



온라인 코딩 테스트 웹 플랫폼

- 프로그래머스쿠 -

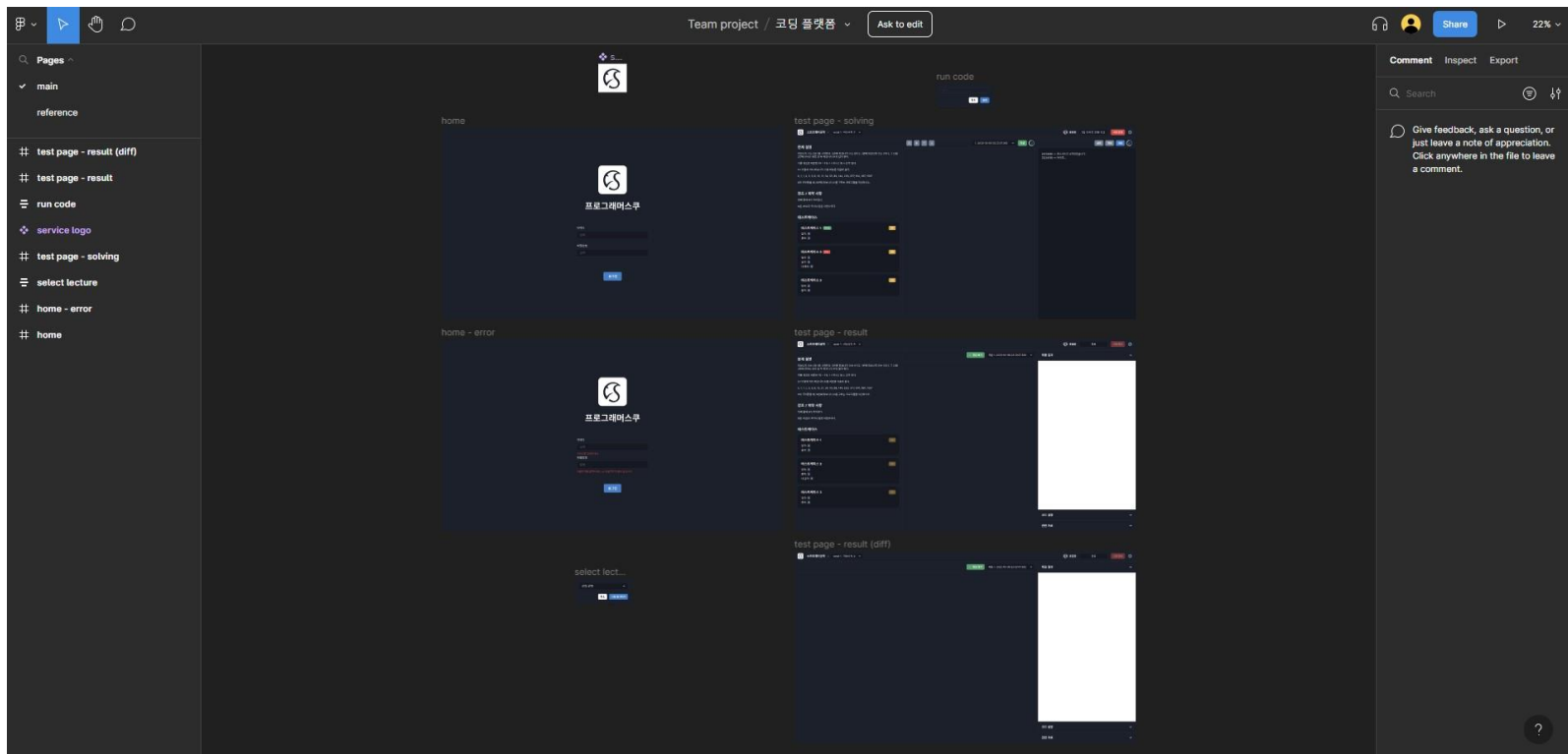
소프트웨어공학개론 TEAM 7
김양선, 김호진, 변영민, 이주빈, 임세아, 전윤태



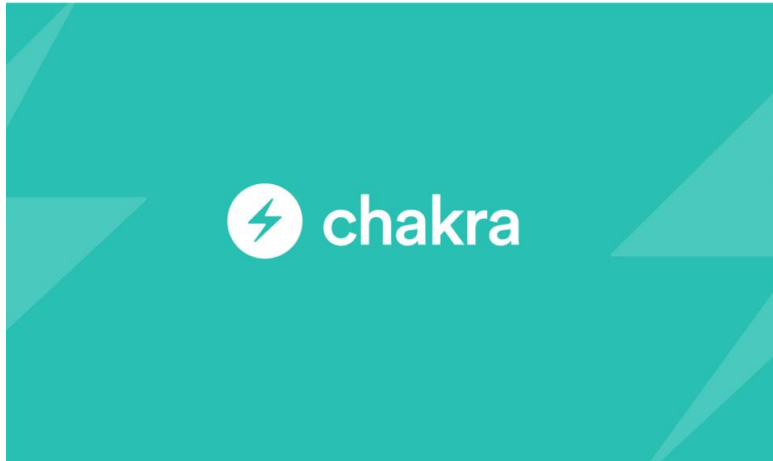
**프로그래머스쿠(Programmerskku)는
프로그래머스에서 영감을 받은
파이썬 코딩 테스트 플랫폼입니다.**

사용자 친화적인 디자인

- Figma를 이용한 디자인 시안 제작



Chakra UI

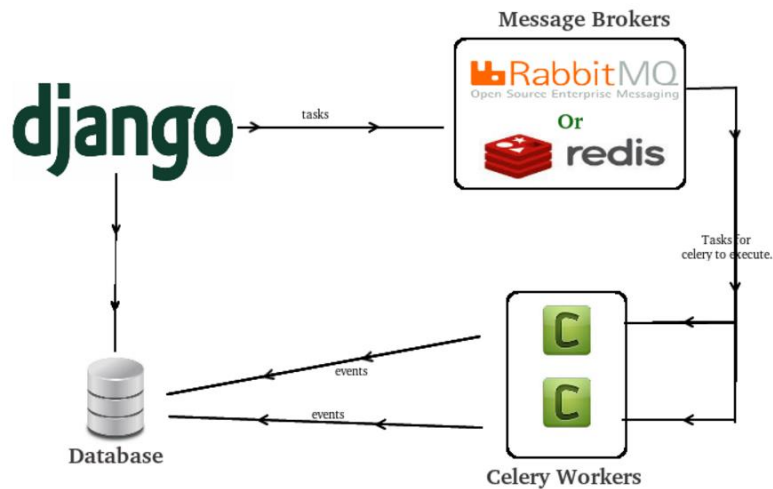


- Chakra UI는 React 위에 구축되어 개발자가 웹 어플리케이션을 위한 액세스를 가능하게 하고, 재사용 가능한 UI 컴포넌트를 빠르게 구축할 수 있도록 돕는 컴포넌트 툴킷이다.
- Chakra UI의 목표는 개발자가 최소한의 노력으로 아름답고, 반응이 빠르며, 접근 가능한 사용자 인터페이스를 만들 수 있도록 하는 것이다. 해당 툴킷에는 쉽게 확장할 수 있게 설계된 다양한 컴포넌트가 포함되어 있다.

왜 Chakra UI를 선택했나?

직접 사용해본 결과, 컴포넌트마다 기본적으로 제공하는 props가 다양하고, 이를 사용한 예시가 많아 활용하기에 편리했다. 또한 토스트, 아코디언 등과 같이 Chakra에 이미 구현된 컴포넌트의 경우 직접 구현할 필요가 없을 뿐더러 기본 UI 자체도 깔끔해서 디자인 구현에 많은 도움이 됐다.

비동기 처리



- POST/submissions API 호출 시 DB에 submission 레코드를 생성한다.
- 이와 동시에 celery라는 **비동기 태스크 큐**에 submission이 생성되었다는 이벤트를 보내고, analyze_submission 함수를 트리거한다.
- 해당 함수에서 코드를 분석한 결과를 DB에 저장하고, 필요시 프론트에 전달한다.

왜 이런 아키텍처를 선택했나?

코드 제출을 처리하는 함수에서 모든 분석을 처리하게 되면 분석이 오래 걸릴수록 제출에 대한 응답이 늦어지게 된다. 코드를 제출하는 API는 제출물이 DB에 저장만 되면 제 역할을 다한 것이기 때문에 바로 응답을 하도록 하고, 제출 코드의 분석은 별개로 동작해야 된다고 생각해서 이러한 방식을 사용하여 분리하게 되었다.

개발 환경

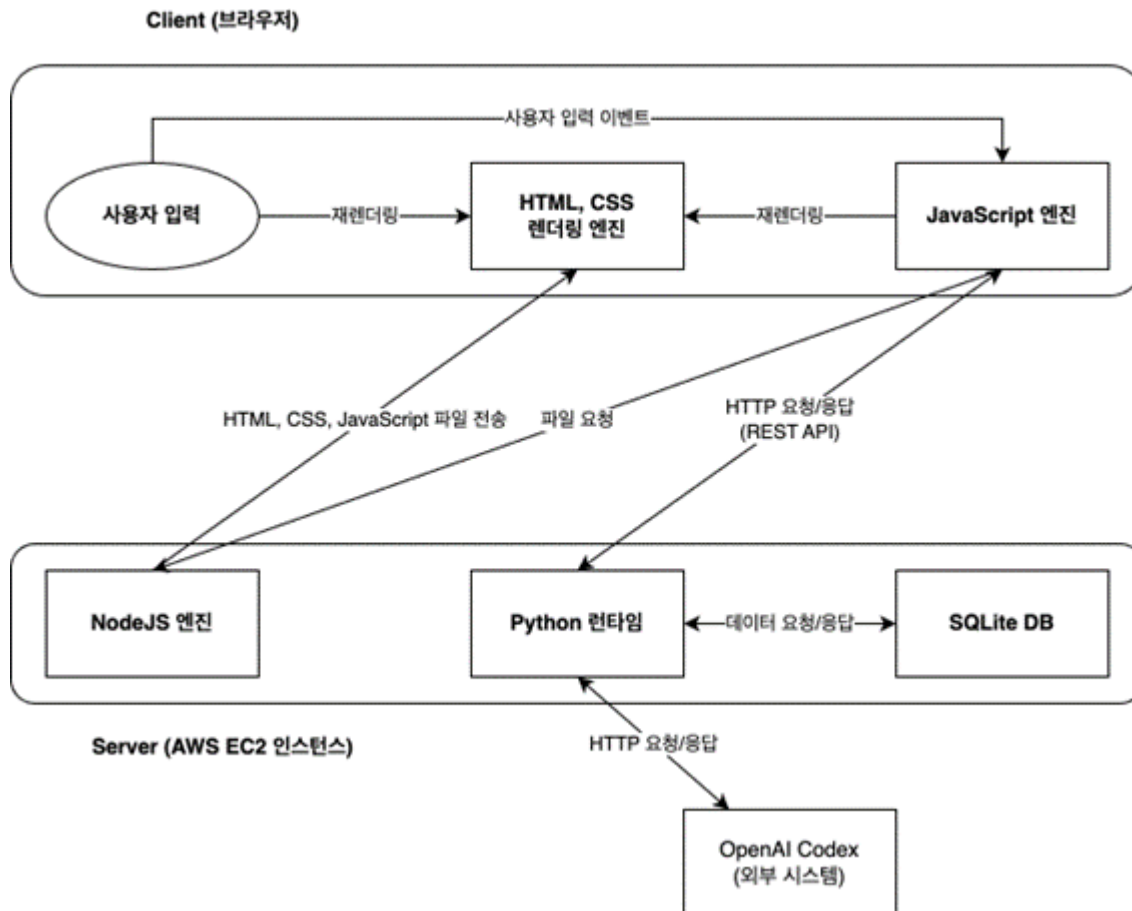
- Frontend

카테고리	기술명
언어	JavaScript (16)
프레임워크	React (18.2.0)
컴포넌트 라이브러리	Chakra-UI (2.3.6)

- Backend

카테고리	기술명
언어	Python (3.9)
프레임워크	Django (4.1), Django REST Framework (3.14)
데이터베이스	SQLite (3)

시스템 구조



- 본 서비스는 클라이언트 - 서버 모델을 적용하여 설계되었다.
- Frontend 애플리케이션은 사용자와의 모든 상호작용을 담당한다.
- Frontend 애플리케이션과 Backend 애플리케이션은 JSON 기반의 HTTP 통신을 통해 데이터를 주고받는다.
- Backend 애플리케이션은 유저의 요청을 Frontend에서 컨트롤러로 배포하고, 데이터베이스에서 필요한 객체 정보를 가져온다.
- Backend 애플리케이션은 데이터베이스에서 요청한 내용을 처리한 이후 JSON 형식으로 값을 전달한다.

단계별 기능 명세

1 단계

가장 기본적인 기능과 전체적인 UI 레이아웃을 구현하는 단계
프론트엔드와 백엔드의 연동 과정에 익숙해지는 과정

- 사용자 인증 (로그인) - Session Authentication
- 홈페이지
- 문제
 - 문제설명 및 참조/제약사항
- 테스트케이스
 - 공개 테스트케이스 표기
 - 테스트케이스 검증
 - 클릭시 output 여부에 따라 pass/fail 결과 제공
 - Fail일 경우 학생 코드의 output 결과도 같이 제공
- 코드 에디터
 - 코드 에디터 구현 (오픈소스 활용)
 - 코드 라인 (숫자)
 - 코드 작성
 - 코드 복사 (클립보드 복사)
 - 파일 불러오기
 - .py 파일만 가능하게 필터링
 - 코드 되돌리기 (skeleton 코드로 초기화)
 - 코드 다운로드 (파일로 저장)

2 단계

복잡도가 있는 기능까지 모두 구현하여 필수 요구 사항은 모두 충족하는 단계
테미널과 연관된 기능들은 모두 해당 단계에서 진행

- 코드 에디터
 - 코드 실행
 - Modal UI로 입력값 받음 (oncode 처리)
 - 코드 중간 저장
 - Dropdown UI 사용
 - 실행 에러 처리
 - 에러 라인 하이라이트
 - 에러 메시지 출력
 - popover UI 사용
- 채점
 - 점수제공
 - Pass/Fail 여부 제공
 - 채점 시 히든 테스트케이스까지 모두 테스트
 - 오픈 테스트케이스가 fail 했을 경우 → 테스트케이스 정보 제공
 - 히든 테스트케이스가 fail 했을 경우 → 테스트케이스 정보 미제공
- 제출
 - 최대 n번 횟수 제한
 - 잔여 횟수 표기
 - 마감일 이후 결과 분석 표기
 - 검사
 - 표절 검사
 - 기능 채점
 - 효율 채점
 - 가독성 채점
 - 과거제출코드
 - 다시 푼 n번의 코드 불러오기
 - 코드 Diff
 - 코드 입력창 확대 시 좌우로 표기
 - 코드 설명
 - OpenAI Codex API 사용
- 추천
 - 관련 자료 추천

- Notion을 활용한 전반적인 개발 일정 관리 및 팀 단위 회의 진행
- 주차별 To-Do List를 작성하여 완성도 있는 개발 진행

공지 게시판

- 이번주 할 일 (13주차)
 - 프로트 UI 조정 2차
 - 1단계 API 연결
- submissions API 구현

일정표

📅 타임라인 🗂 보드 👤 표

2022년 9월																														11월							12월		
30	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	1	2	3	4	5				
팀 규칙 정하기 [완료] → Done																																							
기술영역 정리 [완료] → Done																																							
사용할 라이브러리 선정 [완료] → Done																																							
프로젝트/백엔드 준비팅 [완료] → Done																																							
디자인 시간 채워 [완료] → Done																																							
GitHub 세팅 [완료] → Done																																							
프로젝트 생성 및 세팅 [프로토인도] → Done																																							
DB 스키마 설계 [백엔드] → Done																																							
프로젝트 생성 및 세팅 [백엔드] → Done																																							
API 설계 [백엔드] → Done																																							
프로토타입 UI 구현 [프로토인도] → Done																																							
Home 컴포넌트 Chakra-ui 적용 [백엔드] → Done																																							
axios 세팅 [프로토인도] → Done																																							
API 가발 [백엔드] → Done																																							
UI 수정 [프로토인도] → Done																																							
1차 API 연결 [프로토인도] → Done																																							
1단계 API 서버 조정 [백엔드] → Done																																							
storages API 구현 [백엔드] → In Progress																																							
UI 수정 2차 [프로토인도] → Done																																							
submissions API 구현 [백엔드] → In Progress																																							
프로토타입 QA [프로토인도] → In Progress																																							
2차 API 연결 [프로토인도] → Backlog																																							
시간 여유에 맞춰 [완료]																																							
방과후 안습기 [Backlog]																																							
최종 발표 [Backlog]																																							

조직화된 역할 분담

- Notion을 활용한 조직적인 역할 분담
- 구체적인 작업 명세를 통해 원활한 Frontend와 Backend의 병행 개발 진행

소문개 / API 문서

표

method	path	description	속성	stage	구현 담당...	재확인 ...	내 연동 담당자
POST	/login/	로그인 및 유저 정보 반환		1단계	김호진	✓	김양선
GET	/logout/	로그아웃	인증 필요	1단계	변영민	✓	임세아
							김호진
GET	/problems/id/	문제 정보 반환	인증 필요	1단계	김호진	✓	김양선
GET	/lectures/	강의 목록 반환	인증 필요	1단계	전준태	✓	임세아
GET	/lectures/id/	강의 정보 반환	인증 필요	1단계	전준태	✓	김양선
POST	/lectures/id/enroll/	강의 등록 신청	인증 필요	1단계	전준태	✓	전준태
POST	/lectures/id/end/	강의 종료	인증 필요	2단계	변영민		
POST	/execute/	코드 실행	인증 필요	1단계	변영민	✓	이주빈
POST	/grade/	코드 채점	인증 필요	1단계	변영민	✓	전준태
							김호진
GET	/storages/	코드 저장소에 불러오기	인증 필요	2단계	변영민	✓	전준태
POST	/storages/	코드 저장소의 코드 저장/수정	인증 필요	2단계	변영민	✓	전준태
POST	/submissions/	코드 제출	인증 필요	2단계	변영민		
GET	/submissions/id	제출 결과 반환	인증 필요	2단계	변영민		
					전준태		
					김호진		
GET	/sample/	형식 안내를 위한 샘플 문서					

+ 새로 만들기

동작 설명

- 모든 테스트케이스에 대해 각각 해당 API 호출 (하나로 통합하면 지연시간 너무 김)
- 테스트 결과 반환
- 단, 공개/히든 테스트케이스 구분

Request

- Path Params
 - 없음
- Query Strings
 - problem_id : int (문제 번호, 1부터 시작)
 - testcase : int (테스트케이스 번호, 1부터 시작)
- Headers
 - Content-Type : application/json
 - X-CSRFToken
- Body

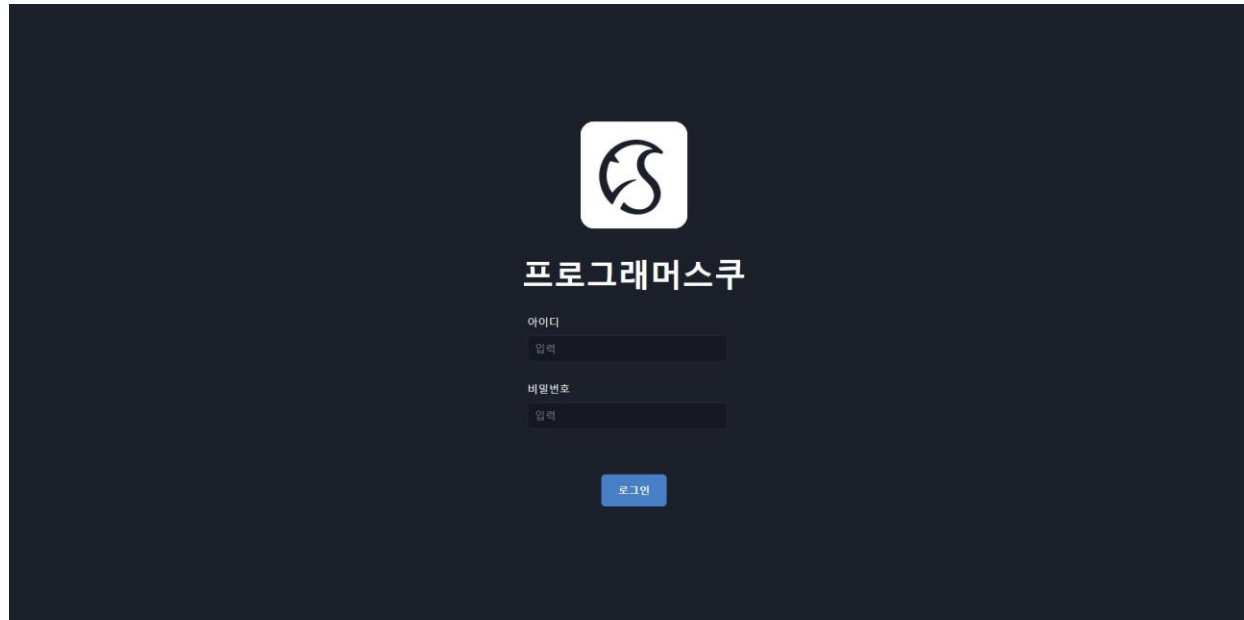
```
{
  "code": string // 실행할 코드 문자열
}
```

Response

```
200

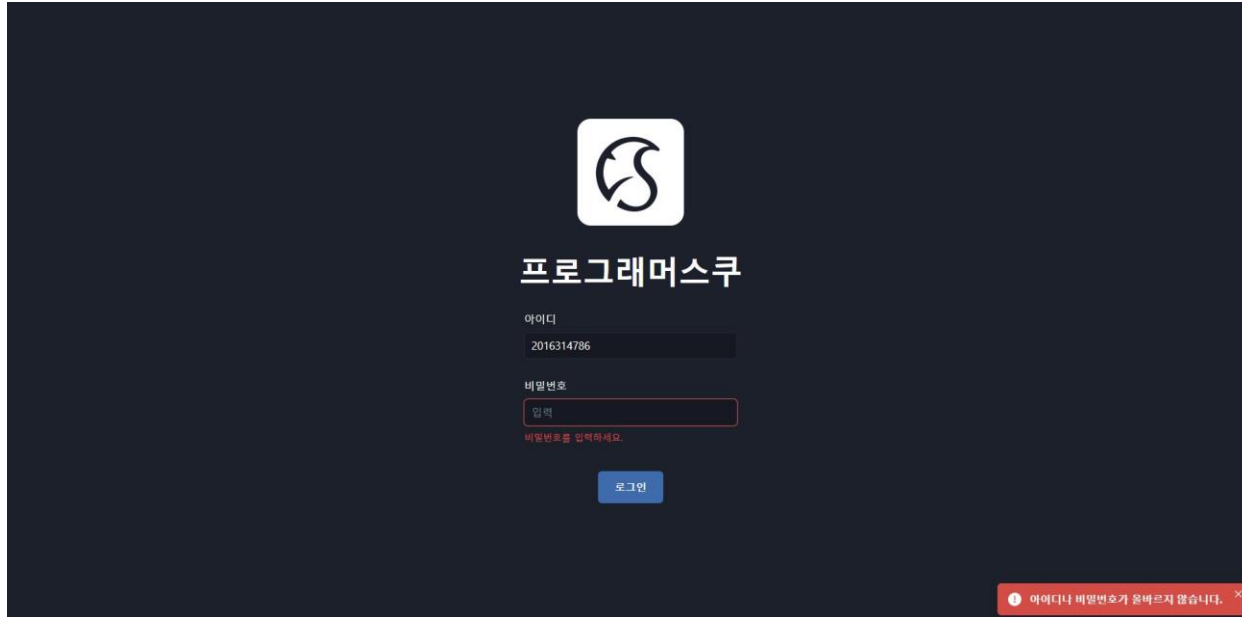
{
  "is_passed": true // 통과 여부
  "input": string | null, // 테스트케이스 입력 (히든 테스트 케이스의 경우 null)
  "output": string | null, // 테스트케이스 출력 (히든 테스트 케이스의 경우 null)
  "result": string | null // 실행 출력 결과 (히든 테스트 케이스의 경우 null)
  "error": string | null, // 에러 출력 결과 (히든 테스트 케이스의 경우 null)
  "error_line": number | null // 에러 일어난 줄 번호
}
```

로그인 페이지



- 다수의 사용자가 동시에 서비스를 이용할 수 있도록 로그인 기능 구현
- Backend에서 페이지에 입력된 학번과 비밀번호를 검증한 이후 비밀번호를 제외한 유저 정보 반환
- Set-Cookie 헤더로 오는 sessionid와 csrftoken을 저장한 뒤 인증이 필요한 모든 요청에 해당 값을 넣어 전송

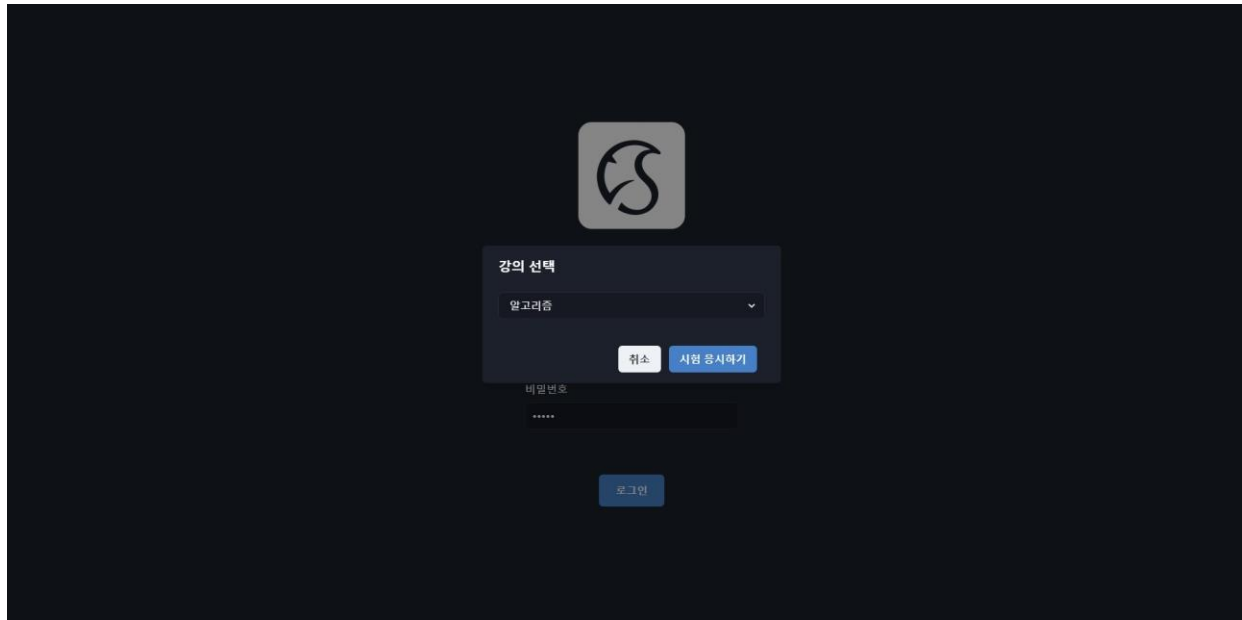
로그인 페이지 (로그인 실패)



The screenshot shows the login interface for '프로그래머스쿠' (Programmer's School). It features a dark blue background with a white logo at the top center. Below the logo, the text '프로그래머스쿠' is displayed. The login form includes two input fields: '아이디' (ID) with the value '2016314786' and '비밀번호' (Password) with the placeholder '입력' (Input). A red border highlights the password field, and a red error message '아이디나 비밀번호가 올바르지 않습니다.' (ID or password is incorrect) is shown below it. A blue '로그인' (Login) button is positioned below the form. At the bottom right, a red notification bar displays the message '아이디나 비밀번호가 올바르지 않습니다.' (ID or password is incorrect).


- 아이디나 비밀번호를 입력하지 않았거나, 잘못 입력한 경우에는 사용자에게 에러메시지 제공



강의 선택 페이지



- 로그인이 성공적으로 진행되면, 강의 선택 페이지 생성
- 반드시 강의를 선택하고, enroll API가 성공적으로 호출되어야만 테스트페이지로 이동 가능

결과 페이지


소프트웨어공학개론 > 피보나치수


변영민
시험 종료


문제설명

피보나치 수는 0과 1로 시작한다. 0번째 피보나치 수는 0이고, 1번째 피보나치 수는 1이다. 그 다음 2번째 부터는 바로 앞 두 피보나치 수의 합이 된다. 이를 식으로 써보면 $F_n = F_{n-1} + F_{n-2}$ ($n \geq 2$)가 된다. $n=17$ 일때 까지 피보나치 수를 써보면 다음과 같다. 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597. n 이 주어졌을 때, n 번째 피보나치 수를

참조 / 제약 사항

입력첫째 줄에 n 이 주어진다. n 은 45보다 작거나 같은 자연수이다. 출력첫째 줄에 n 번째 피보나치 수를 출력한다.

테스트케이스

테스트케이스 1
간증

입력: 0
 출력: 0


테스트케이스 2
간증

입력: 1
 출력: 1

수정 방안

```
1-n = int(input())
1+answer code
```

재출 결과
표절율: 30%



Category	Score
score	50
efficiency	100
readability	100

기능 점수 확인

효율 점수 확인

가독성 점수 확인

- 테스트케이스 - 1: 통과
- 테스트케이스 - 2: 실패

Input: 0
 Output: 1
 Your Output: 0

코드 설명

관련 자료

지금부터는 시연을 진행하겠습니다
감사합니다