BAB I
PENDAHULUAN

1.1 Latar Belakang

Pengembangan perangkat lunak adalah bidang yang sangat menarik dan penting dalam dunia teknologi informasi saat ini. Dalam rangka menciptakan perangkat lunak yang handal, mudah dipahami, dan mudah dirawat, kualitas kode program yang baik menjadi faktor kunci yang harus diperhatikan dengan serius. Namun, Salah satu masalah yang sering muncul dalam pengembangan perangkat lunak adalah adanya *code smell*. *Code smell* merujuk pada tanda-tanda atau karakteristik dalam kode program yang mengindikasikan adanya kelemahan desain atau praktik buruk yang dapat mempengaruhi kualitas dan pemeliharaan perangkat lunak. *Code smell* bisa berupa pola-pola yang tidak efisien, pengulangan kode yang tidak perlu, atau struktur kode yang kompleks dan sulit dipahami.

Code smell dapat menyebabkan berbagai masalah dalam perangkat lunak. Salah satunya adalah kesulitan dalam memahami kode. *Code smell* yang tidak diatasi dapat membuat kode program sulit dipahami oleh pengembang lain atau bahkan oleh pengembang yang sama setelah beberapa waktu. Hal ini dapat menghambat kolaborasi tim, memperlambat proses pengembangan, dan meningkatkan risiko kesalahan (Fowler, 2019). Selain itu, *code smell* dapat memiliki dampak negatif yang signifikan terhadap perangkat lunak. *Code smell* dapat menyebabkan penurunan kualitas, kebingungan, dan kesulitan dalam pemahaman kode. Selain itu, *code smell* juga dapat meningkatkan kompleksitas, memperburuk desain, dan mempengaruhi kemudahan perubahan pada perangkat lunak. Ketika *code smell* tidak ditangani dengan tepat, perangkat lunak dapat menjadi sulit dipelihara, rentan terhadap *bug*, dan sulit dikembangkan untuk memenuhi kebutuhan pengguna (Martin, 2009). Oleh karena itu, penting untuk mendeteksi dan mengatasi *code smell* sejak awal dalam siklus pengembangan perangkat lunak. Dengan menggunakan pendekatan *software metrics*, pengembang dapat mengukur dan menganalisis kualitas dan karakteristik perangkat lunak secara

kuantitatif. Salah satu pendekatan yang digunakan adalah dengan memanfaatkan struktur *Abstract Syntax Tree (AST)* dari kode program.

Dikutip dari indeks komunitas pemrograman *TIOBE*, bahasa pemrograman *Python* telah menjadi salah satu bahasa pemrograman yang paling populer, menduduki peringkat pertama pada Juni 2023. Kesederhanaan sintaksis *Python* dan keluwesannya dalam berbagai bidang seperti analisis data, pembelajaran mesin, dan pengembangan *web* telah memberikan kontribusi signifikan pada popularitasnya.

Software metrics merujuk pada pengukuran dan analisis kuantitatif terhadap atribut dan karakteristik perangkat lunak. Dalam konteks deteksi *code smell*, analisis struktur *AST* menggunakan *software metrics* dapat membantu mengidentifikasi pola-pola dan karakteristik dalam kode program yang berkaitan dengan *code smell* atau kelemahan desain. Metrik yang diperoleh dari analisis struktur *AST* dapat digunakan untuk mendeteksi *code smell* dan mengambil tindakan perbaikan yang tepat.

Beberapa penelitian telah dilakukan untuk mendeteksi *code smell* dengan pendekatan *software* metrik. (Kovaˇcevi´c, dkk., 2022) mengimplementasikan teknik *embedding* kode sumber *neural* dan menghasilkan performa terbaik dalam mendeteksi *Long Method* dan *God Class*. Penelitian lain yang dilakukan oleh (Sa'adah, dkk., 2022) mengembangkan sebuah *tool refactoring* otomatis untuk mendeteksi dan mengatasi *lazy class code smell* pada bahasa pemrograman *Java*. Tool ini menggunakan pendekatan *software* metrik untuk mendeteksi *code smell* serta melakukan *refactoring* otomatis untuk meningkatkan kualitas desain kode. Pada penelitian lain yang dilakukan oleh (Azwega, dkk., 2020) menggunakan perhitungan *software* metrik menjadi alat ukur untuk mengklasifikasikan jenis *code smell god class* dan *brain class*. Penelitian sebelumnya telah berhasil dalam mendeteksi *code smell* pada bahasa pemrograman *Java*, namun terdapat kekurangan dalam kurangnya penelitian yang fokus pada deteksi *code smell* pada bahasa pemrograman *Python*.

Berdasarkan kekurangan yang telah disebutkan sebelumnya, penelitian ini mengembangkan sistem yang mampu mendeteksi *code smell* pada bahasa *Python*.

Diharapkan bahwa hasil penelitian ini akan memberikan kontribusi positif dalam memperbaiki kualitas perangkat lunak *Python*.

1.2 Rumusan Masalah

Rumusan masalah dalam penelitian ini adalah bagaimana membuat sebuah sistem aplikasi yang dapat mendeteksi *code smell* pada bahasa pemrograman *Python* dengan menggunakan pendekatan *software metrics* pada struktur *Abstract Syntax Tree (AST)*.

1.3 Batasan Masalah

Penelitian ini memiliki batasan masalah, yaitu difokuskan pada bahasa pemrograman *Python* dengan menggunakan struktur *Abstract Syntax Tree* sebagai representasi kode program yang dianalisis, dengan penekanan pada deteksi *code smell long method, lazy class, feature envy,* kompleksitas kode dan tidak melibatkan interaksi dengan komponen eksternal seperti *database* atau *API*.

1.4 Tujuan

Tujuan penelitian ini adalah membuat sistem aplikasi *Code Smell Detection* untuk mendeteksi *code smell* pada bahasa pemrograman *Python* dengan menggunakan *software metrics* pada struktur *Abstract Syntax Tree.*

1.5 Manfaat

Adapun manfaat dari penelitian ini adalah sebagai berikut:

1. Membantu pengembang dalam mendeteksi dan memahami potensi kelemahan desain atau praktik buruk yang dapat mempengaruhi kualitas perangkat lunak.
2. Dengan mendeteksi *code smell* secara dini, pengembang dapat melakukan tindakan perbaikan yang spesifik, seperti menghindari *Lazy Classes*, mengatasi *Long Methods*, dan mengurangi *Feature Envy*.

3. Membantu pengembang dalam merencanakan dan melaksanakan pemeliharaan kode dengan lebih efisien.
4. Meningkatkan pemahaman pengembang terhadap struktur kode dan membantu mereka mengambil keputusan yang lebih baik dalam proses pengembangan.
5. Memberikan kontribusi pada ekosistem *Python* dengan menyediakan sistem yang dirancang untuk mendeteksi *code smell Lazy Class, Long Method, Feature Envy* dan Kompleksitas Kode.