



프로젝트 최종 보고서

과목명	사용자인터페이스및실습(가)
교수명	최지웅
팀 장	20212985 이정우
팀원 1	20232217 송채원
팀원 2	20232372 장다은
링크	https://youtu.be/U1uBA7st3HA

목차

I.	최종 기획안 및 초기 작업분해표	4
1.	기획안 요약.....	4
2.	초기 작업 분해표.....	5
II.	주차별 진도보고서	6
1.	10주차 진도보고서	6
2.	11주차 진도보고서	8
3.	12주차 진도보고서	11
4.	13주차 진도보고서	14
5.	14주차 진도보고서	16
III.	기능별 부연설명	21
1.	이정우	21
1)	유세인트 연동 로그인.....	21
2)	친구 목록 불러오기.....	23
3)	스탑워치 기능	24
2.	송채원	25
1)	자동 강의 일정 추가 (Calendar Activity)	25
2)	캘린더/강의 일정/사용자 일정 관리 기능 (Calendar Activity)	27
3)	카테고리 표시 및 할 일 관리 기능 (Calendar Activity).....	29
4)	벼락치기 마법사 (Calendar Activity)	30
5)	Home Activity.....	31
3.	장다은	32

1)	닉네임 변경.....	32
2)	친구 요청 수락	32
3)	카테고리 목록.....	33
4)	카테고리 추가, 수정.....	35
5)	로그아웃	38
IV.	최종 작업분해표	39
1.	최종 작업분해표	39
2.	Firebase 사용 개요.....	39

I. 최종 기획안 및 초기 작업분해표

1. 기획안 요약

1) 프로젝트 목적

대학생들을 위한 종합적인 학술 어플리케이션의 부재를 해소하기 위해, 숭실대학교 학생들을 대상으로 한 종합적 학술 어플을 만들어보고자 한다.

2) 주요 기능

- **유세인트 연동 로그인:**

숭실대학교 유세인트 계정을 활용한 간편 인증 및 로그인 기능을 제공한다.

- **강의 일정 관리:**

유세인트 강의 정보를 기반으로 강의 일정과 학사 계획을 관리한다.

- **친구 추가 기능:**

숭실대학교 학생 간 친구 추가 및 관리 기능을 제공한다.

- **할 일 기록:**

개인 할 일을 기록하고 체계적으로 관리할 수 있다.

- **그룹 할 일 연동:**

사용자가 지정한 그룹 멤버 간 할 일을 공유하고 협업할 수 있다.

- **총 공부 시간 기록:**

타이머 기능을 통해 사용자의 총 공부 시간을 자동으로 기록한다.

- **과목별 공부 시간 랭킹 제공:**

과목별 누적 공부 시간을 기준으로 순위를 제공하여 학습 동기를 부여한다.

2. 초기 작업 분해표

06

작업분해표

이정우

- 로그인
- 친구
- 기록

주요메뉴	화면구성/기능	세부설명	담당자	○ 완료 △ 진행중
				진행도
로그인	xml 생성	시작화면	이정우	
		로그인화면		
		초기 설정 화면		
친구	xml 생성	유세인트 연동 로그인		
		별명, 프로필 사진 설정		
		친구 목록 화면		
기록	기능 개발	친구 추가 dialog 화면		
		친구목록 불러오고 공부시간 순위 매기기		
		친구 추가 요청 전송		
	xml 생성	과목 목록 화면		
		과목별 공부시간 순위 화면		
		스탑워치 화면		
	기능 개발	유세인트 강의목록 불러오기		
		(개인) 과목별 공부시간 불러오기		
		(랭킹) 과목별 공부시간 불러오고 순위 매기기		
		스탑워치 기능 구현		

06

작업분해표

송채원

- 홈
- 캘린더

주요메뉴	화면구성/기능	세부설명	담당자	○ 완료 △ 진행중
				진행도
홈	xml 생성	홈화면	송채원	
		하단 메뉴바		
		DB에서 종합 데이터 가져오기		
캘린더	xml 생성	캘린더 및 to-do list 화면		
		일정추가 dialog 화면		
		강의수정 dialog 화면		
		일정수정 dialog 화면		
		벼락치기 마법사 dialog 화면		
	기능 개발	DB 데이터를 캘린더, to-do list에 표시		
		강의(일정) 수정, 추가		
		벼락치기 마법사		

06

작업분해표

장다은

- 설정 (관리)

주요메뉴	화면구성/기능	세부설명	담당자	○ 완료 △ 진행중
				진행도
설정	xml 생성	설정 화면	장다은	
		친구 요청 수락 dialog 화면		
		닉네임 변경 dialog 화면		
		카테고리 관리 화면		
		카테고리 삭제 dialog 화면		
		카테고리 추가, 수정 화면		
	기능 개발	친구 요청 수락		
		이미지 변경, 닉네임 변경		
		카테고리 삭제		
		카테고리 추가, 수정		

II. 주차별 진도보고서

1. 10주차 진도보고서

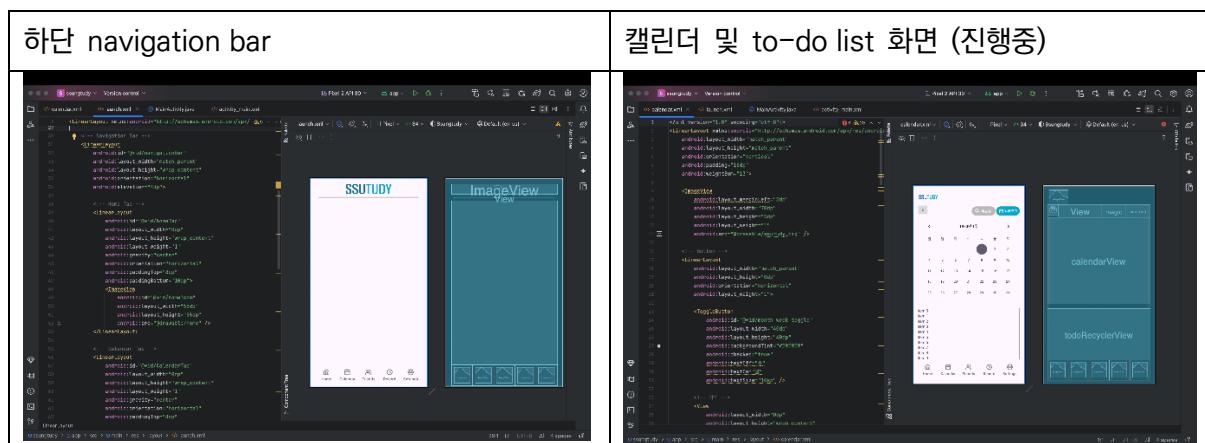
1) 이정우

주요메뉴	화면구성/기능	세부설명	담당자	○	완료
				△	진행중
로그인	xml 생성	시작화면	이정우	○	24.11.6.
		로그인화면		○	24.11.6.
		초기 설정 화면		○	24.11.6.
	기능 개발	유세인트 연동 로그인 별명, 프로필 사진 설정		△	
친구	xml 생성	친구 목록 화면			
		친구 추가 dialog 화면			
	기능 개발	친구목록 불러오고 공부시간 순위 매기기 친구 추가 요청 전송			
기록	xml 생성	과목 목록 화면	이정우		
		과목별 공부시간 순위 화면			
		스탑워치 화면			
	기능 개발	유세인트 강의목록 불러오기 (개인) 과목별 공부시간 불러오기 (랭킹) 과목별 공부시간 불러오고 순위 매기기 스탑워치 기능 구현			

시작화면	로그인화면
초기설정화면	유세인트 연동 로그인 (진행중)

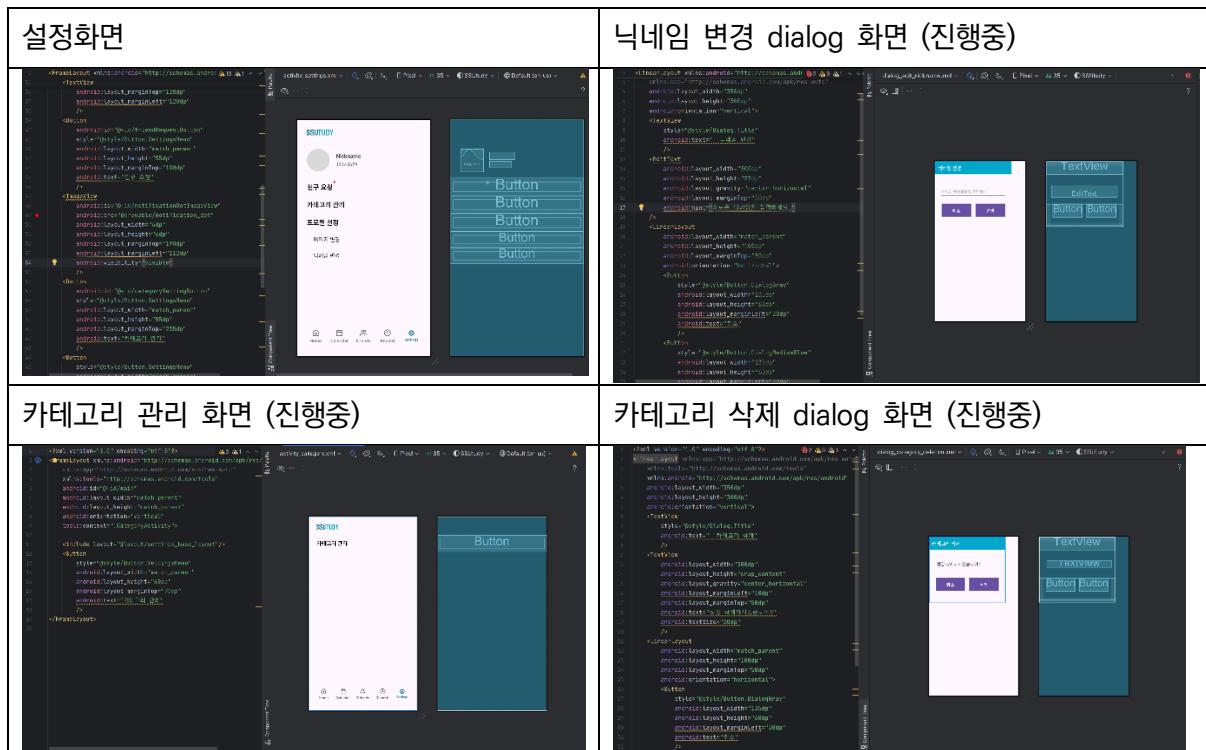
2) 송채원

주요메뉴	화면구성/기능	세부설명	담당자	진행도	완료일자
				○ 완료	△ 진행중
홈	xml 생성	홈화면	송채원		
		하단 navigation bar		○	24.11.08.
	기능 개발	DB에서 종합 데이터 가져오기			
		캘린더 및 to-do list 화면		△	
		일정추가 dialog 화면			
		강의수정 dialog 화면			
		일정수정 dialog 화면			
	xml 생성	벼락치기 마법사 dialog 화면			
		DB 데이터를 캘린더, to-do list에 표시			
		강의(일정) 수정,추가			
	기능 개발	벼락치기 마법사			



3) 장다은

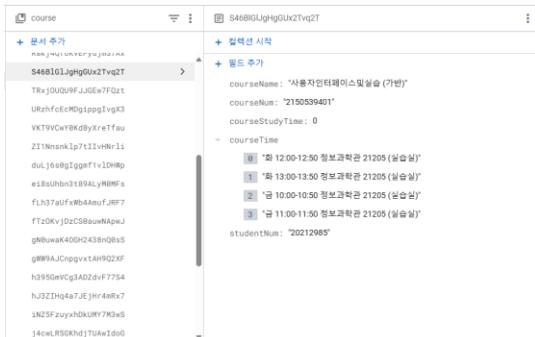
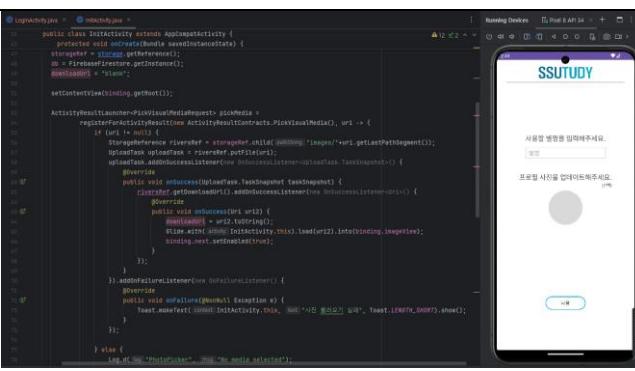
주요메뉴	화면구성/기능	세부설명	담당자	진행도	완료일자
				○ 완료	△ 진행중
설정	xml 생성	설정 화면	장다은	○	2024.11.10.
		친구 요청 수락 dialog 화면			
		닉네임 변경 dialog 화면		△	
		카테고리 관리 화면		△	
		카테고리 삭제 dialog 화면		△	
		카테고리 추가, 수정 화면			
	기능 개발	친구 요청 수락			
		이미지 변경, 닉네임 변경			
		카테고리 삭제			
		카테고리 추가, 수정			



2. 11주차 진도보고서

1) 이정우

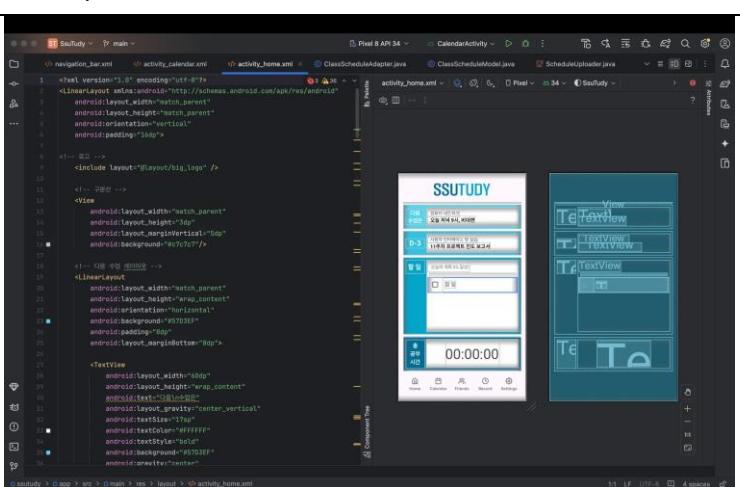
주요메뉴	화면구성/기능	세부설명	담당자	진행도		완료일자
				완료	진행중	
로그인	xml 생성	시작화면	이정우			24.11.6.
		로그인화면				24.11.6.
		초기 설정 화면				24.11.6.
	기능 개발	유세인트 연동 로그인				24.11.17
		별명, 프로필 사진 설정				24.11.17
친구	xml 생성	친구 목록 화면				
		친구 추가 dialog 화면				
	기능 개발	친구목록 불러오고 공부시간 순위 매기기				
		친구 추가 요청 전송				
기록	xml 생성	과목 목록 화면				
		과목별 공부시간 순위 화면				
		스탑워치 화면				
	기능 개발	유세인트 강의목록 불러오기				
		(개인) 과목별 공부시간 불러오기				
		(랭킹) 과목별 공부시간 불러오고 순위 매기기				
		스탑워치 기능 구현				

<h3>유세인트 연동 로그인 (+강의 목록 스크래핑)</h3> 	<h3>별명, 프로필 사진 설정</h3> 
--	---

2) 송재원

주요메뉴	화면구성/기능	세부설명	담당자	○	완료	
				△	진행중	
홈	xml 생성	홈화면	송재원	○	24.11.15.	
		하단 navigation bar		○	24.11.08.	
	기능 개발	DB에서 종합 데이터 가져오기				
	xml 생성	캘린더 및 to-do list 화면		○	24.11.14.	
		일정추가 dialog 화면				
캘린더		강의수정 dialog 화면				
		일정수정 dialog 화면				
		벼락치기 마법사 dialog 화면				
기능 개발	DB 데이터를 캘린더, to-do list에 표시					
	강의(일정) 수정,추가					
	벼락치기 마법사					

- 10주차 진행 중이던 activity_calendar.xml 생성 완료

<h3>activity_home.xml 생성</h3> 

3) 장다은

주요메뉴	화면구성/기능	세부설명	담당자	○	완료	완료일자
				△	진행중	
설정	xml 생성	설정 화면	장다은	○		24.11.10.
		친구 요청 수락 dialog 화면		○		24.11.17.
		닉네임 변경 dialog 화면		○		24.11.14.
		카테고리 관리 화면		○		24.11.16.
		카테고리 삭제 dialog 화면		○		24.11.11.
		카테고리 추가, 수정 화면		○		24.11.17.
	기능 개발	친구 요청 수락				
		이미지 변경, 닉네임 변경				
		카테고리 삭제				
		카테고리 추가, 수정				

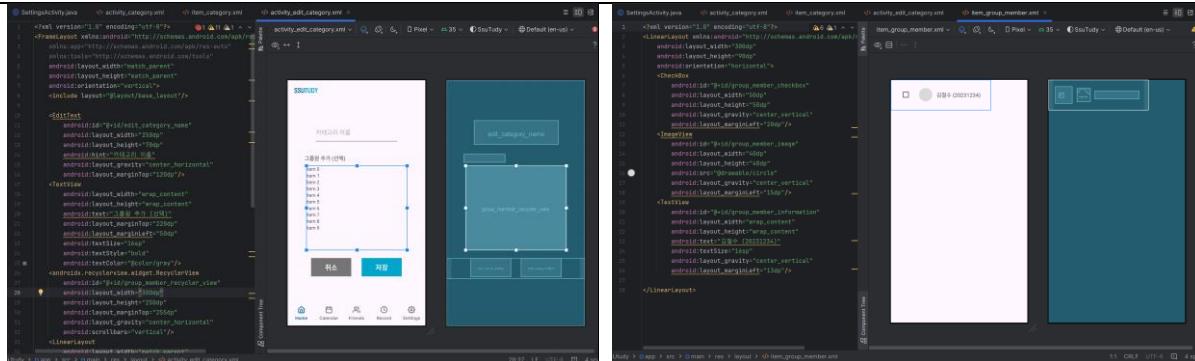
설정화면

닉네임 변경 Dialog 화면

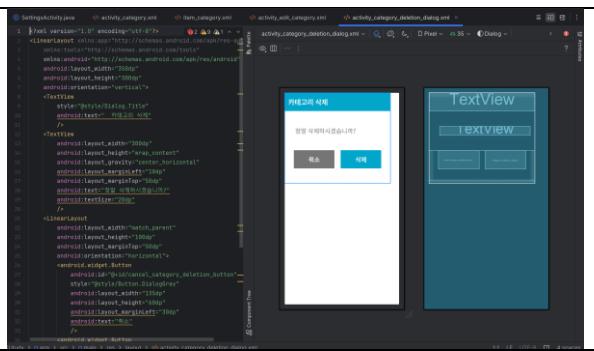
친구 요청 수락 Dialog 화면

카테고리 관리 화면

카테고리 추가/수정 화면



카테고리 삭제 Dialog 화면

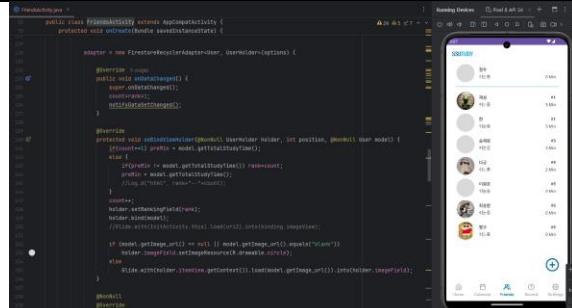


3. 12주차 진도보고서

1) 이정우

주요메뉴	화면구성/기능	세부설명	담당자	○	완료	추가
				△	진행중	삭제
로그인	xml 생성	시작화면	이정우	○	24.11.6.	
		로그인화면		○	24.11.6.	
		초기 설정화면		○	24.11.6.	
	기능 개발	유세인트 연동 로그인		○	24.11.17.	
		별명, 프로필 사진 설정		○	24.11.17.	
친구	xml 생성	친구 목록 화면		○	24.11.23.	
		친구 추가 dialog 화면		○	24.11.18.	
	기능 개발	친구목록 불러오고 공부시간 순위 매기기		○	24.11.23.	
		친구 추가 요청 전송		○	24.11.24.	
기록	xml 생성	과목 목록 화면				
		과목별 공부시간 순위 화면				
		스탑워치 화면				
	기능 개발	유세인트 강의목록 불러오기				
		(개인) 과목별 공부시간 불러오기				
		(랭킹) 과목별 공부시간 불러오고 순위 매기기				
		스탑워치 기능 구현				

친구 목록 불러오기



```

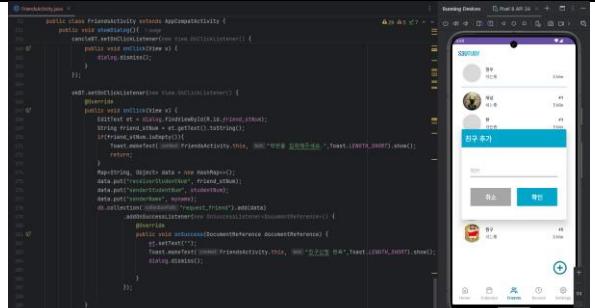
public class FriendsActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_friends);

        ListView lv = findViewById(R.id.listView);
        lv.setAdapter(new FriendsAdapter(this, UserList.getFriendsList()));

        lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView parent, View view, int position, long id) {
                Intent intent = new Intent(getApplicationContext(), DetailActivity.class);
                intent.putExtra("id", UserList.getFriendsList().get(position).getId());
                intent.putExtra("name", UserList.getFriendsList().get(position).getName());
                intent.putExtra("image", UserList.getFriendsList().get(position).getImage());
                intent.putExtra("status", UserList.getFriendsList().get(position).getStatus());
                intent.putExtra("last_update", UserList.getFriendsList().get(position).getLastUpdate());
                startActivity(intent);
            }
        });
    }
}

```

친구 추가 요청 전송



```

public class FriendsActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_friends);

        FirebaseFirestore db = FirebaseFirestore.getInstance();
        db.collection("friends").document("request").set("request", true);
    }
}

public class FriendRequestActivity extends AppCompatActivity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_friend_request);

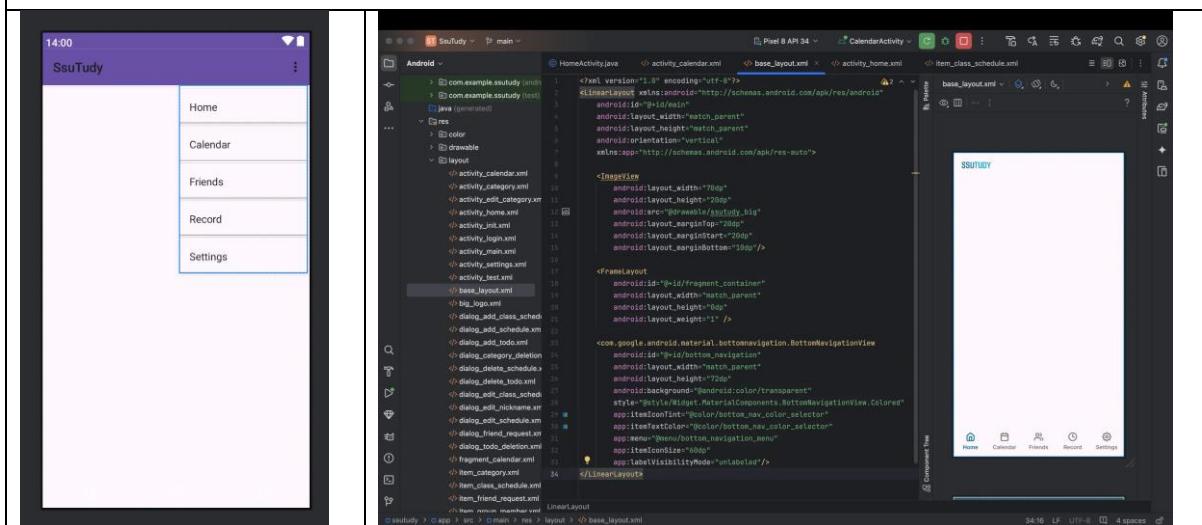
        FirebaseFirestore db = FirebaseFirestore.getInstance();
        db.collection("friends").document("request").set("request", false);
    }
}

```

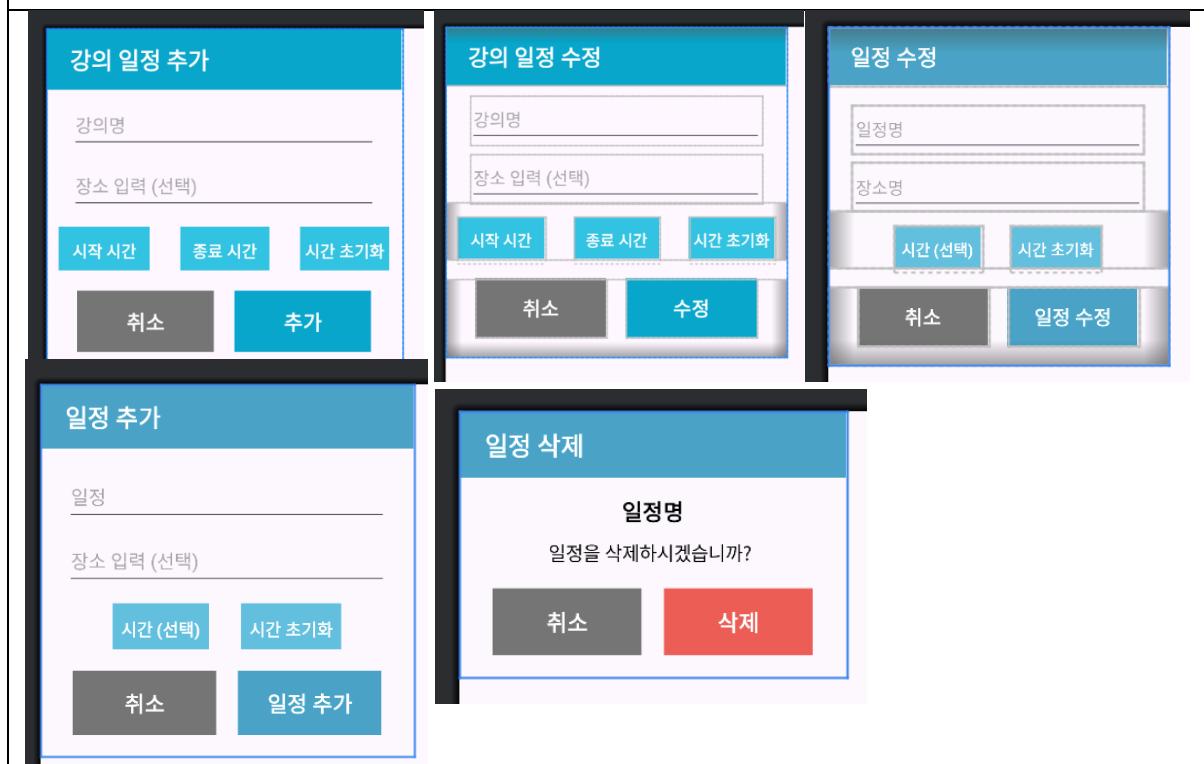
2) 송채원

주요메뉴	화면구성/기능	세부설명	담당자	진행도	완료일자
				○ 완료	
홈	xml 생성	홈화면	송채원	○	24.11.15.
		하단 navigation bar		○	24.11.08.
		수업 내용대로 navigation bar 수정하기		○	24.11.20
		base layout 만들기		○	24.11.20
	기능 개발	DB에서 종합 데이터 가져오기			
캘린더	xml 생성	캘린더 및 to-do list 화면		○	24.11.14.
		일정추가 dialog 화면		○	24.11.22
		강의수정 dialog 화면		○	24.11.24
		일정수정 dialog 화면		○	24.11.24
		벼락치기 마법사 dialog 화면			
	기능 개발	DB 데이터를 캘린더, to-do list에 표시			
		강의(일정) 수정, 추가			
		벼락치기 마법사			

Navigation bar 수정 및 base layout 생성



강의 일정 및 일반 일정 관련 Dialog 생성



3) 장다은

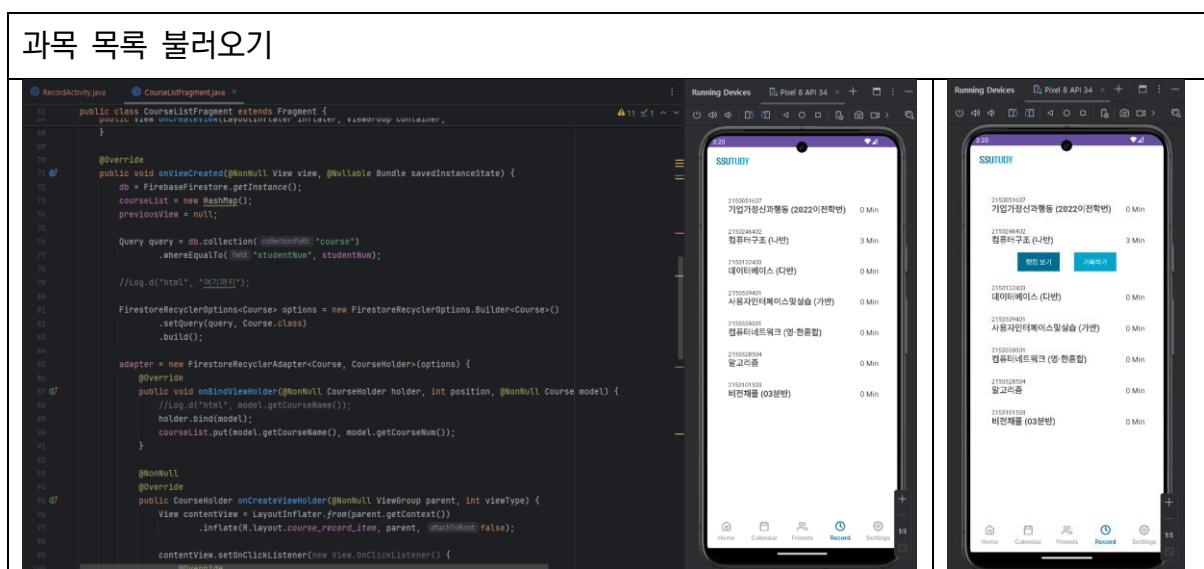
주요메뉴	화면구성/기능	세부설명	담당자	○	완료	추가
				△	진행중	삭제
설정	xml 생성	설정 화면	장다은	○		24.11.10.
		친구 요청 수락 dialog 화면		○		24.11.17.
		닉네임 변경 dialog 화면		○		24.11.14.
		카테고리 관리 화면		○		24.11.16.
		카테고리 삭제 dialog 화면		○		24.11.11.
		카테고리 추가, 수정 화면		○		24.11.17.
	기능 개발	base layout 적용		△		
		친구 요청 수락				
		이미지 변경, 닉네임 변경				
		카테고리 삭제		△		
		카테고리 추가, 수정				

- 친구 요청 기능, 카테고리 관리 기능 개발을 위해 RecyclerView 복습
- 데이터베이스 관리를 위해 Firebase 학습

4. 13주차 진도보고서

1) 이정우

주요메뉴	화면구성/기능	세부설명	담당자	○	완료	추가
				△	진행중	삭제
로그인	xml 생성	시작화면	이정우	○	24.11.6.	
		로그인화면		○	24.11.6.	
		초기 설정 화면		○	24.11.6.	
	기능 개발	유세인트 연동 로그인		○	24.11.17.	
		별명, 프로필 사진 설정		○	24.11.17.	
	xml 생성	친구 목록 화면		○	24.11.23.	
		친구 추가 dialog 화면		○	24.11.18.	
	기능 개발	친구목록 불러오고 공부시간 순위 매기기		○	24.11.23.	
		친구 추가 요청 전송		○	24.11.24.	
친구	xml 생성	과목 목록 화면		○	24.12.01.	
		과목별 공부시간 순위 화면				
		스탑워치 화면				
	기능 개발	유세인트 강의목록 불러오기		○	24.12.01.	
		(개인) 과목별 공부시간 불러오기		○	24.12.01.	
		(랭킹) 과목별 공부시간 불러오고 순위 매기기				
		스탑워치 기능 구현				



2) 송채원

주요메뉴	화면구성/기능	세부설명	담당자	진행도	완료일자	추가 삭제
				○ 완료 △ 진행중		
홈	xml 생성	홈화면	송채원	○	24.11.15.	
		하단 navigation bar		○	24.11.08.	
		수업 내용대로 navigation bar 수정하기		○	24.11.20	12주차때 추가
		base layout 만들기		○	24.11.20	12주차때 추가
	기능 개발	DB에서 종합 데이터 가져오기				
캘린더	xml 생성	캘린더 및 to-do list 화면		○	24.11.14.	
		일정추가 dialog 화면		○	24.11.22	
		강의수정 dialog 화면		○	24.11.24	
		일정수정 dialog 화면		○	24.11.24	
		벼락치기 마법사 dialog 화면				
	기능 개발	DB 데이터를 캘린더, to-do list에 표시				
		강의(일정) 수정,추가				
		벼락치기 마법사				

- 기말고사 대비로 인해 개발 진행 중지
- 대신 Firebase 학습 및 캘린더 화면 재설계

3) 장다은

주요메뉴	화면구성/기능	세부설명	담당자	진행도	완료일자	추가 삭제
				○ 완료 △ 진행중		
설정	xml 생성	설정 화면	장다은	○	24.11.10.	
		친구 요청 수락 dialog 화면		○	24.11.17.	
		닉네임 변경 dialog 화면		○	24.11.14.	
		카테고리 관리 화면		○	24.11.16.	
		카테고리 삭제 dialog 화면		○	24.11.11.	
		카테고리 추가, 수정 화면		○	24.11.17.	
		base layout 적용		△		12주차때 추가
	기능 개발	친구 요청 수락				
		이미지 변경, 닉네임 변경				
		카테고리 삭제		△		

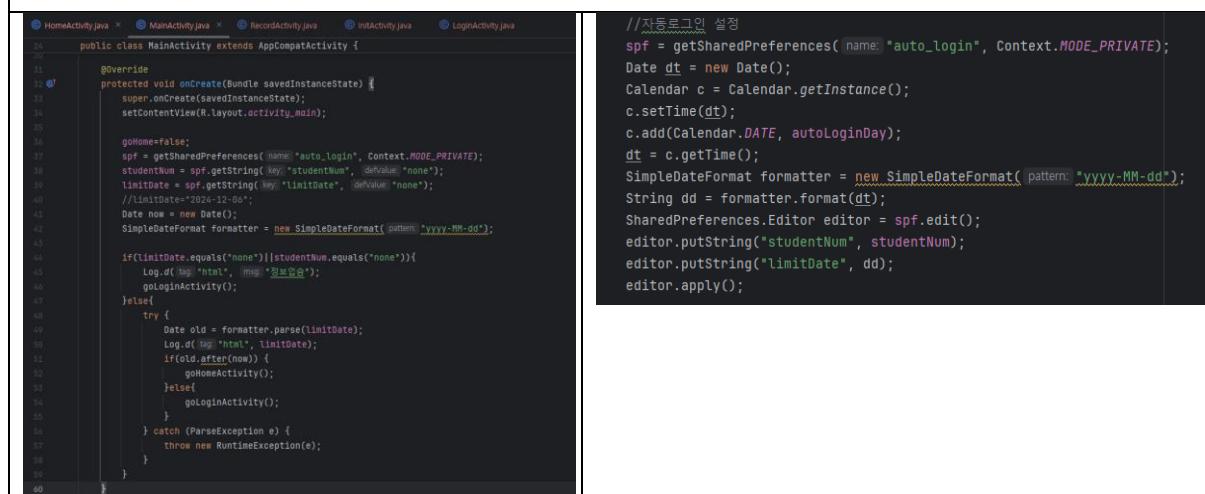
- 기말고사 대비로 개발 진행 중지
- 대신 실습 내용 복습

5. 14주차 진도보고서

1) 이정우

주요메뉴	화면구성/기능	세부설명	담당자	<input type="radio"/> 완료	추가
				<input type="triangle-down"/> 진행중	삭제
로그인	xml 생성	시작화면	이정우	<input type="radio"/>	24.11.6.
		로그인화면		<input type="radio"/>	24.11.6.
		초기 설정 화면		<input type="radio"/>	24.11.6.
	기능 개발	유세인트 연동 로그인		<input type="radio"/>	24.11.17.
		별명, 프로필 사진 설정		<input type="radio"/>	24.11.17.
		자동로그인		<input type="radio"/>	24.12.6.
친구	xml 생성	친구 목록 화면	이정우	<input type="radio"/>	24.11.23.
		친구 추가 dialog 화면		<input type="radio"/>	24.11.18.
	기능 개발	친구목록 불러오고 공부시간 순위 매기기		<input type="radio"/>	24.11.23.
		친구 추가 요청 전송		<input type="radio"/>	24.11.24.
기록	xml 생성	과목 목록 화면	이정우	<input type="radio"/>	24.12.01.
		과목별 공부시간 순위 화면		<input type="radio"/>	24.12.02.
		스탑워치 화면		<input type="radio"/>	24.12.03.
	기능 개발	유세인트 강의목록 불러오기		<input type="radio"/>	24.12.01.
		(개인)과목별 공부시간 불러오기		<input type="radio"/>	24.12.01.
		(랭킹)과목별 공부시간 불러오고 순위 매기기		<input type="radio"/>	24.12.02.
		스탑워치 기능 구현		<input type="radio"/>	24.12.07.
					14주차 추가

자동로그인 기능 개발



```

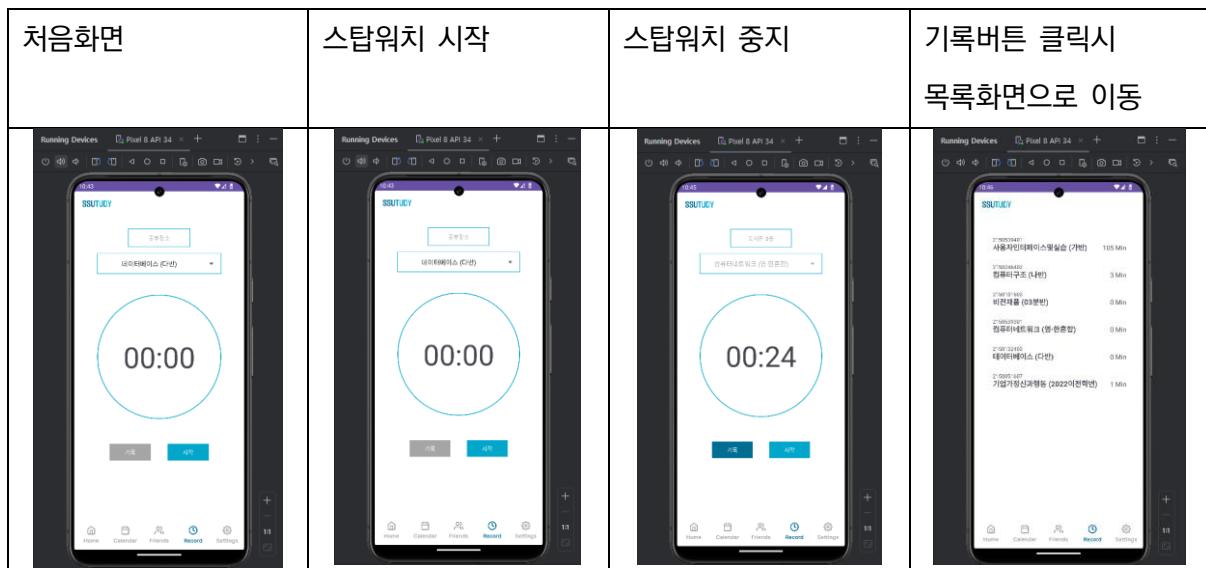
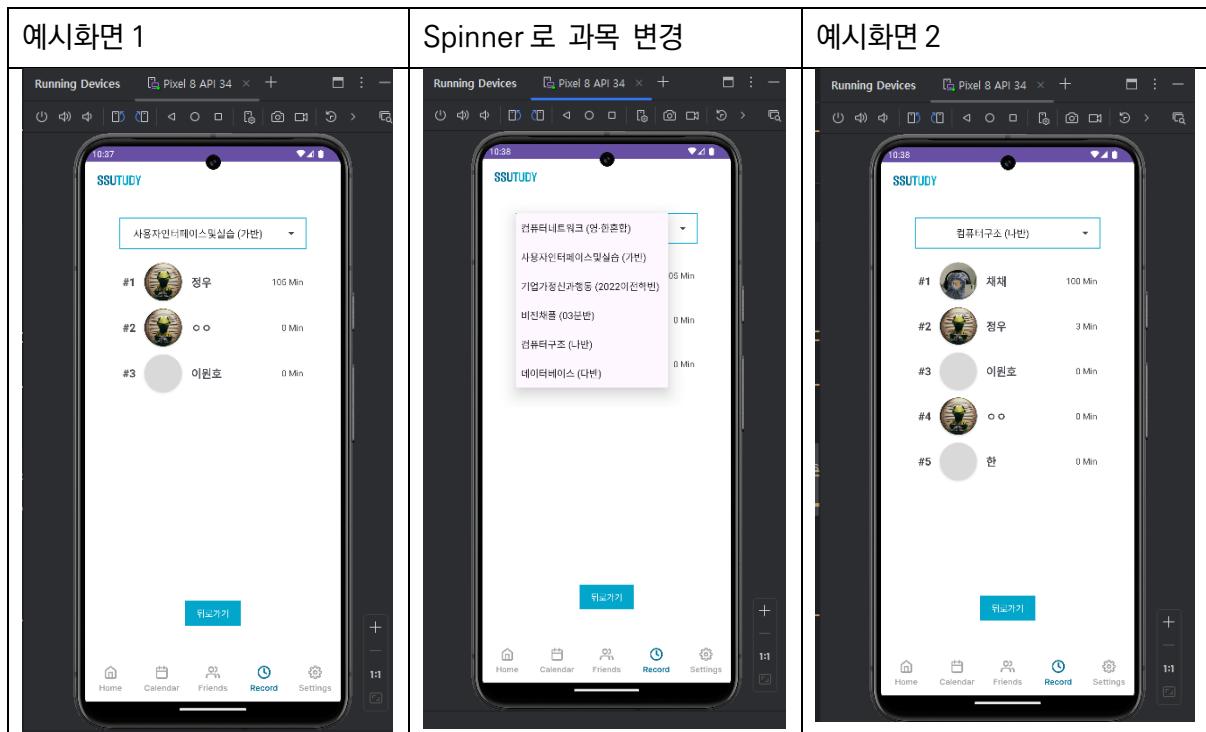
//자동로그인 설정
spf = getSharedPreferences("name", Context.MODE_PRIVATE);
Date dt = new Date();
Calendar c = Calendar.getInstance();
c.setTime(dt);
c.add(Calendar.DATE, autoLoginDay);
dt = c.getTime();
SimpleDateFormat formatter = new SimpleDateFormat(pattern: "yyyy-MM-dd");
String dd = formatter.format(dt);
SharedPreferences.Editor editor = spf.edit();
editor.putString("studentNum", studentNum);
editor.putString("limitDate", dd);
editor.apply();

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        goHome=False;
        spf = getSharedPreferences("name", Context.MODE_PRIVATE);
        studentNum = spf.getString(key: "studentNum", defValue: "none");
        limitDate = spf.getString(key: "limitDate", defValue: "none");
        //limitDate="2024-12-08";
        Date now = new Date();
        SimpleDateFormat formatter = new SimpleDateFormat(pattern: "yyyy-MM-dd");

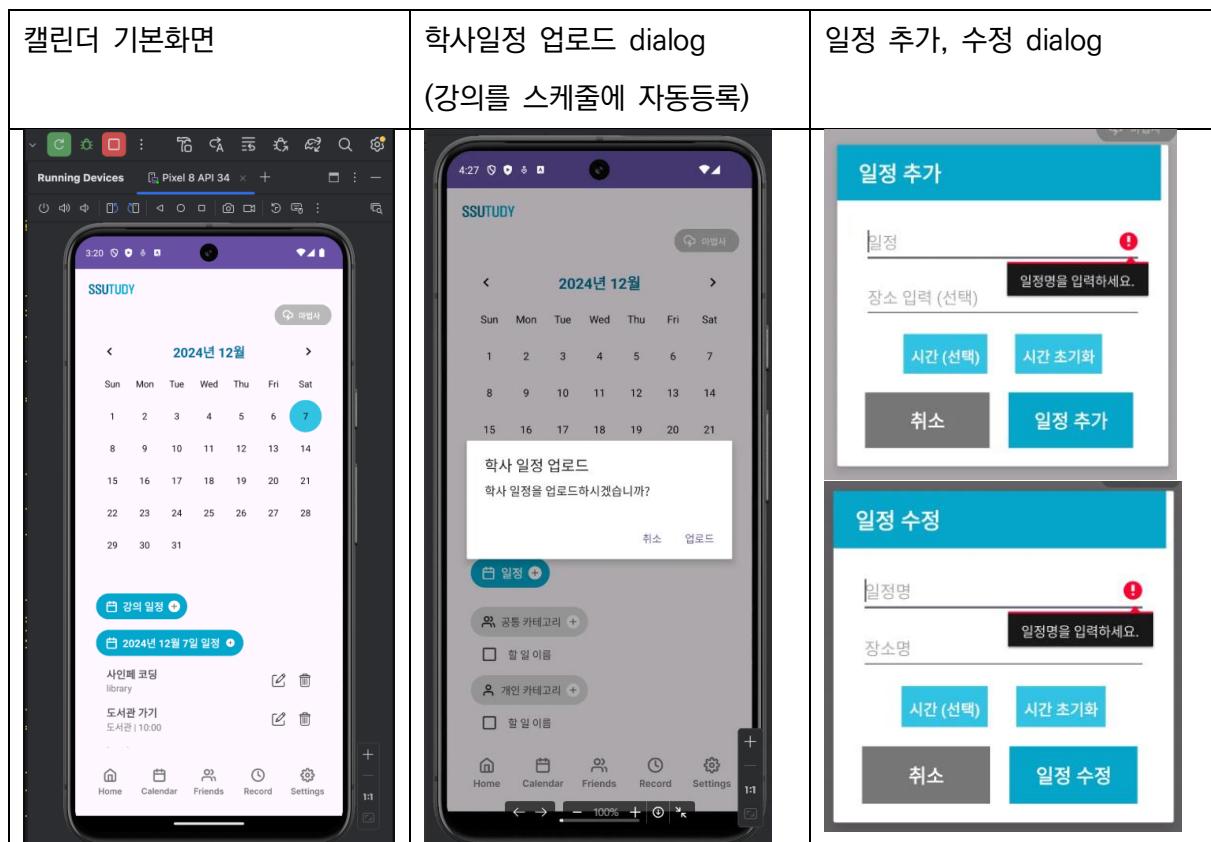
        if(limitDate.equals("none")||studentNum.equals("none")){
            Log.d(tag: "html", msg: "정보없음");
            goLoginActivity();
        }else{
            goHomeActivity();
        }
    } catch (ParseException e) {
        throw new RuntimeException(e);
    }
}

```

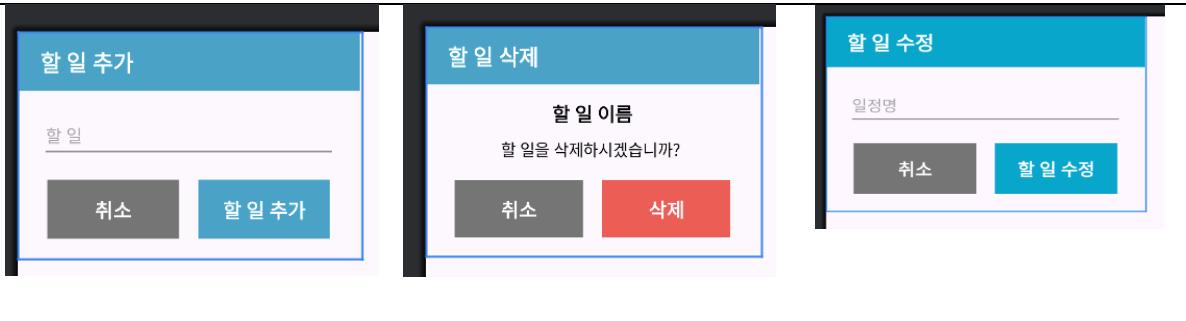


2) 송채원

주요메뉴	화면구성/기능	세부설명	담당자	O	완료	추가
				△	진행중	삭제
홈	xml 생성	홈화면	송채원	O	24.11.15.	
		하단 navigation bar		O	24.11.08.	
		수업 내용대로 navigation bar 수정하기		O	24.11.20	12주차 추가
		base layout 만들기		O	24.11.20	12주차 추가
	기능 개발	DB에서 종합 데이터 가져오기				
캘린더	xml 생성	캘린더 및 to-do list 화면		O	24.11.14.	
		일정추가 dialog 화면		O	24.11.22	
		강의수정 dialog 화면		O	24.11.24	
		일정수정 dialog 화면		O	24.11.24	
		할 일 추가/삭제 dialog 화면		O	24.12.5	13주차 추가
		벼락치기 마법사 dialog 화면		O	24.12.7	캘린더 아래에 세부 일정을 보여주는 것으로 변경
	기능 개발	DB 데이터를 캘린더에 표시		△	14주차 진행중	
		DB 데이터를 to do list에 표시		O	24.12.7	14주차 분해 및 구체화
		2024-2 강의 일정 자동 추가, 수정				
		개인일정 수정, 추가, 삭제 기능				
		벼락치기 마법사				



할 일 추가/삭제/수정 dialog



3) 장다은

주요메뉴	화면구성/기능	세부설명	담당자	진행도	○	완료	추가
					△	진행중	삭제
설정	xml 생성	설정 화면	장다은	○	24.11.10.		
		친구 요청 수락 dialog 화면		○	24.11.17.		
		닉네임 변경 dialog 화면		○	24.11.14.		
		카테고리 관리 화면		○	24.11.16.		
		카테고리 삭제 dialog 화면		○	24.11.11.		
		카테고리 추가, 수정 화면		○	24.11.17.		
		base layout 적용		△		12주차 추가	
	기능 개발	친구 요청 수락		○	24.12.8.		
		유저 정보 불러오기		○	24.12.6.		
		이미지 변경, 닉네임 변경		○	24.12.5.		
		카테고리 목록		○	24.12.8.		
		카테고리 삭제		○	24.12.8.		
		카테고리 추가, 수정 화면		△			
		로그아웃				14주차 추가	

프로필 변경	적용 화면	닉네임 변경	적용 화면

친구 요청 수락

SSUTUDY



다Eng
20232372

친구 요청

카테고리 관리

친구 요청	
박상혁 (20211417)	✓ X
최승환 (20211449)	✓ X

Home Calendar Friends Record Settings

SSUTUDY



정우
20212985

친구 요청

카테고리 관리

친구 요청	
송체원 (20232217)	✓ X
장다은 (20232372)	✓ X

Home Calendar Friends Record Settings

카테고리 삭제

SSUTUDY

카테고리 관리

- 🔒 DELETE TEST ✎ ✖
- 🔒 CUSTOM CATEGORY ✎ ✖

카테고리 삭제

정말 삭제하시겠습니까?

취소
삭제

(+) +

Home Calendar Friends Record Settings

카테고리 수정/생성 (미완)

SSUTUDY

카테고리 관리

- 🔒 CUSTOM CATEGORY ✎ ✖
- 👥 GROUP CATEGORY ✎ ✖

(+) +

Home Calendar Friends Record Settings

카테고리 이름 _____

그룹원 추가 (선택)

-  박상혁 (20211417)
-  이한희 (20211303)
-  최승환 (20211449)
-  이원호 (20212905)
-  이정우 (20212985)

취소
저장

Home Calendar Friends Record Settings

III. 기능별 부연설명

1. 이정우

1) 유세인트 연동 로그인

유세인트를 통한 숭실대 소속 인증과 해당 학기 수강 강의 목록을 스크래핑하는 기능이다.

해당 기능을 구현하기 위해서는 유세인트 파싱이 필수적인데, 이 부분 개발이 굉장히 어려웠다. 유세인트는 동적 웹페이지인데, 안드로이드에서는 보통 WebView를 이용한 정적 웹페이지 파싱이 대부분이고, 동적 웹페이지를 안드로이드 기기에서 직접 파싱하는 경우는 거의 없기 때문이다.

동적 웹페이지를 파싱할 때에는 셀리니움을 많이 사용하는데, 셀레니움이 안드로이드 OS 환경에서 실행되지 않기 때문에 보통은 안드로이드 기기에서 서버에 파싱을 요청하면 서버가 셀레니움을 이용해 파싱을 진행하고, 그 결과를 다시 기기에 돌려주는 방식으로 파싱 기능을 구현한다.

유세인트가 동적 웹페이지이니 셀레니움과 서버를 이용해 파싱하는 게 가장 무난한 방법이라 생각은 했지만, 서버 없이 안드로이드 기기만으로 파싱 기능을 구현하고 싶어서 WebView를 이용해 여러 시도를 하다 많은 시행착오 끝에 구현할 수 있었다.

```
binding.webview.setWebViewClient(new WebViewClient(){
    @Override 3 usages
    public void onPageFinished(WebView view, String url) {
        super.onPageFinished(view, url);
        if(dan==0) {
            binding.login.setText("재로그인");
            binding.login.setEnabled(true);
            view.loadUrl("javascript:document.getElementById('userid').value='" + studentNum +
                         "';document.getElementById('pwd').value='" + pw + "';" +
                         "document.getElementsByClassName('btn_login')[0].click();");
        }else if(dan==1){
            binding.login.setText("로그인");
            binding.login.setEnabled(false);
            Log.d( tag: "html", msg: "dan1");
            view.loadUrl("https://saint.ssu.ac.kr/irj/portal");
        }else if(dan==2){
            Log.d( tag: "html", msg: "dan2");
            view.loadUrl("javascript:document.getElementsByClassName('btn_login')[0].click();");
        } else if (dan == 3) {
            Log.d( tag: "html", msg: "dan3");
            view.loadUrl("https://saint.ssu.ac.kr/irj/portal");
        }
    }
})
```

onPageFinished 함수가 실행될 때마다 dan 변수의 값을 증가시켜 매번 다른 명령어

를 페이지에 전달하여 실행하는 방식으로 자동 로그인을 구현하였다.

```
    } else if(dan==4) {
        Log.d( tag: "html", msg: "dan4");
        view.loadUrl("javascript:window.Android.getHtml(document.getElementsByName('html')[0].innerHTML);");
        view.loadUrl("https://ecc.ssu.ac.kr:8443/sap/bc/webdynpro/SAP/ZCMW2110#");
    }
```

유세인트 페이지에서 학생 이름은 html 파일 내용에 포함되어 있기 때문에 innerHTML을 전송 받은 후 해당 문자열에서 이름이 들어있는 태그를 찾아서 이름을 알아낼 수 있었다.

```
else if(dan==5){
    Log.d( tag: "html", msg: "dan5");
    view.evaluateJavascript( script: "var originalXhrOpen = XMLHttpRequest.prototype.open;" +
        "XMLHttpRequest.prototype.open = function(method, url) {" +
        "    this.addEventListener('load', function() {" +
        "        Android.receiveAjaxResponse(this.responseText);" +
        "    });" +
        "    originalXhrOpen.apply(this, arguments);" +
        "};", resultCallback: null);
}
dan++;
```

가장 어려웠던 부분이 바로 강의 목록을 스크래핑하는 부분이었다. 일단 수강신청내역 조회 페이지에 접근해야 하는데, 해당 페이지가 동적 웹페이지로 개발되어 있어서 메뉴 버튼에서 선택해야만 해당 페이지로 접속할 수 있었다. WebView에서는 click() 함수를 실행해도 페이지 이동 및 전환이 불가능했기에 바로 해당 페이지로 들어갈 수 있는 url 을 찾다가 다음 url을 구하였다.

[https://ecc.ssu.ac.kr:8443/sap/bc/webdynpro/SAP/ZCMW2110:](https://ecc.ssu.ac.kr:8443/sap/bc/webdynpro/SAP/ZCMW2110;)

위 url로 이동하면 바로 수강신청내역조회 페이지로 접속하게 된다.

학생 이름을 가져올 때처럼 innerHTML 텍스트를 전송 받아 강의 목록을 스크래핑하려 했지만 전송된 innerHTML 텍스트는 빈 껍데기였고, 강의 목록 정보는 ajax를 통해 화면에 뿌려진다는 것을 알게 되었다. 웹페이지에 전송되는 ajax 응답을 안드로이드 기기로 가져오기 위해 위 사진과 같은 함수를 웹페이지에서 실행되게 만들어서 ajax 요청이 완료되어 페이지로 응답이 전송되면, 해당 ajax 응답을 안드로이드 기기에도 전송되게 하였다. 이렇게 받아온 ajax 응답에서 아래 정규표현식을 활용해 강의 정보들을 찾아내었다.

```

String regexDigits = ">(\d{10})<";
String regexCourse = "lsTextview--wrap\">(.*)</span></span>";
String regexTime1 = "[([가-힣])&#x20;([0-9]+)&#x3a;([0-9]+)-([0-9]+)&#x20;&#x28;(.*)&#x29;";
String regexTime2 = "[([가-힣])&#x20;([가-힣])&#x20;([0-9]+)&#x3a;([0-9]+)-([0-9]+)&#x20;&#x28;(.*)&#x29;";
String regexTrigger = "본인&#x20;신청";
String regexTrigger2 = "관리자&#x20;지정";

```

찾아낸 강의 정보들은 Map 자료형으로 둑어서 DB에 보내 저장하였다.

2) 친구 목록 불러오기

친구 목록을 불러오기 위해서는 1. 현재 로그인 된 사용자의 친구들 학번을 가져오고 2. 해당 학번들의 user 정보를 가져와야 한다. DB 쿼리를 순차적으로 실행시켜야 이상 없이 작동하는데, 문제는 DB 쿼리가 비동기적이라 순차적으로 작동하게 하려면 쿼리문 안에 또 다른 쿼리문을 작성하여 실행하는 방식밖에 없다고 생각하였다. 데이터를 가져오기만 하는 거였다면 해당 방식으로 했겠지만, RecyclerView와 연동하는 문제 때문에 중첩하여 실행하는 것이 불가능하였다. 그래서 고민하다가 FriendsActivity에서 두 쿼리를 모두 실행하는 것이 아니라 외부에서 하나, FriendsActivity에서 하나 실행하는 방식으로 구현하였다. NavigationView에서 Friends 탭을 클릭하면 친구 학번들을 미리 DB에서 가져온 다음, 그 리스트를 intent에 담아 FriendsActivity로 보내면 FriendsActivity에서 해당 리스트를 이용해 user 정보를 가져와 RecyclerView에 뿐린다.

```

if(itemId == R.id.page_3){
    db.collection("friends").whereEqualTo("field", "rootuser", studentNum).get()
        .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {
                if(task.isSuccessful()){
                    for(QueryDocumentSnapshot document : task.getResult()){
                        //Log.d("html", document.get("childuser").toString());
                        friendList.add(document.get("childuser").toString());
                    }
                }

                Intent intent = new Intent(packageContext, FriendsActivity.class);
                intent.putExtra("studentNum", studentNum);
                intent.putExtra("friendList", friendList);
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                startActivity(intent);
                overridePendingTransition(enterAnim: 0, exitAnim: 0);
            }
        });
    return true;
}

```

NavigationView의 friends 탭을 클릭하면 DB에서 학번들을 불러와 friendList에 저장한다.

```

if(friendList.size() != 0) {
    Query query = db.collection( collectionPath: "users" ) CollectionReference
        .whereIn( field: "studentNum", friendList ) Query
        .orderBy( field: "totalStudyTime", Query.Direction.DESCENDING);
}

```

FriendsActivity에서 friendList를 이용하여 쿼리문을 작성한다.

3) 스탑워치 기능

스탑워치가 백그라운드에서 돌아갈 수 있도록 서비스를 이용해 구현하였다.

또한 스탑워치를 실행시킨 뒤 다른 액티비티로 넘어갔다가 다시 돌아와도 기존 스탑워치 화면이 유지되게끔 하였다.

```

private void toggleStopwatch(){ 1 usage
    Intent intent = new Intent(requireContext(), StopwatchService.class);
    if(first){
        first=false;
        //원복하기 위한 데이터 저장
        SharedPreferences.Editor editor = spf.edit();
        editor.putBoolean("isStudying", true);
        editor.putString("courseName", courseName);
        JSONObject jo = new JSONObject(courseList);
        String jsonStr = jo.toString();
        editor.putString("jsonStr", jsonStr);
        editor.apply();
    }
}

```

스탑워치를 시작하면 스탑워치가 작동 중이라는 것과 과목 정보를 Sharedpreferences를 이용해 저장한다.

```

boolean isStudying = spf.getBoolean( key: "isStudying", defaultValue: false);

if(isStudying){
    //원래 화면 복원
    String courseName= spf.getString( key: "courseName", defaultValue: "0");
    HashMap<String, String> courseList = new HashMap<>();
    String jsonStr = spf.getString( key: "jsonStr", (new JSONObject()).toString());
    try {
        JSONObject jo = new JSONObject(jsonStr);
        Iterator<String> keys = jo.keys();
        while(keys.hasNext()){
            String key = keys.next();
            String value = (String)jo.get(key);
            courseList.put(key, value);
        }
    } catch (JSONException e) {
        throw new RuntimeException(e);
    }
    getSupportFragmentManager().beginTransaction()
        .add(R.id.fragment_container, CourseWatchFragment.newInstance(courseName, courseList, studentNum, true))
        .commit();

} else{
    getSupportFragmentManager().beginTransaction()
        .add(R.id.fragment_container, CourseListFragment.newInstance(studentNum))
        .commit();
}

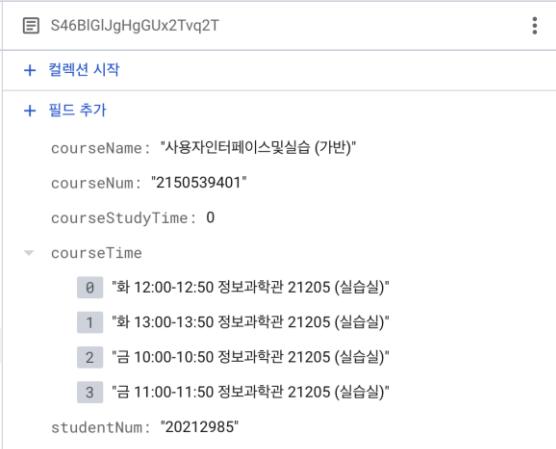
```

스탑워치가 실행 중이면 기존 정보를 복원하며 스탑워치 fragment를 실행시키고, 실행 중이 아니라면 기본 화면인 과목 리스트 fragment를 실행시킨다.

2. 송채원

1) 자동 강의 일정 추가 (Calendar Activity)

스크래핑 된 데이터를 바탕으로 2학기 기간(9/2~12/13)과 요일에 맞추어 강의 일정을 자동으로 추가하는 기능이다. 왼쪽 하단이 스크래핑 된 데이터이며, 이 데이터의 studentNum과 현재 userID를 비교하여 사용자가 수강하는 수업만 불러온다. 이 데이터는 parseAndUploadSchedule()이라는 메소드를 거쳐 데이터베이스에 users/학번/courseSchedules/날짜/tasks 아래에 한 과목 당 하나의 문서로 저장된다. 스크래핑 데이터에서 강의명, 강의 요일, 강의 시간, 건물과 강의실 호수까지 불러와 문서로 저장한다.

	
--	---

스크래핑된 데이터

파싱하여 저장한 모습

 2024-09-06	 2024-09-06
+ 문서 추가 2024-09-04 2024-09-06 > 2024-09-11 2024-09-13 2024-09-18 2024-09-20 2024-09-25 2024-09-27 2024-10-02 2024-10-04 2024-10-09 2024-10-11 2024-10-16 2024-10-18	+ 컬렉션 시작 tasks + 필드 추가 존재하지 않는 문서이며 권리나 스냅샷에 표시되지 않습니다. 자세히 알아보기

저장된 전체 모습

이러한 로직을 구현하고 나서, 어떻게 2학기 일정을 추가하도록 할지 고민을 했다. 이

작업은 우리 앱에서 중요한 초기화 작업이며, 반복적으로 수행되어서는 안됐다. 여러 고민을 해 보았는데, 추가하기는 쉬운 버튼은 사용자가 누르지 않으면 추가가 되지 않고, 별다른 설명이 없다면 사용자가 눌러야 할 당위성을 느끼지 못할 것 같았다. 또한 앱 실행 시 이미 강의 일정이 추가 되었는지 아닌지를 확인하고, 추가하지 않았다면 사용자가 접근할 때마다 다이얼로그를 띄워 추가하도록 자연스럽게 이끌고 싶었다. 따라서 수업시간에 배운 SharedPreferences를 사용하여 일정 추가 상태를 로컬에 저장하고 이를 !uploadComplete인 경우 dialog를 보이게 했다. 이러한 방법이 데이터 중복을 막는데 도움이 될 뿐만 아니라, 사용자 편의성을 높였을 것이라고 생각한다.

```

SharedPreferences sharedPreferences = getSharedPreferences(PREFS_NAME, MODE_PRIVATE);
boolean uploadCompleted = sharedPreferences.getBoolean(KEY_UPLOAD_COMPLETED, b: false);
private void showUploadDialog(SharedPreferences sharedPreferences) { 1 usage
    AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
    builder.setTitle("학사 일정 자동 업로드")
        .setMessage("학사 일정을 캘린더에 업로드하시겠습니까?")
        .setPositiveButton( text: "업로드", (dialog, which) -> {
            Log.d( tag: "CalendarActivity", msg: "Upload initiated.");
            db.collection( collectionPath: "course") CollectionReference
                .get() Task<QuerySnapshot>
                    .addOnSuccessListener(queryDocumentSnapshots -> {
                        if (queryDocumentSnapshots.isEmpty()) {
                            return;
                        }
                    })
        })
        .setNegativeButton( text: "취소", (dialog, which) -> {
            Log.d( tag: "CalendarActivity", msg: "학사 일정 업로드 취소");
            dialog.dismiss();
        });
    AlertDialog dialog = builder.create();
    dialog.show();
}
(... 예외처리로직 ...)

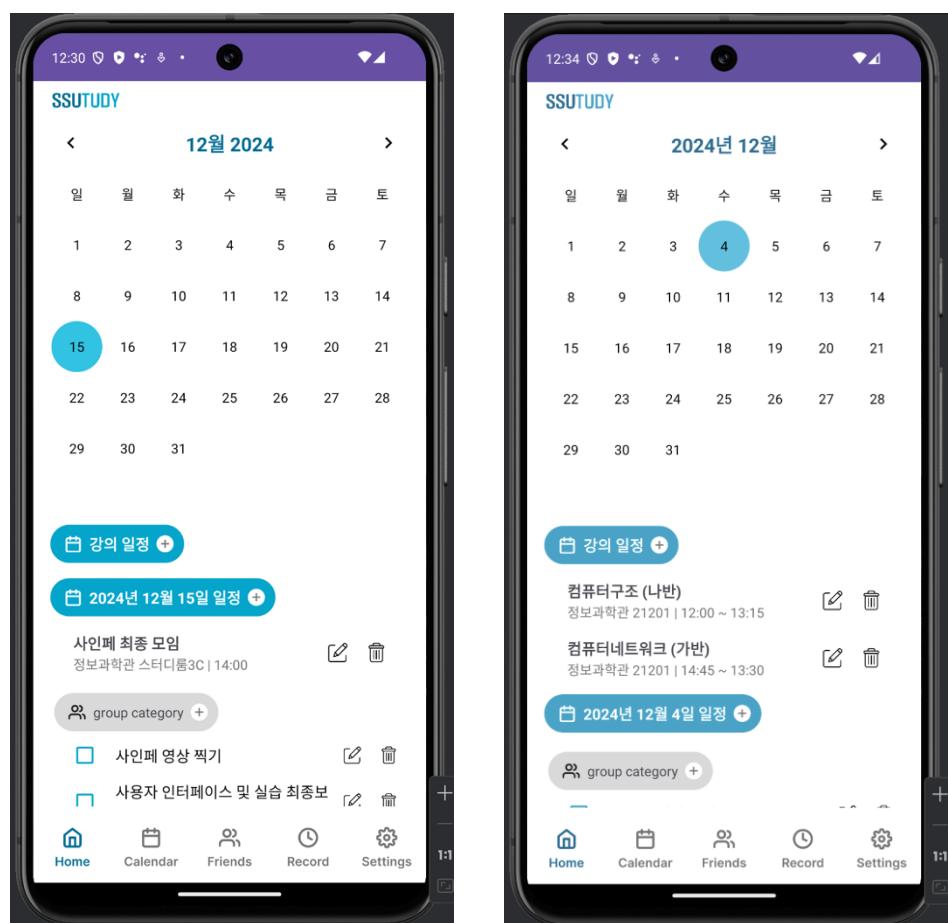
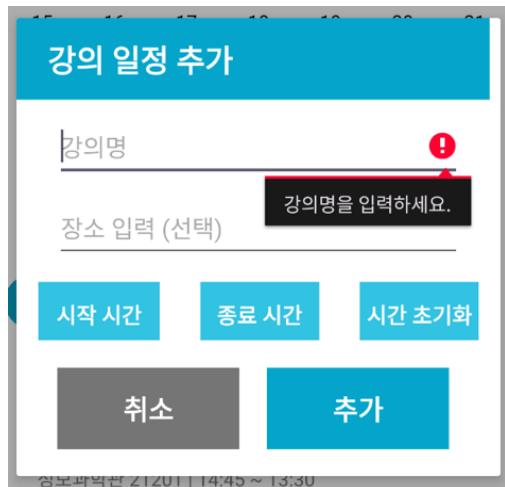
```



2) 캘린더/강의 일정/사용자 일정 관리 기능 (Calendar Activity)

강의 일정은 classSchedule을, 사용자 일정은 schedule이라는 명칭을 사용하였다.

강의 일정이나 사용자 일정을 추가할 때 강의명/일정명이 입력되지 않으면 경고 메시지를 띄우며 일정에 추가되지 않도록 설정하였고 추가, 수정, 삭제될 때 모두 데이터베이스와 연동된다. (우측)

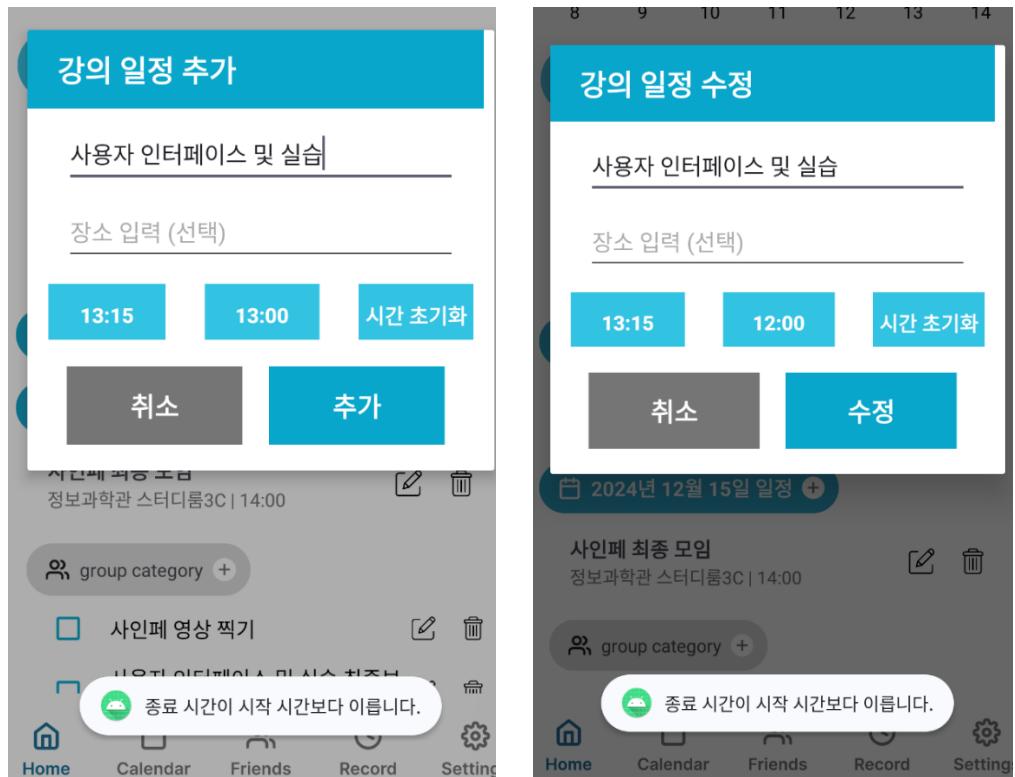


왼쪽은 calendar를 실행하면 바로 보이는 화면. 오늘 날짜에 맞춰 자동으로 화면에 표시되고 일정이 보인다. 우측은 강의 시작 순서에 따라 강의 일정이 정렬되는 모습을 확인할 수 있다. 사용자 일정 또한 동일하게 구현했다.

이 기능을 구현하기 시작할 때 viewBinding이 익숙하지 않아서 findViewById로 시작했는데, 기능까지 추가하다보니 코드가 너무 길어져서 작업이 쉽지 않았다. 또한 강의 일정과 사용자 일정을 보여줄 때 각각 recycler view를 사용하면서 adapter와 model을 사용하였는데 낯설었지만 사용하다보니 감을 익혀 UI를 효율적으로 관리할 수 있었다.

더불어 강의 일정과 사용자 일정의 장소 유무, 시간 유무에 따라 다르게 보여지도록 했다. (우측)

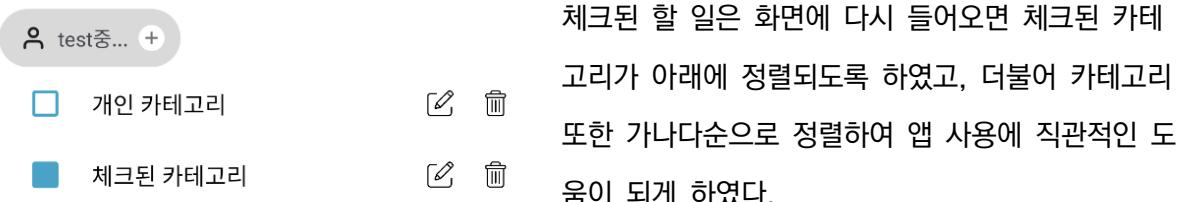
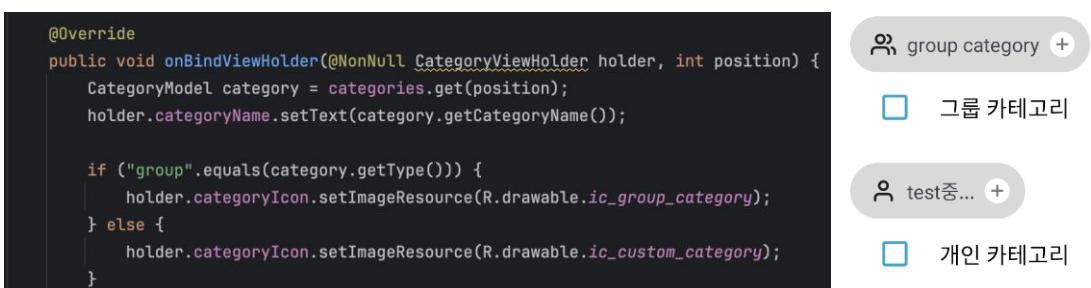
종료시간이 시작시간보다 이른 경우에는 toast를 통해 오류를 알려주었고 강의 일정이 추가/수정되지 않게 하였다.



3) 카테고리 표시 및 할 일 관리 기능 (Calendar Activity)

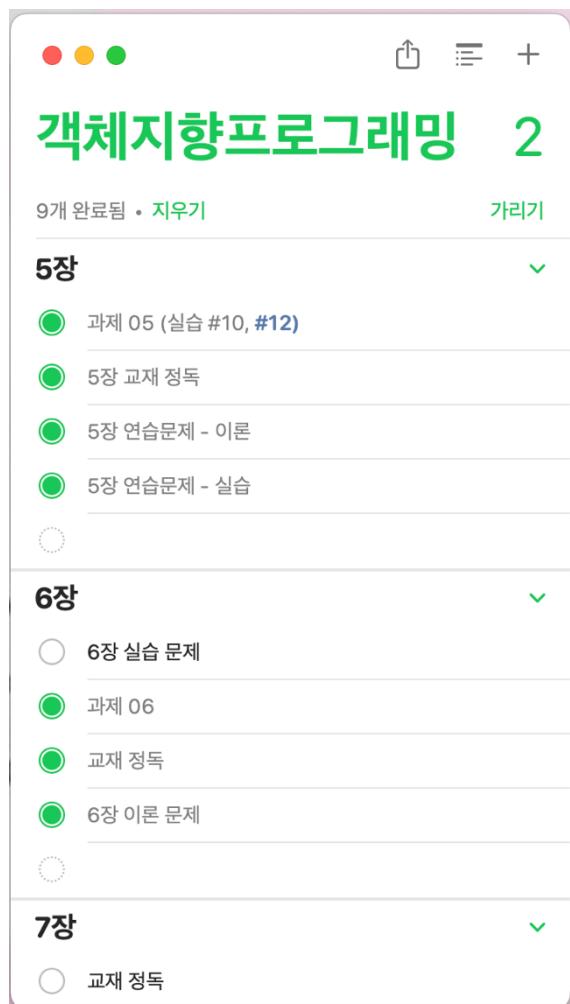
이 부분이 이번 구현에서 가장 어려운 부분이었고 분량에 비해 시간을 가장 많이 소요한 기능이다. 카테고리가 몇 개인지, 타입이 어떤 건지 정해진 것이 아니라, 데이터베이스에서 읽어와야지만 알 수 있었기 때문에 카테고리와 그 안에 속한 할 일 모두 recycler view로 보여줘야 했다. 즉, Nested RecyclerView를 사용했는데 스크롤 충돌도 일어나고, 이유를 알 수 없는 UI 간섭 오류가 일어나며 강제 종료가 되었다. 따라서 각 RecyclerView가 독립적으로 동작하도록 다시 코딩하여 충돌을 방지하였다. 또한 Firestore의 비동기 데이터를 처리할 때, 하위 recyclerView(to do)가 데이터를 완전히 가져올 때까지 상위 recyclerView(카테고리)의 UI 업데이트를 지연시켰다. (onComplete 콜백 사용)

또한 카테고리 타입이 custom인지, group인지에 따라 카테고리 아이콘이 다르게 보여지도록 구현하여 사용자가 쉽게 카테고리 타입을 구분할 수 있도록 했다.



4) 벼락치기 마법사 (Calendar Activity)

일단 할 일을 날짜 단위로 추가할지, 아니면 날짜와는 관계 없이 카테고리 안에 있는 할 일을 모두 보여줄지 고민했는데, 내가 무엇을 했는지 한 눈에 보여주는 어플을 썼을 때 학습에 도움이 되었던 기억을 바탕으로 날짜와는 관계 없이 할 일을 추가할 수 있도록 구현했다. 반면 우리가 초기에 기획한 벼락치기 마법사는 공부할 분량을 입력하면 선택한 날짜에 적당량의 공부 계획을 짜주는 기능이었기 때문에 상충하여 벼락치기 마법사 기능을 제외하였다.



5) Home Activity

자동으로 현재 날짜에 맞추어 모두 불러오도록 구현했다.

Calendar activity에서 사용한 강의 일 정과 일정을 보여주는 adapter와 model을 동일하게 사용하였는데, 이때 delete button과 edit button을 가려 사용하였다.

```
if (showActionButtons) {  
    holder.editButton.setVisibility(View.VISIBLE);  
    holder.deleteButton.setVisibility(View.VISIBLE);  
} else {  
    holder.editButton.setVisibility(View.GONE);  
    holder.deleteButton.setVisibility(View.GONE);  
}  
  
holder.editButton.setOnClickListener(v -> {  
    if (listener != null) {  
        listener.onEditSchedule(schedule);  
    }  
});  
  
holder.deleteButton.setOnClickListener(v -> {  
    if (listener != null) {  
        listener.onDeleteSchedule(schedule);  
    }  
});
```



또한 loadIncompleteTodos() 메소드와 updateRemainingTasks()를 사용하여 데이터베이스에서 할 일을 가져올 때 할 일이 미완료 상태인 경우만 불러와 개수와 상세 내용을 표시하도록 만들었다. 즉, 체크하여 완료했다고 표시하면 남은 할 일 개수와 상세 내용이 갱신되도록 만들었다.,

```
private void updateRemainingTasks(int incompleteCount) { 2 usages  
    String remainingText = "남은 할 일 " + incompleteCount + "개 !";  
    binding.progressText.setText(remainingText);  
}
```

3. 장다은

1) 닉네임 변경

닉네임 변경 후 다이얼로그가 종료되고 원래 화면으로 돌아왔을 때, 변경된 닉네임이 반영되지 않는 문제가 발생했다. 이는 닉네임 변경 작업이 별도의 Activity에서 수행되고, 반환값을 처리하지 않아 발생한 문제였다.

```
161    ActivityResultLauncher<Intent> editNicknameLauncher =
162         registerForActivityResult(new ActivityResultContracts.StartActivityForResult(), result -> {
163             if (result.getResultCode() == RESULT_OK) {
164                 Intent data = result.getData();
165                 String newNickname = data.getStringExtra(name: "newNickname");
166                 binding.userName.setText(newNickname);
167             }
168         });
169
170     binding.menuEditUserName.setOnClickListener(new View.OnClickListener() {
171         @Override
172         public void onClick(View view) {
173             Intent intent=new Intent(packageContext: SettingsActivity.this, EditNicknameDialogActivity.class)
174                 .putExtra(name: "studentNum", studentNum);
175             editNicknameLauncher.launch(intent);
176         }
177     });
178 }
```

이를 해결하기 위해 ActivityResultLauncher를 활용하여 닉네임 변경 결과를 비동기로 처리하도록 구현하였다. 닉네임 수정 화면에서 반환된 결과를 받아 새로운 닉네임 값을 가져오고, 이를 현재 설정 화면에 즉시 반영하도록 하였다.

2) 친구 요청 수락

친구 요청을 수락하거나 삭제하면 아이템이 즉시 목록에서 사라지고, RecyclerView가 업데이트되도록 구현하였다. 다음은 친구 요청 수락 또는 거절 시 호출되는 deleteFriendRequest 함수이다.

```
157     private void deleteFriendRequest(String senderStudentNum, int position) { 2 usages
158         db.collection(collectionPath: "request_friend").CollectionReference
159             .whereEqualTo(field: "receiverStudentNum", studentNum) Query
160             .whereEqualTo(field: "senderStudentNum", senderStudentNum)
161             .get() Task<QuerySnapshot>
162             .addOnSuccessListener(queryDocumentSnapshots -> {
163                 for (DocumentSnapshot doc : queryDocumentSnapshots.getDocuments()) {
164                     doc.getReference().delete();
165                 }
166                 runOnUiThread(() -> {
167                     if (position >= 0 && position < friendRequests.size()){
168                         friendRequests.remove(position);
169                         adapter.notifyItemRemoved(position);
170                     }
171                     loadFriendRequests(friendRequests);
172                 });
173             })
174             .addOnFailureListener(e -> {
175                 e.printStackTrace();
176             });
177         db.collection(collectionPath: "request_friend").CollectionReference
178             .whereEqualTo(field: "receiverStudentNum", senderStudentNum) Query
179             .whereEqualTo(field: "senderStudentNum", studentNum)
180             .get() Task<QuerySnapshot>
181             .addOnSuccessListener(queryDocumentSnapshots -> {
182                 for (DocumentSnapshot doc : queryDocumentSnapshots.getDocuments()) {
183                     doc.getReference().delete();
184                 }
185             });
186     }
```

Firestore에서 데이터를 삭제한 후, adapter.notifyItemRemoved(position) 메서드를 호출하여 해당 아이템이 즉시 RecyclerView에서 사라지도록 하였다. 또한 데이터를 삭제한 후, 화면에 업데이트된 데이터가 반영되도록 loadFriendRequests 함수를 다시 호출하였다.

기능을 테스트하던 중 사용자가 서로 친구 요청을 보낸 경우, 데이터 중복으로 인한 문제가 발생할 수도 있다는 사실을 알게 되었다. 이에 대비하여 친구 요청을 수락 또는 거절한 경우 rootuser와 childuser 모두에서 관련 document를 삭제하고자 사용자 정보를 검색하였고, 양방향의 요청 데이터를 모두 삭제하기 위해 두 번의 쿼리를 수행하도록 하였다.

또한, 친구 요청 아이템이 2개 남은 상태에서 클릭하지 않은 아이템이 의도치 않게 사라지는 문제가 발생하여 이를 해결하는 데 많은 시간을 투자했다. DB에는 요청이 정상적으로 저장되었으나 화면 상에서는 아이템이 사라지지 않았고, 여러 번 클릭하는 경우 중복된 데이터가 여러 번 저장되는 문제도 발생했다.

술한 고민 끝에 RecyclerView의 데이터가 비동기로 갱신되는 과정에서 UI 스레드와 충돌하여 발생하는 문제라고 판단하였다. 이를 해결하기 위해 아이템을 삭제하고 데이터를 갱신하는 작업을 runOnUiThread 내에서 실행되도록 하였고, 아이템이 비정상적으로 삭제되는 문제를 해결할 수 있었다.

3) 카테고리 목록

카테고리를 추가, 삭제, 수정하고 돌아왔을 때, RecyclerView에 업데이트된 데이터가 즉시 반영되도록 구현하였다.

```

29     private final ActivityResultLauncher<Intent> updateCategoryLauncher = registerForActivityResult(new ActivityResultContracts.StartActivityForResult(), result -> {
30         if (result.getResultCode() == RESULT_OK){
31             loadCategories();
32         }
33     });
34 );
35
36     @Override
37     protected void onCreate(Bundle savedInstanceState) {
38         super.onCreate(savedInstanceState);
39         ActivityCategoryBinding binding = ActivityCategoryBinding.inflate(getLayoutInflater());
40         setContentView(binding.getRoot());
41
42         studentNum = getIntent().getStringExtra("studentNum");
43         db = FirebaseFirestore.getInstance();
44         categoryList = new ArrayList<>();
45         adapter = new CategoryAdapter(categoryList);
46
47         binding.categoryRecyclerView.setLayoutManager(new LinearLayoutManager(context: this));
48         binding.categoryRecyclerView.setAdapter(adapter);
49
50         loadCategories();
51
52         binding.addCategoryButton.setOnClickListener(new View.OnClickListener() {
53             @Override
54             public void onClick(View view) {
55                 Intent intent = new Intent(packageContext: CategoryActivity.this, EditCategoryActivity.class)
56                     .putExtra("studentNum", studentNum)
57                     .putExtra("isEdit", false);
58                 updateCategoryLauncher.launch(intent);
59             }
60         });
61     }

```

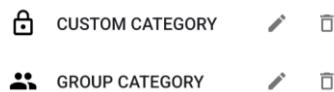
처음에는 `ActivityResultLauncher`를 활용하려 하였으나, 의도한 대로 작동하지 않았다. 카테고리 수정 또는 추가 후 원래 화면으로 돌아왔을 때, 업데이트 된 데이터가 화면에 반영되지 않았다.

```

62     protected void onResume(){
63         super.onResume();
64         loadCategories();
65     }

```

문제를 해결하기 위해 여러 가지 방법을 고민하던 중, 혹시나 하는 마음에 `onResume()`을 추가해 보았다. 그 결과 의도한 대로 작동했다. 구현할 것들이 남아있어 일단 넘어갔지만, 사실 왜 되는지 아직도 잘 모르겠다. 추가적으로 공부가 필요할 것 같다.



(최종: custom category 아이콘 다른 그림으로 변경)

카테고리의 타입은 그룹원이 함께 공유하는 group 카테고리와 사용자 지정 custom 카테고리 두 가지로 구분된다. 사용자가 지정한 타입에 따라 아이콘이 변경되도록 구현하였다.

4) 카테고리 추가, 수정

카테고리 추가와 수정 기능을 구분하기 위해 `isEdit` 플래그를 사용했다.

카테고리 수정 모드에서는 `loadCategoryDetails()` 메서드를 통해 기존 카테고리 정보를 Firestore에서 가져와 화면에 표시한다. 수정하는 카테고리가 group 타입일 경우 그룹 멤버의 목록을 가져와 체크박스로 표시하고, 사용자가 이를 수정할 수 있도록 한다.

```
77     private void loadCategoryDetails() { 1 usage
78         db.collection( collectionPath: "category" ).document( documentId ) DocumentReference
79             .get() Task<DocumentSnapshot>
80             .addOnSuccessListener( doc -> {
81                 if ( doc.exists() ) {
82                     categoryNameInput.setText( doc.getString( field: "categoryName" ) );
83                     studentNum=doc.getString( field: "rootuser" );
84
85                     String type=doc.getString( field: "type" );
86                     List<String> members = (List<String>) doc.get("member");
87                     loadFriends(() ->{
88                         if ("group".equals(type) && members != null){
89                             for (FriendItem item : friendList){
90                                 if (members.contains(item.studentNum)){
91                                     item.isChecked = true;
92                                 }
93                             }
94                         adapter.notifyDataSetChanged();
95                     });
96                 }
97             }
98         ).addOnFailureListener( e -> {
99             e.printStackTrace();
100        });
101    }
102 }
```

카테고리를 저장할 때에는 선택된 멤버가 존재하는지 여부에 따라 타입을 결정한다. 카테고리 이름과 선택된 멤버들을 Map에 담고, group 타입일 경우 멤버 목록을 포함하여 저장한다.

```
139     private void saveCategory() { 1 usage
140         String categoryName = categoryNameInput.getText().toString().trim();
141         if (categoryName.isEmpty()){
142             return;
143         }
144         List<String> selectedMembers = new ArrayList<>();
145         for (FriendItem item:friendList) {
146             if (item.isChecked) {
147                 selectedMembers.add(item.studentNum);
148             }
149         }
150         String type = selectedMembers.isEmpty() ? "custom" : "group";
151         Map<String, Object> categoryData = new HashMap<>();
152         categoryData.put( k: "categoryName", categoryName );
153         categoryData.put( k: "rootuser", studentNum );
154         categoryData.put( k: "type", type );
155         if ("group".equals(type)) {
156             categoryData.put( k: "member", selectedMembers );
157         }
158     }
```

카테고리 수정 시, 이미 해당 카테고리의 document가 존재하므로, `documentId`를 사용해 `set()` 메서드로 덮어쓴다. 새로 생성하는 카테고리인 경우, `add()` 메서드를 사용하여 새로운 document를 추가한다.

```

158     if (isEdit){
159         db.collection( collectionPath: "category" ).document(documentId) DocumentReference
160             .set(categoryData) Task<Void>
161             .addOnSuccessListener(aVoid -> finish())
162             .addOnFailureListener(e -> {
163                 e.printStackTrace();
164             });
165     } else{
166         db.collection( collectionPath: "category" ) CollectionReference
167             .add(categoryData) Task<DocumentReference>
168             .addOnSuccessListener(docRef -> finish())
169             .addOnFailureListener(e -> {
170                 e.printStackTrace();
171             });
172     }

```

다음과 사진과 같이 동작한다. (15주차 작업)

- 카테고리 수정

The screenshots illustrate the steps for modifying a group category:

- Step 1: Select Group Members**
The first screenshot shows a list of group members with checkboxes. Some members have checked boxes, while others are empty. At the bottom are two buttons: "취소" (Cancel) and "저장" (Save).
- Step 2: Set Category Data**
The second screenshot shows the same list of members, but the checkboxes are now all checked. A large blue "저장" (Save) button is prominently displayed at the bottom.
- Step 3: Result View**
The third screenshot shows the "Category Management" screen. It lists categories with edit and delete icons. One category is highlighted with a blue border. At the bottom right is a blue circular button with a white plus sign (+), indicating a new category can be added.

카테고리 이름을 수정할 수 있으며, 체크박스를 통해 그룹원을 추가하거나 삭제할 수 있다. 저장 버튼을 누르면 변경 사항이 화면과 데이터베이스에 즉시 반영된다.

category > SeRbhMjorErg5...

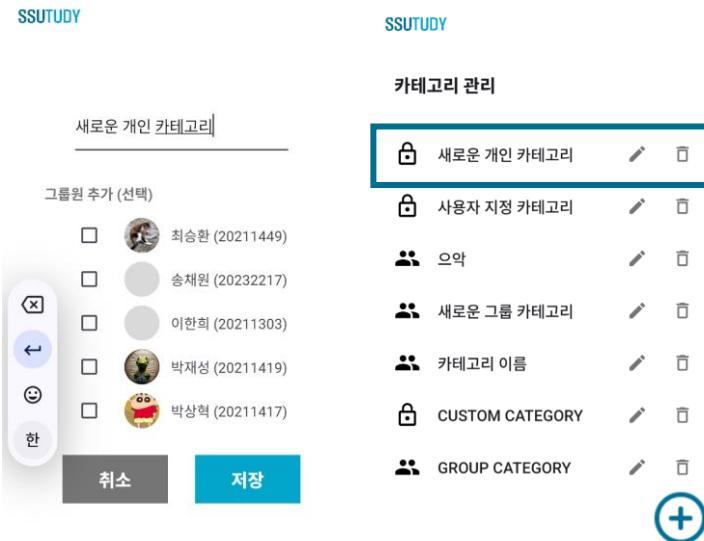
(default)	category	SeRbhMjorErg51XxqcX9
+ Start collection	+ Add document	+ Start collection
category >	BwRWAwpkaM73pNnWlPcY	+ Start collection
course	Efdz6UIMluHN418VigUr	+ Add field
friends	J1HW7gK378t50mpFQmC3	categoryName : "새로운 그룹 카테고리 설정"
request_friend	SeRbhMjorErg51XxqcX9 >	member
users	SuoNI1QLIU9GVU13ZzUm	0 "20232217"
	VslFt2U7j8o1T6SBxWuB	1 "20211417"
	oXoHF37LNUjsR2SkkLa0	rootuser: "20232372"
	pm8HT1qU6kv0v0UsyDeU	type: "group"
	uUXVvv5Pf86hHC9vUovI	
	vLgTDvZaCbD76uBUZtOg	
	yXLKtt90hsCkjT3nvSan	

- 카테고리 생성

The screenshots illustrate the process of creating a new group category:

- Left Screenshot:** A modal for adding a group category. It has a text input field labeled "카테고리 이름" (Category Name) containing "새로운 그룹 카테고리". Below it is a list of users with checkboxes: 최승환 (20211449), 송채원 (20232217), 이한희 (20211303), 박재성 (20211419), and 박상혁 (20211417). At the bottom are "취소" (Cancel) and "저장" (Save) buttons.
- Middle Screenshot:** A confirmation modal showing the selected users: 최승환 (20211449), 송채원 (20232217), 이한희 (20211303), 박재성 (20211419), and 박상혁 (20211417). It also features "취소" and "저장" buttons.
- Right Screenshot:** A "Category Management" screen listing categories. It includes columns for icon, name, edit, and delete. The "새로운 그룹 카테고리" entry is highlighted with a blue border. Other entries include "사용자 지정 카테고리", "으악", "카테고리 이름", "CUSTOM CATEGORY", and "GROUP CATEGORY". A large blue plus sign (+) button is at the bottom right.





카테고리 생성 페이지에서 그룹원을 선택하지 않으면 custom 카테고리로, 한 명 이상의 그룹원을 선택할 경우 group 카테고리로 자동으로 타입이 결정된다.

5) 로그아웃

이 앱은 사용자의 회원가입 정보를 저장하지 않고, 유세인트 인증을 받아 로그인하는 형식이다. 이 때문에 Firebase에서 제공하는 SignOut 기능을 사용할 수 없다. 따라서 로그아웃 버튼을 클릭하면, 이전에 열려 있던 모든 Activity를 종료하고, 새로 시작된 MainActivity만 남도록 구현하였다.

```

180
181
182     binding.logoutButton.setOnClickListener(new View.OnClickListener() {
183         @Override
184         public void onClick(View view) {
185             Intent intent = new Intent(getApplicationContext(), MainActivity.class);
186             intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_CLEAR_TASK);
187             startActivity(intent);
188             Toast.makeText(context, SettingsActivity.this, "로그아웃 되었습니다.", Toast.LENGTH_SHORT).show();
189         }
190     });

```

Intent.FLAG_ACTIVITY_NEW_TASK를 사용하여 새로운 Activity를 새로운 태스크에서 시작하도록 하고, Intent.FLAG_ACTIVITY_CLEAR_TASK를 사용하여 MainActivity를 시작할 때 해당 태스크에 이미 존재하는 모든 Activity를 제거한다. 이로 인해 현재 SettingsActivity를 포함한 이전의 모든 Activity가 종료된다.

IV. 최종 작업분해표

1. 최종 작업분해표

목차 / 페이지	주요메뉴	화면구성/기능	세부설명	담당자	진행도
		로그인	xml 생성		○
1-1 21			초기 설정 화면		○
		기능 개발	유세인트 연동 로그인		○
			별명, 프로필 사진 설정		○
		친구	자동로그인		○
1-2 23			친구 목록 화면		○
		기능 개발	친구 추가 dialog 화면		○
			친구 추가 요청 전송		○
		기록	과목 목록 화면		○
			과목별 공부시간 순위 화면		○
		기능 개발	스탑워치 화면		○
1-3 24			유세인트 강의목록 불러오기		○
		기능 개발	(개인) 과목별 공부시간 불러오기		○
			(랭킹) 과목별 공부시간 불러오고 순위 매기기		○
		스탑워치 기능 구현			
		쪽	총화면		○
2-5 31			하단 navigation bar		○
		기능 개발	수업 내용대로 navigation bar 수정하기		○
			base layout 만들기		○
		DB에서 종합 데이터 가져오기			
		캘린더	캘린더 및 to-do list 화면		○
2-3 29			일정 추가 dialog 화면		○
2-1 25		기능 개발	강의 수정 dialog 화면		○
2-2 27			일정 수정 dialog 화면		○
2-4 30		기능 개발	할 일 추가/삭제 dialog 화면		○
			벼락치기 마법사 dialog 화면		삭제
		기능 개발	일정, 강의 일정 DB 구축 및 calendar activity에 연동		○
			DB의 [카테고리]를 calendar activity에 연동		○
3-1 32		설정	카테고리에 할 일 추가 및 수정, 삭제 기능 개발		○
3-2 32			2024-2 강의 일정 자동 추가 및 수동 추가, 수정, 삭제 기능		○
3-3 33		기능 개발	개인일정 수정, 추가, 삭제 기능		○
3-4 35			벼락치기 마법사		삭제
3-5 38		기능 개발	설정 화면		○
			친구 요청 수락 dialog 화면		○
		기능 개발	닉네임 변경 dialog 화면		○
			카테고리 관리 화면		○
		기능 개발	카테고리 삭제 dialog 화면		○
			카테고리 추가, 수정 화면		○
		기능 개발	base layout 적용		○
			유저 정보 불러오기		○
		기능 개발	이미지 변경, 닉네임 변경		○
			친구 요청 수락		○
		기능 개발	카테고리 목록		○
			카테고리 삭제		○
		기능 개발	카테고리 추가, 수정 화면		○
			로그아웃		○

2. Firebase 사용 개요

- 1) Firestore Database
 - 유저 정보 관리
 - 유세인트를 통해 인증된 사용자 정보를 저장하고 관리한다.
 - 별명, 프로필 사진 링크, 이름, 접속 상태, 학번, 총 공부 시간 등의 정보를 포함한다.
 - 친구 요청 정보 관리
 - 친구 요청 관련 데이터를 관리한다.
 - 요청을 보낸 사람과 받은 사람의 정보를 저장한다.
 - 친구 목록 관리
 - 사용자별 친구 목록을 관리한다.
 - 요청을 보낸 사람과 받은 사람의 정보를 저장한다.
 - 수업 정보 및 일정 관리
 - 사용자별 수업 정보와 일정을 관리한다.
 - 강의 이름, 강의 코드, 강의별 누적 공부 시간, 강의 요일, 강의실 정보, 강의 시작 및 종료 시간, 수강 학생 학번 등의 데이터를 포함한다.
 - 개인 일정 관리
 - 날짜 별 사용자 일정을 관리한다.
 - 사용자 정보, 일정 내용, 장소, 일정 시작 시간을 포함한다.
 - 카테고리 정보 관리
 - 사용자별 카테고리 데이터를 관리한다.
 - Rootuser 정보, 카테고리 유형(group 또는 custom), 카테고리 이름, 그룹 멤버의 학번 등의 정보를 포함한다.
 - 각 카테고리 하위 컬렉션에 할 일 내용과 완료 여부를 포함한다.
- 2) Firebase Storage
 - 프로필 사진 관리
 - 사용자의 프로필 사진을 저장하고 불러온다.