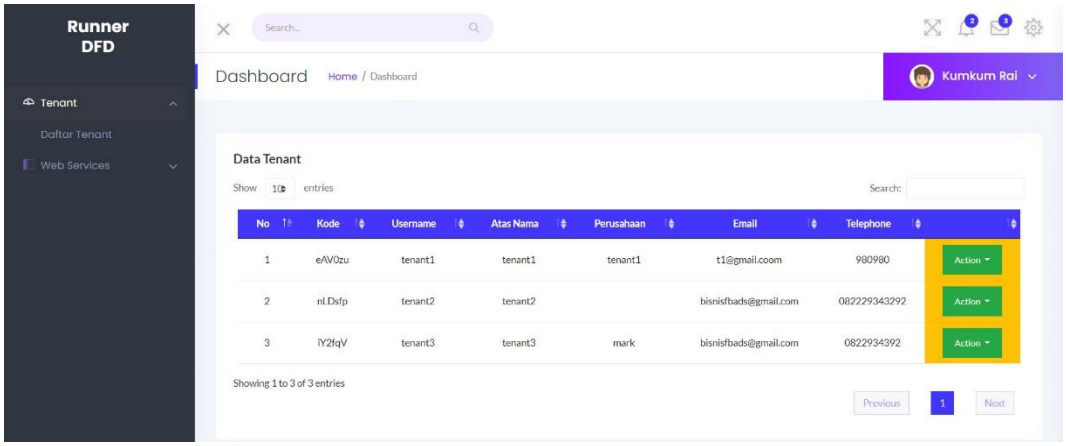


Gambar 4.20 Dasboard tenant

Sesuai kedua gambar 4.19 dan 4.20 bahwa level *superadmin* terdapat tiga *body content* yaitu jumlah *tenant* yang terdaftar, jumlah *request tenant* dan jumlah *web service* yang terdaftar. Untuk *list navigasi* terdapat dua daftar pilihan yaitu daftar *tenant* dan *web service*. Sementara untuk level *tenant* pada *interface dasboard* terdapat dua *body content* yaitu jumlah aplikasi dan jumlah DFD. Sementara untuk bagian navigasinya level *tenant* memiliki satu list aplikasi yang memuat beberapa daftar aplikasi yang dibuat nantinya.

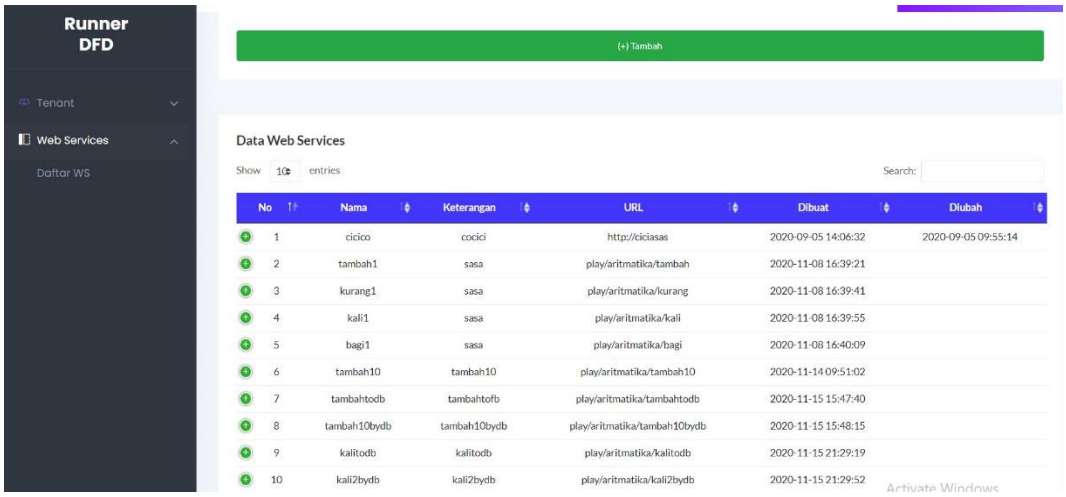
4.4.3. Tampilan Menu Navigasi

Dalam aplikasi ini baik level *superadmin* dan *tenant* memiliki tampilan menu navigasi yang berbeda. Sesuai penjelasan sebelumnya bahwa untuk level *superadmin* memiliki dua menu navigasi dan satu menu navigasi untuk level *tenant.* Berikut *interface* menu navigasi *superadmin* yaitu menu daftar *tenant* dan menu daftar *web service*.



Gambar 4.21 Menu navigasi daftar tenant

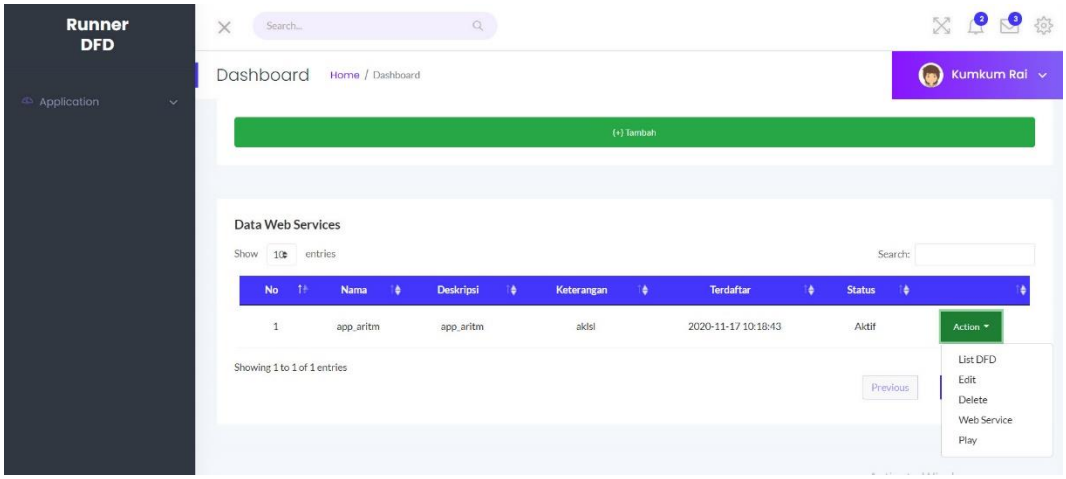
Setelah *tenant* melakukan proses registrasi, maka data masuk ke daftar *tenant. Superadmin* memiliki kontrol terhadap *tenant* yang sudah mendaftar. Ketika tenant yang sudah mendaftar belum mendapat persetujuan dari *superadmin*, maka *tenant* tidak dapat melakukan proses login dan melakukan akses ke aplikasi.



Gambar 4.22 Menu navigasi daftar web service

Daftar *web service* pada gambar 4.22 merupakan *web service* yang nanti akan dipetakan pada setiap alur proses yang dibuat. Dalam kasus ini proses yang dibuat

sebagai uji aplikasi adalah proses aritmatik. Pada kolom URL, penamaan haruslah sama, sesuai dengan kelas-kelas dan fungsi yang dibuat agar tidak terjadi kesalahan dalam pemanggilan fungsi pada pemetaan *web service* ke alur proses aplikasi yang terdapat pada level tenat. Berikut merupakan *interface* navigasi daftar aplikasi dari level *tenant.*

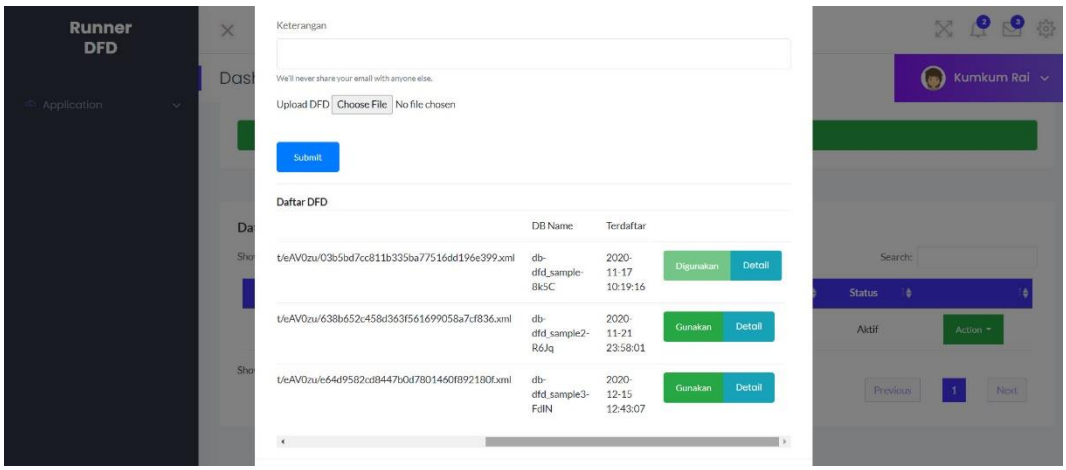


Gambar 4.23 Menu navigasi daftar aplikasi

Pada menu *interface* diatas tenant dapat menambahkan daftar aplikasi yang nantinya dipetakan dengan *web service* yang telah dibuat. Selain itu *tenant* dapat menambahkan list DFD pada aplikasi, artinya dalam satu aplikasi nantinya memuat beberapa DFD. File DFD yang diunggah dalam aplikasi merupakan file XML yang merupakan hasil *generate* DFD, termasuk juga *database* di dalamnya.

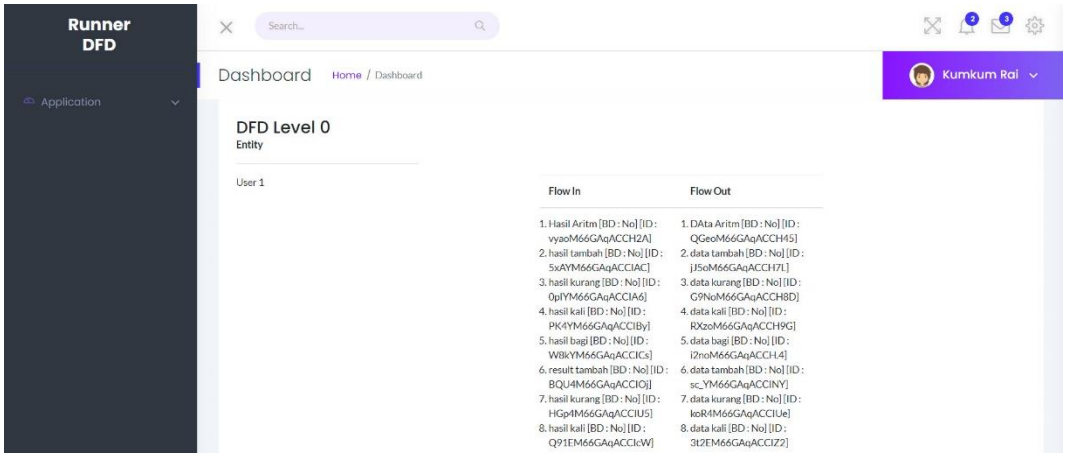
4.4.4. *Interface List* DFD Dan Detail *Flow*

Action button pada gambar 4.23 memiliki bebrapa opsi di antaranya, *list* DFD, *edit delete*, opsi *web service*, dan *play*. Opsi menu list DFD digunakan untuk menambahkan DFD dan menampilkan detail *flow* yang terjadi. Opsi *edit delete* digunakan untuk mengedit dan menghapus DFD yang sudah dibuat. Opsi *web service* digunakan ntuk memetakan *web service* dengan setiap alur proses DFD dan opsi *play* digunakan untuk uji dan *running web service*. Berikut merupakan beberapa tampilan *interface* pada menu opsi *action button*.



Gambar 4.24 Opsi menu action button list DFD

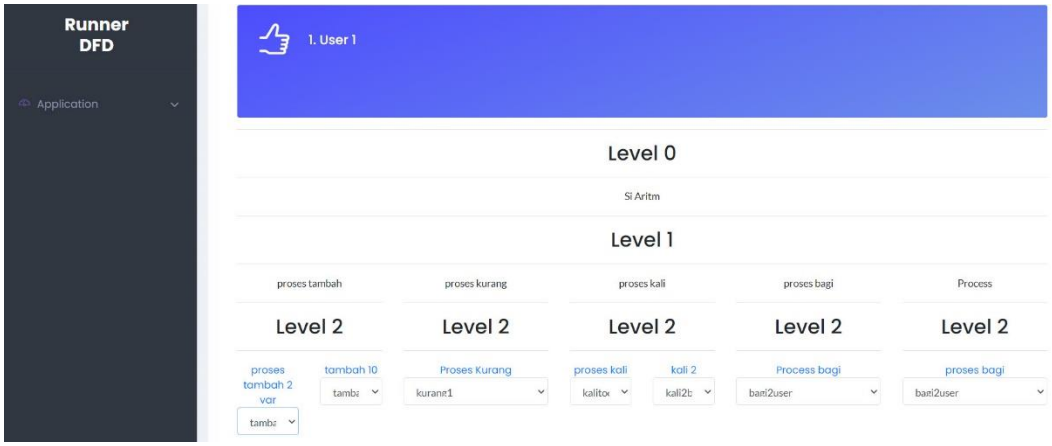
Form pada gambar 4.24 digunakan untuk menambahkan DFD. *File* yang diunggah merupakan file XML hasil *generate* DFD. Terdapat pula daftar DFD yang sudah diunggah beserta *database*. Terdapat pula dua opsi *button*. Untuk *green button* digunakan untuk memilih dan menjalankan DFD yang akan digunakan pada proses *web service*. Sementara untuk detail *button* digunakan untuk menampilkan detail *flow* sesuai file XML hasil *generate* DFD. Untuk *interface* detail DFD yang menunjukan detail *flow* adalah sebagai berikut



Gambar 4.25 Interface detail DFD

4.4.5. *Interface Mapping* DFD dengan *Web Service*

Proses *mapping* ini memetakan setiap alur proses datanya dengan *web service repository*. Sebagai bahan uji aplikasi nanti, proses yang dipetakan adalah proses aritmatik sesuai DFD yang dibuat sebelumnya. Proses pemetaan seperti ditampilkan pada gambar 4.26.



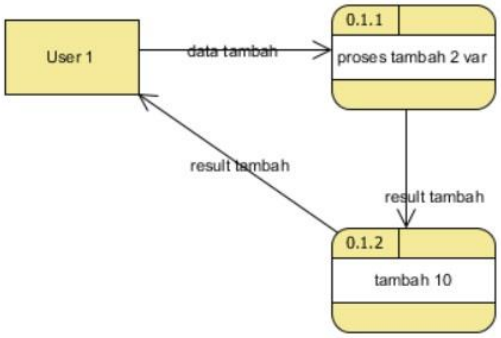
Gambar 4.26 Interface pemetaan web service

Sesuai gambar 4.26, pemetaan dilakaukan pada DFD level 2. Pada proses aritmatika tambah terjadi dua kali proses. Proses pertama x ditambah y, dan proses keduanya

hasil tambah x dan y ditambah 10, sesuai detail *flow* pada gambar 4.9. Pada artimatika proses kurang terjadi satu kali proses (x-y). Untuk aritmatika proses kali terjadi dua kali proses, proses pertama perkalian x dengan y, dan proses kedua merupakan hasil perkalian pertama dikali dua. Pada aritmatika bagi terjadi satu proses bagi, namun memiliki nilai kembali yang berbeda.

4.5. Pengujian Aplikasi

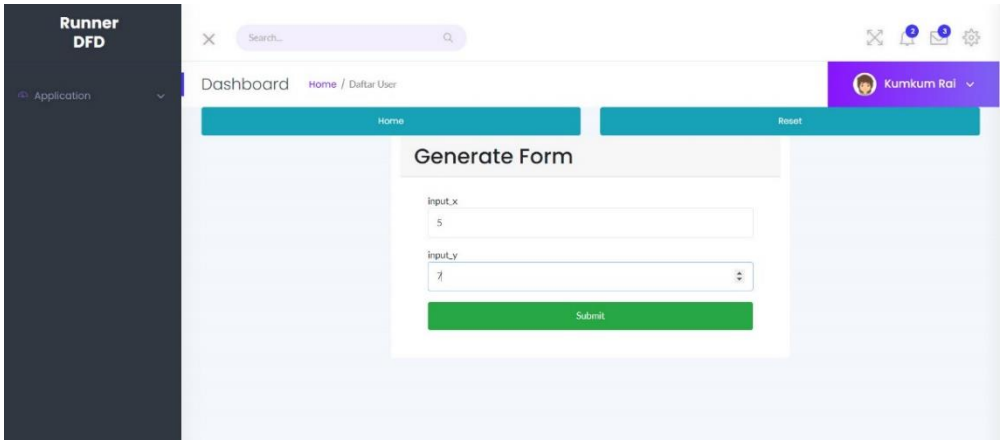
Pada analisa hasil dan pengujian aplikasi ini melibatkan proses artimatik sesuai DFD yang dibuat. Sebagai acuan, DFD yang dijadikan sebagai uji aplikasi adalah DFD level 2, yang merupakan hasil decompose pada DFD level pertama pada gambar 4.2. Ada beberapa asumsi yang dijadiakan sebagai proses uji. Asumsi pertama yaitu jika proses yang terjadi terdapat satu *entity / user* dan dua proses. Hasil proses keduanya dikembalikan hanya kepada *user* pertama. Asumsi atau skenario pertama ini seperti gambar 4.27.



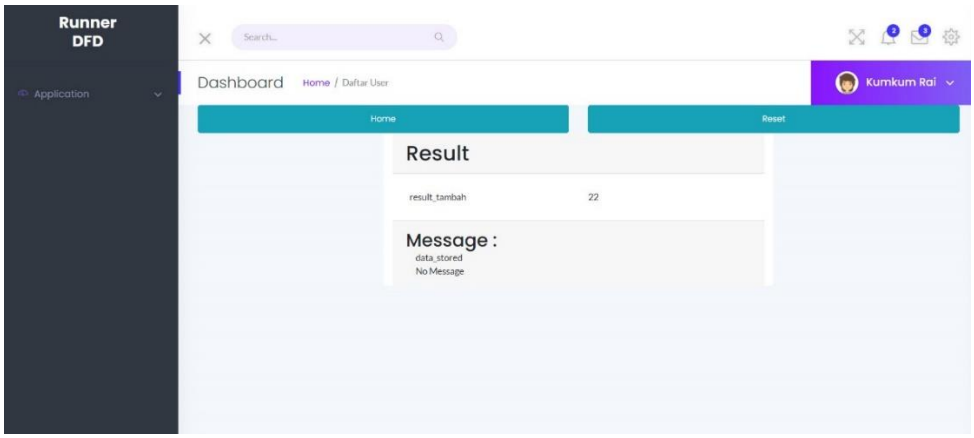
Gambar 4.27 Skenario pertama

Pada skenario gambar 4.27, terdapat satu aktor atau *exteral entity* dan dua diagram proses. Aktor yang berperan dalam proses ini adalah *user* pertama. Aktor

memberikan nilai *input* pada proses pertama. Pada diagram proses pertama terdapat *rule* proses penambahan dua variable yang sudah dimasukan oleh aktor atau *user* pertama. Hasil dari proses pertama pada diagram proses pertama, diteruskan ke proses kedua pada diagram proses kedua. Proses pada diagram kedua yaitu menambahkan hasil dari proses pertama dengan angka sepuluh. Nilai dari proses kedua ini kemudian dikembalikan ke aktor atau *user* pertama. Tampilan uji seperti pada gambar 4.28 dan 4.29



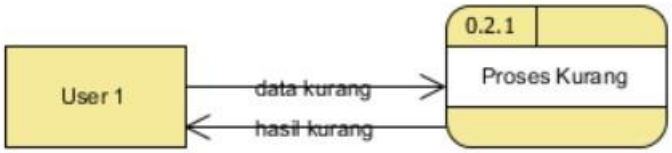
Gambar 4.28 Input proses pertama (tambah)



Gambar 4.29 Hasil proses kedua (tambah)

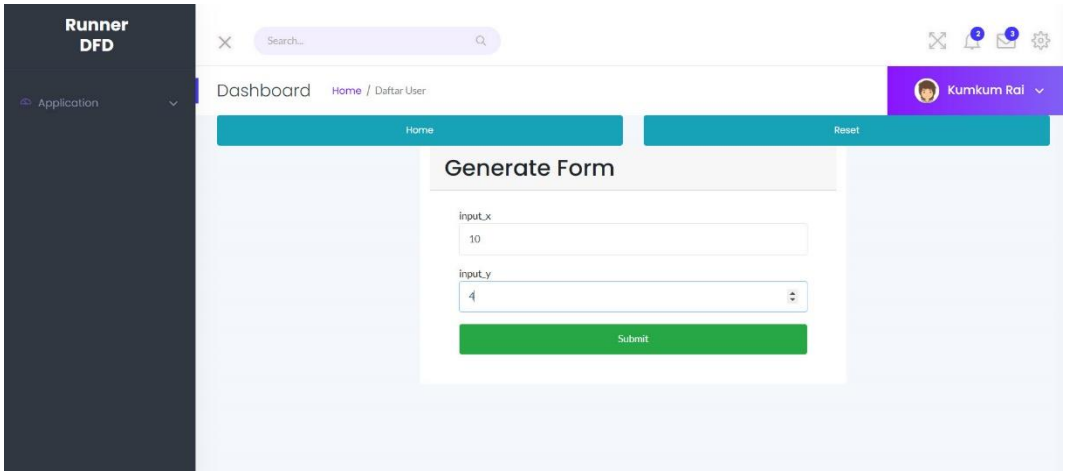
Pada Gambar 4.28 nilai x dan y merupakan nilai *flow in* yang masuk pada proses pertama, kemudaian hasil pada proses diterukan ke proses kedua proses kedua ini menjumlahkan nilai pada proses pertama dengan nilai 10 maka sesuai *result* nilai yang tampil adalah 22. Kemudain nilai itu dikembalikan dan ditampilkan ke *user* pertama sesuai DFD pada gambar 4.27 dan untuk detail *flow* sesuai pada gambar 4.9.

Skenario kedua yaitu jika proses yang terjadi terdapat satu diagram *external entity / user* dan satu diagram proses dan untuk hasilnya dikembalikan ke *user* pertama. Skenario kedua ini digunakan pada proses pengurangan, sesuai gambar 4.30.

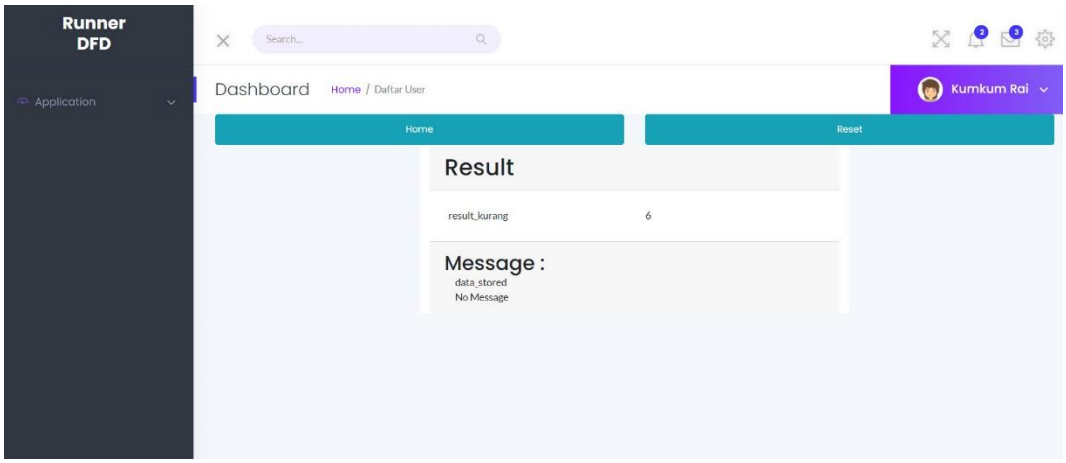


Gambar 4.30 Skenario kedua

Pada skenario gambar 4.30, terdapat satu aktor atau *exteral entity* dan dua diagram proses. Aktor yang berperan dalam proses ini adalah *user* pertama. Aktor memberikan nilai *input* pada proses pertama. Pada diagram proses pertama terdapat *rule* proses pengurangan dua variable yang sudah dimasukan oleh aktor atau *user* pertama. Nilai dari proses ini kemudian dikembalikan ke aktor atau *user* pertama, tanpa melalui proses selanjutnya. Tampilan uji seperti pada gambar 4.28 dan 4.29



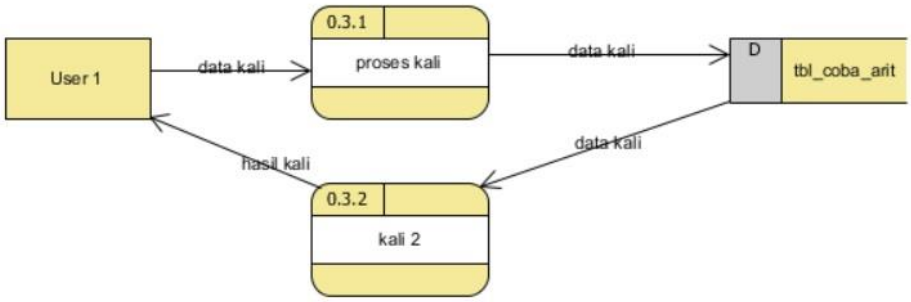
Gambar 4.31 Input proses (pengurangan)



Gambar 4.32 Hasil proses pengurangan

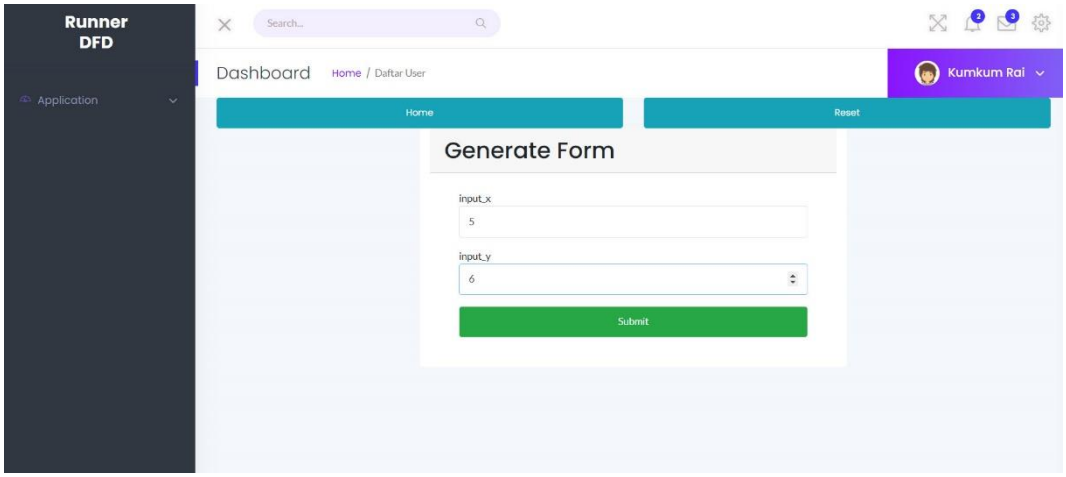
Pada gambar 4.31 nilai x dan y merupakan nilai *flow in* yang masuk pada proses. Untuk variable x bernilai 10 dan y bernilai 4. Hasil dari proses menunjukan angka 6. Hasil nilai itu dikembalikan dan ditampilkan ke *user* pertama sesuai skenario DFD pada gambar 4.30.

Skenario ketiga yaitu jika proses yang terjadi terdapat satu diagram *external entity /user*, dua diagram proses dan satu diagram *data store* serta hasil dikembalikan ke *user* pertama. Skenario ini sesuai pada diagram DFD gambar 4.33.

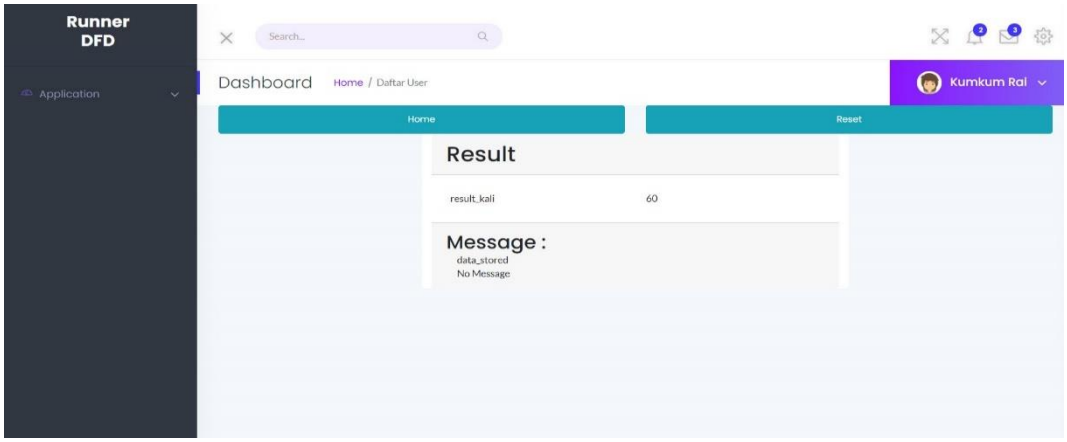


Gambar 4.33 Skenario ketiga

Pada skenario gambar 4.33, terdapat satu diagram *exteral entity*, dua diagram proses dan satu diagram *data store*. Aktor yang berperan dalam proses ini adalah *user* pertama. *User* memberikan nilai *input* pada proses pertama. Pada diagram proses pertama terdapat *rule* proses perkalian dua variable yang sudah dimasukan oleh aktor atau *user* pertama. Hasil dari proses pertama ini kemudian diteruskan pada *data store*. Setelah dari diagram *data store,* nilai hasil perkalian pada proes pertama diteruskan ke proses kedua. Pada proses kedua terjadi perkalian hasil proses pertama dengan angka dua. Hasil dari proses kedua ini kemudian dikembalikan ke *user* pertama. Tampilan uji seperti pada gambar 4.34 dan 4.35.



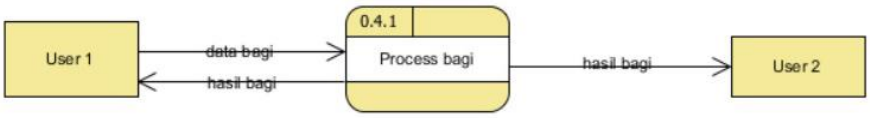
Gambar 4.34 Input proses pertama (perkalian)



Gambar 4.35 Hasil proses kedua (perkalian)

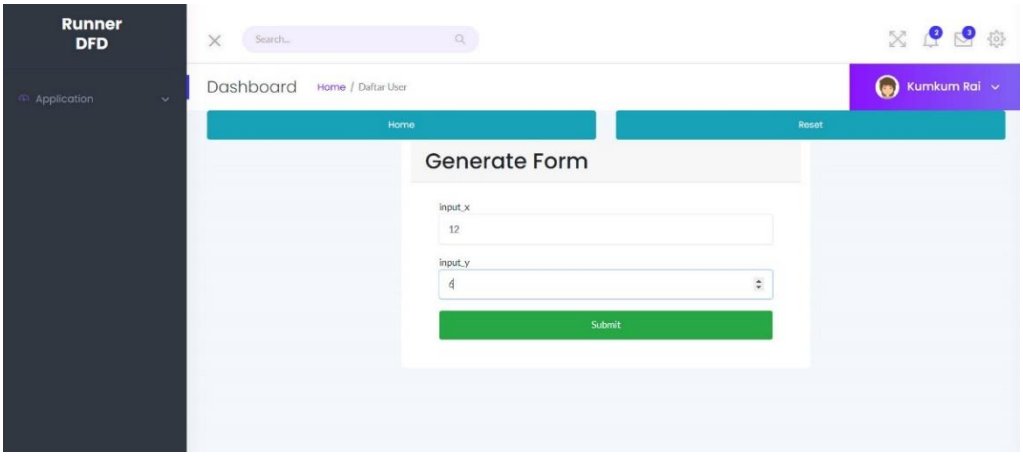
Pada gambar 4.34 nilai x dan y merupakan nilai *flow in* yang masuk pada proses pertama. Sesuai nilai *input* x bernilai 5 dan y bernilai 6. Hasil pada proses pertama disimpan ke *data store*. Data dari *data store* yang bernilai 30, kemudian diteruskan ke proses kedua. Pada proses kedua ini terjadi proses perkalian antaran *flow out* dari *data store* yang bernilai 30 dengan 2. Sesuai hasil pada gambar 4.35 menunjukan nilai 60. Nilai ini ditampilakan / dikembalikan ke *user* pertama sesuai skenario DFD pada gambar 4.33.

Skenario keempat terdapat dua asumsi yang dibuat. Asumsi pertama dengan menampilkan hasil bagi ke kedua *user* dan asusmsi kedua hanya menampilkannya ke *user* kedua. Dalam skenario keempat ini terdapat dua diagram *external* e*ntity* dan satu diagram proses. Skenario keempat ini sesuai pada gambar 4.36.

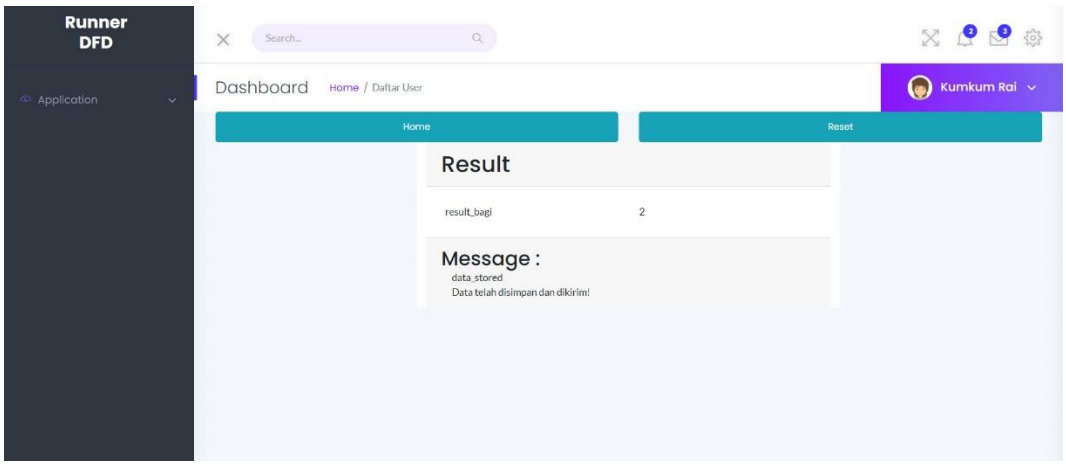


Gambar 4.36 Skenario keempat

Pada skenario gambar 4.36, terdapat dua *exteral entity* dan satu diagram proses. Aktor yang berperan dalam proses ini adalah *user* pertama. Aktor memberikan nilai *input* pada proses pertama. Pada diagram proses pertama terdapat *rule* proses pembagian dua variable yang sudah dimasukan oleh aktor atau *user* pertama. Hasil dari diagram proses dikembalikan pada kedua user (pertama dan kedua). Hasil kembalian pada *user* pertama berupa hasil pembagian pada diagram proses, sedangkan hasil kembalian pada *user* kedua berupa notifikasi hasil pada diagram proses. Tampilan uji seperti pada gambar 4.28 dan 4.29



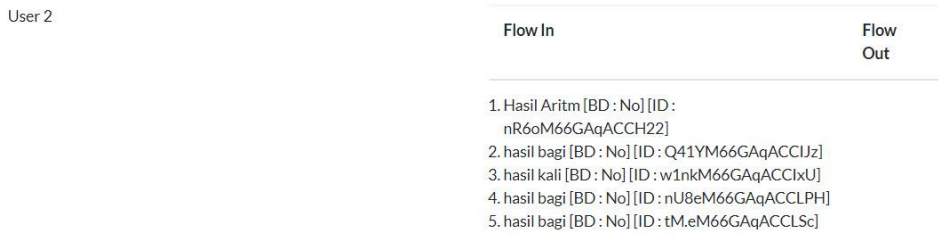
Gambar 4.37 Input bagi



Gambar 4.38 Hasil proses bagi

Pada gambar 4.37 nilai x dan y merupakan nilai *flow in* yang masuk pada proses. Untuk variable x bernilai 12 dan y bernilai 6. Hasil dari proses menunjukan angka 2. Hasil nilai itu dikembalikan dan ditampilkan ke *user* pertama dan dikembalikan ke *user* kedua sebagai notifikasi sesuai skenario keempat DFD pada gambar 4.36. Hal ini dibuktikan sesuai detail flow pada gambar 4.39, 4.40 dan 4. 41.

Gambar 4.39 Detail flow user 1



Gambar 4.40 Detail flow user 2

Gambar 4.41 Detail flow proses

Terbukti sesuai detail *flow* pada gambar 4.39, 4.40 dan 4.41, bahwa hasil *flow out* pada proses pertama sama dengan *flow in* pada *user* pertama dan *flow in* pada *user* kedua yang artinya nilai tersebut di kembalikan ke kedua *user*.

4.6. Analisis Hasil

Dari beberapa pengujian DFMS sesuai skenario DFD yang telah dibuat, diperoleh hasil analisa yang membuktikan setiap skenario DFD dapat dipetakan pada web service.

Tabel 4 1. Hasil analisis sederhana skenario DFD

	x	Operation	y	operation	additional value	Hasil
Skenario 1	**5**	+	**7**	+	**10**	**22**
Skenario 2	**10**	-	**4**	-	-	**6**
Skenario 3	**5**	x	**6**	x	**2**	**60**
Skenario 4	**12**	:	**6**	:	-	**2**

a. **Skenario 1** : Pada nilai x & y merupakan nilai *flow in* yang masuk pada proses pertama, kemudaian hasil pada proses diterukan ke proses kedua proses kedua ini menjumlahkan nilai pada proses pertama dengan nilai 10 maka sesuai result nilai yang tampil adalah 22.
b. **Skenario 2** : nilai x & y merupakan nilai *flow in* yang masuk pada proses . Untuk variable x bernilai 10 dan y bernilai 4 . Hasil dari proses menunjukan angka 6 . Kemudain nilai itu dikembalikan dan ditampilkan ke *user* pertama
c. **Skenario 3** : nilai x & y merupakan nilai *flow in* yang masuk pada proses pertama. Sesuai nilai input x bernilai 5 dan y bernilai 6. Kemudian hasil pada proses pertama disimpan ke data store. Data dari data store bernilai yang bernilai 30 , kemudian diteruskan ke proses kedua. Pada proses kedua ini terjadi proses perkalian antaran flow out dari data store yang bernilai 30 dengan 2. Sesuai hasil akhir pada tabel 4.1 menunjukan nilai 60.
d. **Skenario 4** : Pada skenario 4 nilai x & y merupakan nilai *flow in* yang masuk pada proses . Untuk variable x bernilai 12 dan y bernilai 6 . Hasil dari proses menunjukan angka 2 . Kemudain nilai itu ditampilkan ke *user* pertama dan user kedua.

4.7. Novelty Dan Research Gap

Sudah banyak penelitian yang mengkaji mengenai konfigurasi *management system* atau hal-hal yang memiliki keterkaitan tentang hal tersebut. Salah satunya adalah BPMS. BPMS (*Bussiness Process Management System*) merupakan sebuah platform untuk mengumpulkan, mengatur, menganalisis, mengoptimasi, dan meningkatkan proses bisnis. Banyak vendor BPMS yang digunakan sebagai manajemen proses bisnis, di antaranya Bonita BPM, Bizagi Arabdox, Bonitad dan

Joget. Berbeda dengan BPMS yang befokus pada aliran proses, DFMS sebagai *management sistem* yang tergolong baru berfokus pada aliran data. Aliran data ini didasarkan pada *Data Flow Diagram*.

DFMS (*Data Flow Mangement System*) merupakan sebuah software digunakan untuk mengolah dan memanajement file *Data Flow Diagram* (DFD). Sebelum file DFD diolah pada sistem DFMS, file DFD diexport dalam bentuk XML. Pada file XML ini, informasi yang berkaitan dengan DFD dapat ditransfer secara bersamaan menggunakan metadata melalui *web service*. Pada Sistem DFMS ini terdapat dua *database*. Pertama yaitu *database* untuk ranah aplikasi DFMS, seperti *tenant* (penyewa), *web service*, *user* dan aplikasi. Kedua yaitu *database* untuk ranah DFD, seperti proses, *external entity* dan *data store*. Jadi inti dari novelty ini yaitu menemukan sebuah software *management system* sederhana yang berfokus pada aliran data yang didasarkan pada DFD. Sementara itu, titik riset dalam penelitian ini yaitu menjadikan file DFD menjadi *executable* (dapat dieksekusi) dan mengimplementasikan hasil generate DFD yang tersimpan dalam format XML pada *web service*.

4.8. Integrasi Perencanaan Strategis dalam Al-Quran

Penelitian yang dijalankan memberikan *output* bahwa file DFD dapat dieksekusi (executable). Proses pengeksekusian ini terdapat pada *Data Flow Management System* (DFMS). Sistem DFMS ini terinspirasi dari BPMS. Proses eksekusi pada DFMS ini diharapkan mampu menjadi acuan dalam proses eksekusi file DFD yang sifatnya lebih kompleks dan fleksible, sehingga mempermudah dalam mengetahui aliran data pada sebuah sistem informasi. Allah SWT berfirman dalam QS. Al-Insyirah ayat 5-8 bahwa sesulit apapun tingkat kerumitan dalam

suatu urusan, pasti akan ada suatu jalan keluar agar suatu urusan tersebut menjadi mudah. Hal ini dikarenakan Allah SWT tidak menyertakan suatu kesulitan melainkan juga sudah meyiapkan solusi dan jalan keluar untuk menyelessaikannya.

فَإِنَّ مَعَ الْعُسْرِ يُسْرًا ﴿٥﴾ إِنَّ مَعَ الْعُسْرِ يُسْرًا ﴿٦﴾ فَإِذَا فَرَغْتَ فَانْصَبْ ﴿٧﴾ وَإِلَىٰ رَبِّكَ فَارْغَبْ ﴿٨﴾

Artinya: “Karena sesungguhnya sesudah kesulitan itu ada kemudahan, sesungguhnya sesudah kesulitan itu ada kemudahan. Maka apabila kamu telah selesai (dari sesuatu urusan), kerjakanlah dengan sungguh-sungguh (urusan) yang lain, dan hanya kepada Tuhanmulah hendaknya kamu berharap.

Syaikh Abu Bakr Jabir Al-Jazairi berkata, “Bersama kesulitan itu ada kemudahan seterusnya dan selamanya. Satu kesulitan tidak mungkin mengalahkan dua kemudahan. Harapan seorang mungkin tentunya ingin bahagia selamanya.” (Aysar At-Tafasir, hlm. 1482) [11]

Ayat tersebut telah memotifasi peneliti untuk mencari solusi agar file DFD dapat dieksekusi dalam sistem DFMS. Solusi solusi ini digunakan untuk mempermudah mengetahi aliran aliran data pada DFD, mulai skema DFD yang sederhana sampai yang kompleks, dengan mengambil setiap id proses diagram tersebut. Sesuai dengan yang Allah SWT firmankan, bahwa sungguh pada setiap kesulitan ada solusi untuk mempermudah urusan tersebut. Hanya kepada Allah SWT, kita berharap segala urusan kita akan dipermudah

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan, dapat disimpulkan bahwa

1. Dalam ujicoba yang dilakukan, terdapat empat asumsi skenario. Skenario pertama apabila dalam suatu DFD sistem terdapat satu *external entity* dan proses. Hasil kembalian dari proses ditampilkan ke *user* pertama. Skenario kedua apabila dalam suatu DFD sistem terdapat satu *external entity* dan satu proses. Hasil kembalian dari proses langsung ditampilkan ke *user*. Skenario ketiga apabila dalam suatu DFD sistem terdapat satu *external entity*, dua proses, dan satu *data store*. Hasil kembalian dari proses pertama disimpan pada *data store*. Data dari *data store* diteruskan ke proses kedua, kemudian nilai / hasil dari proses kedua diteruskan ke *user* pertama. Skenario keempat apabila dalam suatu DFD sistem terdapat dua *external entity* dan satu proses. Hasil dari kembalian proses tersebut dikembalikan ke *user* pertama dan *user* kedua.
2. Hasil *generate* DFD yang berupa XML, dapat diimplemetasikan ke *web service* dengan baik, yaitu dengan mengambil setiap ID proses DFD untuk dipetakan ke alur *web service* yang telah dibuat.
3. Sistem dapat dijalankan sesuai dengan alur skenario DFD dan *web service*. Baik itu yang melibatkan satu *tenant* (*user*) dan dua *tenant*, satu proses dan dua proses, serta *data store*.

5.2 Saran

Berikut ini merupakan beberapa saran untuk penelitian di masa akan datang. Saran-saran ini didasarkan pada hasil perancangan, implementasi dan pengujian pada sistem. Saran-saran tersebut antara lain :

1. Proses *mapping web service* dengan DFD dapat diterapkan dan diaplikasikan secara lebih fleksibel.
2. Penerapan sistem lebih mengarah ke *multiuser*, tidak hanya sebatas *tenant*, sehingga dalam sistem yang dijalankan lebih kompleks.
3. Sistem yang dijalankan tidak hanya satu DFD dalam suatu aplikasi, tapi juga dua atau lebih DFD dapat dijalankan secara bersamaan.

Daftar Pustaka

[1] S. S. Kolhatkar, "XML Based Representation of DFD," *International Journal of Advanced Computer Science and Applications,* vol. 2, pp. 22-24, 2011.

[2] F. M. Nafie, "The Comparison of the Workflow Management Systems Bizagi, Arabdox, Bonita and Joget," *International Journal of Engineering Science Invention* , vol. 5 , no. 5, pp. 26-31 , 2016.

[3] H. Zang, W. Liu, H. Xiong and X. Dong, "Analyzing Data Flow Diagrams by Combination of Formal Methods and Visualization Techniques," *Journal Of Visual Languange And Computing,* p. 2, 2018.

[4] . L. J. Mitchell, PHP Web Services, Sebastopol (California): O'Reilly Media, 2016.

[5] M. Rosen, B. Lublinsky, K. T. Smith and M. J. Balcer, Service-Oriented Architecture and Design Strategies, Indianapolis: Wiley Publishing, 2008.

[6] M. Papazoglou, P. Traverso, S. Dustar, F. Leyman and B. Krammer, "Service-Oriented Computing Research Roadmap," *International Journal of Cooperative Information System (IJCIS),* p. 255, 2008.

[7] J. Ericson and K. Siau, "Web Services, Service-Oriented and Service-Oriented Architecture : Separating Hype from Reality," *Journal of Database Management,* vol. 3, pp. 42-54, 2008.

[8] W. van der Aalst, "Don’t go with the flow: web services composition standards exposed," *IEEE Intelligent Systems*, pp. 6-72, 2003.

[9] G. U. Abriani and M. A. Yaqin, "Implementasi Metode Semantic Similarity untuk Pengukuran Kemiripan Makna antar Kalimat," *Journal of Computer Science and Applied Informatics*, vol. 1, pp. 47-57, 2019.

[10] U. Kaur and D. Singh, "BPEL – Business Process Execution Language Process Creation and Fault Localization Framework," *5th International Conference- Confluence The Next Generation Information Technology Summit (Confluence)*, pp. 452-456, 2014.

[11] A. B. J. Al-Jazairi, Aysar At-Tafasir Li Kalam Al-'Aliyy Al-Kabir, Madinah: Nahr Al-Khair, 1993.