Bilkent University Department of Computer Engineering

CS353 Term Project

Student Registration System

Design Report

Doğukan KÖSE, Musab OKŞAŞ, Serkan DELİL, Mustafa Çağrı GÜNGÖR

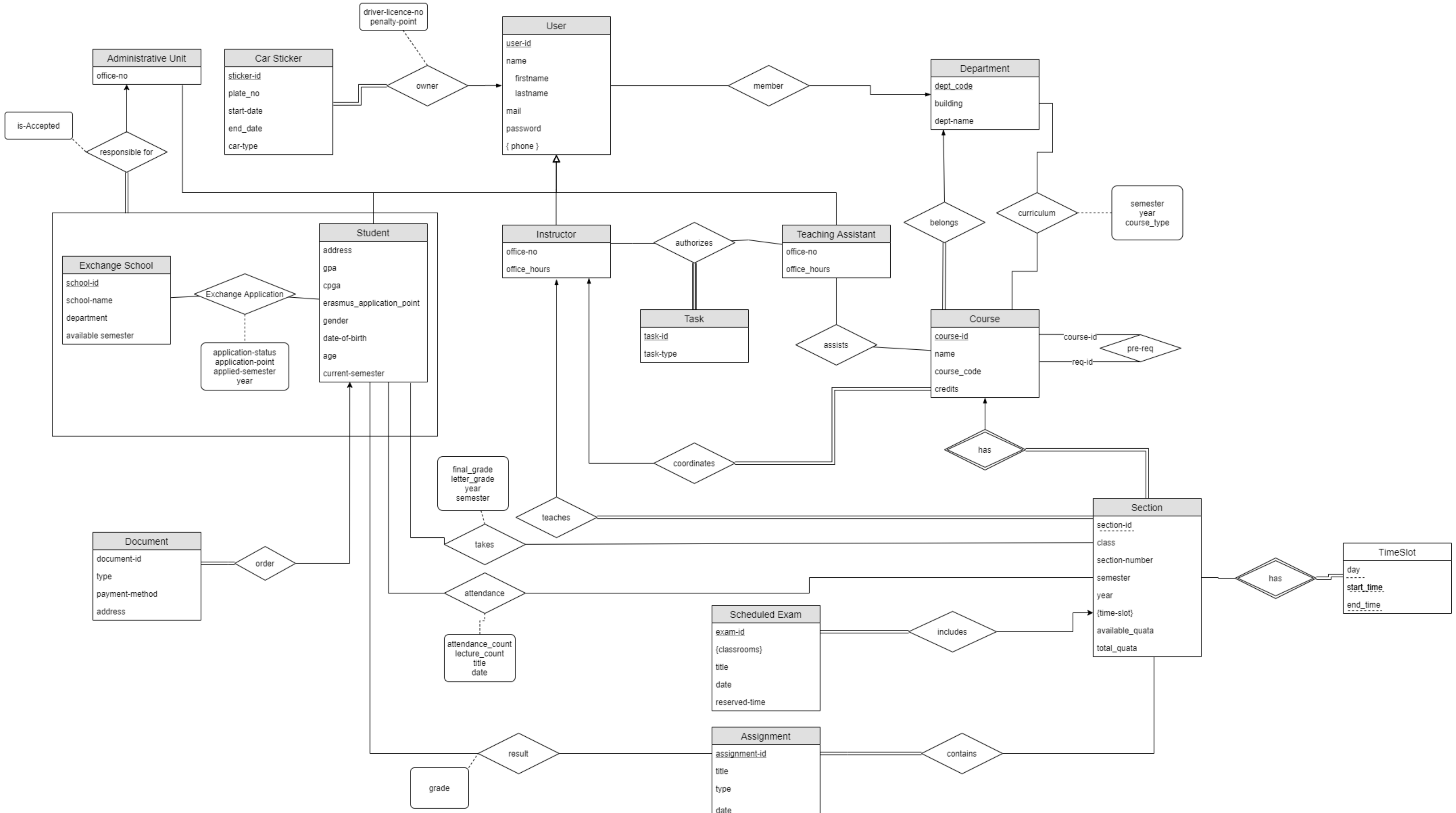
İçindekiler Tablosu

1. Revised E/R Model

According to the feedback of assistant's from project proposal and also during the design process the following changes are made in order to have a better structure for our database structure:

- A Scheduled Exam entity is added which has the attributes of exam-id, classroom-names, title, reserved_time and date.
- The "includes" one to many relations is added between Scheduled Exam and Section entities.
- final_grade and letter_grade attributes added to the "takes" relation between student and section.
- Assignment entity is added which has the attributes of assignment-id, date, title, average and type.
- The "contains" one to many relations is added between Assignment and Section entities.
- The "result" many to many relations is added between Assignment and Student entities, the relation has attributes of grade, assignment-id, user-id.
- Some participation types are changed for different entity-relation couples which are Car Sticker - owner (total participation-many), Section - teaches (total participation), Course - coordinates (total participation), Course - belong (total participation), Instructor - authorizes(many).
- Some attribute demonstrations are changed to use the same style of E/R model with the instructor.
- Exchange result entity is removed and Administrative Unit entity is added to manage the exchange process.
- The "responsible for" one to many relations is added between Administrative Unit and exchange process with the attribute of is-Accepted.
- The Document entity is added which has attributes of document-id, type, payment-method, address.
- The "order" one to many relations is added between Student and Document entities.
- TimeSlot table is added to the Section table in order to keep track of time slots of lectures.
- Offic_hours attributes are added to both Instructor and Teaching Assistant tables.
- Penalty_point attribute removed from Car_Sticker table and added to Owner relation, because, penalty points are specific for car stickers' user.

REVISED ER DIAGRAM



2. Relational Schemas

2.1 User

Relational Model:

User (<u>user_id</u>, firstname, lastname, mail, password)

Functional Dependencies:

user_id → firstname, lastname, mail, password
mail → user_id, firstname, lastname, password

Candidate Keys:

{(user_id), (mail)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE User(
        user_id         INT PRIMARY KEY AUTO_INCREMENT,
        firstname       VARCHAR(16) NOT NULL,
        lastname        VARCHAR(16) NOT NULL,
        mail            VARCHAR(32) NOT NULL UNIQUE,
        password        VARCHAR(16) NOT NULL);
```

2.2 Member

Relational Model:

Member (user_id, dept_code)

Functional Dependencies:

user_id → dept_code

Candidate Keys:

{(user_id)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE Member(
        user_id                 INT PRIMARY KEY,
        dept_code               VARCHAR(8) NOT NULL,
        FOREIGN KEY (user_id) REFERENCES User(user_id),
        FOREIGN KEY (dept_code) REFERENCES  Department(dept_code));
```

2.3 Phone

Relational Model:

Phone (<u>phone_number</u>,phone_id)

Functional Dependencies:

phone_number → phone_id

Candidate Keys:

{(phone_number)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE Phone(
        phone_number         VARCHAR(32) PRIMARY KEY,
        phone_id             INT NOT NULL,
        FOREIGN KEY (phone_id) REFERENCES User(user_id));
```

2.4 Car_Sticker

Relational Model:

Car_Sticker (sticker_id, plate_no, start_date, end_date, car_type, owner_id)

Functional Dependencies:

sticker_id → start_date, end_date, owner_id, plate_no, car_type
plate_no → start_date, end_date, owner_id, plate_no, car_type

Candidate Keys:

{(sticker_id), (plate_no)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE Car_Sticker(
        sticker_id              INT PRIMARY KEY,
        plate_no                VARCHAR(10) NOT NULL UNIQUE,
        start_date              DATE NOT NULL,
        end_date                DATE NOT NULL,
        owner_id                INT NOT NULL,
        car_type                VARCHAR(32) NOT NULL,
        FOREIGN KEY (owner_id) REFERENCES Owner(owner_id));
```

2.5 Student

Relational Model:

Student (student_id, address, gpa, cpga, erasmus_application_point, gender, date_of_birth, age, current_semester)

Functional Dependencies:

student_id → address, gpa, cpga, gender, date_of_birth, age, current_semester, erasmus_application_point

Candidate Keys:

{(student_id)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE Student (
        student_id              INT PRIMARY KEY,
        address                 VARCHAR(64) NOT NULL,
        gpa                     NUMERIC(1,2),
        cgpa                    NUMERIC(1,2),
        erasmus_application_point NUMERIC(3,2),
        gender                  ENUM('Male', 'Female') NOT NULL,
        date_of_birth           DATE,
        age                     TINYINT,
        current_semester        TINYINT,
        FOREIGN KEY (student_id) REFERENCES User(user_id)
        CHECK (gender IN ('Male', 'Female'));
```

2.6 Instructor

Relational Model:

Instructor(instructor_id, office_no, office_hours)

Functional Dependencies:

instructor_id → office_no, office_hours

Candidate Keys:

{(instructor_id)}

Normal Form:

3NF

Table Definition:
```
CREATE TABLE Instructor(
        instructor_id          INT PRIMARY KEY,
        office_no              VARCHAR(8),
        office_hours           VARCHAR(16),
        FOREIGN KEY (instructor_id) REFERENCES User(user_id));
```

2.7 Teaching Assistant

Relational Model:

TeachingAssistant(ta_id, office_no, office_hours)

Functional Dependencies:

ta_id → office_no, office_hours

Candidate Keys:

{(ta_id)}

Normal Form:

3NF

Table Definition:
```
CREATE TABLE TeachingAssistant(
        ta_id           INT PRIMARY KEY,
        office_no       VARCHAR(8),
        office_hours    VARCHAR(16),
        FOREIGN KEY (ta_id) REFERENCES User(user_id));
```

2.8 Task

Relational Model:

Task(task_id, task_type, description)

Functional Dependencies:

task_id → task_type, description

Candidate Keys:

{(task_id)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE Task(
        task_id         INT PRIMARY KEY,
        task_type               VARCHAR(32),
        description             VARCHAR(256));
```

2.9 Department

Relational Model:

Department(dept_code, dept_name, building)

Functional Dependencies:

dept_name → building, dept_code
dept_code→ building, dept_name

Candidate Keys:

{(dept_name), (dept_code)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE Department(
        dept_code               VARCHAR(8) PRIMARY KEY,
        dept_name               VARCHAR(32) NOT NULL UNIQUE,
        building                VARCHAR(16));
```

2.10 Course

Relational Model:

Course(<u>course_id</u>, name, credits, course_code, dept_code, coordinator_id)

Functional Dependencies:

course_id → name, credits, dept_code, coordinator_id, course_code
course_code → course_id, name
name → course_code

Candidate Keys:

{(course_id), (course_code), (name)}

Normal Form:

3NF

Table Definition:
```
CREATE TABLE Course(
        course_id           INT PRIMARY KEY AUTO INCREMENT,
        course_code         VARCHAR(8) NOT NULL UNIQUE,
        name                VARCHAR(32) NOT NULL UNIQUE,
        credits             INT NOT NULL,
        dept_code           VARCHAR(8) NOT NULL,
        coordinator_id      INT NOT NULL,
        FOREIGN KEY (coordinator_id) REFERENCES Instructor(instructor_id),
        FOREIGN KEY (dept_code) REFERENCES Department(dept_code));
```

2.11 Section

Relational Model:

Section(course_id, section_id, class, section_number, semester, year, teacher_id, available_quata, total_quata)

Functional Dependencies:

course_id, section_id → class, section_number, semester, year, teacher_id, available_quata, total_quata

Candidate Keys:

{(course_id, section_id)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE Section(
        course_id           INT,
        section_id          INT,
        class               VARCHAR(8) NOT NULL,
        section_number      TINYINT NOT NULL,
        semester            ENUM('fall','spring', 'summer') NOT NULL,
        year                NUMERIC(4,0) NOT NULL,
        available_quata     TINYINT NOT NULL,
        total_quata         TINYINT NOT NULL,
        teacher_id          INT NOT NULL,
        PRIMARY KEY (course_id, section_id)
        FOREIGN KEY (teacher_id) REFERENCES Instructor(instructor_id),
        FOREIGN KEY (course_id) REFERENCES Course(course_id)
        CHECK (semester IN ('fall','spring', 'summer')));
```

2.12 Exchange School

Relational Model:

ExchangeSchool(school_id, school_name, department, available_semester)

Functional Dependencies:

school_id → department, available_semester, school_name

Candidate Keys:

{(school_id)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE ExchangeSchool(
        school_id                  INT PRIMARY KEY,
        school_name                VARCHAR(32),
        department                 VARCHAR(32),
        available_semester         ENUM('fall','spring', 'summer') NOT NULL,);
```

2.13 Assists

Relational Model:

Assists(ta_id, course_id)

Functional Dependencies:

Not exist

Candidate Keys:

{(ta_id, course_id)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE Assists(
        ta_id           INT,
        course_id       INT,
        PRIMARY KEY (ta_id, course_id),
        FOREIGN KEY (ta_id) REFERENCES TeachingAssistant(ta_id),
        FOREIGN KEY (course_id) REFERENCES Course(course_id));
```

2.14 Authorizes

Relational Model:

Authorizes(task_id, instructor_id, ta_id)

Functional Dependencies:

task_id → instructor_id, ta_id

Candidate Keys:

{(task_id)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE Authorizes(
          task_id            INT PRIMARY KEY,
          instructor_id               INT,
          ta_id                       INT,
          FOREIGN KEY (task_id)                  REFERENCES Task,
          FOREIGN KEY (instructor_id)  REFERENCES Instructor,
          FOREIGN KEY (ta_id)           REFERENCES TeachingAssistant);
```

2.15 Takes

Relational Model:

Takes(student_id, course_id, section_id, attendance, final_grade, letter_grade, year, semester)

Functional Dependencies:

student_id, course_id, section_id → attendance, letter_grade, final_grade, year, semester

Candidate Keys:

{(student_id, course_id, section_id)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE Takes(
        student_id      INT,
        course_id       INT,
        section_id      INT,
        final_grade     NUMERIC(3,2),
        letter_grade    ENUM('A+','A','A-','B+','B','B-','C+','C','C-','D+','D','D-','F','FZ', 'W'),
        year            SMALLINT,
        semester        ENUM('fall', 'spring', 'summer'),
        PRIMARY KEY (student_id, course_id, section_id),
        FOREIGN KEY (student_id) REFERENCES Student(student_id),
        FOREIGN KEY (course_id) REFERENCES Section(course_id),
        FOREIGN KEY (section_id) REFERENCES Section(section_id));
```

2.16 Exchange_Application

Relational Model:

ExchangeApplication(student_id, school_id, application_status, application_point, applied_semester, year)

Functional Dependencies:

student_id, school_id → application_status, applied_semester, year, application_point

Candidate Keys:

{(student_id, school_id)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE ExchangeApplication(
        student_id              INT,
        school_id               INT,
        application_status      VARCHAR(12),
        applied_semester        ENUM('fall', 'spring', 'summer'),
        application_point       NUMERIC(3,2),
        year                    SMALLINT,
        PRIMARY KEY (student_id, school_id),
        FOREIGN KEY (student_id) REFERENCES Student(student_id),
        FOREIGN KEY (school_id) REFERENCES ExchangeSchool(school_id));
```

2.17 Scheduled Exam

Relational Model:

ScheduledExam(exam_id, date, title, reserved_time, course_id, section_id)

Functional Dependencies:

exam_id → title, date, course_id, section_id, reserved_time

Candidate Keys:

{(exam_id)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE ScheduledExam(
        exam_id         INT PRIMARY KEY AUTO INCREMENT,
        course_id       INT NOT NULL,
        section_id      INT NOT NULL,
        classroom       VARCHAR(16) NOT NULL,
        date            DATETIME NOT NULL,
        FOREIGN KEY (course_id) REFERENCES Section(course_id),
        FOREIGN KEY (section_id) REFERENCES Section(section_id));
```

2.18 Assignment

Relational Model:

Assignment(assignment_id, title, type, date, average, course_id, section_id)

Functional Dependencies:

assignment_id → title, date, type, average, course_id, section_id

Candidate Keys:

{(assignment_id)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE Assignment(
        assignment_id       INT PRIMARY KEY AUTO INCREMENT,
        course_id           INT NOT NULL,
        section_id          INT NOT NULL,
        title               VARCHAR(16) NOT NULL,
        date                DATETIME,
        type                VARCHAR(16) NOT NULL,
        average             NUMERIC(3,2),
        FOREIGN KEY (course_id) REFERENCES Section(course_id),
        FOREIGN KEY (section_id) REFERENCES Section(section_id));
```

2.19 Result

Relational Model:

Result(<u>student_id</u>, <u>assignment_id</u>, grade)

Functional Dependencies:

student_id, assignment_id → grade

Candidate Keys:

{(student_id, assignment_id)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE Result(
        student_id              INT,
        assignment_id INT,
        grade                   NUMERIC(3,2),
        PRIMARY KEY (student_id, assignment_id),
        FOREIGN KEY (student_id) REFERENCES Student(student_id),
        FOREING KEY (assignment_id) REFERENCES Assignment(assignment_id));
```

2.20 Classroom

Relational Model:

Classroom(exam_id, classroom_name)

Functional Dependencies:

exam_id→ classroom_name

Candidate Keys:

{(exam_id)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE ScheduledExam(
	exam_id			INT PRIMARY KEY,
	classroom_name		VARCHAR(8),
	FOREIGN KEY (exam_id) REFERENCES ScheduledExam(exam_id));
```

2.21 PreReq

Relational Model:

PreReq(course_id, req_id)

Functional Dependencies:

Not exist

Candidate Keys:

{(course_id, req_id)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE PreReq(
        course_id       INT,
        req_id          INT,
        PRIMARY KEY (course_id, req_id),
        FOREIGN KEY (course_id)      REFERENCES Course(course_id),
        FOREIGN KEY (req_id)                 REFERENCES Course(course_id));
```

2.22 Curriculum

Relational Model:

Curriculum(dept_code, course_id, course_type, semester, year)

Functional Dependencies:

dept_code, course_id → course_type, semester, year

Candidate Keys:

{(dept_code, course_id)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE Curriculum(
        dept_code       VARCHAR(8),
        course_id       INT,
        course_type     ENUM('must','elective','additional')
        semester        ENUM('spring','fall','summer')
        year            SMALLINT,
        FOREIGN KEY (dept_code) REFERENCES Department(dept_code),
        FOREIGN KEY (course_id) REFERENCES Course(course_id));
```

2.23 Administrative Unit

Relational Model:

AdministrativeUnit(admin_id, office_no)

Functional Dependencies:

admin_id → office_no

Candidate Keys:

{(admin_id)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE AdministrativeUnit(
	admin_id	INT PRIMARY KEY,
	office_no	VARCHAR(8),
	FOREIGN KEY (admin_id) REFERENCES User(user_id));
```

2.24 ResponsibleFor

Relational Model:

ResponsibleFor(admin_id, student_id, school_id, isAccepted)

Functional Dependencies:

admin_id, student_id, school_id → isAccepted

Candidate Keys:

{(admin_id, student_id, school_id)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE ResponsibleFor(
        admin_id        INT,
        student_id      INT,
        school_id       INT,
        PRIMARY KEY (admin_id, student_id,school_id),
        FOREIGN KEY (admin_id)       REFERENCES AdministrativeUnit(admin_id),
        FOREIGN KEY (student_id)      REFERENCES Student(student_id));
        FOREIGN KEY (school_id)      REFERENCES ExchangeSchool(school_id));
```

2.25 Document

Relational Model:

Document(document_id, type, payment_method, address)

Functional Dependencies:

document_id → type, payment_method, address

Candidate Keys:

{(document_id)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE Document(
        document_id          INT PRIMARY KEY,
        type                VARCHAR(8) NOT NULL,
        payment_method       VARCHAR(16) NOT NULL,
        address              VARCHAR(64) NOT NULL);
```

2.26 Order

Relational Model:

Order(student_id, document_id)

Functional Dependencies:

Not exist

Candidate Keys:

{(student_id, document_id)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE Order(
        student_id              INT,
        document_id             INT,
        PRIMARY KEY (student_id, document_id),
        FOREIGN KEY (student_id)     REFERENCES Student(student_id),
        FOREIGN KEY (document_id)  REFERENCES Document(document_id));
```

2.27 TimeSlot

Relational Model:

TimeSlot(<u>course_id</u>, <u>section_id</u>, <u>day</u>, <u>start_time</u>, <u>end_time</u>)

Functional Dependencies:

Not exist

Candidate Keys:

{(course_id, section_id, day, start_time, end_time)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE TimeSlot(
        course_id       INT,
        section_id      INT,
        day             ENUM('mon','tue','wed','thu','fri','sat','sun'),
        start_time      TIME,
        end_time        TIME,
        PRIMARY KEY (course_id, section_id, day, start_time, end_time),
        FOREIGN KEY (course_id) REFERENCES Section(course_id)
        FOREIGN KEY (section_id) REFERENCES Section(section_id));
```

2.28 Owner

Relational Model:

Owner(owner_id, driver_licence_no, penalty_point);

Functional Dependencies:

owner_id→ driver_licence_no, penalty_point

Candidate Keys:

{(owner_id)}

Normal Form:

3NF

Table Definition:

```
CREATE TABLE Owner(
        owner_id                INT PRIMARY KEY,
        driver_licence_no       INT NOT NULL,
        penalty_point           TINYINT,
        FOREIGN KEY (owner_id) REFERENCES User(user_id));
```

2.29 Attendance

Relational Model:

Attendance(student_id, course_id, section_id, title, date, attendance_count, lecture_count)

Functional Dependencies:

student_id, course_id, section_id → title, date, attendance_count, lecture_count

Candidate Keys:

{(student_id, course_id, section_id)}

Normal Form:

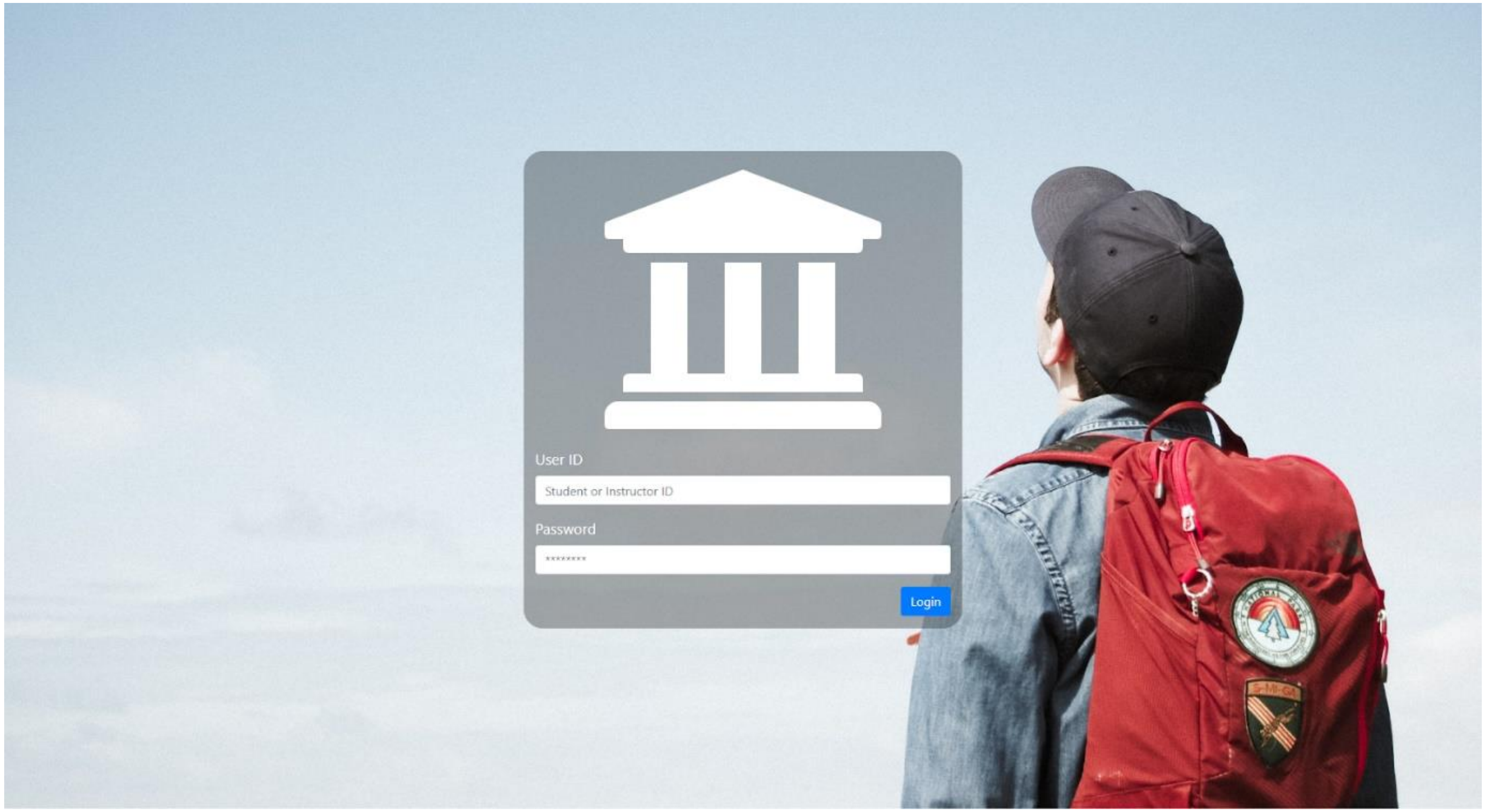
3NF

Table Definition:

```
CREATE TABLE Attendance(
        student_id          INT,
        course_id           INT,
        section_id          INT,
        date                DATE,
        title               VARCHAR(12),
        attendance_count    TINYINT,
        lecture_count TINYINT,
        PRIMARY KEY (student_id, course_id, section_id),
        FOREIGN KEY (student_id) REFERENCES Student(student_id),
        FOREIGN KEY (course_id) REFERENCES Section(course_id),
        FOREIGN KEY (section_id) REFERENCES Section(section_id));
```

3. User Interface & Corresponding SQL Statements

3.1 Login Page



Inputs: @Id, @Password

Process: Users can login to SRS by typing their id and password information then Authentication service checks the typed information. Authentication service first looks at the Instructor table; if the typed id and password information matches with any tuple from the Instructor table then the system redirects the Instructor into the Instructor page. If the system cannot find any matched tuples from that table then it looks at the TeachingAssistant table; if the typed id and password information matches with any tuple then the system redirects the home page of the teaching assistant. Lastly, the authentication service looks at the Student table and if the typed id and password information matches with any tuple system redirect the student to the Student profile page. Otherwise, the Authentication system does not let people login to SRS.

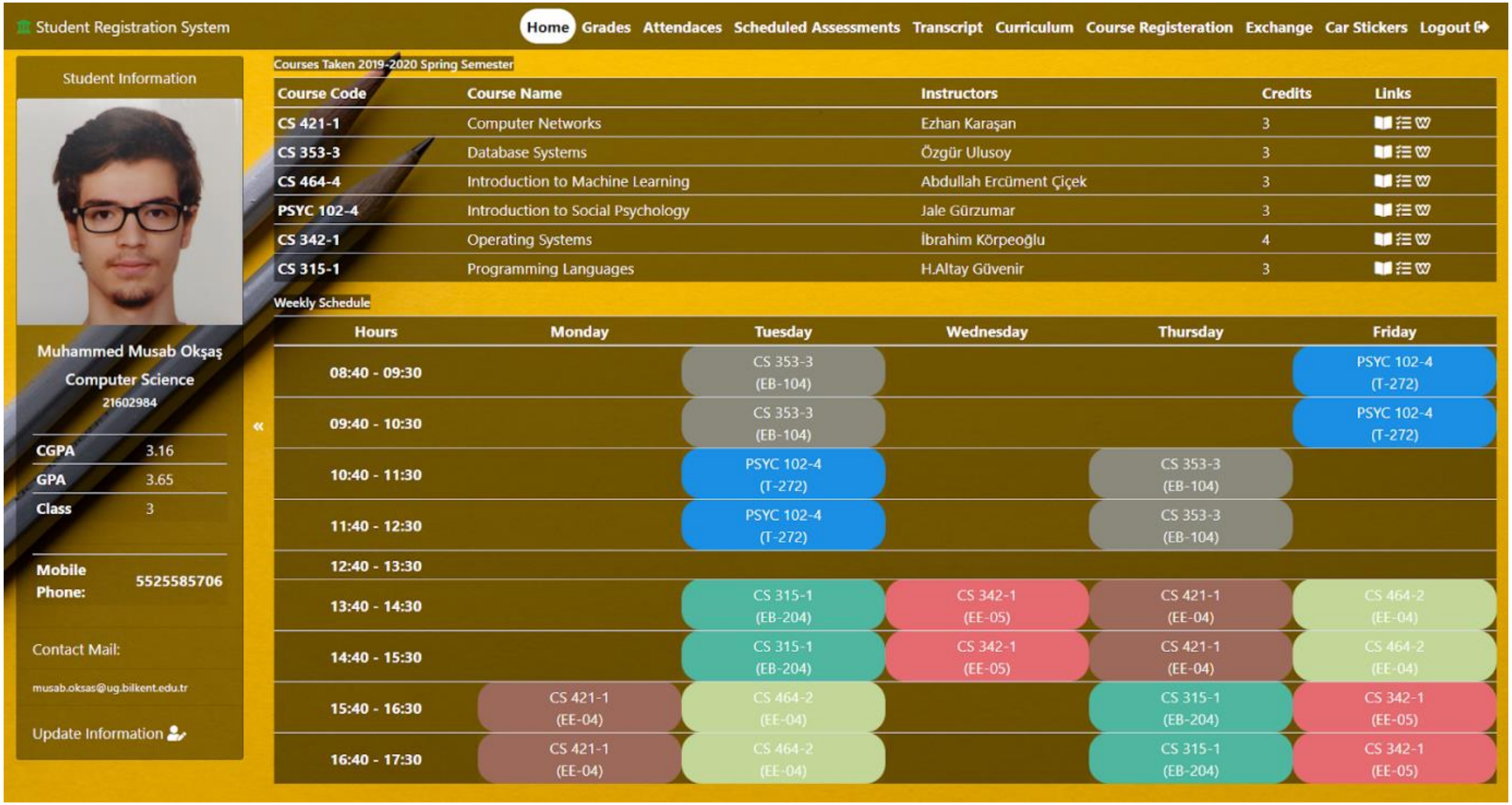
SQL Statements For Submit Button:

1-)
```
SELECT *
FROM Instructor i
INNER JOIN User u ON u.user_id = i.instructor_id
WHERE @Id = i.instructor_id AND @Password = u.password
```

2-)
```
SELECT *
FROM TeachingAssistant t
INNER JOIN User u ON u.user_id = t.ta_id
WHERE @Id = t.ta_id AND @Password = upassword
```

3-)
```
SELECT *
FROM Student s
INNER JOIN User u ON u.user_id = s.student_id
WHERE @Id = s.student_id AND @Password = u.password
```

3.2 Student Home Page



Muhammed Musab Okşaş
Computer Science
21602984

CGPA	3.16
GPA	3.65
Class	3
Mobile Phone:	5525585706

Contact Mail:
musab.oksas@ug.bilkent.edu.tr

Courses Taken 2019-2020 Spring Semester

Course Code	Course Name	Instructors	Credits	Links
CS 421-1	Computer Networks	Ezhan Karaşan	3	📖 ☰ W
CS 353-3	Database Systems	Özgür Ulusoy	3	📖 ☰ W
CS 464-4	Introduction to Machine Learning	Abdullah Ercüment Çiçek	3	📖 ☰ W
PSYC 102-4	Introduction to Social Psychology	Jale Gürzumar	3	📖 ☰ W
CS 342-1	Operating Systems	İbrahim Körpeoğlu	4	📖 ☰ W
CS 315-1	Programming Languages	H.Altay Güvenir	3	📖 ☰ W

Weekly Schedule

Hours	Monday	Tuesday	Wednesday	Thursday	Friday
08:40 - 09:30		CS 353-3 (EB-104)			PSYC 102-4 (T-272)
09:40 - 10:30		CS 353-3 (EB-104)			PSYC 102-4 (T-272)
10:40 - 11:30		PSYC 102-4 (T-272)		CS 353-3 (EB-104)	
11:40 - 12:30		PSYC 102-4 (T-272)		CS 353-3 (EB-104)	
12:40 - 13:30					
13:40 - 14:30		CS 315-1 (EB-204)	CS 342-1 (EE-05)	CS 421-1 (EE-04)	CS 464-2 (EE-04)
14:40 - 15:30		CS 315-1 (EB-204)	CS 342-1 (EE-05)	CS 421-1 (EE-04)	CS 464-2 (EE-04)
15:40 - 16:30	CS 421-1 (EE-04)	CS 464-2 (EE-04)		CS 315-1 (EB-204)	CS 342-1 (EE-05)
16:40 - 17:30	CS 421-1 (EE-04)	CS 464-2 (EE-04)		CS 315-1 (EB-204)	CS 342-1 (EE-05)

Process: Student Home Page can be considered as 3 main parts as Student Information, Taken Courses and Weekly Schedule. The information in the Student Information section is provided using the student's instant ID from the Student, Department and Phone tables. The information in the Courses section is obtained through the Instructor, Section, Course and Section tables. Weekly Schedule is displayed on the screen with the information in the Section and TimeSlot tables of each student. Furthermore, in order to get some information such as course name and user name, some extra tables are used.

SQL Statements For Student Information:

```sql
SELECT u.firstname, u.lastname, d.dept_name, s.student_id, s.cgpa, s.gpa,
s.class, p.phone_number, u.mail
FROM Student s
INNER JOIN User u ON s.student_id=u.user_id
INNER JOIN Member m ON u.user_id=m.user_id
INNER JOIN Department d ON m.dept_code = d.dept_code
INNER JOIN Phone p ON p.phone_id= u.user_id
WHERE s.student_id=@CurrentStudent_id
```

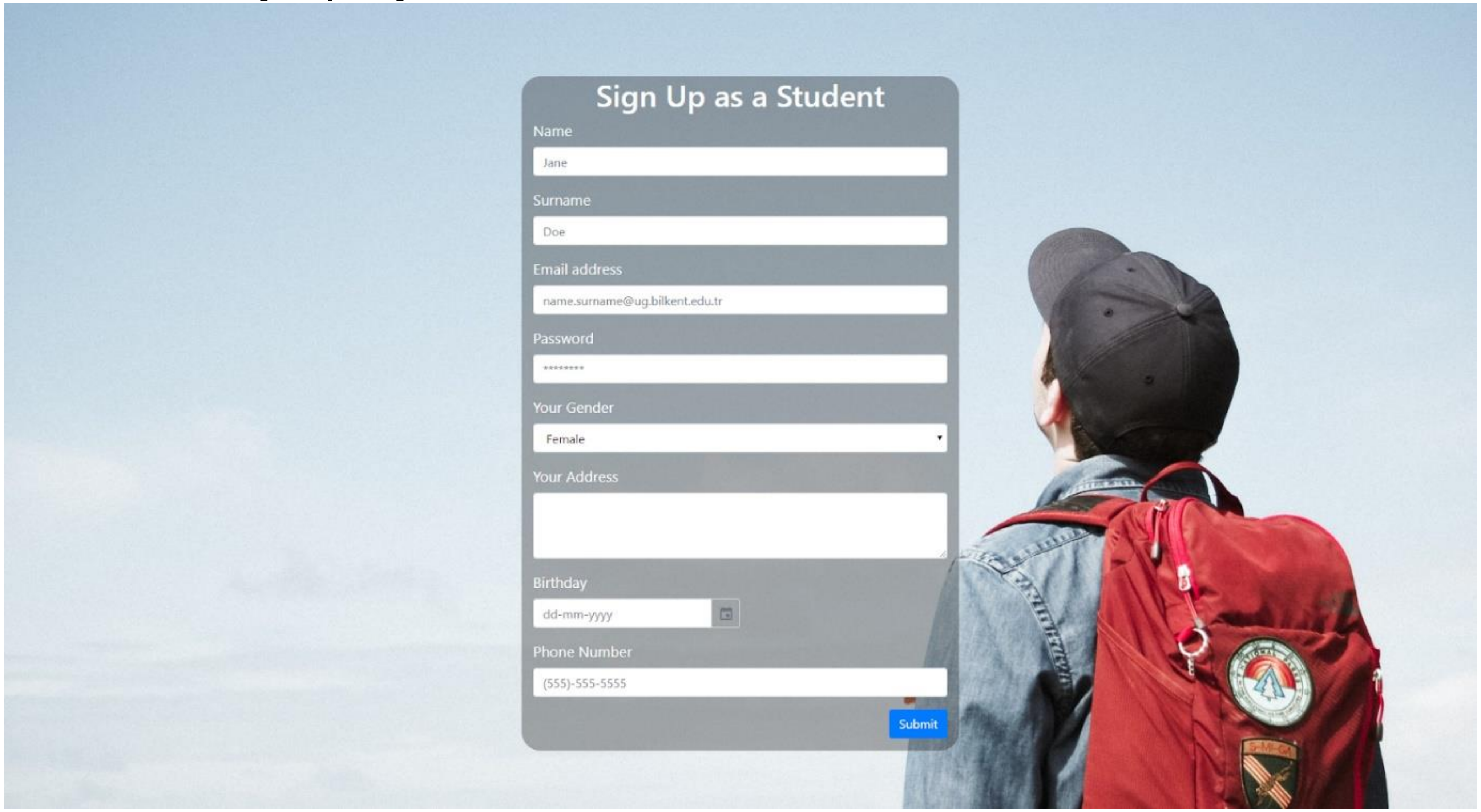
SQL Statements For Courses Taken:

```sql
SELECT c.course_code, sec.section_number, c.name,u.firstname,
u.lastname,c.credits
FROM Student s
INNER JOIN Takes t ON t.student_id=s.student_id
INNER JOIN Section sec ON t.section_id=sec.section_id
INNER JOIN Course c ON c.course_id = sec.course_id
INNER JOIN Instructor i ON i.instructor_id= sec.teacher_id
INNER JOIN User u ON i.instructor_id=u.user_id
WHERE s.student_id=@CurrentStudent_id AND
t.semester=@CurrentSemester AND t.year=@CurrentYear
```

SQL Statements for Weekly Schedule:

```sql
SELECT c.course_code, sec.section_number, sec.class, ts.day, ts.start_time,
ts.end_time
FROM Student s
INNER JOIN Takes t ON t.student_id=s.student_id
INNER JOIN Section sec ON t.section_id=sec.section_id
INNER JOIN Course c ON c.course_id = sec.course_id
INNER JOIN TimeSlot ts ON ts.section_id = sec.section_id
WHERE s.student_id=@CurrentStudent_id AND
t.semester=@CurrentSemester AND t.year=@CurrentY
```

3.3 Student Sign-Up Page



Sign Up as a Student

Name

Jane

Surname

Doe

Email address

name.surname@ug.bilkent.edu.tr

Password

****

Your Gender

Female

Your Address

Birthday

dd-mm-yyyy

Phone Number

(555)-555-5555

Inputs: @Name, @SurName, @Email, @Password, @Gender, @Address, @BirthDay @PhoneNumber

Process: New users can reach this page to become a student by clicking the Submit button on the Sign Up page. They need to type their Name, Surname, Email, Password, Gender, Address, Birthday information to become a student in SRS. New values that are @GeneratedID, Name, Surname, Email, Password inserted into the User table. Also, new values that are student_id, Name, Surname, Email, Password, Gender, Address, Birthday are inserted to the Student Table according to filled information by students. PhoneNumber is added to Phone table with the @GeneratedID as phone_id

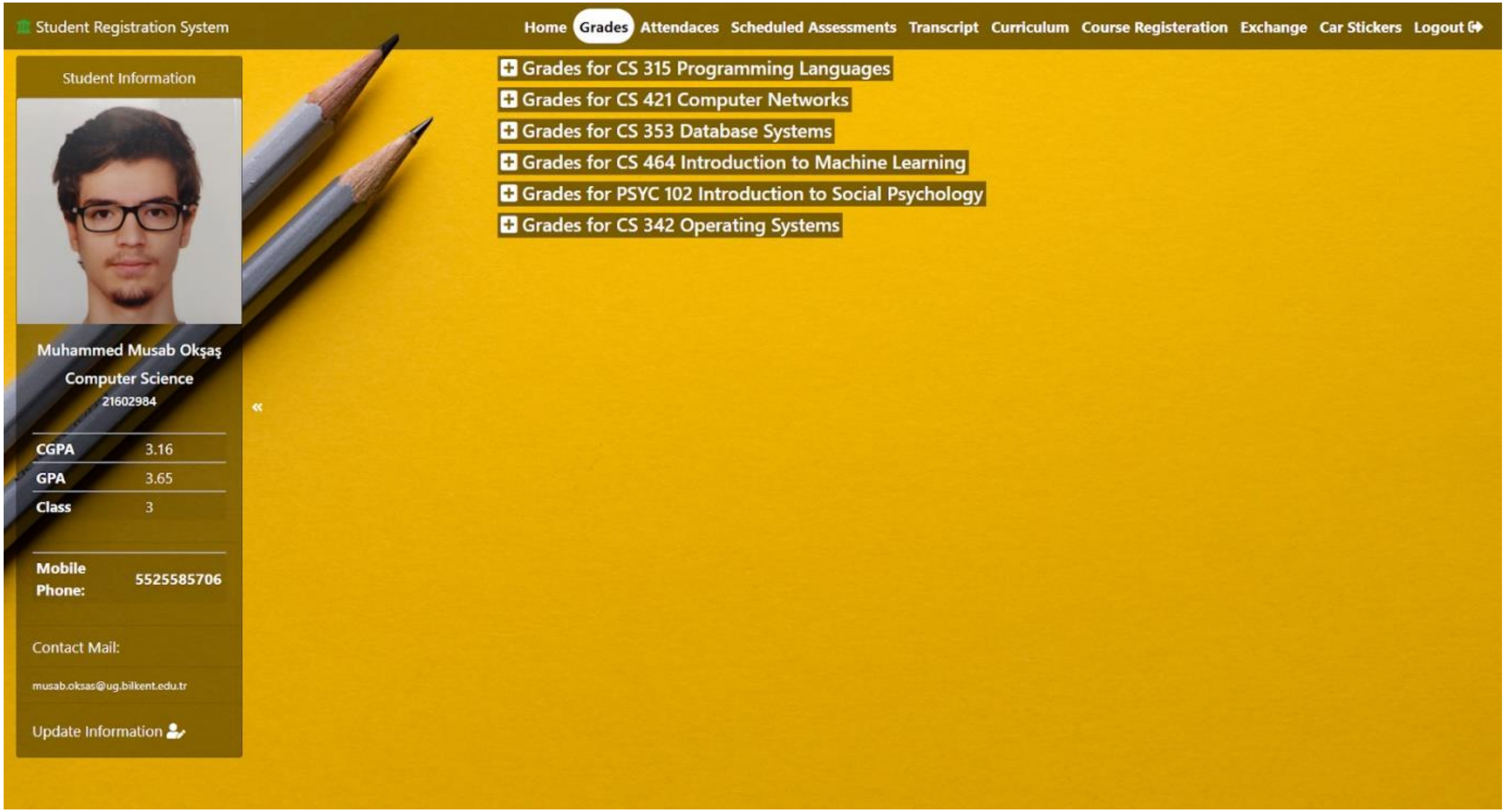
SQL Statements For Submit Button:

INSERT INTO User (user_id, firstname, lastname, mail, password) **VALUES**(@GeneratedID, @Name, @SurName, @Email, @Password);

INSERT INTO Student (student_id, address, gpa, cpga, erasmus_application_point, gender, date_of_birth, age, current_semester) **VALUES**(@GeneratedID,@Address, Null, Null,Null, @Gender, @BirthDay, Null, Null);

INSERT INTO Phone (phone_number, phone_id) **VALUES**(@PhoneNumber, @GeneratedID)

3.4 Grades Page



Student Registration System

Home Grades Attendaces Scheduled Assessments Transcript Curriculum Course Registeration Exchange Car Stickers Logout

Student Information

Muhammed Musab Okşaş
Computer Science
21602984

CGPA	3.16
GPA	3.65
Class	3
Mobile Phone:	5525585706

Contact Mail:

musab.oksas@ug.bilkent.edu.tr

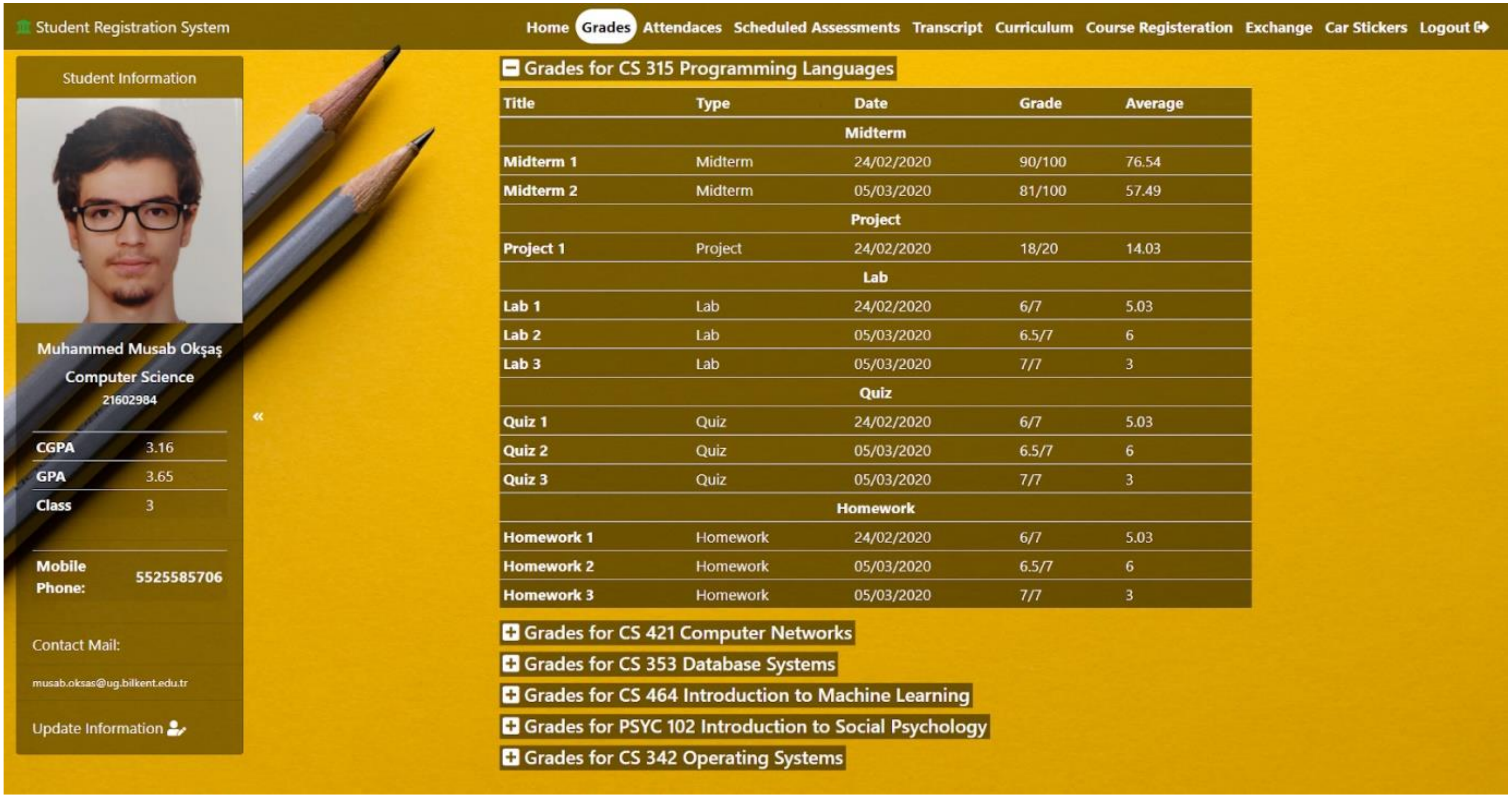
Update Information

- Grades for CS 315 Programming Languages
- Grades for CS 421 Computer Networks
- Grades for CS 353 Database Systems
- Grades for CS 464 Introduction to Machine Learning
- Grades for PSYC 102 Introduction to Social Psychology
- Grades for CS 342 Operating Systems

Process: Students will not show their grades in the first entry to the Grade page. There will be only course names on the page. Course names will be limited according to the time and student ID after the student and course table are linked using intermediate tables.

SQL Statements For All Button Names:

```
SELECT c.course_code, c.name
FROM Student s
INNER JOIN Takes t ON t.student_id=s.student_id
INNER JOIN Section sec ON t.section_id=sec.section_id
INNER JOIN Course c ON c.course_id = sec.course_id
WHERE s.student_id=@CurrentStudent_id AND
t.semester=@CurrentSemester AND t.year=@CurrentYear
```



Student Information

Muhammed Musab Okşaş
Computer Science
21602984

CGPA	3.16
GPA	3.65
Class	3
Mobile Phone:	5525585706

Contact Mail:

musab.oksas@ug.bilkent.edu.tr

Update Information

Grades for CS 315 Programming Languages

Title	Type	Date	Grade	Average
		Midterm		
Midterm 1	Midterm	24/02/2020	90/100	76.54
Midterm 2	Midterm	05/03/2020	81/100	57.49
		Project		
Project 1	Project	24/02/2020	18/20	14.03
		Lab		
Lab 1	Lab	24/02/2020	6/7	5.03
Lab 2	Lab	05/03/2020	6.5/7	6
Lab 3	Lab	05/03/2020	7/7	3
		Quiz		
Quiz 1	Quiz	24/02/2020	6/7	5.03
Quiz 2	Quiz	05/03/2020	6.5/7	6
Quiz 3	Quiz	05/03/2020	7/7	3
		Homework		
Homework 1	Homework	24/02/2020	6/7	5.03
Homework 2	Homework	05/03/2020	6.5/7	6
Homework 3	Homework	05/03/2020	7/7	3

Grades for CS 421 Computer Networks

Grades for CS 353 Database Systems

Grades for CS 464 Introduction to Machine Learning

Grades for PSYC 102 Introduction to Social Psychology

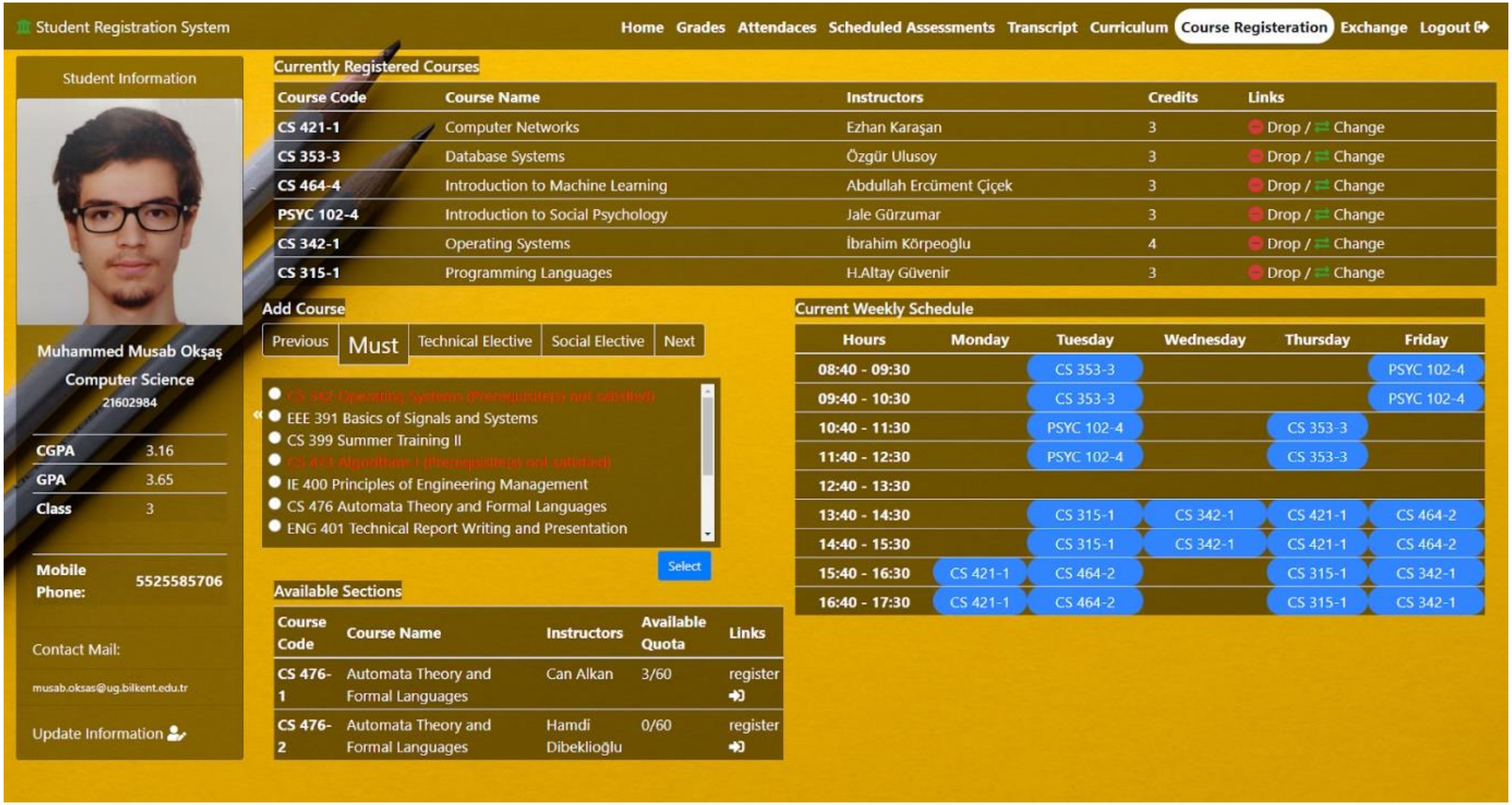
Grades for CS 342 Operating Systems

Process: On this page, students see midterm, project, quiz, lab and final grades. Since students have different grades in each lesson, they are categorized according to the lessons. The grades of each course can be seen by clicking the buttons with the name of that course. When students click the course name parts, @CurrentCourseName variable will take that course name.

SQL Statements For A Button with "@CurrentCourseName":

```
SELECT a.title, a.type, a.date, r.grade, a.average
FROM Result r
INNER JOIN Student s ON r.student_id = s.student_id
INNER JOIN Assignment a ON a.assignment_id=r.assignment_id
INNER JOIN Section sec ON a.section_id=r.section_id
INNER JOIN Course c ON c.course_id = sec.course_id
WHERE c.name = @CurrentCourseName AND
s.student_id=@CurrentStudent_i
```

3.5 Registration Page



Student Information

Muhammed Musab Okşaş

Computer Science

21602984

CGPA	3.16
GPA	3.65
Class	3
Mobile Phone:	5525585706

Contact Mail:

musab.oksas@ug.bilkent.edu.tr

Update Information

Currently Registered Courses

Course Code	Course Name	Instructors	Credits	Links
CS 421-1	Computer Networks	Ezhan Karaşan	3	Drop / Change
CS 353-3	Database Systems	Özgür Ulusoy	3	Drop / Change
CS 464-4	Introduction to Machine Learning	Abdullah Ercüment Çiçek	3	Drop / Change
PSYC 102-4	Introduction to Social Psychology	Jale Gürzumar	3	Drop / Change
CS 342-1	Operating Systems	İbrahim Körpeoğlu	4	Drop / Change
CS 315-1	Programming Languages	H.Altay Güvenir	3	Drop / Change

Add Course

- CS 342 Operating Systems (Prerequisite(s) not satisfied)
- EEE 391 Basics of Signals and Systems
- CS 399 Summer Training II
- CS 473 Algorithms I (Prerequisite(s) not satisfied)
- IE 400 Principles of Engineering Management
- CS 476 Automata Theory and Formal Languages
- ENG 401 Technical Report Writing and Presentation

Available Sections

Course Code	Course Name	Instructors	Available Quota	Links
CS 476-1	Automata Theory and Formal Languages	Can Alkan	3/60	register
CS 476-2	Automata Theory and Formal Languages	Hamdi Dibeklioğlu	0/60	register

Current Weekly Schedule

Hours	Monday	Tuesday	Wednesday	Thursday	Friday
08:40 - 09:30		CS 353-3			PSYC 102-4
09:40 - 10:30		CS 353-3			PSYC 102-4
10:40 - 11:30		PSYC 102-4		CS 353-3	
11:40 - 12:30		PSYC 102-4		CS 353-3	
12:40 - 13:30					
13:40 - 14:30		CS 315-1	CS 342-1	CS 421-1	CS 464-2
14:40 - 15:30		CS 315-1	CS 342-1	CS 421-1	CS 464-2
15:40 - 16:30	CS 421-1	CS 464-2		CS 315-1	CS 342-1
16:40 - 17:30	CS 421-1	CS 464-2		CS 315-1	CS 342-1

Inputs: @course_type, @course_name, @currentStudent_id

Process: In this page, students can select courses for their next semester. In the page there are 5 parts. First part student information part. Seconda part shows the selected courses with their course code, section_number, course name, instruction, credits and two options for giving chance students to change or drop the courses. Third part shows courses that can be taken by students with their type according to the curriculum of the student's department. Fourth part shows the current schedule of students according to the courses taken. Last part shows the section information according to the course that is selected on the add course part.

SQL Statements for Student info will be the same as Student Home Page

SQL Statements for courses taken by student;

SELECT c.course_code, sec.section_number, c.name,u.firstname, u.lastname,c.credits
FROM Student s
INNER JOIN Takes t **ON** t.student_id=s.student_id
INNER JOIN Section sec **ON** t.section_id=sec.section_id
INNER JOIN Course c **ON** c.course_id = sec.course_id
INNER JOIN Instructor i **ON** i.instructor_id= sec.teacher_id
INNER JOIN User u **ON** i.instructor_id=u.user_id
WHERE s.student_id=@CurrentStudent_id **AND** t.semester=@CurrentSemester **AND** t.year=@CurrentYear

SQL Statements for Weekly Schedule Info;

SELECT c.course_code, sec.section_number, sec.class, ts.day, ts.start_time, ts.end_time
FROM Student s
INNER JOIN Takes t **ON** t.student_id=s.student_id
INNER JOIN Section sec **ON** t.section_id=sec.section_id
INNER JOIN Course c **ON** c.course_id = sec.course_id
INNER JOIN TimeSlot ts **ON** ts.section_id = sec.section_id
WHERE s.student_id=@CurrentStudent_id **AND** t.semester=@CurrentSemester **AND** t.year=@CurrentYear

SQL Statements for Adding Courses;

SELECT c.course_name, c.course_code
FROM Student s
INNER JOIN Member m **ON** m.user_id = s.student_id
INNER JOIN Department d **ON** m.dept_code = d.dept_code
INNER JOIN Curriculum cu **ON** d.dept_code = cu_dept_code
INNER JOIN Course c **ON** c.course_id = cu.course_id
WHERE s.student_id = @CurrentStudent_id **AND** cu.course_type=@course_type;

SQL Statements for Selecting Section;

```
SELECT c.course_code, sec.section_number, c.course_name,
i.instructor_name, sec.available_quata, sec.total_quata
FROM Course c
INNER JOIN Section sec ON c.course_id = sec.course_id
INNER JOIN Instructor i ON sec.instructor_id = i.instructor_id
WHERE c.course_name = @course_name;
```

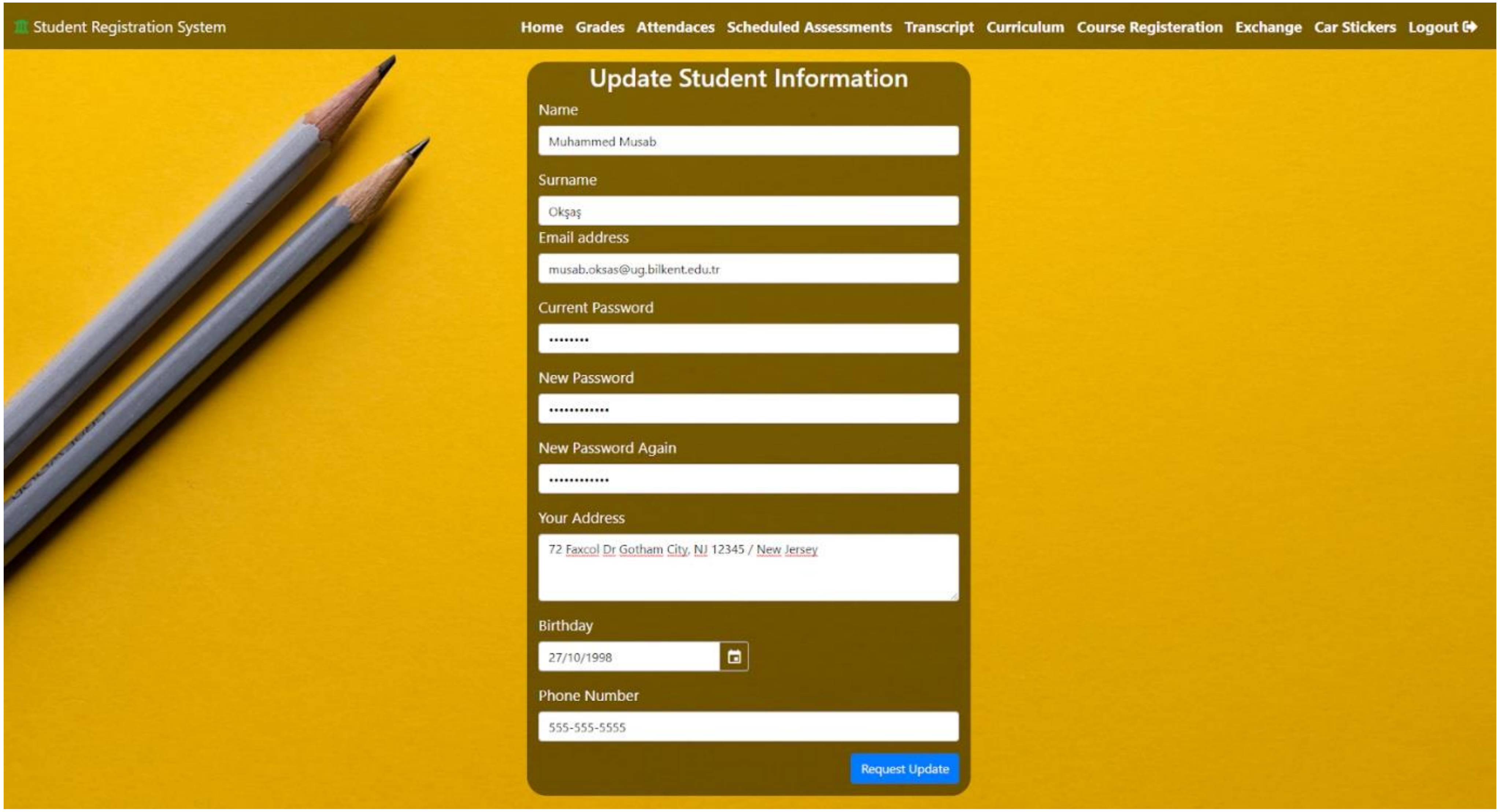
Other SQL Statements ;

```
INSERT INTO Attendance(student_id, course_id, section_id, title, date,
attendance_count, lecture_count)
VALUES(@CurrentStudent_id, c.course_id, sec.section_id, Null, Null, Null,
Null);

INSERT INTO Takes(student_id, course_id, section_id, attendance,
final_grade, letter_grade, year, semester)
VALUES(@CurrentStudent_id, c.course_id, sec.section_id, Null, Null, Null,
sec.year, sec.semester);

DELETE FROM Takes   WHERE section_id = @currentSection_id
DELETE FROM Attendance   WHERE section_id = @currentSection_i
```

3.6 Student Update Page



Update Student Information

Name
Muhammed Musab

Surname
Okşaş

Email address
musab.oksas@ug.bilkent.edu.tr

Current Password
••••••••

New Password
••••••••••••

New Password Again
••••••••••••

Your Address
72 Faxcol Dr Gotham City, NJ 12345 / New Jersey

Birthday
27/10/1998

Phone Number
555-555-5555

Inputs: @Name, @Surname, @Email, @CurrentPassword, @NewPassword1, @NewPassword2, @Address, @BirthDate, @PhoneNumber

Process: Students will be able to change information such as name, surname, email, password, gender, address, birthday, phone number on this page. These updates will not change the student's id information. When the Request Update button is clicked, new information will be updated in place of those in the Student, User and Phone Tables after checking whether current password is true and new passwords are typed the same .

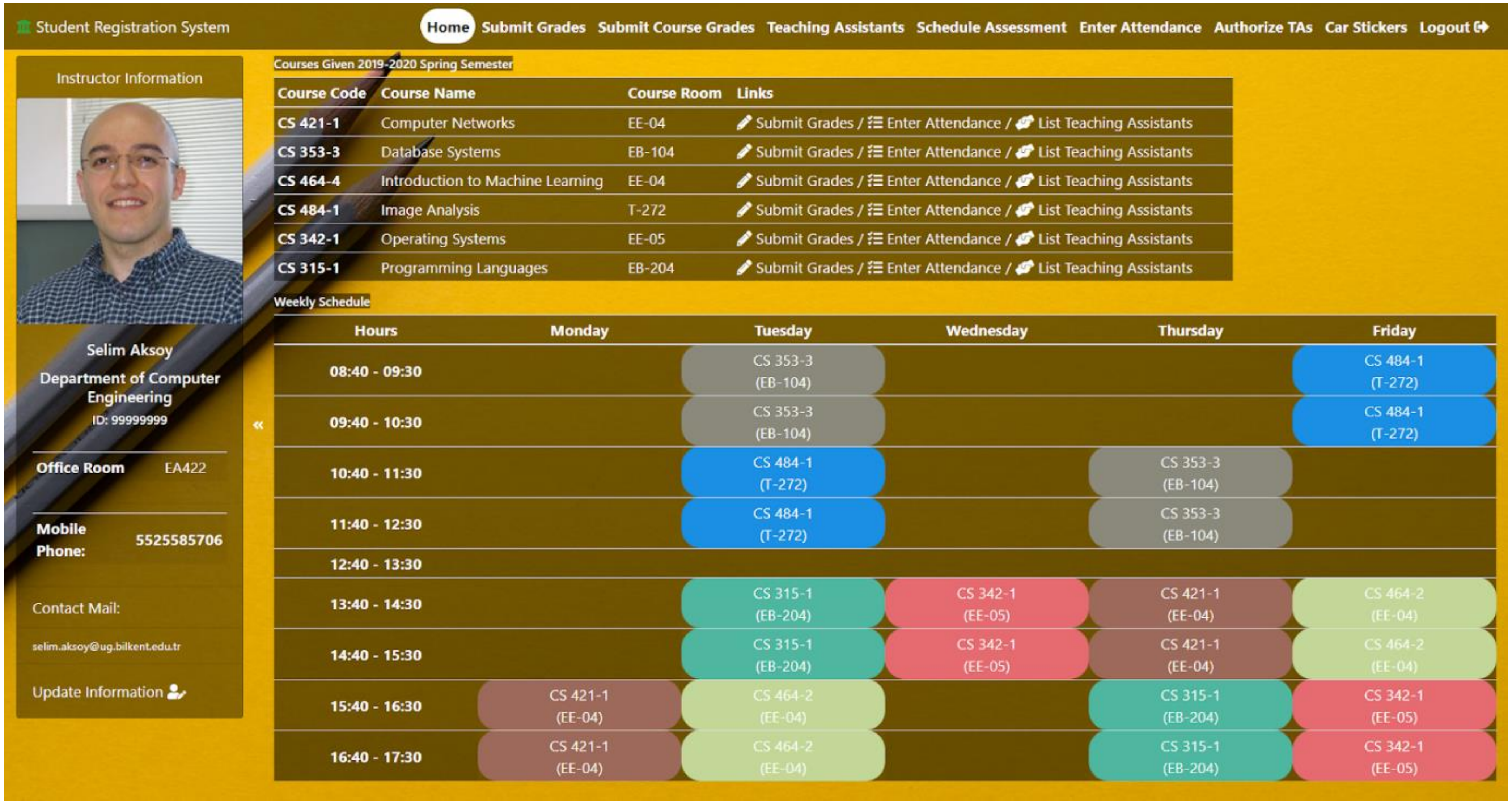
SQL Statements For Request Update Button:

```sql
UPDATE User
SET firstname = @Name, lastname=@Surname, mail=@Email,
password=@NewPassword1
WHERE user_id = @CurrentStudent_id;


UPDATE Student
SET address=@Address, date_of_birth= @BirthDate
WHERE student_id = @CurrentStudent_id;


UPDATE Phone
SET phone_number=@PhoneNumber
WHERE phone_id = @CurrentStudent_id;
```

3.7 Instructor Home Page



Instructor Information

Selim Aksoy

Department of Computer Engineering

ID: 99999999

Office Room	EA422
Mobile Phone:	5525585706

Contact Mail:

selim.aksoy@ug.bilkent.edu.tr

Update Information

Courses Given 2019-2020 Spring Semester

Course Code	Course Name	Course Room	Links
CS 421-1	Computer Networks	EE-04	Submit Grades / Enter Attendance / List Teaching Assistants
CS 353-3	Database Systems	EB-104	Submit Grades / Enter Attendance / List Teaching Assistants
CS 464-4	Introduction to Machine Learning	EE-04	Submit Grades / Enter Attendance / List Teaching Assistants
CS 484-1	Image Analysis	T-272	Submit Grades / Enter Attendance / List Teaching Assistants
CS 342-1	Operating Systems	EE-05	Submit Grades / Enter Attendance / List Teaching Assistants
CS 315-1	Programming Languages	EB-204	Submit Grades / Enter Attendance / List Teaching Assistants

Weekly Schedule

Hours	Monday	Tuesday	Wednesday	Thursday	Friday
08:40 - 09:30		CS 353-3 (EB-104)			CS 484-1 (T-272)
09:40 - 10:30		CS 353-3 (EB-104)			CS 484-1 (T-272)
10:40 - 11:30		CS 484-1 (T-272)		CS 353-3 (EB-104)	
11:40 - 12:30		CS 484-1 (T-272)		CS 353-3 (EB-104)	
12:40 - 13:30					
13:40 - 14:30		CS 315-1 (EB-204)	CS 342-1 (EE-05)	CS 421-1 (EE-04)	CS 464-2 (EE-04)
14:40 - 15:30		CS 315-1 (EB-204)	CS 342-1 (EE-05)	CS 421-1 (EE-04)	CS 464-2 (EE-04)
15:40 - 16:30	CS 421-1 (EE-04)	CS 464-2 (EE-04)		CS 315-1 (EB-204)	CS 342-1 (EE-05)
16:40 - 17:30	CS 421-1 (EE-04)	CS 464-2 (EE-04)		CS 315-1 (EB-204)	CS 342-1 (EE-05)

Process: Instructor Home Page consists of 3 parts such as Student Home Page and the only difference between them is using Instructor Id instead of Student Id.

SQL Statements For Instructor Information:

SELECT u.firstname, u.lastname, d.dept_name, i.instructor_id, i.office_no, p.phone_number, u.mail
FROM Instructor i
INNER JOIN User u **ON** i.instructor_id=m.user_id
INNER JOIN Member m **ON** u.user_id=m.user_id
INNER JOIN Department d **ON**m.dept_code = d.dept_code
INNER JOIN Phone p **ON**p.phone_id= u.user_id
WHERE i.instructor_id=@CurrentInstructor_id

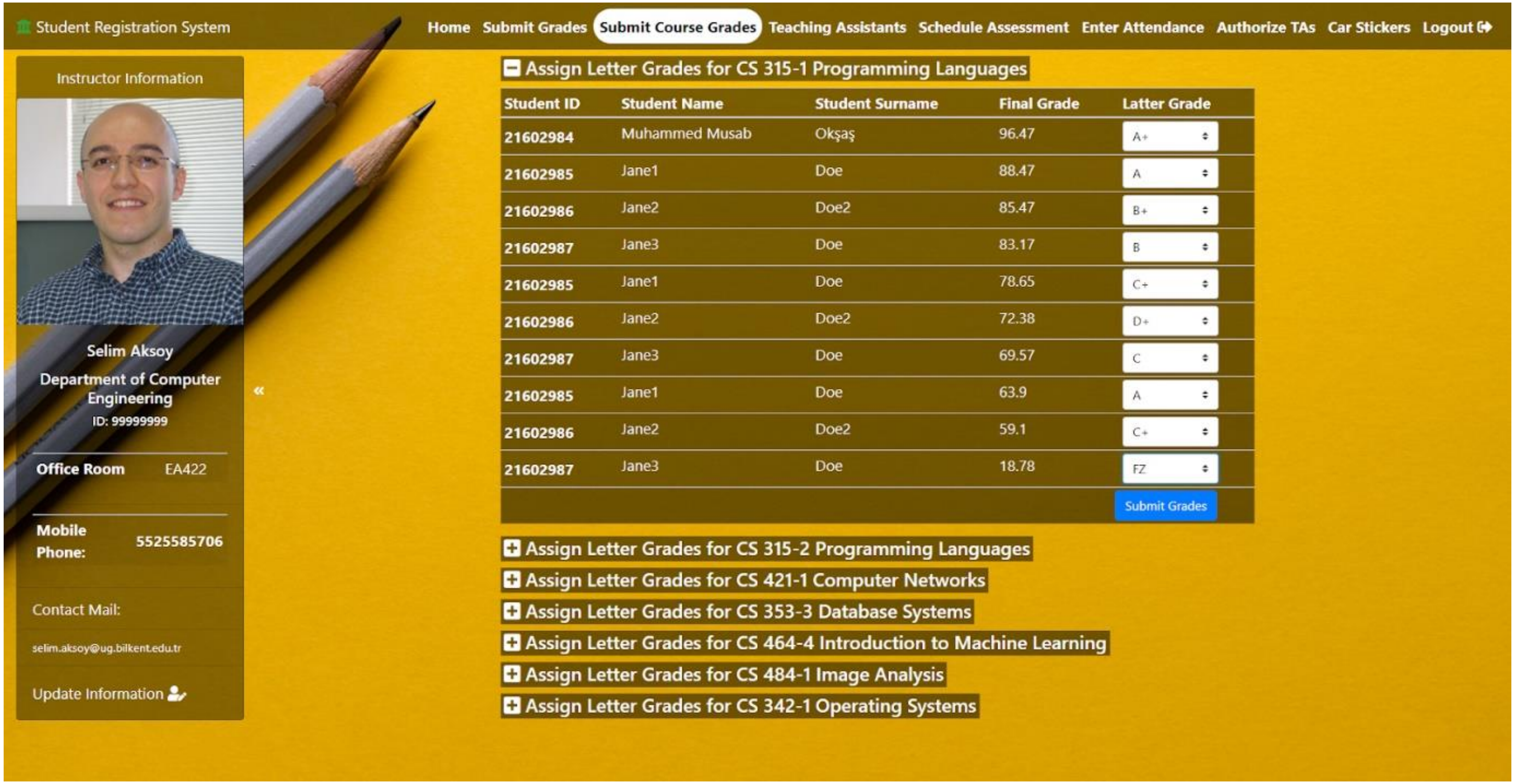
SQL Statements For Courses Given:

SELECT c.course_code, sec.section_number, c.name,sec.class
FROM Instructor i
INNER JOIN Section sec **ON** sec.teacher_id=i.instructor_id
INNER JOIN Course c **ON** c.course_id = sec.course_id
INNER JOIN Instructor i **ON** i.instructor_id= sec.teacher_id
WHERE i.instructor_id=@CurrentInstructor_id **AND** sec.semester=@CurrentSemester **AND** sec.year=@CurrentYear

SQL Statements For Weekly Schedule:

SELECT c.course_code, sec.section_number, sec.class, ts.day, ts.start_time, ts.end_time
FROM Instructor i
INNER JOIN Section sec **ON** sec.teacher_id=i.instructor_id
INNER JOIN Course c **ON** c.course_id = sec.course_id
INNER JOIN TimeSlot ts **ON** ts.section_id = sec.section_id
WHERE i.instructor_id=@CurrentInstructor_id **AND** sec.semester=@CurrentSemester **AND** sec.year=@CurrentYear

3.8 Instructor Submit Course Page



Instructor Information

Selim Aksoy
Department of Computer Engineering
ID: 99999999

Office Room	EA422
Mobile Phone:	5525585706

Contact Mail:
selim.aksoy@ug.bilkent.edu.tr

Assign Letter Grades for CS 315-1 Programming Languages

Student ID	Student Name	Student Surname	Final Grade	Latter Grade
21602984	Muhammed Musab	Okşaş	96.47	A+
21602985	Jane1	Doe	88.47	A
21602986	Jane2	Doe2	85.47	B+
21602987	Jane3	Doe	83.17	B
21602985	Jane1	Doe	78.65	C+
21602986	Jane2	Doe2	72.38	D+
21602987	Jane3	Doe	69.57	C
21602985	Jane1	Doe	63.9	A
21602986	Jane2	Doe2	59.1	C+
21602987	Jane3	Doe	18.78	FZ

Assign Letter Grades for CS 315-2 Programming Languages
Assign Letter Grades for CS 421-1 Computer Networks
Assign Letter Grades for CS 353-3 Database Systems
Assign Letter Grades for CS 464-4 Introduction to Machine Learning
Assign Letter Grades for CS 484-1 Image Analysis
Assign Letter Grades for CS 342-1 Operating Systems

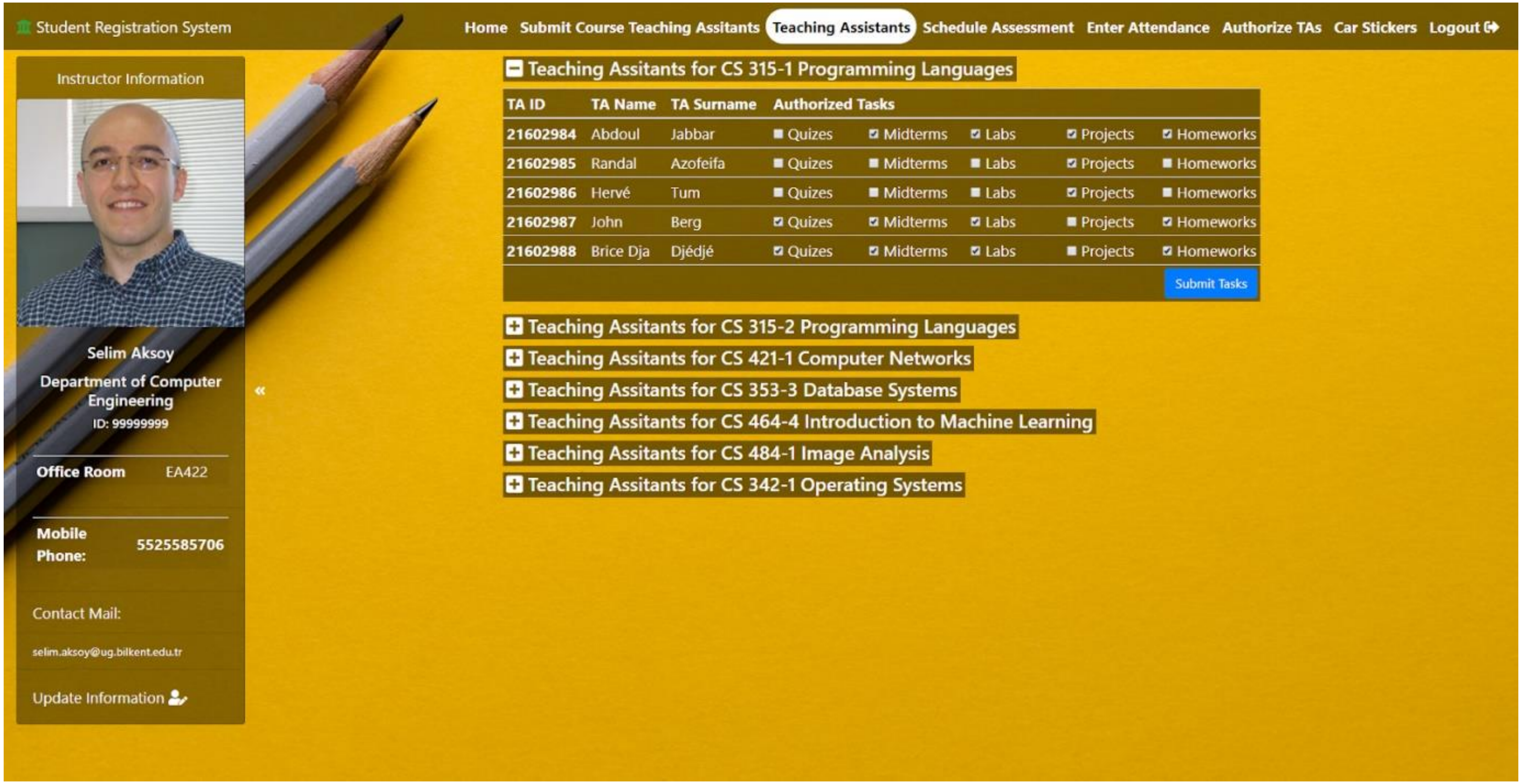
Inputs: @LetterGrade
Process: On this page, Instructors will be able to see the information and final grades of the students in a certain section by pressing the special buttons for each section. Afterwards, the letter grades in the Takes table, where the final grades are taken, will be entered by the teacher as input.

SQL Statements For A Button with “@CurrentSection_id”:

```
SELECT u.user_id, u.firstname, u.lastname, t.final_grade
FROM section sec
INNER JOIN Takes t ON t.section_id = sec.section_id
INNER JOIN Student s ON s.student_id = t.student_id
INNER JOIN User u ON s.student_id = u.user_id
WHERE sec.section_id = @CurrentSection_id


UPDATE Takes t
SET letter_grade = @LetterGrade
WHERE t.section_id = @CurrentSection_id
```

3.9 Instructor Assign Page



Selim Aksoy
Department of Computer Engineering
ID: 99999999

Office Room	EA422
Mobile Phone:	5525585706

Contact Mail:
selim.aksoy@ug.bilkent.edu.tr

⊟ Teaching Assitants for CS 315-1 Programming Languages

TA ID	TA Name	TA Surname	Authorized Tasks				
21602984	Abdoul	Jabbar	☐ Quizes	☑ Midterms	☑ Labs	☑ Projects	☑ Homeworks
21602985	Randal	Azofeifa	☐ Quizes	☐ Midterms	☐ Labs	☑ Projects	☐ Homeworks
21602986	Hervé	Tum	☐ Quizes	☐ Midterms	☐ Labs	☑ Projects	☐ Homeworks
21602987	John	Berg	☑ Quizes	☑ Midterms	☑ Labs	☐ Projects	☑ Homeworks
21602988	Brice Dja	Djédjé	☑ Quizes	☑ Midterms	☑ Labs	☐ Projects	☑ Homeworks

Submit Tasks

⊞ Teaching Assitants for CS 315-2 Programming Languages
⊞ Teaching Assitants for CS 421-1 Computer Networks
⊞ Teaching Assitants for CS 353-3 Database Systems
⊞ Teaching Assitants for CS 464-4 Introduction to Machine Learning
⊞ Teaching Assitants for CS 484-1 Image Analysis
⊞ Teaching Assitants for CS 342-1 Operating Systems

Inputs: @TaskType
Process: In this page, instructors will be able to assign tasks to teaching assistants. As in the Submit Course Page, the CurrentSection_id variable will be updated. In addition, each box that Instructor clicks corresponds to the TaskType variable.

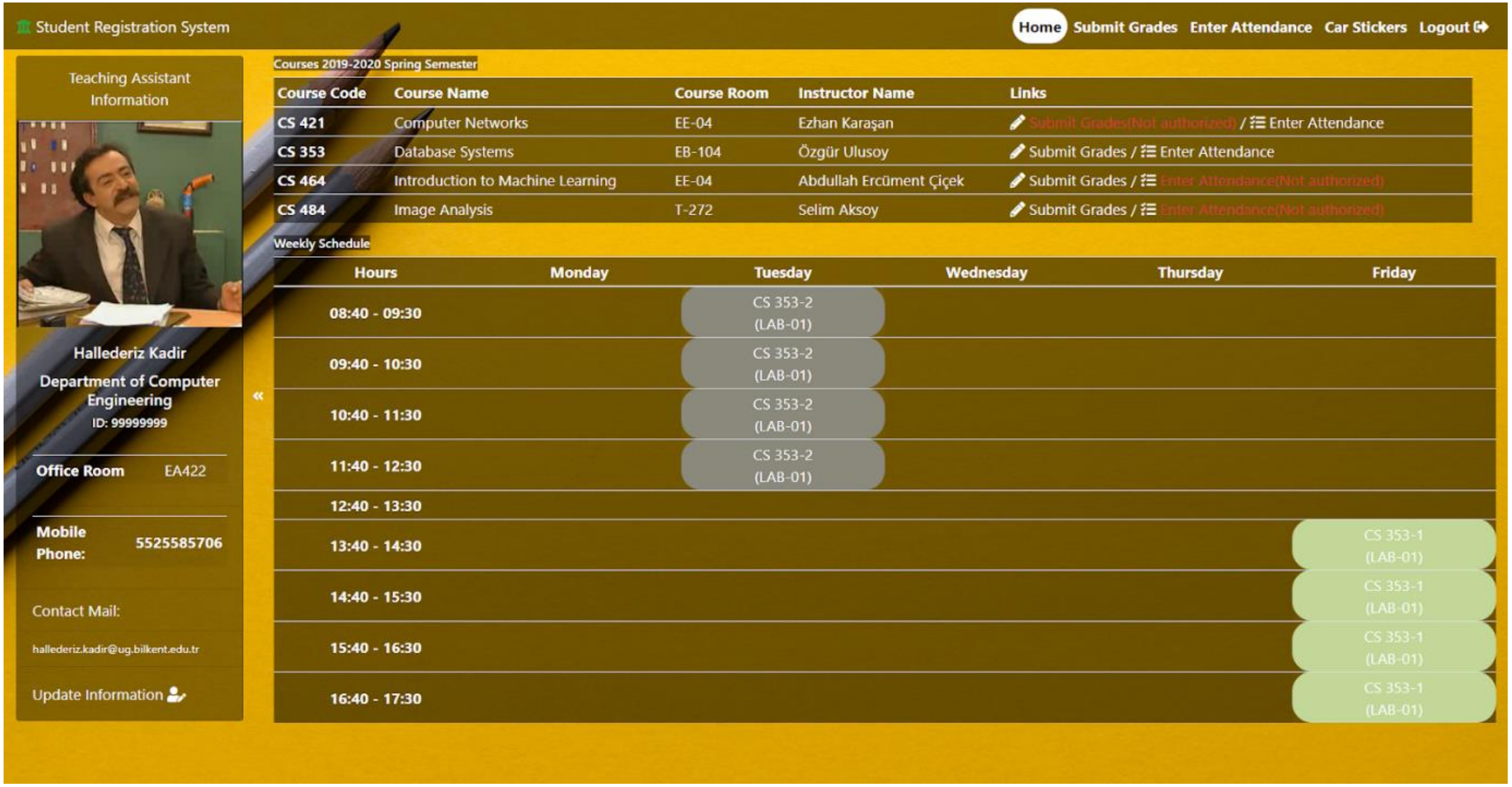
SQL Statements For A Button with "@CurrentSection_id":

```
SELECT ta.ta_id, ta.firstname, ta.lastname
FROM Instructor i
INNER JOIN Authorizes a ON i.instructor_id = a.instructor_id
INNER JOIN Task t ON t.task_id=a.task_id
INNER JOIN TeachingAssistant ta ON ta.ta_id=a.ta_id
INNER JOIN section sec ON sec.teacher_id = i.instructor_id
WHERE sec.section_id = @CurrentSection_id AND
i.instructor_id=@CurrentInstructor_id


INSERT INTO Task (task_id, task_type) VALUES(@GeneratedID,
@TaskType);


INSERT INTO Authorizes (task_id, instructor_id, ta_id)
VALUES(@GeneratedID, @CurrentInstructor_id, ta.ta_id)
```

3.10 Teaching Assistant Home Page



Teaching Assistant Information

Hallederiz Kadir

Department of Computer Engineering

ID: 99999999

Office Room	EA422
Mobile Phone:	5525585706

Contact Mail:

hallederiz.kadir@ug.bilkent.edu.tr

Update Information

Courses 2019-2020 Spring Semester

Course Code	Course Name	Course Room	Instructor Name	Links
CS 421	Computer Networks	EE-04	Ezhan Karaşan	Submit Grades(Not authorized) / Enter Attendance
CS 353	Database Systems	EB-104	Özgür Ulusoy	Submit Grades / Enter Attendance
CS 464	Introduction to Machine Learning	EE-04	Abdullah Ercüment Çiçek	Submit Grades / Enter Attendance(Not authorized)
CS 484	Image Analysis	T-272	Selim Aksoy	Submit Grades / Enter Attendance(Not authorized)

Weekly Schedule

Hours	Monday	Tuesday	Wednesday	Thursday	Friday
08:40 - 09:30		CS 353-2 (LAB-01)			
09:40 - 10:30		CS 353-2 (LAB-01)			
10:40 - 11:30		CS 353-2 (LAB-01)			
11:40 - 12:30		CS 353-2 (LAB-01)			
12:40 - 13:30					
13:40 - 14:30					CS 353-1 (LAB-01)
14:40 - 15:30					CS 353-1 (LAB-01)
15:40 - 16:30					CS 353-1 (LAB-01)
16:40 - 17:30					CS 353-1 (LAB-01)

Process: Instructor Home Page consists of 3 parts such as Student Home Page,Instructor Home Page and the only difference between them is using Teaching Assistant Id instead of Student and Instructor Id.

SQL Statements For Teaching Assistant Information:

SELECT u.firstname, u.lastname, d.dept_name, ta.ta_id, ta.office_no, p.phone_number, u.mail
FROM TeachingAssistant ta
INNER JOIN User u **ON** ta.ta_id=u.user_id
INNER JOIN Member m **ON** u.user_id=m.user_id
INNER JOIN Department d **ON** m.dept_code = d.dept_code
INNER JOIN Phone p **ON** p.phone_id= u.user_id
WHERE ta.ta_id=@CurrentTa_id

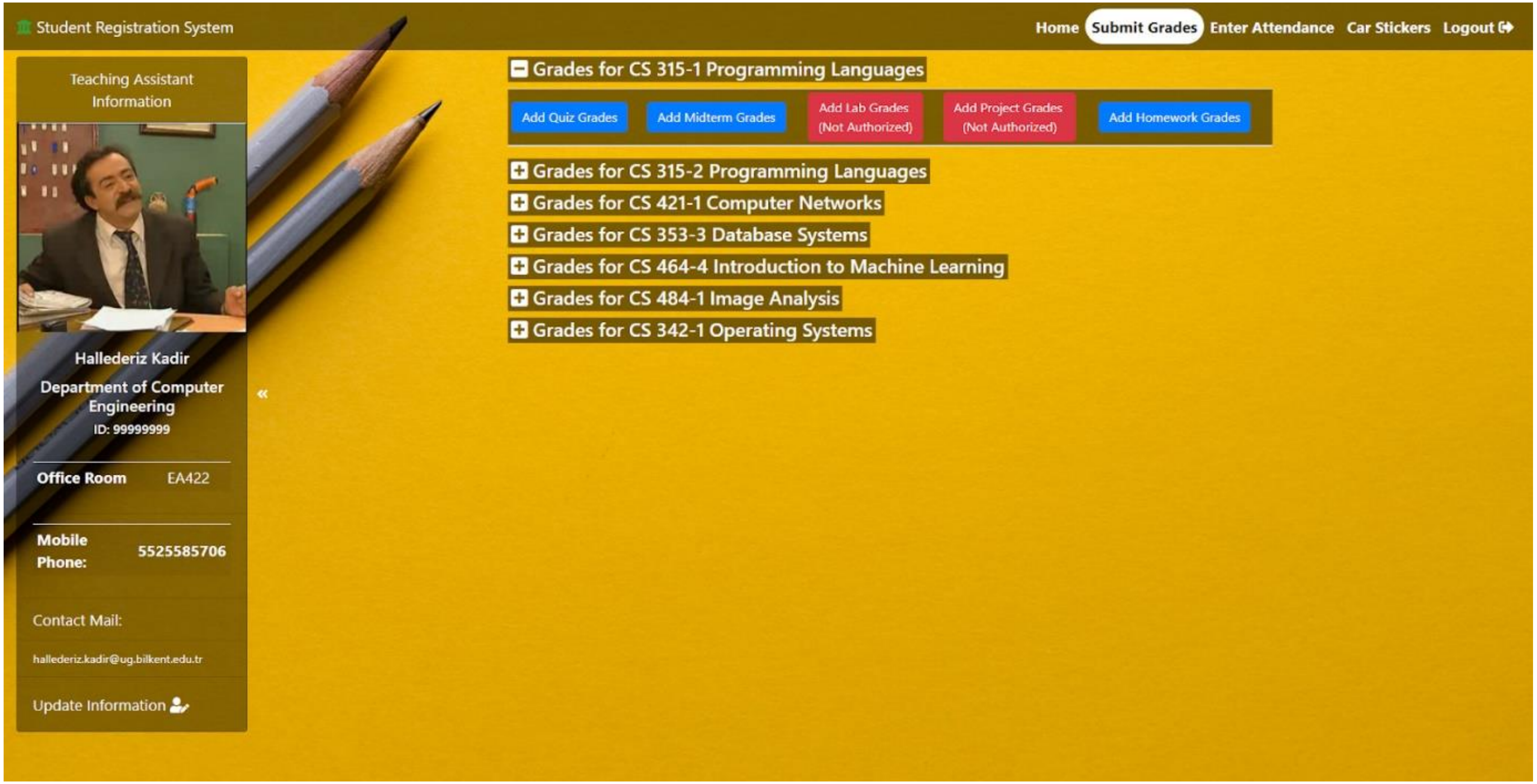
SQL Statements For Courses:

SELECT c.course_code, c.name, sec.class, u.firstname, u.lastname
FROM TeachingAssistant ta
INNER JOIN Assists a **ON** a.ta_id = ta.ta_id
INNER JOIN Course c **ON** c.course_id = a.course_id
INNER JOIN Section sec **ON** c.course_id= sec.course_id
INNER JOIN Instructor i **ON** i.instructor_id= sec.teacher_id
INNER JOIN User u **ON** i.instructor_id=u.user_id
WHERE ta.ta_id=@CurrentTa_id **AND** sec.semester=@CurrentSemester and sec.year=@CurrentYear

SQL Statements For Weekly Schedule:

SELECT c.course_code, sec.section_number, sec.class, ts.day, ts.start_time, ts.end_time
FROM TeachingAssistant ta
INNER JOIN Assists a **ON** a.ta_id = ta.ta_id
INNER JOIN Course c **ON** c.course_id = a.course_id
INNER JOIN Section sec **ON** c.course_id= sec.course_id
INNER JOIN TimeSlot ts **ON** ts.section_id = sec.section_id
WHERE ta.ta_id=@CurrentTa_id **AND** sec.semester=@CurrentSemester
AND sec.year=@CurrentYear

3.11 Teaching Assistant Submit Grades Page


Student Registration System
Home
Submit Grades
Enter Attendance
Car Stickers
Logout
Teaching Assistant Information
Update Information
Add Quiz Grades
Add Midterm Grades
Add Lab Grades (Not Authorized)
Add Project Grades (Not Authorized)
Add Homework Grades

Hallederiz Kadir
Department of Computer Engineering
ID: 99999999

Office Room	EA422
Mobile Phone:	5525585706

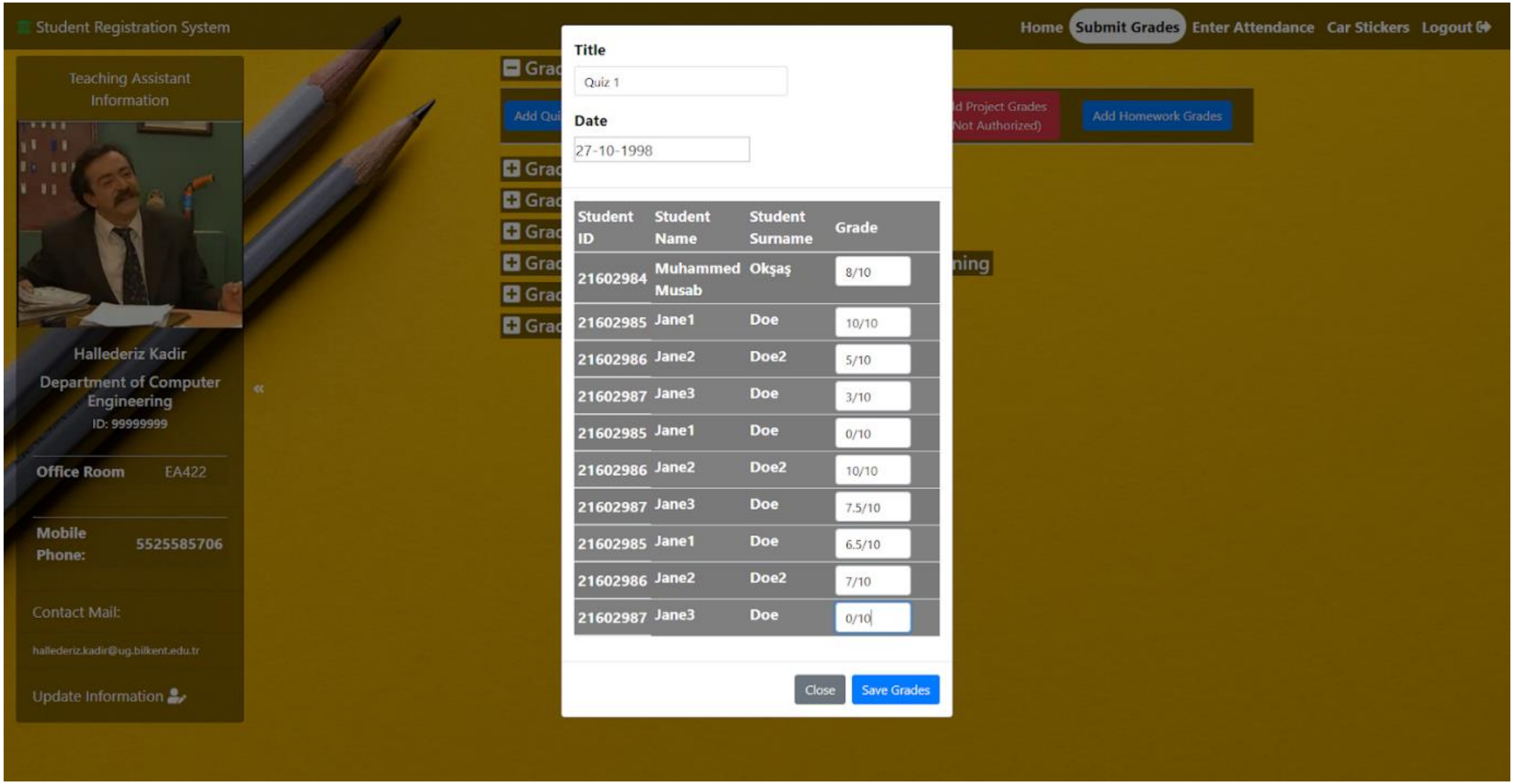
Contact Mail:
hallederiz.kadir@ug.bilkent.edu.tr

Grades for CS 315-1 Programming Languages
Grades for CS 315-2 Programming Languages
Grades for CS 421-1 Computer Networks
Grades for CS 353-3 Database Systems
Grades for CS 464-4 Introduction to Machine Learning
Grades for CS 484-1 Image Analysis
Grades for CS 342-1 Operating Systems

Process: On this page, the Teaching Assistant can see the students according to the assistant sections and grade them according to the assignment given by the teacher.

SQL Statements For All Button Names:

```
SELECT c.course_code, sec.section_number, c.name,
FROM TeachingAssistant ta
INNER JOIN Assists a ON a.ta_id=ta.ta_id
INNER JOIN Course c ON c.course_id = a.course_id
INNER JOIN Section sec ON c.course_id=sec.course_id
WHERE ta.ta_id=@CurrentTa_id AND sec.semester=@CurrentSemester
AND sec.year=@CurrentYear
```



Office Room EA422

Mobile Phone: 5525585706

Contact Mail:

hallederiz.kadir@ug.bilkent.edu.tr

Title

Quiz 1

Date

27-10-1998

Student ID	Student Name	Student Surname	Grade
21602984	Muhammed Musab	Okşaş	8/10
21602985	Jane1	Doe	10/10
21602986	Jane2	Doe2	5/10
21602987	Jane3	Doe	3/10
21602985	Jane1	Doe	0/10
21602986	Jane2	Doe2	10/10
21602987	Jane3	Doe	7.5/10
21602985	Jane1	Doe	6.5/10
21602986	Jane2	Doe2	7/10
21602987	Jane3	Doe	0/10

Inputs: @Grade, @Date, @Title, @Type

Process: In the pop-up menu students' information will be shown with their id, name, surname to the teaching assistants so that teaching assistants can give grades of students. After title, date and grades are entered by teaching assistants, students' grades will be updated.

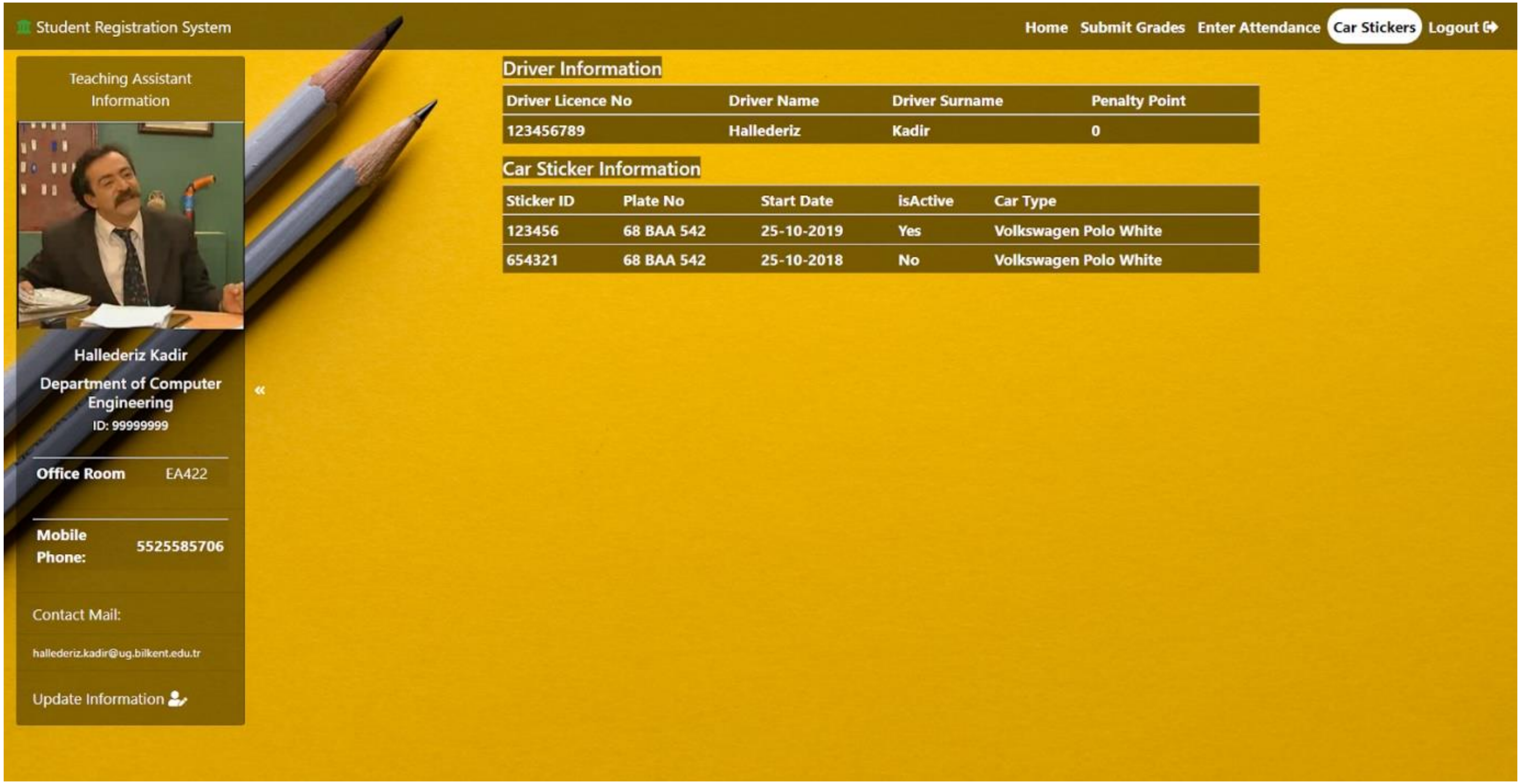
SQL Statements For Save Grades Button :

```
SELECT u.user_id, u.firstname, u.lastname
FROM section sec
INNER JOIN Assignment a ON a.section_id = sec.section_id
INNER JOIN Result r ON r.assignment_id= a.assignment_id
INNER JOIN Student s ON s.student_id = r.student_id
INNER JOIN User u ON s.student_id = u.user_id
WHERE sec.section_id = @CurrentSection_id


INSERT INTO Assignment(assignment_id, title, type, date, average,
course_id, section_id) VALUES(@GeneratedAssignmentID,@Title, @Type,
@Date, Null, sec.course_id, sec.section_id);


UPDATE Result r
SET grade = @Grade
WHERE  r.student_id = @CurrentStudent_id AND
r.assignment_id=@GeneratedAssignmentID
```

3.12 Car Sticker Page



Teaching Assistant Information

Hallederiz Kadir

Department of Computer Engineering

ID: 99999999

Office Room	EA422
Mobile Phone:	5525585706

Contact Mail:

hallederiz.kadir@ug.bilkent.edu.tr

Driver Information

Driver Licence No	Driver Name	Driver Surname	Penalty Point
123456789	Hallederiz	Kadir	0

Car Sticker Information

Sticker ID	Plate No	Start Date	isActive	Car Type
123456	68 BAA 542	25-10-2019	Yes	Volkswagen Polo White
654321	68 BAA 542	25-10-2018	No	Volkswagen Polo White

Process: On this page, sticker owners will be able to learn information such as Driver License and Penalty Point. Also, the information on the sticker is located under the Car Sticker Information section. Relationships between Car_Sticker, Owner and User tables will be sufficient to obtain the information here.

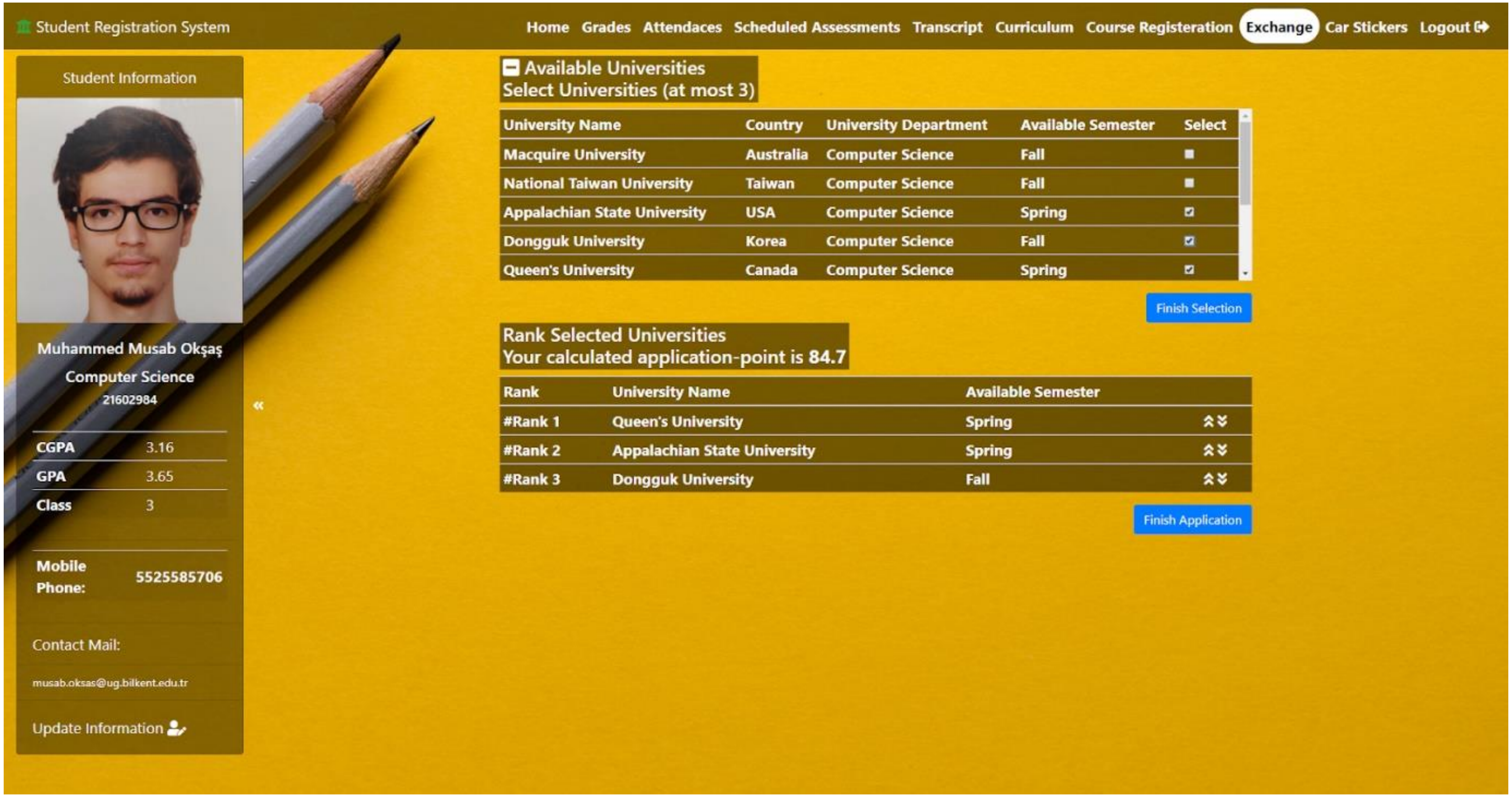
SQL Statements For Driver Information :

```
SELECT o.driver_licence_no, u.firstname, u.lastname, o.penalty_point
FROM User u
INNER JOIN Owner o ON o.owner_id = u.user_id
WHERE u.user_id = @CurrentUser_id
```

SQL Statements For Sticker Information :

```
SELECT cs.sticker_id, cs.plate_no, cs.start_date, cs.isActive, cs.car_type
FROM User u
INNER JOIN Owner o ON o.owner_id = u.user_id
INNER JOIN Car_Sticker cs ON cs.owner_id = u.user_id
WHERE u.user_id = @CurrentUser_id
```

3.13 Exchange Page



Muhammed Musab Okşaş
Computer Science
21602984

CGPA	3.16
GPA	3.65
Class	3
Mobile Phone:	5525585706

Contact Mail:
musab.oksas@ug.bilkent.edu.tr

Available Universities
Select Universities (at most 3)

University Name	Country	University Department	Available Semester	Select
Macquire University	Australia	Computer Science	Fall	■
National Taiwan University	Taiwan	Computer Science	Fall	■
Appalachian State University	USA	Computer Science	Spring	☑
Dongguk University	Korea	Computer Science	Fall	☑
Queen's University	Canada	Computer Science	Spring	☑

Rank Selected Universities
Your calculated application-point is 84.7

Rank	University Name	Available Semester	
#Rank 1	Queen's University	Spring	⌃⌄
#Rank 2	Appalachian State University	Spring	⌃⌄
#Rank 3	Dongguk University	Fall	⌃⌄

Process: On this page, the student will see a list of schools in the ExchangeSchool table that are eligible for the student. In addition, he will list these schools in line with his own request and apply from here. After applying, the necessary information will be added to the ExchangeApplication table

SQL Statements For Exchange University Information :

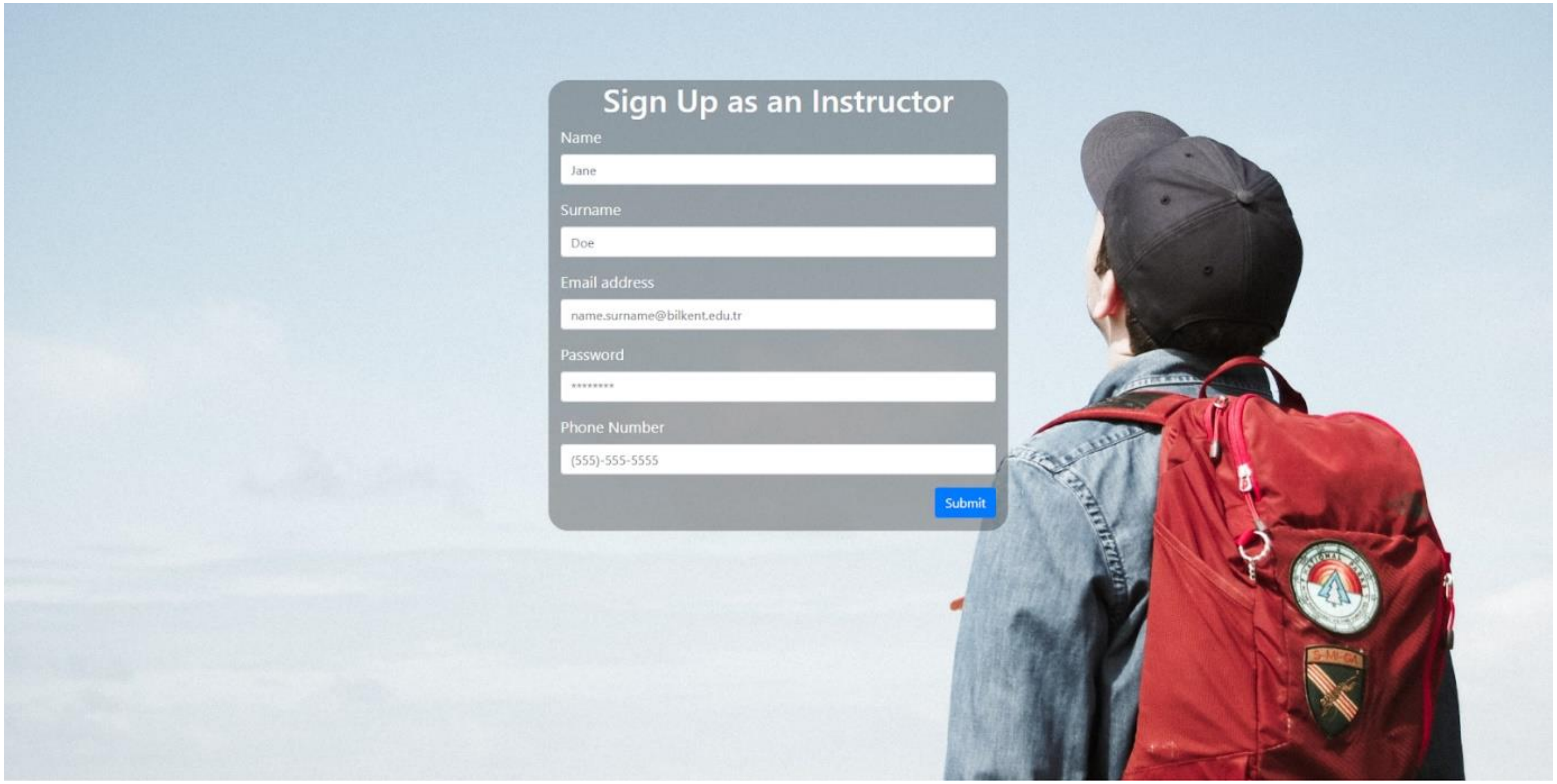
SELECT es.school_name,es.department, es.available_semester
FROM Student s
INNER JOIN ExchangeApplication ea **ON** ea.student_id = s.student_id
INNER JOIN ExchangeSchool es **ON** es.school_id = u.user_id
WHERE u.user_id = @CurrentUser_id

SQL Statements For Applied Exchange Universities :

SELECT es.school_id,es.department, es.available_semester, s.erasmus_application_point,
FROM Student s
INNER JOIN ExchangeApplication ea **ON** ea.student_id = s.student_id
INNER JOIN ExchangeSchool es **ON** es.school_id = u.user_id
WHERE u.user_id = @CurrentUser_id

INSERT INTO ExchangeApplication(student_id, school_id, application_status, application_point, applied_semester, year)
VALUES(@CurrentUser_id, es.school_id, application_status, s.erasmus_application_point, es.available_semester, @currentYear)

3.14 Instructor Sign-Up Page


Sign Up as an Instructor

Name

Jane

Surname

Doe

Email address

name.surname@bilkent.edu.tr

Password

****

Phone Number

(555)-555-5555

Inputs: @Name, @SurName, @Email, @Password, @PhoneNumber

Process: This page allows instructors to sign up. It uses the same methods as the sign up page for students.

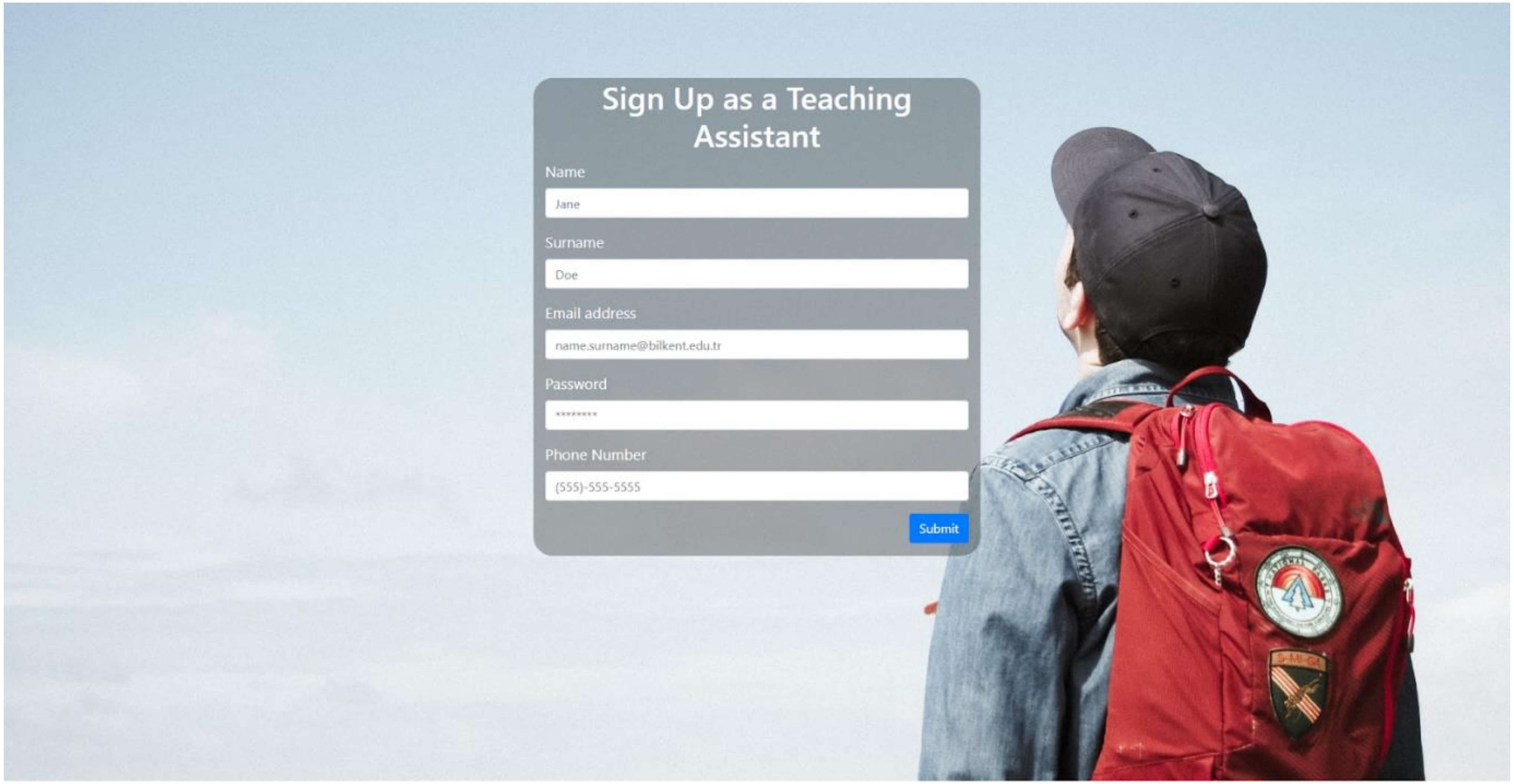
SQL Statements For Submit Button:

INSERT INTO User (user_id, firstname, lastname, mail, password) **VALUES**(@GeneratedID, @Name, @SurName, @Email, @Password);

INSERT INTO Instructor(instructor_id, office_no, office_hours) **VALUES**(@GeneratedID,Null, Null);

INSERT INTO Phone (phone_number, phone_id) **VALUES**(@PhoneNumber, @GeneratedID)

3.15 Teaching Assistant Sign-Up Page



Sign Up as a Teaching Assistant

Name

Jane

Surname

Doe

Email address

name.surname@bilkent.edu.tr

Password

****

Phone Number

(555)-555-5555

Submit

Inputs: @Name, @SurName, @Email, @Password, @PhoneNumber

Process: This page allows teaching assistants to sign up. It uses the same methods as the sign up pages for students and instructors.

SQL Statements For Submit Button:

> **INSERT INTO** User (user_id, firstname, lastname, mail, password) **VALUES**(@GeneratedID, @Name, @SurName, @Email, @Password);
>
> **INSERT INTO** TeachingAssistant(ta_id, office_no, office_hours) **VALUES**(@GeneratedID,Null, Null);
>
> **INSERT INTO** Phone (phone_number, phone_id) **VALUES**(@PhoneNumber, @GeneratedID);

4. Implementation Plan

For frontend of our system, we are planning to use Bootstrap, HTML, CSS, Javascript, React and for backend services we are planning to use Spring and MySQL Server.

5. Website

https://github.com/mmoksas68/CS-353-Student-Registration-System