BAB V
PENUTUP

5.1 Kesimpulan

Kesimpuan yang dapat dibuat dari praktik *automated CI* pada studi kasus aplikasi rekam medis berbasis java *desktop* bernama medrecapp adalah praktik tersebut dapat memberikan manfaat sebagai berikut:

1. Pengurangan resiko kegagalan pada pembangunan aplikasi medrecapp

   Dengan menerapkan praktik *automated CI*, setiap kode program yang disimpan di sebuah *repository* pusat akan diuji secara otomatis oleh sebuah mesin integrasi. Setiap kesalahan pada pengujian tersebut akan diinformasikan kepada para *developer* secara otomatis oleh mesin integrasi, sehingga tim dapat mengetahui kesalahan tersebut dengan cepat dan dapat segera memperbaikinya sebelum kode program semakin bertambah dan kompleks.

2. Penghilangan proses manual yang berulang

   Dengan menerapkan praktik *automated CI*, para *developer* tidak lagi:

   a. Membuat catatan tentang rincian perubahan kode program pada setiap versi modul. Rincian perubahan kode program pada setiap versi modul akan dicatat secara otomatis oleh *VCS tools*.

   b. Men-*trigger* eksekusi kelas pengujian satu per satu. Semua kelas pengujian dapat dieksekusi dengan satu kali eksekusi pengujian oleh *unit testing tools*.

   c. Menguji fungsional aplikasi rekam medis medrecapp dengan mensimulasi *GUI* aplikasi secara manual dan berulang kali. Semua simulasi tersebut dapat diotomasi oleh *functional testing tools*.

   d. Melakukan rangkaian *trigger* eksekusi *build* untuk mendapatkan paket aplikasi yang berisi *file* siap pakai. Rangkaian *trigger* eksekusi *build* akan diotomasi oleh *automated build tools*, sehingga *developer* hanya perlu men-*trigger* satu kali *automated build tools*.

e. Memerlukan seorang *integrator* untuk mengintegrasikan versi modul yang benar dari setiap *developer*, menguji kode program, men-*trigger automated build tools*, menginformasikan hasil pengujian yang salah, mengarsipkan paket aplikasi yang berisi *file* siap pakai dan membuat laporan kemajuan proses pembangunan aplikasi medrecapp di mesin integrasi. Semua kegiatan pada mesin integrasi tersebut akan diotomasi oleh *automated CI tools*.

Adapun kerangka kerja untuk menerapkan praktik *automated CI* yang mencakup prosedur, teknik dan *toolset* pada pembangunan aplikasi rekam medis medrecapp adalah sebagai berikut:

1. Membagi pekerjaan pembangunan aplikasi rekam medis medrecapp menjadi modul-modul. Pembagian pekerjaan tersebut dilakukan tim agar setiap *developer* dapat membuat kode program berdasarkan satu area fungsional.
2. Menyiapkan sebuah mesin integrasi. Mesin integrasi tersebut akan mengintegrasikan modul dari *repository* pusat dan membuat paket aplikasi secara otomatis berdasarkan penjadwalan yang dibuat oleh tim. Untuk menjadwalkan kegiatan tersebut, tim perlu membuat kesepakatan terlebih dahulu. *Automated CI tools* yang digunakan tim untuk menjadwalkan pengintegrasian modul dan pembuatan paket aplikasi rekam medis medrecapp adalah Jenkins.
3. Menyiapkan sebuah *repository* pusat. *Repository* pusat digunakan tim agar setiap *developer* tidak salah dalam memahami versi modul yang telah mereka buat. Jasa penyedia layanan penyimpanan versi kode program terpusat yang digunakan tim untuk membangun aplikasi rekam medis medrecapp adalah Github.
4. Meng-*clone repository* pusat. Kegiatan tersebut dilakukan setiap *developer* untuk mendapatkan *working directory* dengan *environment* yang sama. Dengan meng-*clone repository* pusat, *developer* tetap dapat menyimpan versi modul walaupun sedang dalam keadaan tidak

terhubung dengan *repository* pusat. *VCS tools* yang digunakan tim untuk menyimpan versi secara terdistribusi pada pembangunan aplikasi rekam medis medrecapp adalah Git.

5. Membuat *build script*. *Script* tersebut berisi serangkaian *target* dan *task* yang akan dieksekusi oleh *automated build tools*. *Build script* dibuat oleh salah satu *developer* untuk menyamakan alur kerja tim pada mesin lokal dan mengotomasi kegiatan di mesin integrasi. *Automated build tools* yang digunakan tim pada pembangunan aplikasi rekam medis medrecapp adalah Ant.
6. Mengotomasi pengujian unit. Untuk menguji unit aplikasi medrecapp, *developer* perlu membuat kode pengujian. *Unit testing tools* yang digunakan tim untuk mengotomasi eksekusi kode pengujian aplikasi medrecapp adalah JUnit.
7. Mengotomasi pengujian fungsional. Pengujian fungsional dilakukan setiap *developer* untuk memastikan bahwa kebutuhan fungisonal dari *customer* telah terpenuhi. Pengujian tersebut jika dilakukan secara manual, maka akan membutuhkan usaha yang besar. Oleh karena itu, pengujian fungsional perlu diotomasi. *Functional testing tools* yang digunakan tim untuk mengotomasi pengujian fungsional aplikasi medrecapp adalah FEST.
8. Menyimpan versi modul hanya yang sudah lolos pengujian. Setiap *developer* perlu menguji modul terlebih dahulu pada mesin lokal sebelum menyimpan versi modul tersebut ke *repository* lokal. Kegiatan tersebut dilakukan agar dapat meminimalisasi kesalahan pada saat integrasi modul.
9. Mengambil versi modul dari *repository* pusat. Sebelum *developer* menyimpan versi modul ke *repository* pusat, *developer* perlu mengambil versi modul dari *repository* pusat terlebih dahulu. Kegiatan tersebut dilakukan untuk meminimalisasi kesalahan pada integrasi modul sebelum dijadikan paket aplikasi oleh mesin integrasi.

10. Mengotomasi pemberian notifikasi kesalahan dari mesin integrasi ke setiap *developer*. Tim perlu mengkonfigurasi *tool* Jenkins terlebih dahulu untuk memberikan notifikasi ketika terjadi kesalahan pada satu atau lebih hasil pengujian. Dengan notifikasi tersebut, *developer* dapat memperbaiki kode program sesegera mungkin.
11. Mengotomasi pengarsiapan paket aplikasi di mesin integrasi. Setiap pembuatan paket aplikasi rekam medis yang berhasil dilakukan *tool* Jenkins, akan disimpan secara otomatis di mesin integrasi. Pengarsipan tersebut dilakukan agar tim dapat mengetahui *history* dari paket aplikasi yang telah dibuat *tool* Jenkins dan mengaksesnya.
12. Mengotomasi pembuatan laporan kemajuan proses pembanguan aplikasi. Dengan laporan kemajuan tersebut, setiap *developer* dapat mengetahui *milestone* dari pembangunan aplikasi rekam medis medrecapp. Pembuatan laporan tersebut jika dilakukan secara manual, maka akan membutuhkan usaha yang besar. *Tool* Jenkins dapat mengotomasi pembuatan laporan tersebut setiap berhasil atau gagal membuat paket aplikasi.

5.2 Saran

Intinya prosedur teknik dan toolset pada automated CI dapat berbeda-beda. Penggunaan toolse juga tergantung dengan prosedur dan teknik masing-maisng.

Penngunaan dibutuhkan teri yg lain, stannya bagi yang tidakn wprd sebaiknya konsep.

Bisa memahami konsep terlebih dahulu sebelum munggunkan toolset.

Tidak semua ofiice yg ain bsa mmbuka itu, dibutuhkan diplin ilmu yg trkait dgn.

Setiap tim memilki prosedur yang berbeda pada pembangunan perangkat lunak. Olhe karena itu, proseude, teknik dan toolset yang digunakan belum tenteu daoat ddipakai pada pembanguan perangkat lunakyna g dilakuan tim lain.

Tugas Akhir ini masih belum sempurna. Maka ada baiknya jika dapat dilanjukan atau digunakan sebagai referensi untuk mengetahui metode membangun perangkat lunak. *Continuous Integration* merupakan metode sangat baik untuk menghasilkan perangkat lunak berkualitas dalam pembangunan perangkat lunak bagi tim pengembang.

Continuous Integration dapat dilakukan dengan teknik dan *tool* yang berbeda dari Tugas Akhir ini. Pembaca dapat mencari referensinya di Buku *Continuous Integration* (karya Marthin Fowler) atau *internet*. Perlu diingat juga sebelum melakukan praktik *continuous integration*, maka anggota tim harus mempersiapkan *version control system, automated testing* dan *automated build*.