풀스택 엔지니어링 개론

업무 준비

업무 메일 및 메신저 확인

업무 메일 확인은 수신함의 메일을 확인하고 중요한 매일에 우선적으로 회신

스팸 및 불필요한 메일을 삭제하고 중요한 메일은 분류 및 정리

라벨링을 통해 메일함을 분류해서 관리 깔끔하게 관리

업무 메신저 확인

팀원 및 프로젝트 관련 메시지를 확인하고 미응답 메시지에 답변

중요 메시지를 확인하고 필요한 조치를 취함

키워드로 자동으로 알림이 올 수 있도록 한다

업무 계획

업무 진행 상황 검토

현재 진행 중인 프로젝트와 작업의 진행 상황을 검토

완료된 작업과 진행 중의 작업을 명확히 구분

진행 중인 작업의 경우 마감 일정에 맞출 수 있는지 검토

업무 범위 조율

업무 마감 일정 조율

개발 인원 추가 투입

우선 순위 결정

메일 메신저 오늘 목표 우선순위를 매김

긴급한 업무 중요한 업무를 구분

프로젝트 마감일과 주요 마일스톤을 고려하여 일정을 조정

1순위 중요하고 급한 일 2순위 중요하지만 급하지 않음 3순위 급하지만 중요하지 않음 4순위 급하지도 중요하지도 않음

업무 목록 작성

업무 계획과 우선수위에 따라 오늘 해야 할 작업을 구체적으로 나열

각 작업의 예상 소요 시간을 추정, 일정에 반영

작은 단위로 업무를 나누어 처리의 효율성을 높임

업무 소요시간을 측정하기 위해 Time Tracker를 사용하는 것을 권장

데일리 스탠드업

업무 진행 사항을 공유

진행 사항 공유

각 팀원은 자신이 담당한 작업의 진행 상황을 간략하게 공유

완료된 작업, 진행 중인 작업 , 다음에 진행할 작업 등을 언급

업무 이슈 공유

진행 중인 작업에서 발생한 문제나 도움이 필요한 부분에 대해 공유

이슈의 중요도와 긴급성을 파악하여 해결책을 찾기 위해 논의

해결책 논의

공유된 업무 이슈에 대해 팀원들이 함께 해결 방법을 논의

데일리 스탠드업을 통해 업무 관리 스킬을 배울 수 있다

기능 설계

나의 업무 목표: 매일 발신 시스템 개발

요구 사항 확인 및 분석

엔지니어에게 요구 사항 분석 역량이 요구되는 이유

문제 해결 능력 강화

요구 사항 분석은 사용자의 문제를 식별하고 해결책을 제시하는 과정

엔지니어는 이를 통해 문제 해결 능력을 강화하고, 효율적이고 실용적인 솔루션을 찾을 수 있다.

협업의 중요성 강조

기획자의 원활한 협업을 위해 요구 사항 분석 역량은 필수적입니다.

엔지니어는 기획자와의 소통을 통해 요구 사항을 명확하게 이해하고, 최종 사용자의 기대를 충족시킨다

기능의 목표 및 목적 분석

목표의 명확성

왜 필요한지, 고객이 어떤 가치를 제공하는지 명확하게 정의

기능의 목표를 분명하게 설정

기능 목표에 부합하지 않는 요구 사항은 프로젝트에 포함 X

사용자의 요구 충족

기능의 목표는 최종적으로 고객의 요구를 충족시키기 위함

따라서 고객의 입장에서 요구 사항을 분석하고, 각 기능이 사용자의 요구를 충족시키는지 명확하게 이해해야 함

상세 요구 사항 정의

유스케이스 정의

각 기능의 사용 시나리오를 유스케이스로 작성하여 기능의 상세한 동작을 설명

유스케이스는 시스템이 사용자 또느 다른 시스템과 상호작용하는 방식을 설명하는 기법

시스템의 특정 기능이나 서비스를 사용자가 어떻게 사용하는지

구성 요소

액터

시스템 외부에서 시스템과 상호작용하는 주체로, 주로 사용자나 다른 시스템을 의미

목표

목적을 명확히 정의

흐름

실행 순서를 설명하여, 주요 단계와 각 단계에서의 액터와 시스템 간 상호작용을 포함

대안 시나리오

예외적인 상황을 설명하고, 이를 어떻게 처리할지 설명

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

장점

명확한 요구사항

사용자 중심 설계

사용자 관점에서 시스템의 기능을 설명하기 때문에 실제 필요한 반영하는 설계를 도움

테스트와 검증

협업 및 커뮤니케이션 도구

시스템 아키텍처 설계

시스템의 구성요소 식별

기능 개발을 위해 필요한 구성 요소 작성

구성 요소 간의 데이터 흐름과 상호 작용을 적의

구성 요소 세부 설계

UI

인증 시스템

데이터베이스

API 서버

성능 최적화

로그 및 모니터링

개발

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

개발을 위한 새로운 브랜치 생성

Git flow

Master(출시됨) release(베포) develop(개발용) feature(기능 브랜치) hotfix(긴급) 브랜치 사용하여 업무 흐름에 따라 안정적으로 릴리즈 관리 기능

Github Flow

마스터와 기능 개발 브랜치만 사용하여 단순하고 직관적으로 브랜치를 관리

Gitlab Flow

Master production, pre-production, feature 만 사용

Git flow 와 Github Flow의 중간 절충안으로 봄

프론트엔드 개발

UI/UX 구현

디자인 시안 확인

FIGMA, FRAMER 등의 디자인 도구를 통해 디자이너가 제작한 전체적인 UI/UX 디자인과 구성 요소를 파악합니다.

기능 요구 사항에 맞게 페이지가 디자인 되었는지 검토

색상, 폰트, 간격 등 디자인 가이드라인을 확인하고 프로젝트에 반영

화면 구조 설계 및 개발

컴포넌트 구조 설계

재사용 가능하도록 컴포넌트를 설계 하고 구조를 정의합니다

컴포넌트를 정의 할 때, 각 HTML 요소에 시맨틱한 태그를 사용

페이지 레이아웃 설계

각 페이지의 레이아웃을 정의하고 주요 컴포넌트의 배치를 설계

컴포넌트 및 페이지 개발

각 컴포넌트 개발하고 스타일을 추가

개발한 컴포넌트를 페이지의 적절한 위치에 배치

태블릿 및 모바일에도 UI/UX가 잘 표시될 수 있도록 유의합니다.

상태관리

상태 관리 라이브러리를 사용하여 애플리케이션의 상태를 관리

기능 개발

요구 사항에 따라 각 UI/UX 필요한 ACTION 기능을 개발하여 연동

필요한 API 호출을 추가합니다.

사용자 입력 처리, 데이터 검증, 비동기 작업 등을 포함하여 완전한 기능을 구현

브라우저 렌더링 최적화

페이지 성능 테스트 LIGHT HOUSE 같은 도구를 사용하여 성능 테스트를 수행하고, 개선할 부분을 확인

최적화 기법 적용: 브라우저 렌더링 성능을 최적화 하기 위해 다양한 기법을 적용

렌더링 최적화: React의 경우 컴포넌트를 적절하게 분할하고, useCallback, useMemo 등을 활용하여 렌더링 과정을 최적화

코드 분할: Webpack 과 같은 도구를 사용하여 코드를 분할하고 필요한 부분만 로드

이미지 최적화

캐싱 전략

레이지로딩

텍스트, 스크린샷, 소프트웨어, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

useCallback을 사용해서 항상 불러오는 게 아니라 원한는 상태에서만 불러온다 예시 onDataFetched 가 새로운 값이AP 될때만 된다

백엔드 개발

데이터베이스 선택

테이블 / 컬렉션 설계

필요한 데이터 테이블 혹은 컬렉션을 설계하고 관계를 정의

데이터 조회 및 조작을 위한 쿼리를 정의

API 엔드포인트 개발

API 명세서 확인

엔드포인트 설계: RESTful 또는 GraphQL을 기반으로 API 엔드포인트를 설계하고 구현

인증 및 보안: 사용자 인증 및 권한 부여 메커니즘을 구현하여 데이터가 안전하게 관리 될 수 있도록 설정합니다.

API 문서화: OpenAPI 등 API 문서화 도구를 사용하여 API를 문서화하고 사용 방법을 정의

비즈니스 로직 구현

서비스 계층 구현: 비즈니스 로직을 처리할 서비스 계층을 구현

데이터 처리: 데이터베이스에서 데이터를 CRUD 로직을 구현

성능 최적화

데이터베이스: 쿼리 최적화를 통해 데이터 검색 및 처리 속도를 향상 필요한 인덱싱과 캐싱을 적용하여 데이터 접근을 최적화

비즈니스 로직

사용되는 알고리즘을 분석하고 최적화하여 실행 시간을 단축하거나 자원 사용을 최소화

병목 현상이 발생할 수 있는 부분은 비동기 방식으로 처리하여 시스템의 응답성을 향상

코드 자체의 효율성을 개선하여 불필요한 반복문, 메모리 누수 등을 방지

유닛테스트

유닛 테스트의 목적: 개발된 각 기능이나 모듈이 예상대로 동작하는지 검증하며, 코드의 품질을 유지하고 버그를 사전에 발견

코드 커버리지: 테스트가 커버하는 코드의 범위를 측정하고, 코드 커버리지를 높이는 방법을 고려하여 모든 코드 경로가 테스트되었는지 확인

테스트를 CI/CD통합: 지속적 통합 및 지속적 배포 파이프라인에 유닛 테스트를 통합하여 자동환된 테스트 실행을 설정

코드리뷰

품질 개선: 다양한 관점에서 코들르 검토하여 버그와 품질 문제를 사전에 발견

지식 공유: 팀멤버 간의 지식 공유와 코드 베이스에 대한 이해도를 증가 시킬수 있다

학습 기회: 다른 개발자들의 접근 방식과 최신 기술을 배우고 개선할 수 있는 기회를 제공

팀 협업 강화: 피드백을 통해 협업 강화

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 소프트웨어, 멀티미디어 소프트웨어, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

코드 리뷰 받기

리뷰어는 PR의 변경 사항을 상세히 검토하여 코드 스타일, 기능 구현, 성능 보안, 테스트 커버리지 등을 점검

리뷰어가 남긴 피드백을 확인하고 필요한 수정 사항을 반영하고 피드백에 대한 답변을 남김

브렌치 관리 전략에 따라 브랜치 병합

코드가 모든 테스트에 통과되고, 안정적인 상태임을 확인했다면 브랜치를 병합

병합이 완료된 후, 메인 브랜치의 상태를 검증합니다.

병합이 완료되면, 사용한 기능 브랜치를 삭제하여 브랜치 목록을 정리

QA(테스트)

통합 테스트 /E2E테스트

개별적으로 개발한 요소가 통합되어 있는 상태에서 테스트하는 과정

운영 환경과 유사한 테스트 환경 설정

유스케이스에 따라 테스트케이스 작성

테스트 수행

코드 수정

버그 수정 및 성능 최적화

버그 재현 방법/발생조건/예상 결과/실제 결과 기록

버그의 원인 분석

코드 수정 및 재검증

변경사항을 브랜치에 반영

성능 최적화

성능 분석 및 병목 지점 식별

병목 지점 개선

성능 재검증

변경 사항을 브랜치에 반영

배포

최종 제품을 사용자에게 제공하는 중요한 과정

Main/Realse 브랜치

테스트를 모두 통과하여 릴리즈가 준비가 완료된 상태를 유지

개발 및 테스트가 완료된 개발 내용을 메인 브랜치에 병합

코드가 배포되기전에 최종 점검을 수행

배포 파이프라인 설정하기

CI 코드 변경 사항이 메인 브랜치에 병합될 때마다 자동으로 빌드 및 테스트가 실행

CD 빌드 및 테스트가 성공하면, 자동으로 배포 프로세스가 시작됩니다.

소스코드 빌드 -> 유닛테스트/통합 테스트 -> 배포 분지 ->배포!

회고

디자인 도구를 통해서 회고 작성 및 포스트로 작성