Software Design Specification

for

DevNavi: Web Coding Platform

2018314692 권민성
2018312018 서주원
2017313987 안서현
2016314729 안정민
2019312497 이장엽
2019314961 한수현

소프트웨어공학개론 3조

2023. 05. 21.

CONTENTS

Figures

Tables

1. INTRODUCTION

1.1. Objective

본 문서의 독자층을 정의하고, 문서를 구성하는 항목과 각 항목에 대해 기술한다.

1.2. Readership

본 문서의 주요 독자는 Team3의 구성원이며, 본 문서의 프로젝트가 진행되는 교과목 “소프트웨어공학개론”의 담당 교수자, 조교진을 독자로 상정한다. 본 문서는 독자가 개발진이 개발하고자 하는 시스템의 요구사항과 구성 요소를 이해하는 것을 돕기 위해 작성되었다.

1.3. Document Structure

1.3.1. Introduction

이 장에서는 시스템 설계에 사용한 다이어그램과 전반적인 시스템에 대해 기술하여 개발팀에서 목표로하는 프로젝트를 설명한다.

1.3.2. Overall System Architecture

이 장에서는 개발하는 시스템의 전반적인 아키텍처와 프론트엔드, 백엔드의 아키텍처를 다이어그램을 이용하여 기술한다.

1.3.3. System Architecture - Front-end

이 장에서는 프론트엔드 시스템의 구조, 속성 및 메소드에 대해 기술하고 다이어그램을 통해 시스템을 구성하는 각 요소들 간의 관계를 나타낸다.

1.3.4. System Architecture - Back-end

이 장에서는 백엔드 시스템의 구조를 다이어그램을 통해 제시하고 각 구성 요소들의 속성과 관계를 설명한다.

1.3.5. API Design

이 장에서는 서브 시스템 간의 상호작용을 위해 필요한 API에 대해 설명한다.

1.3.6. Database

이 장에서는 데이터베이스 디자인에 대한 전반적인 내용을 기술한다.

1.4. References

- "IEEE Standard for Information Technology--Systems Design--Software Design Descriptions," in IEEE STD 1016-2009, pp.1-35, 20 July 2009, doi: 10.1109/IEEESTD.2009.5167255.
- Team 2. “CodingCAT: 온라인 코딩 테스트 플랫폼”. SKKU, Last Modified: Dec. 12, 2022 https://github.com/skkuse/2022fall_41class_team2
- Team 6. “Web Coding Test”. SKKU, Last Modified: Dec. 12, 2022 https://github.com/skkuse/2022spring_41class_team6
- Team 10. “온라인 저지 플랫폼”. SKKU, Last Modified: Dec. 12, 2022 https://github.com/skkuse/2022spring_41class_team10

2. INTRODUCTION

2.1. Applied Diagram

본 문서에서는 State Diagram, Sequence Diagram, Class Diagram을 통해 개발하는 시스템에 대해 설명했다.

- State Diagram

State Diagram은 이벤트와 조건에 따른 상태의 변화를 나타내는 다이어그램이다. 프론트엔드 시스템에 대해 기술하는 부분에서 속성과 메소드들이 사용자의 동작에 따라 어떻게 작동하는지를 시각적으로 표현하기 위해 사용되었다.

- Sequence Diagram

Sequence Diagram은 시스템에서 오브젝트와 컴포넌트의 상호작용의 흐름을 나타내는 다이어그램이다. 백엔드 시스템에 대해 기술하는 부분에서 API, 데이터베이스 간의 상호작용을 시각적으로 설명하기 위해 사용되었다.

- Class Diagram

Class Diagram은 클래스의 구조와 클래스 간의 관계를 나타내는 다이어그램이다. 이 다이어그램을 이용하여 개발하는 시스템을 구성하는 클래스와 그 클래스 간의 관계를 나타내어 해당 시스템에 대한 이해를 도왔다.

2.2. Page Overview

“DevNavi” 시스템은 다양한 난이도의 코딩 문제를 제공함과 동시에 IT 기업의 신입사원 채용 과정에서 지원자 실력을 평가하는 코딩 테스트와 면접 시스템을 제공하여 개발 직무를 희망하는 취업 준비자들의 취업 준비 과정에 도움을 주는 서비스이다. 프로그래밍 학습과 코딩 테스트 연습이라는 본 서비스의 목적을 효과적으로 달성하기 위해 PS 모드와 코딩 테스트 모드로 나누어 서비스를 제공한다. PS 모드에서는 사용자가 원하는 문제를 선택하여 문제의 조건에 따라 코드를 작성하여 제출하면 작성된 코드에 대한 피드백이 제공된다. 코딩테스트 모드에서는 코딩 테스트에 대한 조건을

설정하면 설정한 조건에 따라 문제가 구성되고 사용자는 각 문제 조건에 따른 코드를 작성하게 된다. 사용자가 작성한 코드를 제출하면 코드에 대한 전반적인 피드백과 문제별 피드백이 제공된다. 이후 사용자는 면접 시뮬레이션을 통해 코딩 테스트에 대한 면접을 연습할 수 있다. 사용자에게 제공되는 피드백은 AI를 활용하여 다방면으로 코드를 평가하여 사용자의 실력을 효과적으로 향상시킬 수 있도록 했다.

사용자가 해당 시스템의 기능을 쉽게 활용할 수 있는 UI/UX를 설계했다. 서비스를 처음 접한 사람도 빠르게 시스템에 적응하고 사용할 수 있도록 하기 위해 단순화된 UI/UX를 구현하려 한다.

3. OVERALL SYSTEM ARCHITECTURE

3.1. System Organization

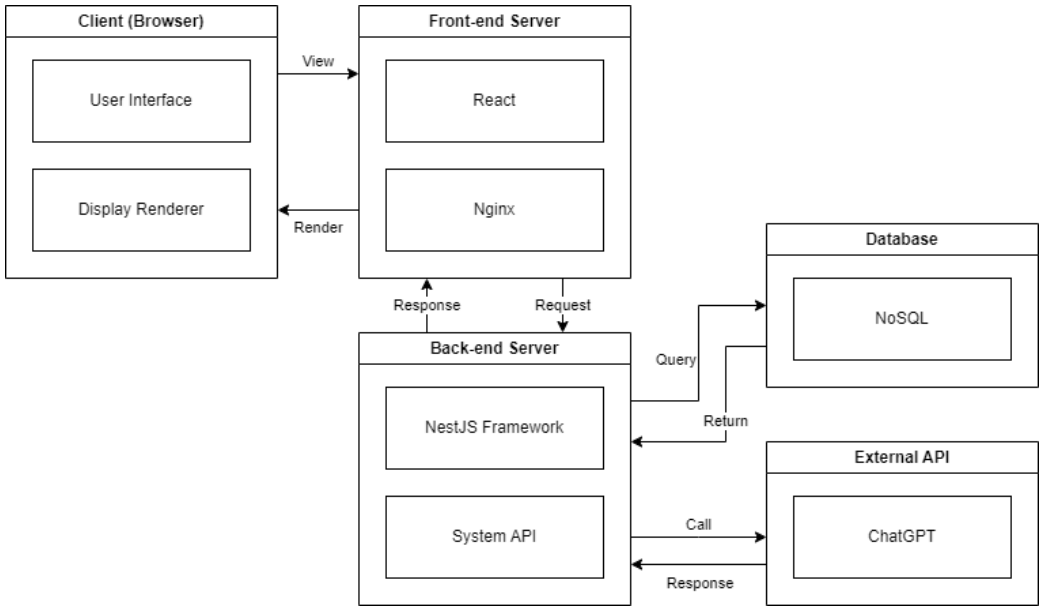

Figure 1. Overall System Architecture

본 시스템은 크게 클라이언트, 프론트엔드, 백엔드로 구성된다. 사용자는 Chrome과 같은 웹 브라우저를 통해 시스템에 접근하고, 프론트엔드 서버는 사용자의 요청을 백엔드 서버에 보낸 후 백엔드 서버에서 처리된 결과를 받아 사용자에게 표시하는 매개가 된다. 백엔드 서버는 요청된 내용을 시스템 내부 API나 데이터베이스 등 정해진 경로로 처리한다.

3.2. System Architecture - Front-end Application

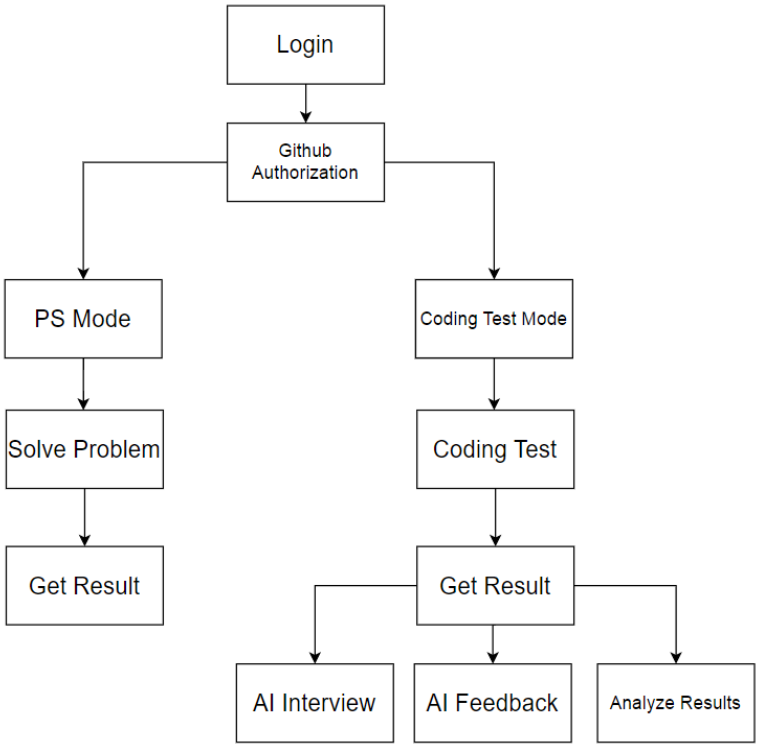

Figure 2. Overall Front-end Architecture

프론트엔드는 HTML, CSS, JavaScript로 화면을 구성하며 서버와 상호작용하며 동작한다. 사용자가 브라우저를 통해 PS Mode, Coding Test Mode 중에 선택하여 Solve Problem 혹은 Coding Test에 참여한다. 이에 대한 결과를 받아볼 수 있으며 Coding Test Mode의 경우, AI 면접과 피드백을 추가로 진행할 수 있다. 서버와 상호작용하며 데이터를 읽어오거나 전달하며 해당 내용을 프론트엔드의 레이아웃에 나타낸다.

3.3. System Architecture - Back-end Application

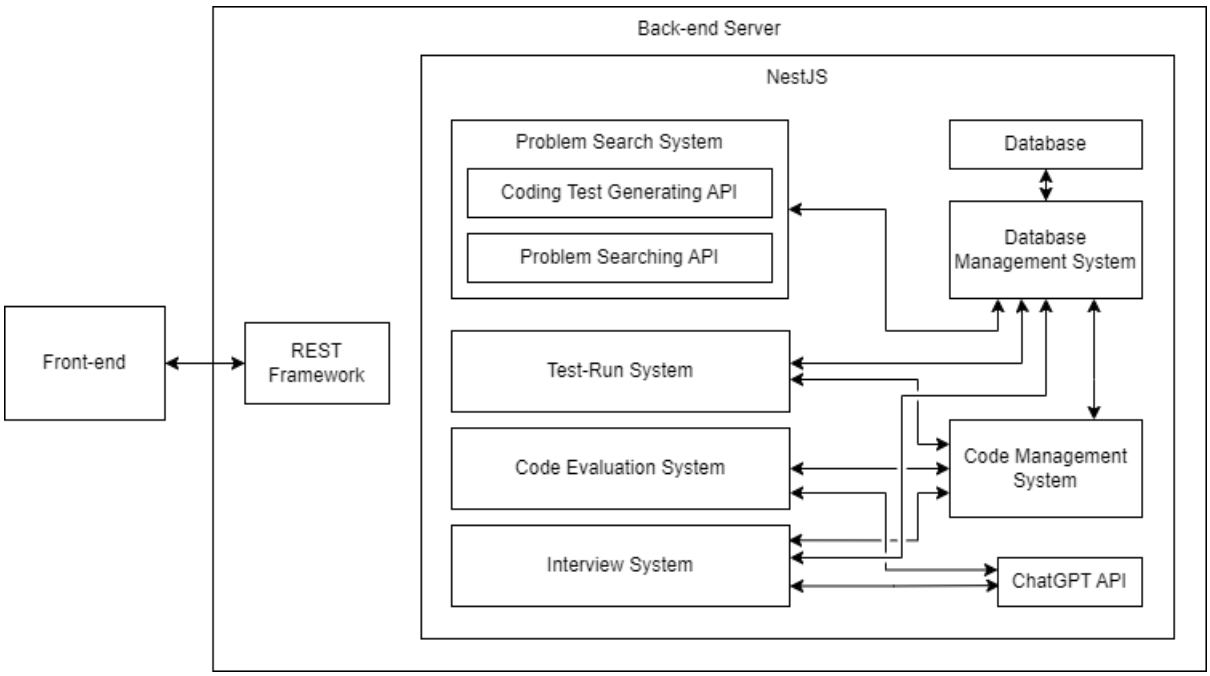

Figure 3. Overall Back-end Architecture

백엔드 서버의 시스템은 크게 다음과 같이 구성된다.

- Problem Search System

  PS에 사용할 문제를 찾거나 코딩 테스트를 생성하는 데에 필요한 시스템이다.

- Test-Run System

  PS 페이지에서 작성한 코드를 테스트케이스를 사용해 테스트하는 데에 필요한 시스템이다.

- Code Evaluation System

  코드의 효율성이나 가독성 등 코드의 품질을 평가하는 데에 필요한 시스템이다.

- Interview System

  작성한 코드나 프로그래밍 지식 등을 기반으로 면접 질문을 생성하거나 면접 과정에서 응답을 처리하는 데에 필요한 시스템이다.

4. SYSTEM ARCHITECTURE - FRONT-END

4.1. Objectives

이 장에서는 시스템을 구성하는 각 웹 페이지의 속성과 기능에 대해 설명하고, 사용자의 동작에 따른 페이지의 state를 diagram으로 표현하여 Front-end Architecture를 설명한다.

4.2. Main Page

DevNavi의 메인 페이지로, 학습자가 시스템에 접속해 로그인을 한 후에 도착하는 페이지이다. 메인 페이지에서는 문제풀이(PS) 모드와 코딩테스트 모드 중에서 선택할 수 있다. 코딩테스트 모드를 선택할 경우, 난이도, 문제 수를 선택할 수 있다.

4.2.1. Attributes

메인 페이지는 아래와 같이 구성되어 있다.

- MainPageBody: 사용자가 문제풀이 모드와 코딩테스트 모드 중에 선택할 수 있다.
- CodingTestModal: 사용자가 코딩테스트 모드 선택 시, 난이도, 문제 수를 선택할 수 있다.

4.2.1.1. Attributes of MainPageBody

- PSMode: 문제풀이 모드 선택 여부를 알려준다.
- CodingTestMode: 코딩테스트 모드 선택 여부를 알려준다.

4.2.1.2. Attributes of CodingTestModal

- Difficulty: 사용자가 설정하는 코딩테스트의 난이도이다.
- ProblemNumber: 사용자가 설정하는 코딩테스트의 문제 수이다.
- ExpectedTime: 코딩테스트의 예상 소요 시간이다.
- CodingTest: 코딩테스트 실행 여부를 알려준다.

4.2.2. Methods

메인 페이지의 메소드는 아래와 같이 구성되어 있다.

4.2.2.1. Methods of MainPageBody

- clickPSMode(): PSMode 버튼을 눌렀을 때 PS Select Page로 넘어간다.
- clickCodingTestMode(): CodingTestMode 버튼을 눌렀을 때 CodingTestModal을 띄운다.

4.2.2.2. Methods of CodingTestModal

- selectDifficulty(): 사용자가 원하는 Difficulty를 설정한다.
- selectProblemNumber(): 사용자가 원하는 ProblemNumber를 설정한다.
- showExpectedTime(): 사용자가 입력한 Difficulty와 ProblemNumber에 맞는 ExpectedTime을 계산해 표시한다.
- clickCodingTest(): CodingTest버튼을 눌렀을 때 CodingTestPage로 이동한다.

4.2.3. State Diagram

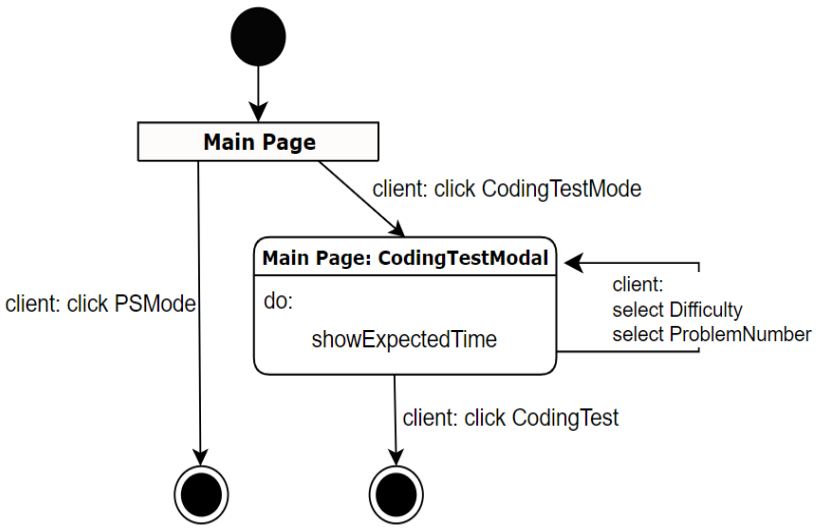

Figure 4. State Diagram of Main Page

사용자가 PSMode를 클릭한 경우, 다음 페이지로 넘어간다. 사용자가 CodingTestMode를 클릭

한 경우, CodingTestModal은 사용자로부터 Difficulty와 Problem Number를 선택받고 이에 맞는 ExpectedTime을 알려준다. 사용자가 CodingTest를 클릭한 경우, 다음 페이지로 넘어간다.

4.3. PS Select Page

DevNavi의 문제 선택 페이지로, 학습자가 문제풀이(PS) 모드를 선택한 경우에 도착하는 페이지이다. 사용자는 키워드로 문제를 검색할 수 있으며 난이도,태그를 설정할 수 있다. 조건에 맞는 문제들은 문제 리스트 박스에 보여진다. 사용자가 원하는 문제를 선택할 경우, 문제풀이를 시작할 수 있다.

4.3.1. Attributes

PS 선택 페이지는 아래와 같이 구성되어 있다.

- SearchBox: 사용자가 난이도, 태그를 설정해 문제를 검색할 수 있는 box이다.
- ProblemListBox: 사용자가 선택할 수 있는 문제 리스트를 보여준다.

4.3.1.1. Attributes of SearchBox

- SearchWord: 사용자가 검색한 단어이다.
- PSDifficulty: 사용자가 설정한 난이도이다. 1부터 5까지 존재한다.
- PSTag: 사용자가 설정한 태그이다.
- SearchButton: 사용자가 문제를 검색하고 싶을 때 누르는 버튼이다.

4.3.1.2. Attributes of ProblemListBox

- ProblemNumber: 문제의 번호이다.
- ProblemTitle: 문제의 제목이다.
- ProblemDifficulty: 문제의 난이도이다.

4.3.2. Methods

PS 선택 페이지의 메소드는 아래와 같이 구성되어 있다.

4.3.2.1. Methods of SearchBox

- selectPSDifficulty(): 사용자가 선택한 PSDifficulty로 PS의 난이도를 설정한다.
- selectPSTag(): 사용자가 선택한 PSTag로 PS의 태그를 설정한다.
- selectSearchWord(): 사용자가 검색할 단어인 SearchWord를 입력한다다.
- clickSearchButton(): 사용자가 SearchButton을 누른다.

4.3.2.2. Methods of ProblemListBox

- searchProblem(): 사용자가 선택한 PSDifficulty, PSTag, SearchWord로 문제를 검색한다.
- showProblemList(): searchProblem()의 결과값으로 나온 문제 리스트를 보여준다.
- clickProblem(): 사용자가 선택한 문제로 PS 페이지를 진행한다.

4.3.3. State Diagram

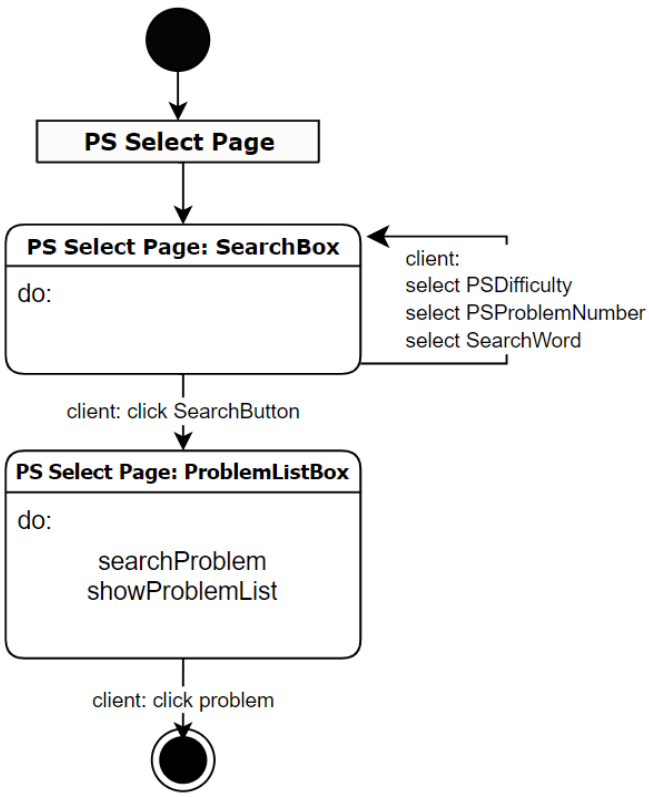

Figure 5. State Diagram of PS Select Page

사용자가 PSMode를 클릭한 경우, PS Select 페이지에 도착한다. 사용자가 PSDifficulty, PSProblemNumber, SearchWord를 선택/입력하고 SearchButton을 클릭하면 ProblemListBox에서 문제를 검색하는 searchProblem을 실행한다. 해당 결과값을 showProblemList에 표시한다. 사용자가 문제를 선택하여 clickProblem한 경우, 다음 페이지로 넘어간다.

4.4. PS Page

해당 페이지에서 사용자는 문제를 풀게 된다. 사용자는 문제에 맞는 코드를 코드 에디터에 작성한다. 해당 페이지에는 언어 선택, 코드 리셋, 테스트 실행, 제출 버튼이 존재한다. 언어 선택은 코딩 언어를 변경하는 버튼이고 코드 리셋은 코드 에디터에 작성된 내용을 초기화하는 버튼이다. 테스트 실행을 통해 문제에 맞는 테스트케이스에 대한 결과를 확인할 수 있고, 제출 버튼을 누르면 페이지가 Feedback Page로 넘어간다.

PS Page는 사용자가 선택한 모드에 따라 다르게 페이지가 구성된다. 사용자가 Main Page에서 코딩 연습 모드를 선택했을 경우, 페이지에 PS Select Page에서 사용자가 선택한 문제가 나타난다. 사용자가 코딩 테스트 모드를 선택했을 경우에는 사용자의 설정에 따른 PS 문제들이 나타나는데, 문제 번호 버튼을 통해 해당하는 문제들을 하나씩 확인할 수 있다.

4.4.1. Attributes

PS 페이지는 아래와 같이 구성되어 있다.

- ProblemBox: 문제가 표시되는 박스이다.
- CodeEditBox: 사용자가 코드를 작성할 수 있는 코드 에디터이다.
- RunBox: 사용자가 테스트 실행을 했을 때 결과가 표시되는 박스이다.
- ButtonBox: PS page에서 필요한 버튼들이 표시되는 박스이다.

4.4.1.1. Attributes of ProblemBox

- ProblemTitle: 문제의 제목을 나타낸다.
- ProblemContent: 문제의 내용을 나타낸다.

사용자가 코딩 테스트 모드를 선택했을 경우,

- ProblemNumButton: 클릭 시, 번호에 해당하는 문제가 표시되도록 하는 버튼이다.

4.4.1.2. Attributes of RunBox

- Code: 사용자가 작성한 코드이다.

4.4.1.3. Attributes of ButtonBox

- TestResult: 테스트케이스를 실행했을 때의 결과를 나타낸다.

4.4.1.4. Attributes of CodeEditBox

- LanguageSeletionButton: 코딩 언어를 선택하는 버튼이다.
- CodeResetButton: 코드 에디터를 초기화하는 버튼이다.
- TestButton: 테스트케이스를 실행하는 버튼이다.
- SubmitButton: 작성한 코드를 제출하는 버튼이다.

4.4.2. Methods

PS 선택 페이지의 메소드는 아래와 같이 구성되어 있다.

- setProblemTitle(): 사용자가 선택한 문제에 해당하는 Title을 가져와서 내용을 표시한다.
- setProblemContent(): 사용자가 선택한 문제에 해당하는 Content를 가져와서 내용을 표시한다.
- clickProblemNum(): 클릭한 문제 번호에 맞는 문제 정보를 가져온다.
- resetCode(): 코드 에디터에 적힌 내용을 초기화한다.
- testCode(): 테스트케이스를 통해 사용자가 작성한 코드를 실행하고 그 결과를 저장한다.
- submitCode(): 코드 에디터에 작성된 코드를 저장한다.
- setCodeLangauge(): 코드 언어를 설정한다.
- setTestResult(): 테스트 실행 결과를 표시한다.

4.4.3. State Diagram

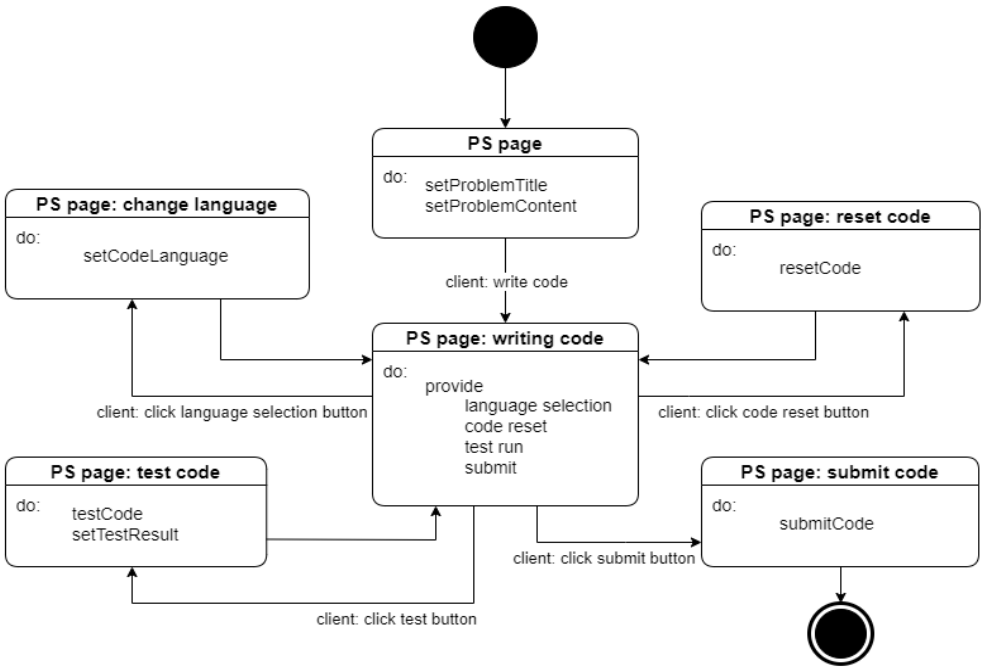

Figure 6. State Diagram of PS Page – PS Mode

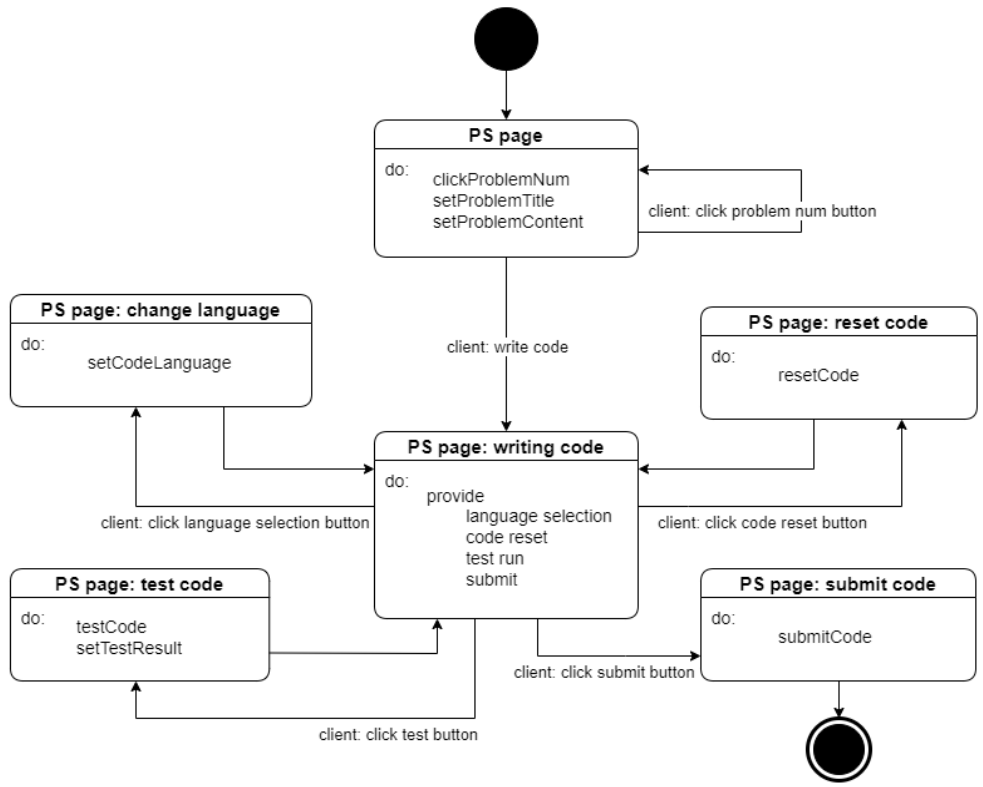

Figure 7. State Diagram of PS Page – Coding Test Mode

4.5. Feedback Page

사용자가 문제를 푼 뒤에 결과를 확인할 수 있는 페이지이다. 화면은 크게 좌측 사이드바, 상단, 중앙 세 개의 섹션으로 구분된다. 상단은 5개의 탭으로 구성되는데, 각각의 탭은 가독성, 시간 복잡도, 개선된 코드, 분석, 피드백이 존재한다. 그리고 좌측 사이드바는 코딩 테스트의 문항 번호가 나타난다. 사용자는 좌측의 문항 번호와 상단의 탭의 조합을 통해 각 문항별 상세한 결과를 알 수 있다. 결과는 중앙에 나타나고 개선된 코드는 중앙이 수직선으로 분할되어 좌측은 사용자가 제출한 코드, 우측은 개선된 코드가 나타난다. 개선된 코드를 제외한 나머지 탭은 모두 중앙이 병합되어 하나로 나타나며 분석, 피드백은 문항의 번호에 관계 없이 모두 동일하다.

우측하단의 '면접' 버튼을 클릭하면 면접 페이지로 이동할 수 있다.

4.5.1. Attributes

피드백 페이지는 아래와 같이 구성되어 있다.

- ProblemList: 화면 좌측에 각각의 문항 번호가 수직정렬되어 표시되는 박스이다.
- MenuBar: 상단에 5개의 탭이 수평정렬되어 표시되는 박스이다.
- ContentArea: 중앙에 결과들이 표시되는 박스이다.
- InterviewButton: 우측 하단에 있으며, 면접 화면으로 이동하는 버튼이다.

4.5.1.1. Attributes of ProblemList

- ProblemNo: 문제의 번호를 나타낸다.

4.5.1.2. Attributes of MenuBar

- ReadabilityButton: 가독성을 표시하는 버튼이다.
- TimeComplexityButton: 시간 복잡도를 표시하는 버튼이다.
- ImprovedCodeButton: 개선된 코드를 표시하는 버튼이다.
- AnalysisButton: 분석을 표시하는 버튼이다.
- FeedbackButton: 피드백을 표시하는 버튼이다.

4.5.2. Methods

피드백 페이지의 메소드는 아래와 같이 구성되어 있다.

- setProblemNo(): 문항 번호를 선택한다.
- setResultType(): 화면에 표시할 결과를 선택한다.
- showResult(): 문항 번호와 선택한 결과가 인자로 전달되며 결과가 중앙에 표시된다.
- showReadability(): 가독성 결과를 표시한다.
- showTimeComplexity(): 시간 복잡도 결과를 표시한다.
- showImprovedCode(): 개선된 결과를 표시한다.
- showAnalysis(): 분석 결과를 표시한다.
- showFeedback(): 피드백 결과를 표시한다.
- loadInterviewPage(): 인터뷰 페이지로 이동한다.

4.5.3. State Diagram

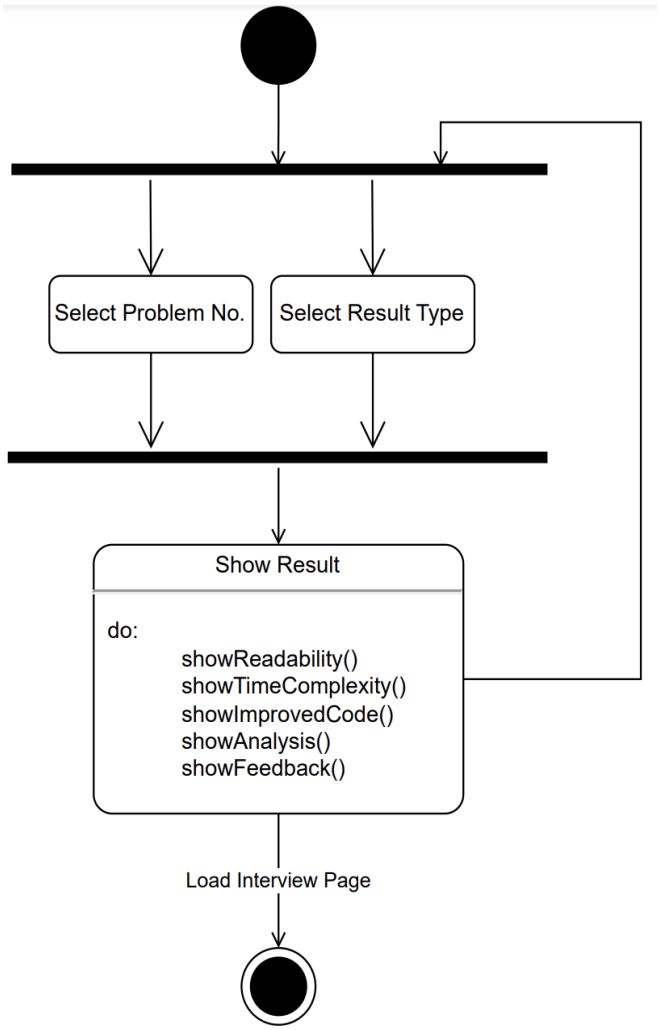

Figure 8. State Diagram of Feedback Page

5. SYSTEM ARCHITECTURE - BACK-END

5.1. Objectives

Back-end의 전반적인 구조와 서브시스템 구조를 설명한다. 또한 각 서브시스템에 대해 Class Diagram과 Sequence Diagram을 제시한다.

5.2. Overall Architecture

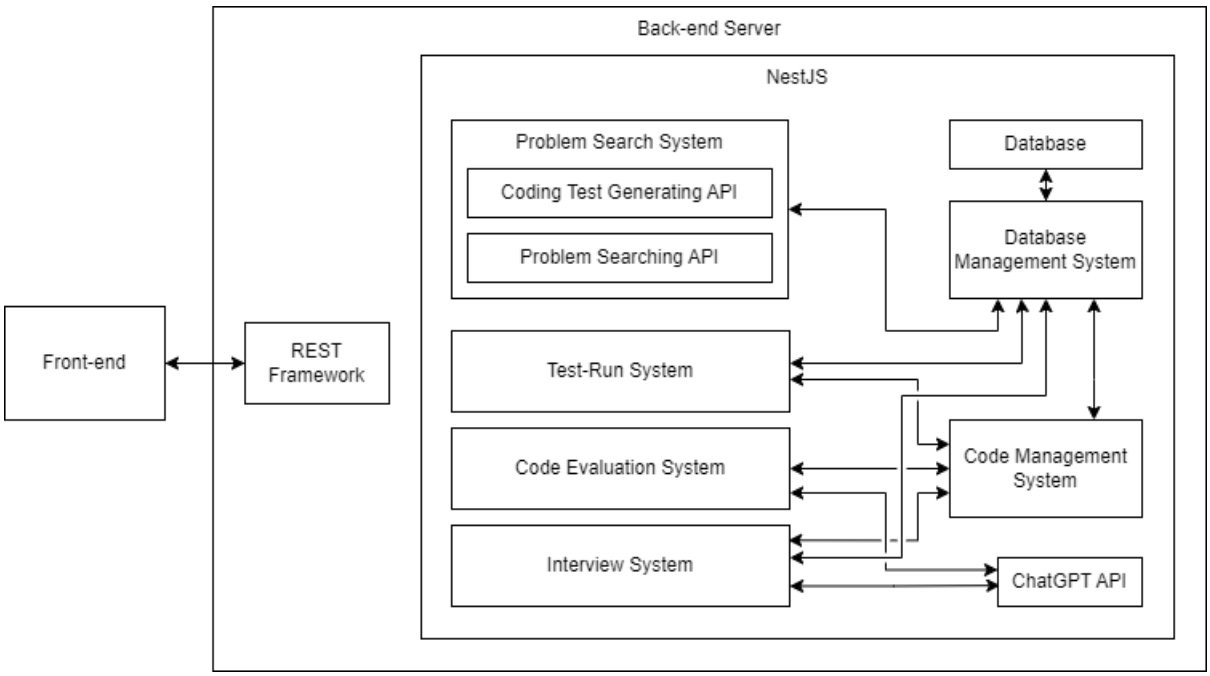

Figure 9. Overall Back-end Architecture

백엔드 서버의 구조는 그림과 같다. 먼저 프론트엔드 서버에서 호출된 API에 해당하는 시스템이 REST Framework를 통해 호출된다. PS에 사용할 문제를 찾거나 코딩 테스트를 생성하는 데에 필요한 Problem Search System, PS 페이지에서 작성한 코드를 테스트케이스를 사용해 테스트하는 데에 필요한 Test-Run System, 코드의 효율성이나 가독성 등 코드의 품질을 평가하는 데에 필요한 Code Evaluation System, 작성한 코드나 프로그래밍 지식 등을 기반으로 면접 질문을 생성하거나 면접 과정에서 응답을 처리하는 데에 필요한 Interview System을 비롯하여 전체적인 코드를 관리하는 Code Management System과 Database Management System이 갖추어져 있다.

5.3. Problem Search System

PS에 사용할 문제를 찾거나 코딩 테스트를 생성하는 시스템이다.

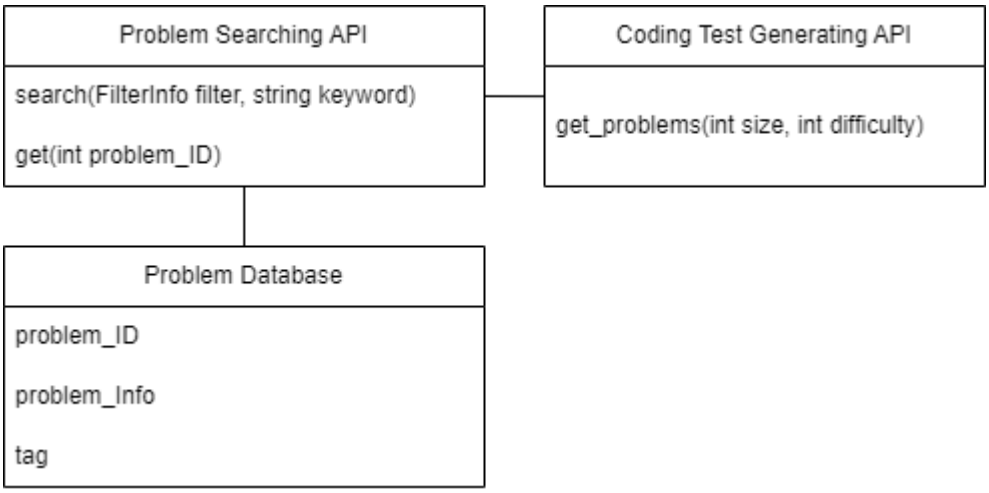

Figure 10. Class Diagram of Problem Search System

- Problem Searching API

해당 API에는 크게 두 가지 메소드가 작동하게 된다. 첫째로, search() 메소드를 이용할 수 있다. 필터나 키워드를 사용하여 검색을 하게 되면 그 정보를 바탕으로 관련된 문제의 리스트를 Problem DB에서 받아올 수 있다. 또한, get() 메소드를 이용하여 풀 문제를 정할 수 있는데, problem_ID에 해당하는 문제의 정보를 마찬가지로 Problem DB에서 받아오게 된다.

- Coding Test Generating API

해당 API에서는 사용자가 선택한 문제풀이 수나 난이도에 대한 정보를 받아 Problem Searching API와 연계하여 문제 정보를 요청하고 받게 된다. 해당 처리 과정은 get_problems() 메소드를 이용하여 진행된다.

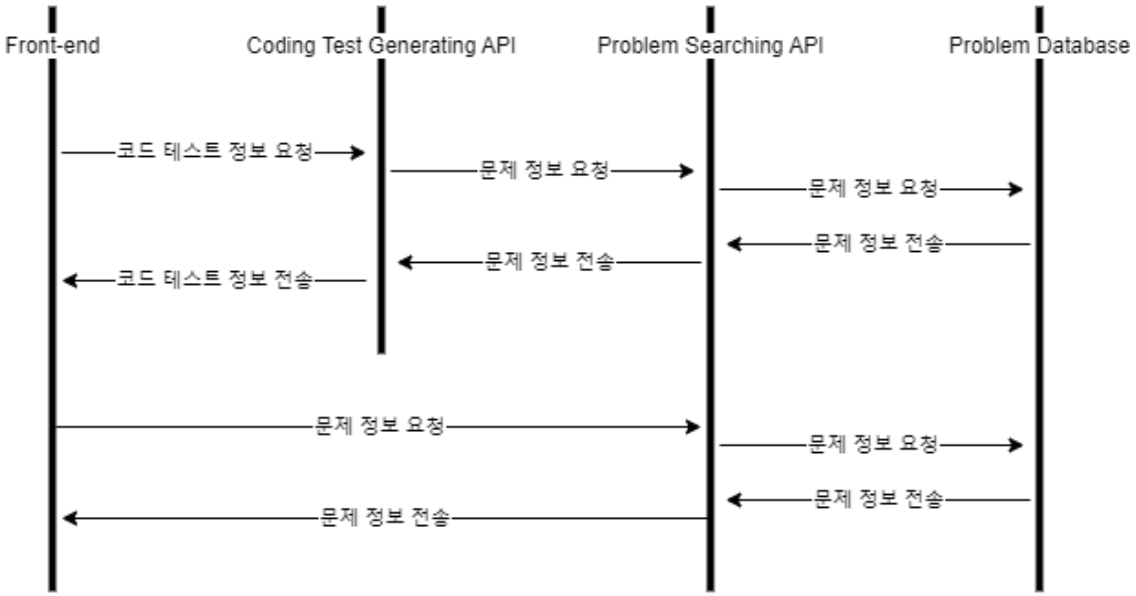

Figure 11. Sequence Diagram of Problem Search System

5.4. Code Management System

전체적인 코드의 관리를 담당하는 시스템이며, 이 시스템은 데이터베이스에 저장되는 코드 정보를 활용해야 하는 다른 시스템과 상호작용하는 데에 주로 사용된다.

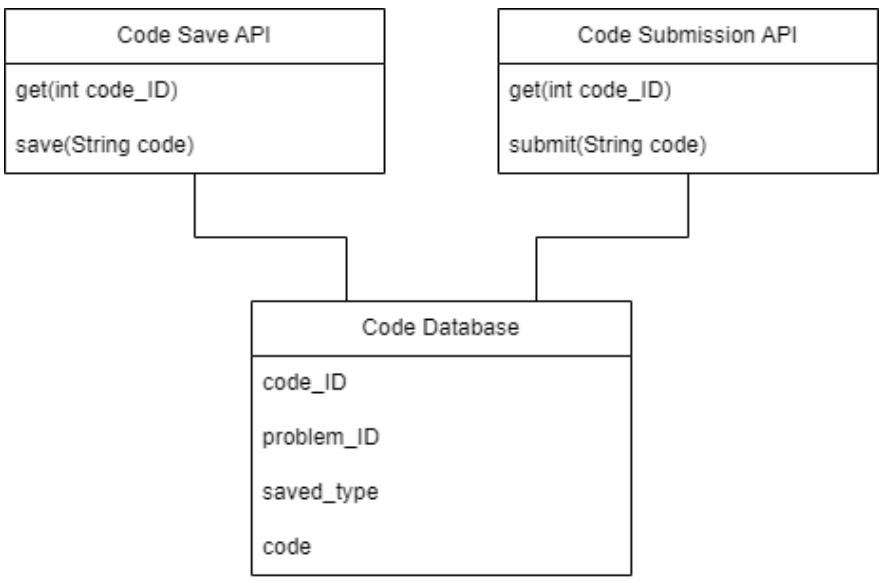

Figure 12. Class Diagram of Code Management System

- Code Save API

Code DB와 상호작용하여 사용자의 코드 저장이나 코드 불러오기를 처리한다. code_ID를 인자로 받아 사용자의 코드를 불러오는 get() 메소드와 작성된 코드 문자열을 인자로 두어 코드를 저장하는 save() 메소드가 존재한다.

- Code Submission API

위와 마찬가지로 Code DB와 상호작용하여 사용자의 코드 저장이나 코드 불러오기를 처리하지만, 코드의 단순저장 과정보다는 코드를 최종적으로 제출하여 PS나 코딩 테스트 결과에 영향을 준다는 점에서 처리 영역에 약간의 차이가 있다. code_ID를 인자로 받아 사용자의 코드를 불러오는 get() 메소드와 작성된 코드 문자열을 인자로 두어 코드를 저장하는 save() 메소드가 존재한다.

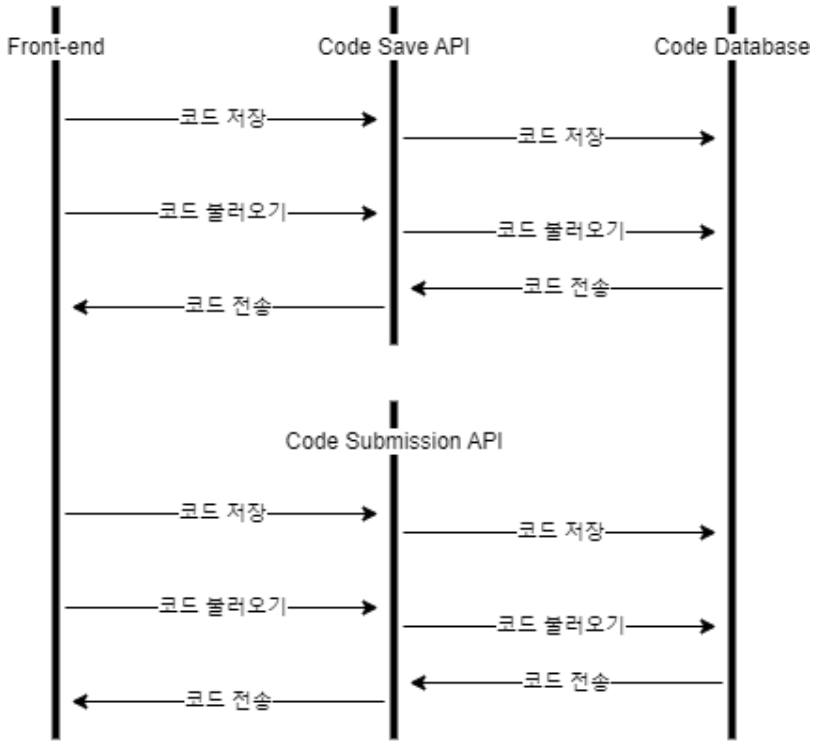

Figure 13. Sequence Diagram of Code Management System

5.5. Test-Run System

사용자가 작성한 코드를 실행하고 그 결과를 테스트 케이스와 비교해서 채점하는 시스템이다.

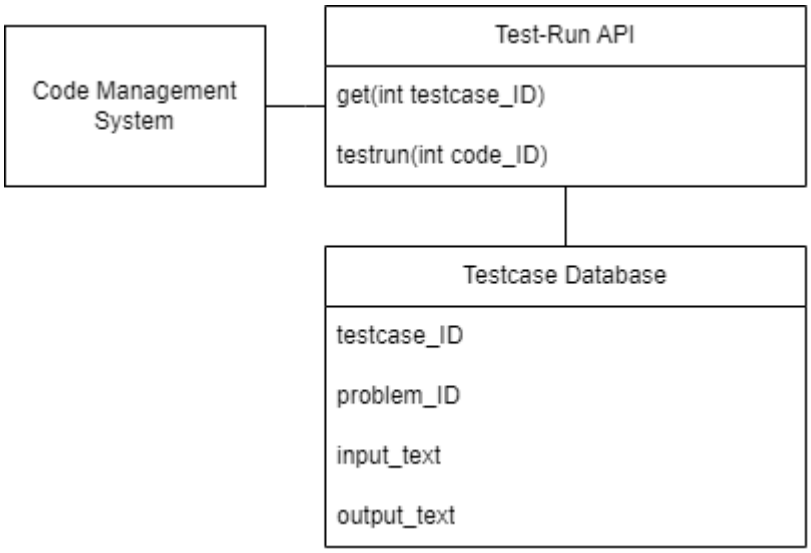

Figure 14. Class Diagram of Test-Run System

- Test-Run API

해당 API에는 크게 두가지 메소드가 정의되어 있다. 먼저 testcase_ID를 이용하여 테스트케이스를 검색할 수 있는 get() 메소드가 있고, 사용자가 작성한 코드를 받아서 채점을 수행하는 testrun() 메소드가 있다. 사용자가 작성한 코드에 대해 채점을 요청하면 test-run system에서 testrun() 메소드를 이용하여 체점을 하고, 채점 결과를 다시 사용자에게 전달해주게 된다.

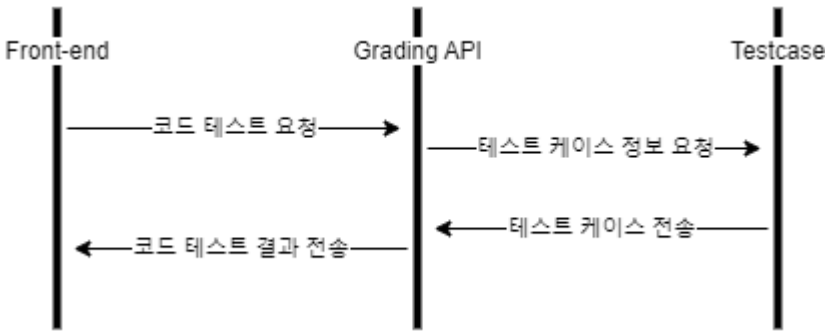

Figure 15. Sequence Diagram of Test-Run System

5.6. Code Evaluation System

사용자가 작성한 코드에 대해 복잡도, 가독성과 같은 특성, 그리고 개선된 코드에 해당하는 ChatGPT의 답변을 사용자에게 전달하는 시스템이다.

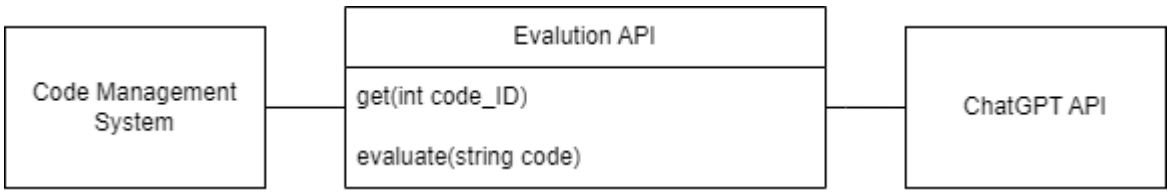

Figure 16. Class Diagram of Code Evaluation System

- Evaluation API

해당 API는 먼저 사용자가 작성한 코드를 전달받고, 그것을 이용하여 코드의 평가를 받을 수 있는 질문을 생성한 후 API를 이용하여 ChatGPT에 전달한다. ChatGPT에게서 답변을 받으면 다시 그 답변을 사용자에게 전달해주게 된다.

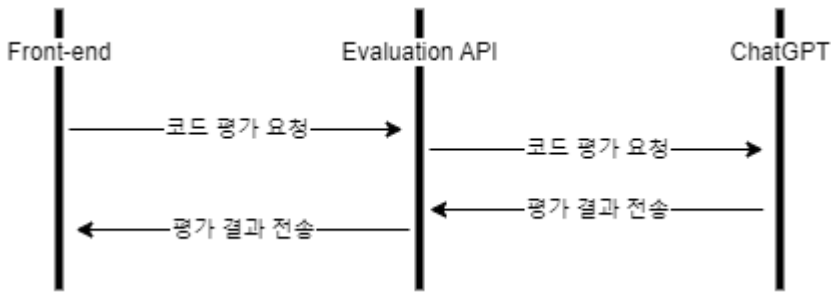

Figure 17. Sequence Diagram of Code Evaluation System

5.7. Interview System

코딩 테스트의 결과 페이지에서 사용자의 가상 면접 요청을 처리하는 시스템이다.

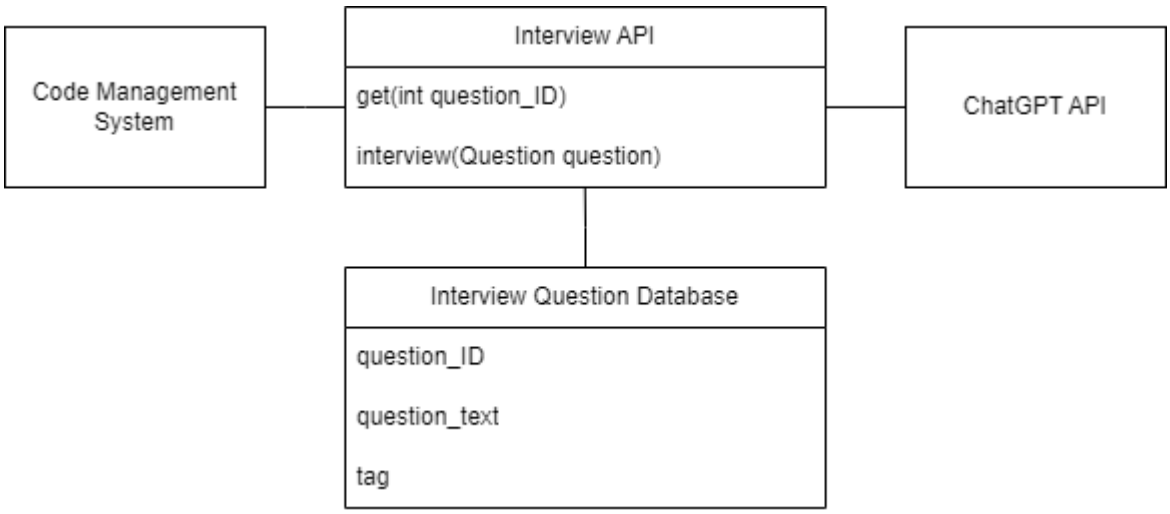

Figure 18. Class Diagram of Interview System

- Interview API

해당 API에는 크게 두가지 메소드가 정의되어 있다. 먼저 question_ID를 인자로 전달받아 해당하는 질문을 반환해주는 get() 메소드가 있다. 그리고 면접 시작 요청을 받았을 때 문제 모음을 생성하여 반환해주는 interview() 메소드가 있다. 사용자에게 면접 실행을 요청받았을 경우, API는 두가지 경로를 통해 면접 질문을 구성한다. 첫번째 경로는 데이터베이스에 사전 저장되어 있는 질문들 중 코딩 테스트 문제와 유사한 문제들 위주로 검색하여 질문을 구성하는 방법이고, 두번째 경로는 사용자가 작성한 코드를 기반으로 ChatGPT에게서 생성된 질문을 이용하여 구성하는 방법이다. 두가지 경로를 이용하여 문제 모음을 구성한 후, 이를 사용자에게 반환해주게 된다.

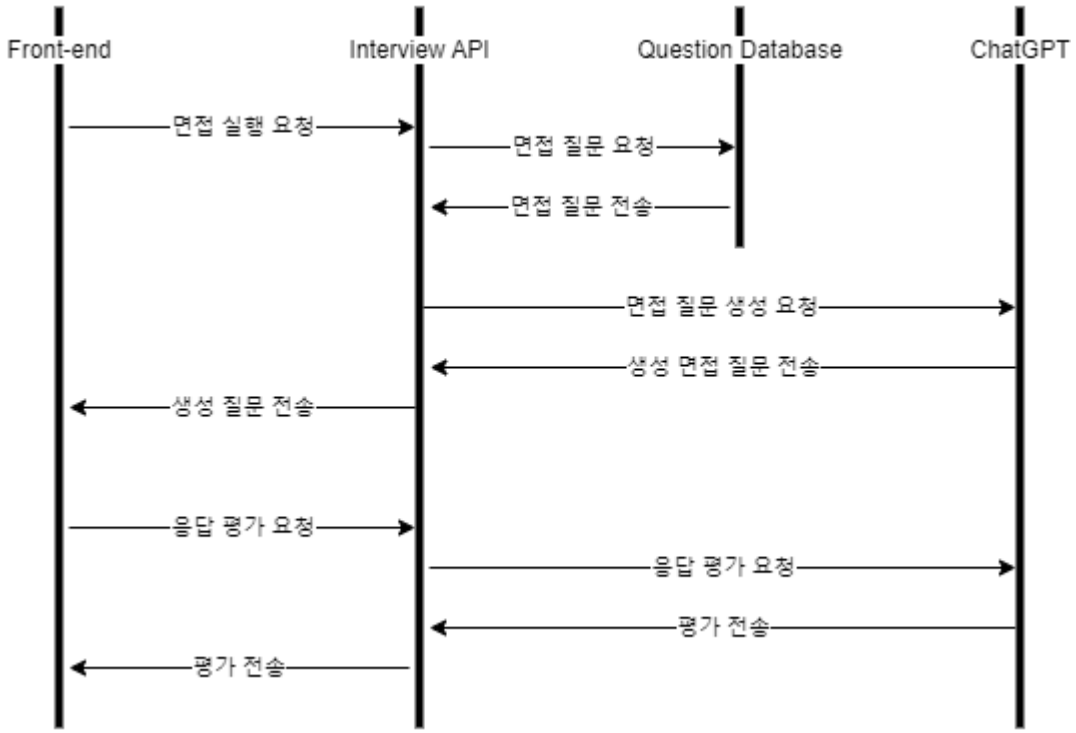

Figure 19. Sequence Diagram of Interview System

6. APPLICATION PROGRAMMING INTERFACE

6.1. HTTP

클라이언트-서버 프로토콜로, 클라이언트에서의 개별 요청이 서버로 보내지면 서버가 그 요청을 처리하여 응답을 제공하여 HTML과 같은 리소스를 가져오는 데에 필요한 체계이다.

6.2. JSON

데이터 구조를 객체 문법으로 표현하는 포맷으로, Serialize가 가능한 값이나 Key-Value 쌍 등으로 이루어진 데이터 객체를 주고받는 데에 사용한다.

6.3. API Description

6.3.1. Login API

Table 1. Properties of Login API

Method	GET	
URI	/api/login/oauth	
Description	유저가 OAuth로 로그인을 하기 위해 provider authentication page에 redirect	
Parameter	provider	OAuth의 제공자 (GitHub)
Body	None	
Response	url	redirect page url
Status Code	301 redirect	OAuth 인증 성공
	401 unauthorized	OAuth 인증 실패

6.3.2. Problem Search API

Table 2. Properties of Problem Search API

Method	GET	
URI	/api/problems	
Parameter	difficulty	코딩 테스트의 난이도
	tag	코딩 문제에 설정되어 있는 tag
	keyword	사용자가 검색하고자 입력한 키워드
Response	problemsarray	검색된 코딩테스트 문제 리스트의 배열
Status Code	200	문제 리스트 반환 성공
	400 bad request	잘못된 요청 형식이나 파라미터
	404 not found	일치하는 문제 존재하지 않음

6.3.3. Coding Test Generating API

Table 3. Properties of Coding Test Generating API

Method	GET	
URI	database/problemDatabase	
Parameter	difficulty	코딩 테스트의 난이도
	num	코딩 테스트를 구성하는 문제 개수
Response	problems	생성된 코딩테스트 문제 리스트
Status Code	200	코딩테스트 문제 구성 성공
	400	코딩테스트 문제 구성 실패

6.3.4. Code Evaluation API

Table 4. Properties of Code Evaluation API

Method	POST	
URI	/submissions/<problem_id>	
parameter	problem_id	문제의 id
Response	results	채점 결과, 실행 시간, 메모리 사용량이 포함됨
Body	code_id	사용자가 제출한 코드
Status Code	200	채점이 성공하고 채점 결과가 올바르게 반환됨
	400 bad request	잘못된 요청 형식이나 파라미터
	404 not found	주어진 problem-id 에 해당하는 문제를 찾을 수 없음
	500 Internal server error	서버에서 채점 과정 중 오류 발생

6.3.5. Interview API

Table 5. Properties of Interview API

Method	GET	
URI	interview/<code_id>	
parameter	code_id	사용자가 작성한 코드의 id
Response	questions	생성된 면접 질문 리스트
Status Code	200	질문 구성 성공
	400	질문 구성 실패

6.3.6. Code Save API

Table 6. Properties of Code Save API - Save

Method	POST	
URI	/submissions/<problem_id>	
Parameter	problem_id	문제의 id
Response	result	코드 저장 성공 여부
Status Code	200	코드 저장 성공
	400	코드 저장 실패

Table 7. Properties of Code Save API - Load

Method	GET	
URI	/submissions/<problem_id>/<code_id>	
Parameter	problem_id	문제의 id
	code_id	사용자가 작성한 코드의 id
Response	code	불러온 코드 내용
Status Code	200	코드 불러오기 성공
	400	코드 불러오기 실패

7. DATABASE DESIGN

7.1. Objectives

DB 디자인의 전반적인 내용과 NoSQL 기반의 Entity를 ER Diagram을 통해 설명한다.

7.2. Collections

NoSQL 데이터베이스의 "collection"은 관계형 데이터베이스의 "table"과 유사한 개념이다. DB table과 달리 유연한 schema를 가지며, 또한 data type도 다양하게 할당할 수 있다.

7.2.1. User

```
{
    "_id": "user1",
    "username"?: "username1",
    "email"?: "user1@example.com",
    "oauth": "GitHub",
    "createAt": "2023-05-18T10:30:00Z",
    "modifiedAt"?: "2023-05-18T10:30:00Z",
    "deletedAt"?: "2023-05-18T10:30:00Z",
    "isDeleted": false
}
```

7.2.2. Problem Metadata

```
{
    "_id": "problem1",
    "title": "problem title",
    "difficulty": 4,
    "tags": ["tag1", "tag2"],
}
```

7.2.3. Problem Description

```
{
    "_id": "problem1",
    "description": "problem content...",
    "contraint": ["0 < a < 10", "a < b < 100"],
    "testcases": [
        {
            "input": "test input",
            "output": 3
        },
        {
            "input": 23,
            "output": "expected output 2"
        }
    ]
}
```

7.2.4. Problem Solution

```
{
    "_id": "solvedProblem1",
    "userId": "user1",
    "problemId": "problem1",
    "language": "python",
    "code": "solution code for problem 1",
    "submitTime": "2023-05-18T10:30:00Z"
}
```

7.2.5. Code Language

```
{
    "language name(id)": "python",
    "default code": "def solution() {\n\tanswer = 0\n\return\n\t}",
}
```

8. ADDITIONAL INFORMATIONS

8.1. Software Design Specification

본 문서는 IEEE 양식(IEEE Standard for Information Technology--Systems Design--Software Design Descriptions)에 따라 작성되었다, 단, 본 시스템 개발 상황과 규모를 고려하여 본래의 양식에서 변형된 부분이 있다.

8.2. Document History

Table 8. Document History

Date	Version	Description	Written by
2023.05.07.	0.1	문서 초안 작성	권민성, 서주원, 안서현, 안정민, 이장엽, 한수현
2023.05.21.	1.0	최초 발행	권민성, 서주원, 안서현, 안정민, 이장엽, 한수현