BAB II

KAJIAN PUSTAKA

2.1 Penelitian Relevan

Fajar Akbar, Susafa'ati (2019). Penerapan Metode *Waterfall* untuk Perancangan Sistem Informasi Kehadiran Siswa berbasis *Web* dan *SMS Gateway* Studi kasus SMA IP Yakin Jakarta. Sistem yang dibuat dibangun menggunakan bahasa pemrograman PHP, Kerangka kerja yang di pakai menggunakan *Framework Codeigniter* dan penyampaian informasinya menggunakan *SMS Gateway*. Pada penelitian ini, metode penelitian yang digunakan pada tahap pengumpulan data menggunakan Metode Observasi dan Studi Pustaka. Pada tahap pengembangan sistem, penelitian ini menggunakan Metode *Waterfall*. Penelitian yang dihasilkan adalah Pengoptimalan kecepatan proses perekapan kehadiran siswa serta memberikan informasi ke wali murid tentang kehadiran anaknya di sekolah melalui *SMS Gateway*.

Mohammad Bhanu Setyawan, Andi Fajaryanto Cobantoro, Angga Prasetyo, (2020). *Prototype* untuk Memonitoring Presensi Siswa Menggunakan *Fingerprint* dengan Kendali *Raspberry Pi*. Sistem yang dibuat pada penelitian ini berbasis web dan telegram dengan kendali *Raspberry Pi.* Metode Pengembangan sistem yang digunakan menggunakan metode *Waterfall*. Pada sistem ini, siswa yang hadir akan melakukan absensi dengan verifikasi sidik jari yang di kontrol oleh *Raspberry Pi*, waktu *scanning* setiap siswa nantinya akan disimpan di dalam *database fingerprint*, kemudian *Raspberry Pi* akan mengirimkan pesan ke grup telegram yang beranggotakan para wali murid menggunakan *Open API* dan Protokol yang disediakan melalui pengembangan Telegram Bot untuk memonitoring kehadiran anaknya. Karena sistem ini tidak menggukanan *SMS Gateway* dalam penyampaian

informasinya, sistem ini dapat menekan biaya pengeluaran dikarenakan tidak membutuhkan pulsa.

Sendy Aprilia, (2020). Sistem Informasi Berbasis *Website* Menggunakan *API* WhatsApp dengan Metodologi *Incremental* (Studi Kasus : SMP Negeri 29 Pekanbaru). Sistem yang dibuat pada penelitian ini menggunakan bahasa pemrograman PHP dan penyampaian informasinya menggunakan *API* WhatsApp. Metode Pengembangan Sistem yang digunakan dalam penelitian ini menggunakan metode *Incremental*. Dimulai dengan tahap analisis yang di peroleh dari hasil wawancara dan *Gap Analysis*. Kemudian dilanjutkan dengan tahap mendesain sistem yang akan dibuat, dilanjutkan dengan pengkodean dan pengujian aplikasi. Pengujian fungsionalitas sistem dilakukan dengan *Blackbox Testing*. Untuk mengetahui aplikasi yang telah di hasilkan sudah sesuai harapan pengguna atau belum, dilakukan dengan cara *User Acceptance Testing* dan *Gap Analysis*. Dengan menggunakan sistem ini, orang tua dapat memantau kehadiran anaknya melalui pesan WhatsApp yang dikirimkan oleh sistem menggunakan *API* WhatsApp.

Penjelasan diatas dapat diringkas dalam tabel berikut :

Tabel 2.1. Penetian Relevan

No	Nama	Judul	Metodologi	Hasil

1.	Fajar Akbar, Susafa'ati (2019)	Penerapan Metode *Waterfall* untuk Perancangan Sistem Informasi Kehadiran Siswa berbasis Web dan *SMS Gateway* Studi kasus SMA IP Yakin Jakarta	Menggunakan *Framework Codeigniter* dan *SMS Gateway*	Mengoptimalkan kecepatan proses perekapan kehadiran siswa serta memberikan informasi ke wali murid tentang kehadiran anaknya di sekolah melalui *SMS Gateway*

Tabel 2. 1. Lanjutan Penelitian Relevan

No	Nama	Judul	Metodologi	Hasil
2.	Mohammad Bhanu Setyawan, Andi Fajaryanto Cobantoro, Angga Prasetyo, (2020)	*Prototype* untuk *Memonitoring* Presensi Siswa Menggunakan *Fingerprint* dengan Kendali *Raspberry Pi*	Menggunakan *Raspberry Pi* Dan *Open API* Telegram	Memberikan informasi ke orang tua tentang kehadiran anaknya di sekolah melalui pesan informasi di grup Telegram.
3.	Sendy Aprilia, (2020)	Sistem Informasi Berbasis *Website* Menggunakan	Menggunakan Bahasa Pemrograma	Memberikan informasi ke orang tua tentang kehadiran

		API WhatsApp dengan Metodologi *Incremental* (Studi Kasus : SMP Negeri 29 Pekanbaru)	n PHP dan *API* WhatsApp	anaknya di sekolah melalui pesan WhatsApp.

Dari Tabel diatas, dapat diambil kesimpulan perbedaan penelitian ini dengan 3 (Tiga) penelitian di atas, yaitu Pada Penelitian Pertama, Sistem yang dibuat Berbasis Web dan Penyampaian Informasi yang digunakan menggunakan *SMS Gateway*. Pada Penelitian Kedua, Sistem yang dibuat menggunakan alat *Fingerprint* yang dikendalikan oleh *Raspberry PI* dan Penyampaian Informasi yang digunakan menggunakan *Open API* Telegram yang akan mengirimkan pesan secara otomatis ke grup Telegram yang beranggotakan para wali murid. Pada Penelitian Ketiga, Sistem yang dibuat berbasis Web dan Penyampaian Informasi yang digunakan Menggunakan Pesan WhatsApp yang telah diintegrasikan dengan *API* WhatsApp.

2.2 Landasan Teori

2.2.1 Bot

Bot merupakan singkatan dari kata robot yang berarti pekerja. Dalam dunia komputer, Bot digunakan untuk melakukan suatu pekerjaan manual menjadi otomatis. Dengan menggunakan Bot, seseorang bisa menghemat tenaga dan waktu dalam melakukan sebuah pekerjaan (Hartoto, 2016).

2.2.2 WhatsApp

WhatsApp merupakan aplikasi pesan instan pada ponsel pintar yang berfumgsi untuk mengirim dan menerima pesan melalui koneksi

internet tanpa menggunakan pulsa (WhatssApp, 2021). Aplikasi ini dapat di unduh secara gratis di *playstore.* Tercatat sejak 14 Februari 2020, 2 (dua) miliar orang secara aktif telah menggunakan WhatsApp (Kompas.com, 2021).

2.2.3 *Database*

a Pengertian *Database*

Database atau dalam bahasa indonesia disebut Basis Data adalah kumpulan data di dalam komputer yang disimpan secara sistematis dan dapat dimanipulasi menggunakan perangkat lunak untuk menghasilkan sebuah informasi. Perangkat lunak yang berfungsi untuk memasukkan dan mengambil data ke dan dari media penyimpanan disebut *Database Management Sytem* (DBMS) (Dahlan, 2009).

b Istilah-istilah dalam *Database*

Berikut kosa kata atau istilah yang ada pada *Database* :

1. *Entity*

Entity atau Entitas adalah sebutan untuk tempat, orang kejadian, atau konsep yang informasinya direkam (Muthma'innah, 2014). Contohnya pada sekolah terdapat *Entity* Guru, Siswa, Mata Pelajaran dan sebagainya.

2. *Attributte*

Attributte atau Atribut adalah sebutan untuk mewakili suatu *Entity*. Setiap *Entity* pasti mempunyai *Attributte*. *Attributte* disebut juga sebagai data *field*, data elemen atau data item (Muthma'innah, 2014). Contohnya dalam *Entity* Siswa terdapat Nomor induk siswa, nama, alamat dan sebagainya.

3. *Data Value*

Data Value adalah informasi atau nilai yang disimpan didalam sebuah atributte (Muthma'innah, 2014). Misal dalam *attributte* nama siswa terdapat *data value* Budi.

4. *Record*

   Record adalah sebutan untuk sebuah baris pada *database* (Muthma'innah, 2014). *Record* berisi elemen-elemen yang menginformasikan suatu *entity* secara lengkap.

5. *Field*

   Field adalah sebutan untuk sebuah kolom pada *database* (Muthma'innah, 2014). *Field* berisikan atribut –atribut dalam sebuah *entity*.

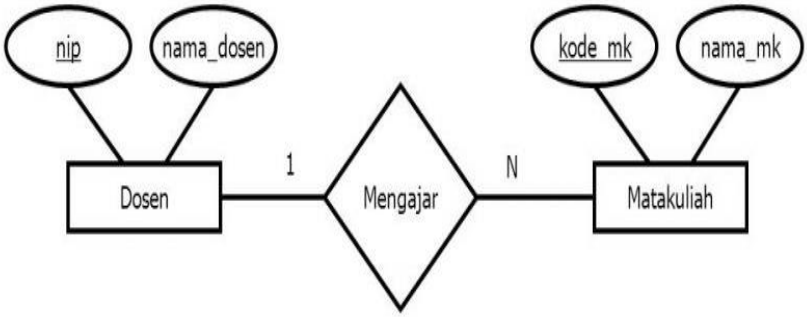
6. *Database Management System*

   Database Managament System atau biasa disingkat DBMS adalah Perangkat Lunak yang memungkinkan pengguna untuk mendefiiniskan, membuat, memelihara dan mengatur akses ke *database* atau basis data (Muthma'innah, 2014).

2.2.4 *Entity Relationship Diagram*

a Pengertian *Entity Relationship Diagram*

Entity Relationship Diagram atau disingkat ERD adalah suatu model penyajian data dengan menggunakan entitas dan atribut yang saling berhubungan (Permana, 2013). ERD digunakan untuk memodelkan struktur data dan hubungan antar data, untuk mendeskripsikannya digunakan beberapa notasi dan simbol.



Gambar 2.1. Contoh ERD

b Simbol-Simbol pada *Entity Relationship Diagram*

Entity Relationship Diagram mempunyai 3 (tiga) simbol, yaitu :

a. Entitas

Entitas adalah objek benda nyata atau abstrak. Simbol dari Entitas ini biasanya menggunakan persegi panjang.

b. Atribut

Setiap entitas memiliki elemen yang disebut atribut yang berfungsi untuk mendeskripsikan karakteristik dari entitas tersebut. Simbol atribut biasanya menggunakan elips.

c. Relasi

Relasi adalah Hubungan antar sejumlah entitas yang berasal dari himpunan entitas yang berbeda. Simbol Relasi biasanya menggunakan simbol panah. Jenis hubungan diantara dua tipe entitas dinyatakan dengan istilah hubungan *one-to-one*, *one-to-many*, dan *many-to-many*.

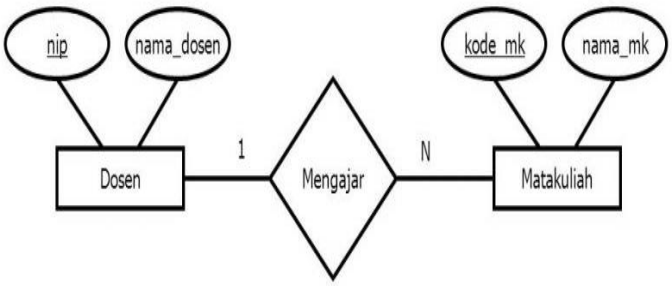
1. Hubungan *one-to-one* (1:1)

   Setiap entitas pada tipe entitas A paling banyak berpasangan dengan satu entitas pada tipe entitas B, begitupun sebaliknya.

Gambar 2. 2. Contoh Hubungan *one-to-one*

2. Hubungan *one-to-many* (1:M)

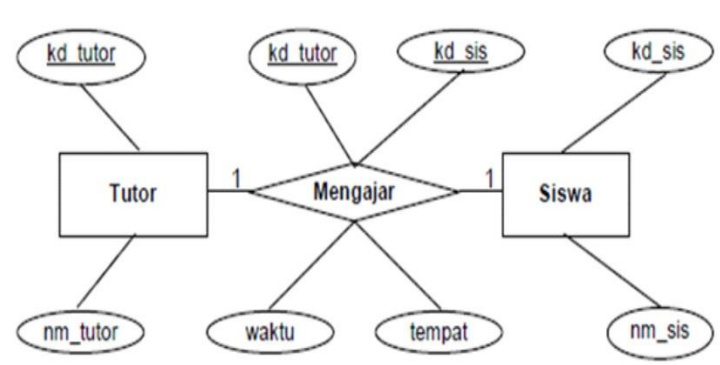
   Setiap entitas pada tipe entitas A bisa berpasangan dengan banyak entitas pada tipe entitas B, sedangkan setiap entitas pada entitas B hanya dapat berpasangan dengan satu entitas pada entitas A.



Gambar 2. 3. Contoh Hubungan *one-to-many*

3. Hubungan *many-to-many* (M:M)

   Setiap entitas pada suatu tipe entitas A bisa berpasangan dengan banyak entitas pada tipe entitas B dan begitupun sebaliknya.



Gambar 2. 4. Contoh Hubungan *many-to-many*

2.2.5 *Flowchart*

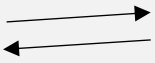
a Pengertian *Flowchart*

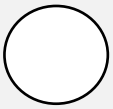
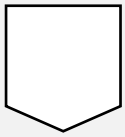
Flowchart atau Bagan Alir adalah Bagan yang menunjukkan aliran (*flow*) prosedur sistem atau program secara logika. Bagan Alir dipakai sebagai alat bantu komunikasi dan dokumentasi (Hartono, 1999).

b Simbol-Simbol pada *Flowchart*

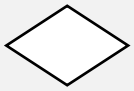
Berikut simbol-simbol yang ada di dalam *Flowchart* :

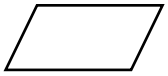
Tabel. 2.2. Simbol-Simbol *Flowchart*

Simbol	Nama	Fungsi
	*Flow Direction Symbol/ Connecting Line*	Digunakan untuk menguhubungkan simbol yang satu dengan yang lainnya.

	*Processing*	Digunakan untuk menunjukkan proses yang dilakukan oleh komputer.
	*Connector*	Digunakan untuk menyatakan sambungan dari suatu proses di halaman yang sama.
	*Terminal*	Digunakan untuk memulai atau mengakhiri program.
	*Offline Connector*	Digunakan untuk menyatakan sambungan dari suatu proses di halaman yang berbeda

Tabel. 2.2. Lanjutan Simbol *Flowchart*

Simbol	Nama	Fungsi
	*Decision*	Digunakan untuk menunjukkan penyeleksian data yang memberikan pilihan untuk langkah selanjutnya.

	*Input/ Output*	Digunakan untuk menyatakan *input* dan *output*.
	*Document*	Digunakan untuk menunjukkan *input* dan *output* yang berasal dari dokumen.
	*Magnetic Disk*	Digunakan untuk menunjukkan *input* dan *output* yang berasal dari *Disk magnetis*.

c Pedoman dalam membuat *Flowchart*

Berikut pedoman dalam membuat sebuah *Flowchat* :

1. *Flowchart* sebaiknya digambar dari atas ke bawah dimulai dari sebelah kiri pada suatu halaman.
2. Kegiatan yang ada di dalam *Flowchart* harus ditunjukkan dengan jelas.
3. Harus ditunjukkan dari mana kegiatan dimulai dan dimana akan berakhir.
4. Masing-masing kegiatan yang ada di dalam *Flowchart* sebaiknya digunakan suatu kata yang mewakili suatu pekerjaan.

5. Kegiatan yang terpotong dan akan disambung di tempat lain harus ditunjukkan dengan jelas menggunakan simbol penghubung.

2.2.6 *Data Flow Diagram*

a Pengertian *Data Flow Diagram*

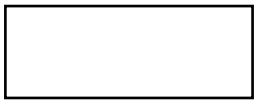
Data flow Diagram atau biasanya disigkat DFD adalah suatu model logika data atau proses yang dibuat untuk menggambarkan darimana asal data dan kemana tujuan data yang keluar dari sistem (Kristanto, 2008). Arus data (*Data Flow*) di DFD diberi simbol suatu anak panah. Arus data mengalir diantara proses (*process*), simpanan data (*data store*), dan kesatuan luar (*external entity*).

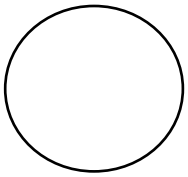
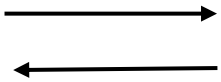
DFD dapat digunakan untuk merepresentasikan sistem atau perangkat lunak pada berbagai level abstraksi. DFD dapat dibagi menjadi beberapa level, yang dapat merepresentasikan arus informasi atau fungsi yang lebih detail. DFD yang digambarkan di awal disebut context diagram. Diagram yang digambar secara lebih rinci lagi yang disebut *overview* diagram.

b Simbol-simbol *Data Flow Diagram*

Terdapat 4 (empat) Simbol yang digunakan dalam DFD, yaitu Entitas (*Source*), Proses (*Process*), Arus data (*Data flow*) dan Penyimpanan data (*Data store*) (Permana, 2013). Adapun bentuk simbol-simbolnya dapat dilihat pada **Tabel 2.3.** Simbol DFD.

Tabel 2.3. Simbol DFD

Simbol	Keterangan
	Source (Entitas)

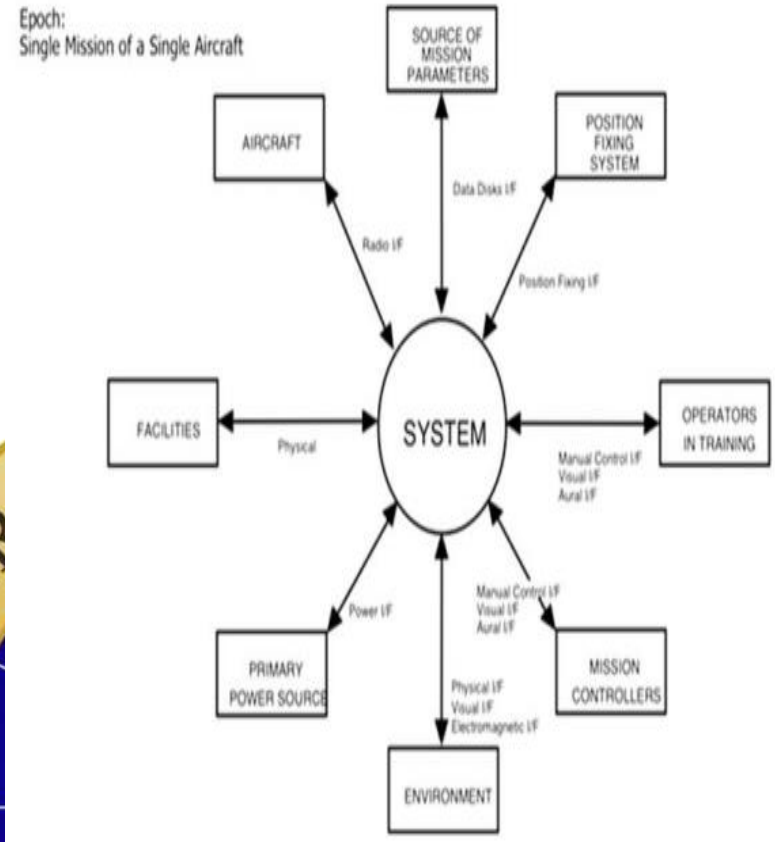
	Process (Proses)
	Data Flow (Aliran/ Arus Data)
	Data Store (Penyimpanan Data)

c. Tahapan Perancangan *Data Flow Diagram*

Adapun Tahapan Perancangan pembuatan *Data Flow Diagram* adalah sebagai berikut (Permana, 2013) :

a. Membuat *Context Diagram.*

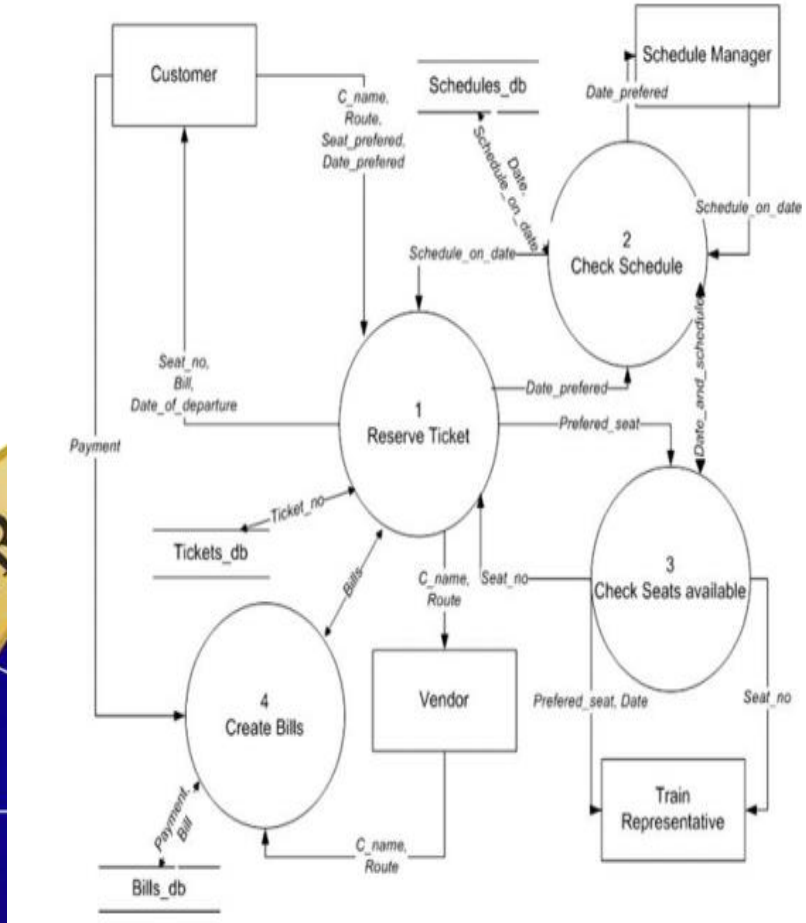
Context Diagram menggambarkan sistem yang akan dibentuk sebagai satu kesatuan yang berinteraksi dengan orang lain dan sistem. Proses pemberian nomor pada *Context Diagram* menggunakan angka 0 dan hanya ada satu proses didalamnya. Pada *Context Diagram* tidak diperbolehkan menyimpan data karena diagram ini hanya menunjukkan sistem secara keseluruhan.



Gambar 2.6. *Context Diagram*

b. Membuat DFD Level 1

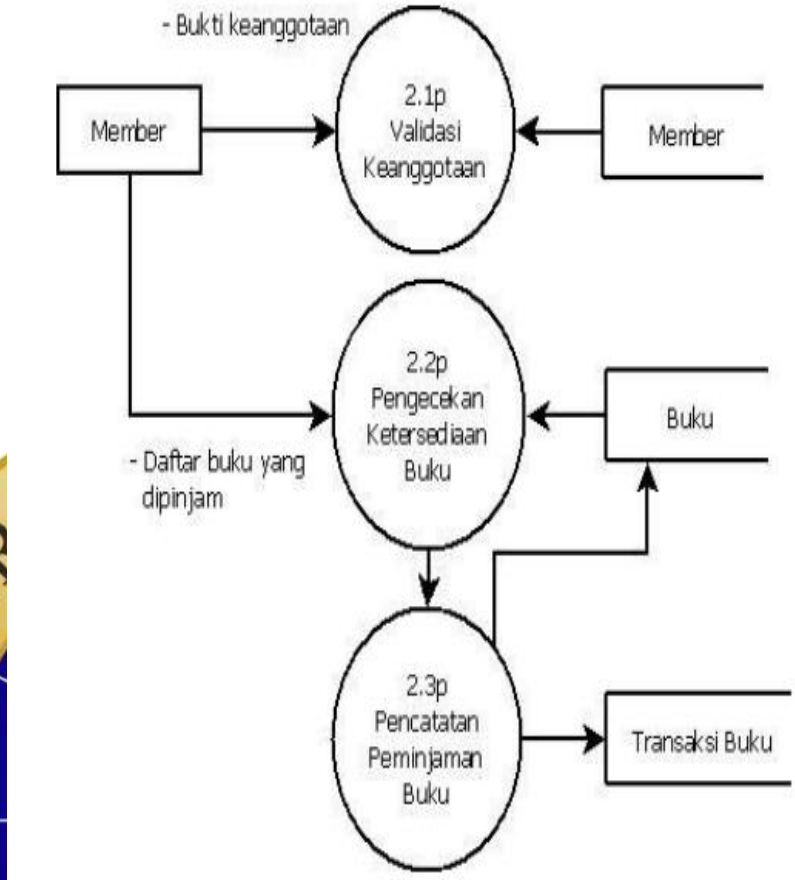
DFD level 1 digunakan untuk mendeskripsikan proses-proses yang ada pada *Context Diagram.* DFD Level 1 merupakan hasil dari pemecahan *Context Diagram* yang telah dibuat. DFD Level 1 disebut juga diagram nol atau lebih sering disebut *overview* diagram. Pada diagaram ini, dapat digambarkan penyimpanan data yg digunakan. Pemberian nomor pada setiap proses yang ada didalam diagram ini dimulai dari angka 1.0, 2.0, 3.0, dan seterusnya.



Gambar 2.7. Contoh DFD Level 1

c. Membuat DFD Level 2 dan seterusnya

DFD Level 2 merupakan hasil dari pemecahan DFD Level 1 yang telah dibuat. DFD level 2 dan seterusnya disebut diagram rinci, misal DFD level 2 dari proses 1.0 maka diagram tersebut dapat disebut Diagram Rinci 1.0. Penomoran proses pada level 2 dimulai dengan angka 1.1, 1.2, 1.3 dan seterusnya. Pada diagram level 2 harus benar-benar diperhatikan keseimbangan aliran data antara diagram nol dan diagram rinci juga keseimbangan pada *data store* yang ada.



Gambar 2.8. Contoh DFD Level 2

2.2.7 Scrum

a Pengertian Scrum

Scrum merupakan suatu pendekatan iteratif pada pengembangan perangkat lunak yang mengusung prinsip *agile* (Swastha, 2001). Scrum menekankan kolaborasi, perangkat lunak yang berfungsi dengan baik, manajemen tim yang baik (*Self-Management*), dan kemampuan untuk beradaptasi dengan perubahan sesuai dengan realitas bisnis yang muncul. Metode ini biasanya dibentuk dalam sebuah tim yang terdiri dari *Product Owner*, Tim Pengembang (*Development Team)* dan *Scrum Master.*

a. *Product Owner*

Poduct Owner adalah seorang penanggung jawab yang bertugas untuk memaksimalkan nilai bisnis dari produk yang dihasilkan oleh *Development Team* (Ken Schwaber, 2017).

b. *Development Team*

Development Team adalah ahli profesi yang bekerja untuk menghasilkan sebuah *increment* pada *product backlog* dalam sebuah sprint (Ken Schwaber, 2017).

c. *Scrum Master*

Scrum Master adalah orang yang ahli dalam Scrum yang bertugas untuk mengenalkan dan membimbing penggunaan Scrum (Ken Schwaber, 2017).

b Artefak Scrum

Metode Scrum mempunyai 3 (tiga) artefak, yaitu *Product Backlog*, *Sprint Backlog*, dan *Increment* (Ken Schwaber, 2017). Ilustrasi Artefak Scrum dapat dilihat pada **Gambar 2.9.** Artefak Scrum.

Gambar 2.9. Artefak Scrum

a. *Product Backlog*

Product Backlog merepresentasikan kebutuhan dari suatu produk perangkat lunak (Ken Schwaber, 2017). *Product Backlog* berisi daftar item yang diurutkan. Item tersebut dinamakan *Product Backlog Item* (PBI) dan menjelaskan apa yang dibutuhkan dalam produk perangkat lunak. *Product Backlog Item* (PBI) dapat memiliki banyak bentuk, seperti fitur, fungsi, penyempurnaan, dan perbaikan. Setiap PBI memiliki beberapa atribut seperti deskripsi, jenis, prioritas, nilai, dan estimasi.

b. *Sprint Backlog*

Sprint Backlog merupakan daftar tugas yang menjelaskan fitur apa saja yang harus dibuat untuk sebuah produk selama *Sprint* tertentu (Ken Schwaber, 2017). *Sprint Backlog* berisi *Product Backlog Item* yang dipilih untuk dikerjakan dalam *Sprint*, serta *plan* yang dapat ditindaklanjuti untuk menghasilkan *Product increment*.

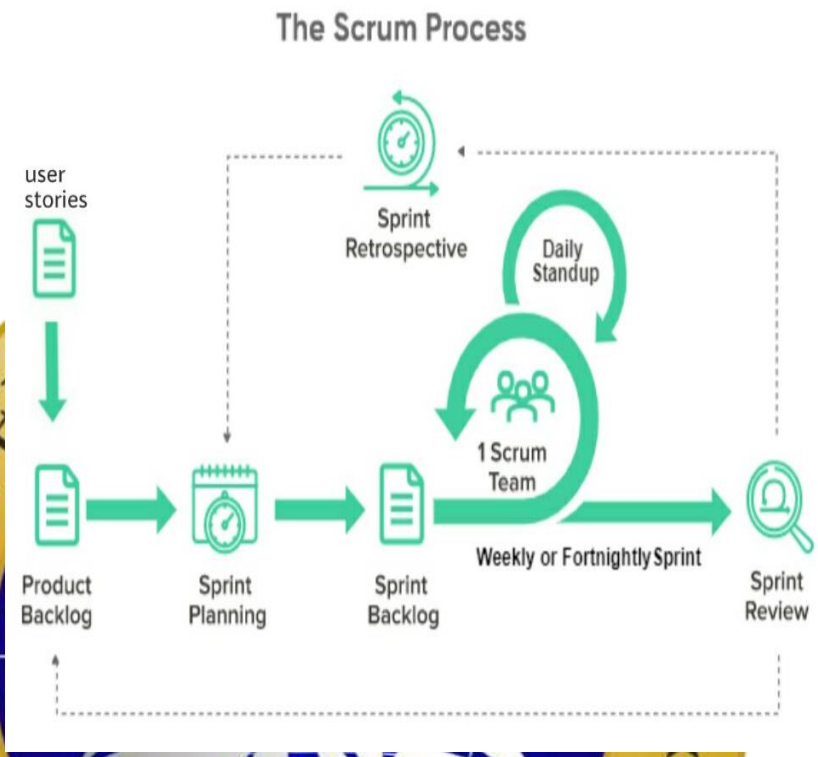
c. *Product Increment*

Product Increment adalah *Product Backlog Item* yang diselesaikan selama *Sprint* berlangsung (Ken Schwaber, 2017). Setiap *Product Increment* akan dikumpulkan dengan *Product Increment* sebelumnya untuk verifikasi menyeluruh dan memastikan bahwa semua *Product Increment* dapat berfungsi bersama.

c Tahapan Scrum

Dalam pelaksanaanya, Metode Scrum mempunyai beberapa tahapan, yaitu *User Stories, Product Backlog, Sprint planning, Daily Scrum, Sprint Review* dan *Sprint Retrespective*

(Irma Kartika Wairooy, 2021). Ilustrasi Tahapan Scrum dapat dilihat pada **Gambar 2.10.** Tahapan Scrum.



Gambar 2.10. Tahapan Scrum

1. *User Stories*

Untuk menganalisis kebutuhan sistem, perlu dilakukan analisis berdasarkan kebutuhan proses bisnis yang dilakukan oleh pengguna. *User Stories* menjelaskan persyaratan sistem dalam bahasa Secara alami mudah dipahami oleh *end-user* yang tidak memiliki latar belakang IT. *User Stories* adalah bahasa semi terstruktur. Dengan *User Stories*, *Product Owner* dapat menemukan persyaratan yang mencakup proses apa saja yang dibutuhkan oleh sistem untuk memenuhi kebutuhan pengguna (Ken Schwaber, 2017).

2. *Product Backlog*

Product Backlog merepresentasikan kebutuhan dari suatu produk perangkat lunak. *Product Backlog* berisi daftar item yang diurutkan. Item tersebut dinamakan *Product Backlog Item* (PBI) dan menjelaskan apa yang dibutuhkan dalam produk perangkat lunak. *Product Backlog Item* (PBI) dapat memiliki banyak bentuk, seperti fitur, fungsi, penyempurnaan, dan perbaikan. Setiap PBI memiliki beberapa atribut seperti deskripsi, jenis, prioritas, nilai, dan estimasi (Ken Schwaber, 2017).

3. *Sprint Planning*

Sprint Planning dibuat untuk merencanakan kolaborasi pekerjaan yang dapat dilakukan oleh tim scrum. S*print planning* membahas tentang hal yang akan dilakukan untuk meningkatkan hasil yang diperoleh dari sprint tersebut. Pada *sprint planning*, *output* yang dihasilkan berupa *sprint backlog*. *Sprint backlog* berisi target modul-modul aplikasi yang harus diselesaikan pada setiap *sprint*. Isi *sprint backlog* merupakan bagian dari *product backlog* (Ken Schwaber, 2017).

4. *Daily Scrum*

Daily Scrum dibuat untuk memantau dan mensinkronisasi pekerjaan. Setiap hari diadakan *standup meeting* untuk melaporkan hal yang telah dikerjakan oleh anggota tim. Pada setiap akhir *meeting*, akan dibahas waktu penyelesaian dari masing-masing fungsi untuk mengetahui waktu penyelesaian yang telah diperbaharui dan mengetahui sisa pengerjaan yang harus dikejar pada sisa waktu sprint berjalan (Ken Schwaber, 2017).

5. *Sprint Review*

Sprint review dilakukan di akhir *sprint* untuk memeriksa apakah diperlukan penambahan atau perubahan *Product backlog*. Dalam pertemuan ini, Tim Scrum bekerja sama dengan *Stakeholde*r untuk membahas berbagai hal yang telah selesai dilakukan selama *sprint*, melakukan peninjauan terhadap *Increment* yang sudah selesai, serta melakukan perubahan atau penambahan *Product backlog* bila di perlukan (Ken Schwaber, 2017).

6. *Sprint Retrospective*

Tahap ini dilakukan oleh seluruh tim Scrum untuk meninjau kegiatan *sprint* yang telah dilakukan agar kinerja di *sprint* berikutnya menjadi lebih baik. Nantinya, Scrum master akan memberikan saran kepada tim Scrum untuk meningkatkan performa agar proses *sprint* berikutnya lebih efektif (Ken Schwaber, 2017).

2.2.8 Selenium

a Pengertian Selenium

Selenium adalah alat pengujian otomatis browser yang digunakan untuk mengotomatiskan pengujian aplikasi web (Selenium, 2020). Dikembangkan pada tahun 2004 oleh Jason Huggins dalam bentuk *Library Java Script*. Selenium bersifat *open source* atau dapat diunduh secara gratis. Sejak tahun 2006 Google mengembangkan Selenium untuk keperluan sendiri maupun publik, khususnya Selenium 2 (*Web Driver*). Selenium diluncurkan dalam beberapa paket perangkat lunak, antara lain:

1. Selenium RC

Selenium RC (*Remote Control*) atau disebut juga Selenium 1 merupakan paket perangkat lunak utama yang berisi

semua fungsi pengujian utama (*Selenium Core*). Paket ini berupa Java *script* dan dikendalikan oleh *server* babasis Java (*Selenium RC Server*).

2. Selenium *Web Driver*

   Selenium *Web Driver* atau disebut Selenium 2 merupakan pengembangan Selenium RC yang dilakukan oleh Simon Stewart pada tahun 2006. Simon Stewart mengembangkan Selenium RC menjadi *Web Driver* yang dapat mengakses browser secara langsung melalui *API* yang telah disediakan tanpa menggunakan *Java Script*. Pada tahun 2012 Simon Stewart dan David Burns dari Mozilla melakukan negosiasi dengan W3C untuk menetapkan *Web Driver* sebagai internet standar.

3. Selenium *IDE*

   Selenium IDE adalah prototipe untuk menguji sebuah *script*. Selenium IDE merupakan plug-in browser Mozilla Firefox yang memiliki antarmuka sederhana dalam membuat skenario pengujian sekaligus menyediakan catatan operasi pengguna yang dapat diekspor ke berbagai bahasa. Kekurangan dari Selenium IDE adalah tidak memiliki kemampuan untuk menangani *Looping* dan Percabangan.

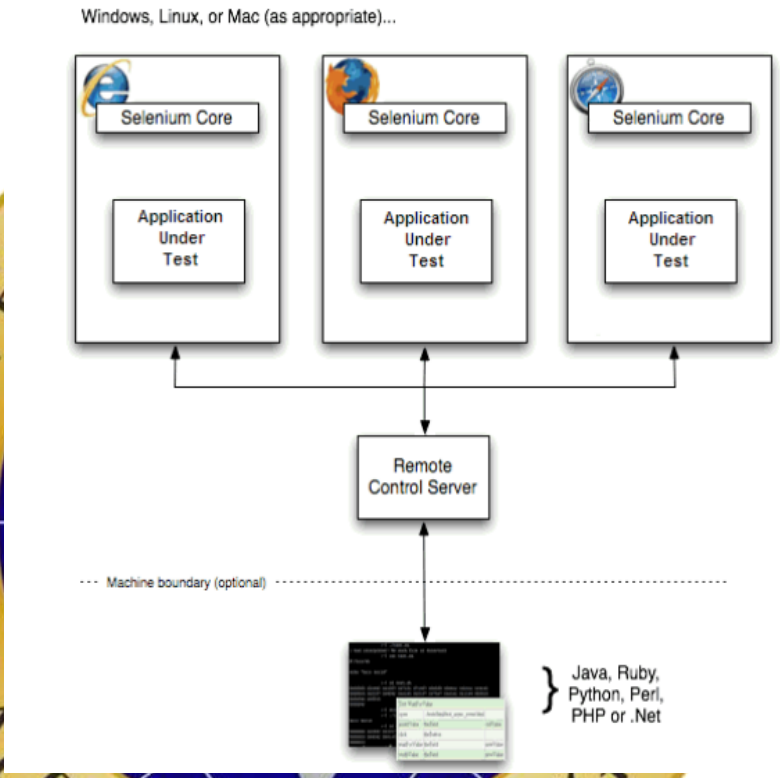
4. Selenium *Grid*

   Selenium *Grid* merupakan paket alat pengujian yang dapat menggabungkan beberapa Selenium RC untuk meningkatkan skalabilitas pengujian sehingga memungkinkan untuk melakukan pengujian secara paralel dalam skala besar.

b Arsitektur Selenium

Pada **Gambar 2.11.** ditunjukkan bahwa Selenium *RC* menjalankan beberapa *browser* dan mengintepretasi perintah pengujian dalam bentuk *selenese command* dari skenario pengujian

melalui *browser* kemudian me-*record* layaknya seperti *HTTP Proxy* antara *browser* dengan aplikasi yang diuji. Selenium *core* merupakan aplikasi *Java Script* yang berperan untuk menerjemahkan perintah *selenese command* kepada *browser* (Gede Karya, 2012).



Gambar 2.11. Arsitektur Selenium

c Pola Desain Testing yang didukung

Selenium dapat di jalankan dalam beberapa pola pengujian, diantaranya:

1. *Static content*, pola ini berfungsi untuk mengecek apakah terdapat suatu konten tertentu dalam sebuah situs.
2. *Link*, pola ini berfungsi untuk mengecek terdapat suatu *link* atau tidak.
3. *Function*, pola ini berfungsi untuk memanipulasi sebuah *input* atau *output* tertentu.

4. *Dynamic element*, pola ini berfungsi untuk mencari sebuah elemen dalam suatu halaman menggunakan sebuah parameter yang berupa identitas unik (ID).
5. *Ajax*, menggunakan JS dan XML yang dinamis tanpa me-*reload* suatu halaman web.
6. D*ata Driven Testing* dan *Database Validation*. D*ata Driven Testing* berfungsi untuk melakukan sebuah pengujian yang dapat diambil dari suatu basis data atau *file* eksternal (CSV atau *text*). Pola ini digunakan untuk membuat variasi data pada saat pengujian. *Database Validation Testing*, berfungsi untuk mencocokkan antara hasil halaman web dengan isi dari suatu basis data. Pola ini dapat digunakan untuk membandingkan hasil tampilan di web dengan hasil *query* basis data tertentu.

2.2.9 Apache POI

Apache POI dikembangkan oleh Apache Software Foundation dan didistribusikan dengan lisensi sumber terbuka atau *open source*. Apache POI merupakan sebuah *Library* yang digunakan untuk memanipulasi file Microsoft Office, baik itu Microsoft Word (.docx), Microsoft Excel (.xlsx), atau PowerPoint (.pptx). Apache POI mendukung beberapa bahasa pemrograman, antara lain Java, Kotlin, Jython, Scala, Groovy dan lain-lain (Foundation, 2021).

2.2.10 Java

Java merupakan bahasa pemrograman yang dibuat dan dikembangkan pada tahun 1995 oleh James Gosling. Bahasa pemrograman ini merupakan bagian dari Platform *Java Sun Microsystem*. Sintaks Java banyak dikembangkan dari bahasa C dan C ++, akan tetapi dikarenakan Java dibuat agar mudah dipelajari dan

mudah dibaca, Sintkasnya lebih sederhana, lebih ketat, dan memiliki akses ke OS yang lebih terbatas (Bima, 2011).

Aplikasi Java ditulis sebagai file dengan ekstensi .java, dan file ini akan dikompilasi menjadi file .class. File .class adalah bytecode, apa pun sistem operasi atau arsitektur prosesor yang digunakan, file ini dapat berjalan di mesin virtual Java apa pun. Java adalah bahasa untuk semua kebutuhan, bersamaan, berbasis kelas, berorientasi objek, dan dirancang untuk tidak bergantung pada lingkungan (sistem operasi dan prosesor) tempat aplikasi berjalan.

Java mempunyai slogan "menulis sekali dan menjalankannya di mana saja". Saat ini, Java merupakan bahasa pemrograman yang paling banyak digunakan untuk membuat aplikasi-aplikasi di sistem tertanam, aplikasi seluler, desktop, dan web. Java mempunyai empat prinsip penting yang dijadikan sebagai tujuannya, keempat prinsip ini adalah :

1. Java harus "sederhana, *object oriented* dan mudah dimengerti".
2. Java harus "kuat dan aman".
3. Java harus "netral terhadap arsitektur sistem (*OS,processor*) dan bisa jalan di manapun".
4. Java harus bisa dijalankan dengan "kinerja yang tinggi.
5. Java harus "*interpreted, threaded* dan dinamis".

Dengan menggunakan 5 (lima) prinsip di atas, aplikasi Java memiliki tingkat popularitas yang tinggi, terutama di bidang aplikasi perusahaan. Semua prinsip di atas sangat cocok untuk jenis aplikasi ini. Java banyak digunakan di industri dengan anggaran TI yang tinggi, seperti perbankan dan telekomunikasi. Terdapat 3 (tiga) buah profil pada Java Platform, yaitu :

1. Java ME (*Java Micro Edition*) adalah java yang dapat dijalankan pada *embedded system* seperti *Java Card* dan *Handphone*.

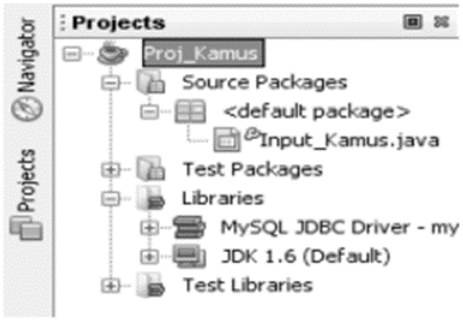
2. Java SE (*Java Standard Edition*) adalah java yang dapat dijalankan pada PC atau server sebagai aplikasi *standalone* dan aplikasi desktop.
3. Java EE (*Java Enterprise Edition*) adalah profil java yang digunakan untuk membuat aplikasi *Enterprise* seperti *Web Application (Servlet)* dan *Enterprise Java Bean (EJB).*

Instalasi platform Java mencakup dua paket aplikasi. Paket pertama adalah JRE (Java Runtime Environment), yang berisi semua aplikasi yang diperlukan untuk menjalankan aplikasi Java, seperti perpustakaan dan JVM (Java Virtual Machine). Paket kedua adalah JDK (Java Development Kit), yang berisi JRE dan menambahkan alat untuk membuat aplikasi Java, seperti kompiler Java (javac), dokumentasi Java (javadoc), dan file arsip Java (Jar).

2.2.11 Netbeans IDE

Netbeans IDE adalah sebuah *Integrated Development Environment* (IDE) yang digunakan untuk menulis, mengkompilasi, menguji dan men-debug aplikasi untuk platform Java dan bahasa pemrograman lainnya (Hudaya, 2013). Netbeans IDE memiliki 3 (tiga) bagian utama, yaitu:

a Bagian Kiri : *Project, Navigator* dan *File*. Pada Bagian ini berisi navigasi *file-file* Java yang akan dirancang.

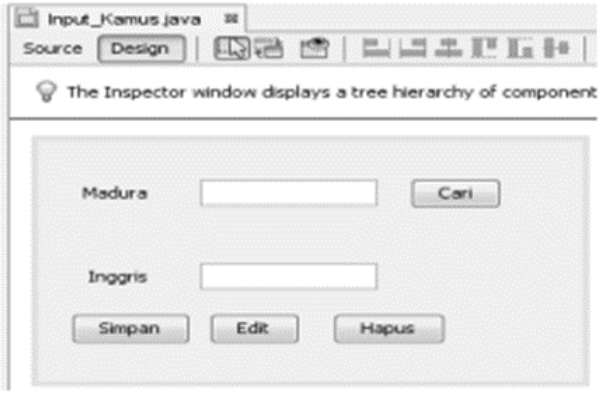


Gambar 2.12. Area *Project*

Gambar 2.13. *Navigator*

Gambar 2.14. *Files*

b Bagian Tengah. Bagian ini merupakan tempat perancangan *JFrame (form)*.

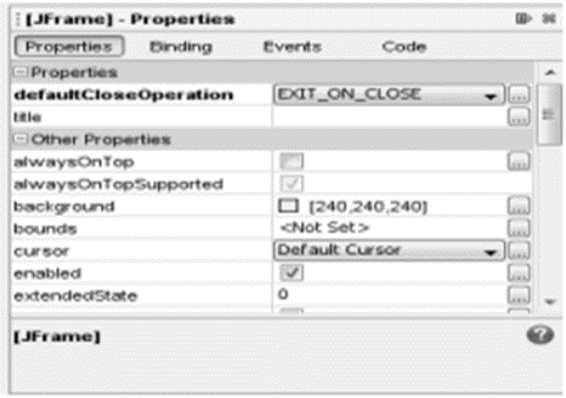


Gambar 2.15. Bagian *Form*

Pada bagian perancangan terdapat 2 (dua) tab, yaitu *Source* dan *Design*. *Source* adalah tempat untuk menulis kode program, sedangkan *Design* adalah tempat untuk mengatur tata letak komponen *form*.

c Bagian Kanan. *Palette* dan *Properties*. *Palette* berisi komponen-komponen yang akan diletakkan pada *JFrame*, sedangkan *Properties* berisi tempat pengaturan nilai untuk tiap *object* tersebut.

Gambar 2.16. *Palette*



Gambar 2.17. *Properties*

d. Bagian Bawah, *Output* Program. Bagian ini menampilkan pesan kesalahan atau keberhasilan kompilasi saat program dijalankan