ISTANBUL KEMERBURGAZ UNIVERSITY
INSTITUTE OF SCIENCE
MSc INFORMATİON TECHNOLOGIES

STUDENT TRACKING SYSTEM FOR VOCATIONAL HIGH SCHOOLS IN TURKEY

Istanbul Kemerburgaz University

Presented to Msc. Information Technologies

ÖZGÜR KORKMAZ

SUPERVISOR: ASST. PROF. MERT OZKAYA

ISTANBUL, 2016

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Information Technologies.

Asst. Prof. Mert ÖZKAYA

Supervisor

Examining Committee Members:

Asst. Prof. Mert ÖZKAYA	(Jury)	
Asst. Prof. Ayşe YILMAZER	(Jury)	
Asst. Prof. Oğuzhan CEYLAN	(Jury)	

I certify thatthis thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Asst. Prof. Tuğçe Ballı ALTUĞLU

Head ofDepartment

Approval of the University 25/05/2016

Assoc. Prof. Oğuz BAYAT

Director

Declaration of Own Work

I declare that this thesis 'An investigation into the conservation impact of research published in the scientific literature' is entirely my own work and that where any material could be construed as the work of others, it is fully cited and referenced, and/or with appropriate acknowledgement given.

Signature:

Name of student: Özgür KORKMAZ

Acknowledgements

I would firstly like to thank Asst. Prof. Mert Özkaya for all his input, advice, encouragement, and always being on hand to provide helpful comments.

I am very grateful to all those who piloted and responded and took an interest in the project. Special thanks go to Mr. Volkan Sivri my colleague and my department manager facilitating the interviews at Kağıthane Profilo Vocational High School, and all those staff members who took the time out of a busy schedule to be interviewed.

Thanks must also go to my former student Bahattin Çiniç and my colleague Mehmet Akif Garip who provided helpful comments and to my mother Ayşe Eröyer who has supported me.

Abstract

Technical and vocational high schools prepare students for employment in various occupations and for higher education. Practical training is a big part of the technical and vocational education system and it can have a big impact on students’ development. Students are tracked by technical teachers at work with the guidance of school management and reports are prepared by teachers at each visit. We can easily say that it is better to save data digitally then on paper and benefit from a database management system. However, there are not any specialized online systems to track practical training for vocational high schools in Turkey. The thesis aims at tracking of practical training process in vocational and technical high schools in Turkey and propose a solution in order to digitalize their current processes and overcome the current issues, which are being faced daily due to lack of computerized solution. The application is implemented as a web site, created using LARAVEL framework. The application creates an online student tracking system where the manager manages the system by inserting teachers, students, classes, companies, controlling added feedbacks and the teachers view their students and add feedbacks to the system for each student. Student Tracking System was evaluated in a vocational and technical high school for a month. First, managers and teachers were given a lecture about the web site then it was used for a month. After the insertion of necessary data to system by manager, teachers visited the students at work and added gathered feedbacks to the web site. Both managers and teachers were delighted with the new solution, agreed on that the new system makes the process easier and pointed out that all the recorded data can be very useful for students and their career.

Table of Contents

Table of Figures

Table List

Abbreviations

DBMS - Database Management System

MoNE - Ministry of National Education

IPT – Student Placement Tracking

PC – Personal Computer

UML – Unified Modelling Language

ER – Entity - Relationship

DML – Data Manipulation Language

PHP - Hypertext Preprocessor

MVC – Model – View - Controller

MVC – Object Relational Mapping

URL – Uniform Resource Locator

HTML – HyperText Markup Language

CSS – Cascading Style Sheets

WEB – World Wide Web

1. INTRODUCTION

An information system is software that provides ways to orchestrate data; so it is possible to solve problems related to work of an organization.

Today all the organizations have to cope with big data. Data are main ingredients and are organized in a database. Component parts of an information system detail the rules for what processes can do to data. (Watson, 2006)

A DBMS as a kind of information system combines software and data that makes it possible to orchestrate and analyze data. By storing data in a DBMS rather than as a system file, we can use the DBMS to manage the data more effectively. As the volume of data and the number of users grow, DBMS support becomes mandatory. (Ramakrishnan & Gehrke, 2003)

Having a DBMS between the end user's applications and the database offers important advantages. First, data in the database can be shared among multiple applications or users. Second, the DBMS integrates all user's views of the data into a single embracing data vault. The DBMS makes data management more efficient and effective. (Coronel, Morris, & Rob, 2012)

Technology can play a crucial role in whole education system. Schools need to manage more information day by day. Student registration operations, student transfer operations, marks, document operations, weekly lesson plans and some other operations all need DBMS because of the enormous data at hand.

E-Okul is a web school management information system, which was opened for use in 2007 by MoNE of Turkey. All the info of students from entrance to graduation are recorded in the system. As we look deeply into the system, we can easily notice that there is not any record of practical training period; except year-end marks of students. So first, vocational and technical education will be dwelled on especially focusing on practical training then motivation for using a DBMS system for practical training period will be explained.

1.1 BACKGROUND

It can be said that the vocational and technical education system in Turkey is divided into two parts: Theoretical (school training) and practical (in-company training).

The vocational education system:

• Vocational and technical high schools provide training in more than 130 occupations and leads to the qualification of specialized worker and technician;

• Theoretical education is provided in schools and practical training is provided in various companies;

• Vocational education centers provide mainly informal education.

There are more than 19 different school types for vocational and technical high schools. There are also vocational education centers, Open Education and special private schools. The MoNE is responsible for all vocational and technical high schools. Vocational and technical education is officially co-educational and boys and girls attend schools designed for the other gender. (TÜNCEL, AKYÜZ, ALTIN, & YALÇIN, 2010)

After 8th class, students are placed to various schools according to transition from primary to secondary education system in Turkey. In ninth class, all students in high schools get the same education. After ninth class, there are two kind of school types in the same school for students in technical and vocational high schools to choose: Technical High school and industrial high school.

Branch courses in technical and vocational high schools offers courses towards various professions. In addition, each branch contains some sub-branches. Technical high schools are not included in this generalization. Curriculums of technical high schools are similar with the general high schools as of the general education courses and branch courses of natural sciences branch. (TÜNCEL, AKYÜZ, ALTIN, & YALÇIN, 2010)

Branches in vocational and technical education institutions:

• Industrial and technical branches: Plastics technology, apparel machinery maintenance and repair, Apparel, olive technology, computer-aided industrial modeling, decorative arts, automotive technologies, furniture and decoration, metal technology, machine technology, information technologies, electric technologies, electronic technologies, industrial casting, nourishment technology, construction technology, plastic arts and design, textile technologies, etc.

• Commerce and tourism branches: Radio, cinema and television, public relations and promotion, accounting and finance, insurance trade and risk management, computers, marketing, catering services, accommodation services, travel agency, travel, recreational services, tourism, journalism, office management and secretary, etc.

• Social services branches: Child development and education, Skin care and hairdressing, organization services, etc.

There are about 225 occupational branches in vocational and technical education institutions.

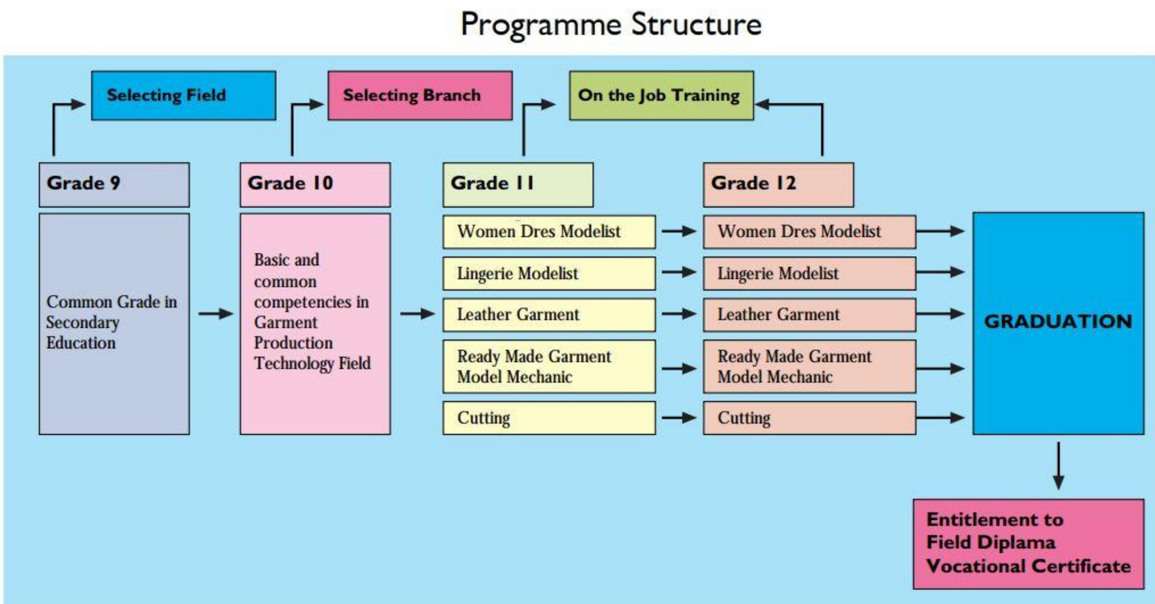

Figure 1: Program Structure

Practical training is 3 days a week during the 12th class for industrial vocational high schools and 5 days a week during the summer break between 11th and 12th class for technical vocational high schools (Figure 1).

The process starts with a meeting between technical manager and the technical teachers. Then, students assigned to teachers according to their distance to school and some other points like number of students to be tracked. If it is possible students assigned to teachers according to their branch to provide better student tracking process. For example, Students of information technologies assigned to teachers of information technologies or students of metal technologies to teachers of metal technologies. Sometimes it is not possible to apply this kind of distribution because of overcrowded students. There should be a fair distribution to prevent any polemic. After getting information and feedback papers from manager; teachers visit assigned students at work according to the predetermined rules at the meeting. The intervals between the visits can be a week, a month, two weeks or even two months. At the visit, teachers ask questions to company manager, person in charge of the student, the student. Then, prepares a report for students and return them to manager to be stored.

1.1.1 Manager

a) Warns head manager about the precautions needed to provide better-planned vocational education.

a) Participate in state employment and vocational education board and provides information to the board.

b) Plans trainings for qualified teachers with the company management.

c) Attends to the board that decides whether the companies are appropriate for the vocational education in specific profession.

d) Carries out the insurance operations of students.

e) Carries out the attendance operations of students.

f) Controls paperwork, which should be given the school management by the teachers and informs the head manager about the troubles encountered during this period.

1.1.2 Teacher

b) Warns manager about the precautions needed to provide better-planned vocational education.

c) Prepares reports about the problems encountered at companies and return them to school management to be used in future development strategies.

d) Guides to the students, companies and qualified teachers.

e) Controls the files of students at each visit.

f) Follows students' development, attendance and discipline and give information to the management.

g) Find out if students are incompatible and give information to the management.

h) Give information to the management about the issues between the school and the company.

i) Returns the terms reports filled by the company.

j) Carries out the paperwork needed for tracking the students at work.

1.1.3 Company

a) Signs a protocol that includes the basics of vocational training in the company and number of students to attend this training.

b) Ensures the vocational training to take place in an appropriate company.

c) Assigns enough qualified instructor to provide training to students.

d) Ensures the attendance of the staff to the pedagogy courses; when number of qualified teachers are not enough.

e) Provides the practicing material.

f) Signs the paperwork, which includes the wage that must be a least percentage 30 of the minimum wage with student's parents and the school.

g) Follows student participation and give information to the school management.

h) Returns the terms reports filled to the school management.

i) Returns the documents of the students to the management at the end of the year.

j) Help teachers about their visit to maintain the vocational training and to prevent problems before arising.

1.1.4 Student

Vocational High schools consist of two different kind of school: Industrial high school and technical high school. The training period is in summer after 11^{th} class for technical students and 3 days in a week during 12^{th} class for industrial students. Every student should perform at least percentage 80 of practical training that belongs to the area of profession and be successful.

1.2 MOTIVATION

Teachers visit assigned students at work after getting the list of the assigned students with some information on paper. These papers include the student's name, class, phone number, company address etc. The intervals between the visits can be a week, a month, two weeks or even two months. At the visit, teachers ask questions to company manager, person in charge of the student and the student; prepares a signed report for each student and return them to manager to be stored.

An online student tracking system will be created to simplify the process and to provide a paper-free solution. The system will have all information about all the members of the system and will reduce paperwork and all the data gathered and stored on database might be used to create career opportunities for students in future. For example, the system will have many information about the students: their pictures, google map links of the companies they work for, their parents' phone numbers, their phone numbers and e-mails. Therefore, it will be easier for teachers to get information about the students during their visit at work. Teachers will be able to see the pictures of the students beforehand, be able to phone to them or their parents if an emergency emerges and all the information can be gathered from their mobile phones or tablets. Teachers will not need to carry around papers.

Teachers have to prepare reports about students at each visit. For instance, let us think that a teacher has five assigned students for a week that means at least five reports a week, twenty a month and two-hundred in a year. So, recording this reports in an online system will reduce the paperwork and creates various opportunities for students: Teachers can see development of students during practical training very easily by inspecting the reports saved online and all the reports gathered can be used for career opportunities.

1.2.1 Statement of Problems with the Existing Systems.

1.2.2 Related Work

In this section, we review works that relate to our solution and discuss in detail how our solution advances. We review other systems that have been developed for student tracking, as well as proposed notions of similarity. We compare these solutions to ours from several perspectives and show how our solution provides better results than most current solutions.

1.2.2.1 Related-work 1 (Bahcelievler Trade Vocational High School Online System)

The system was developed by Bahcelievler Trade Vocational High School Information Systems department to be used by staff and students. The system divided into two systems: Online exam system and student tracking system. The online exam system creates a paperless solution for all the exams in the school. All teachers create their exams using the system, students take the exams online, and the results can be viewed by using the system. Student tracking system gives assigned students' information to teachers and provides all the papers needed by the teacher.

The system has three type of users: Admin also the technical manager adds teachers, students, lessons and classes to the system with unique user names and passwords. Added teachers can create online exams for students and they can view the student list that they have to track and print the papers needed for student tracking. Students can take online exams. The results are recorded.

The tracking system is a web-based monitoring system designed to keep track of students placed in various companies. The system is used in order to track student placements more effectively while providing teachers an easier solution.

The main issue with the system is lack of feedback feature; students' feedbacks about his experiences or companies' feedbacks about the students; also teachers' idea about both the students and companies and other routine monthly feedbacks. Even though; every paper needed by the teacher for the process can be printed from the system, the system does not have a part where the information on the returned papers can be recorded. Although the system provides teachers the information of the assigned students, the information provided lacks important information about students like students' pictures, which is crucial for teachers to have an easier student tracking process.

Another issue is security. The system was scanned with a web vulnerability scanner and a web application scanner. A web vulnerability scanner is used for searching websites and checking for vulnerabilities to PHP attacks. Vulnerable scripts are shown so that they can fixed. A web application scanner is used for checking applications for injections and other vulnerabilities. A few vulnerabilities encountered during the process but let us not forget security means a lot when information about people stored in databases.

1.2.2.2 Related-work 2 (Student Tracker Database)

Open Source Development Group of Georgia Southern University's College of Information Technology is responsible for this project. Students from various computer departments formed the group. The purpose of Student Tracker is to help people in the student employment process to document their activities; help them to know their duties and help them to perform those duties. Student Tracker provides a better and online workflow management.

The system has three type of users: Schools, companies and students. They register to the system with unique user names and passwords. They have particular duties, which involve creating documents of various kinds such as job postings, resumes, and performance assessments. It reduces paperwork by storing data on computer.

Although this system has much better security precautions, a few vulnerabilities encountered. It is a different approach to student tracking and lacks feedback feature again.

1.2.2.3 Related-work 3 (Student Placement Tracking)

The Boise State School of Social Work uses Student Placement Tracking to track all students in fieldwork. The university and companies communicate students in fieldwork using IPT. It was developed by Orem, a Utah-based Alcea Software. The web-based system was developed with the help of Brigham Young University School of Social Work.

IPT manages the information needed by field placement. IPT maintains information about companies, teachers and students and track student assignments. Gathering information and communicating with companies, teachers and Students involved in the program by allowing them to access and update field information using a web-browser is a very simple process.

The system provides communication with both students and teachers and allows manager to create notes regarding the process. Students can look for potential placement opportunities to find a better match for their qualifications. IPT helps to reduce paper wastage and provides the student access to their fieldwork information. IPT allows companies to describe their fieldwork in order to match students' qualifications. Teachers can access information regarding their students.

IPT supports a different kind of feedback system. It allows manager to create notes regarding the process. The processes are different because of the differences in tracking the practical education. For example, students are visited at work at intervals and then their feedbacks are recorded at each visit in technical and vocational high schools in Turkey.

Another issue is security. The system was scanned with a web vulnerability scanner and a web application scanner like other related works before. A few vulnerabilities encountered during the process but security means a lot when information about people stored in databases.

1.2.3 The New System

The Student Tracking system is a web-based data management system that will allow the vocational high schools to better manage student, teacher and company data to facilitate the process. It will also allow teachers the ability to view and complete the process and evaluation online, without the use of any specific software. All that is needed is a computer or a mobile device and web browser and the system can be accessed from any location. This system will enhance the quality of the teacher/student supervisory experience and facilitate better communication. In addition to these benefits, student-tracking system also offers the following benefits:

- Provides a paperless solution for tracking and recording students' data.
- Allows students to research potential field opportunities.
- Information about companies can be accessed and updated very easily
- Allows technical instructor to follow students' developments.
- Provides teachers to easily access student information.
- Allows teachers access to the feedbacks in a paperless system
- Increases communication between the School and companies.

The new system aims to solve the problems encountered in the existing systems while becoming the first web-based student tracking system to be used in vocational high schools in Turkey.

The first and the probably the most important issue is about feedback feature. Most of the systems reviewed either lack feedback feature or the feature is different. For example, (b-atml.com/) lacks the feedback part. Although some systems have the feedback feature like IPT, the feature is undetailed and different because of the differences in the educational systems. Vocational education in Turkey must have a unique student tracking software and our system is going to be designed to meet all the requirements needed.

The second issue is the lack of information needed for all the users in the system. For example, parents' phone numbers are not included in any of the system. However, it is very important for the teachers when they cannot find the students on the field. Another important information deficit is google-map info of the company. There is also other important information needed because of the uniqueness of the vocational education in Turkey.

The third issue is that most of the system out there are created with old technologies. That is why there is some performance, security and visual quality issues. Our system is going to be PHP, LARAVEL based, and LARAVEL's own security precautions, visual standards and performance features are going to be implemented to the system. The system was scanned with a web vulnerability scanner and a web application scanner and no vulnerabilities encountered during the process.

2. ANALYSIS

2.1 FUNCTIONAL REQUIREMENTS

2.1.1 Description of the System Functionalities

Student tracking system is a replacement for the old-fashioned tracking system for vocational high schools, which depend on paper work for tracking students and users' information. Student tracking system will provide an advanced mechanism and will make it easy to track the students.

2.1.2 Actor Description

2.1.2.1 Technical Manager

Manager is also a teacher or a technical manager who is assigned by the school director to manage the student tracking process. She has the responsibility of adding, deleting and modifying the teachers, students or companies to the system. She has also the responsibility of viewing feedback and warning teachers if something missing.

- Manager should be able to add, modify and delete teachers, students and companies.
- Can define username and password to the added teachers.
- Can get the information of any member.
- Can view all the feedbacks.
- Can sign the feedback as controlled or not controlled.

2.1.2.2 Teachers

Teacher is assigned by the technical manager to follow the students closely on the job training. She has the responsibility of adding, deleting and modifying the feedbacks to the system. She can also view the students related to him with their detailed information.

- Can view the related students.
- Can view details of a specific student.

- Can fill up; delete/modify feedback (If it is not checked by the manager).

2.1.3 Use Case Diagram

The use case describes the system's behavior under various conditions as it responds to a request from the primary actor. The primary actor interacts with the system and the system responds. Different scenarios can happen depending on the requests made and terms surrounding the requests. (Cockburn, 2001)

Use cases can be written in many forms: Text form, flow charts, sequence charts or programming languages. People do not need any special training to use them. (Cockburn, 2001)

A team can use the use case form to document the actual requirements. Another team might later use the former document to finalize design. They might do this for a system of a large company, or a small piece of an application. Both work can be done with the same knowledge level. (Cockburn, 2001)

Use cases are used to document an organization's business processes. The stakeholders are the company shareholders, customers, vendors, and government regulatory agencies. The primary actors will include the company's customers and perhaps their suppliers. When the use cases record behavioral requirements for a piece of software, the system under discussion is the computer program. The stakeholders are the people who use the program, the company owning it, government regulatory agencies, and other computer programs. The primary actor will be the user sitting at the computer screen or another computer system. (Cockburn, 2001) (Cockburn, 2001)

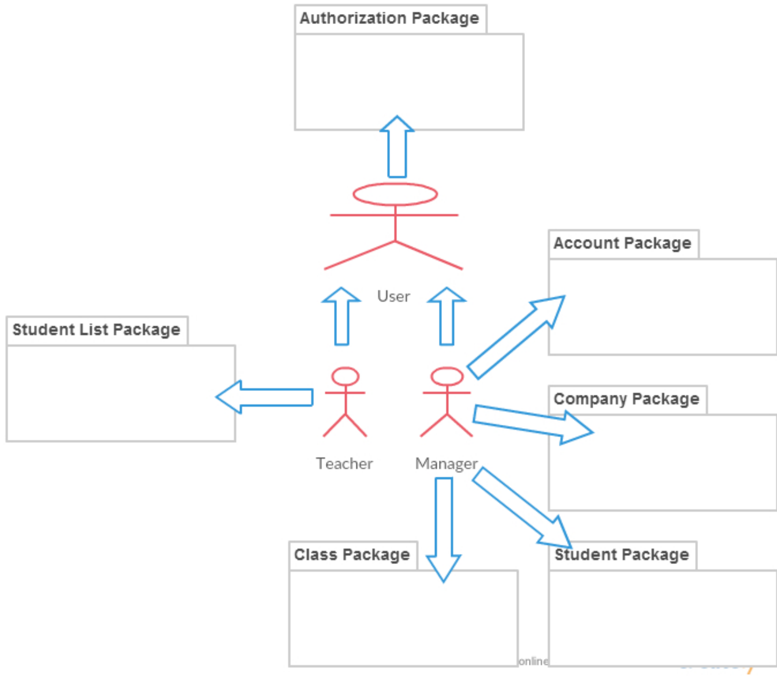

Figure 2: Use case packages of the system.

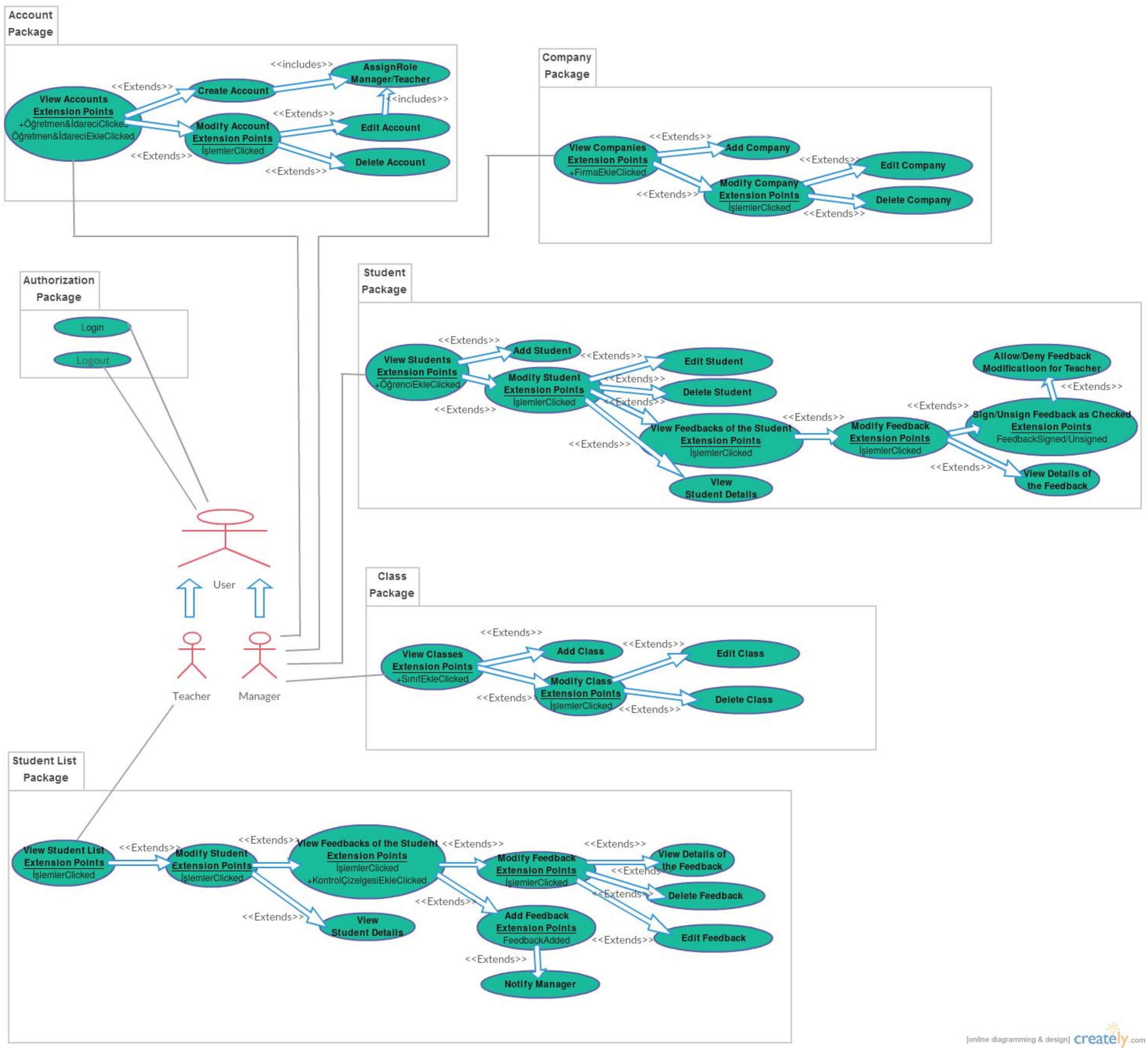

Figure 3: Use case diagram of the system

2.1.3.1 Use Case Description

2.1.3.1.1 Authorization Package

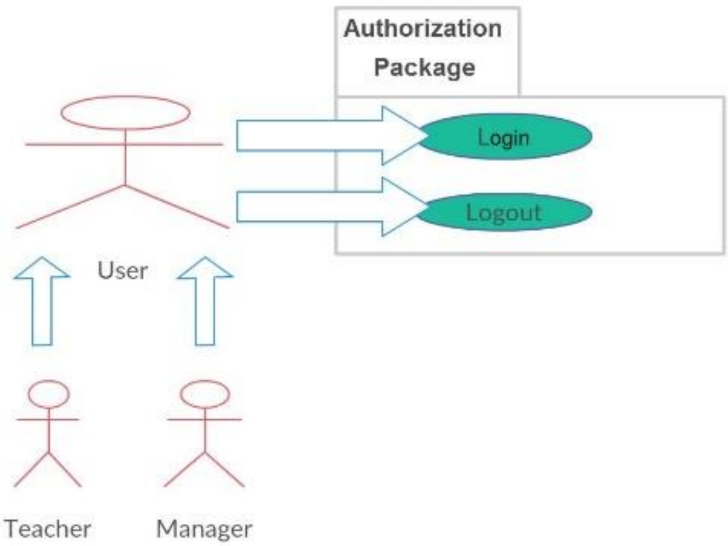

Figure 4: Authorization package

Use Case: Login
Brief Description: The user logins with an account.
Primary Actors: Manager/Teacher
Secondary Actors: None
Preconditions: The Use-Case starts when: • A user tries to log on to the system.

Main Flow: 1. The user enters e-mail and password then clicks “Giriş”. [Invalid Input] 2. The System logins.
Post condition: The Use-Case terminates when:

- The user logs in.

Alternative Flows:

[Invalid Input]

- The User tries to submit the page with invalid input in one of the boxes.
- The System displays an error message prompting the user to re-enter e-mail and password.

Use Case: Log out
Brief Description: The user logs out of the system.
Primary Actors: Manager/Teacher
Secondary Actors: None
Preconditions: The Use-Case starts when: • A User is in the system and wants to logs out.

Main Flow:

1. The user member navigates to his or her name at the top-right of the page and clicks the name.
2. The System displays “Çıkış” button.
3. The user cliks “çıkış” button.
4. The user is logged out of the system.

Post condition:

The Use-Case terminates when:

- The user is logged out of the system.

2.1.3.1.2 Account Package

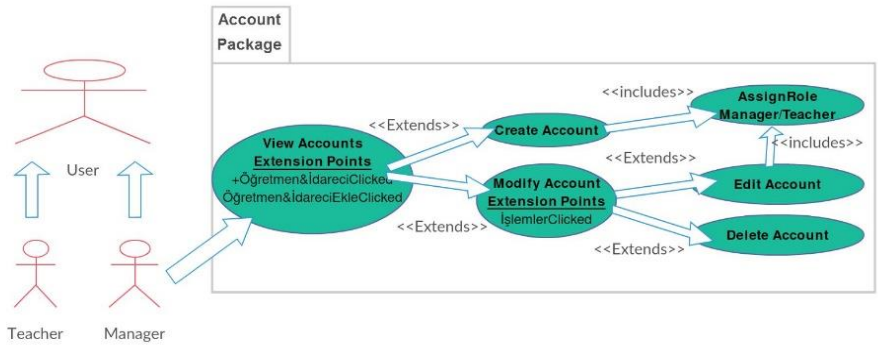

Figure 5: Account package

Use Case: View Accounts
Brief Description: The manager views all the accounts registered.
Primary Actors: Manager
Secondary Actors: None
Preconditions: The Use-Case starts when: • The user member clicks on “Ögretmen&Idareci İşlemleri”.

Main Flow: 1. The user member clicks on “Ögretmen&Idareci İşlemleri”. 2. The system displays “Ögretmen&Idareci İşlemleri” page Extension points: • (Create Account) • (Modify Account)
Post condition: The Use-Case terminates when: • The system displays “Ögretmen&Idareci İşlemleri” page.

Extension Use Case: Create Account
Brief Description: The user creates a new account.
Primary Actors: Manager
Secondary Actors: Manager/Teacher
Preconditions: The Use-Case starts when: • The manager clicks “+Öğretmen&İdareci” button at the bottom-right of the page

Main Flow: 1. The manager clicks “+Öğretmen&İdareci” button at the bottom-right of the page. 2. The system displays “Öğretmen&İdareci Ekle” page. 3. The User enters the following user details and clicks the “Kaydet” button. [Invalid Input] 3.1. Ad 3.2. Soyad 3.3. E-Posta 3.4. Telefon Numarası 3.5. Şifre 3.6 Rol [Include Assign Role] 4. The account is created by the system.
Post condition: The Use-Case terminates when: • An account is created by the System.
Alternative Flows: [Invalid Input] • The User tries to submit the page with invalid input in one of the boxes. • The System displays an error message prompting the user to re-enter the input.

Include Use Case: Assign Role
Brief Description: The user assigns a role to a user.
Primary Actors: Manager
Secondary Actors: Manager/Teacher
Preconditions: The Use-Case starts when: • The system displays drop-down list that includes “Öğretmen” and “Yönetici”.

Main Flow: 1. The manager clicks the drop-down list and chooses either “Ögretmen” or “Yonetici”.
Post condition: The Use-Case terminates when: • The role is chosen.

Extension Use Case: Modify Account
Brief Description: The manager clicks related “İşlemler” button.
Primary Actors: Manager
Secondary Actors: Manager/Teacher
Preconditions: The Use-Case starts when: • The user member clicks on related “İşlemler” button.

Main Flow: 1. The user member clicks on related “İşlemler” button. 2. The System displays “Düzenle” and “Sil” buttons in a drop-down list. 3. The user clicks one of them. Extension points: • (Edit Account) • (Delete Account)

Post condition: The Use-Case terminates when: • The user clicks “Düzenle” or “Sil”.

Extension Use Case: Edit Account
Brief Description: The manager edits an account.
Primary Actors: Manager
Secondary Actors: Manager/Teacher
Preconditions: The Use-Case starts when: • The system displays “Ögretmen&Idareci Düzenle” page

Main Flow: 1. The User edits the following user details and clicks the “Kaydet” button. [Invalid Input] 1.1. Ad 1.2. Soyad 1.3. E-Posta 1.4. Telefon Numarası 1.5. Şifre 1.6 Rol [Include Assign Role] 2. The system saves user details.
Post condition: The Use-Case terminates when: • An account is edited by the System.
Alternative Flows: [Invalid Input] • The User tries to submit the page with invalid input in one of the boxes. • The System displays an error message prompting the user to re-enter the input.

Extension Use Case: Delete Account
Brief Description: The manager deletes an account.
Primary Actors: Manager
Secondary Actors: Manager/Teacher
Preconditions: The Use-Case starts when: • The user clicks “Sil” button.

Main Flow: 1. The user clicks “Sil” button. 2. The system deletes the account.
Post condition: The Use-Case terminates when: • An account is deleted by the System.

2.1.3.1.3 Company Package

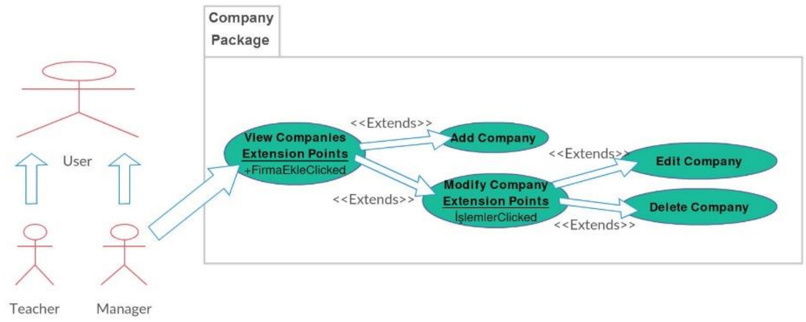

Figure 6: Company package

Use Case: View Companies
Brief Description: The manager views all the companies registered.
Primary Actors: Manager
Secondary Actors: None
Preconditions: The Use-Case starts when: • The user member clicks on “Firma İşlemleri”.

Main Flow: 1. The User member clicks on “Firma İşlemleri”. 2. The System displays “Firma İşlemleri” page Extension points: • (Add Company) • (Modify Company)
Post condition: The Use-Case terminates when: • The system displays “Firma İşlemleri” page.

Extension Use Case: Add Company
Brief Description: The user adds a new company.
Primary Actors: Manager
Secondary Actors: None
Preconditions: The Use-Case starts when: • The system displays “Firma Ekle” page.

Main Flow: 1. The system displays “Firma Ekle” page. 2. The User enters the following user details and clicks the “Kaydet” button. [Invalid Input]

2.1. İsim 2.2. Adres 2.3. Map Link 2.4. Yönetici Adı 2.5. Yönetici Telefon 2.6 Yönetici E-Posta 3. A company is created by the System.
Post condition: The Use-Case terminates when: • A company is created by the System.
Alternative Flows: [Invalid Input] • The User tries to submit the page with invalid input in one of the boxes. • The System displays an error message prompting the user to re-enter the input.

Extension Use Case: Modify Company
Brief Description: The manager clicks related “İşlemler” button.
Primary Actors: Manager
Secondary Actors: None
Preconditions: The Use-Case starts when: • The user member clicks on related “İşlemler” button.

Main Flow:
1. The user member clicks on related “İşlemler” button. 2. The System displays “Düzenle” and “Sil” buttons in a drop-down list. 3. The user clicks one of them. Extension points: • (Edit Company) • (Delete Company)

Post condition: The Use-Case terminates when • The user clicks “Firma Ekle” or “Firma Sil”

Extension Use Case: Edit Company
Brief Description: The manager edits a company.
Primary Actors: Manager
Secondary Actors: None
Preconditions: The Use-Case starts when: • The system displays “Firma Düzenle” page

Main Flow: 1. The User edits the following company details and clicks the “Kaydet” button. [Invalid Input] 1.1. İsim 1.2. Adres 1.3. Map Link 1.4. Yönetici Adı 1.5. Yönetici Telefon 1.6 Yönetici E-Posta 2. The system saves the edited company details.
Post condition: The Use-Case terminates when: • A company is edited by the System.
Alternative Flows: [Invalid Input] • The User tries to submit the page with invalid input in one of the boxes. • The System displays an error message prompting the user to re-enter the input.

Extension Use Case: Delete Company
Brief Description: The manager deletes a company.
Primary Actors: Manager
Secondary Actors: None
Preconditions: The Use-Case starts when: • The system displays “Sil” button in the drop-down list.

Main Flow: 1. The system displays “Sil” button in the drop-down list. 2. The user chooses “Sil” button 3. The system deletes the company.
Post condition: The Use-Case terminates when: • A company is deleted by the System.

2.1.3.1.4 Student Package

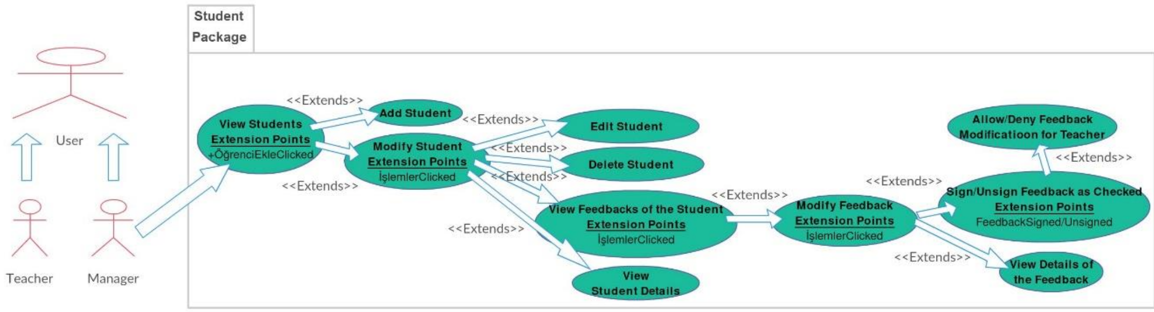

Figure 7: Student package

Use Case: View Students
Brief Description: The manager views all the students registered.
Primary Actors: Manager
Secondary Actors: None
Preconditions: The Use-Case starts when: • The user member clicks on “Öğrenci İşlemleri”.

Main Flow:
1. The user member clicks on “Öğrenci İşlemleri”. 2. The system displays “Öğrenci İşlemleri” page Extension points: • (Add Student) • (Modify Student)
Post condition: The Use-Case terminates when: • The system displays “Öğrenci İşlemleri” page.

Extension Use Case: Add Student
Brief Description: The user adds a new student.
Primary Actors: Manager
Secondary Actors: None
Preconditions: The Use-Case starts when: • The system displays “Öğrenci Ekle” page.

Main Flow:
1. The system displays “Öğrenci Ekle” page. 2. The User enters the following student details and clicks the “Kaydet” button. [Invalid Input]

2.1. Ad 2.2. Soyad 2.3. E-Posta 2.4. Okul Numarası 2.5. Telefon Numarası 2.6. Veli Telefon Numarası 2.7. Staj Başlama Tarihi 2.8. Staj Bitiş Tarihi 3. The system creates the student.
Post condition: The Use-Case terminates when: • A student is created by the System.
Alternative Flows: [Invalid Input] • The User tries to submit the page with invalid input in one of the boxes. • The System displays an error message prompting the user to re-enter the input.

Extension Use Case: Modify Student
Brief Description: The manager clicks related "İşlemler" button.
Primary Actors: Manager
Secondary Actors: None
Preconditions: The Use-Case starts when: • The user member clicks on related "İşlemler" button.

Main Flow:
1. The user member clicks on related "İşlemler" button. 2. The System displays "Düzenle" and "Kontrol Edildi/Edilmedi Olarak İşaretle" buttons in a drop-down list. 3. The user clicks one of them.

Extension points: • (Detay) • (Kontrol Edildi/Edilmedi Olarak İşaretle)
Post condition: The Use-Case terminates when: • The user clicks either "Detay" or "Kontrol Edildi/Edilmedi Olarak İşaretle" button.

Extension Use Case: Edit Student
Brief Description: The manager edits a student.
Primary Actors: Manager
Secondary Actors: None
Preconditions: The Use-Case starts when: • The system displays "Öğrenci Düzenle" page

Main Flow: 1. The User edits the following student details and clicks the "Kaydet" button. [Invalid Input] 1.1. Ad 1.2. Soyad 1.3. E-Posta 1.4. Okul Numarası 1.5. Telefon Numarası 1.6. Veli Telefon Numarası 1.7. Staj Başlama Tarihi 1.8. Staj Bitiş Tarihi 2. The system saves the student details.
Post condition: The Use-Case terminates when: • The system saves the student details.

Alternative Flows:
[Invalid Input] • The User tries to submit the page with invalid input in one of the boxes. • The System displays an error message prompting the user to re-enter the input.

Extension Use Case: Delete Student
Brief Description: The manager deletes a student.
Primary Actors: Manager
Secondary Actors: None
Preconditions: The Use-Case starts when: • The user clicks “Sil” button in the drop-down list.

Main Flow:
1. The user clicks “Sil” button in the drop-down list. 2. The student is deleted.
Post condition: The Use-Case terminates when: • A student is deleted by the System.

Extension Use Case: View Feedbacks of the Student
Brief Description: The manager views the feedbacks of a student.
Primary Actors: Manager
Secondary Actors: None
Preconditions: The Use-Case starts when: • The user clicks “Kontrol Çizelgesi”.

Main Flow:
1. The user member clicks on “Kontrol Çizelgesi”.

2. The System displays “Kontrol Çizelgesi” page. Extension points: • (Modify Feedback)
Post condition: The Use-Case terminates when: • The System displays “Kontrol Çizelgesi” page.

Extension Use Case: Modify Feedback
Brief Description: The manager clicks related “İşlemler” button.
Primary Actors: Manager
Secondary Actors: None
Preconditions: The Use-Case starts when: • The system displays “Kontrol Çizelgesi” page

Main Flow: 1. The user member clicks on related “İşlemler” button. 2. The System displays “Detay” and “Kontrol Edildi/edilmedi olarak işaretle” buttons in a drop-down list. 3. The user clicks one of them. Extension points: • (Sign/Unsign Feedback as Checked) • (View Details of the Feedback)
Post condition: The Use-Case terminates when: • The user clicks “işlemler” button.

Extension Use Case: Sign/Unsign Feedback as Checked
Brief Description: The manager signs or unsigns the feedback as checked.
Primary Actors: Manager
Secondary Actors: None
Preconditions: The Use-Case starts when: • The user clicks related “İşlemler” button.

Main Flow: 1. The user clicks related “İşlemler” button. 2. The system displays “Kontrol Edildi/Edilmedi” and “Detay” drop-down list. 3. The user clicks “Kontrol Edildi” if the feedback is controlled or “Kontrol Edilmedi” if the feedback is not controlled.
Post condition: The Use-Case terminates when: • The manager chooses “Kontrol Edildi/Edilmedi”.

Extension Use Case: View Details of the Feedback
Brief Description: The manager views the details of the feedback.
Primary Actors: Manager
Secondary Actors: None
Preconditions: The Use-Case starts when: • The user clicks related “İşlemler” button.

Main Flow: 1. The user clicks related “İşlemler” button. 2. The system displays “Kontrol Edildi/Edilmedi” and “Detay” drop-down list. 3. The user clicks “Detay”. 4. The system displays view the details of the feedback.
Post condition:

The Use-Case terminates when: • The details of the feedback is displayed.

Extension Use Case: Allow/Deny Feedback Modification for Teacher
Brief Description: The manager allows or denies feedback modification rights for a teacher.
Primary Actors: Manager
Secondary Actors: None
Preconditions: The Use-Case starts when: • The user clicks “Kontrol Edildi” or “Kontol Edilmedi”.

Main Flow: 1. The user clicks “Kontrol Edildi” or “Kontrol Edilmedi”. 2. The system allows teacher users to edit feedback if it is signed “Kontrol Edilmedi” and denies if it is signed “Kontrol Edildi”.
Post condition: The Use-Case terminates when: • The details of the feedback is displayed.

Extension Use Case: View Student Details
Brief Description: The manager views the details of a student.
Primary Actors: Manager
Secondary Actors: None
Preconditions: The Use-Case starts when: • The user clicks related “İşlemler” button.

Main Flow: 1. The user clicks related “İşlemler” button. 2. The system displays “Kontrol Çizelgesi”, “Detay”, Düzenle”, “Sil”.

3. The user clicks “Detay”. 4. The system displays view the details of the student.
Post condition: The Use-Case terminates when: • The details of the student is displayed.

2.1.3.1.5 Class Package

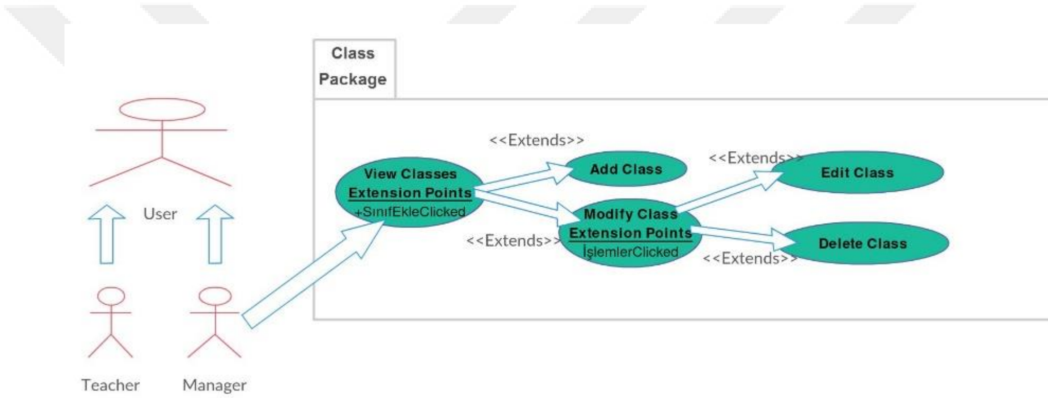

Figure 8: Class package

Use Case: View Classes
Brief Description: The manager views all the classes registered.
Primary Actors: Manager
Secondary Actors: None
Preconditions: The Use-Case starts when: • The User member clicks on “Sınıf İşlemleri”.

Main Flow: 1. The User member clicks on “Sınıf İşlemleri”. 2. The System displays “Sınıf İşlemleri” page Extension points: • (Add Class)

• (Modify Class)
Post condition The Use-Case terminates when: • The system displays "Sınıf İşlemleri" page.

Extension Use Case: Add Class
Brief Description: The user adds a new class.
Primary Actors: Manager
Secondary Actors: None
Preconditions: The Use-Case starts when: • The user clicks "Sınıf Ekle".

Main Flow: 1. The system displays "Sınıf Ekle" page. 2. The User enters the following user details and clicks the "Kaydet" button. [Invalid Input] 2.1. İsim
Post condition: The Use-Case terminates when: • A class is created by the System.
Alternative Flows: [Invalid Input] • The User tries to submit the page with invalid input in one of the boxes. • The System displays an error message prompting the user to re-enter the input.

Extension Use Case: Modify Class
Brief Description: The manager clicks related "İşlemler" button.
Primary Actors: Manager
Secondary Actors: None

Preconditions: The Use-Case starts when: • The user member clicks on related “İşlemler” button.

Main Flow: 1. The user member clicks on related “İşlemler” button. 2. The System displays “Düzenle” and “Sil” buttons in a drop-down list. 3. The user clicks one of them. Extension points: • (Edit Class) • (Delete Class)
Post condition: The Use-Case terminates when: • The user clicks either “Düzenle” or “Sil” button.

Extension Use Case: Edit Class
Brief Description: The manager edits a class.
Primary Actors: Manager
Secondary Actors: None
Preconditions: The Use-Case starts when: • The system displays “Sınıf Düzenle” page

Main Flow: 1. The User edits the following class details and clicks the “Kaydet” button. [Invalid Input] 1.1. İsim
Post condition: The Use-Case terminates when: • A class is edited by the System.

Alternative Flows: [Invalid Input] • The User tries to submit the page with invalid input in one of the boxes. • The System displays an error message prompting the user to re-enter the input.

Extension Use Case: Delete Class
Brief Description: The manager deletes a class.
Primary Actors: Manager
Secondary Actors: None
Preconditions: The Use-Case starts when: • The user clicks “Sil” button in the drop-down list.
Main Flow: 1. The user clicks “Sil” button in the drop-down list. 2. The class is deleted
Post condition: The Use-Case terminates when: • A class is deleted by the System.

2.1.3.1.6 Student List Package

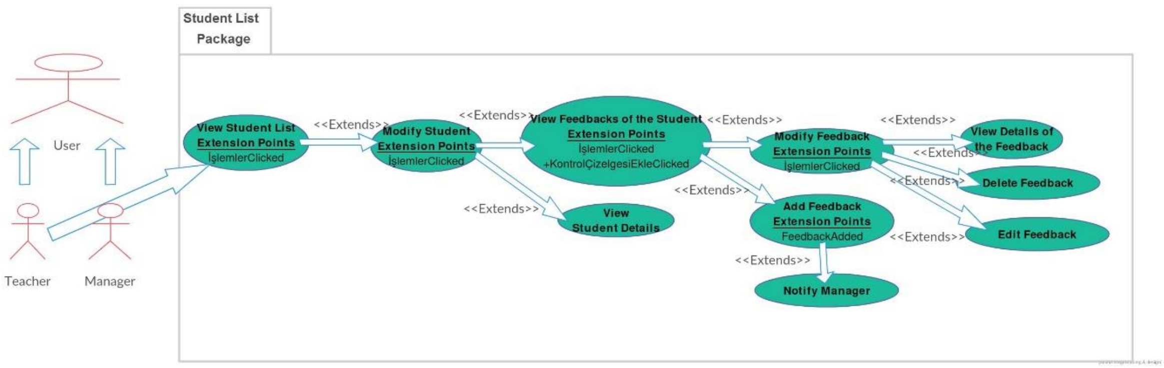

Figure 9: Student list package

Use Case: View Student List
Brief Description: The teacher views students assigned to him/her.
Primary Actors: Teacher
Secondary Actors: None
Preconditions: The Use-Case starts when: • A user is logged on to the System.

Main Flow: 1. The User member clicks on "Öğrenci İşlemleri". 2. The System displays "Öğrenci İşlemleri" page Extension points: • (Modify Students)
Post condition: The Use-Case terminates when: • The system displays "Öğrenci İşlemleri" page.

Extension Use Case: Modify Student
Brief Description: The manager clicks related "İşlemler" button.
Primary Actors: Teacher
Secondary Actors: None
Preconditions: The Use-Case starts when: • The system displays "Öğrenci İşlemleri" page

Main Flow: 1. The user member clicks on related "İşlemler" button. 2. The System displays "Kontrol Çizelgesi" and "Detay" buttons in a drop-down list. 3. The user clicks one of them.

Extension points: • (View Feedback of the Student) • (View Student Details)
Post condition: The Use-Case terminates when: • The user clicks either “Detay” or “Kontrol Çizelgesi” button.

Extension Use Case: View Feedback of the Student.
Brief Description: The teacher views the feedbacks of the student.
Primary Actors: Teacher
Secondary Actors: None
Preconditions: The Use-Case starts when: • A user is logged on to the System.

Main Flow:
1. The User member clicks on “Kontrol Çizelgesi”. 2. The System displays “Kontrol Çizelgesi İşlemleri” page Extension points: • (Modify Feedback) • (Add Feedback)
Post condition: The Use-Case terminates when: • The system displays “Kontrol Çizelgesi İşlemleri” page.

Extension Use Case: Modify Feedback
Brief Description: The manager clicks related “İşlemler” button.
Primary Actors: Teacher
Secondary Actors: None
Preconditions:

The Use-Case starts when: • The user clicks related “İşlemler”.

Main Flow: 1. The user member clicks on related “İşlemler” button. 2. The System displays “Detay”, “Sil” and “Düzenle” buttons in a drop-down list. 3. The user clicks one of them. Extension points: • (View Details of the Feedback) • (Delete Feedback) • (Edit Feedback)
Post condition: The Use-Case terminates when: • The user clicks either “Detay”, “Sil” and “Düzenle” button.

Extension Use Case: View Details of the Feedback
Brief Description: The teacher views details of the feedback.
Primary Actors: Teacher
Secondary Actors: None
Preconditions: The Use-Case starts when: • The user clicks “Detay” button.

Main Flow: 1. The user member clicks on “Detay” button. 2. The System displays “Detaylı Kontrol Çizelgesi” page
Post condition: The Use-Case terminates when: • System displays “Detaylı Kontrol Çizelgesi” page.

Extension Use Case: Delete Feedback
Brief Description: The teacher deletes the feedback.
Primary Actors: Teacher
Secondary Actors: None
Preconditions: The Use-Case starts when: • The user clicks “Sil” button.

Main Flow: 1. The user member clicks on “Sil” button. 2. The System deletes the related feedback.
Post condition: The Use-Case terminates when: • The feedback is deleted.

Extension Use Case: Edit Feedback
Brief Description: The teacher edits the feedback.
Primary Actors: Teacher
Secondary Actors: None
Preconditions: The Use-Case starts when: • The user clicks “Düzenle” button.

Main Flow: 1. The user member clicks on “Düzenle” button. 2. The User edits the following feedback details and clicks the “Kaydet” button. [Invalid Input] 2.1. İşe İlgi 2.2. Katılım/Devam 2.3. İlerleme

2.4. Firma Görüşü 2.5. Öğretmen Görüşü
Post condition: The Use-Case terminates when: • The feedback is edited.
Alternative Flows: [Invalid Input] • The User tries to submit the page with invalid input in one of the boxes. • The System displays an error message prompting the user to re-enter the input

Use Case: Add Feedback
Brief Description: The teacher adds a feedback.
Primary Actors: Teacher
Secondary Actors: None
Preconditions: The Use-Case starts when: • The user clicks "+Kontrol Çizelgesi Ekle" button.

Main Flow: 1. The user member clicks on "+Kontrol Çizelgesi Ekle" button. 2. The User edits the following feedback details and clicks the "Kaydet" button. [Invalid Input] 2.1. İşe İlgi 2.2. Katılım/Devam 2.3. İlerleme 2.4. Firma Görüşü 2.5. Öğretmen Görüşü Extension points: • (Notify Manager)
Post condition: The Use-Case terminates when:

• The feedback is added.
Alternative Flows: [Invalid Input] • The User tries to submit the page with invalid input in one of the boxes. • The System displays an error message prompting the user to re-enter the input

Extension Use Case: Notify Manager
Brief Description: The manager is notified when a feedback is added.
Primary Actors: Teacher
Secondary Actors: Manager
Preconditions: The Use-Case starts when: • The feedback is added.

Main Flow: 1. The Teacher adds a feedback. 2. The System notifies the manager in the main page.
Post condition: The Use-Case terminates when: • The system notifies the manager.

Extension Use Case: View Student Details
Brief Description: The manager views the details of a student.
Primary Actors: Manager
Secondary Actors: None
Preconditions: The Use-Case starts when: • The user clicks related “İşlemler” button.

Main Flow: 1. The user clicks related "İşlemler" button. 2. The system displays "Detay". 3. The user clicks "Detay". 4. The system displays view the details of the student.
Post condition: The Use-Case terminates when: • The details of the student is displayed.

2.1.4 Class Diagram

Class diagram is created simply by reading the class names and their operations off the interaction diagrams.

Because class diagram gathers class operations and attributes in one place, it is easier to size up the relative complexity of classes in the system. The number of operations in a class correlates with the amount of responsibility handled by the class. If some classes have considerably greater number of operations than the other classes, they should be examined for undiscovered classes and misplaced responsibilities. (Marsic, 2012)

MVC is the separation of model, view and controller. It is a way to use when designing classes to avoid mixing code from the other categories. For example, a data grid view should present data once shown but it should not have code on where to retrieve the data. In the same way, while it may have a function to sum up a column, the controller should do the calculation. As can be understood, separation of responsibilities allows flexibility and maintainability. Therefore, I followed rules of MVC model designing the classes in the system.

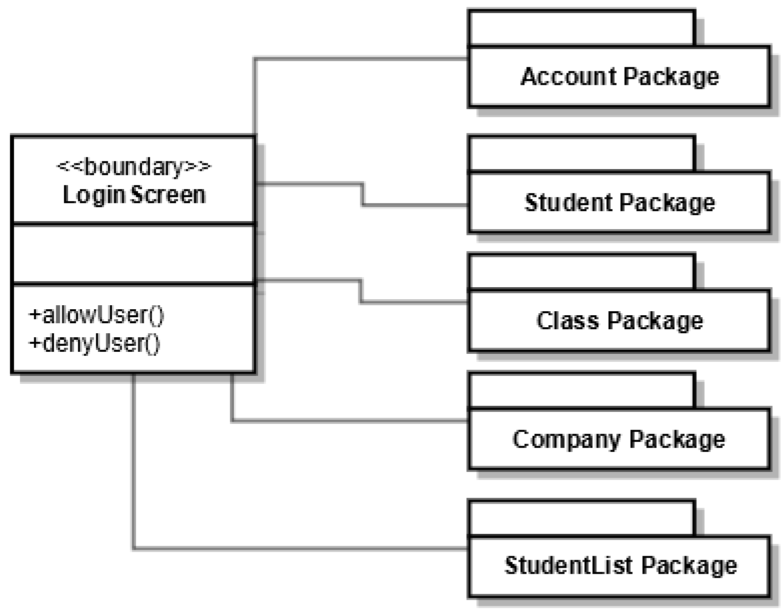

Figure 10: Class diagram packages of all system

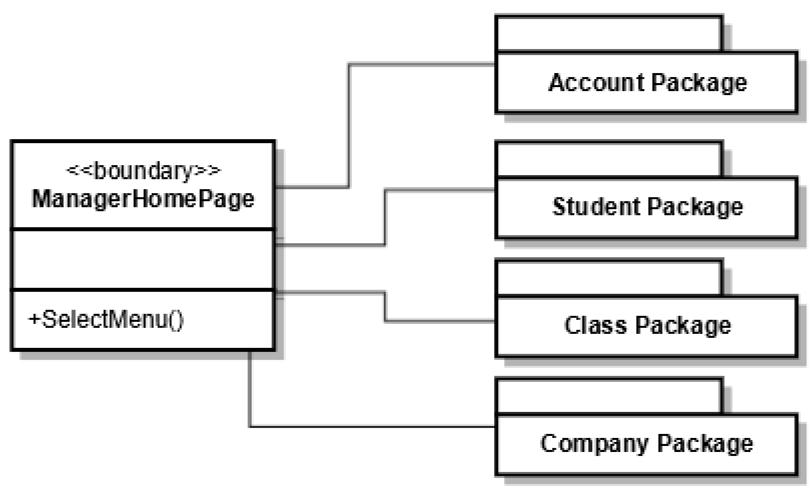

Figure 11: ManagerHomePage class diagram packages of the system

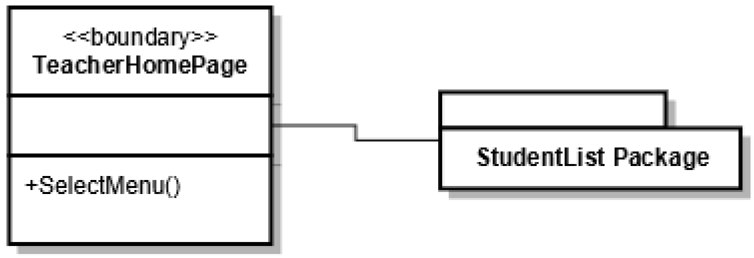

Figure 12: TeacherHomePage class diagram packages of the system

2.1.4.1 ManagerHomePage Class Diagram

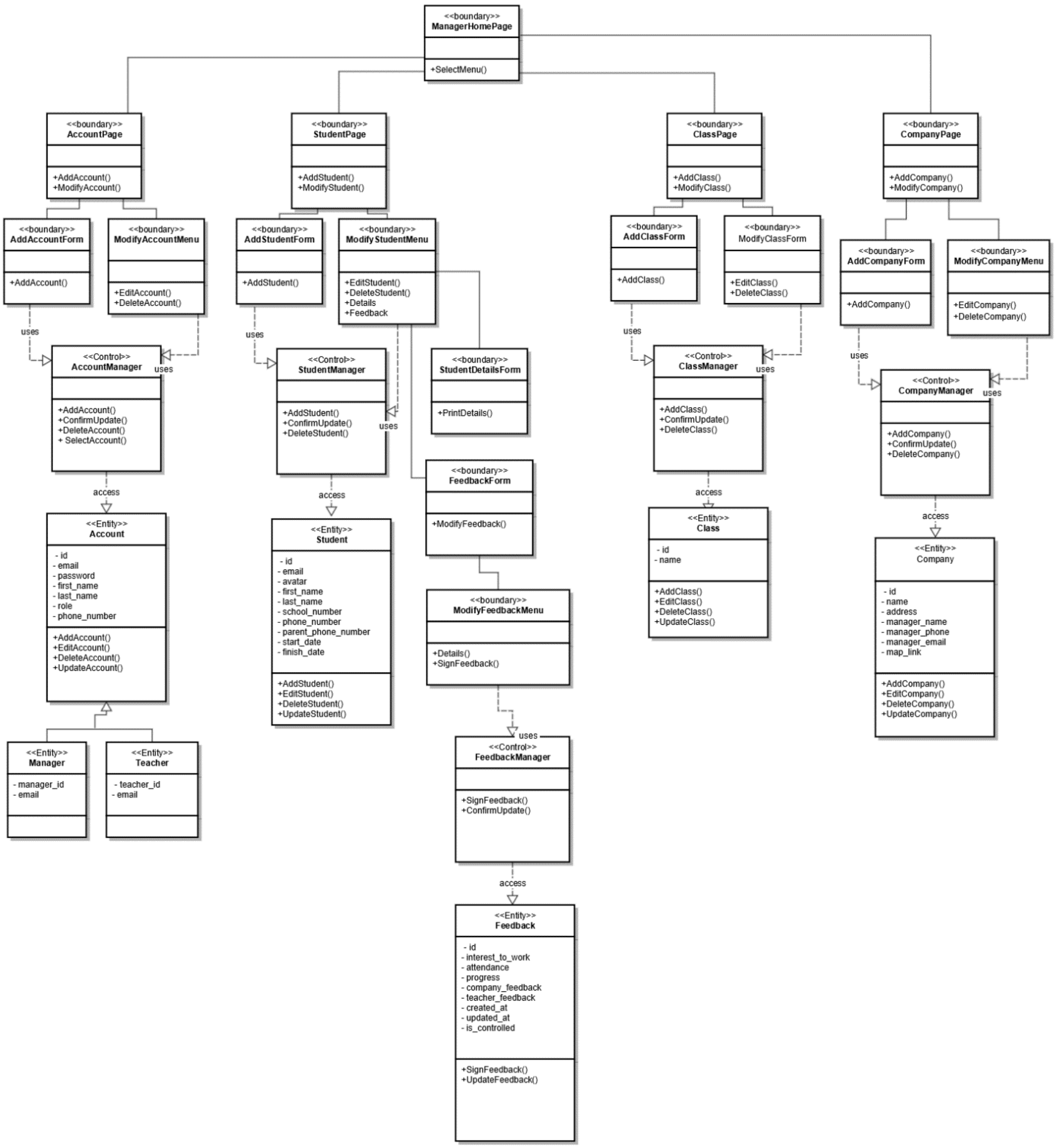

Figure 13: ManagerHomePage class diagram of the system

2.1.4.2 TeacherHomePage Class Diagram

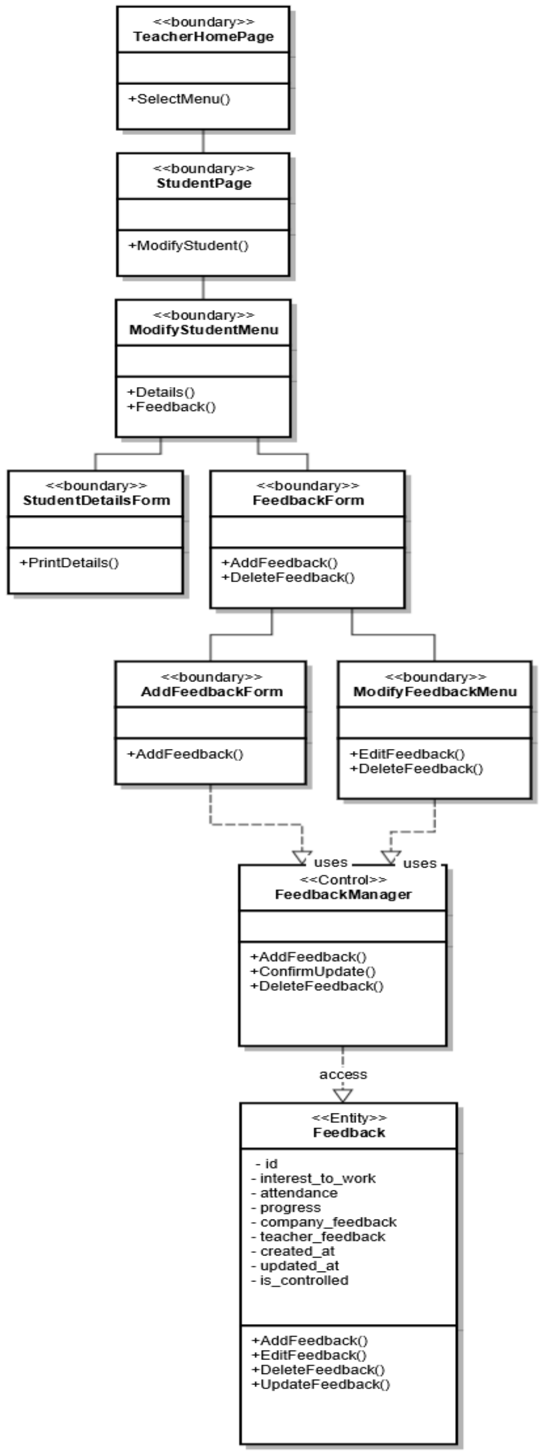

Figure 14: TeacherHomePage class diagram of the system.

2.1.4.3 Student Package Class Diagram

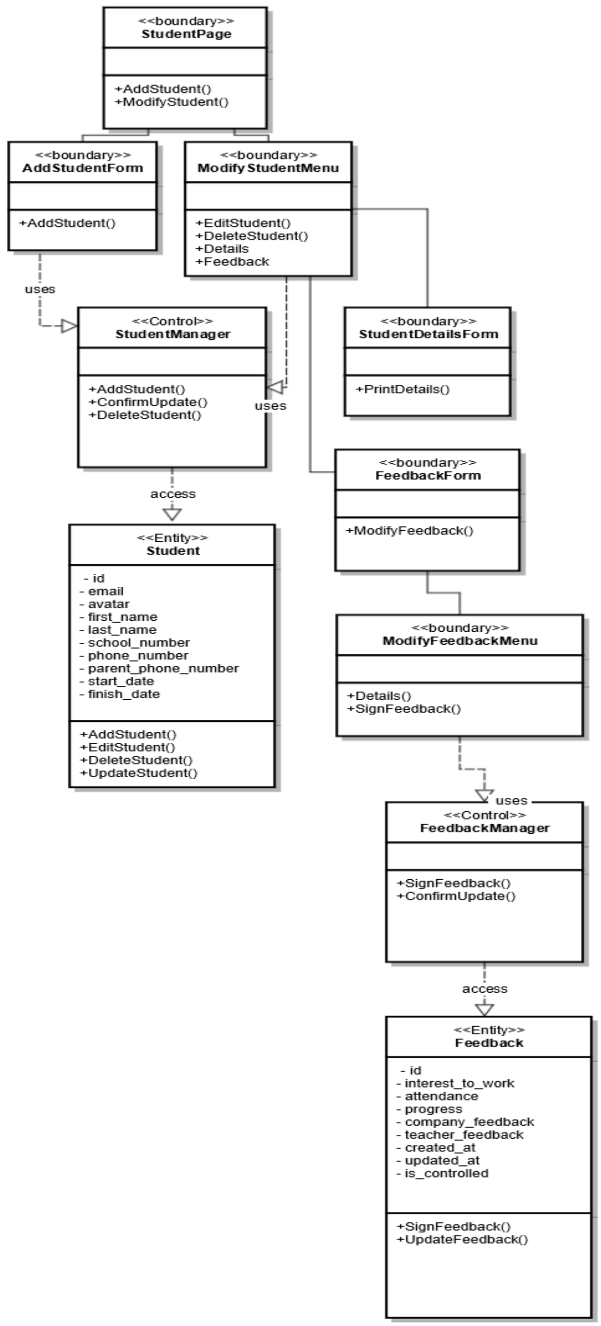

Figure 15: Student package class diagram of the system.

2.1.4.4 Account Package Class Diagram

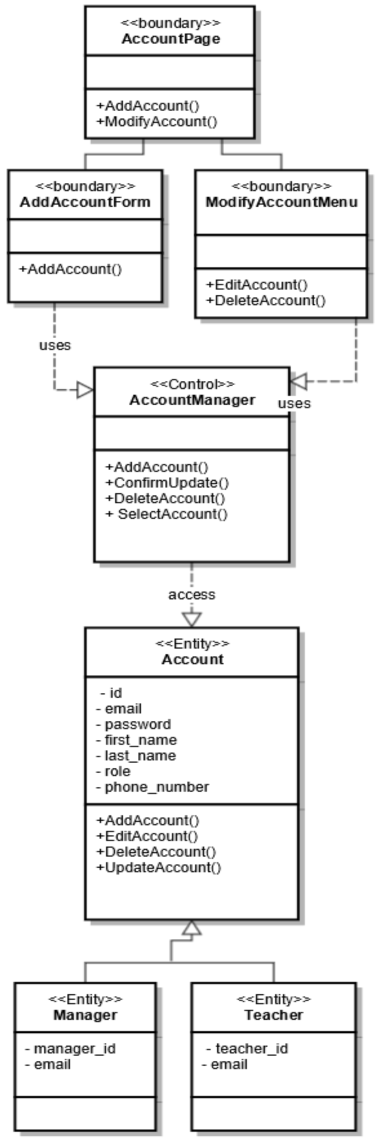

Figure 16: Account package class diagram of the system.

2.1.4.5 Class Package Class Diagram

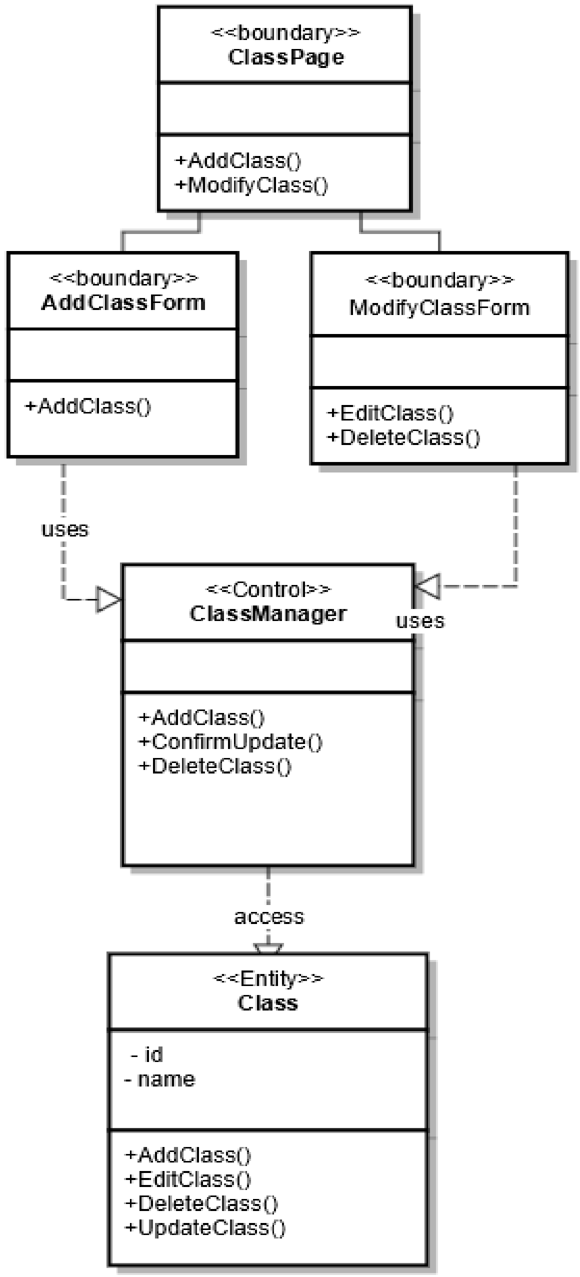

Figure 17: Class Package class diagram of the system

2.1.4.6 Company Package Class Diagram

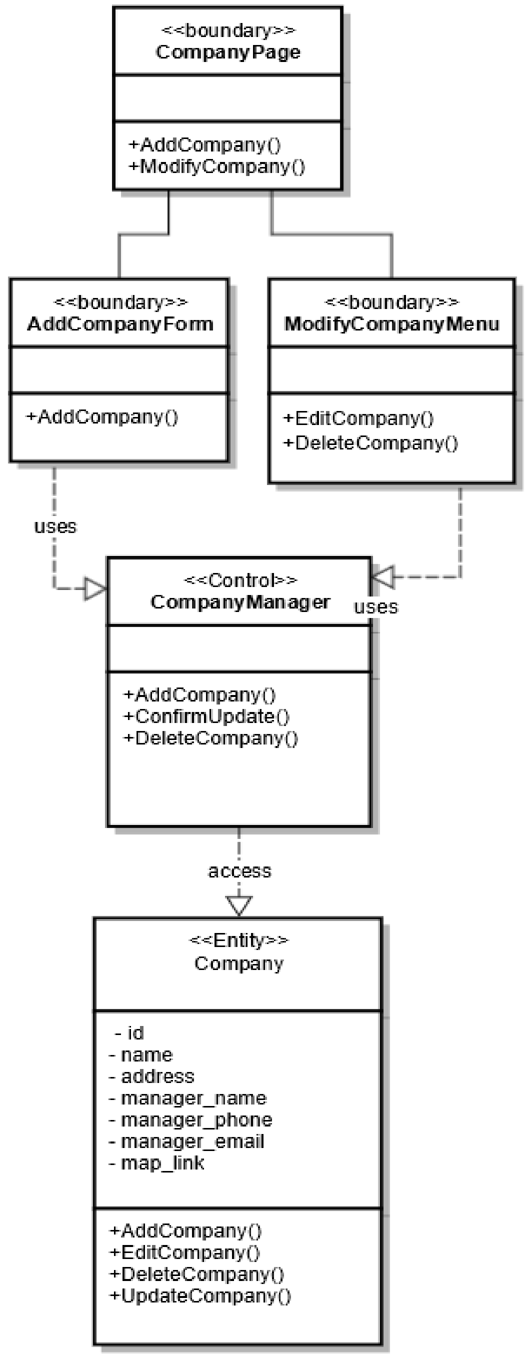

Figure 18: Company Package class diagram of the system.

2.1.4.7 StudentList Package Class Diagram

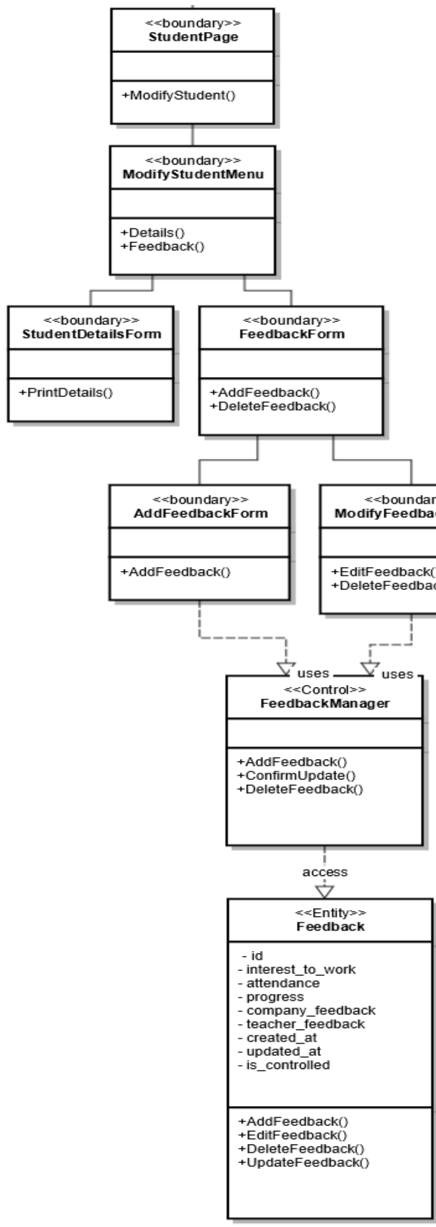

Figure 19: StudentList Package class diagram of the system.

2.2 NON-FUNCTIONAL REQUIREMENTS

2.2.1 Audit-Ability

Description 1: Individual performance report on work undertaken by the members of staff.

Fit Criteria 1: Preliminary reports taken to ensure that the members of staff are working to what is required every week.

Rationale 1: This is useful as it ensures that every member of staff is on track to the specific task given to them.

2.2.2 Availability

Description 1: The system operates 24 hours a day. Every working sector should be operating.

Fit Criteria 1: The availability will be monitored every hour and the uptime of every working sector should be operational.

Rationale 1: The system should be operating 24 hours a day excluding maintenance down time as this will achieve maximum potential output possible.

Description 2: Scheduled down time for maintenance once a year. The system is not available during this time.

Fit Criteria 2: This will be measured once a year and the system should not be operational at this time.

Rationale 2: This will be to ensure that the system is running efficiently up to the standards required.

2.2.3 Backup and Restore

Description 1: When the system fails, initiates restoration of the database on demand to the most recent backup that is operational from the archive.

Fit Criteria 1: Every 3 months the backup will be checked for and data loss.

Rationale 1: Through system failure this is to ensure that the system is restored must be up to date to the latest restoration point.

2.2.4 Capacity

Description 1: The server is able to accommodate up to 100 users.

Fit criteria 1: Every day the system's performance should be checked to ensure that it could satisfy the amount of users logged in at one time without causing the system to slow down.

Rationale 1: This is useful in the system if every member of staff is logged into the system all at once, the system should still perform to a very high and efficient level.

Description 2: The system should be able to handle over 10 MB amount of data stored a day.

Fit criteria 2: Every day the data stored in the system is examined to check that it can handle over 10 MB of data stored.

Rationale 2: This is useful as it keeps tracks of how much data flows within the system.

2.2.5 Security

Description 1: The system should allow for a restricted level of access:

- **Manager**: Access to all sub-systems except Student-List package.

- **Teacher**: Access to Student-List package only.

Fit Criteria 1: System will be tested every day to ensure that each member of staff has the correct login details.

Rationale 1: This is essential as each member of staff have their own access right.

- No restriction to password length or characters.

- Can contain numbers and symbols.

Fit Criteria 2: For each account, created passwords can be of any length and any character.

Rationale 2: This gives users the flexibility as he or she have the ability to choose their own personal passwords.

2.2.6 Usability

Description 1: Look and feel standards:

- Consistent layout and flow throughout the system

- Matching and steady colors in the system

Fit Criteria 1: The system will be monitored once updated to ensure the layout and structure is consistent.

Rationale 1: This will make it very easy for the user to work in the environment provided.

Description 2: Localization; Turkish keyboard and Language.

Fit Criteria 2: System will be used every day therefore this will be tested daily to ensure that the language and the keyboard layouts are of the Turkish standard.

Rationale 2: This is useful because all members of staff should be able to use and work with the Turkish language and keyboard.

2.2.7 Reliability

Description 1: Mean time between Failures expected no more than twice a month.

Fit Criteria 1: The system will be tested every day to insure there are no errors and Users can use the system without any failures.

Rationale 1: This is essential, as it will allow users to use the system smoothly without any system breakdowns.

Description 2: Mean time to recovery when system fails should not exceed 2-3 hours.

Fit Criteria 2: System will be tested every minute to avoid any failures in the process of recovery.

Rationale 2: This is essential because there should be a smooth recovery operation on the system to ensure that it will be up and running again as soon as possible.

2.2.8 Performance

Description 1: Response times:

- Application loading not more than 2 seconds at most

- Screen open and refresh time not more than 2 seconds at most.

Fit Criteria 1: System performance will be examined every day to make sure there are no delays at any given time for any process made.

Rationale 1: This is important because in order to achieving the highest level of output, the application loading time and screen refresh rate should not take longer than it should.

2.2.9 Maintainability

Description 1: On system downtime, repairs should not exceed 3 hours' maintenance when restoring the system.

Fit Criteria 1: System will be tested every minute to avoid any failures in the process of restoration.

Rationale 1: This is essential, as there should be a smooth recovery operation on the system to ensure that it will be up and again as soon as possible.

2.2.10 Privacy

Description 1: Customer Loyalty:

- Customer details are to be kept private
- Details of customer are to be utilized only when necessary

Fit Criteria 1: This principle will be maintained and revised every day to insure customers' details are kept safe and not leaked without authorization.

Rationale 1: This requirement is essential to gain customer loyalty and trust at a professional standard. Their details have to be kept private and only used when necessary.

3. DESIGN

3.1 DYNAMIC MODELS

3.1.1 Activity Diagram

Activity Diagrams used to model sequence of actions as part of the process flow. It focuses on the work performed in the implementation of an operation and the activities in a use case instance or in an object. (Bhattacharjee & Shyamasundar, 2009)

Concurrent activities like using forks, joins and rendezvous can be shown in the activity diagrams. (Lethbridge & Laganière, 2005)

- A fork has one incoming transition and multiple outgoing transitions. (Lethbridge & Laganière, 2005)

- A join has multiple incoming transitions and one outgoing transition. (Lethbridge & Laganière, 2005)

- A rendezvous has multiple incoming and multiple outgoing transitions. An activity diagram also has two types of nodes for branching within a single thread. (Lethbridge & Laganière, 2005)

- A decision node has one incoming transition and multiple outgoing transitions, each with a Boolean guard in square brackets. (Lethbridge & Laganière, 2005)

- A merge node has two incoming transitions and one outgoing transition. It is used to bring together paths had been split by decision nodes. (Lethbridge & Laganière, 2005)

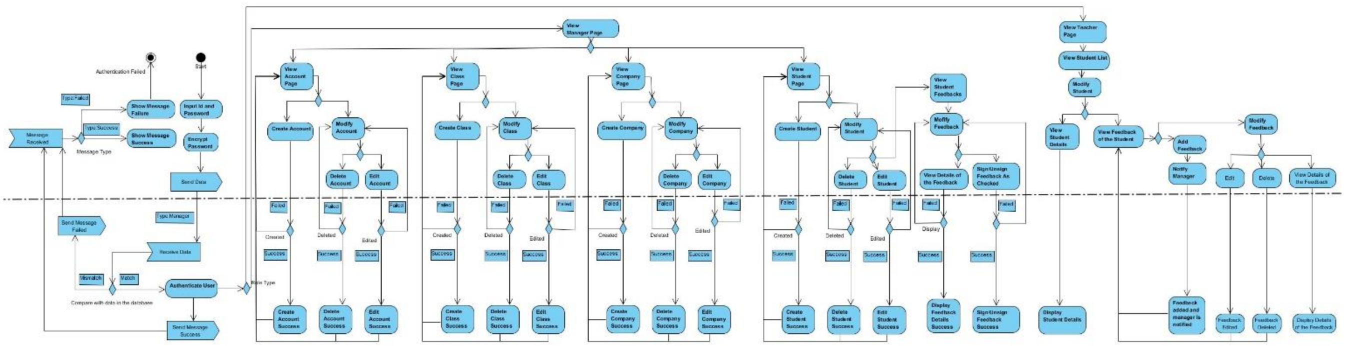

Figure 20: Full activity diagram of the system

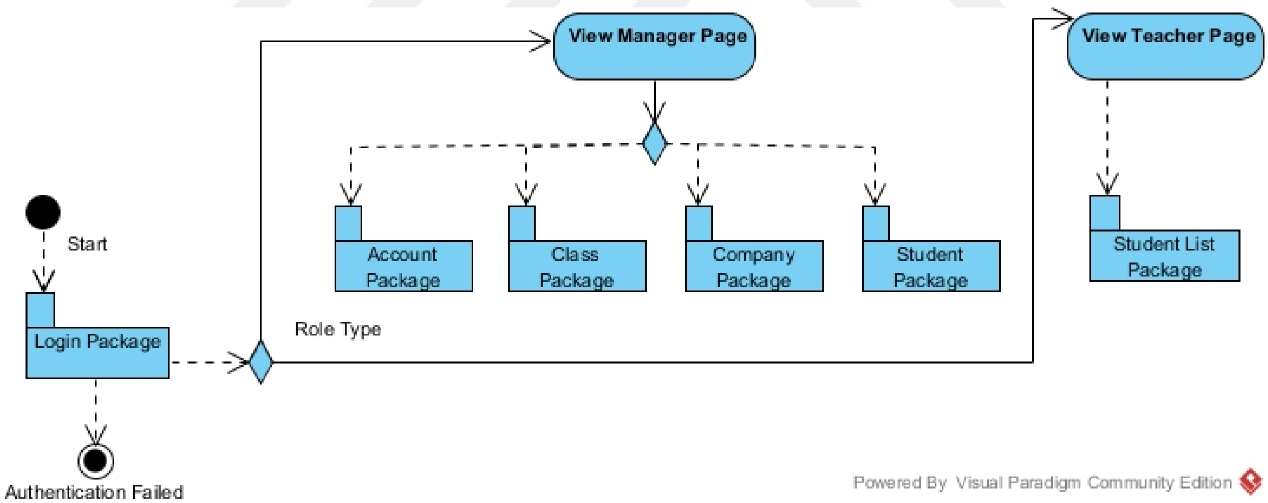

Figure 21: Full package activity diagram of the system

3.1.1.1 Login Package

Figure 22 shows the process of logging into a website, from entering your username and password, to successfully logging in to the system.

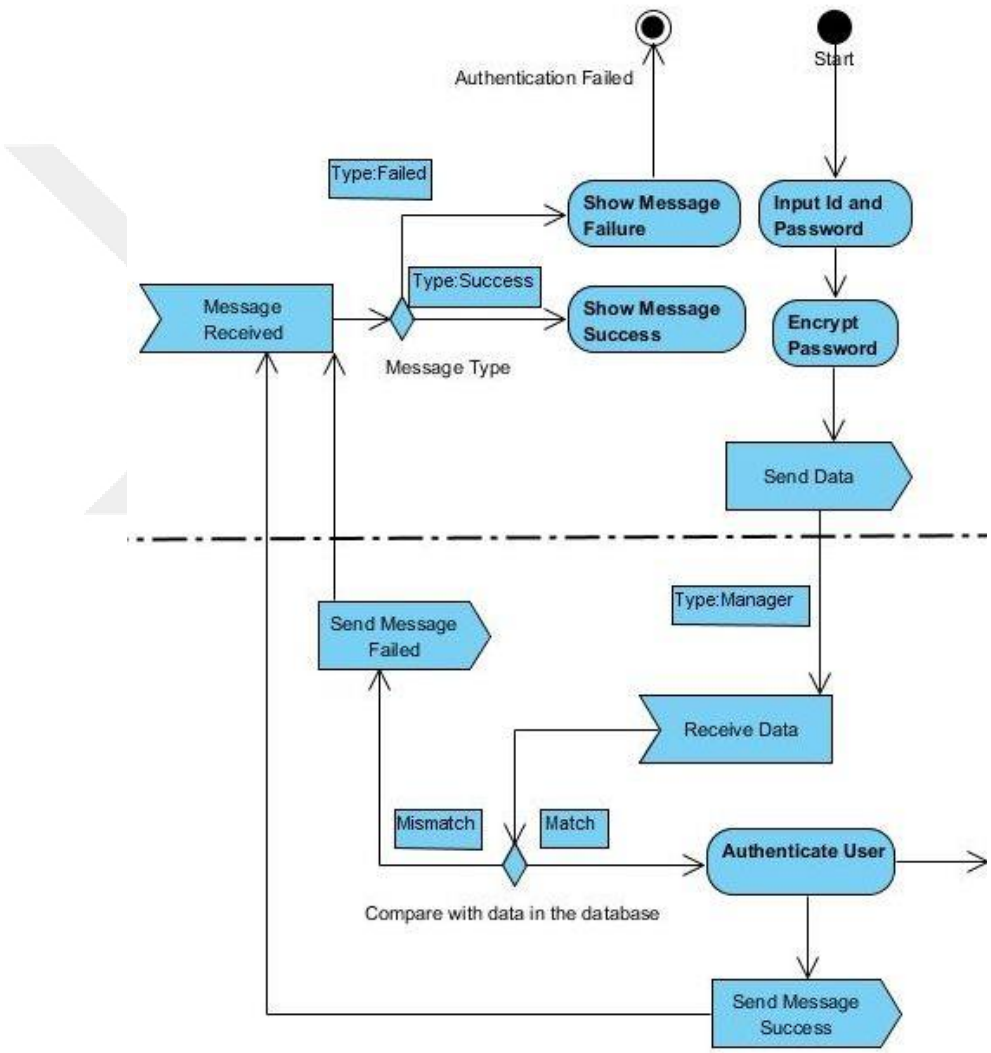

Figure 22: Login activity diagram of the system

3.1.1.2 Account Package

Figure 23 shows the process of adding an account, deleting it or editing it.

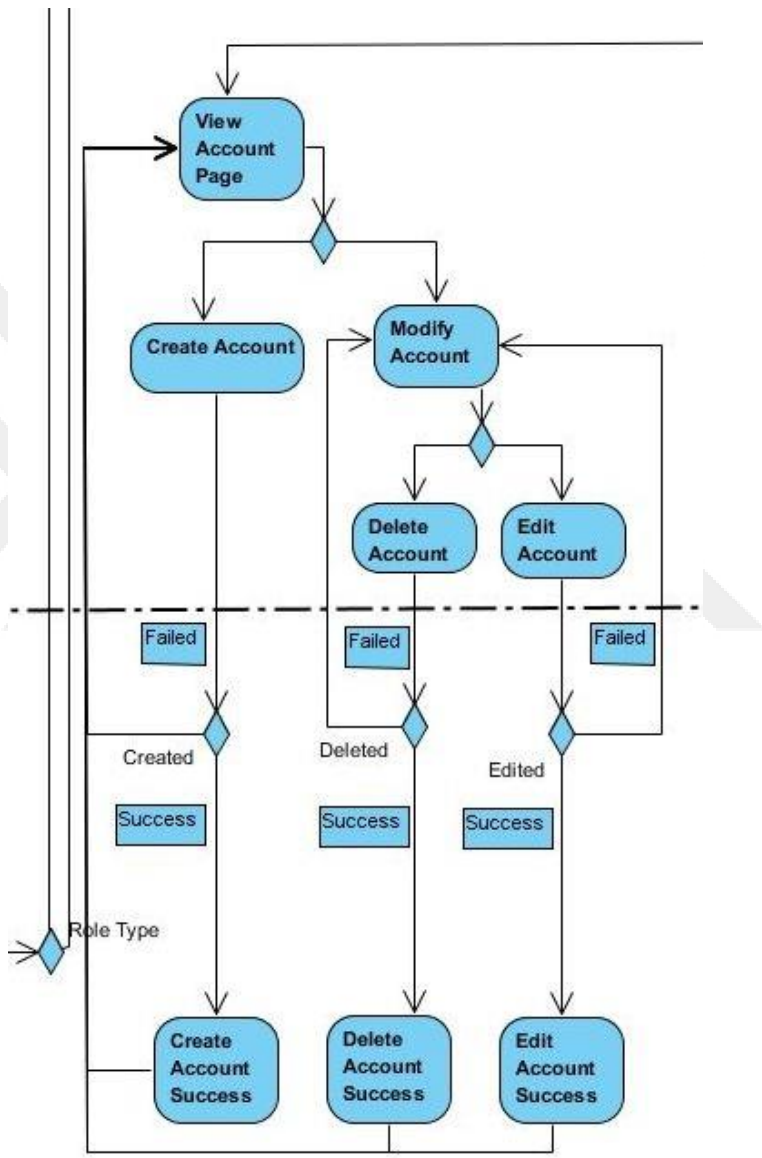

Figure 23: Account activity diagram of the system

3.1.1.3 Class Package

Figure 24 shows the process of adding a class, deleting it or editing it.

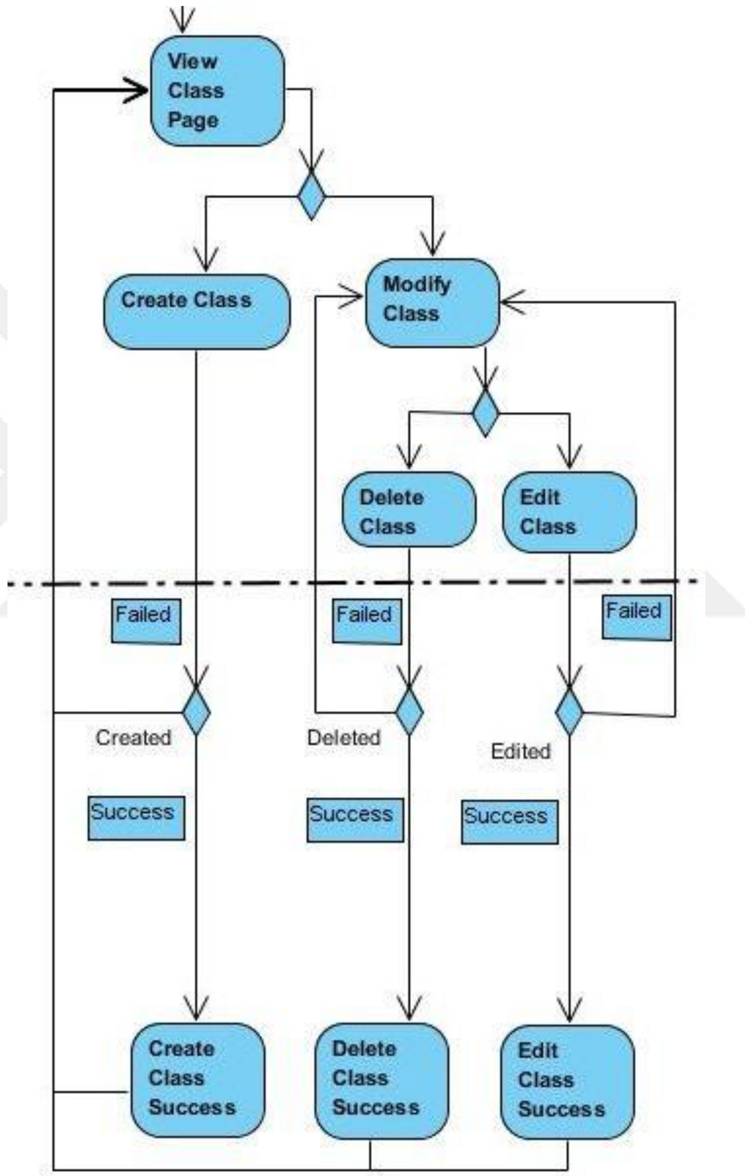

Figure 24: Class activity diagram of the system

3.1.1.4 Company Package

Figure 25 shows the process of adding a company, deleting it or editing it.

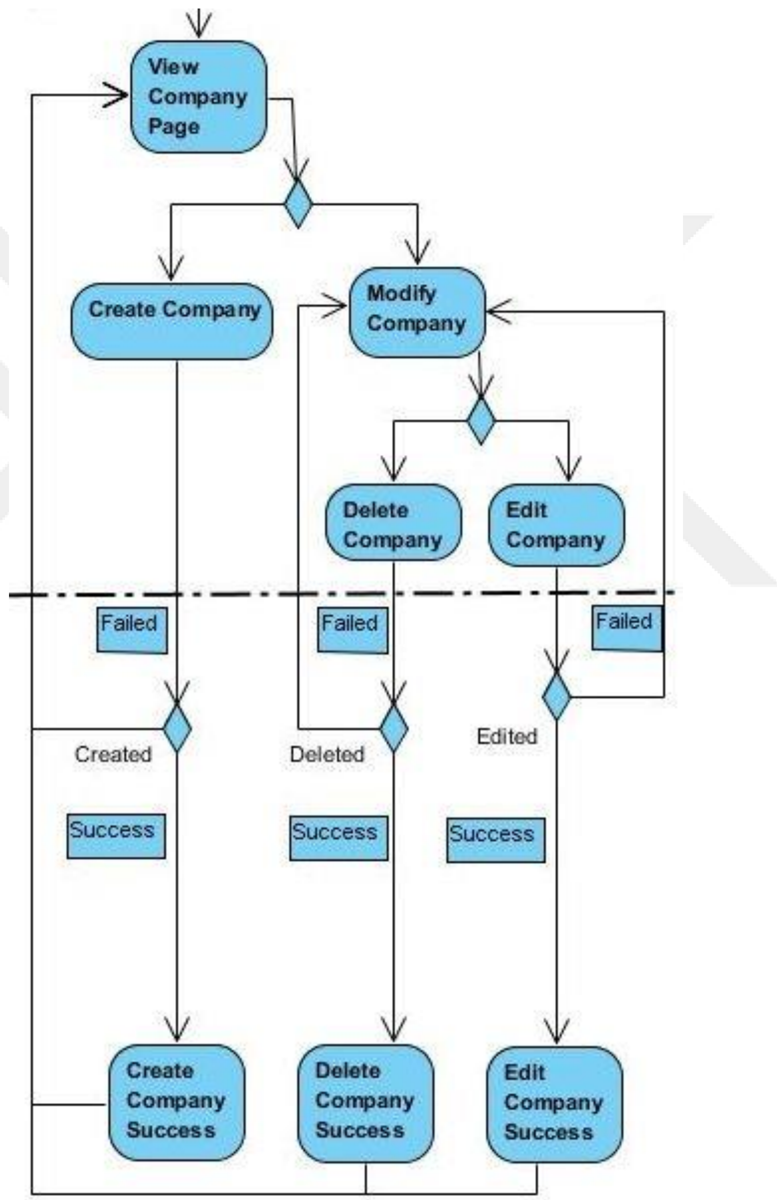

Figure 25: Company activity diagram of the system

3.1.1.5 Student Package

Figure 26 shows the process of adding a student, deleting it or editing it also see feedbacks of the students and sign them as controlled.

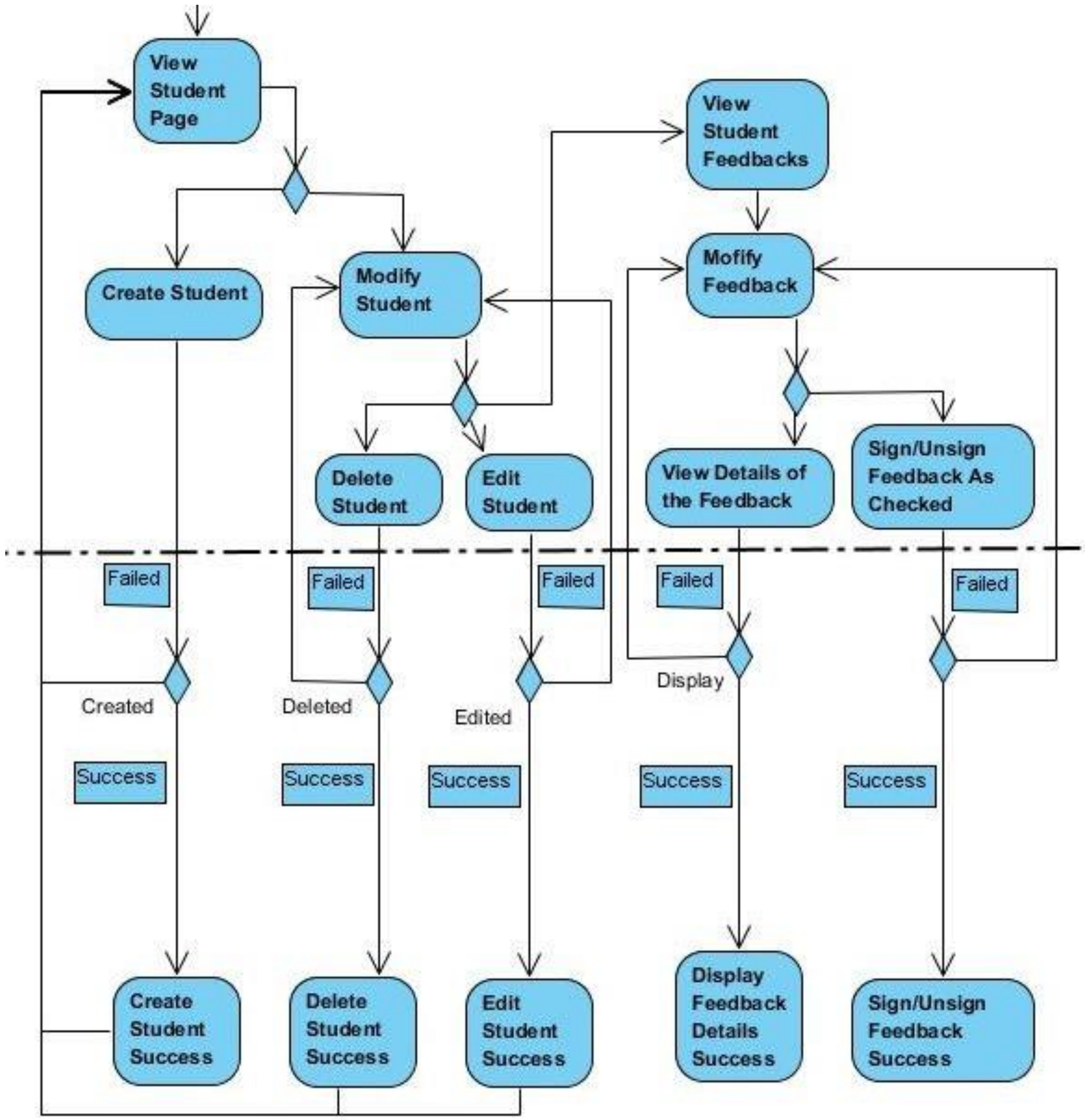

Figure 26: Student activity diagram of the system

3.1.1.6 Student List Package

Figure 27 shows the process of adding a feedback to a student, deleting it or editing it.

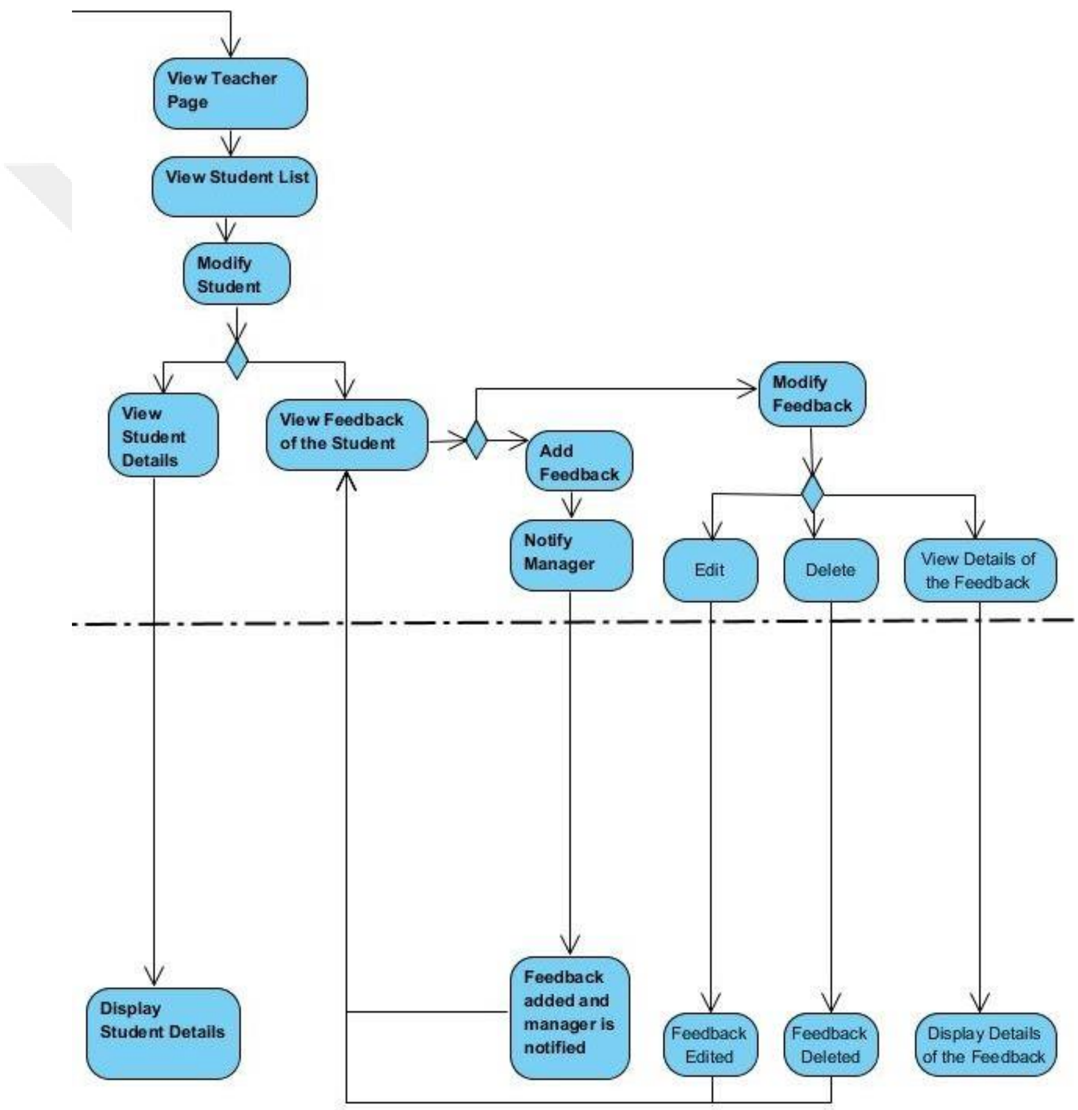

Figure 27: Student list activity diagram of the system

3.1.2 Sequence Diagram

A sequence diagram is used to display exchanging of sequential messages between interaction objects in the system. Information about the flow of control during the interaction, such as if-then-else conditions and iteration might be displayed in the diagram. (Garousi, Briand, & Labiche, 2005)

The sequence diagrams for “Add Student”, “Login”, “Sign Feedback as Controlled”, “Display Feedbacks of a Student”, “Add Feedback” are shown respectively.

3.1.2.1 Add Student

Figure 28 shows the process of adding a student when manager user is on the Student Page.

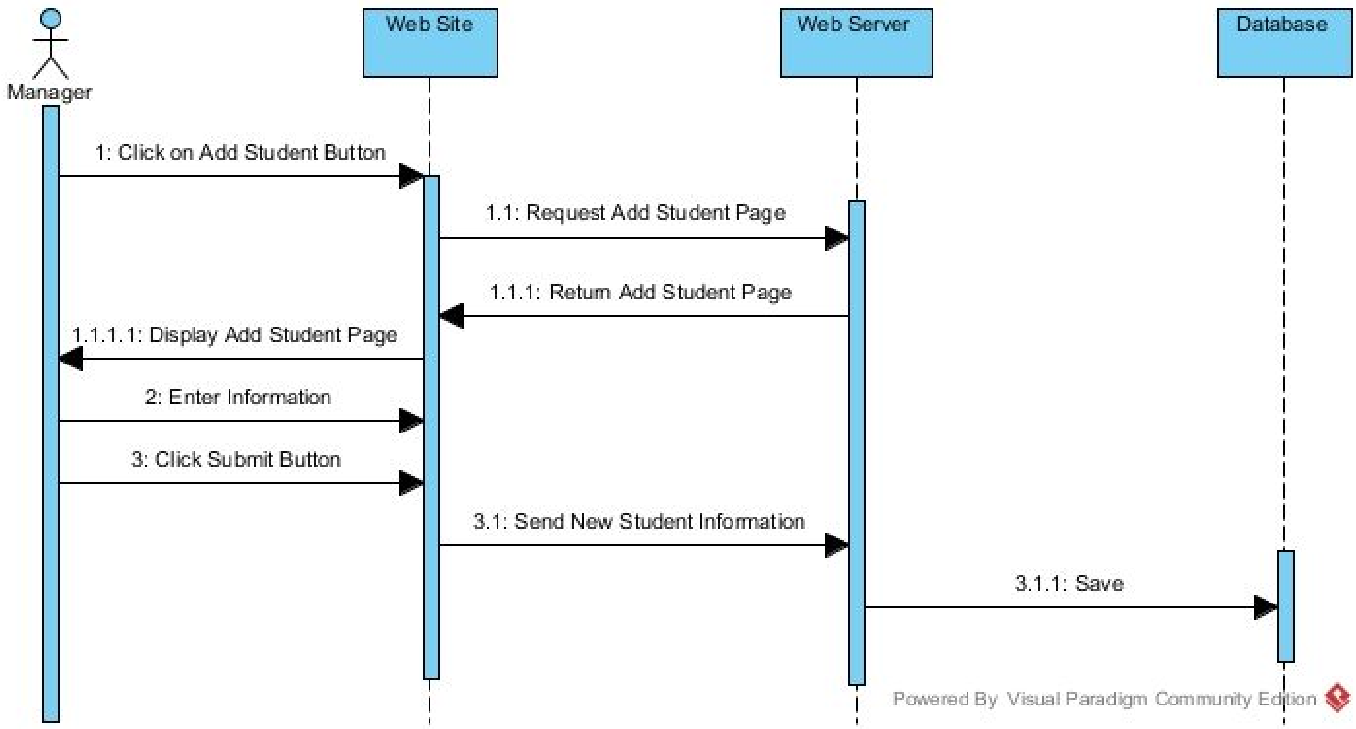

Figure 28: Add student sequence diagram of the system

3.1.2.2 Login

Figure 29 shows the process of logging into a website, from entering your username and password, to successfully logging in to the system.

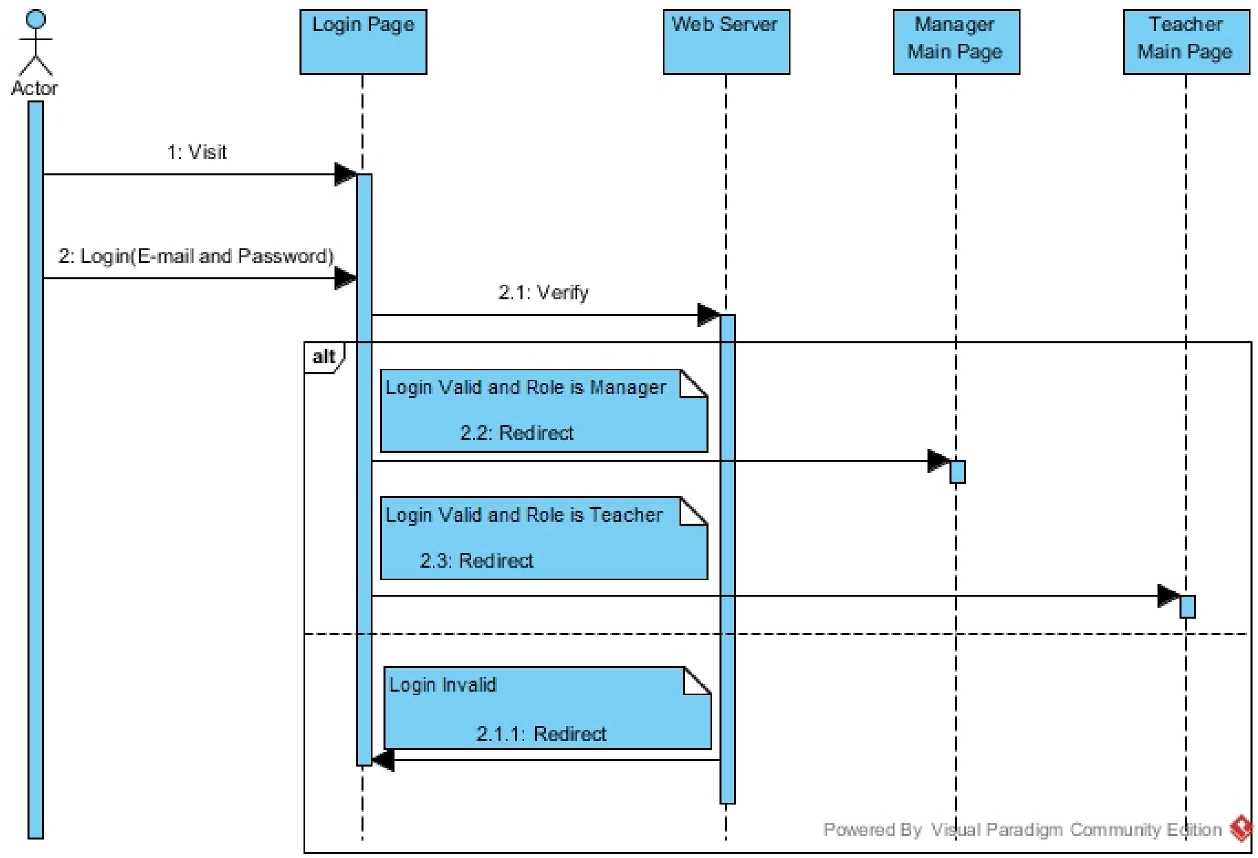

Figure 29: Login sequence diagram of the system.

3.1.2.3 Sign Feedback as Controlled

Figure 30 shows the process of a manager signing a feedback as controlled after login to the system.

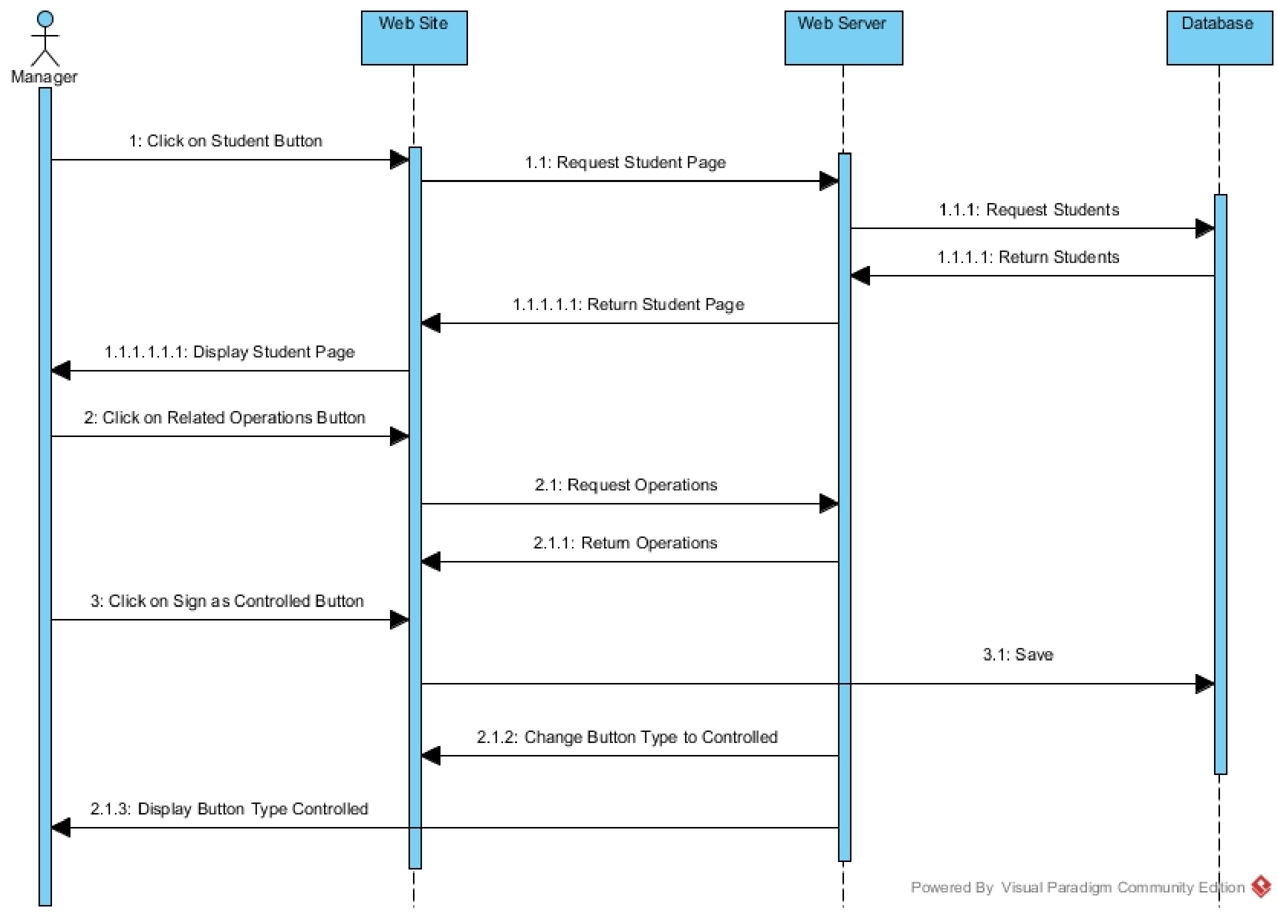

Figure 30: Sign feedback as controlled sequence diagram of the system

3.1.2.4 Display Feedbacks of a Student

Figure 31 shows the process of a manager views the feedbacks of a chosen student. This sequence diagram is the same for the teacher user too.

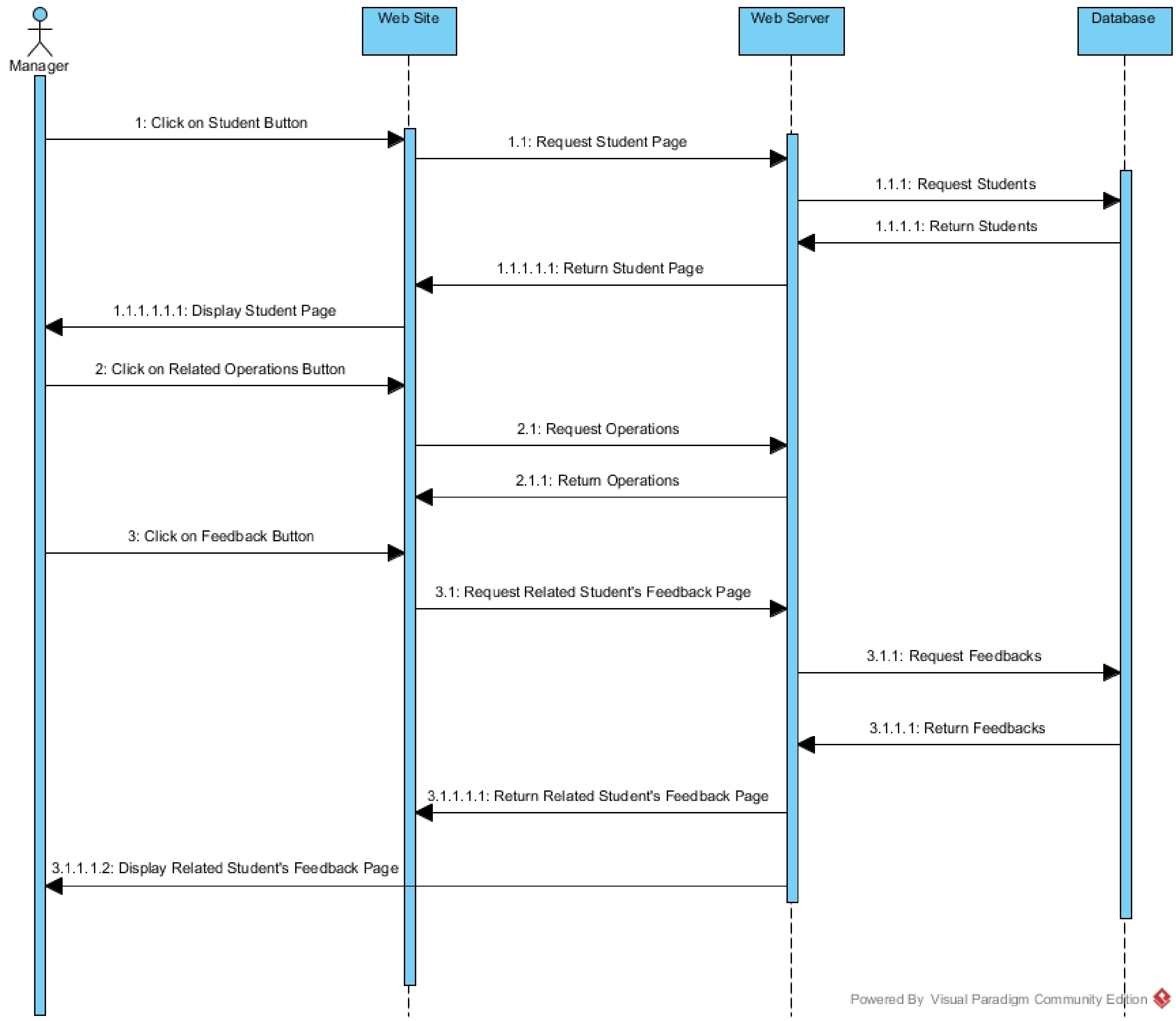

Figure 31: Display feedback of a student sequence diagram of the system

3.1.2.5 Add Feedback

Figure 32 shows the process of adding a feedback when teacher user is on the feedbacks of a student page.

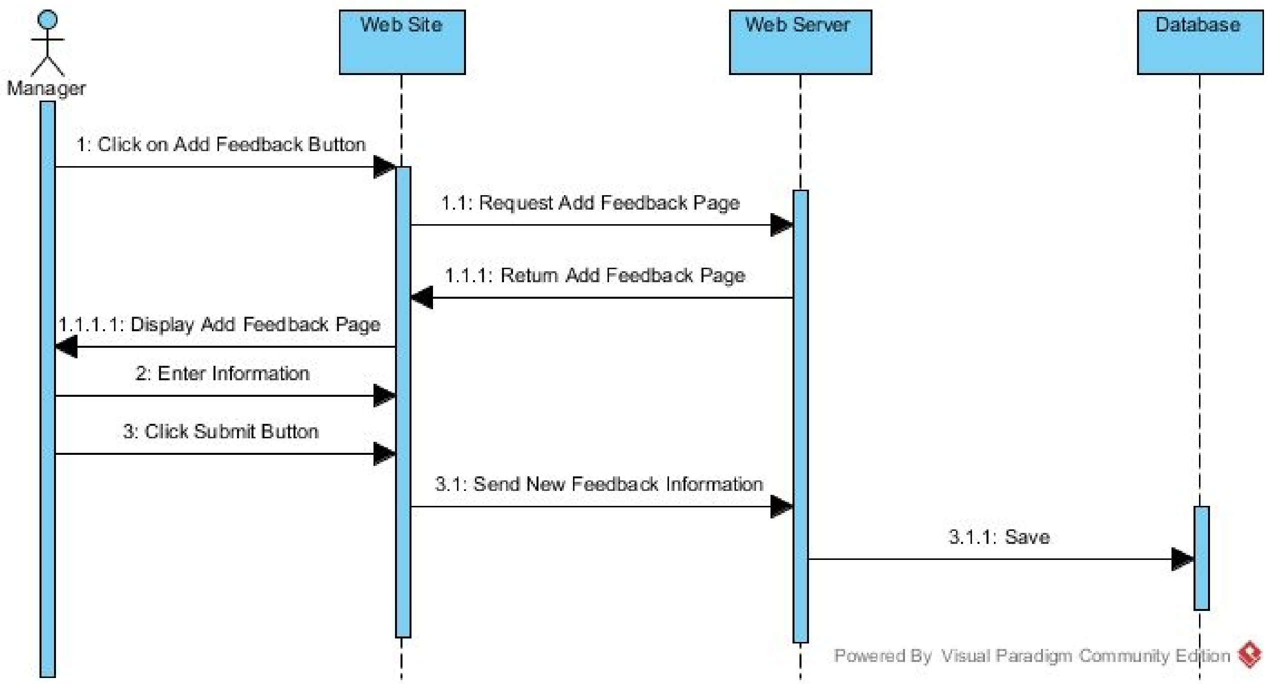

Figure 32: Add feedback sequence diagram of the system

3.1.3 Component Diagram

A component diagram shows relationship between the components of a system. A box with a little plug symbol in the top-right corner is the symbol for the component. (Lethbridge & Laganière, 2005)

A component provides one or more interfaces for other components to use. A semi-circle at the end of a line is used to show a component is using another component's interface. (Lethbridge & Laganière, 2005)

Various relationships among components:

- Component mat execute each other's or methods in another components. (Lethbridge & Laganière, 2005)

- It is possible for a component to generate another component. (Lethbridge & Laganière, 2005)

- Communication is possible between two components. (Lethbridge & Laganière, 2005)

As depicted in Figure 33, Intern Tracking System consists of management components and users, which interacts with each other to safely maintain an Intern Tracking System.

Teacher user only interacts with Feedback Management component directly. Manager user interacts with all the components. However, when interacting with Feedback Management component, manager only requires the provided Sign/Unsign Feedback Interface. This interface also required by Feedback Management component to stop teacher user from deleting or editing the signed feedback.

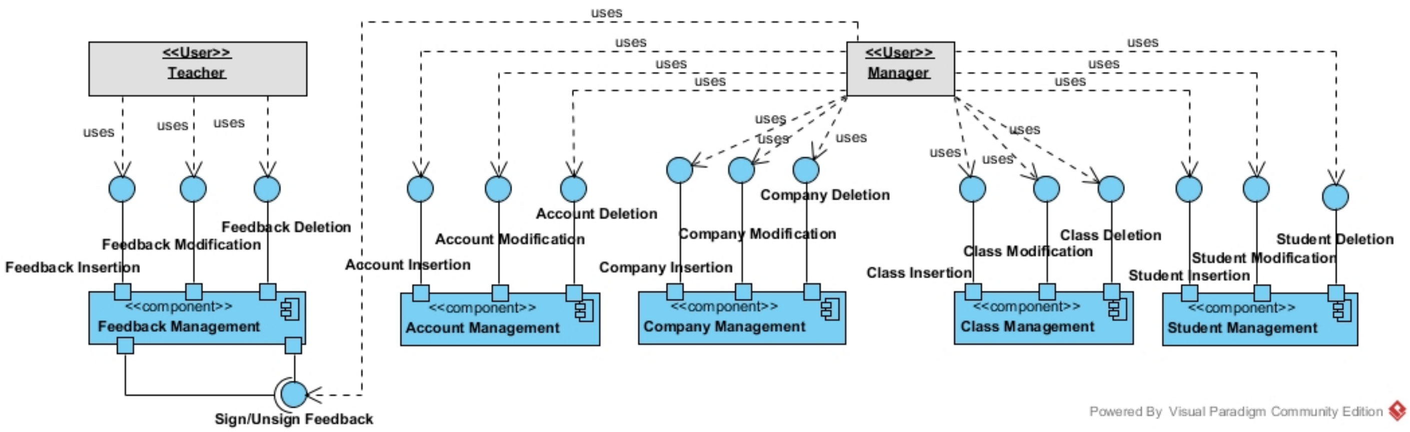

Figure 33: Component diagram of the system

3.2 DATABASE DESIGN

3.2.1 Entity Relationship Model

An ER data model defines data as entities, attributes, and relationships. ER diagrams show how data will be presented and organized in the final database. However, the model diagrams do not indicate the actual data, or how it is stored. The users and applications will create the data content and the DBMS will create the database to store the content. (Ricardi, 2002)

The figure 34 and 35 below shows the entity relationship model of the student tracking system. There are companies, students, users, feedbacks and class_rooms entities. The relationship between these entities can be seen in the diagram.

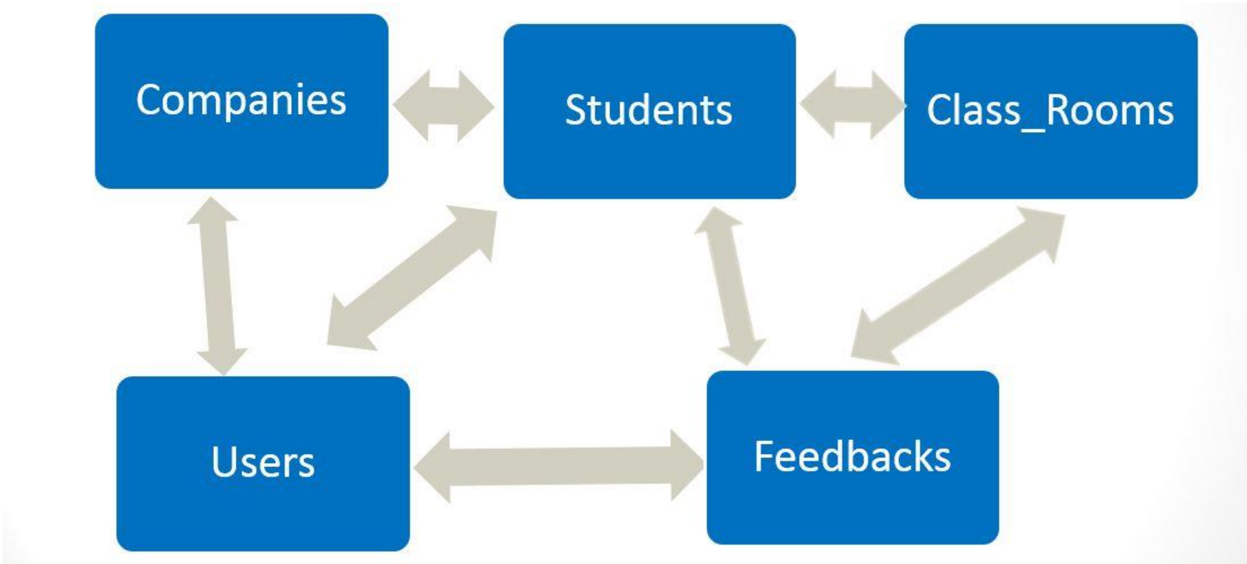

Figure 34: Simplified ER diagram of the system.

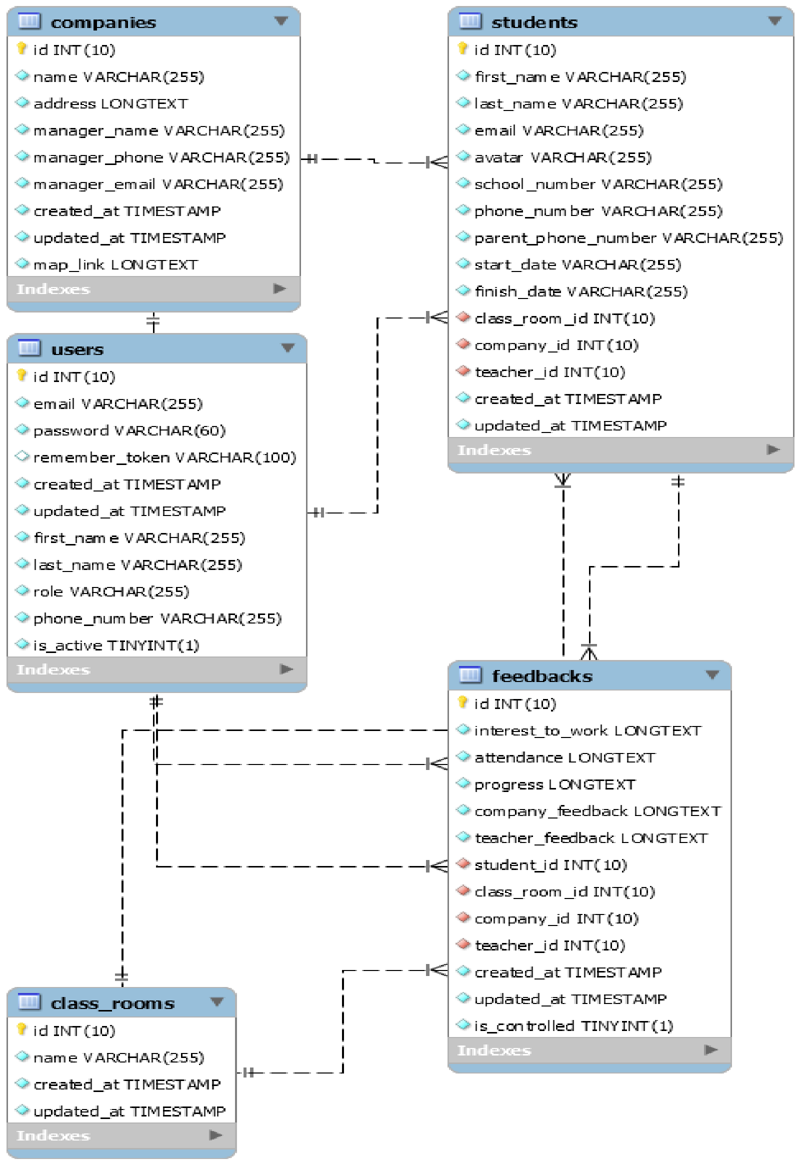

Figure 35: ER diagram of the system.

3.2.2 Database Query

3.2.2.1 What is a Query

Query is a way to extract data from a database. (Silberschatz, Korth, & Sudarshan, 2001) The extraction does not affect the schema of the database, but affects the content of the database.

DML statements are handled by two separate subsystems, as follows. (Garcia-Molina, Ullman, & Widom, 2002)

3.2.2.1.1 Answering the Query

The query is parsed and optimized by a query compiler. The resulting query plan, or sequence of actions the DBMS will perform to answer the query, is passed to the execution engine. The execution engine issues a sequence of requests for small pieces of data, typically records or tuples of a relation, to a resource manager that knows about data files (holding relations), the format and size of records in those files, and index files, which help find elements of data files quickly. The requests for data are passed to the buffer manager. The buffer manager's task is to bring appropriate portions of the data from secondary storage (disk) where it is kept permanently, to the main-memory buffers. Normally, the page or "disk block" is the unit of transfer between buffers and disk. The buffer manager communicates with a storage manager to get data from disk. The storage manager might involve operating-system commands, but more typically, the DBMS issues commands directly to the disk controller. (Garcia-Molina, Ullman, & Widom, 2002)

3.2.2.1.2 Transaction Processing

Queries and other DML actions are grouped into transactions, which are units that must be executed atomically and in isolation from one another. Any query or modification action can be a transaction by itself. In addition, the execution of transactions must be durable, meaning that the effect of any completed transaction must be preserved even if the system fails in some way right after completion of the transaction. (Garcia-Molina, Ullman, & Widom, 2002)

3.2.2.2 Queries

3.2.2.2.1 Manager Main Page

SELECT COUNT(*) from `feedbacks` where `is_controlled` = '0';
The query above shows the number of the uncontrolled feedbacks

SELECT COUNT(*) from `users` where `role` = 'TEACHER';
The query above shows the number of the users whose role is teacher.

SELECT COUNT(*) from `users` where `role` = 'MANAGER';
The query above shows the number of the users whose role is manager.

SELECT COUNT(*) from `feedbacks`;

The query above shows the number of total feedbacks.

SELECT COUNT(*) from `class_rooms`;

The query above shows the number of total classes.

SELECT COUNT(*) from `students`;

The query above shows the number of total students.

SELECT COUNT(*) from `companies`;

The query above shows the number of total companies.

3.2.2.2.2 User Login Page

SELECT * from `users` where `users`.`email` = '<email>' and `users`.`password` = '<password>';

The query above shows the registered user if the email and password is correct.

3.2.2.2.3 Manager - Student Page

SELECT * from `students` limit 15 offset 0;

The query above shows 15 of all the students.

DELETE from `students` where `students`.`id` = '2';

The query above deletes the student according to the id which is “2” for this query.

SELECT * from `students` where `students`.`id` = '2' limit 1;

The query above is used to select the student to show information according to the id which is “2” for this query.

INSERT INTO `students` (name,...) VALUES("Microsoft",...)

The query above inserts a new student with all his data.

UPDATE `students` SET name="Ali"... where `students`.`id` = '2'
The query above only updates the name field of the student whose id is '2'.

3.2.2.2.4 Manager - Class Page

SELECT * from `class_rooms` limit 15 offset 0;
The query above shows 15 of all the classes.

DELETE from `class_rooms` where `class_rooms`.`id` = '2';
The query above deletes the class according to the id which is “2” for this query.

SELECT * from `class_rooms` where `class_rooms`.`id` = '2' limit 1;
The query above is used to select the class to show information according to the id which is “2” for this query.

INSERT INTO `class_rooms` (name,...) VALUES("12A",...)
The query above inserts a new class with all its data.

UPDATE ` class_rooms ` SET name="12B"... where `class_rooms`.`id` = '2'
The query above only updates the name field of the class whose id is '2'.

3.2.2.2.5 Manager - User Page

SELECT * from `users` limit 15 offset 0;
The query above shows 15 of all the users.

DELETE from `users` where `users`.`id` = '2';
The query above deletes the user according to the id which is “2” for this query.

SELECT * from `users` where `users`.`id` = '2' limit 1;
The query above is used to select the user to show information according to the id which is “2” for this query.

INSERT INTO `users` (email,...) VALUES("corvustheheretic@hotmail.com",...)

The query above inserts a new user with all its data.

UPDATE ` users ` SET email="ozgurkorkmaz@gmail.com"... where `users`.`id` = '2'

The query above only updates the email field of the user whose id is '2'.

3.2.2.2.6 Manager-Company Page

SELECT * from `companies` limit 15 offset 0;

The query above shows 15 of all the companies.

DELETE from `companies` where `companies`.`id` = '2';

The query above deletes the company according to the id which is “2” for this query.

SELECT * from `companies` where `companies`.`id` = '2' limit 1;

The query above is used to select the company to show information according to the id which is “2” for this query.

INSERT INTO ` companies ` (name,...) VALUES("Microsoft",...)

The query above inserts a new company with all its data.

UPDATE ` companies ` SET name="Siemens"... where `users`.`id` = '2'

The query above only updates the name field of the company which id is '2'.

3.2.2.2.7 Teacher – Student List Page

SELECT * from `students` where `teacher_id` = `users`.`id`;

The query above is used to select the students that are connected to the online teacher.

3.3 WEB SITE INTERFACES AND POSSIBLE SCENARIOS

3.3.1 Login Interface

We can see "Öğretmen ve İdareci Girişi" interface in Figure 36 which asks the user to type his e-mail and password then click "Giriş ". If the user entered either his e-mail or password empty or incorrectly then an error message shown for each situation.

Figure 36: Login interface

3.3.2 Manager Main Interface

The manager can see Manager Main Interface interface in Figure 37, which includes “Öğrenci”, “Sınıf”, “Öğretmen”, “Firma” buttons, a notification with red text about feedbacks that warns the manager about the uncontrolled feedbacks. There is also a section about the school and the intern tracking process. At the bottom of the page, we can see all the information about records. For example; from this figure we can understand that the system has a manager, 2 students, a teacher, a company, 3 feedbacks and a class in total and by clicking “Detay”, we can see details about the records existing.

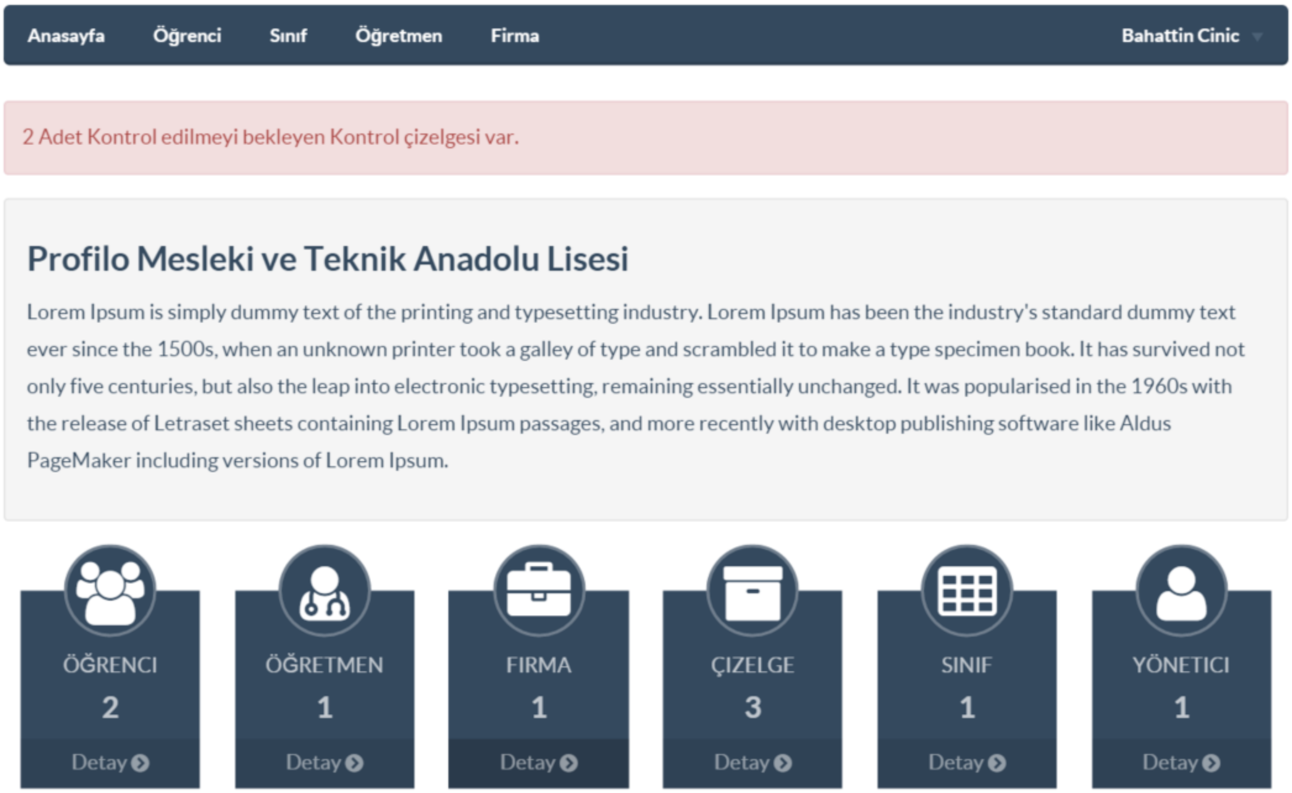

Figure 37: Manager main interface

3.3.3 Manager – Student Operations Interface

The manager can see Student Operations Interface after clicking “Öğrenci”. The manager can click on “Öğrenci Ekle” which links to the “Öğrenci Ekle” interface or from “İşlemler” and “Sil” to delete the student or “Düzenle” which links to “Öğrenci Düzenle” interface and finally the manager can see feedbacks belonging to a student with “Kontrol Çizelgesi” option.

Figure 38: Manager - Student operations Interface

3.3.4 Manager – Add Student Interface

The manager can see Add Student Interface after clicking “Öğrenci” and then “Öğrenci Ekle”. After the manager completes the form and adds the student. If the manager fills the blanks incorrectly or tries to save without filling them then an error message shown for each situation.

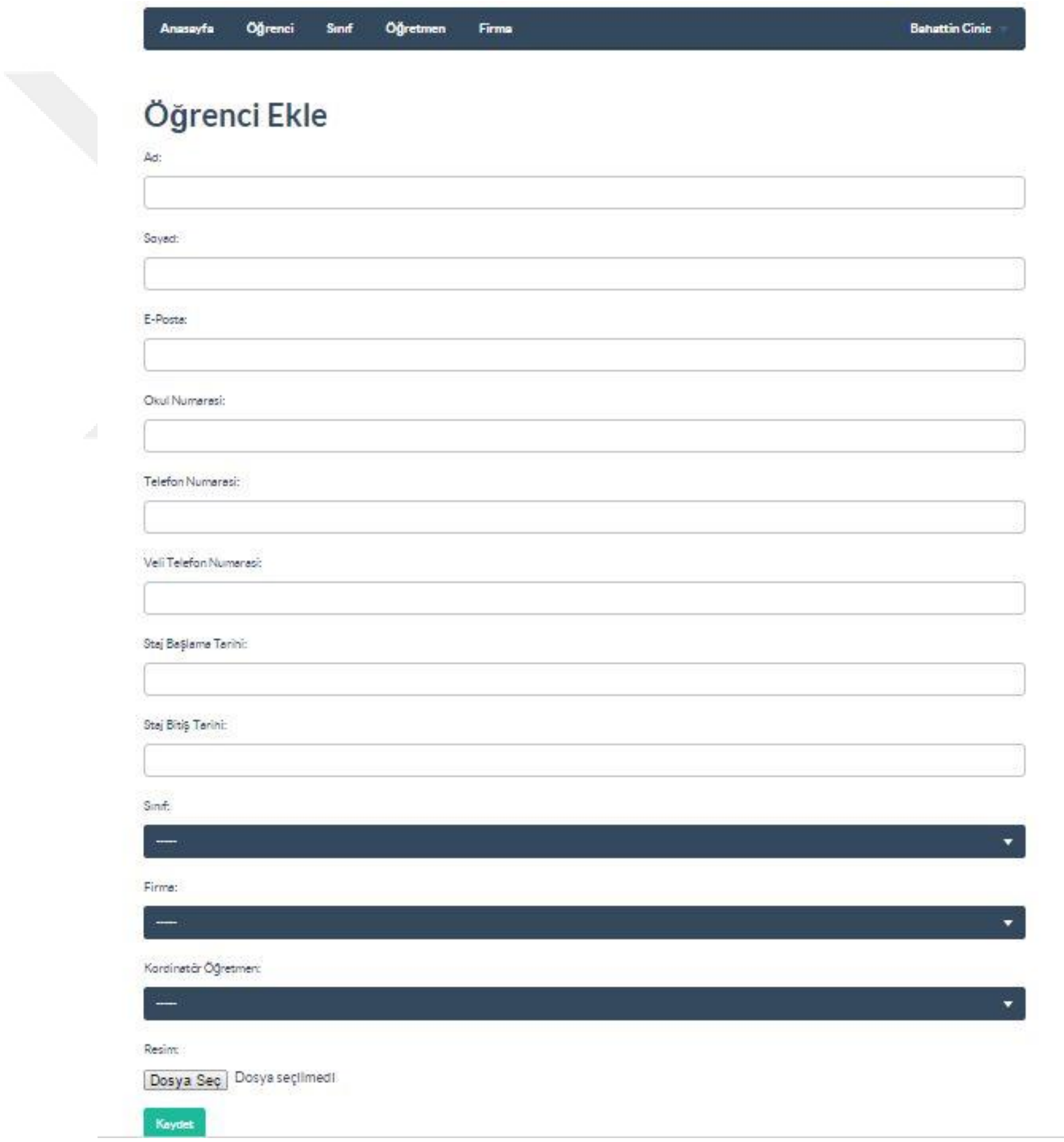

Figure 39: Manager – Add student interface

3.3.5 Manager – Feedbacks Interface

The manager can see Feedback Interface after clicking “Öğrenci” then “İşlemler” and finally “Kontol Çizelgesi”. The manager can see feedbacks related to a student here and the manager can click the “İşlemler” then “Detay” to see details of a feedback and can sign the feedback as controlled or can unsign it too.

Figure 40: Manager – Feedback interface

3.3.6 Manager – Class Operations Interface

The manager can see Class Operations Interface after clicking “Sınıf”. The manager can click on “Sınıf Ekle” which links to the “Sınıf Ekle” interface or from “İşlemler” and “Sil” to delete the class or “Düzenle” which links to “Sınıf Düzenle” interface.

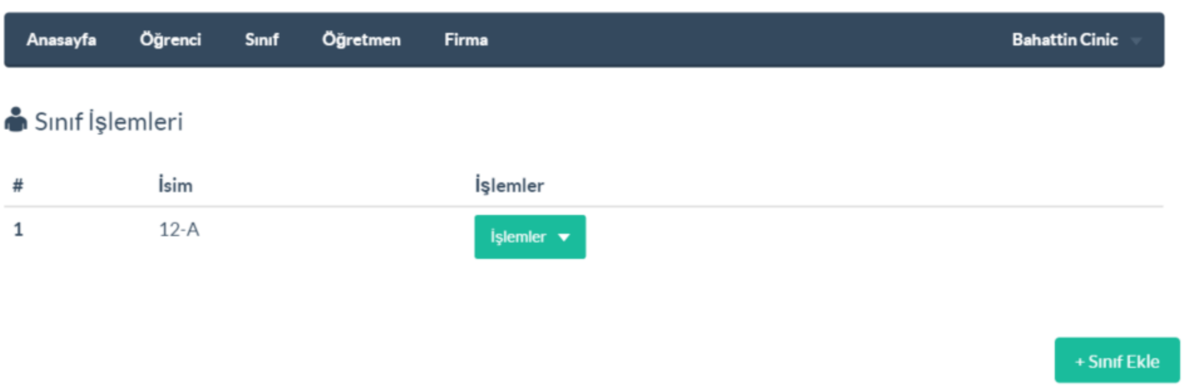

Figure 41: Manager – Class operations interface

3.3.7 Manager – Account Operations Interface

The manager can see Account Operations Interface after clicking “Öğretmen”. The manager can click on “+Öğretmenler & İdareci” which links to the “Öğretmenler & İdareci Ekle” interface or from “İşlemler” and “Sil” to delete the Account or “Düzenle” which links to “Öğretmenler & İdareci Düzenle” interface.

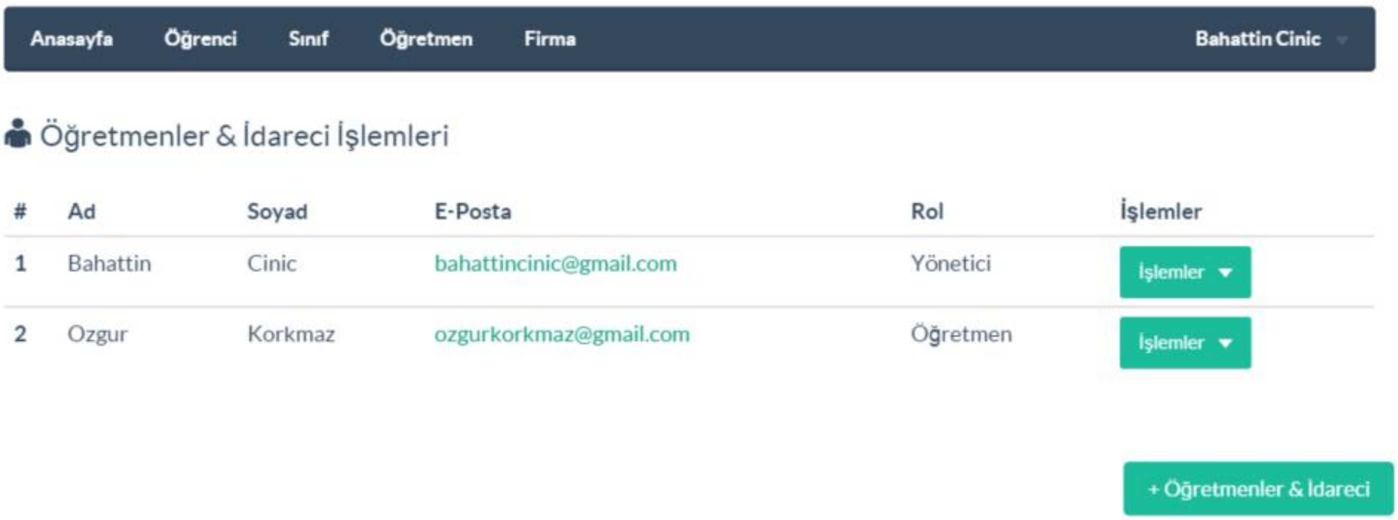

Figure 42: Manager – Account operations interface

3.3.8 Manager – Add Account Interface

The manager can see Add Account Interface after clicking “Öğretmen” and then “+Öğretmenler & İdareci”. After that, the manager completes the form and adds the account. If the manager fills the blanks incorrectly or tries to save without filling them then an error message shown for each situation.

Figure 43: Manager – Add account interface

3.3.9 Manager – Company Operations Interface

The manager can see Company Operations Interface after clicking “Firma”. The manager can click on “+ Firma Ekle” which links to the “Firma Ekle” interface or from “İşlemler” and “Sil” to delete the company or “Düzenle” which links to “Firma Düzenle” interface.

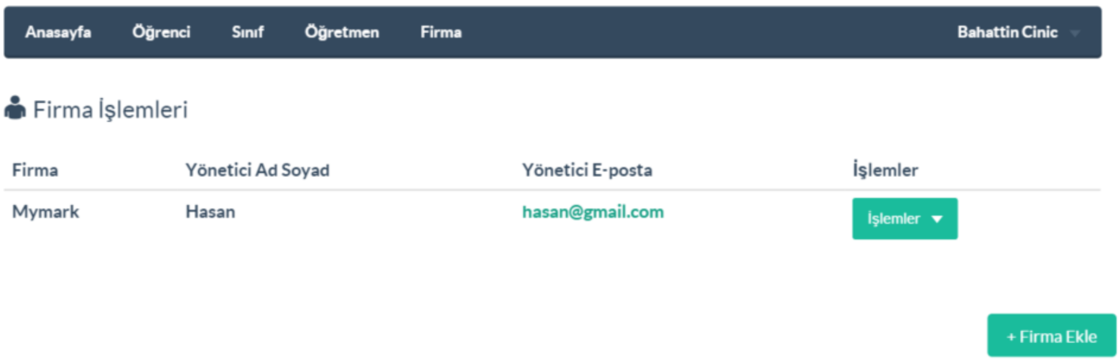

Figure 44: Manager – Company operations interface

3.3.10 Teacher Main Interface

Figure 45 can be seen if a teacher login. The teacher can see his student by clicking “Öğrenci”.

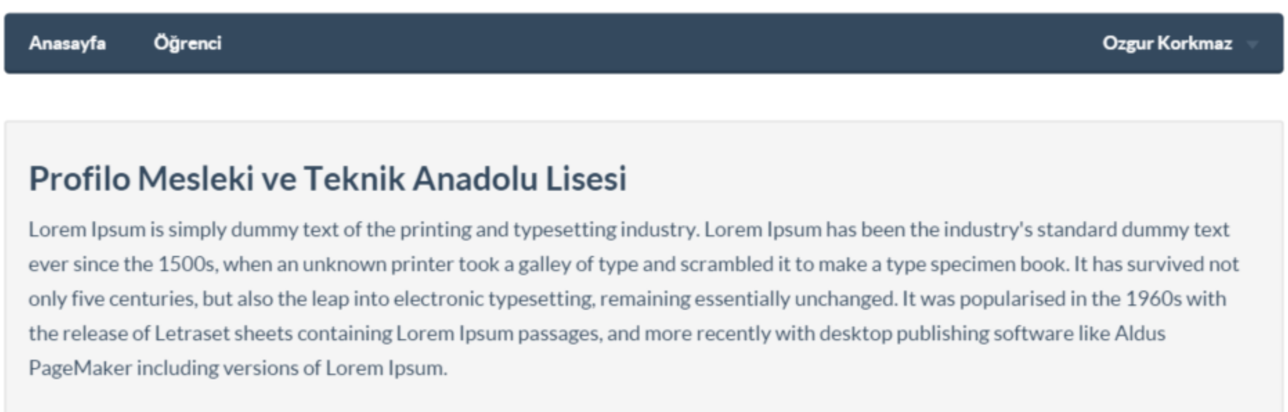

Figure 45: Teacher main interface

3.3.11 Teacher - Student Operations Interface

The teacher can see Student Operations Interface after clicking “Öğrenci”. The manager can click on “İşlemler” and “Kontrol Çizelgesi” which links to the “Kontrol Çizelgesi” interface or “Detay” which links to the “Öğrenci Detay” interface.

Figure 46: Teacher – Student operations interface

3.3.12 Teacher - Feedbacks Interface

The teacher can see Feedback Interface after clicking "Öğrenci" and then "İşlemler" and finally "Kontrol Çizelgesi". All the feedback about the related student can be seen in this interface. The teacher can delete or edit the unsigned feedbacks, which are red colored but not the controlled and signed ones, which are green.

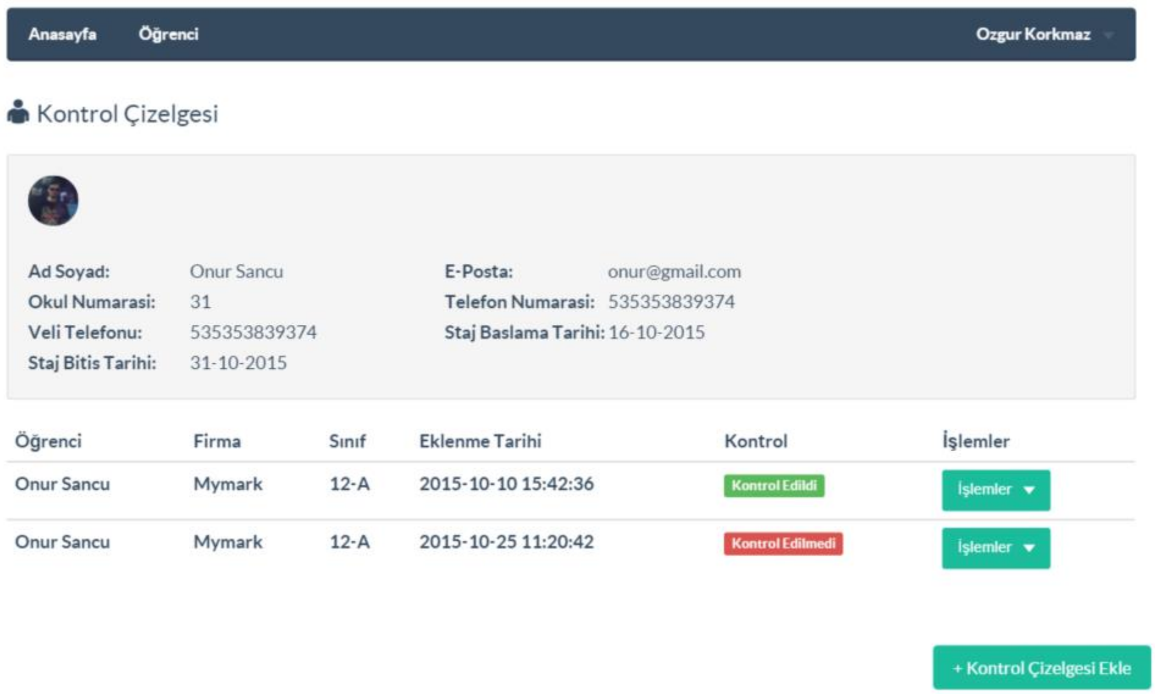

Figure 47: Teacher – Feedbacks interface

3.3.13 Teacher - Add Feedback Interface

The teacher can see Add Feedback Interface after clicking “Öğrenci” then”İşlemler” and finally “Kontrol Çizelgesi”. After that, the teacher completes the form and adds the feedback. If the teacher fills the blanks incorrectly or tries to save without filling them then an error message shown for each situation.

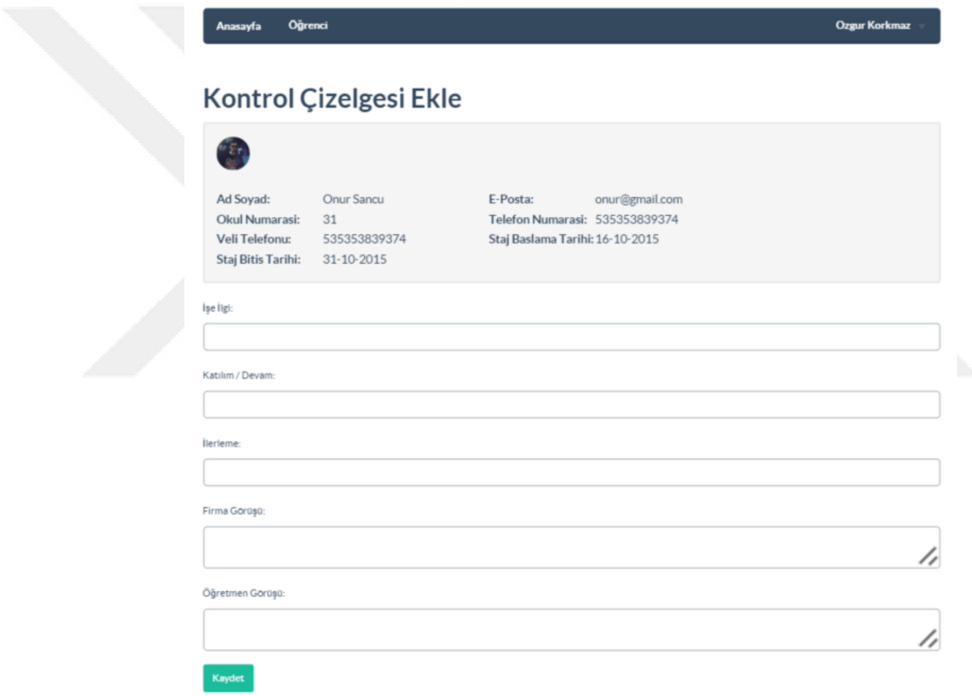

Figure 48: Teacher – Add feedback interface

4. IMPLEMENTATION & TESTING

This system is a web application that carries out the student tracking process of a vocational and technical high school in Turkey. It is responsive, mobile compatible, user friendly and it has Session based authentication for granting access to two different user roles. PHP is used as the programming language and Mysql is used as the DBMS. Laravel framework is used as the web framework while HTML5, Bootstrap and JavaScript are used for the designing the interfaces and their behaviors. Sublime text editor and Phpstorm are used as development platforms. After development, Nginx is used for deploying and running the web application.

4.1 USED TECHNOLOGIES

4.1.1 HTML5 - CSS - JavaScript

Using web standards is the biggest rule for designing. Standard development techniques guarantee the most consistent presentation and functionality with all modern browsers. HTML5, CSS and JavaScript are the modern web standards nowadays. Main reason behind the triumph of Web is HTML. HTML is a simple, text-based markup language. It can be learnt easily and can be read by any device with a basic Web browser. (Elizabeth Castro, 2012) HTML specifies the content of web pages; CSS specifies the appearance of the content and JavaScript is the programming language of the web that specifies the behavior of web pages. (Flanagan, 2011)

4.1.2 Bootstrap

Bootstrap is a front-end framework that is solely CSS and JavaScript created by Twitter. It is easy to use, provides mobile-first responsive design, consistency and complex responsive features. Bootstrap also has excellent documentation and provides many templates. There are other front-end frameworks is out there. However, browser support and performance difference between Bootstrap and others is almost negligible and are highly dependent on the context. (Radford, 2015)

4.1.3 MySQL

MySQL is the most preferred database for web applications. Open-source projects that require a full-featured database management system mostly use MySQL.

4.1.4 Nginx

Nginx is a lightweight HTTP server that is designed to handle high levels of traffic. It is efficient, lightweight and a very powerful web server. (Nedelcu, 2010)

4.1.5 PHP – Phpstorm - Laravel

PHP is a popular server side scripting language for web development, which is used for making dynamic and interactive web pages. The use of a standard programming language, such as PHP makes the maintenance task easier. During the development with simple PHP, business logic is mixed with database queries and presentation markups. Due to mixture of this development mode, the maintenance and scalability of the application becomes difficult. (Michael Schacht Hansen, 2001)

PHP has brought different development frameworks like Laravel to solve this. Laravel is based on MVC design pattern. MVC separates the application into three different layers, Model, View and Controller. The Controller takes the data from the Model, process this data and sends it to the View to render it on to the browser.

Phpstorm is an integrated development environment built to carry out the process of developing PHP based Internet applications. (Gajda, 2013)

4.2 WHY LARAVEL?

PHP: Hypertext Preprocessor is a very popular scripting language related with web development. There are many PHP frameworks now. CodeIgniter, Symfony, Yii, Cakephp, Zend Framework 2 and Laravel are the most popular and promising frameworks out there. Instead of doing similar tasks repeatedly for the same type of applications, it is much simpler to use a framework having all those resources in a package.

Let us think of a web developer who is going to develop a project with PHP. Should plain PHP or a PHP framework be chosen in the development? Which of the frameworks should be chosen in case of choosing a PHP framework?

Phalcon is PHP framework designed for people who are lazy and do not want to deal 100% of the time with low-level details like segmentation faults or memory leaks. Phalcon is a contribution to everyday work of a programmer and everyday work can be done faster. (Phalcon Team, 2015)

CodeIgniter is an open source web framework for PHP. It can run in shared hosting environments. It uses MVC design pattern. In addition, it has great library and built-in functions. (Griffiths, 2010)

Symfony is a web framework for building develop tools that let you develop faster and build applications that are more robust. The tools and concepts represent the efforts of thousands of people, over many years. By learning "Symfony", not only the fundamentals of the web development are learnt and many amazing new PHP libraries are found out too. (Team, 2016)

A performance comparison between plain PHP and some of its popular frameworks shows that Phalcon is the only lightweight framework among all the tested frameworks. Followed by CodeIgniter. Symfony and Laravel are the two heavy weight frameworks performing worst in terms of both execution time (Table 1) and memory usage (Table 2). (Samra, Comparing Performance of Plain PHP and Four of Its Popular Frameworks, 2015)

Test Targets	Create	Read	Update	Delete
Plain PHP (MVC)	17	13	17	10
Phalcon (1.3.4)	43	28	45	41
CodeIgniter (2.2)	117	101	121	105
Laravel (5.0.23)	785	752	815	726
Symfony (2.3)	787	812	1110	591

Table 1: Time execution measurement results

Test Targets	Create	Read	Update	Delete
Plain PHP (MVC)	17	13	17	10
Phalcon (1.3.4)	43	28	45	41
CodeIgniter (2.2)	117	101	121	105
Laravel (5.0.23)	785	752	815	726
Symfony (2.3)	787	812	1110	591

Table 2: Memory usage measurement results

According to the google Trends (Figure 49), Laravel is the most popular framework in 2015. (Google, 2016)

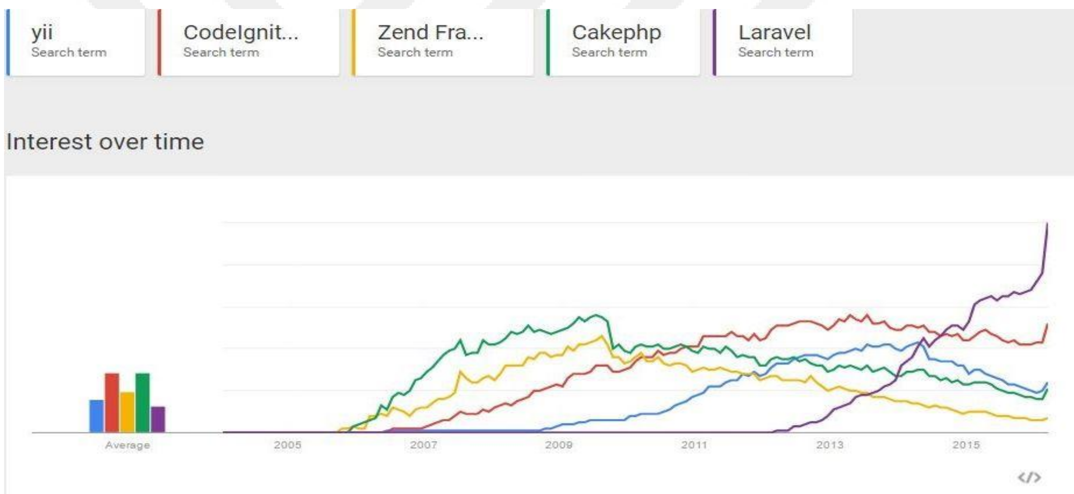

Figure 49: Comparison between CodeIgniter, Symfony, Yii, Cakephp, Zend Framework 2 and Laravel

Deciding the best framework is not easy. However, deciding a framework that could be better for a particular job or a particular developer is easier. All frameworks have strengths and weaknesses. Some developers prefer performance, some prefer better documentation and some prefer built-in functions like authentication, routing, sessions, queuing, and caching. Laravel was the only framework boosted me. It has a great community, better documentation and important built-in functions like authentication and sessions.

4.3 TESTİNG PLANS AND RESULTS

4.3.1 Testing of Functions

4.3.1.1 Login Interface

Scenario	Verify that the “E-posta” field needs @ character between two string series
Test Step	Try to login using @ character between two string series
Expected Outcome	Web site should be able to accept the login request
Actual Outcome	Web site accepts the login request

Table 3: Login interface positive

Scenario	Verify that the “E-posta” field needs @ character between two string series
Test Step	Try to login without using @ character between two string series
Expected Outcome	Web site shouldn’t be able to accept the login request
Actual Outcome	Application denies the login request and gives a warning message

Table 4: Login interface negative

4.3.1.2 Manager – Student - Add Student Interface

Scenario	Verify that “ad”, “soyad”, “E-Posta” (“E-posta” field needs @ character between two string series and “.com” at the end), Okul Numarası”, “Telefon Numarası”, “Veli Telefon Numarası”, “Staj Başlama Tarihi”, “Staj Bitiş Tarihi” fields must be filled in also “Sınıf”, “Firma”, “Kordinatör Öğretmen” fields must be chosen from the drop-down list
Test Step	Try to add a student after filling in the “ad”, “soyad”, “E-Posta” (“E-posta” field needs @ character between two string series and “.com” at the end), Okul Numarası”, “Telefon Numarası”, “Veli Telefon Numarası”, “Staj Başlama Tarihi”, “Staj Bitiş Tarihi” fields and choosing “Sınıf”, “Firma”, “Kordinatör Öğretmen”fields from drop-down lists

Expected Outcome	Web site should be able to accept the student insertion request to insert the student to database and student list page should be displayed with a confirmation message
Actual Outcome	Web site accepts the student insertion request, the student is inserted into the database and student list page is displayed with a confirmation message

Table 5: Manager – Student - Add student interface positive

Scenario	Verify that "ad", "soyad", "E-Posta" ("E-posta" field needs @ character between two string series and ".com" at the end), Okul Numarası", "Telefon Numarası", "Veli Telefon Numarası", "Staj Başlama Tarihi", "Staj Bitiş Tarihi"fields must be filled in also "Sınıf", "Firma", "Kordinatör Öğretmen" fields must be chosen from the drop-down list
Test Step	Try to add a student without filling in the "ad", "soyad", "E-Posta" ("E-posta" field needs @ character between two string series and ".com" at the end), Okul Numarası", "Telefon Numarası", "Veli Telefon Numarası", "Staj Başlama Tarihi", "Staj Bitiş Tarihi" fields or not choosing "Sınıf", "Firma", "Kordinatör Öğretmen"fields from drop-down lists
Expected Outcome	Web site shouldn't be able to accept the student insertion request so the database stays the same and a warning message should be displayed
Actual Outcome	Application denies the student insertion request and gives a warning

Table 6: Manager – Student - Add student interface negative

4.3.1.3 Manager – Student – Student's Feedbacks Interface

Scenario	Verify that "İşlemler" button must be clicked and "Kontrol Edildi Olarak İşaretle" must be chosen in order to sign the feedback as controlled
Test Step	Try to sign feedback as controlled by clicking "İşlemler" then "Kontrol Edildi Olarak İşaretle"

Expected Outcome	Web site should be able to accept the request to sign the feedback as controlled and should change related feedback table field in the database and finally the page is reloaded with a confirmation message
Actual Outcome	Web site accepts the request and gives a confirmation message and related field in the database is changed and finally the result can be seen when the same page is reloaded with a confirmation message

Table 7: Manager – Student – Student's feedbacks interface

Scenario	Verify that "İşlemler" button must be clicked and "Kontrol Edilmedi Olarak İşaretle" must be chosen in order to sign the feedback as controlled
Test Step	Try to unsign feedback by clicking "İşlemler" then "Kontrol Edilmedi Olarak İşaretle"
Expected Outcome	Web site should be able to accept the request to sign the feedback as not controlled and should change related feedback table field in the database and finally the page is reloaded with a confirmation message
Actual Outcome	Web site accepts the request and gives a confirmation message and related field in the database is changed and finally the result can be seen when the same page is reloaded with a confirmation message

Table 8: Manager – Student – Student's feedbacks interface

4.3.1.4 Manager – Student - Edit Student Interface

Scenario	Verify that "ad", "soyad", "E-Posta" ("E-posta" field needs @ character between two string series and ".com" at the end), Okul Numarası", "Telefon Numarası", "Veli Telefon Numarası", "Staj Başlama Tarihi", "Staj Bitiş Tarihi" fields must be altered to edit also "Sınıf", "Firma", "Kordinatör Öğretmen" fields must be chosen again from the drop-down list
Test Step	Try to edit student information by altering "ad", "soyad", "E-Posta" ("E-posta" field needs @ character between two string series and

	".com" at the end), Okul Numarası", "Telefon Numarası", "Veli Telefon Numarası", "Staj Başlama Tarihi", "Staj Bitiş Tarihi" fields and choosing "Sınıf", "Firma", "Kordinatör Öğretmen"fields from drop-down lists
Expected Outcome	Web site should be able to accept the student modification request to modify the student info in the database and student list page should be displayed with a confirmation message
Actual Outcome	Web site accepts the student modification request and the student info in the database is modified and finally student list page is displayed with a confirmation message

Table 9: Manager – Student - Edit student interface positive

Scenario	Verify that "ad", "soyad", "E-Posta" ("E-posta" field needs @ character between two string series and ".com" at the end), Okul Numarası", "Telefon Numarası", "Veli Telefon Numarası", "Staj Başlama Tarihi", "Staj Bitiş Tarihi" fields must be altered to edit also "Sınıf", "Firma", "Kordinatör Öğretmen" fields must be chosen again from the drop-down list
Test Step	Try to edit student information by leaving the "ad", "soyad", "E-Posta" ("E-posta" field needs @ character between two string series and ".com" at the end), Okul Numarası", "Telefon Numarası", "Veli Telefon Numarası", "Staj Başlama Tarihi", "Staj Bitiş Tarihi" fields empty or not choosing "Sınıf", "Firma", "Kordinatör Öğretmen"fields from drop-down lists
Expected Outcome	Web site should not be able to accept the student modification request to modify the student info in the database and should give a warning message
Actual Outcome	Web site does not accept the student modification request and the student info in the database is not modified and the related warning message is displayed

Table 10: Manager – Student - Edit student interface negative

4.3.1.5 Manager – Student - Delete Student Interface

Scenario	Verify that “İşlemler” then “Sil” is clicked
Test Step	Try to delete a student after clicking “İşlemler” then “Sil”
Expected Outcome	Web site should be able to accept the student deletion request to delete the student from the database and display the student list page with a confirmation message
Actual Outcome	Web site accepts the student deletion request and the student is deleted from the database and finally student list page is displayed with a confirmation message

Table 11: Manager – Student - Delete student interface positive

Scenario	Verify that “İşlemler” then “Sil” is clicked
Test Step	Try to delete a student after clicking “İşlemler” then “Sil”
Expected Outcome	Web site should not be able to accept the student deletion request to delete the student from the database with a warning message
Actual Outcome	Web site denies the student deletion with a warning message

Table 12: Manager – Student - Delete student interface negative

4.3.1.6 Manager – Class - Add Class Interface

Scenario	Verify that “İsim” field must be filled in
Test Step	Try to add a class after filling in the “İsim” field
Expected Outcome	Web site should be able to accept the insertion request to insert the class to database and display the class list with a message
Actual Outcome	Web site accepts the class insertion request and the class is inserted into the database and finally class list page is displayed with a confirmation message

Table 13: Manager – Class - Add class interface positive

Scenario	Verify that “İsim” field must be filled in
Test Step	Try to add a class by leaving the “İsim” field empty
Expected Outcome	Web site should not be able to accept the student insertion request to insert the class to database and display the class list page giving a warning message

Actual Outcome	Web site does not accept the class insertion request and the class is not inserted into the database so a warning

Table 14: Manager – Class - Add class interface negative

4.3.1.7 Manager – Class - Edit Class Interface

Scenario	Verify that “İsim” field must be altered
Test Step	Try to edit a class after altering and filling in the “İsim” field
Expected Outcome	Web site should be able to accept the class modification request to modify the class info in the database and display the class list page with a confirmation message
Actual Outcome	Web site accepts the class modification request and the class info in the database is modified. Then the class list page is displayed with a confirmation message

Table 15: Manager – Class - Edit class interface positive

Scenario	Verify that “İsim” field must be altered
Test Step	Try to edit a class without filling in the “İsim” field
Expected Outcome	Web site should not be able to accept the class modification request to modify the class info with a warning message
Actual Outcome	Web site does not accept the class modification request and the class info in the database is not modified and a warning message is displayed

Table 16: Manager – Class - Edit class interface negative

4.3.1.8 Manager – Class - Delete Class Interface

Scenario	Verify that “İşlemler” then “Sil” is clicked
Test Step	Try to delete a class after clicking “İşlemler” then “Sil”
Expected Outcome	Web site should be able to accept the class deletion request to delete the class from the database and display the class list page with a confirmation
Actual Outcome	Web site accepts the class deletion request and the class is deleted from the database and class list page is displayed with confirmation

Table 17: Manager – Class - Delete class interface positive

Scenario	Verify that “İşlemler” then “Sil” is clicked
Test Step	Try to delete a class after clicking “İşlemler” then “Sil”
Expected Outcome	Web site should not be able to accept the class deletion request to delete the class from the database and with a warning message
Actual Outcome	Web site denies the class deletion request and the class is not deleted from the database and finally class list page is displayed with a warning message

Table 18: Manager – Class - Delete class interface negative

4.3.1.9 Manager – Teacher & Manager - Add Teacher & Manager Interface

Scenario	Verify that “Ad”,”Soyad”, “E-Posta”, “Telefon Numarası”, “Şifre” fields must be filled in and a role must be chosen
Test Step	Try to add a Teacher or a Manager after filling in the “Ad”,”Soyad”, “E-Posta”, “Telefon Numarası”, “Şifre” fields and choosing a role.
Expected Outcome	Web site should be able to accept the teacher or manager insertion request and display the teacher & manager list page should be displayed with a confirmation message
Actual Outcome	Web site accepts the insertion request, the teacher or the manager is inserted into the database and the teacher & manager list page is displayed with a confirmation message

Table 19: Manager – Teacher & Manager - Add teacher & manager interface positive

Scenario	Verify that “Ad”,”Soyad”, “E-Posta”, “Telefon Numarası”, “Şifre” fields must be filled in and a role must be chosen
Test Step	Try to add a class without filling in the “Ad”,”Soyad”, “E-Posta”, “Telefon Numarası”, “Şifre” fields or not choosing a role.
Expected Outcome	Web site should be able to accept the teacher or the manager modification request to modify their info in the database and teacher & manager list page should be displayed
Actual Outcome	Web site does not accept the insertion request and the teacher or the manager is not inserted and a warning message is displayed

Table 20: Manager – Teacher & Manager - Add teacher & manager interface negative

4.3.1.10 Manager – Teacher & Manager - Edit Teacher & Manager Interface

Scenario	Verify that "Ad","Soyad", "E-Posta", "Telefon Numarası", "Şifre" fields must be filled in and a role must be chosen
Test Step	Try to edit a Teacher by filling in the "Ad","Soyad", "E-Posta", "Telefon Numarası", "Şifre" fields and choosing a role.
Expected Outcome	Web site should be able to accept the teacher or the manager modification request to modify their info in the database and teacher & manager list page should be displayed
Actual Outcome	Web site accepts the modification request and the teacher or the manager info in the database is modified then teacher & manager list page is displayed with a confirmation message

Table 21: Manager – Teacher & Manager - Edit teacher & manager interface positive

Scenario	Verify that "Ad","Soyad", "E-Posta", "Telefon Numarası", "Şifre" fields must be filled in and a role must be chosen
Test Step	Try to edit a Teacher or a Manager leaving the "Ad","Soyad", "E-Posta", "Telefon Numarası", "Şifre" fields empty or without choosing a role.
Expected Outcome	Web site should not be able to accept the teacher or the manager modification request to modify the teacher or the manager info
Actual Outcome	Web site does not accept the teacher or the manager modification request and the teacher or the manager info in the database is not modified and a warning message is displayed

Table 22: Manager – Teacher & Manager - Edit teacher & manager interface negative

4.3.1.11 Manager – Teacher & Manager - Delete Teacher & Manager Interface

Scenario	Verify that "İşlemler" then "Sil" is clicked
Test Step	Try to delete teacher or manager after clicking "İşlemler" then "Sil"
Expected Outcome	Web site should be able to accept the teacher or manager deletion request to delete teacher or manager from the database and display the teacher & manager list page with a confirmation message

Actual Outcome	Web site accepts teacher or manager deletion request and teacher or manager is deleted from the database and finally teacher & manager list page is displayed with a confirmation message

Table 23: Manager – Teacher & Manager - Delete teacher & manager interface positive

Scenario	Verify that “İşlemler” then “Sil” is clicked
Test Step	Try to delete teacher or manager after clicking “İşlemler” then “Sil”
Expected Outcome	Web site should not be able to accept the teacher or manager deletion request to delete teacher or manager from the database with a warning message
Actual Outcome	Web site denies teacher or manager deletion request and teacher or manager is not deleted from the database and finally teacher & manager list page is displayed with a warning message

Table 24: Manager – Teacher & Manager - Delete teacher & manager interface negative

4.3.1.12 Manager – Company - Add Company Interface

Scenario	Verify that “İsim”, “Adres”, “Map Link”, “Yönetici Ad”, “Yönetici Telefon”, “Yönetici E-Posta” fields must be filled in
Test Step	Try to add a company after filling in the “İsim”, “Adres”, “Map Link”, “Yönetici Ad”, “Yönetici Telefon”, “Yönetici E-Posta” fields
Expected Outcome	Web site should be able to accept the company insertion request to insert the company to database
Actual Outcome	Web site accepts the company insertion request and the company is inserted into the database

Table 25: Manager – Company - Add company interface positive

Scenario	Verify that “İsim”, “Adres”, “Map Link”, “Yönetici Ad”, “Yönetici Telefon”, “Yönetici E-Posta” fields must be filled in
Test Step	Try to add a class by leaving the “İsim”, “Adres”, “Map Link”, “Yönetici Ad”, “Yönetici Telefon”, “Yönetici E-Posta” fields empty
Expected Outcome	Web site should not be able to accept the student insertion request to insert the company to database

Actual Outcome	Web site does not accept the company insertion request and the company is not inserted into the database

Table 26: Manager – Company - Add company interface negative

4.3.1.13 Manager – Company - Edit Company Interface

Scenario	Verify that "İsim", "Adres", "Map Link", "Yönetici Ad", "Yönetici Telefon", "Yönetici E-Posta" fields must be altered
Test Step	Try to edit a class after altering and filling in the "İsim", "Adres", "Map Link", "Yönetici Ad", "Yönetici Telefon", "Yönetici E-Posta" fields
Expected Outcome	Web site should be able to accept the company modification request to modify the company info in the database and company list page should be displayed with a confirmation message
Actual Outcome	Web site accepts the company modification request and the company info in the database is modified, then company list page is displayed with a confirmation message

Table 27: Manager – Company - Edit company interface positive

Scenario	Verify that "İsim", "Adres", "Map Link", "Yönetici Ad", "Yönetici Telefon", "Yönetici E-Posta" fields must be altered
Test Step	Try to edit a class without filling in the "İsim", "Adres", "Map Link", "Yönetici Ad", "Yönetici Telefon", "Yönetici E-Posta" fields
Expected Outcome	Web site should not be able to accept the company modification request to modify the company info in the database
Actual Outcome	Web site does not accept the company modification request and the company info in the database is not modified and a warning message is displayed

Table 28: Manager – Company - Edit company interface negative

4.3.1.14 Manager – Company - Delete Company Interface

Scenario	Verify that "İşlemler" then "Sil" is clicked
Test Step	Try to delete a company after clicking "İşlemler" then "Sil"

Expected Outcome	Web site should be able to accept the company deletion request to delete the company from the database and display the company list page with a confirmation message
Actual Outcome	Web site accepts the company deletion request and the company is deleted from the database and finally company list page is displayed with a confirmation message

Table 29: Manager – Company - Delete company interface positive

Scenario	Verify that "İşlemler" then "Sil" is clicked
Test Step	Try to delete a company after clicking "İşlemler" then "Sil"
Expected Outcome	Web site should not be able to accept the company deletion request to delete the company from the database with a warning
Actual Outcome	Web site denies the request and the company is not deleted

Table 30: Manager – Company - Delete company interface negative

4.3.1.15 Teacher – Student - Add Feedback Interface

Scenario	Verify that "İşe İlgi", "Katılım/Devam", "İlerleme", "Firma Görüşü", "Öğretmen Görüşü" fields must be filled in
Test Step	Try to add a feedback to the student after filling in the ""İşe İlgi", "Katılım/Devam", "İlerleme", "Firma Görüşü", "Öğretmen Görüşü" fields
Expected Outcome	Web site should be able to accept the feedback insertion request to insert the feedback to database then feedback list page should be displayed with a confirmation message
Actual Outcome	Web site accepts the feedback insertion request and the feedback is inserted into the database displaying feedback list page with a confirmation message

Table 31: Teacher – Student – Add feedback interface positive

Scenario	Verify that "İşe İlgi", "Katılım/Devam", "İlerleme", "Firma Görüşü", "Öğretmen Görüşü" fields must be filled in
Test Step	Try to add a feedback to the student by leaving the "İşe İlgi", "Katılım/Devam", "İlerleme", "Firma Görüşü", "Öğretmen Görüşü"

	fields empty
Expected Outcome	Web site should not be able to accept the feedback insertion request to insert the feedback to database
Actual Outcome	Web site does not accept the feedback insertion request and the feedback is not inserted into the database with a warning message

Table 32: Teacher – Student – Add feedback interface negative

4.3.1.16 Teacher – Student – Edit Feedback Interface

Scenario	Verify "İşe İlgi", "Katılım/Devam", "İlerleme", "Firma Görüşü", "Öğretmen Görüşü" fields must be altered
Test Step	Try to edit a feedback after altering and filling in the "İşe İlgi", "Katılım/Devam", "İlerleme", "Firma Görüşü", "Öğretmen Görüşü" fields
Expected Outcome	Web site should be able to accept the feedback modification request to modify the feedback info in the database and should display the feedback list page with a confirmation message
Actual Outcome	Web site accepts the feedback modification request and the feedback info in the database is modified then feedback list page is displayed with a confirmation message

Table 33: Teacher – Student – Edit feedback interface positive

Scenario	Verify that "İşe İlgi", "Katılım/Devam", "İlerleme", "Firma Görüşü", "Öğretmen Görüşü" fields must be altered
Test Step	Try to edit a feedback without filling in the "İşe İlgi", "Katılım/Devam", "İlerleme", "Firma Görüşü", "Öğretmen Görüşü" fields
Expected Outcome	Web site should not be able to accept the feedback modification request to modify the feedback info in the database with a warning message
Actual Outcome	Web site does not accept the feedback modification request with a warning message

Table 34: Teacher – Student – Edit feedback interface negative

4.3.1.17 Teacher – Student - Delete Feedback Interface

Scenario	Verify that “İşlemler” then “Sil” is clicked
Test Step	Try to delete a feedback after clicking “İşlemler” then “Sil”
Expected Outcome	Web site should be able to accept the feedback deletion request to delete the feedback from the database and display the feedback list page with a confirmation message
Actual Outcome	Web site accepts the feedback deletion request and the feedback is deleted from the database and finally feedback list page is displayed with a confirmation message

Table 35: Teacher – Student - Delete feedback interface positive

Scenario	Verify that “İşlemler” then “Sil” is clicked
Test Step	Try to delete a feedback after clicking “İşlemler” then “Sil”
Expected Outcome	Web site should not be able to accept the feedback deletion request to delete the feedback from the database with a warning message
Actual Outcome	Web site denies the feedback deletion request and the feedback is not deleted from the database and finally feedback list page is displayed with a warning message

Table 36: Teacher – Student - Delete feedback interface positive

4.3.2 Testing of Non-Functions

4.3.2.1 Audit-Ability

Members of staff were controlled during their work every week for a month. Every member of staff was on track to the specific task had given to them and the system worked accordingly.

4.3.2.2 Availability

The system operated 24 hours a day for a month. Every working sector was operating perfectly.

4.3.2.3 Backup and Restore

The system never failed during test month. So, it was restored from a recent backup to see the restoration process was completed without any problems.

4.3.2.4 Capacity

Every member of the school staff (120 people) was logged into the system all at once. The system still performed to a very high and efficient level. System's performance was checked to ensure that it could satisfy the amount of users logged in at one time, without causing the system to slow down during the test month and it performed efficiently.

Every day the data stored in the system was examined to check that it could handle data stored. Server handled the situation without any trouble.

4.3.2.5 Security

All major web-testing methods were used.

- Internal url was pasted directly into address bar without login. Internal pages did not open.
- Url options were changed directly after a successful login. Access denied this user to view others pages.
- Invalid inputs in input fields like login username, password, and input text boxes were tried. The system reactions were controlled.
- SSL is tested for security measures. A message was displayed when user switch from non-secure pages to secure pages.
- All transactions, error messages, security breach attempts were logged in log files on web server.

4.3.2.6 Usability

System was monitored to ensure the layout and structure is consistent. This made it very easy for the user to work in the environment provided. System was used every day therefore this was tested daily to ensure that the language and the keyboard layouts are of the Turkish standard. This was useful because all members of staff should be able to use and work with the Turkish language and keyboard.

4.3.2.7 Reliability

- Mean Time Between Failures - The system was 100% operational 99.9% of the calendar month during its first month of operation.

- Mean Time To Recovery - If broken, three hours was enough to get the system backed up.

4.3.2.8 Performance

Web Load Testing and Web Stress Testing were done firstly.

- All staff (120 people) was logged into the system all at once. System's performance was checked to ensure that it could satisfy the amount of users logged in at one time, without causing the system to slow down during the test month and it performed efficiently. Site handled many simultaneous user requests, large input data from users, Simultaneous connection to database, heavy load on specific pages etc.

- Web Stress Testing: Web stress testing was performed to break the site by giving stress. The system handled the stress and recovered from crashes.

After these tests, Site Analyzer was used to see the performance of the system. Site-analyzer gets a complete diagnosis of your pages one by one. Content, keywords, design, performance, accessibility are all diagnosed deeply. Programmers can monitor developments with Site Analyzer Score, improve the website. All web pages can be analyzed in times of need. Installation is not required and the process is very fast. (SITEANALYSER, 2016)

- 1.02 KB download is needed to get your HTML page. Your webpages should not go over 50 KB (without images and other media. Only the weight of the page is considered).

- Execution time does not exceed 0.5 seconds. The execution time is the time a server needs to generate the webpage. The time in not related to the internet connection but depends on the server configuration and the associated code.

- The website does not allow page caching. Caching webpages will decrease their loading time by storing them temporarily on the user's device.

- Server uses file compression. Compression rate is 60%. This enables to optimize loading time by decreasing the amount of data to be downloaded. The server compresses the data while sending to the client, then the browser decompresses and displays it.

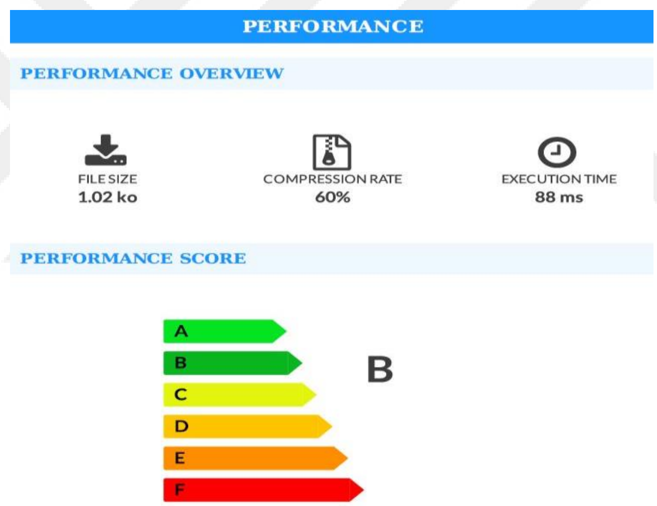

Figure 50: Site Analyzer test results.

4.3.2.9 Maintainability

The system facilitates the correction of errors can be expanded to satisfy new or changing requirements, which includes enhancing existing functions and correcting code errors.

Early planning provides us a way to see design problems, and is the most effective way for modifying or adding features later. Early in the definition phase, expected changes were identified. The system functionality was decomposed into manageable segments. The format of these modules were standardized to make adding, deleting or modifying code easier for expected or unexpected changes.

Instead of having a mountain of code, the application was divided into small meaningful modules that can make the development more manageable and maintainable. Considering the scale of application which is huge, the different modules that it was going to have, instead of blending everything in the directories that Laravel provides, modules was implemented separately.

The application code was generated based on objects and their behaviors and it was built without thinking about loops, branches and control structures because events cause changes in the object-oriented technique. Each state change is simple to be programed, so the application was divided into pieces. Each object performs a specific function unrelated to other objects. Therefore, the classes can be changed as much as wanted independent of other classes. So, testing and modifying the class becomes easier.

Software coding standards are important in producing maintainable code. All code was done according to these standards during both the development and maintenance phases. In addition, documentation standards, release standards, common languages and other methods were utilized.

4.3.2.10 Privacy

The system is maintained and revised every day to insure customers' details are kept safe and not leaked. Their details have to be kept private and only be used when necessary.

5. PROJECT MANAGEMENT

This Chapter would include information about the Waterfall Approach Model that is used for the project and a Gant Chart of the project would be given showing activities displayed against time.

5.1 Waterfall Approach

The waterfall model suggests that programmers should work in phases, rather than directly and instantly developing a product. They should complete each phase and verify the quality of completed phase then move on so next phase can have a good base. The waterfall model also allows stepping back to earlier phases when there is a problem. (Lethbridge & Laganière, 2005)

The sequential phases in Waterfall model:

- **Requirement Gathering and analysis:** Functional and non-functional requirements are defined in this phase. Use cases, which describe the users' interactions with the software. (Bassil, 2011)

- **System Design:** It is the process of planning and problem solving for a software solution. It deals with choosing the appropriate algorithm design, software architecture design, database conceptual schema, logical diagram design, and data structure definition. (Adel Alshamrani, 2015)

- **Implementation:** In this phase a product was built from the whole requirements (Adel Alshamrani, 2015)

- **Integration and Testing:** Software solutions that have been developed are tested and checked in this phase. In addition, bugs and system glitches are found, fixed up, and refined in this phase. (Adel Alshamrani, 2015)

- **Deployment of system:** After the tests, the product is released into the market. (Adel Alshamrani, 2015)

- **Maintenance:** After the release, concerns like the modifications, improvements and errors correction are taken care in this phase. (Adel Alshamrani, 2015)

5.2 Gant Chart

Gantt charts do not need any explanation or training because anyone who can understand a calendar should be able to understand it. (Lock, 2007)

Gantt charts are great and very effective visual aids and their effectiveness can be improved using of different colors. Gantt charts are useful for very simple resource scheduling because the amount of any particular kind of resource needed in a given time period can be calculated by adding the number of times that tasks needing that resource appear in each period column. (Lock, 2007)

To summarize, a Gantt chart shows you what has to be done (the activities) and when (the schedule).

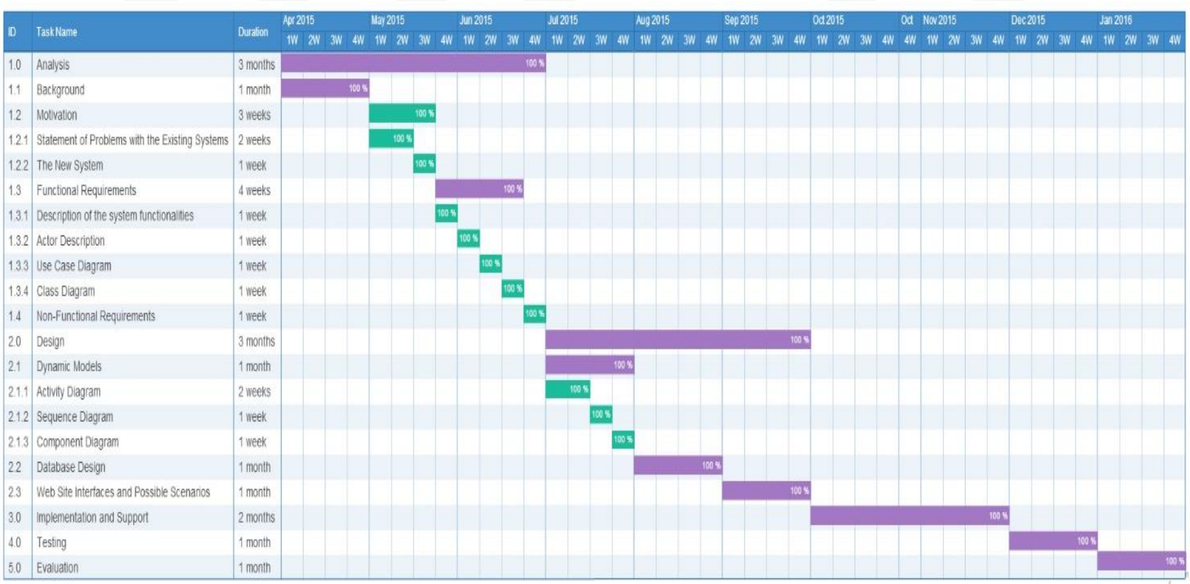

Figure 51: Gantt chart of the project

The figure 51 above shows the Gant chart of the project. After deciding what is to be done, first phase was the analysis. The analysis phase resumed for three months. First month was for the background searching and then three weeks was for the motivation phase that included statements of the problems with the existing systems and how the new system is going to be like. Then another month was spent on functional requirements that included description of the system functionalities, actor description and use case diagram after that a week was spent on the non-functional requirements. Second phase was the design. The design phase was resumed for three months. First month was for the dynamic models: activity diagram, sequence diagram and component diagram and then another month was for the database design and another for the web site interfaces and possible scenarios. After completing the design phase, the implementation phase was about two months then another month spent on for the testing part finally another month for the evaluation.

6. EVALUATION

Every school or educational organization wish to have modern management systems that can simplify every aspect of the administrative functions in a seamless manner. Precisely, the Student Tracking System offers a helping hand to the administrative sector by reducing the paper works and other burdens of their staffs. Such software solutions are electronically operated systems that automate all the administrative tasks, which in turn, improve the efficiency of the student tracking management.

Student Tracking System was evaluated in a vocational and technical high school for a month. Managers and teachers were given a lecture about the web site. Manager added the required data to the system. Teachers visited the students at work and added gathered feedbacks to the web site.

The teachers and the manager expressed their experiences and feelings during the process. Teachers pointed out that the system simplifies the process by providing features like the pictures of the students, phone number of students and their parents and all the features can be accessed from their mobile phones or tablets. They also pointed out that recording feedbacks to the database simplifies monitoring students' developments, reduce paper work and may create career opportunities for students. Managers pointed out that it made to process easier to manage.

Use of latest technologies like Bootstrap, Laravel, Html5 and Mysql makes the system secure and reliable and user friendly. These technologies also provides high performance and great maintainability.

6.1 PROS

- The Student Tracking System is the first intern tracking system for vocational schools in Turkey.

- The Student Tracking System is a unique system. Because the vocational education in Turkey is unique.

- The Student Tracking System can be used by all the vocational schools in Turkey even if their approach the student tracking is very different. It can be used by small vocational high schools and departments of big vocational high schools can use it just for their department.

- Provides a paperless solution for tracking and archiving students' placement and data.

- Allows students to research potential field placement descriptions and learning opportunities.

- Easily expand company database and make revisions or updates as needs change.

- Allows technical instructor to describe learning opportunities in order to match students in ideal learning environments.

- Provides teachers with a resource to easily access student information from their computer or even mobile devices.

- Allows teachers' access to the all data in a paperless system.

- Increases communication between the School and companies.

6.2 CONS

a) The system lacks the academic year part. At the end of each year, the database must be backed up. Former students and teachers must be deleted.

b) The system is developed to be used by all the vocational schools in Turkey. It may not be the best solution for a specific school.

c) An announcement section can be used for the teacher in the middle of the teacher main page.

d) Printing of student list for teacher with a date and company signature and stamp fields can be added to system then teachers can get the document to company to be signed and stamped and return.

7. CONCLUSION

The Internet has become a major resource in modern business, thus tracking systems has gained significance not only from the school's managers but also from the technical teachers' point of view. For the school manager, Student Tracking System generates new opportunities to simplify the tracking process and for the teachers, it makes it easier to track students and their data. In-company training is a big part of the vocational education and have big impact on students' career. Therefore, recording the process with detailed information is very important for students too. Hence, we have designed the project to provide the user with easy navigation, retrieval of data and necessary feedback as much as possible.

In this project, the user is provided with a student tracking web site that can be used to track students online. To implement this as a web application we used LARAVEL as the Technology. LARAVEL has several advantages such as enhanced performance, scalability, built-in security and simplicity. To build any web application using LARAVEL we need a programming language such as PHP; so PHP was the language used to build this application. LARAVEL uses MySQL to interact with the database as it can easily deploy and maintain a LARAVEL application, provides fast data access, easy installation and simplicity.

A good design must be accompanied with user-friendly application logic. It should be convenient for the manager to view the contents of the system and to be able to remove or add students, teachers or managers, companies, classes to the system and for teachers to be able to view students and add feedbacks to them. The application described in this project provides a number of features that are designed to make the users more comfortable.

This project helps in understanding the creation of an interactive web page and the technologies used to implement it. The building of the project has given me a precise knowledge about how LARAVEL is used to develop a website, how it connects to the database to access the data and how the data and web pages are modified to provide the user with an application.

7.1 FUTURE WORK

There are some limitations for the current system to which solutions can be provided as a future development:

a) A button to delete all students can be added to provide manager to delete all the students when needed because the system lacks the academic year feature or the academic year feature can be added.

b) The executives of Ministry of National Education can be given a presentation about the tracking system and importance of practical training that leads to integration of standard Student Tracking System for all vocational high schools to e-Okul. This project and results can be used for this project with some modification.

   It might be nearly impossible to integrate student-tracking system with e-Okul now. Because, all technical and vocational high schools should perform the same kind of student tracking process. However, it is impossible because guides are hermeneutical and every manager might interpret it differently. Therefore, before the integration, a standard student tracking process should be formed. MoNE should prepare a new guideline that is not hermeneutical and all technical and vocational high schools should follow it.

c) After the integration, career sites like “kariyer.net” can be integrated to the system, which may lead to the new horizons for the vocational education.

References

Adel Alshamrani, A. B. (2015). A Comparison Between Three SDLC Models Waterfall Model Spiral Model and Incremental/Iterative Model. *IJCSI International Journal of Computer Science Issues*, 106-111.

Bassil, Y. (2011). *A Simulation Model for the Waterfall Software Development Life Cycle.* Lebanon: International Journal of Engineering & Technology (iJET).

Bhattacharjee, A., & Shyamasundar, R. (2009, January). *Activity Diagrams : A Formal Framework to Model Business Processes and Code Generation.* Mumbai: ETH Zurich, Chair of Software Engineering.

Cockburn, A. (2001). *Writing Effective Use Cases.* Addison-Wesley, c.

Coronel, C., Morris, S., & Rob, P. (2012). *Database Systems. Design, Implementation and Management.* Boston: Cengage Learning.

Elizabeth Castro, B. H. (2012). *HTML5 and CSS3.* Berkeley: Peachpit Press.

Flanagan, D. (2011). *JavaScript: The Definitive Guide.* Sebastopol: O'Reilly Media, Inc.

Gajda, W. (2013). *Instant PhpStorm Starter.* Birmingham: Packt Publishing Ltd.

Garcia-Molina, H., Ullman, J. D., & Widom, J. (2002). *DATABASE SYSTEMS The Complete Book.* New Jersey: Pearson Prentice Hall.

Garousi, V., Briand, L., & Labiche, Y. (2005, September). *Control Flow Analysis of UML 2.0 Sequence Diagrams.* Ottawa,: Software Quality Engineering Laboratory.

Google. (2016, 3). *Google Trends*. Google Trends: http://www.google.com/trends/explore?hl=en-US#q=yii,+CodeIgniter,+Zend+Framework,+Cakephp,+Laravel&cmpt=q&tz&tz

Griffiths, A. (2010). *CodeIgniter 1.7 Professional Development.* Birmingham: Packt Publishing Ltd.

Lethbridge, T. C., & Laganière, R. (2005). *Object-Oriented Software Engineering Practical software development using UML and Java.* New York: McGraw-Hill Education.

Lock, D. (2007). *Project Management.* Hampshire: Gower Publishing Limited.

Marsic, I. (2012). *Software Engineering.* Rutgers University: New Jersey.

Michael Schacht Hansen, L. R. (2001). *Server-side scripting with PHP4.* Denmark: University of Aarhus.

Nedelcu, C. (2010). *Nginx HTTP Server.* Birmingham: Packt Publishing Ltd.

Phalcon Team. (2015). *Phalcon PHP Framework Internal.*

Radford, S. (2015). *Learning Web Development with Bootstrap and AngularJS.* BIRMINGHAM - MUMBAI: Packt Publishing.

Ramakrishnan, R., & Gehrke, J. (2003). *Database Management Systems.* New York,: McGraw-Hill Higher Education.

Ricardi, G. (2002). *Database Management with Web Site Development Applications.* New Jersey: Prentice Hall; United States ed edition.

Samra, J. (2015). Comparing Performance of Plain PHP and Four of Its Popular Frameworks. *Thesis Project*. Växjö & Kalmar, Småland, Sweden.

Silberschatz, A., Korth, & Sudarshan. (2001). *Database System Concepts, Fourth Edition.* © The McGraw−Hill.

SITEANALYSER. (2016, 03 26). SITEANALYSER: https://www.site-analyzer.com/en/features/pageanalysis

Team, S. (2016). *The Book.* SensioLabs .

TÜNCEL, M., AKYÜZ, Ü., ALTIN, D. R., & YALÇIN, O. (2010, 02 07). *REFORMING TURKISH VOCATIONAL EDUCATION ANDte.* Ankara: Punto Grafik Tasarım & Baskı. Retrieved from http://projeler.meb.gov.tr/pkmtr/document/e_kitap/vet_reform_turkey.pdf

Watson, R. T. (2006). *Information Systems.* Georgia: Global Text Project.