Software Design Specification

Eco Master

by

임성훈, 황인성, 양승환, 김민성, 문수현

TEAM 8

Instructor: 이은석

Document Date: 2024/05/24

Faculty: Sungkyunkwan University

Contents

1

Preface

1.1 Readership

본 문서는 시스템 제작에 관여하는 독자들의 이해를 돕기 위해 작성되었습니다. 본 문서의 주요 독자는 5 명으로 구성된 Team 8 과 소프트웨어공학개론 강의 조교님, 이은석 교수님, 그리고 2024 년 1 학기 소프트웨어공학개론 수강생들입니다. 이 문서를 상업적 용도로 사용하려면 Team 8 의 허가를 받아야 하며, 학습 및 교육 용도, 정보 취득 목적으로 사용할 경우에는 재배포 및 수정에 제한이 없습니다.

1.2 Scope / Objective

본 문서는 사용자가 작성한 Java 코드의 탄소배출량과 관련 지표를 직관적으로 확인할 수 있는 EcoMaster 서비스의 개요와 디자인 패턴을 소개하기 위한 문서입니다. 이 서비스는 성균관대학교 이은석 교수님의 2024 년 1 학기 소프트웨어공학개론 수업을 수강한 Team 8 멤버들에 의해 제작되었습니다. 전 세계적으로 탄소배출량이 기하급수적으로 증가하고 있으며, 이러한 추세는 앞으로도 계속될 것으로 예상됩니다. 특히, AI 의 수요와 발전이 가속화되면서 소프트웨어 개발로 인한 탄소배출량이 증가하고 있습니다. 따라서, 개발자들이 자신이 작성한 코드의 탄소배출량을 명확히 인식하고 친환경적인 코드 개발에 노력하는 것이 중요합니다. 이에 따라, 본 개발진은 사용자가 자신의 PC 사양을 입력하고 Java 코드를 제출하면, 탄소배출량에 대한 객관적인 지표를 제공하고, 이를 기반으로 개발자들이 Java 코드를 수정하여 점차 탄소배출량을 줄일 수 있도록 하는 것을 목표로 하고 있습니다. 추가로, 본 문서는 클라이언트와 서버,

API 등을 설계하는 과정에서 도출된 요구사항 명세서를 바탕으로 작성되었습니다.

1.3 Document Structure

1) Preface: 본 문서의 Readership, Scope/Objective 에 관해 간결히 소개합니다.

2) Introduction: 본 문서를 작성하는데 사용된 다양한 다이어그램, 도구, Reference 들의 전반을 소개합니다.

3) Overall System Architecture: 시스템의 전체적 구조를 Context diagram, Use-case diagram, Sequence diagram 을 통해 설명합니다.

4) System Architecture (Frontend): Frontend 부문에 대한 시스템 구조를 기능별로 나누고, 각 부분을 다양한 Diagram 을 통해 설명합니다.

5) System Architecture (Backend): Backend 부문에 대한 시스템 구조를 기능별로 나누고, 각 부분을 다양한 Diagram 을 통해 설명합니다.

6) Protocol Design: 클라이언트와 서버 간의 통신을 위한 API 등의 프로토콜 디자인을 설명합니다.

7) Database Design: 데이터베이스에 저장될 데이터의 형식과 구조에 대해 설명합니다.

8) Testing Plan: 본 시스템의 이후 진행할 테스팅 계획에 대해 설명합니다.

9) Development Plan: 본 시스템을 개발하는 데 있어 사용된 개발 환경 및 도구에 관해 설명합니다.

10) Supporting Information: 본 문서의 수정 및 변경을 날짜별로 기록합니다.

2

Introduction

2.1 Objectives

이 장에서는 서비스를 디자인하는 데 사용된 다이어그램, 이러한 다이어그램을 만드는 데 사용된 도구 및 개발 범위에 대한 개요를 제공합니다.

2.2 Applied Diagrams

2.2.1 Used Tools

본 문서에서 사용된 다이어그램은 Google 문서를 통해 기본적으로 제공되는 도형과 아이콘을 활용하여 작성되었습니다.

2.2.2 Usecase Diagram

Use case diagram 은 시스템과 사용자 간의 상호작용을 그림으로 나타낸 것입니다.

시스템이 제공해야 하는 기능이나 서비스를 상세하게 정의한 것으로, 일반적으로 System, Actor, Use case, Relation 으로 구성됩니다.

System 은 구현하려는 서비스를 나타냅니다. Actor 는 시스템과 상호작용하는 외부 요소로, 사용자 또는 다른 시스템이 될 수 있습니다.

Use case 는 시스템이 Actor 에게 제공하는 기능 또는 서비스를 나타냅니다.

마지막으로 Relation 은 Actor 와 Use case 사이의 관계를 나타내며, 주로 네 가지 유형(Association, Include, Generalization, Extend)으로 구성됩니다.

2.2.3 Context Diagram

Context diagram 은 고수준의 데이터 흐름 다이어그램입니다. 시스템과 해당 시스템과 상호작용하는 외부 구성 요소들을 나타내며, 그들 간의 세부 정보 흐름을 보여줍니다. 많은 비즈니스에서 프로젝트의 위험 상황을 미리 대비하기 위해 사용됩니다. 앞서 언급한 대로, 세부 정보의 흐름을 파악하여 시스템을 더욱 세부적으로 이해하는 데 도움이 됩니다. 이는 개발자보다는 프로젝트 이해 관계자가 더 많이 사용하는데, 따라서 관련 내용을 쉽게 이해할 수 있도록 작성되어야 합니다.

2.2.4 Sequence Diagram

Sequence diagrams 은 특정한 순서대로 객체들이 상호작용하는 과정을 나타냅니다. 문제 해결에 필요한 객체들을 정의하고, 객체들 사이에서 교환되는 메시지들을 시간 순서대로 표시합니다. Sequence diagrams 의 구성요소로는 lifelines, activation boxes, messages, 그리고 objects 가 있습니다. Lifelines 은 객체의 생명 주기를 나타내며, 객체가 생성되거나 소멸되는 시점을 보여줍니다. Activation boxes 는 객체들이 상호작용하고 활성화되는 상태를 나타냅니다. Messages 는 객체들 간에 교환되는 데이터를 나타내며, 주로 요청과 응답으로 이루어집니다. Objects 는 시나리오의 기능을 수행하는 데 필요한 클래스의 객체들을 말합니다.

2.2.5 Class Diagram

Class diagram 은 시스템의 구조를 표현하는 정적인 다이어그램으로, 클래스들의 구성, 속성, 기능, 동작 방식, 그리고 클래스들 간의 관계를 보여줍니다. 이는 시스템의 전반적인 구조와 클래스들 간의 의존성을 이해하는 데 도움이 됩니다. 또한, 클래스 간의 정적인 관계를 정의하여 시스템을 분석하고 설계하는 일관된 방법을 제공하며, 객체지향에 가장 가까운 형식으로 시스템을 표현합니다.

2.3 Project Scope

이 문서에서 개발 중인 서비스는 사용자들이 자신의 Java 소스 코드의 탄소 배출량을 확인하고, 그 결과를 통해 탄소 배출량을 줄이기 위한 코드 최적화를

진행하는 데 도움을 주기 위해 제작되었습니다. 이 서비스를 사용하는 사람들, 개발자를 포함한 모든 사용자들이 자신의 소스 코드의 탄소 배출 수준을 확인하고, 프로젝트를 개발하는 동안 지속적으로 코드를 최적화하면서 탄소 배출량을 고려할 수 있도록 하고자 합니다.

2.4 References

https://github.com/skkuse/2023fall_41class_team2
https://github.com/skkuse/2023fall_41class_team4

3

System Architecture - Overall

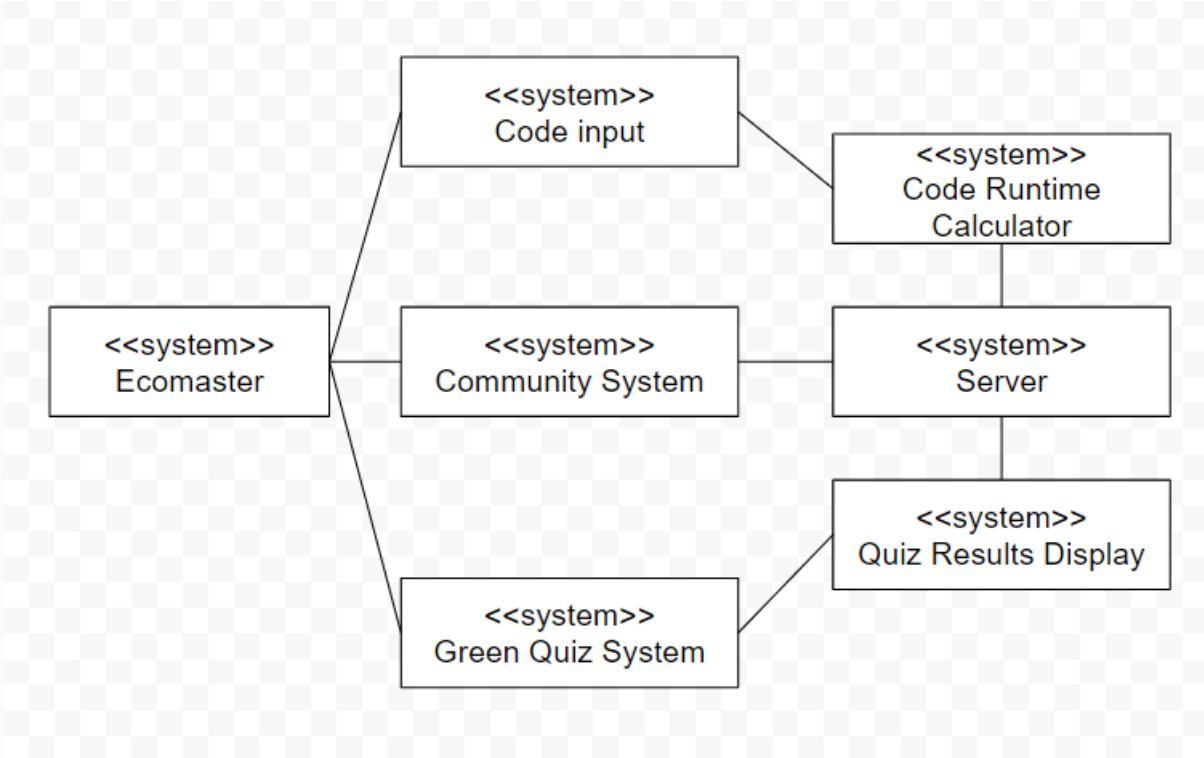
3.1 Objectives

이 챕터에서는 시스템 내부의 구조가 대략적으로 어떻게 설계, 구성되어 있는지에 대해 설명합니다.

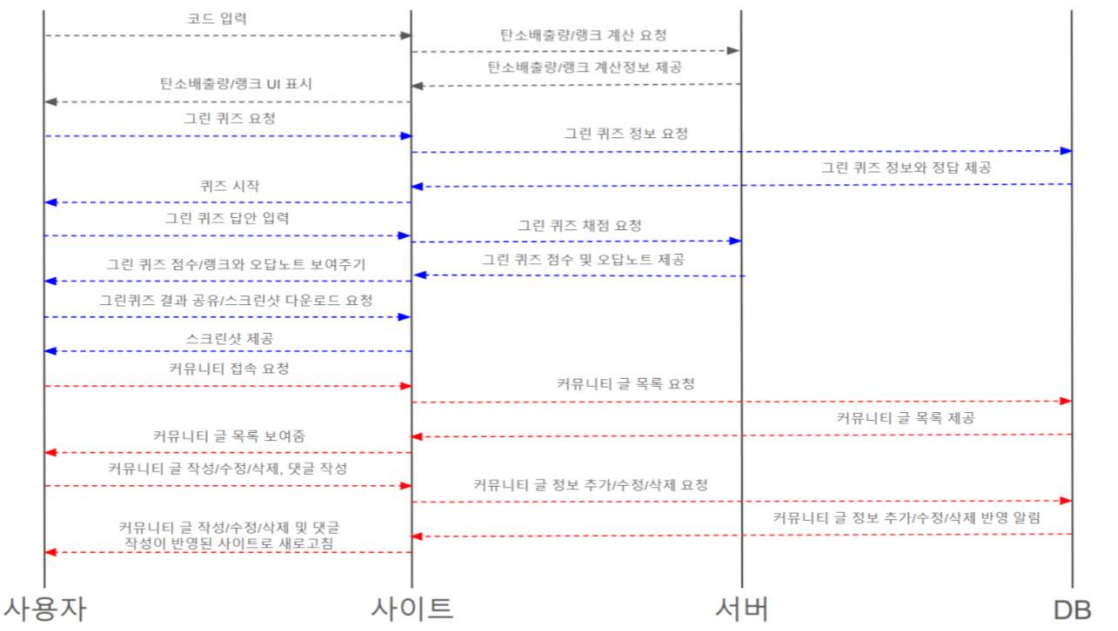
3.2 System Organization

이 서비스는 client-server 모델을 채택하여 구축되었습니다. Frontend 애플리케이션은 사용자와의 모든 상호작용을 담당합니다. Frontend 및 Backend 애플리케이션은 JSON 기반의 HTTP 통신을 이용하여 데이터를 주고받습니다. Backend 애플리케이션은 사용자로부터 자바 코드 및 하드웨어 스펙 요청을 받습니다. 또한, 설계 사양 요청을 프론트 엔드 컨트롤러로부터 받아 처리하고, JSON 형식으로 응답합니다. Backend 애플리케이션은 사용자가 작성한 JAVA 코드를 수신하고 실행한 후 탄소 배추량을 반환합니다. 결과를 반환하는 동시에, 그린 패턴을 이용해 코드를 개선한 후, 사용자에게 제공합니다.

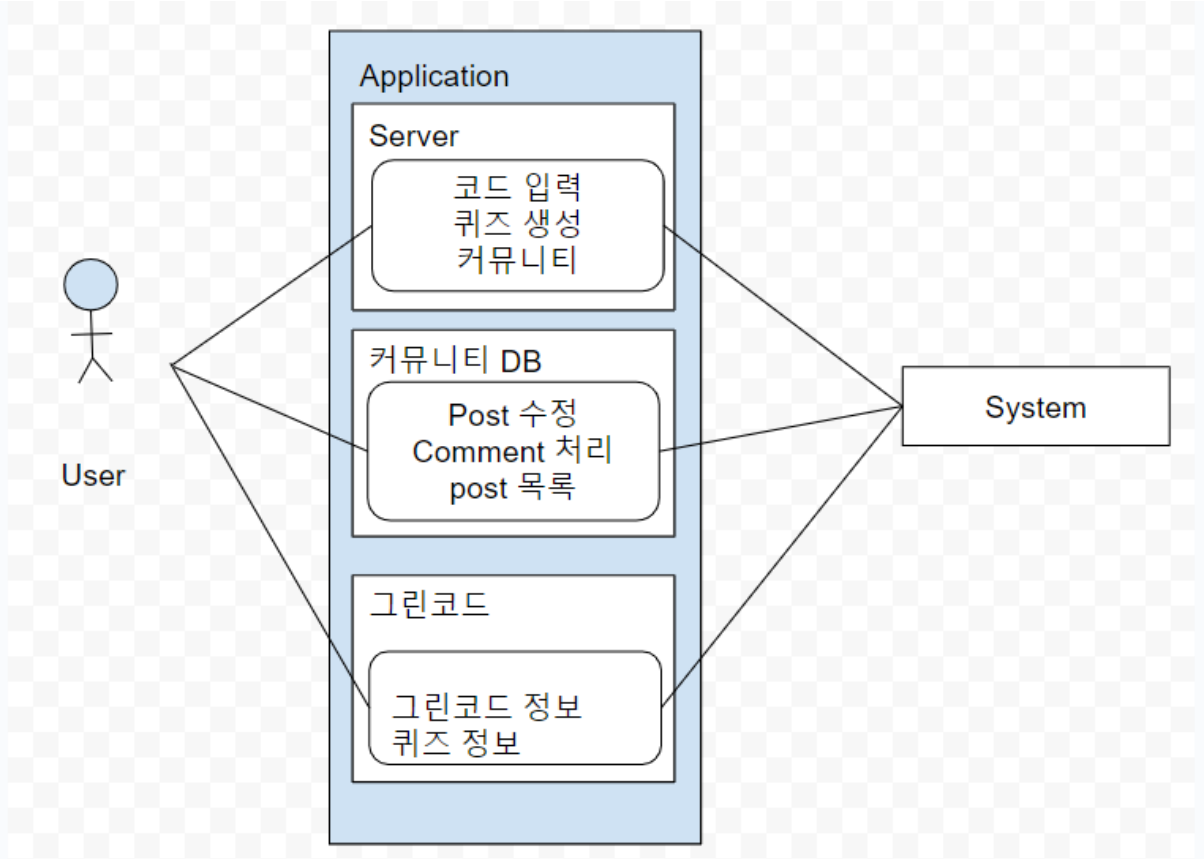
3.2.1 Context Diagram



3.2.2 Sequence Diagram



3.2.3 Use case Diagram



4

System Architecture - Frontend

4.1 Objectives

이 장에서는 프론트엔드 시스템의 구조, 속성 및 기능을 설명하고 탄소배출량 계산기에서 각 구성 요소의 관계, 커뮤니티 존과 그린패턴 퀴즈 존의 구성 요소의 관계 역시 들어있습니다.

4.2 Subcomponents

본 소프트웨어의 프론트엔드는 원활한 사용자 경험을 제공하기 위해 여러 주요 구성 요소로 구성됩니다. 이 구성 요소에는 코드 입력 및 분석 모듈, 커뮤니티 존, 그린 패턴 퀴즈 존이 포함됩니다. 각 구성 요소는 특정 속성과 메서드를 갖춘 하위 구성 요소로 세분화됩니다.

4.2.1 Code Input and Analysis

이 모듈은 주요 페이지에서 사용자가 Java 소스 코드를 입력하고 분석하는 영역입니다. 사용자는 여기서 소스 코드를 입력한 뒤, 제출 버튼을 통해 데이터를 백엔드로 전송하여 탄소 배출량을 분석할 수 있습니다. 분석 결과는 시각적으로 표시되며, 탄소 배출량을 줄이기 위한 제안도 함께 제공됩니다.

4.2.1.1 Attribute

codeInputField: 사용자가 소스 코드를 입력할 수 있는 텍스트 영역.

- 유형: HTMLTextAreaElement
- 설명: 사용자의 소스 코드를 캡처합니다.

analyzeButton: 입력된 코드 분석을 시작하는 버튼.

- 유형: HTMLButtonElement

- 설명: 클릭 시 코드 분석 과정을 시작합니다.

carbonFootprintDisplay: 계산된 탄소 배출량을 표시하는 요소.

- 유형: HTMLDivElement
- 설명: 사용자에게 탄소 배출 결과를 알기 쉽게 보여줍니다.

suggestionsPanel: 탄소 배출량을 줄이기 위한 제안을 표시하는 패널.

- 유형: HTMLDivElement
- 설명: 코드를 더 환경 친화적으로 만들기 위한 팁과 모범 사례를 제공합니다.

4.2.1.2 Methods

captureCodeInput(): 텍스트 영역에 입력된 코드를 캡처합니다.

- 반환 값: 문자열 (소스 코드)

sendCodeForAnalysis(): 캡처된 코드를 백엔드로 보내 탄소 발자국 분석을 수행합니다.

- 매개변수: code (String)
- 반환 값: Promise (분석 결과가 포함된 JSON 응답)

displayAnalysisResults(): 분석 결과를 구문 분석하고 carbonFootprintDisplay 를 업데이트합니다.

- 매개변수: analysisResults (Object)
- 업데이트: 탄소 배출량 데이터를 포함한 carbonFootprintDisplay

showReductionSuggestions(): 탄소 배출량을 줄이기 위한 제안을 검색하고 표시합니다.

- 매개변수: suggestions (Array<Object>)
- 업데이트: 실행 가능한 조언을 포함한 suggestionsPanel

4.2.2 Community Zone

커뮤니티 존은 사용자들이 소스 코드의 탄소 배출량에 대해 토론하고 정보를 공유할 수 있는 공간이다. 이곳에서 사용자들은 경험을 공유하고, 질문을 하고, 답변을 찾을 수 있습니다. 이를 통해 사용자 간의 상호작용과 지식 공유가 이루어집니다.

4.2.2.1 Attribute

discussionBoard: 사용자 게시물과 토론을 표시하는 요소.

- 유형: HTMLDivElement
- 설명: 탄소 배출량 관련 사용자 생성 콘텐츠와 토론을 호스팅합니다.

postInputField: 사용자가 새 게시물을 작성할 수 있는 텍스트 영역.

- 유형: HTMLTextAreaElement
- 설명: 사용자가 새 토론 주제나 댓글을 입력할 수 있습니다.

postButton: 새 게시물 또는 댓글을 제출하는 버튼.

- 유형: HTMLButtonElement
- 설명: 사용자 생성 콘텐츠를 토론 게시판에 제출합니다.

userProfiles: 사용자 프로필과 기여도를 표시하는 요소.

- 유형: HTMLDivElement
- 설명: 활발한 사용자와 그들의 게시물을 보여주어 커뮤니티 참여를 촉진합니다.

4.2.2.2 Methods

fetchDiscussionPosts(): 서버에서 게시물을 가져와 discussionBoard 에 표시합니다.

- 반환 값: Promise (Array<Post>)

submitNewPost(): 사용자가 작성한 새 게시물을 서버로 보냅니다.

- 매개변수: postContent (String)
- 반환 값: Promise (Ack)

displayUserProfiles(): 사용자 프로필과 기여도를 가져와 표시합니다.

- 반환 값: Promise (Array<Profile>)

- 업데이트: 프로필 데이터를 포함한 userProfiles

4.2.3 Green Pattern Quiz Zone

그린 패턴 퀴즈 존은 사용자가 환경 친화적인 코딩 패턴을 학습하고 퀴즈를 통해 지식을 테스트할 수 있는 공간입니다. 퀴즈를 통해 지속 가능한 코딩 실습에 대한 이해를 높이는 것이 목적입니다.

4.2.3.1 Attribute

quizQuestionDisplay: 퀴즈 질문을 표시하는 요소.

- 유형: HTMLDivElement
- 설명: 사용자에게 현재 퀴즈 질문을 보여줍니다.

answerOptions: 퀴즈 답변 옵션을 표시하는 요소.

- 유형: HTMLButtonElement[]
- 설명: 사용자가 답변을 선택할 수 있는 버튼을 제공합니다.

quizProgressTracker: 사용자의 퀴즈 진행 상황을 추적하고 표시하는 요소.

- 유형: HTMLDivElement
- 설명: 사용자의 진행 상황을 시각적으로 나타냅니다.

feedbackDisplay: 각 퀴즈 질문 후 피드백을 표시하는 요소.

- 유형: HTMLDivElement
- 설명: 사용자의 답변에 대한 즉각적인 피드백을 제공합니다.

4.2.3.2 Methods

fetchQuizQuestions(): 서버에서 퀴즈 질문을 가져옵니다.

- 반환 값: Promise (Array<String>)

displayNextQuestion(): quizQuestionDisplay 에 다음 질문을 업데이트합니다.

- 매개변수: question (Object)
- 업데이트: quizQuestionDisplay와 answerOptions

checkAnswer(): 사용자의 답변을 검증하고 피드백을 제공합니다.

- 매개변수: selectedAnswer (String), correctAnswer (String)
- 업데이트: 검증 결과를 포함한 feedbackDisplay

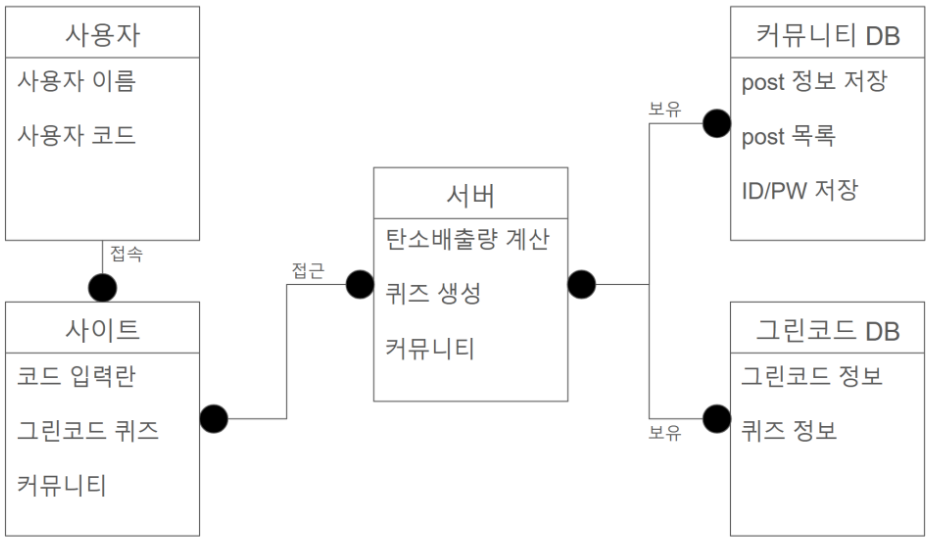
updateProgressTracker(): 사용자의 진행 상황에 따라 quizProgressTracker 를 업데이트합니다.

- 매개변수: currentProgress (Int)
- 업데이트: 현재 진행 상황 백분율을 포함한 quizProgressTracker

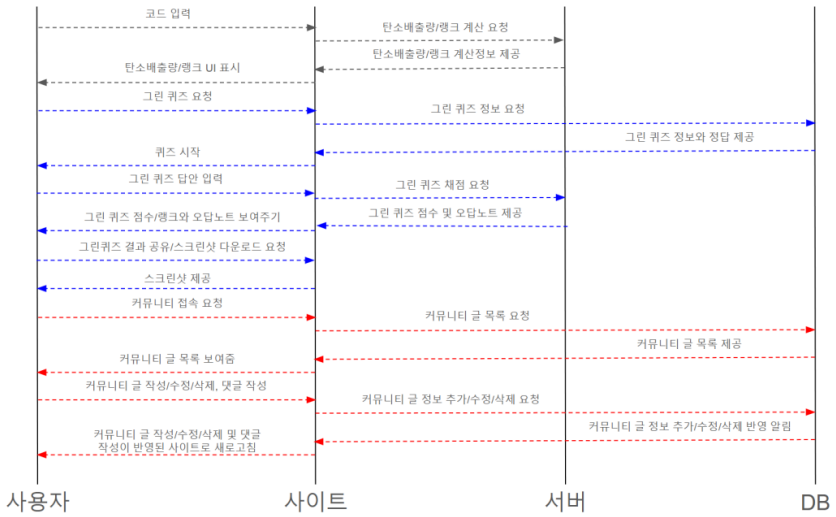
4.2.4 Diagram

이 항목에서는 프론트엔드 시스템의 설계를 명확히 이해하고, 각 구성 요소의 역할 및 상호 작용을 시각적으로 표현하기 위해 다양한 다이어그램을 제공합니다. Code Input and Analysis, Community Zone, Green Pattern Quiz Zone 에 대해 설명하는 다이어그램입니다.

4.2.4.1 Attributes and relationships

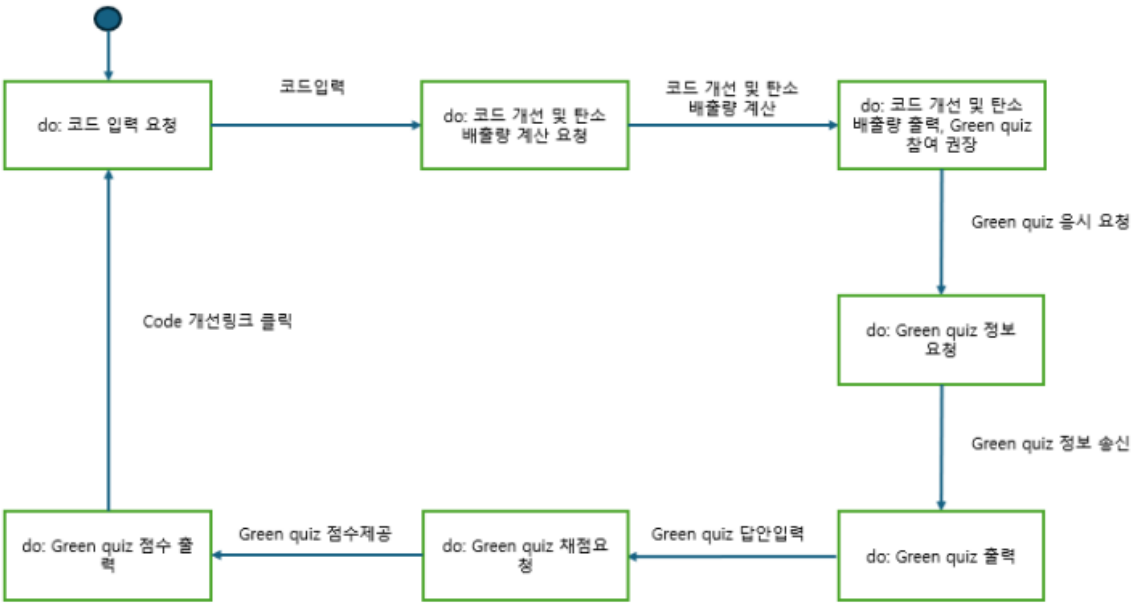


4.2.4.2 Sequence Diagram

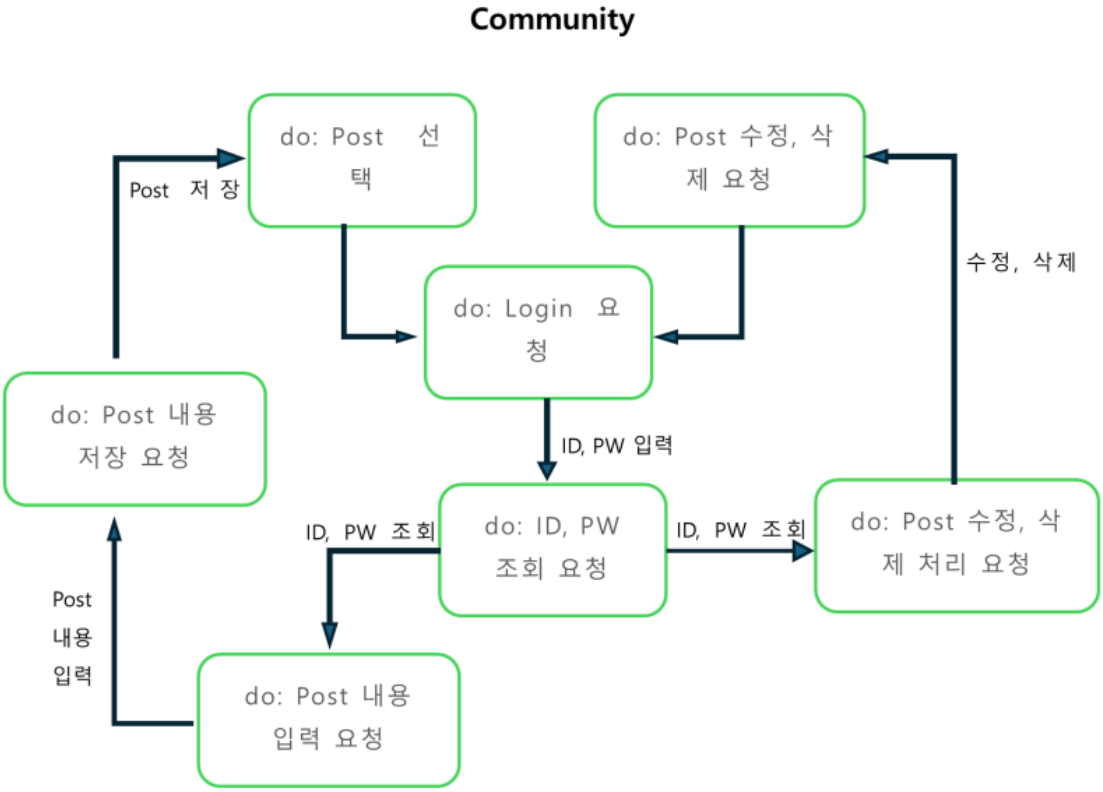


4.2.4.3 State Diagram

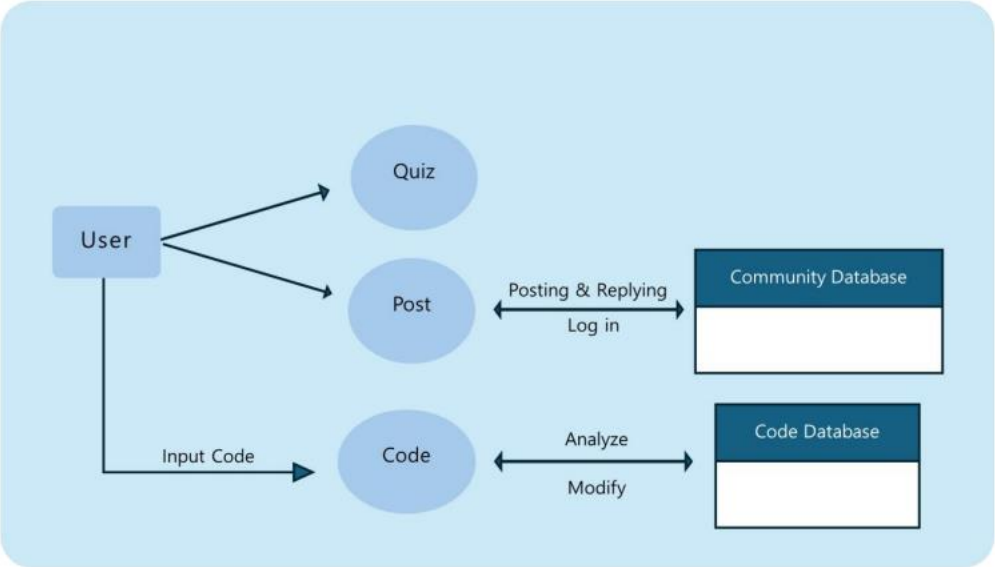
4.2.4.3.1 Code Input and Analysis



4.2.4.3.2 Community Zone



4.2.4.4 Data Flow Diagram



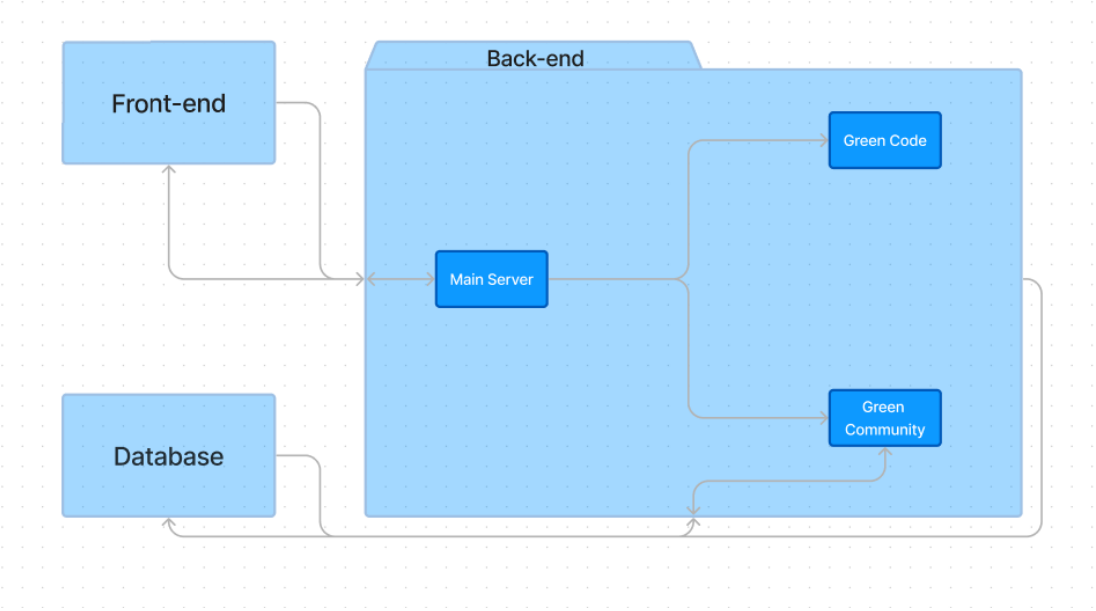
5

System Architecture - Backend

5.1 Objectives

이 장에서는 Back-end 시스템의 구조와 기능에 대해서 기술합니다.

5.2 Overall Architecture

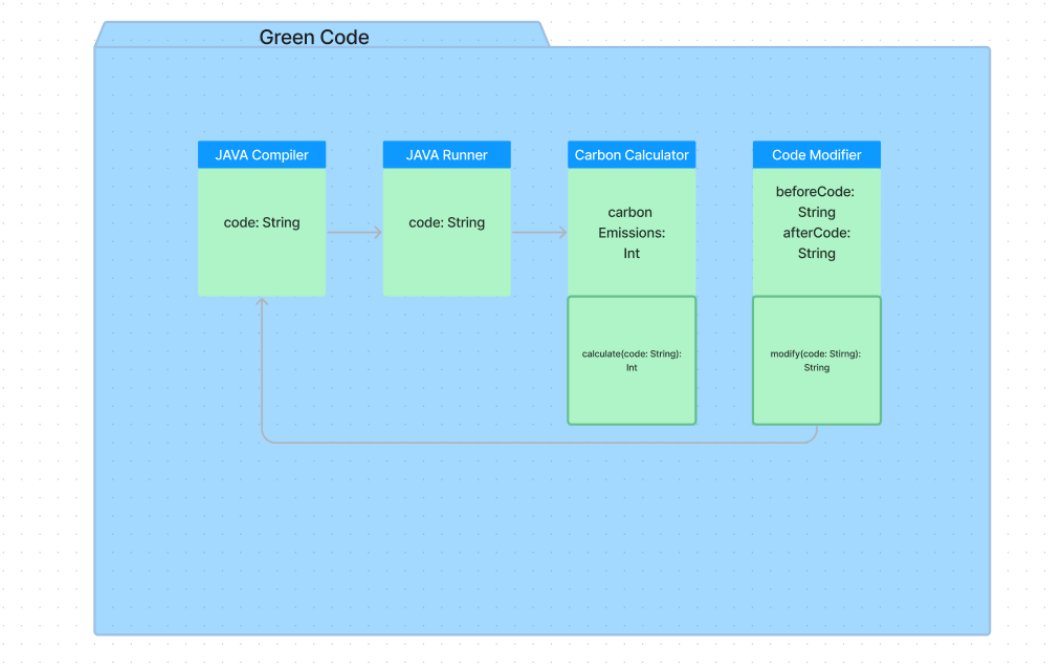


사용자에게 Green Pattern 에 대해 알리고, 개선된 코드를 제시하는 시스템의 전반적인 구조는 위와 같습니다. Front-end 로부터 요청이 발생하면 Main Server 는 요청의 종류에 따라 하위 시스템에서 요청을 처리하도록 합니다. 처리하는 요청의 종류는 먼저 코드의 탄소 배출량 측정 및 Green 화, Community 이용 등이 있습니다.

5.3 Sub systems

5.3.1 Green Code

Class Diagram



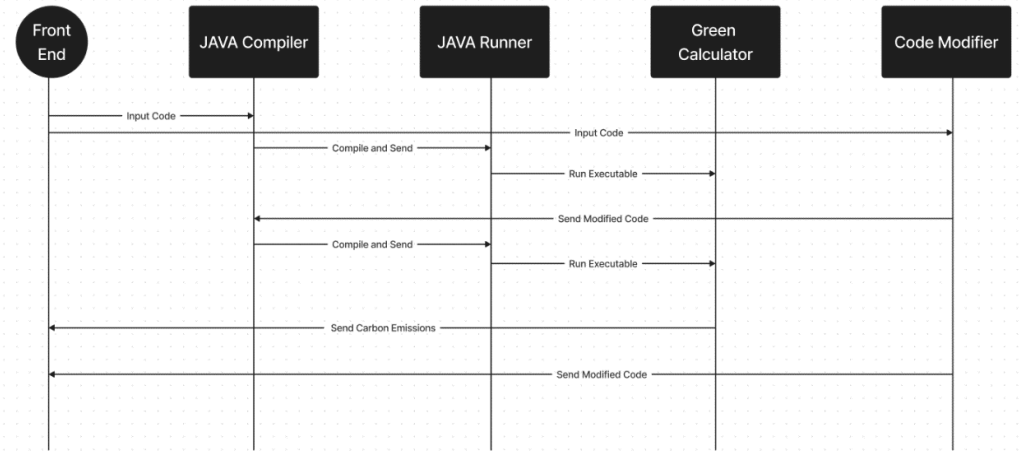
JAVA Compiler: 사용자가 입력한 JAVA Code 를 Compile 합니다.

JAVA Runner: Compile 한 결과물을 실행합니다.

Carbon Calculator: 컴파일하고 실행한 과정에서 발생한 탄소배출량을 계산합니다.

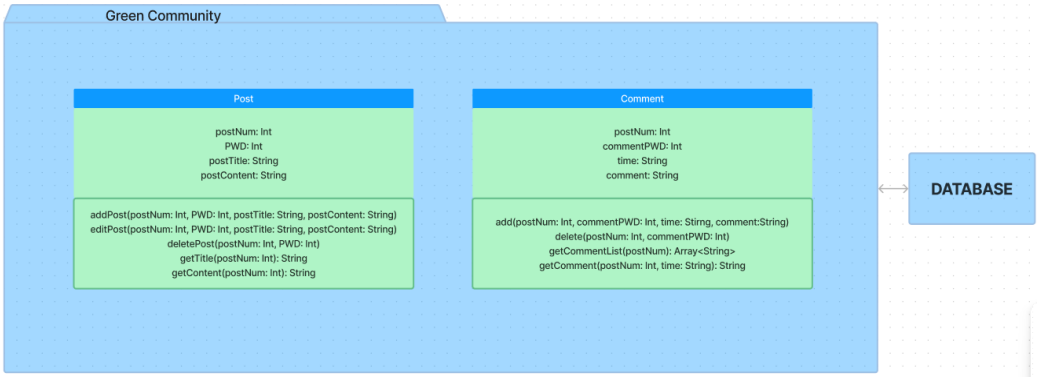
Code Modifier: 사용자가 입력한 Code 를 Green Pattern 을 이용해 개선합니다. 개선 후 Compiler 에 제공해 탄소 배출량을 측정합니다.

Sequence Diagram



5.3.2 Green Community

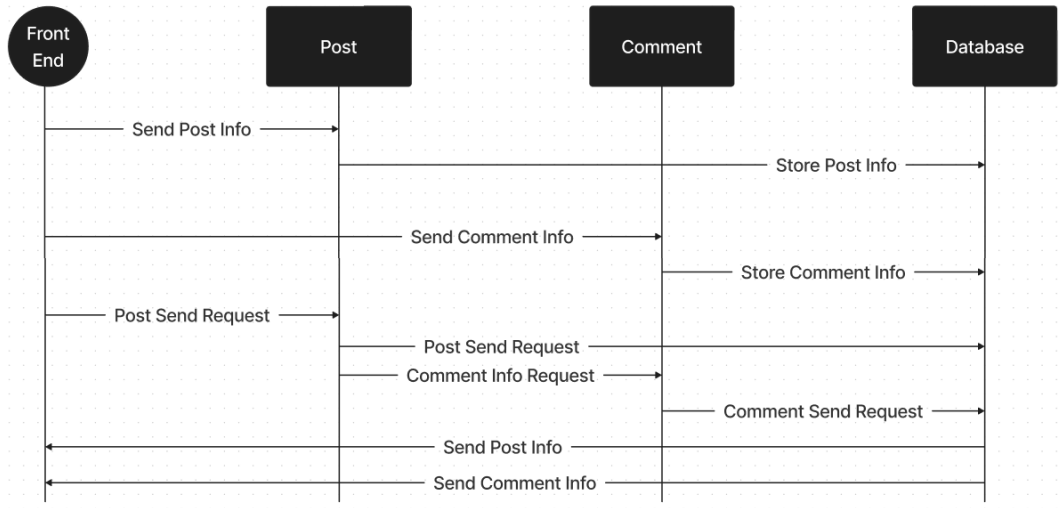
Class Diagram



Post: User 가 Community 에 글을 등록하면 글에 대한 정보를 Database 에 저장합니다. 저장된 글을 불러오는 역할도 합니다.

Comment: User 가 글에 Comment 를 달면 Database 에 Comment 에 대한 정보를 저장합니다. 각 Post 에 저장된 Comment 를 불러오는 역할도 합니다.

Sequence Diagram



6

Protocol Design

6.1 Objectives

이 장에서는 front-end 와 back-end 가 상호작용하는 방법과 과정에 대해 설명합니다. 시스템에서 사용되는 protocol, API 에 대한 설명과 기능에 따른 HTTP Methods 를 제공합니다.

6.2 Protocol Details

6.2.1 HTTP

HTTP (HyperText Transfer Protocol)는 브라우저와 서버 사이의 상호 작용을 정의하며, HTML 문서와 같은 데이터를 주고 받기 위해 사용되는 프로토콜입니다. 클라이언트(브라우저)가 Request 를 생성하면 서버가 이를 처리하고 Response 를 반환합니다. Request 는 Method, URL, Body 가 포함되며, Response 는 HTTP Status Code, Body 를 포함하는 메시지를 의미합니다.

6.2.2 REST API

REST (Representational State Transfer) API 는 HTTP 를 기반으로 웹 서비스에서 클라이언트와 서버 사이의 상호 작용을 정의합니다. REST API 는 Stateless 프로토콜을 따르기 때문에 클라이언트는 모든 정보를 Request 에 포함해야 하며,

Request 를 받은 서버는 JSON 형식으로 Response 를 반환하고 HTTP 상태 코드로 Request 의 결과를 알립니다.

6.2.3 Cloud Firebase API

Firebase 는 클라우드 기반의 애플리케이션 개발 플랫폼으로, 백엔드 서비스, 데이터베이스, 인증 등의 다양한 API 를 제공합니다. 본 서비스에서는 커뮤니티 post 를 저장하고 관리하는 목적으로 Cloud Firebase API 를 사용하고 있습니다. Cloud Firebase API 는 확장 가능한 NoSQL 클라우드 데이터베이스이며, document 와 collection 구조를 사용해 데이터를 저장합니다.

6.3 HTTP Methods

6.3.1 Code Input and Analysis

6.3.1.1 Request

Attribute	**Descriptions**	
Method	POST	
URL	/calculate	
Body	code	사용자로부터 입력된 code (string)

6.3.1.2 Response

Attribute	**Descriptions**	
Status Code	200 OK	
Error Code	400 Bad Request	
Body	carbon_emissions	계산된 탄소 배출량 (int)
	Optimal_carbon_emissions	최적 탄소 배출량 (int)
	rank	최적 탄소 배출량 대비 유저 탄소 배출량 (string)

6.3.2 Green Quiz

6.3.2.1 Request

Attribute	**Descriptions**	
Method	POST	
Path	/greenquiz	
Body	quiz_problem	Green quiz 의 문제 리스트 (string)

	quiz_answer	Green quiz 의 정답 리스트 (int)
	user_answer	유저가 입력한 정답 리스트 (int)

6.3.2.2 Response

Attribute	**Descriptions**	
Status Code	200 OK	
Error Code	400 Bad Request	
Body	user_quiz_score	유저의 퀴즈 점수 (int)
	quiz_message	퀴즈 완료 후 출력하는 message (string)

6.3.3 Community

6.3.3.1 Request

Attribute	**Descriptions**	
Method	POST	
Path	/community	
Body	post_title	게시글 제목 리스트 (string)
	uid	게시글의 uid (string)
	password	게시글 수정/삭제를 위한 password (string)

6.3.3.2 Response

Attribute	**Descriptions**	
Status Code	200 OK	
Error Code	400 Bad Request	
Body	post	게시글의 내용 (string)
	post_replies	게시글의 댓글 리스트 (string)

7

Database Design

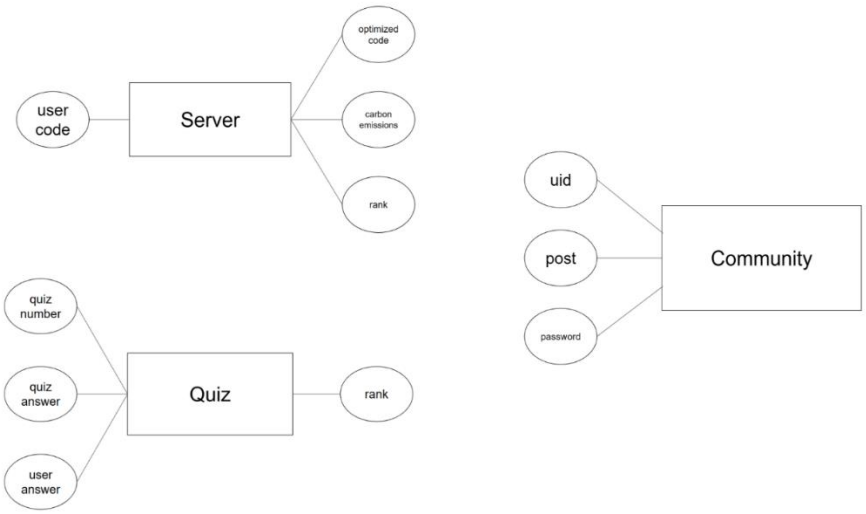
7.1. Objectives

7 장에서는 시스템 데이터 구조와 이러한 구조가 데이터베이스로 어떻게 구현되어 있는지 설명합니다. 먼저 ER 다이어그램(Entity Relationship diagram)을 통해 entity 와 그 관계를 식별한 후, 관계형 schema 및 SQL DDL(Data Definition Language)을 작성합니다.

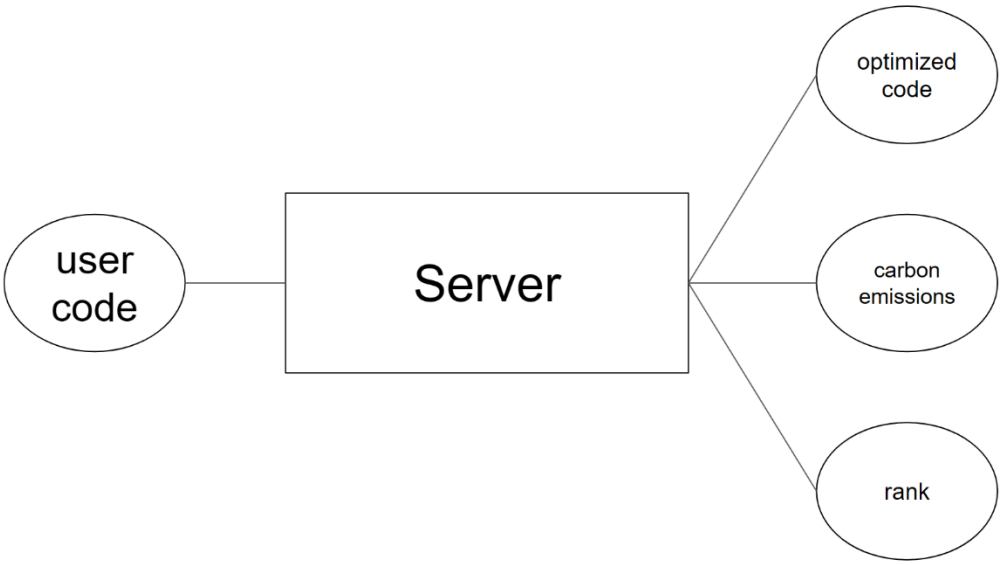
7.2. ER Diagram

본 어플리케이션 시스템은 총 3 가지 entity 로 이루어져 있습니다: Server, Quiz, Community.

ER-Diagram 은 entity 간의 관계, 그리고 entity 와 attribute 의 관계를 다이어그램으로 설명합니다. 각 entity 의 primary key 는 밑줄로 표시되어 있다. 각 entity 마다 대응되는 개수는 entity 를 연결하는 선 주변에 표기되어 있어 확인 가능하다.

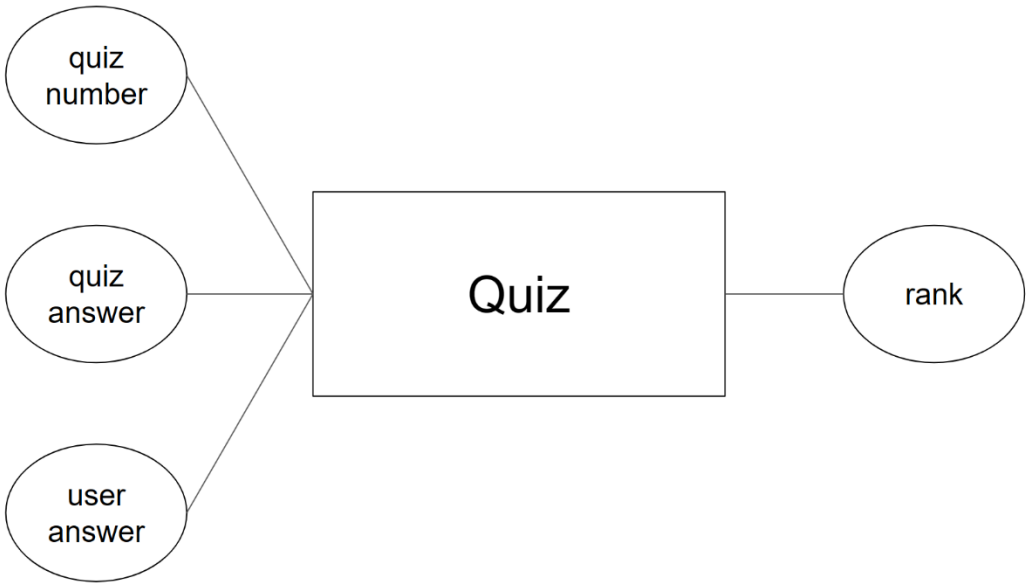


7.2.1. Server



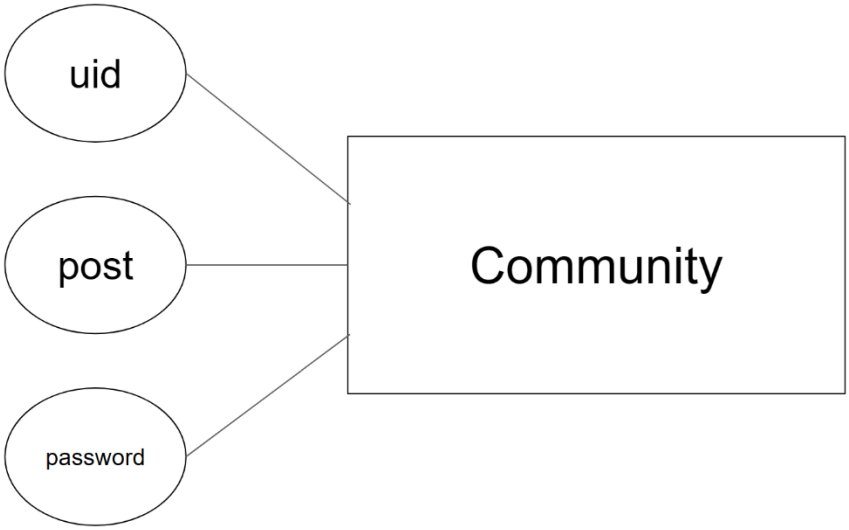
Server Entity 는 사용자에게 코드를 입력받은 후, 탄소배출량을 계산하고 최적화된 코드를 제공합니다. 또한 계산된 탄소배출량에 따라 랭크를 부여합니다.

7.2.2. Quiz



Quiz Entity 는 사용자에게 제시할 퀴즈를 저장합니다. 퀴즈 넘버와 유저가 입력한 퀴즈 정답, 그리고 실제 퀴즈 정답을 입력으로 받습니다. 이후 계산을 거쳐 유저의 랭크를 매깁니다.

7.2.3. Community



Community Entity 에서는 게시글의 uid 와 post(내용), 그리고 게시글의 수정/삭제 권한을 가지는 데 필요한 비밀번호를 데이터베이스에 저장합니다.

7.3. Schema Figure

Quiz	
PK	quiznumber
	quiz quizanswer useranswer rank

Community	
PK	uid
	post password

Server	
PK	carbon
	usercode opcode rank

7.4. SQL DDL

7.4.1. Server

```
CREATE TABLE Server(
      usercode VARCHAR(2000),
      opcode VARCHAR(2000),
      carbon INTEGER NOT NULL,
      rank VARCHAR(10),
      PRIMARY KEY(carbon)
)
```

7.4.2. Quiz

```
CREATE TABLE Quiz(
      quiznumber INTEGER NOT NULL,
      quiz VARCHAR(200) NOT NULL,
      quizanswer VARCHAR(100) NOT NULL,
      useranswer VARCHAR(1000) NOT NULL,
      rank VARCHAR(10) NOT NULL,
      PRIMARY KEY(quiznumber)
)
```

7.4.3. Community

```
CREATE TABLE Community(
      uid INTEGER NOT NULL,
      post VARCHAR(200) NOT NULL,
      password VARCHAR(20) NOT NULL,
      PRIMARY KEY(password)
)
```

8

Testing Plan

8.1. Objectives

이번 장에서는 development testing, release testing, 그리고 user testing 3 가지 주요 하위 그룹을 포함하는 테스트에 대한 계획을 설명합니다. 이 테스트들 시행하는 목적은 어플리케이션의 잠재적 오류와 결함을 미리 찾아내고, 어플리케이션의 완성도를 높여 안정적인 시스템을 사용자에게 제공하기 위함입니다.

8.2. Testing Policy

8.2.1. Development Testing

Development Testing 은 현재 시스템 개발 과정 전체에서 발생할 수 있는 오류를 최대한 많이 감지하고, 이 과정에서 감지된 오류를 수정하기 위한 testing 입니다. 소프트웨어가 계속해서 개발하는 중인 단계에서 진행하기 때문에 다소 불안정할 수 있고, frontend 와 backend (server 와 website)의 충돌이 발생할 수도 있습니다. 따라서 이 단계에서는 dataflow 검토 및 개발 팀원들 간의 code review 등을 계획하며, 이를 통해 소프트웨어의 성능과 신뢰성, 보안을 보장할 수 있도록 초점을 맞추어 테스트를 진행해야 합니다.

Performance

본 시스템의 performance 에 관한 지표는 다음과 같습니다:

- 그린코드 교육용으로 사용되는 본 시스템의 특성상 사용자의 코드에 대한 탄소배출량 계산에 소모되는 시간이 3 분 이내로 완료되어야 합니다.

- 탄소배출량 계산 결과는 5 초 이내로 데이터베이스에 저장되어야 하며, 이를 알 수 있어야 합니다.
- 사용자의 퀴즈 채점 결과를 5 초 이내로 계산해내어 사용자가 알 수 있도록 해야 합니다.
- 사용자가 커뮤니티에 게시글을 작성했을 때, 그 정보들을 5 초 이내로 데이터베이스에 저장하고 이를 사용자에게 알려줄 수 있어야 합니다.
- 10000 개 까지의 커뮤니티 게시글 정보를 관리할 수 있어야 합니다. 따라서 다음과 같은 지표를 만족할 수 있도록 테스트 과정에서 uid 와 게시글 내용, 그리고 비밀번호가 다양한 길이로 저장된 경우를 준비하여 테스트합니다.

  위에서 제시한 기준을 만족할 수 있도록 실제 테스트를 다양한 환경에서 여러 번 수행하여 소요시간이 위에서 제시한 기준을 만족했는지 확인합니다.

Reliability(신뢰성)

시스템이 오류를 발생하지 않고 안전하고 안정적으로 작동하려면 시스템을 구성하는 각각의 하위 구성 시스템과 장치가 정상적으로 작동한 뒤, 전체 시스템으로 안정적으로 연결되어야 합니다. 따라서 하위 시스템 개발 단계부터 개발 테스트를 순차적으로 거쳐 각 시스템이 전체 시스템에 통합되는 동안 오류가 발생하는지 반복적으로 확인하고 그에 따라 수정하도록 하겠습니다.

Security

본 시스템에서는 개인정보를 따로 요구하고 있지 않기 때문에 금융 관련 어플리케이션만큼의 높은 보안 수준은 요구되지 않습니다. 다만 사용자들이 설정한 게시글 비밀번호가 유출되지 않을 정도의 보안을 구축해야 할 것입니다. 사용자들은 자신이 게시글에 설정한 비밀번호가 다른 사용자에게 유출되지 않도록 하고, 외부 시스템에 의한 DB 접근이 없도록 막아야 합니다.

8.2.2. Release Testing

Release Testing 은 어떻게 이 시스템을 배포할 지에 대한 것을 테스트하는 단계입니다. Release testing 은 소프트웨어와 어플리케이션이 버전을 업데이트할 때마다 테스트하여 소프트웨어가 완전한 상태로 배포될 수 있도록 해야 합니다. 아무런 하자가 없는지 실제로 배포하기 전 수행되어야 합니다.

이 소프트웨어의 배포는 일반적으로 모든 시스템의 기본 구현을 포함한 알파 버전부터 시작해, 개발 테스트를 시작하고 사용자 및 release test 를 포함한 추가 테스트를 위해 베타 버전 또한 배포할 예정입니다. Github 의 버전 추적을 통해 얻을 수 있으며, 버전 배포 후 사용자들로부터 피드백을 받을 수 있도록 할 것입니다.

8.2.3. User Testing

User Testing 에서는 실제 사용자가 이 소프트웨어를 사용한다는 가정 하에 테스트를 진행하게 됩니다. 우선 팀 내에서 테스트를 진행한 후 주변인에게도 배포하여 테스트를 진행할 예정입니다. 베타 버전을 배포해 여러 시나리오로 테스트하여 각각 오류가 발생하는지 아닌지 검토하는 방식으로 진행될 예정입니다.

8.2.4. Testing Case

Testing case 는 앞서 살펴본 performance, reliability, security 에 초점을 맞추어 설계됩니다. 각각의 측면에서 최소 3 개 이상의 test case 와 시나리오를 통해 본 소프트웨어에 대한 테스트를 진행하며 평가 보고서를 작성할 것입니다.

9

Development Plan

9.1. Objectives

해당 챕터에서는 시스템의 개발 환경 및 기술에 대하여 설명합니다.

9.2. Frontend Environment

9.2.1. JavaScript Figure

JavaScript 는 객체 기반의 스크립트 프로그래밍 언어입니다. 주로 웹 브라우저 내에서 사용하며, 다른 응용 프로그램의 내장 객체에도 접근할 수 있는 기능을 가지고 있습니다. 본 소프트웨어에서는 사용자가 풀이하는 퀴즈와 탄소배출량 계산 등 관련 기능들을 제공하는 데 사용합니다.

9.2.2. HyperText Markup Language (HTML)

HTML 은 웹 페이지를 위한 마크업 언어로, 사이트 제작 시 흔하게 사용되는 언어 중 하나입니다. HTML 을 통해 제목, 단락, 목록 등 본문을 위한 구조적 의미를 나타내는 것뿐만 아니라 링크, 인용과 그 밖의 항목으로 구조적 문서를 만들 수 있는 방법을 제공받을 수 있습니다. 이는 태그로 되어있는 HTML 요소 형태로 작성되며 웹 브라우저와 같은 HTML 처리 장치의 행동에 영향을 주는 자바스크립트를 포함하거나 불러올 수 있습니다. 따라서 우리는 본 소프트웨어의 웹 페이지를 구현하기 위하여 HTML 을 사용합니다.

9.3. Backend Environment

9.3.1. Github

Github 는 소프트웨어를 개발하고 Git 을 통해 버전을 관리하기 위해 사용되는 툴입니다. 이를 통해 여러 명의 개발자가 하나의 프로젝트를 동시에 관리하며 개발할 수 있으며, 각각의 컴포넌트들을 통합하는데 이점이 있다. 또한 버전 관리가 용이하기 때문에 알파/베타 버전 배포에도 유리합니다. 이러한 이유로 저희 팀은 본 소프트웨어의 개발 및 버전관리를 위해 이를 사용할 것입니다.

9.3.2. FastAPI

FastAPI 는 파이썬 프레임워크로 빠른 성능을 보장해주며, 가볍게 웹개발을 할 때 자주 사용되는 것들 중 하나입니다. 사용법이 상당히 간단하고 성능 또한 우수하기 때문에 선정하였습니다.

9.3.3. Firebase

Firebase 는 데이터베이스 중 하나로, 계정 기능 사용이 용이하며 실시간 데이터베이스를 제공하고 저장소 또한 같이 일정 용량까지는 무료로 제공합니다. 이번 프로젝트로 대용량 저장소를 사용하거나 문서 생성/수정/삭제를 많이 할 예정은 없으며, 간단하게 사용하기도 용이하기 때문에 선정하였습니다.

9.4. Constraints

본 시스템은 이 문서에서 언급된 내용들에 기반하여 설계되고 구현될 것입니다. 이를 위한 세부적인 제약사항은 다음과 같이 나타냅니다.

- 기존에 널리 쓰이고 있는 기술 및 언어들을 사용합니다.
- 사용자의 입력 및 코드를 실행하고 결과를 저장하는 시간은 5초를 넘기면 안 됩니다.
- 로열티를 지불하거나 separate license를 요구하는 기술 및 software의 사용을 피합니다.
- 전반적인 시스템 성능을 향상시킬 수 있도록 개발합니다.
- 사용자가 편리하게 이용할 수 있도록 개발합니다.
- 가능한 오픈소스 소프트웨어를 최대한 사용합니다.
- 시스템 비용과 유지보수 비용을 고려하여 개발을 진행합니다.
- AI 사용 및 시스템 확장 가능성을 고려하여 개발을 진행합니다.
- 시스템 자원 낭비를 막을 수 있도록 소스 코드를 최적화합니다.
- 소스 코드 작성시 유지보수에 대하여 고려하며 필요한 부분에 대해서는 주석을 남김으로써 이해하기 쉽도록 노력합니다.
- 개발은 최소 윈도우 7 이상에서 이루어져야 하며 윈도우 10, 11 환경을 타겟으로 합니다.
- 윈도우 10, 11 버전에서 실행하는 것을 기준으로 개발을 진행합니다.

9.5. Assumptions and Dependencies

이 문서의 모든 시스템은 데스크탑 환경에 기반하여 설계 및 구현되었다고 가정하여 작성되었습니다. 또한 윈도우 10, 11 기반의 OS 환경을 기반으로 하여 작성되었기 때문에 다른 OS 나 조건을 만족하지 않는 환경에서 시스템의 지원은 보장할 수 없습니다.

10

Supporting Information

10.1. Software Design Specification

본 소프트웨어 디자인 명세서는 IEEE 권장 사항에 맞추어 작성되었습니다.

(IEEE Standard for Information Technology Systems Design Software Design Descriptions, IEEE-Std-830)

10.1. Software Design Specification

Date	Description	Version	Writer
2024/05/24	Purpose, Introduction, Overall Architecture	1.0	문수현
2024/05/24	Front-End Architecture	1.0	양승환
2024/05/24	Back-End Architecture	1.0	임성훈
2024/05/24	Protocol Design	1.0	김민성
2024/05/24	Database Design, Testing Plan, Development Plan	1.0	황인성