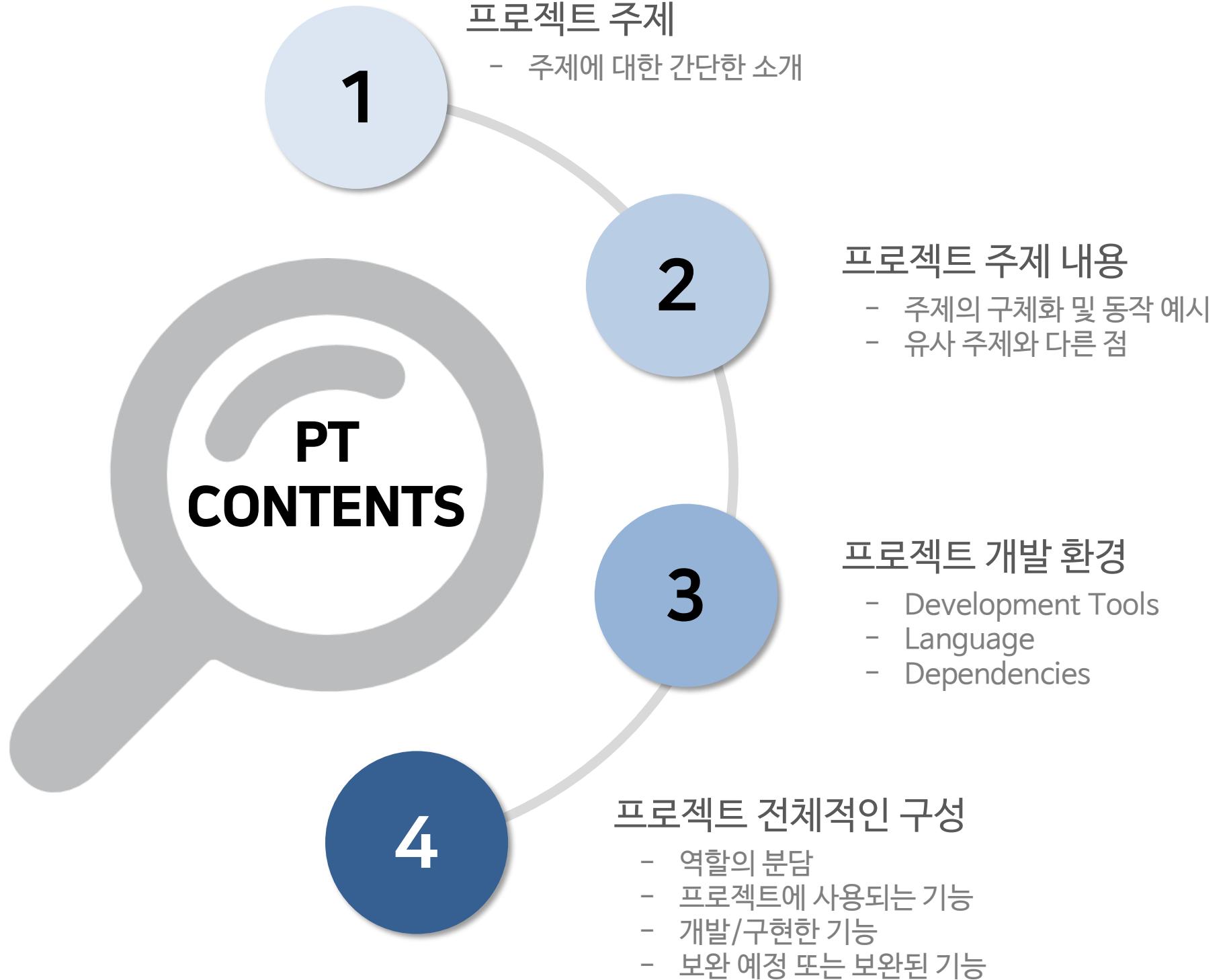


QGIS와 네트워크 카메라를 결합한 관제 및 순찰 시스템의 개발

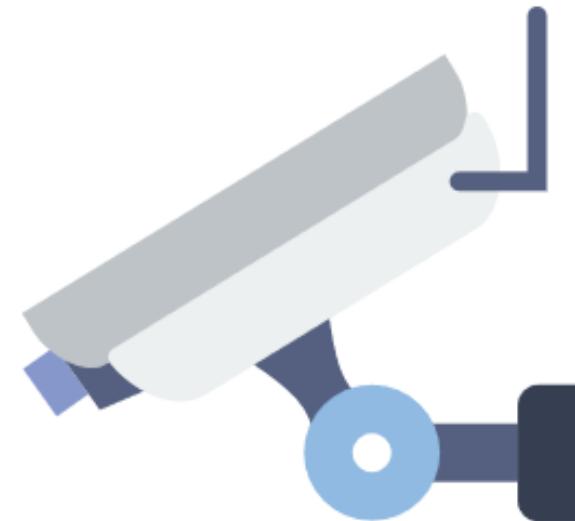
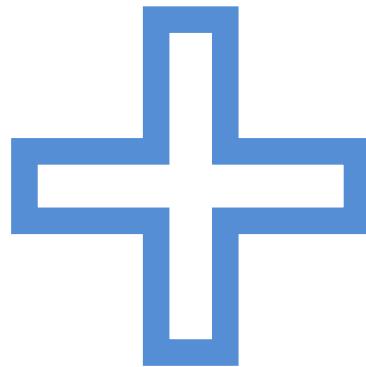
송용승

이한솔





Q G I S

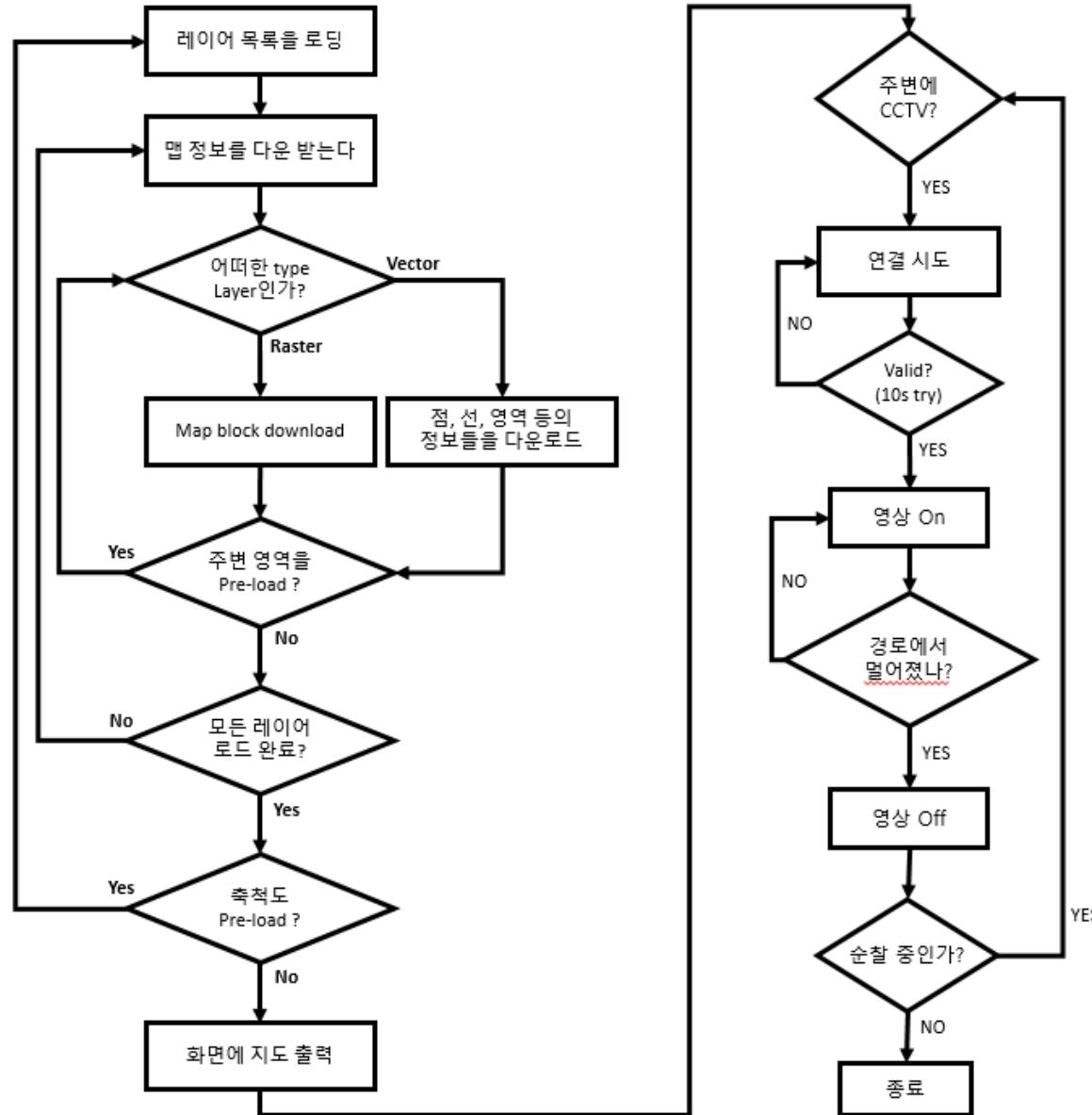


**Network
Camera**

2

프로젝트 주제 내용

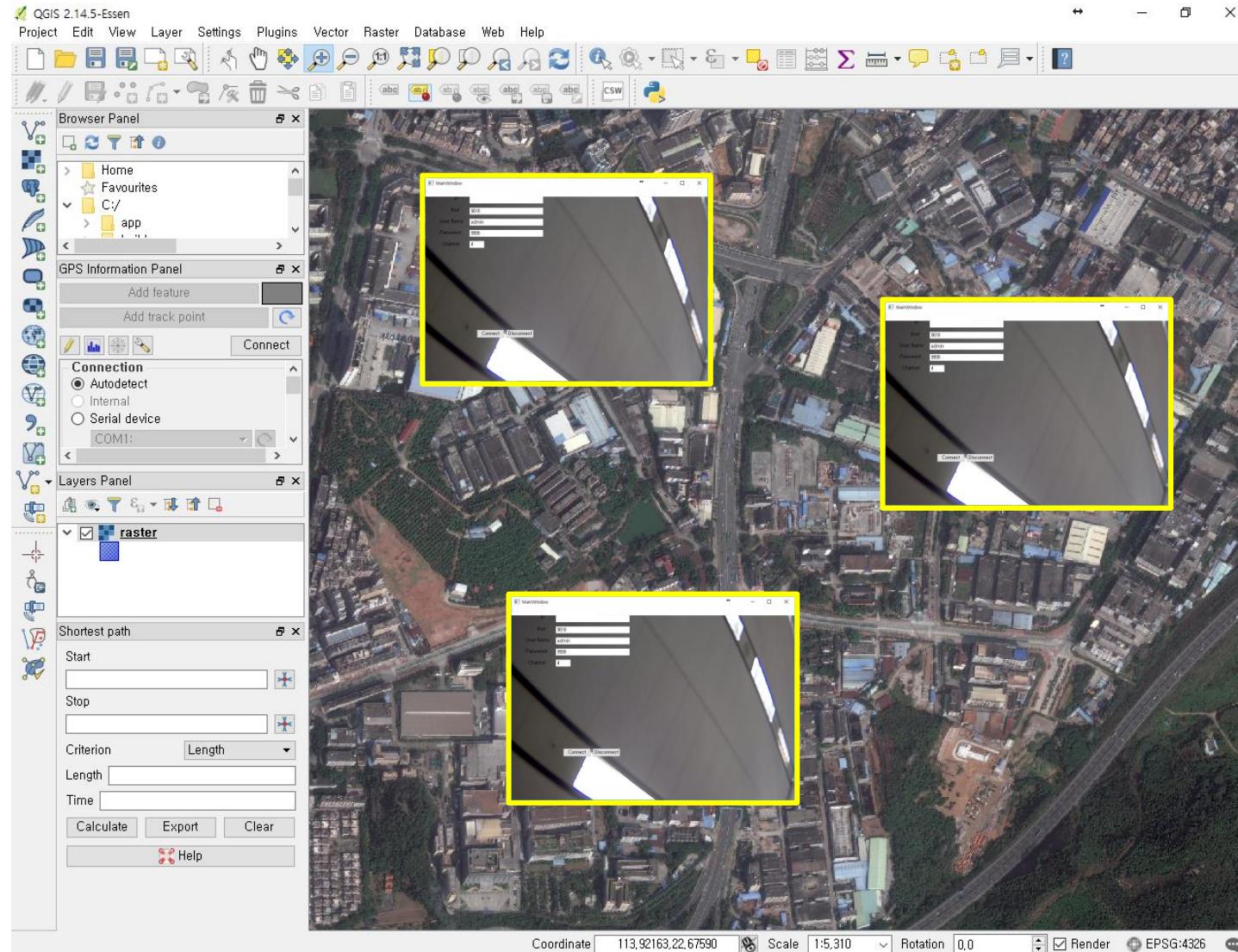
주제의 구체화 및 동작 예시



2

프로젝트 주제 내용

주제의 구체화 및 동작 예시

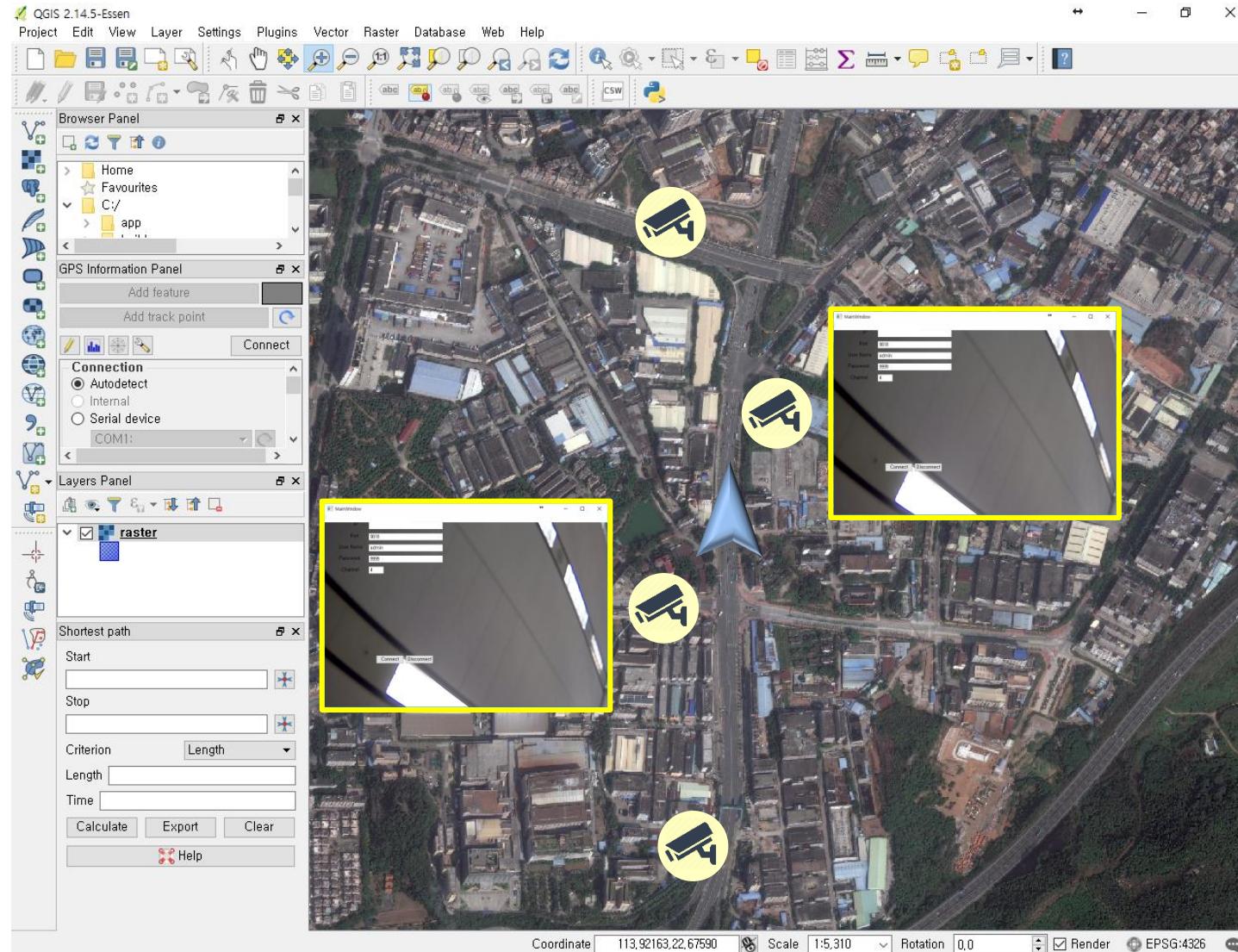


“ 영상 재생 ”

2

프로젝트 주제 내용

주제의 구체화 및 동작 예시

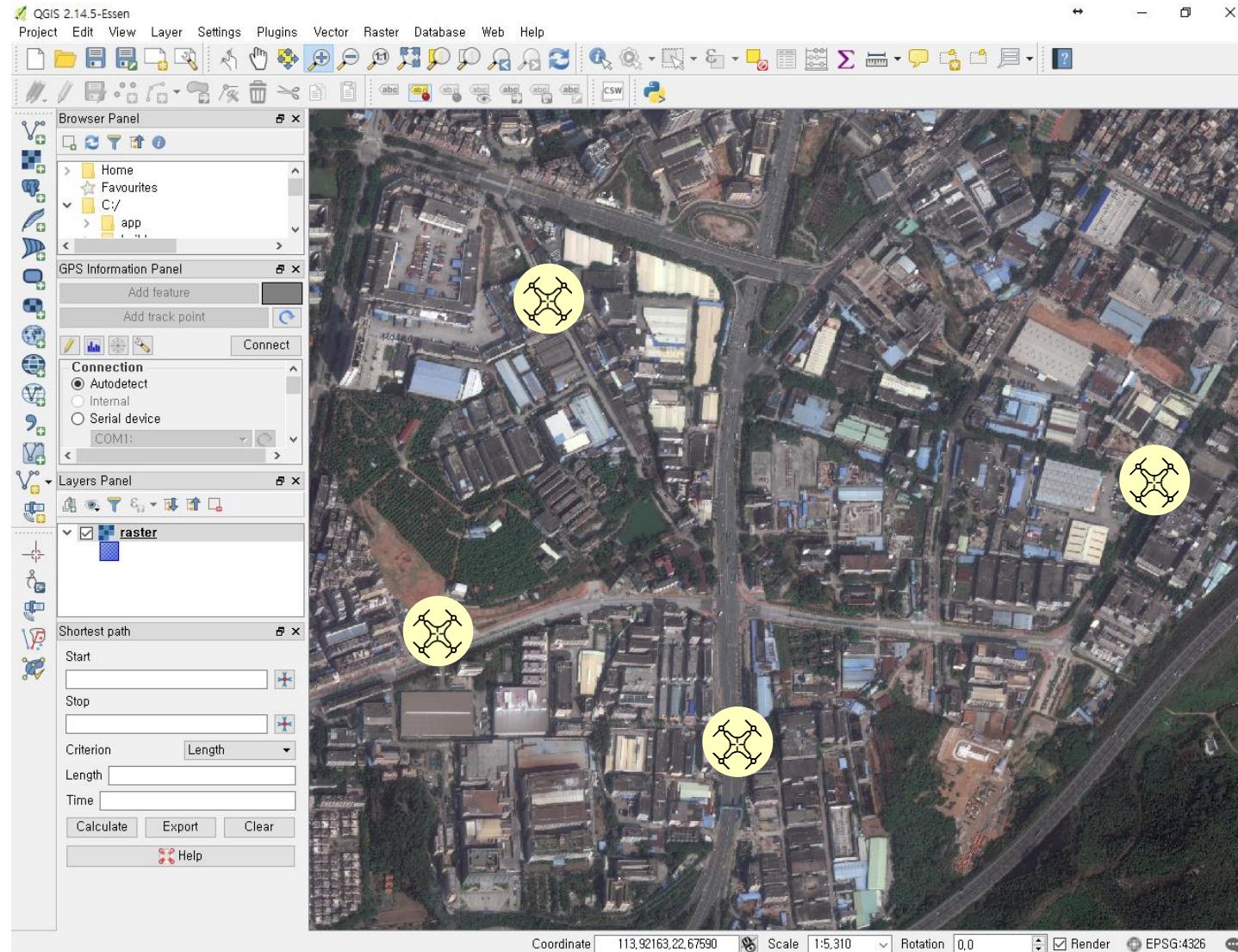


“ Patrol System ”

2

프로젝트 주제 내용

주제의 구체화 및 동작 예시

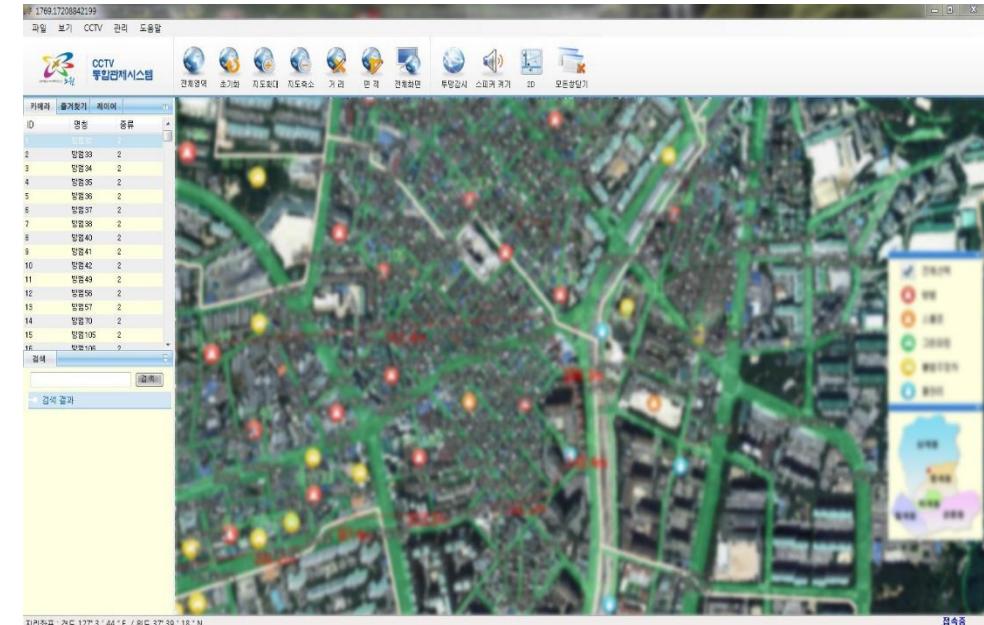
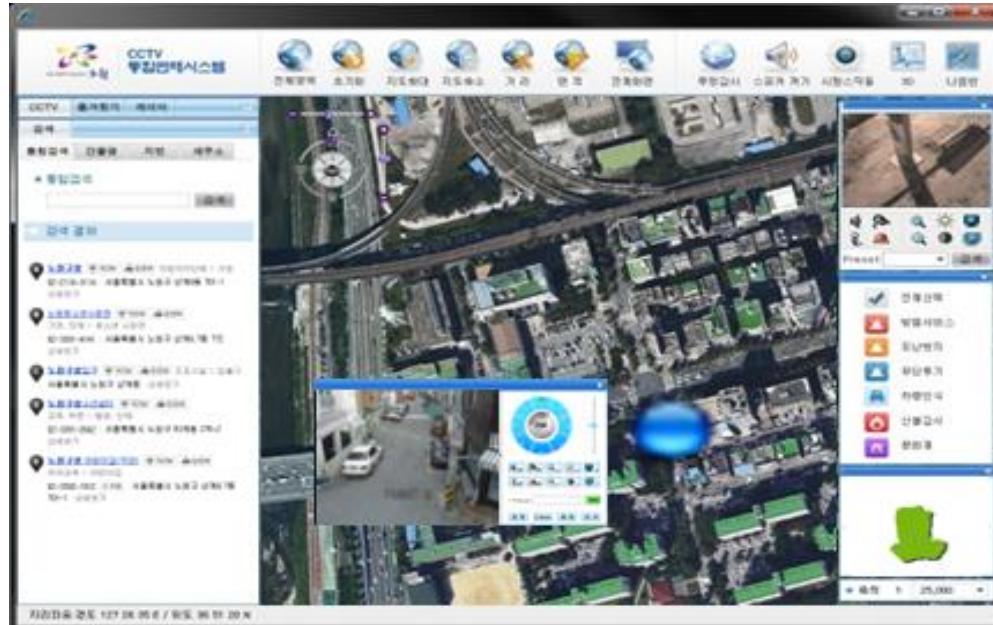


“ G P S ”

2

프로젝트 주제 내용

유사 주제와 다른 점

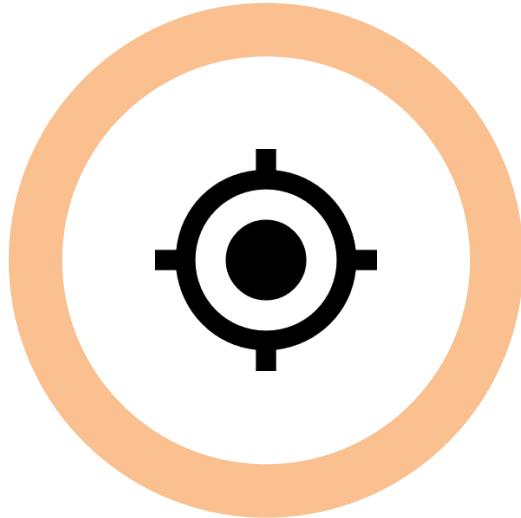


- 기능 : 위성 지도와 CCTV 정보를 합하여 통합 관제 시스템의 구축
- 구현 : 온라인으로 제공하는 지도 API를 이용한 것으로 보임
- 문제점 : 특정 API를 이용하였기에 특정 지도 밖에 출력하지 못함

2

프로젝트 주제 내용

유사 주제와 다른 점



GPS



Various
Formats



Patrol

3

개발 환경

Development Tools





C++

3

개발 환경

Dependencies



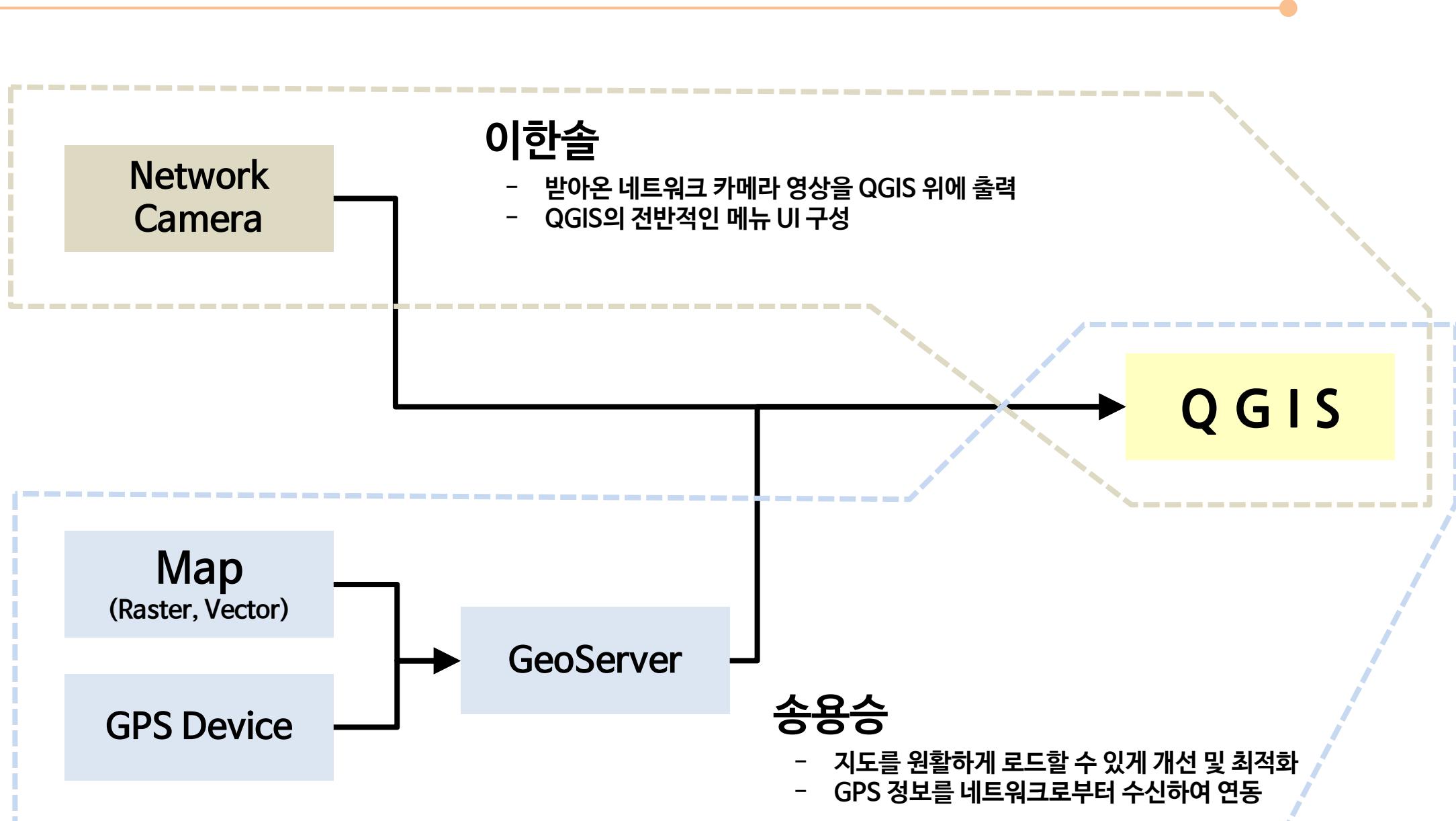
Qt 4.8.6



Python 2.7



PyQt 4.11.3





Map Part

(생략)



UI Part

4

프로젝트 전체적인 구성

개발/구현한 기능 - UI Part



Splash Image

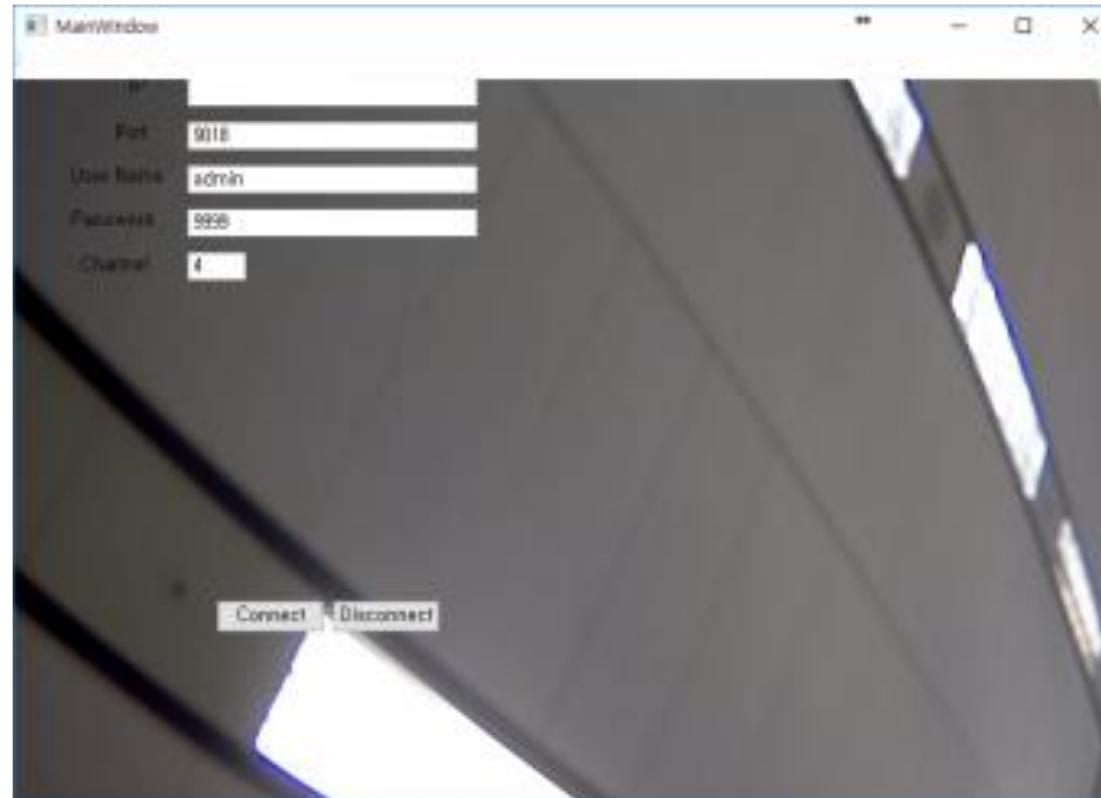


Icons

4

프로젝트 전체적인 구성

프로젝트에 사용되는 기능 - UI Part



영상 재생에 사용되는 인코더 SDK

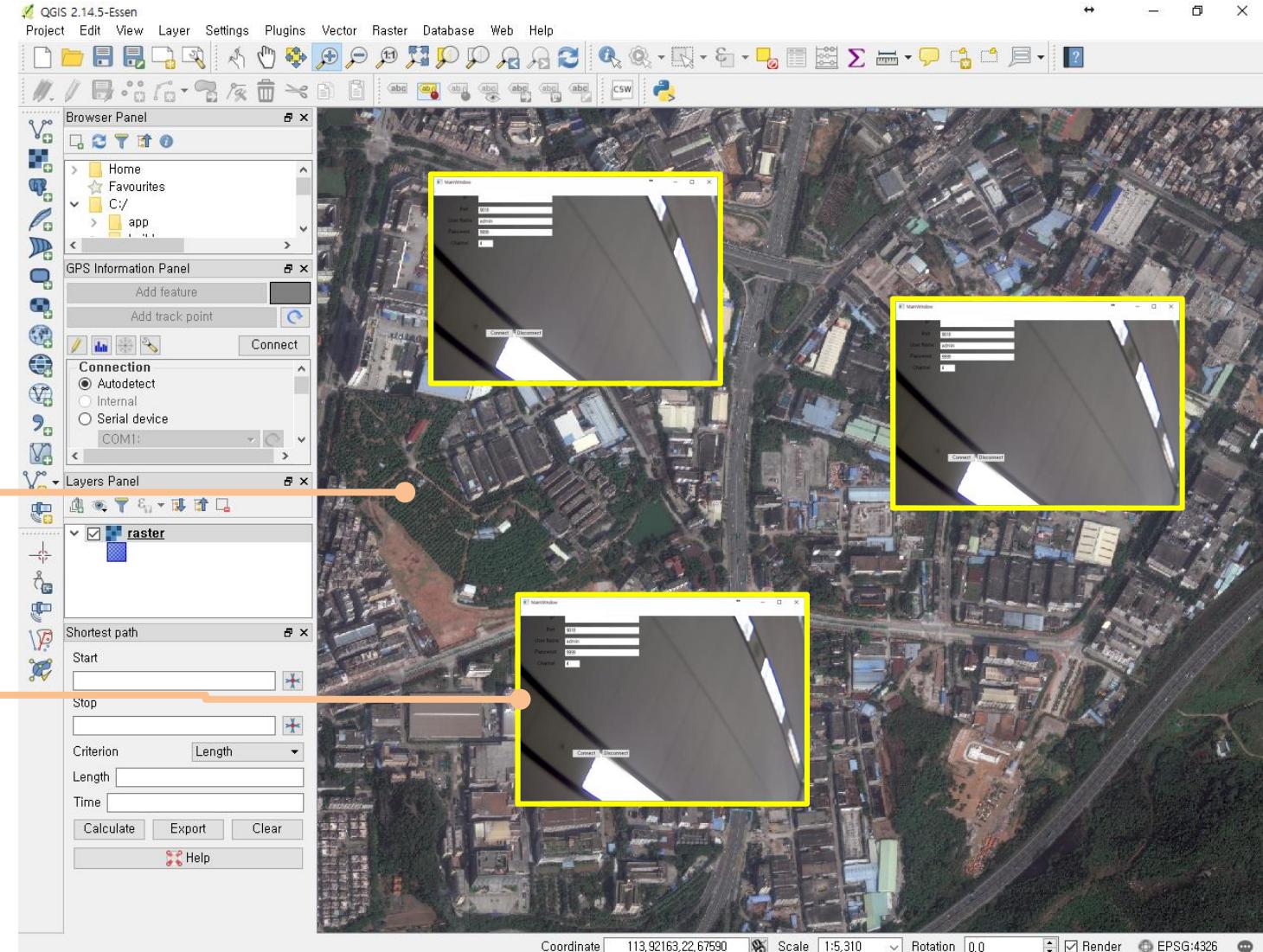
4

프로젝트 전체적인 구성

개발/구현한 기능 - UI Part



지도 및 네트워크 카메라
위치 정보 도시



특정 위치의 실시간 카메라
영상 재생



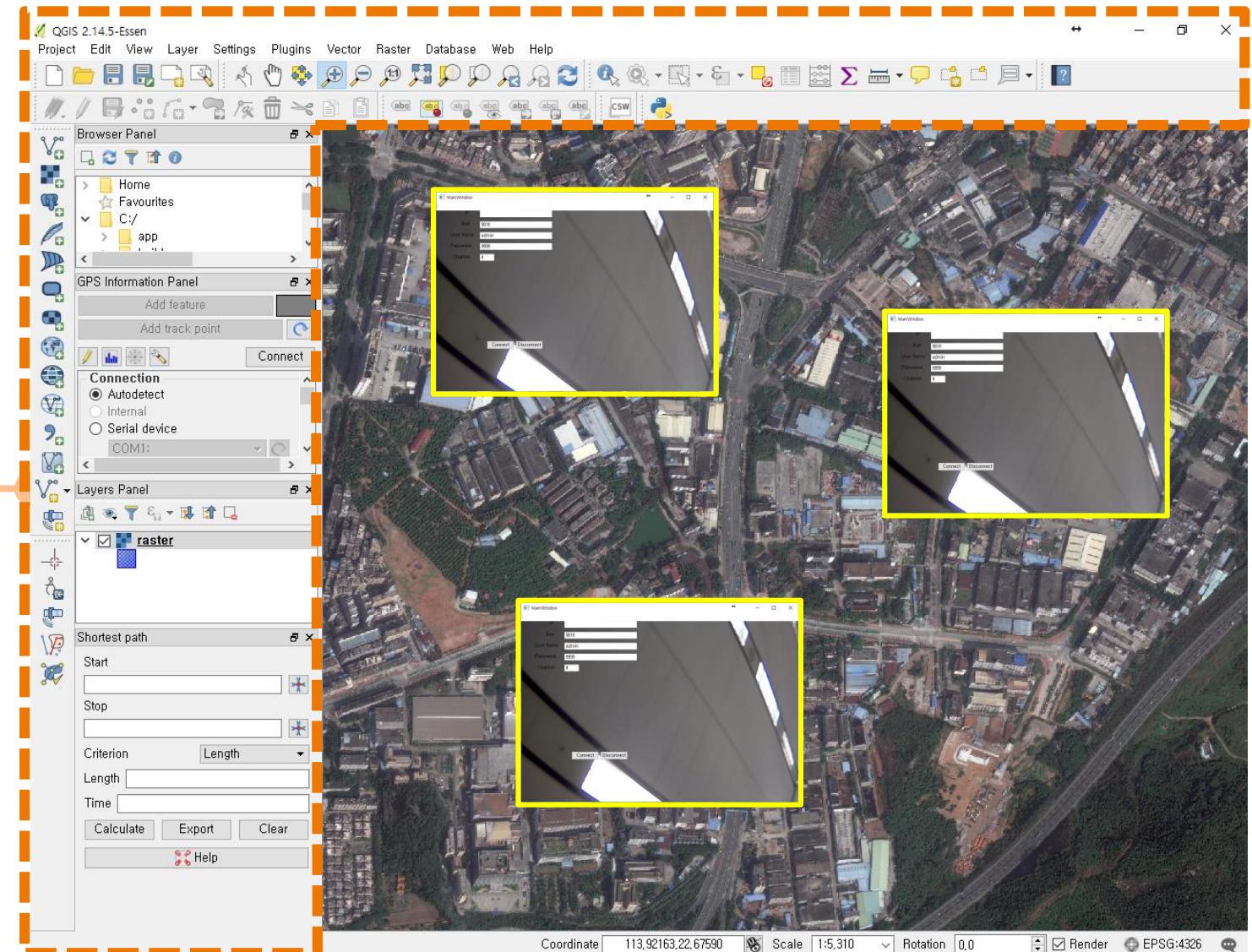
4

프로젝트 전체적인 구성

개발/구현한 기능 - UI Part



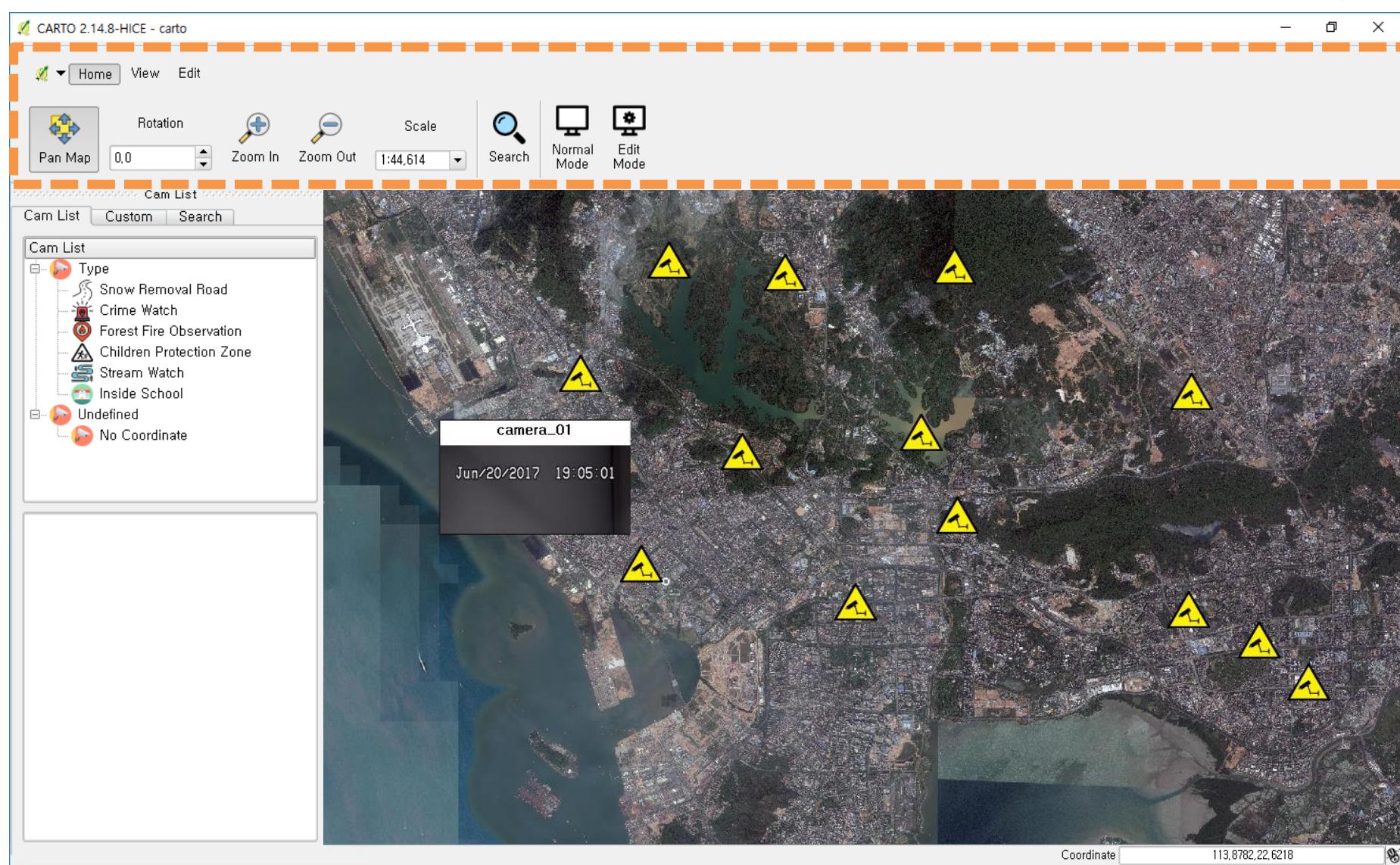
불필요한 메뉴는 Cut-out
아이콘, 기능 등을 추가/수정



4

프로젝트 전체적인 구성

개발/구현한 기능 - UI Part

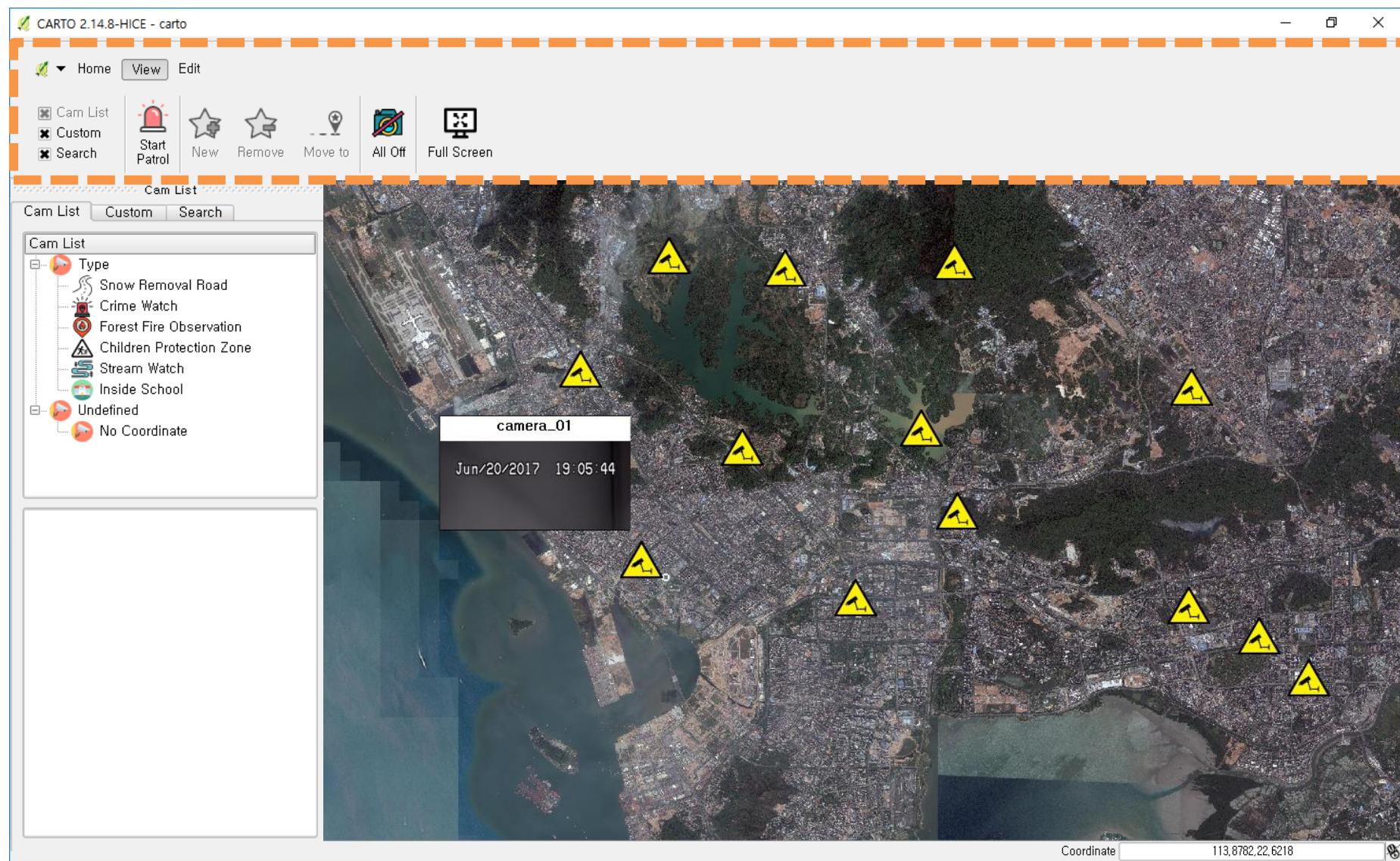


“필요한 메뉴 추가 / 필요 없는 메뉴 제거”

4

프로젝트 전체적인 구성

개발/구현한 기능 - UI Part



“필요한 메뉴 추가 / 필요 없는 메뉴 제거”

4

프로젝트 전체적인 구성

개발/구현한 기능 - UI Part

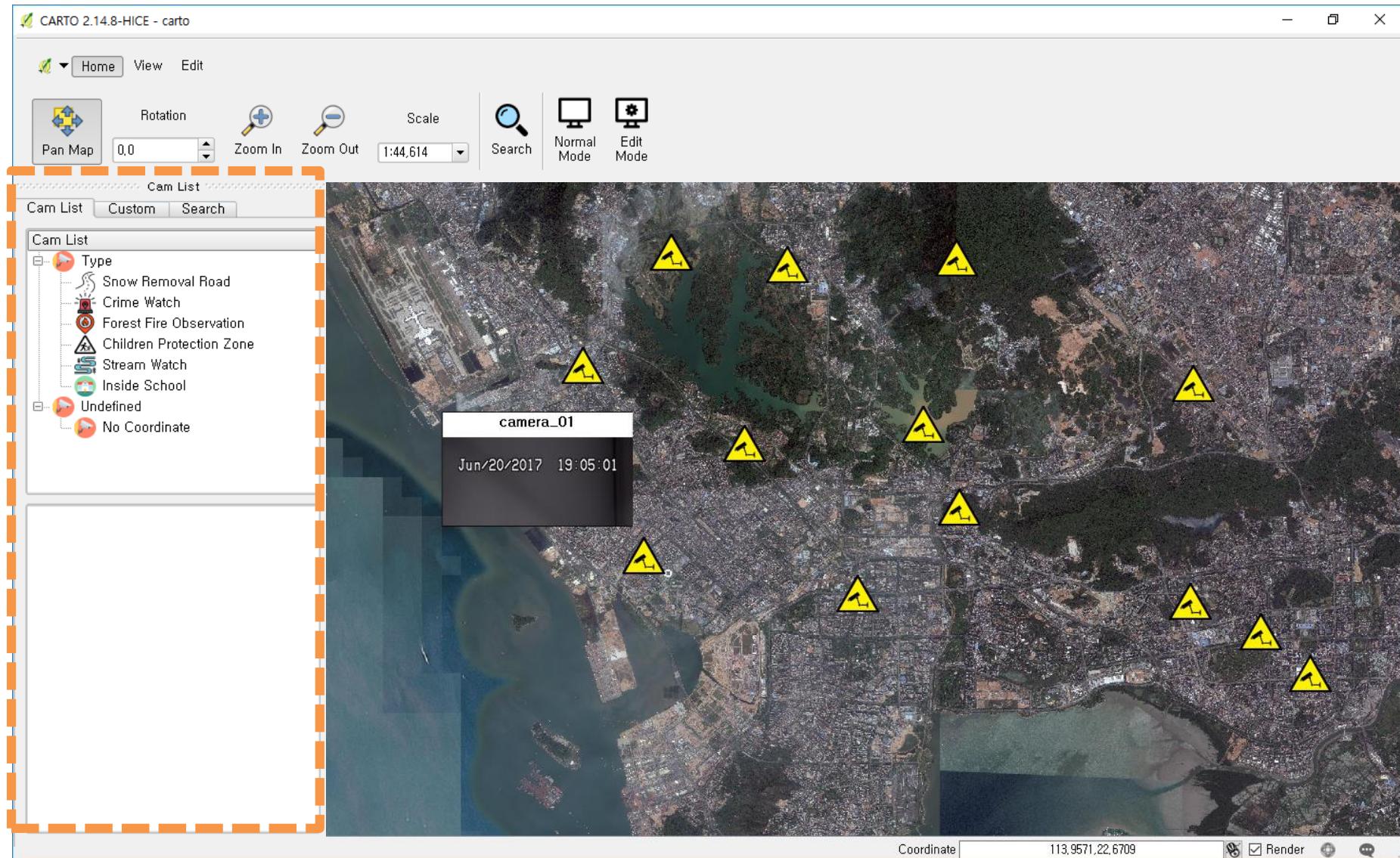


“필요한 메뉴 추가 / 필요 없는 메뉴 제거”

4

프로젝트 전체적인 구성

개발/구현한 기능 - UI Part



“ 좌측 패널 필요한 항목 추가 / 필요 없는 항목 제거 ”

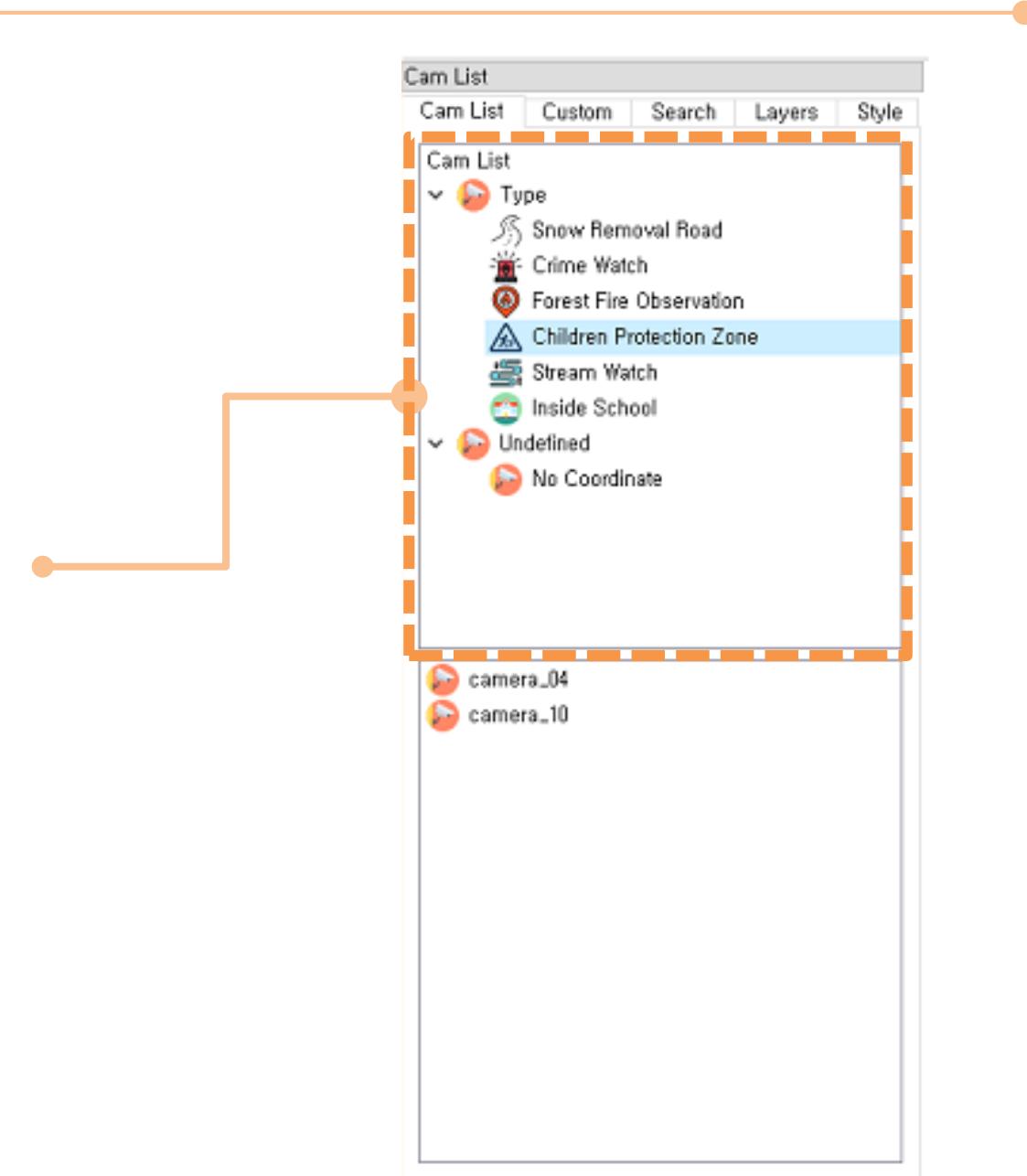
4

프로젝트 전체적인 구성

개발/구현한 기능 - UI Part



Cam마다 각자의 Type을 설정 후
각 타입별로 리스트화



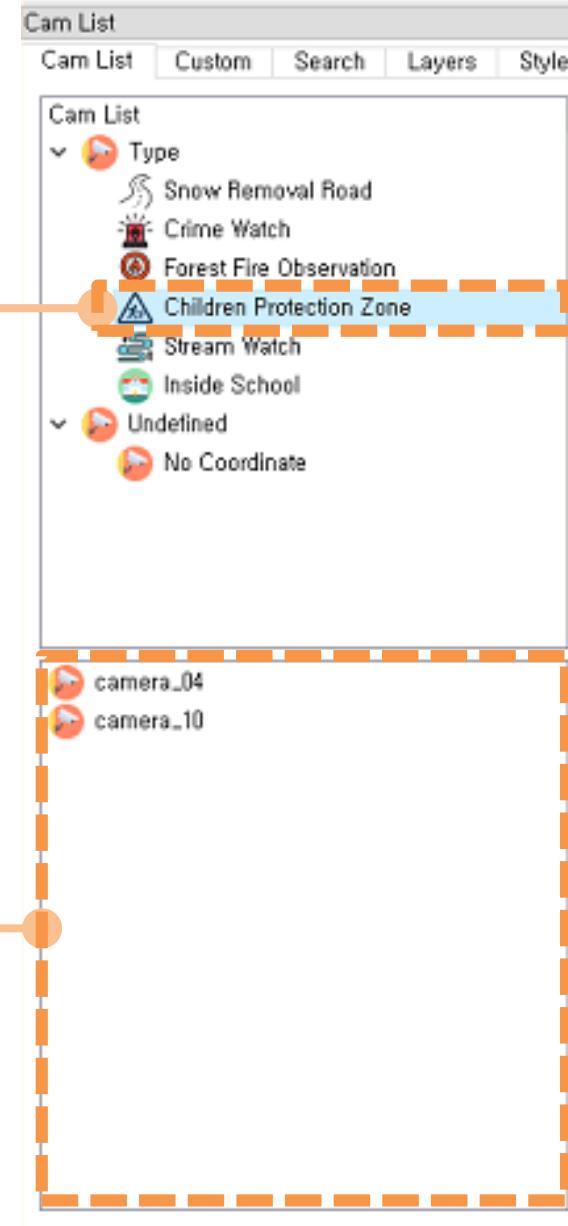
4

프로젝트 전체적인 구성

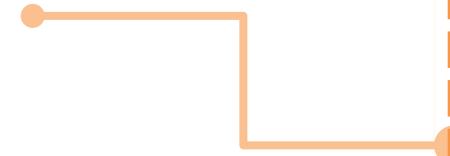
개발/구현한 기능 - UI Part



① 특정 타입을 선택하면



② 해당하는 타입의 Cam
영상이 하단 리스트에 표시



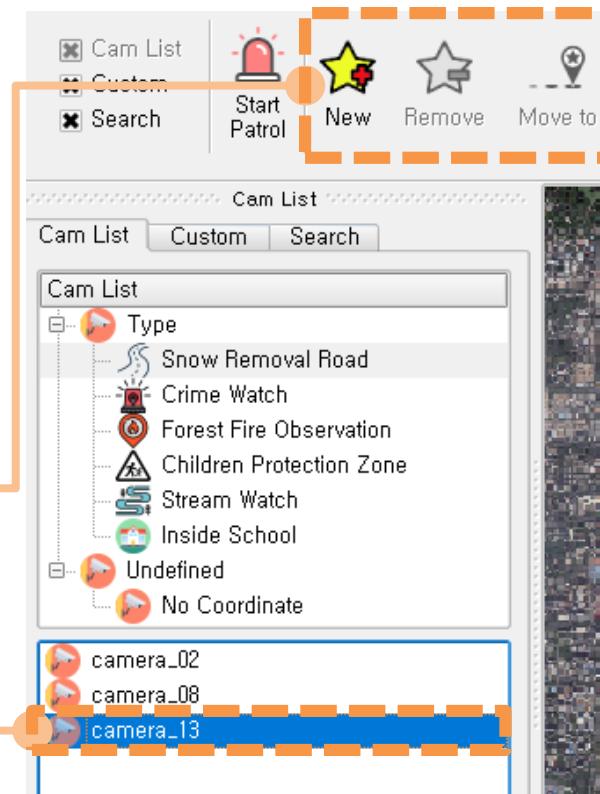
4

프로젝트 전체적인 구성

개발/구현한 기능 - UI Part



② 즐겨찾기 추가 버튼 활성화,
즐겨찾기에 추가 가능



① 특정 카메라를 선택하면

4

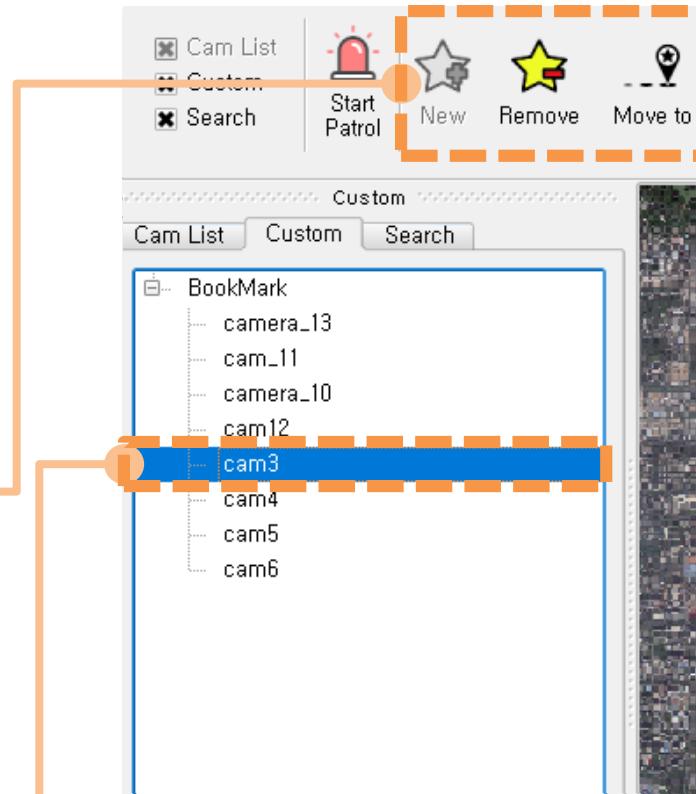
프로젝트 전체적인 구성

개발/구현한 기능 - UI Part



② 즐겨찾기 삭제/이동 버튼 활성화,
삭제/이동 가능

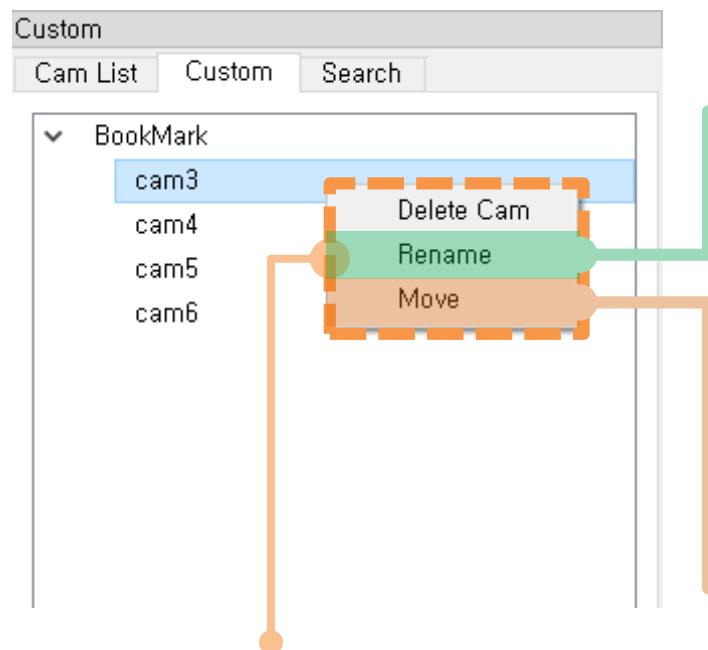
① 즐겨찾기에서 카메라를 선택하면



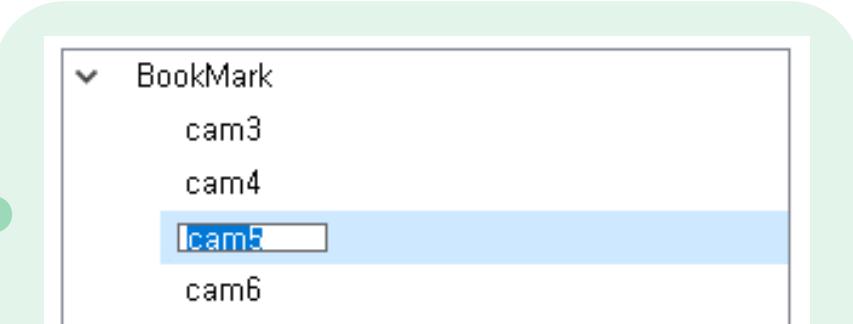
4

프로젝트 전체적인 구성

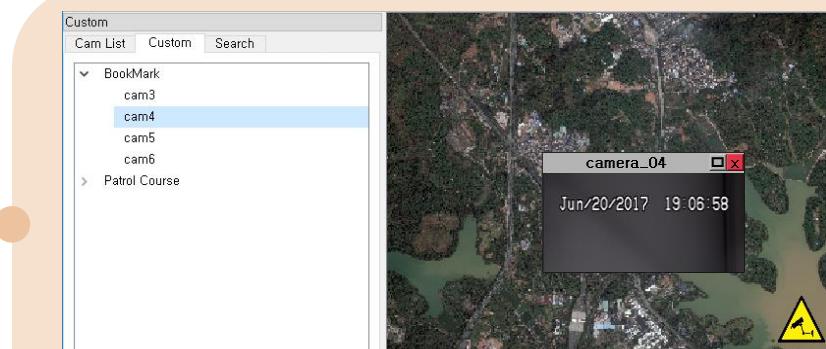
개발/구현한 기능 - UI Part



Custom 패널에 즐겨찾기를
위한 Context 메뉴 추가



즐겨찾기 Rename(이름 수정)

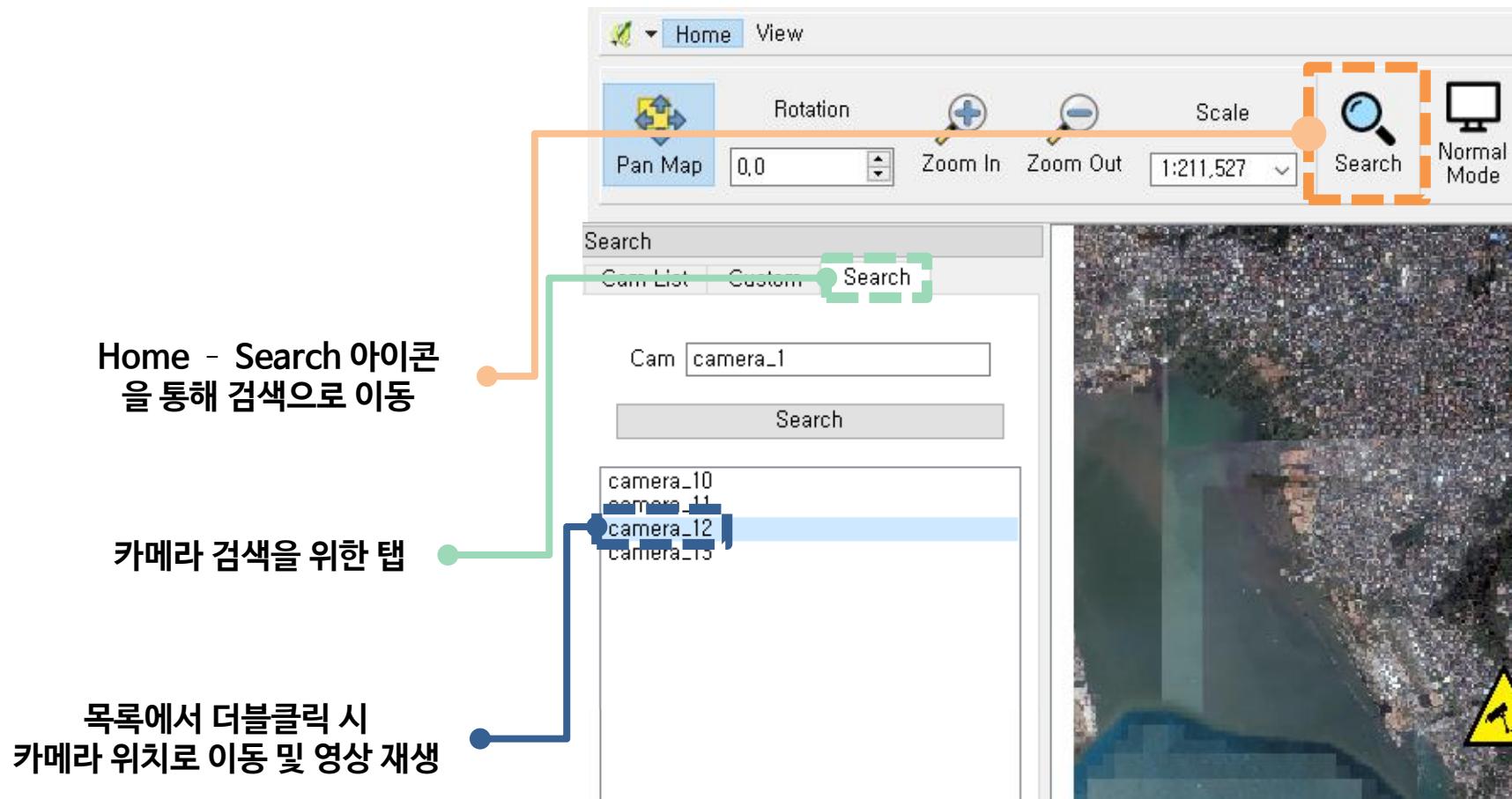


더블클릭 또는 Move를 눌러
카메라 위치로 이동 및 영상 재생

4

프로젝트 전체적인 구성

개발/구현한 기능 - UI Part





근접한 두 개 이상의 점(영상) 클릭 시 모아서 리스트 출력,
선택한 영상만을 재생

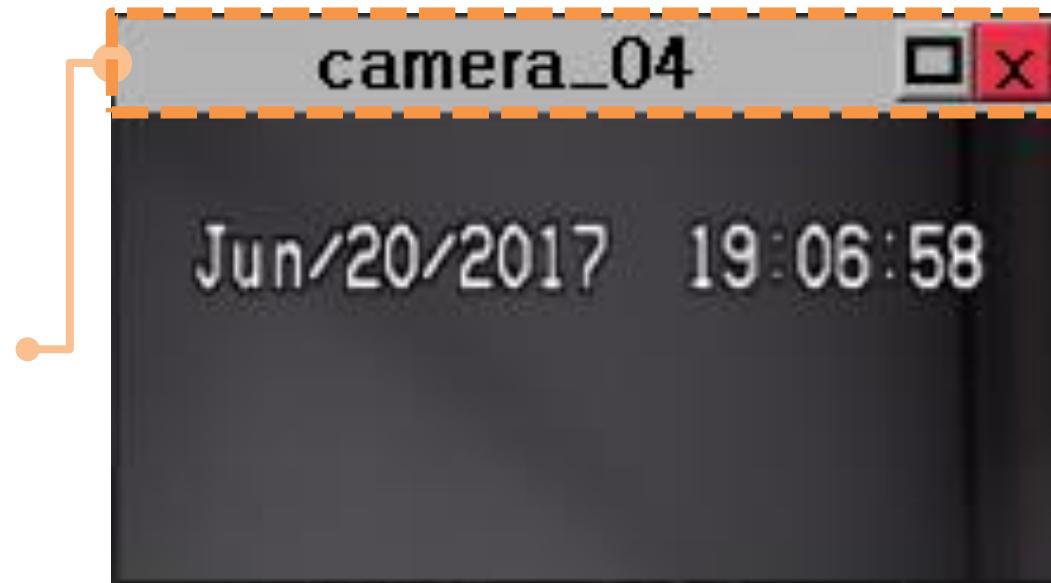
4

프로젝트 전체적인 구성

개발/구현한 기능 - UI Part



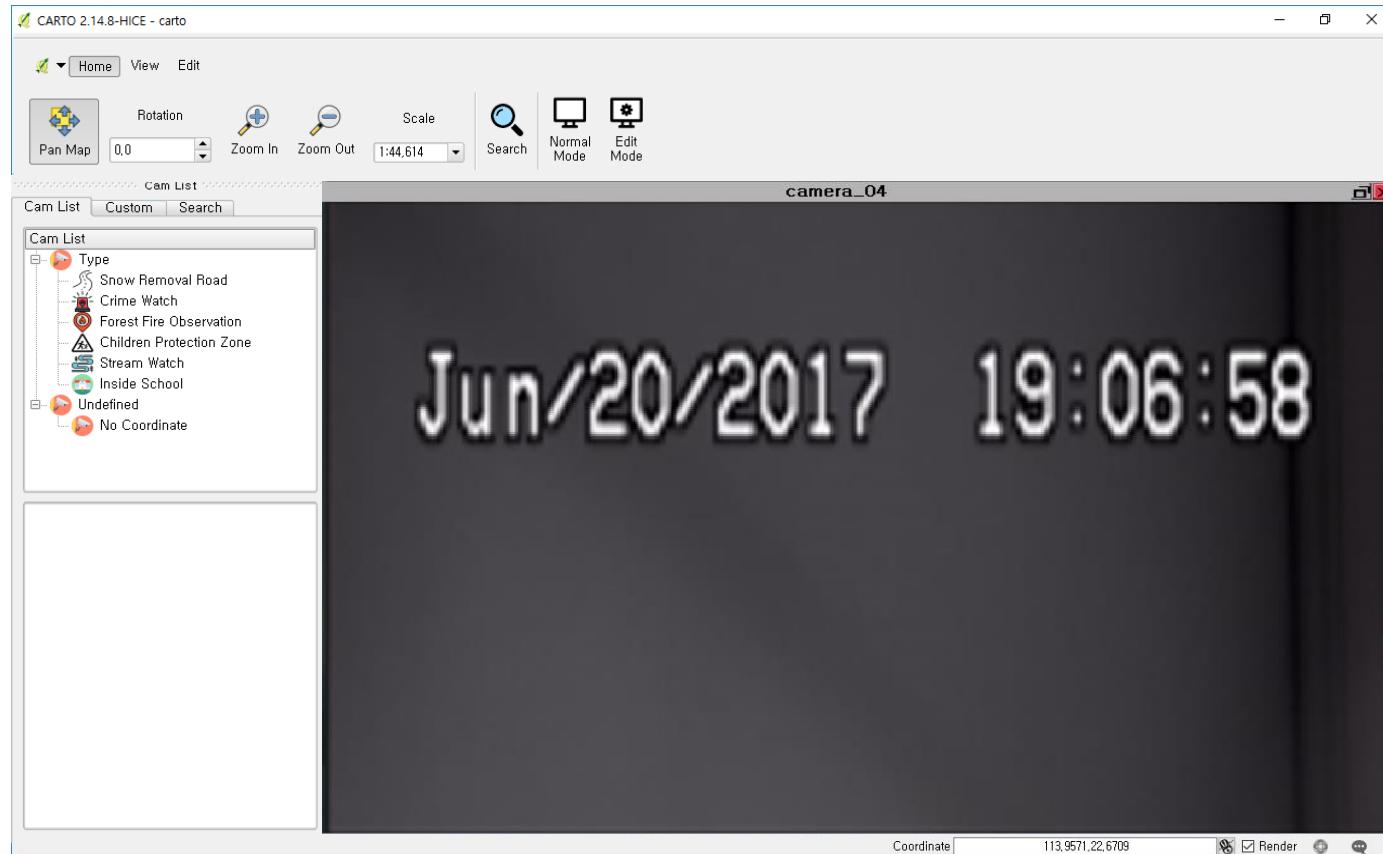
영상 위젯의 상단 Title 바에
최대/최소화 및 닫기 버튼 추가



4

프로젝트 전체적인 구성

개발/구현한 기능 - UI Part



CCTV 영상을 최대화 시킨 화면

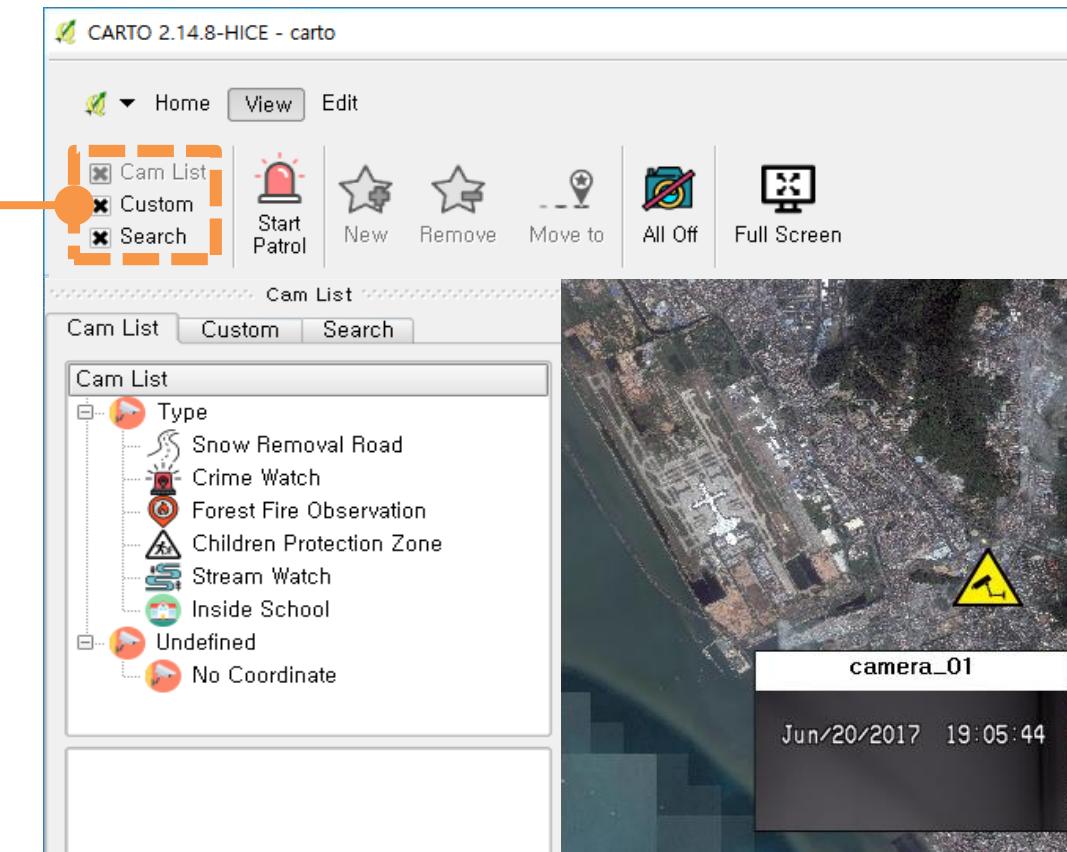
4

프로젝트 전체적인 구성

개발/구현한 기능 - UI Part



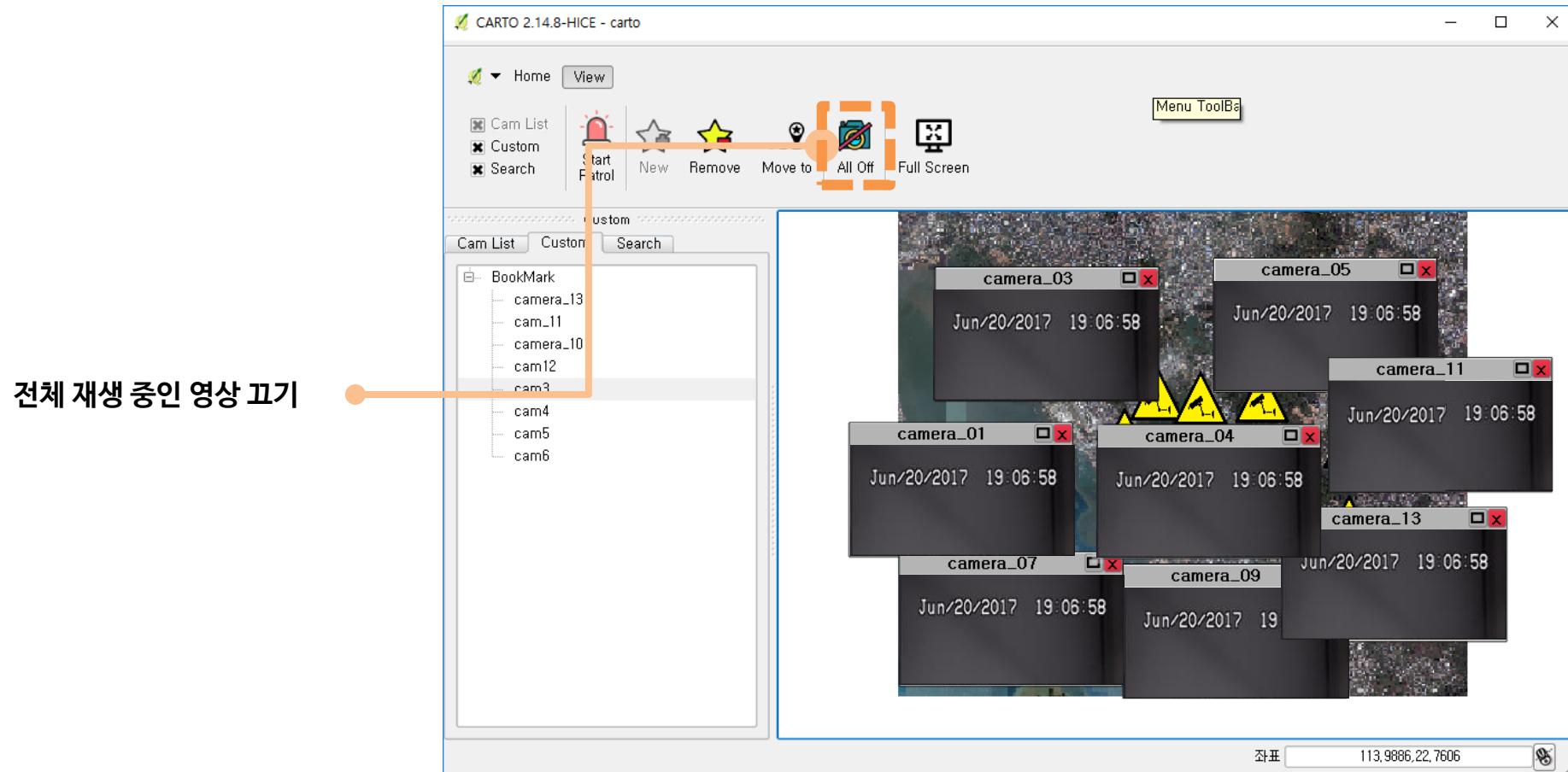
좌측 패널의 노출 설정(체크박스)



4

프로젝트 전체적인 구성

개발/구현한 기능 - UI Part

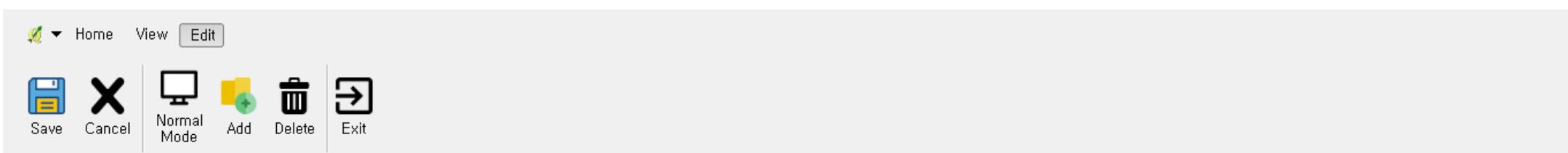
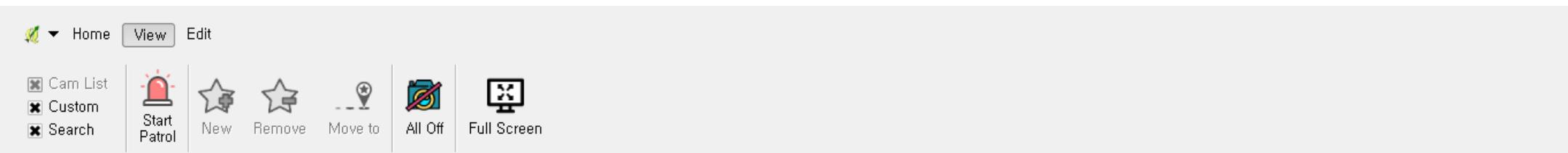
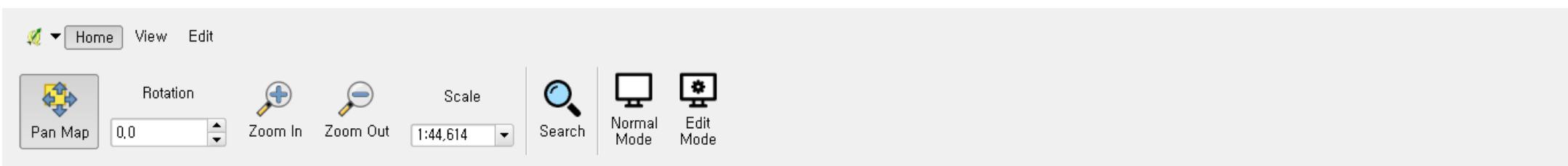


전체 재생 중인 영상 보기

4

프로젝트 전체적인 구성

보완된 기능 - UI Part

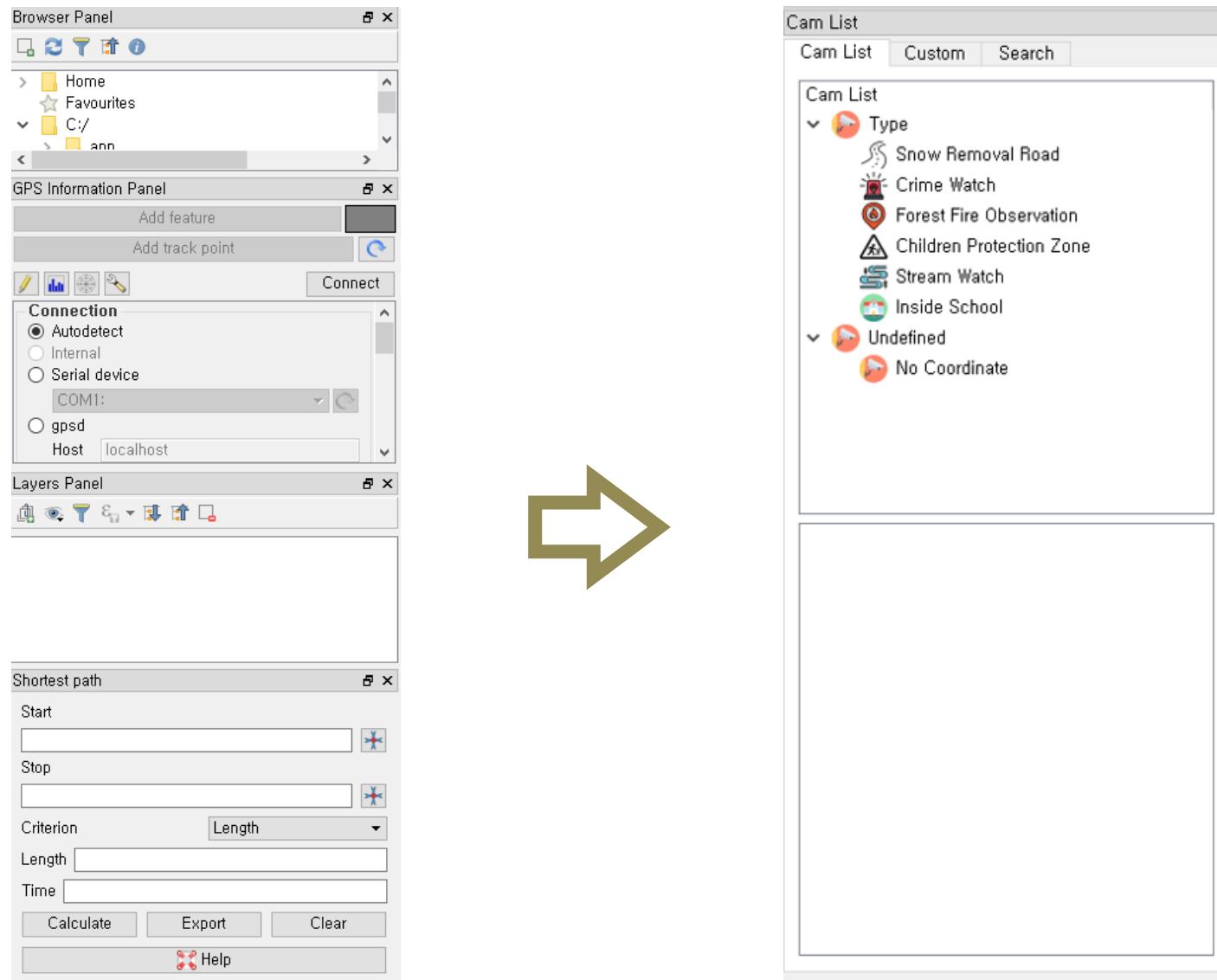


“직관적으로 쉽게 이해할 수 있도록 툴바 추가 보완”

4

프로젝트 전체적인 구성

보완된 기능 - UI Part



“어지럽게 늘어져 있는 패널들을 탭 형태로 정리”

감사합니다



MINI GOLF

송용승
윤천주
이한솔
서종인

목차

- 제목 및 동기
- 프로젝트에 대한 상세 설명
- 시연

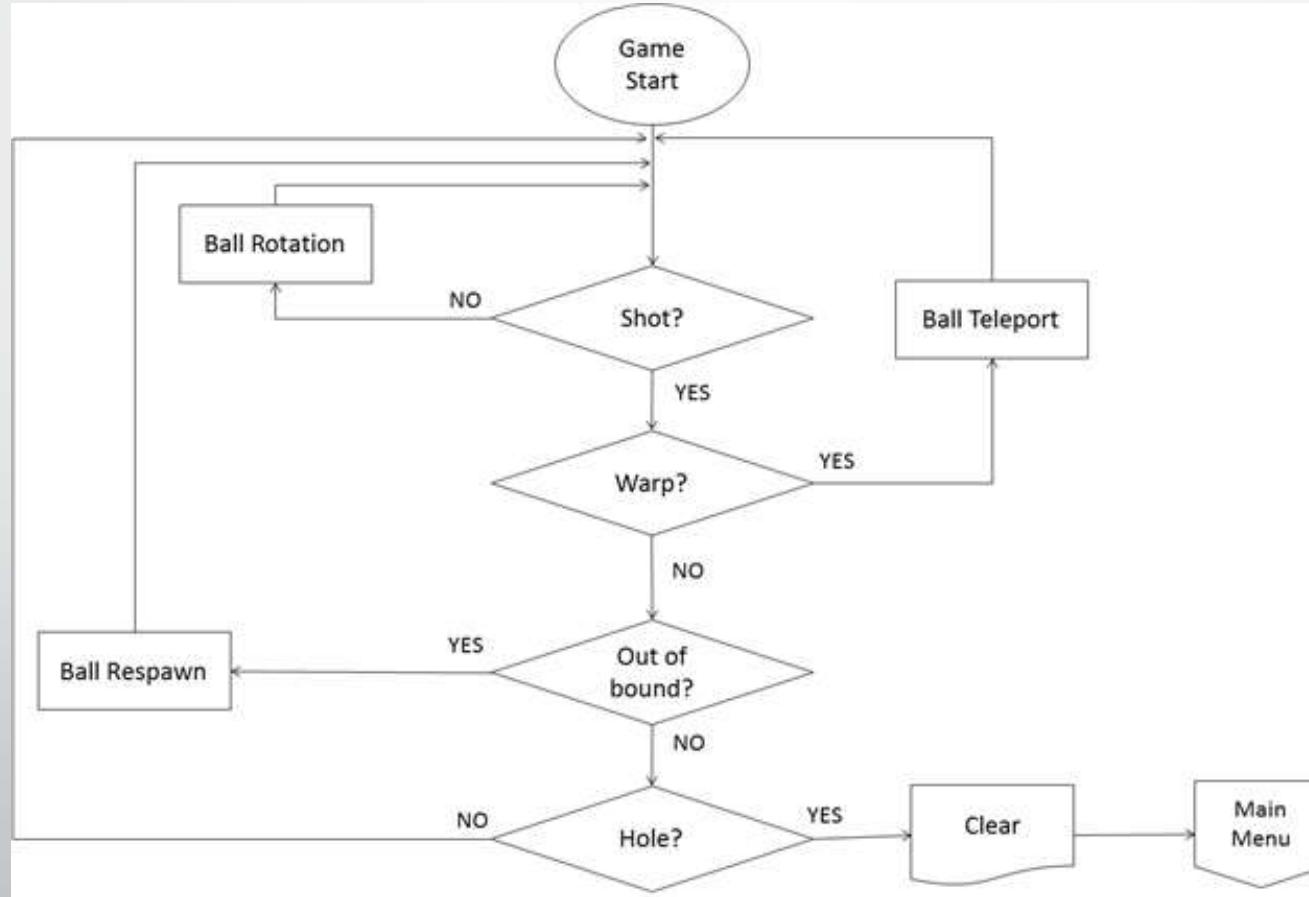
제목 및 동기



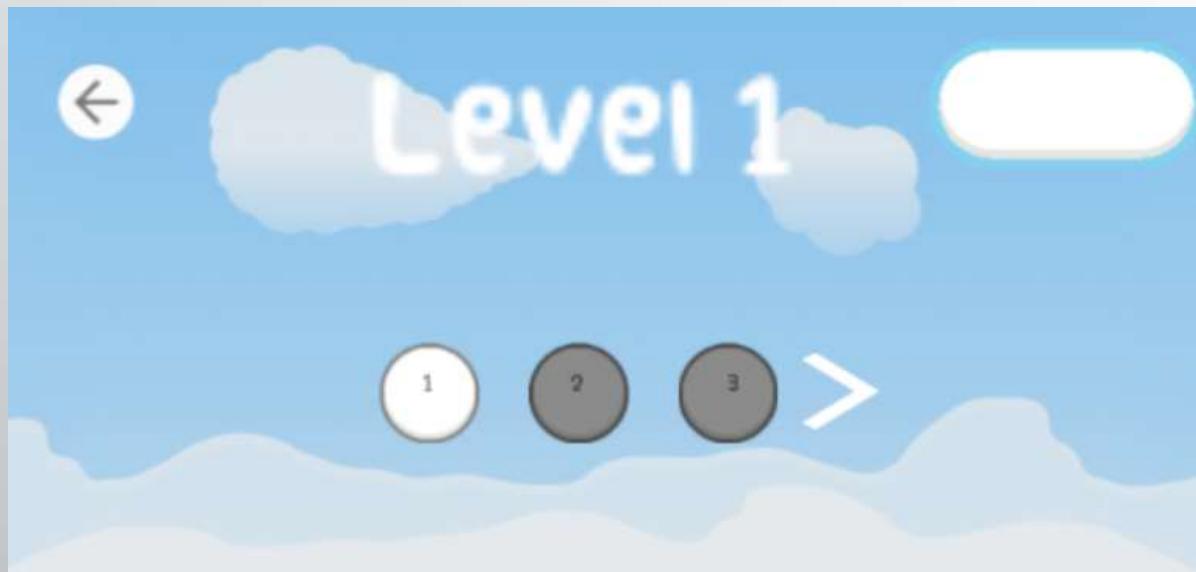
원터치 방식으로 간단히 즐길 수 있는 골프게임을 개발하고자 하였다.

프로젝트에 대한 상세 설명

- 플랫폼 및 개발환경
 - Unity 5.4.2f2(64bit), C#(Visual Studio 2013)
- 알고리즘



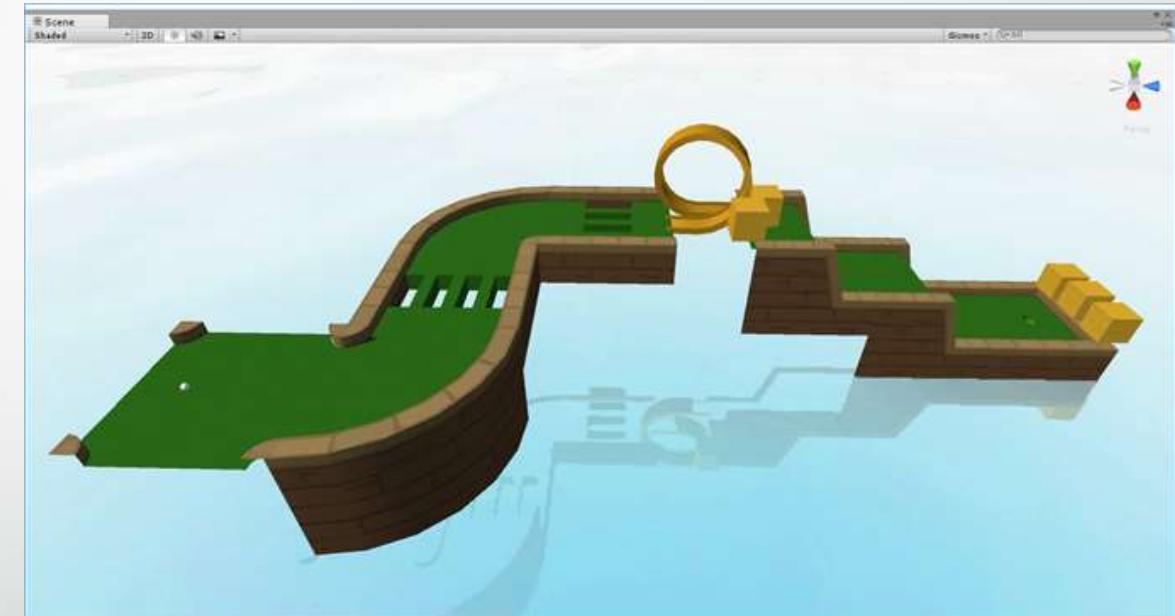
프로젝트에 대한 상세 설명



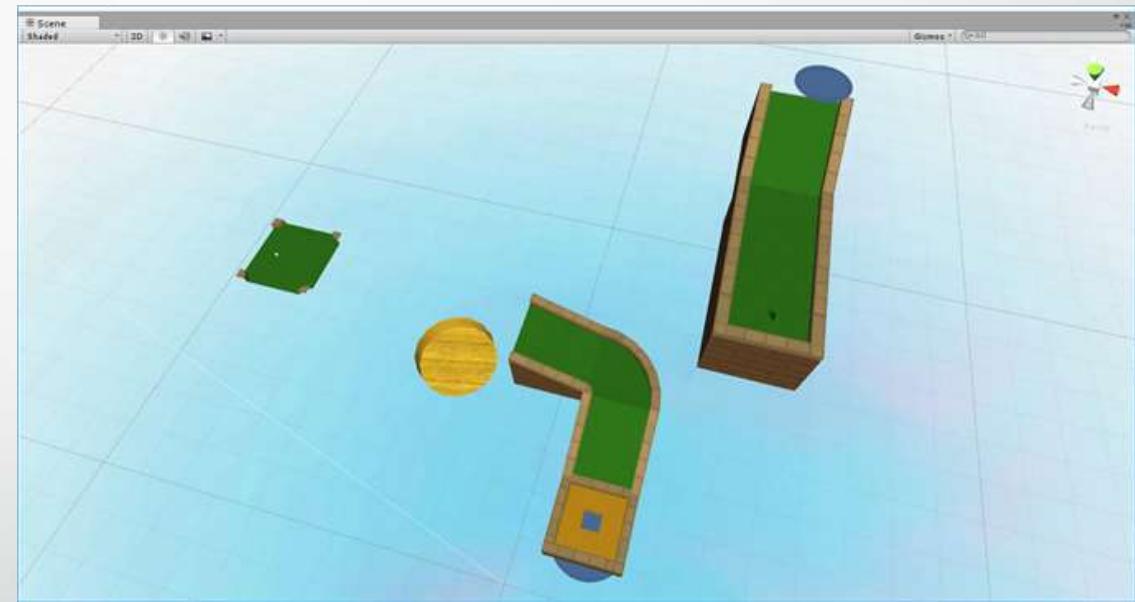
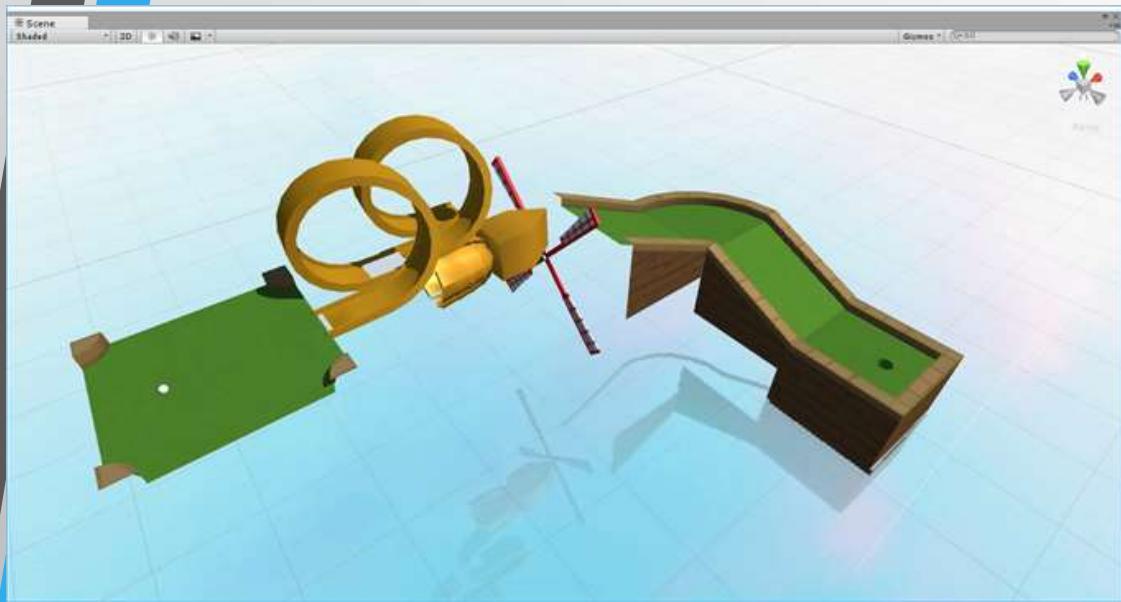
프로젝트에 대한 상세 설명



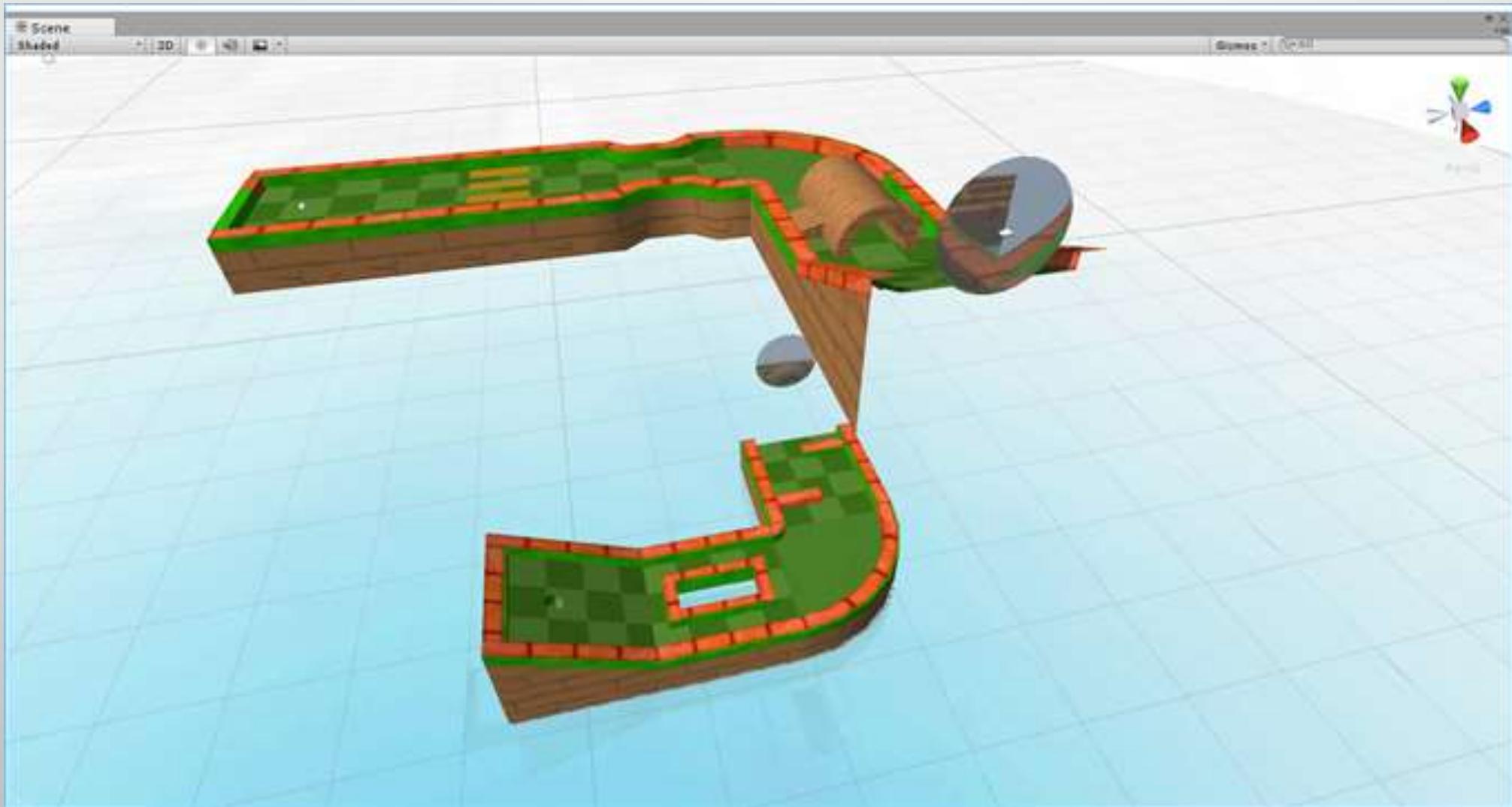
프로젝트에 대한 상세 설명



프로젝트에 대한 상세 설명



프로젝트에 대한 상세 설명

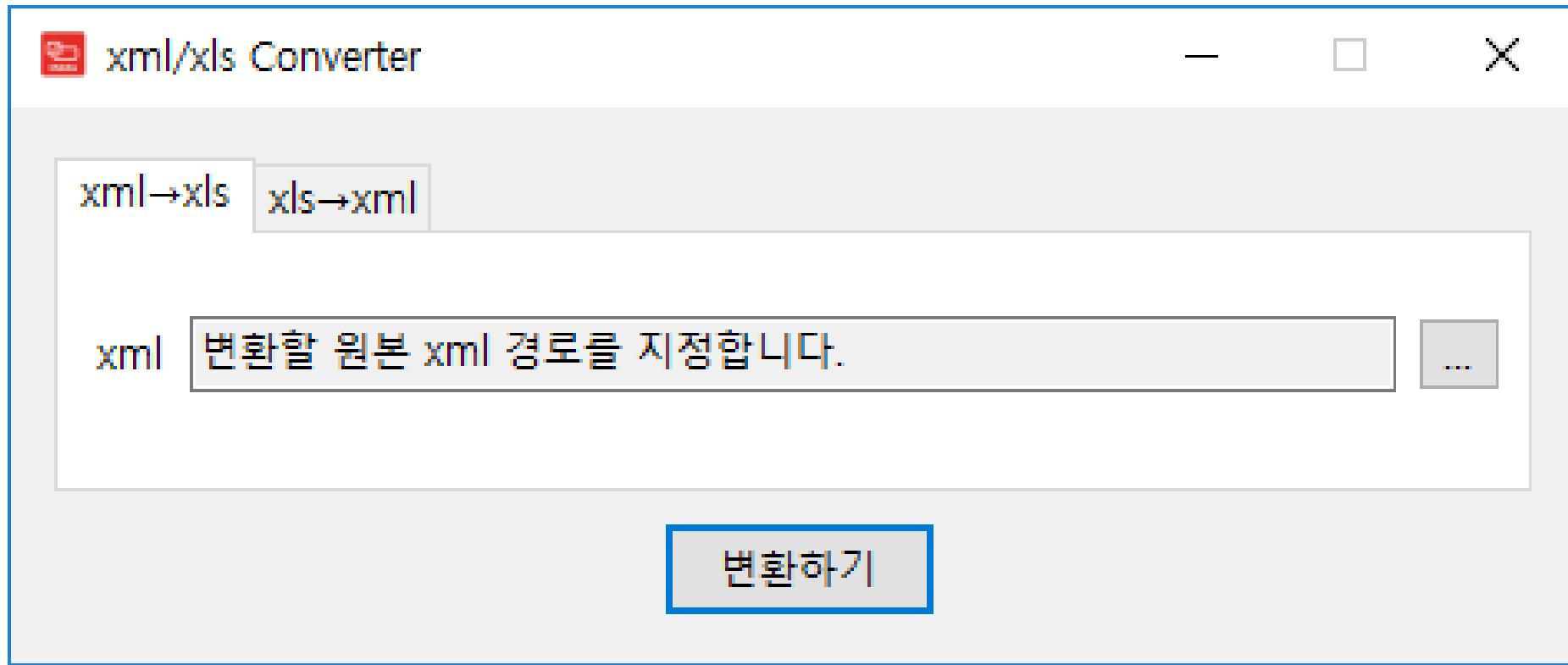


프로젝트에 대한 상세 설명

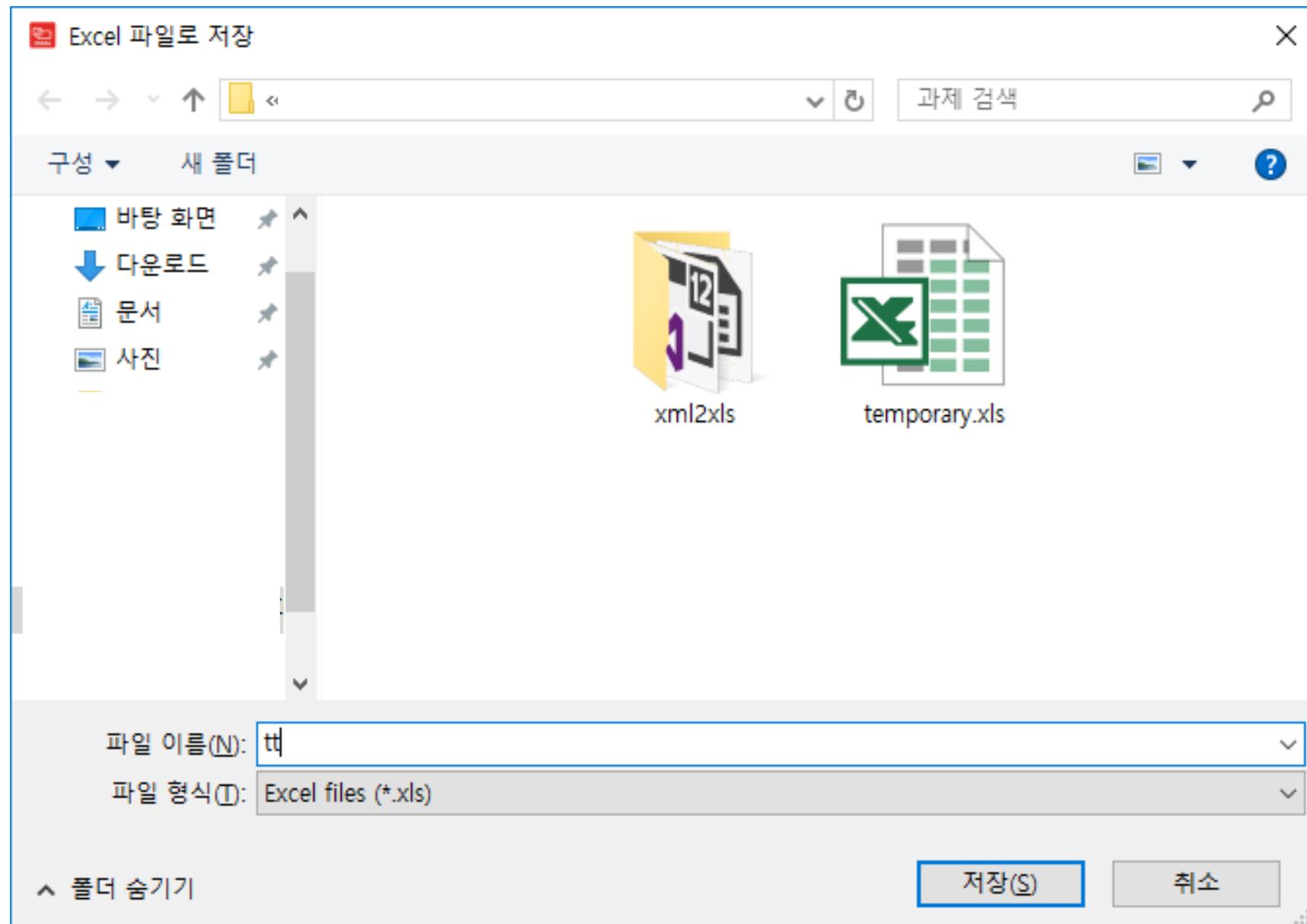


XML / XLS Converter

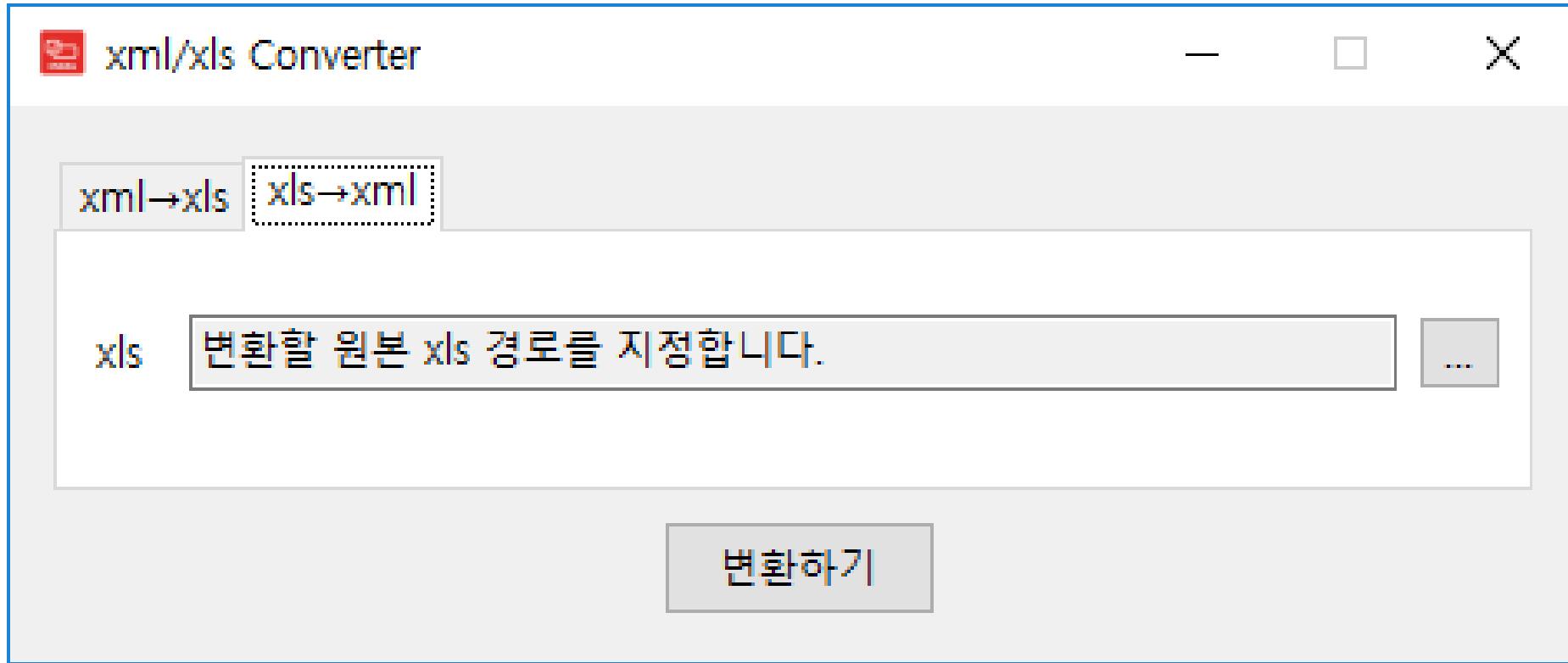
이한솔



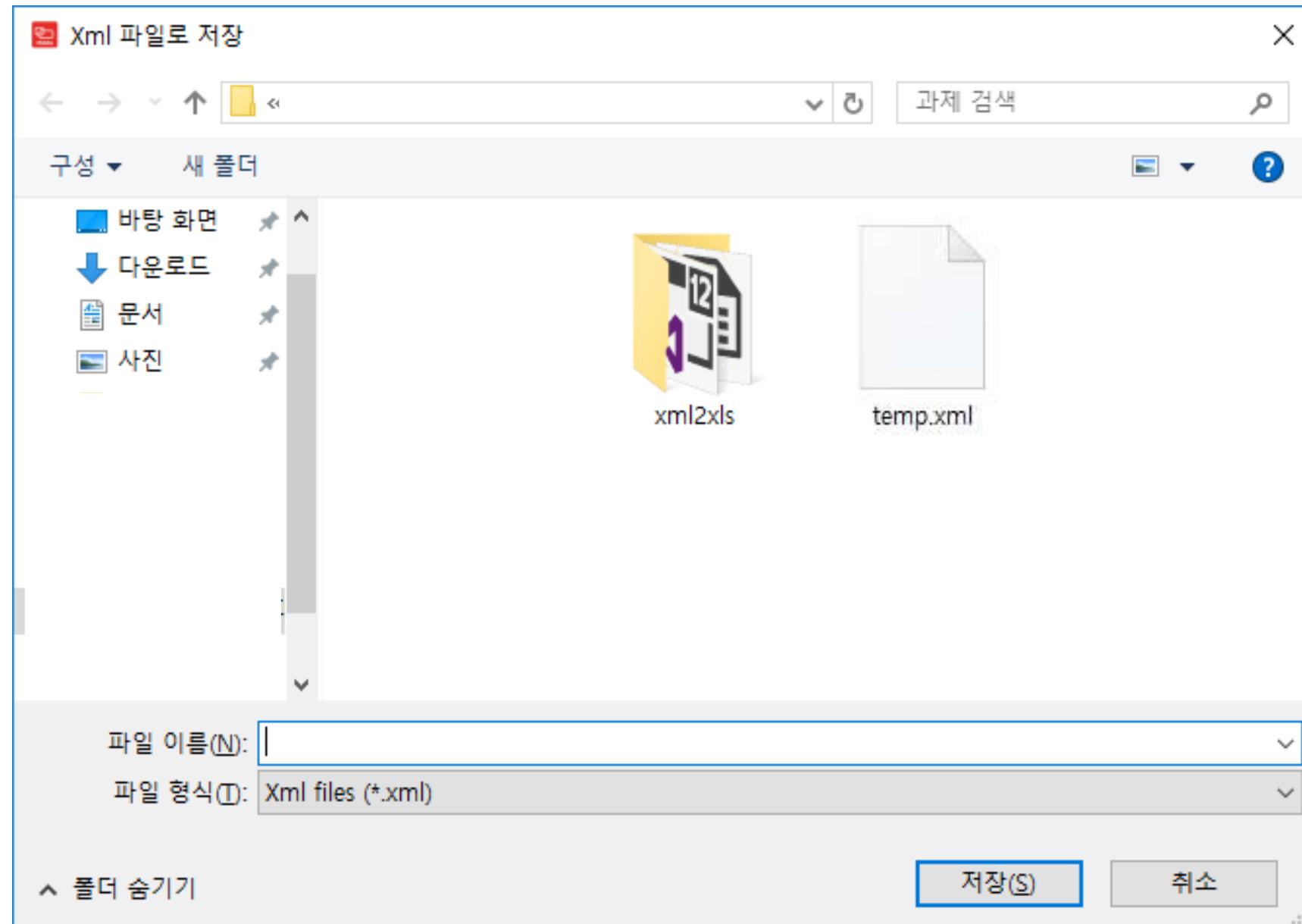
변환할 원본 XML 파일을 불러온 후 변환 버튼을 클릭



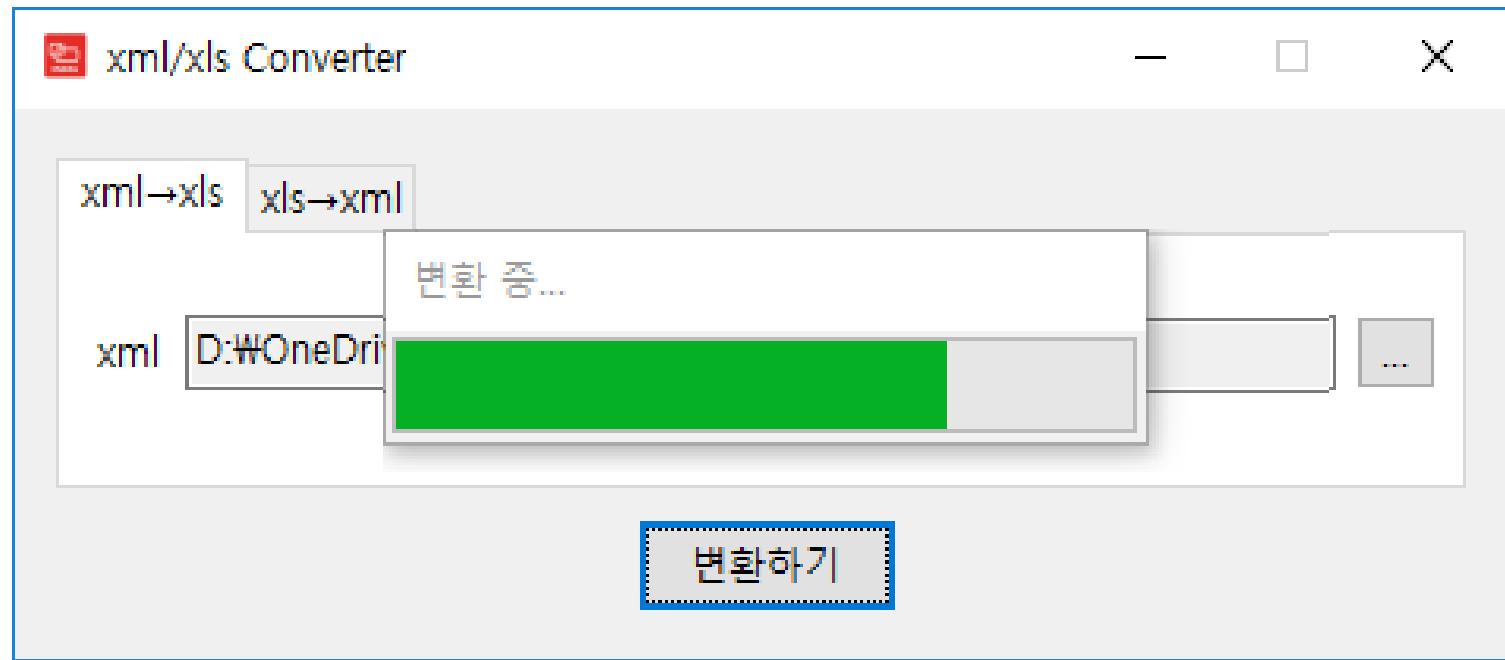
저장할 XLS 파일의 경로 및 파일 이름 지정 후 저장



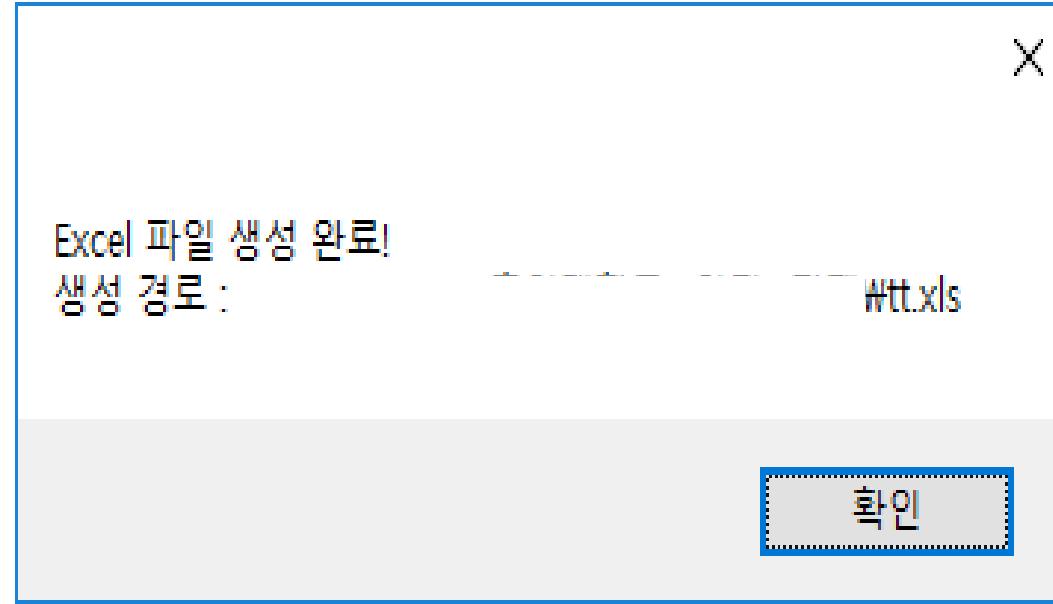
변환할 원본 XLS 파일을 불러온 후 변환 버튼을 클릭



저장할 XML 파일의 경로 및 파일 이름 지정 후 저장



변환 중 Progress Bar로 진행 상황 표시



변환 완료 후 완료 메시지 팝업

temporary.xls [호환 모드] - Excel

파일 흡입 페이지 레이아웃 수식 데이터 검토 보기 추가 기능 템 편집 입력하세요... Hansol Rhee 공유

붙여넣기 클립보드 글꼴 맞춤 표시 형식 스타일 셀 편집

A1 CHANNEL

	A	B	C	D	E	F	G
1	CHANNEL	ENABLE	URL	PORT	TYPE	STATUS	OP
2	1-1	1	ndsp://	0	0	1	-model -u adr
3	2-2	1	ndsp://	0	0	1	-model -u adr
4	3-3	1	ndsp://	0	0	1	-model -u adr
5	4-4	1	ndsp://	0	0	1	-model -u adr
6	5-5	1	ndsp://	0	0	1	-model -u adr
7	6-6	1	ndsp://	0	0	1	-model -u adr
8	7-7	0	http://	0	2	0	-model OTHER
9	8-8	0	http://	0	2	0	-model OTHER
10	9-9	0	http://	0	2	0	-model OTHER
11	10-10	0	http://	0	2	0	-model OTHER
12	11-11	0	http://	0	2	0	-model OTHER
13	12-12	0	http://	0	2	0	-model OTHER
14	13-13	0	http://	0	2	0	-model OTHER
15	14-14	0	http://	0	2	0	-model OTHER
16	15-15	0	http://	0	2	0	-model OTHER
17	16-16	0	http://	0	2	0	-model OTHER

准备 RELAY NETWORK NDAS ENABLE ... + 100 %

변환된 XLS 파일 예시

```
<?xml version="1.0" encoding="UTF-8"?>
<DATABASE>
  <RELAY>
    <CHANNEL RANGE="1-1">...</CHANNEL>
    <CHANNEL RANGE="2-2">...</CHANNEL>
    <CHANNEL RANGE="3-3">...</CHANNEL>
    <CHANNEL RANGE="4-4">...</CHANNEL>
    <CHANNEL RANGE="5-5">...</CHANNEL>
    <CHANNEL RANGE="6-6">...</CHANNEL>
    <CHANNEL RANGE="7-7">...</CHANNEL>
    <CHANNEL RANGE="8-8">...</CHANNEL>
    <CHANNEL RANGE="9-9">...</CHANNEL>
    <CHANNEL RANGE="10-10">...</CHANNEL>
  </RELAY>
  <NETWORK>
    <IP_CONFIG>...</IP_CONFIG>
    <SNTP_CONFIG>...</SNTP_CONFIG>
    <RTSP_CONFIG>...</RTSP_CONFIG>
    <SYSLOG_CONFIG>...</SYSLOG_CONFIG>
  </NETWORK>
  <APPLICATION>
    <NDAS_CONFIG>...</NDAS_CONFIG>
    <ENABLE_CONFIG>...</ENABLE_CONFIG>
    <SCHEDULE_CONFIG>...</SCHEDULE_CONFIG>
    <SCHEDULE_DATE_CONFIG>...</SCHEDULE_DATE_CONFIG>
    <DISK_REGISTER_CONFIG>...</DISK_REGISTER_CONFIG>
  </APPLICATION>
</DATABASE>
```

변환된 XLS 파일 예시

<MyShell – 명령어 옵션 처리>

```
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>
#include <signal.h>

#define MAX_STR 1024
#define MAX_ARG 64
#define bool int
#define true 1
#define false 0

static void sig_child(int);
void executeOne(int argc, char *argv[]);
int parsingPipes(char *str, char **argv);

bool isBG = false;

int main(int argc, char *argv[]) {
    char *str = (char*)malloc(MAX_STR);
    char *str2 = (char*)malloc(MAX_STR);
    char **argv1 = (char**)malloc(MAX_ARG);
    char **argv2 = (char**)malloc(MAX_ARG);
    char **argv3 = (char**)malloc(MAX_ARG);
    char **argv4 = (char**)malloc(MAX_ARG);

    char **parsedPipe = (char**)malloc(MAX_ARG);
    int i, status, pipeCount, index;
    char *temp;
    char *before, *after;
    int fdr;
    int pfd[MAX_ARG][2];
    pid_t pid;
    pid_t pids[64] = { 0 };

    signal(SIGCHLD, sig_child);

    if (argc >= 2)
    {
        executeOne(argc, &argv[1]);
    }
    else
    {
        while (1) {
            printf("myshell$ ");

            if (fgets(str, MAX_STR, stdin) == NULL) {
                printf("Ctrl-D\n");
                exit(0);
            }

            if (!strcmp(str, "\n")) continue;

            strtok(str, "\n");

```

```

strcat(str, "WO");
fflush(stdin);

if (strchr(str, '|') != NULL) {
    //PIPE!!!!!!!!!!!!!!
    pipeCount = parsingPipes(str, parsedPipe);

    for (i = 0; i < pipeCount; i++)
    {
        if (i != 0)
            {//이전거의 1번을 닫는다
                close(pfd[i - 1][1]);
            }
        if (i != pipeCount - 1)
            {//마지막. 현재의 0번을 닫는다
                if (pipe(pfd[i]) == -1)
                    perror("pipe");
                exit(1);
            }
    }

    if ((pids[i] = fork()) != 0)
    {
        continue;
    }
    if (i != pipeCount - 1)
    {
        dup2(pfd[i][1], STDOUT_FILENO);
        close(pfd[i][0]);
        close(pfd[i][1]);
    }

    if (i != 0)
    {
        dup2(pfd[i - 1][0], STDIN_FILENO);
        close(pfd[i - 1][0]);
    }

    index = 0;
    temp = strtok(parsedPipe[i], " ");
}

while (temp != NULL)
{
    argv1[index] = temp;
    temp = strtok(NULL, " ");
    index++;
}

argv1[index] = (char*)0;

execvp(argv1[0], argv1);
perror("execvp error");
exit(-1);
}

for(i=0; i<pipeCount; i++)
{
    close(pfd[i][0]);
    close(pfd[i][1]);
}

for (i = 0; i < pipeCount; i++)

```

```

        {
            wait(&status);
        }
    }
else if ((strstr(str, "2>") != NULL) ||
          (strchr(str, '>') != NULL) || (strchr(str, '<') != NULL))
{
    if ((pid = fork()) == 0)
    {
        if (strstr(str, "2>") != NULL) {
            //Redirection error out!!!!!!!!!!!!!!
            before = strtok(str, "2");
            after = strtok(NULL, ">");
            i = 0;
            temp = strtok(after, " ");
            while (temp != NULL)
            {
                argv2[i] = temp;
                temp = strtok(NULL, " ");
                i++;
            }
            argv2[i] = (char*)0;

            fdr = open(argv2[0], O_WRONLY | O_CREAT | O_TRUNC, 0644);
            if (fdr == -1) {
                perror("file open error"); exit(1);
            }
            if (dup2(fdr, STDERR_FILENO) == -1) {
                perror("fd dup error"); exit(1);
            }
            close(fdr);

            strncpy(str2, before, MAX_STR);
            strncpy(str, str2, MAX_STR);
        }
        if (strchr(str, '>') != NULL)
        {
            //Redirection out!!!!!!!!!!!!!!
            before = strtok(str, ">");
            after = strtok(NULL, ">");
            i = 0;
            temp = strtok(after, " ");
            while (temp != NULL)
            {
                argv3[i] = temp;
                temp = strtok(NULL, " ");
                i++;
            }
            argv3[i] = (char*)0;

            fdr = open(argv3[0], O_WRONLY | O_CREAT | O_TRUNC, 0644);
            if (fdr == -1) {
                perror("file open error"); exit(1);
            }
            if (dup2(fdr, STDOUT_FILENO) == -1) {
                perror("fd dup error"); exit(1);
            }
            close(fdr);
        }
    }
}

```

```

        strncpy(str2, before, MAX_STR);
        strncpy(str, str2, MAX_STR);
    }
    if (strchr(str, '<') != NULL) {
        //Redirection in!!!!!!!!!!!!!!
        before = strtok(str, "<");
        after = strtok(NULL, "<");

        i = 0;
        temp = strtok(after, " ");
        while (temp != NULL)
        {
            argv4[i] = temp;
            temp = strtok(NULL, " ");
            i++;
        }
        argv4[i] = (char*)0;

        fdr = open(argv4[0], O_RDONLY);
        if (fdr == -1) {
            perror("file open error"); exit(1);
        }
        if (dup2(fdr, STDIN_FILENO) == -1) {
            perror("fdr dup error"); exit(1);
        }
        close(fdr);
    }
    temp = strtok(before, " ");
    i = 0;
    while (temp != NULL)
    {
        argv1[i] = temp;
        temp = strtok(NULL, " ");
        i++;
    }
    argv1[i] = (char*)0;
    execvp(argv1[0], argv1);
    printf("myshell: %s: command not found\n", argv1[0]);
    exit(0);
}
else
{
    if (strchr(str, '&') != NULL) {
        //Background!!!!!!!!!!!!!!
        isBG = true;
        strtok(str, "&");
        strcat(str, " ");
    }

    strcat(str, "W0");

    i = 0;
    temp = strtok(str, " ");

    while (temp != NULL)
    {
        argv1[i] = temp;
        temp = strtok(NULL, " ");

```

```

                i++;
            }
            argv1[ i ] = (char*)0;

            if (argv1[0] != NULL) {
                if ((pid = fork()) == 0) {
                    execvp(argv1[0], argv1);
                    printf( "-myshell: %s: command not found\n", argv1[0]);
                    exit(0);
                }
                else {
                    if (isBG) {
                        isBG = false;
                        printf( "[background job start : %d]\n", pid);
                        /* add pid to some list to track jobs */
                    }
                    else {
                        /* wait for child to exit */
                        wait(NULL);
                    }
                }
            }
        }
    }
    return 0;
}

void executeOne( int argc, char *argv[] ) {
    char *str_tmp = (char*)malloc(MAX_STR);
    int status;

    if (fork() == 0) {
        if (execvp(argv[0], argv) == -1) {
            printf( "-myshell: %s: command not found\n", argv[0]);
            exit(1);
        }
    }
    else {
        wait(&status);
    }
    free(str_tmp);
}

static void sig_child( int signo )
{
    pid_t pid; int status;

    while ( (pid = waitpid(-1, &status, WNOHANG)) > 0)
        printf( "[background job finished : %d]\n", pid);
}

int parsingPipes( char *str, char **argv )
{
    int argc = 1;
    argv[0] = strtok(str, "|");
    while ( (argv[argc] = strtok(NULL, "|")) != NULL){
        argc++;
    }
    return argc;
}

```

<MyShell – Simple File System>

프로그램의 기본 틀은 인터넷 검색을 통해 획득하였고, 구현한 함수에 해당하는 부분만 첨부하였습니다.

```
int main()
{
    char buf[256];
    int argc;
    char* argv[MAX_ARGC];

    printf("OS SFS shell\n");

    while(!feof(stdin))
    {
        printf("os_shell> ");

        fgets( buf, sizeof(buf), stdin);

        argv[0] = strtok(buf, DELIMS);
        if( !argv[0] )
            continue;

        argc = 1;
        while((argv[argc] = strtok(NULL, DELIMS)) != NULL) {
            argc++;
        }

        if( !strcmp(argv[0], "mount") )
        {
            if(      argc != 2 )
            {
                printf("usage: mount disk_img\n");
                continue;
            }

            sfs_mount(argv[1]);
            continue;
        }

        if( !strcmp(argv[0], "umount") )
        {
            sfs_umount();
            continue;
        }

        if( !strcmp(argv[0], "ls") )
        {
            if( argc == 1 )
                sfs_ls(NULL);
            else if( argc == 2 )
                sfs_ls(argv[1]);
            else
            {
                printf("usage: ls [path]\n");
            }
            continue;
        }

        if( !strcmp(argv[0], "cd") )
```

```

{
    if( argc == 1 )
        sfs_cd(NULL);
    else if( argc != 2 )
    {
        printf("usage: cd [path]\n");
        continue;
    }

    sfs_cd(argv[1]);
    continue;
}

if( !strcmp(argv[0], "dump") )
{
    sfs_dump();
    continue;
}

if( !strcmp(argv[0], "touch") )
{
    if( argc != 2 )
    {
        printf("usage: touch path\n");
        continue;
    }

    sfs_touch(argv[1]);
    continue;
}

if( !strcmp(argv[0], "mkdir") )
{
    if( argc != 2 )
    {
        printf("usage: mkdir directory\n");
        continue;
    }

    sfs_mkdir(argv[1]);
    continue;
}

if( !strcmp(argv[0], "rmdir") )
{
    if( argc != 2 )
    {
        printf("usage: rmdir directory\n");
        continue;
    }

    sfs_rmdir(argv[1]);
    continue;
}

if( !strcmp(argv[0], "rm") )
{
    if( argc != 2 )
    {
        printf("usage: rm path\n");
        continue;
    }
}

```

```

    }

    sfs_rm(argv[1]);
    continue;
}

if( !strcmp(argv[0], "mv") )
{
    if( argc != 3 )
    {
        printf("usage: mv src dst\n");
        continue;
    }

    sfs_mv(argv[1], argv[2]);
    continue;
}

if( !strcmp(argv[0], "cpin") )
{
    if( argc != 3 )
    {
        printf("usage: copyin local-file file(source)\n");
        continue;
    }

    sfs_cpin(argv[1], argv[2]);
    continue;
}

if( !strcmp(argv[0], "cpout") )
{
    if( argc != 3 )
    {
        printf("usage: copyout local-file(source) file\n");
        continue;
    }

    sfs_cpout(argv[1], argv[2]);
    continue;
}

if( !strcmp(argv[0], "exit") )
{
    printf("bye\n");
    return 0;
}

if( !strcmp(argv[0], "fsck") )
{
    sfs_fsck();
    continue;
}

if( !strcmp(argv[0], "bitmap") )
{
    sfs_bitmap();
    continue;
}
printf("%s command not found\n", argv[0]);
}

```

```

    return 0;
}

...

void sfs_touch(const char* path)
{
    struct sfs_inode si;
    disk_read(&si, sd_cwd.sfd_ino);

    //for consistency
    assert(si.sfi_type == SFS_TYPE_DIR);

    int i, idx, idx2, idx3;
    u_int32_t more_direct = 0;
    struct sfs_dir sd[SFS_DENTRYPERBLOCK];

    for (i = 0; i < SFS_NDIRECT; i++) {
        if (si.sfi_direct[i] == SFS_SB_LOCATION) continue;

        disk_read(sd, si.sfi_direct[i]);

        for (idx = 0; idx < SFS_DENTRYPERBLOCK; idx++) {
            if (sd[idx].sfd_ino == SFS_NOINO) continue;

            struct sfs_inode i_temp;
            disk_read(&i_temp, sd[idx].sfd_ino);

            if (!strcmp(path, sd[idx].sfd_name)) {
                error_message("touch", path, -6);
                return;
            }
        }
    }
}

```

```
}
```

```
int bit, bit2;  
int found = 0;  
u_int32_t block, block2;  
u_int32_t free_ino;  
u_int32_t bitmap[128], bitmap2[128];  
  
for (block = SFS_MAP_LOCATION; block < SFS_MAP_LOCATION + SFS_BITBLOCKS(spb.sp_nblocks); block++)  
{  
    disk_read(bitmap, block);  
    for (idx = 0; idx < 128; idx++) {  
        for (bit = 0; bit < 32; bit++) { //BIT_CHECK(a,b) ((a) & (1<<(b)))  
            if (BIT_CHECK(bitmap[idx], bit) == 0) {  
                //이 때의! (왼쪽에서부터의 자리) 수가 비어있는 블록 번호!  
                free_ino = ((block - SFS_MAP_LOCATION)*SFS_BLOCKBITS) + (idx * 32)  
+ bit;  
                found = 1;  
                break;  
            }  
        }  
        if (found) break;  
    }  
    if (found) break;  
}  
  
if (free_ino == 0)  
{  
    error_message("touch", path, -4);  
    return;  
}
```

```

found = 0;

for (i = 0; i < SFS_NDIRECT; i++) {
    if (si.sfi_direct[i] == SFS_SB_LOCATION) continue;
    disk_read(sd, si.sfi_direct[i]);

    for (idx2 = 0; idx2 < SFS_DENTRYPERBLOCK; idx2++) {
        if (sd[idx2].sfd_ino == SFS_NOINO) {
            found = 1;
            break;
        }
    }
    if (found) break;
}

if (!found) {
    int j;
    for (j = 0; j < SFS_NDIRECT; j++) {
        if (si.sfi_direct[j] == SFS_SB_LOCATION) {
            more_direct = j;
            break;
        }
    }
    if (!more_direct) {
        error_message("touch", path, -3);
        return;
    }
}

```

```

if (more_direct) {

    found = 0;

    //direct block 하나 할당해 줘야 함! 비트부터 찾읍시다.

    for (block2 = block; block2 < SFS_MAP_LOCATION + SFS_BITBLOCKS(spb.sp_nbblocks); block2++)

    {

        disk_read(bitmap2, block2);

        for (idx3 = idx; idx3 < 128; idx3++) {

            for (bit2 = bit + 1; bit2 < 32; bit2++) { //BIT_CHECK(a,b) ((a) & (1<<(b)))

                if (BIT_CHECK(bitmap[idx3], bit2) == 0) {

                    //이 때의! (왼쪽에서부터의 자리) 수가 비어있는 블록 번호!

                    si.sfi_direct[more_direct] = free_ino;

                    free_ino = ((block2 - SFS_MAP_LOCATION)*SFS_BLOCKBITS) +

(idx3 * 32) + bit2;

                    found = 1;

                    break;

                }

            }

            if (found) break;

        }

        if (found) break;

    }

    if (!found) {

        error_message("touch", path, -3);

        return;

    }

}

//allocate new block

int newbie_ino = free_ino;

if (more_direct) {

```

```

BIT_SET(bitmap2[idx3], bit2);

disk_write(bitmap2, block2);

disk_read(sd, si.sfi_direct[more_direct]);

int k;

for (k = 0; k < SFS_DENTRYPERBLOCK; k++) {

    sd[k].sfd_ino = SFS_NOINO;

    strncpy(sd[k].sfd_name, "", SFS_NAMELEN);

}

sd[0].sfd_ino = newbie_ino;

strncpy(sd[0].sfd_name, path, SFS_NAMELEN);

disk_write(sd, si.sfi_direct[more_direct]);

}

else {

    sd[idx2].sfd_ino = newbie_ino;

    strncpy(sd[idx2].sfd_name, path, SFS_NAMELEN);

    disk_write(sd, si.sfi_direct[i]);

}

disk_read(bitmap, block);

BIT_SET(bitmap[idx], bit);

disk_write(bitmap, block);

si.sfi_size += sizeof(struct sfs_dir);

disk_write(&si, sd_cwd.sfd_ino);

struct sfs_inode newbie;

```

```

bzero(&newbie, SFS_BLOCKSIZE); // initailize sfi_direct[] and sfi_indirect

newbie.sfi_size = 0;

newbie.sfi_type = SFS_TYPE_FILE;

disk_write(&newbie, newbie_ino);

}

void sfs_cd(const char* path)

{

struct sfs_inode si;

disk_read(&si, sd_cwd.sfd_ino);

assert(si.sfi_type == SFS_TYPE_DIR);

int i, idx;

int found = 0;

struct sfs_dir sd[SFS_DENTRYPERBLOCK];



if (path == NULL) {

    sd_cwd.sfd_ino = SFS_ROOT_LOCATION;

    sd_cwd.sfd_name[0] = '/';

    sd_cwd.sfd_name[1] = '\0';

}

else {

    for (i = 0; i < SFS_NDIRECT; i++) {

        if (si.sfi_direct[i] == SFS_SB_LOCATION) continue;

        disk_read(sd, si.sfi_direct[i]);



        for (idx = 0; idx < SFS_DENTRYPERBLOCK; idx++) {

```

```

        if (sd[idx].sfid_ino == SFS_NOINO) continue;

        struct sfs_inode temp;

        disk_read(&temp, sd[idx].sfid_ino);

        if (!strcmp(path, sd[idx].sfid_name)) {

            if (temp.sfi_type == SFS_TYPE_DIR) {

                sd_cwd.sfid_ino = sd[idx].sfid_ino;
                strcpy(sd_cwd.sfid_name, sd[idx].sfid_name);
                found = 1;
                break;
            }

            else {

                error_message("cd", path, -2);
                return;
            }
        }

        if (found) break;
    }

    if (!found) {

        error_message("cd", path, -1);
        return;
    }
}

```

```
void sfs_ls(const char* path)
```

```
{
```

```
    struct sfs_inode si;
```

```

disk_read(&si, sd_cwd.sfd_ino);

assert(si.sfi_type == SFS_TYPE_DIR);

int i, idx;

int found = 0;

struct sfs_dir sd[SFS_DENTRYPERBLOCK];

if (path != NULL) {

    for (i = 0; i < SFS_NDIRECT; i++) {

        if (si.sfi_direct[i] == SFS_SB_LOCATION) continue;

        disk_read(sd, si.sfi_direct[i]);

        for (idx = 0; idx < SFS_DENTRYPERBLOCK; idx++) {

            if (sd[idx].sfd_ino == SFS_NOINO) continue;

            struct sfs_inode temp;

            disk_read(&temp, sd[idx].sfd_ino);

            if (!strcmp(path, sd[idx].sfd_name)) {

                if (temp.sfi_type == SFS_TYPE_DIR) {

                    disk_read(&si, sd[idx].sfd_ino);

                    found = 1;

                    break;
                }
            }
        }
    }
}

printf("%s\n", path);

return;

```

```

        }

    }

    if (found) break;

}

if (!found) {

    error_message("ls", path, -1);

    return;

}

for (i = 0; i < SFS_NDIRECT; i++) {

    if (si.sfi_direct[i] == SFS_SB_LOCATION) continue;

    disk_read(sd, si.sfi_direct[i]);

    for (idx = 0; idx < SFS_DENTRYPERBLOCK; idx++) {

        if (sd[idx].sfid_ino == SFS_NOINO) continue;

        struct sfs_inode temp;

        disk_read(&temp, sd[idx].sfid_ino);

        printf("%s", sd[idx].sfid_name);

        if (temp.sfi_type == SFS_TYPE_DIR) printf("/");

        printf("%t");

    }

    printf("\n");
}

void sfs_mkdir(const char* org_path)

```

```

{

struct sfs_inode si;

disk_read(&si, sd_cwd.sfd_ino);

//for consistency

assert(si.sfi_type == SFS_TYPE_DIR);

int i, idx;

u_int32_t more_direct = 0;

struct sfs_dir sd[SFS_DENTRYPERBLOCK];

for (i = 0; i < SFS_NDIRECT; i++) {

    if (si.sfi_direct[i] == SFS_SB_LOCATION) continue;

    disk_read(sd, si.sfi_direct[i]);

    for (idx = 0; idx < SFS_DENTRYPERBLOCK; idx++) {

        if (sd[idx].sfd_ino == SFS_NOINO) continue;

        struct sfs_inode i_temp;

        disk_read(&i_temp, sd[idx].sfd_ino);

        if (!strcmp(org_path, sd[idx].sfd_name)) {

            error_message("mkdir", org_path, -6);

            return;

        }

    }

}

int bit, bit1, bit2, bit3;

int found = 0;

```

```

u_int32_t block, block1, block2, block3;

u_int32_t bitidx1, bitidx2, bitidx3;

u_int32_t free_ino1, free_ino2, free_ino3;

u_int32_t bitmap[128];

for (block = SFS_MAP_LOCATION; block < SFS_MAP_LOCATION + SFS_BITBLOCKS(spb.sp_nblocks); block++) {
    disk_read(bitmap, block);

    for (idx = 0; idx < 128; idx++) {
        for (bit = 0; bit < 32; bit++) {
            if (BIT_CHECK(bitmap[idx], bit) == 0) {
                if (found == 0) {
                    free_ino1 = ((block - SFS_MAP_LOCATION)*SFS_BLOCKBITS) +
                    (idx * 32) + bit;
                    block1 = block; bitidx1 = idx; bit1 = bit;
                    found++;
                }
            } else if (found == 1) {
                free_ino2 = ((block - SFS_MAP_LOCATION)*SFS_BLOCKBITS) +
                (idx * 32) + bit;
                block2 = block; bitidx2 = idx; bit2 = bit;
                found++;
            }
            else if (found == 2) {
                free_ino3 = ((block - SFS_MAP_LOCATION)*SFS_BLOCKBITS) +
                (idx * 32) + bit;
                block3 = block; bitidx3 = idx; bit3 = bit;
                found++;
            }
        }
        if (found == 3) break;
    }
}

```

```

    }

    if (found == 3) break;

}

if (found < 2)

{
    error_message("mkdir", org_path, -4);

    return;
}

int idx2;

found = 0;

for (i = 0; i < SFS_NDIRECT; i++) {

    if (si.sfi_direct[i] == SFS_SB_LOCATION) continue;

    disk_read(sd, si.sfi_direct[i]);

    for (idx2 = 0; idx2 < SFS_DENTRYPERBLOCK; idx2++) {

        if (sd[idx2].sfd_ino == SFS_NOINO) {

            found = 1;

            break;
        }
    }

    if (found) break;
}

```

//다이렉트가 더 필요한지 확인해야함!

```

if (!found) {

    int j;

```

```

for (j = 0; j < SFS_NDIRECT; j++) {
    if (si.sfi_direct[j] == SFS_SB_LOCATION) {
        more_direct = j;
        break;
    }
}

if (!more_direct) {
    error_message("mkdir", org_path, -3);
    return;
}

else if (!free_ino3) {
    error_message("mkdir", org_path, -4);
    return;
}
}

```

```

disk_read(bitmap, block1);
BIT_SET(bitmap[bitidx1], bit1);
disk_write(bitmap, block1);

disk_read(bitmap, block2);
BIT_SET(bitmap[bitidx2], bit2);
disk_write(bitmap, block2);

```

```

//allocate new block
int newbie_ino1 = free_ino1;
int newbie_ino2 = free_ino2;

if (more_direct && free_ino3) {
    disk_read(bitmap, block3);
}

```

```

BIT_SET(bitmap[bitidx3], bit3);

disk_write(bitmap, block3);

newbie_ino1 = free_ino2;

newbie_ino2 = free_ino3;

si.sfi_direct[more_direct] = free_ino1;

disk_read(sd, si.sfi_direct[more_direct]);

int k;

for (k = 0; k < SFS_DENTRYPERBLOCK; k++) {

    sd[k].sfid_ino = SFS_NOINO;

    strncpy(sd[k].sfid_name, "", SFS_NAMELEN);

}

sd[0].sfid_ino = newbie_ino1;

strncpy(sd[0].sfid_name, org_path, SFS_NAMELEN);

disk_write(sd, si.sfi_direct[more_direct]);

}

else {

    sd[idx2].sfid_ino = newbie_ino1;

    strncpy(sd[idx2].sfid_name, org_path, SFS_NAMELEN);

    disk_write(sd, si.sfi_direct[i]);

}

si.sfi_size += sizeof(struct sfs_dir);

disk_write(&si, sd_cwd.sfid_ino);

```

```

struct sfs_inode newbie;

struct sfs_dir newbie_block[SFS_DENTRYPERBLOCK];

bzero(&newbie, SFS_BLOCKSIZE); // initalize sfi_direct[] and sfi_indirect
newbie.sfi_size = 0;
newbie.sfi_type = SFS_TYPE_DIR;

newbie.sfi_direct[0] = newbie_ino2;

disk_read(newbie_block, newbie.sfi_direct[0]);

for (i = 0; i < SFS_DENTRYPERBLOCK; i++) {
    newbie_block[i].sfd_ino = SFS_NOINO;
    strncpy(newbie_block[i].sfd_name, "", SFS_NAMELEN);
}

newbie_block[0].sfd_ino = newbie_ino1;
strncpy(newbie_block[0].sfd_name, ".", SFS_NAMELEN);
newbie.sfi_size += sizeof(struct sfs_dir);

newbie_block[1].sfd_ino = sd_cwd.sfd_ino;
strncpy(newbie_block[1].sfd_name, "..", SFS_NAMELEN);
newbie.sfi_size += sizeof(struct sfs_dir);

disk_write(newbie_block, newbie.sfi_direct[0]);
disk_write(&newbie, newbie_ino1);

}

void sfs_rmdir(const char* org_path)

```

```

{

struct sfs_inode si;

disk_read(&si, sd_cwd.sfd_ino);

assert(si.sfi_type == SFS_TYPE_DIR);

if (!strcmp(org_path, "."))
{
    error_message("rmdir", org_path, -8);

    return;
}

int i, idx;

int found = 0;

u_int32_t num_direct, block, block2;

struct sfs_dir sd[SFS_DENTRYPERBLOCK], sd2[SFS_DENTRYPERBLOCK];

num_direct = SFS_ROUNDUP(si.sfi_size, SFS_BLOCKSIZE) / SFS_BLOCKSIZE;

for (i = 0; i < SFS_NDIRECT; i++) {
    if (si.sfi_direct[i] == SFS_SB_LOCATION) continue;

    disk_read(sd, si.sfi_direct[i]);
}

for (idx = 0; idx < SFS_DENTRYPERBLOCK; idx++) {
    if (sd[idx].sfd_ino == SFS_NOINO) continue;

    if (!strcmp(org_path, sd[idx].sfd_name)) {
        found = 1;

        break;
    }
}
}
```

```

    if (found == 1) break;

}

if (found == 1) {

    struct sfs_inode temp;

    disk_read(&temp, sd[idx].sfid_ino);

    if (temp.sfi_type != SFS_TYPE_DIR) {

        error_message("rmdir", org_path, -5);

        return;
    }

    if (temp.sfi_size / sizeof(struct sfs_dir) > 2) {

        error_message("rmdir", org_path, -7);

        return;
    }

    block = sd[idx].sfid_ino;

    sd[idx].sfid_ino = SFS_NOINO;

    disk_write(sd, si.sfi_direct[i]);

    si.sfi_size -= sizeof(struct sfs_dir);

    disk_write(&si, sd_cwd.sfid_ino);

    u_int32_t bitmap[128];

    int x;

    for (x = 0; x < SFS_DENTRYPERBLOCK; x++) {

        if (temp.sfi_direct[x] != SFS_NOINO) {

            block2 = temp.sfi_direct[x];

            disk_read(bitmap, (block2 / SFS_BLOCKBITS) + 2);

            BIT_CLEAR(bitmap[(block2 % SFS_BLOCKBITS) / 32], ((block2 % SFS_BLOCKBITS) %
32));
        }
    }
}

```

```

        disk_write(bitmap, (block2 / SFS_BLOCKBITS) + 2);

    }

}

disk_read(bitmap, (block / SFS_BLOCKBITS) + 2);

BIT_CLEAR(bitmap[(block%SFS_BLOCKBITS) / 32], ((block % SFS_BLOCKBITS) % 32));

disk_write(bitmap, (block / SFS_BLOCKBITS) + 2);

}

else if (found == 0) {

    error_message("rmdir", org_path, -1);

    return;

}

void sfs_mv(const char* src_name, const char* dst_name)

{

    struct sfs_inode si;

    disk_read(&si, sd_cwd.sfd_ino);

    assert(si.sfi_type == SFS_TYPE_DIR);

    if (!strcmp(src_name, ".") || !strcmp(src_name, "..")) {

        error_message("mv", src_name, -8);

        return;

    }

    else if (!strcmp(dst_name, ".") || !strcmp(dst_name, "..")) {

        error_message("mv", dst_name, -8);

        return;

    }

    int i, idx, idx2;

```

```

int found = 0;

u_int32_t block;

struct sfs_dir sd[SFS_DENTRYPERBLOCK], sd2[SFS_DENTRYPERBLOCK];

for (i = 0; i < SFS_NDIRECT; i++) {

    if (si.sfi_direct[i] == SFS_SB_LOCATION) continue;

    disk_read(sd, si.sfi_direct[i]);


    for (idx = 0; idx < SFS_DENTRYPERBLOCK; idx++) {

        if (sd[idx].sfid_ino == SFS_NOINO) continue;

        if (!strcmp(src_name, sd[idx].sfid_name)) {

            disk_read(sd2, si.sfi_direct[i]);

            idx2 = idx;

            block = si.sfi_direct[i];

            found = 1;

        }

        if (!strcmp(dst_name, sd[idx].sfid_name)) {

            error_message("mv", dst_name, -6);

            return;

        }

    }

}

if (found == 1) {

    strcpy(sd2[idx2].sfid_name, dst_name);

    disk_write(sd2, block);

}

else if (found == 0) {

```

```

        error_message("mv", src_name, -1);

        return;
    }

}

void sfs_rm(const char* path)
{
    struct sfs_inode si;

    disk_read(&si, sd_cwd.sfd_ino);

    assert(si.sfi_type == SFS_TYPE_DIR);

    int i, idx, dir_idx, ind_idx;

    int found = 0;

    u_int32_t block, num_block;

    u_int32_t bitmap[SFS_BLOCKSIZE / sizeof(u_int32_t)];

    struct sfs_dir sd[SFS_DENTRYPERBLOCK], sd2[SFS_DENTRYPERBLOCK];

    for (i = 0; i < SFS_NDIRECT; i++) {

        if (si.sfi_direct[i] == SFS_SB_LOCATION) continue;

        disk_read(sd, si.sfi_direct[i]);

        for (idx = 0; idx < SFS_DENTRYPERBLOCK; idx++) {

            if (sd[idx].sfd_ino == SFS_NOINO) continue;

            if (!strcmp(path, sd[idx].sfd_name)) {

                found = 1;

                break;
            }
        }
    }
}

```

```

    if (found) break;
}

if (found) {
    struct sfs_inode temp;      //지울 파일 inode

    disk_read(&temp, sd[idx].sfid_ino);

    if (temp.sfi_type == SFS_TYPE_DIR) {
        error_message("rm", path, -9);
        return;
    }

    else if (temp.sfi_type != SFS_TYPE_FILE) {
        error_message("rm", path, -10);
        return;
    }

    block = sd[idx].sfid_ino;           //지울 놈 노드번호
    //temp가 그 노드다. 이제
    루프를 돌면서 direct부터 해제해 보자.

    for (dir_idx = 0; dir_idx < SFS_NDIRECT; dir_idx++) {
        num_block = temp.sfi_direct[dir_idx];

        if (num_block == SFS_NOINO) continue;

        disk_read(bitmap, (num_block / SFS_BLOCKBITS) + 2);
        BIT_CLEAR(bitmap[(num_block % SFS_BLOCKBITS) / 32], ((num_block % SFS_BLOCKBITS) %
32));
        disk_write(bitmap, (num_block / SFS_BLOCKBITS) + 2);
    }
}

```

```
}
```

```
//indirect 해제 해 보자
```

```
if (temp.sfi_indirect != SFS_NOINO) {
```

```
    u_int32_t indirect[SFS_DBPERIDB];
```

```
    u_int32_t block_num = temp.sfi_indirect;
```

```
    disk_read(indirect, temp.sfi_indirect);
```

```
    for (ind_idx = 0; ind_idx < SFS_DBPERIDB; ind_idx++) {
```

```
        num_block = indirect[ind_idx];
```

```
        if (num_block == SFS_NOINO) continue;
```

```
        disk_read(bitmap, (num_block / SFS_BLOCKBITS) + 2);
```

```
        BIT_CLEAR(bitmap[(num_block % SFS_BLOCKBITS) / 32], ((num_block % SFS_BLOCKBITS) % 32));
```

```
        disk_write(bitmap, (num_block / SFS_BLOCKBITS) + 2);
```

```
}
```

```
    disk_read(bitmap, (block_num / SFS_BLOCKBITS) + 2);
```

```
    BIT_CLEAR(bitmap[(block_num % SFS_BLOCKBITS) / 32], ((block_num % SFS_BLOCKBITS) % 32));
```

```
    disk_write(bitmap, (block_num / SFS_BLOCKBITS) + 2);
```

```
}
```

```
    disk_write(&temp, sd[idx].sfd_ino);
```

```
//현재 디렉토리에서 얘에 해당하는 엔트리 숫자만 초기화.
```

```
    sd[idx].sfd_ino = SFS_NOINO;
```



```

disk_read(sd, si.sfi_direct[i]);

for (idx = 0; idx < SFS_DENTRYPERBLOCK; idx++) {
    if (sd[idx].sfid_ino == SFS_NOINO) continue;
    struct sfs_inode temp;
    disk_read(&temp, sd[idx].sfid_ino);

    if (!strcmp(local_path, sd[idx].sfid_name)) {
        error_message("cpin", local_path, -6);
        return;
    }
}

FILE *fp;
char buffer[SFS_BLOCKSIZE];
u_int32_t fsize;

if ((fp = fopen(path, "rb")) == NULL) {
    error_message("cpin", path, -11);
    return;
}

fseek(fp, 0, SEEK_END); // 파일포인터를 파일의 끝으로 이동
fsize = ftell(fp); // 현재 파일포인터를 읽으면 파일크기가 됨

if (fsize > (SFS_BLOCKSIZE*(SFS_NDIRECT + SFS_DBPERIDB))) {
    error_message("cpin", NULL, -12);
    return;
}

```

```

fseek(fp, 0, SEEK_SET);

int bit, bit1, bit2, bit3;

int found = 0;

u_int32_t block, block1, block2, block3;

u_int32_t bitidx1, bitidx2, bitidx3;

u_int32_t free_ino1, free_ino2, free_ino3;

u_int32_t bitmap[SFS_BLOCKSIZE / sizeof(u_int32_t)];

for (block = SFS_MAP_LOCATION; block < SFS_MAP_LOCATION + SFS_BITBLOCKS(spb.sp_nblocks); block++) {
    disk_read(bitmap, block);

    for (idx = 0; idx < 128; idx++) {
        for (bit = 0; bit < 32; bit++) {
            if (BIT_CHECK(bitmap[idx], bit) == 0) {
                if (found == 0) {
                    free_ino1 = ((block - SFS_MAP_LOCATION)*SFS_BLOCKBITS) +
                    (idx * 32) + bit;

                    block1 = block; bitidx1 = idx; bit1 = bit;
                    found++;
                }
                else if (found == 1) {
                    free_ino2 = ((block - SFS_MAP_LOCATION)*SFS_BLOCKBITS) +
                    (idx * 32) + bit;

                    block2 = block; bitidx2 = idx; bit2 = bit;
                    found++;
                }
                else if (found == 2) {
                    free_ino3 = ((block - SFS_MAP_LOCATION)*SFS_BLOCKBITS) +
                    (idx * 32) + bit;

                    block3 = block; bitidx3 = idx; bit3 = bit;
                    found++;
                }
            }
        }
    }
}

```

```

        }

    }

    if (found == 3) break;

}

if (found == 3) break;

}

if (found < 2)

{

    error_message("cpin", local_path, -13);

    return;

}

//여기까지 넘어올 수 있었다면, 최소한 한 개는 읽어들일 수 있단 소리. 맞지?

found = 0;

for (i = 0; i < SFS_NDIRECT; i++) {

    if (si.sfi_direct[i] == SFS_SB_LOCATION) continue;

    disk_read(sd, si.sfi_direct[i]);



    for (idx2 = 0; idx2 < SFS_DENTRYPERBLOCK; idx2++) {

        if (sd[idx2].sfd_ino == SFS_NOINO) {

            found = 1;

            break;

        }

    }

    if (found) break;

}

```

```

if (!found) {

    int j;

    for (j = 0; j < SFS_NDIRECT; j++) {

        if (si.sfi_direct[j] == SFS_SB_LOCATION) {

            more_direct = j;

            break;

        }

    }

    if (!more_direct) {

        error_message("cpin", local_path, -3);

        return;

    }

    else if (!free_ino3) {

        error_message("cpin", local_path, -4);

        return;

    }

}

```

```

disk_read(bitmap, block1);

BIT_SET(bitmap[bitidx1], bit1);

disk_write(bitmap, block1);

```

```

disk_read(bitmap, block2);

BIT_SET(bitmap[bitidx2], bit2);

disk_write(bitmap, block2);

```

```

//allocate new block

int newbie_ino1 = free_ino1;

int newbie_ino2 = free_ino2;

```

```

if (more_direct && free_ino3) {

    disk_read(bitmap, block3);

    BIT_SET(bitmap[bitidx3], bit3);

    disk_write(bitmap, block3);


    newbie_ino1 = free_ino2;

    newbie_ino2 = free_ino3;




    si.sfi_direct[more_direct] = free_ino1;

    disk_read(sd, si.sfi_direct[more_direct]);


    int k;

    for (k = 0; k < SFS_DENTRYPERBLOCK; k++) {

        sd[k].sfd_ino = SFS_NOINO;

        strncpy(sd[k].sfd_name, "", SFS_NAMELEN);

    }






    sd[0].sfd_ino = newbie_ino1;

    strncpy(sd[0].sfd_name, local_path, SFS_NAMELEN);




    disk_write(sd, si.sfi_direct[more_direct]);


}

else {

    sd[idx2].sfd_ino = newbie_ino1;

    strncpy(sd[idx2].sfd_name, local_path, SFS_NAMELEN);




    disk_write(sd, si.sfi_direct[i]);


}

```

```
si.sfi_size += sizeof(struct sfs_dir);
disk_write(&si, sd_cwd.sfd_ino);

struct sfs_inode newbie;
char newbie_block[SFS_BLOCKSIZE];
u_int32_t readlen;

bzero(&newbie, SFS_BLOCKSIZE); // initilize sfi_direct[] and sfi_indirect
newbie.sfi_size = 0;
newbie.sfi_type = SFS_TYPE_FILE;

newbie.sfi_direct[0] = newbie_ino2; //direct 0번 할당 완료

disk_read(newbie_block, newbie.sfi_direct[0]);

readlen = fread(buffer, 1, SFS_BLOCKSIZE, fp);

int y;

for (y = 0; y < SFS_BLOCKSIZE; y++) newbie_block[y] = buffer[y];

if (readlen < SFS_BLOCKSIZE) {
    if (feof(fp) != 0) {
        newbie.sfi_size += readlen;
        disk_write(newbie_block, newbie.sfi_direct[0]);
        disk_write(&newbie, newbie_ino1);
        return;
    }
    else {
        disk_write(&newbie, newbie_ino1);
    }
}
```

```

    }

    return;
}

newbie.sfi_size += readlen;

disk_write(newbie_block, newbie.sfi_direct[0]);

disk_write(&newbie, newbie_ino1);

int index;

for (index = 1; index < SFS_NDIRECT; index++) {

    found = 0;

    for (block = SFS_MAP_LOCATION; block < SFS_MAP_LOCATION + SFS_BITBLOCKS(spb.sp_nblocks);
block++) {

        disk_read(bitmap, block);

        for (idx = 0; idx < 128; idx++) {

            for (bit = 0; bit < 32; bit++) {

                if (BIT_CHECK(bitmap[idx], bit) == 0) {

                    if (found == 0) {

                        free_ino1 = ((block -
SFS_MAP_LOCATION)*SFS_BLOCKBITS) + (idx * 32) + bit;

                        block1 = block; bitidx1 = idx; bit1 = bit;

                        found++;

                    }

                    break;
                }
            }
        }

        if (found) break;
    }

    if (found) break;
}

if (found) {

```

```
newbie.sfi_direct[index] = free_ino1;           //direct 번호 할당
```

```
disk_read(newbie_block, newbie.sfi_direct[index]);
```

```
readlen = fread(buffer, 1, SFS_BLOCKSIZE, fp);
```

```
for (y = 0; y < SFS_BLOCKSIZE; y++) newbie_block[y] = buffer[y];
```

```
if (readlen < SFS_BLOCKSIZE) {
```

```
    if (feof(fp) != 0) {
```

```
        newbie.sfi_size += readlen;
```

```
        disk_write(newbie_block, newbie.sfi_direct[index]);
```

```
        disk_write(&newbie, newbie_ino1);
```

```
        disk_read(bitmap, block1);
```

```
        BIT_SET(bitmap[bitidx1], bit1);
```

```
        disk_write(bitmap, block1);
```

```
        fclose(fp);
```

```
    }
```

```
    break;
```

```
}
```

```
newbie.sfi_size += readlen;
```

```
disk_write(newbie_block, newbie.sfi_direct[index]);
```

```
disk_write(&newbie, newbie_ino1);
```

```
disk_read(bitmap, block1);
```

```
BIT_SET(bitmap[bitidx1], bit1);
```

```

        disk_write(bitmap, block1);

    }

    else {
        break;
    }

}

found = 0;

for (block = SFS_MAP_LOCATION; block < SFS_MAP_LOCATION + SFS_BITBLOCKS(spb.sp_nblocks); block++)
{
    disk_read(bitmap, block);

    for (idx = 0; idx < 128; idx++) {
        for (bit = 0; bit < 32; bit++) {
            if (BIT_CHECK(bitmap[idx], bit) == 0) {

                if (found == 0) {

                    free_ino1 = ((block - SFS_MAP_LOCATION)*SFS_BLOCKBITS) +
                    (idx * 32) + bit;

                    block1 = block; bitidx1 = idx; bit1 = bit;
                    found++;
                    break;
                }
            }
        }
        if (found) break;
    }

    if (found) break;
}

if (found) {

    newbie.sfi_indirect = free_ino1;      //indirect 번호 할당

    disk_read(newbie_block, newbie.sfi_indirect);
}

```

```

bzero(newbie_block, SFS_BLOCKSIZE);

disk_write(newbie_block, newbie.sfi_indirect);
disk_write(&newbie, newbie_ino1);

disk_read(bitmap, block1);
BIT_SET(bitmap[bitidx1], bit1);
disk_write(bitmap, block1);

}

else return;

u_int32_t indirect[SFS_DBPERIDB];
disk_read(indirect, newbie.sfi_indirect);

for (index = 0; index < SFS_DBPERIDB; index++) {
    found = 0;
    for (block = SFS_MAP_LOCATION;
        block < SFS_MAP_LOCATION + SFS_BITBLOCKS(spb.sp_nbblocks); block++) {
        disk_read(bitmap, block);
        for (idx = 0; idx < 128; idx++) {
            for (bit = 0; bit < 32; bit++) {
                if (BIT_CHECK(bitmap[idx], bit) == 0) {
                    if (found == 0) {
                        free_ino1 = ((block -
SFS_MAP_LOCATION)*SFS_BLOCKBITS) + (idx * 32) + bit;
                        block1 = block; bitidx1 = idx; bit1 = bit;
                        found++;
                    }
                    break;
                }
            }
        }
    }
}

```

```

    }

    if (found) break;

}

if (found) {

    indirect[index] = free_ino1;

    disk_read(newbie_block, indirect[index]);

    bzero(newbie_block, SFS_BLOCKSIZE);

    readlen = fread(buffer, 1, SFS_BLOCKSIZE, fp);

    for (y = 0; y < SFS_BLOCKSIZE; y++) newbie_block[y] = buffer[y];

    if (readlen < SFS_BLOCKSIZE) {

        if (feof(fp) != 0) {

            newbie.sfi_size += readlen;

            disk_write(newbie_block, indirect[index]);

            disk_write(&newbie, newbie_ino1);

            disk_read(bitmap, block1);

            BIT_SET(bitmap[bitidx1], bit1);

            disk_write(bitmap, block1);

        }

        break;

}

newbie.sfi_size += readlen;

```

```

        disk_write(newbie_block, indirect[index]);

        disk_write(&newbie, newbie_ino1);

        disk_read(bitmap, block1);

        BIT_SET(bitmap[bitidx1], bit1);

        disk_write(bitmap, block1);

    }

    else {

        break;

    }

}

disk_write(indirect, newbie.sfi_indirect);

fclose(fp);

}

void sfs_cpout(const char* local_path, const char* path)
{
    struct sfs_inode si;

    disk_read(&si, sd_cwd.sfd_ino);

    //for consistency

    assert(si.sfi_type == SFS_TYPE_DIR);

    int i, idx, idx2;

    int found = 0;

    u_int32_t block;

    struct sfs_dir sd[SFS_DENTRYPERBLOCK], sd2[SFS_DENTRYPERBLOCK];

```

```
FILE *fp;

char buffer[SFS_BLOCKSIZE];

u_int32_t fsize;

for (i = 0; i < SFS_NDIRECT; i++) {

    if (si.sfi_direct[i] == SFS_SB_LOCATION) continue;

    disk_read(sd, si.sfi_direct[i]);

    for (idx = 0; idx < SFS_DENTRYPERBLOCK; idx++) {

        if (sd[idx].sfid_ino == SFS_NOINO) continue;

        if (!strcmp(local_path, sd[idx].sfid_name)) {

            found = 1;

            break;
        }
    }

    if (found) break;
}

if (found == 0) {

    error_message("cpout", local_path, -1);

    return;
}

if ((fp = fopen(path, "rb")) != NULL) {

    error_message("cpout", path, -6);

    return;
}

fp = fopen(path, "wb");
```

```

// 이게 찾은 놈 노드 번호. -> sd[idx].sfid_ino

//이걸 가지고 사이즈 부터 파악한다.

struct sfs_inode temp;

disk_read(&temp, sd[idx].sfid_ino);

char direct[SFS_BLOCKSIZE];

u_int32_t blocks = SFS_ROUNDUP(temp.sfi_size, SFS_BLOCKSIZE) / SFS_BLOCKSIZE;

if (blocks > SFS_NDIRECT)

{

    for (idx2 = 0; idx2 < SFS_NDIRECT; idx2++) {

        disk_read(direct, temp.sfi_direct[idx2]);

        int x;

        for (x = 0; x < SFS_BLOCKSIZE; x++)

            buffer[x] = direct[x];



        fwrite(buffer, 1, SFS_BLOCKSIZE, fp);

    }

    //이제 인다이렉트.....

    u_int32_t indirect[SFS_DBPERIDB];

    disk_read(indirect, temp.sfi_indirect);

    for (idx2 = 0; idx2 < blocks - SFS_NDIRECT; idx2++) {

        disk_read(direct, indirect[idx2]);

        int x;

        for (x = 0; x < SFS_BLOCKSIZE; x++)

            buffer[x] = direct[x];



        if ((idx2 + 1) == blocks - SFS_NDIRECT) {

```

```

        fwrite(buffer, 1, temp.sfi_size%SFS_BLOCKSIZE, fp);

        break;

    }

    fwrite(buffer, 1, SFS_BLOCKSIZE, fp);

}

}

else {

    for (idx2 = 0; idx2 < blocks; idx2++) {

        disk_read(direct, temp.sfi_direct[idx]);

        int x;

        for (x = 0; x < SFS_BLOCKSIZE; x++)

            buffer[x] = direct[x];



        if ((idx2 + 1) == blocks) {

            fwrite(buffer, 1, temp.sfi_size%SFS_BLOCKSIZE, fp);

            break;

        }

        fwrite(buffer, 1, SFS_BLOCKSIZE, fp);

    }

}

fclose(fp);

}

```



온라인 투표 시스템

Detailed Design & Implementation



김상준
이한솔
심민우
강석원

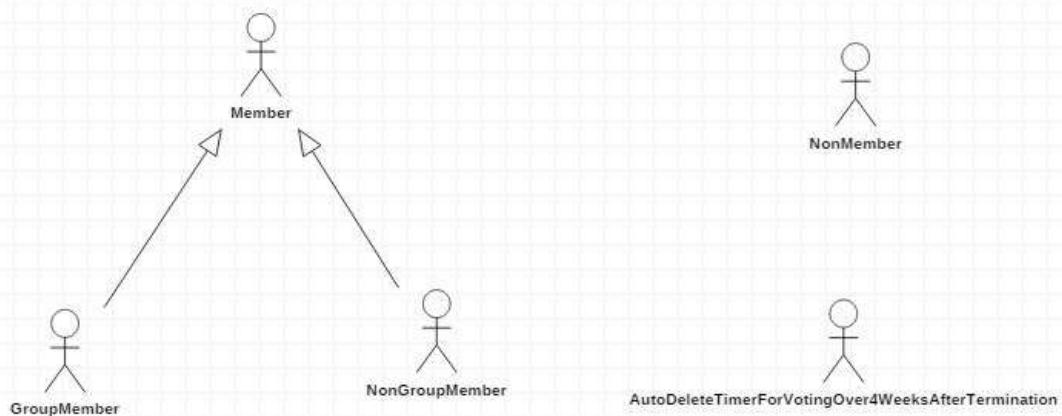
요구사항 목록(Requirement List)

No.	Requirement	Use Case(s)
1	기본적인 정보(이름, 주민번호, 주소, ID, PW)를 입력 받아 사용자에게 시스템을 이용할 수 있는 권한을 부여한다. 단, 회원의 ID는 중복될 수 없다.	회원가입
2	현재 생성되어 있는 전체 그룹 리스트를 보여준다. 그 중 원하는 하나의 그룹에만 가입할 수 있다. 단, 이미 그룹에 가입되어 있는 경우 현재 가입되어 있는 그룹을 탈퇴 후 그룹에 가입할 수 있다.	전체 그룹 조회 및 그룹 가입
3	사용자가 시스템에서 탈퇴할 수 있다. 탈퇴와 동시에 사용자의 시스템의 사용 권한은 소멸한다. 단, 그룹을 생성한 사용자는 탈퇴할 수 없다.	회원탈퇴
4	ID, PW를 입력 받아 사용자에 따른 권한을 부여한다.	로그인
5	회원이 로그아웃 한다.	로그아웃
6	그룹 이름을 입력 받아 그룹을 생성한다. 그룹이 생성되면 그 그룹 생성을 요청한 회원은 그 그룹에 자동으로 가입된다. 단, 그룹 이름은 중복되지 않아야 한다.	그룹 생성
7	사용자가 속한 그룹의 정보를 출력한다. 사용자의 선택에 따라 그룹에서 탈퇴할 수 있다.	가입 그룹 조회
8	그룹원이 그룹을 탈퇴 할 수 있다. 사용자의 그룹과 관련된 모든 투표 권한 및 정보 접근에 대한 권한을 박탈한다. 단, 그룹 탈퇴는 가입 그룹 조회를 한 직후에 가능하며, 그룹을 생성한 사용자는 그 그룹에서 탈퇴 할 수 없다	그룹 탈퇴
9	그룹 가입자는 본인이 속한 그룹 내에서 투표를 제안할 수 있다. 이 때 투표 주제, 투표 항목, 투표 시작 시간 및 투표 마감 시간에 대하여 기록한다. 단, 같은 그룹 내에서 같은 이름의 투표주제로 투표를 제안할 수 없다.	투표 제안
10	해당 그룹 내의 현재 진행중인 투표를 조회 할 때, 투표 주제, 투표 항목, 남은 시간을 출력한다. 투표자의 선택에 따라 투표권을 행사 할 수 있다.	현재 진행중인 투표 조회
11	해당 그룹 내의 향후 진행 예정인 투표를 조회할 때, 투표 리스트, 투표 주제, 투표 항목, 투표 시작시간 및 마감시간을 출력한다.	향후 진행 예정인 투표 조회
12	해당 그룹 내의 종료된 투표 리스트를 조회 할 때, 투표 주제, 투표 항목, 투표 결과를 출력한다.	종료된 투표 조회
13	투표 주제, 항목 번호를 입력하여 투표권을 행사한다. 단, 투표는 현재 진행중인 투표 조회 직후에 가능하며, 이미 투표한	투표

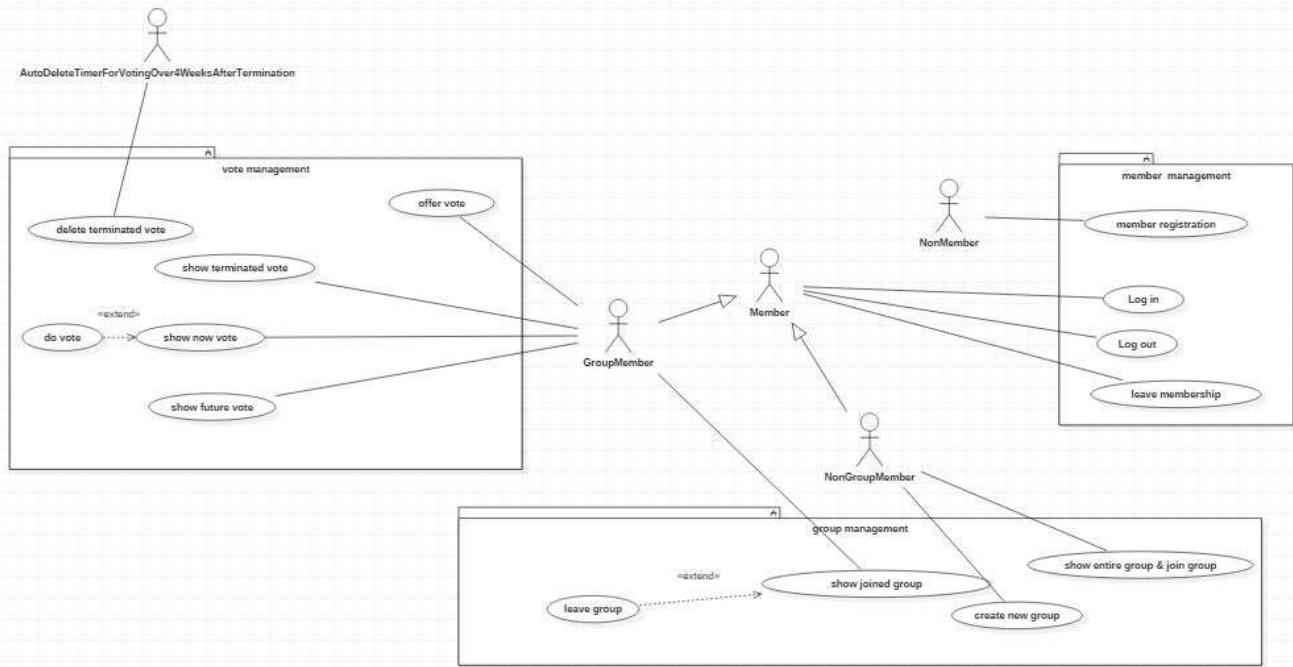
	투표 주제에는 다시 투표할 수 없다.	
14	종료 후 4주가 지난 투표를 자동으로 삭제한다.	종료된 투표 삭제

사용자 설명(Actor Description)

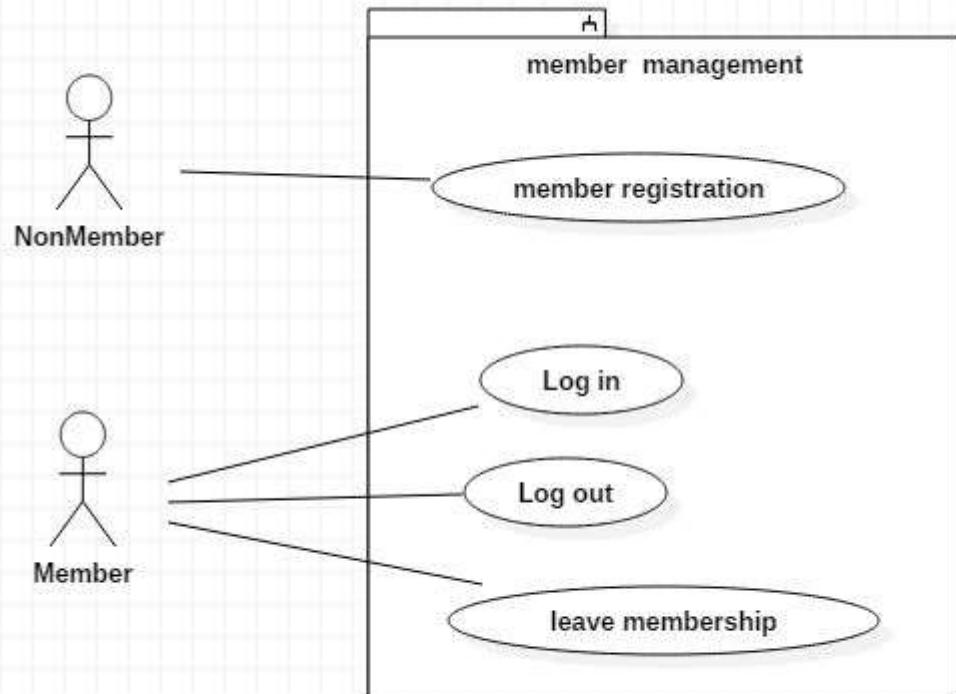
Actor	Description
비회원 (non-member)	온라인 투표 시스템에 가입되지 않은 사용자
회원 (member)	온라인 투표 시스템에 가입된 모든 사용자
그룹 가입자 (group member)	특정 그룹에 속한 회원
그룹 미 가입자 (non-group member)	어떤 그룹에도 속하지 않은 회원
종료 후 4주 이상된 투표 삭제 타이머 (Auto-delete timer for voting more than 4 weeks after termination)	종료 후 4주가 지난 투표 정보를 삭제하는 타이머 시스템



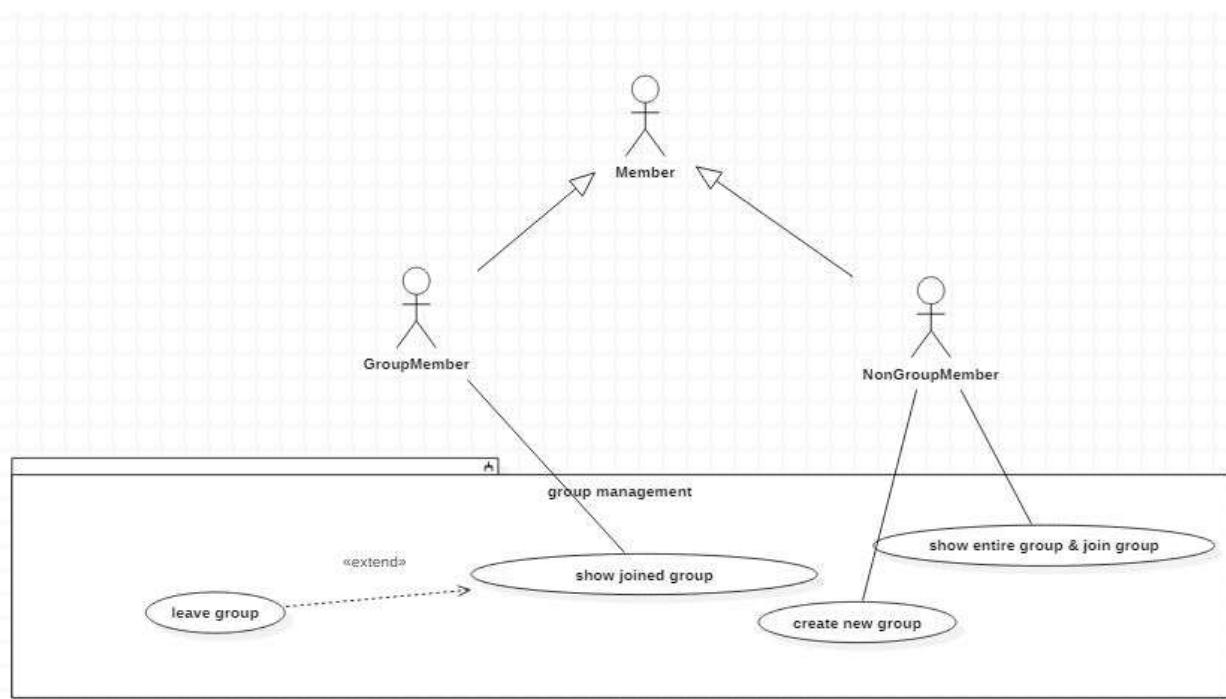
용례 도표(Use Case Diagram)



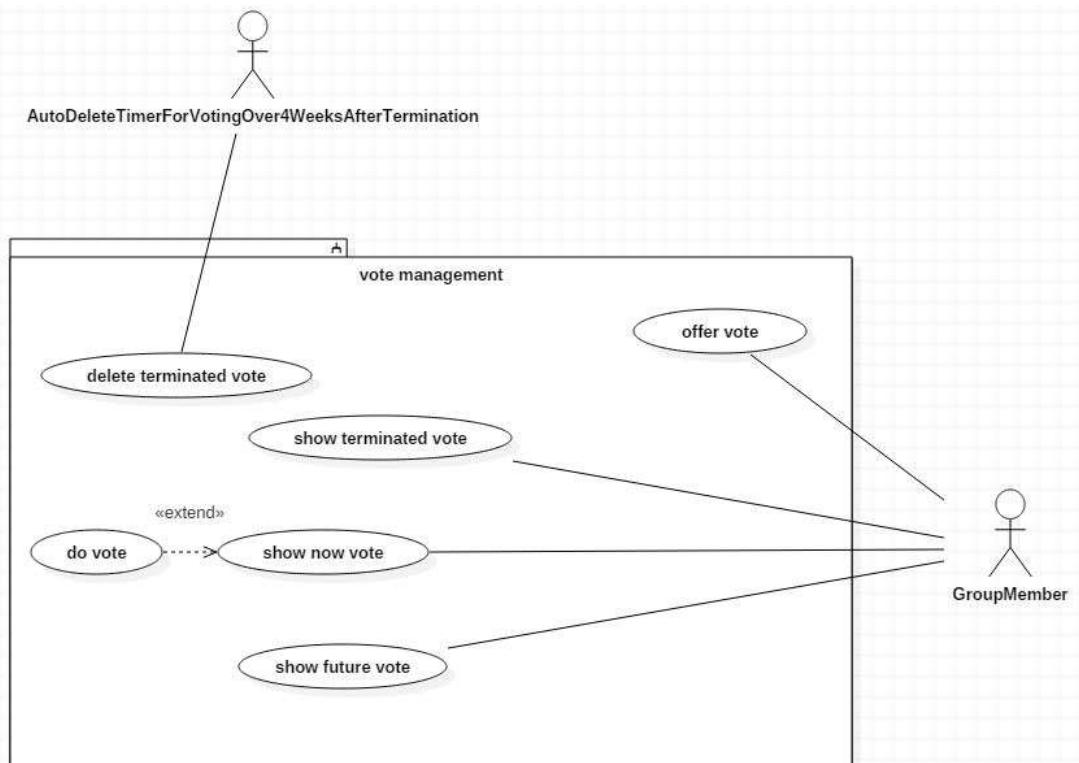
회원관리



그룹관리



투표관리



용례 설명(Use case Description)

회원 관리

Use Case	Description	
	Actor Action	System Response
회원 가입 (Member Registration)	1. 개인정보를 입력한다. 이름, 주소, 주민번호, 이메일 등을 입력하고 ID, 패스워드를 설정한다.	2. 입력한 정보에 따라 처리한다. 2.1 유효한 정보일 경우 회원가입이 완료 되었다고 출력한다 2.2 유효하지 않은 정보일 경우 정보가 유효하지 않다고 출력한다.
로그인 (Login)	1. ID, 패스워드를 입력한다	2. 입력한 정보에 따라 출력한다 2.1 유효한 정보인 경우 로그인 완료를 출력한다 2.2 유효하지 않은 정보인 경우 정보가 유효하지 않다고 출력한다.
로그아웃 (Logout)	1. None	2. 로그아웃이 완료되었다고 출력한다.
회원 탈퇴 (Leave Membership)	1. None	2. 응답에 대한 알림 2.1. 회원이 그룹장이 아닌 경우 회원탈퇴가 완료되었다고 출력한다. 2.2. 회원이 그룹장인 경우 회원탈퇴가 불가능하다고 출력한다.

그룹 관리

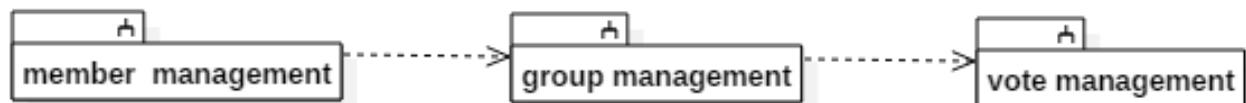
Use Case	Description	
	Actor Action	System Response
전체 그룹 조회 및 그룹 가입 (Show Entire Group & Join Group)	1. None 3.1 그룹가입을 진행하지 않는다. 3.2 선택할 그룹이 있을 시 가입할 그룹을 선택한다	2. 전체 그룹 리스트를 출력한다. 4.1 None 4.2 응답에 대한 알림 4.2.1 사용자가 가입한 그룹이 아직 없는 경우, 그룹에 성공적으로 가입되었다고 출력한다. 4.2.2 사용자가 가입한 그룹이 있는 경우, 하나의 그룹에만 가입할 수 있다고 출력한다 4.2.3 입력한 그룹이 존재하지 않을 경우, 그룹이 존재하지 않는다고 출력한다
그룹 생성 (Create Group)	1. 그룹의 이름을 입력한다	2. 그룹 생성에 대해 출력한다 2.1 사용자가 이미 가입한 그룹이 없는 경우, 그룹 생성이 완료되었다고 출력한다 2.2 사용자가 이미 가입한 그룹이 있는 경우, 그룹 생성이 불가능하다고 출력한다. 2.3 중복된 이름이 있는 경우, 중복된 이름의 그룹이 존재한다고 출력한다.
가입 그룹 조회 및 그룹 탈퇴 (Show Joined Group & Leave Group)	1. None 3.1 그룹 탈퇴를 진행하지 않는다 3.2 그룹 탈퇴를 진행한다	2. 가입한 그룹에 대한 정보를 출력한다. 2.1 사용자가 가입한 그룹이 있을 경우, 사용자가 가입한 그룹에 대하여 출력한다 2.2 사용자가 가입한 그룹이 없는 경우, 가입한 그룹이 없다고 출력한다. 4. 응답에 대한 알림 4.1 None 4.2.1 사용자가 그룹 생성자가 아닌 경우 그룹 탈퇴가 완료되었다고 그룹 이름과 함께 출력한다. 4.2.2 사용자가 그룹 생성자인 경우 그룹 탈퇴가 불가능하다고 출력한다. 4.2.3 가입한 그룹이 없는 경우 가입한 그룹이 없다고 출력한다.

투표 관리

Use Case	Description	
	Actor Action	System Response
투표 제안 (Offer Vote)	<p>1. 투표주제, 투표항목, 투표 시작시간 및 투표 마감시간을 입력한다.</p>	<p>2. 사용자의 상황에 따라 처리한다</p> <p>2.1 사용자가 그룹에 가입되어 있지 않는 경우, 투표 제안이 불가능 하다고 출력한다.</p> <p>2.2 사용자가 그룹에 가입되어 있는 경우, 투표 제안이 완료되었다고 출력한다.</p> <p>2.3 입력된 정보가 잘못 된 경우, 투표 제안이 잘못되었다고 출력한다</p>
현재 진행 중인 투표 조회 및 투표 (Show Now-voting Vote & Do Vote)	<p>1. None</p> <p>3.1 투표를 선택하지 않는다</p> <p>3.2 투표할 투표와 투표항목을 입력한다</p>	<p>2. 사용자의 상황에 따라 처리한다.</p> <p>2.1 사용자가 그룹에 가입되어 있지 않는 경우, 향후 진행 예정인 투표 조회가 불가능 하다고 출력한다.</p> <p>2.2 사용자가 그룹에 가입되어 있는 경우, 그룹 내의 향후 진행 예정인 투표들의 투표 주제, 투표 항목 수, 시작시간, 마감 시간을 출력한다.</p> <p>4. 사용자의 상황에 처리한다.</p> <p>4.1. None</p> <p>4.2.1. 사용자가 그룹에 가입되어 있지 않는 경우 투표가 불가능 하다고 출력한다.</p> <p>4.2.2. 사용자가 선택한 투표가 존재하지 않거나 현재 진행중인 투표가 아닌 경우 투표가 존재하지 않는다고 출력한다</p> <p>4.2.3. 사용자가 선택한 항목이 존재하지 않는 경우 투표가 불가능하다고 출력한다.</p> <p>4.2.4. 사용자가 이미 그 투표에 대해 투표한 경우 투표가 불가능하다고 출력한다</p> <p>4.2.5. 성공적으로 투표가 된 경우, 투표주제와 투표 항목을 출력한다.</p>
향후 진행 예정인 투표 조회 (Show Future Vote)	<p>1. None</p>	<p>2. 사용자의 상황에 따라 처리한다.</p> <p>2.1 사용자가 그룹에 가입되어 있지 않는 경우, 향후 진행 예정인 투표 조회가 불가능 하다고 출력한다.</p> <p>2.2 사용자가 그룹에 가입되어 있는 경우, 그룹 내의 향후 진행 예정인 투표들의 투표 주제, 투표 항목 수, 시작시간, 마감 시간을 출력한다.</p>

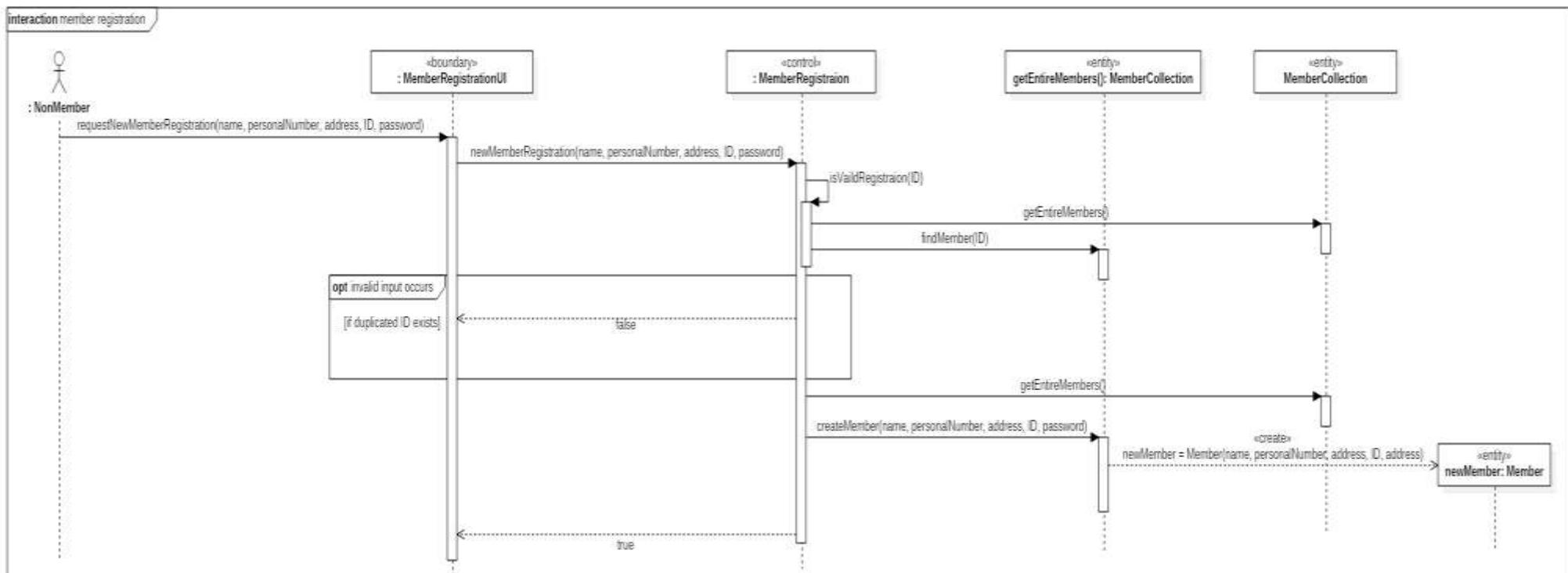
종료된 투표 조회 (Show Terminated Vote)	<p>1. None</p> <p>2. 사용자의 상황에 따라 처리한다.</p> <p>2.1 사용자가 그룹에 가입되어 있지 않는 경우, 향후 진행 예정인 투표 조회가 불가능 하다고 출력한다.</p> <p>2.2 사용자가 그룹에 가입되어 있는 경우, 그룹 내의 종료된 투표들의 투표 주제, 투표 결과를 출력한다</p>
종료된 투표 삭제 (Delete Terminated Vote)	<p>1. 4주 이상 투표 자동 삭제 타이머는 종료 된다. 4주가 지난 투표에 대해 자동 삭제 이 벤트를 발생시킨다.</p> <p>2. None</p>

초기 구조(initial Architecture)

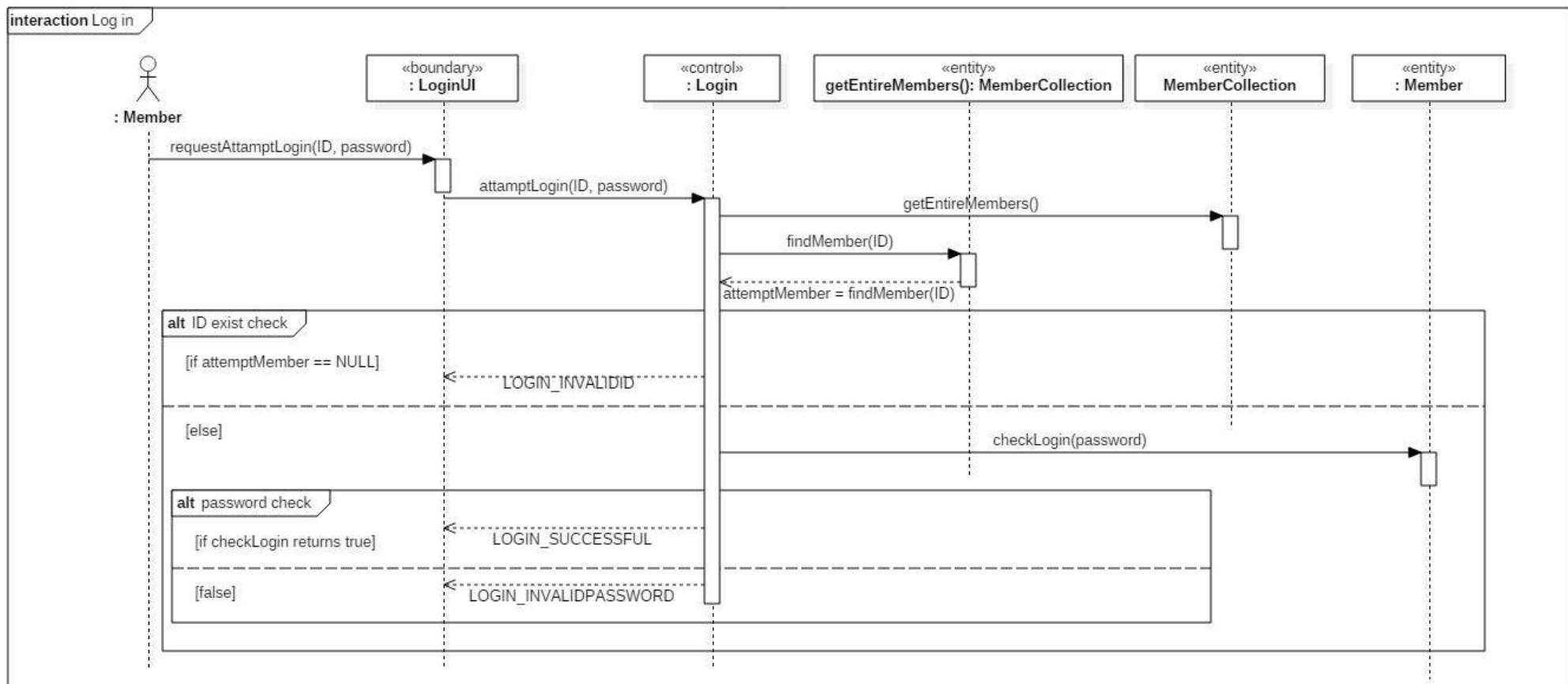


시퀀스 다이어그램(Sequence Diagram)

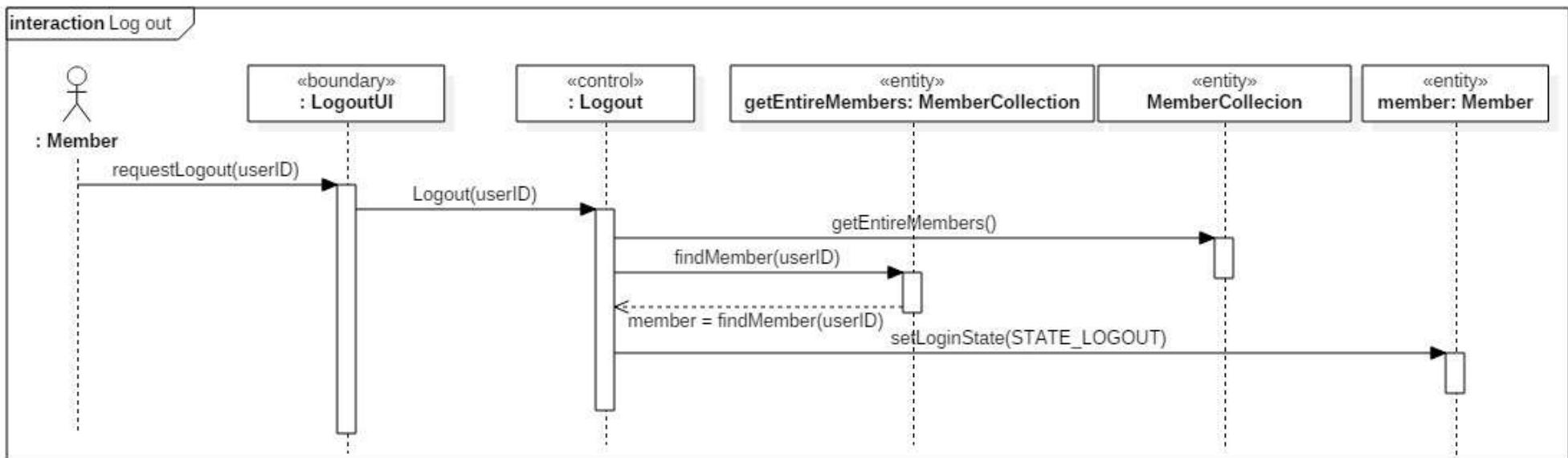
회원 가입(Member Registration)



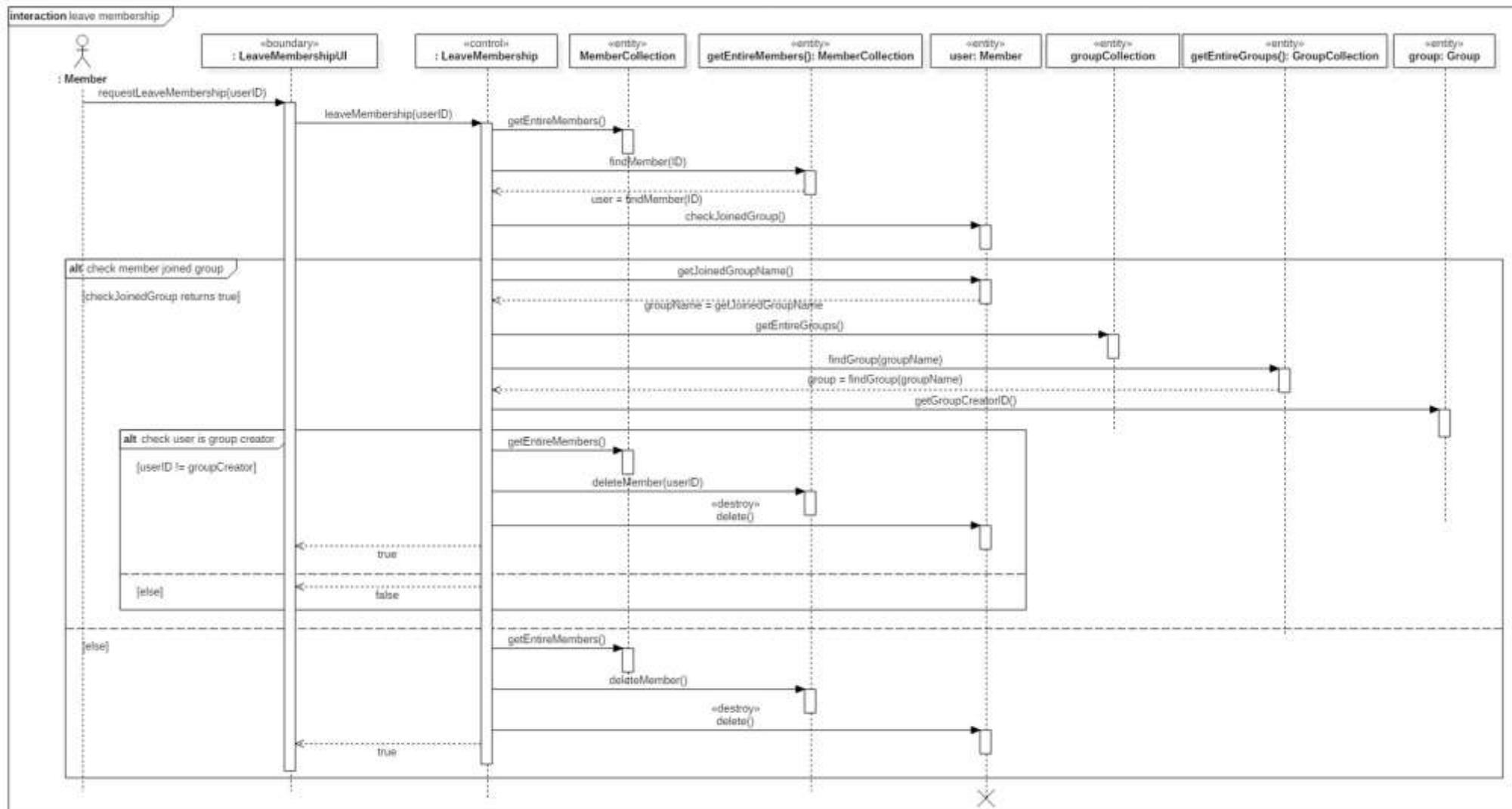
로그인(Login)



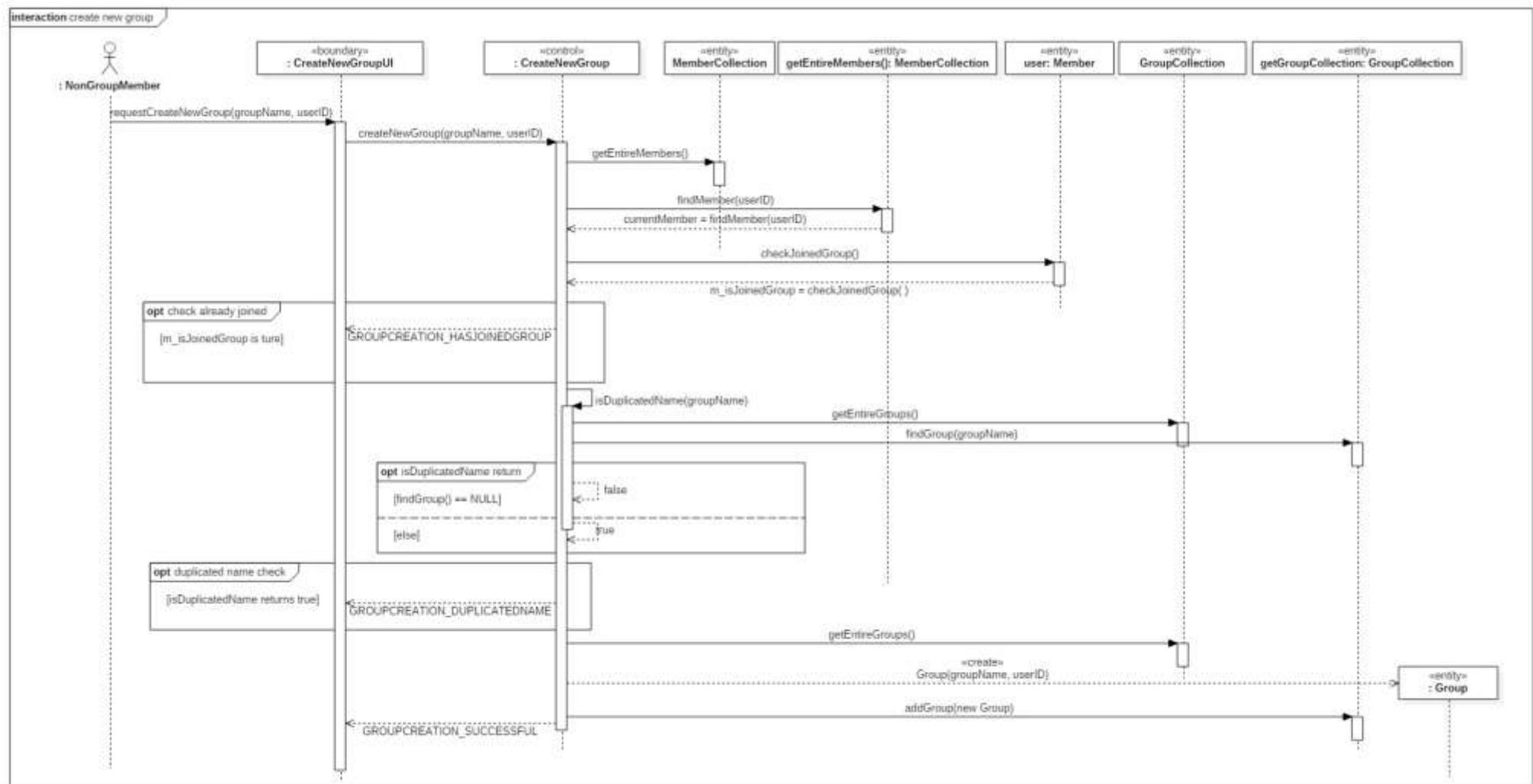
로그아웃(Logout)



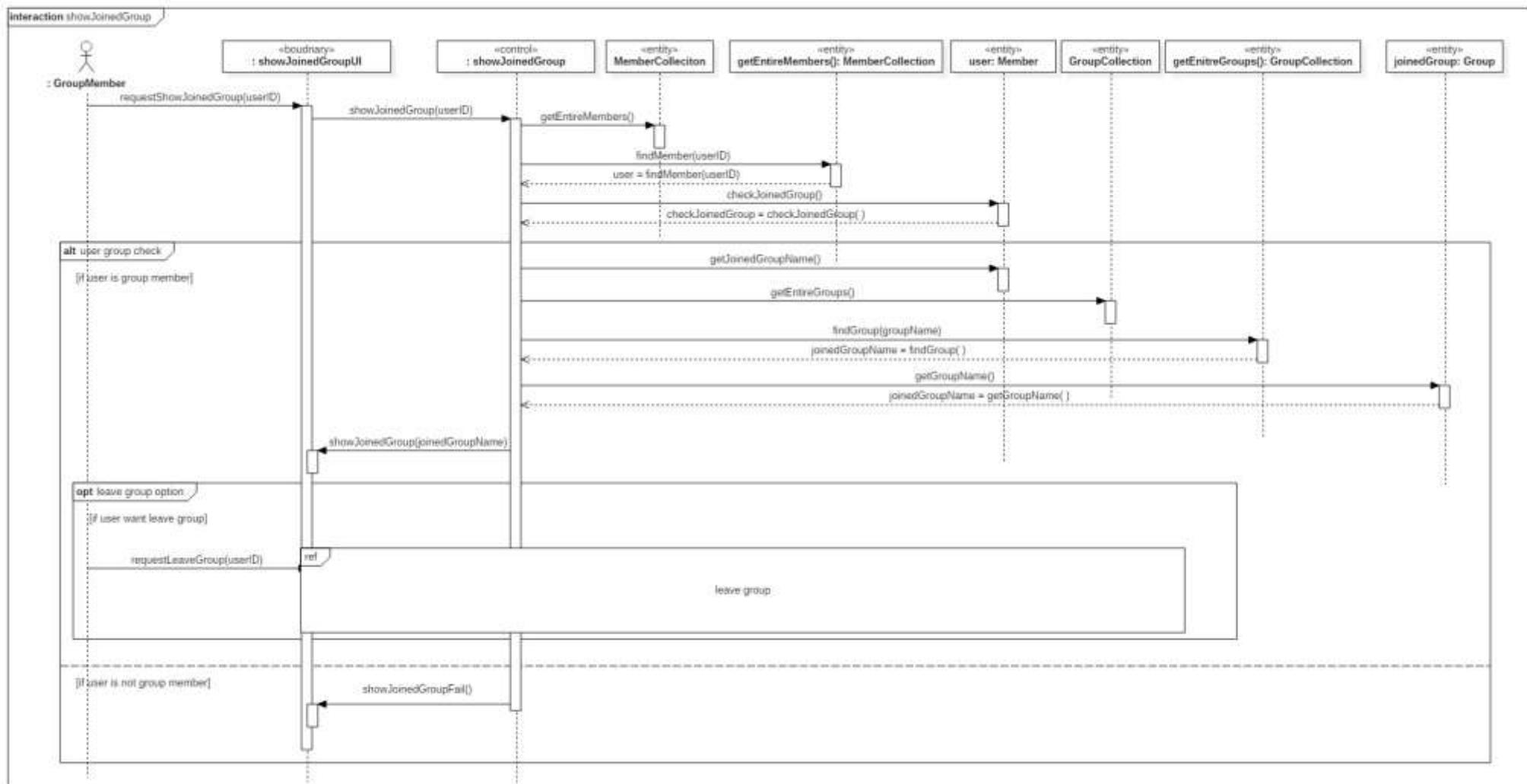
회원 탈퇴(Leave Membership)



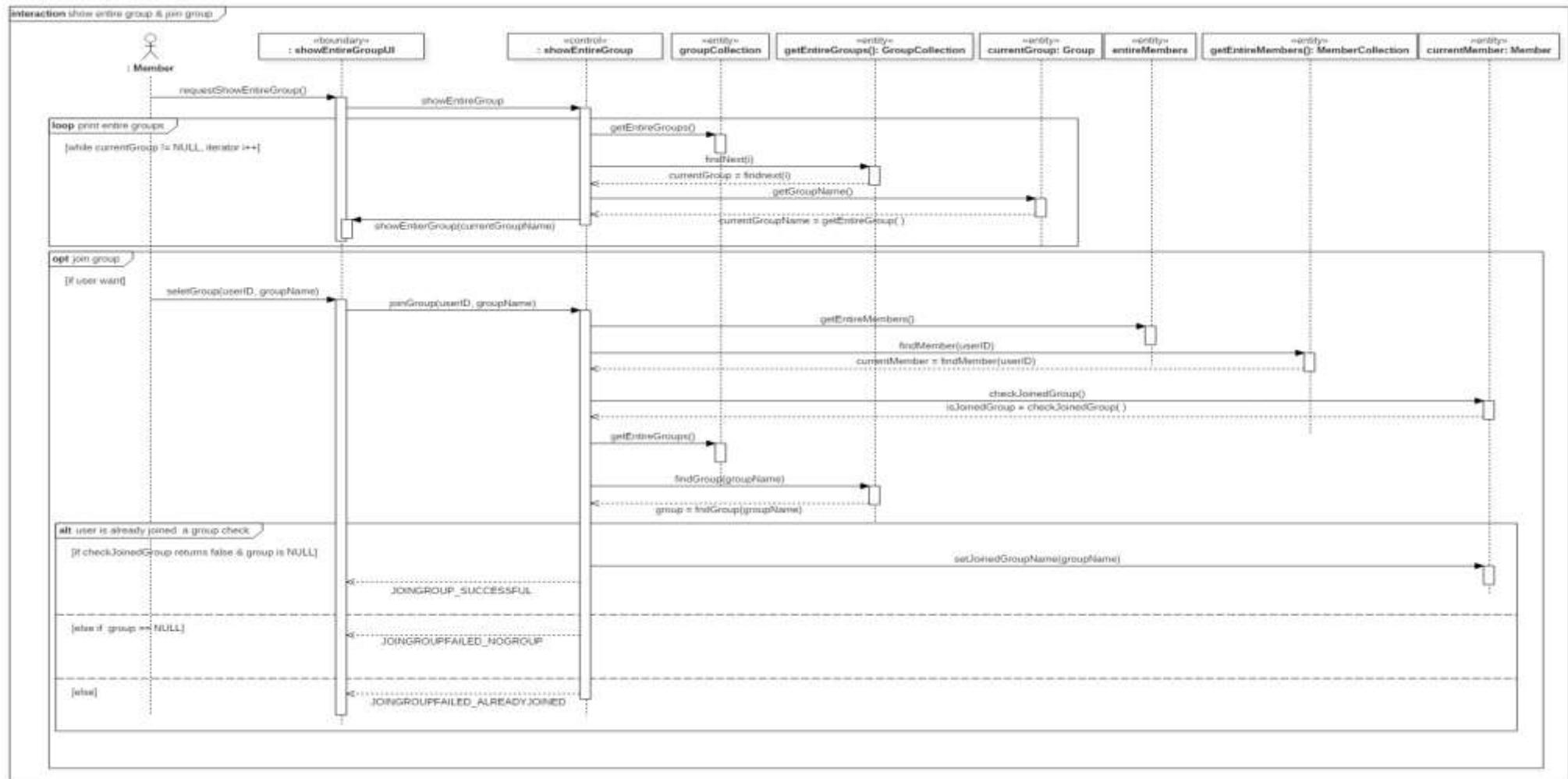
그룹 생성(Create Group)



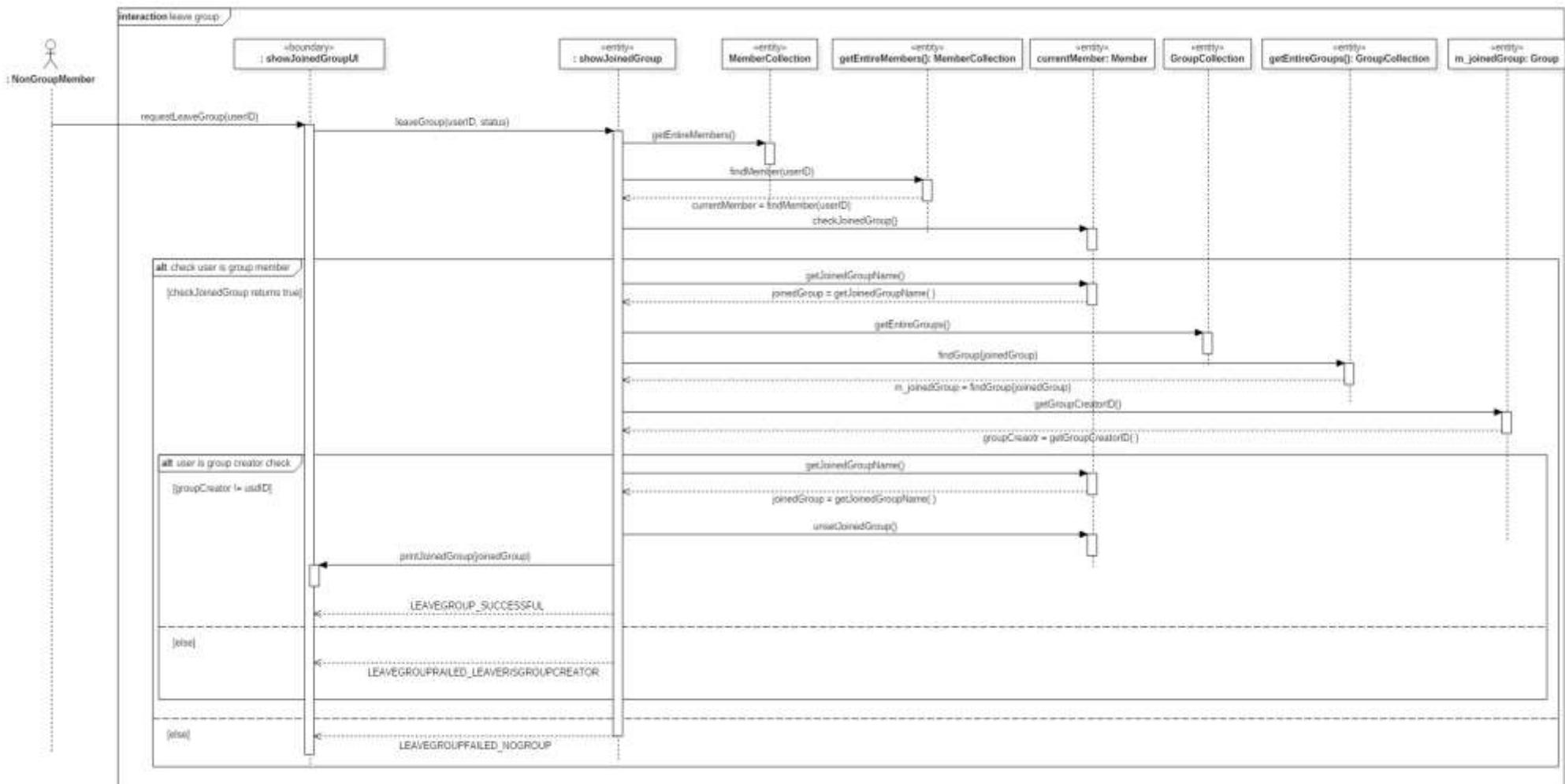
가입 그룹 조회(Show Joined Group)



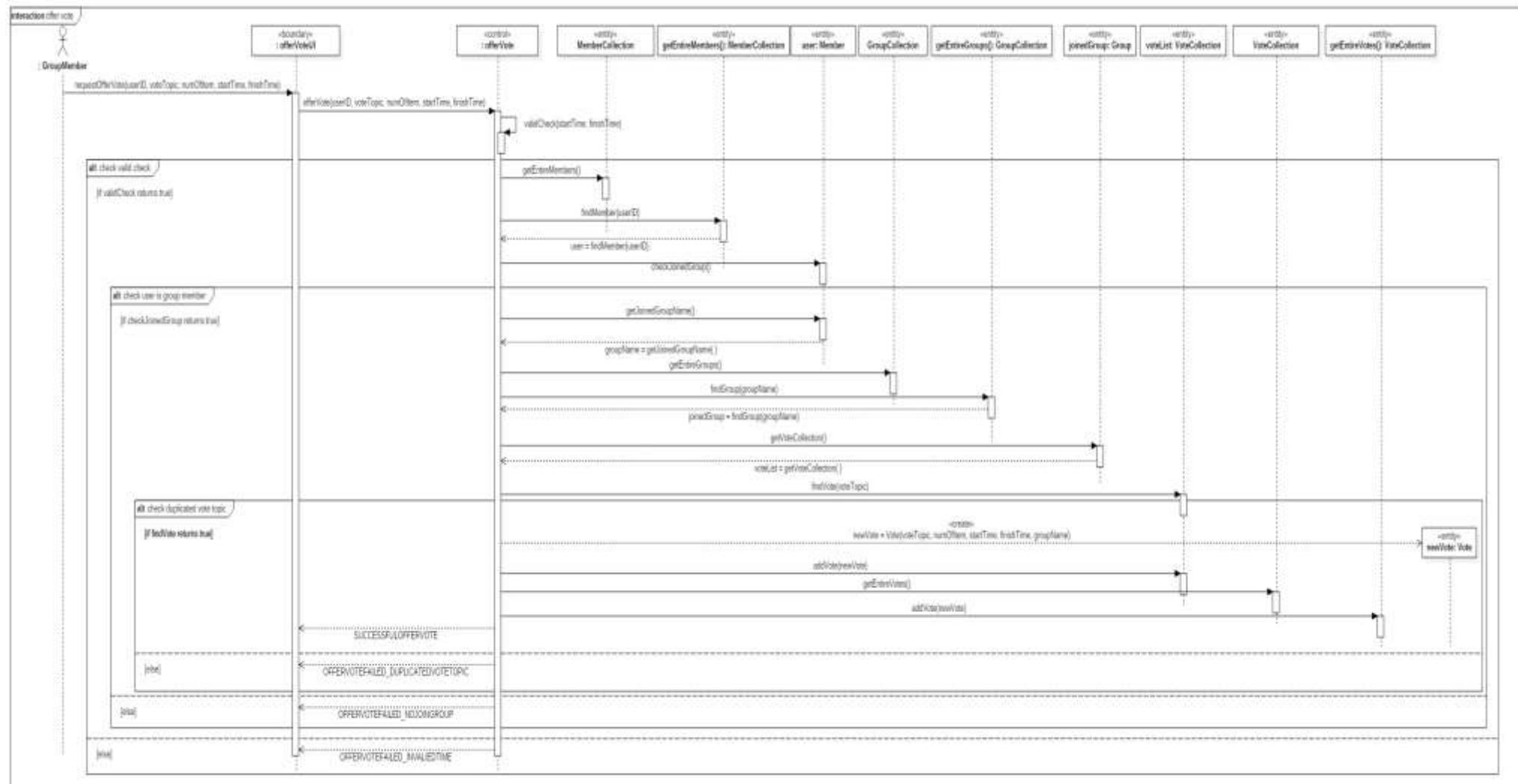
전체 그룹 조회 및 그룹 가입(Show Entire Group & Join Group)



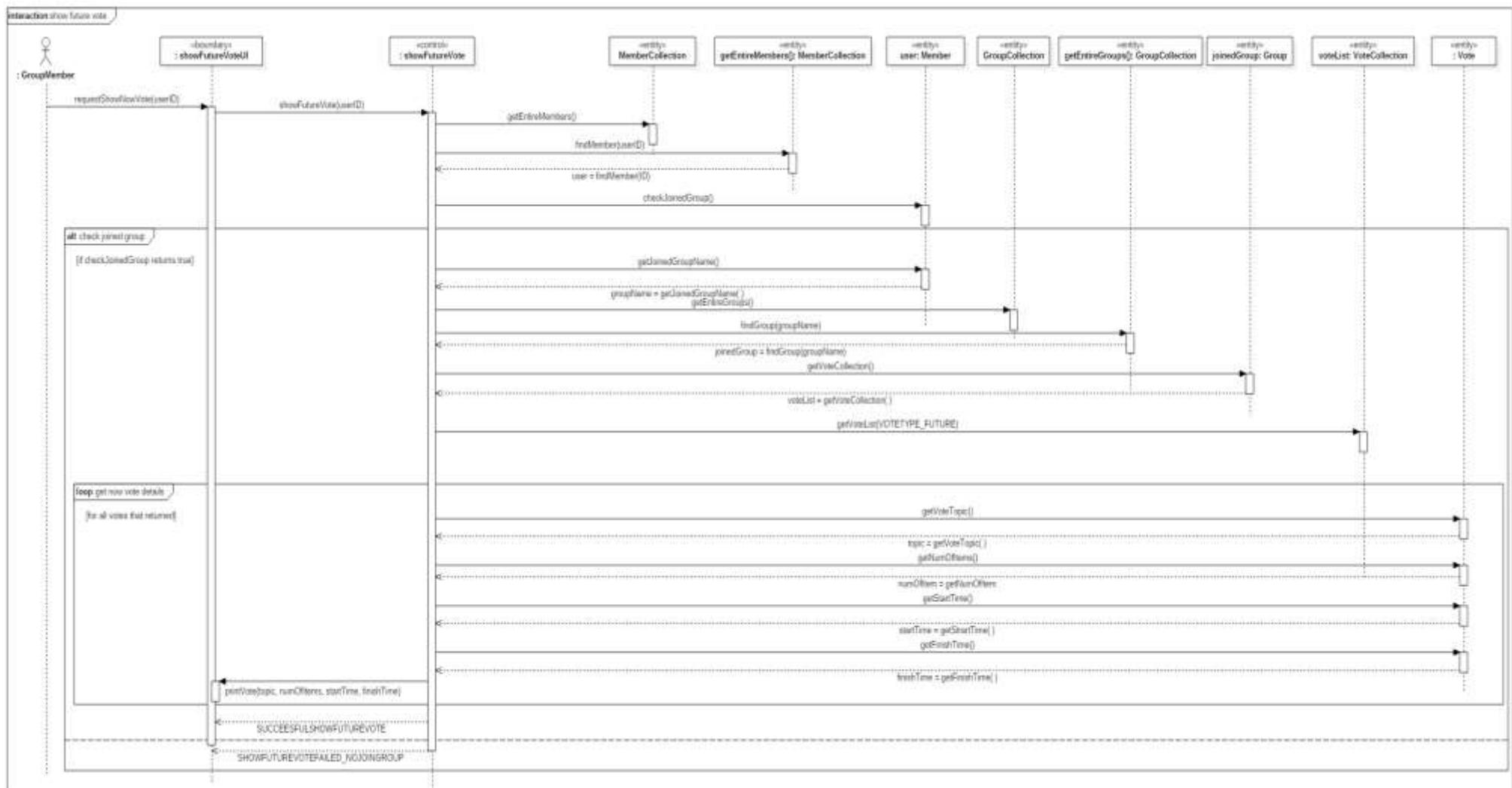
그룹 탈퇴(Leave Group)



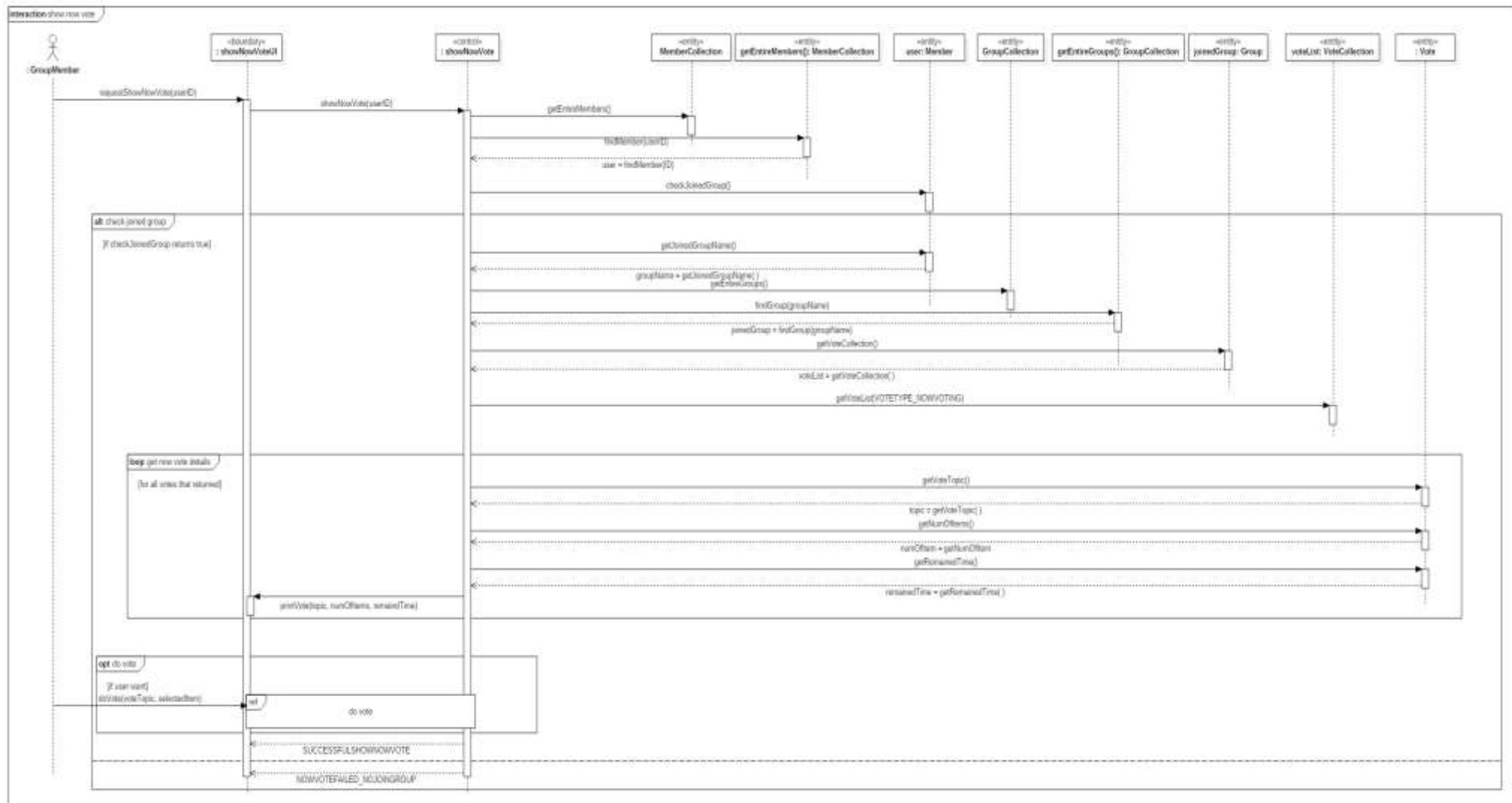
투표 제안(Offer Vote)



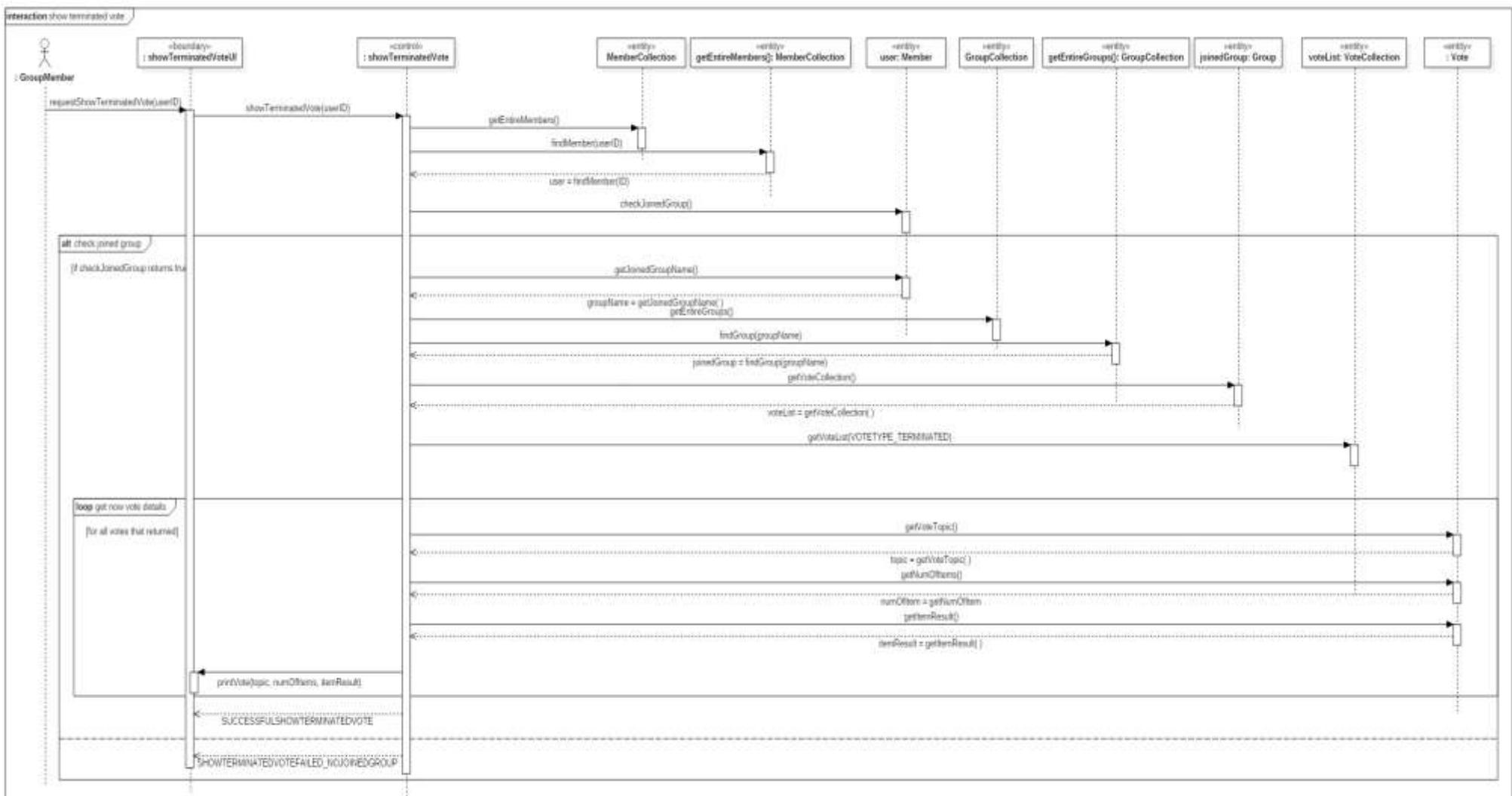
향후 진행 예정인 투표 조회(show future vote)



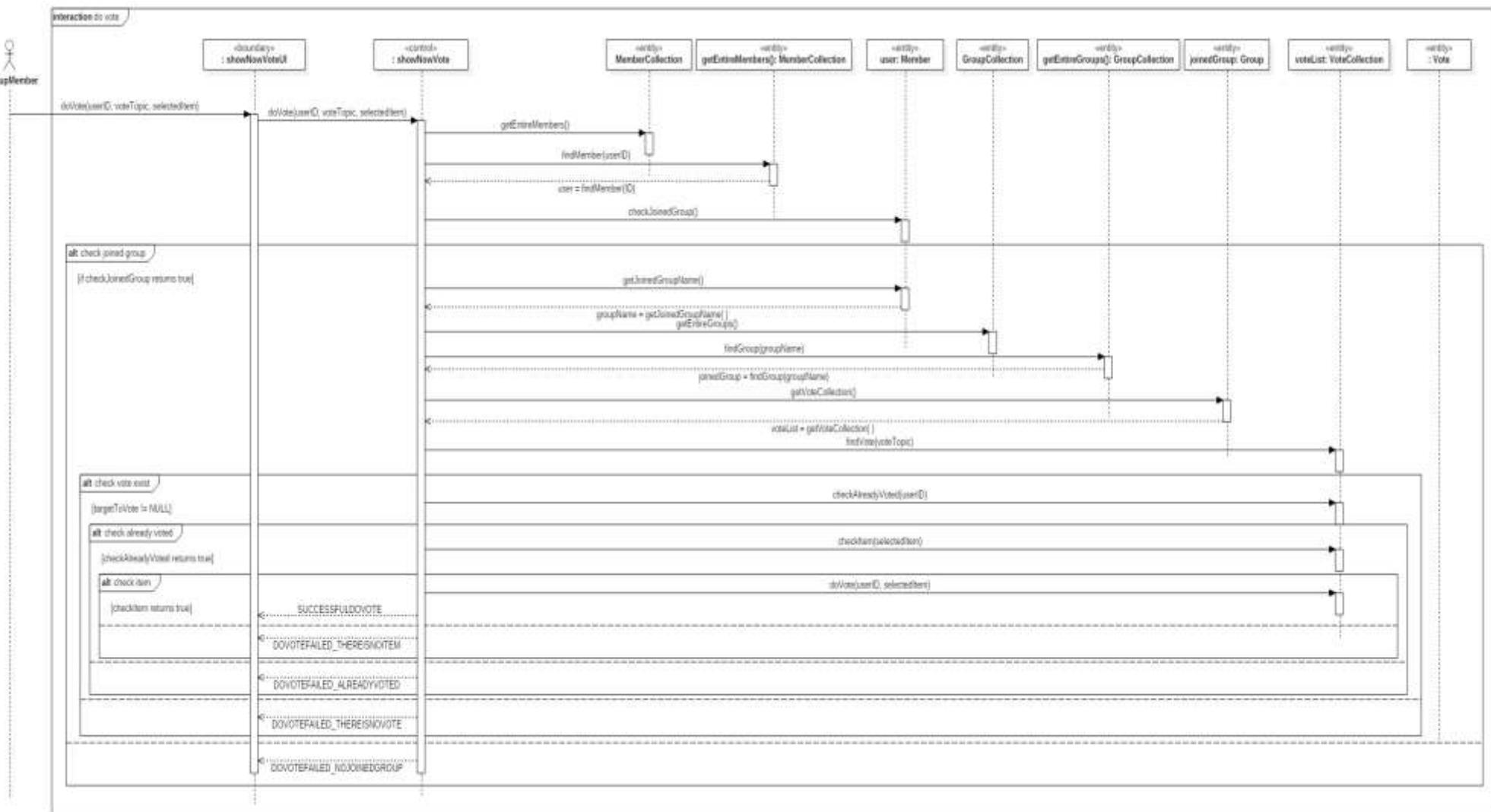
현재 진행중인 투표 조회(show now vote)



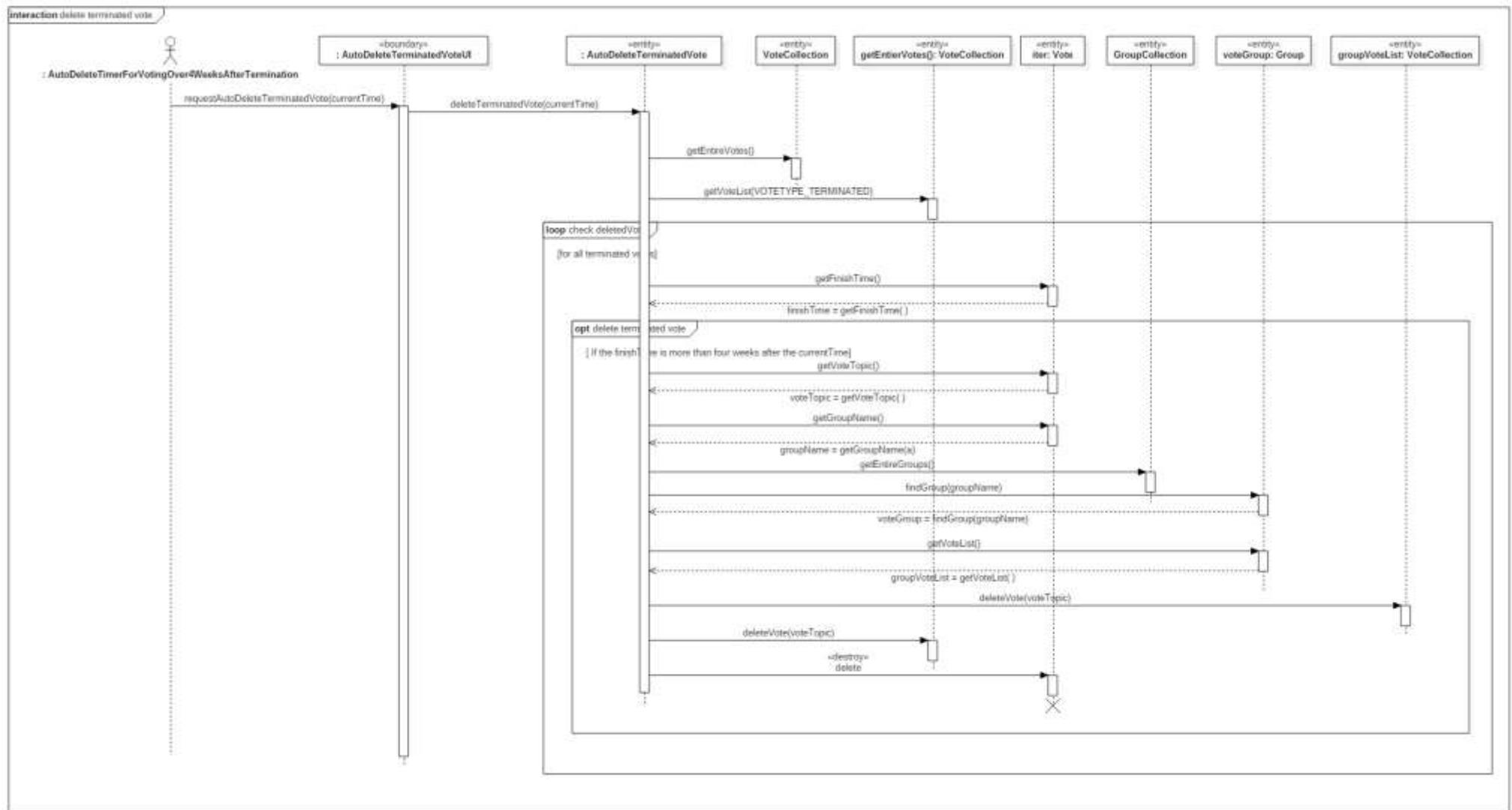
종료된 투표 조회(show terminated vote)



투표(do vote)

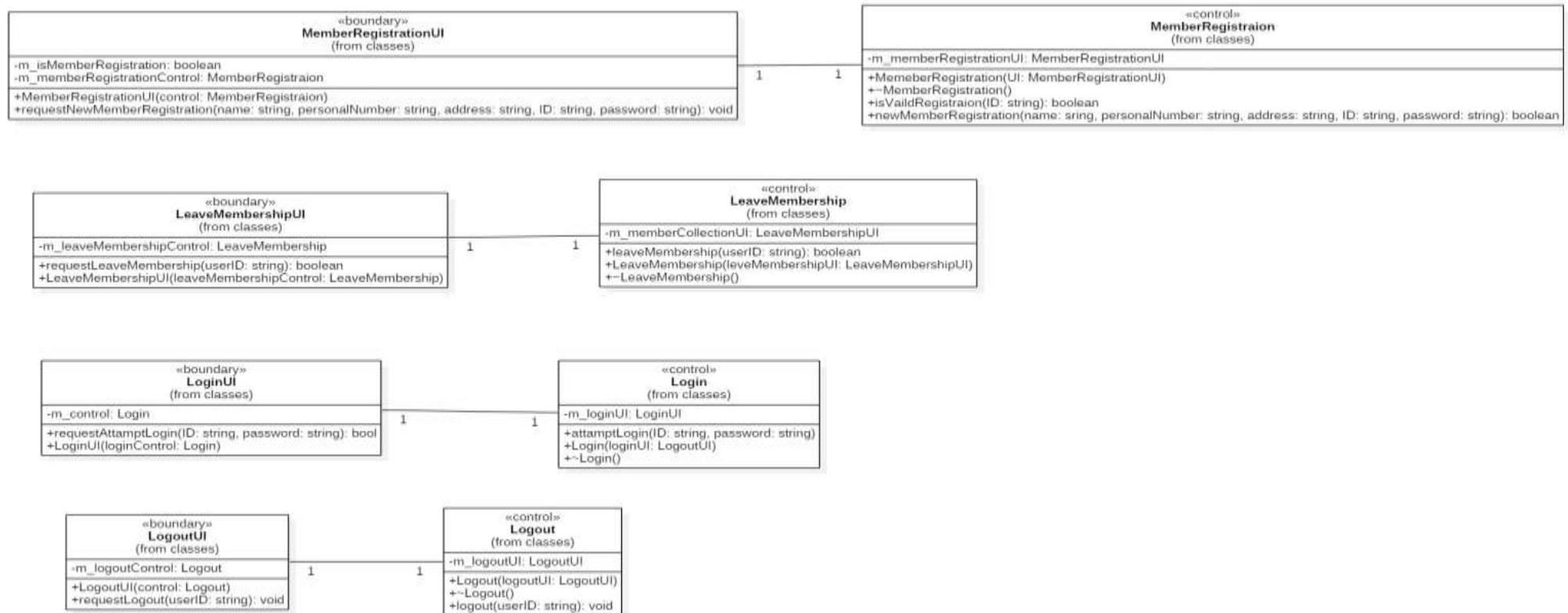


종료 후 4주 지난 투표 삭제

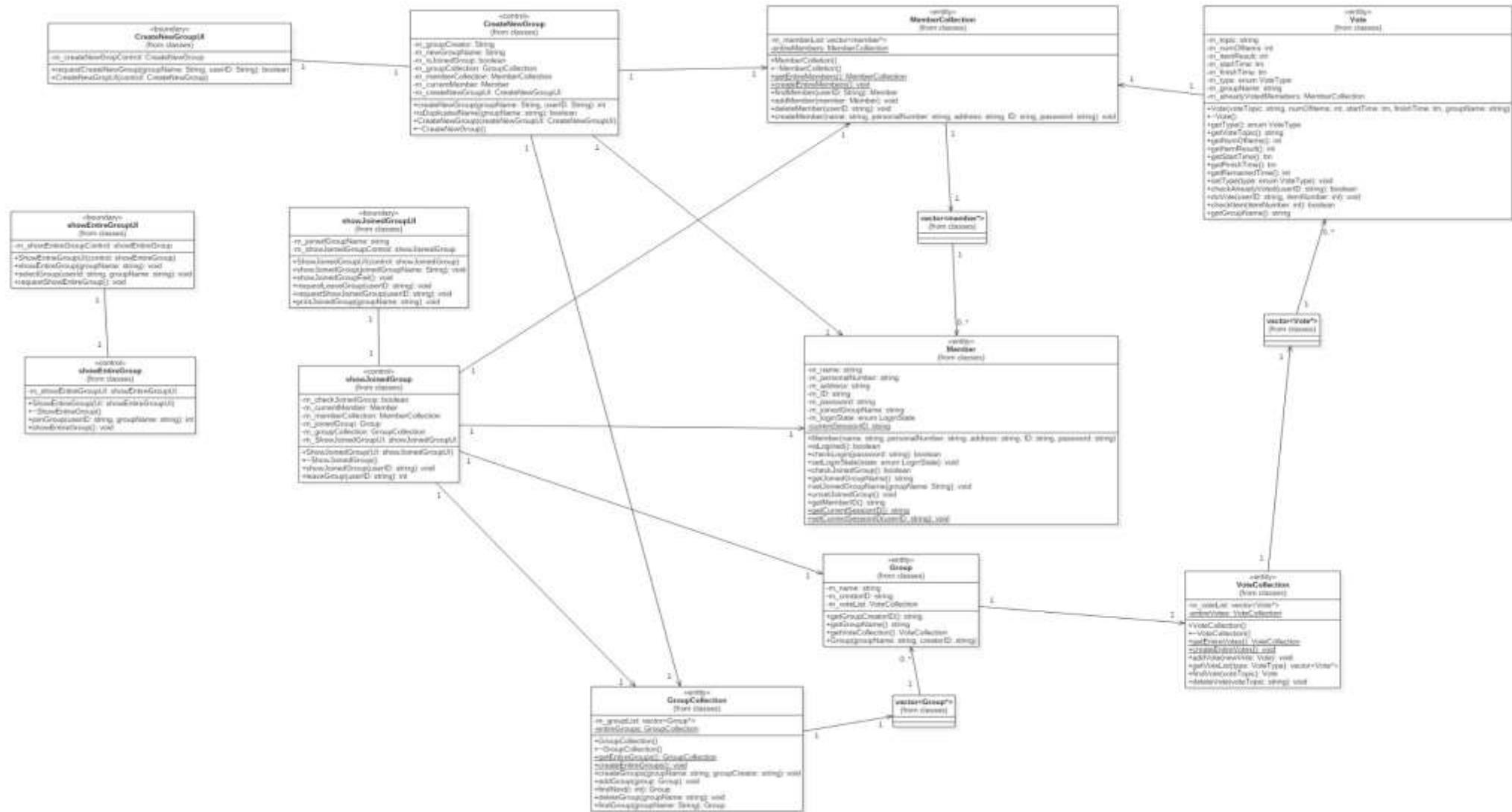


디자인 클래스 다이어그램(Design Class Diagram)

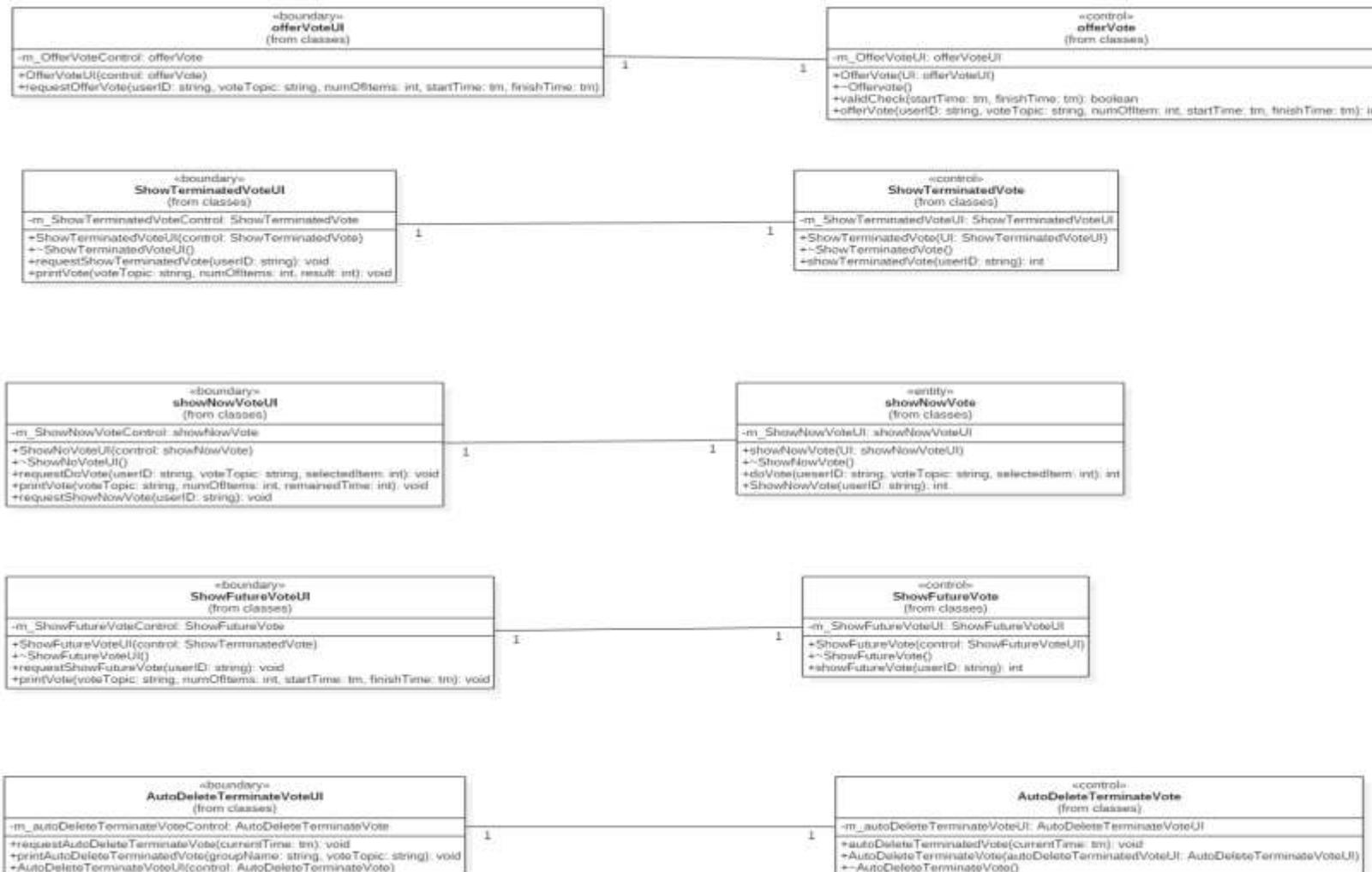
멤버 관리부분(Member Management)



그룹 관리부분 (Group Management)



투표 관리부분 (Vote Management)



소스코드

Member.h

```
// Class : Member
// Description : 멤버 객체를 위한 Member Class
// 작성자 : 심민우
// 작성 날짜 : 2017.05.27 13:00
// 설명 : class Member의 정의 (.h)
// 수정자 : 강석원
// 수정 날짜 : 2017.05.27 15:00
// 수정 이유 : 순환참조를 위한 #ifndef 추가
// 수정자 : 이한솔
// 수정 시간 : 2017.05.29 16:45
// 수정 이유 : 현재 세션의 ID를 저장할 static 변수
currentSessionID, 값을 변경할 get, set 함수 추가

#ifndef MEMBER_H
#define MEMBER_H

#include <string>
using namespace std;

enum LoginState {
    // 로그인 상태 정보를 위한 enum 변수
    // STATE_LOGIN = 로그인 상태
    // STATE_LOGOUT = 로그아웃 상태
    STATE_LOGIN,
    STATE_LOGOUT
};

class Member{
private:
    string m_name;
    string m_personalNumber;
    string m_address;
    string m_ID;
    string m_password;
    string m_joinedGroupName;
    LoginState m_loginState;

    static string currentSessionID;
public:

    //Function : Member(string name, string
personalNumber, string address, string ID, string
password)
    //Description : Member 생성자
    //Parameter : string name, string personalNumber,
string address, string ID, string password
    //Return Value : Member
    Member(string name, string personalNumber, string
address, string ID, string password);
}
```

```
//Function : bool isLoggedIn()
//Description : 로그인 한 상태인지 확인하기 위한
함수
//Parameter : void
//Return Value :
// 1) true : 로그인 되어 있는 상태인 경우
// 2) false : 로그인 되어있지 않은 상태인 경우
bool isLoggedIn();

//Function : bool checkLogin(string password)
//Description : 유효한 password 인지 확인하기 위한
함수
//Parameter : string password
//Return Value :
//1) true : password가 유효한 경우
//2) false : password가 유효하지 않은 경우
bool checkLogin(string password);

//Function : void setLoginState(enum LoginState
state)
//Description : 로그인 상태 설정을 위한 함수
//Parameter : enum LoginState state
//Return Value : void
void setLoginState(enum LoginState state);

//Function : bool checkJoinedGroup()
//Description : 가입한 그룹이 있는지 확인하기 위한
함수
//Parameter : void
//Return Value :
//1) true : 가입한 그룹이 있는 경우
//2) false : 가입한 그룹이 없는 경우
bool checkJoinedGroup();

//Function : string getJoinedGroupName()
//Description : 가입한 그룹의 이름을 가져오기 위한
함수
//Parameter : void
//Return Value : string
string getJoinedGroupName();

//Function : void setJoinedGroupName(string
groupName)
//Description : 가입한 그룹을 세팅하기 위한 함수
//Parameter : string groupName
//Return Value : void
void setJoinedGroupName(string groupName);

//Function : void unsetJoinedGroup()
//Description : 가입한 그룹을 해제하기 위한 함수
//Parameter : void
//Return Value : void
void unsetJoinedGroup();

//Function : string getMemberID()
//Description : Member의 ID를 가져오기 위한 함수
```

```

//Parameter : void
//Return Value : string
string getMemberID();

//Function : static string getCurrentSessionID()
//Description : currentSessionID를 가져오기 위한
함수
//Parameter : void
//Return Value : string
static string getCurrentSessionID();

//Function : static void setCurrentSessionID(string
userID)
//Description : currentSessionID를 설정하기 위한
함수
//Parameter : string userID
//Return Value : void
static void setCurrentSessionID(string userID);
};

#endif

```

Member .cpp

```

// 작성자 : 심민우
// 작성 날짜 : 2017.05.27
// 설명 : Member Class operation 정의 부분
// 수정자 : 이한솔
// 수정 시간 : 2017.05.29 19:45
// 수정 이유 : m_loginState 추가 및 checkLogin()
수정

#include "Member.h"

string Member::currentSessionID;

Member::Member(string name, string personalNumber,
string address, string ID, string password)
{
    m_name = name;
    m_personalNumber = personalNumber;
    m_address = address;
    m_ID = ID;
    m_password = password;
    string str("");
    m_joinedGroupName = str;
    m_loginState = STATE_LOGOUT;
}

bool Member::isLogined()
{

```

```

    if (m_loginState == STATE_LOGIN)
        return true;
    else
        return false;
}

bool Member::checkLogin(string password)
{
    if (m_password == password)
        return true;
    else
        return false;
}

void Member::setLoginState(LoginState state)
{
    m_loginState = state;
}

bool Member::checkJoinedGroup()
{
    if (m_joinedGroupName.empty()) return false;
    else return true;
}

string Member::getJoinedGroupName()
{
    return m_joinedGroupName;
}

void Member::setJoinedGroupName(string groupName)
{
    m_joinedGroupName = groupName;
}

void Member::unsetJoinedGroup()
{
    m_joinedGroupName = "";
}

string Member::getMemberID()
{
    return m_ID;
}

string Member::getCurrentSessionID()
{
    return currentSessionID;
}

void Member::setCurrentSessionID(string userID)
{
    currentSessionID = userID;
}

```

Group.h

```
// Class : Group
// Description : 그룹의 정보에 대한 클래스
// 작성자 : 심민우
// 작성 날짜 : 2017.05.27 13:00
// 설명 : Group class 의 Header
// 수정자 : 강석원
// 수정 날짜 : 2017.05.27 15:00
// 수정 이유 : 순환참조를 위한 #ifndef 추가
// 수정자 : 심민우
// 수정 날짜 : 2017.05.27 18:00
// 수정 이유 : 생성자 Group(string, string) 정의
// 수정자 : 이한솔
// 수정 시간 : 2017.05.30 02:30
// 수정 이유 : 불필요한 operation 삭제

#ifndef GROUP_H
#define GROUP_H

#include <string>
#include "VoteCollection.h"
using namespace std;

class Group {

private:
    string m_name;
    string m_creatorID;
    VoteCollection m_voteList;

public:
    //Function : string getGroupCreatorID()
    //Description : m_creatorID(그룹을 생성한 회원의 ID)를 반환
    //Parameter : 없음
    //Return Value : string m_creatorID
    string getGroupCreatorID();
    //Function : string getGroupName()
    //Description : m_name(그룹 이름)을 반환
    //Parameter : 없음
    //Return Value : string m_name
    string getGroupName();
    //Function : VoteCollection& getVoteCollection()
    //Description : m_voteList(그룹의 투표리스트)의 주소값을 반환
    //Parameter : 없음
    //Return Value : VoteCollection& m_voteList
    VoteCollection& getVoteCollection();
    //Function : Group(string groupName, string creatorID)
    //Description : Group 오브젝트 생성자
    //Parameter : string groupName, string creatorID
    //Return Value : Group
    Group(string groupName, string creatorID);
};

#endif
```

```
#endif
```

Group.cpp

```
// 작성자 : 심민우
// 작성 날짜 : 2017.05.27
// 설명 : Group Class Operation 정의부분
// 수정자 : 심민우
// 수정 날짜 : 2017.05.27 18시
// 수정 이유 : 생성자 함수 Group(string, string)
// 정의

// 수정자 : 이한솔
// 수정 시간 : 2017.05.30 02:30
// 수정 이유 : 불필요한 operation 삭제

#include "Group.h"
#include "GroupCollection.h"

string Group::getGroupCreatorID()
{
    return m_creatorID;
}

string Group::getGroupName()
{
    return m_name;
}

VoteCollection& Group::getVoteCollection()
{
    return m_voteList;
}

Group::Group(string groupName, string creatorID)
{
    m_name = groupName;
    m_creatorID = creatorID;
}
```

Vote.h

```
// Class : Vote
// Description : 투표 객체를 위한 class

// 작성자 : 심민우
// 작성 날짜 : 2017.05.27 13:00
// 설명 : class Vote의 정의(.h)

// 수정자 : 강석원
// 수정 날짜 : 2017.05.27 15:00
// 수정 이유 : 순환참조를 위한 #ifndef 추가

// 수정자 : 강석원
// 수정 날짜 : 2017.05.27 18:09
// 수정 이유 : 생성자 추가, 전체 구현

// 수정자 : 이한솔
// 수정 날짜 : 2017.05.28 04:20
// 수정 이유 : enum VoteType에 VOTETYPE_ALL 추가.
setType() 추가, 구현

// 수정자 : 이한솔
// 수정 시간 : 2017.05.30 03:33
// 수정 이유 : Vote::getRemainedTime() 수정

#ifndef VOTE_H
#define VOTE_H

#include "MemberCollection.h"
#include <string>
#include <ctime>
using namespace std;

enum VoteType {
    // Vote의 Type을 설정하기 위한 enum
    // VOTETYPE_ALL = 모든 투표
    // VOTETYPE_FUTURE = 향후 진행 예정인 투표
    // VOTETYPE_NOWVOTING = 현재 진행중인 투표
    // VOTETYPE_TERMINATED = 종료된 투표
    VOTETYPE_ALL,
    VOTETYPE_FUTURE,
    VOTETYPE_NOWVOTING,
    VOTETYPE_TERMINATED
};

typedef enum VoteType VoteType;

class Vote
{
private:
    string m_topic;
    int m_numOfItems;
    int *m_itemResult;
    tm m_startTime;
    tm m_finishTime;
```

```
    VoteType m_type;
    string m_groupName;
    MemberCollection* m_alreadyVotedMembers;
public:
    //Function : Vote(string voteTopic, int numOfItems, tm startTime, tm finishTime, string groupName)
    //Description : parameter로 받은 변수들을 멤버변수에 대입하는 생성자
    //Parameter : string voteTopic, int numOfItems, tm startTime, tm finishTime, string groupName
    //Return Value : Vote
    Vote(string voteTopic, int numOfItems, tm startTime, tm finishTime, string groupName);
    ~Vote();

    //Function : enum VoteType getType()
    //Description : vote의 type을 얻어오기 위한 함수
    //Parameter : void
    //Return Value: votetype (m_type)
    enum VoteType getType();

    //Function : string getVoteTopic()
    //Description : vote의 voteTopic을 얻어오기 위한 함수
    //Parameter : void
    //Return Value : 해당 투표의 voteTopic
    string getVoteTopic();

    //Function : int getNumOfItems()
    //Description : vote의 투표 항목 갯수를 얻어오기 위한 함수
    //Parameter : void
    //Return Value : 해당 vote의 투표 항목 갯수 (m_numOfItems)
    int getNumOfItems();

    //Function : int* getItemResult()
    //Description : vote의 투표 결과를 얻어오기 위한 함수
    //Parameter : void
    //Return Value : 해당 vote의 투표 결과 (*m_itemResult)
    int* getItemResult();

    //Function : tm getStartTime()
    //Description : vote의 시작 시간을 얻어오기 위한 함수
    //Parameter : void
    //Return Value : 해당 vote의 시작 시간 (m_startTime)
    tm getStartTime();

    //Function : tm getEndTime()
    //Description : vote의 종료 시간을 얻어오기 위한 함수
    //Parameter : void
```

```

//Return Value : 해당 vote 의 종료 시간
(tm_getFinishTime)
tm getFinishTime();

//Function : int getRemainedTime()
//Description : vote의 남아있는 시간을 얻어오기 위한
함수
//Parameter : void
//Return Value : int (남아있는 시간)
int getRemainedTime();

//Function : void setType(enum VoteType type)
//Description : vote의 type을 설정하기 위한 함수
//Parameter : enum VoteType type
//Return Value : void
void setType(enum VoteType type);

//Function : bool checkAlreadyVoted(string userID)
//Description : 투표하기를 한 유저가 해당 투표에
이미 투표한 유저인지 확인하는 함수
//Parameter : string userID
//Return Value :
//1) true : 해당 투표에 투표하지 않았을 시
//2) false : 해당 투표에 이미 투표를 했을 시
bool checkAlreadyVoted(string userID);

//Function : void doVote(string userID, int
itemNumber)
//Description : 투표하기를 위한 함수
//Parameter : string userID, int itemNumbers
//Return Value : void
void doVote(string userID, int itemNumber);

//Function : bool checkItem(int itemNumber)
//Description : 투표시 투표항목의 갯수를 넘는 항목에
대해 투표하는지 확인하는 함수
//Parameter : int itemNumber
//Return Value :
//1) true : 해당 투표의 투표항목 갯수를 넘지 않았을
시 true
//2) false : 해당 투표의 투표항목 갯수를 넘었을 시
false
bool checkItem(int itemNumber);

//Function : string getGroupName()
//Description : 해당 투표가 속한 그룹의 이름을
가져오기 위한 함수
//Parameter : void
//Returnvalue : string (m_groupName)
string getGroupName();
};

#endif

```

Vote.cpp

```

// 작성자 : 심민우
// 작성 날짜 : 2017.05.27
// 설명 : Vote Class Operation 정의
// 수정자 : 강석원
// 수정 날짜 : 2017.05.27 18:09
// 수정 이유 : class Vote 생성자 및 소멸자 추가, 전역
변수들의 선언 및 전제 정의
// 수정자 : 이한솔
// 수정 날짜 : 2017.05.28 04:00
// 수정 이유 : largeTm() main.cpp으로 이동, >연산자
오버로딩으로 대체.
// 수정자 : 이한솔
// 수정 날짜 : 2017.05.28 04:20
// 수정 이유 : setType() 추가, 구현
// 수정자 : 이한솔
// 수정 날짜 : 2017.05.29 18:30
// 수정 이유 : doVote() null reference 오류 제어
추가
// 수정자 : 이한솔
// 수정 시간 : 2017.05.30 03:33
// 수정 이유 : Vote::getRemainedTime() 수정

#include "Vote.h"

extern tm currentTime;
extern bool operator>(tm, tm);
extern time_t GetTimeT(tm tmTime);

Vote::Vote(string voteTopic, int numOfItems, tm
startTime, tm finishTime, string groupName)
{
    int i;

    m_topic = voteTopic;
    m_numOfItems = numOfItems;
    m_startTime = startTime;
    m_finishTime = finishTime;
    m_groupName = groupName;

    m_itemResult = new int[m_numOfItems];
    for (i = 0; i < m_numOfItems; i++)
        m_itemResult[i] = 0;

    if (startTime > currentTime)
        m_type = VOTETYPE_FUTURE;
    else if (finishTime > currentTime)
        m_type = VOTETYPE_NOWVOTING;
    else
        m_type = VOTETYPE_TERMINATED;

    m_alreadyVotedMembers = new MemberCollection();
}

Vote::~Vote()

```

```

{
    delete m_itemResult;
    delete m_alreadyVotedMembers;
}

enum VoteType Vote::getType()
{
    return m_type;
}

string Vote::getVoteTopic()
{
    return m_topic;
}

int Vote::getNumOfItems()
{
    return m_numOfItems;
}

int * Vote::getItemResult()
{
    return m_itemResult;
}

tm Vote::getStartTime()
{
    return m_startTime;
}

tm Vote::getFinishTime()
{
    return m_finishTime;
}

int Vote::getRemainedTime()
{
    // 남아있는 투표 시간을 반환하는 함수
    // 해당 vote의 멤버변수인 m_finishTime 을 가져오고
    // 현재 시간과 비교하여 남아있는 시간 반환
    if (m_type == VOTETYPE_TERMINATED)
    {
        return 0;
    }
    else
    {
        time_t finishTimeT = GetTimeT(m_finishTime);
        time_t currentTimeT = GetTimeT(currentTime);

        int remainSeconds = (int)diffTime(finishTimeT,
                                         currentTimeT);

        return remainSeconds;
    }
}

void Vote::setType(VoteType type)
{
    m_type = type;
}

bool Vote::checkAlreadyVoted(string userID)
{
    // 이미 해당 투표에 유저가 투표했는지 확인하는
    // 함수
    // 투표를 했을 시 false 반환
    // 투표를 하지 않았을 시 true 반환
    if (m_alreadyVotedMembers->findMember(userID) ==
        nullptr)
        return true;
    else
        return false;
}

void Vote::doVote(string userID, int itemNumber)
{
    // 투표를 실행하는 함수
    // parameter로 받은 userID로 해당 멤버의
    // 객체주소를 가져오고
    // 멤버변수인 m_alreadyVotedMembers에 해당 멤버
    // 추가
    // 투표 항목에 대해 투표 반영
    Member* voteMember =
    MemberCollection::getEntireMembers()-
    >findMember(userID);

    m_alreadyVotedMembers->addMember(voteMember);
    m_itemResult[itemNumber - 1]++;
}

bool Vote::checkItem(int itemNumber)
{
    // 투표 한 항목이 해당 투표의 투표 항목 수보다
    // 많은지 확인하는 함수
    // 해당 투표의 투표 항목이 제안한 투표 항목 보다
    // 큰 경우 true 반환
    // 작을 경우 false 반환
    if (m_numOfItems > itemNumber - 1) return true;
    else return false;
}

string Vote::getGroupName()
{
    return m_groupName;
}

```

MemberCollection.h

```
// Class : MemberCollection
// Description : Member 객체리스트 조작을 위한
// 클래스
// 작성자 : 심민우
// 작성 날짜 : 2017.05.27 13:00
// 설명 : class MemberCollection의 정의(.h)
// 수정자 : 강석원 15:00
// 수정 날짜 : 2017.05.27 15:00
// 수정 이유 : 순환참조를 위한 #ifndef 추가
// 수정자 : 강석원
// 수정 날짜 : 2017.05.27 19:16
// 수정 이유 : 전체 구현
// 수정자 : 강석원
// 수정 날짜 : 2017.05.30 01:02
// 수정 이유 : static MemberCollection* entireMembers
// 추가

#ifndef MEMBERCOLLECTION_H
#define MEMBERCOLLECTION_H

#include <vector>
#include <string>
#include "Member.h"
using namespace std;

class MemberCollection
{
private:
    vector<Member *>* m_memberList;
    static MemberCollection* entireMembers;
public:
    MemberCollection();
    ~MemberCollection();

    //Function : static MemberCollection*
    getEntireMembers(void)
    //Description : static 멤버변수 entireMembers 에
    접근하기 위한 함수
    //Parameter : void
    //Return Value : MemberCollection*
    static MemberCollection* getEntireMembers(void);

    //Function : static void createEntireMembers(void)
    //Description : static 멤버변수 entireMembers 생성을
    위한 함수
    //Parameter : void
    //Return Value : void
    static void createEntireMembers(void);

    //Function : Member* findMember(string userID)
    //Description : memberList 에서 userID 에 해당하는
    member를 찾기 위한 함수
    //Parameter : string userID
```

```
//Return Value : Member*
Member* findMember(string userID);

//Function : void addMember(Member* member)
//Description : memberList 에 새 멤버를 추가하기
//위한 함수
//Parameter : Member* member
//Return Value : void
void addMember(Member* member);

//Function : void deleteMember(string userID)
//Description : memberList 에서 userID 에 해당하는
//멤버를 삭제하기 위한 함수
//Parameter : string userID
//Return Value : void
void deleteMember(string userID);

//Function : void createMember(string name, string
personalNumber, string address, string ID, string
password)
//Description : 새로운 멤버를 생성하기 위한 함수
//Parameter : string name, string personalNumber,
string address, string ID, string password
//Return Value : void
void createMember(string name, string
personalNumber, string address, string ID, string
password);
};

#endif
```

MemberCollection.cpp

```
// 작성자 : 심민우
// 작성 날짜 : 2017.05.27
// 설명 : MemberCollection Class operation 정의부분
// 수정자 : 강석원
// 수정 날짜 : 2017.05.27 19:09
// 수정 이유 : 전체 구현
// 수정자 : 심민우
// 수정 날짜 : 2017.05.28 13:00
// 수정 이유 : findMember 수정
// 수정자 : 이한솔
// 수정 날짜 : 2017.05.29 18:30
// 수정 이유 : findMember 리턴 값(nullptr) 수정

#include "MemberCollection.h"

MemberCollection* MemberCollection::entireMembers =
NULL;

MemberCollection::MemberCollection()
{
    m_memberList = new vector<Member*>;
}

MemberCollection::~MemberCollection()
{
    delete m_memberList;
}

MemberCollection *
MemberCollection::getEntireMembers(void)
{
    return entireMembers;
}

void MemberCollection::createEntireMembers(void)
{
    entireMembers = new MemberCollection();
}

Member* MemberCollection::findMember(string userID)
{
    // parameter로 받은 userID에 대한 멤버를 찾는
    // 함수
    // m_memberList의 size가 0인 경우 nullptr를
    // 반환하고
    // 0이 아닐 시 m_memberList에서 각 멤버에 대해
    // Member ID를 가져와 parameter로 받은 userID와 같은지
    // 비교
    // 같은 것이 있으면 iterator 반환하고 없을 시
    // nullptr 반환
    size_t Size = m_memberList->size();
    if (Size == 0)
    {
        return nullptr;
    }
}
```

```
}
```

```
for (int i = 0; i < Size; i++)
{
    Member* iterator = m_memberList->at(i);
    if (iterator->getMemberID() == userID)
        return iterator;
}
return nullptr;
}

void MemberCollection::addMember(Member* member)
{
    m_memberList->push_back(member);
}

void MemberCollection::deleteMember(string userID)
{
    // 멤버변수 m_memberList에서 멤버를 삭제하는 함수
    // m_memberList에서 각 member에 대해 member ID를
    // 가져와서 parameter로 받은 userID와 같은지 비교
    // 같은 것이 있으면 해당 멤버를 삭제함
    vector<Member*>::iterator iter;

    for (iter = m_memberList->begin(); iter != m_memberList->end(); iter++)
    {
        if ((*iter)->getMemberID() == userID)
        {
            m_memberList->erase(iter);
            return;
        }
    }
}

void MemberCollection::createMember(string name,
string personalNumber, string address, string ID,
string password)
{
    Member* newMember = new Member(name,
personalNumber, address, ID, password);
    m_memberList->push_back(newMember);
}
```

VoteCollection.h

```
// Class : VoteCollection
// Description : Vote 객체리스트 조작을 위한 클래스
// 작성자 : 심민우
// 작성 날짜 : 2017.05.27 13:00
// 설명 : class VoteCollection의 정의(.h)
// 수정자 : 강석원
// 수정 날짜 : 2017.05.27 18:42
// 수정 이유 : 전체 구현 및 생성자, 소멸자 정의
// 수정자 : 이한솔
// 수정 날짜 : 2017.05.28 03:52
// 수정 이유 : getVoteList()로
get(Now,Future,Terminated)VoteList() 합체

#ifndef VOTECOLLECTION_H
#define VOTECOLLECTION_H

#include <vector>
#include "Vote.h"
using namespace std;

class VoteCollection
{
private:
    vector<Vote *> *m_voteList;
    static VoteCollection* entireVotes;
public:
    VoteCollection();
    ~VoteCollection();

    //Function : static VoteCollection*
    getEntireVotes(void)
    //Description : VoteCollection 의 member 변수인
    entireVotes 를 반환
    //Parameter : void
    //Return Value : (VoteCollection*) - entireVotes 의
    주소값
    static VoteCollection* getEntireVotes(void);

    //Function : static void createEntireVotes(void);
    //Description : static 변수인 entireVotes 초기화
    //Parameter : void
    //Return Value : void
    static void createEntireVotes(void);

    //Function : void addVote(Vote* newVote)
    //Description : 새 투표를 생성
    //Parameter : Vote* newVote
    //Return Value : void
    void addVote(Vote* newVote);

    //Function : vector<Vote *> *getVoteList(VoteType
    type = VOTETYPE_ALL)
```

```
//Description : 모든 투표리스트를 가져옴
//Parameter : VoteType type = VOTETYPE_ALL
//Return Value : <Vote *> type 의 vector
vector<Vote *> *getVoteList(VoteType type =
VOTETYPE_ALL);

//Function : Vote* findVote(string voteTopic);
//Description : voteTopic 에 해당하는 vote를 찾음
//Parameter : string voteTopic
//Return Value : (Vote*) - 해당 하는 Vote 의 pointer
Vote* findVote(string voteTopic);

//Function : void deleteVote(string voteTopic)
//Description : voteTopic 에 해당하는 vote를 삭제함
//Parameter : string voteTopic
//Return Value : void
void deleteVote(string voteTopic);
};

#endif
```

VoteCollection.cpp

```
// 작성자 : 심민우
// 작성 날짜 : 2017.05.27
// 설명 : VoteCollection Operation 정의
// 수정자 : 강석원
// 수정 날짜 : 2017.05.27 18:42
// 수정 이유 : 전체 구현 및 생성자, 소멸자 정의
// 수정자 : 이한솔
// 수정 날짜 : 2017.05.28 03:52
// 수정 이유 : getVoteList()로
get(Now,Future,Terminated)VoteList() 합체
// 수정자 : 강석원
// 수정 날짜 : 2017.05.30 01:31
// 수정 이유 : static 변수들의 추가.

#include "VoteCollection.h"

VoteCollection* VoteCollection::entireVotes = NULL;

VoteCollection::VoteCollection()
{
    m_voteList = new vector<Vote*>;
}

VoteCollection::~VoteCollection()
{
    delete m_voteList;
}

VoteCollection *
VoteCollection::getEntireVotes(void)
{
    return entireVotes;
}

void VoteCollection::createEntireVotes(void)
{
    entireVotes = new VoteCollection();
}

void VoteCollection::addVote(Vote* newVote)
{
    m_voteList->push_back(newVote);
}

vector<Vote*>* VoteCollection::getVoteList(VoteType type)
{
    // votelist 를 가져오는 함수
    // 멤버 변수 m_voteList 에서 모든 타입의 리스트를
    가져온 후
    // 지역 변수 voteList 에 해당 vote push_back 후
    // voteList 반환
    vector<Vote*>* voteList;
    size_t Size = m_voteList->size();
```

```
voteList = new vector<Vote*>;

for (int i = 0; i < Size; i++)
{
    Vote* iterator = m_voteList->at(i);
    if (iterator->getType() == type || type ==
VOTETYPE_ALL)
        voteList->push_back(iterator);
}
return voteList;
}

Vote* VoteCollection::findVote(string voteTopic)
{
    // vote 를 찾는 함수
    // 멤버변수 m_voteList 에서
    // parameter로 받은 voteTopic 에 대해 각 vote에서
    voteTopic 을 가져와서 같은지 비교
    // 같은 시 iterator 반환
    size_t Size = m_voteList->size();

    for (int i = 0; i < Size; i++)
    {
        Vote* iterator = m_voteList->at(i);
        if (iterator->getVoteTopic() == voteTopic)
            return iterator;
    }
    return nullptr;
}

void VoteCollection::deleteVote(string voteTopic)
{
    // vote를 삭제하는 함수
    // 멤버변수 m_voteList 를 탐색하여 parameter로
    받은 voteTopic 을 찾고
    // 찾을 시 해당하는 vote를 voteList 에서 erase
    시켜줌
    vector<Vote*>::iterator iter;

    for (iter = m_voteList->begin(); iter !=
m_voteList->end(); iter++)
    {
        if ((*iter)->getVoteTopic() == voteTopic)
        {
            m_voteList->erase(iter);
            return;
        }
    }
}
```

GroupCollection.h

```
// Class : GroupCollection
// Description : Group 객체리스트 조작을 위한 클래스
// 작성자 : 심민우
// 작성 날짜 : 2017.05.27 13:00
// 설명 : GroupCollection Class Header
// 수정자 : 강석원
// 수정 날짜 : 2017.05.27 15:00
// 수정 이유 : 순환참조를 위한 #ifndef 추가
// 수정자 : 심민우
// 수정 날짜 : 2017.05.27 18:00
// 수정 이유 : 생성자 GroupCollection() 정의
// 수정자 : 심민우
// 수정 날짜 : 2017.05.28 12:00
// 수정 이유 : findNext() 정의
```

```
#ifndef GROUPCOLLECTION_H_
#define GROUPCOLLECTION_H_
```

```
#include <vector>
#include <string>
#include "Group.h"
using namespace std;

class GroupCollection
{
private:
    vector<Group * *>* m_groupList;
    static GroupCollection* entireGroups;
```

public:

```
GroupCollection();
~GroupCollection();
```

```
//Function : static GroupCollection*
getEntireGroups(void)
//Description : static 멤버변수 entireGroups 를
가져오기 위한 함수
//Parameter : void
//Return Value : GroupCollection*
static GroupCollection* getEntireGroups(void);
```

```
//Function : static void createEntireGroups(void)
//Description : static 멤버변수 entireGroups 를
생성하기 위한 함수
//Parameter : void
//Return Value : void
static void createEntireGroups(void);
```

```
//Function : void createGroup(string groupName,
string groupCreator)
//Description : 새 그룹 생성을 위한 함수
//Parameter : string groupName, string groupCreator
```

```
//Return Value : void
void createGroup(string groupName, string
groupCreator);

//Function : void addGroup(Group* group)
//Description : groupList 에 새 그룹 추가를 위한
함수
//Parameter : Group* group
//Return Value : void
void addGroup(Group* group);

//Function : Group* findNext(int i)
//Description : groupList 에서 다음 그룹을 찾기 위한
함수
//Parameter : int i
//Return Value : Group*
Group* findNext(int i);

//Function : void deleteGroup(string groupName)
//Description : 그룹삭제를 위한 함수
//Parameter : string groupName
//Return Value : void
void deleteGroup(string groupName);

//Function : Group* findGroup(string groupName)
//Description : groupList 에서 해당하는 그룹을 찾기
위한 함수
//Parameter : string groupName
//Return Value : Group*
Group* findGroup(string groupName);
};

#endif
```

GroupCollection.cpp

```
// 작성자 : 심민우  
// 작성 날짜 : 2017.05.27  
// 설명 : GroupCollection Operation 정의  
// 수정자 : 심민우  
// 수정 날짜 : 2017.05.27 18시  
// 수정 이유 : 생성자 GroupCollection() 정의  
// 수정자 : 강석원  
// 수정 날짜 : 2017.05.27 19:19  
// 수정 이유 : 전체 구현  
// 수정자 : 심민우  
// 수정 날짜 : 2017.05.28 12:00  
// 수정 이유 : findNext() 정의  
// 수정자 : 강석원  
// 수정 날짜 : 2017.05.30 01:26  
// 수정 이유 : static 변수 entireGroups 추가
```

```
#include "GroupCollection.h"
```

```
GroupCollection * GroupCollection::entireGroups =  
NULL;
```

```
void GroupCollection::createGroup(string groupName,  
string groupCreator)
```

```
{  
    // Group 객체 생성하고 GroupCollection List 에  
    // 새로 생성한 객체 추가
```

```
    Group *addGroup = new Group(groupName,  
groupCreator);  
    m_groupList->push_back(addGroup);
```

```
}
```

```
void GroupCollection::addGroup(Group * group)
```

```
{  
    m_groupList->push_back(group);
```

```
Group* GroupCollection::findNext( int i )
```

```
{  
    // 그룹 리스트에서 다음 그룹을 가져오기 위한 함수  
    // 다음 그룹이 있으면 iterator 반환 없을시 NULL  
    // 반환
```

```
    size_t Size = m_groupList->size();
```

```
    if (Size == 0 || i >= Size)
```

```
    {  
        return NULL;  
    }
```

```
    Group* iterator = m_groupList->at(i);
```

```
    if (iterator != NULL)
```

```
    {  
        return iterator;  
    }  
    else  
    {  
        return NULL;  
    }  
}
```

```
void GroupCollection::deleteGroup(string groupName)  
{
```

```
    // 그룹 리스트에서 그룹을 삭제하기 위한 함수  
    // 그룹 리스트에서 각 그룹에 대해 groupName 을  
    // 가져온 후 parameter로 받은 groupName과 같은지 비교  
    // 같으면 그 그룹을 삭제함
```

```
    vector<Group*>::iterator iter;
```

```
    for (iter = m_groupList->begin(); iter !=  
m_groupList->end(); iter++)  
    {  
        if ((*iter)->getGroupName() == groupName)  
        {  
            m_groupList->erase(iter);  
            return;  
        }  
    }
```

```
Group* GroupCollection::findGroup(string groupName)  
{
```

```
    // 그룹 리스트에서 그룹을 찾기 위한 함수  
    // 그룹 리스트에서 각 그룹에 대해 groupName 을  
    // 가져온 후 parameter로 받은 groupName과 같은지 비교  
    // 같으면 그 그룹을 반환하고 없을 시 NULL 반환
```

```
    size_t Size = m_groupList->size();
```

```
    for (int i = 0; i < Size; i++)  
{
```

```
    Group* iterator = m_groupList->at(i);  
    if (iterator->getGroupName() == groupName)  
        return iterator;  
}
```

```
    return NULL;
```

```
GroupCollection::GroupCollection()
```

```
{  
    m_groupList = new vector<Group*>;
```

```
GroupCollection::~GroupCollection()
```

```
{  
    delete m_groupList;  
}
```

```

GroupCollection *
GroupCollection::getEntireGroups(void)
{
    return entireGroups;
}

void GroupCollection::createEntireGroups(void)
{
    entireGroups = new GroupCollection();
}

```

MemberRegistrationUI.h

```

// Class : MemberRegistrationUI
// Description : 회원가입을 위한 boundary 클래스
// 작성자: 김상준
// 작성 날짜: 2017.05.28 12:00
// 설명 : class MemberRegistration의 정의(.h)

#ifndef MEMBERREGISTRATIONUI_H
#define MEMBERREGISTRATIONUI_H

#include <string>
#include "MemberRegistration.h"
using namespace std;

class MemberRegistration;

class MemberRegistrationUI
{
private:
    bool m_isMemberRegistration;
    MemberRegistration* m_MemberRegistrationControl;

public:
    //Function : MemberRegistrationUI(MemberRegistration*
    * control)
    //Description : MemberRegistrationUI 생성자,
    parameter로 받은 control을 멤버변수
    m_MemberRegistrationControl에 할당
    //Parameter : MemberRegistration * control
    //Return Value : MemberRegistration
    MemberRegistrationUI(MemberRegistration * control);

    //Function : void
    requestNewMemberRegistration(string name, string
    personalNumber, string address, string ID, string
    password)
    //Description : 회원가입 요청을 받기 위한 함수
    //Parameter : string name, string personalNumber,
    string address, string ID, string password
    //Return Value : void
    void requestNewMemberRegistration(string name,
    string personalNumber, string address, string ID,
    string password);

};

#endif

```

MemberRegistrationUI.cpp

```
// 작성자 : 김상준  
// 작성 날짜 : 2017.05.28  
// 설명 : MemberRegistrationUI(MemberRegistration *  
control)  
//     requestNewMemberRegistration(string name,  
string personalNumber, string address, string ID,  
string password)  
  
#include <iostream>  
#include "MemberRegistrationUI.h"  
#include <string>  
  
extern FILE* out_fp;  
  
MemberRegistrationUI::MemberRegistrationUI(MemberRegistration * control)  
{  
    m_MemberRegistrationControl = control;  
}  
  
void  
MemberRegistrationUI::requestNewMemberRegistration(s  
tring name, string personalNumber, string address,  
string ID, string password)  
{  
    fprintf(out_fp, "1.1. 회원가입\n");  
    printf("1.1. 회원가입\n");  
    if (m_MemberRegistrationControl->  
newMemberRegistration(name, personalNumber,  
address, ID, password))  
    {  
        fprintf(out_fp, "> %s %s %s %s %s\n",  
name.c_str(), personalNumber.c_str(),  
address.c_str(), ID.c_str(), password.c_str());  
        printf("> %s %s %s %s\n", name.c_str(),  
personalNumber.c_str(), address.c_str(), ID.c_str(),  
password.c_str());  
    }  
    else  
    {  
        fprintf(out_fp, "> 중복된 ID가 존재합니다.\n");  
        printf("> 중복된 ID가 존재합니다.\n");  
    }  
}
```

MemberRegistration.h

```
// Class : MemberRegistration  
// Description : 회원가입을 위한 control 클래스  
// 작성자: 김상준  
// 작성 날짜: 2017.05.29 12:00  
// 설명 : class MemberRegistrationUI의 정의(.h)  
  
#ifndef MEMBERREGISTRATION_H  
#define MEMBERREGISTRATION_H  
  
#include <string>  
#include "MemberCollection.h"  
#include "MemberRegistrationUI.h"  
using namespace std;  
  
class MemberRegistrationUI;  
  
class MemberRegistration  
{  
private:  
    MemberRegistrationUI* m_MemberRegistrationUI;  
public:  
    //Function : MemberRegistration(MemberRegistration**  
    UI)  
    //Description : MemberRegistration 생성자, parameter  
    //로 받은 UI 를 멤버변수  
    // m_MemberRegistrationUI 에 할당  
    //Parameter : MemberRegistration ** UI  
    //Return Value : MemberRegistration  
    MemberRegistration(MemberRegistrationUI** UI);  
    ~MemberRegistration();  
  
    // Function : bool isValidRegistration(string ID)  
    // Description : 회원가입 요청한 ID가 유효한 아이디  
    //인지 확인하는 함수  
    // Parameter : string ID  
    // Return Value :  
    // 1) false : 해당하는 ID의 멤버가 이미 존재 할 경우  
    // 2) true : 해당하는 ID의 멤버가 존재하지 않는 경우  
    bool isValidRegistration(string ID);  
  
    //Function : bool newMemberRegistration(string name,  
    string personalNumber, string address, string ID,  
    string password)  
    //Description : 회원가입을 위한 함수  
    //Parameter : string name, string personalNumber,  
    string address, string ID, string password  
    //Return Value :  
    // 1) true : 회원 가입 성공 시  
    // 2) false : 회원 가입 실패 시  
    bool newMemberRegistration(string name, string  
personalNumber, string address, string ID, string  
password);  
};
```

```
#endif
```

MemberRegistration.cpp

```
// 작성자 : 김상준  
// 작성 날짜 : 2017.05.27  
// 설명 : 회원가입 class 구현 부분  
//     isvalidRegistration()  
//     newMemberRegistration()  
//     MemberRegistration()  
//     isvalidRegistration() 정의
```

```
#include "MemberRegistrationUI.h"
```

```
MemberRegistration::MemberRegistration(MemberRegistrationUI** UI)  
{  
    m_MemberRegistrationUI = new  
    MemberRegistrationUI(this);  
    (*UI) = m_MemberRegistrationUI;  
}
```

```
MemberRegistration::~MemberRegistration()  
{  
    delete m_MemberRegistrationUI;  
}
```

```
bool MemberRegistration::isvalidRegistration(string  
ID)  
{  
    Member* member =  
    MemberCollection::getEntireMembers()-  
    >findMember(ID);  
    if (member == nullptr) // findMember로 ID를  
    검색했을 때 반환값이 nullptr인 경우는 멤버콜렉션에  
    해당 ID가 없다는 뜻이며 해당 ID로 계정을 생성할 수  
    있는것을 의미  
        return true;  
    else  
        return false;  
}
```

```
bool  
MemberRegistration::newMemberRegistration(string  
name, string personalNumber, string address, string  
ID, string password)  
{  
    if (isvalidRegistration(ID))
```

```
    {//회원가입 성공  
        MemberCollection::getEntireMembers()-  
        >createMember(name, personalNumber, address, ID,  
        password);  
        return true;  
    }  
    else  
    {//회원가입 실패  
        return false;  
    }  
}
```

LoginUI.h

```
// Class : LoginUI
// Description: 로그인을 위한 boundary 클래스
// 작성자 : 김상준
// 작성 날짜 : 2017.05.29 13:00
// 설명 : class LoginUI의 정의(.h)

#ifndef LOGINUI_H
#define LOGINUI_H

#include <string>
#include "Login.h"
using namespace std;

class Login;

class LoginUI
{
private:
    Login *m_loginControl;
public:
    //Function : bool requestAttemptLogin(string ID,
    string password)
    //Description : 입력받은 ID, password를 컨트롤에게 넘겨줌.
    //컨트롤이 반환하는 결과에 따라 출력문 출력 및 bool 반환
    //Parameter : string ID, string password
    //Return Value :
    // 1) true : 로그인 성공
    // 2) false : 로그인 실패
    bool requestAttemptLogin(string ID, string password);
    //Function : LoginUI(Login *loginControl)
    //Description : LoginUI 생성자
    //Parameter : Login *loginControl
    //Return Value : LoginUI
    LoginUI(Login * loginControl);
};

#endif
```

LoginUI.cpp

```
// 작성자 : 김상준
// 작성 날짜 : 2017.05.29
// 설명 : requestAttemptLogin(string ID, string password)
// 수정자 : 이한솔
// 수정 시간 : 2017.05.30 02:20
// 수정 이유 : LOGIN 성공 여부에 관련한 flag 추가

#include <iostream>
#include "LoginUI.h"
#include <string>

extern FILE* out_fp;

LoginUI::LoginUI(Login * loginControl)
{
    m_loginControl = loginControl;
}

bool LoginUI::requestAttemptLogin(string ID, string password)
{
    fprintf(out_fp, "2.1. 로그인\n");
    printf("2.1. 로그인\n");
    int result = m_loginControl->attemptLogin(ID,
password);

    if (result == LOGIN_INVALIDPASSWORD)
    {
        fprintf(out_fp, "> 비밀번호가 맞지 않습니다.\n");
        printf("> 비밀번호가 맞지 않습니다.\n");
        return false;
    }
    else if (result == LOGIN_INVALIDID)
    {
        fprintf(out_fp, "> 존재하지 않는 ID입니다.\n");
        printf("> 존재하지 않는 ID입니다.\n");
        return false;
    }
    else
    {
        fprintf(out_fp, "> %s %s\n", ID.c_str(),
password.c_str());
        printf("> %s %s\n", ID.c_str(),
password.c_str());
        return true;
    }
}
```

Login.h

```
// Class : Login
// Description: 로그인을 위한 control 클래스
// 작성자 : 김상준
// 작성 날짜 : 2017.05.29 13:00
// 설명 : class Login의 정의(.h)

#ifndef LOGIN_H
#define LOGIN_H

#include <string>
#include "MemberCollection.h"
#include "Member.h"

#define LOGIN_SUCCESSFUL 1
#define LOGIN_INVALIDPASSWORD 0
#define LOGIN_INVALIDID -1

using namespace std;

class LoginUI;

class Login
{
private:
    LoginUI *m_loginUI;
public:
    //Function : int attemptLogin(string ID, string password)
    //Description : 입력받은 ID, password를 활용하여
    //로그인.
    // 해당하는 ID가 이미 가입한 그룹이 없고, 그룹
    //이름이 중복되지 않을 경우 그룹 생성.
    //Parameter :string ID, string password
    //Return Value :
    //1) LOGIN_SUCCESSFUL : 로그인이 정상적으로
    //진행되었음
    //2) LOGIN_INVALIDPASSWORD : 비밀번호 틀림. 로그인
    //실패
    //3) LOGIN_INVALIDID : 해당하는 ID가 존재하지 않음.
    //로그인 실패
    int attemptLogin(string ID, string password);
    //Function : Login(LoginUI** loginUI)
    //Description : 로그인 컨트롤 생성자. 로그인 UI도
    //생성함
    //Parameter : LoginUI** loginUI
    //Return Value : Login
    Login(LoginUI** loginUI);
    //Function : ~Login()
    //Description : 로그인 컨트롤 소멸자. 로그인 UI도
    delete함
    //Parameter : 없음
    //Return Value : 없음
    ~Login();
};

#endif
```

```
#endif
```

Login.cpp

```
// 작성자 : 김상준
// 작성 날짜 : 2017.05.29
// 설명 : Login(LoginUI** pointerFromMain))
// attemptMember(string ID)
// checkLogin(string ID, string password)
// 수정자 : 이한솔
// 수정 시간 : 2017.05.30 02:20
// 수정 이유 : LOGIN 성공 여부에 관련한 flag 추가

#include "LoginUI.h"

extern FILE* out_fp;

int Login::attemptLogin(string ID, string password)
{
    Member* attemptedMember =
    MemberCollection::getEntireMembers()-
    >findMember(ID);

    if (attemptedMember == nullptr) // findMember로
    ID를 검색했을 때 반환값이 nullptr인 경우는
    멤버콜렉션에 해당 ID가 없다는 뜻
    {//로그인 실패. 아이디 존재하지 않음
        return LOGIN_INVALIDID;
    }
    else if (attemptedMember->checkLogin(password))
    {//로그인 성공
        attemptedMember->setLoginState(STATE_LOGIN);
        return LOGIN_SUCCESSFUL;
    }
    else
    {//로그인 실패. 비밀번호 안 맞음
        return LOGIN_INVALIDPASSWORD;
    }
}

Login::Login(LoginUI** loginUI)
{
    m_loginUI = new LoginUI(this);
    (*loginUI) = m_loginUI;
}

Login::~Login()
{
    delete m_loginUI;
}
```

LogoutUI.h

```
// Class : LogoutUI
// Description : Logout 을 위한 boundary 클래스
// 작성자 : 김상준
// 작성 날짜 : 2017.05.29 12:00
// 설명 : class LogoutUI의 정의(.h)

#ifndef LOGOUTUI_H
#define LOGOUTUI_H

#include <string>
#include "Logout.h"
using namespace std;

class Logout;

class LogoutUI
{
private:
Logout* m_logoutControl;
public:
//Function : LogoutUI(Logout* control)
//Description : LogoutUI 생성자, parameter로 받은
control 을 멤버변수 m_logoutControl 에 할당
//Parameter : Logout* control
//Return Value : LogoutUI
LogoutUI(Logout* control);

//Function : void requestLogout(string userID)
//Description : logout 요청을 받기 위한 함수
//Parameter : string userID
//Return Value : void
void requestLogout(string userID);
};

#endif
```

LogoutUI.cpp

```
// 작성자 : 김상준
// 작성 날짜 : 2017.05.28
// 설명 : requestLogout(std::string userID)
//         LogoutUI(Logout* control)

#include <iostream>
#include "LogoutUI.h"
#include <string>

extern FILE* out_fp;

void LogoutUI::requestLogout(string userID)
{
    m_logoutControl->logout(userID);
    fprintf(out_fp, "2.2. 로그아웃\n");
    fprintf(out_fp, "> %s\n", userID.c_str());

    printf("2.2. 로그아웃\n");
    printf("> %s\n", userID.c_str());
}

LogoutUI::LogoutUI(Logout* control)
{
    m_logoutControl = control;
}
```

Logout.h

```
// Class : Logout
// Description : Logout 을 위한 control 클래스
// 작성자 : 김상준
// 작성 날짜 : 2017.05.29 12:00
// 설명 : class Logout의 정의(.h)

#ifndef LOGOUT_H
#define LOGOUT_H

#include <string>
#include "MemberCollection.h"

using namespace std;

class LogoutUI;

class Logout
{
private:
    LogoutUI *m_logoutUI;
public:
    //Function : Logout(LogoutUI** logoutUI)
    //Description : Logout 생성자, parameter로 받은
    //logoutUI를 멤버변수
    // m_logoutUI 에 할당
    //Parameter : LogoutUI** logoutUI
    //Return Value : Logout
    Logout(LogoutUI** logoutUI);
    ~Logout();

    //Function : void logout(string userID)
    //Description : 로그아웃을 위한 함수
    //Parameter : string userID
    //Return Value : void
    void logout(string userID);
};

#endif // !LOGOUT_H
```

Logout.cpp

```
// 작성자 : 김상준
// 작성 날짜 : 2017.05.28
// Logout 클래스 구현

#include "LogoutUI.h"

extern FILE* out_fp;

Logout::Logout(LogoutUI** logoutUI)
{
    m_logoutUI = new LogoutUI(this);
    (*logoutUI) = m_logoutUI;
}

Logout::~Logout()
{
    delete m_logoutUI;
}

void Logout::logout(string userID)
{
    Member* member =
    MemberCollection::getEntireMembers()-
    >findMember(userID);
    member->setLoginState(STATE_LOGOUT);
}
```

LeaveMembershipUI.h

```
// Class : LeaveMembershipUI  
// Description : 회원탈퇴를 위한 boundary 클래스  
// 작성자 : 김상준  
// 작성 날짜 : 2017.05.29 12:00  
// 설명 : class LeaveMembershipUI의 정의(.h)  
  
#ifndef LEAVEMBERSHIPUI_H  
#define LEAVEMBERSHIPUI_H  
  
#include <string>  
#include "leaveMembership.h"  
using namespace std;  
  
class LeaveMembership;  
  
class LeaveMembershipUI  
{  
private:  
    LeaveMembership *m_leaveMembershipControl;  
public:  
    //Function : bool requestLeaveMembership(string user ID)  
    //Description : 입력받은 ID를 control에게 넘겨주어 회원탈퇴 요청  
    //Parameter : string user ID  
    //Return Value :  
    //1) true : LEAVEMBERSHIP_SUCCESSFUL. 회원탈퇴 성공  
    //2) false: LEAVEMBERSHIP_FAIL. 회원탈퇴 실패(그룹을 생성한 회원인 경우)  
    bool requestLeaveMembership(string user ID);  
  
    //Function : LeaveMembershipUI(LeaveMembership * leaveMembershipControl)  
    //Description : LeaveMembershipUI 생성자  
    //Parameter : LeaveMembership * leaveMembershipControl  
    //Return Value : LeaveMembershipUI  
    LeaveMembershipUI(LeaveMembership * leaveMembershipControl);  
};  
  
#endif
```

LeaveMembershipUI.cpp

```
// 작성자 : 김상준  
// 작성 날짜 : 2017.05.29  
// 설명 : requestLeaveMembership(string user ID)  
  
#include <iostream>  
#include "LeaveMembershipUI.h"  
#include <string>  
  
extern FILE* out_fp;  
  
bool  
LeaveMembershipUI::requestLeaveMembership(string user ID)  
{  
    fprintf(out_fp, "1.2. 회원탈퇴\n");  
    printf("1.2. 회원탈퇴\n");  
    if (m_leaveMembershipControl->leaveMembership(user ID))  
    {  
        fprintf(out_fp, "> %s\n", user ID.c_str());  
        printf("> %s\n", user ID.c_str());  
        return LEAVEMBERSHIP_SUCCESSFUL;  
    }  
    else  
    {  
        //그룹장이므로 회원탈퇴 못 함  
        fprintf(out_fp, "> 그룹장은 회원탈퇴 할 수 없습니다.\n");  
        printf("> 그룹장은 회원탈퇴 할 수 없습니다.\n");  
        return LEAVEMBERSHIP_FAIL;  
    }  
}  
  
LeaveMembershipUI::LeaveMembershipUI(LeaveMembership * leaveMembershipControl)  
{  
    m_leaveMembershipControl = leaveMembershipControl;  
}
```

LeaveMembership.h

```
// Class : LeaveMembership
// Description: 회원탈퇴를 위한 control 클래스
// 작성자 : 김상준
// 작성 날짜 : 2017.05.29 12:00
// 설명 : class LeaveMembership의 정의(.h)

#ifndef LEAVEMBERSHIP_H
#define LEAVEMBERSHIP_H

#include <string>
#include "MemberCollection.h"
#include "GroupCollection.h"
#include "Member.h"

#define LEAVEMBERSHIP_SUCCESSFUL true
#define LEAVEMBERSHIP_FAIL false

using namespace std;

class LeaveMembershipUI;
class LeaveMembership
{
private:
    LeaveMembershipUI *m_memberCollectionUI;
public:
    //Function : bool leaveMembership(string userID)
    //Description : 입력받은 ID를 활용하여 회원탈퇴.
    // 해당하는 ID가 그룹을 생성한 회원이 아닌 경우
    //회원탈퇴.
    //Parameter : string userID
    //Return Value :
    //1) LEAVEMBERSHIP_SUCCESSFUL : 회원탈퇴 성공
    //2) LEAVEMBERSHIP_FAIL : user가 그룹을 생성한
    //회원(회원탈퇴 실패)
    bool leaveMembership(string userID);
    //Function : LeaveMembership(LeaveMembershipUI
    **leaveMembershipUI)
    //Description : 회원탈퇴 컨트를 생성자. 회원탈퇴
    //UI도 생성함
    //Parameter : LeaveMembershipUI **leaveMembershipUI
    //Return Value : LeaveMembership
    LeaveMembership(LeaveMembershipUI**
    leaveMembershipUI);
    //Function : ~LeaveMembership()
    //Description : 회원탈퇴 컨트를 소멸자. 회원탈퇴
    //UI도 delete함
    //Parameter : 없음
    //Return Value : 없음
    ~LeaveMembership();
};

#endif
```

LeaveMembership.cpp

```
// 작성자 : 김상준
// 작성 날짜 : 2017.05.29
// 설명 : leaveMembership(string userID)

#include "LeaveMembershipUI.h"

bool LeaveMembership::leaveMembership(string userID)
{
    Member *user =
    MemberCollection::getEntireMembers()-
    >findMember(userID);
    if (user->checkJoinedGroup())
    {
        string groupName = user->getJoinedGroupName();
        Group* group =
        GroupCollection::getEntireGroups()-
        >findGroup(groupName);
        if (userID != group->getGroupCreatorID())
            //그룹장이 아니니까 회원탈퇴.
            MemberCollection::getEntireMembers()-
            >deleteMember(userID);
            delete user;
            return LEAVEMBERSHIP_SUCCESSFUL;
        }
        else
            //그룹장이므로 회원탈퇴 못 함
            return LEAVEMBERSHIP_FAIL;
    }
    else
        //가입 그룹 없으니까 그냥 탈퇴
        MemberCollection::getEntireMembers()-
        >deleteMember(userID);
        delete user;
        return LEAVEMBERSHIP_SUCCESSFUL;
    }
}

LeaveMembership::LeaveMembership(LeaveMembershipUI
** leaveMembershipUI)
{
    m_memberCollectionUI = new
    LeaveMembershipUI(this);
    (*leaveMembershipUI) = m_memberCollectionUI;
}
LeaveMembership::~LeaveMembership()
{
    delete m_memberCollectionUI;
}
```

ShowEntireGroupUI.h

```
// Class : ShowEntireGroupUI
// Description : 전체그룹 조회를 위한 boundary
// 클래스
// 작성자: 심민우
// 작성 날짜: 2017.05.28 12:00
// 설명 : class ShowEntireGroupUI의 정의(.h)

#ifndef SHOWENTIREGROUPUI_H_
#define SHOWENTIREGROUPUI_H_

#include "ShowEntireGroup.h"
#include "GroupCollection.h"
#include <string>
using namespace std;

class ShowEntireGroup;

class ShowEntireGroupUI {
private:
    ShowEntireGroup* m_ShowEntireGroupControl;
public:
    //Function : ShowEntireGroupUI(ShowEntireGroup* control)
    //Description : ShowEntireGroupUI 생성자, parameter로 받은 control을 멤버변수
    //m_ShowEntireGroupControl에 할당
    //Parameter : ShowEntireGroup* control
    //Return Value : ShowEntireGroupUI
    ShowEntireGroupUI(ShowEntireGroup* control);

    //Function : void showEntireGroup()
    //Description : 전체그룹 출력을 위한 함수
    //Parameter : string groupName
    //Return Value : void
    void showEntireGroup(string groupName);

    //Function : void selectGroup(string userID, string groupName)
    //Description : 그룹에 가입하기 위한 함수
    //Parameter : string userID, string groupName
    //Return Value : void
    void selectGroup(string userID, string groupName);

    //Function : void requestShowEntireGroup(void)
    //Description : 전체그룹조회 요청을 받기 위한 함수
    //Parameter : void
    //Return Value : void
    void requestShowEntireGroup(void);
};

#endif
```

ShowEntireGroupUI.cpp

```
//작성자: 심민우
//작성 날짜: 2017.05.28 12:00
//설명 : class ShowEntireGroupUI operation 구현

#include "ShowEntireGroup.h"
#include "GroupCollection.h"
#include "Group.h"
#include "MemberCollection.h"
#include "Member.h"
#define JOINGROUP_SUCCESSFUL 1
#define JOINGROUPFAILED_NOGROUP -1
#define JOINGROUPFAILED_ALREADYJOINED 0

using namespace std;

extern FILE* out_fp;

void ShowEntireGroupUI::showEntireGroup(string groupName)
{
    fprintf(out_fp, "> %s\n", groupName.c_str());
    printf("> %s\n", groupName.c_str());
}

ShowEntireGroupUI::ShowEntireGroupUI(ShowEntireGroup* showEntireGroupControl)
{
    m_ShowEntireGroupControl = showEntireGroupControl;
}

void ShowEntireGroupUI::requestShowEntireGroup(void)
{
    fprintf(out_fp, "5.1. 전체그룹 조회\n");
    printf("5.1. 전체그룹 조회\n");
    m_ShowEntireGroupControl->showEntireGroup();
}

void ShowEntireGroupUI::selectGroup(string userID,
                                    string groupName)
{
    fprintf(out_fp, "5.2. 그룹 가입\n");
    printf("5.2. 그룹 가입\n");
    int result = m_ShowEntireGroupControl-
    >joinGroup(userID, groupName);
    if (result == JOINGROUP_SUCCESSFUL)
    {
        fprintf(out_fp, "> %s\n", groupName.c_str());
        printf("> %s\n", groupName.c_str());
    }
    else if(result == JOINGROUPFAILED_ALREADYJOINED)
```

```

{
    fprintf(out_fp, "> 이미 가입된 그룹이
    있습니다.\n");
    printf("> 이미 가입된 그룹이 있습니다.\n");
}
else
{ //JOINGROUPFAILED_NOGROUP
    fprintf(out_fp, "> 그룹이 존재하지
    않습니다.\n");
    printf("> 그룹이 존재하지 않습니다.\n");
}
}

```

ShowEntireGroup.h

```

// Class : ShowEntireGroup
// Description : 전체그룹 조회를 위한 control 클래스
// 작성자: 심민우
// 작성 날짜: 2017.05.28 12:00
// 설명 : class ShowEntireGroup의 정의( .h)

#ifndef SHOWENTIREGROUP_H_
#define SHOWENTIREGROUP_H_

#include "ShowEntireGroupUI.h"
#include "GroupCollection.h"
#include <string>

#define JOINGROUP_SUCCESSFUL 1
#define JOINGROUPFAILED_NOGROUP -1
#define JOINGROUPFAILED_ALREADYJOINED 0

using namespace std;
class ShowEntireGroupUI;

class ShowEntireGroup {
private:
    ShowEntireGroupUI *m_ShowEntireGroupUI;
public:
    //Function : ShowEntireGroup(ShowEntireGroup** UI)
    //Description : ShowEntireGroup 생성자, parameter로
    //받은 UI를 멤버변수
    // m_ShowEntireGroupUI에 할당
    //Parameter : ShowEntireGroup** UI
    //Return Value : ShowEntireGroup
    ShowEntireGroup(ShowEntireGroupUI **UI);
    ~ShowEntireGroup();
    //Function : int joinGroup(string userID, string
    //groupName)
    //Description : 그룹 가입을 위한 함수
    //Parameter : string userID, string groupName
    //Return Value : int - 그룹 가입 성공시
    //SUCCESSFULJOINGROUP 반환 가입을 원하는 그룹이
    //존재하지 않을 시
    //      JOINGROUPFAILED_NOGROUP 반환 이미 가입한
    //그룹이 있을 시 JOINGROUPFAILED_ALREADYJOINED 반환
    int joinGroup(string userID, string groupName);

    void showEntireGroup();

};

#endif

```

ShowEntireGroup.cpp

```

//작성자: 심민우
//작성 날짜: 2017.05.28 12:00
//설명 : class ShowEntireGroup operation 구현
//수정자 : 이한솔
//수정 시간 : 2017.05.28 14:20
//수정 이유 : 출력부분 수정(파일 입출력 및 출력 내용
추가)

//수정자 : 심민우
//수정 시간 : 2017.05.30 00:10
//수정 이유 : joinGroup bool 조건 수정

#include "ShowEntireGroup.h"
#include "GroupCollection.h"
#include "Group.h"
#include "MemberCollection.h"
#include "Member.h"
#include <string>

using namespace std;

extern FILE* out_fp;

ShowEntireGroup::ShowEntireGroup(ShowEntireGroupUI
**UI)
{
    m_showEntireGroupUI = new ShowEntireGroupUI(this);
    (*UI) = m_showEntireGroupUI;
}

ShowEntireGroup::~ShowEntireGroup()
{
    delete m_showEntireGroupUI;
}

int ShowEntireGroup::joinGroup(string userID, string
groupName)
{
    // 그룹 가입 부분
    // parameter로 받은 userID로
    MemberCollection::getEntireMembers() Collection
    탐색 후
    // 그 멤버가 가입된 그룹이 있는지 checkJoinedGroup
    으로 확인
    // 가입한 그룹이 있을 경우 에러 메시지 출력하고
    // 없을 경우 user에 joinedGroupName 을 세팅해주고
    결과 출력

    Member* currentMember;
    bool isJoinedGroup;
    currentMember =
    MemberCollection::getEntireMembers()-
    >findMember(userID);

    isJoinedGroup = currentMember->checkJoinedGroup();
}

```

```

Group* group = GroupCollection::getEntireGroups()-
>findGroup(groupName);

if (isJoinedGroup == false && group !=NULL)
{
    currentMember->setJoinedGroupName(groupName);
    return JOINGROUP_SUCCESSFUL;
}
else if(group == NULL)
{
    return JOINGROUPFAILED_NOGROUP;
}
else
{
    return JOINGROUPFAILED_ALREADYJOINED;
}

}

void ShowEntireGroup::showEntireGroup()
{
    int i;
    string currentGroupName;
    Group *currentGroup;
    // 전체그룹 조회 부분
    // GroupCollection::getEntireGroups() Collection
    List를 조회하여
    // 그룹이 존재하면 currentGroup 으로 받아와
    groupName 호출 후 출력

    for (i = 0; ; i++)
    {
        currentGroup =
        GroupCollection::getEntireGroups()->findNext(i);

        if (currentGroup != NULL)
        {
            currentGroupName = currentGroup-
            >getGroupName();
            m_showEntireGroupUI-
            >showEntireGroup(currentGroupName);
        }
        else
        {
            break;
        }
    }
}

```

ShowJoinedGroupUI.h

```
// Class : ShowJoinedGroupUI  
// Description : 가입그룹조회를 위한 boundary 클래스  
// 작성자 : 심민우  
// 작성 날짜 : 2017.05.27 20:00  
// 설명 : ShowJoinedGroup Class의 정의(.h)
```

```
#ifndef SHOWJOINEDGROUPUI_H_  
#define SHOWJOINEDGROUPUI_H_  
  
#include <string>  
#include "ShowJoinedGroup.h"  
using namespace std;  
class ShowJoinedGroup;  
  
class ShowJoinedGroupUI  
{  
private:  
    string m_joinedGroupName;  
    ShowJoinedGroup* m_ShowJoinedGroupControl;  
public:  
    //Function : ShowJoinedGroupUI(ShowJoinedGroup  
    *control)  
    //Description : ShowJoinedGroupUI 생성자 parameter로  
    받아온 control 을 멤버변수  
    //    m_ShowJoinedGroupControl 에 할당  
    //Parameter : ShowJoinedGroup *control  
    //Return Value : ShowJoinedGroup  
    ShowJoinedGroupUI(ShowJoinedGroup *control);  
  
    //Function : void showJoinedGroup(string  
    joinedGroupName)  
    //Description : 가입그룹조회 성공시 가입그룹의  
    이름을 출력하는 함수  
    //Parameter : string joinedGroupName  
    //Return Value : void  
    void showJoinedGroup(string joinedGroupName);  
  
    //Function : void showJoinedGroupFail()  
    //Description : 가입한 그룹이 없을시 가입한 그룹이  
    없다는 정보를 출력하는 함수  
    //Parameter : void  
    //Return Value : void  
    void showJoinedGroupFail();  
  
    //Function : void requestLeaveGroup(string userID)  
    //Description : 그룹 탈퇴 요청을 받는 함수  
    //Parameter : string userID  
    //Return Value : void  
    void requestLeaveGroup(string userID);  
  
    //Function : void requestShowJoinedGroup(string  
    userID)
```

```
//Description : 가입 그룹 조회 요청을 받는 함수  
//Parameter : string userID  
//Return Value : void  
void requestShowJoinedGroup(string userID);
```

```
//Function : void printJoinedGroup(string groupName)  
//Description : 가입한 그룹을 출력하는 함수  
//Parameter : string userID  
//return Value : void  
void printJoinedGroup(string groupName);  
};  
#endif
```

ShowJoinedGroupUI.cpp

```
// 작성자 : 심민우  
// 작성 날짜 : 2017.05.27 20:00  
// 설명 : Member Class operation 정의 부분  
//         requestShowJoinedGroup, 생성자 operation  
구현  
// 수정자 : 이한솔  
// 수정 시간 : 2017.05.28 14:30  
// 수정 이유 : 파일 입출력을 위한 printf문 추가
```

```
#include "ShowJoinedGroupUI.h"  
#include <iostream>  
  
extern FILE* out_fp;  
  
void ShowJoinedGroupUI::showJoinedGroup(string  
joinedGroupName)  
{  
    fprintf(out_fp, "5.4. 가입그룹 조회\n");  
    fprintf(out_fp, "> %s\n",  
joinedGroupName.c_str());  
  
    printf("5.4. 가입그룹 조회\n");  
    printf("> %s\n", joinedGroupName.c_str());  
}  
  
void ShowJoinedGroupUI::showJoinedGroupFail()  
{
```

```

        fprintf(out_fp, "5.4. 가입그룹 조회\n");
        fprintf(out_fp, "> 가입그룹이 없습니다.\n");

    printf("5.4. 가입그룹 조회\n");
    printf("> 가입그룹이 없습니다.\n");
}

void ShowJoinedGroupUI::requestLeaveGroup(string userID)
{
    fprintf(out_fp, "5.5. 그룹 탈퇴\n");
    printf("5.5. 그룹 탈퇴\n");
    string groupName;
    int result;
    result = m_ShowJoinedGroupControl-
>leaveGroup(userID);
    if(result ==
LEAVEGROUPFAILED_LEAVEISGROUPCREATOR)
    {
        fprintf(out_fp, "> 그룹장은 탈퇴할수
없습니다.\n");
        printf("> 그룹장은 탈퇴할수 없습니다.\n");
    }
    else if(result == LEAVEGROUPFAILED_NOGROUP)
    {
        fprintf(out_fp, "> 그룹에 가입되어 있지
않습니다.\n");
        printf("> 그룹에 가입되어 있지 않습니다.\n");
    }
    else
    { //LEAVEGROUP_SUCCESSFUL
    }
}

void
ShowJoinedGroupUI::requestShowJoinedGroup(string userID)
{
    m_ShowJoinedGroupControl->showJoinedGroup(userID);
}

void ShowJoinedGroupUI::printJoinedGroup(string groupName)
{
    fprintf(out_fp, "> %s\n", groupName.c_str());
    printf("> %s\n", groupName.c_str());
}

ShowJoinedGroupUI::ShowJoinedGroupUI(ShowJoinedGroup
*control)
{
    m_ShowJoinedGroupControl = control;
}

```

ShowJoinedGroup.h

```

// Class : ShowJoinedGroup
// Description : 가입그룹조회를 위한 control 클래스
// 작성자 : 심민우
// 작성 날짜 : 2017.05.27 20:00
// 설명 : ShowJoinedGroup Class의 정의(.h)

using namespace std;

#ifndef SHOWJOINEDGROUP_H_
#define SHOWJOINEDGROUP_H_

#include "ShowJoinedGroupUI.h"
#include <string>
#include "Member.h"
#include "MemberCollection.h"
#include "Group.h"
#include "GroupCollection.h"

#define LEAVEGROUP_SUCCESSFUL 1
#define LEAVEGROUPFAILED_LEAVEISGROUPCREATOR -1
#define LEAVEGROUPFAILED_NOGROUP 0

class ShowJoinedGroupUI;

class ShowJoinedGroup {
private:
    bool m_checkJoinedGroup;
    Member* m_currentMember;
    MemberCollection* m_memberCollection;
    Group* m_joinedGroup;
    GroupCollection* m_groupCollection;
    ShowJoinedGroupUI *m_ShowJoinedGroupUI;
public:
    //Function : ShowJoinedGroup(ShowJoinedGroupUI** UI)
    //Description : ShowJoinedGroup 생성자 ,
    parameter로 받은 UI 를 멤버변수
    // m_ShowJoinedGroupUI 에 할당
    //Parameter : ShowJoinedGroupUI** UI
    //Return Value : ShowJoinedGroup
    ShowJoinedGroup(ShowJoinedGroupUI** UI);
    ~ShowJoinedGroup();
    //Function : void showJoinedGroup(string)
    //Description : 가입 그룹 조회 출력을 위한 함수
    //Parameter : string userID
    //Return Value : void
    void showJoinedGroup(string userID);

    //Function : string leaveGroup(string userID, int*
    status)
    //Description : 그룹 탈퇴를 위한 함수
    //Parameter : string userID
}

```

```

//Return Value :
// 1) LEAVEGROUP_SUCCESSFUL : 가입한 그룹이 있고
// 그룹 생성자가 아닌경우
// 2) LEAVEGROUPFAILED_LEAVEISGROUPCREATOR :
//가입한 그룹의 그룹 생성자인 경우
// 3) LEAVEGROUPFAILED_NOGROUP : 가입한 그룹이
//없는 경우
int leaveGroup(string userID);

};

#endif

```

ShowJoinedGroup.cpp

```

// 작성자 : 심민우
// 작성 날짜 : 2017.05.27 20:00
// 설명 : 가입그룹 조회 ShowJoinedGroup control
// 수정자 : 이한솔
// 수정 시간 : 2017.05.28 14:30
// 수정 이유 : 파일 입출력을 위한 print문 추가

#include "ShowJoinedGroupUI.h"

using namespace std;

extern FILE* out_fp;

void ShowJoinedGroup::showJoinedGroup(string userID)
{
    // 가입 그룹 조회 부분
    // parameter로 받은 userID로
    MemberCollection::getEntireMembers() 탐색 후
    // 해당 멤버가 가입한 그룹이 있는지 확인

    Member *currentMember;
    string joinedGroup;
    string joinedGroupName;
    currentMember =
    MemberCollection::getEntireMembers()-
    >findMember(userID);

```

```

    m_checkJoinedGroup = currentMember-
    >checkJoinedGroup();

    // 가입한 그룹이 있을경우
    // 가입한 그룹의 이름을 가져와서
    GroupCollection::getEntireGroups() Collection List
    탐색후
    // 해당 그룹의 이름을 가져와서 출력해줌
    // 가입한 그룹이 없을 경우
    // 에러 메시지 출력을 위한 showJoinedGroupFail
    call
    if (m_checkJoinedGroup == true)
    {

        joinedGroup = currentMember-
        >getJoinedGroupName();
        m_joinedGroup =
        GroupCollection::getEntireGroups()-
        >findGroup(joinedGroup);
        joinedGroupName = m_joinedGroup->getGroupName();
        m_ShowJoinedGroupUI-
        >showJoinedGroup(joinedGroupName);

    }
    else
    {
        m_ShowJoinedGroupUI->showJoinedGroupFail();
    }
}

int ShowJoinedGroup::leaveGroup(string userID)
{

    Member *currentMember;
    string joinedGroup;
    string groupCreator;
    string fail = "";
    currentMember =
    MemberCollection::getEntireMembers()-
    >findMember(userID);

    m_checkJoinedGroup = currentMember-
    >checkJoinedGroup();

    if (m_checkJoinedGroup)
    {
        joinedGroup = currentMember-
        >getJoinedGroupName();
        m_joinedGroup =
        GroupCollection::getEntireGroups()-
        >findGroup(joinedGroup);

        groupCreator = m_joinedGroup-
        >getGroupCreatorID();
    }
}

```

```

if (groupCreator == userID)
{
    return LEAVEGROUPFAILED_LEAVEVISGROUPCREATOR;
}
else
{
    joinedGroup = currentMember-
>getJoinedGroupName();
    currentMember->unsetJoinedGroup();
    m_ShowJoinedGroupUI-
>printJoinedGroup(joinedGroup);
    return LEAVEGROUP_SUCCESSFUL;
}
else
{
    return LEAVEGROUPFAILED_NOGROUP;
}
}

ShowJoinedGroup::ShowJoinedGroup(ShowJoinedGroupUI**
showJoinedGroupUI)
{
    m_ShowJoinedGroupUI = new ShowJoinedGroupUI(this);
    (*showJoinedGroupUI) = m_ShowJoinedGroupUI;
}

ShowJoinedGroup::~ShowJoinedGroup()
{
    delete m_ShowJoinedGroupUI;
}

```

CreateNewGroupUI.h

```

// Class : CreateNewGroupUI
// Description: 그룹 생성을 위한 boundary 클래스
// 작성자 : 심민우
// 작성 날짜 : 2017.05.27 13:00
// 설명 : 그룹 생성 boundary class
// 수정자 : 강석원
// 수정 날짜 : 2017.05.27 15:00
// 수정 이유 : 순환참조를 위한 #ifndef 추가

#ifndef CREATENEWGROUPUI_H
#define CREATENEWGROUPUI_H

#include <string>
#include "CreateNewGroup.h"

using namespace std;

class CreateNewGroup;

class CreateNewGroupUI {
private:
    CreateNewGroup *m_createNewGroupControl;
public:
    //Function : void requestCreateNewGroup(string
    groupName, string userID)
    //Description : 입력받은 그룹 이름, ID를
    control에게 넘겨주어 그룹 생성 실행
    //Parameter : string groupName, string userID
    //Return Value : void
    void requestCreateNewGroup(string groupName,
    string userID);
    //Function : CreateNewGroupUI(CreateNewGroup
    *control)
    //Description : CreateNewGroupUI 생성자
    //Parameter : CreateNewGroup *control
    //Return Value : CreateNewGroupUI
    CreateNewGroupUI(CreateNewGroup *control);
};

#endif

```

CreateNewGroupUI .cpp

```
// 작성자 : 심민우
// 작성 날짜 : 2017.05.27
// 설명 : 그룹 생성 boundary class Operation 정의
부분
//    그룹 생성 요청 함수
//    requestCreateNewGroup 함수 정의
//    CreateNewGroupUI 생성자 정의
// 수정자 : 이한솔
// 수정 시간 : 2017.05.28 14:30
// 수정 이유 : 파일 입출력을 위한 print문 추가

#include <iostream>
#include "CreateNewGroupUI.h"

extern FILE* out_fp;

void CreateNewGroupUI::requestCreateNewGroup(string
groupName, string userID)
{
    fprintf(out_fp, "5.3. 그룹 생성 \n");
    printf("5.3. 그룹 생성 \n");
    int groupCreation = m_createNewGroupControl-
>createNewGroup(groupName, userID);
    // control createNewGroup call
    if (groupCreation == GROUPCREATION_HASJOINEDGROUP)
    { //GROUPCREATION_HASJOINEDGROUP
        fprintf(out_fp, "> 이미 그룹에 가입한
상태입니다.\n");
        printf("> 이미 그룹에 가입한 상태입니다.\n");
    }
    else if (groupCreation ==
GROUPCREATION_DUPLICATEDNAME)
    { //GROUPCREATION_DUPLICATEDNAME
        fprintf(out_fp, "> 같은 이름의 그룹이
존재합니다.\n");
        printf("> 같은 이름의 그룹이 존재합니다.\n");
    }
    else
    { //GROUPCREATION_SUCCESSFUL
        fprintf(out_fp, "> %s \n", groupName.c_str());
        printf("> %s \n", groupName.c_str());
    }
}

CreateNewGroupUI::CreateNewGroupUI(CreateNewGroup
*control)
{
    // 생성자
    // parameter로 받은 control 을 member변수
m_createGroupControl 에 할당
    m_createNewGroupControl = control;
}
```

CreateNewGroup.h

```
// Class : CreateNewGroup
// Description: 그룹 생성을 위한 control 클래스
// 작성자 : 심민우
// 작성 날짜 : 2017.05.27 13:00
// 설명 : 그룹 생성 control class header
// 수정자 : 강석원
// 수정 날짜 : 2017.05.27 15:00
// 수정 이유 : 순환참조를 위한 #ifndef 추가

#ifndef CREATENEWGROUP_H
#define CREATENEWGROUP_H

#include <string>
#include "MemberCollection.h"
#include "GroupCollection.h"
#include "CreateNewGroupUI.h"

#define GROUPCREATION_SUCCESSFUL 1
#define GROUPCREATION_HASJOINEDGROUP 0
#define GROUPCREATION_DUPLICATEDNAME -1

using namespace std;

class CreateNewGroupUI;

class CreateNewGroup
{
private:
    string m_groupCreator;
    string m_newGroupName;
    bool m_isJoinedGroup;
    GroupCollection *m_groupCollection;
    MemberCollection *m_memberCollection;
    Member *m_currentMember;
    CreateNewGroupUI *m_createNewGroupUI;
public:
    //Function : int createNewGroup(string groupName,
string userID)
    //Description : 입력받은 그룹 이름, ID를 활용하여
새로운 그룹 생성.
    // 해당하는 ID가 이미 가입한 그룹이 없고, 그룹
이름이 중복되지 않을 경우 그룹 생성.
    //Parameter : string groupName, string userID
    //Return Value :
    // 1) GROUPCREATION_SUCCESSFUL : 그룹 생성이
정상적으로 완료되었음
    // 2) GROUPCREATION_HASJOINEDGROUP : user가 이미
가입한 그룹이 있음(그룹 생성 실패)
    // 3) GROUPCREATION_DUPLICATEDNAME : 생성 요청한
이름과 중복된 그룹이 존재함(그룹 생성 실패)
    int createNewGroup(string groupName, string
userID);
    //Function : bool isDuplicatedName(string
```

```

groupName)
//Description : 입력받은 그룹 이름과 같은 이름의
그룹이 이미 존재하는지 확인
//Parameter : string groupName
//Return Value :
// 1) true : 중복된 이름의 그룹이 이미 존재함
// 2) false : 중복된 이름의 그룹이 없음
bool isDuplicatedName(string groupName);
//Function : CreateNewGroup(CreateNewGroupUI
**createNewGroupUI)
//Description : 그룹 생성 컨트롤 생성자. 그룹 생성
UI도 생성함
//Parameter : CreateNewGroupUI **createNewGroupUI
//Return Value : CreateNewGroup
CreateNewGroup(CreateNewGroupUI
**createNewGroupUI);
//Function : ~CreateNewGroup()
//Description : 그룹 생성 컨트롤 소멸자. 그룹 생성
UI도 delete함
//Parameter : 없음
//Return Value : 없음
~CreateNewGroup();
};

#endif

```

CreateNewGroup.cpp

```

// 작성자 : 심민우
// 작성 날짜 : 2017.05.27
// 설명 : 그룹 생성 Control class Operation 정의
//         createNewGroup() 함수 정의
//         CreateNewGroup() 생성자 정의

#include "CreateNewGroupUI.h"

extern FILE* out_fp;

int CreateNewGroup::createNewGroup(string groupName,
string userID)
{// currentMember에 memberCollection에서 찾은
Member 객체 반환
Member *currentMember =
MemberCollection::getEntireMembers()-
>findMember(userID);

// Member 객체로 이미 가입한 그룹이 있는지 확인
m_isJoinedGroup = currentMember-
>checkJoinedGroup();

```

```

// m_isJoinedGroup == true -> 이미 가입한 그룹이
있는 경우
// return false
if (m_isJoinedGroup == true)
{
    return GROUPCREATION_HASJOINEDGROUP;
}

if (isDuplicatedName(groupName))
{
    return GROUPCREATION_DUPLICATEDNAME;
}

// group Collection에 새로 생성한 group 추가
// group Collection에서 새 그룹 생성
함수(Group::groupCreate) 호출
GroupCollection::getEntireGroups()->addGroup( new
Group(groupName, user ID) );

currentMember->setJoinedGroupName(groupName);

return GROUPCREATION_SUCCESSFUL;
}

bool CreateNewGroup::isDuplicatedName(string
groupName)
{
    Group* group = GroupCollection::getEntireGroups()-
>findGroup(groupName);
    if (group == nullptr) // findGroup의 반환값이
nullptr인 경우는 중복된 group이름이 없으므로 그룹
생성이 가능함
        return false;
    else
        return true;
}

CreateNewGroup::CreateNewGroup(CreateNewGroupUI
**createNewGroupUI)
{
    // boundary class 생성
    m_createNewGroupUI = new CreateNewGroupUI(this);
    (*createNewGroupUI) = m_createNewGroupUI;
}

CreateNewGroup::~CreateNewGroup()
{
    delete m_createNewGroupUI;
}

```

ShowNowVoteUI.h

```
// Class : ShowNowVoteUI
// Description : 현재 진행 중인 투표 조회를 위한
boundary 클래스
// 작성자 : 강석원
// 작성 시간 : 2017.05.27 20:10
// 설명 : class ShowNowVoteUI의 정의(.h)
// 수정자 : 이한솔
// 수정 시간 : 2017.05.30 03:33
// 수정 이유 : ShowNowVoteUI::printVote() 인자 타입
수정

#ifndef SHOWNOWVOTEUI_H
#define SHOWNOWVOTEUI_H

#include "showNowVote.h"
#include <string>
#include <ctime>
using namespace std;

class ShowNowVote;

class ShowNowVoteUI
{
private:
    ShowNowVote *m_ShowNowVoteControl;
public:
    //Function : ShowNowVoteUI(ShowNowVote* control)
    //Description : ShowNowVoteUI 생성자, parameter
    //로 받은 control 을 m_ShowNowVoteControl 에 할당
    //Parameter : ShowNowVote* control
    //Return Value : ShowNowVote
    ShowNowVoteUI(ShowNowVote* control);
    ~ShowNowVoteUI();

    //Function : void requestDoVote(string userID,
    string voteTopic, int selectedItem)
    //Description : 투표하기를 위한 함수
    //Parameter : string userID, string voteTopic, int
    selectedItem
    //Return Value : void
    void requestDoVote(string userID, string
    voteTopic, int selectedItem);

    //Function : void printVote(string voteTopic, int
    numOfItems, int remainedTime)
    //Description : 현재 진행중인 투표를 출력하기 위한
    함수
    //Parameter : string voteTopic, int numOfItems,
    int remainedTime
```

```
//Return Value : void
void printVote(string voteTopic, int numOfItems,
int remainedTime);

//Function : void requestShowNowVote(string
userID)
//Description : 현재 진행중인 투표를 요청받는 함수
//Parameter : string userID
//Return Value : void
void requestShowNowVote(string userID);
};

#endif
```

ShowNowVoteUI.cpp

```
//작성자 : 강석원
//작성 시간 : 2017.05.27 20:10
//설명 : class ShowNowVoteUI의 구현(.cpp)
//수정자 : 이한솔
//수정 시간 : 2017.05.28 02:39
//수정 이유 : printVote() 출력문 수정
//수정자 : 이한솔
//수정 시간 : 2017.05.30 03:33
//수정 이유 : ShowNowVoteUI::printVote() 인자 및
내용 수정

#include "showNowVoteUI.h"
#include <string>
#include <ctime>

extern FILE* out_fp;

ShowNowVoteUI::ShowNowVoteUI(ShowNowVote* control)
{
    m_ShowNowVoteControl = control;
}

ShowNowVoteUI::~ShowNowVoteUI()
{
}

void ShowNowVoteUI::requestDoVote(string userID,
string voteTopic, int selectedItem)
```

```

int result;
fprintf(out_fp, "4.2. 투표\n");
printf("4.2. 투표\n");

result = m_ShowNowVoteControl->doVote(userID,
voteTopic, selectedItem);

switch (result)
{
case DOVOTEFAILED_THEREISNOITEM:
    fprintf(out_fp, "> 아이템 %d이 없습니다.\n",
selectedItem);
    printf("> 아이템 %d이 없습니다.\n",
selectedItem);
    return;
case DOVOTEFAILED_ALREADYVOTED:
    fprintf(out_fp, "> %s은 이미
투표하셨습니다.\n", voteTopic.c_str());
    printf("> %s은 이미 투표하셨습니다.\n",
voteTopic.c_str());
    return;
case DOVOTEFAILED_THEREISNOVOTE:
    fprintf(out_fp, "> %s이 없습니다.\n",
voteTopic.c_str());
    printf("> %s이 없습니다.\n", voteTopic.c_str());
    return;
case DOVOTEFAILED_NOJOINEDGROUP:
    fprintf(out_fp, "> 가입한 그룹 정보가 없습니다.
그룹 가입을 먼저 하세요.\n");
    printf("> 가입한 그룹 정보가 없습니다. 그룹
가입을 먼저 하세요.\n");
    return;
}

fprintf(out_fp, "> %s %d\n", voteTopic.c_str(),
selectedItem);
printf("> %s %d\n", voteTopic.c_str(),
selectedItem);
}

void ShowNowVoteUI::printVote(string voteTopic, int
numOfItems, int remainedTime)
{
    int hours, minutes;
    hours = remainedTime / 3600;
    minutes = (remainedTime % 3600) / 60;

    fprintf(out_fp, "> %s %d ", voteTopic.c_str(),
numOfItems);
    fprintf(out_fp, "%02d:%02d\n", hours, minutes);

    printf("> %s %d ", voteTopic.c_str(), numOfItems);
    printf("%02d:%02d\n", hours, minutes);
}

```

```

void ShowNowVoteUI::requestShowNowVote(string
userID)
{
    int result;
    fprintf(out_fp, "4.1. 현재 진행 중인 투표
리스트\n");
    printf("4.1. 현재 진행 중인 투표 리스트\n");

    result = m_ShowNowVoteControl-
>showNowVote(userID);
    if (result == SHOWNOWVOTEFAILED_NOJOINEDGROUP)
    {
        fprintf(out_fp, "> 가입한 그룹 정보가 없습니다.
그룹 가입을 먼저 하세요.\n");
        printf("> 가입한 그룹 정보가 없습니다. 그룹
가입을 먼저 하세요.\n");
    }
}

```

ShowNowVote.h

```

// Class : ShowNowVoteUI
// Description : 현재 진행 중인 투표 조회를 위한
control 클래스
// 작성자 : 강석원
// 작성 시간 : 2017.05.27 20:10
// 설명 : class ShowNowVote의 정의(.h)

#ifndef SHOWNOWVOTE_H
#define SHOWNOWVOTE_H

#define SUCCESSFULSHOWNOWVOTE 0
#define SHOWNOWVOTEFAILED_NOJOINEDGROUP -1
#define SUCCESSFULDOVOTE 0
#define DOVOTEFAILED_NOJOINEDGROUP -1
#define DOVOTEFAILED_THEREISNOITEM -2
#define DOVOTEFAILED_ALREADYVOTED -3
#define DOVOTEFAILED_THEREISNOVOTE -4

#include "showNowVoteUI.h"
#include <string>
using namespace std;

class ShowNowVoteUI;

class ShowNowVote
{
private:
    ShowNowVoteUI *m_ShowNowVoteUI;
public:
    //Function : ShowNowVote(ShowNowVoteUI **UI)
}

```

```

//Description : ShowNowVote 생성자 parameter 로
받은 UI 를 멤버변수 m_ShowNowVoteUI 에 할당
//Parameter : ShowNowVoteUI **UI
//Return Value : ShowNowVote
ShowNowVote(ShowNowVoteUI **UI);
~ShowNowVote();

//Function : int doVote(string userID, string
voteTopic, int selectedItem)
//Description : 투표하기를 위한 함수
//Parameter : string userID, string voteTopic, int
selectedItem
//Return Value :
// 1) SUCCESSFULDOVOTE : 성공적으로 투표 완료시
// 2) DOVOTEFAILED_THEREISNOITEM : 투표 요청한
항목이 해당 투표의 항목 갯수보다 많을 때
// 3) DOVOTEFAILED_ALREADYVOTED : 이미 해당
투표에 투표한 경우
// 4) DOVOTEFAILED_THEREISNOVOTE : 투표 요청한
투표가 존재하지 않는 경우
// 5) DOVOTEFAILED_NOJOINEDGROUP : 가입한 그룹이
없는 경우
int doVote(string userID, string voteTopic, int
selectedItem);

//Function : int showNowVote(string userID)
//Description : 현재 진행중인 투표를 보여주기 위한
함수
//Parameter : string userID
//Return Value :
// 1) SUCCESSFULSHOWNOWVOTE : 현재 진행중인
투표를 성공적으로 출력 할 시
// 2) SHOWNOWVOTEFAILED_NOJOINEDGROUP : 가입한
그룹이 없는 경우
int showNowVote(string userID);
};

#endif

```

ShowNowVote.cpp

```

//작성자 : 강석원
//작성 시간 : 2017.05.27 20:10
//설명 : class ShowNowVote의 구현(.cpp)
//수정자 : 이한솔
//수정 시간 : 2017.05.28 02:32
//수정 이유 : showNowVote() print부분 수정
//수정자 : 이한솔
//수정 시간 : 2017.05.28 03:46
//수정 이유 : showNowVote() getVoteList(TYPE)으로
통일
//수정자 : 이한솔
//수정 시간 : 2017.05.30 03:33
//수정 이유 : Vote::getRemainedTime() 수정

```

```

#include "showNowVote.h"
#include "MemberCollection.h"
#include "GroupCollection.h"

extern FILE* in_fp, *out_fp;

ShowNowVote::ShowNowVote(ShowNowVoteUI **UI)
{
    m_ShowNowVoteUI = new ShowNowVoteUI(this);
    (*UI) = m_ShowNowVoteUI;
}

ShowNowVote::~ShowNowVote()
{
    delete m_ShowNowVoteUI;
}

int ShowNowVote::doVote(string userID, string
voteTopic, int selectedItem)
{
    Member* user =
    MemberCollection::getEntireMembers()->findMember(userID);

    if (user->checkJoinedGroup())
    {
        string groupName = user->getJoinedGroupName();
        Group* joinedGroup =
        GroupCollection::getEntireGroups()->findGroup(groupName);
        VoteCollection& voteList = joinedGroup-
        >getVoteCollection();
        Vote* targetToVote =
        voteList.findVote(voteTopic);

        if (targetToVote != NULL)
        {
            if (targetToVote->checkAlreadyVoted(userID))
            {
                if (targetToVote->checkItem(selectedItem))
                {
                    targetToVote->doVote(userID, selectedItem);
                    return SUCCESSFULDOVOTE;
                }
            }
            else
            {
                /* error return : there is no item to vote */
                return DOVOTEFAILED_THEREISNOITEM;
            }
        }
        else
        {
            /*error return : already voted*/
        }
    }
}

```

```

        return DOVOTEFAILED_ALREADYVOTED;
    }
}
else
{
    /* error return : there is no target to vote */
    return DOVOTEFAILED_THEREISNOVOTE;
}
else
{
    /* error return : current user dosen't have
joinedGroup */
    return DOVOTEFAILED_NOJOINEDGROUP;
}
}

int ShowNowVote::showNowVote(string userID)
{
    Member* user =
MemberCollection::getEntireMembers()-
>findMember(userID);

    if (user->checkJoinedGroup())
    {
        string groupName = user->getJoinedGroupName();
        Group* joinedGroup =
GroupCollection::getEntireGroups()-
>findGroup(groupName);
        VoteCollection& voteList = joinedGroup-
>getVoteCollection();
        vector<Vote*>& nowVoteList =
voteList.getVoteList(VOTETYPE_NOWVOTING);
        size_t Size = nowVoteList->size();

        for (int i = 0; i < Size; i++)
        {
            Vote* iterator = nowVoteList->at(i);
            string topic = iterator->getVoteTopic();
            int num0fItems = iterator->getNum0fItems();
            int remainedTime = iterator-
>getRemainedTime();

            m_ShowNowVoteUI->printVote(topic, num0fItems,
remainedTime);
        }

        delete nowVoteList;
    }

    return SUCCESSFULSHOWNOWVOTE;
}
else
{
    return SHOWNOWVOTEFAILED_NOJOINEDGROUP; /*show
now vote failed*/
}

```

```

    }
}
```

ShowFutureVoteUI.h

```

// Class : ShowFutureVoteUI
// Description : 향후 진행 예정인 투표 조회를 위한
boundary 클래스
// 작성자 : 강석원
// 작성 시간 : 2017.05.27 20:10
// 설명 : class ShowFutureVoteUI의 정의(.h)

#ifndef SHOWFUTUREVOTEUI_H
#define SHOWFUTUREVOTEUI_H

#include "ShowFutureVote.h"
#include <string>
using namespace std;

class ShowFutureVote;

class ShowFutureVoteUI
{
private:
    ShowFutureVote* m_ShowFutureVoteControl;
public:
    //Function : ShowFutureVoteUI(ShowFutureVote*
control)
    //Description : ShowFutureVoteUI 생성자, parameter
로 받아온 control 을 멤버변수
m_ShowFutureVoteControl 에 할당
    //Parameter : ShowFutureVote* control
    //Return Value : ShowFutureVoteUI
    ShowFutureVoteUI(ShowFutureVote* control);
    ~ShowFutureVoteUI();

    //Function : void requestShowFutureVote(string
userID)
    //Description : 향후 진행 예정인 투표 출력을
요청받는 함수
    //Parameter : string userID
    //Return Value : void
    void requestShowFutureVote(string userID);

    //Function : void printVote(string voteTopic, int
num0fItems, tm startTime, tm finishTime)
    //Description : 향후 진행 예정인 투표 출력을 위한
함수
    //Parameter : string voteTopic, int num0fItems, tm
startTime, tm finishTime)
    //Return Value : void
    void printVote(string voteTopic, int num0fItems,
tm startTime, tm finishTime);
};
```

```
#endif
```

ShowFutureVoteUI.cpp

```
//작성자 : 강석원  
//작성 시간 : 2017.05.27 20:10  
//설명 : class ShowFutureVote의 구현(.cpp)  
//수정자 : 이한솔  
//수정 시간 : 2017.05.28 02:45  
//수정 이유 : printVote() 출력문 수정
```

```
#include "ShowFutureVoteUI.h"  
#include <ctime>  
  
extern FILE* out_fp;  
  
void ShowFutureVoteUI::requestShowFutureVote(string  
userID)  
{  
    int result;  
  
    fprintf(out_fp, "4.3. 향후 진행 예정인 투표  
리스트\n");  
    printf("4.3. 향후 진행 예정인 투표 리스트\n");  
  
    result = m_ShowFutureVoteControl->  
showFutureVote(userID);  
  
    if (result == SHOWFUTUREVOTEFAILED_NOJOINEDGROUP)  
    {  
        fprintf(out_fp, "> 가입한 그룹 정보가 없습니다.  
그룹 가입을 먼저 하세요.\n");  
        printf("> 가입한 그룹 정보가 없습니다. 그룹  
가입을 먼저 하세요.\n");  
    }  
}  
  
ShowFutureVoteUI::ShowFutureVoteUI(ShowFutureVote*  
control)  
{  
    m_ShowFutureVoteControl = control;  
}  
  
ShowFutureVoteUI::~ShowFutureVoteUI()  
{  
}  
  
void ShowFutureVoteUI::printVote(string voteTopic,  
int numOfItems, tm startTime, tm finishTime)  
{  
    fprintf(out_fp, "> %s %d ", voteTopic.c_str(),  
numOfItems);  
    fprintf(out_fp, "%04d:%02d:%02d:%02d:%02d ",  
startTime.tm_year, startTime.tm_mon,
```

```
startTime.tm_mday, startTime.tm_hour,  
startTime.tm_min);  
    fprintf(out_fp, "%04d:%02d:%02d:%02d:%02d\n",  
finishTime.tm_year, finishTime.tm_mon,  
finishTime.tm_mday, finishTime.tm_hour,  
finishTime.tm_min);  
  
    printf("> %s %d ", voteTopic.c_str(), numOfItems);  
    printf("%04d:%02d:%02d:%02d:%02d ",  
startTime.tm_year, startTime.tm_mon,  
startTime.tm_mday, startTime.tm_hour,  
startTime.tm_min);  
    printf("%04d:%02d:%02d:%02d:%02d\n",  
finishTime.tm_year, finishTime.tm_mon,  
finishTime.tm_mday, finishTime.tm_hour,  
finishTime.tm_min);  
}
```

ShowFutureVote.h

```
// Class : ShowFutureVote  
// Description : 향후 진행 예정인 투표 조회를 위한  
control 클래스  
// 작성자 : 강석원  
// 작성 시간 : 2017.05.27 20:10  
// 설명 : class ShowNowVote의 정의(.h)  
  
#ifndef SHOWFUTUREVOTE_H  
#define SHOWFUTUREVOTE_H  
  
#define SUCCESSFULSHOWFUTUREVOTE 0  
#define SHOWFUTUREVOTEFAILED_NOJOINEDGROUP -1  
  
#include "ShowFutureVoteUI.h"  
#include <string>  
using namespace std;  
  
class ShowFutureVoteUI;  
  
class ShowFutureVote  
{  
private:  
    ShowFutureVoteUI* m_ShowFutureVoteUI;  
public:  
    //Function : ShowFutureVote(ShowFutureVoteUI**  
control)  
    //Description : ShowFutureVote 생성자 ,  
parameter로 받은 control을 멤버변수  
    // m_ShowFutureVoteUI 에 할당  
    //Parameter : ShowFutureVoteUI** control
```

```

//Return Value : ShowFutureVote
ShowFutureVote(ShowFutureVoteUI** control);
~ShowFutureVote();

//Function : int showFutureVote(string userID)
//Description : 향후 진행 예정인 투표를 출력하기 위한 함수
//Parameter : string userID
//Return Value :
//    1) SUCCESSFULSHOWFUTUREVOTE : 향후 진행 예정인 투표 출력 성공시
//    2) SHOWFUTUREVOTEFAILED_NOJOINEDGROUP : 가입 그룹이 없을 시
int showFutureVote(string userID);
};

#endif

```

ShowFutureVote.cpp

```

//작성자 : 강석원
//작성 시간 : 2017.05.27 20:10
//설명 : class ShowNowVote의 구현(.cpp)
//수정자 : 이한솔
//수정 시간 : 2017.05.28 02:32
//수정 이유 : showNowVote() print부분 수정
//수정자 : 이한솔
//수정 시간 : 2017.05.28 03:46
//수정 이유 : showFutureVote() getVoteList(TYPE)으로 통일
//수정자 : 강석원
//수정 시간 : 2017

#include "ShowFutureVote.h"
#include "MemberCollection.h"
#include "Member.h"
#include "GroupCollection.h"
#include "Group.h"

extern FILE* out_fp;

ShowFutureVote::ShowFutureVote(ShowFutureVoteUI** control)
{
    m_ShowFutureVoteUI = new ShowFutureVoteUI(this);
    (*control) = m_ShowFutureVoteUI;
}

```

```

ShowFutureVote::~ShowFutureVote()
{
    delete m_ShowFutureVoteUI;
}

int ShowFutureVote::showFutureVote(string userID)
{
    Member* user =
    MemberCollection::getEntireMembers()-
    >findMember(userID);

    if (user->checkJoinedGroup())
    {
        string groupName = user->getJoinedGroupName();
        Group* joinedGroup =
        GroupCollection::getEntireGroups()-
        >findGroup(groupName);
        VoteCollection& voteList = joinedGroup-
        >getVoteCollection();
        vector<Vote*>& futureVoteList =
        voteList.getVoteList(VOTETYPE_FUTURE);
        size_t Size = futureVoteList->size();

        for (int i = 0; i < Size; i++)
        {
            Vote* iterator = futureVoteList->at(i);
            string topic = iterator->getVoteTopic();
            int numItems = iterator->getNumOfItems();
            tm startTime = iterator->getStartTime();
            tm finishTime = iterator->getFinishTime();

            m_ShowFutureVoteUI->printVote(topic,
            numItems, startTime, finishTime);
        }

        delete futureVoteList;
    }

    return SUCCESSFULSHOWFUTUREVOTE;
}
else
{
    return SHOWFUTUREVOTEFAILED_NOJOINEDGROUP;
}
}

```

ShowTerminatedVoteUI.h

```
// Class : ShowTerminatedVoteUI  
// Description : 종료된 투표 조회를 위한 boundary  
클래스  
// 작성자 : 강석원  
// 작성 시간 : 2017.05.27 21:25  
// 설명 class TerminatedVoteUI의 정의(.h)  
  
#ifndef SHOWTERMINATEDVOTEUI_H  
#define SHOWTERMINATEDVOTEUI_H  
  
#include <string>  
#include "ShowTerminatedVote.h"  
  
using namespace std;  
  
class ShowTerminatedVote;  
  
class ShowTerminatedVoteUI  
{  
private:  
    ShowTerminatedVote* m_ShowTerminatedVoteControl;  
public:  
    //Function :  
    ShowTerminatedVoteUI(ShowTerminatedVote* control)  
        //Description : ShowTerminatedVoteUI 생성자 ,  
        parameter로 받아온 control을 멤버변수  
        m_ShowTerminatedVoteControl에 할당  
        //Parameter : ShowTerminatedVote* control  
        //Return Value : ShowTerminatedVoteUI  
        ShowTerminatedVoteUI(ShowTerminatedVote* control);  
        ~ShowTerminatedVoteUI();  
  
    //Function : void requestShowTerminatedVote(string  
    userID)  
        //Description : 종료된 투표 리스트를 얻어오기 위한  
        함수  
        //Parameter : string userID  
        //Return Value : void  
        void requestShowTerminatedVote(string userID);  
  
    //Function : void printVote(string voteTopic, int  
    numOfItems, int* result)  
        //Description : 종료된 투표의 정보를 출력하기 위한  
        함수  
        //Parameter : string voteTopic, int numOfItems,  
        int* result  
        //Return Value : void  
        void printVote(string voteTopic, int numOfItems,  
        int* result);
```

```
};
```

```
#endif
```

ShowTerminatedVoteUI.cpp

```
//작성자 : 강석원  
//작성 시간 : 2017.05.27 21:27  
//설명 class TerminatedVoteUI의 구현(.cpp)부분  
//수정자 : 이한솔  
//수정 시간 : 2017.05.28 02:49  
//수정 이유 : printVote() 출력문 수정  
//수정자 : 강석원  
//수정 시간 : 2017.05.30 00:26  
//수정 이유 : printVote, requestShowTerminatedVote  
출력  
  
#include "ShowTerminatedVoteUI.h"  
#include <ctime>  
extern FILE* out_fp;  
  
ShowTerminatedVoteUI::ShowTerminatedVoteUI(ShowTerminatedVote* control)  
{  
    m_ShowTerminatedVoteControl = control;  
}  
  
ShowTerminatedVoteUI::~ShowTerminatedVoteUI()  
{  
}  
  
void  
ShowTerminatedVoteUI::requestShowTerminatedVote(string userID)  
{  
    int result;  
    fprintf(out_fp, "4.4. 종료된 투표 리스트\n");  
    printf("4.4. 종료된 투표 리스트\n");  
  
    result = m_ShowTerminatedVoteControl->showTerminatedVote(userID);  
  
    if (result ==  
        SHOWTERMINATEDVOTEFAILED_NOJOINEDGROUP)  
    {  
        fprintf(out_fp, "> 가입한 그룹 정보가 없습니다.  
그룹 가입을 먼저 하세요.\n");  
        printf("> 가입한 그룹 정보가 없습니다. 그룹  
가입을 먼저 하세요.\n");  
    }  
}  
  
void ShowTerminatedVoteUI::printVote(string  
voteTopic, int numOfItems, int* result)  
{
```

```

int i;
fprintf(out_fp, "> %s ", voteTopic.c_str());
printf("> %s ", voteTopic.c_str());

for (i = 0; i < numOfItems; i++)
{
    fprintf(out_fp, "%d:%0d ", i+1, result[i]);
    printf("%d:%d ", i+1, result[i]);
}
fprintf(out_fp, "%n");
printf("%n");
}

```

ShowTerminatedVote.h

```

// Class : ShowTerminatedVote
// Description : 종료된 투표 조회를 위한 control
클래스
// 작성자 : 강석원
// 작성 시간 : 2017.05.27 21:25
// 설명 : class TerminatedVote의 정의(.h)
// 수정자 : 강석원
// 수정 시간 : 2017.05.30 00:54
// 수정 이유 : 에러처리를 위한 반환 처리

#ifndef SHOWTERMINATEDVOTE_H
#define SHOWTERMINATEDVOTE_H

#define SUCCESSFULSHOWTERMINATEDVOTE 0
#define SHOWTERMINATEDVOTEFAILED_NOJOINEDGROUP -1

#include "ShowTerminatedVoteUI.h"
#include <string>
using namespace std;

class ShowTerminatedVoteUI;

class ShowTerminatedVote
{
private:
    ShowTerminatedVoteUI* m_ShowTerminatedVoteUI;
public:
    //Function :
    ShowTerminatedVote(ShowTerminatedVoteUI** pointerFromMain)
        //Description : ShowTerminatedVote 생성자,
        parameter로 받아온 UI 를 m_ShowTerminatedVoteUI에
        할당
        //Parameter : ShowTerminatedVoteUI** UI
        //Return Value : ShowTerminatedVote
    ShowTerminatedVote(ShowTerminatedVoteUI** UI);
    ~ShowTerminatedVote();

    //Function : int showTerminatedVote(string userID)

```

```

//Description : 종료된 투표 조회하기 위한 함수
//Parameter : string userID
//Return Value :
// 1) SUCCESSFULSHOWTERMINATEDVOTE : 가입 된
그룹이 있을 시
// 2) SHOWTERMINATEDVOTEFAILED_NOJOINEDGROUP :
가입 된 그룹이 없을 시
int showTerminatedVote(string userID);
};

#endif

```

ShowTerminatedVote.cpp

```

//작성자 : 강석원
//작성 시간 : 2017.05.27 21:28
//설명 class TerminatedVote의 구현(.cpp)부분
//수정자 : 이한솔
//수정 시간 : 2017.05.28 02:47
//수정 이유 : showTerminatedVote() print부분 수정
//수정자 : 이한솔
//수정 시간 : 2017.05.28 03:46
//수정 이유 : showTerminatedVote()
getVoteList(TYPE)으로 통일
//수정자 : 강석원
//수정 시간 : 2017.05.30 12:53
//수정 이유 : 출력문 처리를 위한 에러 반환 수정

```

```

#include "ShowTerminatedVote.h"
#include "MemberCollection.h"
#include "Member.h"
#include "GroupCollection.h"
#include "Group.h"

```

```

ShowTerminatedVote::ShowTerminatedVote(ShowTerminatedVoteUI** UI)
{
    m_ShowTerminatedVoteUI = new
    ShowTerminatedVoteUI(this);
    (*UI) = m_ShowTerminatedVoteUI;
}

ShowTerminatedVote::~ShowTerminatedVote()
{
    delete m_ShowTerminatedVoteUI;
}

int ShowTerminatedVote::showTerminatedVote(string userID)
{
    Member* user =
    MemberCollection::getEntireMembers()->

```

```

>findMember(userID);

if (user->checkJoinedGroup())
{
    string groupName = user->getJoinedGroupName();
    Group* joinedGroup =
    GroupCollection::getEntireGroups()-
    >findGroup(groupName);
    VoteCollection& voteList = joinedGroup-
    >getVoteCollection();
    vector<Vote*>* terminatedVoteList =
    voteList.getVoteList(VOTETYPE_TERMINATED);
    size_t Size = terminatedVoteList->size();

    for (int i = 0; i < Size; i++)
    {
        Vote* iterator = terminatedVoteList->at(i);
        string topic = iterator->getVoteTopic();
        int numofItems = iterator->getNumofItems();
        int *itemResult = iterator->getItemResult();

        m_ShowTerminatedVoteUI->printVote(topic,
        numofItems, itemResult);
    }

    delete terminatedVoteList;
    return SUCCESSFULSHOWTERMINATEDVOTE;
}
else
{
    /* error return : user doen't have joined
group*/
    return SHOWTERMINATEDVOTEFAILED_NOJOINEDGROUP;
}
}

```

OfferVoteUI.h

```

// Class : OfferVoteUI
// Description : 투표제안을 위한 boundary 클래스
// 작성자 : 강석원
// 작성 날짜 : 2017.05.27 20:30
// 설명 : class OfferVoteUI의 정의(.h)
// 수정자 : 강석원
// 수정 날짜 : 2017.05.30 12:37
// 수정 이유 : 출력 부분 수정

#ifndef OFFERVOTEUI_H
#define OFFERVOTEUI_H

#include "OfferVote.h"
#include <string>
using namespace std;

class OfferVote;

class OfferVoteUI
{
private:
    OfferVote* m_OfferVoteControl;
public:
    //Function : OfferVoteUI(OfferVote* control)
    //Description : OfferVoteUI 생성자, parameter로
    받은 control 을
    //      m_OfferVoteControl에 할당
    //Parameter : OfferVote* control
    //Return Value : OfferVoteUI
    OfferVoteUI(OfferVote* control);

    //Function : void requestOfferVote(string userID,
    string voteTopic, int numofItems, tm startTime, tm
    finishTime)
    //Description : 투표 제안 요청을 받기 위한 함수
    //Parameter : string userID, string voteTopic, int
    numofItems, tm startTime, tm finishTime
    //Return Value : void
    void requestOfferVote(string userID, string
    voteTopic, int numofItems, tm startTime, tm
    finishTime);
};

#endif

```

OfferVoteUI.cpp

```
//작성자 : 강석원
//작성 시간 : 2017.05.27 18:58
//설명 : class OfferVoteUI의 구현(.cpp) 부분
//수정자 : 강석원
//수정 시간 : 2017.05.30 12:36
//수정 이유 : 출력 부분 수정

#include "OfferVoteUI.h"
#include <string>
#include <ctime>
using namespace std;

extern FILE* in_fp;
extern FILE* out_fp;

OfferVoteUI::OfferVoteUI(OfferVote * control)
{
    m_OfferVoteControl = control;
}

void OfferVoteUI::requestOfferVote(string userID,
string voteTopic, int numOfItems, tm startTime, tm
finishTime)
{
    int result;
    fprintf(out_fp, "3.1. 투표 제안\n");
    printf("3.1. 투표 제안\n");

    result = m_OfferVoteControl->offerVote(userID,
voteTopic, numOfItems, startTime, finishTime);
    switch(result)
    {
        case OFFERVOTEFAILED_NOJOINEDGROUP:
            fprintf(out_fp, "> 가입한 그룹 정보가 없습니다.
그룹 가입을 먼저 하세요.\n");
            printf("> 가입한 그룹 정보가 없습니다. 그룹
가입을 먼저 하세요.\n");
            return;
        case OFFERVOTEFAILED_DUPLICATEDVOTETOPIC:
            fprintf(out_fp, "> 중복된 투표 주제가
있습니다\n");
            printf("> 중복된 투표 주제가 있습니다\n");
            return;
        case OFFERVOTEFAILED_INVALIDTIME:
            fprintf(out_fp, "> 입력한 투표 시간이 올바르지
않습니다. (투표 시작 시간 < 투표 종료 시간)\n");
            printf("> 입력한 투표 시간이 올바르지 않습니다.
(투표 시작 시간 < 투표 종료 시간)\n");
            return;
    }

    fprintf(out_fp, "> %s %d ", voteTopic.c_str(),
```

```
numOfItems);

    fprintf(out_fp, "%04d:%02d:%02d:%02d:%02d ",
startTime.tm_year, startTime.tm_mon,
startTime.tm_mday, startTime.tm_hour,
startTime.tm_min);

    fprintf(out_fp, "%04d:%02d:%02d:%02d\n",
finishTime.tm_year, finishTime.tm_mon,
finishTime.tm_mday, finishTime.tm_hour,
finishTime.tm_min);

    printf("> %s %d ", voteTopic.c_str(), numOfItems);
    printf("%04d:%02d:%02d:%02d:%02d ",
startTime.tm_year, startTime.tm_mon,
startTime.tm_mday, startTime.tm_hour,
startTime.tm_min);

    printf("%04d:%02d:%02d:%02d:%02d\n",
finishTime.tm_year, finishTime.tm_mon,
finishTime.tm_mday, finishTime.tm_hour,
finishTime.tm_min);

}
```

OfferVote.h

```
// Class : OfferVote
// Description : 투표제안을 위한 control 클래스
// 작성자: 강석원
// 작성 날짜: 2017.05.27 20:30
// 설명 : class OfferVote의 정의(.h)

#ifndef OFFERVOTE_H
#define OFFERVOTE_H

#define SUCCESSFULOFFERVOTE 0
#define OFFERVOTEFAILED_NOJOINEDGROUP -1
#define OFFERVOTEFAILED_DUPLICATEDVOTETOPIC -2
#define OFFERVOTEFAILED_INVALIDTIME -3

#include "OfferVoteUI.h"

#include <string>
#include <ctime>
using namespace std;

class OfferVoteUI;

class OfferVote
```

```

{
private:
    OfferVoteUI *m_OfferVoteUI;
public:
    //Function : OfferVote(OfferVoteUI** UI)
    //Description : OfferVoteUI 생성자, parameter로
    받은 UI 를
        //      m_OfferVoteUI 에 할당
    //Parameter : OfferVote** UI
    //Return Value : OfferVote
    OfferVote(OfferVoteUI **UI);
    ~OfferVote();

    //Function : bool validCheck(tm startTime, tm
    finishTime)
    //Description : 투표 제안 요청을 받은 투표의
    시각이 유효한지 검사하는 함수
    //Parameter : tm startTime, tm finishTime
    //Return Value : boolean - 시작시간이 종료시간보다
    앞서 있을 시 false 반환
        //          시작시간이 종료시간보다 뒤일 경우
    true 반환
    bool validCheck(tm startTime, tm finishTime);

    //Function : int offerVote(string userID, string
    voteTopic, int numOfItem, tm startTime, tm
    finishTime)
    //Description : 투표 제안을 위한 함수
    //Parameter : string userID, string voteTopic, int
    numOfItem, tm startTime, tm finishTime
    //Return Value : int - 성공적으로 투표가 생성된
    경우 SUCCESSFULOFFERVOTE 반환
        // 동일 이름의 투표가 이미 존재 할 경우
    OFFERVOTEFAILED_DUPLICATEDVOTETOPIC 반환
        // 가입 한 그룹이 존재하지 않을 경우
    OFFERVOTEFAILED_NOJOINEDGROUP 반환
        // 투표 시각이 유효하지 않은 경우
    OFFERVOTEFAILED_INVALIDTIME 반환
    int offerVote(string userID, string voteTopic, int
    numOfItem, tm startTime, tm finishTime);
};

#endif

```

OfferVote.cpp

//작성자 : 강석원
//작성 시간 : 2017.05.27 17:55
//설명 : class OfferVote 의 구현(.cpp) 부분

```

#include "OfferVote.h"
#include "OfferVoteUI.h"
#include "Member.h"
#include "MemberCollection.h"
#include "Group.h"
#include "GroupCollection.h"
#include "Vote.h"
#include "VoteCollection.h"

extern tm currentTime;
extern bool operator>(tm, tm);
extern FILE* out_fp;

OfferVote::OfferVote(OfferVoteUI** UI)
{
    m_OfferVoteUI = new OfferVoteUI(this);
    (*UI) = m_OfferVoteUI;
}

OfferVote::~OfferVote()
{
    delete m_OfferVoteUI;
}

bool OfferVote::validCheck(tm startTime, tm
finishTime)
{
    if (startTime > finishTime)
        return false;
    else
        return true;
}

int OfferVote::offerVote(string userID, string
voteTopic, int numOfItems, tm startTime, tm
finishTime)
{
    if (validCheck(startTime, finishTime))
    {
        Member* user =
        MemberCollection::getEntireMembers()-
        >findMember(userID);

        if (user->checkJoinedGroup())
        {
            string groupName = user->getJoinedGroupName();
            Group* joinedGroup =
            GroupCollection::getEntireGroups()-

```

```

>findGroup(groupName);
    VoteCollection& voteList = joinedGroup-
>getVoteCollection();
    if (voteList.findVote(voteTopic) == NULL)
    {
        Vote* newVote = new Vote(voteTopic,
numOfItems, startTime, finishTime, groupName);
        voteList.addVote(newVote);
        VoteCollection::getEntireVotes()-
>addVote(newVote);

        return SUCCESSFULOFFERVOTE;
    }
    else
    {
        return OFFERVOTEFAILED_DUPLICATEDVOTETOPIC;
    }
}
else
{
    /* user is not have joinedGroup */
    return OFFERVOTEFAILED_NOJOINEDGROUP;
}
else
{
    /* valied check error*/
    return OFFERVOTEFAILED_INVALIDTIME;
}
}

```

AutoDeleteTerminatedVoteUI.h

```

// Class : AutoDeleteTerminatedVoteUI
// Description: 종료 후 4주가 지난 투표 자동 삭제를
// 위한 boundary 클래스
// 작성자 : 이한솔
// 작성 날짜 : 2017.06.01 18:00
// 설명 : 종료 후 4주가 지난 투표 자동 삭제를 위한
// boundary class

#ifndef AUTODELETETERMINATEDVOTEUI_H
#define AUTODELETETERMINATEDVOTEUI_H

#include <string>
#include "AutoDeleteTerminatedVote.h"

using namespace std;

class AutoDeleteTerminatedVote;

class AutoDeleteTerminatedVoteUI
{
private:
    AutoDeleteTerminatedVote
*m_autoDeleteTerminatedVoteControl;
public:
    //Function : void requestCreateNewGroup()
    //Description : control이 입력받은 현재 시간과
비교해 종료후 4주가 지난 투표를 삭제하도록 요청
    //Parameter : tm currentTime
    //Return Value : void
    void requestAutoDeleteTerminatedVote(tm
currentTime);
    //Function : void
printAutoDeleteTerminatedVote(string groupName,
string voteTopic)
    //Description : control이 삭제 완료한 투표와 그
투표가 있었던 그룹 이름을 넘겨받아 출력문 출력
    //Parameter : string groupName, string voteTopic
    //Return Value : void
    void printAutoDeleteTerminatedVote(string
groupName, string voteTopic);
    //Function :
AutoDeleteTerminatedVoteUI(AutoDeleteTerminatedVote
*control)
    //Description : AutoDeleteTerminatedVoteUI 생성자
    //Parameter : AutoDeleteTerminatedVote *control
    //Return Value : AutoDeleteTerminatedVoteUI

AutoDeleteTerminatedVoteUI(AutoDeleteTerminatedVote
*control);
};

#endif // !AUTODELETETERMINATEDVOTEUI_H

```

AutoDeleteTerminatedVoteUI.cpp

```
// 작성자 : 이한솔  
// 작성 날짜 : 2017.06.01 18:00  
// 설명 : 종료후 4주 지난 투표 자동 삭제 boundary  
class Operation 정의 부분  
//      requestAutoDeleteTerminatedVote() 정의  
//      AutoDeleteTerminatedVoteUI 생성자 정의  
  
#include <iostream>  
#include "AutoDeleteTerminatedVoteUI.h"  
  
extern FILE* out_fp;  
  
void  
AutoDeleteTerminatedVoteUI::requestAutoDeleteTerminatedVote(tm currentTime)  
{  
    fprintf(out_fp, "> 종료 후 4주 지난 투표를  
자동으로 삭제합니다.\n");  
    printf("> 종료 후 4주 지난 투표를 자동으로  
삭제합니다.\n");  
  
    m_autoDeleteTerminatedVoteControl-  
>autoDeleteTerminatedVote(currentTime);  
}  
  
void  
AutoDeleteTerminatedVoteUI::printAutoDeleteTerminatedVote(string groupName, string voteTopic)  
{  
    fprintf(out_fp, "> 투표가 삭제되었습니다.  
그룹 : %s, 투표주제 : %s\n", groupName.c_str(),  
voteTopic.c_str());  
    printf("> 투표가 삭제되었습니다. 그룹 : %s,  
투표주제 : %s\n", groupName.c_str(),  
voteTopic.c_str());  
}  
  
AutoDeleteTerminatedVoteUI::AutoDeleteTerminatedVoteUI(AutoDeleteTerminatedVote * control)  
{  
    m_autoDeleteTerminatedVoteControl = control;  
}
```

AutoDeleteTerminateVote.h

```
// Class : AutoDeleteTerminatedVoteUI  
// Description: 종료 후 4주가 지난 투표 자동 삭제를  
위한 control 클래스  
// 작성자 : 이한솔  
// 작성 날짜 : 2017.06.01 18:00  
// 설명 : 종료 후 4주가 지난 투표 자동 삭제를 위한  
control class  
  
#ifndef AUTODELETETERMINATEDVOTE_H  
#define AUTODELETETERMINATEDVOTE_H  
  
#include <string>  
#include "GroupCollection.h"  
#include "VoteCollection.h"  
#include "AutoDeleteTerminatedVoteUI.h"  
  
using namespace std;  
  
class AutoDeleteTerminatedVoteUI;  
  
class AutoDeleteTerminatedVote  
{  
private:  
    AutoDeleteTerminatedVoteUI  
*m_autoDeleteTerminatedVoteUI;  
public:  
    //Function : void autoDeleteTerminatedVote(tm  
currentTime)  
    //Description : 입력받은 현재 시간과 비교해 종료후  
4주가 지난 투표를 삭제.  
    //      삭제가 정상적으로 완료되면 삭제된 투표의  
정보(그룹 이름, 투표 주제)를 출력함  
    //Parameter : tm currentTime  
    //Return Value : void  
    void autoDeleteTerminatedVote(tm currentTime);  
    //Function :  
    AutoDeleteTerminatedVote(AutoDeleteTerminatedVoteUI  
**autoDeleteTerminatedVoteUI)  
    //Description : 종료후 4주 지난 투표 자동 삭제  
컨트롤 생성자. UI도 생성함  
    //Parameter : AutoDeleteTerminatedVoteUI  
    **autoDeleteTerminatedVoteUI  
    //Return Value : AutoDeleteTerminatedVote  
  
    AutoDeleteTerminatedVote(AutoDeleteTerminatedVoteUI  
**autoDeleteTerminatedVoteUI);  
    //Function : ~AutoDeleteTerminatedVote()  
    //Description : 종료후 4주 지난 투표 자동 삭제  
컨트롤 소멸자. UI도 delete함  
    //Parameter : 없음  
    //Return Value : 없음  
    ~AutoDeleteTerminatedVote();
```

```
};

#endif
```

AutoDeleteTerminateVote.cpp

```
#include "AutoDeleteTerminatedVote.h"

extern time_t GetTimeT(tm tmTime);

void
AutoDeleteTerminatedVote::autoDeleteTerminatedVote( t
m currentTime)
{
    vector<Vote*>* allVoteList =
VoteCollection::getEntireVotes()->getVoteList(VOTETYPE_TERMINATED);
    size_t Size = allVoteList->size();
    vector<Vote*>::iterator iter;

    iter = allVoteList->begin();
    while (1)
    {
        if (iter == allVoteList->end()) break;

        tm finishTime = (*iter)->getFinishTime();

        int diff = (int)difftime(GetTimeT(currentTime),
GetTimeT(finishTime));
        int weeks = diff / (3600 * 24 * 7);
        if (weeks >= 4)
        {
            string voteTopic = (*iter)->getVoteTopic();
            string groupName = (*iter)->getGroupName();
            Group* voteGroup =
GroupCollection::getEntireGroups()->findGroup(groupName);
            VoteCollection& groupVoteList = voteGroup->getVoteCollection();
            Vote* deleteVote = (*iter);
            vector<Vote*>::iterator removeIter = iter;

            groupVoteList.deleteVote(voteTopic);
            iter = allVoteList->erase(removeIter);

            delete deleteVote;

            m_autoDeleteTerminatedVoteUI-
>printAutoDeleteTerminatedVote(groupName,
voteTopic);
            continue;
        }
        iter++;
    }
}
```

```
AutoDeleteTerminatedVote::AutoDeleteTerminatedVote(A
utoDeleteTerminatedVoteUI
**autoDeleteTerminatedVoteUI)
```

```
{
    m_autoDeleteTerminatedVoteUI = new
AutoDeleteTerminatedVoteUI( this);
    (*autoDeleteTerminatedVoteUI) =
m_autoDeleteTerminatedVoteUI;
}
```

```
AutoDeleteTerminatedVote::~AutoDeleteTerminatedVote(
)
{
    delete m_autoDeleteTerminatedVoteUI;
}
```

Main.cpp

```
// 소프트웨어 공학
// 3분반 6조
// B011033 김상준, B111175 이한솔, B211126 심민우,
B411005 강석원
// 과제3 : 투표 시스템 프로그램 구현

// 작성자 : 이한솔
// 작성 날짜 : 2017.05.27 19:00
// 설명 : main 및 각 메뉴 항목 정의
// 수정자 : 이한솔
// 수정 시간 : 2017.05.27 19:30
// 수정 이유 : 정의한 클래스 추가(include)
// 수정자 : 강석원
// 수정 시간 : 2017.05.27 21:39
// 수정 이유 : OfferVote, ShowNowVote,
ShowFutureVote, ShowTerminatedVote 추가, 전역변수
추가
// 수정자 : 이한솔
// 수정 시간 : 2017.05.28 01:45
// 수정 이유 : main()에서
collection변수들(MemberCollection::entireMembers,
GroupCollection::entireGroups,
VoteCollection::entireVotes) 메모리 해제 추가
// 수정자 : 이한솔
// 수정 시간 : 2017.05.28 04:20
// 수정 이유 : setCurrentTime() 구현, bool largeTm()
main.cpp로 이동(from Vote.cpp)
//         largeTm()을 >연산자 오버로딩으로 대체
// 수정자 : 심민우
// 수정 시간 : 2017.05.28 14:00
// 수정 이유 : ShowEntireGroup, ShowJoinedGroup,
CreateNewGroup 추가
// 수정자 : 이한솔
// 수정 시간 : 2017.05.28 14:10
// 수정 이유 : 로그인한 세션 리스트를 저장하기 위한
MemberCollection포인터 loginSessions 추가
// 수정자 : 이한솔
// 수정 시간 : 2017.05.28 16:00
// 수정 이유 : 각 메뉴의 실행조건(로그인)에 대한
오류 처리
// 수정자 : 이한솔
// 수정 시간 : 2017.05.29 16:45
// 수정 이유 : 전역변수인 currentSessionID를
Member클래스의 static 변수로 수정
// 수정자 : 이한솔
// 수정 시간 : 2017.05.29 19:00
// 수정 이유 : 각 메뉴의 실행조건(로그인)에 대한
오류 처리2 : isValidID() 추가
// 수정자 : 심민우
// 수정 시간 : 2017.05.29 23:50
// 수정 이유 : showJoinedGroup 부분 오류 수정
// 수정자 : 강석원
```

```
// 수정 시간 : 2017.05.30 01:28
// 수정 이유 : static 변수들 추가. 기타 버그 수정
// 수정자 : 이한솔
// 수정 시간 : 2017.05.30 04:00
// 수정 이유 : 경고 요인 수정. 4주후 투표 삭제 추가.
시간 계산 함수 GetTimeT() 추가.
// 수정자 : 심민우
// 수정 시간 : 2017.05.30 09:00
// 수정 이유 : 유효하지 않은 ID 정보일 때
투표제안에서 무한루프 도는 문제 수정
// 수정자 : 이한솔
// 수정 시간 : 2017.05.31 17:55
// 수정 이유 : 각 메뉴의 기능에 대한 주석 추가.

// 헤더 선언
#include <stdio.h>
#include <string.h>

#include "CreateNewGroup.h"
#include "CreateNewGroupUI.h"
#include "Group.h"
#include "GroupCollection.h"
#include "Member.h"
#include "MemberCollection.h"
#include "OfferVote.h"
#include "OfferVoteUI.h"
#include "ShowFutureVote.h"
#include "ShowFutureVoteUI.h"
#include "ShowNowVote.h"
#include "ShowNowVoteUI.h"
#include "ShowTerminatedVote.h"
#include "ShowTerminatedVoteUI.h"
#include "Vote.h"
#include "VoteCollection.h"
#include "ShowEntireGroup.h"
#include "ShowEntireGroupUI.h"
#include "CreateNewGroup.h"
#include "CreateNewGroupUI.h"
#include "ShowJoinedGroup.h"
#include "ShowJoinedGroupUI.h"
#include "MemberRegistration.h"
#include "MemberRegistrationUI.h"
#include "Logout.h"
#include "LogoutUI.h"
#include "Login.h"
#include "LoginUI.h"
#include "leaveMembership.h"
#include "leaveMembershipUI.h"
#include "AutoDeleteTerminatedVote.h"
#include "AutoDeleteTerminatedVoteUI.h"

// 상수 선언
#define MAX_STRING 32
#define INPUT_FILE_NAME "input.txt"
#define OUTPUT_FILE_NAME "output.txt"
```

```

#define SESSION_VALID 1
#define SESSION_NOID -1
#define SESSION_GUEST 0

// 함수 선언
void doTask();           //메뉴 인식, 실행을 위한 함수

int isValidID();          //guest 세션일 경우(SESSION_GUEST), 현재 세션의 ID가 등록되어있지 않은 경우(SESSION_NOID)
                           //세션이 유효한 경우(SESSION_VALID)
bool operator>(tm time1, tm time2); //시간 값을 비교하기 위한 연산자 재정의
time_t GetTimeT(tm tmTime); //tm 구조체를 time_t로 변환하는 함수

//각 메뉴에 해당하는 함수 선언
void memberRegistration(); //1.1 회원가입
void leaveMembership();   //1.2 회원탈퇴

void login();             //2.1 로그인
void logOut();            //2.2 로그아웃

void offerVote();         //3.1 투표 제안

void showNowVote();       //4.1 현재 진행중인 투표 조회(출력). 직후 투표를 요청한 경우에만 투표가 가능함.
void doVote();            //4.2 투표 ( 현재 진행중인 투표를 조회한 직후에만 가능하도록 구현. 오류제어용 )
void showFutureVote();    //4.3 향후 진행 예정인 투표 조회(출력)
void showTerminatedVote(); //4.4 종료된 투표 조회(출력)

void showEntireGroup();   //5.1 전체 그룹 조회(출력). 직후 그룹가입을 요청한 경우에만 그룹 가입이 가능함.
void joinGroup();          //5.2 그룹가입 ( 전체 그룹을 조회한 직후에만 가능하도록 구현. 오류제어용 )
void createNewGroup();    //5.3 그룹생성
void showJoinedGroup();   //5.4 가입 그룹 조회(출력). 직후 그룹탈퇴를 요청한 경우에만 탈퇴가 가능함.
void leaveGroup();         //5.5 그룹탈퇴 ( 가입 그룹을 조회한 직후에만 가능하도록 구현. 오류제어용 )

void setCurrentTime();    //6.1 현재 시간 설정

void changeSession();     //7.1 세션 변경
void changeToGuestSession(); //7.2 guest 세션으로 변경

void program_exit();      //8.1 프로그램 종료

// 변수 선언

```

```

FILE* in_fp, *out_fp;
tm currentTime;
/* 변수선언및헤더선언 */

int main() {
    in_fp = fopen(INPUT_FILE_NAME, "r+");
    out_fp = fopen(OUTPUT_FILE_NAME, "w+");

    MemberCollection::createEntireMembers();
    GroupCollection::createEntireGroups();
    VoteCollection::createEntireVotes();

    Member::setCurrentSessionID(string ""); //초기
    현재세션 ID를 ""(공백)로 설정

    doTask();

    delete MemberCollection::getEntireMembers();
    delete GroupCollection::getEntireGroups();
    delete VoteCollection::getEntireVotes();

    return 0;
}

void doTask()
{
    // 메뉴 파싱을 위한 level 구분을 위한 변수
    int menu_level_1 = 0, menu_level_2 = 0;

    // 종료 메뉴(8.1)가 입력되기 전까지 반복함
    while (menu_level_1 != 8 || menu_level_2 != 1)
    {
        // 입력파일에서 메뉴 숫자 2개를 읽기
        fscanf(in_fp, "%d %d ", &menu_level_1,
        &menu_level_2);

        switch (menu_level_1)
        {
        case 1:
        {
            switch (menu_level_2)
            {
            case 1:
            {
                memberRegistration();
                break;
            }
            case 2:
            {
                leaveMembership();
                break;
            }
            }
        }
        break;
    }
}


```

```

case 2:
{
    switch (menu_level_2)
    {
        case 1:
        {
            login();
            break;
        }
        case 2:
        {
            logOut();
            break;
        }
    }
    break;
}
case 3:
{
    switch (menu_level_2)
    {
        case 1:
        {
            offerVote();
            break;
        }
    }
    break;
}
case 4:
{
    switch (menu_level_2)
    {
        case 1:
        {
            showNowVote();
            break;
        }
        // implemented in showNowVote, only for error
control
        case 2:
        {
            doVote();
            break;
        }
        case 3:
        {
            showFutureVote();
            break;
        }
        case 4:
        {
            showTerminatedVote();
            break;
        }
    }
}
break;
}
case 5:
{
    switch (menu_level_2)
    {
        case 1:
        {
            showEntireGroup();
            break;
        }
        case 2:
        {
            joinGroup();
            break;
        }
        case 3:
        {
            createNewGroup();
            break;
        }
        case 4:
        {
            showJoinedGroup();
            break;
        }
        case 5:
        {
            leaveGroup();
            break;
        }
    }
}
break;
}
case 6:
{
    switch (menu_level_2)
    {
        case 1:
        {
            setCurrentTime();
            break;
        }
    }
}
break;
}
case 7:
{
    switch (menu_level_2)
    {
        case 1:
        {
            changeSession();
            break;
        }
        case 2:
    }
}

```

```

    {
        changeToGuestSession();
        break;
    }
}
break;
}

case 8:
{
    switch (menu_level_2)
    {
        case 1:
        {
            program_exit();
            break;
        }
        break;
    }
}
}

void memberRegistration()
{
    char name[MAX_STRING], personalNumber[MAX_STRING],
        address[MAX_STRING], ID[MAX_STRING],
        password[MAX_STRING];

    // 입력 형식 : 이름, 주민번호, 주소, ID,
    // 비밀번호를 파일로부터 읽음
    fscanf(in_fp, "%s %s %s %s %s\n", name,
           personalNumber, address, ID, password);

    MemberRegistrationUI* UI = nullptr;
    MemberRegistration* Control = new
    MemberRegistration(&UI);

    UI->requestNewMemberRegistration((string)name,
        (string)personalNumber, (string)address, (string)ID,
        (string)password);

    delete Control;
}

void leaveMembership()
{
    int sessionValid = isValidID();
    if (sessionValid != SESSION_VALID)
    {
        fprintf(out_fp, "1.2. 회원탈퇴\n");
        printf("1.2. 회원탈퇴\n");
        if (sessionValid == SESSION_GUEST)
        {
            fprintf(out_fp, "> 로그인 후에 실행 가능한
메뉴입니다.\n");
        }
        printf("> 로그인 후에 실행 가능한
메뉴입니다.\n");
    }
    else if (sessionValid == SESSION_NOID)
    {
        fprintf(out_fp, "> 존재하지 않는
ID입니다.\n");
        printf("> 존재하지 않는 ID입니다.\n");
    }
    return;
}

LeaveMembershipUI* UI = nullptr;
LeaveMembership* Control = new
LeaveMembership(&UI);

if (UI-
>requestLeaveMembership(Member::getCurrentSessionID(
)))
{
    // 회원탈퇴 성공. 세션 변경
    changeToGuestSession();
}

delete Control;
}

void logIn()
{
    char ID[MAX_STRING], password[MAX_STRING];

    // 입력 형식 : 이름, 주민번호, 주소, ID,
    // 비밀번호를 파일로부터 읽음
    fscanf(in_fp, "%s %s\n", ID, password);

    // 해당 기능 수행
    LoginUI* UI = nullptr;
    Login* Control = new Login(&UI);

    if (UI->requestAttemptLogin((string)ID,
        (string)password))
    {
        //로그인 성공. 세션 변경
        Member::setCurrentSessionID(ID);
    }
    delete Control;
}

void logOut()
{
    int sessionValid = isValidID();
    if (sessionValid != SESSION_VALID)
    {
        fprintf(out_fp, "2.2. 로그아웃\n");
        printf("2.2. 로그아웃\n");
        if (sessionValid == SESSION_GUEST)
        {
            fprintf(out_fp, "> 로그인 후에 실행 가능한
메뉴입니다.\n");
        }
        printf("> 로그인 후에 실행 가능한
메뉴입니다.\n");
    }
}

```

```

메뉴입니다.\n");
printf("> 로그인 후에 실행 가능한
메뉴입니다.\n");
}
else if (sessionValid == SESSION_NOID)
{
    fprintf(out_fp, "> 존재하지 않는
ID입니다.\n");
    printf("> 존재하지 않는 ID입니다.\n");
}
return;
}

LogoutUI* UI = nullptr;
Logout* Control = new Logout(&UI);

UI->requestLogout(Member::getCurrentSessionID());
//로그아웃. 게스트로 세션 변경
changeToGuestSession();

delete Control;
}

void offerVote()
{
char voteTopic[MAX_STRING];
int numOfItems;

tm startTime = { 0 };
tm finishTime = { 0 };

fscanf(in_fp, "%s %d ", voteTopic, &numOfItems);
fscanf(in_fp, "%d:%d:%d:%d:%d ",
&startTime.tm_year, &startTime.tm_mon,
&startTime.tm_mday, &startTime.tm_hour,
&startTime.tm_min);
fscanf(in_fp, "%d:%d:%d:%d:%d ",
&finishTime.tm_year, &finishTime.tm_mon,
&finishTime.tm_mday, &finishTime.tm_hour,
&finishTime.tm_min);

int sessionValid = isValidID();
if (sessionValid != SESSION_VALID)
{
    fprintf(out_fp, "3.1. 투표 제안\n");
    printf("3.1. 투표 제안\n");
    if (sessionValid == SESSION_GUEST)
    {
        fprintf(out_fp, "> 로그인 후에 실행 가능한
메뉴입니다.\n");
        printf("> 로그인 후에 실행 가능한
메뉴입니다.\n");
    }
    else if (sessionValid == SESSION_NOID)
    {
        fprintf(out_fp, "> 존재하지 않는

```

```

ID입니다.\n");
        printf("> 존재하지 않는 ID입니다.\n");
    }
    return;
}

OfferVoteUI *UI = nullptr;
OfferVote *Control = new OfferVote(&UI);

UI-
>requestOfferVote(Member::getCurrentSessionID(),
string(voteTopic), numOfItems, startTime,
finishTime);

delete Control;
}

void showNowVote()
{
int sessionValid = isValidID();
if (sessionValid != SESSION_VALID)
{
    fprintf(out_fp, "4.1. 현재 진행 중인 투표
리스트\n");
    printf("4.1. 현재 진행 중인 투표 리스트\n");
    if (sessionValid == SESSION_GUEST)
    {
        fprintf(out_fp, "> 로그인 후에 실행 가능한
메뉴입니다.\n");
        printf("> 로그인 후에 실행 가능한
메뉴입니다.\n");
    }
    else if (sessionValid == SESSION_NOID)
    {
        fprintf(out_fp, "> 존재하지 않는
ID입니다.\n");
        printf("> 존재하지 않는 ID입니다.\n");
    }
    return;
}

ShowNowVoteUI *UI = nullptr;
ShowNowVote *Control = new ShowNowVote(&UI);
int menu_level_1 = 0, menu_level_2 = 0;
int currentFileOffset;
char voteTopic[MAX_STRING];
int selectedItem;

UI-
>requestShowNowVote(Member::getCurrentSessionID());

currentFileOffset = ftell(in_fp);
fscanf(in_fp, "%d %d", &menu_level_1,
&menu_level_2);

if (!(menu_level_1 == 4 && menu_level_2 == 2))

```

```

{
    fseek(in_fp, currentFileOffset, SEEK_SET);
}
else
{
    fscanf(in_fp, "%s %d", voteTopic,
&selectedItem);
    UI->requestDoVote(Member::getCurrentSessionID(),
string(voteTopic), selectedItem);
}

delete Control;
}

void doVote() //현재 진행중인 투표 조회를 직전에
하지 않고 투표하기를 실행했을 때 오류제어용
{
    char voteTopic[MAX_STRING];
    int selectedItem;
    fscanf(in_fp, "%s %d", voteTopic, &selectedItem);

    fprintf(out_fp, "4.2 투표\n");
    fprintf(out_fp, "> 현재 진행중인 투표 조회를 먼저
해야합니다.\n");

    printf("4.2 투표\n");
    printf("> 현재 진행중인 투표 조회를 먼저
해야합니다.\n");
}

void showFutureVote()
{
    int sessionValid = isValidID();
    if (sessionValid != SESSION_VALID)
    {
        fprintf(out_fp, "4.3. 향후 진행 예정인 투표
리스트\n");
        printf("4.3. 향후 진행 예정인 투표 리스트\n");
        if (sessionValid == SESSION_GUEST)
        {
            fprintf(out_fp, "> 로그인 후에 실행 가능한
메뉴입니다.\n");
            printf("> 로그인 후에 실행 가능한
메뉴입니다.\n");
        }
        else if (sessionValid == SESSION_NOID)
        {
            fprintf(out_fp, "> 존재하지 않는
ID입니다.\n");
            printf("> 존재하지 않는 ID입니다.\n");
        }
        return;
    }

    ShowFutureVoteUI *UI = nullptr;
    ShowFutureVote *Control = new ShowFutureVote(&UI);
}

```

```

UI-
>requestShowFutureVote(Member::getCurrentSessionID()
);

delete Control;
}

void showTerminatedVote()
{
    int sessionValid = isValidID();
    if (sessionValid != SESSION_VALID)
    {
        fprintf(out_fp, "4.4. 종료된 투표 리스트\n");
        printf("4.4. 종료된 투표 리스트\n");
        if (sessionValid == SESSION_GUEST)
        {
            fprintf(out_fp, "> 로그인 후에 실행 가능한
메뉴입니다.\n");
            printf("> 로그인 후에 실행 가능한
메뉴입니다.\n");
        }
        else if (sessionValid == SESSION_NOID)
        {
            fprintf(out_fp, "> 존재하지 않는
ID입니다.\n");
            printf("> 존재하지 않는 ID입니다.\n");
        }
        return;
    }

    ShowTerminatedVoteUI *UI = nullptr;
    ShowTerminatedVote *Control = new ShowTerminatedVote(&UI);

    UI-
>requestShowTerminatedVote(Member::getCurrentSession
ID());

delete Control;
}

void showEntireGroup()
{
    int sessionValid = isValidID();
    if (sessionValid != SESSION_VALID)
    {
        fprintf(out_fp, "5.1. 전체그룹 조회\n");
        printf("5.1. 전체그룹 조회\n");
        if (sessionValid == SESSION_GUEST)
        {
            fprintf(out_fp, "> 로그인 후에 실행 가능한
메뉴입니다.\n");
            printf("> 로그인 후에 실행 가능한
메뉴입니다.\n");
        }
    }
}

```

```

else if (sessionValid == SESSION_NOID)
{
    fprintf(out_fp, "> 존재하지 않는
ID입니다.\n");
    printf("> 존재하지 않는 ID입니다.\n");
}
return;
}

ShowEntireGroupUI* UI = nullptr;
ShowEntireGroup *Control = new
ShowEntireGroup(&UI);

UI->requestShowEntireGroup();

int menu_level_1 = 0, menu_level_2 = 0;
int currentFileOffset;
char groupName[MAX_STRING];

currentFileOffset = ftell(in_fp);
fscanf(in_fp, "%d %d", &menu_level_1,
&menu_level_2);

if (!(menu_level_1 == 5 && menu_level_2 == 2))
{
    fseek(in_fp, currentFileOffset, SEEK_SET);
}
else
{
    fscanf(in_fp, "%s", groupName);
    UI->selectGroup(Member::getCurrentSessionID(),
string(groupName));
}

delete Control;
}

void joinGroup()
{
char groupName[MAX_STRING];
fscanf(in_fp, "%s", groupName);

fprintf(out_fp, "5.1. 전체그룹 조회\n");
fprintf(out_fp, "> 전체 그룹 조회를 먼저
해야합니다.\n");

printf("5.1. 전체그룹 조회\n");
printf("> 전체 그룹 조회를 먼저 해야합니다.\n");
}

void createNewGroup()
{
char groupName[MAX_STRING];

fscanf(in_fp, "%s", groupName);
}

```

```

int sessionValid = isValidID();
if (sessionValid != SESSION_VALID)
{
    fprintf(out_fp, "5.3. 그룹 생성\n");
    printf("5.3. 그룹 생성\n");
    if (sessionValid == SESSION_GUEST)
    {
        fprintf(out_fp, "> 로그인 후에 실행 가능한
메뉴입니다.\n");
        printf("> 로그인 후에 실행 가능한
메뉴입니다.\n");
    }
    else if (sessionValid == SESSION_NOID)
    {
        fprintf(out_fp, "> 존재하지 않는
ID입니다.\n");
        printf("> 존재하지 않는 ID입니다.\n");
    }
    return;
}

CreateNewGroupUI *UI = nullptr;
CreateNewGroup *Control = new CreateNewGroup(&UI);

//UI-
>requestCreateNewGroup(Member::getCurrentSessionID(),
(string)groupName);
UI->requestCreateNewGroup((string)groupName,
Member::getCurrentSessionID());

delete Control;
}

void showJoinedGroup()
{
int sessionValid = isValidID();
if (sessionValid != SESSION_VALID)
{
    fprintf(out_fp, "5.4. 가입그룹 조회\n");
    printf("5.4. 가입그룹 조회\n");
    if (sessionValid == SESSION_GUEST)
    {
        fprintf(out_fp, "> 로그인 후에 실행 가능한
메뉴입니다.\n");
        printf("> 로그인 후에 실행 가능한
메뉴입니다.\n");
    }
    else if (sessionValid == SESSION_NOID)
    {
        fprintf(out_fp, "> 존재하지 않는
ID입니다.\n");
        printf("> 존재하지 않는 ID입니다.\n");
    }
    return;
}

```

```

ShowJoinedGroupUI *UI = nullptr;
ShowJoinedGroup *Control = new ShowJoinedGroup(&UI);

UI-
>requestShowJoinedGroup(Member::getCurrentSessionID());

int menu_level_1 = 0, menu_level_2 = 0;
int currentFileOffset;

currentFileOffset = ftell(in_fp);
fscanf(in_fp, "%d %d", &menu_level_1,
&menu_level_2);

if (!(menu_level_1 == 5 && menu_level_2 == 5))
{
    fseek(in_fp, currentFileOffset, SEEK_SET);
}
else
{
    UI-
>requestLeaveGroup(Member::getCurrentSessionID());
}

delete Control;
}

void leaveGroup()
{
    fprintf(out_fp, "5.5 그룹 탈퇴\n");
    fprintf(out_fp, "> 현재 가입중인 그룹 조회를 먼저
해야합니다.\n");

    printf("5.5 그룹 탈퇴\n");
    printf("> 현재 가입중인 그룹 조회를 먼저
해야합니다.\n");
}

// Function : void setCurrentTime()
// Description : 현재 시간을 설정
//           입력 받은 시간 값으로 년,월,일,시간,분을
//           새로 설정.
//           전체 투표 리스트를 순회하며 현재 시간에
//           맞는 상태로 변경.
//           VOTETYPE_FUTURE(향후 진행 예정) : 투표
//           시작시간 > 현재 시간
//           VOTETYPE_NOEVOTING(현재 진행중) : 투표
//           종료시간 > 현재 시간 >= 투표 시작시간
//           VOTETYPE_TERMINATED(종료) : 현재시간 >=
//           종료된 투표
//           VOTETYPE_TERMINATED로 설정된 투표의
//           리스트를 순회하여 종료후 4주 이상 지난 경우 투표
//           삭제.
// Parameter : 없음

```

```

// Return Value : 없음
void setCurrentTime()
{
    fscanf(in_fp, "%d:%d:%d:%d:%d",
    &currentTime.tm_year, &currentTime.tm_mon,
    &currentTime.tm_mday, &currentTime.tm_hour,
    &currentTime.tm_min);

    fprintf(out_fp, "6.1. 현재시간 설정\n");
    fprintf(out_fp, "> %04d:%02d:%02d:%02d:%02d\n",
    currentTime.tm_year, currentTime.tm_mon,
    currentTime.tm_mday, currentTime.tm_hour,
    currentTime.tm_min);

    printf("6.1. 현재시간 설정\n");
    printf("> %04d:%02d:%02d:%02d:%02d\n",
    currentTime.tm_year, currentTime.tm_mon,
    currentTime.tm_mday, currentTime.tm_hour,
    currentTime.tm_min);

    vector<Vote*>* allVoteList =
    VoteCollection::getEntireVotes()->getVoteList();
    size_t Size = allVoteList->size();
    vector<Vote*>::iterator iter;

    for(iter = allVoteList->begin(); iter != allVoteList->end(); iter++)
    {
        tm startTime = (*iter)->getStartTime();
        tm finishTime = (*iter)->getFinishTime();

        if (startTime > currentTime)
        { //투표 시작시간이 현재시간보다 나중이면 향후
        진행 예정 투표로 변경
            (*iter)->setType(VOTETYPE_FUTURE);
        }
        else if (finishTime > currentTime)
        { //투표 종료시간이 현재시간보다
        나중이면(시작시간 >= 현재시간 이면서) 현재 진행중인
        투표로 변경
            (*iter)->setType(VOTETYPE_NOWVOTING);
        }
        else
        { //투표 종료시간 >= 현재시간이면 종료된 투표로
        변경. 단, 종료 후 4주가 지났다면 투표 삭제.
            (*iter)->setType(VOTETYPE_TERMINATED);
        }
    }

    AutoDeleteTerminatedVoteUI *UI = nullptr;
    AutoDeleteTerminatedVote *control = new
    AutoDeleteTerminatedVote(&UI);

    UI->requestAutoDeleteTerminatedVote(currentTime);

    delete UI;
}

```

```

// Function : void changeSession()
// Description : 현재 세션을 변경
//           입력 받은 ID로 현재 세션을 변경
//           전체 멤버 리스트에 ID가 존재하고, 로그인이
// 되어 있는 경우 현재 세션의 ID를 변경.
// Parameter : 없음
// Return Value : 없음
void changeSession()
{
    char requestedID[MAX_STRING];

    fscanf(in_fp, "%s", requestedID);

    Member *changeMember =
    MemberCollection::getEntireMembers()-
    >findMember((string)requestedID);

    fprintf(out_fp, "7.1. Session 변경\n");
    printf("7.1. Session 변경\n");

    if (changeMember == nullptr)
    {
        fprintf(out_fp, "> 존재하지 않는 ID입니다.\n");
        printf("> 존재하지 않는 ID입니다.\n");
        return;
    }

    if (changeMember->isLogined())
    {
        Member::setCurrentSessionID(changeMember-
        >getMemberID());

        fprintf(out_fp, "> %s\n", changeMember-
        >getMemberID().c_str());
        printf("> %s\n", changeMember-
        >getMemberID().c_str());
    }
    else
    {
        fprintf(out_fp, "> %s는 현재 로그인되어 있지
        않습니다.\n", changeMember->getMemberID().c_str());
        printf("> %s는 현재 로그인되어 있지
        않습니다.\n", changeMember->getMemberID().c_str());
    }
}

// Function : void changeToGuestSession()
// Description : 현재 세션의 ID값을 guest
// 세션(공백 : "")으로 변경.
// Parameter : 없음
// Return Value : 없음
void changeToGuestSession()
{
    Member::setCurrentSessionID(string(""));
    printf(out_fp, "7.2. guest session으로 변경\n");
}

```

```

printf("7.2. guest session으로 변경\n");
}

// Function : void program_exit()
// Description : 프로그램을 종료한다고 출력.
// Parameter : 없음
// Return Value : 없음
void program_exit()
{
    fprintf(out_fp, "8.1. 종료\n");
    printf("8.1. 종료\n");
}

// Function : int isValidID()
// Description : 현재 세션의 ID가 guest("")인지,
// 혹은 ID가 등록되어 있는지 판별
// Parameter : 없음
// Return Value :
//           1) SESSION_GUEST : 현재 세션이 guest인
// 경우 로그인이 필요하다는 문구를 출력
//           2) SESSION_NOID : 현재 세션의 ID가
// MemberCollection::entireMembers에 등록되어 있지 않은
// 경우
//           3) SESSION_VALID : 위의 1), 2)에 해당하지
// 않는 경우
int isValidID()
{
    if (Member::getCurrentSessionID() == string(""))
    {
        return SESSION_GUEST;
    }
    else if (MemberCollection::getEntireMembers()-
    >findMember(Member::getCurrentSessionID()) ==
    nullptr)
    {
        return SESSION_NOID;
    }
    else
        return SESSION_VALID;
}

// Function : bool operator>(tm time1, tm time2)
// Description : parameter로 받은 time1, time2
// 변수의 값을 비교하여
//           time1 > time2 인지 판별
// Parameter : tm time1 - 비교할 좌측 시간 변수,
//           tm time2 - 비교할 우측 시간 변수
// Return Value :
//           1) time1이 큰 경우 true
//           2) time2가 큰 경우 false를 반환
bool operator>(tm time1, tm time2)
{
    int year1 = time1.tm_year, year2 = time2.tm_year;
    if (year1 > year2) return true;
    else if (year1 < year2) return false;
    else

```

```

{
    int mon1 = time1.tm_mon, mon2 = time2.tm_mon;
    if (mon1 > mon2) return true;
    else if (mon1 < mon2) return false;
    else
    {
        int mday1 = time1.tm_mday, mday2 =
time2.tm_mday;
        if (mday1 > mday2) return true;
        else if (mday1 < mday2) return false;
        else
        {
            int hour1 = time1.tm_hour, hour2 =
time2.tm_hour;
            if (hour1 > hour2) return true;
            else if (hour1 < hour2) return false;
            else
            {
                int min1 = time1.tm_min, min2 =
time2.tm_min;
                if (min1 > min2) return true;
                else return false;
            }
        }
    }
}

// Function : time_t GetTimeT(tm tmTime)
// Description : parameter로 받은 tmTime 변수를
//               해당 값과 일치하는 정보를 갖는 time_t
// 타입의 변수로 변환
// Parameter : tm tmTime - time_t 변수로 변환할 tm
// 구조체 변수
// Return Value : 변환된 time_t 변수를 반환
time_t GetTimeT(tm tmTime)
{
    tm curr = { 0 };
    curr.tm_year = tmTime.tm_year - 1900;
    curr.tm_mon = tmTime.tm_mon - 1;
    curr.tm_mday = tmTime.tm_mday;
    curr.tm_hour = tmTime.tm_hour;
    curr.tm_min = tmTime.tm_min;

    return mktime(&curr);
}

```