프로젝트

BOOK REVIEW

작성자: 정우성(WEARE2415@NAVER.COM)

CONTENS

BOOK REVIEW

CONTENS 01

프로젝트 배경

선정하게 된 이유

사용자가 책을 즐겨찾기에 추가하면, 알고리즘이 그 책의 장르, 저자, 주제를 분석하여 사용자의 취향을 파악합니다. 그리고 다양한 책의 리뷰로 통해 더 쉽게 다가갈수 있게 할려고 선정하게 되었습니다

자들이 책에 대해 솔직하고 진정성 있는 리뷰를 공유할 수 있는 공간을 마련하여, 과도한 광고나 상업적 리뷰에 의존하지 않고 독서 경험을 공유하도 합니다

개인의 독서 이력을 바탕으로, 관심 분야와 취향에 맞는 책을 추천하는 시스템을 구축하여 독자들이 새로운 책을 발견할 수 있도록 돕습니다.

다양한 책 리뷰는 독자들이 서로 다른 장르와 주제에 대해 깊이 있는 의견을 나누고, 다양한 관점에서 책을 분석을 하며 더 쉽게 접근할 수 있습니다

CONTENS 02

개발 환경 및 라이브러리

Front-end

REACT

HTML

CSS

JS

NODE.JS

라이브러리

AXIOS

MOMENT.JS

REACT-RATING

REACT-ROUTER-DOM

STYLED-COMPONENTS

CONTENS 03

나의 역할

리뷰 기능 구현

리뷰 추가 및 수정, 리뷰 목록, 리뷰 삭제, 책 상세정보(GOOGLE BOOKS API)를 통해 가져와 정보를 표시합니다

전체적인 구조 틀 구현

컴포넌트별로 리뷰 관리, 책 정보, 즐겨찾기 기능을 분리하여 구조화했습니다. 사용자가 책 정보를 확인하고 기능을 자연스럽게 설계 하였습니다.

GOOGLE BOOKS API 연동

GOOGLE BOOKS API를 통해 신간 도서 목록을 불러오고, 각 책의 상세 정보를 API로 받아와 화면에 표시하는 기능을 구현했습니다.

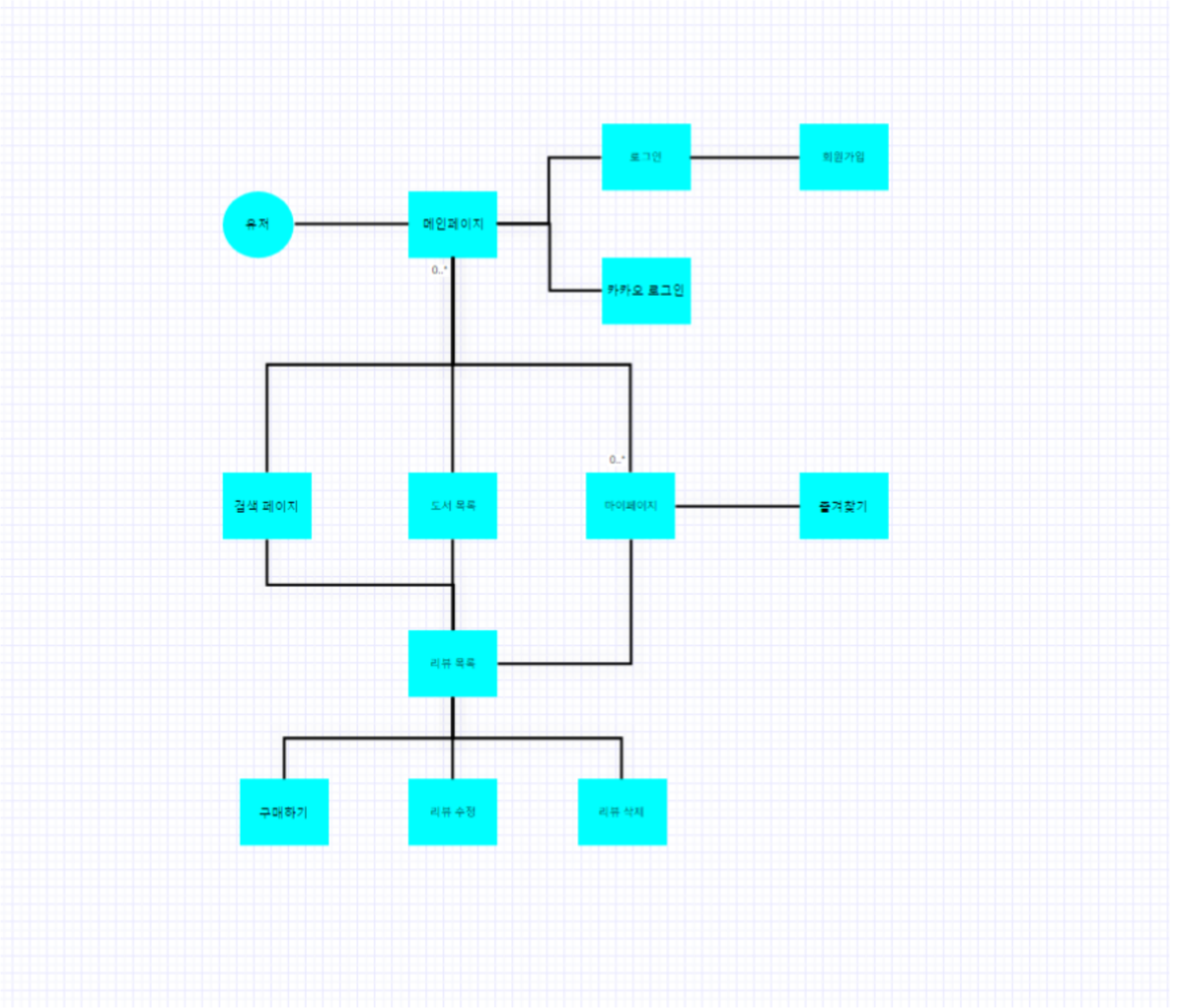
렌더링, CONTEXT API 최적화

컴포넌트 재렌더링 시 함수가 불필요하게 생성되지 않도록 최적화했습니다. 이를 통해 컴포넌트가 빠르게 렌더링되며, 특히 리뷰 추가, 수정, 삭제 등의 작업에서 성능을 개선해 UI 반응 속도를 높였습니다.

CONTENS 04

기능 구현

유저 및 도서 목록 계층 구조



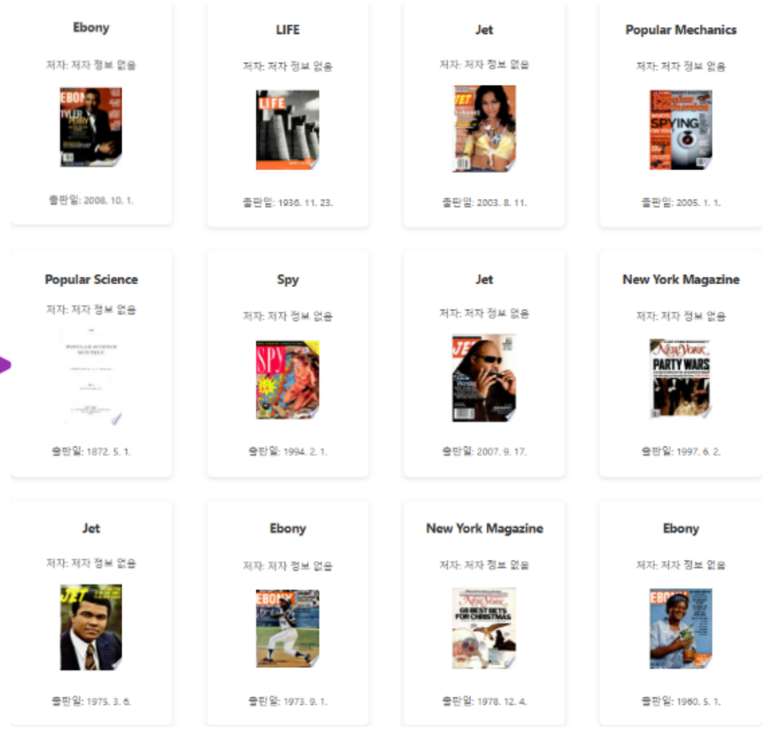
CONTENS 04

기능 구현

Google Books API

```
const fetchBooks = async (query, startIndex = 0) => {
  const API_KEY = 'AIzaSyDuwqSlIYYHpDmKYvCfFqKdkBjQpyA5ZkI';
  const url = query
    ? `https://www.googleapis.com/books/v1/volumes?q=${query}&startIndex=${startIndex}&maxResults=10&key=${API_KEY}`
    : `https://www.googleapis.com/books/v1/volumes?q=books&orderBy=relevance&startIndex=${startIndex}&maxResults=10&key=${API_KEY}`;

  try {
    const response = await axios.get(url);
    return response.data.items || [];
  } catch (error) {
    console.error("Failed to fetch books:", error.response ? error.response.data : error.message);
    throw new Error("Failed to fetch books");
  }
};

// 책 상세 정보를 불러오는 함수 추가
const fetchBookDetail = async (bookId) => {
  const API_KEY = 'AIzaSyDuwqSlIYYHpDmKYvCfFqKdkBjQpyA5ZkI';
  const url = `https://www.googleapis.com/books/v1/volumes/${bookId}?key=${API_KEY}`;
```

```
const fetchBookDetail = async () => {
  const foundBook = books.find(b => b.id === bookId);
  if (foundBook) {
    setBook(foundBook);
    setLoading(false);
  } else {
    try {
      const bookDetail = await getBookDetail(bookId);
      setBook(bookDetail);
    } catch (error) {
      setError('책 정보를 찾을 수 없습니다.');
    }
    setLoading(false);
  }
};
```



CONTENS 04

기능 구현

리뷰(CRUD)

```
const addReview = (newReview) => {
    const updatedReviews = [...reviews, newReview];
    setReviews(updatedReviews);
    localStorage.setItem('reviews', JSON.stringify(updatedReviews));
};

// 리뷰 삭제
const deleteReview = (id) => {
    const updatedReviews = reviews.filter((review) => review.id !== id);
    setReviews(updatedReviews);
    localStorage.setItem('reviews', JSON.stringify(updatedReviews));
};

// 리뷰 수정
const editReview = (id, updatedTitle, updatedContent) => {
    const updatedReviews = reviews.map((review) =>
        review.id === id
            ? { ...review, title: updatedTitle, content: updatedContent }
            : review
    );
    setReviews(updatedReviews);
    localStorage.setItem('reviews', JSON.stringify(updatedReviews));
};
```

```
export const BookContext = createContext();

// Reducer 함수 (책 목록 관리)
const bookListReducer = (state, action) => {
    switch (action.type) {
        case 'FETCH_SUCCESS':
            return {
                ...state,
                books: [...state.books, ...action.payload],
                loading: false,
                error: null,
            };
        case 'FETCH_ERROR':
            return { ...state, books: [], loading: false, error: action.payload };
        case 'SEARCH_SUCCESS':
            return { ...state, searchResults: action.payload, loading: false, error: null };
        default:
            return state;
    }
};
```

```
  e.preventDefault();
  if (reviewTitle.trim() && reviewContent.trim() && inputPassword.trim()) {
    if (editReviewId) {
      // 수정할 경우 비밀번호 확인
      const review = reviews.find(review => review.id === editReviewId);
      if (review && review.password === inputPassword) {
        editReview(editReviewId, reviewTitle, reviewContent);
      } else {
        alert('비밀번호가 일치하지 않습니다.');
        return;
      }
    } else {
      // 새 리뷰 추가
      addReview({ id: Date.now(), bookId, title: reviewTitle, content: reviewContent, password: inputPassword });
```

CONTENS 04

기능 구현

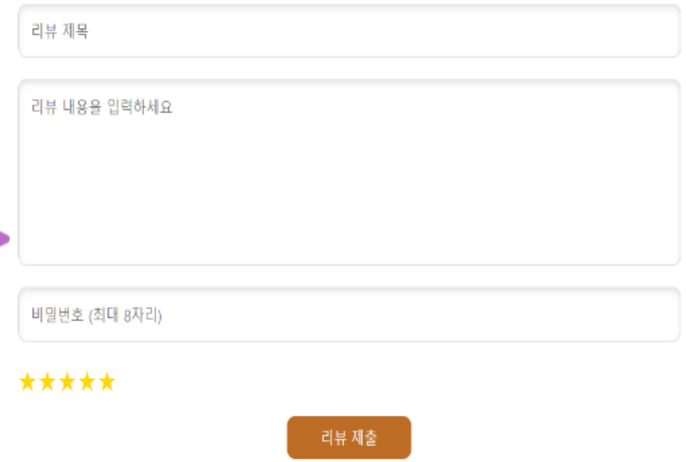
리류 별점 기능

```
<Rating
  count={5}
  size={24}
  value={rating}
  onChange={newRating => setRating(newRating)} // 별점 업데이트
/>
```

```
<Rating
  count={5}
  size={20}
  value={review.rating}
  edit={false}
/>
```

```
"react-rating-stars-component": "^2.2.0",
```

라이브러리 사용



CONTENS 05

문제 해결

CONTEXT API 최적화

ID별로 개별 관리되던 상태와 데이터를 Context API로 통합하여 전역적으로 관리할 수 있도록 변경하였습니다. 이를 통해 데이터 접근이 더 효율적이고 체계적으로 이루어지며, 불필요한 props 전달을 줄여서 코드의 가독성과 유지 보수성을 향상시켰습니다.

```
const getBookReviews = (bookId) => {
    return reviews.filter((review) => review.bookId === bookId);
};

// 리뷰 추가
const addReview = (newReview) => {
    const updatedReviews = [...reviews, newReview];
    setReviews(updatedReviews);
    localStorage.setItem('reviews', JSON.stringify(updatedReviews));
};

// 리뷰 삭제
const deleteReview = (id) => {
    const updatedReviews = reviews.filter((review) => review.id !== id);
    setReviews(updatedReviews);
    localStorage.setItem('reviews', JSON.stringify(updatedReviews));
};

// 리뷰 수정
const editReview = (id, updatedTitle, updatedContent) => {
    const updatedReviews = reviews.map((review) =>
        review.id === id
            ? { ...review, title: updatedTitle, content: updatedContent }
            : review
    );
    setReviews(updatedReviews);
    localStorage.setItem('reviews', JSON.stringify(updatedReviews));
};
```

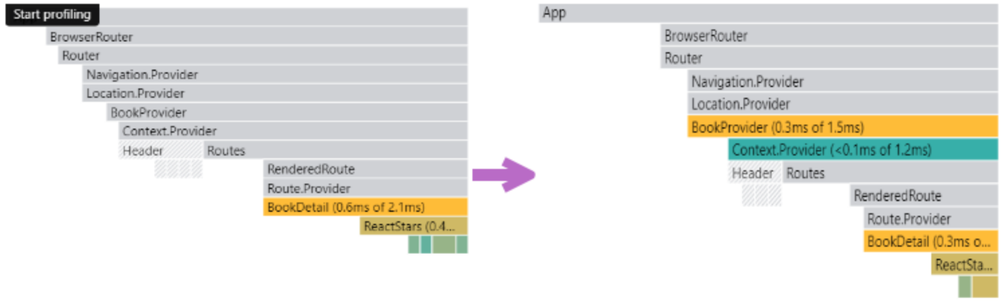
```
import React, { createContext, useState, useEffect, useReducer, useContext } from 'react';
import axios from 'axios';

// Context 생성
export const BookContext = createContext();

// Reducer 함수 (책 목록 관리)
const bookListReducer = (state, action) => {
    switch (action.type) {
        case 'FETCH_SUCCESS':
            return {
                ...state,
                books: [...state.books, ...action.payload],
                loading: false,
                error: null,
            };
        case 'FETCH_ERROR':
            return { ...state, books: [], loading: false, error: action.payload };
        case 'SEARCH_SUCCESS':
            return { ...state, searchResults: action.payload, loading: false, error: null };
        default:
            return state;
    }
};

// 초기 상태
const initialState = {
    books: [],
    searchResults: [],
    loading: true,
    error: null,
};
```

```
export const BookProvider = ({ children }) => {
    const [state, dispatch] = useReducer(bookListReducer, initialState);
    const [reviews, setReviews] = useState([]);
    const [favorites, setFavorites] = useState([]);
    const [page, setPage] = useState(0);
    const [hasMore, setHasMore] = useState(true);
    const [loadingMore, setLoadingMore] = useState(false);

    // 책 목록 API 호출 및 상태 업데이트
    const loadMoreBooks = async () => {
        if (loadingMore || !hasMore) return;
        setLoadingMore(true);

        try {
            const booksData = await fetchBooks('', page * 20);
            if (booksData.length === 0) {
                setHasMore(false);
            } else {
                dispatch({ type: 'FETCH_SUCCESS', payload: booksData });
                setPage((prevPage) => prevPage + 1);
            }
        } catch (error) {
            dispatch({ type: 'FETCH_ERROR', payload: error.message });
        } finally {
            setLoadingMore(false);
        }
    };

    useEffect(() => {
        loadMoreBooks();
    }, []);
```

CONTENS 05

문제 해결

렌더링 최적화




```jsx
const addReview = (newReview) => {
    const updatedReviews = [...reviews, newReview];
    setReviews(updatedReviews);
    localStorage.setItem('reviews', JSON.stringify(updatedReviews));
};
```


```jsx
const addReview = (newReview) => {
    const updatedReviews = [...reviews, newReview];
    setReviews(updatedReviews);
    localStorage.setItem('reviews', JSON.stringify(updatedReviews));
};
```


```jsx
// 리뷰 삭제
const deleteReview = (id) => {
    const updatedReviews = reviews.filter((review) => review.id !== id);
    setReviews(updatedReviews);
    localStorage.setItem('reviews', JSON.stringify(updatedReviews));
};

// 리뷰 수정
const editReview = (id, updatedTitle, updatedContent) => {
    const updatedReviews = reviews.map((review) =>
        review.id === id
            ? { ...review, title: updatedTitle, content: updatedContent }
            : review
    );
    setReviews(updatedReviews);
    localStorage.setItem('reviews', JSON.stringify(updatedReviews));
};
```

```jsx
const getBookReviews = useMemo(() => (bookId) => {
    return reviews.filter((review) => review.bookId === bookId);
}, [reviews]); // useMemo로 메모이제이션

const getBookDetail = useCallback(async (bookId) => {
    try {
        const bookDetail = await fetchBookDetail(bookId);
        return bookDetail;
    } catch (error) {
        console.error(error);
        return null;
    }
}, []);

// LocalStorage에서 리뷰 및 즐겨찾기 불러오기
useEffect(() => {
    const storedReviews = JSON.parse(localStorage.getItem('reviews')) || [];
    const storedFavorites = JSON.parse(localStorage.getItem('favorites')) || [];
    setReviews(storedReviews);
    setFavorites(storedFavorites);
}, []);

// 특정 책에 대한 리뷰 반환
const getBookReviews = useCallback((bookId) => {
    return reviews.filter((review) => review.bookId === bookId);
}, [reviews]);

// 리뷰 추가
const addReview = useCallback((newReview) => {
    const updatedReviews = [...reviews, newReview];
    setReviews(updatedReviews);
    localStorage.setItem('reviews', JSON.stringify(updatedReviews));
}, [reviews]); // useCallback으로 최적화
```

```jsx
import React, { useCallback, useMemo } from 'react';
```

useCallback과 useMemo를 사용하여 함수와 데이터를 메모이제이션함으로써, 불필요한 재렌더링을 방지하고 성능을 최적화했습니다. 특히, 리뷰 추가, 수정, 삭제 함수와 책 데이터를 처리하는 로직의 효율성을 높였습니다.

CONTENS 06

느낀 점

어려움 극복

프로젝트 진행 중 발생한 오류와 코드 최적화 과정에서 어려움을 겪었지만, 다양한 방법을 시도하면서 끝내 해결해낸 부분이 가장 보람찼습니다. 특히 렌더링 최적화, CONTEXT API 사용, 별점 기능 같은 복잡한 부분들을 성공적으로 처리한 점이 큰 성과였습니다.
그리고 모르는 부분의 패키지나 라이브러리를 알아보며 구현하도록 하였습니다.

잘한점

팀원들과 적극적으로 소통하면서 문제를 해결하려고 노력했습니다. 서로 의견을 공유하고, 다양한 아이디어를 주고받으면서 해결 방안을 찾는 데 큰 도움이 되었습니다. 그리고 문제가 발생하였을때 팀원들이랑 다양한 방법을 찾아봐 해결하려는 의지를 보였던거 같습니다.

아쉬운 부분

처음부터 프로젝트 구조나 기능 설계를 조금 더 세밀하게 했더라면, 시간을 단축하고 필요한 기능들을 더 추가 할수 있었던거 같고, 추가적으로 테스트 부분 조금 더 자세히 했더라면 부족한부분을 보안을 할수 있었던거 같습니다.

앞으로 각오

초기 설계를 더욱 철저히 준비하여, 중간에 수정이 필요 없도록 할 계획입니다. 더 체계적인 계획을 통해 프로젝트가 더욱 효율적으로 진행될 수 있도록 하고, 사용자의 피드백과 개선할 부분을 철저히 분석하고 반영해, 사용자 경험을 개선하고 더 나은 프로젝트를 만들고자 합니다.

BOOK REVIEW

THANK YOU

함께해주셔서 감사합니다

담당자 : 정우성(WEARE2415@NAVER.COM)