GoAbroad

Prepared by

Salmane Dazine

Hanane Mohaouchane

Houda Ouhmad

Supervised by

Dr. Assem Nasser

11/12/2018

Contents

I. Introduction

A. Project Description and Objectives

As students who went through the difficult and confusing procedures of applying to study abroad programs, we thought of creating a tool that combines all kinds of opportunities brought by the university's Office of International Programs (OIP) under one portal. This will give the students an idea about all their study abroad options in a simple and less time-consuming way without having to check several emails and separate links sent by the OIP.

The GoAbroad project consists of creating a web-based application that facilitates for Al Akhawayn University (AUI) students the application procedure for exchange programs, and a desktop application that can be used by the OIP, to receive and manage the students' applications.

AUI students will be able to search all universities, create an account, enter their personal and academic information (e.g. their full names, GPAs, number of credits, semester, year, and so on), upload their transcripts, and apply for three universities based on their eligibility and preferences. The OIP will be able to see the list of applications received, and approve or reject them, all through the desktop application. Finally, the students will be notified by the result of their applications through our web-based application.

B. Project Scope

In this section, we will present the scope of requirements that the system needs to fulfill.

We spoke to the OIP to get an idea of their requirements and business rules. And, we checked similar purpose platforms like Erasmus and ISEP to complete the requirements gathering. We have identified the need for two different applications:

1. **A web-based application** that will be accessed by students to view and apply to universities.
2. **A desktop application** that will be managed by an administrator (the OIP) to manage the universities and the students' applications.

The web-based application which is accessed by students should cover the following:

- The user (student) is able to view the universities offered and the courses that they offer.
- The user (student) is able to login to the website.
- The user (student) is able to apply to a university.
- The user (student) is able to apply to a maximum of 3 universities.
- The user (student) is able to view the progress of his application.
- The user (student) is able to delete an application and replace it with another before the deadline.
- The user (student) is able to view the list of his applications.

The desktop application which is accessed by the administrator (the OIP) should cover the following:

- The user (admin) is able to login to the application.
- The user (admin) is able to view the student information and the universities each student has applied to.
- The user (admin) is able to accept or refuse the application of a student.
- The user (admin) is able to add or delete universities or update the information on universities.
- The user (admin) is able to assign each student to 0 or 1 university.

C. Project Plan

1. Dates and Tasks

Week 1: Discussion about the project's idea and purpose.

Week 2: Planning the project's tasks over the semester.

Week 3: Meeting with the Office of International Programs in AUI and specifying the requirements of the project.

23 September: Submission of the project proposal.

Week 4,5: Converting the requirements to a detailed website, desktop, and database design.

Week 6: Building an entity-relationship diagram for the software.

Week 7,8: Converting the design to functions to be implemented.

28 October: Submission of the detailed mid report.

Week 9,10,11: Implementing the database tables, web interface, and desktop application.

18 November: Submission of the project implementation progress.

Week 12,13: Documenting the design and implementation, and preparing the project's presentation.

5 December: Submission of the final report.

6 December: Project presentation and demo to the OIP.

2. Responsibilities

Work will be distributed evenly among the team members throughout all the deliverables of the project, and assistance will be offered and received by all the team members in case of any need. However, each of us will be accorded a task to work on when it comes to the design and implementation of the applications or database.

Salmane Dazine will take care of designing the ERD and creating and managing the database system.

Hanane Mouhaouchane will take care of designing and implementing the website and updating the database system.

Houda Ouhmad will take care of designing and implementing the desktop application and updating the database system.

3. Procedures

For us to meet all the requirements of our projects within their deadlines we will make sure to start enhancing our knowledge about the programming languages needed to implement our required functionalities in an early time of the semester. This will allow us to deal with programming errors whenever we encounter them, have enough time to use the expertise of our supervisor in case of any inquiry, and thereby build the best version possible of this project.

Among the technologies we will start getting familiar with there is SQL and SQL Server for the database system, HTML, CSS, C#, ASP.NET, and Visual Studio for the web interface and the desktop application.

II. Design

A. Conceptual design

1. Requirements Specification

The first step of the design process is the requirements specification where we will go into more detail about what functions our system must contain and the different data that it will need.

The data needed for this application can be divided into 5 categories: Student account, Admin Account, University, Application information, and Review information. The following list specifies the different data in each category:

- **Student account:**

- Student ID
- Password
- First Name
- Last Name
- GPA
- Number of credits
- School
- Major

- **Admin account:**

- Username
- Password

- **University:**

- Name
- Location
- Description
- Min GPA
- Courses
- Website

- Period of program
- Additional requirements

- **Application:**

- Motivation
- Status

- **Review Information:**

- Rating
- Comment text

The functions that the system will need to perform can be divided into functions for the web-based application, and functions for the desktop application.

1.1. Web-based Application Specification:

Function	Description	Input	Output	Process
Sign up	The student is able to create an account by entering his ID, a confirmation email with a code is then sent to his university email account, once he enters this code he is prompted to make a password for his account	ID Code Password	If successful, the student is redirected to a page in order to update his personal information	Each new user is added to the table of users while the primary key is the ID. The password is hashed before getting stored in the database

Update personal info	The student is prompted for his First name, Last name, GPA, and other information	First name Last name GPA Transcript Number of credits School Major	Redirect to profile page	The columns of the student table are filled
Log in	The student enters his ID and password to login to his account	ID Password	If successful, the user is redirected to the home page with the addition of a toolbar. If unsuccessful, an error message is displayed	The student account database is checked for a matching ID and password after hashing the password
Log out	The student logs out of his account and is redirected to the home page	None	The user is redirected to the homepage	
Change password	The student is able to modify his password	Old password New Password	Confirmation message if successful. Error message if unsuccessful	Changing the column password in the student account table after hashing

Search	Search for a university by name	University name	University data	Lookup the name of the university in the university table and display the data of that university
Apply	Apply to a university, the student must write why he chose this university, the student can only apply if he meets the conditions for the application (for example minimum GPA)	Motivation	Message of success or failure of application with explanation	Add a relationship between the university and the student, this relationship is added to the application table
View universities	List the universities and their data	None	List of universities	Display the table of universities
Cancel application	The student can cancel his application to a university	University Name		Remove the university from the list of applications of the student

View application status	The student can view the progress of his application: submitted, rejected, accepted		Status of the application	Retrieve the status information from the application table
Rate and comment	The student is able to rate and comment on a university	Comment text and ratings		The comment and rating is added to the list of reviews of the university
View University page	The student is able to view the data of the university including other student's ratings and comments		The university data	The university data is retrieved from the University table
View application list	The student is able to view the list of his applications		The list of the student's applications	The application information is retrieved from the applications table

1.2. Desktop Application Specifications:

Function	Description	Input	Output	Process
Login	The admin enters his username and password to log in	Username Password	If successful, the user is redirected to the admin page If not, an error message is displayed	The student account database is checked for a matching ID and password after hashing the password
Logout	The admin can log out of his account	None	The admin is logged out, the login page is displayed	
Change password	The admin is able to modify his password	Old password New Password	Confirmation message if successful. Error message if unsuccessful	Changing the column password in the admin account table after hashing
View student information	The admin is able to view the information of a specific student	Student ID	Student data (First Name, Last Name, GPA…)	Retrieving the student information from the student table

Add university	The admin is able to add a university to the list of offered universities	University Name Location Description Min GPA Courses Website Period of program Additional requirements		Add the university to the university table
Delete university	The admin is able to delete a university from the list of offered universities	University Name		Delete the university from the university table
View universities	The admin is able to view the list of universities		List of universities	Retrieve the list of universities from the university table
View applications	View the list of students who applied to a university		List of students	Retrieve the list of applications to a university from the application table

Contact student	Send a message to a student	Student ID Message text		
Approve application	Approve one application for a student, the student's other applications are automatically canceled	Student ID University		The application status of a student to a university is changed to accepted, the other applications are set to cancel
Reject all applications	Reject all of the student's applications	Student Id		The application status on all of the student's applications are changed to rejected

2. Business Rules, Entities, Relationships

2.1. Business Rules

This program is a product of communications between clients willing to apply for international programs (AUI students), administrators managing the programs (employees of the OIP), and designers working on combining the requirements of each party into the design of the program. We, the designers, are well aware of the business processes undertaken by the OIP in their agreements with other universities and the nature, role, and scope of data stored and manipulated in this program. Thereby, we developed appropriate relationships between the entities and appropriate constraints for certain attributes.

Business rules of this program include the following points:

- Students can apply to zero or many program types in the same semester.
- Students can apply to a maximum of 3 programs of the same program type.
- Students must list the priority of each program they apply to.
- Students can apply to the same university figuring in different programs.
- Employees of the OIP shall be able to view the students' personal information and application information to chosen programs.
- Employees rank applicants based on students' GPA and credits.
- Employees must assign zero or one program from each program type for each student.
- Students can confirm or deny their applications after receiving an "Accepted" status.
- Students must not apply for in host university's website until they confirm their application.

2.2. Entities and Relationships

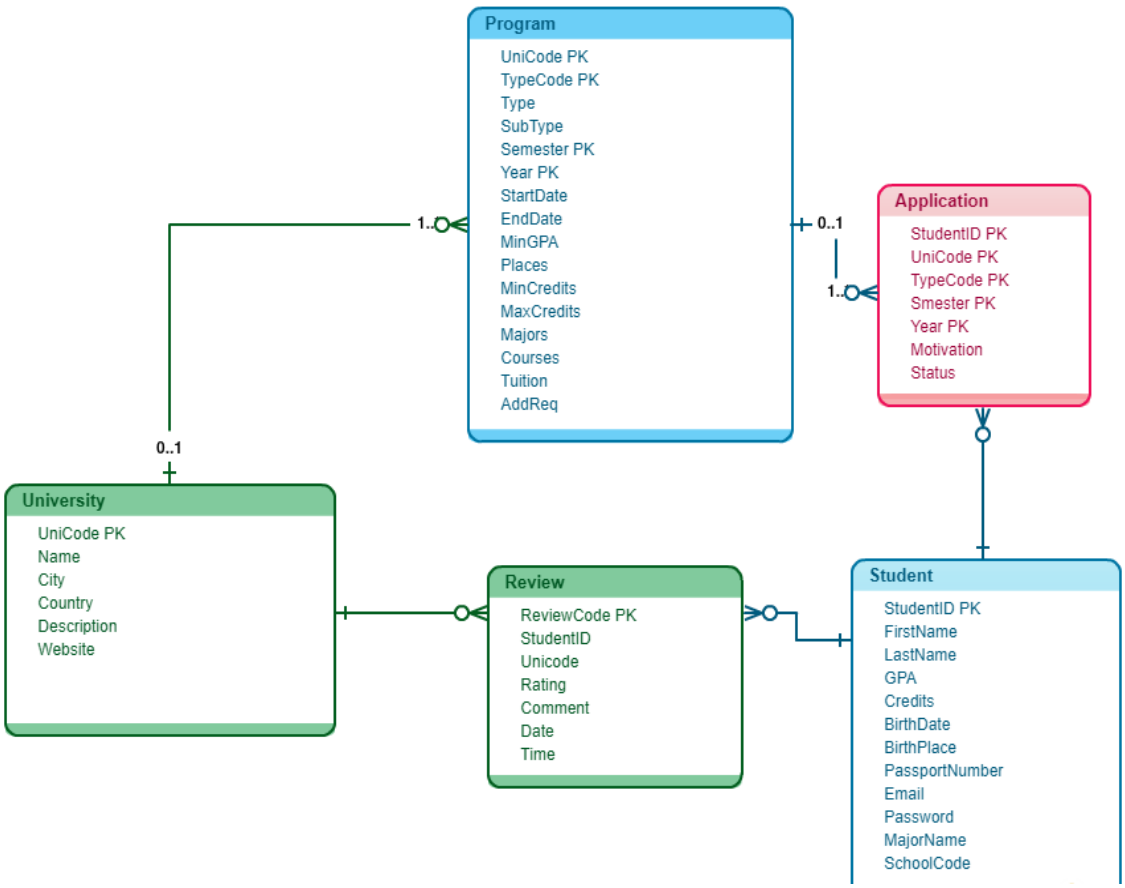
Our model consists of 10 entities with different relations between them:

- **Student:** contains the profile information of the student. This entity has a 1 to 1 relation with the entity Student Account because a student can only have one account. It also has a 1 to many relation with Review because a student can write many reviews. And finally a many to many relation with the entity Program because a student can apply to many programs and a program can be applied to by many students. This relation is simplified as a one to many relation to the bridge table Application.
- **Student Account:** contains the login data such as the email and the password. It has a one to one relation with the entity Student.
- **University:** contains information about the different available universities. It has a one to many relation to the entity Review because a university can have many reviews and a review is associated with one university. It also has a one to many relation with the entity Program since a university can offer many exchange or study abroad programs across different semesters.
- **Program:** contains information about the programs offered by each university including the semester and year that the program is offered among other attributes. It has a one to many relation with the entity University as discussed previously. It also

has a one to many relation with the entity Application because a program can have many applications.

- **Application:** bridge entity that connects students to the program they applied to. Contains additional information about the application such as the motivation and the status of the application. It has a one to many relation with Student and a one to many relation with Program.
- **Program Type:** contains information about the types of programs available to students such as bilateral exchange, study abroad… It has a one to many relation with the entity Program because many programs can belong to the same program type.
- **Major:** contains information about the different majors that students can belong to. It has a one to many relation with the entity Student since many students can belong to the same major.
- **School:** contains the different schools that majors can majors are provided by. It has a one to many relation with the entity Major.

3. Initial ER Model



4. Table Normalization

All the tables in the database were designed to be normalized following all the normal forms:

- **The first normal form**
  All attributes depend on the primary key.

- **The second normal form**
  All attributes depend on the whole key. There are no partial dependencies.

- **The third normal form**
  All attributes depend on nothing but the key. There are no transitive dependencies.

- **The Boyce-Codd normal form**
  No prime attribute depends on a non-prime attribute.

- **The fourth normal form**
  No independent multi-valued dependencies.

At first, the table Program was in the second normal form. We have designed it to include the program type and subtype as attributes. However, the subtype determines the type so there is a transitive dependency. We then split the table into two tables, one for the program and one for the program type.

5. Model Verification

In this section, we will define the processes and transaction steps for each of our requirements in order to verify our data model.

For the web-based application:

- Sign-up process: The sign-up process will be an insert query to the table Student. However, it might be more helpful to split the table student into Student and Student Account.
- Update personal info: This is an update query to the table Student where we can update the attributes that the student wishes to change.
- Log in: This is a procedure that will include a select query that tries to select a student with the ID and password to match the entered ID and password. If this select statement does not return anything, then the credentials are invalid.
- Change password: an update query to the table Student.
- Search for university: select from the University table where the name is equal to the given name.
- Apply: This is an insert query to the table Application with the appropriate student ID and university code, program type and subtype, semester, and year.
- View Universities: This is a select statement from the table University.
- Cancel application: This is a delete statement from the table Application given the student and the program that the student wishes to cancel his application for.
- View application status: This is a select statement where we select the status of the concerned application.
- Rate and comment: This is an insert statement to the table review which contains the attributes rate and comment and has a relationship with the tables University and Student.
- View university page: This is a select statement from the joint tables University and Review so that the student can view the university information as well as view the reviews of that university.
- View application list: This is a select statement from the table Application with the appropriate student ID.

For the desktop application:

- Login: There is a need for a table to store the username and the password of the OIP. This process will have a select statement which tries to select a username and password that match the credentials that were entered. If it returns null then the credentials are invalid.
- Change password: this is an update to the Employee Account table

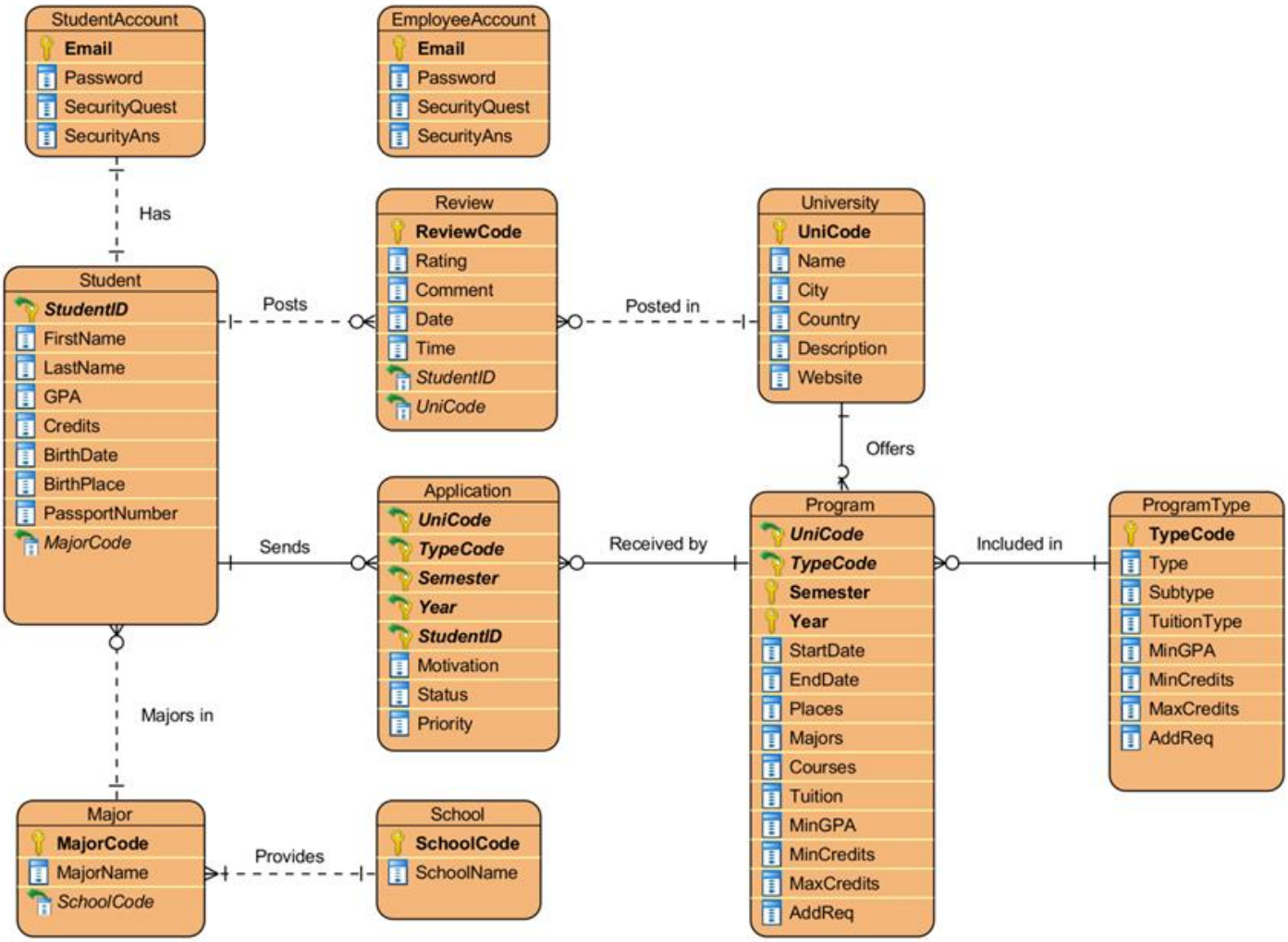
- View student information: this is a select statement from the Table student given the appropriate student ID or name.
- Add university: this is an insert statement to the tables university and program because the program is where the period and requirements are stored.
- Delete university: this is a delete statement to the table program in order to indicate that a program is no longer offered. We can also give the option of entirely deleting a university if the OIP checks that the university will no longer offer any programs.
- View Universities: this is a select statement from the table University.
- View application: this is a select statement from the table Application given the university code.
- Approve application/ Reject application: this an update statement to the table Application where we change the status to either Accepted or Rejected.

6. Final ER Model

6.1. Modifications

- We added the entity of StudentAccount to store the login information needed for each of the clients.
- We added the EmployeeAccount table.
- We added BirthDate, BirthPlace, and PassportNumber as attributes for the entity of Student, as they are necessary information for international programs' applications.
- We moved the attributes School and Major to new entities instead of being attributes in the Student entity.
- We added the entity of ProgramType, as there can be different types of programs that specify their own characteristics and requirements.
- We added City and Country as attributes for the entity of University, and UniCode as a primary key for it.
- We added Date and Time for the entity of Review to specify it more.
- We added Priority as an attribute for the entity Application, so that students can list the priority of each program they apply to.

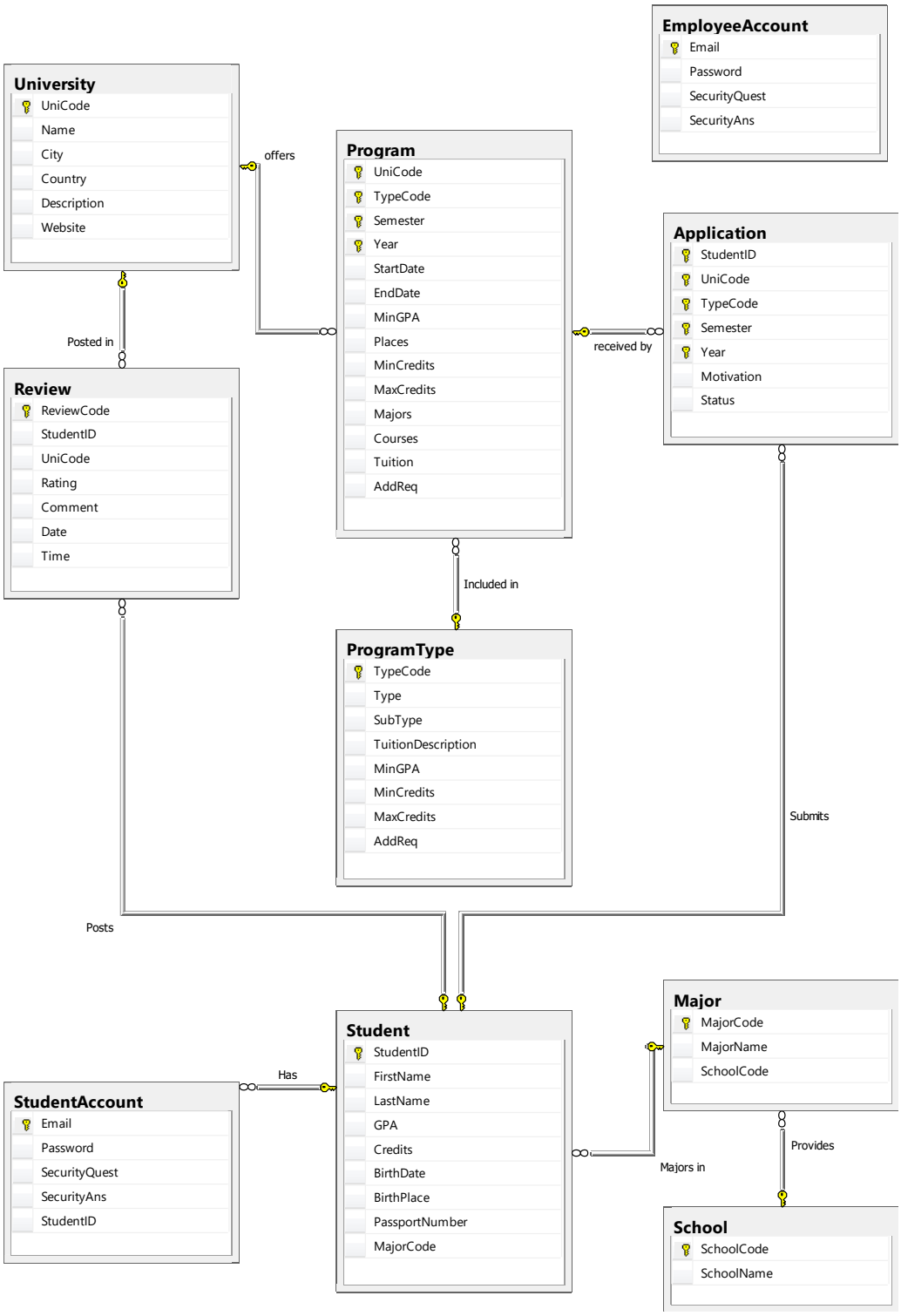
6.2. ERD


StudentAccount
Email
Password
SecurityQuest
SecurityAns
EmployeeAccount
Email
Password
SecurityQuest
SecurityAns
Has
Review
ReviewCode
Rating
Comment
Date
Time
StudentID
UniCode
University
UniCode
Name
City
Country
Description
Website
Student
StudentID
FirstName
LastName
GPA
Credits
BirthDate
BirthPlace
PassportNumber
MajorCode
Posts
Posted in
Offers
Application
UniCode
TypeCode
Semester
Year
StudentID
Motivation
Status
Priority
Program
UniCode
TypeCode
Semester
Year
StartDate
EndDate
Places
Majors
Courses
Tuition
MinGPA
MinCredits
MaxCredits
AddReq
ProgramType
TypeCode
Type
Subtype
TuitionType
MinGPA
MinCredits
MaxCredits
AddReq
Sends
Received by
Included in
Majors in
Major
MajorCode
MajorName
SchoolCode
Provides
School
SchoolCode
SchoolName

A. Logical design:

1. Database Diagram:

The tables we have defined correspond to the entities in the ERD and are described in the following database diagram.



2. Check Constraints

Check that the GPA is between 0.0 and 4.0

```
CREATE TABLE Student (
StudentID int PRIMARY KEY,
FirstName varchar(20) NOT NULL,
LastName varchar(30) NOT NULL,
GPA decimal(3,2) CHECK (0.0<=GPA AND GPA<=4.0)NOT NULL,
Credits int NOT NULL,
BirthDate date NOT NULL,
BirthPlace varchar(40) NOT NULL,
PassportNumber varchar(20) NOT NULL,
MajorCode varchar(10) NOT NULL,
FOREIGN KEY (MajorCode) REFERENCES Major ON UPDATE CASCADE ON DELETE NO ACTION,
);
```

Check that the application status is either accepted, rejected, or pending

```
CREATE TABLE Application (
StudentID int,
UniCode varchar(20),
TypeCode INTEGER,
Semester varchar(10),
Year int,
Motivation varchar(200),
Status varchar(20),
CONSTRAINT chk_Status CHECK (Status IN ('Accepted', 'Rejected', 'Pending')),
FOREIGN KEY (StudentID) REFERENCES Student (StudentID) ON UPDATE CASCADE ON DELETE NO ACTION,
FOREIGN KEY (UniCode,TypeCode,Semester,Year) REFERENCES Program ON UPDATE CASCADE ON DELETE CASCADE,
PRIMARY KEY(StudentID,UniCode,TypeCode,Semester,Year),
);
```

3. Views

A view to see past applications of a student.

```
CREATE VIEW vwPastApplication AS
SELECT StudentID,U.Name AS 'University',Semester, Year, Type,SubType, Motivation, Status
FROM University U INNER JOIN Application A
ON U.UniCode = A.Unicode
INNER JOIN ProgramType PT
ON A.TypeCode = PT.TypeCode
WHERE Status<>'Pending'
GO
```

View of the current applications of a student.

```sql
IF OBJECT_ID ('vwCurrentApplication' , 'V') IS NOT NULL
    DROP VIEW vwCurrentApplication;
    GO
CREATE VIEW vwCurrentApplication AS
SELECT StudentID,A.Unicode,U.Name AS 'University',Semester, Year,A.TypeCode, Type,SubType, Motivation, Status
FROM University U INNER JOIN Application A
ON U.UniCode = A.Unicode
INNER JOIN ProgramType PT
ON A.TypeCode = PT.TypeCode
WHERE Status='Pending'
GO
```

4. Indexes

Index created for the University name to make the search easier and faster.

The second index is created for the Program type because we often would like to search by program type.

```sql
CREATE INDEX Index_UniversityName on University(Name);
CREATE INDEX Index_ProgramType on ProgramType(Type);
GO
```

5. Stored Procedures

A stored procedure to add a review

```sql
IF (OBJECT_ID ('sp_AddReview') IS NOT NULL)
    DROP PROCEDURE sp_AddReview
GO
CREATE PROCEDURE sp_AddReview @StudentID int, @UniName varchar(40),@Rating int, @Comment varchar(200)
AS
BEGIN
    DECLARE @UniCode int, @Date date, @Time time;
    SET @UniCode = (SELECT UniCode FROM University WHERE Name = @UniName);
    SET @Date = (SELECT CONVERT (date, GETDATE()));
    SET @Time = (CONVERT (time, GETDATE()));
    INSERT INTO Review VALUES (@StudentID,@UniCode,@Rating,@Comment,@Date,@Time);
END;
GO
```

A stored procedure to search programs by name

```sql
IF (OBJECT_ID('sp_SearchProgramType') IS NOT NULL)
  DROP PROCEDURE sp_SearchProgramType
GO
CREATE PROCEDURE sp_SearchProgramType @ProgType varchar(40)
AS
BEGIN
    SELECT * FROM ProgramType WHERE Type = @ProgType;
END;
GO
```

A stored procedure for login

```sql
IF (OBJECT_ID ('Validate_User') IS NOT NULL)
    DROP PROCEDURE Validate_User
GO

CREATE PROCEDURE [dbo].[Validate_User]
        @Username NVARCHAR(20),
        @Password NVARCHAR(20)
AS
BEGIN
        SET NOCOUNT ON;
        DECLARE @UserId INT;

        SELECT @UserId = StudentID
        FROM StudentAccount WHERE StudentID = @Username AND Password = @Password

          IF @UserId IS NOT NULL
              BEGIN

               SELECT @UserId [UserId] -- User Valid

              END
          ELSE
              BEGIN
                  SELECT -1 -- User invalid.
              END
END;
GO
```

A stored procedure to insert the student and the student account in one statement, and it takes as argument the major name instead of the major code.

```sql
IF (OBJECT_ID('spInsertStudentAndAccount') IS NOT NULL)
  DROP PROCEDURE spInsertStudentAndAccount
GO
CREATE PROCEDURE spInsertStudentAndAccount @StudentId int,@FirstName varchar(20),
@LastName varchar(30),@GPA decimal(3,2),@Credits int,@BirthDate date,@BirthPlace varchar(40),
@PasportNumber varchar(20),@MajorName varchar(500),@Email varchar(500),@Password varchar(50)
AS
BEGIN
DECLARE @MajorCode varchar(10)
SET @MajorCode = (Select MajorCode FROM Major WHERE MajorName =@MajorName);
IF EXISTS (Select StudentID FROM Student WHERE StudentID = @StudentId)
  BEGIN
  SELECT -1
  END;
ELSE IF EXISTS (Select Email FROM StudentAccount WHERE Email = @Email)
  BEGIN
  SELECT -2
  END;
ELSE
  BEGIN
INSERT INTO Student VALUES (@StudentId,@FirstName,@LastName,@GPA,@Credits,@BirthDate,@BirthPlace,@PasportNumber,@MajorCode);
INSERT INTO StudentAccount VALUES (@Email,@Password,NULL,NULL,@StudentId);
  SELECT 0;
  END;
END;
GO
```

A stored procedure to search universities by name

```sql
IF (OBJECT_ID('sp_SearchUniversity') IS NOT NULL)
  DROP PROCEDURE sp_SearchUniversity
GO
CREATE PROCEDURE sp_SearchUniversity @UniName varchar(40)
AS
BEGIN
    SELECT * FROM University WHERE Name = @UniName;
END;
GO
```

A stored procedure to delete an application

```
IF (OBJECT_ID('sp_DeleteApplication')IS NOT NULL)
    DROP PROCEDURE sp_DeleteApplication
GO

CREATE PROCEDURE sp_DeleteApplication @StudentId int, @UniName varchar(40), @TypeName varchar(30),@SubType varchar(30),
@Semester varchar(10),@Year int
AS
BEGIN
DECLARE @UniCode int,@TypeCode int;
SET @UniCode = (SELECT UniCode FROM University WHERE Name = @UniName);
SET @TypeCode = (SELECT TypeCode FROM ProgramType WHERE Type= @TypeName AND SubType = @SubType);
DELETE FROM Application WHERE
UniCode = @UniCode
AND
TypeCode = @TypeCode
AND
StudentID = @StudentId
AND
Semester = @Semester
AND
Year = @Year
END
GO
```

6. Triggers

A trigger is created instead of Insert to check the number of applications per student, and to reinforce the constraint of no more than 3 applications per student. This trigger checks the number of application whenever an Insert is about to happen. If the number of applications per student is less than 3, then the Insert is performed otherwise the transaction commits.

```
CREATE TRIGGER tri_AddNewApplication ON Application INSTEAD OF INSERT
AS
BEGIN
DECLARE @app_count int, @studentID int, @UniCode int, @TypeCode int,
@Semester varchar(20),@Year int, @Motivation varchar(500), @Status varchar(30);
DECLARE app_cursor CURSOR FOR
    SELECT StudentID,UniCode,TypeCode,Semester,Year,Motivation,Status FROM inserted;

OPEN app_cursor;
FETCH NEXT FROM app_cursor INTO @studentID,@UniCode,@TypeCode,@Semester,@Year,@Motivation,@Status;
WHILE(@@FETCH_STATUS=0)
BEGIN
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
BEGIN TRANSACTION;
SET @app_count = (SELECT COUNT(*) AS 'ApplicationCOUNT'FROM Application A
                  WHERE A.StudentID = @studentID);
IF @app_count < 3
  BEGIN
  INSERT INTO Application VALUES (@studentID,@UniCode,@TypeCode,@Semester,@Year,@Motivation,@Status);
  END;
COMMIT TRANSACTION;
FETCH NEXT FROM app_cursor INTO @studentID,@UniCode,@TypeCode,@Semester,@Year,@Motivation,@Status;
END;
END;
```

B. Physical design:

DB and tables' storage space requirements are very low. It does not require that much space since the number of universities and the programs offered that we have used as a sample in this project is not that high. Right now the database is only filled with examples to test and present the functionality of the website and desktop application. Once the database is populated and filled with the universities and programs, the storage space needed will be higher.

Also, storage space requirements will increase as students keep signing up and entering their personal information. Therefore, our application should take into account the number of students that will need to sign up for the website. As the years go by, previous programs also keep getting stored in the database. This number could increase a lot after many years. Therefore, the database should discard information about programs from the very far past that are no longer useful in order to reduce the storage requirements.

III. Implementation

A. Transaction management issues:

As we have already mentioned, concurrency may happen if a student wants to apply for more than 3 universities. To solve this problem, we created a trigger instead of Insert to check the number of applications per student. If the number of applications per student is less than 3, then the Insert is performed otherwise the transaction commits.

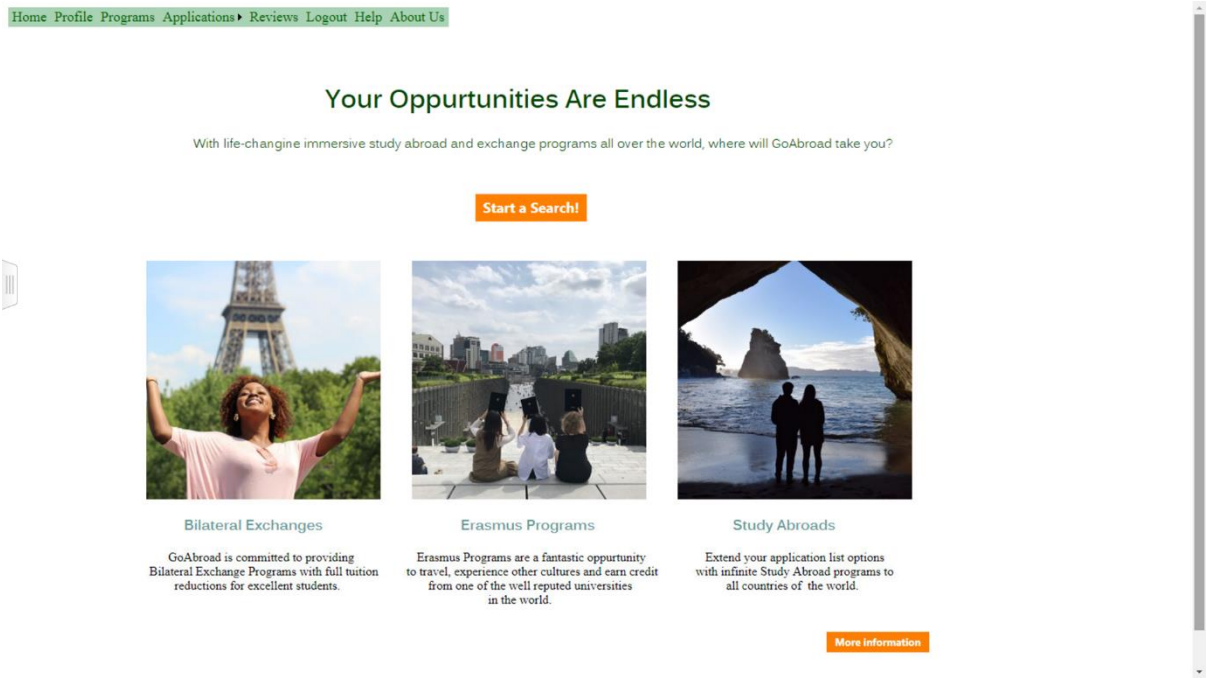
```sql
CREATE TRIGGER tri_AddNewApplication ON Application INSTEAD OF INSERT
AS
BEGIN
DECLARE @app_count int, @studentID int, @UniCode int, @TypeCode int,
@Semester varchar(20),@Year int, @Motivation varchar(500), @Status varchar(30);
DECLARE app_cursor CURSOR FOR
    SELECT StudentID,UniCode,TypeCode,Semester,Year,Motivation,Status FROM inserted;

OPEN app_cursor;
FETCH NEXT FROM app_cursor INTO @studentID,@UniCode,@TypeCode,@Semester,@Year,@Motivation,@Status;
WHILE(@@FETCH_STATUS=0)
BEGIN
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
BEGIN TRANSACTION;
SET @app_count = (SELECT COUNT(*) AS 'ApplicationCOUNT'FROM Application A
                    WHERE A.StudentID = @studentID);
IF @app_count < 3
  BEGIN
  INSERT INTO Application VALUES (@studentID,@UniCode,@TypeCode,@Semester,@Year,@Motivation,@Status);
  END;
COMMIT TRANSACTION;
FETCH NEXT FROM app_cursor INTO @studentID,@UniCode,@TypeCode,@Semester,@Year,@Motivation,@Status;
END;
END;
```

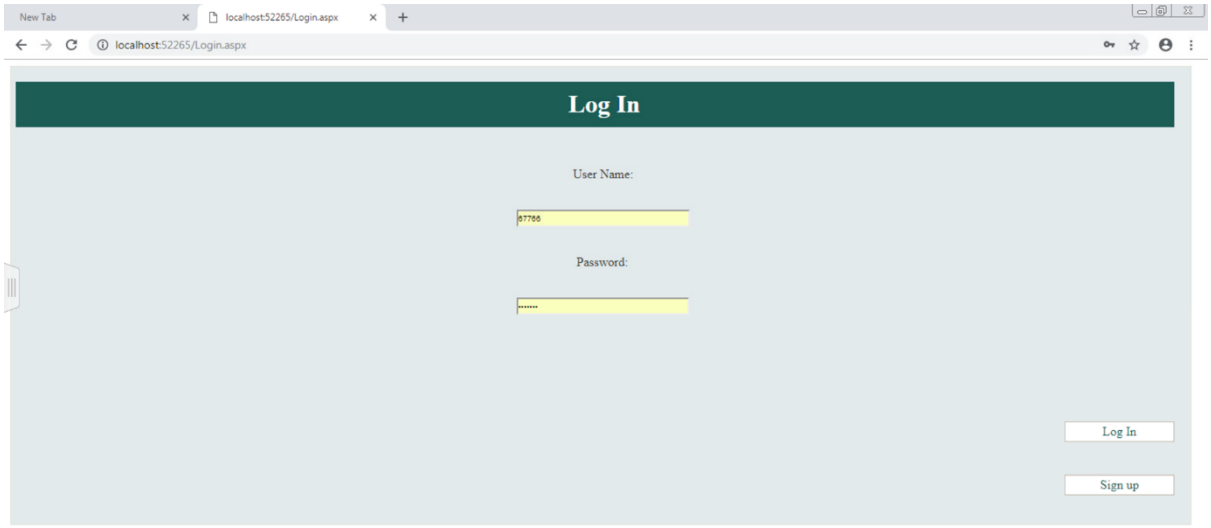
B. Application architecture:

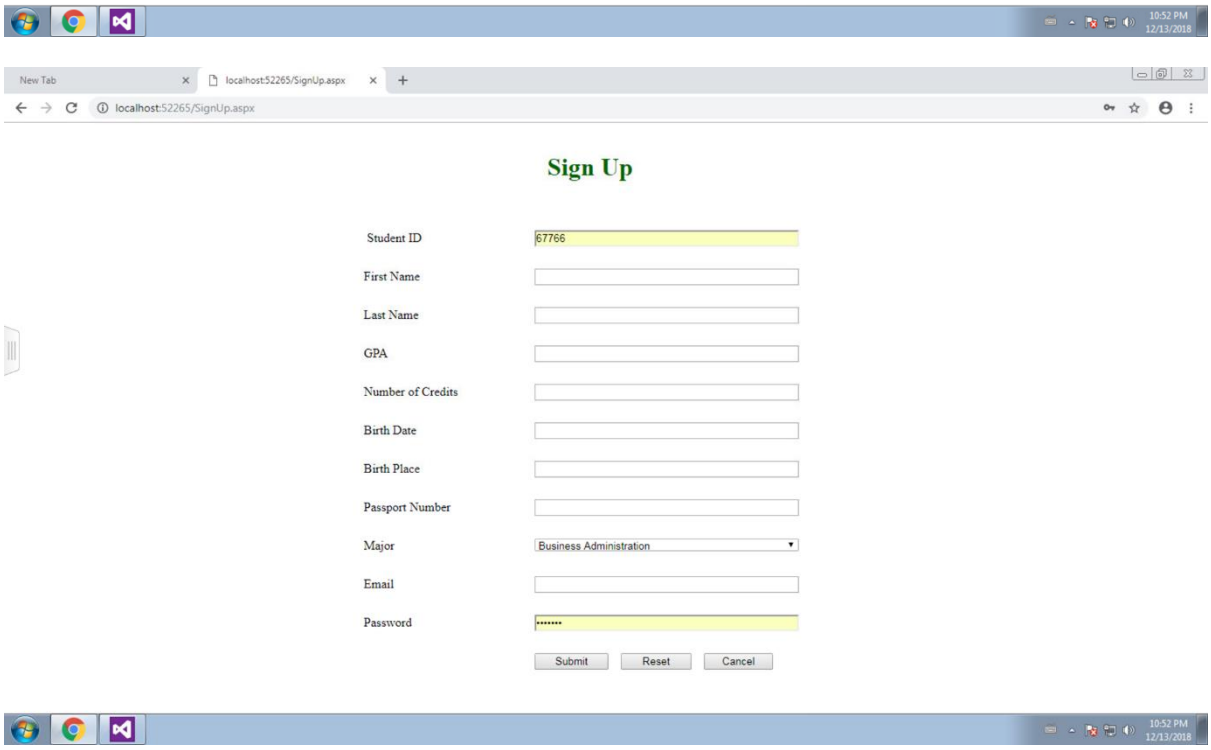
- We used the university's local SSEDB database for the creation and the population of our database, the tools used for this purpose are the language SQL along with database management system Microsoft SQL Server.
- We used C# along with ASP.NET to build our web-based and desktops applications in Visual Studio, after connecting them to our implemented database.

C. Application structure:

Our GUI consists of a simple user-friendly interface that contains a home page and a login page as shown in the images below.







IV. Testing and fine-tuning

We tested our applications with different inputs to test for any unhandled error.

While testing we found many errors that include the following:

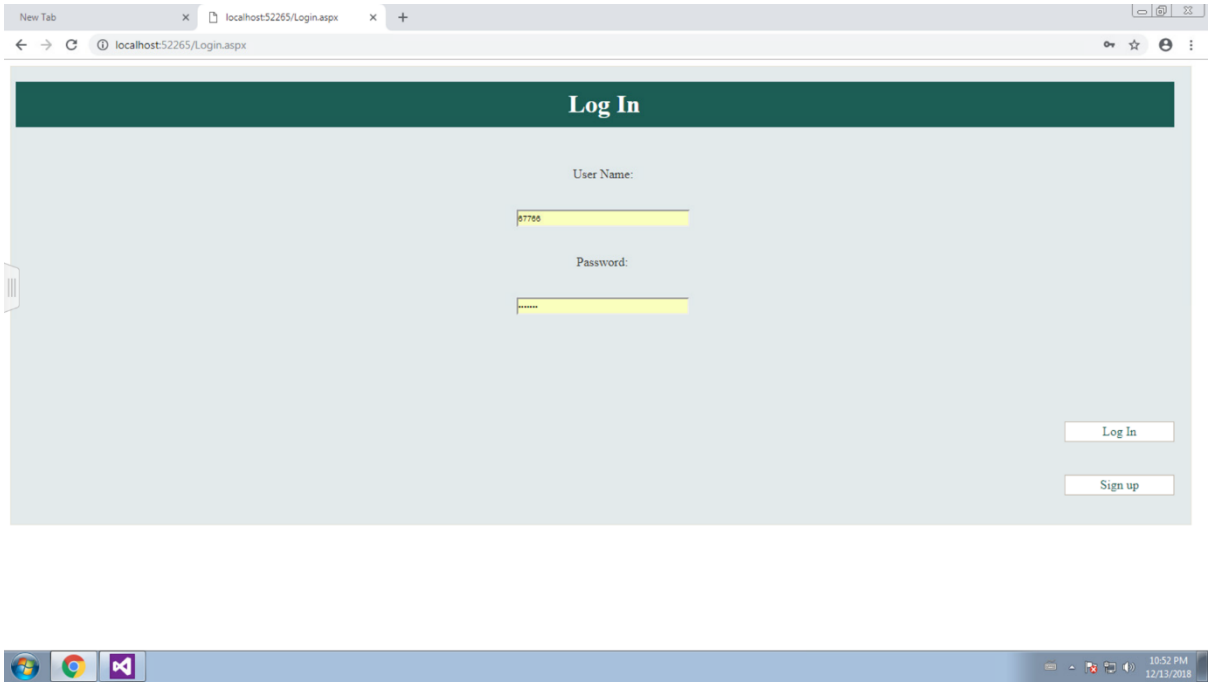
- Applying to the same program twice.
- Clicking the Apply button without selecting a program.
- Clicking the review button without selecting the university.

- Trying to sign up with an already existing student ID or email.

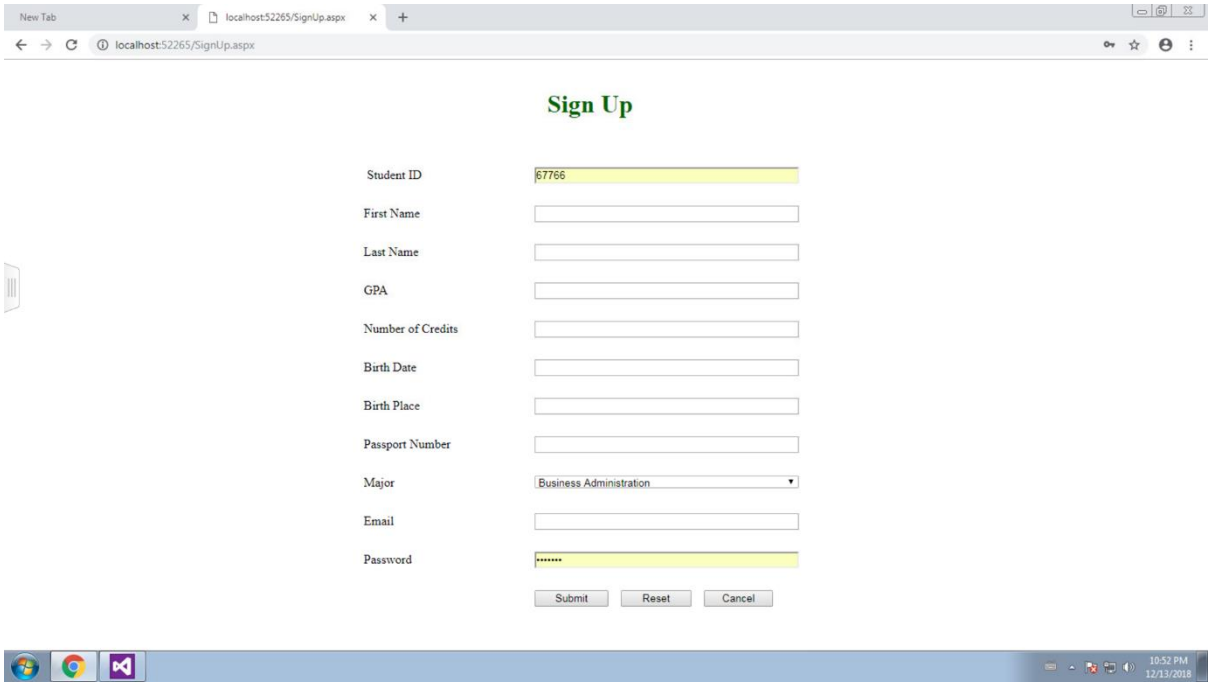
The errors were fixed by catching the exceptions and displaying an error message.

V. User manual:

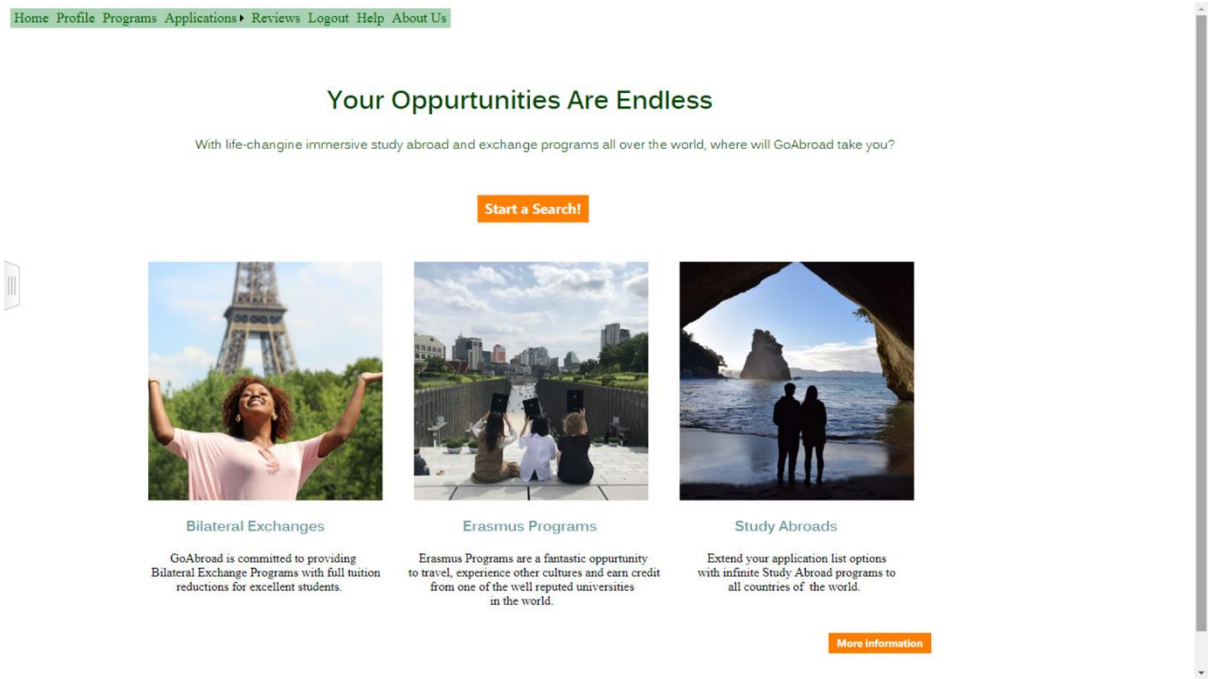
A. Web Application



Log in page: consists of Username and Password text areas to allow successful authentication. Log in directs us to the home page after filling the areas. The sign-up button directs us to a sign-up form.



Sign Up page: Sign up form to fill, while respecting the following rules: No Student ID repetition, No Email repetition, No empty text areas. Reset empties all the text areas. Cancel and Submit directs us at the login page.



The Home page: a Portal to all other windows. Start a Search button directs us to the Programs page. More information button directs us to the Help page.

Home Profile Programs Applications Reviews Logout Help About Us

My Profile

StudentID	67766
FirstName	Salmane
LastName	Dazine
GPA	3.15
Credits	85
BirthDate	8/7/1998 12:00:00 AM
BirthPlace	Casablanca
PassportNumber	JC1516309
MajorCode	CSC
Edit	

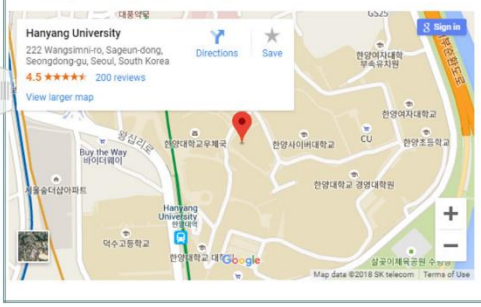
Profile page: consists of profile information and allows updating them.

Home Profile Programs Applications Reviews Logout Help About Us

Programs

University: hanyang university Search Apply

Name: **Hanyang University**
City: Seoul
Country: South Korea
Description: One of the well-reputed universities in South Korea. Includes English-taught courses from all majors
Website: http://www.hanyang.ac.kr/web/eng



	Semester	year	Type	SubType	StartDate	EndDate	MinGPA	Places	MinCredits	MaxCredits	Majors	Majors1	Courses	Tuition	AddReq
Select	Spring	2018	Bilateral Exchange	Partial Exchange	1/3/2018 12:00:00 AM	6/21/2018 12:00:00 AM	3.00	5	45	95	GenEd, BAIS, BACS, BBA, BSGE, BSHRD, BBA	GenEd, BAIS, BACS, BBA, BSGE, BSHRD, BBA	http://www.hanyangexchange.com/academics/syllabus/	20000	
Select	Summer	2018	Bilateral Exchange	Partial Exchange	7/1/2018 12:00:00 AM	7/31/2018 12:00:00 AM	3.00	4	45	90	BSCSC,BSGE, BSEMS,BAIS, BSHRD	BSCSC,BSGE, BSEMS,BAIS, BSHRD	http://www.hanyangsummer.com/seoul/courses/	30000	

	Semester	year	Type	SubType	StartDate	EndDate	MinGPA	Places	MinCredits	MaxCredits	Majors	Majors1	Courses	Tuition	AddReq
Select	Spring	2018	Bilateral Exchange	Partial Exchange	1/3/2018 12:00:00 AM	6/21/2018 12:00:00 AM	3.00	5	45	95	GenEd, BAIS, BACS, BBA, BSGE, BSHRD, BBA	GenEd, BAIS, BACS, BBA, BSGE, BSHRD, BBA	http://www.hanyangexchange.com/academics/syllabus/	20000	
Select	Summer	2018	Bilateral Exchange	Partial Exchange	7/1/2018 12:00:00 AM	7/31/2018 12:00:00 AM	3.00	4	45	90	BSCSC,BSGE, BSEMS,BAIS, BSHRD	BSCSC,BSGE, BSEMS,BAIS, BSHRD	http://www.hanyangsummer.com/seoul/courses/	30000	

Review:

FirstName: Salmane
LastName: Dazine
Rating: 7
Comment: Great Program! Awesome Country!
Date: 10/23/2018 12:00:00 AM
Time: 10:30:23

Review:

FirstName: Hanane
LastName: Mohaouchane
Rating: 5
Comment: One of the greatest experiences ever! Well recommended.
Date: 12/13/2018 12:00:00 AM
Time: 22:11:36.9430000

Review:

FirstName: Houda
LastName: Ouhmad
Rating: 4
Comment: Well recommended! Interesting courses
Date: 12/13/2018 12:00:00 AM
Time: 22:16:15.7070000

Programs page: the university's name must be written fully and correctly before clicking the search button. Results contain information about the selected university and the available programs to apply to and reviews written by students. To apply to a program, you must select it first.

Home Profile Programs Applications Reviews Logout Help About Us

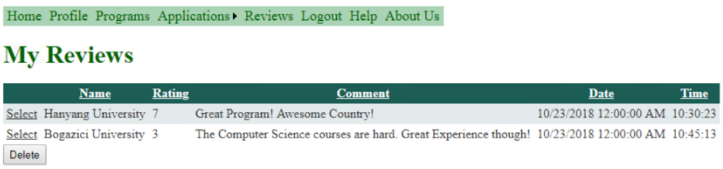
My Past Applications

	University	Semester	Year	Type	SubType	Motivation	Status
Select	Hanyang University	Spring	2018	Bilateral Exchange	Partial Exchange	I'm applying to this university in view of my interest in the Computer Graphics course offered, and also my intention in visiting on of the most beautiful countries in the world.	Accepted

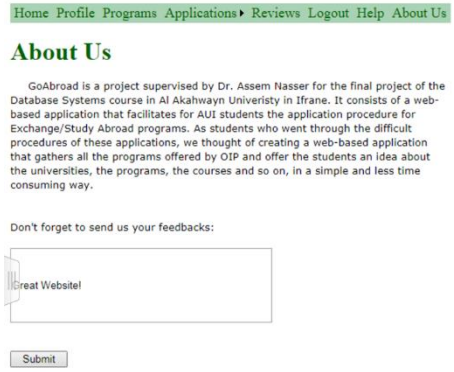
Write a Review

Past Applications page: contains applications that the student has already been accepted to, and allows the user to write reviews on their universities after selecting them.

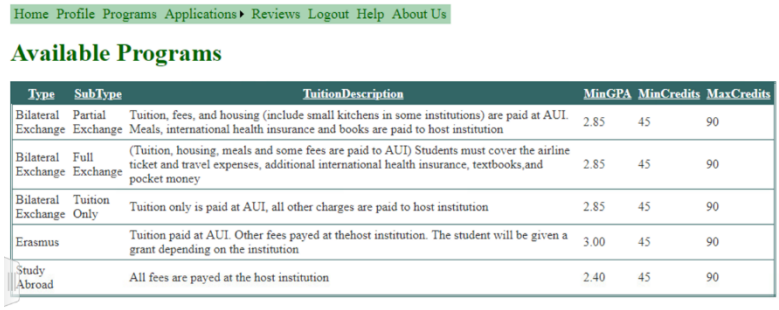
Current Applications page: contains applications that their status is still pending, and allows the user to cancel them after selecting them.



Reviews page: contains all the reviews written by the user on different universities.



About Us page: contains an overview of the project, and allows users to write feedback to the designers.



Type	SubType	TuitionDescription	MinGPA	MinCredits	MaxCredits
Bilateral Exchange	Partial Exchange	Tuition, fees, and housing (include small kitchens in some institutions) are paid at AUI. Meals, international health insurance and books are paid to host institution	2.85	45	90
Bilateral Exchange	Full Exchange	(Tuition, housing, meals and some fees are paid to AUI) Students must cover the airline ticket and travel expenses, additional international health insurance, textbooks,and pocket money	2.85	45	90
Bilateral Exchange	Tuition Only	Tuition only is paid at AUI, all other charges are paid to host institution	2.85	45	90
Erasmus		Tuition paid at AUI. Other fees payed at thehost institution. The student will be given a grant depending on the institution	3.00	45	90
Study Abroad		All fees are payed at the host institution	2.40	45	90

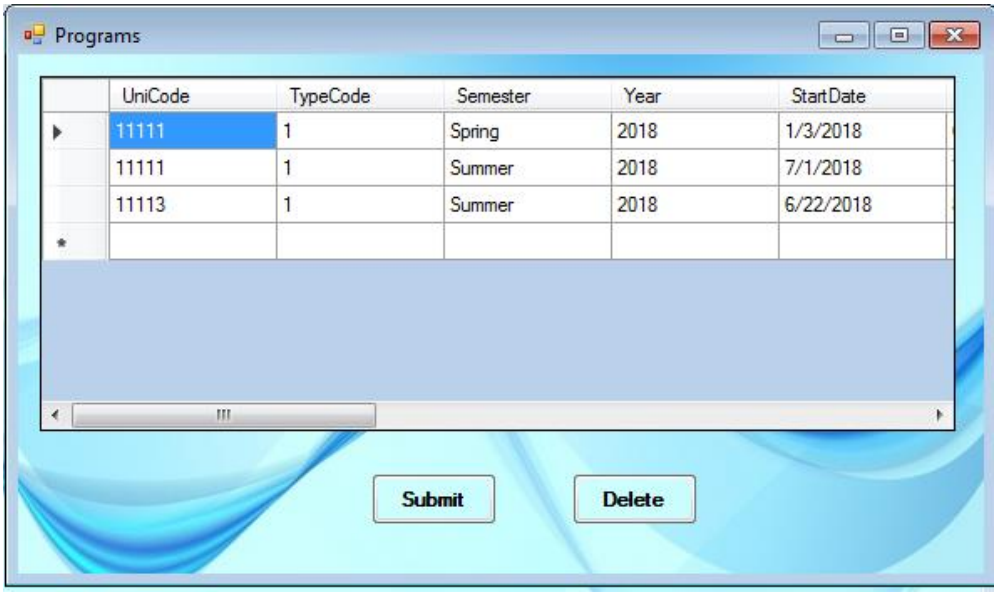
Help page: contains information about the available programs, their types, subtypes, and the differences between them.

B. Desktop Application

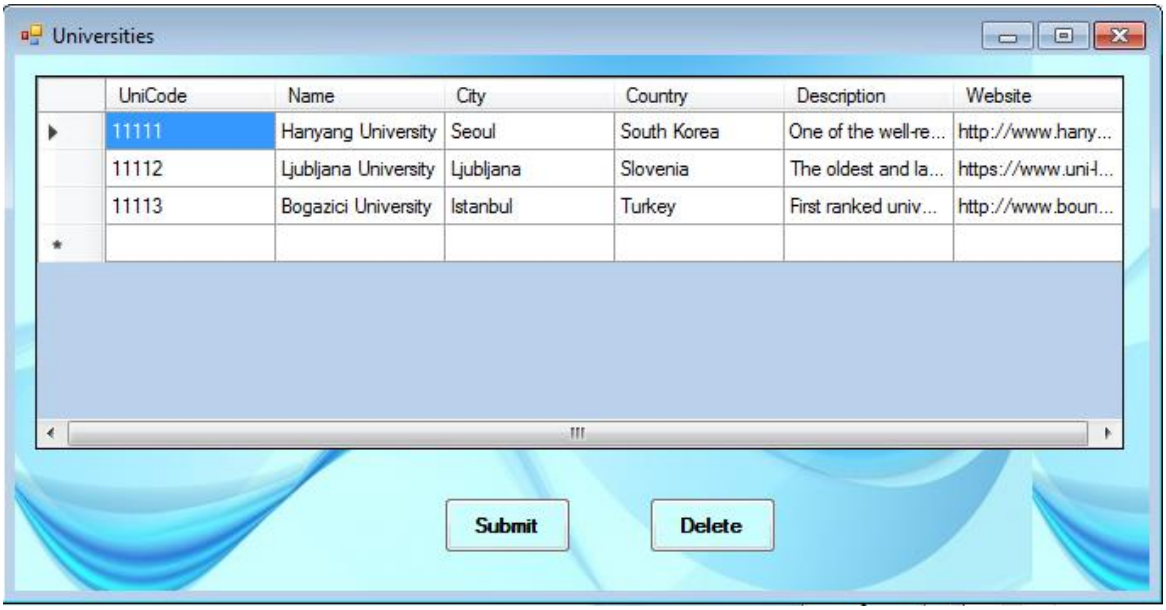
The first page you will be taken to is the login page

If you enter the correct credentials (username: 11234, password: oip123456789) You will be taken to the home page

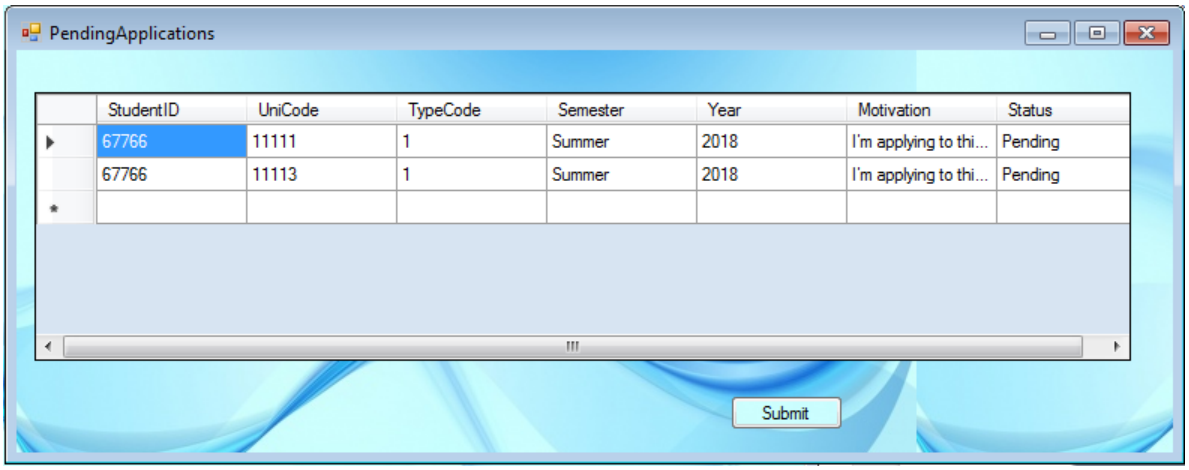
If you click on Programs, you will be taken to this form which has the list of programs. You can update the information of a program by changing the values in the field and clicking submit. You can insert a new program by filling in the information in the new line and clicking submit. You cand delete a program by selecting it and clicking delete.



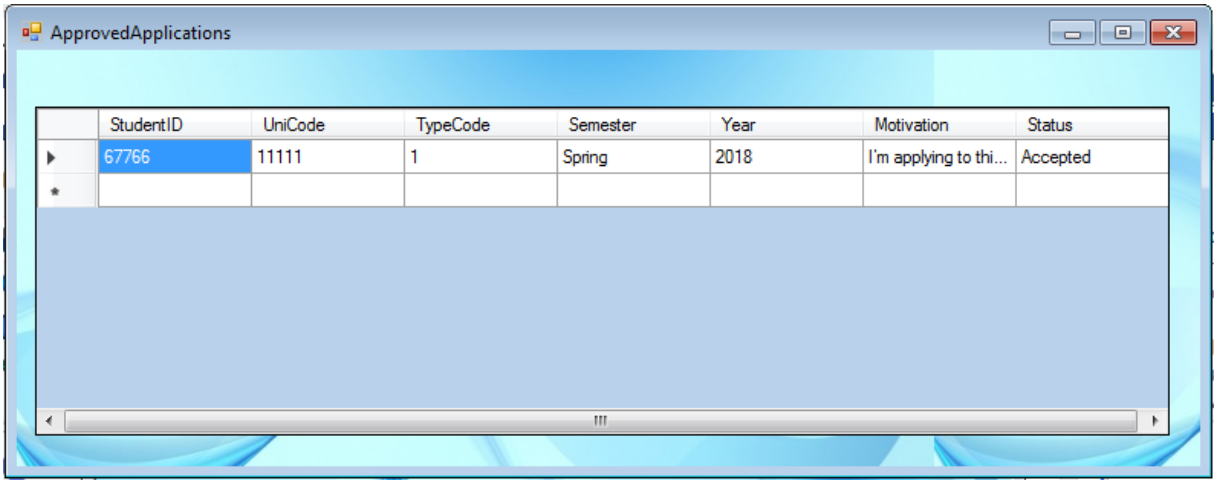
If you click on Universities on the home page, you will be redirected to this form where you can also update, insert, and delete similar to the program's form.



If you click on Pending Applications on the home page you will be taken to this form where you can update the status of a student's application to accept or reject it.



If you click on Past Applications on the home page you will be taken to this form which shows the past applications of students.



VI. Conclusion

A good website and desktop application require a well-designed database. This latter is the most difficult part as a simple missing or redundant relationship between entities may cause many problems in the database, and thereby on the website or desktop application as well.

To conclude, careful decisions should be taken during the design phase to avoid concurrency and to ensure synchronicity where it is needed, to enhance the overall performance of the applications.

VII. Future work

Our future work consists of working more on our desktop application and adding more functionalities to our web-based application.

As we want to get more control over our database, we will ask the Information Technology System department at our university to provide us with a big storage space that will allow our website to grow without any limitations.

We will also ask the OIP to provide us with the full database of the students who went or are currently on exchange, in addition to the list of universities that they have an agreement with so that we can update our database. This will allow us to deliver a complete and usable website to the OIP.