**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNOLOGI INFORMASI**

**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**

**USULAN TUGAS AKHIR**

# **IDENTITAS PENGUSUL**

**Nama** : **Simson Partogi Situmeang**

**NRP** : **5108 100 086**

Dosen Wali : Umi Laili Yuhana, S.Kom, M.Sc.

1. **JUDUL TUGAS AKHIR**

***Web Automated Testing Framework menggunakan Ruby dan library Watir-WebDriver dengan Pendekatan Hybrid-Testing.***

1. **LATAR BELAKANG**

Perkembangan teknologi informasi dan komunikasi sangat pesat. Perkembangan ini juga didukung dengan pengembangan perangkat lunak yang cepat. Pengembangan perangkat lunak terdiri dari berbagai macam tahapan, pada umumnya yaitu analisis, desain, implementasi, testing, dan perawatan. Setiap tahap dalam pengembangan perangkat lunak memiliki peran masing-masing dan harus dijalankan secara tepat demi kelancaran proyek. Dengan demikian, proyek pengembangan perangkat lunak akan dapat diselesaikan tepat waktu dan tepat biaya. Terdapat berbagai metode yang digunakan untuk meningkatkan kinerja dalam proyek pengembangan perangkat lunak sehingga dapat mencapai sasaran serta meningkatkan produktivitas.

Kini, aplikasi berbasis web banyak dikembangkan. Hal ini disebabkan karena aplikasi berbasis web memiliki beberapa keunggulan dibandingkan dengan aplikasi desktop. Beberapa keunggulan aplikasi web, yaitu:

1. Berjalan di berbagai sistem operasi
2. Mudah diubah
3. Tidak memerlukan bermacam-macam instalasi tambahan, cukup dengan web browser saja.

Dalam menangani proyek berbasis web, diperlukan sistem pendukung untuk mempercepat pengerjaan proyek dan menjaga kualitas produk. Salah satu tahap yang dapat dipercepat adalah tahap testing. Di tahap testing ini diharapkan dapat dilakukan secara cepat dengan tidak mengabaikan kualitas dari produk. Testing yang dilakukan pada proyek berbasis web seringkali diulangi berkali-kali selama pengerjaan proyek. Tentu saja hal ini memerlukan biaya, tenaga, dan waktu yang lebih.

Tes secara otomatis adalah salah satu cara untuk meningkatkan kinerja tester dalam tahap testing. Dengan adanya otomasi tes, testing dapat dilakukan secara cepat, rapi, dan mudah dikembangkan lebih lagi. Otomatisasi tes cocok diterapkan dalam proyek pengembangan aplikasi berbasis web. Dengan otomatisasi tes, tester dimudahkan dalam melakukan testing terutama pada kasus-kasus yang menggunakan data yang banyak serta berulang-ulang.

Pembangunan otomatisasi tes yang cepat memerlukan suatu framework yang mendukungnya, yaitu web automated testing (WAT) framework. Dengan adanya framework WAT ini, tester akan lebih mudah dan cepat dalam membuat otomatisasi tes. Framework WAT digunakan dalam pembacaan data masukan, pelaksanaan langkah-langkah testing, hingga pencatatan hasil keluaran testing.

Bahasa pemrograman Ruby dipilih sebagai bahasa yang digunakan dalam framework WAT ini. WAT menggunakan Ruby karena Ruby memiliki fitur-fitur lengkap yang diperlukan oleh WAT. Selain itu, Ruby merupakan bahasa pemrograman yang mudah dipelajari.

WAT yang diujikan pada aplikasi berbasis web ini menggunakan library Watir-WebDriver. Watir-WebDriver adalah open source library dari Ruby yang merupakan pengembangan dari Watir (*Web Application Testing in Ruby*). Watir-WebDriver menjalakan otomasi pada browser. Dengan demikian WAT dapat dikembangkan sedemikian rupa sehingga mampu menjadi framework yang digunakan untuk menjalankan otomatisasi tes pada aplikasi berbasis web.

Pembangunan otomatisasi tes memiliki beberapa pendekatan, antara lain data-driven testing, modularity testing, keyword-driven testing, hybrid testing, model-based testing. Framework WAT sendiri menggunakan pendekatan hybrid-testing. Hybrid testing merupakan gabungan dari pendekatan modularity testing dan keyword testing. Dengan pendekatan hybrid testing, tester memungkinkan untuk melakukan kustomisasi pada untuk meningkatkan peforma dari WAT.

1. **RUMUSAN MASALAH**

Berdasarkan latar belakang yang telah diuraikan diatas, maka dapat diambil kesimpulan sebagai berikut:

* Dibutuhkan framework yang membantu dalam mempermudah tester membuat otomatisasi tes pada aplikasi berbasis web.

1. **BATASAN MASALAH**

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, diantaranya sebagai berikut :

1. Framework berjalan pada sistem operasi Windows.
2. Framework dijalankan dengan interpreter Ruby versi 1.9.2 dilengkapi dengan library Watir-WebDriver.
3. Framework ditujukan pada otomatisasi browser Mozilla Firefox versi 4.0 atau versi yang lebih baru.
4. Framework menerima input berkas *spreadsheet* dalam format Microsoft Office Excel.
5. Hasil output dari framework ini berupa *spreadsheet* dalam format Microsoft Office Excel sesuai dengan *template* yang disediakan serta berkas gambar dengan format jpg.
6. Framework hanya dapat dijalankan tanpa proxy.
7. Framework diujikan dengan membuat aplikasi testing pada website tertentu.
8. **TUJUAN TUGAS AKHIR**

Tugas Akhir ini memiliki tujuan sebagai berikut :

* Membuat framework otomatisasi tes pada aplikasi berbasis web menggunakan bahasa pemrograman Ruby dengan library Watir-WebDriver.

1. **MANFAAT TUGAS AKHIR**

Manfaat yang diharapkan dengan adanya perancangan dan pembangunan framework Tugas Akhir ini adalah secara umum dapat membangun sebuah framework untuk pembangunan otomatisasi tes pada aplikasi berbasis web. Dengan adanya framework WAT ini diperoleh beberapa manfaat antara lain:

* 1. Memudahkan tester dalam pembangunan aplikasi otomatisasi tes bagi aplikasi berbasis web.
  2. Adanya fitur-fitur pendukung yang disediakan oleh framework.
  3. Pembuatan aplikasi testing menjadi lebih cepat sehingga dapat pula mempercepat pembangunan proyek perangkat lunak.
  4. Tidak mengeluarkan biaya lagi untuk membeli kakas testing otomatis.

1. **DASAR TEORI**

Terdapat beberapa teori yang mendasari pengerjaan tugas akhir ini, diantaranya adalah:

* 1. **Automation Testing**

Setiap pengembang perangkat lunak melakukan pengujian atau tes terhadap produk mereka. Namun, seringkali ditemukan kekurangan dari produk mereka. Tester bertugas untuk menemukan kesalahan tersebut sebelum produk perangkat lunak diproduksi. Penggunaan automation testing dapat meningkatkan efektifitas, efisiensi, dan memperlebar cakupan dari pengujian perangkat lunak.

Automation testing adalah suatu metode penggunaan perangkat lunak untuk melakukan testing yang termasuk pembandingan hasil keluaran, pengaturan kondisi awal, dan pembuatan laporan. Sekalipun kita dapat melakukan testing secara manual, namun hal ini bias menjadi tidak efektif karena memerlukan waktu yang tidak sedikit. Dengan adanya automation testing, kita dapat melakukan testing dengan cepat dan berulang.

Ada dua pendekatan secara umum yang diterapkan dalam automation test[1]:

1. Code-driven testing

Cara testing ini dilakukan dengan memberikan bermacam-macam input ke dalam kelas, modul, atau fungsi-fungsi kemudian memverifikasi hasil keluarannya apakah sesuai atau tidak.

1. Graphical user interface testing

Pendekatan ini dilakukan dengan cara menciptakan event antarmuka pengguna seperti klik, pengisian data, dan pengamatan pada objek tertentu kemudian memverifikasi hasil keluaran program sesuai dengan yang diharapkan atau tidak.

Kakas yang digunakan untuk melakukan automation testing ini terbilang mahal. Namun, bila digunakan untuk jangka waktu yang panjang, biaya yang diperlukan untuk automation testing ini terbilang cukup efektif, terutama jika kita sering melakukan testing dalam pengembangan perangkat lunak. Setiap kakas automation testing biasanya digunakan pada lingkungan tertentu.

|  |  |
| --- | --- |
| Nama Kakas | Produsen |
| HP QuickTest Professional | HP |
| IBM Rational Functional Tester | IBM Rational |
| Parasoft SOAtest | Parasoft |
| QF-Test | Quality FirstSoftware GmbH |
| Ranorex | Ranorex GmbH |
| Rational robot | IBM Rational |
| Selenium | Open source |
| HTTP Test Tool | Open source |
| SilkTest | Micro Focus |
| TestArchitect | LogiGear |
| TestComplete | SmartBear Software |
| Testing Anywhere | Automation Anywhere |
| TestPartner | Micro Focus |
| TOSCA Testsuite | TRICENTIS Technology & Consulting |
| Visual Studio Test Professional | Microsoft |
| WATIR | Open source |
| WebUI Test Studio | Telerik, Inc. |

Infrastruktur dari kakas automation testing berbeda-beda sesuai dengan yang dibutuhkan oleh pengembang. Beberapa infrastrukutur dari automation testing antara lain:

1. Data-driven testing

Data-driven testing menekankan pada interaksi antara skrip tes dengan datanya. Variabel digunakan sebagai nilai masukan dan keluaran, navigasi melalui program, membaca sumber data, dan pencatatan status tes. Alur logika tes dieksekusi di skrip dengan menggunakan nilai dari data-data.

1. Modularity testing

Skrip tes dari framework yang modular memerlukan:

* Pembuatan skrip yang kecil, independen, mewakili modul-modul, bagian-bagian, dan fungsi-fungsi aplikasi yang dites.
* Skrip kecil ini kemudian digunakan secara hirarkis untuk membangun tes yang besar, menyesuaikan dengan kasus tes tertentu.

Infrastruktur ini meningkatkan *maintainability* dan skalabilitas dari automated test suites.

1. Keyword-driven testing

Keyword-driven testing juga dikenal sebagai table-driven testing atau action-word testing, adalah metodologi testing perangkat lunak untuk automated testing yang memisahkan proses pembuatan tes menjadi dua tahap yang berbeda, yaitu tahap perencanaan dan tahap implementasi.

* Tahap Perencanaan

Pembuatan tabel yang terdiri dari beberapa kata kunci, misal:

|  |  |  |
| --- | --- | --- |
| Langkah | Aksi | Data |
| 1 | Login | DataLogin |

* Tahap Implementasi

Menyatakan aksi dalam bentuk skrip tes.

1. Hybrid testing

Framework otomasi tes hybrid adalah infrastruktur yang paling sering digunakan untuk bermacam-macam projek. Framework ini menggabungkan keyword-driven testing dengan data-driven testing. Infrastruktur ini mampu menutupi kekurangan yang biasa terjadi pada infrastruktur lainnya.

1. Model-based testing

Model-based testing adalah aplikasi dari desain berbasis model untuk pendesainan dan pengeksekusian artifak yang diperlukan untuk melakukan testing perangkat lunak. Hal ini dilakukan dengan membuat sebuah model yang mendeskripsikan semua aspek dari data testing, terutama test case dan lingkungan eksekusi tes.

Automation testing memiliki beberapa keunggulan daripada manual testing, yaitu[2]:

1. Menghemat waktu testing
2. Meningkatkan akurasi testing
3. Meningkatkan jangkuan testing
4. Membantu tim pengembang dan tester
   1. **Ruby**

Ruby merupakan bahasa yang seimbang. Pencipta Ruby, **Yukihiro Matsumoto**, menggabungkan bagian-bagian dari bahasa-bahasa favorit beliau (*Perl, Smalltalk, Eiffel, Ada* dan *Lisp*) untuk membentuk bahasa baru yang seimbang antara pemrograman fungsional dengan pemrograman imperatif[3].

Sejak Ruby pertama kali dirilis ke publik pada tahun 1995, banyak programmer profesional dari seluruh dunia serius ikut mengembangkan Ruby. Pada tahun 2006, Ruby diterima oleh banyak orang. Dengan komunitas pengguna Ruby yang aktif di banyak kota-kota di seluruh dunia dan konferensi-konferensi beserta pertemuan Ruby terkait.

Ruby juga sepenuhnya bebas. Tidak hanya gratis, tetapi juga bebas untuk menggunakan, memodifikasi dan mendistribusikan Ruby.

Di Ruby, semua adalah obyek. Setiap informasi dan kode bisa diberi *property* dan *action*. Pemrograman berorientasi obyek memanggil *property* dengan nama variabel instan dan *action*, yang disebut sebagai metode. Pendekatan murni berorientasi obyek terutama terlihat pada demonstrasi sedikit kode yang diberikan pada *number*.

Di banyak bahasa-bahasa lain, number dan tipe primitif bukan obyek. Ruby mengikuti pengaruh bahasa Smalltalk dengan memberikan metode dan variabel instan pada semua tipe. Ini memudahkan menggunakan Ruby, karena peraturan-peraturan mengenai obyek semua berlaku pada Ruby.

Ruby dianggap sebagai bahasa yang fleksibel, karena bagian-bagian dari Ruby bisa diubah-ubah dengan bebas. Bagian-bagian yang esensi di Ruby bisa dihapus maupun didefinisikan ulang. Bagian-bagian yang sudah ada bisa ditambahkan. Ruby mencoba untuk tidak membatasi programer.

Demi kemudahan, operator-operator Ruby juga termasuk metode. Anda juga bisa mendefinisikan ulang operator.

Blok Ruby juga dianggap sebagai sumber kekuatan Ruby yang sangat fleksibel. Programer dapat menyertakan *closure* pada setiap metode, menjelaskan bagaimana metode yang bersangkutan seharusnya berperilaku. *Closure* disebut blok dan telah menjadi satu diantara banyak fitur-fitur Ruby yang paling populer pada banyak pendatang baru Ruby dari bahasa-bahasa imperatif lain seperti *PHP* atau *Visual* *Basic*.

Ruby kaya fitur, antara lain sebagai berikut:

* Ruby memiliki fitur-fitur yang menangani *exception*, seperti *Java* atau *Python*, untuk mempermudah menangani *error*.
* Ruby menyediakan *mark-and-sweep garbage collector* untuk semua obyek Ruby.
* Menulis *extension* *C* di Ruby lebih mudah daripada di *Perl* ataupun di *Python*, dengan *API* yang elegan untuk memanggil Ruby dari *C*. Ini termasuk memanggil Ruby *embedded* di perangkat lunak, untuk digunakan sebagai bahasa *scripting*. Antarmuka *SWIG* juga tersedia.
* Ruby bisa memuat *library* *extension* secara dinamis jika Sistem Operasi mengijinkan.
* Ruby menyediakan fitur *OS threading* yang *independent*. Maka, untuk semua platform dimana Ruby berjalan, Anda juga punya multithreading, terlepas dari apakah Sistem Operasi mendukung multithreading atau tidak, bahkan pada *MS-DOS* sekalipun!
* Ruby sangat *portable*: Ruby kebanyakan dikembangkan di *GNU/Linux*, tetapi juga berjalan di banyak tipe *UNIX, Mac OS X, Windows 95/98/Me/NT/2000/XP, DOS, BeOS, OS/2*, dan lain-lain.
  1. **Watir-WebDriver**

Watir-WebDriver adalah library pengembangan dari library Watir yang mampu berjalan pada berbagai web browser. Sedangkan Watir sendiri adalah sebuah library Ruby yang *open-source* (lisensi BSD) yang digunakan untuk mengotomasi web broser. Kata Watir sendiri berasal dari kalimat “Web Application Testing in Ruby”. Watir-WebDriver membantu kita dalam membuat tes yang mudah dibaca dan diolah. Library ini simpel dan fleksibel.

Watir-WebDriver mampu menjalankan browser sama seperti yang dilakukan manusia. Watir dapat mengeklik tautan, mengisi form, menekan tombol. Watir-WebDriver juga mampu mengecek hasil dari otomasinya, apakah teks atau elemen yang diharapkan terdapat di halaman web atau tidak.

Watir-WebDriver merupakan keluarga dari library Ruby, tapi library ini mampu mendukung aplikasi Anda tanpa bergantung teknologi apa yang Anda gunakan. Watir-WebDriver mendukung browser Chrome, Firefox, Internet Explorer, dan Opera[4].

Watir-WebDriver menggunalan bahasa pemrograman Ruby. Seperti halnya bahasa pemrograman lainnya, Ruby memampukan kita untuk berkoneksi dengan basis data, membaca file data dan *spreadsheet*, membuat XML, dan membuat library yang dapat digunakan kembali (*reusable*). Tidak seperti bahasa pemrograman lainnya, Ruby adalah bahasa pemrograman yang ringkas dan tidak jarang enak untuk dibaca.

Keunggulan Watir-WebDriver:

* Watir-WebDriver merupakan kakas yang open-source. Tidak perlu biaya untuk menggunakannya.
* Terdapat komunitas yang aktif dan berkembang di belakangnya.
* Watir-WebDriver menggunakan Ruby, skrip bahasa yang memiliki banyak fitur.
* Watir-WebDriver mendukung apliakasi web Anda, tak peduli teknologi apa yang Anda gunakan.
* Mendukung multi browser pada platform yang berbeda.
* Watir-WebDriver tangguh dan mudah digunakan, namun sangat ringan.

1. **RINGKASAN TUGAS AKHIR**

Framework dari web automated testing (WAT) menggunakan pola desain Hybrid Testing yang mencakup data-driven, table-driven (keyword-driven), dan modularity-driven testing. Dengan pola framework seperti ini, WAT sangat dimungkinkan untuk digunakan berulang kali pada bermacam-macam proyek. Gambar di bawah ini akan menunjukkan bagaimana cara kerja WAT Framework ini:



Gambar 1 Workflow WAT Framework

Infrastruktur table-driven (keyword-driven) testing menggunakan kata kunci (keyword), dan kode skrip tes yang akan menjalankan aplikasi yang dites beserta data-datanya. File berinfrastruktur keyword-driven antara lain seperti konfigurasi utama testing, skenario testing, serta konfigurasi elemen tambahan. File-file tersebut cocok menggunakan infrastruktur keyword driven.

Data-driven testing adalah infrastruktur di mana masukan tes dan nilai keluaran yang diharapkan dibaca dari data file lainnya yang akan dimuat dalam variabel program. Pada infrastruktur ini variabel dapat digunakan untuk menyimpan nilai masukan maupun nilai keluaran yang akan digunakan pada verifikasi akhir. Pada framework WAT ini, infrastruktur Data-driven diterapkan dalam file skenario testing pada worksheet bagian data input.

Framework WAT menggabungkan kedua model tersebut sehingga membuat framework ini semakin lengkap. Skrip kode hanya menjalankan testing (*driver*), sedangkan pengendali utamanya adalah table-driven yang memuat masukan dari data-driven. Tester memungkinkan untuk melakukan kustomisasi pada framework ini untuk meningkatkan peforma dari WAT ini sendiri.

Dengan adanya framewowk WAT ini, Tester akan mendapatkan kemudahan, antara lain sebagai berikut:

* + - 1. Memudahkan dalam konfigurasi testing.

Konfigurasi testing akan diatur dalam file *spreadsheet* konfigurasi utama. Tester tidak perlu melakukan modifikasi kode program jika tidak ingin melakukan perubahan alur testing.

* + - 1. Memudahkan dalam pembuatan skenario testing.

Skenario testing dibuat pada file *spreadsheet* yang berinfrastruktur keyword-driven pada pengaturan aksi testing dan berinfrastruktur data-driven pada pengaturan masukan data.

* + - 1. Memudahkan dalam penanganan popup.

Penanganan popup dapat dilakukan dengan library Watir, namun diperlukan penulisan kode yang cukup rumit. Framework WAT akan membantu penanganan popup sehingga, pengkodean dilakukan menjadi lebih sederhana.

* + - 1. Memudahkan dalam pengaksesan file *spreadsheet*.

Pengaksesan file-file *spreadsheet* yang digunakan dalam testing tidak memerlukan pengkodean yang rumit. Tester hanya perlu mengikuti standarisasi penulisan tabel yang diperlukan untuk testing.

* + - 1. Adanya fitur pendukung.

Framework WAT dilengkapi dengan fitur *screenshot* yang memudahkan dalam pendokumentasian.

1. **METODOLOGI**
2. Penyusunan Proposal Tugas Akhir

Tahap awal untuk memulai pengerjaan Tugas Akhir adalah Penyusunan Proposal Tugas Akhir. Pada proposal ini, diajukan gagasan pembuatan framework otomatisasi tes pada aplikasi berbasis web.

1. Studi Literatur

Pada tahapan ini akan dilakukan studi literatur, diantaranya :

* + - * 1. Automation Testing
        2. Ruby
        3. Watir-WebDriver

1. Analisa Kebutuhan Perangkat Lunak

Pada tahapan ini akan dilakukan analisa terhadap kebutuhan – kebutuhan apa saja yang dimiliki oleh sistem yang akan dijadikan sebagai lingkungan uji coba, dalam hal ini adalah Interpreter Ruby dan library Watir-WebDriver.

1. Perancangan Perangkat Lunak

Pada tahap ini dilakukan untuk perancangan arsitektur aplikasi dan fitur-fitur yang akan dibuat sesuai dengan informasi yang diperoleh dari hasil analisa kebutuhan perangkat lunak.

1. Implementasi

Implementasi merupakan tahap penerapan framework menjadi suatu aplikasi testing yang akan dijalankan.

1. Pengujian dan Evaluasi

Pada tahap ini dilakukan uji coba terhadap pengimplementasian framework.

1. Penyusunan Buku Tugas Akhir

Tahap terakhir merupakan penyusunan laporan yang memuat dokumentasi mengenai pembuatan serta hasil dari implementasi perancangan dan prototyping yang telah dibuat. Secara garis besar, buku laporan tugas akhir ini terdiri atas beberapa bagian yaitu:

1. Pendahuluan
   1. Latar Belakang
   2. Permasalahan
   3. Batasan Tugas Akhir
   4. Tujuan
   5. Metodologi
   6. Sistematika Penulisan
2. Tinjauan Pustaka
3. Desain dan Implementasi
4. Uji Coba dan Evaluasi
5. Kesimpulan dan Saran
6. Daftar Pustaka
7. **JADWAL KEGIATAN TUGAS AKHIR**

Tugas akhir ini diharapkan bisa dikerjakan menurut jadwal sebagai berikut:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| No. | Kegiatan |  | | Bulan | | | | | | | |
| September | | Oktober | | November | | Desember | | Januari | |
| 1. | Penyusunan Proposal Tugas Akhir |  | |  |  |  |  |  |  |  |  |
| 2. | Studi Literatur |  |  |  | |  |  |  |  |  | |
| 3. | Implementasi |  | |  |  |  |  |  |  |  |  |
| 4. | Pengujian dan Evaluasi |  | |  |  |  |  |  |  |  |  |
| 5. | Penyusunan Buku Tugas Akhir |  | |  |  |  |  |  |  |  |  |

1. **DAFTAR PUSTAKA**

[1] <http://en.wikipedia.org/wiki/Test_automation>. Diakses pada 26 Agustus 2011 10:00.

[2] Kerry Zallar, *Automated Software Testing - A Perspective*, 2004.

[3] <http://www.ruby-lang.org/id/about/>. Diakses pada 25 Agustus 2011 11:44.

[4] <http://www.watirwebdriver.com>. Diakses pada 26 Agustus 2011 10:00.

|  |  |
| --- | --- |
| Lembar Pengesahan | |
| Proposal Tugas Akhir | |
|  |  |
| Web Automated Testing Framework menggunakan Ruby dan library Watir-WebDriver dengan Pendekatan Hybrid-Testing | |
| (2115) | |
|  |  |
| SIMSON PARTOGI SITUMEANG | |
| 5108100086 | |
|  |  |
| Surabaya, 6 Oktober 2011 | |
|  |  |
| Menyetujui, | |
| Dosen Pembimbing 1 | Dosen Pembimbing 2 |
|  |  |
| SITI ROCHIMAH | -- |
|  |  |
|  |  |
| NIP. 132103631 | NIP. 000000000 |