Project Report

	Name	ID
Student 1:	**Yasin Hamad**	**105307**

Project Title

Online Learning Platform.

Project Description

- Offer free online courses to help users improve their skills across various topics and become better at solving real-life problems.
- It will be a new free education environment on the internet. It is not the only one on the internet, but it will help make knowledge and science free for everyone.

Target Audience

People interested in personal development and skill improvement.

Core Features

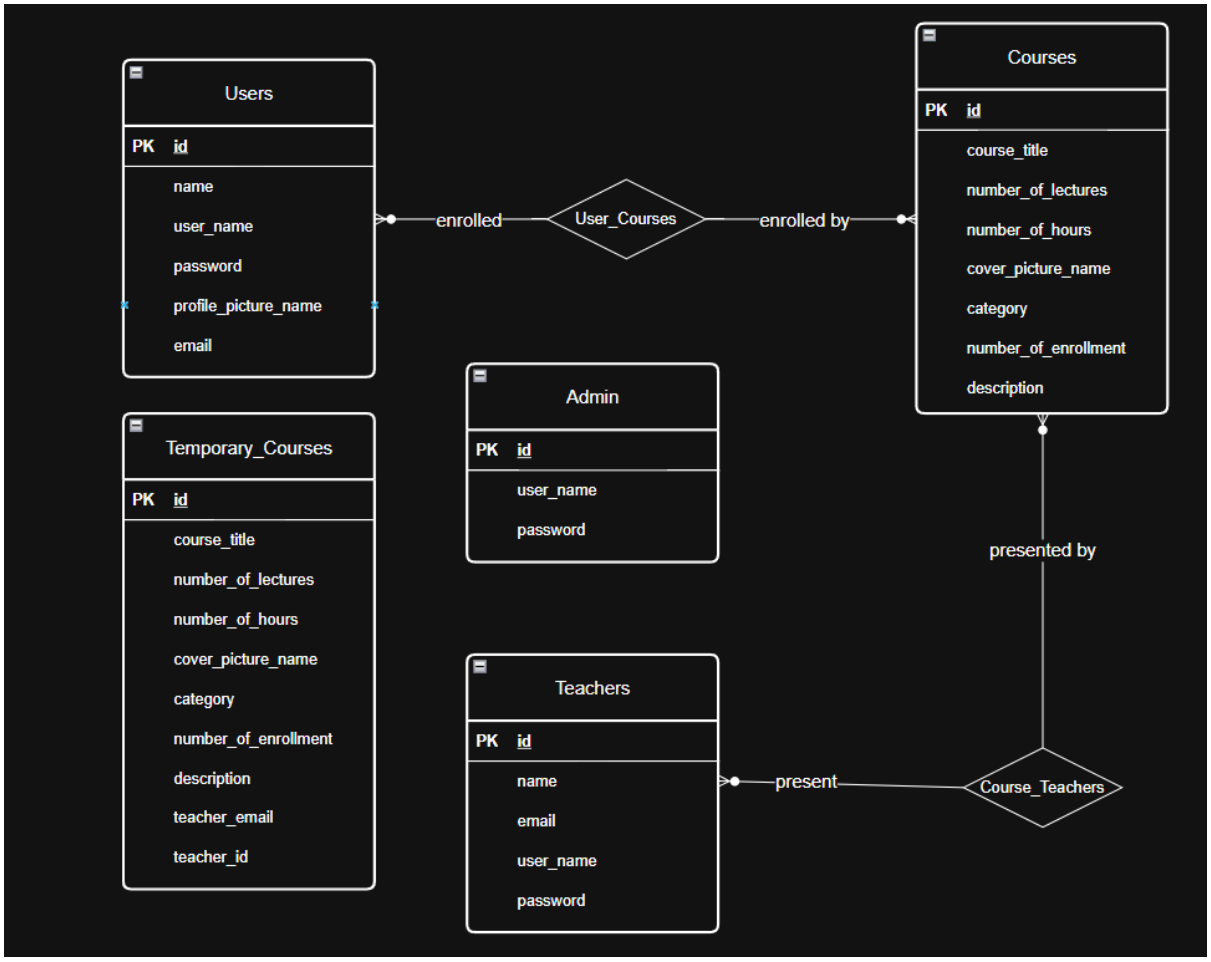
- Authentication.
- CRUD operations (users only).
- Search functionality.
- Role-based access control.

Database Design

- The key entities in the system are course, user, teacher, and admin.
- There are two relationships in the ER diagram:
    - Course Teachers (many-to-many): each teacher can have many courses, and each course can be created by multiple teachers (I did not implement a way to allow multiple teachers to add the same course).

- User Courses (many-to-many): a user can enroll many courses, and a course can be enrolled by multiple users.

- **ER diagram:**

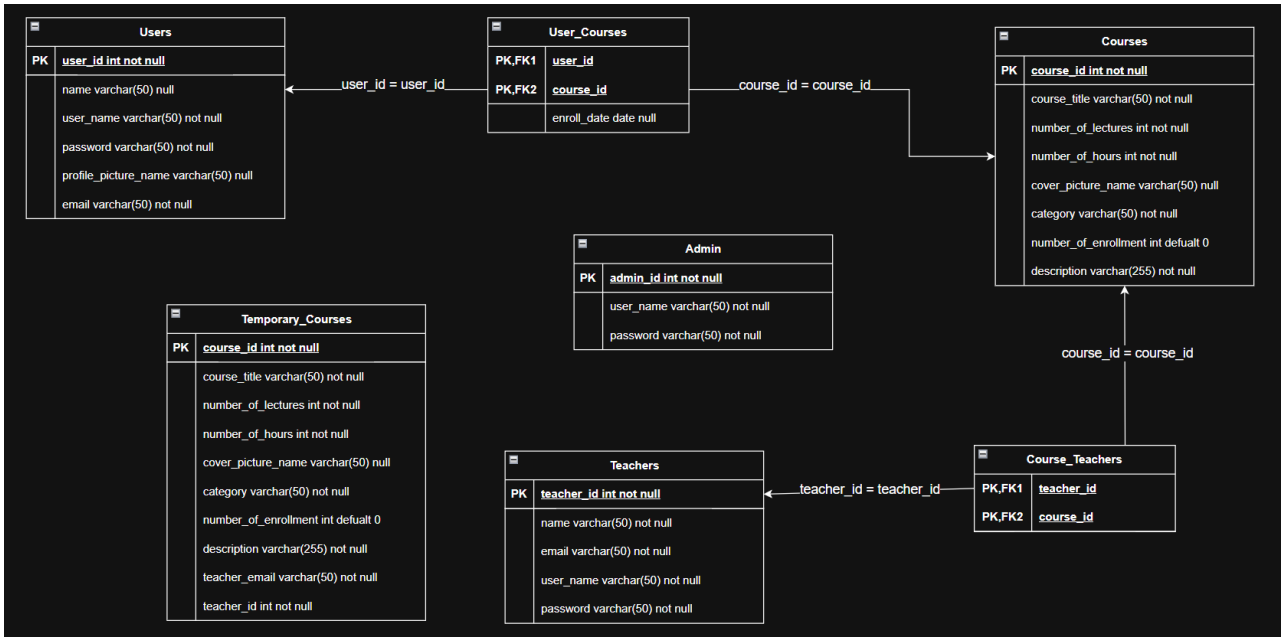


- There are five main tables in the database and two additional tables due to many-to-many relationships:
  - Courses with these attributes:
    1. Course id.
    2. Course title.
    3. Number of lectures.
    4. Number of hours.
    5. Cover picture name (I store the name here so I can know which picture in the folder belongs to this course).
    6. Category.
    7. Number of enrollments (I use this to display courses in descending order in the "most popular courses" section).
    8. Description (I use attributes 1,6 and 7 to find the right courses during the search process).
  - Temporary Courses (courses waiting for admin approval) with the same attributes as the Courses table, plus:
    1. Teacher email (To send the acceptance or rejection message to the teacher after the admin makes a decision. I did not implement this feature due to an issue in the mail engine I could not solve).

2. Teacher id (I use this to display the id to the admin and to send it to the `temporary course view` page so the admin can see if the teacher has courses in the database or is new).

- Admin with these attributes:
  1. Admin id.
  2. Username.
  3. Password.
     The only admin in the database is 'admin' with the password 'main'. The only way to add a new admin is to manually insert the username and password into the database.
- Teachers with these attributes:
  1. Teacher id.
  2. Name.
  3. Email.
  4. Username.
  5. Password.
- Users with these attributes:
  1. User id.
  2. Name.
  3. Username.
  4. Password.
  5. Profile picture name (same as the cover picture for courses, I store the name here so I can know which picture in the folder belongs to this user).
  6. Email.
- Course teachers with these attributes:
  1. Teacher id.
  2. Course id.
- User courses with these attributes:
  1. User id.
  2. course id.
  3. Enrollment date (I use this to display the new courses first on the `profile` page).

- **Physical diagram (tables):**



Frontend Overview

- I have used HTML, CSS, and JS for building the frontend, I did not use any framework.

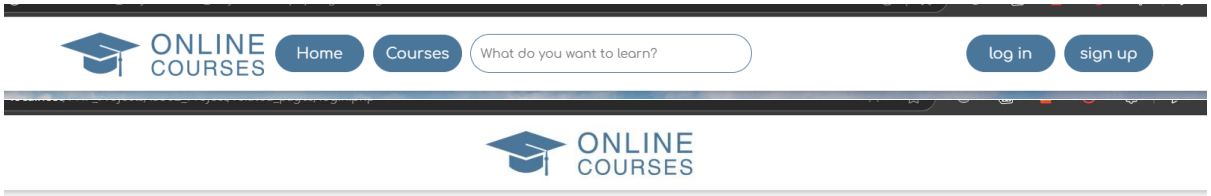
- The website is responsive, allowing users to learn from their phones. The container holding the website will primarily have four width variations based on screen sizes.

From	To	Screens	Container Width
0px	767px	phones	100%
768px	991px	tablets	750px
992px	1199px	tablets and desktops	970px
1200px	infinity	desktops	1170px

- The main pages are (home, courses, login, signup, and profile). All other pages follow the same structure and style of the main pages.

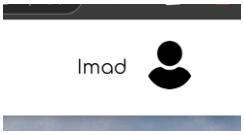
- In this section, I will include images only for dynamic elements (i.e. with PHP). I will not include images for static elements (only HTML and CSS)

- Every page in the website consists of three main parts (header, main, and footer).
    - header:
        - Contains all the links that users can use to navigate the main pages. (Only the logo appears on certain pages like login, signup, and admin pages)

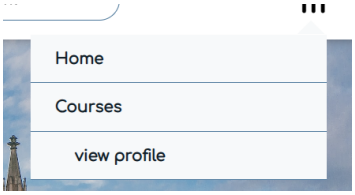


- The log in and signup buttons disappear from all pages when `$_SESSION['user_name']` is set. (i.e., when the user is logged in or signed up).
- The user's first name and profile image appear instead of the login and signup buttons when `$_SESSION['user_name']` is set. If the user has not uploaded a profile picture yet, a default image (`null.png`) appears.
  - With profile image:

  - Without profile image:



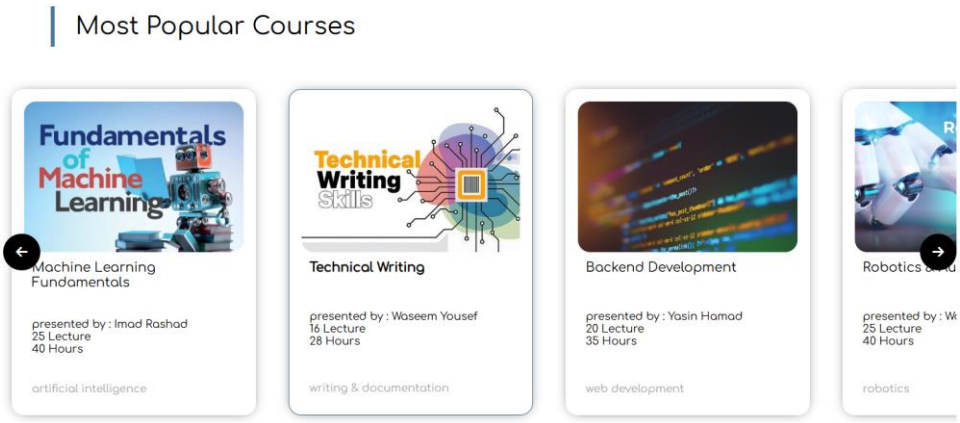
- On desktop screens, users can click their profile image to access their profile. On smaller screens, a new `<li>` element, `view profile`, appears if `$_SESSION['user_name']` is set.



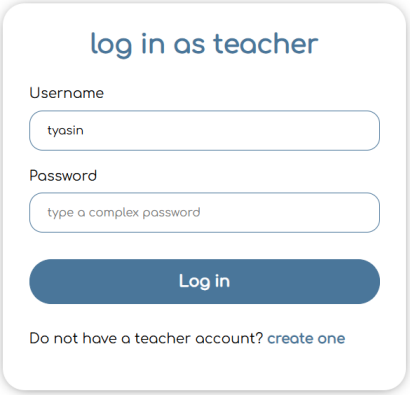
- A logout button appears in the profile header page, if the user clicks it, the system directs the user to the home page and unsets the session.

- main:
  - The content changes based on the page.
  - Home Page sections:
    - Most Popular Courses: Displays the top 8 courses with the highest number of enrollments.

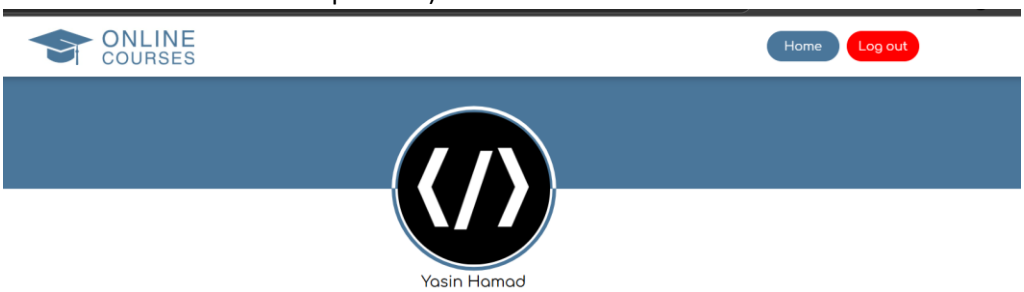
      All pages follow this main design.

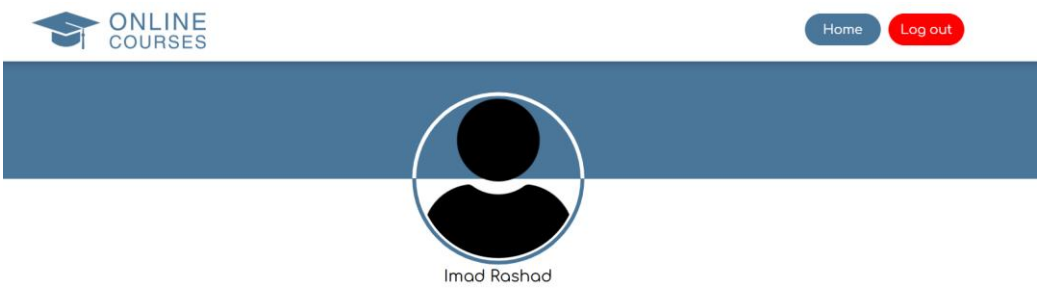


  - Teachers: Showcases some teachers and their experience.
  - User Feedback: Displays comments from users.
  - Positive Quote: Features an inspirational quote.
- Courses Page sections:
  - Course Filter: Allows users to filter courses.
  - Available Courses: Displays all courses on the platform.
- Login, Signup, Add Course, Check Password, and Edit Account pages only contain forms.
  All forms follow the same style.

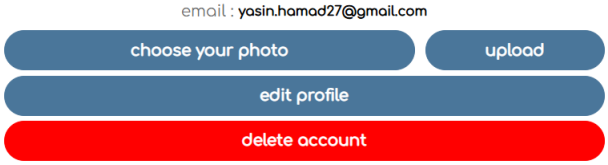


- Profile Page sections:
  - Profile: Displays the user's photo (default photo appears if no photo is uploaded).

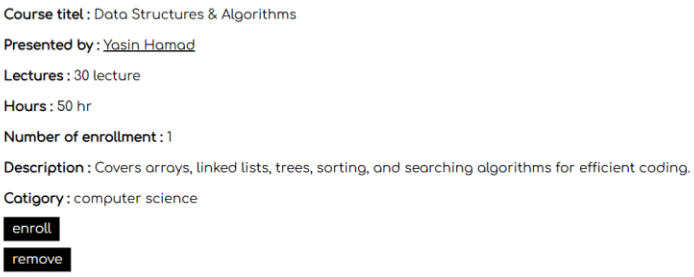




- User Courses: Lists courses the user is enrolled in.
- Account Management: Displays the user's email and includes forms for uploading a photo, editing the account, and deleting the account (covered in the Form Handling & Validation section in this report).



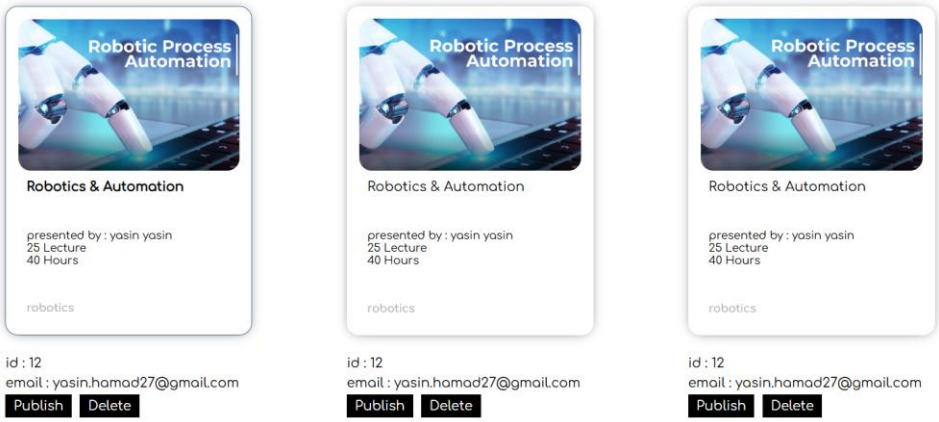
- View Course Page: (only one section):
  - Displays course details.
  - If `$_SESSION['user_name']` is set, an **Enroll** button appears.
  - When a user enrolls in a course:
    - The enrollment number increases by 1.
    - `course_id, user_id` is inserted into the database, allowing the course to appear in the user's profile.
    - If the user is already enrolled, the enrollment number does not increase again.
  - After enrolling, a Remove button appears.
  - Clicking the Remove button deletes `course_id, user_id` from the database, removing the course from the user's profile.



- Temporary View Course Page: similar to the View Course Page but without Enroll and Remove buttons (for admin use).
- Teacher View Page sections:
  - Teacher information: Displays the teacher's name and email.
  - Teacher Courses: Lists courses created by the teacher.

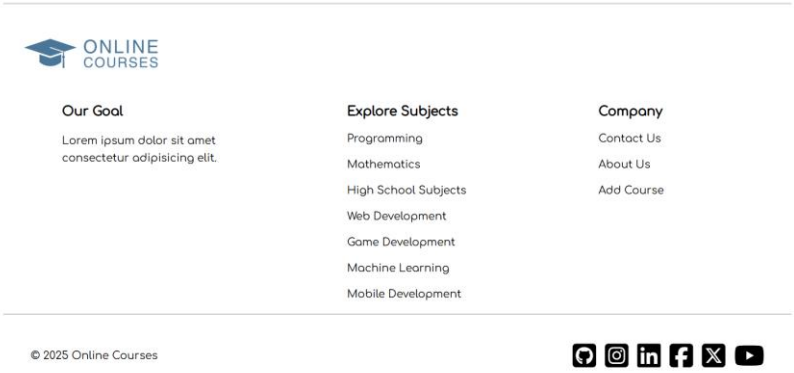
- Accept/Reject Page: (only one section)
  - Displays temporary courses with two buttons: Publish and Reject.
  - If the admin clicks Publish, the system deletes the course data from the Temporary Courses table and inserts it into the Courses and Teacher Courses tables, making it visible to users.

- If the admin clicks Reject, the system deletes the course data from the Temporary Courses table.

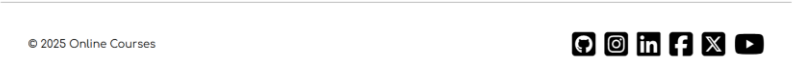


- footer:
  - Consists of two main sections:
    - Section 1: Contains links to main website topics, including other links like About Us and Contact Us.
    - Section 2: Displays social media accounts.
  - The first section is hidden on certain pages like Login and Signup but visible on main pages like Home and Courses.

  Big footer:



Mini footer:



- When the user enters the website, they will be on the `home` page, where they can:
  - Search by typing in the search input in the header.
  - See the most popular courses.
  - Click on a course to see more details, then click on the teacher's name to see the courses they have made. They can also enroll in the course.
  - Go to the `courses` page to see all the courses on the website.
  - Login or signup. If they do, they can go to the `profile` page to see the courses they enrolled in, remove a course, edit their profile, logout or possibly delete their account.
  - Add a course (option in the footer). If they do, the website will ask for the teacher's username and password. If they do not have an account, they can sign up as a teacher and then submit the course to the admin.

Backend Overview

- The application follows a functional programming approach. There will be a function for each task, and some functions will depend on others. For example:
  `print_user_courses($user_id)` will call another function `does_user_have_courses($user_id)`. If it returns true, the first function will work with other functions like this:

  ```
  get_all_info_about_these_courses(convert_table_of_ids_to_normal_arra
  y(get_user_courses($user_name)));
  ```

  - The first function returns the ids table.
  - The second function converts it from an associative array to an indexed array.
  - The third function returns an associative array containing all information about the courses.
  - The main function then continues and prints the courses.

- Any feature that depends on user data will check `$_SESSION('user_name')` to see if it exists or not and then takes the appropriate operation. For example:
  - The user's first name appears if `$_SESSION('user_name')` is set. The same applies to the Enroll and Remove buttons on the Course View page.

- The backend will handle user interactions and requests using `$_POST[]` or `$_GET[]`, processing the data and calling the relevant functions. If there is a message for the user, it will appear in the frontend after the function places the message inside a variable passed by reference.

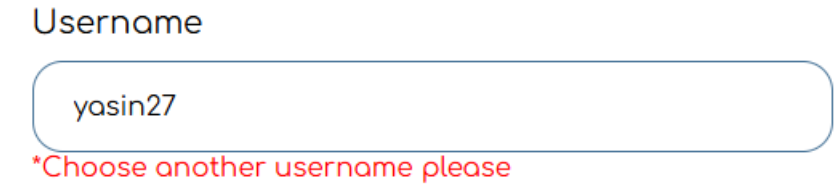
Form Handling & Validation

- All the forms in the system work in the same way. If the user clicks submit, it will check each input by calling a function and passing the variables (input data, message) by reference. This allows the function to trim the data (to prepare it for another use) and put the appropriate message in the message variable if it returns false. It returns true if everything is correct. The functions are nested like this: (the signup form for example)

```php
if(check_full_name($full_name, $correct_full_name)){
    if(check_user_name($user_name, $correct_user_name)){
        if(check_email($email, $correct_email)){
            if(check_password($password, $correct_password)){
                $_SESSION['user_name'] = $user_name;
                $_SESSION['name'] = get_first_name($full_name);
                $_SESSION['full_name'] = $full_name;
                $_SESSION['email'] = $email;
                add_user_to_database($full_name, $user_name, $password, $email);
                header("location: ../index.php");
            }
        }
    }
}
```

If a function returns false, a message will appear under the input, like this:

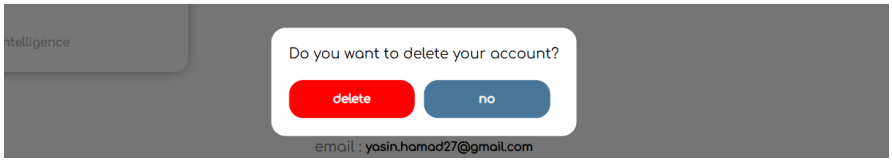
```php
<div class="input-container">
    <label for="user_name">Username</label>
    <input type="text" id="user_name" placeholder="username" required name="user_name"
    value="<?php if(isset($_POST['submit'])){echo $user_name;}?>">
    <?php
        if(isset($_POST['submit'])){
            echo "<div class ='correct-input'>$correct_user_name</div>";
        }
    ?>
</div>
```



If the function returns true, the variable will be empty, so nothing will appear.

- There are nine main forms in the system:
    1. Search form: it has one input where users can type specific words related to the course they want to study. The form sends the data to the `courses` page, which processes the search and displays the appropriate courses.
       Search process explanation: after the search form sends the data to the `courses` page, the functions will:
        1. Convert the user-entered sentence into an array, with each word in a separate index.
        2. Make the associative array `course_id => course_words`, where `course_words` is an indexed array containing all words in (course title, course category, course description) each word in an index.
        3. Loop through the associative array, compare each course word with the user-entered words, count the number of matches, and put them in an associative array `course_id => number_of_matches`.

    4. Sort the associative array so the courses with the most matches appear first when printing the courses.
2. Signup form: it has four inputs:
    1. Full name. (must be [2-4] words)
    2. Username (the user should choose a username that does not exist in the database).
    3. Email.
    4. Password. (must be 5 chars or more)
3. Login form: it has two inputs:
    1. Username.
    2. Password.
4. Upload profile photo form: contains a file input. (must be square i.e. width = height)
5. Edit account form: similar to the signup form. For the username field, the system checks if the user has changed their username. If they haven't, it does not check if the username exists in the database or not. Before accessing this form, the user must verify their current password. If the password is incorrect, they can't access the `edit account` page.
6. Delete account form: this form appears when the user clicks the Delete Account button in their profile. This is for checking that the user does want to delete their account.



7. Signup and Login as teacher forms: the login form has two inputs: username and password. The signup form has the following inputs:
    1. Full name.
    2. Username.
    3. Email.
    4. Password.
8. Add course form: if the user completes the teacher signup/login process, they can access this form. It has the following inputs:
    1. Course full name.
    2. Number of lectures.
    3. Number of hours.
    4. Category.
    5. Description (must be less than or equal to 255 chars).
    6. File (course cover image).
9. Admin login form: this form appears when someone accesses the `accept_reject_courses` page, where the admin can approve or reject courses.

Roles and Permissions

There are four roles in the system: admin, user, guest and teacher.

- Guest can do the following:
    - See the courses and courses content.

    - See teacher's courses.
    - Search for specific topics.
    - Login and signup (to become a user).
    - Login and signup (to become a teacher).

- User can do the following:
    - Everything the guest can do, except for (login and signup to become a user), since they already did one of them.
    - Edit their account (full name, username, email, password).
    - Enroll courses.
    - Remove courses from their list.
    - Add a profile picture.
    - Logout.
    - Delete their account.

- Admin can do the following:
    - View the temporary courses.
    - See course content.
    - Check if the teacher has other courses on the website.
    - Accept or reject a temporary course.

- Teacher can do the following:
    - Add a new course.

How will each roll access the website?

- A guest is anyone who visits the website.
- A user is a guest who has logged in or signed up.
- The admin needs to access the `admin_log_in` page.
- A Teacher is a guest or user who has logged in or signed up as a teacher. (link is in the footer `add course`)

Web API (if applicable)

- GET / any data that pages need to display everything correctly. For example:
    - When the user clicks on a course box, the system sends the course id and teacher id to the Course View page through an `<a>` HTML element. The page then can get the course information, displays it, and includes the teacher id in another `<a>` HTML element (teacher's name). If the user clicks on the teacher's name, the system redirects them to the Teacher View page, which gets and displays the teacher's information and courses.
    - If the user clicks the Logout button on their profile, the system sends a variable `logout=logout` and redirects the user to the home page. The Home page then checks whether `$_GET['logout']` exists and takes the appropriate action.

- POST / handles user's data. For example:

  - When the teacher fills in the form on Add Course page, the system sends the data to the same page, where it validates the data before adding it to the Temporary Course table, where the admin can review and approve the course.
  - When the user fills in the Login form, the system sends the data to the same page and checks whether the username and password exist in the database. If they do, the system gets the user's data from the database using functions, stores it in `$_SESSION`, and redirects the user to the Home page.

- If the user wants to login, they need to provide their username and password.

Challenges

- I built the project folder structure incorrectly. I put some pages in a folder called "related pages", which forced me to rewrite many functions (for example, I wrote the `print_these_courses` function four times for different pages, changing only a few things). I could put all PHP files in the same folder and wrote functions work for all files.

- Implementing the accept and reject feature for the admin role. That required creating a new table in the database to store teacher-submitted data, which would then be moved to the courses table after admin approval.

- Handling user profile pictures.

- There was an issue rendering the CSS and JS files, and that took some time to fix. I found on the internet that the following script will fix the issue.
  (it forces the server to reload the CSS and JS files on each refresh):

```
<link rel="stylesheet" href="css/master.css?v=<?php echo time(); ?>">
<script src="js/fuctions.js?v=<?php echo time(); ?>"></script>
```

  Before putting this script, the browser couldn't see the additional classes or functions I added to the main CSS or JS files.

- I encountered some CSS issues where certain elements took unwanted styles, after some research I learned the (initial value) in CSS and the (element *) selector which selects all the elements in the parent. Using (element * {all: initial}) all the elements inside the parent will back to their default styles.

- Implementing the search feature.

Timeline

- (1->2 weeks) developing the frontend for the main pages (home, courses, login, signup) pages.
- (one day) designing the ER diagram, physical diagram and implementing the database. (I edit these multiple times later)

- (1->2 weeks) developing the backend and completing the frontend for the remaining website pages.