TSA

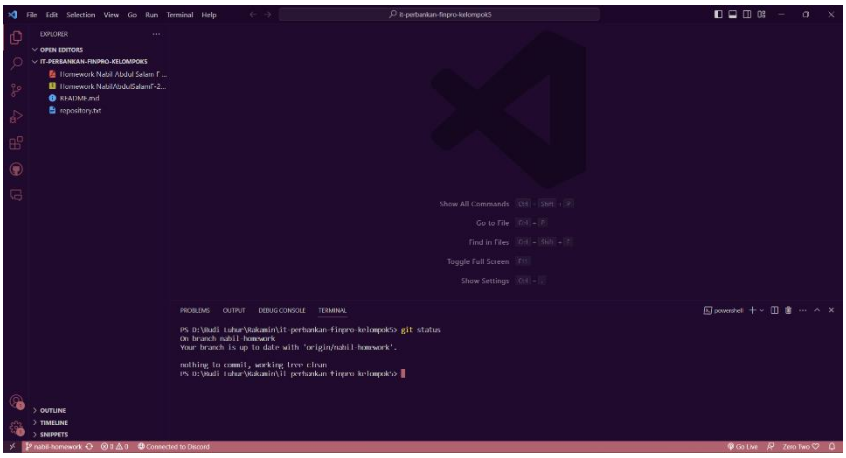
IT Perbankan

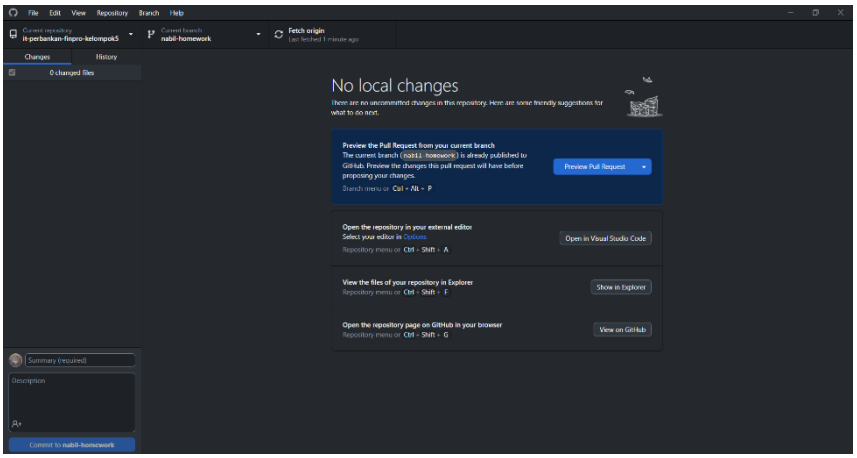
Universitas Budi Luhur

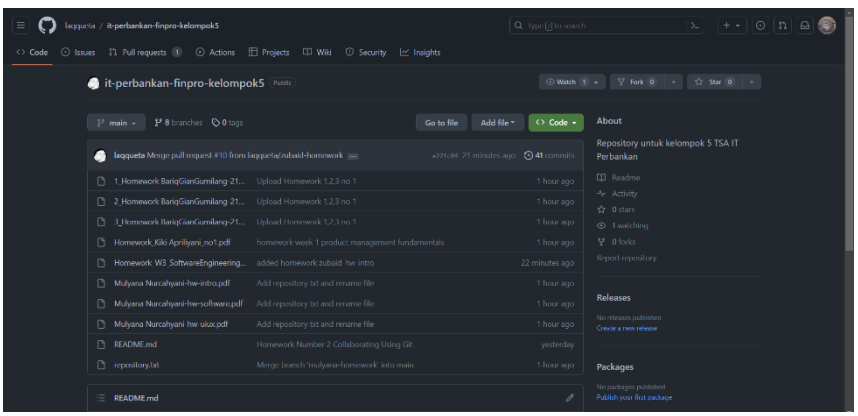
Nama : Nabil Abdul Salam F

NIM : 2111500209

Introduction to Software Engineering Summary







- **Full Stack Development (Pengembangan Full Stack)**

  Full Stack Development merupakan pengembangan perangkat lunak di mana seorang pengembang memiliki keterampilan dan pengetahuan untuk mengembangkan seluruh aplikasi secara end-to-end, mulai dari sisi depan (front-end) hingga sisi belakang (back-end) dan, dalam beberapa kasus, hingga sisi klien (client-side).

- **Scope Pada Full Stack Development**

  1. **Front-End Development (Pengembangan Sisi Depan) :**
     - Fokus pada pembuatan antarmuka pengguna yang menarik dan interaktif menggunakan HTML, CSS, dan JavaScript.
     - Menggunakan framework dan library front-end seperti React, Angular, Vue.js, atau jQuery untuk mempercepat pengembangan dan meningkatkan efisiensi.

  2. **Back-End Development (Pengembangan Sisi Belakang) :**
     - Bertanggung jawab atas pembuatan server dan aplikasi yang berfungsi sebagai otak dari aplikasi, memproses permintaan dari Front-End, dan memberikan respons yang sesuai.
     - Menggunakan bahasa pemrograman server-side seperti Node.js, Python, Ruby, Java, PHP, atau C#.

  3. **Database Management (Manajemen Basis Data):**
     - Melibatkan desain dan pengelolaan basis data untuk menyimpan, mengambil, dan memanipulasi data aplikasi.
     - Menggunakan teknologi database seperti MySQL, PostgreSQL, MongoDB, atau Firebase.

  4. **Integration of Front-End and Back-End (Integrasi Sisi Depan dan Sisi Belakang):**
     - Menghubungkan komponen front-end dengan layanan back-end melalui API (Application Programming Interface) untuk berkomunikasi dengan server dan database.
     - Menyelaraskan data dan tampilan antara sisi depan dan sisi belakang aplikasi.

  5. **Version Control and Collaboration (Pengendalian Versi dan Kolaborasi):**
     - Menggunakan sistem pengendalian versi seperti Git untuk mengelola perubahan kode dan kolaborasi dalam tim pengembang.
     - Memastikan bahwa kode terus berkembang dengan aman dan sesuai dengan tujuan proyek.

6. **Mobile Development (Pengembangan Aplikasi Seluler):**
    - Beberapa Pengembang Full Stack juga memiliki kemampuan untuk mengembangkan aplikasi seluler menggunakan framework seperti React Native atau Flutter.

- **Pengembangan Aplikasi End to End**

Pengembangan perangkat lunak yang melibatkan seluruh siklus pembuatan aplikasi, mulai dari perencanaan hingga pengujian dan implementasi. Tujuannya adalah menghasilkan aplikasi yang siap digunakan, lengkap, dan berfungsi dengan baik bagi pengguna akhir / end-user.

- **Tahap-Tahap Pengembangan Aplikasi end-to-end**

1. **Perencanaan dan Analisis :**
    - Tahap awal yang melibatkan pengumpulan kebutuhan, pemahaman tujuan aplikasi, sasaran pengguna, dan lingkungan operasional.
    - Analisis kebutuhan dan riset pasar dilakukan untuk mengidentifikasi fitur utama aplikasi.

2. **Desain :**
    - Proses desain mencakup perancangan antarmuka pengguna (UI) dan pengalaman pengguna (UX) yang intuitif dan menarik.
    - Perencanaan arsitektur aplikasi, termasuk pemilihan teknologi, database, dan framework yang sesuai.

3. **Pengembangan Front-End :**
    - Tim pengembang bekerja pada bagian depan aplikasi, menggunakan HTML, CSS, dan JavaScript untuk menciptakan tampilan dan interaksi yang menarik bagi pengguna.
    - Mungkin menggunakan framework front-end seperti React, Angular, atau Vue.js.

4. **Pengembangan Back-End :**
    - Melibatkan pengembangan sisi server dan logika bisnis aplikasi.
    - Menggunakan bahasa pemrograman server-side seperti Node.js, Python, Ruby, atau Java, dan framework back-end seperti Express.js, Flask, atau Ruby on Rails.

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

6. **Pemeliharaan dan Peningkatan :**
    - Setelah peluncuran, aplikasi memerlukan pemeliharaan untuk mengatasi bug dan perubahan lingkungan atau kebutuhan bisnis.
    - Terus melakukan peningkatan untuk memperbarui fitur, meningkatkan kinerja, dan menjaga relevansi aplikasi.

- **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** : Source tree, GitLens
    4) **DBMS :** PostgreSQL**,** MySQL, ORACLE, mongoDB, redis
    5) **API :** postman, swagger
    6) **Tests dan Debugging :** Jest, Junit, Mocha & Chai
    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)**

  Serangkaian proses dan metodologi yang terstruktur untuk mengembangkan perangkat lunak dari awal hingga selesai. Tahap-tahapnya dilakukan secara berurutan untuk memastikan pengembangan perangkat lunak sesuai dengan kebutuhan dan tujuan yang ditentukan.

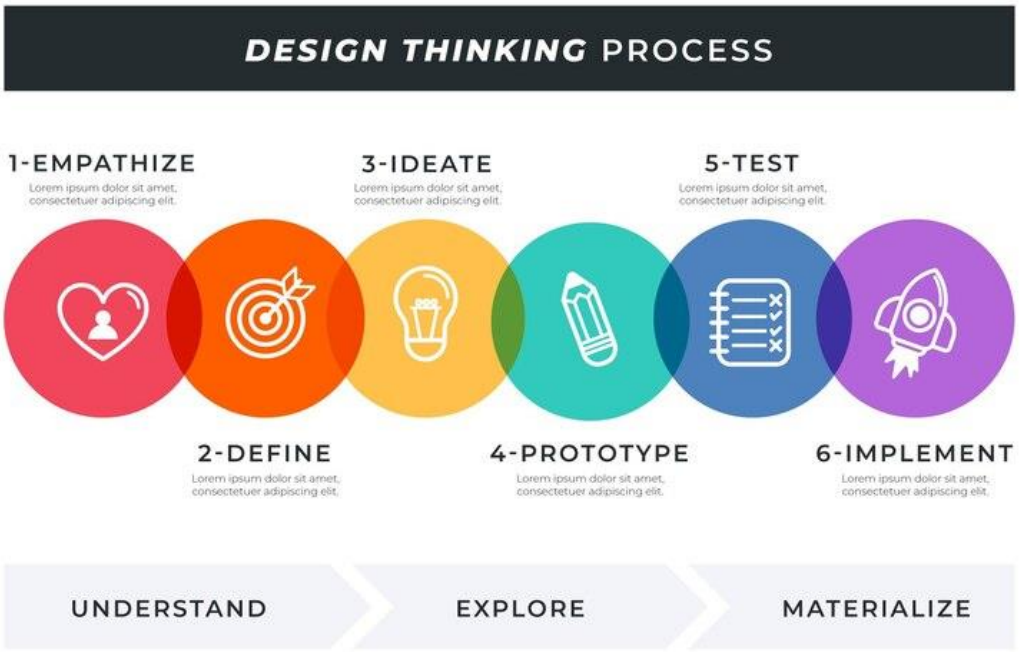
- **Siklus SDLC**
  1) Perencanaan dan Analisis
  2) Desain
  3) Pengembangan
  4) Pengujian
  5) Penerapan
  6) Pemeliharaan

- **Manfaat Penggunaan SDLC**
  1) Prediktabilitas dan Pengendalian Proyek
  2) Peningkatan Kualitas Perangkat Lunak
  3) Efisiensi Tim dan Kolaborasi
  4) Pengelolaan Risiko yang Lebih Baik
  5) Memenuhi Kebutuhan Pengguna
  6) Penghematan Biaya dan Waktu
  7) Meningkatkan Pengawasan dan Evaluasi
  8) Peningkatan Dokumentas

- **Model Model SDLC**
  1) **Waterfall Model** : model SDLC yang linier dan berurutan
  2) **V-Shaped Model** : model yang terkait erat dengan model waterfall, tetapi menekankan pada pengujian.
  3) **Prototype Model** : model pengembangan perangkat lunak yang bertujuan untuk menciptakan prototipe atau contoh awal sebelum mengembangkan versi finalnya.
  4) **Spiral Model** : Model ini menggabungkan elemen model spiral dengan pendekatan inkremental.
  5) **Iterative Incremental Model** : Model ini melibatkan pengulangan siklus pembangunan dan peningkatan perangkat lunak dalam tahapan-tahapan kecil.
  6) **Big Bang Model** : Model Big Bang adalah model yang kurang terstruktur, di mana semua tahapan pengembangan dilakukan tanpa perencanaan yang detail.
  7) **Agile Model** : Model Agile adalah pendekatan kolaboratif dan iteratif yang berfokus pada pengiriman perangkat lunak secara berkala dan inkremental.

- **Design Thinking Implementation**

  Design Thinking adalah pendekatan berpusat pada pengguna yang melibatkan pemahaman mendalam terhadap masalah pengguna, mendefinisikan masalah yang akan diselesaikan, menghasilkan ide-ide kreatif, membuat prototipe, menguji dengan pengguna, mengimplementasikan solusi perangkat lunak, dan mengevaluasi keberhasilan solusi tersebut. Proses ini bertujuan untuk menciptakan solusi perangkat lunak yang inovatif dan sesuai dengan kebutuhan pengguna akhir.



- **Design Thinking Proceses**
  1) **Empathize:** Understand User Needs
  2) **Define:** Define the Problem
  3) **Ideate:** Generate Ideas
  4) **Prototype:** Build and Iterative Solutions
  5) **Test:** Gather User Feedback
  6) **Implement:** Develop the Software

- **Terminal**

  Dalam era teknologi dan perangkat lunak yang berkembang pesat, terminal tetap menjadi alat yang penting untuk para pengembang perangkat lunak, administrator sistem, dan pengguna teknis lainnya. Walaupun antarmuka grafis semakin mutakhir, terminal tetap memberikan fleksibilitas dan kemampuan yang sangat diperlukan untuk menjalankan tugas-tugas khusus dan otomatisasi dalam konteks komputer modern. Dengan kata lain, terminal masih memiliki peran yang tak tergantikan dalam dunia teknologi saat ini.

- **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.

- **Version Control Git**

  Kontrol versi adalah metode yang digunakan untuk melacak dan mengelola perubahan dalam kode sumber atau berkas proyek. Git merupakan salah satu sistem kontrol versi terdistribusi yang paling populer dan kuat.

  1) **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).

  2) **Sistem Kontrol Versi Terdistribusi (Distributed Version Control System)**

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

- **Installing GIT :**

  Git dapat di install dan dipakai pada berbagai macam Operating System seperti Windows, Linux, dan MacOS

- **Dasar – Dasar Command Git**
  1) git init: Memulai repositori GIT baru di direktori.
  2) git clone: Mengunduh repositori GIT yang ada dari URL ke direktori lokal.
  3) git add: Menambahkan perubahan file ke area staging untuk persiapan komit.
  4) git commit -m: Mengeksekusi komit untuk menyimpan perubahan yang sudah di-staging dengan pesan komit yang menjelaskan perubahan.

5) git status: Menampilkan status file dalam repositori, termasuk perubahan yang belum di-commit.
6) git branch: Menampilkan daftar cabang (branch) yang ada dalam repositori.
7) git checkout: Beralih ke cabang (branch) yang lain.
8) git merge: Menggabungkan perubahan dari satu cabang ke cabang yang lain.
9) git pull: Mengambil perubahan terbaru dari repositori jarak jauh (remote repository).
10) git push: Mengirimkan perubahan lokal ke repositori jarak jauh.
11) git log: Menampilkan riwayat komit dalam repositori.
12) git diff: Membandingkan perbedaan antara versi berbeda dari file.
13) git remote add: Menambahkan repositori jarak jauh dengan nama tertentu.
14) git remote -v: Menampilkan daftar repositori jarak jauh yang telah ditambahkan.

- **GIT Collaboration Simulation**
    1) **Menyiapkan Repository :** Buat repositori GitHub baru
    2) **Persiapan Lokal :** Mengklon repositori ke mesin local
    3) **Pembuatan Branch dan Alur Kerja :** Setiap peserta harus membuat cabang untuk pekerjaannya.
    4) **Melakukan Perubahan :** Peserta bekerja di cabang masing-masing dan Setelah melakukan perubahan, mereka harus melakukan commit dengan pesan commit yang deskriptif.
    5) **Mengirim Perubahan ke Remote :** Peserta mengirimkan branch mereka ke repositori remote
    6) **Mengelola Konflik :** Konflik dapat terjadi ketika dua atau lebih peserta membuat perubahan pada berkas yang sama di branch yang berbeda, Jika terjadi konflik, mereka harus menyelesaikannya secara manual. Setelah menyelesaikan konflik, maka harus melakukan commit perubahan dan push branch yang digabungkan
    7) **Pull Request dan Merge :** permintaan tarik (pull request) dari branch peserta ke branch utama (biasanya "main" atau "master"), pemilik repository (atau orang lain yang ditunjuk) dapat merge pull request ke branch utama melalui GitHub.
    8) **Memperbarui Repositori Lokal :** memperbarui repositori lokal untuk mencerminkan perubahan yang dilakukan di branch utama, Setiap peserta harus beralih kembali ke branch masing-masing dan menggabungkan branch utama ke cabang fitur peserta untuk stay up to date.

  Simulasi selesai ketika semua peserta berhasil mengirimkan perubahan mereka, menyelesaikan konflik melalui pull request, dan menggabungkan branch peserta ke branch utama.