BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Kajian Terkait

Penelitian Muhammad Fajri Saparianto (2019), dalam Jurnal yang berjudul Aplikasi Pemesanan Tukang Berbasis *Mobile*, menjelaskan bahwa tujuan dari dibuatnya penelitian ini membuat sebuah aplikasi yang dapat digunakan untuk memesan tukang dengan kriteria yang diinginkan berdasarkan jangkauan lokasi sang tukang dengan lokasi pengguna.

Penelitian Risnu Arya Kusuma (2020), yang berjudul Aplikasi Peringatan Rambu Lalu Lintas Dengan Metode *Location Based Service* Berbasis *Mobile*. Penelitian ini menjelaskan tentang pembuat sebuah aplikasi peringatan berupa notifikasi suara mengenai rambu-rambu lalu lintas yang akan dilewati oleh pengguna kendaraan. Dengan tujuan pengguna dapat selalu menaati rambu-rambu lalu lintas yang tersebar di segala penjuru jalan di kota Pontianak.

Penelitian Igo Suparius Ipo (2021), yang berjudul Rancang Bangun Sistem Informasi Geografis Persebaran Bengkel, Tambal Ban dan Minyak Eceran di Kota Pontianak Berbasis *Web*. Sebuah penelitian yang dilakukan untuk membuat sebuah sistem informasi persebaran bengkel, tambal ban dan minyak eceran di kota Pontianak guna memudahkan pengguna jalan yang sedang dalam kesulitan karena beberapa masalah yang sudah diteliti. Penelitian yang menggunakan metode *Location Based Service* sehingga dapat membantu pengguna dalam mencari kebutuhan yang mereka cari seusai jarak terdekat dari lokasi mereka. Kajian terkait dapat dilihat pada tabel 2.3.

Tabel 2. 1 Perbandingan Penelitian

No.	Penulis	Judul	Keterangan
1.	Muhammad Fajri Saparianto (2019). Universitas	Aplikasi Pemesanan Tukang	Sistem ini menggunakan metode *Location Based Service* untuk menentukan jarak terdekat tukang dengan lokasi pembangunan

No.	Penulis	Judul	Keterangan
	Tanjungpura Pontianak	Berbasis *Mobile*	Pengujian sistem menggunakan metode *black box*, *kuesioner* dan pengujian *customer interface (UI)* aplikasi.
			Output dalam aplikasi ini adalah penampilan *data* tukang yang diinginkan oleh *customer.*
2.	Risnu Arya Kusuma (2020). Universitas Tanjungpura Pontianak	Aplikasi Peringatan Rambu Lalu Lintas Dengan Metode *Location Based Service* Berbasis *Mobile*	Sistem menggunakan metode *Location Based Service* untuk mengetahui lokasi pengguna dengan rambu-rambu lalu lintas terdekat.
			Pengujian sistem menggunakan metode *black box*, kuesioner dan pengujian *customer interface* (UI) aplikasi
			Hasil dari penelitian berupa notifikasi *output* suara yang memberitahukan letak rambu lalu lintas terdekat dan keterangan rambu tersebut serta hal yang harus dilakukan.
3.	Igo Suparius Ipo (2021). Universitas Tanjungpura Pontianak.	Rancang Bangun Sistem Informasi Geografis Persebaran Bengkel, Tambal Ban dan Minyak	Sistem ini menggunakan metode *Location Based Service* untuk mengetahui lokasi bengkel, tambal ban serta minyak eceran terdekat dari posisi *customer*.
			Sistem dibangun berbasis *website* sehingga dapat diakses oleh semua perangkat yang terhubung ke *internet* dan memiliki *browser*.

No.	Penulis	Judul	Keterangan
		Eceran di Kota Pontianak Berbasis *Web*	Hasil penelitian ini adalah mempermudah para pengguna jalan yang sedang mengalami masalah dengan kendaraan mereka di masa dan waktu yang genting sehingga tidak perlu khawatir dengan status waktu dan jam buka pihak bengkel, tambal ban dan penyedia minyak eceran.

Tabel 2. 2 Penelitian yang Akan Dilakukan

No.	Penulis	Judul	Keterangan
1.	Royhan Restiandi (2021), Universitas Tanjungpura Pontianak	Aplikasi Jasa Ojek *Pick-up* Berbasis Lokasi Geospasial dan Google *Map*s dengan Metode *Location Based Service*	Aplikasi menggunakan metode *Location Based Service* untuk menentukan jarak terdekat antara pengguna dan penyedia jasa ojek *pick-up*.
			Pengujian sistem akan menggunakan metode *black box*, kuesioner dan pengujian *customer interface* (UI) aplikasi.
			Output dari aplikasi ini adalah terciptanya sebuah aplikasi yang dapat memudahkan pengguna dalam mencari informasi tentang penyedia jasa ojek *pick-up* terdekat dari posisi saat ini berada sesuai kriteria yang diinginkan.

No.	Penulis	Judul	Keterangan
			Pengguna dapat menggunakan aplikasi untuk memesan ojek *pick-up* tanpa harus pergi keluar rumah.

2.2 Sistem Informasi Geografis

Aronoff dalam Adil (2017) berpendapat "Sistem Informasi Geografis adalah suatu sistem berbasis komputer yang memiliki kemampuan dalam menangani *data* bereferensi geografi, yaitu memasukkan *data*, manajemen *data* (penyimpanan dan pemanggilan kembali), manipulasi dan analisis *data*, serta keluaran sebagai hasil akhir (*output*). Hasil akhir (*output*) dapat dijadikan acuan dalam pengambilan keputusan pada masalah yang berhubungan dengan geografi.

Gistut dalam Adil (2017) mengatakan bahwa "Sistem Informasi Geografis (SIG) adalah sistem yang dapat mendukung pengambilan keputusan spasial dan mampu mengintegrasikan deskripsi-deskripsi lokasi dengan karakteristik-karakteristik fenomena yang ditemukan di lokasi tersebut. SIG yang lengkap mencakup metodologi dan teknologi yang diperlukan, yaitu *data* spasial, perangkat keras, perangkat lunak, dan struktur organisasi.

2.2.1 *Komponen Sistem Informasi Geografis*

John E. Harmon dan Steven J. Anderson dalam Adil (2017) menyatakan secara rinci SIG dapat beroperasi dengan komponen-komponen sebagai berikut:

a) Pengguna: orang yang menjalankan sistem, meliputi orang yang mengoperasikan, mengembangkan, bahkan memperoleh manfaat dari sistem. Kategori orang yang menjadi bagian dari SIG beragam, misalnya *operator*, analis, *programmer*, *database admin*istrator, bahkan *stakeholder*.
b) Aplikasi: prosedur yang digunakan untuk mengolah *data* menjadi informasi. Misalnya penjumlahan, klasifikasi, rotasi, koreksi geometri, *query*, *overlay*, *buffer*, *join table*, dan sebagainya.
c) *Data*: *data* yang digunakan dalam SIG dapat berupa grafis dan *data* atribut.
    1. *Data* posisi/koordinat/grafis/ruang/spasial: merupakan *data* yang merupakan representasi fenomena permukaan bumi/keruangan yang

memiliki referensi (koordinat) lazim berupa peta, foto udara, citra satelit, dan sebagainya atau hasil dari interpretasi *data-data* tersebut.

2. *Data* atribut/non spasial: *data* yang merepresentasikan aspek-aspek deskriptif dari fenomena yang dimodelkannya. Misalnya *data* sensus penduduk, catatan survey, *data statistic* lainnya.

d) Software: perangkat lunak SIG berupa *program* aplikasi yang memiliki kemampuan pengelolaan, penyimpanan, pemrosesan, analisis, dan penayangan *data* spasial (contoh: ArcView, Idrisi, ARC/INFO, ILWIS, MapInfo, dan lain-lain).

e) Hardware: perangkat keras yang dibutuhkan untuk menjalankan sistem berupa perangkat komputer, *Central Processing Unit* (CPU), *printer*, *scanner*, *digitizer*, *plotter*, dan perangkat pendukung lainnya.

2.2.2 *Sumber Data Spasial dan Non-Spasial*

SIG membutuhkan masukan *data* yang bersifat spasial maupun deskriptif. Beberapa sumber *data* tersebut antara lain adalah:

1. Peta *analog* (peta topografi, peta tanah, dan sebagainya).
   Peta *analog* adalah peta dalam bentuk cetakan. Pada umumnya peta *analog* dibuat dengan teknik kartografi, sehingga sudah mempunyai referensi spasial seperti koordinat, skala, arah mata angin dan sebagainya. Peta *analog* dikonversi menjadi peta digital dengan berbagai cara. Referensi spasial dari peta *analog* memberikan koordinat sebenarnya di permukaan bumi pada peta digital yang dihasilkan. Biasanya peta *analog* direpresentasikan dalam format vektor.
2. *Data* dari sistem Penginderaan Jauh (citra satelit, foto-udara, dan sebagainya).
   Data Penginderaan Jauh dapat dikatakan sebagai sumber *data* yang terpenting bagi SIG karena ketersediaannya secara berkala. Adanya bermacam-macam satelit di ruang angkasa dengan spesifikasinya masing-masing, kita bisa menerima berbagai jenis citra satelit untuk berbagai tujuan pemakaian. *Data* ini biasanya direpresentasikan dalam *format* raster.
3. *Data* hasil pengukuran lapangan.

Contoh *data* hasil pengukuran lapangan adalah *data* batas *admin*istrasi, batas kepemilikan lahan, batas persil, batas hak pengusahaan hutan, dan sebagainya yang dihasilkan berdasarkan teknik perhitungan tersendiri. Pada umumnya *data* ini merupakan sumber *data* atribut.

4. *Data* GPS (*Global Positioning System*)

   Global Positioning System (Sistem Pencari Posisi *Global*), adalah suatu jaringan satelit yang secara terus menerus memancarkan sinyal radio dengan *frekuensi* yang sangat rendah. Alat penerima GPS secara pasif menerima sinyal ini, dengan syarat bahwa pandangan ke langit tidak boleh terhalang, sehingga biasanya alat ini hanya bekerja di ruang terbuka. Satelit GPS bekerja pada referensi waktu yang sangat teliti dan memancarkan *data* yang menunjukkan lokasi dan waktu pada saat itu. Operasi dari seluruh satelit GPS yang ada di sinkronisasi sehingga memancarkan sinyal yang sama. Alat penerima GPS akan bekerja jika ia menerima sinyal dari sedikitnya 4 buah satelit GPS, sehingga posisinya dalam tiga dimensi bisa dihitung. Pada saat ini sedikitnya ada 24 satelit GPS yang beroperasi setiap waktu dan dilengkapi dengan beberapa cadangan. Satelit tersebut dioperasikan oleh Departemen Pertahanan Amerika Serikat, megorbit selama 12 jam (dua orbit per hari) pada ketinggian sekitar 11.500 mil dan bergerak dengan kecepatan 2000 mil per jam.

2.3 Google Maps

Google *Maps* adalah layanan aplikasi peta *online* yang disediakan oleh Google secara gratis. Layanan peta Google *Maps* secara resmi dapat diakses melalui situs http://maps.google.com. Pada situs tersebut dapat dilihat informasi geografis pada hampir semua permukaan di bumi kecuali daerah kutub utara dan selatan. Layanan ini dibuat sangat interaktif, karena di dalamnya peta dapat digeser sesuai keinginan pengguna, mengubah level *zoom*, serta mengubah tampilan jenis peta.

Google *Maps* mempunyai banyak fasilitas yang dapat dipergunakan misalnya pencarian lokasi dengan memasukkan kata kunci, kata kunci yang dimaksud seperti nama tempat, kota, atau jalan, fasilitas lainnya yaitu perhitungan rute perjalanan dari satu tempat ke tempat lainnya (Siswanto, 2012).

2.4 Google Maps API

API atau *Application Programming Interface* merupakan suatu dokumentasi yang terdiri dari *interface*, fungsi, kelas, struktur dan sebagainya untuk membangun sebuah perangkat lunak. Dengan adanya API ini, maka memudahkan *programmer* untuk "membongkar" suatu *software* untuk kemudian dapat dikembangkan atau diintegrasikan dengan perangkat lunak yang lain. API dapat dikatakan sebagai penghubung suatu aplikasi dengan aplikasi lainnya yang memungkinkan *programmer* menggunakan sistem *function*. Proses ini dikelola melalui *operating system*. Keunggulan dari API ini adalah memungkinkan suatu aplikasi dengan aplikasi lainnya dapat saling berhubungan dan berinteraksi. Bahasa pemrograman yang digunakan oleh Google *Maps* yang terdiri dari *HTML*, *Javascript* dan *AJAX* serta *XML*, memungkinkan untuk menampilkan peta Google *Maps* di *website* lain.

Google juga menyediakan layanan Google *Maps* API yang memungkinkan para pengembang untuk mengintegrasikan Google *Maps* ke dalam *website* masing-masing dengan menambahkan *data point* sendiri. Dengan menggunakan Google *Maps* API, Google *Maps* dapat ditampilkan pada *website* eksternal. Agar aplikasi Google *Maps* dapat muncul di *website* tertentu, diperlukan adanya API *key*. API *key* merupakan kode unik yang digenerasikan oleh google untuk suatu *website* tertentu, agar *server* Google *Maps* dapat mengenali. (Siswanto, 2013)

2.5 Google Direction API

Google *Direction* API adalah layanan untuk menghitung arah/jarak antar lokasi menggunakan permintaan *HTTP*. Arah dapat dicari beberapa *mode* transportasi, termasuk transit, mengemudi, berjalan atau bersepeda. Untuk menggunakan *Direction* API dalam aplikasi android perlu mendapatkan kunci Direction API dengan cara yang sama seperti mendapatkan Google *Map*s API key (Doshi, dkk, 2014).

2.6 Google Distance Matrix API

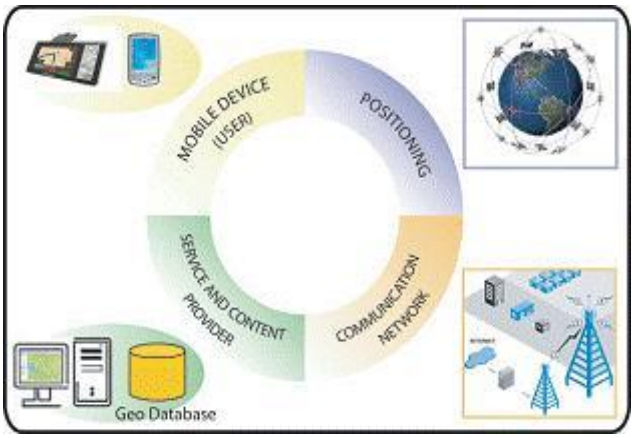
Distance Matrix API adalah layanan yang menyediakan jarak dan waktu tempuh untuk matriks lokasi asal dan tujuan. Berbeda dengan Directions API,

Distance Matrix API tidak memberikan informasi rute perjalanan, namun hanya jarak dan waktu tempuh saja yang secara umum sering digunakan oleh pengguna. Contoh implementasi Distance Matrix API dalam kehidupan sehari-hari biasanya digunakan untuk melakukan persebaran titik-titik lokasi toko atau kendaraan terdekat.

2.7 Location Based Service (LBS)

Location Based Service (LBS) atau Layanan Berbasis Lokasi merupakan layanan informasi yang memanfaatkan kemampuan untuk menggunakan informasi lokasi dari perangkat bergerak dan dapat diakses dengan perangkat bergerak melalui jaringan telekomunikasi bergerak (Steiniger, 2006).

Dalam Layanan Berbasis Lokasi terdapat empat komponen penting seperti terlihat pada Gambar 2.1.



Gambar 2. 1 Komponen Dasar LBS

Setiap komponen mempunyai fungsi :

1. *Mobile Devices*, merupakan suatu alat yang digunakan oleh pengguna untuk meminta informasi yang dibutuhkan. Informasi dapat diberikan dalam bentuk suara, gambar, dan *text*.
2. *Communication Network*, komponen ini mengirim *data* pengguna dan informasi yang diminta dari *mobile* terminal ke *Service Provider* kemudian mengirimkan kembali informasi yang diminta ke pengguna. *Communication network* dapat berupa jaringan seluler (GSM, CDMA), *Wireless Local Area Network* (WLAN), atau *Wireless Wide Area Network* (WWAN).

3. *Positioning Component*, digunakan untuk memproses suatu layanan maka posisi pengguna harus diketahui.
4. *Service and Application Provider*, penyedia layanan menawarkan berbagai macam layanan kepada pengguna dan bertanggungjawab untuk memproses informasi yang diminta oleh pengguna.

Data and Content Provider, penyedia layanan tidak selalu menyimpan semua *data* yang dibutuhkan yang bisa diakses oleh pengguna.

2.8 Firebase

Firebase adalah suatu layanan dari Google untuk memberikan kemudahan bahkan mempermudah para *developer* aplikasi dalam mengembangkan aplikasinya. Firebase alias BaaS (*Backend as a Service*) merupakan solusi yang ditawarkan oleh Google untuk mempercepat pekerjaan *developer*. Dengan menggunakan Firebase, *apps developer* bisa fokus dalam mengembangkan aplikasi tanpa memberikan *effort* yang besar untuk urusan *backend*. Singkat cerita mengenai sejarah dari Firebase didirikan pertama kali pada tahun 2011 oleh Andrew Lee dan James Tamplin. Produk Firebase yang pertama kali adalah *Realtime Database*. *Realtime Database* digunakan developer untuk menyimpan *data* dan *synchronize* ke banyak *customer*. Kemudian ia berkembang sebagai layanan pengembang aplikasi. Pada bulan Oktober 2014, perusahaan tersebut diakuisisi oleh Google (Dicoding Intern, 2020).

2.9 Framework7

Framework7 adalah sebuah *framework* gratis dan *open source* untuk mengembangkan aplikasi seluler, *desktop*, atau *web* dengan tampilan dan nuansa asli. Ini juga merupakan alat *prototyping* yang sangat diperlukan untuk menunjukkan *prototipe* aplikasi yang berfungsi sesegera mungkin jika perlu. Dengan seperangkat komponen UI yang menakjubkan yang disediakan Framework7 langsung dari kotak, memungkinkan untuk membuat aplikasi *web*, aplikasi *web progresif* (PWA) dan aplikasi iOS dan Android dengan tampilan dan nuansa asli (Vladimir Kharlampidi. 2022).

2.10 Web Service

Web service merupakan halaman *web* dan dokumen dalam *HTML* yang direnderer secara grafis oleh *browser* anda, sehingga anda dapat memindai halaman secara *visual* untuk mencari *formular* (L Richardson, S Ruby - 2008). Adapun aplikasi terdistribusi tersebut dapat diakses oleh aplikasi-aplikasi client tanpa memperhatikan sistem operasi maupun bahasa pemrograman. Sebelum adanya *web service* terdapat teknologi CORBA dari OMG yang menggunakan bahasa Java dan DCOM dari Microsoft. Kekurangan yang dimiliki oleh kedua teknologi ini adalah *program* yang akan dipakai untuk mengakses komponen tersebut harus dibuat dengan bahasa yang sama dengan bahasa yang dipakai untuk membuat komponen tersebut untuk CORBA dan untuk DCOM cuma bisa di *platform* Microsoft.

2.11 Android

Android secara sederhana bisa diartikan sebagai sebuah *software* yang digunakan pada perangkat mobile yang mencakup sistem operasi, *middleware*, dan aplikasi kunci yang dirilis oleh Google (EMS Tim, 2015). Pada awal peluncurannya, Google meyakini bahwa *platform* perangkat *mobile* Android memiliki kesempatan yang sangat besar dalam pengembangan aplikasi. Google mengumumkan *Open Handset Alliance* (OHA) dan *platform* Android pada November 2007, dan meluncurkan Android *Software Development Kit* (SDK) pertama yang masih dalam versi *beta* di waktu yang sama. Dalam waktu yang tidak lama, lebih dari satu juta orang mengunduh Android SDK dari *website* 14 Google. Di Amerika Serikat, *T-Mobile* mengumumkan perangkat *mobile* Android bergelar G1 pada Oktober 2008, dan diperkirakan ratusan ribu perangkat G1 terjual pada akhir tahun yang sama. Android memiliki potensi yang besar untuk menghilangkan batasan dan kendala yang selama ini muncul dalam mengembangkan suatu perangkat lunak versi *mobile phone*.

2.12 Progresive Web App

Progressive Web App merupakan penggabungan yang terbaik dari aplikasi web dan seluler sehingga memberikan pengalaman yang kaya seperti aplikasi asli.

Ini adalah situs *web* yang dibangun menggunakan teknologi *web* yang berfungsi seperti aplikasi. PWA adalah situs *web* yang dibangun menggunakan teknologi *web* yang berfungsi seperti aplikasi dan tidak perlu diinstal seperti aplikasi asli (S Tandel, A Jamadar, 2018). Aplikasi *web* yang beroperasi seperti aplikasi native dengan UX dan notifikasi menyerupai *program* tersebut pula. Kemudian, PWA mengusung konsep *offline first* dan *Web* APIs teranyar, sehingga tak akan memunculkan pesan terkait *network error* maupun *white screen* yang biasanya hadir di tengah jaringan buruk.

Dengan menggunakan PWA, UI dan data-data terakhir yang diakses seseorang dalam peramban tidak akan hilang walau ada gangguan maupun tak ada jaringan. Kecanggihan ini tak terlepas dari *Service Worker*, teknologi yang membekali PWA dengan fungsionalitas luring (luar jaringan), notifikasi, *update* konten, pergantian konektivitas, dan lain sebagainya. *Service Worker* sendiri adalah *proxy* yang berada di antara peramban, aplikasi *web*, dengan jaringan. Aplikasi dengan *proxy* tersebut biasanya akan berjalan secara *offline* dan dapat mengembalikan data dari *cache* bila permintaan ke jaringan gagal terhubung.

2.13 XAMPP

XAMPP adalah distribusi Apache kecil dan ringan yang berisi pengembangan web paling umum teknologi dalam satu paket (DD Dvorski, 2007). Isi konten, ukurannya yang kecil, dan mudah dibawa menjadikannya alat yang ideal untuk mengembangkan dan menguji aplikasi di PHP dan MySQL. XAMPP berisi Apache HTTP Server, PHP, MySQL, phpMy*Admin*, Openssl, dan SQLite.

2.14 Alat Bantu Perancangan Sistem

Berikut adalah alat-alat yang digunakan guna membantu dalam melakukan perancangan sistem.

2.14.1 *Unified Modeling System* (UML)

Unified Modeling Language (UML) merupakan bahasa *visual* untuk pemodelan dan komunikasi mengenai sistem dengan menggunakan *diagram* dan

teks-teks pendukung (Sukamto, Salahudin, 2013).

2.14.2 *Use case Diagram*

Use case diagram mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dengan kata lain, *use case diagram* digunakan untuk mengetahui fungsi-fungsi apa saja yang terdapat di dalam sistem dan siapa saja yang berhak mengakses fungsi tersebut (Sukamto, Salahudin, 2013).

2.14.3 *Activity Diagram*

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis (Sukamto, Salahudin, 2013).

2.14.4 *Class Diagram*

Class diagram menggunakan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. *Class diagram* memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Metode atau operasi adalah fungsi-fungsi yang dimiliki oleh suatu kelas (Sukamto, Salahudin, 2013).

2.14.5 *Sequence Diagram*

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Dalam menggambarkan *sequence diagram* perlu memperhatikan objek-objek yang terlibat di dalam *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu (Sukamto, Salahudin, 2013).

2.15 Pengujian Perangkat Lunak

Berikut adalah beberapa metode yang dipakai untuk menguji perangkat lunak yang dibuat.

2.15.1 *Metode Black Box*

Ada beberapa metode yang dapat digunakan untuk memilih *data* pengujian, salah satunya adalah metode *black box*. Metode *black box* yaitu *data* pengujian dipilih berdasarkan spesifikasi masalah tanpa memperhatikan *detail internal* dari *program* (Sukamto, 2009), untuk memeriksa apakah *program* dapat berjalan dengan benar. Pemilihan *data* pengujian paling tidak harus dipilih dengan cara berikut (Sukamto, 2009):

1. *Extreme values*, banyak *program error* pada suatu batas *range* dari aplikasi.
2. *Ilegal values*, yaitu suatu *data* atau nilai yang tidak diperbolehkan maupun *data* yang tidak berguna.
3. *Values*, yaitu *data* yang mudah diperiksa.
4. *Typical realistic value*, yaitu mencoba *program* dengan *data* pengujian untuk melihat bagaimana *program* melakukannya. *Data* ini harus cukup sederhana sehingga hasilnya dapat dihitung secara manual.

Adapun beberapa teknik pengujian menggunakan *black box* antara lain sebagai berikut:

a. *Requirement Testing* adalah spesifikasi kebutuhan yang terasosiasi dengan perangkat lunak (*input, output*, fungsi, performansi) diidentifikasi pada tahap spesifikasi kebutuhan dan desain. *Requirement testing* melibatkan pembuatan kasus uji untuk setiap spesifikasi kebutuhan yang terkait dengan *program* (Sukamto, 2009)
b. *Performance Testing* adalah mengevaluasi kemampuan *program* untuk beroperasi dengan benar dipandang dari sisi acuan kebutuhan misalnya: aliran *data*, ukuran pemakaian memori, kecepatan eksekusi dan lain-lain. Untuk mencari tahu beban kerja atau kondisi konfigurasi *program* dan dapat digunakan untuk menguji batasan lingkungan *program* (Sukamto, 2009). *Scenario Testing* adalah pengujian yang rea*list*is, kredibel dan memotivasi *stakeholder*, tantangan untuk *program* dan mempermudah pengujian untuk melakukan evaluasi. Pengujian ini menyediakan kombinasi variabel-variabel dan fungsi yang sangat berarti daripada kombinasi buatan yang didapatkan dengan pengujian *domain* atau desain pengujian kombinasi.

2.15.2 *Pengujian User Acceptance Testing (UAT)*

User acceptance testing (UAT) merupakan pengujian yang melibatkan validasi perangkat lunak dalam pengaturan nyata oleh *audiens* yang dituju. Tujuannya bukan untuk memeriksa persyaratan yang ditentukan tetapi untuk memastikan bahwa perangkat lunak memenuhi kebutuhan pelanggan (I Otaduy, O Díaz, 2017). Pengujian UAT yang dilakukan oleh pengguna langsung aplikasi yang akan menggunakan perangkat lunak yang sedang dikembangkan untuk memastikan perangkat lunak sudah sesuai dengan kebutuhan dari pengguna. Adapun jenis-jenis UAT terdiri dari:

1. *Alpha Testing*

Pengujian *Alpha* adalah pengujian akhir sebelum perangkat lunak diluncurkan untuk pengguna secara umum. *Alpha testing* dilakukan *in-house* atau internal serta melibatkan pengembang, analis bisnis dan tim penguji yang merupakan *user*. Misalnya PIC atau staf IT dari perusahaan terlebih dahulu, tanpa para pengguna akhir dari perusahaan tersebut (bidang akuntansi, HRD, dsb). *Alpha test* memiliki dua fase, yakni sebagai berikut.

a. Pada tahap pertama dari pengujian *alpha*, perangkat lunak diuji oleh pengembang di lingkungan *internal developer*. Mereka menggunakan perangkat lunak *debugger*, atau *debugger hardware-assisted*. Tujuannya adalah untuk menangkap *bug* dengan cepat.
b. Pada tahap kedua pengujian *alpha*, perangkat lunak ini diserahkan kepada staf QA (*Quality assurance*) perangkat lunak, untuk pengujian tambahan dalam lingkungan yang mirip dengan penggunaan yang dimaksudkan. Hal ini untuk menyimulasikan suasana atau lingkungan pengujian yang sebenarnya sehingga ketika sistem tersebut dipasang, sudah tidak terjadi kegagalan maupun cacat sistem secara *real* (Udz Md, 2014).

2. *Beta Testing*

Pengujian *beta* juga dikenal sebagai pengujian pengguna berlangsung di lokasi pengguna akhir (*end user*) untuk memvalidasi kegunaan, fungsi, kompatibilitas, dan uji reliabilitas dari perangkat lunak yang dibuat. Hal ini juga dikenal sebagai uji lapangan. *Test* ini dilakukan di lingkungan *end-user* (*server* kantor pengguna, bukan *server* pengembang). *Beta test* merupakan tahap kedua dari pengujian perangkat lunak di mana pengguna sebenarnya dari aplikasi mencoba langsung aplikasi yang akan mereka gunakan.

Tujuan dari pengujian *beta* adalah untuk menempatkan aplikasi anda di tangan pengguna yang sebenarnya yang berada di luar tim teknik anda untuk menemukan setiap kekurangan atau masalah dari perspektif pengguna akhir (Buche, 2021).