

Homework

Introduction to Software Engineering

Nama: Nahla Putri Gunawan

Kelas: IT Full Stack Developer

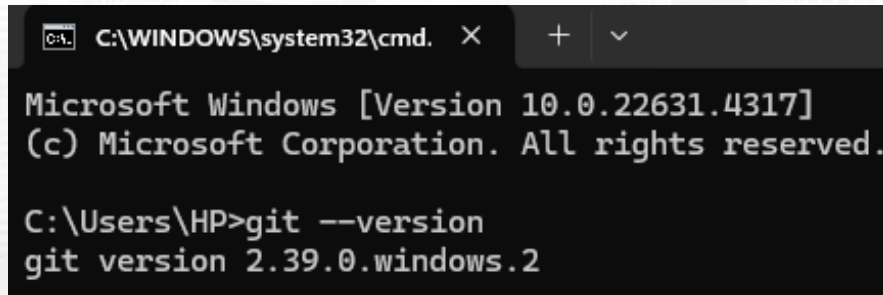
Batch: SIB7



1. Soal & Instruksi - Instalasi Tools

Silahkan teman teman melakukan proses instalasi beberapa tools dibawah ini :

- Git



```
C:\WINDOWS\system32\cmd. X + v
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>git --version
git version 2.39.0.windows.2
```



- Visual Studio Code

```
C:\Users\HP>code --version  
1.94.2  
384ff7382de624fb94dbaf6da11977bba1ecd427  
x64
```



Visual Studio Code
App

- Browser

```
C:\Users\HP>reg query "HKEY_CURRENT_USER\Software\Google\Chrome\BLBeacon" /v version

HKEY_CURRENT_USER\Software\Google\Chrome\BLBeacon
    version    REG_SZ      129.0.6668.101
```



Google Chrome
App

Setelah melakukan instalasi, buat sebuah summary untuk apa yang sudah dipelajari dalam materi pengenalan software engineer sebelumnya pada google docs.

1. Full Stack Developer Career Path

- **Introduction Full Stack Web/Mobile Developer**

Pengembangan Fullstack mengacu pada pengembangan aplikasi secara end-to-end, meliputi pengembangan frontend, backend, dan kadang-kadang juga sisi klien.

Front-end development berfokus pada tampilan dan interaksi pengguna pada aplikasi. **Back-end development** menangani logika dan proses di balik layar aplikasi. **Database management** mengelola data yang digunakan oleh aplikasi. **Integrasi Front-end dan Back-end** menghubungkan bagian tampilan (front-end) dengan bagian logika (back-end) melalui API untuk berkomunikasi dengan server dan database. **Version Control and Collaboration** menggunakan sistem pengendalian versi seperti Git untuk mengelola perubahan kode dan kolaborasi dalam tim pengembang. **Mobile Development** mengembangkan aplikasi mobile menggunakan framework React Native dan Flutter.

Pengembangan front-end adalah proses membuat tampilan dan interaksi pada website atau aplikasi web. Bahasa pemrograman yang digunakan dalam front-end adalah **HTML** untuk menentukan struktur konten web. **CSS** untuk menata tampilan visual web. **JavaScript** untuk membuat web menjadi interaktif.

Backend development adalah bagian dari pengembangan web yang berfokus pada apa yang terjadi di balik layar. Bagian ini bertanggung jawab untuk menerima permintaan, mengolah data dan memberikan respons. **Bahasa pemrograman:** Node.js, Python, Ruby, Java, PHP, dll. **Framework:** Express.js, Flask, Ruby on Rails, Spring, Laravel, dll. **Database:** MySQL, PostgreSQL, MongoDB, Firebase, dll.

- **Skillset Full Stack Web/Mobile Developer**

Pengembangan Aplikasi End-to-End adalah sebuah pendekatan dalam membuat aplikasi yang mencakup seluruh proses pembuatan aplikasi, mulai dari tahap perencanaan awal hingga aplikasi tersebut siap digunakan oleh pengguna. Dengan kata lain, pengembangan ini melibatkan semua tahapan yang diperlukan untuk menghasilkan sebuah produk aplikasi yang utuh dan berfungsi dengan baik.

Tahap-Tahap Pengembangan Aplikasi End-To-End

- Perencanaan dan Analisis
- Desain
- Pengembangan Front-End
- Pengembangan Back-End
- Integrasi dan Pengujian
- Pemeliharaan dan Peningkatan

Version control adalah sistem yang digunakan untuk melacak perubahan pada kode sumber suatu proyek. Sistem ini sangat berguna untuk tim pengembang karena memungkinkan mereka untuk bekerja sama secara efisien dan efektif. Manfaat menggunakan version control:

- Tim dapat bekerja secara bersamaan tanpa saling mengganggu.
- Setiap perubahan pada kode akan tercatat dengan jelas, sehingga mudah untuk melacak siapa yang membuat perubahan dan kapan.
- Jika terjadi kesalahan, tim dapat dengan mudah mengembalikan kode ke versi sebelumnya.
- Version control membantu menjaga kode tetap terstruktur dan mudah dikelola.

- **Tools Full Stack Web/Mobile Developer**

Sebagai seorang Full Stack Developer, Anda membutuhkan berbagai macam alat dan teknologi untuk membangun dan mengelola aplikasi secara efisien. Berikut tools penting yang mungkin digunakan oleh pengembang Full Stack:

- IDE - Code Editor: Visual Studio Code
- Version Control - Repository: GitHub, GitLab, Bitbucket
- Version Control - Git Tools: Sourcetree, GitLens
- DBMS: PostgreSQL, MySQL, Oracle, MongoDB, Redis
- API: Postman, Swagger
- Tests dan Debugging: Jest, Mocha, Chai, Junit 5
- Mobile Development: React Native, Flutter
- Layanan Cloud: AWS, Google Cloud, Azure
- CI/CD: Jenkins, Circleci
- Desain UI/UX: Figma, Sketch

2. SDLC & Design Thinking Implementation

- **What is SDLC**

SDLC (Siklus Hidup Pengembangan Perangkat Lunak) adalah rangkaian proses terstruktur yang digunakan untuk mengembangkan perangkat lunak dari awal hingga akhir. Terdiri dari enam fase yang saling terkait dan dilakukan secara berurutan, yaitu: 1) Perencanaan dan Analisis, 2) Desain, 3) Pengembangan, 4) Pengujian, 5) Penerapan, dan 6) Pemeliharaan. SDLC memastikan bahwa pengembangan perangkat lunak berjalan dengan baik dan sesuai dengan kebutuhan serta tujuan yang ditentukan.

Penggunaan Software Development Life Cycle (SDLC) memiliki berbagai manfaat, antara lain meningkatkan prediktabilitas dan pengendalian proyek, meningkatkan kualitas perangkat lunak, serta efisiensi tim dan kolaborasi. SDLC juga memperbaiki dokumentasi, memastikan pemenuhan kebutuhan pengguna, serta menghemat biaya dan waktu. Selain itu, SDLC memungkinkan pengelolaan risiko yang lebih baik dan meningkatkan pengawasan serta evaluasi selama proses pengembangan.

- **Model-Model Software Development Life Cycle (SDLC)**

Berikut adalah penjelasan singkat tentang beberapa model SDLC:

- **Waterfall Model:** Model linier dan berurutan dengan tahap-tahap yang harus diselesaikan secara berurutan. Cocok untuk proyek dengan persyaratan stabil.
- **V-Shaped Model:** Mirip dengan Waterfall, tetapi menekankan pengujian pada setiap tahap pengembangan. Cocok untuk proyek yang memprioritaskan kualitas.
- **Prototype Model:** Fokus pada pembuatan prototipe untuk memahami kebutuhan pengguna sebelum pengembangan final.
- **Spiral Model:** Menggabungkan elemen spiral dan inkremental, cocok untuk proyek besar dan kompleks dengan banyak risiko.
- **Incremental Model:** Mengembangkan perangkat lunak dalam iterasi kecil, menambahkan fitur hingga produk akhir siap.
- **Big Bang Model:** Model tidak terstruktur yang dimulai tanpa perencanaan mendetail, cocok untuk proyek kecil.
- **Agile Model:** Pendekatan kolaboratif dan iteratif yang berfokus pada pengiriman perangkat lunak secara berkala dan responsif terhadap perubahan.

- **Design Thinking Implementation**

Tahapan Design Thinking dalam pengembangan perangkat lunak dimulai dengan **Empathize**, memahami kebutuhan dan masalah pengguna. Selanjutnya, pada **Define**, informasi dianalisis untuk menentukan masalah dan tujuan proyek. Dalam fase **Ideate**, ide-ide kreatif dihasilkan, diikuti oleh **Prototype**, di mana representasi nyata dibuat untuk mengumpulkan umpan balik. Tahap **Test** melibatkan pengumpulan umpan balik dari pengguna untuk memvalidasi solusi, dan pada **Implement**, desain diterjemahkan ke dalam kode. Integrasi Design Thinking dalam SDLC membantu tim menciptakan produk yang lebih berorientasi pada pengguna dan responsif terhadap perubahan kebutuhan.

3. Basic Git & Collaborating Using Git

- **Terminal and IDE**

Terminal adalah alat penting yang tetap relevan meskipun teknologi dan perangkat lunak terus berkembang. Walaupun antarmuka grafis menjadi lebih canggih dan populer, terminal tetap digunakan oleh pengembang perangkat lunak, administrator sistem, dan pengguna teknis lainnya. Keunggulan terminal terletak pada fleksibilitas dan kekuatannya untuk menjalankan tugas-tugas khusus serta otomatisasi dalam lingkungan komputer modern.

Sejarah terminal dimulai pada 1960-an sebagai antarmuka teks untuk komputer mainframe. Pada 1970-an, terminal DEC VT100 dan UNIX memperkenalkan kontrol dan fleksibilitas yang lebih baik. Di 1980-an, emulator terminal menggantikan perangkat fisik, meski GUI semakin populer. Hingga kini, terminal tetap relevan bagi pengembang dan administrator untuk tugas otomatisasi dan sistem.

- **Installing, initializing and committing GIT**

Git adalah sistem kontrol versi terdistribusi yang membantu pengembang melacak perubahan kode, berkolaborasi dengan tim, dan mengelola revisi secara efektif. Topik ini mencakup cara menginstal Git pada berbagai sistem operasi, menginisialisasi repositori baru, dan melakukan commit pertama.

Dasar-Dasar Command Git:

- `git init`: Menginisialisasi direktori sebagai repositori Git kosong
- `git clone`: Menduplikasi repositori Git yang sudah ada ke direktori lokal.
- `git status`: Menampilkan status perubahan yang belum dikomit di repositori lokal.
- `git add`: Menambahkan perubahan ke area persiapan (staging area) untuk disiapkan menjadi commit
- `git commit`: Membuat commit dari perubahan yang sudah di-staging dan menambahkan pesan commit
- `git push`: Mengirimkan commit ke repositori jarak jauh (remote repository).
- `git pull`: Mengambil commit terbaru dari repositori jarak jauh dan menggabungkannya ke repositori lokal.
- `git branch`: Menampilkan daftar cabang (branch) yang ada di repositori dan menunjukkan cabang aktif.
- `git checkout`: Beralih ke cabang lain atau ke commit tertentu.
- `git merge`: Menggabungkan perubahan dari satu cabang ke cabang aktif.
- `git log`: Menampilkan daftar commit beserta riwayatnya dalam repositori.
- `git remote`: Menampilkan daftar repositori jarak jauh yang terhubung dengan repositori lokal.
- `git fetch`: Mengambil informasi terbaru dari repositori jarak jauh tanpa menggabungkan perubahan.
- `git diff`: Menampilkan perbedaan antara versi yang sudah di-staging dengan versi sebelumnya.
- `git reset`: Mengembalikan file yang sudah di-staging ke direktori kerja sebelumnya.

2. Soal & Instruksi - Collaborating Using Git

Buatlah repositori GitHub baru dengan nama "finpro-msib-7-kelompok X"

Contoh : finpro msib 7 kelompok 1

**1 kelompok hanya membuat 1 repositori saja*

- Undang teman-teman kelompokmu sebagai kolaborator pada repositori tersebut
- Setiap peserta meng-*clone* repositori ke mesin lokal masing-masing
- Setiap peserta harus membuat cabang untuk pekerjaannya. Misal : doni-homework, nita-homework, etc
- Masukkan Homework dari minggu sebelumnya ke branch lokalmu.
- Setelah melakukan perubahan, teman-teman harus melakukan commit dengan pesan commit yang deskriptif
- Lakukan push and pull ke branch homework dari branch lokalmu.
- Pastikan konflik sudah terselesaikan sebelum melakukan pull request

Setelah selesai, bisa teman teman zip hasil soal pertama dan kedua, lalu upload ke dalam LMS Rakamin.

Terima kasih!