**REVISI VERSION 0.4**

Sampul Depan

Lembar Judul

Lembar Pengesahan

Lembar Persetujuan Dewan Penguji

Lembar Pernyataan Keaslian

Abstrak

Abstract

Kata Pengantar

Daftar Isi

Daftar Tabel

Daftar Gambar

Daftar Lampiran

Daftar Singkatan dan Lambang

# PENDAHULUAN

## Latar Belakang

*Programmers* adalah orang yang bertugas untuk mengembangkan perangkat lunak atau aplikasi. Ketika membuat suatu program, ada bagan alur yang harus diperhatikan kemudian diterjemahkan ke dalam bahasa komputer menggunakan kode tertentu sehingga menjadi sebuah perintah kepada komputer untuk menjalankan tugas tertentu [1].

Kode Sumber (*Source Code*) adalah komponen dasar dari program komputer yang dibuat oleh seorang *programmers*, yang sering kali ditulis dalam bentuk fungsi, deskripsi, definisi, pemanggilan, metode, dan pernyataan operasional lainnya. Kode ini dirancang agar dapat dibaca manusia dan diformat dengan cara yang dapat dimengerti oleh pengembang dan pengguna lain[2].

Penguji (*source code review*) adalah peran yang melakukan pemeriksaan kode sumber aplikasi untuk menemukan kesalahan yang terlewatkan pada tahap pengembangan awal. Penguji meluncurkan penganalisis kode yang memindai kode aplikasi baris demi baris. Setelah penganalisis, yang diterapkan di lingkungan pengujian, menemukan kerentanan, pentester memeriksanya secara manual untuk menghilangkan positif false. *(penguji = orang/peran, pengujian = aktifitas).*

Jumlah waktu yang dihabiskan penguji untuk meninjau kode sumber bervariasi menurut bahasa pemrograman dan ukuran aplikasi. Misalnya, 1000 baris kode mungkin memerlukan waktu 0,5 – 2 jam untuk dianalisis[3].

Positif palsu adalah saat penguji menerima hasil positif untuk suatu tes, padahal programmer seharusnya menerima hasil negatif. Kadang-kadang disebut sebagai “ alarm palsu” atau “kesalahan positif palsu”[4].

Dalam mempelajari sebuah pemrograman *programmers* tentunya harus sering belajar yang harus melibatkan penguji untuk menilai kode yang dibuatnya. Rata-rata penguji sangat kerepotan karena harus menguji beberapa *project* dari *programmers* sehingga membutuhkan waktu yang cukup lama dan sangat rentan dengan human *error*. Untuk penguji sebagian masih ada yang menilai *source code* secara manual satu persatu, melihat *output* yang sesuai, bahkan jika dijalankan tidak ada *error* maka dinilai berhasil. Kekurangan dalam proses penilaian ini adalah waktu yang diperlukan yang cukup lama karena setiap tugas dari programmer harus diuji satu per satu. Selain itu, terdapat risiko bahwa penguji mungkin kurang fokus saat melakukan pengujian, yang dapat mengakibatkan keterlewatannya kesalahan atau masalah yang penting dalam kode yang diuji.

Sehingga, penguji membutuhkan sistem yang dapat memeriksa dan memperbaiki sintaks kode sesuai dengan standar umum yang digunakan oleh programmer. Selain itu, mereka juga membutuhkan fitur penilaian otomatis untuk mengevaluasi kualitas kode secara keseluruhan, yang dapat mempercepat proses penilaian.

Cara kerja sistem yang diusulkan diharapkan penguji membuat terlebih dahulu *syntax* dan TDD (*Test Driven Development*) yang nantinya digunakan untuk referensi jawaban yang tepat, selanjutnya *syntax* akan diproses terlebih dahulu yang akan dicek oleh *library* yang mendukung *syntax* seperti Eslint, Prettier. Jika tidak sesuai prosedur maka akan masuk ke *library* yang dibuat pada penelitian ini agar dimasukan ke *system* yang nanti akan digunakan untuk referensi jawaban dari *programmers*. Selanjutnya dari sisi *programmers* melakukan input *syntax* yang akan diproses oleh *library* pada penelitian ini yang nantinya akan dianalisis secara *syntax* menggunakan beberapa *library* pendukung dari NPM (*Node Package Manager*) serta bantuan metode BlackBox dan sesuai prosedur *test* yang diberikan penguji menggunakan metode TDD (*Test Driven Development*).

## Rumusan Masalah

Adapun rumusan masalah dari penelitian ini, antara lain :

1. Bagaimana perancangan dan pembuatan *library* penilaian *source code* otomatis?
2. Bagaimana cara praktis menerapkan library penilaian otomatis untuk menilai kode JavaScript?

## Batasan Masalah

Pada penelitian ini adapun batasan masalahnya, antara lain :

1. Pengujian ini akan dilakukan hanya dengan bantuan *library* yang menggunakan bahasa pemrograman Javascript.
2. Pengujian berfokus pada mengecek kode sumber bahasa pemrograman Javascript.
3. Pembuatan *library* akan didukung dengan metode *BlackBox* dan *Test Driven Development*.

## Tujuan Penelitian

Tujuan dari penelitian ini adalah :

1. Meningkatkan efisiensi dan konsistensi penilaian kode sumber.
2. Memudahkan penguji dan *programmers* dalam evaluasi kualitas kode sumber.
3. Meningkatkan transparansi dan objektivitas dalam penilaian kode sumber.

## Metode Penelitian

### Metode Pengumpulan Data

Sumber Data

1. Data Primer: Merupakan informasi yang diperoleh dari sumber primer, seperti dari orang atau individu melalui wawancara, survei, dan kuesioner wawancara, yang dilakukan oleh peneliti[5].
2. Data Sekunder: Merupakan informasi yang sudah diolah sebelumnya dan diperoleh oleh peneliti dari berbagai sumber lain untuk memperoleh tambahan informasi. Beberapa sumber data sekunder juga bisa berupa literatur, jurnal, serta publikasi pemerintah dan sumber daya online lainnya[5].

Observasi

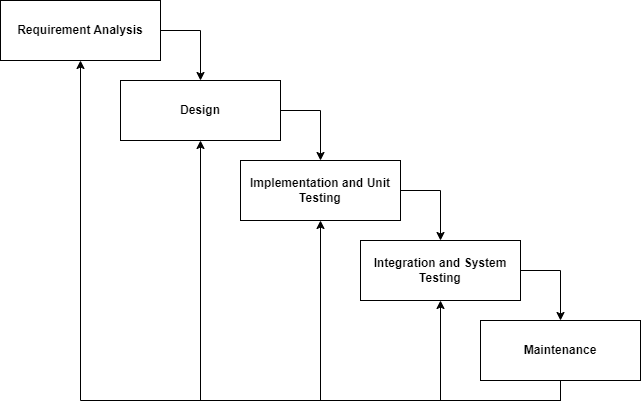
Peneliti melakukan pengamatan secara langsung yang berkaitan dengan masalah yang diteliti untuk dianalisa.

Studi Pustaka

Studi pustaka dikenal menjadi tinjauan pustaka, merupakan proses pengumpulan, analisis, dan buatan literatur yang relevan menggunakan topik atau kasus penelitian tertentu. Ini melibatkan membaca aneka macam asal fakta misalnya jurnal ilmiah, buku, makalah konferensi, artikel daring, dan asal-asal lainnya yang terkait menggunakan subjek yang sedang diteliti[5].

### Metode Pengembangan Sistem

Metode pengembangan sistem yang peneliti gunakan dalam penulisan skripsi ini adalah Metode *Waterfall*. Model *waterfall* adalah model pengembangan perangkat lunak yang terdiri dari analisa kebutuhan perangkat lunak, desain, pembuatan kode program, pengujian, pendukung dan pemeliharaan sistem. Model *waterfall* dikerjakan secara berurutan sesuai dengan tahapan awal sampai akhir, sehingga proyek yang dikerjakan memiliki perencanaan yang lebih cermat, dokumentasi lebih rinci dan eksekusi yang berurutan[6].



Gambar 1. 1 Metode Waterfall

#### Requirement Analysis

Tahap ini merupakan awal dari pengembangan sistem di mana peneliti akan mengumpulkan dan menganalisis kebutuhan sistem yang akan dikembangkan. hal ini akan melibatkan kebutuhan dalam fungsional dan non-fungsional, serta memahami kebutuhan pengguna dan tujuan sistem. Proses ini membantu memastikan bahwa pemahaman yang jelas tentang apa yang harus dicapai oleh sistem.

#### Design

Setelah kebutuhan sistem dipahami dengan baik, langkah berikutnya adalah merancang arsitektur sistem secara keseluruhan dan detail implementasi. Ini mencakup merancang struktur data, algoritma, antarmuka pengguna, dan komponen sistem lainnya. Desain ini harus memastikan bahwa sistem yang akan dibangun akan mencapai tujuan yang telah ditetapkan dan memenuhi kebutuhan pengguna.

#### Implementation and Unit Testing

Pada tahap ini, pengembang mulai mengimplementasikan desain sistem ke dalam kode program. Ini melibatkan menulis kode, membangun modul atau komponen, dan mengintegrasikannya secara bertahap. Selama proses ini, unit tes juga dilakukan untuk memastikan setiap komponen atau modul berfungsi sesuai yang diharapkan.

#### Integration and System Testing

Setelah semua komponen atau modul sistem telah diimplementasikan dan diuji secara individual, tahap selanjutnya adalah mengintegrasikan mereka menjadi satu sistem utuh. Proses ini melibatkan menguji interaksi antara komponen-komponen tersebut dan memastikan bahwa sistem berfungsi dengan baik sebagai satu kesatuan. Pengujian sistem dilakukan untuk memverifikasi apakah sistem memenuhi spesifikasi yang telah ditetapkan dan berperilaku sesuai dengan harapan.

#### Maintenance

Tahap ini terjadi setelah sistem telah dirilis dan digunakan oleh pengguna. Pada tahap ini, perawatan sistem dilakukan untuk memastikan bahwa sistem tetap berjalan dengan baik dan memenuhi kebutuhan pengguna. Ini melibatkan pemecahan bug, peningkatan fitur, dan perbaikan yang mungkin diperlukan seiring waktu.

## Sistematika Penulisan

Sistematika penyusunan laporan tugas akhir ini adalah sebagai berikut:

**BAB I Pendahuluan**

Dalam bab ini akan diuraikan mengenai latar belakang, rumusan masalah, tujuan penelitian, batasan masalah, metodologi penelitian, manfaat penelitian dan sistematika penulisan.

**BAB II Landasan Teori**

Dalam bab ini menjelaskan dari beberapa teori yang digunakan untuk melakukan penelitian dan penulisan laporan.

**BAB III Analisis dan Perancangan**

Dalam bab ini menjelaskan analisa rancangan dan kebutuhan pengujian yang sudah dibuat untuk melakukan pengujian *library* yang dapat menilai *source code* otomatis.

**BAB IV Implementasi dan Pengujian**

Dalam bab ini menjelaskan hasil dari pengujian yang telah dilakukan dan pembahasan hasil yang telah didapatkan dari penelitian.

**BAB V Kesimpulan dan Saran**

Bab ini berisi kesimpulan yang diambil dari penelitan yang telah dilakukan serta saran yang berkaitan dengan pengujian *source code* secara otomatis untuk pengujian soal *programmers*.

# LANDASAN TEORI

# ANALISIS DAN PERANCANGAN

# IMPLEMENTASI DAN PENGUJIAN

# KESIMPULAN DAN SARAN

**Daftar Pustaka**

[1] Quipper, “Programmer (Software Developer),” Quipper Campus. Accessed: Mar. 08, 2024. [Online]. Available: https://campus.quipper.com/careers/programmer

[2] Scott Wallask, “Source Code,” www.techtarget.com. Accessed: Dec. 13, 2023. [Online]. Available: https://www.techtarget.com/searchapparchitecture/definition/source-code

[3] Uladzislau Murashka, “Source Code Review vs. Penetration Testing for Web Application Security,” ScienceSoft. Accessed: Apr. 01, 2024. [Online]. Available: https://www.scnsoft.com/blog/web-applications-security-source-code-review-vs-penetration-testing

[4] Statistics How To, “False Positive dan False Negative: Pengertian dan Contohnya,” www.statisticshowto.com. Accessed: Apr. 01, 2024. [Online]. Available: https://www.statisticshowto.com/false-positive-definition-and-examples/

[5] ADITIA GUSTI ANANDA, “PEMANFAATAN TOOL KATALON STUDIO DALAM PENGUJIAN WEB LEARNING MANAGEMENT SYSTEM (Studi Kasus : Galeri Kejuruan),” 2023.

[6] E. Rahmi, E. Yumami, and N. Hidayasari, “Analisis Metode Pengembangan Sistem Informasi Berbasis Website: Systematic Literature Review,” *ResearchGate*, vol. 7, pp. 821–834, Jan. 2023, doi: 10.33395/remik.v7i1.12177.

Riwayat Hidup

Lampiran