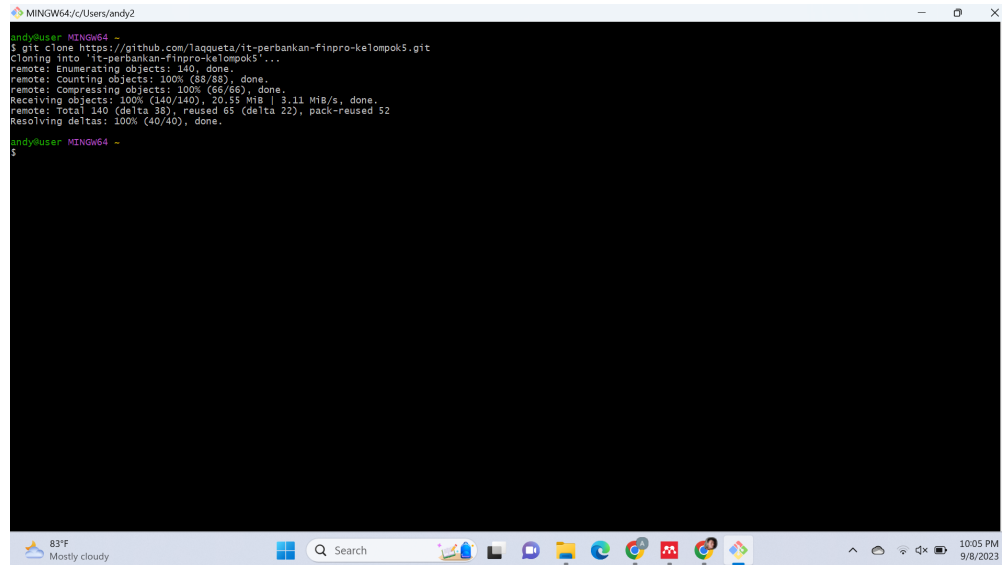


NAMA : ANDY SIBARANI  
KELOMPOK : 5  
TSA IT PERBANKAN

8 September 2023

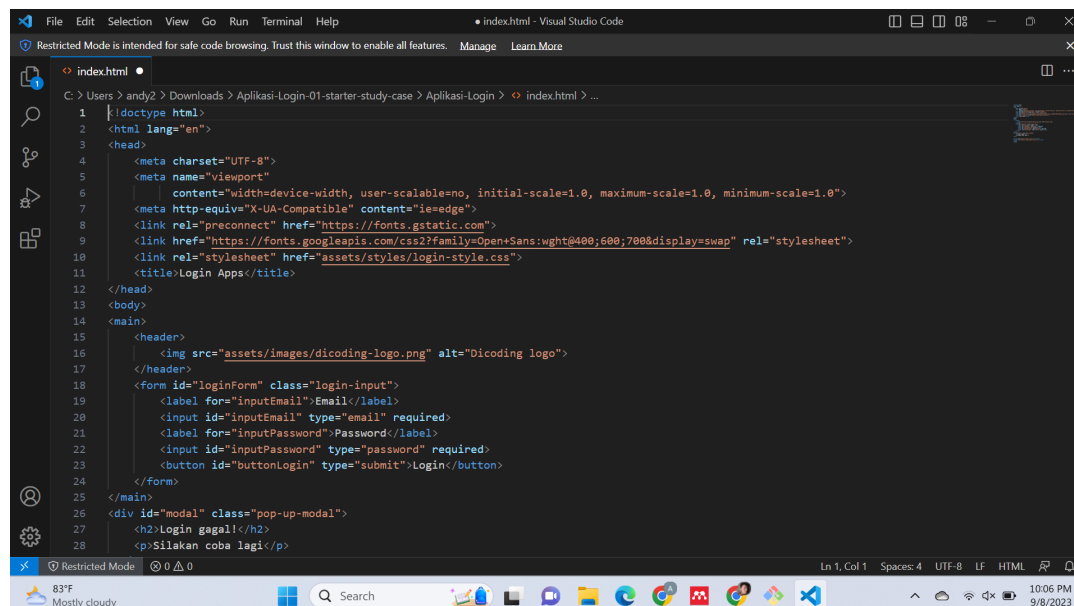
## HOMEWORK - INTRODUCTION TO SOFTWARE ENGINEERING

### SCREENSHOT GIT



```
MINGW64/c/Users/andy2
andy@user MINGW64 ~
$ git clone https://github.com/laqueta/it-perbankan-finpro-kelompok5.git
Cloning into 'it-perbankan-finpro-kelompok5'...
remote: Enumerating objects: 140, done.
remote: Counting objects: 100% (88/88), done.
remote: Compressing objects: 100% (66/66), done.
Receiving objects: 100% (140/140), 20.55 MiB | 3.11 MiB/s, done.
remote: Total 140 (delta 38), reused 65 (delta 22), pack-reused 52
Resolving deltas: 100% (40/40), done.
andy@user MINGW64 ~
$
```

### SCREENSHOT VISUAL STUDIO CODE



```
File Edit Selection View Go Run Terminal Help
index.html - Visual Studio Code
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
index.html
C:\Users\andy2\Downloads> Aplikasi-Login-01-starter-study-case > Aplikasi-Login > index.html > ...
1 <!doctype html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport"
6 | content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
7 <meta http-equiv="X-UA-Compatible" content="ie=edge">
8 <link rel="preconnect" href="https://fonts.gstatic.com">
9 <link href="https://fonts.googleapis.com/css2?family=Open+Sans:wght@400;600;700&display=swap" rel="stylesheet">
10 <link rel="stylesheet" href="assets/styles/login-style.css">
11 <title>Login Apps</title>
12 </head>
13 <body>
14 <main>
15 <header>
16 
17 </header>
18 <form id="loginForm" class="login-input">
19 <label for="inputEmail">Email</label>
20 <input id="inputEmail" type="email" required>
21 <label for="inputPassword">Password</label>
22 <input id="inputPassword" type="password" required>
23 <button id="buttonLogin" type="submit">Login</button>
24 </form>
25 </main>
26 <div id="modal" class="pop-up-modal">
27 <h2>Login gagal!</h2>
28 <p>Silakan coba lagi</p>
```

## **SUMMARY SOFTWARE ENGINEERING**

### **A. Introduction Full Stack Web/Mobile Developer**

Seorang Full Stack Web/Mobile Developer memiliki kemampuan dalam mengembangkan aplikasi web dan mobile dari sisi depan (front-end) hingga sisi belakang (back-end). Dalam peran ini, mereka bertanggung jawab untuk menguasai berbagai teknologi dan bahasa pemrograman yang relevan untuk membangun aplikasi yang lengkap dan fungsional.

Seorang Full Stack Developer harus memiliki pemahaman yang kuat tentang pengembangan web dan mobile, termasuk desain antarmuka pengguna, logika bisnis, manajemen basis data, dan integrasi dengan API. Anda juga harus dapat bekerja dengan berbagai teknologi seperti HTML, CSS, JavaScript, serta bahasa pemrograman seperti Python, Ruby, PHP, atau JavaScript (Node.js).

Mereka juga harus memiliki pengetahuan tentang kerangka kerja (framework) populer seperti React, Angular, atau Vue.js untuk mengembangkan antarmuka pengguna yang responsif dan menarik. Di sisi server, mereka perlu menguasai kerangka kerja seperti Express.js atau Django untuk mengelola logika bisnis dan komunikasi dengan basis data.

### **B. Skillset Full Stack Web/Mobile Developer**

Sebagai seorang Full Stack Web/Mobile Developer, terdapat beberapa keterampilan yang penting untuk dikuasai. Berikut adalah beberapa skillset yang perlu dimiliki oleh seorang Full Stack Developer:

1. Pengembangan Front-end:  
memiliki pemahaman yang baik tentang HTML, CSS, dan JavaScript. Kemampuan dalam menggunakan kerangka kerja (framework) seperti React, Angular, atau Vue.js juga sangat diperlukan untuk membangun antarmuka pengguna yang menarik dan responsif.
2. Pengembangan Back-end:  
perlu menguasai bahasa pemrograman server-side seperti Python, Ruby, PHP, atau JavaScript (Node.js). Kemampuan dalam menggunakan kerangka kerja seperti Express.js, Django, atau Laravel juga penting untuk mengelola logika bisnis dan komunikasi dengan basis data.
3. Basis Data:  
harus memiliki pengetahuan tentang manajemen basis data dan bahasa kueri seperti SQL. Memahami konsep dasar basis data dan penggunaan teknologi seperti MySQL, PostgreSQL, atau MongoDB sangat diperlukan.
4. API dan Integrasi:  
Kemampuan untuk mengintegrasikan aplikasi dengan API eksternal, seperti API pihak ketiga atau layanan cloud, sangat penting. Anda harus memahami konsep RESTful API dan memiliki pengetahuan tentang protokol komunikasi seperti JSON atau XML.
5. Pengujian dan Debugging:  
harus memiliki keterampilan dalam menguji dan melakukan debugging pada aplikasi. Memahami konsep pengujian otomatis dan menggunakan alat-alat seperti Jest, Mocha, atau Selenium dapat membantu memastikan kualitas aplikasi yang dikembangkan.
6. Keamanan Aplikasi:  
Memahami prinsip keamanan aplikasi dan memiliki pengetahuan tentang praktik terbaik dalam melindungi data pengguna sangat penting. Mereka harus dapat menerapkan langkah-langkah keamanan seperti validasi input, enkripsi data, dan perlindungan terhadap serangan umum seperti serangan injeksi SQL atau serangan lintas situs (XSS).
7. Keterampilan Komunikasi dan Kolaborasi:  
seorang Full Stack Developer akan bekerja dalam tim dan berinteraksi dengan berbagai pemangku kepentingan. Keterampilan komunikasi yang baik dan kemampuan untuk bekerja dalam tim sangat penting untuk mencapai kesuksesan proyek.
8. Kemampuan Analitis:

harus memiliki kemampuan analitis yang baik untuk memecahkan masalah dan mengoptimalkan kinerja aplikasi. Memahami konsep algoritma dan struktur data juga dapat membantu dalam mengembangkan solusi yang efisien dan scalable.

### **C. Tools Full Stack Web/Mobile Developer**

Sebagai seorang Full Stack Web/Mobile Developer, terdapat beberapa alat (tools) yang dapat membantu Anda dalam mengembangkan aplikasi web dan mobile. Berikut adalah beberapa alat yang umum digunakan oleh Full Stack Developer:

1. Editor Kode:  
seorang pengembang akan menghabiskan banyak waktu untuk menulis dan mengedit kode. Beberapa editor kode populer yang dapat digunakan adalah Visual Studio Code, Sublime Text, atau Atom. Editor kode ini dilengkapi dengan fitur-fitur yang membantu meningkatkan produktivitas dan memudahkan dalam menulis kode.
2. Git:  
Git adalah sistem kontrol versi yang penting dalam pengembangan perangkat lunak. Dengan Git, mereka dapat melacak perubahan kode, bekerja secara kolaboratif dengan tim, dan mengelola proyek dengan lebih efisien. Platform hosting Git seperti GitHub, GitLab, atau Bitbucket juga sering digunakan untuk menyimpan dan berbagi kode.
3. Framework:  
Framework adalah kerangka kerja yang membantu mempercepat pengembangan aplikasi dengan menyediakan struktur dan komponen yang sudah siap pakai. Beberapa framework populer untuk pengembangan web adalah React, Angular, dan Vue.js. Untuk pengembangan aplikasi mobile, mereka dapat menggunakan framework seperti React Native atau Flutter.
4. Basis Data:  
Dalam pengembangan aplikasi, mereka akan sering berinteraksi dengan basis data. Beberapa basis data populer yang digunakan oleh Full Stack Developer adalah MySQL, PostgreSQL, MongoDB, atau Firebase. Alat-alat seperti MySQL Workbench atau MongoDB Compass dapat membantu Anda dalam mengelola dan mengakses basis data.
5. Postman:  
Postman adalah alat yang digunakan untuk menguji dan mengelola API. Dengan Postman, mereka dapat mengirim permintaan HTTP ke API, menguji respons, dan melihat hasilnya. Alat ini sangat berguna dalam mengintegrasikan aplikasi dengan API eksternal.
6. DevTools:  
Setiap browser modern memiliki alat pengembang (developer tools) yang dapat membantu mereka dalam menginspeksi dan memperbaiki aplikasi web. DevTools memungkinkan Anda untuk melihat dan mengubah elemen HTML, menganalisis jaringan, dan melakukan debugging JavaScript.
7. Docker:  
Docker adalah platform yang memungkinkan mereka untuk mengemas aplikasi ke dalam wadah (container) yang dapat dijalankan di berbagai lingkungan. Dengan Docker, mereka dapat memastikan aplikasi berjalan dengan konsisten di berbagai mesin dan menghindari masalah konfigurasi.
8. Task Runner dan Bundler:  
Task runner seperti Gulp atau Grunt membantu mereka dalam mengotomatisasi tugas-tugas pengembangan seperti kompilasi CSS atau JavaScript, optimasi gambar, atau menjalankan tes. Bundler seperti Webpack atau Parcel digunakan untuk mengelola dependensi dan menggabungkan file-file sumber menjadi satu file yang lebih efisien.
9. Platform Cloud:

Platform cloud seperti AWS, Google Cloud, atau Microsoft Azure menyediakan infrastruktur dan layanan yang dapat membantu mereka dalam menyebarkan dan mengelola aplikasi web dan mobile. Dengan menggunakan platform cloud, Anda dapat mengurangi kompleksitas operasional dan meningkatkan skalabilitas aplikasi.

#### **D. What is SDLC**

SDLC (Siklus Hidup Pengembangan Sistem) adalah suatu pendekatan atau metodologi yang digunakan dalam pengembangan perangkat lunak atau sistem. Tahapan-tahapan dalam SDLC dapat bervariasi tergantung pada metodologi yang digunakan, namun secara umum terdapat beberapa tahapan yang umum ditemui dalam SDLC, antara lain:

1. **Perencanaan dan Analisis:**  
Tahap ini melibatkan identifikasi kebutuhan pengguna, analisis risiko, dan perencanaan proyek secara keseluruhan. Pada tahap ini, tim pengembang juga merumuskan tujuan proyek dan mengidentifikasi sumber daya yang diperlukan. Tim pengembang melakukan analisis kebutuhan, menganalisis proses bisnis yang ada, dan merancang solusi yang sesuai.
2. **Desain:**  
Tahap ini melibatkan merancang arsitektur sistem, desain antarmuka pengguna, dan desain basis data. Tim pengembang juga merancang algoritma dan struktur data yang diperlukan.
3. **Pengembangan:**  
Tahap ini melibatkan implementasi desain menjadi kode program yang dapat dijalankan. Tim pengembang menulis kode, menguji fungsionalitas, dan melakukan debugging untuk memastikan bahwa sistem berjalan dengan baik.
4. **Pengujian:**  
Tahap ini melibatkan pengujian sistem secara menyeluruh untuk memastikan bahwa sistem berfungsi sesuai dengan kebutuhan dan spesifikasi yang telah ditetapkan. Pengujian dilakukan untuk mengidentifikasi dan memperbaiki bug atau kesalahan yang mungkin ada.
5. **Penerapan:**  
Tahap ini melibatkan peluncuran sistem ke lingkungan produksi. Tim pengembang melakukan migrasi data, instalasi perangkat lunak, dan pelatihan pengguna.
6. **Pemeliharaan:**  
Tahap ini melibatkan pemeliharaan sistem setelah diluncurkan. Tim pengembang melakukan pemantauan, perbaikan bug, dan peningkatan sistem sesuai dengan kebutuhan yang muncul.

#### **E. Model-Model Software Development Life Cycle (SDLC)**

Siklus Hidup Pengembangan Perangkat Lunak (SDLC) memiliki beberapa model yang digunakan dalam proses pengembangan perangkat lunak. Berikut adalah beberapa model SDLC yang umum digunakan:

1. **Model Air Terjun (Waterfall Model):**  
Model ini mengikuti pendekatan linear, di mana setiap tahap dalam SDLC dilakukan secara berurutan. Tahap-tahapnya meliputi analisis kebutuhan, desain, implementasi, pengujian, dan pemeliharaan. Setelah satu tahap selesai, baru dilanjutkan ke tahap berikutnya. Model ini cocok untuk proyek dengan kebutuhan yang stabil dan jelas.
2. **Model V-Shaped:**  
Model ini merupakan variasi dari model Air Terjun, di mana setiap tahap pengujian memiliki tahap yang sesuai. Setelah tahap analisis dan desain selesai, tahap pengujian yang sesuai dilakukan. Model ini memastikan bahwa pengujian dilakukan sejalan dengan tahap pengembangan utama.
3. **Model Prototipe (Prototype Model):**  
Model ini melibatkan pembuatan prototipe awal yang digunakan untuk memvalidasi kebutuhan pengguna. Prototipe ini dapat diuji dan diberikan kepada pengguna untuk mendapatkan umpan

balik. Berdasarkan umpan balik tersebut, perubahan dan perbaikan dapat dilakukan sebelum memulai pengembangan penuh.

4. Model Spiral:

Model ini menggabungkan elemen-elemen dari model Air Terjun dengan pendekatan iteratif. Proses pengembangan dilakukan dalam siklus spiral, di mana setiap siklus melibatkan perencanaan, analisis risiko, pengembangan, dan evaluasi. Model ini cocok untuk proyek yang kompleks dan berisiko tinggi.

5. Model Iteratif Inkremental (Iterative Incremental Model):

Model ini melibatkan pengembangan sistem dalam bagian-bagian kecil yang disebut inkremen. Setiap inkremen merupakan versi fungsional dari sistem yang dapat digunakan oleh pengguna. Setiap inkremen ditambahkan secara bertahap hingga sistem mencapai fungsionalitas penuh.

6. Model Big Bang:

Model ini melibatkan pengembangan perangkat lunak tanpa adanya perencanaan yang terperinci. Semua tahap pengembangan dilakukan secara bersamaan tanpa mengikuti urutan yang jelas. Model ini cocok untuk proyek yang kecil dan tidak kompleks.

7. Model Agile:

Model ini melibatkan pengembangan perangkat lunak dalam iterasi pendek yang disebut sprint. Setiap sprint melibatkan perencanaan, analisis, desain, implementasi, dan pengujian. Tim pengembang bekerja secara kolaboratif dan fleksibel untuk menghasilkan perangkat lunak yang dapat diubah dengan cepat sesuai dengan kebutuhan pengguna.

## **F. Design Thinking Implementation**

Design Thinking adalah pendekatan yang digunakan untuk memecahkan masalah dan menciptakan solusi inovatif dengan fokus pada kebutuhan pengguna. Berikut adalah langkah-langkah yang terlibat dalam implementasi Design Thinking:

1. Empatis:

Tahap pertama dalam implementasi Design Thinking adalah memahami kebutuhan pengguna. Tim proyek harus berusaha memahami pengguna secara mendalam, termasuk tujuan, tantangan, dan preferensi mereka. Hal ini dapat dilakukan melalui wawancara, observasi, atau pengamatan langsung.

2. Definisi:

Setelah memahami pengguna, tim proyek perlu mendefinisikan masalah yang akan diselesaikan. Definisi yang jelas dan spesifik membantu tim tetap fokus pada tujuan yang ingin dicapai. Dalam hal ini, tim harus memperjelas kebutuhan pengguna dan mengidentifikasi hambatan yang perlu diatasi.

3. Ideasi:

Setelah masalah didefinisikan, tim proyek perlu menghasilkan sebanyak mungkin ide solusi yang kreatif dan inovatif. Dalam tahap ini, tidak ada batasan dalam berpikir, dan semua ide diterima. Teknik brainstorming, mind mapping, atau prototip bisa digunakan untuk menghasilkan ide-ide baru.

4. Prototipe:

Ide-ide yang dihasilkan pada tahap sebelumnya perlu diuji dan dievaluasi. Tim proyek dapat membuat prototipe yang cepat dan iteratif dari solusi yang diusulkan. Prototipe ini bisa berupa model fisik, wireframe, atau mock-up. Tujuan dari pembuatan prototipe adalah untuk mendapatkan umpan balik dari pengguna dan mengidentifikasi perbaikan yang perlu dilakukan.

5. Pengujian:

Prototipe yang dibuat harus diuji dengan pengguna yang sebenarnya. Pengujian ini membantu tim proyek memahami keefektifan solusi yang diusulkan dan menemukan kelemahan yang perlu

diperbaiki. Pengujian dapat dilakukan melalui wawancara, observasi, atau penggunaan alat pengujian lainnya. Umpan balik dari pengguna harus dikumpulkan dan dianalisis.

6. Implementasi:

Setelah solusi yang dihasilkan telah diuji dan diverifikasi, tahap implementasi dimulai. Tim proyek perlu mengembangkan perangkat lunak sesuai dengan desain solusi yang telah diuji. Proses implementasi melibatkan pengkodean, pemrograman, dan pengujian lebih lanjut untuk memastikan keberhasilan solusi yang dikembangkan.

### **G. Terminal and IDE**

Terminal adalah antarmuka teks yang digunakan untuk berinteraksi dengan sistem operasi melalui perintah-perintah baris perintah. Dalam terminal, pengguna dapat menjalankan perintah-perintah untuk menjalankan program, mengelola file dan direktori, dan melakukan tugas-tugas lain yang terkait dengan sistem operasi. Terminal memberikan fleksibilitas dan kekuatan kepada pengguna untuk mengontrol sistem operasi dengan cara yang tidak mungkin dilakukan melalui antarmuka grafis.

IDE adalah lingkungan pengembangan perangkat lunak yang menyediakan berbagai fitur untuk memudahkan pengembang dalam menulis, mengedit, dan mengelola kode program. IDE biasanya terdiri dari editor teks yang kuat, kompiler atau interpreter yang terintegrasi, dan alat bantu lainnya seperti debugger dan pengaturan proyek. IDE memberikan antarmuka yang intuitif dan berorientasi pengembangan yang mempercepat proses pengembangan perangkat lunak dan meningkatkan produktivitas pengembang.

### **H. Installing, initializing and committing GIT**

Git adalah sistem kontrol versi yang digunakan untuk mengelola perubahan pada file dokumen, source code, atau informasi lainnya. Berikut adalah beberapa dasar-dasar command Git yang sering digunakan:

1. `git init`:  
Command ini digunakan untuk membuat repositori Git baru di direktori kerja saat ini.
2. `git clone [URL]`:  
Command ini digunakan untuk mengunduh repositori Git yang sudah ada dari URL yang diberikan.
3. `git add [file]`:  
Command ini digunakan untuk menambahkan file ke staging area, yang akan disertakan dalam commit berikutnya.
4. `git commit -m "[pesan commit]"`:  
Command ini digunakan untuk membuat commit baru dengan pesan yang menjelaskan perubahan yang dilakukan.
5. `git status`:  
Command ini digunakan untuk melihat status repositori Git saat ini, termasuk file yang telah diubah, ditambahkan, atau dihapus.
6. `git push`:  
Command ini digunakan untuk mengirim perubahan lokal ke repositori jarak jauh (remote repository), seperti GitHub.
7. `git pull`:  
Command ini digunakan untuk mengambil perubahan terbaru dari repositori jarak jauh dan menggabungkannya dengan perubahan lokal.
8. `git branch`:  
Command ini digunakan untuk melihat daftar branch yang ada dalam repositori.
9. `git checkout [branch]`:  
Command ini digunakan untuk beralih ke branch lain dalam repositori.

10. `git merge [branch]`:

Command ini digunakan untuk menggabungkan perubahan dari branch lain ke branch saat ini.

**I. Collaborating Using Git**

Berikut adalah beberapa cara untuk berkolaborasi menggunakan Git:

1. Membuat Repositori:

Pertama, Anda perlu membuat repositori Git di platform seperti GitHub atau GitLab. Repositori ini akan menjadi tempat untuk menyimpan dan berbagi kode dengan tim Anda.

2. Mengunduh Repositori:

Setiap anggota tim dapat mengunduh repositori menggunakan perintah `git clone [URL]`. URL ini dapat ditemukan di halaman repositori di platform yang Anda gunakan.

3. Menggunakan Branch:

Setiap anggota tim dapat membuat branch baru untuk bekerja pada fitur atau perubahan tertentu. Misalnya, dengan perintah `git branch [nama branch]`, Anda dapat membuat branch baru dengan nama "fitur-login".

4. Mengubah Kode:

Setiap anggota tim dapat melakukan perubahan pada kode di branch mereka masing-masing. Anda dapat menggunakan perintah `git add [nama file]` untuk menambahkan perubahan ke staging area, dan `git commit -m "[pesan commit]"` untuk membuat commit dengan pesan yang menjelaskan perubahan yang dilakukan.

5. Menggabungkan Perubahan:

Setelah selesai membuat perubahan, Anda dapat menggabungkan branch Anda dengan branch utama menggunakan perintah `git merge [nama branch]`. Ini akan menggabungkan perubahan Anda ke dalam branch utama.

6. Mengirim Perubahan:

Setelah perubahan selesai, Anda dapat mengirim perubahan Anda ke repositori jarak jauh menggunakan perintah `git push`. Ini akan mengunggah perubahan Anda ke platform Git yang Anda gunakan.

7. Mengambil Perubahan:

Jika ada perubahan dari anggota tim lain, Anda dapat mengambil perubahan terbaru menggunakan perintah `git pull`. Ini akan menggabungkan perubahan terbaru ke dalam branch Anda.