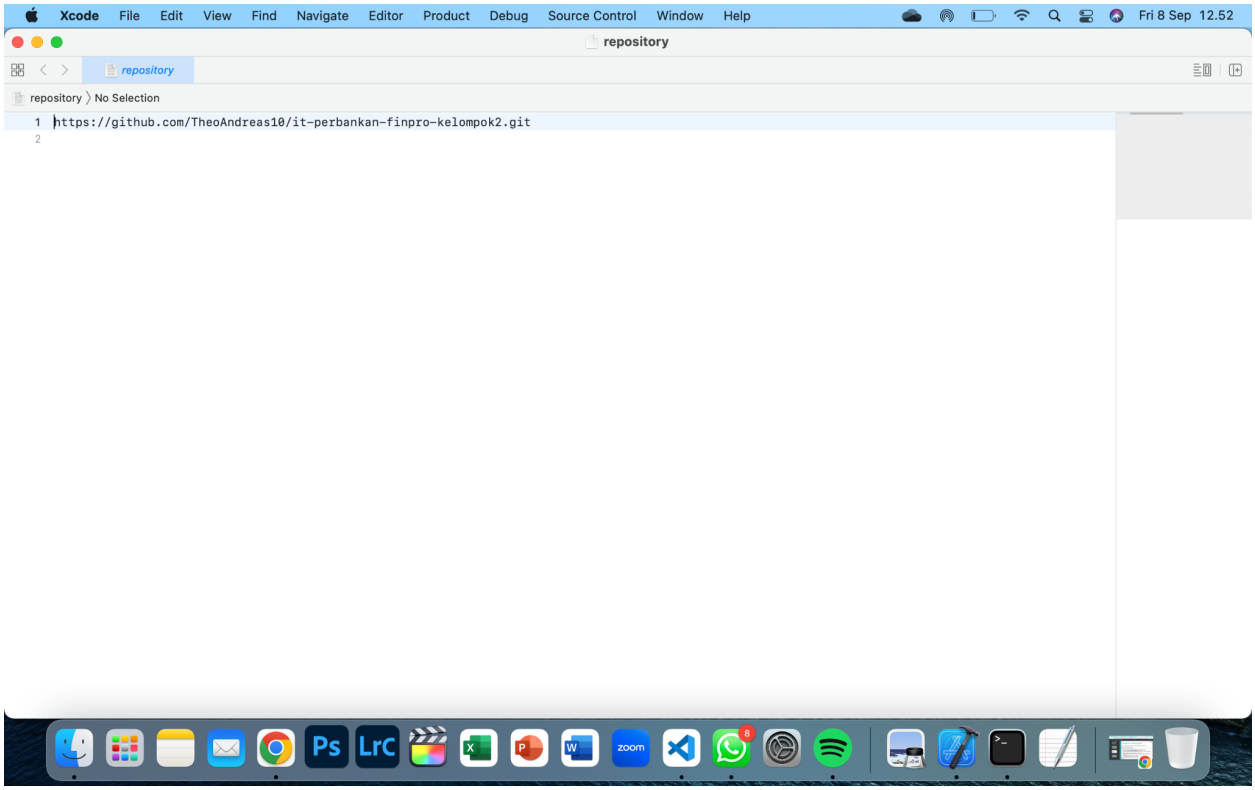
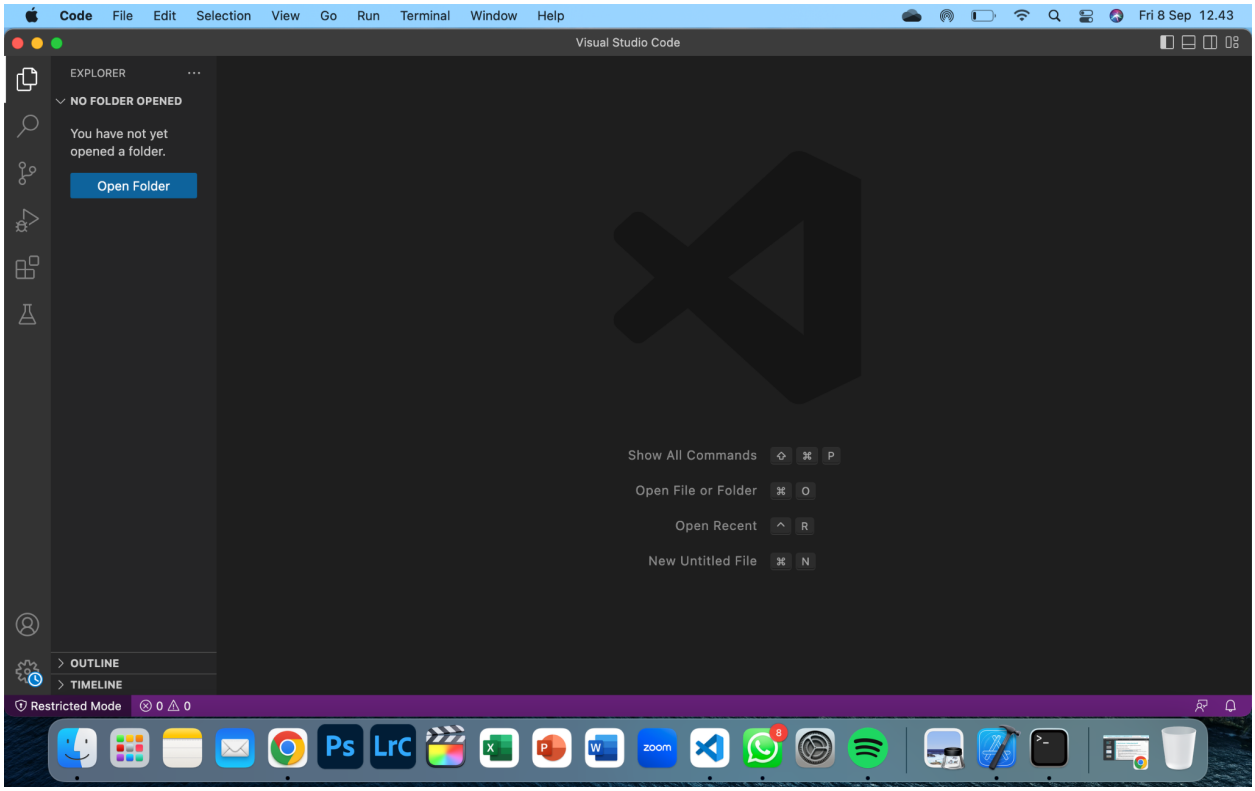

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Fri 8 Sep 12.52
repository
repository
repository 〉No Selection
1 https://github.com/TheoAndreas10/it-perbankan-finpro-kelompok2.git
2


Code File Edit Selection View Go Run Terminal Window Help
Fri 8 Sep 12.43
Visual Studio Code
EXPLORER
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
Show All Commands ⇧ ⌘ P
Open File or Folder ⌘ O
Open Recent ^ R
New Untitled File ⌘ N
OUTLINE
TIMELINE
Restricted Mode
⊗ 0 ⚠ 0

1. Fullstack Developer Career Path

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

Pengembangan Full Stack (FSWDM) merujuk pada pengembangan aplikasi secara end-to-end, dari sisi depan (front-end) hingga sisi belakang (back-end) dan, dalam beberapa kasus, hingga sisi klien (client-side). FSWDM melibatkan beberapa komponen penting seperti front-end development, back-end development, database management, integrasi antara front-end dan back-end, version control dan collaboration, serta mobile development. Beberapa teknologi dan framework yang populer dalam FSWDM termasuk React, Vue.js, AngularJS untuk front-end, ExpressJS, Laravel, Spring, Rails untuk back-end, dan Oracle, PostgreSQL, MySQL, MongoDB, Redis untuk database management.

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

Pengembangan Aplikasi End to End, merupakan pendekatan komprehensif yang mencakup seluruh siklus pembuatan aplikasi, dari perencanaan hingga pemeliharaan. Tahapan-tahapannya meliputi perencanaan, desain, pengembangan front-end, pengembangan back-end, integrasi dan pengujian, serta pemeliharaan dan peningkatan aplikasi.

Version control (pengendalian versi) adalah sistem yang memungkinkan pengembang melacak perubahan pada kode sumber aplikasi selama pengembangan. Alat version control yang populer dalam pengembangan Full Stack Web Development adalah Git dan Mercurial. Manfaatnya termasuk pencatatan perubahan, pencatatan riwayat, pemecahan konflik, dan kemampuan pemulihan yang mudah. Penggunaan version control melibatkan inisialisasi proyek, pengembangan paralel, branching untuk isolasi perubahan, merging untuk menggabungkan perubahan, dan penggunaan pull request untuk tinjauan oleh tim sebelum penggabungan. Dengan menggunakan pengembangan aplikasi end-to-end dan sistem pengendalian versi, tim pengembang dapat bekerja secara efisien dalam kolaborasi untuk menciptakan aplikasi yang berkualitas dan mudah dikelola.

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

1. IDE - Code Editor: Visual Studio Code
2. Version Control - Repository: GitHub, GitLab, dan Bitbucket
3. Version Control - Git Tools: Sourcetree dan GitLens
4. DBMS (Database Management Systems): PostgreSQL, MySQL, ORACLE, mongoDB, dan redis
5. API: POSTMAN dan swagger
6. Tests dan Debugging: Jest, Mocha, Chai, dan JUnit5
7. Mobile Development: React Native dan Flutter

8. Layanan Cloud: AWS (Amazon Web Services), Google Cloud, dan Azure
9. CI/CD (Continuous Integration/Continuous Deployment): Jenkins dan CircleCI
10. Desain UI/UX: Figma dan Sketch

2. SDLC & Design Thinking Implementation

- **What is SDLC**

Secara keseluruhan, SDLC (Siklus Hidup Pengembangan Perangkat Lunak) adalah suatu proses yang terstruktur untuk mengembangkan perangkat lunak dari awal hingga selesai. SDLC terdiri dari tahap-tahap seperti perencanaan, analisis, desain, pengembangan, pengujian, penerapan, dan pemeliharaan. Dalam penggunaan SDLC, terdapat beberapa manfaat seperti prediktabilitas dan pengendalian proyek, peningkatan kualitas perangkat lunak, pengelolaan risiko yang lebih baik, efisiensi tim dan kolaborasi, memenuhi kebutuhan pengguna, penghematan biaya dan waktu, meningkatkan pengawasan dan evaluasi, serta peningkatan dokumentasi. Dengan menggunakan SDLC, pengembangan perangkat lunak dapat dilakukan dengan lebih terstruktur, efisien, dan memastikan bahwa perangkat lunak yang dihasilkan sesuai dengan kebutuhan dan tujuan yang ditentukan.

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

Model-model SDLC adalah pendekatan yang berbeda dalam mengelola proses pengembangan perangkat lunak. Setiap model memiliki karakteristik dan penggunaan yang unik sesuai dengan kebutuhan proyek.
- *Waterfall Model* adalah pendekatan berurutan yang cocok untuk proyek dengan persyaratan yang stabil.
- *V-Shaped Model* menekankan pengujian tingkat tinggi dan kualitas.
- *Prototype Model* memfokuskan pada pemahaman kebutuhan pengguna melalui pembuatan prototipe.
- *Spiral Model* menggabungkan pendekatan spiral dengan inkremental untuk proyek besar dan kompleks.
- *Iterative Incremental Model* melibatkan pengulangan siklus kecil untuk proyek dengan anggaran dan waktu terbatas.
- *Big Bang Model* adalah pendekatan kurang terstruktur yang cocok untuk proyek kecil atau eksploratif.
- *Agile Model* adalah pendekatan kolaboratif dan fleksibel yang cocok untuk lingkungan yang berubah-ubah.

Pemilihan model SDLC yang tepat harus mempertimbangkan sifat proyek, persyaratan, dan lingkungan. Tidak ada satu model yang cocok untuk semua proyek, dan seringkali organisasi mengadaptasi atau menggabungkan model sesuai kebutuhan.

Penting untuk memahami setiap model dan menerapkannya dengan bijak untuk mencapai kesuksesan dalam pengembangan perangkat lunak.

- **Design Thinking Implementation**

Design Thinking adalah pendekatan kreatif yang berfokus pada pengguna dalam pengembangan perangkat lunak. Tahapan-tahapan dalam Design Thinking :
1. Empathize: Memahami kebutuhan pengguna dengan melakukan riset dan membangun pemahaman mendalam terhadap pengguna.
2. Define: Mendefinisikan masalah secara jelas dan menetapkan tujuan proyek berdasarkan hasil riset dan pemahaman.
3. Ideate: Menghasilkan berbagai ide kreatif untuk mengatasi masalah yang ditemukan.
4. Prototype: Membangun representasi nyata dari ide-ide yang dipilih untuk mendapatkan umpan balik dari pengguna.
5. Test: Mengumpulkan umpan balik dari pengguna untuk memvalidasi solusi-solusi yang diusulkan.
6. Implement: Mengembangkan perangkat lunak berdasarkan desain yang telah diuji dan divalidasi.

3. Basic Git & Collaborating Using Git

Terminal merupakan alat yang penting dalam dunia teknologi dan pengembangan perangkat lunak, meskipun antarmuka grafis semakin maju. Terminal memberikan fleksibilitas dan kemampuan otomatisasi yang vital dalam pengelolaan sistem dan tugas khusus di lingkungan komputer modern.

Git adalah salah satu sistem kontrol versi terdistribusi yang paling populer dan kuat. Kontrol versi adalah metode penting untuk melacak perubahan dalam kode sumber atau proyek, dan Git memungkinkan pengembang untuk melacak sejarah revisi, berkolaborasi dengan tim, dan mengelola kode dengan efisien. Dasar-dasar command/perintah Git, seperti :
-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.

Simulasi kolaborasi menggunakan Git dengan tiga orang melibatkan langkah-langkah mulai dari persiapan repository, pembuatan cabang, melakukan perubahan, mengatasi konflik, hingga penggabungan perubahan dengan permintaan tarik (pull request). Ini mencerminkan bagaimana Git memfasilitasi kolaborasi tim dalam pengembangan perangkat lunak. Dengan pemahaman tentang Git dan penggunaannya, pengembang perangkat lunak dapat bekerja secara lebih efisien, melacak sejarah perubahan, berkolaborasi dengan tim, dan mengelola proyek dengan lebih baik. Git telah menjadi alat yang esensial dalam dunia pengembangan perangkat lunak modern.