

Software Requirements Specification

# 코드 탄소 배출량 측정 시스템

Team 6

김선우, 설현원, 이찬구, 이현민, 조재범, 한수민

# Software Requirements Specification

## 코드 탄소 배출량 측정 시스템

Team 6

김선우, 설현원, 이찬구, 이현민, 조재범, 한수민

Instructor: 이은석

Document Date: 29 Oct, 2023

Faculty: SungKyunKwan University

# Contents

|  |    |
|--|----|
| <b>1. Introduction</b>                       | 1  |
| 1.1 Purpose                                  | 1  |
| 1.2 Scope                                    | 1  |
| 1.3 Definitions, Acronyms, and Abbreviations | 1  |
| 1.4 References                               | 2  |
| 1.5 Overview                                 | 2  |
| <b>2. Overall Description</b>                | 3  |
| 2.1 Product Perspective                      | 3  |
| 2.1.1 System Interfaces                      | 3  |
| 2.1.2 User Interfaces                        | 4  |
| 2.1.3 Hardware Interfaces                    | 4  |
| 2.1.4 Software Interfaces                    | 4  |
| 2.1.5 Communication Interfaces               | 4  |
| 2.1.6 Memory                                 | 5  |
| 2.1.7 Operations                             | 5  |
| 2.1.8 Site Adaptation Requirements           | 5  |
| 2.2 Product Functions                        | 5  |
| 2.2.1 코드 입력 방법 제공                            | 6  |
| 2.2.2 코드 정상 실행 여부 확인                         | 6  |
| 2.2.3 탄소 배출량 제공                              | 6  |
| 2.2.4 탄소 배출량에 상응하는 환경적 척도 제공                 | 6  |
| 2.3 User Classes and Characteristics         | 6  |
| 2.3.1 User Classes                           | 6  |
| 2.3.2. Characteristics                       | 6  |
| 2.4 Design and Implementation Constraints    | 7  |
| 2.5 Assumptions and Dependencies             | 7  |
| <b>3. External Interface Requirements</b>    | 8  |
| 3.1 External Interfaces                      | 8  |
| 3.1.1 User Interface                         | 8  |
| 3.1.2 Hardware Interface                     | 9  |
| 3.1.3 Software Interface                     | 9  |
| 3.1.4 Communication Interface                | 10 |
| 3.2 Function Requirement                     | 10 |
| 3.2.1 User Case                              | 10 |
| 3.2.2 Use Case Diagram                       | 11 |

|  |           |
|--|-----------|
| 3.2.3 Data Dictionary .....                    | 11        |
| 3.2.4 Data Flow Diagram .....                  | 12        |
| 3.3 Performance Requirements .....             | 13        |
| 3.3.1 Static Numerical Requirement .....       | 13        |
| 3.3.2 Dynamic Numerical Requirement .....      | 13        |
| 3.4 Logical Database Requirements .....        | 13        |
| 3.5 Design Constraints .....                   | 14        |
| 3.5.1 Physical Design Constraints .....        | 14        |
| 3.5.2 Standards Compliance .....               | 14        |
| 3.6 Software System Attributes .....           | 14        |
| 3.6.1 Reality .....                            | 14        |
| 3.6.2 Availability .....                       | 14        |
| 3.6.3 Security .....                           | 15        |
| 3.6.4 Maintainability .....                    | 15        |
| 3.6.5 Portability .....                        | 15        |
| 3.7 Organizing the Specific Requirements ..... | 16        |
| 3.7.1 System Architecture .....                | 16        |
| 3.7.2 Process Model .....                      | 16        |
| 3.8 System Evolution .....                     | 17        |
| 3.8.1 Limitation and Assumption .....          | 17        |
| 3.8.1.1 고정된 실행 서버 정보 .....                     | 17        |
| 3.8.1.2 지원 가능한 프로그래밍 언어의 부족 .....              | 17        |
| 3.8.1.3 GPU 사용 제한 .....                        | 17        |
| 3.8.1.4 악의적인 사용자 .....                         | 17        |
| 3.8.2 Evolution .....                          | 18        |
| 3.8.2.1 다양한 프로그래밍 언어 지원 확대 .....               | 18        |
| 3.8.2.2 서버 환경의 다양화 .....                       | 18        |
| 3.8.2.3 악성 코드 필터링 기능 추가 .....                  | 18        |
| 3.8.2.4 사용자 코드 분석 및 피드백 제공 .....               | 18        |
| 3.8.2.5 기계 학습 모델의 도입 .....                     | 18        |
| 3.8.2.6 사용자 제출 코드 데이터 활용 .....                 | 18        |
| <b>4. Supporting Information .....</b>         | <b>19</b> |
| 4.1 Software Requirements Specification .....  | 19        |
| 4.2 Document History .....                     | 19        |

# List of Figures

|                                     |    |
|-------------------------------------|----|
| 2.1 Application Function Flow ..... | 5  |
| 3.1 Use Case Diagram .....          | 11 |
| 3.2 Data Flow Diagram .....         | 12 |
| 3.3 System Architecture .....       | 16 |
| 3.4 Process Model .....             | 16 |

# List of Tables

|   |    |
|---|----|
| 1.1 Table of acronyms and abbreviations .....                         | 1  |
| 1.2. Table of terms and definitions .....                             | 1  |
| 3.1 User Interface of uploading code .....                            | 8  |
| 3.2 Hardware Interface of applicable device for the system .....      | 9  |
| 3.3 Software Interface of applicable device for the system .....      | 9  |
| 3.4 Communication Interface of applicable device for the system ..... | 10 |
| 3.5 Use Case of 탄소 배출량 계산 .....                                       | 10 |
| 3.6 User .....  | 11 |
| 3.7 Carbon .....  | 11 |
| 3.8 Sample .....  | 12 |

# 1

## Introduction

### 1.1. Purpose

이 문서는 소프트웨어 엔지니어를 대상으로 Java 코드의 탄소 배출량 측정을 도와주는 프로그램의 requirement 를 명확히 하고, 추후 유지 보수하는 과정에서 참고할 수 있도록 제작되었다.

### 1.2. Scope

본 소프트웨어 제품은 Java 코드의 탄소 배출량 측정 웹 플랫폼이다. 본 제품은 사용자의 Java 코드의 탄소 배출량을 측정하고 이를 시각적으로 표시하는 기능을 포함한다. 본 제품의 목적은 환경 친화적 개발을 통하여 소프트웨어 개발 생태계의 지속 가능성을 높이고자함에 있다.

### 1.3. Definitions, Acronyms, and Abbreviation

| Acronyms & Abbreviations | Explanation                   |
|--------------------------|-------------------------------|
| CPU                      | Central Processing Unit       |
| JRE                      | Java Runtime Environment      |
| HTTP                     | Hypertext Transfer Protocol   |
| TCP                      | Transmission Control Protocol |
| RAM                      | Random Access Memory          |
| OS                       | Operating System              |

Table 1.1: Table of acronyms and abbreviations

| Terms       | Definitions                |
|-------------|----------------------------|
| 탄소 배출량      | 소프트웨어 실행에 따른 탄소 배출량        |
| Code Editor | 코드 입력 창으로, 간단한 ide 기능을 제공함 |

Table 1.2: Table of terms and definitions

## 1.4. References

IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications, In  
IEEE Xplore Digital Library

<http://ieeexplore.ieee.org/Xplore/guesthome.jsp>

2023 Spring 41 class team 6

[https://github.com/skkuse/2023fall\\_41class\\_team6.git](https://github.com/skkuse/2023fall_41class_team6.git)

## 1.5. Overview

본 소프트웨어 요구사항 명세서는 네 챕터로 구성되어 있다. 첫 번째 챕터에서는 코드 탄소 배출량 측정 시스템과 그 목적에 대한 소개 및 본 문서에서 사용하는 약어 및 용어에 대한 설명을 하고 있다. 두 번째 챕터에서는 product perspective 와 system interface, 제약 사항 등을 소개하며 전체적인 설명을 제공한다. 세 번째 챕터에서는 외부 interface 에 대한 requirements 와 함께 자세한 시스템 요구 사항을 정의한다. 마지막 챕터에서는 본 SRS 의 작성 기준과 팀원의 명세서 작성 관련 역할을 알려준다.



## 2

## Overall Description

### 2.1. Product Perspective

본 시스템의 목적은 유저(JAVA 프로그래머)를 대상으로 단일 파일 JAVA 소스코드의 잠재적 탄소배출량을 제공하는 것이다. 특히, 유저의 특별한 조작없이 단순히 소스코드를 입력하거나 업로드하는 것만으로 탄소배출량을 확인할 수 있도록 설계되었다. 유저는 원하는 소스코드를 editor 에 입력하거나 버튼을 눌러 원하는 소스코드 파일을 업로드 할 수 있다. 입력된 소스코드의 탄소배출량은 소스코드의 CPU runtime 과 Memory Usage 를 기반으로 산출된다. 유저는 소스코드에 대응되는 탄소배출량 결과를 얻을 수 있고 해당 탄소배출량에 상응하는 환경적 척도(차량이나 항공기 매연 등)를 확인할 수 있다. 결과적으로 유저들에게는 본 시스템을 도입함으로써, 소스코드간 환경친화적 관점에서의 객관적 비교 능력 획득이 기대된다.

#### 2.1.1. System Interfaces

본 시스템은 유저가 소스코드를 입력하고 탄소배출량을 확인하는 web application과 입력된 소스코드를 기반으로 탄소배출량을 계산하는 server로 구성된다. 유저가 web application에 접근하여 JAVA 소스코드를 입력하면, http 통신을 통해 server로 전달된다. server는 server내에 포함된 JRE(JAVA 런타임 환경)로 소스코드의 CPU 실행시간을 측정한다. server는 측정된 실행시간을 기반으로 탄소배출량과 그에 상응하는 환경적 척도를 산출한다. server는 http 통신으로 web application에 탄소배출량과 환경적 척도 값을 전달한다. 유저는 web application에 표현된 탄소배출량을 확인할 수 있다.

- server는 JRE에서 빈 파일이나, 컴파일/런타임 에러를 유발하는 소스코드의 컴파일이 시도되었을 때, available해야 한다.
- server는 7일간 downtime이 30분 이하여야 한다.

### 2.1.2. User Interfaces

유저 인터페이스는 퍼스널 컴퓨터의 web 형태로 제공된다. 유저들은 퍼스널 컴퓨터의 브라우저 소프트웨어로 본 시스템의 web application에 접근할 수 있다. 유저는 web application의 메인 페이지 상단 editor에 소스코드를 입력하거나, 버튼을 눌러 소스코드 파일을 업로드할 수 있다. 유저는 입력한 소스코드의 탄소배출량과 그에 상응하는 환경적 척도들을 확인할 수 있다.

- 유저가 사용하기 쉽게 만들어져야 하며, 별도의 훈련없이 사용할 수 있어야 한다.
- 유저는 64KB이하의 코드 파일을 제출할 수 있어야 한다.
- 유저는 입력 또는 업로드를 완료하고 탄소배출량을 요청한 후, 3분 안에 탄소배출량을 제공받을 수 있어야 한다.
- 유저는 빈 파일이나, 컴파일/런타임 에러를 유발하는 소스코드를 입력 시에, 에러메시지를 받을 수 있어야 한다.

### 2.1.3. Hardware Interfaces

본 시스템의 web application은 퍼스널 컴퓨터로 접근할 수 있다. Web application이 브라우저 위에서 작동하기 때문에 브라우저를 운용하기 위한 Intel Pentium 4 프로세서 또는 SSE3 1.0GHz 이상의 프로세서가 요구된다.

본 시스템의 server는 server computer에서 운용된다. Server는 node.js에 종속적이기 때문에 node.js를 운용하기 위한 Intel Pentium 4 이상의 싱글 프로세서가 요구된다.

### 2.1.4. Software Interfaces

본 시스템의 web application은 퍼스널 컴퓨터의 브라우저 소프트웨어에 종속적이다. 브라우저 소프트웨어는 Chrome, Firefox, Edge, Safari를 지원하며, 각 SW의 최신버전이 권장된다.

본 시스템의 server는 ubuntu 18.04 이상의 operation system 위에서 동작한다. Server는 python과 node.js에 종속적이다. server와 python, node.js 간의 interface는 operating system에 환경변수를 지정하는 것으로 정의된다.

### 2.1.5. Communication Interfaces

시스템 내부에서 web application과 server는 전송계층에서 TCP, 응용계층에서 http을 이용하는 웹 통신 방법을 사용한다. 이 때, http 통신은 REST API 디자인 가이드를 준수해야 한다.

시스템 외부의 다른 시스템과 네트워크 통신은 특별히 요구되지 않는다.

### 2.1.6. Memory

Web application 과 server 모두 4GB 이상의 RAM 이 요구된다.

### 2.1.7. Operations

Normal operation 에서 유저는 web application 에 접속하여 정상적으로 컴파일이 되는 또는 에러를 유발하는 소스코드를 제출한다. 소스코드의 컴파일과 실행이 정상적이면 탄소배출량을 표시하고 그렇지 않으면 에러메시지를 표시한다.

이외의 special operation 은 고려하지 않는다.

### 2.1.8. Site Adaptation Requirements

본 시스템은 배포되는 국가에 따라 다른 탄소배출량을 가진다. 따라서 배포 지역에 맞춘 server 의 PUE (Power usage effectiveness) 값 configure 가 요구된다. 국가에 따른 PUE 값은 다음을 참고할 수 있다.([https://github.com/GreenAlgorithms/green-algorithms-tool/blob/master/data/v2.2/CI\\_aggregated.csv](https://github.com/GreenAlgorithms/green-algorithms-tool/blob/master/data/v2.2/CI_aggregated.csv))

## 2.2. Product functions

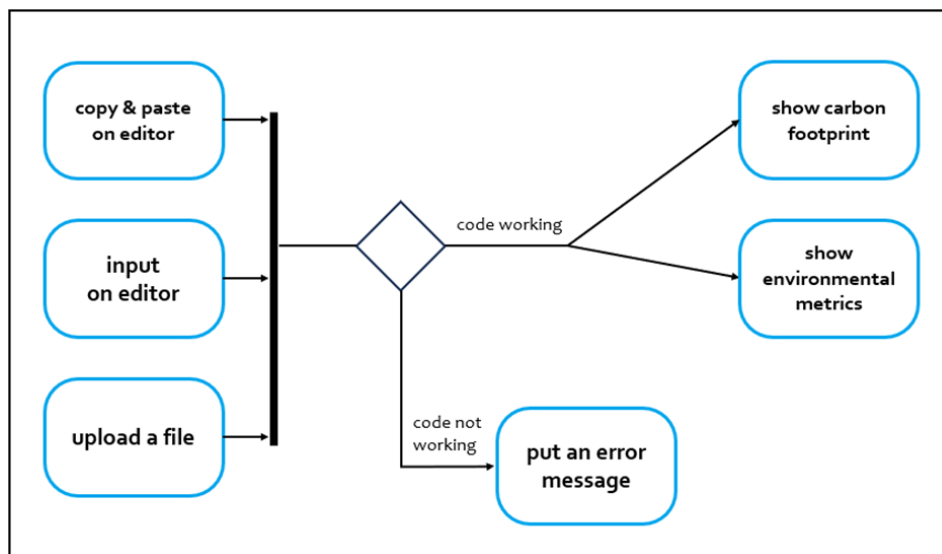


Figure 2.1: Application function flow

### 2.2.1. 로그인

시스템에 유저의 정보를 기록하고 유저가 요청했던 코드와 그에 따른 탄소배출량을 저장하기 위해 필요한 기능이다. 회원가입을 통해 유저가 아이디와 비밀번호를 등록하고, 로그인 버튼을 통해 유저가 로그인 할 수 있어야 한다. 또한, 로그인한 유저의 요청 코드와 탄소배출량 결과는 데이터베이스에 기록되어야 한다.

### 2.2.2. 코드 입력 방법 제공

유저가 탄소배출량을 확인할 소스코드를 다양한 방법으로 입력할 수 있어야 한다. 다양한 방법은 다음과 같다.

- 유저는 editor에 탄소배출량을 확인할 소스코드를 작성할 수 있어야 한다.
- 유저는 OS에서 지원하는 클립보드 복사, 붙여 넣기 기능을 이용하여 editor에 붙여 넣기로 소스코드를 입력할 수 있어야 한다.
- 유저는 파일의 형태로 소스코드를 업로드할 수 있어야 한다.

### 2.2.3. 코드 정상 실행 여부 확인

유저가 제출한 소스코드가 정상적으로 컴파일 되고, 정상적으로 실행가능한지 판별할 수 있어야 한다. 만약 정상적으로 실행된다면, 별도의 메시지없이 2.2.3, 2.2.4 의 기능이 실행된다. 소스코드가 정상적으로 실행되지 않는다면 유저가 에러 내용이 담긴 에러메시지를 확인할 수 있어야한다.

### 2.2.4. 탄소 배출량 제공

유저는 소스코드를 입력하면 입력한 소스코드의 실행 시간에 기반한 탄소배출량을 확인할 수 있어야 한다. 탄소배출량은 kgCO<sub>2</sub>e 의 단위로 주어진다.

### 2.2.5. 탄소 배출량에 상응하는 환경적 척도 제공

유저는 소스코드를 입력하면 탄소배출량과 함께 탄소배출량에 상응하는 환경적 척도를 제공받을 수 있어야 한다. 환경적 척도의 예시로는 주어진 소스코드와 동일한 효과의 차량 매연의 양 등이 있다.

## 2.3. User Classes and Characteristics

### 2.3.1. User Classes

이 제품을 사용하는 것의 필요조건은 프로그래밍을 할 줄 알아야 한다는 것이며, 그린화 패턴의 적용 전후 코드의 차이를 어느 정도 이해할 수 있어야 하므로, 프로그래머, 혹은 프로그래밍 관련 업자가 이 제품의 사용군으로 예상된다.

### 2.3.2. Characteristics

제품의 역할은 제출된 코드의 탄소배출량을 계산하고, 그린화패턴을 통한 탄소배출량 감소를 목표로 하고 있으므로, 위의 user classes 중 환경보호에 관심이 있는 사람들이 이 제품의 주 사용자로 볼 수 있다.

## 2.4. Design and Implementation Constraints

개발 환경에 따른 제한 요소는 다음과 같은 것이 있다.

- 코드를 서버에서 실행하므로 실행 환경의 기준은 서버의 환경으로 제한된다.
- 입력 받는 코드는 java 프로그래밍 언어로 제한된다.
- 코드의 실행 가능 여부 외의 다른 오류는 유저가 직접 다루어야 한다.
- 코드의 라이브러리가 서버와 호환되지 않을 수 있다.
- 그린화 패턴의 적용으로 인한 코드의 최적화를 보장하지 않는다.

즉 그린화 패턴은 탄소배출량의 절감이 목표이므로 그 외의 요소에 대한 최적화 여부는 보장하지 않는다.

## 2.5. Assumptions and Dependencies

본 제품은 서버와의 통신이 필수적이므로 인터넷이 되는 환경에서 사용해야 한다. 만약 이런 환경에서 실행하기 어려운 사용자를 위한 제품을 개발한다면, 현재 이 개발하려는 제품의 수정, 보완 보다는 새로운 설계를 하는 것이 더 나을 것이다.

## 3

## External Interface Requirements

### 3.1. External Interfaces

#### 3.1.1. User Interface

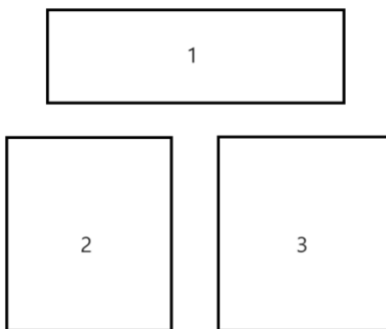
| 이름           | 코드 업로드  |
|--------------|---|
| 목적/내용        | 사용자가 작성한 코드를 코드 입력 칸에 입력 혹은 업로드   |
| 입력 주체/출력 목적지 | 사용자 기반의 컴퓨터 기기  |
| 범위/정확도/허용 오차 | <ul style="list-style-type: none"> <li>● 범위: 키보드에 따른 입력 범위</li> <li>● 정확도: 유저의 키보드 입력에 따른 정확도</li> <li>● 허용 오차: 해당 없음</li> </ul>  |
| 단위           | 마우스 클릭/키보드 입력   |
| 시간/속도        | 사용자의 입력 시간/ 사용자 명령 수행   |
| 타 입출력과 관계    | 입력 내용에 따라 클라이언트에서 처리 혹은 서버로 명령 요청   |
| 화면 형식 및 구성   |  <ol style="list-style-type: none"> <li>1. 코드 입력 창</li> <li>2. 서버 정보</li> <li>3. 탄소 배출량 분석 정보</li> </ol> |
| 윈도우 형식       | 화면 1: Monaco 라이브러리에 따른 코드 에디터   |
| 데이터 형식 및 구성  | Java 언어의 코드 형식의 데이터   |
| 명령 형식        | 해당 없음   |
| 종료 메시지       | 해당 없음   |

Table 3.1: User Interface of uploading code

### 3.1.2. Hardware Interface

| 이름           | 시스템에서 사용 가능한 디바이스             |
|--------------|-------------------------------|
| 목적/내용        | 키보드, 마우스를 사용한 사용자의 입력         |
| 입력 주체/출력 목적지 | 사용자/서버                        |
| 범위/정확도/허용 오차 | 해당 없음                         |
| 단위           | 해당 없음                         |
| 시간/속도        | 사용자의 입력/실행/탄소 배출량 계산에 해당하는 처리 |
| 타 입출력과의 관계   | 해당 없음                         |
| 화면 형식 및 구성   | 해당 없음                         |
| 윈도우 형식       | 해당 없음                         |
| 데이터 형식 및 구성  | 해당 없음                         |
| 명령 형식        | Java 코드                       |
| 종료 메시지       | 해당 없음                         |

Table 3.2: Hardware Interface of applicable device for the system

### 3.1.3. Software Interface

| 이름           | 웹 사이트   |
|--------------|---|
| 목적/내용        | 화면 출력   |
| 입력 주체/출력 목적지 | 사용자/서버  |
| 범위/정확도/허용 오차 | Chrome, Edge, Firefox, Safari 와 같은 웹 브라우저에서 사용 가능 |
| 단위           | 해당 없음   |
| 시간/속도        | 새로 고침에 따른 즉각적인 처리                                 |
| 타 입출력과의 관계   | 해당 없음   |
| 화면 형식 및 구성   | 웹 브라우저를 통한 웹사이트 출력                                |
| 윈도우 형식       | 해당 없음   |
| 데이터 형식 및 구성  | 해당 없음   |
| 명령 형식        | 해당 없음   |
| 종료 메시지       | 해당 없음   |

Table 3.3: Software Interface of applicable device for the system

### 3.1.4. Communication Interface

| 이름           | 호스트 서버 - 클라이언트  |
|--------------|---|
| 목적/내용        | 각 클라이언트에서 호스트 서버에 접속을 요청하고, 사용자가 입력한 코드를 호스트 서버에서 각 클라이언트에게 전달 받고 이에 해당하는 결과를 제공          |
| 입력 주체/출력 목적지 | 클라이언트와 호스트 서버   |
| 범위/정확도/허용 오차 | 해당 없음   |
| 단위           | 패킷  |
| 시간/속도        | 최소 10Mbps 이상  |
| 타 입출력과의 관계   | 해당 없음   |
| 데이터 형식       | <ul style="list-style-type: none"> <li>Struct 를 이용한 명령 코드 (코드 실행, 탄소 배출량 계산 등)</li> </ul> |
| 명령 형식        | Send() 콜에 의한 통신   |
| 종료 메시지       | Close() 콜에 의한 소켓 종료   |

Table 3.4: Communication Interface of applicable device for the system

## 3.2. Function Requirement

### 3.2.1. Use Case

| Use case name  | 탄소 배출량 계산  |
|----------------|--|
| Actor          | 측정하려는 코드를 입력하는 사용자   |
| Description    | 이용자는 코드 입력창에 소스코드를 입력하고, 서버는 이를 서버 환경에서 실행하여 발생하는 탄소 배출량을 미리 입력된 배출량 계산 공식에 따라 계산하여 이용자에게 출력하는 프로세스이다.   |
| Normal Course  | <ol style="list-style-type: none"> <li>초기 화면은 코드 입력이 가능한 코드 입력 칸이 나타난다.</li> <li>모든 사용자는 코드 입력 칸에 java 코드를 입력한다.</li> <li>서버는 입력된 코드를 전달받아 서버 환경에서 실행하여 실행 시간, 사용 메모리 등을 계산한다.</li> <li>산출된 결과값을 이용하여 탄소 배출량 계산 공식에 대입하여 코드가 배출하는 탄소 배출량을 측정한다.</li> </ol> |
| Pre-condition  | <p>사용자는 64KB 이하의 코드나 파일을 입력해야 한다.</p> <p>사용자가 입력한 코드 및 파일은 Java 를 사용해야 한다.</p>   |
| Post-condition | 해당 자바 코드의 탄소 배출량 계산 결과를 사용자에게 출력한다.  |
| Assumptions    | 해당 없음  |

Table 3.5: Use Case of 탄소 배출량 계산



### 3.2.2. Use Case Diagram

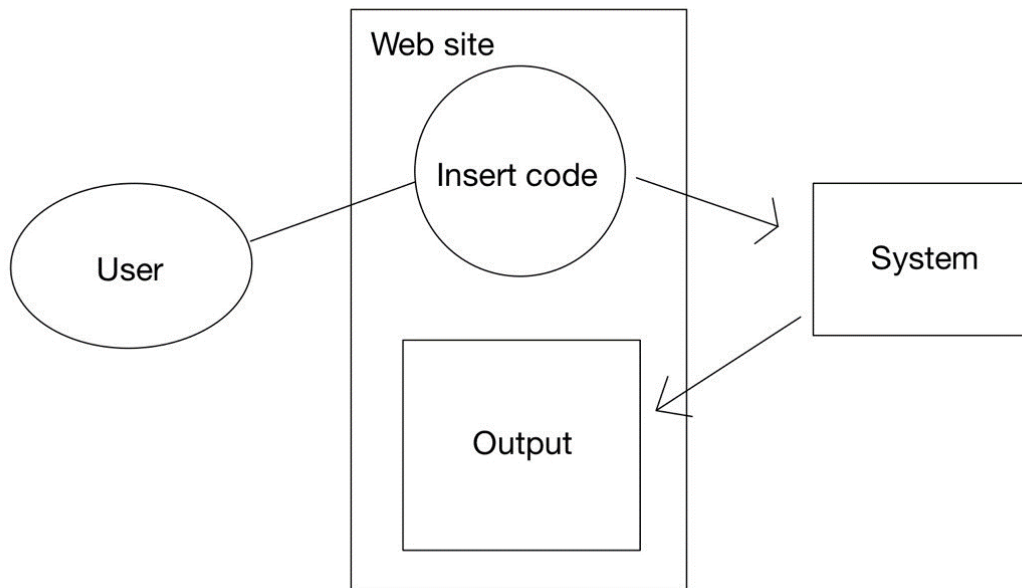


Figure 3.1: Use Case Diagram

### 3.2.3. Data Dictionary

| User  |     |            |             |
|-------|-----|------------|-------------|
| Field | Key | Constraint | Description |
| id    | PK  | Not Null   | User id     |
| name  |     | Not Null   | User name   |

Table 3.6: User

| Carbon  |     |            |                        |
|---------|-----|------------|------------------------|
| Field   | Key | Constraint | Description            |
| id      | PK  | Not Null   | Carbon id              |
| user_id | FK  | Not Null   | User id                |
| carbon  |     | Not Null   | Code's carbon emission |
| code    |     | Not Null   | code                   |

Table 3.7: Carbon

Carbon 테이블의 user\_id 는 FK 로, User 테이블과 의존성을 가진다.

| Sample      |     |            |                      |
|-------------|-----|------------|----------------------|
| Field       | Key | Constraint | Description          |
| id          | PK  | Not Null   | Sample id            |
| name        |     | Not Null   | Sample name          |
| figure      |     | Not Null   | Sample's figure      |
| description |     | Not Null   | Sample's description |

Table 3.8: Sample

### 3.2.4. Data Flow Diagram

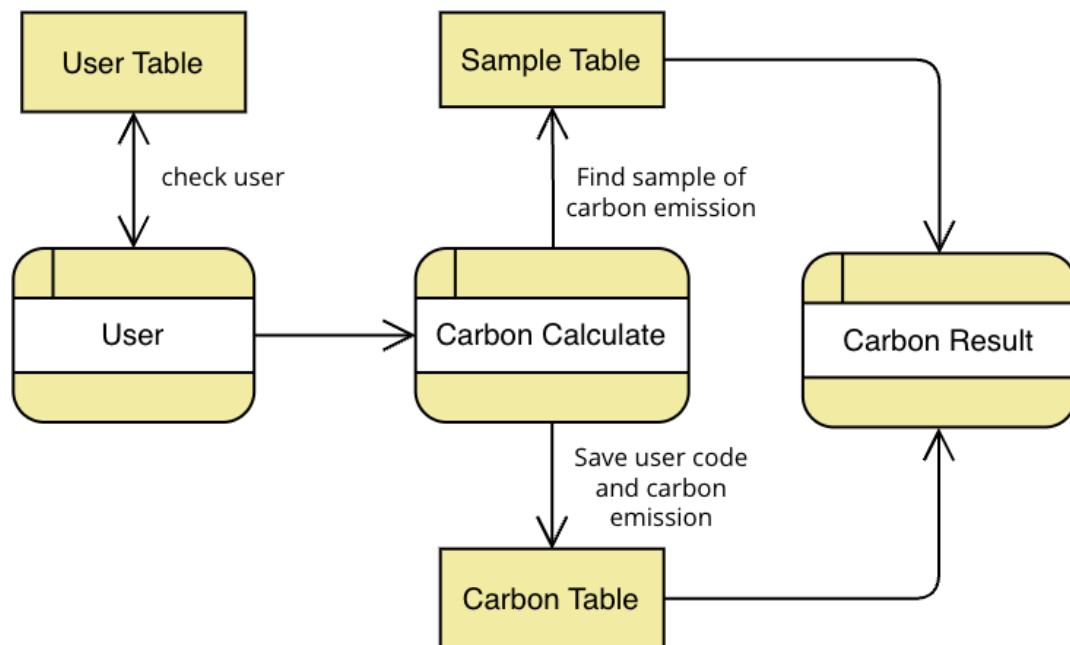


Figure 3.2: Data Flow Diagram

### 3.3. Performance Requirements

#### 3.3.1. Static Numerical Requirement

- 시스템은 Window 7 이상, Linux Kernel 5.0 이상, 또는 macOS 10.0 이상의 운영체제에서 실행되어야 한다. 또한, 시스템 사용을 위해 웹 어플리케이션의 원활한 동작과 최소 10Mbps의 인터넷 속도가 요구된다.
- 시스템은 다수의 사용자가 동시에 코드를 입력하고, 그에 따른 탄소 배출량 계산 결과와 실행 서버 정보를 각각 반환 받을 수 있도록 동작해야 한다. 이때, 사용자는 일반 유저와 관리자로 구분된다.
- 사용자가 입력하는 코드는 Java로 제한되며, GPU를 사용하지 않고 실행 가능해야 한다.

#### 3.3.2. Dynamic Numerical Requirement

- 사용자 코드를 입력하고, 그에 따른 탄소배출량 계산 결과를 확인하는데 걸리는 시간은 3분 내로 완료되어야 한다.
- 탄소 배출량 계산 결과와 코드 실행 후 실행한 환경 정보는 1분 이내로 계산 시스템에 의해 도출되고, 이를 나타낼 수 있어야 한다. 또한, 위 데이터는 10초 이내에 데이터베이스에 저장되어야 한다.
- 최소 300명의 사용자가 동시에 접속하더라도, 시스템은 부드럽고 끊김없이 동작해야 한다.
- 사용자가 코드를 수정 후, 다시 시스템에 입력할 때, 즉시 업데이트된 코드에 따른 탄소 배출량 계산 결과와 실행 서버 정보를 실시간으로 새롭게 반환해야 한다.
- 사용자의 회원가입 요청은 10초 이내로 완료되어야 하며, 새로운 사용자 정보는 10초 이내로 데이터베이스에 저장되어야 한다.
- 사용자의 로그인/로그아웃 요청은 5초 이내로 완료되어야 한다.

### 3.4. Logical Database Requirements

시스템은 MySQL 데이터베이스를 기반으로 동작한다. 해당 데이터베이스를 이용해 시스템 사용자들의 기본 계정 정보를 저장한다. 또한, 각 사용자들의 제출 코드, 그에 따른 탄소 배출량 계산 결과, 실행 서버 정보, 측정 일자 등을 데이터베이스에 저장한다. 사용자가 원할 때, 본인이 작성하였던 코드, 탄소 배출량 계산 결과, 실행 서버 정보, 측정 날짜 등을 조회할 수 있어야 한다.

## 3.5. Design Constraints

### 3.5.1. Physical Design Constraints

- 시스템은 웹 어플리케이션을 기반으로 동작하지만, 사용자가 직접 코드를 입력해야하는 시스템 특성상 모바일 기기보다는 데스크톱 또는 노트북(PC) 기기의 이용이 권장된다.
- 시스템은 MySQL 데이터베이스를 기반으로 필요한 데이터를 해당 데이터베이스에 저장해야 한다.

### 3.5.2. Standards Compliance

- 시스템은 웹 어플리케이션으로 기본적으로 Node.js 를 기반으로 개발된다. 이때, 프론트엔드는 JavaScript 기반의 React.js, 백엔드는 JavaScript 기반의 Express.js 로 개발된다.
- 변수명, 함수명은 Camel Case 를, 데이터베이스는 Snake Case 를 따른다.

## 3.6. Software System Attributes

### 3.6.1. Reality

- 본 시스템은 정확하고 일관된 탄소배출 측정 결과를 제공해야 한다. 따라서 탄소배출량 계산 시, 많은 test 과정을 거쳐 오류를 최소화하고, 신뢰성 있는 데이터가 도출되도록 한다.
- 예외 처리 코드를 작성해 네트워크 장애, 데이터베이스 오류, 하드웨어 문제 등 예상치 못한 에러로부터 회복하고, 시스템이 원활히 동작할 수 있도록 한다.

### 3.6.2. Availability

- 본 시스템은 시스템 중단 시간을 최소화하고 사용자 서비스 중단을 방지해야 한다. 따라서 웹 서버 및 백엔드 서버가 중단되었을 때, 자동으로 재시작 되도록 설계해 사용자에게 지속적이고, 끊김 없는 서비스를 제공하도록 한다.

### 3.6.3. Security

- 일반 사용자에게는 데이터 읽기 권한만 부여하며, 관리자에게는 데이터 수정 및 삭제 권한을 부여해, 모든 사용자가 데이터베이스에 직접적으로 접근하는 것을 금지하고, 데이터베이스의 보안을 유지한다.
- 모든 사용자는 시스템을 이용하기 전, 시스템에 접속하기 위한 계정이 요구된다. 위 계정을 통해 로그인하고, 인증을 받은 사용자만이 시스템에 접속해 서비스를 이용할 수 있다.

### 3.6.4. Maintainability

- 시스템을 관리하기 용이하게 하기 위해서는 모듈화된 디자인을 사용해 개별 구성 요소를 독립적으로 유지하고, 쉽게 확장할 수 있어야 한다. 따라서 프론트엔드의 경우, code editor 등의 UI 와 API 연결 관리를 개별 component 로 관리하며, 백엔드의 경우 router 와 end-point 별로 component 로 관리한다.
- 코드의 가독성을 높이기 위해 변수명 및 함수명, 데이터베이스 표기법을 통일한다. 또한, 코드에 주석을 필수적으로 추가하며, API 문서나 기능정의서 작성 등을 통해 다른 개발자들이 코드와 현 진행사항을 파악해 유지보수가 용이하도록 한다.
- Docker Container 의 사용을 통해 다른 개발자들이 동일한 실행 환경에서 개발할 수 있도록 해, 유지보수가 용이하도록 한다.

### 3.6.5. Portability

- 다양한 운영 체제와 환경에서 프로젝트의 실행을 용이하기 위해 Docker Container 를 사용한다. Docker Container 를 통해 어플리케이션과 그 종속성을 패키지로 하면, 프로젝트의 모든 환경에서 동일한 실행 환경을 보장할 수 있다. 따라서 본 시스템을 다른 호스트 시스템이나 운영 체제로 쉽게 이식하거나 확장시킬 수 있다. 위 특성을 기반으로 서버나 클라우드 인프라가 변동되었을 때, 새로운 환경에서 프로젝트를 배포하거나, 개발 및 테스트 환경을 비교적 빠르게 구축할 수 있다.
- 호스트 의존적인 코드의 비율을 감소시키기 위해 절대 경로 대신, 상대 경로 또는 환경 변수를 사용하여 운영 체제에 독립적인 방식으로 파일에 접근하도록 한다.
- 웹 기반 도구인 React.js 로 프론트엔드를 개발해, 본 시스템이 다양한 웹 브라우저에서 호환성을 유지할 수 있도록 한다.

## 3.7. Organizing the Specific Requirements

### 3.7.1. System Architecture

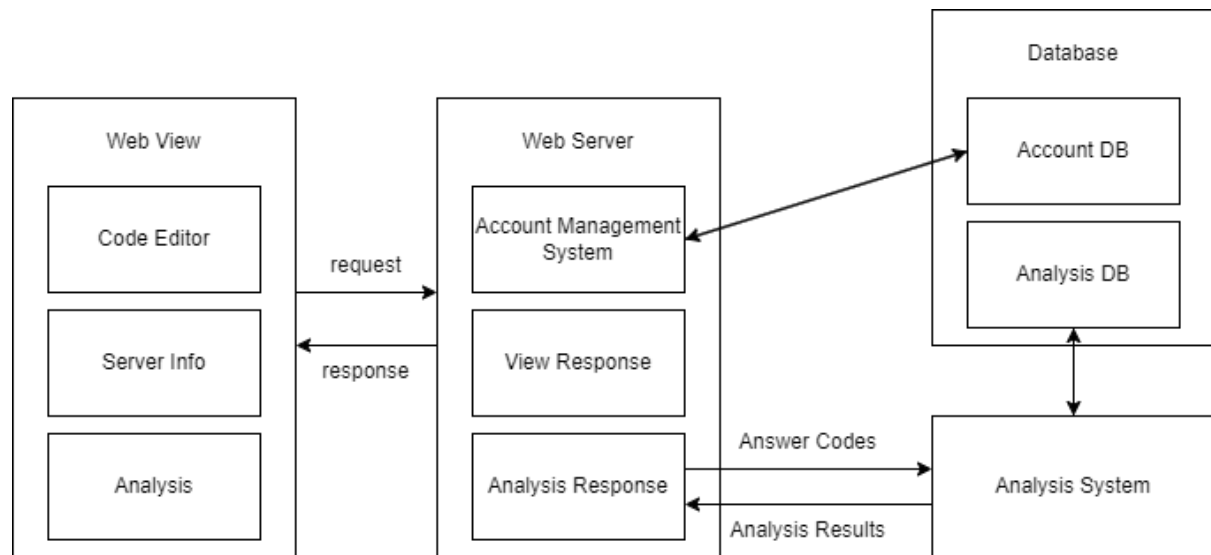


Figure 3.3: System Architecture

### 3.7.2. Process Model

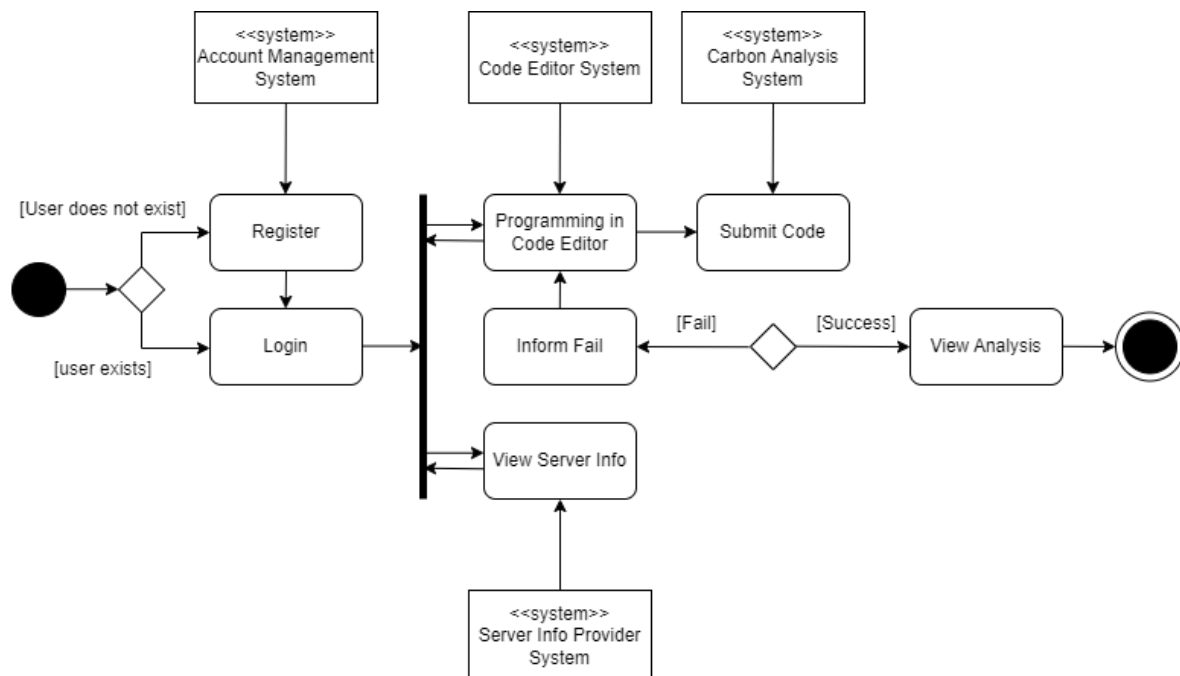


Figure 3.4: Process Model

## 3.8. System Evolution

### 3.8.1. Limitation and Assumption

이번 장에서는 본 시스템의 기본적인 가정과 한계에 대해 서술한다. 즉 소스 코드의 탄소배출량 측정 시스템의 기본 가정과 하드웨어 변화, 사용자 요구 변화 등으로 인한 가능한 모든 변화를 서술한다. 이번 System Evolution 장을 통해 시스템 설계자가 향후 변경에 대한 제약을 피하는 데 도움이 될 수 있다.

#### 3.8.1.1. 고정된 실행 서버 정보

- 본 시스템은 사용자로부터 소스 코드를 입력 받아 탄소배출량을 계산하는 것이 주 목적이다. 이를 위해 특정 서버에서 코드를 실행하며, 이 서버의 정보는 고정되어 있다. 사용자는 코드만 입력하며, 실행 환경에 대한 정보를 입력하지 않는다. 따라서, 서버의 하드웨어 자원이나 운영체제 등의 정보는 고정된 상태에서 코드의 탄소배출량을 계산하게 된다. 이로 인해 특정 서버 환경에서의 탄소배출량만을 측정할 수 있다는 한계가 있다.

#### 3.8.1.2. 지원 가능한 프로그래밍 언어의 부족

- 현재 시스템은 Java 언어로 작성된 소스 코드의 탄소배출량만을 측정할 수 있다. 이는 사용자가 다양한 언어로 작성된 코드의 탄소배출량을 측정하고자 할 경우 제한적일 수 있다. 하지만, 시스템의 초기 목적과 설계 가정을 고려할 때, Java 언어로 제한하는 것은 합리적인 선택이며 추후 사용자의 요구나 시스템의 발전에 따라 다양한 언어를 지원하는 기능을 추가할 수 있다.

#### 3.8.1.3. GPU 사용 제한

- 본 시스템은 탄소배출량을 측정하는데 GPU 를 사용하지 않는다. 이는 시스템이 복잡한 계산이나 대규모 데이터 처리를 요구하는 작업을 수행하지 않기 때문이다. 따라서 GPU 를 사용하지 않고도 실행 가능한 코드만을 측정할 수 있다는 제한이 있다.

#### 3.8.1.4. 악의적인 사용자

- 해당 시스템은 사용자가 제출한 코드를 고정된 실행 서버 환경에 따라 바로 실행한다. 코드의 검증 과정 없이 실행을 진행하므로, 악의적인 사용자를 걸러내는 것은 상당히 어려운 일이며, 이는 시스템의 보안에 위협이 된다.

### 3.8.2. Evolution

#### 3.8.2.1. 다양한 프로그래밍 언어 지원 확대

- 시스템은 처음에는 Java 언어만 지원하도록 설계되었지만, 향후 사용자의 요구와 시스템의 발전에 따라 다양한 프로그래밍 언어를 지원하도록 확장할 수 있다. 이를 통해 사용자는 자신이 편한 언어로 탄소 배출량을 측정할 수 있게 된다.

#### 3.8.2.2. 서버 환경의 다양화

- 현재는 고정된 서버 환경에서 코드를 실행하고 있지만, 시스템의 발전에 따라 다양한 서버 환경을 지원하도록 확장할 수 있다. 이를 통해 사용자는 다양한 하드웨어 자원과 운영체제에서의 탄소배출량을 측정할 수 있게 된다.

#### 3.8.2.3. 악성 코드 필터링 기능 추가

- 시스템은 사용자가 제출한 코드를 바로 실행하도록 설계되었지만, 악의적인 사용자를 필터링하고 보안을 강화하기 위해 악성 코드를 감지하고 필터링하는 기능을 추가할 수 있다.

#### 3.8.2.4. 사용자 코드 분석 및 피드백 제공

- 사용자의 코드를 분석하여 어떤 부분에서 탄소배출량이 많이 발생하는지, 어떤 부분을 개선하면 탄소배출량을 줄일 수 있는지에 대한 피드백을 제공하는 기능을 추가할 수 있다. 이를 통해 사용자는 자신의 코드를 개선하는 데 도움을 받을 수 있다.

#### 3.8.2.5. 기계 학습 모델의 도입

- 탄소배출량을 예측하는 데 더 정확한 결과를 제공하기 위해 기계 학습 모델을 도입할 수 있다. 또한, 사용자의 코드 스타일과 선호 언어 등을 학습하여 개인화된 탄소배출량 측정 결과를 제공할 수 있다.

#### 3.8.2.6. 사용자 제출 코드 데이터 활용

- 사용자가 제출한 코드를 데이터로 활용하여 시스템을 지속적으로 개선하고, 더 다양하고 정확한 탄소배출량 측정 기능을 제공하는데 활용할 수 있다. 이를 통해 시스템은 사용자의 요구에 더욱 잘 맞는 서비스를 제공하게 된다.



## 4

## Supporting Information

### 4.1. Software Requirements Specification

소프트웨어 요구사항 명세서 IEEE 권장 사항 (IEEE Recommend Practice for Software Requirements Specifications, IEEE-Std-830)에 따라 작성되었다.

### 4.2. Document History

| Date       | Version | Description  | Writer |
|------------|---------|--|--------|
| 2023-10-29 | V1.00   | Introduction, Formatting   | 김선우    |
| 2023-10-29 | V1.00   | User Classes and Characteristics, Design and Implementation Constraints, Assumptions and Dependencies, External Interfaces | 설현원    |
| 2023-10-29 | V1.00   | Product Perspective, Product Functions   | 이찬구    |
| 2023-10-29 | V1.00   | Function Requirement   | 이현민    |
| 2023-10-29 | V1.00   | Organizing the Specific Requirements, System Evolution   | 조재범    |
| 2023-10-29 | V1.00   | Performance Requirements, Logical Database, Design Constraints, Software System Attributes                                 | 한수민    |