

CS 353

Database Systems  
Spring 2015

**SITS – Student Internship Tracking System**

Final Report

Group Members  
Sinem Sav - 21102143

Doğancan Demirtaş - 21201824

Can Akgün - 21102268

GROUP-18

Table Of Contents

[1. DESCRIPTION OF THE SYSTEM 3](#_Toc419818468)

[2. FINAL E/R DIAGRAM 4](#_Toc419818469)

[3. FINAL TABLES 5](#_Toc419818470)

[3.1 Faculty 5](#_Toc419818471)

[3.2 Department 6](#_Toc419818472)

[3.3 Person 7](#_Toc419818473)

[3.4 Student 8](#_Toc419818474)

[3.5 StudentAdvisor 9](#_Toc419818475)

[3.6 Secretary 10](#_Toc419818476)

[3.7 Company 11](#_Toc419818477)

[3.8 RegisteredCompany 12](#_Toc419818478)

[3.9 SelfFoundCompany 13](#_Toc419818479)

[3.10 Quota 14](#_Toc419818480)

[3.11 Opens 15](#_Toc419818481)

[3.12 Application 16](#_Toc419818482)

[3.13 QuotaApply 17](#_Toc419818483)

[3.14 DirectApply 18](#_Toc419818484)

[3.15 Announcement 19](#_Toc419818485)

[3.16 AppFeedbackAnnouncement 20](#_Toc419818486)

[3.17 GeneralAnnouncement 21](#_Toc419818487)

[4. FUNCTIONAL DEPENDENCIES AND NORMALIZATION 22](#_Toc419818488)

[5. FUNCTIONAL COMPONENTS 23](#_Toc419818489)

[5.1 Use Cases 23](#_Toc419818490)

[5.2 Scenarios 26](#_Toc419818491)

[6. User Interface Design and Corresponding SQL Statements 29](#_Toc419818492)

[6.1 Login 29](#_Toc419818493)

[6.2 My Applications - Student 30](#_Toc419818494)

[6.3 Show Quotas - Student 32](#_Toc419818495)

[6.4 Quota Apply - Student 34](#_Toc419818496)

[6.5 Show Companies - Student 35](#_Toc419818497)

[6.6 Add Company - Student 36](#_Toc419818498)

[6.7 Direct Apply - Student 37](#_Toc419818499)

[6.8 Announcements - Student 38](#_Toc419818500)

[6.9 Approve Company – Advisor 39](#_Toc419818501)

[6.10 See Quotas - Secretary 41](#_Toc419818502)

[6.11 Add Quota - Secretary 42](#_Toc419818503)

[6.12 See Detailed Quota Applications - Secretary 43](#_Toc419818504)

[6.13 Approve Applications - Secretary 44](#_Toc419818505)

[6.14 See Announcements – Secretary 45](#_Toc419818506)

[6.15 Add Announcement – Secretary 46](#_Toc419818507)

[7. ADVANCED DATABASE COMPONENTS 47](#_Toc419818508)

[7.1 Reports 47](#_Toc419818509)

[7.1.1 Total Number Of Registered and Self-Found Companies 47](#_Toc419818510)

[7.1.2 Total Number of Applications 47](#_Toc419818511)

[7.1.3 Total Number of Approved Applications 47](#_Toc419818512)

[7.2 Views 47](#_Toc419818513)

[7.2.1 Secretary-Quota View 47](#_Toc419818514)

[7.2.2 Secretary-FilteredQuota View 48](#_Toc419818515)

[7.2.3 Student-Feedback Announcement View 48](#_Toc419818516)

[7.3 Triggers 48](#_Toc419818517)

[7.4 Constraints 50](#_Toc419818518)

[7.5 Stored Procedures 51](#_Toc419818519)

[8. IMPLEMENTATION DETAILS 51](#_Toc419818520)

[9. USER MANUAL 53](#_Toc419818521)

[9.1 User Manual for Student 54](#_Toc419818522)

# 1. DESCRIPTION OF THE SYSTEM

Student Internship Tracking System is implemented for the Bilkent University students and to keep track of their internship processes. In our system, there are three user types namely students, department secretaries and advisors. In this system, students can login and see the list of existing companies in the system, see the company quoas and apply for those quotas, see their applications and their results (if the result is positive they can accept or decline this quota) as well as the announcements. When a students find a new company for the internship, they can add this company to the system and wait for the approval for this company. The one who approves these companies is the advisors. They can login to the system, see the self-found companies and approve/disapprove them. Secretaries can open quotas, make announcements and approve applications.

With our system, all processes about internship will be much easier compared to traditional pen and paper works. Students, secretaries and advisors will save from their time and energy with the use of this system in keeping track of internship processes.

# 2. FINAL E/R DIAGRAM

# 3. FINAL TABLES

## 3.1 Faculty

**Relational Model:**

Faculty(facultyName)

**Functional Dependencies:**

No functional dependency.

**Candidate Keys:**

{ (facultyName) }

**Normal Form:**

BCNF

**Table Definition:**

CREATE TABLE faculty(

facultyName char(30) PRIMARY KEY

)ENGINE = InnoDB;

## 3.2 Department

**Relational Model:**

Department(deptName, facultyName)

**Functional Dependencies:**

deptName -> facultyName

**Candidate Keys:**

{ (deptName) }

**Normal Form:**

BCNF

**Table Definition:**

CREATE TABLE department(

deptName char(5) PRIMARY KEY,

facultyName char(30),

FOREIGN KEY (facultyName) references faculty(facultyName) )ENGINE = InnoDB;

## 3.3 Person

**Relational Model:**

Person (userID, name, surname, password, phone, deptName, userType)

**Functional Dependencies:**

userID -> name, surname, password, phone, deptName, userType

**Candidate Keys:**

{ (userID) }

**Normal Form:**

BCNF

**Table Definition:**

CREATE TABLE person(

userID int PRIMARY KEY AUTO\_INCREMENT,

name char(100) NOT NULL,

surname char(100) NOT NULL

password char(12) NOT NULL,

phone char(15),

deptName char(5),

userType char(20),

FOREIGN KEY (deptName) references department(deptName)

)ENGINE = InnoDB;

## 3.4 Student

**Relational Model:**

Student (userID, cgpa)

**Functional Dependencies:**

userID -> cgpa

**Candidate Keys:**

{ (userID) }

**Normal Form:**

BCNF

**Table Definition:**

CREATE TABLE student(

userID int PRIMARY KEY,

cgpa float,

FOREIGN KEY (userID) references person(userID)

)ENGINE = InnoDB;

## 3.5 StudentAdvisor

**Relational Model:**

Student (userID)

**Functional Dependencies:**

No functional dependency.

**Candidate Keys:**

{ (userID) }

**Normal Form:**

BCNF

**Table Definition:**

CREATE TABLE studentAdvisor(

userID int PRIMARY KEY,

FOREIGN KEY (userID) references person(userID)

)ENGINE = InnoDB;

## 3.6 Secretary

**Relational Model:**

Secretary (userID)

**Functional Dependencies:**

No functional dependency.

**Candidate Keys:**

{ (userID) }

**Normal Form:**

BCNF

**Table Definition:**

CREATE TABLE secretary(

userID int PRIMARY KEY,

FOREIGN KEY (userID) references person(userID)

)ENGINE = InnoDB;

## 3.7 Company

**Relational Model:**

Company (compID, name, password, address, phone, applicableDepts, status, supervisorName, supervisorPhone, city, country, evaluatorRating, studentRating, sector)

**Functional Dependencies:**

compID -> name, password, address, phone, applicableDepts, status, supervisorName, supervisorPhone, city, country, evaluatorRating, studentRating, sector)

**Candidate Keys:**

{ (compID) }

**Normal Form:**

BCNF

**Table Definition:**

CREATE TABLE company(

compID int PRIMARY KEY AUTO\_INCREMENT,

name char(100) NOT NULL,

password char(12) NOT NULL,

address char(100),

phone char(15),

applicableDepts char(30),

status char(15) NOT NULL,

supervisorName char(30),

supervisorPhone char(15),

city char(20),

country char(20),

evaluatorRating int,

studentRating int,

sector char(20)

)ENGINE = InnoDB;

## 3.8 RegisteredCompany

**Relational Model:**

RegisteredCompany (compID, registrationDate, studentID)

**Functional Dependencies:**

compID -> registrationDate, studentID

**Candidate Keys:**

{ (compID) }

**Normal Form:**

BCNF

**Table Definition:**

CREATE TABLE registeredCompany(

compID int PRIMARY KEY,

registrationDate TIMESTAMP,

FOREIGN KEY (compID) references company(compID)

)ENGINE = InnoDB;

## 3.9 SelfFoundCompany

**Relational Model:**

SelfFoundCompany (compID, advisorID)

**Functional Dependencies:**

compID -> advisorID

**Candidate Keys:**

{ (compID) }

**Normal Form:**

BCNF

**Table Definition:**

CREATE TABLE selfFoundCompany (

compID int PRIMARY KEY,

advisorID int,

FOREIGN KEY (compID) references company(compID),

FOREIGN KEY (advisorID) references studentAdvisor(userID)

)ENGINE = InnoDB;

## 3.10 Quota

**Relational Model:**

Quota (quotaID, internshipDuration, internshipStartDate, internshipEndDate, availableYears, status, quotaAmount, quotaDeadline,compID, generalAnnouncementID)

**Functional Dependencies:**

quotaID -> internshipDuration, internshipStartDate, internshipEndDate,availableYears, status,

quotaAmount, quotaDeadline, compID, generalAnnouncementID

**Candidate Keys:**

{ (quotaID) }

**Normal Form:**

BCNF

**Table Definition:**

CREATE TABLE quota(

quotaID int PRIMARY KEY AUTO\_INCREMENT,

generalAnnouncementID int,

compID int,

internship\_Duration int NOT NULL,

internshipStart\_Date date NOT NULL,

internshipEnd\_Date date NOT NULL,

availableYears int NOT NULL,

status char(30),

quotaAmount int NOT NULL,

quotaDeadline date NOT NULL,

FOREIGN KEY (generalAnnouncementID) references generalAnnouncement(announcementID),

FOREIGN KEY (compID) references company(compID)

)ENGINE = InnoDB;

## 3.11 Opens

**Relational Model:**

Opens(quotaID, compID, deptName)

**Functional Dependencies:**

quotaID -> compID, deptName

**Candidate Keys:**

{ (quotaID) }

**Normal Form:**

BCNF

**Table Definition:**

CREATE TABLE opens (

quotaID int,

compID int,

deptName char(5),

FOREIGN KEY (quotaID) references quota(quotaID),

FOREIGN KEY (compID) references registeredCompany (compID),

FOREIGN KEY (deptName) references department(deptName)

)ENGINE = InnoDB;

## 3.12 Application

**Relational Model:**

Application (appID, appSubmitDate, approval, appType, secretaryID)

**Functional Dependencies:**

appID ->appSubmitDate, approval, appType, secretaryID

**Candidate Keys:**

{ (appID) }

**Normal Form:**

BCNF

**Table Definition:**

CREATE TABLE application(

appID int PRIMARY KEY AUTO\_INCREMENT,

appSubmitDate TIMESTAMP,

approval char(12),

appType char(12),

secretaryID int,

FOREIGN KEY (secretaryID) references secretary(userID)

)ENGINE = InnoDB;

## 3.13 QuotaApply

**Relational Model:**

QuotaApply(appID, quotaID, compID, studentID, drawResult, announced)

**Functional Dependencies:**

appID ->quotaID, compID, studentID, drawResult, announced

**Candidate Keys:**

{ (appID) }

**Normal Form:**

BCNF

**Table Definition:**

CREATE TABLE quotaApply(

appID int PRIMARY KEY,

quotaID int,

compID int,

studentID int,

drawResult int,

announced int,

FOREIGN KEY (appID) references application(appID),

FOREIGN KEY (quotaID) references quota(quotaID),

FOREIGN KEY (compID) references registeredCompany(compID),

FOREIGN KEY (studentID) references student(userID)

)ENGINE = InnoDB;

## 3.14 DirectApply

**Relational Model:**

DirectApply(appID, compID, studentID, internshipStartDate, internshipEndDate)

**Functional Dependencies:**

appID -> compID, studentID,internshipStartDate, internshipEndDate

**Candidate Keys:**

{ (appID) }

**Normal Form:**

BCNF

**Table Definition:**

CREATE TABLE directApply(

appID int PRIMARY KEY,

compID int,

studentID int,

internshipStartDate date,

internshipEndDate date,

FOREIGN KEY (appID) references application (appID),

FOREIGN KEY (compID) references registeredCompany (compID),

FOREIGN KEY (studentID) references student (userID)

)ENGINE = InnoDB;

## 3.15 Announcement

**Relational Model:**

Announcement(announcementID, date, announcementType)

**Functional Dependencies:**

announcementID -> date, announcementType

**Candidate Keys:**

{ (announcementID) }

**Normal Form:**

BCNF

**Table Definition:**

CREATE TABLE announcement(

announcementID int PRIMARY KEY AUTO\_INCREMENT,

date TIMESTAMP,

announcementType char(8)

)ENGINE = InnoDB;

## 3.16 AppFeedbackAnnouncement

**Relational Model:**

AppFeedbackAnnouncement(announcementID, studentApproval, deadline, studentID, secretaryID,appID)

**Functional Dependencies:**

announcementID -> studentApproval, deadline, studentID, secretaryID, appID

**Candidate Keys:**

{ (announcementID) }

**Normal Form:**

BCNF

**Table Definition:**

CREATE TABLE appFeedbackAnnouncement (

announcementID int PRIMARY KEY,

studentApproval char(3),

deadline date,

secretaryID int,

studentID int,

appID int,

FOREIGN KEY (announcementID) references announcement (announcementID),

FOREIGN KEY (secretaryID) references secretary(userID),

FOREIGN KEY (studentID) references student(userID)

)ENGINE = InnoDB;

## 3.17 GeneralAnnouncement

**Relational Model:**

GeneralAnnouncement(announcementID, title, message, secretaryID)

**Functional Dependencies:**

announcementID -> title, message, secretaryID

**Candidate Keys:**

{ (announcementID) }

**Normal Form:**

BCNF

**Table Definition:**

CREATE TABLE generalAnnouncement (

announcementID int PRIMARY KEY,

title char(30),

message varchar(7000),

secretaryID int,

FOREIGN KEY (announcementID) references announcement(announcementID),

FOREIGN KEY (secretaryID) references secretary(userID)

)ENGINE = InnoDB;

# 4. FUNCTIONAL DEPENDENCIES AND NORMALIZATION

As it can be seen from previous section, all of our functional dependencies include left hand side as candidate key which is also superkey. As a consequent of this, all of our tables are in Boyce-Codd Normal Form and no decomposition is required.

# 5. FUNCTIONAL COMPONENTS

## 5.1 Use Cases

As indicated earlier in this report, Student Internship Tracking System has four different types of users namely students , secretaries, company staff and advisors(internship coordinators of department. In this section, specific features of the program according to user types will be specified using use case diagrams.

**Student:**

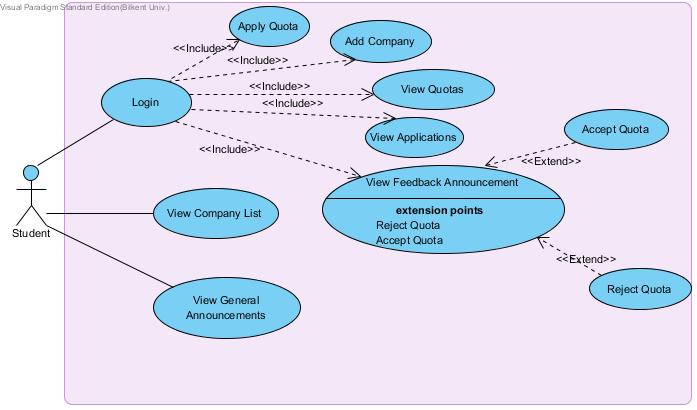
****

Figure -Use Case Diagram for Student

* Students have to login to the system for use cases specified with <<include>>.
* All students can view company list and general announcements from faculty and departments about internship processes regardless of their login operation.
* Students can view quotas of companies.
* Students can apply an existing company's quotas.
* Students can add their own internship company if it does not exist in the company list.
* Students can view their own applications.
* Students can view feedback announcements which are to specify status of their applications.
* After they view feedback announcements, students can accept or reject the quota of a specific company if they are approved. (Specified with <<extends>> in use case diagram).

**Secretary:**

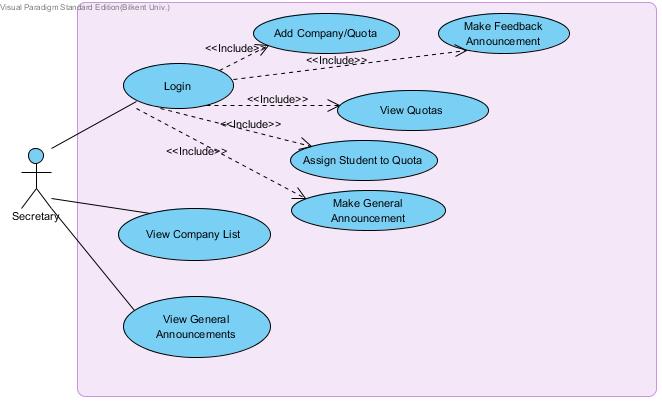
****

Figure -Use Case for Secretary

* Secretaries have to login to the system for use cases specified with <<include>>.
* All secretaries can view company list and general announcements from faculty and departments about internship processes regardless of their login operation.
* Secretaries can view quotas of companies.
* Secretaries can add company/quota pairs to the list.
* Secretaries can make feedback announcement which shows result of students' applications.
* Secretaries can make general announcements which can be seen from everybody regardless of login operation.
* Secretaries can see "open" quota and assign it to other students if the quota is rejected by a student.

**Advisor:**

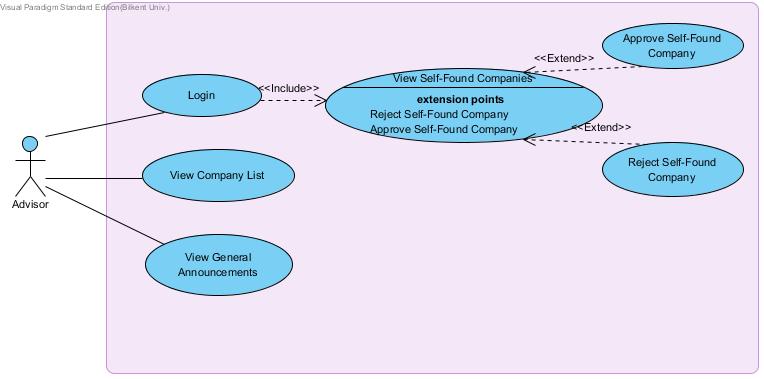


Figure -Use Case for Advisor

* Advisors who are internship coordinators of the department also have to login to the system for use cases specified with <<include>>.
* Advisors can also view company list and general announcements from faculty and departments about internship processes regardless of their login operation.
* Advisors can view self-found companies which are not existing in company list but added by students.
* Advisors can approve or reject self-found company that they viewed.

## 5.2 Scenarios

In this section, some of the most important scenarios for the Student Internship Tracking System will be specified.

**Scenario 1**

**Scenario Name:** QUOTA APPLY

**Participating Actor**: Student

**Flow of Events:**

1. Student has to login into system by using user id and password.

2. Student will click on "Companies" tab.

3. Student will see the list of existing companies and quotas.

4. By clicking apply button next to any specific company, s/he will apply the quota and will wait for result of this application.

**Scenario 2**

**Scenario Name:** ADD SELF-FOUND COMPANY

**Participating Actor:** Student

**Flow of Events:**

1. Student has to login into system by using user id and password.

2. Student will click on "Companies" tab.

3. Student will see the list of existing companies and quotas.

4. In this screen, student will click on "Add Company" button and enter information about self-found company (Company Information, address, phone and website).

5. After specification of company, student will click on "add" button and will wait for approval of this company from advisor(internship coordinator).

**Scenario 3**

**Scenario Name:** ADD COMPANY/QUOTA

**Participating Actor:** Secretary

**Flow of Events:**

1. Secretary has to login into system by using user id and password.

2. Secretary will click on "Companies" tab.

3. Secretary will see the list of existing companies and quotas.

4. In this screen, secretary will click on "Add Company/Quota" button and enter information about company (Company Information, address, phone and website as well as quota information including deadline or max number of students that will be able to apply).

5. After specification of company, secretary will click on "add" button and this company with specified quota will be added to existing company list.

**Scenario 4**

**Scenario Name:** VIEW COMPANY QUOTA

**Participating Actor:** Student

**Flow of Events:**

1. Student has to login into system by using user id and password.

2. Student will click on "Quotas" tab.

3. Student can see companies and their quota status in this screen.

**Scenario 5**

**Scenario Name:** APPROVE/REJECT SELF-FOUND COMPANIES

**Participating Actor**: Advisor (Internship Coordinator)

**Flow of Events:**

1. Advisor has to login into system by using user id and password.

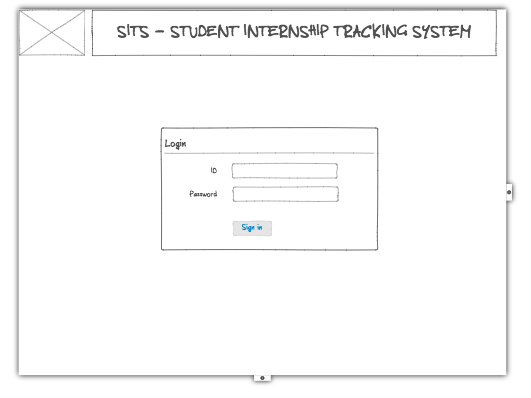
2. Advisor will click on "Companies" tab.

3. Advisor will select "Self-Found Companies" and click on "Filter" to see self-found companies only.

4. In the filtered list of "Self-Found Companies", advisor will click on either approve or reject buttons which are next to company name.

# 6. User Interface Design and Corresponding SQL Statements

## 6.1 Login



Figure

**Process:** The user enters his userID and password to sign in the Student Internship Tracking System.

**SQL Statements:**

Logging in:

SELECT userID, password

FROM Person

WHERE userID = @userID AND password = @password;

## SSD:Users:Dogancan:Desktop:cs353_git:interface mockups:My Applications - No Direct.png6.2 My Applications - Student

Figure

**Process:**

The user, in this case student, clicks on My Applications link on the navigation bar. The system lists all applications of that student. The student can cancel the quota application(s) or direct application via the action buttons.

**SQL Statements:**

Listing quota applications:

WITH

SELECT quotaID, count(\*) AS allApplications (quotaID, count)

FROM quotaApply

GROUP BY quotaID

SELECT name, city, quotaDeadline, internshipStartDate, internshipEndDate, quotaAmount - count, quotaAmount, status

FROM quotaApply NATURAL JOIN allApplications NATURAL JOIN quota NATURAL JOIN opens NATURAL JOIN company

WHERE userID = @userID;

Listing direct applications:

SELECT name, city, internshipStartDate, internshipEndDate, approval

FROM directApply NATURAL JOIN company NATURAL JOIN application

WHERE userID = @userID;

Cancelling quota application:

DELETE FROM quotaApply

WHERE appID = @appID;

Cancelling direct application:

DELETE FROM directApply

WHERE appID = @appID;

## SSD:Users:Dogancan:Desktop:cs353_git:interface mockups:Quotas.png6.3 Show Quotas - Student

Figure

**Process:**

Student clicks on Quotas link on the navigation bar. All quotas which are opened for his/her department is listed. S/he can apply for the quotas which are available to apply. Also when the student selects a specific city from dropdown menu, system displays all quotas from that selected city.

**SQL Statements:**

Listing all quotas:

SELECT name, city, quotaDeadline, internshipStartDate, internshipEndDate, quotaAmount - qcount, quotaAmount, quota.status, availableYears

FROM (SELECT quotaID as quotaID, count(\*) as qcount, compID as compID

FROM quotaApply

GROUP BY quotaID) as allAplications, quota, opens, company

WHERE allAplications.compID = quota.compID = opens.compID = company.compID AND deptName = @deptName;

Applying for an available quota:

INSERT INTO quotaApply VALUES (@quotaID, @compID, @userID, 0);

Populating dropdown list:

SELECT name

FROM company;

Filtering quotas with respect to city:

WITH

SELECT quotaID, count(\*) AS allApplications (quotaID, count)

FROM quotaApply

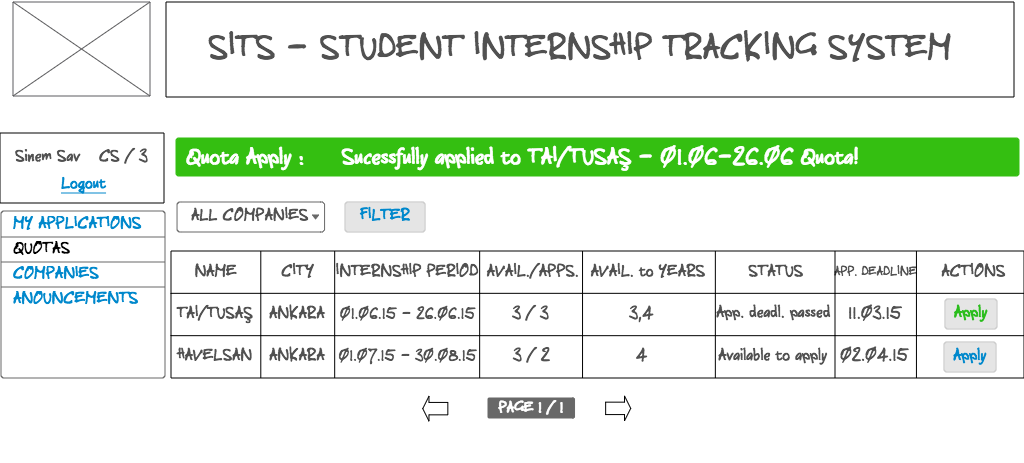
GROUP BY quotaID

SELECT name, city, quotaDeadline, internshipStartDate, internshipEndDate, quotaAmount - count, quotaAmount, status, availableYears

FROM allApplications NATURAL JOIN quota NATURAL JOIN opens NATURAL JOIN company

WHERE city = @city;

## 6.4 Quota Apply - Student



Figure

**Process:**

When student successfully applies for an available quota, a message is shown with company name and internship period.

**SQL Statements:**

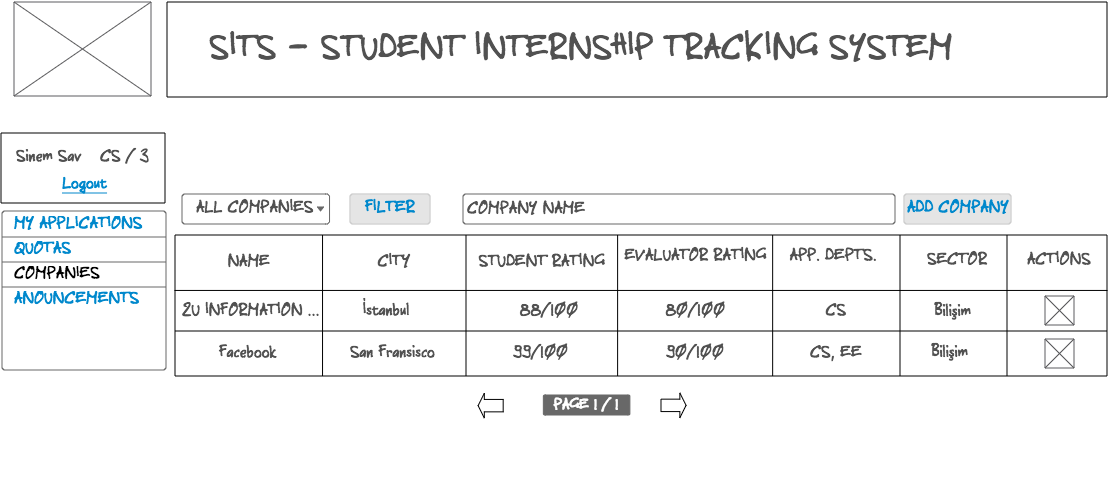
Displaying message:

SELECT name, internshipStartDate, internshipEndDate

FROM opens NATURAL JOIN quota NATURAL JOIN company

WHERE quotaID = @quotaID;

## 6.5 Show Companies - Student



Figure

**Process:**

When student clicks on Companies link on the navigation bar, system displays all companies. Student can filter the companies according to their cities. Also student can search a company with its name.

**SQL Statements:**

Displaying all companies:

SELECT name, city, studentRating, evaluatorRating, applicableDepts, sector

FROM company;

Filtering companies with their cities:

SELECT name, city, studentRating, evaluatorRating, applicableDepts, sector

FROM company

WHERE city = @city;

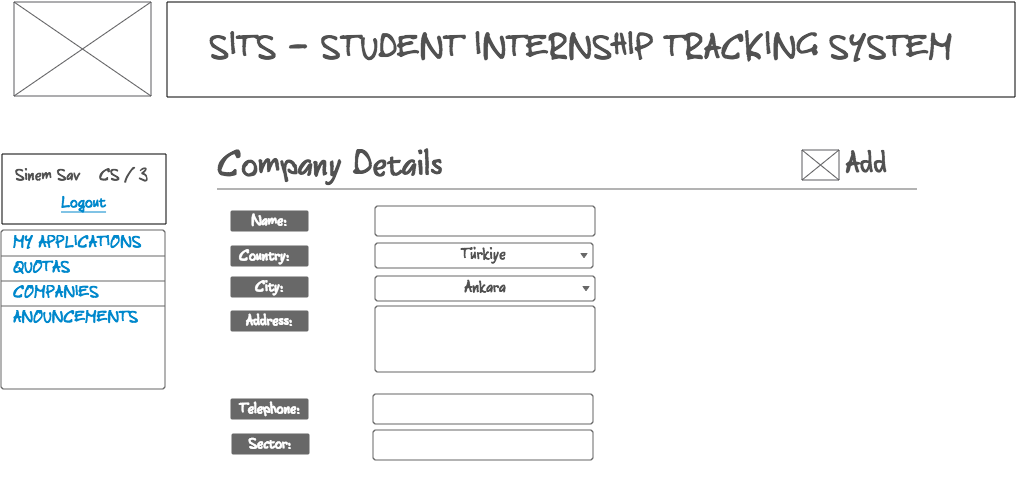
Filtering companies with company name:

SELECT name, city, studentRating, evaluatorRating, applicableDepts, sector

FROM company

WHERE name like ‘%search\_name’;

## 6.6 Add Company - Student



Figure

**Process:**

When student clicks on Add Company button in 5.4, system will display a form to add company details. NULL values will be initialized by other actors (secretary, company staff and advisor).

**SQL Statements:**

Adding company:

INSERT INTO selfFoundCompany VALUES (NULL);

//Trigger of this action is involved; a new tuple in company table is created. Then company table is updated with the rest of the form values.

UPDATE company

SET name=@name, password=@auto\_generated\_password, address=@address, phone=@telephone, status = ‘not approved’, city = @city, country = @country, sector = @sector;

## SSD:Users:Dogancan:Desktop:cs353_git:interface mockups:Direct Application.png6.7 Direct Apply - Student

Figure

**Process:**

When student clicks on action button in 5.4, the company should be applicable for that student’s department, this form is displayed. Student enters internship start and end dates.

**SQL Statements:**

Making direct apply:

INSERT INTO directApply VALUES (@compID, @userID, @internshipStartDate, @internshipEndDate);

## SSD:Users:Dogancan:Desktop:cs353_git:interface mockups:Anouncements.png6.8 Announcements - Student

Figure

**Process:**

When student clicks on Announcements button on the navigation bar, system displays current announcements for his/her department.

**SQL Statements:**

Displaying announcements:

SELECT title, message, date

FROM generalAnnouncement NATURAL JOIN announcement

WHERE deptName = (SELECT deptName

FROM student NATURAL JOIN person

WHERE userID = @userID);

## SSD:Users:Dogancan:Desktop:cs353_git:interface mockups:Approve Company - Advisor.png6.9 Approve Company – Advisor

Figure

**Process:**

When advisor clicks on Companies link on the navigation bar, all companies, registered and self-found companies, are listed. Advisor can approve/disapprove a company to the system by specifying its applicable departments.

**SQL Statements:**

Displaying companies:

SELECT name, city, studentRating, evaluatorRating, applicableDepts, sector

FROM company;

Filtering companies as Self Found Company:

SELECT name, city, studentRating, evaluatorRating, applicableDepts, sector

FROM company

WHERE advisorID is not NULL;

Filtering companies with company name:

SELECT name, city, studentRating, evaluatorRating, applicableDepts, sector

FROM company

WHERE name like ‘%search\_name’;

Initializing applicable departments of a company and approving that company:

UPDATE company

SET applicableDepts = @dropdownApplicableDepts, status = ‘approved’

WHERE compID = @compID;

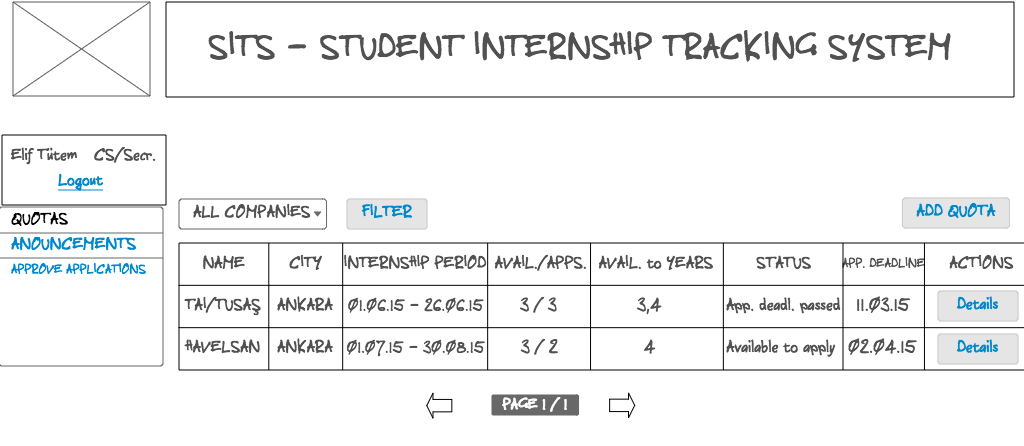
NOTE: Since the status of a company is initialized as ‘not approved’, there is no need to update that value when advisor disapproves the company.

UPDATE selfFoundCompany

SET advisorID = @userID

WHERE compID = @compID;

## 6.10 See Quotas - Secretary



Figure

**Process:**

When secretary logins to the system and clicks on Quotas link on the navigation bar, system displays all the quotas which are opened to his/her department. S/he can filter the quotas according to companies’ cities. Also s/he can add quota by simply clicking on Add Quota button on the right. Finally, s/he can see the specific quota details by clicking on Details button on each row.

**SQL Statements:**

Listing all quotas:

We use view in section 6.2.1 for this operation.

Filtering quotas with respect to city:

We use view in section 6.2.2 for this operation.

## SSD:Users:Dogancan:Desktop:cs353_git:interface mockups:Secretary - Add Quota.png6.11 Add Quota - Secretary

Figure

**Process:**

When secretary clicks on Add Quota button in 5.11, system displays respective forms to add a quota for a company.

**SQL Statements:**

Adding quota:

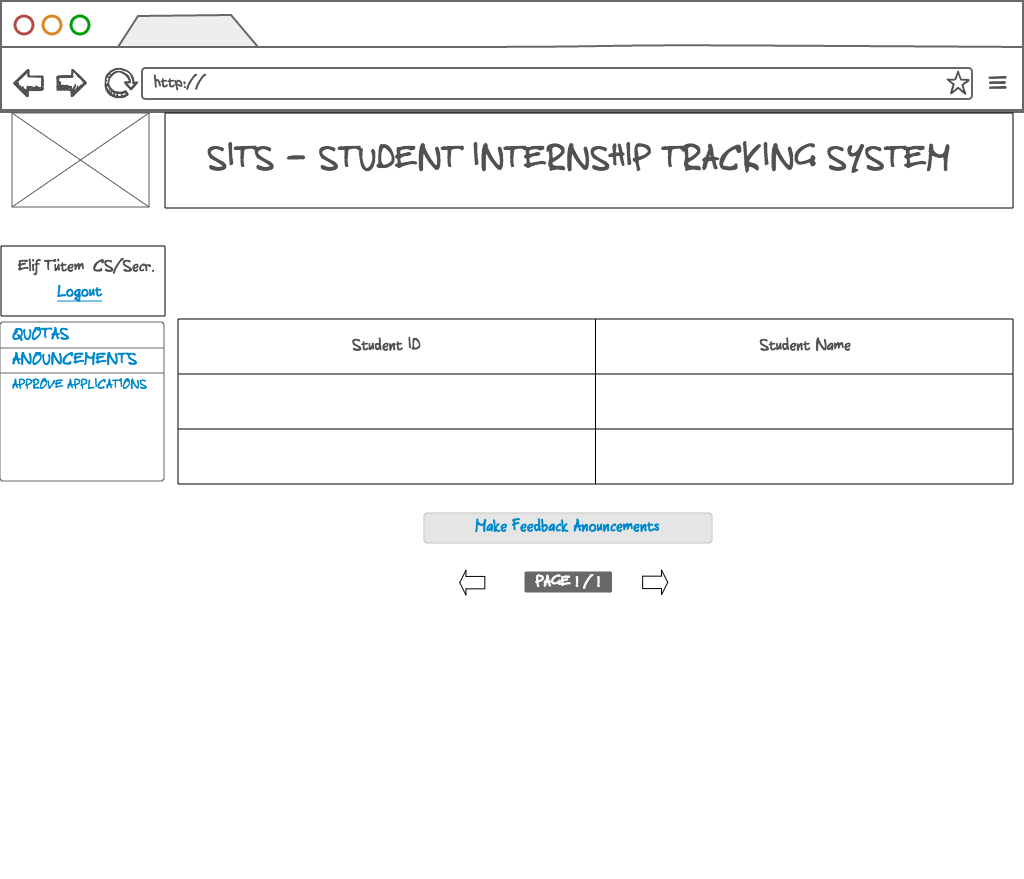
INSERT INTO opens VALUES (@compID, @deptName);

// Trigger of this action is invoked; a new tuple in quota table is created. Then quota table is updated with the rest of the form values.

UPDATE quota

SET internshipDuration = @duration, internshipStartDate = @internshipStartDate, internshipEndDate = @internshipEndDate, avaliableYears =@avaliableYears, status = @status, quotaAmount = @quotaAmount, quotaDeadline = @quotaDeadline;

## 6.12 See Detailed Quota Applications - Secretary



Figure

**Process:**

When secretary clicks on Details button in 5.11, system displays applicants of that quota.

**SQL Statements:**

Displaying applicants of a specific quota:

SELECT userID, name

FROM student NATURAL JOIN person NATURAL JOIN quotaApply

WHERE quotaID = @quotaID;

Making feedback announcements:

INSERT INTO appFeedbackAnnouncement VALUES (NULL, @auto\_deadline, @userID, @userID);

## SSD:Users:Dogancan:Desktop:cs353_git:interface mockups:Secretary Approve Applications.png6.13 Approve Applications - Secretary

Figure

**Process:**

When secretary clicks on Approve Applications button on the navigation bar, system displays all applications (quota and direct). Secretary can approve/disapprove this applications via the action buttons in each row.

**SQL Statements:**

Displaying all applications:

SELECT compID, userID, appType

FROM application NATURAL JOIN directApply NATURAL JOIN quotaApply;

Approving an application:

UPDATE application

SET approval = ‘approved’

WHERE appID = @appID;

Disapproving an application:

UPDATE application

SET approval = ‘disapproved’

WHERE appID = @appID;

## SSD:Users:Dogancan:Desktop:cs353_git:interface mockups:Secretary Anouncements.png6.14 See Announcements – Secretary

Figure

**Process:**

When secretary clicks on Announcements button on the navigation bar, system displays all announcements. Secretary can add announcement by simple clicking on Add Announcement button on the right.

**SQL Statements:**

Displaying all announcements:

SELECT title, message, date

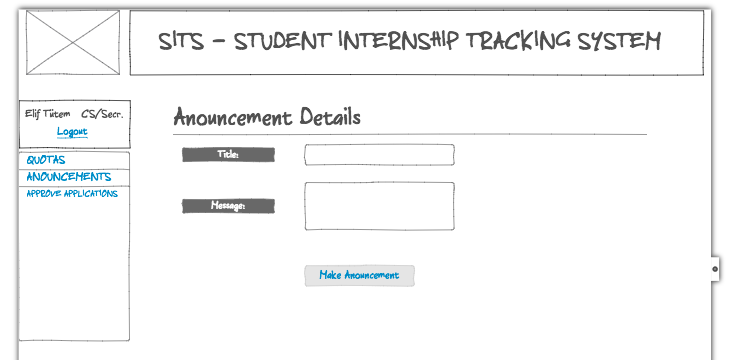
FROM generalAnnouncement NATURAL JOIN announcement

WHERE deptName = (SELECT deptName

FROM secretary NATURAL JOIN person

WHERE userID = @userID);

## 6.15 Add Announcement – Secretary



Figure

**Process:**

When secretary clicks on Add Announcement button in 5.15, system displays respective forms to get general announcement title and message.

**SQL Statements:**

Making an general announcement:

INSERT INTO generalAnnouncement VALUES (@title, @message, @userID);

# 7. ADVANCED DATABASE COMPONENTS

## 7.1 Reports

### 7.1.1 Total Number Of Registered and Self-Found Companies

WITH self\_found\_companies (comp\_id, comp\_name, numberCompanies)

AS (SELECT count(compID)

FROM Company

WHERE status = "not approved"),

WITH registered\_companies (comp\_id, comp\_name, numberCompanies)

AS (SELECT count(compID)

FROM Company

WHERE status = " approved")

SELECT S.numberCompanies, R.numberCompanies

FROM self\_found\_companies S, registered\_companies R;

### 7.1.2 Total Number of Applications

SELECT count(\*)

FROM Application;

### 7.1.3 Total Number of Approved Applications

SELECT count(appID)

FROM Application

WHERE approval = "approved";

## 7.2 Views

### 7.2.1 Secretary-Quota View

A secretary cannot view any quota which is not from his/her department (thus, cannot approve or reject these quotas). Therefore, we will use a view for this operation and use secretary id to get department:

WITH

SELECT quotaID, count(\*) AS allApplications (quotaID, count)

FROM quotaApply

GROUP BY quotaID

CREATE VIEW secretary\_quotas\_view AS

SELECT name, city, quotaDeadline, internshipStartDate, internshipEndDate, quotaAmount - count, quotaAmount, status, availableYears

FROM allApplications

WHERE deptName = @deptName;

### 7.2.2 Secretary-FilteredQuota View

This view is same as the previous one except it uses filtering operation:

WITH

SELECT quotaID, count(\*) AS allApplications (quotaID, count)

FROM quotaApply

GROUP BY quotaID

CREATE VIEW secretary\_filteredQuota\_view AS

SELECT name, city, quotaDeadline, internshipStartDate, internshipEndDate, quotaAmount - count, quotaAmount, status, availableYears

FROM allApplications NATURAL JOIN quota NATURAL JOIN opens NATURAL JOIN company

WHERE city = @city AND deptName = @deptName;

### 7.2.3 Student-Feedback Announcement View

A student cannot view any application feedback announcement which doesn't belong to them(doesn't include result of his/her application). Therefore, we will use a view for this operation and use student id to get correct feedback announcement:

CREATE VIEW student\_feedbackAnnouncement\_view AS

SELECT studentApproval, deadline

FROM AppFeedbackAnnouncement

WHERE studentID = @studentID;

## 7.3 Triggers

* When an application is added to application table, depending on the type of the application(quota or direct application), we will update and add this application to the corresponding tables by using trigger. This trigger is implemented as:

CREATE DEFINER=`root`@`localhost` TRIGGER `project`.`newApplication` AFTER INSERT ON project.application FOR EACH ROW

IF NEW.appType = "directApply" THEN

INSERT INTO directApply VALUES (NEW.appID, null, null, null, null);

ELSE

INSERT INTO quotaApply VALUES (NEW.appID, null, null, null, **0**, **0**);

END IF;

* If a student cancels an application, it is deleted from Application table and it means it should be also deleted from directApply or quotaApply tables. This process will be managed by a trigger. This trigger is implemented as:

CREATE DEFINER=`root`@`localhost` TRIGGER `project`.`delete\_application` BEFORE DELETE ON project.application FOR EACH ROW

BEGIN

IF OLD.appType = "directApply" THEN

DELETE FROM directApply WHERE appID = OLD.appID;

ELSE

DELETE FROM quotaApply WHERE appID = OLD.appID;

END IF;

END;

* If a student, secretary or advisor is added to Person table, it should be also added to the corresponding table(student, advisor or secretary table) which again will be managed by a trigger. This trigger is implemented as:

CREATE DEFINER=`root`@`localhost` TRIGGER `project`.`add\_person` AFTER INSERT ON project.person FOR EACH ROW

BEGIN

IF NEW.userType = "student" THEN

INSERT INTO student VALUES (NEW.userID, null);

ELSEIF NEW.userType = "secretary" THEN

INSERT INTO secretary VALUES (NEW.userID);

ELSEIF NEW.userType = "advisor" THEN

INSERT INTO studentAdvisor VALUES (NEW.userID);

END IF;

END;

* When a self-found company is approved, it should be added to registeredCompany table with a trigger (if it is a self-found company, its’ status will be “not approved” initially). This trigger is implemented as:

CREATE DEFINER=`root`@`localhost` TRIGGER `project`.`advisor\_approval` AFTER UPDATE ON project.company FOR EACH ROW

BEGIN

IF

NEW.status = "approved" AND OLD.status = "not approved" THEN

INSERT INTO registeredCompany VALUES (OLD.compID,DEFAULT);

END IF;

END;

* If a tuple is deleted from Person table, and it is existing in subclasses(sub-tables of Person) it should be deleted from them also. This trigger is implemented as:

CREATE DEFINER=`root`@`localhost` TRIGGER `project`.`delete person` BEFORE DELETE ON project.person FOR EACH ROW

BEGIN

IF OLD.userType = "student" THEN

DELETE FROM student WHERE userID = OLD.userID;

ELSEIF OLD.userType = "secretary" THEN

DELETE FROM secretary WHERE userID = OLD.userID;

ELSEIF OLD.userType = "advisor" THEN

DELETE FROM studentAdvisor WHERE userID = OLD.userID;

END IF;

END;

* If a draw result for a quota application is affirmative, the status of the quota should be "waiting for student approval". This process is implemented as trigger with the following code:

CREATE DEFINER=`root`@`localhost` TRIGGER `project`.`after\_draw` AFTER UPDATE ON project.quotaApply FOR EACH ROW

BEGIN

IF OLD.drawResult = **0** AND NEW.drawResult = **1** THEN

UPDATE quota SET status = "waiting for student approval"

WHERE quotaID = OLD.quotaID;

END IF;

END;

## 7.4 Constraints

* Student Internship Tracking System cannot be used without login operation except open services (View Company List and General Announcements).
* Status of any company cannot be null since it indicates whether the company is registered or self-found (waiting for approval).
* Password of any user cannot be less than 6 characters and more than 12 characters.
* Internship end date cannot be earlier than its start date.
* Student who make a quota application cannot make direct application and should cancel the quota application to make direct application (other way around is also not applicable).
* Available years of a quota cannot include years less than 1 and more than 3.
* Duration of any internship must be bigger than 20 weekdays.
* Students cannot apply to quotas which are not applicable for their department.
* Students cannot apply to quotas if deadline is passed.
* Secretaries cannot see applications of departments other than their own department.
* If the deadline of a quota application is not passed, secretaries cannot make feedback announcements for these applications (cannot announce result before the deadline).

## 7.5 Stored Procedures

* Our company list page is available for anyone regardless of their login operation. Thus, we plan to use stored procedure to get all companies.
* When a student, secretary or advisor is added we should check if they exist in Person table and if not we should first add them to the Person table. This process is same for all insertion into these tables and because of this; we will use a stored procedure.
* Similarly, when a registered or self-found company is added it should first added to Company table which will be a stored procedure.
* When a quota is rejected by a student, it will be automatically open quota and this procedure will be a stored procedure.

# 8. IMPLEMENTATION DETAILS

* For this project, MySQL is used for database management and InnoDB is used as a database engine.
* For back-end development, PHP is used.
* For front-end development, HTML and JavaScript are used and the appearance of the page is arranged with CSS.
* We created our tables from MySQL shell. However, changing columns of tables using TOAD is much easier. Therefore, for further development and arranging columns, we used TOAD for MySQL.
* In Student-Internship Tracking System, any page uses following code to connect database:

// Variables

$servername = "localhost";

$username = "root";

$password = "comodo365";

$dbname = "project";

// Create connection

$conn = mysqli\_connect($servername, $username, $password, $dbname);

// Check connection

if (!$conn) {

die("Connection failed: " . mysqli\_connect\_error());

}

* For creating SQL statements and sending to database, following code is used (as an example, login page is chosen where we get userID and password from database to check):

$query = "SELECT \* FROM person WHERE userID='$usr' AND password='$psw'";

// run the query and store result

$result = mysqli\_query($conn, $ query);

* For enforcing constraints, we used GUI and warnings. For example, any user trying to use "Login" button without entering user id or password will face a warning saying "Please Fill All Required Fields". In addition, GUI is used in a way that position of buttons, list of buttons and views to different users are enforcing constraints. For example, the constraint about open services (any user can see general announcements and list of companies without a login operation) is enforced with two buttons in the same page with login so that any user can click these on without clicking on login. Lastly, we have different GUIs for different user types which also enforces constraints. For example, GUI for student type is chosen in a way that students cannot approve any company, quota etc.(there is no button for that).

# 9. USER MANUAL

**Open Services (for all user types):**

Any user can see announcements and list of companies regardless of login operation. For these services, we have Announcements and Company List buttons shown in Figure 19. Clicking on "Announcements" will result in page in Figure 20 where the general announcements from secretaries shown and clicking on "Company List" will result in Figure 21 where the list of existing (approved) companies for the internship system is shown.

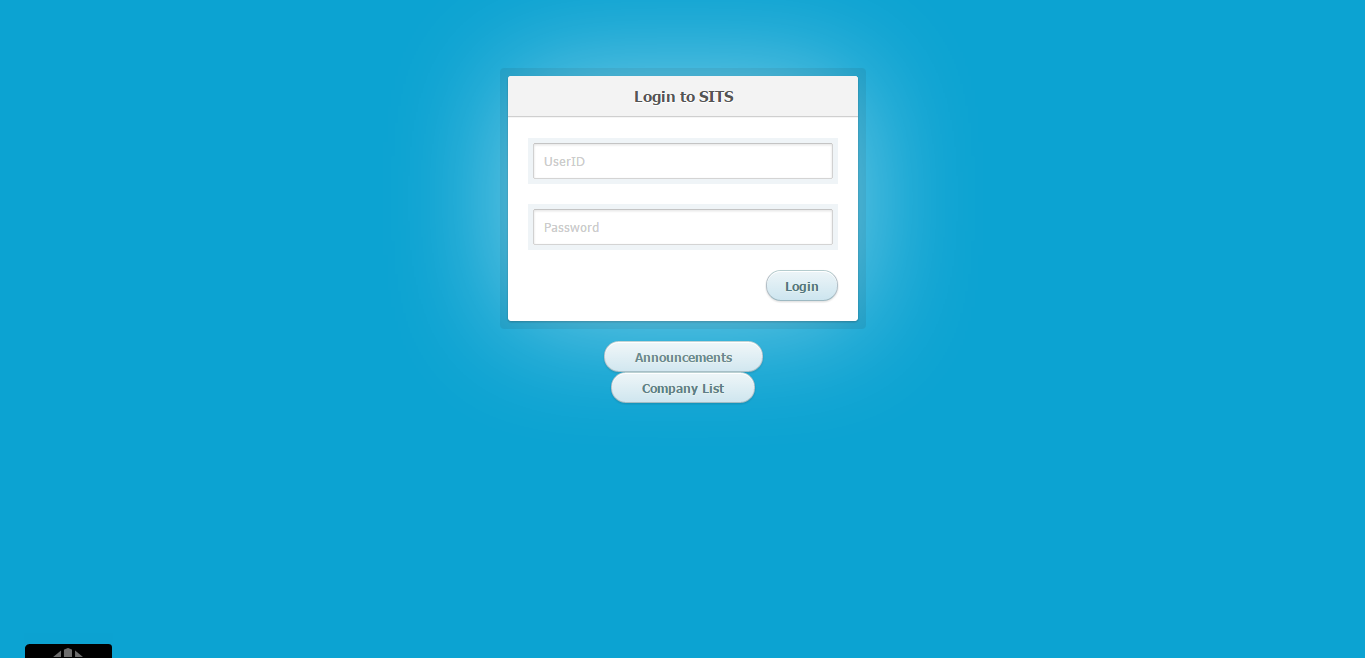


Figure - Login Page

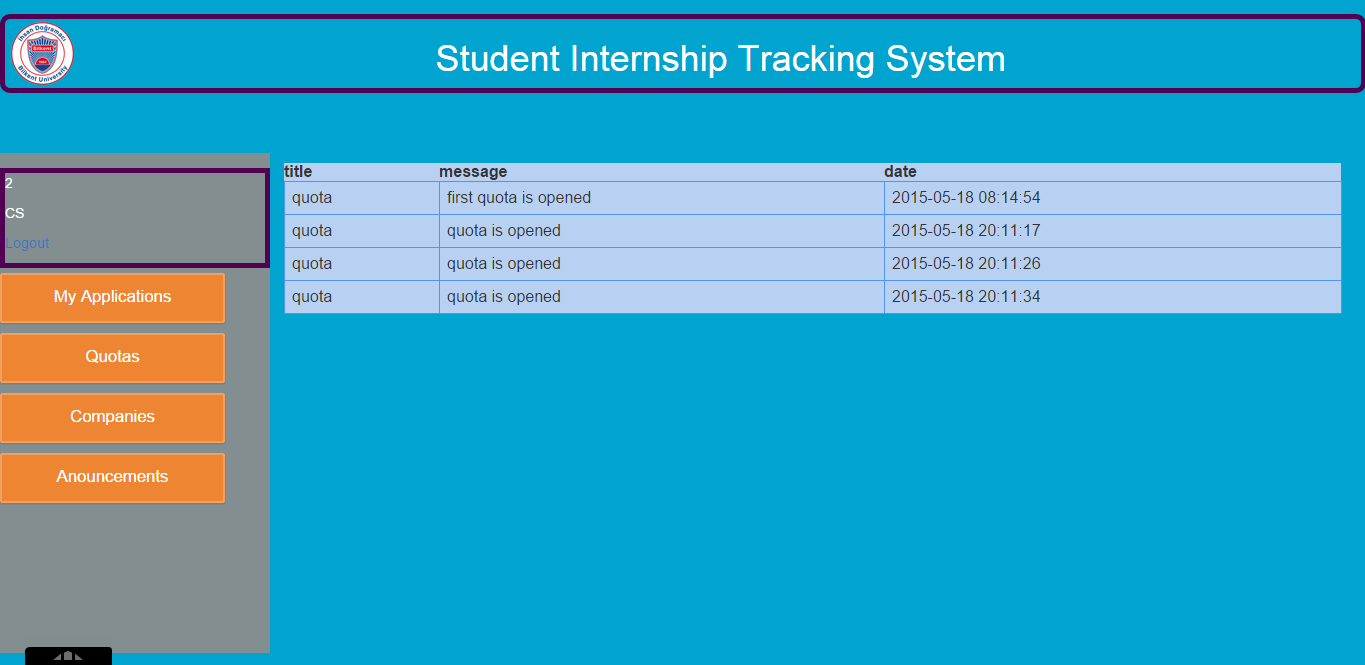


Figure - Announcements Page

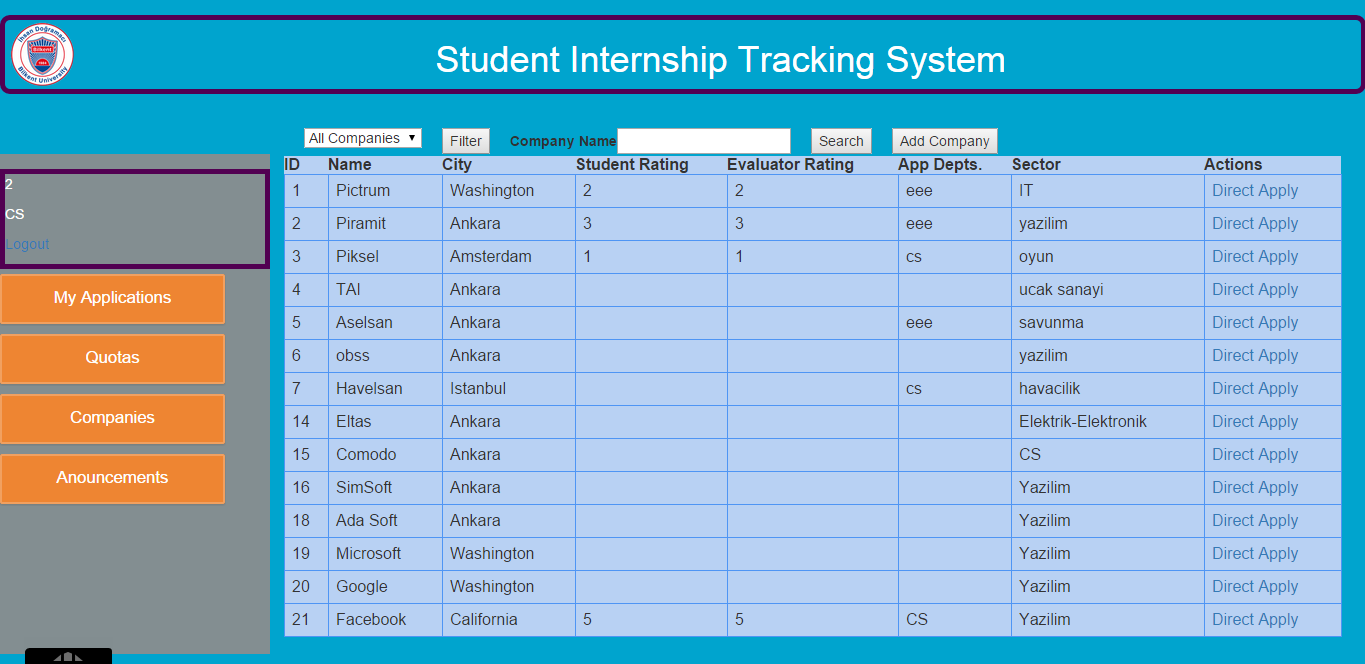


Figure - Company List Page

## 9.1 User Manual for Student

**Login Page:**

The login page for student includes User ID and Password fields for login operation as well as open service buttons ( Announcements and Company List buttons). Login Page is shown in Figure 19. Students can login to the system for further operations using their pre-determined user ID and password. By clicking one of the buttons for open services, they will see the pages previously discussed under **Open Services.**

**Main Page:**

After login operation, students will see the main page showed in Figure 22. In this page, left side shows the id and the department of the student as well as some buttons for further operations namely "My Applications", "Quotas", "Companies" and "Announcements". This page shows the list of companies as default and name, city, evaluator and student rating for that company, applicable departments (App. Depts.) and sectors. Further operations that students can do in this page are listed below:

* Direct Apply to any company by clicking on "Direct Apply" next to each listed company. This operation will result in Figure 23 where the start and end date of the internship should be entered by student. After, clicking on "Apply" button will result in question asking "Drop other applications?" to enforce only one direct application for each student. By admitting, application will be shown under My Applications which will be explained in My Applications section.

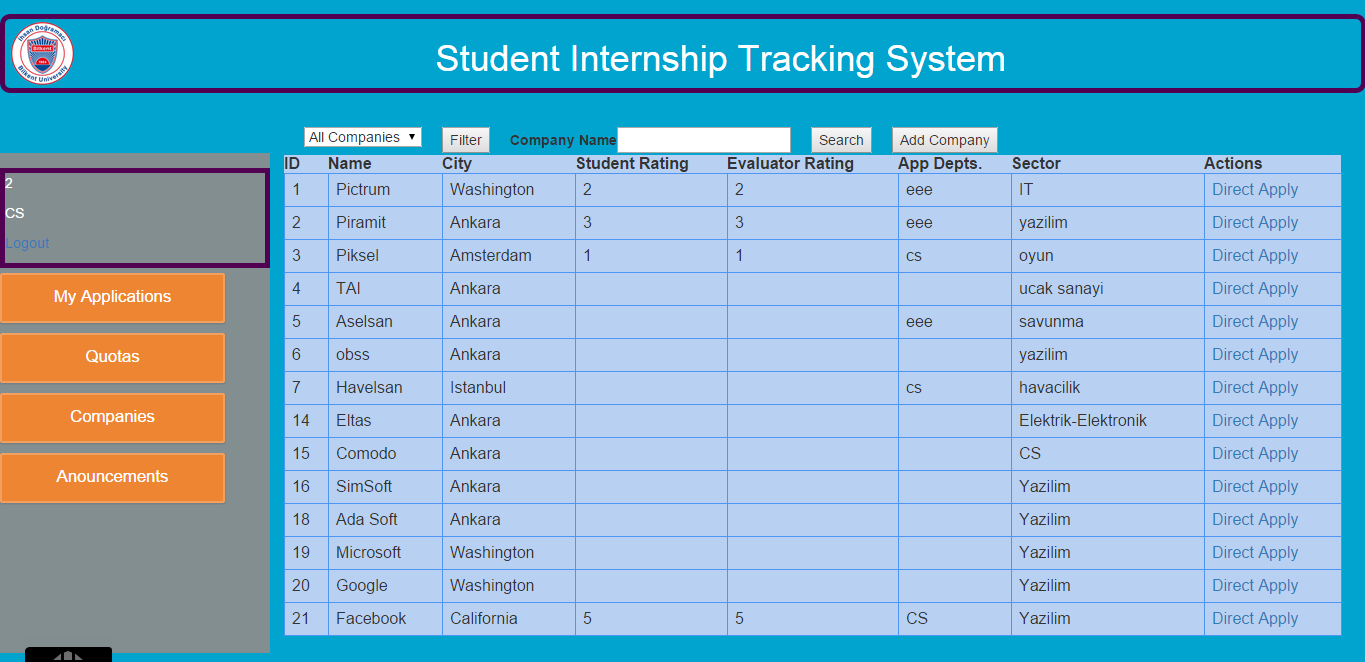


Figure - Main Page for Student

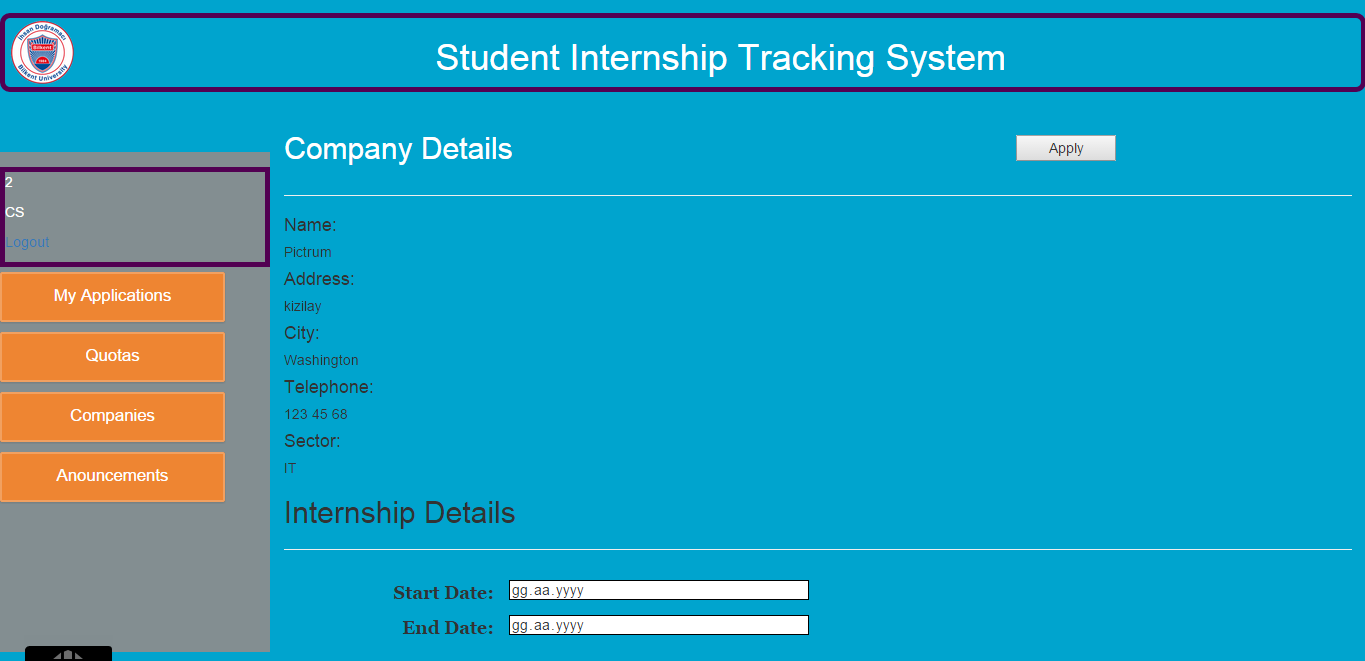


Figure - "Direct Apply" Page

* Students can filter the list of companies using filter option on the top of the page. Selecting any city from dropdown list and clicking on Filter will result in page showing companies in only that city.
* Student can also search for company and see whether it exists in the list or not. For this operation, text field on the top of the main page named with "Company Name" is used and students can type the company name and click on "Search" button for this operation.
* Students can add a new company to the system by using "Add Company" button on top of the page. This button will result page shown in Figure 24 and by filling out the necessary fields, students can add the company.

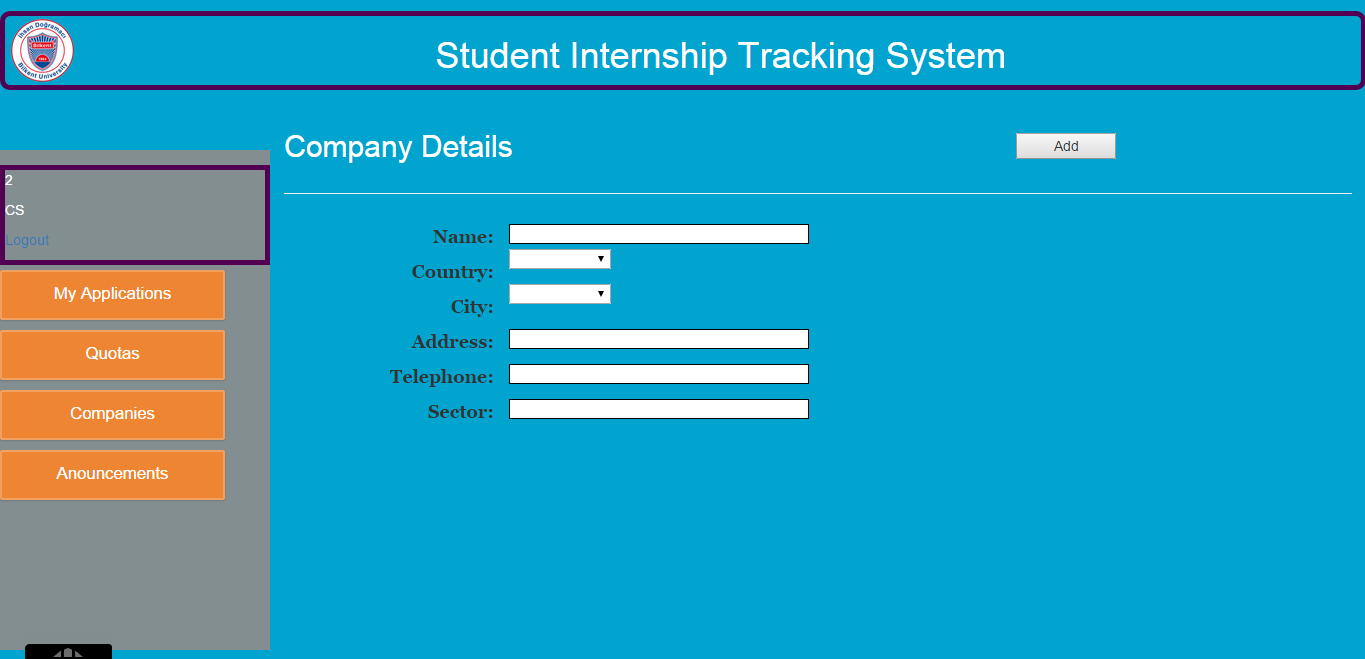


Figure - Add Company Page

**My Applications:**

This page (shown in Figure 25) shows the current applications of the student and students can see this page by clicking on "My Applications" button from any page. In this page, students can see their quota and direct applications as well as training period, total applications, quota amount and status for their applications(feedback announcements). In Figure 25, the corresponding student has one quota and one direct applications. The status for the quota application is "waiting for student approval" which shows the secretary approved the application and student should click on either Yes or No depending on his/her final approval for this internship from Actions tab. If the status is "waiting for drawal", it means there is still time for deadline and quota for the selection. On direct applications, status is either "Not Approved" or "Approved" depending on advisor approval for the new company. For the direct applications, student has opportunity to cancel application using Actions tab(shown in Figure 28).

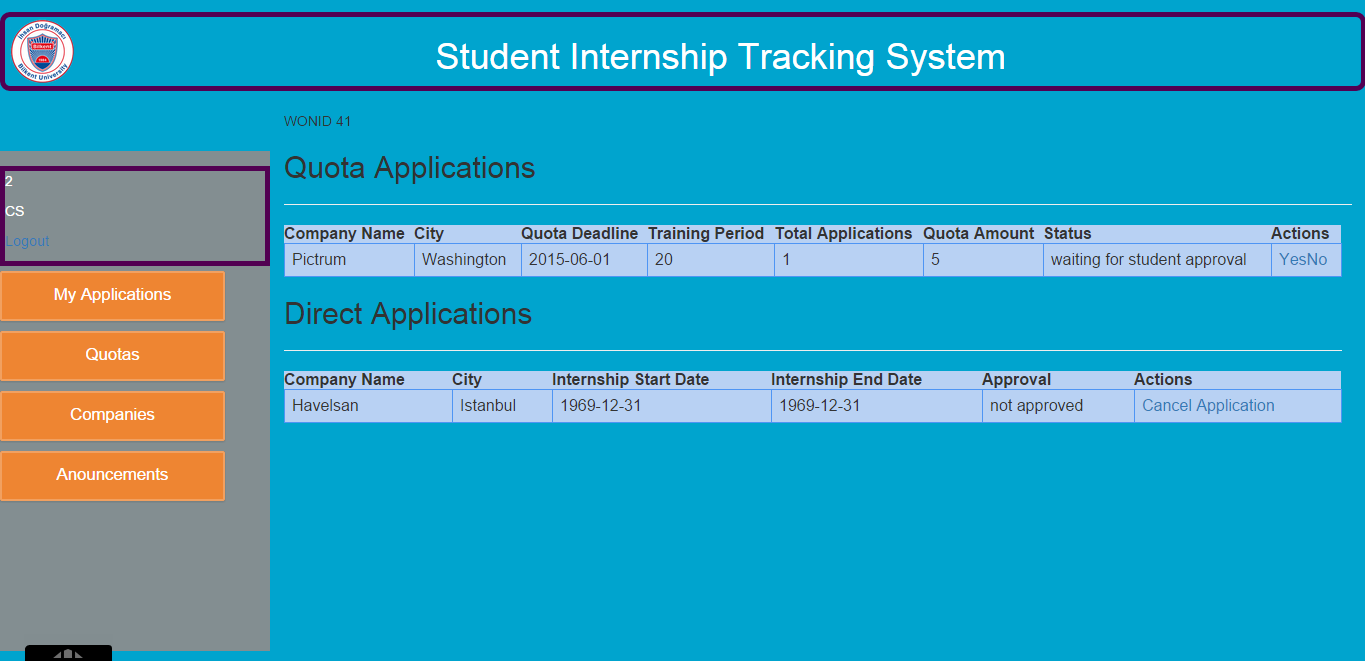


Figure - My Applications Page

**Quotas:**

This page (shown in Figure 26) shows the available quotas for quota applications and students can see this page by clicking on "Quotas" button from any page. In this page, students can see the list of companies, quota deadline, internship start and end dates, total number of applications for that company, quota amount, status and available years. As an example, in Figure 26, company called Pictrum has opened 3 different quotas for different dates and Ada Soft has opened one quota. Student in Figure 26 has one quota application and the status for this application is "waiting for student approval" (mentioned in My Applications section). As this quota application is made by the student, the actions part is Not Applicable(N/A) for this company. For other companies, students can click on "Apply" under Actions.

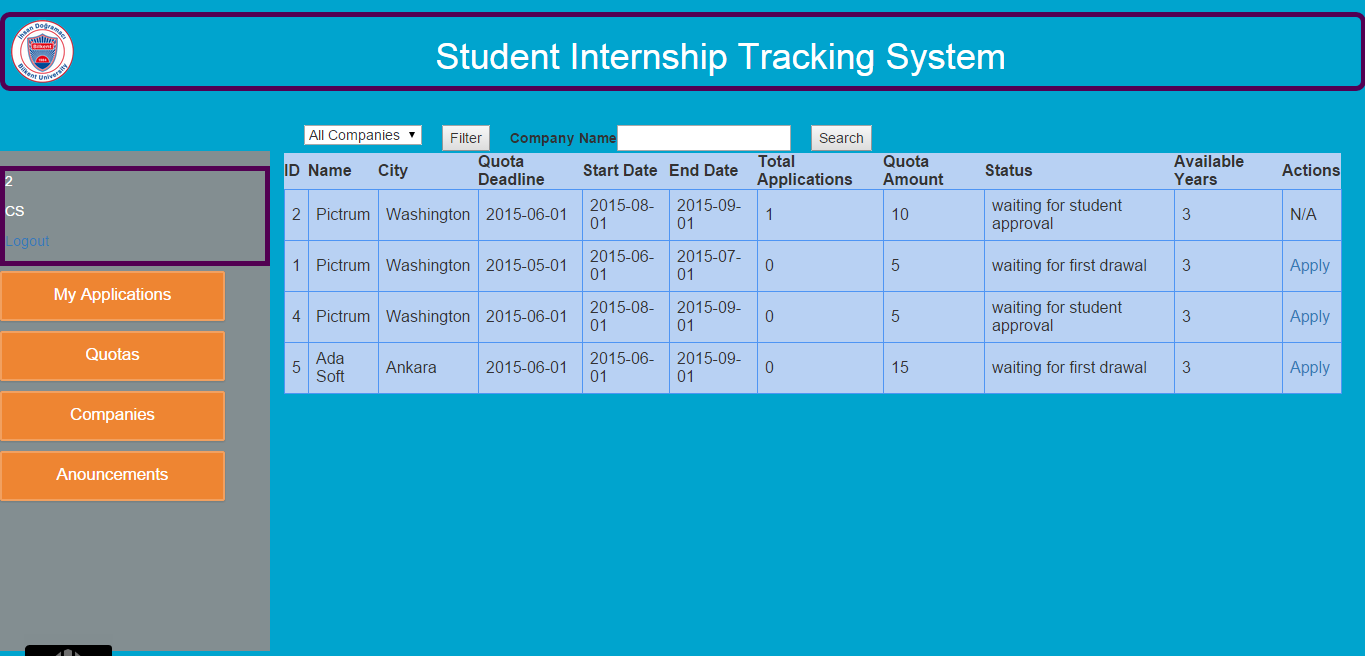


Figure - Quotas Page

**Companies:**

This page is exactly the same as the main page and shows the list of companies (See Figure 22).

**Announcements:**

This page is exactly same as the announcements page under open services (See Figure 20).