[CODEMETER]

Software Requirements Specification

by

김동한, 안낙균, 유지훈,
임소리, 조민호, 조유지
of
TEAM 4

Instructor: 이은석

Teaching Assistant: 김소현, 김영경, 김진영, 최동욱, 허진석

Document Date: 29 October, 2023

Faculty: SungKyunKwan University

1. Introduction

1.1. Purpose

CODEMETER는 Java 코드 탄소 배출량 측정 시스템이다. 이 문서는 해당 프로그램에 대한 requirements를 상세하게 정리하여, 추후 시스템을 명확하게 설계하고 개발 과정을 빠르게 진행하고자 하는 데에 목적을 두고 있다.

1.2. Scope

CODEMETER는 사용자로부터 Java 코드를 입력받아 이를 컴파일 및 실행하고, 이 과정에서 컴파일된 프로그램의 실행 시간과 메모리 사용량을 측정한 뒤, 측정값을 바탕으로 서버의 컴퓨팅 환경에서 코드가 탄소를 얼마나 배출하는지를 계산하여 사용자에게 출력하는 웹 서비스이다. 안정적인 서비스가 요구되는 경우 실행 시간 제한, 리소스 격리 작업까지 수행한다.

1.3. Definitions, Acronyms and Abbreviations

Acronyms & Abbreviation	Definition
탄소 배출량(탄소 발자국)	개인 또는 단체가 발생시키는 온실가스의 총량으로, 본 프로젝트에서는 특정 코드를 실행할 때 컴퓨터가 연산 처리 과정에서 발생시키는 전력소비량을 기준으로 하여 온실가스 배출량을 측정한다.
전력소비량(소비 전력)	전자 기기의 정상적인 운영을 위해 공급되어야 하는, 시간에 따른 전기 에너지양을 의미한다. 특정 코드의 탄소 배출량 측정 시에는 실행 시간과 CPU 사용량, 메모리 사용량 등을 통해 계산한다.

1.4. References

- Lannelongue, L., Grealey, J. & Inouye, M., ‘Green Algorithms: Quantifying the Carbon Footprint of Computation’, Advanced Science, 8(12), 2021. https://doi.org/10.1002/advs.202100707
- "IEEE Recommended Practice for Software Requirements Specifications," in IEEE Std 830-1998 , pp.1-40, 1998. https://ieeexplore.ieee.org/document/720574

1.5. Overview

본 요구사항 명세서는 전반적인 시스템 설명과 요구사항 세부 설명으로 구성된다. 먼저 전반적인 시스템 설명에서는 Java 코드 탄소 배출량 측정 시스템 CODEMETER를 개발하는 동기와 목적, 기대 효과를 설명하고, 시스템에서 제공하는 주요 기능에 대해 요약한다. 이와 더불어 해당 시스템을 사용할 사용자를 정의하고, 개발 과정에서의 제한사항 및 요구사항에서 가정하고 있는 정보를 정의한다. 요구사항 세부 설명에서는 시스템의 기능적 요구사항, 성능 요구사항에 대한 정의를 자세하게 서술하며, UML을 통해 시스템 구조를 시각적으로 쉽게 이해할 수 있는 여러 가지 다이어그램을 제공한다.

2. Overall Description

2.1. Product Perspective

프로그래밍 코드의 탄소 배출량 측정 도구인 CODEMETER는 환경 보호와 소프트웨어가 발생시키는 탄소 배출량을 최소화할 책임을 가지는 개발자가 코드의 탄소 배출량을 쉽게 측정할 수 있도록 설계되었다. 이때 입력받는 코드는 Java 코드로 한정되며, 해당 코드가 실행될 때 발생하는 탄소 배출량을 측정함으로써 개발자가 소프트웨어의 탄소 배출량을 실감할 수 있도록 도와주어 환경 친화적인 개발을 장려한다. 또한 이렇게 측정된 탄소 배출량을 전력 소모량과 함께 일상에서 친숙하게 접할 수 있는 값(예: A4용지 사용량 등)으로 환산하여 표현함으로써 해당 코드가 발생시키는 탄소 배출량에 대해 단순 계산 수치값보다 비교적 더 직관적으로 체감할 수 있게 한다.

2.1.1. System Interfaces

사용자가 탄소 배출량을 측정하고자 하는 Java 코드를 웹 어플리케이션 입력하면 서버 시스템은 이를 전송받고 코드를 컴파일 및 실행한다. 컴파일 및 실행 중 오류가 발생하면 사용자에게 오류 내용을 전달하고, 오류가 없다면 실행시간과 CPU, 메모리 사용량 등을 측정 후 정상적으로 탄소 배출량을 계산한다.

2.1.2. User Interfaces

유저 인터페이스는 웹 어플리케이션을 통해 제공된다. 코드를 입력할 수 있는 section을 제공하고, 하단에는 입력한 코드를 실행하는 RUN 버튼과 입력 내용을 초기화하는 새로고침 버튼이 있다. 사용자가 맨 처음 컴파일을 하기 전에는 코드 입력 section 하단에 CODEMETER가 어떻게 작동하는지에 대한 사용 설명서와 그 의의가 간략하게 표시된다. 사용자가 컴파일을 성공하게 될 경우, 해당 설명 section은 코드의 탄소 배출량을 계산하기 위한 실행 서버 스펙에 대한 정보 및 측정한 탄소 배출량과 관련 정보들을 표시하는 section으로 교체된다. 이때 관련 정보는 전력 소모량을 포함하여 TV 시청 시간, 자동차 주행 거리, 엘리베이터 이동 층수, A4용지 사용량과 같이 실생활에서 사용되는 탄소 배출량으로 환산된 값을 일컫는다. 만약 코드 컴파일에 실패하거나 런타임 에러가 발생하는 경우, 해당되는 에러 메시지를 알리는 토스트가 화면에 표시된다.

2.1.3. Hardware Interfaces

웹으로 제공되는 서비스이므로 웹 브라우저를 통해 접근 가능하다. 어떤 기기를 사용해도 무방하나 노트북 혹은 데스크탑의 16:9 화면 비율을 권장한다.

2.1.4. Software Interfaces

Window 10 이상, Google Chrome 버전 96 이상을 타겟으로 하고 있다. 보다 낮은 버전으로 실행하거나 V8 Engine이 아닌 브라우저를 사용할 시 정상 동작하지 않을 수 있다.

2.2. Product Functions

이하는 CODEMETER가 수행할 주요 기능에 대한 요약이다.

2.2.1. Java 코드 입력 및 설정

탄소 배출량을 측정할 Java 코드를 사용자로부터 입력받는 기능이다. 사용자는 코드 입력란을 통해 탄소 배출량을 측정할 코드를 입력할 수 있으며, RUN 버튼을 클릭하면 입력한 코드를 서버에 전송할 수 있다. 이후 서버는 사용자에게 입력 받은 코드를 컴파일하고 실행하며, 이에 대한 성공 여부를 파악하고 실패 시 사용자에게 알려야 한다.

2.2.2. 탄소 배출량 측정

사용자에게서 입력받은 코드를 컴파일 및 실행하고, 배포된 서버 환경을 바탕으로 코드의 실행시간, 메모리 사용량 등의 정보를 통해 탄소 배출량을 측정한다. 측정한 탄소 배출량과 전력 소모량을 사용자에게 보여 주고, 측정한 탄소 배출량을 실생활에서의 여러 가지 탄소 배출량으로 환산한 값 또한 함께 제공한다.

2.3. User classes and Characteristics

2.3.1. 개발자 및 기업

탄소 배출량 측정 도구의 사용자는 환경 보호 및 탄소 배출량 감소에 관심을 갖는 기업 및 개발자이다. 작성한 코드의 탄소 배출량과 감소 방법에 노력을 기울이기 위해 탄소 배출량 측정 도구를 사용할 것으로 예상된다.

2.3.2. 환경 단체

환경 보호를 위해 노력하는 환경 단체는 탄소 배출량 측정 도구의 또다른 사용자이다. 탄소 배출량 측정 도구를 사용하여 일반적인 코드 패턴에서 탄소 배출량을 줄이기 위한 패턴을 찾기 위해 측정 도구를 사용할 것이며, 도출한 패턴을 기업들에게 가이드 할 수 있다.

2.4. Design and Implementation Constraints

2.4.1. 안전 및 보안사항

사용자가 입력한 Java 코드가 정상적으로 실행된다고 보장할 수 없다. 또한 실행 서버를 공격할 수 있는 코드가 포함될 수 있다. 따라서 실행시간을 제한하고 리소스를 격리하여 악의적인 코드를 방어한다.

2.4.2. 서버 제한 사항

Linux가 아닌 다른 운영체제를 사용하여 프로세스를 격리시키는 것은 난이도가 높다. 따라서 탄소 배출량 계산 서버는 Linux 운영체제 위에서 실행되어야 한다.

2.4.3. 사용자 제약 사항

Google Chrome 버전 96 이상을 타겟으로 개발하기 때문에, Chrome 기반 브라우저를 기준으로 한다.

2.5. Assumptions and Dependencies

2.5.1. 보안

높은 수준의 보안사항은 요구사항에 영향을 줄 수 있다. 실행 서버의 높은 보안 수준과 안정성이 필요하게 되면, 사용자가 입력한 Java 코드를 분석하여 위험한 코드를 감지하는 기능을 추가해야 한다. 또한 실행 서버를 안정적으로 관리하기 위해서는 서버 이중화 구성이 필요하다.

2.5.2. 탄소 배출량 계산 과정에서의 하드웨어 스펙

본 시스템에서 사용하는 탄소 배출량 계산식에는 코드를 실행하는 하드웨어 스펙에 따른 특정 상수값을 포함하고 있다. 또한 코드를 실행하는 환경에 따라 실행 시간이 다르게 측정될 수 있으며, 이는 코드 탄소 배출량 계산에 큰 영향을 미친다. 따라서 사용자에게서 실행 환경에 대한 하드웨어 스펙 정보를 제공받지 않고 실행 서버의 하드웨어 스펙으로 고정해서 계산하며, 이러한 고정된 환경에서 실행한 코드의 실행 시간을 통해 탄소 배출량을 계산한다.

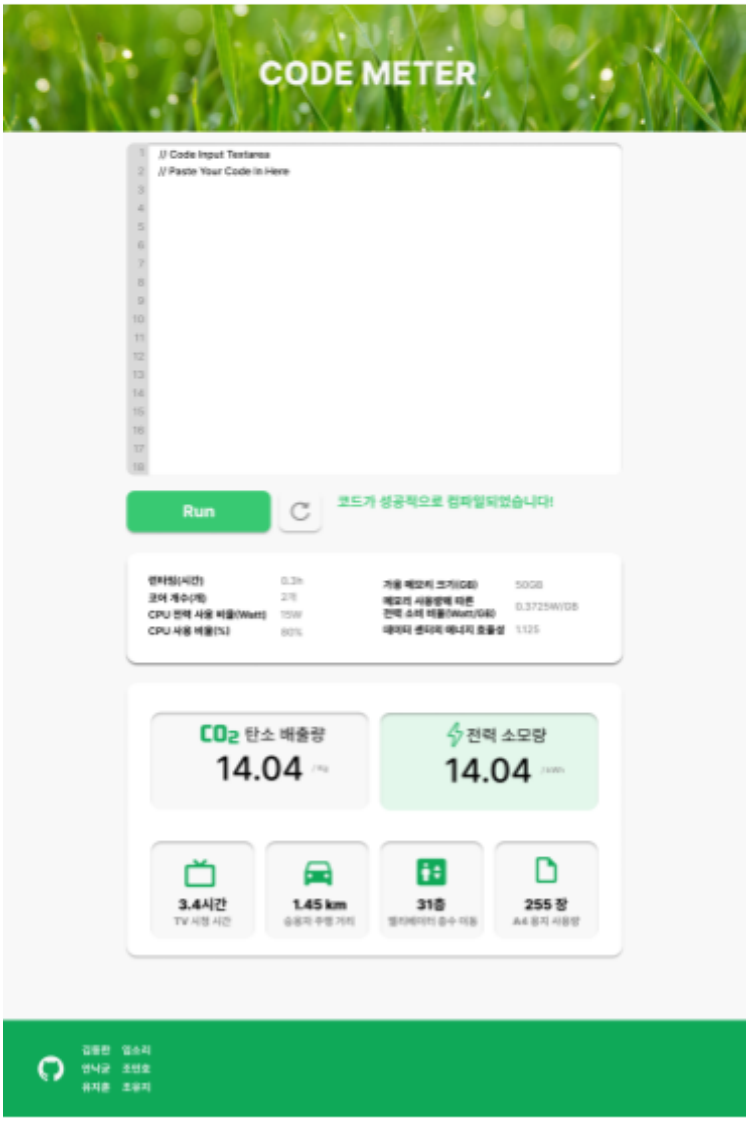
3. External Interface Requirements

3.1. External Interfaces

3.1.1. user interface

이름	마우스 및 키보드를 통한 입력 처리
목적/내용	사용자가 키보드, 마우스의 입력을 통해 시스템에 명령 전달
입력 주체/출력 목적지	사용자/Linux 기반의 웹 서버
범위/정확도	- 범위: 화면에서의 버튼의 개수에 따른 입력 범위 - 정확도: 사용자의 키보드, 마우스 입력의 정확도
단위	버튼 클릭/키보드 입력
시간/속도	비정기적인 사용자의 입력/즉각적인 사용자 명령 수행
타 입출력과의 관계	서버로 명령 요청 및 응답 사용
데이터 형식 및 구성	main 함수가 있는 Java 프로그래밍 코드
명령 형식	Java 프로그래밍 코드 입력
이름	모니터를 통한 메인 화면 출력
목적/내용	시스템 사용자에게 제공하는 인터페이스
입력 주체/출력 목적지	클라이언트/사용자 디스플레이 디바이스
범위/정확도	- 범위: 화면에서의 버튼에 따른 범위 - 정확도: 사용자의 키보드, 마우스의 입력 정확도
단위	화면
시간/속도	사용자의 입력에 따른 화면 전환

타 입출력과의 관계	사용자의 입력을 위한 인터페이스 출력 후 사용자의 입력 대기

화면 형식 및 구성	 그림 3.1.1.1 메인섹션 구성도  그림 3.1.1.2 메인섹션 프로토타입

	[그림 3.1.1.1 및 3.1.1.2 - 메인 섹션] - 메인 섹션은 Java 코드 입력 창, 실행 서버 정보, 가이드라인으로 구성 - 사용자는 Java 코드 입력 창에서 클래스 단위로 Java 프로그래밍 코드를 작성 - 실행 서버 정보는 사용자의 코드를 실행시키는 실제 배포된 서버의 정보를 출력 - 가이드라인은 본 시스템을 사용하는 방법 및 계산 식에 대한 내용을 출력

그림 3.1.1.3 결과 섹션 구성도

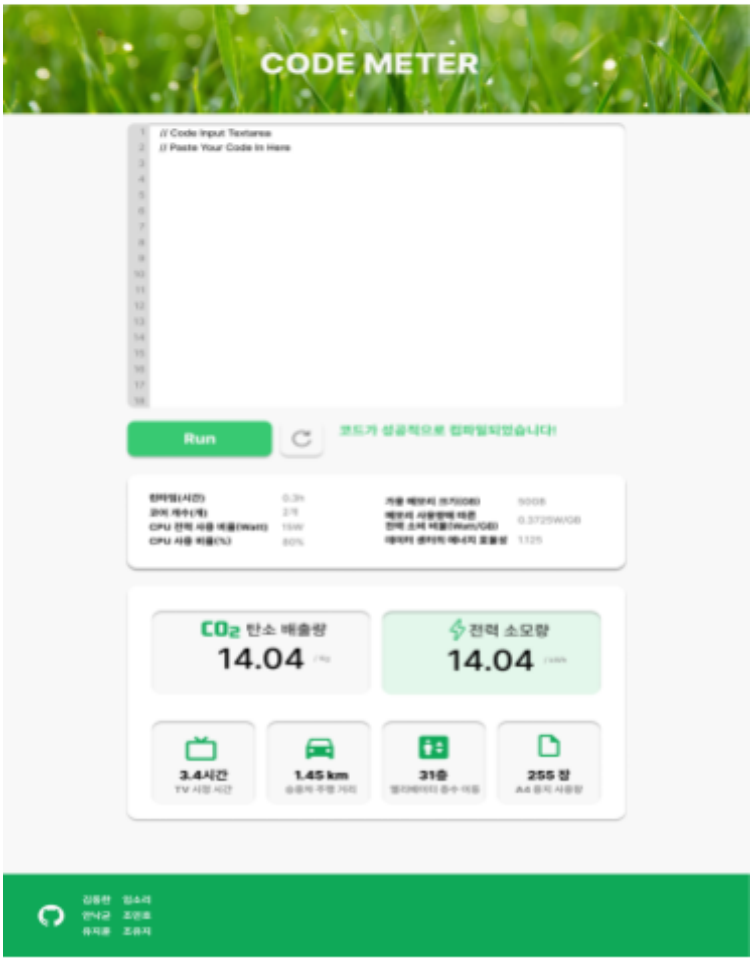


그림 3.1.1.4 결과 섹션 프로토타입

	[그림 3.1.1.3 및 3.1.1.4 - 결과 섹션] - 결과 섹션은 탄소 배출량 정보, 기준치 대비 배출량 정보로 구성 - 탄소 배출량 정보는 절대적인 탄소 배출량 수치를 출력 - 기준치 대비 배출량 정보는 각종 기준치 (자가용 등) 대비 탄소 배출량 수치를 출력
윈도우 형식 및 구성	- Linear Layout 형식으로 메인 페이지, 결과 페이지를 구성 - Java code를 작성하는 IDE 창 - 탄소 배출량 측정 결과창
데이터 형식 및 구성	이미지, 텍스트, 프로그래밍 코드
이름	Java 코드 입력 및 탄소 배출량 출력 인터페이스
목적/내용	Java 코드의 입력 및 탄소 배출량 출력
입력 주체/출력 목적지	사용자/서버
단위	화면
화면 형식 및 구성	- Java 코드 입력을 위한 IDE - 코드 제출을 위한 버튼
데이터 형식 및 구성	Java code
명령 형식	제출 버튼을 통한 명령
종료 메시지	[정상적인 입력을 한 경우] 해당 없음 [코드를 입력하지 않고 제출한 경우] “Java 코드를 입력해주세요.” [컴파일 에러가 발생한 경우] “컴파일 에러가 발생했습니다. 다시 시도해주세요.” [런타임 에러가 발생한 경우] “런타임 에러가 발생했습니다. 다시 시도해주세요.”

3.1.2. hardware interface

이름	IO 디바이스
목적/내용	키보드, 마우스를 사용한 사용자의 입력
입력 주체/출력 목적지	사용자/서버
시간/속도	사용자의 입력
타 입출력과의 관계	입력된 코드를 웹 서버로 전송
데이터 형식 및 구성	Java code
명령 형식	HTTP request
이름	웹 서버
목적/내용	웹 서버를 이용한 Java 코드 컴파일 및 탄소 배출량 측정
입력 주체/출력 목적지	IO 디바이스/서버
시간/속도	Java 컴파일 시간/Java 코드 실행 시간/탄소 배출량 측정 시간
타 입출력과의 관계	클라이언트에게서 Java 코드를 전달받고, 클라이언트에게 탄소 배출량 측정 결과를 전송
명령 형식	HTTP response

3.1.3. software interface

이름	웹 사이트
목적/내용	서비스 화면 출력

범위/정확도/허용 오차	웹 브라우저에서 사용 가능
시간/속도	웹 서버와의 HTTP 통신 시간
타 입출력과의 관계	웹 서버로 Java 코드 전송
화면 형식 및 구성	웹 브라우저를 통한 서비스 화면 출력

3.1.4. communication interface

이름	프론트엔드 - 백엔드
목적/내용	클라이언트(프론트엔드)에서 사용자가 입력한 Java 코드를 서버(백엔드)로 전송하고, 서버는 해당 코드에 대한 컴파일 결과 및 탄소 배출량 측정 결과를 클라이언트에게 제공
입력 주체/출력 목적지	프론트엔드/백엔드
단위	HTTP request/HTTP response
시간/속도	클라이언트/서버와의 HTTP 통신 시간
데이터 형식 및 구성	- Java 프로그래밍 코드가 포함된 JSON 데이터 - 예시) `{ "javaProgrammingCode": "public class Main { public static void main(String[] args) { System.out.println(\"Hello World!\"); } }" }`

3.2. Function Requirements

3.2.1. Use case

Use case name	Input Java code
Actor	서비스 사용자 (Java 프로그래머)
Description	사용자가 Java 코드의 탄소 배출량 측정을 위해 Java 코드를

	입력하고, 서버에서 전달받은 탄소 배출량 측정 결과를 확인하는 과정이다.
Normal Course	1. 웹사이트 접속 후 상단에 Java 코드 입력 창이 있는 메인 페이지가 나타난다. 2. 프로그래머가 작성한 Java 코드를 입력한다. 3. 사용자는 제출 버튼을 누른다. 4. 시스템은 입력된 Java 코드에 대해 다음과 같이 동작한다. a. 실행된 코드에 대한 탄소 배출량을 계산한다. b. 계산된 탄소 배출량을 반환한다. 5. 사용자는 결과로 반환된 탄소 배출량이 출력되는 결과 페이지로 이동한다. - 탄소 배출량은 gCO2e 단위로 출력한다. - 측정된 탄소 배출량을 절감하기 위해 한 달 간 심어야 하는 나무의 양, 측정된 탄소 배출량을 자동차가 배출하기까지 걸리는 거리, 측정된 탄소 배출량을 소모하여 비행기가 날아갈 수 있는 거리, 측정된 탄소 배출량을 에너지로 환산한 값을 각각 출력한다.
Precondition	- 사용자가 작성한 Java 코드에는 public static void main 함수가 정의되어 있어야 한다. - 사용자가 작성한 Java 코드는 클래스 형식으로 작성되어 있어야 한다. - 사용자는 코드를 입력한 후 제출 버튼을 눌러야 한다. - 코드를 입력하지 않고 제출 버튼을 누른 경우 코드 입력을 요구하는 오류 메시지를 출력한다.
Post Condition	- 사용자는 컴파일 에러가 발생하지 않는 코드를 제출해야 한다. - 컴파일 에러가 발생할 경우 컴파일 에러 메시지를 출력한다. - 사용자는 런타임 에러가 발생하지 않는 코드를

	제출해야 한다. - 런타임 에러가 발생할 경우 런타임 에러 메시지를 출력한다.
Assumptions	- 탄소 배출량 계산 시 서버의 HW 사양은 사용자의 컴퓨터가 아닌, 서버의 컴퓨터 기준으로 계산한다.

3.2.2. Use Case Diagram

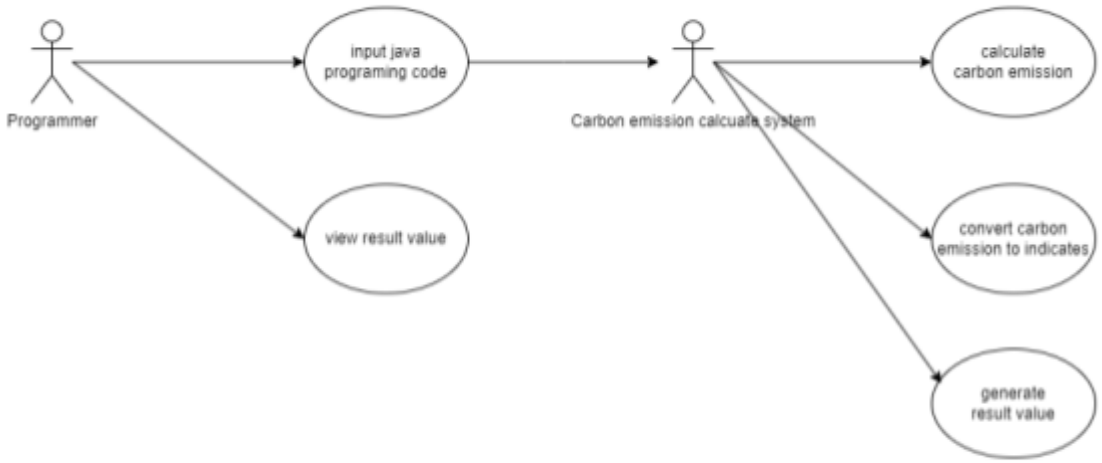


그림 3.2.2.1 Use Case Diagram

3.2.3. Data Dictionary

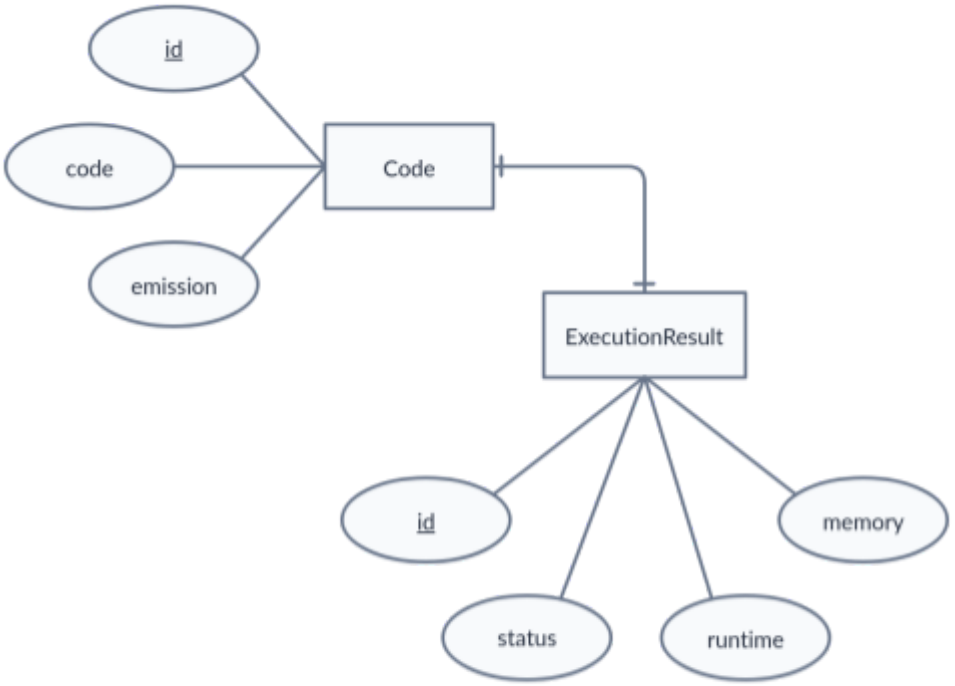


그림 3.2.3.1 Data Dictionary

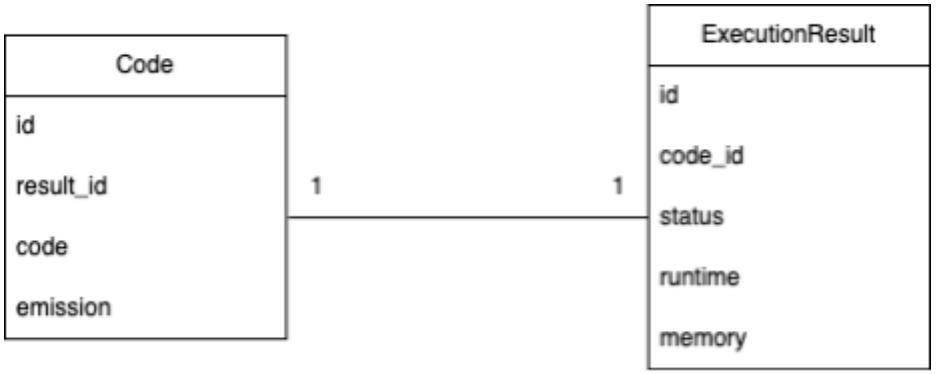


그림 3.2.3.2 E-R Diagram

Code				
Field	Key	Domain	Constraint	Description
id	PK	int	Not Null	Entity id
code		text	Not Null	Java code
result_id	FK	int		execution result

emission		float		estimated amount of carbon emission

ExecutionResult				
Field	Key	Domain	Constraint	Description
id	PK	int	Not Null	Entity id
code_id	FK	int	Not Null	
status		varchar(255)	Not Null	(“SUCCESS”, “COMPILE_ERROR”, “RUNTIME_ERROR”, “TIME_EXCEEDED”, “MEMORY_EXCEEDED”)
runtime		BigInt	Not Null	cpu time for execution (ms)
memory		BigInt	Not Null	memory usage (Byte)

3.2.4. Data flow Diagram

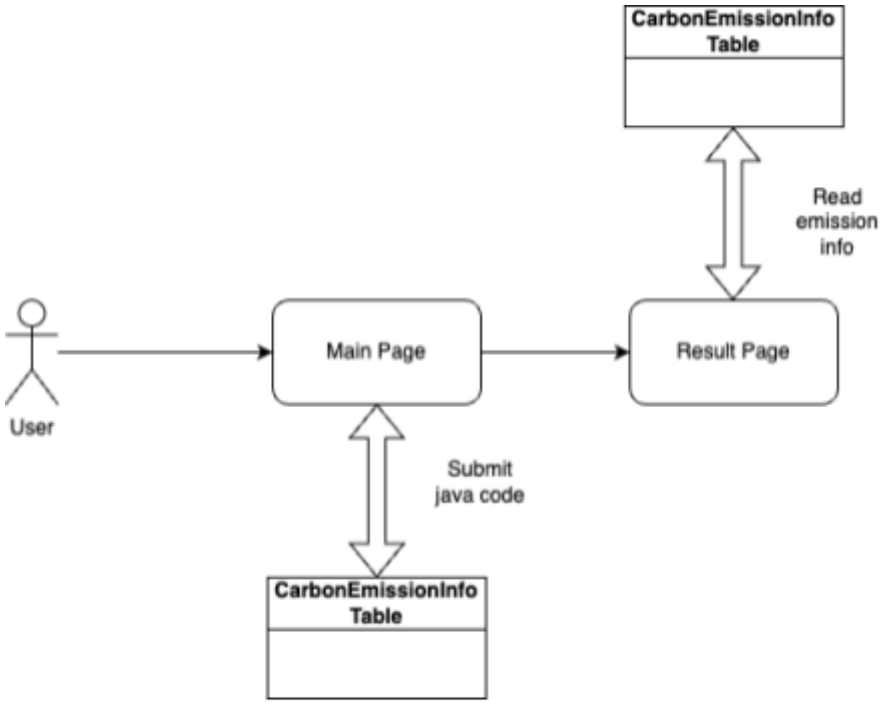


그림 3.2.4.1 Data Diagram

3.3. Performance Requirements

본 시스템이 만족해야 하는 시스템 요구사항을 정의한다. 이는 예측 기반의 최소 요구사항이며, 실제 요구사항은 이와 달라질 수 있다.

3.3.1. Static Numerical Requirements

- 시스템은 동시에 여러 명의 사용자가 Java 코드를 입력하고 탄소 배출량을 조회할 수 있어야 한다.
- 시스템 최소 요구사항은 다음과 같다.
    - 2세대 Core i5(2GHz 이상), 3세대/4세대 Core i5 또는 이에 상응하는 프로세서
        - Windows 10 이상
        - Intel HD4000 또는 이에 상응하는 그래픽
- 네트워크 최소 대역폭은 다음과 같다.

- 다운로드/업로드 속도 최소 5mbps 이상

3.3.2. Dynamic Numerical Requirements

- 시스템은 동시에 최소 20명의 사용자 접속을 유지할 수 있어야 한다.
- 시스템은 최대 10000건의 탄소 배출량 정보를 저장할 수 있어야 한다.
- 웹 브라우저가 메인 페이지를 5초 이내에 로드할 수 있어야 한다.
- 사용자가 Java code를 입력 후 5초 이내에 결과 페이지를 로드할 수 있어야 한다.
  - 단, 결과 페이지 로드 시간에 Java code 빌드 및 실행 시간은 포함하지 않는다.

3.4. Logical Database Requirements

- 시스템은 MySQL 데이터베이스를 이용하여 데이터를 관리한다. 해당 데이터베이스를 이용하여 제출된 코드, 해당 코드의 컴파일, 실행 성공 여부, 런타임, 탄소 배출량 정보를 관리한다.
- NestJS를 사용하는 시스템 특성을 고려, 데이터베이스 접근 기술로써 Node.js 기반 TypeORM을 사용하여 데이터를 매핑 및 관리한다.

3.5. Design Constraints

3.5.1. Physical design constraints

- 해당 시스템의 목적은 Java 프로그래밍 코드의 탄소 배출량 측정이다. 따라서 PC 환경을 권장하며, 모바일 환경은 권장되지 않는다.
- 해당 시스템은 MySQL을 사용하여 Java 프로그래밍 코드와 그에 따른 탄소 배출량을 데이터베이스에 저장할 수 있어야 한다.

3.5.2. Standard constraints

- 개발 및 유지보수 과정에서 ESLint 및 Prettier를 적극 활용하여 코드의 일관성과 품질을 유지하며, 컴파일 에러 및 코드 스타일 관련 문제를 사전에 예방한다.

- 협업 과정에서는 Git 커밋 메시지의 일관성을 유지하기 위해 AngularJS Git 커밋 컨벤션을 따른다.
- 자바스크립트 스타일 가이드와 HTML 표준을 따른다.
- 변수 이름과 함수 이름은 camel case를 사용한다.

3.6. Software System Attributes

3.6.1. Reliability

- 서버에서 Java코드를 실행하는 런타임 환경은 안정성과 성능을 고려하여 설계한다.
- 컴파일 에러와 같은 예외적인 상황에 대한 처리 및 복구 메커니즘을 구현하여, 사용자의 코드가 안정적인 환경에서 동작할 수 있도록 한다.

3.6.2. Availability

- 시스템은 Java 코드의 CPU 사용량과 메모리 사용량을 측정하기 위해 동시 사용자를 허용하되, 모든 사용자의 요청은 순차적으로 처리한다.
- 시스템은 각 Java 코드를 실행할 때 메모리 점유율 60%을 넘기지 않아야 한다.
- 시스템은 메모리 점유율 90%를 넘겼을 경우, 더 이상 해당 Java 코드의 실행을 진행하지 않아야 하며, 추가 사용자를 허용하지 않아야 한다.

3.6.3. Security

- 사용자가 입력한 Java 코드에는 서버를 공격하고자 하는 악의적인 코드가 포함될 수 있다. 이로 인한 치명적인 결함이 발생하는 것을 방지하기 위해 사용자 input 코드를 실행시키는 환경을 샌드박싱하여 리소스를 격리하는 것이 필수적이다.
- 또한, 서버의 성능이 저하되는 것을 방지하기 위하여 코드 런타임의 제한 시간 및 최대 메모 사용량도 반드시 설정되어야 한다. 실행된 코드가 제한을 넘어서는 리소스를 사용하면 실행을 강제 중단한다.

3.6.4. Maintainability

- Front-end

- React 라이브러리를 활용하여 UI 요소들을 컴포넌트 단위로 모듈화하여 관리한다.
- 또한 각 컴포넌트의 스타일은 별도의 파일로 분리하여 작업한다. 이를 통해 컴포넌트 간 상호 의존성을 최소화하고, 컴포넌트의 재사용성을 높인다.

- Back-end
  - NestJs 프레임워크를 활용, 객체 지향적 설계 원칙을 준수하여 서버를 구성한다.
  - NestJs의 모듈 시스템을 적극적으로 활용하여 코드를 모듈화하고 의존성을 관리한다.

3.6.5. Portability

- 시스템은 Google Chrome, Firefox, Safari 등 모든 주요 데스크탑 웹 브라우저에서 동작할 수 있어야 한다.
- 시스템은 모든 모바일 환경 웹 브라우저에서 동작할 수 있어야 한다.
- Linux 서버 상에서 구동될 서비스 특성 상, Linux 상에서 빌드 가능한 API를 사용해서 시스템이 동작할 수 있어야 한다.

3.7. Organizing the Specific Requirements

3.7.1. System Mode

본 시스템은 다음과 같은 모드를 통해 동작한다.

- 대기 : 사용자에게서 탄소 배출량을 측정할 Java 코드를 입력받을 때까지 대기한다. 코드를 입력받으면 실행 서버에 전송 후 사용자 코드 컴파일 모드로 들어간다.
- 사용자 코드 컴파일 : 사용자에게서 전달받은 Java 코드를 컴파일한다. 이때 컴파일 에러가 발생하면 해당 에러 문구를 사용자에게 전송하고 대기 모드로 돌아간다. 컴파일에 성공하면 사용자 코드 실행 모드로 이동한다.
- 사용자 코드 실행 : 컴파일에 성공한 사용자 코드를 실행하고, 탄소 배출량 계산에 필요한 실행 시간과 메모리 사용량을 측정한다. 만약 런타임 에러가 발생한다면 해당 에러 문구를 사용자에게 전송하고 대기 모드로 돌아간다. 실행에 성공하면 탄소 배출량 측정 및 변환 모드에 돌입한다.

- 탄소 배출량 측정 및 변환 : 사용자 코드의 실행 시간과 메모리 사용량을 바탕으로 탄소 배출량을 측정하며, 이를 일상생활에서의 여러 가지 탄소 배출 예시로 변환한다. 최종 결과물이 생성되면 사용자에게 해당 결과를 전송하고 대기 모드로 돌아간다.

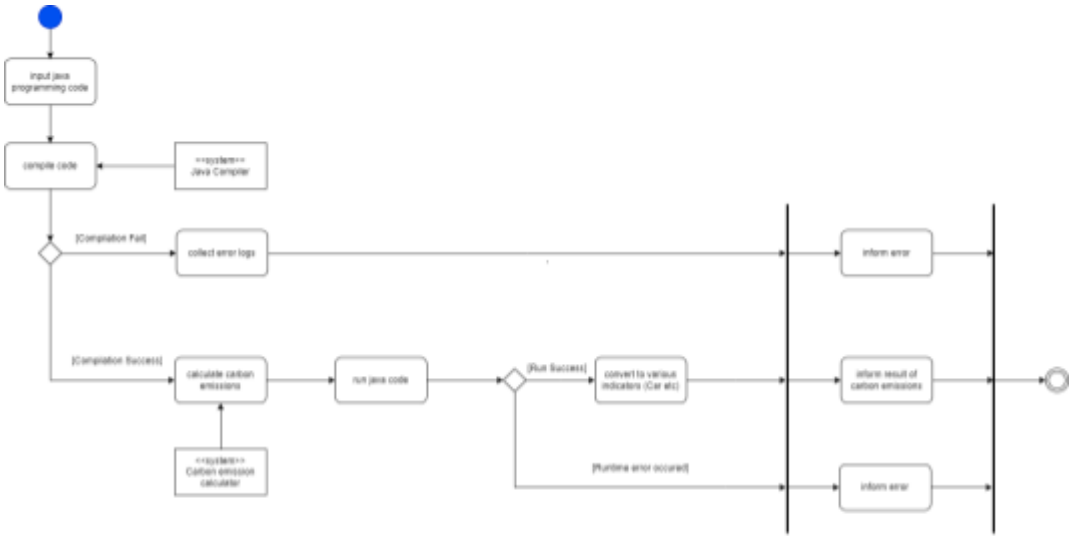


그림 3.7.1.1 Process Diagram

3.7.2. User Class

본 시스템의 사용자는 코드의 탄소 배출량을 안정적인 환경에서 계산해보고자 하는 Java 개발자로 가정한다.

3.7.3. Feature

사용자로부터 입력받은 Java 소스코드를 실행하였을 때 발생하는 탄소 배출량을 측정하여 화면에 출력한다. 또한 일상생활에서의 여러 가지 탄소 배출 예시와 비교해 사용자 코드의 탄소 배출량에 대한 이해를 돕는다.

3.7.4. Stimulus

본 시스템은 사용자의 데이터 입력을 유일한 자극(stimulus)으로 하며, 소스 코드 데이터를 입력받았을 때 이를 컴파일 및 실행하고 코드의 탄소 배출량을 측정하는 것이 주 작업이다.

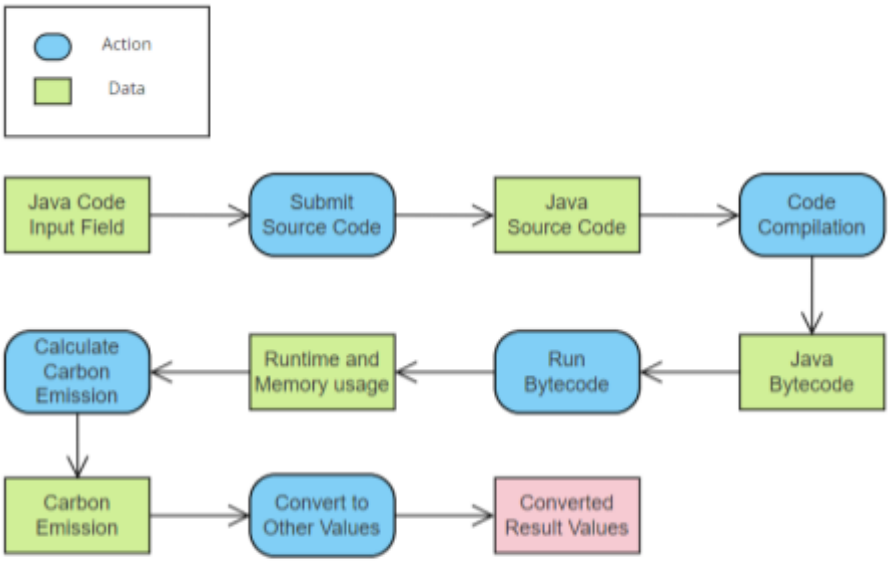


그림 3.7.5.1 Activity model

3.7.5. Response

컴파일된 프로그램의 실행 결과를 바탕으로 탄소 배출량을 계산하고, 실행 결과 및 탄소 배출량 연산 결과를 응답으로 반환한다.

3.7.6. Functional Hierarchy

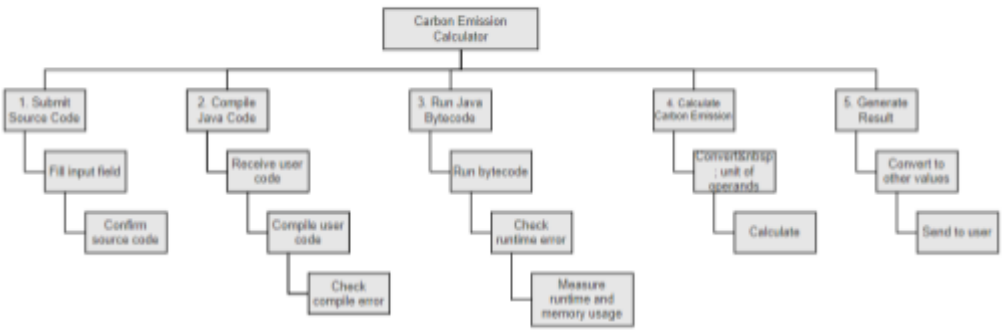


그림 3.7.7.1 Functional Hierarchy

3.7.7. Additional Comments

사용자는 배포된 실행 서버 환경의 하드웨어 스펙을 반영한 계산 결과만 제공받을 수 있으며, 사용자가 원하는 하드웨어 스펙에서의 탄소 배출량은 제공받지 못한다. 이러한 제약 사항에 대해 사용자가 인지하고 서비스를 이용할 수 있도록 충분한 설명이 기술되어야 한다.

4. Appendixes

4.1. 탄소 배출량 계산식 (Green Algorithm)

$$C = E \times CI$$

$$E = t \times (n_c \times P_c \times u_c + n_m \times P_m) \times PUE \times 0.001$$

$$C = t \times (n_c \times P_c \times u_c + n_m \times P_m) \times PUE \times CI \times 0.001$$

기호	설명	단위	비고
C	Carbon Impact (탄소 배출량)	gCO2e	$C = E \times CI$
E	전체 에너지 소비량	kWh	-
t	Runtime (사용자 코드 실행 시간)	h	실제로 사용자 코드를 실행할 때에는 런타임이 ms 단위로 굉장히 짧은 경우가 대부분이지만, 계산 시 h로 변환하여 사용해야 함
n_c	사용하는 코어 수	-	본 시스템에서는 사용자 코드를 실행하기 위해 하나의 코어를 isolation하여 사용함
P_c	Thermal Design Power	Watt	CPU 모델마다 제조

	(TDP - CPU 전력 사용 비율)		회사에서 상수값으로 제공함
u_c	CPU 사용 비율	%	사용한 전체 코어에 대한 평균값으로 사용하나, 실시간으로 추적하기 힘든 관계로 1로 설정
n_m	사용 가능한 메모리 사이즈	GB	사용자 코드를 실행할 때의 메모리 Peak 사용량을 n_m으로 취급하며, 이를 위해 프로세스 status의 MAXRSS 값 사용
P_m	메모리 사용량에 따른 전력 소비 비율	Watt/GB	다양한 요인에 영향을 받으나, 이전 실험들을 기반으로 간단하게 0.3725 W/GB로 설정
PUE	Power Usage Effectiveness (데이터 센터의 에너지 효율성)	-	각 데이터 센터의 PUE에 대한 값이 제공됨
CI	Carbon Intensity (탄소 집약도)	gCO2e/kWh	지역에 따라 다른 값을 가짐

- x 0.001: W→ kW 단위 변환을 위해 곱한다.

4.2. IEEE STD 830-1998

CODEMETER 서비스의 요구사항 명세서는 IEEE에서 권장하는 표준 스타일인 IEEE STD 830-1998을 반영해 작성되었다.