ZWA semester assignment

"iCourses" website

Document version: 1.0.0

Deadline: 07/01/2023

Name of Project: iCourses

Link to the project page:
http://wa.toad.cz/~rastvdmy/semestralka/iCourses

Solver: Dmytro Rastvorov

Exercise date: WED – 4:15 p.m.

Practitioner Name: Zdenek Vlach

Content

- Job description
- Entry
- User Manual
- Description of website functionality
- Site UI samples
- Description of the implementation
- Description of the main features of the program (functionality of the main scripts)

Job description

Assignment

(Formal submission as if coming from an external customer. The submission will include conditions for acceptance)

Description

The iCourses website is for people who want to learn programming. On the site you will find a variety of popular courses that have been selected based on popularity criteria, as well as the materials that teachers use for their courses.

Website pages

The website will contain 7 pages:

1. Authorization page
2. Registration page
3. Main page
4. Course page
5. Video page
6. Page "About us"
7. Contact page.

Each page will have a footer with links to Instagram pages as well as to the official website of the faculty's specialization.

The text should also be in the footer section dedicated to registration and authorization. After authorization, several buttons should appear at the bottom

of the page: HP, Course, About Us, Contact Information, Log Out. Each button should be a different color. The Logout button should always have the same color.

1. Authorization page

The page will contain a form where the user will have to enter their information to login and enter the home page of the website. The page will have its own design and 2 buttons (Login and Register).

2. Registration page

If the user does not have a profile, he must register. A form will appear on the page where they must enter their email, username, password, and date of birth. All these data will be important, and the user must fill all the fields in the profile creation form. There will also be two buttons (Create and Cancel).

3. Main page

On this page you will find a description of the courses we offer. The page contains text and images for them.

4. Course page

Various courses will be available on the site. Each course will have its own icon and will be signed by the teacher teaching the user. There can be a maximum of 9 courses per page. Each course must have a link.

5. Video page

The video page should include the video itself, a video guide, and a Contact Us button that will direct the user to the contact page.

6. "About Us" page

This page should describe the purpose for which the website was created and what the creation of the website brought to the student.

7. Contact Information Page

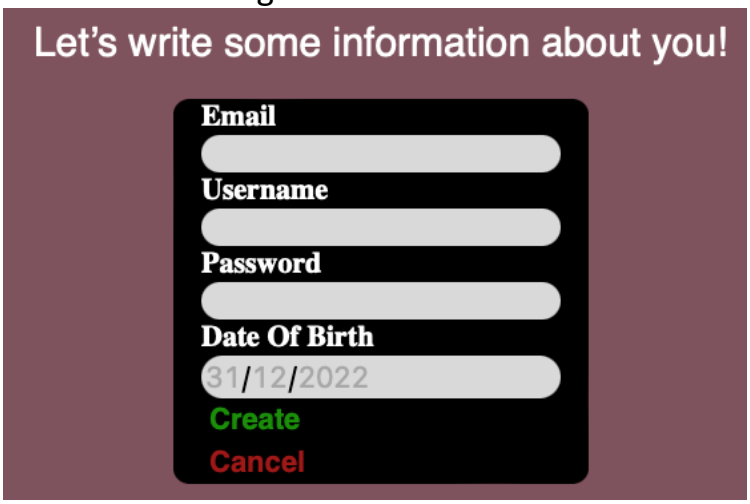
This page should include the website creator's contact information and their photo.

User Manual

Description of website functionality

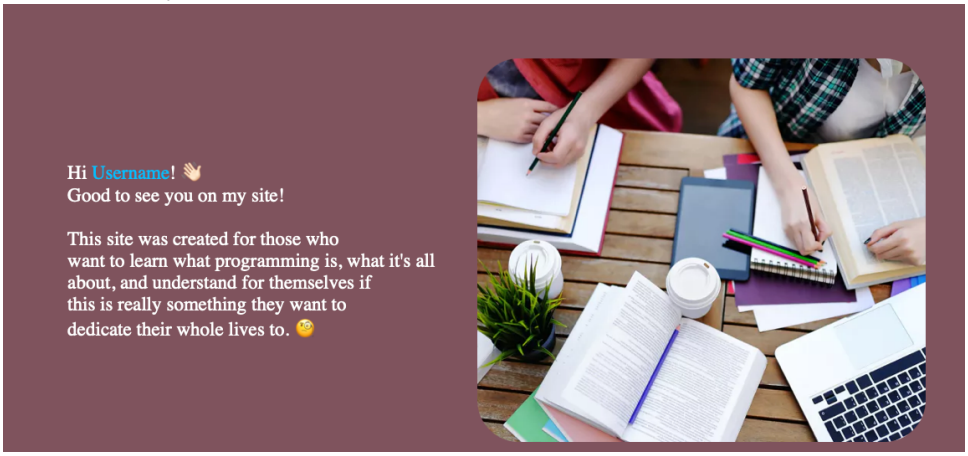
You must register to access the website itself. In the "registration" section, you need to enter data that will then become your identity: address, username, password, date of birth. After entering the data, the user can use it to access the website.

Fields for user registration:



Fields for user authorization:

After opening the website, we get to the home page (HP) with a description of the website, we see our username as a confirmation that the user has been authorized, and what it offers to the user.



In the Courses section, you will be taken to the courses page. Each page contains 9 different programming courses, and each lesson contains the name of the

instructor who will be teaching you.

Selecting a course will take you to the video page. If the user has any questions, it is always possible to click on the (Contact us) button, which will open a page with the contact details of the website creator.

On the page (About us) you will find a description of what the purpose of the site was and what I managed to find out.

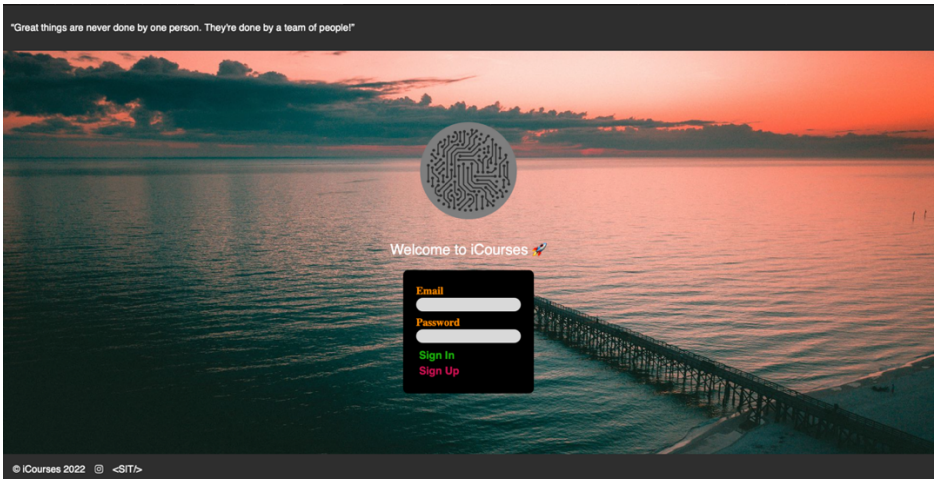
Contact information contains information about the creator of the site, where you can write / call and ask the user questions.

Click (Logout) to log out of the site, suspending the session between users and returning to the login page.

Web UI samples

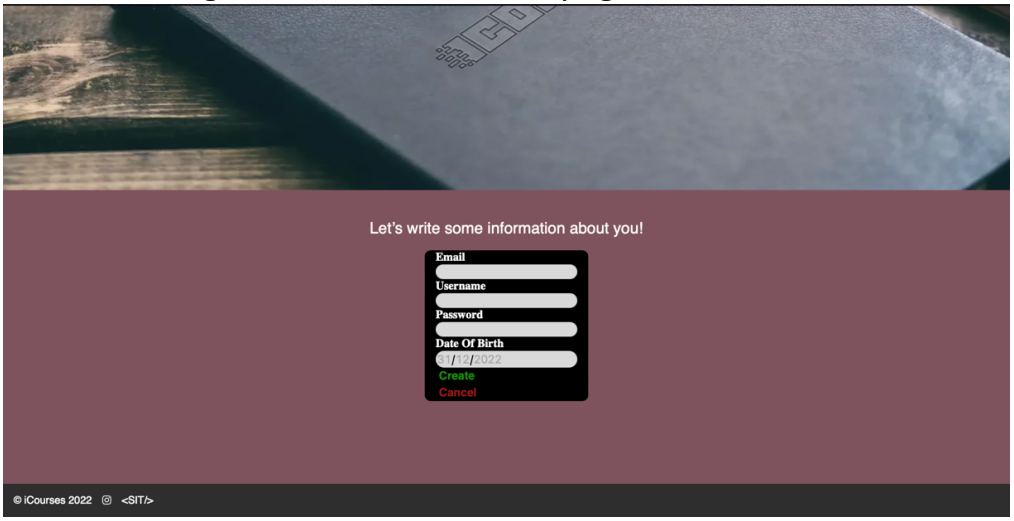
The user interface design was created using Figma. After clicking on this link.
It will be possible to see the prototype of the website with which the page was created.

On the login page, the user has the option of either logging in after logging in and clicking the Login button or creating a brand-new account by clicking the Register button.



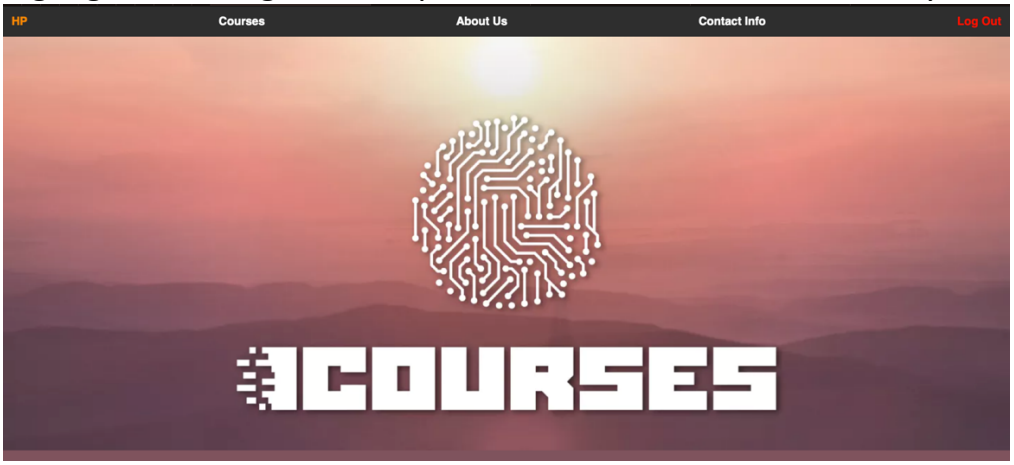
At the bottom of the page (footer) there are links to SIT's Instagram page, as well as a link to the industry's official website.

Clicking Register will take the user to the registration page where they must enter their details and click Create to create a profile or cancel to cancel account creation and go to the authorization page.

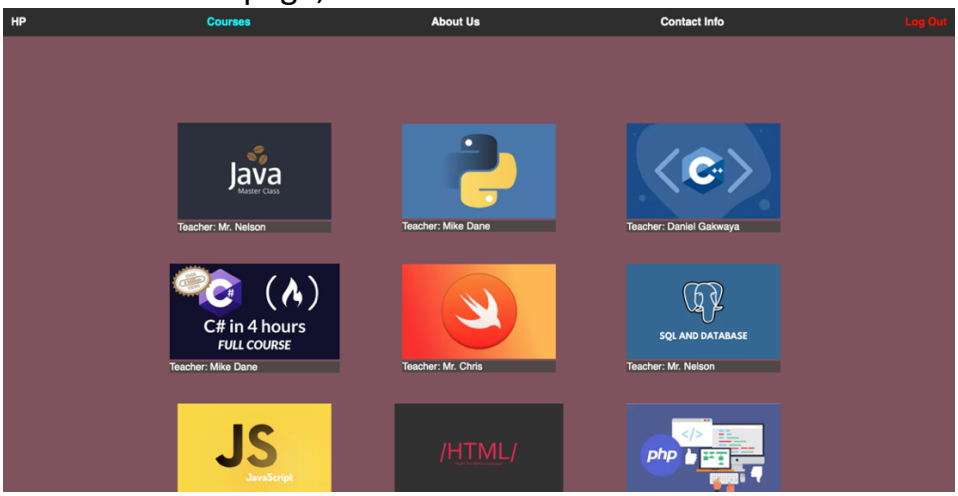


On the main page, after logging in, we have 5 buttons available. Clicking on the button will open the content page and the button will be color coded to show the user which page they are on and highlight it. The Log Out button takes the user to the login page, suspending their activity on the site, which is why the button is

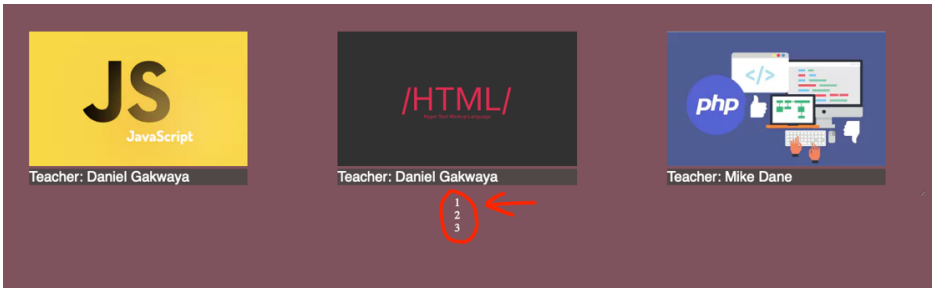
highlighted in bright red to prevent the user from accidentally clicking it.



On the Courses page, the user has access to various courses to choose from.

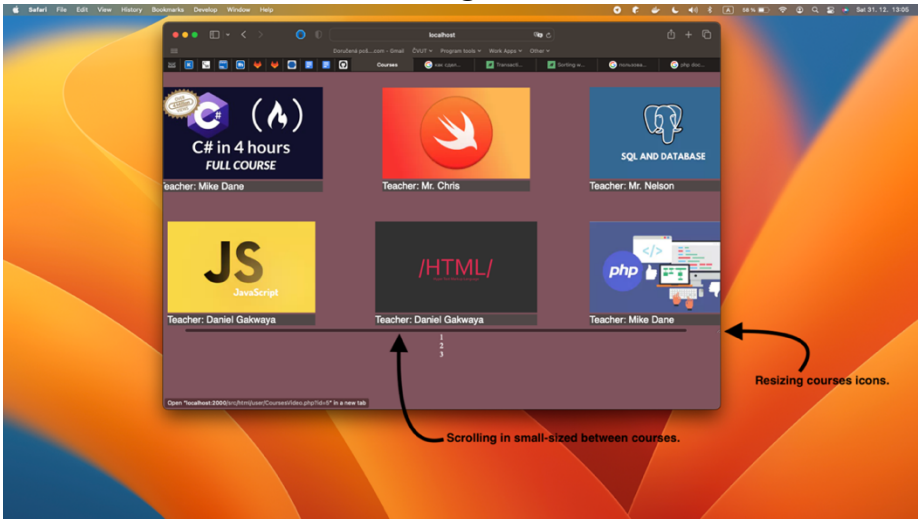


Paging is also available. It allows the user to go to the next page and select the course of interest.

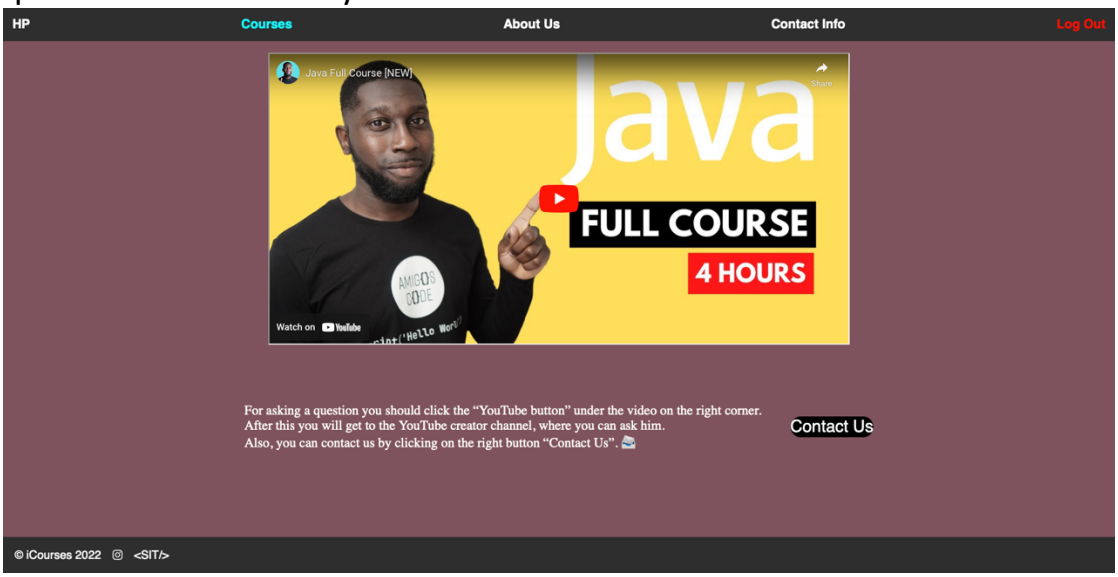


This page also allows the user to move the positions of the course icons, either by zooming in or out. If the user minimizes the browser window, he can move

between the courses. The image shows how it looks from the user's perspective:



After opening the course, the user has the option to turn on and watch the video. In case of any questions, the user can click on the Contact Us button, which contains the contact details of the site administrator, who will try to help with any questions the user may have.

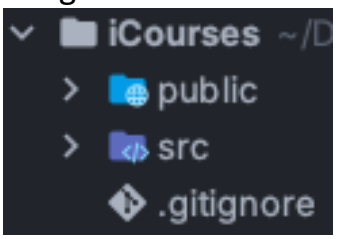


Implementation description

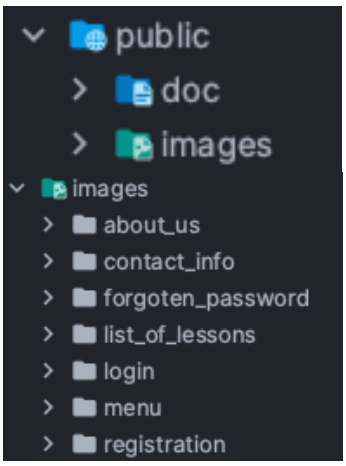
Description of the main features of the program (functionality of the main scripts)

Architecture

The project has been divided into different folders, each folder containing a file corresponding to its name. This way it will be much easier to find the necessary materials and we will also stick to the correct architectural system. Our project is divided into two folders: src, which contains the code, and public, which contains images and documentation.

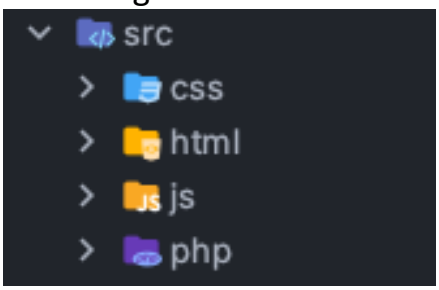


The public/images folder contains the gif images we use on our website. Each folder has its own name corresponding to the page that uses these images. The public/doc folder contains the project documentation in pdf, docx format, as well as the generated version.

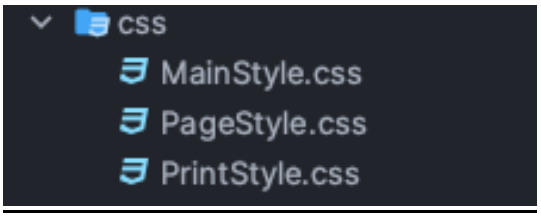


The folder "src" = "source" contains the code of our website. Since we used different languages for the website, the files were divided into different folders

according to their names.



CSS



MainStyle.css

The file contains cascading styles that were written exclusively for existing HTML types. Media styles, ids and classes were also created for these styles that were used in different contexts.

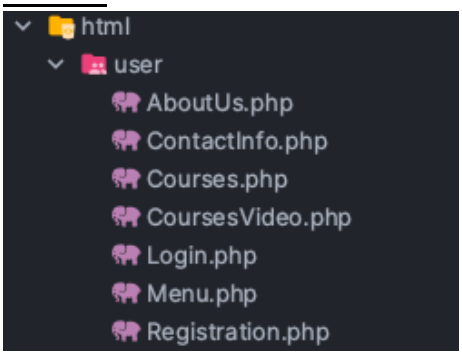
PageStyle.css

The file contains various created classes and ids with cascading styles that are used in HTML pages.

PrintStyle.css

The file contains the media style for printing the page. This style contains various cascading types, classes, and ids.

HTML



AboutUs.php

The page contains text and an image of the site's logo.

ContactInfo.php

The page contains the administrator's contact information, a picture, and a photo of the site administrator.

Courses.php

The page will show the user a list of courses that are available on the site. It also has a logic according to which we do pagination, so if the list of courses is large, the page has a maximum of 9 and the others are on the next page, which we get to by selecting from the available list of pages under the course icons. The file also interacts with the _dbController.php file.

CoursesVideo.php

The page will display the course video, text, and a button to go to the administrator's contact page. The page contains its own logic. The logic was implemented by looping through the database and checking if the selected course ID matches the course ID in the database. The file also interacts with the _dbController.php file.

Login.php

This page is the home page when you go to the website. The file contains a form that we enter to authorize the profile and open the website. The file works with the logValidation.php and LoginScript.js files to let the user understand whether the entered data is correct or not.

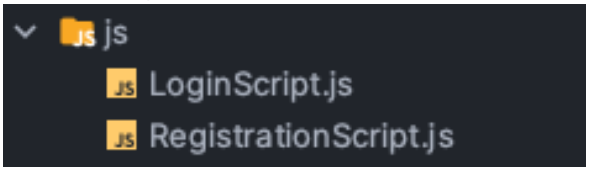
Menu.php

This page is displayed after authorization and is the main, initial page. Once we get to that page, the session starts, and the user is considered authorized. The page contains introductory text as well as the user's name, which is highlighted in a unique color, and images. The file interacts with the _dbController.php file.

Registration.php

The page contains a form for the new user to enter and the completed form is submitted to the database. The file interacts with _dbController.php and RegistrationScript.js files. The page will also display errors to the user and indicate which errors the user made. Errors are highlighted in a bright color so the user can easily see and correct them.

JavaScript



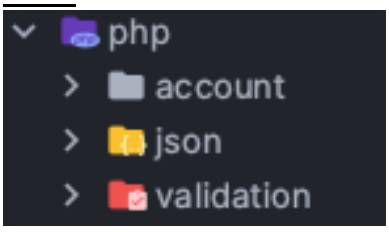
LoginScript.js

The file is used to implement an error output in case of incorrect e-mail or password entry. The checking process is done using "parsing". If everything is fine, the software will print a confirmation and transfer us to the website page.

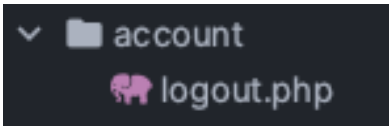
RegistrationScript.js

The file is used to implement an error output in case of incorrect data entry during registration. The process of checking the entered data is carried out using "parsing". If the user has entered the data correctly, the software will print a confirmation of the profile creation and transfer the user to the login page.

PHP



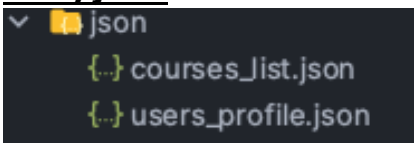
PHP/account



logout.php

This file is used when the user leaves the website page. This pauses the session and takes the user to the login page.

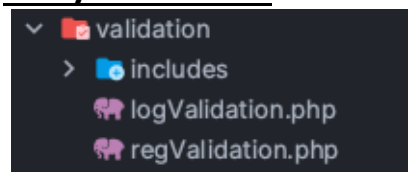
PHP/json



courses_list.json & users_profile.json

These files are an alternative to the database where we store information about users (courses_list.json) and courses (users_profile.json).

PHP/validation



logValidation.php

The file contains the parsing logic, where we step through the individual fields of the form and check them. If the information was entered incorrectly, the software will display an error message to the user describing the problem encountered. If the user entered everything correctly, he is redirected to the main page of the website.

regValidation.php

The file contains the "parsing" logic, where we check the fields in the form one by one. If the data is incorrect, the user will receive an error message informing them what was entered incorrectly and what needs to be corrected for the registration to be successful. If successful, the user will be redirected to the login page.

PHP/validation/includes

_createCR_L.php

The file contains logic that if we remove our file (courses_list.json) from the folder, when we add new courses, it will automatically be created and the course information we entered will be added to the file.

_createDB.php

The file contains logic that if we delete our file (users_profile.json) from the folder, when we add new users, it will automatically be created and add the user information we specified to the file.

_dbController.php

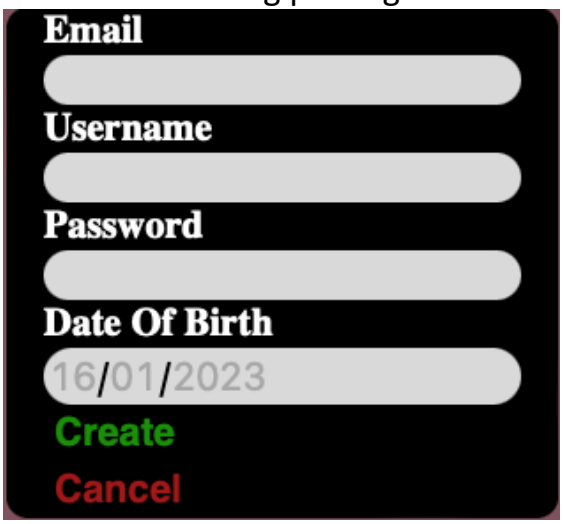
The file contains the CRUD logic by which we can add courses or videos to us .json file and read it. The file is closely related to the _createDB.php and _createCR_L.php files.

Handling forms

The application has 2 types of forms: for login and registration. These pages are maintained on the server side using php.

The login form is maintained using the regValidation.php file where we validate using email and password parsing.

In the case of registrations, the logValidation.php file takes care of this service, where we also check the e-mail, the availability of the nickname, password and date of birth using parsing. All fields are important and must be filled out.



Security

For user-side security, the htmlspecialchars() function is used for login (Login.php) and registration (Registration.php). Used to convert special characters to HTML entities.

User passwords are stored in the password_hash() hash format in the addUser function to prevent intruders from gaining access to the account. At login, the user's password is checked using the password_verify() function, and if the passwords match, the user is allowed to login under their name.

To prevent cybercriminals from accessing user information, we have ensured that information is sent to the database using $_POST during registration and login. In this way, attackers do not get information about the user.

Data processing algorithms

The following classes were created to work with the database:

_create_L.php and _createDB.php

It will automatically create a .json file if there is no .json file(s) in the json folder.

_create_L.php

```php
<?php
$userData = array();

$encodedData = json_encode($userData,  flags: JSON_PRETTY_PRINT);

file_put_contents( filename: "../../php/validation/json/courses_list.json", $encodedData);
```

_createDB.php

```php
<?php
$userData = array();

$encodedData = json_encode($userData,  flags: JSON_PRETTY_PRINT);   Unknown, Yesterday

file_put_contents( filename: "../../php/json/users_profile.json", $encodedData);
```

_dbController.php

It serves to connect our database and the entire program.

addUser() - used to add a user to the database.

```php
function addUser($data, $email, $username, $password, $date) {
    $user = array(
        'id' => uniqid(),
        'email' => $email,
        'username' => $username,
        'password' => password_hash($password, algo: PASSWORD_DEFAULT),
        'date' => $date
    );

    $data[] = $user;
    file_put_contents( filename: DB_FILE, json_encode($data, flags: JSON_PRETTY_PRINT));
}
```

addCourse() - used to add a user to the database.

```php
function addCourse($data, $id, $src, $link, $figcaption) {
    $courses = array(
        'id' => $id,
        'src' => $src,
        'link' => $link,
        'figcaption' => $figcaption
    );

    $data[] = $courses;
    file_put_contents( filename: CR_FILE, json_encode($data, flags: JSON_PRETTY_PRINT));
}
```

getUserByEmail() - used to search for a user by email.

```php
function getUserByEmail($email) {
    $userInfo = json_decode(file_get_contents( filename: DB_FILE), associative: JSON_OBJECT_AS_ARRAY);

    foreach ($userInfo as $userEmail) {   Unknown, Yesterday • Added iCourses project
        if ($email == $userEmail['email']) {
            return $userEmail;
        }
    }
    return null;
}
```

getUserByUid() - used to search for a user by id.

```php
function getUserByUid($uid) {
    $userInfo = json_decode(file_get_contents( filename: DB_FILE), associative: JSON_OBJECT_AS_ARRAY);

    foreach ($userInfo as $userId) {
        if ($uid == $userId['id']) {
            return $userId;
        }
    }
    return null;
}
```

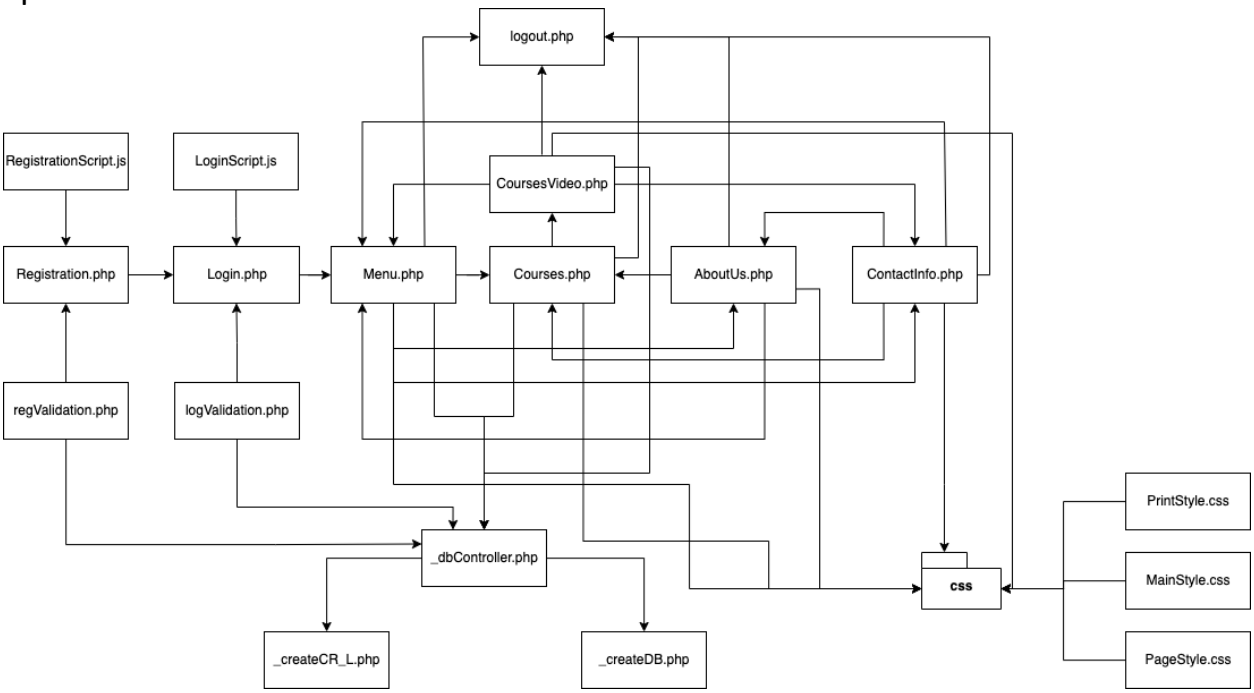
getCourseByid() - used to search for a course by id.

```php
function getCourseById($id) {
    $courseInfo = json_decode(file_get_contents( filename: CR_FILE),  associative: JSON_OBJECT_AS_ARRAY);

    foreach ($courseInfo as $courseId) {
        if ($id == $courseId['id']) {
            return $courseId;
        }
    }
    return null;
}
```

UML (made of help draw.io)

Specifies the basic connection between classes



Description of the use of frameworks

No frameworks were used in the project. The project is created on pure php, js, html, css.