BAB I
PENDAHULUAN

1.1. Latar Belakang

Rekayasa perangkat lunak adalah suatu strategi pengembangan perangkat lunak yang melingkupi lapisan proses, metode, dan *tools* **Error! Reference source not found.**. Siklus hidup pengembangan perangkat lunak terdiri tahapan *requirement*, *design*, *code*, *testing*, *deployment* dan *maintenance* **Error! Reference source not found.**. Pada pembangunan perangkat lunak skala besar, setiap penambahan fitur akan dilakukan secara *increment*. Penambahan fitur tersebut akan melewati siklus hidup pengembangan perangkat lunak beberapa kali hingga produk versi tertentu dirilis. Berdasarkan survei pengembangan perangkat lunak perusahaan mengeluarkan biaya paling besar untuk *maintenance* **Error! Reference source not found.**.

Fakta yang terjadi pada saat membangun perangkat lunak adalah kebutuhan *customer* terhadap perangkat lunak yang sering berubah dan *customer* tidak ingin menunggu lebih lama lagi untuk memperoleh perangkat lunak.

Pada dasarnya sebuah perangkat lunak dibangun dengan memperhatikan dua fokus utama [49]. Di antaranya: (1)Seberapa besar biaya yang dibutuhkan untuk membangun perangkat lunak?. (2)Berapa lama waktu yang dibutuhkan untuk membangun perangkat lunak? Kedua hal tersebut memiliki keterikatan satu sama lain. Permasalahan yang sering terjadi pada pembangunan perangkat lunak adalah produk *error* pada tahap integrasi. Jika proses penanganan *error* tersebut adalah manual maka dibutuhkan usaha maksimal dan waktu jangka panjang untuk memperbaiki produk tersebut.

Otomasi adalah kunci untuk melakukan proses yang sama berulang kali. Dengan penerapan otomasi maka proses *build*, *deploy*, *testing* dan rilis dapat dilakukan dengan cepat. Integrasi perangkat lunak juga harus dilakukan berulang kali agar dapat mengurangi resiko dan memperbaiki kualitas perangkat lunak **Error! Reference source not found.**. Integrasi tersebut dapat dilakukan dengan menggunakan praktik *Continuous Integration* (CI).

CI adalah praktik pengembangan perangkat lunak yang mengintegrasikan hasil pekerjaan anggota tim secara rutin untuk menemukan kesalahan pada proses integrasi secepat mungkin **Error! Reference source not found.**. Praktik CI dengan *toolset* dinamakan *automated* CI. Dengan penerapan *automated* CI maka perangkat lunak *customer* dipastikan dapat bekerja terhadap setiap perubahan baru. Jika ada kesalahan pada proses integrasi, maka tim dapat memperbaikinya dengan cepat. Tim yang menggunakan *automated* CI secara efektif mampu mendeteksi *bug* lebih awal, menghasilkan perangkat lunak lebih cepat dengan sedikit *bug*, mengurangi biaya dan jangka waktu perbaikan perangkat lunak pada proses *delivery* dibandingkan tim yang tidak menggunakan CI **Error! Reference source not found.**.

Automated CI akan diterapkan pada aplikasi `medrecapp` yang dibangun oleh sebuah tim. Tidak adanya *automated testing* pada tahap integrasi mengakibatkan sulitnya memperoleh produk versi tertentu tanpa cacat, meskipun tim telah menggunakan *version control system*. Dari segi kegunaan, aplikasi tersebut memiliki potensi besar untuk dapat dikembangkan dengan fitur-fitur baru yang sesuai dengan kebutuhan rumah sakit X.

Berdasarkan uraian tersebut, maka judul tugas akhir yang diangkat adalah ”Penerapan *Continuous Integration* Pada Studi Kasus Aplikasi Rekam Medis ”.

1.2. Tujuan

Berdasarkan latar belakang yang telah diuraikan sebelumnya, Tugas Akhir ini bertujuan untuk membentuk kerangka kerja yang meliputi prosedur, teknik, *toolset* yang mendukung *continuous integration* sebuah aplikasi.

1.3. Rumusan Masalah

Pada tugas akhir ini, rumusan masalah yang diangkat adalah bagaimana membentuk kerangka kerja yang meliputi prosedur, teknik, dan *toolset* yang mendukung *continuous integration* pada aplikasi `medrecapp` berbasis Java *desktop*?

1.4. Ruang Lingkup Masalah

Ada beberapa ruang lingkup pada *continuous integration* yang dilakukan, diantaranya:

1. Mencakup aplikasi `medrecapp` berbasis Java *desktop* yang sudah ada yang terdiri dari modul master, modul *front office*, modul poliklinik dan modul rekam medis. Proses *development* mencakup *compile source program*, membuat dan menjalankan *unit test* dan *integration test*.
2. Prosedur yang digunakan pada CI mencakup:
    a. Penentuan komitmen kesepatan awal.
    b. Penggunaan VCS.
    c. Pengunaan *automated testing.*
    d. Penggunaan *automated build.*
3. Teknik yang digunakan untuk *version control system*, *automated testing*, *automated build* dan *continuous integration* adalah:
    a. VCS : *distributed version control system* dengan alur kerja *centralized workflow*.
    b. *Unit test* dan *integration testing*: *automated testing* dan *buttom-up integration.*
    c. *Build tool* : *scripting* yang dihasilkan dari Netbeans dan dimodifikasi.
    d. *Continuous Integration* : menggabungkan teknik dari VCS, *testing* dan *build.*
4. *Tools* yang digunakan untuk *version control system*, *automated testing*, *continuous integration* adalah *open source* dan *free software* dengan rincian sebagai berikut:
    a. *Version control* dan *software hosting* : Git dan Github.
    b. *Unit testing* dan *integration testing*: Junit dan Fest.
    c. *Automated build* dan *CI :* Ant dan Jenkins.
5. Tidak mencakup *inspection code* dan *deployment* pada *continuous integration.*

1.5. Metodologi

Pada bagian ini diuraikan mengenai metodologi penelitian sebagai berikut:

1. Studi Literatur

    Hal yang perlu dilakukan untuk melakukan studi literatur sebagai berikut:

    a. Mengumpulkan, mempelajari dan memahami bahan dan konsep *continuous integration.*
    b. Memilih *sample* aplikasi `medrecapp` berbasis Java *desktop* yang telah dibangun oleh sebuah tim sebagai studi kasus penerapan *continuous integration.*

c. Mempelajari, memahami dan menentukan prosedur, teknik dan *toolset* yang mendukung *continuous integration* pada aplikasi `medrecapp`.

2. Analisis

   Menentukan ide pemikiran kerangka kerja yang mendukung *continuous integration* pada aplikasi `medrecapp`.

3. Implementasi dan Pengujian
   a. Menerapkan prosedur dan teknik dengan menggunakan *toolset* yang ada pada VCS, *testing*, *build* yang mendukung *continuos integration.*
   b. Menguji alur *continuous integration* pada aplikasi `medrecapp`.
   c. Kendala yang dihadapi pada saat melakukan *continuous integration*.
4. Penutup

   Berisi kesimpulan yang dapat diperoleh dari hasil implementasi dan pengujian CI pada aplikasi `medrecapp` berbasis Java *desktop* dan saran untuk perbaikan penelitian selanjutnya.