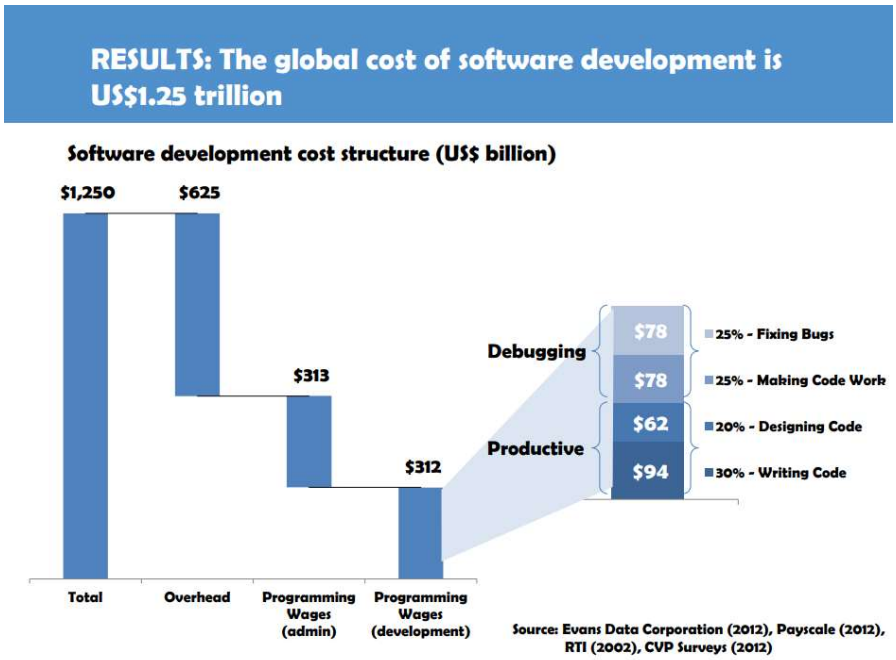
BAB I
PENDAHULUAN

1.1 Latar Belakang

Seiring dengan perkembangan era globalisasi yang serba *modern* seperti sekarang ini, terutama dalam bidang teknologi informasi, menunjukkan adanya perkembangan yang begitu pesat, baik dalam bidang kesehatan, bisnis, industri maupun aspek personal. Perkembangan teknologi yang begitu pesat ditandai dengan adanya berbagai *system* informasi, salah satunya aplikasi. Pembuatan atau perancangan suatu *system* aplikasi dilakukan oleh seorang pengembang perangkat lunak dengan menuliskan kode program (*syntax*) [1]. Dalam penulisan kode program, seringkali para *software developer* menghadapi kendala. Salah satu kendala yang sering muncul karena adanya anomali pada struktur kode program [2].

Kendala tersebut mengharuskan pengembang perangkat lunak menghabiskan waktu yang signifikan untuk melakukan *debugging* yang berfungsi untuk mendeteksi dan memperbaiki anomali pada struktur kode program. Berdasarkan penelitian yang diterbitkan oleh *CiteSeerX*, ditemukan bahwa pengembang perangkat lunak mengalokasikan 50% dari total waktu pemrograman mereka untuk *debugging*. Penelitian ini dilakukan melalui survei *Cambridge Venture Project* (CVP) dan wawancara dengan pengembang perangkat lunak yang menyatakan bahwa setengah dari waktu pemrograman digunakan untuk memperbaiki *bug* dalam kode program mereka. Hal ini menunjukkan bahwa anoamali merupakan salah satu aktivitas dominan dalam siklus pengembangan perangkat lunak [3], seperti yang ditunjukkan Gambar 1.1



Gambar 1. 1 *Software Development cost structure*

Anomali sendiri merupakan suatu keanehan, ketidakbiasaan dan penyimpangan dari rutinitas atau kondisi normal yang berbeda dari norma mayoritas pada program yang digunakan dalam pengoperasian *computer.* Dampak yang ditimbulkan akibat adanya fungsi anomali pada kode program dapat berpotensi berbahaya, seperti menyebabkan kerentanan keamanan, gangguan pada kinerja *system*, kesalahan dalam pemrosesan data, hingga kegagalan aplikasi yang dapat berdampak pada pengguna dan operasional bisnis. Pada pengembangan perangkat lunak atau kode program tahapan dalam mendeteksi anomali sangat penting agar proses perancangan kode program menjadi efisien dan aman [4].

Salah satu contoh fungsi anomali pada kode program yang dapat membahayakan *user* dalam suatu aplikasi yaitu "READ_PHONE_STATE". Fungsi tersebut merupakan *permission* pada android yang dapat mengakses informasi pribadi dari pengguna seperti informasi mengenai status panggilan yang sedang aktif dan informasi mengenai daftar akun telepon yang tersedia pada perangkat telepon pengguna. Contoh lain fungsi anomali yang dapat membahayakan user yaitu "GET_ACCOUNTS" yang dapat mengakses informasi akun atau kontak yang ada pada telepon pengguna [5].

Sehingga, penting untuk membangun *system* pendeteksi anomali dalam suatu aplikasi guna mengidentifikasi dan mencegah penggunaan izin yang berisiko terhadap privasi pengguna. *System* ini dapat membantu mendeteksi perilaku mencurigakan dalam kode program, seperti akses yang tidak wajar terhadap data sensitif. Dengan adanya pendeteksi anomali, pengembang dapat segera mengambil tindakan untuk membatasi atau menghapus izin yang tidak diperlukan, sehingga mengurangi risiko penyalahgunaan data. Dampaknya, keamanan dan kepercayaan pengguna terhadap aplikasi akan meningkat, serta potensi kebocoran informasi pribadi dapat diminimalkan.

Pada *system* yang akan dibuat, akan ada sebuah fitur deteksi anomali yang tidak hanya menginputkan sebuah *File* untuk dideteksi tetapi juga dapat menginputkan folder sehingga akan mempermudah *user* dalam mendeteksi anomali. Maka dari itu, untuk mendukung dalam merancang sebuah *system* pendeteksi anomali ini agar menjadi maksimal maka dibutuhkan sebuah penelitian dengan menggunakan metode *prototype* yang diterapkan dalam pembuatan aplikasi tersebut. Dengan metode *prototype*, pengembang dapat mengidentifikasi kelemahan dan melakukan perbaikan lebih cepat berdasarkan umpan balik pengguna. Kelebihannya adalah fleksibilitas dalam pengembangan, efisiensi waktu, serta kemudahan dalam mengadaptasi perubahan kebutuhan. Selain itu, metode ini memungkinkan pengujian awal terhadap keefektifan *system* dalam mendeteksi anomali sebelum diterapkan secara luas, sehingga dapat memastikan bahwa *system* bekerja dengan optimal sebelum diimplementasikan secara penuh. Metode ini menitikberatkan pada bagaimana sebuah *system* akan ditampilkan dan dipersepsikan. *Prototype* yang dihasilkan akan dievaluasi oleh *user* untuk mengidentifikasi kebutuhan pengembangan dari *system*. Pendekatan *prototyping* melibatkan beberapa proses, yaitu mengumpulkan informasi dari *user*, membuat dan mendesain *prototype*, mengevaluasi desain *prototype*, *develop* program atau *system*, serta mengevaluasi *system* dan pengujian untuk evaluasi dari *system* [6].

Dalam proses pengkodean atau *develop* program, pada umumnya kode yang telah dihasilkan memiliki tingkat kompleksitas yang tinggi karena terdiri

dari ribuan hingga ratusan ribu baris kode dalam program dan seringkali para pengembang perangkat lunak mengalami kendala dalam kode yang telah dibuat [7]. Salah satu penyebab kendala yang dialami adalah keberadaan anomali. Anomali tersebut dapat menyebabkan masalah dalam kode program dari tingkatan kecil hingga dampak yang serius pada *system*. Namun, dalam kenyataannya, tidak semua pengembang perangkat lunak memiliki kemampuan untuk mengidentifikasi anomali dengan cepat dan tepat, sehingga memerlukan dukungan dari aplikasi yang mampu mendeteksi anomali pada kode program yang telah dibuat [8].

Penelitian ini membuat *system* deteksi anomali pada media *input* tidak hanya berupa *file* tetapi juga berupa folder sehingga dapat memudahkan *user* dalam penginputan karena tidak memerlukan ekstraksi terlebih dahulu serta dibuatnya fitur *search* sehingga kata yang terdapat dalam kode program yang dicari dapat ditemukan dengan cepat dan mudah. Selain itu, ekstensi *file* yang dideteksi tidak hanya berupa TXT, tetapi dapat mencakup XML serta menggunakan bahasa pemrograman *python* dan *JavaScript*.

Berdasarkan latar belakang yang telah diuraikan diatas, peneliti bertujuan untuk membuat aplikasi pendeteksi anomali untuk membantu para pengembang perangkat lunak dalam mendeteksi anomali pada kode program dengan judul "Penerapan Metode *Prototyping* dalam Perancangan Aplikasi Pendeteksi Anomali Pada Kode Program". Aplikasi yang dibuat pada teks editor VS*Code* (*Visual Studio Code*) dalam penulisan kode programnya, lalu menggunakan bahasa pemrogramman *python* dan *javascripst*. Oleh karena itu, penelitian ini bertujuan untuk membuat sebuah aplikasi pendeteksi anomali kode program menggunakan metode *prototype*.

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas, diketahui permasalahan dalam penelitian ini, yaitu belum adanya pembuatan suatu *system* aplikasi deteksi yang didalamnya dapat melakukan *scanning* deteksi anomali yang dapat melakukan *input* tidak hanya berupa *file*, tetapi juga berupa folder serta dapat melakukan *search* untuk mencari *anomaly* pada *file* yang telah diinputkan.

1.3 Pertanyaan Penelitian

Adapun pertanyaan peneliti dalam melakukan penelitian ini yaitu:

1. Bagaimana cara merancang *system scanning* deteksi *anomaly* dan menambahkan fungsi-fungsi *anomaly* pada kode program?
2. Bagaimana cara merancang *system* untuk menambahkan fitur *search anomaly* sehingga kata dalam kode program dapat langsung ditemukan dengan cepat dan mudah?
3. Bagaimana cara merancang *system* agar ekstensi *file* yang dideteksi tidak hanya TXT, tetapi mencakup ekstensi XML dan dapat menginputkan sebuah folder?

1.4 Batasan Masalah

Dalam mewujudkan penelitian yang tepat dengan masalah yang ada, batasan-batasan masalah penelitian dapat diuraikan sebagai berikut :

1. Penelitian berfokus pada deteksi dan *scanning anomaly* yang terjadi dalam kode program
2. Metode yang akan digunakan dalam penelitian ini yaitu menggunakan metode *Prototype* sebagai pendekatan dalam pengembangan aplikasi deteksi *anomaly* kode program.
3. Bahasa pemrograman yang digunakan dalam pembuatan aplikasi yaitu *python* dan *JavaScript*.
4. Bahasa pemrograman yang akan dideteksi meliputi Bahasa yang digunakan pada aplikasi berbasis *mobile*, seperti *Kotlin*, C/C++, *Java*, *Python*.
5. Ekstensi *file* yang dapat dideteksi XML dan TXT.

1.5 Tujuan Penelitian

Adapun tujuan yang diperoleh dari penelitian ini, yaitu untuk mendeteksi *anomaly* dan melakukan *scanning* pada sebuah kode program yang tidak hanya menginputkan *file*, tetapi juga dapat menginputkan folder. Pada *system* yang akan dibangun juga akan membuat fitur *search* sehingga dengan kata kunci tertentu akan menampilkan beberapa kata yang serupa pada *file* yang telah dideteksi.

1.6 Manfaat Penelitian

Berdasarkan rumusan masalah, batasan masalah dan tujuan penelitian yang telah diuraikan diatas, maka dapat diketahui manfaat yang dapat diperoleh dari penelitian ini adalah:

1. Dapat meningkatkan pengalaman kepuasaan *user*. *User* akan membuat perangkat lunak yang lebih aman melalui deteksi *anomaly* kode program.
2. Dapat memahami penerapan metode *prototype* dalam perancangan suatu *system* aplikasi *scanning anomaly* pada kode program.