서버 프로젝트 설계서

6. 컴포넌트 모듈 설계

6.1 컴포넌트 모듈 클래스 구성도 – controller

**WebSocketConfig** (웹소켓 설정)

* 브로커 설정 (구독 및 메시지 전송).
* STOMP 프로토콜을 통해 클라이언트와 서버 간의 통신을 설정.

**ChessController** (실시간 체스 게임 관리)

* 클라이언트 간의 체스 게임 상태 관리.
* 웹과 VR 클라이언트 참여 관리.
* 체스 말 이동 처리.
* STOMP 메시징 템플릿을 사용하여 메시지 전송

6.2 각 클래스들의 설계

6.2.1 멤버 데이터 설명

**WebSocketConfig** **주요 데이터:**

없음.

**ChessController** **주요 데이터**:

gameState: 게임 상태를 관리하는 서비스 클래스 객체

connectedWebSessions: 현재 접속한 웹 사용자 세션을 관리

connectedVRSessions: 현재 접속한 VR 사용자 세션을 관리

6.2.2 멤버 함수 설명

**WebSocketConfig** **주요 메서드**:

* configureMessageBroker(MessageBrokerRegistry config): 메시지 브로커 설정.
* registerStompEndpoints(StompEndpointRegistry registry): STOMP 엔드포인트 등록.

**ChessController** **주요 메서드**:

 Webjoin(SimpMessageHeaderAccessor headerAccessor): 웹 사용자가 게임에 참여하는 기능.

 VRjoin(SimpMessageHeaderAccessor headerAccessor): VR 사용자가 게임에 참여하는 기능.

 processMoveWeb(String move, StompHeaderAccessor headerAccessor): 웹 사용자의 체스 말 움직임을 처리.

 processMoveVR(String move, StompHeaderAccessor headerAccessor): VR 사용자의 체스 말 움직임을 처리.

 handleTimeUp(String message): 웹 사용자가 제한 시간을 초과한 경우 처리

 resetGame(): 게임을 재시작하는 경우 처리

7. 컴포넌트 모듈 설계

7.1 컴포넌트 모듈 클래스 구성도 – dto

**Move (체스 말의 이동)**

* **역할**: 체스 말이 움직인 정보를 관리합니다.

**Position (체스판의 위치)**

* **역할**: 체스판의 위치 좌표를 나타내는 클래스입니다.

**ChessPiece (체스 말)**

* **역할**: 각 체스 말의 타입, 색상, 위치를 관리합니다.

7.2 각 클래스들의 설계

7.2.1 멤버 데이터 설명

**Move (체스 말의 이동)**

 eventTime: 체스 말이 움직인 시간.

 from: 말이 움직이기 전의 위치 (Position 객체).

 to: 말이 이동한 후의 위치 (Position 객체).

 color: 말의 색상 (흑 또는 백).

 type: 말의 종류 (룩, 비숍 등).

**Position (체스판의 위치)**

 x: 체스판의 가로 좌표.

 y: 체스판의 세로 좌표.

**ChessPiece (체스 말)**

 type: 말의 종류 (룩, 비숍 등)

 color: 말의 색상 (흑/백)

 position: 말의 현재 위치 (Position 객체)

7.2.2 멤버 함수 설명

**Move (체스 말의 이동)**

 Move(String eventTime, Position from, Position to, String color, String type): 체스 말의 이동을 생성합니다.

 Move(Position from, Position to): 기본 이동 생성자.

**Position (체스판의 위치)**

 fromString(String positionString): 문자열을 받아 위치 좌표를 생성하는 정적 메서드.

 equals(): 두 위치가 같은지 비교하는 메서드.

 hashCode(): 위치 좌표의 해시코드를 생성.

**ChessPiece (체스 말)**

 equals(): 체스 말의 위치가 같은지 비교.

 hashCode(): 위치에 따른 해시코드를 생성.

8. 컴포넌트 모듈 설계

8.1 컴포넌트 모듈 클래스 구성도 – service

**gamestate** (게임 상태 저장 및c 관리)

controller로부터 전달받은 사용자의 요청을 처리하고 상태를 저장.

8.2 각 클래스들의 설계

8.2.1 멤버 데이터 설명

board  
currentPlayer: 현재 플레이어의 차례를 나타냄(White or Black)  
currentRole: 현재 플레이어의 역할을 나타냄(Commander or Actor)  
whiteKingMoved, blackKingMoved: 캐슬링 가능 여부 판별을 위한 변수  
whiteRook1Moved, whiteRook2Moved: 캐슬링 가능 여부 판별을 위한 변수  
blackRook1Moved, blackRook2Moved: 캐슬링 가능 여부 판별을 위한 변수  
enPassantTarget: 앙파상 처리를 위한 변수

8.2.2 멤버 함수 설명

initializeBoard(): 보드 상태 초기화

processMoveWEB(Move move), processMoveVR(Move move): move 이벤트 처리

getBoardState(): 현재 보드 상태 출력

updateBoard(Move move): 보드 상태 업데이트

switchPlayer(), switchTurn(): 색상 및 역할 변경

isValidMove(Move move, Map<Position, ChessPiece> board): 유효한 움직임인지 확인

isPathClear(Position from, Position to, Map<Position, ChessPiece> board): 경로 상에 장애물이 있는지 확인

isCheckmate(): 체크메이트인지 확인

handleSpecialMoves(Move move, ChessPiece piece): 앙파상, 캐슬링, 프로모션 처리

**기능 시험 명세서 (Functional Test Cases)**

1. **기능 명세서 1: 클라이언트의 체스 게임 참가**
   * **목적**: 사용자가 게임에 성공적으로 참가할 수 있는지 확인.
   * **사전 조건**: 클라이언트가 체스 페이지에 접속.
   * **입력 데이터**: 클라이언트의 연결 요청.
   * **예상 결과**: 클라이언트가 세션에 성공적으로 연결되고, 대기 중인 상대방이 없으면 대기 상태로 전환. 상대방이 있다면 게임이 시작됨.
2. **기능 명세서 2: 체스 말 이동 처리**
   * **목적**: 체스 말의 이동이 규칙에 따라 정상적으로 처리되는지 확인.
   * **사전 조건**: 두 플레이어가 게임에 참가한 상태.
   * **입력 데이터**: 플레이어가 룩을 (0,0)에서 (0,4)로 이동하려는 요청.
   * **예상 결과**: 룩이 (0,4)로 정상적으로 이동하고, 체스판 상태가 업데이트됨.
3. **기능 명세서 3: 잘못된 체스 말 이동**
   * **목적**: 잘못된 체스 말 이동이 시스템에 의해 차단되는지 확인.
   * **사전 조건**: 게임 진행 중.
   * **입력 데이터**: 플레이어가 폰을 규칙에 맞지 않게 (0,1)에서 (0,5)로 이동하려는 요청.
   * **예상 결과**: 요청이 거부되며 "잘못된 움직임" 경고 메시지가 표시됨.
4. **기능 명세서 4: 게임 초기화 기능**
   * **목적**: 게임이 재설정되는지 확인.
   * **사전 조건**: 게임 진행 중.
   * **입력 데이터**: 사용자로부터 게임 초기화 요청.
   * **예상 결과**: 체스판이 초기 상태로 돌아가고, 모든 게임 기록이 초기화됨.
5. **기능 명세서 5: 클라이언트 간 실시간 메시징**
   * **목적**: 체스판 상태와 플레이어 간의 실시간 메시지가 정상적으로 전송되는지 확인.
   * **사전 조건**: 플레이어가 모두 게임에 참가한 상태.
   * **입력 데이터**: 한 플레이어가 체스 말을 이동한 후 상대방에게 메시지 전송.
   * **예상 결과**: 상대방이 즉시 업데이트된 체스판을 확인하고, 전송된 메시지를 받을 수 있음.

**비기능 시험 명세서 (Non-Functional Test Cases)**

1. **비기능 명세서 1: 보안 테스트**
   * **목적**: 체스 게임 시스템이 적절한 보안 조치를 통해 외부 공격을 차단하는지 확인.
   * **사전 조건**: 클라이언트가 웹소켓을 통해 서버에 연결.
   * **입력 데이터**: 비정상적인 웹소켓 연결 요청.
   * **예상 결과**: 서버가 비정상적인 요청을 차단하고, 클라이언트의 비정상적인 접근이 방지됨.
2. **비기능 명세서 2: 사용성 테스트**
   * **목적**: 사용자가 직관적으로 게임을 플레이할 수 있는지 확인.
   * **사전 조건**: 웹 클라이언트가 체스 게임에 접속.
   * **입력 데이터**: 사용자가 게임 UI를 통해 체스 말을 이동하는 과정.
   * **예상 결과**: 사용자가 직관적으로 UI를 이용해 게임을 진행할 수 있으며, 모든 기능이 쉽게 접근 가능함.
3. **비기능 명세서 3: 가용성 테스트**
   * **목적**: 시스템이 다운타임 없이 안정적으로 작동하는지 확인.
   * **사전 조건**: 서버가 정상 작동 중.
   * **입력 데이터**: 체스 게임을 24시간 연속 실행.
   * **예상 결과**: 시스템이 중단 없이 24시간 동안 정상적으로 동작하며, 모든 클라이언트 요청이 처리됨.