**SOFTWARE ENGINEERING**

**Term Project**

**SMART BUILDING SYSTEM FOR UNIVERSITIES (SBSU)**

Fourth Iteration: Design Document

Group No: 12

05-11-02 Ahmet GÜL

05-11-44 Onurhan ÇELİK

05-11-86 Tuğba KARABULUT

05-11-93 Gökcan ÜSTÜN

Submission Date: 24.12.2015

Table of Contents

[1. Introduction 2](#_Toc438669352)

[2. System Sequence Diagram of each Use Case 2](#_Toc438669353)

[3. Sequence Diagram of Main Use Cases 8](#_Toc438669354)

[4. Class Diagram 12](#_Toc438669355)

[5. Glossary 13](#_Toc438669356)

[6. Implementation of a main use case 13](#_Toc438669357)

[7. Tests of the implemented use case 16](#_Toc438669358)

[8. Requirements validation with the domain expert 17](#_Toc438669359)

[9. Updated Requirements Document 18](#_Toc438669360)

[10. What Did We Consider During Design Of User Interface? 20](#_Toc438669361)

[11. Conclusion 21](#_Toc438669362)

# Introduction

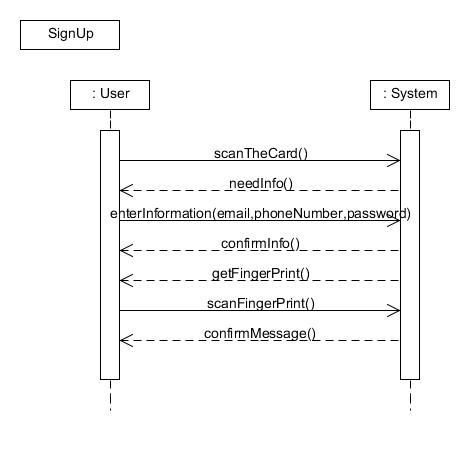
In this report, we will write design document of our project. This report will include System Sequence Diagrams (SSD), Sequence Diagrams (SD), Class Diagram and Implementation of a main use case. By using the use cases that we created, we have to design SDs, SSDs and class diagram before implementation.

Also, we will write test cases of the implemented use cases. We will meet with domain experts to discuss requirements validation. After meeting, we have to update our requirements document.

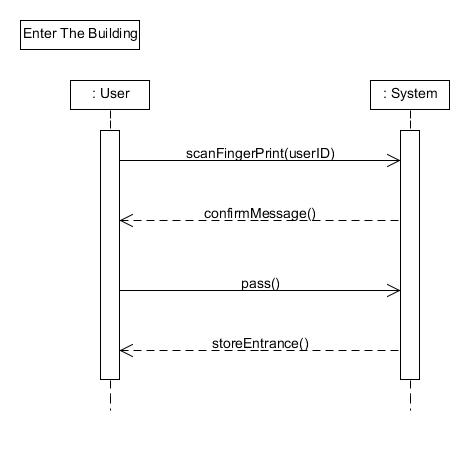
While we are implementing, we will consider some cases in User Interface Design. Furthermore, this report includes glossary that contain our terms.

# System Sequence Diagram of each Use Case

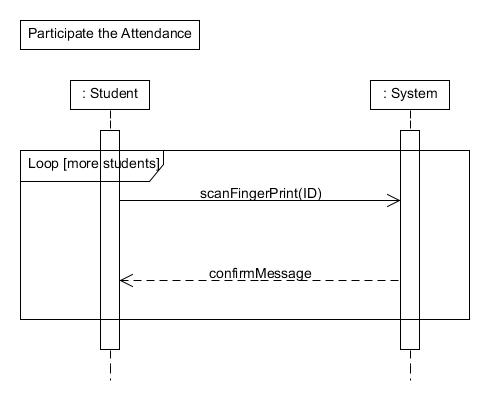
* **Sign Up SSD**

****

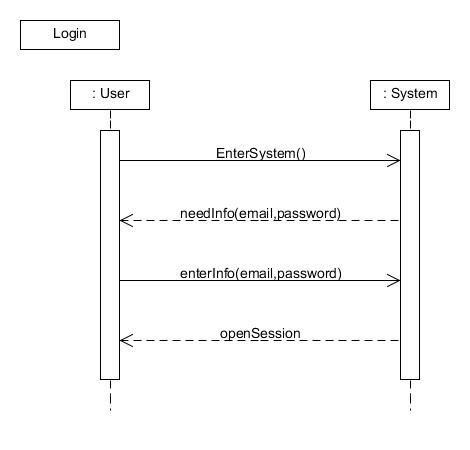
* **Enter the Building**

****

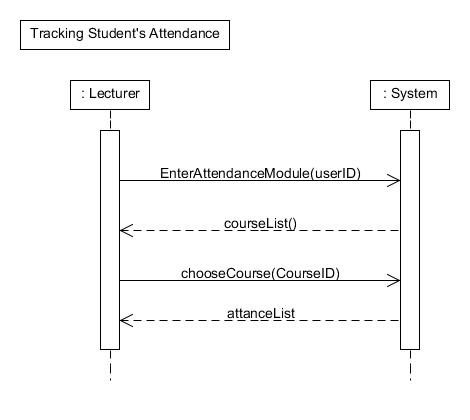
* **Participate the Attendance**

****

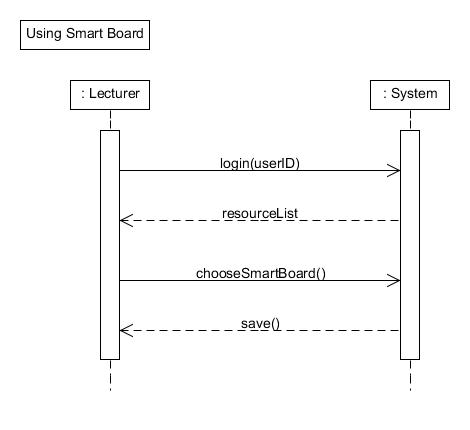
* **Login**

****

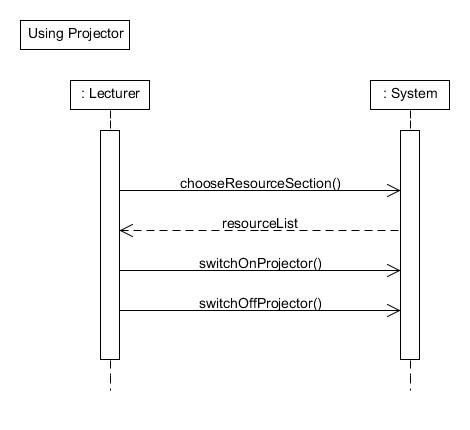
* **Tracking Student’s Attendance**

****

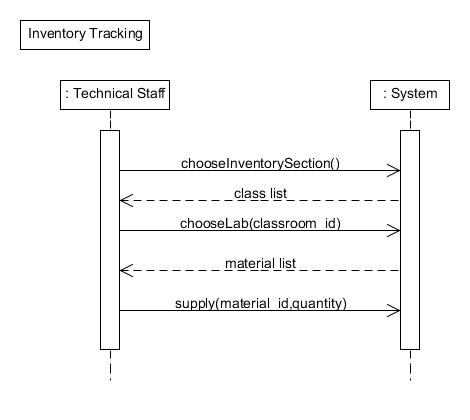
* **Using Smart Board**

****

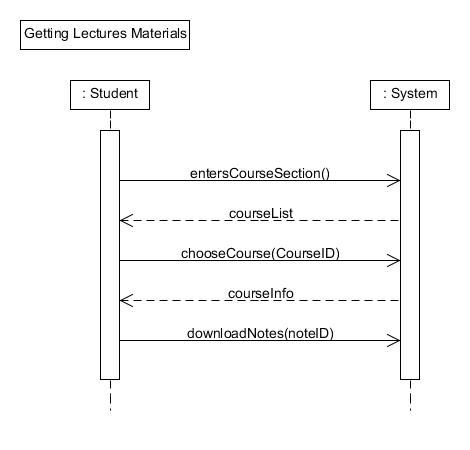
* **Using Projector**

****

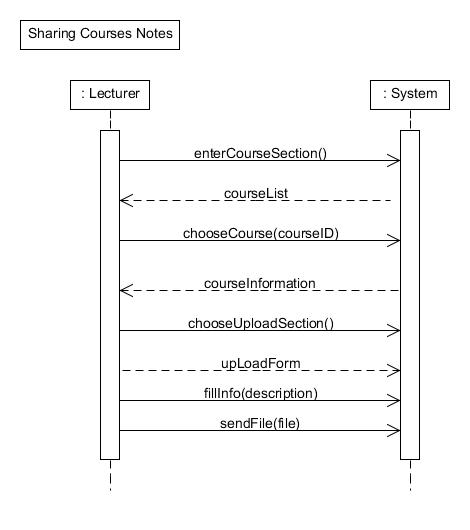
* **Inventory Tracking**

****

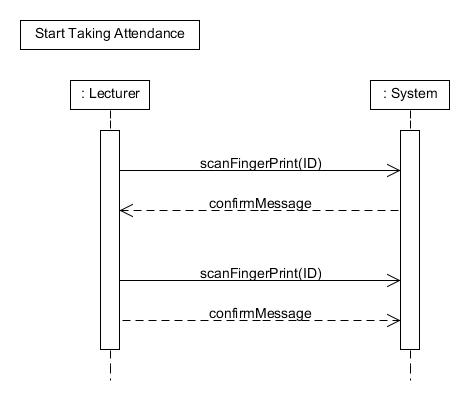
* **Getting Lectures Materials**

****

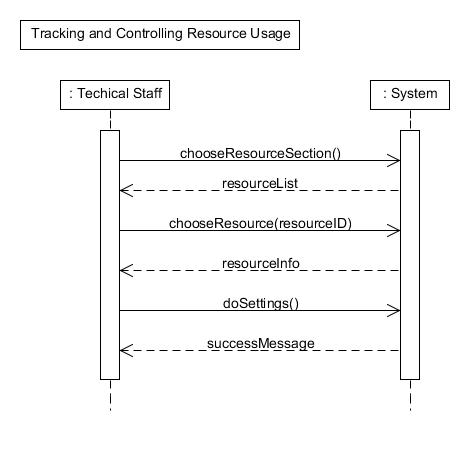
* **Sharing Courses Notes**

****

* **Start Taking Attendance**

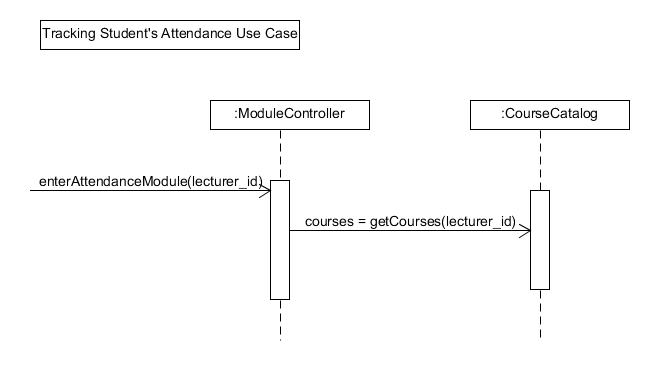
****

* **Tracking and Controlling Resource Usage**

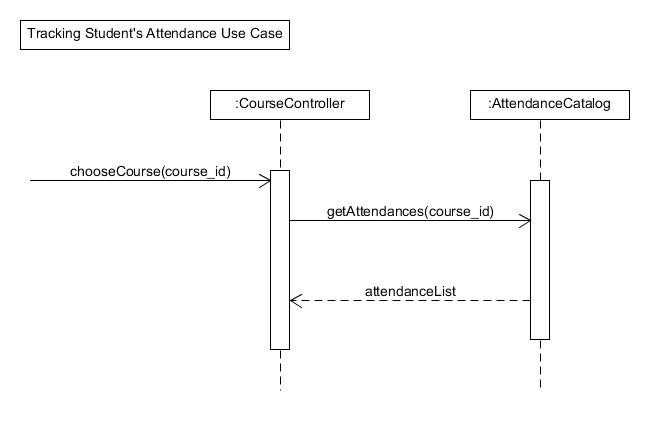
****

# Sequence Diagram of Main Use Cases

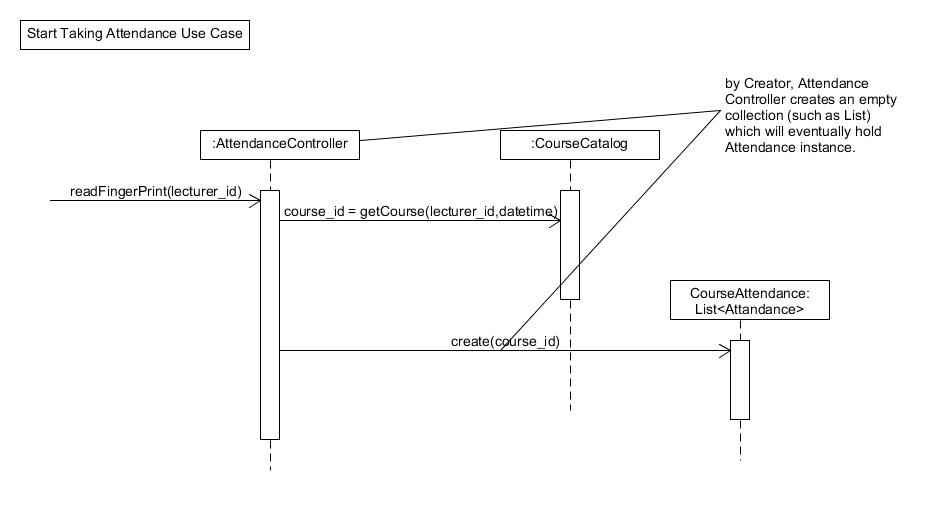
* **enterAttandanceModule SD**

****

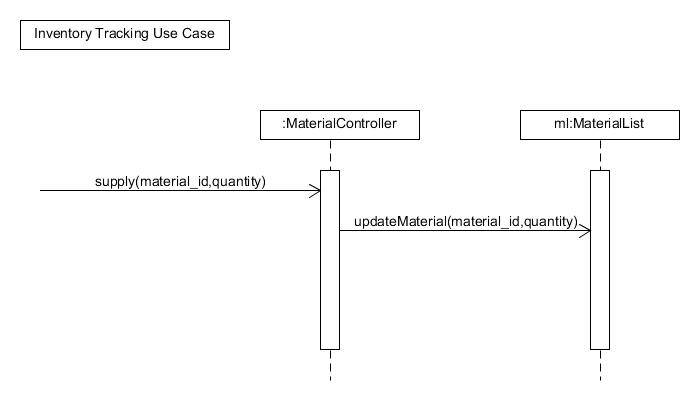
* **chooseCourse SD**

****

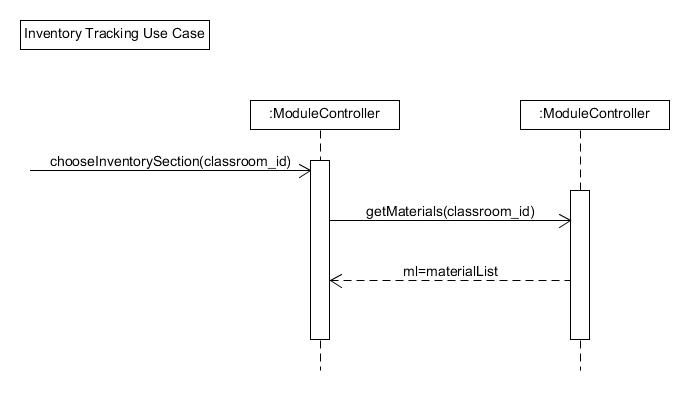
* **readFingerPrint SD**

****

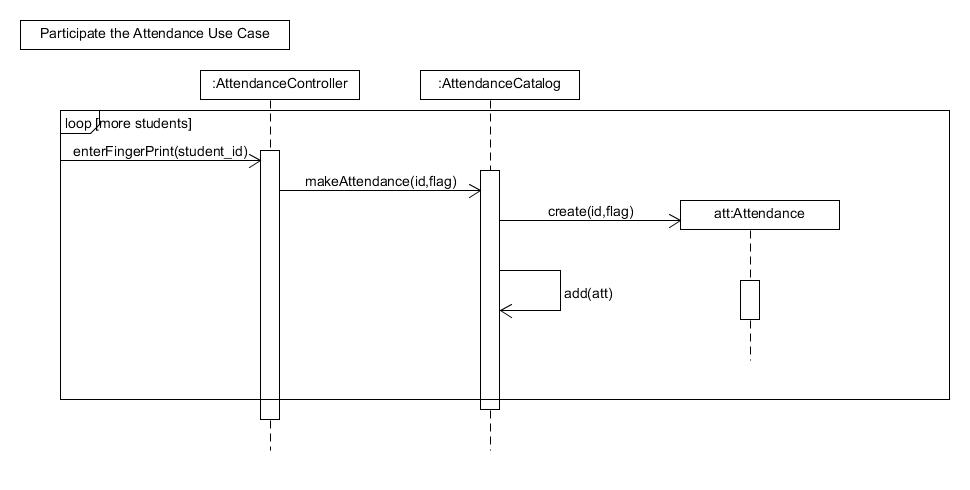
* **supply SD**

****

* **chooseInventorySection SD**

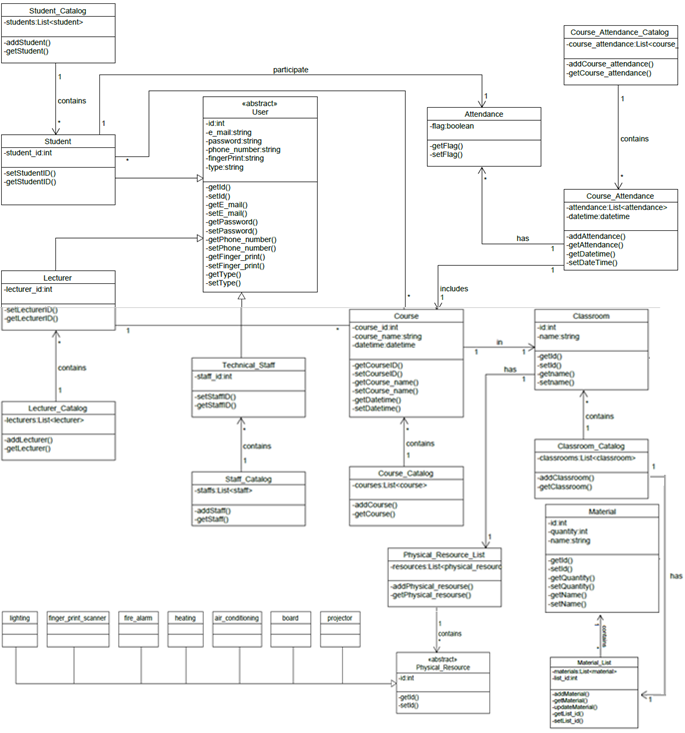
****

* **enterFingerPrint SD**

****

# Class Diagram

There is a view of the class diagram below figure. You can open “Class Diagram.pdf” file to see readable way class diagram.



**Figure 1:** Class Diagram

# Glossary

* **Scan:** Convert (a document or picture) into digital form for storage or processing on a computer.
* **Resource:** A stock or supply of materials, staff, and other assets that can be drawn on by a person or organization in order to function effectively.
* **User:** someone who uses system, such as lecturer, student, technical staff.
* **Material:** a solid substance from which things can be made.
* **Module:** Each of a set of standardized parts or independent units that can be used to construct a more complex structure, such as an item of furniture or a building.
* **Staff:** the people who work for an organization.

# Implementation of a main use case

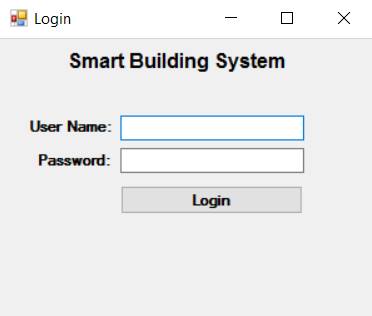
We implemented of our main use case and user interfaces of all use cases. We used **C#** programming language to implement and our IDE was **Microsoft Visual Studio 2015**. Also we used **MS SQL Server** to store data. You can find project files and database file in “*Group12\_SBSU\_SourceCodes.rar*” file.

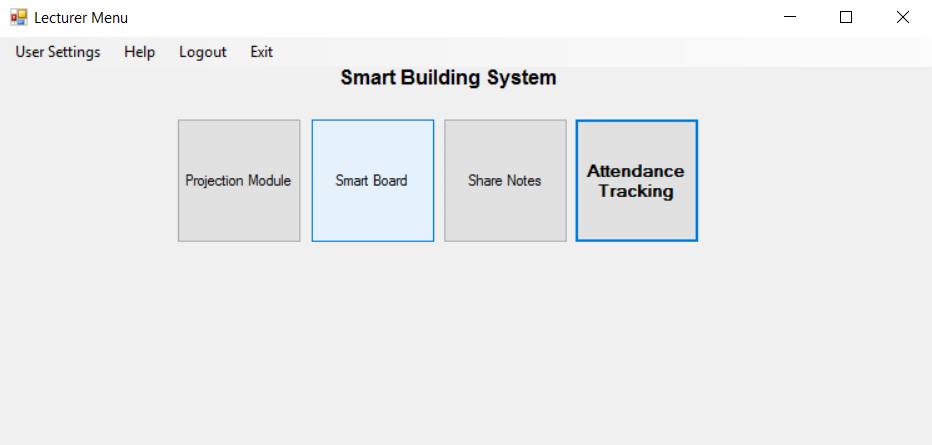
There are some example data to login the system in Table 1.

**Table 1:** Login data examples

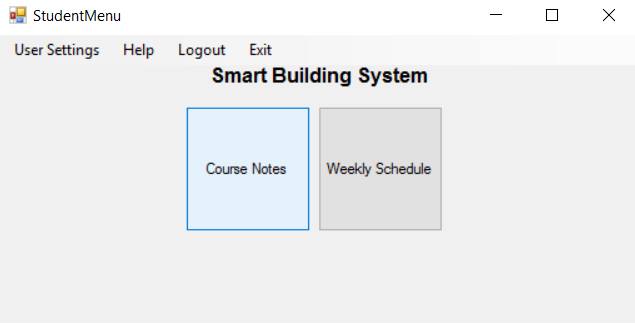
|  |  |  |
| --- | --- | --- |
| Role | E-mail(username) | Password |
| Student | [gokcan@gmail.com](mailto:gokcan@gmail.com) | 123456 |
| Lecturer | yasemin.topaloglu@ege.edu.tr | 123456 |
| Lecturer | ozgur.gumus@ege.edu.tr | 123456 |
| Technical Staff | [okan@gmail.com](mailto:okan@gmail.com) | 123456 |

You must be login to use Smart Building System. There are three different roles in our system. Each of these roles has their specific menu. In the below figures, you can see login and menu screens.

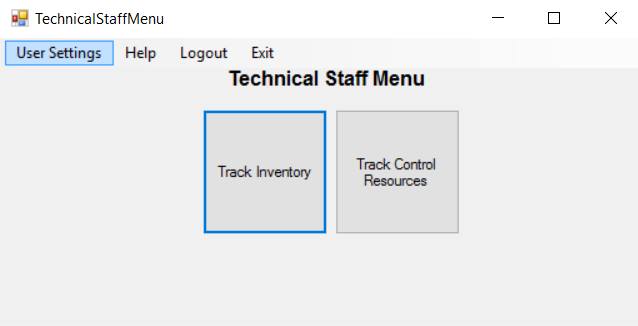
  
**Figure 2:** Login Screen



**Figure 3:** Lecturer Menu

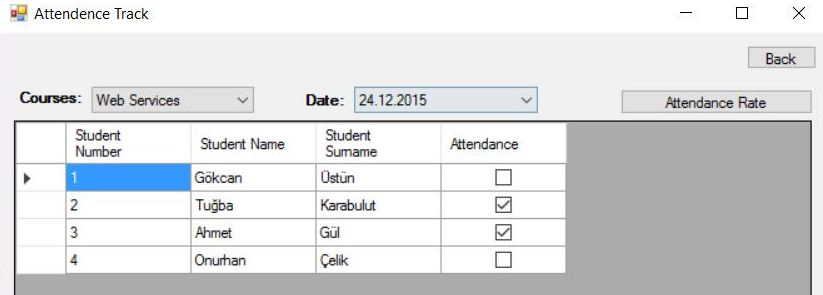


**Figure 4:** Student Menu

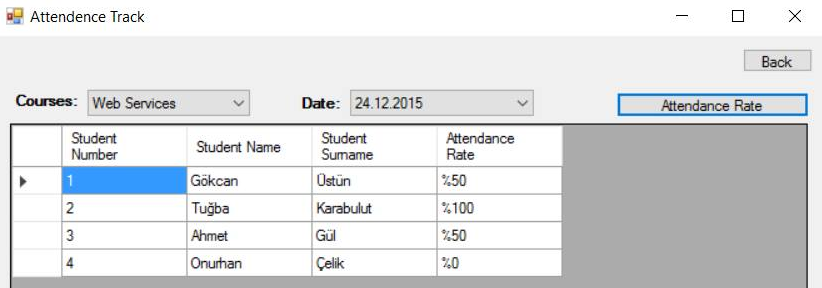
**Figure 5:** Technical Staff Menu

We made all user interfaces of each use cases. But we implemented only Attendance Tracking module of Lecturer Menu. Normally, you must take attendance to track attendances. Taking attendance in our system is done with fingerprint machines. Data incomes database with sensors through local network. We created database and added examples of attendance data manually.

When lecturer entered the “Attendance Tracking Module” then below screens has appeared. Lecturer can track attendances in two different ways. One is weekly attendance, another one is attendance rate of all weeks. You can see these tracking ways in the below figures.

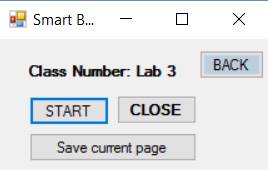


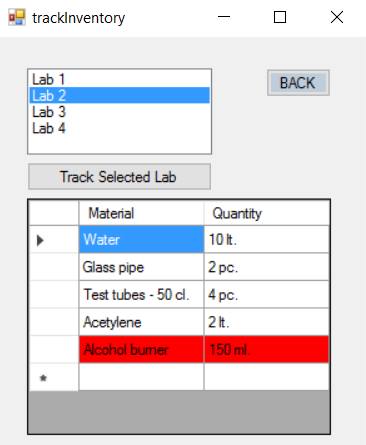
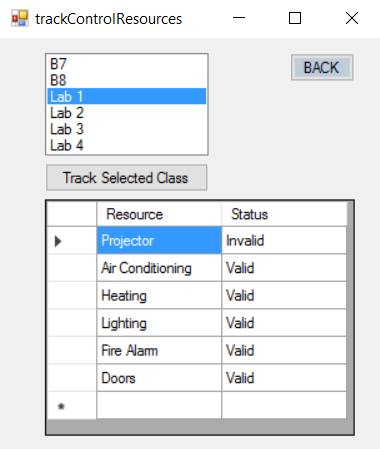
**Figure 6:** Tracking Attendance Weekly



**Figure 7:** Tracking Attendance Rate

Other modules (e.g. Projection, Smart board, Tracking Inventory, Tracking and Controlling Resources) do not have any function, because they were not implemented. But you can see their interface views in the below figures with example data.

   
**Figure 8:** Projection Module **Figure 9:** Smart Board Module

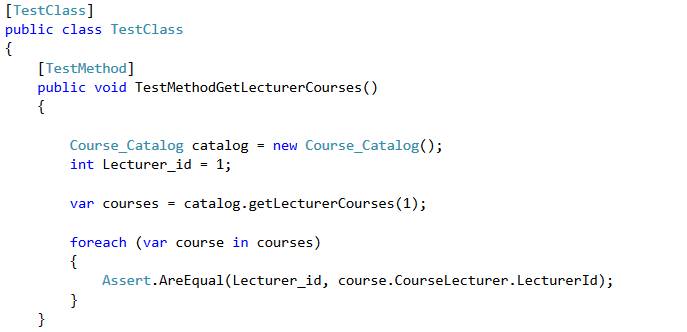
 

**Figure 10:** Tracking Inventory Module **Figure 11:** Tracking & Controlling Resources

# Tests of the implemented use case

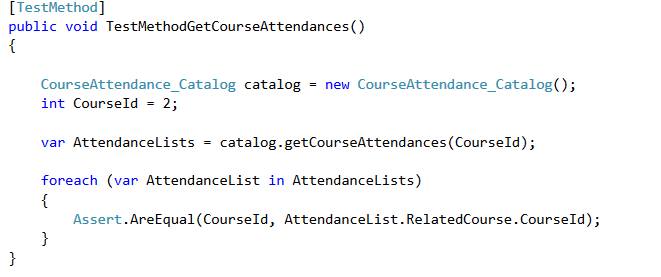
We tested our implemented use case with unit testing technique. We have two important methods in main use case. So, we tested these methods.

When lecturer enters tracking attendance module, his/her courses are brought. To do this, GetLecturerCourses method is called. This method takes parameter that is lecturer id. We tested this method in Figure 1. We know that in the database, we have created example variable in setup stage. Then, we called method with the variable, results were compared with example variable in verify stage.



**Figure 11:** Test of “GetLecturerCourses” method

The second test was performed on GetCourseAttendances method. We setup variable that is CourseId and called the method. In verify stage, we asserted results.



**Figure 12:** Test of “GetCourseAttendances” method

# Requirements validation with the domain expert

* **Requirements validation with person x**
* There is air-conditioning in the server’s room and also all rooms. These air-conditionings are in our system. They can be managed and tracked with system.
* There is a fingerprint scanner in all classrooms to take attendance.
* There is a turnstile machine to enter building.
* **Requirements validation with person y**
* There are heating, lighting, air-conditioning resources in the classrooms. They can be automatically managed and tracked by system.
* There is a smart board in the classroom. So, it can be cleaned easily.
* Technical staff tracks inventory of labs.
* Other domain experts’ requirements are the same as above. We already choose requirements in the negotiation phase.

# Updated Requirements Document

We made some changes on use cases that were made in the requirement analysis phase. These changing use cases are in the below.

* **Use Case: Start Taking Attendance**

**Scope:** SBSU

**Primary Actor:** Lecturer

**Precondition:** There must be a fingerprint scanner in the class.

**Postcondition:** Make attendance started.

**Stakeholder and Interests:**

Lecturer: Lecturer wants to start taking attendance.

**Main Success Scenario:**

* Lecturer push button on the portable fingerprint scanner in order to start machine.
* System wants to fingerprint from lecturer.
* Lecturer puts his/her fingerprint to scanner.
* System starts attendance.

**Extensions:**

* Fingerprint scanner can be crashed.
* **Use Case: Sharing Courses Notes**

**Scope:** SBSU

**Primary Actor:** Lecturer

**Precondition:** Lecturers must be logged in.

**Postcondition:** Lecturer uploads materials to system.

**Stakeholder and Interests:**

Lecturer: Lecturer wants to upload.

Student: Students can access materials or notes.

**Main Success Scenario:**

* Lecturer enters share notes section.
* System open new form to upload.
* Lecturer chooses course from course list.
* Lecturer fills the form.
* Lecturer selects files from local computer and fills the information that s/he wants.
* Lecturer uploads the files.
* System saves the files.

**Extensions:**

* System can be failed.
* **Use Case: Tracking Student’s Attendance**

**Scope:** SBSU

**Primary Actor:** Lecturer

**Precondition:** Lecturer must login system correctly.

**Postcondition:** Lecturer sees student’s attendance.

**Stakeholder and Interests:**

Lecturer: Lecturer wants to see students’ attendance.

**Main Success Scenario:**

* Lecturers enter lesson attendance module.
* System shows lecturer’s lessons.
* Lecturer chooses one of the courses that want to track.
* System shows student list with attendance of the chosen course.

**Extensions:**

* System can be failed.
* **Use Case: Participate the Attendance**

**Scope:** SBSU

**Primary Actor:** Student

**Precondition:** Student must be enrolled to system.

Lecturer must start taking attendance.

**Postcondition:** System stores the information of student’s attendance to lesson.

**Stakeholders and Interest:**

Lecturer: Lecturer wants to take attendance.

Student: Student wants to show his/her attendance to system.

**Main Scenarios:**

* Student puts his/her finger to fingerprint scanner.
* System stores student’s attendance of current lesson in that class.
* System switches on the green light on the fingerprint scanner machine.
* Student gives machine to next student.

**Extensions:**

* System can be failed.
* **Use Case: Using Smart Board**

**Scope:** SBSU

**Primary Actor:** Lecturer

**Precondition:** There must be a smart board in the class.

Lecturers must be logged in.

**Postcondition:** Lecturer uses board successfully.

**Stakeholder and Interests:**

Lecturer: Lecturer wants to use smart board.

Student: Students want to get what have written on the board.

**Main Success Scenario:**

* Lecturer starts the smart board.
* Lecturer uses smart board during the session.
* Lecturer saves to share what have written on the board.
* System saves what lecturer wants to share according to current course.

**Extensions:**

* System can be failed.
* Lecturer does not want to share everything on the board.
* **Use Case: Using Projector**

**Scope:** SBSU

**Primary Actor:** Lecturer

**Precondition:** There must be a projector in the class.

Lecturers must be logged in.

**Postcondition:** Lecturer switches on/off projector successfully.

**Stakeholder and Interests:**

Lecturer: Lecturer wants to use projector.

Student: Students want to see course notes.

**Main Success Scenario:**

* Lecturer starts the projector.
* Lecturer clicks projector button on a control panel to switch on projector.
* Lecturer clicks projector button on a control panel to switch off projector at the end of the course.

**Extensions**

* Projector can be failed.

# What Did We Consider During Design Of User Interface?

We gave great importance to design of user interfaces. Since, we reckon that user interfaces have a great impact on acceptability and usability of our software. Our principles that we consider to design interface are below-mentioned.

1. **Clear:** System was designed for user who have experience and who do not have any experience of using similar systems. Thus, we designed interface simple and understandable as much as it can be. Users feel relax while using the software.
2. **Efficient:** Users want to spend minimum energy while using the system. That’s why we designed with minimum button and name of the sections are quite easy to understand. Our interface allows users to perform functions on the system faster and with less effort. For example;while teacher is looking attendance interface, teacher can click to back to see modules and sharing his/her notes without logging system out.
3. **Attractive:** Most important modules are more attractive than other modules in our system in order to increase usability. On the other hand, attractive modules are not distracting for users.
4. **Visibility:** Users only see what they have to see while using the system. That’s why they can do their wishes without confusion.

# Conclusion

In the final report of our project, first of all, system sequence diagrams for all uses cases are created. Then, sequence diagrams are created for three use cases. Last of all, class diagram is designed for all system and started implementation with unit tests. Also, during the implementation, we consider different criteria for user interfaces. Finally, we added glossary words to make our report more understandable.