**IMPLEMENTASI *UNIT TESTING* PADA *WEBSITE***

# BAB I

# PENDAHULUAN

## Latar Belakang

*Website* menjadi salah satu bagian dari internet yang dapat dimanfaatkan sebagai laman informasi dan transaksi. *Website* adalah serangkaian Elemen yang terdiri dari teks, gambar, audio, dan animasi sehingga membentuk media informasi yang menarik untuk dikunjungi (Adelheid, 2015). Halaman *web* dapat diakses menggunakan teknologi bernama *web server* yang berfungsi untuk menyediakan halaman *web* (Solichin, 2016). Perkembangan *Website* telah mengalami transformasi yang signifikan selama beberapa dekade terakhir. Dari tampilan statis, *Website* saat ini telah menjadi *platform* interaktif yang dapat memberikan pengalaman pengguna untuk melakukan interaksi, saling membagikan informasi, dan bertransaksi secara *real-time*. Perkembangan teknologi *web* menjadi sorotan berbagai industri untuk mengikuti perkembangan teknologi. *Website* telah menjadi salah satu alat utama yang mendukung perusahaan dalam mencapai *audiens* global, meningkatkan efisiensi operasional, dan mengoptimalkan pengalaman pelanggan.

Dalam era digital ini, *Website* juga memfasilitasi interaksi dengan pelanggan, dengan fitur-fitur seperti *chat online* dan formulir kontak yang memungkinkan perusahaan memberikan dukungan pelanggan secara instan. Selain itu, *Website* memungkinkan perusahaan untuk mengumpulkan data berharga mengenai perilaku

pelanggan, yang dapat digunakan untuk meningkatkan strategi pemasaran dan pengembangan produk. Rincian diatas membantu perusahaan dalam meningkatkan daya saingnya di pasar yang semakin kompetitif (Sullivan, A., 2022). *Website* memiliki fungsi yang sesuai dengan tujuan pembuatannya. Pengembangan *Website* pada perusahaan menjadi salah satu faktor penting penentu keberhasilan kegiatan operasional perusahaan, setelah adanya pengembangan *Website* sesuai kebutuhan, perlu dilakukan *testing* agar pengembangan dapat diuji dan dikaji untuk mengetahui bahwa *output* terbaru sesuai dengan kebutuhan yang telah ditentukan.

*Testing* adalah suatu proses yang penting dalam menilai kualitas suatu perangkat lunak sebelum perangkat lunak tersebut dapat diperkenalkan atau dirilis. Proses ini penting untuk memberikan keyakinan kepada pelanggan atau pengguna bahwa perangkat lunak tersebut telah mencapai standar yang memadai dan memenuhi persyaratan kinerja yang dibutuhkan oleh pelanggan atau pengguna (J. F. Andry & Reinaldo, 2017) *Testing* memiliki peran penting pada pembuatan maupun pengembangan *Website*, *testing* berperan untuk menguji kinerja *Website* guna mengetahui bahwa *output*  yang diharapkan sesuai dengan yang diharapkan sesuai *flow* dan perencanaan pembuatan *Website.* Terdapat beberapa jenis *testing,* algoritma *generate* dan *test case* menjadi algoritma paling sederhana (Amri & Safwati, 2019). Pada algoritma ini terdapat dua proses prosedur yaitu *generate* dimana akan dilakukan pembangkitan semua solusi yang mungkin dan *test* untuk menguji mengenai solusi yang telah dibangkitkan sebelumnya. Algoritma *generate* dan *test case* merupakan penggabungan algoritma DFS dengan adanya pelacakan mundur yaitu *backtrscking* dimana *flow* akan bergerak mundur menuju pada bagian *state* awal. Algoritma *generate* dan *test case* dapat disimpulkan sebagai adanya pembangkitan semua solusi yang mungkin, dilakukan pengujian tiap *node* solusi dengan melakukan perbandingan node sebelumnya dengan hasil *node* yang ada diakhir dari lintasan yang ditentukan atas rangkaian tujuan yang diinginkan. Selanjutnya, apabila solusi sudah diketahaui, maka sistem selesai, apabila solusi belum ditemukan maka akan kembali ke langkah pertama.

*Test case* adalah metode pengujian perangkat lunak yang melibatkan rangkaian skenario eksekusi. Fungsinya adalah untuk menilai sejauh mana *Website* yang sedang dikembangkan memenuhi spesifikasi yang telah ditentukan oleh Perusahaan Columbia Catanta. Walaupun pengujian tidak dapat secara mutlak membuktikan kebenaran dalam semua kemungkinan eksekusi pengujian, pendekatan ini dapat mendekati keakuratan dengan merencanakan dan merancang *test case* secara cermat. Dengan demikian, *test case* dapat memberikan keyakinan akan efektivitas perangkat lunak hingga pada tingkat tertentu sesuai dengan harapan.

Penelitian ini bertujuan untuk menerapkan *unit testing* dengan menggunakan algoritma *generate* dan *test case* pada *website* Columbia Catanta. *Framework* yang digunakan dalam pengujian adalah PHPUnit. Hasil yang diharapkan dari penelitian ini adalah untuk mengetahui keberhasilan *unit testing* berdasarkan analisis data hasil pengujian.

## Rumusan Masalah

Berdasarkan perumusan masalah yang telah dipaparkan, maka permasalahan yang dapat dirumuskan dalam penelitian ini adalah sebagai berikut:

1. Bagaimana implementasi *unit testing* pada *Website Columbia Catanta* menggunakan algoritma *generate* dan *test case?*
2. Bagaimana tingkat keberhasilan *unit testing* pada *Website Columbia Catanta* berdasarkan *test case* yang dibuat terhadap studi kasus yang digunakan?

## Tujuan Penelitian

Berdasarkan perumusan masalah yang telah dipaparkan, tujuan dari pelaksanaan penelitian ini adalah sebagai berikut:

1. Mengetahui hasil implementasi *unit testing Website Columbia Catanta* menggunakan algoritma *generate* dan *test case.*
2. Menganalisis tingkat keberhasilan *unit testing* pada *Website Columbia Catanta* berdasarkan *test case* yang dibuat terhadap studi kasus

## Manfaat Penelitian

Berdasarkan perumusan masalah yang telah dipaparkan, manfaat dari pelaksanaan penelitian ini adalah sebagai berikut:

1. Manfaat Teoritis:
2. Perkembangan Metodologi Pengujian: Penelitian ini akan berkontribusi pada perkembangan metodologi pengujian perangkat lunak dengan menguji dan memvalidasi penggunaan algoritma generate dan *test case*.
3. Pemahaman Peran *Unit testing*: Penelitian ini akan membantu dalam pemahaman lebih mendalam tentang peran *unit testing* dalam pengembangan *Website*.
4. Manfaat Praktis
5. Bagi Perusahaan

Diharapkan mampu mempermudah dan memfasilitasi *testing* pada *Website* Columbia Catanta.

1. Bagi Pengguna *Website* (client Columbia Catanta)

Manfaat penelitian dan implementasi ini diharapkan mampu mempermudah aktifitas client pada *Website* dengan adanya fitur terbaik dengan fungsi yang sesuai.

1. Bagi Penulis

Manfaat penelitian dan implementasi bagi penulis diharapkan mampu menerapkan dan mengimplimentasikan suatu inovasi *unit testing* pada pengembangan *Website*.

## Batasan Penelitian

Berdasarkan rumusan masalah diatas, maka penelitian

mengambil Batasan masalah, sebagai berikut:

1. Implementasi ini akan menyajikan *unit testing* pada *Website* Columbia Catanta menggunakan Bahasa pemrograman *php.*
2. Algoritma yang akan digunakan sebagai berikut:
3. *Generate*
4. *Test case*
5. *Unit testing* yang akan digunakan adalah *PHPUnit.*

# BAB II

# KAJIAN PUSTAKA

## Kajian Teori

### Penelitian Implementasi Unit testing pada Website

1. **Pengertian *Website***

*Website* merupakan suatu himpunan dari halaman web dimana himpunan tersebut mempunyai topik yang berkaitan satu sama lain, biasanya disimpan di sebuah web server, dan bisa dilakukan pengaksesan melalui jaringan internet atau jaringan lokal (LAN) (Yeni, 2019). Sedangkan menurut pendapat lain, *Website* adalah sekelompok *pages* yang dipergunakan untuk menampilkan beragam informasi, termasuk teks, gambar, animasi, audio, atau bahkan gabungan dari semua elemen tersebut (Bekti, 2015). Berawal pada 1989, seorang *programmer* komputer yang berasal dari Inggris dengan nama Tim Berners-Lee, bekerja di European Pysics Laboratory (CERN) di Genewa, Swiss, melakukan sesuatu yang inovatif. Ia menggabungkan konsep hypermedia dengan berbagai sumber informasi yang tersebar luas di internet. Sebelum adanya World Wide Web (WWW), internet memang memungkinkan berbagai aktivitas. Barulah pada tahun 1991, situs web pertama yang terhubung dengan jaringan muncul. Tim tersebut melakukan perancangan situs web dengan tujuan memberikan fasilitas pertukaran dan pembaruan informasi di dalam komunitas penelitian tempat mereka bekerja. Kemudian, pada tanggal 30 April 1993, CERN, tempat Tim Berners-Lee bekerja, mengumumkan bahwa WWW dapat dipergunakan dengan bebas bagi masyarakat umum (Kurniawan, 2019).

1. **Pengertian *Unit testing***

*Unit testing* adalah metode yang digunakan untuk menguji perangkat lunak dengan konsentrasi pada elemen paling kecil dari aplikasi tersebut. Dalam *unit testing*, perangkat lunak diuji secara terisolasi, biasanya dengan menguji komponen terkecil seperti fungsi atau bagian kode tertentu. Proses *unit testing* melibatkan pembuatan kode *unit testing* serta *test case* yang dilakukan dengan cara manual saat ingin melakukan pengujian, yang memerlukan waktu dan berisiko terhadap kesalahan (J. F. Andry & Reinaldo, 2017). Tujuan *unit testing* meliputi: Untuk memverifikasi keakuratan bagian kode, untuk menemukan dan mengatasi bug di awal pengembangan perangkat lunak untuk meningkatkan pemahaman programmer tentang basis kode, untuk dapat melakukan pengubahan dan pengembangan dengan lebih mudah, serta untuk memudahkan proses validasi pengujian output yang diharapkan.

*Unit testing* memiliki fokus kepada kegiatan verifikasi dari unit terkecil dalam desain pada perangkat lunak, seperti elemen atau modul pada perangkat lunak. Tiap unit dari perangkat lunak diuji untuk memeriksa *flow* masukan *(input)* dan keluaran *(output)* dari unit tersebut memenuhi spesifikasi yang diinginkan. Biasanya *unit testing* dilakukan ketika *source code* sedang pada proses pengembangan (Rosa & Shalahuddin, 2013). Pada penelitiannya dengan berjudul "*Unit testing*," Kua (2019) mengemukakan beberapa alasan mengenai seberapa penting penggunaan *unit testing*, berikut merupakan *point-*nya:

1. *Unit testing* membantu mengurangi waktu yang dibutuhkan saat kegiatan pencarian *bug* atau *debugging*, sehingga pengembang menjadi lebih percaya apabila perubahan yang telah dibuat buat dalam kode adalah benar.
2. Prinsip cakupan pengujian yang minimal memudahkan dalam menemukan letak kegagalan saat terjadi kesalahan.
3. *Unit testing* mendorong pengembang aplikasi untuk berpikir secara lebih mendalam tentang desain kode mereka, yang pada akhirnya menghasilkan desain yang lebih baik.
4. *Unit testing* memberikan alat regresi yang kuat, sehingga *developer* merasa memiliki dasar kepercayaan bahwa perubahan yang telah dibuat buat dalam kode tidak akan memberikan dampak yang negatif kepada fungsi keseluruhan aplikasi.
5. Penggunaan *unit testing* dapat memberikan pengurangan biaya di masa depan, mengingat banyak penelitian menunjukkan bila memperbaiki bug setelah aplikasi selesai dibangun jauh lebih mahal dibandingkan dengan mengidentifikasi dan memperbaiki mereka selama pengembangan awal.

Penjelasan tentang setiap tahap pengujian pada perangkat lunak diuraikan seperti berikut (Kartanti, 2015):

1. Pengujian Unit

Pada tahap ini, *unit testing* difokuskan pada kegiatan pemverifikasian komponen terkecil dalam desain perangkat lunak, seperti modul maupun komponen perangkat lunak. Berdasarkan pendapat (Kartanti, 2015), *unit testing* ini dilaksanakan oleh *developer* sendiri dan memanfaatkan metode pengujian *white box*.

1. Pengujian Integrasi

Pengujian Integrasi merupakan pengujian kepada masing-masing unit program dimana program tersebut saling melakukan interaksi atau terintegrasi untuk membangun arsitektur perangkat lunak.

1. Pengujian Validasi

Pada tahap ini, kegiatan akan berfokus kepada hasil keluaran sistem berdasarkan kegiatan yang dilaksanakan oleh *user*. Validasi dianggap berhasil apabila perangkat lunak beroperasi sesuai dengan harapan *user*.

1. Pengujian Sistem

Pengujian Sistem merupakan pengujian dengan tujuan utama untuk menguji keseluruhan sistem dengan berbasis komputer. Menurut Pressman (2010: 571), pengujian sistem mencakup pengujian keamanan, pengujian ketahanan, dan pengujian kinerja.

*Unit testing* menjadi salah satu teknik yang dapat dimanfaatkan guna melakukan kegiatan pengujian pada perangka lunak dan memiliki fokus pada bagian terkecil dari aplikasi yang diuji. *Unit testing* memungkinkan kita untuk melakukan pengujian pada perangkat lunak dengan cara terpisah. Pengujian ini dapat berupa pengujian pada bagian *unit* paling kecil seperti *function*  maupun beberapa potongan *source code.* Langkah yang digunakan untuk melakukan *unit testing* adalah dengan membuat kode *Unit testing* dan *Test case* secara manual ketika akan melakukan pengujian.

Pengujian unit secara umum menggunakan metode pengujian kotak putih atau *white box*. Pengujian *white box* merupakan pendekatan yang mempertimbangkan mekanisme internal suatu sistem atau komponen (IEEE, 1990). Disebut juga sebagai pengujian struktural, pengujian kotak kosong, atau pengujian kotak kaca, metode white box memungkinkan pemeriksaan rinci terhadap cara kerja internal dan struktur perangkat lunak. Penerapan pengujian white box mencakup berbagai tingkatan, seperti integrasi, unit, dan sistem, yang merupakan bagian integral dari proses pengujian perangkat lunak. Dalam konteks ini, pengujian white box melibatkan inspeksi langsung terhadap kode untuk mengidentifikasi unit kode yang mungkin menunjukkan perilaku yang tidak sesuai (Khan, 2012).

Tujuan dari pengujian white-box, seperti yang dijelaskan oleh Pressman, mencakup beberapa aspek, di antaranya:

* 1. Memverifikasi bahwa semua jalur independen telah dijalankan setidaknya satu kali.
  2. Mengeksekusi semua keputusan logis untuk sisi benar dan salah.
  3. Menjalankan semua loop pada batas operasionalnya.
  4. Memeriksa struktur data internal untuk memastikan keabsahannya.

Salah satu teknik pengujian yang digunakan dalam white-box testing adalah statement coverage. Statement coverage adalah bentuk pengujian yang memastikan bahwa setiap pernyataan (statement) dalam kode dieksekusi minimal satu kali dalam satu test case, sehingga tidak ada test case yang menjalankan pernyataan yang sama (Kusumaningtiyas, Fitria, dan Diputra, 2016). Statement coverage digunakan untuk menguji setiap pernyataan (baris kode) dengan menjalankan pengujian berdasarkan data uji yang telah ditetapkan (Yunisa, 2018). Ada sejumlah beragam kerangka kerja atau *tools* yang tersedia untuk melakukan pengujian unit dalam konteks pengembangan aplikasi berbasis bahasa pemrograman PHP. Beberapa di antaranya termasuk behat, selenium, codeception, atoum, PestPHP, *PHPUnit*, serta berbagai kerangka kerja lain. Pada proses pengujian, penulis telah memilih *PHPUnit* sebagai kerangka kerja yang akan dimanfaatkan. Keputusan ini didasarkan pada dua alasan utama. Pertama, *PHPUnit* merupakan kerangka kerja yang sangat sesuai bagi para pengembang PHP karena dirancang dengan orientasi pemrogram. Kedua, *PHPUnit* mempromosikan ide mengenai pengembang aplikasi perlu mendeteksi error atau kesalahan dalam proyek yang baru secara cepat, tanpa menyebabkan regresi kode di bagian lain aplikasi.

1. ***PHPUnit***

*PHPUnit* merupakan *framework* yang diciptakan oleh Sebastian Bergmann dan termasuk bagian dari xUnit. *PHPUnit* dimanfaatkan untuk melakukan kegiatan uji pada perangkat lunak yang ditulis dalam bahasa pemrograman bahasa PHP. Filosofi utama di balik *PHPUnit* adalah memastikan bahwa pengembang aplikasi dapat dengan cepat mendeteksi kesalahan pada kode yang baru dibuat, dengan menjaga agar tidak terjadi regresi kode dalam kode yang sudah ada. *PHPUnit* juga memiliki kemampuan untuk menghasilkan berbagai jenis output, seperti Junit XML dan TestDox. Dalam hal fungsionalitas, efisiensi, keandalan, dan portabilitas, *PHPUnit* dianggap sebagai alat pengujian unit yang unggul dibandingkan dengan alat lain seperti Codeception dan SimpleTest (Sandin, Yassin, dan Mohamad, 2016). Berikut langkah-langkah pengujian menggunakan *PHPUnit*:

1. Membuat kelas atau fungsi yang akan dibuat pengujian.
2. Mendesign rangka tes yang akan difungsikan sebagai pengecek semua *function* pada kelas yang sebelumnya telah dibuat.
3. Menerapkan kelas atau fungsi dan menghasilkan objek yang telah diterapkan.
4. Mengevaluasi apakah rangkaian tes berhasil atau tidak.
5. *Fixing* kesalahan atau adanya error dan mengulang pada *step* nomor 4 jika terdapat error saat proses pengujian.
6. ***Test case***

*Test case* adalah elemen dokumen pengujian yang digunakan oleh penguji sebagai petunjuk dalam menjalankan pengujian. Namun, perlu diingat bahwa penggunaan *test case* secara manual tidak dapat memberikan kepastian bahwa semua kebutuhan dari sistem sudah tercakup dalam *test case* yang telah disusun (Novelia, 2008). Sebagai ilustrasi, contoh pembuatan dari *test case* yang umum digunakan dalam pengujian dapat ditemukan dalam Tabel 2.1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No | *Test case ID* | *Test case Description* | *Pre-condition* | *Test Steps* | *Expected Result* | *Actual Result* |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**Tabel 2. 1 Tabel *Example* Model *Test case***

Sumber : Authorities dalam Annisa & Teduh (2018)

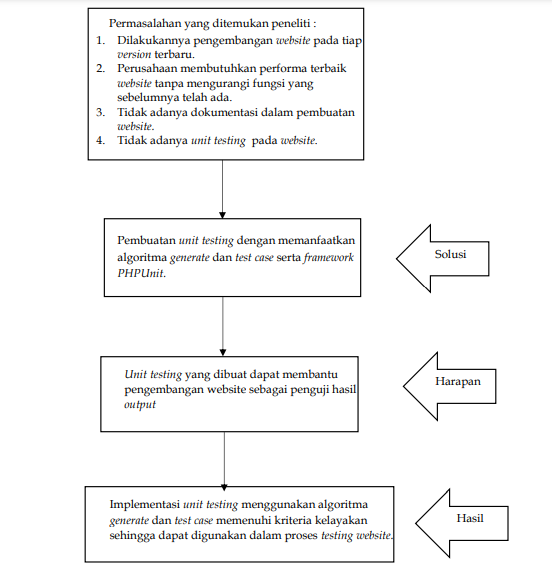
## Penelitian yang Relevan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No** | **Nama** | **Judul Penelitian** | **Tujuan Penelitian** | **Persamaan** | **Perbedaan** |
| 1. | Laksito, 2019 | API Gateway Menggunakan SlimPHP pada Aplikasi Kantin Amikom | Penggunaan Framework *PHPUnit* pada pengujian white box untuk melakukan uji kesesuaian algoritma dalam aplikasi API Gateway. | Menggunakan *PHPUnit* | Berfokus pada pengujian API Gateway |
| 2. | Wijonarko & Mulya 2018 | Pengembangan Antarmuka Pemrograman Aplikasi Menggunakan Metode RESTful pada Sistem Informasi Akademik Politeknik Kota  Malang | Berfokus pada *White Box* dan *Black Box Testing* | Menggunakan *PHPUnit* | Berfokus pada Metode RESTful |

**Tabel 2. 2 Penelitian yang Relevan**

## Kerangka Konseptual

Berdasarkan hasil observasi peneliti, program *unit testing* bertujuan untuk memberikan fasilitas pengujian kinerja dan fungsi dari tiap komponen *website* sehingga *website* dapat memberikan kinerja terbaik dan paling optimal untuk kegiatan *client. Unit testing* juga dapat menjadi pembantu bagi *developer* dalam pengembangan *website* pada versi terbaru namun tidak mengurangi performa dari hasil *website*. Pada suatu *testing* dibutuhkan media dan bahan pengujian yang sesuai. Salah satu dari media tersebut dapat berupa *framework, tools,* hingga *software* yang mumpuni agar *testing* dapat dijalankan dengan baik. Tahap pra pelaksanaan peneliti pertama kali mengenali mengenai permasalahan yang didapatkan, kemudian peneliti mengobservasi rincian *version* pendukung *website* saat dilakukan *building.* Kemudian dilanjutkan dengan pencarian referensi mengenai implementasi yang sama dari berbagai penelitian sebelumnya. Setelah berbagai referensi terkumpul langkah selanjutnya adalah mendesign *unit testing,* kemudian dilanjutkan dengan melakukan revisi *unit testing* awal yang telah divalidasi kemudian *unit testing* diimplementasikan pada *website* untuk dilakukan pengujian fungsi *website* setelah dilakukan pengembangan. Berikut merupakan bagan kerangka berpikir dalam penelitian ini untuk mempermudah memahami penjelasan di atas yang ditunjukkan pada Gambar 2.1.



**Gambar 2. 1 Kerangka Konseptual**

**Noted :**

**Daftar Pustaka**

**Italic**

**Cek spasi awal paragraf**

# BAB III

# METODE PENELITIAN

## Jenis dan Rancangan Penelitian

### Jenis Penelitian

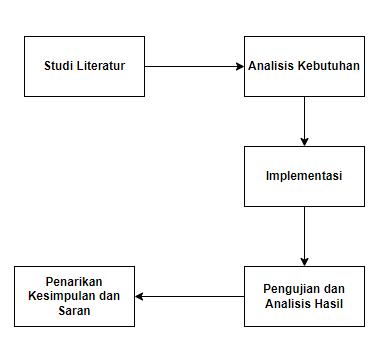
Tujuan utama penelitian ini adalah untuk menerapkan suatu metode pengujian yang efektif dan efisien menggunakan PHPUnit, yang secara khusus dirancang untuk memastikan kualitas dan kehandalan perangkat lunak yang dikembangkan. Dalam konteks penelitian ini, fokus utama adalah untuk menerapkan solusi dalam bentuk unit testing dengan memanfaatkan framework PHPUnit dan algoritma *generate* dan *test case*. Pada penelitian ini terbatas pada menghasilkan produk dan pengujian hasil pengembangan sehingga tidak dilakukan penilaian pada nilai keefektifan produk yang dihasilkan. Dalam penelitian ini akan menghasilkan *unit testing* yang memanfaatkan PHPUnit dan algorima *generate dan test case.*

### Rancangan Penelitian

Rancangan penelitian ini menggunakan algoritma *generate and test case* yang nanti difokuskan dengan memanfaatkan PHPUnit. Website Columbia Catanta pertama kali dikembangkan pada tahun 2014 menggunakan Bahasa pemrograman PHP versi 5.4, website difokuskan pada fungsi perpanjangan kontrak kerja sama dari tiap perusahaan yang bekerja sama dengan Perusahaan Columbia Catanta. Setiap tahun tiap Bahasa pemrograman memiliki perkembangan mengenai versinya, PHP menjadi

salah satu bahasa pemrograman yang mengalami upgrade version hingga versi ke 7 pada 2023. Terdapat beberapa *page* yang sudah tidak dapat digunakan sehingga website Columbia Catanta perlu dilakukan testing agar mengetahui fungsi yang harus diperbarui dan mana yang dapat dipertahankan.

Algoritma *Generate and Test* adalah salah satu algoritma sederhana dalam teknik pencarian heuristik. Dalam pendekatan *Generate and Test*, terdapat dua langkah kunci, yakni generate (membangkitkan), yang melibatkan pembangkitan semua solusi yang mungkin, dan test (pengetesan), yang melibatkan pengujian solusi yang telah dibangkitkan. Pada testingnya, *test case* dipilih sebagai method dalam pengujian. Manfaat menggunakan metode *test case* mencakup kemampuannya untuk menemukan kesalahan dalam kode dengan mengidentifikasi dari *case* yang telah ditentukan. Selain itu, metode ini dapat memberikan cakupan pengujian aplikasi yang maksimal saat menguji suatu skenario uji coba (Harya & Sri, 2022). Model penelitian ini dipilih karena lebih sistematis dan lebih lengkap hal tersebut dikarenakan setiap tahapannya dilakukan pembangkitan setiap solusi yang ada. Penelitian ini memiliki beberapa tahapan yang akan dijadikan acuan dalam setiap proses yang akan dilakukan. Metodologi atau rancangan penelitian dapat dilihat berdasarkan gambar dibawah ini:

**

**Gambar 3. 1 Gambar Rancangan Penelitian**

Berikut penjelasan dari masing-masing tahapan:

1. Studi Literatur

Peneliti mengambil informasi dari jurnal maupun penelitian yang telah dilakukan dari 2018-2023, sumber yang akan digunakan adalah *Google Scholars, Scopus*, dan beberapa sumber jurnal lainnya. Poin utama yang akan di *highlight* adalah literatur mengenai tahapan pengujian dan literatur mengenai *test case* untuk menjadi pedoman pengujian *website.*

1. Analisis Kebutuhan

Pada tahap ini peneliti akan melakukan penentuan persyaratan dan kebutuhan yang dibutuhkan atau perlu ada dalam pengujian *unit testing.* Langkah-langkah analisis kebutuhan pada penelitian ini sebagai berikut:

1. Identifikasi *Unit* untuk Pengujian:

Pada tahap ini peneliti akan menentukan unit(fungsi, metode, atau kelas) yang akan diuji menggunakan *test case*. Berikut merupakan *unit* yang telah ditentukan untuk dilakukan pengujian:

**Tabel 3.1 Tabel Identifikasi *Unit* Pengujian**

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Menu | Point Test | Jumlah Bug Sebelum *Testing* |
|  | TOP | 1. Fungsi tautan ke halaman utama (TOP). |  |
|  | Constant Settings | 1. Add setting. 2. Pengeditan setting yang sudah disimpan. |  |
|  | Site Settings | 1. HTML 2. PARAM |  |
|  | DB Settings | 1. Show List Table 2. Param List (Detail, Edit Delete). 3. XML List (Display XML code). |  |
|  | Email Settings | 1. Display table 2. Add Data 3. Go to Confirmation data |  |
|  | Development Tools | 1. Generation password |  |
|  | User Management | 1. Display table user (Edit, Access settings, Corporate settings 2. Add user |  |
|  | Contract Display | 1. Display table list contract (Detail, Permitted). 2. Search Contract |  |
|  | Contract Management | 1. Display table list contract. 2. Add contract 3. Export CSV file |  |
|  | Import Contract Information | 1. Import file |  |
|  | PDF Upload | 1. PDF upload |  |
|  | Corporate Settings | 1. Display table list corporate 2. Add Corporate 3. Search Corporate |  |
|  | Contract Update Process | 1. Contract Update Process |  |

1. Rincian Persyaratan Fungsional:

Pada tahap ini peneliti akan merincikan mengenai fungsionalitas yang diharapkan dari setiap unit dan rincian bagaimana fungsionalitas tersebut akan diuji. Pada rincian persyaratan fungsional akan tersaji data mengenai identitas case yang memiliki syarat disebut dengan CASE-Id dan Information yang berisi informasi dari syarat *case* tersebut. Berikut merupakan rincian mengenai syarat fungsional yang akan diuji:

**Tabel 3.2 Tabel Persyaratan Fungsional**

|  |  |
| --- | --- |
| **Case- Id** | **Information** |
| CASE- F000 | Login |
| CASE- F001 | Add contract |
| CASE- F002 | Edit contract |
| CASE- F003 | Delete contract |
| CASE- F004 | Check detail data contract |
| CASE- F005 | Export data CSV |
| CASE- F006 | Import data CSV |
| CASE- F007 | Generate link |
| CASE- F008 | Add User |
| CASE- F009 | Edit user data |
| CASE- F010 | Search data |
| CASE- F011 | Upload PDF file |
| CASE- F012 | Check detail data |
| CASE- F013 | Delete PDF file |
| CASE- F014 | Add email |
| CASE- F015 | Edit email |
| CASE- F014 | Set constant setting |
| CASE- F015 | Generate password |
| CASE- F016 | Add company |
| CASE- F017 | Edit company |
| CASE- F018 | Perform automatic renewal contract processing |

1. Spesifikasikan Batasan dan Kriteria Pengujian:

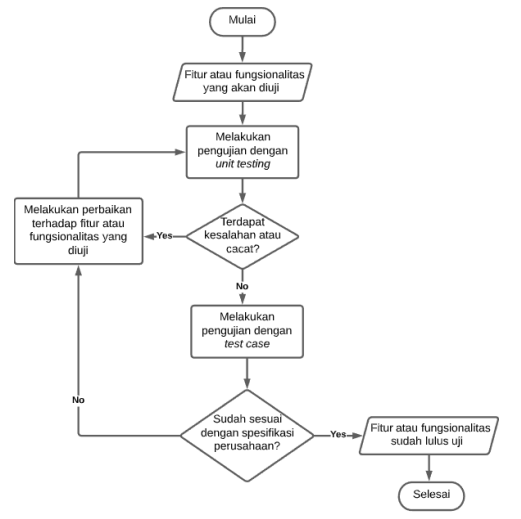
Pada tahap ini peneliti akan menetapkan batasan pengujian dan keterangan mengenai kelulusan pengujian *test case*. Batasan pengujian *test case*  ini hanya menggunakan PHPUnit sebagai *framework* pengujian dan penulisan keterangan kelulusan akan dituliskan dalam bentuk point dari jumlah *case* yang telah lolos pengujian.

1. Implementasi

Pada tahap ini, peneliti mengimplementasikan rancangan unit testing kedalam bahasa pemrograman. Penyusunan kode program dilakukan dengan menggunakan PHPUnit.

1. Pengujian dan Analisis Hasil

Pada tahap ini peneliti akan melakukan pengujian *unit testing* yang telah ditentukan sesuai dengan *case* yang telah dibuat. Tahapan pada proses pengujian ini sebagai berikut:

******

**Gambar 3. 2 Gambar Rancangan Pengujian dan Analisis (**Annisa & Teduh , 2018)

Pengujian akan dilakukan sesuai tahapan yang telah ditentukan, pada masa pengujian terdapat acuan *checklist list* yang akan digunakan dalam pengujian. Hasil dari pengujian akan disajikan sebagai berikut:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No | *Test case ID* | *Test case Description* | *Pre-condition* | *Test Steps* | *Expected Result* | *Actual Result* |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**Tabel 3.3 Tabel Pengujian *Test Case***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | *Test case ID* | Jumlah *bug* sebelum *testing* | Waktu perbaikan *bug* sebelum *testing* | Jumlah *bug* setelah *testing* | Waktu perbaikan *bug* setelah *testing* |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Tabel 3.4 Tabel Kondisi**

1. Penarikan Kesimpulan dan Saran

Pada tahap ini peneliti akan menyajikan seluruh hasil pengujian dalam bentuk tabel yang memuat informasi mengenai presentase hasil kelulusan *test case* pada setiap *page* yang telah diuji. Penghitungan presentase berdasarkan rumus: **(*Test case* yang lolos/Jumlah *Test case)* X 100%.** Data yang dihasilkan sebagai berikut:

**Tabel 3.5 Tabel Hasil Akhir Pengujian**

|  |  |  |  |
| --- | --- | --- | --- |
| No. | *Test case ID* | Jumlah Test Case | Presentase Lulus Uji |
|  | TOP | 2 |  |
|  | Constant Settings | 2 |  |
|  | Site Settings | 2 |  |
|  | DB Settings | 3 |  |
|  | Email Settings | 3 |  |
|  | Development Tools | 1 |  |
|  | User Management | 2 |  |
|  | Contract Display | 2 |  |
|  | Contract Management | 3 |  |
|  | Import Contract Information | 1 |  |
|  | PDF Upload | 1 |  |
|  | Corporate Settings | 3 |  |
|  | Contract Update Process | 1 |  |

### Lokasi Penelitian

Dalam penelitian ini, peneliti melakukan pengujian secara *online*.

### Populasi Penelitian

Fokus penelitian ini tertuju pada populasi yang melibatkan setiap halaman atau page yang tersedia di Website Manajemen Kontrak dari Perusahaan Columbia Catanta. Populasi tersebut mencakup seluruh komponen yang membentuk struktur website, termasuk fitur-fitur, fungsi-fungsi, dan elemen-elemen lainnya yang terlibat dalam tata kelola dan manajemen kontrak.

### Sasaran Penelitian

Sasaran dari implementasi unit testing merupakan website manajemen kontrak perusahaan Columbia Catanta. Berdasarkan pengamatan pada kinerja website selama kurun waktu dua bulan (Agustus, September. Website Columbia Catanta dikembangkan dengan adanya beberapa fitur baru namun belum memilki unit testing yang dapat digunakan untuk menguji output dari pengembangan fitur terbaru.

### Definisi Operasional

1. ATRIBUT TEST CASE

**Tabel 3. 6 Tabel Atribut *Test Case***

|  |  |
| --- | --- |
| Atribut | Keterangan Atribut |
| Fitur | Nama fitur atau fungsionalitas yang akan diuji |
| Test case id | Identitas dari fitur atau fungsionalitas yang akan diuji |
| Test case  description | Deskripsi fitur atau fungsionalitas yang akan diuji |
| Pre-condition | Kondisi sistem sebelum melakukan pengujian |
| Test steps | Langkah-langkah saat melakukan pengujian |
| Expected  result | Hasil yang diinginkan dari fitur atau fungsionalitas  yang diuji |
| Actual result | Hasil yang sebenarnya terjadi pada fitur atau  fungsionalitas yang telah diuji |
| Status | Status sudah melakukan test (pass or fail) |

Sumber: Authorities dalam Annisa & Teduh (2018)

### Instrumen Penelitian

Intrumen ini digunakan untuk mengetahui apakah *unit testing* sudah layak dan mengukur kesesuaian terhadap *unit testing* dengan *output* yang diharapkan oleh pengembang. Intrumen penelitian yang digunakan dalam implementasi *unit testing* ini berbentuk *test case* terkait hasil *output* yang akan diisi oleh penguji. Dalam pembuatan instrument penelitian ini, terdapat kegunaan kisi-kisi pada setiap masing-masing instrument penelitian, yaitu sebagai berikut:

1. *Checklist* Pengujian Unit:

**Tabel 3.7 Tabel *Checklist* Pengujian Unit**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Menu | Point Test | Jumlah Test Case | Hasil pengujian |
|  | TOP | * Fungsi tautan ke halaman utama (TOP). * Elemen-elemen antarmuka pengguna (UI) di halaman TOP ditampilkan dengan benar? | 2 |  |
|  | Constant Settings | * Add setting. * Pengeditan setting yang sudah disimpan. | 2 |  |
|  | Site Settings | * HTML * PARAM | 2 |  |
|  | DB Settings | * Show List Table * Param List (Detail, Edit Delete). * XML List (Display XML code). | 3 |  |
|  | Email Settings | * Display table * Add Data * Go to Confirmation data | 3 |  |
|  | Development Tools | * Generation password | 1 |  |
|  | User Management | * Display table user (Edit, Access settings, Corporate settings * Add user | 2 |  |
|  | Contract Display | * Display table list contract (Detail, Permitted). * Search Contract | 2 |  |
|  | Contract Management | * Display table list contract. * Add contract * Export CSV file | 3 |  |
|  | Import Contract Information | * Import file | 1 |  |
|  | PDF Upload | * PDF upload | 1 |  |
|  | Corporate Settings | * Display table list corporate * Add Corporate * Search Corporate | 3 |  |
|  | Contract Update Process | * Contract Update Process | 1 |  |

### Teknik Pengumpulan Data

Penelitian pengembangan ini menggunakan 2 (dua) teknik pengumpulan data, diantaranya sebagai berikut:

1. Observasi

Observasi dilakukan saat pengembangan dan sebelum dilakukan *testing*. Hal ini untuk mengetahui kebutuhan *website* dalam pengujian, seperti kebutuhan *software* dan media atau *tools* lain yang digunakan. Observasi ini meliputi observasi pada sistem (*webste),* pengiriman data pada database, dan penerimaan data pada database. Selain itu, terdapat 3 user yang akan diobservasi, yaitu : Master, Admin, User. Master merupakan tingkat tertinggi yang memiliki serta dapat mengatur semua akses. Admin merupakan tingkatan kedua setelah Master. User merupakan tingkatan terendah dari pengguna *website* ini dimana akses yang dimiliki paling terbatas.

1. Tes

Tes merupakan salah satu alat untuk melakukan pengukuran, yaitu alat untuk mengumpulkan informasi karakteristik suatu objek. Pada penelitian ini, tes dilakukan untuk mengetahui tingkat keberhasilan pengujian. Tes yang digunakan berupa pengisian table keterangan pada tiap *test case.*

### Teknik Analisis Data

Teknik analisis data yang dipilih untuk penelitian ini adalah kunci utama dalam membongkar potensi keberhasilan implementasi unit testing. Melalui penekanan pada jumlah bug sebelum dan setelah penerapan unit testing, waktu yang diperlukan untuk perbaikan bug, dan coverage unit testing. . Hasil yang diharapkan dari penelitian ini adalah untuk mengetahui keberhasilan *unit testing* berdasarkan analisis data hasil pengujian.

**Tabel 3.8 Tabel *Checklist* Pengujian Unit**

|  |  |  |
| --- | --- | --- |
| No. | Kriteria Evaluasi | Data Pengukuran/Hasil |
| 1. | Jumlah Bug Sebelum Unit Testing | Jumlah bug sebelum implementasi unit testing |
| 2. | Jumlah Bug Setelah Unit Testing | Jumlah bug setelah implementasi unit testing |
| 3. | Waktu Perbaikan Bug Sebelum Unit Testing | Rata-rata waktu perbaikan bug sebelum implementasi unit testing |
| 4. | Waktu Perbaikan Bug Setelah Unit Testing | Rata-rata waktu perbaikan bug setelah implementasi unit testing |
| 5. | Coverage Unit Testing | Persentase coverage unit testing terhadap total kode program |