Database Coursework 2

Horia-Gabriel Radu

COMP23111

Databases Systems

November - December 2020

Contents:

1 Introduction

This is the PDF for my second coursework at Database Systems.
Some parts of this PDF were directly copy pasted from my Visual Studio Code and thus will have a black background.

2 Part A: Normalisation

2.1 Introduction to section

I will do my best to try to make this data as efficiently stored.
I will separate each data into two tables, one in which the information for a primary key is stored and one in which the relations are made.
This is done so that we reduce as much as we can database anomalies and we increase efficiency in accessing and modifying anything in the database.
Plus making it much more readable and easier to implement

2.2 UNF

Explanation:

Translated the information source into a UNF by extracting the data and putting in a table, so that I can further work on normalising it.
For the last row, due to lack of space I will fully write the data here:

Question 1: Which SQL statement is used to extract data from a database?
Answers: SELECT, OPEN, EXTRACT, GET

Question 2: Which SQL statement is used to insert new data in a database?
Answers: INSERT NEW, INSERT INTO, ADD RECORD, ADD NEW

Question 3: With SQL, how do you select all the records from a table named “Persons” where the value of the column “FirstName” is “Peter”?
Answers: SELECT * FROM Persons WHERE FIRSTNAME <> ‘Peter’, SELECT [all] FROM Person WHERE FirstName = ‘Peter’, SELECT * FROM Persons WHERE FirstName = ‘Peter’, SELECT [akk] FROM Person WHERE FirstName LIKE ‘Peter’

Table:

Primary Key Quiz ID	34		
Quiz Name	SQL		
Quiz Author	Peter Parker		
Quiz Available	Yes		
Quiz Duration	60 minutes		
Primary Key Student ID	44		
Student Name	Duncan Hull		
Date of Attempt	22/11/2020		
Questions	1 + Answers	2 + Answers	3 + Answers

2.3 1NF

Explanation:

Separated the repeating groups (questions and answers) into new relations. And made two more primary keys for those two groups, in which I also added foreign keys, so that the whole form would be connected.

Relations:

Quiz Table:

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

Question Table:

Question ID = 1, 2 or 3 - **Primary key**
Quiz ID = 34 - **Foreign key**
Question String = String for each questions (1, 2 or 3)

Answers Table:

Answer ID = An ID for all the answers for the questions - **Primary key**
Question ID = 1, 2 or 3 - **Foreign key**
Answer String = All the answers strings for the questions

2.4 2NF

Explanation:

First I made the functional dependencies from 1NF to help in creating the 2NF. Then made 3 new tables, Student Quiz_Question and Question_Answers so that each relation is connected more efficiently.

Relations:

Functional Dependencies:

Quiz ID -> Quiz Name, Quiz Author, Quiz Available, Quiz Duration
Student ID -> Student Name, Date of Attempt
Quiz ID -> Question ID
Question ID -> Question String
Question ID -> Answer ID
Answer ID -> Answer String

Quiz Table:

Quiz ID = 34 - **Primary key**
Quiz Name = SQL
Quiz Author = Peter Parker
Quiz Available = Yes
Quiz Duration = 60 minutes

Student Table:

Student ID = 44 - **Primary key**
Quiz ID = 34 - **Foreign key**
Student Name = Duncan Hull
Date of Attempt = 22/11/2020

Question Table:

Question ID = 1, 2 or 3 - **Primary key**
Question String = String for each questions (1, 2 or 3)

Quiz_Question Table:

Question ID = 1, 2 or 3 - **Foreign key**
Quiz ID = 34 - **Foreign key**

Answers Table:

Answer ID = An ID for all the answers for the questions - **Primary key**
Answer String = All the answers strings for the questions

Question_Answers Table:

Answer ID = An ID for all the answers for the questions - **Foreign key**
Question ID = 1, 2 or 3 - **Foreign key**

2.5 3NF

Explanation:

Firstly, I split the Student Table into Student Table and Quiz_Student Table, so that it is easier to track which student took which quiz, and on what date, plus, it stores the student info more efficiently.

Secondly, I added an attribute (Is Correct) to the Question_Answers Table to keep track which answers are correct and which are incorrect (this way makes it so that you can have questions with multiple correct answers).

Relations:

Quiz Table:

Quiz ID = 34 - **Primary key**
Quiz Name = SQL
Quiz Author = Peter Parker
Quiz Available = Yes
Quiz Duration = 60 minutes

Student Table:

Student ID = 44 - **Primary key**
Student Name = Duncan Hull

Quiz_Student Table:

Student ID = 44 - **Foreign key**
Quiz ID = 34 - **Foreign key**
Date of Attempt = 22/11/2020

Question Table:

Question ID = 1, 2 or 3 - **Primary key**
Question String = String for each questions (1, 2 or 3)

Quiz_Question Table:

Question ID = 1, 2 or 3 - **Foreign key**
Quiz ID = 34 - **Foreign key**

Answers Table:

Answer ID = An ID for all the answers for the questions - **Primary key**
Answer String = All the answers strings for the questions

Question_Answers Table:

Answer ID = An ID for all the answers for the questions - **Foreign key**
Question ID = 1, 2 or 3 - **Foreign key**
Is Correct = Yes or No

3 Part B: Relational Schema

3.1 Introduction to section

Made this Relational Schema to highlight each table's attributes and primary and foreign key, while also making sure to point out every relation between tables.

3.2 My Schema

Quiz Table (Quiz ID, Quiz Name, Quiz Author, Quiz Available, Quiz Duration):
    pk[Quiz ID]

Student Table (Student ID, Student Name):
    pk[Student ID]

Quiz_Student Table (Student ID, Quiz ID, Date of Attempt):
    fk[Student ID -> Student.Student ID]
    fk[Quiz ID -> Quiz.Quiz ID]

Question Table (Question ID, Question String):
    pk[Question ID]

Quiz_Question Table (Question ID, Quiz ID):
    fk[Question ID -> Question.Question ID]
    fk[Quiz ID -> Quiz.Quiz ID]

Answers Table (Answer ID, Answer String):
    pk[Answer ID]

Question_Answers Table (Answer ID, Question ID, Is Correct):
    fk[Answer ID -> Answers.Answer ID]
    fk[Question ID -> Question.Question ID]

4 Part C: Implementation

4.1 Introduction to section

I have done my best to write the code that will be used to implement the database, based on the schema from the last part.
I might still have to make a few changes to it in the following parts as I start creating the application.
That is because issues may show up and I will have to adapt to the circumstances, making modifications to the MySQL database statements as I go.

4.2 MySQL Statements used to implement your schema

```
CREATE TABLE `Quiz` (
        `Quiz_ID` int PRIMARY KEY,
        `Quiz_Name` varchar(255) NOT NULL,
        `Quiz_Author` varchar(255) NOT NULL,
        `Quiz_Available` boolean NOT NULL,
        `Quiz_Duration` int NOT NULL
        );

CREATE TABLE `Student` (
        `Student_ID` int PRIMARY KEY,
        `Student_Name` varchar(255) NOT NULL
        );

CREATE TABLE `Quiz_Student` (
        `Student_ID` int NOT NULL,
        `Quiz_ID` int NOT NULL,
        `Date_of_Attempt` date NOT NULL,
        PRIMARY KEY (`Student_ID`, `Quiz_ID`)
        );

CREATE TABLE `Question` (
        `Question_ID` int PRIMARY KEY,
        `Question_String` varchar(255) NOT NULL
        );

CREATE TABLE `Quiz_Question` (
        `Question_ID` int NOT NULL,
        `Quiz_ID` int NOT NULL,
        PRIMARY KEY (`Question_ID`, `Quiz_ID`)
        );

CREATE TABLE `Answers` (
        `Answer_ID` int PRIMARY KEY,
        `Answer_String` varchar(255) NOT NULL
        );
```

```
CREATE TABLE `Question_Answers` (
        `Answer_ID` int NOT NULL,
        `Question_ID` int NOT NULL,
        `Is_Correct` boolean NOT NULL,
        PRIMARY KEY (`Answer_ID`, `Question_ID`)
        );

ALTER TABLE `Quiz_Student` ADD FOREIGN KEY (`Student_ID`) REFERENCES
`Student` (`Student_ID`);
ALTER TABLE `Quiz_Student` ADD FOREIGN KEY (`Quiz_ID`) REFERENCES
`Quiz` (`Quiz_ID`);

ALTER TABLE `Quiz_Question` ADD FOREIGN KEY (`Question_ID`)
REFERENCES `Question` (`Question_ID`);
ALTER TABLE `Quiz_Question` ADD FOREIGN KEY (`Quiz_ID`) REFERENCES
`Quiz` (`Quiz_ID`);

ALTER TABLE `Question_Answers` ADD FOREIGN KEY (`Answer_ID`)
REFERENCES `Answers` (`Answer_ID`);
ALTER TABLE `Question_Answers` ADD FOREIGN KEY (`Question_ID`)
REFERENCES `Question` (`Question_ID`);
```

4.3 MySQL Statements Improved

After finishing the application, I have made quite a few design changes to my database from the Implementation I made based on the Relational Schema.
I will include the MySQL Statements that were used for the app below:

```
DROP DATABASE IF EXISTS `quiz_db`;
CREATE DATABASE `quiz_db`;
USE `quiz_db`;

CREATE TABLE `users` (
   `id` int PRIMARY KEY AUTO_INCREMENT,
   `username` varchar(255) NOT NULL UNIQUE,
   `password` varchar(255) NOT NULL
);

CREATE TABLE `Quiz` (
   `Quiz_ID` int PRIMARY KEY,
   `Quiz_Name` varchar(255) NOT NULL,
   `Quiz_Author_ID` int NOT NULL,
   `Quiz_Available` boolean NOT NULL,
   `Quiz_Duration` int NOT NULL
);
```

```
CREATE TABLE `Quiz_Student` (
   `Attempt_ID` int AUTO_INCREMENT,
   `Student_ID` int NOT NULL,
   `Quiz_ID` int NOT NULL,
   `Score` int NOT NULL,
   `Max_Score` int NOT NULL,
   `Date_of_Attempt` date NOT NULL,
   PRIMARY KEY (`Attempt_ID`, `Student_ID`, `Quiz_ID`)
);

CREATE TABLE `Question` (
   `Question_ID` int PRIMARY KEY NOT NULL,
   `Question_String` varchar(255) NOT NULL
);

CREATE TABLE `Quiz_Question` (
   `Question_ID` int AUTO_INCREMENT,
   `Quiz_ID` int NOT NULL,
   PRIMARY KEY (`Question_ID`, `Quiz_ID`)
);

CREATE TABLE `Answers` (
   `Answer_ID` int PRIMARY KEY NOT NULL,
   `Answer_String` varchar(255) NOT NULL
);

CREATE TABLE `Question_Answers` (
   `Answer_ID` int AUTO_INCREMENT,
   `Question_ID` int NOT NULL,
   `Is_Correct` boolean NOT NULL,
   PRIMARY KEY (`Answer_ID`, `Question_ID`)
);

CREATE TABLE `Quiz_Deleted_Log` (
   `Log_ID` int PRIMARY KEY AUTO_INCREMENT,
   `Staff_ID` int NOT NULL,
   `Quiz_ID` int NOT NULL,
   `Date_Time` datetime NOT NULL
);

ALTER TABLE `Quiz` ADD FOREIGN KEY (`Quiz_Author_ID`) REFERENCES
`users` (`id`);
ALTER TABLE `Quiz_Student` ADD FOREIGN KEY (`Student_ID`) REFERENCES
`users` (`id`);
```

```
ALTER TABLE `Quiz_Student` ADD FOREIGN KEY (`Quiz_ID`) REFERENCES
`Quiz` (`Quiz_ID`) ON DELETE CASCADE;

ALTER TABLE `Question` ADD FOREIGN KEY (`Question_ID`) REFERENCES
`Quiz_Question` (`Question_ID`) ON DELETE CASCADE;
ALTER TABLE `Quiz_Question` ADD FOREIGN KEY (`Quiz_ID`) REFERENCES
`Quiz` (`Quiz_ID`) ON DELETE CASCADE;

ALTER TABLE `Answers` ADD FOREIGN KEY (`Answer_ID`) REFERENCES
`Question_Answers` (`Answer_ID`) ON DELETE CASCADE;
ALTER TABLE `Question_Answers` ADD FOREIGN KEY (`Question_ID`)
REFERENCES `Quiz_Question` (`Question_ID`) ON DELETE CASCADE;


DROP TRIGGER IF EXISTS quizdeleting;
DELIMITER //
CREATE TRIGGER quizdeleting
AFTER DELETE
   ON
       Quiz FOR EACH ROW
BEGIN
   INSERT INTO Quiz_Deleted_Log (Staff_ID, Quiz_ID, Date_Time) VALUES
(@staff, OLD.Quiz_ID, CURRENT_TIMESTAMP);
END; //
DELIMITER ;

DROP PROCEDURE IF EXISTS failingstudents;
DELIMITER //
CREATE PROCEDURE failingstudents
()
BEGIN
   SELECT  users.username,
           Quiz_Student.Score,
           Quiz_Student.Max_Score,
           Quiz_Student.Quiz_ID
   FROM
           Quiz_Student
   INNER JOIN
           users
   ON
           users.id = Quiz_Student.Student_ID
   AND
           (Quiz_Student.Score/Quiz_Student.Max_Score) < 0.4;
END; //
DELIMITER ;
```

5 Part D: The Application

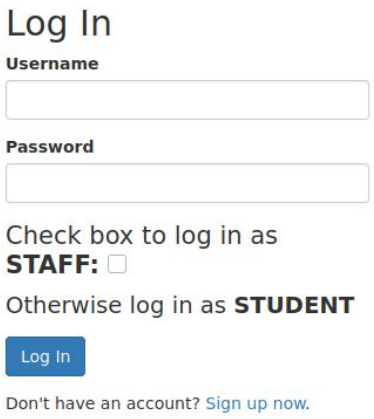
5.1 Introduction to section.

I have added in the zip file, apart from this report, my PHP app files + the files used for SQL, those are the main quizdb.sql , which if ran, it will reset the whole database with the correct tables, foreign keys etc. The db_backup.sql already contains some premade data from me, to test the functionality of the PHP and SQL.

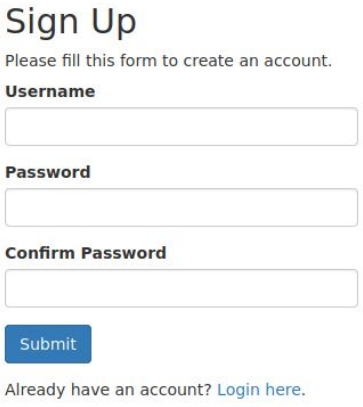
The premade users are
Username: Horia - Password: 123
Username: John - Password: 1234

5.2 User guide on how to use your application, you may wish to include screenshots.

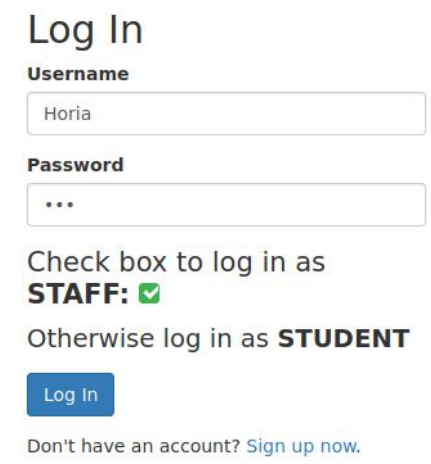
*** To go to this first page, go to login.php**



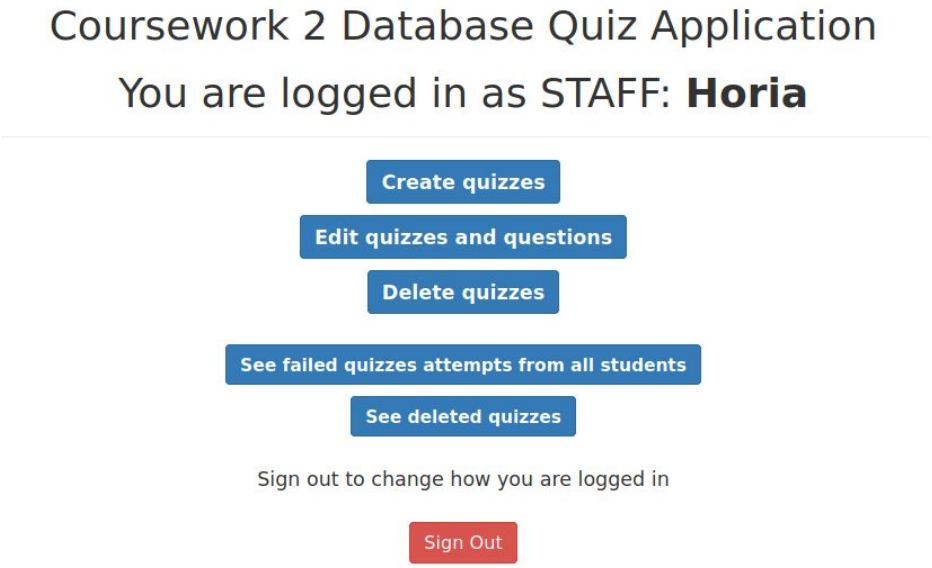
*** First you have to create an account, select Sign up now**



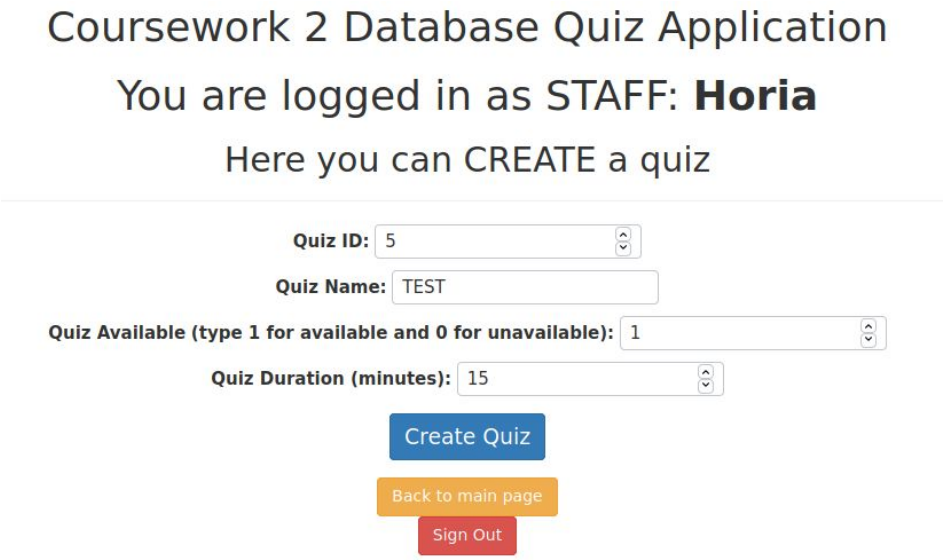
*** Here you can create an account**



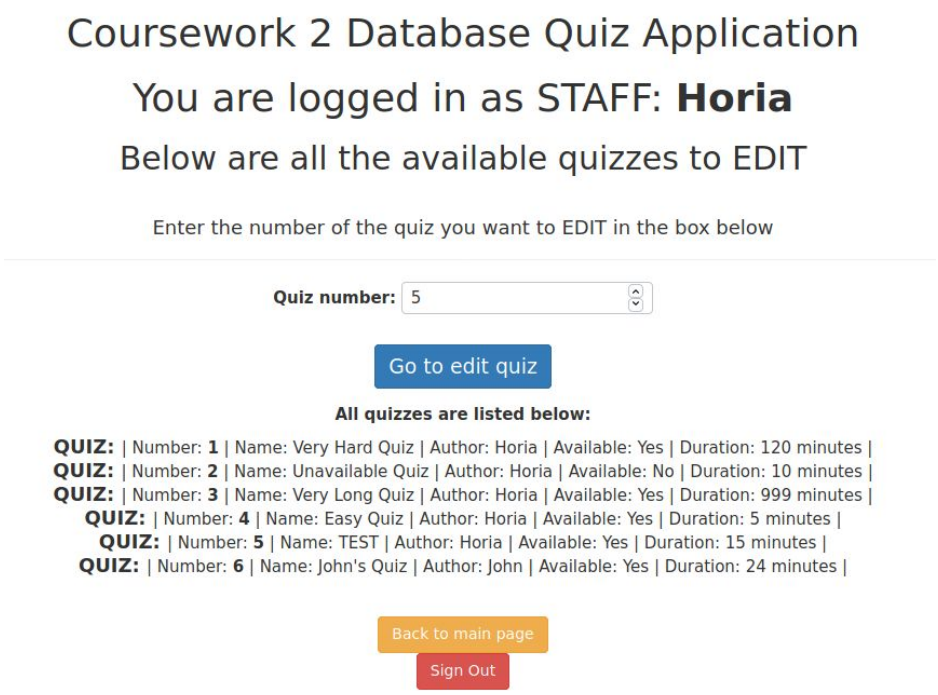
*** Use an account to log in and check the box to log in as STAFF**



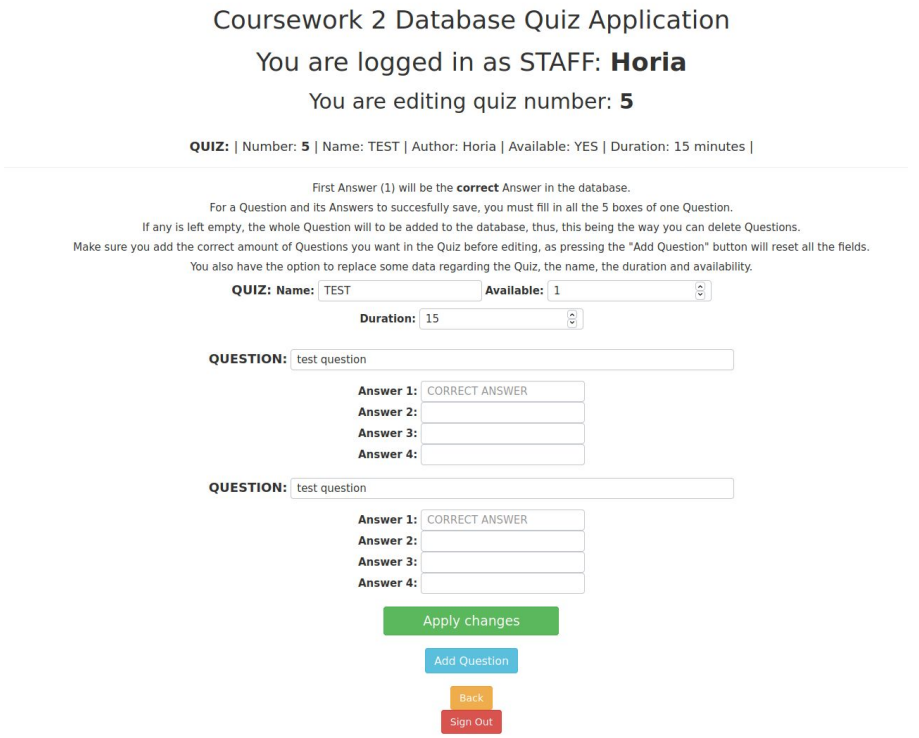
*** Once you are logged in, you will be in the main menu for STAFF, you can get into 5 more pages from here, or just sign out**



*** The Create quizzes page will let you create quizzes, with an ID that was not used before, a name for the quiz, whether it's available (1) or unavailable (0) and the duration of the quiz.**



*** After you have a quiz, you can edit it in the Edit Quizzes and Questions, just select the ID of the quiz you want to edit and press Go to edit quiz**

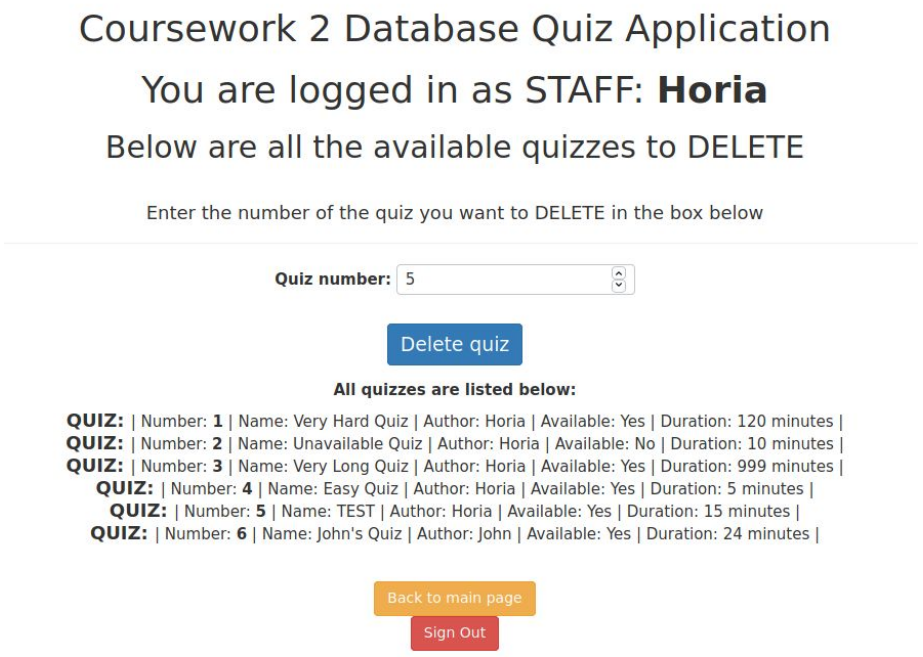


*** Here you can change some data of the Quiz, Add Questions and write the Questions and Answers**

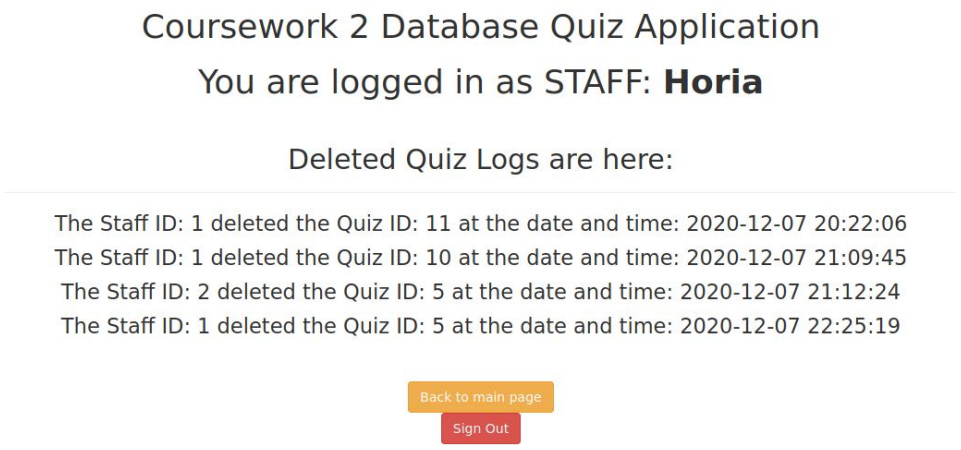
*** To successfully insert a Question in the Quiz, you have to complete all the input boxes for a Question, including 4 answers, otherwise the data won't be saved in the database.**

*** To delete a question simply leave it blank and Apply changes**

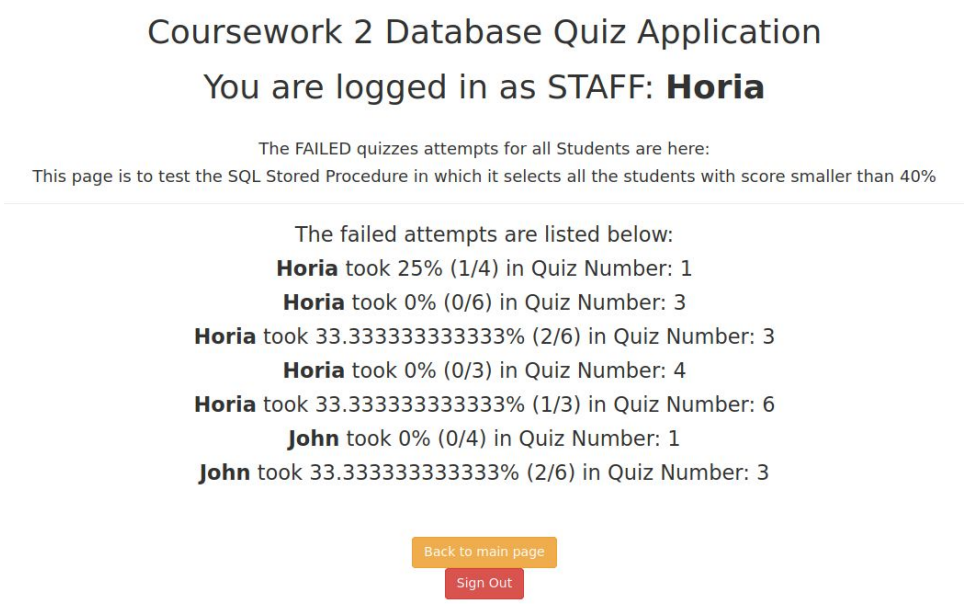
*** Keep in mind that the correct answer will always be Answer 1 in the Edit Quiz (when someone will take the quiz, the answers will be shuffled, so the correct answer will not always be the first)**
*** Make sure you add the desired amount of questions before starting to enter the input, as pressing the Add Question button will reset what you have already written**
*** If the Quiz already contains Questions and Answers, those will be displayed when you edit it, so that you can change them, delete them or simply add more questions.**
*** When you are done with how you want your Quiz to look like, hit Apply changes, and your Quiz will be saved, if you hit Back there won't be any changes made.**



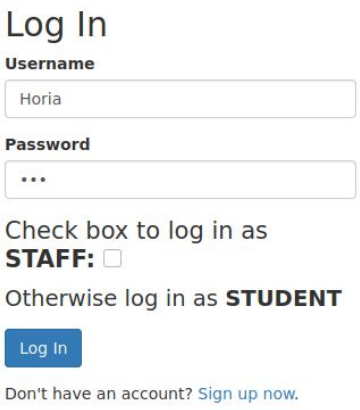
*** If you go to the Delete Quizzes page, you can delete any Quiz you want just by entering its ID Number and pressing Delete quiz**



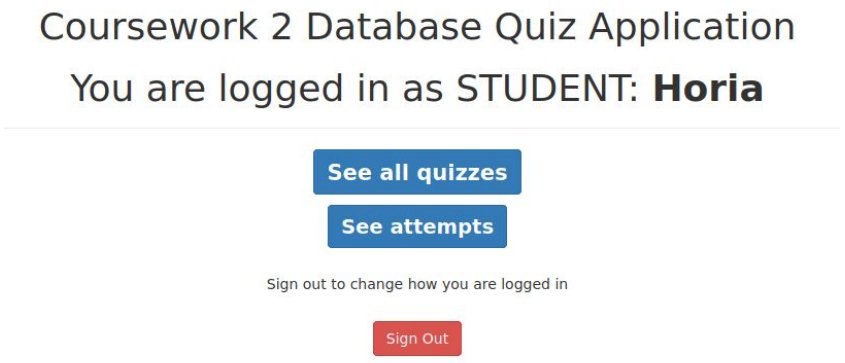
*** This page I made to display and test the trigger that will log the staff id, the quiz id and the current date and time,when a staff user deletes a quiz, which was asked of us in Part E**



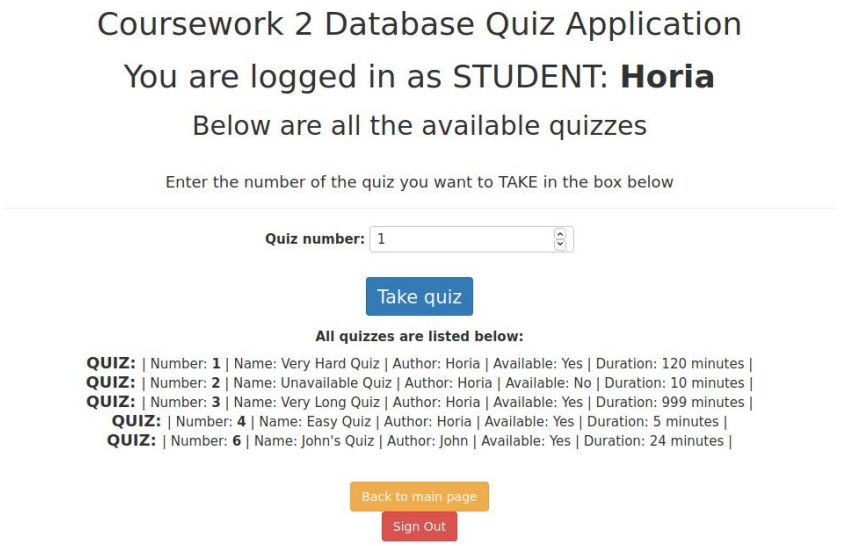
*** This page I made to display and test the stored procedure that displays the student names and their scores for the quizzes where they achieved less than 40%**



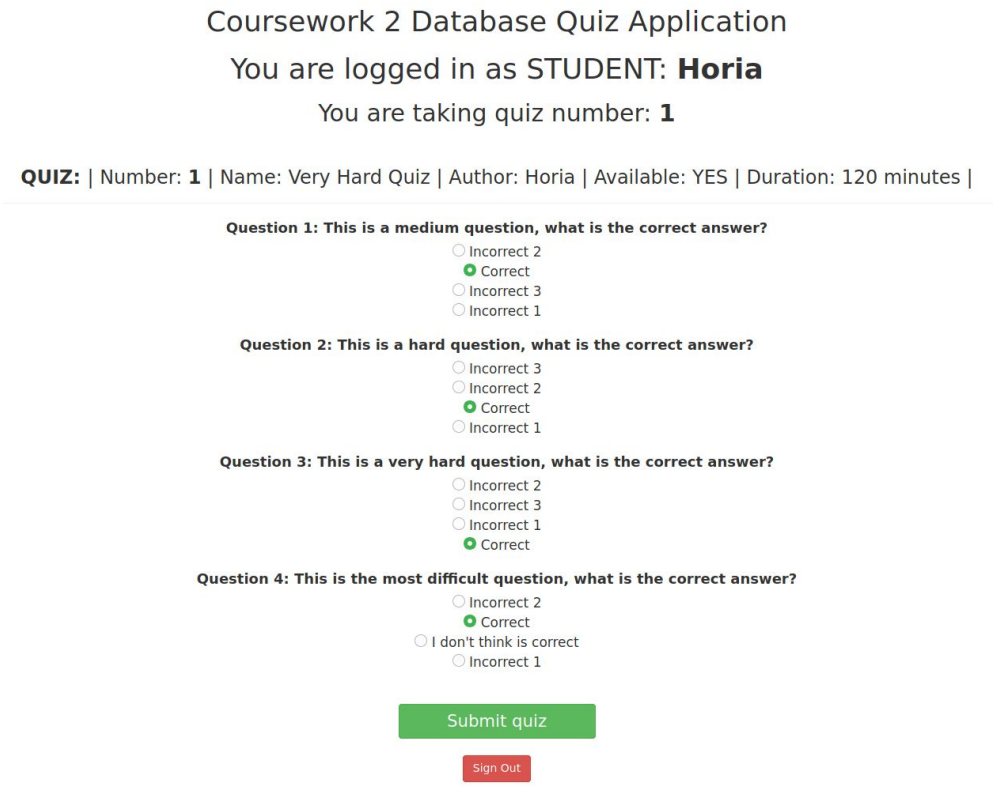
*** You can Sign Out if you want to change to a STUDENT account, just press Log In without ticking STAFF check box**



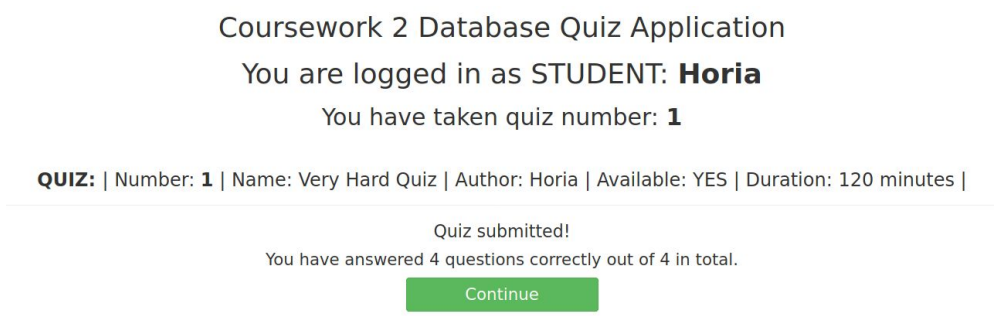
*** Here you will get to the Student Main Menu**



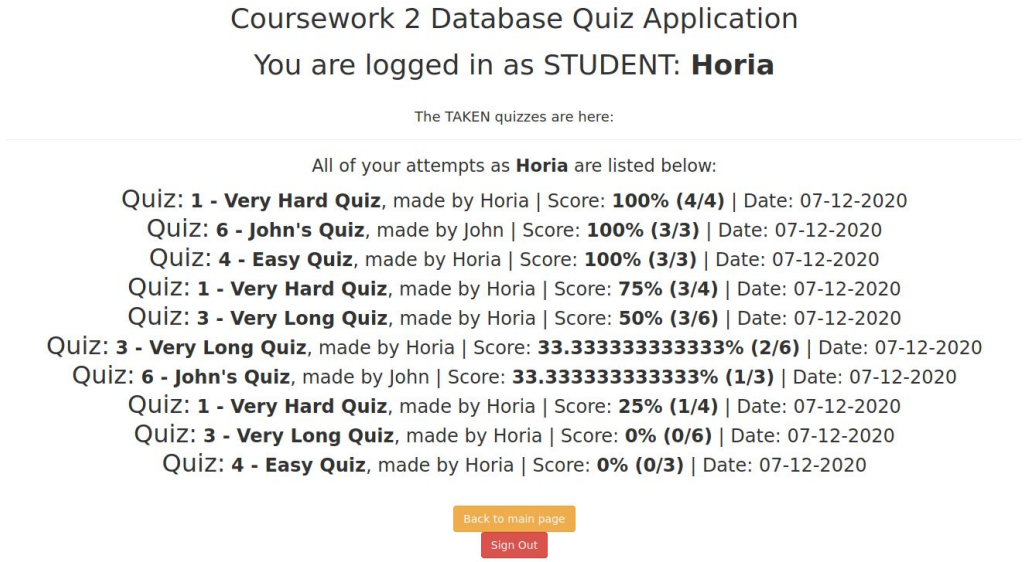
*** You can go to See all quizzes to see all the Quizzes in the database**
*** If you want to Take a quiz just enter its ID Number and press the Take quiz button**



*** Once you get here the Quiz will be displayed, with randomly arranged answers**
*** Now just take the quiz**
*** After you are finished selecting the Answers, you can hit Submit quiz to submit the quiz**



*** After that, you will be prompted with a results page, that will tell you how many answers you got right and wrong**
*** Just hit Continue**



*** In the See attempts page, you can see all your attempts on all the quizzes, with your score displayed also as a percentage**
*** You can see only the attempts made by the current user you are logged in as**

---

6 Part E: Stored Procedures and Triggers

6.1 Introduction to section

I have written the stored procedure and the trigger in the pdf, but also in the source files so that I can test them.
They both include a IF EXISTS statement so that they can be run easily without problem of being overwritten or overwriting any other procedures or triggers, and to also be easy to test on my app.
I have written this DELIMITER so that it is easier to include those two in my SQL database creation.

6.2 MySQL used to create the stored procedure:

```
DROP PROCEDURE IF EXISTS failingstudents;
DELIMITER //
CREATE PROCEDURE failingstudents
()
BEGIN
   SELECT  users.username,
```

```
            Quiz_Student.Score,
            Quiz_Student.Max_Score,
            Quiz_Student.Quiz_ID
    FROM
            Quiz_Student
    INNER JOIN
            users
    ON
            users.id = Quiz_Student.Student_ID
    AND
            (Quiz_Student.Score/Quiz_Student.Max_Score) < 0.4;
END; //
DELIMITER ;
```

6.3 MySQL used to create the trigger:

```
DROP TRIGGER IF EXISTS quizdeleting;
DELIMITER //
CREATE TRIGGER quizdeleting
AFTER DELETE
    ON
        Quiz FOR EACH ROW
BEGIN
    INSERT INTO Quiz_Deleted_Log (Staff_ID, Quiz_ID, Date_Time) VALUES
(@staff, OLD.Quiz_ID, CURRENT_TIMESTAMP);
END; //
DELIMITER ;
```