

Requirements Specification



Team 1

2016311264 김준우

2017310463 최재익

2017312780 유찬우

2017314819 장윤진

2019314623 장채운

2019314352 김이지

목차

1. Introduction – 5 page ~ 8 page
 - 1.1. Purpose – 5 page
 - 1.2. Scope – 5 page
 - 1.3. Definition, Acronyms, and Abbreviation – 6 page ~ 8 page
 - 1.4. References – 8 page
2. Overall Description – 8 page ~14 page
 - 2.1. Product Perspective – 8 page ~10 page
 - 2.2. User Classes and Characteristics – 10 page ~11page
 - 2.3. Operating Environment – 12 page
 - 2.4. Design and Implementation Constraints – 12 page
 - 2.5. Assumptions and Dependencies – 13 page ~14 page
3. Specific Requirements – 14 page ~ 32 page
 - 3.1. External Interface Requirements – 14 page ~20 page
 - 3.2. Functional Requirements 20 page ~28 page
 - 3.3. Non-functional Requirements 28 page~32 page
4. System Models – 32 page ~37 page
 - 4.1. System Architecture – 32 page
 - 4.2. System Context Diagram – 33 page
 - 4.3. Process Model – 33 page

- 4.4. Interaction Model – 34 page
- 4.5. Behavior Model – 37 page

- 5. System Evolution – 38 page ~ 39 page
 - 5.1. Limitation and Assumption – 38 page
 - 5.2. Change of User Requirements – 39 page

그림 목차

- [그림 1] 메인화면의 섹션별 구성 – 9 page
- [그림 2] Overall System Architecture – 32 page
- [그림 3] Overall System Context Diagram – 33 page
- [그림 4] Overall Process Model Diagram – 33 page
- [그림 5] Usecase Diagram – 34 page
- [그림 6] Sequence Diagram for the Lecture – 35 page
- [그림 7] Sequence Diagram for the Students – 36 page
- [그림 8] Overall State Diagram – 37 page

표 목차

- [표 1] 약어 정리 - 7 page
- [표 2] 사용자 상호작용(1) - 로그인 창 - 15 page
- [표 3] 사용자 상호작용(2) - 회원가입 창 - 16 page
- [표 4] 사용자 상호작용(3) - 코딩 문제 창 - 17 page
- [표 5] 사용자 상호작용(4) - 코딩 결과 창 - 19 page
- [표 6] 기능별 요구사항(1) - 회원가입 - 20 page
- [표 7] 기능별 요구사항(2) - 로그인 - 21 page
- [표 8] 기능별 요구사항(3) - 문제 보여주기 - 22 page
- [표 9] 기능별 요구사항(4) - 테스트 케이스 보여주기 - 22 page
- [표 10] 기능별 요구사항(5) - 코드 하이라이트 - 23 page
- [표 11] 기능별 요구사항(6) - 코드 저장 - 23 page
- [표 12] 기능별 요구사항(7) - 파일 불러오기 - 24 page
- [표 13] 기능별 요구사항(8) - 코드 초기화 - 24 page
- [표 14] 기능별 요구사항(9) - 코드 복사 - 25 page
- [표 15] 기능별 요구사항(10) - 코드 다운로드 - 25 page
- [표 16] 기능별 요구사항(11) - 코드 실행 - 26 page
- [표 17] 기능별 요구사항(12) - 코드 채점 - 26 page
- [표 18] 기능별 요구사항(13) - 코드 제출 - 27 page
- [표 19] 기능별 요구사항(14) - 코드 결과 분석 - 28 page

1. Introduction

1.1. Purpose

- 본 문서는 파이썬 교육 시스템을 설계 및 구현하기 위한 요구사항 명세서이며 SWE3002-41분반 Team 1에 의해 운용된다. 본 문서에서 파이썬 교육 시스템을 위한 요구사항을 정리, 분석, 기재한 내용을 바탕으로 시스템을 설계 및 구현한다.
- 본 문서는 전반적인 요구 사항을 서술하여 완성도 높은 웹 서비스를 개발하고 발전된 서비스를 제공하는 것을 최우선 목적으로 한다.
- 본 코딩 교육 프로그램의 목적은 사용자들에게 양질의 코딩 문제들을 전달함과 동시에 이를 웹상에서 풀어 구체적으로 평가받을 수 있는 인터페이스를 제공함으로써 비전공자들이 코딩에 대한 보다 발전된 역량을 기르도록 돕는 것이다.
- 본 문서는 해당 프로그램을 위한 front-end, back-end, web design, server, data-base 등의 사용 및 설계를 위한 요구 사항들을 정리하였다.

1.2. Scope

- (기능 1) 헤드 버튼 기능
- (기능 2) 문제 및 참조/제약사항 설명 기능
- (기능 3) 테스트 케이스 검증 기능
- (기능 4) 코드 작성 및 중간 저장 기능
- (기능 5) 파일 불러오기, 코드 초기화/복사/다운로드 기능
- (기능 6) 코드 실행 기능
- (기능 7) 채점 기능
- (기능 8) 제출 및 결과 분석 기능

1.3. Definition, Acronyms, and Abbreviation

1.3.1. Definition

- Code: 소프트웨어(프로그램)를 만들 때 쓰는 설계도
- Test case: 프로그램이 의도대로, 혹은 요구사항대로 잘 동작하는 지 확인하기 위해 설계된 입력값과 출력값의 모음
- Input & Output: 사용자가 input을 코드에 입력하면 output을 출력하게 된다. 이를 테스트 케이스와 비교해보면서 평가 내릴 수 있다.
- 프로그래밍 언어: 실제로 프로그램을 작성할 때 사용되는 언어로, 기계어와, 어셈블러에 의해 번역되는 어셈블리 언어, 컴파일러에 의해 번역되는 컴파일러 언어, 인터프리터에 의해 번역되는 베이식 등이 있다.
- 고급 언어: 프로그래밍 언어 중 프로그래머가 특정 형식의 컴퓨터와는 무관하게, 독립적으로 프로그램을 작성할 수 있는 언어로, 기계어보다 인간의 언어에 더 가깝고, 보통 컴파일러나 인터프리터가 해석하여 기계어로 바꾸어 실행하게 된다.
- Python: 흔히 사용되는 고급 언어중 하나로, 1991년에 발표된 인터프리터 방식의 프로그래밍 언어다. 영어와 비슷해서 읽고 쓰기 쉬운 특유의 문법과 그로 인해 전 세계의 개발자들로부터 만들어진 수많은 패키지들 덕분에 사용할 수 있는 용도가 무궁무진하여 2010년대 중반부터 주류 프로그래밍 언어로 쓰인다.
- Parsing: 문장을 그것을 이루고 있는 구성 성분으로 분해하고 그들 사이의 위계 관계를 분석하여 문장의 구조를 결정하는 것이다. 해당

프로젝트에서는 파이썬 언어를 작성하였을 때, 작성된 문자열들을 의미를 가지는 토큰 단위로 분해하는 과정을 파싱이라고 하게 된다.

- 코드 실행: 실제로 컴파일러나 인터프리터 (파이썬 같은 경우는 python interpreter)를 사용하여 언어로써 작성된 프로그램을 실행하는 것이다.
- 쿠키: 웹사이트 접속시 접속자의 개인장치에 다운로드 되고 브라우저에 저장되는 작은 텍스트 파일. 쿠키를 통해 접속자의 장치를 인식하고, 접속자의 설정과 과거 이용내역에 대한 일부 데이터를 저장한다.
- DB: 구조화된 정보 또는 데이터의 조직화된 모음으로서 일반적으로 컴퓨터 시스템에 전자적으로 저장되며 일반적으로 데이터베이스 관리 시스템(DBMS)에 의해 제어된다.
- DBMS: 데이터 베이스를 쉽게 구축하고 관리할 수 있게끔 만들어 놓은 관리 시스템과 그 인터페이스이다. 해당 프로젝트에서는 프로젝트에서 사용할 DBMS인 SQLITE와 구분 없이 사용하였다.

1.3.2. Acronyms, Abbreviations

[표 1] 약어정리

Acronyms, Abbreviations	Software requirement specification
UI	User Interface

UX	User Experience
APP	Application
API	Application Programming Interface
OS	Operating System
HTTP	Hypertext Transfer Protocol

1.4. References

"IEEE Recommended Practice for Software Requirements Specifications," in *IEEE Std 830-1998*, vol., no., pp.1-40, 20 Oct. 1998, doi: 10.1109/IEEESTD.1998.88286.

2. Overall Description

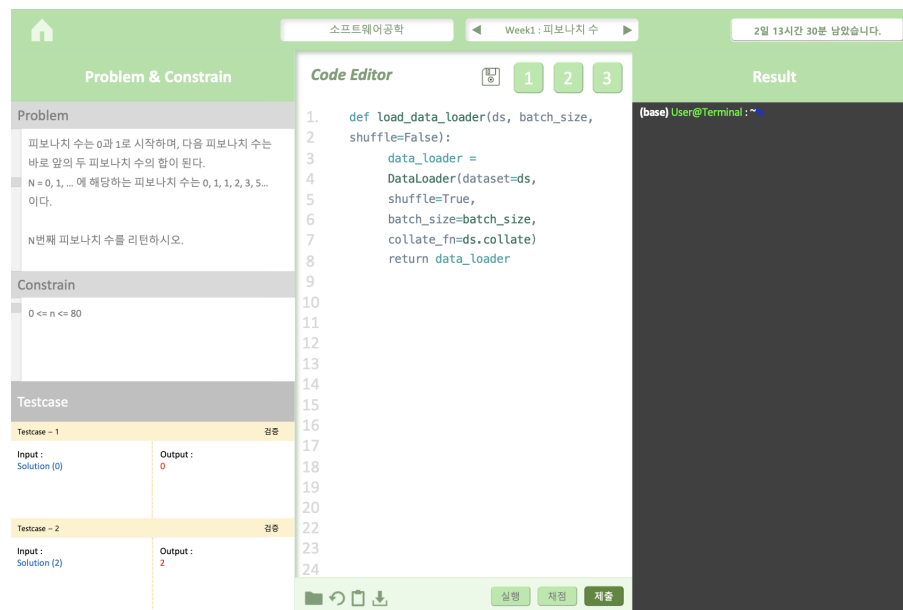
2.1. Product Perspective

본 제품은 기본적으로 사용자가 웹상에서 수강 과목 별 과제 / 문제들을 풀어보고, 평가받는 기본적인 코딩 테스트의 목적을 만족해야 한다. 메인화면은 크게 아래와 같이 구분되어 각각의 기능을 제공한다.

- 헤드 섹션
- 문제 설명 섹션
- 테스트 케이스 섹션

- 코드 에디터 섹션
- 각종 기능 버튼 섹션
- 결과 섹션 (터미널)

[그림 1] 메인화면의 섹션별 구성



- 2.1.1. 헤드 섹션에는 클릭 시 맨 처음으로 돌아갈 수 있는 홈버튼이 있고, 강의 제목과 과제 제목, 마감일이 명시되어 있으며 배경 색과 코드 에디터의 변경이 가능한 설정 버튼 또한 포함되어 있다.
- 2.1.2. 문제 설명 섹션은 문제와 참조/제약사항, 두 가지 세부 섹션으로 나누어져 있으며, 각 섹션에서는 문제에 대한 설명과 해당 문제에서의 각종 참조 내용 혹은 제약사항에 대한 설명을 나타낸다.

- 2.1.3. 테스트 케이스 섹션에서는 코드를 채점하기 위한 테스트 케이스를 작성하여 검증 버튼 클릭 시 output 여부에 따라 pass/fail 결과를 제공한다. 이때 테스트 케이스는 공개와 비공개로 구분되며 공개 테스트 케이스만 보이도록 설정되어 있다. 또한, fail일 경우 학생 코드의 output 결과를 같이 제공함에 따라 테스트 케이스를 일부 보여주고, 이에 해당하는 output이 어떻게 나와야 하는지를 보여준다.
- 2.1.4. 코드 에디터 섹션은 사용자가 코드를 작성할 수 있는 에디터를 제공하며, 작성 중 중간 저장을 할 수 있는 버튼 또한 포함된다.
- 2.1.5. 각종 기능 버튼 섹션은 파일 불러오기, 코드 초기화, 코드 복사, 코드 다운로드, 실행, 채점, 제출 버튼 등으로 구성되어 있다. 그중 실행, 채점은 여러 번 가능하게 설정되어 있으며 제출은 최대 3번까지만 가능하다.
- 2.1.6. 결과 섹션 (터미널)에서는 코드의 실행 결과를 보여준다. 에러가 아닐 경우는 출력값을 제공하고, 에러일 경우에는 에러 메시지, 에러 라인 하이라이트 등 에러에 대한 정보를 사용자에게 전달한다. 또한, 제출 이후에는 채점 결과를 점수, 오픈/히든 테스트 케이스의 pass/fail 여부 제공 등의 항목을 통해 보여준다.

2.2. User Classes and Characteristics

2.2.1. 사용자

- 학습자

학습자는 프로그래밍 언어의 기초 문법을 아는 학생으로 수강하는 과목의 과제를 본 서비스를 이용하여 풀게 된다. 코딩 문제를 푸는 데 있어 필요한 환경을 모두 제공받을 수 있어야 한다.

- 교수자

코딩 문제 출제 및 학습자의 제출결과를 바탕으로 학습자의 성적을 평가하는 사용자이다. 자신이 주관하는 과목이 있으며 자신의 과목을 수강하는 학습자의 코드 및 제출 결과를 확인할 수 있다.

2.2.2. 플랫폼 관리자 계층

- 문제 검증자

문제 출제자가 낸 문제들에 이상이 없는지, 실제로 테스트 케이스들을 잘 만족하는지, 그리고 특정한 수준을 만족하는지를 검증하거나 평가하는 관리자층이다.

- 서버 및 사용자 관리자

실제로 웹상에서 로그인하는 사용자들에게 문제점은 없는지, 혹은 이후 서비스 관리 차원에서의 대응은 해당 관리자가 맡게 된다. (실제 서버 가동하는 곳에서 고용한다.)

2.3. Operating Environment

2.3.1. 사용 애플리케이션에 따라

- 크롬
- 인터넷 익스플로러
- 마이크로소프트 edge
- 네이버 웨일
- 사파리

2.3.2. 사용 기기에 따라

- 데스크톱
- 노트북

2.3.3. 인터넷 환경

- 인터넷의 원활한 접속 상태
- 인터넷 쿠키 활성화 상태

2.4. Design and Implementation Constraints

2.4.1. 인터넷이 접속환경이 원활해야 한다.

DB와 인터페이스 상에서 데이터가 오가므로 중간에 인터넷이 끊길 경우 User에게 원활한 서비스 제공이 힘들 수 있다.

2.4.2. 반응형 웹페이지로 만들어져야 한다.

사용자들의 디스플레이 환경이 전부 다르므로 이에 맞춰서 웹페이지를 반응형으로 만들어 여러 디스플레이에서 UI를 의도한대로 보여주어야 한다.

2.5. Assumptions and Dependencies

2.5.1. 사용 기기는 데스크탑이나 노트북이라는 가정

태블릿과 스마트폰으로 코딩을 하는 사람은 거의 없다는 것을 고려하여, 사용자가 데스크탑이나 노트북에서 해당 플랫폼에 접근할 것이라는 가정이다.

2.5.2. 인터넷이 원활하다는 가정

웹 사이트는 웹 서버로서 존재하기 때문에 플랫폼에 접근하기 위해서는 인터넷 연결이 필수적이다. 이에 따라 플랫폼에 연결되어 있을 때는 인터넷 연결이 원활한 상태라고 가정 가능하다. 다만 중간에 인터넷이 한두번씩 끊기거나 실수로

종료하는 상황 등을 위하여 일정시간이 지나면 자동적으로 저장해주거나, 코드 실행 시 자동으로 저장해주는 등의 기능을 추가할 수 있다.

2.5.3. 쿠키가 허용 설정되어 있다는 가정

근래의 웹 사이트들은 쿠키가 필요한 시점에 사용자들에게 해당 웹 페이지의 쿠키를 허용할 것인지를 묻는다. 이때 만약 사용자가 사용 중인 기기가 신뢰할 만하다고 생각한다면 쿠키를 허용할 것이다. 이에 따라, 필요한 경우 쿠키가 허용 설정되어 있다고 가정할 수 있다.

2.5.4. 동시 접속자가 100명 이하라는 가정

서비스를 시작하고 나서 일반적으로 접근하게 될 사용자들의 수를 고려하여 제품의 초기 동시 접속자의 수는 100명 이하라고 가정하고 진행한다. 이후 시스템의 사용자가 증가하면 차근차근 동시 접속자 한도를 늘리는 방식으로 진행한다.

3. Specific Requirements

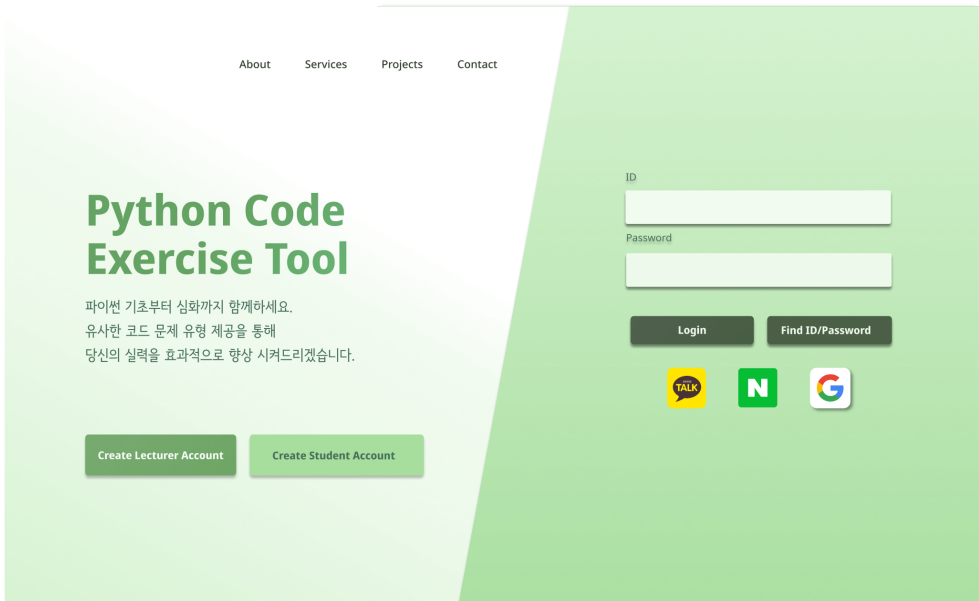
3.1. External Interface Requirements

3.1.1. User Interface

3.1.1.1. 로그인 창

[표 2] 사용자 상호작용(1) - 로그인 창


이름	로그인 창
목적 & 설명	<p>사용자가 ip주소, 도메인 이름을 통해 서비스에 처음 접근했을 때 보여지는 화면이다.</p> <p>이미 본 서비스의 계정을 보유하고 있는 사용자라면 ID, PW를 입력하여 본인의 계정에 접속할 수 있고, 처음 서비스를 이용하는 사용자라면 회원가입을 통하여 본인의 계정을 생성할 수 있다.</p>
입출력 형태	<p>사용자는 버튼 클릭으로 각 사용자 유형별 계정 등록 버튼을 선택할 수 있다. 또한 각 사용자는 클릭으로 ID/PW 창을 선택할 수 있으며, 엔터키와 클릭으로 로그인을 할 수 있다. 아이디, 비밀번호 찾기 탭 역시 클릭으로 선택할 수 있으며, 소셜로그인 탭 역시 클릭으로 수행할 수 있다. 위와 같은 클릭으로 동작을 수행할 경우 서버에 제출 가능한 HTTP POST 요청의 종류가 바뀐다. 본 인터페이스는 서버에 최초 접근할 때 수신한 Html, Css, JavaScript를 입력으로 형성된다.</p>

화면의 형태	
-----------	--

3.1.1.2. 회원가입 창

[표 3] 사용자 상호작용(2) – 회원가입 창

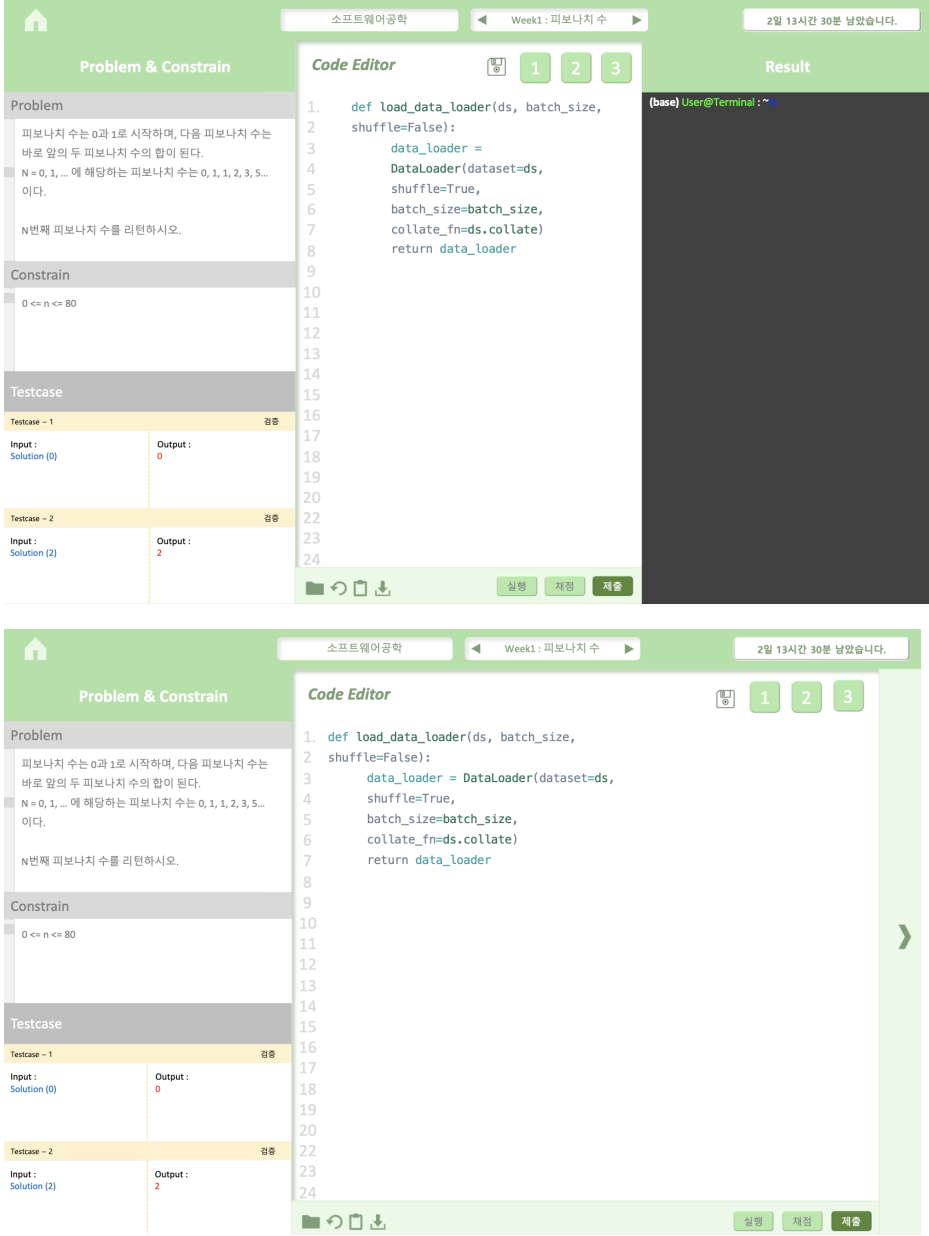
이름	회원가입 창
목적 & 설명	처음 서비스를 이용하는 사용자가 본 서비스를 이용하기 위해 계정을 생성하기 위해 이용하는 창이다. 이름, 아이디, 비밀번호, 비밀번호 확인, 이메일을 입력한 후 계정을 생성할 수 있다.
입출력 형태	사용자는 각 정보에 해당하는 탭을 마우스 클릭으로 선택할 수 있으며 탭키를 눌러 다음 칸을 선택할 수 있다. 가입하기 버튼을 마우스로 클릭할 경우 시스템의 데이터베이스에 접근하여 사용자가 입력한 정보에 대하여 유효성 검사를 실시한 뒤 아무런 문제가 없으면 계정 등록 완료 창을 출력하고, 문제가 존재할 경우 해당 정보에 대한 오류창을 출력한다.

화면의 형태	
-----------	--

3.1.1.3. 코딩 문제 창

[표 4] 사용자 상호작용(3) - 코딩 문제 창

이름	코딩 문제 창
목적 & 설명	코딩 문제/테스트 케이스를 보여주고 사용자에게 코드를 입력받으며, 이를 실행할 수 있다.
입출력 형태	<p>사용자는 상단의 과목명과 문제를 클릭으로 선택함으로써 문제를 볼 수 있다.</p> <p>에디터 확장: 에디터 창과 터미널 창의 경계를 마우스 클릭으로 이동시켜서 확장할 수 있다.</p> <p>저장: 디스크 버튼을 클릭하면 사용자가 쓴 코드가 사용자 데이터에 저장된다.</p> <p>제출코드 불러오기: 디스크 버튼 옆의 숫자 버튼을 클릭하면 해당</p>

	<p>제출 버전의 코드를 불러올 수 있다.</p> <p>초기화: 에디터 창의 하단에 있는 원형 화살표를 마우스로 클릭하여 동작하게 한다. 버튼을 클릭한 경우 해당 문제의 스텔레톤 코드를 불러와 기존의 내용을 초기화 한다.</p>
화면의 형태	 <p>The image displays two screenshots of a web application interface for a programming problem. The top screenshot shows the 'Code Editor' with a Python function 'load_data_loader' and a 'Result' terminal showing '(base) User@Terminal: ~'. The bottom screenshot shows the same interface but with the 'Code Editor' expanded to show more code and a green arrow pointing right, indicating a next step or action.</p>

3.1.2. 코딩 결과 창

[표 5] 사용자 상호작용(4) - 코딩 결과 창

이름	코딩 결과 창
목적 & 설명	사용자들 코드의 실행, 채점, 제출의 결과를 보여주는 창. 코드의 결과를 통해서 사용자들은 자신의 코드의 pass/fail여부를 확인할 수 있다.
입출력 형태	<p>코드 에디터 하단에 있는 실행, 채점, 제출버튼을 클릭해서 사용자 코드의 결과를 확인할 수 있다. 이때 HTTP POST요청을 통해서 코드 에디터에 작성한 코드를 렌더링한다.</p> <p>실행: 실행시 에러가 났을 경우 에러 위치를 하이라이트로 표시하고 에러 메시지를 화면에 나타낸다. 에러가 없으면 출력값을 제공한다.</p> <p>채점: 내부에서 테스트 케이스를 넣어서 검증을 한다. 오픈 테스트가 fail했을 경우 정보를 제공하고 히든 테스트가 fail했을 때는 정보를 제공하지 않는다.</p> <p>제출: 제출한 코드의 기능 점수, 오픈소스를 사용하여 계산한 효율,가독성 점수 그리고 문제와 관련된 코드 설명과 관련자료를 출력한다.</p>

화면의 형태

Problem & Constrain

Problem

피보나치 수는 0과 1로 시작하며, 다음 피보나치 수는 바로 앞의 두 피보나치 수의 합이 된다.

$N = 0, 1, \dots$ 에 해당하는 피보나치 수는 $0, 1, 1, 2, 3, 5, \dots$ 이다.

N 번째 피보나치 수를 리턴하시오.

Constrain

$0 \leq n \leq 80$

Testcase

Testcase	Input	Output	결과
Testcase - 1	Solution (0)	0	검증
Testcase - 2	Solution (2)	2	검증

Code Editor

```

1. def load_data_loader(ds, batch_size,
2. shuffle=False):
3.     data_loader = DataLoader(dataset=ds,
4.                               shuffle=True,
5.                               batch_size=batch_size,
6.                               collate_fn=ds.collate)
7.     return data_loader
8.
9.
10.
11.
12.
13.
14.
15.
16.
17.
18.
19.
20.
21.
22.
23.
24.

```

Result

```

(base) User@Terminal: ~ python solution.py
여러줄 통해 왼쪽의 라인을 수정하세요
(base) User@Terminal: ~

```

Problem & Constrain

Problem

피보나치 수는 0과 1로 시작하며, 다음 피보나치 수는 바로 앞의 두 피보나치 수의 합이 된다.

$N = 0, 1, \dots$ 에 해당하는 피보나치 수는 $0, 1, 1, 2, 3, 5, \dots$ 이다.

N 번째 피보나치 수를 리턴하시오.

Constrain

$0 \leq n \leq 80$

Testcase

Testcase	Input	Output	결과
Testcase - 1	Solution (0)	0	검증
Testcase - 2	Solution (2)	2	검증

Code Editor

```

1. def load_data_loader(ds, batch_size,
2. shuffle=False):
3.     data_loader = DataLoader(dataset=ds,
4.                               shuffle=True,
5.                               batch_size=batch_size,
6.                               collate_fn=ds.collate)
7.     return data_loader
8.
9.
10.
11.
12.
13.
14.
15.
16.
17.
18.
19.
20.
21.
22.
23.
24.

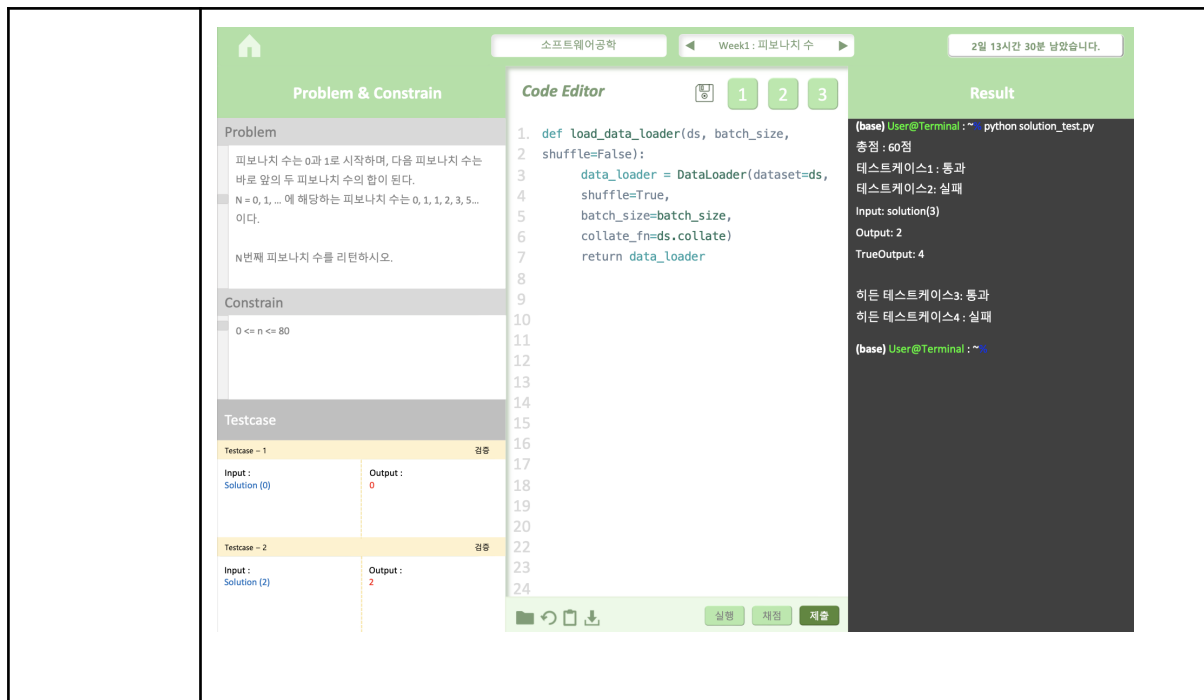
```

Result

```

(base) User@Terminal: ~ python solution.py
여러줄 통해 왼쪽의 라인을 수정하세요
(base) User@Terminal: ~
Traceback (more recent call):
  File <stdin>~

```



3.2. Functional Requirements

3.2.1. 회원가입

[표 6] 기능별 요구사항(1) - 회원가입

이름	회원가입
행위자	사용자
설명	사용자의 아이디가 중복되는지, 비밀번호가 8자리 이상인지 확인해야 한다. 조건에 부합하지 않는다면 회원가입을 불가능하게 해야 한다.
정상 흐름	<ol style="list-style-type: none"> 1. 사용자는 시스템을 이용하기 위해서 반드시 계정을 생성해야 한다. 2. 계정을 생성하기 위해서는 아이디, 비밀번호, 비밀번호 재입력, 사용자의 이름, 이메일을 입력해야 한다. 3-1. 아이디가 중복된 경우 아이디를 다시 입력하라는 메시지를 띄워준다.

	<p>3-2. 비밀번호가 8자리 이상이 아닌 경우 비밀번호를 다시 입력하라는 메시지를 띄워준다.</p> <p>3-3. 비밀번호와 비밀번호 재입력 칸의 문자열이 일치하지 않으면 다시 일치하지 않는다는 메시지를 띄워준다.</p> <p>4. 모든 조건이 만족되면 계정을 생성한다.</p>
조건	<p>1. 아이디는 중복되지 않아야 한다.</p> <p>2. 비밀번호는 8자리 이상이어야 한다.</p>

3.2.2. 로그인

[표 7] 기능별 요구사항(2) - 로그인

이름	로그인
행위자	사용자
설명	사용자의 아이디와 비밀번호를 입력받아 아이디와 비밀번호가 유효한지 확인하고 유효하다면 알맞은 계정에 접속시켜준다.
정상 흐름	<p>1. 사용자는 아이디와 비밀번호를 입력한다.</p> <p>2-1. 아이디가 존재하지 않는다면 유효하지 않은 아이디라는 창을 띄운다.</p> <p>2-2. 아이디는 존재하지만 비밀번호가 아이디와 맞지 않는다면 비밀번호 오류라는 창을 띄운다.</p> <p>3. 아이디와 비밀번호가 모두 일치한다면 해당 계정으로 접속한다.</p>
조건	1. 아이디가 이미 회원가입이 되어있는 상태여야 한다.

3.2.3. 문제 보여주기

[표 8] 기능별 요구사항(3) - 문제 보여주기

이름	문제 보여주기
----	---------

행위자	시스템
설명	코딩 테스트 문제는 화면의 왼쪽에 제공되어야 한다.
정상 흐름	1. 사용자는 왼쪽에서 문제에 대한 설명과 제약사항에 대해 확인할 수 있다.
조건	1. 문제 출제자가 사전에 문제/제약사항을 출제한 상태여야 한다.

3.2.4. 테스트 케이스 보여주기

[표 9] 기능별 요구사항(4) - 테스트 케이스 보여주기

이름	테스트 케이스 보여주기
행위자	시스템
설명	테스트 케이스의 일부는 문제가 제공되는 텍스트 박스 밑의 테스트 케이스 텍스트박스에 보여져야 하며, 나머지 숨겨진 테스트케이는 보여지지 않고 채점할 때에만 사용된다.
정상 흐름	1. 사용자는 왼쪽 아래에서 공개된 테스트 케이스를 확인할 수 있다.
조건	1. 문제 출제자가 사전에 테스트 케이스를 준비한 상태여야 한다.

3.2.5. 코드 하이라이트

[표 10] 기능별 요구사항(5) - 코드 하이라이트

이름	코드 하이라이트
행위자	시스템
설명	코드를 작성하면 해당 언어(python)에 맞게 바로 파싱하여 알맞은 색으로 보여줌으로써 사용자들에게 가독성을 주어야 한다.
정상 흐름	1. 사용자가 코드(python)를 입력하면 알맞는 문법의 색으로

	텍스트를 변환한다. 2. 만약 존재하지 않는 문법이라면 변경하지 않는다.
조건	1. 사용자가 입력하는 언어는 기본 영어이다.

3.2.6. 코드 저장

[표 11] 기능별 요구사항(6) - 코드 저장

이름	코드 저장
행위자	사용자
설명	자신이 적던 코드를 저장할 수 있어야 한다. 코드 저장에 횟수 제한은 없다.
정상 흐름	1. 사용자가 코드를 전부 적고 저장 버튼을 누른다. 2.. 저장 버튼을 누르면 코드가 자동으로 저장된다.

3.2.7. 파일 불러오기

[표 12] 기능별 요구사항(7) - 파일 불러오기

이름	파일 불러오기
행위자	사용자
설명	사용자는 본인이 가지고 있는 .py 파일의 코드를 불러올 수 있어야 한다.
정상 흐름	1. 파일 불러오기 버튼을 선택한다. 2. 사용자의 데스크탑에서 해당 파일을 선택할 수 있는 창이 띄워진다. 3. 사용자가 해당 파일을 선택 후 불러오기를 누른다.
조건	1. 파일 형식은 .py이어야 한다.

3.2.8. 코드 초기화

[표 13] 기능별 요구사항(8) - 코드 초기화

이름	코드 초기화
행위자	사용자
설명	사용자는 자신이 현재 작성하고 있는 코드를 전부 삭제할 수 있어야 한다.
정상 흐름	1. 코드 초기화 버튼을 누른다. 2. 현재 작성하고 있는 코드가 전부 삭제되고 skeleton code만 남게 된다.

3.2.9. 코드 복사

[표 14] 기능별 요구사항(9) - 코드 복사

이름	코드 복사
행위자	사용자
설명	사용자는 본인이 작성하고 있는 코드를 복사할 수 있다.
정상 흐름	1. 코드 복사 버튼을 선택한다. 2. 현재 사용자가 작성하고 있는 코드 전체를 복사한다.

3.2.10. 코드 다운로드

[표 15] 기능별 요구사항(10) - 코드 다운로드

이름	코드 다운로드
행위자	사용자

설명	사용자는 본인이 작성하고 있는 코드를 .py파일로 다운 받을 수 있어야한다.
정상 흐름	1. 코드 다운로드 버튼을 선택한다. 2. 현재 작성하고 있는 코드를 어디에 다운받을지 선택하는 창을 띄워준다. 3. 사용자가 위치를 선택하고 다운로드 버튼을 누르면 작성하고 있는 코드를 .py파일로 다운받는다.

3.2.11. 코드 실행

[표 16] 기능별 요구사항(11) - 코드 실행

이름	코드 실행
행위자	사용자
설명	사용자는 터미널을 통해 본인이 작성하고 있는 코드를 실행시켜볼 수 있다.
정상 흐름	1. 사용자가 코드 실행 버튼을 누른다. 2-1. 코드 실행 결과가 에러 아닐 경우 정상적으로 결과값을 출력한다. 2-2. 코드 실행 결과가 에러일 경우 에러 메시지를 이용하여 에러가 난 코드의 위치를 사용자에게 하이라이트 표시로 보여준다. 또한 사용자에게 에러 메시지를 제공한다.
조건	1. 파일 형식은 .py이어야 한다.

3.2.12. 코드 채점

[표 17] 기능별 요구사항(12) - 코드 채점

이름	코드 채점
----	-------

행위자	사용자
설명	사용자는 코드를 채점하여 점수, Pass/Fail 여부를 확인할 수 있다. 채점의 경우 숨겨진 테스트 케이스까지 전부 테스트한다.
정상 흐름	1. 사용자가 코드 채점 버튼을 누른다. 2. 모든 테스트 케이스를 적용하여 채점한다. 3-1. 만약 사용자에게 공개된 테스트 케이스가 fail하였다면 테스트 케이스 정보를 제공한다. 3-2. fail한 테스트 케이스가 숨겨진 테스트 케이스라면 정보를 공개하지 않는다. 4. “성공한 테스트 케이스 / 전체 테스트 케이스” 를 계산하여 점수를 출력한다. 5. 점수에 따른 pass/fail여부를 출력한다.
조건	1. 출제자가 사전에 테스트 케이스를 준비해 놓아야 한다.

3.2.13. 코드 제출

[표 18] 기능별 요구사항(13) - 코드 제출

이름	코드 제출
행위자	사용자
설명	사용자는 코드를 출제자에게 제출한다. 또한 사용자는 과거에 제출한 코드를 다시 불러올 수 있다.
정상 흐름	1. 사용자가 코드 제출 버튼을 누른다. 2. 제출한 코드가 출제자가 준비한 정답 코드와 얼마나 다른지 비교하여 보여준다. 3. 제출 후 사용자가 과거에 제출한 코드와 비교할 수 있고, 과거에 제출한 코드를 불러와서 수정한 후 재제출할 수 있다.
조건	1. 출제자가 사전에 준비한 정답 코드가 있어야 한다.

3.2.14. 코드 결과 분석

[표 19] 기능별 요구사항(14) - 코드 결과 분석

이름	코드 결과 분석
행위자	시스템
설명	<p>제출된 코드의 표절 여부를 검사한다.</p> <p>코드의 결과 분석은 3 type으로 나누어서 제공된다.</p> <p>1. 기능 점수 : 테스트 케이스를 얼마나 맞추었는지를 평가한다.</p> <p>2. 효율 점수 : LOC, Halstead, Data flow, Control flow 항목을 채점한다.</p> <p>3. 코드 가독성 점수 : 코드 가독성을 채점한다.</p> <p>또한 OpenAI Codex API를 이용한 코드 설명을 제공한다. 또한 관련된 자료와 문제, 영상 등을 추천해준다.</p>
정상 흐름	1. 코드가 제출되면 오른쪽에 코드 결과를 3 type으로 나누어서 보여준다.

3.3. Non-functional Requirements

3.3.1. Product Requirements

본 시스템이 작동하는 도중 지켜져야 하는 요구조건들을 말한다.

3.3.1.1. Usability Requirements

본 제품은 사용자 친화적으로 디자인 되어 별도의 설명 없이도 사용자가 UI만 보고 제품을 어떻게 사용해야 할지 예측할 수 있어야 한다. 현재 본 시스템은 접근성이 좋은 브라우저 기반의 웹 어플리케이션으로 개발되었다. 우수한 접근성을 유지하기 위해 사용자가 본 시스템을 사용할 때 별도의 설치 과정이나 조건을 요구하는 것을 지양한다. 또한, 사용자의 오류를 최소화하도록 구성되어야 한다. 전문적인 용어의 사용을 최소화해야 하고, 이해하기 쉽게 설명해야 한다. 각 사용자는 별도의 매뉴얼 없이 모든 시스템 기능을 직관적으로 사용할 수 있어야 한다. 만약 충분한 배경지식이 있어야만 이해할 수 있는 내용이라면, 시스템 관리자가 배경지식에 해당하는 교육을 받을 수 있도록 교육 자료를 제공해줘야 한다.

3.3.1.2. Performance Requirements

본 시스템은 시간당 최대 100명의 동시접속을 허용하도록 한다. 만약 서비스의 규모가 더 확장될 경우 하드웨어, 소프트웨어 측면에서 개선이 필요하다. 위 조건이 충족된다면 페이지 렌더링을 포함한 본 시스템에서 제공하는 모든 서비스를 5초 이하의 응답 시간 내에서 실행될 수 있도록 시스템을 설계한다.

3.3.1.3. Dependability Requirements

본 제품은 시스템 구성요소들이 결합하였을 때 화면이 깨지거나 오류가 나는 일 없이 부드럽게 상호작용해야 하며, 본 문서에서 제한한 환경요건만 충족한다면 어떠한 장치에서 접근하더라도 화면 구성이 깨지지 않도록 UI를 구성해야 한다.

3.3.1.4. Security Requirements

시스템의 보안을 위해 등록된 사용자가 아니면 해당 시스템에서 제공하는 기능을 사용할 수 없도록 해야 한다. 또, 사용자들의 개인 정보, ID, 비밀번호와 관련된 정보는 권한이 있는 사용자만이 접근할 수 있다. 다음으로, 사용자는 각자의 직급에 맞는 행동만 할 수 있도록 설계한다. 교수자와 시스템 관리자를 제외한 학습자 신분의 사용자는 개인의 코드만 확인할 수 있으며 다른 학습자의 코드에 접근할 수 없다.

본 시스템은 외부로부터의 공격도 방어해야 한다. 본 서비스 이용 도중, 사용자는 서버에 임의의 데이터를 보낼 수 있다. 그 과정은 TCP 프로토콜을 기반으로 충분히 안전하지만, MITM 중간자 공격을 고려하여 사용자 데이터는 비식별화하여 전송한다. 사용자의 정보보호 이외에도, 웹 서비스는 각 사용자에게 의해 데이터 유출 공격의 대상이 될 수 있다. 이러한 상황을 방지하기 위해 서버는 사용자가 전송하는 어떠한 HTTP 요청의 값도 신뢰하지 않고 검증한다. 또한 관리자 권한이 필요한 모든 동작에 대하여 세션 및 쿠키 기반으로 올바른 접근을 수행하고 있는지 확인한다.

3.3.2. Organizational Requirements

본 시스템의 개발자, 사용자, 운영자가 지켜야 할 규칙이나 절차에 의해 발생하는 요구사항을 의미한다.

3.3.2.1. Operational Requirements

본 제품은 Windows7 이상의 운영체제 환경에서 실행되어야 한다. 또한 원활한 접속 및 서비스 제공을 위해 Chrome 브라우저 사용을 권장한다.

3.3.2.2. Environment Requirements

본 제품은 웹 어플리케이션 서비스로, 프로그래밍 학습 환경을 제공하는 것을 목표로 하기에 데스크탑이나 노트북 사용 환경만을 지원한다. 사용자는 인터넷에 연결된 상태에서 서비스를 제공받을 수 있으며, 크롬을 사용한 학습을 권장한다.

본 시스템의 최종 라이선스인 Apache2 라이선스에 위배되거나 충돌하지 않는 한, 개발자는 어떠한 오픈 소스 API든 활용할 수 있다. 단, 이 경우 본 요구사항 명세서를 적절히 수정하여야 한다.

3.3.3. External Requirements

시스템 자체적으로 발생하거나 시스템 운영 및 개발에 있어 필요한 요구사항이 아닌 외부적으로 발생하여 준수해야 하는 요구사항을 서술한다.

3.3.4. Regulatory requirements

시스템이 제공하는 문제의 정답 코드의 출처를 명시하고 저작권을 확인해야 한다. 사용자의 정보 공개 범위는 사용자 본인과 관리자에 한정된다. 회원의 개인정보가 필요한 경우에는 사용자로부터 정보

제공 동의를 얻은 후 진행해야 한다. 사용자가 자신의 정보 삭제를 원할 경우 시스템 관리자에게 문의하여 삭제가 이루어질 수 있으나, 서비스 이용은 제한된다. 또한 본 시스템에서 이용하는 외부 API의 라이선스를 철저히 지켜야 한다. 만일 라이선스가 충돌할 경우 대안을 찾아야 한다.

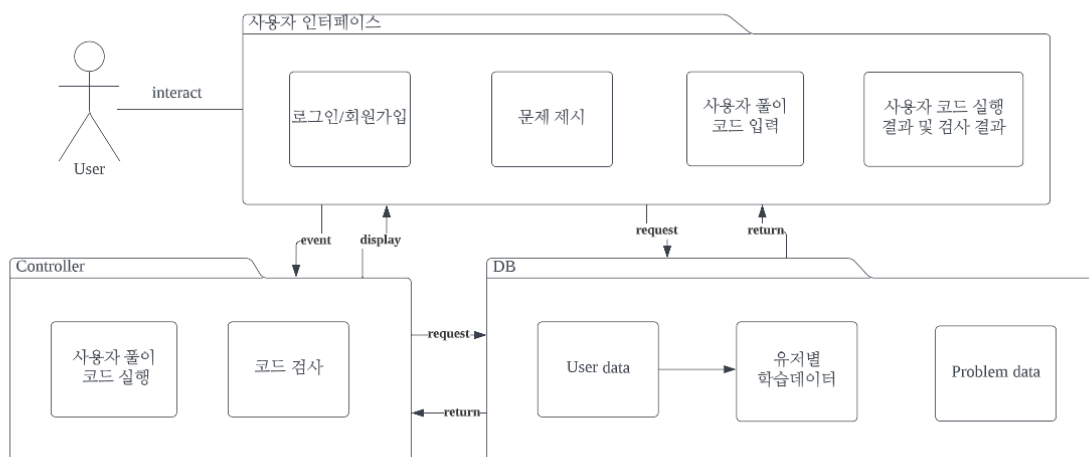
3.3.4.1. Safety/Security Requirements

개인정보보호법에 따라 사용자의 개인정보가 외부/내부의 공격으로부터 안전하게 보호될 수 있도록 해야 한다.

4. System Models

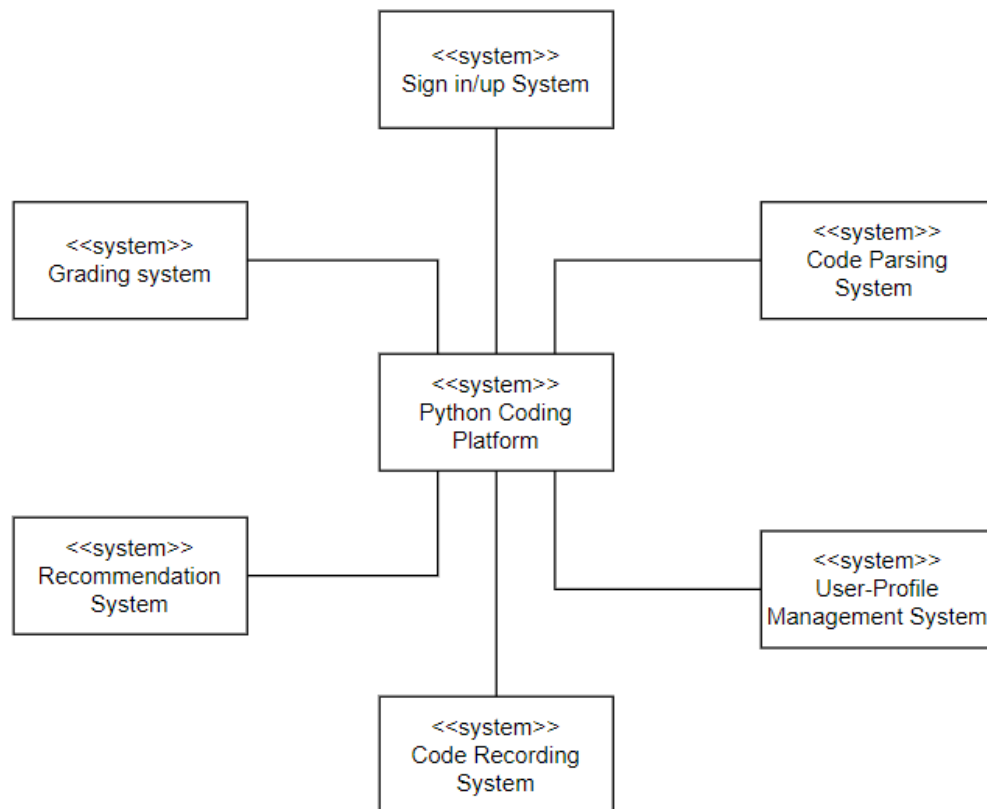
4.1. System Architecture

[그림 2] Overall System Architecture



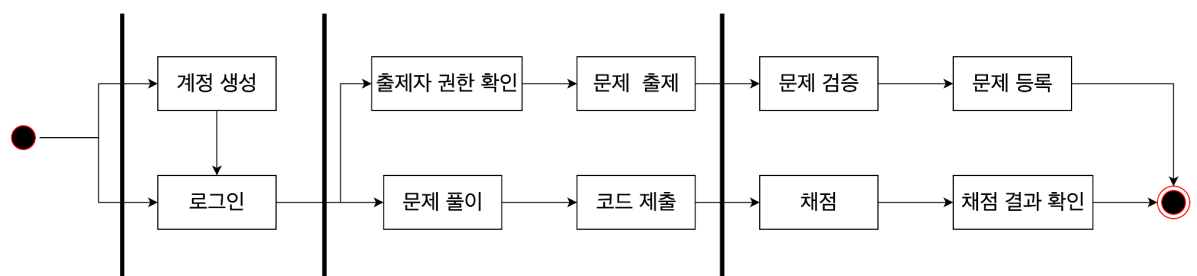
4.2. System Context Diagram

[그림 3] Overall System Context Diagram



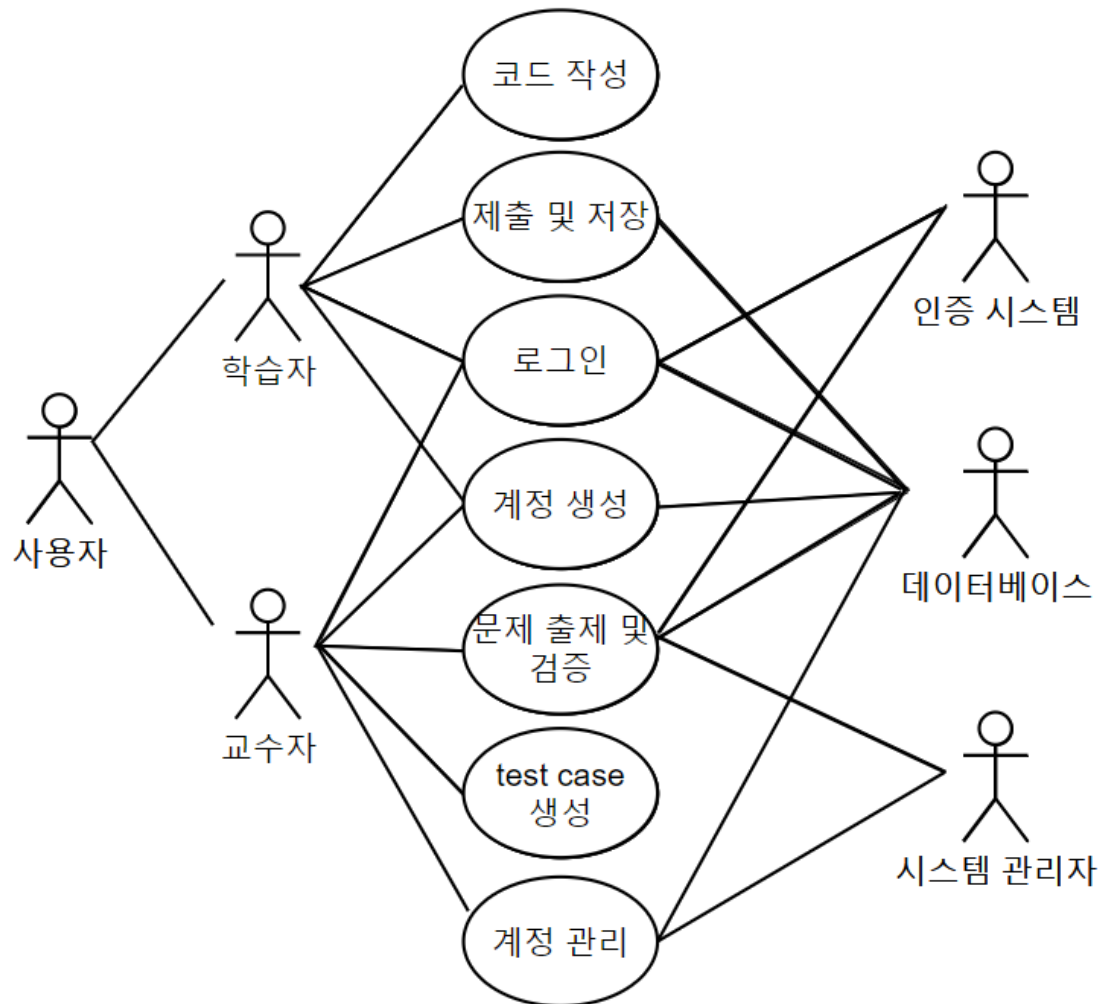
4.3. Process Model

[그림 4] Overall Process Model Diagram

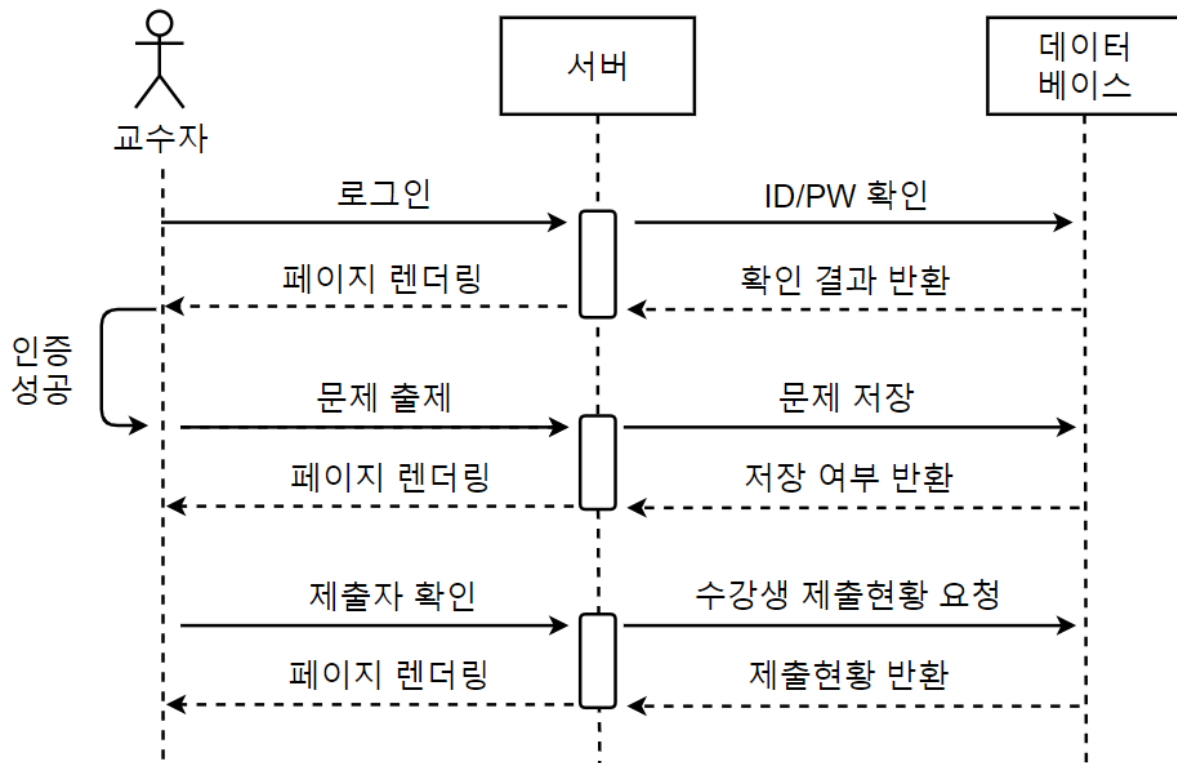


4.4. Interaction Model

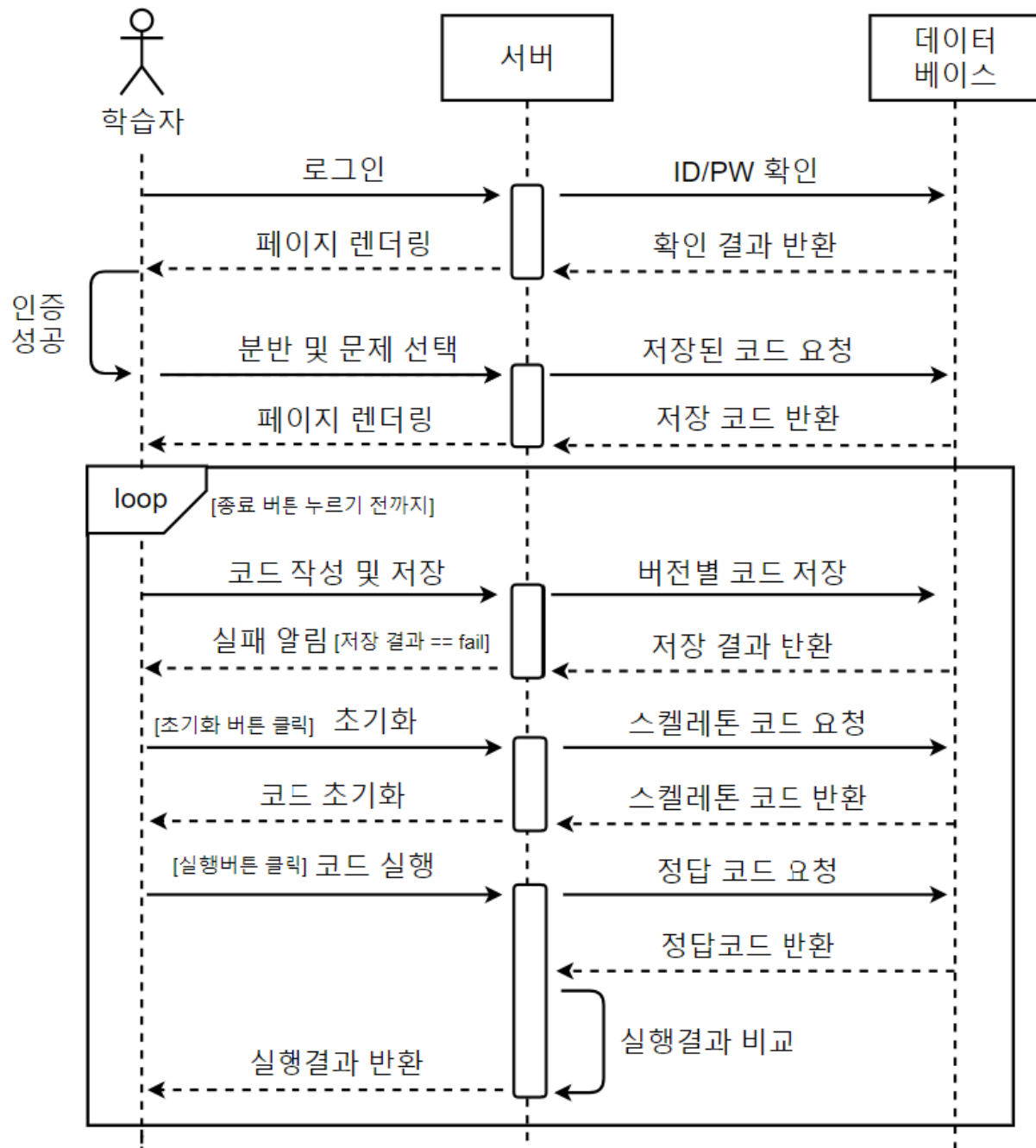
[그림 5] Usecase Diagram



[그림 6] Sequence Diagram for the Lecturer

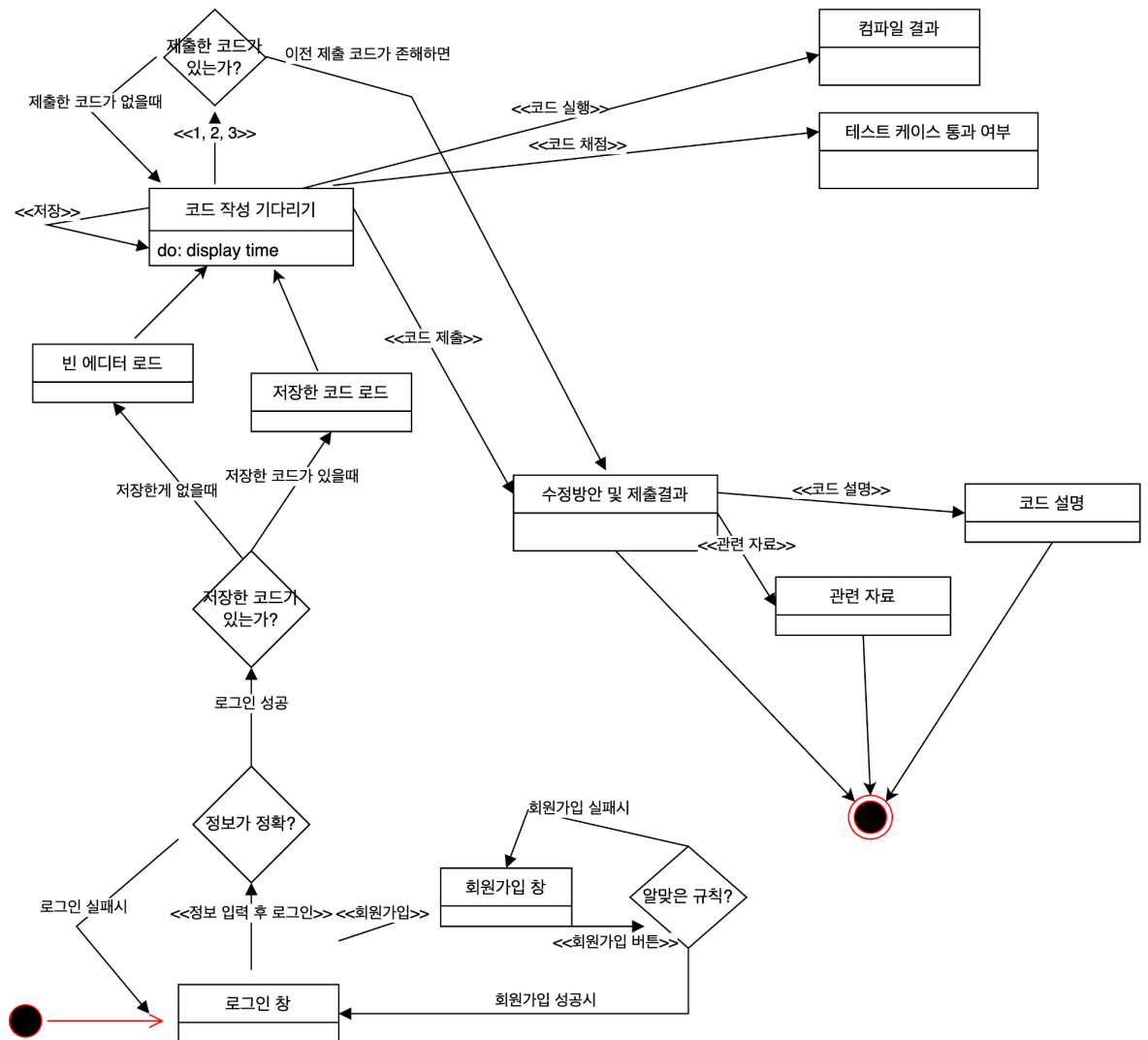


[그림 7] Sequence Diagram for the Students



4.5. Behavior Model (State diagram)

[그림 8] Overall State Diagram



5. System Evolution

본 장에서는 현 시스템의 한계와 이로 인해 발생할 수 있는 소프트웨어의 개선, 또는 하드웨어의 개선이나 시스템이 사용되는 환경의 변화로 인해 발생하는 새로운 요구사항을 예측해보고 이에 대한 대응방안을 제시한다. 이는 요구사항의 변경에 대한 기본적인 입장을 서술한 것이기 때문에 시스템 디자이너는 이를 고려하여 시스템을 설계하여야 한다.

5.1. Limitation and Assumption

본 서비스는 브라우저 기반으로 사용자가 과목별 코딩 과제를 수행할 수 있는 환경을 제공한다. 출시하는 시점에 본 제품은 Python 프로그래밍 언어만 지원하고 있다. Python 프로그래밍 언어가 대세를 이루고 있고 다양한 분야에서 쓰이기 때문에 여러 수업에서도 Python을 필수로 가르치는 추세라 시스템 이용에 불편함은 없을 것이다. 하지만 분명 다른 프로그래밍 언어를 사용하고자 하는 사용자들이 있을 것이기에 다른 언어를 지원하지 않는다는 한계점이 있다.

또, 본 제품에서 제공하는 문제는 교수자가 제공하는 문제와 자체 제작 혹은 타 플랫폼에서 제공하는 것으로 구성되어 있다. 어디까지나 본 제품은 교수자의 수업내에서 요구하는 과제를 풀고 채점하여 제출하는 기능을 제공하는 것이 주 목적이기에 자체 제작한 문제의 수가 타 플랫폼에서 제공하는 수에 비해 부족하다는 한계가 있다.

5.2. Change of User Requirements

사용자층이 확대됨에 따라 다양한 언어를 지원해줘야 한다. 따라서 학교 수업에서 많이 쓰이거나 개발에 자주 사용되는 언어, 가령 C, C++, Java, Javascript와 같은 언어부터 점차적으로 지원을 확대한다. 또, 학생 사용자들이 다양한 문제를 풀고 연습할 수 있는 환경을 제공하기 위해 교수자가 제출한 문제 이외에도 PS 유형별로 다양한 문제를 풀 수 있도록 문제를 지속적으로 추가할 수 있다. 마지막으로 교수자가 학생의 코드를 평가하는 데 있어 실행시간과 같은 지표를 사용할 수 있고, 학생들이 문제를 푸는 데 있어 토론 기능의 필요성을 느낄 수 있기에 실행시간 측정기능 추가 및 학생 커뮤니티 공간을 추가할 수도 있다.

한편, 앞서 예측한 것 이외에 사용자 요청 사항이 발생할 경우, 만일 새로운 기능을 개발하는 것이 아니라 단순히 내용물의 추가라면 시스템 관리자 선에서 해당 내용을 추가할 수 있다. 새로운 기능을 개발해야 하는 경우, 본 요구사항 명세서에 있는 내용들을 기반으로 새로운 API를 추가하거나 새로운 웹 서비스 페이지를 설계할 수 있다. 단 각 설계 단계에 있어 기존 요구사항 명세와 통일된 양식을 가져야 하며, 작성 후 본 문서 또한 수정되어야 한다.