10월 3주차 (2024.10.14~2024.10.20)

FrontEnd

- 수정된 구역(24개 구역) 반영된 Map 제작 → 실시간 위치 모니터링 페이지에 적용

수정사항 1.

- **기존의 .png 파일에서 .svg 로 변경**
  - 변경 이유?
    - svg 파일은 vector 파일 포맷(픽셀로 이루어진 png 파일과는 다르게 라인과 곡선들로 이루어져 있음)

      - 파일을 확대하거나 줄였을 때 이미지가 깨지지 않고 그대로 선명함을 유지
    - svg 파일은 XML로 작성되기 때문에 텍스트 편집기에서 열고 편집이 가능 즉, 값을 조작 할 수 있기 때문에 js를 이용해서 svg 아이콘의 색상을 변경하거나 애니메이션을 적용시킬 수도 있어서 원하는 스타일에 맞게 변경이 가능

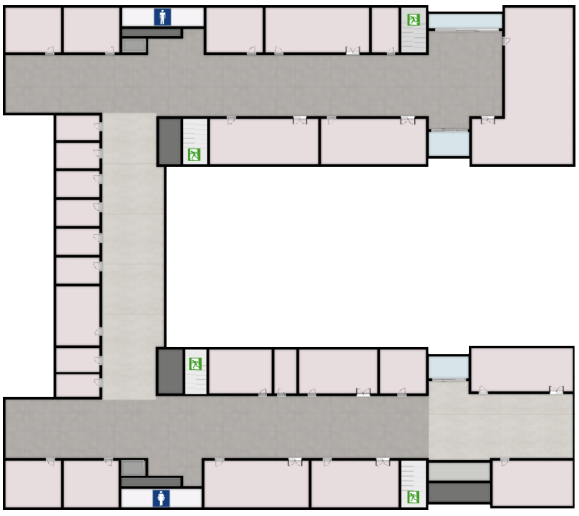
      → svg 파일의 특성을 활용하여 작업자가 위치한 구역과 다른 부분과 대비시키기 위해 해당 구역의 색상을 변경하고자 함.

수정사항 2.

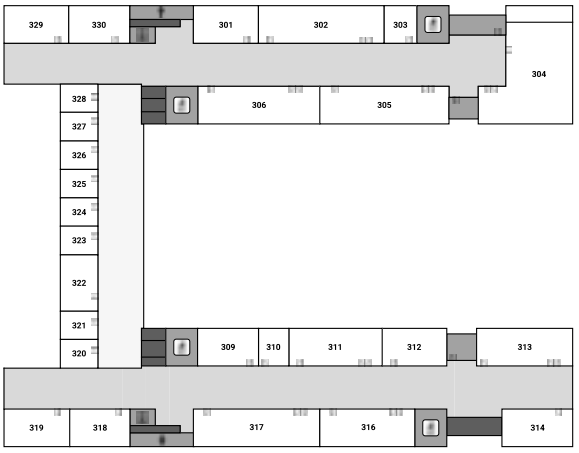
- **전체적인 Map 색상 변경(컬러 Map → 흑백 Map)**
  - 변경 이유?
    - 기존의 Map은 여러가지 색상이 존재하기 때문에 작업자 위치 모니터링 시에 시선이 분산 될 수 있었음.

    → 전체를 흑백으로 하면서 필요로 하는 특정 부분(ex - 작업자가 위치해있는 구역)만 눈에 잘 보이게 색상변경 하는 것으로 변경함.

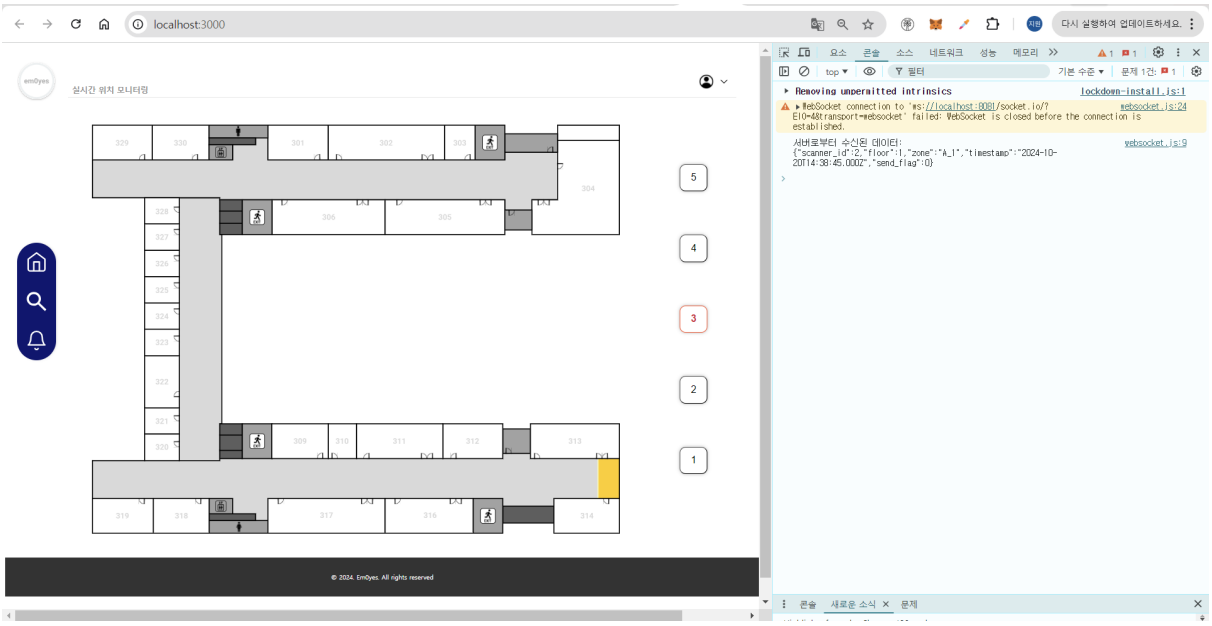
→ 수정 전



→ 수정 후



→ 실제결과 화면(실시간 위치 모니터링 페이지) (서버로 수신된 data를 통해 A_1구역이 변경된 모습)



추가예정- 작업자 아이콘

BackEnd

API명세서

로그인, 검색과 같은 **실시간 처리가 필요 없는 기능**들은 REST API로 구현

HTTP 기반의 요청-응답 방식으로, 서버와 클라이언트가 데이터를 교환하는 방법
(실시간 작업자 위치 표시와 같은 실시간 통신이 필수적인 기능은 WebSocket 통신으로 처리)

api명세서

index	Aa 기능	HTTP 메...	요청 URL	백엔드개...	프론트 개발현황
[관리자]	관리자 로그인	POST	/api/admin/login	진행 중	시작 전
	관리자 비밀번호 변경	POST	/api/admin/changepw	진행 중	시작 전
[작업자 설정]				시작 전	시작 전
	작업자 검색	GET	/api/worker/search	시작 전	시작 전
	스캐너 - 작업자 매핑	PATCH	/api/worker/setworker	시작 전	시작 전
[작업 위치]				시작 전	시작 전
	장소선택 (7호관)	GET	/api/building/select	시작 전	시작 전
	장소추가	POST	/api/building/add	시작 전	시작 전
				시작 전	시작 전

기존 회원가입 기능을 제거

→ 해당 웹 서비스는 관리자 전용으로, 일반 사용자가 회원가입을 통해 관리자 계정을 생성하는 것은 보안 상 바람직하지 않음

→ 관리자 계정은 미리 생성해두고, 로그인 기능과 비밀번호 변경 ㅏ기능만 제공하는 방식으로 수정 (서버 측에서 직접 생성, 데이터베이스에 기본적으로 하나 이상의 관리자 계정을 생성해두고, 이를 통해 시스템에 접근)

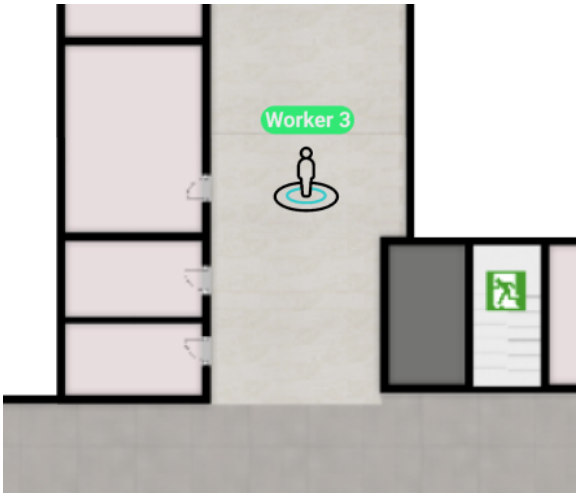
DB & 전송 데이터 수정

기존 'beacon_scanner' 테이블에 'worker' 속성을 추가함

Field	Type	Null
Filter...	Filter...	Filter...
id	int	NO
mac_address	varchar(255)	NO
device_name	varchar(255)	NO
worker	varchar(255)	YES

관리자가 왼쪽사진과 같이 해당 페이지에서 특정스캐너에 작업자의 이름을 작성하여 저장하면, 백엔드에서 ‘beacon_scanner’ 테이블에서 해당 스캐너의 worker 속성을 관리자가 작성한 이름으로 수정 및 추가할 수 있도록 수정

```
// 작업자 이름을 스캐너에 매핑
Beacon.setWorker = (scanner_id, worker_name, callback) => {
    const query = 'UPDATE beacon_scanners SET worker = ? WHERE
    connection.query(query, [worker_name, scanner_id], callba
};
```



기존 코드에서는 프론트 측에 위치 정보를 전송할 때, ‘estimated_locations’ 테이블에서 최신 데이터를 전송함. 따라서 scanner_id만 알 수 있었음.

현재는 worker 필드를 추가한 ‘beacon_scanner’를 JOIN하여 처리

→ 프론트측에서 worker정보 확인 가능. 즉, worker3가 실제 어떤 작업자인지 확인이 가능하다.

```
// 전송되지 않은 비콘 데이터를 가져오는 함수
 Beacon.getUnsentData2 = (callback) => {
     const query =
     'SELECT * FROM estimated_locations WHERE send_flag = FAL
     connection.query(query, callback);
 };
```

```
Beacon.getUnsentData2 = (callback) => {
    const query = `
        SELECT el.*, bs.worker
        FROM estimated_locations el
        JOIN beacon_scanners bs ON el.scanner_id = bs.id
        WHERE el.send_flag = FALSE
        ORDER BY el.timestamp ASC
        LIMIT 18;
    `;
    connection.query(query, callback);
};
```

향후계획

- api명세서 기반 백/프론트 구현