2023학년도 프로젝트 완료보고서

바디 다이어리

2023년 6월 19일

컴퓨터정보과

|  |  |
| --- | --- |
| 팀명 | 이대준 |
| 팀장 | 이대준(201944056) |
| 팀원 | 이대준(201944056) |



|  |
| --- |
| **본 과제(결과물)는 인하공업전문대학 컴퓨터정보과 Start-Up 프로젝트 교과목의 프로젝트 완료보고서 입니다.** |

2023년도

**프로젝트 완료보고서**

과제명 : Body-Diary

(운동 수행능력 및 몸 상태를 관리하는 다이어리 웹 개발)

2023. 6. 19.



제 출 문

**• 과 제 명 : (예: 지그비 통신을 이용한 메신져 개발)**

**• 프로젝트 책임자 : 인하공업전문대학 컴퓨터정보과 홍길동**

**• 프로젝트 참여 팀원 : 인하공업전문대학 컴퓨터정보과 홍이동**

**인하공업전문대학 컴퓨터정보과 홍삼동**

2023년도 인하공업전문대학 컴퓨터정보과 Start-Up 프로젝트 교과목의

결과물로 이 완료보고서를 제출합니다.

2023. 6. 19.

**프로젝트 책임자 : 컴퓨터정보과(201944056) 이 대 준 (인)**

**인하공업전문대학 컴퓨터정보과 최효현 교수님 귀하**

프로젝트 요약문

|  |  |
| --- | --- |
| 지도교수 | 최효현 |
| 프로젝트명 | Body-Diary |
| 프로젝트  책임자 | 컴퓨터정보과(201944056) 이대준 |
| 프로젝트  참여 팀원 | 컴퓨터정보과(201944056) 이대준 |
| 그림입니다. 원본 그림의 이름: CLP00004ea4000e.bmp 원본 그림의 크기: 가로 975pixel, 세로 599pixel그림입니다. 원본 그림의 이름: CLP00004ea4000f.bmp 원본 그림의 크기: 가로 540pixel, 세로 652pixel그림입니다. 원본 그림의 이름: CLP00004ea40010.bmp 원본 그림의 크기: 가로 548pixel, 세로 651pixel  [그림 1] 요약 메인화면  [그림 2] 요약 다이어리  [그림 3] 요약 마이페이지  “Body-Diary” 프로젝트는 다이어리 기반 웹 프로젝트로 다이어리 자체의 성향에 비중을 더 많이 두어 웹 사용자들에게 일기장을 작성하는 것과 같은 인터페이스와 여러 가지 웨이트 트레이닝 프로그램을 자유도를 해치지 않는 선에서 제공하여, 자유로운 운동 기록 기능을 제공하면서도, 최적화된 운동 계획 및 작성 프레임을 제공하여 더 효율적인 수행능력 증진 및 근비대를 도모하는 것을 목표로 하고 있다.  UI는 메인 페이지로 캘린더 페이지와 다이어리 페이지, 마이페이지로 기본 페이지들을 구성했다. 개발 계획에 있는 기능들로는 운동 루틴 프로그램 제공 기능, 특이점 메모 저장 기능, 무게 변화 추이를 보여주는 기능이 있다.  개발환경은 운영체제-windows11, 프레임워크-React js, 서버-Node js, 데이터베이스-MySQL을 사용했다. | |

참여자 인적사항

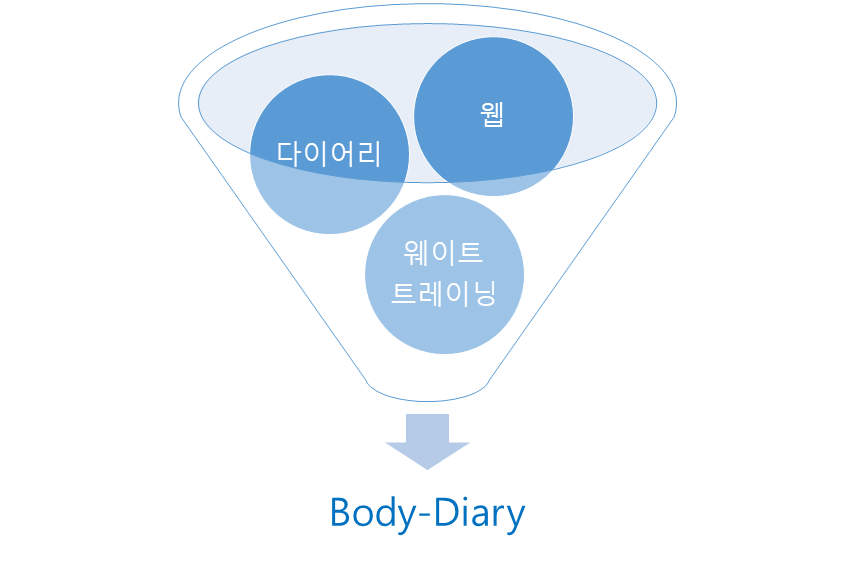
1. 프로젝트 팀장

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 성 명 | | 이 대 준 | 학 번 | 201944056 |
| 학 과 | | 컴퓨터정보과 | | |
| 연락처 | H.P | 010 – 8514 - 5374 | | |
| E-mail | dkdnl232@gmail.com | | |

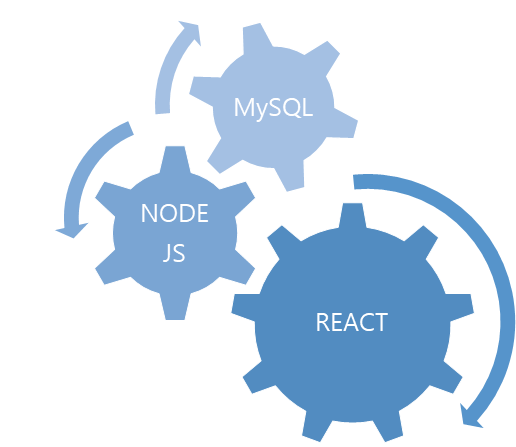
|  |  |
| --- | --- |
|  | 목 차 |
| 1. 프로젝트 목적 1  2. 프로젝트 목표 및 기대효과 2  2-1. 프로젝트 목표 2  2-2. 프로젝트 기대효과 2  3. 기존 사례 분석 및 SWOT 분석 3  3-1. 기존 사례 분석 3  3-2. SWOT 분석 3  4. 프로젝트 진행범위 및 방법 4  4-1. 프로젝트 진행 범위 4  4-2. 프로젝트 진행 방법 4  5. 프로젝트 주요 내용 5  5-1. 프로젝트 주요 개발 내용 5  5-2. 개발 환경 5  6. 프로젝트 수행 일정 6  7. 프로젝트 위험 요소 관리 결과 7  8. 프로젝트 개발 결과물 요약 8  9. 요구 사양 9  10. 설계 내용 10  11. 시험 결과 11  12. 프로젝트 수행 소감 12  부록  1. 프로그램 Source 14  2. 사용자 설명서 (Manual) 15 |

1. 프로젝트 목적

1.1 주제 선정 계기

웨이트 트레이닝 관련 다이어리라는 아이디어를 생각하게 된 것은 평소 운동에 관심이 많았고 이와 관련된 보조 어플은 시장에 많았지만 웹에서는 찾기 어려웠고, 본인은 현재 재학중인 과와 꿈꾸고 있는 직업 특성상 랩탑과 데스크톱을 자주 사용하다 보니 ‘웹사이트 중 헬스 기록을 남길 다이어리가 있다면 좋겠다’는 생각을 하게 됐고, 이에 본 프로젝트를 진행하게 되었다. 

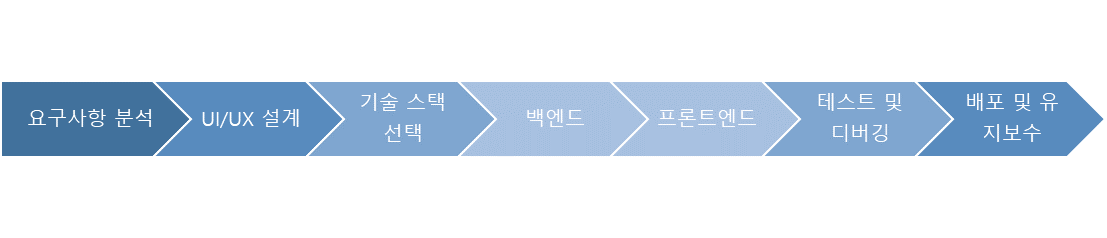
[그림 4] 주제 선정



[그림 5] 개발 환경

1.2 목적

저번 학기, 마무리 짓지 못하고 실패로 돌아간 웹 프로젝트에서 실패 요인을 분석하고 보완하여 성장하는 것을 첫번째 목적으로 하고, 때문에 웹 관련 프론트 중심으로 방향을 설정하며 주제 선정을 시작하였다. React, Node.js 및 MySQL를 사용하여 ‘Body-diary’ 웹 개발을 진행하며 여러 최신 자료들을 찾아보고 해당 부문의 개발 동향(문법, 프로젝트 폴더 구조, 서버 통신 방법 등)과 개발 흐름을 파악하는 것이 두번째 목적이다.



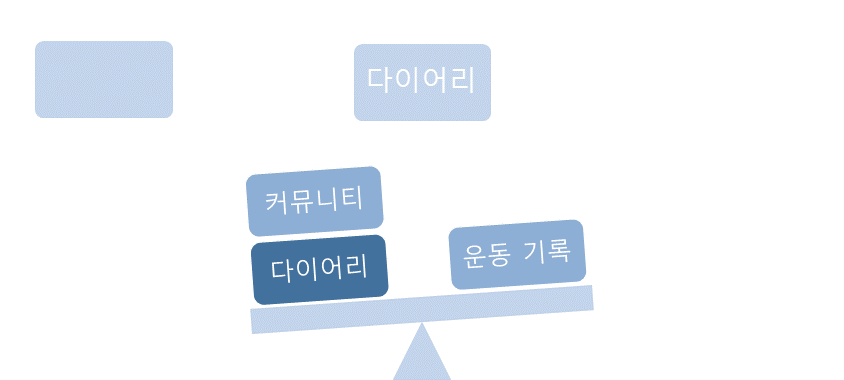
[그림 6] 개발 프로세스

2. 프로젝트 목표 및 기대효과

2-1. 프로젝트 목표

“Body-Diary” 다이어리 기반 웹 프로젝트로 다이어리 자체의 성향에 비중을 더 많이 두어서 웹 사용자들에게 일기장을 작성하는 것과 같은 인터페이스와 여러 가지 웨이트 트레이닝 프로그램을 자유도를 해치지 않는 선에서 제공하여, 자유로운 운동 기록 기능을 제공하면서도, 최적화된 운동 계획 및 작성 프레임을 제공하여 더 효율적인 수행능력 증진 및 근비대를 도모하는 것을 목표로 하고 있다.

[그림 7] 프로젝트 목표



2-2. 프로젝트 기대효과

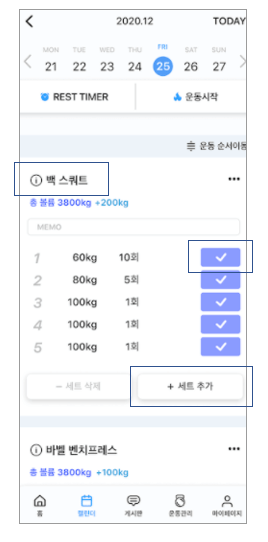
‘Body-Diary’를 사용하여 사용자들이 운동 수행 내용들을 자유롭게 기록하고, 공유할 수 있고 이에 따라 지난 운동 기록들을 돌아보며 자신의 성장을 객관적으로 분석할 수 있고 성취감도 얻을 수 있다. 또한 운동 기록들을 공유하며 다른 사용자들과 의견을 나누며 성장하고 즐거움을 얻음을 기대해 볼 수 있다.

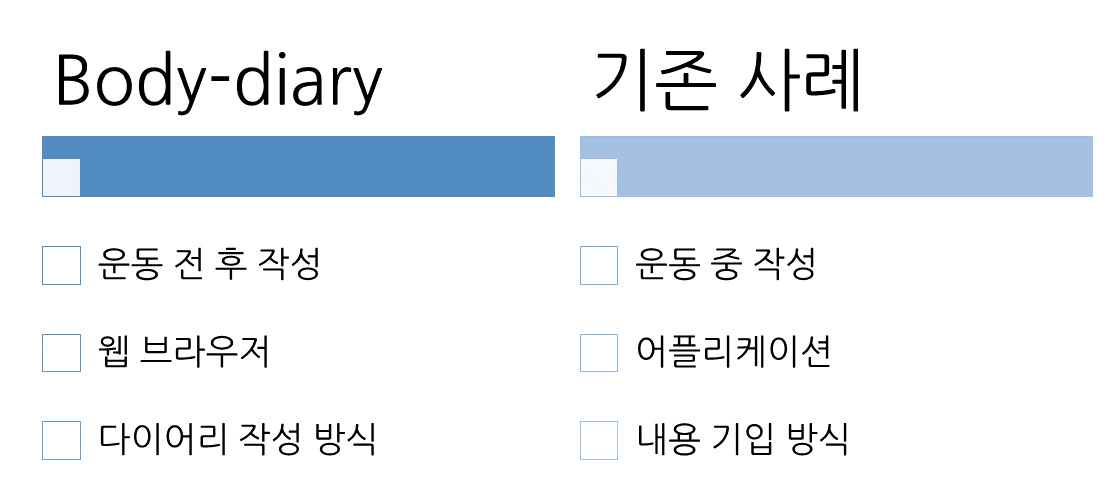
3. 기존 사례 분석 및 SWOT 분석

3-1. 기존 사례 분석

기존에 존재하는 운동 다이어리들은 대부분이 어플리케이션으로 존재하며 다이어리 작성 시간을 운동중으로 초점을 맞춰 놓은 것들이 대부분이다. 이러한 이유로 인해, 대부분의 다이어리가 어플리케이션 형태로 개발되며, 빠르게 작성되야 하기 때문에 사용자의 입력을 최소화 시키고 거의 대부분의 내용들을 인터페이스에서 제공하는 모습을 보인다. 하지만 이러한 방식은 사용자의 작성 자유도를 제한한다고 생각했다.

[그림 8] 기존 사례 분석





다이어리 작성 시간을 운동중이 아닌 전후로 초점을 맞추고, 그에 따라 웹으로 다이어리를 개발한다는 점과 프리한 인터페이스로 사용자에게 자유도를 제공해 준다는 점에서 기존 사례들과 차이점이 있다.

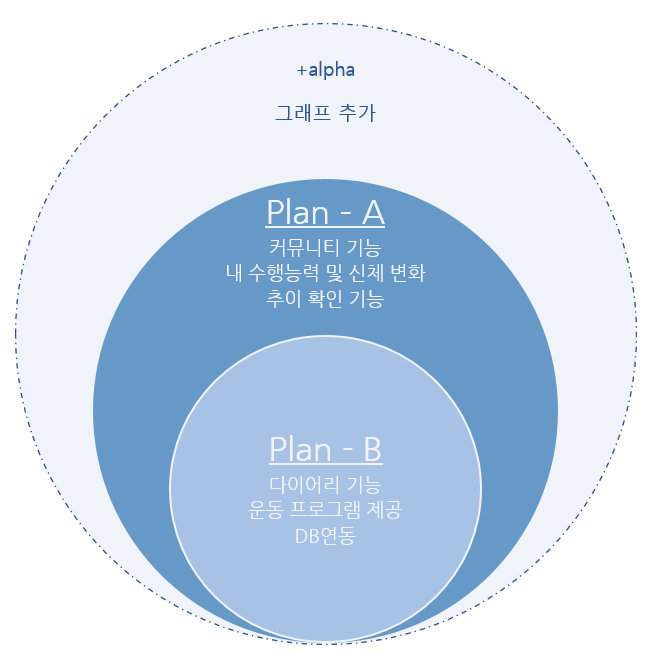
3-2. SWOT 분석

|  |  |
| --- | --- |
| Strengths (강점, 장점)  - 개발에 필요한 수준의 해당 분야에 대한 전문지식이 충분함  -본 프로젝트 주제와 비슷한 기존의 어플리케이션을 사용한 경험이 많음  유사한 환경(개발 환경, 기간)에서 프로젝트를 진행해 본 경험이 있음 | Weakness (단점, 약점)  - 개발 실력이 부족함   * 새로운 언어를 시도함   - 많은 시간을 투입하지 못함 |
| Opportunities (기회)  - 앞으로의 진로를 고민할 때 프론트엔드쪽 개발이 나의 적성에 맞을지 경험해볼 기회가 될 수 있음   * 다이어리 작성을 좋아하고 운동을 좋아하는 사람에게 좋은 인터페이스가 될 수 있음 | Threats (위협)  - 시간 부족으로 완성이 안될 수 있음 |

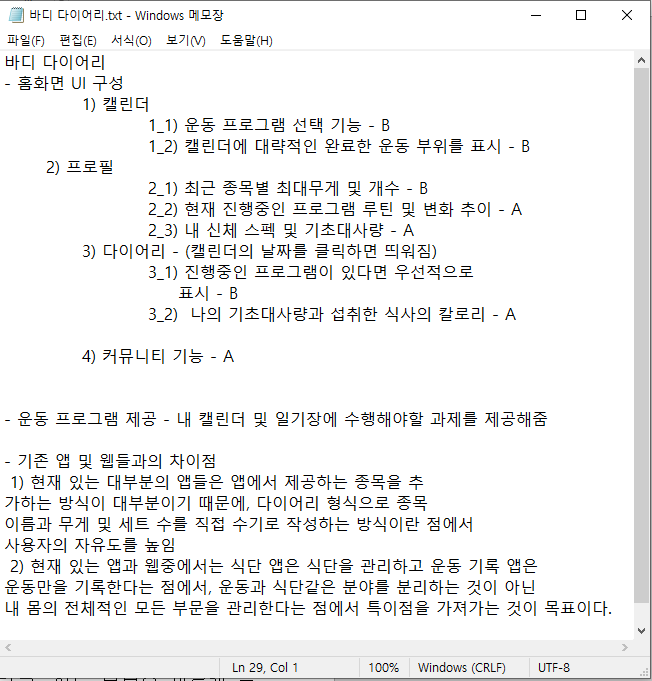
4. 프로젝트 진행범위 및 방법

4-1. 프로젝트 진행 범위

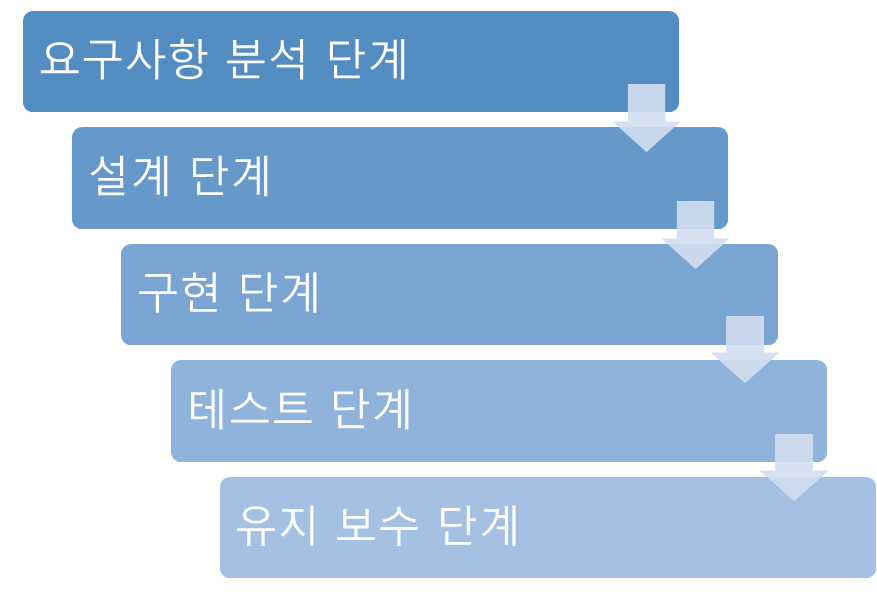
WT 전략에 기반하여 지난 학기에 시간 부족 및 난이도 설정 미스로 프로젝트를 완성하지 못한 점을 생각했다. 이러한 이유로 프로젝트 진행 범위는 초기 구상에서 플랜 A와 플랜 B의 기능 중 개발 난이도와 소요 기간을 고려하여 적절히 바꾸고, 소요 기간과 개발 난이도에서 지금의 나의 기량으로 현실성이 없는 기능들은 제외하며 설정했다.



[그림 9] 진행 범위



4-2. 프로젝트 진행 방법



[그림 10] 소프트웨어 공학

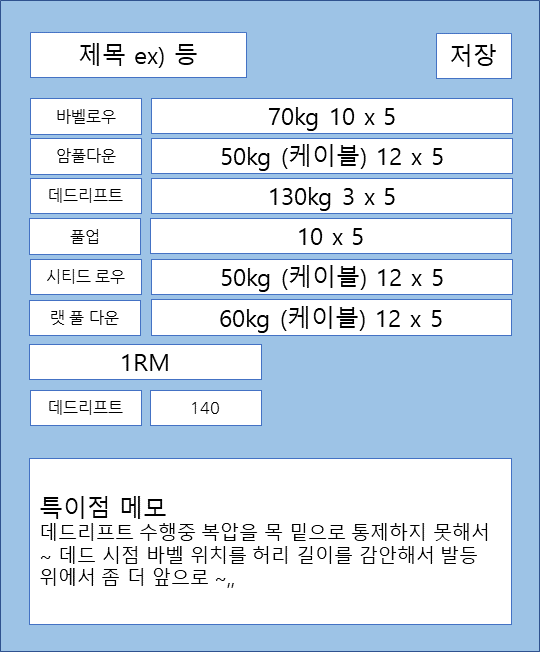
이번 프로젝트는 소프트웨어 공학의 개발 과정을 기반으로 진행하며, 실제 과정은 테스트 단계에서 마칠 생각이지만, 실제 클라이언트에게 배포한다고 가정하여 설계 및 구현 단계에서 유지 보수에 용이하도록 코드를 논리적이고 간결하게 작성할 계획이다. 이러한 이유로 유지 보수 단계를 개발 과정에 남겨두었다.

5. 프로젝트 주요 내용

5-1. 프로젝트 주요 개발 내용

메인 페이지는 캘린더 형식으로 구성되어 있고, 날짜를 클릭하면 다이어리 페이지가 열린다.



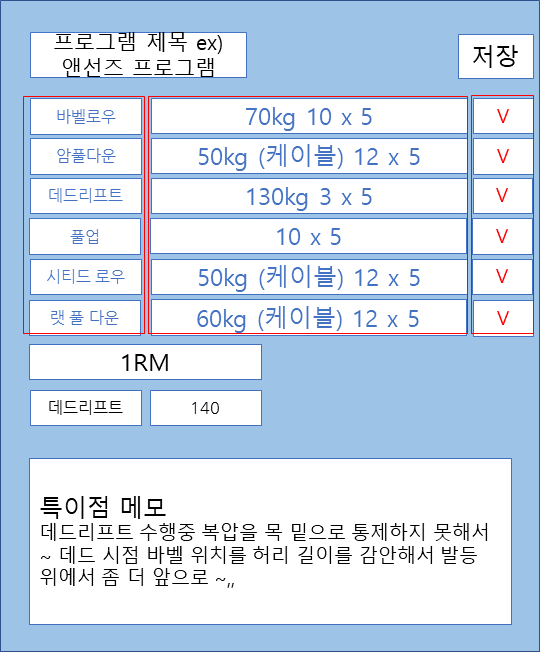


[그림 11] 메인 페이지

다이어리 페이지에는 종목별 기입할 수 있는 칸들이 존재하고, 1rm 측정 결과를 입력할 수 있는 칸과, 그 날 운동의 특이점을 메모할 수 있는 메모장이 존재한다. 1rm 측정 종목은 사용자가 입력하는 형식이 아닌 셀렉트 박스형으로 구현되어 있다.

캘린더의 날짜 옆에는 제목에 입력한 그 날의 운동 종목(등, 가슴, 어깨, 하체 등)이 간략하게 표시된다.

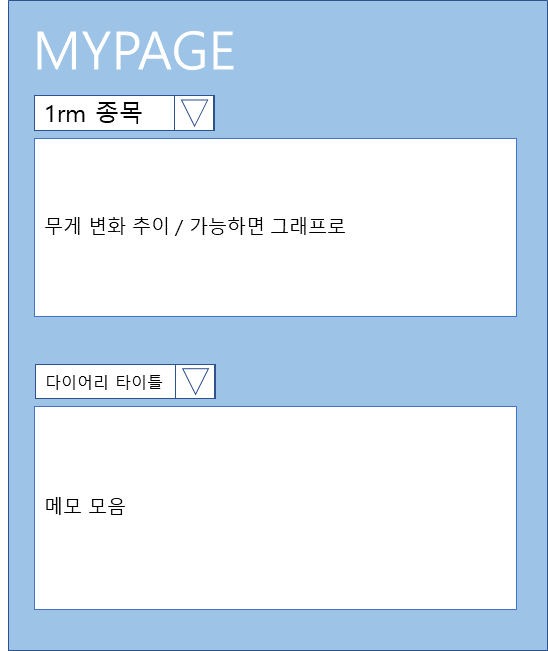
운동 프로그램을 선택후 시작 버튼을 누르면 해당 날짜부터 다이어리의 종목과 종목 내용 입력란들이 정해진 데이터들로 고정되고, 그 옆에 체크박스가 생긴다.



[그림 12] 프로그램 선택

캘린더 날짜 옆에는 프로그램 이름이 표시된다.

마이페이지에는 그동안 기록한 1RM 종목들의 무게와 특이점을 기록한 메모들을 볼 수 있는 기능이 있다. 1RM 무게 변화 추이는 플랜 A까지 진행하게 되면 그래프로 나타나도록 개발할 계획이다.



[그림 13] 마이 페이지

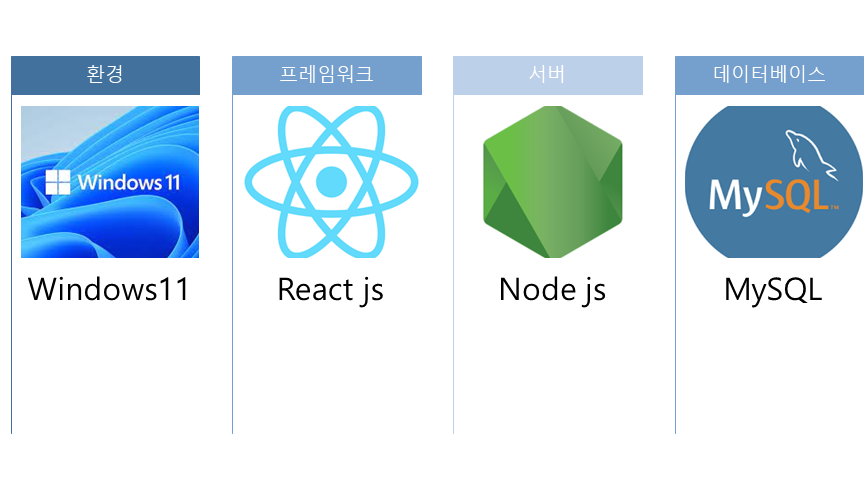
5-2. 개발 환경

운영체제 – windows11

프레임워크 – React js

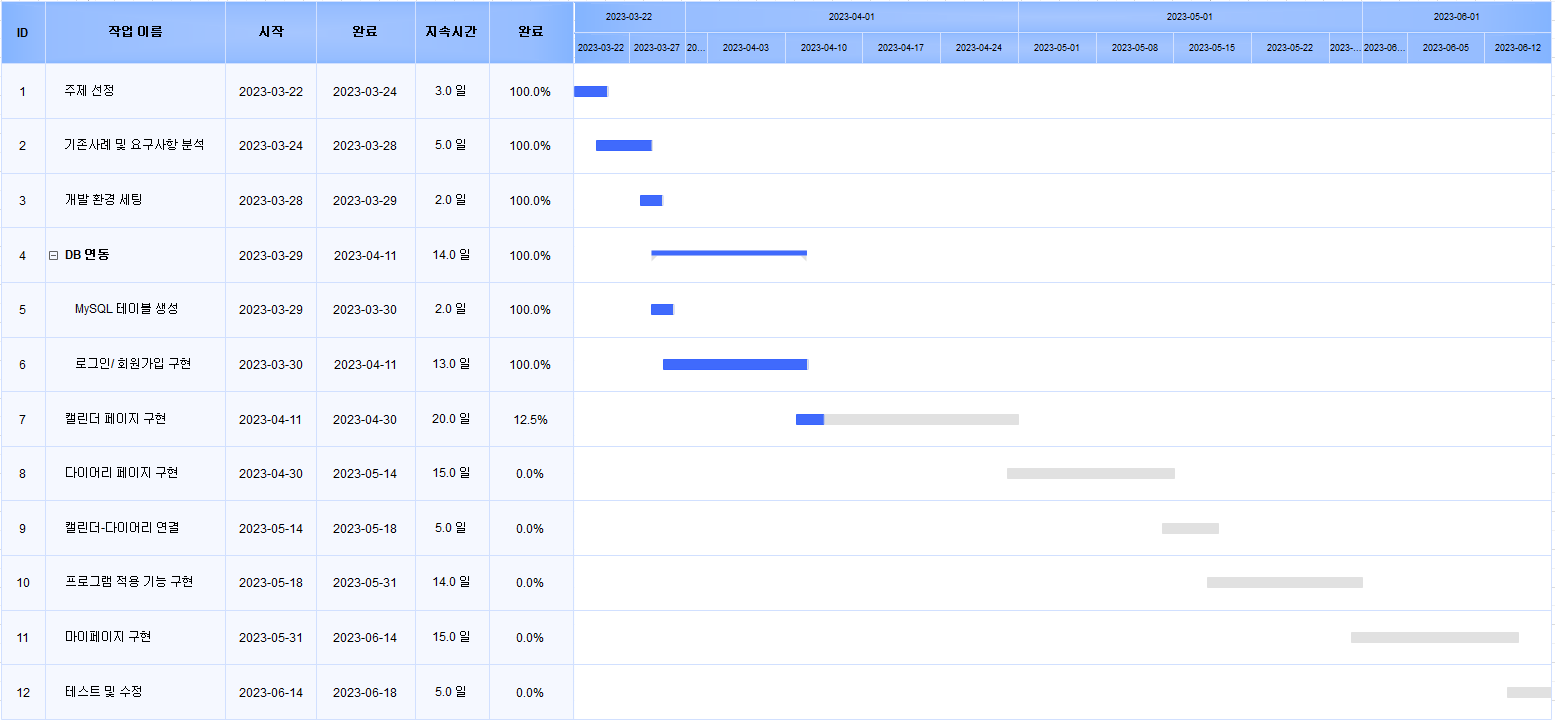
서버 - Node js

데이터베이스 - MySQL



[그림 14] 개발 환경

6. 프로젝트 수행 일정



[그림 15] 간트 차트

7. 프로젝트 위험 요소 관리 결과

위험요소 : 1. 중간고사 및 기말고사 2. 캘린더 구현 소요기간 초과

--> 위험회피 계획 : 단계별 소요기간을 세분화하여 예상 소요기간과 실제 소요기간 의 오차범위를 줄여 리스크 발생 가능성을 줄인다.

--> 위험 회피 결과 : 기능별로 최대한 컴포넌트로 분리하여 가시적으로 일의 진 행을 분리하여 할 수 있도록 하였고 단계별 소요기간을 세 분화하고 위험요소를 효과적으로 회피하였다.

8. 프로젝트 개발 결과물 요약

메인 페이지는 캘린더, 프로그램 설정과 설명으로 이루어져 있으며 드롭다운에서 프로그램을 선택하면 해당 프로그램에 대한 설명이 나오고, start 버튼을 클릭하면 프로그램 일정이 시작된다. 캘린더의 타일은 작성된 다이어리의 제목 -> 프로그램 일정 -> 기본 날짜 순으로 우선 출력된다. 캘린더의 타일을 클릭하면 다이어리 페이지가 나타난다.

텍스트, 스크린샷, 소프트웨어, 컴퓨터이(가) 표시된 사진

자동 생성된 설명

[그림 16] 메인 화면

프로그램의 일정이 포함된 다이어리 페이지는 placeholder 형태로 제공되어 있는 프로그램 수행 내용을 직접 입력할 수도 있고 체크박스를 선택하여 입력할 수도 있다. 아래에는 1rm 종목과 무게를 입력하는 공간이 있고, 오른쪽에는 그 날 특이점을 메모할 수 있는 공간이 있다. - Plan\_B

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

자동 생성된 설명

[그림 17] 다이어리 페이지

작성된 다이어리는 아래와 같이 출력된다.

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

[그림 18] 작성된 다이어리

텍스트, 스크린샷, 도표, 폰트이(가) 표시된 사진

자동 생성된 설명

[그림 19] 마이페이지

마이페이지의 화면 왼쪽에는 그간 기록된 1rm 무게의 변화 추이를 그래프로 나타내고 오른쪽에는 특이점을 적어놓은 메모들이 카드 형태로 출력된다. - Plan\_A

9. 요구 사양

필요한 요구 사양을 분석하고 이를 적는다.

9-1. 웹서버 사양

사양 1: 웹서버는 HTTP, HTTPS 프로토콜을 지원하는 소프트웨어를 포함해야 한다. (예: Apache, Nginx, IIS 등)

사양 2: 웹서버는 충분한 저장공간을 가지고 있어야 웹사이트와 관련된 파일들을 저장할 수 있다.

사양 3: 웹서버는 웹사이트의 트래픽을 처리하기 충분한 네트워크 대역폭을 가지고 있어야 한다.

9-2. 데이터베이스 서버 사양

사양 4: 데이터베이스 서버는 관계형 또는 비관계형 데이터베이스 시스템을 실행할 수 있는 소프트웨어를 포함해야 한다. (예: MySQL, PostgreSQL, MongoDB 등)

사양 5: 데이터베이스 서버는 웹사이트가 필요로 하는 데이터를 저장하기 위한 충분한 저장 공간을 가지고 있어야 한다.

9-3. 백엔드 프로그래밍 언어 / 프레임워크 사양

사양 6: 서버는 선택한 백엔드 프로그래밍 언어 또는 프레임워크를 실행할 수 있어야 한다. (예: Node.js, Ruby on Rails, Django, Laravel 등)

9-4. 클라이언트 사이드 사양

사양 7: 클라이언트 디바이스는 웹 브라우저를 실행할 수 있어야 한다. (예: Google Chrome, Mozilla Firefox, Safari, Microsoft Edge 등)

사양 8: 웹 브라우저는 HTML, CSS, JavaScript를 지원해야 한다.

10. 설계 내용

1. 애플리케이션 설계

1.1 DiaryPage 컴포넌트 동작 설계

1.1.1. 데이터 초기화

- 먼저, 컴포넌트가 로드될 때 `useState`를 통해 상태변수들을 초기화합니다. 해당 상태변수들은 사용자가 입력하는 다이어리의 내용과, 사용자의 운동 프로그램 정보를 저장합니다.

1.1.2. 텍스트 필드 입력 처리

- 운동 일지의 제목, 1RM 이벤트, 최대 중량, 그리고 메모에 해당하는 텍스트 필드가 입력되면, 각각의 `handle` 함수가 실행되어 상태변수에 해당 값을 저장합니다.

1.1.3. 운동 항목 체크박스와 텍스트 필드 입력 처리

- 각 운동 항목의 체크박스가 체크되면, 해당 항목의 `isChecked` 상태가 true로 설정되고, 이름과 내용이 해당 프로그램 데이터로 자동으로 채워집니다. 체크가 해제되면 `isChecked` 상태가 false로 설정되고, 이름과 내용이 빈 문자열로 설정됩니다.

- 운동 항목의 이름과 내용 텍스트 필드에 입력하면, `handleChange` 함수가 실행되어 해당 항목의 이름과 내용 상태값이 사용자가 입력한 값으로 업데이트됩니다.

1.1.4. '저장' 버튼 클릭 처리

- '저장' 버튼이 클릭되면, `handleSubmit` 함수가 실행됩니다. 이 함수에서는 axios를 사용해 서버에 POST 요청을 보내, 현재 상태변수에 저장된 운동 일지 데이터를 서버에 저장합니다. 그 후, 알림창을 통해 사용자에게 데이터가 저장되었다는 메시지를 보여주고, 홈페이지로 이동합니다.

1.2. 서버 통신

1.2.1. 운동 프로그램 데이터 요청

- `useEffect` 훅이 실행될 때(즉, 컴포넌트가 마운트되거나 `programTodo`가 업데이트될 때), `fetchData` 함수를 실행하여 서버에 운동 프로그램 데이터를 요청합니다.

11. 시험 결과

개발하면서 수행하였던 시험 결과를 표현한다.

가능한 자세하게 수록한다.

예:

11.1 단위시험

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 시험 항목  (function) | 시험내용 | 예상시험결과 | 시험결과  (round 1) | 시험결과  (round 2) |
| handleTitle | 제목 텍스트 필드에 값을 입력한다. | 입력한 값이 상태변수에 저장되어야 한다. | pass | pass |
| handleEvent | 드롭다운 메뉴에서 이벤트를 선택한다. | 선택한 이벤트가 상태변수에 저장되어야 한다. | pass | pass |
| 통신 | 서버측에 상태변수에 저장된 값들을 보내고 서버측에서 데이터를 받아와 상태변수에 저장한다. | 서버에 요청한 데이터가 상태변수에 저장되고, 클라이언트 측에서 보낸 값이 서버의 콘솔창에 출력이 되어야 한다. | fail | pass |
| chart | 날짜 범위와 종목을 선택하고 load 버튼을 클릭한다. | 해당 종목의 날짜 범위안에 있는 DB의 데이터들을 기반으로 그래프가 그려져야 한다. | pass | pass |
| 운동 program 적용 | 운동 프로그램을 선택하고 버튼을 클릭한다. | 타겟 날짜의 다이어리들에 해당 프로그램 내용들이 제공되어야 한다. | fail | pass |

시험 결과:

1 round: 항목 20개 중 18개 pass, 2개 fail

2 round: 항목 20개 중 20개 pass, 0개 fail

12. 프로젝트 수행 소감

이번 프로젝트를 통해 다양한 기술을 배우고 경험할 수 있었습니다. 우선, 웹 기반 운동 기록 관리 시스템을 개발하기 위해 React와 Node.js를 사용해보았습니다. 이를 통해 웹 개발에 대한 이해도를 높이고, 현대적인 웹 프레임워크에 대한 경험을 쌓을 수 있었습니다.

또한, 실제 사용자의 요구사항을 반영하여 시스템을 설계하고 구현하는 과정에서 사용자 중심의 개발에 대한 중요성을 깨닫게 되었습니다. 사용자의 편의성과 사용성을 고려하는 것이 얼마나 중요한지를 체감하였고, 이를 프로젝트에 반영하기 위해 노력하였습니다.

더불어, 코드 테스팅 및 디버깅의 중요성도 새삼 느꼈습니다. 특히 단위 테스트를 통해 개별 기능을 체계적으로 확인하고, 통합 테스트를 통해 전체 시스템의 동작을 검증하는 과정에서 많은 문제점을 발견하고 개선할 수 있었습니다.

이번 프로젝트 수행을 통해 얻은 경험과 지식은 앞으로의 개발 업무에 있어 큰 자산이 될 것이라고 생각합니다. 앞으로도 지속적으로 배우고 성장하는 개발자가 되기 위해 노력하겠습니다

부 록

1. 프로그램 Source

2. 사용자 설명서 (Manual)

1. 프로그램 Source

[CalendarComponent.js]

import React, { useState, useEffect } from "react";

import Calendar from "react-calendar";

import "./CalendarComponent.css";

import { useNavigate } from "react-router-dom";

import axios from "axios";

import InputLabel from "@mui/material/InputLabel";

import MenuItem from "@mui/material/MenuItem";

import FormControl from "@mui/material/FormControl";

import Select from "@mui/material/Select";

import Button from "@mui/material/Button";

import Explain1 from "./explains/explain1";

import Explain2 from "./explains/explain2";

import Explain3 from "./explains/explain3";

import Explain4 from "./explains/explain4";

import Explain5 from "./explains/explain5";

const CalendarComponent = () => {

const navigate = useNavigate();

useEffect(() => { // 로그인 확인

if (localStorage.getItem("isLogin") !== "true") {

navigate("/login");

}

// eslint-disable-next-line react-hooks/exhaustive-deps

}, []);

const [value, setValue] = useState(new Date());

const [tileContent, setTileContent] = useState([]);

const [type, setType] = useState("");

const [programSchedule, setProgramSchedule] = useState({});

const [startDate, setStartDate] = useState("");

useEffect(() => { // 커스텀 타일을 위한 날짜 변수 관리

const tiles = [];

const startOfYear = new Date(value.getFullYear(), 0, 1);

const endOfYear = new Date(value.getFullYear(), 11, 31);

let day = startOfYear;

while (day <= endOfYear) {

tiles.push({ date: new Date(day), view: "month" });

day = addDays(day, 1);

}

Promise.all(

tiles.map((tile) =>

renderTileContent(tile).then((content) => ({

...tile,

content,

}))

)

).then(setTileContent);

// eslint-disable-next-line react-hooks/exhaustive-deps

}, [value, programSchedule]);

//db에서 프로그램 가져오는 로직

useEffect(() => {

const fetchData = async () => {

try {

const response = await axios.post("http://localhost:3001/get\_program", {

userid: localStorage.getItem("userid"),

});

setStartDate(response.data.date);

const schedule = createProgramSchedule(response.data.schedule);

setProgramSchedule(schedule);

} catch (error) {

console.error("Error fetching diary data:", error);

}

};

fetchData();

// eslint-disable-next-line react-hooks/exhaustive-deps

}, [value, programSchedule]);

const handleType = (e) => {

setType(e.target.value);

};

//선택한 프로그램 저장

const saveProgramSchedule = async (type) => {

try {

console.log(type);

const response = await axios.post(

"http://localhost:3001/insert\_program",

{

userid: localStorage.getItem("userid"),

schedule: type,

date: toLocalDateString(new Date()),

}

);

console.log("Server response:", response.data);

} catch (error) {

console.error("Error sending data to server:", error);

}

};

const getType = (e) => {

alert("프로그램 일정이 내일부터 시작됩니다.");

saveProgramSchedule(type);

};

const createProgramSchedule = (type) => {

const schedule = {};

const parsedStartDate = new Date(startDate);

for (let i = 1; i <= 42; i++) {

const date = addDays(parsedStartDate, i);

schedule[toLocalDateString(date)] = `${type}${i}`;

}

return schedule;

};

const handleDelete = (e) => {

try {

alert("프로그램 일정이 초기화 되었습니다.");

axios.post(

"http://localhost:3001/delete\_program",

{

userid: localStorage.getItem("userid"),

}

);

} catch (error) {

console.error("Error sending data to server:", error);

}

}

const onChange = (nextValue) => {

setValue(nextValue);

const dateString = toLocalDateString(nextValue);

const data = {

userid: localStorage.getItem("userid"),

date: dateString,

};

axios

.post("http://localhost:3001/diary\_url", data)

.then((response) => {

if (response.data.exist === true) {

navigate(`/diary\_view/${dateString}`);

} else {

navigate(`/diary/${dateString}`, {

state: { programTodo: programSchedule[dateString] },

});

}

})

.catch((error) => {

console.log(error);

// 에러가 발생했을 때 처리할 코드

});

};

const renderTileContent = ({ date, view }) => {

if (view === "month") {

const dateString = toLocalDateString(date);

const data = {

userid: localStorage.getItem("userid"),

date: dateString,

};

return axios

.post("http://localhost:3001/diary\_url", data)

.then((response) => {

if (response.data.exist === true) {

return <div className="customTitle">{response.data.title}</div>;

} else if (programSchedule[dateString]) {

return (

<div className="customTitle">{programSchedule[dateString]}</div>

);

} else {

return null;

}

})

.catch((error) => {

console.log(error);

return null;

});

}

return null;

};

return (

<div className="background">

<div className="bg\_left">

<div style={{ borderBottom: "1px solid #d0d7de", width: "500px" }}>

<div style={{ float: "left" }}>

<FormControl variant="standard" sx={{ m: 1, minWidth: 120 }}>

<InputLabel id="demo-simple-select-standard-label">

Programs

</InputLabel>

<Select

labelId="demo-simple-select-standard-label"

id="demo-simple-select-standard"

value={type}

onChange={handleType}

label="Programs"

>

<MenuItem value="">

<em>None</em>

</MenuItem>

<MenuItem value={"Candito"}>Candito 6 Week Strength</MenuItem>

<MenuItem value={"PHUL"}>PHUL</MenuItem>

<MenuItem value={"PHAT"}>PHAT</MenuItem>

<MenuItem value={"HST"}>HST</MenuItem>

<MenuItem value={"5x5"}>why do not 5x5?</MenuItem>

</Select>

</FormControl>

</div>

<div style={{ paddingTop: "20px" }}>

<Button variant="text" sx={{ color: "grey" }} onClick={getType}>

start

</Button>

<Button variant="text" sx={{ color: "grey" }} onClick={handleDelete}>

delete

</Button>

</div>

</div>

<div className="bg\_right\_bottom">

{type === "Candito" && <Explain1></Explain1>}

{type === "PHUL" && <Explain2></Explain2>}

{type === "PHAT" && <Explain3></Explain3>}

{type === "HST" && <Explain4></Explain4>}

{type === "5x5" && <Explain5></Explain5>}

</div>

</div>

<div className="bg\_right">

<div>

<h5 className="font\_content">

You've been stronger for 10 days, and go to the gym again today

</h5>

</div>

<div className="Calendar\_box">

<Calendar

locale="en-GB"

onChange={onChange}

value={value}

tileContent={({ date, view }) => {

const tile = tileContent.find(

(t) =>

isSameDay(t.date, date) &&

(view === "month" || view === "year") &&

t.content !== null

);

return tile?.content ?? null;

}}

/>

</div>

</div>

</div>

);

};

function addDays(date, days) {

const result = new Date(date);

result.setDate(result.getDate() + days);

return result;

}

function isSameDay(a, b) {

return (

a.getFullYear() === b.getFullYear() &&

a.getMonth() === b.getMonth() &&

a.getDate() === b.getDate()

);

}

function toLocalDateString(date) {

const year = date.getFullYear();

const month = String(date.getMonth() + 1).padStart(2, "0");

const day = String(date.getDate()).padStart(2, "0");

return `${year}-${month}-${day}`;

}

export default CalendarComponent;

[DiaryPage.js]

import React, { useState, useEffect } from "react";

import { useParams, useLocation } from "react-router-dom";

import axios from "axios";

import {

Input,

Checkbox,

Button,

TextField,

FormControl,

InputLabel,

MenuItem,

Select,

} from "@mui/material";

import "./DiaryPage.css";

import { useNavigate } from "react-router-dom";

const DiaryPage = () => {

const navigate = useNavigate();

const location = useLocation();

const programTodo = location.state?.programTodo;

const { date } = useParams();

const [event, setEvent] = useState("");

const [title, setTitle] = useState("");

const [memo, setMemo] = useState("");

const [maxWeight, setMaxWeight] = useState(0);

const [programData, setProgramData] = useState(null);

const [exerciseData, setExerciseData] = useState({ //다이어리 입력 내용 상태변수 관리

exercise1: { name: "", content: "", isChecked: false },

exercise2: { name: "", content: "", isChecked: false },

exercise3: { name: "", content: "", isChecked: false },

exercise4: { name: "", content: "", isChecked: false },

exercise5: { name: "", content: "", isChecked: false },

exercise6: { name: "", content: "", isChecked: false },

});

const handleTitle = (e) => setTitle(e.target.value);

const handleMaxWeight = (e) => setMaxWeight(e.target.value);

const handleChange = (e, exerciseKey) => {

const { name, value } = e.target;

setExerciseData({

...exerciseData,

[exerciseKey]: { ...exerciseData[exerciseKey], [name]: value },

});

};

const handleCheckbox = (e, exerciseKey) => {

const { checked } = e.target;

setExerciseData({

...exerciseData,

[exerciseKey]: {

...exerciseData[exerciseKey],

isChecked: checked,

name: checked

? programData.programs[exerciseKey.slice(-1) - 1]?.title

: "",

content: checked

? programData.programs[exerciseKey.slice(-1) - 1]?.content

: "",

},

});

};

const handleEvent = (e) => {

setEvent(e.target.value);

};

const handleMemo = (e) => setMemo(e.target.value);

const handleSubmit = async () => {

try {

const response = await axios.post("http://localhost:3001/exercises", { // eslint-disable-line no-unused-vars

userid: localStorage.getItem("userid"),

date: date,

title: title,

maxWeight: maxWeight,

event: event,

memo: memo,

...exerciseData,

});

alert("저장되었습니다.");

navigate("/");

} catch (error) {

console.error("Error sending data to server:", error);

}

};

useEffect(() => {

const fetchData = async () => {

try {

const response = await axios.post(

"http://localhost:3001/program\_data",

{ name: programTodo }

);

setProgramData(response.data);

} catch (error) {

console.error("Error fetching diary data:", error);

}

};

fetchData();

}, [programTodo]);

return (

<div>

<div className="container">

<div className="box-left">

<TextField

helperText="Enter the Title"

label="Title"

onChange={handleTitle}

></TextField>

<div style={{ float: "right" }}>

<div

style={{

float: "left",

margin: "4px",

paddingRight: "10px",

color: "#656D76",

}}

>

<h1>{date}</h1>

</div>

<Button variant="contained" onClick={handleSubmit}>

Save

</Button>

</div>

<div style={{ float: "right" }}></div>

<form className="diary" style={{ backgroundColor: "white" }}>

{Object.keys(exerciseData).map((exerciseKey, index) => (

<div style={{ width: "600px" }} key={exerciseKey}>

<Input

style={{ marginRight: "10px" }}

type="text"

name="name"

placeholder={

(programData && programData.programs[index]?.title) ||

"content"

}

value={exerciseData[exerciseKey].name}

onChange={(e) => handleChange(e, exerciseKey)}

/>

<Input

type="text"

name="content"

placeholder={

(programData && programData.programs[index]?.content) ||

"set/lap/weight"

}

value={exerciseData[exerciseKey].content}

onChange={(e) => handleChange(e, exerciseKey)}

/>

{programData && programData.programs[index] && (

<Checkbox

checked={exerciseData[exerciseKey].isChecked}

onChange={(e) => handleCheckbox(e, exerciseKey)}

/>

)}

</div>

))}

</form>

<div>

<div style={{ float: "left" }}>

<FormControl sx={{ m: 1, minWidth: 80 }}>

<InputLabel id="demo-simple-select-autowidth-label">

1RM

</InputLabel>

<Select

labelId="demo-simple-select-autowidth-label"

id="demo-simple-select-autowidth"

value={event}

onChange={handleEvent}

autoWidth

label="Event"

>

<MenuItem value="">

<em>None</em>

</MenuItem>

<MenuItem value={"Benchpress"}>Benchpress</MenuItem>

<MenuItem value={"Deadlift"}>Deadlift</MenuItem>

<MenuItem value={"Squat"}>Squat</MenuItem>

<MenuItem value={"O.H.P"}>O.H.P</MenuItem>

</Select>

</FormControl>

</div>

<div style={{ paddingTop: "8px" }}>

<TextField

label="1RM-WEIGHT"

type="text"

onChange={handleMaxWeight}

></TextField>

</div>

</div>

</div>

</div>

<div className="box-right">

<div className="header">

<div className="memo-box">

<div style={{float : "right", color : "#656D76", padding : "7px"}}>

<h3>Memo</h3>

</div>

<TextField

fullWidth

id="filled-multiline-static"

label="How was your workout today?"

multiline

rows={16}

variant="filled"

onChange={handleMemo}

/>

</div>

</div>

</div>

</div>

);

};

export default DiaryPage;