소프트웨어 테스트결과 보고서

작성자: 8조

20193098 최영은

20193051 김수진

20173127 안상욱

20113310 이희원

[1. 목적 및 범위 3](#_Toc89799310)

[2. 테스트 시나리오 3](#_Toc89799311)

[2.1. 로그인 4](#_Toc89799314)

[2.2. 과목 추가 4](#_Toc89799315)

[3. 테스트 및 개발 환경 4](#_Toc89799316)

[4. Unit 테스트 시나리오 4](#_Toc89799317)

[4.1. SQLiteDBHelper 4](#_Toc89799322)

[4.2. SubjectManagement 6](#_Toc89799323)

[4.3. ToDoManagement 9](#_Toc89799324)

[4.4. AlarmManagement 10](#_Toc89799325)

[5. Unit 테스트 결과 11](#_Toc89799326)

[5.1. SQLiteDBHelper 11](#_Toc89799332)

[5.2. SubjectManagement 12](#_Toc89799333)

[5.3. ToDoManagement 12](#_Toc89799334)

[5.4. AlarmManagement 13](#_Toc89799335)

[6. 시스템 테스트 시나리오 13](#_Toc89799336)

[6.1 과목추가 삭제 13](#_Toc89799343)

[6.2 시험 과제추가 삭제 13](#_Toc89799344)

[6.3 수업일정 조회 13](#_Toc89799345)

[6.4 친구목록 조회 14](#_Toc89799346)

[6.5 친구 수업일정 조회 14](#_Toc89799347)

[6.6 친구 추가 삭제 14](#_Toc89799348)

[6.7 알림 추가 삭제 14](#_Toc89799349)

[7. 시스템 테스트 결과 15](#_Toc89799350)

[7.1. 과목 추가 삭제 15](#_Toc89799358)

[7.2. 시험 과제추가, 삭제 17](#_Toc89799359)

[7.3. 수업일정 조회 19](#_Toc89799360)

[7.4. 친구목록 조회 20](#_Toc89799361)

[7.5. 친구 수업일정 조회 21](#_Toc89799362)

[7.6. 친구 추가 삭제 22](#_Toc89799363)

[7.7. 알림 추가 삭제 24](#_Toc89799364)

[8. 비기능 요구사항 테스트 시나리오 26](#_Toc89799365)

[8.1. 품질 요구사항 26](#_Toc89799374)

[9. 비기능 요구사항 테스트 결과 27](#_Toc89799375)

[9.1. 품질 요구사항 27](#_Toc89799385)

[10. 결함 및 이슈 보고 30](#_Toc89799386)

[10.1. 알림 관련 이슈 30](#_Toc89799397)

[10.2. 회의기록 31](#_Toc89799398)

# 목적 및 범위

본 문서는 소프트웨어공학 과제로 개발한 소프트웨어를 기능 단위로 테스트하도록 계획한다. 테스트를 진행한 뒤 결과를 작성한 문서로서 남아있는 결함을 발견하고 명세를 충족하였는지를 확인하기 위함을 목적으로 한다. 테스트의 범위는 Unit 테스트와 시스템 테스트, 품질 요구사항 테스트까지다. 각 테스트에 대한 시나리오와 결과를 기본으로 본 문서를 서술하였고 문서의 맨 마지막에는 끝내 해결하지 못한 결함과 이슈에 대하여 기술하였다.

# 테스트 시나리오

각각의 기능에 대하여 다음과 같은 시나리오로 테스트를 진행하고자 한다.



## **로그인**

로그인은 이미 검증된 소프트웨어인 구글 로그인 API를 사용하였으므로 테스트하지 않는다.

## **과목 추가**

과목 추가에서는 사용자에게서 과목명, 생성 년도, 학기, 강의 개수, 생성 요일, 생성시간, 종료 요일, 종료시간을 입력 받고 데이터베이스에 과목정보와 강의정보를 저장해야 한다.

이는 SubjectManagement 클래스와 SQLiteDBHelper 클래스, FirebaseDBHelepr 클래스가 사용된다.

따라서 Unit 테스트에서 SQLiteDBHelper의 executeQuery()가 실행이 잘 되는지, 과목목록 및 강의 목록에 저장을 하는지 확인하고, FirebaseDBHeleper의 uploadMyLecture()가 FirebaseDB에 저장을 하는지 확인하고, SubjectManagement의 addSubject()의 인자로 올바르지 않은 값을 넣은 경우, 에러를 반환하는지, 올바른 값을 넣은 경우 에러가 나지 않고 잘 동작하는지 확인한다.

시스템 테스트에서는 과목 관리 화면에서 과목추가 버튼을 누르고 화면에 정보를 입력한 후 확인 버튼을 누르면 과목 추가 성공 메시지가 뜨는지, 추가한 과목이 과목 관리 화면의 목록에 제대로 추가되었는지를 확인한다.

* 1. **과목 삭제**

과목 삭제에서는 사용자가 선택한 과목명에 대한 과목정보, 강의정보, 알림 정보를 모두 데이터베이스에서 삭제해야 한다.

이는 SubjectManagement 클래스와 AlarmManagement 클래스, SQLiteDBHelper 클래스, FirebaseDBHelepr 클래스가 사용된다.

따라서 Unit 테스트에서 SQLiteDBHelper의 executeQuery()가 실행이 잘 되는지, 과목목록에서 해당 과목이 제거되었는지 확인해야 한다. 과목에 알림이 설정되어 있을 경우를 위해 loadAlarmInfo()가 과목에 설정된 알림 정보를 반환하는지, AlarmManagement의 delSubjectAlarm()이 제대로 동작하는지 확인해야 한다. 그리고 FirebaseDBHelper의 delSubject()가 과목을 FirebaseDB에서 삭제하는지 확인해야 한다.

시스템 테스트에서는 과목 관리 화면에서 과목삭제 버튼을 누르고 삭제할 과목을 선택한 후 삭제 버튼을 누르면 과목 삭제 성공 메시지가 뜨는지, 과목 관리 화면의 목록에서 해당 과목이 제거되는지 확인한다.

* 1. **시험과제 등록**

시험 등록은 사용자가 과목명을 선택하고, 시험명, 날짜, 시간을 입력한다. 과제 등록은 사용자가 과목명을 선택하고, 과제명, 시작날짜, 시작시간, 종료날짜, 종료시간을 입력한다. 이후 시험 및 과제 정보를 데이터베이스에 저장해야 한다. 만약 해당 시험 또는 과제에 알림이 설정되어 있다면, 데이터베이스에 알림을 저장하고, 시스템 알림을 추가해야 한다.

이는 ToDoManagement 클래스와 AlarmManagement 클래스, SQLiteDBHelper 클래스, FirebaseDBHelper 클래스가 사용된다.

따라서 Unit 테스트에서 SQLiteDBHelper의 executeQuery()가 실행이 잘 되는지, 시험목록 또는 과제 목록에 저장을 하는지 확인하고, loadAlarmInfo()가 실행이 되는지, 해당 과목의 알림 정보를 올바르게 가져오는지 확인한다. FirebaseDBHelper의 uploadMyAssignment(), uploadMyExam()이 FirebaseDB에 저장을 하는지 확인하고, AlarmManagement 클래스의 addSystemAlarm()이 제대로 동작하는지 확인한다. 그리고 ToDoManagement의 addAssignment(), addExam()의 인자로 올바르지 않은 값을 넣은 경우 에러를 반환하는지, 올바른 값을 넣은 경우 에러가 나지 않고 잘 동작하는지 확인한다.

시스템 테스트에서는 수업 일정 조회 화면에서 추가 버튼을 누르고 화면에 정보를 입력한 후 확인 버튼을 누르면 시험과제 추가 성공 메시지가 뜨는지, 일정 조회 화면 하단의 과목목록에 시험 또는 과제가 추가되는지 확인한다.

* 1. **시험과제 삭제**

시험과제 삭제에서는 사용자가 선택한 시험 또는 과제에 대한 정보를 데이터베이스에서 삭제하고, 그에 해당하는 알림 정보를 시스템에서 제거해야 한다.

이는 ToDoManagement 클래스, AlarmManagement 클래스, SQLiteDBHelper 클래스, FirebaseDBHelper 클래스가 사용된다.

따라서 Unit 테스트에서 SQLiteDBHelper의 executeQuery()가 실행이 되는지, 시험목록 또는 과제목록에서 정보가 삭제가 되었는지 확인하고, 과목에 알림이 설정되어 있을 경우를 위해 loadAlarmNum()이 과제 또는 시험에 존재하는 알림 번호를 반환하는지, AlarmManagement의 delSystemAlarm()이 제대로 동작하는지 확인해야 한다. 그리고 FirebaseDBHelper의 delMyAssignment 또는 delMyExam()이 시험 또는 과제를 FirebaseDB에서 삭제하는지 확인해야 한다.

시스템 테스트에서는 수업 일정 조회 화면에서 시험과제 삭제 버튼을 누르면 과목목록에서 시험과제 삭제 성공 메시지가 뜨는지, 시험 또는 과제가 제거되는지 확인한다.

* 1. **수업일정 조회**

수업일정 조회는 사용자가 날짜를 선택하면 해당 날짜에 해당하는 SQLiteDB의 강의, 과제, 시험 정보들을 알맞게 가져와져야 한다.

이는 ToDoManagement 클래스와 SQLiteDBHelper 클래스가 사용된다.

따라서 Unit 테스트에서 SQLiteDBHelper의 loadLectureList(), loadAssignmentList(), loadExamList()가 해당하는 정보들을 반환하는지 확인하고, ToDoManagement의 getToDoList()가 해당 정보들을 잘 가공하여 반환하는지 확인한다.

시스템 테스트에서는 수업 일정 조회 화면에 접속하면 오늘 날짜에 해당하는 할 일 목록이 올바르게 표시되는지 확인하고, 다른 날짜를 선택하면 해당 날짜의 할 일 목록이 올바르게 표시되는지 확인한다.

* 1. **친구목록 조회**

친구목록 조회는 사용자가 일정 조회 화면에 있는 친구 관리 버튼을 선택하면 FirebaseDB에서 사용자의 서로 친구 목록 또는 받은 친구 신청 목록이 가져와져야 한다.

이는 FriendsManagement 클래스와 FirebaseDBHelper 클래스가 사용된다.

시스템 테스트에서는 친구목록 조회 화면에 접속하거나 서로친구목록 버튼을 선택하면 사용자의 서로 친구 목록이 올바르게 표시되는지 확인하고, 받은 친구신청 버튼을 선택하면 받은 친구 신청 목록이 올바르게 표시되는지 확인해야 한다.

* 1. **친구 수업일정 조회**

친구 수업일정 조회는 사용자가 친구와 날짜를 선택하면 해당 날짜에 해당하는 FirebaseDB의 친구강의, 친구 과제, 친구 시험 정보들을 알맞게 가져와져야 한다.

이는 ToDoManagement 클래스와 FirebaseDB 클래스가 사용된다.

시스템 테스트에서 친구 관리 화면의 서로친구목록에서 친구를 선택하면 해당 친구의 오늘 날짜의 할 일 목록이 화면에 올바르게 표시되는지 확인하고, 다른 날짜를 선택하면 해당 날짜의 친구의 할 일 목록이 올바르게 표시되는지 확인한다.

* 1. **친구 추가**

친구 추가는 사용자가 친구 ID를 입력하고 확인 버튼을 누르면 해당 ID가 존재하고, 현재 서로친구 상태가 아니라면 데이터베이스에 친구신청을 저장해야 한다.

이는 FirendsManagement 클래스와 FirebaseDBHelper 클래스가 사용된다.

시스템 테스트에서 친구 관리 화면에서 친구 추가 버튼을 선택하고 존재하는 사용자에게 친구 신청을 보내면 친구 신청 완료 메시지가 뜨는지 확인한다.

* 1. **친구 삭제**

친구 삭제는 사용자가 선택한 친구에 대한 정보를 데이터베이스에서 삭제해야 한다.

이는 FriendsManagement 클래스와 FirebaseDBHelper 클래스가 사용된다.

시스템 테스트에서는 친구 관리 화면에서 친구삭제 버튼을 선택하고 사용자가 삭제할 친구를 선택한 후 삭제 버튼을 누르면 친구 삭제 완료 메시지가 뜨는지, 친구 관리 화면의 목록에서 해당 친구가 제거되는지 확인한다.

* 1. **알림 추가**

알림 추가는 사용자가 선택한 과목에 대하여 입력한 정보로 알림을 데이터베이스에 저장하고, 시스템 알림을 생성해야 한다.

이는 AlarmManagement 클래스와 SQLiteDBHelper 클래스가 사용된다.

따라서 Unit 테스트에서 SQLiteDBHelper의 executeQuery()가 실행이 되는지, 알림 정보 목록과 알림 목록에 알림정보가 추가되었는지를 확인하고, loadLectureDateList가 해당 과목의 강의목록들을 불러오는지, setAlarmNum()이 해당 알람의 고유번호를 설정하고 반환하는지, 해당 고유번호가 중복여부를 확인해야한다.

시스템 테스트에서는 알림 관리 화면에서 알림 추가 버튼을 누르고 사용자가 알림 추가 창에서 정보들을 입력한 후 확인 버튼을 눌렀을 때 알림 추가 성공 메시지가 뜨는지, 알림 관리 화면의 목록에 해당 알림이 추가되는지 확인한다.

시스템 알림 추가는 이미 검증된 Android AlarmManager를 사용하였기 때문에 테스트하지 않는다.

* 1. **알림 삭제**

알림 삭제는 사용자가 선택한 과목에 대한 알림 정보를 데이터베이스에서 삭제해야 하고, 생성된 시스템 알림들을 삭제해야 한다.

이는 AlarmManagement 클래스와 SQLiteDBHelper 클래스가 사용된다.

따라서 Unit 테스트에서 SQLiteDBHelper의 executeQuery()가 실행이 되는지, 알림 정보 목록과 알림 목록에서 해당 과목의 정보들이 삭제되었는지 확인하고, loadAlarmSubjectList()가 알림 목록에 있는 해당 과목들을 모두 불러오는지 확인한다.

시스템 테스트에서는 알림 관리 화면에서 알림 삭제 버튼을 누르고 과목을 선택한 뒤 삭제 버튼을 눌렀을 때 알림 삭제 성공 메시지가 뜨는지, 알림 관리 화면의 목록에서 해당 과목이 제거되는지 확인한다.

시스템 알림 삭제는 이미 검증된 Android AlarmManager를 사용하였기 때문에 테스트하지 않는다.

# 테스트 및 개발 환경

|  |  |
| --- | --- |
| 이름 | 환경 |
| 개발 환경 | Android Studio Arctic Fox | 2020.3.1 Patch 3 |
| Unit 테스트 | Android Studio – androidTest - Junit |
| 시스템 테스트 | Android Studio – androidTest – Junit, esspresso |
| 품질 요구사항 테스트 | Android Studio JunitTest |

Unit 테스트는 androidTest가 아닌 test에서 파일을 만들어 테스트를 진행해야 하지만, SQLiteDB에서 ApplicationContext를 필요로 하기 때문에 androidTest에서 파일을 만들어 진행하도록 하였다.

# Unit 테스트 시나리오



## **SQLiteDBHelper**

SQLiteDBHelper클래스의 각각의 함수에 다음과 같은 인자로 테스트를 진행한다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 번호 | case | method | parameter | test |
| 0\_1 | 과목 추가 | Execute  Query | INSERT INTO SubjectList VALUES('test1', 1, 2, 0000, 2, 0000); | return true 인가 |
| 0\_2 | 과목 조회 | Load  SubjectList |  | 1에서 추가한 과목이 DB에 저장되었는가 |
| 0\_3 | 동일한  과목 추가 | Execute  Query | INSERT INTO SubjectList VALUES('test1', 1, 2, 0000, 2, 0000); | return false 인가 |
| 0\_4 | 과목 삭제 | executeQuery | DELETE FROM SubjectList WHERE subjectName = 'test1'; | return true 인가 |
| 1\_1 | 강의 추가 | executeQuery | INSERT INTO LectureList VALUES('test1','lname',20210901,0000,20211231,0000,0); | return true 인가 |
| 1\_2 | 강의 목록 조회 | Load  LectureList | 20210902 | 5에서 추가한 강의가 DB에 저장되었는가 |
| 1\_3 | 빈 강의 목록 조회 | Load  LectureList | 20210801 | 비어 있는 강의 목록이 반환되는가 |
| 1\_4 | 강의 날짜리스트  가져오기 | Load  Lecture  DateList | test1 | 가져온 날짜 정보가  5에서 추가한 강의 종료날짜와 같은가 |
| 1\_5 | 강의 삭제 | executeQuery | DELETE FROM LectureList WHERE subjectName = 'test1'; | return ture인가 |
| 1\_6 | 빈 강의 목록 조회 | Load  LectureList | 20210902 | 9에서 삭제한 강의가 DB에서 삭제되었는가 |
| 2\_1 | 과제 추가 | Execute  Query | INSERT INTO AssignmentList VALUES('test1','aname',20210901,0000,20211231,0000,0); | return true인가 |
| 2\_2 | 과제 목록 조회 | Load  AssignmentList | 20210902 | 11에서 추가한 과제가 DB에 저장 되어있는가 |
| 2\_3 | 빈 과제 목록 조회 | Load  AssignmentList | 20210801 | 비어있는 과제 목록이 반환되는가 |
| 2\_4 | 과제날짜리스트  가져오기 | Load  Assignment  DateList | test1 | 가져온 날짜 정보가  7에서 추가한 과제의  정보와 같은가 |
| 2\_5 | 과제 삭제 | Execute  Query | DELETE FROM AssignmentList WHERE subjectName = 'test1'; | return true인가 |
| 2\_6 | 삭제 후 과제 목록 조회 | Load  AssignmentList | 20210902 | 15에서 삭제한 과제가 DB에서 삭제 되었는가 |
| 3\_1 | 시험 추가 | Execute  Query | "INSERT INTO ExamList VALUES('test1', 'ename', 20211231, 0900);" | retrun true인가 |
| 3\_2 | 시험 목록 조회 | Load  ExamList | 20211231 | 17에서 추가한 시험이 DB에 저장 되었는가 |
| 3\_3 | 빈 시험 목록 조회 | Load  ExamList | 20210901 | 비어있는 시험 목록이 반환 되는가 |
| 3\_4 | 시험 날짜리스트  가져오기 | Load  Exam  DateList | test1 | 가져온 날짜 정보가  10에서 추가한 시험의  날짜와 같은가 |
| 3\_5 | 시험 삭제 | Execute  Query | DELETE FROM ExamList WHERE subjectName = 'test1'; | return true인가  시험 목록에서 삭제  되었는가 |
| 3\_6 | 삭제 후 시험목록 조회 | Load  ExamList | 20211231 | 21에서 삭제한 시험이 DB에서 삭제 되었는가 |
| 4\_1 | 알림 정보 추가 | Execute  Query | INSERT INTO AlarmInfoList VALUES('test1', '1일 전', '1시간 전', '1시간 전'); | return true인가 |
| 4\_2 | 알림 목록 조회 | loadAlarmList |  | 23에서 추가한 알림이 DB에 저장 되었는가 |
| 4\_3 | 동일한  알림 정보 추가 | Execute  Query | INSERT INTO AlarmInfoList VALUES('test1', '1일 전', '1시간 전', '1시간 전'); | Return false인가 |
| 4\_4 | 알림 정보 삭제 | Execute  Query | DELETE FROM AlarmInfoList WHERE subjectName = 'test1'; | Return true인가  알림 목록에서 삭제 되었는가 |
| 4\_5 | 알림 목록 조회 | Load  AlarmList |  | 26에서 삭제한 알림이 DB에서 제대로 삭제되었는가 |
| 5\_1 | 시스템 알림 추가 | Set  AlarmNum | a, test  b, test  c, test  d, test | 알림 목록에 알림을  저장하며 받은  고유번호가 각각 1, 2, 3, 4로 부여되는가 |
| 5\_2 | 시스템  알림  번호  불러오기 | Load  AlarmNum | a  b  c  d | 이름에 해당하는  알림의 고유번호가  알맞게 반환되는가 |
| 5\_3 | 과목에  설정된  알림 번호  불러오기 | Load  Alarm  SubjectList | test | 과목에 등록된 알림  번호들의 번호가  1, 2, 3, 4 리스트와  일치하는가 |
| 5\_4 | 시스템  알림 삭제 | Execute  Query | DELETE FROM AlarmList WHERE number = 1; | Return true인가 |
| 5\_5 | 존재하지 않는  알림 번호 불러오기 | Load  AlarmNum | A | 31에서 삭제한 알림이 DB에서 삭제 되었는가 |

## **SubjectManagement**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 번호 | case | method | parameter | test |
| 0\_01 | 2학기시작날짜계산 | getstartDate | 2021,2,4 | 계산한 날짜가 “2021-09-01"과 일치하는가 |
| 0\_02 | 1학기시작날짜계산 | getstartDate | 2021,1,3 | 계산한 날짜가 “2021-03-02"와 일치하는가 |
| 1\_01 | 정상적인 과목 입력 | addSubject | "test1",1,2,"0900",2,"1030",2021,2 | return 1 |
| 1\_02 | 강의개수가1개인경우 | addSubject | "test2",1,2,"0900",2,"1030",2021,2 | return 1 |
| 1\_03 | 강의개수가9개인경우 | addSubject | "test3",9,2,"0900",2,"1030",2021,2 | return 1 |
| 1\_04 | 강의개수가 마이너스  1개인경우 | addSubject | "test4",-1,2,"0900",2,"1030",2021,2 | return -1 |
| 1\_05 | 강의개수가  10개인경우 | addSubject | "test5",10,2,"0900",2,"1030",2021,2 | return -1 |
| 1\_06 | 시가 24시인  경우 | addSubject | "test6",1,2,"2401",2,"2401",2021,2 | return -1 |
| 1\_07 | 분이 60분인  경우 | addSubject | "test7",10,2,"0960",2,"1060",2021,2 | return -1 |
| 1\_08 | 년도가1899년인경우 | addSubject | "test8",1,2,"0900",2,"1030",1899,2 | return -1 |
| 1\_09 | 년도가1900년인경우 | addSubject | "test9",1,2,"0900",2,"1030",1900,2 | return 1 |
| 1\_10 | 년도가2100년인경우 | addSubject | "test10",1,2,"0900",2,"1030",2100,2 | return 1 |
| 1\_11 | 년도가2101년인경우 | addSubject | "test11",1,2,"0900",2,"1030",2101,2 | return -1 |
| 1\_12 | 시작 요일이  2인경우 | addSubject | "test12",1,2,"0900",2,"1030",2101,2 | return 1 |
| 1\_13 | 시작 요일이  6인경우 | addSubject | "test13",1,6,"0900",6,"1030",2021,2 | return 1 |
| 1\_14 | 시작 요일이1인경우 | addSubject | "test14",1,1,"0900",1,"1030",2021,2 | return -1 |
| 1\_15 | 시작 요일이7인경우 | addSubject | "test15",1,7,"0900",7,"1030",2021,2 | return -1 |
| 1\_16 | 학기가  0인경우 | addSubject | "test16",1,2,"0900",7,"1030",2021,0 | return -1 |
| 1\_17 | 학기가  1인경우 | addSubject | "test17",1,2,"0900",2,"1030",2021,1 | return 1 |
| 1\_18 | 학기가  3인경우 | addSubject | "test18",1,2,"0900",2,"1030",2021,3 | return -1 |
| 1\_19 | 이미  존재하는  과목입력 | addSubject | "test19",1,2,"0900",2,"1030",2021,2 | return 0 |
| 2 | 과목목록조회 | getSubjectList |  | 앞의 테스트에서 추가된  과목의 목록과  getSubjectList의 결과가  일치하는가 |
| 3 | 과목삭제 | delSubject, getSubjectList | “test1” | test1 이 결과 getSubjectList의 결과에서 삭제 되었는가 |

## **ToDoManagement**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 번호 | case | method | Parameter | test |
| 0\_01 | 과제 추가 | addAssignment | "test","asstest1", "20211001", "0000", "20211101", "0000" | return true |
| 0\_02 | 년도가 1889년인 경우 | addAssignment | "test","asstest2", "18891001", "0000", "18891101", "0000" | return false |
| 0\_03 | 년도가 2101년인 경우 | addAssignment | "test","asstest3", "21011001", "0000", "21011101", "0000" | return false |
| 0\_04 | 월이 0인 경우 | addAssignment | "test","asstest4", "20210001", "0000", "20210002", "0000" | return false |
| 0\_05 | 월이 13인 경우 | addAssignment | "test","asstest5", "20211301", "0000", "20211302", "0000" | return false |
| 0\_06 | 일이 0인 경우 | addAssignment | "test","asstest6", "20211000", "0000", "20211100", "0000" | return false |
| 0\_07 | 일이 32인 경우 | addAssignment | "test","asstest7", "20211032", "0000", "20211132", "0000" | return false |
| 0\_08 | 시가 24인 경우 | addAssignment | "test","asstest8", "18891001", "2400", "18891101", "2400" | return false |
| 0\_09 | 분이 60인 경우 | addAssignment | "test","asstest9", "18891001", "0060", "18891101", "0060" | return false |
| 1\_01 | 시험 추가 | addExam | "test","examtest1",  "20211001","0000" | return true |
| 1\_02 | 년도가 1889년인 경우 | addExam | "test","examtest2",  "18991001","0000" | return false |
| 1\_03 | 년도가 2101년인 경우 | addExam | "test","examtest3",  "21011001","0000" | return false |
| 1\_04 | 월이 0인 경우 | addExam | "test","examtest4",  "20211001","0000" | return false |
| 1\_05 | 월이 13인 경우 | addExam | "test","examtest5",  "20211301","0000" | return false |
| 1\_06 | 일이 0인 경우 | addExam | "test","examtest6",  "20211000","0000" | return false |
| 1\_07 | 일이 32인 경우 | addExam | "test","examtest7",  "20211132","0000" | return false |
| 1\_08 | 시가 24인 경우 | addExam | "test","examtest8",  "20211001","2400" | return false |
| 1\_09 | 분이 60인 경우 | addExam | "test","examtest9",  "20211001","0060" | return false |
| 2\_01 | 할 일 목록 조회 | getToDoList | “20211001” | 0\_01, 1\_01에서  추가된 할 일이 getToDoList의  결과와 일치하는가 |
| 2\_02 | 빈 할 일 목록 조회 | getToDoList | “20211102” | 비어있는 할 일  목록이 반환되는가 |
| 2\_03 | 완료여부변경 | changeIsDone | "asstest1", "test", "assignment", 1 | return 1 |
| 2\_04 | 완료여부변경 후 할 일 목록 조회 | getToDoList | “20211001” | 2\_02에서 변경한 완료여부가 getToDoList의  결과와 일치하는가 |
| 3\_01 | 과제 삭제 | delAssignment | "asstest1", "test" |  |
| 3\_02 | 과제 삭제 후  할 일 목록 조회 | getToDoList | “20211001 | 3\_01에서 삭제한  과제가  getToDoList의  결과에서 삭제  되었는가 |
| 4\_01 | 시험 삭제 | delExam | "examtest1", "test" |  |

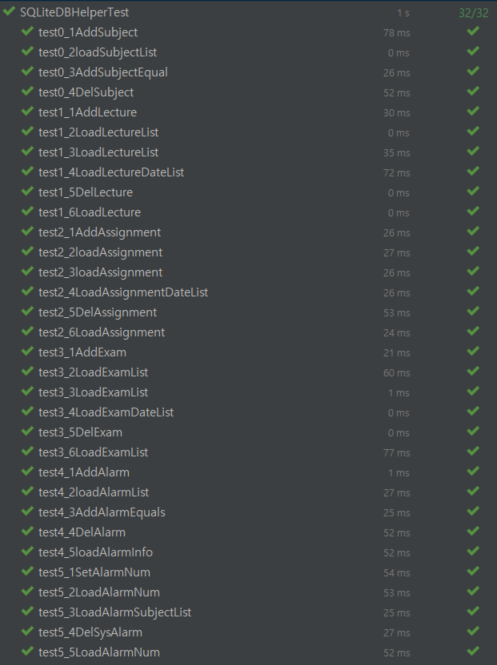
## **AlarmManagement**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 번호 | case | method | parameter | test |
| 1\_01 | 정상적인 알림 추가 | addAlarm | "testAlarm","1일 전","1시간 전", "1시간 전") | return true |
| 1\_02 | 알림 목록 조회 | Get  AlarmList |  | 1에서 추가한 알림 정보가 결과리스트에 들어가 있는가 |
| 1\_03 | 이미 존재하는 알림 추가 | addAlarm | "testAlarm","1일 전","1시간 전", "1시간 전" | return false |
| 1\_04 | 알림 파라미터가 형식을 벗어나는 경우 | addAlarm | "testAlarm","a","b", "c" | return false |
| 2\_01 | 정상적인 알림 삭제 | delAlarm | " testAlarm " | return true |
| 2\_02 | 알림 목록조회 | get  AlarmList |  | 결과 리스트가 비어있는가 |
| 2\_03 | 존재하지 않는  알림 삭제 | del  Alarm | "test0" | return false |

# Unit 테스트 결과



## **SQLiteDBHelper**

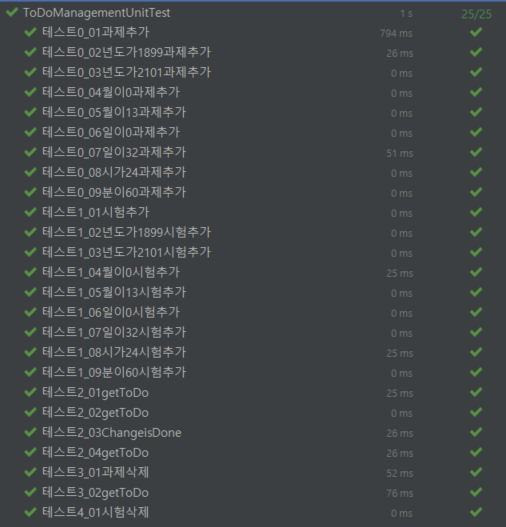


## **SubjectManagement**

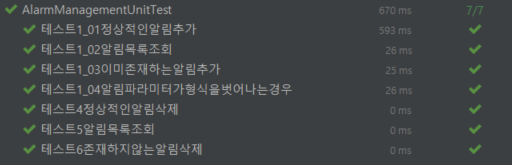
텍스트이(가) 표시된 사진

자동 생성된 설명

## **ToDoManagement**



## **AlarmManagement**



# 시스템 테스트 시나리오



## **과목추가 삭제**

|  |  |  |
| --- | --- | --- |
| 번호 | Case | 테스트 |
| 1 | 과목입력 | "test1","1","월","0900","월","1030", 2021, 2 |
| 2 | 과목삭제 | 과목 선택 |

## **시험 과제추가 삭제**

|  |  |  |
| --- | --- | --- |
| 번호 | case | 테스트 |
| 1 | 시험 과제추가 | "test1","20210901","0000","20211001","0000" |
| 2 | 시험 과제삭제 | 시험 과제 선택 |

## **수업일정 조회**

|  |  |  |
| --- | --- | --- |
| 번호 | case | 테스트 |
| 1 | 수업일정 조회 | 일정 목록 조회 화면에 접속 |
| 2 | 날짜를 선택한 수업일정 조회 | 2021/09/06 선택 |
| 3 | 할 일의 완료구분 변경 | 완료구분 완료로 설정 |

## **친구목록 조회**

|  |  |  |
| --- | --- | --- |
| 번호 | case | 테스트 |
| 1 | 서로 친구목록 조회 | 친구 관리 화면에 접속 또는 서로 친구 목록 버튼 선택 |
| 2 | 받은 친구신청목록 조회 | 친구 관리 화면에서 받은 친구 신청 버튼 선택 |
| 3 | 받은 친구신청 수락 | 친구 관리 화면의 받은 친구 신청 목록에서 수락 버튼 선택 |

## **친구 수업일정 조회**

|  |  |  |
| --- | --- | --- |
| 번호 | case | 테스트 |
| 1 | 친구 수업일정 조회 | 친구 관리 화면의 서로 친구 목록에서 친구 선택 |
| 2 | 날짜를 선택한 친구 수업일정 조회 | 친구 일정 조회 화면에 접속 |

## **친구 추가 삭제**

|  |  |  |
| --- | --- | --- |
| 번호 | case | 테스트 |
| 1 | 존재하는 사용자 추가 | 시스템에 로그인 한 적이 있는 다른 사용자의 아이디 |
| 2 | 친구 신청 수락 | 수락 버튼 선택 |
| 3 | 친구 삭제 | 친구 선택 |

## **알림 추가 삭제**

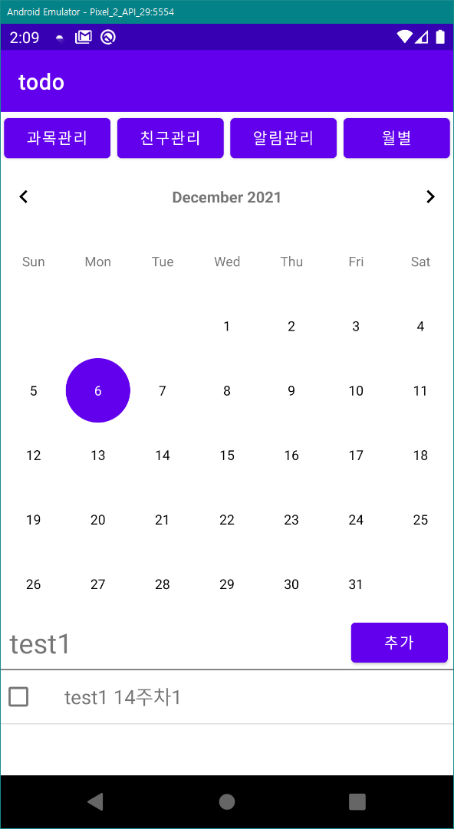
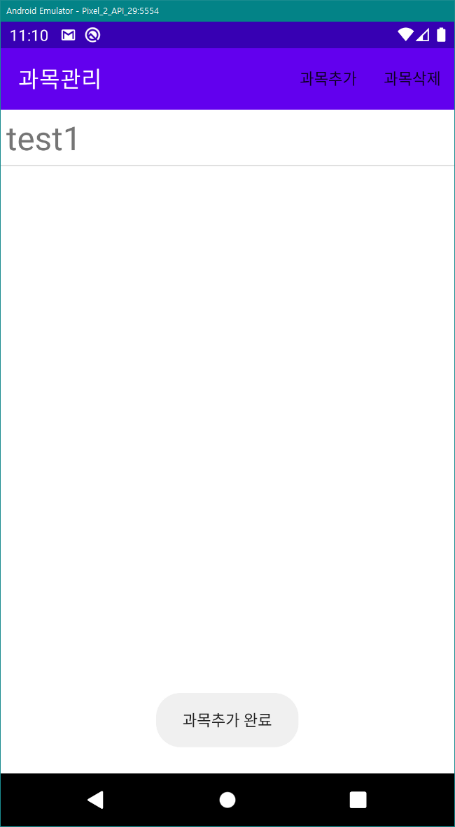
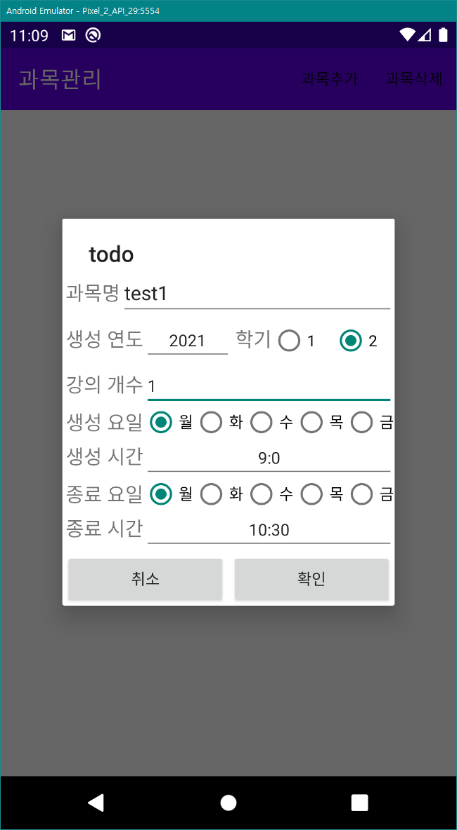
|  |  |  |
| --- | --- | --- |
| 번호 | Case | 테스트 |
| 1 | 알림 추가 | "testAlarm","1일 전","1시간 전","1시간 전" |
| 2 | 알림 삭제 | 알림 선택 |

# 시스템 테스트 결과



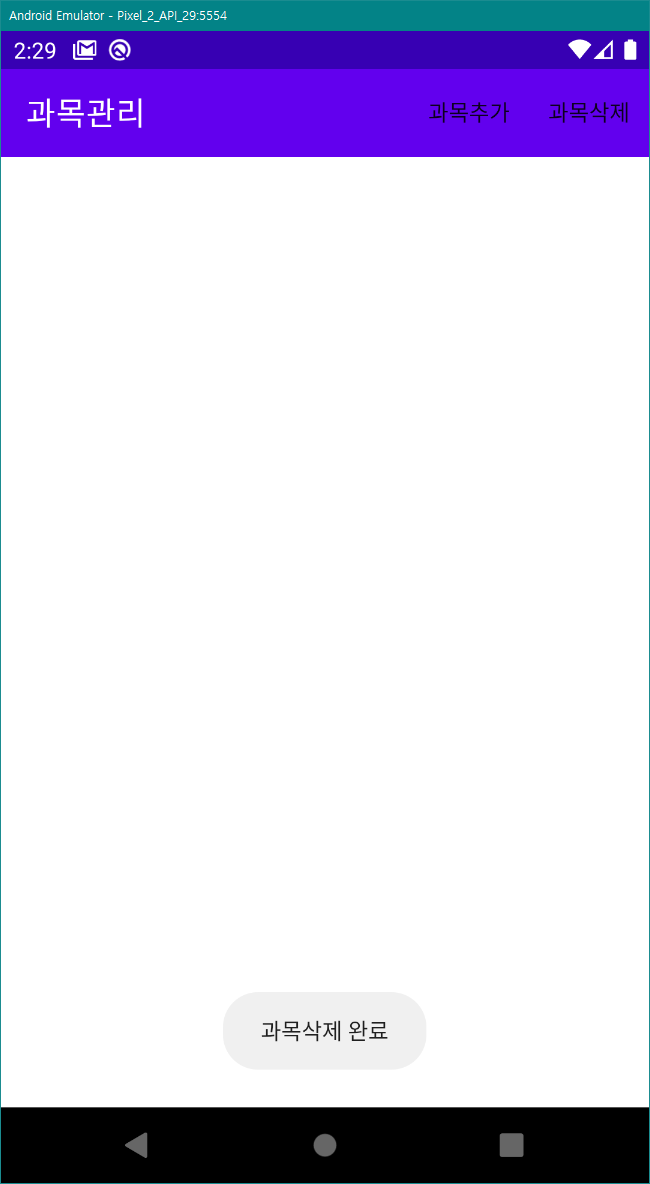
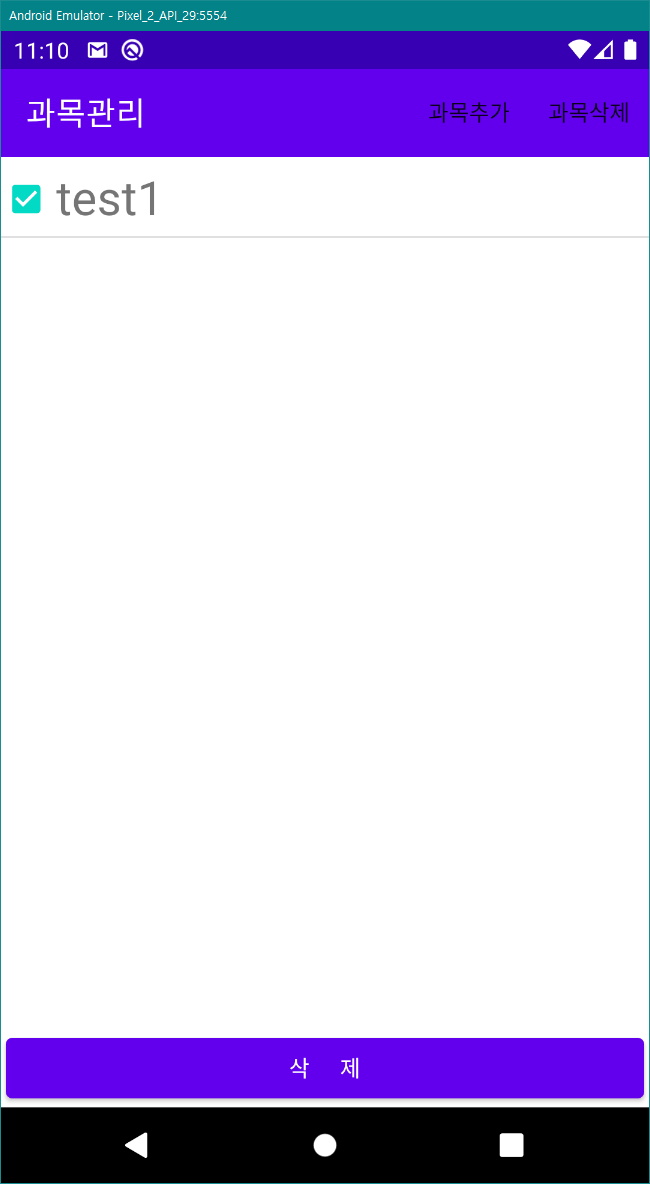
## **과목 추가 삭제**

* + 1. **과목 추가**



과목 추가 창에서 과목 정보를 입력하고 확인버튼을 누르면, 과목추가 완료 메시지가 제대로 뜨고 관리 화면에 과목이 추가 되어있음을 확인할 수 있다.

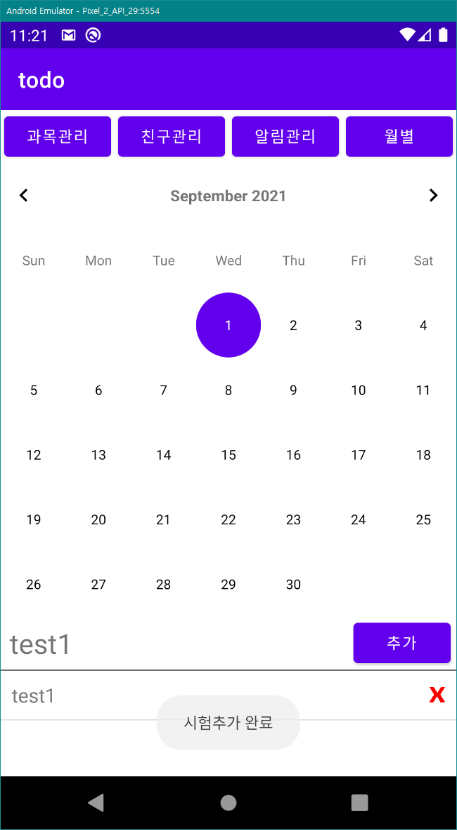
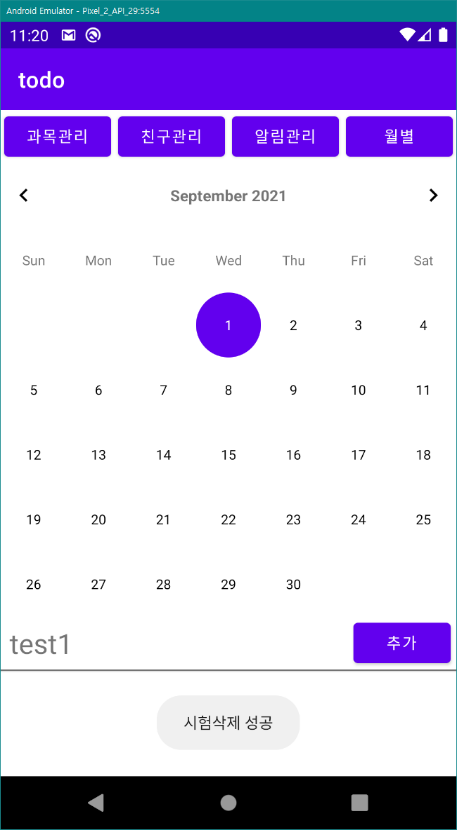
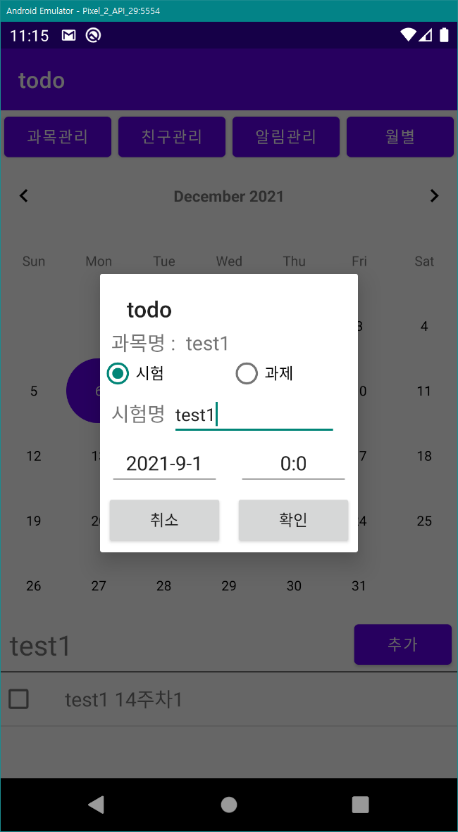
* + 1. **과목 삭제**



과목을 선택한 후 삭제버튼을 누르면, 완료 메시지가 제대로 뜨고 있으며 과목관리 화면에서 삭제되었음을 확인할 수 있다.

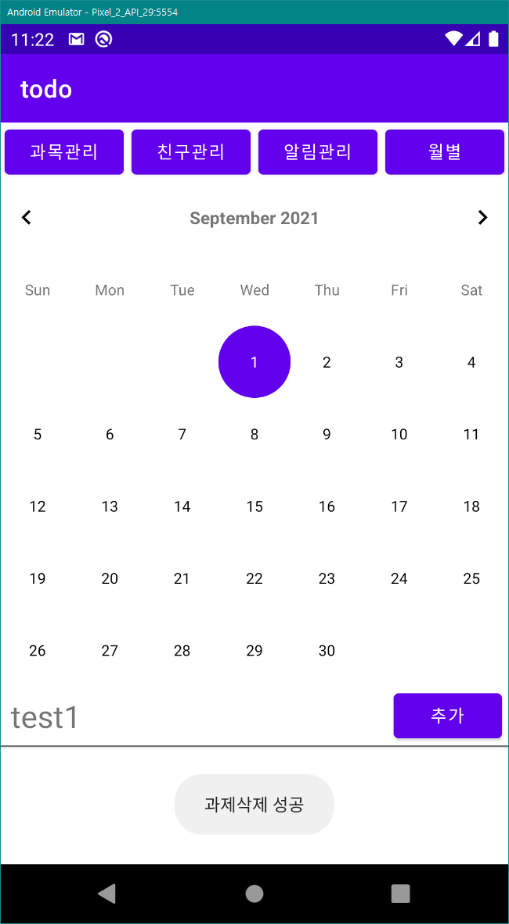
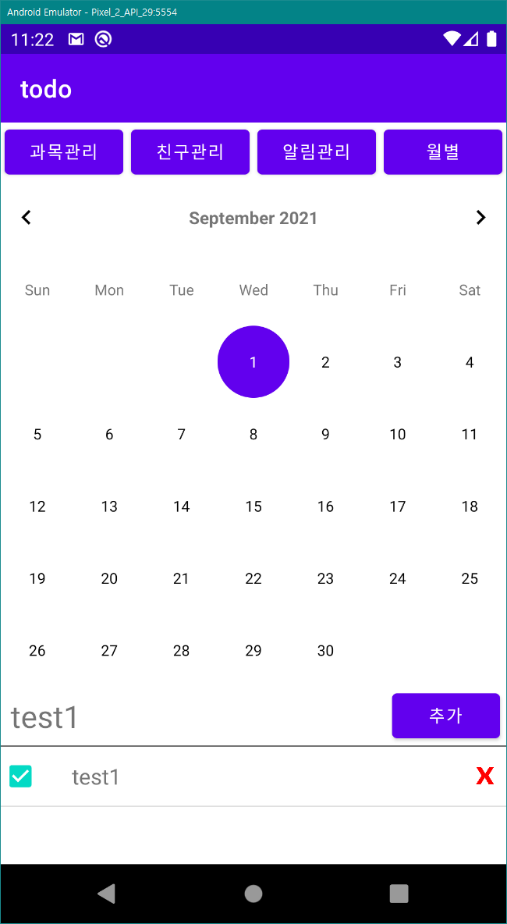
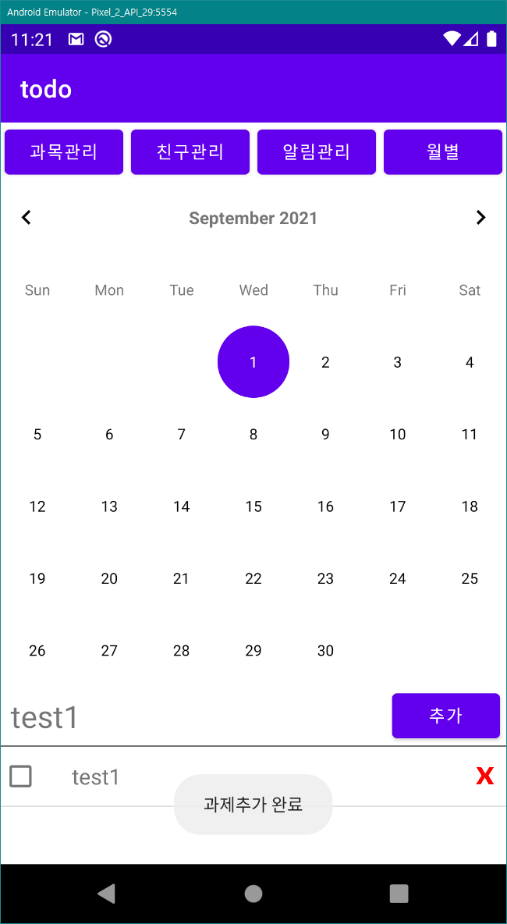
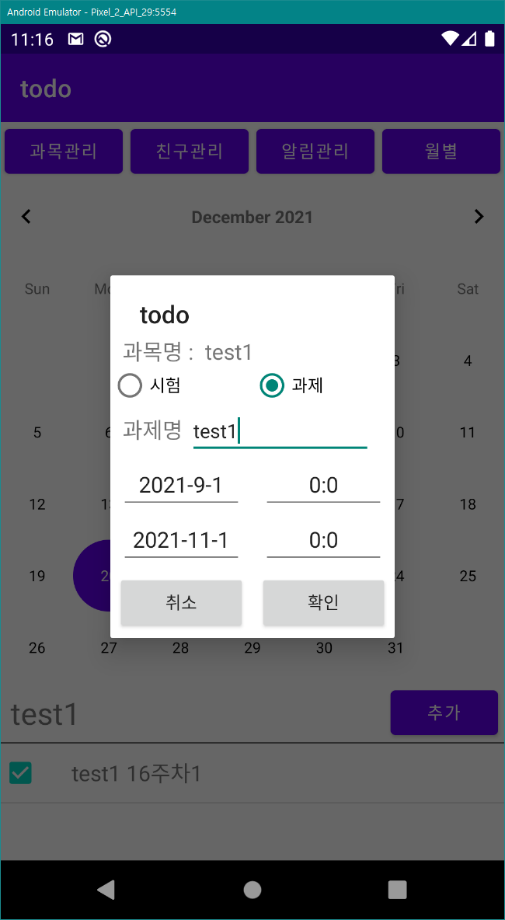
## **시험 과제추가, 삭제**

* + 1. **시험추가, 삭제**



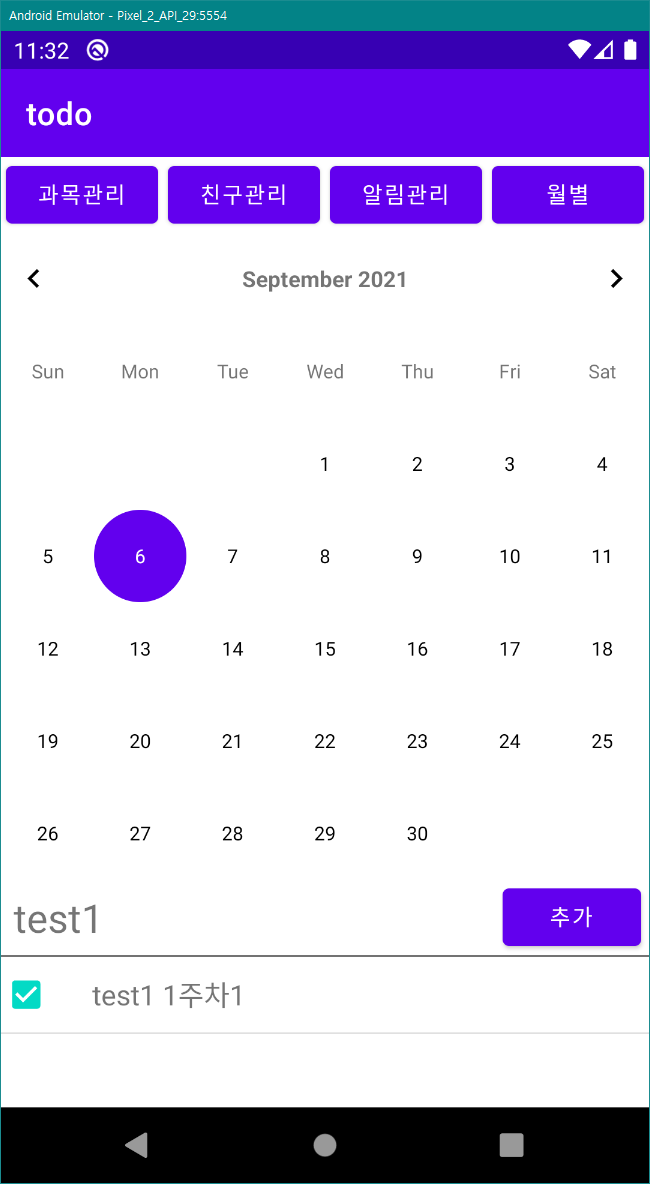
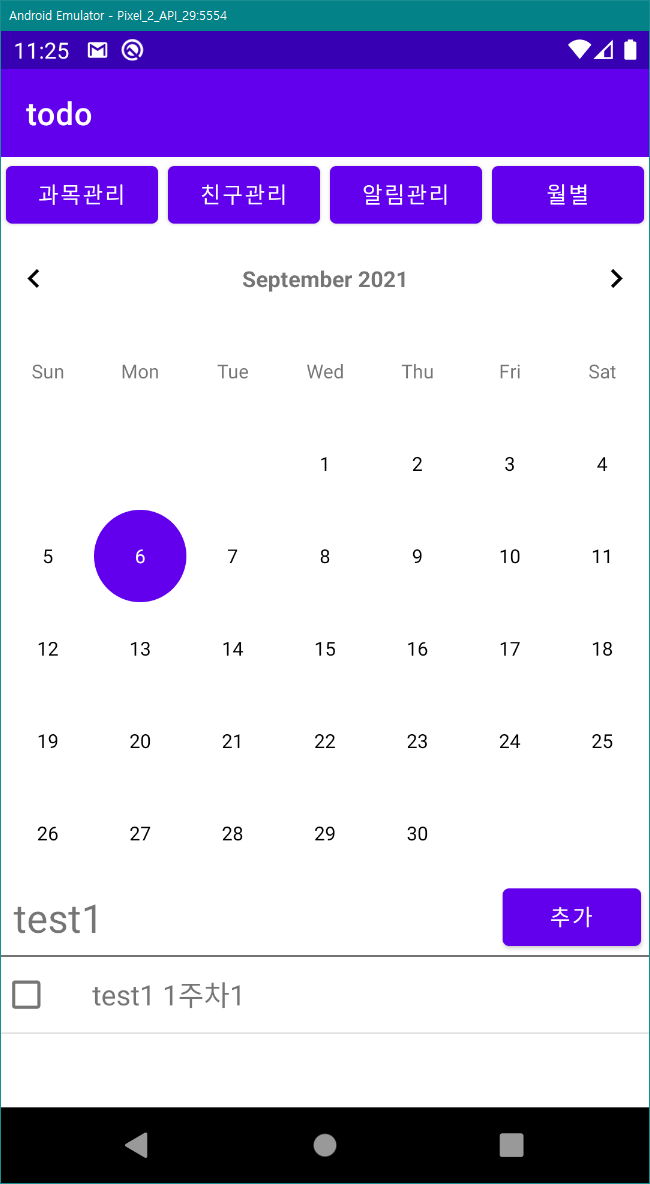
시험추가 완료 메시지가 제대로 뜨고 있으며 일정조회 화면 하단의 과목목록에 시험이 추가되어 있음을 확인할 수 있다. 또한 우측의 X버튼을 눌렀을 때 시험삭제 완료 메시지가 제대로 뜨고 화면에서도 사라졌음을 확인할 수 있다.

* + 1. **과제 추가 삭제**



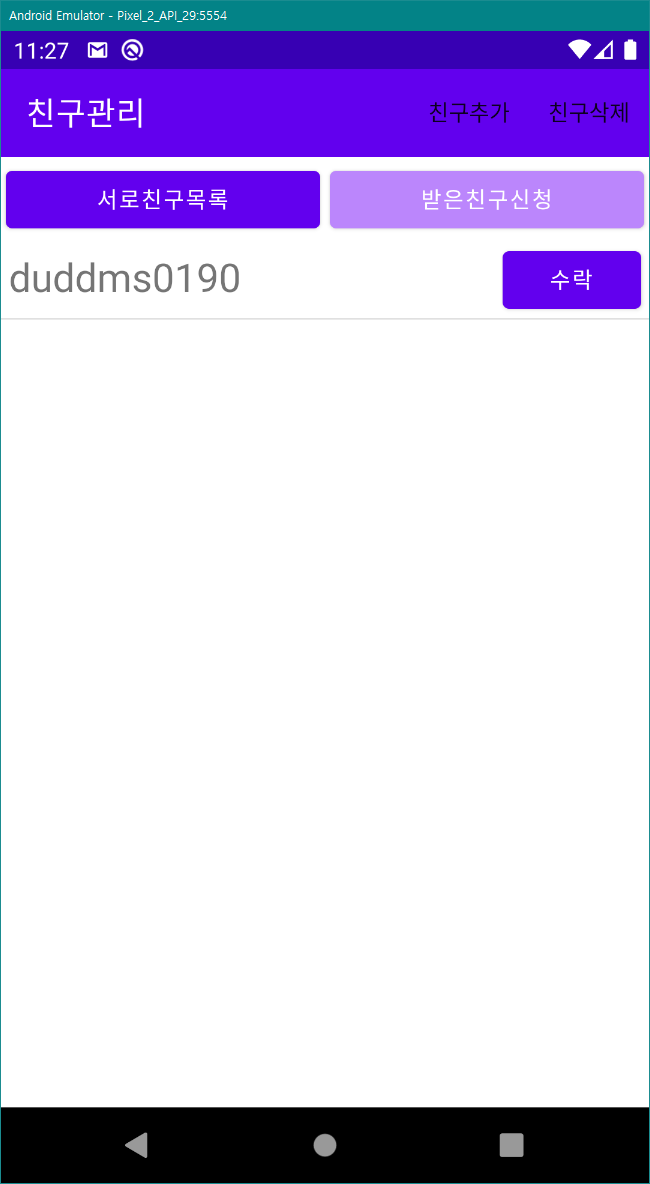
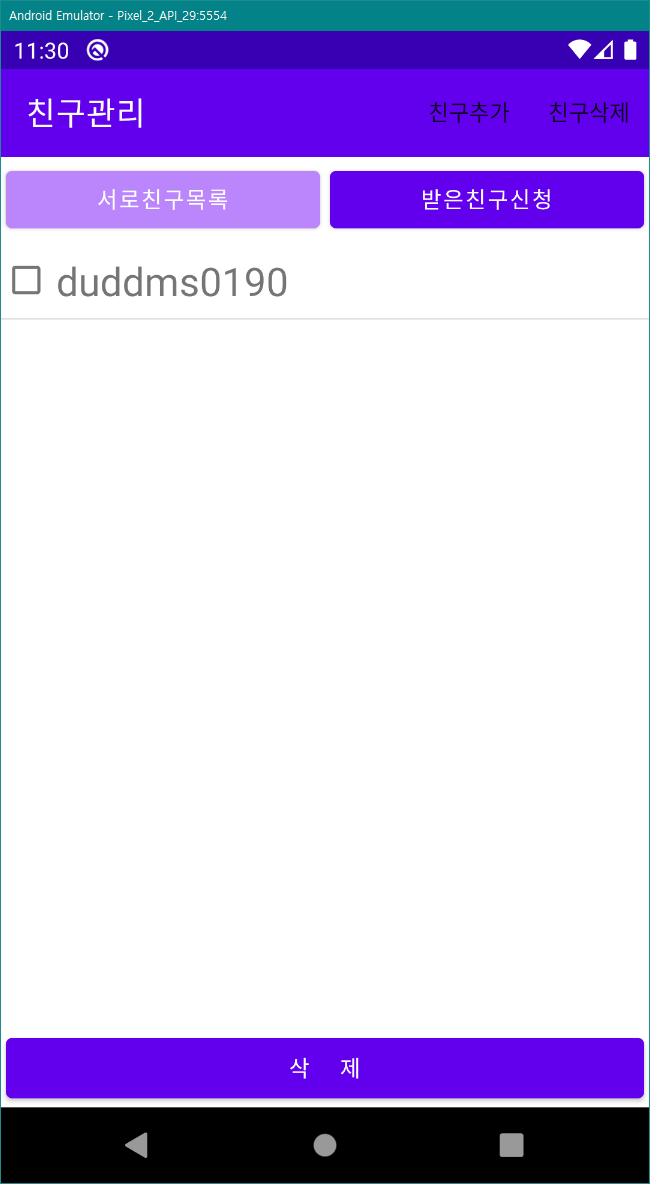
과제추가 완료 메시지가 제대로 뜨고 있으며 일정조회 화면 하단의 과목목록에 과제가 추가되어 있음을 확인할 수 있다. 또한 우측의 X버튼을 눌렀을 때 과제삭제 완료 메시지가 제대로 뜨고 화면에서도 사라졌음을 확인할 수 있다.

## **수업일정 조회**



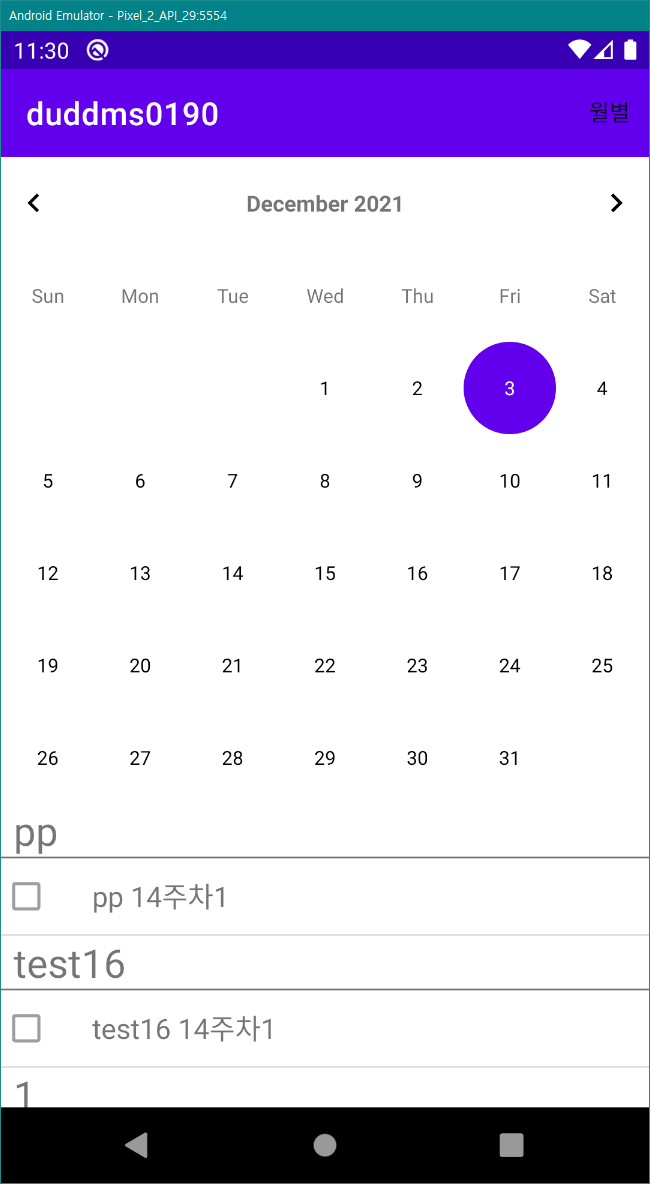
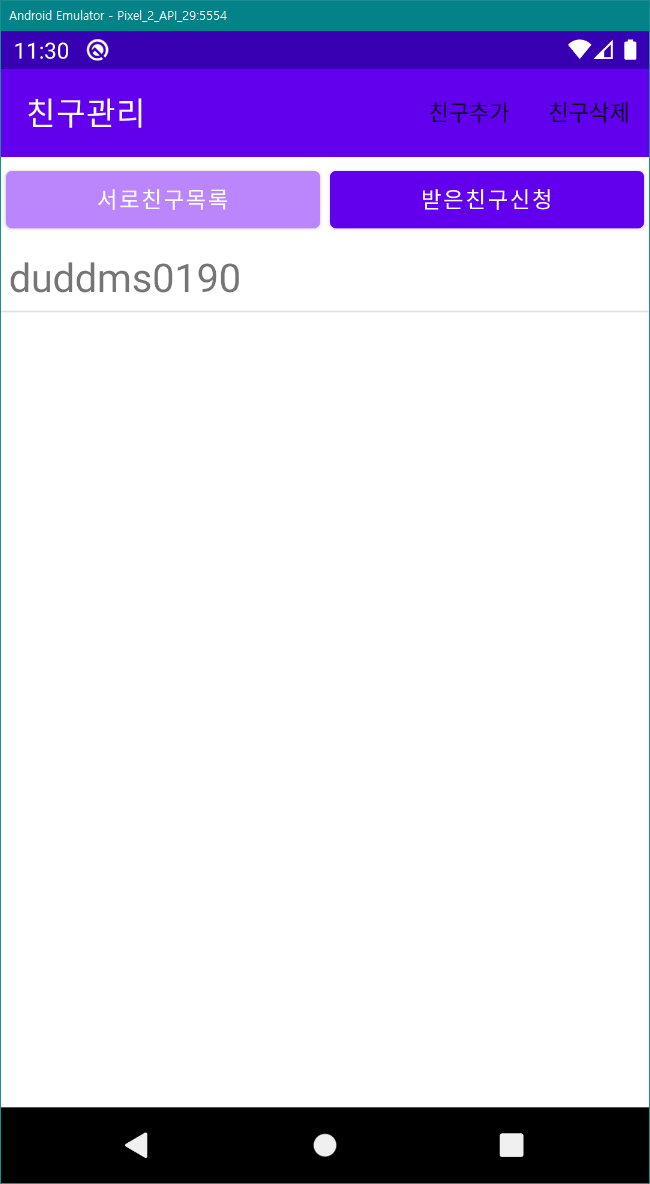
수업 일정 조회 화면에서 오늘 날짜에 해당하는 할 일 목록이 올바르게 표시되고 있음을 확인할 수 있으며 다른 날짜도 마찬가지로 해당 날짜의 할 일 목록도 올바르게 표시되고 있다.

## **친구목록 조회**



친구목록 조회 화면에 접속하거나 서로 친구 목록 버튼을 선택 시, 서로 친구 목록이 올바르게 표시되어 있고 받은 친구 신청 목록 버튼을 선택하면 받은 친구 신청 목록이 올바르게 표시되고 있음을 확인할 수 있다.

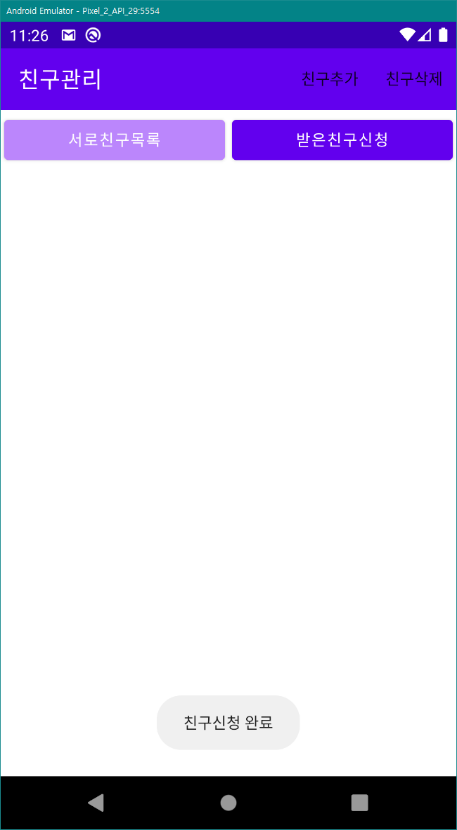
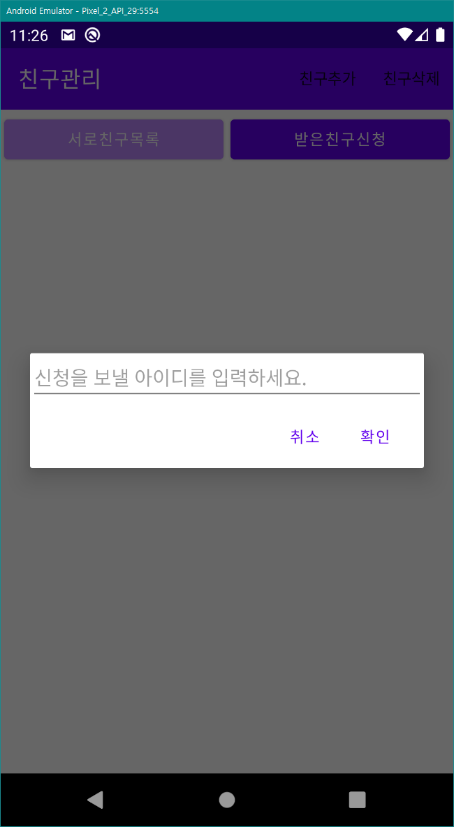
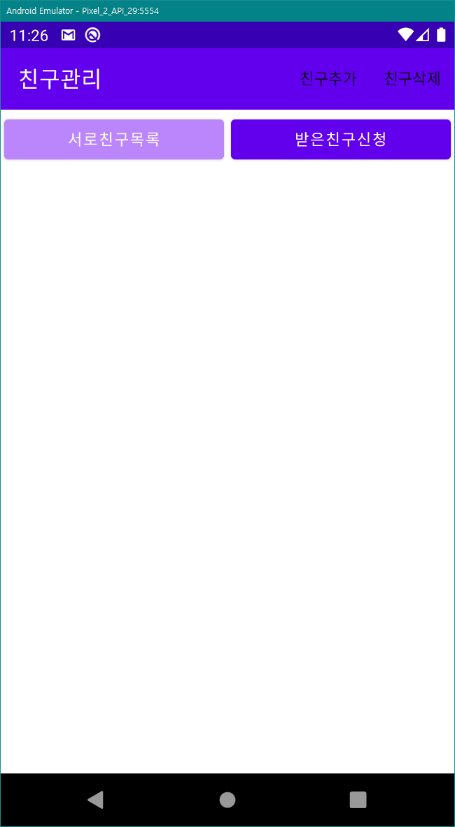
## **친구 수업일정 조회**

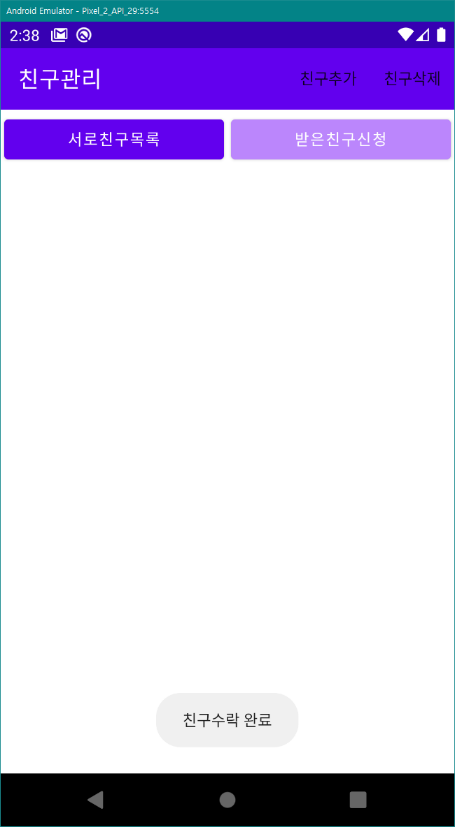
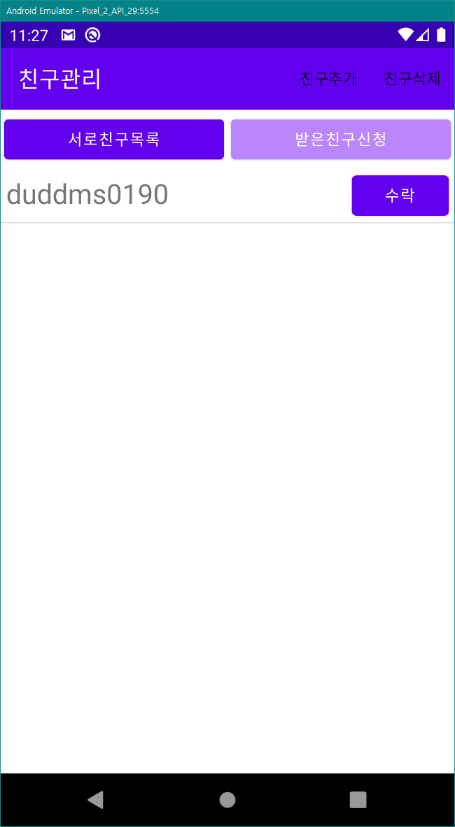


친구를 선택하여 들어간 친구 수업일정 조회 화면에서 오늘 날짜에 해당하는 친구의 할 일 목록이 올바르게 표시되어 있음을 확인할 수 있다.

## **친구 추가 삭제**

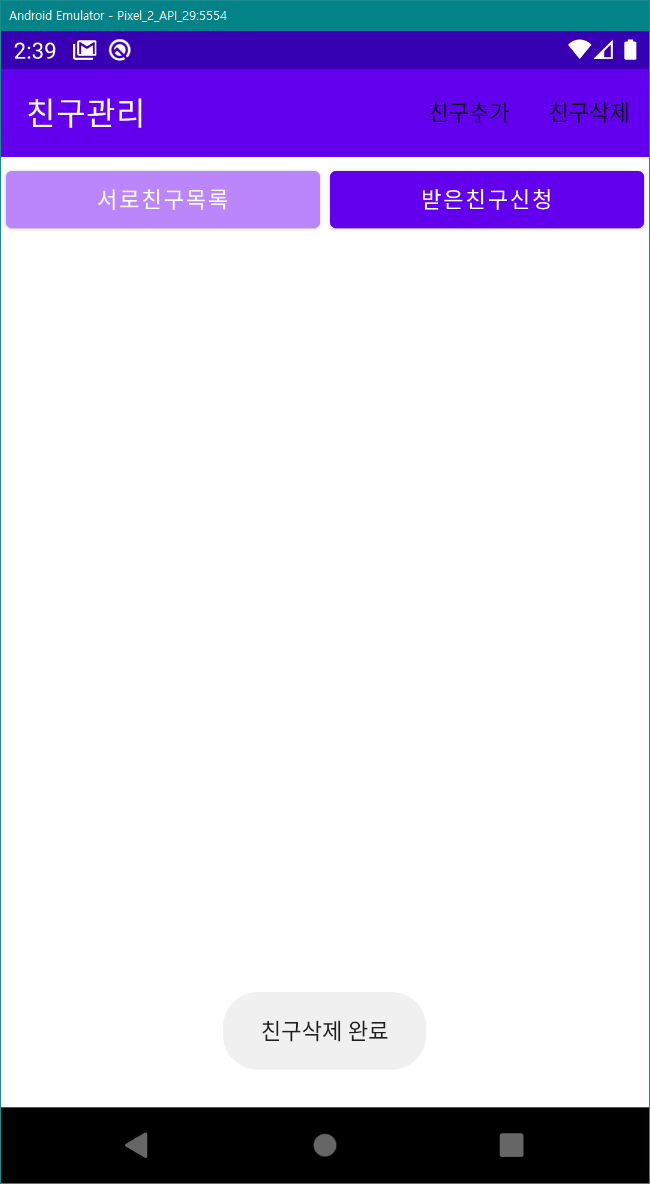
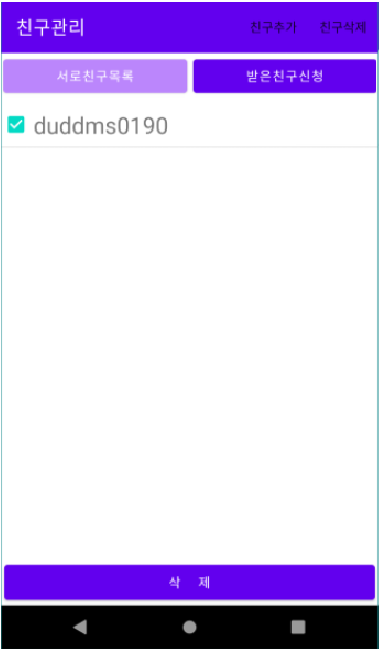
* + 1. **친구 추가**





친구관리화면의 친구 추가 버튼을 선택하고 존재하는 사용자에게 친구 신청을 보내어 친구신청 완료 메시지가 뜨는 것을 확인할 수 있다. 받은 친구 신청 목록에서는 받은 친구신청을 수락하고 친구수락 완료 메시지가 뜨는 것을 확인할 수 있다.

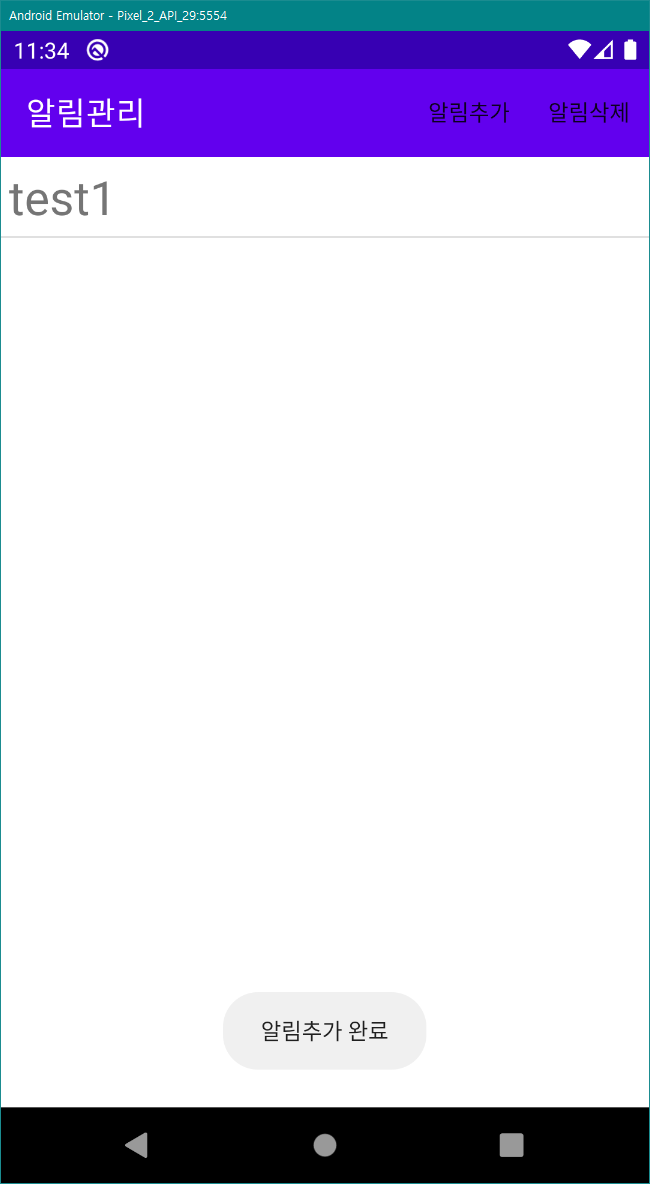
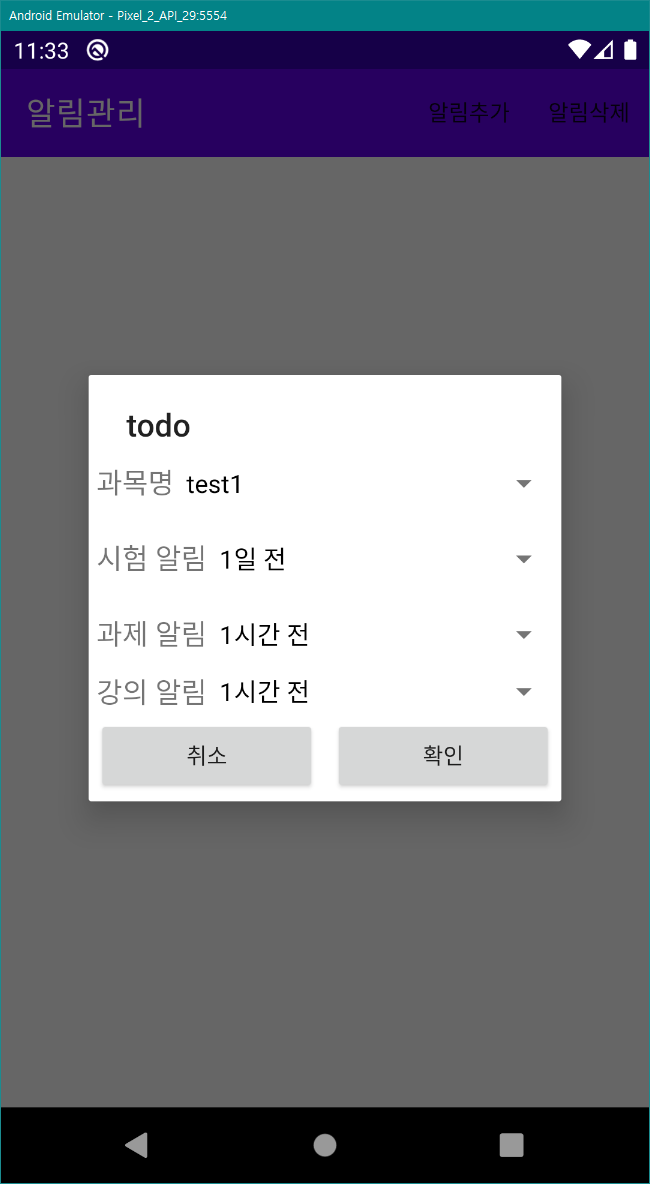
* + 1. **친구 삭제**



서로친구목록 화면에서 친구 삭제버튼을 누르면 친구이름 옆에 체크박스가 설정된 목록이 출력된다. 삭제할 친구의 체크박스를 선택하고 삭제 버튼을 누르면 친구삭제완료 메시지가 출력되고, 친구목록에서 친구 아이디가 제거된 것을 확인할 수 있다.

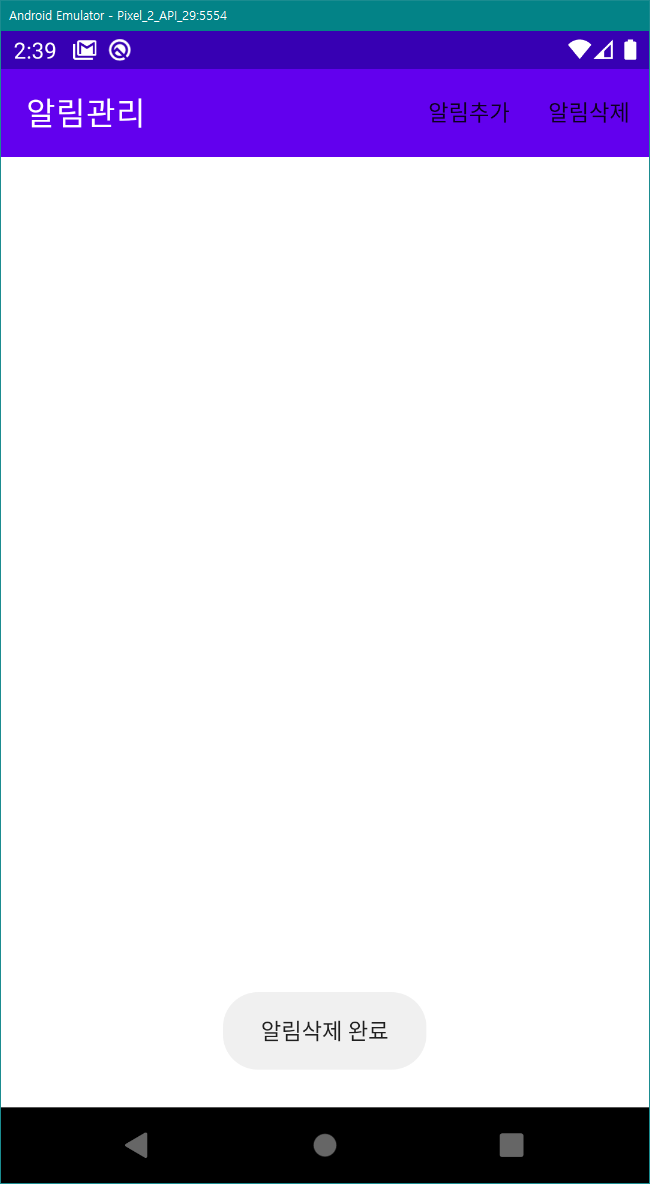
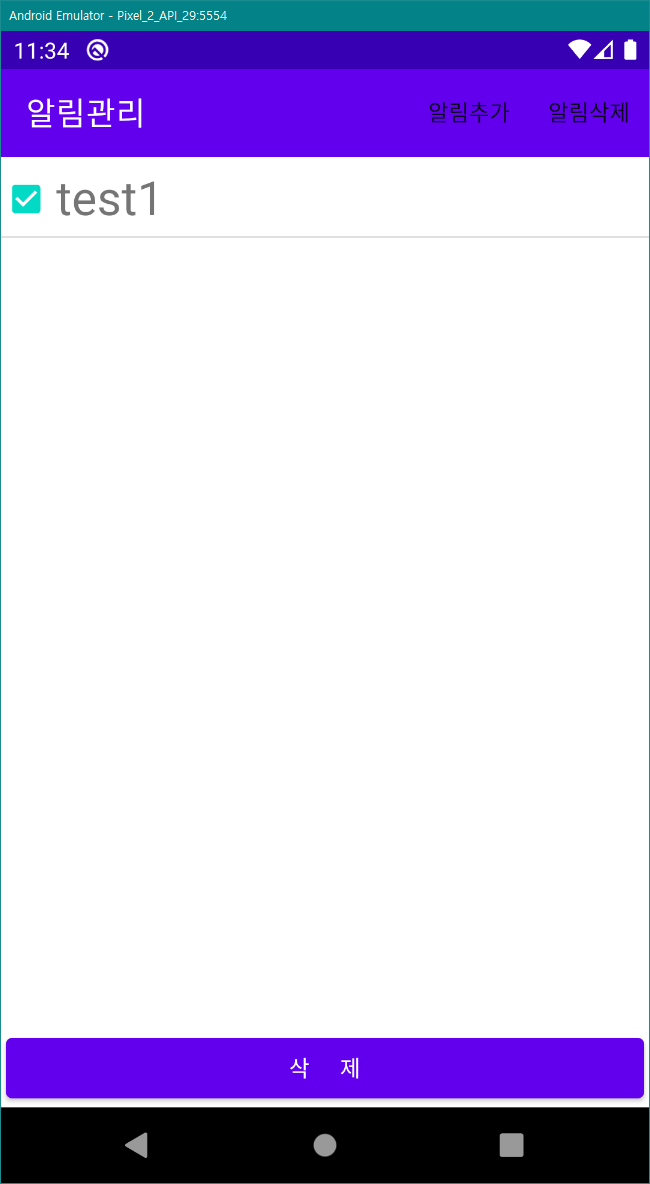
## **알림 추가 삭제**

* + 1. **알림 추가**



알림 관리 화면에서 알림을 추가한 후, 목록에 추가한 알림이 올바르게 표시되고 알림 추가 완료 메시지가 뜨는 것을 확인할 수 있다.

* + 1. **알림 삭제**



알림 관리 화면에서 알림 삭제 버튼을 누르면 알림 목록에 체크박스가 추가된 목록이 표시된다. 그 후, 체크박스를 선택하고 삭제버튼을 누르면 알림 삭제완료 메시지가 출력되고, 목록에서 알림이 삭제된 것을 확인할 수 있다.

# 비기능 요구사항 테스트 시나리오



## **품질 요구사항**

소프트웨어의 성능에 대한 요구사항이다. 아래의 표는 요구사항에 대한 설명이다.

|  |  |
| --- | --- |
| 품질 요구사항 | 설명 |
| 수업일정 조회속도 | 선택한 날짜의 수업 일정을 빠르게 불러오는 성능에 대한  요구사항 |
| 친구목록 조회속도 | 친구 목록을 선택 했을 때 저장된 친구 목록을 빠르게  불러오는 성능에 대한 요구사항 |
| 친구 수업일정 조회속도 | 친구 목록에서 선택한 친구의 수업 일정을 빠르게 불러오는 성능에 대한 요구사항 |
| 로그인 유지 | 시스템에 재접속 했을 때 로그인이 유지되어야 하는  편의성에 대한 요구사항 |

각각의 요구사항은 다음과 같은 방법으로 소요시간을 측정하도록 한다.

수업일정 조회속도: 프로그램을 최초로 실행하거나, 수업일정 조회화면에서 날짜를 선택하는 경우 시작시간을 초기화한다. 수업일정 목록이 화면에 출력이 완료되면 종료시간을 초기화 하고, 종료시간에서 시작시간의 차이를 구하여 출력한다.

친구목록 조회속도: 수업일정 조회화면에서 친구 관리 버튼을 누르거나, 친구 목록 조회 화면에서 서로 친구 목록, 받은 친구 신청 버튼을 누르는 경우 시작시간을 초기화한다. 친구 목록이 화면에 출력이 완료되면 종료시간을 초기화 하고, 종료시간에서 시작시간의 차이를 구하여 출력한다.

친구 수업일정 조회속도: 친구목록 화면에서 친구를 선택하거나, 친구 수업일정 조회화면에서 날짜를 선택하는 경우 시작시간을 초기화한다. 친구 수업일정 목록이 화면에 출력이 완료되면 종료시간을 초기화 하고, 종료시간에서 시작시간의 차이를 구하여 출력한다.

로그인 유지: 앱을 최초 설치 시에만 로그인을 진행하고, 그 이후 앱에 접속시에는 이전 로그인 기록을 통하여 로그인이 유지가가 되는 것을 앱의 실행 종료를 반복해보며 테스트한다.

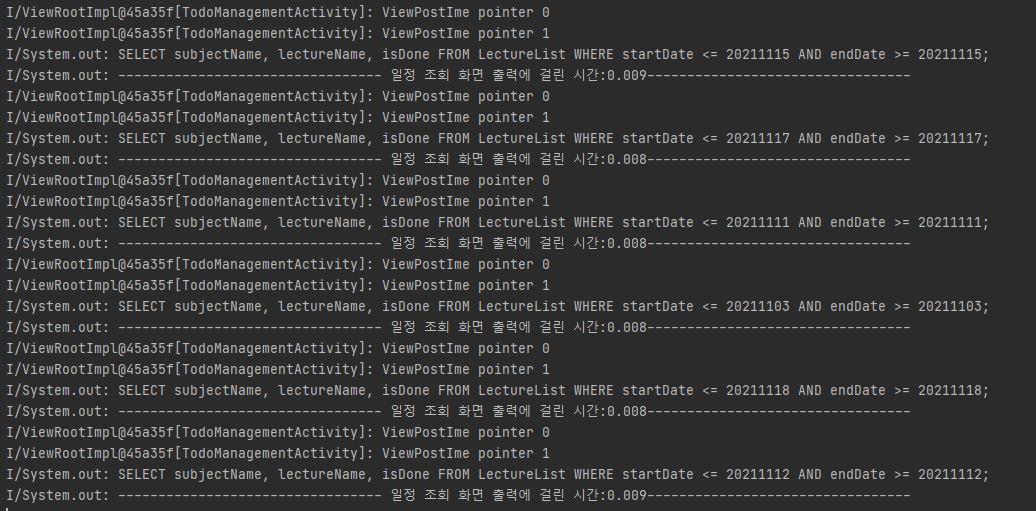
# 비기능 요구사항 테스트 결과



## **품질 요구사항**

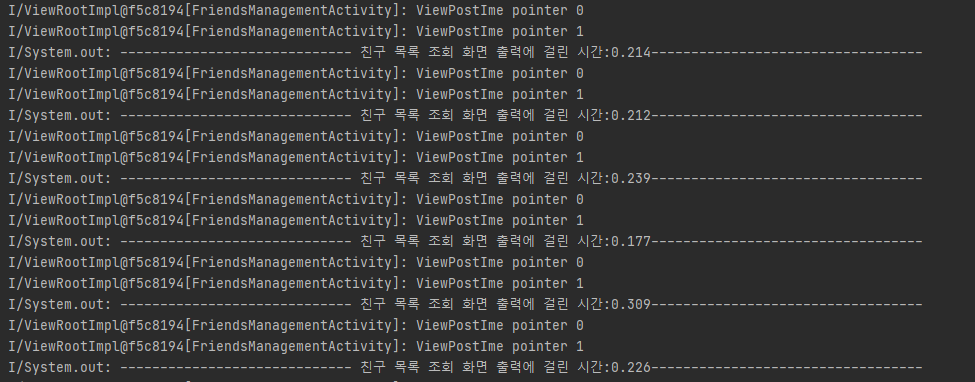
|  |  |  |
| --- | --- | --- |
| 품질 요구사항 | 충족 기준 | 결과 |
| 수업일정 조회속도 | 1초 이내 화면 출력 | 0.13초 |
| 친구목록 조회속도 | 3초 이내 화면 출력 | 0.3 |
| 친구 수업일정 조회속도 | 3초 이내 화면 출력 | 0.4 |
| 로그인 유지 | 다시 접속했을 때 로그인 유지 | 성공 |

앱을 최초로 실행하여 일정 조회 화면에 접속하는 경우, 평균적으로 0.13초의 시간이 소요되었으며, 최초 실행이 아닌 날짜 선택으로 일정을 조회하는 경우, 평균적으로 0.01초의 시간이 소요되었다.



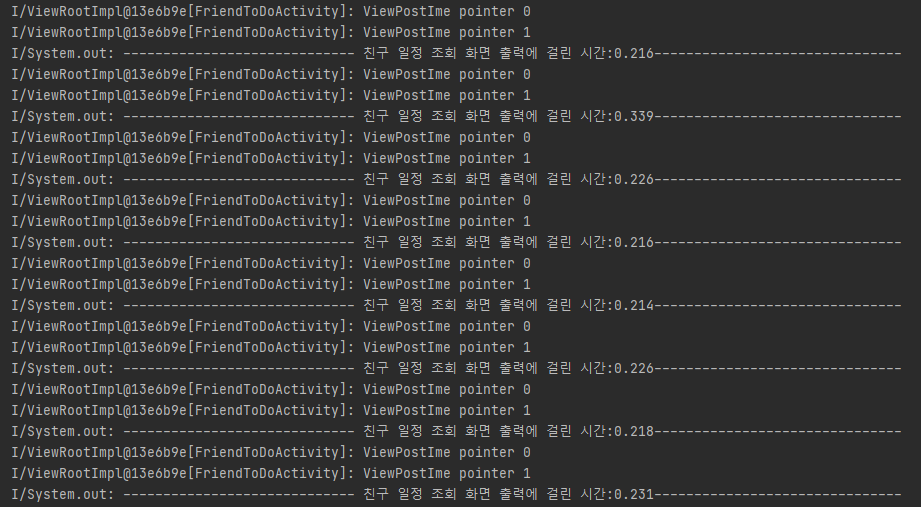
날짜 선택 시 일정 조회 소요 시간

일정 조회 화면에서 친구 관리 버튼을 클릭하여 처음으로 친구 목록 조회 화면을 출력하는 경우, 평균 적으로 0.3초의 시간이 소요되었으며, 친구관리 화면에서 서로친구 목록 또는 받은 친구 신청 버튼을 클릭하여 친구 목록 화면을 출력하는 경우, 평균적으로 0.2초의 시간이 소요되었다. 친구가 1명인 경우와 친구가 10명인 경우를 시험해 보았으며, 모두 비슷한 결과를 가졌다.



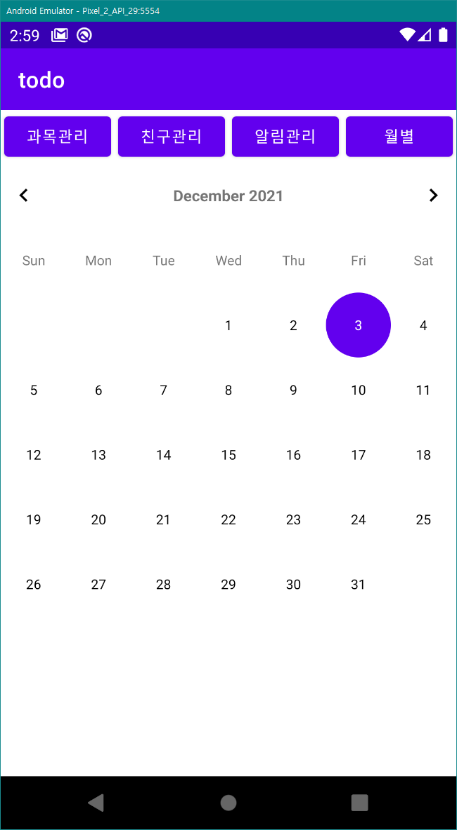
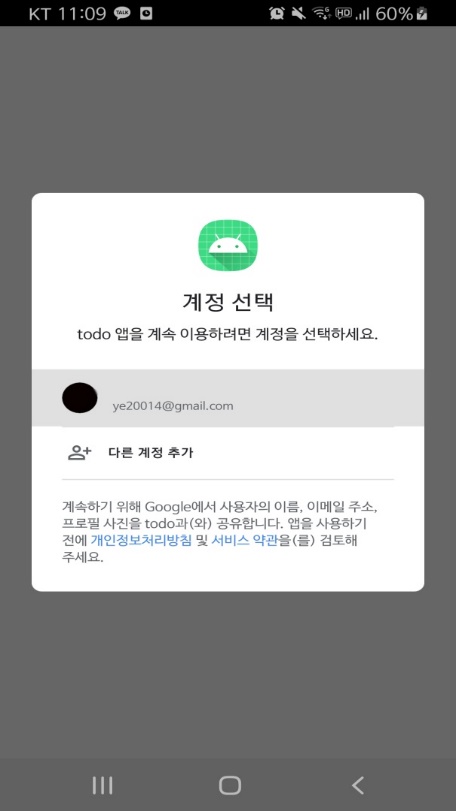
서로친구목록, 받은 친구신청 버튼 클릭 시 친구 목록 조회 소요시간

친구 목록에서 친구를 선택하여 친구 일정 조회 화면에 접속하는 경우, 평균적으로 0.4초의 시간이 소요되었으며, 날짜 선택으로 친구 일정을 조회하는 경우, 평균적으로 0.25초의 시간이 소요되었다.



날짜 선택 시 친구 일정 조회 소요시간

로그인 유지는 조원들의 휴대폰에서 모두 테스트 한 결과, 모두 성공하였다

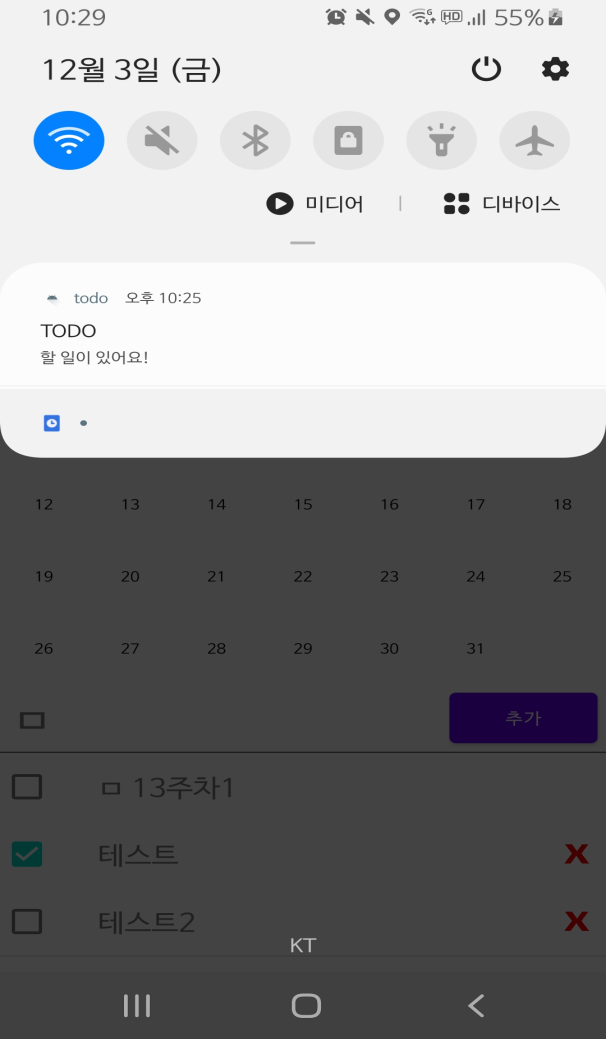
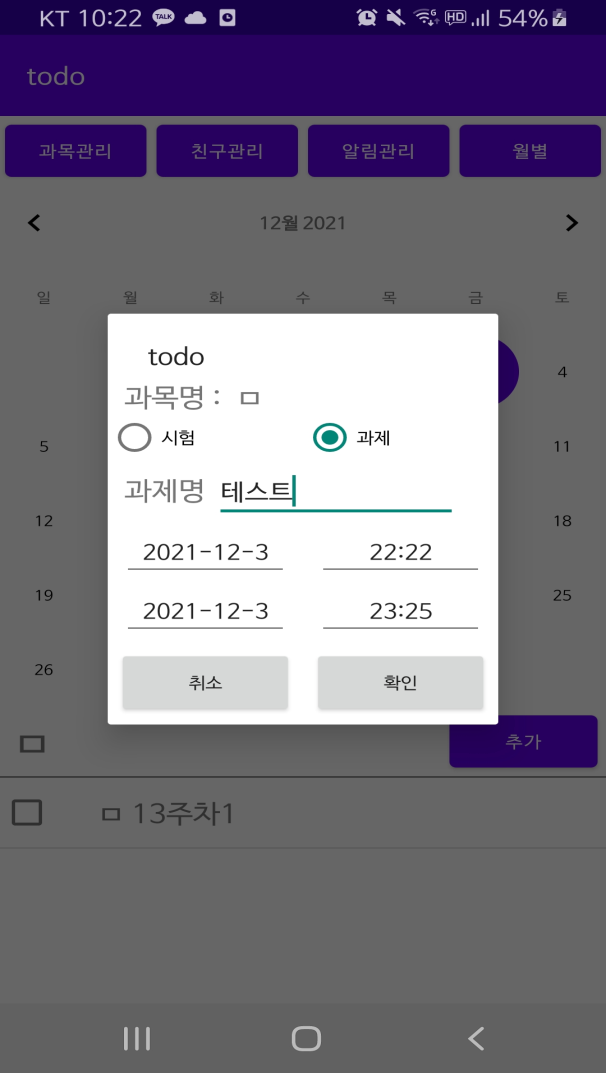


첫 로그인

# 결함 및 이슈 보고



## **알림 관련 이슈**



알림이 제대로 동작하는지 테스트하기 위해 과목에 알림을 설정한 후, 과제를 추가하고 체크표시를 하였으나 해당 과제에 대한 시스템 알림이 제거되지 않고 그대로 표시된 것을 확인할 수 있었다.

이는 개발 시간 부족으로 인하여 발생한 문제로, 추후 코드 수정을 통하여 개선할 예정이다.

## **회의기록**

|  |  |
| --- | --- |
| 일자 | 내용 |
| 2021/11/25  (zoom) | 발표자료 목차 작성 및 세부내용 계획 |
| 2021/11/26  (zoom) | 테스트 파트를 제외한 발표자료 완성 |
| 2021/11/28  (zoom) | 테스트 케이스 작성 |
| 2021/11/30  (zoom) | 모의 발표 및 발표자료 피드백, 제출 |
| 2021/12/02  (zoom) | 테스트 문서 작성 |
| 2021/12/03  (zoom) | 테스트 문서 작성 완료 |