네트워크게임프로그래밍

팀 프로젝트

2018182027 이예림

2016182041 조영환

2016182025 이규민

내용

[구현할 게임 3](#_Toc55250629)

[게임 기본 규칙 3](#_Toc55250630)

[개발 환경 4](#_Toc55250631)

[역할 분담 4](#_Toc55250632)

[서버 구현 5](#_Toc55250633)

[High-Level Design – MainThread – 이규민 담당 7](#_Toc55250634)

[High-Level Design – GameLogic – 이예림 담당 8](#_Toc55250635)

[8](#_Toc55250636)

[High-Level Design – UI – 조영환 담당 10](#_Toc55250637)

[Low-Level- Design – MainThread – 이규민 담당 11](#_Toc55250638)

[Low-Level- Design –GameLogic– 이예림 담당 16](#_Toc55250639)

[Low-Level- Design –UI – 조영환 담당 20](#_Toc55250640)

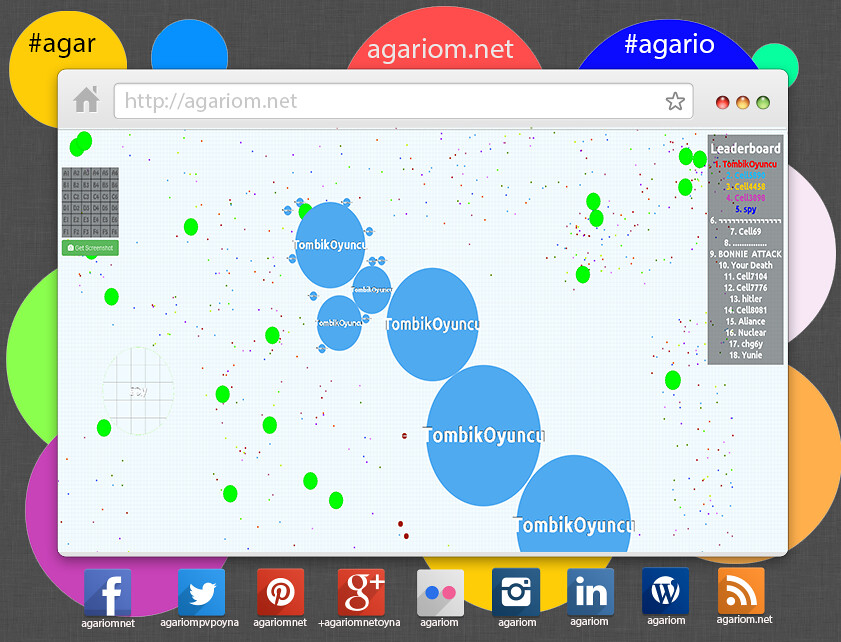
[개발일정 22](#_Toc55250641)

# 구현할 게임

* Agario

# 게임 기본 규칙

1. 반지름이 큰 Player가 작은 반지름을 가진 Player 을 흡수 할 수 있음
2. 맵내에 자동으로 부려지는 feed 를 먹고 점수를 획득해 반지름을 키울수 있다.
3. 획득한 점수에 따라 반지름이 달라지며 이동속도가 느려진다..
4. 게임시간 5분이 경과했을 경우 게임이 종료되며 가장 점수를 많이 획득한 player 가 승리한다.



# 개발 환경

|  |  |
| --- | --- |
| 개 발 환 경 | |
| 플랫폼 | PC |
| 운영체제 | Window 10 |
| 개발언어 | C / C++ |
| 사용 프로토콜 | TCP / IP |

# 역할 분담

이예림 – 서버에서 처리하는 게임 로직과 클라이언트

조영환 – UI 와 채팅서버와 클라이언트 구현

이규민 – MainThread 내에 쓰레드 생성 및 통신

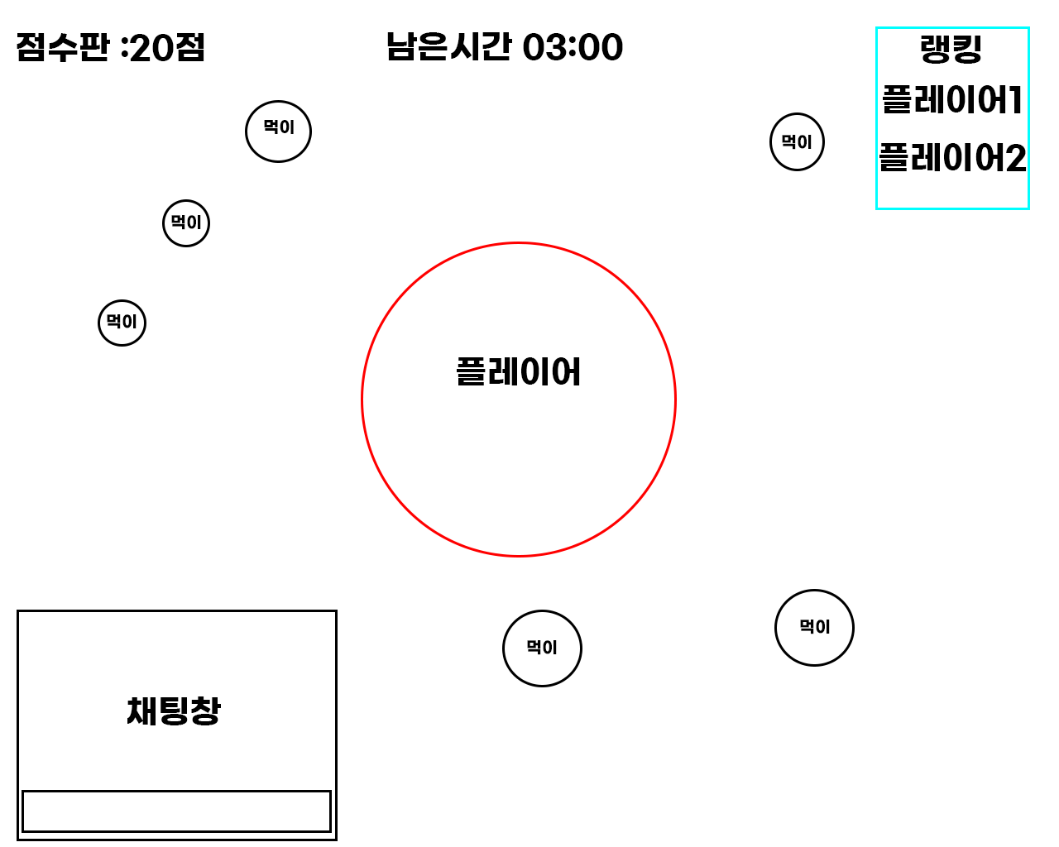
# 서버 구현

**Main Thread 구현 내용**

* TCP/IP 프로토콜을 사용
* 멀티 쓰레드를 사용하여 최대 3명의 유저 수만큼 생성 되는 통신 쓰레드 구현하고 게임 내 오브젝트 간의 충돌을 계산하는 쓰레드 구현
* 통신 쓰레드로 각 클라이언트에서 받은 값(키 입력 값)을 충돌 체크, 승패의 여부, 플레이어의 반지름의 길이 등 계산 된 값과 게임 시간을 각 플레이어에 이벤트 동기화를 한 후 전송한다.

**Pure Client-Server 방식으로 처리**

* 클라이언트에서 input 을 주면 서버에서 게임로직을 처리하는 구조
* 즉 서버가 게임 오브젝트들간의 상태를 동기화한다.
* LoL에서 사용하는 방식

**UI**  


점수판: 각 플레이어들의 개인 점수를 알려준다.

랭킹: 현재 플레이어의 개인점수들을 통하여 랭킹을 알려주게 된다.

남은 시간: 게임의 남은 시간을 알려줌으로 게임이 끝나는 시간을 보여주게 된다.

채팅창: 플레이어들 끼리 대화를 할 수 있게 만든 창

먹이: 플레이어가 점수를 얻기 위한 오브젝트 먹이 먹이는 플레이어보다 작아야한다.

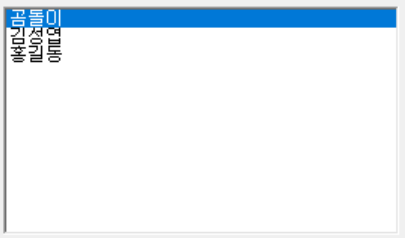
플레이어: 플레이어는 먹이를 먹고 점수를 얻으며 각 플레이어들 보다 높은 점수를 얻어야한다.

* UI 디자인

플레이어: winapi를 통한 winapi함수를 통하여 구현

먹이 : 위와 동일

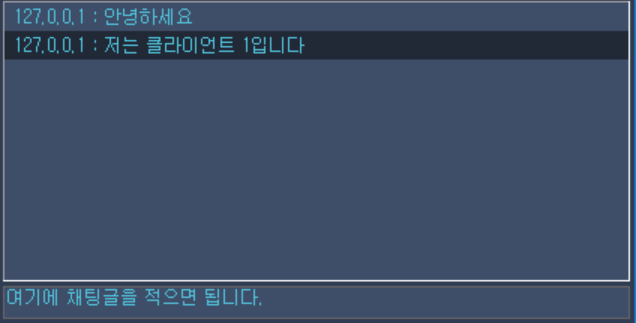
랭킹: winapi의 List box를 통하여 구현



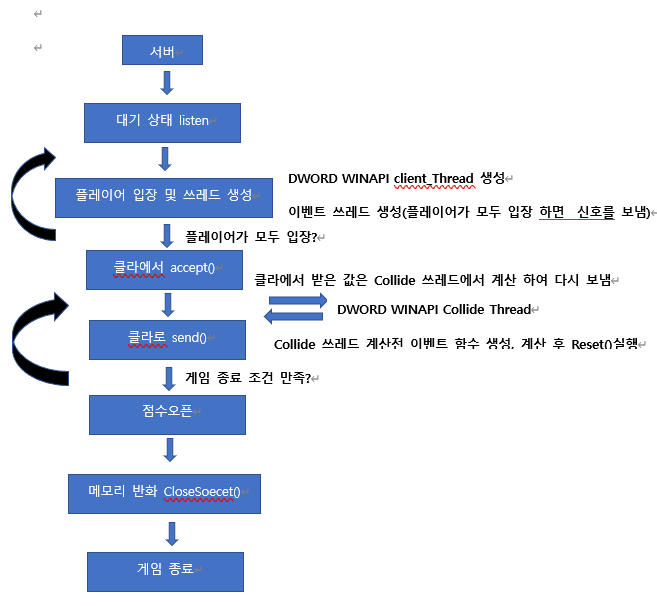
점수판: 문자열을 통한 점수를 보여줌

남은시간: 위와 동일

채팅창: winapi Listbox를 통하여 채팅을 입력 및 출력을 할 수 있도록 구현



# High-Level Design – MainThread – 이규민 담당

* MainThread 부분
* 

# High-Level Design – GameLogic – 이예림 담당

# 

클라이언트

서버

클라이언트 접속

클라이언트 접속 요청시

Thread 2개 생성

DWORD WINAPIMapUpdate()

1프레임 after

DWORD WINAPI InputUpdate()

update() 호출

Input

받아서 맵에 적용

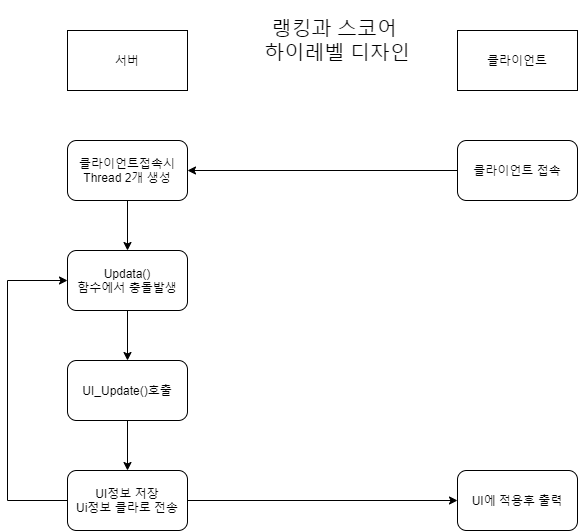
map 에 적용 후

맵 정보 클라로 전송

출력

|  |  |
| --- | --- |
| InputSystem() | |
| 서버 | 클라 |
| 클라이언트 수만큼 Thread 생성 | 플레이어 키보드 입력  서버로 전송 |
| 전송 받은 후 서버 맵에 적용  Update()함수호출 \*충돌체크 및 사이즈 조절 |  |
| 다른 클라이언트에 전송 |  |
|  | 받은 맵 데이터 맵에 적용 |
| Map update | |
| 서버 | 클라이언트 |
| 프레임 마다(맵데이터를 보냄) |  |
|  | 서버에서 받은 데이터 맵에 넣음 |

# High-Level Design – UI – 조영환 담당



# Low-Level- Design – MainThread – 이규민 담당

void initobject()

- 서버에서 오브젝트의 위치와 충돌박스 정보를 초기화

보낼 데이터

* 세포들의 위치
* 세포들의 반지름
* 세포들의 색

Void PlayerInfo()

- 플레이어의 정보

구현

- 플레이어의 초기 위치를 설정

- 플레이어의 아이디 값(들어온 순서대로 받은 값) 초기화

bool IsPlayerReady

- 서버와 클라의 연결 상태를 확인한다.

- 처음에는 false로

- 클라가 서버에 connect 되면 true로 리턴

DWORD WINAPI Client\_Thread(LPVOID arg)

- 클라이언트와의 통신을 하는 쓰레드

클라이언트에서 보낼 것들

- 들어 온 순서 대로 번호를 전송

- 플레이어의 초기의 위치를 전송

- 게임 내 시간을 전송

- 계산 된 값을 전송

- 승패 유무

클라이언트에서 받은 것들

* 플레이어의 좌표

이벤트 동기화

* 플레이어가 모두 입장 할 때 까지 대기
* 충돌 쓰레드가 끝날 때 까지 대기

DWORD WINAPI Collide\_Thread(LPVOID arg)

* 각 오브젝트들의 충돌을 계산 하는 쓰레드
* 플레이어와 세포(먹이)의 충돌
* 플레이어들 간의 충돌

이벤트 동기화

* 플레이어가 모두 입장 할 때 까지 대기
* 충돌 계산이 진행 후 이벤트 신호 상태로 변경

받을 데이터

* 플레이어의 좌표 부딪히는 다른 좌표

보낼 데이터

* 충돌된 값 : 플레이어들 간의 충돌로 커지는 플레이어와
* 초기화 되는 플레이어들 정보
* 플레이어와 부딪친 세포(삭제), 플레이어의 커진 반지름

DWORD WINAPI Time\_Thread(LPVOID arg) :

* 게임 내 시간을 나타내는 쓰레드
* 모든 플레이어가 게임에 입장하고 게임이 진행되면 시간을 측정

이벤트 동기화

* 플레이어가 모두 입장을 할 때 까지 대기

Int main()

* 모든 오브젝트 초기화
* 서버에 관련된 변수와 함수 선언
* 이벤트 선언 변수 선언
* 모든 플레이어가 입장 후 이벤트 쓰레드 생성
* 충돌 쓰레드 생성
* 시간 쓰레드 생성
* Reset()메인 쓰레드 // 모든 플레이어가 들어오면
* 클라이언트 쓰레드 생성
* 메모리들 반환

**사용되는 자료형과 구조체**

HANDLE hcollideEvent[3] // 플레이어 대기

HANDLE hmainEcent; // 메인 이벤트

Bool Win = false;

* 이외에 위치를 받는 정보 같은 경우에는 GameLogic 에 있는 구조체들을 활용해   
  통신할 예정

# Low-Level- Design –GameLogic– 이예림 담당

**서버에 쓰레드에 들어가는 함수**

DWORD WINAPI MapUpdate()

* 주로 send() 하는 역할
* Recv 하는 값이 없더라도 1프레임마다 map을 update하고 정보를 send해주는 역할

클라이언트에서 보낼 것들

* 내 object 의 위치

클라이언트에서 받을 것들

* 모든 object 의 위치

서버에서 처리 할 것들

* 플레이어와 objects 들의 충돌처리
* 점수 획득
* 사이즈 변화

서버에서 클라로 보내줘야 할 것들

* 모든 플레이어들의 위치
* 점수 획득
* 사이즈 변화

void InputSystem()

* 접속시 생성된 mapupdate 함수 내부에 존재
* 서버의 소캣과 연결된 클라이언트에서 보내는 값이 있을시 실행
* 서버 내에서는 recv하는 역할

클라에서 보낼 것

* Input된 키값

서버

* Input된 키값을 받아 충돌체크 구현후 mapupdate 이후 map 정보 재전송

**클라이언트에 사용될 함수**

* Send 함수로 player가 누르는 키값을 enum Direction 형으로 send 예정

**사용할 구조체**

struct Position

* x 좌표 – int 형
* y 좌표 – int 형

class CircleObject

* size – 반지름 – int 형
* position – Position
* void SetPosition()
* Position GetPosition()
* void SetSize(int s)
* int GetSize()
* void CrashCheck(Circle Object& object) –
* void Draw()

class Map

* feed[500] – CircleObject – 랜덤으로 불특정 위치에 생성됨
* player[4] – Player
* void Update() – 에서 CrashCheck() 실행, feed 위치와 사이즈 렌덤으로 배치
* void Draw()
  + 모든 오브젝트 그리기
* void CrashCheck()
  + 플레이어와 모든 object 충돌체크

enum Direction

* UP, DOWN, RIGHT, LEFT 로 이루어짐

class Player

* CircleObject의 상속을 받음
* name – string
* score – int 점수기반으로 Circle object의 size 변경
* Direction Input() – 인풋된 키보드 값을 Enum 형식으로 변환 후 리턴.

# Low-Level- Design –UI – 조영환 담당

기능 분석

1. 쓰레드 이용

Sever: 클라이언트 연결 요청이 있을 때 마다 클라이언트를 다루기 위한 쓰레드 생성

Client: 서버에 접속 시 송신 쓰레드와 수신 쓰레드 생성 (전역변수로 서로 정보를 주고 받을 수 있음)

1. 쓰레드 동기화 방식

동기-이벤트 방식

1. 사용할 소켓

TCP소켓을 이용한 다중 이용 채팅

1. 채팅기능

Server: 클라이언트로부터 받은 메시지를 임시 배열에 저장, 배열에 저장된 메시지를 모든 클라이언트에게 보낸다.

Client: 전달할 메시지를 입력하면 임시 배열에 저장, 배열에 저장된 메시지를 서버에 보낸다. (송신 쓰레드), 서버로부터 오는 메시지를 출력(수신 쓰레드)

필요한 변수,함수

1. Client

struct UI\_List{

int Ranking;

int Score;

char MsgBuf[];

};

1. Sever

int MsgBuf[]

int Ranking;

int Score;

char MsgBuf[];

int MainTimer; //서버의 메인 시간

int FeedCount; //먹이의 숫자

void UI\_Update(int ranking,int score);//update()에서 충돌체크가 될 때 ranking, score, 계산

void UI\_MainTimer(int MainTimer,int LimitTimer,int FeedTimer);// LimitTimer와 feedTimer를 계산

1. 쓰레드 함수

DWORD WINAPI UI\_info(int ranking,int score) //변화된 ranking과 score를 클라이언트에게 보낼 함수

DWORD WINAPI UI\_SeverTimer(int LimitTimer, int FeedTimer) //LimitTimer 와 FeedTimer를 클라이언트에게 보낼 함수

DWORD WINAPI MsgLog(char MsgBuf[])//메시지를 송신할 함수

# 개발일정

|  |  |
| --- | --- |
| 날짜 | 내용 |
| 11월 4일 | **이예림**  클라이언트 winapi 로 제작   * class 만들기 (서버없이 작동하는 게임)   **이규민**  클라와 서버를 통신하는 쓰레드구현   * DWORD WINAPI Collide\_Thread(LPVOID arg)   **조영환**   * 클라이언트 채팅프로그래밍 개발 |
| 11월 5일 | **이예림**  클라이언트   * draw 함수들 정의(각 오브젝트별)   **이규민**  클라와 서버를 통신하는 쓰레드구현   * DWORD WINAPI Collide\_Thread(LPVOID arg)   **조영환**  클라이언트 채팅프로그래밍 개발 |
| 11월 6일 | **이규민**  클라와 서버를 통신하는 쓰레드구현   * DWORD WINAPI Collide\_Thread(LPVOID arg)   **조영환**  클라이언트 채팅프로그래밍 개발 |
| 11월 7일 | **이예림**  클라이언트   * 델타 타임을 이용한 frame 별로 업데이트 함수 구현   **이규민** int score = player[i].GetScore() + player[j].GetScore();  player[i].SetScore(score);  player[j].SetRandomPosition();  player[j].SetSize(50);  player[j].SetScore(0);  클라와 서버를 통신하는 쓰레드구현   * DWORD WINAPI Collide\_Thread(LPVOID arg)   **조영환**   * 클라이언트 채팅프로그래밍 개발 |
| 11월 8일 | **이예림**  클라이언트   * Feed 시간마다 생성되도록 제작   **이규민**   * 오브젝트 간의 충돌을 관련 쓰레드 구현  DWORD WINAPI Collide\_Thread(LPVOID arg)   **조영환**   * 클라이언트 채팅프로그래밍 개발 |
| 11월 9일 | **이예림**  클라이언트   * 충돌체크 함수 구현   **이규민**   * 오브젝트 간의 충돌을 관련 쓰레드 구현  DWORD WINAPI Collide\_Thread(LPVOID arg) |
| 11월 10일 | **이예림**  클라이언트   * Player 키보드로 이동과 feed 와 충돌체크 부분 map update 에 구현   **이규민**   * 오브젝트 간의 충돌을 관련 쓰레드 구현  DWORD WINAPI Collide\_Thread(LPVOID arg)   **조영환**   * 서버 채팅프로그래밍 개발 |
| 11월 11일 | **이규민**   * 오브젝트 간의 충돌을 관련 쓰레드 구현  DWORD WINAPI Collide\_Thread(LPVOID arg)   **조영환**   * 서버 채팅프로그래밍 개발 |
| 11월 12일 | **조영환**   * 서버 채팅프로그래밍 개발 |
| 11월 13일 | **조영환**   * 서버 채팅프로그래밍 개발 |
| 11월 14일 | **이예림**  서버   * 클라이언트에 있는 map update 부분 모든 게임 플레이요소 가져오고  클라이언트 내에서는 draw 와 class 들을 제외한 나머지 요소들 삭제 |
| 11월 15일 | **이예림**  서버   * 클라이언트와 서버 쓰레드 생성으로 여러 클라이언트들과 연결하고 서버 데이터 전송 부분 고정길이 전송으로 구현   **조영환**   * 서버 채팅이벤트 설정 |
| 11월 16일 | **이예림**  클라이언트   * 클라이언트에 input 전송부분 구D604   서버   * 서버에서 클라이언트의 input 부분이 잘 적용되는지 테스트   **조영환**   * 서버 채팅이벤트 설정 |
| 11월 17일 | **조영환**   * 서버 채팅이벤트 설정 |
| 11월 18일 | **이규민**   * 클라에서 받은 값을 받아서 충돌 쓰레드에서 계산된 값 전송 하는 부분 구현   **조영환**   * 클라UI랭킹프로그램 개발 |
| 11월 19일 | **이예림**  서버   * 버그 및 코드 정리   **이규민**   * 클라에서 받은 값을 받아서 충돌 쓰레드에서 계산된 값 전송 하는 부분 구현   **조영환**   * 클라UI랭킹프로그램 개발 |
| 11월 20일 | **이규민**   * 클라에서 받은 값을 받아서 충돌 쓰레드에서 계산된 값 전송 하는 부분 구현   **조영환**   * 클라UI랭킹프로그램 개발 |
| 11월 21일 | **이규민**   * 클라에서 받은 값을 받아서 충돌 쓰레드에서 계산된 값 전송 하는 부분 구현   **조영환**   * 클라UI점수개발 |
| 11월 22일 | **이규민**   * 전송 된 값이 클라에 update 되게 구현   **조영환**   * 클라UI점수개발 |
| 11월 23일 | **이규민**   * 전송 된 값이 클라에 update 되게 구현   **조영환**   * 클라UI점수개발 |
| 11월 24일 | **이규민**   * 전송 된 값이 클라에 update 되게 구현   **조영환**   * 서버 UI쓰레드 개발 |
| 11월 25일 | **이규민**   * 버그 및 코드 수정   **조영환**   * 서버 UI쓰레드 개발 |
| 11월 26일 | **이규민**   * 버그 및 코드 수정   **조영환**   * 서버 UI시간 쓰레드 개발 |
| 11월 27일 | **이규민**   * 버그 및 코드 수정   **조영환**   * 서버 UI시간 쓰레드 개발 |
| 11월 28일 | **조영환**   * 서버 UI시간 쓰레드 개발 |
| 11월 29일 |  |
| 11월 30일 | **조영환**   * 클라이언트 UI시간관련 개발 |
| 12월 1일 | **조영환**   * 클라이언트 UI시간관련 개발 |
| 12월 2일 | **조영환**   * 클라이언트 UI시간관련 개발 |
| 12월 3일 |  |
| 12월 4일 | **이예림**   * 테스트 후 수정사항 작업   **조영환**   * 팀원간 프로그램 병합 |
| 12월 5일 | **조영환**   * 팀원간 프로그램 병합 |
| 12월 6일 | **조영환**   * 팀원간 프로그램 병합 |
| 12월 10일 | **발표** |