Online Game Server

Togater

차준범
이태형
김가람
남경현

목차

1. Introduction

Project Information

본 프로젝트는 로그인, 로비(채널), 채팅, 게임 로직을 처리하는 분산 게임 서버 시스템을 개발합니다. 팀원들은 시스템을 구성하는 서버를 하나씩 담당하고 서버 고유의 기능을 개발함으로써 개인의 역량을 향상시키고 팀 프로젝트를 완성시키는 것이 목적입니다.

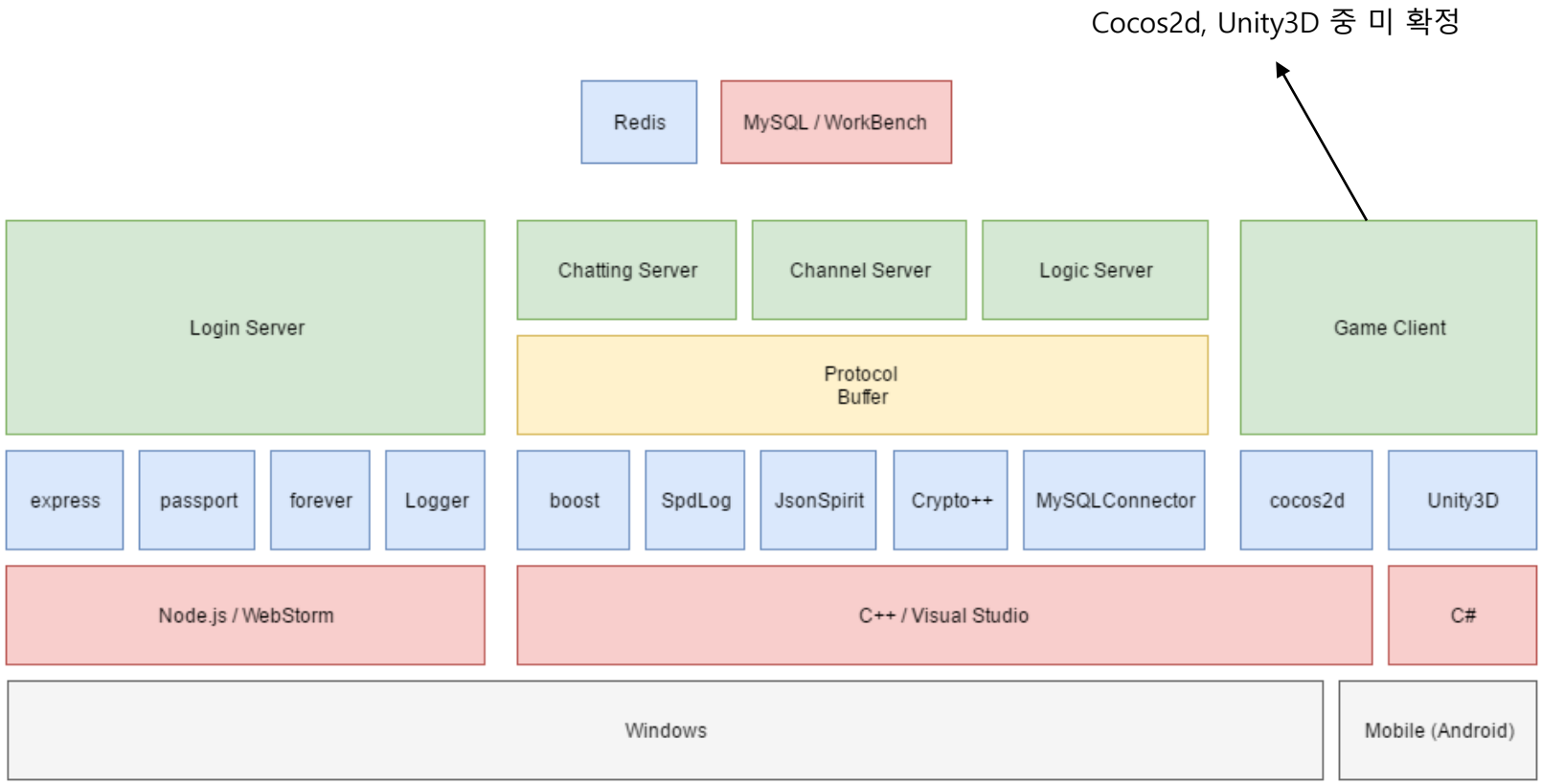
1. Introduction

Game Information

1. 게임 클라이언트는 인디언 홀덤을 1:1로 네트워크 상에서 진행한다.
   (Rule 설명 : https://www.youtube.com/watch?v=vcfTvU7wkL4)
2. 게임은 친선 경기와 랭크 게임 두 가지 방식으로 진행 가능하다.
   1) 친선 경기는 특정 유저와 전적에 결과를 주지 않으면서 게임을 진행할 수 있다.
   2) 랭크 경기는 전적을 통한 점수를 통해 매칭 되며 매 게임 결과가 반영 된다.
3. 게임 계정은 자체 회원 가입과 Facebook 인증을 통한 로그인 두가지 방법을 허용 한다.
4. 채팅은 로비, 게임, 귓속말, 전체 공지 등을 사용 가능 하다.
5. 최대 5명의 친구를 등록할 수 있다.

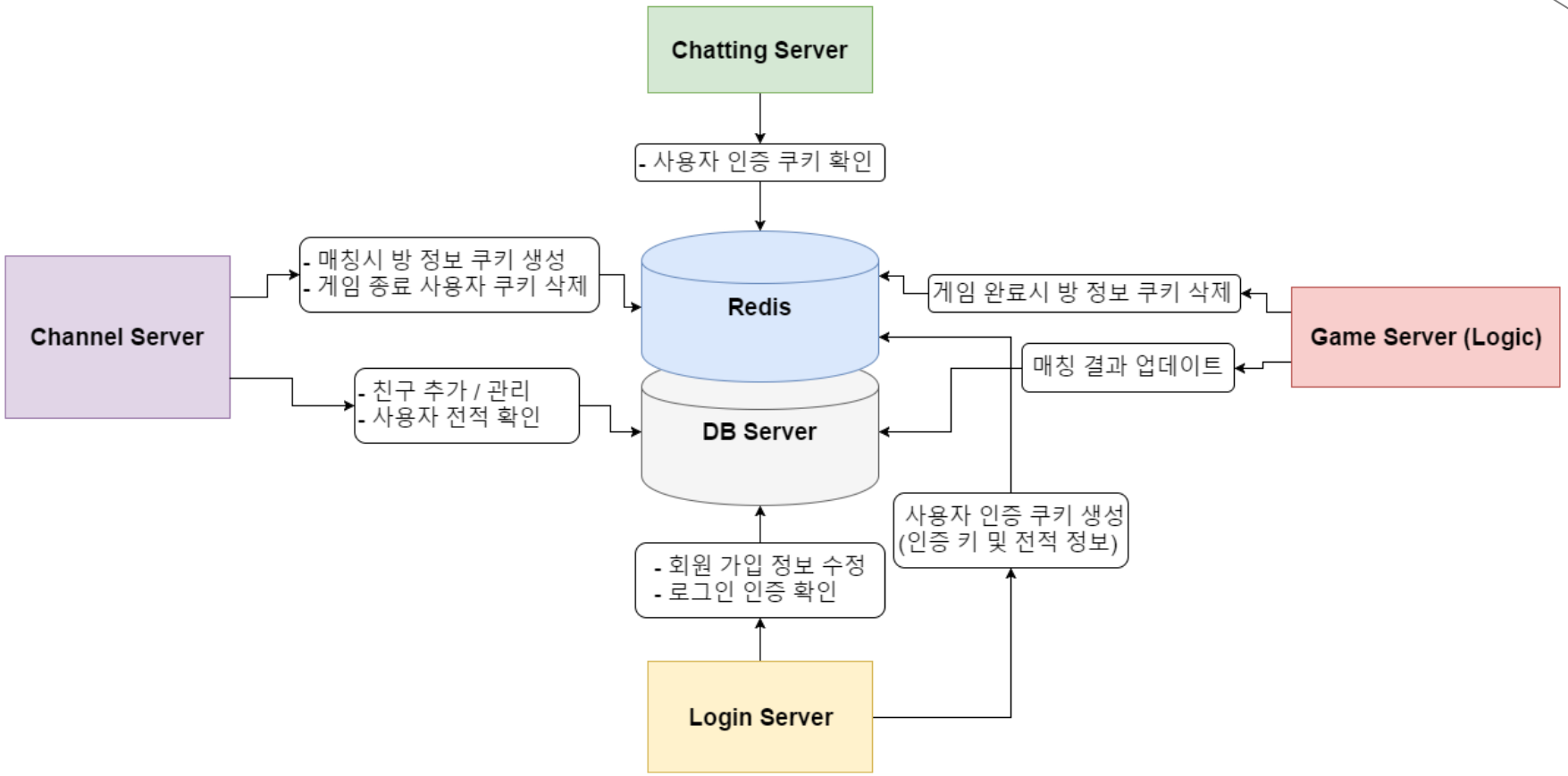
2. Architecture

Environment, Library…



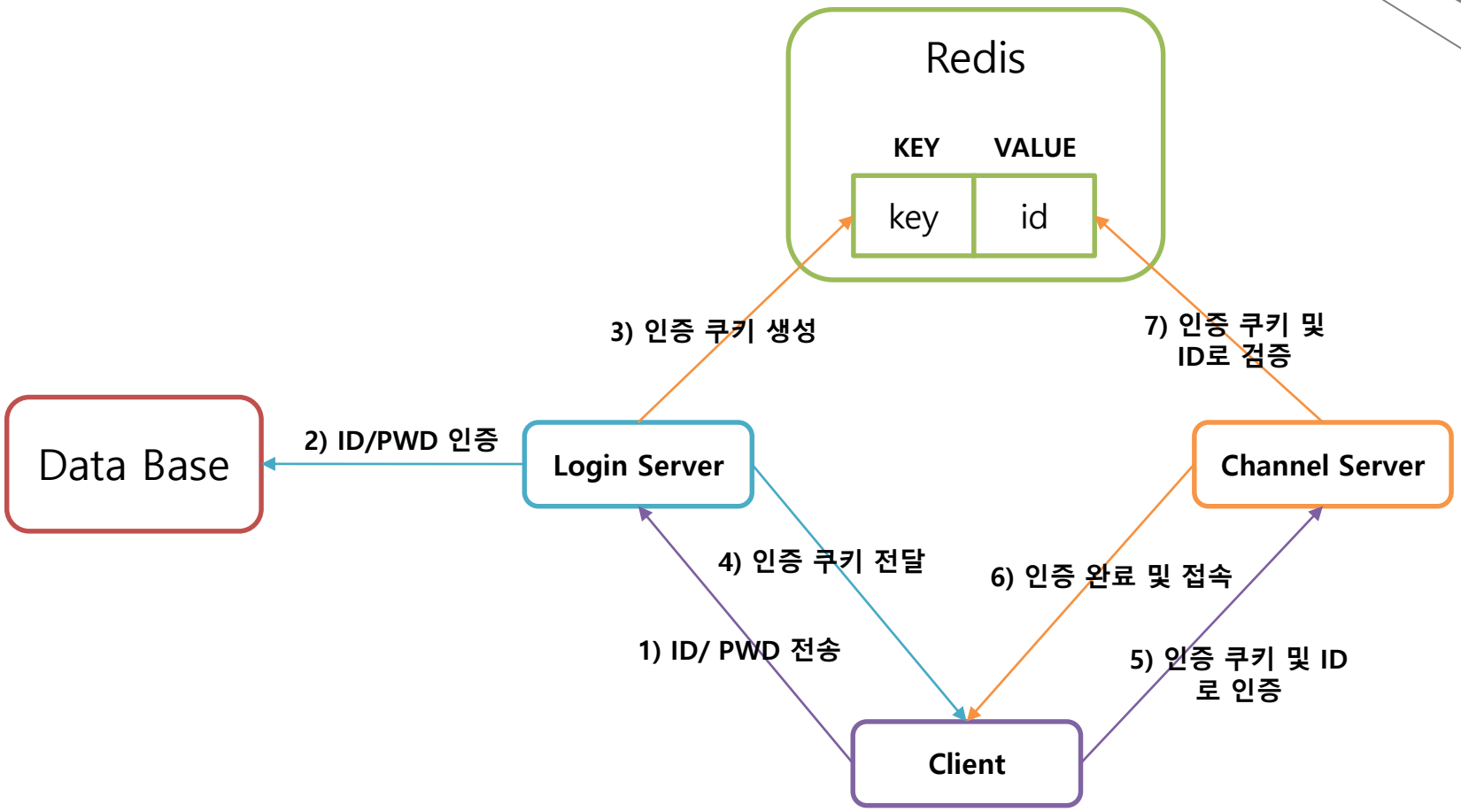
3. Server/Client Feature

Relation Diagram



3. Server/Client Feature

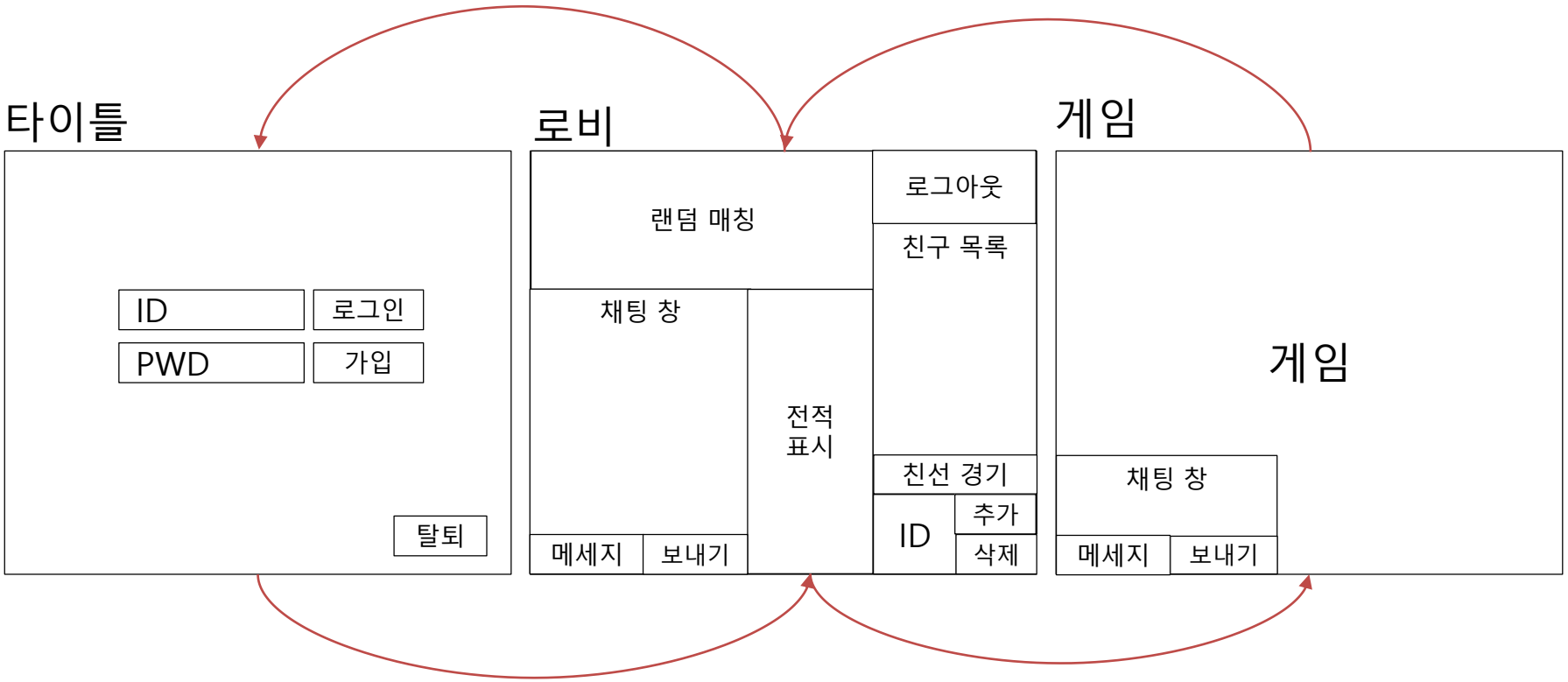
Redis Authenticationcookie



추가로 인증 쿠키 이외에도 게임 방 정보도 Redis를 통해 처리

3. Server/Client Feature

Client Layout



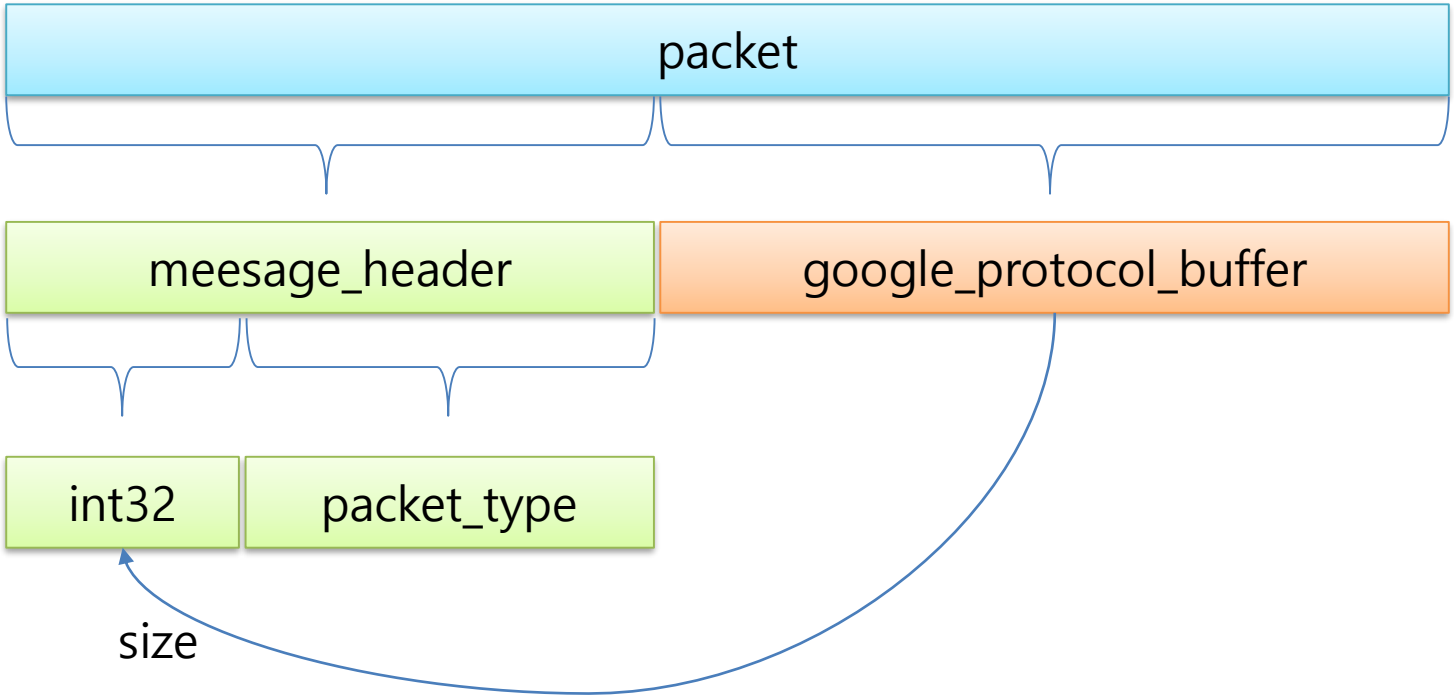
3. Server/Client Feature

Client Feature

분류	조건	내용	결과
타이틀	로그인	ID, PWD를 입력하여 로비 입장.	로비 화면으로 이동
	가입	가입 다이얼로그 표시. ID, PWD, Email입력.	가입 다이얼로그 종료
	탈퇴	탈퇴 다이얼로그 표시. ID, PWD, Email입력.	탈퇴 다이얼로그 종료
로비	랜덤 매칭	전적 레이팅 점수를 통한 랭크 게임 자동 매칭 수행.	게임 화면 입장.
	보내기	채팅 메시지를 보냄. 타입은 귓속말, 일반 채팅이 존재. 일반 채팅은 로비 인원들과 진행.	귓속말의 경우, 성공 유무 표시.
	로그아웃	버튼 클릭	타이틀 화면으로 이동
	친선 경기	선택한 친구와 게임 진행	게임 화면 입장.
	추가	친구 추가	전체 친구 수가 5인 이하인 경우, 친구가 추가됨.
게임	보내기	채팅 메시지를 보냄. 타입은 귓속말, 일반 채팅이 존재.일반 채팅은 게임 룸 인원과 진행.	귓속말의 경우, 성공 유무 표시.
	게임 진행 UI	기획중	

4. Packet Structure

Google Protocol Buffer



로그인 서버와 클라이언트는 http 프로토콜을 사용하여 통신 합니다.

5. Packet Information

Packet Type/Value Definition

TCP/IP 패킷 정보

서버 이름	송신자	수신자	패킷 타입 (Enum)	패킷 설명	값1		값2		값3		값4	
					이름	타입	이름	타입	이름	타입	이름	타입
channel_server	C	S	packet_friends_req	클라이언트의 친구 추가,삭제,찾기에 대해 처리	req	enum(F_REQ)	user_id	string				
	S	C	packet_friends_ans	친구 관리 요청에 대한 응답	ans	enum(F_ANS)	user(repeated)	user_status				
			user_status		user_id	string	rating	enum(RATING)	on	bool		
	C	S	packet_play_req	게임 플레이 요청 (친선경기, 랜덤매칭)	req	enum(P_REQ)	user_id(optional)	string				
	S	C	packet_play_ans	매칭된 게임방,상대플레이어에 대한 정보 메시지 응답	ans	enum(P_ANS)	user_id(optional)	string	room_num(optional)	int32	yes	bool
	C	S	packet_join_ntf	로비 입장에 대한 알림	token	int32						
	C	S	packet_logout_ntf	로그 아웃에 대한 알림	token	int32						
logic_server	C	S	packet_enter_req	로직 서버에 대한 클라이언트의 인증 유효성 검사를 실시합니다.	player_key	int32	room_key	int32				
	S	C	packet_enter_ans	로직 서버가 실시한 클라이언트 유효성 검사의 결과를 전송합니다.	result	int32						
	S	C	packet_game_state_ntf	게임 로직의 단계를 전송합니다. (0 : Ready, 1 : Playing, 3 : End)	state	int32						
	S	C	packet_process_turn_req	현재 유저의 금액을 반환합니다.	total_money	int32						
	C	S	packet_process_turn_ans	배팅 금액을 전송합니다.	player_key	int32	money	int32				
	S	C	packet_process_turn_ntf	배팅 결과를 전송합니다.	public_card_number1	int32	public_card_number_2	int32	opponent_card_number	int32	money	int32
chat_server	Echo		packet_chat_normal	일반 채팅 보내기	user_id	string	chat_message	string				
	Echo		packet_chat_whisper	귓속말 보내기	user_id	string	target_id	string	chat_message	string		
	Echo		packet_chat_room	게임방 채팅 보내기	user_id	string	chat_message	string				
	Echo		packet_chat_notice	공지 채팅 보내기	user_id	string	chat_message	string				

5. Packet Information

Packet Type/Value Definition

HTTP 프로토콜 (Login Server - Client)

기능	HTTP 메서드	HTTP URL & Body
로그인	HTTP GET	/login
회원가입	HTTP POST	/register
회원탈퇴	HTTP POST	/drop

6. Class Diagram

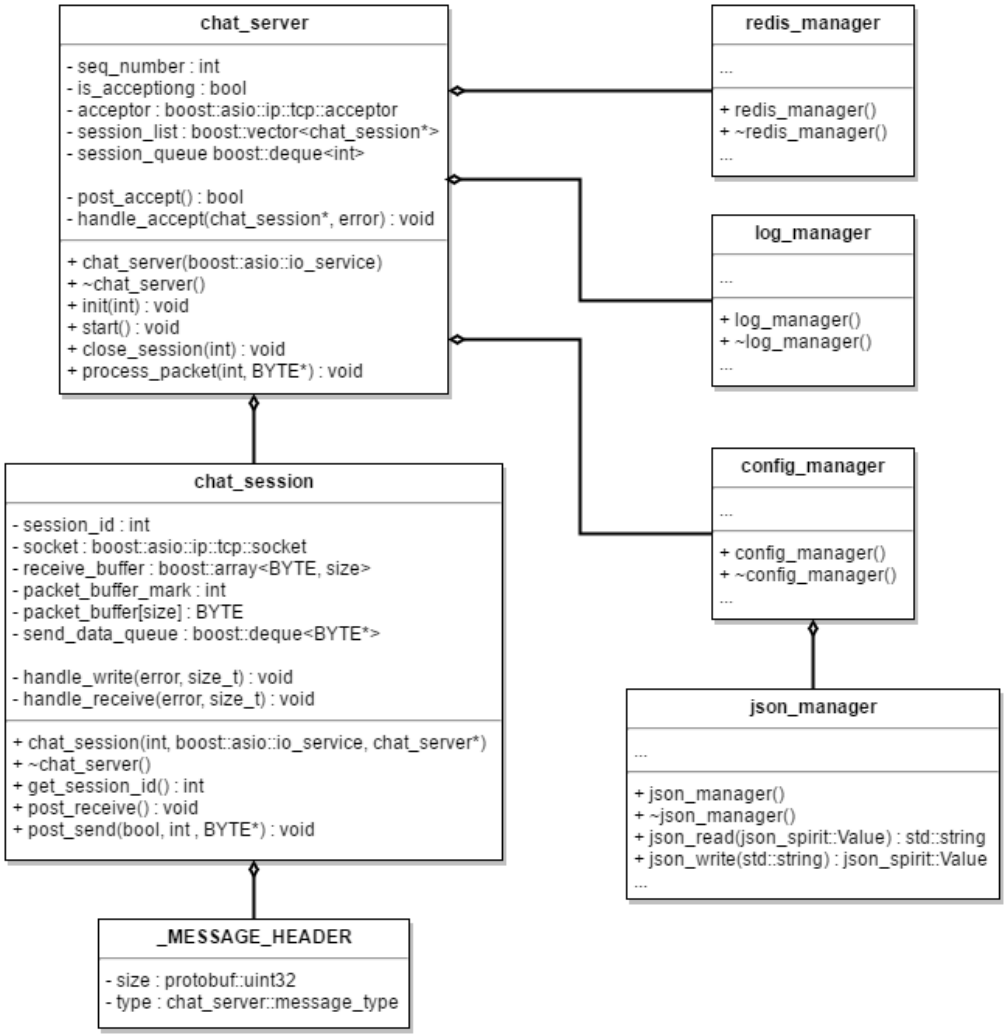
Login Server

```
app.post('/register', function (request, response) {
    client.query('select * from userinfo where id = ?', [request.body.id] , function(err, rows){
        if(rows.length === 1)   //id 중복
        {
            response.writeHead(200, {'Content-Type' : 'text/html'});
            response.end('fail');
        }
        else if(rows.length === 0)  //회원가입
        {
            response.writeHead(200, {'Content-Type' : 'text/html'});
            response.end('succ');
            console.log(request.session.id);
            client.query('insert into userinfo (id, password, win, lose) values(?, ?, ?, ?)',[request.body.id, request.body.password, 0, 0]);
        }
    });
});

app.get('/login', function (request, response) {

    var originalAuth = request.headers.authorization;

    var parts = originalAuth.split(' ');
    var auth = new Buffer(parts[1], 'base64').toString().split(':');

    var id = auth[0];
    var pass = auth[1];

    client.query('select * from userinfo where id = ? and password = ?', [auth[0], auth[1]] , function(err, rows){

        if(rows.length === 1)   //만약 로그인에 성공했다면
        {
            response.writeHead(200, {'Content-Type' : 'text/html'});
            response.end('login ok');
            request.session.userID = auth[0];

            console.log(request.session.id);
            console.log(request.session.userID);
            redisClient.set(request.session.id, request.session.userID); //레디스에 request.session.id 와 userID를 쌍으로 저장

            redisClient.get(request.session.id, function (err, reply) {
                console.log(reply);
            });
        }
        else if(rows.length === 0)  //로그인 실패
        {
            response.writeHead(200, {'Content-Type' : 'text/html'});
            response.end('login fail');
        }
    });
});
```

Node.js는 작성 코드로 Diagram을 대체 합니다.

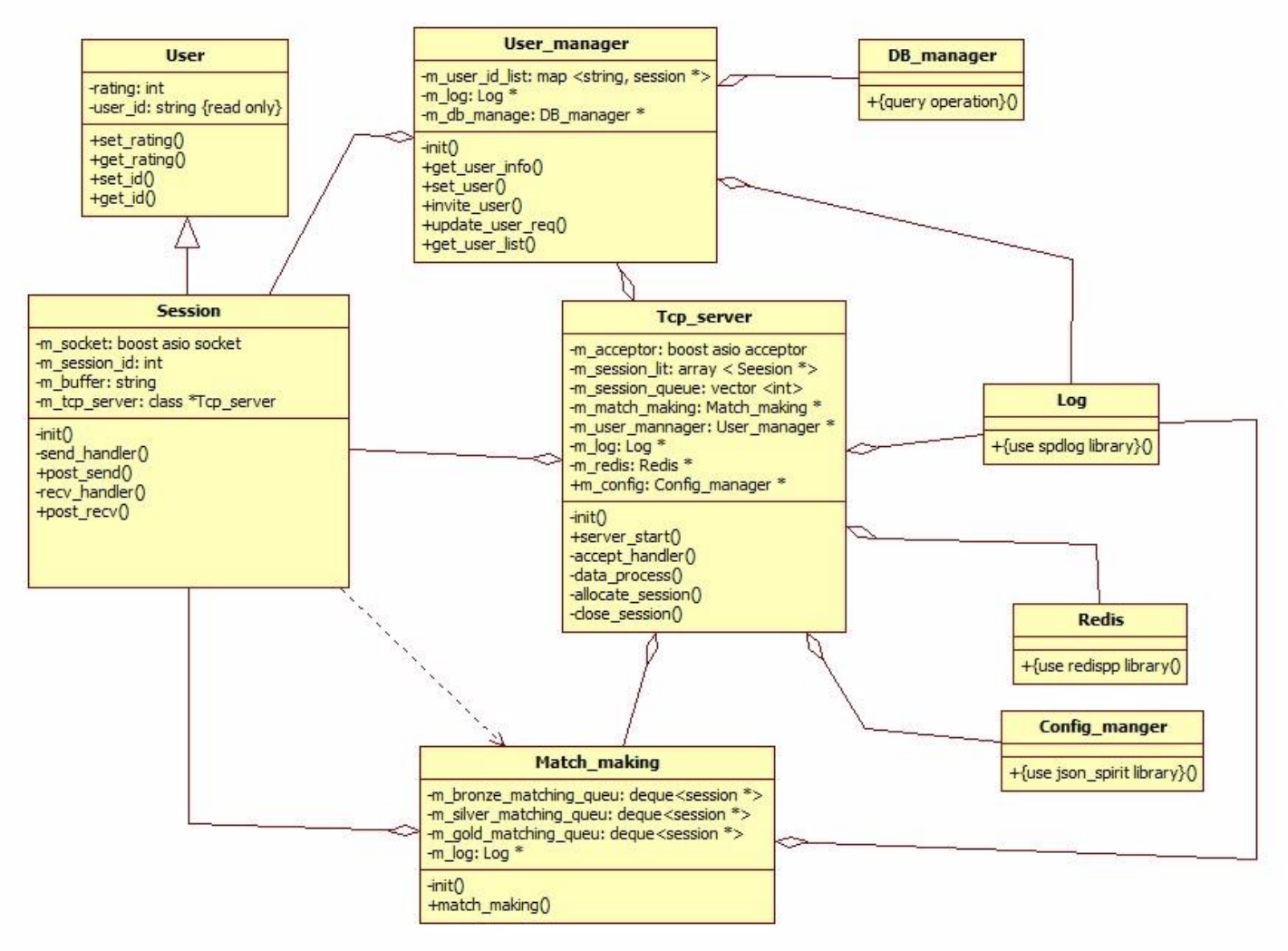
6. Class Diagram

Chat Server



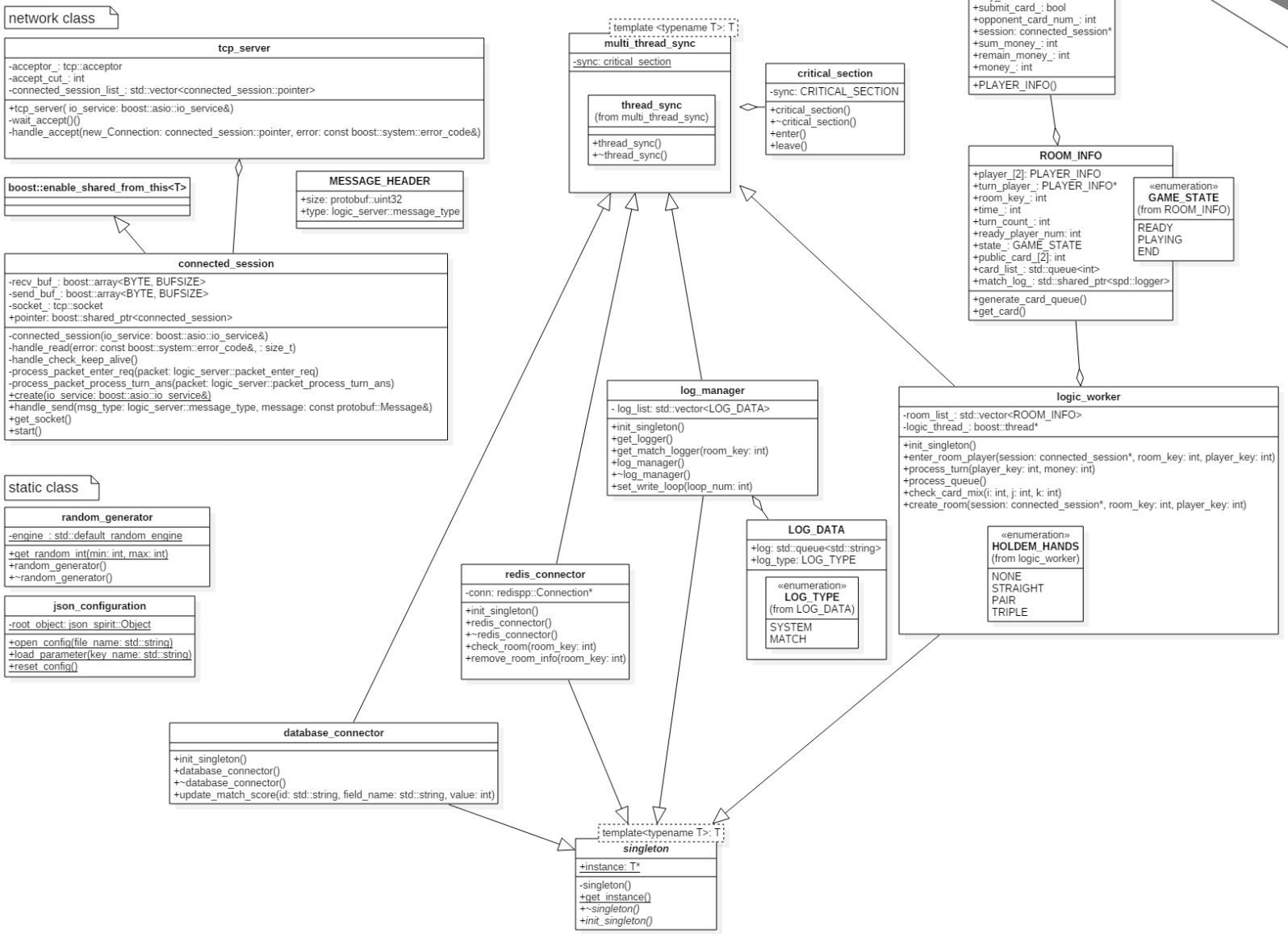
6. Class Diagram

Lobby Server



6. Class Diagram

Logic Server



감사합니다