자동 문제 생성 시스템

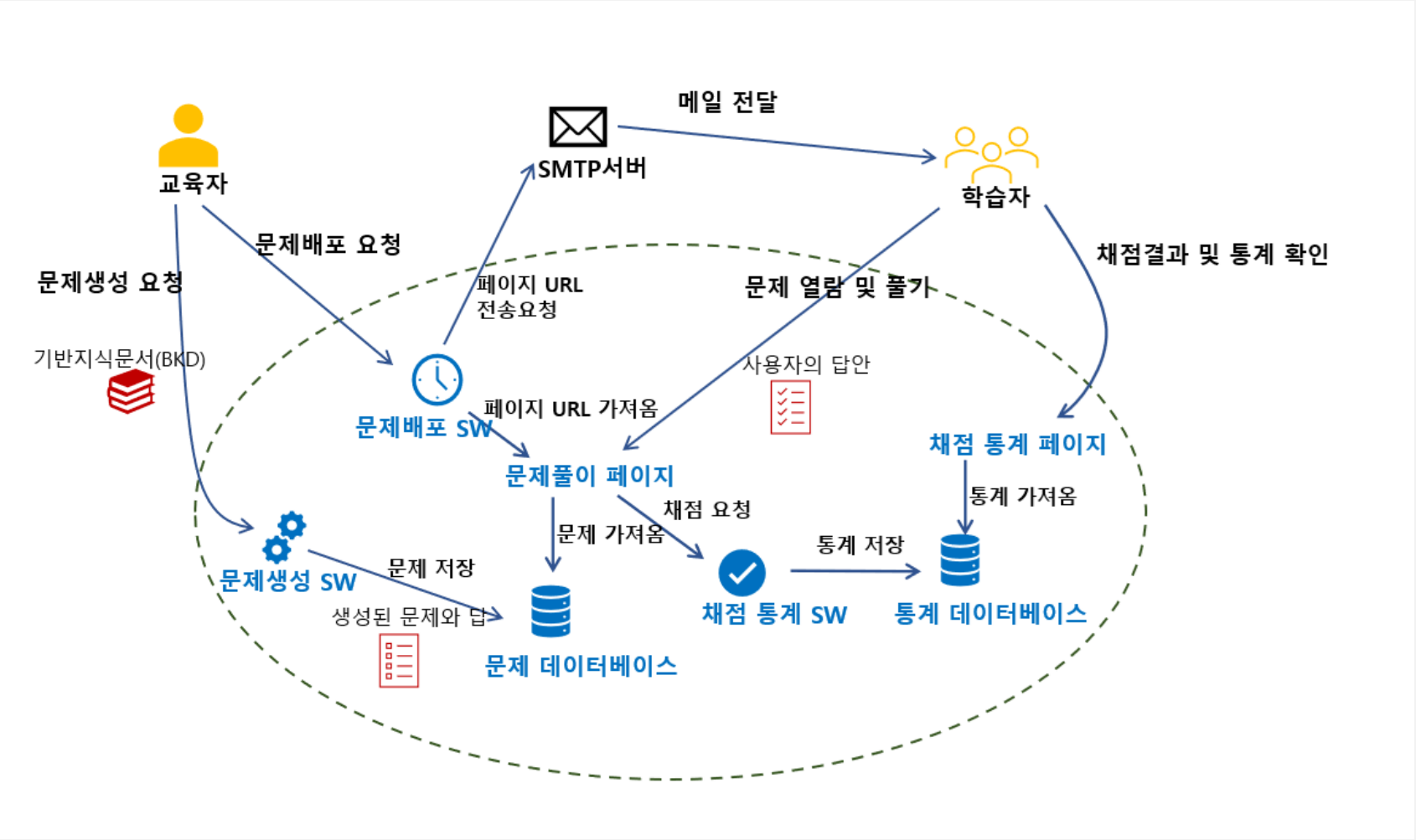
Automatic Problem Generation System

2020.07.09

곽신우 (201411260)
문성찬 (201511260)
배율희 (201511266)
주재빈 (201511298)

Contents

Business Model



Non-Functional Requirements & Glossary

Non-Functional Requirements

- 여러 사용자가 동시에 서비스를 받을 수 있다.
- 생성된 문제의 90% 이상을 사용할 수 있다.

Glossary

Name	Description
BKD (Base Knowledge Documents)	문제 생성의 기반이 되는 정보가 작성된 문서
채널	BKD를 통해 생성된 문제를 배포받을 수 있는 개인 및 집단
문제 생성자	BKD를 사용하여 문제를 생성 및 배포하는 주체자
문제 소비자	생성된 문제를 배포받아 풀이하는 사람

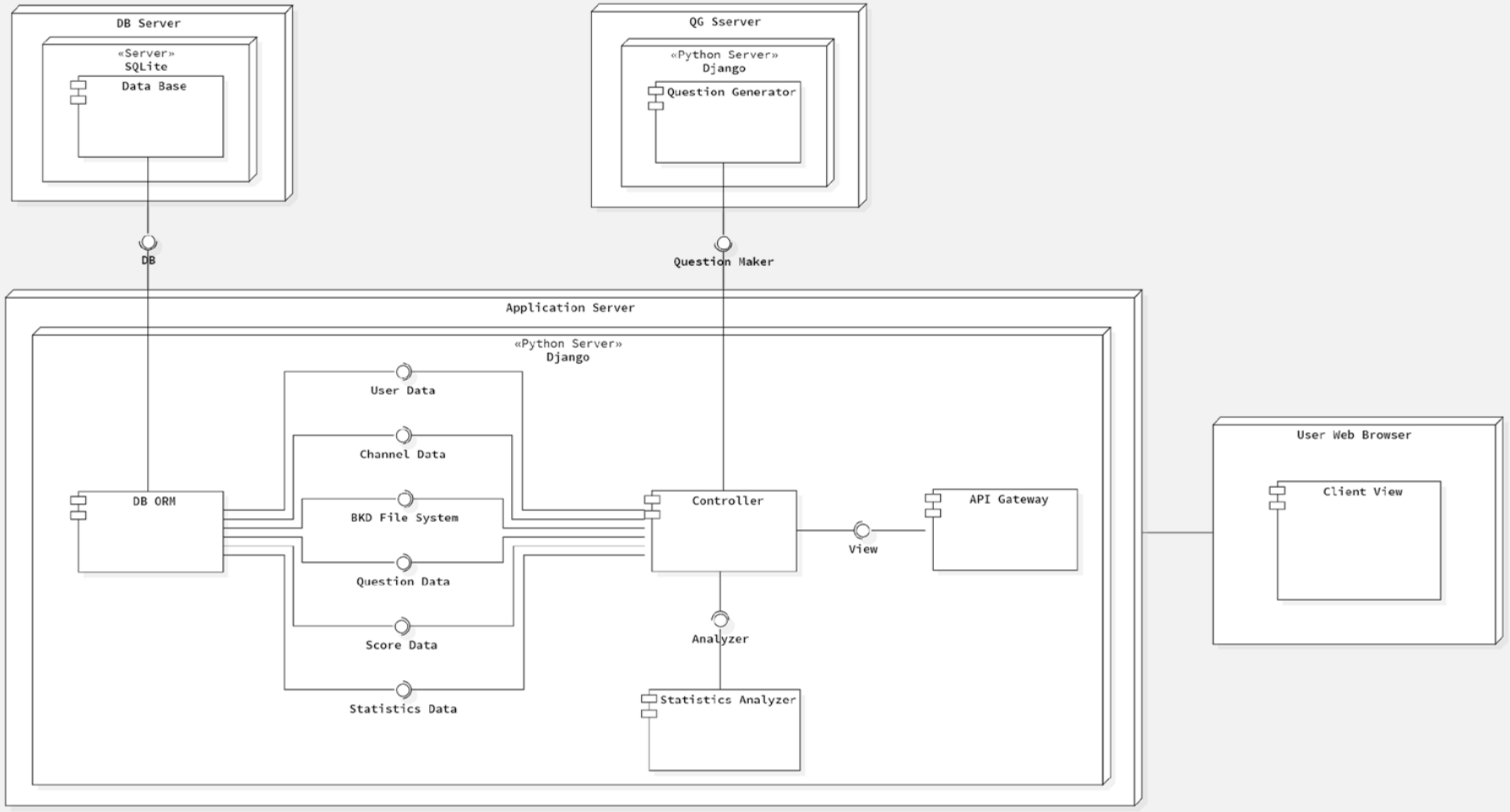
Functional Requirements

Usecase ID	Use Case	Actor	Description
1.1	채널 생성	User	- 채널 항목을 생성하여 채널 목록에 추가 - 생성 시 채널 이름을 입력
1.4	채널 입장 경로 생성	User	- 채널 목록에서 채널을 선택 - 선택한 채널의 입장 경로(URL 등)을 생성
2.1	BKD 디렉토리 생성	User	- BKD 목록에서 하나를 선택하여 BKD 디렉토리를 생성 - 생성된 BKD 디렉토리를 BKD 디렉토리 목록에 추가 - 생성 시 BKD 디렉토리의 이름 등의 정보를 입력
3.1	BKD 생성	User	- 마크다운 파일을 시스템에 업로드, 혹은 직접 화면에서 파일 작성 - BKD의 이름을 부여해야 함
4.1	BKD 선택	User	- 저장된 BKD를 선택할 수 있게 표시된 상태이다 - 문제를 생성하고자 하는 BKD를 선택
4.2	문제 유형 선택	User	- 만들고자 하는 문제의 유형을 선택 - 선택지의 수, 답의 수 지정
4.3	문제 수 선택	User	- 생성할 문제의 수를 선택 - 선택한 문제의 수에 따라 적정한 수인지를 표시
4.4	문제 생성	User	- 선택된 옵션에 의해 문제와 답안을 생성 - 문제 생성 방식에 따라서 서로 다른 알고리즘 사용
5.2	문제 배포	System	- 문제를 사용자에게 실제로 전달 - 전달 시간과 전달 대상은 사전에 정의된 정보를 따름
6.1	통계 계산	System	- 채널을 이루는 개인의 결과를 통합하여 통계를 산출한다
7.1	채널 참여	User	- 채널 입장 경로(URL 등)를 통해 특정 채널에 참여
9.1	문제 수신	User	- 배포된 문제를 수신한다. - 수신한 문제를 확인한다.
11.1	답안 제출	User	- 수신한 문제를 풀고 난 후의 답안을 제출한다
11.2	채점	System	- 제출한 답안을 채점한다 - 개인마다 문제를 푼 결과를 저장한다

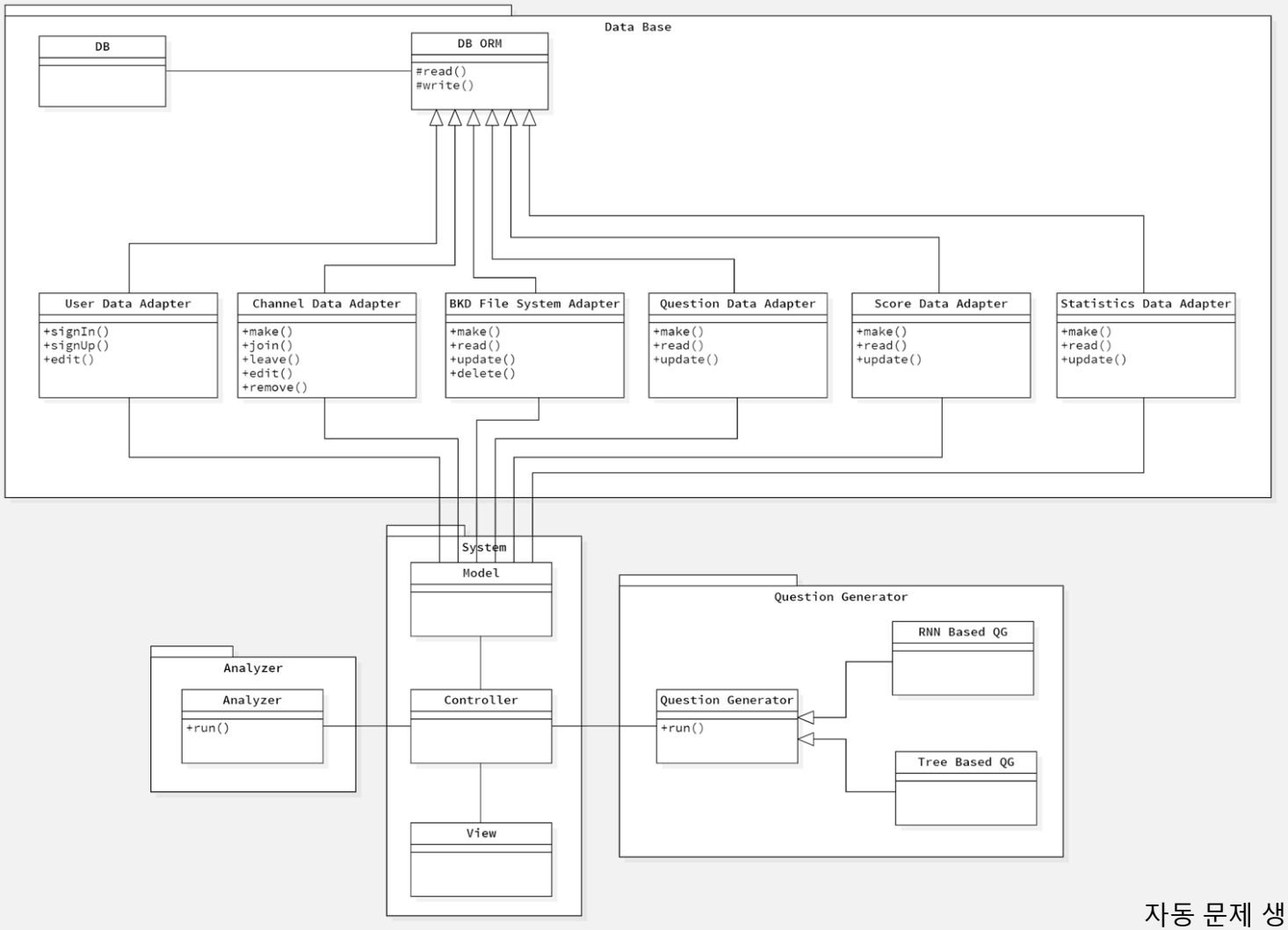
주요 기능만 명시하였습니다.

모든 기능 : https://drive.google.com/file/d/10T0ix-qzh3I-7dHIzlS2DElQQpgxFWtQ/view?usp=sharing

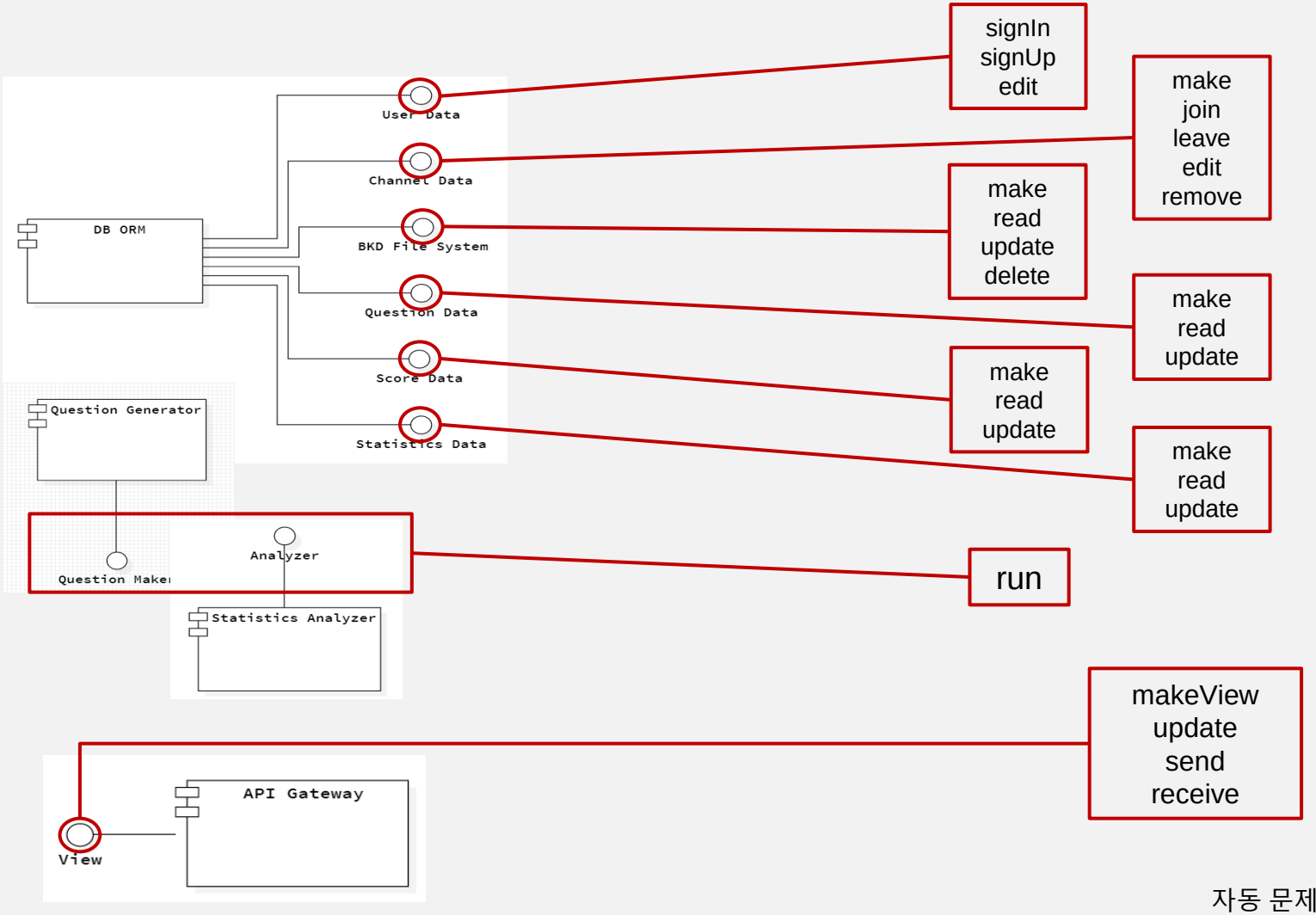
Deployment Diagram



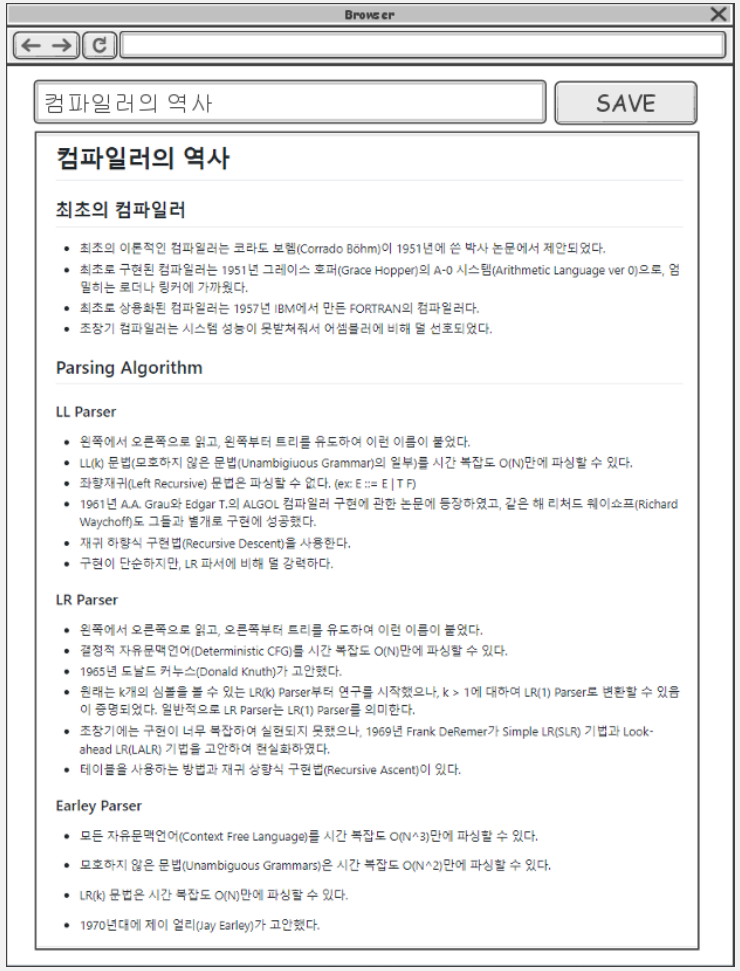
Component Diagram

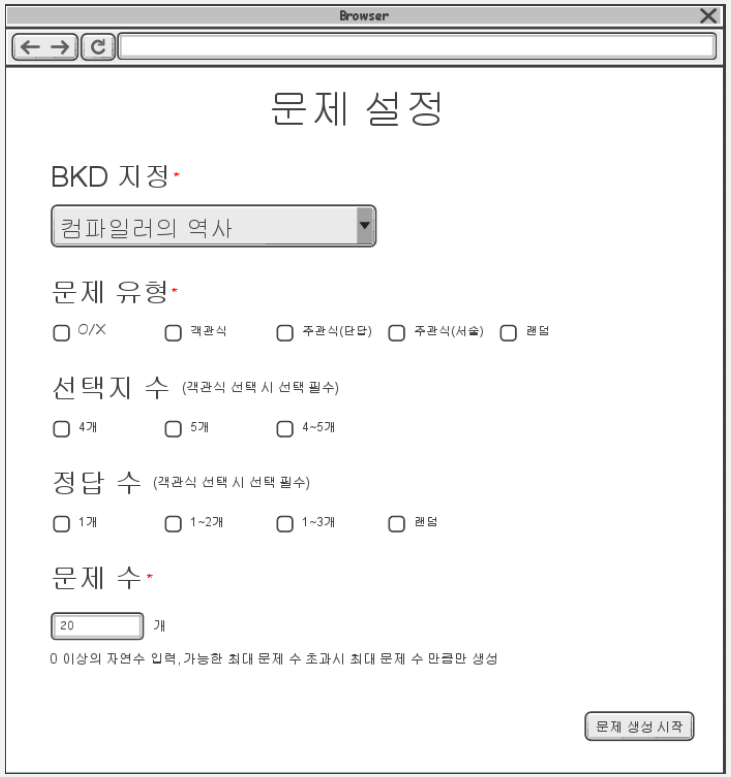


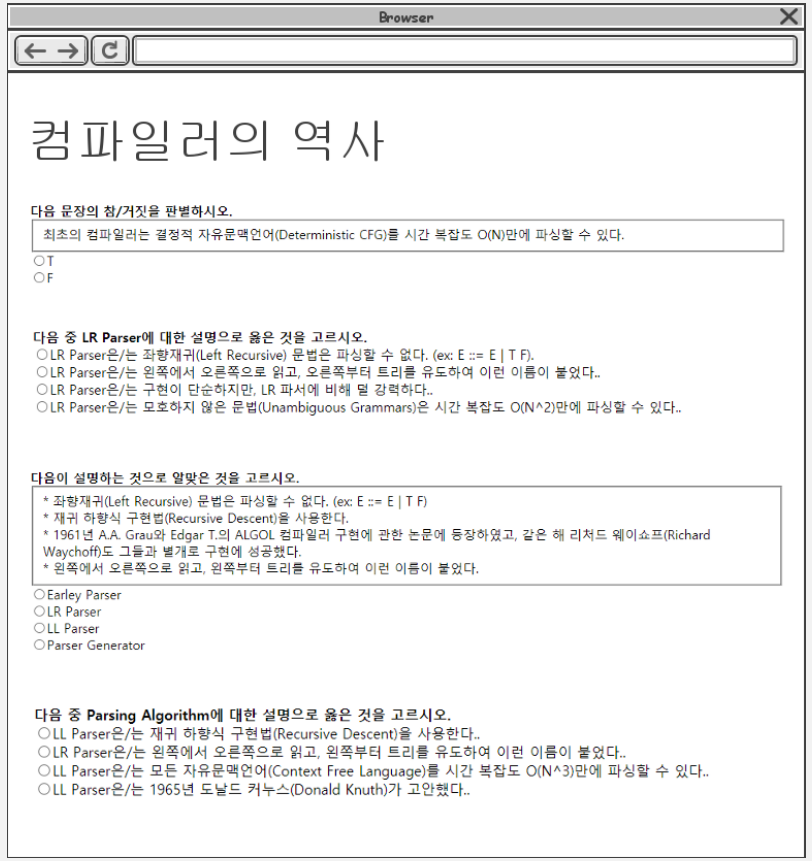
Component Interface



Wireframe


Browser
컴파일러의 역사
SAVE
컴파일러의 역사
최초의 컴파일러
최초의 이론적인 컴파일러는 코라도 보헴(Corrado Böhm)이 1951년에 쓴 박사 논문에서 제안되었다.
최초로 구현된 컴파일러는 1951년 그레이스 호퍼(Grace Hopper)의 A-0 시스템(Arithmetic Language ver 0)으로, 엄밀히는 로더나 링커에 가까웠다.
최초로 상용화된 컴파일러는 1957년 IBM에서 만든 FORTRAN의 컴파일러다.
초창기 컴파일러는 시스템 성능이 뭇받쳐줘서 어셈블러에 비해 덜 선호되었다.
Parsing Algorithm
LL Parser
왼쪽에서 오른쪽으로 읽고, 왼쪽부터 트리를 유도하여 이런 이름이 붙었다.
LL(k) 문법(모호하지 않은 문법(Unambigiuous Grammar)의 일부)를 시간 복잡도 O(N)만에 파싱할 수 있다.
좌향재귀(Left Recursive) 문법은 파싱할 수 없다. (ex: E ::= E | T F)
1961년 A.A. Grau와 Edgar T.의 ALGOL 컴파일러 구현에 관한 논문에 등장하였고, 같은 해 리처드 웨이쇼프(Richard Waychoff)도 그들과 별개로 구현에 성공했다.
재귀 하향식 구현법(Recursive Descent)을 사용한다.
구현이 단순하지만, LR 파서에 비해 덜 강력하다.
LR Parser
왼쪽에서 오른쪽으로 읽고, 오른쪽부터 트리를 유도하여 이런 이름이 붙었다.
결정적 자유문맥언어(Deterministic CFG)를 시간 복잡도 O(N)만에 파싱할 수 있다.
1965년 도날드 커누스(Donald Knuth)가 고안했다.
원래는 k개의 심볼을 볼 수 있는 LR(k) Parser부터 연구를 시작했으나, k > 1에 대하여 LR(1) Parser로 변환할 수 있음이 증명되었다. 일반적으로 LR Parser는 LR(1) Parser를 의미한다.
초창기에는 구현이 너무 복잡하여 실현되지 못했으나, 1969년 Frank DeRemer가 Simple LR(SLR) 기법과 Look-ahead LR(LALR) 기법을 고안하여 현실화하였다.
테이블을 사용하는 방법과 재귀 상향식 구현법(Recursive Ascent)이 있다.
Earley Parser
모든 자유문맥언어(Context Free Language)를 시간 복잡도 O(N^3)만에 파싱할 수 있다.
모호하지 않은 문법(Unambiguous Grammars)은 시간 복잡도 O(N^2)만에 파싱할 수 있다.
LR(k) 문법은 시간 복잡도 O(N)만에 파싱할 수 있다.
1970년대에 제이 얼리(Jay Earley)가 고안했다.


Browser
문제 설정
BKD 지정*
컴파일러의 역사
문제 유형*
O/X
객관식
주관식(단답)
주관식(서술)
랜덤
선택지 수 (객관식 선택 시 선택 필수)
4개
5개
4~5개
정답 수 (객관식 선택 시 선택 필수)
1개
1~2개
1~3개
랜덤
문제 수*
20
개
0 이상의 자연수 입력, 가능한 최대 문제 수 초과시 최대 문제 수 만큼만 생성
문제 생성 시작


Browser
컴파일러의 역사
다음 문장의 참/거짓을 판별하시오.
최초의 컴파일러는 결정적 자유문맥언어(Deterministic CFG)를 시간 복잡도 O(N)만에 파싱할 수 있다.
T
F
다음 중 LR Parser에 대한 설명으로 옳은 것을 고르시오.
LR Parser은/는 좌향재귀(Left Recursive) 문법은 파싱할 수 없다. (ex: E ::= E | T F).
LR Parser은/는 왼쪽에서 오른쪽으로 읽고, 오른쪽부터 트리를 유도하여 이런 이름이 붙었다..
LR Parser은/는 구현이 단순하지만, LR 파서에 비해 덜 강력하다..
LR Parser은/는 모호하지 않은 문법(Unambiguous Grammars)은 시간 복잡도 O(N^2)만에 파싱할 수 있다..
다음이 설명하는 것으로 알맞은 것을 고르시오.
* 좌향재귀(Left Recursive) 문법은 파싱할 수 없다. (ex: E ::= E | T F)
* 재귀 하향식 구현법(Recursive Descent)을 사용한다.
* 1961년 A.A. Grau와 Edgar T.의 ALGOL 컴파일러 구현에 관한 논문에 등장하였고, 같은 해 리처드 웨이쇼프(Richard Waychoff)도 그들과 별개로 구현에 성공했다.
* 왼쪽에서 오른쪽으로 읽고, 왼쪽부터 트리를 유도하여 이런 이름이 붙었다.
Earley Parser
LR Parser
LL Parser
Parser Generator
다음 중 Parsing Algorithm에 대한 설명으로 옳은 것을 고르시오.
LL Parser은/는 재귀 하향식 구현법(Recursive Descent)을 사용한다..
LR Parser은/는 왼쪽에서 오른쪽으로 읽고, 왼쪽부터 트리를 유도하여 이런 이름이 붙었다..
LL Parser은/는 모든 자유문맥언어(Context Free Language)를 시간 복잡도 O(N^3)만에 파싱할 수 있다..
LL Parser은/는 1965년 도날드 커누스(Donald Knuth)가 고안했다..

System Test Case

Usecase ID	Use Case	System Test ID	Description
1.1	채널 생성	ST.1.1.1	- 기존 채널 항목과 중복되지 않는 이름으로 채널 생성
1.4	채널 입장 경로 생성	ST.1.4.1	- 채널 입장 경로의 생성
2.1	BKD 디렉토리 생성	ST.2.1.1	- BKD를 하나 선택한 상태로 BKD 디렉토리 생성. 해당 BKD를 포함하는 BKD 디렉토리가 생성된다
3.1	BKD 생성	ST.3.1.1	- *.md 파일을 업로드하여 BKD를 생성한다. 파일의 이름대로 BKD가 생성되는 지 확인.
3.1	BKD 생성	ST.3.1.3	- BKD를 직접 작성하여 생성한다. 작성한 BKD가 목록에 추가되는지 확인
4.1	BKD 선택	ST.4.1.3	- 호환이 불가능한 *.md파일을 업로드하여 BKD 선택
4.2	문제 유형 선택	ST.4.2.1	- 생성할 문제의 수보다 많거나 같은 수의 문제 유형을 선택
4.2	문제 유형 선택	ST.4.2.2	- 생성할 문제의 수보다 적은 수의 문제 유형을 선택
4.3	문제 수 선택	ST.4.3.1	- 선택된 BKD에서 생성할 수 있는 문제의 적정 수보다 많은 수를 선택
4.3	문제 수 선택	ST.4.3.2	- 선택된 BKD에서 생성할 수 있는 문제의 적정 수보다 적은 수를 선택
4.3	문제 수 선택	ST.4.3.3	- 선택된 BKD에서 생성할 수 있는 문제의 적정 수와 같은 수를 선택
4.4	문제 생성	ST.4.4.1	- BKD를 지정하지 않고 문제 생성 설정
4.4	문제 생성	ST.4.4.6	- 모든 옵션을 지정하고 방법 1로 문제 생성 설정
4.4	문제 생성	ST.4.4.7	- 모든 옵션을 지정하고 방법 2로 문제 생성 설정
7.1	채널 참여	ST.7.1.1	- URL을 통해 특정 채널에 참여한다
7.1	채널 참여	ST.7.1.2	- 해당 채널의 참여코드를 올바르게 입력
9.1	문제 수신	ST.9.1.1	- 배포된 문제를 수신한다.
9.1	문제 수신	ST.9.1.2	- 수신한 문제 목록에서 특정 문제를 확인한다.
11.1	답안 제출	ST.11.1.1	- 풀이된 답안을 제출

주요 기능에 대한 테스트만 명시하였습니다.

모든 테스트 : https://drive.google.com/file/d/10T0ix-qzh3I-7dHIzlS2DElQQpgxFWtQ/view?usp=sharing