Gebze Technical University
Faculty of Engineering

Project 2nd Report

University Management System

Group 12

Submission Date: May 9, 2021
Supervisors: Fatih Erdoğan Sevilgen

1 Group Members

Our group members names and their student ID's are given below.

- 171044057 Doruk Ergün
- 171044063 Erdi Bayır
- 141044025 Derya Kaptan
- 161044017 Doruk Can Başıbüyük
- 161044046 Ferdi Sönmez
- 161044019 İrfan Karatekin
- 1901042668 Şevval Ateş
- 161044116 Tuba Toprak
- 205008014002 Fatih Mahmud Koç

2 Problem Definition

In the period when information was manually kept and managed in education, the administrator, teacher and student faced many problems. When we consider these problems, the first problem we encounter is that teachers and system administrators have more workloads. Another problem is the possibility of loss of student information and documents belonging to the university in case of any adverse situation or accident. On the other hand, it is almost impossible to keep the information kept in this system up-to-date and to access this information at the desired time and place. The solution of all these problems is to design a system that can be dynamically managed and easy to use for all system users. It is extremely important that this system is extremely easy to access and use and that it is accessible from anywhere for all users.

These problem and solution suggestions led to the emergence of the idea of the University management system.With this system, the possibility of data loss, keeping the information up-to-date, reducing the workload of administrators and teachers, minimizing the loss of time, and allowing students to access data easily and quickly.

3 Users of the System

University Management System consists of four users. These users are administrators, school administrators, teachers and students. Each user effect the system in a way.

3.1 Administrator

Administrators manage the whole system by adding or removing schools and it's administrators.

3.2 School Administrator

School administrators manage their universities system only by adding or removing teachers, courses, classrooms etc.

3.3 Teacher

Teachers interact mostly with students and their needs. Teachers give notes, homeworks to students and see their schedules.

3.4 Students

Students are the main users of the system since they can interact the system in a various ways like seeing their grades, schedules, calculating their grade point average etc. Detailed requirements is given at section 4, requirements in details.

4 Requirements in Details

4.1 Functional Requirements

The system will be used for administrator, school administrator, teacher and student. All users will enter the system with their username and passwords. Administrator,

- Must add/remove university.
- Must add/remove university administrator.
- Must search university.
- Must search school Administrator.

School Administrator,

- Must see all teachers in university.
- Must search course.
  - Assign classroom to course.
  - Assign teacher to course
- Must add/remove course.
- Must change course informations.
- Must add/remove teachers.
- Must add/remove Classrooms.
- Must change classroom informations.

Teacher,

- Must search students.
  - Enter student grade.
- Must add/remove message to course.
- Must add/remove homework.
  - Update homeworks.
- Must add/remove student grade.

Student,

- Must search course.
  - Register to course.
- Must see grade.
- Must see homeworks.
- Must see messages.
- Must calculate GPA.
- Must add task to calendar.

4.2 Non - Functional Requirements

4.2.1 Usability Requirements

The system is useful because users will perform their operations with an interactive menu, they can select the option they want from the menu and proceed and username/password mechanism should be useful for user.

4.2.2 Security Requirements

All passwords must be hard to guess and ideally require upper/lower case letters and special symbols to ensure high security also only admins can access this information.

4.2.3 Performance Requirements

The data in the system will be dynamic and this data will expand as users log in. At the same time, the algorithms we use will improve the performance.

4.2.4 Space Requirements

The data in the system will be dynamic and user information will be in this data such as name, surname, school number, notes. Because of these, an expandable space is required.

4.2.5 Operational Requirements

The system must be in communication with the users in order to perform the necessary operations. User data and interaction is one of the most basic requirements in this system.

4.2.6 Environmental Requirements

The program is designed for the computer and will be written in a way to run on a computer terminal.

4.2.7 Learnability Requirements

When users see the interface, they can understand and implement the main actions.

4.2.8 Accounting Requirements

The system must have administrator. Administrators can confirm student teacher and school administrators to register in the system. The operations to be performed by these accounts are determined by the administrators.

4.2.9 Development Requirements

This system will be developed by many developers at the same time, and the development environment must be suitable for it.

4.2.10 Manageability Requirements

When editing the code for student teacher or school administrator, the rest of the part stays up and running.

4.3 User Requirements

We provide a system for students, teachers and school administrators in universities to perform their tasks at school. This system will be used for universities and in this system, students can view their grades, their homework, courses. Teachers can see their assignments, students, student grades, assign assignments. School administrators can view and make changes on courses, students, courses and lecturers. The administrator can see universities and make changes on them. The tasks are found in more detail in the use case diagram.

5 Use-Case Diagrams

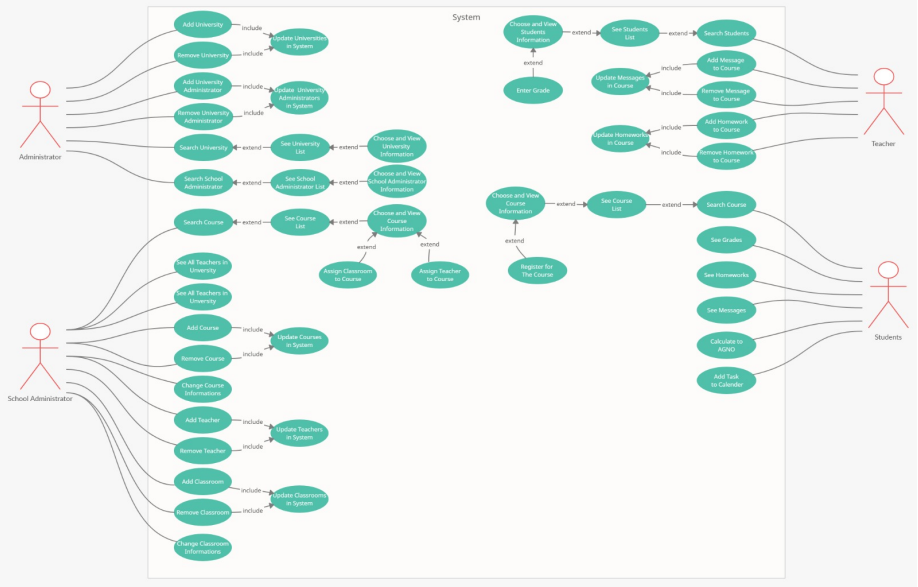

Figure 1: Use Case Diagram

6 C4 Model of the System

6.1 System Context Diagram

Projects system can be seen in a box in the centre, surrounded by its users and the other system that it interacts wtih.

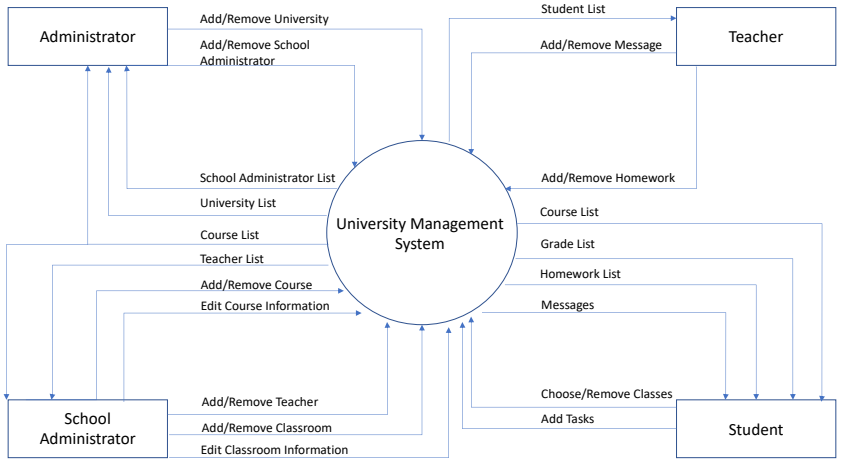

Figure 2: System Context Diagram

6.2 Container Diagram

Container diagram had changed due to feedback. Now the users are seperated from the API, they only see the console application. Container diagram shows the high-level shape of the software architecture and how responsibilities are distributed across it.

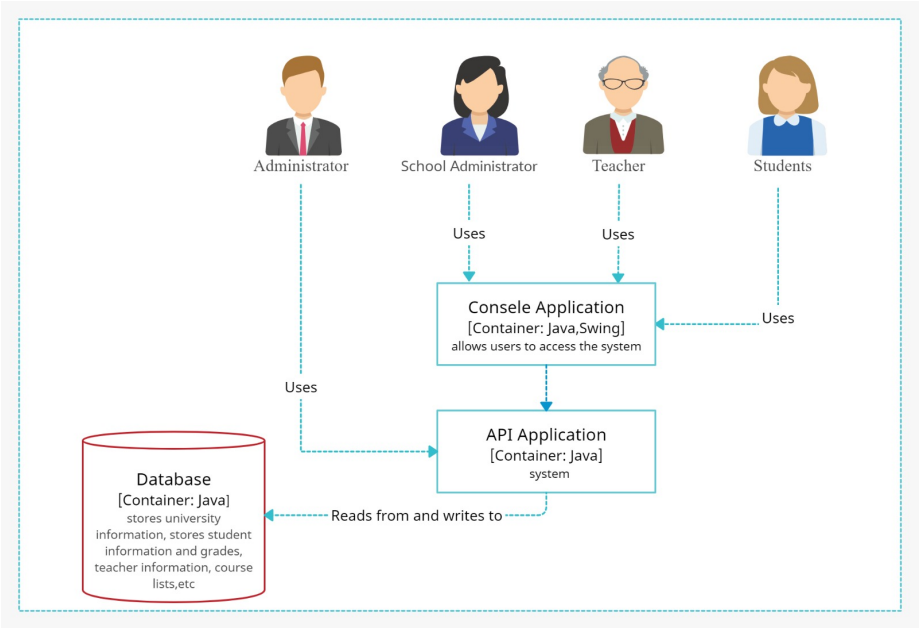

Figure 3: Container Diagram

6.3 Component Diagram

A component diagram breaks down the actual system under development into various high levels of functionality. A copy of the below diagram can be found on sent folder.

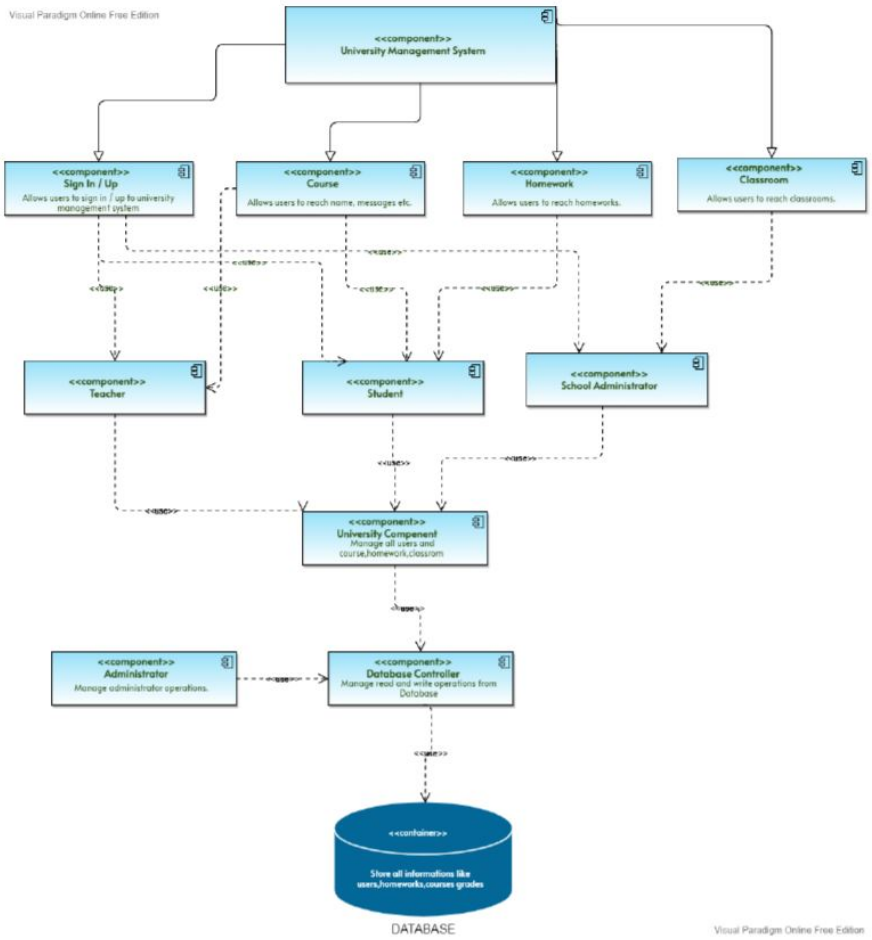

Figure 4: Component Diagram

Using of Data Structures

6.3.1 ArrayList

We used ArrayList to store university, courses, homework, and classrooms. Because we can add these classes to the end of the array. Once you add them, these classes will not be removed much and there is not much need to do sort. So we think the most useful use is arrayList.

6.3.2 Stack

We kept the declarations in a stack structure so that the last notification appears first in the notifications. We also kept the messages in the Course in a stack structure. Thus, we will be able to see the last sent message in the first place.

6.3.3 Queue

While students register to the system, their requests must be approved by the school Administrator. We kept these requests in a queue structure so that the first request appears first

6.3.4 BinarySearchTree

We kept all users in the BST structure. Because it will be easier to search for users in the database while logging into the system. And users will be added sequentially as they are added to the database.

6.3.5 PriorityQueue

We kept homeworks in PriorityQueue structure. In this way, we will be able to keep the assignments in order according to the deadline date. We think PriorityQueue is the most useful structure for keeping these homeworks organized according to deadlines.

7 Class Diagrams

A copy of the below diagram can be found on sent folder.

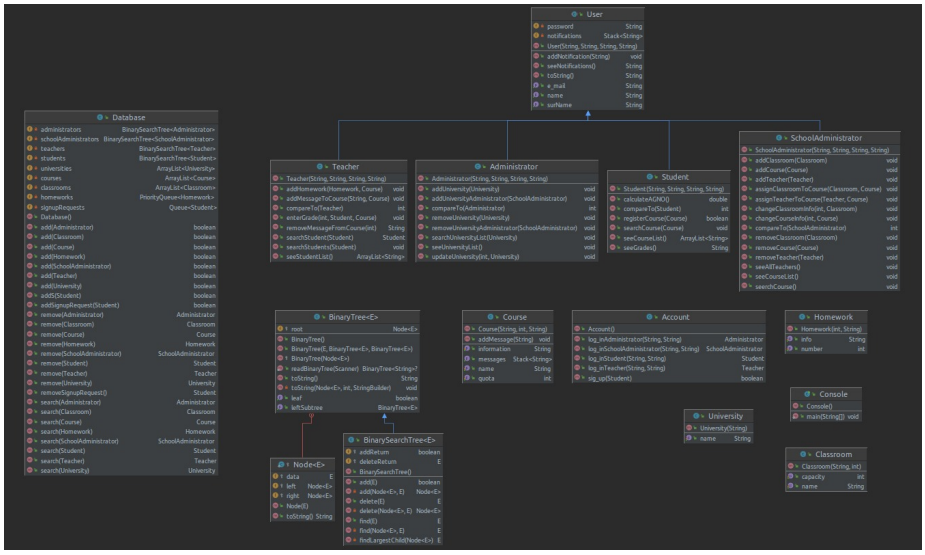

Figure 5: Class Diagram

8 Sequence Diagrams

A copy of the below diagram can be found on sent folder.

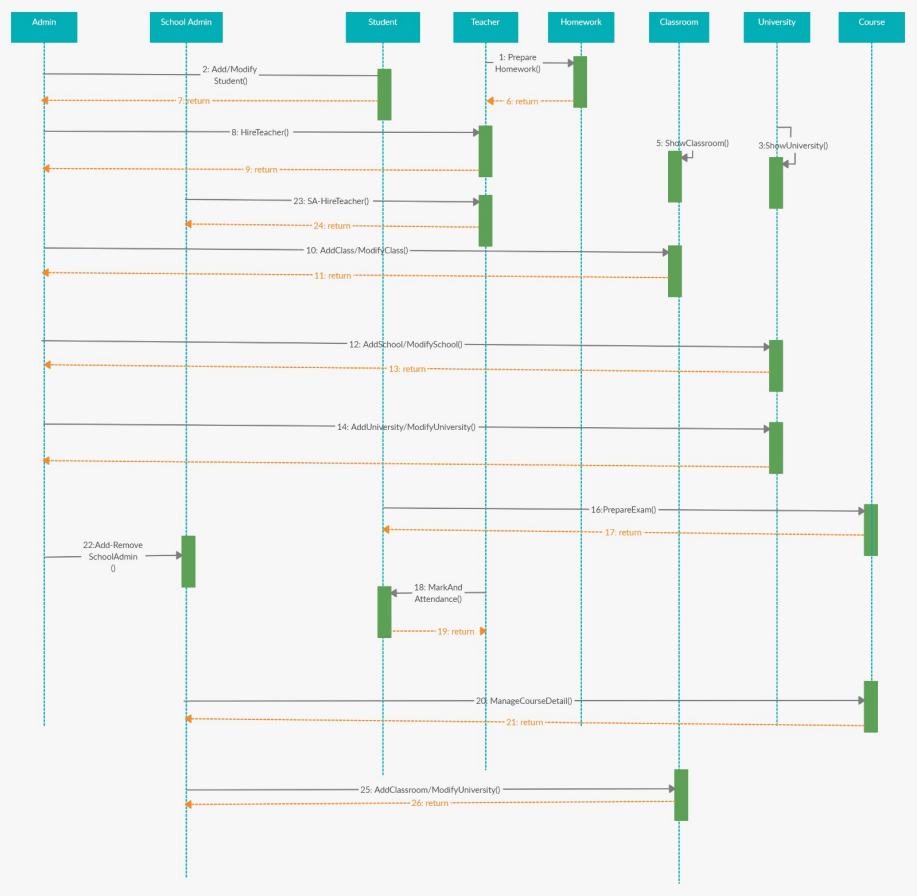

Figure 6: Sequence Diagram

9 Non-trivial Implementation Details

9.1 Administrator

- Administrator(String, String, String, String): We will use the given info to construct a Administrator Object.
- addUniversity(University): Before adding a university, a university should already be created with it's constructor method. And then adding the already created University makes more sense.
- addSchoolAdministrator(SchoolAdministrator): Already created SchoolAdministrator instance will probably be sent to Database through API.
- removeUniversity(University): Given University is sent to API. If the given University is on the Universities list it will be removed from the list.
- searchUniversityList(University): Given university is sent to the API, if is on the list, a positive integer must be sent back.
- seeUniversity(): This function will want the Universities from API, All universities will be seen on screen.

9.2 Account

- Account(): Default constructor does nothing.
- *login...*: Whenever users want to log in, they enter their email and password into the console, login functions of the account class take that email and password as parameters, and thanks to the search functions of the database, that person returns their object if they are present in the system.
- *signup*(Student): The only user who can directly sign into the system is a student. Student enters his information in Console. Then console give these information to sign-up method of Account class, this method creates the request in the database using the database method.

9.3 Teacher

- Teacher(String,String,String,String) :While creating the constructor, we can create it by entering the teacher's information.
- addHomework(Homework,Course):Each assignment is registered to the Homework class, first of all the homework class and the course class assigned to the assignment are sent as parameters.

- addMessageToCourse(): The course, on which the message will be given, is sent as a parameter. Students can see the message when they enter the course.
- enterGrade(int ,Student, Course):The student and the lesson he / she took are sent as parameters.so that the student can understand her-his grade without confusing the lectures.
- removeMessageFromCourse(int):it deletes the message by sending the message number as a parameter.
- searchStudent(Student):By sending the student as a parameter, it can view the student's information.
- seeStudentList():the teacher can view all of her/his students.

9.4 Student

- Student() : Constructor
- CalculateAGNO() : It calculates AGNO according to student's taken credits and grades.
- compareTo() : It receives student and compare the student with another student
- registerCourse() : It receives course and adds to student's courses Database.
- searchCourse(): It receives course and look in Database if the student takes the course.
- seeCourseList(): It returns all the courses that student takes in a list.
- seeGrades(): It returns the grades of the taken courses.

9.5 SchoolAdministrator

- SchoolAdministrator(String, String, String, String): While creating the constructor, we can create it by entering the school administrator's information.
- addClassroom(Classroom): Adds Classroom object to the system using the add (Classroom) method of database.
- addCourse(Course): Adds Course object to the system using the add (Course) method of database.
- addTeacher(Teacher): Adds Teacher object to the system using the add (Teacher) method of database.

- assignClassroomToCourse(Classroom, Course): Assign classroom in which it will be held to course.
- assignTeacherToCourse(Teacher, Course): Assigns the teacher who will give the course to the course.
- changeClassroomInfo(int, Classroom): Assigns the information of the new Classroom object to the classroom in index.
- compareTo(SchoolAdministrator): It compares itself with the object taken as a parameter.
- removeClassroom(Classroom): Using the remove(Classroom) mehod of the database, it removes the Classrom object from the system.
- removeCourse(Course): Using the remove(Course) mehod of the database, it removes the Course object from the system
- removeTeacher(Teacher): Using the remove(Teacher) mehod of the database, it removes the Teacher object from the system.
- seeAllTeacher(): Returns the list of teachers in the school.
- seeCourseList(): Returns the list of courses in the school.

9.6 Database

- Database(): Constructor that crates a Database for the system
- add(Adminstrator) : Adds an Adminstrator object , which is sent as a parameter, to the Database
- add(Classroom) : Adds a Classroom object , which is sent as a parameter, to the Database
- add(Course) : Adds a Course object , which is sent as a parameter, to the Database
- add(Homework): Adds a Homework object , which is sent as a parameter, to the Database
- add(SchoolAdministrator) : Adds a SchoolAdminstrator object , which is sent as a parameter, to the Database
- add(Teacher): Adds a Teacher object , which is sent as a parameter, to the Database
- add(University) : Adds an University object , which is sent as a parameter, to the Database
- addS(Student) : Adds a Student object , which is sent as a parameter, to the Database

- addSignupRequest(Student) : Adds a Student object , which is sent as a parameter, to the Database
- remove(Adminstrator) : Removes an Adminstrator object , which is sent as a parameter, from the Database
- remove(Classroom) : Removes a Classroom object , which is sent as a parameter, from the Database
- remove(Course) : Removes a Course object , which is sent as a parameter, from the Database
- remove(Homework): Removes a Homework object , which is sent as a parameter, from the Database
- remove(SchoolAdministrator) : Removes a SchoolAdminstrator object , which is sent as a parameter, from the Database
- remove(Teacher): Removes a Teacher object , which is sent as a parameter, from the Database
- remove(University) : Removes an University object , which is sent as a parameter, from the Database
- remove(Student) : Removes a Student object , which is sent as a parameter ,from the Database
- removeSignupRequest(Student) : Removes a Student object , which is sent as a parameter, from the Database
- search (Adminstrator) : Searchs an Adminstrator object , which is sent as a parameter, into the Database
- search(Classroom) : Searchs a Classroom object , which is sent as a parameter, into the Database
- search(Course) : Searchs a Course object , which is sent as a parameter, into the Database
- search(Homework): Searchs a Homework object , which is sent as a parameter, into the Database
- search(SchoolAdministrator) : Searchs a SchoolAdminstrator object , which is sent as a parameter, into the Database
- search(Teacher): Searchs a Teacher object , which is sent as a parameter, into the Database
- search(University) : Searchs an University object , which is sent as a parameter, into the Database
- searchS(Student) : Searchs a Student object , which is sent as a parameter, into the Database

10 Test Cases

A copy of the below test cases table can be found on sent folder.

Test Case ID	Test Case Description	Expected Result
1	Admin adds university.	Given university should be added to DB.
2	Admin adds pre-existing university.	Given university should not be added to DB.
3	Admin removes university.	Given university gets removed from the DB
4	Admin removes non-existing university.	There should be no changes on DB.
5	Admin searches universities.	All universities that are on the DB should appear on the screen.
6	Admin searches school administrators.	All school administrators that are on the DB should appear on the screen.
7	School administrator searches teachers in a given university.	All teachers on a given university that are on the DB should appear on the screen.
8	School administrator searches courses in a given university.	All courses on a given university that are on the DB should appear on the screen.
9	School administrator assigns classroom to a course.	Courses classroom on the DB should be updated.
10	School administrator assigns teacher to a course.	Courses teacher on the DB should be updated.
11	School administrator adds course.	Given course should be added to DB.
12	School administrator adds pre-existing course.	Given course should not be added to DB.
13	School administrator removes course.	Given course gets removed from the DB
14	School administrator removes non-existing course.	There should be no changes on DB.
15	School administrator changes course information	Course information in the DB should be updated
16	School administrator adds teacher.	Given teacher should be added to DB.
17	School administrator adds pre-existing teacher.	Given teacher should not be added to DB.
18	School administrator removes teacher.	Given teacher gets removed from the DB
19	School administrator removes non-existing teacher.	There should be no changes on DB.
20	School administrator adds classroom.	Given classroom should be added to DB.
21	School administrator adds pre-existing classroom.	Given classroom should not be added to DB.
22	School administrator removes classroom.	Given classroom gets removed from the DB
23	School administrator removes non-existing classroom.	There should be no changes on DB.
24	School administrator changes. classroom information information	Classroom information in the DB should be updated
25	Teacher searches students.	All students in the teacher's school should appear on the screen.
26	Teacher enters students grade on a given subject.	The students grade on the given subject is added to the DB.
27	Teacher adds a message to course.	All students who takes the given course should get the message. Message should be added to the messages of the students in the DB.
28	Teacher removes a message from a course.	Message should be removed from the messages of the students in the DB.
29	Teacher adds homework.	All students who takes the given course should get the new Homework. Homework should be added to the homeworks of the students in the DB.
30	Teacher removes homework.	Homeworks should be removed from the homeworks of the students in the DB.
31	Student searches courses.	All courses in the university which the student is enrolled in should be shown to the student.
32	Student looks at grades	All grades that are given by teachers should be shown to the student.
33	Student looks at messages.	All messages that the student recieved should be shown to the student.
34	Student gives a number bigger than 4 or lower than 0 while giving grades at calculating the GPA	Student should be warned about the boundaries of the grades.
35	Student gives possible grades for GPA calculation	GPA is calculated and shown to the student.
36	Student adds a task to calendar.	Students calendar is updated on the DB.

Table 1: