팀프로젝트 완료 보고서

김민재

1. Overview (프로젝트 개요)

● 프로젝트 명칭 및 선정 이유

○ 프로젝트 명칭 - 1300k - 독서 기록 앱

○ 선정이유 - 독서를 좋아하는 사용자를 위해 언제 어디서든 기록하며 다른 사람들과 공유할 수 있는 앱을 만들고자 선정

● 프로젝트 기간

○ 2023/12/01 ~ 2023/12/21

● 프로젝트 목적

1. api를 활용하여 데이터를 출력한다. 2. 외부 데이터 사용을 위해 node.js를 활용 해 서버를 구축하여 api를 호출한다

3. 출력 된 data를 이용하여 도서를 기록, 저장한다. 4. 도서 기록 시 닉네임, 작성 시간, 한줄 리뷰, 책 표지를 보여주는 화면을 출력한다. 5. cors 이슈를 해결한다. 6. 컴포넌트 간에 정보 전달 및 연결을 성공한다.

▶ 프로젝트 구조

○ 디렉토리 구조

```
src——sever.js
|
├—— components
|   ├—— feed
|   ├—— footer
|   ├—— library
|   ├—— login
|   ├—— mainhome
|   ├—— mybook
|   └—— search
|
├—— context
|   ├—— FeedContext.js
|   └—— FeedHighContext.js
|
├—— page
|   ├—— Feed.js
|   ├—— Login.js
|   ├—— Mainhome.js
|   ├—— MyBook.js
|   └—— Search.js
|
|
├—— app.js
```

├── app.css
├── index.js
├── index.css
└── LocalStorage.js

● 각 담당 업무

리더 : 장원준 - github관리, 리더

팀원 : 김민재 - 발표,파일 및 자료관리

팀원 : 권지민 - notion 관리, 작업 기록자

● 협업 및 의사 소통

○ figma

- 실시간 공동 작업으로 ui/ux 디자인 작업 및 의견 조율
- 작업물에 대한 피드백

○ notion

- 회의록 및 문제사항 기록
- 공통 문서를 사용해 요구사항 및 가이드라인 공동 작업
- 참고 사이트 정리

○ discord

- 실시간 화면 공유를 통해 팀원들과 문제점 발생 시 실시간 공유

● 프로젝트 기대 효과

○ 기존 페이지에서 사용자 편리성 향상을 위해 더욱 편리한 방향으로 수정 및 기능 추가

2. 프로젝트 범위

● 고려사항

Front-End

- 언어: JavaScript (React)
- 프레임워크: React.js
- 모듈화된 CSS인 module.css 사용

Back-End

- Node.js, Express, Axios 활용한 서버 개발
- 엔드포인트, GET 및 POST 메소드 이해
- mongoDB 연결을 통한 데이터 저장

Data-Base

- CORS 이슈 해결 및 데이터베이스 통신 원활하게
- Open API 사용법 숙지 및 데이터 가공 후 내보내기 계획

● 각 페이지별 요구사항 분석
- 기능적 : 보여지는 부분에 대한 주요 기능
- 비기능적 : 사용자의 편리성을 위해 추가된 작업
● 페이지 기능적 요구사항
SPLASH: 페이지 로딩 화면으로 3초가 지나면 자동으로 로그인 페이지로 이동
LOGIN: 특정 아이디와 비밀번호를 설정해서 로그인이 가능, 카카오톡 로그인 API를 받아와 실제 카카오톡 아이디 로그인 기능 구현
MAIN: 우측 상단에 토글 메뉴바를 만들어 각 페이지 이동 하단부분 박스 버튼을 누르면 내서재 페이지로 이동
SEARCH: 검색창에 읽고싶은 책명 검색. SEARCH LIST: 검색창에 책명을 검색하면 관련된 책들이 나옴. SEARCH INFO: 관련된 책들이 클릭하면 그 책에 대한 정보들이 나옴
SEARCH SAVE: 저장하기 버튼을 누르면 읽은 책, 읽고 있는 책, 읽고 싶은 책 세가지로 나눠 책들을 저장할 수 있음. MY BOOK: 검색 저장 부분에서 저장했던 책들이 3가지로 나눠서 사용자에게 보이게 함. FEED: 내서재에서 저장한 책들을 선택해서 기록하고 저장할 수 있는 페이지. LIBRARY: 읽을 책을 추천 받고 싶을때 베스트셀러,주목할 만한 신간, 신간 책들을 볼 수
있는 페이지. ● 페이지 비기능적 요구사항
SPLASH: 페이지 로딩은 3초 이내에 완료
LOGIN: 안전한 토큰 기반의 인증 방식을 사용
MAIN: 토글 메뉴바, 내 서제 이동 페이지 : 빠르고 쉽게 이동할 수 있게 구현. SEARCH: 검색창에 책명을 입력하고 결과를 확인하는 과정은 빠르게 이루어지게 구현. SEARCH LIST: 검색 결과는 정확하게 나와야 하며, 사용자에게 신속하게 제공되게 구현. SEARCH INFO: 데이터의 정확성과 빠른 응답 속도
SEARCH SAVE: 저장된 책 목록은 정확하게 분류되어 있어야 하며, 저장된 정보는 안전하게 보호되게 구현. MY BOOK: 검색 저장 부분에서 저장한 책들이 각각 읽은 책, 읽고 있는 책, 읽고 싶은
책으로 나눠서 사용자에게 효과적으로 보여지게 구현. FEED: 내서재에서 선택한 책들을 기록하고 저장하는 기능은 안전하게 이루어지게 하고, 사용자에게 투명하게 제공
LIBRARY: 베스트셀러, 주목할 만한 신간, 신간 책 추천은 정확하게 이루어져야 하며, 사용자에게 흥미로운 도서를 제안할 수 있게 구현. 3. 작업 순서
● Front-DB-Back-End
4. 사용기술
-프로젝트 관리 Tool : GitHub, Notion
-개발 Tool : Visual Studio Code, Figma
-사용 언어 : React,Css,Node.js,MongoDB
- 알라딘 API 선정 이유
- 10자리 13자리 isbn을 개별 제공
- 상세 검색시 페이지 정보 제공
- 도서별 p(페이지) 제공
- 베스트셀러 및 다양한 카테고리 형태의 자료 제공
5. 프로젝트 내용

- 초기 수집한 요구사항 요약
 - 참고 app 목록
 - 밀리의서재
 - 북적북적
 - 리딩트리
 - 필수 요구사항
 - 도서 API를 연결해 도서 검색 데이터 추출
 - TodoList 기능을 통한 기록창 구현
 - 다수페이지 간의 데이터 전송
- 프론트엔드에서 구현 필요시 하는 목록
 - 목록
 1. API 통합:
 - Axios를 사용하여 알라딘 Open API에 요청을 보내고 도서 정보를 받아옴.
 - CORS 이슈를 해결하기 위해 백엔드에 설정된 CORS를 확인하고 필요한 조치를 취함.
 2. 도서 정보 표시:
 - 받아온 도서 정보를 화면에 표시하는 컴포넌트를 구현.
 - 도서의 제목, 저자, 표지 등을 사용자에게 보여줌.
 3. 도서 저장소 및 이동:
 - 읽은 책, 읽고 싶은 책, 읽고 있는 책의 세 가지 저장소를 구현.
 - 선택한 도서를 각 저장소로 이동시키는 기능을 추가.
 4. 로컬 저장소 활용:
 - localStorage를 이용하여 프론트엔드에서 도서 저장소를 구현.
 - 저장소 간의 데이터 이동 및 로컬 저장소에 데이터를 저장하는 로직을 구현.
 5. React Router 사용:
 - react-router-dom을 사용하여 여러 페이지 간의 라우팅 구현.
 - 각 페이지에서 필요한 데이터를 전달하고, 페이지 간에 이동하는 기능 추가.
 6. 사용자 인터랙션 구현:
 - 도서를 클릭하면 도서의 상세 정보를 보여주는 페이지로 이동하는 기능 추가.
 - 저장소 간 이동이나 저장, 삭제 등의 사용자 액션에 대한 버튼 또는 링크 추가
 7. 스타일링 및 UI/UX 개선:
 - CSS 또는 스타일링 라이브러리를 사용하여 사용자 인터페이스 디자인.
 - 사용자 경험을 향상시키기 위한 애니메이션 또는 효과 등 추가.
 8. 에러 핸들링:
 - API 요청 중 발생할 수 있는 에러에 대한 핸들링 구현.
 - 사용자에게 적절한 메시지를 표시하여 오류에 대한 투명성 확보.
 9. 페이지 네비게이션 구현:
 - 여러 페이지 간의 네비게이션을 가능하게 하는 메뉴 또는 탭 구현.
 10. 테스트:
 - 유닛 테스트 또는 통합 테스트를 작성하여 안정성 확보. - 작업시 추가 및 변경된 요구사

항
 - 내서재 데이터 삭제 버튼
 - 기존: 도서 클릭 시 삭제 모달창 출력
 - 변경 : 도서 클릭 시 이동된 해당 도서 페이지 안으로 삭제 버튼 출력
 - UI/UX 구조 변경 및 디자인 수정
 - 기존 : 디자인을 중점으로 헤더 및 푸터 요소들에 ISO 가이드라인 미적용
 - 변경 : 기준점으로 잡았던 13pro ISO 가이드라인에 맞춰 수정.
 - 글꼴 변경
 - 기존: SpokaHansanse체를 적용 후 폰트의 깨짐이 발견.
 - 변경 : NotoSansKR 폰트로 변경.
 - css 변경
 - 기존: 기존 css로 작업을 하였지만 페이지 병합시 css겹침으로 인한 문제 발생.
 - 변경: module.css로 각 클래스를 정의하여 겹침 문제 해결. - **주요 기능 구현 설명**
 - Open API 책 정보 추출
 - Aladin Open API를 활용하여 TTB 키를 획득하고 API 링크를 추출하여 도서 정보를 가져옴.
 - CORS(Cross-Origin Resource Sharing) 문제를 CORS 라이브러리를 이용하여 Node.js 및 Express 환경에서 Axios를 활용해 해결.
 - 알리딘 API 가이드를 참고하여 서버에 적용시켜 구현
 - Kakao Login API 연동
 - 카카오톡 로그인 API를 사용하기 위해 JavaScript 키를 발급받아 프로젝트에 적용
 - JavaScript 키를 사용하여 카카오 서버에 안전하게 인증을 요청하고, 사용자에 대한 권한을 관리
 - 받아온 토큰을 안전하게 저장하고 관리하여 사용자가 로그인 상태를 유지할 수 있도록 구현
 - 데이터 처리
 - 추출된 도서 고유의 id값으로 링크를 설정하여 정해진 id값의 정보를 추출
 - 읽은책, 읽고싶은책, 읽고있는책 3가지 저장소간의 데이터 이동 구현
 - 각 태그별 3가지로 로컬저장소를 나눠서 구현
 - 3개의 저장소를 DetailPage,New,List 등 각 고유의 저장소로 옮겨 글을 작성함
 - 작성한 글은 DetailPage만의 저장소에 보관하여 저장
 - LocalStorage
 - 저장소를 구현하기 위해 로컬스토리지를 이용
 - const name을 이용하여 여러 저장소를 생성하여 구현
 - getItem,setItem에 대한 이해
 - key,value에 대한 이해
 - Server
 - PORT,Proxy 설정에 대한 이해
 - cors,express 라이브러리를 설치해 서버 구현에 성공
 - app.get, post, async 등을 이해
 - 서버 엔드포인트를 이해하여 하나의 서버파일안에 여러 정보를 추출

- /site , /site2 서버 주소의 끝에 고유한 링크를 설정하여 추출
- React-router-dom
- 데이터 이동을 위해 react-router-dom 라이브러리 활용.
- <Router>를 이용하여 책 정보의 고유 ID값 전달 (예: /book:id).
- 링크를 통한 페이지 간 이동, 저장, 삭제 등의 이해. 6. 결과

● 문제 및 해결

○ cors 이슈

- 잘못된 점:
알라딘 API를 활용하여 정보를 가져오는중 CORS발생
브라우저에서 직접적인 API 호출이 차단 되어 정보를 가져올 수 없었음. - 해결 방안:
EXPRESS 서버를 도입하여 클라이언트가 EXPRESS 서버를 통해 알라딘
API의 정보를 받기 위해 간단한 서버 구축과 CORS 미들웨어를 활용해
모든 도메인의 요청을 허용하도록 설정 - 결과: O

○ api 글자 수 제한

- 잘못된 점:
알라딘 오픈 API를 사용해 책 정보를 가져오는 중 글자 수 제한 코드가
적용 X
불러온 API 값이 문자열이 아닌 다른 데이터 형식이라 슬라이스 태그를
넣어도 적용 X
- 해결방안:
디버깅 과정을 통해 데이터 형식 확인 후 투스트링 태그를 사용해 데이터
형식을 문자열로 반환 이후 슬라이스 태그를 같이 사용해 글자 수 제한
- 결과 : O

○ api 출력을 위한 server 구조 문제

- 잘못된 점:
API를 통해 추출하는 데이터에 대한 태그가 많을 경우 NODE 사용
미숙으로 문제 발생 추출할 데이터를 변경 시킬때마다 URL을 수정해서
새로 열어 데이터를 가져오기만 해서 원하는 태그에 따라 SERVER가
많아져야 하는 것에 대한 고민
- 해결 방안:
학원 선생님께 질문을 해서 하나의 SERVER에서 작업 할 수 있다는
피드백을 받음 GET을 사용해 경로를 지정할 경우 하나의 SERVER를
사용해도 다수의 태그에 결과값을 불러 올 수 있다는 걸 깨닫고 실행
하면서 문제를 해결
- 결과: O

○ 카카오톡 자동로그인

- 잘못된 점:
로그인 페이지에서 카카오톡 로그인을 이용해서 로그인을 하고
로그아웃을 하고 다시 로그인을 하는 과정에서 로그인 창이 뜨지 않고
로그인했던 아이디로 자동 로그인 문제
- 해결 방안:

로그아웃 버튼을 누르면 로컬 스토리지와 쿠키 삭제 코드를 적용 시킴
카카오톡 로그아웃 함수도 적용 로컬과 쿠키는 삭제를 했지만
로그인페이지에 정보가 그대로 남아있어 경험과 지식 부족으로 문제를
해결하지 못함. - 결과: X
7. 테스트 및 수정사항
- 테스트 결과 및 오류
1. 내서재 페이지
1. 같은 isbn값을 가진 도서가 읽은 책과 읽고 있는 책에 각각 저장되어 있을 경우
한 쪽에서 메모를 작성하면 두 페이지에 모두 메모가 저장
- 수정 필요사항 및 보안점
1. 게시글 수정 기능:
- 사용자가 다른 사용자의 게시글을 수정할 수 있는 보안 취약성이 있는지 검토
필요.
- 수정된 게시글이 유효한지 확인하는 서버 측의 검증 메커니즘이 필요함.
- 적절한 권한 확인 없이 게시글을 수정하는 시도에 대한 보호 필요.
2. UI 디자인:
- 사용자 경험을 향상시킬 수 있는 UI/UX 개선 사항을 고려해야 함.
- 모바일 및 데스크톱 등 다양한 플랫폼에서의 사용을 고려하여 반응형 디자인 적용
필요.
- 색상, 레이아웃, 아이콘 등 시각적인 디자인 측면에서 피드백 수렴 필요. 8. 프로젝트 완료 리뷰

잘한 부분
효과적인 UI/UX 디자인:
프로젝트의 사용자 경험을 향상시키기 위해 사용자 친화적이고 직관적인 디자인을 구현했습니다. 간결하면서도 효과적인 UI 요소들을 선택하여 사용자가 앱을 쉽게 이해하고 활용할 수 있도록 했습니다.

리액트 활용 능력:
프로젝트에서 리액트를 효과적으로 활용하여 컴포넌트 기반 아키텍처를 구현하고, 상태 관리와 라우팅을 효과적으로 다뤘습니다. 이는 유지보수성을 높이고 확장성 있는 코드를 작성하는 데 기여했습니다.

도서 관리 기능 구현:
도서 앱의 핵심인 도서 관리 기능을 잘 구현했습니다. 사용자들이 도서를 검색하고 추가, 삭제할 수 있도록 하는 기능을 효과적으로 개발하여 앱의 핵심 기능을 강화했습니다.

코드 품질 및 주석:
코드의 가독성을 높이기 위해 적절한 주석을 추가하고, 코드 스타일에 일관성을 유지했습니다. 이는 협업과 유지보수를 용이하게 만들어 주었습니다.

못한 부분 및 경험 부족
카카오톡 로그인 API 자동로그인 기능 미구현:
카카오톡 로그인 API를 활용한 자동로그인 기능을 구현하지 못했습니다. 이 부분에 대한 어려움을 극복하지 못해, 사용자 편의성 측면에서 부족한 부분으로 남아있습니다. 앞으로 API 통합에 대한 심층적인 이해와 구현 능력을 향상시키는 것이 필요합니다.

지식 및 경험 부족:
프로젝트 도중에 마주한 어려운 기술적인 문제들과 부족한 경험으로 인해 일부 기능을 원하는 대로 구현하지 못했습니다. 앞으로는 더 폭넓은 학습과 실전 경험을 쌓아가며 이러한 난관을 극복하는 능력을 키우는 것이 중요합니다.

향후 계획
기술적인 역량 강화:
사용한 기술 스택에 대한 깊은 이해를 바탕으로 자신만의 독립적인 문제 해결 능력을 키우고, 더 다양한 외부 API와의 통합에 대한 능력을 향상시킬 것입니다.

팀 협업 능력 강화:
팀 프로젝트에서의 협업 경험을 토대로 팀원들과의 원활한 소통과 협업 능력을 강화하여 효율적인 팀 프로젝트 진행을 도모할 것입니다.

프로젝트 완성도 향상:
현재 프로젝트의 부족한 부분을 보완하여 완성도를 높이고, 사용자 피드백을 수용하여 더 나은 버전의 앱을 개선해 나갈 계획입니다.

이번 프로젝트를 통해 얻은 경험과 부족한 부분을 극복하기 위한 노력을 통해, 앞으로의 프로젝트에서 더 나은 성과를 이루고자 합니다.