1. Definisi Full Stack Development

Pengembangan Full Stuck (Full Stack Development) merujuk pada pengembangan seluruh aplikasi secara ***end-to-end,*** dari sisi depan ***(front-end)*** hingga sisi belakang ***(back-end)*** dan, dalam beberapa kasus, hingga sisi klien (client-side).

· F**ront End Development**

Melakukan pengembangan sisi klien sehingga pengguna dapat melihat dan berinteraksi dengannya secara langsung.

1. **HTML** (HyperText Markup Language) adalah blokbangunan paling dasar dari Web. Ini mendefinisikan arti dan struktur konten web.

2. **CSS** (Cascading Style Sheet) digunakan untuk menata halaman Web agar menarik.

3. **Javascript** adalah Bahasa pemrograman paling populer di dunia sebagai Bahasa pemrograman Web. JS digunakan untuk membuat web lebih interaktif.

Framework yang digunakan di Frontend React, Vue.js, Angular.js.

· **Back End Development**

Bertanggung jawab untuk memproses permintaan dari pengguna, mengelola dan menyimpan data di database, serta memberikan respons kepada klien (front-end) berdasarkan permintaan yang diterima.

1. **Bahasa Pemrograman Server-Side** : Node.js (JavaScript), Python, Ruby, Java, PHP, C#.

2. **Server Framework** : Express.js untuk Node.js, Flask untuk Python, Ruby on Rails, Spring untuk Java, dan Laravel untuk PHP.

3. **Database Management** : SQL (MYSQL, PostgreSQL, SQL Server) dan NoSQL (MongoDB, Firebase).

· **Database Management**

Bagian kritis dari aplikasi karena menyimpan dan mengorganisir informasi yang diperlukan untuk menjalankan aplikasi dengan benar. Atau serangkaian konsep dan Teknik yang digunakan untuk mengelola data dalam sebuah aplikasi atau system.

1. **Database Management System** merupakan perangkat lunak yang memungkinkan pengguna untuk mengelola dan mengakses data dalam database. DBMS menyediakan antarmuka untuk berinteraksi dengan database.

2. Tipe Database : ada dua pertama database relasional SQL (Structured Query Language) kedua database non-relasional NoSQL (Not Only SQL)

3. Bahasa Query : SQL adalah Bahasa query yang digunakan untuk berinteraksi dengan database SQL. Bahasa query memungkinkan pengguna untuk melakukan operasi seperti SELECT, INSERT, UPDATE DELETE.

4. Database Management : SQL (ORACLE, PostgreSQL, MYSQL), NoSQL (mongoDB, Redis).

· **Mobile Development**

Serangkaian konsep dan teknologi yang digunakan untuk mwmbangun aplikasi yang dapat digunakan untuk membangun aplikasi yang dapat dijalankan di perangkat mobile, seperti smartphone dan tablet.

1. **Platform Mobile :** aplikasi mobile dapat dikembangkan untuk berbagai platform, termasuk Android, iOS, dan Windows Phone. Android menggunakan Bahasa Java atau Kotlin, iOS menggunakan Swift atau Objective-C.

2. **IDE (Integrated Development Environment)** adalah perangkat lunak yang digunakan untuk mengembangkan aplikasi mobile. IDE menyediakan alat bantu, penyunting kode, pengelola proyek, simulator perangkat, dan fasilitas debugging untuk mempermudah proses pengembangan.

3. **Framework** yang digunakan React Native, Flutter.

Pengembangan Aplikasi End to End

Tahap-tahap pengembangan aplikasi end-to-end

1. Perencanaan dan Analisis, mengumpulkan kebutuhan dan pemahaman tentang tujuan aplikasi, sasaran pengguna, serta lingkungan operasional untuk mengidentifikasi fitur utama yang diperlukan dalam aplikasi.

2. Desain, merancang **User Interface (UI)** dan **User Experience (UX)** yang intuitif dan menarik. Serta merencanakan arsitektur aplikasi, termasuk pemilihan teknologi, database, dan framework yang sesuai.

3. Pengembangan Front-End, membuat tampilan dan interaksi yang menarik bagi pengguna.

4. Pengembangan Back-End, mengembangkan sisi server dan logika bisnis aplikasi .

5. Integrasi dan Pengujian, dilakukan untuk memastikan semua fitur berfungsi dengan benar, mengidentifikasi dan memperbaiki bug yang mungkin ada. Bagian depan dan belakang aplikasi harus diintegrasikan melalui API (Application Programming Interface) sehingga mereka dapat berkomunikasi dan berbagi data.

6. Pemeliharaan dan Peningkatan, setelah diluncurkan, aplikasi harus dipelihara dengan memperbaiki bug dan menangani perubahan lingkungan atau kebutuhan. Peningkatan terus menerus dilakukan untuk memperbarui fitur, meningkatkan kinerja, dan memastikan aplikasi tetap relevan dalam waktu yang berlanjut.

Kolaborasi Efektif

Version Control (pengendali versi) adalah sistem yang memungkinkan pengembang perangkat lunak untuk melacak perubahan pada kode sumber aplikasi selama pengembangan. Sehingga kolaborasi efisien di antara anggota tim. Popular version control, yaitu git, mercurial.

Manfaat version control:

1. Rekam perubahan
2. Pencatatan Riwayat

3. Pemecahan konflik

4. Pemulihan mudah

Penggunaan version control untuk berkolaborasi:

1. Inisialisasi proyek, tim membuat repository version control untuk menyimpan semua kode sumber, file, dan perubahan yang dilakukan selama pengembangan.

2. Pengembangan parallel, anggota memiliki salinan repository sehingga dapat bekerja secara parallel, membuat perubahan.

3. Branching, version control memungkinkan pembuatan cabang (branch) yang terpisah dari kode utama. Memungkinkan tim untuk mengisolasi perubahan dan fitur yang sedang dikembangan.

4. Merge, setelah fitur atau perubahan selesai, cabang dapat digabungkan Kembali ke cabang utama.

5. Pull request, di beberapa platform version control seperti GitHub, GitLab, dan Bitbucket, pull request adalah mekanisme yang memungkinkan pengembanga untuk mengajukan perubahan mereka untuk ditinjau oleh anggota tim lain sebelum digabungkan ke cabang utama.

Tools Sets sebagai Full Stack Developer

1. **IDE - Code Editor (Visual Studio Code)**
2. **Version Control - Repository (GitHub, GitLab, Bitbucket)**
3. **Version Control – Git Tools (Sourcetree, GitLens)**
4. **DBMS (PostgreSQL, MYSQL, ORACLE, mongoDB, redis)**
5. **API (POSTMAN, swagger)**
6. **Test dan Debugging (Jest, mocha chai, Junit)**
7. **Mobile Development (React Native, Flutter)**
8. **Layanan Cloud (AWS, Google Cloud, Azure)**
9. **CI/CD (Jenkins, circleci)**
10. **Desain UI/UX (Figma, Sketch) .**

SDLC (Siklus Hidup Pengembangan Perangkat Lunak) adalah rangkaian proses yang terstruktur dan metodologi yang digunakan untuk mengembangan perangkat lunak dari awal hingga selesai.

Siklus SDLC

1. Perencanaan dan Analisis

Tahap pertama melibatkan identifikasi masalah atau kebutuhan bisnis yang perlu diselesaikan oleh perangkat lunak. para pemangku kepentingan berinteraksi untuk mengumpulkan persyaratan dan menentukan ruang lingkup proyek. Rencana keseluruhan untuk proyek perangkat lunak dibuat. Rencana ini mencakup alokasi sumber daya, jadwal waktu, dan definisi tugas dan tanggung jawab anggota tim.

2. Desain Produk

   Di tahap ini, perangkat lunak dirancang secara rinci berdasarkan persyaratan yang telah dikumpulkan. Desain mencakup **arsitektur sistem**, **antarmuka pengguna** dan **desain database.**

3. Pengembangan Produk

   Tahap ini melibatkan implementasi rancangan perangkat lunak yang telah disetujui sebelumnya. Para pengembang menulis kode untuk menghasilkan produk perangkat lunak yang berfungsi.

4. Pengujian Produk

   Setelah perangkat lunak dikembangkan, tahap pengujian dilakukan untuk memastikan bahwa perangkat lunak berfungsi sesuai dengan persyaratan yang telah ditentukan. Pengujian mencakup verifikasi fungsionalitas, kinerja, keamanan, dan kualitas keseluruhan perangkat lunak.

5. Penerapan Produk

   Tahap ini melibatkan implementasi rancangan perangkat lunak yang telah disetujui sebelumnya. Para pengembang menulis kode untuk menghasilkan produk perangkat lunak yang berfungsi.

6. Pemeliharaan Produk

   Setelah perangkat lunak diimplementasikan, pemeliharaan dilakukan untuk memperbaiki bug, meningkatkan fitur, dan menjaga perangkat lunak agar tetap sesuai dengan perubahan kebutuhan bisnis.

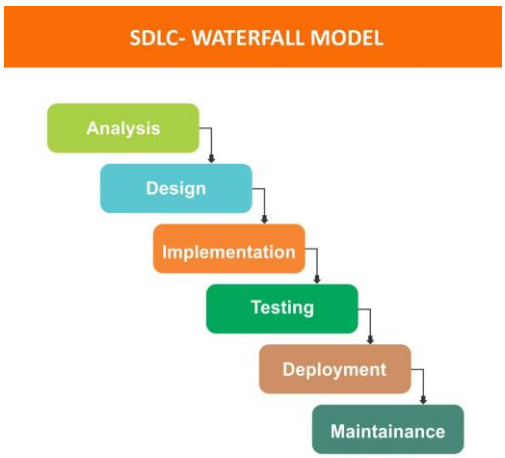
Manfaat Penggunaan SDLC

1. Prediktabilitas dan Pengendalian Proyek
2. Peningkatan Kualitas Perangkat Lunak
3. Pengelolaan Risiko yang Lebih Baik
4. Efisiensi Tim dan Kolaborasi
5. Memenuhi Kebutuhan Pengguna
6. Penghematan Biaya dan Waktu
7. Meningkatkan Pengawasan dan Evaluasi
8. Peningkatan Dokumentasi

Model Model SDLC

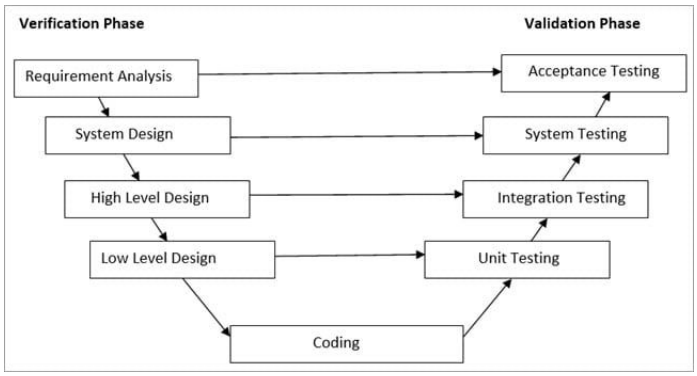
1. **Waterfall Model**

Waterfall model adalah model SDLC yang linier dan berurutan. Setiap tahap dalam model ini harus selesai sebelum memulai tahap berikutnya. Tahapannya meliputi analisis, perencanaan, desain, pengembangan, pengujian, implementasi, dan pemeliharaan. Cocok untuk proyek dengan persyaratan yang jelas dan stabil.



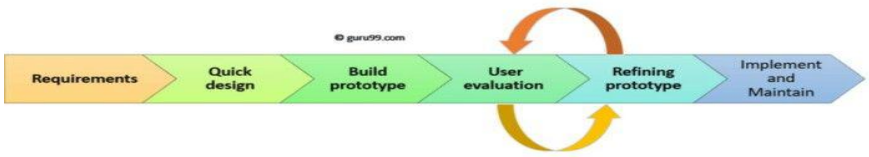
2. V-Shaped Model

Model V-Shaped adalah model yang terkait erat dengan model waterfall, tetapi menekankan pada pengujian. Tahapan pengujian diwakili oleh garis miring "V", yang berarti bahwa setiap tahap pengembangan memiliki tahapan pengujian yang sesuai. Cocok untuk proyek dengan fokus pada kualitas tinggi.



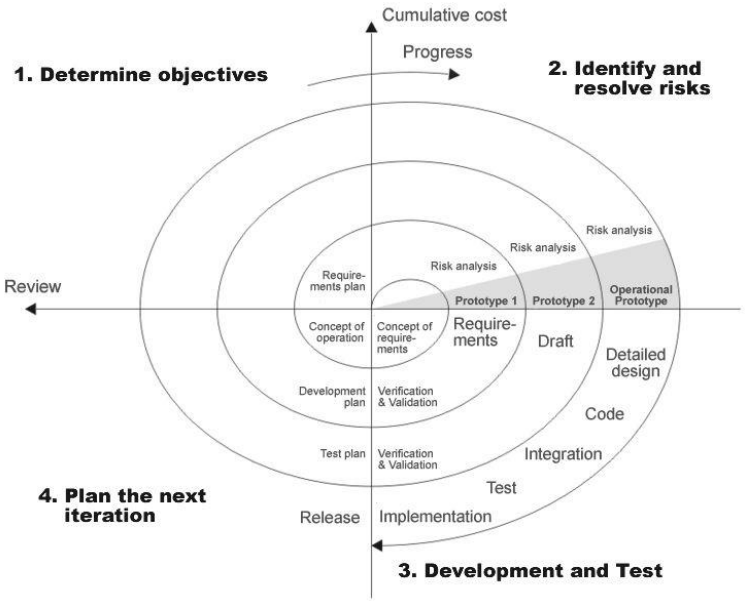
3. Prototype Model

Model Prototype adalah model pengembangan perangkat lunak yang bertujuan untuk menciptakan prototipe atau contoh awal sebelum mengembangkan versi finalnya. Model ini fokus pada pemahaman kebutuhan pengguna dan mengumpulkan umpan balik untuk memastikan bahwa perangkat lunak akhir sesuai dengan ekspektasi dan persyaratan pengguna.



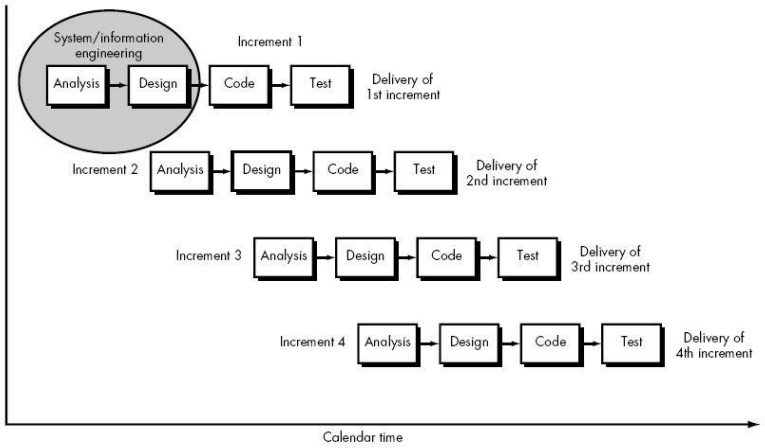
4. Spiral Model

Model ini menggabungkan elemen model spiral dengan pendekatan inkremental. Setiap siklus spiral membangun pada inkrementasi sebelumnya, menghasilkan perangkat lunak yang semakin berkembang dengan fitur yang lebih banyak setiap siklusnya. Cocok untuk proyek besar dan kompleks dengan banyak risiko.



5. Iterative Incremental Model

Model ini melibatkan pengulangan siklus pembangunan dan peningkatan perangkat lunak dalam tahapan-tahapan kecil. Setiap iterasi menambahkan lebih banyak fitur hingga produk akhir mencapai tingkat kesempurnaan yang diinginkan. Cocok untuk proyek dengan waktu dan anggaran yang terbatas.



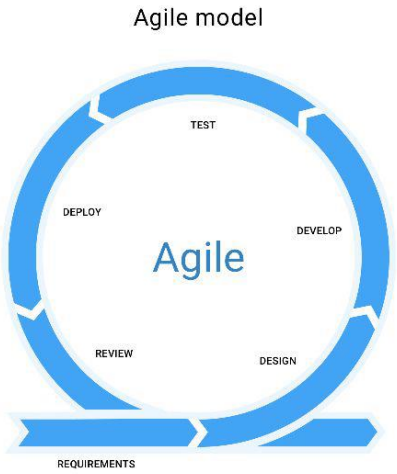
6. Big Bang Model

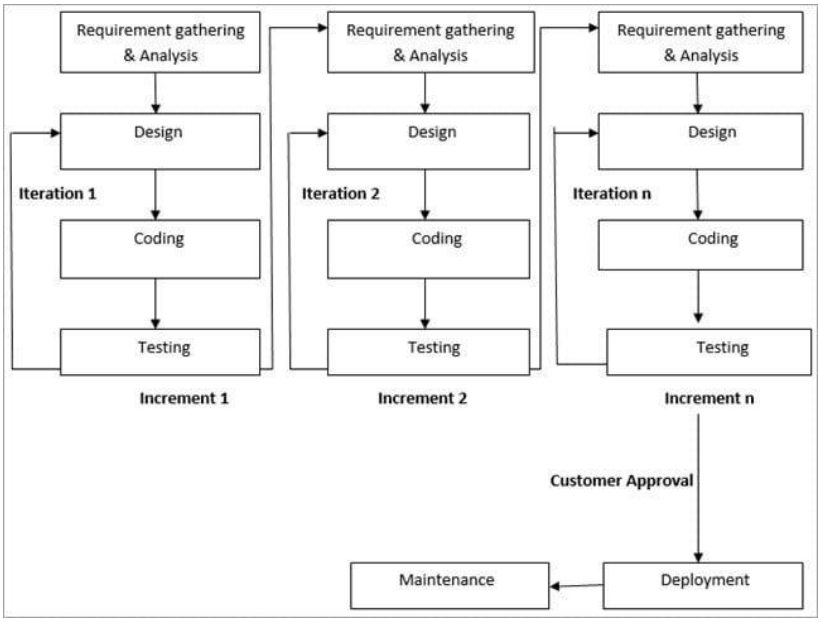
Model Big Bang adalah model yang kurang terstruktur, di mana semua tahapan pengembangan dilakukan tanpa perencanaan yang detail. Pengembangan dimulai tanpa

melakukan analisis dan perencanaan yang mendalam. Cocok untuk proyek kecil atau prototyping.

7. Agile Model

Model Agile adalah pendekatan kolaboratif dan iteratif yang berfokus pada pengiriman perangkat lunak secara berkala dan inkremental. Tim bekerja dalam sprint (iterasi singkat) dan selalu terbuka untuk perubahan persyaratan pengguna. Cocok untuk proyek dengan lingkungan yang dinamis dan persyaratan yang berubah-ubah.





Steps Design Thinking

1. **Empathize : Understand User Needs**

   Pada tahap ini, fokus pada memahami secara mendalam pengguna akhir dan kebutuhan mereka, keinginan, serta masalah yang dihadapi. Anda dapat melakukan berbagai kegiatan, seperti

   User Research

   Lakukan wawancara, observasi, dan survei untuk mengumpulkan data kualitatif dan kuantitatif tentang perilaku dan preferensi pengguna.

   Empathy Mapping

   Buat pemetaan empati yang menggambarkan sikap, pemikiran, perasaan, dan masalah pengguna.

   User Personas

   Buat persona pengguna fiktif yang mewakili berbagai kelompok pengguna.

2. **Define : Define the Problem**

   Pada tahap ini, informasi yang dikumpulkan selama fase empati dianalisis untuk menentukan masalah dan menetapkan tujuan yang jelas untuk proyek. Kegiatan kunci meliputi:

   Problem Statement

   Susun pernyataan masalah yang jelas dan ringkas yang mengartikulasikan tantangan dari sudut pandang pengguna.

Stakeholder Alignment

Kolaborasi dengan pemangku kepentingan, termasuk pemilik produk, manajer proyek, desainer, dan pengembang, untuk memastikan semua orang sejalan dengan tujuan dan ruang lingkup proyek.

3. **Ideate : Generate Ideas**

Fase ideasi mendorong pemikiran kreatif dan menghasilkan berbagai solusi potensial. Kegiatan kunci meliputi:

Brainstorming Sessions

Lakukan sesi brainstorming kolaboratif dengan tim lintas fungsi untuk menghasilkan banyak ide tanpa menghakimi.

Idea Consolidation

Setelah sesi brainstorming, kelompokkan dan konsolidasikan ide-ide terkait. Saring daftar hingga sekumpulan solusi yang dapat diwujudkan dan sesuai dengan tujuan dan batasan proyek.

4. **Prototype : Build and Iterative Solutions**

Pada tahap ini, fokus pada menciptakan representasi nyata dari ide-ide yang dipilih. Prototype digunakan untuk mengumpulkan umpan balik dan memvalidasi asumsi. Kegiatan kunci meliputi:

Low-Fidelity Prototypes

Bangun prototipe rendah seperti sketsa kertas, wireframe, atau mock-up. Prototipe ini cepat dan mudah dibuat, memungkinkan iterasi yang cepat.

High-Fidelity Prototypes

Kembangkan prototipe yang lebih detail atau bahkan Minimum Viable Product (MVP) yang menyerupai penampilan dan fungsionalitas produk akhir.

5. **Test : Gather User Feedback**

Tahap pengujian melibatkan pengumpulan umpan balik dari pengguna nyata untuk memvalidasi solusi-solusi tersebut. Kegiatan kunci meliputi:

Usability Testing

Lakukan sesi satu lawan satu dengan pengguna untuk mengamati interaksi mereka dengan prototipe.

Iterative Testing

Gunakan umpan balik dari pengujian kebergunaan untuk berinteraksi pada desain dan mengatasi masalah ketergunaan atau kekhawatiran.

6. **Implement : Develop the Software**

   Pada tahap ini, desain diterjemahkan ke dalam kode yang sebenarnya dan diimplementasikan. Kegiatan kunci meliputi:

   Agile Development

   Manfaatkan metodologi pengembangan agile seperti Scrum atau Kanban untuk memungkinkan pengembangan secara bertahap dan pengiriman berkelanjutan.

   Cross-Functional Collaboration

   Tingkatkan kolaborasi antara desainer, pengembang, penguji, dan pemangku kepentingan lainnya untuk memastikan implementasi yang selaras dengan solusi yang telah dirancang.

Terminal and IDE

Command	Windows	Linux / macOS	Description
List Files	`dir`	`ls`	List files and directories in the current folder
Change Directory	`cd`	`cd`	Change current working directory
Make Directory	`mkdir`	`mkdir`	Create a new directory
Remove Directory	`rmdir`	`rm -r`	Remove a directory
Delete Files	`del`	`rm`	Delete files
Copy Files	`copy`	`cp`	Copy files
Move / Rename	`move`	`mv`	Move or rename files/directories
Display File	`type`	`cat`	Display file content
Display Text	`echo`	`echo`	Display text or enable/disable echoing of commands

Display File	`type`	`cat`	Display file content
Display Text	`echo`	`echo`	Display text or enable/disable echoing of commands
List Processes	`tasklist`	`ps`	List running processes
Terminate Process	`taskkill`	`kill`	Terminate processes

Print Directory	N/A	`pwd`	Print the current working directory
Create File	N/A	`touch`	Create an empty file or update timestamp
Change Permissions	N/A	`chmod`	Change file permissions
Change Ownership	N/A	`chown`	Change file ownership
Search in Files	N/A	`grep`	Search for patterns in files
Display Manual	N/A	`man`	Display the manual page for a command
Execute as Admin	N/A	`sudo`	Execute a command with administrative privileges
Ping	N/A	`ping`	Send ICMP echo requests to check network connectivity
Network Config	`ipconfig`	`ifconfig`	Display network interface configurations
Secure Shell	N/A	`ssh`	Connect to a remote host using SSH

Sistem Kontrol Versi Terpusat (Centralized Version Control System)

Dalam sistem kontrol versi terpusat, ada satu repositori sentral yang berfungsi sebagai "master" untuk menyimpan seluruh sejarah proyek. Setiap pengembang melakukan perubahan pada salinan lokal, kemudian mengirimkan perubahan tersebut ke repositori sentral. Contoh sistem kontrol versi terpusat adalah Subversion (SVN).

Sistem Kontrol Versi Terdistribusi (Distributed Version Control System)

Dalam sistem kontrol versi terdistribusi, setiap anggota tim memiliki salinan lengkap dari seluruh repository. Ini berarti setiap pengembang memiliki salinan lengkap sejarah perubahan, tidak hanya salinan terbaru. Contoh sistem kontrol versi terdistribusi adalah Git, Mercurial, dan Bazaar.

GIT adalah sistem kontrol versi terdistribusi yang memungkinkan pengembang perangkat lunak untuk melacak perubahan dalam kode mereka, berkolaborasi dengan anggota tim, dan mengelola revisi kode secara efektif.

Dasar Dasar Command GIT

1. **git init**

   Menginisialisasi direktori sebagai repositori Git kosong

   ```
   git init
   ```

2. **git clone**

   Menduplikasi repositori Git yang sudah ada ke direktori lokal.

```
git clone https://github.com/username/repo.git
```

3. **git status**

   Menampilkan status perubahan yang belum di commit di repository lokal.

```
git status
```

4. **git add**

   Menambahkan perubahan ke area persiapan (staging area) untuk disiapkan menjadi commit.

```
git add file1.txt
git add .
```

5. **git commit**

   Membuat commit dari perubahan yang sudah di-staging dan menambahkan pesan commit.

```
git commit -m "Menambahkan fitur login"
```

6. **git push**

   Mengirimkan commit ke repositori jarak jauh (remote repository).

```
git push origin main
```

7. **git pull**

   Mengambil commit terbaru dari repositori jarak jauh dan menggabungkannya ke repositori lokal.

```
git pull origin main
```

8. **git branch**

Menampilkan daftar cabang (branch) yang ada di repositori dan menunjukkan cabang aktif.

```
git branch
```

9. **git checkout**

   Beralih ke cabang lain atau ke commit tertentu.

```
git checkout feature-branch
git checkout abc1234 (commit hash)
```

10. **git merge**

    Menggabungkan perubahan dari satu cabang ke cabang aktif.

```
git merge feature-branch
```

11. **git log**

    Menampilkan daftar commit beserta riwayatnya dalam repositori.

```
git log
```

12. **git remote**

    Menampilkan daftar repositori jarak jauh yang terhubung dengan repositori lokal.

```
git remote -v
```

13. **git fetch**

    Mengambil informasi terbaru dari repositori jarak jauh tanpa menggabungkan perubahan.

```
git fetch origin
```

14. **git diff**

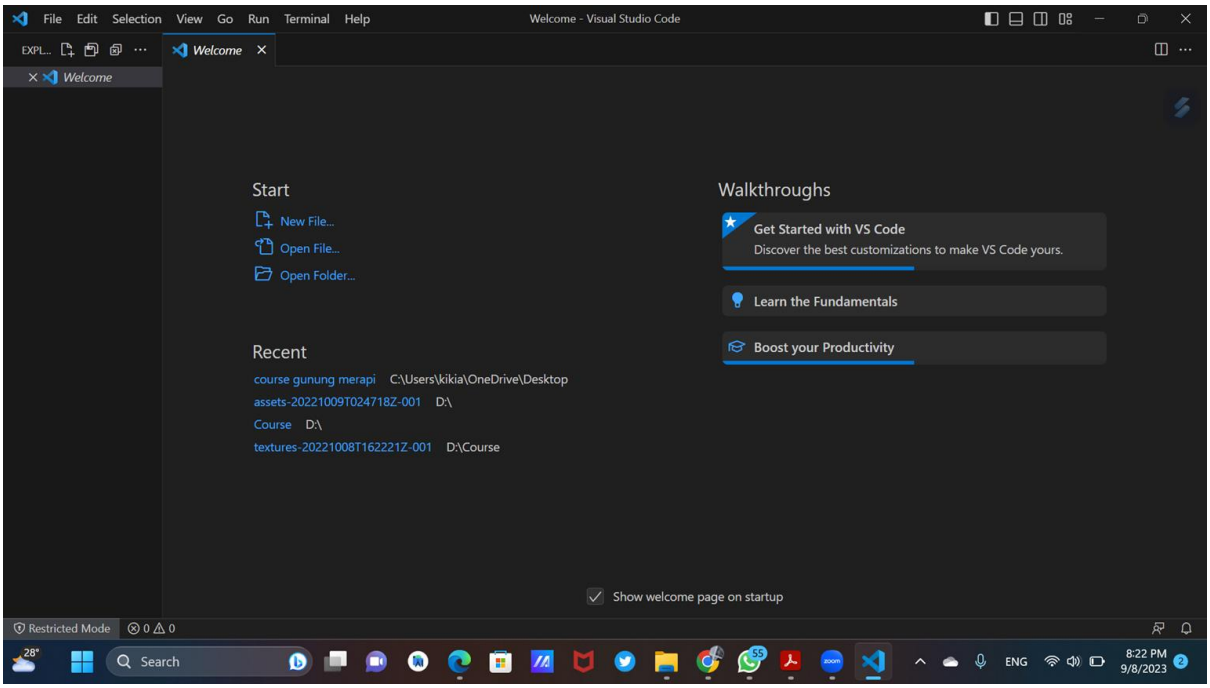
    Menampilkan perbedaan antara versi yang sudah di-staging dengan versi sebelumnya.

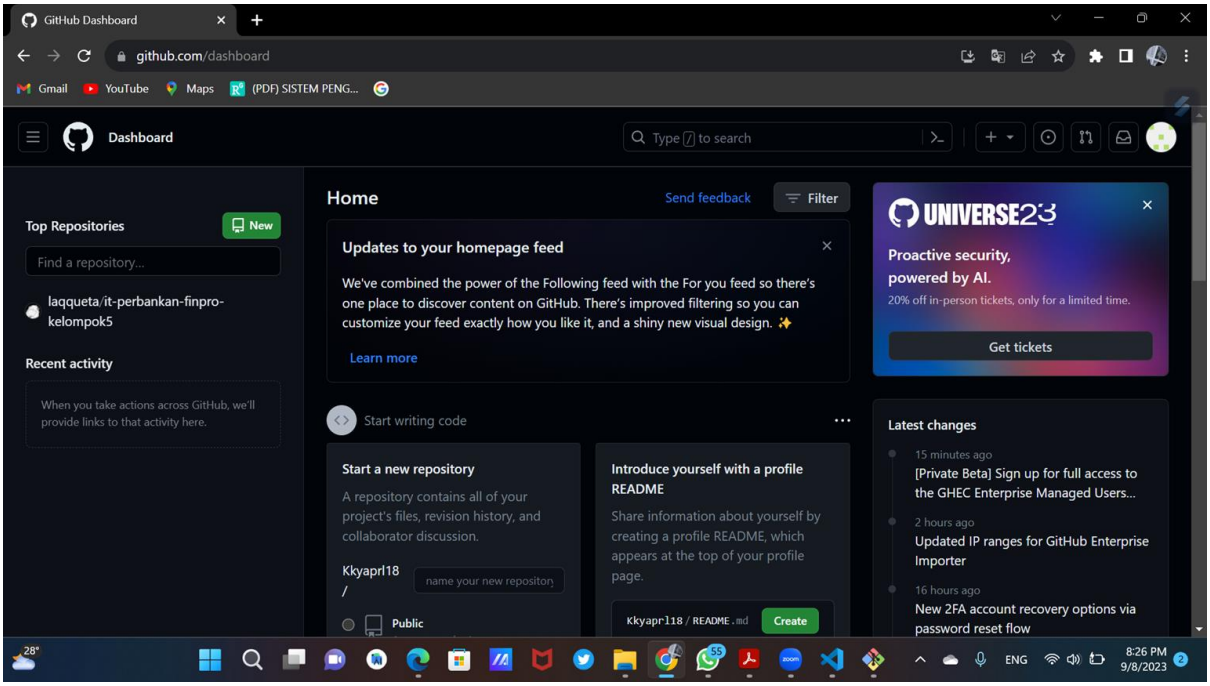
```
git diff
```

15. git reset

Mengembalikan file yang sudah di-staging ke direktori kerja sebelumnya.

```
git reset file.txt
```




GitHub Dashboard
github.com/dashboard
Gmail
YouTube
Maps
(PDF) SISTEM PENG...
Dashboard
Type / to search
Top Repositories
New
Find a repository...
laqqueta/it-perbankan-finpro-kelompok5
Recent activity
When you take actions across GitHub, we'll provide links to that activity here.
Home
Send feedback
Filter
Updates to your homepage feed
We've combined the power of the Following feed with the For you feed so there's one place to discover content on GitHub. There's improved filtering so you can customize your feed exactly how you like it, and a shiny new visual design. ✨
Learn more
UNIVERSE23
Proactive security, powered by AI.
20% off in-person tickets, only for a limited time.
Get tickets
Start writing code
Start a new repository
A repository contains all of your project's files, revision history, and collaborator discussion.
Kkyaprl18 /
name your new repository
Public
Introduce yourself with a profile README
Share information about yourself by creating a profile README, which appears at the top of your profile page.
Kkyaprl18 / README.md
Create
Latest changes
15 minutes ago
[Private Beta] Sign up for full access to the GHEC Enterprise Managed Users...
2 hours ago
Updated IP ranges for GitHub Enterprise Importer
16 hours ago
New 2FA account recovery options via password reset flow
28°
ENG
8:26 PM
9/8/2023