SW상세설계서

아마존 익스프레스 상세 설계서

제출일자 : 2024.10.17(목)

팀 : 6팀 200 OK

1. 프로젝트 개요

1.1 목적 :

이 설계서의 목적은 '아마존 익스프레스'의 핵심 구성 요소를 구체적으로 설계하여 향후 개발을 효율적으로 진행하기 위함입니다.

1.2 스토리:

우주여행이 보편화되고 행성 간 이동이 활발해진 미래, 우주 배달 서비스가 성행합니다. 주인공은 이러한 세계에서 우주를 오가는 배달 기사입니다.

어느 날 평소처럼 배달 중이던 주인공은 우주 쓰레기와 충돌하여 지구의 아마존으로 추락합니다. 이제 주인공은 아마존에서 생존해야 하는 상황에 처합니다.

1.3 참조 문서 :

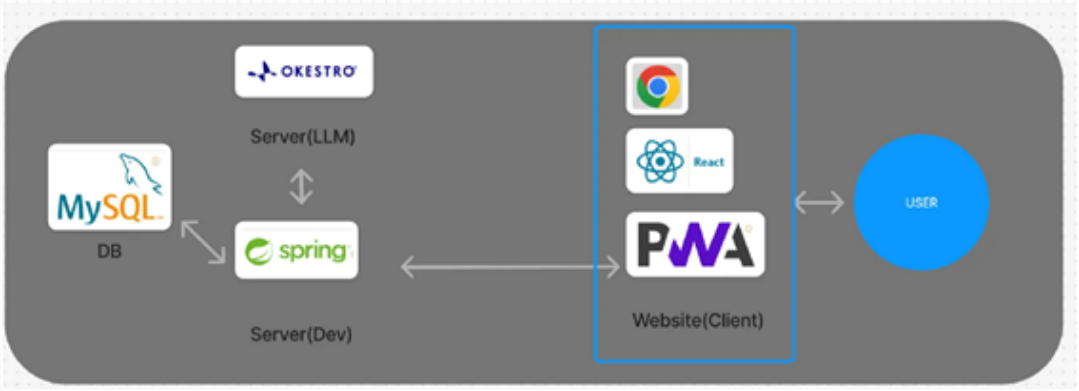
프로젝트 최종 계획서, 소프트웨어 상세 설계서 예시

1.4 용어 정의 :

아마존 익스프레스: '익스프레스'(배달 기사를 의미)와 '아마존'(지역)의 합성어

2. 시스템 구조

2.1 전체 시스템 구조도



2.2 주요 컴포넌트 설명

3. 모듈 상세 설계

3.1 스토리 및 홈화면

- Home()
- 메인 스토리 출력
- 환경 설정 버튼
- 게임 시작 버튼: route to Contract();

3.2 유저 정보 입력, 계약서 작성: 계약서 형식으로 입력하여 게임 몰입감 상승

- Contract()
- 유저 정보 입력 받음 - 게임에 적용
- 계약하기 버튼: route to MiniGame();

3.3 미니게임 화면 (자원 모으기)

- MiniGame()
- Dodge 게임 형식의 미니게임
- 미니게임 진행 동안 게임에 필요한 최적화 작업 수행
- 미니게임 종료: route to Game();

3.4 게임 메인화면

- Game()
- 메인 게임 화면
- 이미지 생성 및 출력
- 게임 진행 상황 텍스트 생성 및 출력
- 진행 상황에 따른 3개의 선택지 제공
- 자원 보유 현황 표시
- 게임 중 LLM의 데이터 처리 시 로딩 화면 표시, LLM API를 사용하여 선택지 생성 및 다음 화면 데이터 준비

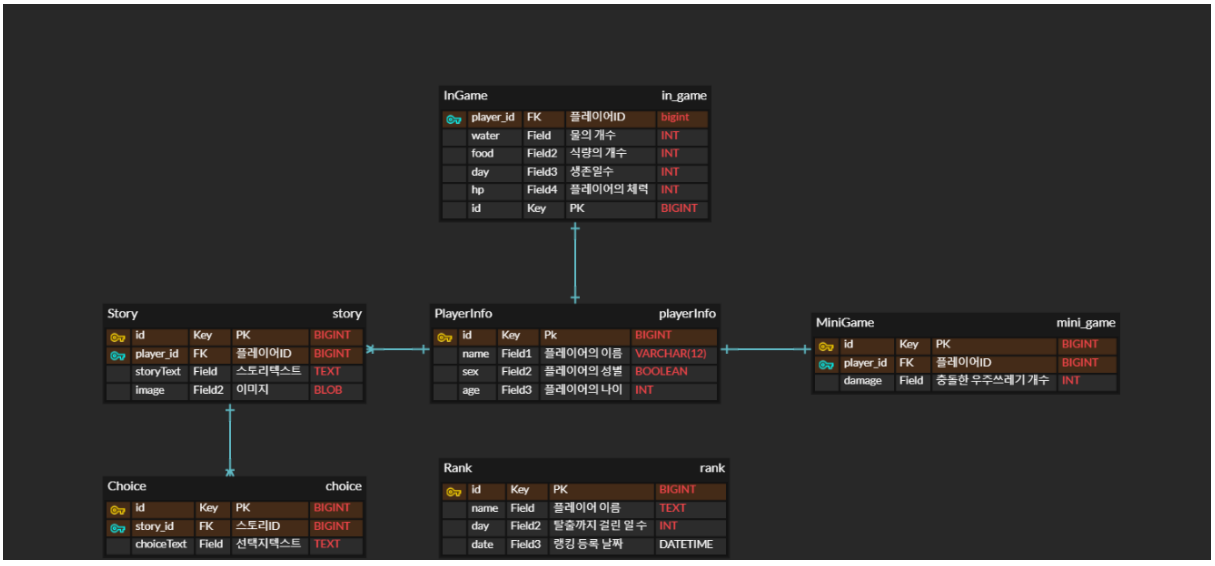
3.5 결과화면

- Ending()
- 게임 결과에 따른 이미지 출력
- 시작화면으로 가기 버튼 route to Home();
- 랭킹화면 띄우기(모달 효과) route to Ranking();, 여백 누르면 Ending() 보여주기

---

4. 데이터베이스 설계

4.1 ER 다이어그램



4.2 테이블 정의

1. `PlayerInfo` 테이블

```
CREATE TABLE PlayerInfo (
  id BIGINT PRIMARY KEY AUTO_INCREMENT,   -- 고유 플레이어 ID
  name VARCHAR(12),                       -- 플레이어 이름
  sex BOOLEAN,                            -- 성별 (TRUE = 남성, FALS
  age INT                                 -- 플레이어 나이
);
```

2. `InGame` 테이블

```
CREATE TABLE InGame (
  id BIGINT PRIMARY KEY AUTO_INCREMENT,   -- 고유 인게임 ID
  player_id BIGINT,                       -- 외래 키, 플레이어 ID
  water INT,                              -- 수집한 물 자원
  food INT,                               -- 수집한 음식 자원
  day INT,                                -- 생존 일수
  hp INT,                                 -- 체력
  FOREIGN KEY (player_id) REFERENCES PlayerInfo(id)
);
```

3. `MiniGame` 테이블

```
CREATE TABLE MiniGame (
  id BIGINT PRIMARY KEY AUTO_INCREMENT,   -- 고유 미니게임 ID
  player_id BIGINT,                       -- 외래 키, 플레이어 ID
  damage INT,                             -- 충돌한 우주쓰레기 개수
  FOREIGN KEY (player_id) REFERENCES PlayerInfo(id)
);
```

4. `Story` 테이블

```
CREATE TABLE Story (
  id BIGINT PRIMARY KEY AUTO_INCREMENT,   -- 고유 스토리 ID
  player_id BIGINT,                       -- 외래 키, 플레이어 ID
  storyText TEXT,                         -- 스토리 텍스트
```

```
  image BLOB,                                -- 스토리 이미지
  FOREIGN KEY (player_id) REFERENCES PlayerInfo(id)
);
```

5. `Choice` **테이블**

```
CREATE TABLE Choice (
  id BIGINT PRIMARY KEY AUTO_INCREMENT,   -- 고유 선택지 ID
  story_id BIGINT,                        -- 외래 키, 스토리 ID
  choiceText TEXT,                        -- 선택지 텍스트
  FOREIGN KEY (story_id) REFERENCES Story(id)
);
```

6. `Rank` **테이블**

```
CREATE TABLE Rank (
  id BIGINT PRIMARY KEY AUTO_INCREMENT,   -- 고유 랭킹 ID
  name TEXT,                              -- 플레이어 이름
  day INT,                                -- 플레이어 생존 일수
  date DATETIME                           -- 랭킹 등록 날짜
);
```

4.3 테이블 간 관계

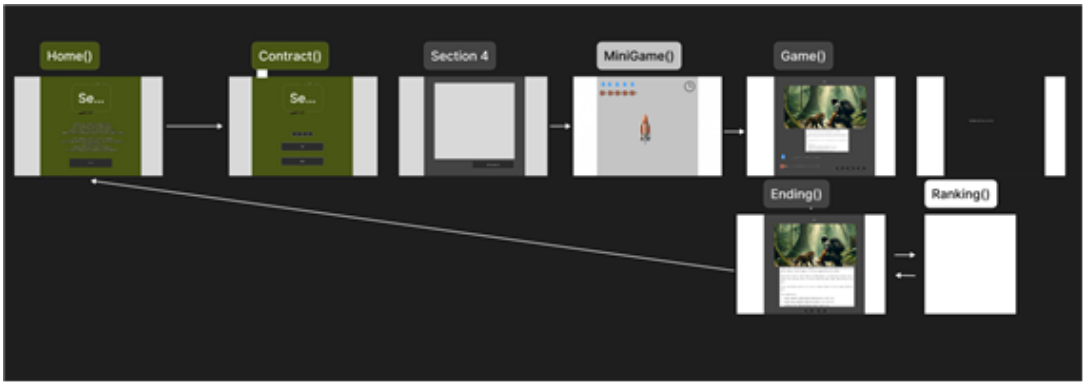
- `PlayerInfo (1)` : `InGame (1)`
  - 한 플레이어는 한 게임 진행 기록을 가집니다.
- `PlayerInfo (1)` : `MiniGame (1)`
  - 한 플레이어는 한 미니게임에 참여할 수 있습니다.
- `PlayerInfo (1)` : `Story (N)`
  - 한 플레이어는 여러 스토리를 진행할 수 있습니다.
- `Story (1)` : `Choice (N)`
  - 한 스토리는 여러 선택지를 가집니다.

- `PlayerInfo`와 `Rank`는 **직접 연결되어 있지 않지만**, 플레이어의 이름과 생존 일수 정보가 기록됩니다.

---

5. 사용자 인터페이스 설계

5.1 화면 구조도



5.2 사용자 시나리오

1. 웹 페이지에 접속을 하면, 사용자는 간략한 스토리 설명이 담긴 시작버튼을 마주한다.
2. 시작 버튼을 통해 넘어온 화면에서는 사용자의 플레이어명, 성별, 나이 등의 정보를 입력하고, 계약서에 동의 버튼을 통해 게임이 시작된다. 이 때 DB에 PlayerInfo 튜플 하나를 생성한다.
3. 잠시 후, 닷지 게임을 레퍼런스로 하는 자원지키기 미니 게임에 진입하게 된다. JavaScript로 구현되며, 사용자는 웹에서는 클릭 앤 드래그, 모바일에서는 터치 앤 드래그 방식으로 우주선으로 소행성을 회피하며, 제한시간을 버티게 된다. 제한시간 동안에 피격 수에 비례하여 초기 식량 식수를 삭감. 실패시 자원이 최소치인 채로 게임을 시작하게 된다. 이 미니게임의 결과를 바탕으로 서버에서 게임에 필요한 자원들을 생성하고 초기화한다. LLM 측에 이 식량 식수 정보가 주어지고 기초 스토리가 생성된다.
4. 잠시 로딩을 거친 뒤, 사용자는 LLM에 의해 생성된 시나리오가 서버를 통해 클라이언트에게 전달되면, 플레이어는 게임 화면을 마주하게 된다. 사용자는 생성된 이미지와 함께 조난당한 스토리를 읽은 뒤, LLM에 의해 생성되는 세 가지 선택지 중 하나를 선택하게 된다. 유저가 선택한 선택지는 다시 LLM으로 전달된다.

   4-1. 선택지가 확정됐을 때, 다음 게임 상황 생성을 위한 로딩 시간에, 플레이어는 검은색 화면과 흰 바탕의 텍스트, (아기자기한 그림)을 마주하게 된다. 간밤에 있었던 일(비가 왔다거나, 동물의 울음소리, 특수 식량이벤트 따위)이 미니 이벤트로 출력되게 되며, 로딩이 완료되면, 자동으로 익일로 넘어간다.

5. 식량과 식수는 선택지 바탕으로 LLM이 변동시키고, 선택지를 바탕으로 ai가 출력해주는 이미지와 텍스트 스토리를 마주하게 되며, 엔딩에 이르기까지 사용자는 4번 시나리오를 반복적으로 진행하게 된다.
6. 엔딩에 이르게 되면, 사용자의 플레이 기록 정리와 함께, 엔딩 시나리오와 사진을 제시한다. 여기서 메인화면으로 돌아갈지, 랭킹에 저장할지를 선택할 수 있으며, 자신의 플레이어명 혹은 이니셜 등으로 자신의 생존 day 기록을 남길 수 있으며, 상위(게임이 빨리 끝날수록 높은 순위) 10개까지 랭킹 DB에 등록된다. 등록이 완료된 뒤에 홈으로 돌아갈 수 있다.

---

6. API 설계

6-1. 플레이어 정보 입력받는 API

```
Endpoint: /api/playerInfo
HTTP 메서드: POST
Request Body:
{
  "playerName": "string",
  "sex": "boolean",
  "age": "int"
}
```

6-2. 게임의 스토리를 오케스트로 AI로부터 받는 API

```
Endpoint: /api/gameStory
HTTP 메서드: GET
Response:
{
  "story": "string"
}
```

6-3. 오케스트로 AI로부터 스토리를 텍스트로 묘사한 정보를 받는 API

이 텍스트는 DALL·E AI로 보내서 이미지를 생성할 때 사용될예정임

```
Endpoint: /api/storyDescription
HTTP 메서드: GET
```

```
Response:
{
  "description": "string"
}
```

6-4. 스토리에서 3개의 선택지를 받는 API

```
Endpoint: /api/storyChoices
HTTP 메서드: GET
Response:
{
  "choices": [
    "choice1",
    "choice2",
    "choice3"
  ]
}
```

6-5. 스토리의 텍스트를 DALL·E에 보내서 이미지를 생성하는 API

```
Endpoint: /api/dalle/storyImage
HTTP 메서드: POST를 생성하는 API
Request Body:
{
  "description": "string"
}
Response:
{
  "imageUrl": "string"
}
```

6-6. 플레이어가 선택한 선택지를 오케스트로 AI로 보내는 API

```
Endpoint: /api/sendChoice
HTTP 메서드: POST
Request Body:
```

```
{
  "playerId": "string",
  "choice": "string"
}
```

6-7. 로딩창에 필요한 독백 대사를 오케스트로 AI로부터 받는 API

```
Endpoint: /api/loadingDialogue
HTTP 메서드: GET
Response:
{
  "dialogue": "string"
}
```

6-8. 미니게임 결과로 나온 자원 정보를 오케스트로 AI로 보내는 API

```
Endpoint: /api/sendResources
HTTP 메서드: POST
Request Body:
{
  "playerId": "string",
  "resources": {
    "water": "int",
    "food": "int",
  }
}
```

6-9. 최종 점수를 스프링 서버로 보내는 API

```
Endpoint: /api/sendFinalScore
HTTP 메서드: POST
Request Body:
{
  "playerId": "string",
  "finalScore": "int"
}
```

6-10. 최종 점수를 스프링 서버로부터 받는 API

```
Endpoint: /api/getFinalScore
HTTP 메서드: GET
Response:
{
  "playerId": "string",
  "finalScore": "int"
}
```

---

7. 성능 및 확장성 고려사항

1. 코드 최적화
   적합한 알고리즘 사용 및 코드 최적화로 가독성 증가 및 처리 시간 단축
2. LLM 정보 생성 시간 최적화
   플레이어 선택에 따른 다음 이미지 및 텍스트 미리 생성 , 플레이어 체감 시간 축소
3. 데이터 통합화
   플레이어 정보에 대한 부가 내용들을 SessionId 한가지에 묶어서 취급하여 개발 속도 향상 및 손실율 최소화

---

8. 테스트 계획

8.1 단위 테스트 계획

- 시작 부터 엔딩까지 각 서비스 분야에서 오류 발생건 없는지 확인

8.2 통합 테스트 계획

- 통합 테스트: 벡엔드 , 프론트엔드에서 정보 교환 과정에서 오류 발생 검증

8.3 성능 테스트

- 동시 사용자 사용 오류 발생 검증 및 LLM 스토리 및 이미지 생성 속도 검증

8.4 AI 정확도 테스트

- 다양한 입력에 대한 AI 모델의 응답 정확도 테스트