Комп'ютерна Академія IT STEP
Професійна комп'ютерна освіта
Кафедра розробки програмного забезпечення

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО КУРСОВОГО ПРОЕКТУ

на тему: "Тестовий веб-портал для проведення веб-тестування**"**

Розробила студентка гр. ПВ-123 Кізілпінар С.М.
Керівник КП Рубан С.А.

Кривий Ріг
2023

ЗМІСТ

Вступ

Тестування є одним із найважливіших етапів у процесі оцінки знань, вмінь і навичок студентів, співробітників чи абітурієнтів. Воно дозволяє перевірити рівень засвоєння матеріалу, виявити сильні та слабкі сторони і допомагає визначити дальші кроки в навчанні або кар'єрному рості. З розвитком технологій і поширенням інтернету, веб-тестування стало популярним і ефективним засобом здійснення такого контролю.

Цей проект, "Тестовий веб-портал для проведення веб-тестування", спрямований на надання зручного та надійного інструменту для створення та проведення онлайн-тестів. Портал дозволяє аміну створювати тести з різними типами питань, такими як вибір одного варіанта з багатьох, і відповіді на них. Також надається можливість завантажувати картинки до запитань, що дозволяє розширити можливості тестового контролю.

Основні функціональні можливості проекту включають:

- створення та збереження тестових завдань у базі даних;
- можливість обирання різних категорій для опитування;
- підтримка завантаження картинок до запитань для більшого інтерактиву тестів;
- розділення порталу на дві частини: редактор тестів та тестова система;
- реєстрація користувачів з різними ролями, за замовченням при реєстрації користувач отримує роль “Customer”;
- перегляд та редагування даних профілю для користувачів з обмеженням доступу для адміністратора;
- проведення тестування для авторизованих користувачів з відображенням результатів, та їх збереженням в базі даних.

У цьому звіті ми розглянемо детальніше структуру та функціонал веб-порталу, а також опис функцій, доступних для кожного типу користувачів. Крім того, буде представлено опис процесу реєстрації та входу, а також інструкції щодо створення та редагування тестів. Ми також проаналізуємо переваги та перспективи проекту та запропоную можливі напрямки для його подальшого розвитку.

1 Постановка завдання

1.1 Загальний опис

Метою даного проекту є створення веб-порталу для проведення веб-тестування, який надасть зручний та ефективний інструмент для оцінки знань та вмінь користувачів. Портал буде допомагати створювати тести, додавати питання та відповіді до них, а також проводити тестування з можливістю завантаження картинок для більшого інтерактиву.

1.2 Основні функціональні можливості

Редактор тестів:

- Адміністратор (Admin) має можливість створювати, редагувати та видаляти категорії для тестів.
- Адміністратор може додавати різні типи питань (вибір одного варіанта з багатьох) і відповіді до них.
- Картинки можуть бути завантажені для кожного питання для більшого інтерактивного досвіду тестування.

Тестова система:

- Зареєстровані користувачі з ролью "Customer" мають можливість проходити тести.
- Користувачі можуть брати участь у тестуванні з не обмеженням кількості спроб.
- Результати проходження тестів зберігаються в базі даних.

Авторизація та реєстрація:

- Користувачі мають можливість реєструватися на порталі, за замовченням надається роль "Customer".
- Авторизовані користувачі мають доступ до функціоналу відповідно до своєї ролі.

Управління користувачами:

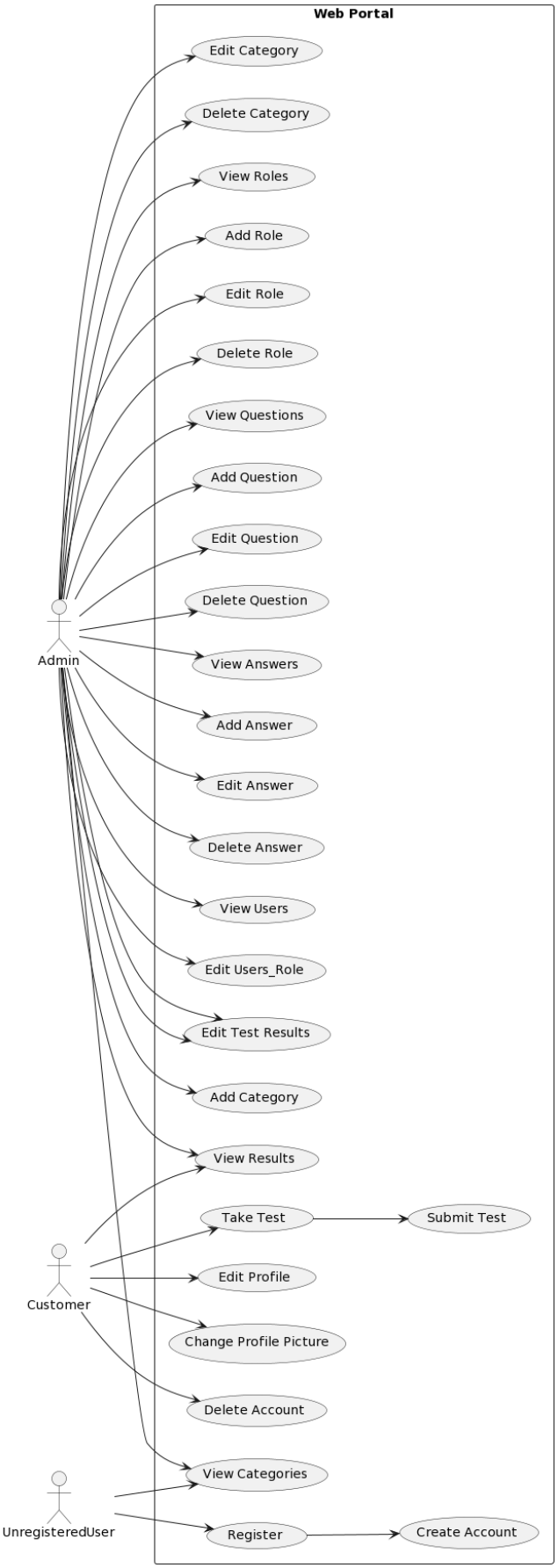
- Адміністратор може переглядати список користувачів і змінювати їх роль.
- Користувачі з роллю "Admin" не можуть бути видалені або змінені, але їхні дані можуть бути кореговані.

Управління категоріями та питаннями:

- Адміністратор має можливість додавати, редагувати та видаляти категорії тестів.
- Адміністратор може додавати та редагувати питання та відповіді, а також змінювати тип питань.

Управління результатами:

- Адміністратор може переглядати результати проходження тестів користувачів, але не може змінювати їх або видаляти.



Мал.1.1 Діаграма використання

1.3 Архітектура системи

Система буде побудована у вигляді веб-додатку з використанням мов програмування PHP для взаємодії з базою даних і обробки запитів від користувачів. Для зберігання даних буде використовуватись реляційна база даних (наприклад, MySQL). Взаємодія з користувачами буде забезпечена за допомогою HTML, CSS і JavaScript для створення інтерфейсу та реалізації взаємодії.

1.4 Очікуваний результат

Очікується створення повноцінного веб-порталу для проведення веб-тестування, який забезпечить зручність у створенні та проходженні тестів, надійність зберігання даних і забезпечення безпеки. Кінцевий продукт повинен включати функції авторизації та реєстрації користувачів, можливість створення та редагування тестів, а також зберігання результатів проходження тестів. Крім того, очікується реалізація функціоналу з обмеженням доступу для адміністратора та змінами ролей користувачів.

2 Проектування системи

Проектування системи передбачає важливий аспект - зберігання всіх даних, необхідних для роботи веб-порталу. В даному проекті ми розглядаємо систему для проведення веб-тестування, де велика кількість інформації, таких як тести, питання, відповіді, користувачі та результати тестувань, потребує зберігання, оновлення та доступу.
Використання бази даних є обґрунтованим рішенням з кількох причин:

1. *Постійне зберігання даних*: База даних дозволяє зберігати всі дані довгостроково. Це важливо для збереження результатів тестування, інформації про користувачів та всіх інших даних, які необхідні для подальшої обробки.
2. *Швидкий доступ до даних*: Використання бази даних дозволяє швидко отримати необхідну інформацію для відображення на веб-сторінках. База даних допомагає зменшити час завантаження сторінок та запитів до сервера.
3. *Ефективна організація даних:* База даних дозволяє структуровано зберігати дані за допомогою таблиць, полів і зв'язків між ними. Це допомагає легше управляти даними та робити потрібні запити до бази даних.
4. *Безпека даних:* Бази даних забезпечують можливість захистити дані за допомогою різних механізмів доступу, автентифікації та авторизації. Це дозволяє уникнути несанкціонованого доступу до важливої інформації.

5. Масштабованість: Використання бази даних дозволяє системі масштабуватися в разі збільшення кількості даних та користувачів. База даних може оптимально обробляти великі обсяги інформації.

6. *Інтеграція з різними компонентами системи:* Застосування бази даних дозволяє інтегрувати різні компоненти системи, наприклад, функціонал редактора тестів та тестової системи, забезпечуючи цілісність даних та послідовність дій.

З урахуванням різноманітності даних, які необхідно зберігати в системі, та необхідності ефективного керування тестами, питаннями, відповідями, користувачами та результатами, база даних стає незамінним елементом інфраструктури системи проведення веб-тестування. Це допоможе створити надійний, ефективний та безпечний веб-портал для користувачів з різними ролями та правами доступу.

2.1 Система управляння базою даних

Система управління базою даних (СУБД) MySQL є однією з найпопулярніших відкритих реляційних баз даних на сьогодні. Вона має ряд переваг та недоліків, які потрібно врахувати при виборі для проекту.

Позитивні сторони MySQL:

1. *Відкритість та безкоштовність:* MySQL доступна на умовах вільної ліцензії, що дозволяє використовувати її безкоштовно та звільняє від необхідності придбання платних ліцензій.

2. *Простота в використанні:* MySQL має дружній та простий для розуміння інтерфейс, що дозволяє швидко оволодіти основами роботи з базою даних.

3. *Добра швидкодія:* MySQL має високу швидкодію при обробці запитів та великих обсягів даних, що робить її ефективною для багатьох веб-проектів.

4. *Масштабованість:* MySQL може ефективно працювати як з невеликими базами даних, так і з дуже великими, що дозволяє масштабувати проект в разі його зростання.

5. *Активна спільнота:* MySQL має велику та активну спільноту розробників, що дозволяє швидко отримувати підтримку та вирішувати проблеми.

Недоліки MySQL:

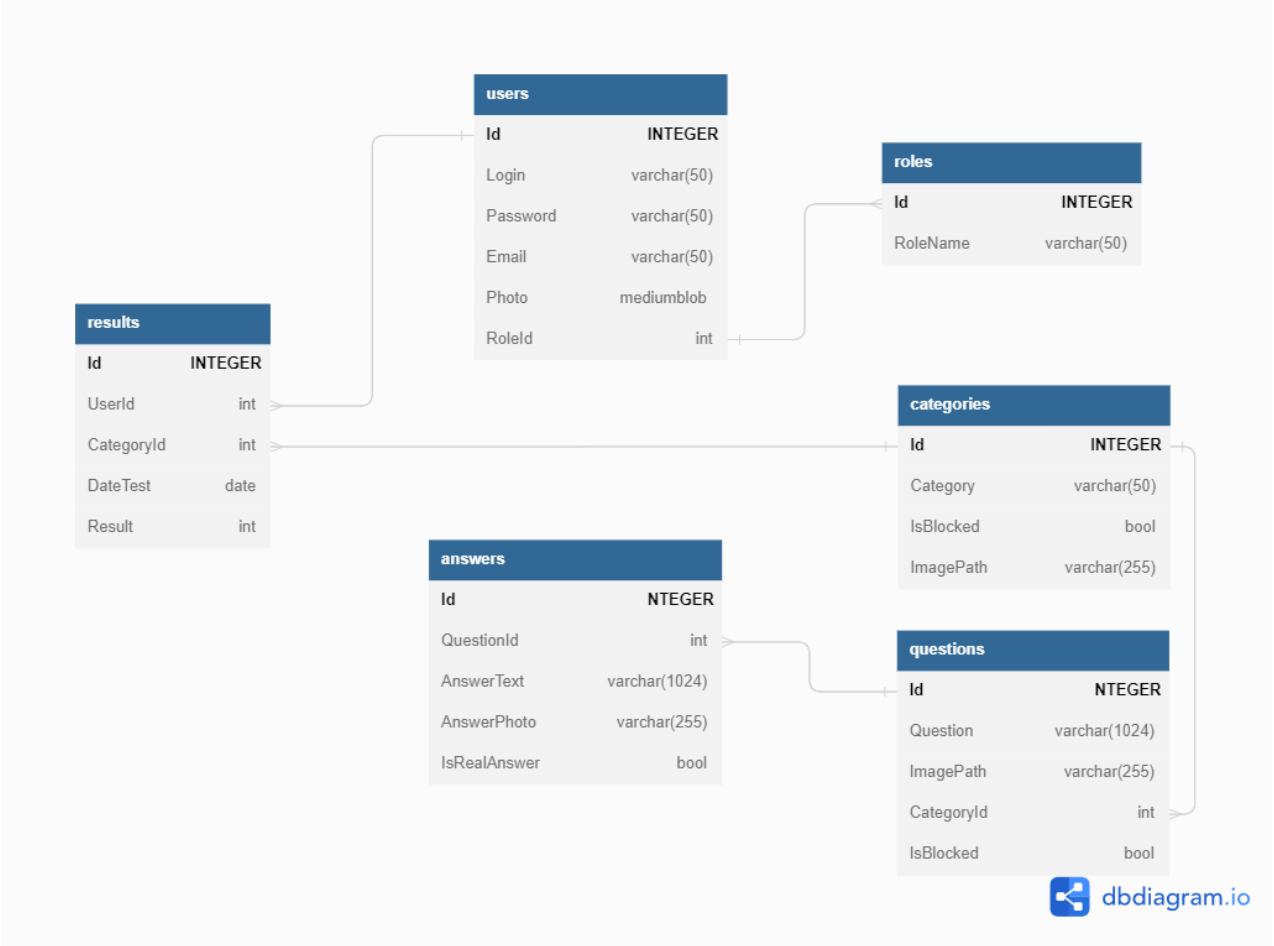
1. *Обмежена підтримка деяких функцій:* У порівнянні з деякими іншими реляційними СУБД, MySQL може мати обмежену підтримку деяких функцій, таких як тригери або віконні функції. Однак, у новіших версіях ці обмеження частково зняті.

2. *Обмеження в роботі з великими об'ємами даних:* Хоча MySQL ефективна в роботі з великими обсягами даних, у деяких випадках, при надмірно великих обсягах, вона може стикатися з обмеженнями продуктивності.

3. *Безпека:* В деяких випадках безпека MySQL може бути під загрозою, якщо не налаштована належним чином. Варто докладно дослідити та прийняти заходи для захисту даних.

4. *Немає підтримки для деяких додаткових функцій:* Деякі додаткові функції, які можуть бути доступні у комерційних СУБД, можуть бути відсутніми у MySQL.

На мал. 2.1 представлена структура таблиць нашого додатка в СУБД.



Мал. 2.1 Структура таблиць бази даних додатка

У випадку з проектом веб-порталу для проведення веб-тестування, MySQL є хорошим вибором. Вона дозволяє ефективно зберігати, керувати та отримувати доступ до великої кількості даних, що потрібні для проведення тестування та збереження результатів. Простота в використанні та підтримка широкого спектру розробників робить MySQL популярним вибором для багатьох веб-проектів. Однак, перед використанням слід ретельно спланувати структуру бази даних та прийняти необхідні заходи для забезпечення безпеки даних.

2.2 Вибір мови програмування

Проект тестового веб-порталу розробляється з використанням різних технологій, таких як PHP, HTML, CSS, JavaScript та AJAX, для створення потужної та зручної системи для проведення тестування.

1. PHP (PHP: Hypertext Preprocessor):

Мова програмування PHP використовується для розробки бекенду веб-порталу. Основна перевага PHP полягає у тому, що вона добре сумісна з веб-сервером Apache та базою даних MySQL, що дозволяє легко забезпечити зв'язок з базою даних та зберігати всі необхідні дані. PHP надає широкі можливості для обробки форм, роботи з сесіями, кукі, а також для взаємодії з зовнішніми API та іншими сервісами. Ця мова дозволяє забезпечити ефективний обмін даними між клієнтом та сервером, а також швидку обробку запитів.

2. HTML (HyperText Markup Language):

Мова розмітки HTML використовується для створення структури веб-сторінок. Вона дозволяє визначити заголовки, абзаци, таблиці та інші елементи сторінки. Основна перевага HTML полягає у простоті та зрозумілості коду, що дозволяє легко створювати статичний контент на сторінках та визначати їх структуру. Завдяки HTML, розробники можуть забезпечити логічну організацію сторінок та додати потрібні елементи для взаємодії з користувачами.

3. CSS (Cascading Style Sheets):

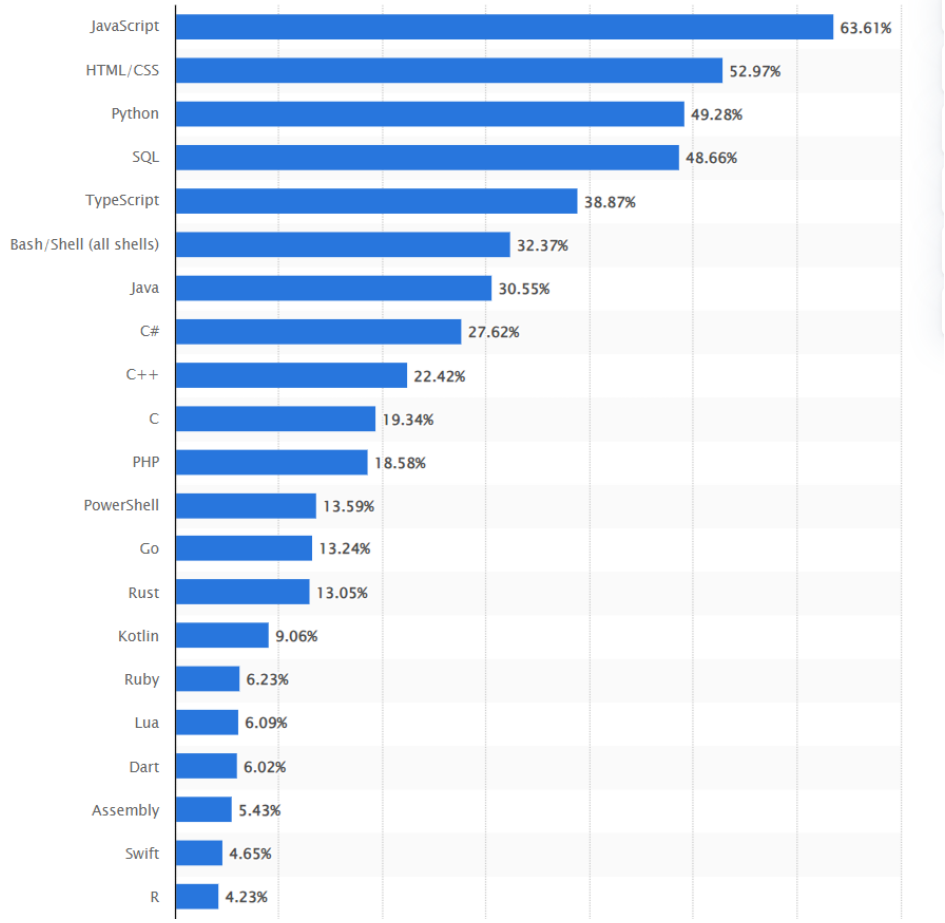
CSS використовується для стилізації веб-сторінок та забезпечення їх зовнішнього вигляду. Завдяки CSS, можна встановлювати кольори, розміри, шрифти, відступи та інші властивості для елементів сторінки. Основна перевага CSS полягає у тому, що вона дозволяє розділити структуру та вигляд сторінки, що спрощує редагування та підтримку дизайну. Завдяки CSS, можна легко змінювати вигляд сторінок, а також робити їх адаптивними під різні пристрої.

У проекті також використовується CSS-фреймворк Bootstrap 5.3. Bootstrap забезпечує готові стилі та компоненти, що дозволяють швидко створювати зручний та привабливий дизайн без необхідності писати багато власного CSS. Він також забезпечує адаптивний дизайн, що дозволяє коректно відображати сторінки на різних пристроях, таких як комп'ютери, планшети та смартфони.

4. JavaScript:

Мова програмування JavaScript використовується для додавання інтерактивності та динамічності на стороні клієнта. Завдяки JavaScript, можна реалізувати різні функції, такі як перевірка правильності вибраної відповіді, анімації, валідацію даних, та інші динамічні ефекти. Основна перевага JavaScript полягає у тому, що вона дозволяє забезпечити інтерактивність та реактивність веб-порталу без необхідності перезавантаження сторінки. Використання JavaScript допомагає створити зручний та взаємодіючий інтерфейс для користувачів, що робить процес тестування більш захоплюючим та забезпечує зручний та швидкий доступ до результатів тестів.

Розглянемо на мал. 2.1 статистичну діаграму використання мов програмування у всьому світі станом на 2023:



Мал. 2.1 Статистичні дані використання мов програмування у світі станом на 2023 рік

Хоча відсоток використання PHP (18.58%) у світі може здатися невеликим порівняно з іншими мовами програмування, вибір PHP для розробки веб-додатку тестового веб-порталу був доцільним і обґрунтованим з кількох причин:

1. Веб-орієнтовані можливості: PHP є однією з найпопулярніших мов програмування для веб-розробки. Вона була спеціально створена для створення динамічних веб-сторінок, і має широкий набір інструментів та бібліотек, що дозволяє швидко розробляти функціональні веб-додатки.

2. Велика спільнота розробників: PHP має велику та активну спільноту розробників. Це означає, що можна легко знайти документацію, приклади коду та підтримку у випадку проблем. Також, існує велика кількість сторонніх бібліотек та фреймворків, які допомагають прискорити розробку та поліпшити якість веб-додатків.

3. Легкість навчання: PHP є досить простою мовою програмування для вивчення, особливо для розробників, які вже знайомі з іншими мовами програмування. Вона має синтаксис, схожий на мови, такі як C або Java, що дозволяє легко переходити з однієї мови на іншу.

4. Швидкість виконання: PHP є інтерпретованою мовою програмування, що дозволяє швидко розробляти та тестувати код без необхідності компіляції. Крім того, з випуском PHP 7, вона стала значно швидшою, що дозволяє забезпечити хорошу продуктивність веб-додатків.

5. Розширені можливості: PHP дозволяє легко інтегрувати з іншими технологіями та сервісами, такими як бази даних, сервери електронної пошти та інші. Це робить її потужним інструментом для створення різноманітних веб-додатків.

6. Велика кількість готових рішень: Завдяки популярності PHP, існує велика кількість готових рішень та сторонніх бібліотек для розробки різних функціональних компонентів, таких як автентифікація, робота з файлами, взаємодія з API та інші. Це дозволяє економити час розробки та знижує ризик помилок.

Загалом, обрів PHP для розробки тестового веб-порталу був обґрунтований через його веб-орієнтовані можливості, велику спільноту розробників, легкість навчання, швидкість виконання, розширені можливості та наявність готових рішень.

В проекті також використовується технологія AJAX (Asynchronous JavaScript and XML), яка дозволяє робити асинхронні запити до сервера з використанням JavaScript без необхідності перезавантаження сторінки. Завдяки цьому, користувачі можуть отримувати дані та результати тестування в режимі реального часу, без затримок. AJAX дозволяє забезпечити плавну та ефективну взаємодію з сервером, що робить процес тестування більш зручним та ефективним.

Комбінація PHP, HTML, CSS, JavaScript та AJAX дозволяє створити повноцінний тестовий веб-портал з великим функціоналом та привабливим інтерфейсом. PHP забезпечує потужний бекенд для роботи з даними та базою даних, HTML та CSS допомагають створити зрозумілий та естетичний дизайн, а JavaScript забезпечує динамічність та інтерактивність, що робить процес тестування більш захоплюючим для користувачів. Такий підхід дозволяє забезпечити зручну та ефективну роботу із тестами та даними користувачів, та створити надійний та функціональний тестовий веб-портал, що відповідає всім потребам та очікуванням користувачів.

2.3 GIT

Тепер, коли ми визначились з усіма технологіями та етапами проектування, можемо розпочинати розробку нашого сайту.

Але перед цим, був створений репозиторій на GitHub, де буде зберігатися цей проект. Це надає нам багато переваг. По-перше, цей репозиторій дозволить нам працювати разом над проектом декількома розробниками, спрощуючи спільну роботу та уникнення конфліктів. Кожен з нас зможе вносити свої зміни, створювати гілки та об'єднувати їх усі разом.

По-друге, завдяки GitHub ми зможемо легко відстежувати стан проекту. Ми будемо мати історію всіх комітів, змін та внесених покращень, що дозволить нам розуміти, наскільки виконана робота і які зміни були внесені в проект. Це допоможе нам краще координувати роботу та вносити необхідні корективи.

Таким чином, ми маємо потужний інструмент для спільної розробки та контролю версій, який дозволить нам ефективно працювати над нашим проектом тестового веб-порталу і забезпечить якість та успіх у його розробці. Разом ми зможемо зробити проект кращим і більш функціональним для користувачів.

Посилання на проект в GitHub

3 Принцип праці програми

Принцип роботи програми базується на використанні Open Server Panel, що є веб-серверним середовищем, яке дозволяє запускати та тестувати веб-додатки на локальному комп'ютері. Цей сервер дозволяє нам створити локальний веб-сервер, що мімікує роботу реального сервера, але працює тільки на нашому комп'ютері.

Для роботи з проектом тестового веб-порталу, першим кроком є встановлення Open Server Panel на комп'ютері. Після успішного встановлення та запуску Open Server Panel, ми отримуємо доступ до веб-сервера, бази даних та інших необхідних сервісів для розробки та тестування нашого проекту.

Після встановлення Open Server Panel та його налаштування, ми можемо почати розробку та тестування нашого тестового веб-порталу.

3.1 Створення бази даних

Для забезпечення можливості підключення до бази даних MySQL у проекті, створимо спеціальну функцію у файлі **courseproject/functions/functions.php**. Ця функція дозволить зручно та безпечно здійснити з'єднання з базою даних. Використовуючи об'єкт PDO (PHP Data Objects), ми забезпечимо універсальний доступ до різних типів баз даних.

functions > functions.php > deleteUserFromSQL

```php
<?
function connect($host = "localhost:3307", $user = "root", $password = "", $dbname = "testssystemdb")
{
    $cs = "mysql:host=$host;dbname=$dbname;charset=utf8;";
    $options = array(
        PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
        PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
        PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES UTF8"
    );

    try {
        $pdo = new PDO($cs, $user, $password, $options);
        return $pdo;
    } catch (PDOException $excep) {
        echo $excep->getMessage();
        return false;
    }
}

```

Мал. 3.1 Функція з’єднання з MySQL

Після створення функції для з'єднання з СУБД, ми можемо розробити таблиці для зберігання даних у проекті. Для цього в файлі **courseproject/createdb/createdb.php** додамо наступний код:

createdb > createdb.php > html > body

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap/5.3.0/css/bootstrap.min.css">
</head>
<body>
<?
include_once("../functions/functions.php");

$pdo = connect();
$tableNameCategories = 'categories';
$tableNameRoles = 'roles';
$tableNameQuestions = 'questions';
$tableNameAnswers = 'answers';
$tableNameUsers = 'users';
$tableNameResults = 'results';

$ct1 = "CREATE TABLE IF NOT EXISTS $tableNameCategories(
    Id int primary key not null auto_increment,
    Category varchar(50) not null unique,
    IsBlocked bool not null default false,
    ImagePath varchar(255)) default charset='utf8'";

$ct2 = "CREATE TABLE IF NOT EXISTS $tableNameRoles(
    Id int primary key not null auto_increment,
    RoleName varchar(50) not null unique) default charset='utf8'";
```

Мал. 3.2 Код створення реляційної бази даних для додатка

createdb > createdb.php > html > body

```
31
32  $ct3 = "CREATE TABLE IF NOT EXISTS $tableNameQuestions(
33      Id int primary key not null auto_increment,
34      Question varchar(1024) not null,
35      ImagePath varchar(255),
36      CategoryId int not null,
37      IsBlocked bool not null default false,
38      FOREIGN KEY (CategoryId) REFERENCES $tableNameCategories(Id) ON DELETE CASCADE) default charset='utf8'";
39
40  $ct4 = "CREATE TABLE IF NOT EXISTS $tableNameAnswers(
41      Id int primary key not null auto_increment,
42      QuestionId int not null,
43      AnswerText varchar(1024) unique,
44      AnswerPhoto varchar(255) unique,
45      IsRealAnswer bool not null,
46      FOREIGN KEY (QuestionId) REFERENCES $tableNameQuestions(Id) ON DELETE CASCADE) default charset='utf8'";
47
48  $ct5 = "CREATE TABLE IF NOT EXISTS $tableNameUsers(
49      Id int primary key not null auto_increment,
50      Login varchar(50) not null unique,
51      Password varchar(255) not null,
52      Email varchar(255) not null,
53      Photo mediumblob,
54      RoleId int not null default 2,
55      FOREIGN KEY (RoleId) REFERENCES $tableNameRoles(Id) ON DELETE CASCADE) default charset='utf8'";
56
57  $ct6 = "CREATE TABLE IF NOT EXISTS $tableNameResults(
58      Id int primary key not null auto_increment,
59      UserId int not null,
60      CategoryId int not null,
61      DateTest date not null,
62      Result int not null,
63      FOREIGN KEY (UserId) REFERENCES $tableNameUsers(Id) ON DELETE CASCADE,
64      FOREIGN KEY (CategoryId) REFERENCES categories(Id) ON DELETE CASCADE) default charset='utf8'";
```

Мал. 3.3 Код створення реляційної бази даних для додатка

createdb > createdb.php > html > body

```
65
66  try{
67      $pdo->exec($ct1);
68      $pdo->exec($ct2);
69      $pdo->exec($ct3);
70      $pdo->exec($ct4);
71      $pdo->exec($ct5);
72      $pdo->exec($ct6);
73      echo "<div class='alert alert-success'>Table $tableNameCategories created successfully.</div>";
74      echo "<div class='alert alert-success'>Table $tableNameRoles created successfully.</div>";
75      echo "<div class='alert alert-success'>Table $tableNameQuestions created successfully.</div>";
76      echo "<div class='alert alert-success'>Table $tableNameAnswers created successfully.</div>";
77      echo "<div class='alert alert-success'>Table $tableNameUsers created successfully.</div>";
78      echo "<div class='alert alert-success'>Table $tableNameResults created successfully.</div>";
79  }catch(PDOException $ex){
80      echo "Error creating table: ".$ex->getMessage();
81  }
82  ?>
83  </body>
84  </html>
```

Мал. 3.3 Код створення реляційної бази даних для додатка

3.2 Загальний опис роботи ‘courseproject/index.php’

Файл `**index.php**` є головним файлом веб-порталу. Його принцип роботи включає наступні етапи:

1. Початок сесії: Спершу в файлі стартує сесія за допомогою `**session_start()**`. Це дозволяє зберігати та обробляти дані між різними сторінками в рамках одного відвідувача.

2. Визначення поточної сторінки: Зчитуємо значення параметру `page` з URL, якщо він переданий, інакше встановлюємо його значення за замовчуванням на 1.

3. Підключення стилів та скриптів: Завантажуємо необхідні зовнішні стилі та скрипти, такі як Bootstrap і іконки. Також використовується внутрішній CSS для встановлення деяких стилів.

4. Навігаційне меню: Застосовується `**include_once("pages/menu.php")**` для включення вмісту сторінки **menu.php**, яка містить навігаційне меню з посиланнями на різні сторінки.

5. Секція вмісту сторінки: Застосовується `**switch**` для вибору потрібної сторінки, в залежності від значення параметру `**page**`. Це дозволяє відображати відповідний вміст для різних сторінок, наприклад, головної сторінки, сторінки авторизації, адміністратора тощо.

6. Футер: Розміщення футера сторінки з контактною інформацією та посиланнями на соціальні мережі.

7. Підключення скриптів: Завантаження зовнішніх скриптів, таких як `Bootstrap`, для коректної роботи динамічних елементів сторінки.

Принцип роботи полягає в тому, що в залежності від значення параметру `**page**`, на сторінці виводиться вміст відповідної сторінки. Наприклад, якщо `**page=1**`, виводиться головна сторінка, якщо `**page=2**`, виводиться сторінка "**Home**", і так далі. При цьому, якщо користувач не авторизований, але спробує звернутися до сторінки, яка доступна тільки авторизованим користувачам (наприклад, сторінка "**Home**" або "**Admin**"), то йому буде виведено повідомлення про необхідність авторизації та перенаправлено на головну сторінку. Також передбачено виведення повідомлень про помилки, наприклад, якщо сторінку не знайдено.

Основна ідея полягає в тому, що кожна сторінка розділена на окремий файл, що полегшує редагування та дозволяє добре структурувати проект. Також використовуються сесії для зберігання даних, що дозволяє здійснювати авторизацію користувачів та контролювати доступ до окремих сторінок.

3.3 Загальний опис папки ‘pages’

Папка "pages" містить наступні файли:

1. **admin.php**: Файл, який містить сторінку для адміністратора з можливістю управління різними аспектами проекту.

2. **home.php**: Файл, який представляє сторінку "Home" для зареєстрованих користувачів, де вони можуть здійснювати певні дії чи переглядати особисту інформацію.

3. **login.php**: Файл, що містить сторінку авторизації, де користувачі можуть увійти в свої облікові записи.

4. **logout.php**: Файл, що містить процедуру виходу з облікового запису та завершення сесії.

5. **main.php**: Файл сторінки "Tests", який містить основний контент проекту, зокрема тести та питання.

6. **menu.php**: Файл, що включає навігаційне меню з посиланнями на різні сторінки проекту.

7. **registration.php**: Файл, що представляє сторінку реєстрації нового користувача.

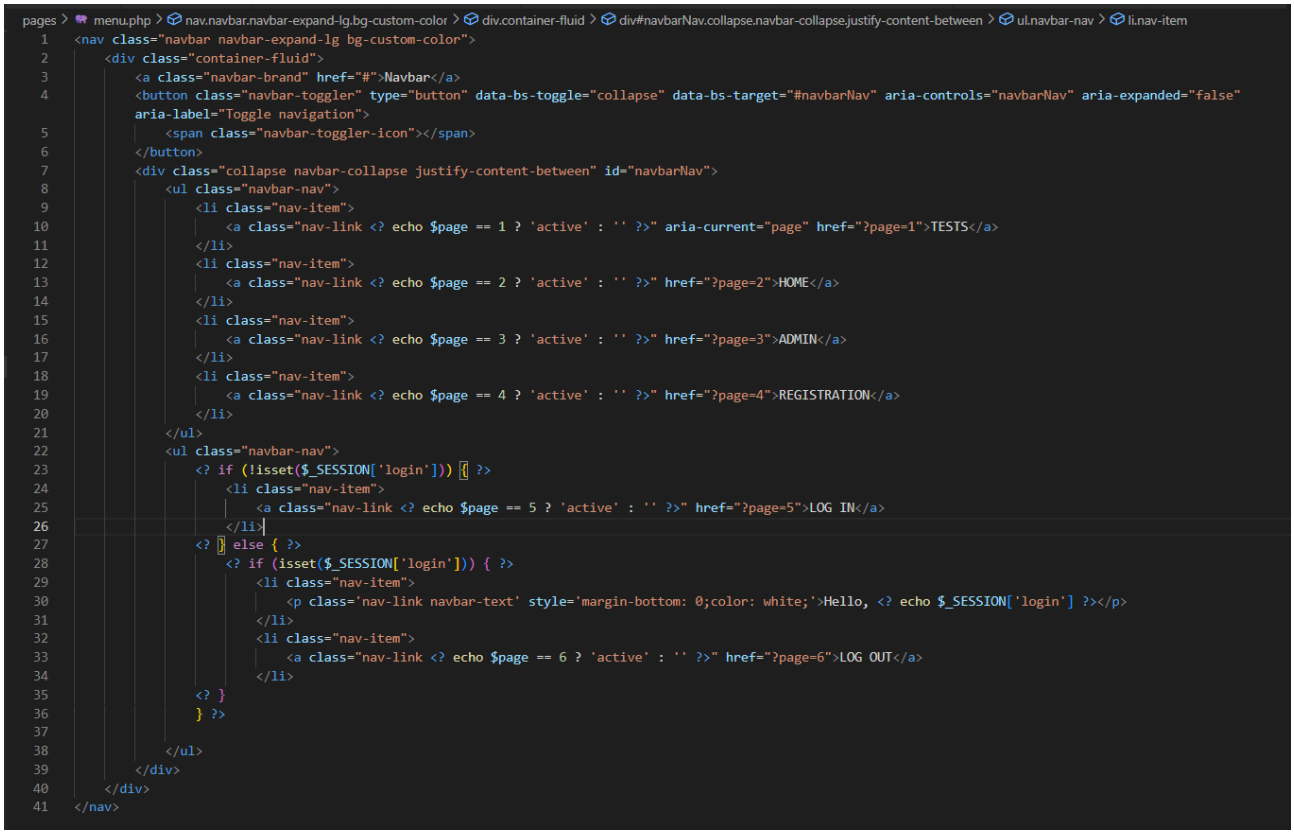
Ці файли допомагають структурувати проект та забезпечують функціональність різних сторінок, дозволяючи користувачам здійснювати різні дії, переходити між різними розділами та керувати проектом.

3.3.1 Загальний опис ‘pages/menu.php’

Файл `menu.php`, мал.3.4, містить навігаційне меню, яке виводить посилання на різні сторінки проекту, залежно від обраного "page". В основній частині меню розміщено чотири посилання на сторінки "TESTS", "HOME", "ADMIN" та "REGISTRATION". Залежно від активної сторінки, відповідному посиланню додається клас "active" для візуального підсвічування.

Для користувачів, які не увійшли в систему, показується посилання на сторінку "LOG IN", де вони можуть здійснити авторизацію. Якщо користувач вже авторизований (сесія запущена та змінна $_SESSION['login'] існує), замість посилання на "LOG IN" виводиться повідомлення з привітанням та іменем користувача, а також посилання "LOG OUT" для виходу з системи.

Це меню забезпечує зручний спосіб переходу між різними сторінками проекту та взаємодію з авторизацією користувачів. Коли користувач натискає на посилання, відповідна сторінка завантажується через параметри URL "?page=". Таким чином, допомагається користувачеві легко навігуватися по проекту та виконувати потрібні дії.



Мал. 3.4 Код створення меню

3.3.2 Загальний опис ‘pages/admin.php’

Сторінка **admin.php** - це сторінка адміністратора, де адміністратор проекту може керувати різними аспектами системи. На цій сторінці адміністратор може бачити та змінювати інформацію, пов'язану з категоріями тестів, ролями користувачів, питаннями, відповідями, списком користувачів та результатами тестів.

Вигляд сторінки адміністратора поділений на два стовпці. В першому стовпці відображаються розділи керування категоріями тестів та ролями користувачів. Вони мають посилання, при кліці на яке, відкривається відповідна сторінка з можливістю змінювати інформацію. Наприклад, після кліку на "Categories" виводиться сторінка **admin/categories.php**, де адміністратор може додавати, редагувати чи видаляти категорії тестів.

У другому стовпці знаходяться посилання на сторінки, де адміністратор може переглядати дані або здійснювати дії з питаннями, відповідями, користувачами та результатами тестів. Наприклад, відкривши сторінку **admin/questions.php**, адміністратор може переглядати всі наявні питання та вносити до них зміни. Аналогічно для інших сторінок.

Окрім того, на кожному посиланні є зображення-іконка "new_click.png", що показує, що заходячи на сторінку, користувач може виконати нові дії або переглянути нові дані.

Ця сторінка забезпечує адміністратору зручний спосіб керування різними аспектами системи, а також легкий доступ до інформації, що стосується користувачів та тестів.

3.3.2 Загальний опис ‘pages/admin.php’

На сторінці **home.php** знаходиться основний контент для зареєстрованого користувача після успішного входу в систему.
Основними елементами сторінки **home.php** є:

- Показ інформації про користувача: На сторінці виводиться ім'я користувача, яке взяте з сесії після авторизації. Це дає користувачеві можливість бачити свої персональні дані.
- Перелік тестів: Користувач може бачити перелік виконаних тестів.
- Зміна даних: Користувач може змінити свої фото і поштову адресу або додати фото в свій профіль. Користувач може видалити свій аккаунт якщо його роль **Customer**.
- Вихід з системи: Є можливість вийти зі свого облікового запису. Після виходу з системи користувач переадресовується на сторінку **index.php?page=1**, де він може бачити загальний перелік тестів для неавторизованих користувачів.

Принцип роботи сторінки **home.php** полягає в тому, щоб надати користувачеві зручний доступ до основної інформації і дозволити взаємодіяти з системою шляхом перегляду тестів та зміни свого пароля. Це забезпечує зручність та персоналізацію користування системою для зареєстрованих користувачів.

3.3.3 Загальний опис ‘pages/login.php’

pages/login.php відповідає за обробку процесу входу користувача до системи. Основні кроки і принципи роботи програми такі:

- Спершу, ми включаємо необхідний файл **functions.php**, який містить різноманітні функції для роботи з базою даних та іншими функціональними можливостями.
- Сторінка розділена на дві частини за допомогою умови **if (!isset($_POST['submit']))**. У першій частині виводиться форма для входу, де користувач має ввести свій логін та пароль.
- Якщо користувач відправляє форму за допомогою натискання кнопки "Log in", тоді виконується друга частина коду. В ній:
- Отримуємо дані, які ввів користувач (логін та пароль).
- Перевіряємо ці дані, порівнюючи їх з даними з бази даних.
- Якщо користувач ввів правильний логін та пароль, то встановлюємо змінну **$isLogIn** в **true**, і це дозволяє нам зрозуміти, що авторизація успішна.

- Якщо авторизація пройшла успішно, зберігаємо інформацію про користувача в сесії (**$_SESSION**). В даному випадку, ми зберігаємо логін користувача, його роль та ідентифікатор (ID) в таблиці користувачів.
- Виводимо сповіщення про успішну авторизацію.
- Після чого, через 2 секунди, за допомогою JavaScript, перенаправляємо користувача на головну сторінку (**index.php?page=1**).
- Якщо ж авторизація була неуспішною (змінна **$isLogIn** дорівнює **false**), тоді виконуємо іншу гілку коду:
- Спершу, перевіряємо, чи користувач був авторизований раніше (перевірка за допомогою **isset($_SESSION['login'])**). Якщо так, тоді розлогінюємо його, видаляючи інформацію про нього з сесії.
- Потім виводимо сповіщення про невірний логін або пароль.
- Також, через 2 секунди, за допомогою JavaScript, перенаправляємо користувача на сторінку з формою для входу (**index.php?page=5**), щоб він міг спробувати увійти знову.

Таким чином, код **pages/login.php** виконує перевірку введених даних користувача, перевіряє, чи такий користувач існує в базі даних, і здійснює авторизацію або розлогінювання, а також виводить відповідні сповіщення і перенаправляє користувача на інші сторінки при необхідності.

3.3.4 Загальний опис ‘pages/logout.php’

Файл **pages/logout.php** відповідає за вихід користувача з системи (логаут). Основний принцип роботи програми на цій сторінці такий:

- Спочатку ми викликаємо функцію **session_start()** для початку роботи із сесіями, оскільки ми хочемо видалити деякі дані з сесії.
- Далі ми викликаємо функцію **unset()** для видалення деяких змінних з сесії, які були збережені там при авторизації. Конкретно, це включає змінні **$_SESSION['login']**, **$_SESSION['roleUser']** та **$_SESSION['id']**.
- Після цього перевіряємо, чи є деякі інші змінні у сесії, які стосуються проходження тестів (**$_SESSION['randomQuestion']**, **$_SESSION['category']**, **$_SESSION['answers']**, **$_SESSION['currentQuestionIndex']**). Якщо так, то також видаляємо їх за допомогою функції **unset()**.
- В кінці змінюємо адресу сторінки за допомогою функції **header("Location: index.php?page=5")**. У даному випадку, ми перенаправляємо користувача на сторінку логіну (**index.php?page=5**), після того, як він вийшов з системи.

Таким чином, код **pages/logout.php** просто видаляє інформацію про користувача з сесії, що робить його неавторизованим, та перенаправляє його на сторінку логіну.

3.3.5 Загальний опис ‘pages/main.php’

Файл **pages/main.php** відповідає за відображення головної сторінки тестів. Основний принцип роботи програми на цій сторінці такий:

- Спочатку ми включаємо файли функцій, які необхідні для роботи на цій сторінці, за допомогою **include_once("/OSPanel/domains/CourseProject/functions/functions.php");**.
- Виводимо заголовок "TESTS" за допомогою **<h3>TESTS</h3>**.
- Створюємо контейнер **div** з класом "container", в якому розміщуємо карусель із категоріями тестів.
- Виконуємо запит до бази даних, щоб отримати всі доступні категорії тестів, які не заблоковані. Для цього використовуємо функцію **getAllCategoriesIsNotBloked()**.
- Використовуючи цикл **while**, перебираємо результат запиту і для кожної категорії створюємо картку **div** з класом "card". Кожна карта містить наступне:
- Зображення категорії, яке виводимо за допомогою тегу **<img>** із властивістю **src** зі значенням **$row['ImagePath']**.
- Заголовок категорії, який має посилання **a**, яке веде на сторінку тестів для цієї категорії. Посилання містить параметри **?page=<?=$countLink?>&category=<?=$encodedCategory?>**, де **$countLink** - номер сторінки для тестів, а **$encodedCategory** - назва категорії, закодована для використання в URL.
- Заголовок категорії виводимо за допомогою **<h4>** з властивістю **strtoupper($row['Category'])**, щоб вивести заголовок у верхньому регістрі.

Таким чином, код **pages/main.php** виводить на сторінку головну сторінку тестів, де кожна категорія тестується окремою карткою з зображенням та посиланням на відповідну сторінку тестів для цієї категорії.

3.3.6 Загальний опис ‘pages/registration.php’

Файл **registration.php** містить форму реєстрації користувачів і обробку введених даних. Основний принцип роботи програми на цій сторінці такий:

- Спочатку включаємо файли функцій, які необхідні для роботи на цій сторінці, за допомогою **include_once("/OSPanel/domains/CourseProject/functions/functions.php");**.
- Виводимо заголовок "Registration Form" за допомогою **<h2>Registration Form</h2>**.
- Перевіряємо, чи форма була відправлена за допомогою **if(!isset($_POST['submit']))**. Якщо форма ще не була відправлена, виводимо форму реєстрації з допомогою HTML-коду.

- Якщо форма була відправлена (тобто був натиснутий кнопку "Register"), виконуємо наступні кроки:
- Створюємо клас **User**, який представляє об'єкт користувача зі змінними **login**, **password** та **email**.
- Отримуємо дані, введені користувачем, з полями **Login**, **Password**, **ConfirmPassword** та **Email**.
- Виконуємо перевірку на валідність пароля за допомогою функції **validatePassword()**. Якщо пароль не відповідає критеріям, виводимо повідомлення про помилку і перенаправляємо користувача назад на сторінку реєстрації через 2.5 секунди.
- Перевіряємо, чи підтвердження пароля відповідає введеному паролю. Якщо ні, виводимо повідомлення про помилку і перенаправляємо користувача назад на сторінку реєстрації негайно.
- Якщо пароль пройшов перевірку та підтвердження пароля є правильним, шифруємо пароль за допомогою функції **hashPassword()**.
- Створюємо об'єкт класу **User** зі змінними **login**, зашифрованим **password** та **email**.
- Викликаємо функцію **register()**, яка додає дані користувача до бази даних.
- Виводимо повідомлення про успішну реєстрацію і перенаправляємо користувача на сторінку входу через 2 секунди.

Таким чином, на сторінці реєстрації користувачів здійснюється перевірка та обробка даних, які вводить користувач у форму реєстрації. Після успішної реєстрації користувач перенаправляється на сторінку входу, або отримує повідомлення про помилку, якщо введені дані не відповідають вимогам.

3.3.7 Загальний опис ‘pages/admin/...’

answers.php :

Файл "answers.php" в папці "admin" містить код для відображення, додавання та керування відповідями на питання. Він включає наступні функціональності:

- Показати/приховати список відповідей.
- Відображення списку відповідей у вигляді таблиці.
- Додавання нової відповіді через форму.
- Вибір типу відповіді (текст або зображення).
- Редагування відповіді.
- Обробка форм та додавання відповідей до бази даних.
- Видалення відповідей.

Цей файл дозволяє адміністраторам керувати відповідями для різних питань у системі тестування.

categories.php :

Файл "categories.php" містить код для керування категоріями в системі. Основні елементи коду:

- Форма для додавання нової категорії:
- Поле для введення назви категорії.
- Поле для завантаження фотографії, яка буде асоційована з категорією.
- Кнопка "Add" для відправлення форми та додавання нової категорії.

Цей код дозволяє адміністраторам додавати нові категорії до системи, вказуючи їх назви та фотографію, яка буде відображатися в системі. Також в коді присутня функція **editCategory(categoryId)**, яка виконує перехід на іншу сторінку з параметром **categoryId** при натисканні кнопки "Edit".

questions.php :

Файл "questions.php" включає код для управління питаннями в системі. Основні елементи коду:

1. Переключальна кнопка для відображення списку питань:

- Кнопка "Show list of Questions" відображає або приховує список питань.

2. Список питань:

- Таблиця, що відображає питання та їх асоційовані дані, такі як ідентифікатор, текст питання, шлях до зображення, категорія та інші.
- При наведенні курсора на зображення, його розмір збільшується.
- Кнопка "Edit" для редагування питання.

3. Форма для додавання нового питання:

- Поле для введення тексту питання.
- Поле для завантаження зображення, пов'язаного з питанням.
- Вибір категорії зі списку.
- Кнопки "Add" для відправлення форми додавання питання та "Delete" для видалення питань.

Файл "questions.php" також включає наступний скрипт:

1. Функція `editQuestion(questionId)`:

- Виконує перехід на іншу сторінку з параметром `questionId` при натисканні кнопки "Edit".
- Ця функція дозволяє адміністраторам редагувати конкретне питання.

2. Функція `showHidden()`:

- Відображає або приховує список питань при натисканні кнопки "Show list of Questions".
- Змінює текст кнопки в залежності від поточного стану списку.

3. Функція `enlargeImage(imageElement)`:

- Збільшує розмір зображення при наведенні курсора на нього.
- Змінює курсор на показник, вказуючи на можливість збільшення зображення.

4. Функція `resetImage(imageElement)`:

- Повертає розмір зображення до вихідного значення після зняття курсора.
- Відновлює стандартний курсор для зображення.

Ці функції додають інтерактивність до сторінки, дозволяючи адміністраторам комфортно редагувати та переглядати питання.

Також дозволяє адміністраторам додавати нові питання до системи, вказуючи текст, асоційоване зображення та категорію.

results.php :

Файл "results.php" надає адміністратору можливість переглядати результати всіх користувачів. У цьому файлі виводяться результати тестів, які користувачі пройшли в системі. Адміністратор може переглядати ці результати та аналізувати їх.

Загальна структура "results.php":

1. Створення таблиці, яка відображатиме результати.
2. Запит до бази даних для отримання результатів користувачів.
3. Виведення результатів в таблицю.

Цей файл допомагає адміністраторам відстежувати успішність та активність користувачів у системі, дозволяючи їм зосередитися на потенційних проблемах та вдосконаленні досвіду користувачів.

roles.php :

Файл "roles.php" дозволяє адміністратору керувати ролями користувачів у системі.

Основні елементи коду:

1. Таблиця, яка відображає список ролей в системі.
2. Запит до бази даних для отримання ролей.
3. Виведення ролей в таблицю.
4. Форма для додавання нової ролі.
5. Поле для введення назви ролі.
6. Кнопка "Add" для додавання нової ролі.

Цей файл дозволяє адміністраторам додавати нові ролі до системи та керувати існуючими ролями користувачів. Ролі є важливим елементом системи контролю доступу та безпеки.

Крім того, цей скрипт також дозволяє відправити запит до сторінки "index.php?page=8" для редагування ролі з певним ідентифікатором (roleId). Це використовується для відображення форми редагування ролі на сторінці та взаємодії з нею для зміни інформації про роль у системі.

users.php :

Файл "users.php" відповідає за управління користувачами в системі.

Основні елементи коду:

1.Таблиця, яка відображає список користувачів у системі.

2.Запит до бази даних для отримання даних про користувачів.

3.Виведення даних про користувачів у таблицю.

4.Можливість редагування даних про користувачів.

5.Поля для відображення імені користувача, електронної пошти, фотографії та ролі.

6.Кнопка "Edit" для переходу до сторінки редагування користувача.

Сценарій також містить JavaScript-функцію з назвою `editUser(userId)`. Ця функція призначена для переходу користувача на конкретну сторінку при виклику. Сторінка, на яку відбувається перехід, має адресу "index.php?page=11" і до неї додається параметр запиту "userId", який містить значення параметра `userId`, переданого до функції.

Цей функціонал використовується для спрощеного доступу адміністраторів до сторінки редагування конкретного користувача. Передача ідентифікатора користувача дозволяє адміністраторам зручно здійснювати зміни в даних користувача через відповідну сторінку редагування. Це покращує можливості управління користувачами в системі.

3.3.8 Загальний опис ‘pages/edit/...’

answeredit.php :

Файл "answeredit.php" відповідає за редагування відповідей на питання у системі.

Основні елементи сторінки:

1.HTML-структура з підключенням стилів та скриптів для правильного відображення.

2.PHP-код для обробки редагування відповіді. Він визначає параметр "answerId" з URL та звертається до бази даних для отримання інформації про відповідь з вказаним ідентифікатором.

Форма для редагування відповіді містить:

1.Вибір питання, до якого відноситься відповідь.

2.Тип відповіді: текст або фотографія.

3.Поле для введення тексту відповіді або відображення завантаженої фотографії.

4.Вибір правильності відповіді (істинна чи хибна).

JavaScript забезпечує інтерактивність форми, відображаючи потрібне поле залежно від вибору типу відповіді.

Цей файл дозволяє адміністраторам редагувати відповіді на питання в системі. Редагування тексту та фотографії відповідей є важливою складовою управлінням змістом питань та відповідей в системі навчання.

categoryedit.php :

Файл "categoryedit.php" відповідає за редагування категорій у системі.

Основні елементи сторінки:

1. HTML-структура з підключенням стилів та скриптів для правильного відображення.

2. PHP-код для обробки редагування категорії. Він перевіряє наявність параметра "categoryId" у URL та звертається до бази даних для отримання інформації про категорію за вказаним ідентифікатором.

Форма для редагування категорії містить:

1. Поле для введення назви категорії.

2. Прапорець для вказання, чи категорія заблокована.

3. Поле для завантаження фотографії, пов'язаної з категорією.

4. Поточне зображення категорії, якщо воно існує.

5. Можливість збереження змін.

Цей файл дозволяє адміністраторам редагувати параметри категорій у системі, включаючи назву, статус блокування та зображення. Редагування категорій є важливою складовою системи організації контенту та надання користувачам зручності у виборі відповідного матеріалу.

questionedit.php :

Файл "questionedit.php" відповідає за редагування питань у системі.

Основні елементи сторінки:

1. HTML-структура з підключенням стилів та скриптів для правильного відображення.

2. PHP-код для обробки редагування питання. Він перевіряє наявність параметра "questionId" у URL та звертається до бази даних для отримання інформації про питання за вказаним ідентифікатором.

Форма для редагування питання містить:

1. Поле для введення тексту питання.

2. Поле для завантаження фотографії, пов'язаної з питанням.

3. Поле для вибору категорії питання з використанням випадаючого списку.

4. Поточне зображення питання, якщо воно існує.

5. Можливість збереження змін.

Цей файл дозволяє адміністраторам редагувати параметри питань у системі, включаючи текст питання, фотографію та категорію. Редагування питань є важливою складовою системи формулювання контенту та вдосконалення усіх аспектів вивчення.

roleedit.php :

Файл "roleedit.php" відповідає за редагування ролей користувачів у системі.

Основні елементи сторінки:

1. HTML-структура з підключенням стилів та скриптів для правильного відображення.

2. PHP-код для обробки редагування ролі. Він перевіряє наявність параметра "roleId" у URL та звертається до бази даних для отримання інформації про роль за вказаним ідентифікатором.

Форма для редагування ролі містить:

1. Поле для введення назви ролі.

2. Можливість збереження змін.

Цей файл дозволяє адміністраторам редагувати назви ролей користувачів у системі. Ролі є важливим елементом управління доступом та функціональністю користувачів. Редагування ролей відіграє важливу роль у забезпеченні належної безпеки та визначенні прав доступу для різних груп користувачів.

useredit.php :

Файл "useredit.php" відповідає за редагування ролі користувача в системі.

Основні елементи сторінки:

1. HTML-структура з підключенням стилів та скриптів для належного відображення.

2. PHP-код для обробки редагування користувача. Він перевіряє, чи є параметр "userId" у URL, і звертається до бази даних для отримання інформації про користувача за вказаним ідентифікатором. Вивід інформації про користувача, зокрема, його логін, електронну пошту, фотографію.

Форма для редагування ролі користувача, яка містить:

1. Вибір ролі за допомогою випадаючого списку.

2. Можливість збереження змін.

Цей файл дозволяє адміністраторам редагувати ролю користувача в системі, змінюючи її через відповідну сторінку редагування. Роль визначає права та обмеження користувача в системі. Редагування ролі впливає на функціональні можливості користувача та його доступ до різних розділів системи.

3.3.9 Загальний опис ‘pages/user/...’

addphotouser.php :

Файл "addphotouser.php" відповідає за додавання фотографії користувача до системи.

Основні елементи сторінки:

1. HTML-структура з формою для завантаження фотографії користувача.

2. Форма містить відповідне поле для завантаження фотографії та кнопку "Add" для виконання дії.

3. PHP-код, який обробляє завантаження фотографії:

1. Перевіряє наявність параметра "userId" у URL.

2. Перевіряє, чи була обрана фотографія та чи немає помилок завантаження.

3. Перевіряє розмір фотографії та порівнює його з максимально допустимим розміром (500 кБ). Якщо розмір відповідає вимогам, то відбувається збереження фотографії у базу даних для відповідного користувача.

Цей файл дозволяє користувачу, який вже увійшов до системи, додавати фотографію. Фотографія відображається поруч з даними користувача і може слугувати для ідентифікації та персоналізації облікового запису.

edituser.php :

Файл "edituser.php" відповідає за редагування даних користувача в системі.

Основні елементи сторінки:

1. PHP-код, який отримує параметр "userId" з URL та отримує дані користувача за вказаним ідентифікатором.

Форма для редагування даних користувача, яка містить:

1. Поля для редагування електронної пошти користувача.

2. Поле для завантаження фотографії користувача. Якщо у користувача вже є фотографія, то вона відображається поруч з полям редагування.

3. Кнопка "Save" для виконання збереження змін. PHP-код для обробки змін:

Перевіряє наявність параметра "edituser" у POST-запиті. Отримує дані з форми, включаючи ідентифікатор користувача та новий емейл. Перевіряє, чи відбувається завантаження нової фотографії. Проводить перевірку розміру фотографії та зберігає її у базу даних, якщо розмір відповідає вимогам. Оновлює дані користувача у базі даних після редагування.

Цей файл дозволяє користувачеві змінювати свої дані, включаючи електронну пошту та фотографію, і зберігати ці зміни у системі. Фотографія допомагає ідентифікувати користувача та надає персоналізацію його облікового запису.

removeuser.php :

Файл "removeuser.php" відповідає за видалення облікового запису користувача з системи.

Основні елементи сторінки:

1. PHP-код, який отримує параметр "userId" з URL та викликає функцію для видалення користувача з бази даних.
2. Код відновлення сесій та очищення змінних сесії, щоб коректно вийти з облікового запису.
3. Перенаправлення користувача на іншу сторінку після видалення аккаунту.

Цей файл дозволяє користувачам видаляти свої облікові записи з системи, одночасно очищаючи їхню сесію та перенаправляючи їх на визначену сторінку після видалення.

3.3.10 Загальний опис ‘pages/tests/...’

testbycategory.php :

Файл "testbycategory.php", яка відповідає за проведення тестів за певною категорією питань.

Основні елементи сторінки:

1. Підключення необхідних функцій з файлу "functions.php".
2. Отримання параметра "category" з URL та розпочаття сесії.
3. Перевірка наявності питань для тестування в даній категорії.
4. Виведення питань та відповідей на сторінці.
5. Перевірка обраної відповіді, обрахунок результатів тесту та виведення їх.
6. Запис результатів тесту в базу даних.
7. Завершення сесії після завершення тесту.

Цей код дозволяє користувачам проходити тестування за обраною категорією питань, відображати питання та відповіді, а також підраховувати результати тесту та зберігати їх в базі даних.

set_session_variable.php :

Файл "set_session_variable.php" служить для очищення сесійних змінних, які використовуються для збереження даних під час тестування. Коли користувач хоче почати новий тест або змінити категорію, цей файл видаляє збережені дані, такі як обрана категорія, питання, відповіді та інші параметри, щоб підготувати сесію для нового тестування.

3.4 Загальний опис папки ‘functions/...’

functions.php:

Папка “functions” має в собі єдиний файл "functions.php", який містить низку функцій, які використовуються для взаємодії з базою даних та обробки даних у вашому проекті.

Основні функції:

- **hashPasswor($password)** - Генерує хеш паролю для збереження в базі даних.
- **validatePassword($password)** - Перевіряє, чи відповідає пароль вимогам до складності.
- **register($user)** - Реєструє нового користувача в базі даних.
- **getUsersFromSQL()** - Отримує список всіх користувачів з бази даних.
- **getRoleUser($login)** - Отримує роль користувача за його логіном.
- **getIdUser($login)** - Отримує ідентифікатор користувача за його логіном.
- **getCategoryById($id)** - Отримує інформацію про категорію за ідентифікатором.
- **getCategoryId($category)** - Отримує ідентифікатор категорії за назвою.
- **getRoleById($id)** - Отримує інформацію про роль за ідентифікатором.
- **getQuestionById($id)** - Отримує інформацію про питання за ідентифікатором.
- **getAnswerById($id)** - Отримує інформацію про відповідь за ідентифікатором.
- **getAnswerByQuestionId($id)** - Отримує список відповідей за ідентифікатором питання.
- **getRealAnswerByQuestionId($id)** - Отримує інформацію про правильну відповідь за ідентифікатором питання.
- **getUserById($id)** - Отримує інформацію про користувача за ідентифікатором.
- **getAllCategories($category = '')** - Отримує список усіх категорій з можливістю фільтрації за назвою.
- **getAllCategoriesIsNotBloked()** - Отримує список усіх не заблокованих категорій.
- **getAllQuestions()** - Отримує список усіх питань.

- **getAllQuestionsIsNotBlockedLanguageRandom($category)** - Отримує список питань обраної категорії, які не заблоковані та перемішані в випадковому порядку.
- **getAllRoles()** - Отримує список усіх ролей.
- **getAllResultsByUserId($userId)** - Отримує список усіх результатів тестування користувача за його ідентифікатором.
- **updateCategory($category, $isBlocked, $categoryId)** - Оновлює дані категорії в базі даних.
- **updateRole($role, $roleId)** - Оновлює назву ролі в базі даних.
- **updateQuestion($question, $categoryId, $questionId)** - Оновлює дані питання в базі даних.
- **updateAnswer($questionId, $answerText, $answerPhoto, $isRealAnswer, $answerId)** - Оновлює дані відповіді в базі даних.
- **updateUser($roleId, $userId)** - Оновлює дані користувача (роль) в базі даних.
- **updateUserWithPhoto($userId, $email, $photoData)** - Оновлює дані користувача (email, фото) в базі даних.
- **deleteQuestion($questionId)** - Заблоковує питання в базі даних.
- **deleteAnswer($answerId)** - Видаляє відповідь з бази даних.
- **deleteUserFromSQL($userId)** - Видаляє користувача з бази даних.
- **addPhotoUserToSQL($photoData, $userId)** - Додає фото користувача до бази даних.
- **writeResultToSQLFromTest($userId, $categoryId, $dateTest, $result)** - Записує результати тестування в базу даних.

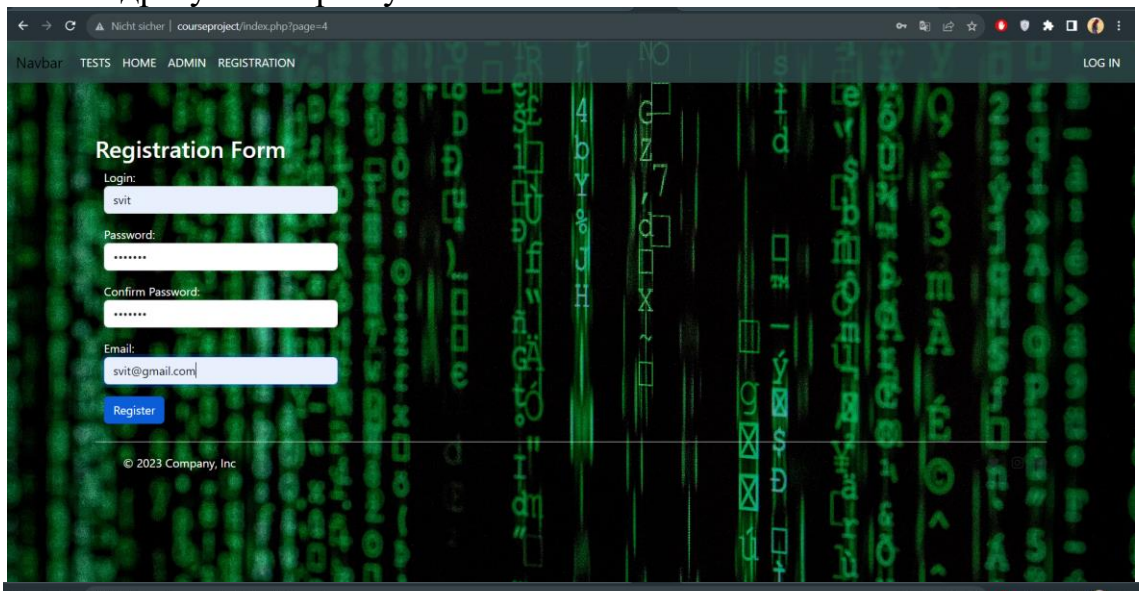
Ці функції реалізують основну функціональність проекту, спрощуючи взаємодію з базою даних, опрацювання даних і управління користувачами та тестуванням.

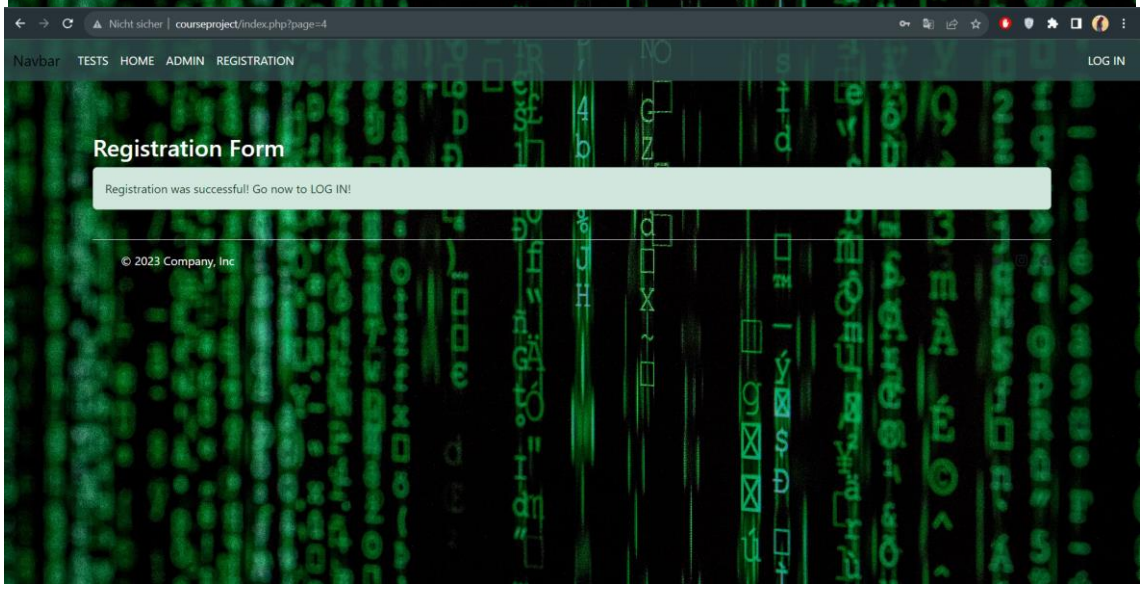
4 Інструкція користування для користувача

REGISTRATION:

1. Відкрийте веб-сторінку проекту.
2. Натисніть на посилання “**REGISTRATION”**.
3. Заповніть обов'язкові поля: логін, пароль та електронну пошту.
4. Перевірте відповідність пароля вимогам безпеки.
5. Натисніть кнопку "Зареєструватися".
6. Якщо реєстрація пройшла успішно, ви будете перенаправлені на сторінку входу.

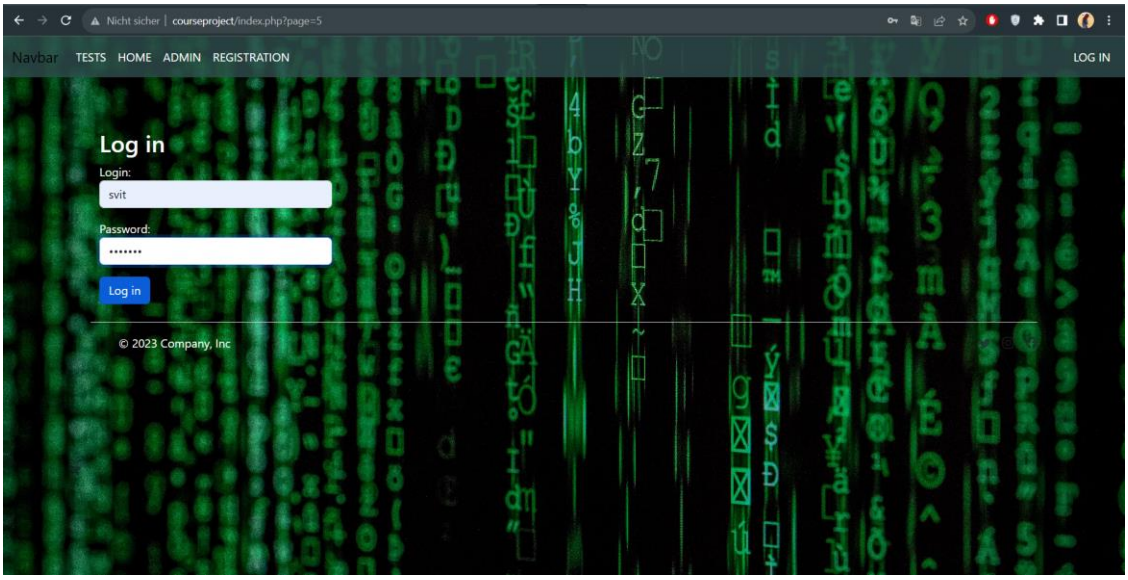
Якщо реєстрація була виконана раніше, то цей крок можна пропустити, зайшовши одразу на сторінку “**LOG IN**”

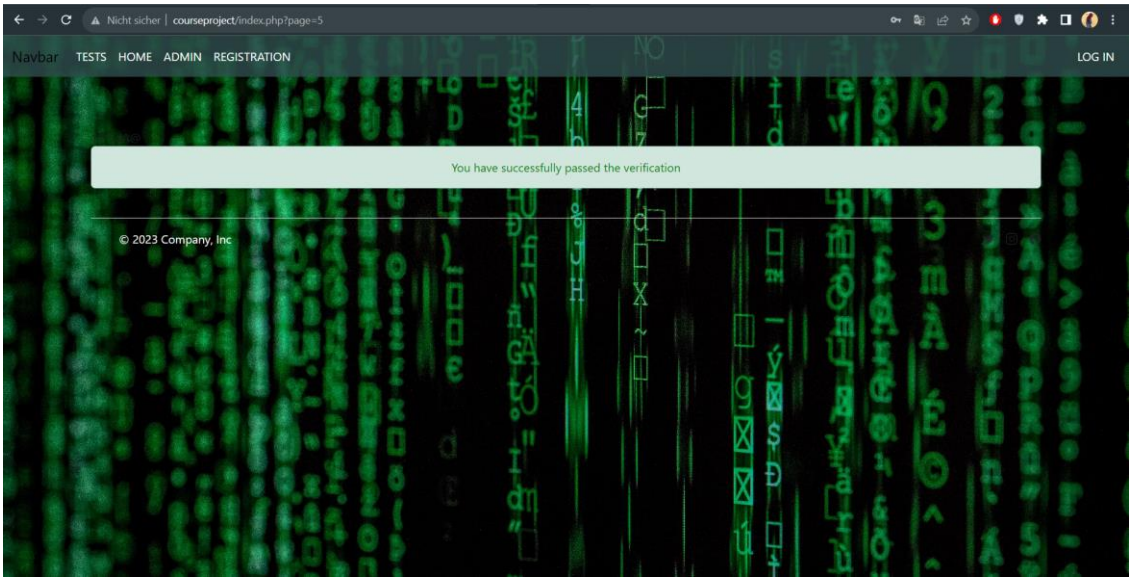




LOG IN:

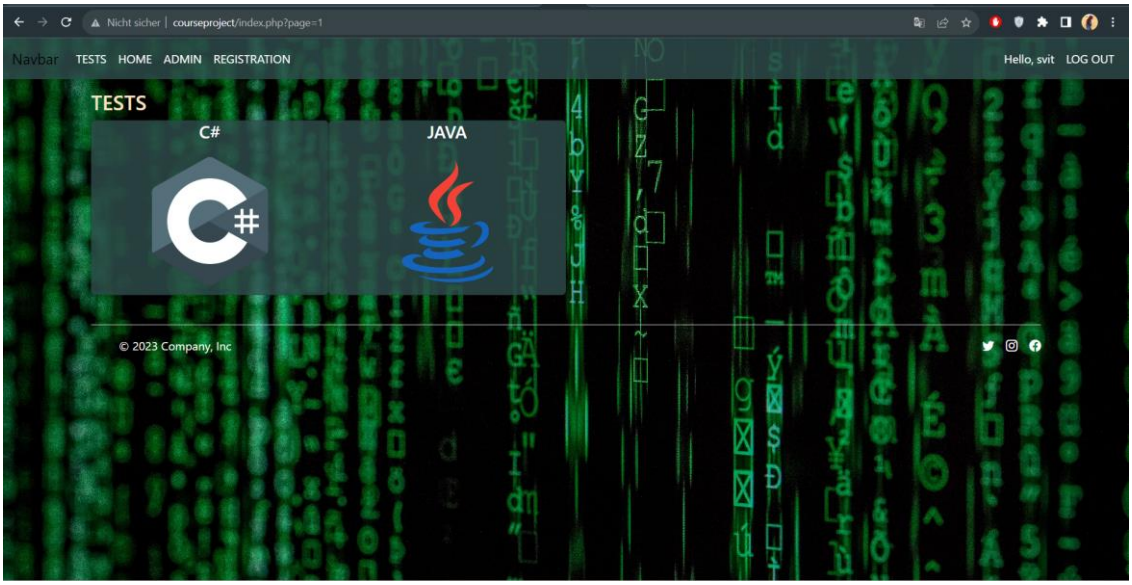
1. Відкрийте веб-сторінку проекту.
2. Введіть свій логін та пароль.
3. Натисніть кнопку “Log in”.





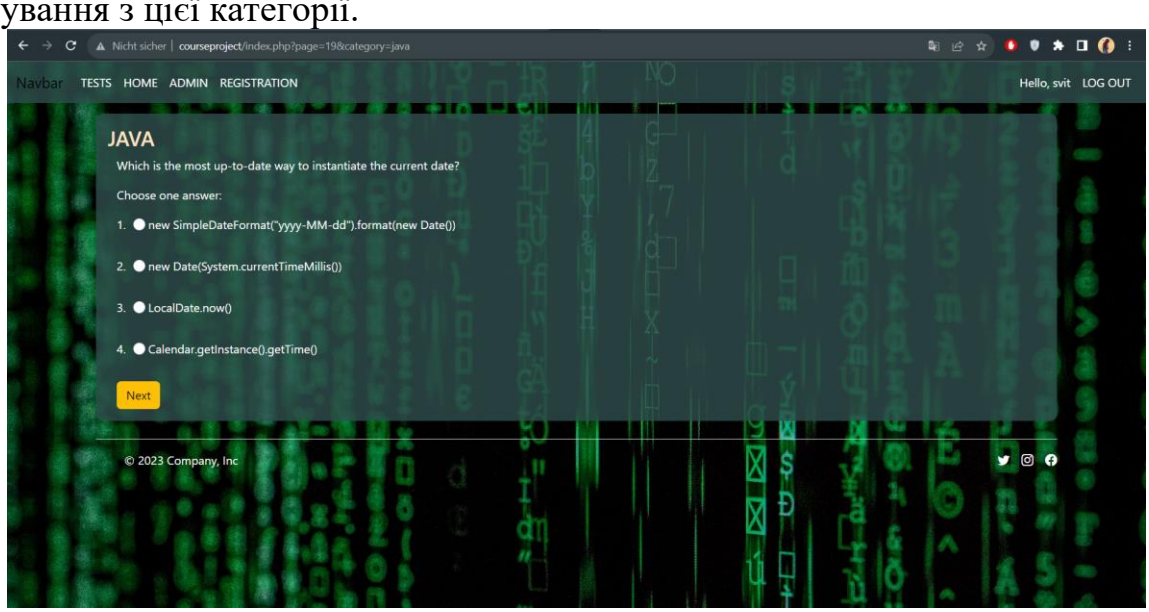
TESTS:

1. Після входу ви потрапите на головну сторінку “**TESTS**”.
2. Тут ви зможете бачити доступні категорії та вибрати ту, яка вас цікавить.

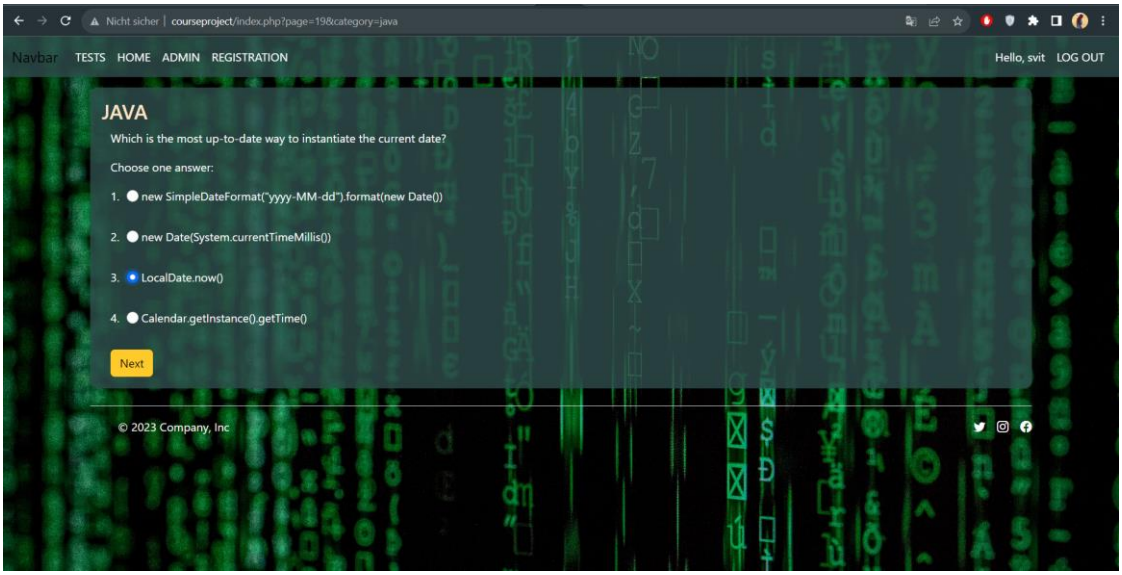


Тестування:

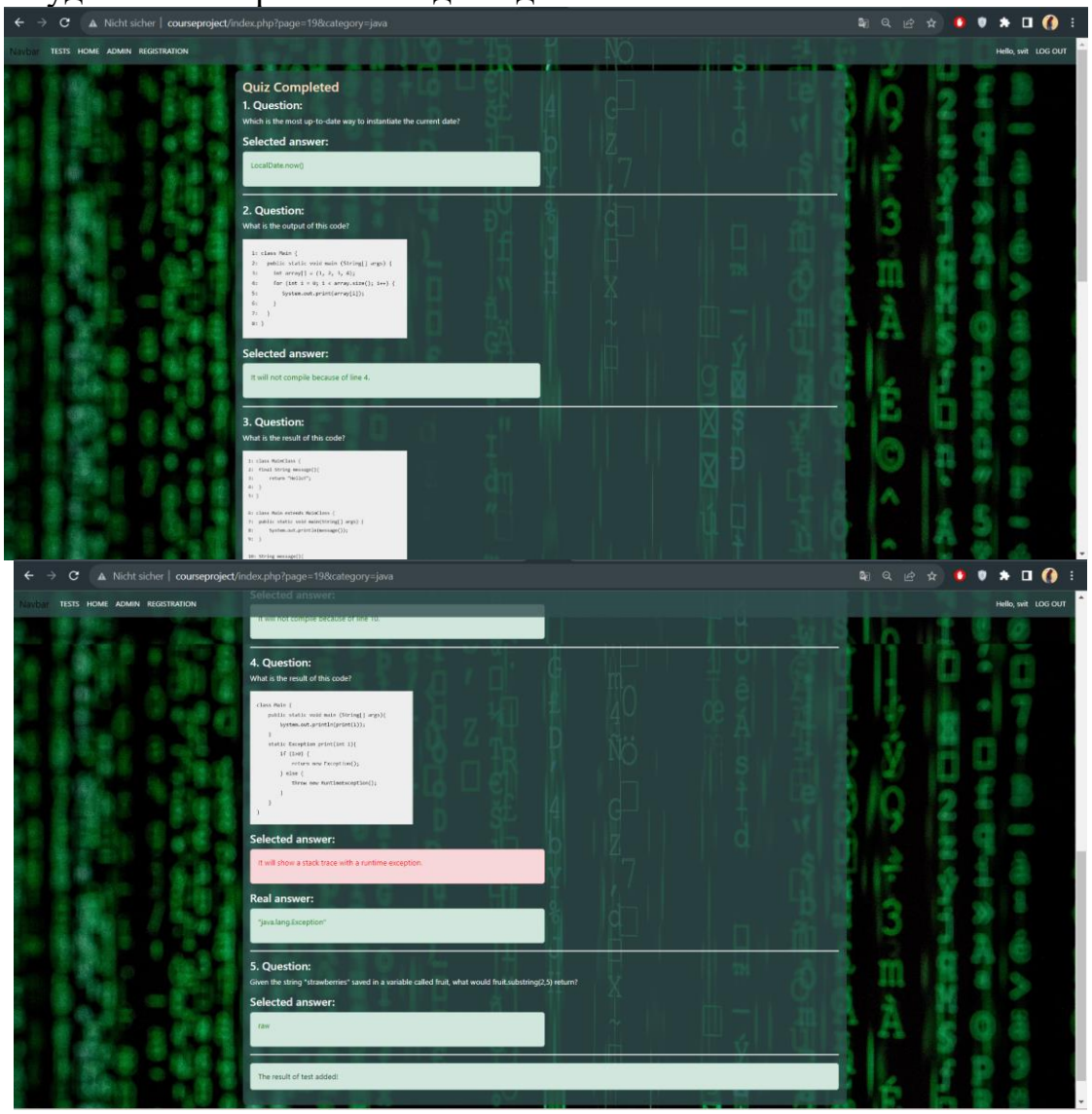
1. Після вибору категорії ви будете перенаправленні на сторінку тестування з цієї категорії.



2. Вам буде запропоновано відповіді на питання, і ви повинні вибрати правильну.

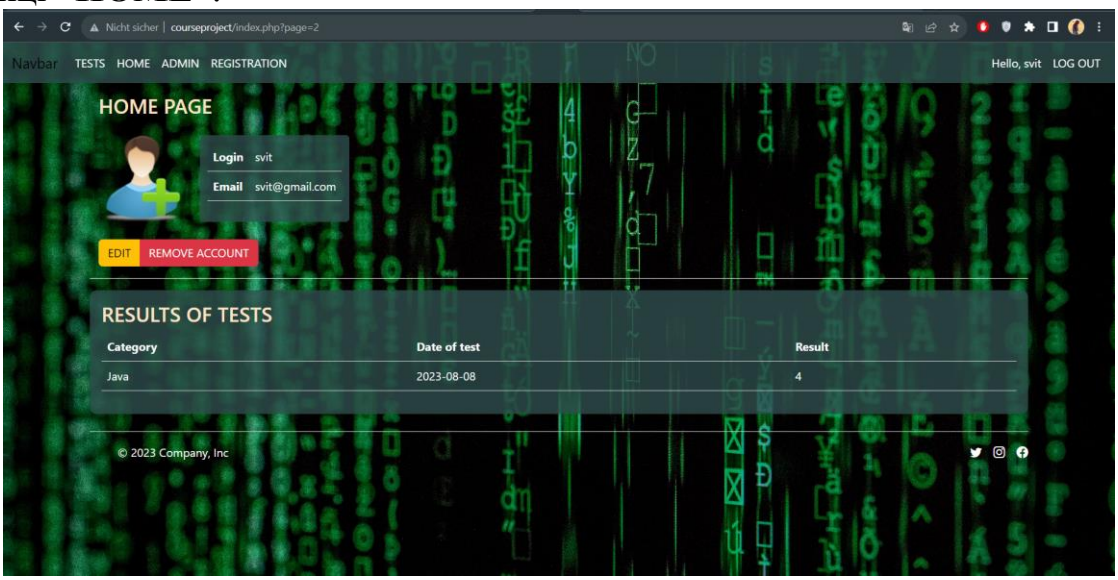


3. По закінченні тестування вам буде показано ваш результат у вигляді запитання і відповіді, якщо відповідь не вірна, то вона буде червоного кольору і нижче буде вказано правильна відповідь.



Результати тестування:

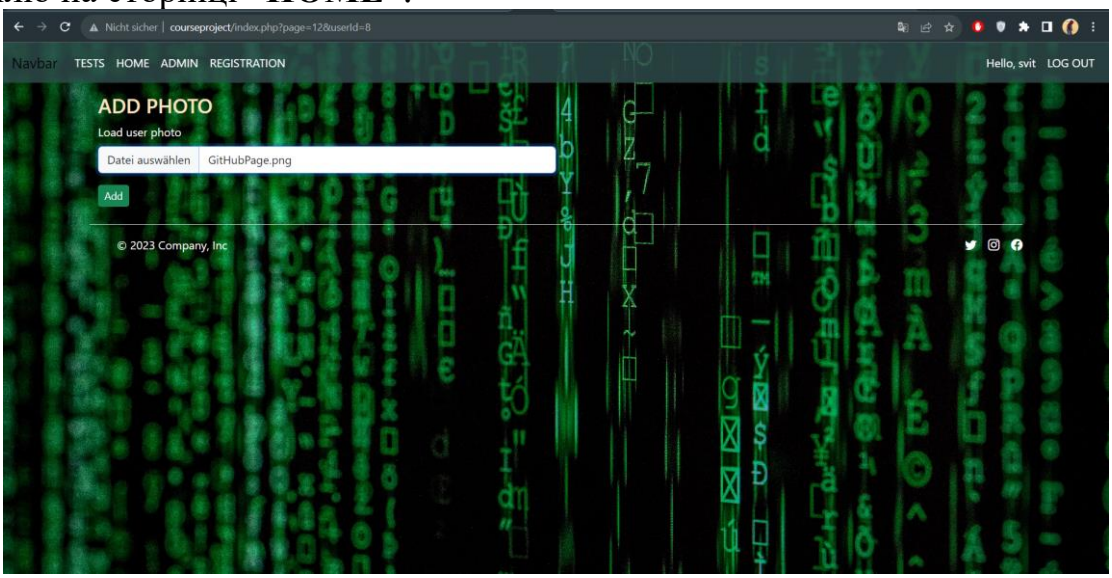
1. Ви можете переглянути свої результати тестувань на спеціальній сторінці **“HOME”**.

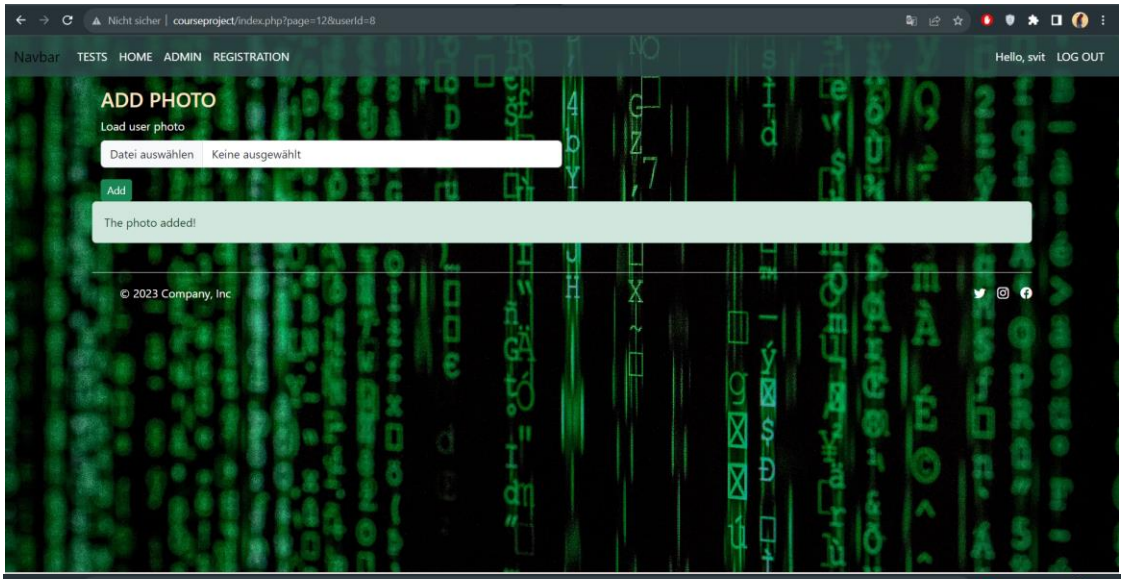


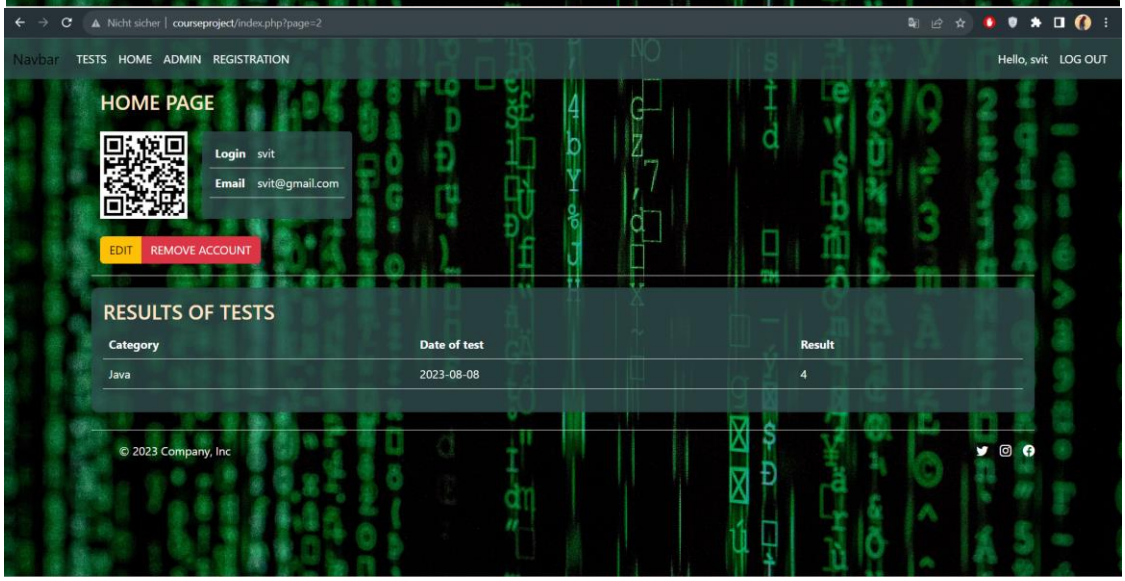
2. Тут ви знайдете інформацію про ваші попередні тести з різних категорій.

Редагування профілю:

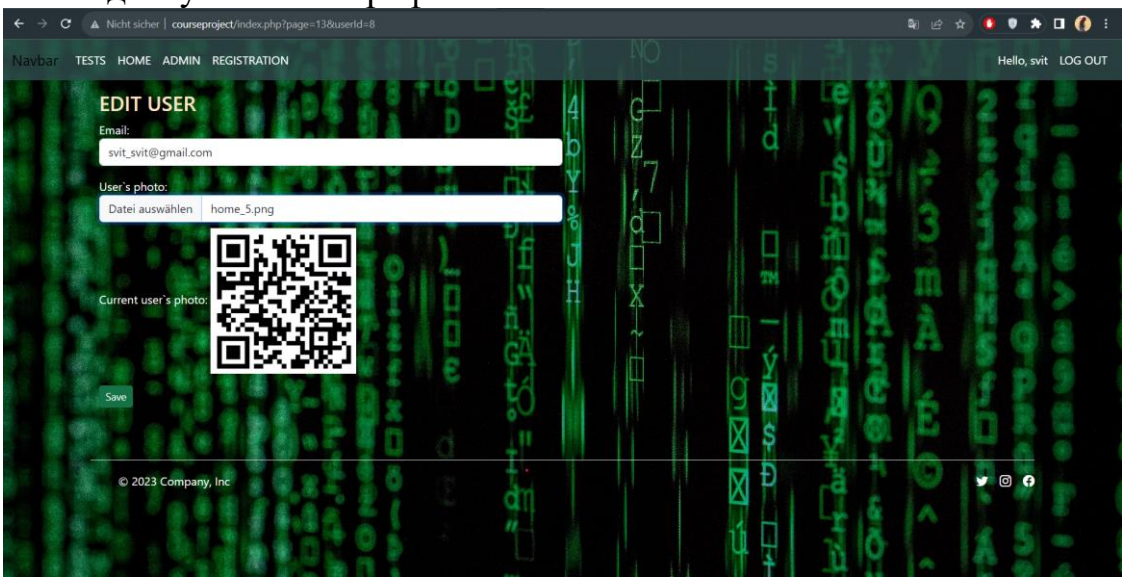
1. У вас є можливість змінити свій електронний лист та фотографію профілю на сторінці **“HOME”**.

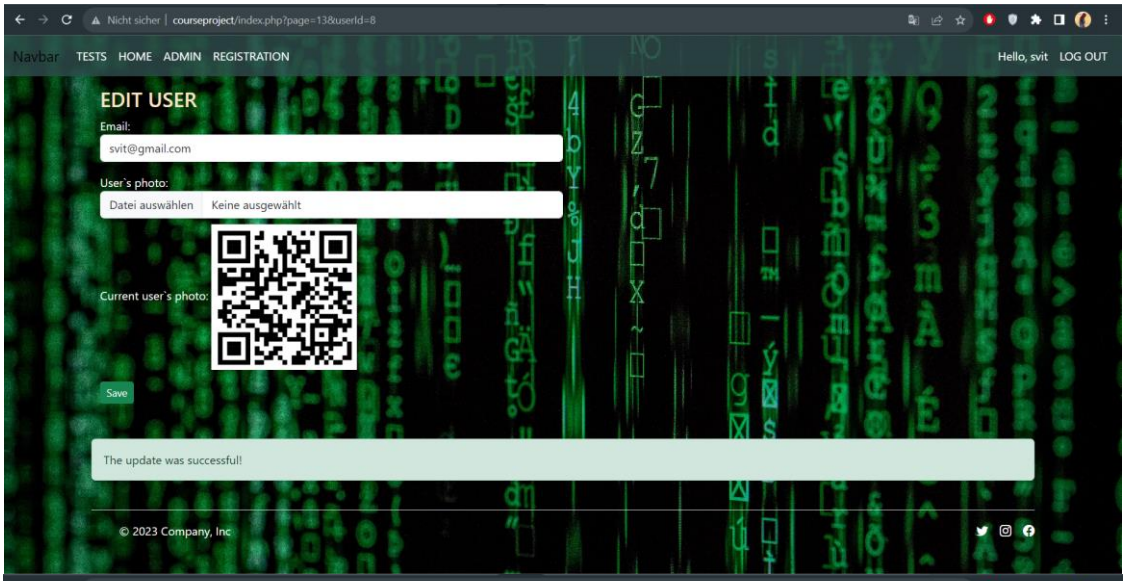


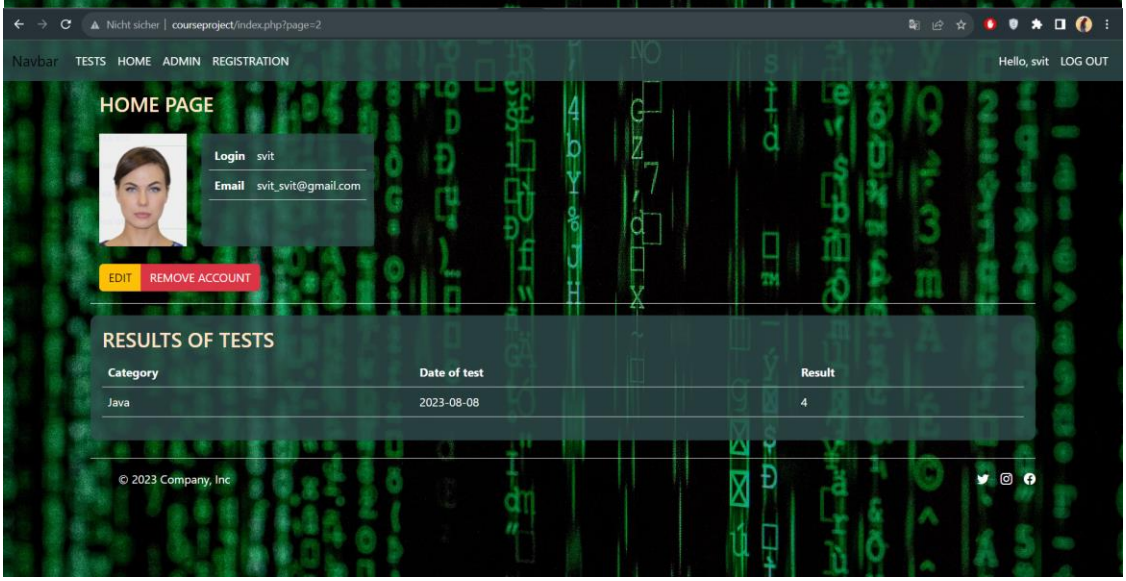




2. Зайдіть у власний профіль та виконайте зміни.

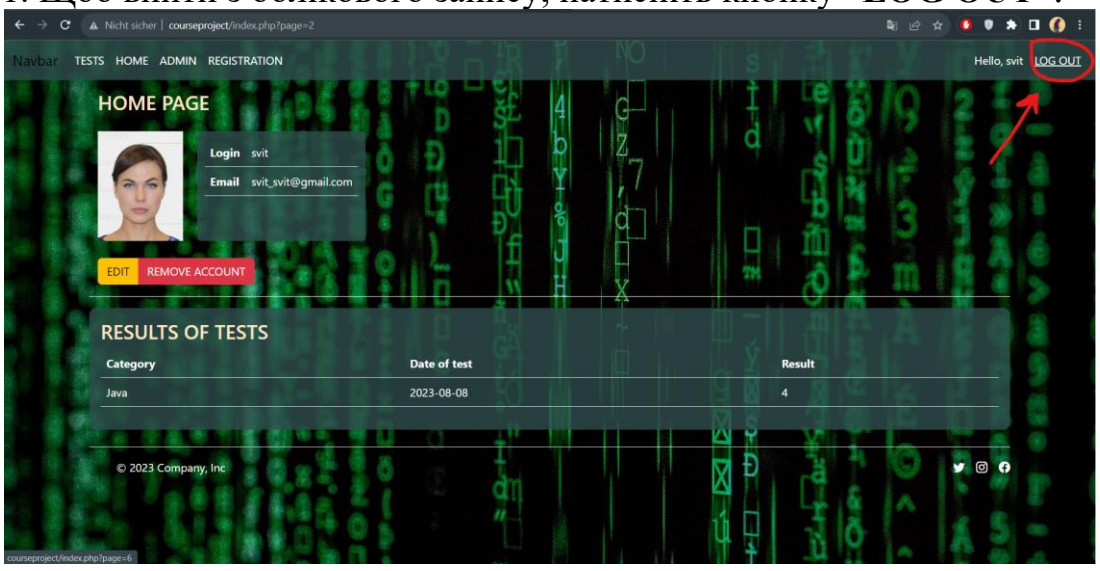






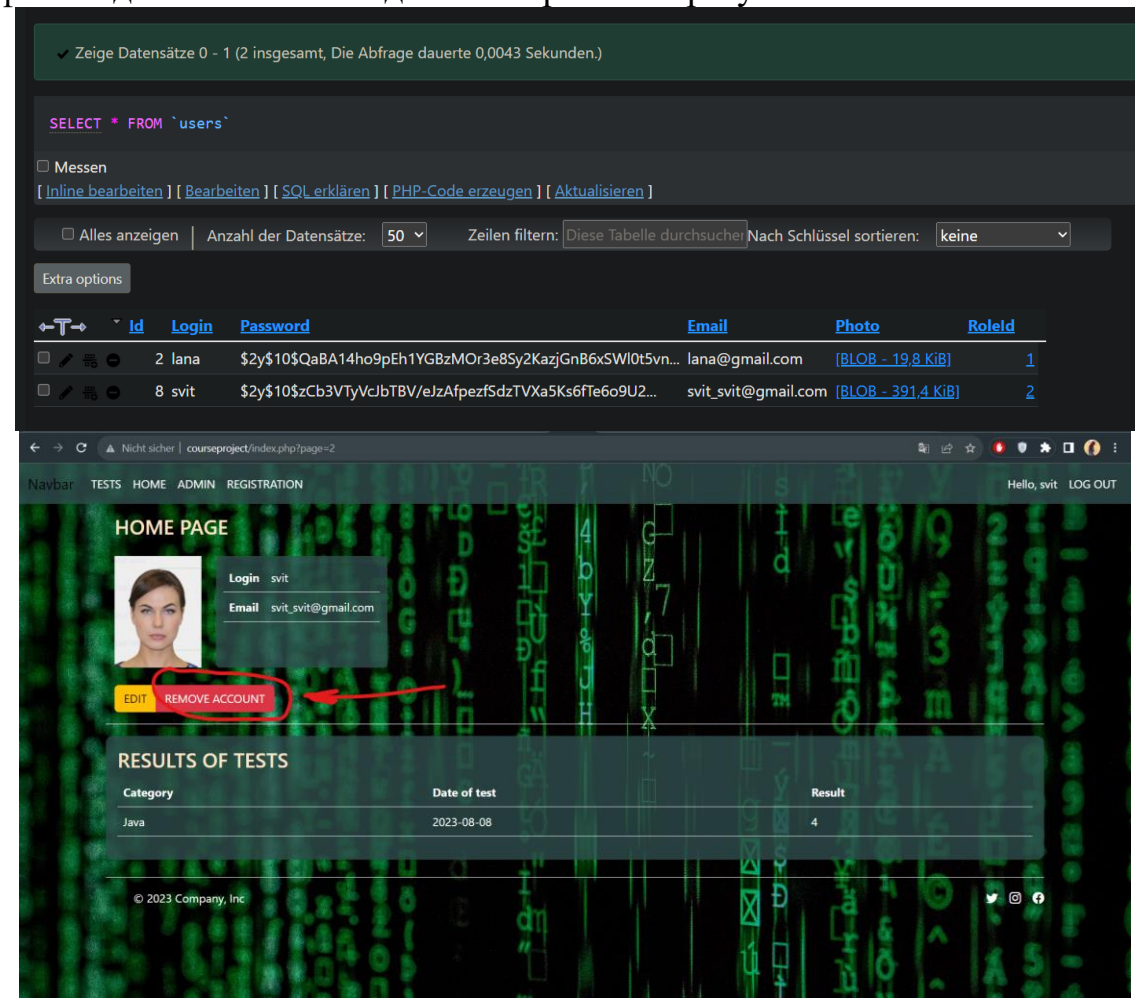
LOG OUT:

1. Щоб вийти з облікового запису, натисніть кнопку “**LOG OUT**”.

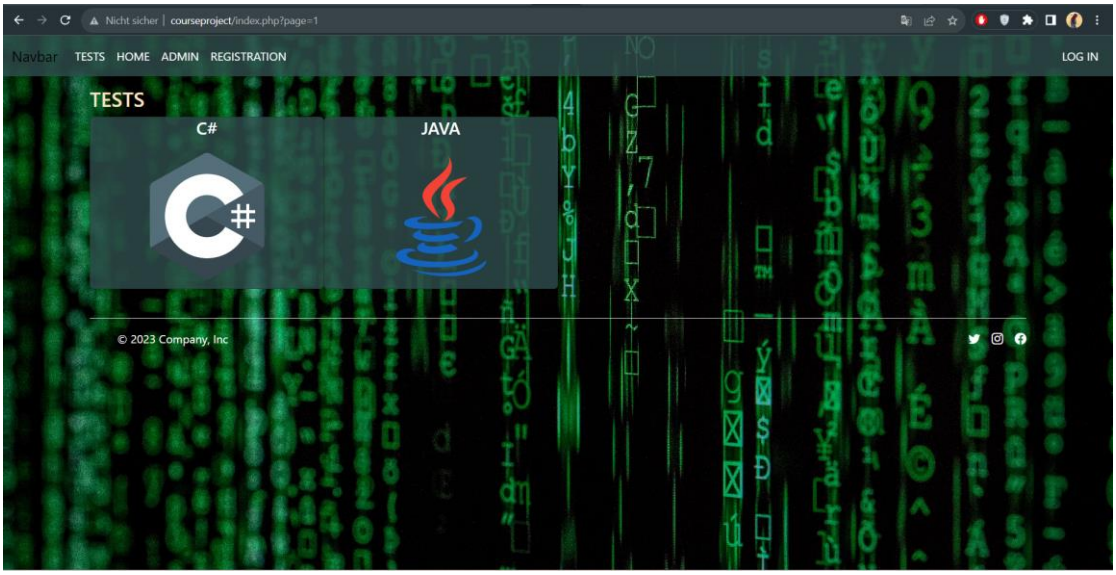


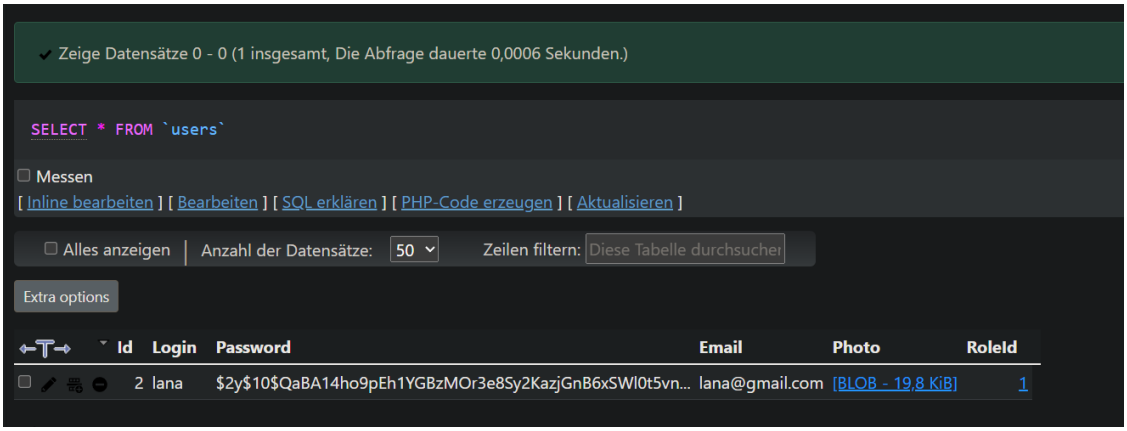
REMOVE ACCOUNT:

1. Щоб видалити аккаунт необхідно натиснути “REMOVE ACCOUNT”. Профіль видалиться на завжди без збереження результатів.



2. Після видалення профіля всі сесії пов’язані з ним знищуються також і користувач переходить на головну сторінку вже як не зареєстрований в даній програмі.





Це загальна інструкція користування вашим проектом. Будь ласка, звертайте увагу на специфічні особливості і можливі функції, які можуть бути додані на вашому сайті.

5 Плани на майбутнє

Цей проект вже маємо декілька чудових функцій, але я прагну зробити його ще кращим і зручнішим для всіх користувачів. Ось кілька ідей, які я маю на увазі:

- **Пошук у результатах для адміністратора:** Додання можливості пошуку в результатах тестувань для адміністратора дозволить швидко знаходити і аналізувати інформацію за категоріями, логінами користувачів та датами.
- **Фільтрація результатів для користувача:** Додаткові опції фільтрації для користувачів, такі як можливість відфільтровувати результати за датами, категоріями та кількістю балів, дозволять їм краще аналізувати свої досягнення.
- **Фільтрація таблиці користувачів для адміністратора:** Додання опції фільтрації таблиці користувачів за логінами та адресами зробить адміністрування більш зручним.
- **Фільтрація таблиці питань за категоріями:** Дозвіл на фільтрацію таблиці питань за категоріями допоможе адміністраторам знаходити та управляти питаннями в різних категоріях.
- **Фільтрація таблиці відповідей за категоріями та правильністю:** Ця можливість допоможе адміністраторам швидко знаходити відповіді за категоріями та виділяти правильні відповіді.
- **Розширена інформація про категорії:** Додавання сторінки з додатковою інформацією по категоріям, такою як порівняння визначень, опис елементів та посилання на інші ресурси, розширить знання користувачів про тематику кожної категорії.

ВИСНОВОК

У підсумку, проект "Тестовий веб-портал для проведення веб-тестування" є відмінною платформою для проведення тестувань та аналізу знань. Завдяки широкому спектру функцій, користувачі мають можливість реєструватися, виконувати тести в різних категоріях та переглядати свої результати. Реєстраційний процес забезпечує безпеку та конфіденційність особистої інформації користувачів завдяки хешуванню паролів та перевірці валідності.

Функція вибору категорій дозволяє користувачам вибирати область, в якій вони хочуть протестувати свої знання. Генерація питань випадковим чином забезпечує різноманітність тесту та унеможливлює підготовку до відповідей. Після завершення тесту користувачі можуть переглянути результати та зрозуміти свій рівень знань у вибраній категорії.

Адміністратори мають повний контроль над системою, вони можуть додавати та редагувати питання, відповіді, категорії, ролі користувачів та багато іншого. Можливість блокування категорій та вибір реальних відповідей додає гнучкості та точності до тестів.

Плани на майбутнє передбачають покращення пошуку та фільтрації для результатів, забезпечуючи зручний аналіз для адміністраторів та користувачів. Додаткова інформація про категорії розширить можливості вивчення матеріалу. Враховуючи зворотній зв'язок користувачів та їхні пропозиції, я прагну зробити проект найкращим інструментом для тестування та навчання.

Загалом, проект "Тестовий веб-портал для проведення веб-тестування " ставить перед собою мету створити зручний та ефективний інструмент для тестування знань, що задовольняє потреби як користувачів, так і адміністраторів.