


# SUMMARY SOFTWARE ENGINEERING

Eriyana Azhara | <https://www.linkedin.com/in/eriyana-azhara-598237278/>



```
MINGW64 ~/Users/Eriyana/Downloads/Software Engineering/It-perbaikan-fispro-kelompok3
eriyana-kelompok3 (main) MINGW64
$ git pull
error: You have not concluded your merge (merge_head exists).
hint: Please, commit your changes before merging.
fatal: Exiting because of unfinished merge.

eriyana-kelompok3 (main) MINGW64 --Down/laah/Software Engineering/It-perbaikan-FI
$ git commit -m "resolve conflict"
$ git commit -m "resolve conflict"
fatal: cannot do a partial commit during a merge.

eriyana-kelompok3 (main) MINGW64 --Down/laah/Software Engineering/It-perbaikan-FI
$ git commit -m "resolve conflict"
[master 9088773] resolve conflict

eriyana-kelompok3 (main) MINGW64 --Down/laah/Software Engineering/It-perbaikan-FI
$ git pull origin main
From https://github.com/1kras57/it-perbaikan-fispro-kelompok3
* branch      main      -> FETCH_HEAD
Already up to date.

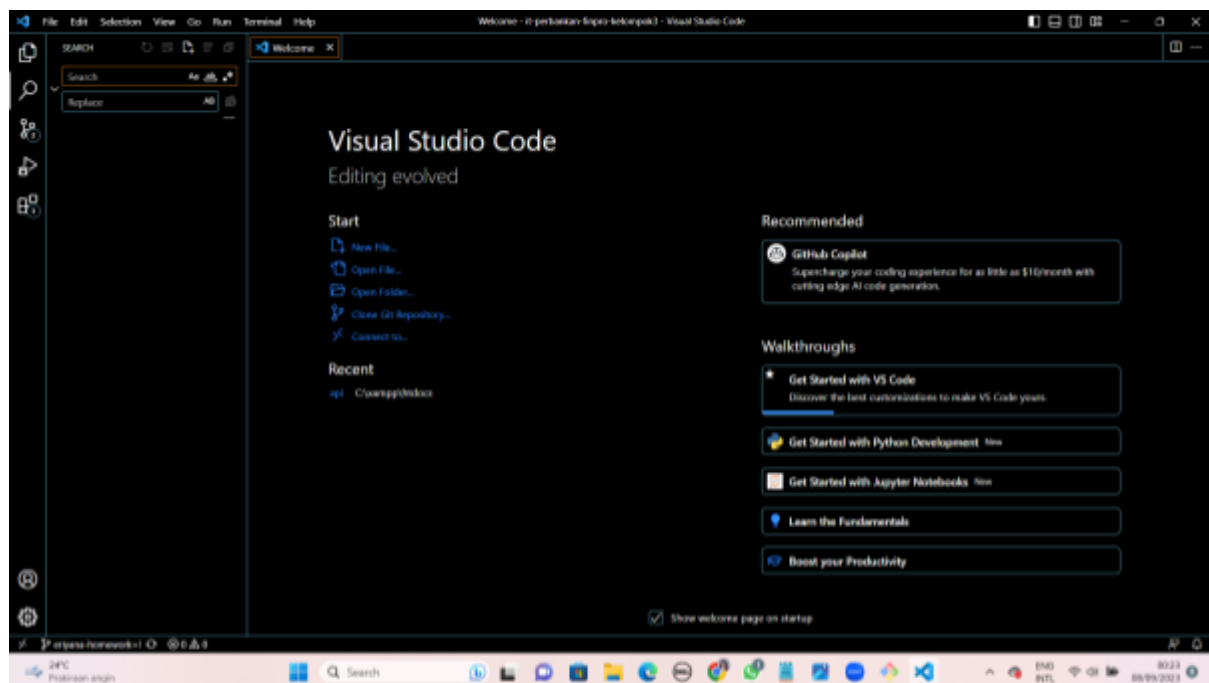
eriyana-kelompok3 (main) MINGW64 --Down/laah/Software Engineering/It-perbaikan-FI
$ git checkout eriyana-homework
Switched to branch 'eriyana-homework'
Your branch is ahead of 'origin/eriyana-homework' by 14 commits.
(use "git push" to publish your local commits)

eriyana-kelompok3 (eriyana-homework) MINGW64
$ git merge main
Auto-merging repository.txt
CONFLICT (content): Merge conflict in repository.txt
Automatic merge failed; fix conflicts and then commit the result.

eriyana-kelompok3 (eriyana-homework) MINGW64 --Down/laah/Software Engineering/It-perbaikan-FI
$ !C
eriyana-kelompok3 (eriyana-homework) MINGW64
$ git push
Enumerating objects: 12, done.
Counting objects: 100% (12/12), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (12/12), 961 bytes | 961.00 KiB/s, done.
Total 7 (delta 5), reused 6 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (12/12), completed with 1 local object.
To https://github.com/1kras57/it-perbaikan-fispro-kelompok3.git
  R15716..7d686d7  eriyana-homework -> eriyana-homework

eriyana-kelompok3 (eriyana-homework) MINGW64
$ git pull
```

Screenshot GIT



Screenshot Visual Studio Code

# Introduction Full Stack Web/Mobile Developer

## Definisi dan Scope Pengembangan Full Stack

Pengembangan Full Stack (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)

### Scope Penting Full Stack Development

- **Front End Development**  
Berfokus pada pembuatan antarmuka pengguna yang menarik dan interaktif menggunakan HTML, CSS, dan JavaScript.
- **Back End Development (Pengembangan sisi server)**  
Membangun server dan aplikasi yang berfungsi sebagai "otak" dari aplikasi, menerima permintaan dari sisi depan, memproses data, dan memberikan respons yang sesuai.
- **Database Management**  
Mendesain dan mengelola basis data untuk menyimpan, mengambil, dan memanipulasi data aplikasi.
- **Integration of Front-End and Back-End**  
Menghubungkan komponen front-end dengan layanan back-end melalui API (Application Programming Interface) 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**  
Beberapa Pengembang Full Stack juga memiliki kemampuan untuk mengembangkan aplikasi mobile menggunakan framework seperti React Native, Flutter.

### Dasar-dasar Front End Web Development

- HTML
- CSS
- Javascript

### Dasar-dasar Back End Web Development

- Bahasa Pemrograman Server-Side
- Server Framework
- Database Management

### Dasar-dasar Database Management

- Database Management System
- Tipe Database
- Bahasa Query:

### Dasar-dasar Mobile Development

- Platform Mobile
- IDE (Integrated Development Environment)

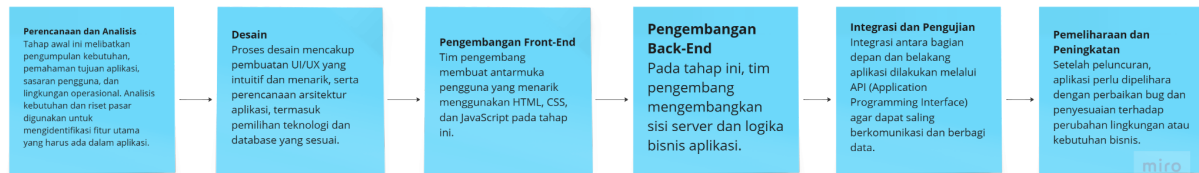
# Skillset Full Stack Web/Mobile Developer

## Pengembangan Aplikasi End to End

Merupakan pendekatan pengembangan perangkat lunak yang mencakup keseluruhan siklus pembuatan aplikasi, dari tahap perencanaan hingga tahap pengujian dan implementasi.

Tujuannya adalah untuk menghasilkan aplikasi yang lengkap, fungsional, dan siap digunakan oleh pengguna akhir.

### Tahap tahap pengembangan aplikasi end-to-end



### Penggunaan Version Control untuk Berkolaborasi

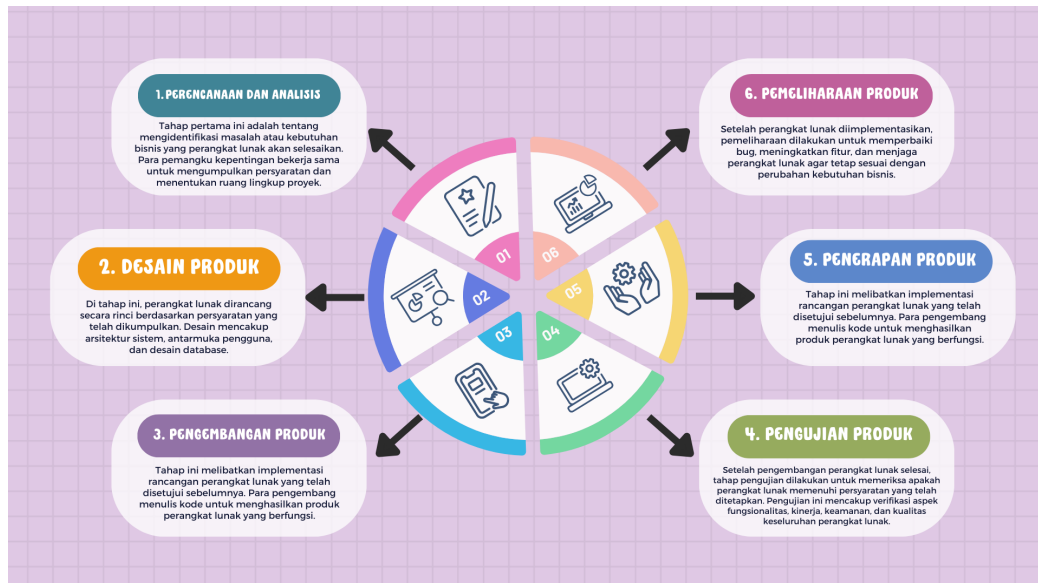
- **Inisialisasi Proyek**
  - Tim memulai proyek dengan membuat repository version control. Repository ini akan menyimpan semua kode sumber, file, dan perubahan yang dilakukan selama pengembangan.
- **Pengembangan Paralel**
  - Setiap anggota tim akan memiliki salinan repository pada komputernya sendiri. Mereka dapat bekerja secara paralel, membuat perubahan
- **Branching**
  - Version control memungkinkan pembuatan cabang (branch) yang terpisah dari kode utama. Ini memungkinkan tim untuk mengisolasi perubahan dan fitur yang sedang dikembangkan
- **Branching**
  - Version control memungkinkan pembuatan cabang (branch) yang terpisah dari kode utama. Ini memungkinkan tim untuk mengisolasi perubahan dan fitur yang sedang dikembangkan
- **Merge**
  - Setelah fitur atau perubahan selesai, cabang dapat digabungkan kembali ke cabang utama (biasanya disebut sebagai "merge").
- **Pull Request**
  - Di beberapa platform version control seperti GitHub, GitLab, dan Bitbucket, pull request adalah mekanisme yang memungkinkan pengembang untuk mengajukan perubahan mereka untuk ditinjau oleh anggota tim lain sebelum digabungkan ke cabang utama.

# SDLC & Design Thinking Implementation

## Apa itu SDLC?

SDLC (Siklus Hidup Pengembangan Perangkat Lunak) adalah rangkaian proses yang terstruktur dan metodologi yang digunakan untuk mengembangkan perangkat lunak dari awal hingga selesai. SDLC terdiri dari serangkaian tahap yang saling terkait dan dilakukan secara berurutan untuk memastikan bahwa pengembangan perangkat lunak berjalan dengan baik dan sesuai dengan kebutuhan dan tujuan yang ditentukan.

## Siklus SDLC



## Manfaat Penggunaan SDLC

- Prediktabilitas dan Pengendalian Proyek
- Peningkatan Kualitas Perangkat Lunak
- Efisiensi Tim dan Kolaborasi
- Peningkatan Dokumentasi
- Memenuhi Kebutuhan Pengguna
- Penghematan Biaya dan Waktu
- Pengelolaan Risiko yang Lebih Baik

## Model-model SDLC

### • Waterfall Mode

Waterfall model adalah model SDLC yang linier dan berurutan. Setiap tahap 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.

### • V-Shaped Model

Model V-Shaped adalah model yang terkait erat dengan model waterfall, namun menekankan pada pengujian. Tahapan pengujian diwakili oleh garis miring "V", yang berarti setiap tahap pengembangan memiliki tahapan pengujian yang sesuai.

- **Prototype Model**

Model Prototype adalah pendekatan pengembangan perangkat lunak yang membuat versi awal atau prototipe untuk memahami kebutuhan pengguna dan mengumpulkan umpan balik, sehingga perangkat lunak akhir sesuai dengan ekspektasi dan persyaratan pengguna.

- **Spiral Model**

Model ini menggabungkan pendekatan model spiral dengan inkremental. Dalam setiap siklus spiral, perangkat lunak yang semakin berkembang dengan fitur yang lebih banyak dibangun berdasarkan inkrementasi sebelumnya.

- **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.

- **Big Bang Model**

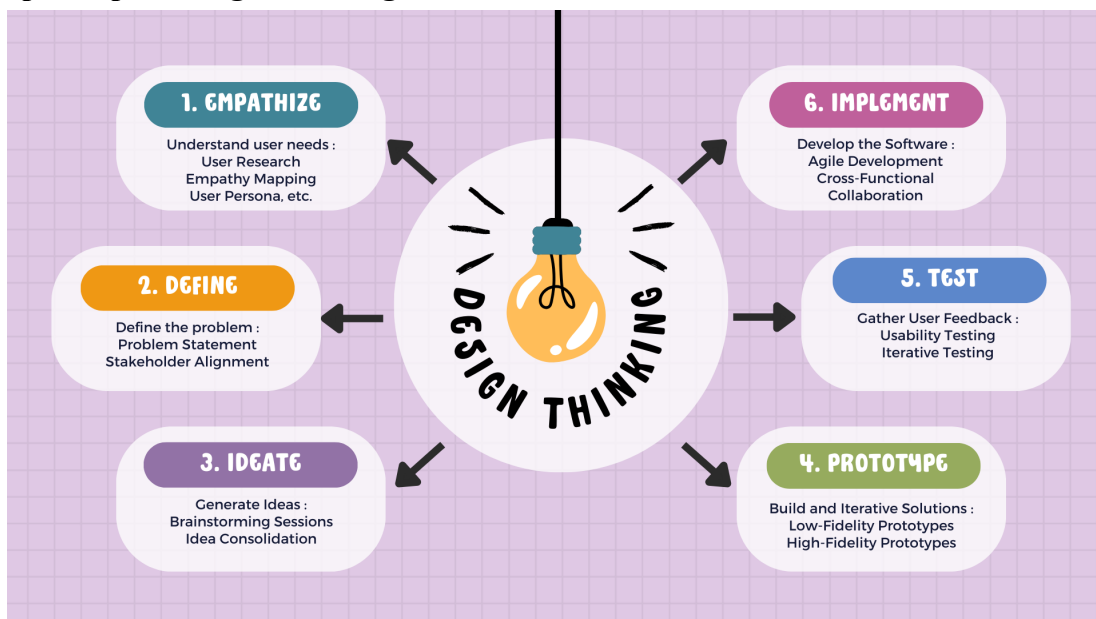
Model Big Bang adalah model yang kurang terstruktur, di mana semua tahapan pengembangan dilakukan tanpa perencanaan yang detail. Pengembangan dimulai tanpa melakukan analisis dan perencanaan yang mendalam.

- **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.

Setiap model SDLC memiliki kelebihan dan kelemahan tergantung pada jenis proyek dan kebutuhan organisasi. Pemilihan model SDLC yang tepat sangat penting untuk mencapai keberhasilan proyek pengembangan perangkat lunak.

### Tahap-tahapan Design Thinking



# Basic Git & Collaborating Using Git

## Sejarah Singkat Terminal

Dengan perkembangan teknologi dan perangkat lunak, terminal tetap menjadi alat penting bagi para pengembang perangkat lunak, administrator sistem, dan pengguna teknis lainnya. Meskipun antarmuka grafis semakin canggih dan populer, terminal tetap memberikan fleksibilitas dan kekuatan untuk melakukan tugas-tugas khusus dan otomatisasi dalam lingkungan komputer modern.

## Pengertian GIT

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

- `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 di commit 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 repository 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 repository 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.