소프트웨어 개발의 원리 및 실제 - 실습 과제 4

Apr. 14

온라인 오목 게임

1) 소개

숙제 3.B 에서 구현한 오목 게임이 온라인 대전이 가능하도록 확장해 본다.

1. **백엔드 서버를 구현한다. 백엔드 서버는 오목 게임과 관련된 정보를 저장한다. 오목 게임에는 여러 명의 유저가 있을 수 있으며, 또 여러 개의 방이 있을 수 있다. 한 방에 두 명이 들어갈 경우 게임이 시작된다. 방에 입장한 유저의 정보 및 게임의 진행 상황은 백엔드에 기록된다.**

2. **프론트엔드에 로그인 기능을 추가한다. 회원가입 기능은 없고, 테스트에 쓰일 유저는 이미 만들어져 있다고 가정한다. 로그인 할 때 유저 정보와 함께 어떤 방에 접속할 것인지를 기입한다. 각 방에는 번호가 매겨져 있고, 접속할 때는 해당 방 번호를 기입한다. 유저와 게임을 진행할 방은 이미 (빈 방의 상태로) 만들어져 있다고 가정한다. 2명이 채워지는 순간, 게임을 시작한다. 이미 2명이 채워진 방에 접속하려고 한 경우 프론트엔드는 아무 행동을 하지 않는다.**

3. **두 사람이 웹 브라우저에서 게임을 진행한다. 방에 먼저 로그인한 사람이 선공이다(O로 시작한다). 바둑판의 크기는 이전 과제와 동일하게 19 x 19이다. 한 사람이 돌을 놓으면, 다른 사람이 보는 웹 브라우저에도 1초 이내에 해당 내역이 업데이트 되어야 한다. 중간에 누가 웹 브라우저를 닫는 일은 없다고 가정한다.**

4. **게임이 끝나면 승리/패배 정보를 화면에 표현한다. 게임이 끝난 다음에 유저는 웹 브라우저를 종료하는 것 이외에 아무것도 할 수 없다. 방은 게임이 끝나더라도 사라지거나 초기화 되지 않는다. (유저 정보 또한 포함)**

2) 구현 상세 정보

(1) 백엔드 서버

백엔드 서버는 Django REST 로 구현한다. 구현해야 할 URL 은 다음과 같다.

URL 명	설명
/rooms/	- GET : 현재 방 목록을 반환 (퍼미션 조건 없음 (authorize 되지 않은 사람도 요청 가능)) - POST : 새로운 방을 추가 ("omok_admin"이라는 유저만 가능) - PUT, DELETE : 구현하지 않음

/rooms/ID	- GET : 방번호(id)가 ID 인 방의 정보를 반환 (퍼미션 조건 없음) - POST, PUT, DELETE : 구현하지 않음
/rooms/ID/players	- GET : 방번호가 ID 인 방에 들어간 사람을 리스트 형태로 반환 (퍼미션 조건 없음) * 한 방에는 최대 2 명의 대전자가 있으므로, GET 이 리턴하는 리스트의 길이는 최대 2 이다. - POST : 대전자 목록에 유저를 추가. (현재 유저를 리스트에 등록) * 만약 방에 이미 2 명의 대전자가 존재할 경우 아무것도 하지 않고 반환 - PUT, DELETE : 구현하지 않음
/rooms/ID/history	- GET : 방번호가 ID 인 방에서 놓여진 수들을 리스트 형태로 반환 (퍼미션 조건 없음) * [(i1, j1, userid1), (i2, j2, userid2), (i3, j3, userid1), (i4, j4, userid2), …] 꼴을 띈다 - POST : 새로운 수를 등록함. (바둑돌 위치는 값으로 받음, 유저는 현재 유저를 등록) * POST 할 수 있는 유저는 해당 방에 들어온 사람들 중 한명이어야 함 * 가장 마지막에 POST 한 유저가 또 다시 POST 를 할 수 없음 - PUT, DELETE : 구현하지 않음

(2) 프론트엔드 서버

프론트엔드는 React + Redux 로 구현한다. 백엔드와 프론트엔드는 Redux Saga 를 이용해서 연결한다.

프론트엔드에서 로그인시, 유저 이름의 입력칸은 "username_field"라는 id 를 가져야 하고, 패스워드 입력칸은 "password_field"라는 id 를 가져야 한다. (username 이 중복되는 유저는 존재하지 않는다고 가정한다) 패스워드 입력칸은 패스워드를 보여주어도 괜찮다.(즉, 패스워드가 ***와 같이 표현될 필요는 없음) 로그인시에는 username_field 으로부터 username 을, password_field 으로부터 password 를 받는다. 방 번호 입력칸은 "room_field"라는 id 를 가져야 한다. 상대 유저의 username 을 나타내는 칸은 "enemy_field"라는 id 를 가져야 한다. 로그인하는 버튼은 "connect"란 id 를 가져야 한다. 로그인에 실패를 하거나 존재하지 않는 방번호를 입력했을 경우 connect 버튼을 눌러도 아무 일이 일어나지 않는다. 성공적인 로그인 이후로 "connect" 버튼이 다시 클릭되는 일은 없다고 가정한다.

바둑판은 19 x 19 개의 칸으로 이루어져 있으며, 각 칸은 이전 숙제와 동일하게 "y_x" 꼴의 id 를 가진다. 화면의 가장 아래에는 현재 상태를 나타내는 구역이 있으며 id는 이전 숙제와 동일하게 "status_label"이다. status_label 은 로그인 이전에는 아무 것도 표현하고 있지 않으며, 로그인 이후에 상대방을 기다리고 있는 중에는 "WAITING"을 보여준다. 접속한 상대와 대전을 시작한 때부터는 이전 숙제와 동일한 방식으로 "Next O / Next X / O win / X win" 중 하나를 보여준다.

다음 장에 나오는 그림들은 각각 로그인 전, 로그인 후, 대전 중에 화면에 나타나야 하는 요소들을 그린 것이다. 테두리가 점선인 칸은 사용자로부터 입력을 받는 구역이고, 테두리가 실선인 칸은 텍스트를 나타내기만 하는 구역이다.

A. Login 전 화면

Login : **PW:** **Room Number:**

Enemy : **Connect**

- 초기에는 빈 화면
- 로그인 후 이 화면에 보드가 나타나고 게임을 진행할 수 있도록 함

(status_label)

B. Login 후 화면 (대전할 상대를 기다릴 때)

Login : **PW:** **Room Number:**

Enemy : **Connect**

-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-

WAITING

C. 대전 중 화면

Login : **PW:** **Room Number:**

Enemy :

CONNECT

-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	**O**	-	-	-	-
-	-	-	-	**X**	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-

Next O

4) 제출 및 채점

채점은 수업 및 실습 시간에 공지된 바와 같이, 학생들이 제출한 bitbucket 리포지토리를 사용해 자동 채점을 할 것이다. **자동 채점은 4 월 21 일 금요일 23 시 59 분부터 시작한다.**

제출은 자신의 bitbucket 개인 리포지토리를 이용한다. 프론트엔드는 ~/omok_expand 에, 백엔드는 ~/omok_backend 에 구현한다.

~/omok_expand, ~/omok_backend 를 아래와 같이 통째로 복사한 후, **assn4** 라는 branch 로 push 한다

```
(env) swpp@ip-172-31-30-158:~/swpp-assn201701$ cp -r ../omok_expand/ .
(env) swpp@ip-172-31-30-158:~/swpp-assn201701$ cp -r ../omok_backend/ .
(env) swpp@ip-172-31-30-158:~/swpp-assn201701$ ls
omok_expand omok_backend ..... (다른파일,ex. mysite)
```

채점에 사용할 스크립트는 due 이전에 공지할 것이다.

이번 숙제 역시 채점 환경에서 테스트가 원활하게 돌지 않은 경우, 0 점 처리됩니다.