BAB III

LANDASAN TEORI

3.1. Properti

Properti merupakan sebuah istilah atau sebuah sebutan ketika berbicara mengenai tanah atau sebuah bangunan. Properti sendiri merupakan sebuah istilah dalam Bahasa Inggris yang kemudian diterjemahkan ke dalam Bahasa Indonesia sebagai properti. Definisi dari properti adalah hak dan kepemilikan atas suatu tanah atau bangunan[16]. Menurut Kamus Besar Bahasa Indonesia atau KBBI, properti atau *property* merupakan sebuah harta dalam bentuk tanah atau bangungan.

Properti merupakan sebuah bisnis yang banyak diminati oleh para investor atau bahkan para pemegang saham dikarenakan bisnis properti dianggap sebagai sebuah investasi yang paling aman[17]. Bisnis properti dapat dikatakan sebagai salah satu indikator dalam mengukur pertumbuhan ekonomi suatu negara[18]. Menurut Michael C. Thomsett dan juga Jean Freestone Thomsett, pasar properti dibagi menjadi tiga yaitu

1. *Residental Property*, yang termasuk didalamnya adalah apartemen, perumahan dan tanah.
2. *Commercial Property*, biasanya dirancang untuk bisnis contoh area parkir atau gedung untuk menyimpan barang
3. *Industrial Property,* biasanya untuk keperluan industri seperti pabrik [16]

3.2. Pengujian Perangkat Lunak atau *Software Testing*

Pengujian perangkat lunak atau *Software Testing* mulai dibahas pada tahun 1978 oleh G.J Mayers dalam buku *The Art of Software Testing*[19]. Pengujian perangkat Lunak atau *Software Testing* adalah proses yang dilakukan untuk memastikan kualitas dari sebuah produk sudah dapat digunakan atau sudah sesuai dengan standar penggunaan. Pengujian

perangkat lunak tidak hanya memastikan bahwa semua fungsi dalam produk dapat berjalan dalam semua kondisi, akan tetapi juga memastikan bahwa produk tidak berfungsi dengan baik dalam kondisi tertentu[1]. Proses pengujian dimulai dengan asumsi bahwa perangkat lunak memiliki *error* lalu pengujian dilakukan untuk menemukan *error* tersebut, jadi dapat dikatakan bahwa tujuan dari pengujian perangkat lunak adalah untuk memastikan bahwa tidak ada *error* dan menunjukan bahwa setiap fungsi dalam sebuah program berjalan dengan baik.

Pengujian perangkat lunak atau *Software Testing* termasuk kedalam bagian *Software Development Life Cycle* (SDLC). Tahap *Software Testing* menjadi tahap yang penting dalam proses SDLC dikarenakan tahap ini akan memastikan kualitas dari produk yang dibuat sebelum dikirimkan ke *customer* atau *user*. Selain menemukan *error* dalam sebuah *software*, orang melakukan proses *testing* atau sering disebut dengan *tester* dapat memberikan masukan terhadap pihak pengembang terkait *user interface* dari *software* yang diuji coba. Tahap pengujian di bagi menjadi pengujian statis dan pengujian dinamis. Statis pengujian *source code* atau sering disebut dengan *White Box Testing* dan dinamis pengujian berdasarkan fungsional atau disebut dengan *Black Box Testing*.

3.3. Software Testing Life Cycle (STLC)

Software Testing Life Cycle atau bisa disingkat dengan STLC merupakan sebuah siklus pengujian yang dirancang dengan baik secara sistematis yang dilakukan untuk memastikan kualitas produk. Siklus dalam STLC berupa *Requirement Analysis, Test Planning, Test Case Development, Environment*

Setup, Test Execution, dan *Test Cycle Closure*[26].

Gambar 3.2 Software Testing Life Cylce(STLC)

1. *Requirement Analysis*

   Requirement Analysis adalah proses yang dilakukan untuk menghasilkan dokumen untuk kebutuhan pengguna[27]. Pada tahap ini memungkinkan pihak *developer* baik *tester* maupun *programmer* untuk mengetahui kebutuhan seperti apa atau kriteria seperti apa yang diinginkan *customer*. Biasanya di awali dengan wawancara yang dilakukan terhadap *customer* kemudian dihasilkan *output* berupa lembar hasil wawancara yang akan dijadikan sebagai dokumen kebutuhan pengguna. Selain itu tim *Quality Assurance* (QA) atau para *tester* akan melakukan analisa untuk menentukan apakah akan dilakukan *Manual Testing* atau *Automated Testing*.

2. *Test Planning*

   Test Planning adalah fase kedua dalam siklus *Software Testing Life Cycle*. Pada tahap ini akan dilakukan perencanaan pengujian berdasarkan *output* atau hasil dari *Requirement Analysis*[28].

3. *Test Case Development*

   Test Case Development merupakan tahap atau fase ke 3 dalam siklus *Software Testing Life Cycle*. *Test Case* adalah suatu rancangan yang

dibuat dalam bentuk dokumen yang berisi mengai rangkaian pengujian terhadap setiap fitur atau fungsi yang ada dalam sebuah aplikasi. Tujuan dari pembuatan *test case* adalah diharapkan bahwa pengujian dapat dilakukan terhadap semua fitur atau fungsi. Hal ini dilakukan untuk memastikan bahwa setiap fitur sudah berjalan dengan baik sesuai dengan kebutuhan awal *customer* dan diharapkan dapat memberikan sebuah respon ketika terdapat inputan *invalid*. Pada tahap ini tim Quality Assurance (QA) atau bisa disebut para tester akan membuat dokumen *Test Case* atau *Test Scenario*/Skenario *testing*.

4. *Environment Setup*

   Environment Setup merupakan tahanp ke 4 dalam *Software Testing Life Cycle* dan juga langkah vital dari STLC. *Test Environment* adalah kondisi dari *software* ketika pengujian dilakukan[29]. Kondisi *Hardware* beserta spesifikasi termasuk hal yang perlu diperhatikan dalam *Environment Setup*.

5. *Test Execution*

   Test Execution merupakan tahap ke 5 dalam siklus ini. *Test Execution* adalah sebuah proses dimana akan dijalankannya proses pengujian berdasarkan *test case* yang dibuat sebelumnya. *Output* dari fase ini adalah hasi pengujian dari *Test Case*/*Scenario* apakah berhasil atau tidak berserta dengan *expected output* dan *actual output.* Jika masih terdapat *error* atau *bug* dalam *software* tersebut maka akan di masukan ke dalam *dokumen bug report* dan diberikan kembali ke tim *developer*.

6. *Test Cycle Closure*

   Test Cycle Closure merupakan tahap terakhir dari siklus *Software Testing Life Cycle.* Tahap ini adalah tahap dimana dokumen laporan hasil pengujian, Evaluasi dan dokumentasi dilakukan. Pada tahap ini juga akan membahasa strategi untuk kedepannya seperti apa berdasarkan siklus pengujian yang dilakukan.

3.4. Manual Testing

Manual Testing merupakan salah satu bentuk pengujian yang dilakukan dalam proses pengujian perangkat lunak dengan metode *Black Box Testing*. *Manual Testing* atau pengujian secara manual adalah proses pengujian yang dilakukan tanpa memerlukan *tools* apapun. Proses pengujian ini dilakukan dengan cara mempersiapkan *test case* secara *manual*, kemudian pengujian secara manual berdasarkan sudut pandang *user* tanpa bantuan aplikasi atau *tools* untuk menemukan kesalahan. Pengujian ini membutuhkan kesabaran, ketelitian, kreativitas, inovasi serta berwawasan luas.

Manual Testing merupakan teknik yang mudah digunakan. Teknik ini tidak membutuhkan *script testing* dalam pengujiannya. Keunggulan dari *manual testing* sendiri adalah lebih mudah dijalankan dan lebih efektif dalam menemukan kesalahan fungsionalitas. Kekurangan dari *manual testing* adalah kurang efektif dalam hal waktu dikarenakan memakan waktu dalam proses pengujian[30]. Menurut penelitian yang dilakukan, langkah – langkah dalam proses *manual testing* sebagai berikut[31]:

a) Analisa Kebutuhan
b) Pembuatan *Test Plan*
c) Pembuatan *Test Case*
d) Melakukan testing berdasarkan *Test Case*
e) Mendokumentasikan *Error*
f) Melaporkan *error* dan memverifikasi ulang *error*

3.5. Black Box Testing

Black Box Testing adalah sebuah metode pengujian dinamis atau metode pengujian yang berfokus pada *output* yang dihasilkan dengan memberikan input dan kondisi eksekusi. *Black Box Testing* juga sering disebut sebagai pengujian fungsionalitas, maksudnya adalah metode *Black Box Testing* ini hanya menguji fungsi dari perangkat lunak berdasarkan *user interface* (UI). *Black Box Testing* sendiri lebih mudah digunakan ketimbang dengan

metode *White Box Testing* karena ketika pengujian dilakukan dengan metode *Black Box Testing*, penguji atau *tester* tidak perlu mengerti code yang digunakan untuk membuat sebuah perangkat lunak atau *software*. Metode *Black Box Testing* sendiri juga dapat dikatakan sebagai metode yang lebih efisien dalam hal waktu jika dibandingkan dengan *White Box Testing* dikarenakan metode ini hanya melakukan pengujian terhadap fungsi yang kelihatan saja tanpa perlu dilakukannya pengujian terhadap struktur *code* yang digunakan.

Black Box Testing bekerja dengan cara memfokuskan perhatian pada sebuah informasi *domain* sehingga memungkinkan *developer* untuk dapat membuat kondisi input yang dimana hal ini akan melatih syarat-syarat fungsionalitas dalam suatu *software*[20]. Kelebihan dari penggunaan metode ini adalah pengujian dilakukan dari sudut pandang pengguna sehingga seorang penguji atau *tester* tidak memerlukan pengetahuan akan bahasa pemrograman yang digunakan. Metode *Black Box Testing* dapat dikatakan lebih efisien dalam segi waktu dikarenakan hanya melakukan *functional testing saja*, akan tetapi kekurangan dari metode ini adalah pengujian yang dilakukan terbatas hanya dalam lingkup fungsionalitas *software* saja. Metode *Black Box Testing* juga memiliki beberapa teknik pengujian seperti *Equivalence Partitioning* [21].

Equivalence Partitioning merupakan salah satu teknik pengujian dalam metode *Black Box Testing* yang melibatkan pembagian dari menjadi *valid* dan tidak *valid*[22]. Secara Umum, pengujian dengan teknik *Equivalence Partitioning* dilakukan berdasarkan *test case* dari sebuah fungsi yang terdapat dalam perangkat lunak yang sudah ada. Teknik ini akan mempartisi domain input menjadi kelompok atau kelas data dimana *test case* akan diambil. Pengujian dengan teknik *Equivalence Partitioning* melibatkan penentuan *test case*, penentuan kriteria, pendifinisian partisi, pembuatan data uji, pembuatan kasus uji, serta pengujian dan evaluasi[23].