

BAB 2

TINJAUAN PUSTAKA

2.1 Renovasi Rumah

Renovasi rumah merupakan kegiatan yang dilakukan untuk memperbaiki, memperbaiki atau mengganti sebagian bangunan rumah untuk mencapai kondisi yang lebih baik.

Dilihat dari skalanya, renovasi dapat berupa pengerjaan yang kecil, seperti sekedar mengganti keramik dinding kamar mandi yang berskala sedang seperti membongkar dapur, sampai yang berskala besar seperti merombak ulang seluruh ruang di rumah. Sementara bila dilihat tujuannya, beragam alasan dikemukakan pemilik rumah soal renovasi. Beberapa yang umum ditemui adalah alasan-alasan seperti: bangunan rumah mengalami kerusakan, kebutuhan ruangan dalam rumah bertambah, rumah kurang nyaman, selera desain rumah berubah dan perawatan berkala.[2]

Estimasi biaya adalah penghitungan kebutuhan biaya yang diperlukan untuk menyelesaikan suatu kegiatan atau pekerjaan[2]. Estimasi biaya diaplikasi renovin akan digunakan untuk estimasi biaya renovasi rumah. Aplikasi renovin akan memberikan informasi estimasi biaya seperti jumlah bahan material dan estimasi harga bahan material.

Pencarian tukang terdekat diaplikasi renovin digunakan untuk mempermudah pengguna untuk mencari tukang terdekat disaat pengguna membutuhkan tukang untuk merenovasi rumah.

2.2 Landasan Teori

Landasan teori yang berkaitan dengan materi atau teori yang digunakan sebagai acuan melakukan penelitian. Landasan teori yang diuraikan merupakan hasil studi literatur, buku-buku, maupun situs internet.

2.2.1 Aplikasi

Aplikasi adalah penggunaan dalam suatu komputer, instruksi (*instruction*) atau pernyataan (*statement*) yang disusun sedemikian rupa sehingga komputer dapat memproses input menjadi output. Menurut Kamus Kamus Besar Bahasa Indonesia, aplikasi adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu. Aplikasi adalah suatu program komputer yang dibuat untuk mengerjakan dan melaksanakan tugas khusus dari pengguna. Aplikasi merupakan rangkaian kegiatan atau perintah untuk dieksekusi oleh komputer. Program merupakan kumpulan instruction set yang akan dijalankan oleh pemroses, yaitu berupa software. Bagaimana sebuah sistem komputer berpikir diatur oleh program ini.

Program inilah yang mengendalikan semua aktifitas yang ada pada pemroses. Program berisi konstruksi logika yang dibuat oleh manusia, dan sudah diterjemahkan ke dalam bahasa mesin sesuai dengan format yang ada pada instructionset. Program aplikasi merupakan program siap pakai yang dirancang untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain. Contoh aplikasi ialah program pemroses kata dan *Web Browser*. Aplikasi akan menggunakan sistem operasi (OS) komputer dan aplikasi yang lainnya yang mendukung. Istilah ini mulai perlahan masuk ke dalam istilah Teknologi Informasi semenjak tahun 1993, yang biasanya juga disingkat dengan app.

Secara historis, aplikasi adalah software yang dikembangkan oleh sebuah perusahaan. App adalah software yang dibeli perusahaan dari tempat pembuatnya. Industri PC tampaknya menciptakan istilah ini untuk merefleksikan medan pertempuran persaingan yang baru, yang parallel dengan yang terjadi antar sistem operasi yang dimunculkan[2].

2.2.2 Android

Sejarah *Android* pada mulanya berasal dari perusahaan bernama Android, Inc. didirikan tempatnya di Palo Alto, California, pada Oktober tahun 2003 oleh Andy Rubin (pendiri Danger), Rich Miner seorang pendiri *Wildfire Communications, Inc.*, Nick Sears seorang mantan VP T-Mobile, dan Chris White

seorang kepala desain dan pengembangan antarmuka Web TV untuk mengembangkan mengembangkan sebuah "perangkat seluler pintar yang lebih sadar tentang lokasi dan preferensi penggunaannya"[3]. Logo *Android* bisa di lihat pada gambar 1.1



Gambar 2.1 Logo *Android*[2].

Android adalah sistem operasi untuk telepon seluler yang berbasis Linux. *Android* menyediakan platform terbuka bagi para pengembang buat menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam piranti bergerak. Awalnya, Google Inc. membeli *Android Inc.*, pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan *Android*, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia[3].

Di dunia terdapat dua jenis distributor sistem operasi Android. Pertama yang mendapat dukungan penuh dari Google atau Google Mail Service (GMS) dan kedua adalah yang benar – benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai Open Handset Distribution (OHD). Sekitar September 2007 Google mengenalkan Nexus One, salah satu jenis *smartphone* yang menggunakan *Android* sebagai sistem operasinya. Telepon selular ini diproduksi oleh HTC Corporation dan tersedia di pasaran pada 5 Januari 2008. Pada 9 Desember 2008, diumumkan anggota baru yang bergabung dalam program kerja *Android ARM Holdings*, *Atheros Communication*, diproduksi oleh Asustek Computer Inc, Garmin Ltd, *Softbank*, Sony Ericson, Toshiba Corp, dan Vodafone Group Plc. Seiring

pembentukan Open Handset Alliance, OHA mengumumkan produk perdana mereka *Android*, perangkat *mobile* yang merupakan modifikasi *kernel Linux 2.6*. Sejak *Android* dirilis telah dilakukan berbagai pembaharuan berupa perbaikan *bug* dan penambahan fitur baru[3].

Android adalah sistem operasi untuk telepon seluler yang berbasis *Linux*. Yang dirancang untuk perangkat bergerak layar sentuh seperti telepon pintar dan komputer tablet. Antarmuka pengguna *Android* umumnya berupa manipulasi langsung, menggunakan gerakan sentuh yang serupa dengan tindakan nyata, misalnya menggeser, mengetuk untuk memanipulasi objek di layar, serta papan ketik virtual untuk menulis teks. *Android* menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri dan untuk digunakan oleh bermacam peranti bergerak (*mobile device*). Hal ini memungkinkan para pengembang menulis kode terkelola (*managed code*) dalam bahasa pemrograman Java, mengontrol peranti via perpustakaan Java yang dikembangkan Google. Telepon pertama yang menggunakan sistem operasi *Android* adalah HTC *Dream* yang dirilis pada 22 Oktober 2008. Adapun versi-versi *Android* yang pernah dirilis adalah sebagai berikut:

1. *Android* versi 1.1
2. *Android* versi 1.5 (*Cupcake*)
3. *Android* 1.6 (*Donut*)
4. *Android* versi 2.0/2.1 (*Eclair*)
5. *Android* Versi 2.2 (*Froyo*)
6. *Android* Versi 2.3 (*Gingerbread*)
7. *Android* Versi 3.0 (*Honeycomb*)
8. *Android* Versi 4.0 (*Ice Cream Sandwich*)
9. *Android* Versi 4.1 (*Jelly Bean*)
10. *Android* Versi 4.4 (*KitKat*)
11. *Android* Versi 5.0 (*Lollipop*)
12. *Android* Versi 6.0 (*Marshmallow*)
13. *Android* Versi 7.0 (*Nougat*)

2.2.2.1 Arsitektur Android

Arsitektur *Android* Google menggambarkan *Android* seperti sebuah tumpukan *software*. Setiap lapisan dari tumpukan ini terdiri dari beberapa program yang mendukung fungsi-fungsi spesifik dari sistem operasi. Dalam paket sistem operasi *Android* terdiri dari beberapa unsur seperti tampak pada gambar 2.6. Secara sederhana arsitektur *Android* merupakan sebuah *kernel Linux* dan sekumpulan pustaka C / C++ dalam suatu *Framework* yang menyediakan dan mengatur alur proses aplikasi[3].

2.2.2.2 Android Life Cycle

Aplikasi *Android* terdiri dari beberapa fungsi dasar seperti mengedit catatan, memutar *file* musik, membunyikan alarm, atau membuka kontak telepon. Fungsi-fungsi tersebut dapat diklasifikasikan ke dalam empat komponen *Android* yang berbeda seperti ditunjukkan pada, klasifikasi tersebut berdasarkan kelas-kelas dasar *java* yang digunakan

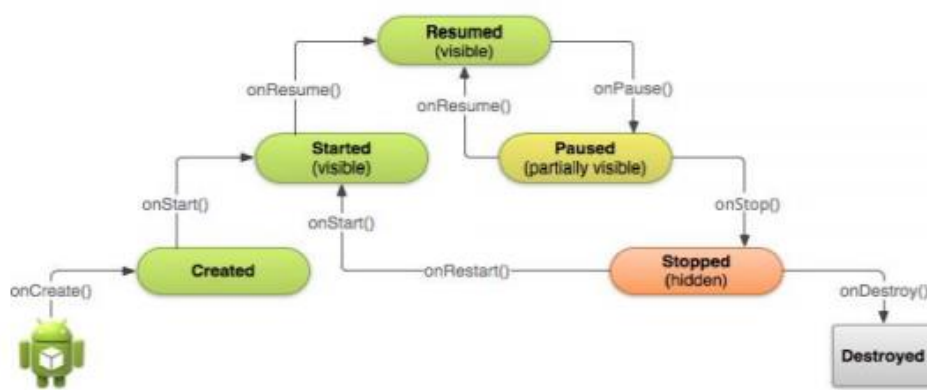


Gambar 2.2 Komponen *Android*[3].

Setiap aplikasi pasti menggunakan minimal satu dari komponen tersebut akan, tetapi terdapat beberapa komponen yang mengharuskan mencantumkan *specified permission* sebelum digunakan seperti komponen *Service*, *Broadcast Receiver*, *Content Provider*[4].

Android memiliki paradigma pemrograman lain tidak seperti paradigma pemrograman biasa di mana aplikasi yang dijalankan pada fungsi *main()*, sistem *Android* menjalankan kode dalam *method Activity* dengan menerapkan metode

callback tertentu yang sesuai dengan tahap tertentu dari siklus hidup. Setiap aplikasi yang berjalan dalam sistem operasi *Android* memiliki siklus hidup yang berbeda dengan aplikasi desktop atau web. Hal ini dikarenakan aplikasi *mobile* memiliki tingkat interupsi proses yang lumayan tinggi seperti ketika *handling* panggilan masuk aplikasi diharuskan menghentikan proses sementara. Penerapan siklus hidup juga berguna untuk memastikan aplikasi tidak menghabiskan sumber daya baterai pengguna[4].



Gambar 2.3 Siklus Hidup *Android*[4].

Terdapat beberapa state dalam siklus hidup *Android* yang terjadi. Siklus Hidup *Android* akan tetapi hanya beberapa dari state tersebut yang menjadi Statis diantaranya :

1. *Resumed*

Resumed terjadi ketika aplikasi berjalan setelah *state paused*. State ini akan menjalankan perintah program yang ditulis pada *method onResume()*[4].

2. *Paused*

Dalam keadaan ini aktivitas yang terjadi dihentikan secara sementara tetapi masih terlihat oleh pengguna karena terdapat proses yang memiliki prioritas lebih tinggi seperti panggilan telepon. Aplikasi tidak dapat menjalankan perintah apa pun ataupun menampilkan apa pun dalam step ini[4].

3. *Stopped*

Dalam keadaan ini, aplikasi benar-benar tidak ditampilkan dan tidak terlihat oleh pengguna tetapi masih meninggalkan *service* di *background*[4].

State lain seperti *Created* dan *Started* bersifat sementara dan sistem dengan cepat menjalankan state berikutnya dengan memanggil metode *life cycle callback* berikutnya. Artinya, setelah sistem *OnCreate()* dipanggil, dengan cepat sistem akan memanggil *method OnStart()*, kemudian diikuti oleh *onResume()*[4].

Pada penelitian ini versi *Android* yang digunakan adalah *Android* versi 5.0 (nougat) karena mempunyai material merupakan hasil desain terbaru sebagai spesifikasi minimal cocok untuk pengembangan aplikasi karena sudah mempunyai hasil desain *Android* terbaru selain itu. Versi mempunyai kelebihan dalam peningkatan pada pencarian kontekstual sehingga bisa menentukan lokasi dengan tepat ke tempat aplikasi tertentu dan akan mulai secara tepat. Secara garis besar arsitektur *Android* dapat dijalankan dan digambarkan sebagai berikut : *Application* dan *Widgets*. *Application* dan *Widgets* adalah layer di mana *user* berhubungan dengan aplikasi saja, di mana biasanya *user* men-download aplikasi, melakukan instalasi dan menjalankan aplikasi[4].

1. *Framework*

Android adalah “Open Development Platform” yaitu *Android* menawarkan kepada pengembang atau member kemampuan untuk membangun aplikasi yang inovatif. Pengembang bebas untuk mengakses perangkat keras, akses informasi *resource*, menjalankan *service background*, mengatur alarm, dan menambahkan status notifikasi. Komponen-komponen yang termasuk di dalam *applications frameworks* adalah sebagai berikut:

- a. *Views*
- b. *Content provider*
- c. *Resource manager*
- d. *Notification manager*
- e. *Activity manager*

2. *Libraries*

Libraries adalah layer dimana fitur-fitur *Android* berada, biasanya para pembuat aplikasi mengakses *libraries* untuk menjalankan aplikasinya.

3. *Android Runtime*

Layer yang membuat aplikasi *Android* dapat dijalankan dimana dalam prosesnya menggunakan Implementasi *Linux. Dalvik virtual machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi *Android*.

4. *Linux Kernel*

Linux Kernel adalah layer di mana inti dari *operating system* dari *Android* itu berada. Berisi *file-file system* yang mengatur sistem *processing, memory, recourse, drivers*, dan sistem-sistem operasi *Android* lainnya[4].

2.2.2.3 *Android Studio*

Definisi *Android Studio* *Android Studio* adalah IDE resmi untuk pengembangan aplikasi *Android*, berdasarkan IntelliJ IDEA. Di atas IntelliJ yang kuat code editor dan pengembang alat, *Android Studio* menawarkan lebih banyak fitur yang meningkatkan produktivitas Anda ketika membangun aplikasi *Android*, seperti [8]: 1. Sebuah fleksibel berbasis Gradle membangun sistem 2. Membangun varian dan beberapa generasi *file* APK 3. Kode template untuk membantu Anda membangun fitur aplikasi umum 4. Sebuah layout editor kaya dengan dukungan untuk drag dan drop tema *editing* 5. alat Lint untuk menangkap kinerja, kegunaan, kompatibilitas versi, dan masalah lainnya[4]

Sistem operasi android adalah sistem informasi yang di pilih oleh aplikasi Renovin karena lebih mudah dan efisien dikarenakan pengguna yang Rata-rata menggunakan smarphone berbasis sistem operasi android.

2.2.3 *Global Positioning System (GPS)*

Global Positioning System (GPS) merupakan sebuah alat atau sistem yang dapat digunakan untuk menginformasikan penggunanya di mana dia berada (secara global) di permukaan bumi yang berbasis satelit. Data dikirim dari satelit berupa sinyal radio dengan data digital[5].

2.2.3.1 Definisi *Global Positioning System* (GPS)

GPS (*Global Positioning System*) adalah sistem navigasi yang berbasis satelit yang saling berhubungan yang berada di orbitnya. Satelit-satelit itu milik Departemen Pertahanan (*Departemen of Defense*) Amerika Serikat yang pertama kali diperkenalkan mulai tahun 1978 dan pada tahun 1994 sudah memakai 24 satelit. Untuk dapat mengetahui posisi seseorang maka diperlukan alat yang diberi nama GPS *reciever* yang berfungsi untuk menerima sinyal yang dikirim dari satelit GPS. Posisi diubah menjadi titik yang dikenal dengan nama *Way-point* nantinya akan berupa titik-titik koordinat lintang dan bujur dari posisi seseorang atau suatu lokasi kemudian di layar pada peta elektronik[5].

2.2.3.2 Cara kerja GPS

Bagian yang paling penting dalam sistem navigasi GPS adalah beberapa satelit yang berada di orbit bumi atau yang sering kita sebut di ruang angkasa. Satelit GPS saat ini berjumlah 24 unit yang semuanya dapat memancarkan sinyal ke bumi yang lalu dapat ditangkap oleh alat penerima sinyal tersebut atau GPS *Tracker*. Selain satelit terdapat 2 sistem lain yang saling berhubungan, sehingga jadilah 3 bagian penting dalam sistem GPS. Ketiga bagian tersebut terdiri dari: GPS *Control Segment* (Bagian Kontrol), GPS *Space Segment* (bagian angkasa), dan GPS *User Segment* (bagian pengguna)[5].

2.2.3.2.1 GPS *Control Segment*

Control segment GPS terdiri dari lima stasiun yang berada di pangkalan *Falcon Air Force*, Colorado Springs, *Ascension Island*, Hawaii, Diego Garcia dan Kwajalein. Kelima stasiun ini adalah mata dan telinga bagi GPS. Sinyal-sinyal dari satelit diterima oleh bagian kontrol, kemudian dikoreksi, dan dikirimkan kembali ke satelit. Data koreksi lokasi yang tepat dari satelit ini disebut data *ephemeris*, yang kemudian nantinya dikirimkan ke alat navigasi yang kita miliki[5].

2.2.3.2.2 GPS Space Segment

Space Segment adalah terdiri dari sebuah jaringan satelit yang terdiri dari beberapa satelit yang berada pada orbit lingkaran yang terdekat dengan tinggi nominal sekitar 20.183km di atas permukaan bumi. Sinyal yang dipancarkan oleh seluruh satelit tersebut dapat menembus awan, plastik dan kaca, namun tidak bisa menembus benda padat seperti tembok dan rapatnya pepohonan. Terdapat 2 jenis gelombang yang hingga saat ini digunakan sebagai alat navigasi berbasis satelit. Masing-masingnya adalah gelombang L1 dan L2, di mana L1 berjalan pada *frequensi* 1575.42 MHz yang bisa digunakan oleh masyarakat umum, dan L2 berjalan pada *frequensi* 1227.6 Mhz di mana jenis ini hanya untuk kebutuhan militer saja[5].

2.2.3.2.3 GPS User Segment

User segment terdiri dari antenna dan prosesor *receiver* yang menyediakan positioning, kecepatan dan ketepatan waktu ke pengguna. Bagian ini menerima data dari satelit-satelit melalui sinyal radio yang dikirimkan setelah mengalami koreksi oleh stasiun pengendali (*GPS Control Segment*[5]).

2.2.3.3 Fungsi dan Kegunaan GPS

2.2.3.3.1 Navigasi

Dalam kebutuhan berkendara sistem GPS pun sangat membantu, dengan adanya *GPS Tracker* terpasang pada kendaraan maka akan membuat perjalanan semakin nyaman karena arah dan tujuan jalan bisa diketahui setelah GPS mengirim posisi kendaraan kita yang diterjemahkan ke dalam bentuk peta digital[5].

2.2.3.3.2 Geografis

GPS sering juga digunakan untuk keperluan sistem informasi geografis, seperti untuk pembuatan peta, mengukur jarak perbatasan, atau bisa dijadikan sebagai referensi pengukuran suatu wilayah[5].

Teknologi Gps dalam aplikasi renovin digunakan untuk menentukan tukang terdekat yang bermanfaat untuk mempermudah mencari tukang terdekat berdasarkan lokasi pengguna.

2.2.4 API

Application Programming Interface (API) Android API adalah Seperangkat fungsi standar yang disediakan oleh OS atau Bahasa. Dalam Java, API dimasukkan ke dalam *package-package* yang sesuai dengan fungsinya. Berikut adalah beberapa API utama yang disediakan oleh *Android*, yaitu API untuk manipulasi *Graphical User Interface (GUI)*, akses *storage*, manipulasi grafik, akses *location based service*, dan manipulasi peta.

1. *Graphical User Interface (GUI) Package android.view* menyediakan berbagai kelas-kelas yang akan digunakan untuk menangani *screen, layout*, dan interaksinya dengan pengguna.
2. Akses *Storage Android* menggunakan mekanisme storage yang berbeda dengan sistem operasi yang konvensional dimana setiap file dalam *Android* bersifat *private* terhadap aplikasi tersebut.
3. Manipulasi Grafik *Package android.graphics* menyediakan manipulasi grafik *low-level* seperti kanvas, point, pewarnaan, dan manipulasi bentuk pada *screen*.
4. Manipulasi Peta *Package com.google.android.maps* menyediakan API untuk mengakses Google Maps.
5. Akses *Location-based Service Package android.location* berisi kelas-kelas untuk mengakses berbagai layanan berbasis lokasi[6].

Aplikasi Renovin menggunakan API bukalapak untuk mengambil data harga bahan material dan API google maps untuk mengetahui lokasi tukang terdekat.

Adapun cara kerja Api bukalapak pada aplikasi renovin adalah Api bukalapak akan mengirimkan request pada server bukalapak agar mengirimkan data bahan material sesuai request pengguna.

Adapun cara kerja *Google direction API* pada aplikasi renovin adalah Gps akan mengirimkan data lokasi tukang terdekat menurut *request* dari pengguna, sehingga pengguna dapat mengetahui lokasi tukang terdekat.

Google Directions API adalah layanan dari google yang memudahkan kita (*developer*) untuk mencari rute dan navigasi dari satu tempat ke tempat tertentu. Kita tinggal memasukkan latitude dan longitude posisi berangkat dan juga latitude longitude posisi tujuan. Keunggulan dari API ini adalah dia mudah digunakan, kita hanya tinggal melakukan HTTP Request untuk memanggil *Google Directions API*. Selain mudah, dia juga menyediakan banyak moda transportasi, setiap moda transportasi bisa saja memiliki rute tersendiri dan waktu tempuh tersendiri (misal, orang berjalan bisa melawan arus di jalan raya sedangkan mobil tidak, mobil harus memutar jika tempat yang seharusnya didatangi terlewati sedangkan jalan hanya satu arah)

Fasilitas ini bisa kita gunakan secara gratis tetapi mempunyai batasan sebagai berikut:

- 2500 *request* dalam 24 jam
- Mode transit dihitung 4 request
- Kita bisa menggunakan sampai 8 *waypoint* dari setiap *request*

Jika aplikasi kita sangatlah besar dan 2500 request tidaklah cukup, kita bisa menggunakan *Google Maps API for Bussiness*.

2.2.5 Data Koefisien Bahan Material Bangunan

Koefisien adalah jumlah bahan material yang dibutuhkan dalam satuan yang telah ditentukan. Misal untuk 1 m² pengecatan tembok baru di butuhkan :

- a. Bahan Plamur 0,10 Kg
- b. Bahan Cat Dasar 0,10 Kg
- c. Bahan Cat Penutup 0.26 Kg

Angka 0,10 pada bahan plamur adalah koefisien yang telah di tentukan oleh badan Penelitian pekerjaan umum di indonesia begitupun bahan cat dasar dan bahan cat penutup angka ini sudah menjadi standar nasional atau (SNI). Jadi dalam pekerjaan cat tembok baru di butuhkan :

- a. Bahan Plamur 0,10 Kg
- b. Bahan Cat Dasar 0,10 Kg
- c. Bahan Cat Penutup 0.26 Kg

Masih banyak lagi pekerjaan lainnya, adapun data pekerjaan atau Analisa Hasil Standar Pekerjaan (AHSP) yang digunakan didalam aplikasi renovin adalah sebagai berikut :

1. Pekerjaan Cat

Pengecatan 1 m² tembok baru. 1 Lapis cat Plamuur, 1 Lapis cat dasar, 2 Lapis cat penutup)

Tabel 2.1 Standar Pekerjaan Cat Tembok Baru [10].

No	Bahan	Satuan	Koefisien
1	Plamur	Kg	0,10
2	Cat dasar	Kg	0,10
3	Cat penutup	Kg	0,26

Pengecatan 1 m² tembok lama. (1 Lapis cat dasar, 2 Lapis cat penutup)

Tabel 2.2 Standar Pekerjaan Cat Tembok Baru [10].

No	Bahan	Satuan	Koefisien
1	Cat dasar	Kg	0,12
2	Cat Penutup	Kg	0,18

2. Pekerjaan Walpaper.

Pemasangan 1 m² Walpaper.

Tabel 2.3 Standar Pekerjaan Walpaper [10].

No	Bahan	Satuan	Koefisien
1	Walpaper	m ²	1,20
2	Lem	Kg	0,20

3. Pekerjaan Ubin

Pemasangan 1 m² Lantai ubin PC abu-abu 20cm X 20cm

Tabel 2.4 Standar Pekerjaan Ubin Pc Abu-abu 20xmX20cm [10].

No	Bahan	Satuan	Koefisien
1	Ubin abu-abu	Bh	26,50
2	Semen Portlan	Kg	10,40
3	Pasir pasang	m ³	0,045

Pemasangan 1 m² Lantai ubin PC warna 20cm X 20cm

Tabel 2.5 Standar Pekerjaan Ubin Pc Warna 20xmX20cm.[10]

No	Bahan	Satuan	Koefisien
1	Ubin abu-abu	Bh	26,50
2	Semen Portlan	Kg	10,40
3	Pasir pasang	m ³	0,045
4	Semen Warna	Kg	1,62

4. Pekerjaan Kaca

Pemasangan 1 m² Kaca tebal 3mm

Tabel 2.6 Standar Pekerjaan Kaca Tebal 3mm [10].

No	Bahan	Satuan	Koefisien
1	Kacat tebal 3mm	m ²	1,10
2	Sealant	Kg	0,05

Pemasangan 1 m² Kaca tebal 5mm

Tabel 2.7 Standar Pekerjaan Kaca Tebal 5mm [10].

No	Bahan	Satuan	Koefisien
1	Kacat tebal 3mm	m ²	1,10
2	Sealant	Kg	0,05

Kalkulasi pekerjaan merupakan fitur yang memudahkan pengguna umum untuk mengkalkulasi penggunaan bahan material berdasarkan analisa harga satuan pekerjaan (AHSP) yang di publikasikan oleh BALITBANG PU hal ini ditujukan

agar pengguna umum dapat mengetahui estimasi pengguna bahan material meliputi pekerjaan Cat tembok, Pekerjaan Ubin, Pekerjaan Walpaper dan Pekerjaan Kaca.

Adapun rumus untuk kalkulasi pekerjaan adalah

$$\text{Luas bidang} * \text{koefisien} = \text{Hasil kalkulasi [10]}$$

2.2.6 Object Oriented Programming (OOP)

Booch menyatakan “pemrograman berorientasi objek adalah metode implementasi di mana program diorganisasikan sebagai kumpulan objek yang bekerja sama. Masing-masing objek merepresentasikan instan dari kelas, dan kelas-kelas itu anggota suatu hirarki kelas-kelas yang disatukan lewat keterhubungan pewarisan”. Stroustrup mendeskripsikan pemrograman berorientasi objek sebagai penggabungan abstraksi data dengan pewarisan. Fitur-fitur penting untuk mendukung abstraksi data adalah *constructor* serta *destructor* untuk penciptaan dan pemusnahan objek. Tipe berparameter, mekanisme *exception*, *object coercions* dan iterator. dalam penelitian ini menggunakan *oop* karena memiliki keunggulan-keunggulan. Keunggulan pendekatan berorientasi objek adalah sebagai berikut:

1. Bekerja yang mendekati kognisi manusia.
2. Menghasilkan sistem yang dibangun di atas bentuk-bentuk antara yang stabil. Dan dengan demikian lebih mampu untuk mengikuti perubahan.
3. Dapat digunakan tidak hanya pada perancangan perangkat lunak tapi juga seluruh proses pengembangan perangkat lunak.
4. Mereduksi resiko pengembangan sistem-sistem kompleks terutama karena pemaduan terjadi menyebar pada siklus kehidupan tidak terjadi dalam sekejap sebagaimana kejadian ledakan besar (*big bang*)[8].

Proses pengembangan aplikasi renovasi di bangunan menggunakan OOP dikarenakan OOP menyediakan struktur modular yang jelas untuk program sehingga OOP sangat bagus digunakan untuk mendefinisikan tipe data abstrak di mana detail implementasinya tersembunyi. OOP akan mempermudah dalam memaintain dan memodifikasi kode yang sudah ada. Objek yang baru dapat dibuat tanpa mengubah kode yang sudah ada. OOP menyediakan *framework* untuk *library* kode di mana

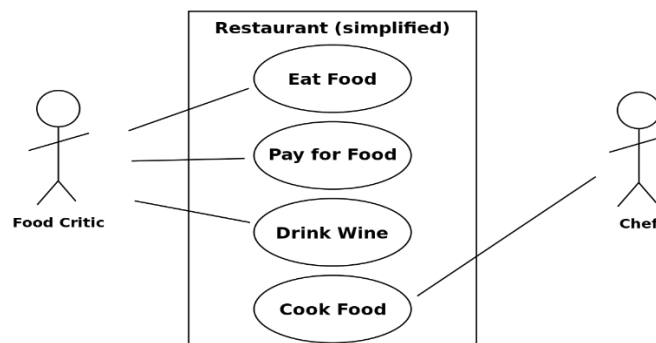
komponen software yang tersedia dapat dengan mudah diadaptasi dan dimodifikasi oleh programmer.

2.2.7 *Unified Modeling Language (UML)*

Diagram *use case* adalah diagram yang menunjukkan fungsionalitas suatu sistem atau kelas dan bagaimana sistem tersebut berinteraksi dengan dunia luar, dan menjelaskan sistem secara fungsional yang terlihat *user*. Biasanya dibuat pada awal pengembangan. *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. *Use case* merupakan sebuah pekerjaan tertentu, misalnya *login* ke sistem, membuat sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan system untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua feature yang ada pada sistem. Sebuah *use case* dapat *use case* memanggil (*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 memanggil (*include*) dieksekusi secara normal. Sebuah *use case* dapat dipanggil (*include*) oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang common. Sebuah *use case* juga dapat meng-*extend* *use case* lain dengan behaviour-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain[9].

Aplikasi renovin menggunakan UML dikarenakan Pengembang cukup menggunakan 1 metodologi dari tahap analisis hingga perancangan. Perancangan komponen antarmuka terintegrasi dengan perancangan PL dan struktur data

2.2.7.1 Diagram *Use case* (*Use case diagram*)



Gambar 2.4 Contoh *Use case*[9].

Diagram *use case* merupakan salah satu diagram untuk memodelkan aspek perilaku sistem. *Use case* adalah interaksi antara *actor eksternal* dan sistem digunakan untuk mendeskripsikan apa yang seharusnya dilakukan oleh sistem. Diagram *use case* menyediakan cara mendeskripsikan pandangan eksternal terhadap sistem dan interaksi-interaksinya dengan dunia luar. Elemen-elemen yang ada pada diagram *use case* adalah aktor, *use case*, dan hubungan ketergantungan, generalisasi dan asosiasi[9].

a. Aktor

Aktor adalah pemakai sistem, dapat berupa manusia atau sistem terotomatisasi lain. Aktor adalah sesuatu atau seseorang yang berinteraksi dengan sistem, yaitu siapa atau apa yang menggunakan sistem. Aktor berkomunikasi dengan sistem lewat pengiriman dan penerimaan pesan. *Use case* selalu diawali oleh aktor yang mengirim pesan[9].

b. *Use case*

Use case adalah cara spesifikasi penggunaan sistem oleh aktor. *Use case* menspesifikasikan perilaku sistem atau bagian sistem dan merupakan deskripsi sekumpulan sekuen aksi termasuk varian-varian yang dilakukan sistem untuk memproduksi hasil atau nilai ke actor[9].

c. Hubungan Antar *Use case*

Keterhubungan antar *use case* dengan *use case* lain berupa generalisasi antara *use case*, yaitu include dan extend. *Include* adalah perilaku *use case*

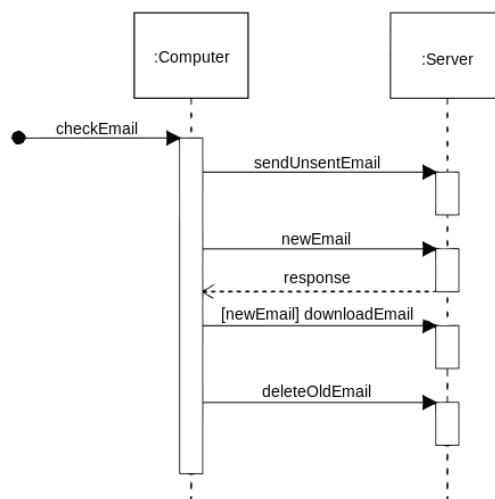
merupakan bagian dari *use case* yang lain. Sedangkan *extend* adalah perilaku *use case* merupakan perilaku *use case* yang lain.

Adapun level dalam usecase diantara lain sebagai berikut :

Sea level khususnya mewakili sebuah interaksi diskrit antara aktor utama dan sistem.

Fish level *use case* yang ada disana hanya karena telah masuk *use case sea level*[9].

2.2.7.2 Diagram Sekuensial (*Sequence Diagram*)

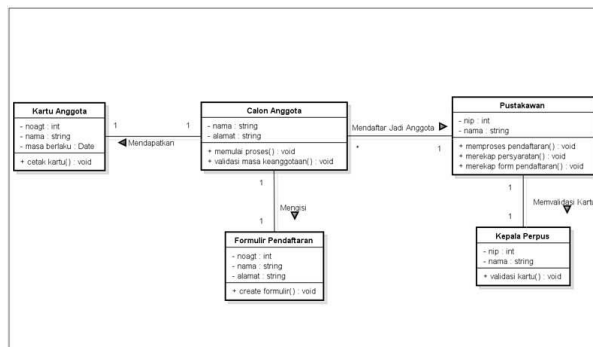


Gambar 2.5 *Sequence Diagram*[9].

Pada penelitian ini sekuensial diagram digunakan untuk memodelkan skenario penggunaan. Penggunaan adalah barisan kejadian yang terjadi selama satu eksekusi sistem. Diagram sekuean menunjukkan objek sebagai garis vertikal dan tiap kejadian sebagai panah horisontal dari objek pengirim ke objek

penerima. Waktu berlalu dari atas ke bawah dengan lama waktu tidak relevan . Diagram sekuensial ini juga akan menggambarkan *behavior* yang ada pada aplikasi. Oleh sebab itu diagram sekuensial ini digunakan untuk menggambarkan bagaimana antiitas dalam sistem berinteraksi[9].

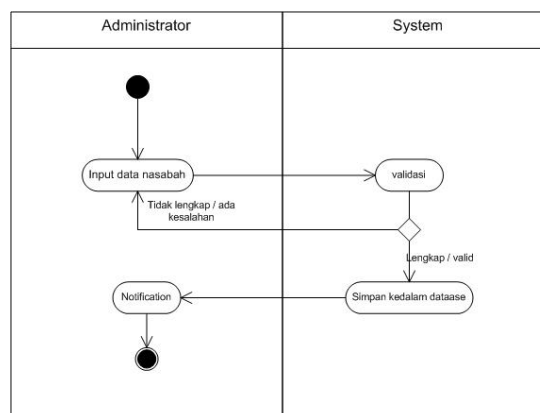
2.2.7.3 Diagram Kelas (*Class Diagram*)



Gambar 2.6 Class Diagram[9].

Class diagram adalah model statis yang menggambarkan struktur dan deskripsi class serta hubungannya antara *class*. *Class diagram* mirip ER-Diagram pada perancangan *database*, bedanya pada ER-diagram tidak terdapat operasi/*methode* tapi hanya atribut. *Class* terdiri dari nama kelas, atribut dan operasi/*methode*[9].

2.2.7.4 Diagram Aktivitas (*Activity diagram*)



Gambar 2.7 Activity diagram[9].

Diagram aktivitas pada umumnya adalah diagram *flowchart* yang diperluas yang menunjukkan aliran kendali satu aktivitas ke aktivitas lain. Diagram aktivitas berfokus pada aktivitas-aktivitas, potongan-potongan dari proses yang boleh jadi berkorespondensi dengan metode – metode atau fungsi-fungsi anggota dan pengurutan dari aktivitas-aktivitas ini. Elemen-elemen yang ada pada diagram aktivitas adalah *activity state* dan *action state*, transisi, dan objek. Pada penelitian ini diagram aktifitas digunakan untuk menggambarkan alur sistem dari awal hingga proses berakhir[9].

2.2.8 **Blackbox Testing**

Black Box Testing berfokus pada spesifikasi fungsional dari perangkat lunak. Tester dapat mendefinisikan kumpulan kondisi *input* dan melakukan pengetesan pada spesifikasi fungsional program. *Black Box Testing* bukanlah solusi alternatif dari *White Box Testing* tapi lebih merupakan pelengkap untuk menguji hal-hal yang tidak dicakup oleh *White Box Testing*.

1. Fungsi yang salah atau hilang.
2. Kesalahan pada *interface*.
3. Kesalahan pada struktur data atau akses *database*.
4. Kesalahan *performasi*.
5. Kesalahan *inisialisasi* dan tujuan akhir

Pengujian *blackbox* di lakukan pada pembangunan perangkat lunak ini untuk mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari perangkat lunak, mengevaluasi *User Interface* dan fungsionalitasnya[7].

2.2.9 **Pengujian Beta**

Pengujian beta dianggap sebagai bentuk pengujian penerimaan perangkat lunak oleh pengguna eksternal. Dibagikan kepada pengguna yang berada di luar tim pemrogram sebagai penguji beta. Perangkat lunak dibagikan ke sekelompok orang sehingga pengujian bisa memastikan bahwa produknya memiliki kesalahan atau bug yang kecil[7].

2.2.10 Skala *likert*

Skala *likert* adalah metode yang digunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau kelompok orang tentang fenomena sosial. Dengan skala *likert*, variabel yang akan diukur dijabarkan menjadi indikator variabel. Selanjutnya indikator tersebut dijadikan sebagai titik tolak untuk menyusun item-item instrumen pernyataan atau pertanyaan. Instrumen penelitian dapat dibuat dalam bentuk checklist ataupun pilihan ganda.

Jawaban setiap item instrumen yang menggunakan skala *likert* mempunyai gradasi dari sangat positif sampai sangat negatif dimana jawaban yang diberikan akan diberikan skor. Jawaban dapat berupa kata-kata sebagai berikut[7]:

- a. Sangat setuju dengan skor 5
- b. Setuju dengan skor 4
- c. Ragu-ragu dengan skor 3
- d. Tidak setuju dengan skor 2
- e. Sangat tidak setuju dengan skor 1

Langkah-langkah skala likert adalah :

1. Menghitung jumlah skor ideal (kriterium) yaitu dengan rumus :
Kriterium = nilai bobot maksimal X jumlah responden
2. Menghitung jumlah jawaban dari responden dalam bentuk persentase, digunakan rumus :

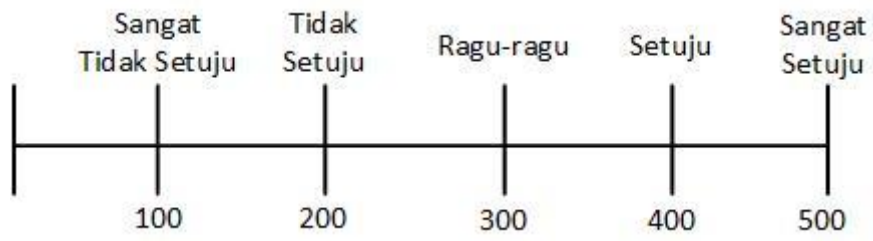
$$P = \frac{\text{total nilai}}{\text{kriterium}} \times 100\%$$

Keterangan :

P : nilai persentasi yang dicari

Total nilai : jumlah frekuensi X nilai bobot

3. Selanjutnya skor dimasukkan kedalam bentuk *Interval Rating Scale*



Gambar 2.8 *Interval Rating Scale*[7].