BAB II
LANDASAN TEORI

2.1 Penelitian Terkait

Hendra Setyo Adi Nugroho, dkk, 2013, STMIK STIKOM Surabaya. Program Studi Sistem Informasi dalan jurnal ilmiahnya yang berjudul "Sistem Informasi IT *Helpdesk* Prioritas Kerja Berbasis *Web* (Studi Kasus: PT Pelabuhan Indonesia III Cabang Tanjung Perak)". Tujuan penulisan jurnal tersebut adalah merancang bangun sistem informasi IT *Helpdesk* dengan prioritas kerja pada PT Pelabuhan Indonesia III cabang Tanjung Perak Surabaya, memberikan *helpdesk* solution berkaitan dengan permasalahan perangkat IT dan membuat laporan mengenai informasi keandalan dan ketersediaan kinerja perangkat TIK. Metode pengembangan yang digunakan untuk mengembangkan sistem ini adalah metode *System Development Life Cycle* (SDLC). Metode ini mempunyai kelebihan yaitu menyediakan tahapan yang dapat digunakan sebagai pedoman mengembangkan sistem dan akan memberikan hasil sistem yang lebih baik karena sistem dianalisis dan dirancang secara keseluruhan sebelum diimplementasikan. Hasil dari pengembangan aplikasi *helpdesk* ini adalah adanya sistem informasi IT *Helpdesk* yang terbukti melalui hasil iji coba dapat menyelesaikan permasalahan IT yang ada pada PT Pelabuhan Indonesia III Cabang Tanjung Perak Surabaya. Perbedaan penelitian tersebut dengan penelitian yang penulis lakukan adalah pada metode yang digunakan dan ruang lingkup permasalahan. Peneliti sebelumnya menggunakan metode SDLC dan membatasi ruang lingkup pada permasalahan IT pada PT Pelabuhan Indonesia III Cabang Tanjung Perak Surabaya, sedangkan penulis menggunakan metode pengembangan *Rapid Application Development* (RAD) dan ruang lingkup permasalahan pada permasalahan operasional di STMIK Duta Bangsa Surakarta.

Nurwati dan Anita Diana. 2012 Universitas Budi Luhur Jakarta. Program Studi Sistem Informasi dalan jurnal ilmiahnya yang berjudul "Analisa dan Perancangan *Helpdesk* untuk Layanan Mahasiswa FTI Universitas Budi Luhur". Tujuan penulisan jurnal tersebut adalah program yang bertujuan untuk membuat aplikasi yang dapat menampung dan mewadahi setiap pertanyaan-pertanyaan dan keluhan-keluhan yang diajukan oleh mahasiswa. Metode pengembangan yang

digunakan untuk mengembangkan sistem ini adalah metode *System Development Life Cycle* (SDLC). Metode ini mempunyai kelebihan yaitu menyediakan tahapan yang dapat digunakan sebagai pedoman mengembangkan sistem dan akan memberikan hasil sistem yang lebih baik karena sistem dianalisis dan dirancang secara keseluruhan sebelum diimplementasikan. Hasil dari pengembangan aplikasi helpdesk ini dapat mempermudah kegiatan pelayanan kepada mahasiswa, terutama dalam kegiatan *helpdesk* sehingga proses *helpdesk* dapat dilakukan dengan cepat, akurat, dan terkomputerisasi.

Toni Kurniawan, 2011, Universitas Islam Negeri Syarif Hidayatullah Jakarta. Program Studi Sistem Informasi dalan jurnal ilmiahnya yang berjudul "Pengembangan Aplikasi *Helpdesk* pada PT. Jakarta International Container Terminal". Tujuan penulisan jurnal tersebut adalah mempermudah dan mempercepat pengolahan data *helpdesk* dan mengurangi masalah keterlambatan dan ketidakakuratan pelaporan kepada manajer. Metode pengembangan yang digunakan untuk mengembangkan sistem ini adalah metode *Rapid Application Development* (RAD). Metode ini mempunyai kelebihan yaitu lebih efektif dari pendekatan *waterfall/sequential linear* dalam menghasilkan sistem yang memenuhi kebutuhan langsung dari pelanggan dan cocok untuk proyek yang memerlukan waktu yang singkat. Hasil dari pengembangan aplikasi *helpdesk* ini mampu mempersingkat waktu pengolahan data *helpdesk*, dan dapat menghasilkan laporan-laporan *helpdesk* dengan cepat dan akurat, serta dapat membantu proses pengambilan keputusan.

Penelitian yang di lakukan oleh Nurmalasari pada tahun 2013 yang berjudul Perancangan *Aplikasi Service Desk* Penanganan Keluhan Kerusakan Perangkat Teknologi Informasi Fakultas Teknik Universitas Tanjung Pura. Penelitian ini bertujuan untuk menghasilkan suatu aplikasi *service desk* berbasis web yang mampu mengirimkan keluhan atau permasalahan ke unit yang bertanggung jawab sehingga mempercepat penanganannya, serta memnghasilkan aplikasi yang dapat menentukan prioritas penanganan masalah berdasarkan bobot *presentase* kegawatan permasalahan, tingkat kebutuhan TI serta ketersediaan teknisi. Penelitian ini menggunakan *framework* ITIL *(Information Technology Infrastrukcture Library*), penggunaan *framework* ini dimaksudkan untuk

meminimalisir insiden yang terjadi sehingga dapat menjaga keberlangsungan layanan TI.

Berdasarkan penelitian terkait yang sudah dipaparkan di atas, maka pada penelitian ini penulis akan membangun sebuah aplikasi *helpdesk* pada Kanwil DJKN provinsi Kalimantan Barat menggunakan sistem *ticketing*. Penelitian ini bertujuan untuk menghasilkan suatu aplikasi *helpdesk* berbasis web yang mampu menjawab permasalahan ke unit yang bertanggung jawab sehingga mempercepat penanganannya.

2.2 Aplikasi

Aplikasi adalah sebuah perangkat lunak yang berisi sebuah *coding* atau perintah yang mana bisa diubah sesuai keinginan (Syani dan Werstantia, 2019: 88). Selanjutnya Menurut Rachmad Hakim S (2018), Aplikasi adalah perangkat lunak yang digunakan untuk tujuan tertentu, seperti mengolah dokumen, mengatur *Windows* &, permainan (game), dan sebagainya.

2.3 Helpdesk

Menurut (Rikip, 2016). *Helpdesk* adalah sebuah sumber daya yang dimaksudkan menyediakan informasi dan dukungan kepada *user* yang berkaitan dengan produk dan jasa dalam perusahaan. Tujuan dari *helpdesk* yaitu untuk memecahkan atau menyelesaikan masalah tentang produk seperti komputer, peralatan elektronik, dll. Biasanya perusahaan merancang *helpdesk* untuk memberikan bantuan kepada karyawan.

2.4 HTML

Menurut (Endra dan Aprilita, 2018), HTML atau *Hypertext Markup Languange* merupakan salah satu bahasa yang biasa digunakan oleh pengguna dalam membuat tampilan yang digunakan oleh *web application*. *Hypertext markup language* (HTML) merupakan bahasa dasar pembuatan *web*. HTML menggunakan tanda (*mark*) untuk menandai bagian-bagian dari *text*. HTML disebut sebagai bahas dasar, karena dalam membuat *web*, jika hanya menggunakan HTML maka tampilan *web* terasa hambar (Rerung, 2018:18).

2.5 PHP

Menurut Supono dan Putratama (2018) mengemukakan bahwa “PHP (*Hypertext Preprocessor*) adalah suatu bahasa pemograman yang digunakan untuk menerjemahkan baris kode program menjadi kode mesin yang dapat dimengerti oleh computer yang bersifat *server-side* yang ditambahkan ke HTML.

2.6 CSS

CSS adalah suatu Bahasa pemrograman *web* yang digunakan untuk mengendalikan dan membangun berbagai komponen dalam *web* sehingga tampilan *web* akan lebih rapi, terstruktur, dan seragam. CSS merupakan salah satu kode pemrograman yang bertujuan untuk menghias dan mengatur gaya tampilan atau *layout* halaman *web* agar lebih menarik. CSS adalah sebuah teknologi internet yang direkomendasikan oleh *World Wide Web Consortium* atau W3C pada tahun 1996 (Wahyudi, 2017).

2.7 XAMPP

Menurut Iqbal (2019) menyatakan XAMPP merupakan *software* server *apache* dimana dalam XAMPP yang telah tersedia *database* server seperti MySQL dan PHP *programming*.

2.8 *Bootstrap*

Menurut (Nugroho dan Setiyawati, 2019) *bootstrap* adalah *framework* css untuk membuat tampilan web. *Bootstrap* merupakan salah satu *framework* HTML, CSS dan *javascript* yang paling populer di kalangan *web developer*. Pada saat ini hampir semua *web developer* telah menggunakan *bootstrap* untuk membuat tampilan *front-end* menjadi lebih mudah dan sangat cepat, karena anda hanya perlu menambahkan *class-class* tertentu untuk misalnya membuat tombol, *grid, navigasi* dan lainnya.

Bootstrap telah menyediakan kumpulan komponen *class interface* dasar yang telah dirancang sedemikian rupa untuk menciptakan tampilan yang menarik, bersih dan ringan. selain komponen *class interface*, *bootstrap* juga memiliki fitur *grid* yang berfungsi untuk mengatur *layout* pada halaman *website* yang bisa digunakan dengan sangat mudah dan cepat, dengan menggunakan bootstrap kita juga diberi keleluasaan dalam mengembangkan tampilan *website* yang

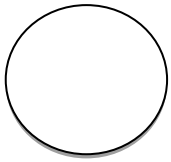
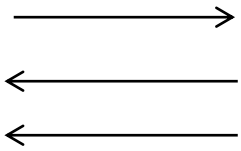
menggunakan *bootstrap* yaitu dengan cara mengubah tampilan *bootstrap* dengan menambahkan *class* dan CSS sendiri.

2.9 Diagram Arus Data (*Data Flow Diagram* – DFD)

Menurut Saputra (2018:11), *Data Flow Diagram* merupakan suatu diagram yang menggambarkan alir data dalam suatu entitas ke sistem atau ke entitas. *Data Flow Diagram* juga dapat diartikan sebagai teknik grafis yang menggambarkan alir data dan transformasi yang digunakan sebagai perjalanan data dari input atau masukan menuju keluaran atau output.

Berikut ini adalah simbol *Data Flow Diagram* menurut Gane atau Sarson serta Yourdon atau De Marco, yang dapat dilihat pada tabel 2.1:

Tabel 0.1 Simbol-simbol DFD menurut Yourdon

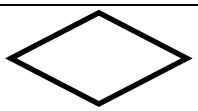
Yourdon/De Marco	Nama Simbol	Keterangan
	Entitas Luar	Entitas eksternal dapat berupa orang atau unit terkait yang berinteraksi dengan system tetapi di luar sistem.
	Proses	Orang, unit yang mempergunakan atau melakukan transformasi data. Komponen fisik tidak diidentifikasikan.
	Aliran Data	Aliran data dengan arah khusus dari sumber ke tujuan.
	Data Store	Penyimpanan data atau tempat data direfer oleh proses.

Sumber: Saputra (2018:11)

2.10 Diagram Hubungan Entitas *(Entity Relationship Diagram)*

Menurut (Sukamto & Shalahuddin, 2018:50) menyatakan bahwa "ERD digunakan untuk permodelan basis data relasional". Menurut Al-Bahra dalam (Rahmayu, 2016:34) menerangkan bahwa "*Entity Relationship Diagram* (ERD) adalah diagram yang menunjukkan informasi dibuat, disimpan, dan digunakan dalam sistem bisnis". Adapaun simbol-simbol yang sering digunakan dalam *Entity Relationship Diagram* (ERD) dilihat pada tabel berikut:

Tabel 0.2 Simbol *Entity Relationship Diagram*

	Entitas	suatu yang nyata atau abstrak yang mempunyai karakteristik di mana kita akan menyimpan data.
	Atribut	ciri umum semua atau sebagian besar instansi pada entitas tertentu.
	Relasi	hubungan alamiah yang terjadi antara satu atau lebih entitas.
	Link	garis penghubung atribut dengan kumpulan entitas dan kumpulan entitas dengan relasi.

Sumber: (Sukamto & Shalahuddin, 2018:50)

Menurut Koko Mukti Wibowo (2015) kardinalitas Relasi (derajat relasi) menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Kardinalitas di antara dua himpunan entitas (misal A dan B) dapat berupa :

1) Satu ke satu (one to one / 1 : 1): Setiap entitas pada himpunan entitas A berhubungan dengan paling banyak dengan satu entitas pada him punan entitas B, begitu juga sebaliknya.

2) Satu ke Banyak (one to many / 1 : M) :Setiap entitas pada himpunan entitas A dapat berhub ungan dengan banyak entitas pada himpunan entitas B. Tetapi tidak sebaliknya, dimana setiap entitas pada himpunan entitas B berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas A.
3) Banyak ke Satu (many to one / M : 1): Setiap entitas pada himpunan entitas A berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas B. Tetapi tidak sebaliknya, dimana setiap entitas pada himpunan entitas A berhubungan dengan paling banyak satu entitas pada himpunan entitas B.
4) Banyak ke Banyak (many to many / M : N): Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, demikian juga sebaliknya.

2.11 *Systems Development Life Cycle* (SDLC)

Menurut Prof. Dr. Sri Mulyani, AK., CA. (2017) SDLC adalah proses logika yang digunakan oleh seorang analis sistem untuk mengembangkan sebuah sistem informasi yang melibatkan *requirments, validation, training* dan pemilik sistem. Kesimpulannya adalah pengertian SDLC adalah siklus atau tahapan yang digunakan dalam pembuatan/pengembangan suatu sistem informasi agar pengerjaan sistem berjalan secara terstruktur, efektif dan sesuai dengan tujuan yang diinginkan.

SDLC berisi tahapan-tahapan yang dikembangkan untuk tujuan tertentu. Berikut ini tujuh tahapan yang harus dilewati.

1. Tahapan Analisis Sistem

   Tahapan pertama, yaitu analisis sistem. Pada tahap ini, sistem akan dianalisis bagaimana akan dijalankan nantinya. Hasil analisis berupa kelebihan dan kekurangan sistem, fungsi sistem, hingga pembaharuan yang dapat diterapkan.

   Bagian ini termasuk dalam bagian perencanaan. Bagian lain yang termasuk dalam perencanaan ialah alokasi sumber daya, perencanaan kapasitas, penjadwalan proyek, estimasi biaya, dan penetapan. Dengan demikian, hasil dari tahap perencanaan ialah rencana proyek, jadwal, estimasi biaya, dan ketentuan. Idealnya manajer proyek dan pengembang dapat bekerja maksimal pada tahap ini.

2. Tahapan Perancangan Sistem

Setelah persyaratan dipahami, perancang dan pengembang dapat mulai mendesain *software*. Tahapan ini akan menghasilkan *prototype* dan beberapa *output* lain meliputi dokumen berisi desain, pola, dan komponen yang diperlukan untuk mewujudkan proyek tersebut. Setelah spesifikasi, kemudian dilakukan perancangan sistem sebagai tahapan kelanjutannya. Tahap ini ialah tahap di mana seluruh hasil analisis dan pembahasan tentang spesifikasi sistem diterapkan menjadi rancangan atau cetak biru sebuah sistem. Tahap ini disebut sebagai cetak biru, di mana sistem sudah siap untuk dikembangkan mulai dari implementasi, analisis sistem, hingga tenaga pendukung sistem yang akan dikembangkan.

3. Tahap Pembangunan Sistem

   Pengembangan sistem ialah tahap di mana rancangan mulai dikerjakan, dibuat, atau diimplementasikan menjadi sistem yang utuh dan dapat digunakan. Jika diibaratkan bangunan, tahap ini merupakan tahap membangun. Tahap ini memakan waktu cukup lama karena akan muncul kendala-kendala baru yang mungkin dapat menghambat jalannya pengembangan sistem. Pada tahapan ini, perancangan bisa saja berubah karena satu atau banyak hal. Tahap selanjutnya ialah memproduksi perangkat lunak di bawah proses pengembangan. Menurut metodologi yang sudah digunakan, tahap ini dapat dilakukan dengan cepat. *Output* yang dihasilkan pada tahap ini ialah perangkat lunak yang telah berfungsi dan siap diuji.

4. Tahap Pengujian Sistem

   Sesudah sistem selesai dikembangkan, sistem harus melalui pengujian sebelum digunakan atau dikomersialisasikan. Tahap pengujian sistem harus dijalankan untuk mencoba apakah sistem yang dikembangkan dapat bekerja optimal atau tidak. Pada tahap ini, ada beberapa hal yang harus diperhatikan, seperti kemudahan penggunaan sampai pencapaian tujuan dari sistem yang sudah disusun sejak perancangan sistem dilakukan. Jika ada kesalahan, tahap pertama hingga keempat harus diperbarui, diulangi, atau pun dirombak total. Tahap tes SDLC ialah bagian paling penting dalam rangkaian pembuatan sebuah perangkat lunak. Karena sangat tidak mungkin mempublikasikan sebuah *software* tanpa melalui pengujian terlebih dahulu.

Beberapa pengujian yang harus dilewati, antara lain kualitas kode, tes fungsional, tes integrasi, tes performa, dan tes keamanan. Untuk memastikan pengujian berjalan teratur dan tidak ada bagian yang terlewati, tes dapat dilakukan menggunakan perangkat *Continuous Integration* seperti *Codeship*. Dari tahap ini, akan dihasilkan perangkat lunak yang telah dites dan siap untuk disebarkan ke dalam proses produksi.

5. Implementasi

Implementasi dan pemeliharaan merupakan tahap akhir dalam pembuatan SDLC. Di tahap ini sistem sudah dibuat, diuji coba, dan dipastikan dapat bekerja optimal. Setelah tahap pembuatan selesai, dilakukan implementasi dan pemeliharaan oleh pengguna. Pemeliharaan sangat penting untuk memastikan sistem bekerja dengan optimal setiap saat. Untuk implementasi, langkah yang harus dilakukan adalah sebagai berikut.

- Melakukan survei dan penilaian terhadap kelayakan sistem yang sudah dikembangkan.
- Menganalisis dan mempelajari sistem yang sudah ada dan sedang berjalan. Melakukan pemecahan masalah dalam pengembangan sistem.
- Menentukan penggunaan *hardware* dan *software* yang tepat.
- Merancang dan mengembangkan sistem baru.
- Memelihara dan meningkatkan sistem yang baru jika diperlukan.

Fase ini disebut juga sebagai tahap penyebaran. Pada tahap ini, *software* disebarkan setelah melewati proses yang melibatkan beberapa persetujuan manual. Tahap ini dilakukan sebelum menurunkan *software* ke produksi. Proses penyebaran dapat dilakukan menggunakan *Application Release Automation* (ARA) sebelum masuk ke proses produksi. *Output* yang didapat dari tahap ini ialah perangkat lunak yang siap untuk diproduksi secara massal.

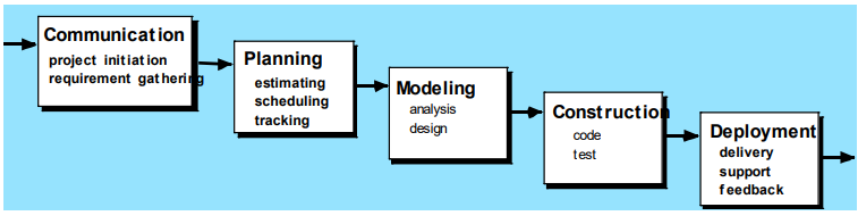
6. Pemeliharaan Sistem

Pemeliharaan sistem yang sudah dibuat sangat penting untuk referensi di kemudian hari. Pemeliharaan ialah tahap akhir yang menjadi permulaan fase yang baru yaitu penggunaan. SDLC belum berakhir di tahap ini. *Software* yang dihasilkan harus terus dipantau untuk memastikan ia berjalan sempurna. Celah

dan kerusakan yang ditemukan pada proses produksi harus dilaporkan dan diselesaikan. Jika ditemukan sebelum diproduksi massal, ini akan lebih baik daripada menyelesaikan dengan merombak semuanya dari awal ke akhir.

2.12 *Waterfall*

Menurut Pressman (2015:42), Metode *Waterfall* adalah suatu model pengembangan yang bersifat sistematis dan berurutan dalam membangun sebuah perangkat lunak. Proses pembuatannya mengikuti alur mulai dari analisis, desain, kode, pengujian dan pemeliharaan. Gambar tahapan metode *waterfall* menurut Pressman:



Sumber (Pressman (2015:42)

Gambar 2. 1 *Waterfall*

Penjelasan tahapan metode *waterfall* menurut Pressman:

- *Communication*. Tahap komunikasi antara pengembang dengan konsumen agar dapat mencapai tujuan yang ingin dicapai, yang hasilnya adalah inisialisasi proyek, seperti menganalisis permasalahan yang dihadapi dan mengumpulkan data-data yang diperlukan, serta membantu mendefinisikan fitur dan fungsi software.
- *Planning*. Tahap yang membahas tentang tugas teknis yang akan dilakukan, sumber daya manusia, resiko yang dapat terjadi, penjadwalan kerja, dan *tracking* proses pengerjaan sistem.
- *Modeling*. Tahap desain dan pemodelan arsitektur sistem, yang berfokus pada desain struktur data, arsitektur perangkat lunak, tampilan antarmuka, dan algoritma program. Tujuannya adalah untuk lebih memahami gambaran keseluruhan tentang apa yang akan dilakukan.
- *Construction*. Tahap konstruksi ini adalah proses menerjemahkan bentuk desain ke dalam kode atau bentuk/bahasa yang dapat dibaca mesin. Setelah

pengkodean selesai, uji sistem dan kode yang dibuat. Tujuannya adalah untuk mengidentifikasi kemungkinan kesalahan sehingga dapat diperbaiki kemudian.

- *Deployment*. Tahap implementasi perangkat lunak kepada pelanggan, pemeliharaan perangkat lunak secara berkala, perbaikan perangkat lunak, evaluasi perangkat lunak, dan pengembangan perangkat lunak berdasarkan umpan balik yang diberikan, sehingga sistem dapat terus beroperasi dan berkembang sesuai fungsinya.

Kelebihan dan kekurangan metode *waterfall*.

1. Kelebihannya, Pada tahap awal pengembangan melalui metode ini diperlukan analisis data yang jelas dan lengkap. Hal ini membuat proyek memiliki tujuan akhir yang jelas. Dengan cara ini, tentu produk akhir akan sesuai dengan konsep aslinya. Dengan proses yang jelas pengerjaan proyek lebih detail. Dengan cara ini, kesalahan dapat dikurangi. Semakin rinci tugas yang harus dilakukan, semakin kecil kemungkinan untuk membuat kesalahan. Salah satu keunggulan model ini adalah dokumentasi yang baik. Sehingga setiap perkembangan dan informasi dapat terekam dan dapat diakses oleh *developer* lain

2. Kekurangannya, karena pengerjaannya yang struktual, proses yang dilakukan akan lama. Karena produk perangkat lunak baru hanya dapat dilihat setelah hasil akhir selesai, jika pelanggan tidak puas dan dimodifikasi, perlu dikerjakan ulang. Karena pengulangan, tentunya biaya dan tenaga akan lebih besar. Tidak dapat mengganti konsep ditengah jalan, akan sulit untuk pengembah menggantinya. Karena jika harus diubah, berarti harus diubah semua dari awal.

2.13 Pengujian Aplikasi

2.13.1 Pengujian *Black Box*

Black Box testing berfokus pada persyaratan fungsional perangkat lunak yang memungkinkan *engineers* untuk memperoleh set kondisi input yang sepenuhnya akan melaksanakan persyaratan fungsional untuk sebuah program (Pressman, 2010).

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*. *Black Box Testing* cenderung untuk menemukan hal-hal berikut (Mustaqbal dan Firdaus, 2015).

1. Fungsi yang tidak benar atau tidak ada
2. Kesalahan antarmuka (*interface errors*)
3. Kesalahan pada struktur data dan akses basis data
4. Kesalahan performansi (*performance errors*)
5. Kesalahan inisialisasi dan terminasi

2.13.2 Pengujian *User Acceptence Test* (UAT)

Menurut Ferry, William E, *User Acceptance Testing* (UAT) merupakan pengujian yang dilakukan oleh *end-user* dimana *user* tersebut adalah staff/karyawan perusahaan yang langsung berinteraksi dengan sistem dan dilakukan verifikasi apakah fungsi yang ada telah berjalan sesuai dengan kebutuhan/fungsinya. Setelah dilakukan sistem *testing*, *acceptance testing* menyatakan bahwa sistem perangkat lunak memenuhi persyaratan.

Pengujian penerimaan pengguna (UAT) adalah fase terakhir dari proses pengujian perangkat lunak. Selama UAT, perangkat lunak perangkat lunak diuji untuk memastikan tugas-tugas apakah sudah sesuai dengan spesifikasinya. UAT adalah salah satu prosedur proyek perangkat lunak final dan paling penting yang harus terjadi sebelum perangkat lunak tersebut dikembangkan dan diluncurkan ke pasar. UAT juga dikenal sebagai pengujian beta, pengujian aplikasi atau pengujian pengguna akhir.

Menurut Black, *acceptance testing* biasanya berusaha menunjukkan bahwa sistem telah memenuhi persyaratan-persyaratan tertentu. Pada pengembangan *software* dan *hardware* komersial, *acceptance test* biasanya disebut juga "*alpha tests*" (yang dilakukan oleh pengguna *in-house*) dan "*beta tests*" (yang dilakukan oleh pengguna yang sedang menggunakan atau akan menggunakan sistem tersebut). *Alpha* dan *beta test* biasanya juga menunjukkan bahwa produk sudah siap untuk dijual atau dipasarkan. *Acceptance testing* mencakup data, *environment*

dan skenario yang sama atau hampir sama pada saat *live* yang biasanya berfokus pada skenario penggunaan produk tertentu.

Alpha dan *beta test* biasanya juga menunjukkan bahwa produk sudah siap untuk dijual atau dipasarkan. Pengguna tes biasanya dilakukan oleh klien atau pengguna akhir, dan tidak fokus pada identifikasi masalah sederhana seperti kesalahan ejaan dan cacat *showstopper* atau *crash* perangkat lunak tapi juga masalah – masalah lainnya. Hasil tes ini memberikan kepercayaan kepada klien tentang bagaimana sistem akan siap di produksi. Pada pengembangan perangkat lunak, *user acceptance testing* (UAT) juga disebut pengujian beta (*beta testing*), pengujian aplikasi (*application testing*) dan pengujian pengguna akhir (*end user testing*) adalah tahapan pengembangan perangkat lunak ketika perangkat lunak diuji pada dunia nyata.

Proses UAT memastikan bahwa aplikasi sistem rekomendasi pelayanan jasa yang peneliti implementasikan tersebut akan memberi solusi, memenuhi harapan pengguna dan bekerja seperti yang diharapkan serta meyakinkan user atau pelanggan aplikasi tersebut apakah sistem bisa diterima dengan baik atau tidak. Dari definisi di atas dapat disimpulkan bahwa *user acceptance testing* adalah pengujian yang dilakukan oleh pengguna dari sebuah sistem untuk memastikan fungsi-fungsi yang ada pada sistem tersebut telah berjalan dengan baik dan sesuai dengan kebutuhan pengguna.