수업 일정 관리

소프트웨어 설계서

작성자: 8조

20193098 최영은

20193051 김수진

20173127 안상욱

20113310 이희원

목차

**1. 프로젝트 개요4**

1.1. 프로젝트 목적4

1.2. 개발 환경4

**2. 화면 UI 설계5**

**3. Architecture 구조도**9

3.1. 소프트웨어 구조도9

3.2. SQLiteDB구조도10

3.3. FirebaseDB 구조도11

**4. Diagram12**

4.1. Use case Diagram 12

4.2. Activity Diagram 13

4.3. State Diagram 14

4.4. Class Diagram 15

4.4.1. Login16

4.4.2. SubjectInfo16

4.4.3. LectureInfo17

4.4.4. ExamInfo18

4.4.5. AssigmentInfo19

4.4.6. AlarmInfo20

4.4.7. FriendInfo22

4.4.8. TodoManagement23

4.4.9. AddAssigmentExam24

4.4.10. SubjectManagement24

4.4.11. AddSubject25

4.4.12. AlarmManagement26

4.4.13. AddAlarm27

4.4.14. FriendsManagement27

4.4.15. FriendsToDo27

4.4.16. SQLiteDB27

4.4.17. SQLiteHelper28

4.4.18. UploadInfo30

4.4.19. UploadExamInfo31

4.4.20. FirebaseDBHelper32

4.5. Sequence Diagram 35

4.4.1. 로그인35

4.4.2. 과목 추가36

4.4.3. 과목 삭제38

4.4.4. 시험과제 등록39

4.4.5. 시험과제삭제40

4.4.6. 수업일정 조회41

4.4.7. 친구목록 조회43

4.4.8. 친구 수업일정 조회44

4.4.9. 친구 추가45

4.4.10. 친구 삭제46

4.4.11. 알림 추가47

4.4.12. 알림 삭제48

**5. 프로젝트 계획49**

5.1. 프로젝트 일정표 49

5.2. 프로젝트 계획 50

1. **프로젝트 개요**
   1. 프로젝트 목적

본 문서는 소프트웨어 공학 1차 과제인 수업 일정 관리 소프트웨어 요구사항 명세서를 기반으로 하여 작성한 수업 일정 관리 소프트웨어 설계서이다. 이 프로젝트의 목적은 소프트웨어를 설계하는 것뿐만이 아닌 그 과정에 포함된 일련의 과정에 있다.

첫 번째로 주제를 정하면서 이 프로젝트가 정해진 기간 내에 가능한지, 별도의 비용이 들진 않는지 등등을 조사하여 가장 걸맞은 주제를 선택한다. 이번 프로젝트의 주제를 정할 때의 메인 테마가 “있으면 좋을 것 같다고 한 번쯤 생각해본 소프트웨어”였기 때문에 책의 예제를 수행했을 때에 비해 성취감이 더욱더 클 것이라 기대하고 있다.

두 번째로 코드를 짜기 전에 Architecture구조도, 유스케이스 다이어그램, 액티비티, 스테이트 다이어그램을 그리면서 전체적인 구조를 이해한다. 이는 개발하려는 소프트웨어의 코드를 구상하는데 있어 전체적인 그림을 미리 알고 작업할 수 있기 때문에 이 과정은 소프트웨어 개발의 초석이라 할 수 있다.

마지막으로 클래스 다이어그램과 시퀀스 다이어그램을 작성하면서 클래스 간의 관계와 의존관계를 파악하고 어떠한 순서로 어떤 객체들과 어떻게 상호작용할지를 연구하여 이에 적합한 코드를 조금 더 구조적으로 구상할 수 있게 해주는 것에 목적이 있다. 이를 수행하는 과정에서 팀원 간의 협동심을 기르고 계획한 일정에 맞추기 위해 노력함으로써 미래의 개발자가 되기 위한 과정에 한 걸음 더 다가갈 수 있을 것이다.

* 1. 개발 환경

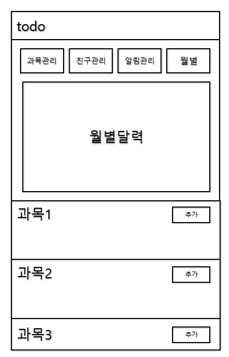
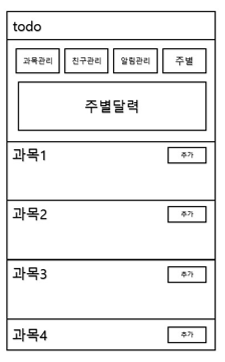
OS: Windows 10 64bit

개발 TOOL: 안드로이드 스튜디오 2020.3.1 for Windows 64-bit (914 MiB)

개발 언어: JAVA

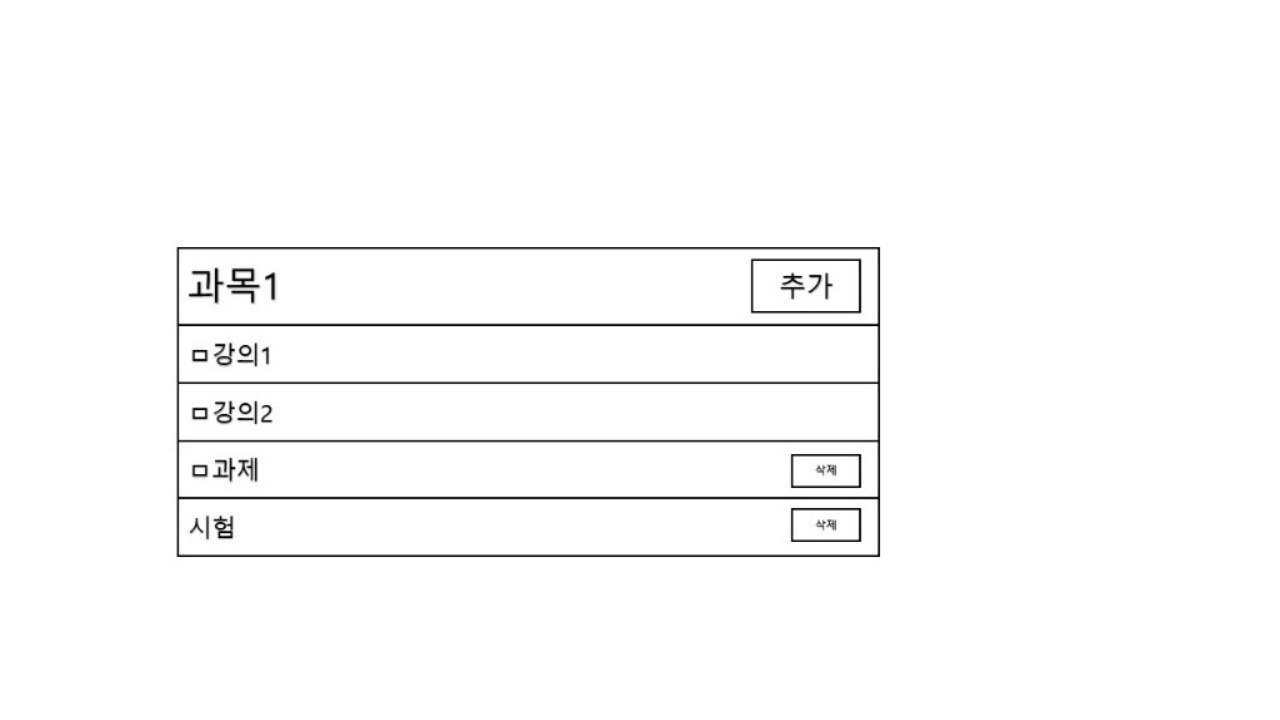
1. **화면 UI 설계**

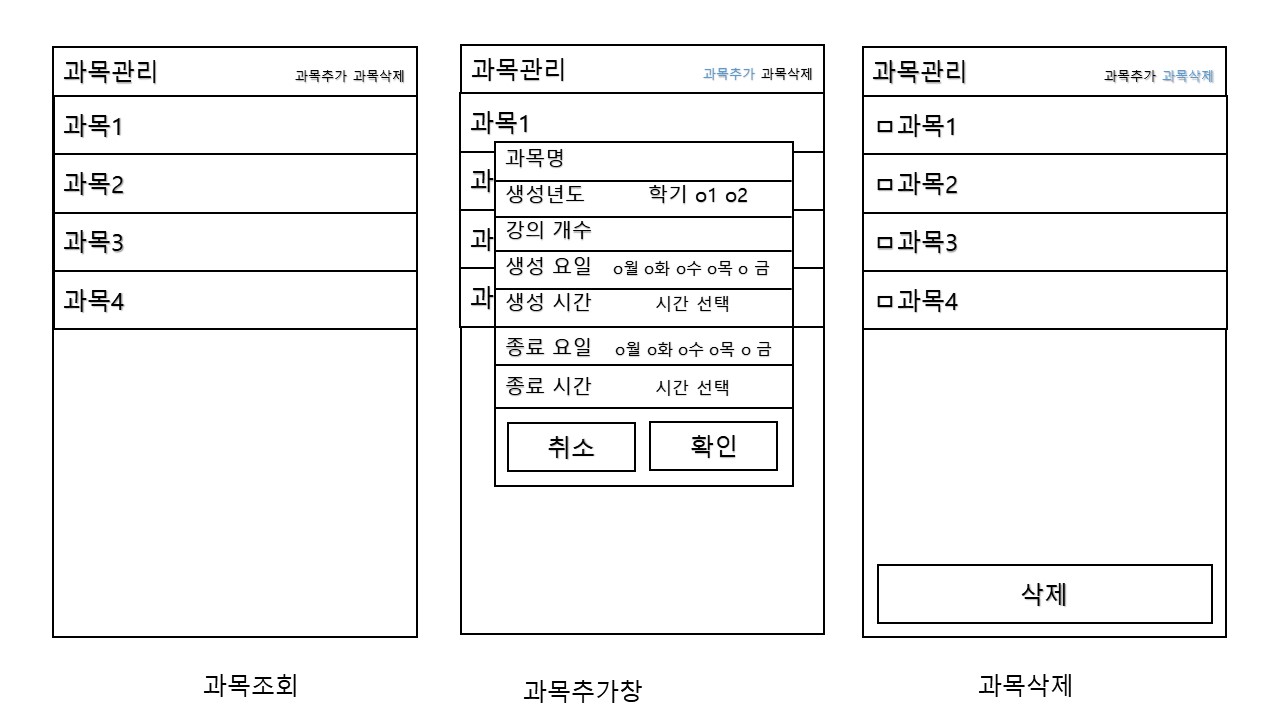
<주별 일정관리>



<월별 일정관리>

<일정목록 상세>



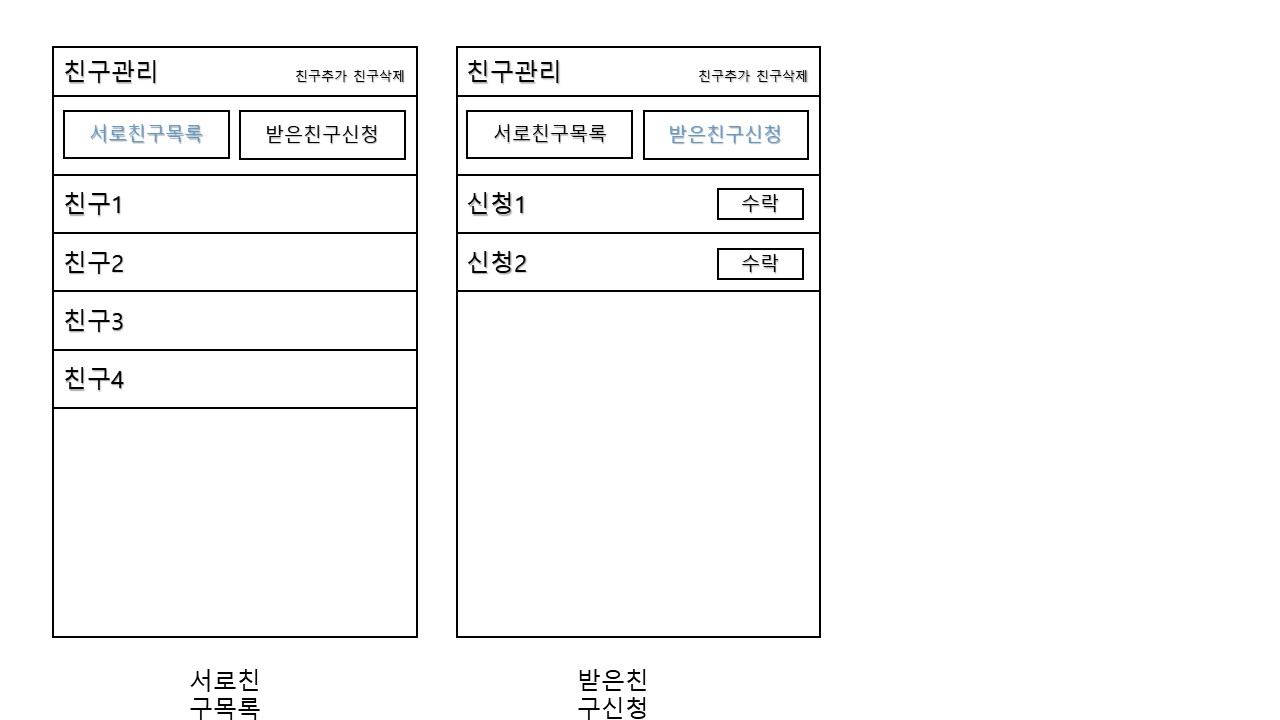


<과목 삭제>

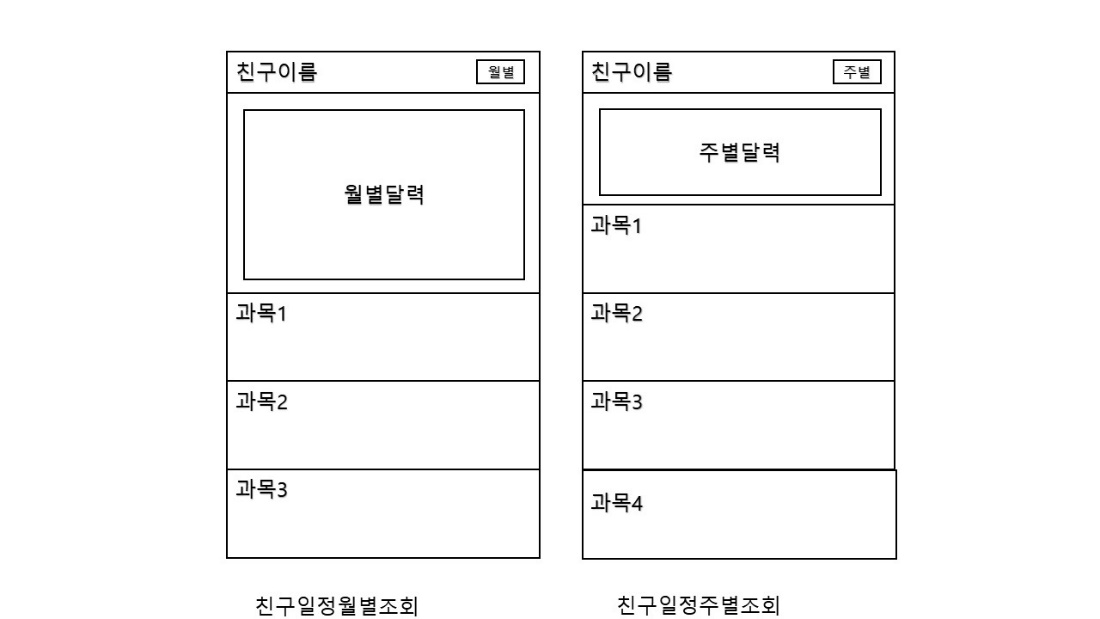
<과목 추가 창>

<과목 조회>

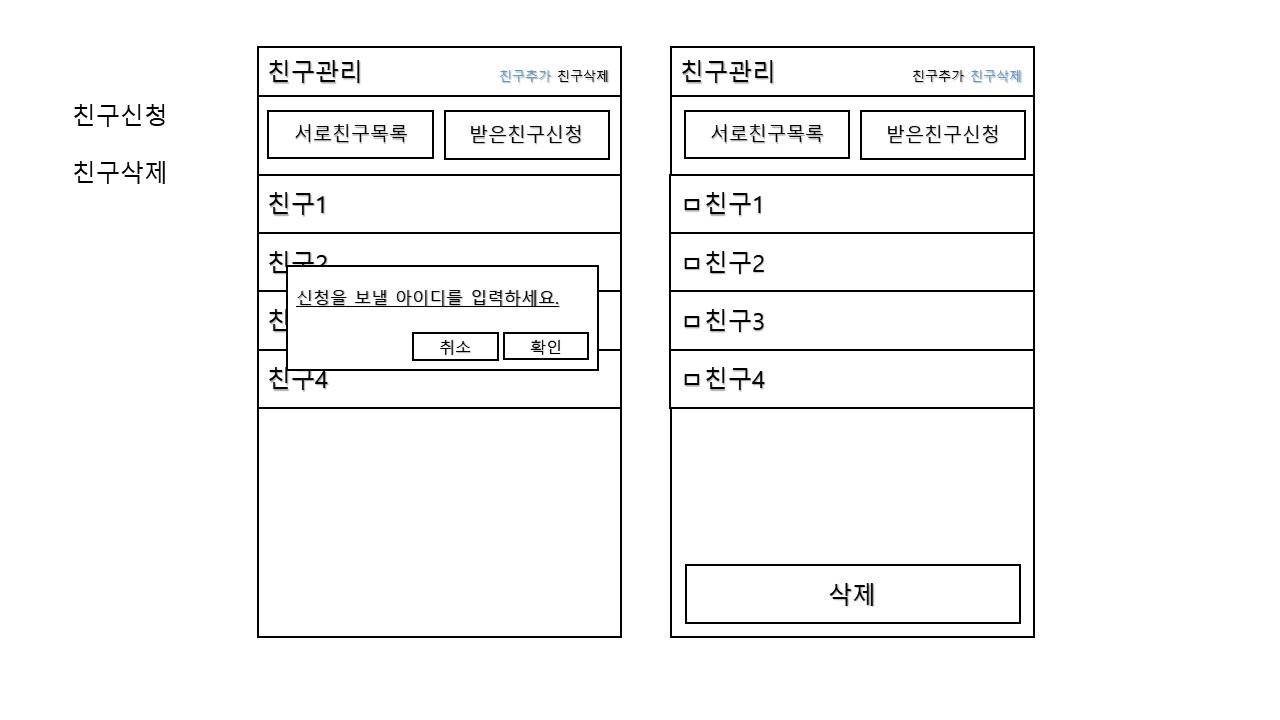




<서로 친구 목록> <받은 친구 신청>



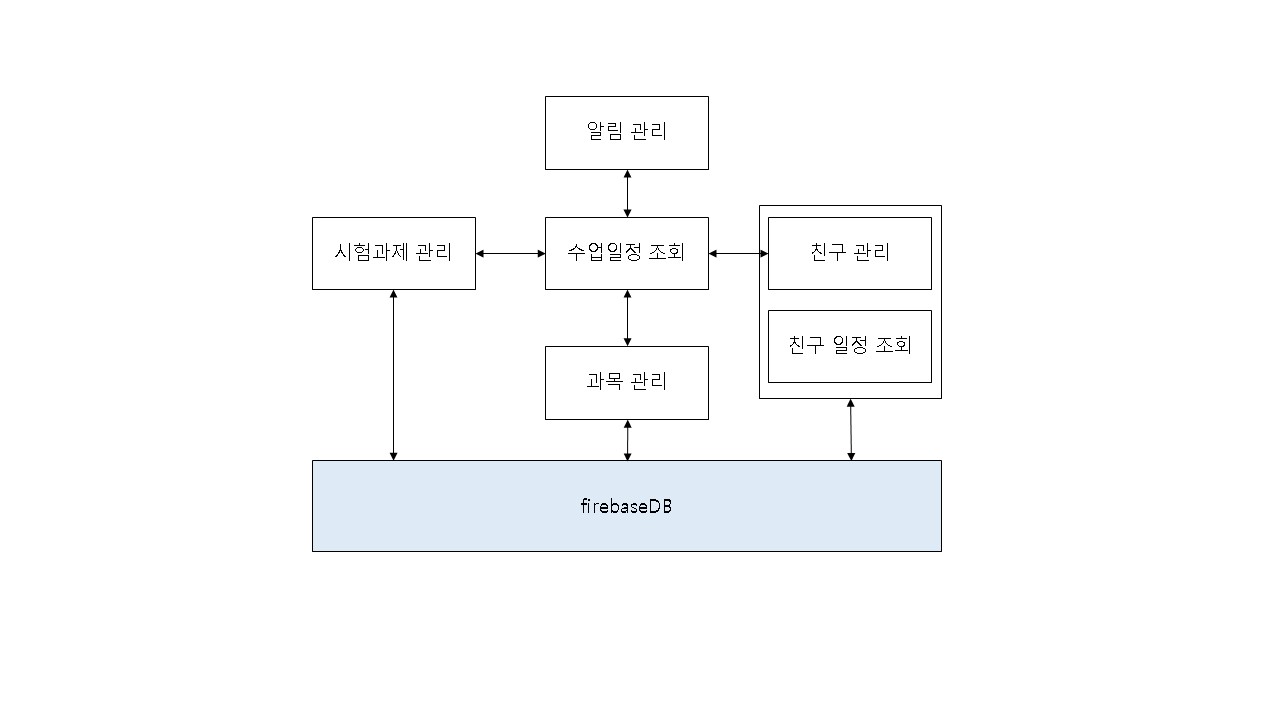
<친구 일정 월별 조회> <친구 일정 주별 조회>



<친구 신청> <친구 삭제>

1. **Architecture 구조도**
   1. 소프트웨어 구조도

본 소프트웨어의 Architecture구조도는 다음과 같다.



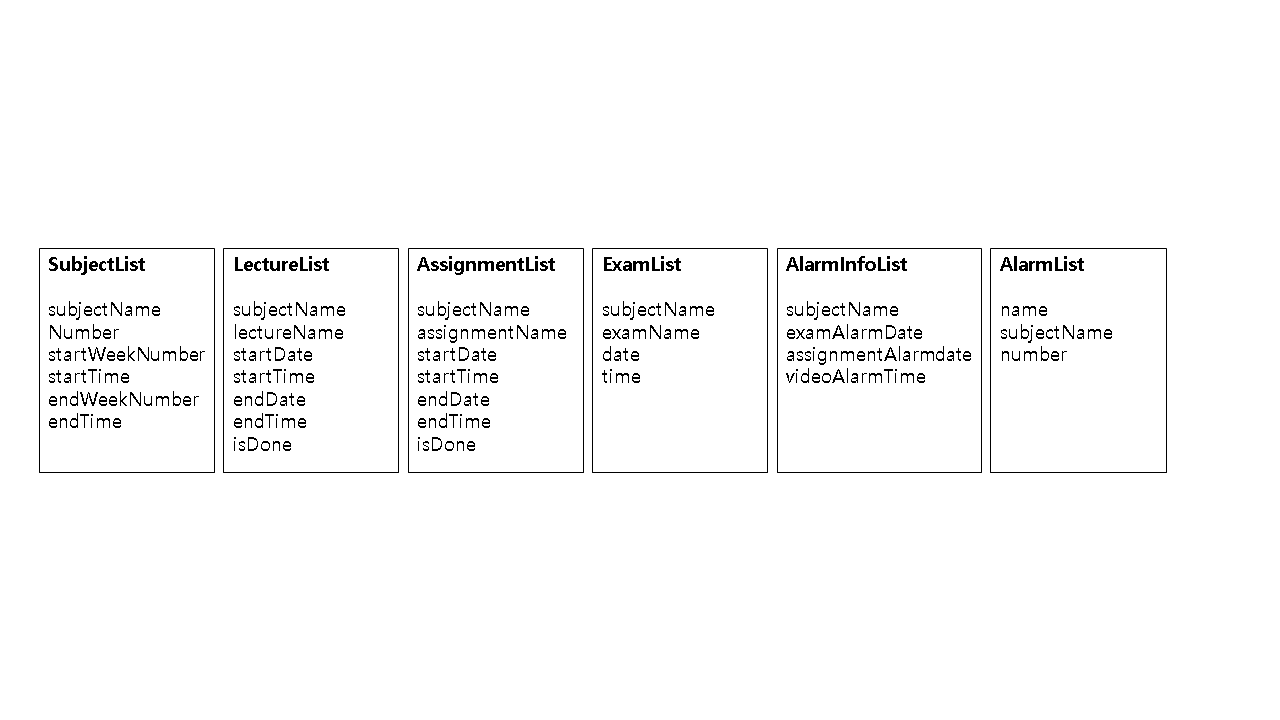
<Architecture 구조도>

메인 화면은 수업일정 조회 화면이고, 수업일정 조회 화면에서 과목 관리, 시험과제 관리, 알림 관리, 친구 관리로 이동할 수 있다. 시스템 데이터베이스로 SQLiteDB를 사용하여 사용자 자신의 데이터 저장한다. 시스템 데이터베이스는 구조도에 따로 표현하지 않았다. 외부 데이터베이스로는 FirebaseDB를 사용하여 친구의 정보를 저장하고 조회한다. FirebaseDB는 시험과제 관리, 과목 관리, 친구관리, 친구 일정 조회와 연동한다.

* 1. SQLiteDB 구조도

시스템 DB로 사용되는 SQLiteDB의 구조도는 다음과 같다.

<SQLiteDB 구조도>



과목, 강의, 과제, 시험, 알림에 대하여 각각 테이블을 만들어 해당 데이터들을 관리하도록 한다. sqlite는 foreign key를 설정할 수 없다. 따라서 과목을 삭제할 경우, 모든 테이블에서 각각 과목이름을 조건으로 삭제를 한다.

알림 리스트는 시스템 알림을 추가한 강의 또는 과제 또는 시험에 대하여 고유한 알림 번호를 생성하는 테이블이다. number에 AUTOINCREMENT속성을 추가하여 번호를 자동적으로 관리하도록 한다.

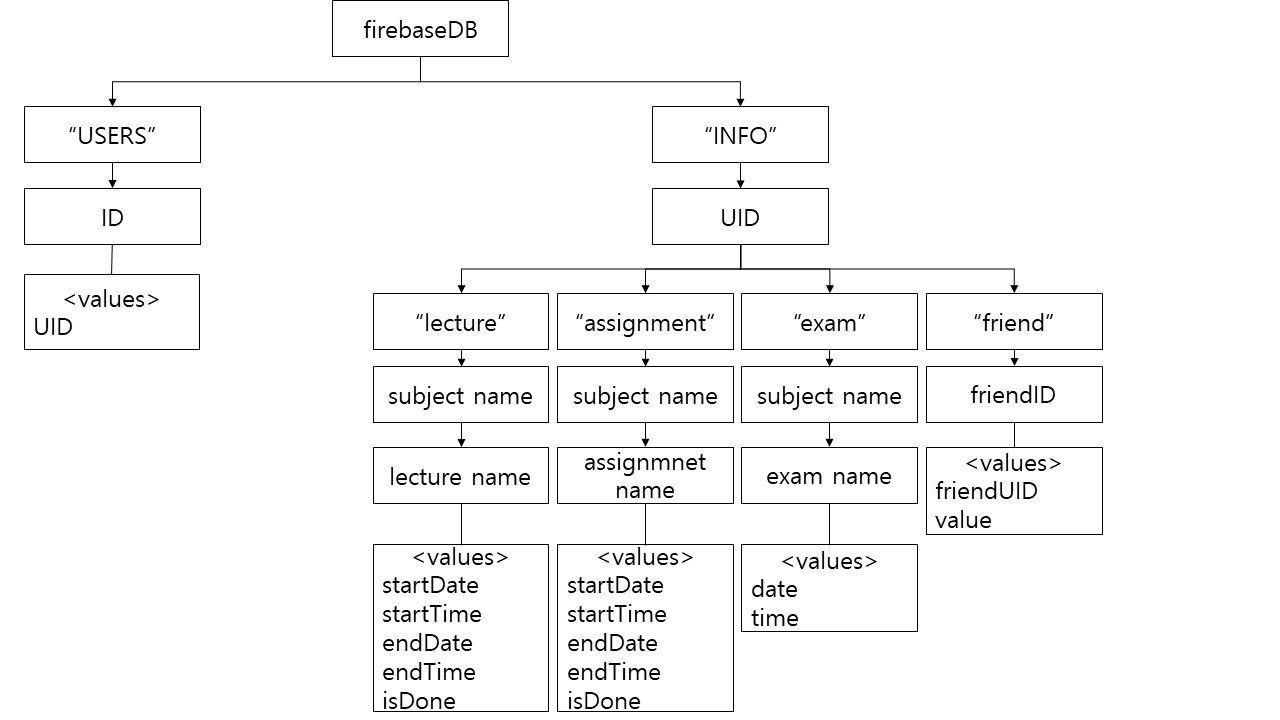
sqliteDB는 SQLiteOpenHelper를 상속받아 정의한다. DB를 create할 시 테이블이 없으면 생성하도록정의한다. sqliteDB에 연속적으로 접근하는 일이 많기 때문에 sqliteDB를 singleton 패턴으로 정의하고, sqliteDBHelper에서 sqliteDB를 getInstence()하여 데이터베이스를 관리하는 방식을 취하도록 한다.

friendList는 firebaseDB에서만 관리하고, sqliteDB에서는 관리하지 않는다.

* 1. FirebaseDB 구조도

firebaseDB 구조도는 다음과 같다.

<Firebase 구조도>



사용자 목록을 관리하는 USERS파트와 사용자의 정보들을 관리하는 INFO파트로 구성된다. USERS에는 사용자가 로그인을 진행하는 단 한번만 key:ID, value:UID로 사용자 등록을 진행한다. USERS는 이후 친구추가 기능에서 id가 사용자로 등록되어 있는지 검사할 때 사용한다. INFO는 사용자의 UID로 접근하여 정보를 저장하고 읽을 수 있다. 친구 정보의 value는 1: 서로친구상태, 0: 친구신청상태로, 친구 신청을 하면 해당 친구의 friend테이블에 접근하여 자신의 정보와 value 0으로 신청을 보낸다.

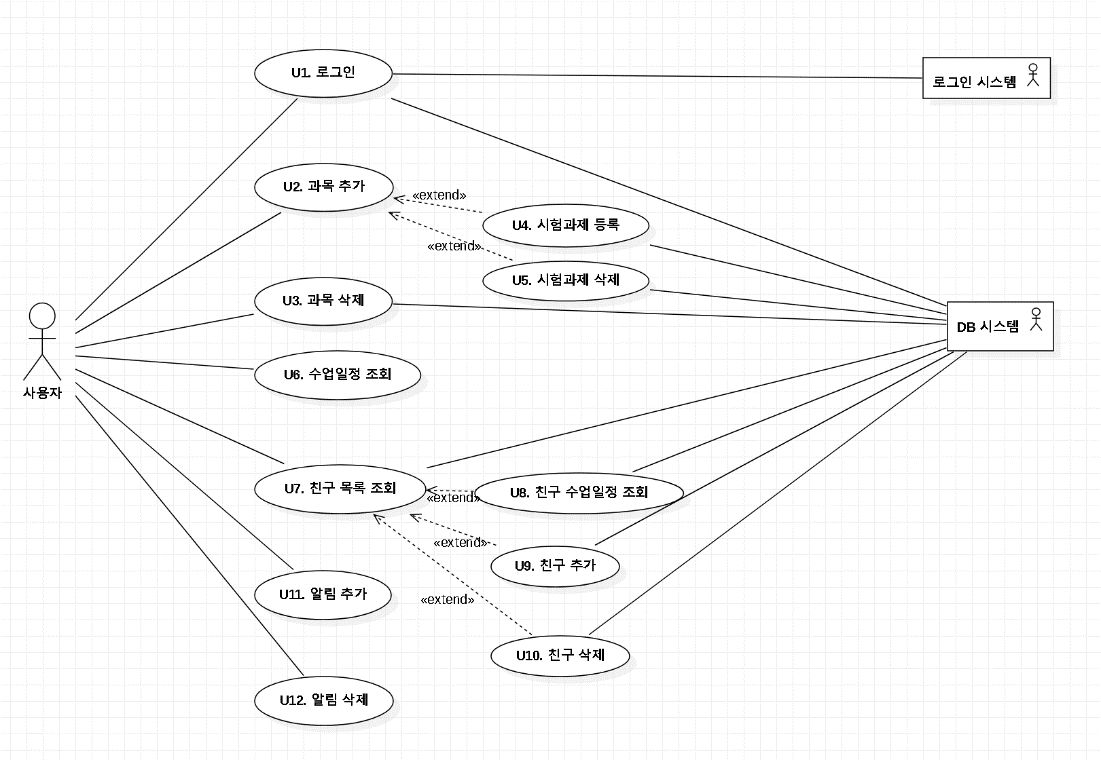
firebaseDB의 규칙을 사용하여 사용자의 강의, 과제, 시험 목록은 친구도 읽을 수 있지만, 작성은 사용자의 UID와 같을 때 만 가능하도록 하고, 친구 목록은 친구가 작성할 수 있지만, 읽기는 사용자의 UID와 같을 때 만 가능하게 하여 정보를 보안하도록 한다.

firebaseDB는 비동기로 작동하기 때문에 값을 가져올 때에는 Listener를 사용해야 한다. 따라서 값을 받아오는 함수에는 return을 정의하지 않고, 이전 클래스가 생성될 때 static 변수로 context를 선언하고, 해당 context의 메소드를 호출하여 값의 전달 및 값의 변화를 알려주어 화면을 설정하는 방식을 취하도록 한다.

firebaseDB는 오프라인일 경우, 캐시된 데이터를 사용할 수 있고, 네트워크 연결이 복원되면 firebase에서 모든 쓰기작업을 다시 전송한다. 따라서 firebaseDB에 데이터 저장에 대한 결과값을 따로 반환하지 않도록 설계한다.

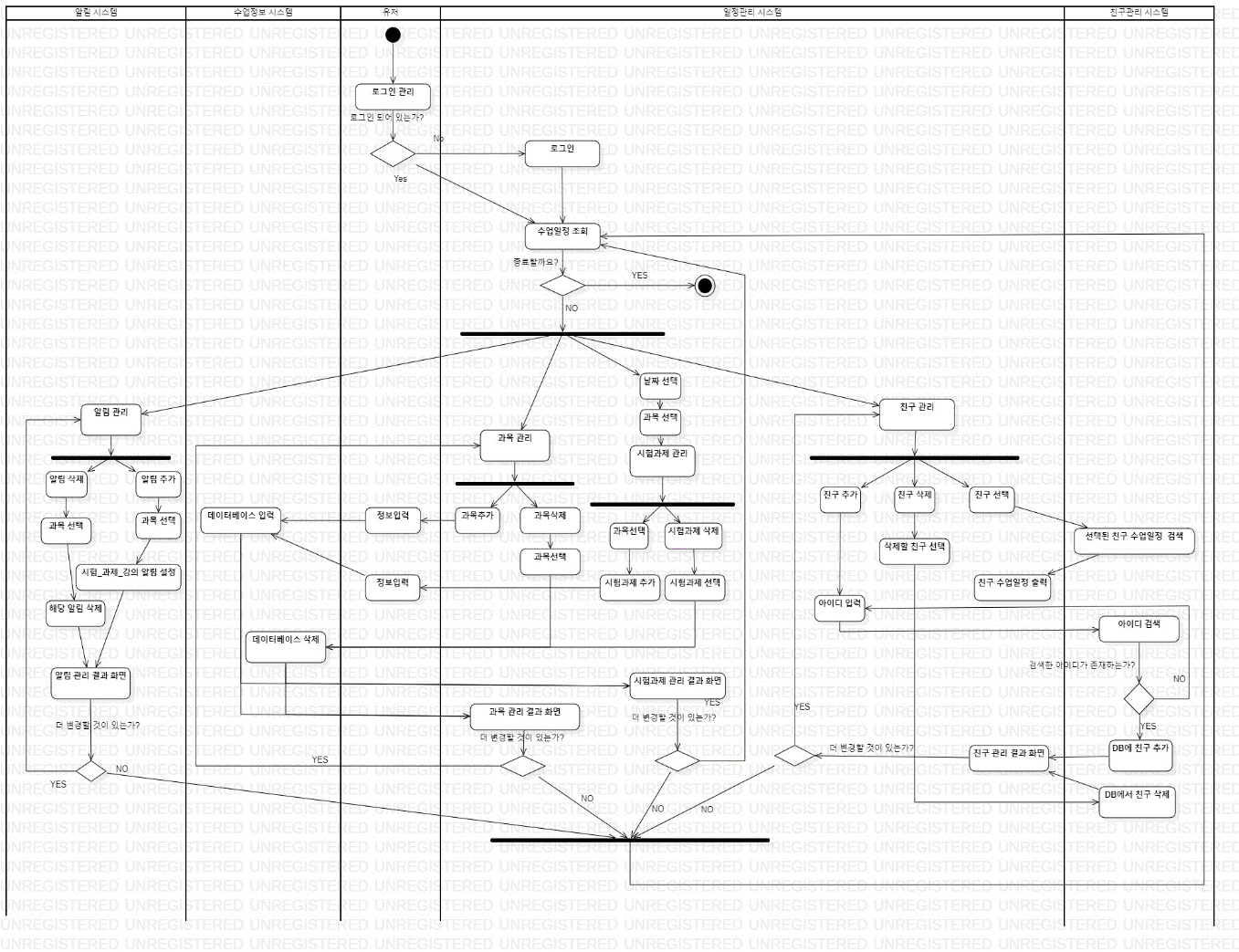
1. **Diagram**
   1. Use case Diagram

<Use case Diagram>



각각의 유즈케이스에 대한 명세서는 요구사항 명세서에 기재 되어있다.

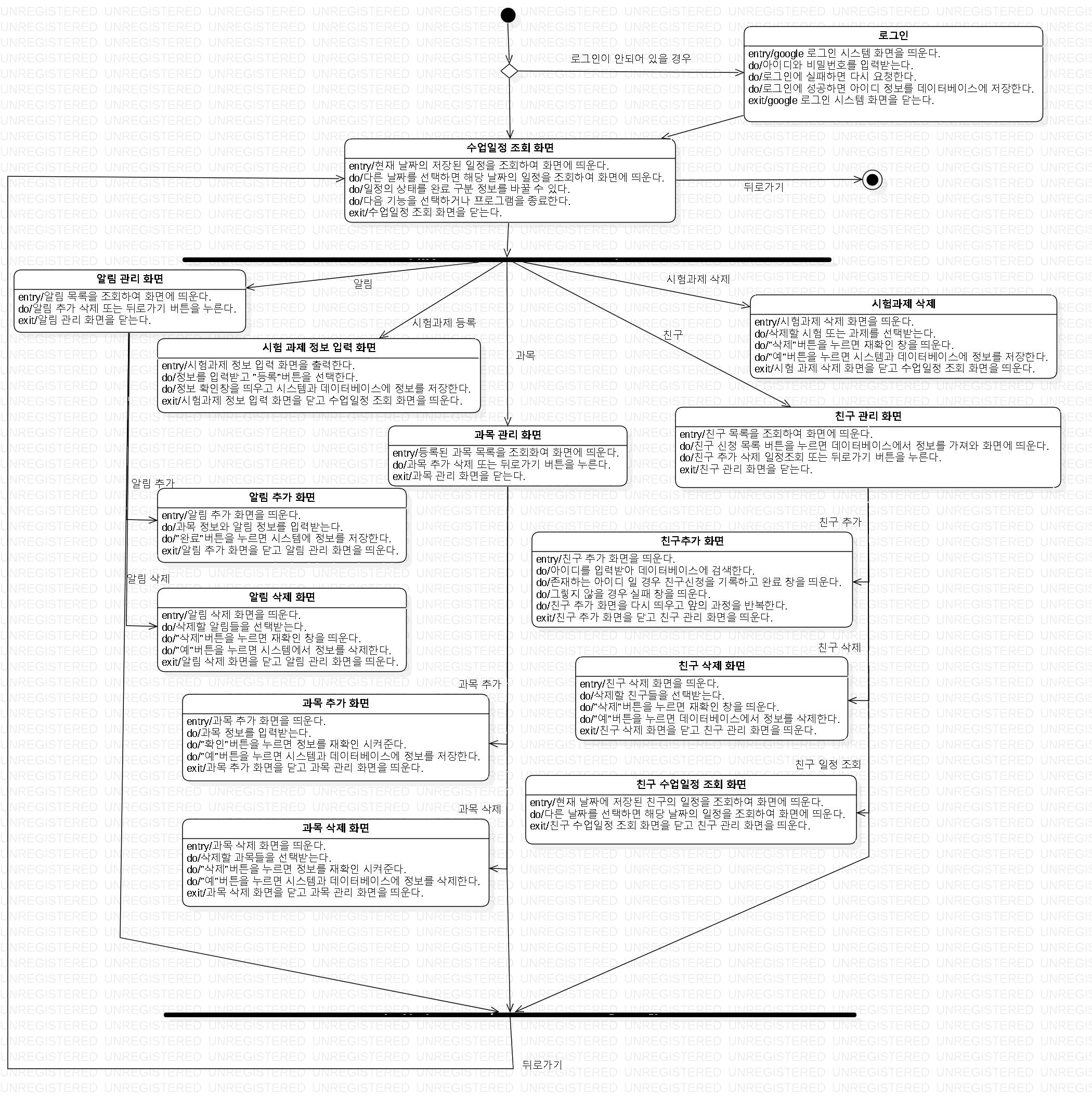
* 1. Activity Diagram



<Activity Diagram>

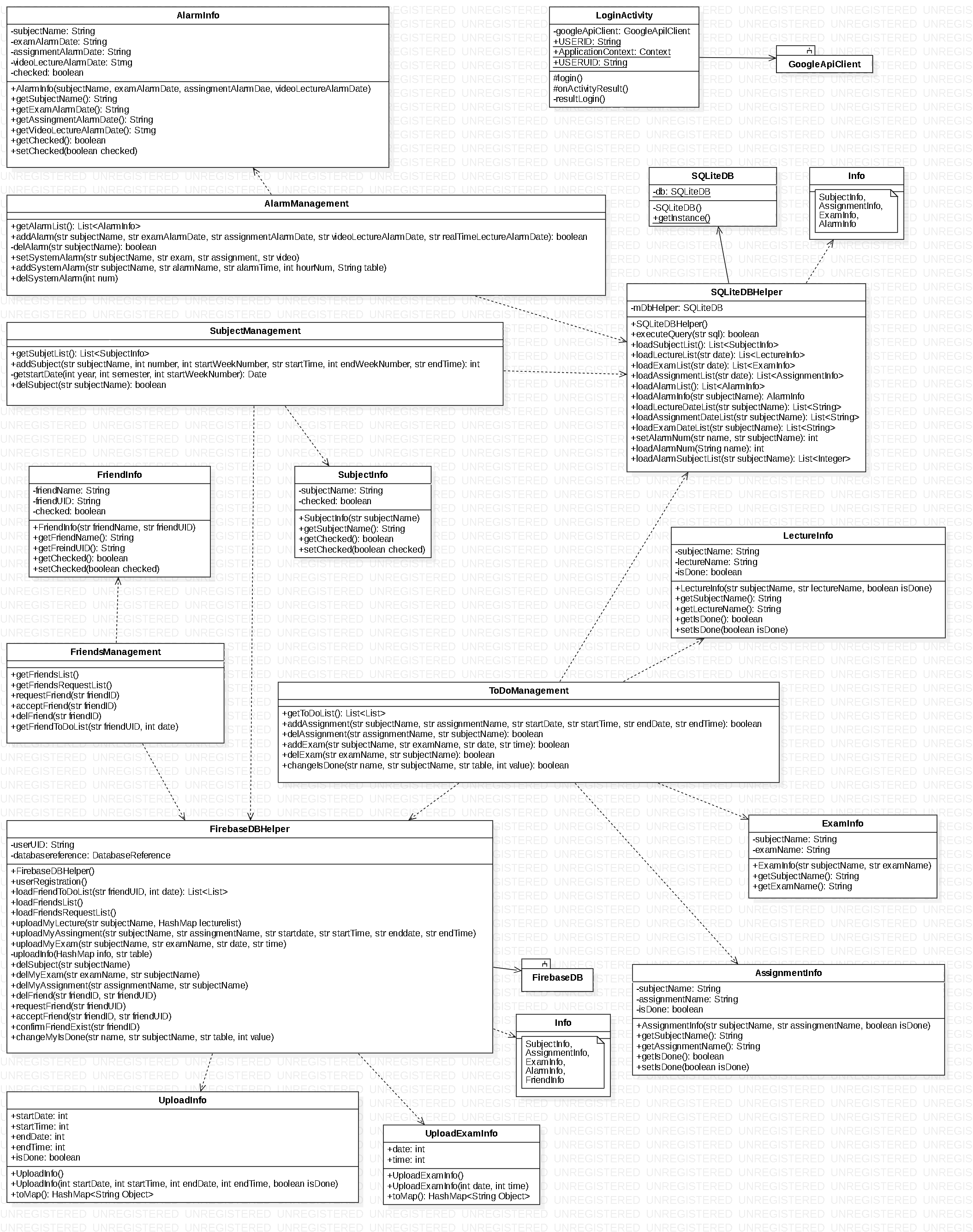
* 1. State Diagram

<State Diagram>



|  |  |
| --- | --- |
| 자극 | 설명 |
| 알림 | 사용자가 수업일정 조회 화면의 “알림” 버튼을 클릭한다. |
| 과목 | 사용자가 수업일정 조회 화면의, “과목 관리” 버튼을 클릭한다. |
| 시험과제 등록 | 사용자가 수업일정 조회 화면에서 과목 하나를 선택하여 “시험과제 추가” 버튼을 클릭한다. |
| 시험과제 삭제 | 사용자가 수업일정 조회 화면의 “시험과제 삭제” 버튼을 클릭한다. |
| 친구 | 사용자가 수업일정 조회 화면의 “친구” 버튼을 클릭한다. |
| 알림 추가 | 사용자가 알림 관리 화면의 “알림 추가” 버튼을 클릭한다. |
| 알림 삭제 | 사용자가 알림 관리 화면의 “알림 삭제” 버튼을 클릭한다. |
| 과목 추가 | 사용자가 과목 관리 화면의 “과목 추가” 버튼을 클릭한다. |
| 과목 삭제 | 사용자가 과목 관리 화면의 “과목 삭제” 버튼을 클릭한다. |
| 친구 추가 | 사용자가 친구 관리 화면의 “친구 추가” 버튼을 클릭한다. |
| 친구 삭제 | 사용자가 친구 관리 화면의 “친구 삭제” 버튼을 클릭한다. |
| 친구 일정 조회 | 사용자가 친구 관리 화면에서 친구 한 명을 선택하여 “친구 수업일정 조회” 버튼을 클릭한다. |
| 뒤로 가기 | 사용자가 뒤로 가기 기능을 사용한다. |

* 1. Class Diagram



<Class Diagram>

SQLiteDBHelper의 메소드 안에서 SubjectInfo, LectureInfo, AssingmentInfo, ExamInfo를 생성하고 반환하기 때문에 의존관계를 가진다. 또한 FirebaseDBHelper의 메소드 안에서 SubjectInfo, LectureInfo, AssingmentInfo, ExamInfo, FriendInfo를 생성하고 반환하기 때문에 의존관계를 가진다. 이 관계들을 class diagram에서 나타내고자 할 때 선이 매우 복잡해짐으로, 간소하게 Info 클래스로 나타냈다.

각각의 클래스에 대한 설명은 다음과 같다.

* + 1. **LoginActivity**

사용자가 어플리케이션을 실행하였을 때 가장 먼저 실행되는 클래스로 로그인 확인을 진행하고, 로그인이 안되어 있을 경우, 로그인을 실행한다.

Attribute

* -GoogleApiClient: GoogleApiClient
* 구글 API 클라이언트 객체이다.
* +USERUID: String
* Firebase에 정보를 저장할 때 사용하는 사용자의 UID를 저장한다.
* +USERID: String
* Firebase에 정보를 저장할 때 사용하는 사용자의 ID를 저장한다.
* +ApplicationContext : Context
* 어플리케이션의 cotext정보로, sqliteDB를 생성할 때 단 한 번 필요로 한다.

Operation

* #Login(): void
* 로그인이 이미 되어 있다면 바로 일정조회화면으로 넘어가고, 아니라면 구글로그인 화면으로 넘어가서 로그인을 진행한다.
* #onActivityResult(): void
* 구글로그인을 인증을 요청한 후 결과 값을 되돌려 받고, resultLogin()을 호출한다.
* -resultLogin(): void
* 로그인이 성공했다면 USERID와 USERUID를 이용하여 FirebaseDBHelper의 login을 호출하고 firebaseDB에 사용자를 등록한다. 실패한 경우 로그인이 실패했다는 메시지를 보여준다.
  + 1. **SubjectInfo**

과목정보를 sqlite에서 들고와서 화면에 보여주고자 할 때 정보를 List<SubjectInfo>와 같이 넘겨주기 위하여 사용하는 클래스이다. 변수는 private로 존재한다. subjectName에 대하여 메소드를 getter만 두어 Constructor로 생성시에만 초기화할 수 있도록 하고, checked는 getter와 setter를 만들어 과목 삭제 시 체크 값을 설정 및 받아올 수 있도록 한다.

Attribute

* -subjectName: String
* 과목명 정보이다. 생성자에 의해서만 값을 설정할 수 있고, 값을 변경할 수 없다.
* -checked: boolean
* 과목 삭제 화면에서 체크박스 체크 여부를 확인하기 위해 추가된 변수이다. true일 경우 체크상태, false일 경우 비 체크 상태를 나타내며, false 상태로 생성된다.

Operation

* +SubjectInfo(str subjectName): void
* 생성자이다. 생성될 때 subjectName 변수를 초기화한다.
* +getSubjectName(): String
* subjectName 변수의 값을 String 형태로 반환한다.
* +getchecked(): boolean
* checked 변수의 값을 boolean 형태로 반환한다.
* +setchecked(boolean checked): void
* checked 변수의 값을 설정한다.
  + 1. **LectureInfo**

강의 정보를 sqlite 또는 firebase에서 들고 와서 화면에 보여주고자 할 때 정보를 List<LectureInfo>형태로 넘겨주기 위하여 사용하는 클래스이다. 변수는 private로 존재한다. subjectName과 letureName에 대하여 메소드를 getter만 두어 Constructor로 생성시에만 초기화 할 수 있도록 하고, isDone에 대한 메소드로는 getter와 setter를 두어 이후에도 정보를 수정할 수 있도록 한다.

Attribute

* -subjectName: String
* 과목명 정보이다.
* -lectureName: String
* 강의명 정보이다.
* -isDone: boolean
* 강의의 일정 완료 여부 정보이다.

Operation

* +LectureInfo(str subjectName, str lectureName, boolean isDone): void
* 생성자이다. SubjectName, lectureName, isDone 변수들을 초기화한다.
* +getSubjectName(): String
* subjectName 변수를 string 형태로 반환한다.
* +getLectureName(): String
* lectuerName 변수를 string 형태로 반환한다.
* +getIsDone(): boolean
* isDone 변수를 boolean 형태로 반환한다.
* +setisDone(boolean isDone): void
* isDone 변수의 값을 설정한다.
  + 1. **ExamInfo**

시험 정보를 sqlite 또는 firebase에서 들고 와서 화면에 보여주고자 할 때 정보를 List<ExamInfo>형태로 넘겨주기 위하여 사용하는 클래스이다. 변수는 private로 존재한다. subjectName과 examName에 대하여 메소드를 getter만 두어 Constructor로 생성시에만 초기화 할 수 있도록 한다.

Attribute

* -subjectName: String
* 과목명 정보이다.
* -examName: String
* 시험명 정보이다.

Operation

* +ExamInfo(subjectName, examName): void
* 생성자이다. subjectName, examName 변수들을 초기화한다.
* +getSubjectName(): String
* subjectName 변수를 string 형태로 반환한다.
* +getExamName(): String
* ExamName 변수를 string 형태로 반환한다.
  + 1. **AssignmentInfo**

과제 정보를 sqlite 또는 firebase에서 들고 와서 화면에 보여주고자 할 때 정보를 List<AssignmentInfo>형태로 넘겨주기 위하여 사용하는 클래스이다. 변수는 private로 존재한다. subjectName과 assignmentName에 대하여 메소드를 getter만 두어 Constructor로 생성시에만 초기화 할 수 있도록 하고, isDone에 대한 메소드로는 getter와 setter를 두어 이후에도 정보를 수정할 수 있도록 한다.

Attribute

* -subjectName: String
* 과목명 정보이다.
* -assignmentName: String
* 과제명 정보이다.
* -isDone: boolean
* 과제의 일정 완료 여부 정보이다.

Operation

* +AssignmentInfo(str subjectName, str assignmentName, boolean isDone): void
* 생성자이다. SubjectName, assignmentName, isDone 변수들을 초기화한다.
* +getSubjectName(): String
* subjectName 변수를 string 형태로 반환한다.
* +getAssignmentName(): String
* assignmentName 변수를 string 형태로 반환한다.
* +getIsDone(): boolean
* IsDone 변수를 boolean 형태로 반환한다.
* +setIsDone(boolean): void
* isDone 변수의 값을 설정한다.
  + 1. **AlarmInfo**

강의 정보를 sqlite에서 들고와서 화면에 보여주고자 할 때 정보를 List<Alarm>형태로 넘겨주기 위하여 사용하는 클래스이다. 변수는 private로 존재하며, subjectName, examAlarmDate,   
assignmentAlarmDate, videoLectureAlarmDate 에 대하여 메소드를 getter만 두어 생성시에만 초기화할 수 있도록 하고, checked에 대하여 메소드를 getter와 setter를 두어 알림 삭제 시 체크값을 설정 및 받아올 수 있도록 한다.

Attribute

* -subjectName: String
* 과목명 정보이다.
* -examAlarmDate: String
* 시험의 알림을 언제 받을 것인지에 대한 정보이다.
* -assignmentAlarmDate: String
* 과제의 알림을 언제 받을 것인지에 대한 정보이다.
* -videoLectureAlarmDate: String
* 동영상 강의에 대한 알림을 언제 받을 것인지에 대한 정보이다. 강의유형을 선택해서 입력 받으므로 빈 값이 들어올 수 있다.
* -checked: boolean
* 알람 정보 화면에서 체크박스 체크 여부를 확인하기 위해 추가된 변수이다. true일 경우 체크상태, false일 경우 비 체크 상태를 나타내며, false 상태로 생성된다.

Operation

* +AlarmInfo(subjectName, examAlarmDate, assignmentAlarmDate, videoLectureAlarmDate): void
* 생성자이다. subjectName, examAlarmDate, assignmentAlarmDate, videoLectuerAlarmDate 변수들의 값을 초기화한다.
* +getSubjectName(): String
* subjectName 변수를 string 형태로 반환한다.
* +getExamAlarmDate(): String
* examAlarmDate 변수를 string 형태로 반환한다.
* +getAssignmentAlarmDate(): String
* assignmentAlarmDate 변수를 string 형태로 반환한다.
* +getVideoLectureAlarmDate(): String
* videoLectureAlarmDate 변수를 string 형태로 반환한다.
* +getChecked():boolean
* checked 변수의 값을 boolean 형태로 반환한다
* +setChecked(boolean checked): void
* checked 변수의 값을 설정한다.
  + 1. **FriendInfo**

친구 정보를 firebase에서 들고 와서 화면에 보여주고자 할 때 정보를 List<FriendInfo>형태로 넘겨주기 위하여 사용하는 클래스이다. 변수는 private로 존재한다. friendName 과 friendUID에 대하여 메소드를 getter만 두어 Constructor로 생성시에만 초기화 할 수 있도록 하고, cheked에 대한 메소드로는 getter와 setter를 두어 친구 삭제 시 체크 값을 설정 및 받아올 수 있도록 한다.

Attribute

* -friendName: String
* 친구의 ID정보이다.
* -friendUID: String
* 친구의 UID 정보이다.
* -cheked: boolean
* 강의의 일정 완료 여부 정보이다.

Operation

* +FriendInfo(str friendName, str friendUID): void
* 생성자이다. friendName, friendUID 변수들을 초기화한다.
* +getFriendName(): String
* friendName변수를 string 형태로 반환한다.
* +getFriendUID(): String
* friendUID변수를 string 형태로 반환한다.
* +getChecked(): boolean
* checked변수를 boolean 형태로 반환한다.
* +setChecked(boolean checked): void
* checked변수의 값을 설정한다.
  + 1. **ToDoManagement**

시스템의 일정 조회 화면에서 사용되는 기능들을 정의한 클래스이다. 할 일 목록을 가져오거나 목록에 시험 또는 과제를 추가, 삭제할 수 있고 할 일에 대한 체크변경이 가능하다.

Operation

* +getToDoList(str date): List<List>
* 사용자가 선택한 날짜에 대한 할 일 목록을 불러온다. SQLiteDBHeper의 loadLectureList(), loadAssignmentList(), loadExamList()를 사용하여 불러온 리스트들을 묶어 List 형태로 반환한다.
* +addAssignment(str subjectName, str name, str startDate, str startTime, str endDate, str endTime): boolean
* 사용자가 입력한 과제정보들을 조합하여 query를 생성하고, SQLiteDBHelper의 executeQuery()를 호출하여 sqliteDB에 과제 정보를 저장한다. firebseDB에 저장하기 위하여 FirebaseHelper의 uploadMyAssignment()를 호출한다. SQLiteDBHelper의 loadAlarmInfo()를 호출하여 과목에 알림이 설정되어 있는지 확인한다. 만약 과목에 알림이 설정되어 있다면, 과제에 대하여 AlarmManagement의 addSystemAlarm()을 호출하여 시스템 알림을 추가한다.
* +delAssignment(str assignmentName, str subjectName): boolean
* 과제 목록 중에서 사용자가 선택한 과제의 과제명, 과목명 정보를 받아와 query를 생성하고, SQLiteDBHelper의 executeQuery()를 호출하여 sqliteDB에서 정보를 삭제한다. firebaseDB에에서 삭제하기 위하여 FirebaseHelper의 delMyAssignment()를 호출한다. SQLiteDBHelper의 loadAlarmInfo()를 호출하여 과목에 알림이 설정되어 있는지 확인한다. 만약 과목에 알림이 설정되어 있다면, 과제에 대하여 AlarmManagement의 delSystemAlarm()을 호출하여 시스템 알림을 삭제한다.
* +addExam(str subjectName, str name, str Date): boolean
* 사용자가 입력한 시험 정보를 조합하여 query를 생성하고, SQLiteDBHelper의 executeQuery()를 호출하여 sqlite에 시험 정보를 저장한다. firebaseDB에 저장하기 위하여 FirebaseHelper의 uploadMyExam()을 호출한다. SQLiteDBHelper의 loadAlarmInfo()를 호출하여 과목에 알림이 설정되어 있는지 확인한다. 만약 과목에 알림이 설정되어 있다면, 시험에 대하여 AlarmManagement의 addSystemAlarm()을 호출하여 시스템 알림을 추가한다.
* +delExam(str examName, str subjectName): boolean
* 시험 목록중에서 사용자가 선택한 시험의 시험명, 과목명 정보를 받아와 query를 생성하고 SQLiteDBHelper의 executeQuery()를 호출하여 sqliteDB에서 정보를 삭제한다. FirebaseDB에서 삭제하하기 위하여 FirebaseHelper의 delMyExam()을 호출한다. SQLiteDBHelper의 loadAlarmInfo()를 호출하여 과목에 알림이 설정되어 있는지 확인한다. 만약 과목에 알림이 설정되어 있다면, 시험에 대하여 AlarmManagement의 delSystemAlarm()을 호출하여 시스템 알림을 삭제한다.
* +changeIsDone(str name, str subjectName, str table, int value): boolean
* 강의, 과제 목록 중에서 사용자가 선택한 강의, 과제의 이름, 강의명, “Lecture”또는 “Assignment”값을 가지는 테이블이름, 덮어써지는 isDone값을 받아와 query를 생성하고, SQLiteDBHelper의 executeQuery()를 호출하여 sqliteDB에서 정보를 삭제한다. firebaseDB의 값을 변경하기 위하여 FirebaseHelper의 changeMyIsDone()을 호출한다. SQLiteDBHelper의 loadAlarmInfo()를 호출하여 과목에 알림이 설정되어 있는지 확인한다. 만약 과목에 알림이 설정되어 있다면, isdone을 false로 설정해야 하는 경우, AlarmManagement의 delSystemAlarm()을 호출하여 시스템 알림을 삭제하고, isdone을 true로 설정해야 하는 경우, AlarmManagement의 addSystemAlarm을 호출하여 시스템 알림을 추가한다.
  + 1. **SubjectManagement**

시스템의 과목 관리 화면에서 사용되는 기능들을 정의한 클래스이다. 과목 목록을 가져오거나 목록에 과목 추가, 삭제를 할 수 있고 강의 추가를 위한 시작 날짜 계산이 가능하다.

Operation

* +getSubjectList(): List<SubjectInfo>
* SQLiteDBHelper의 loadSubjectList()를 호출하여 과목목록을 받아와서 반환한다.
* +addSubject(str subjectName, int number, int startWeekNumber, str startTime, int endWeekNumber, str endTime): int
* 사용자가 입력한 과목 정보들을 조합하여 query를 생성하고, SQLiteDBHelper의 executeQuery()를 호출하여 sqliteDB에 정보를 저장한다. 강의정보를 저장하기 위하여 getstartDate()를 호출하여 1주차 강의의 시작 날짜를 구한다. 그 후, 1주차부터 16주차까지 강의를 sqliteDB와 firebaseDB에 저장한다. sqliteDB에는 16개 강의를 추가하는 query를 생성하여 SQLiteDBHelper의 executeQuery()를 호출하여 저장하고, firebaseDB에는 각각의 강의정보를 HashMap으로 저장하고, 16개의 강의를 HashMap<강의명, 강의정보HashMap>으로 저장하여 FirebaseHelper의 uploadMyLecture()을 호출하여 강의를 저장한다. 반환값으로는 추가에 성공한경우 1, 존재하는 과목을 추가한 경우 0, 입력할 수 없는값을 입력한 경우 -1을 반환한다.
* +delSubject(str subjectName): boolean
* 사용자가 과목목록에서 선택한 과목에 대하여 과목목록, 강의목록, 과제목록, 시험목록에서 해당 과목명에 대한 정보를 삭제하는 query를 생성하여 SQLiteDBHelper의 executeQuery()를 호출하여 sqliteDB에서 정보를 삭제한다. SQLiteDBHelper의 loadAlarmInfo()를 호출하여 과목에 알림이 설정되어 있는지 확인한다. 만약 과목에 알림이 설정되어 있다면, AlarmManagement의 delSubjectAlarm()을 호출하여 과목전체의 시스템 알림을 삭제한다. 그 후, sqliteDB의 알림 정보 리스트와 알림 리스트에서 해당 과목명에 대한 정보를 삭제하는 query를 생성하여 SQLiteDBHelper의 executeQuery()를 호출하여 sqliteDB에서 정보를 삭제한다. firebaseDB에서 정보를 삭제하기 위하여 FirebaseDBHelper의 delSubject()를 호출한다.
* -getstartDate(int year, int semester, int startWeekNumber): Date
* 사용자가 입력한 년도, 학기, 시작 요일 정보를 이용하여 1주차 강의가 시작되는 날짜를 구하여 반환한다.
  + 1. **AlarmManagement**

알림 관리 화면에서 사용되는 기능들을 정의한 클래스이다. 알람 목록을 가져오거나 목록에 알림을 추가, 삭제하고 시스템에 알림을 등록, 취소하는 것이 가능하다.

Operation

* +getAlarmList(): List<AlarmInfo>
* SQLiteDBAdapter의 loadAlarmList()를 호출하여 알림 목록을 받아와서 반환한다.
* +addAlarm(str subjectName, str examAlarmDate, str assignmentAlarmDate, str videoLectureAlarmDate, str realTimeLectureAlarmDate): boolean
* 사용자가 입력한 알림 정보들을 조합하여 query를 생성하고, SQLiteDBHelper의 executeQuery()를 호출하여 sqliteDB에 알림 정보를 추가한다. 만약 추가에 실패하면 false를 반환한다. 그 후, setSystemAlarm()을 호출하여 시스템 알림을 추가하고, true를 반환한다.
* +delAlarm(str subjectName): boolean
* 알림목록에서 사용자가 선택한 알림 정보를 사용하여 query를 생성하고, SQLiteDBHelper의 executeQuery()를 호출하여 sqliteDB에서 알림 정보를 삭제한다. delSubjectAlarm()을 호출하여 해당 과목의 시스템 알림을 삭제한다.
* +setSystemAlarm(str subjectName, String exam, String assignment, String video): void
* 강의, 과제, 시험의 알림 정보를 시간단위의 숫자로 변환한다. 과목명을 이용하여 SQLiteDBHelper의 loadLectureDateList(), loadAssignmentDateList(), loadExamDateList()를 호출하여 강의, 과제, 시험 정보중 알림을 설정해야하는 날짜와 이름을 받아온다. 각각의 정보에 대하여 addSystemAlarm()을 호출하여 시스템 알림을 추가한다.
* +addSystemAlarm(str subjectName, str alarmName, str alarmTime, int hourNum, String table): void
* alarmTime과 hourNum을 이용하여 알림이 울려야 할 시간을 구한다. SQLiteDBHelper의 setAlarmNum()을 호출하여 sqliteDB에 알림을 추가하고, 고유번호를 받아온다. 시간과 고유번호를 이용하여 시스템에 알림을 추가한다.
* +delSystemAlarm(int num): void
* 고유번호를 이용하여 시스템에서 알림을 삭제한다.
  + 1. **FriendsManagement**

친구 관리 화면에서 사용되는 기능들을 정의한 클래스이다. 친구 목록을 가져오거나 친구 수락, 삭제가 가능하다.

Operation

* +getFriendsList(): void
* FirebaseDBHelper의 loadFriendsList()를 호출하여 서로친구목록을 firebaseDB에서 불러오도록 한다. 해당 결과는 notifyFriendsList()로 friendList변수를 초기화한다.
* +getFriendsRequestList(): void
* FirebaseDBHelper의 loadFriendsRequestList()를 호출하여 친구신청목록을 firebaseDB에서 불러오도록 한다.
* +requestFriend(str friendID): void
* 사용자에게 입력 받은 friendID에 대하여 firebaseDB에서 친구신청을 진행한다. FirebaseDBHelper의 confirmFriendExit()를 호출하여 해당 friendID가 존재하는지 확인 후 친구 신청을 보낸다. 해당 결과는 showResult()로 메시지를 출력한다.
* +acceptFriend(str friendID): void
* 친구신청목록의 아이디에서 사용자가 선택한 friendID값에 대하여 FirebaseDBHelper의 acceptFriend()를 호출하여 firebaseDB에서 친구신청을 수락한다.
* +delFriend(str friendID): void
* 친구신청목록의 아이디에서 사용자가 선택한 friendID값에 대하여 FirebaseDBHelper의 acceptFriend()를 호출하여 firebaseDB에서 친구를 삭제한다.
* +getFriendToDoList(str friendUID, int date): void
* 사용자가 선택한 친구와 날짜에 대하여 FirebaseDBHelper의 loadFriendToDoList()를 호출하여 firebaseDB에서 친구의 일정을 firebaseDB에서 불러오도록 한다.
  + 1. **SQLiteDB**

SQLiteOpenHelper를 상속받아, DB를 정의하고, DB버전을 관리하는데 사용되는 클래스이다. 어플리케이션의 기능이 정보를 읽거나 저장하는 것의 연속이기 때문에 DB를 singleton패턴으로 작성하였다.

Attribute

* -db: SQLiteDB
* SQLiteDB의 인스턴스 정보를 가진다.

Operation

* -SQLiteDB()
* 생성자이다. Login의 ApplicationContext로 sqliteDB를 생성한다.
* +getInstace()
* db가 null이라면 생성하고, db를 반환한다.
  + 1. **SQLiteDBHelper**

과목, 강의, 과제, 시험, 알림 정보를 sqliteDB에 저장하고, sqlite에 저장된 정보를 가져오기 위한 클래스이다.

Attribute

* -mDbHelper: SQLiteDBHelper
* SQLiteDBHelper의 인스턴스 정보를 가진다.

Operation

* +SQLiteDBHelper()
* 생성자이다. mDbHelper변수를 초기화한다.
* +executeQuery(String sql): boolean
* 입력 받은 쿼리문을 sqliteDB에 실행한다. 해당 쿼리문은 INSERT 또는 DELETE문으로 이루어져 있다. 쿼리문 실행 후 성공 시 true를 반환하고, 실패 시 false를 반환한다.
* +loadSubjectList(str date): List<SubjectInfo>
* sqliteDB에 저장된 과목정보를 불러와 List<SubjectInfo>형태로 가공하여 반환한다.
* +loadLectureList(str date): List<Lecture>
* sqliteDB에 저장된 강의 정보에서 선택된 날짜에 해당하는 강의를 반환하는 메소드이다. 강의의 시작날짜가 사용자가 선택한 날짜보다 더 전이고, 강의의 종료 날짜가 선택 날짜보다 이후인 강의목록들을 가져오는 query문을 작성하여 sqliteDB에서 가져온다. 가져온 정보들을 List<LectureInfo>형태로 가공하여 반환한다.
* +loadExamList(str date): List<ExamInfo>
* sqliteDB에 저장된 시험 정보에서 선택된 날짜에 해당하는 강의를 반환하는 메소드이다. 시험날짜가 입력한 날짜와 같은 시험 정보를 가져오는 query문을 작성하여 sqliteDB에서 가져온다. 가져온 정보들을 List<ExamInfo>형태로 가공하여 반환한다.
* +loadAssignmentList(str date): List<AssignmentInfo>
* sqliteDB에 저장된 과제 정보에서 선택된 날짜에 해당하는 강의를 반환하는 메소드이다. 강의의 시작날짜가 사용자가 선택한 날짜보다 더 전이고, 강의의 종료 날짜가 선택 날짜보다 이후인 과제목록들을 가져오는 query문을 작성하여 sqliteDB에서 가져온다. 가져온 정보들을 List<AssignmentInfo>형태로 가공하여 반환한다.
* +loadAlarmList(): List<AlarmInfo>
* sqliteDB에 저장된 알림 정보 목록을 불러와 List<AlarmInfo>형태로 가공하여 반환한다.
* +loadAlarmInfo(String subjectName): AlarmInfo
* sqliteDB에 저장된 알림 정보 목록에서 과목명이 subjectName에 해당하는 알림 정보를 반환한다.
* +loadLectureDateList(String subjectName): List<String>
* 알림 추가시 사용하는 메소드로, sqliteDB에 저장된 강의 정보에서 과목명이 subjectName에 해당하는 강의들 중, 종료날짜가 현재 날짜보다 뒤인 강의들의 날짜와 이름을 List<Stirng> 형태로 가공하여 반환한다.
* +loadAssignmentDateList(String subjectName): List<String>
* 알림 추가시 사용하는 메소드로, sqliteDB에 저장된 과제 정보에서 과목명이 subjectName에 해당하는 과제들 중, 종료날짜가 현재 날짜보다 뒤인 과제들의 날짜와 이름을 List<Stirng> 형태로 가공하여 반환한다.
* +loadExamDateList(String subjectName): List<String>
* 알림 추가시 사용하는 메소드로, sqliteDB에 저장된 시험 정보에서 과목명이 subjectName에 해당하는 시험들 중, 종료날짜가 현재 날짜보다 뒤인 시험들의 날짜와 이름을 List<Stirng> 형태로 가공하여 반환한다.
* +setAlarmNum(String name, String subjectName): int
* 시스템 알림 추가시 사용하는 메소드로, sqliteDB의 알림 리스트에 알림을 생성하고, loadAlarmNum()을 호출하여 고유번호를 반환한다.
* +loadAlarmNum(String name): int
* sqliteDB의 알림 리스트에 이름이 name인 알림의 고유번호를 반환한다. 만약 해당하는 알림이 없다면 -1을 반환한다.
* +loadAlarmSubjectList(String subjectName): List<Integer>
* sqliteDB의 알림 리스트에 과목명이 subjectName인 알림의 고유번호들을 List<Integer>형태로 가공하여 반환한다.
  + 1. **UploadInfo**

firebaseDB에서 데이터를 받아올 때 class의 형태로 받아올 수 있는데, 이를 위하여 생성한 클래스이다. 또한, firebase에 입력가능한 데이터 형태인 HashMap으로의 변환을 메소드로서 정의하였다.

Attribute

* +startDate: int
* 강의 또는 과제의 시작날짜 정보이다.
* +startTime: int
* 강의 또는 과제의 시작시간 정보이다.
* +endDate: int
* 강의 또는 과제의 종료날짜 정보이다.
* +endTime: int
* 강의 또는 과제의 종료시간 정보이다.
* +isDone: boolean
* 강의 또는 과제의 일정 완료 여부 정보이다.

Operation

* +UploadInfo(): void
* 생성자이다. firebaseDB에서 정보를 클래스 형태로 가져오기 위해 필요하다.
* +UploadInfo(int startDate, int startTime, int endDate, int endTime, boolean isDone): void
* 생성자이다. startDate, startTime, endDate, endTime, isDone변수를 초기화한다.
* +toMap(): HashMap<String, Object>
* 클래스의 변수들의 값을 모두 HashMap에 저장하여 반환한다.
  + 1. **UploadExamInfo**

firebaseDB에서 데이터를 받아올 때 class의 형태로 받아올 수 있는데, 이를 위하여 생성한 클래스이다. 강의와 과제는 데이터 형태가 동일하지만, 과제는 다른 정보들을 저장하기 때문에 강의과제와 분리하여 class를 정의하였다. 또한, firebase에 입력가능한 데이터 형태인 HashMap으로의 변환을 메소드로서 정의하였다.

Attribute

* +date: int
* 시험의 날짜 정보이다.
* +time: int
* 시험의 시간 정보이다.

Operation

* +UploadExamInfo(): void
* 생성자이다. firebaseDB에서 정보를 클래스 형태로 가져오기 위해 필요하다.
* +UploadExamInfo(int date, int time): void
* 생성자이다. date, time 변수를 초기화한다.
* +toMap(): HashMap<String, Object>
* 클래스의 변수들의 값을 모두 HashMap에 저장하여 반환한다.
  + 1. **FirebaseDBHelper**

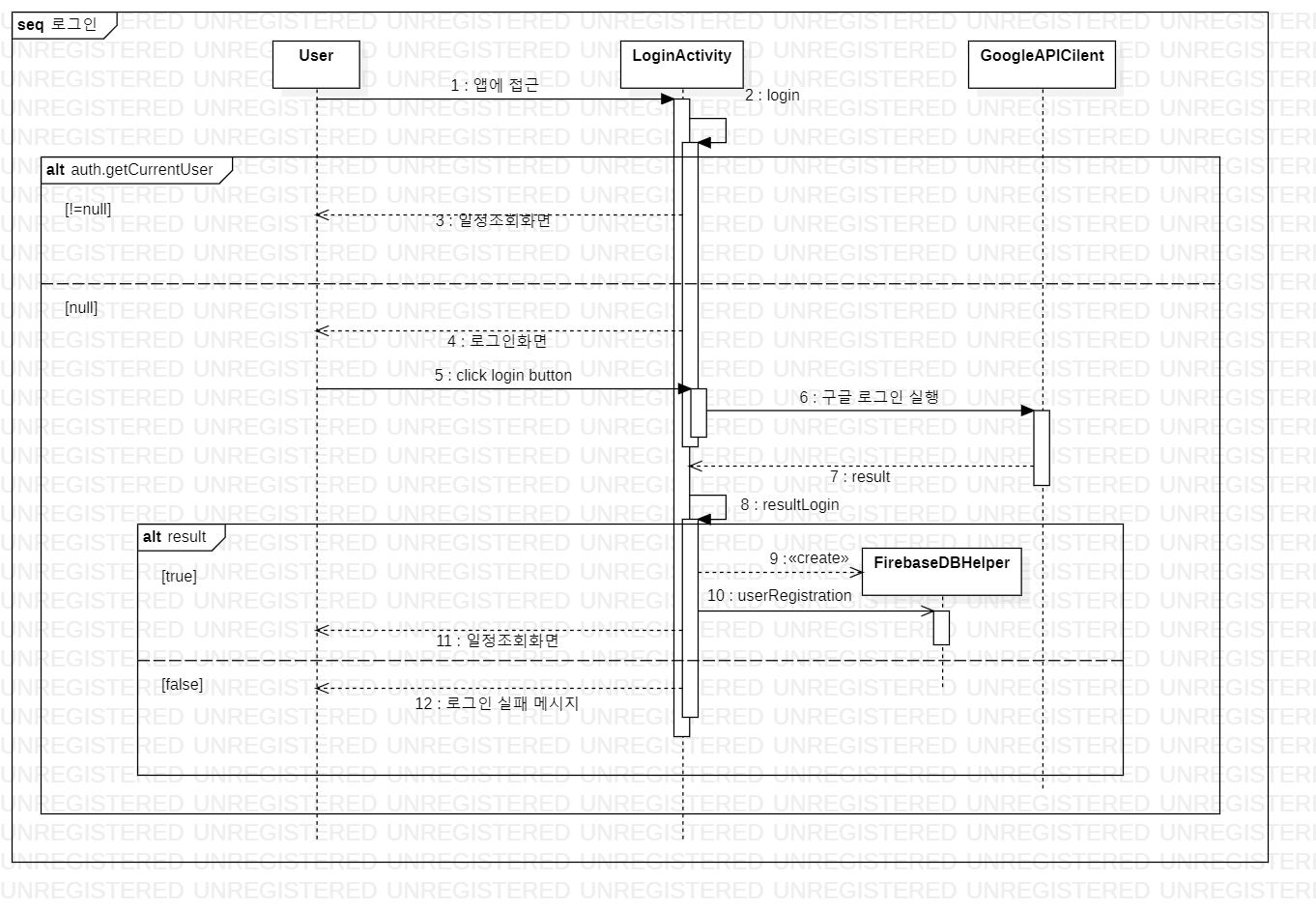
파이어베이스 데이터베이스에 정보를 업데이트하거나, 친구목록을 불러오기, 친구신청, 친구의 일정을 조회할 경우 데이터베이스 서버에 데이터를 전송하거나, 데이터를 받아오는 기능을 하는 클래스이다.

Attribute

* -userUID: String
* 사용자의 UID 정보이다.
* -databaseReference: DatabaseReference
* Firedatabase의 레퍼런스정보이다.

Operation

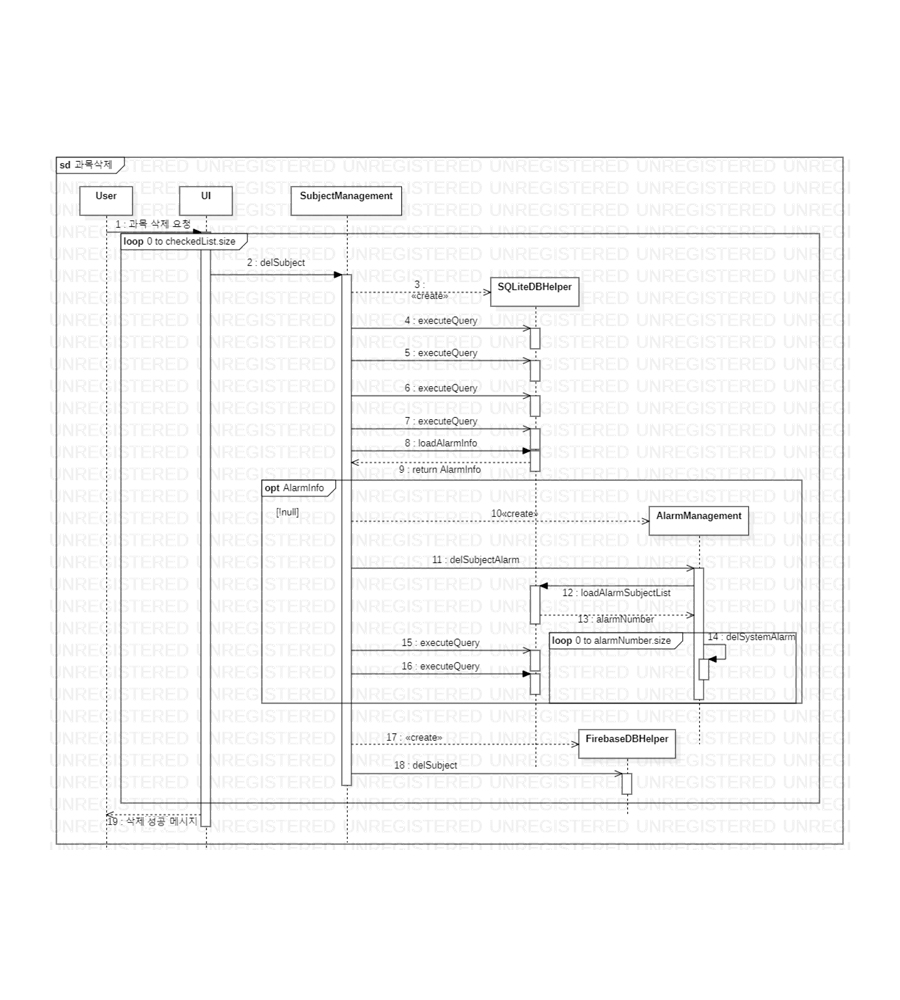
* +FirebaseDBHelper(): void
* 생성자이다. userUID와 databasereference 변수를 초기화한다.
* +userRegistration(): void
* 사용자가 기기에 처음 접속하였을 때 호출되는 메소드이다. firebaseDB의 USERS 레퍼런스에 사용자를 등록한다.
* +confirmFriendExist(str friendID): void
* firebaseDB의 USERS 레퍼런스에 friendID로 저장된 UID를 불러오는 방법으로 친구의 존재여부를 확인한다. 정보가 있다면 해당 UID로 requestFriend()를 호출하여 친구신청을 보낸낸다. 해당 결과는 FriendManagement의 showResult()를 호출하여 결과를 알려준다.
* +loadFriendsRequestList(): void
* firebaseDB의 userUID자식의 friend 테이블의 정보를 가져온다. 그 후 정보들을 확인하여 친구신청 상태인 ID만 List로 가공하여 FriendManagement클래스의 notifyFriendsList()를 호출하여 결과를 반환한다.
* +loadFriendsList(): void
* firebaseDB의 userUID자식의 friend 테이블의 정보를 가져온다. 그 후 정보들을 확인하여 서로친구 상태인 ID 만 List 로 가공하여 FriendManagement클래스의 notifyFriendsList()를 호출하여 결과를 반환한다.
* +requestFriend(str friendUID): void
* 친구신청을 보내는 함수이다. firebaseDB의 friendUID자식의 friend 테이블에 userID-(key:friendUID, value: userUID, key:value, value:0) 을 저장한다.
* +acceptFriend(String friendID, str friendUID): void
* 친구신청을 수락하는 함수이다. firebaseDB의 userUID자식의 friend 테이블에 friendID의 value를 1로 설정하고, firebaseDB의 friendUID자식의 friend 테이블에 userID의 value를 1로 저장한다.
* +loadFriendToDoList(str friendUID, int date): void
* 친구의 todo목록을 불러오는 함수이다. firebaseDB의 friendUID 자식의 모든 정보를 get 해오고, 그 중에서 강의, 과제, 시험 정보를 가공하여 List형태로 만들고, FriendToDo클래스의 notifyFriendToDoList()를 호출하여 list를 반환한다.
* +uploadMyLecture(str subjectName, HashMap lecturelist): void
* 사용자의 강의를 firebaseDB에 저장하는 함수이다. HashMap<강의명, 강의목록>으로 uploadInfo()를 호출하여 firebaseDB에 정보를 저장한다.
* +uploadMyAssignment(str subjectName, str assignmentName, str startdate, str startTime, str enddate, str endTime): void
* 사용자의 과제를 firebaseDB에 저장하는 함수이다. 입력 받은 정보로 UploadInfo를 만들고, toMap()을 호출하여 HashMap형태로 과제정보를 변환한다. 그 후, HashMap<과제명, 과제정보>으로 uploadInfo()를 호출하여 firebaseDB에 정보를 저장한다.
* +uploadMyExam(str subjectName, str examName, str date, str time): void
* 사용자의 시험을 firebaseDB에 저장하는 함수이다. 입력 받은 정보로 UploadExamInfo를 만들고, toMap()을 호출하여 HashMap형태로 시험정보를 변환한다. 그 후, HashMap<시험명, 시험정보>으로 uploadInfo()를 호출하여 firebaseDB에 정보를 저장한다.
* -uploadInfo(HashMap info, str table): void
* table으로 “lecture”, “assignment”, “exam”중 하나의 테이블을 선택하여 firebaseDB의 userUID자식의 table 테이블에 info 정보를 저장한다.
* +delSubject(str subjectName): void
* 사용자가 과목을 삭제할 경우 해당 과목의 강의, 과제, 시험 정보를 모두 삭제한다. firebaseDB의 userUID자식의 lecture, assignment, exam테이블의 과목명자식의 value를 null로 설정한다.
* +delMyAssignment(str assignmentName, str subjectName): void
* 사용자가 과제를 삭제할 경우 해당 과제를 삭제한다. firebaseDB의 userUID 자식의 assignment 테이블의 과목명자식의 과제명key의 value를 null로 설정한다.
* +delMyExam(str examName, str subjectName): void
* 사용자가 시험을 삭제할 경우 해당 시험을 삭제한다. firebaseDB의 userUID 자식의 exam 테이블의 과목명자식의 시험명key의 value를 null로 설정한다.
* +delFriend(str friendID, str friendUID): void
* 사용자가 친구를 삭제할 경우 해당 친구를 firebaseDB에서 삭제한다. firebaseDB의 userUID 자식의 friend 테이블의 friendID의 value를 null로 설정하고, friendUID자식의 friend 테이블의 userID의 value를 null로 설정한다.
* +changeMyIsDone(str name, str subjectName, str tabe, int value): void
* 사용자가 강의 또는 과제의 일정 완료 여부를 바꿀 경우, firebaseDB에서 정보를 업데이트 한다. name에는 강의 또는 과제의 이름을 입력 받고, table으로 “lecture”, “assignment”중 하나의 테이블을 선택하여 firebaseDB의 userUID자식의 table테이블의 subjectName자식의 name자식의 isdone의 value를 입력 받은 value값으로 set한다.
  1. Sequence Diagram
     1. **로그인**



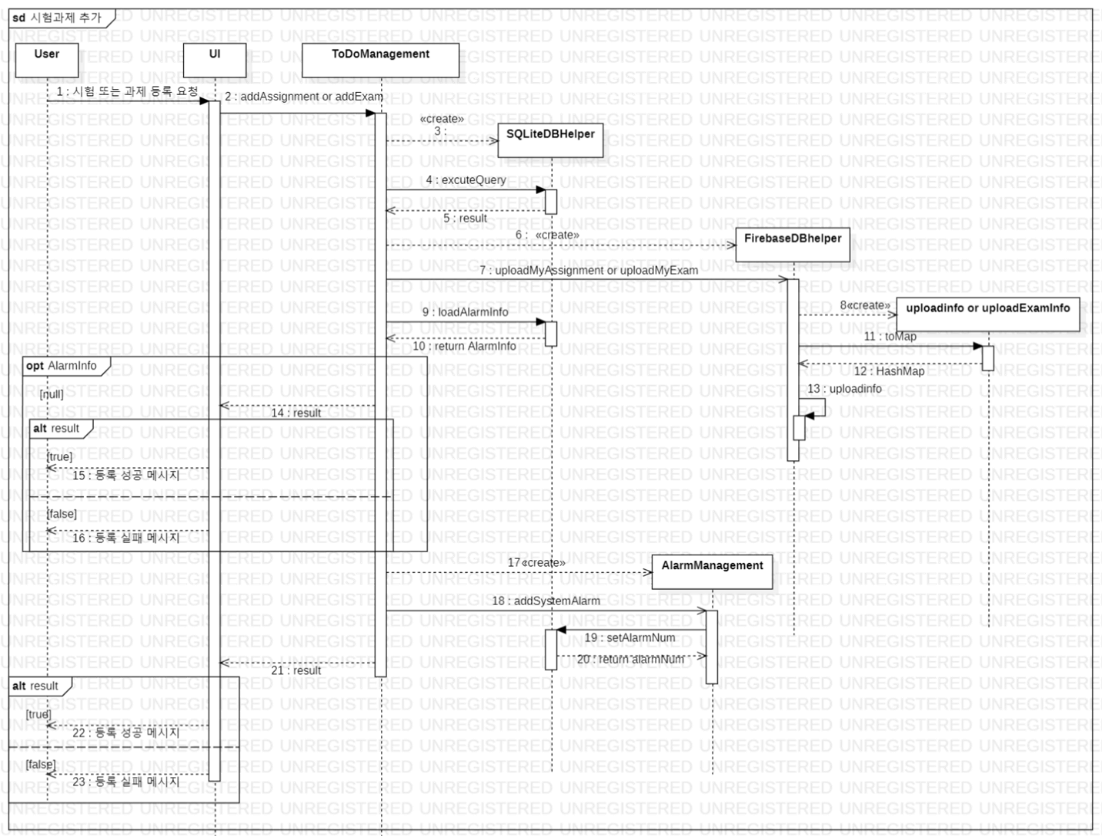
1. 사용자가 시스템에 접근한다.
2. LoginActivity 클래스는 login() 메소드를 호출한다.
3. **만약** auth.getCurrentUser가 null이 아니라면 일정 조회 화면을 보여준다.
4. **만약** auth.getCurrentUser가 null이라면 로그인 화면을 보여준다.
5. 사용자가 로그인 버튼을 클릭한다.
6. GoogleAPIClient를 통해 구글로그인을 실행한다.
7. 로그인의 성공 여부를 반환한다.
8. LoginActivity 클래스는 resultLogin()을 호출한다.
9. **만약** 로그인의 result가 true라면 resultLogin()은 FirebaseDbHelper 객체를 생성한다.
10. resultLogin()은 FirebaseDBHelper의 userRegistration()을 호출한다.
11. 일정 조회 화면으로 이동한다.
12. **만약** 로그인의 result가 false라면 로그인 실패 메시지를 출력한다.
    * 1. **과목 추가**



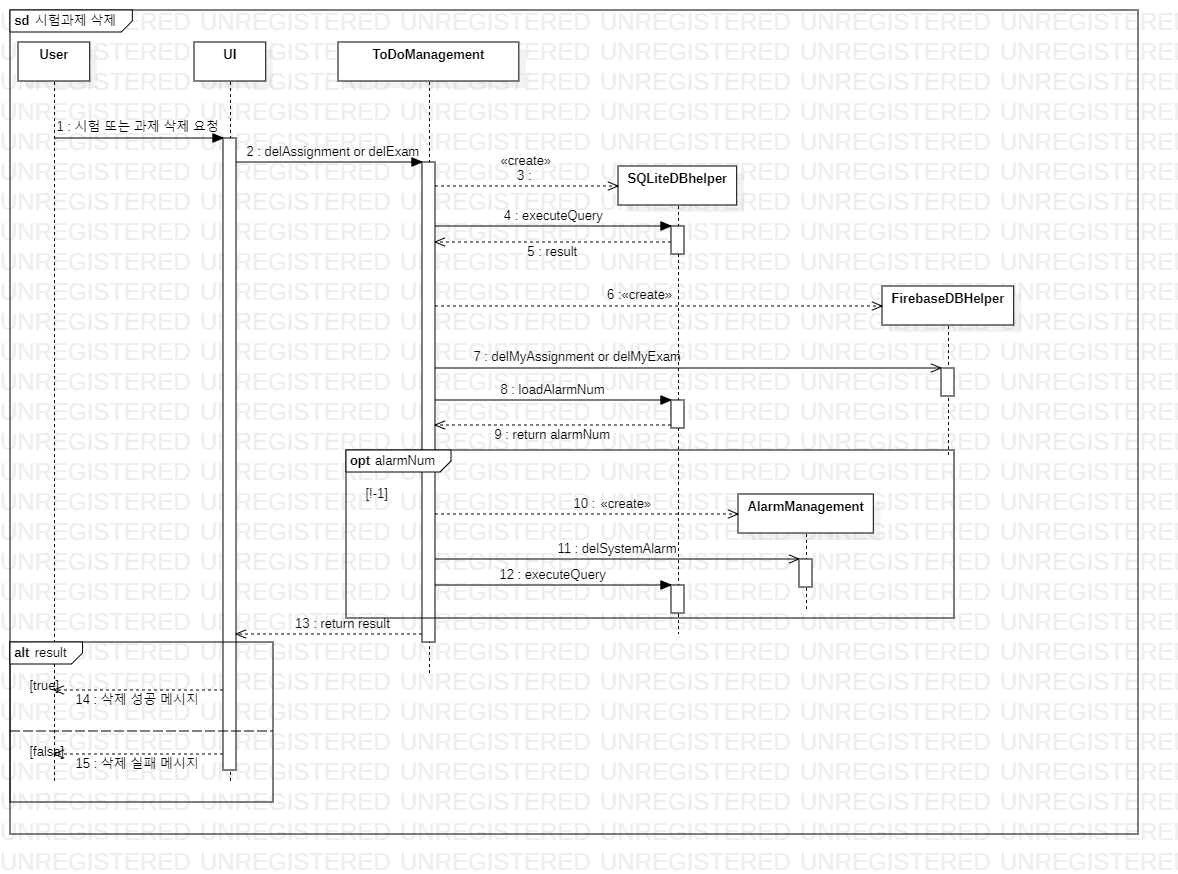
1. 사용자가 과목 추가창에서 확인 버튼을 선택한다.
2. SubjectManagement 클래스의 addSubject()를 호출한다.
3. SubjectManagment의 addSubject()는 SQLiteDBHelper 객체를 생성한다.
4. addSubject()는 쿼리문을 생성하여 SQLiteDBHelper의 executequery()를 호출하고 과목 정보를 sqliteDB에 저장한다.
5. executeQuery()의 과목 정보 저장 성공 결과를 반환한다.
6. addSubejct()는 FirebaseDBHelper 객체를 생성한다.
7. addSubject()는 SubjectManagement의 getstartDate()를 호출하여 강의시작날짜를 계산한다.
8. 1주차부터 16주차까지 사용자가 입력한 한 주의 강의 개수만큼 SQLiteDBHelper 클래스의 executeQuery()를 호출하여 강의를 sqliteDB에 저장한다.
9. executeQuery()에서 강의 정보 저장 성공 결과를 반환한다.
10. FirebaseDBHelper의 uploadMyLecture()를 호출하여 강의들을 firebaseDB에 저장한다.
11. FirebaseDBHelper 클래스에서 uploadInfo()를 호출하여 강의들을 firebaseDB에 저장한다.
12. sqliteDB의 저장 성공 결과를 반환한다.
13. **만약** 결과가 true라면 과목 추가 성공 메시지를 띄운다.
14. **만약** 결과가 false라면 과목 추가 실패 메시지를 띄운다.
    * 1. **과목 삭제**



1. 사용자가 삭제할 과목들을 선택하고, 과목 삭제 버튼을 누른다.
2. SubjectManagement 클래스는 checkedList의 size만큼 delSubject()를 호출한다.
3. delSubject()는 SQLiteDBHelper를 생성한다.
4. delSubject()는 executeQuery()를 호출하여 sqliteDB의 과목 목록에서 해당 과목을 삭제한다.
5. delSubject()는 executeQuery()를 호출하여 sqliteDB의 강의 목록에서 해당 과목을 삭제한다.
6. delSubject()는 executeQuery()를 호출하여 sqliteDB의 과제 목록에서 해당 과목을 삭제한다.
7. delSubject()는 executeQuery()를 호출하여 sqliteDB의 시험 목록에서 해당 과목을 삭제한다.
8. delSubject()는 loadAlarmInfo()를 호출하여 해당 과목에 알림이 등록되어있는지 확인한다.
9. loadAlarmInfo()는 AlarmInfo 결과를 반환한다.
10. **만약** AlarmInfo가 null이 아니라면 delSubject()는 AlarmManagement를 생성한다.
11. delSubject()는 delSubjectAlarm()을 호출한다.
12. delSubjectAlarm()은 SQLiteDBHelper를 생성하고, loadAlarmSubjectList()를 호출한다.
13. loadAlarmSubjectList()는 alarmNumber 결과를 반환한다.
14. delSubjectAlarm()은 alarmNumber의 size만큼 delSystemAlarm()을 호출한다.
15. delSubject()는 executeQuery()를 호출하여 sqliteDB의 알림 정보 목록에서 해당 과목을 삭제한다.
16. delSubject()는 executeQuery()를 호출하여 sqliteDB의 알림 목록에서 해당 과목을 삭제한다.
17. delSubject()는 FirebaseDBHelper를 생성한다.
18. delSubject()는 FirebaseDBHelper의 delSubject()를 호출하여 firebaseDB의 강의, 과제, 시험 정보를 삭제한다.
19. 과목 삭제 성공 메시지를 띄운다.
    * 1. **시험 과제 등록**



1. 사용자가 추가하려는 시험 또는 과제의 정보를 입력하고 등록 버튼을 누른다.
2. 과제를 추가하는 경우 ToDoManagement의 addAssigment()를 호출하고, 시험을 추가하는 경우 ToDoManagement의 addExam()을 호출한다.
3. addAssignment(), addExam()은 SQLiteDBHelper 객체를 생성한다.
4. addAssignment(), addExam()은 입력 받은 정보로 쿼리문을 생성하고 executeQuery()를 호출하여 SQLiteDB에 시험 또는 과제를 등록한다.
5. executeQuery()는 result 결과를 반환한다.
6. addAssignment(), addExam()은 FirebaseDBHelper를 생성한다.
7. addAssignment()는 uploadMyAssignment()를 호출한다. addExam()은 uploadMyExam()을 호출한다.
8. addAssignment()는 uploadInfo를 생성한다. addExam()은 uploadExamInfo를 생성한다.
9. addAssignment(), addExam()은 loadAlarmInfo()를 호출한다.
10. loadAlarmInfo()는 AlarmInfo 결과를 반환한다.
11. uploadMyAssignment(), uploadMyExam()은 toMap()을 호출한다.
12. toMap()은 정보를 HashMap형태로 반환한다.
13. uploadMyAssignment(), uploadMyExam()은 uploadInfo()을 호출하여 firebaseDB에 저장한다.
14. 만약 10번의 결과가 null이라면 5번의 결과를 반환한다.
15. **만약** result가 true면 시험, 과제 등록 성공 메시지를 띄운다.
16. **만약** result가 false면 시험, 과제 등록 실패 메시지를 띄운다.
17. addAssignment(), addExam()은 AlarmManagement를 생성한다.
18. addAssignment(), addExam()은 addSystemAlarm()을 호출한다.
19. addSystemAlarm()은 SQLiteDBHelper를 생성하고 setAlarmNum()을 호출한다.
20. setAlarmNum()은 알림 고유번호를 반환한다.
21. addAssignment(), addExam()은 5번의 결과를 반환한다.
22. **만약** result가 true면 시험, 과제 등록 성공 메시지를 띄운다.
23. **만약** result가 false면 시험, 과제 등록 실패 메시지를 띄운다.
    * 1. **시험 과제 삭제**



1. 사용자가 시험, 과제 삭제 버튼을 누른다.
2. 과제를 삭제하는 경우 ToDoManagement의 delAssigment()를 호출하고, 시험을 삭제하는 경우 ToDoManagement의 delExam()을 호출한다.
3. delAssigment(), delExam()은 SQLiteDBHelper객체를 생성한다.
4. delAssigment(), delExam()은 쿼리문을 생성하고 executeQuery()를 호출하여, SQLiteDB에서 시험 목록 또는 과제 목록에서 정보를 삭제한다.
5. executeQuery()의 result 결과를 반환한다.
6. delAssignment(), delExam()은 FirebaseDBHelper객체를 생성한다
7. delAssignment()는 delMyAssignment()를 호출한다. delExam()은 delMyExam()을 호출한다.
8. delAssigment(), delExam()은 loadAlarmNum()을 호출하여 해당 과제, 시험에 알림이 설정되어 있는 지 확인한다.
9. loadAlarmNum()은 alarmNum()결과를 반환한다.
10. **만약** alarmNum이 -1이 아니라면 delAssigment(), delExam()은 AlarmManagement를 생성한다.
11. delAssigment(), delExam()은 delSystemAlarm()을 호출하여 시스템 알림을 삭제한다.
12. delAssigment(), delExam()은 executeQuery()를 호출하여 알림정보에서 시험 또는 과제를 삭제한다.
13. 5번의 result 결과를 반환한다.
14. **만약** result가 true라면 삭제 성공 메시지를 출력한다.
15. **만약** result가 false라면 삭제 성공 메시지를 출력한다.
    * 1. **수업 일정 조회**



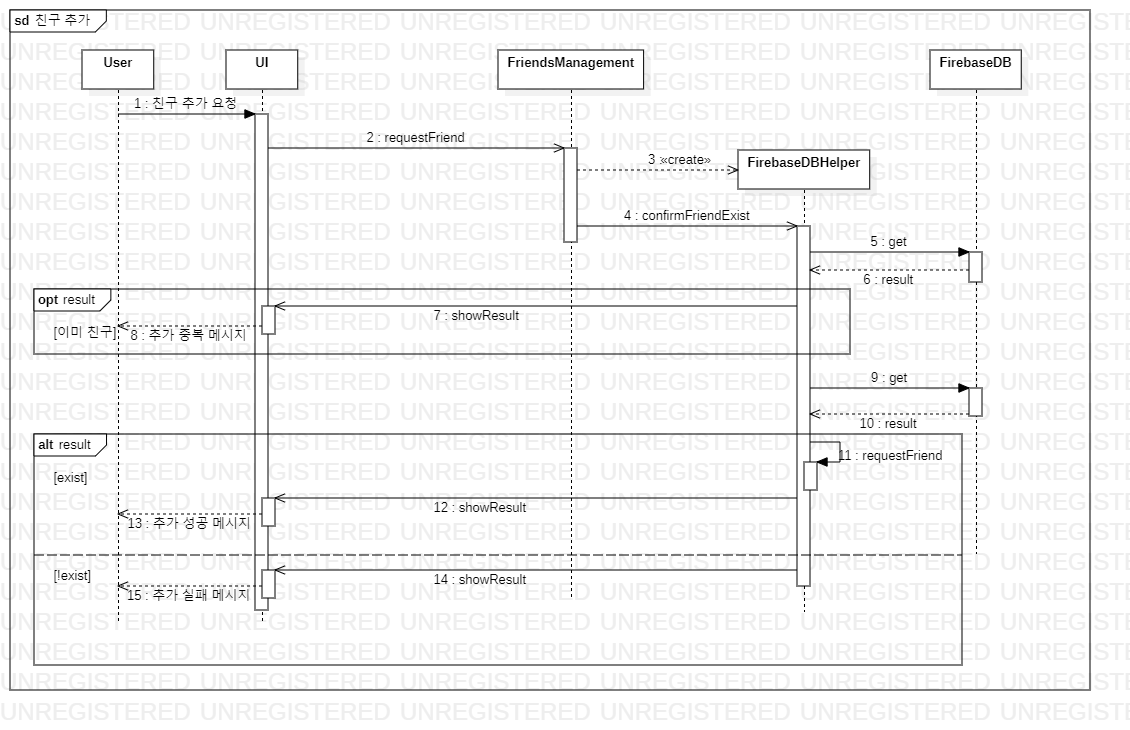
1. 사용자가 날짜를 선택한다.
2. ToDoManageMent클래스의 getToDoList()를 호출한다.
3. getToDoList()는 SQLiteDBHelper 객체를 생성한다.
4. getToDoList()는 SQLiteDBHelper의 loadLecureList()를 호출한다.
5. lectureList를 반환한다.
6. getToDoList()는 SQLiteDBHelper의 loadAssignmentList()를 호출한다.
7. assignmentList를 반환한다.
8. getToDoList()는 SQLiteDBHelper의 loadExamList()를 호출한다.
9. examList를 반환한다.
10. lectureList, assignmentList, examList를 가공하여 ToDoList를 생성하여 반환한다.
11. 수업일정조회화면을 출력한다.
    * 1. **친구 목록 조회**



1. 사용자가 친구 목록 조회 버튼을 누른다.
2. FriendManagement클래스의 getFriendsList()를 호출한다.
3. getFriendsList()는 FirebaseDBHelper 객체를 생성한다.
4. getFriendsList()는 loadFriendsList()를 호출한다
5. FirebaseDBHelper는 FirebaseDB에 정보를 요청한다.
6. FirebaseDB는 해당 결과를 반환한다.
7. FirebaseDBHelper는 notify를 이용하여 화면에 친구 목록을 업데이트 하였다고 알린다.
8. 친구 목록을 출력한다.
   * 1. **친구 수업 일정 조회**



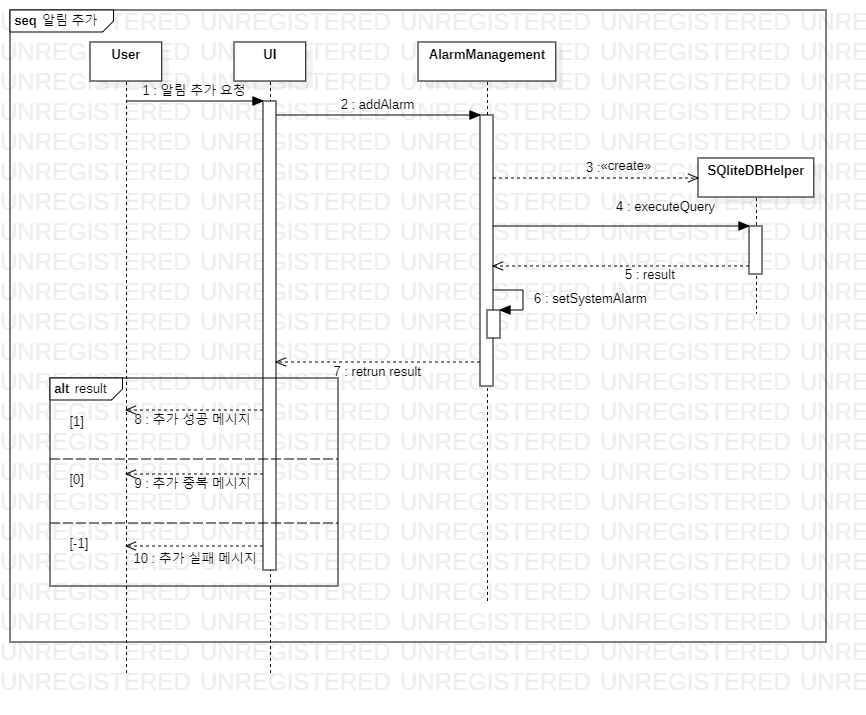
1. 사용자가 친구목록에서 친구를 선택한다.
2. FreindManagement 클래스의 getFreindToDoList()를 호출한다.
3. getFriendToDoList()는 FirebaseDBHelper객체를 생성한다.
4. getFriendToDoList()는 FirebaseDBHelper의 loadFriendsToDoList()를 호출한다.
5. FirebaseDBHelper는 FirebaseDB에 정보를 요청한다.
6. FirebaseDB는 해당 결과를 반환한다.
7. FirebaseDBHelper는 notifyFriendToDoList를 이용하여 화면에 친구 일정을 업데이트 하였다고 알린다.
8. 친구 일정 조회 화면을 출력한다.
   * 1. **친구 추가**



1. 사용자가 친구 추가를 할 id를 입력하고, 친구 추가 버튼을 누른다.
2. FriendManagement클래스의 requestFriend()를 호출한다.
3. FriendManagement는 FirebaseDBHelper객체를 생성한다.
4. FriendManagement는 FirebaseDBHelper에게 confirmFriendExist()를 호출한다.
5. FirebaseDBHelper는 FirebaseDB에게 이미 친구인지 확인을 요청한다.
6. FirebaseDB는 result 결과를 FirebaseDBHelper에게 반환한다.
7. **만약** id가 이미 친구인 사용자라면 confirmFriendExist()는 showResult를 이용하여
8. 추가 중복 메시지를 출력한다.
9. FirebaseDBHelper는 FirebaseDB에게 존재하는 사용자인지 확인을 요청한다.
10. FirebaseDB는 result 결과를 FirebaseDBHelper에게 반환한다.
11. **만약** id가 존재하는 사용자라면 FirebaseDBHelper는 requestFriend()메서드를 호출한다.
12. confirmFriendExist()는 showResult를 이용하여
13. 친구 신청 성공 메시지를 출력한다.
14. **만약** id가 존재하지 않는 사용자라면 FriendManagement의 showResult를 이용하여
15. 친구 신청 실패 메시지를 출력한다.
    * 1. **친구 삭제**



1. 사용자가 삭제할 친구를 선택하고, 친구 삭제 버튼을 누른다.
2. FriendManagement클래스의 delFriend()를 호출한다.
3. checkedList의size만큼 delFriend()는 FirebaseDBHelper객체를 생성한다.
4. delFriend()를 호출하여 firebaseDB에서 친구 정보를 삭제한다.
5. 친구 삭제 성공 메시지를 출력한다.
   * 1. **알림 추가**



1. 사용자가 추가할 알림의 정보를 입력하고, 알림 추가 버튼을 누른다.
2. AddAlarm 클래스는 addAlarm()을 호출한다.
3. addAlarm()은 SQLiteDBHelper객체를 생성한다.
4. addAlarm()은 입력 받은 정보로 query를 작성하고, executeQuery()를 호출한다.
5. executeQuery()는 추가 결과를 반환한다.
6. addAlarm()은 setSystemAlarm()을 호출하여 시스템에 알림을 설정한다.
7. addAlarm()은 5번의 결과를 반환한다.
8. **만약** 결과가 1이라면 알림 추가 성공 메시지를 출력한다.
9. **만약** 결과가 0이라면 알림 추가 중복 메시지를 출력한다.
10. **만약** 결과가 -1이라면 알림 추가 실패 메시지를 출력한다.
    * 1. **알림 삭제**



1. 사용자가 알림을 삭제할 과목을 선택하고, 알림 삭제 버튼을 누른다.
2. AlarmManagement클래스의 delAlarm()를 호출한다.
3. delAlarm()은 SQLliteHelper객체를 생성한다.
4. checkedList의 size만큼 delAlarm()은 입력 받은 정보로 쿼리문을 생성하고, SQLliteHelper의 executeQuery()를 호출하여 sqliteDB에서 알림 정보를 삭제한다.
5. executeQuery()는 삭제 결과를 반환한다.
6. delAlarm()은 delSubjectAlarm()을 호출하여 시스템에 설정된 알림을 삭제한다.
7. delSubjectAlarm()은 delSystemAlarm()을 호출하여 과목의 모든 시스템 알림을 삭제한다.
8. 알람 삭제 성공 메시지를 출력한다.
9. **프로젝트 계획**
   1. 프로젝트 일정표

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 순서 | 기간  작업내용 | 10월 | 11월 | | | | 12월 | |
| 4주 | 1주 | 2주 | 3주 | 4주 | 1주 | 2주 |
| 1 | 설계서 작성(구조도) | O |  |  |  |  |  |  |
| 2 | 설계서 작성(use case) | O |  |  |  |  |  |  |
| 3 | 설계서 작성(activity) | O |  |  |  |  |  |  |
| 4 | 설계서 작성(state) | O | O |  |  |  |  |  |
| 5 | 설계서 작성(class) |  | O | O |  |  |  |  |
| 6 | 설계서 작성(sequence) |  | O | O | O |  |  |  |
| 7 | 설계서 검토 |  |  | O | O |  |  |  |
| 8 | 코드작성(sqlite) |  |  | O | O |  |  |  |
| 9 | 코드작성(firebase) |  |  | O | O | O |  |  |
| 10 | 코드작성(과목) |  |  | O | O |  |  |  |
| 11 | 코드작성(조회) |  |  | O | O | O |  |  |
| 12 | 코드작성(과제시험) |  |  | O | O |  |  |  |
| 13 | 코드작성(알림) |  |  | O | O | O |  |  |
| 14 | 코드작성(친구) |  |  | O | O | O |  |  |
| 15 | 테스트 |  |  |  | O | O | O |  |
| 16 | 결과 보고서 작성 |  |  |  |  | O | O | O |

* 1. 회의 기록

|  |  |
| --- | --- |
| 일자 | 내용 |
| 2021/10/31  (zoom) | -설계서 큰 틀 작성  -use case, activity diagram 작성  -architecture구조도 작성  -state diagram 작성 |
| 2021/11/02  (zoom) | -class diagram 작성  -database 결정 |
| 2021/11/05  (zoom) | -구현 파트 분배 |
| 2021/11/10  (zoom) | -구현 진행상황 공유  -class, sequence diagram 파트분배 |
| 2021/11/13  (zoom) | -class, sequence diagram 검토 및 작성  -activity diagram 수정 |
| 2021/11/14  (zoom) | -sequence diagram 설명 작성  -class diagram 수정  -state diagram 수정 |
| 2021/11/15  (zoom) | -명세서와 싱크 맞추기 |