# BAB III

# LANDASAN TEORI

Untuk menunjang sebuah penelitan dibutuhkan dasar-dasar yang kuat. Dasar dasar tersebut digunakan untuk menguatkan hasil dari penelitian yang akan dilakukan. Landasan teori juga memaparkan kondisi atau Batasan-batasan yang akan dilakukan pada penelitian. Batasan ini digunakan untuk memfokuskan tujuan penelitian. Dengan landasan tori alur dan hasil penelitian yang dinginkan dapat tercapai.

## A. Pengujian Perangkat Lunak

Dalam melakukan *testing software* terdapat beberapa manfaat yang didapatkan. Jadi *software testing* bukanlah kegiatan yang tidak produktif. Selama ini *software testing* dianggap menjadi tindakan yang tidak produktif hal ini dikarenakan minimnya SDM dalam sebuah perusahaan. Selain minimnya jumlah SDM tidak kompetennya seseorang dalam melakukan *testing* adalah hal yang mempengaruhi juga. Karena itu *software testin*g harus memenuhi sebuah aturan. Aturan aturan untuk *quality check* adalah *fungtional, reliability, usability, efficiency, maintainability, portability*. Dengan semua aturan tersebut maka proses *testing* akan menjadi lebih produktif. Produktifitas dari *software testing* akan mengurangi biaya. Biaya dapat dikurangi karena berkurangnya juga resiko kegagalan *software* setelah di-*release*. Apalagi jika *software testing* dilakukan secara otomatis maka akan dapat lebih mengurangi waktu dan sumber daya manusia (SDA) dalam membangun sebuah program. Mengunakan *software testing* tersebut *software* akan lebih berkembang dan menjadi *software* yang berkualitas [10].

## B. Tipe-Tipe *Testing*

Dalam melakukan *testing* dapat dilakukan dengan dua cara *automated testing* dan *manual testing*. Keduanya memiliki kekurangan dan kelebihan masing masing.

1. *Manual Testing*

   *Manual Testing* adalah proses pengujian *software* tanpa mengunakan *tool* atau tanpa mengunakan *script*. *Testsing* jenis ini berperan sebagai *end user* yang akan mengunakan aplikasi ini. *Manual testing* berfungsi untuk mengeksplorasi

*bug* yang ada pada perangkat lunak. Hal ini juga dillakukan untuk mensimulasikan perilaku dari *end user*. Dengan adanya hal tersebut, manual *testing* memerlukan pengalaman dan pengetahuan tentang *end user*-nya.

2. *Automated Testing*

   *Automated Testing* adalah jenis *testing* yang mengunakan bantuan *tool testing* atau mengunakan *script*. *Automatic tasting* dilakukan biasanya karena jumlah *Test Case* yang sangat banyak dan merupakan *software* yang sangat penting. *Automated testing* biasanya digunakan untuk mensimulasi *load* pada *server* dan aplikasi dengan banyak *virtual user*, yang jika digunakan akan memberi gambaran jika *software* digunakan di lingkungan yang sebenarnya. Pada *testing* secara otomatis kita harus mengidentifikasi lebih dulu *Test Case* yang akan diuji. Untuk hasil yang baik tidak semua bagian dalam apikasi dilakukan secara otomatis. Sehingga identifikasikan bagian bagian mana saja yang harus dilakukan *test* secara otomatis.

Dengan kolaborasi kedua jenis *test* tersebut maka *testing software* akan menghasilkan produk yang *minim bug* dan menjadi *system* yang kuat [10].

## C. Metode *Testing*

Dalam melakukan testing terdapat dua metode yaitu *black box* dan *white box testing*. Untuk *Black box testing*, penguji tidak membutuhkan akses kedalam kode *program*. Sementara untuk *White box*, penguji mendapatkan akses untuk melihat kode program tersebut dan memahami cara kerja dari program tersebut. Berikut adalah keunggulan dan kelemahan dari masing-masing metode pengujian [10] :

### 1. *Black-box Testing*

**Tabel C.1 Keuggulan dan Kekurangan Black-box Testing**

| No. | Keunggulan | Kekurangan |
|---|---|---|
| 1 | Sangat jelas digunakan untuk melihat prespektif penguna yang dilihat dari prespektif pengembang | Cakup pengujian terbatas karena hanya pengujian sekenario saja yang dilakukan |
| 2 | Banya orang dapat melakukan *Testing* tanpa harus tau bahasa pemrograman atau implementasinya | *Test Case* sangat sulit untuk di desain |
| 3 | | Ada penguji yang tidak tahu alur atau cara kerja dari program yang di *test* |

### 2. *White-box Testing*

**Tabel C.2 Keuggulan dan Kekurangan Black-box Testing**

| No. | Keunggulan | Kekurangan |
|---|---|---|
| 1 | Karena mengetahui kode sehingga mengetahui cara kerja dan dapat lebih mudah untuk melakukan *testing* | Dibutuhkan penguji yang terampil untuk melakukan *testing* |
| 2 | Membantu dalam mengoptimalkan *code* yang digunaknan | Memerlukan *tool* tersusun untuk pengujian efisiensi *code* dan alat *debuging* |
| 3 | Dapat langsung mengubah *code* yang mengakibatkan *error* | |

## D. Level Testing

Secara garis besar *level testing* terbagi menjadi dua yaitu *functional* dan *non functional testing*.

***Functional Testing***

*Functional testing* bedasar pada metode *black box testing* yang berguna untuk mengevaluasi softwate agar sesuai dengan keinginan. Seperti halnya namanya yang di *test* adalah fungsi fungsi yang terdapat didalam aplikasi. Didalam *fungtional testing* terdapat bermacam macam jenis *testing* berikut adalah jenis jenis *testing*-nya :

1. *Unit Testing*

   *Unit testing* adalah jenis *testing* yang dilakukan oleh penguna *software* sebelum diserahkan ke *tester*. Hal ini bertujuan untuk melihat kembali apakah *program* sudah sesuai dengan persyaratan dan fungsionalitasnya.

2. *Integration Testing*

   Pada *Integration testing* disini versi terbaru dari sebuah aplikasi dilihat integritasnya atau kesamaan tata letak *tool* dan *button* serta symbol-simbol yang ada dengan aplikasi sebelumnya. Ada dua cara melakukan tes ini yaitu dengan *bottom-up* atau dengan *top-down*.

3. *System Testing*

   Pada tahap ini seluruh *system* diuji secara menyeluruh dari setiap komponennya yang dilakukan oleh tim kusus. Tim ini berfungsi untuk menguji keseluruhan *system* apakah sudah memenuhi spesifikasi fungsional dan teknis.

4. *Regression Testing*

   Dalam setiap *update* dari sebuah apliaksi bisa terjadi perubahan pada bagian lain dalam apliaksi tersebut. Disini ditinjau kembali apakah terdapat *bug* yang akan mempengaruhi bagian lain aplikasi tersebut.

5. *Acceptance Testing*

   Pada saat ini *testing* tidak hanya dilakukan untuk mengecek kesalahannya saja tetapi tampilan dan keakuratan *software*. Pada tahap ini seleksi dilakukan oleh tim penjaminan mutu aplikasi. Pengecekan juga di dasarkan pada jika sebuah *bug* muncul maka tidak akan mengakibatkan *crashes* atau *error* yang fatal pada aplikasi.

6. *Alpha Testing*

   Tes ini adalah melakukan simulasi di kalangan internal *developer* untuk

melakukan tes pada *system* .

7. *Beta Testing*

   Pada tahap ini tes dilakukan ditempat asli atau tempat *software* tersebut akan digunakan. Kemudian pada tahap ini juga jika sudah lolos uji maka dapat digunakan secara luas dan langsung oleh *user* apliaksi.

***Non-Functional Testing***

*Non-Functional testing* adalah pengujian *Non-Functional* atribut. Hal hal yang dites adalah hal yang penting dalam kenyamanan *user* seperti kecepatan atau ketahanan aplikasi tersebut dan kesesuaianya dengan hardware yang dipakai. Berikut merupakan testing aplikasi *non-functional* :

1. *Performance Testing*

   Pada tahap ini hal yang dites adalah performa dari aplikasi tersebut seperti kecepatan mengeksekusi sebuah apliaksi , data *rendering*, dan masih banyak lagi.

2. *Load Testing*

   Pada tahap ini aplikasi diujicoba sampai tingkat *load* maksimum dari *software* tersebut. Apakah sesuai dengan kebutuhan atau masih belum.

3. *Stress Testing*

   *Stress Testing* digunakan untuk mengetes *software* di atas kondisi *normal*, sebagai contoh data yang di *load* kedalam *software* di-*input* lebih banyak disbanding seharusnya.

4. *Usability Testing*

   *Usability testing* merupakan *black box method* yang digunakan bukan untuk mengidentifikasi *error* namun untuk mengidentifikasi baik atau buruk desain dari aplikasi tersebut.

5. *Security Testing*

   Tes ini dilakukan dengan cara mencoba membobol masuk kedalam *System* aplikasi sehingga didapatkan celah yang kemudian akan diperbaiki kembali.

6. *Portability Testing*

   Pengujian ini ditujukan untuk mencari tau apakah *software* yang telah dibuat

ini apakah dapat digunakan kembali atau disatukan ke dalam *software* yang lainnya [10].

## E. Pengujian Otomatis

Pengujian otomatis adalah teknik yang menggunakan perangkat lunak yang sepesial karena memiliki tool untuk menjalankan *test case* secara otomatis. Berbeda dengan pengujian manual yang dilakukan oleh manusia yang duduk di depan komputer dan secara hati-hati menjalankan pengujian secara *step by step*. *Software* pengujian otomatis memiliki *tool* untuk dapat merekam dan memutar kembali *test case* yang akan dikerjakan. Selain itu juga dapat mengubah data pada *test case* yang sudah ada atau yang baru dijalankan. Hal tersebut dimaksutkan agar mendapatkan hasil *test* yang lebih detail. Pengujian secara otomatis juga mengurangi interaksi dengan manusia sehingga dapat mengurangi atau menghulangkan kebutuhan akan pengujian manual. Pengujian otomatis biasanya dibutuhkan untuk menghemat uang dan sumber daya manusia[14].

## F. Aplikasi Pengujian Otomatis

Dalam mempercepat proses *testing* program aplikasi maka menggunakan aplikasi yang dapat melakukan *automation testing* adalah pilihan yang tepat untuk itu ini adalah beberapa aplikasi *automation testing* :

### 1. Selenium Integrated Development Environment (IDE)

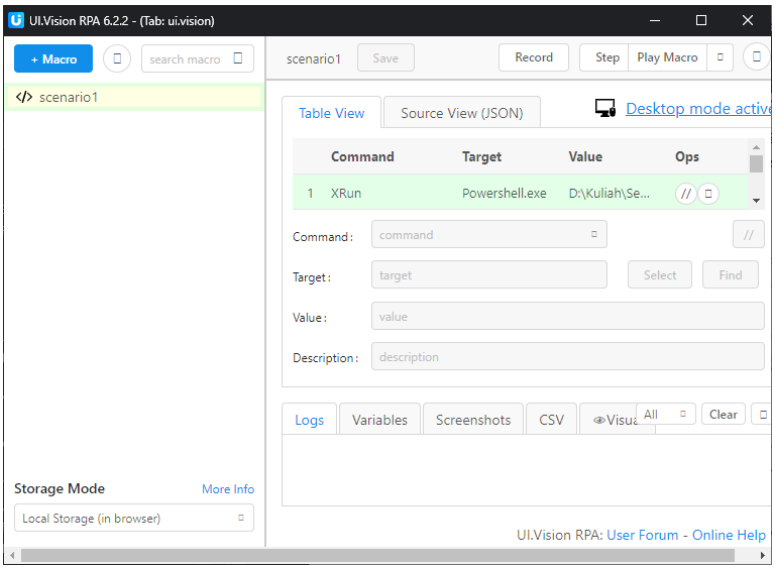
Selenium IDE adalah sebuah *open source web automation testing tool*. Sistem ini tidak membutuhkan logika pemrograman pada saat pengujian. Selenium IDE hanya merekam interaksi yang sedang dilakukan. Software ini juga dapat memutar kembali hasil rekaman yang telah dia simpan. Aplikasi ini banyak diterapkan untuk pengujian pada platform Web atau pada aplikasi *web automation testing* seperti UI.Vision [13].

### 2. UI.Vision

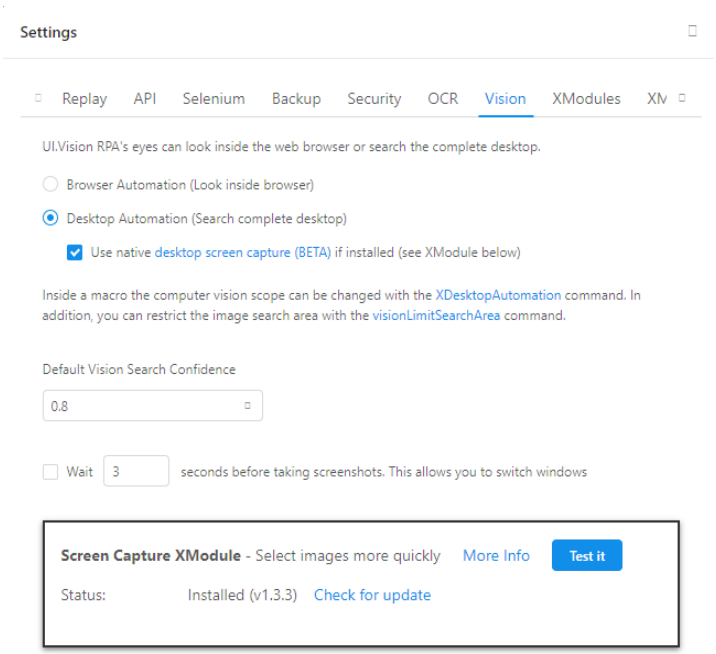

**Gambar F.1 Logo UI Vision**

Aplikasi ini adalah sebuah Platform yang berbasis Selenium IDE. Dengan mengunakan Selenium IDE yang biasanya digunakan untuk melakukan automation

testing untuk sebuah website. Karena sering digunakan untuk website aplikasi ini dijalankan pada *web browser*. Tetapi untuk UI.Vision ini dapat digunakan untuk melakukan automation testing pada desktop juga. Untuk dapat melakukannya aplikasi ini mengunakan teknologi pengenalan gambar dan tulisan dengan cara melakukan *screen scraping*. Dengan metode tersebut aplikasi ini dapat berjalan secara cross patform baik di Windows, Linux, dan MacOS [11] .




**Gambar F.2 Tampilan Aplikasi UI.Vision**




***Gambar F.3* Tampilan xModule untuk Pengenalan Gambar**