# ANKARA UNIVERSITY

**ENGINEERING FACULTY**

**DEPARTMENT OF COMPUTER ENGINEERING**

****

**RESEARCH TECHNIQUES REPORT**

**Student Information System**

**Ömer Anıl ARDIÇ – 13290081**

**Berkan UZ – 16290621**

**ADVISOR: DR. YILMAZ AR**

**30.05.2018**

**ABSTRACT**

Our project is a student information system project for universities. The system has an introducing screen and three different entry modes. These three modes are as follows; Student mode, Teacher mode, Administrator mode.

We created our project using ASP.NET MVC Entity Framework. At first, had decided to develop the project by using ASP.NET, but at the last moment we decided to develop it with PHP. This year's work together with the ones we learned in the last semester, caused our project to be able to develop with ASP.NET. We fully learned about the Model, View and Controller structures.

**Table of Content**

Abstract2

Table of Content3

1.Introduction4

2.Models5

3.Helper Methods10

4.Repositories11

5.Areas-Management13

6.Controllers16

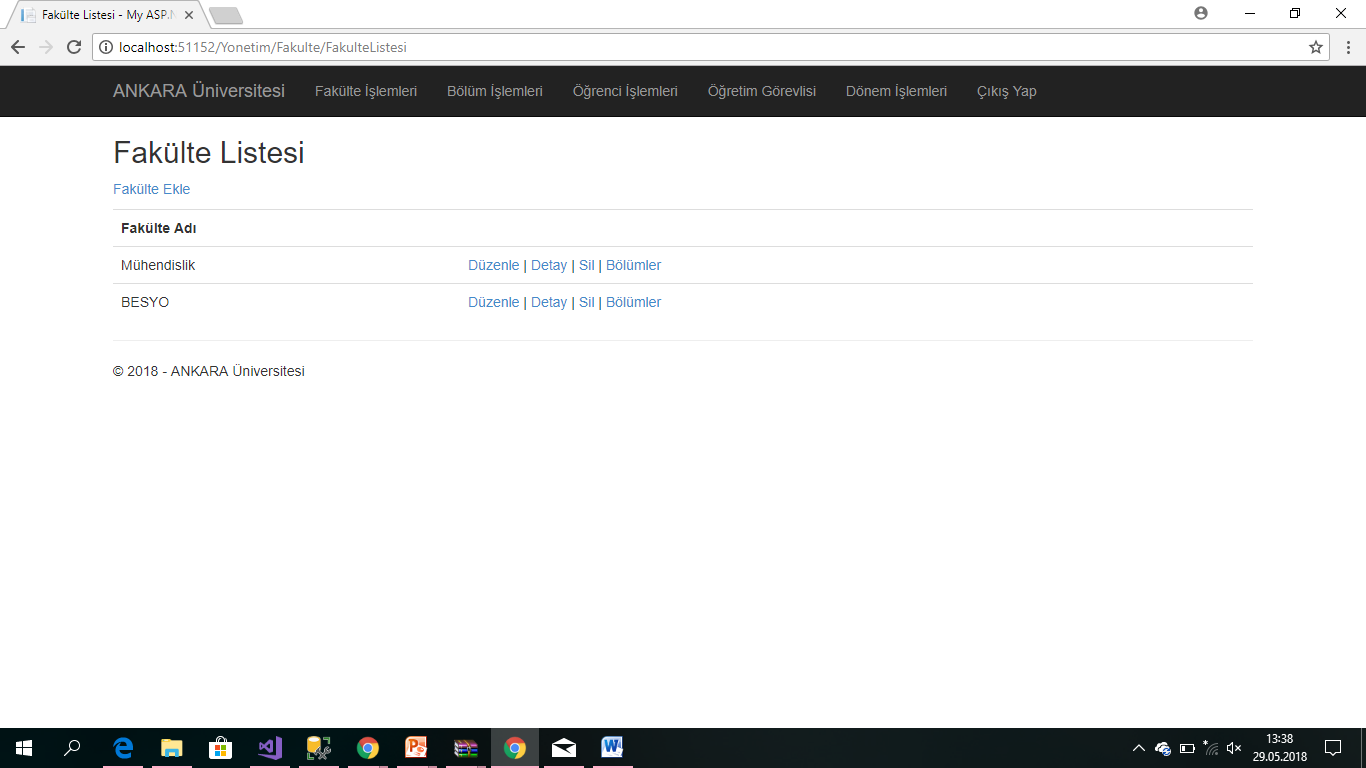
7.Views18

18.Bibliography19

1. **INTRODUCTION**

Our project is a student information system project. In this part of our project, we will talk about the Model and Views. There are three different types of users in our project. These are Student, Lecturer, Information Transaction Staff. IT Staff can do:

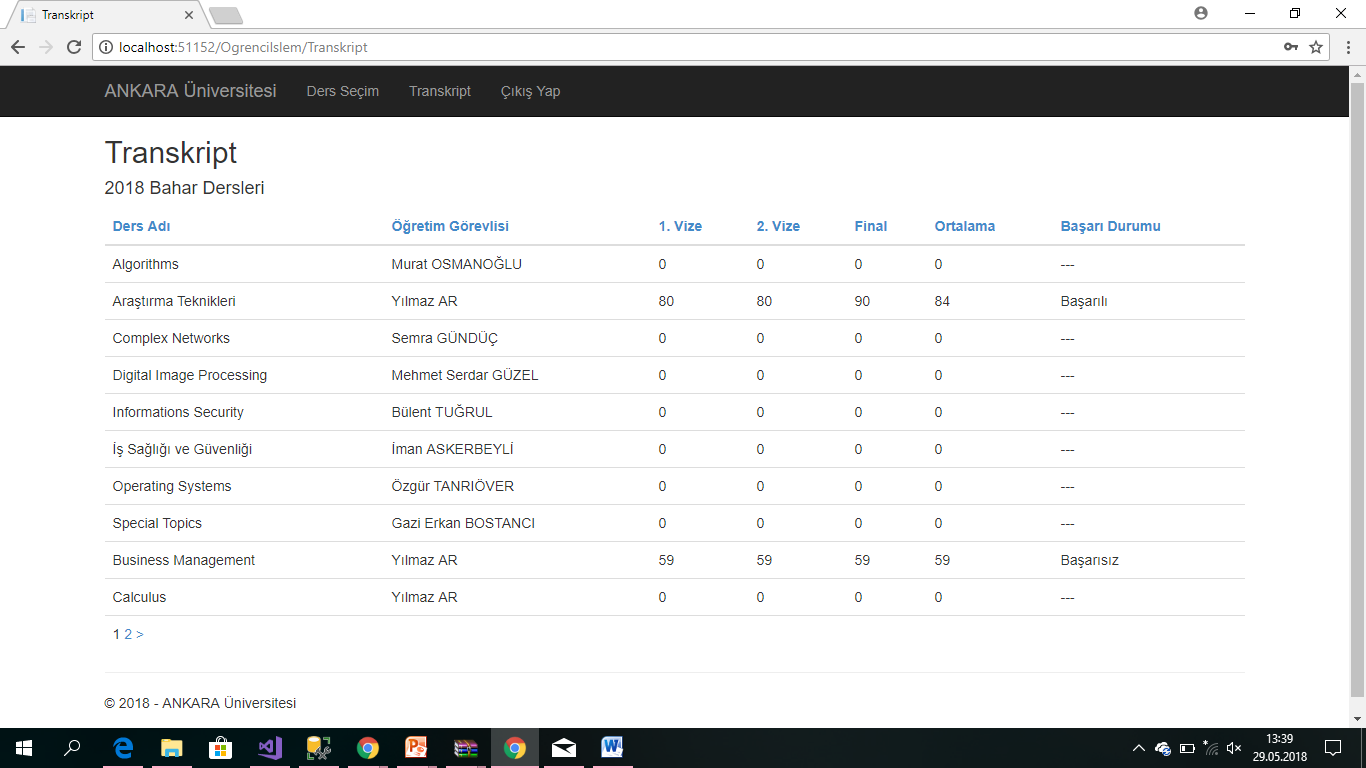
* User Operations (Login-Logout)
* Faculty Operations (Adding and Removing faculty, View and Modify faculty information, List faculties)
* Department Operations(Adding and Removing department, View and Modify department information, List all department and department of faculty)
* Lecturer Operations(Adding and Removing lecturer, View and Modify lecturer information, List all Lecturer and lecturer of department)
* Semester Operations(Adding and Removing semester, View and Modify semester information, List semester)
* Course Operations(Adding and Removing course, View and Modify course information, List all course and course of department)
* Student Operations(Adding and Removing student, View and Modify student information, List all student and student of department)



(Figure 1.1.)

Student can do:

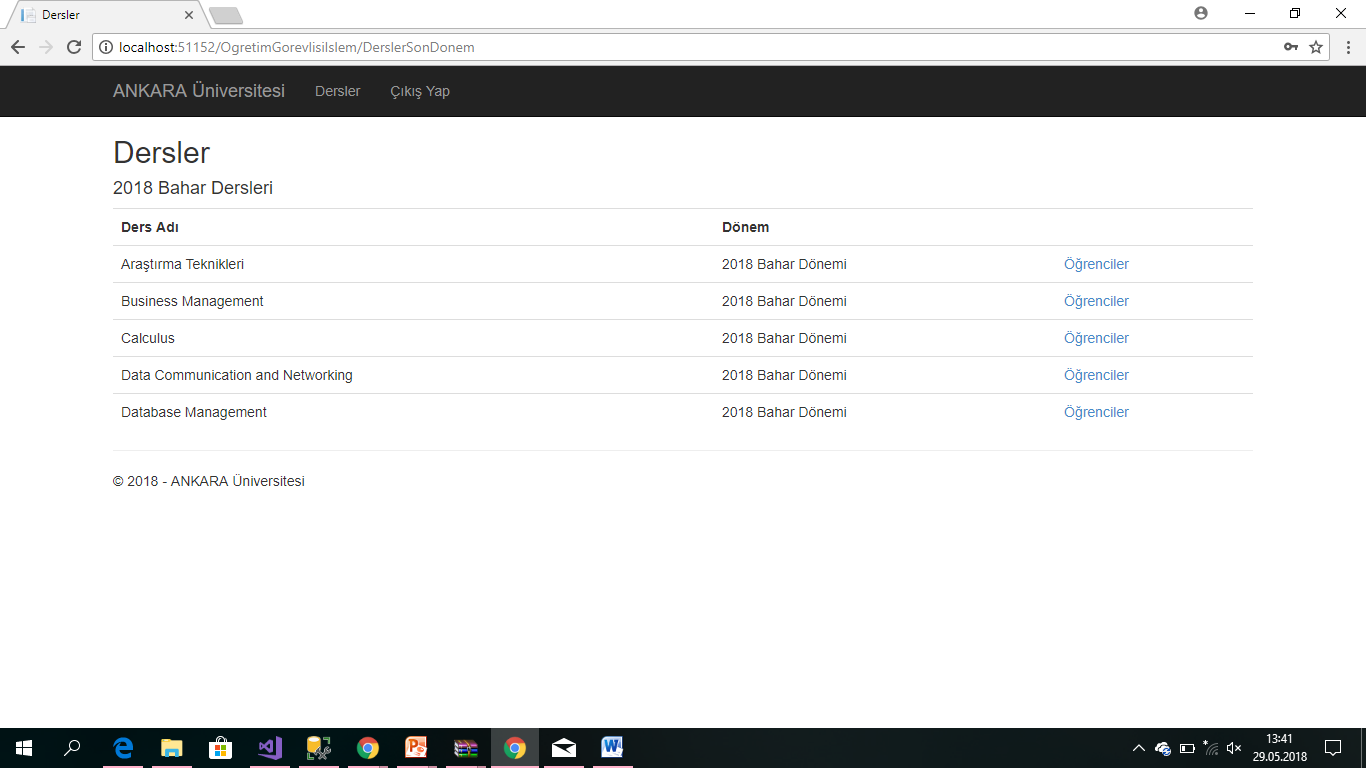
* User Operations (Login-Logout)
* Course Operations(Course selection Operations, View transcript)



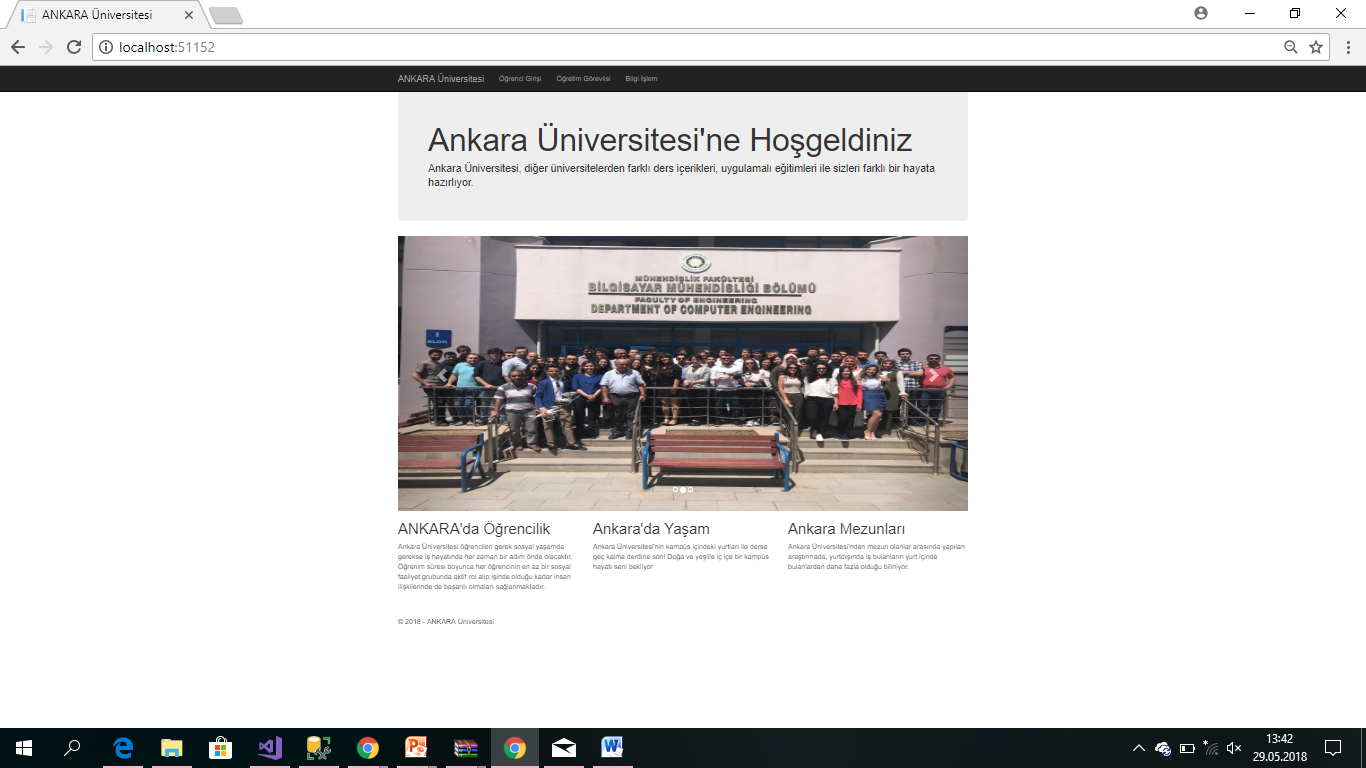
(Figure 1.2.)

Lecturer can do:

* User Operations (Login-Logout)
* Course Operations(View all course and course in semester, View students taking courses, Grade and Register operations)



(Figure 1.3.)



(Figure 1.4.)

1. **MODELS**

The main task of our database is to accurately record the transactions which are made by users. We created 10 tables in our project database. These tables are Fakulte Table, Bolum Table, Ders Table, Donem Table, DonemDers Table, Enumlar Table, ApplicationUser Table, Ogrenci Table, OgrenciDonemDers Table, OgretimGorevlisi Table. We created the database with the Microsoft Sql Server.

* 1. **Fakulte Table**

We started with the faculty object, which is the biggest object for model. Fakulte table consists of FakulteID, Name and Address column. We keep the courses of the faculty virtual as a property of type ICollection<section>. FakulteID is the Primary Key of table.

* public virtual ICollection<Bolum> Bolumler { get; set; }
  1. **Bolum Table**

Bolum table consists of BolumID, Name, Address, Code, FakulteID column. BolumID is the Primary Key of the table. The section code must consist of 3 characters. We used the FakulteID column because each department must be in a faculty. We kept the student and course information of the department as follows:

* public virtual ICollection<Ogrenci> Ogrenciler { get; set; }
* public virtual ICollection<Ders> Dersler { get; set; }
  1. **OgretimGorevlisi Table**

OgretimGorevlisi table consists of OgretimGorevlisiID, Name, Surname, Identification Number, E-mail, DateofBirth, Entry-exit date and BolumID column. OgretimGorevlisiID is the Primary Key of the table. We used the BolumID column because each lecturer must be in a department. We kept the Donemders information of the department as follows:

* public virtual ICollection<DonemDers> DonemDersler { get; set; }
  1. **Donem Table**

Donem table consists of DonemID, Year, Donemtip column. DonemID is the Primary Key of the table. There are 3 different term types. These are summer, spring and autumn. We set these types in the Enum table. We reach the courses opened during the semester as follows:

* public virtual ICollection<DonemDers> DonemDersler { get; set; }
  1. **Ders Table**

Ders table consist of DersID, Name, Code and BolumID column. DersID is the Primary Key of table. We used the BolumID column because each course must in a department. We will reach the courses opened during the semester in the following way:

* public virtual ICollection<DonemDers> DonemDersler { get; set; }
  1. **DonemDers Table**

DonemDers table consist of DonemDersID, DonemID, DersID, OgretimGorevlisiID column. DonemDersID is the Primary Key of the table. We understand which students selected which course of during the semester as follows:

* public virtual ICollection<OgrenciDonemDers> OgrenciDonemDersler {get;set;}
  1. **Ogrenci Table**

Ogrenci table consist of OgrenciID, Name, Surname, E-mail, Identification Number, Entry-exit date, Date of Birth and BolumID column. OgrenciID is the Primary Key of the table. We used the BolumID column because each student must be in a department. We understand which students selected which course of during the semester as follows:

* public virtual ICollection<OgrenciDonemDers> OgrenciDonemDersler {get;set;}
  1. **OgrenciDonemDers Table**

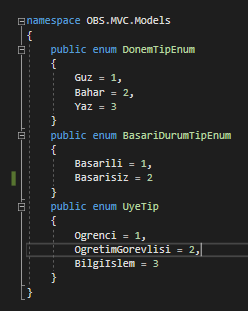
OgrenciDonemDers table consists of ID, DonemDersID, OgrenciID, Midterm1, Midterm2, Final, Average and BasariDurumTip column.

* 1. **ApplicationUser Table**

We created this table to determine the roles of our users. We achieved this as follows:

* using Microsoft.AspNet.Identity.EntityFramework;
* public class ApplicationUser : IdentityUser { }
  1. **Enumlar Table**

We created this table to determine the types. These are Semester Types, User Types, Success Status Types.



(Figure 2.10.1)

* 1. **Other Models**

We created two models that will help us out for the models we use in the project. These are HelperModels and ViewModels. These models are not for recording to the database. We use it to do small operations within the project.

* + 1. **Helper Models**

The methods in the Helper Model work as follows:

* If the action we have made is successful, it will return true, otherwise it will return false. We will also send a warning message according to the result of the status.
* If we need to return a data, we use:
* public class NIslemSonuc<T> : NIslemSonuc
* We also keep the ID and BolumID information of the logged-in users here.
  + 1. **View Models**

In this model, there are auxiliary operations to the operations performed on the View. These operations are Course operations, Name operations, Input operations, Grade operations and List operations.

1. **HELPER METHODS**

We defined our helper methods in the project with Helper classes in Helper folder. We have two helper classes. These are EnumHelper and SelectListHelper. In the EnumHelper class we processed enums defined in the system. In this class we created the static method which is called GetirDonemAdi. This method returns the name of the semester according to the value of the submitted type.

* switch (tip)
* {
* case DonemTipEnum.Bahar: return "Bahar";
* case DonemTipEnum.Guz: return "Guz";
* case DonemTipEnum.Yaz: return "Yaz";
* }

We used the SelectListHelper method for listing. With this method, we can list Faculty, Department, Course, Student and Teacher according to user's request. We did this with the ASP.NET programming language's IEnumarable property. For example:

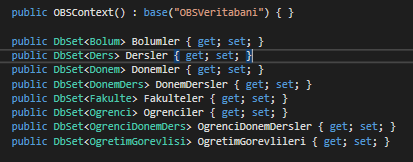
* public static IEnumerable<SelectListItem> GetirFakulteler()

1. **REPOSITORIES**

We will do all database operations in the repository class

* 1. **OBSContext**

After creating the models we created the OBSContext class, which we will perform database operations on. The name of the database is defined in the Constructor method of the DBcontext class. Then we created a <ConnectionString> via web.config. Later, we created the data set of the models which we will hold information in the database.



(Figure 4.1.1)

* 1. **BaseRepository**

We will perform student operations in the OgrenciRepository class, Lecturer operations in the OgretimGorevlisiRepository class. The BaseRepository class is a class that contains the database objects (OBSContext and UyelikRepository). All repository classes will inherit the BaseRepository class. We will get rid of doing the same operations again and again in each repository class. The BaseRepository class inherits the IBaseRepository interface.

* 1. **UyelikRepository**

In this class we will perform all the membership opreations of the application. Operations in this class:

* Add a user
* Add a role to the user
* Enter and exit the system
* Reach the logged-in user's information

We will do this in the following way:

* UserManager<ApplicationUser> UserManager { get; set; }
* IAuthenticationManager AuthenticationManager

Most of the user actions were done using the NIslemSource class in the Helpers folder

* 1. **FakulteRepository**

We performed all the database operations related to the faculty in the FakulteRepository class. In the FacultyRepository class, the following operations will be performed;

* List all faculties
* Display faculty information according to its ID
* Record the faculty
* Faculty update and deletion

* 1. **BolumRepository**

We performed all the database operations related to the department in the BolumRepository class. In the BolumRepository class , the following operations will be performed;

* List all departments
* List the departments of the faculty
* Display department information according to its ID
* Record the department
* Department update and deletion

1. **AREAS-MANAGEMENT**

The Information Transaction user will perform operations through the Management panel. We defined the operations that the Information Transaction user is authorized in the Introduction. We created the Management panel that the Information Transaction user uses. All operations that the Information Transaction user will perform are defined in this panel.

* 1. **YonetimAreaRegistiration**

In the management area, we created the default settings and DonemDers settings in Route configuration. We created the Default settings as follows:

* context.MapRoute(
* "Yonetim\_default",
* "Yonetim/{controller}/{action}/{id}",
* new { action = "Index", id = UrlParameter.Optional } );

We created the Default settings as follows:

* context.MapRoute(
* "Yonetim\_DonemDers",
* "Yonetim/Dersler/{action}/{donemid}/{bolumid}/{dersid}",
* new {action ="Index", controller ="DonemDers", dersid = UrlParameter.Optional } );
  1. **YonetimController and Other User Controllers**

In the YonetimController class, the Information Transaction user will be logged in and out operations of the system. We defined this action as [allowananymous] because the user login process can be done by everyone. We have defined

[Authorize (Roles = "BilgiIslem")] so that only the logged-in Information Transaction user can perform the exit operation.

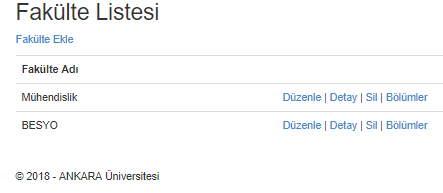


Other users can enter the system after they pass the same securityn protocols.

* 1. **FakulteController**

The class in which the Information Transaction user will perform faculty operations. We have defined

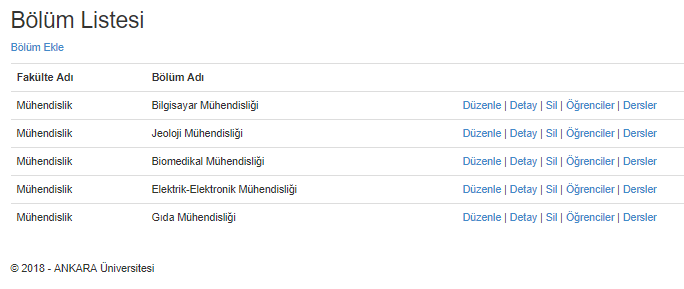
[Authorize (Roles = "BilgiIslem")] so that only IT operations are performed by the user.



* 1. **BolumController**

The class in which the Information Transaction user will perform department operations. We have defined

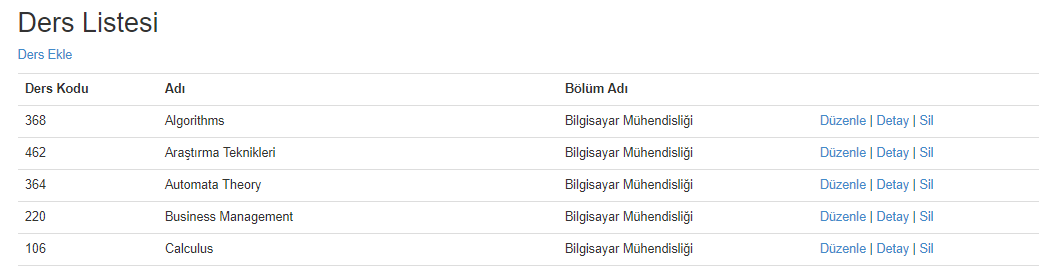
[Authorize (Roles = "BilgiIslem")] so that only IT operations are performed by the user.



* 1. **DersController**

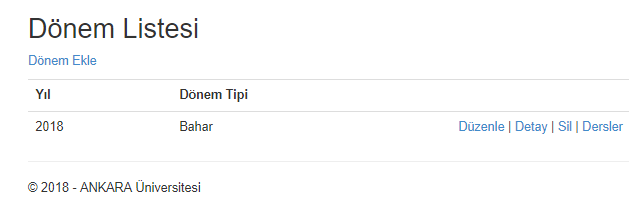
The class in which the Information Transaction user will perform course operations. In this part of the Project IT user will add the course only to department. We have defined

[Authorize (Roles = "BilgiIslem")] so that only IT operations are performed by the user.



* 1. **DonemController**

The class in which the Information Transaction user will perform semester operations. IT user will just add semester. We have defined [Authorize (Roles = "BilgiIslem")] so that only IT operations are performed by the user.



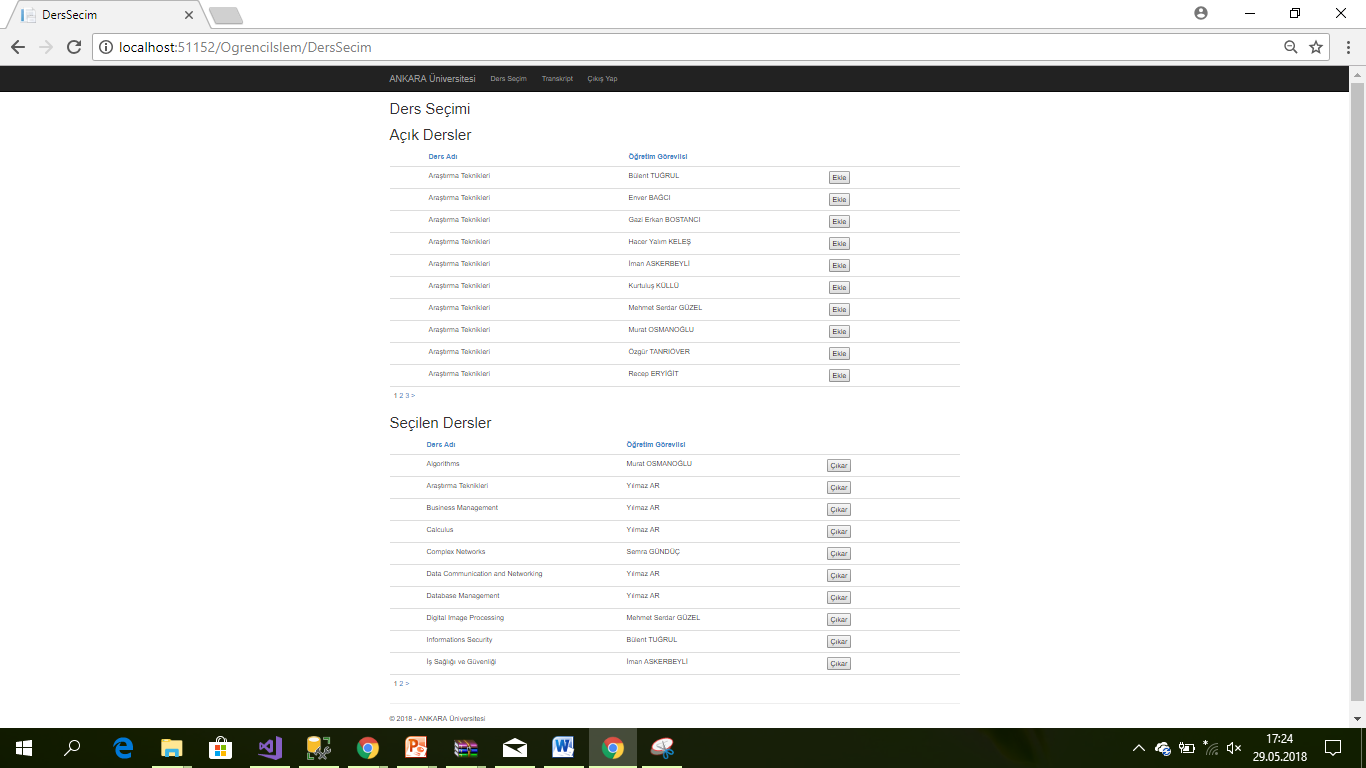
1. **CONTROLLERS**

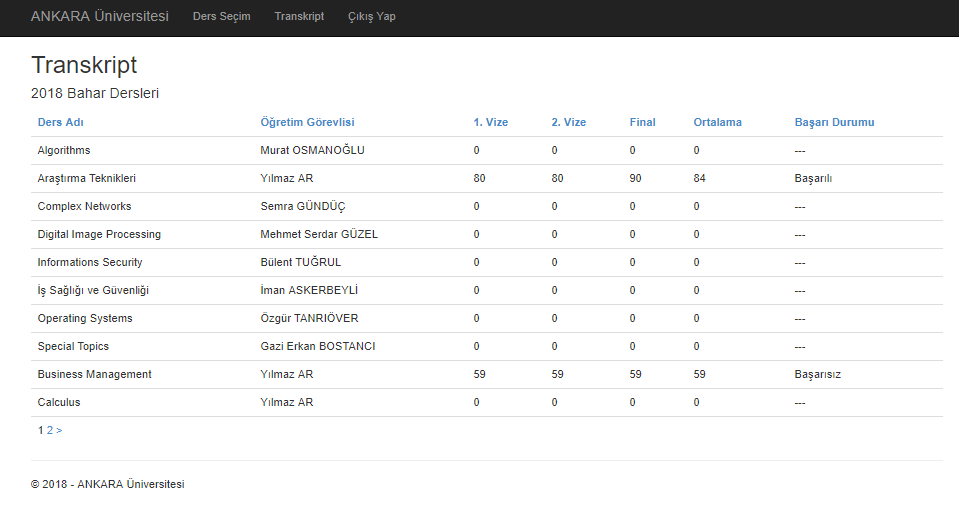
We created this folder for other users (Student and Lecturer) operations.

* 1. **OgrenciIslemController**

We defined [Authorize (Roles = "Ogrenci")] so that only Student operations are performed by the user. The actions that the student can perform through this class are:

* Entering and leaving the system
* Selection of courses opened in the semester
* View Transcript

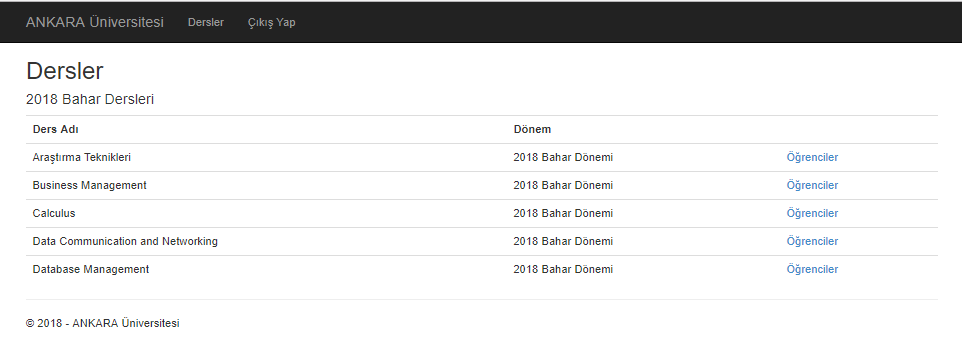


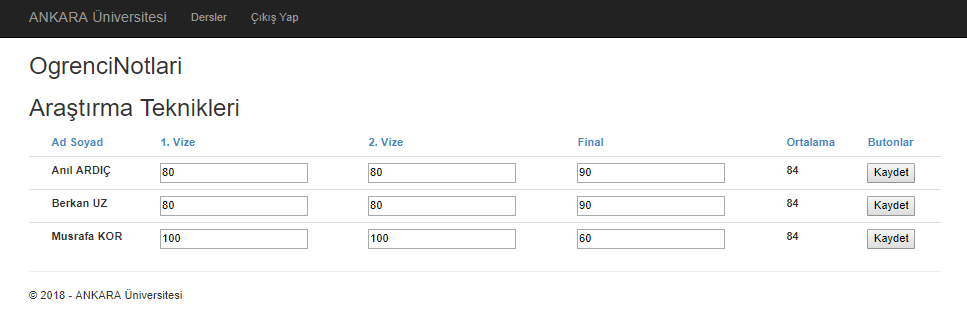


* 1. **OgretimGorevlisiController**

We defined [Authorize (Roles = "OgretimGorevlisi")] so that only Lecturer operations are performed by the user. The actions that the Lecturer can perform through this class are:

* Login-Logout
* View all course and course in semester
* View students taking courses
* Grade and Register operations





1. **Views**

Our views were created according to the operations performed by the controller classes. Since these files are user interface, they will be explained during project presentation. Briefly, this file consists of four components. These are; Home, OgrenciIslem, OgretimGorevlisiIslem, and Shared folders. These components present the operations in our controller classes to the user as an views.

1. **BIBLIOGRAPHY**
2. <https://startbootstrap.com/>
3. <http://ustaderslik.com/kategori.php?k=BOOTSTRAP>
4. <http://aslanbakan.com/tr/blog/javascript>
5. <https://www.dijitalders.com>
6. <http://jquery.sitesi.web.tr/jquery-ajax.html>
7. <https://www.w3schools.com/>
8. <https://stackoverflow.com/>
9. <https://www.asp.net/mvc>
10. <http://cagatayyildiz.com/repository-pattern-nedir/>
11. <http://www.eniskurtayyilmaz.com/json-nedir-ve-c-sharp-ile-json-kullanimi/>
12. <https://www.udemy.com/>