COURSEWORK 2

Zlatomira Dimcheva

COMP23111

12/03/2021

TABLE OF CONTENTS

1. Part A: Normalisation – page 3
    a. UNF – page 3
    b. 1NF – page 4
    c. 2NF – page 5
    d. 3NF – page 6

2. Part B: Relational Schema – page 7

3. Part C: Implementation – page 10

4. Part D: The Application – page 13

5. Part E: Stored Procedures and Triggers – page 20
    a. The Procedure – page 20
    b. The Trigger – page 20

1. Part A: Normalisation
   In this section I give explanation how I went from UNF → 1NF → 2NF → 3NF

a - UNF

Quiz ID	34											
Quiz Name	SQL											
Quiz Author	Peter Parker											
Quiz Available	Yes											
Quiz Duration	60 minutes											
Student ID	44											
Student Name	Duncan Hull											
Date of Attempt	22/11/2020											
Question Number	1				2				3			
Question	Which SQL statement…extract data..?				Which SQL statement…insert data..?				With SQL, how do you select…?			
Answer	Select	Open	Extract	Get	Insert New	Insert Into	Add Record	Add New	SELECT * FROM…	SELECT [all]…	SELECT * FROM….	SELECT [all]…

Rules: I identify all fields and sample data and also the primary key and underline it.

As a first step I create the UNF. We know that a relation is in UNF if it contains repeating groups and lacks automicity. This steps serves as a point where we can see the form of the database.

I take all the data from the information source and separate it in appropriate rows in the table.

In our case the primary key will consist of two parts – Quiz ID and Student ID. We underline the two rows as shown in the table.

We note that in this step we can have redundant data, but we need to deal with it later in the process.

The relation can also look like this:

Quiz (**QuizID, Student ID**, Quiz Name, Quiz Author, Quiz Available, Quiz Duration, Student Name, Date of Attempt, Question Number, Question, Answer)

b - 1NF

Quiz ID	34
Quiz Name	SQL
Quiz Author	Peter Parker
Quiz Available	Yes
Quiz Duration	60 minutes
Student ID	44
Student Name	Duncan Hull
Date of Attempt	22/11/2020

Quiz ID	34
Question Number	1
Question	Which SQL statement…extract data..?

Quiz ID	34
Question Number	1
Answer	Select

Rules: Remove Repeating Attributes, identify new compound key.

As a second step I create the 1NF. We know that a relation is in 1NF if it does not contain repeating groups and columns contain atomic values. In this step we use the UNF and remove all the repeating groups as shown above.

I create a separate relation for the questions, where we need to identify a new key and also copy the key from the previous relation to get the compound key – Quiz ID and Question Number form it in this case.

I also separate the answers in a separate relation where the compound key will be – Answer, Quiz ID and Question Number.

The relation can also look like this:

Quiz (**QuizID, Student ID**, Quiz Name, Quiz Author, Quiz Available, Quiz Duration, Student Name, Date of Attempt)

Question (**QuizID, Question Number,** Question)

Answer (**QuizID, Question Number, Answer**)

c - 2NF

Quiz ID	34
Quiz Name	SQL
Quiz Author	Peter Parker
Quiz Available	Yes
Quiz Duration	60 minutes

Student ID	44
Student Name	Duncan Hull

Quiz ID	34
Student ID	44
Date of Attempt	22/11/2020

Quiz ID	34
Question Number	1
Question	Which SQL statement…extract data..?

Quiz ID	34
Question Number	1
Answer	Select

Rules: Check Partial Dependencies, Remove any to new table with copy of determinant.

As a third step I create the 2NF. We know that a relation is in 2NF if it is in 1NF and also each non-key attribute is fully functionally dependent on the primary key. In case they are not I create a new relation.

I separate a relation for the students where the key becomes StudentID – In the table I put also Student Name because it is only dependent on StudentID.

I also create a relation for the quiz separately where QuizID is key – The same rule applies here. All attributes that are left are dependent on the QuizID.

And because Date of Attempt is dependent on both QuizID and StudentID it stays in a separate table for the attempts taken.

The relation can also look like this:

Quiz (**QuizID,** Quiz Name, Quiz Author, Quiz Available, Quiz Duration)

Question (**QuizID, Question Number,** Question)

Answer (**QuizID, Question Number, Answer**)

Attempt (**QuizID, Student ID,** Date of Attempt)

Student (**Student ID,** Student Name)

d - 3NF

Quiz ID	34
Quiz Name	SQL
Quiz Author	Peter Parker
Quiz Available	Yes
Quiz Duration	60 minutes

Student ID	44
Student Name	Duncan Hull

Quiz ID	34
Student ID	44
Date of Attempt	22/11/2020

Quiz ID	34
Question Number	1
Question	Which SQL statement…extract data..?

Quiz ID	34
Question Number	1
Answer	Select

Rules: Check Transitive Dependencies, Remove any to new table with copy of determinant

As a forth step I create the 3NF. We know that a relation is in 3NF if it is in 2NF and 1NF and also each non-key attribute is not transitively dependent on the key. In case they are I create a new relation.

After thoroughly searching the dependencies I realized that 2NF and 3NF stay the same if we make the assumption that an Author can create two quizzes with the same name.

The relation can also look like this:

Quiz (**QuizID,** Quiz Name, Quiz Author, Quiz Available, Quiz Duration, Student Name)

Question (**QuizID, Question Number,** Question)

Answer (**QuizID, Question Number, Answer**)

Attempt (**QuizID, Student ID,** Date of Attempt)

Student (**Student ID,** Student Name)

2. Part B: Relational Schema

The following represents the relational schema. All primary keys are clearly shown.

The relations between tables are clearly identified by the foreign keys.

All attributes are given and I have added all needed constraints.

student (ID, Name, Email, Password)

- PK ID
- ID → Integer, Unsigned 0-4294967295, AUTO_INCREMENT , primary key
- Name → Varchar, 255, NOT NULL
- Email → Varchar , 255, NOT NULL, UNIQUE
- Password → Varchar, 255, NOT NULL

→We take the email, name and password from the register page and we need to store it for future log in. The ID is used to identify the student.

staff (Email, Name, Password)

- PK Email
- Email → Varchar , 255, primary key
- Name → Varchar, 255, NOT NULL
- Password → Varchar, 255, NOT NULL

→We take the email, name and password from the register page and we need to store it for future log in. The Email is used to identify the staff member.

quiz (ID, QuizName, AuthorEmail, Availability, TimeLimit)

- PK ID
- FK AuthorEmail → staff (Email)
- ID → Integer, Unsigned 0-4294967295, AUTO_INCREMENT, primary key
- QuizName → Varchar, 255, NOT NULL
- AuthorEmail → Varchar, 255, NOT NULL, foreign key
- Availability → Varchar, 100, NOT NULL
- Timelimit → Integer, Unsigned 0-4294967295

→We take the name, author, availability and time limit from the create quiz or update quiz page and we need to store it for future updates on quiz. The ID is used to identify the quiz.

takesstudent (TakesID, StudentID, QuizID, Score)

- PK TakesID
- FK StudentID → student (ID)
- FK QuizID → quiz (ID)
- TakesID → Integer, Unsigned 0-4294967295, AUTO_INCREMENT, primary key
- StudentID → Integer, Unsigned 0-4294967295, NOT NULL
- QuizID → Integer, Unsigned 0-4294967295, NOT NULL
- Score → Integer, Unsigned 0-4294967295, NOT NULL

→We take the student id and quiz id from the take quiz page and we need to store it for future references when we print scores. The TakesID is used to identify the attempt.

takesstaff (TakesID, StaffEmail, QuizID, Score)

- PK TakesID
- FK StaffEmail → staff (Email)
- FK QuizID → quiz (ID)
- TakesID → Integer, Unsigned 0-4294967295, AUTO_INCREMENT, primary key
- StaffEmail → Varchar, 255, NOT NULL
- QuizID → Integer, Unsigned 0-4294967295, NOT NULL
- Score → Integer, Unsigned 0-4294967295, NOT NULL

→ We take the stafft id and quiz id from the take quiz page and we need to store it for future references when we print scores. The TakesID is used to identify the attempt.

questions (QuizID, QuestionNumber, Question, AnswerOne, AnswerTwo, AnswerThree, AnswerFour)

- PK QuizID, QuestionNumber
- FK QuizID → quiz (ID)
- QuizID → Integer, Unsigned 0-4294967295, primary key
- QuestionNumber → Integer, Unsigned 0-4294967295, primary key
- Question → Varchar, 255, NOT NULL
- AnswerOne → Varchar, 255, NOT NULL
- AnswerTwo → Varchar, 255, NOT NULL
- AnswerThree → Varchar, 255, NOT NULL
- AnswerFour → Varchar, 255, NOT NULL

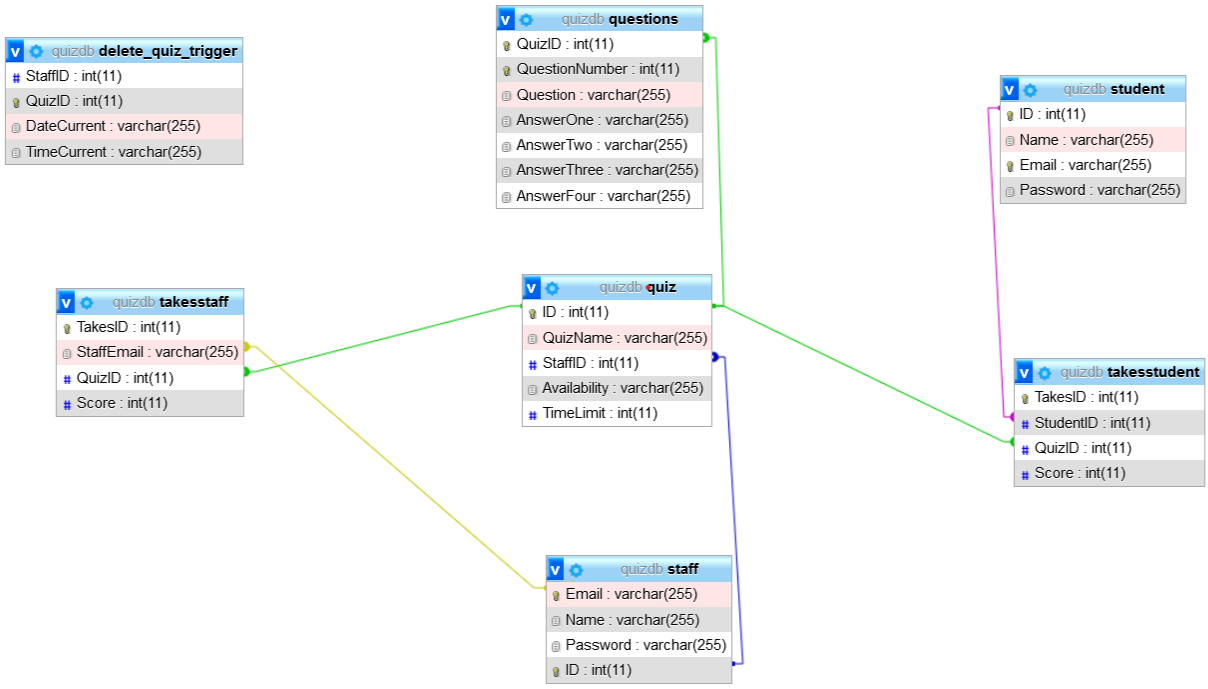
→ We take all the information from the create questions page and we need to store it for future references when we print the wuizzes. The QuizID and QuestionNumber are used to identify the question.

delete_quiz_trigger (QuizID, StaffID, DateCurrent, TimeCurrent)

- PK QuizID
- StaffID → Integer, Unsigned 0-4294967295, NOT NULL
- QuizID → Integer, Unsigned 0-4294967295, primary key
- DateCurrent → Varchar, 255, NOT NULL
- TimeCurrent → Varchar, 255, NOT NULL

→ We take the information when we delete a quiz and we need to store it in this table if we need it. The QuizID is used to identify the deletion of a quiz.

You can see the database design below:



3. Part C: Implementation

In this section I include all MySQL statements needed to implement my relational schema.

```
CREATE TABLE IF NOT EXISTS student (
    ID int AUTO_INCREMENT NOT NULL,
    Name varchar(255) NOT NULL,
    Email varchar(255) NOT NULL UNIQUE,
    Password  varchar(255) NOT NULL,
    PRIMARY KEY(ID)
);

CREATE TABLE IF NOT EXISTS staff (
    Email varchar(255) NOT NULL UNIQUE,
    Name varchar(255) NOT NULL,
    Password  varchar(255) NOT NULL,
    ID int AUTO_INCREMENT,
    PRIMARY KEY(ID)
);

CREATE TABLE IF NOT EXISTS quiz (
    ID int AUTO_INCREMENT NOT NULL,
    QuizName varchar(255) NOT NULL,
    StaffID int NOT NULL,
    Availability varchar(255) NOT NULL,
    TimeLimit int,
    CONSTRAINT `staff_ibfk_1` FOREIGN KEY (`StaffID`) REFERENCES `staff`(`ID`) ON DELETE CASCADE ON UPDATE CASCADE,
    PRIMARY KEY(ID)
);
```

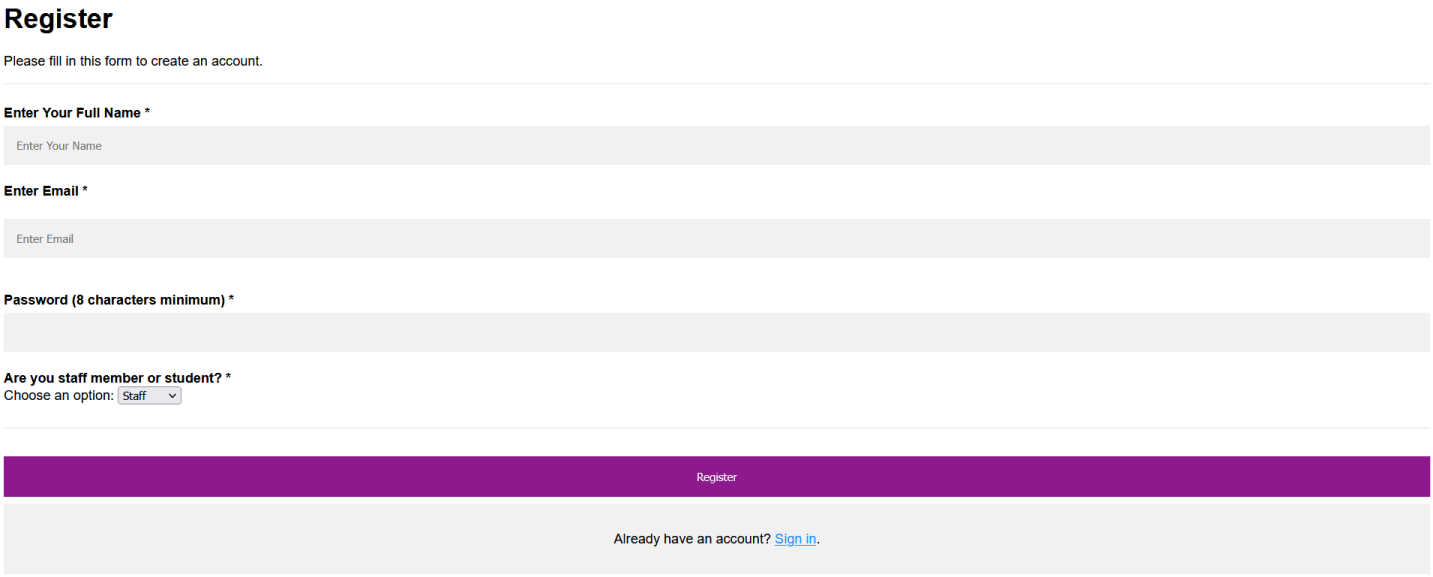
```
CREATE TABLE IF NOT EXISTS takesstudent (
    TakesID int AUTO_INCREMENT,
    StudentID int NOT NULL,
    QuizID int NOT NULL,
    Score int NOT NULL,
    CONSTRAINT `student_ibfk_1` FOREIGN KEY (`StudentID`) REFERENCES `student`(`ID`) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT `quizi_ibfk_1` FOREIGN KEY (`QuizID`) REFERENCES `quiz`(`ID`) ON DELETE CASCADE ON UPDATE CASCADE,
    PRIMARY KEY(TakesID)
);

CREATE TABLE IF NOT EXISTS takesstaff (
    TakesID int AUTO_INCREMENT,
    StaffEmail varchar(255) NOT NULL,
    QuizID int NOT NULL,
    Score int NOT NULL,
    CONSTRAINT `staffid_ibfk_3` FOREIGN KEY (`StaffEmail`) REFERENCES `staff`(`Email`) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT `quizid_ibfk_3` FOREIGN KEY (`QuizID`) REFERENCES `quiz`(`ID`) ON DELETE CASCADE ON UPDATE CASCADE,
    PRIMARY KEY(TakesID)
);

CREATE TABLE IF NOT EXISTS questions (
    QuizID int,
    QuestionNumber int,
    Question varchar(255) NOT NULL,
    AnswerOne varchar(255) NOT NULL,
    AnswerTwo varchar(255) NOT NULL,
    AnswerThree varchar(255) NOT NULL,
    AnswerFour varchar(255) NOT NULL,
    CONSTRAINT `quiz_ibfk_2` FOREIGN KEY (`QuizID`) REFERENCES `quiz`(`ID`) ON DELETE CASCADE ON UPDATE CASCADE,
    PRIMARY KEY (QuizID, QuestionNumber));
```

```
CREATE TABLE IF NOT EXISTS delete_quiz_trigger (
    StaffID int NOT NULL,
    QuizID int,
    DateCurrent varchar(255) NOT NULL,
    TimeCurrent varchar(255) NOT NULL,
    PRIMARY KEY(QuizID)
);
```

4. Part D: The Application

In this section I include all screenshots and explanation needed for a user to navigate through the application.

The first step is always registering using the following register page:

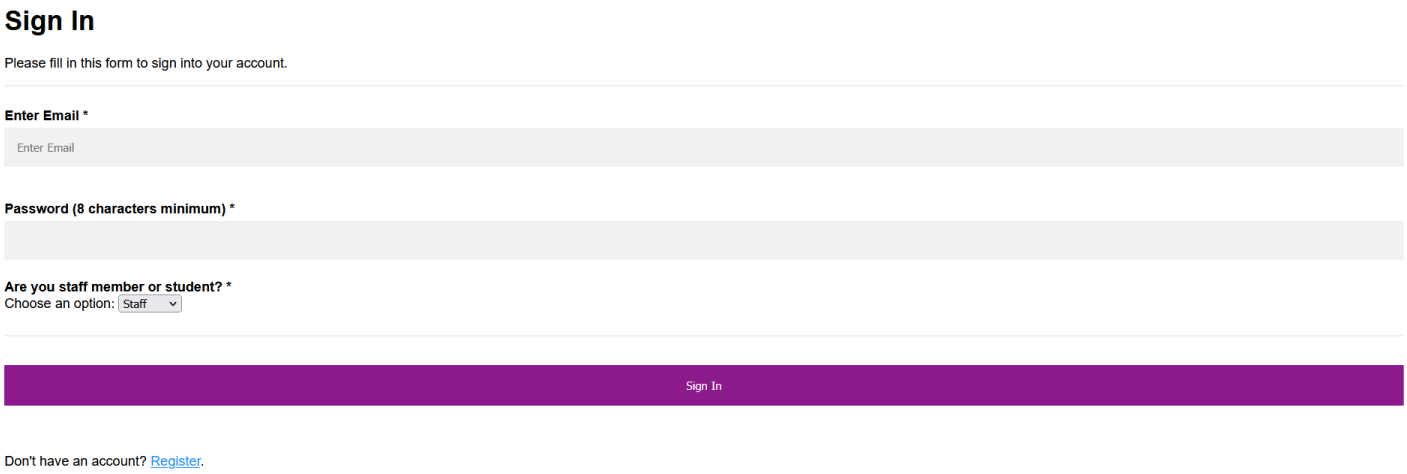


By inputting the requested data – name, email, password and occupation – the user can press the Register button, which will take them to the Sign In page.

It is important to note that the user has to include at least two names and an email can only be used once to register as either staff or student. The password must contain at least one number and one uppercase and lowercase letter, and at least 8 or more characters. The password is hashed and stored in the database, as is all the other information – depending on the occupation in either table staff or table student.

In case the user already has an account they can go to the Sign In page directly by clicking the blue link at the bottom of the page.

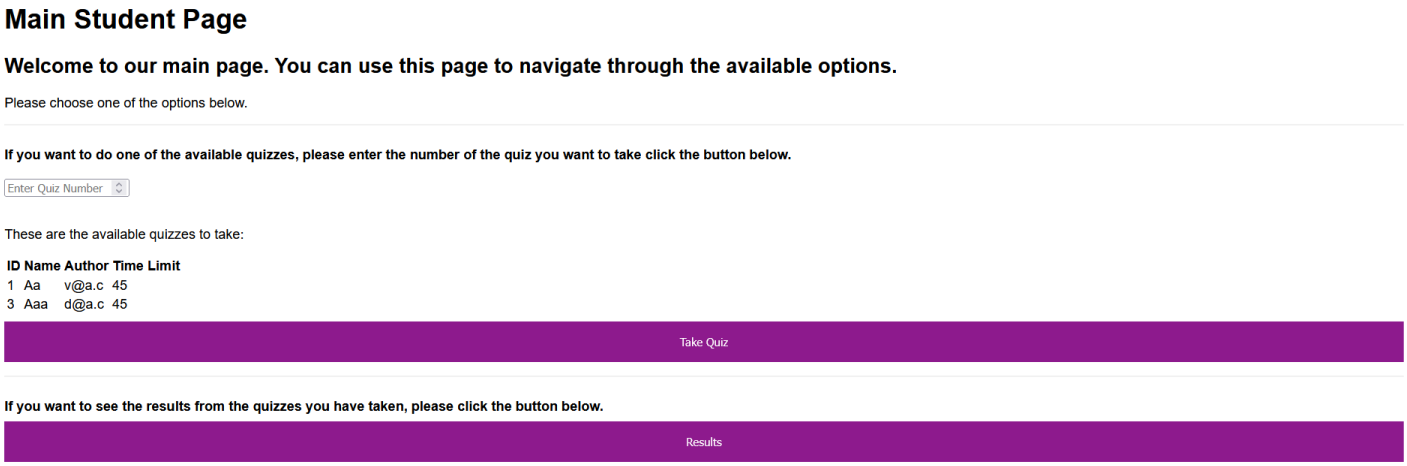
Otherwise after correct input is validated the user is transferred to the Sign In page, which looks like this:



As with the register page when user inputs email, password and occupation and presses the Sign In button, the information is checked in the appropriate table in the database.

In case all input is correct the user is transferred to either the Main Page for staff or the Main Page for students.

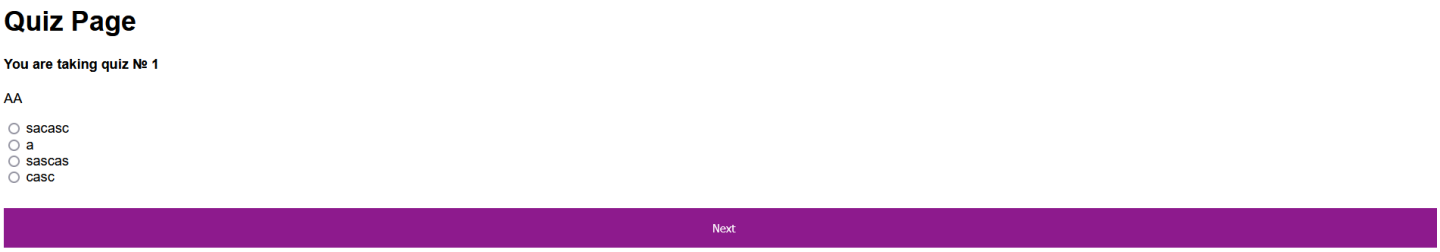
The Main Page for the students looks like this:



The student can take a quiz by inputting the number of the quiz they want to take and clicking on the Take Quiz button. All available quizzes are taken from the database and printed using ID, Name, Author and Time Limit.

The other available option for the student is to see all results for the quizzes taken by him/her by pressing the button Results.

The take quiz page looks like this:



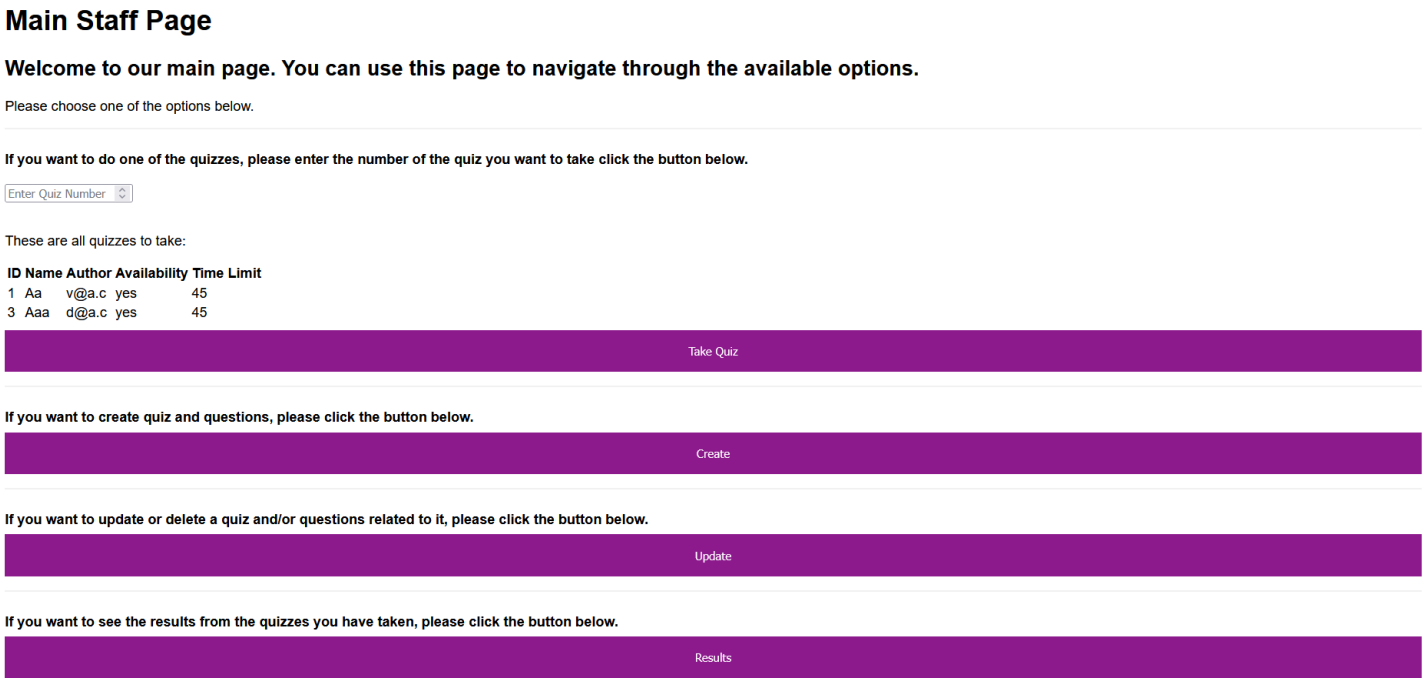
One question at a time is being shown on the page using the database table questions. The student can see which quiz they are taking and the question underneath that (For the purpose of showing functionality I have just given random letters as question/options). Four options are provided. Once zero or one option is chosen and the button Next is pressed the student can't go back to the previous question.

For the score to be recorded and stored as percentage in the database the user has to reach the last question, when the button changes from Next to Submit. After pressing the Submit button, the student is transferred to the Results page.

The results page takes the information from the database table takesstudent and looks like this:

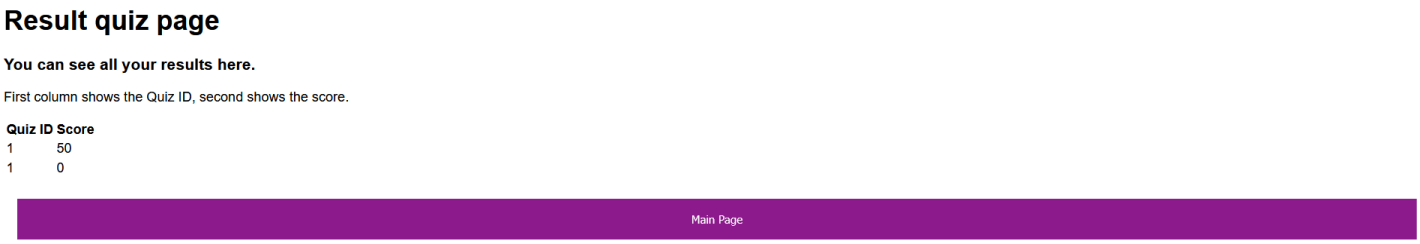
The user can see all the quizzes he/she has taken and the scores for them. Then by pressing Main Page button the student is sent back to the Main Student Page shown above. Then they can take another quiz or go straight to the Result page.

In case the user logs in as staff they are transferred to a different main page with more option. The Staff Main Page looks like this:

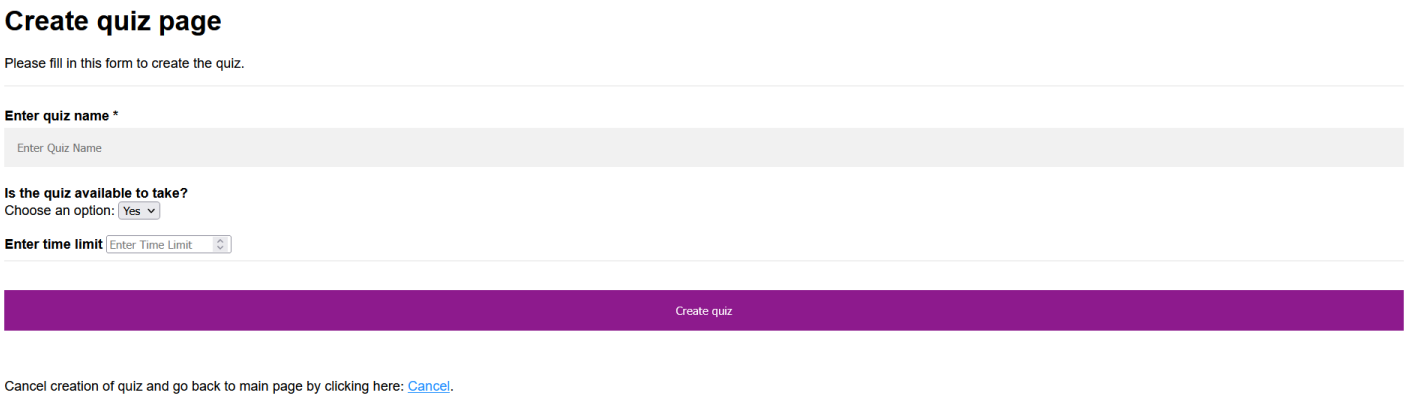


The staff has the same two options as the students. They can take a quiz by clicking on the Take Quiz button, which takes them to the same Take Quiz Page shown above (again only one question is shown at a time). In the end of the quiz they are sent to the Results page.

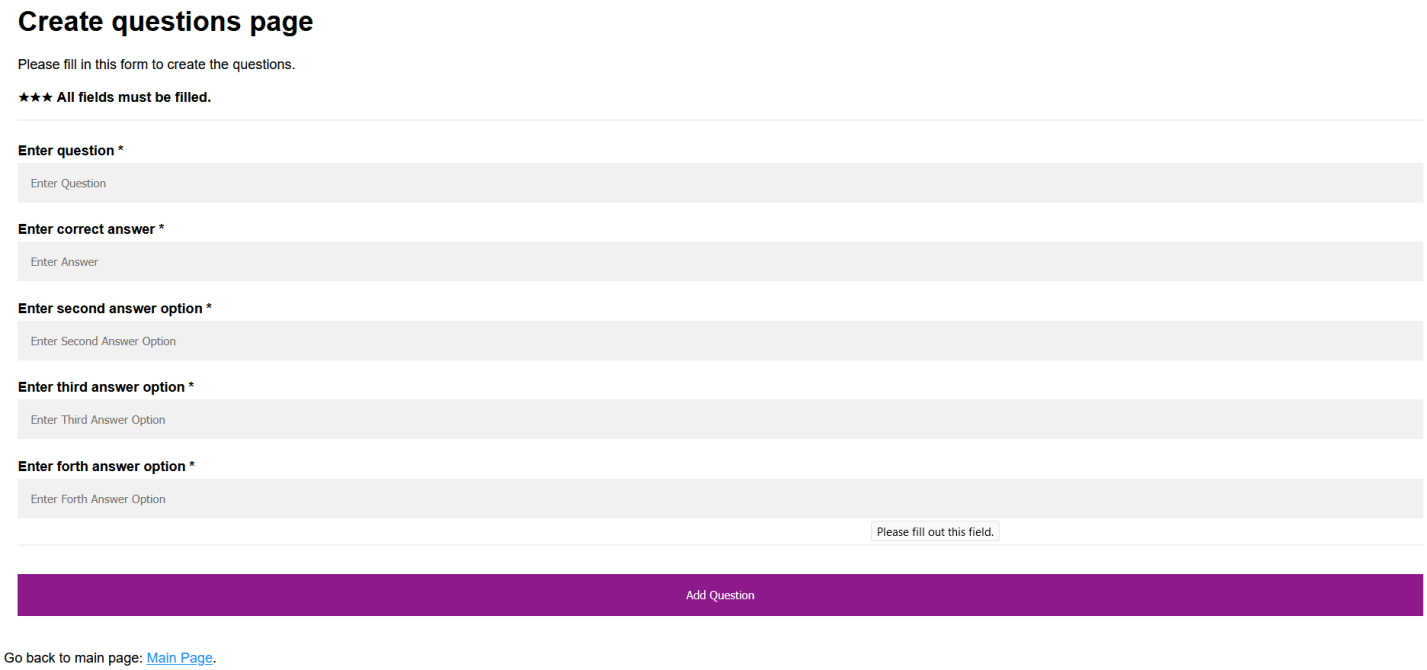
From the main page they can also go straight to the Results page. The results page looks like this (same as with students):



There are two other options for staff members. They can click on the button Create which will transfer them to a page where they can create a new quiz. It looks like this:

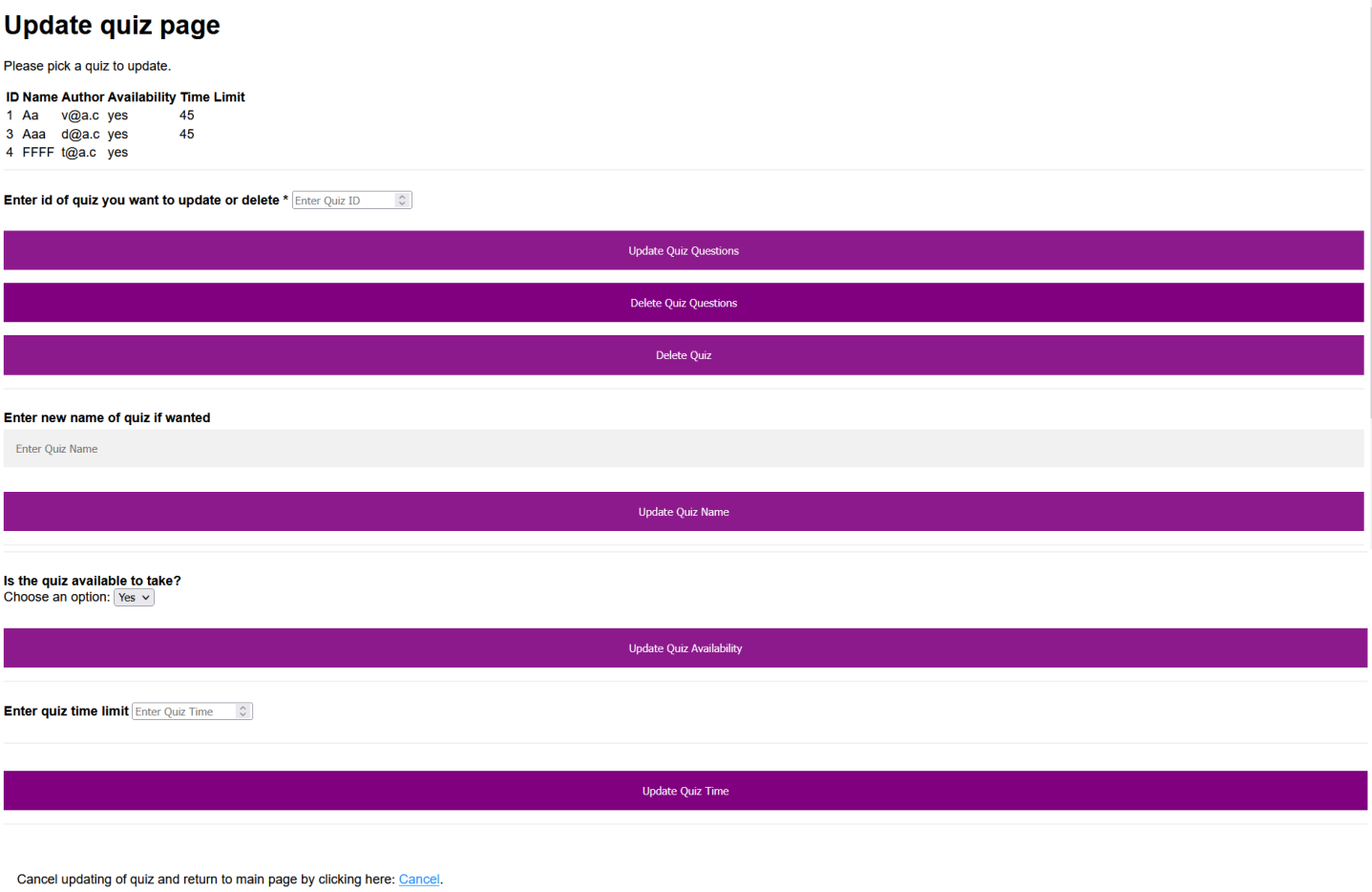


It is required that they give a name for the quiz and availability. They can also give if needed a time limit. Then the user presses Create quiz, all the information is stored in the database table quiz and the user is sent to a page where they can create questions for this quiz. The page looks like this:



The staff member then fills all the input fields and presses the Add Question button. All the information from this page is stored in the database in a table called questions. When the user wants to stop adding question the button Main Page can be pressed and the staff member is sent back to the Main Staff Page.

We can look at the forth option now. When the button Update is pressed the user is sent to a page where they can update and delete quizzes and/or question related to them. It looks like this:



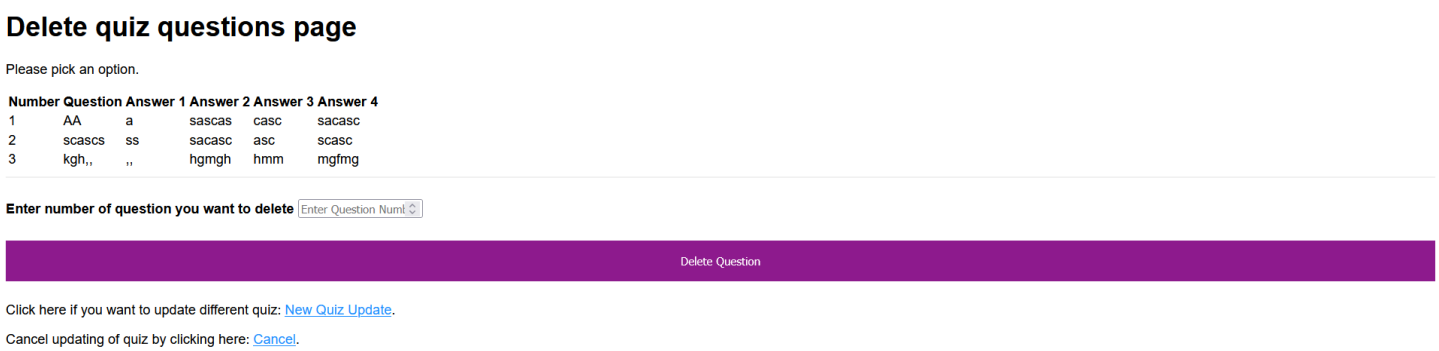
All quizzes are taken from the database table quiz and printed as options. The user can change the quiz name by entering the new name and the quiz id (beginning of page, required) and pressing the button Update Quiz Name.

Same applies for Availability and Time Limit. When one of these options is executed the appropriate quiz in the table quiz is updated in the database. Important to note here that Quiz ID is always required to enter before pressing any of the six buttons.

The other three options are: Update Quiz Questions, Delete Quiz Questions, Delete Quiz.

By entering quiz ID and pressing Delete Quiz the user deletes the quiz and all questions related to it. This is secured by the CASCADE in the database tables. In case the user wants to only delete questions they can enter quiz ID and press Delete Quiz Questions.

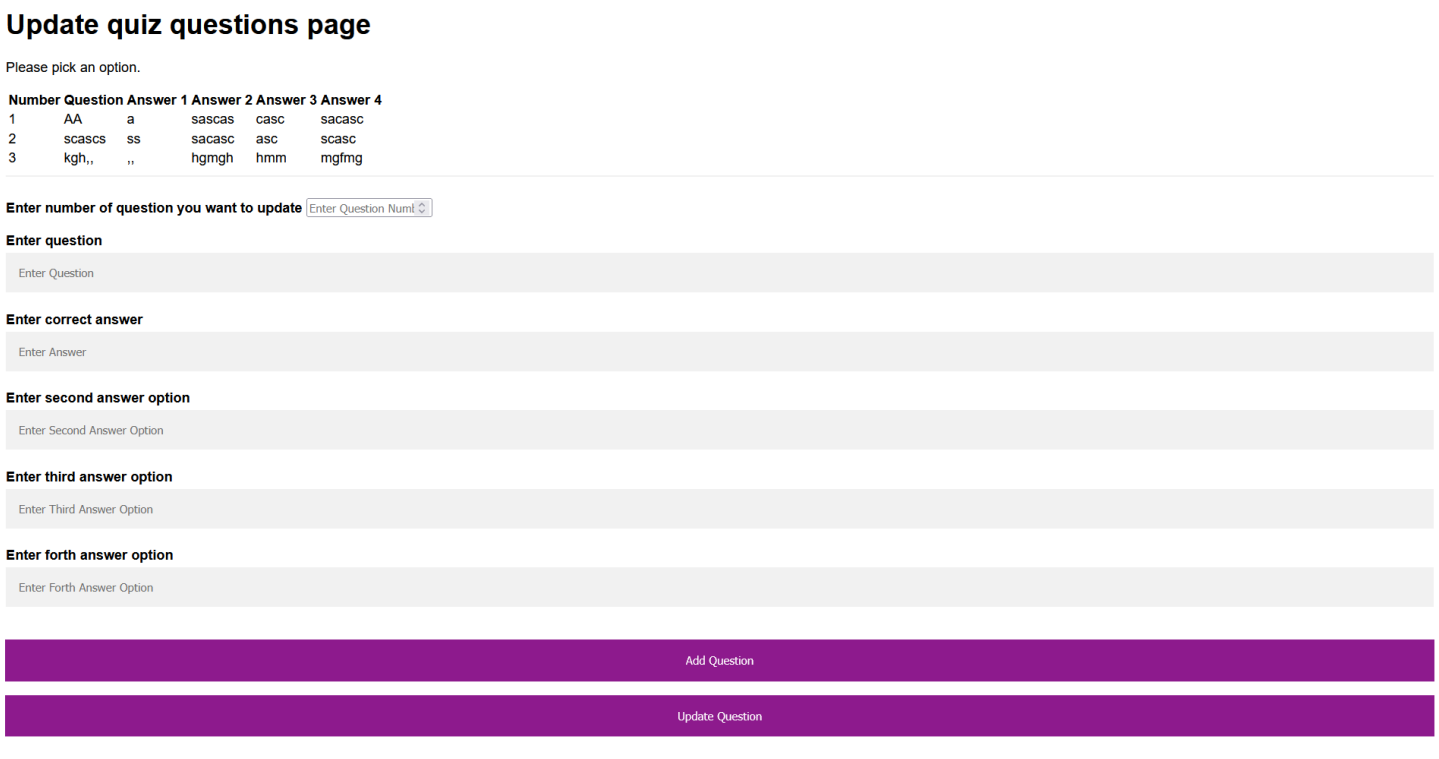
The page the user is sent to looks like this:



The number of the question to delete is required. Then by clicking Delete Question the question is deleted and the numbers of the other questions is updated accordingly in the database.

There are two options below the button – go back to Main Page by clicking Cancel which sends user back to Main Staff Page or clicking New Quiz Update which sends user back to Update Quiz page.

The last option for staff members is the Update Quiz Question button. There the user can add and update quiz questions. The page looks like this:



All question related to the quiz are taken from the database and printed as options. The user has to enter the number of question they want to update if they want to see a change in the question and database update. They can do that by entering question number and all the new information required to update the question.

The other option is to just add a new question by clicking Add Question where the user does not need to enter question number. It is automatically incremented in the database.

Again they have the options of cancelling the questions update and going back to either Main Page or Update Quiz Page.

Every page communicates with the database – selects, updates, inserts or deletes information from it. And all the required functionality is implemented.

5. Part E: Stored Procedures and Triggers

In this section I have the code for the procedure and the trigger required from the instructions

a - The Procedure - Create a stored procedure that displays the student names and their scores for the quizzes where they achieved less than 40%.

```
DELIMITER ^^
CREATE PROCEDURE getResultsBelowForty()
	BEGIN
		SELECT student.`Name`, quizz.`QuizName`, takesstudent.`TakesID`, takesstudent.`Score`
		FROM students, takesstudent, quiz
		WHERE quiz.`ID` = takesstudent.`QuizID` AND student.`ID` = takesstudent.`StudentID`
AND Score < 40;
	END ^^
DELIMITER ;
```

To call the procedure we use **CALL getResultsBelowForty();**

b - The Trigger - Create a trigger that will log the staff id, the quiz id and the current date and time, when a staff user deletes a quiz.

```
DELIMITER ^
CREATE TRIGGER before_quiz_delete
	AFTER DELETE ON quiz FOR EACH ROW
	BEGIN
		INSERT INTO delete_quiz_trigger
		SET
		StaffID = OLD.StaffID,
		QuizID = OLD.ID,
		DateCurrent = curdate(),
		TimeCurrent = curtime();
	END ^
DELIMITER ;
```