**LAPORAN PRAKTIKUM**

**PEMROGRAMAN WEB**



Disusun Oleh :

Martonius Epta Sidabutar

NIM : 202312051

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TINGGI TEKNOLOGI BONTANG**

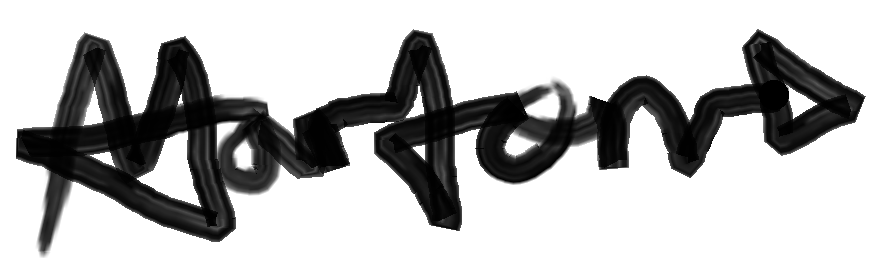
**TAHUN 2025**

# KATA PENGANTAR

Assalamu’alaikum Warahmatullahi Wabarakatuh. Puji dan syukur penulis panjatkan kehadirat Allah SWT, yang telah melimpahkan rahmat dan Hidayah-Nya lah sehingga laporan praktikum sistem basis data ini dapat terselesaikan, tetapi masih sangat jauh dari kesempurnaan.

Penulisan laporan praktikum ini dimaksudkan sebagai tolak ukur Saya dalam mencapai pemahaman beserta pembahasan mengenai Program dari praktikum Pemrograman Dasar kali ini, disamping itu penulis berharap dalam penulisan laporan praktikum ini dapat bermanfaat bagi penulis, pembaca, khususnya mahasiswa Sekolah Tinggi Teknologi Bontang dan masyarakat umumnya.

Bontang, 6 Juni 2025



Martonius Epta Sidabutar

# DAFTAR ISI

Daftar Isi

[KATA PENGANTAR ii](#_Toc204062087)

[DAFTAR ISI iii](#_Toc204062088)

[LEMBAR ASISTENSI iv](#_Toc204062089)

[Modul 1. Pemrograman WEB HTML 1](#_Toc204062090)

[A. Tujuan 1](#_Toc204062091)

[B. Alat dan Bahan 1](#_Toc204062092)

[C. Landasan Teori 1](#_Toc204062093)

[D. Kegiatan Praktikum 3](#_Toc204062094)

# LEMBAR ASISTENSI

**Modul 1. Git & GitHub**

## A. Tujuan Praktikum

1. Memahami konsep dasar dan perbedaan antara Git sebagai *Version Control System* dan GitHub sebagai platforn *hosting*.
2. Melakukan instalasi Git dan konfigurasi awal yang diperlukan pada sistem operasi Windows dan macOS.
3. Mengintegrasikan Git dengan Visual Studi Code (VS Code) dan menggunakan terminal terintegrasinya.
4. Menguasai alur kerja dasar Git: init, add, commit.
5. Memahami dan menerapkan penggunaan branch (cabang) untuk mengisolasi pekerjaan..
6. Menghubungkan repositori lokal ke repositori remote di GitHub.
7. Melakukan push (mendorong) perubahan dari repositori lokal ke GitHub.

## B. Teori Dasar

1. **Git vs GitHub: Apa Bedanya?**

Seringkali pemula bingung antara Git dan GitHub. Penting untuk memahami bahwa keduanya adalah dua hal yang berbeda namun saling melengkapi.

* Git: Anggap Git sebagai sebuah "mesin waktu" untuk kode Anda. Git adalah Distributed Version Control System (DVCS), sebuah perangkat lunak yang berjalan di komputer lokal Anda. Fungsinya adalah untuk melacak setiap perubahan yang terjadi pada file-file proyek Anda. Anda dapat "menyimpan" sebuah versi (disebut commit), dan jika terjadi kesalahan, Anda bisa kembali ke versi sebelumnya dengan mudah. Git tidak memerlukan koneksi internet untuk bekerja.
* GitHub: Anggap GitHub sebagai sebuah "garasi online" sekaligus "arena kolaborasi" untuk proyek-proyek Git Anda. GitHub adalah platform berbasis web yang menyediakan layanan hosting untuk repositori Git. Dengan GitHub, Anda bisa menyimpan salinan proyek Anda secara online (sebagai backup), berbagi kode dengan orang lain, dan bekerja sama dalam sebuah tim pada proyek yang sama.

Secara analogi: Jika proyek Anda adalah mobil, Git adalah mesin dan sistem mekanik yang membuatnya berjalan dan bisa diperbaiki. GitHub adalah garasi tempat Anda memarkir mobil, memamerkannya, dan tempat teman-teman Anda bisa ikut membantu memperbaikinya.

**2.** Setiap proyek Git di komputer Anda memiliki tiga “area” utama:

* **Working Directory**: Folder proyek tempat Anda membuat dan mengedit file secara langsung
* **Staging Area (Index):** Sebuah “ruang tunggu” atau area persiapan. Di sini Anda mendaftarkan perubahan-perubahan dari *Working DIrectory* yang siap untuk “disimpan” secara permanen dalam riwayat proyek.
* **Local Repository (.git directory):** “Buku catatan sejarah” proyek Anda. Ketika melakukan *commit*, perubahan dan *Staging* *Area* akan disimpan secara permanen di sini. Semua riwayat *commmit* tersimpan di dalam folder tersembunyi .git.

Alur kerjanya adalah sebagai berikut:



**3. Konssep Branching (Percabangan)**

Branch adalah salah sattu fitur terkuat di Git. Bayangkan riwayat proyek Anda adalah sebuah batang pohon utama (biasanya disebut main atau master). Ketika Anda ingin mengerjakan fitur baru atau mencoba sesuatu yang eksperimental, Anda bisa membuat branch (cabang) baru.

Anda bisa bekerja di cabang baru ini tanpa mengganggu kode yang stabil di cabang main. Setelah pekerjaan Anda di cabang baru selesai dan diuji, Anda bisa menggabungkannya (merge) kembali ke cabang main. Ini adalah praktik standar dalam pengembangan perangkat lunak untuk menghindari konflik dan menjaga stabilitas kode utama.

## C. Pra-Syarat: Work-Environment Setup

Sebelum memulai pratikum, pastikan Anda telah menyiapkan semua yang diperlukan.

1. **Membuat Akun Github**

Jika Anda belum memiliki akun, daftar terlebih dahulu

* **Langkah 1:** Buka browser dan kunjungi <https://github.com/>. Klik tombol **Sign Up.**
* **Langkah 2:** Ikuti semua proses pendaftaran: masukkan email, buat *password*, dan pilih *username*. Verivikasi akun Anda melalui email yang dikirimkan oleh GitHub.

1. **Instalasi Git**

* **Untuk Pengguna Windows:**
  + **Langkah 1:** Unduh installer Git dari <https://git-scm.com/download/win>.
  + **Langkah 2:** Jalankan installer. Pada sebagian besar langkah, Anda bisa menggunakan pengatruran *default* (cukup klik “Next”). Pastikan pada bagian “Choosing the default editor used by Git”, Anda memilih **Visual Studio Code** jika sudah terinstal
  + **Langkah 3:** Selesaikan instalasi
* Untuk Pengguna macOS:
  + **Langkah 1**: Buka aplikasi Terminal.
  + **Langkah 2**: Ketik git --version dan tekan Enter. Jika Git belum terinstal, macOS akan secara otomatis menawarkan untuk menginstal Xcode Command Line Tools. Klik Install dan ikuti prosesnya.
  + **Langkah 3**: (Alternatif): Anda juga bisa menginstal Git menggunakan [Homebrew](https://brew.sh/) (jika sudah terinstal) dengan perintah brew install git.

1. **Instalasi dan Konfigurasi Visual Studio Code (VS Code)**
   * **Langkah 1: Instal VS Code**
     + Unduh VS Code dari situs resminya: <https://code.visualstudio.com/>
     + Jalankan installer dan ikuti petunjuknya.
   * **Langkah 2: Instal Ekstensi yang Direkomendasikan**
     + Buka VS Code. Klik ikon Ekstensi di bilah samping kiri (terlihat seperti 4 kotak).
     + Cari dan instal ekstensi berikut:
2. **GitLens – Git supercharged:** Ekstensi ini sangat membantu untuk melihat riwayat file, siapa yang mengubah baris kode tertentu (blame), dan banyak lagi, langsung di dalam editor.
3. **GitHub Pull Request and Issues:** Memudahkan interaksi dengan GitHub langsung dari VS Code

* Langkah 3: **Konfigurasi Awal Git (Sangat Penting!)**
  + Buka terminal terintegrasi di VS Code dengan menekan Ctrl + `` (tombol backtick, di sebelah kiri angka 1) atau melalui menu View > Terminal`.
  + Di terminal, Anda perlu mengatur nama dan email Anda. **Ini hanya perlu dilakukan sekali**. Git akan menggunakan informasi ini untuk menandai setiap commit yang Anda buat. Ganti Nama Anda dan emailanda@contoh.com dengan data Anda sendiri. Lakukan satu per satu

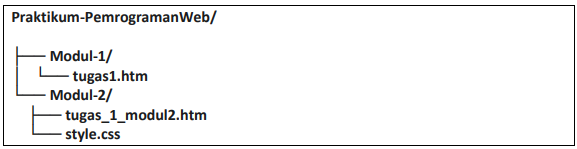


## D. Langkah-langkah Praktikum

**Skenario:** Anda telah menyelesaikan tugas dari **Modul 1 (HTML Dasar)** dan **Modul 2 (Pengenalan CSS)**. Sekarang kita akan menyimpan dan mengelola kedua proyek ini menggunakan Git dan GitHub

**Asumsi Struktur Folder:**

Untuk latihan ini, mari kita asumsikan Anda memiliki struktur folder proyek seperti ini di suatu tempat di komputer Anda:



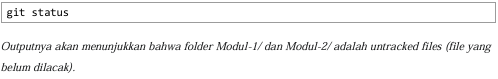
1. **Membuka Proyek dan Menginisiasi Git**

* **Langkah 1:** Buka VS Code, lalu klik File > Open Folder .. dan pilih folder induk proyek Anda, yaitu Praktikum-PemrogramanWeb.
* **Langkah 2:** Buka terminal terintegrasi (Ctrl+``). Pastikan Anda berada di direktori yang benar (Praktikum-PemrogramanWeb).
* **Langkah 3:** ketik perintah berikut untuk menginisiasi Git di folder ini.

****

**2. Perintah Dasar Bash dan Alur Kerja Commit Pertama**

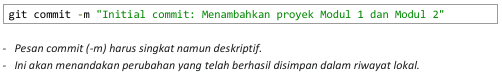
* **Langkah 1: Memeriksa Status (git status)**
  + Perintah ini adalah teman terbaik Anda. Gunakan untuk melihat status proyek Anda saat ini.

****

* **Langkah 2: Menambahkan File ke *Staging Area* (git add)**
  + Kita akan menambahkan semua file dan folder dalam proyek ini ke *stagging area*.

****

* **Langkah 3: Melakukan *Commit* (git commit)**
  + Sekarang simpan perubahan dari *stagging area* ke riwayat repositori lokal Anda dengan pesan yang jelas.

****

**3. Membuat Repositori di GitHub dan Menghubungkannya**

* **Langkah 1:** Buka GitHub di browser. Klik ikon + di pojok kanan atas, lalu pilih **New Repository**
* **Langkah 2:** Isi detailnya:
* **Langkah 3:** Klik **Create Repository.**
* **Langkah 4:** GitHub akan menampilkan halaman dengan beberapa perintah.

Salin URL HTTPS dan repositori Anda. **–** *URL ini adalah alamat dari repositori remote Anda*

* **Langkah 5**: Kembali ke terminal VS Code dan hubungkan repositori lokal Anda ke *remote* di GitHub dengan perintah berikut(ganti URL dengan milik Anda):

|  |
| --- |
| *git remote add origin https://github.com/usernameAnda/portofolio-pemweb.git* |

**4. Praktik Terbaik: Bekerja dengan Branch**

Sebelum kita mendorong kode kita, mari kita praktikkan alur kerja yang benar dengan membuat branch baru. Jangan bekerja langsung di branch main.

* **Langkah 1: Membuat *Branch* Baru**
  + Buat *branch* baru dengan nama yang deskriptif. Misalnya, kita akan menamainya tugas/modul-1-dan-2

|  |
| --- |
| git branch tugas/modul-1-dan-2 |

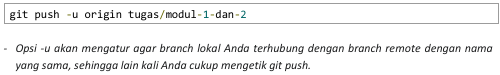
* **Langkah 2: Pindah ke *Branch* Baru**
  + Sekarang, pindah dari main ke *branch* baru tersebut.

|  |
| --- |
| git checkout tugas/modul-1-dan-2 |

Anda juga bisa melakukan kedua langkah di atas dengan satu perintah:

|  |
| --- |
| git checkout -b tugas/modul-1-dan-2 |

* **Langkah 3: Mendorong (*Push) Branch* Baru ke GitHub**
  + Sekarang, unggah *branch* baru ini berserta semua *commit*-nya ke GitHub.



**5. Pentingnya tidak melakukan push langsung ke main**

Dalam proyek tim, mendorong kode langsung ke branch main sangat berbahaya. Ini bisa memasukkan bug atau kode yang belum selesai ke dalam basis kode utama yang digunakan semua orang. Alur kerja yang benar adalah:

1. Bekerja di *branch* terpisah(seperti yang kita lakukan).
2. Setelah selesai, buat *Pull Request* di GitHub.
3. *Pull Request* adalah permintaan untuk menggabungkan kode dari *branch* Anda ke *branch* main. Ini memberi kesempatan kepada anggota tim lain untuk meninjau (*review*)kode Anda sebelum digabungkan

Untuk proyek pribadi seperti ini, Anda bisa saja langsung mendorong ke main, tetapi membiasakn diri dengan alur kerja *branching* adalah investasi yang sangat berharga.

## E. Tugas Praktikum

