Green Coding
디자인 명세서

소프트웨어공학개론

3조

김지수 배경엽 이동국 이유진 임현서 조준형

Ⅰ. Introduction

본 문서의 예상 독자, 문서의 목적 그리고 각 목차 별 내용에 대한 설명이다.

1.1 Readership

본 문서의 예상 독자들은 다음과 같다. 탄소 배출량 감소에 관심을 갖는 세계의 프로그래머. 또한 환경 보호에 관심을 가지고 코드의 탄소배출 영향력을 알고 싶어 하는 친환경 개발자 및 회사 프로젝트의 일원이 포함된다.

1.2 Scope

본 문서에서는 Green Coding의 System Architecture와 개발 과정에서 적용되는 Testing 기법 및 Development plan에 대해 다룬다.

1.3 Objective

본 문서의 목적은 System과 Development적인 관점에서의 Green Coding 프로젝트의 분석이다. 프로젝트는 Front-end, Back-end, Database 3가지 System Architecture의 측면에서 분석되었으며, 프로젝트의 품질을 보증하기 위한 Testing plan과 Development 계획을 포함한다.

1.4 Document Structure

Ⅱ. Introduction

2.1 Objectives

본 차례에서는 차후 시스템 설계 설명에 사용될 다어어그램과 툴들에 대한 설명을 포함한다.

2.2 Applied Diagrams

2.2.1 Used Tools

본 문서에서 사용되는 Diagram들은 Microsoft Powerpoint를 기반으로 작성되었다.

2.2.2 Use Case Diagram

Use case diagram은 시스템이 사용자와 상호작용하는 방식을 모델링하는 데 사용되는 UML (Unified Modeling Language) 다이어그램이다. 주로 시스템의 기능 요구사항을 표현하고, 시스템 외부의 엔터티(사용자 또는 다른 시스템)와의 관계를 시각적으로 나타낸다. Green Coding에서 사용자가 상호작용할 수 있는 게시판, code 입력등의 상호작용을 diagram으로 나타낼 수 있다.

2.2.3 Sequence Diagram

시퀀스 다이어그램(Sequence Diagram)은 UML 다이어그램의 한 종류로, 객체 간의 상호작용을 시간 순서대로 나타내는 다이어그램이다. 주로 시스템 내에서 발생하는 사건의 순서와 상호작용을 시각적으로 표현하는 데 사용된다. Green Coding Mainpage에서부터 시작하여 code input, code detection 그리고 output까지 일련의 과정을 표현할 수 있다.

2.2.4 Class Diagram

클래스 다이어그램(Class Diagram)은 객체 지향 소프트웨어 설계에서 사용되는 UML 다이어그램의 한 종류로, 시스템의 정적 구조를 시각적으로 표현한다. Green Coding에서는 사용자 / Web / Server를 시스템적 class로 분류하며, 시스템의 계층적인 모습을 보여줄 뿐 코드의 객체화와는 유사하지 않을 수 있다.

2.2.5 Context Diagram

컨텍스트 다이어그램(Context Diagram)은 상위레벨에서 시스템의 전체적인 구조와 외부 엔티티 간의 상호작용을 나타내는 다이어그램이다. 주로 시스템 분석 단계에서 사용되며, 시스템 경계를 정의하고 시스템과 외부 요소(사용자, 다른 시스템 등) 간의 데이터 흐름을 시각적으로 표현한다.

2.2.6 Project Scope

Green Coding은 환경 보호에 관심이 많은 개발자들을 위해 코드의 탄소 배출량을 검사하고, 불필요하게 낭비되는 탄소의 양을 줄이기 위한 Code Refactoring 서비르를 제공하는 웹 어플리케이션이다. 세계의 소프트웨어 엔지니어들에게 코드로 인한 환경오염을 재조명하고 환경 보호 인식을 각인시키고자 한다.

III. System Architecture Overall

3.1 Objectives

본 챕터에서는 Frontend에서 Backend 설계에 이르는 프로젝트의 웹 서비스 시스템 구성에 대해 설명한다.

3.2 System Organization

이 서비스는 클라이언트 - 서버 모델을 적용하여 설계되었으며, Frontend 어플리케이션은 사용자와의 모든 상호작용을 담당한다. 프론트 엔드 어플리케이션과 백 엔드 어플리케이션은 JSON 기반의 HTTP 통신을 통해 데이터를 주고받는다. Backend 어플리케이션은 설계 사양 요청을 Frontend로부터 컨트롤러로 배포하고, 처리한 후 JSON 형식으로 전달한다. Backend 어플리케이션은 사용자가 작성한 JAVA 코드를 전달받고 이를 실행하여 얻은 결과(수정 코드, 탄소 배출량 등)를 다시 내보낸다. 결과를 내보냄과 동시에 데이터베이스에 문제 및 코드의 정오, accuracy를 저장하고 사용자가 결과에 관한 정보를 요청하면 업데이트된 정보가 전달된다.

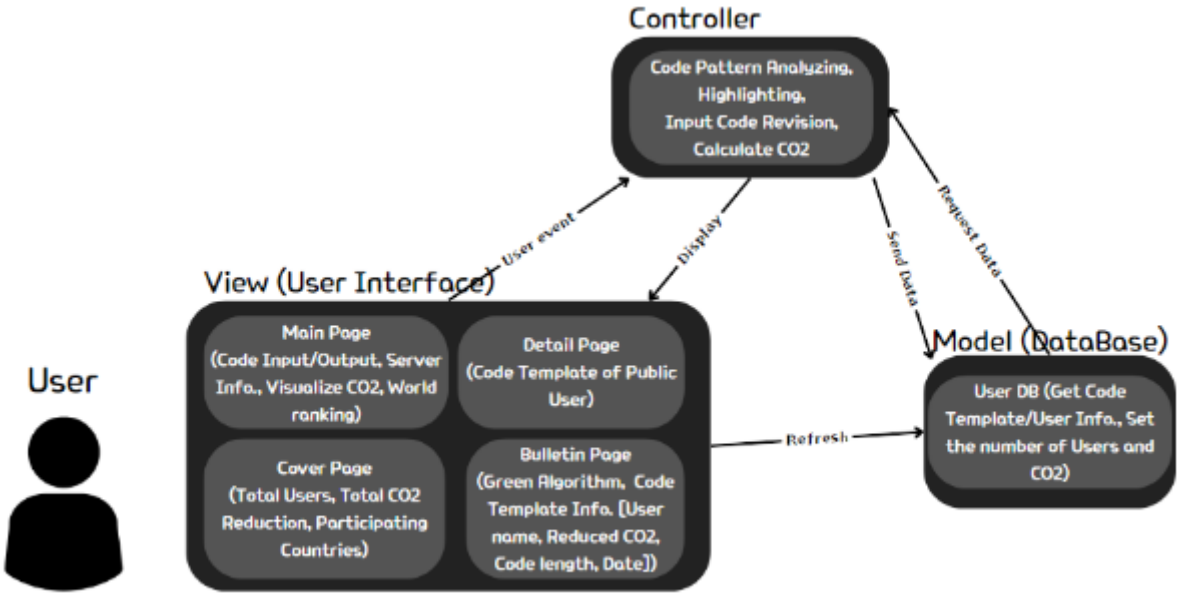


Figure 3.1: Overall System Architecture (MVC)

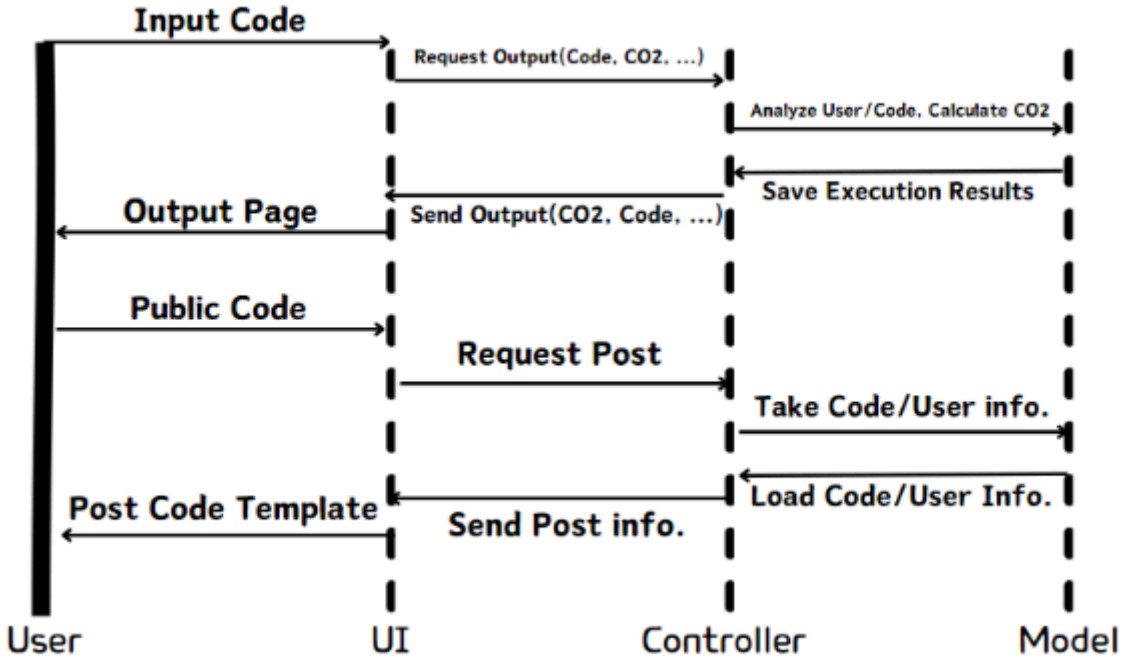


Figure 3.2: Sequence Diagram

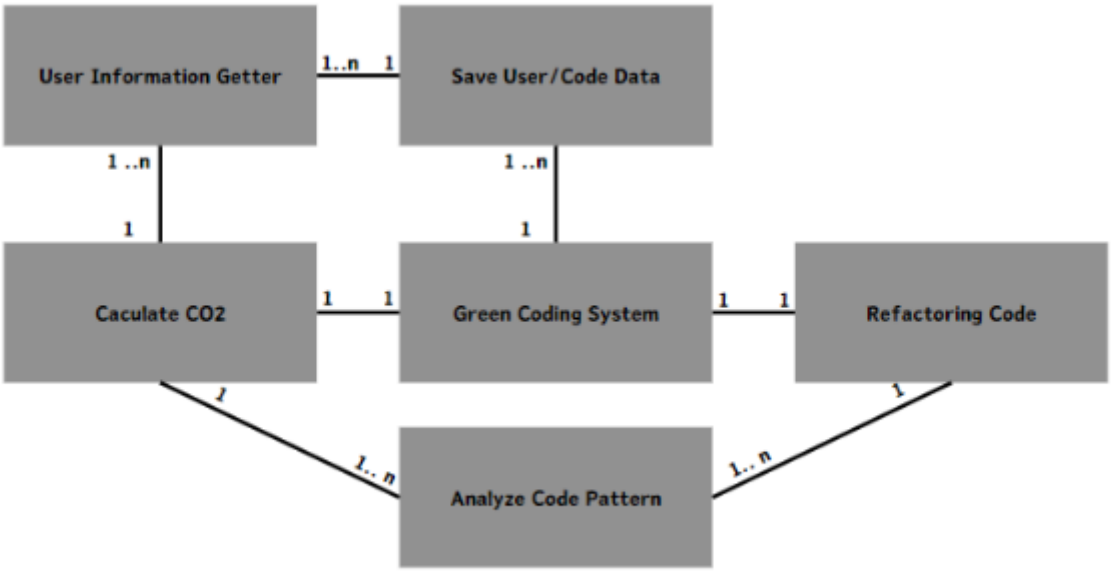


Figure 3.3: Context Diagram

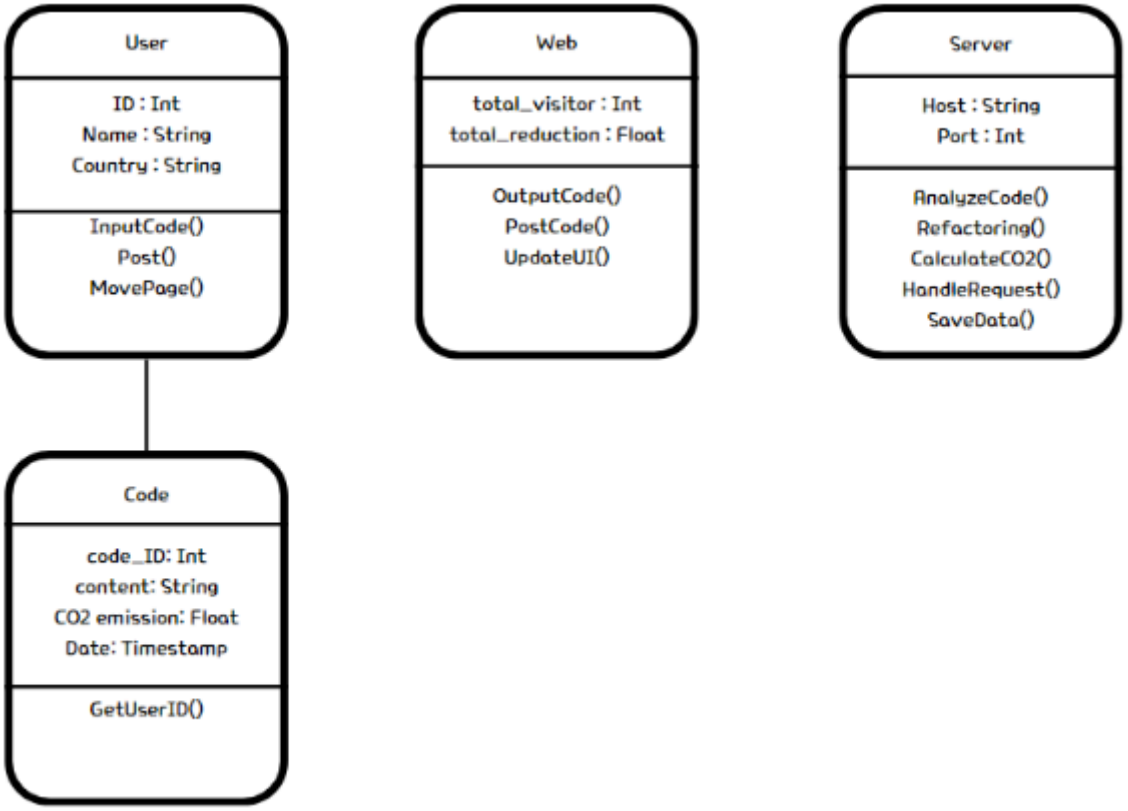


Figure 3.4: Class Diagram

3.3 Use Case Diagram

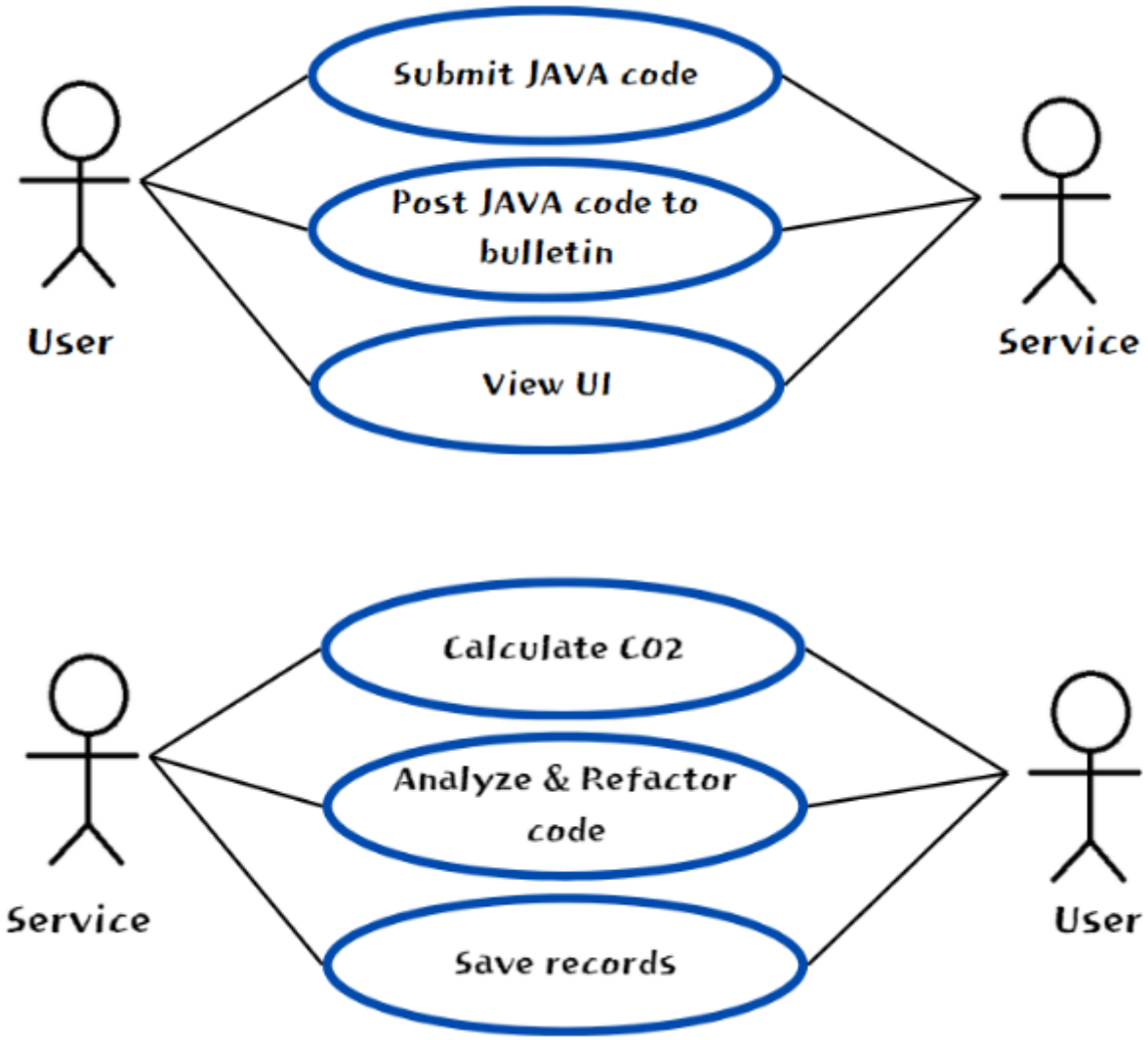


Figure 3.5: Use case Diagram

3.3.1 Use Case Description

1) User

Use case	**Submit JAVA code**
Actor	User

Description	사용자가 입력창에 JAVA 코드를 입력한다.
Stimulus	메인페이지에서 사용자가 자신의 JAVA 코드를 입력한 후 제출 버튼을 누른다.
Response	입력한 코드의 탄소 배출량, 수정된 코드, 사용자 정보(ID, 국가, 배출량 등) 기록
Comments	웹 서비스에서 코드 리팩토링 및 정보 제공을 위한 가장 기본적인 이벤트

Use case	**Post JAVA code to bulletin**
Actor	User
Description	입력을 마친 후 유저의 코드 템플릿을 공개한다.
Stimulus	메인페이지에서 사용자가 자신의 JAVA 코드를 입력한 후 코드를 게시판에 공개할지에 대한 물음에 '예' 버튼을 누른다. 1) 공개 사용자 : 자신의 github 아이디를 입력하여 코드 공유 2) 비공개 사용자 : 익명으로 코드 공유
Response	게시판 페이지에 알고리즘에 따른 코드 템플릿 정보 게시&기록
Comments	유저가 자신의 코드 템플릿을 공유하는 선택사항

Use case	**View UI**
Actor	User

Description	웹에서 제공하는 페이지 UI를 제공받는다.
Stimulus	페이지 이동 이벤트(버튼 클릭, 스크롤)
Response	탄소 배출량, 코드 리팩토링, 공개 유저 코드 템플릿 데이터
Comments	커버, 메인, 게시판, 디테일 페이지를 볼 수 있다.

2) Service

Use case	**Calculate CO2**
Actor	Service
Description	입력한 코드의 탄소 배출량을 API를 이용한 계산식을 통해 구한다.
Stimulus	사용자가 JAVA 코드를 제출했을 때
Response	탄소 배출량을 UI를 통해 표시
Comments	입력한 코드와 사용자 데이터를 분석하여 서비스 목표에 중요한 정보 제공

Use case	**Analyze & Refactor Code**
Actor	Service

Description	입력한 코드를 그린 알고리즘을 통해 탄소 낭비 패턴을 찾고 하이라이팅하여 수정된 코드를 출력한다.
Stimulus	사용자가 JAVA 코드를 제출했을 때
Response	코드 분석 및 리팩토링
Comments	사용자에게 코드 리팩토링 정보를 제공

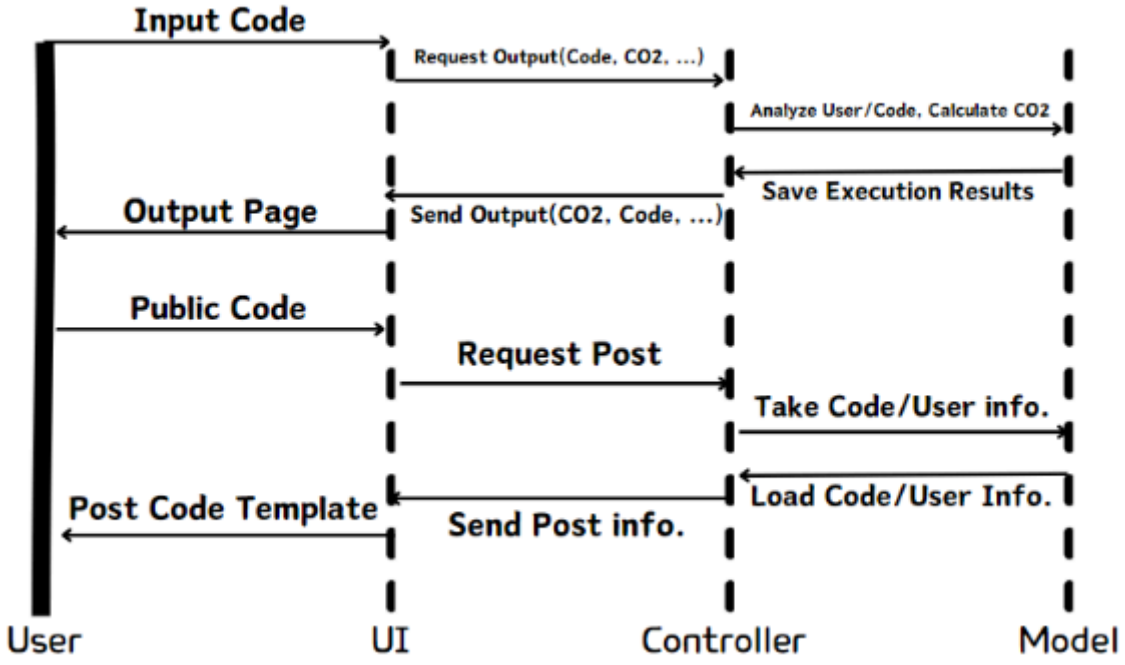
Use case	**Save records**
Actor	Service
Description	웹 서비스가 주는 유용한 정보들을 백엔드에서 저장한 후 필요할 때 제공한다.
Stimulus	사용자가 JAVA 코드 제출/게시 또는 페이지 이동할 때
Response	전체 탄소 감축량, 사용자 수, 참여 국가, 코드 템플릿/유저 정보
Comments	서비스가 만드는 데이터를 다채로운 시점에서 보여주기 위한 부가적인 정보 저장/기록

IV. System Architecture Frontend

4.1 Objectives

이 장에서는 Green Coding 웹 사이트 프론트엔드 시스템의 구조, 속성 및 기능을 기술하고, 다이어그램을 이용하여 표현한다.

4.1.1 Overall Sequence Diagram



4.2 Cover page

Green Coding 웹 사이트에 접속하면 가장 먼저 볼 수 있는 소개 페이지로, 웹 사이트의 이름인 Green Coding과 간단한 정보(총 방문자 수, 총 탄소 절감량, 참여국 목록)를 확인할 수 있다.

4.2.1 Attributes

Cover page는 스크롤을 통해 이동할 수 있는 세 개의 섹션으로 구성된다.

(1) 첫번째 섹션
- Title: 웹 사이트의 이름
- Cover Total Info: 총 방문자 수, 총 탄소 절감량

(2) 두번째 섹션
- Cover Countries: 참여국 국기

세번째 섹션은 Main page로 이동하기 위한 빈 섹션으로, 특별한 별도의 attribute가 없다.

4.2.2 Methods

Cover page의 각 섹션에 포함된 method를 설명한다.

(1) 첫번째 섹션
- setCoverTotalInfo(): Cover Total Info에 총 방문자 수와 총 탄소 절감량을 표시한다.

(2) 두번째 섹션
- setCoverCountries(): Cover Countries에 탄소 절감량이 많은 상위 6개국까지의 국기를 표시한다.

(3) 세번째 섹션
- moveToMainPage(): 스크롤을 통해 세번째 섹션에 도달하면 자동으로 Main page로 이동한다.

4.3 Main page

Green Coding 웹 사이트의 핵심 기능이 집약된 페이지로, 자바 코드를 입력하고 이를 처리한 결과(그린 패턴에 의해 개선된 자바 코드, 코드를 실행한 서버의 스펙, 입력 및 출력 코드에 대한 탄소 배출량, 탄소 절감량 및 이를 시각화한 정보 등)를 확인할 수 있다.

4.3.1 Attributes

Main page는 Header와 Body, Modal로 구성된다.

(1) Header
- Home Button: 웹 사이트의 로고가 그려져 있는 버튼
- Title: 웹 사이트의 이름
- Bulletin Button: Bulletin page로 이동하기 위한 버튼

(2) Body
- Input: 자바 코드 입력창
- Output: 개선된 자바 코드 출력창
- Submit Button: Input에 코드를 입력한 뒤 서버로 전송하기 위한 버튼
- Server Info: Submit Button을 통해 전송된 자바 코드를 실행한 서버의 스펙 정보
- Result: 입력 및 출력 코드의 탄소 배출량, 탄소 절감량을 시각화한 정보
- World Ranking: 각 국가의 국기, 이름, 탄소 절감량 랭킹 정보

(3) Modal
- Modal: 사용자의 자바 코드와 Submit Button을 통해 실행 및 계산된 결과(개선된 자바 코드, 입력 및 출력 코드의 탄소 배출량, 탄소 절감량, 자바 코드를 실행한 서버의 스펙 정보)를 Bulletin page 및 Detail page에 공개할 것인지에 대한 모달창. github ID 첨부 여부를 선택할 수 있다.

4.3.2 Methods

Main page의 각 구성 요소에 포함된 method를 설명한다.

(1) Header
- moveToMainPage(): Home Button 클릭 시 Main page로 이동한다.
- moveToBulletinPage(): Bulletin Button 클릭 시 Bulletin page로 이동한다.

(2) Body
- submitInputCode(): Submit Button 클릭 시 Input에 입력된 사용자의 자바 코드와 사용자의 국가를 서버로 전송한다.
- setInputEmission(): 입력 코드의 탄소 배출량을 표시한다.
- setOutputEmission(): 개선된 출력 코드의 탄소 배출량을 표시한다.
- setOutputCode(): 개선된 출력 코드를 표시한다. 이때, 개선된 부분을 하이라이팅하여 강조한다.
- visualize(): 탄소 절감량을 시각화한 정보를 표시한다.
- setServerInfo(): 자바 코드를 실행한 서버의 스펙 정보를 표시한다.
- setWorldRanking(): 탄소 절감량이 많은 순으로 국가의 국기, 이름, 절감량을 표시한다.

(3) Modal
- submitModal(): 사용자가 코드 및 결과 공개를 선택한 경우, code_id, anonymous, github_id를 서버로 전송한다.

4.4 Bulletin page

Main page의 우측 상단의 Bulletin Button을 클릭해 진입할 수 있는 게시판 페이지로, Main page에서 공개 설정된 코드에 한해서 열람할 수 있다. 게시글 목록은 총 다섯 개의 알고리즘 타입에 의해 분류돼있다.

4.4.1 Attributes

Bulletin page는 Header, Body로 구성된다.

(1) Header

4.3 Main page의 Header와 동일하다.

(2) Body
- Algorithm Type: Main page에서 입력된 코드를 실행 및 계산하는 과정에서 분류된 알고리즘 패턴의 타입
- List: 각 Algorithm Type에 해당하는 게시글 목록
- Content: 각 List에 해당하는 게시글. github id, reduced amount, code length, date 정보가 표시돼있다.

4.4.2 Methods

Bulletin page의 각 구성 요소에 포함된 method를 설명한다.

(1) Header

4.3 Main page의 Header와 동일하다.

(2) Body
- moveToAlogorithmType(): 각 알고리즘 타입에 대한 List로 이동한다.
- setContent(): 각 Content의 github id, reduced amount, code length, date 정보를 표시한다.

4.5 Detail page

Bulletin page에서 각 Content, 즉 게시글을 클릭해 진입할 수 있는 페이지로, 각 게시글에 대한 세부 정보(입력 및 개선된 출력 코드와 각각의 탄소 배출량, 탄소 절감량, 자바 코드를 실행한 서버의 스펙 정보)를 포함한다.

4.5.1 Attributes

Detail page는 Header, Body로 구성된다.

(1) Header

4.3 Main page, 4.4 Bulletin page의 Header와 동일하다.

(2) Body
- Input: 자바 코드 입력창
- Output: 개선된 자바 코드 출력창
- Server Info: Submit Button을 통해 전송된 자바 코드를 실행한 서버의 스펙 정보
- Result: 입력 및 출력 코드의 탄소 배출량, 탄소 절감량을 시각화한 정보

4.5.2 Methods

Detail page의 각 구성 요소에 포함된 method를 설명한다.

(1) Header

4.3 Main page, 4.4 Bulletin page의 Header와 동일하다.

(2) Body
- setInputCode(): 사용자의 입력 코드를 표시한다.
- setOutputCode(): 개선된 출력 코드를 표시한다. 이때, 개선된 부분을 하이라이팅하여 강조한다.
- setInputEmission(): 입력 코드의 탄소 배출량을 표시한다.
- setOutputEmission(): 개선된 출력 코드의 탄소 배출량을 표시한다.
- setServerInfo(): 자바 코드를 실행한 서버의 스펙 정보를 표시한다.
- visualize(): 탄소 절감량을 시각화한 정보를 표시한다.

V. System Architecture Backend

5.1 Objectives

이 챕터는 FastAPI 백엔드 시스템의 구조를 기술합니다. Docker와 AWS EC2를 활용하여 배포되는 시스템의 주요 구성 요소와 그 상호작용을 명확히 설명하는 것을 목표로 합니다.

백엔드 시스템의 목표는 다음과 같습니다:

- 확장성: 증가하는 트래픽을 처리할 수 있도록 시스템을 확장 가능하게 설계합니다.
- 신뢰성: 시스템이 항상 안정적으로 작동하도록 보장합니다.
- 유지보수성: 코드와 아키텍처가 쉽게 이해되고 수정될 수 있도록 합니다.
- 보안: 사용자 데이터와 시스템 자산을 보호합니다.
- 성능: 빠른 응답 시간과 높은 처리량을 유지합니다.

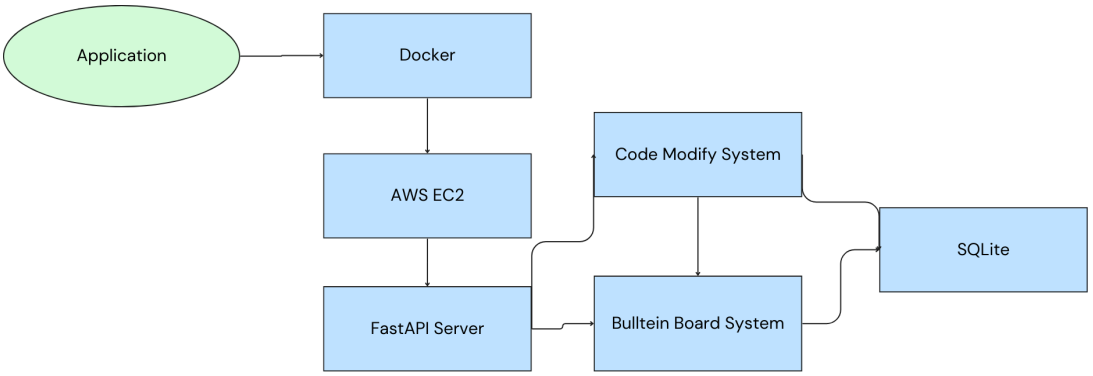


Figure 5.1: Overall Architecture

1. Application ↔ Docker
    - 사용자가 애플리케이션을 통해 요청을 보내면, 이 요청은 Docker 컨테이너 내에서 실행되는 FastAPI 서버로 전달됩니다.
2. Docker ↔ AWS EC2
    - Docker 컨테이너는 AWS EC2 인스턴스에서 실행됩니다. EC2 인스턴스는 Docker 컨테이너를 호스팅하여, FastAPI 서버 및 기타 서비스들이 안정적으로 실행될 수 있도록 지원합니다.
3. FastAPI Server ↔ Code Modify System & Bulletin Board System
    - FastAPI 서버는 Code Modify System과 Bulletin Board System을 통해 사용자 요청을 처리합니다. 사용자가 코드 수정이나 게시판 관련 요청을 보내면, FastAPI 서버가 이를 처리하여 SQLite 데이터베이스에 반영합니다.
4. Code Modify System & Bulletin Board System ↔ SQLite
    - Code Modify System과 Bulletin Board System은 SQLite 데이터베이스를 사용하여 데이터를 저장하고 관리합니다. FastAPI 서버를 통해

데이터베이스와 상호작용하며, 필요한 데이터를 읽고 씁니다.

5.2 Subcomponents

5.2.1 Code Modify System

Class Diagram

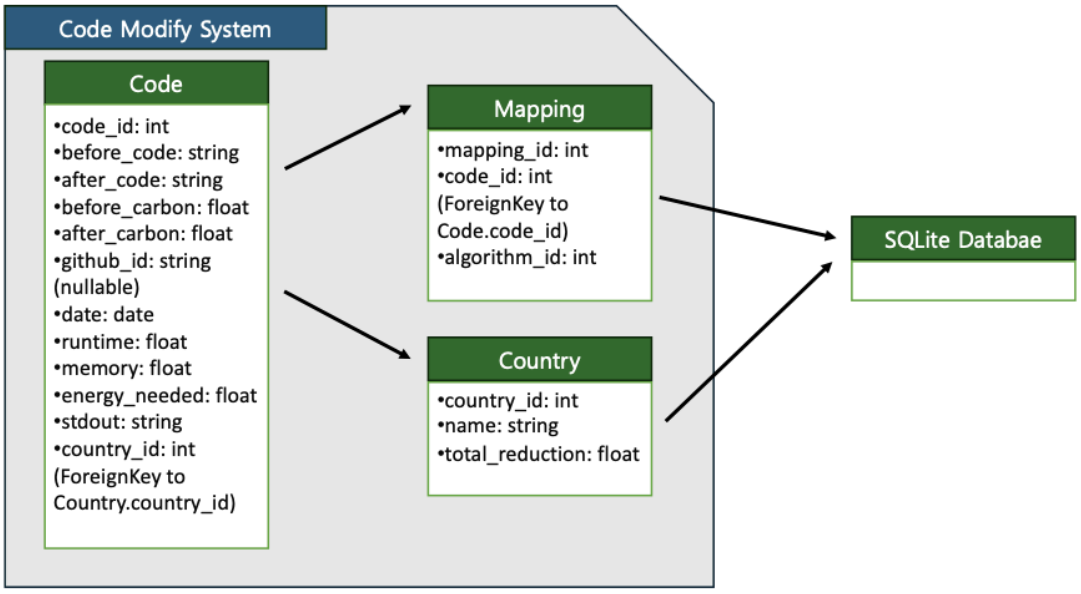


Figure 5.2: Class Diagram - Code Modify System

Mapping class: 입력된 코드에 맞는 알고리즘을 매핑하고, Database에서 알고리즘 데이터를 확인하고, 불러오는 역할을 한다.

Country Class: 입력된 코드의 국가를 Databse에 저장하고 불러온다.

Sequence Diagram

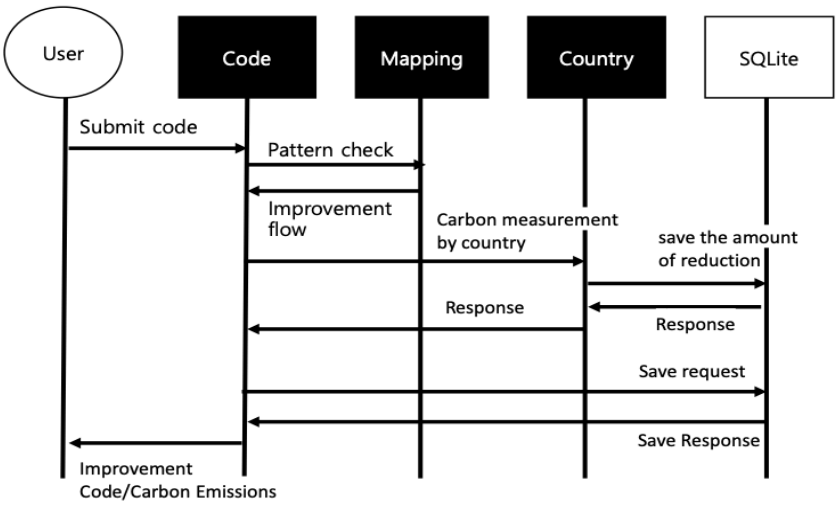


Figure 5.3: Sequence Diagram - Code Modify System

5.2.2 Code Modify System

Class Diagram

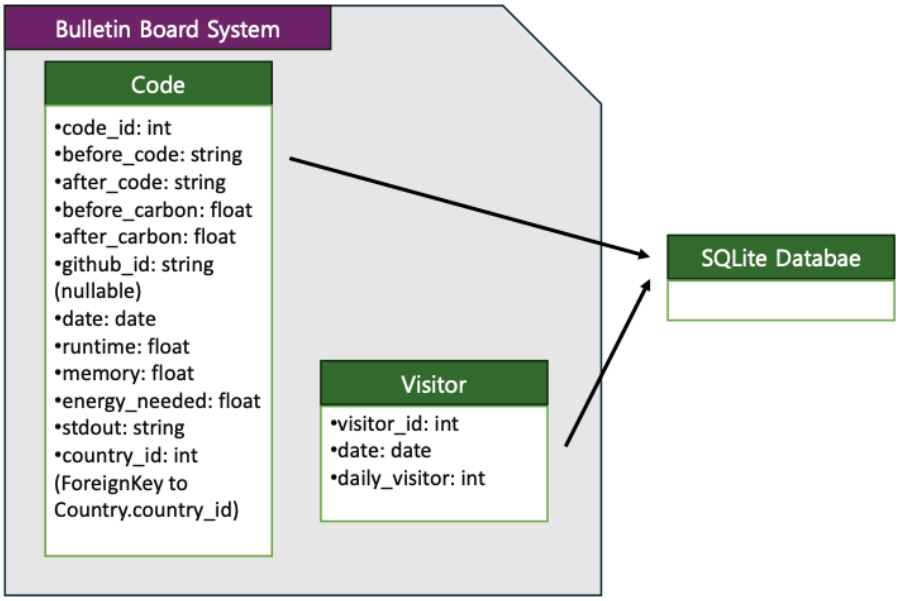


Figure 5.4: Class Diagram - Bulletin Board System

Code class: 하나의 입력된 코드를 하나의 객체로 보아 변경된 코드, 탄소 감축량 등 다양한 attribute들을 연산하고 Database에 저장한다.

Visitor class: 서비스의 방문자 수를 측정하고 Database에 저장한다.

Sequence Diagram

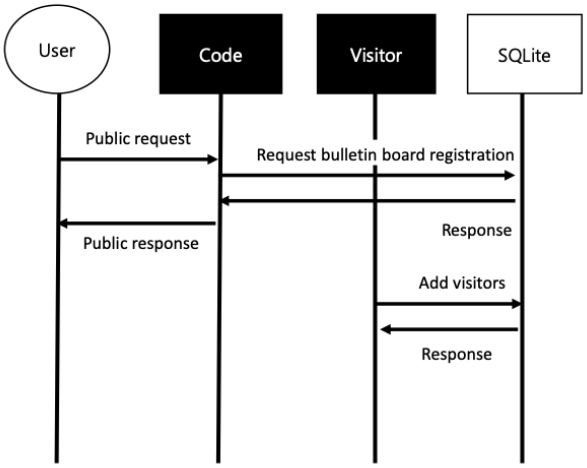


Figure 5.5: Sequence Diagram - Bulletin Board System

VI. Protocol Design

6.1 Ajax

AJAX(Asynchronous JavaScript and XML)는 비동기 자바스크립트와 XML을 의미하며, 서버와 통신하기 위해 XMLHttpRequest 객체나 Fetch API를 사용하는 것을 말한다. AJAX는 JSON, XML, HTML, 일반 텍스트 형식 등 다양한 포맷을 주고받을 수 있다. AJAX의 강력한 특징은 페이지 전체를 리프레시 하지 않고도 수행되는 비동기성으로, 이러한 비동기성을 통해 사용자의 이벤트가 있으면 전체 페이지가 아닌 일부분만을 업데이트할 수 있다.

리액트와 FastAPI를 사용하는 프로젝트에서는 주로 JSON 포맷을 사용하여 클라이언트와 서버 간의 데이터를 주고받는다. 리액트는 Fetch API 또는 Axios와 같은 라이브러리를 사용하여 AJAX 요청을 보낸다.

6.2 TLS and HTTPS

TLS(Transport Layer Security)는 웹사이트와 브라우저 사이(또는 두 서버 사이)에 전송되는 데이터를 암호화하여 인터넷 연결을 보호하기 위한 표준 기술이다. HTTPS(HyperText Transfer Protocol over Transport Layer Security)는 월드 와이드 웹 통신 프로토콜인 HTTP를 TLS 위에서 패킹하여 강화된 보안을 제공하는 버전이다.

리액트와 FastAPI를 사용하는 프로젝트에서는 HTTPS를 통해 데이터 전송을 암호화하여 보안을 강화한다. 이를 위해 FastAPI 서버는 TLS 인증서를 설정하고, 클라이언트는 HTTPS URL을 통해 서버와 통신한다.

6.3 CSRF and CSRF Token

CSRF 공격(Cross Site Request Forgery)은 웹 애플리케이션 취약점 중 하나로 인터넷 사용자(희생자)가 자신의 의지와는 무관하게 공격자가 의도한 행위(수정, 삭제, 등록 등)를 특정 웹사이트에 요청하게 만드는 공격이다. CSRF Token은 이러한 공격을 방지하기 위해 서버에서 난수 토큰을 생성하여 서버에 요청을 주는 페이지에 부여하여 정당하지 않은 페이지에서 오는 요청을 막는 기법이다.

리액트와 FastAPI를 사용하는 프로젝트에서는 CSRF 방지를 위해 CSRF

토큰을 사용한다. 서버는 클라이언트에 CSRF 토큰을 제공하고, 클라이언트는 이를 각 요청 시 포함하여 서버에 전달한다. FastAPI에서는 fastapi-csrf-protect 같은 라이브러리를 사용할 수 있다.

6.4 Restful API

Representational State Transfer(REST)는 API 작동 방식에 대한 소프트웨어 아키텍처이다. REST 기반 아키텍처를 사용하여 대규모의 고성능 통신을 안정적으로 지원한다. 또한 쉽게 구현하고 수정할 수 있어 API 시스템 파악에 용이하다. RESTful API는 REST가 클라이언트-서버 상호 작용을 최적화하고 완전한 분리를 지원하여 확장성, 유연성, 독립성 측면에서 이점을 갖는다.

6.5 API Description

6.5.1 Cover Page

Method	GET	
API Path	/api/default	
Description	커버페이지 입장 시 보여지는 방문자 수, 총 탄소배출 감소량, 참여 국가 정보들을 포함한 통계 조회	
Status Code	200 OK	
Response(200)	daily_visitor	integer
	total_visitor	integer
	daily_reduction	integer
	total_reduction	integer

	countries	Array of Objects

Table 6.1 Properties of Get Cover Page API

6.5.2 Main Page

Method	POST	
API Path	/api/code	
Description	코드 실행을 위한 요청을 받고 결과를 반환	
Status Code	200 OK , 400 Bad Request	
Request	code	String
	country	String
Response(200)	code_id	Integer
	runtime	Integer
	memory	Integer
	before_code	string
	after_code	String
	before_carbon	Integer
	after_carbon	Integer

	energy_needed	Integer
	runtime_stdout	String
	algorithm_type	Integer
Response(400)	compile_stderr	String
	runtime_stderr	String

Table 6.2 Properties of Create Code Submission API

Method	POST	
API Path	/api/sharing	
Description	공개 여부 전송	
Status Code	200 OK	
Request	code_id	Integer
	anonymous	boolean
	github_id	String

Table 6.3 Properties of Create Sharing API

6.5.3 Bulletin Page

Method	GET
API Path	/api/bulletin/{algorithm_type}

Description	알고리즘 유형별로 게시물을 가져옵니다.	
Status Code	200 OK	
Request	codes	Array of Object
	code_id	Integer
	before_carbon	Integer
	after_carbon	Integer
	github_id	String
	runtime	Integer
	memory	Integer

Table 6.4 Properties of Get Bulletin Lists API

6.5.4 Detail Page

Method	GET	
API Path	/api/detail/{code_id}	
Description	특정 코드 ID에 대한 상세 정보 가져오기	
Status Code	200 OK	
Request	code_id	Integer

Response(200)	code_id	Integer
	runtime	Integer
	memory	Integer
	before_code	String
	after_code	String
	before_carbon	Integer
	after_carbon	Integer
	energy_needed	Integer
	runtime_stdout	String
	algorithm_type	Number
	anonymous	Boolean
	github_id	String

Table 6.5 Properties of Get Code Details API

VII. Database Design

7.1 Objectives

7장에서는 데이터베이스 디자인을 명확히 정의하고, 시스템에서 관리할 데이터 구조를 설명한다. 데이터베이스는 ORM방식으로 생성되며, 각 엔티티의 기본 키 생성을 데이터베이스에 위임한다. 또한, 데이터 모델을 시각화하여 데이터베이스 엔티티 간의 관계를 이해하기 쉽게 한다. 이를 통해 데이터베이스 설계의 효율성을 높이고, 개발 과정에서 데이터 무결성과 일관성을 유지한다.

7.2 ER Diagram

본 어플리케이션 시스템은 총 4가지 entity로 이루어져 있다. (Code, Mapping, Country, Visitor) ER-Diagram은 entity간의 관계를 나타내며, 각각의 entity와 attribute의 관계를 세부항목(7.2.*)에서 설명한다. entity간의 관계는 entity를 연결하는 선 주변에 표기되어 있어 확인 가능하다.

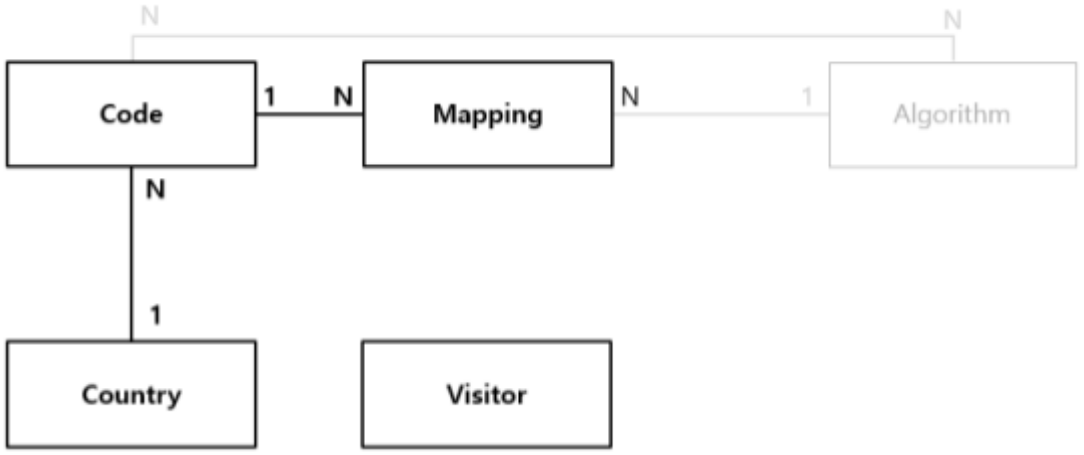


Figure 7.1: ER-Diagram

7.2.1 Code

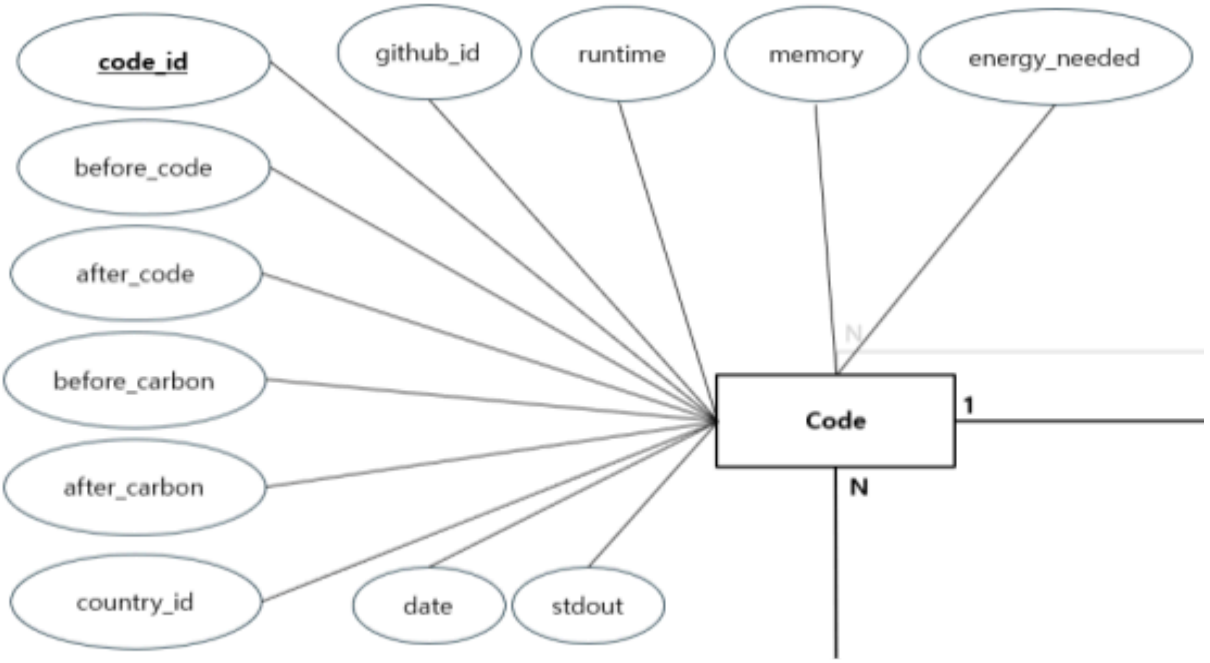


Figure 7.2: Code entity

Code entity 는 사용자가 서버로 전송한 코드에 대한 정보를 담는다. Integer형의 code_id를 기본 키로 사용한다. Code Entity 는 before_code, after_code, before_carbon, after_carbon, country_id, github_id, date, stdout, runtime, memory, energy_needed를 attribute로 가진다. 사용자가 자바 코드를 전송하게 되면 개선 패턴에 의한 수정, 탄소 배출량 측정 이후 attribute들이 저장된다. 사용자가 코드 공개/비공개를 결정할 수 있다. github_id attribute는 nullable하며, country_id attribute는 Country entity의 기본키를 참조하는 외래키이다.

7.2.2 Mapping

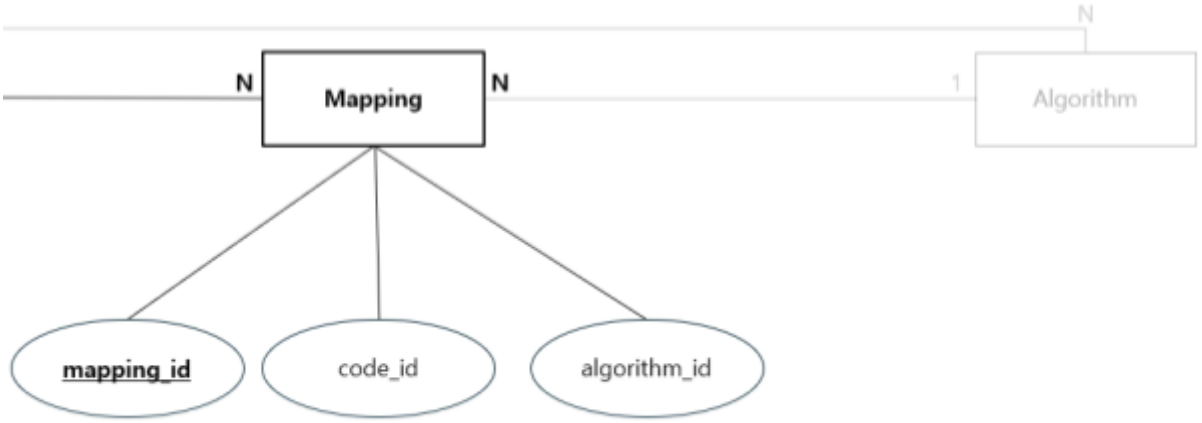


Figure 7.3: Mapping entity

Mapping entity는 Code entity와 가상의 Algorithm entity의 N:N관계를 중개하는 entity이다. Mapping Entity는 Integer형의 mapping_id를 기본키로 사용한다. code_id와 algorithm_id는 각각 Code entity와 Algorithm entity의 기본키를 참조하는 외래키이다. Algorithm entity는 Integer형의 기본키 attribute만을 가지므로 실제로 구현하지 않는다.

7.2.3 Country

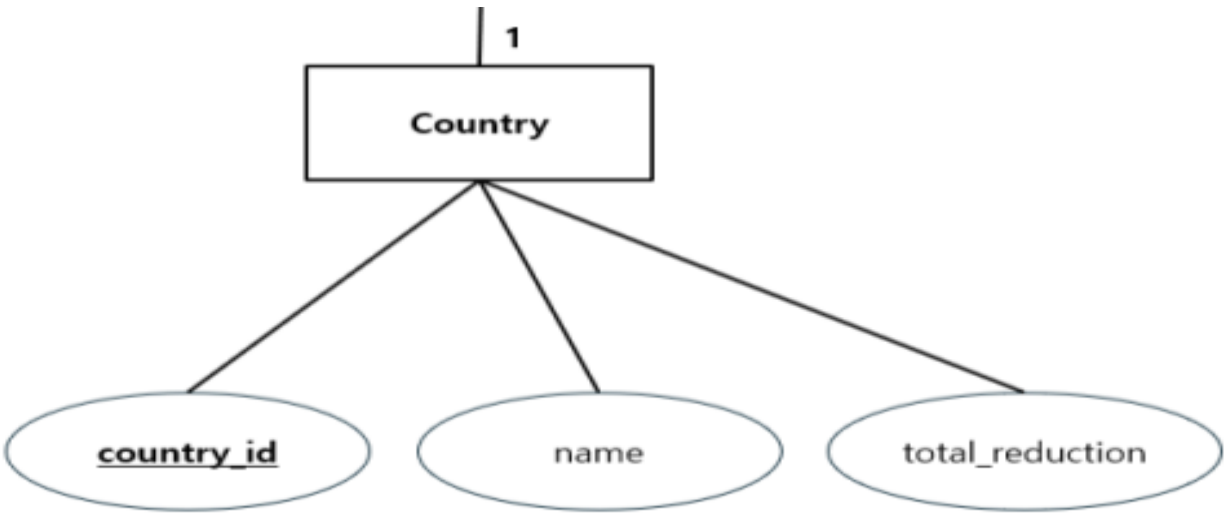


Figure 7.4: Country entity

Country entity는 애플리케이션에서 국가에 해당한다. country_id, name, total_reduction을 attribute로 가지며, Integer형의 country_id를 기본키로 사용한다. 사용자의 코드를 전송하게 되면 사용자가 속한 국가를 확인하고, 국가별로 탄소 배출 감소량이 누적된다.

7.2.4 Visitor

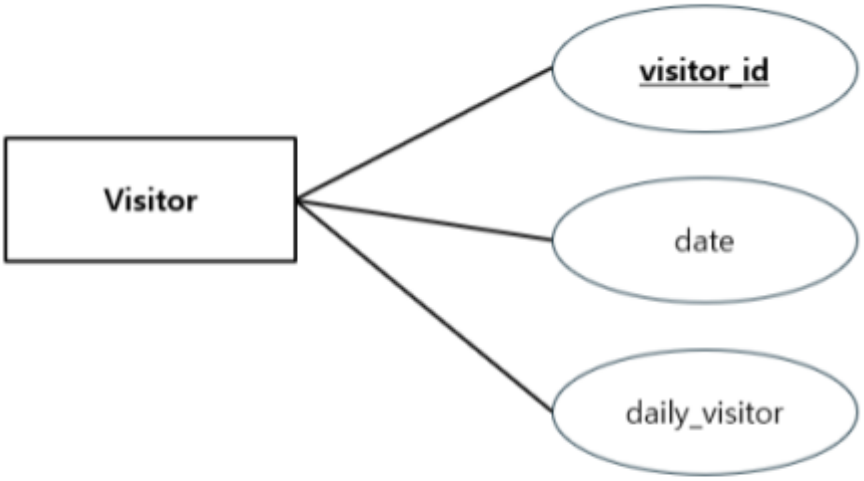


Figure 7.5: Visitor entity

Visitor entity는 애플리케이션 방문자 수에 해당한다. visitor_id, date, daily_visitor를 attribute로 가지며, visitor_id를 기본키로 사용한다. date attribute는 한국 기준 Date형이며, daily_visitor attribute는 해당 날짜의 애플리케이션 방문자수이다.

7.3 Relational Schema

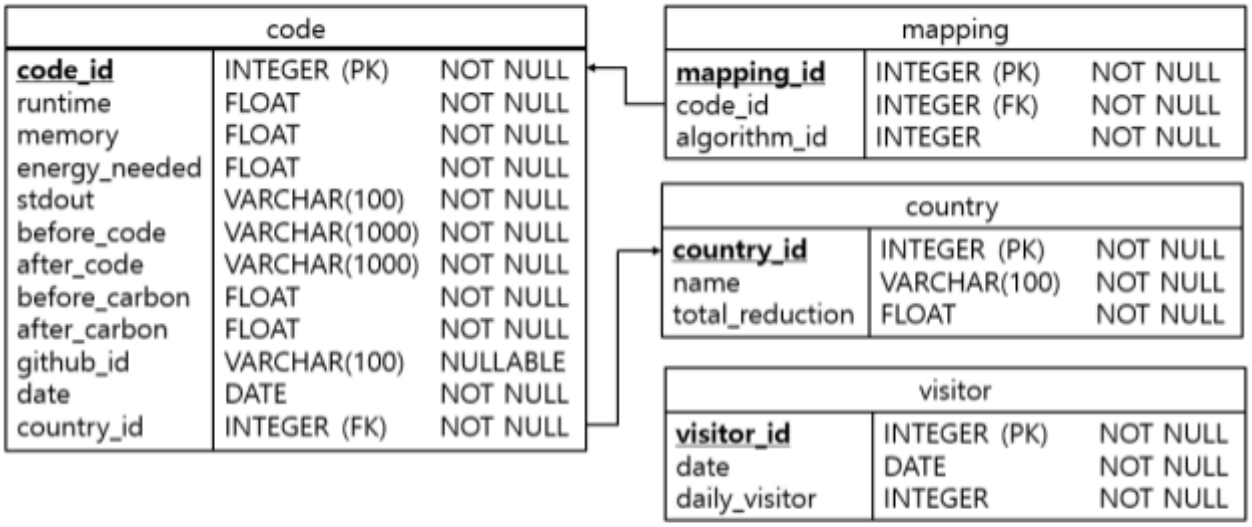


Figure 7.6: Relational Schema

VIII. Testing Plan

8.1 Objectives

이번 차례에서는 Green Coding 프로젝트 Development 과정에서 진행되는 testing process에 대해 설명한다.

8.2 Testing Policy

8.2.1 Automated Development Testing

Green Coding은 Frontend, Backend, Logics 3가지 파트로 나뉘어 개발된다.각 개발 내용이 병합되는 과정에서 Sub System간의 충돌을 방지하기 위해 매 병합마다 Automated Testing이 진행된다. 각 Testcase는 Input Code와 Output value로 구성되어 있으며, Output Value에는 탄소 낭비 코드 유무, 수정된 라인과 refactored code로 구성되어 있다. 이러한 Test Case는 어플리케이션의 Reliability에 중점이 맞춰져 있다.

8.2.2 Release Testing

Green Coding은 요구사항을 만족하는 환경을 가지는 모든 사용자에 대해 원활한 서비스를 제공하기 위해, 서비스 배포 중에서 Testing을 가진다. 이러한 Test Case는 웹 어플리케이션의 Performance에 중점이 맞춰져 있다.

Performance

- 계산 결과 및 refactored된 코드는 Submit버튼 클릭 후 5초 이내에 배출된다.
- 사용자의 정보와 코드를 저장하는 과정은 5초 이내에 완료된다.
- 게시판 page와 Detail page를 불러오는 과정은 3초 이내에 완료된다.
- 국가별 이용량 및 하루 이용량의 표시는 실시간 성을 보장한다

.

Reliability

- 사용자의 코드가 실행이 불가할 경우, ERROR를 반환한다.
- Refactoring 후에 사용자의 코드의 기능이 변하지 않음을 보장한다.
- 요구사항을 만족하는 크기 이내의 큰 용량의 파일에도 성능 감소가 없다.

8.2.2 Test Cases

Test case의 목적은 Performance와 Reliability에서 언급된 사항들을 검수하기 위해서 이다. 각 특성 별로 최소 5개 이상의 testcase를 구현하여 어플리케이션의 품질을 보장한다.

IX. Development Plan

9.1 Objectives

해당 챕터에서는 시스템의 개발 환경 및 기술에 대하여 설명한다..

9.2 Frontend Environment

9.2.1 JavaScript

JavaScript는 객체 기반의 프로그래밍 언어로, 동적인 웹 페이지를 만들고 사용자와 상호작용 하는 데 사용된다. Green coding에서는 JavaScript를 사용하여 웹 애플리케이션의 핵심 로직을 작성하고, React와 함께 사용하여 동적이고 효율적인 UI를 구현한다.

figure 9.1 javascript logo

9.2.2 ReactJS

React는 사용자 인터페이스를 만들기 위한 Javascript 라이브러리로, 재사용 가능한 컴포넌트 기반 구조의 개발을 지원한다. 가상 DOM을 사용하여 효율적으로 UI를 업데이트하고 선언적 프로그래밍을 통해 가독성과 유지보수성을 높인다. Green coding은 React를 사용해 재사용 가능한 컴포넌트를 생성하고 상태 관리를 통해 유저의 입력에 따른 동적인 UI를 제공한다.

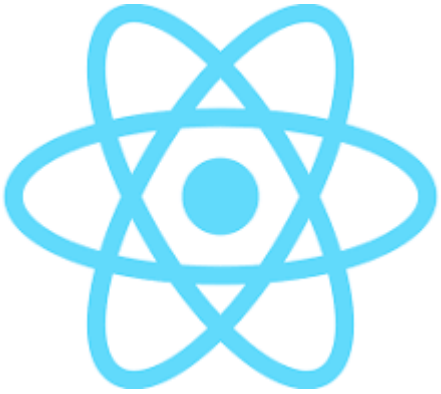


figure 9.2 ReactJS logo

9.2.3 TailwindCSS

TailwindCSS는 Utility-First 컨셉의 CSS 프레임워크로, 클래스 기반 스타일링을 제공한다. 사전에 정의된 유틸리티 클래스를 활용해 HTML 코드 내에서 UI를 빠르게 스타일링할 수 있으며, 커스텀 디자인 시스템을 쉽게 구축할 수 있다. 또한 JIT(Just-in-Time) 컴파일러를 통해 빌드 시간을 줄이고 개발 생산성을 향상시킨다. Green coding에서는 TailwindCSS를 사용하여 UI를 빠르게 구축하고 불필요한 class-name을 짓는 데 걸리는 시간을 단축한다.

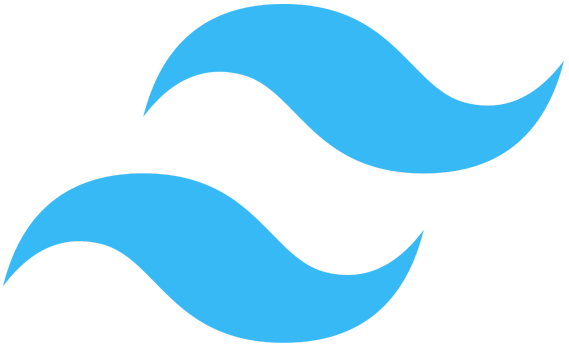


figure 9.3 TailwindCSS logo

9.3 Backend Environment

9.3.1 Fast API

FastAPI는 Python을 사용하여 고성능의 API를 빠르고 쉽게 작성할 수 있도록 도와주는 웹 프레임워크이다. FastAPI의 특징으로는 파이썬 프레임워크 중 가장 빠른 속도를 자랑하며, 구현이 쉽고 짧다는 것이 있다. 또한 api에 대해 개방형 표준 기반을 적용함으로써 편리한 frontend와 연결을 제공한다.

figure 9.4 FastAPI logo

9.3.2 SQLite

SQlite는경량의 관계형 데이터베이스 관리 시스템(RDBMS)이다. 서버 없이 작동하며, 임베디드 데이터베이스로 설계되었다. SQLite는 파일 기반으로 작동하며, 데이터베이스의 모든 데이터는 단일 파일에 저장된다. SQLite는 다양한 환경에서 사용되며, 별도의 설치가 필요 없고, 복잡한 설정 없이 바로 사용할 수 있기 때문에 소규모 프로젝트나 단일 사용자 애플리케이션에 적합하다.

figure 9.5 SQLite logo

9.4 Constraints

- 상용화 되어 있는 tool 및 framework를 사용한다
- 보안 및 non-functional한 성능의 향상이 사용자의 편의성을 저해하지 않는다.
- 시스템 유지보수를 고려한 문서화 및 코드 refactoring을 장려한다.
- 예상 사용자 수에 따라 서버 크기의 증감을 결정한다.
- 개인 코드에 대한 지적 재산권을 보장한다.
- input code는 java code로 제한한다.
- 시스템은 24시간 full-time 가동을 전제로 한다.
- 서버는 3초 이내에 코드 분석과 refactoring을 완료해야 한다.

- System UI는 사용자의 편의성을 고려하여 Design된다.
- 요구사항을 만족하는 모든 시스템 환경에 대해서 신뢰성을 보장한다.
- Source code는 energy efficiency를 고려하여 작성한다.

9.5 Assumptions and Dependencies

Green Coding System은 데스크탑 웹 어플리케이션 환경을 전제로 만들어진 시스템이다. 윈도우 XP 이후의 운영체제를 타겟으로 만들어진 어플리케이션으로, 이전의 운영체제에 대해서는 성능을 보장할 수 없다.