COMP23111 Database Systems Coursework 2- Advanced Databases

Samuel Zureick

3 December 2021

Table of Contents

Part A: Normalisation

Section Introduction

In this coursework, we are provided with a loose idea of the sort of data we are going to be storing and how we are going to store this. The data is presented in a way that makes it easy to implement a very unorganized and inefficient structure. This next section will be focused on "normalising" the information source that we are presented with.

UNF

For my UNF, I essentially copied the plain structure given by the information source into a table, and didn't do much else. Each quiz has a ton of attributes, and many of these are not atomic, like the attempt dates and the answers attributes. Also, there are repeated attributes, shown by Question1 and Question2. There are a lot of functional dependencies, and there are many ways that this initial table can be broken up. However, this isn't really a result of logic, it is just basically tabulating the information schema. This however does have a primary key, which is the quiz ID.

Quiz ID
Quiz Name
Author
Available
Duration
Question1
Answers1
Question2
Answers2
...

Student ID
Student Name
Attempt Dates
Scores

1NF

For 1NF, I split the table up to remove all of the repeating groups. One of the repeating groups was the question and answer section, so I split this into a relation containing the question number, the question, and the different answers, along with the correct answer. Now there is also a relation detailing the student and their score with the attempt date, so this field also will not have to repeat in the schema. All of the values now are also atomic, and this was achieved by making the new relation with the compound key of quiz ID and question number.

Quiz ID
Quiz Name
Author
Available
Duration

Quiz ID
Question Number
Question
A1
A2
A3
A4
Correct Answer

Quiz ID
Student ID
Attempt Number
Attempt Date
Score
Student Name

2NF

To translate this into 2NF, we must start with our 1NF relation. Then I had to eliminate partial dependency. In the above relations, Student name was only partially dependent on the compound primary key of quiz ID and student ID, so this was added to its own relation to eliminate this partial dependency. I also split the relation modelling the question and answers into two, one which just stores the question for the quiz given the question number and quiz ID, and the other which stores the different answers for this question and whether they are correct.

Quiz ID
Quiz Name
Author
Available
Duration

Quiz ID
Question Number
Question

Quiz ID
Question Number
Answer Number
Answer
Correct?

Quiz ID
Student ID
Attempt Number
Attempt Date
Score

Student ID
Student Name

3NF

For 3rd normal form, I have to start with a set of tables in 2nd normal form. My goal is then to get rid of all transitive dependencies. During my conversion from 1NF to 2NF, I believe that I eliminated all of the transitive dependencies, so there is nothing to show for this step but the same set of tables as above.

Quiz ID
Quiz Name
Author
Available
Duration

Quiz ID
Question Number
Question

Quiz ID
Question Number
Answer Number
Answer
Correct?

Quiz ID
Student ID
Attempt Number
Attempt Date
Score

Student ID
Student Name

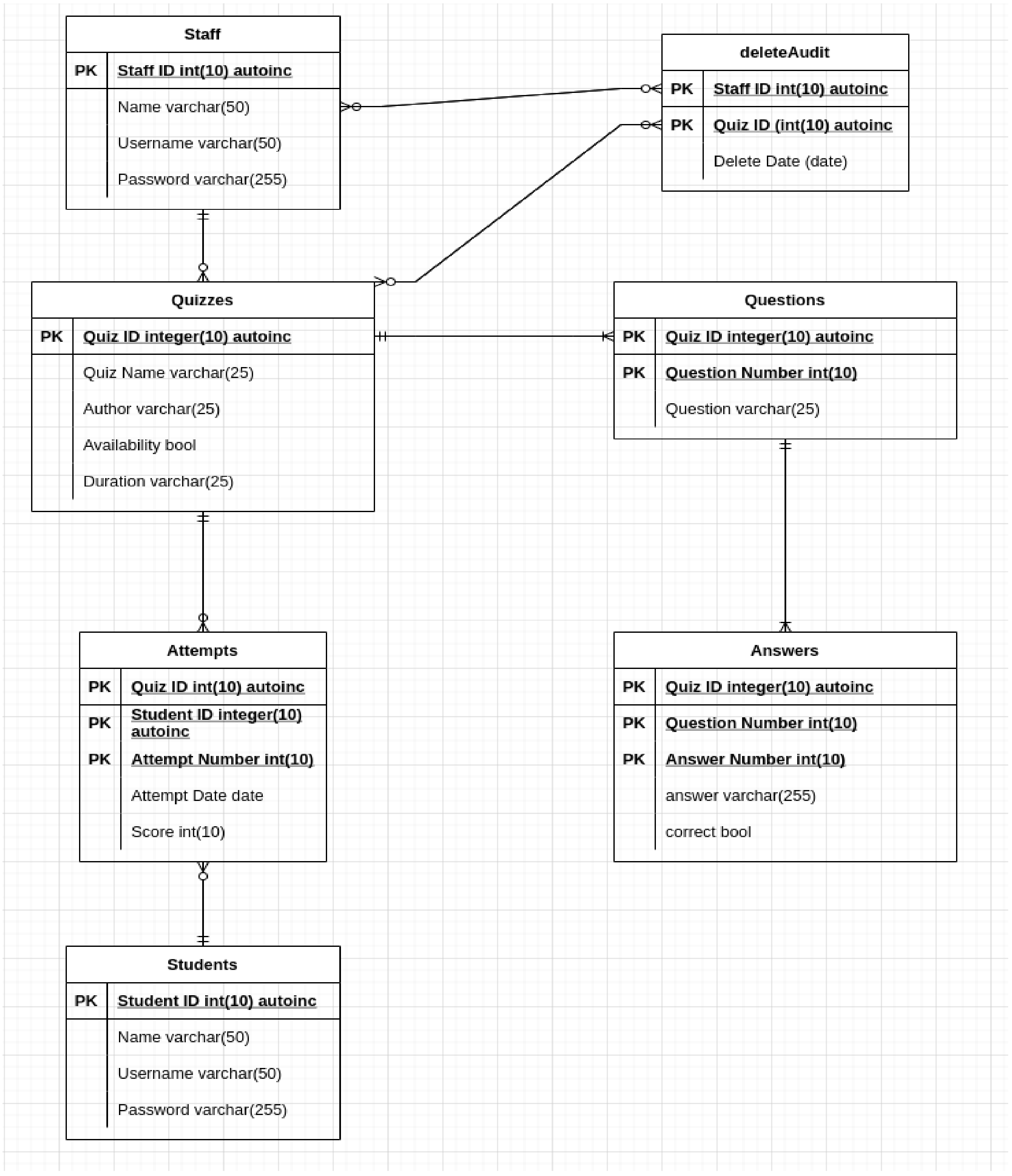
Part B: Relational Schema

Section Introduction

In this section I will present my relational schema. This will be different from the tables I presented in the normalization section because here I have to consider some aspects outside of the provided data description, such as passwords, staff users, etc. I use crow's foot notation to show how each table relates to each other.

Some differences here from the original source include a staff table (essentially mirroring student table) and passwords for both the student and staff tables, and a deleteAudit table that gets updated as a result of the delete trigger implemented in part E. The staff table is necessary because there needs to be an account type that allows quizzes to be created and modified; that was not presented in the data spec. Other than these changes made to allow for a login system and auditing, the schema is pretty similar to the tables presented above.

Relational Schema



Part C: Implementation

Section Introduction

In this section, I detail my sql implementation of my relational schema. It was a pretty simple translation, but one thing that can be difficult is identifying foriegn and primary keys given the ER diagram. In my implementation, if in the relational schema two tables are connected by a relation and they share an attribute as one of their primary keys, a foreign key exists between the two. Similar to how I created my schema, I began by creating the relations and then started establishing constraints on the relations, to ensure that I have a correct structure before connecting everything together.

SQL Statements

```
-------------------------------------
-- answers table
-------------------------------------

CREATE TABLE answers (
  qID int NOT NULL AUTO_INCREMENT,
  qNo int NOT NULL,
  ansNo int NOT NULL,
  answer varchar(511) NOT NULL,
  correct tinyint(1) NOT NULL DEFAULT '0'
) ENGINE=InnoDB COLLATE=utf8_unicode_ci;

-------------------------------------
-- attempts table
-------------------------------------

CREATE TABLE attempts (
  qID int NOT NULL AUTO_INCREMENT,
  sID int NOT NULL,
  attemptNo int NOT NULL,
  attemptDate date NOT NULL,
  score int NOT NULL DEFAULT '0'
) ENGINE=InnoDB DEFAULT COLLATE=utf8_unicode_ci;



-------------------------------------
-- deleteAudit table
```

```
-------------------------------------

CREATE TABLE deleteAudit (
  sID int NOT NULL,
  qID int NOT NULL,
  delDate datetime NOT NULL
) ENGINE=InnoDB DEFAULT COLLATE=utf8_unicode_ci;


-------------------------------------
-- questions table
-------------------------------------

CREATE TABLE questions (
  qID int NOT NULL AUTO_INCREMENT,
  qNo int NOT NULL,
  question varchar(511) NOT NULL
) ENGINE=InnoDB COLLATE=utf8_unicode_ci;

-------------------------------------
-- quizzes table
-------------------------------------

CREATE TABLE quizzes (
  qID int NOT NULL AUTO_INCREMENT,
  qName varchar(30) NOT NULL DEFAULT 'My Quiz',
  author varchar(100) DEFAULT NULL,
  availability tinyint(1) NOT NULL DEFAULT '0',
  duration varchar(50) DEFAULT NULL
) ENGINE=InnoDB DEFAULT COLLATE=utf8_unicode_ci;

-------------------------------------
-- staff table
-------------------------------------

CREATE TABLE staff (
  staffID int NOT NULL AUTO_INCREMENT,
  username varchar(50) NOT NULL,
  name varchar(100) NOT NULL,
  pw varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT COLLATE=utf8_unicode_ci;

-------------------------------------
-- students table
```

```
-------------------------------------

CREATE TABLE students (
  sID int NOT NULL AUTO_INCREMENT,
  username varchar(255)  NOT NULL,
  name varchar(100) DEFAULT NULL,
  pw varchar(255) NOT NULL
) ENGINE=InnoDB COLLATE=utf8_unicode_ci;

-------------------------------------
-- Indexes
-------------------------------------
ALTER TABLE answers
  ADD PRIMARY KEY (qID,qNo,ansNo),
  ADD KEY qnoFK (qNo);

ALTER TABLE attempts
  ADD PRIMARY KEY (qID,sID,attemptNo),
  ADD KEY qIDSID (sID);

ALTER TABLE deleteAudit
  ADD PRIMARY KEY (sID,qID);

ALTER TABLE questions
  ADD PRIMARY KEY (qID,qNo),
  ADD KEY qNo (qNo);

ALTER TABLE quizzes
  ADD PRIMARY KEY (qID);

ALTER TABLE staff
  ADD PRIMARY KEY (staffID),
  ADD UNIQUE KEY username (username);

ALTER TABLE students
  ADD PRIMARY KEY (sID),
  ADD UNIQUE KEY username (username);



------------------------------------
-- Constraints
------------------------------------
ALTER TABLE answers
```

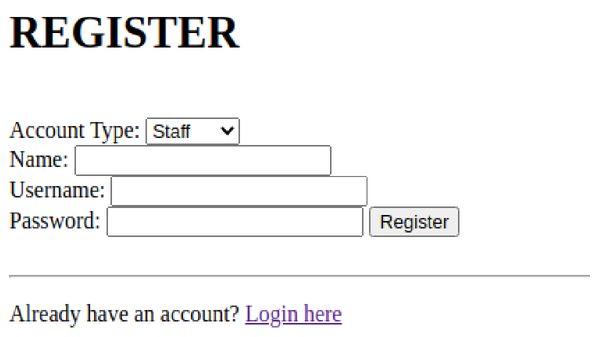
```
  ADD CONSTRAINT `qnoFK` FOREIGN KEY (`qNo`,`qID`) REFERENCES
`questions` (`qNo`, `qID`) ON DELETE CASCADE ON UPDATE CASCADE;


ALTER TABLE attempts
  ADD CONSTRAINT qiDDel FOREIGN KEY (qID) REFERENCES quizzes (qID)
ON DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT qIDSID FOREIGN KEY (sID) REFERENCES students (sID)
ON DELETE CASCADE ON UPDATE CASCADE;


ALTER TABLE questions
  ADD CONSTRAINT qIDFK FOREIGN KEY (qID) REFERENCES quizzes (qID)
ON DELETE CASCADE ON UPDATE CASCADE;
```

Part D: The Application

Section Introduction

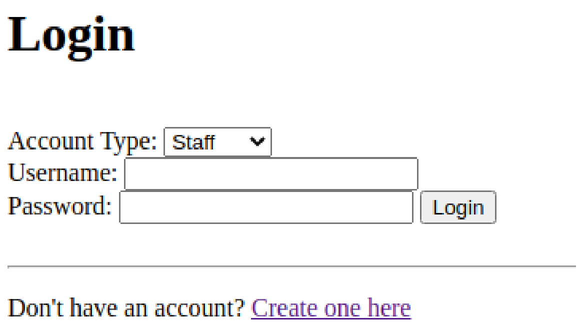
In this section, I detail my web application and explain how to use it. Everything is accomplished using php, sql, and html. My application connects to the Uni database and is connected through Git, so it can be accessed by others as long as they are on the University network or vpn. I did not focus much on the user interface, I simply wanted to make an application that was intuitive and worked well.

User Guide: Staff


REGISTER

Account Type: Staff
Name:
Username:
Password: Register

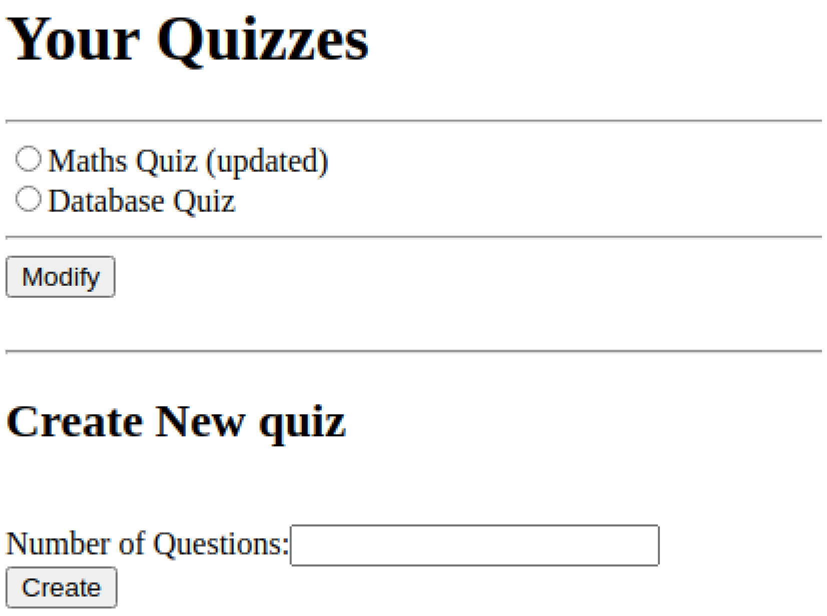
Already have an account? Login here


Login

Account Type: Staff
Username:
Password: Login

Don't have an account? Create one here

The user is first presented with the register page, where they can provide a name, username, and password for registration, or can choose to login to an existing account. During registration, the application checks to make sure an account with this username doesn't exist, and uses php's built in password hash and verify functions for login so no sensitive information is stored in the database.



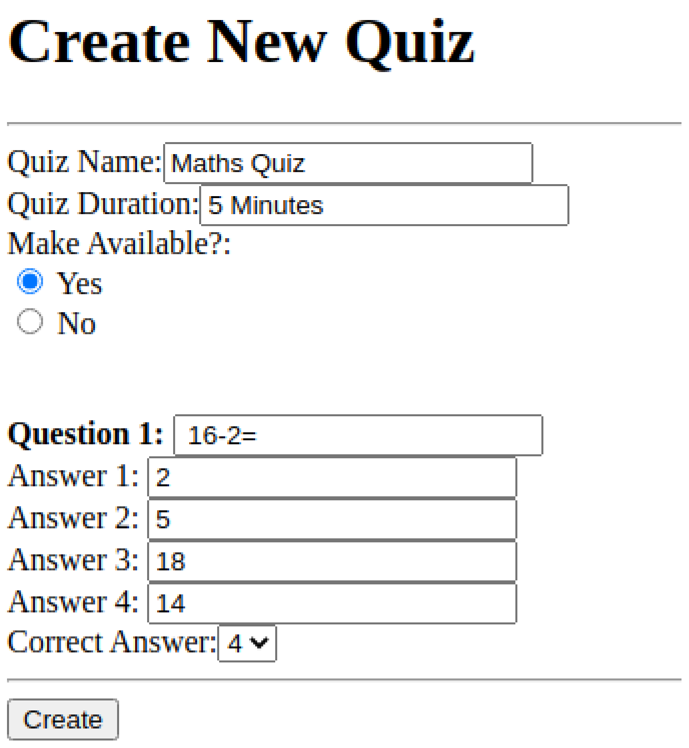
Your Quizzes

○ Maths Quiz (updated)
○ Database Quiz

Create New quiz

Number of Questions:

The staff, if successfully signed in, is taken to a dashboard where they can either edit quizzes that they have already made or create a new quiz of a certain size.

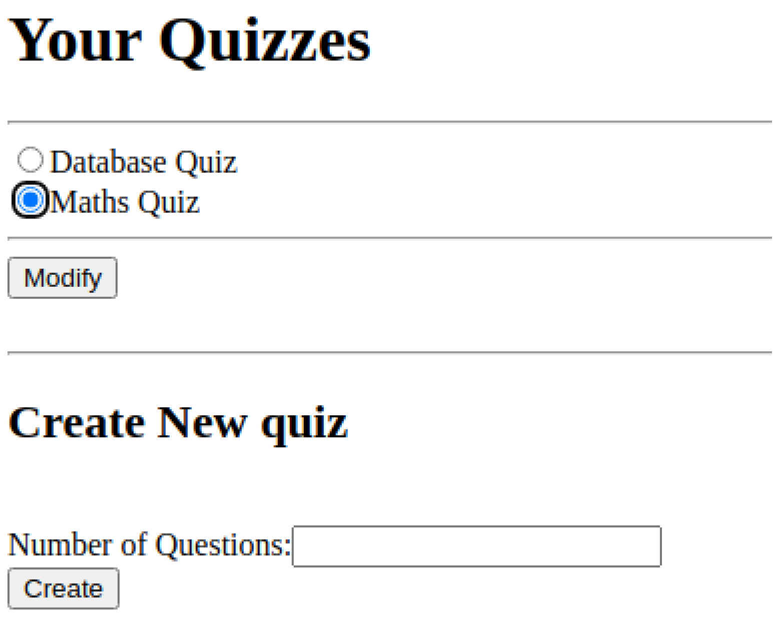


Create New Quiz

Quiz Name: Maths Quiz
Quiz Duration: 5 Minutes
Make Available?:
[✓ Yes]
No

Question 1: 16-2=
Answer 1: 2
Answer 2: 5
Answer 3: 18
Answer 4: 14
Correct Answer: 4

If the user chooses to create a new quiz of length 1, they will be presented with this page, which has text fields for them to fill out to create the quiz. Every section is required to submit. After submission, the quiz is posted to a quiz processing script, and the user is then redirected to their dashboard.



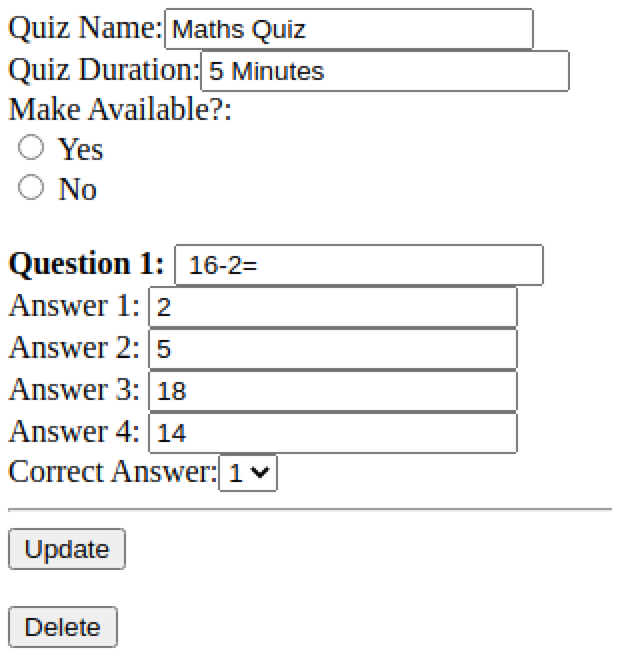
Your Quizzes

Database Quiz
[✓ Maths Quiz]

Create New quiz

Number of Questions:

The staff member can also select a radio button and modify a quiz that they have created.

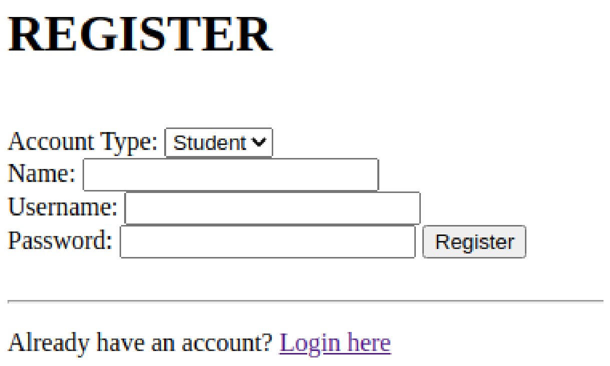


Quiz Name: Maths Quiz
Quiz Duration: 5 Minutes
Make Available?:
Yes
No

Question 1: 16-2=
Answer 1: 2
Answer 2: 5
Answer 3: 18
Answer 4: 14
Correct Answer: 1

Selecting the "Maths Quiz" radio button will take the user to this page, where they can edit the name, duration, availability, questions, answers, and correct answers for a quiz. The user also has the ability to delete a quiz. Because of my database design and the triggers that have been implemented, if a quiz is deleted, so are the entries referring to it's unique ID in Questions, Answers, and Attempts, and the quiz id, staff id, and date and time are logged in the audit table.

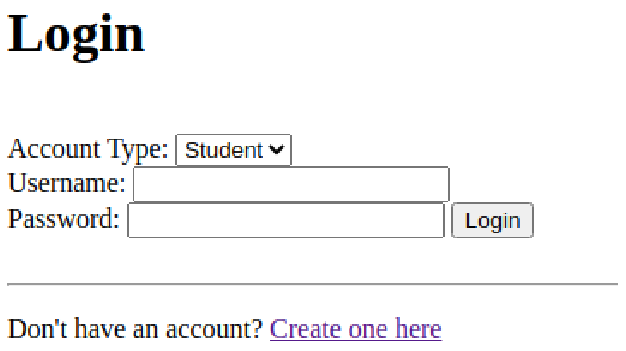
User Guide: Student



REGISTER

Account Type: Student
Name:
Username:
Password:

Already have an account? Login here

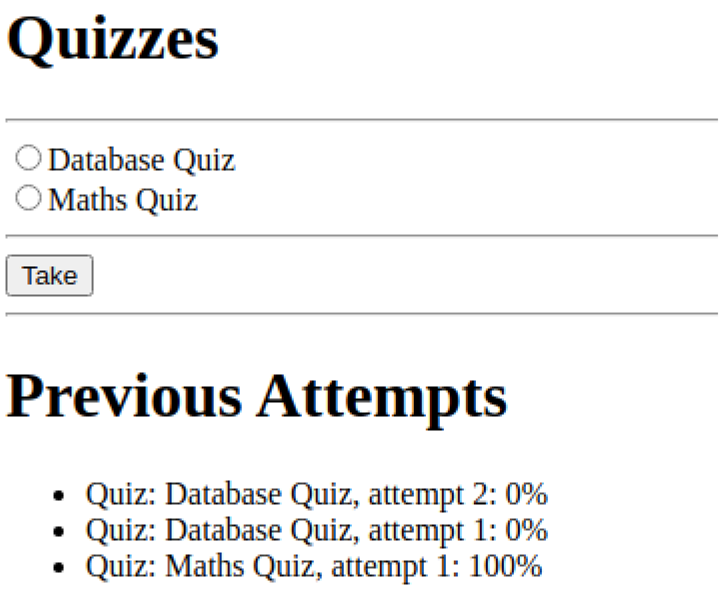


Login

Account Type: Student
Username:
Password:

Don't have an account? Create one here

Similar to the staff, the student is greeted with a register page and given the option to log in if they have an account already. The logic, security, and code for the most part is the same for student and staff, the data just gets stored in different tables and the accounts have access to different parts of the application depending on the account type.



Quizzes

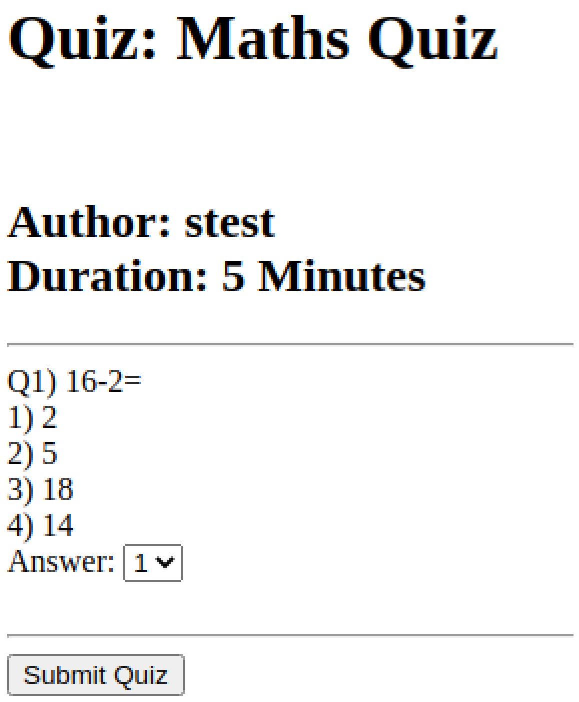
Database Quiz
Maths Quiz

Take

Previous Attempts

- Quiz: Database Quiz, attempt 2: 0%
- Quiz: Database Quiz, attempt 1: 0%
- Quiz: Maths Quiz, attempt 1: 100%

Once at the student dashboard, the user can see their previous attempts listed below, which details the quiz name, attempt number, and score. The student can also press radio buttons to select a quiz from the list of available quizzes to take.



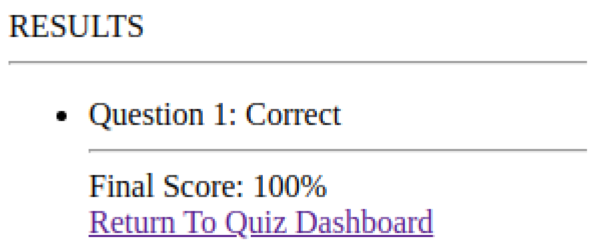
Quiz: Maths Quiz

Author: stest
Duration: 5 Minutes

Q1) 16-2=
1) 2
2) 5
3) 18
4) 14
Answer: 1

Submit Quiz

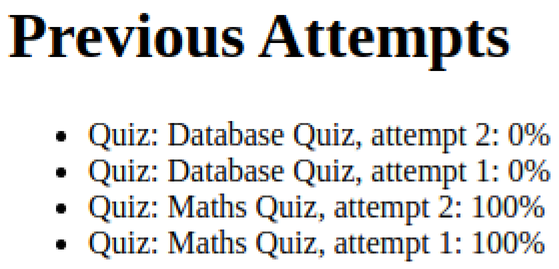
The student is given the name, author duration, and then the questions are listed with answer options and a drop down menu to select answers.


RESULTS

- Question 1: Correct

Final Score: 100%
Return To Quiz Dashboard

The quiz is then graded, and a result report is outputted, for each question telling the student if they were correct or incorrect, and then providing the final score. When the submitted quiz is processed, the attempts table is updated with this latest attempt.


Previous Attempts

- Quiz: Database Quiz, attempt 2: 0%
- Quiz: Database Quiz, attempt 1: 0%
- Quiz: Maths Quiz, attempt 2: 100%
- Quiz: Maths Quiz, attempt 1: 100%

Upon returning to the dashboard, the previous attempts section should be updated with the most recent attempt.

Part E: Stored Procedures and Triggers

Section Introduction

In this part, I was tasked with creating one stored procedure and one trigger. The stored procedure is basically a function that outputs the Name, quiz name, and quiz grade for all students who have achieved under 40% correct on a quiz. This is done by joining my name, attempts, and quiz tables together, only selecting those rows where score is under 40, and then outputting name, quiz name, and score. The trigger is an audit trigger that takes log of teacher name, quiz id, and date and time when a quiz is deleted, and this is accomplished by inserting into an audit table the current date and time, the old quiz id, and the staff ID associated with the author of the quiz.

Stored Procedures & Triggers MySQL

```
DELIMITER //

CREATE PROCEDURE under40()
  BEGIN
    SELECT
      qName, name, score
    FROM
      quizzes
        natural join
          (attempts
            NATURAL JOIN
              students)
    WHERE
      score < 40;
END //
DELIMITER ;

-------------------


DELIMITER //
CREATE TRIGGER
  after_quiz_delete
AFTER
  DELETE
ON
  quizzes FOR EACH ROW
BEGIN
  DECLARE staffid INT;

  SELECT
    MAX(sID)
  INTO
    staffid
  FROM
    staff
  WHERE
    name = OLD.author;

  INSERT INTO
    deleteAudit(sID, qID, delDate)
  VALUES
    (staffid, OLD.qID, GETDATE());
END //
```

```
DELIMITER ;
```