

CSCB20 - Databases and Web Applications

ASSIGNMENT 3 REPORT

Name: Sibong, Dann Sioson
Student Number: 1003400269, 1001346377
Instructor: Professor Attarwala
Date: April 6, 2018

Accessing our website

To access our website, please click the following link:

<https://mathlab.utoronto.ca/cscb20/dongsibo/Assignments/Assignment3/htdocs/index.php>

If you (the marker) would like to login. Sample accounts:

- *Instructor account* - utoid: alices, password: momson99
- *TA account* - utoid: taps, password: pseun
- *Student* - utoid: rexoc, password: countybounty

Introduction

In this assignment, we are instructed to generate a personal view of a general “CSCB20: Introduction to Databases and Web Applications” website as in A2. However, we are further challenged in A3 to integrate PHP as an addition to our pre-existing assignment 2 website. For this report, we will be discussing the issues faced during project implementation. This includes lessons learned from assignment 2, issues faced with previous iterations, how we faced these issues, and personal challenges faced by each member.

Lessons learned from Assignment 2

We had other high priorities during our previous assignment which interfered with work flow. Our situation for this assignment was similar due to the approaching end of the term. However all was not for naught. A lesson learned from our previous assignment was to prepare: learn all fundamentals in order to properly handle the assignment. We exercised our fundamentals in both PHP and CSS with practice problems to prepare for A3.

Our first step was to fix our assignment 2 CSS file. After obtaining feedback, we fixed anything that appeared nonfunctional (for example, “ghost” hovering outside of any anchored elements). After these fixes, we were able to commit to a more structured (basic) format. Due to our previous assignment’s messy structure, we learned the importance of modularity and thus made more prudent planning decisions before implementing any actual code.

Issues prior

We decided upon the following schema during our first meeting:

- `accounts(utoid, first_name, last_name, account_type, password)`
A relation that stores all accounts, holding their unique utoid, first name, last name, what account type they are (student, instructor, TA), and their password
- `student_marks(utoid, mat_type, mark, out_of, weight)`
A relation that stores all of the students’ marks, what their mark is out of, and their weight
- `remark_req(utoid, mat_type, why)`
A table that stores all remark requests, specifically what material the student is requesting for, and why they are requesting for it.

- anon_feedback(utorid, last_name, fq1, fq2)

A table that stores all anonymous feedback, where the fq1 is an answer to “What do you dislike about this instructor?”, and fq2 is an answer to “What would you like to change about this course?”. The last name and utorid stored is the instructors’.

We happened upon minor issues once we started implementing our schema. An example of such would be student marks; if an instructor submits a mark for a student, how do we, *as people developing*, find out the mark and weight of the assignment? A possible solution would be to hard-code, but it would be immensely rigorous and impractical.

Another issue we ran into was setting the primary keys for our remark requests and anonymous feedback tables. We wanted the primary keys to consist of all table attributes, including the feedback field. However, our feedback field’s data-type was “BLOB”; a text data type that could not be part of a primary key. This left us in a bit of a dilemma.

Though we could not find holes in our database schema on the back-end, we did observe small errors on the front-end portion of our website, which prompted some important questions. Examples of such questions would include the following:

- “If a student were to submit blank feedback, wouldn’t the instructor/TAs be annoyed to review it?”
- “If a student were to submit a remark request for course work they did not submit, how would the instructor/TAs proceed?”
- “Does the instructor even like viewing the marks in raw data, without any parsing?”

We found that questions like these deserved to be addressed (given our CS background). This prompted us to think of possible exceptions that we would have to deal with.

Addressing these issues

After identifying our problems, we quickly communicated and negotiated a workable schema:

- accounts(utorid, first_name, last_name, account_type, password)

- course_materials(mat_type, out_of, weight)

Given this relation, we are able to index marks accordingly with what material is out of and its weight, and we were able to remove two of the attributes in the student marks relation.

- student_marks(utorid, mat_type, mark)

- remark_req(req_num_id, utorid, mat_type, why)

In this relation, we added an attribute “req_num_id” which incrementally increases as each row is entered. Thus, this replaces all attributes as a primary key.

- anon_feedback(num_id, utorid, last_name, fq1, fq2)

Similar to remark requests, num_id is used as an incremental key to dictate the uniqueness of a specified row.

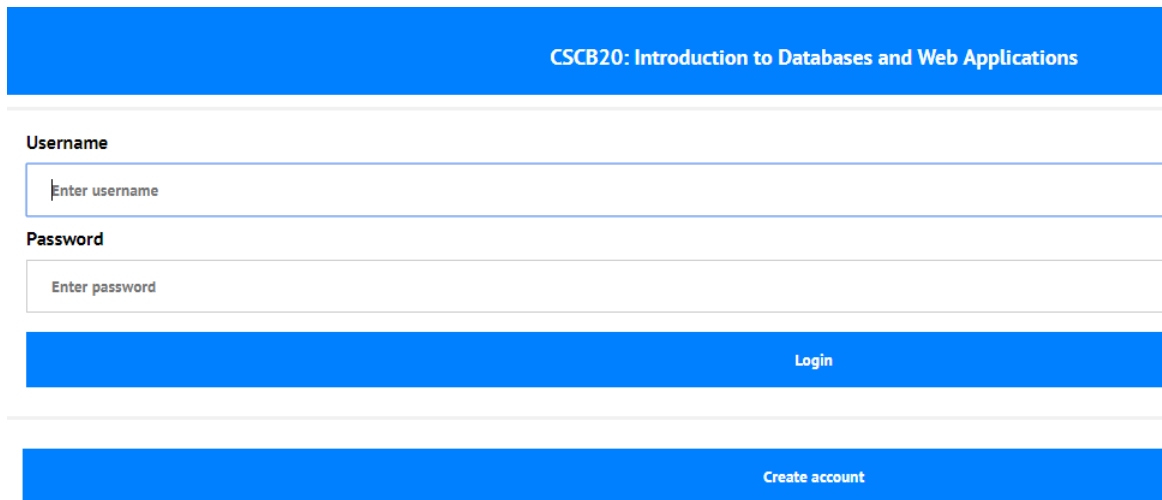
Given this schema, we were able to further triumph over small obstacles, like entering marks from the instructor/TA side, and entering remark requests/anonymous feedback through the student side.

What we were able to do on the front-end was to catch all unreasonable actions from users. Once we caught them, users would be notified by a pop up window stating the unreasonable action along with a page refresh.

Addressing Project Requirements

Main Requirements:

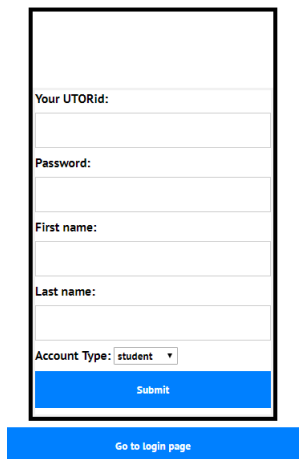
1. Login



The login form features a blue header bar with the text "CSCB20: Introduction to Databases and Web Applications". Below the header, there are two input fields: "Username" and "Password". The "Username" field has a placeholder text "Enter username". The "Password" field has a placeholder text "Enter password". Below the password field is a blue "Login" button. At the bottom of the form is a blue "Create account" button.

We addressed the requirement of a login page by providing a login page. Only users in our student_marks relation are allowed to access all available content.

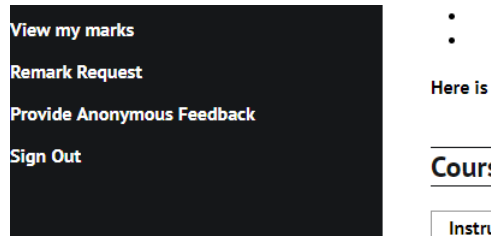
Create Account



The "Create Account" form is a vertical stack of input fields. It starts with "Your UTORid:", followed by "Password:", "First name:", and "Last name:". Below these is an "Account Type:" dropdown menu with "student" selected. At the bottom of the form is a blue "Submit" button. Below the form is a blue button labeled "Go to login page".

We also the issue of new accounts by providing an account creation page.

2. Customizing Based on User Type



We addressed the requirement of displaying different pages to different types of users by creating a special php package that would check the user's account type and display the appropriate pages. Above is a picture of the student perspective, but our package dynamically changes by the type of account logged in.

Welcome Rex Ocounty!

Click on the side to view your marks

Course Overview

Welcome to CSCB20! The text here should be related to CSCB20 in some way! This is more text. Filler text. I

Welcome Alice Son!

Click on the side to view student marks

Course Overview

Welcome to CSCB20! The text here should be related to CSCB20 in some way! This is more text. Filler te

We also changed our homepage greeting based on the user's account type.

3. Fetching Data

Lab 1 Marks

UTORid	First Name	Last Name	Material	Mark	Out of	Weight
rexoc	Rex	Ocounty	Lab 1	3	3	3

Class average: 3.00 / 3 (100.00%)

Assignment 1 Marks

UTORid	First Name	Last Name	Material	Mark	Out of	Weight
clairrog	Clair	Ohno	Assignment 1	50	50	15
cornali	Corna	Li	Assignment 1	44	50	15
rexoc	Rex	Ocounty	Assignment 1	14	50	15
stdoba	Stdman	Board	Assignment 1	40	50	15
student	Student	Student	Assignment 1	20	50	15

Class average: 33.60 / 50 (67.20%)

We addressed the requirement of fetching data by correctly connecting to the database and parsing the information in a meaningful way that could be understood by the browser and reader.

Personal Challenges

Like in the last assignment, time was again a large issue. We had high personal priorities to attend to in addition to this assignment. However, we managed ourselves much more efficiently this time around. We persisted in making conscious efforts to learn HTML/CSS/PHP which we then put forth in our assignment. This, in addition to our strong collaboration allowed us to overcome the restriction put upon us by a lack of time.

Due to the tightly cooperative nature of the project, it was very difficult to make progress when we worked alone. Blindly working on one section of the project with little knowledge of our partner's intentions was a daunting task. We could depend only on the principles of black box programming and faith. This schism was further widened by the difference in proficiency in PHP, as more time that could have been dedicated towards the project was spent reviewing PHP fundamentals and worrying about not knowing PHP fundamentals.