Nama Kelompok :

1. Khoirum Nurfatikha A (202412030)
2. Mosyarofah (202412031)
3. Yovitha Gracia Tavares (202412044)

Mata kuliah : Rekayasa Perangkat Lunak

LAPORAN MANAJEMEN KONFIGURASI PERANGKAT LUNAK (SCM)

WEBSITE CUCI SEPATU

A. Penentuan proyek

Proyek **CleanKicks** bertujuan mengembangkan sistem informasi berbasis *website* untuk layanan cuci sepatu. Sistem ini mencakup tiga pengguna utama: Pelanggan (untuk pemesanan dan pelacakan *laundry*), Staf (untuk *update* status pengerjaan), dan Admin (untuk manajemen data dan laporan).

Tujuan Proyek:

1. Menyediakan platform pemesanan *online* yang mudah diakses.
2. Meningkatkan efisiensi pelacakan status pekerjaan (*real-time*).
3. Mendukung pertumbuhan bisnis *laundry* sepatu melalui digitalisasi.

B. Identifikasi dan Dokumentasi Configuration Items (CI)

Semua artefak yang penting untuk membangun, menguji, dan memelihara proyek CleanKicks diidentifikasi sebagai Item Konfigurasi (CI) dan dikendalikan oleh SCM.

CI ID	Nama CI	Tipe	Lokasi Kontrol
CI-001	Source Code Frontend	Kode	Repositori **Git** (`/src/frontend`)
CI-002	Source Code Backend	Kode	Repositori **Git** (`/src/backend`)
CI-003	Skema dan Migrasi Basis Data	Skrip	Repositori **Git** (`/database`)
CI-004	Dokumen Spesifikasi Kebutuhan	Dokumen	**Confluence** (Diatur via CI-006)
CI-005	Dokumen Pengujian (*Test Cases*)	Dokumen	**Jira** / Repositori **Git** (`/tests`)
CI-006	Environment Configuration Files	Konfigurasi	Repositori **Git** (`.env`, `Dockerfile`)

C. Penggunaan Sistem Version Control (Git) dan Proses

Kami menggunakan **Git** sebagai Sistem Kontrol Versi dan **GitHub** sebagai *Remote Repository*

Aktivitas	Prosedur	Kepatuhan
Inisiasi Proyek	Repositori tunggal (*Monorepo* atau *Polyrepo*) dibuat dan *branch* `main` di-*protect*.	100%
Kontrol Akses	Hanya *Lead Developer* yang memiliki izin *merge* ke `develop` dan `main`.	100%
Proses *Commit*	Setiap *commit* harus menyertakan pesan yang deskriptif (**feat/fix/docs/style**).	**95%** (Perlu peningkatan pada konsistensi `style:` *commits*).
Code Review	Semua *Merge Request* (MR) ke `develop` harus melalui **minimal 2** persetujuan *developer* lain.	100%

D. Penerapan Branching Strategy dalam Git

Proyek mengadopsi modifikasi dari model **Gitflow** untuk menjaga stabilitas *branch*:

1. **main**: Mencerminkan kode *Production*. Hanya menerima *merge* dari release atau hotfix.
2. **develop**: *Branch* Integrasi Utama. Tempat semua fitur yang disetujui di-*merge*.
3. **feature/nama-fitur**: Dibuat dari develop untuk pengembangan fitur baru.
4. **hotfix/deskripsi**: Dibuat dari main untuk perbaikan *bug* mendesak di *Production*. Setelah selesai, di-*merge* kembali ke main dan develop.

E. Continuous Integration (CI) dan Continuous Deployment (CD)

Kami menggunakan **GitLab CI/CD** untuk otomatisasi *build* dan *deployment*.

Fase CI/CD	Deskripsi	Status Implementasi
Continuous Integration (CI)	Memicu *build* otomatis dan pengujian unit (*Unit Tests*) setiap kali terjadi *push* ke *branch* `develop` dan *feature*.	Berjalan Penuh
Continuous Delivery (CD)	Secara otomatis membuat paket rilis (*Docker Image*) ketika *merge* berhasil ke *branch* `develop`.	**Berjalan Penuh** (Deploy ke *Staging*)
Continuous Deployment (CD)	*Deployment* otomatis ke lingkungan *Production* (dari `main`) dilakukan setelah *Manual Approval* oleh Manajer Proyek dan Tim QA.	**Manual Approval** (Mematuhi kebijakan risiko)

F. Manajemen Perubahan dan Dokumentasi

Permintaan Perubahan (*Change Request*) dikelola menggunakan **Jira**.

1. **Pengajuan**: Perubahan diajukan sebagai **Issue** di Jira (tipe: *Bug*, *Task*, *New Feature*).
2. **Peninjauan (CCB)**: Manajer Proyek, *Lead Developer*, dan *Product Owner* berfungsi sebagai Dewan Kontrol Konfigurasi (CCB).
3. **Implementasi**: Setiap *commit* atau *Merge Request* harus merujuk ke ID *Issue* Jira terkait (misalnya, `fix: Memperbaiki error [CK-101]`).
4. **Dokumentasi**: Dokumen (CI-004) diperbarui di **Confluence** setiap kali *milestone* tercapai atau ketika terdapat perubahan besar pada kebutuhan/desain.

G. Proses Version Control dengan Git

Hasil: Tim berhasil mengelola lebih dari **250 *commits*** dan **40 *Merge Request***.

Pencapaian Kunci: Stabilitas *branch* `main` selalu terjaga karena adanya *branch protection* dan kewajiban *Code Review*. Hanya *Lead Developer* yang melakukan *squash merge* untuk menjaga histori `develop` tetap bersih.

H. Continuous Integration/Continuous Deployment (CI/CD)

Hasil: Waktu *Build* rata-rata di lingkungan CI adalah **4 menit 12 detik**.

Pencapaian Kunci: Mengurangi waktu yang dihabiskan *developer* untuk *deployment* manual dan memastikan bahwa kode yang di-*merge* ke `develop` telah melewati *linting* dan *unit testing* otomatis, sehingga mengurangi *bug* integrasi.

I. Pengujian dan Kualitas Perangkat Lunak

Pengujian adalah bagian integral dari SCM, memastikan hanya CI yang diverifikasi yang maju ke tahap berikutnya.

Jenis Pengujian	Metrik	Hasil
Unit Test	Code Coverage	85%
Integration Test	Tingkat Kegagalan CI	**0%** (Pada saat *merge* ke `main`)
System Test (UAT)	*Defects* Ditemukan	**5 Minor Defects** (Telah diperbaiki di *hotfix* V 1.0.1)

J. Kesulitan yang Dihadapi dan Solusi yang Diterapkan

Kesulitan	Solusi yang Diterapkan
Konflik *Merge*	Menerapkan *commit* yang lebih kecil dan sering (*small, frequent commits*) serta memaksa *developer* untuk secara rutin menarik (*pull*) perubahan terbaru dari `develop`.
Inkonsistensi Lingkungan	Menggunakan **Docker** untuk membuat *container* yang menjamin konsistensi antara lingkungan *development*, *staging*, dan *production* (CI-006).
Keterlambatan Dokumentasi	Menghubungkan *Issue* Jira dengan Dokumentasi Confluence, mewajibkan *developer* memperbarui dokumentasi *inline* saat mengembangkan fitur.

K. Kesimpulan dan Pembelajaran

Kesimpulan: Implementasi SCM dalam proyek CleanKicks telah berhasil memelihara integritas dan keterlacakan semua Item Konfigurasi. Proses *version control* yang ketat, dikombinasikan dengan CI/CD yang terotomasi, memungkinkan tim untuk mencapai rilis MVP (V 1.0.0) secara tepat waktu dengan kualitas kode yang tinggi (85% *Unit Test Coverage*).

Pembelajaran:

1. **Pentingnya *Branch Protection***: Ini adalah kunci untuk menjaga *branch* stabil (`main` dan `develop`).
2. **SCM = Disiplin Tim**: Keberhasilan SCM sangat bergantung pada kedisiplinan setiap anggota tim dalam mengikuti konvensi *commit* dan prosedur *Code Review*.

L. Pengelolaan Konfigurasi dan Audit

Aktivitas	Status	Detail
Baseline Konfigurasi	Ditetapkan	**BL-1.0.0** di-tag pada *branch* `main` setelah rilis. Baseline ini mencakup CI-001 hingga CI-006 pada status V 1.0.0.
Audit Konfigurasi	Berhasil	Audit internal terakhir mengkonfirmasi bahwa seluruh kode yang di-*deploy* ke *Production* (kode dari `main`) identik dengan kode yang di-*tag* sebagai BL-1.0.0. Tidak ditemukan *bug* regresi yang disebabkan oleh *mismatch* kode.
Lingkungan Rilis	Stabil	Konfigurasi lingkungan telah distandarisasi menggunakan *file* **Docker** dan *environment variables* yang dikelola.