BAB II
TINJAUAN PUSTAKA

II.1 Profil Komunitas Gerakan Bantu Seribu Anak Asuh

GBSA adalah sebuah komunitas sosial yang bergerak di bidang sosial terutama membantu anak-anak. Komunitas ini terbentuk pada tanggal 14 Januari 2012, tercetus dari beberapa orang yang peduli terhadap anak-anak. Diawal berdirinya komunitasi gerakan bantu seribu anak asuh ini hanya berupa sekumpulan orang yang dengan rutin membantu anak-anak disekitar dengan bantuan berupa peralatan sekolah. Seiring berjalannya waktu, kelompok ini akhirnya sepakat untuk konsisten membantu anak-anak dengan mendirikan komunitas gerakan bantu seribu anak asuh atau dikenal dengan GBSA.

GBSA hadir untuk ikut memberi solusi terhadap anak-anak putus sekolah. Tujuannya menjadi fasilitator penyambung kepanjangan para donatur kepada anak-anak yang putus sekolah. Bantuan yang diberikan berupa beasiswa (donasi rutin bersifat terikat) ataupun perlengkapan sekolah (donasi insidental bersifat bebas).

Logo dari komunitas gerakan bantu seribu anak asuh ini dapat dilihat pada Gambar II.1.



Gambar II.1 Logo Komunitas

II.1.1 Gambaran Umum Komunitas

Komunitas GBSA mencari anak asuh memiliki arti bahwa dimana GBSA memiliki tim dan mitra dari berbagai LSM dan komunitas yang siap menyediakan data anak asuh yang benar-benar layak untuk menerima bantuan. Data yang GBSA miliki telah melalui proses pengecekan lapangan oleh korwil di

wilayahnya. Kemudian donatur bantu berikan beasiswa dimana para donatur diminta kesediaannya untuk memberikan bantuan beasiswa kepada setiap anak asuhnya minimal selama satu tahun. Boleh didonasikan setiap bulan atau langsung berdonasi selama satu tahun. Donatur boleh memilih data yang ada dan melakukan transfer donasi setelah itu data anak asuh akan administrator GBSA kirimkan via email.

Anak asuh dapat meraih mimpi mereka maksudnya adalah GBSA akan menyalurkan donasi dari donatur kepada anak asuhnya dan melakukan pembinaan setiap bulannya. GBSA akan memantau perkembangan mereka dan memberikan laporan perkembangan anak setiap semesternya kepada donatur. Gambaran umum dari komunitas gerakan bantu seribu anak asuh dapat dilihat pada Gambar II.2.



Gambar II.2 Gambaran Umum Sistem

II.1.2 Visi dan Misi Komunitas

Visi dan misi dari komunitas gerakan bantu seribu anak asuh adalah:

1. Visi

Visi gerakan bantu seribu anak asuh adalah turut serta dalam mencerdaskan anak Indonesia.

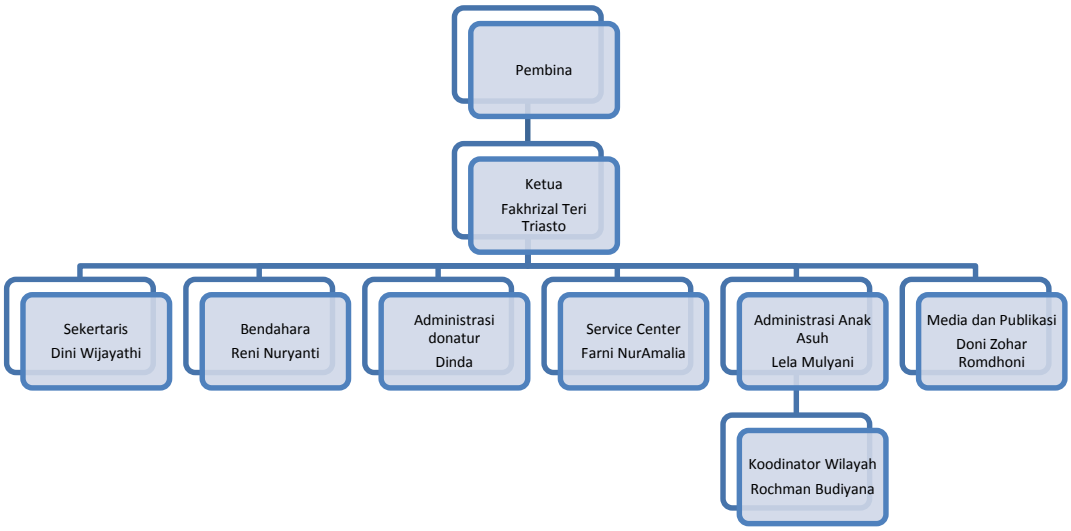
2. Misi

Misi dari komunitas gerakan bantu seribu anak asuh ini adalah:

a. Memberikan bantuan kepada anak yang kurang beruntung agar dapat terus melanjutkan sekolah dan mengembangkan bakatnya.
b. Bersama-sama memberikan pembinaan kepada anak asuh untuk membentuk generasi muda Indonesia yang cerdas, berkualitas dan kreatif.

II.1.3 Struktur Organisasi Komunitas

Struktur organisasi komunitas gerakan bantu seribu anak asuh dapat dilihat pada Gambar II.3.



Gambar II.3 Struktur Organisasi

Dari struktur organisasi komunitas tersebut yang akan menggunakan perangkat lunak yang dibangun adalah bagian *service center* yang akan bertindak sebagai administrator dari website GBSA.

II.1.4 Deskripsi Kerja Jabatan

Deskripsi kerja dari jabatan yang akan ikut serta dalam penggunaan perangkat lunak ini adalah *service center* dimana deskripsi pekerjaan yang dilakukan adalah:

1. Melayani calon donatur dan donatur yang ingin berkonsultasi seputar program donasi anak asuh.
2. Menerima Keluhan (*Complain*).
3. Mengelola website GBSA.
4. Memasukkan data anak asuh dari IJF.
5. Mendistribusikan laporan data anak asuh ke masing masing donatur.
6. Merekap data anak asuh, donatur, dan laporan.
7. Melakukan pendataan konfirmasi donasi (Klaim).

8. Menginformasikan *news* terbaru dari komunitas kepada donatur.

II.2 Landasan Teori

Landasan teori merupakan penjelasan berbagai konsep dasar dan teori-teori yang berkaitan dalam pembangunan perangkat lunak GBSA ini. Beberapa teori terkait dengan pembangunan perangkat lunak ini adalah konsep anak asuh, konsep *charity software*, sistem informasi, android, *structure analysis and design*, *object oriented analysis and design*, web service, JSON, pengujian *black box*, dan pengujian kuesioner.

II.2.1 Konsep Anak Asuh

Dalam hukum Islam, konsep anak asuh adalah disebut sebagai *Ihtidhàn* yaitu menjadikan seseorang yang bukan anaknya untuk di didik, di asuh dan diperlakukan dengan baik[5]. Ihtidhan berarti membiarkan anak asuh tetap menggunakan nama aslinya, tidak menasabkannya kepada orangtua asuhnya, tidak diwarisi. Semua kebaikan yang diberikan kepada anak asuh hanya sebatas pada pengertian berbuat baik kepada sesama yang memang dianjurkan oleh syari'at Islam. Anak-anak asuh tetap menjadi orang lain. Ia bukan mahram bagi keluarga yang mengasuhnya. Hal itu berarti harus memperlakukan anak asuh sesuai dengan apa yang telah disyari'atkan Islam sewaktu berinteraksi kepada orang lain yang bukan mahram.

Jika dilihat dari pandangan sosial, konsep anak asuh ini terkait dengan proses atau aktifitas pemberdayaan anak dimana menurut Jules Siboro, pemberdayaan anak asuh adalah upaya untuk mengembangkan diri dari keadaan tidak atau kurang berdaya menjadi berdaya, guna mencapai kehidupan yang lebih baik[6]. Pemberdayaan anak asuh adalah kegiatan dalam bentuk sosial, budaya, ekonomi, politik, dan kemapanan masyarakat untuk meningkatkan kekuatan, peranan dan keswadayaan mesyarakat miskin dalam satu kehidupan untuk meningkatkan kesejahteraan dan keamanannya.

II.2.2 Konsep Perangkat Lunak Anak Asuh

Munculnya berbagai macam komunitas di bidang sosial tentunya tak lepas dari perkembangan teknologi yang semakin canggih. Teknologi biasanya dijadikan media untuk lebih memasyarakatkan kegiatan sosial dari berbagai komunitas kepada masyarakat dengan tujuan masing-masing. Salah satu bentuk yang ada saat ini adalah perangkat lunak berupa website.

Dari hasil analisis terhadap lima buah website sosial yang bergerak di bidang anak asuh yang dapat dilihat pada Tabel II.1 ditemukan satu pola kesamaan konsep website yaitu adanya penyajian data tentang komunitas untuk mengajak pengunjung berdonasi, penyajian profil anak asuh, registrasi donatur, serta proses pendonasian dengan cara transfer ke bank tertentu secara *offline*.

Tabel II.1 Daftar Website Komunitas Anak Asuh

No	Nama Komunitas	Website
1	Yayasan Anak Anak Terang Indonesia	http://anakanakterang.web.id/
2	Yayasan Ayo' Asakola	http://www.ayoasakola.org/
3	Komunitas teplok	http://komunitasteplok.org/
4	Organisasi *Save Street Child*	http://savestreetchild.org/
5	Komunitas Mentari	http://komunitasmentari.com/

Selain itu ada pula beberapa website yang memang secara transparan menyajikan data keuangannya dalam website. Tentunya hal ini sangat baik untuk transparansi donasi mengingat komunitasnya bergerak di bidang sosial.

II.2.3 Sistem Informasi

Sistem adalah kumpulan komponen dimana masing-masing komponen memiliki fungsi yang saling berinteraksi dan saling tergantung serta memiliki satu kesatuan yang utuh untuk bekerja mencapai tujuan tertentu[7]. Sedangkan informasi adalah data yang sudah diolah ke dalam sebuah formulir bentuk yang bermanfaat dan dapat digunakan untuk manusia[8].

Jika disimpulkan, definisi dari sistem informasi adalah sekumpulan komponen yang memiliki fungsi yang saling ketergantungan dalam satu kesatuan

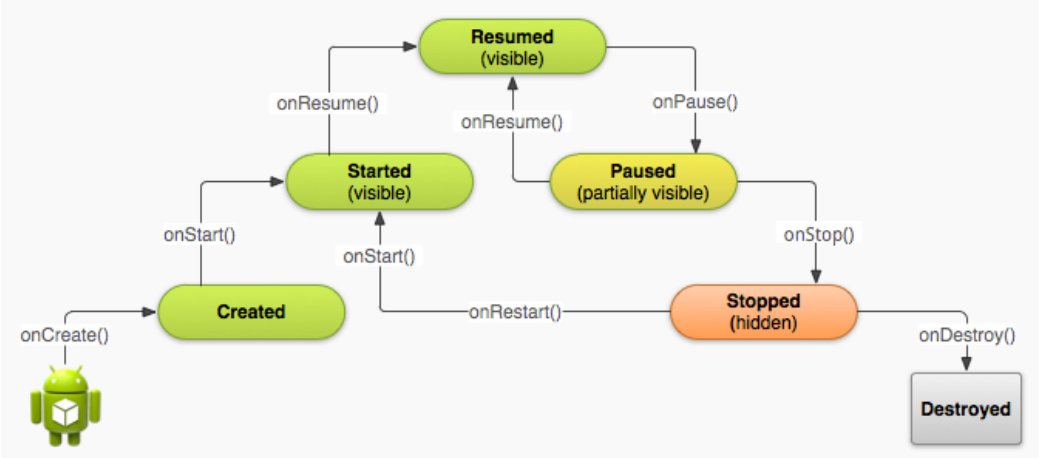
utuh dalam bentuk satu data yang telah diolah dalam sebuah bentuk yang dapat digunakan untuk manusia mencapai tujuan tertentu.

Dari pendapat lain, dapat dijelaskan bahwa sistem informasi merupakan seperangkat prosedur yang terorganisasi dengan sistematik yang jika dilaksanakan akan menyediakan informasi yang dapat dimanfaatkan dalam proses pembangunan keputusan[9].

II.2.4 Android

Android adalah sistem operasi smartphone layar sentuh seperti iOS iPhone dan OS BlackBerry yang dalam pengembangannya dipimpin oleh google. Sistem operasi ini bersifat Open Source dan dikembangkan berdasarkan kernel Linux yang disematkan pada gadget, baik itu handphone atau tablet[10]. Android memiliki OS yang sangat baik, cepat dan kuat serta memiliki antarmuka pengguna intuitif yang dikemas dengan pilihan dan fleksibilitas. Sedangkan android SDK (*Software Development Kit*) menyediakan *tools* dan API yang diperlukan untuk mengembangkan aplikasi pada platform android dengan menggunakan bahasa pemrograman Java.

Di dalam android terdapat *activity* dimana komponen ini memberi interaksi antara user dan aplikasi yang dibangun melalui user interface. *Activity* ini memiliki Siklus hidup yang dinamakan *Android Life Cycle*. Flowchart siklus hidup tersebut dapat dilihat pada Gambar 0.4.



Gambar 0.4 Android Life Cycle

Gambar diatas menjelaskan mengenai sebuah ilustrasi sederhana dari siklus hidup *activity* yang dinyatakan sebagai langkah piramida. Hal ini menunjukkan bagaimana untuk setiap *callback* yang digunakan untuk mengambil aktivitas langkah menuju *state* atas kemudian dilanjutkan ada metode *callback* yang mengambil langkah menurun. Kegiatan ini juga dapat kembali ke keadaan dari *statepause* dan *stop*. Siklus hidup *activity* ini menjadi sangat penting ketika kita hendak membuat aplikasi berbasis android. Gunanya adalah agar aplikasi yang dibangun berjalan dengan baik. Berikut penjelasan pada masing-masing *state*:

1. onCreate(), Method ini dipanggil ketika *activity* pertama kali dibuat.
2. onStart(), Method ini dipanggil ketika sebuah *activity* tampil ke pengguna
3. onResume(), Method ini dipanggil ketika *activity* yang berjalan pada saat itu dihentikan sementara (*paused*) dan *activity* sebelumnya dijalankan kembali(*resumed*). (Hasil dari method OnRestart())
4. onPause(), Method ini di panggil ketika *activity* di hentikan sementara (pause) dan berikutnya ketika dijalankan kembali akan berada dalam posisi resume dan memanggil method OnResume()
5. onStop(), Method ini dipanggil ketika *activity* tidak lagi tampak kepada pengguna
6. on*Destroy*(), Method ini dipanggil sebelum *activity* dihancurkan (*destroy*) oleh sistem (baik secara manual maupun untuk kepentingan pelonggaran memori).

Banyak sekali fitur yang dapat dimanfaatkan pada platform android ini. Diantaranya adalah *Location Based Services* (LBS) dan *Google Cloud Messaging* (GCM). *Location Based Services* (LBS) dapat dimanfaat untuk layanan yang menggunakan informasi geografis dalam memberikan informasi lokasi kepada pengguna, sebagai petunjuk posisi atau lokasi piranti *mobile* pengguna berada, dan menemukan rute jalan sesuai permintaan. LBS dapat digambarkan sebagai satu layanan yang berada pada pertemuan tiga teknologi yaitu: *Geographic Information System* (GIS), *Internet Service*, dan *Mobile Devices*[11]. Google

Cloud Messaging atau disingkat GCM merupakan suatu layanan yang disediakan oleh google, dengan layanan ini kita dapat mengirimkan data dari server ke user yang pada aplikasinya menggunakan layanan ini. Google Cloud Messaging dapat digunakan developer, khususnya developer android untuk mengirimkan *push notification.*

II.2.5 Structure Analysis and Design

Struktur analisis dan desain adalah sebuah metodologi yang di gunakan pada rekayasa perangkat lunak untuk mendeskripsikan sistem kearah fungsional. Pendekatan ini memecahkan masalah-masalah dalam aktifitas bisnis menjadi bagian-bagian kecil yang dapat disatukan kembali menjadi satu kesatuan yang utuh untuk memecahkan masalah[12].

Tools yang dapat digunakan pada pendekatan analisis pengembangan sistem secara terstruktur adalah:

1. ERD (*Entity Relationship Diagram*) adalah satu model jaringan yang menggunakan susunan data yang disimpan dalam sistem. ERD merupakan model jaringan data yang menekankan pada struktur dan hubungan antardata[13].
2. Diagram Konteks adalah diagram yang menggambarkan sebuah sistem secara menyeluruh yang selanjutnya akan lebih didetailkan oleh DFD. Konteks diagram juga biasa disebut sebagai DFD level 0.
3. Data Flow Diagram atau biasa di singkat DFD merupakan serangkaian diagram yang menggambarkan kegiatan-kegiatan yang ada dalam satu sistem. Teknik pembangunan DFD dimulai dengan menggambarkan sistem secara global dan dilanjutkan dengan analisis masing-masing bagian. Pada awalnya digambarkan konteks diagram yang menggambarkan sebuah sistem secara menyeluruh yang akan diinvestigasi. Konteks diagram tersebut dapat dikatakan sebagai DFD level 0. Analisis sistem yang lebih detail selanjutnya dapat dilakukan dengan menggambarkan DFD level 1,2 dan seterusnya.
4. Spesifikasi Proses merupakan tabel yang berisi keterangan deskripsi dari semua proses yang terdapat di DFD. Logika proses harus dituliskan secara

jelas baik menggunakan bahasa deskriptif atau *pseudo code* (tidak boleh campuran.

5. Kamus Data (*Data Dictionary*) merupakan fakta tentang data dan kebutuhan-kebutuhan informasi dari sistem informasi. Dengan menggunakan *data dictionary*, analis sistem dapat mendefinisikan data yang mengalir dalam sistem dengan lengkap[14].

II.2.6 Object Oriented Analysis and Design

Analisis dan desain berorientasi objek adalah cara baru dalam memikirkan satu masalah dengan menggunakan model yang dibuat menurut konsep sekitar dunia nyata. Tujuan dari analisis berorientasi objek adalah untuk mengembangkan model yang menggambarkan perangkat lunak komputer karena bekerja untuk memenuhi seperangkat persyaratan yang ditentukan user[15]. *Tools* yang dapat digunakan pada pendekatan analisis pengembangan sistem secara objek dapat menggunakan UML.

Unified Modelling Language (UML) adalah sebuah bahasa yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menggunakan *class* dan *operation object* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa bahasa berorientasi objek[16]. Dalam membangun block UML ada 3 hal yang harus diperhatikan, yaitu *object* (memodelkan konsep), *relationship* (mengkoneksikan *object*), dan *diagram* (*grouping* yang saling mengkoneksikan antara *object* dan *relationship*. Diagram yang umum dipakai dalam analisis dan desain adalah:

1. *Use Case Diagram*

Use Case Diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *Use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa use

case yang di-*include* akan dipanggil setiap kali use case yang meng-*include* dieksekusi secara normal. Sebuah use case dapat di- include oleh lebih dari satu use case lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang serupa. Sebuah use case juga dapat meng-*extend usecase* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar use case menunjukkan bahwa use case yang satu merupakan spesialisasi dari yang lain.

Dasar menentukan sebuah *use case* adalah *use case* merupakan sesuatu yang menyediakan beberapa hasil terukur kepada pengguna atau sistem eksternal. *Use case* harus memiliki sangat jelas kriteria lulus / gagal. Pengembang, tester, penulis teknis, dan pengguna harus secara eksplisit tahu apakah sistem memenuhi kasus penggunaan atau tidak. Setiap bagian dari *use case* yang memenuhi tes sederhana ini mungkin menjadi kandidat yang baik untuk *use case*[17].

2. *Use Case Scenario*

Sebuah diagram yang menunjukkan *use case* dan aktor mungkin menjadi titik awal yang bagus, tetapi tidak memberikan detail yang cukup untuk desainer sistem untuk benar-benar memahami persis bagaimana sistem dapat terpenuhi. Cara terbaik untuk mengungkapkan informasi penting ini adalah dalam bentuk penggunaan *use case scenario* berbasis teks per u*se case*nya. Berikut adalah dasar format penulisan *use case scenario*[17]. Dasar pembangunan *use case scenario* dapat dilihat pada Tabel II.2.

Tabel II.2 Dasar Pembangunan Use case scenario

Use Case Name	Berisi nama dari *Use case* yang akan digunakan	
Goal In Context	Menjelaskan apa yang aktor coba untuk dapatkan dari *Use case*	
Description	Menjelaskan gambaran dari *Use case*	
Related Use Case	Daftar *Use case* yang berhubungan dengan *Use case* tersebut	
Successful End Condition	Kondisi *Use case* jika berhasil	
Failed End Condition	Kondisi *Use case* jika gagal	
Actors	Daftar aktor yang dapat mengakses *Use case*	
Trigger	Aktifitas yang dilakukan untuk mengawali *Use case*	
Main Flow	Step	Action
	1	Deskripsi urutan aksi dari aktifitas *Use case*
	2	
	3	
Extension	Step	Branching Action

	2.1	Deskripsi urutan aksi lain selain urutan aksi utama
	2.2	

3. *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan *Output* tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *Output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, message akan dipetakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message [16]. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan icon khusus untuk objek *boundary*, *controller* dan *persistent entity*.

4. *Class Diagram*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) satu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok:

1. Nama dan stereotype
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut:

1. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
2. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
3. *Public*, dapat dipanggil oleh siapa saja.

Class dapat merupakan implementasi dari sebuah interface, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Sesuai dengan perkembangan *class* model, *class* dapat dikelompokkan menjadi *package*. Kita juga dapat membuat diagram yang terdiri atas *package*.

Class memiliki tipe-tipe *relationship*, diantaranya :

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain.
2. Agregasi, yaitu hubungan yang menyatakan bagian terdiri atas dimana ketika satu class di share atau direferensikan kepada objek yang ada di class lain.
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Komposisi, yaitu jenis relasi class diagram yang kuat dimana jika sebuah class tidak bisa berdiri sendiri dan harus merupakan bagian dari class yang lain, maka class tersebut memiliki relasi Composition terhadap class tempat dia bergantung tersebut. Sebuah relationship composition digambarkan sebagai garis dengan ujung berbentuk jajaran genjang berisi/solid.
5. Depedensi, salah satu jenis relasi class diagram yang lemah dimana objek dalam suatu class akan bekerja sangat singkat dengan objek yang ada pada class lain

5. *Activity Diagram*

Activity diagram menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing aliran berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. Sebuah aktivitas dapat direalisasikan oleh satu use case atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara use case menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity diagram* dapat dibagi menjadi beberapa object *swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu[16].

II.2.7 Web Service

Web Service adalah sekumpulan *application logic* beserta objek-objek dan metode-metode yang dimilikinya yang terletak di satu server yang terhubung ke internet [18]. Tujuan dari teknologi ini adalah untuk memudahkan beberapa aplikasi atau komponennya untuk saling berhubungan dengan aplikasi lain dalam sebuah organisasi maupun diluar organisasi menggunakan standar yang tidak terikat platform (*platform-neutral*) dan tidak terikat akan bahasa pemrograman yang digunakan (*language-neutral*). Hal tersebut dapat terjadi karena penggunaan XML standar yang didukung oleh banyak perusahaan besar di dunia, yang digunakan untuk bertukar data. Selain XML, terdapat pula format pertukaran data lain yaitu JSON.

II.2.8 JSON (*Javascript Object Notation*)

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generated*) oleh komputer[19]. JSON terbuat dari dua struktur yaitu:

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau associative array.
2. Daftar nilai terurutkan (*list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*Sequence*).

JSON menggunakan bentuk sebagai berikut:

1. Objek

Objek adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma).

2. Larik

Larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma).

3. Nilai

Nilai (value) dapat berupa sebuah string dalam tanda kutip ganda, atau angka, atau true atau false atau null, atau sebuah objek atau sebuah larik. Struktur-struktur tersebut dapat disusun bertingkat.

4. String

String adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan backslash *escapes* "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java.

5. Angka, Format oktal dan heksadesimal tidak digunakan.

II.2.9 Pengujian *Black Box*

Metode pengujian *blackbox* fokus pada keperluan penelusuran kesalahan fungsional dari software. Ujicoba *blackbox* berusaha untuk menemukan kesalahan dalam beberapa kategori, diantaranya :

1. Fungsi-fungsi yang salah atau hilang
2. Kesalahan interface
3. Kesalahan dalam struktur data atau akses database eksternal
4. Kesalahan performa
5. kesalahan inisialisasi dan terminasi

Teknik pengujian black box terdiri dari 10 jenis diantaranya *Equivalence Partitioning, Boundary Value Analysis/Limit Testing, Comparison Testing, Sample Testing, Robustness Testing, Behavior Testing, Requirement Testing, Performance Testing, Endurance Testing, Cause-Effect Relationship Testing.* Salah satunya yang akan dibahas adalah Equivalence partitioning.

Equivalence partioning merupakan metode ujicoba blackbox yang membagi domain input dari program menjadi beberapa kelas data dari kasus ujicoba yang dihasilkan. Kasus uji penanganan single yang ideal menemukan sejumlah kesalahan (misalnya : kesalahan pemrosesan dari seluruh data karakter) yang merupakan syarat lain dari suatu kasus yang dieksekusi sebelum kesalahan umum diamati.

II.2.10 Pengujian Kuesioner

Kuesioner adalah sebuah daftar pernyataan yang harus diisi oleh orang yang akan dievaluasi (responden). Metode yang digunakan dalam kuesioner pada penelitian ini adalah skala Likert. Dalam skala likert, responden diminta untuk membaca dengan seksama setiap pernyataan yang disajikan, kemudian ia diminta untuk menilai pernyataan-pernyataan tersebut.

Derajat penilaian responden terhadap suatu pernyataan terbagi dalam 5 kategori yang tersusun secara bertingkat, mulai dari Sangat Tidak Setuju (ST), Tidak Setuju (T), Tidak Memutuskan (N), Setuju (S), dan Sangat Setuju (SS). Atau dapat pula sebaliknya. Pernyataan tiap kuesioner dibuat berdasarkan aspek-aspek yang diteliti. Bobot pemberian skor yang digunakan dapat dilihat pada Tabel II.3.

Tabel II.3 Panduan pemberian skor[20]

Jenis Pernyataan	Bobot pendapat				
	SS	S	N	T	TS
Positif	5	4	3	2	1
Negatif	1	2	3	4	5

Skor yang telah dihitung pada setiap pernyataan kemudian dikalikan dengan masing-masing bobot tersebut sesuai dengan skenario kuesioner yang telah dibuat. Setelah itu total kan seluruh bobot jawaban tersebut kemudian bagi dengan total responden yang nantinya menjadi nilai rata-rata. Nilai rata-rata inilah yang diambil sebagai acuan sikap dimana jika nilai rata-rata kurang dari 3, maka dapat diartikan responden bersikap negatif dan jika nilai rata-rata lebih dari sama dengan 3, maka dapat diartikan responden bersikap positif terhadap tujuan yang ingin peneliti capai. Untuk lebih jelasnya dapat dilihat pada rumus dibawah ini.

$$\overline{X} = \frac{\sum Total}{n} ,$$

Dimana,

$\overline{X} \geq 3\ bersikap\ positif$

$\overline{X} < 3\ bersikap\ negatif$

Keterangan :

$\overline{X}$ = rata-rata

$\sum Total = jumlah\ seluruh\ nilai\ setelah\ dikalikan\ dengan\ bobot$

n = total responden