Software Requirements Specification Template(SRS)

작성자: 김성환, 이민형

생성일자 : 2022-10-21

최종 갱신: 2022-11-02

버전: v0.0.1

목차

1. 개요

'스마트 항구' 프로젝트와 프로젝트의 목적을 소개하고, '스마트 항구' 프로젝트에 사용되는 소프트웨어의 기능, 환경, 요구사항 등을 소개한다.

1.1 목적

'스마트 항구'란, '자동화된 항구'를 말한다. 자동화된 항구는 관리자(사용자)의 직접적인 기계 조작 없이 컨테이너를 쉽게 관리할 수 있게한다. 따라서, 기존 전통적인 항구를 대체하여 화물을 선박에서 육지로 이동하는 과정부터, 화물을 보관하는 과정까지 자동화하여 사용자가 쉽게 항구를 관리할 수 있는 항구를 제작하는 것이 본 프로젝트의 목적이다. 실제 크기로 제작하기에는 현실적으로 불가능하다. 따라서 본 프로젝트는 항구를 추상화하고 그것을 실제 항구로 여겨, '관리자가 쉽게 컨테이너를 관리할 수 있다'라는 목적이 성공하면 프로젝트는 마무리된다.

1.2 관련부서/관계자

국내에 '스마트 항구' 기술을 도입하고자 하는 물류/운송 업계 종사자들

(ex. 해양 수산부의 보도자료를 통해 해양 수산부는 자동화 항만 도입을 위한 기술 개발을 착수하고 있음을 확인 할 수 있다.)

'항만의 무인 자동화' 기술에 관심을 가지는 종사자들

IoT로 구현한 '물류 운송 기술' 기술에 관심을 가지는 기술자들

1.3 개발범위

사용자는 웹을 통해 항구를 간단하게 관리할 수 있게 된다. 이를 실현하는 것이 본 프로젝트의 목적이자 개발 범위이다.

2. 전체개관

용어 정의를 시작으로 간단한 프로젝트 구조를 설명한다. 그 다음, 본 프로젝트는 하드웨어적/소프트웨어적으로 무엇이 요구되는지 소개하겠다.

2.1 프로젝트 개요

용어 정의부터 하겠다.

- 트랜스퍼 크레인 : 주로 실외에 설치되는 크레인으로 지상에 주행레일을 설치하고 다리구조물을 세워 그위에 거더를 설치하는 방식의 크레인으로 실외의 넓은 작업공간이나 하역작업을 위해 설치하는 크레인을 말한다. 본 프로젝트에서, 트랜스퍼 크레인은 컨테이너 운송 로봇에 컨테이너를 오르고 싣는 역할을 한다.

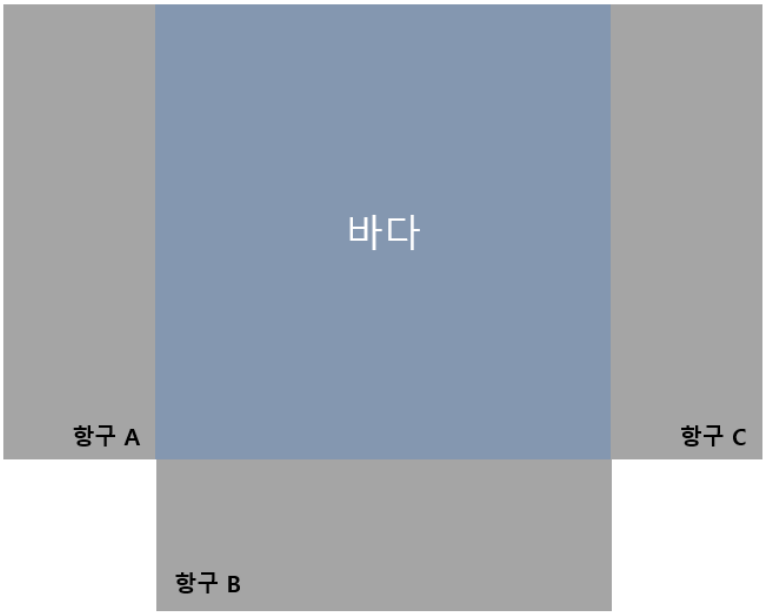
- 갠트리 크레인 : 배에 있는 컨테이너를 컨테이너 운송 로봇에 오르고 싣는 역할을 한다.

- 컨테이너 운송 로봇 : 컨테이너의 입장에서, 갠트리 크레인과 트랜스퍼 크레인의 중간자 역할을 한다. 즉, 갠트리 크레인이 동작하는 공간과 트랜스퍼 크레인이 동작하는 공간을 이어주는 역할을 한다.

- 트레일러 운송 배 : 컨테이너를 운송할 배를 의미한다.

간단한 프로젝트 구조를 설명하겠다. 본 프로젝트를 마치면, 사용자는 웹을 통해 항구를 간단하게 관리할 수 있게 된다. 따라서, '웹 어플리케이션 서버'가 필요하다. 그리고 사용자가 웹을 조작하면 진행과정을 보여줄 추상화된 '항구'가 필요하다. 다시, 항구에는 앞서 얘기한 트랜스퍼 크레인과 갠트리 크레인 등 여러가지 추상화된 '컨테이너 운반 기계'들이 요구된다.

작품의 전체적인 제작 형태는 'ㅜ'자 형태이다. 가운데에 정사각형 모양의 바다가 있고, 바다의 3개의 변에 항구가 위치한다. 이해가 안 된다면, 아래의 사진을 참고하면 된다.



하드웨어적으로는 다음이 요구된다.

갠트리 크레인, 트랜스퍼 크레인은 각각 항구당 1개, 즉 3개가 필요하다. 또한, 추가적으로 컨테이너 운송 로봇은 3개, 컨테이너 N개, 트레일러 운송 배 1개가 필요하다. 컨테이너 개수는 하한/상한이 없다.

소프트웨어적으로는 다음이 요구된다.

갠트리 크레인, 트랜스퍼 크레인, 컨테이너 운송 로봇, 트레일러 운송 배를 중앙제어적으로 관리할 항구 서버(웹 서버)가 요구된다. 데이터베이스로는 Django의 SQLite를 활용한다. 클라이언트쪽은 바닐라 CSS, 바닐라 JS를 선택한다. 상황이 허락한다면 React와 같은 자바스크립트 라이브러리와 Material UI나 Bootstrap과 같은 CSS 프레임워크를 사용한다. 항구 서버(웹 서버)와 나머지 하드웨어적 객체들 간의 관계는 3.3 소프트웨어 인터페이스를 참고하길 부탁드린다.

2.2 Product Function

3.3 소프트웨어 인터페이스에 상세 기술하였다.

2.3 사용자 특성

본 프로젝트의 웹은 항구의 관리자가 사용한다. 기존의 항구 관리자는 항구에 사용되는 장비들에 대한 이해도, 컨테이너 운송 기계에 대한 숙련도 등이 높을 것으로 예상된다. 하지만 본 프로젝트를 달성함으로서 그러한 능력과 이해가 없는 사용자도 항구를 쉽게 제어할 수 있게 된다. 따라서 기존의 항구 관리자는 보다 안전하게 작업할 수 있게 된다.

2.4 소프트웨어 운영환경

웹 클라이언트) - 프론트엔드

하드웨어 : 스마트폰, 노트북, 데스크탑 등 웹 브라우저가 동작하는 환경

OS : 윈도우, 맥, 리눅스 등 웹 브라우저가 동작하는 대부분의 OS

소프트웨어 환경 : Chrome

사용 프로그래밍 언어 : Javascript

사용 자바스크립트 라이브러리 : vanilla JS

사용 CSS 프레임워크 : vanilla CSS

항구 서버 : 웹 서버) - 백엔드

하드웨어: Mac book

OS: Mac OS

사용 프로그래밍 언어: Python

사용 웹 어플리케이션 프레임워크 : Django

트랜스퍼 크레인, 갠트리 크레인, 컨테이너 운송 로봇, 트레일러 운송 배의 소프트웨어)

하드웨어: ESP8266

SDK: Arduino 1.8.18

사용 프로그래밍 언어: C++

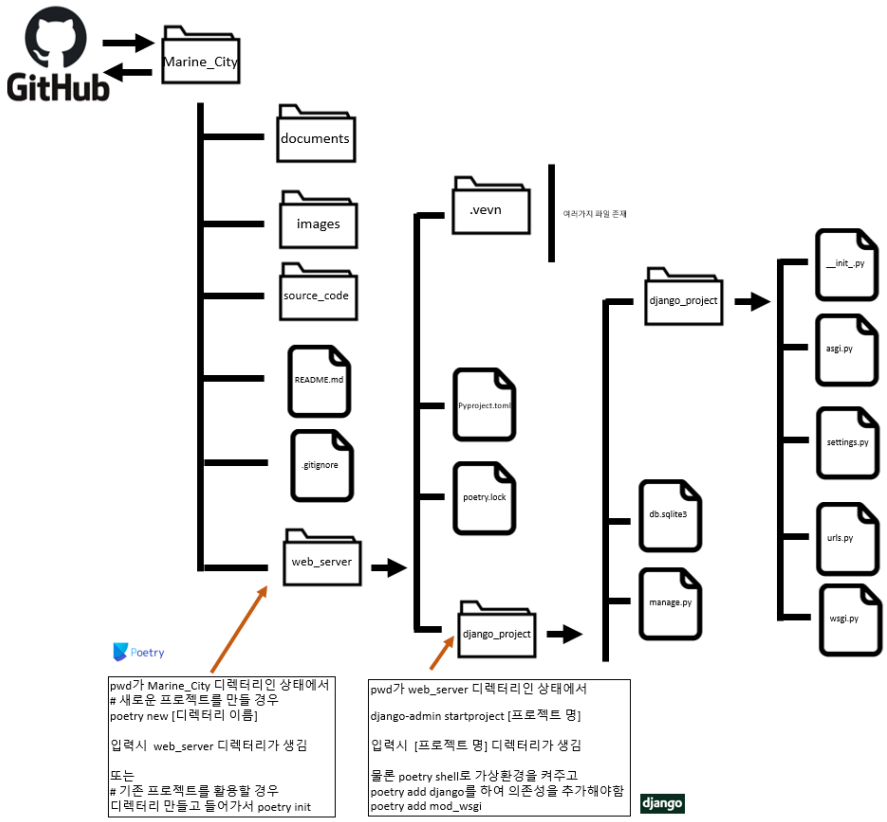
2.5 디자인과 배포 제약사항

- 프로그래밍 규약

1. .html, .css, .js, .py 등 정말 다양한 확장자의 파일을 다룬다. 따라서 언어와 작성자의 코드 컨벤션을 존중한다.

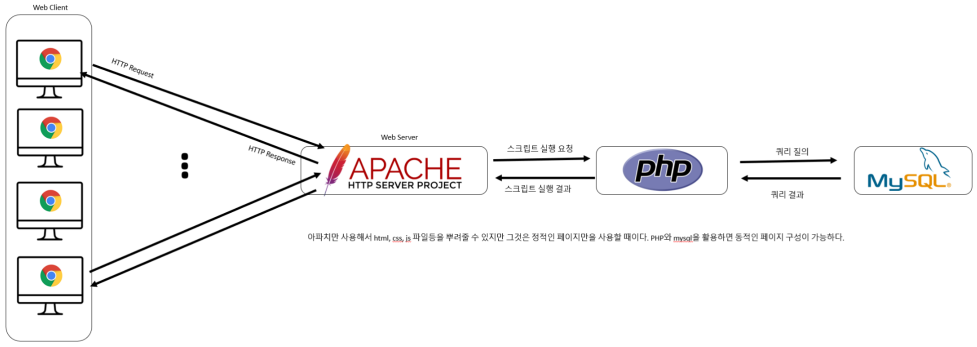
- 특정 기술 사용

1. 버전 관리/이슈 관리는 깃허브를 활용한다. 아래의 사진은 "대략적인" 디렉터리 구조이다. UI/UX를 비롯한 Django 활용 방안에 대한 구체적 계획은 반영하지 않았다.



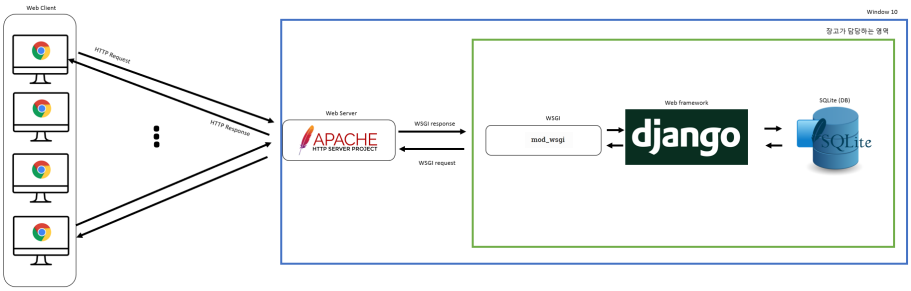
2. (파이썬 패키지/라이브러리 간의 호환성 이슈 예방) 파이썬 패키지/라이브러리 의존성 관리는 Poetry를 이용한다. Poetry를 사용하여 프로젝트가 사용하고 있는 파이썬 라이브러리들을 관리한다. 프로젝트의 협업자들은 충돌없이 작업할 수 있다는 장점이 생긴다.
3. 웹 서버 선택 과정

4.1 웹 서버를 구축하기 위해 APM(APACHE + PHP + MYSQL)환경을 구성하는게 정석적이라고 본 기억이 있다. 아래 다이어그램은 ARM 기반 웹 서버 구축의 FLOW를 나타낸다.

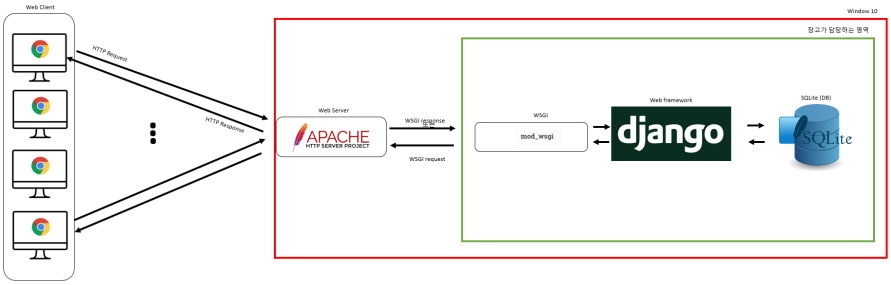


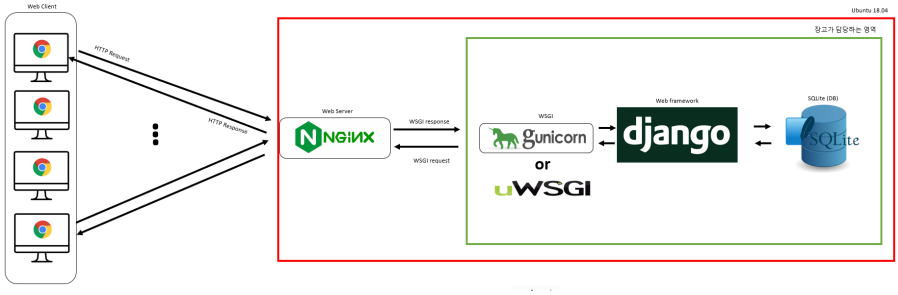
이러한 APM 환경을 구축하는 것도 나쁘지 않은 선택이라고 생각한다. 하지만, PHP 언어와 SQL문 등 공부해야할 분량이 늘어난다고 0부터 구현해야해서 구현할 양이 늘어난다고 생각했다. 따라서, PHP와 MySQL을 대체하여 파이썬 Django 웹 어플리케이션 프레임워크를 사용하기로 결정했다.

4.2 윈도우 운영 체제에서 Django 프레임워크를 사용하면 아래의 FLOW만 가능함을 확인했다.



4.3 Unix-like 운영 체제에서 Django 프레임워크를 사용하면 아래의 FLOW들이 가능하다.





공식 문서에서는 apache + mod_wsgi + django의 조합을 소개하고 있다. 데이터베이스는 선택적인데, 위 다이어그램처럼 SQLite를 사용할 수 있고 postgreSQL을 사용할 수 있다. 우리가 수많은 데이터를 처리할 것도 아니고 더군다나 postgreSQL를 사용하면 Django와 postgreSQL을 연동시켜줘야하기 때문에 부담이 늘어난다. 따라서, SQLite를 이용한다.

결과적으로, nginx + gunicorn + django 조합을 선택한다.

2.6 가정과 의존성

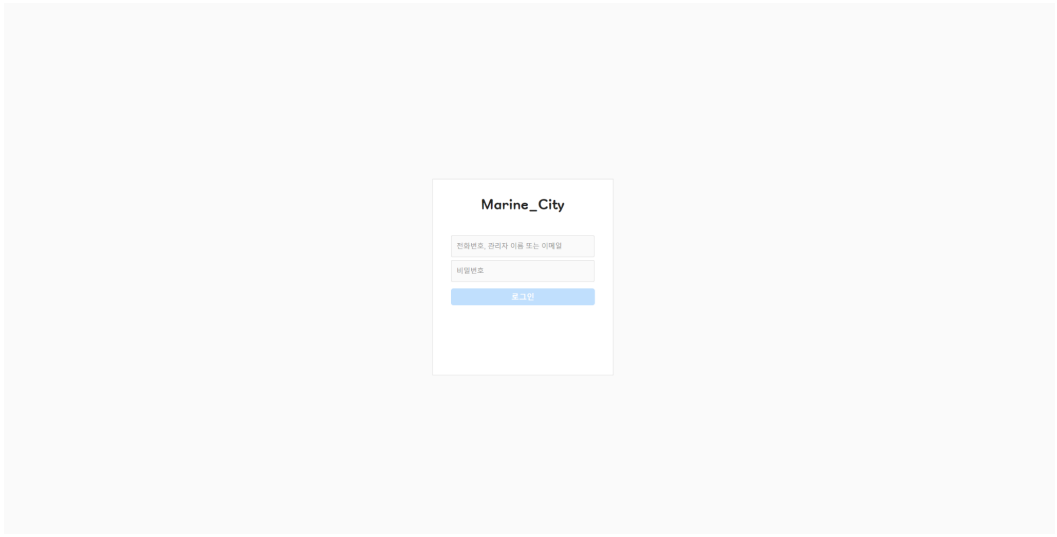
1. 관리자는 로그인을 통해 특정 항구를 제어할 수 있는 권한을 가진다. 즉, 다른 관리자는 자신의 항구가 아닌 항구를 제어할 수 없다.
2. 관리자의 요청에 의해 발생하는 모든 태스크는 QUEUE로 관리한다. 즉, 현재 진행중인 태스크가 다 마무리되기 전까지는 다음 태스크를 절대 실행하지 않는다.
3. 진행중인 태스크가 마무리되면, 항구의 구성 요소들은 초기 위치로 돌아간다.
4. 프로젝트의 진행에 따라 본 가정/의존성은 추가/변동될 수 있음을 밝힌다.

2.7 데이터 요구사항

웹 서버의 데이터베이스는 Django를 활용한다. 그 외는 확답하기 어렵다. 구현해야할 것이 꽤 많고, 더군다나 UI/UX를 아직 의논하지 않아 조심스러운 부분있기 때문이다. 즉, UI/UX가 결정되었다는 것은 '관리자가 쉽게 항구를 관리할 수 있는 방법을 결정했다'와 동치 표현이다. 그 방법을 아직 의논하지 않았다.

2.8 사용 시나리오

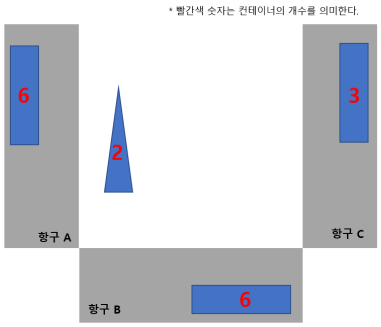
웹에 접속했을 때의 초기화면은 다음과 같다.



관리자는 로그인을 통해, 자신의 항구를 관리할 수 있는 권한을 얻는다.

알림을 통해 다른 관리자들의 요청에 응답할 수 있고, 지난 요청에 대한 결과를 확인할 수 있다. 지금의 이미지에서는 드러나진 않지만, 간단한 UI조작을 통해 다른 관리자들에게 컨테이너를 요청할 수 있는 등의 다양한 작업이 가능하다.

이는 웹 상에서 '추상적'으로 작동하는 것이다. 우리가 제작하는 작품에서는 다음과 같이 동작한다.



명령 : 항구 A에 있는 컨테이너 3개를 항구 C로 옮겨라.

‘항구 A’가 ‘부산항’, ‘항구 C’가 ‘싱가포르 항구’를 나타낸다고 가정하자. 부산항의 컨테이너를 관리할 수 있는 권한을 가진 관리자는 싱가포르 항구에 “항구 A에 있는 컨테이너 3개를 항구 C로 옮겨라”를 명령한다. 이는 적절한 전처리를 통해 SQL문으로 변환(ORM)되거나 특정한 자바스크립트 함수가 호출되며 싱가포르 항구 관리자의 계정으로 접속했을 때, 알림으로 다음과 같은 내용이 올 수 있다. “부산항의 3개의 컨테이너가 싱가포르 컨테이너에 입항하고자 합니다. 허락하시겠습니까? (Y/N)” 싱가포르 항구 관리자가 “Y”를 선택했다고 가정하자. 현재 모든 장치가 멈춰있다면, 작업을 수행한다. 멈춰있지 않다면, 선행된 작업이 끝날 때까지 기다린다. 위 태스크는 QUEUE로 관리하게 된다.

3. 인터페이스

3.1 사용자 인터페이스

웹에서, 간단한 조작으로 컨테이너를 관리한다. 다음의 사진은 대략적인 웹의 형태를 보여준다.

현재 '부산항'을 관리하는 계정에 접속해있다.

그림에서는 나타나진 않았지만, 상단의 메뉴바를 통해 간단한 터치로 컨테이너를 관리할 수 있다. 또한, '알림' 탭을 통해 변경사항을 확인할 수 있고 다른 항구 관리자의 요청에 응답할 수 있다. 위 기능은 빠질수도, 추가될 수도 있음을 밝힌다.

3.2 하드웨어 인터페이스

웹 클라이언트 - 웹 서버)

사용자의 명령을 터치로 입력받아 웹 서버에 그 명령을 전송한다.

사용 프로토콜: HTTP

웹 서버 - 트랜스퍼 크레인)

사용 프로토콜:

소프트웨어 제품과 시스템의 하드웨어 구성 요소 사이의 각 인터페이스의 논리적 및 물리적 특성을 설명합니다. 여기에는 지원되는 장치 유형, 소프트웨어와 하드웨어 간의 데이터 및 제어 상호작용의 특성, 사용되는 통신 프로토콜이 포함될 수 있습니다.

3.3 소프트웨어 인터페이스

- 항구 서버
  - 개요 : 컨테이너 상태 테이블 / 컨테이너 정보 테이블. 그리고 그 테이블에 관한 CRUD 함수와 QR 코드 검출함수가 필요할 것이다. 다만, Django를 활용하면 ORM(Object-Relational Mapping)을 사용하여 SQLite 데이터베이스를 운용할 수 있기 때문에 대부분의 것들은 직접 구현할 필요가 없어진다.

  - 컨테이너 상태 테이블(container State table) / 컨테이너 정보 테이블(container Information table)

    1) 두 테이블 모두 QR코드의 일련 번호를 고유 키로 가지는 테이블이다. 편의상 용어를 나눈 것일 뿐, 두 테이블은 하나의 테이블로 운영할 수 있다.
    2) 컨테이너 상태 테이블 : 어떤 컨테이너가 어떤 항구에 몇 번째 열, 행, 높이에 있는지에 관한 상태를 가지고 있다.
    3) 컨테이너 정보 테이블 : 제품명, 제조사, 제조일시, 발송지에 관한 정보를 가지고 있다.

  - 컨테이너 상태 테이블 - 생성 함수(container State Table - CREATE function) / 컨테이너 정보 테이블 - 생성 함수(container Info Table - CREATE function)

1) 사용자로부터 쿼리를 받아 테이블에 데이터(레코드)를 생성(추가)하는 역할을 한다.
2) 새로운 물품에 대한 QR코드를 할당받고, 사용자가 입력한 쿼리문으로부터 컨테이너 상태 테이블 / 컨테이너 정보 테이블에 레코드를 추가한다.

- 컨테이너 상태 테이블 - 읽기 함수(container State Table - READ function) / 컨테이너 정보 테이블 - 생성 함수(container Info Table - READ function)

1) 다른 함수에 호출로 인해, 요청한 정보(레코드)를 반환하는 함수이다.

- 컨테이너 상태 테이블 - 갱신 함수(container State Table - UPDATE function) / 컨테이너 정보 테이블 - 갱신 함수(container Info Table - UPDATE function)

1) 사용자로부터 쿼리를 받아 테이블의 레코드를 갱신하는 역할을 한다.
2) (예시) 컨테이너 상태 테이블 - 갱신 함수 : 사용자는 일련번호 0x2515213의 컨테이너를 기존의 위치와 다른 항구로 보내는 쿼리를 작성한다. 위 함수는 사용자의 쿼리를 바탕으로 컨테이너 상태 테이블을 갱신한다.
3) (예시) 컨테이너 정보 테이블 - 갱신 함수 : 사용자는 기존에 A물품을 보관하던 컨테이너에 A물품을 빼고 새로운 물품 B를 넣고싶다는 쿼리를 작성한다. 위 함수는 사용자의 쿼리를 바탕으로 컨테이너 정보 테이블을 갱신한다. 컨테이너에 그려진 QR코드가 같더라도 웹 상에서는 일련번호가 다르도록 처리할 것이다.

- QR 코드 검출 함수

1) 트랜스퍼 크레인, 트레일러 운송 배로부터 전송받은 이미지에서 QR코드를 검출하고 '컨테이너 상태 테이블 - 갱신 함수'를 호출한다. '컨테이너 정보 테이블 - 갱신 함수'는 호출할 필요가 없다.

- 컨테이너 운송 로봇
  - 항구 서버 함수

1) 항구 서버로부터 명령을 순차적으로 받는다. 컨테이너 운송 로봇은 명령에 따라 적절한 함수들을 단순히 호출한다. 즉, 제어권은 항구 서버에게 있다고 할 수 있다.

  - 라인 트레이서 함수

1) 항구 서버로부터 신호를 받으면 주행을 준비한다.
2) 컨테이너 운송 로봇의 주행을 위한 함수이다. 사용자의 명령에 따라 주행을 시작한다.
3) 목적지에 도달했음을 알리는 신호를 항구 서버로 전송한다.

  - 컨테이너 운송 로봇 - 위치 함수, 방향 함수

1) 컨테이너 운송 로봇 - 위치 함수 : 컨테이너 운송 로봇이 갠트리 크레인에 위치해있는지 또는 트랜스퍼 크레인에 위치해있는지 반환한다. 그곳들 중 아무것도 아닐 수 있다. 즉, 3가지 상태를 가질 수 있다.
2) 컨테이너 운송 로봇 - 방향 함수 : 컨테이너 운송 로봇이 현재의 방향으로 주행을 계속한다면 갠트리 크레인에 도달하는지 또는 트랜스퍼 크레인에 도달하는지 반환한다.

  - 컨테이너 운송 로봇 - 위치 초기화 함수

1) 컨테이너 운송 로봇 위치 함수 / 컨테이너 운송 로봇 방향 함수를 바탕으로 컨테이너 운송 로봇을 초기 위치로 돌려놓는 역할을 하는 함수이다. 비효율적이겠지만, 위 함수를 작성함으로써 사용자의 비정상적인 작동을 예방할 수 있다.

- 갠트리 크레인
  - 컨테이너 적재 함수

1) 항구에 있는 컨테이너를 컨테이너 운송 로봇에 컨테이너를 오르고/싣는 역할을 한다. 오르고/싣는 역할을 할때, 항구 서버로부터 작업을 완료했다는 시그널을 보낸다.
2) 항구에 위치한 컨테이너를 분류/정렬하고 효율적인 컨테이너 관리를 수행한다.

  - 주행레일 함수

1) 주행을 위한 함수이다. 절대적인 좌표를 입력받으면 그 위치로 이동한다.

  - 갠트리 크레인 위치 - 위치 초기화 함수

1) 갠트리 크레인을 초기 위치로 돌려놓는 역할을 하는 함수이다. 비효율적이겠지만, 위 함수를 작성함으로써 사용자의 비정상적인 작동을 예방할 수 있다.

- 트랜스퍼 크레인
  - 이미지 전송 함수

1) 현재 전자석에 붙일 컨테이너의 사진을 찍고 항구 서버로 보내는 함수를 작성한다.

- 컨테이너 적재 함수

1) 일차적으로는, 배에 있는 컨테이너를 컨테이너 로봇에 컨테이너를 오르고/싣는 역할을 한다.
2) 나아가서, 배에 위치한 컨테이너를 분류/정렬하고 효율적인 컨테이너 관리를 수행한다.

- 트랜스퍼 크레인 위치 - 위치 초기화 함수

1) 트랜스퍼 크레인을 초기 위치로 돌려놓는 역할을 하는 함수이다. 비효율적이겠지만, 위 함수를 작성함으로써 사용자의 비정상적인 작동을 예방할 수 있다.

- 트레일러 운송 배
  - 주행 함수
    1) 웹 서버로 부터 명령받아, 특정 항구로 이동한다.

4. 기능 요구사항

4.1 <Function name>

Django 프레임워크를 사용한다. 따라서, Django가 규정하는 디렉터리 구조를 따른다. 함수 구조/반환 형식/이름도 Django가 규정하는 형식을 따른다.

ESP8266에는 Nginx과 HTTP 통신을 위한 함수를 작성한다. 따라서, 다음 헤더파일을 포함한다. 따라서, 각 헤더파일이 규정하는 객체와 멤버 함수들의 Purpose, inputs, outputs과 동일하다.

```
#include <ESP8266WiFi.h>
```

```
#include <ESP8266HTTPClient.h>
```

```
#include <WiFiClient.h>
```

5. 비 기능 요구사항

5.1 성능 요구사항

본 프로젝트에서 성능은 목적에 두지 않는다.

5.2 보안 요구사항

본 프로젝트에서 보안은 목적에 두지 않는다.

5.3 소프트웨어 품질 요구사항

본 프로젝트에서 소프트웨어 품질은 목적에 두지 않는다.