IAT 352: Database-powered Website with User Management

“Education Nation”

A platform that connects students with tutors

Professor: Helmine Serban

TA: Fatemeh Salehian Kia

Lab: D102

Date Submitted: December 3, 2020

Team Members:

Jimmy Chu (301326988)

Terence Liu (301304106)

Theodore Tang (301305214)

1. Prepare the report describing all parts of the system.

Overview of Web Application

For Assignment 3, we have implemented a large majority of what we have proposed in our proposal. Currently, we have two different members (tutor and student), which is set during the sign-up process.

Visitors are able to see an overview of what the company is about, the types of tutors available, and the ability to narrow down available tutors through their price range, subject knowledge, and teaching age groups. When landing on the homepage these visitors will see a generalized view of the homepage, such as a list of the top-rated tutors. We have implemented this function by sending a query to MySQL searching for the top 5 average rated courses. Writing a review will not be available until the visitor creates an account and becomes a member, and instead will pop up a prompt to login or create an account (Figure 1). From the tutor listing page, visitors are able to view all the listed courses. Furthermore, visitors are able to filter and sort through the available courses to fit their needs/requirements (Figure 2 & Figure 4).Moreover, visitors will not be able to add courses/tutors to their cart. Implementation-wise, when a visitor tries to write or create a review for a tutor, they will be prompted with a popup asking them to login or sign up for an account before they can write a review. By checking the sessions, we are able to determine if the user is a visitor or a member. In addition, there are account settings pages that retrieve current user's data and allow them to make updates to their information. Again by using sessions to check if the user is a member or a visitor, we update the login and sign up section of the top navigation bar to instead display a pill dropdown allowing the user to access the account settings page (Figure 3).

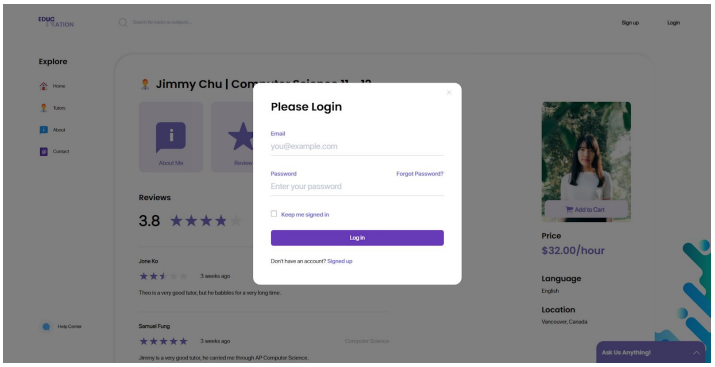


Figure 1: Login popup dialog

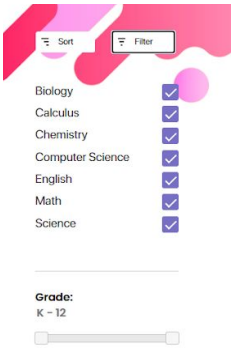


Figure 2: Filtering and Sorting Feature on the tutor listing page

Figure 3: Login or Sign up top navigation

```
// XHR object
var myReq = getXMLHTTPRequest();

// function to get XHR object, should works on all browser
function getXMLHTTPRequest() {
  var req = false;
  try
  {
    // for fire fox
    req = new XMLHttpRequest();
  } catch (err) {
    try
    {
      // for some versions of IE
      req = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (err) {
      try
      {
        // for some other versions of IE
        req = new ActiveXObject("Microsoft.XMLHTTP");
      } catch (err) {
        req = false;
      }
    }
  }

  return req;
}
```

```
// AJAX for tutor listing cards
var course = null;
var grade = "";
var sortby = "";

// call when sort link is clicked
function sortBy(sort) {
  event.preventDefault();
  sortby = sort;
  sortAndFilter();
  return false;
}

// couse when course check box is selected
function courseSelect() {
  let courses = document.getElementsByName("courses[]");
  course = "";
  courses.forEach(checkCourse);
  sortAndFilter();
}

function checkCourse(item) {
  if (item.checked) {
    course = course + item.value + "-";
  }
}
```

```
function sortAndFilter() {
  let targetPage = 'shared/tutorListCards.php';
  let myRand = parseInt(Math.random() * 99999999999999999);
  let theURL = targetPage + "?rand=" + myRand;

  // add filter condition
  if (sortby != "") {
    theURL = theURL + "&sortby=" + sortby;
  }

  if (grade != "") {
    theURL = theURL + "&grade=" + grade;
  }

  if (course != null) {
    theURL = theURL + "&course=" + course;
  }


  myReq.open("GET", theURL, true);
  myReq.onreadystatechange = theHTTPResponse;
  myReq.send();
}

function theHTTPResponse() {
  if (myReq.readyState == 4) {
    if (myReq.status == 200) {

      var myObj = JSON.parse(myReq.responseText);
      document.getElementById("cardLists").innerHTML = myObj.newList;
    }
  }
}
```

```
  // print the tutor card
  $jsonText .= "<a href='$url' class='card-container'>";
  $jsonText .= "<img src='$profileImage' alt='$tutorfame Profile Picture'>";
  $jsonText .= "<div class='info-wrapper'>";
  $jsonText .= "<div class='card-info'>";
  $jsonText .= "<p class='heading-4'>$tutorfame $tutorlame</p>";
  $jsonText .= "<p class='body-text tutor-spec'>$courseName | $minGrade - $maxGrade</p>";
  $jsonText .= "</div>";
  $jsonText .= "</div>";
  $jsonText .= "</a>";


}
$jsonText .= "<div class='max-flex-box-item'></div>";
$jsonText .= "<div class='max-flex-box-item'></div>";
$jsonText .= "<div class='max-flex-box-item'></div>";
$jsonText .= "<div class='max-flex-box-item'></div>";
$jsonText .= "<div class='max-flex-box-item'></div>";
$jsonText .= "<div class='max-flex-box-item'></div>";
$jsonText .= "<div class='max-flex-box-item'></div>";

if (!isset($myObj)) $myObj = new stdClass();

$myObj->newList = $jsonText;
$myJSON = json_encode($myObj);

echo $myJSON;
```

Figure 4: Code for AJAX Filter and Sort Feature

Member’s registration and login are operated by the account section on the top navigation bar. This section will either display a member’s status (profile picture and name) or link to login and sign up, which depends on if the user is logged in or not (Figure 5).

On the signup page, the user must input first name, last name, email, password, and select a radio button to either register as a student or a tutor. Once the sign-up page input fields are filled, it will call the ‘new-member.php’ file which completes the registration procedure by adding the new user into the database as a student or tutor. From the ‘new-member.php’ file if users input a duplicate email, an alert will let the user know that the email already exists and will send the user back to the signup page. The page stores the user’s information to the member (parent class) table and tutor (subclass) or student (subclass) table depending on what the user chooses. It will also create a new user account with a password allowing the member to have access to the database. Username and password for the privileges are the same as the user’s email and password.

Once the information is entered, the website will jump to a new page to check if the email is in the database and if the password matches the email. The website will notify the user if there is an issue with the email or password such as if they are incorrect or they do not exist. Otherwise, the user will successfully log in and receive a welcome alert. Once the user is logged in, a session will be used to store the user’s login information including, the user’s email, name, unique id, and if the user is a tutor. By default, the session will expire in 90 minutes. However, users can prevent it by checking the “Keep me signed in” checkbox on the login page. By checking “Keep me signed in” the session will not expire until the user logs out of the account themselves or 30 days later. The top navigation bar on every page will display the user’s profile image and their name after they log in. Upon hovering over their profile image, a dropdown menu will appear allowing users to access the account settings page and log out from the top navigation

As a student, users have all the same privileges as a visitor. In addition, students will be able to write reviews on tutors. From the account settings page, students will be able to make changes to their account email, password, etc. Moreover, students have access to the cart page. From the course/tutor details page, students are able to add the course/tutor to their cart. Upon adding a course to cart, the total price of the cart will be updated and an additional tax will be applied (Figure 6). Students are able to personalize their home page recommendations by inserting subjects of interest, their grade, preferred language, and location in the personalization tab under settings (Figure 7). After the student enters their preferences the recommended tutors would be updated and will display tutors who fall under the specified criteria (Figure 8).

As a tutor, users have all the same privileges as a visitor. Additionally, they are able to view their profile, edit their name, email, password, biography. Moreover, tutors are able to create new courses with the subject name, hourly rate, and grade range. Tutors will be able to make edits to the courses as well as delete them when needed. Furthermore, tutors are able to view their reviews under their account profile page as well. Currently, we would assume that time bookings would be done on the side through other messaging services. The tutor availability setting is under construction, but once completed tutors can set their time availability. From there, students would be able to book based on the time available and have the scheduled booking added to their cart (not implemented yet). Once a booking has

been made it would be displayed on the tutor and student's profile under upcoming bookings (currently, not implemented).

Figure 5: Top Navigation for logged in user

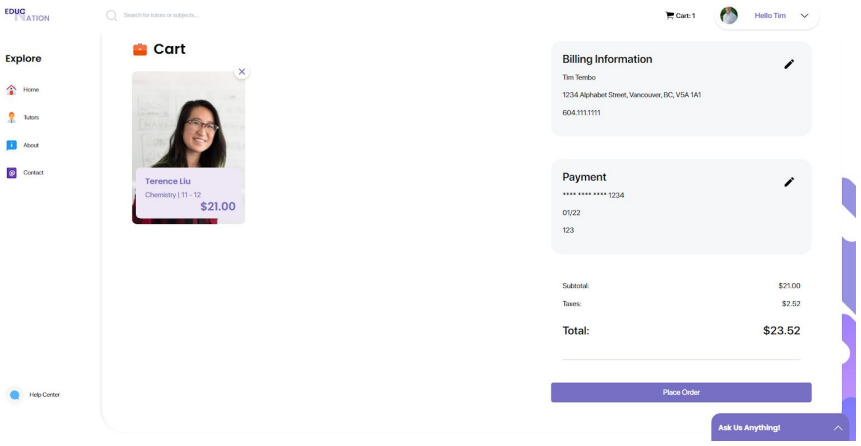


Figure 6: Cart Page

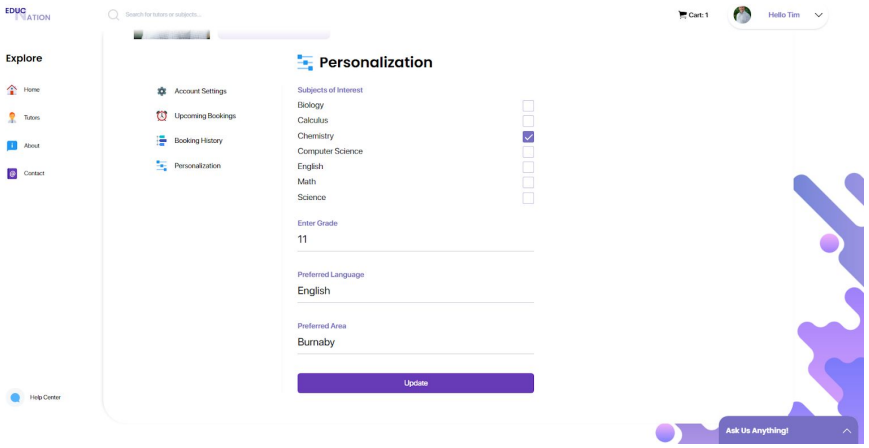


Figure 7: Personalization setting

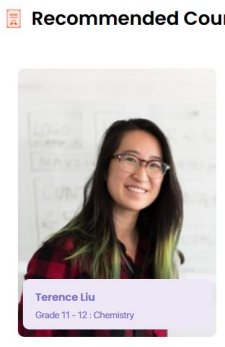


Figure 8: Recommended courses on the home page

AJAX

We used AJAX in our filtering and sorting features on the tutor/course listing page. Users are able to filter courses by subjects and grade range, as well as sort by the rating, price, or alphabetically. Whenever the user edits on the filtering or sorting condition, each will trigger a corresponding javascript function that stores all the filter and sort conditions. This criteria information will be saved as javascript variables, which will trigger a function called sortAndFilter(). sortAndFilter() sends the conditions/criteria to a PHP page called tutorListCards.php via GET. The PHP page will receive the data and write a query to request courses that follow the criteria from the MySQL database. The PHP page stores the response in a JSON format once everything is complete. A javascript function called theHTTPResponse() is called when sortAndFilter() sends a request to the tutorListCards page. When the response is ready, theHTTPResponse() will get the data from the tutorListCards page, and will finally update the webpage DOM. (Refer to Figure 4 for code)

Database Connectivity

To connect to the database from our PHP code, we use the following lines of code:

```php
<?php
// Connect to database using view privilege
$connection = mysqli_connect("localhost", "view", "", "terence_liu");
//Check if database connection was a success or not
if(mysqli_connect_errno()) {
  // if fail, skip all php and print errors
  die("Database connect failed: " .
    mysqli_connect_error() .
    " (" . mysqli_connect_errno(). ")"
  );
}
```

Figure 9: View Privilege Connection

```php
// get database using login privilege
$connection = mysqli_connect("localhost", "login", "", "terence_liu");
//Check if database connection was a success or not
if(mysqli_connect_errno()) {
  // if fail, skip all php and print errors
  die("Database connect failed: " .
    mysqli_connect_error() .
    " (" . mysqli_connect_errno(). ")"
  );
}
```

Figure 10: Login Privilege Connection

First, we create a variable that stores the active connection to the database, “terence_liu.” Then using an “if-statement” we check if the code has connected to the MySQL database. If the database connection fails, then we show an error code. When the connection to the

database is no longer needed and the query and result have been performed, we free the result variable, if it is not a boolean and close the connection.

```php
// release returned data
mysqli_free_result($result);
mysqli_close($connection);
```

Figure 11: Free the result and closing database connection

We then store the member identifier, as well as other variables via a key-value pair stored in the session for ease of access. With the member identifier saved in the session, we can perform queries and verify we retrieve the correct rows, by comparing all member identifiers with the key-value pair in the session.

```php
// starting session if it hasn't been started
if (session_status() == PHP_SESSION_NONE) {
    session_start();
}
$_SESSION["loggedin"] = true;
$_SESSION["email"] = $email;
$_SESSION["name"] = $row['fname'];
$_SESSION["isTutor"] = $isTutor;
$_SESSION["m_id"] = $row['m_id'];
$_SESSION["password"] = $password;
```

Figure 12: Storing session variables

Shared Files:

- reviewOverlay.php
    - Gets course subject and id to show in a dropdown for the written review overlay module
- account-settings-userOverlay.php
    - Gets tutor name and balance to show in the top header
- topNav.php
    - Gets member profile picture, name, and cart quantity

Single File:

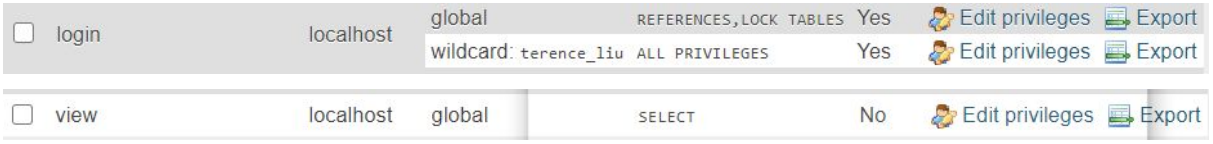
- tutor_about-me.php
    - Gets tutor about me content to populate text area.
- Update-bio.php

    - Updates tutor's about me page biography
- Update-account.php
    - Updates the tutor's account information (i.e. password, name, etc.)
- Tutor-listing.php
    - Gets all course options and populates the grid with tutor name and subject
- Tutor-detail.php
    - Gets course and reviews information
- Tutor_reviews.php
    - Gets reviews for tutor and populates the card with information
- Tutor_courses.php
    - Gets currently taught courses from the tutor
- Returning-member.php
    - Retrieves email and password to verify if the logging in user exists in the database
- New-member.php
    - Gets emails to check if the signing up user is using an already existing email address
- Insert-db-data.php
    - Inserts filler data into the database (ie. tutor, student, and course data)
- index.html
    - Gets basic course information to populate best courses cards
- Create-course.php
    - Gets courses table of the logged-in tutor to see if they've added a duplicate course
    - Adds course to database
- addReview.php
    - Inserts new review for tutor and checks if there is a duplicate review
- Account-settings.php
    - Gets user account information

Secure authentication handling

When a new member is created, its email and password are stored in the database, and the password is hashed with password_hashed($password, PASSWORD_DEFAULT).

Moreover, the database has 2 premade user accounts to access the database, one for users to log in, and one for visitors and members to browse the web page.



When a user is browsing the website, if the page only requires access to the database via SELECT, the database is accessed with mysqli_connect("localhost", "view", "", "terence_liu"); This way, we can ensure that the connection does not have access to modify the database. When a user is logged in or has registered a new account, the site will access the database with “login” which is done by mysqli(“localhost”, “login”, “”, “terence_liu”);. This account has more privileges for the database, which allows them to create a new user account for new users to access the database. At any time, if the member needs to modify the database, such as writing a review, changing their account setting, or creating a new course, we will connect the database with the account set up for that member, which allows for queries to SELECT, INSERT, and UPDATE to the database. To keep the user's email and password the same as their account access to the database, we will be dropping the previous user and creating a new user account to access the database with the updated email and/or password.

If visitors try to access a page that only members can see (i.e. the account settings) by editing the link, our site will check if the user is logged in first. If the user is not logged in as a member, they will be redirected to the login page.

2. Describe your learning experience, your challenges, and include a personal reflection.

Overall, this project had required quite a steep learning curve, because we were unable to get more practical experience on applying code concepts that were used in the project. For example, it would have been helpful to have an assignment that went over the AJAX topic in more depth, so that we could smoothly apply it into our project assignment 3. To contrast, the structure of the project assignment 2 was laid out in a way that made it easier to apply the database code into the project, because we were able to learn a lot from the database query assignment 1. We found that over the semester, the pacing of the course seemed to have spent too much time covering databases, and too little time with other topics like security.

The main challenge that we faced had to do with the AJAX topic, because there were not enough extensive examples provided to apply into our own projects. Additionally, the AJAX material we went over in the tutorial/lab focused more on text input searching, compared with our sort and filter function. So, it was more difficult to apply the code provided in the tutorial into our project. In order for everyone to understand all of the code in our project, we

would get together in a group after completing our individually assigned tasks and explain our code and logic to one another. The challenge with this is that we are only able to understand the general theory and logic behind the code, and do not get to directly apply the code. We initially coded as a group, but this was challenging because it was very inefficient and took much longer compared to dividing up tasks. Another challenge we faced was having too many features and functions in our web application. In the end, we were unable to complete all the proposed features (calendar/booking feature), as it was beyond the scope of the course and we were unable to find adequate tutorials on how to code this feature.

To sum up, as a group we were all very happy about the results of this project, and believe that we would not be able to have come this far working alone. We each were able to learn a lot from one another, because we all think differently. We all had more practice coding the project as a team and getting more comfortable with using github. Moreover, we got to practice focusing on the necessity of our website, and only working on secondary features once primary functions were completed. If we had a chance to redo this project, we would have first tried to mockup or layout our website earlier in the week, so that we would have a better understanding of the user flow. We made the mistake of adjusting the user flow as we coded, which caused us to have an over complicated user flow, so we had to waste time and remove some unnecessary features.