Database Systems

CS 353

Design Report

Section 1/2

Group 18

Muhammad Ramish Saeed 21503436
F. Serdar Atalay 21300738
Sine Mete 21302158
Hilal Öztürk 21000633

Contents

1. Introduction 3
2. Revised E/R Diagram 3
3. Relational Schemas 3
4. Functional Components 14
5. User Interface and SQL statements 15
6. Advanced Database Components 26
7. Implementation Plan 28

1. Introduction

This is our design report for our project: A Course Information and Enrollment System similar to STARS system that is used by Bilkent University. This report will contain the revised E/R diagram after the changes made according to the feedback we received, the relational schema as it will be made from the diagram, along with the domains, candidate keys, and foreign keys for each table. The relations will all be in BCNF. Our report will also include the use case scenarios for each distinct user group. There will be GUI mockups and sketches for each individual screen showed to the users, along with clarifications of any confusing aspects of the mockups. SQL queries that populate the GUI mockups will also be provided. The reports, views, triggers, constraints, and stored procedures will be provided, where they are needed.

2. Revised E/R Diagram

See diagram attached at end of report.

Note: After getting beneficial feedback, we decided to remove the Academic Management table as we realize that it is not really needed in our system. Also we decided to include a new table "Account" which will hold all the accounts created by the instructor, teaching assistants or students.

3. Relational Schemas

Note: All the tables below have only a single candidate key, and so that is their primary key as well, by default.

3.1. University Employee

Relational Model:

University Employee(<u>emp-ID,</u> information, experience, appoint-date)

Functional Dependencies:

{information} → {experience, appoint-date}

Candidate Keys:

{(emp-ID)}

Foreign Keys:

No foreign key

Normal Form:

BCNF

Table Definition:

Create Table University Employee (

emp-ID	int,
information	varchar(50),
experience	varchar(25),
appoint-date	datetime(),
primary key (emp-ID));	

3.2. Management

Relational Model:

Management(<u>emp-name</u>, qualification, emp-job)

Functional Dependencies:

{qualification} → {emp-job}

Candidate Keys:

{(emp-ID)}

Foreign Keys:

No foreign key

Normal Form:

BCNF

Table Definition:

Create Table Management (

qualification	varchar(15),

emp-name varchar(20),

emp-job varchar(10));

3.3. Instructor

Relational Model:

Instructor(i-ID, i-name, i-salary, i-email)

Functional Dependencies:

{i-ID} → {i-name, i-salary, i-email}

Candidate Keys:

{(i-ID)}

Foreign Keys:

No foreign key

Normal Form:

BCNF

Table Definition:

Create Table Instructor (

i-ID varchar(8),

i-name varchar(20),

i-salary int,

i-email varchar(30),

primary key (i-ID));

3.4. Program

Relational Model:

Program(prog-ID, prog-name)

Functional Dependencies:

{prog-ID} → {prog-name}

Candidate Keys:

{(prog-ID)}

Foreign Keys:

No foreign key

Normal Form:

BCNF

Table Definition:

Create Table Program (

prog-ID varchar(8),

prog-name varchar(15),

primary key(prog-ID));

3.5. Course

Relational Model:

Course(c-ID, c-name, c-credit)

Functional Dependencies:

{c-ID} → {c-name, c-credit}

Candidate Keys:

{c-ID)}

Foreign Keys:

No foreign key

Normal Form:

BCNF

Table Definition:

Create Table Course (

c-ID varchar(6),

c-name varchar(15),

c-credit varchar(1),

primary key(c-ID));

3.6. Section

Relational Model:

Section(sec-ID, sec-name, sec-capacity)

Functional Dependencies:

{sec-ID} → {sec-name, sec-capacity}

Candidate Keys:

{(sec-ID)}

Foreign Keys:

No foreign key

Normal Form:

BCNF

Table Definition:

```
Create Table Section (
        sec-id          varchar(3),
        sec-name        varchar(10),
        capacity        int,
        primary key(sec-ID));
```

3.7. Account

Relational Model:

Account(username, password)

Functional Dependencies:

{username} → {password}

Candidate Keys:

{(username)}

Foreign Keys:

No foreign key

Normal Form:

BCNF

Table Definition:

Create Table Account(

username varchar(8),

password varchar(20),

primary key(username));

3.8. Teaching Assistant

Relational Model:

Teaching Assistant(ta-ID, ta-name, ta-salary, ta-email)

Functional Dependencies:

{ta-ID} → {ta-name, ta-salary, ta-email}

Candidate Keys:

{(ta-ID)}

Foreign Keys:

No foreign key

Normal Form:

BCNF

Table Definition:

Create Table Teaching Assistant (

ta-ID varchar(8),

ta-name varchar(20),

ta-email varchar(25),

ta-salary int,

primary key(ta-ID));

3.9. Grading

Relational Model:

Grading(grades, type)

Functional Dependencies:

No functional dependency

Candidate Keys:

No candidate key

Foreign Keys:

No foreign key

Normal Form:

BCNF

Table Definition:

Create Table Grading (

grading	char(2)
type	varchar(10));

3.10. Student

Relational Model:

Student(<u>s-ID</u>, s-name, s-email, s-phone, s-gpa)

Functional Dependencies:

{s-ID} → {s-name, s-email, s-phone, s-gpa}

Candidate Keys:

{(s-ID)}

Foreign Keys:

No foreign key

Normal Form:

BCNF

Table Definition:

Create Table Student (

s-ID	varchar(8),

s-name varchar(20),
s-email varchar(25),
s-phone varchar(10),
s-gpa float(3),
primary key(s-ID));

3.11. Exchange Program

Relational Model:

Exchange Program(country, uni-name, duration)

Functional Dependencies:

{country} → {uni-name, duration}

Candidate Keys:

No candidate key

Foreign Keys:

No foreign key

Normal Form:

BCNF

Table Definition:

Create Table Exchange Program (
country varchar(15),
uni-name varchar(30),
duration int);

3.11. part of

Relational Model:

Part of(<u>i-ID</u>, <u>prog-ID</u>)

Functional Dependencies:

No functional dependencies

Candidate Keys:

{(i-ID, prog-ID)}

Foreign Keys:

'i-ID' foreign key referenced to Instructor

'prog-ID' foreign key referenced to Program

Normal Form:

BCNF

Table Definition:

Create Table part of (

i-ID varchar(8),

prog-ID varchar(8),

primary key(i-ID, prog-ID),

foreign key (i-ID) references Instructor,

foreign key (prog-ID) references Program);

3.12. enrolls in

Relational Model:

enrolls in(s-ID, c-ID)

Functional Dependencies:

No functional dependencies

Candidate Keys:

{(s-ID, c-ID)}

Foreign Keys:

's-ID' foreign key referenced to Student

'c-ID' foreign key referenced to Course

Normal Form:

BCNF

Table Definition:

Create Table enrolls in (

s-ID varchar(8),

c-ID varchar(6),

primary key(s-ID, c-ID),

foreign key (s-ID) references Student,

foreign key (c-ID) references Course);

3.13. Has

Relational Model:

has(ta-ID. i-ID)

Functional Dependencies:

No functional dependencies

Candidate Keys:

{(ta-ID, i-ID)}

Foreign Keys:

'ta-ID' foreign key referenced to Teaching Assistant

'i-ID' foreign key referenced to Instructor

Normal Form:

BCNF

Table Definition:

Create Table has (

ta-ID varchar(8),

i-ID varchar(6),

primary key(ta-ID, i-ID),

foreign key (ta-ID) references Teaching Assistant,

foreign key (i-ID) references Instructor);

3.14. Assist

Relational Model:

assist(ta-ID, sec-ID)

Functional Dependencies:

No functional dependencies

Candidate Keys:

{(ta-ID, sec-ID)}

Foreign Keys:

'ta-ID' foreign key referenced to Teaching Assistant

'sec-ID' foreign key referenced to Section

Normal Form:

BCNF

Table Definition:

```
Create Table assist (
        ta-ID                   varchar(8),
        sec-ID                  varchar(8),
        primary key(ta-ID, sec-ID),
        foreign key (ta-ID) references Teaching Assistant
        foreign key (sec-ID) references Section);
```

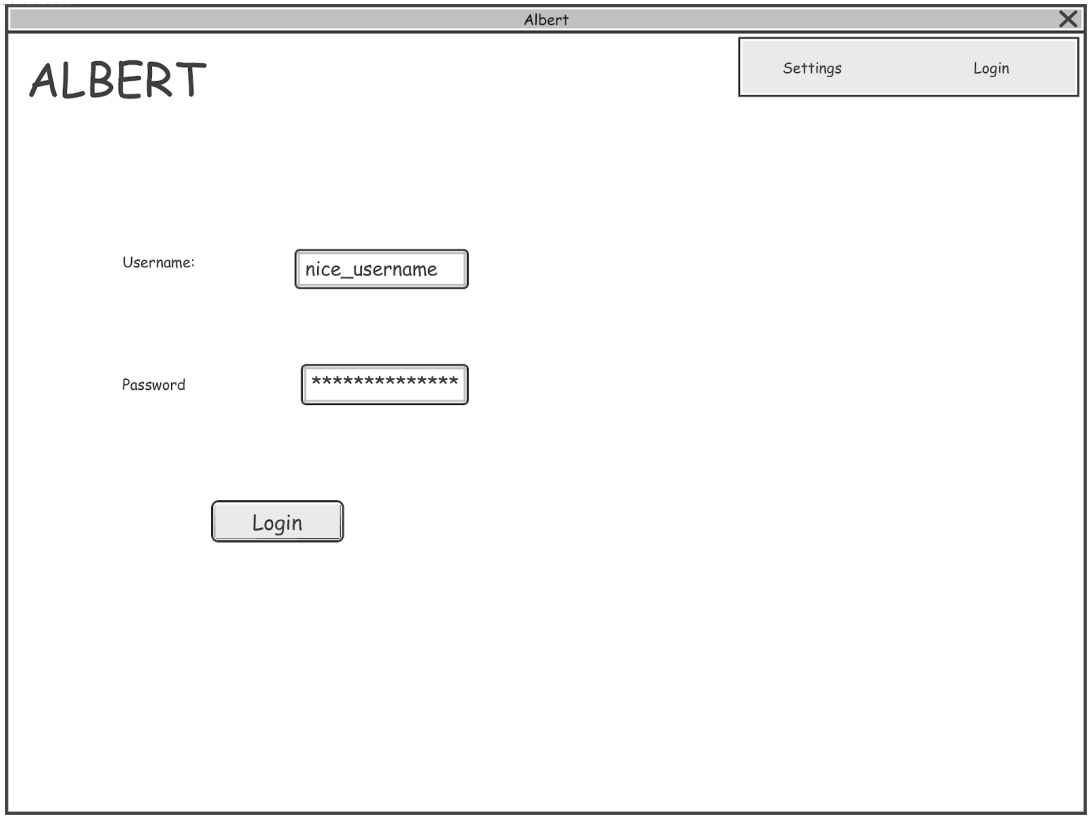
The functional dependencies have all been listed above with the relational schema of each table. The relations have all been analyzed with respect to these dependencies and there is no need to further decompose them, since they all satisfy Boyce-Codd normal form.

4. Functional Components

4.1. Use Case Diagram

5. User Interface and SQL statements

5.1. Login View



Inputs: username, password

Here the user, if registered, enters their username and password. If the combination is valid, they are redirected to their homepage.

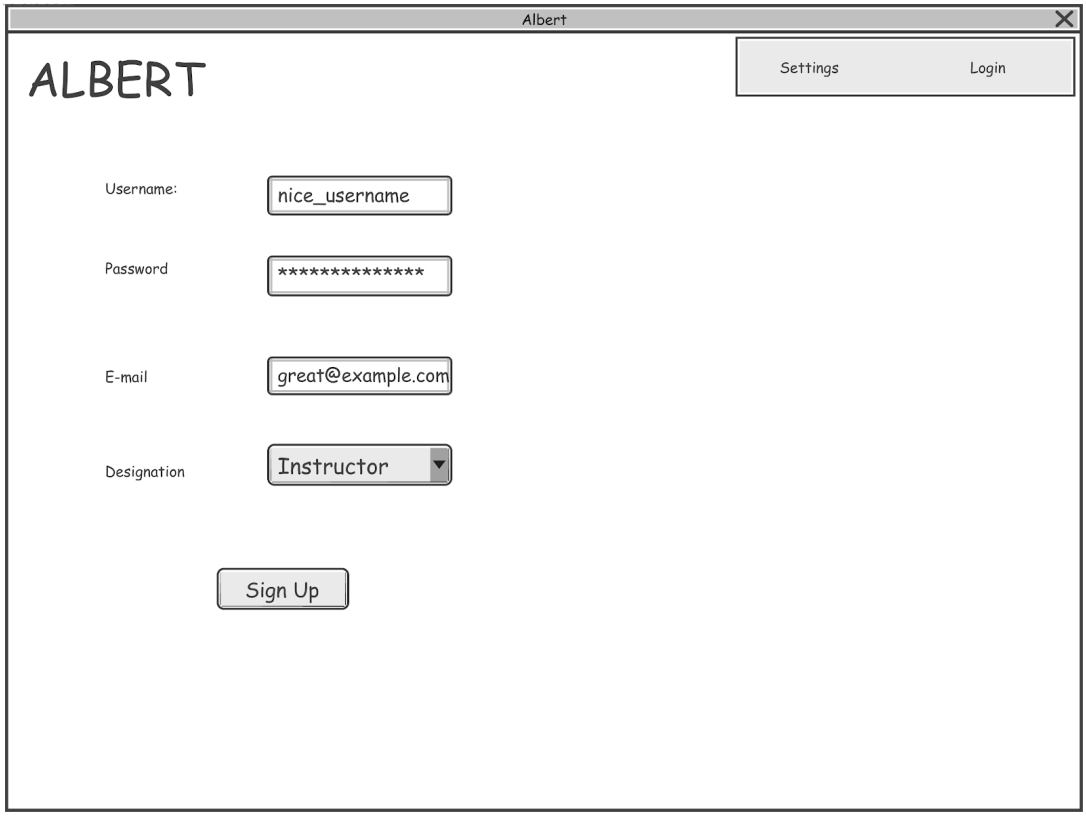
SQL Statement:

SELECT username

FROM Account

WHERE (username = “<input>”) and (password = “<input>”)

5.2. Sign Up



Inputs: username, password, email

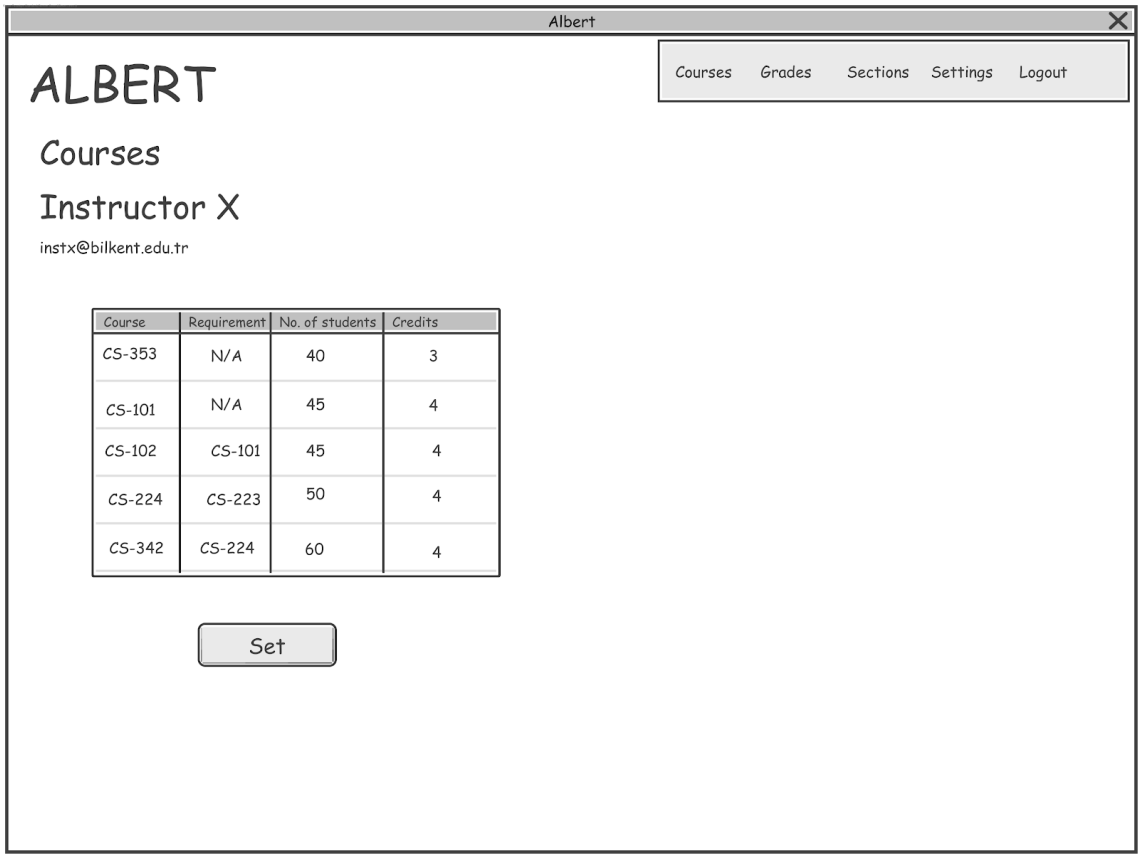
Here the user enters their desired username, password, email. The user also has to sign up according to the designation they occupy in the university i.e as Instructor, Teaching Assistant or Student and that is done from the drop-down option.

SQL Statement:

INSERT INTO Account

VALUES(username, password, email);

5.3. Instructor Course Page



Inputs: c-ID, c-name, c-credit

This is the page where the instructor will be able to determine course requirements for the courses they teach and also sets the number of students he will teach for each section and further assign the credits for each course. The name and email of the instructor will be displayed on top left hand side.

SQL statement:

INSERT INTO courses

VALUES (c-ID, c-name, c-credit);

//This query selects and displays all the courses which the instructor teaches

SELECT(*)

FROM courses C

WHERE C.c-ID = “<input>”;

//This query selects those instructors which are part of a particular program(department) in the university

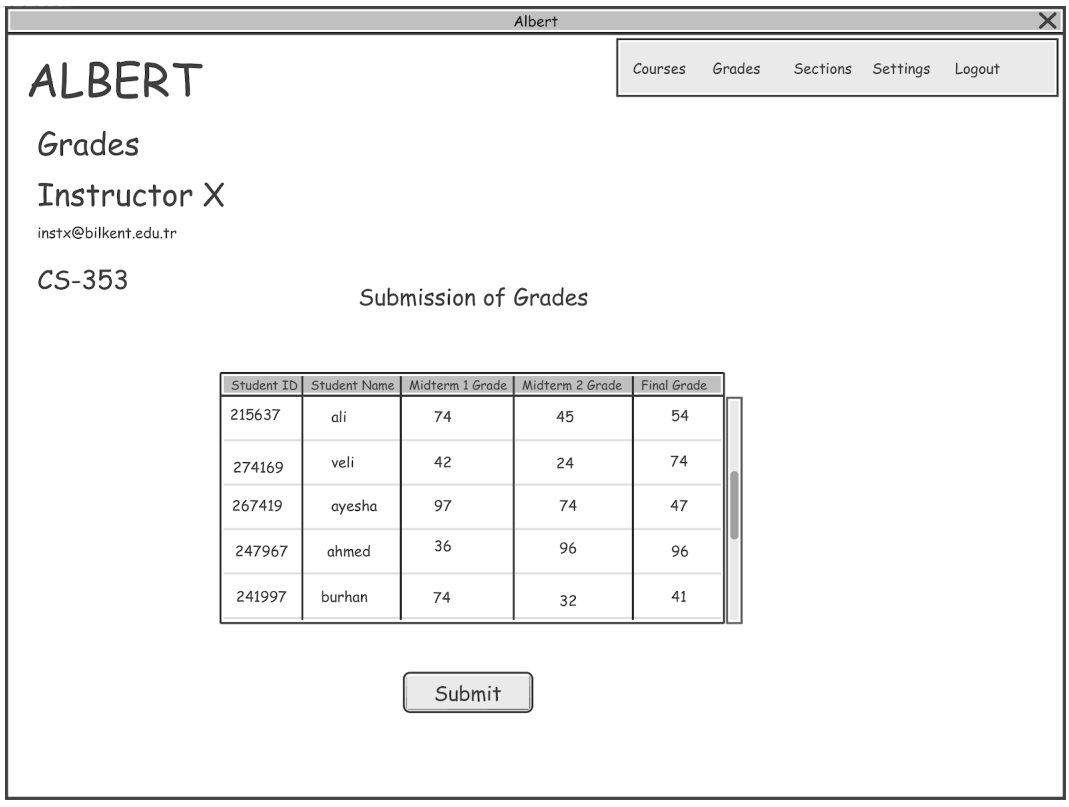
SELECT Y.i-ID, Y.i-name

FROM part of Y, Instructor I

WHERE Y.i-ID = I.i-ID

GROUP BY Y.prog-ID

5.4. Instructor Grade tests(midterm and finals)



Inputs: s-ID, s-name, grades

This is the page where the instructor will be able to enter the midterm and final exam grades for each student.

SQL statement:

INSERT INTO grading

VALUES (s-ID, s-name, grades);

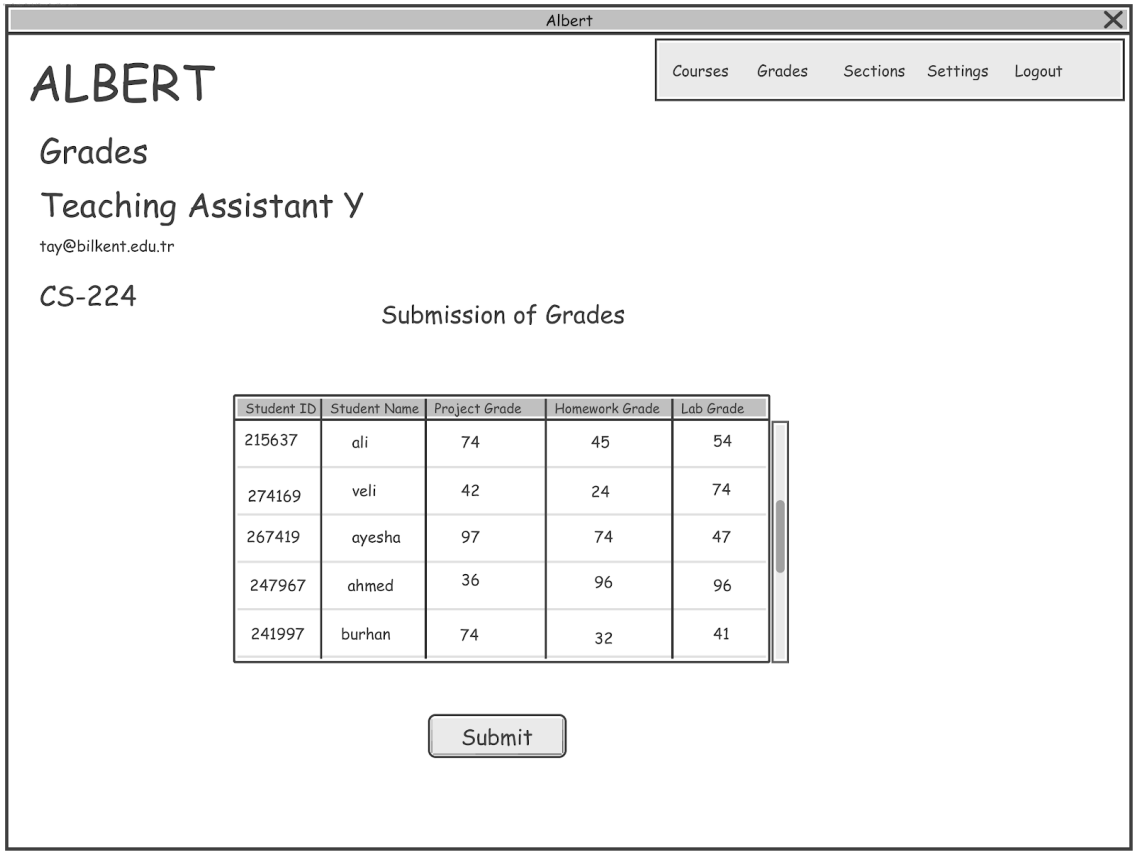
//This query selects exams that are graded by instructor

SELECT grades

FROM Grading natural join grade_exams

WHERE type = ‘exam’

5.5. Teaching Assistant Grading (labs, homeworks, projects)



Inputs: s-ID, s-name, grades

This is the page where the instructor will be able to enter the midterm and final exam grades for each student.

SQL statement:

INSERT INTO grading

VALUES (s-ID, s-name, grades);

// This query selects only those TA's which are assisting instructors and not particularly doing research work.

SELECT T.ta-ID, T.ta-name

FROM has T, Teaching Assistant X

WHERE T.ta-ID = “<input>” and T.ta-ID = X.ta-ID;

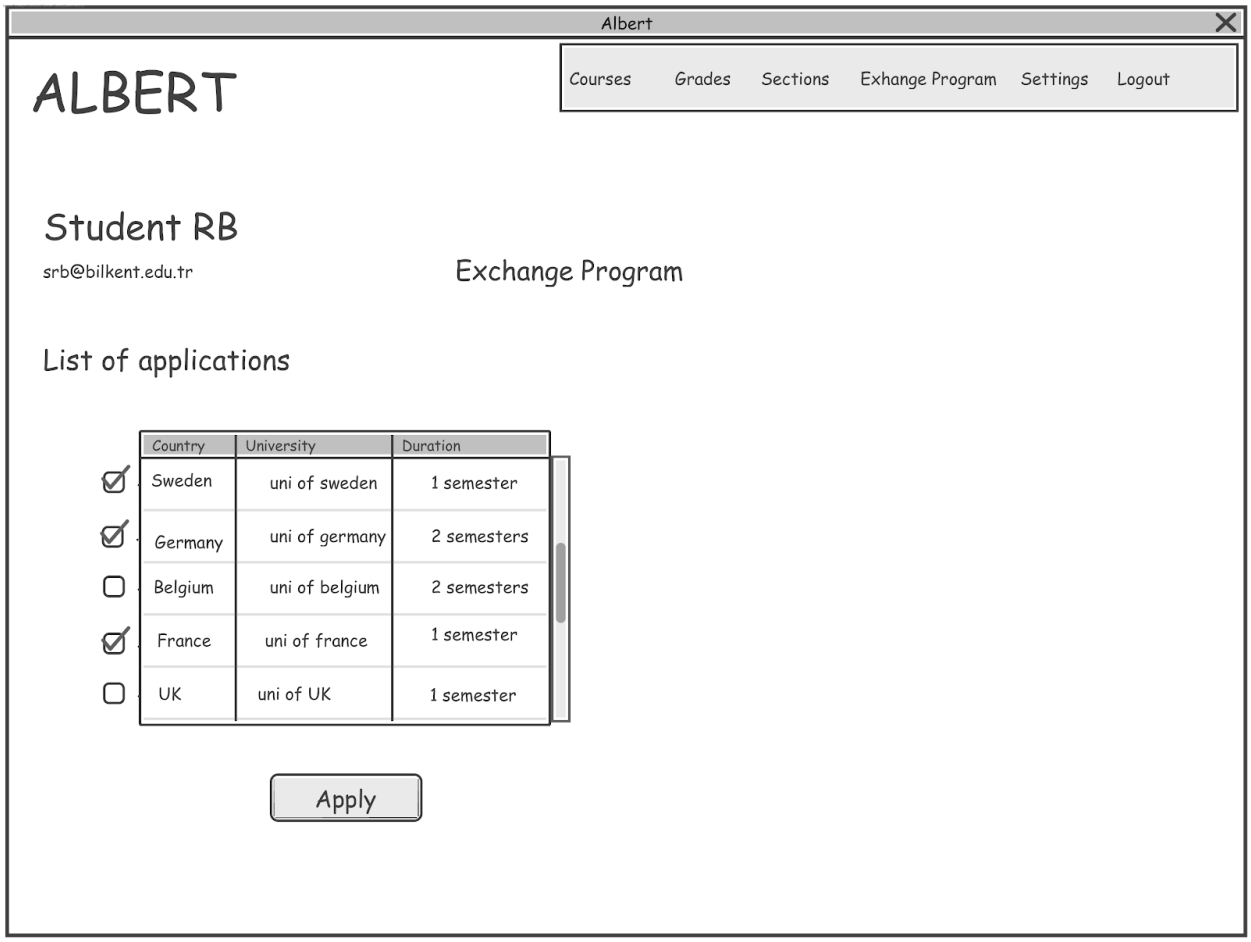
//This query selects assignments graded by TA’s

SELECT grades

FROM Grading natural join grade_assignments

WHERE type = ‘labs’ or ‘projects’ or ‘homeworks’

5.6. Exchange Program Application



Inputs: s-ID, country, uni-name, duration

This is the page where the student will be able to apply to exchange programs in different countries by checking all those options where he/she wishes to apply

SQL statement:

INSERT INTO Exchange Program

VALUES (s-ID, country, uni-name, duration);

UPDATE applies

Set country = <input>

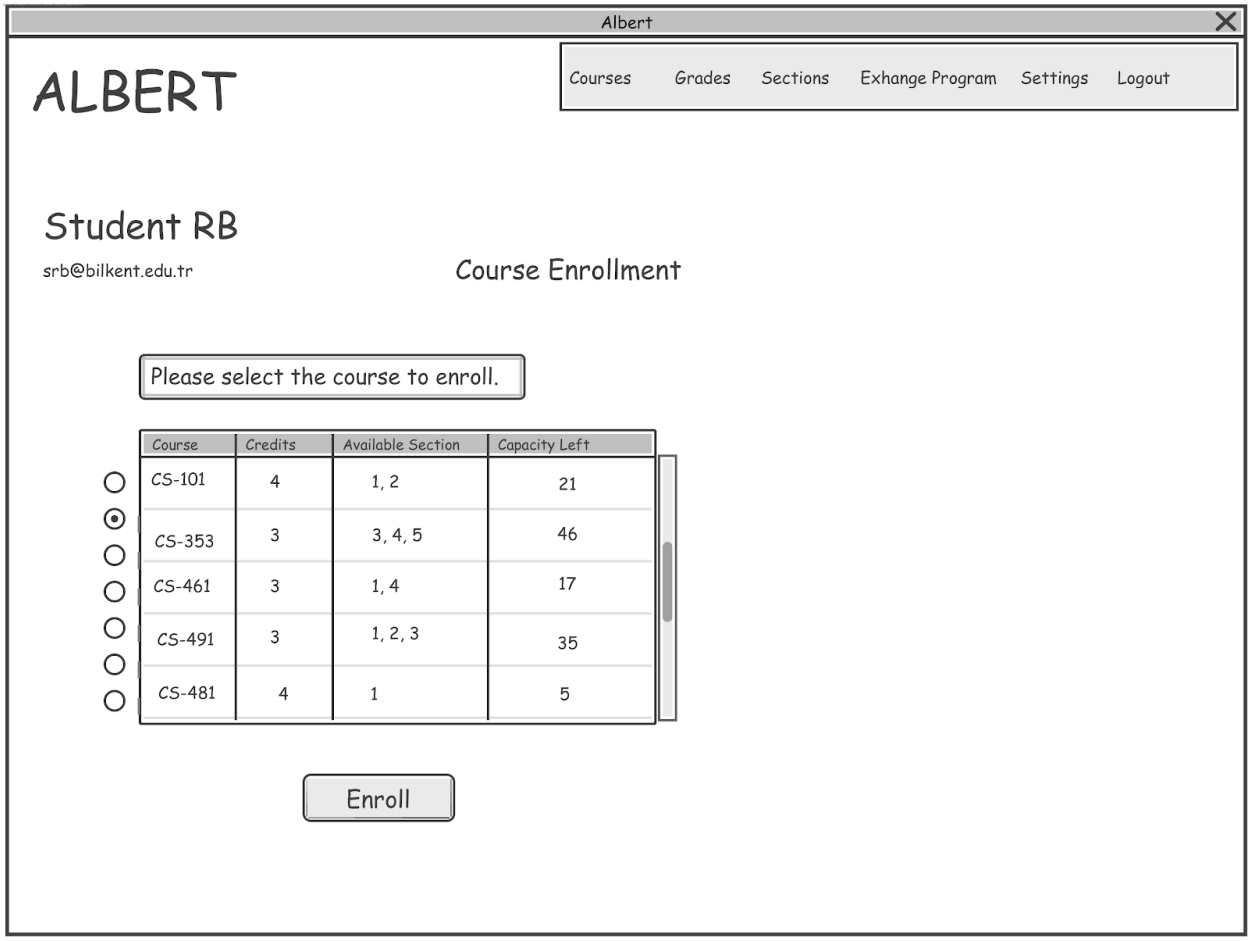
Set uni-name = <input>

Set duration = <input>

Set s-ID = <input>

WHERE s-gpa >= ‘2.5’

5.7. Course Enrollment Page



Inputs: c-ID, credit, sec-ID, capacity

This is the page where the student will be able to enroll for courses according to the space left in the various sections where there is still some quota available.

SQL statement:

INSERT INTO enrolls in

VALUES (c-ID, credit, sec-ID, capacity);

//Display list of courses for students who have enrolled for a particular course

SELECT C.c-name

FROM course C

WHERE (SELECT count(*)

FROM enrolls_in E

WHERE E.c-ID = C.c-ID

UPDATE applies

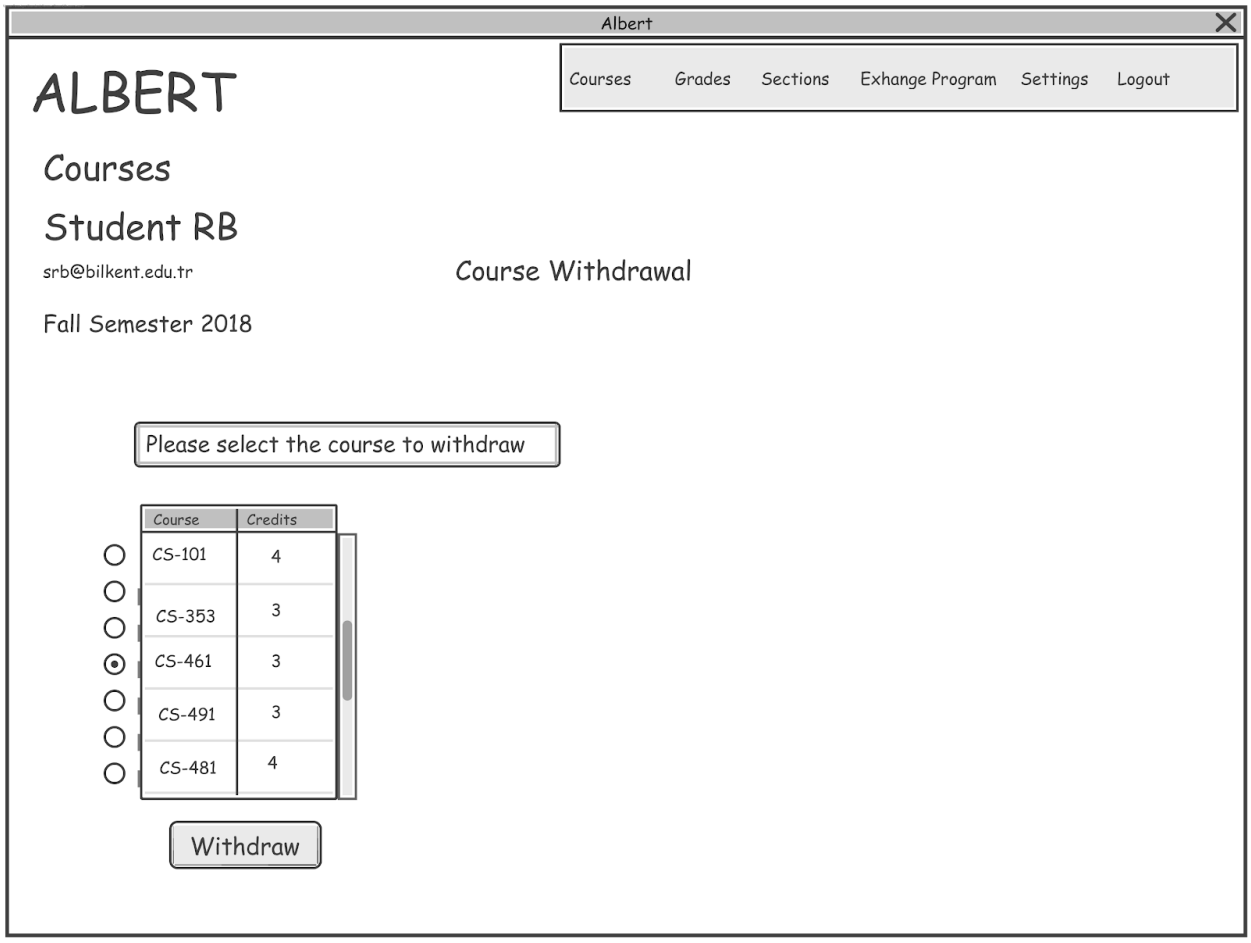
Set country = <input>

Set uni-name = <input>

Set duration = <input>

Set s-ID = <input>;

5.8. Course Withdrawal Page



Inputs: c-ID, credit

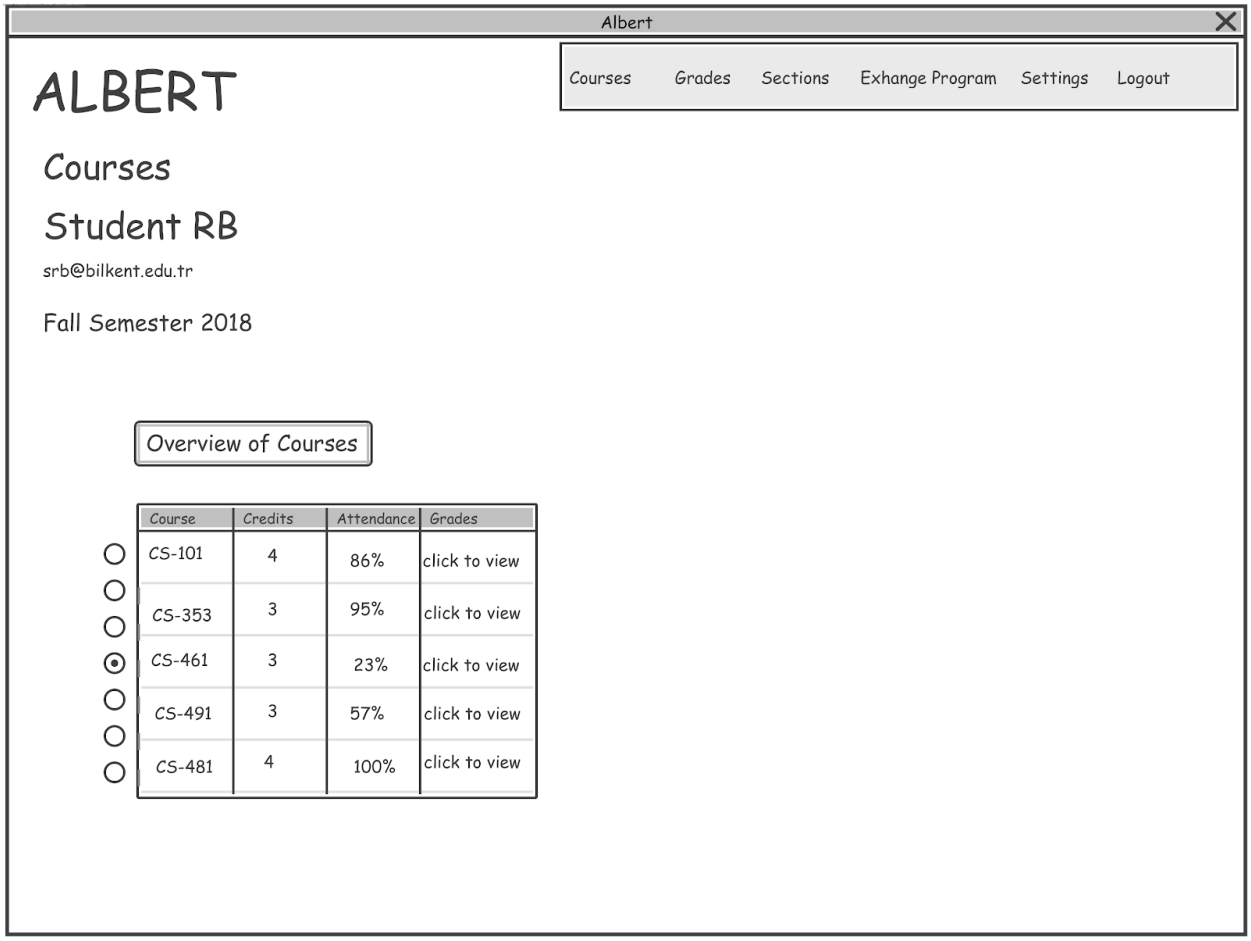
This is the page where the student will be able to withdraw from a course and will only be allowed to withdraw from 1 course per semester.

SQL statement:

DELETE FROM enrolls in

VALUES (c-ID);

5.9. Student Home Page



Inputs: c-id

This is the page where the student will be able to view his attendance record and also for each course can view his/her grades which will open a new window where the student will be able to view grades for each component of that course

SQL statement:

SELECT (*)

FROM courses natural join enrolls_in

6. Advanced Database Components

6.1. Views

- The following is a view for Teaching Assistants to see Students since they don't need to learn their cumulative GPA and phone numbers:

```
Create view course_student as
    Select s-ID, s-name, s-email
    From Student
```

- The following is a view for Students and Teaching Assistant to see Instructors since they shouldn't see the salary of the instructors:

```
Create view course_instructor as
    Select I-ID, I-name, I-email
    From Instructor
```

  Similar to Instructors, students shouldn't need to see the TA's salary as well. They can reach the remaining information via the following view:

```
Create view course_assistant as
    Select ta-ID, ta_name, ta_email
    From Teaching Assistant
```

6.2. Triggers

- Before students apply to Exchange Programs, their CGPA is checked to be greater than 2.5
- Before students enroll in a course, sections of the course is checked if it's capacity is full or not.

6.3. Reports

- The following function gets the number of students in each section, and sorts them according to courses.

  SELECT c-ID, sec-ID, COUNT s-ID
  FROM gets natural join enrolls_in
  GROUP BY c-ID

- The following functions reports the average grade of each assignment for an instructor, sorted in assignment number.

  SELECT AVG(grades)
  FROM Grading join Instructor

- Gets the number of students in every section, and arranges them in ascending order

  SELECT COUNT AS.username
  FROM account AS join Student S join Section SE
  GROUP BY sec-name
  ORDER BY ASC;

6.4. Constraints

- Users can not change their ID numbers.
- Users cannot change information other than their password once they have signed up.
- Students can not edit their grades.
- Each ID has to be unique.
- Instructors cannot offer more than 3 courses.

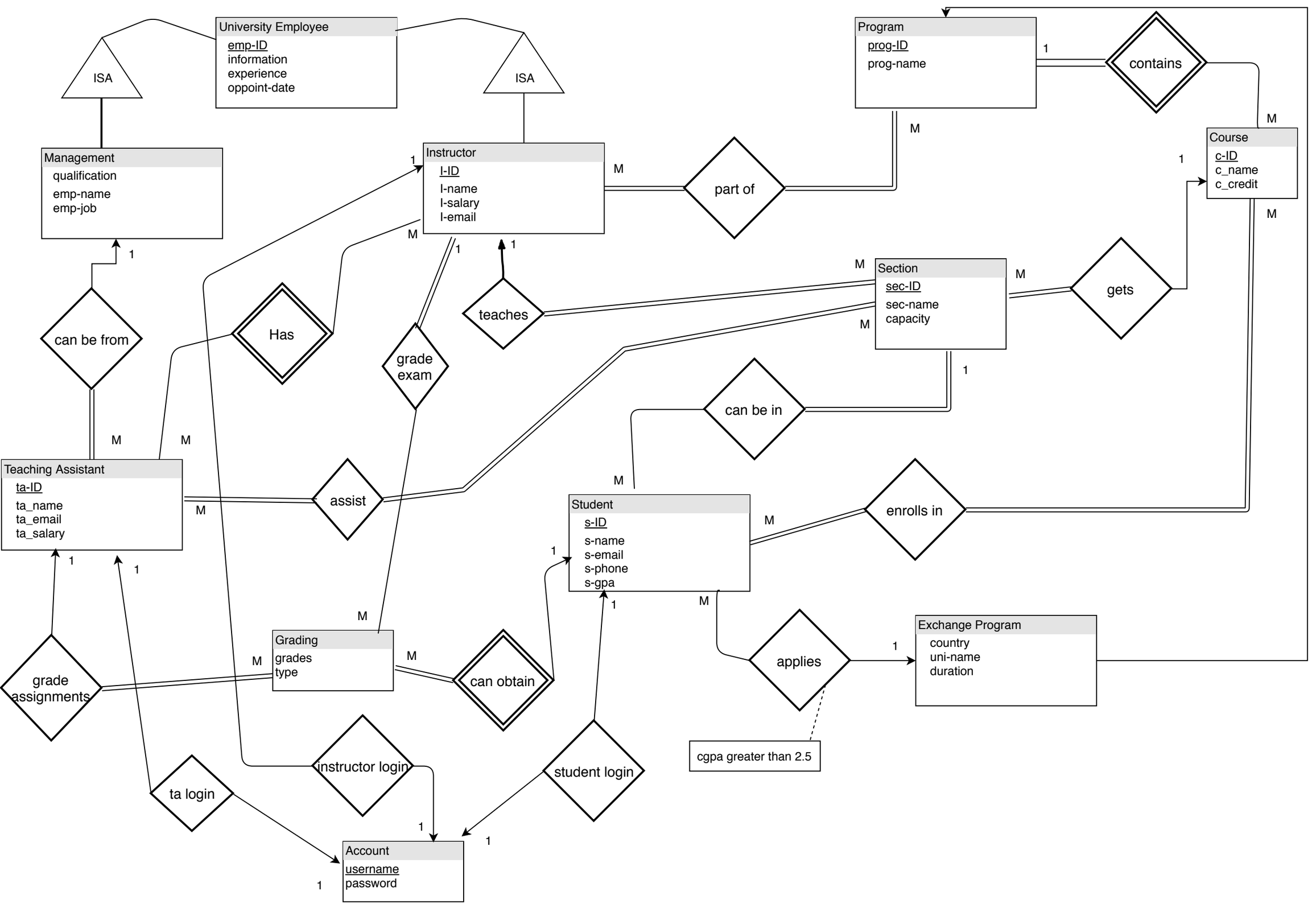
- Maximum number of enrolled courses by students will be 6 and minimum will be 5.

6.5. Stored Procedures

Stored procedures will create a default number of assignments and sections for each course for each course. Grade of assignments and capacity of sections will be adjusted by teaching assistants and instructors respectively.

7. Implementation Plan

We plan to use MySQL to implement the database for this project. PHP and Javascript will be used for the user interface, the communication between computers, and the rest of the application logic.


University Employee
emp-ID
information
experience
oppoint-date
ISA
ISA
Management
qualification
emp-name
emp-job
Instructor
I-ID
I-name
I-salary
I-email
part of
Program
prog-ID
prog-name
contains
Course
c-ID
c_name
c_credit
Section
sec-ID
sec-name
capacity
gets
teaches
Has
grade exam
can be from
can be in
Teaching Assistant
ta-ID
ta_name
ta_email
ta_salary
assist
Student
s-ID
s-name
s-email
s-phone
s-gpa
enrolls in
Grading
grades
type
grade assignments
can obtain
applies
Exchange Program
country
uni-name
duration
cgpa greater than 2.5
instructor login
student login
ta login
Account
username
password
1
M