ChatSparrow 0.1.0v

다자간채팅프로그램

학　　과 : 컴퓨터 미디어 융합
목　　표 : 채팅프로그램 구현
학　　번 : 201133271
이　　름 : 조　　용　　진
완 료 일 : 2015.11.27

목 차

1. 서론

1.1 배경

ChatSparrow, 새들이 지저귀는 것을 twitter 라고 하지만, 채팅을 하는 주체는 참새(Sparrow) 이므로, 채팅과 참새를 합쳐 ChatSparrow 라고 하였다. 이 단어는 캐리비언 해적에 나오는 주인공 JackSparrow와 발음이 비슷하여, 채팅의 테마를 해적에 두었다. 즉 해적들의 채팅을 테마로 서비스를 정하였다.

해적을 테마로 채팅프로그램을 구현하였기 때문에, 해적의 대표적인 규칙이 채팅에 적용하였다. 특정한 이벤트를 할 경우 해적인 사용자는 돈을 벌게 된다. 그 이벤트는 크게 5가지 이다.

첫 번째, 해적은 해적끼리 싸움을 할 때 개인과 개인 이렇게 싸우기보다 팀원들끼리 싸우므로, 만약 같은 팀원이 팀명으로 방을 만든다면 같은 팀명을 사용하는 사용자는 방을 생성할 수 없다. 또한 방을 만드는 사용자에게는 포인트(돈)이 적립되게 된다.

두 번째, 해적은 해적단의 싸움에 참가할 때 마다 약탈을 한다. 그렇기 때문에 사용자가 채팅방에 입장할 때 마다 포인트를 얻을 수 있다.

세 번째, 사용자는 메시지를 보낼 때 마다 돈을 얻는다.

네 번째, 해적이 싸움에 패배하거나 포기하여 전투에서 후퇴하게 될 경우 되려 자원을 잃는다. 그러므로 사용자가 채팅방을 나오게 될 경우 포인트가 차감된다.

다섯 번째, 해적이 잠깐 쉬고 있으면 식량만 축내기 때문에 자원이 줄어든다. 그러므로 사용자가 로그아웃을 하면 채팅방을 나와서 차감되는 양보다 더 크게 차감된다.

여섯 번째, 해적이 피를 일정량 이상 흘리게 되면 죽는다. 그것을 적용하여 사용자의 돈이 -2000이 될 경우에 계정을 삭제 시킨다.

이러한 6가지 룰을 다자간 채팅프로그램에 적용하여 해적을 테마로한 ChatSparrow 0.1.0v를 기획하고 구현하였다.

1.2 주요 기능 및 범위

1) 클라이언트 주요 기능

- **로그인**

사용자는 회원 가입한 아이디와 비밀번호를 이용해 서버로 로그인을 요청할 수 있다.

- **회원가입**

사용자는 회원가입을 할 수 있으며 해적단 이름, 아이디, 비밀번호를 이용하여 가입할 수 있다.

- **로그인 유효성 검증**

클라이언트에서의 유효성 검증은 문자열의 자리 수를 이용한다. 올바르게 입력하지 않으면 Alert Dialog가 출력된다.

- **회원가입 유효성 검증**

클라이언트에서의 회원가입 유효성 검증은 문자열의 자리 수를 이용한다. 올바르게 입력하지 않으면 회원가입 버튼이 Disabled 된다.

- **Alert Dialog 알림**

ChatThread에서 UI 객체를 생성자로 넘겨받아서 지정된 UIFrame에서 Alert Dialog가 출력된다. ChatThread 에서는 UI객체를 가지고 있어야 한다.

- **대화방 만들기 및 종료**

해적단의 이름으로 채팅방이 만들어져 있지 않다면 대화방이 생성되고 포인트가 오른다. 그렇지 않다면 Alert Dialog가 출력되고 생성이 되지 않는다.

- **대화방 입장 및 퇴장**

사용자가 대화방에 입장하거나 퇴장 할 때마다 포인트가 증가되거나 차감되며 사용자들의 리스트를 실시간으로 받아볼 수 있다.

- **사용자 상태 확인**

사용자는 처음 로그인해서 대화방 리스트 화면에 머무르게 되면 사용자의 해적단 이름, 아이디, 포인트, 가입일 이 보이며 포인트는 동기화 된 상태이다.

- **일반 대화, 귓속말 대화, 전체 대화**

일반 대화는 메시지를 타이핑하고 엔터를 누르면 작동되며 귓속말은 /w [아이디] [할 말], 전체 채팅은 /! [할 말]을 이용하면 된다.

2) 서버 주요 기능

- **로그인 유효성 검증**

클라이언트의 로그인 요청에 대한 유효를 검사한다. 기본 적인 유효성 검증 이외에 회원이 로그인중이거나 존재하지 않거나에 대한 예외 상황에 대해 Alert를 출력하도록 메시지를 보낸다.

- **회원가입 유효성 검증**

클라이언트의 회원가입 요청에 대한 유효성을 검사한다. 아이디에서는 이미 등록되어 있는지에 대한 유효성을 검사하고, 해적단에 관련한 가입처리는 트랜잭션을 이용하여 데이터베이스의 무결성을 추구하도록 한다.

- **사용자 세션 관리**

사용자의 세션관리는 이미 접속되어 있는 사용자면 로그인이 불가능하게 막으며, 로그인으로 채팅프로그램에 접속한 사용자들의 접속정보를프로그램의 메모리 영역에서 관리한다. 그 이외의 사용자인 “Anonymous”들에 대해서는 관리하지 않는다. 또한 사용자의 의도적인 종료와 예외적 종료 둘 다의 상황에 대비하여 서버 프로그램의 메모리영역에 저장되어있는 특정 유저의 자원을 해제한다. 또한 DB에서는 특정 사용자의 접속의 여부에 관련된 속성값을 변경해준다.

- **사용자 이용 정보 관리**

사용자의 이용 정보 역시 서버 프로그램 메모리 영역에서 관리하며, 로그아웃이 되면 자원관리를 위해 자원을 해제하며, 사용자가 어느 방에 들어가 있는지에 대한 정보도 가지고 관리한다. 또한 서버는 사용자들의 정보를 계속해서 방송 해준다.

- **포인트 적립 및 차감**

서버 프로그램은 사용자의 특정한 이벤트에 반응하여 포인트를 추가적립해주거나 차감한다. 1, 채팅방 생성, 2, 채팅방 입장, 3, 대화할 경우 4, 채팅방 퇴장 5, 로그아웃

- **사용자 메시지 전달**

사용자의 메시지를 수신하여 메시지가 귓속말, 전체 그리고 일반 인지 구분하여 요청한 메시지 전달 방식에 알맞은 동작을 실행한다. 이 때 서버 프로그램은 사용자 이용 정보를 참조하여 메시지를 전달한다.

- **로그 출력**

서버 프로그램은 어떤 이벤트가 발생 하더라도 로그 메세지를 출력한다.

1.3 프로젝트 일정

	9월	10월	11월/1주	11월/2주	11월/3주	11월/4주
구상	23일 구상시작	아이템 선정				
기획		이용 규칙 기획	이용 배경 정의	요구사항 정리		
설계		이용 툴 선정	아키텍쳐 구상	프로토콜 메시지 정함		
구현				MySQL Maven 이용	구현 완료	
보고서					보고서 작성	제출

2. 배경 지식

2.1 관련 지식

1) 자바

자바는 자바 컴파일러가 자바 언어로 작성된 프로그램을 바이트코드라는 특수한 바이너리 형태로 변환한다. 바이트코드를 실행하기 위해서는 JVM(자바 가상 머신, Java Virtual Machine)이라는 특수한 가상 머신이 필요한데, 이 가상 머신은 자바 바이트코드를 어느 플랫폼에서나 동일한 형태로 실행시킨다. 때문에 자바로 개발된 프로그램은 CPU나 운영 체제의 종류에 관계없이 JVM을 설치할 수 있는 시스템에서는 어디서나 실행할 수 있다.

2) MySQL

오픈 소스의 관계형 데이터베이스 관리 시스템(RDBMS)이다. 다중 스레드, 다중 사용자 형식의 구조질의어 형식으로 되어있으며, 데이터 스토리지의 메인 메모리에 설치되어 운영되는 방식의 데이터베이스 관리 시스템이다.

3) Maven

아파치 메이븐(Apache Maven)은 자바용 프로젝트 관리 도구이다. 아파치 앤트의 대안으로 만들어졌다. 아파치 라이선스로 배포되는 오픈 소스 소프트웨어이다. 라이브러리는 Maven Repository 서버에 있으며 POM.xml 에 기술을 하는 즉시 라이브러리가 동기화 되어 프로젝트 폴더에 자동으로 설치된다.

4) JSON

JSON(제이슨, JavaScript Object Notation)은 속성-값 쌍으로 이루어진 데이터 오브젝트를 전달하기 위해 인간이 읽을 수 있는 텍스트를 사용하는 개방형 표준 포맷이다. 비동기 브라우저/서버 통신 (AJAJ)을 위해, 넓게는 XML(AJAX가 사용)을 대체하는 주요 데이터 포맷이다. 특히, 인터넷에서 자료를 주고 받을 때 그 자료를 표현하는 방법으로 알려져 있다. 자료의 종류에 큰 제한은 없으며, 특히 컴퓨터 프로그램의 변수값을 표현하는 데 적합하다. 본래는 자바스크립트 언어로부터 파생되어 자바스크립트의 구문 형식을 따르지만 언어 독립형 데이터 포맷이다.

2.2 배경 지식

1) 동기 소켓 통신과 비동기 소켓 통신

동기화 소켓

-동기화 소켓의 경우는 한쪽이 write이면 다른 한쪽은 read로서 read대한 작업만을 처리한다.

-그러므로 write하는 곳에서 자료를 다 보낼 때 까지 read하는 쪽은 다 받을 때 까지 Block(다음 작업을 처리하지 않음) 되어 있어야한다.

-클라이언트 측에서는 다 받을 때 까지 기다리고 난후 다음 일을 하면되겠지만, 서버측에서는 read에만 집중할 수 없고 다른 일을 해야한다. 즉 클라이언트의 다른 요청에 대해서 Listen도 해야하고, 기타 다른 처리도 해야하기 때문이다.

-서버측에서는 Forking(자기 자신을 복제함)을 하든가 Threading을 하던가해서 현재의 일을 하나의 Thread(Forking의 경우는 Process)에게 맡기고 다시 서버의 본질적인 작업을 실행하게 된다.(자바의 경우는 Threading을 한다.)

-Forking은 유닉스 계열에서 사용하는 방법이고, Threading은 윈도우즈 계열에서 사용하는 방법이다. 단순 서빙이나 안정성에 있어서는 Forking이 좋은 방법이나 정보 교환(게임 서버의 경우)을 할 이유가 있을 때 는 Threading이 훨씬 효율적이다.

-Thread의 경우는 독자적으로 움직이기는 하지만 하나의 Process내에 있어서 공용 변수를 설정/사용하기가 쉽다. 다만 동기화(이 동기화는 Thread 동기화를 의미 함)에서 섬세한 작업을 요구한다. Thread-Safe한 객체에 대해서는 굳이 동기화를 시킬 필요는 없다.

-Fork의 경우에는 각 Process가 다른 주소 공간에 있기 때문에 이런 작업을 위해 공유 메모리나 Signal을 사용해한다. 사용하는게 그다지 어렵지는 않지만 디버깅하는 것도 까다롭고 여러모로 까다롭다.

비 동기화 소켓

-비동기화 소켓의 경우는 한쪽이 Write를 하건 뭘 하건간에 read하는 쪽과 무관하게 동작한다.
-Write를 하는 쪽은 그냥 알아서 Write하고 받는 쪽은 그냥 받는대로 read한다.
-소켓 라이브러리에서 받는대로 현재까지 받은 자료를 상위 프로그램에 event와 함께 알려주는데, 이런 이유는 비동기 소켓은 Threading을 할 필요가 없기 때문이다. 그러므로 실제로 비 동기화 소켓을 구현하는데 에는 생산력이 뛰어나다.
-그러나 댓가가 따릅니다. 한쪽에서 ABC1234567890 이라는 자료를 날렸다고 하면, 받는 쪽에서는 ABC1234567890라는 Data를 event와 함께 받으면 아무런 고민할 것도 없지만, 절대 이 데이터에 대한 무결성을 보장해주지 않는다.
-ABC1234567890이 ABC1234 한번 읽고, 567890 요렇게 읽고 해서 두 번 읽을 수도 있고, ABC1234567890 다음에 PQR7777이라는 자료를 잽싸게 보냈다면, ABC1234567890PQ 읽고, R7777 이렇게 읽을 수도 있다.
-이것의 이유에는, TCP는 Stream으로 다루어지며 그 Stream의 스케줄은 전적으로 TCP를 관장하는 소켓 라이브러리가 하기 때문인데, 이렇게 읽은 자료를 사용하기 위해서는 읽은 자료를 순서대로 저장하여 모은 후 다시 Parsing을 해야한다. 즉, parsing을 하기위한 간단한 프로토콜을 사용자가 다시 만들어서 사용해야 한다. (그러나 좀 하다보면 parsing을 하는 것도 그다지 어렵지 않다.)
-만약 비동기 소켓을 사용하고도 한쪽에서 ABC1234567890를 보내면 받는 쪽도 ABC123456 7890를 항상 받을거라고 생각하고 코딩한 프로그램이 있다면 그건 손을 아주 많이 봐야한다는 말이다.

결론

-"동기화 소켓+Threading"과 "비동기화 소켓"은 모두 장단점을 가지고 있지만 TCP의 동기화와 비동기화 작동 원리에 대한 완벽한 이해가 없다면 "동기화 소켓+Threading"를 사용해야 한다. "비동기화 소켓"은 위에서 말했다시피 자료 재조합과정과 프로토콜 파싱 과정이 필요하며, 클라이언트가 n명이라면 n개가 필요하게 됩니다. 이러한 사항에 대하여 구현하면 되지만, 고려 하지 않고 만들기 때문에 문제가 된다. 그러므로 이것만 제대로 만들고 사용한다면, Seria-lization이나 Data-Binding에서 동기화 방식보다 나은 성능과 깔끔한 코딩 복잡도가 나오게 됩니다.

2) 자바 소켓 프로그램의 구현 방법론

1, 최소한 N개의 사용자와 1개의 서비스 서버가 존재해야한다.

n개의 사용자가 존재하고 서비스를 이용하기 위해서는 서버는 사용자의 소켓과 스레드를 수용(Accept)하고 제2 이상의 사용자의 소켓을 수용 할 준비를 해야한다.

2, 사용자의 접속 세션에 대한 식별하여 사용할 수 있어야 한다.

서버는 사용자의 세션을 저장하고 그 저장된 세션에게 고유한 ID 값을 주어 특정한 사용자에게만 데이터를 송신할 수 있어야만 한다. 세션은 DB의 사용자 튜플에서 의존을 갖게 되지만 서버가 종료했는대도 세션이 유지될 필요가 없기 때문에 휘발성 메모리 영역에 저장을 해야한다.

3, 클라이언트가 동작을 정지하더라도 지속적으로 변화에 대한 값을 갱신해야 한다.
 서비스를 운영할 때 클라이언트의 동작이 발생하지 않는다 하더라도 서버는 지속해서 클라이언트에게 서비스의 상태를 BroadCasting 해줘야 하며 클라이언트는 지속적으로 서비스의 상태를 표시해야한다. 이를 구현하기 위해
 서버는 클라이언트에게 아무런 요청 값을 받지 못하면 어떠한 값도 내보내지 않게 되는데, 이러한 점을 극복하기 위해선 다른 클라이언트에게 요청값을 받았을 때, 아무런 동작하지 않고있는 클라이언트들에게 응답 패킷을 전송해주어야 한다.

4, 메시지의 타입을 구분하기위해 헤더의 포맷을 정해야 한다.

 특정 프로토콜이 메시지를 주고 받을 때 해당 메시지가 어떤 용도의 메시지이고 어느 곳으로 흘러야 하는지에 대한 구분을 하기 위해서는 헤더를 구분해야만 한다. 단순한 채팅 프로그램이라면 헤더에다가는 사용자의 식별 ID와 요청되는 메시지의 타입만 지정하면 된다.

3. 시스템 구축

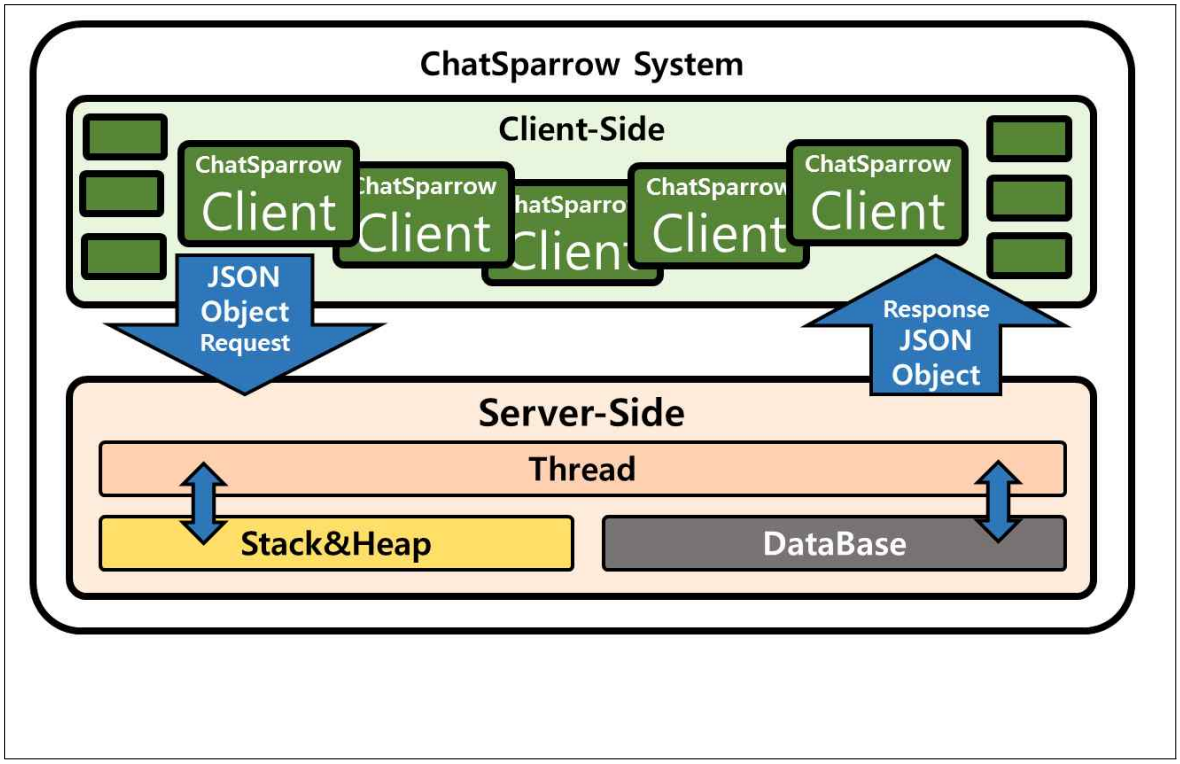
3.1 개발 및 시스템 환경

	개발 환경	
	Client	Server
H/W	CPU i7-4500U @1.80GHz |2.40GHz RAM - 8GB	CPU Intel(R) Xeon E5504 @2.00GHz RAM - 8GM
S/W	OS - Microsoft Windows 8.1 Tool - Eclipse Spring	OS - Linux CentOS7.0 Tool - putty.exe

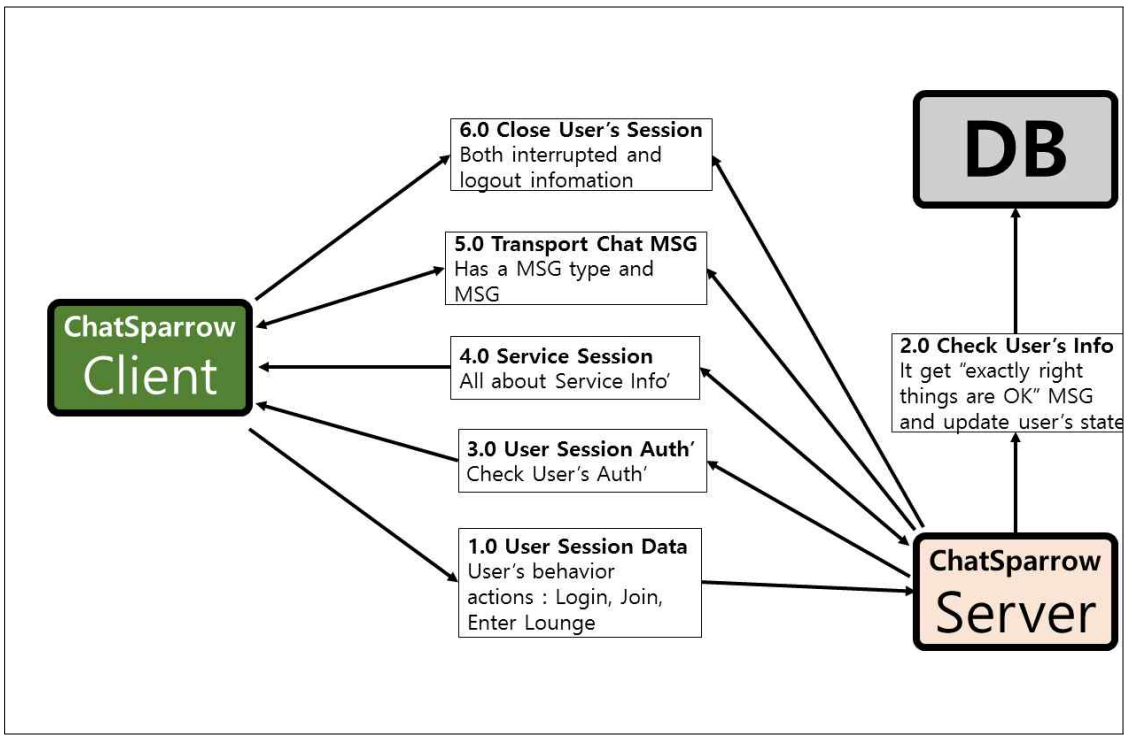
	테스트 환경	
	Client	Server
H/W	CPU i7-4500U @1.80GHz |2.40GHz RAM - 8GB	CPU Intel(R) Xeon E5504 @2.00GHz RAM - 8GB
S/W	OS - Microsoft Windows 8.1 Tool - Eclipse Spring[sts]	OS - Linux CentOS7.0 Tool - putty.exe

3.2 시스템 구성

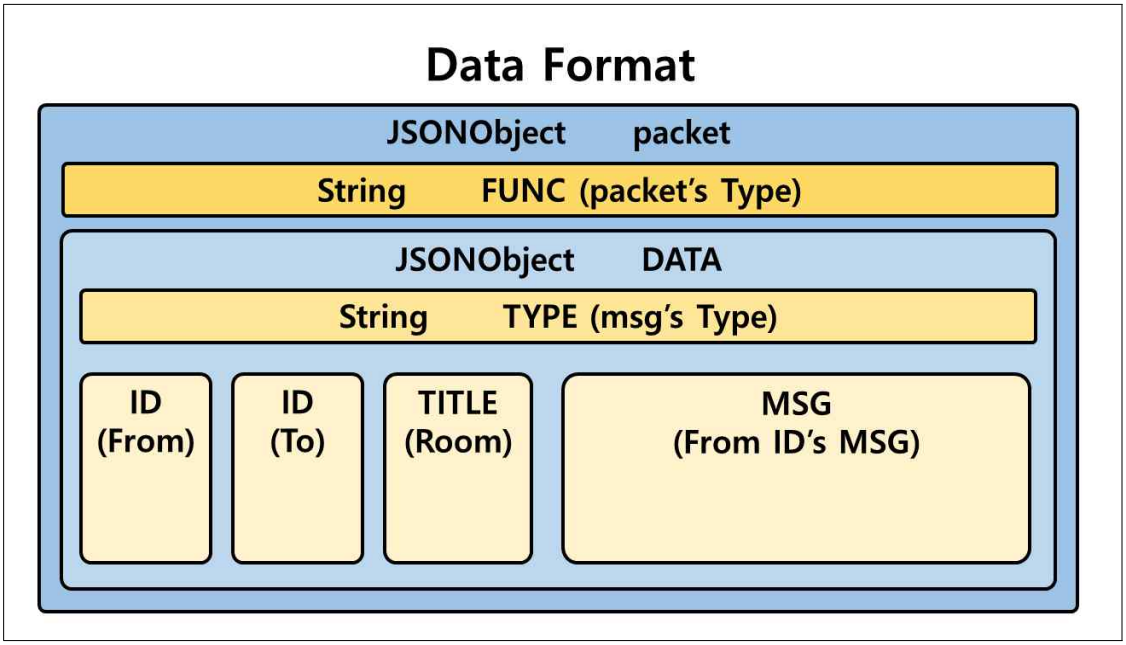
1) 시스템 구성도



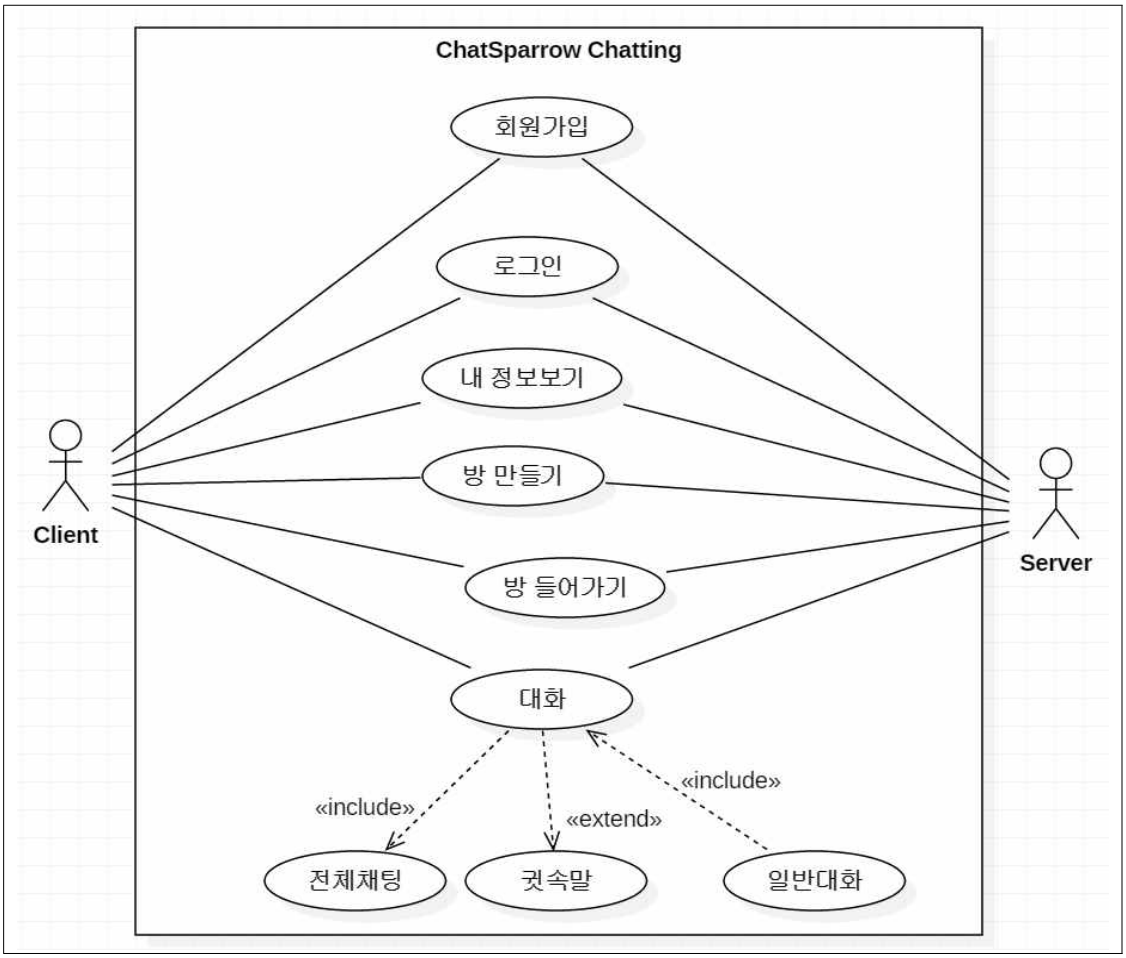
2) DFD



3) DataFormat



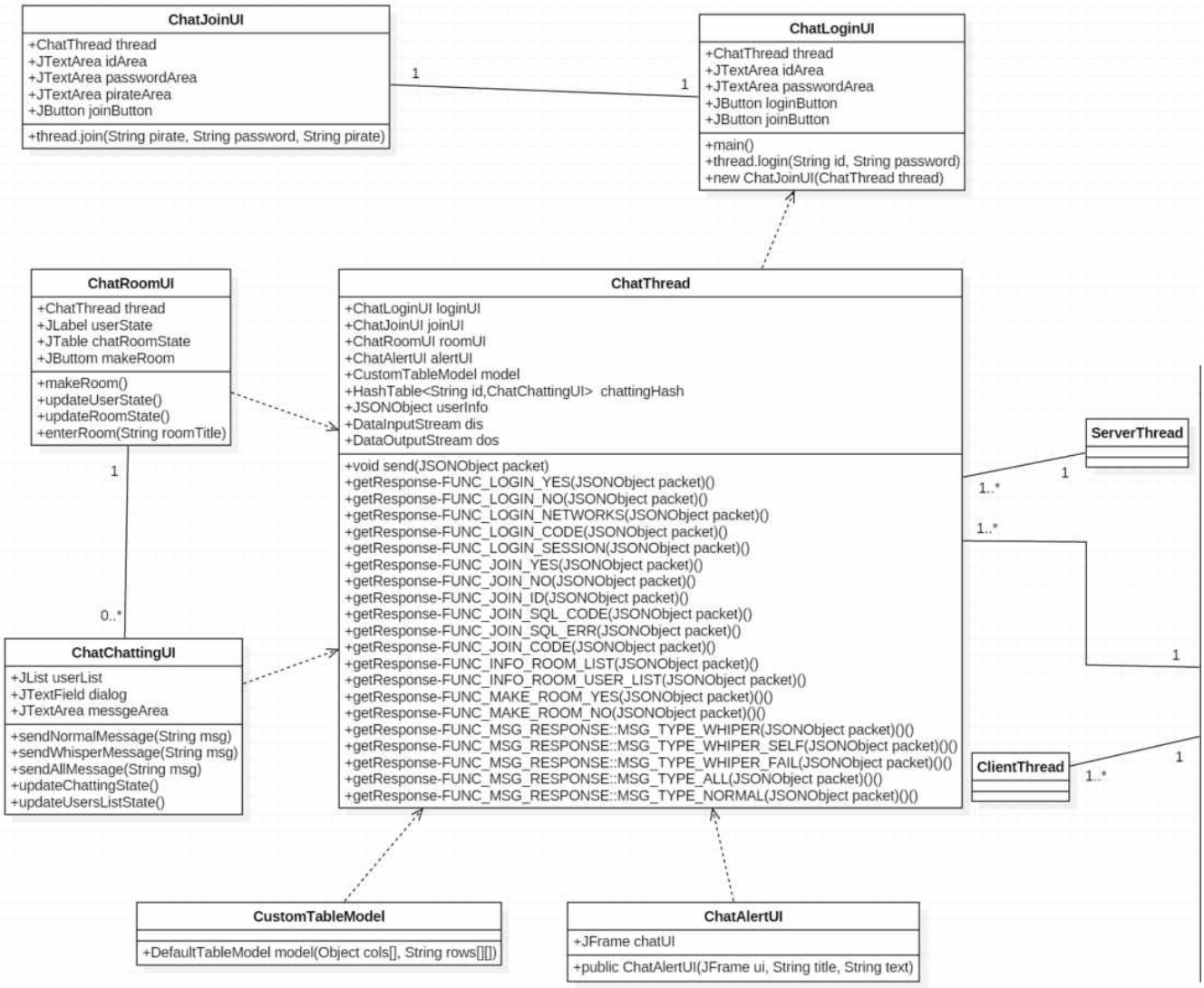
4) Use Case Diagram



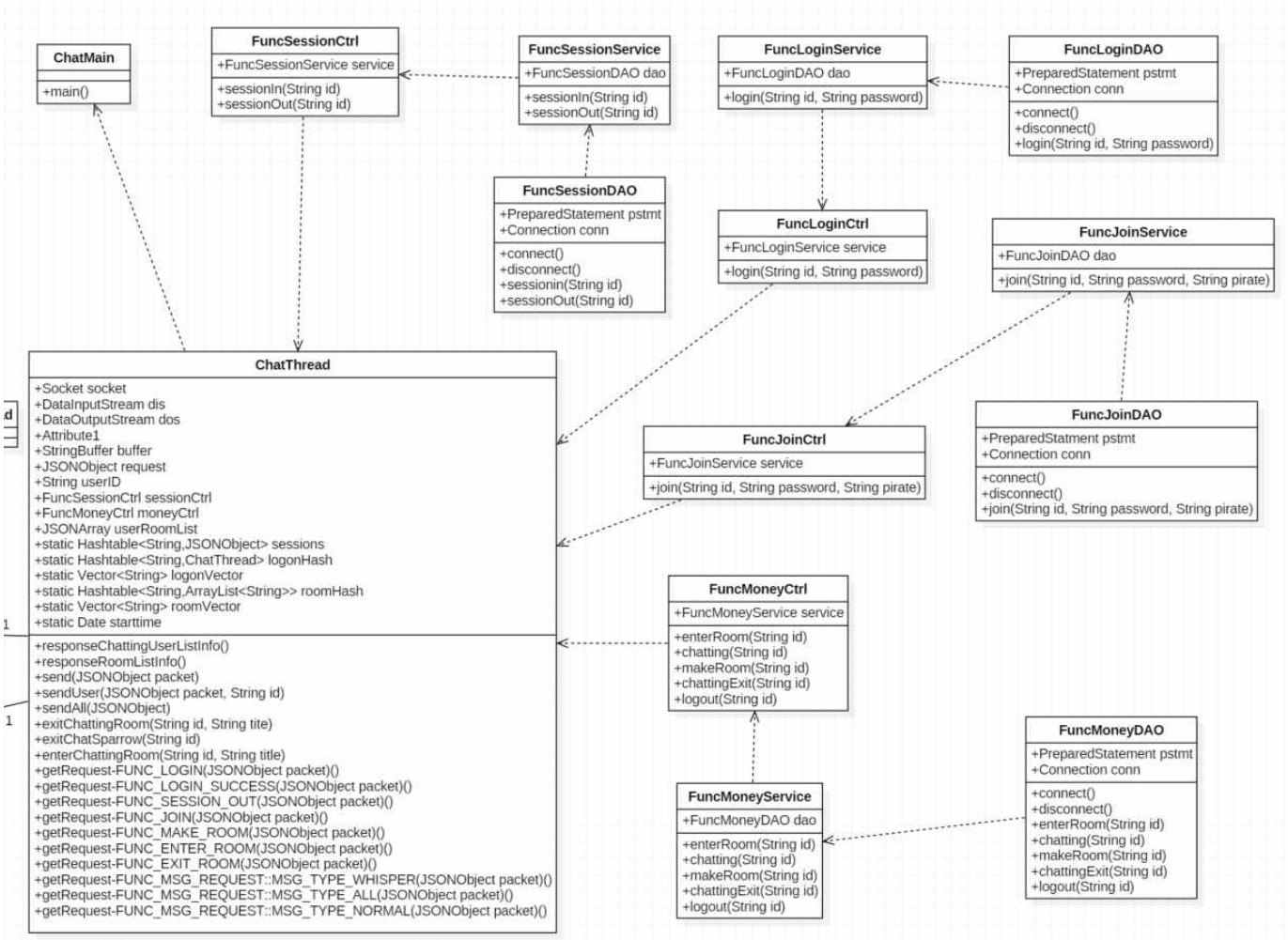
4. 시스템 설계

4.1 Class Diagram

1)Client 응용프로그램에 대한 클래스 다이어그램



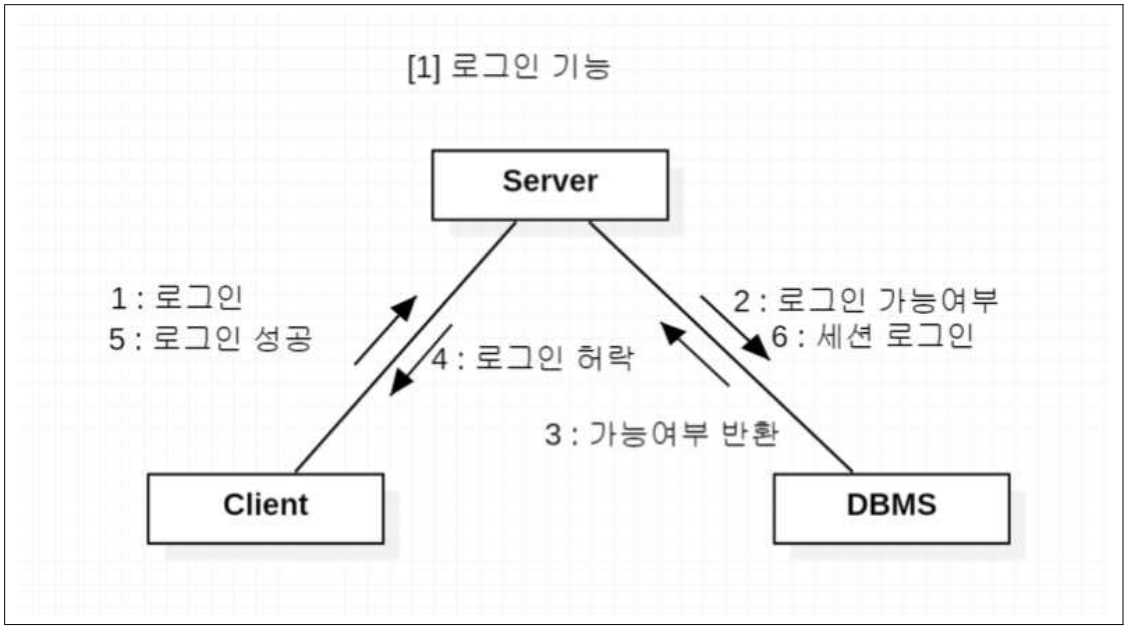
2) Server 서비스 운용 프로그램에 대한 클래스 다이어그램



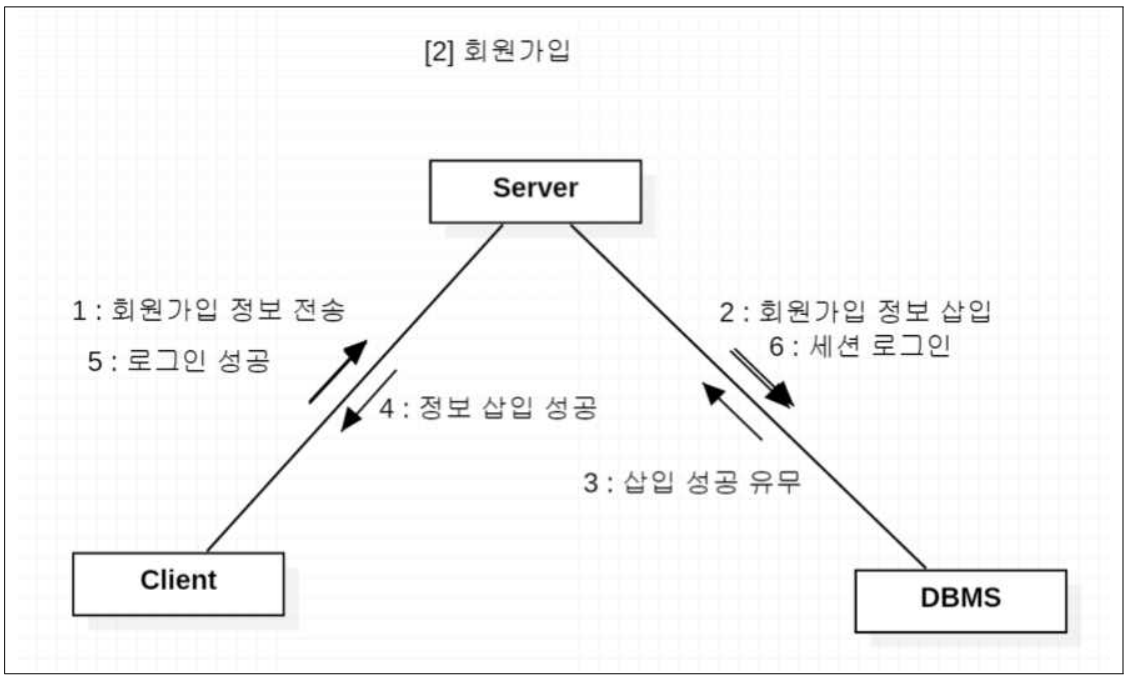
3) Client to Server 전체 클래스 다이어그램

4.2 Communication Diagram

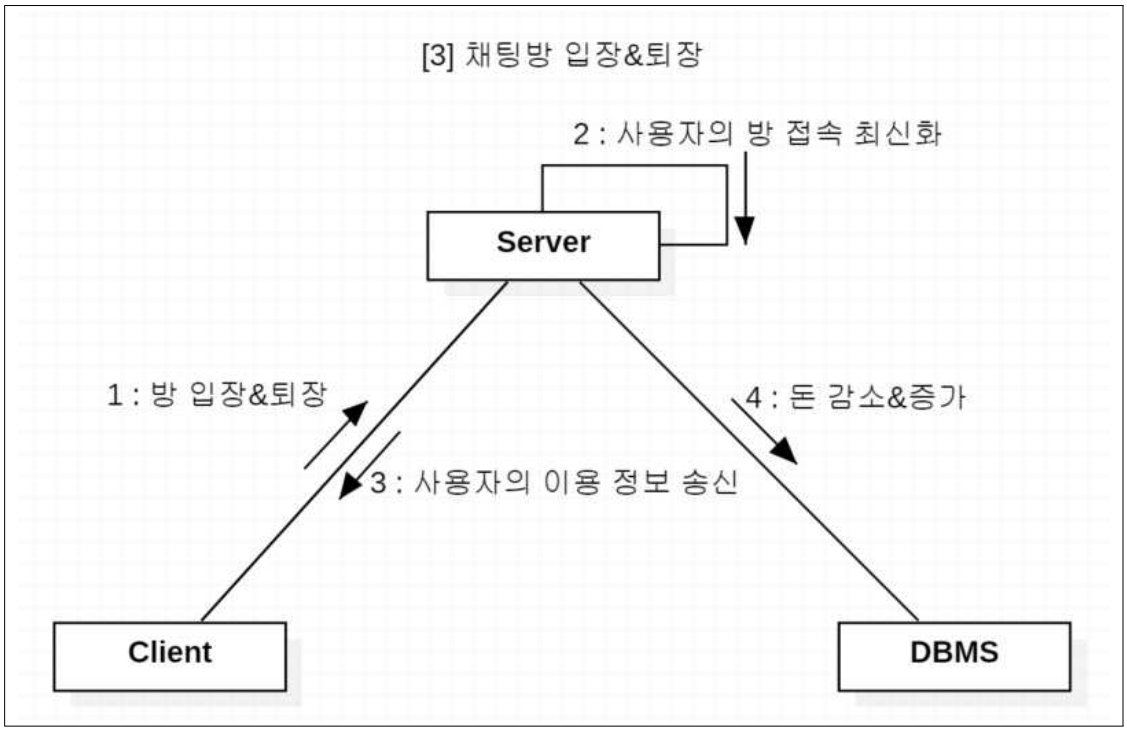
1) 로그인 기능



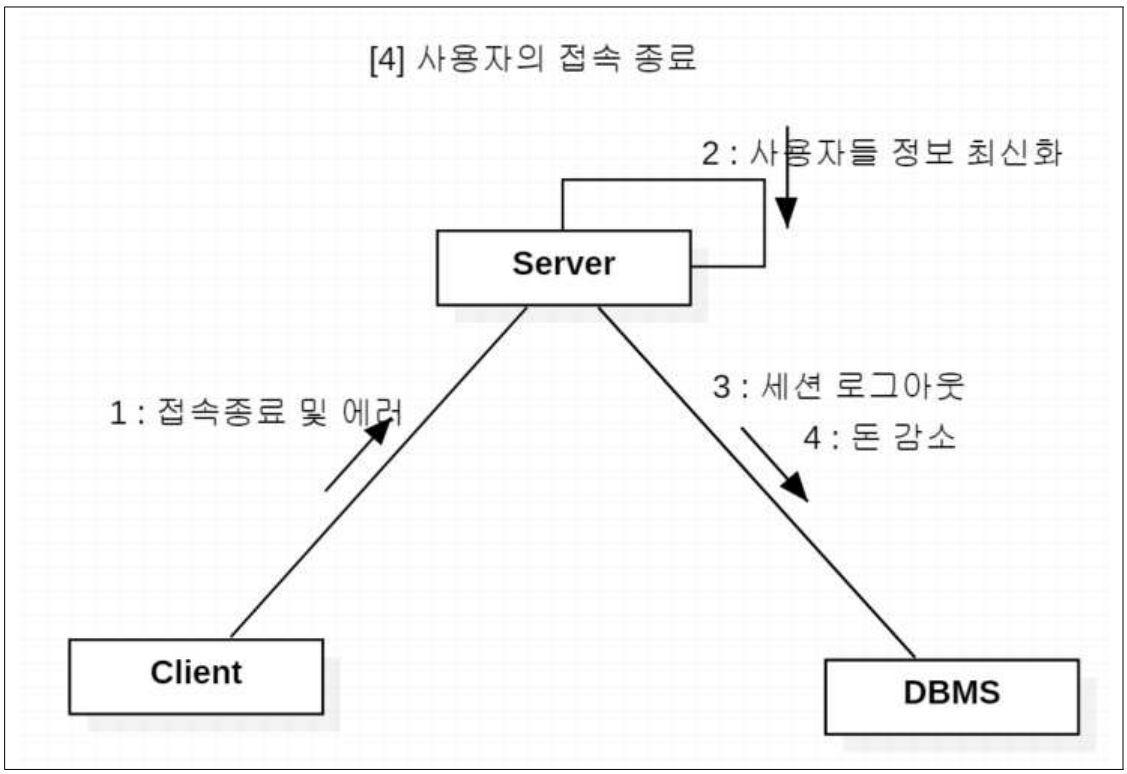
2) 회원가입



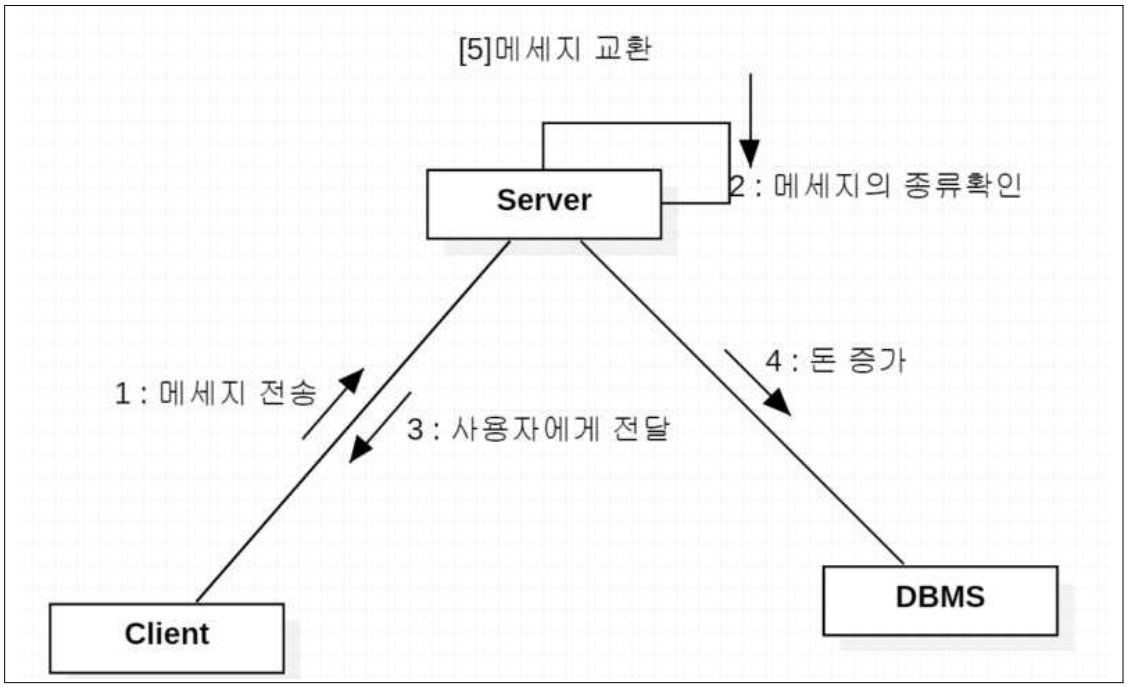
3) 채팅방 입장 및 퇴장



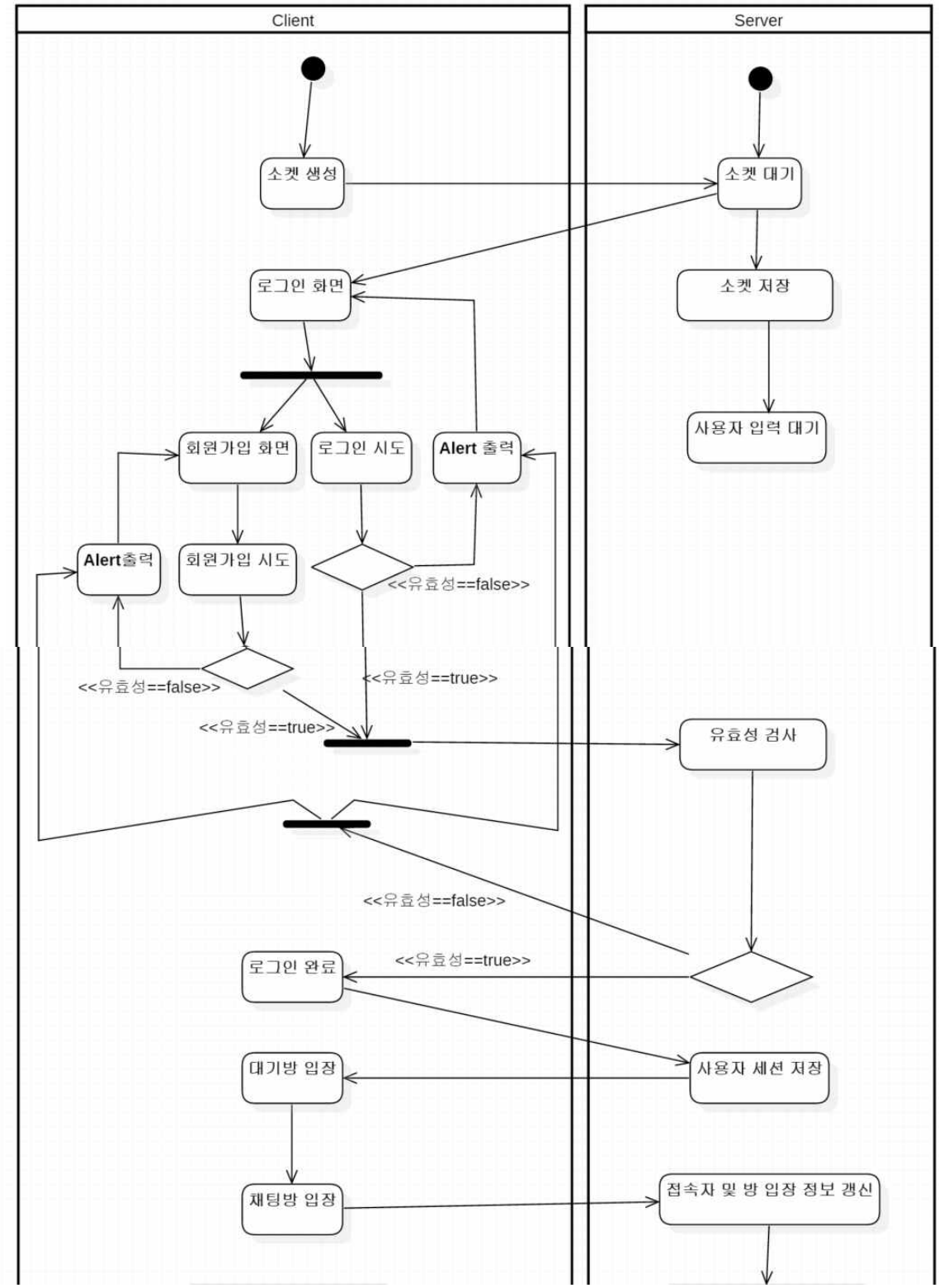
4) 사용자 접속 종료

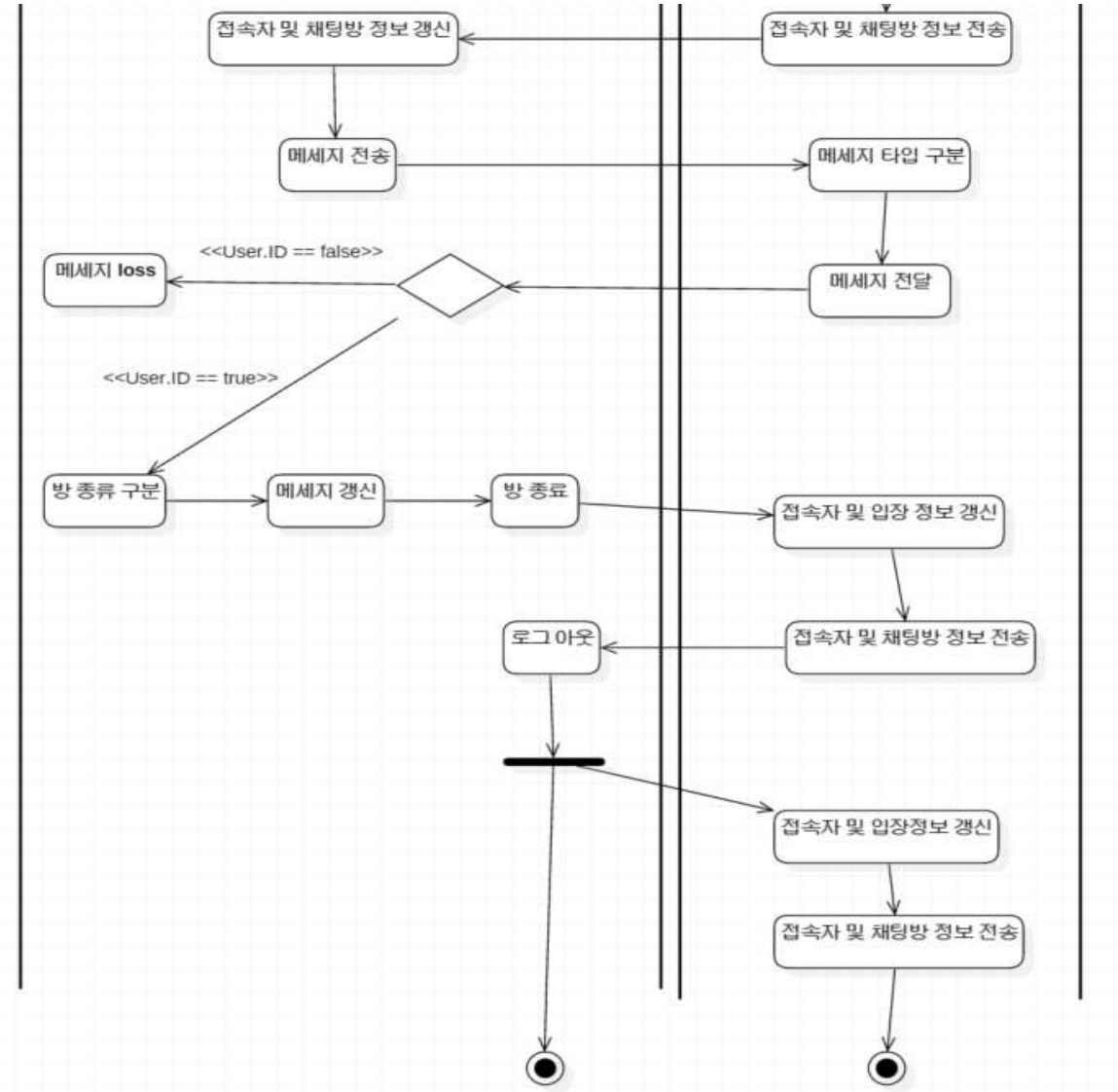


5) 메시지 교환



4.3 Active Diagram





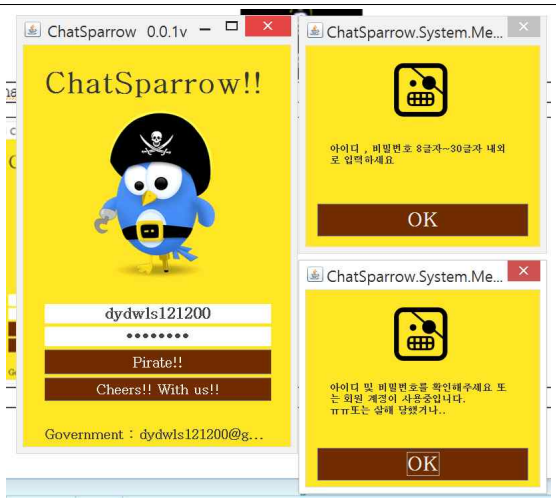
4.4 ERD

Pirate		
Attribute Name	Type	Description
group	Char(200) PK	그룹 이름
money	int(11)	돈 , 포인트
member	int(11)	그룹 멤버 수
date	DateTime	그룹 생성된 날짜

User		
Attribute Name	Type	Description
id	Char(100) PK	회원 아이디
password	char(100)	회원 비밀번호
group	char(200)	가입한 그룹 not FK
money	int(11)	돈 , 포인트
login	tinyint(4)	세션의 상태
data	Date	가입한 날짜

5. 실행 화면

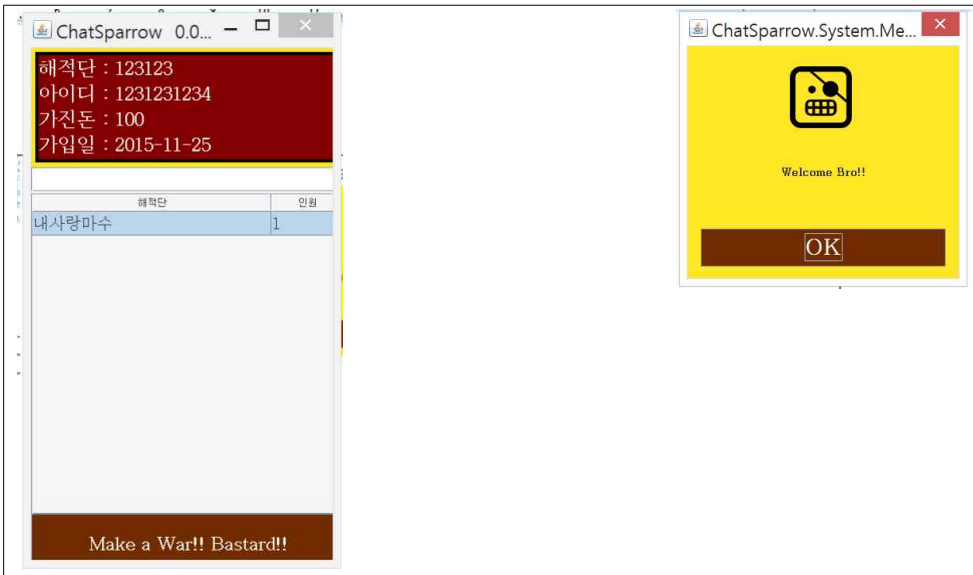
ChatSparrow의 아이콘입니다.



로그인화면이며 각각의 상황에 대하여 Alert Dialog를 출력합니다.

클라이언트 단에서는 문자열의 길이를 체크합니다. 그리고 인풋박스의 문자열들이 6~8글자 이상이 되어야지 버튼이 활성화가 되며 그러지 않게될경우에는 비활성화가 됩니다.

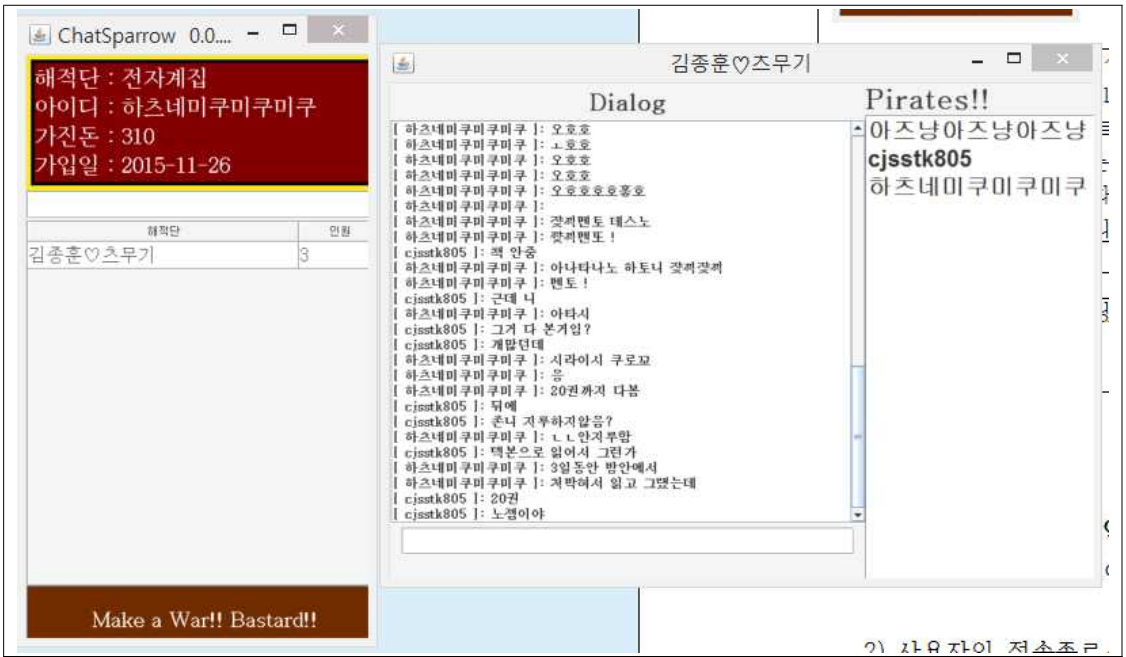
마찬가지로 서버에셔 역시 유효성을 검사하며 그에 대한 결과값을 클라이언트에게 반환해줍니다. 그 결과값을 이용하여 클라이언트는 AlertDialog를 출력합니다.



회원가입이나 로그인에 성공하면 다음과 같은 창으로 넘어가게 되며, 회원가입이 성공할 경우 이런 Alert Dialog가 출력됩니다.

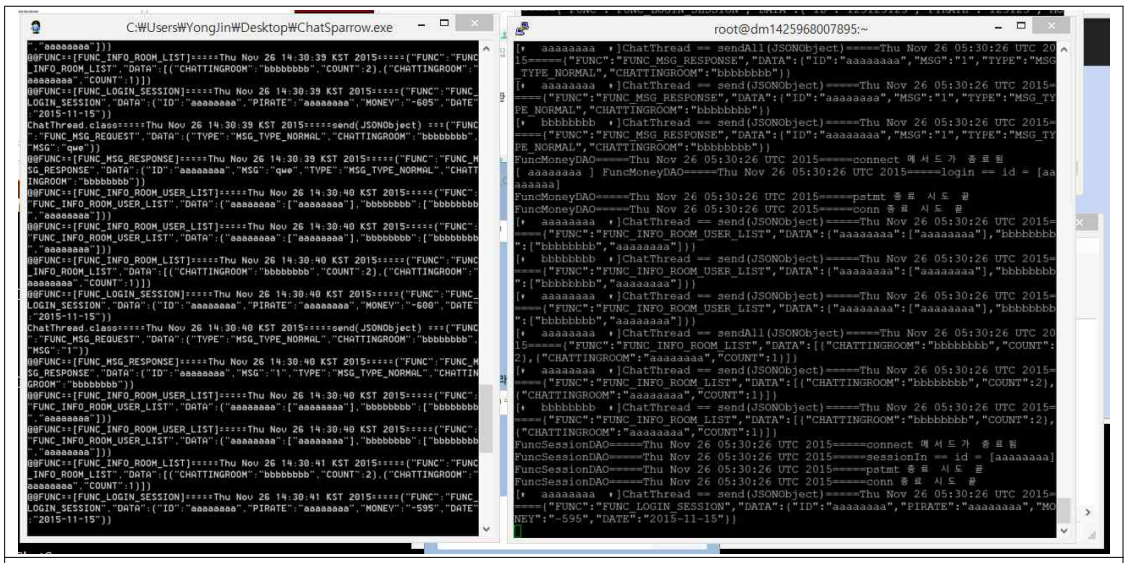
또한 사용자의 방 리스트에는 해적단, 아이디, 가진돈, 가입일과 같은 회원정보가 표시되며, 가진돈 항목에서는 회원의 서비스 이용하면서 돈을 벌게 되는데, 이러한 돈의 변동에 대한 표시가 동기화 되어 표시됩니다.

마지막으로 회원의 방 리스트역시 마찬가지로, 서버의 정보랑 동기화가 됩니다.



회원의 서비스 이용인 채팅방 입니다. 방을 여러개 들어갈 수 있습니다.

또한 채팅방의 회원 리스트들은 계속해서 최신화 됩니다.



좌측의 콘솔은 클라이언트의 로그입니다.
우측의 콘솔은 서버의 로그입니다.

6. 결론 및 개선방향

6.1 문제점 및 개선방안

1) 사용자의 접속종료에 관련된 모든 예외에 대처하지 못함.

2) 사용자의 접속종료시에 특정 인원일 때 중복 접속이 가능함.

3) MySQL DB에다가 접속 상태를 나타내는 Column에 대한 컨트롤이 부족함.

4) Command Log 내용의 획일화가 되지 않았음.

5) 사용자의 그룹에 대한 서비스 이용 기능을 추가하지 않음.

6.2 프로젝트 완료 후

나에게 이번의 네트워크 수업은 저번 2학년 2학기 때 부터 3학년 2학기가 되면 무조건 네트워크 관련 수업을 듣겠다고 기다리던 수업이었다.

왜냐하면 나는 시스템 아키텍쳐가 내 장래희망이었고 시스템 아키텍쳐는 네트워크에 대한 지식이 기본 베이스가 되어있어야 하기 때문이다. 그래서 그런지 소켓 프로그래밍을 할 때 가장 즐겁게 하지 않았나 싶다.

처음에 수업 시작할 때 멀티스레딩에 대한 개념만 알고있었지, 소켓의 입출력 작동 Flow 나 메시지에 대한 정의를 어떻게 내려야할지 감도 잡히지 않았다. 책에 나온 예제와 잘 숙달된 조교의 예제를 따라함으로써 덕분에 내 생각에 괜찮은 프로그램이 만들어 진 것 같다.

덧붙여서 괜찮은 프로그램이 나오기 이전에 PPT발표 준비가 나에게 있어 가장 도움이 되었다. 그 이유는 바로 ppt 내용에서 인코딩과 디코딩문자열 타입에 대해 약간 조사를 하게 되었는데, 각각의 호환되는 문자열 인코딩을 찾느라 애를 먹었던 기억이 구현하는데 도움이 되었다.

그 이후에는 수업이 끝나고 난 이후에도 따로 소켓프로그래밍에 대한 예제를 직접 쳐보거나, 혼자 만들려고 노력하기도 했다. 그리고 또한 한빛미디어에서 출간한 Netty 자바 네트워크 프로그래밍에 대한 내용에도 관심이 있어서 사용하려고 시도도 해보기도 했다.

이번 수업의 기말 프로젝트를 진행하면서 나에게는 고된 시간보다 되려 즐거운 놀이시간이 아니었나 싶다. 다른것에 신경쓰지 않고 좋아하는 개발에 대해 신경쓸 수 있었다. 나에게 있어 즐거운 시간이었다.

7. 참고 문헌

[1] 동기화와 비 동기화 소켓 통신
http://www.devpia.com/Maeul/Contents/Detail.aspx?BoardID=51&MAEULNO=20&no=5622&page=88
[2] 실행용 jar 파일 만들기 & 라이브러리 jar 파일 만들기
https://trello.com/c/FUigU4nP/99-jar-jar
[3] mysql 의 사용자 생성 및 권한 관리
http://link2me.tistory.com/431
[4] 자바 네트워크 프로그래밍 안용화 저
[5] 컴퓨터 네트워크 Top down Approach