BAB 1
PENDAHULUAN

1.1 Latar Belakang Masalah

Sudah banyak sekali aplikasi-aplikasi yang sudah diciptakan, maupun mendapat perkembangan, namun tidak sedikit juga aplikasi-aplikasi tersebut dirancang dengan *baik* dan *teliti*. Banyak sekali software yang rentan gagal dalam menghadapi perubahan dikarenakan struktur sistem yang kurang baik[1]. Perubahan merupakan hal yang dinamis dan akan selalu terjadi mengikuti lingkungan teknologi itu sendiri. Oleh karena itu, sangat penting dalam menjaga "Kesehatan" dari sebuah software guna menjaga kualitas dan keutuhan dari software tersebut. "Kesehatan" software terbagi ke dalam beberapa jenis faktor kualitas, yang diantaranya adalah Reliability, Testability, Integrity, Maintainability dan lain-lain[2]. Maintainability adalah salah satu faktor dari metrics software Product Revision[2], dimana Maintainability merupakan kemampuan sebuah software dalam menghadapi perubahan ataupun perkembangan komponen maupun environtment ketika dilakukannya maintenance[3]. Perancangan sebuah software tentu harus dibuat dengan tujuan agar aplikasi tersebut bisa beroperasi dengan baik dan benar, namun software yang baik juga merupakan software yang memiliki jangka waktu yang panjang dalam usia penggunaan, maupun pengembangan lebih lanjut, karena memaintain "Long-Living Software" akan sangat terkait dengan kualitas code dari system yang dibangun[4]. Beberapa software dapat dibangun dengan cepat, namun banyak developer yang hanya mengejar tujuan agar software tersebut selesai dibangun tanpa memperhatikan aspek kualitas code dan aspek lainnya seperti kompleksitas maupun sktruktur code, maka diperlukan kesamaan dalam ide maupun pemahaman dalam membangun sebuah software, karena dengan menemukan kesamaan dapat secara signifikan menentukan tentang kegagalan atau keberhasilan dalam membuat kode yang maintainable.

Website Pikobar merupakan sebuah Web App yang menyediakan informasi dan koordinasi COVID-19 Jawa Barat. Dibuat dengan bahasa pemrograman based

on Javascript dan Framework Vue.Js, Web App tersebut memiliki lebih dari 50 Komponen script, dimana masing-masing script hampir terdapat ratusan lines of code, serta terdapat beberapa dependency API. Banyaknya lines of code serta kurang jelasnya fungsi yang dibuat, dapat menghambat perkembangan Web App tersebut. Terdapat beberapa fungsi yang tidak memiliki penjelasan mengenai tugas dari fungsi tersebut (Tidak terdapat comments) pada Web App Pikobar, beberapa penamaan variable dan fungsi yang kurang tepat, dan apabila hal-hal tersebut tidak diatasi, maka akan menghambat proses *Maintenance* source code pada software dikemudian hari. Salah satu solusi dalam memecahkan masalah pada source kode adalah dengan metode Clean Code.

Clean Code merupakan sebuah prinsip/metode/landasan dalam merangkai sebuah Code yang memiliki logika yang jelas. Code yang berkualitas, merupakan Code yang bisa dipahami oleh seseorang yang tidak membuat Code tersebut (Readable). Code yang berkualitas mampu memberikan ide baru dalam proses development, seseorang akan tertarik untuk memberikan ide tanpa merubah Code yang sudah dibuat (Reuseable). Code yang berkualitas mampu menyediakan output yang diperlukan, serta memiliki fungsi yang pasti tanpa ada hal yang tidak penting (Reliable). Code yang bersih tentu mudah dipahami tanpa adanya ketergantungan (dependency) yang berlebihan, dengan minimun resource yang terdefinisi dengan baik dan jelas kegunaannya (Complexity). Tentu saja minimum resource akan lebih baik diolah dibandingkan resource yang berlebihan, sama halnya dengan Code, akan lebih baik menjaga ukuran Code tersebut optimal karena "Size does matter" (Unit Size). Code yang mudah dibaca belum tentu bisa dirubah, tetapi code yang mudah dipahami akan lebih mudah untuk mendapatkan perubahan (Maintainable) [5]. Maka dengan Clean Code, diharapkan bisa menambah kualitas maintainability Web App Pikobar menjadi lebih baik.

1.2 **Rumusan Masalah**

Berdasarkan Latar belakang yang telah diringkas, maka dapat diidentifikasi beberapa masalah sebagai berikut:

1. Apakah Web App Pikobar dapat lebih mudah untuk dikembangkan lebih lanjut oleh developer yang tidak terkait atau developer baru?
2. Apakah Maintainability Web App Pikobar sudah tergolong baik?
3. Apakah Clean Code mampu meningkatkan Maintainability dari Website Pikobar?

1.3 **Maksud dan Tujuan**

Maksud dilaksanakannya penelitian ini adalah untuk mengetahui apakah Web App Pikobar dapat dipahami oleh developer lain, mengetahui berapa besar persentase kejanggalan Code pada Web App pikobar, serta mengetahui apakah Clean Code bisa meningkatkan maintainability Web App Pikobar. Adapun tujuan dari penelitian ini, yaitu meningkatkan readability Code pada Web App Pikobar, mengurangi persentase kejanggalan Code Web App Pikobar, serta mengimplementasikan Clean Code untuk meningkatkan maintainability Web App Pikobar.

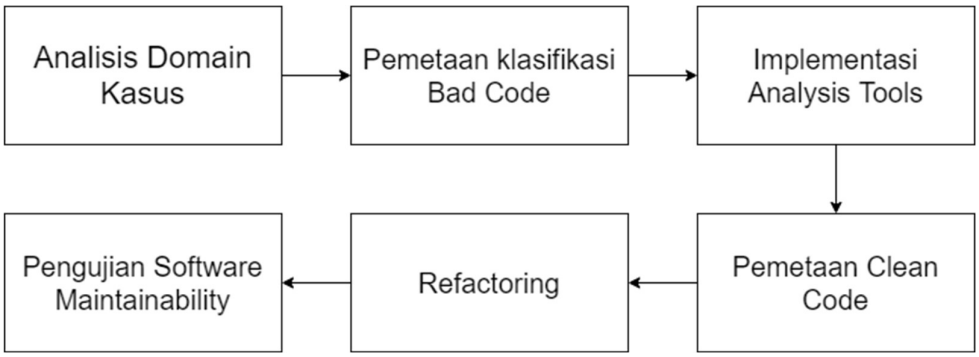
1.4 **Batasan Masalah**

Terdapat beberapa batasan masalah yang akan dipakau guna menjaga konsistensi penelitian ini, batasan-batasan tersebut adalah sebagai berikut :

1. Bahasa pemrograman yang digunakan adalah Javascript dan Framework Vue.JS.
2. Analisis tools yang dipakai, menggunakan based Javascript di Visual Studio Code.
3. Implementasi Clean Code menggunakan Test-Driven Development.
4. Target penilaian hanya Source Code dari Web App Pikobar.

1.5 Metodologi Penelitian

Penelitian ini merupakan jenis penelitian Kuantitatif, dan metodologi penelitian akan disesuaikan dengan kebutuhan penelitian. Berikut adalah gambar tahapan penelitian;



Gambar 1 Alur Metode Penelitian

Berikut ini adalah penjelasan mengenai tahapan penelitian ini :

1. Analisis Domain Kasus

Analisis domain kasus akan dilakukan pertama kali, guna mendapatkan informasi terkait masalah yang diangkat pada penelitian ini, seperti fungsi-fungsi pada source code serta komponen pada code lainnya.

2. Pemetaan Klasifikasi Bad Code

Pada tahapan ini, akan dilakukan pemetaan terhadap source code yang akan menjadi bahan penelitian. Klasifikasi ini akan berguna sebagai bahan perbandingan dengan Good Code yang diangkat pada referensi buku Clean Code Robert C. Martin.

3. Implementasi Analysis Tools

Setelah mendapatkan referensi perbandingan Bad Code dan Good Code, maka akan dilakukan analisa menggunakan tools yang tersedia, seperti JS Hint, dan kemudian akan dilakukan pengukuran kompleksitas menggunakan JS Complexity Analysis dalam bentuk persentase.

4. Pemetaan Clean Code

Pada tahap ini, hasil analisa akan dipakai untuk pemetaan Clean Code dengan menggunakan hasil pemetaan klasifikasi Bad Code.

5. Refactoring

Setelah mendapatkan hasil analisa dan pemetaan Code, langkah selanjutnya adalah refactoring Code dengan Good Code yang sudah didapat menggunakan pemetaan Clean Code.

6. Pengujian Software Maintainability

Tahap terakhir adalah implementasi pengujian terhadap maintainability, dengan membandingkan persentase Bad Code dan Good Code.

1.6 Sistematika Penulisan

Sistematika penulisan pada penelitian ini disusun seperti di bawah ini :

BAB 1 PENDAHULUAN

BAB 1 berisikan Latar Belakang, Rumusan Masalah, Maksud dan Tujuan, Batasan Masalah, serta Metodologi Penelitian dan Sistematika Penelitian.

BAB 2 LANDASAN TEORI

BAB 2 berisi tentang teori-teori yang membantu pelaksanaan penelitian ini, serta dengan konsep yang sudah terangkai oleh para ahli. Mengumpulkan informasi yang akan dijadikan acuan pada penelitian ini.

BAB 3 ANALISIS SERTA PEMETAAN BAD CODE DAN CLEAN CODE

BAB 3 akan membahas tentang analisa-analisa yang dilakukan pada penelitian ini, dengan hasil yang akan dikaji serta pemetaan dan perbandingan bad code dan clean code guna menemukan solusi permasalah kasus penelitian ini.

BAB 4 IMPLEMENTASI REFACTORING DAN PENGUJIAN MAINTAINABILITY

BAB 4 berisikan tahapan implementasi refactoring serta pengujian maintainability berdasarkan hasl yang diperoleh dari analisis dan pemetaan.

BAB 5 KESIMPULAN DAN SARAN

BAB 5 berisikan tentang kesimpulan dari hasil penelitian ini, serta saran untuk pengembangan lebih lanjut terkait bahan penelitian.