

Institute Automation System

Requirements Specification and Analysis

Version 2.0

20 April 2018

Oğuzhan Ulusoy

Prepared for  
CSE490 Thesis of Bachelor of Science



## Table of Contents

1.	Introduction.....	2
1.1.	Purpose of the System .....	2
1.2.	Scope of the System .....	2
1.3.	Objectives and Success Criteria of the Project .....	3
1.4.	Definitions, Acronyms, and Abbreviations .....	3
1.5.	Overview .....	3
2.	Current System.....	4
3.	Proposed System .....	4
3.1.	Overview .....	4
3.2.	Functional Requirements.....	4
3.3.	Nonfunctional Requirements.....	4
	Usability .....	5
	Reliability .....	5
	Performance.....	5
	Supportability .....	5
	Implementation .....	6
	Interface.....	6
	Packaging .....	7
	Legal.....	6
3.4.	System Models.....	5
	Scenarios.....	7
	Use case model .....	19
	Object model.....	19
	Dynamic model.....	19
	User interface—navigational paths and screen mock-ups .....	21
4.	Glossary .....	24
5.	References.....	24

## REQUIREMENTS ANALYSIS DOCUMENT[1]

### 1. Introduction

#### 1.1. Purpose of the System

Institute Automation System is a web-based application that provides a school interface to grad students, academic instructors and institute staff. The main goal of the mentioned interface is to satisfy to make easier and faster their activities.

#### 1.2. Scope of the System

As mentioned at Purpose of the System section, Institute Automation System is a web-based application that satisfies a school interface to actors in system. The main target of these interfaces is to provide to make easier and faster their activities. Whole project is related to management side of institute.

Institute Automation System has three major actors which are grad students, staffs and system administrator. Staffs are divided into two different groups that are academic staff and institute staff. Actually, Institute Automation System has five actors, individually. Therefore, Institute Automation System has different interfaces for each actor.

Each interface is going to be had many useful functions according to actor's role in institute. Main purposes are already existing in another system. We are going to be develop a system that contains these main purposes and also useful different functions.

The system is going to be included personal information (*briefly detailed*) for each actor in database. Also, there will be many database operations. Lectures are shown by departments with all information.

Grand student can register and pre-register for institute. Grand student can access their CCR, transcript, graduation time, weekly schedule, lecture lists, personal information, notifications. Also, they can create weekly schedule. They can write message to their academic supervisor or institute staff. They can display their scholarship and when it is ended. They can calculate their semester average. They can reach to each lecture's details.

Academic instructors are added on to the system by institute staff. Their personal information is stored in memory. Some of them are changeable. They can add their lectures with details. They can reach their class lists. They can reach some statistics. They can reach classes where lectures. They can display message from institute or grad students, also they can reply and write new. They can create push notification to all class. They can reach their student's CCR. They can accept or reject their student's weekly schedule, then that is transmitted on to the institute staff.

Institute staff can manage institute staff (*over role*), academic instructors and grad students. All secretarial and instituted tasks can be done. They can reach detailed lists (*i.e. scholarship statues, graduation statues, class lists, lecture lists, personal lists and so on.*) They can make report. They can accept or reject lectures that were added by academic instructors. They can accept or reject registrations that were done by grad students. They can accept weekly schedules that were done by grad students. They can reach to achieve.

### 1.3. Objectives and Success Criteria of the Project

We are going to improve a system that is more exclusive and capable than existing system. Some functions are mentioned at Scope of the System section are included on existing system. Whereas, some functions are mentioned at Scope of the System section are completely new and useful function. It should be more reliable, efficient and lossless data.

The implementation of Institute Automation System should be understandable, clear and efficient. The all needs should be planned to relevant domain. Security is another objective when we develop the system. The all information are kept protected in secure and the system should be guaranteed this.

The first success criteria of the project are to develop existing functions, then the second success criteria of the project are to develop completely new functions. Yet, the system must be faster running properly. The database will be complicated, but we must use two nested database operation maximums to prevent deadlock or decrease waiting time in queue.

### 1.4. Definitions, Acronyms, and Abbreviations

IAS: Institute Automation System that is going to be developed by us.

Academic personal: People on system that have lectures.

Grad students: People who are master and doctorate students.

Institute staff: People who are secretaries, manager and other.

Existing system: The system that has University of Işık.

*until now...*

### 1.5. Overview

- Rest of the Requirement Analysis Document (*RAD*) contains non-functional (*includes usability, reliability, performance, supportability, implementation, interface, operational, packaging and legal requirements*) and functional requirements (*includes high-level functionality of the system*).
- System models are given. Scenarios are in side of system models section (*that will be mentioned later with instances*). Scenarios are to tell us the details of functional requirements. Use case models, object model, dynamic model and user interface view (mockup) are parts of system models section.

## 2. Current System

There is an existing system that has University of Işık. That is to provide some function to academic instructors, institute staffs and grad students. We have observed Campus Online [2] and have analyzed it. Also, there are some similar automation systems of other universities. We have also observed and analyzed the system of University of Galatasaray[3] and the system of University of Sabancı[4].

## 3. Proposed System

Proposed system for Institute Automation System is to have existing systematic functions and completely new and useful functions. New functions are going to be introduced in Functional Requirements section. The main aim of us is to develop more reliable, faster, exclusive, capable and lossless data system.

### 3.1. Overview

The system is going to be designed according to academic staff, institute staff and grad students. Although we have said there are three major actors, implementation will be set up to sub-actors. System administrator has a chance to use default Django administrator panel. System administrator is manager of all system, so she has full authentication. Functional Requirements section contains what any actor will be able to do. Nonfunctional requirements section contains system details, interface details and implementation details.

### 3.2. Functional Requirements

Functional requirements for institute staff are given in below:

1. Institute staff shall be able to login and logout.
2. Institute staff shall be able to display institutes, departments, programs, curriculums, courses, available courses, course types and sections with own details on Institute section. (Essential)
3. Institute staff shall be able to define a new course. (Essential)
4. Institute staff shall be able to open or close an existing source. (Essential)
5. Institute staff shall be able to define a section for a course. (Optional)
6. Institute staff shall be able to display all academic staff, institute heads, department heads, program heads and quota managers with own details on Academic Staff section. (Optional)
7. Institute staff shall be able to define a new academic staff. (Optional)
8. Institute staff shall be able to define a new quota manager for a department. (Optional)
9. Institute staff shall be able to display all institute staffs with own details. (Optional)
10. Institute staff shall be able to define an institute staff. (Optional)
11. Institute staff shall be able to display all grad students, applications with own details on Grad Student section. (Optional)
12. Institute staff shall be able to accept or reject an application. (Essential)

13. Institute staff shall be able to display all completed courses, all curriculums, all CCRs, all transcripts with own details. (Optional)
14. Institute staff shall be able to withdraw for a student. (Essential)
15. Institute staff shall be reach to statistics. (Optional)

### 3.3. Nonfunctional Requirements

#### Usability

Institute Automation System should have three different interfaces for all sub-actors. All functions are divided in to the capabilities of actors. Each division should represent an interface. By the side of Institute Staff, she is able to do in activities in mentioned at Functional Requirement part. By the side of Institute Staff, she has a menu navigator in top of the page. This menu contains sub-actor names. Each menu has own sub menu in body part of the page. The sub menu is rendered by request of menu. Interfaces should be having similar blocks and design. That is important to usability. There should no guide for all actors, because Institute Automation System should be created easier.

#### Reliability

The Institute Automation System should be provided to access for all actors by secure. By the side of management, in other words automation side, should be lossless data. We are going to develop the system according to Service Oriented Architecture and Object Relation Model. We have no query-based a database. We designed a relational database model, therefore we defined objects with own attributes. We are not going to work class-based programming. Unlike, we are going to work with service-based programming. We are going to use Proxy design pattern to prevent to reach directly to objects.

#### Performance

Model-view-controller design pattern is going to be implemented in this project. This design pattern satisfies relation among these entities bidirectionally. There are five associations among model-view-controller entities. The connection -in other words association- should be faster. Moreover, observer and listener design pattern structures are going to be used to satisfy real-time synchronization and orchestration. The system is powerful with Python.

Thanks to Django, the system uses less hardware of the server. Therefore, the system will be more performance.

#### Supportability

The system should be reachable over browser in standard computer and mobile devices. All management of Institute Automation System belongs to the system administrator. All support operations can be handled by system administrator, if it is necessary. Because of the implementation of system is developed with strategy design pattern, the implementation folder has a layout structure. This makes to easier supportability. Thanks to Django Template

Language, update operations is done easily and rapidly. Furthermore, default Django Administration Panel has a basic management structure.

### Implementation

Institute Automation System is going to be implemented by using Python language.

All back-end side of project runs with virtual environment (virtualenv). Back-end side of the project is going to be developed with Python. Django framework is going to be used. Django has own model-view-controller structure. When Django framework sets up, it comes with a default project folder that includes settings, urls and init files. When Django app is created, it comes with model, view, admin, tests, apps and init files. All of these files are Python files. Because of there is no complex back-end structure, we are going to work with single app that is called Institute. By the back-end side, Regular Expressions (RegEx) is going to be used for model structure. Also, back-end needs Pillow library of Python.

Django framework uses SQLite3 relational database as default, it can be changed later. We store all data as encrypted for security of the system. Moreover, to prevent to reach to directly to objects in the database, proxy design-pattern is going to be used. It is satisfied by Django.

We are going to use HTML5 and Bootstrap (*to satisfy responsiveness for computers and mobile devices*) to implement front-end side of the project. Settling of files should be set up according to strategy design-pattern. Django Template Language (DTL) is also going to be implemented. It is called jinja2. It has own tags like HTML, those tags are placed in to relevant HTML files to satisfy relation among back-end and front-end. Therefore, observer and listener design-patterns have been satisfied.

We are going to work Object Relational Model (ORM) because of Django is object-oriented framework. We are going to work service-based programming, unlike class-based programming. This is called Service Oriented Architecture (SOA).

We are going to follow Iterative Development Process/Approach throughout whole improvement period. All major activities should be divided into smaller activities, then they should be divided into tasks. This is called Work Breakdown Structure (WBS). This process makes easier to implementation.

When the project finishes, requirements file is created.

### Interface

The interface should be quite simple for all actors, because usability has to be easier. There should no be guide. The interface should be less colorful. The colors of school should be used. Menu navigation should change according to actor. Similar tasks, functions should be put together.

### Packaging

When the project finishes, the requirements should be created. To do that, “freeze” command runs and creates the requirements.txt file. When we start to project to implement, we create a virtual environment (*using virtualenv library*). We can make container as the project finishes as by using Docker.

## 3.4. System Models

### Scenarios and Use-case Scenarios

A scenario is an instance of a use case. Use-case scenario is representation association among user and system.

**Scenario Name:** Login and Logout

**Participating Actor Instance(s):** Oğuzhan: Visitor

**Flow of Events:**

1. Oğuzhan is a visitor who is not defined as an actor in the system. By the side of the project, he is defined as institute staff. He should be logged in to the system to remember as an actor. Therefore, he wants to log in to the system.
2. It is assumed that he has entered to web-address of the project over browser in his computer.
3. He clicks to “Login” linked text label.
4. “Authentication” page is loaded. When it comes, there is a form with own attributes that are e-mail and password fields.
5. Oğuzhan types own e-mail address and password. Then, he clicks to “Submit” button.
6. If there is no error about authentication, he is redirected to inside of the system. Otherwise, he should be typed e-mail and password again.
7. When Oğuzhan desires to log out from the system, he clicks to “Logout” linked text label on menu navigator. Therefore, he could have logged out of the system.

**Use-case Name:** Login and Logout

**Participating Actor(s):** Visitor

**Pre-condition:** It is assumed that he has entered to web-address of the project over browser in his computer earlier.

**Flow of Events:**

1. Visitor is a person who is not defined as an actor in the system. By the side of the project, he is defined as institute staff. He should be logged in to the system to remember as an actor. Therefore, he wants to log in to the system. He clicks to “Login” linked text label.
2. The relevant method in back-end receives the request, and prepares a form including get-message with this request. Then, it returns rendering the content that includes request, link and created form. The link is a new link that it is incoming page after clicking “Login” linked text label.



3. When incoming page is loaded, visitor types his own authentication information to the form, then clicks to “Submit” button.
4. Still, the relevant method is running in back-end. It receives the typed fields and cleans data, then checks the records on database. If there is matching typed information with records on database, visitor is redirected to home page as logged successfully. If there is a problem about authentication, he is redirected to again same page with error tag.
5. If institute staff desires to log out of the system, clicks to “Logout” linked text label in menu navigator.
6. Then, system receives the request and transmits it to logout method in Django. Therefore, the person could have logged out of the system.

**Entry Condition(s):**

- ❖ Institute Automation System must be opened over browser.

**Exit Condition(s):**

- ❖ Institute staff can log in to the system.
- ❖ Institute staff can log out of the system.

**Exceptional Case(s):**

- ❖ The form is sensitive for invalid input.
- ❖ Server might be down.

**Scenario Name:** Display to Institutes/Departments/Programs/Curriculums/Courses/  
Available Courses/Course Types/Sections

**Participating Actor Instance(s):** Oğuzhan: Institute Staff

**Pre-condition:** It is assumed that he has entered to web-address of the project over browser in his computer earlier, then he has logged in to the system.

**Flow of Events:**

1. Displaying to institutes, departments, programs, curriculums, courses, available courses, course types and sections have same scenario. So, only “Displaying Courses” is going to introduce.
2. Oğuzhan clicks to “Institute” linked text label on menu navigation. Then, incoming page has own sub-menu. That includes Institutes, Departments, Programs, Curriculums, Courses (*also, it has own sub-categories*), Available Courses, Course Types and Sections respectively. Oğuzhan clicks to “Courses” linked text label.
3. Incoming page has own table. The table contains code, title, program and view as columns. For instance, “CSE501” for code, “Advanced Software Engineering” for title, “Computer Engineering” for program. Oğuzhan clicks to view button of this row.
4. Incoming page has own attributes, these are course number, name, code, description, credit, program, university, is valid, is deleted, created date. Course number is identification number of this data on system. Name is title of course, that is “Advanced Software Engineering”. Code is “CSE501”. Description includes briefly information about the lecture. Program is “Computer Engineering”, it is not department. There is chance to

define a course from another university, so university should be defined. “University of Isik” is for university. Is valid is to show its statue (open or close), is deleted is to show its statue (removed or not removed). Created date is default established date.

5. We have mentioned at first item, displaying to institutes, departments, programs, curriculums, courses, available courses, course types and sections have same scenario. They are similar to each other about to show data and view details of them. There is just a difference, that is to add new item. Institutes, departments, programs, curriculums, course types and sections have same scenario about adding new item. Adding course scenario is going to introduced later in another case. We are going to introduce adding new item in terms of departments.
6. Oğuzhan clicks to “Departments” linked text label from sub-menu of Institute section. Incoming page has own table and there is a button that is named “Add New Department”, below the table. Oğuzhan clicks to “Add New Department” button. Then, a modal is opened.
7. The opened modal has own attributes. In terms of Department, it has name, institute and head. Oğuzhan types “Computer Science” for name, “Natural Science” for institute and “Olçay Taner Yildiz” for head. Then, clicks to “Submit” button. If there is no error about adding new item, it is added to relevant model in database successfully.

**Use-case Name:** Display to Institutes/Departments/Programs/Curriculums/Courses/Available Courses/Course Types/Sections

**Participating Actor(s):** Institute Staff

**Pre-condition:** It is assumed that he has entered to web-address of the project over browser in his computer earlier, then he has logged in to the system.

**Flow of Events:**

1. Institute Staff is a person who is defined as an actor in the system. By the side of the project, he must be logged in to the system to make process. It is assumed that he has entered to web-address of the project {Scenario and Use-case: Login and Logout}. Institute staff clicks to “Institute” linked text label from menu navigation bar.
2. The request is received, and Institute section is opened after it is rendering. It has own sub-menu and details.
3. Institute staff reaches to sub-menu including institutes, departments, programs, courses, course types and sections. Moreover, courses item has own sub-menu. All of these items have same structure. They have a table and modal. Therefore, dictionary data structure and Django form structure are used in these cases. Institute staff clicks to “Courses” item.
4. The request is received and relevant method in back-end is called, then this request is transmitted to this method. This method gets all data from relevant model, in this case it is Course model. Order changes according to item, such as Courses case gets data according to created date, Departments cases gets data according to alphabetical. Furthermore, these methods create a Django form with request. It is going to be explained with details later. Finally, the method returns rendering content that includes request, url, and all courses dictionary. Therefore, the content can be handled in front-end side successfully.

5. Then, Institute Staff should add to new department item. He clicks to “Department” item from sub-menu of Institute section.
6. The request is received and relevant method in back-end is called, then this request is transmitted to this method. As we have mentioned earlier, this method has two code blocks. First is to get all data from relevant model, second is to create a form to add new item.
7. Incoming page has a table and button to add new item. Institute staff clicks to “Add New Department” button.
8. Thus, a modal is opened by front-end side. The form is created by Django form and it is showed with own attributes on modal. In this process, institute staff types information to relevant fields. If the form is valid, relevant fields are cleaned and saved. If there is an invalid input, institute staff is redirected to invalid page. Otherwise, error is represented on modal.

**Entry Condition(s):**

- ❖ Institute Automation System must be opened over browser.

**Exit Condition(s):**

- ❖ Institute staff can display any item on Institute section.
- ❖ Institute staff can add new item for any item on Institute section.

**Exceptional Case(s):**

- ❖ The form is sensitive for invalid input.
- ❖ Server might be down.

**Scenario Name:** Define A New Course

**Participating Actor Instance(s):** Oğuzhan: Institute Staff

**Pre-condition:** It is assumed that he has entered to web-address of the project over browser in his computer earlier, then he has logged in to the system.

**Flow of Events:**

1. Oğuzhan desires to add new course item. So, he goes to “Define Course” item under Course item under Institute section.
2. Incoming page own table and button. The table is related to courses. It has code, title, program and view as columns. For instance: “CSE501” is for code, “Advanced Software Engineering” is for title, “Computer Engineering” is for program.
3. Button is related to add new item. When it is clicked, modal is opened. Oğuzhan clicks to “Define Course” button.
4. Modal is opened, and it has own attributes. These are course code, course name, course description, course credit, course ETCS credit, program, university and submit button. Oğuzhan types “CSE520” for course code, “Advanced Database Systems” for course name, “This course contains advanced database systems” for course description, “3” for course credit, “6” for course ETCS credit, “Computer Engineering” for program, “Isik University” for university. Then, he clicks to submit button to finish the operation.

**Use-case Name:** Define A New Course

**Participating Actor(s):** Institute Staff

**Pre-condition:** It is assumed that he has entered to web-address of the project over browser in his computer earlier, then he has logged in to the system.

**Flow of Events:**

1. Institute Staff is a person who is defined as an actor in the system. By the side of the project, he must be logged in to the system to make process. It is assumed that he has entered to web-address of the project {Scenario and Use-case: Login and Logout}. Institute staff clicks to “Institute” linked text label from menu navigation bar.
2. The request is received, and Institute section is opened after it is rendering. It has own sub-menu and details.
3. Institute staff reaches to sub-menu including institutes, departments, programs, courses, course types and sections. Moreover, courses item has own sub-menu. Courses item have “define course”, “open/close course”, “available courses”, “remove course” items. All of these items have same structure. Institute staff should define a new course, so clicks to “Define Course” item under Course item.
4. The request is received and relevant method in back-end is called, then this request is transmitted to this method. This method creates a Django form that runs with post method. It returns rendering content that includes request, url, dictionary including form.
5. Institute staff types information to relevant fields on modal screen, then clicks to “Submit” button to finish the operation.
6. Still, the relevant method is running in back-end side of the project. If the form is valid, it receives typed information, cleans them and saves to database. If there is an invalid input, it is redirected to invalid page. Otherwise, error tags are displayed on modal screen.

**Entry Condition(s):**

- ❖ Institute Automation System must be opened over browser.

**Exit Condition(s):**

- ❖ Institute staff can define new course.

**Exceptional Case(s):**

- ❖ The form is sensitive for invalid input.
- ❖ Server might be down.

**Scenario Name:** Open or Close Course

**Participating Actor Instance(s):** Oğuzhan: Institute Staff

**Pre-condition:** It is assumed that he has entered to web-address of the project over browser in his computer earlier, then he has logged in to the system.

**Flow of Events:**

1. Oğuzhan desires to open or close an existing course. So, he goes to “Open/Close Course” item under Course item under Institute section.

2. There is a table including code, title, program and view columns. “CSE501” for code, “Advanced Software Engineering” for title, “Computer Engineering” for program. Oğuzhan should open this course. If there is no data, it does not have to define again. Therefore, he clicks to view linked text label.
3. Incoming page has details of selected course. These are course number, name, code, description, credit, program, university, is valid, is deleted, created date. Course number is identification number of this data on system. Name is title of course, that is “Advanced Software Engineering”. Code is “CSE501”. Description includes briefly information about the lecture. Program is “Computer Engineering”, it is not department. There is chance to define a course from another university, so university should be defined. “University of Isik” is for university. Is valid is to show its statue (open or close), is deleted is to show its statue (removed or not removed). Created date is default established date.
4. Above the details content, there is a button is named “Open/Close Course”. Oğuzhan clicks to button. Then, a modal is opened. There are three components which are is valid, edited date and submit button. True value is for to open the course, false value is for to close the course. Edited date is necessary. Then, Oğuzhan puts tick to is valid field, that means true and types a date YYYY-MM-DD format. Finally, clicks to submit button to finish the operation.

**Use-case Name:** Open or Close Course

**Participating Actor(s):** Institute Staff

**Pre-condition:** It is assumed that he has entered to web-address of the project over browser in his computer earlier, then he has logged in to the system.

**Flow of Events:**

1. Institute Staff is a person who is defined as an actor in the system. By the side of the project, he must be logged in to the system to make process. It is assumed that he has entered to web-address of the project {Scenario and Use-case: Login and Logout}. Institute staff clicks to “Institute” linked text label from menu navigation bar.
2. The request is received, and Institute section is opened after it is rendering. It has own sub-menu and details.
3. Institute staff should open, or close, an existing course. Therefore, he goes to Open/Close Course item under Courses item Institute section.
4. The request is received, and the relevant method in back-end is called, then this request is transmitted to the method. All data in Courses model is showed in table in here.
5. Institute staff clicks to any item on table.
6. The request and id are received, and the relevant method is called, and these are transmitted to this method. Course details are got according to identification number of selected course. They are displayed on content block.
7. Institute staff clicks to “Open/Close Course” button. A modal is opened.
8. Modal is handled in front-end side. There is a similar form. It is handled mentioned method. When the institute staff does something and clicks to submit button, if the form is valid, data cleans and saves to database. If there is an invalid input, he is redirected to invalid page. Otherwise, if there is an error, it is showed with error tag.

**Entry Condition(s):**

- ❖ Institute Automation System must be opened over browser.

**Exit Condition(s):**

- ❖ Institute staff can open or close an existing course.

**Exceptional Case(s):**

- ❖ The form is sensitive for invalid input.
- ❖ Server might be down.
- ❖ Edited date must be YYYY-MM-DD format.

**Scenario Name:** Define Section

**Participating Actor Instance(s):** Oğuzhan: Institute Staff

**Pre-condition:** It is assumed that he has entered to web-address of the project over browser in his computer earlier, then he has logged in to the system.

**Flow of Events:**

1. Oğuzhan desires to define a section for specific course. So, he goes to “Add Section” item on Section item under Institute section.
2. Incoming page has a table and a button. The table contains own attributes which are course, number, instructor, year/semester and view columns. {Sections part has been introduced in scenario and use case: display to institutes - departments - programs curriculums - courses - available courses - course types and sections.
3. Oğuzhan should define a section for a specific course. Therefore, he clicks to add section button to adding new item.
4. A modal is opened. Incoming page has own components which are course, section number, quota quantity, instructor, semester and submit button.
5. Oğuzhan selects a course from dropdown menu, “Artificial Intelligence” for course Then, he types “1” for section number, “20” for quota quantity. He selects an instructor from dropdown menu, “Taner Eskil” for instructor. He selects a semester from dropdown menu, “2017/Winter” for semester. Then, he clicks to submit button to finish the operation, finally.

**Use-case Name:** Define Section

**Participating Actor(s):** Institute Staff

**Pre-condition:** It is assumed that he has entered to web-address of the project over browser in his computer earlier, then he has logged in to the system.

**Flow of Events:**

1. Institute Staff is a person who is defined as an actor in the system. By the side of the project, he must be logged in to the system to make process. It is assumed that he has entered to web-address of the project {Scenario and Use-case: Login and Logout}. Institute staff clicks to “Institute” linked text label from menu navigation bar.

2. The request is received, and Institute section is opened after it is rendering. It has own sub-menu and details.
3. Institute staff should define a section for a specific course. Therefore, he goes to Section item under Institute section. There is a structure that has been explained in earlier. Institute staff clicks to “Add Section” button.
4. A modal is opened. It is handled by front-end. There is a form which is created by Django. Components in here are course, section number, quota quantity, instructor, semester and submit button.
5. Institute staff types, selects and clicks to submit button to finish the operation.
6. The request is received, and the relevant method in back-end is called, and the request is transmitted to this method. If there is no problem such as invalid input, data is cleaned and saved in to database, if the form is valid.

**Entry Condition(s):**

- ❖ Institute Automation System must be opened over browser.

**Exit Condition(s):**

- ❖ Institute staff can define a section for a specific course.

**Exceptional Case(s):**

- ❖ The form is sensitive for invalid input.
- ❖ Server might be down.

**Scenario Name:** Display to All Academic Staffs, Institute Heads, Department Heads, Program Heads, Quota Managers

**Participating Actor Instance(s):** Oğuzhan: Institute Staff

**Pre-condition:** It is assumed that he has entered to web-address of the project over browser in his computer earlier, then he has logged in to the system.

**Flow of Events:**

1. Oğuzhan desires to display all academic staffs, institute heads, department heads, program heads and quota managers. So, he goes to “Academic Staffs” on menu navigator.
2. Incoming page has own sub-menu under Academic Staffs section. The sub-menu contains all academic staffs, institute heads, department heads, program heads, and quota managers. All of these items have same structure, the structure has a table. Academic staff item has also view column in table to display details of every row.
3. Oğuzhan desires to display department heads, so clicks to “Department Heads” linked text label. Incoming page has department name, head and institute as table. “Computer Science” for department name, “Olca Taner Yıldız” for head and “Natural Science” for institute.

**Use-case Name:** Display to All Academic Staffs, Institute Heads, Department Heads, Program Heads, Quota Managers

**Participating Actor(s):** Institute Staff

**Pre-condition:** It is assumed that he has entered to web-address of the project over browser in his computer earlier, then he has logged in to the system.

**Flow of Events:**

1. Institute Staff is a person who is defined as an actor in the system. By the side of the project, he must be logged in to the system to make process. It is assumed that he has entered to web-address of the project {Scenario and Use-case: Login and Logout}. Institute staff clicks to “Academic Staffs” linked text label from menu navigation bar.
2. The request is received, and Academic Staffs section is opened after it is rendering. It has own sub-menu and details. Sub-menu contains all academic staffs, institute heads, department heads, program heads and quota managers items.
3. Institute staff should display all academic staffs or institute heads, department heads, program heads, and quota managers. He clicks to “Department Heads” linked text label.
4. The request is received, the relevant method in back-end is called and the request is transmitted to that method. The method gets all data of Department model, selects some of them. It prepares a dictionary and return rendering content that is including request, url, and the dictionary. Each item on sub-menu under Academic Staff section has own method to handle the self-task.

**Entry Condition(s):**

- ❖ Institute Automation System must be opened over browser.

**Exit Condition(s):**

- ❖ Institute staff can display any item on sub-menu.

**Exceptional Case(s):**

- ❖ Server might be down.

**Scenario Name:** Add Quota Manager

**Participating Actor Instance(s):** Oğuzhan: Institute Staff

**Pre-condition:** It is assumed that he has entered to web-address of the project over browser in his computer earlier, then he has logged in to the system.

**Flow of Events:**

1. Oğuzhan desires to add new quota manager, so clicks to “Quota Managers” linked text label.
2. Incoming page has a table including quota manager and program, and a button.
3. Oğuzhan clicks to “Add New Quota Manager” button.
4. A modal opens and has two components which are dropdown menu from academic staff list and submit button. Oğuzhan selects “Fahrettin Ay” for quota manager, then clicks to submit button to finish the operation.



**Use-case Name:** Add Quota Manager

**Participating Actor(s):** Institute Staff

**Pre-condition:** It is assumed that he has entered to web-address of the project over browser in his computer earlier, then he has logged in to the system.

**Flow of Events:**

1. Institute Staff clicks to “Quota Manager” under Academic Staff section.
2. The request is received and the relevant method in back-end is called and the request is transmitted to the method. The content is displayed on table.
3. Institute Staff clicks to “Add New Quota Manager” button.
4. A modal is opened, still the relevant method is running on back-end side. A form is created by the method.
5. Institute Staff does something and clicks to submit button.
6. The relevant method controls the form, if it is valid, received data is saved to database.

**Entry Condition(s):**

- ❖ Institute Automation System must be opened over browser.

**Exit Condition(s):**

- ❖ Institute staff can add new quota manager.

**Exceptional Case(s):**

- ❖ Server might be down.
- ❖ Dropdown menu might be empty.

**Scenario Name:** Add New Academic Staff

**Participating Actor Instance(s):** Oğuzhan: Institute Staff

**Pre-condition:** It is assumed that he has entered to web-address of the project over browser in his computer earlier, then he has logged in to the system.

**Flow of Events:**

1. Oğuzhan desires to add new academic staff, so clicks to “All Academic Staffs” linked text label.
2. Incoming page has two components which are table and button.
3. Oğuzhan clicks to “Add New Academic Staff” button.
4. A modal is opened. The modal has four components which are staff, university, institute and submit button.
5. Oğuzhan selects “Ahmet Feyzi Ateş” is for staff, “MEF University” is for university, “Natural Science” is for institute. Then, Oğuzhan clicks to submit button.

**Use-case Name:** Add New Academic Staff

**Participating Actor(s):** Institute Staff

**Pre-condition:** It is assumed that he has entered to web-address of the project over browser in his computer earlier, then he has logged in to the system.

**Flow of Events:**

1. Institute Staff clicks to “All Academic Staff” item under Academic Staff section.
2. The request is received and the relevant method in back-end is called and the request is transmitted to the method. The content is displayed on table.
3. Institute Staff clicks to “Add New Quota Manager” button.
4. A modal is opened, still the relevant method is running on back-end side. A form is created by the method.
5. Institute Staff does something and clicks to submit button.
6. The relevant method controls the form, if it is valid, received data is saved to database.

**Entry Condition(s):**

- ❖ Institute Automation System must be opened over browser.

**Exit Condition(s):**

- ❖ Institute staff can add new quota manager.

**Exceptional Case(s):**

- ❖ Server might be down.
- ❖ Dropdown menu might be empty.

**Scenario Name:** Accept or Reject an Application

**Participating Actor Instance(s):** Oğuzhan: Institute Staff

**Pre-condition:** It is assumed that he has entered to web-address of the project over browser in his computer earlier, then he has logged in to the system.

**Flow of Events:**

1. Oğuzhan should accept or reject an application, so clicks to “Grad Students” linked text label on menu navigation.
2. Incoming page has own sub-menu that includes all grad students, applications, and so on. Then, Oğuzhan clicks to Applications item.
3. Incoming page has own table with self-attributes which are full name, application date, ALES, YDS, GPA, degree and view columns. Oğuzhan sees some data “Gözde Ninda Şakir” for full name, “15 May 2018” for application date, “80” for ALES, “70” for YDS, “3” for GPA, “University Degree – Bachelor Science” for degree.
4. Oğuzhan goes inside of selected data over view linked text label. Incoming page has content block that includes many information about person who did application.
5. Moreover, there are two buttons to accept or reject. When the accept button is clicked, a modal is opened. It has own components which are student ID, school e-mail address, curriculum, program, advisor, hold state, registration open statue, approval statue and submit button. Oğuzhan enters information about new student, Gözde Ninda Şakir. Oğuzhan types “214CS2015” for student ID, [gozde.sakir@isik.edu.tr](mailto:gozde.sakir@isik.edu.tr) for school e-mail address, “CSE2016” for curriculum, “Computer Engineering” for program, “Emine Ekin”

for advisor, “true” for hold state, “false” for registration open statue and “true” for approval statue. Then, Oğuzhan clicks to submit button to finish the operation.

6. Oğuzhan sees some data “Çağrı Kandaş” for full name, “15 May 2018” for application date, “56” for ALES, “60” for YDS, “2.8” for GPA, “University Degree – Bachelor Science” for degree. Oğuzhan clicks to view button for this data, then goes to inside of selected data.
7. He clicks to “Reject” button. When a modal is loaded, there is a variable is “called is removed”. Oğuzhan selects it as “Yes”, then clicks to submit button.

**Use-case Name:** Add New Academic Staff

**Participating Actor(s):** Institute Staff

**Pre-condition:** It is assumed that he has entered to web-address of the project over browser in his computer earlier, then he has logged in to the system.

**Flow of Events:**

1. Institute Staff clicks to “Applications” item under Academic Staff section.
2. The request is received and returns Application page with own components.
3. Institute Staff selects a data to go inside.
4. The request is received, and the relevant method in back-end is called and the request is transmitted to this method. It gets all data about selected item by using primary key, identification number.
5. Institute Staff clicks to “Accept” button. When he clicked to the button, Django form is created by relevant handling method. Thus, “student ID, school e-mail address, curriculum, program, advisor, hold state, registration open statue, approval statue” are displayed. They are checked on relevant method in back-end. If there is no invalid input or null point, that means that if the form is valid, the all data is received, cleaned and saved into database. Otherwise, when he clicks to “Reject” button, again Django form is created, only “is removed” variable is displayed.

**Entry Condition(s):**

- ❖ Institute Automation System must be opened over browser.

**Exit Condition(s):**

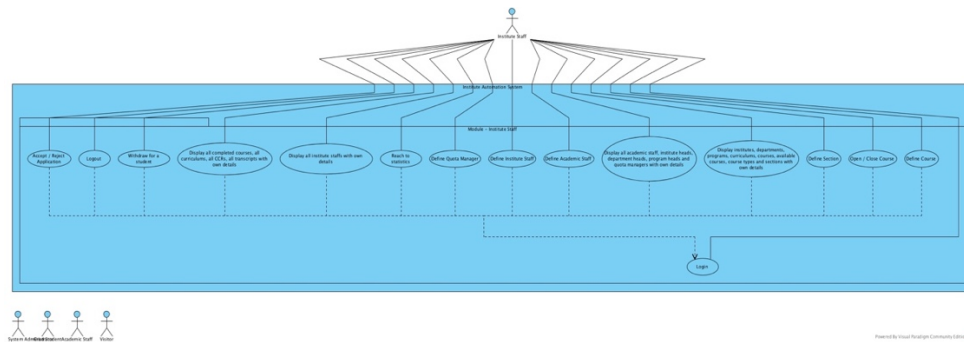
- ❖ Institute staff can accept or reject an application.

**Exceptional Case(s):**

- ❖ Server might be down.
- ❖ Dropdown menu might be empty.
- ❖ Conflict may be occurring.

Use case model

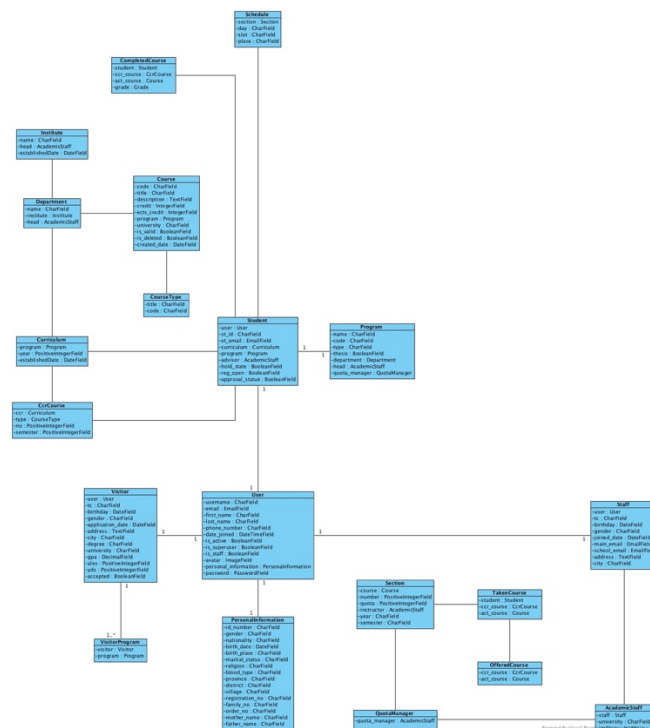
A use case is a generalization of a number of scenarios. Therefore, the number of scenarios must be equal to or greater than the number of use cases. It is reachable in attachments file of the project.



*Diagram 1 – Use-case model*

### Object model

The analysis object model, depicted with UML class diagrams, includes classes, attributes, and operations. The analysis object model is a visual dictionary of the main concepts visible to the user. It is reachable in attachments file of the project. (*Deliverable code: IAS004*)



*Diagram 2 – Object model*

### Dynamic model

Sequence diagrams represent the interactions among a set of objects during a single use case. The dynamic model serves to assign responsibilities to individual classes and, in the process, to identify new classes, associations, and attributes to be added to the analysis object model. All sequence diagrams are given in attachments folder.

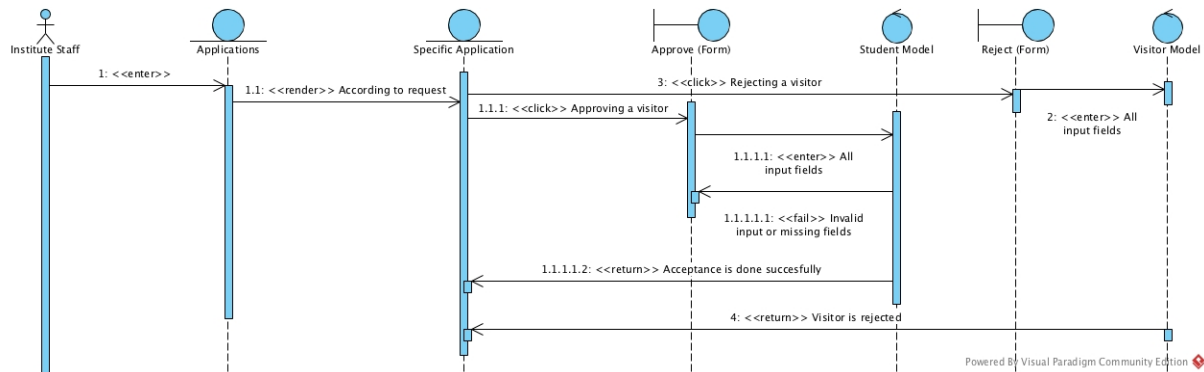


Diagram 3 – Sequence diagram: Accept or reject a visitor

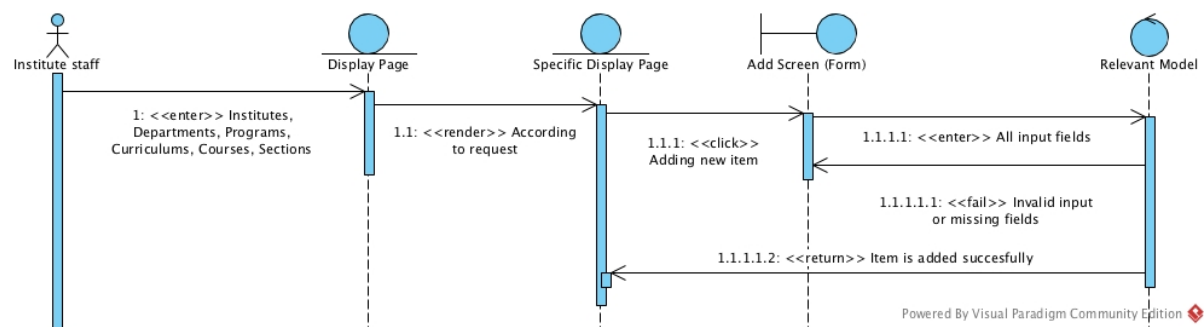


Diagram 4 – Sequence diagram: Display all entities

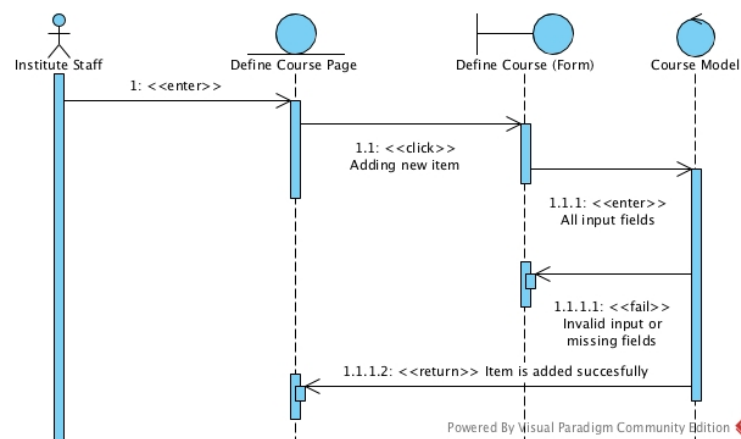


Diagram 5 – Sequence diagram: Define a new course

## Institute Automation System – IAS

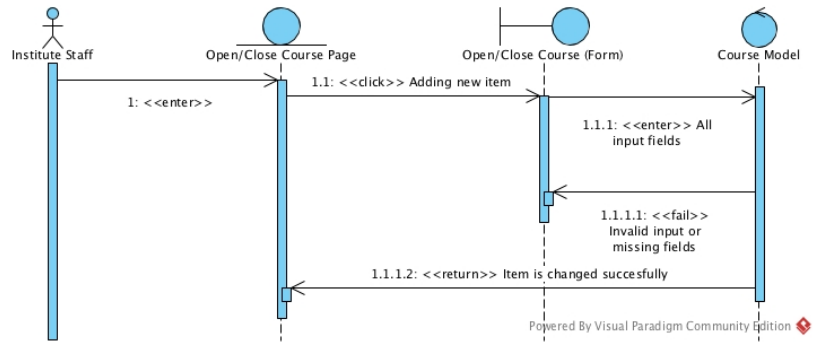


Diagram 6 – Sequence diagram: Open/close an existing course

### User interface—navigational paths and screen mock-ups

Custom menu:

## Institute

<a href="#">Institutes</a>
<a href="#">Departments</a>
<a href="#">Programs</a>
<a href="#">Cirriculums</a>
<a href="#">Courses</a>
<a href="#">- Define Course</a>
<a href="#">- Available Courses</a>
<a href="#">- Open/Close Course</a>
<a href="#">- Remove Course</a>
<a href="#">Course Types</a>
<a href="#">Sections</a>

Custom table:


/institutes/			
You can see the names of all institutes, the heads of institutes and the date they establish.			
Name	Head	Established Date	View
Sosyal Bilimler	Oguzhan Ulusoy	Nov. 11, 2018	<a href="#">View</a>
Doğa ve Fen Bilimleri	None	April 17, 2018	<a href="#">View</a>
<a href="#">Add Institute</a>			

Custom modal:

### Add Institute Screen

Enter given fields correctly.

Institute Name:

Head:  

Established date:

Menu navigator:

### 3.5. Project Schedule

Work Breakdown Structure for Institute Automation System is given:

<b>1) Full Fundamental Implementation</b>	
a) Institute	
i) Institutes	
(1) Add Institute	
(2) Display Selected	
ii) Departments	
(1) Add Department	
(2) Display Selected	
iii) Programs	
(1) Add Program	
(2) Display Selected	
iv) Curriculums	
(1) Add Curriculums	
(2) Display Selected	
v) Courses	
(1) Define Course	
(2) Available Courses	
(a) Display Selected	
(3) Open/Close Course	

	(a) Display Selected
	(4) Remove Course
	(a) Display Selected
vi)	Course Types
	(1) Add Course Type
vii)	Sections
	(1) Add Section
	(2) Display Selected
b)	Academic Staffs
i)	All Academic Staffs
	(1) Add Academic Staff
	(2) Display Selected
ii)	Institute Heads
iii)	Department Heads
iv)	Program Heads
v)	Quota Managers
	(1) Add Quota Manager
c)	Institute Staffs
i)	Add Institute Staff
ii)	Display Selected
d)	Grad students
i)	All Grad students
ii)	Display Selected
iii)	Display Applications of Visitors
iv)	Display Selected
	(1) Approve Visitor
	(2) Reject Visitor
<b>2) Testing – Validation and Verification</b>	
a)	Add 3 data for independent each table individual
b)	Write test methods
i)	Approve mechanism
ii)	Reject mechanism
iii)	Define course mechanism
iv)	Close course mechanism
v)	Remove course mechanism
<b>3) Extra Development</b>	
a)	Graphs by Matplotlib for statistic for PR works

Gantt Chart is tentative and it is given in attachments file of the project as Excel file.



#### 4. Glossary

<i>Academic instructor</i> : An actor on the system. S/he has full authentication of course.
<i>Actor</i> : Defined people on the system (i.e. <i>academic instructor</i> , <i>grand student</i> , <i>institute staff</i> ).
<i>Back-end</i> : Server side of the system. All functionality runs in here.
<i>Bootstrap</i> : An approach to web design for responsiveness.
<i>CSS</i> : An approach to web design for make-up.
<i>Def</i> : The function in Python is defined as <i>def</i> .
<i>Design methodology</i> : The development approaches (i.e. <i>Agile</i> , <i>Iterative</i> , <i>Hacking</i> , and so on).
<i>Design pattern</i> : Reusable solution to a commonly occurring problem (i.e. <i>proxy</i> , <i>observer</i> , <i>listener</i> , <i>strategy</i> , <i>model-view-controller</i> , and so on).
<i>Django</i> : A framework for web design, developed by <i>Python</i> .
<i>Docker</i> : A special container way.
<i>DTL</i> : Django template language, it is called <i>jinja2</i> .
<i>Freeze</i> : A command of Python that creates <i>requirements.txt</i> .
<i>Front-end</i> : Client side of the system. All viewable content is displayed in here.
<i>Function</i> : Solution of the problem.
<i>Grand student</i> : An actor on the system.
<i>Html5</i> : New version of hypertext making language.
<i>Institute staff</i> : An actor on the system.
<i>Iterative development approach</i> : A development approach that is made of splints.
<i>Method</i> : On the other words, <i>def</i> .
<i>MVC</i> : A modern design pattern: model-view-controller.
<i>Observer</i> : Checks the database in disc, properly.
<i>Python</i> : It is a modern programming language.
<i>System administrator</i> : It a person that has full authentication to manage the system.
<i>Template</i> : Viewable contents.
<i>Virtualenv</i> : Work place of Python with required libraries.

## 5. References

### Book:

1. Bruegge B. & Dutoit A.H.. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Prentice Hall, 3<sup>rd</sup> ed.

### Web:

2. Campus Automation System (2002) University of Işık, Campus Online
3. Student Automation System (2015) University of Galatasaray, <http://otomasyon.gsu.edu.tr>
4. Student Automation System, University of Sabanci, <http://sabanci.edu.tr>
5. Django Project Documents: <http://docs.djangoproject.com>

# Institute Automation System

## Requirements Specification and Analysis

Version 1.0

14 March 2018

Oğuzhan Ulusoy

Prepared for  
CSE490 Project



IŞIK UNIVERSITY  
COMPUTER  
SCIENCE AND  
ENGINEERING

## Table of Contents

1.	Introduction.....	2
1.1.	Purpose of the System .....	2
1.2.	Scope of the System .....	2
1.3.	Objectives and Success Criteria of the Project .....	3
1.4.	Definitions, Acronyms, and Abbreviations .....	3
1.5.	Overview .....	3
2.	Current System.....	4
3.	Proposed System .....	4
3.1.	Overview .....	4
3.2.	Functional Requirements.....	4
3.3.	Nonfunctional Requirements.....	4
	Usability .....	5
	Reliability .....	5
	Performance.....	5
	Supportability .....	5
	Implementation .....	6
	Interface.....	6
	Packaging .....	7
	Legal.....	6
3.4.	System Models.....	5
	Scenarios.....	7
	Use case model .....	19
	Object model.....	19
	Dynamic model.....	19
	User interface—navigational paths and screen mock-ups .....	21
4.	Glossary .....	24
5.	References.....	24

## REQUIREMENTS ANALYSIS DOCUMENT[1]

### 1. Introduction

#### 1.1. Purpose of the System

Institute Automation System is a web-based application that provides a school interface to grad students, academic instructors and institute staff. The main goal of the mentioned interface is to satisfy to make easier and faster their activities.

#### 1.1. Scope of the System

As mentioned at Purpose of the System section, Institute Automation System is a web-based application that satisfies a school interface to actors in system. The main target of these interfaces is to provide to make easier and faster their activities.

Institute Automation System has three actors which are grad students, academic instructors and institute staff. Therefore, Institute Automation System has three different interfaces for each actor.

Each interface is going to be had many useful functions according to actor's role in institute. Main purposes are already existing in another system. We are going to be develop a system that contains these main purposes and also useful different functions.

The system is going to be included personal information (*briefly detailed*) for each actor in database. Also, there will be many database operations. Lectures are shown by departments with all information.

Grand student can register and pre-register on school. Grand student can access their CCR, transcript, graduation time, weekly schedule, lecture lists, personal information, notifications. Also, they can create weekly schedule as live preview. They can write message to their academic supervisor or institute staff. They can display their scholarship and when it is ended. They can calculate their semester average. They can reach to each lecture's details.

Academic instructors are added on to the system by institute staff. Their personal information is stored in memory. Some of them are changeable. They can add their lectures with details. They can reach their class lists. They can reach some statistics. They can reach classes where lectures. They can display message from institute or grad students, also they can reply and write new. They can create push notification to all class. They can reach their student's CCR. They can accept or reject their student's weekly schedule, then that is transmitted on to the institute staff.

Institute staff can manage institute staff (*over role*), academic instructors and grad students. All secretarial and instituted tasks can be done. They can reach detailed lists (*i.e. scholarship statues, graduation statues, class lists, lecture lists, personal lists and so on.*) They can make report. They can accept or reject lectures that were added by academic instructors. They can

## Institute Automation System

accept or reject registrations that were done by grad students. They can accept weekly schedules that were done by grad students. They can reach to achieve.

### 1.2. Objectives and Success Criteria of the Project

We are going to improve a system that is more exclusive and capable than existing system. Some functions are mentioned at Scope of the System section are included on existing system. Whereas, some functions are mentioned at Scope of the System section are completely new and useful function. It should be more reliable, efficient and lossless data.

The implementation of Institute Automation System should be understandable, clear and efficient. The all needs should be planned to relevant domain. Security is another objective when we develop the system. The all information (i.e. all actors, reports, schedules, lectures, statistics, lists and so on) are kept protected in secure and the system should be guaranteed this.

The first success criteria of the project are to develop existing functions, then the second success criteria of the project are to develop completely new functions. Yet, the system must be faster running properly. The database will be complicated, but we must use two nested database operation maximums to prevent deadlock or decrease waiting time in queue.

### 1.3. Definitions, Acronyms, and Abbreviations

IAS: Institute Automation System that is going to be developed by us.

Academic personal: People on system that have lectures.

Grad students: People who are master and doctorate students.

Institute staff: People who are secretaries, manager and other.

Existing system: The system that has University of Işık.

*until now...*

### 1.4. Overview

- Rest of the Requirement Analysis Document (*RAD*) contains non-functional (*includes usability, reliability, performance, supportability, implementation, interface, operational, packaging and legal requirements*) and functional requirements (*includes high-level functionality of the system*).
- System models are given. Scenarios are in side of system models section (*that will be mentioned later with instances*). Scenarios are to tell us the details of functional requirements. Use case models, object model, dynamic model and user interface view (mockup) are parts of system models section.

## 2. Current System

There is an existing system that has University of Işık. That is to provide some function to academic instructors, institute staffs and grad students. We have observed Campus Online [2] and have analyzed it. Also, there are some similar automation systems of other universities. We have also observed and analyzed the system of University of Galatasaray[3].

### 3. Proposed System

Proposed system for Institute Automation System is to have existing systematic functions and completely new and useful functions which are to display detailed lists, make report, push notification, display statues (*i.e. scholarship, graduation and so on*), provide writing messages among academic instructors, institute staffs and grad students and so on. The main aim of us is to develop more reliable, faster, exclusive, capable and lossless data system.

#### 3.1. Overview

The system is going to be including three actors which are academic instructors, institute staffs and grad students. System administrator is manager of all system, she has full authentication, but it is not an actor that will be explained in this document. Functional Requirements section contains what any actor will be able to do. Nonfunctional requirements section contains system details, interface details and implementation details.

#### 3.2. Functional Requirements

- i) Functional requirements for grand student are given in below:
  - 1. Grad students shall be able to pre-register.
  - 2. Grad students shall be able to register.
  - 3. Grad students shall be able to login.
  - 4. Grad students shall be able to reach their personal information and to update changeable ones.
  - 5. Grad students shall be able to reach their academic statues.
  - 6. Grad students shall be able to select lectures for weekly schedules.
  - 7. Grad students shall be able to display transcript and CCR.
  - 8. Grad students shall be able to display their military statue (*just for men*).
  - 9. Grad students shall be able to display their scholarship statues.
  - 10. Grad students shall be able to display their graduation statues.
  - 11. Grad students shall be able to display their weekly schedule.
  - 12. Grad students shall be able to reach some information about lectures.
  - 13. Grad students shall be able to send message to institute staff and academic instructors.
  - 14. Grad students shall be able to reach notification about lectures.
  - 15. Grad students shall be able to reach uniform forms.
  - 16. Grad students shall be able to display lectures that will be taken and will be able to take.
  - 17. Grad students shall be able to display previous lectures.
  - 18. Grad students shall be able to calculate average.
- ii) Functional requirements for academic instructor are given in below:
  - 1. Academic instructor shall be able to login.
  - 2. Academic instructor shall be able to add lecture on to the system.
  - 3. Academic instructor shall be able to reach the personal information and update changeable ones.

## Institute Automation System

4. Academic instructor shall be able to display her weekly schedule.
5. Academic instructor shall be able to display her list of the lectures with details.
6. Academic instructor shall be able to grade for grad students.
7. Academic instructor shall be able to send and reply to messages.
8. Academic instructor shall be able to push notification.
9. Academic instructor shall be able to accept or reject the weekly schedule of students.
10. Academic instructor shall be able to display the weekly schedule and CCR of students.
11. Academic instructor shall be able to reach to list of her courses.
12. Academic instructor shall be able to reach to forms.

iii) Functional requirements for institute staff are given in below:

16. Institute staff shall be able to login.
17. Institute staff shall be able to display and reach to some statistics and data about grad students and registrations (*i.e. distribution to places, ages, department, school, faculty, and so on*).
18. Institute staff shall be able to display personal information about grad students, institute staffs and academic personals.
19. Institute staff shall be able to display academic situation about grad students (*i.e. taken lectures, passed lectures, general point average, graduation date, scholarship statue, and so on*). Furthermore, they are able to be displayed as many functional lists.
20. Institute staff shall be able to display some statistics about academic instructors, that are lecture loads, number of students, student lists, given lectures, and so on.
21. Institute staff shall be able to display use of labs and classes.
22. Institute staff shall be able to display success rates.
23. Institute staff shall be able to send message to academic personal and grad students and reply them.
24. Institute staff shall be able to push notification for all actors.
25. Institute staff shall be able to reach forms
26. Institute staff shall be able to make report.
27. Institute staff shall be able to display pre-registration.
28. Institute staff shall be able to accept or reject added lectures.
29. Institute staff shall be able to reach to achieve.
30. Institute staff shall be able to select a place for a course.

### 3.3. Nonfunctional Requirements

#### *Usability*

Institute Automation System should have three different interfaces for all actors. All functions are divided in to the capabilities of actors. Each division should represent an interface. Interfaces should be having similar blocks and design. That is important to



## Institute Automation System

usability. There should no guide for all actors, because Institute Automation System should be created easier.

### *Reliability*

The Institute Automation System should be provided to access for all actors by secure. Proxy design pattern should be implemented in project. In addition to this, ORM (*object relation model*) should be used in project. Each table (*in other say model classes*) should have unique identification number for each instance. Unregistered people should not be anything. Session control is used to provide reliability. The password of all people should be stored as hashed password in database. Details will be mentioned later.

### *Performance*

Model-view-controller design methodology is going to be implemented in this project. According to this methodology, there are five associations among model-view-controller entities. The connection -in other words association- should be faster. The system is powerful with Python.

### *Supportability*

The system should be reachable over browser in standard computer. Management of the all IA system belongs to the system administrator. Database relations, tables and instances should be shown in administrator panel. The design should be implemented with blocks to maintain and update rapidly and easily.

### *Implementation*

Institute Automation System is going to be implemented by using Python language. Backend is going to be developed Python. Django framework is going to be used. Django has a model, view, admin py (*Python files*) files. Therefore, the implementation is going to be improved at relevant sections. Front end is going to develop by HTML5 and CSS. The web-site should be responsive for all devices. Therefore, Bootstrap is going to use for responsiveness. Django Template Language (*DTL*) is going to use to develop the front end. That is called jinja2. It has own components, (*or tags*). Iterative Development Process/Approach is going to be used in this project. In addition to these, some design pattern will be used, which are composite, observer, proxy, model-view-controller.

### *Interface*

The interface should be quite simple for all actors, because usability has to be easier. There should no be guide. The interface should be less colorful. The colors of school should be used. Menu navigation should change according to actor. Similar tasks, functions should be put together.

### *Packaging*

When the project finishes, the requirements should be created. To do that, “freeze” command runs and creates the requirements.txt file. When we start to project to implement,

we create a virtual environment (*using virtualenv library*). We can make container as the project finishes as by using Docker.

### *Legal*

There is no license and purchased product. However, we will receive data from Campus Online. That is required to permission.

### **3.4. System Models**

Obtained actor instances are:

- I. Oğuzhan: Grand student
- II. Recep: Institute staff
- III. Ayşe: Academic instructor
- IV. Yağmur: Visitor

### *Scenarios*

**Scenario Name:** Pre-Registration

**Participating Actor Instance(s):** Yağmur: Visitor

**Flow of Events:**

1. Yağmur is a visitor who is not defined as an actor in the system. She should be pre-registration on to the Institute Automation System. She enters in to the system over her browser.
2. Yağmur goes to Pre-Registration linked text-label.
3. When “Pre-Registration” page is loaded, there is a form that is called pre-registration form. That has own attributes which are first name, middle name, last name, TC number, home address, e-mail address, phone number, PDF of transcript and graduated university, faculty, department.
4. Yağmur types some variables in to the relevant fields. “Yağmur” is for first name, “Zeynep” is for middle name, “Özaydın” is for last name, “17969062492” is for TC number, “Cagribey street, No: 6/15, 34779, Atasehir, Istanbul” is for home address, “[yağmur.ozaydin@gmail.com](mailto:yağmur.ozaydin@gmail.com)” is for e-mail address, “05385607279” is for phone number, “...Documents/mytranscript.pdf” is for PDF of transcript and “University of Işık” is for graduated university, “Faculty of Engineering” is for graduated faculty, “Computer Science” is for graduated department.
5. She clicks to “Save” button to finish the operation.
6. She receives an e-mail that includes random password that was generated by system. She can log in to the system with authentication information (*i.e. e-mail address and password*), then can display her registration process.

**Scenario Name:** Registration

**Participating Actor Instance(s):** Yağmur: Visitor

**Flow of Events:**

1. Yağmur is a visitor who is not defined as an actor in the system. She should do registration on to the Institute Automation System. She enters in to the system over her browser. She can log in to the system with her e-mail and password (*the password is determined on pre-registration*).
2. Yağmur clicks to “Login” linked text-label. As loading completes as, a login form is going to be here with own attributes that are e-mail and password input text fields. She types her authentication information to relevant fields. “[yagmur.ozaydin@gmail.com](mailto:yagmur.ozaydin@gmail.com)” is for e-mail, “\*\*\*\*\*” is for password. If there is no problem, she will be redirected to main page.
3. She clicks to “Registration” linked text-label, then goes to inside.
4. When “Registration” page is loaded, there is a form that is called registration form. That has own attributes which are first name, middle name, last name, TC number, home address, e-mail address, phone number, PDF of transcript and graduated university, faculty, department. Mentioned variables come from pre-registration. Also, there are going to be extra attributes: identification information, program, description, PDF of identification.
5. Yağmur’s information comes from pre-registration form which have been saved to database. “Yağmur” is for first name, “Zeynep” is for middle name, “Özaydın” is for last name, “17969062492” is for TC number, “Cagribey street, No: 6/15, 34779, Atasehir, Istanbul” is for home address, “[yagmur.ozaydin@gmail.com](mailto:yagmur.ozaydin@gmail.com)” is for e-mail address, “05385607279” is for phone number, “...Documents/mytranscript.pdf” is for PDF of transcript and “University of Işık” is for graduated university, “Faculty of Engineering” is for graduated faculty, “Computer Science” is for graduated department. She uploads PDF of identification, types program and description variables to relevant fields.
6. She clicks to “Save” button to finish the operation. The form is going to forward to institute staff for approval.

**Scenario Name:** Login

**Participating Actor Instance(s):** Oğuzhan: Grand student, Academic instructor, Institute staff

**Flow of Events:**

1. Oğuzhan is an actor who is defined as grand student on Institute Automation System. He should log in. He enters to system with his browser.
2. He clicks to “Login” linked text-label.
3. There is a form that includes two components: e-mail address and password.
4. Oğuzhan types his authentication information to authenticate. “[ulusoyoguzhan@gmail.com](mailto:ulusoyoguzhan@gmail.com)” is for e-mail address and “\*\*\*\*\*” is for password. Then, he clicks to “Save” button to finish the operation.
5. If there is no problem, he will be redirected to home.

**Scenario Name:** Display-and-Update Personal Information

**Participating Actor Instance(s):** Oğuzhan: Grand student

**Flow of Events:**

1. Oğuzhan is an actor who is defined as grand student on Institute Automation System. He should log in to display and update his personal information. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “ulusoyoguzhan@gmail.com” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Oğuzhan. He clicks to “Personal Information” linked text-label. Incoming page displays his personal information by orderly which are which are first name, middle name, last name, TC number, home address, e-mail address, phone number, PDF of transcript and graduated university, faculty, department (*the attribute count will increase later*). Each variable has a “Change” button. Oğuzhan should update his phone number. He enters a new phone number in to input char-field. “05373119592” is for old phone number, “05385207279” is for new phone number. Then, clicks to “Save” button to finish the operation.

**Scenario Name:** Display Academic Statue

**Participating Actor Instance(s):** Oğuzhan: Grand student

**Flow of Events:**

1. Oğuzhan is an actor who is defined as grand student on Institute Automation System. He should log in to display academic statue. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “ulusoyoguzhan@gmail.com” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Oğuzhan. He clicks to “Display Academic Statue” linked text-label.
4. Incoming page is actually a summary. There are some attributes that are current school, current program, starting and finishing (*probably*) dates for current school, GPA for current program, graduated school, graduated program, starting and finishing dates for graduated school, GPA for graduated program, scholarship (*if it exists*), academic supervisor, school identification number.
5. “University of Galatasaray” is for current school, “Software Engineering” is for current program, “2018-2019” is for starting and finishing dates for current school, “2.9/4.0” is for GPA for current program, “University of Işık” is for graduated school, “Computer Science and Engineering” is for graduated program, “2015-2018” is for starting and finishing dates for graduated school, “2.6/4.0” is for GPA for graduated program, “%50” is for scholarship, “Ayşe Ulusoy” is for academic supervisor, “214CS2015” is for school identification number.

**Scenario Name:** Select Lecture: Prepare Weekly Schedule

**Participating Actor Instance(s):** Oğuzhan: Grand student

**Flow of Events:**

1. Oğuzhan is an actor who is defined as grand student on Institute Automation System. He should log in to prepare weekly schedule. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “[ulusoyoguzhan@gmail.com](mailto:ulusoyoguzhan@gmail.com)” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Oğuzhan. He clicks to “Prepare Weekly Schedule” linked text-label. (*We assume that registration for lecture is open.*)
4. Incoming page is divided on to the two sections (*horizontally*). First section is for live preview, other section is for selection. Selection part is prepared through each grand student individually. Each student displays the lectures that he will be able to take. Each lecture has time/date, instructor name and a ratio button.
5. Oğuzhan should prepare a weekly schedule. He displays a list that includes the lectures that he will be able to take. Therefore, he can handle the selection rapidly. He selects “CSE520 89/WW Olcay Taner Yildiz”, “CSE658 89FF Taner Eskil”. When he selects, he can see live preview.
6. Finally, he clicks to “Save” button to finish the operation.

**Scenario Name:** Display Transcript

**Participating Actor Instance(s):** Oğuzhan: Grand student

**Flow of Events:**

1. Oğuzhan is an actor who is defined as grand student on Institute Automation System. He should log in to display transcript. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “[ulusoyoguzhan@gmail.com](mailto:ulusoyoguzhan@gmail.com)” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Oğuzhan. He clicks to “Display Transcript” linked text-label.
4. Incoming page has semester, course code, course title, credits, grade and points as a list. Completed lectures are shown in here. “2017 Fall” for semester, “CSE460” for course code, “Artificial Intelligence” for course title, “3” for credits, “AA” for grade and “12” for points. The under position of list is going to be some dynamic attributes which are total credits attempted, total credit completed, total point earned, GPA, overall status. “12” for total credit attempted, “9” for total credit completed, “32” for total point earned, “3.33” for GPA, “Satisfactory” for overall status.

**Scenario Name:** Display CCR

**Participating Actor Instance(s):** Oğuzhan: Grand student

**Flow of Events:**

1. Oğuzhan is an actor who is defined as grand student on Institute Automation System. He should log in to display CCR. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “[ulusoyoguzhan@gmail.com](mailto:ulusoyoguzhan@gmail.com)” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Oğuzhan. He clicks to “Display CCR” linked text-label.
4. Incoming page has semester, course code, course title, grade as a list. Completed lectures are shown in here. “2016 Fall” for semester, “CSE482” for course code, “Advanced Java” for course title, “BA” for grade.

**Scenario Name:** Display Military Statue

**Participating Actor Instance(s):** Oğuzhan: Grand student

**Flow of Events:**

1. Oğuzhan is an actor who is defined as grand student on Institute Automation System. He should log in to display military statue. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “[ulusoyoguzhan@gmail.com](mailto:ulusoyoguzhan@gmail.com)” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Oğuzhan. He clicks to “Display Military Statue” linked text-label.
4. Incoming page has one single attribute. If the grand student is a girl “There is no military statue”, otherwise there is going to be a result. System receives some information at the first time grand student has entered. According to this info, system shows the result. “Your military statue is okay until 2019”.

**Scenario Name:** Display Scholarship Statue

**Participating Actor Instance(s):** Oğuzhan: Grand student

**Flow of Events:**

1. Oğuzhan is an actor who is defined as grand student on Institute Automation System. He should log in to display scholarship statue. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “[ulusoyoguzhan@gmail.com](mailto:ulusoyoguzhan@gmail.com)” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Oğuzhan. He clicks to “Display Scholarship Statue” linked text-label.
4. Incoming page has two attributes that are scholarship percentage, date of starting to school. Also, the ended date is going to be here. “2017” is for date of starting to school,

## Institute Automation System

“%90” scholarship percentage. “2019” is for he ended date. It is calculated through some references (*i.e. total credits, completed credits*).

**Scenario Name:** Display Graduation Statue

**Participating Actor Instance(s):** Oğuzhan: Grand student

**Flow of Events:**

1. Oğuzhan is an actor who is defined as grand student on Institute Automation System. He should log in to display graduation statue. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “[ulusoyoguzhan@gmail.com](mailto:ulusoyoguzhan@gmail.com)” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Oğuzhan. He clicks to “Display Graduation Statue” linked text-label.
4. Incoming page has two attributes that are starting date, ending date and probably ending date. “2018” for starting date, “2019” ending date and “2019” probably ending date.

**Scenario Name:** Display Weekly Schedule

**Participating Actor Instance(s):** Oğuzhan: Grand student

**Flow of Events:**

1. Oğuzhan is an actor who is defined as grand student on Institute Automation System. He should log in to display weekly schedule. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “[ulusoyoguzhan@gmail.com](mailto:ulusoyoguzhan@gmail.com)” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Oğuzhan. He clicks to “Display Weekly Schedule” linked text-label.
4. Incoming page includes time slots and lectures in appropriate positions. “CSE578” is for 78WW, “SE600” is for FF789.

**Scenario Name:** Display Lecture Details

**Participating Actor Instance(s):** Oğuzhan: Grand student

**Flow of Events:**

1. Oğuzhan is an actor who is defined as grand student on Institute Automation System. He should log in to display lecture details. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “[ulusoyoguzhan@gmail.com](mailto:ulusoyoguzhan@gmail.com)” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Oğuzhan. He clicks to “Display Lecture Details” linked text-label.
4. Incoming page includes a list that contains course names and course department. Each of them has a separated link.
5. When the grand student has clicked to any link, a new small window opens.

6. There is going to be details of selected lecture. The displayed lectures can be filtered according to some options (*i.e. department, level*). The new window includes a list that contains course name, course code, course instructor, course class, course credit, course details with link on Course Online. “SE650” for clicked course: “Advanced Software Engineering Project Lecture” for course name, “Computer Science” for course department. When Oğuzhan goes to inside of details page, “Advanced Software Engineering Project Lecture” for course name, “SE650” for course code, “Emine Ekin” for course instructor, “Maslak Campus/302” for course class, “4” for course credit, “.../283467” for course detail link on Course Online.
7. Thus, he can display the course with details.

**Scenario Name:** Send Message

**Participating Actor Instance(s):** Oğuzhan: Grand student

**Flow of Events:**

1. Oğuzhan is an actor who is defined as grand student on Institute Automation System. He should log in to send message. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “ulusoyoguzhan@gmail.com” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Oğuzhan. He clicks to “Send Message” linked text-label.
4. Incoming page has a list that contains name of academic instructors and institute staffs. Also, there is going to be a form. Each name variable contains check box. In addition to these, there are going to be subject and text field. Oğuzhan selects a name from list by clicking a check-box. Oğuzhan types input to relevant fields. “Taner Eskil” for whom, “CSE490 SDD” for subject, “Dear teacher, I have question about SDD of CSE490...” for text.
5. He clicks to “Save” button to finish the operation.

**Scenario Name:** Display Notifications

**Participating Actor Instance(s):** Oğuzhan: Grand student

**Flow of Events:**

1. Oğuzhan is an actor who is defined as grand student on Institute Automation System. He should log in to display notifications. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “ulusoyoguzhan@gmail.com” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Oğuzhan. He clicks to “Display Notifications” linked text-label.
4. Incoming page includes a list that contains notifications with details. The details of any notification are ID number, order number, text, date and sent person. “3” order number, “Today’s lecture is going to be at 9 PM” for text, “11.03.2018” for date and “Olçay Taner Yıldız” for sent person.



**Scenario Name:** Reach-2-Forms

**Participating Actor Instance(s):** Oğuzhan: Grand student, Institute staff, Academic instructor

**Flow of Events:**

1. Oğuzhan is an actor who is defined as grand student on Institute Automation System. He should log in to reach to forms. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “[ulusoyoguzhan@gmail.com](mailto:ulusoyoguzhan@gmail.com)” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Oğuzhan. He clicks to “Reach-to-Form” linked text-label.
4. When “Reach-to-Form” is loaded, there is a list that contains form name, form description, form download link, added date for each item.

**Scenario Name:** Display Lectures (by filtering)

**Participating Actor Instance(s):** Oğuzhan: Grand student

**Flow of Events:**

1. Oğuzhan is an actor who is defined as grand student on Institute Automation System. He should log in to display lectures as filtering. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “[ulusoyoguzhan@gmail.com](mailto:ulusoyoguzhan@gmail.com)” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Oğuzhan. He clicks to “Display Lectures” linked text-label.
4. When “Display Lectures” is loaded, there is a list that contains some linked text labels. They are “Display Completed Lectures”, “Display Open Lectures”, “Display Appropriate Lectures”, “Display Lectures for Program”.
5. The items are basically a list to display the appropriate data. “Display Completed Lectures” is a list that includes the lectures that grand student has passed and completed. “Display Open Lectures” is a list that includes all open courses in that semester. “Display Lectures for Program” is a list that includes all lectures of each institute program. “Display Appropriate Lectures” is a list that includes just the lectures grand student will be able to take.
6. Oğuzhan sees linked text labels and clicks to “Display Completed Lectures”.
7. When “Display Completed Lectures” page is loaded, there are going to be some attributes that are course code, course name, course credit and grade for each completed course. “CSE550” for course code, “High Level Network Design” for course name, “3” for course credit and “BA” for grade.
8. Thus, Oğuzhan can display the completed lectures.
9. Oğuzhan goes to back and clicks to “Display Appropriate Lectures” linked text label. At this time, when “Display Appropriate Lectures” page is loaded, there will be just the lectures that will be able to take. The list contains course code, course name, course instructor, course credit, course date. “CSE562” for course code, “Artificial Intelligence

Engineering” for course name, “Berk Kenan Mataracı” for course instructor, “3” for course credit, “FF89” for course date.

10. Thus, Oğuzhan can display the appropriate lectures.

**Scenario Name:** Display Previous (Completed) Lectures

**Participating Actor Instance(s):** Oğuzhan: Grand student

**Flow of Events:**

1. Oğuzhan is an actor who is defined as grand student on Institute Automation System. He should log in to display previous – completed lectures. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “ulusoyoguzhan@gmail.com” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Oğuzhan. He clicks to “Display Lectures” linked text-label.
4. When “Display Lectures” is loaded, there are going to be some linked text labels. Oğuzhan clicks to “Display Completed Lectures” linked text label.
5. When “Display Completed Lectures” page is loaded, there are going to be some attributes that are course code, course name, course credit and grade for each completed course. “CSE550” for course code, “High Level Network Design” for course name, “3” for course credit and “BA” for grade.
6. Thus, Oğuzhan can display the completed lectures.

**Scenario Name:** Display-and-Calculate Average

**Participating Actor Instance(s):** Oğuzhan: Grand student

**Flow of Events:**

1. Oğuzhan is an actor who is defined as grand student on Institute Automation System. He should log in to display and calculate the average. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “ulusoyoguzhan@gmail.com” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Oğuzhan. He clicks to “Display-and-Calculate Average” linked text-label.
4. When “Display-and-Calculate Average” is loaded, there is going to be GPA (*general point average*). Moreover, there are going to be here total credits attempted, total credit completed, total point earned except GPA.
5. In addition to these, there is going to be a web input form. That is to provide to calculate the average.

**Scenario Name:** Add Lecture

**Participating Actor Instance(s):** Ayşe: Academic instructor

**Flow of Events:**

1. Ayşe is an actor who is defined as academic instructor on Institute Automation System. She should log in to add lecture. She clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. She types her authentication information to log in. “[ulusoyayse@gmail.com](mailto:ulusoyayse@gmail.com)” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Ayşe. She clicks to “Add Lecture” linked text-label.
4. When “Add Lecture” page is loaded, there is a form that has own attributes. If the course exists on the IAS, academic instructor must select the ratio button from displaying course list. Otherwise, academic instructor must type course code, course name, course date, course description, course credit.
5. Ayşe selects a ratio button from the displaying course list. “CSE550 : High Level Network Design” for ratio button. Then, clicks to “Save” button to finish the operation. Otherwise, she must add to these: “CSE622” for course code, “Cyber Security with Linux Kali” for course name, “FF89” for course date, “Fundamentals of cyber security with Linux Kali opsys” for course description, “3” for course credit. Then, clicks to “Save” button to finish the operation.
6. Thus, the added lecture goes to take approval from institute.

**Scenario Name:** Display-and-Update Personal Information

**Participating Actor Instance(s):** Ayşe: Academic instructor

**Flow of Events:**

1. Ayşe is an actor who is defined as academic instructor on Institute Automation System. She should log in to display and update her personal information. She clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. She types her authentication information to log in. “[ulusoyayse@gmail.com](mailto:ulusoyayse@gmail.com)” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Ayşe. She clicks to “Personal Information” linked text-label.
4. When the page is loaded, there are some attributes that are first name, middle name, last name, TC number, home address, e-mail address, phone number, link of CV, department, degree.
5. “Ayşe” for first name, “Ulusoy” for last name, “18181772112” for TC number, “... Atasehir” for home address, “[ulusoyayse@gmail.com](mailto:ulusoyayse@gmail.com)” for e-mail address, “05373119592” for phone number, “.../cv.pdf” for link of CV, “Computer Science” for department, “Ph.D” for degree.
6. Ayşe should change e-mail address, so she types new variable on to the relevant field. “[ulusoyayse@gmail.com](mailto:ulusoyayse@gmail.com)” for old e-mail address, “[ayse.uluso@gsu.edu.tr](mailto:ayse.uluso@gsu.edu.tr)” for new e-mail address. Then, clicks to “Save” button to finish the operation.

**Scenario Name:** Display Weekly Schedule

**Participating Actor Instance(s):** Ayşe: Academic instructor

**Flow of Events:**

1. Ayşe is an actor who is defined as academic instructor on Institute Automation System. She should log in to display weekly schedule. She clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. She types her authentication information to log in. “[ulusoyayse@gmail.com](mailto:ulusoyayse@gmail.com)” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Ayşe. She clicks to “Display Weekly Schedule” linked text-label.
4. Incoming page includes time slots and lectures in appropriate positions. “CSE578” is for 78WW, “SE600” is for FF789.

**Scenario Name:** Display Lecture Details

**Participating Actor Instance(s):** Ayşe: Academic instructor

**Flow of Events:**

1. Ayşe is an actor who is defined as academic instructor on Institute Automation System. She should log in to display lecture with details. She clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. She types her authentication information to log in. “[ulusoyayse@gmail.com](mailto:ulusoyayse@gmail.com)” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Ayşe. She clicks to “Display Lecture with Details” linked text-label. Incoming page includes a list that contains course names and course department. Each of them has a separated link.
4. When the academic instructor has clicked to any link, a new small window opens.
5. There is going to be details of selected lecture. The displayed lectures can be filtered according to some options (*i.e. department, level*). The new window includes a list that contains course name, course code, course class, course credit, course details with link on Course Online.
6. This page displays just lectures that instructor has own. “SE650” for clicked course: “Advanced Software Engineering Project Lecture” for course name, “Computer Science” for course department. When Ayşe goes to inside of details page, “Advanced Software Engineering Project Lecture” for course name, “SE650” for course code, “Ayşe Ulusoy” for course instructor, “Maslak Campus/302” for course class, “4” for course credit, “.../283467” for course detail link on Course Online.
7. Thus, she can display the course with details.

**Scenario Name:** Grade

**Participating Actor Instance(s):** Ayşe: Academic instructor

**Flow of Events:**

1. Ayşe is an actor who is defined as academic instructor on Institute Automation System. She should log in to grade. She clicks to “Login” linked text-label.

## Institute Automation System

2. Incoming page has a form with own attributes. That e-mail address and password. She types her authentication information to log in. “ulusoyayse@gmail.com” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Ayşe. She clicks to “Grade” linked text-label. Incoming page includes a list that contains course code, course name and student count. Each instance has a separated link.
4. Ayşe clicks to “SE650” for course code, “Advanced Engineering Project Lecture” for course name, “10” for student count.
5. When the relevant page is loaded, the class list (*student ID*, *student full name*) will be shown as horizontally. Each instance has a “Grade” button.
6. When Ayşe clicks to “Grade” button of any student, new window will be opened. This window should include input fields and text labels according to course criteria.
7. Then, Ayşe enters variables on to the relevant fields. To quit, she clicks to “Save” button to finish the operation of selected student.
8. After she has graded all class, clicks to “Save” button to finish the operation.

**Scenario Name:** Send-and-Reply Message

**Participating Actor Instance(s):** Ayşe: Academic instructor, Institue Staff

**Flow of Events:**

1. Ayşe is an actor who is defined as academic instructor on Institute Automation System. She should log in to grade. She clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. She types her authentication information to log in. “ulusoyayse@gmail.com” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Ayşe. She clicks to “My Messages” linked text-label.
4. Incoming page includes two tabs – Inbox and New Message.
5. New Message tab looks like: three tabs. These are named “My Students”, “Academic Instructors” and “Institute Staffs”. Also, there is going to be a form. Each name variable contains check box. In addition to these, there are going to be subject and text field. Ayşe selects a name from list by clicking a check-box. Ayşe types input to relevant fields. “Taner Eskil” for whom, “CSE490 SDD” for subject, “Dear teacher, I have question about SDD of CSE490...” for text.
6. She clicks to “Save” button to finish the operation.
7. Inbox tab looks like: a list that contains date, subject, from. When any instance is selected, a new window will be opened. All details of the message can be displayed in here. That are subject, text, from, date. Below the message, the reply field will be and save button to finish the operation.

**Scenario Name:** Push Notification

**Participating Actor Instance(s):** Ayşe: Academic instructor, Institute staff

**Flow of Events:**

- 1) Ayşe is an actor who is defined as academic instructor on Institute Automation System. She should log in to push notification. She clicks to “Login” linked text-label.
- 2) Incoming page has a form with own attributes. That e-mail address and password. She types her authentication information to log in. “ulusoyayse@gmail.com” is for e-mail address, “\*\*\*\*\*” is for password.
- 3) There is a menu navigation bar is for Ayşe. She clicks to “Push Notification” linked text label.
- 4) When “Push Notification” page is loaded, academic instructor must select a class and types her message. Ayşe selects “SE650: Advanced Software Engineering Project” for receiver and types “Dear SE650 students, today’s lecture has been cancelled” for message. Then, clicks to “Push” button to send the message.

**Scenario Name:** Evaluate the Schedule (Accept/Reject)

**Participating Actor Instance(s):** Ayşe: Academic instructor

**Flow of Events:**

1. Ayşe is an actor who is defined as academic instructor on Institute Automation System. She should log in to evaluate the schedule. She clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. She types her authentication information to log in. “ulusoyayse@gmail.com” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Ayşe. She clicks to “Evaluate the Schedule” linked text-label to accept or reject.
4. When “Evaluation” page is loaded, there is a list that contains the students that belong to (*responsibilities of*) academic instructor. Each instance has “Evaluate” button.
5. Academic instructor clicks to any “Evaluate” button to accept or reject the schedule. The details of schedule will be displayed as a list in new window. At end of the list, there will be “accept” or “reject” buttons.
6. Ayşe selects an instance “214CS2015” for student number, “Oğuzhan Ulusoy” for student name and clicks to “Evaluate” button. The schedule of Oğuzhan will be shown as a list. Ayşe can press “Accept” or “Reject” button.

**Scenario Name:** Display CCR or Weekly Schedule

**Participating Actor Instance(s):** Ayşe: Academic instructor

**Flow of Events:**

- 1) Ayşe is an actor who is defined as academic instructor on Institute Automation System. She should log in to display CCR or weekly schedule of any grand student in her section. She clicks to “Login” linked text-label.

- 2) Incoming page has a form with own attributes. That e-mail address and password. She types her authentication information to log in. “[ulusoyayse@gmail.com](mailto:ulusoyayse@gmail.com)” is for e-mail address, “\*\*\*\*\*” is for password.
- 3) There is a menu navigation bar is for Ayşe. She clicks to:
  - a) “Display CCRs” linked text-label.
    - i) “Display CCRs” page is loaded. There is a list of students in *My Student* model-class of Instructor app. The list consists of student number, student full name attributes. They are ordered by alphabetically. Also, there are search box for this page.
    - ii) Ayşe loads “Display CCrs” page. “214CS2015 Oğuzhan Ulusoy, 214CS2016 Murat Akatay, 213CS2016 Gözde Ninda Şakir, 212CS2008 Berk Kenan Mataracı, 216SE2000 Çağrı Kandaş, ...” for list. Ayşe selects an instance. “216SE2000 Çağrı Kandaş” for instance.
    - iii) A new window is opened. There will be CCR table for selected grand student. The table includes that semester, course code, course title, grade as a list. Completed lectures are shown in here. “2016 Fall” for semester, “CSE482” for course code, “Advanced Java” for course title, “BA” for grade.
  - b) “Display Weekly Schedules” linked text-label.
    - i) “Display Weekly Schedules” page is loaded. There is a list of students in *My Student* model-class of Instructor app. The list consists of student number, student full name, attributes. They are ordered by alphabetically. Also, there are search box for this page.
    - ii) Ayşe loads “Display Weekly Schedules” page. “214CS2015 Oğuzhan Ulusoy, 214CS2016 Murat Akatay, 213CS2016 Gözde Ninda Şakir, 212CS2008 Berk Kenan Mataracı, 216SE2000 Çağrı Kandaş, ...” for list. Ayşe selects an instance. “213CS2016 Gözde Ninda Şakir” for instance.
    - iii) A new window is opened. Incoming page includes time slots, and lectures in appropriate positions. “CSE578” is for 78WW, “SE600” is for FF789.

**Scenario Name:** Display List

**Participating Actor Instance(s):** Ayşe: Academic instructor

**Flow of Events:**

1. Ayşe is an actor who is defined as academic instructor on Institute Automation System. She should log in to display CCR or weekly schedule of any grand student in her section. She clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. She types her authentication information to log in. “[ulusoyayse@gmail.com](mailto:ulusoyayse@gmail.com)” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Ayşe. She clicks to “My Classes” linked text label.
4. When “My Classes” page is loaded, academic instructor can see her own classes as linked text label. Then, can go inside of anyone.
5. Ayşe clicks to “CSE578: Cyber Security with Linux Kali” and goes to inside of it.
6. When the link is clicked – opened, the list will be here. That consists of student name, student full name and date of day. At the end of the list will be here count of class.

**Scenario Name:** Display Statistics for Grad students

**Participating Actor Instance(s):** Recep: Institute Staff

**Flow of Events:**

1. Recep is an actor who is defined as institute staff on Institute Automation System. He should log in to display statistics. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “ulusoyrecep@gmail.com” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Recep. He clicks to “Statistics for Grad students” linked text label.
4. When “Statistics” page is loaded, statistics are shown as a list. They are count of grad students, count of academic instructors, count of institute staffs, count of programs, count of departments, distribution to places, nations, ages, sexes, departments, programs, faculties, schools.

**Scenario Name:** Display Personal Information

**Participating Actor Instance(s):** Recep: Institute Staff

**Flow of Events:**

1. Recep is an actor who is defined as institute staff on Institute Automation System. He should log in to display statistics. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “ulusoyrecep@gmail.com” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Recep. He clicks to “Display Personal Information” linked text label. There are three tabs.
  - 3.1. “Academic Instructors”
    - i) “Academic Instructors” page is loaded. There is a list that contains unique identification number on the system, full name. Each instance is connected with detail page.
    - ii) When any instance is clicked, a new page is opened by system.
    - iii) Incoming page has first name, middle name, last name, e-mail address, and many information about the selected person.
  - 3.2. “Grad students”
    - i) “Grad students” page is loaded. There is a list that contains unique identification number on the system, full name. Each instance is connected with detail page.
    - ii) When any instance is clicked, a new page is opened by system.
    - iii) Incoming page has first name, middle name, last name, e-mail address, and many information about the selected person.
  - 3.3. “Institute Staffs”
    - i) “Institute Staffs” page is loaded. There is a list that contains unique identification number on the system, full name. Each instance is connected with detail page.
    - ii) When any instance is clicked, a new page is opened by system.



- iii) Incoming page has first name, middle name, last name, e-mail address, and many information about the selected person.
- 4. There is a search box in personal details main page and instance pages.
- 5. Recep clicks to “Grand Student”. There is going to a list, when the page is loaded. “4545 Oğuzhan Ulusoy, 3434 Serdar Ünlüsoy, 9695 Çağla Çınar, 1325 Mercan Gültekin, ...” for list. Recep clicks to “4545 Oğuzhan Ulusoy” to display details. When the detail page is loaded, there are going to be details of Oğuzhan. “Oğuzhan” for first name, “Ulusoy” for last name, “ulusoyoguzhan@gmail.com” for e-mail address, “Cagribey Street No:6/15, Atasehir” for home address, and so on.
- 6. Also, they can be listed as filtering (*i.e. programs, departments*).

**Scenario Name:** Display Academic Lists

**Participating Actor Instance(s):** Recep: Institute Staff

**Flow of Events:**

- 1. Recep is an actor who is defined as institute staff on Institute Automation System. He should log in to display statistics. He clicks to “Login” linked text-label.
- 2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “ulusoyrecep@gmail.com” is for e-mail address, “\*\*\*\*\*” is for password.
- 3. There is a menu navigation bar is for Recep. He clicks to “Display Academic Lists” linked text label. There are five tabs. (*The tabs will be modified later.*)
  - 3.1. “Taken Courses”
    - i) “Taken Courses” page is loaded. There is a list that contains course code, course name, count of students.
    - ii) “CSE550: High Level Network Design 15, SE650: Advanced Software Engineering Lecture, ...” for given.
  - 3.2. “Open Courses”
    - i) “Open Courses” page is loaded. There is a list that contains course code, course name, count of students.
    - ii) “CSE550: High Level Network Design 15, SE650: Advanced Software Engineering Lecture, ...” for given.
    - iii) They can be filtered by programs.
  - 3.3. “All Courses”
    - i) “All Courses” page is loaded. There is a list that contains course code, course name, count of students.
    - ii) “CSE550: High Level Network Design 15, SE650: Advanced Software Engineering Lecture, ...” for given.
    - iii) They can filtered by programs.
  - 3.4. “Graduation Dates”
    - i) “Graduation Dates” page is loaded. There is a list that contains student number, full name, and graduation date.
    - ii) “214CS2015 Oğuzhan Ulusoy 2018, 214CS2017 Serdar Ünlüsoy 2018, ...” for list.
    - iii) They are ordered by date.

- iv) They can be filtered by programs.

3.5. “General Point Average”

- i) “General Point Average” page is loaded. There is a list that contains student number, full name, GPA.
- ii) They are ordered from big to small.
- iii) They can be filtered by programs.

**Scenario Name:** Display Statistics for Academic Instructors

**Participating Actor Instance(s):** Recep: Institute Staff

**Flow of Events:**

1. Recep is an actor who is defined as institute staff on Institute Automation System. He should log in to display statistics. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “ulusoyrecep@gmail.com” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Recep. He clicks to “Display Statistics for Academic Instructors” linked text label.
4. When “Display Statistics for Academic Instructors” page is loaded, there will be a list consists of unique identification number on the system and full name. They can be filtered according to programs.
5. Recep goes to inside of this page, then the list has “234 Ayşe Ulusoy, 456 Ninda Gözde Şakir, ...” as instance.
6. There are going to be four tabs.
  - 6.1. “Lecture loads”
    - i) “Lecture loads” page is loaded.
    - ii) When the page is loaded, there is going to be a list that consists of instructor ID, full name, lecture loads, department/program.
    - iii) They can be filtered by programs.
  - 6.2. “Number of students”
    - i) “Number of students” page is loaded.
    - ii) When the page is loaded, there is going to be a list that consists of instructor ID, full name, lecture loads, department/program, course code and student count.
  - 6.3. “Student lists”
    - i) “Student lists” page is loaded.
    - ii) When the page is loaded, there is going to be a list that includes instructor ID, full name. Each row is a connected with a URL that is prepared according to instructor ID.
    - iii) When the any instance is clicked, there will be lists that contain course code and students which belong to that course.

## Institute Automation System

**Scenario Name:** Display Use of Classes/Labs

**Participating Actor Instance(s):** Recep: Institute Staff

**Flow of Events:**

1. Recep is an actor who is defined as institute staff on Institute Automation System. He should log in to display statistics. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “[ulusoyrecep@gmail.com](mailto:ulusoyrecep@gmail.com)” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Recep. He clicks to “Display Use of Classes/Labs” linked text label.
4. When “Display Use of Classes/Labs” page is loaded, institute staff can display details.
5. There is going to be a list that contains of classes. They will be shown with use details date-by-date.

**Scenario Name:** Make Report

**Participating Actor Instance(s):** Recep: Institute Staff

**Flow of Events:**

1. Recep is an actor who is defined as institute staff on Institute Automation System. He should log in to display statistics. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “[ulusoyrecep@gmail.com](mailto:ulusoyrecep@gmail.com)” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Recep. He clicks to “Make Report” linked text label.
4. When “Make Report” page is loaded, institute staff can display details.
5. There is going to be a web form that requires input that are title, text, subject.
6. Recep types something relevant field. “Before Semester, to Instructors!” for title, “You are expected to start the lecture 1 hour lately. The purpose of this...” for text, “Announcement” for subject.
7. Then, he clicks to “Save” button to finish the operation.

**Scenario Name:** Display Pre-registration

**Participating Actor Instance(s):** Recep: Institute Staff

**Flow of Events:**

1. Recep is an actor who is defined as institute staff on Institute Automation System. He should log in to display statistics. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “[ulusoyrecep@gmail.com](mailto:ulusoyrecep@gmail.com)” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Recep. He clicks to “Display Pre-registration” linked text label.
4. When the page is loaded, a list will be here. The list includes identification number and full name. Each row is connected with a special location. When the link is clicked, inside of it

## Institute Automation System

is prepared by unique identification number. This unique identification is for pre-registration. It is generated by form, and added on to model in disc.

5. When Recep is gone to inside of any instance, he can display the attributes. “Oğuzhan Ulusoy, CmPE, Computer Engineering, BOUN, 3.4,...” for instance. (*Details of this has explained in Pre-Registration scenario.*)
6. Each detail page 3-buttons: accept, reject, print.

**Scenario Name:** Accept/Reject Added Lectures

**Participating Actor Instance(s):** Recep: Institute Staff

**Flow of Events:**

1. Recep is an actor who is defined as institute staff on Institute Automation System. He should log in to display statistics. He clicks to “Login” linked text-label.
2. Incoming page has a form with own attributes. That e-mail address and password. He types his authentication information to log in. “ulusoyrecep@gmail.com” is for e-mail address, “\*\*\*\*\*” is for password.
3. There is a menu navigation bar is for Recep. He clicks to “Accept/Reject Added Lectures” linked text label.
4. When the page is loaded, a list will be here. The list includes identification number, course code, course name, course description, course date, course credit, instructor name, instructor id. “34 CSE760 Mathematical Fundamentals of AI Ayşe Ulusoy ...” for instance.
5. In addition to these, each row has two-buttons. These are to accept or reject.
6. When “accept” button is clicked, the added course is sent in to another model in disc. Otherwise, it is removed from system.

*Use case scenarios*

**Use-Case Name:** Pre-Registration

**Participating Actor(s):** Visitor

**Flow of Events:**

1. Visitor is a person who is not defined as an actor in the system. She should be pre-registration on to the Institute Automation System. She enters in to the system over her browser.
2. Browser sends request to reach to server. Then, the result returns and web-site is opened.
3. Visitor clicks to “Pre-Registration” linked text-label.
4. System opens “Pre-Registration” page.
5. When “Pre-Registration” page is loaded, there is a form that is called pre-registration form. That has own attributes which are first name, middle name, last name, TC number, home address, e-mail address, phone number, PDF of transcript and graduated university, faculty, department. Visitor types some variables in to the relevant fields. She clicks to “Save” button to finish the operation.
6. The relevant method in back-end is called by form. The input that was sent by HTTP Message that is type POST, is controlled. If there is no problem (*i.e. invalid input*), it calls related method in back-end. The calling method adds variables to object, and sends it to

## Institute Automation System

disc. If there is no problem, system redirects her to another page to display the operation was successful. Pre-registration table receives these items. Also, a unique password is going to generate randomly, then it is sent over e-mail to visitor's e-mail.

*Entry Condition(s):*

- There is no entry condition.

*Exit Condition(s):*

- Visitor can do apply successfully.

*Exceptional Case(s):*

- The form is sensitive for invalid input.
- Server might be down.

**Use-Case Name:** Registration

**Participating Actor(s):** Visitor

**Flow of Events:**

1. Visitor is a person who is not defined as an actor in the system. She should do registration on to the Institute Automation System. She enters in to the system over her browser.
2. Browser sends request to reach to server. Then, the result returns and web-site is opened.
3. She clicks to "Register" linked text-label, then waits the result.
4. System opens "Login" page. There is a form that includes two components which are e-mail address and password. Before "Register" page loads, she has to authenticate.
5. She enters her authentication information to relevant field. Then, she clicks to "Login" button to go inside.
6. The relevant method in back-end handles this. It checks the information in database's relevant model, if it exists. If so, she should be redirected to registration page. It has own form with some attributes. Many of them are stored in database. Others are null fields. The attributes are first name, middle name, last name, TC number, home address, e-mail address, phone number, PDF of transcript and graduated university, faculty, department, identification information, program, description, PDF of identification.
7. She types some variables to relevant fields. Then, she clicks to "Save" button to finish the operation.
8. System receives some data with own methods from database. They are first name, middle name, last name, TC number, home address, e-mail address, phone number, PDF of transcript and graduated university, faculty, department. The information come from pre-registration model is added on to dictionary through its keys, and new variables also are added to on to same dictionary through its keys. The dictionary is added on to the relevant model in database over another method. If there is no problem (*i.e. invalid input or null point exception*), they add. Otherwise, the warning is appeared.

*Entry Condition(s):*

- Visitor must be logged in to Institute Automation System.

*Exit Condition(s):*

- Visitor can do registration successfully.

*Exceptional Case(s):*

- The form is sensitive for invalid input.

## Institute Automation System

- Server might be down.

### **Use-Case Name:** Login

**Participating Actor(s):** Academic instructor, Institute staff, Grand student

#### **Flow of Events:**

1. After any actor has entered in to the system, she clicks to “Login” linked text-label
2. Browser sends request to reach to server. Then, the result returns and web-site is opened.
3. She clicks to “Register” linked text-label, then waits the result.
4. System opens “Login” page. There is a form that includes two components which are e-mail address and password. Before “Register” page loads, she has to authenticate.
5. She enters her authentication information to relevant field. Then, she clicks to “Login” button to go inside.
6. The relevant controller in back-end checks the input (coming from login page), if there is no problem, it calls related method in back-end. The calling method begins to check variables in relevant model in database, if it exists. If so, system redirects the person to home. Home page is changed according to actor type by system.

#### *Entry Condition(s):*

- Any actor must be logged in to Institute Automation System.

#### *Exit Condition(s):*

- Any actor can log in to Institute Automation System successfully.

#### *Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

### **Use-Case Name:** Display-and-Update Personal Information

**Participating Actor(s):** Grand student

#### **Flow of Events:**

**Pre-condition:** Grand student must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, grand student is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as grand student. (*It is created at initialization of project at the beginning.*)
3. Grand student clicks to “Personal Information” linked text-label.
4. System opens the page. The incoming page lists the attributes of grand student, which are first name, middle name, last name, TC number, home address, e-mail address, phone number, PDF of transcript and graduated university, faculty, department (*the attribute count will increase later*) orderly. Also, the changeable ones have “Change” button at same rows.
5. Grand student clicks to any “Change” button to update his any information.

## Institute Automation System

6. System opens the pop-up window. There are label of attribute, old value, input text field and save button.
7. Grand student types new information, then clicks to “Save” button to finish the operation.
8. Form in relevant *def* in back-end is called, and it takes input. The relevant controller in back-end checks the input (*coming from change page*), if there is no problem, it calls related method in back-end. The calling method begins to add new variable on to relevant model in database. System redirects the grand student to “Personal Information” page.

### *Entry Condition(s):*

- Grand student actor must be logged in to Institute Automation System.

### *Exit Condition(s):*

- Grand student actor can display and update his personal information successfully.

### *Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

## **Use-Case Name:** Display Academic Statue

### **Participating Actor(s):** Grand student

### **Flow of Events:**

**Pre-condition:** Grand student must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, grand student is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as grand student. (*It is created at initialization of project at the beginning.*)
3. Grand student clicks to “Display Academic Statue” linked text-label.
4. System opens the page. The incoming page has own attributes which are current school, current program, starting and finishing (*probably*) dates for current school, GPA for current program, graduated school, graduated program, starting and finishing dates for graduated school, GPA for graduated program, scholarship (*if it exists*), academic supervisor, school identification number. The data of relevant attributes comes from different model-classes through grand student’s unique identification number in the system.

### *Entry Condition(s):*

- Grand student actor must be logged in to Institute Automation System.

### *Exit Condition(s):*

- Grand student actor can display academic statue successfully.

### *Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

**Use-Case Name:** Select Lecture: Prepare Weekly Schedule

**Participating Actor(s):** Grand student

**Flow of Events:**

**Pre-condition:** Grand student must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, grand student is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as grand student. (*It is created at initialization of project at the beginning.*)
3. Grand student clicks to “Prepare Weekly Schedule” linked text-label.
4. System opens the page, if the registration for lectures is open. The Incoming page is divided on to the two sections (*horizontally*). First section is for live preview, other section is for selection. Selection part is prepared through each grand student individually. It is obtained according to grand student’s unique identification number in the system. Each student displays the lectures that he will be able to take. Each lecture has time/date, instructor name and a ratio button. Lecture details (*time, instructor name, lecture name, lecture code*) comes from *Lecture Details* model. Appropriate lectures are prepared by another *def* in back-end and are added on to relevant model class. They are stored with grand student’s unique identification number. If grand student desires to display the all open lectures, should click to “Display All Open Lectures” button.
5. Grand student clicks to a ratio button, whatever he should select. Each ratio button runs with identification number of the selected lecture. When “Save” button is clicked, input is determined as identification number that ratio button has.
6. The relevant *def* in back-end is called and takes the input. The information adds to *Taken Lecture* model-class. This operation is repeated for each selection. The criteria is this unique identification number of each student (*might store student ID*). That schedule waits for approval by academic supervisor.

*Entry Condition(s):*

- Grand student actor must be logged in to Institute Automation System.

*Exit Condition(s):*

- Grand student actor can prepare weekly schedule successfully.

*Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.
- The quota problem might be appeared.
- The registration violation might be appeared.



## Institute Automation System

**Use-Case Name:** Display Transcript

**Participating Actor(s):** Grand student

**Flow of Events:**

**Pre-condition:** Grand student must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, grand student is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as grand student. (*It is created at initialization of project at the beginning.*)
3. Grand student clicks to “Display Transcript” linked text-label.
4. System opens the page, and income page has own attributes that are semester, course code, course title, credits, grade and points. Furthermore, there are total credits attempted, total credit completed, total point earned, GPA, overall status. The data comes from relevant model classes that are called *Completed Lectures* and *Completed Academic Statue*.
5. Completed Lectures contains unique identification number of the students and a dictionary that includes semester, course code, grade.
6. The relevant *def* in back-end calls and takes the data.
7. Another *def* is called in here. That *def* takes input as course code, course credits and grade. It calculates the point.
8. The another *def* calls course title, credits from different model class that is called *Lecture List*.
9. When all objects return, the display transcript *def* handles the operation and returns a dictionary that includes each of these.

*Entry Condition(s):*

- Grand student actor must be logged in to Institute Automation System.

*Exit Condition(s):*

- Grand student actor can display transcript successfully.

*Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

**Use-Case Name:** Display CCR

**Participating Actor(s):** Grand student

**Flow of Events:**

**Pre-condition:** Grand student must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, grand student is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as grand student. (*It is created at initialization of project at the beginning.*)
3. Grand student clicks to “Display CCR” linked text-label.

## Institute Automation System

4. System opens the page, and income page has own attributes that are semester, course code, course title, grade. The data comes from relevant model class that is called *Completed Lectures*. Completed Lectures contains unique identification number of the students and a dictionary that includes semester, course code, grade. The relevant *def* in back-end calls and takes the data. The another *def* calls course title, credits from different model class that is called *Lecture List*. When all objects return, the display transcript *def* handles the operation and returns a dictionary that includes each of these.

### *Entry Condition(s):*

- Grand student actor must be logged in to Institute Automation System.

### *Exit Condition(s):*

- Grand student actor can display CCR successfully.

### *Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

### **Use-Case Name:** Display Military Statue

### **Participating Actor(s):** Grand student

### **Flow of Events:**

**Pre-condition:** Grand student must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, grand student is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as grand student. (*It is created at initialization of project at the beginning.*)
3. Grand student clicks to “Display CCR” linked text-label.
4. System opens the page, and income page has own one single attribute. Firstly, system checks the grand student by using ID. The *def* in back-end checks sex of the grand student. If the grand student is female, returns a static result. Otherwise, the system checks the military date due to when as second step. According to this data, a dynamic result is returned.

### *Entry Condition(s):*

- Grand student actor must be logged in to Institute Automation System.

### *Exit Condition(s):*

- Grand student actor can display military statue successfully.

### *Exceptional Case(s):*

- The form is sensitive for invalid input.
- Server might be down.
- The matching problem might be appeared, the searching in relevant disc part might be failed.

**Use-Case Name:** Display Scholarship Statue

**Participating Actor(s):** Grand student

**Flow of Events:**

**Pre-condition:** Grand student must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, grand student is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as grand student. (*It is created at initialization of project at the beginning.*)
3. Grand student clicks to “Display Scholarship Statue” linked text-label.
4. System opens the page, and incoming page has two attributes that are scholarship percentage, date of starting to school. Also, the ended date is going to be here. The model class that is *Personal Information*. The ended date is going to be set up through the starting date. The scholarship’s ended date is set up through references. The setting runs with total credits and completed credits. These comes from *Completed Lectures* of relevant attributes. They are received by student’s identification number on the system.

*Entry Condition(s):*

- Grand student actor must be logged in to Institute Automation System.

*Exit Condition(s):*

- Grand student actor can display scholarship statue successfully.

*Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

**Use-Case Name:** Display Graduation Statue

**Participating Actor(s):** Grand student

**Flow of Events:**

**Pre-condition:** Grand student must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, grand student is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as grand student. (*It is created at initialization of project at the beginning.*)
3. Grand student clicks to “Display Graduation Statue” linked text-label.
4. System opens the page, and incoming page has three attributes that are starting date, ending date and probably ending date. The starting date and ending date comes from model class that is *Personal Information*. Another def inside of relevant def in back-end calculates the probably ending date by using another attribute total credits and completed credits that comes from *Completed Lectures* of relevant attributes. The result is added on to a dictionary and returns.

## Institute Automation System

### *Entry Condition(s):*

- Grand student actor must be logged in to Institute Automation System.

### *Exit Condition(s):*

- Grand student actor can display graduation statue successfully.

### *Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

### **Use-Case Name:** Display Weekly Schedule

**Participating Actor(s):** Grand student

**Flow of Events:**

**Pre-condition:** Grand student must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, grand student is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as grand student. (*It is created at initialization of project at the beginning.*)
3. Grand student clicks to “Display Weekly Schedule” linked text-label. Incoming page includes time slots and lectures in appropriate positions. The data comes from model class that is *My Schedule*. My Schedule model includes ID of student, lectures through time slots. The all objects are received and they are sent in to relevant view.

### *Entry Condition(s):*

- Grand student actor must be logged in to Institute Automation System.

### *Exit Condition(s):*

- Grand student actor can display weekly schedule successfully.

### *Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

### **Use-Case Name:** Display Lecture Details

**Participating Actor(s):** Grand student

**Flow of Events:**

**Pre-condition:** Grand student must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, grand student is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as grand student. (*It is created at initialization of project at the beginning.*)
3. Grand student clicks to “Display Lecture Details” linked text-label. Incoming page includes a list that contains course names and course department. Each of them has a

## Institute Automation System

separated link. When the grand student has clicked to any link, a new small window opens. There is going to be details of the selected lecture. The displayed lectures can be filtered according to some options (*i.e. department, level*). The options are showed by dropdown menu. Each option actually is a query set. It sets the results through coming option from drop-down menu. When the link is clicked, pop-up window is opened. The new window includes a list that contains course name, course code, course instructor, course class, course credit, course details with link on Course Online. The data comes from *All Courses List (Lecture List)* model class and *Open Courses* model class.

*Entry Condition(s):*

- Grand student actor must be logged in to Institute Automation System.

*Exit Condition(s):*

- Grand student actor can display lecture details successfully.

*Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

**Use-Case Name:** Send Message

**Participating Actor(s):** Grand student

**Flow of Events:**

**Pre-condition:** Grand student must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, grand student is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as grand student. (*It is created at initialization of project at the beginning.*)
3. Grand student clicks to “Send Message” linked text-label. Incoming page has a list that contains name of academic instructors and institute staffs. Also, there is going to be a form. Each name variable contains check box. In addition to these, there are going to be subject and text field. When any check box is selected, the unique identification number of instructor or institute staff on text label is received. The *def* takes subject and text as input. The getting inputs and calling input are added on to relevant model class in back-end.

*Entry Condition(s):*

- Grand student actor must be logged in to Institute Automation System.

*Exit Condition(s):*

- Grand student actor can send message successfully.

*Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

## Institute Automation System

**Use-Case Name:** Display Notifications

**Participating Actor(s):** Grand student

**Flow of Events:**

**Pre-condition:** Grand student must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, grand student is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as grand student. (*It is created at initialization of project at the beginning.*)
3. Grand student clicks to “Display Notifications” linked text-label.
4. Incoming page has a list that contains order number, text, date and sent person.
5. The relevant *def* in back-end takes data from relevant model class in back-end. *Notifications* model class is going to be implemented.
6. The objects are ordered by date. Because of dynamic content, there is going to be 10-12 instances. The *list* functionality of Django is going to be used.

*Entry Condition(s):*

- Grand student actor must be logged in to Institute Automation System.

*Exit Condition(s):*

- Grand student actor can display notification successfully.

*Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

**Use-Case Name:** Reach-to-Forms

**Participating Actor(s):** Grand student, Institute staff, Academic instructor

**Flow of Events:**

**Pre-condition:** Any actor must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, any actor is redirected to home page.
2. System sets the appropriate menu navigation bar for all actor differently. (*It is created at initialization of project at the beginning.*) Assume that the actor is grand student.
3. Grand student clicks to “Reach-to-Forms” linked text-label.
4. Incoming page has a list that contains form name, form description, form download link, added date.
5. The relevant *def* in back-end receives data from relevant model class in back-end, *Forms*.
6. The objects are ordered by date. The *list* functionality of Django is going to be used.

*Entry Condition(s):*

- Grand student actor must be logged in to Institute Automation System.

*Exit Condition(s):*

- Grand student actor can reach to forms successfully.

*Exceptional Case(s):*

## Institute Automation System

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

**Use-Case Name:** Display Lectures (by filtering)

**Participating Actor(s):** Grand student

**Flow of Events:**

**Pre-condition:** Grand student must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, grand student is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as grand student. (*It is created at initialization of project at the beginning.*)
3. Grand student clicks to “Display Lectures” linked text-label.
4. System opens the page. When “Display Lectures” page is loaded, there is a list that contains some linked text labels. They are “Display Completed Lectures”, “Display Open Lectures”, “Display Appropriate Lectures”, “Display Lectures for Program”. They are connected with “Display Lecture Inside” view. When any is clicked, the relevant *view* in back-end handles this. Then, the relevant list is displayed by the *def*. Course code, course name, course credit and grade are displayed for each completed course; course code, course name, course credit, course instructor, course date are displayed for each open course. “Display Appropriate Lectures” is prepared according to other lists. First of all, all courses for a program are obtained in result list. Then, completed courses are removed from result list. Then, open courses and result list is compared. The missing ones are removed from result list. Then, finally the result list returns.

**Entry Condition(s):**

- Grand student actor must be logged in to Institute Automation System.

**Exit Condition(s):**

- Grand student actor can display lectures successfully.

**Exceptional Case(s):**

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

**Use-Case Name:** Display Previous (Completed) Lectures

**Participating Actor(s):** Grand student

**Flow of Events:**

**Pre-condition:** Grand student must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, grand student is redirected to home page.

## Institute Automation System

2. System sets the appropriate menu navigation bar for actor as grand student. (*It is created at initialization of project at the beginning.*)
3. Grand student clicks to “Display Lectures” linked text-label to go to display completed lectures.
4. System opens the page. When “Display Lectures” page is loaded, there is a list that contains some linked text labels. They are “Display Completed Lectures”, “Display Open Lectures”, “Display Appropriate Lectures”, “Display Lectures for Program”. They are connected with “Display Lecture Inside” view. When any is clicked, the relevant *view* in back-end handles this. Then, the relevant list is displayed by the *def*.
5. Grand student clicks to “Display Completed Lectures”.
6. System opens another page. Each instance has course code, course name, course credit and grade are displayed for each completed course.

### Entry Condition(s):

- Grand student actor must be logged in to Institute Automation System.

### Exit Condition(s):

- Grand student actor can display completed lectures successfully.

### Exceptional Case(s):

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

## Use-Case Name: Display-and-Calculate Average

### Participating Actor(s): Grand student

### Flow of Events:

**Pre-condition:** Grand student must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, grand student is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as grand student. (*It is created at initialization of project at the beginning.*)
3. Grand student clicks to “Display-and-Calculate Average” linked text-label.
4. System opens the page. When “Display-and-Calculate Average” is loaded, there is going to be GPA (*general point average*). Moreover, there are going to be here total credits attempted, total credit completed, total point earned except GPA. The data comes from relevant model class in back-end that is called *Completed Academic Statue*.
5. Thus, grand student can display the shown data.
6. In addition to forth, there is a form to calculate the average. It runs with a basically mathematical equation in relevant *def* in back-end.

### Entry Condition(s):

- Grand student actor must be logged in to Institute Automation System.

### Exit Condition(s):

- Grand student actor can display and calculate the average successfully.

### Exceptional Case(s):



## Institute Automation System

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

**Use-Case Name:** Add Lecture

**Participating Actor(s):** Academic instructor

**Flow of Events:**

**Pre-condition:** Academic instructor must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, academic instructor is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as academic instructor. (*It is created at initialization of project at the beginning.*)
3. Academic instructor clicks to “Add Lecture” linked text-label.
4. System opens the page. When the page is loaded, there is a list that contains existing courses list on IAS, and each of them has a ratio button item. They come from *All Lecture* model class. In addition to these, a form is going to be here. The form takes academic instructor name and ID (automatically), added date (automatically), course code, course name, course date, course description and course credit as input.
5. Ayşe selects a ratio button or adds new course. Then, she clicks to “Save” button to finish the operation.
6. When she clicks to “Save” button, the relevant *def* handles and operates this operation. All variables are added in to relevant model class that is called *WaitingForApproveLectures* model class. Each instance that has added in to that model, goes to take approve or reject from institute. Therefore, the institute staff receives a notification about this adding operation.

**Entry Condition(s):**

- Academic instructor actor must be logged in to Institute Automation System.

**Exit Condition(s):**

- Academic instructor actor can add course successfully.

**Exceptional Case(s):**

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- The course already exists on the system.
- Server might be down.

## Institute Automation System

**Use-Case Name:** Display-and-Update Personal Information

**Participating Actor(s):** Academic instructor

**Flow of Events:**

**Pre-condition:** Academic instructor must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, academic instructor is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as academic instructor. (*It is created at initialization of project at the beginning.*)
3. Academic instructor clicks to “Personal Information” linked text-label.
4. System opens the page. The incoming page lists the attributes of academic instructor, which are first name, middle name, last name, TC number, home address, e-mail address, phone number, link of CV, degree, department orderly. Also, the changeable ones have “Change” button at same rows.
5. Academic instructor clicks to any “Change” button to update his any information.
6. System opens the pop-up window. There are label of attribute, old value, input text field and save button.
7. Academic instructor types new information, then clicks to “Save” button to finish the operation.
8. Form in relevant *def* in back-end is called, and it takes input. The relevant controller in back-end checks the input (*coming from change page*), if there is no problem, it calls related method in back-end. The calling method begins to add new variable on to relevant model in database. System redirects the grand student to “Personal Information” page.

**Entry Condition(s):**

- Academic instructor actor must be logged in to Institute Automation System.

**Exit Condition(s):**

- Academic instructor actor can display and update her personal information successfully.

**Exceptional Case(s):**

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

**Use-Case Name:** Display Weekly Schedule

**Participating Actor(s):** Academic instructor

**Flow of Events:**

**Pre-condition:** Academic instructor must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, academic instructor is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as academic instructor. (*It is created at initialization of project at the beginning.*)
3. Academic instructor clicks to “Display Weekly Schedule” linked text-label.

## Institute Automation System

4. System opens the page. Incoming page includes time slots and lectures in appropriate positions. The data comes from model class that is *My Schedule*. My Schedule model includes ID of instructor, lectures through time slots. The all objects are received and they are sent in to relevant view.

*Entry Condition(s):*

- Academic instructor actor must be logged in to Institute Automation System.

*Exit Condition(s):*

- Academic instructor actor can display weekly schedule successfully.

*Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

**Use-Case Name:** Display Lecture Details

**Participating Actor(s):** Academic instructor

**Flow of Events:**

**Pre-condition:** Academic instructor must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, academic instructor is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as academic instructor. (*It is created at initialization of project at the beginning.*)
3. Academic instructor clicks to “Display Lecture’s Details” linked text-label.
4. System opens the that page. Incoming page includes a list that contains course names and course department. Each of them has a separated link. When the academic instructor has clicked to any link, a new small window opens. There is going to be details of the selected lecture. The displayed lectures can be filtered according to some options (*i.e. department, level*). The options are showed by dropdown menu. Each option actually is a query set. It sets the results through coming option from drop-down menu. When the link is clicked, pop-up window is opened. The new window includes a list that contains course name, course code, course class, course credit, course details with link on Course Online. The data comes from relevant model class of *Instructor* app.

*Entry Condition(s):*

- Academic instructor actor must be logged in to Institute Automation System.

*Exit Condition(s):*

- Academic instructor actor can display lecture details successfully.

*Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

## Institute Automation System

**Use-Case Name:** Grade

**Participating Actor(s):** Academic instructor

**Flow of Events:**

**Pre-condition:** Academic instructor must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, academic instructor is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as academic instructor. (*It is created at initialization of project at the beginning.*)
3. Academic instructor clicks to “Grade” linked text-label.
4. System opens “Grade” page. When page is loaded, there are some attributes that are course code, course name, student count. Each instance has a linked text label that is called “go to”.
5. Academic instructor clicks to any course’s link.
6. When the details page is prepared according the course ID (*comes from previous page*). There are class list with attributes student ID, student full name. Each instance has a “Grade” button.
7. Academic instructor clicks to any “Grade” button.
8. When it is clicked, pop-up window will be opened. That includes some input fields that are prepared course criteria.
9. Academic instructor grades the grand student. Then, she clicks to “Save” button finish the operation.
10. Pop-up window will be closed. (Assume that, academic instructor has graded all grad students), there is a “Save” button under the class list. She clicks to “Save” button to finish the operation. When it is clicked, all data received and are saved to relevant model class.

*Entry Condition(s):*

- Academic instructor actor must be logged in to Institute Automation System.

*Exit Condition(s):*

- Academic instructor actor can grade successfully.

*Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

**Use-Case Name:** Send-and-Reply Message

**Participating Actor(s):** Academic instructor, Institute staff

**Flow of Events:**

**Pre-condition:** Actor must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, academic instructor is redirected to home page.

## Institute Automation System

2. System sets the appropriate menu navigation bar for actor as academic instructor. (*It is created at initialization of project at the beginning.*)
3. Academic instructor clicks to “My Messages” linked text label.
4. System opens “My Messages” page. Incoming page includes two tabs – Inbox and New Message.
  - a) New Message tab looks like: three tabs. These are named “My Students”, “Academic Instructors” and “Institute Staffs”. Also, there is going to be a form. Each name variable contains check box. In addition to these, there are going to be subject and text field. Ayşe selects a name from list by clicking a check-box. The data is sent to *Messages* model.
  - b) Inbox tab looks like: a list that contains date, subject, from. When any instance is selected, a new pop-up window will be opened. All details of the message can be displayed in here. That are subject, text, from, date. Below the message, the reply field will be and save button to finish the operation. The data comes *Messages* model according to ID of student.

### Entry Condition(s):

- Academic instructor, institute staff must be logged in to Institute Automation System.

### Exit Condition(s):

- Academic instructor, institute staff can send and reply the message successfully.

### Exceptional Case(s):

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

### Use-Case Name: Push Notification

**Participating Actor(s):** Academic instructor, institute staff

### Flow of Events:

**Pre-condition:** Actor must be logged in to the Institute Automation System.

- 1) Login use-case has explained. After authentication operation, actor is redirected to home page.
- 2) System sets the appropriate menu navigation bar for actor’s role on the system. (*It is created at initialization of project at the beginning.*)
- 3) (*We assume that actor is an academic instructor.*) Academic instructor clicks to “Push Notification” linked text label.
- 4) System opens “Push Notification” page. Incoming page includes two input fields: “to” and “text”.
  - a) If the actor is an academic instructor, “to” dropdown menu is prepared for her and that includes lectures the instructor has given.
  - b) If the actor is an institute staff, “to” dropdown menu is prepared for her and that includes course class, departments and all option.

## Institute Automation System

- 5) Both app (*academic instructor and institute staff*) has model files. They include *Push Notification* class for separately. When a new message is sent by academic instructor, the data goes to *Push Notification* model class of academic instructor app.

### Entry Condition(s):

- Academic instructor, institute staff actor must be logged in to Institute Automation System.

### Exit Condition(s):

- Academic instructor, institute staff actor can push notification successfully.

### Exceptional Case(s):

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

### Use-Case Name: Evaluate the Schedule (Accept/Reject)

**Participating Actor(s):** Academic instructor

### Flow of Events:

**Pre-condition:** Academic instructor must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, academic instructor is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as academic instructor. (*It is created at initialization of project at the beginning.*)
3. Academic instructor clicks to “Evaluate the Schedule” linked text label to accept or reject the schedule.
4. System opens the evaluation page. When page is a loaded, there will be a list that include of student number and student full name. Each instance also has “Evaluate” button. The data comes from *My Students* model-class in Instructor app’s model file.
5. Academic instructor clicks to any “Evaluate” button to accept or reject.
6. System opens the pop-up window through student ID. The handled def in back-end compares the student in *My Students* model with *Taken Lectures* model. Then, gets data according to student ID.
7. When academic instructor clicks to any button:
  - a) Accept: The role variable is changed as 1 in *Taken Lectures* model.
  - b) Reject: The role variable is changed as -1 in *Taken Lectures* model.

### Entry Condition(s):

- Academic instructor actor must be logged in to Institute Automation System.

### Exit Condition(s):

- Academic instructor actor can evaluate successfully.

### Exceptional Case(s):

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

**Use-Case Name:** Display CCR or Weekly Schedule

**Participating Actor(s):** Academic instructor

**Flow of Events:**

**Pre-condition:** Academic instructor must be logged in to the Institute Automation System.

- 1) Login use-case has explained. After authentication operation, academic instructor is redirected to home page.
- 2) System sets the appropriate menu navigation bar for actor as academic instructor. (*It is created at initialization of project at the beginning.*)
- 3) There is a menu navigation bar is for Ayşe. She clicks to:
  - a) “Display CCRs” linked text-label.
    - i) “Display CCRs” page is loaded. There is a list of students in *My Student* model-class of Instructor app. The list consists of student number, student full name attributes. Each row in the table is connected with details page running by ID. They are ordered by alphabetically. Also, there are search box for this page.
    - ii) Academic instructor clicks to go in side of the instance.
    - iii) A new window (pop-up) is opened by system, according to ID. There will be CCR table for selected grand student. The table includes that semester, course code, course title, grade as a list. Completed lectures are shown in here. The data comes from *Completed Lectures* model-class in Grand Student app.
    - iv) Thus, academic instructor can display the CCR of selected grand student.
  - b) “Display Weekly Schedules” linked text-label.
    - i) “Display Weekly Schedules” page is loaded. There is a list of students in *My Student* model-class of Instructor app. The list consists of student number, student full name, attributes. They are ordered by alphabetically. Also, there are search box for this page.
    - ii) Academic Instructor clicks to any student in the list.
    - iii) A new pop-up window is opened. Incoming page includes time slots, and lectures in appropriate position.
    - iv) Thus, academic instructor can display the Weekly Schedule of selected grand student.

*Entry Condition(s):*

- Academic instructor actor must be logged in to Institute Automation System.

*Exit Condition(s):*

- Academic instructor actor can display weekly schedules and ccrs successfully.

*Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

## Institute Automation System

**Use-Case Name:** Display List

**Participating Actor(s):** Academic instructor

**Flow of Events:**

**Pre-condition:** Academic instructor must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, academic instructor is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as academic instructor. (*It is created at initialization of project at the beginning.*)
3. Academic instructor clicks to “My Classes” linked text label.
4. System opens the page and will be here course list that academic instructor has own. The data comes from *My Courses* model-class in Instructor app according to ID. They are ordered by alphabetically and connected to details page.
5. Instructor clicks to any item.
- 6.1. System opens the page. There will be a list that consists of student number, student full name that has taken that lecture. *My Schedule/Taken Lectures* model-classes in Grand Student app is searched by relevant *def* in back-end. The comparison condition is that course ID. When the course IDs in both table is equal to each other, the handled *def* receives student full name and number. They are ordered by alphabetically. The date of that day will be appeared in above the list and count of the class will be appeared in below the table.
- 6.2. Also, the same page has “More details” option. It is handled by another *def* in view file. It gets more data by student ID from *all students* model-class in Grand Student app. The more details are: e-mail address, program.

**Entry Condition(s):**

- Academic instructor actor must be logged in to Institute Automation System.

**Exit Condition(s):**

- Academic instructor actor can display list successfully.

**Exceptional Case(s):**

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

**Use-Case Name:** Display Statistics for Grad students

**Participating Actor(s):** Institute staff

**Flow of Events:**

**Pre-condition:** Institute staff must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, institute staff is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as institute staff. (*It is created at initialization of project at the beginning.*)
3. Institute staff clicks to “Statistics for Grad students” linked text label.



## Institute Automation System

4. System opens “Statistics for Grad students” page.
5. When the page is loaded, there are going to be displayed some statistics that are count of grad students, academic instructors, institute staffs, programs, department and also distribution to places, nations, ages, sexes, departments, programs, faculties, schools. The data comes from *Personal Information* model-classes in each apps (*grand student*, *academic instructor*, *institute staff*). Count *def* in Python is used to get data. It is a basically for-loop counter method.

### Entry Condition(s):

- Institute staff must be logged in to Institute Automation System.

### Exit Condition(s):

- Institute staff can display statistics successfully.

### Exceptional Case(s):

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

### Use-Case Name: Display Personal Information

**Participating Actor(s):** Institute staff

**Flow of Events:**

**Pre-condition:** Institute staff must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, institute staff is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as institute staff. (*It is created at initialization of project at the beginning.*)
3. There is a menu navigation bar is for given actor. He clicks to “Display Personal Information” linked text label.
4. System opens “Display Personal Information” page. There are three tabs.
  - 4.1. “Academic Instructors”
    - i) “Academic Instructors” page is loaded. There is a list that contains unique identification number on the system, full name. Each instance is connected with detail page. They are connected with by ID.
    - ii) When any instance is clicked, a new page is opened by system.
    - iii) Incoming page has first name, middle name, last name, e-mail address, and many information about the selected person. That page is prepared by clicked instance’s ID.
    - iv) The data comes from *Personal Information* model-class in Academic Instructors app in back-end.
  - 4.2. “Grad students”
    - i) “Grad students” page is loaded. There is a list that contains unique identification number on the system, full name. Each instance is connected with detail page. They are connected with by ID.
    - ii) When any instance is clicked, a new page is opened by system.

## Institute Automation System

- iii) Incoming page has first name, middle name, last name, e-mail address, and many information about the selected person. That page is prepared by clicked instance's ID.
- iv) The data comes from *Personal Information* model-class in Grad students app in back-end.

### 4.3. "Institute Staffs"

- i) "Institute Staffs" page is loaded. There is a list that contains unique identification number on the system, full name. Each instance is connected with detail page. They are connected with by ID.
- ii) When any instance is clicked, a new page is opened by system.
- iii) Incoming page has first name, middle name, last name, e-mail address, and many information about the selected person. That page is prepared by clicked instance's ID.
- iv) The data comes from *Personal Information* model-class in Institute Staff app in back-end.

### 5. There is a search box in personal details main page and instance pages.

#### *Entry Condition(s):*

- Institute staff must be logged in to Institute Automation System.

#### *Exit Condition(s):*

- Institute staff can display personal informations successfully.

#### *Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

### **Use-Case Name:** Display Academic Lists

### **Participating Actor(s):** Institute staff

### **Flow of Events:**

**Pre-condition:** Institute staff must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, institute staff is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as institute staff. (*It is created at initialization of project at the beginning.*)
3. There is a menu navigation bar is for given actor. He clicks to "Display Academic Lists" linked text label. There are some tabs.
4. "Taken Courses"
  - a. "Taken Courses" page is loaded. There is a list that contains course code, course name, count of students.
  - b. The data comes from *Taken Courses* model-class in Grand Student app id-by-id.
  - c. They can be filtered by programs (*using order def of Python*).
5. "Open Courses"

## Institute Automation System

- a. “Open Courses” page is loaded. There is a list that contains course code, course name, count of students.
  - b. The data comes from *Open Courses* model-class in Grand Student app id-by-id.
  - c. They can be filtered by programs (*using order def of Python*).
6. “All Courses”
- a. “All Courses” page is loaded. There is a list that contains course code, course name, count of students.
  - b. The data comes from *All Courses* model-class in Grand Student app id-by-id.
  - c. They can be filtered by programs (*using order def of Python*).
7. “Graduation Dates”
- a. “Graduation Dates” page is loaded. There is a list that contains student number, full name, and graduation date.
  - b. The data comes from *Personal Information* model-class in Grand Student app id-by-id.
  - c. They are ordered by date (*using order def of Python*).
  - d. They can be filtered by programs (*using order def of Python*).
8. “General Point Average”
- a. “General Point Average” page is loaded. There is a list that contains student number, full name, GPA.
  - b. They are ordered from big to small.
  - c. They can be filtered by programs (*using order def of Python*).

### Entry Condition(s):

- Institute staff must be logged in to Institute Automation System.

### Exit Condition(s):

- Institute staff can display academic lists successfully.

### Exceptional Case(s):

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

### Use-Case Name: Display Statistics for Academic Instructors

**Participating Actor(s):** Institute staff

**Flow of Events:**

**Pre-condition:** Institute staff must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, institute staff is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as institute staff. (*It is created at initialization of project at the beginning.*)
3. There is a menu navigation bar is for given actor. He clicks to “Display Statistics for Academic Instructors” linked text label.

## Institute Automation System

4. When “Display Statistics for Academic Instructors” page is loaded, there will be a list consists of unique identification number on the system and full name. They can be filtered according to programs.
5. There are going to be four tabs.
6. “Lecture loads”
  - a. “Lecture loads” page is loaded.
  - b. When the page is loaded, there is going to be a list that consists of instructor ID, full name, lecture loads, department/program.
  - c. They can be filtered by programs.
7. “Number of students”
  - a. “Number of students” page is loaded.
  - b. When the page is loaded, there is going to be a list that consists of instructor ID, full name, lecture loads, department/program, course code and student count.
8. “Student lists”
  - a. “Student lists” page is loaded.
  - b. When the page is loaded, there is going to be a list that includes instructor ID, full name. Each row is a connected with a URL that is prepared according to instructor ID.
  - c. When the any instance is clicked, there will be lists that contain course code and students which belong to that course.

### *Entry Condition(s):*

- Institute staff must be logged in to Institute Automation System.

### *Exit Condition(s):*

- Institute staff can display statistics for academic instructors successfully.

### *Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

**Use-Case Name:** Display Use of Classes/Labs

**Participating Actor(s):** Institute staff

**Flow of Events:**

**Pre-condition:** Institute staff must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, institute staff is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as institute staff. *(It is created at initialization of project at the beginning.*
3. There is a menu navigation bar is for given actor. He clicks to “Display Use of Classes/Labs” linked text label.
4. When “Display Use of Classes/Labs” page is loaded, institute staff can display details.

## Institute Automation System

5. There is going to be a list that contains of classes. They will be shown with use details date-by-date. The data comes from *Class Detail* model-class in Instructor app.

*Entry Condition(s):*

- Institute staff must be logged in to Institute Automation System.

*Exit Condition(s):*

- Institute staff can display utilization of classes and labs successfully.

*Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

**Use-Case Name:** Display Success Rates

**Participating Actor(s):** Institute staff

**Flow of Events:**

**Pre-condition:** Institute staff must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, institute staff is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as institute staff. (*It is created at initialization of project at the beginning.*)
3. There is a menu navigation bar is for given actor. He clicks to “Display Success Rates” linked text label. When “Display Success Rates” page is loaded, institute staff can display details.
4. There is going to be a list that contains first three students in each program. They are displayed by student ID, student full name, program and GPA. The data comes from *Personal Information* model class in Grand Student app.

*Entry Condition(s):*

- Institute staff must be logged in to Institute Automation System.

*Exit Condition(s):*

- Institute staff can display success rates successfully.

*Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

**Use-Case Name:** Make Report

**Participating Actor(s):** Institute staff

**Flow of Events:**

**Pre-condition:** Institute staff must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, institute staff is redirected to home page.

## Institute Automation System

2. System sets the appropriate menu navigation bar for actor as institute staff. (*It is created at initialization of project at the beginning.*)
3. There is a menu navigation bar is for institute staff. He clicks to “Make Report” linked text label. When “Make Report” page is loaded, institute staff can display details.
4. There is going to be a web form that requires input that are title, text, subject.
5. Institute staff types something relevant field. Then, he clicks to “Save” button to finish the operation.
6. When he clicks to “Save” button, the relevant *def* in back-end is called. The data that comes from form, is added on to the relevant model in disc. *Reports* is name of the model-class in Institute staff app.

### *Entry Condition(s):*

- Institute staff must be logged in to Institute Automation System.

### *Exit Condition(s):*

- Institute staff can make reports successfully.

### *Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

## **Use-Case Name:** Display Pre-registration

**Participating Actor(s):** Institute staff

**Flow of Events:**

**Pre-condition:** Institute staff must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, institute staff is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as institute staff. (*It is created at initialization of project at the beginning.*)
3. There is a menu navigation bar is for institute staff. He clicks to “Display Pre-registration” linked text label.
4. System opens the page. When the page is loaded, a list will be here. The list includes identification number and full name. Each row is connected with a special location. This unique identification is for pre-registration. It is generated by form (*pre-registration form*), and added on to model in disc (*pre-registration model*).
5. When institute staff is gone to inside of any instance, he can display the attributes. Details of this has explained in Pre-Registration scenario and use-case scenario.
6. System opens the page. When the link is clicked, inside of it is prepared by unique identification number. Each detail page 3-buttons: accept, reject, print. Accept button is to accept the apply, reject is to ignore it. When the “accept” button is clicked, role attribute that pre-registration has, is changed from 0 to 1. Otherwise, it is going to be -1.

### *Entry Condition(s):*

- Institute staff must be logged in to Institute Automation System.

### *Exit Condition(s):*

## Institute Automation System

- Institute staff can display applies for pre-registration successfully.

### *Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

### **Use-Case Name:** Accept/Reject Added Lectures

**Participating Actor(s):** Institute staff

**Flow of Events:**

**Pre-condition:** Institute staff must be logged in to the Institute Automation System.

1. Login use-case has explained. After authentication operation, institute staff is redirected to home page.
2. System sets the appropriate menu navigation bar for actor as institute staff. (*It is created at initialization of project at the beginning.*)
3. There is a menu navigation bar is for institute staff. He clicks to “Accept/Reject Added Lectures” linked text label.
4. System opens the page. When the page is loaded, a list will be here. The list includes identification number, course code, course name, course description, course date, course credit, instructor name, instructor id. In addition to these, each row has two-buttons. These are to accept or reject.
5. When “accept” button is clicked, the added course is sent in to another model in disc. Otherwise, it is removed from system. As the “accept” button is clicked, the relevant *def* in back-end receives the data from *waitingForApproveLectures* model-class in Instructor app and send them to another model-class that is called *All Lectures/Courses*. The sending data is course code, course name, course description, course date, course credit, instructor name, instructor id.

### *Entry Condition(s):*

- Institute staff must be logged in to Institute Automation System.

### *Exit Condition(s):*

- Institute staff can accept or reject added lectures successfully.

### *Exceptional Case(s):*

- The form is sensitive for invalid input.
- The matching problem might be appeared, the searching in relevant disc part might be failed.
- Server might be down.

**Note:** The methods which are used in database operations are Python’s own methods. They are add(), save(), remove(), and so on[4].

**Note:** These functional requirements are first step to base design. The functional requirement will be increased later. We are going to follow Iterative Development Approach. Functional requirements that are not in this document, can be added in other sprints.

### 3.5. Project Schedule

Prepare Gantt Chart, and add it to this section.

## 4. Glossary

*Academic instructor*: An actor on the system. S/he has full authentication of course.

*Actor*: Defined people on the system (*i.e. academic instructor, grand student, institute staff*).

*Back-end*: Server side of the system. All functionality runs in here.

*Bootstrap*: An approach to web design for responsiveness.

*CSS*: An approach to web design for make-up.

*Def*: The function in Python is defined as *def*.

*Design methodology*: The development approaches (*i.e. Agile, Iterative, Hacking, and so on*).

*Design pattern*: Reusable solution to a commonly occurring problem (*i.e. proxy, observer, listener, strategy, model-view-controller, and so on*).

*Django*: A framework for web design, developed by *Python*.

*Docker*: A special container way.

*DTL*: Django template language, it is called *jinja2*.

*Freeze*: A command of *Python* that creates *requirements.txt*.

*Front-end*: Client side of the system. All viewable content is displayed in here.

*Function*: Solution of the problem.

*Grand student*: An actor on the system.

*Html5*: New version of hypertext making language.

*Institute staff*: An actor on the system.

*Iterative development approach*: A development approach that is made of splints.

*Method*: On the other words, *def*.

*MVC*: A modern design pattern: model-view-controller.

*Observer*: Checks the database in disc, properly.

*Python*: It is a modern programming language.

*System administrator*: It a person that has full authentication to manage the system.

*Template*: Viewable contents.

*Virtualenv*: Work place of *Python* with required libraries.

## 5. References

- Bruegge B. & Dutoit A.H.. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Prentice Hall, 3<sup>rd</sup> ed.
- Campus Automation System (2002) University of Işık, Campus Online
- Student Automation System (2015) University of Galatasaray, <http://otomasyon.gsu.edu.tr>
- Django Documents – <http://docs.django.org>