BAB I

PENDAHULUAN

1.1 Latar Belakang

Pemrograman merupakan bagian dari keilmuan bidang Ilmu Komputer. Pemrograman diambil dari kata "program", yang menurut KBBI berarti "program yang diciptakan secara khusus sehingga memungkinkan komputer melakukan fungsi tertentu". Sedangkan pihak yang melakukan pemrograman disebut dengan programmer. Pada Program Studi Ilmu Komputer Universitas Pendidikan Indonesia (UPI), terdapat mata kuliah Algoritma dan Pemrograman yang mengajarkan mahasiswa dasar-dasar pemrograman. Dalam proses pembelajaran pemrograman, mahasiswa tidak hanya sekedar mempelajari teori pemrograman saja, namun juga harus diiringi latihan yang rutin agar dapat menguasai materi pemrograman sepenuhnya dan bisa mengaplikasikannya dengan baik. *Programmer* yang baik tidak akan hanya memperhatikan sisi fungsionalitasnya saja, namun harus juga memperhatikan sisi dokumentasinya.

Dalam dokumentasi *source code*, *programmer* sebaiknya memperhatikan apakah *source code* yang dibuat telah ditulis dengan baik. Karena dokumentasi yang kurang baik dapat menimbulkan kesalahpahaman (Buse & Weimer, 2010) dan menyulitkan proses pemeliharaan dan pengembangan selanjutnya (Lientz, 1983). Penulisan *source code* yang baik dapat terlihat dari beberapa kriteria. Pertama, *source code* yang dibuat harus *aesthetic*. Kriteria ini menguji penulisan *source code* yang mudah dibaca dan dipahami. Salah satu contohnya adalah penggunaan indentasi pada *condition if* dengan *statement* didalamnya yang harus lebih menjorok kedalam. Kriteria selanjutnya yang harus diperhatikan adalah *Efficient*. Kriteria ini menguji pemahaman mahasiswa terhadap penggunaan struktur program dalam kasus yang dihadapi. Sebagai contoh adalah penggunaan "break" pada perulangan *for*. Hal ini akan menunjukkan bahwa pembuat *source code* tersebut tidak paham kapan posisi berhenti yang tepat untuk kasus yang dihadapi (Kurnia dkk., 2001).

Di Ilmu Komputer Universitas Pendidikan Indonesia (UPI), tepatnya pada mata kuliah Algoritma dan Pemrograman. Dosen pengampu dan asisten dosennya memanfaatkan Sistem *Online Judge* bernama *Computer Science Programming Contest* (CSPC) (Sukamto, 2011), yaitu sebuah program penilaian *source code* program otomatis. Sistem ini biasa digunakan dalam proses latihan, kuis, evalusi, ujian tengah semester, ujian akhir semester dan perlombaan. Pada proses-proses tersebut, CSPC memberikan kemudahan dalam penilaian ketepatan *output* dari *source code* yang dibuat oleh mahasiswa yang mengontrak. Mahasiswa dapat mengunggah *file source code* lebih dari 1 kali dengan indikasi warna hijau jika *output source code* yang diunggap benar dan indikasi warna merah jika *output source code* yang diunggah salah.

Penentuan benar atau salah dari *source code* yang diunggah oleh mahasiswa yang mengontrak adalah kesamaan antara *source code* mahasiswa dengan *test case* yang disediakan sesuai soal yang diberikan. *Test case* ini terdiri dari *input* yang akan di *scan* oleh program dan *output* yang diharapkan dari program sesuai dengan input yang ditetapkan. *Test case* tersebut disediakan dan dimasukkan kedalam soal oleh penyelenggara baik dosen pengampu maupun asisten dosen.

Dalam pemanfaatan CSPC untuk menilai kebenaran *output* dari *source code* yang di unggah oleh mahasiswa yang mengontrak, CSPC belum dapat menilai penulisan *source code* secara otomatis, sehingga untuk penilaian tersebut harus tetap melalui penilaian manual. Namun penilaian secara manual memiliki kekurangan pada beberapa aspek. Pertama, Aspek *correctness*. Aspek ini menjadi suatu kekurangan ketika mahasiswa yang mengontrak diharapkan untuk menyelesaikan suatu soal. Namun ada kemungkinan bahwa penilai yang satu dan yang lainnya memiliki bobot penilaian yang berbeda terhadap penggunaan algoritma yang dipakai oleh masing-masing mahasiswa. Sehingga penilaian dapat menjadi menjadi subjektif dan tidak konsisten (Lampiran 31). Aspek selanjutnya adalah *aesthetic*. Aspek ini dapat menjadi masalah ketika penggunaan indentasi masing-masing mahasiswa berbeda. Maka ada kemungkinan penilaian yang diberikan penilai hanya berdasarkan gaya penulisan indentasi yang masing-masing penilai inginkan. Aspek lainnya adalah *runtime*. Hal ini menjadi masalah

karena lama waktu eksekusi suatu program bergantung juga pada fasilitas yang digunakan. Sehingga penilai yang berbeda dapat memberi penilaian berbeda. Aspek selanjutnya adalah aspek *efficiency*. Aspek ini menunjukkan bahwa waktu yang dibutuhkan dalam penilaian manual dapat menyulitkan penilai (Ruslan dkk., 2018). Penilaian manual juga dapat memberatkan penilai jika jumlah mahasiswa yang harus dinilai terlalu banyak. Hal ini dapat mengakibatkan performa penilai menurun terutama jika waktu untuk menilai hanya sedikit (Lampiran 31). Sehingga penilaian otomatis akan sangat membantu baik bagi dosen maupun asisten dosen dalam proses penilaian.

Penelitian tentang penilaian *source code* otomatis yang dibuat untuk CSPC bukanlah hal baru. Penelitian skripsi berjudul "Perhitungan *Text Similarity* Berbasis *Word Embedding* dengan *Word Mover Distance* untuk Penilaian Komentar Otomatis dalam Sistem *Online Judge*" (Rischa dkk. 2021) menunjukkan bahwa pembuatan suatu modul penilaian otomatis untuk membantu meringankan pekerjaan penilai sangat mungkin untuk dilakukan. Perbedaan dari penelitian tersebut dengan penelitian yang penulis usulkan adalah pada penelitian tersebut yang dinilai oleh modul penilaian otomatis adalah komentar, sedangkan yang akan dinilai pada penelitian yang penulis usulkan adalah penulisan *source code* yang dikirim mahasiswa. Sehingga penulis membutuhkan metode yang berbeda untuk menilai penulisan *source code* tersebut.

Terdapat penelitian berjudul "*Analogy Mapping Development for Learning Programming*" yang menunjukkan bahwa dengan metode Model *State Machine* dapat memungkinkan suatu program mengetahui bagaimana penulisan dari *source code* yang di *input* (Sukamto dkk., 2017) . Pada penelitian tersebut, peneliti membuat berbagai model state machine yang dapat menerima berbagai bentuk penulisan. Salah satu model yang dibuat adalah untuk menerima penulisan *source code* perulangan *for*. Model yang dibuat akan mengecek apakah saat ini ditemukan ";". Setelah ditemukan maka akan mengecek kembali apakah kata setelahnya adalah "for" dan karakter setelahnya adalah "(". Jika kondisi tersebut terpenuhi maka dapat dikatakan bahwa pada *source code* tersebut terdapat perulangan *for*. Dari penelitian tersebut tersebut, penulis mengusulkan untuk menerapkan metode Model *State Machine*.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, maka rumusan masalah penelitian ini adalah sebagai berikut.

1) Bagaimana cara menerapkan model *state machine* pada modul penilaian penulisan *source code* otomatis?
2) Bagaimana hasil penerapan model *state machine* dalam penilaian penulisan *source code* otomatis setelah dihitung menggunakan pendekatan nilai koefisien korelasi?

1.3 Tujuan

Tujuan penelitian ini adalah sebagai berikut.

1) Menerapkan model *state machine* pada modul penilaian penulisan *source code* otomatis.
2) Mengetahui hasil implementasi model *state machine* dalam penilaian penulisan *source code* otomatis menggunakan pendekatan nilai koefisien korelasi.

1.4 Manfaat Penelitian

Beberapa manfaat dari penelitian ini adalah sebagai berikut.

1) Membuka kemungkinan perancangan modul penilaian lain yang belum dimiliki CSPC.
2) Meringankan beban penilai untuk kasus penilaian yang termasuk kriteria penilaian.
3) Menjadi referensi untuk penilitian serupa di masa depan.

1.4 Batasan Masalah

Batasan masalah pada penelitian ini adalah sebagai berikut.

1) *Source code* yang diolah harus dalam bahasa pemrograman C.
2) Penilaian penulisan dilakukan pada *source code* peserta CSPC.

3) Kriteria penilaian penulisan hanya berfokus terhadap indentasi, penggunaan *semicolon,* penggunaan *curly parentheses*, fungsi percabangan (*if, else if,* dan *else*) dan perulangan (*for, while* dan *do while*).

4) Ketepatan penggunaan komentar tidak menjadi penilaian pada penelitian ini.

5) Konsep kompilasi tidak dilibatkan dalam penelitian ini sehingga kriteria penilaian yang membutuhkan *input user* tidak digunakan.

6) Implementasi dilakukan pada aplikasi tunggal dengan *database* lokal.

1.5 Sistematika

Sistematika penulisan berfungsi sebagai pedoman bagi penulis untuk struktur penulisan yang sistematis agar mencapai tujuan penelitian. Sistematika penulisan terdiri dari 5 bab yaitu sebagai berikut.

Bab I Pendahuluan

Bab ini menjelaskan tentang seberapa penting pembangunan modul penilaian penulisan *source code* otomatis dan masalah apa yang akan di selesaikan dengan adanya modul tesebut.

Bab II Kajian Teori

Bab ini berisi teori-teori yang dikutip dari jurnal, skripsi, atau buku yang menjadi dasar dari penelitian ini

Bab III Metode Penelitian

Bab ini menjelaskan tahapan yang dilakukan dalam penelitian ini. Tahapan tersebut dimulai dari studi literatur dan diakhiri dengan penarikan kesimpulan. Bab ini juga menjelaskan bahwa dalam proses pembangunan aplikasi, penulis menggunakan metode *linear sequential model* yang berisi tahapan pembangunan aplikasi dimulai dari analisis kebutuhan hingga pengujian.

Bab IV Hasil dan Pembahasan

Bab ini menjelaskan hasil dari penelitian yang dimulai dengan pengumpulan kriteria penilaian, perancangan model *state machine,* pembangunan aplikasi hingga pengujian beserta pembahasannya.

Bab V Kesimpulan dan Saran

Bab ini menjelaskan Kesimpulan yang ditarik atas hasil penelitian yang didapatkan yaitu nilai koefisien korelasi antara penilaian modul dan manusia yang cukup tinggi. Pada bab ini juga penulis menyampaikan saran untuk melakukan pengembangan lebih lanjut agar dapat mencapai nilai korelasi yang l