МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра кібербезпеки

«До захисту допущено»
Завідувач кафедри
_____ Володимир ЛЮБЧАК
(підпис)
«__»_____ 20__ р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня бакалавр

зі спеціальності 125 Кібербезпека та захист інформації, освітньо-професійної програми Кібербезпека
на тему: «Розробка інформаційної системи для оцінки рівня кібербезпеки серед робочого персоналу»

Здобувача групи КБ-01 Бреусенка Віталія Юрійовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і тестів інших авторів мають посилання на відповідне джерело.

_____ Бреусенко В. Ю.
(підпис)

Керівник _____ _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра кібербезпеки

«Затверджую»
Завідувач кафедри

_____ Володимир ЛЮБЧАК
(підпис)

«___»_____ 20___ р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
зі спеціальності 125 – Кібербезпека, освітньо-професійної програми «Кібербезпека»
здобувача групи КБ-01 Бреусенка Віталія Юрійовича

1. Тема роботи: «Розробка інформаційної системи для оцінки рівня кібербезпеки серед робочого персоналу».

затверджено наказом по СумДУ №0212-VI від « 04 »_____03_____2024 р.

2. Термін подання студентом роботи: «____» _____ 20 ___ р.

3. Вихідні дані до роботи: _____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити): ___
_____
_____
_____

5. Перелік графічного матеріалу (із зазначенням плакатів, презентацій тощо) _____

6. Консультанти до проекту (роботи), із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	Оцінка рівня кібербезпеки на підприємствах		
2	Огляд та підбір інструментарію для оцінки рівня кібербезпеки		
3	Програмна реалізація системи E-PHISHING		

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Кваліфікаційна робота виконана на 47 аркушах та містить 64 рисунки, 1 додаток та 13 джерел.

Об'єкт дослідження: системи для оцінки рівня кібербезпеки серед робочого персоналу.

Мета роботи: розробка інформаційної системи для оцінки рівня кібербезпеки серед робочого персоналу за допомогою популярної мови програмування, мови розмітки та каскадних стилів.

Методи дослідження: сучасні інструменти для оцінки рівня кібербезпеки, впровадження системи для оцінки рівня кібербезпеки з використанням інструментів для керування та зберіганням даних.

Результати роботи:

Розглянуто сучасні інструменти для оцінки рівня кібербезпеки.

Розроблено та реалізовано систему для оцінки рівня кібербезпеки серед робочого персоналу. Система дозволяє: створювати «фішингові» посилання, відправляти їх, фіксувати переходи по посиланню та отримані дані, формувати загальну статистику.

Результати кваліфікаційної роботи обговорювалися на міжнародній конференції IMA-2024.

Ключові слова: інформаційна система, оцінка рівня кібербезпеки серед робочого персоналу, розробка системи, E-PHSHING.

ЗМІСТ

ВСТУП

Метою кваліфікаційної роботи є розробка інформаційної системи E-PHISHING для оцінки рівня кібербезпеки серед робочого персоналу. Доступ до даної системи можна отримати через веб-інтерфейс.

E-PHISHING – це універсальний інструмент для створення та відправки «фішингових» посилань. Також він зберігає:

1. створене посилання;
2. факт відправки повідомлення та адресата;
3. всі переходи по даному посиланню (час переходу, IP-адреса);
4. отримані дані (час відправки даних, IP-адреса, логін:пароль).

Система E-PHISHING веде статистику по всім вищевказаним пунктам та фіксує скільки всього було операцій зі вказаного акаунту відображаючи це у вигляді карток з картинками (детальніше далі у п. 3.7).

Розробник веб-додатку піклується про користувачів системи, тому було розроблено систему відправки заявок розробнику, а у свій же час адміністратори отримують дані заявки та обробляють їх (статус своєї заявки користувач може бачити з ліва від форми відправки заявок).

Користувач також може залишити свій коментар про сервіс у відповідному розділі «Відгуки».

1 ОЦІНКА РІВНЯ КІБЕРБЕЗПЕКИ НА ПІДПРИЄМСТВАХ

Оцінка рівня кібербезпеки на підприємствах серед робочого персоналу є важливою складовою безпеки. Для проведення оцінки використовують різні методи та підходи, які допоможуть виявити слабкі місця та підвищити обізнаність співробітників.

Було переглянуто і дібрано кілька таких ключових методів:

1. Оцінка знань – використання анкет та різного роду опитувань для визначення рівня обізнаності робітників. Щоб перевірити знання про основні принципи кібербезпеки, політик компанії та актуальних загроз. Також необхідним моментом є проведення регулярних тестів для перевірки знань в які входять як і теоретичні питання, так і практичні завдання.

2. Навчання – підприємство має проводити регулярні навчальні тренінги для робітників, щоб підвищити їхню обізнаність у сфері кібербезпеки. Підприємство також може проводити фішингові атаки для перевірки готовності працівників реагувати на реальні загрози, що саме і буде реалізовано в кваліфікаційній роботі.

3. Аналіз поведінки – перегляд атак на підприємство, порушення безпеки для того, щоб визначити чи були вони спричинені робітником та чи у випадку зіграла роль людського фактору чи порушення безпеки було навмисне. Також підприємство має використовувати необхідні інструменти для відстеження поведінки користувачів.

4. Політики – компанія має переглядати існуючі політики кібербезпеки для забезпечення їх актуальності та відповідності сучасним нормам і загрозам. Якщо ж ні, то компанія має розробити нові політики та рекомендації на основі результатів оцінки знань і аналізу поведінки робітників.

5. Технологічні заходи – підприємство повинно використовувати сучасні інструменти захисту, це антивіруси, фаєрволи, системи виявлення вторгнень для зниження ризиків кіберзагроз. Повинні бути впроваджені правила контролю доступу до критичних даних, та управління правами доступу.

Для ефективної оцінки рівня кібербезпеки на підприємствах потрібно поєднувати вищевказані методи. Постійні навчання співробітників є одним з кращих технологій, але потрібно звернути увагу на проводження фішингових атак на робітників з метою встановлення готовності працівників реагувати на реальні загрози. Саме це розглянемо і програмно реалізуємо, створимо систему, яка практично зможе перевіряти робітників.

2 ОГЛЯД ТА ПІДБІР ІСТРУМЕНТАРІЮ ДЛЯ ОЦІНКИ РІВНЯ КІБЕРБЕЗПЕКИ

Спочатку розглянемо готовий інструментарій для перевірки робітників. Переглянувши інструментарій в мережі інтернет, можна навести 5 популярних програмних рішень:

1. KnowBe4 – інструмент для проведення інтерактивних тренінгів з кібербезпеки та фішингової симуляції з метою розпізнання та уникання фішингових атак [9].
2. PhishMe – платформа для навчанні працівників через симуляції фішингових атак [10].
3. Wombat Security – платформа, яка містить в собі інтерактивні модулі навчання з кібербезпеки та оцінки ризиків [11].
4. Terranova Security – це інтегровані навчальні платформи з кібербезпеки [12].
5. Okta – програмний інструмент для управління ідентичностями та доступом, щоб виявляти ідентичності сесій робітників [13].

Програмні інструменти підбираються після проведення анкетування робітників, отримавши їх результат, можна підібрати правильний програмний продукт. Якщо створити свій інструмент та інтегрувати у систему підприємства, то можливо отримати кращий результат та в любий час робити перевірки на підприємстві, зберігати результати для формування тренувань.

Тепер створимо свою систему. Для цього нам потрібно підібрати власні інструменти, за допомогою яких ми програмно реалізуємо систему E-PHISHING (п. 2.1-2.3).

2.1 ПІДБІР СЕРВЕРУ

OpenServer – локальний веб-сервер для розробки та тестування веб-додатків на персональному комп'ютері. Він створений для розробників, яким потрібно швидко налаштувати середовище розробки без необхідності розгортання справжнього веб-сервера [7].

Основні компоненти OpenServer:

1) Apache/Nginx – популярні веб-сервери.
2) PHP – серверна мова програмування.
3) MySQL/PostgreSQL – системи управління базами даних.
4) FTP-сервер – для роботи з файлами через FTP.
5) Email-сервер – для тестування відправки поштових повідомлень.

Саме для нашого проєкту ми будемо використовувати такі компоненти OpenServer як Apache 2.4, PHP 8.0, MySQL 8.0 та спосіб збереження поштових листів у файл (поки що ми не маємо поштового серверу) (рис. 2.1.1).

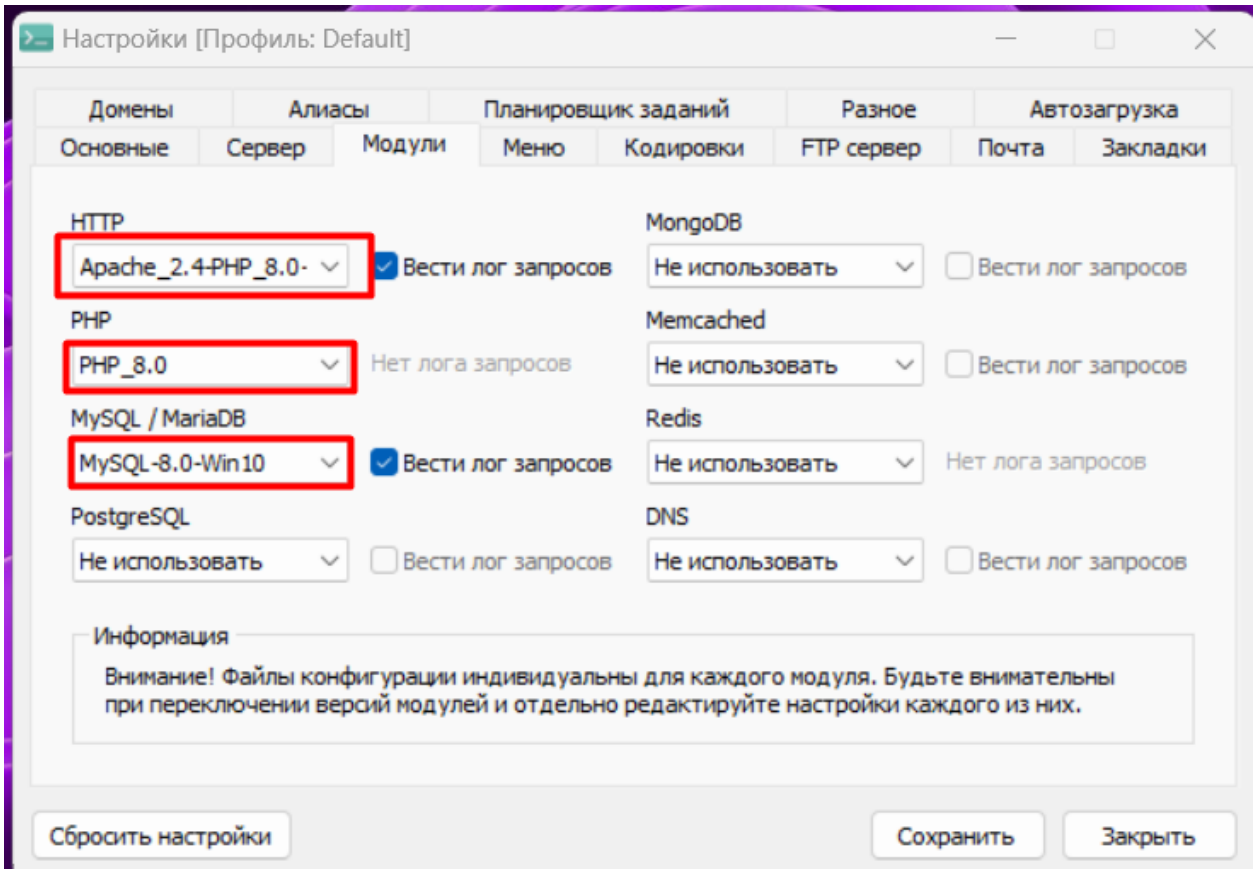


Рисунок 2.1.1 – Компоненти OpenServer

2.2 ПІДБІР МОВ ПРОГРАМУВАННЯ

У даному проєкті використаємо мову розмітки HTML 5, каскадні стилі CSS 3 та фреймворк Bootstrap, мову програмування PHP 8.0.

HTML — мова розмітки, яка використовується для створення та структурування веб-сторінок. HTML дозволяє визначати різні елементи веб-сторінки, такі як заголовки, абзаци, посилання, зображення, таблиці, форми та інші [2].

CSS — мова стилів, яка використовується для опису вигляду і форматування документа, написаного на мові розмітки, такої як HTML. CSS 3 є останньою основною версією CSS, яка додає нові функції та можливості [3].

Bootstrap — фреймворк для розробки інтерфейсів веб-застосунків і веб-сайтів, який використовує CSS, JavaScript і HTML. Він розроблений для полегшення створення адаптивних, мобільно-дружніх веб-сторінок [3].

PHP — мова програмування, яка використовується переважно для розробки веб-додатків і динамічних веб-сторінок. PHP виконується на сервері, а результат, який отримує користувач, передається у вигляді HTML [1].

За допомогою HTML робимо верстку сайту, стилізуємо за допомогою CSS та фреймворком Bootstrap. Та сама головна мова програмування PHP – збирає динамічно сторінку для користувача, працює з базою даних, використовує сесії для ідентифікації користувача та відображає результати обробки даних для користувача.

2.3 ВИБІР БАЗИ ДАНИХ

MySQL — це популярна система управління реляційними базами даних, яка використовується для зберігання та управління даними в багатьох веб-додатках. MySQL є відкритим програмним забезпеченням і підтримує стандартну мову запитів SQL [4].

Основні можливості MySQL:

1) Використання мови SQL для управління даними (вибірка, вставка, оновлення, видалення).
2) Підтримка індексів для пошуку даних.
3) Підтримка ключів для встановлення зв'язків між таблицями.
4) Управління доступом і контроль над правами доступу для користувачів.
5) Підтримка реплікації для розподілу даних між декількома серверами.
6) Оптимізація для обробки великої кількості запитів і даних.

У проєкті E-PHISHING використовується MySQL разом з PHP. PHP надає розширення, такі як mysqli (MySQL Improved) та PDO (PHP Data Objects), для роботи з MySQL. E-PHISHING же під’єднується до бази даних за допомогою mysqli (далі в п. 4 підключимось до бази).

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ E-PHISHING

3.1 НАЛАШТУВАННЯ БАЗИ ДАНИХ

Оскільки база даних уже створена, то перейдемо до phpMyAdmin та авторизуємось для створення необхідних таблиць у базі E-PHISHING (рис. 3.1.1).

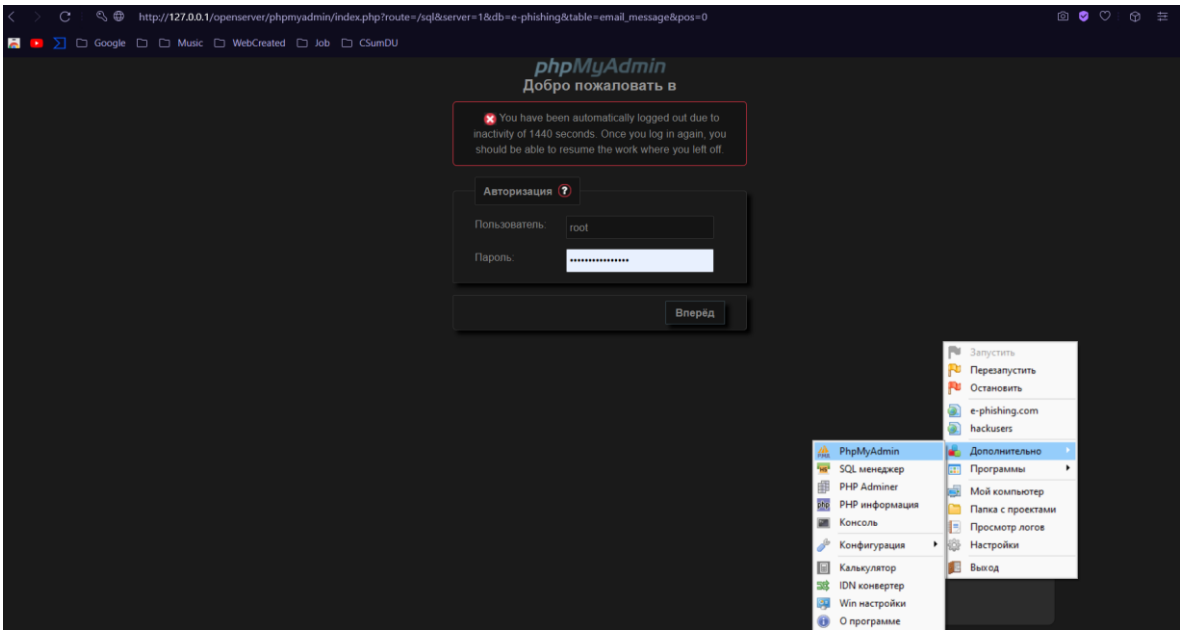


Рисунок 3.1.1 – Перехід та авторизація

Входимо в систему та створюємо табличку email_message та message_type у яких будуть дані про відправлені повідомлення та шаблони повідомлень (рис. 3.1.2-3.1.4). Табличка з користувачами вже існує (рис. 3.1.5).

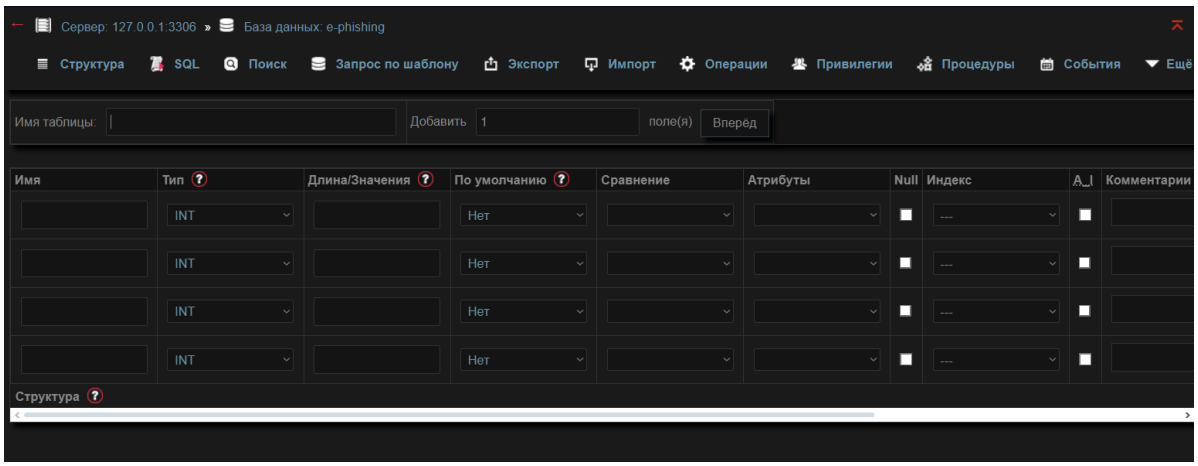


Рисунок 3.1.2 – Заповнюємо необхідні поля

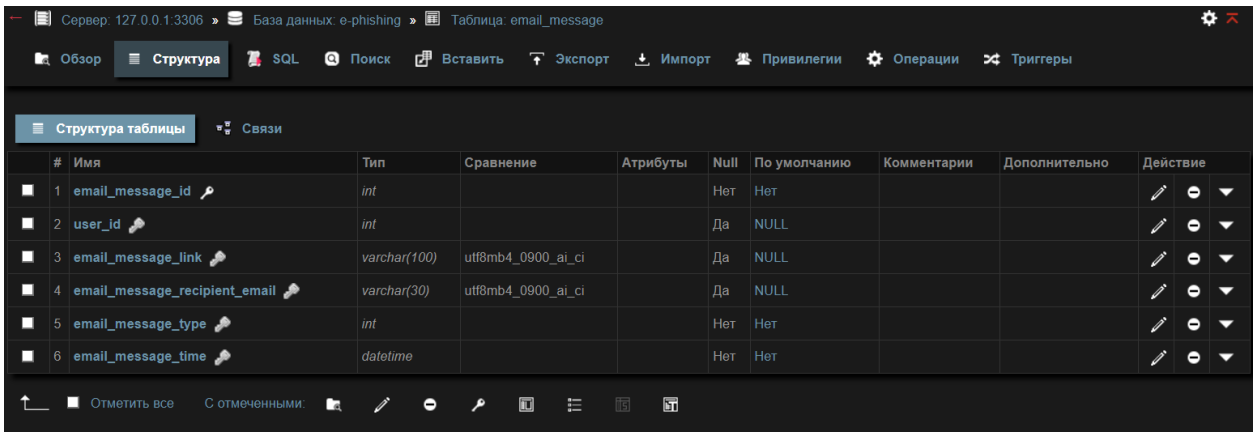


Рисунок 3.1.3 – Отримуємо таблицю email_message

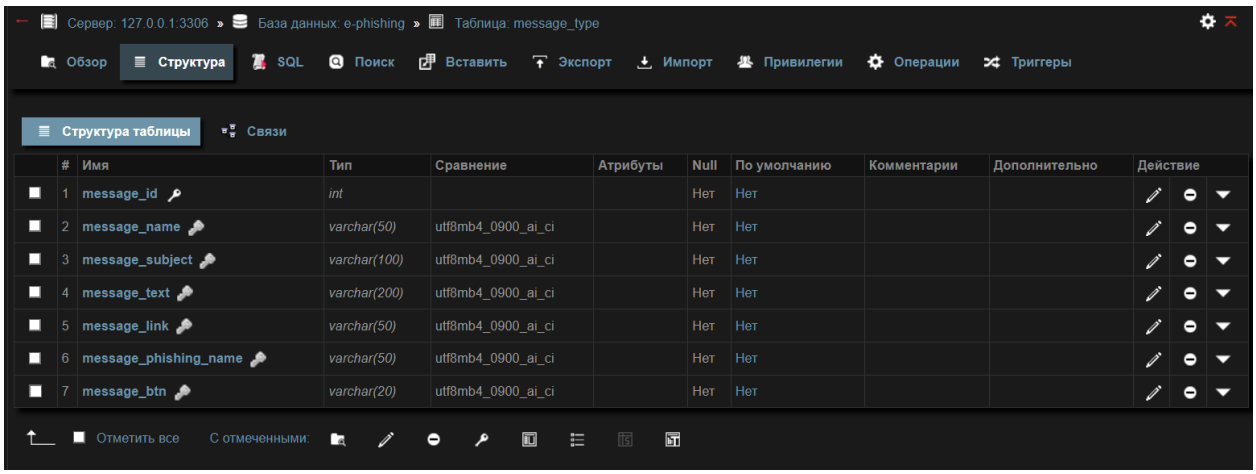


Рисунок 3.1.4 – Отримуємо табличку message_type

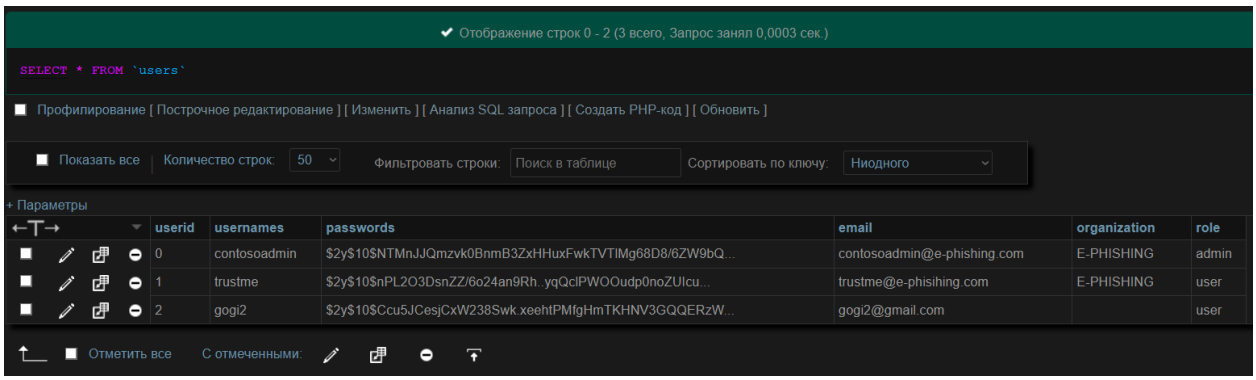


Рисунок 3.1.5 – Табличка users уже існує та заповнена даними

Далі заповнимо табличку message_type тестовими даними (рис. 3.1.6-3.1.7). Табличка email_message заповнюватиметься на основі відправленого листа.

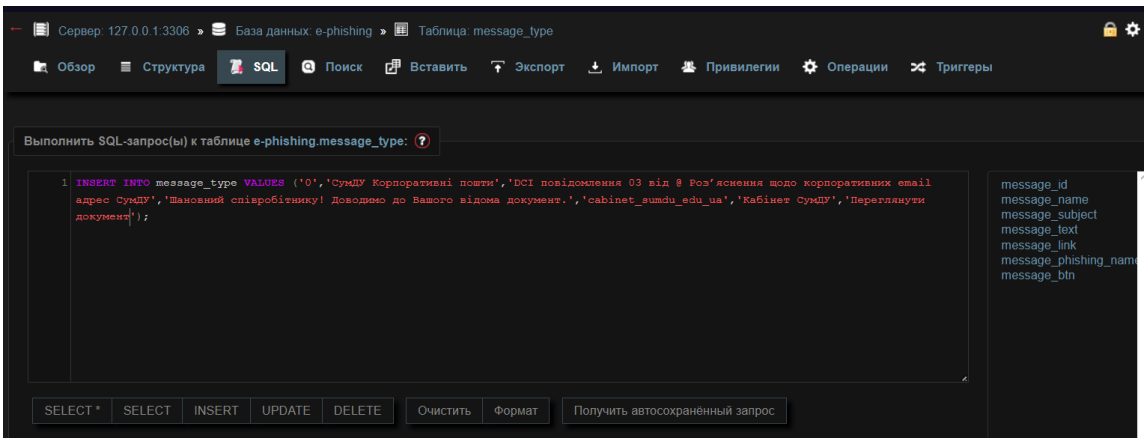


Рисунок 3.1.6 – SQL-код заповнення таблички

За таким принципом рис. 3.1.6 додаймо ще даних.

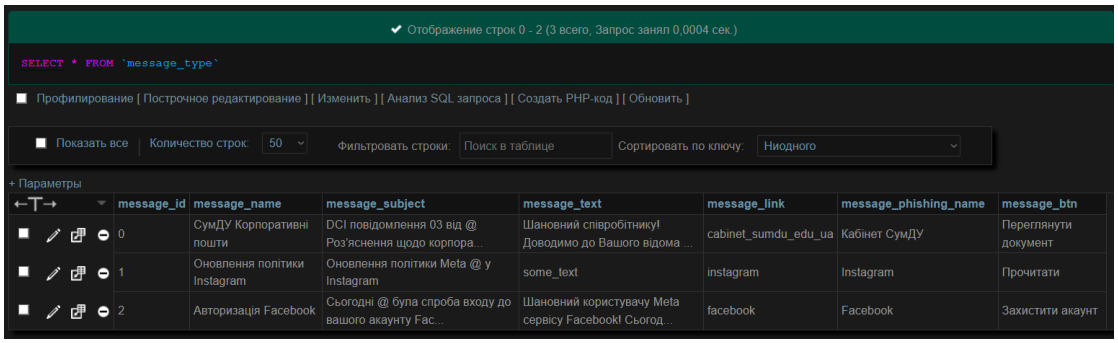


Рисунок 3.1.7 – Заповнена табличка message_type

Далі створимо всі необхідні таблиці та заповнимо ті, які необхідно (рис. 3.1.8).

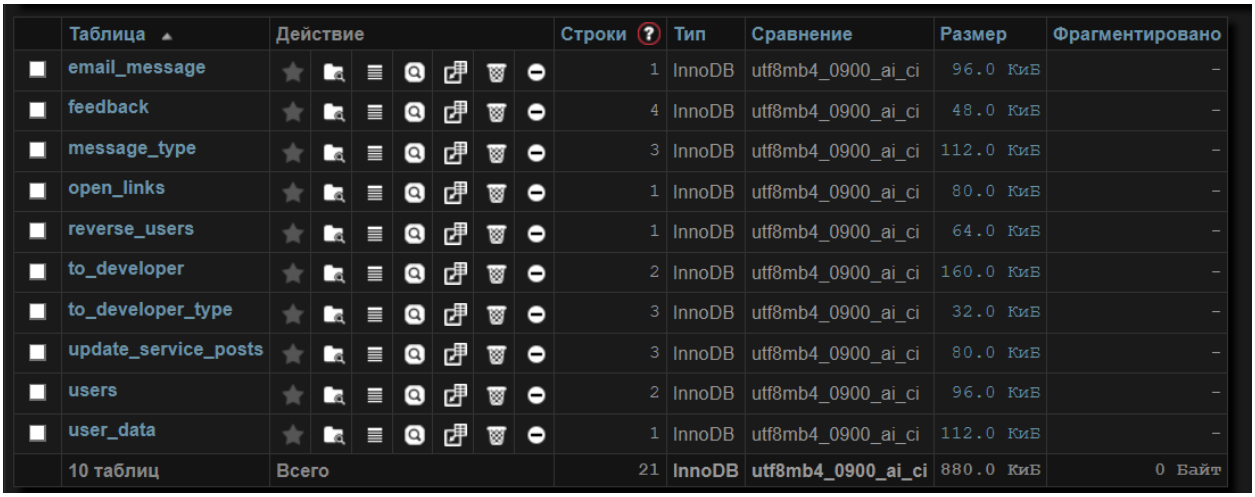


Рисунок 3.1.8 – Створені всі необхідні таблиці

3.2 ПІДКЛЮЧЕННЯ БАЗИ ДАНИХ ДО ВЕБ-САЙТУ

Проведемо підключення до бази даних за допомогою PHP [5]. Створюємо файл connect.php. Далі використовуючи метод $conn (mysqli) підключаємось до бази як показано на картинці нижче (рис. 3.2.1).

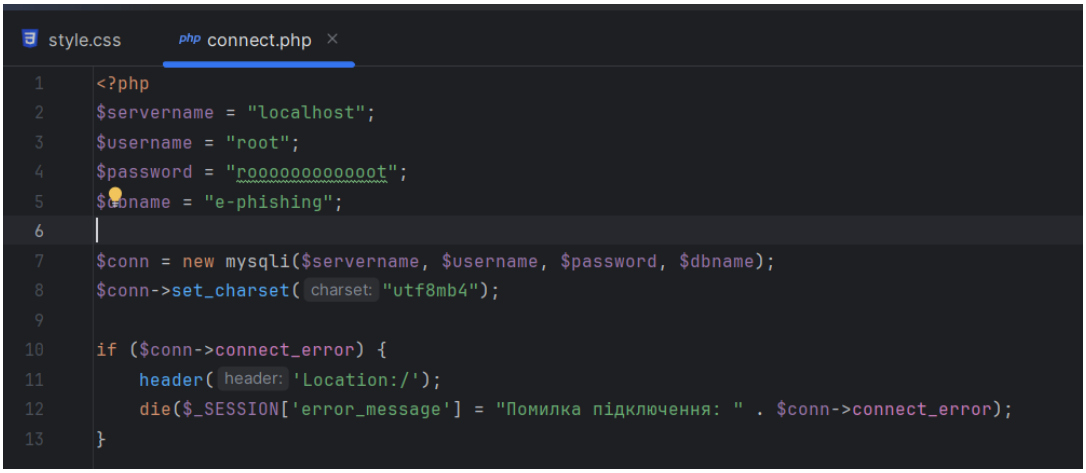


Рисунок 3.2.1 – Підключення бази даних до веб-сторінки

Змінна $servername містить назву серверу, $username – ім’я користувача, $password – пароль користувача, $dbname – назву бази даних. За допомогою змінної $conn та розширення mysqli робимо запит на з’єднання з базою даних за вищевказаними параметрами. Далі наступною командою в рядку 8 встановлюємо набір символів utf8m4, щоб уникнути помилок при виведенні/записі даних. І в кінці перевірка, якщо виникає помилка підключення до бази, то перекинути користувача на головну сторінку та за допомогою сесії вивести помилку підключення.

3.3 РЕЄСТРАЦІЯ ТА АВТОРИЗАЦІЯ В СИСТЕМІ

Реєстрація та авторизація відбуваються на 1 сторінці, за допомогою PHP запитів ми розмежовуємо 2 форми (рис. 3.1.1-3.1.2).

```php
$formPanel = $_GET['page-element'] ?? '';
if ($formPanel === 'authorization') {$heading_h1 = 'Вхід';}
else if ($formPanel === 'registration') {$heading_h1 = 'Реєстрація';}
```

Рисунок 3.1.1 – Отримання атрибуту з рядка веб-браузеру

```php
<div class="login-form">
    <div class="container-fluid">
        <div class="offset-md-4 col-md-4 offset-sm-3 col-sm-6">
            <?php
            if($formPanel === 'authorization') {
                require_once __DIR__ . '/include/forms/login-form.php';
            }else if($formPanel === 'registration'){
                require_once __DIR__ . '/include/forms/registration-form.php';
            }else{
                header( header: 'Location:/login');
            }
            ?>
            <script src="/js/password-visibility.js"></script>
        </div>
    </div>
</div>
```

Рисунок 3.1.2 – Відображення форми на основі атрибуту

Для спрощення посилань та захисту від різних атак, через файл налаштувань серверу створюємо правила редіректу (рис. 3.1.3).

```
RewriteEngine on
Options All -Indexes

RewriteRule ^logout$ /include/authentication/logout.php [L]
RewriteRule ^login$ /login.php?page-element=authorization [L,QSA]
RewriteRule ^registration$ /login.php?page-element=registration [L,QSA]
```

Рисунок 3.1.3 – Правила заміни посилань

Створимо форми авторизації і реєстрації, вони майже ідентичні з посиланнями на різні файли обробки, тому приведемо код однієї форми (рис. 3.1.4).

```
<form class='form-horizontal needs-validation' action='/include/authentication/login.php'
      method='POST' novalidate>
    <div class='form-group'>
        <label>
            <input type='text' name='logins' id="login" class='form-control' autocomplete='on'
                   placeholder='Логін' oninput='checkForm()' required>
        </label>
    </div>
    <div class='form-group'>
        <label>
            <input type='password' name='passwords' id='password' autocomplete='on'
                   class='form-control' placeholder='Пароль' minlength='8' oninput='checkForm()' required>
            <input name='no-bot' type='hidden' value=''>
        </label>
    </div>
    <div class='form-group'>
        <label>
            <input type='checkbox' id='showPassword' class='checkbox'
                   onclick='togglePasswordVisibility()'>
            <span class='check-label'>Показати пароль</span><br>
            <input type='checkbox' name='agree' class='checkbox' checked required>
            <span class='check-label'>Ви приймаєте наші <a href='#'>Правила та Положення</a></span>
        </label>
    </div>
    <button type='submit' id="submitBtn" class='btn color-btn' disabled>Увійти</button>
    <a class='btn-net' href='/registration'>Немає акаунту?</a>
</form>
```

Рисунок 3.1.4 – Форма авторизації

Далі файли обробки інформації, у файлі авторизації система перевіряє дані, які вводить від користувач, та якщо всі дані вірні система дає доступ користувачу до сайту (рис. 3.1.5). У файлі реєстрації система перевіряє введені дані, на наявність копій в базі даних, якщо копій не має – то користувач успішно реєструється, та попадає на сторінку авторизації (рис. 3.1.6).

```php
$sql_pass = "SELECT passwords FROM `e-phishing`.users WHERE usernames = '$username'";
$result_pass = $conn->query($sql_pass);
if ($result_pass->num_rows == 1) {
    $row = $result_pass->fetch_assoc();
    $storedHashedPassword = $row['passwords'];
}

if (password_verify($password, $storedHashedPassword)) {

    $sql = "SELECT role FROM `e-phishing`.users WHERE usernames = '$username' AND passwords = '$storedHashedPassword'";
    $result_login = $conn->query($sql);

    if ($result_login->num_rows == 1) {
        $row = $result_login->fetch_assoc();
        $_SESSION['usernames'] = $username;
        $_SESSION['logged'] = true;
        $_SESSION['role'] = $row['role'];
        header( header: "Location: /");
        $_SESSION["success_message"] = "Ви успішно ввійшли в систему!";
    } else {
        header( header: 'Location:/login');
        $_SESSION["error_message"] = "Невірне ім'я користувача або пароль!";
    }
    $result_login->close();
} else {
    header( header: 'Location:/login');
    $_SESSION["error_message"] = "Невірне ім'я користувача або пароль!";
}
```

Рисунок 3.1.5 – Авторизація на сайті

```php
list($nick, $domain) = explode( separator: '@', $reg_name);

$hashedPassword = password_hash($reg_pass,  algo: PASSWORD_DEFAULT);

$sql_max_id = "SELECT MAX(userid) as max_id FROM `e-phishing`.USERS";
$result_max_id = $conn->query($sql_max_id);
$row_max_id = $result_max_id->fetch_assoc();
$max_id = $row_max_id['max_id'];

$new_id = $max_id + 1;

$sql_add_user = "INSERT INTO `e-phishing`.USERS (userid, usernames, passwords, email, organization, role)
        VALUES ('$new_id', '$nick', '$hashedPassword', '$reg_name', '$reg_org', 'user')";
```

Рисунок 3.1.6 – Реєстрація користувачів на сайті

І нарешті готовий вигляд форм авторизації та реєстрації на сайті E-PHISHNG (рис. 3.1.7-3.1.8).

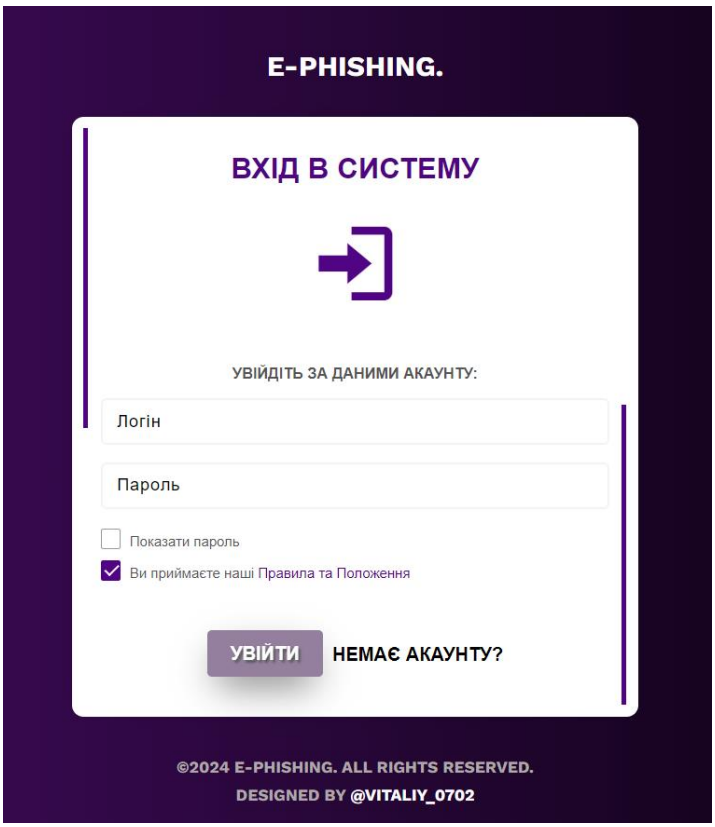


Рисунок 3.1.7 – Готовий вигляд форми авторизації на сайті

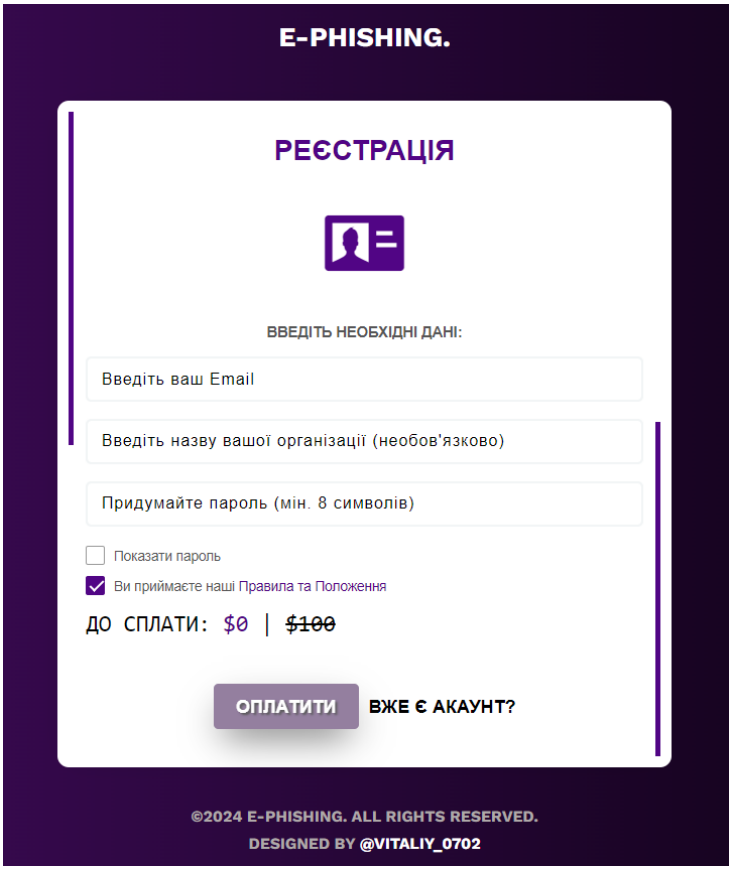


Рисунок 3.1.8 – Готовий вигляд форми реєстрації на сайті

3.4 ВІДПРАВКА «ФІШИНГОВИХ» ПОВІДОМЛЕНЬ

Основне завдання проєкту E-PHISHING – відправка «фішингових» повідомлень на пошту робітникам або ж у будь-який месенджер чи соціальну мережу. Для цього спочатку система формує посилання, а потім пропонує його відправити на пошту, користувачі можуть і не відправляти їх, сформоване посилання і так збережеться.

Почнемо з форми вибору шаблону «фішингового» сайту, на вибір поки що є 3 різні шаблони – це сторінка входу в кабінет СумДУ, сторінка входу в Instagram та сторінка входу в Facebook (рис. 3.4.1-3.4.2).

```
<form method='POST' class='needs-validation' id='$heading_link' novalidate>
  <h3 class='text-center pt-3'><span class='border-number system-background-color'>1</span> Створити посилання
    <div class='row form-group'>
        <div class='col-md-12 animate-box' data-animate-effect='fadeInUp'>
            <label for='theme_social'>Шаблон Фішингового сайту</label>
            <select name='theme_social' id='theme_social' class='form-control' required>";
            $social = $_POST['theme_social'];
            echo "<option value='' disabled selected>Виберіть шаблон &#128315;</option>";
            require_once __DIR__ . '/../db/message_type_sql.php';
            if ($result_message_type) {
                while ($row = $result_message_type->fetch_assoc()) {
                    $message_link = htmlspecialchars($row["message_link"]);
                    $message_phishing_name = htmlspecialchars($row["message_phishing_name"]);
                    $message_name = htmlspecialchars($row["message_name"]);
                    echo "<option value='$message_link' data-message-name='$message_name'>$message_phishing_name</option>";
                }
                $result_message_type->close();
            } else {
                $_SESSION['error_message'] = 'Шаблони не завантажені або відсутні!';
            }
        echo "</select>
        <input type='hidden' name='message_name' id='message_name'>
        <script src='/js/send-message-name.js' defer loading='lazy'></script>
    </div>
</div>
<div class='form-group text-center'>
    <button type='submit' class='btn color-btn mt-3 mb-3 w-100'>Створити</button>
</div>
</form>";
```

Рисунок 3.4.1 – Програмний код форми вибору «фішингового» шаблону

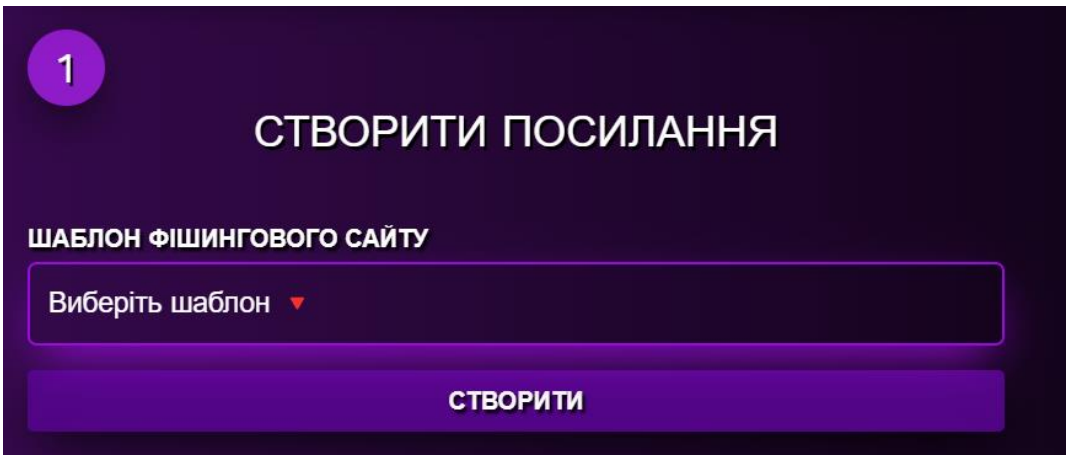


Рисунок 3.4.2 – Вигляд форми вибору «фішингового» шаблону на сайті

Вибравши необхідний шаблон, наприклад «Кабінет СумДУ» створюється посилання, яке можна скопіювати чи замаскувати (кнопка замаскувати перенаправляє на сервіс cutt.ly) і також можна перейти до форми відправки повідомлення на пошту (бонусом є кнопка, яка відкриває інструкцію, як заховати посилання в текст у соціальній мережі телеграм) (рис. 3.4.3-.3.4.4).

```php
require_once __DIR__ . '/../functions/generateRandomString.php';
$gen_id = generateRandomString();

$link = $domain.$social."?id=".$gen_id;

$sql_reverse = "INSERT INTO `e-phishing`.reverse_users VALUES ('$gen_id','$my_name', '$link')"
$result_sql_reverse = $conn->query($sql_reverse);

$message_names = $_POST['message_name'];
```

Рисунок 3.4.3 – Програмний код сформованого посилання

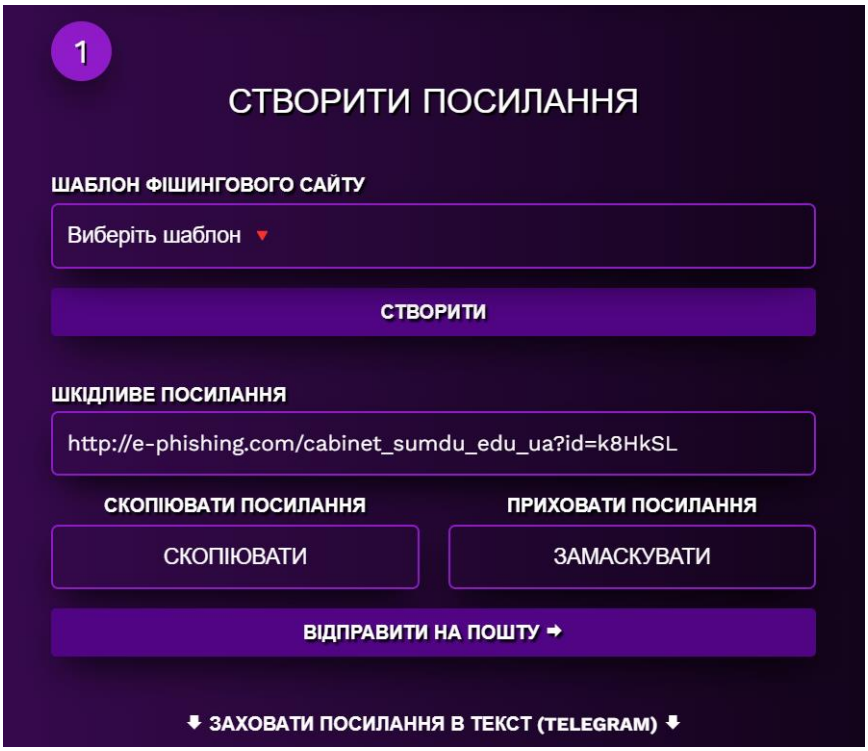
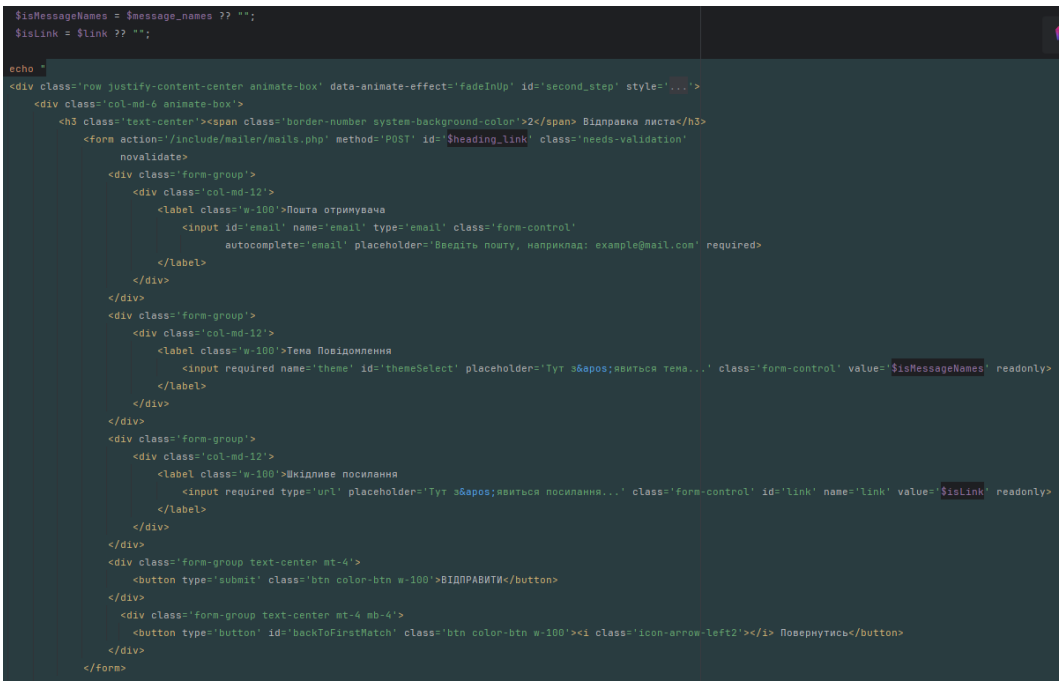


Рисунок 3.4.4 – Вигляд сформованого посилання на сайті

Сформувавши посилання користувач має можливість відправити його на пошту, то ж перейдемо до відправки посилання на пошту. Тема повідомлення отримується з бази даних на основі вибраного «фішингового» шаблону, система також має текстові шаблони для відправки на пошту. Програмний код форми виглядає так (рис. 3.4.5). На сайті ж відображується так (рис. 3.4.6).



```
$isMessageNames = $message_names ?? "";
$isLink = $link ?? "";

echo "
<div class='row justify-content-center animate-box' data-animate-effect='fadeInUp' id='second_step' style='...'>
    <div class='col-md-6 animate-box'>
        <h3 class='text-center'><span class='border-number system-background-color'>2</span> Відправка листа</h3>
            <form action='/include/mailer/mails.php' method='POST' id='$heading_link' class='needs-validation'
                  novalidate>
                <div class='form-group'>
                    <div class='col-md-12'>
                        <label class='w-100'>Пошта отримувача
                            <input id='email' name='email' type='email' class='form-control'
                                   autocomplete='email' placeholder='Введіть пошту, наприклад: example@mail.com' required>
                        </label>
                    </div>
                </div>
                <div class='form-group'>
                    <div class='col-md-12'>
                        <label class='w-100'>Тема Повідомлення
                            <input required name='theme' id='themeSelect' placeholder='Тут з&apos;явиться тема...' class='form-control' value='$isMessageNames' readonly>
                        </label>
                    </div>
                </div>
                <div class='form-group'>
                    <div class='col-md-12'>
                        <label class='w-100'>Шкідливе посилання
                            <input required type='url' placeholder='Тут з&apos;явиться посилання...' class='form-control' id='link' name='link' value='$isLink' readonly>
                        </label>
                    </div>
                </div>
                <div class='form-group text-center mt-4'>
                    <button type='submit' class='btn color-btn w-100'>ВІДПРАВИТИ</button>
                </div>
                  <div class='form-group text-center mt-4 mb-4'>
                    <button type='button' id='backToFirstMatch' class='btn color-btn w-100'><i class='icon-arrow-left2'></i> Повернутись</button>
                </div>
            </form>
```

Рисунок 3.4.5 – Форма відправки «фішингового» повідомлення на пошту

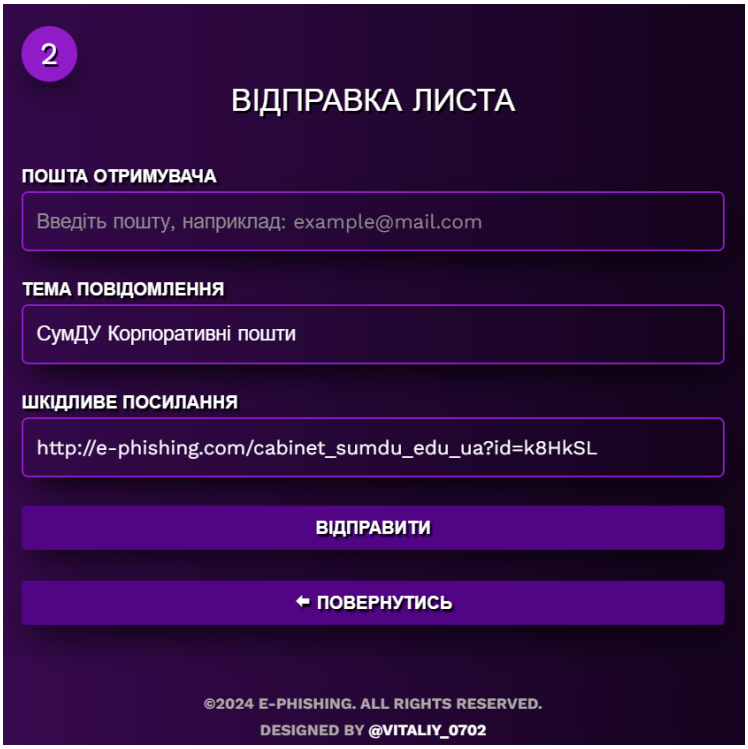


Рисунок 3.4.6 – Форма відправки «фішингового» повідомлення на сайті

Процес відправки повідомлення простий, за допомогою вбудованої функції в PHP – mail. Код відправлення повідомлення дуже громіздкий, тому його можна знайти в Додатку А.

Після його відправки чи навіть після формування посилання, на цій же сторінці зверху з'являється карточка з сформованими посиланнями (рис. 3.4.7).

```
if ($result_id) {
    while ($row = $result_id->fetch_assoc()) {
        $link_num += 1;
        echo "<form METHOD='POST' ACTION='/include/handler-phishing/delete-phishing-links.php'>
                    <div class='min-size-text shadow-max m-2'>
                        <div class='px-3 py-4'>$link_num. {$row["reverse_link"]}
                            <button type='submit' class='p-1 icon-right icoяn-background'
                            name='delete-link' value='{$row["reverse_id"]}'>
                                <i class='min-size-text icon-trash2' title='Видалити'></i>
                            </button>
                        </div>
                    </div>
                </form>";
    }
    $result_id->close();
} else {
    die($_SESSION['error_message'] = 'ID не завантажені!');
}
```

Рисунок 3.4.7 – Програмний код відображення сформованих посилань

Якщо ви хочете відправити ще раз посилання з таким же шаблоном і не на пошту, то краще скористатися уже створеним посиланням. На сайті карточка з посиланнями виглядає так (рис. 3.4.8).

Рисунок 3.4.8 – Сформовані посилання, вигляд на сайті

3.5 «МОЇ ЛИСТИ» - МОНІТОРИНГ ЗА ПОСИЛАННЯМ

Сформували і відправили посилання? Далі ж потрібно фіксувати переходи по посиланню і введенню даних на «фішингових» формах. Для цього було створено сторінку «Мої листи». Не вдаючись в подробиці як збирається інформація з «фішингових» форм переходимо до відображення на сайті. Спочатку потрібна форма з уже створеними ідентифікаторами для перегляду потрібного відправленого посилання. Програмний код виглядає так (рис. 3.5.1).

```
echo "<form method='POST' class='m-5 text-center' id='$heading_link'>
        <div class='form-group row justify-content-center'>
           <div class='application pb-5 w-70'>
               <label class='' for='my_ids'><i class='icon-light-bulb'></i> Пошук за ідентифікатором листа
                   <select name='my_ids' id='my_ids' class='form-control shadow-max' required>
                   <option value='' class='text-center' disabled selected> Виберіть ID &#128315;</option>";
                   $my_ids = $_POST['my_ids'];

                   require_once __DIR__ . '/../functions/extractServiceName.php';

                   if ($result_id) {
                       while ($row = $result_id->fetch_assoc()) {
                           $social_name = extractServiceName($row["reverse_link"]);
                           echo "<option data-message-name='{$row["reverse_link"]}'
                                 value='{$row["reverse_id"]}'> {$row["reverse_id"]} ($social_name)</option>";
                       }
                       $result_id->close();
                   } else {
                   echo "<option value=''>ID не завантажені!</option>";
                   }
                   echo "</select>
                         <input type='hidden' name='reverse_link' id='reverse_link'>
                         <script src='/js/send-message-name.js' defer loading='lazy'></script>
                   <button type='submit' class='btn color-btn mt-2 w-100'><i class='icon-search'></i> Знайти</button>
               </label>
           </div>
        </div>
</form>";
```

Рисунок 3.5.1 Програмний код форми пошуку відправлених листів по ID

На сайті форма пошуку виглядає саме так (рис. 3.5.2).

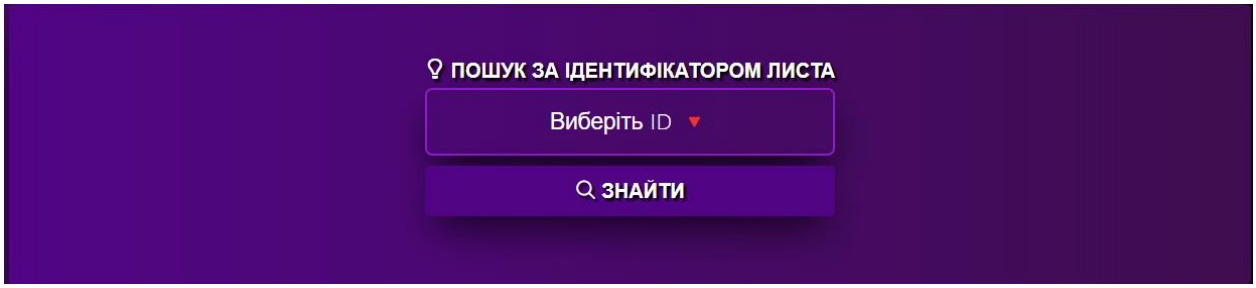


Рисунок 3.5.2 – Форма пошуку відправлених листів по ID на сайті

У п. 3.4 ми створили посилання з ідентифікатором k8HkSL, тому шукатимемо дані саме по ньому. Форма відображення контенту написана саме так (рис. 3.5.3).

```
echo "<div class='container mt-3' id='letters-search-result'>
        <div class='row justify-content-center'>
            <div class='application smart-size animate-box' data-animate-effect='fadeInUp'>
                <h3 class='text-center mt-4'>Результат пошуку: <span class='fw-bold text-transform-none'>$my_ids</span></h3>
                <button type='submit' class='top-right-reload fs-4 p-1' onclick='#'>
                    <i class='icon-reload'></i>
                </button>
                <div class='user-info p-lg-5'>";
                   echo "
                    <p>Сформоване посилання:<br>
                        <span class='text-transform-none'>[ <a href='$phishing_url' target='_blank'>$phishing_url</a> ]</span>
                    </p>
                    <p>Відправлений лист:<br>";
                        require_once 'include/data/email_message_data.php';
                        echo "
                    </p>
                    <p>Переходи:<br>";
                        require_once 'include/data/open_links_data.php';
                        echo "
                    </p>
                    <p>Отримані дані:<br>";
                        require_once 'include/data/user_data_data.php';
                        echo "
                    </p>";
                echo "
                </div>
            </div>
        </div>
```

Рисунок 3.5.3 – Форма відображення взаємодії з відправленими посиланнями

Відправлені листи, переходи та отримані дані збираються з бази даних та виводяться у дану форму, на прикладі отриманих даних відображено як було зроблено запит до бази, та які дані взяли для виведення користувачу (рис. 3.5.4).

```
<?php
require_once __DIR__ . '/../db/user_data_sql.php';
if ($victim_result) {
    if ($victim_result->num_rows > 0) {
        while ($row = $victim_result->fetch_assoc()) {
            $timestamp = strtotime($row["victim_date"]);
            $date_victim = date( format: "d.m.Y H:i:s", $timestamp);

            echo "<span class='text-transform-none'>[  $date_victim  |
                            {$row["victim_ip"]} <a class='color-red text-decoration-none'
                            title='Пошук по IP' href='https://2ip.ua/ru/services/information-service/whois?ip=$ip_op'
                            target='_blank'>(?)</a> |
                            {$row["victim_login"]}:{$row["victim_password"]} ]</span><br>";
        }
        $victim_result->close();
    } else {
        echo "По даному ID отриманих даних не знайдено!";
    }
} else {
    echo "Помилка при обробці запиту.";
}
```

Рисунок 3.5.4 – Відображення інформації з бази даних про отримані дані

Перейдемо по «фішинговому» посиланню та введемо тестові дані у поля. Пошта – test@test.com, пароль – 123452. Повернемось у нашу систему та перевіримо чи записались всі дану у форму відображення взаємодії з відправленими повідомленнями (рис. 3.5.5).

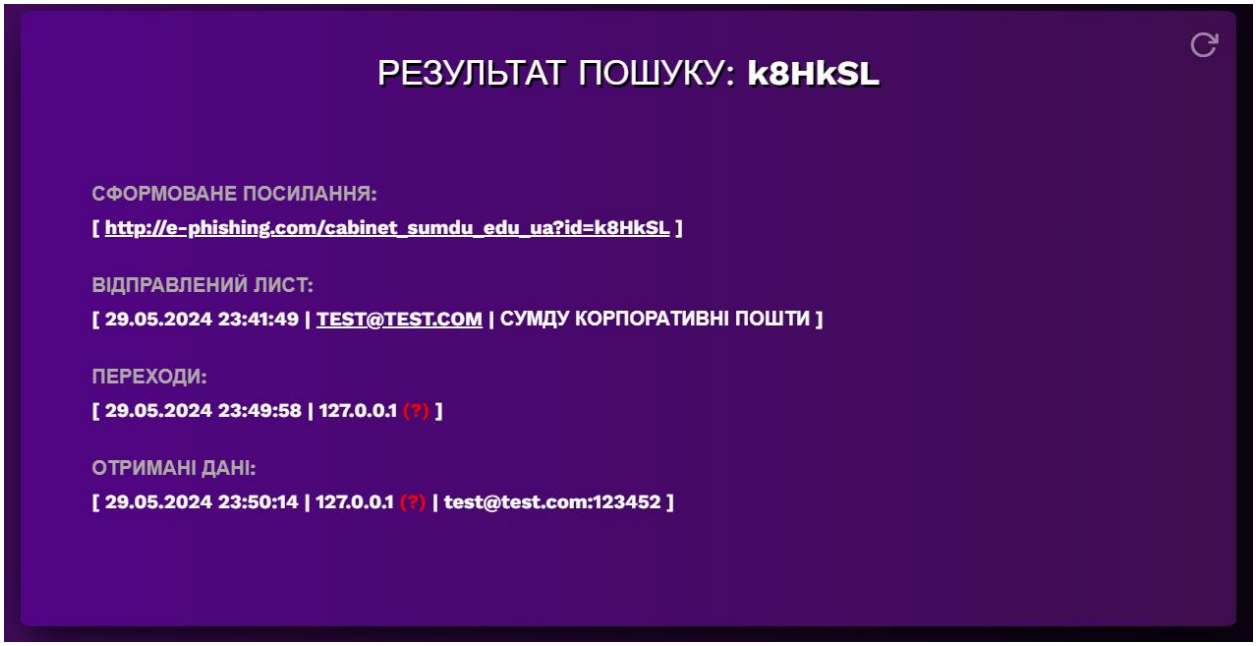


Рисунок 3.5.5 – Форма відображення взаємодії з відправленими повідомленнями на сайті

Як бачимо все успішно відображається на вище зображеній картинці. Також біля IP-адрес є знаки (?). Натиснувши на нього ми можемо перейти на сервіс 2ip.ua і переглянути інформацію за відображеною IP- адресою.

3.6 ЛИСТ РОЗРОБНИКУ: ВІД ВІДПРАКИ ДО ОБРОБКИ

Важливою можливістю на проєкті E-PHISHING є відправка листа розробнику з темами: заявка на додавання вашого шаблону в теми «фішингових» листів, заявка на створення сайту, допомога по використанню сервісу. Форма виглядить достатньо стандартно і майже не відрізняється від попередніх (рис. 3.6.1-3.6.2). У самій формі відправки листа розробнику рядки пошта та організація виставляються автоматично з даних користувача, які він вводив при реєстрації, якщо необов'язковий пункт про організацію був пропущений, то система поставить мітку «Без організації», якщо ж є назва організації, то буде її назва і з'явиться нове поле – посада.

```
<form action='/include/contact/contact' method='POST' class='needs-validation' novalidate>
    <div class='row form-group'>
        <div class='col-md-6'>
            <label for='first_name'>Ім'я</label>
            <input type="text" id="first_name" name="first_name"
                   autocomplete="off" placeholder="Введіть своє ім'я..." class="form-control" required>
        </div>
        <div class='col-md-6'>
            <label for='last_name'>Прізвище</label>
            <input type='text' id='last_name' name='last_name'
                   autocomplete='off' placeholder='Введіть своє прізвище...' class='form-control' required>
        </div>
    </div>

    <div class='row form-group'>
        <div class='col-md-12'>
            <label for='email'>Пошта</label>
            <input type='email' id='email' name='email' class='form-control'
                   value='<?= $email ?>' autocomplete='off' required readonly>
        </div>
    </div>

    <div class='row form-group'>
        <div class='col-md-12'>
            <label for='organization'>Організація</label>
            <input type='text' id='organization' name='organization'
                   autocomplete='off' value='<?= $organization ?>' class='form-control' readonly
                   required>
        </div>
    </div>
    <?= $org_position ?>
```

Рисунок 3.6.1 – Форма «Лист розробнику» №1

```
<?= $org_position ?>
<div class="row form-group">
    <div class="col-md-12">
        <label for="theme_contact">Тема</label>
        <select name="theme_contact" id="theme_contact" class="form-control" required>
            <?php
            $attr_theme = $_GET['attr_theme'] ?? '';
            if($_GET['attr_theme'] == 'create-site'){
                echo "<option value='1' selected>Заявка на створення сайту</option>";
            }else{
                echo "<option value='' disabled selected>Виберіть тему &#128315;</option>";
                if ($result_to_dev_type) {
                    while ($row = $result_to_dev_type->fetch_assoc()) {
                        echo "<option value='{$row["to_dev_type_id"]}'>{$row["to_dev_type_name"]}</option>";
                    }
                    $result_to_dev_type->close();
                } else {
                    $_SESSION['error_message'] = 'Теми повідомлень не завантажені або відсутні!';
                }
            }
            ?>
        </select>
    </div>
</div>
<div class="form-group text-center">
    <button type="submit" class="btn color-btn mt-3 mb-3 w-100">Відправити</button>
</div>
</form>
```

Рисунок 3.6.2 – Форма «Лист розробнику» №2

Далі на цій сторінці є ще одна форма, у якій відображаються уже відправлені листи (із змінним статусом обробки повідомлення) (рис. 3.6.3).

```
echo "<form action='/include/contact/to-dev-delete-message.php' method='POST'>
        <div class='min-size-text shadow-max m-2'>
            <div class='px-3 py-4'>
                <button class='p-1 m-0 icon-right icon-background top' name='delete-to-dev-message' value='{$row["to_dev_id"]}'>
                    <i class='icon-trash2' title='Видалити'></i>
                </button>
                <div class='px-5'>$to_dev_ids. {$row_dev_name["to_dev_type_name"]} <i class='color-red icon-arrow-right2'></i>
                    $status_text
                </div>
            </div>
        </div>
    </form>";
```

Рисунок 3.6.3 – Форма вже відправлених повідомлень розробнику

Змінна $to_dev_ids відповідає за ідентифікатор відправленого листа, далі наступна змінна в цьому ж рядку відповідає за тему листа, і $status_text відповідає за статус обробки повідомлення, наприклад: прийнята, відмінено і тп.

Дані з форми передаються до файлу обробника під назвою contact.php, він приймає дані, визначає ідентифікатор, з яким записати у базу дану інформацію та записує її (рис. 3.6.4).

```php
$user = $_SESSION['usernames'];
$sql_max_id = "SELECT MAX(to_dev_id) as max_id FROM `e-phishing`.to_developer";
$result_max_id = $conn->query($sql_max_id);
$row_max_id = $result_max_id->fetch_assoc();
$max_id = $row_max_id['max_id'];
$new_id = $max_id + 1;

require_once __DIR__ . '/../user-identification/current-date-time.php';

$sql_add_message = "INSERT INTO `e-phishing`.to_developer (to_dev_id, to_dev_first_name,
                                to_dev_last_name, to_dev_login, to_dev_email,
                                to_dev_org,to_dev_pos,to_dev_theme_contact,to_dev_time_mail, to_dev_status)
            VALUES ('$new_id', '$first_name', '$last_name', '$user', '$email', '$org', '$position',
                    '$theme_contact', '$currentDateTime', 'no')";

if ($conn->query($sql_add_message) === TRUE) {
    $_SESSION["success_message"] = "Повідомлення відправлено!";
} else {
    $_SESSION["error_message"] = "Помилка при відправці повідомлення!";
}
header( header: "Location:/contact-us#contact");
$conn->close();
```

Рисунок 3.6.4 – Обробник даних листа розробнику

На самому сайті вищеописані форми виглядають саме так (рис. 3.6.5).

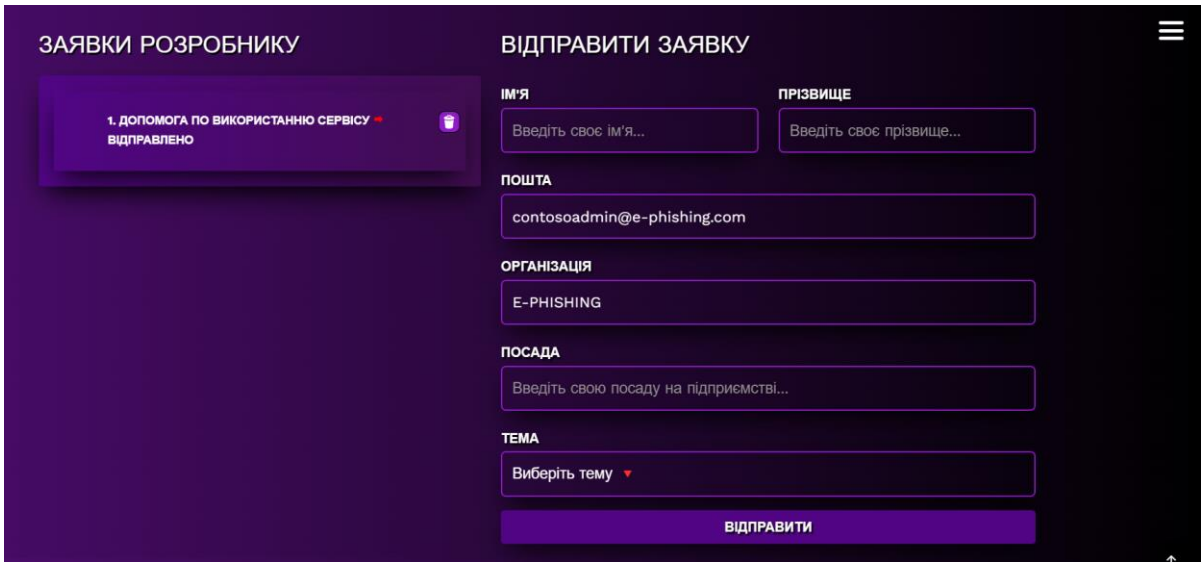


Рисунок 3.6.5 – Вигляд сторінки «Контакти»

Для того, щоб адміністратор міг обробляти листи розробнику, для них доступний у меню користувача пункт «Заявки» (рис. 3.6.6).

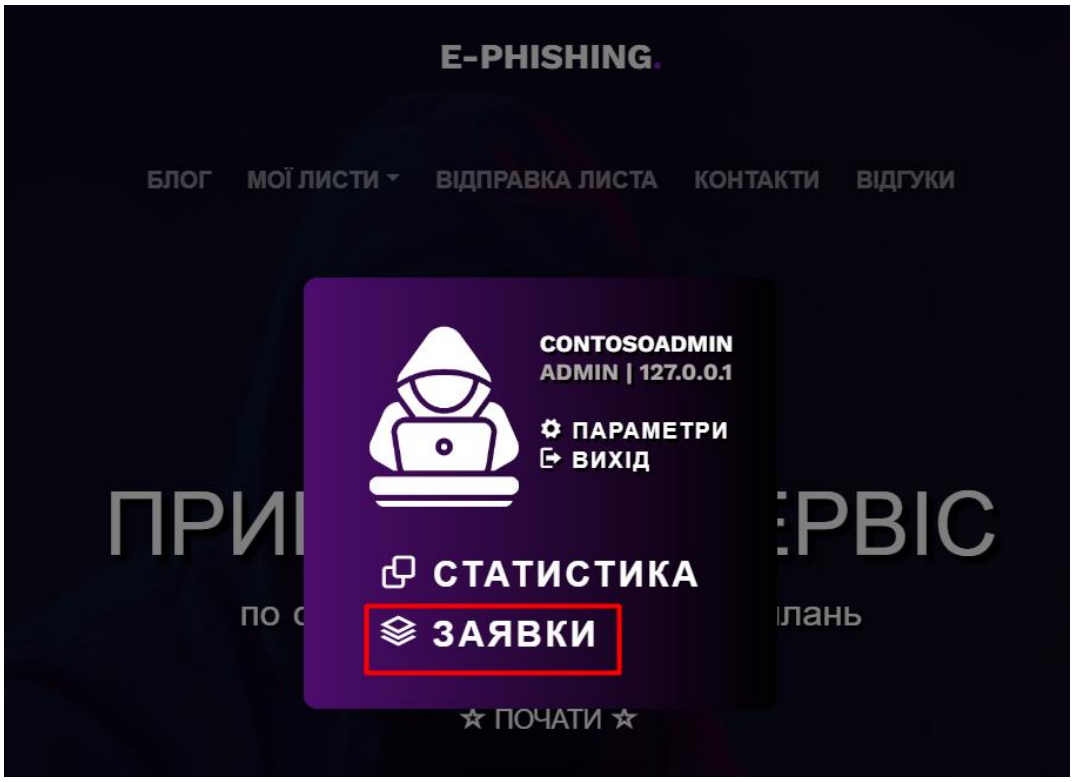


Рисунок 3.6.6 – Пункт «Заявки» у меню користувача

У ньому виводяться всі заявки, які відправляв користувач (рис. 3.6.7).

```
$to_dev_id = $row["to_dev_id"];
if ($row["to_dev_status"] == 'no' || $row["to_dev_status"] == 'pro') {
    $user_info_html = "<div class='user-info'>
            <h4>№{$row['to_dev_id']}</h4>
            <h2>{$row["to_dev_first_name"]} {$row["to_dev_last_name"]}</h2>
            <p>Email: <a href='mailto:{$row["to_dev_email"]}'>{$row["to_dev_email"]}</a></p>
            <p>Організація, посада: <span>$org_pos</span></p>
        </div>";

    $application_details_html = "<div class='application-details'>
                        <p>Тема: <span>{$row_dev_name["to_dev_type_name"]}</span></p>
                        <p>Дата відправки: <span>" . date( format: "d.m.Y H:i:s", $timestamp) . "</span></p>";

    if ($row["to_dev_status"] == 'no') {
        $application_details_html .= "<p class='top-right'>Нова заявка</p>
                        <form action='/include/contact/to-dev-submit.php' method='post'>
                            <input value='$to_dev_id' name='dev_mail_id' hidden='hidden'>
                            <button type='submit' name='submit-letter' class='approve-btn'>Обробити</button>
                            <button type='submit' name='delete-letter' class='reject-btn'>Відмінити</button>
                        </form>";
    }
    if ($row["to_dev_status"] == 'pro') {
        $application_details_html .= "<p class='top-right'>Заявка в обробці!</p>
                        <form action='/include/contact/to-dev-submit.php' method='post'>
                            <input value='$to_dev_id' name='dev_mail_id' hidden='hidden'>
                            <button type='submit' name='submit-process' class='approve-btn'>Підтвердити виконання</button>
                            <button type='submit' name='delete-letter' class='reject-btn'>Відмінити</button>
                        </form>";
    }
```

Рисунок 3.6.7 – Формування карток з заявками користувачів

Також форми з обробкою повідомлень передаються до файлу-обробнику to-dev-submit.php, там проходить змінення статусу заявки, яке потім і відображається у користувачів у їхній формі «Заявки розробнику» (рис. 3.6.8).

```php
if (isset($_POST['dev_mail_id'])) {
    $mail_id = $_POST['dev_mail_id'];
    if (isset($_POST['submit-letter'])) {

        $sql = "UPDATE `e-phishing`.`to_developer` SET `to_dev_status` = 'pro' WHERE `to_dev_id` = '$mail_id'";

        if ($conn->query($sql) === TRUE) {
            $_SESSION["success_message"] = "Заявку відправлено в обробку!";
        } else {
            $_SESSION["error_message"] = "Помилка при відправці заявки до обробки!";
        }
    }
    if (isset($_POST['delete-letter'])) {

        $sql = "UPDATE `e-phishing`.`to_developer` SET `to_dev_status` = 'del' WHERE `to_dev_id` = '$mail_id'";

        if ($conn->query($sql) === TRUE) {
            $_SESSION["error_message"] = "Заявку видалено!";
        } else {
            $_SESSION["error_message"] = "Помилка при видаленні заявки!";
        }
    }
    if (isset($_POST['submit-process'])) {

        $sql = "UPDATE `e-phishing`.`to_developer` SET `to_dev_status` = 'yes' WHERE `to_dev_id` = '$mail_id'";

        if ($conn->query($sql) === TRUE) {
            $_SESSION["success_message"] = "Підтверджено! Заяка закрита";
        } else {
            $_SESSION["error_message"] = "Помилка при підтвердженні!";
        }
    }
```

Рисунок 3.6.8 – Обробка статусу заявки

І нарешті поглянемо як виглядає така сторінка з заявками (із акаунту адміністратора) (рис. 3.6.9).

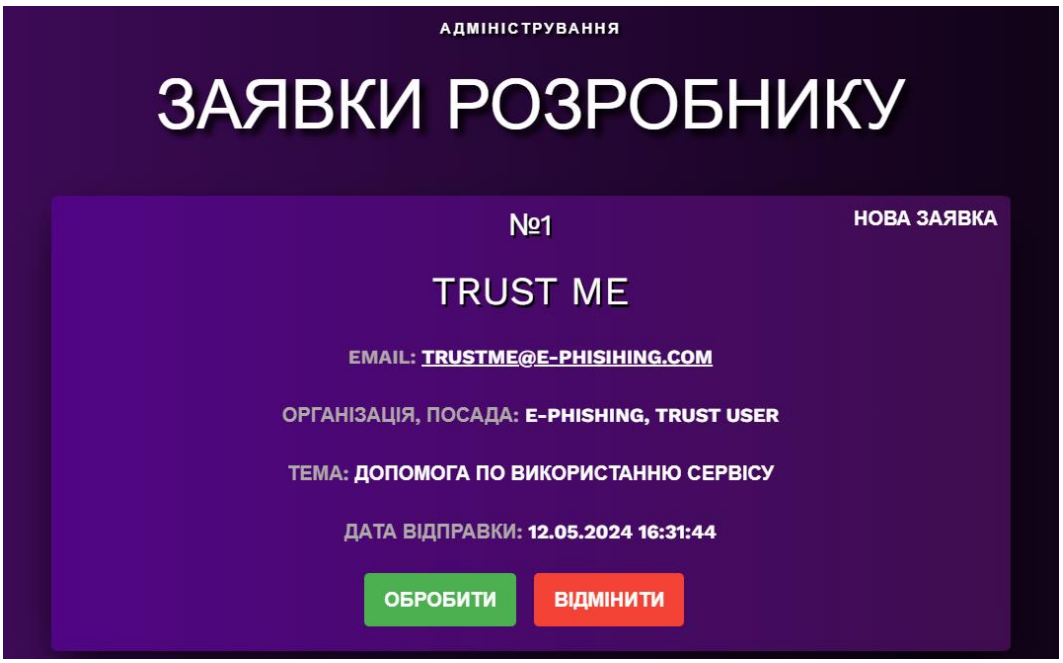


Рисунок 3.6.9 – Вигляд сторінки з заявками адміністратору

3.7 ФОРМУВАННЯ СТАТИСТИКИ

Статистика у системі E-PHISHING формується на основі всіх відправлених листів, переходів по вашим посиланням, отриманим даним, листів розробнику та залишеним відгукам.

Спочатку окремо зробимо запити до бази даних по всім вищевказаним даним (запити робляться до таблиць, в яких містяться ці дані) (рис. 3.7.1).

```php
<?php
//Відправлені
$max_message = $conn->query( query: "select COUNT(e.`email_message_id`) as max_message
                                    from `e-phishing`.`email_message` e JOIN `e-phishing`.`users` u on e.`user_id` = u.`userid`
                                    where u.`usernames` = '$my_name';")->fetch_assoc()['max_message'];
//=============================================
//Переходи і отримані
$totalMatches1 = 0;
$totalMatches2 = 0;
require_once __DIR__ . '/search_reverse_id.php';
if ($result_id->num_rows > 0) {
    while($row = $result_id->fetch_assoc()) {
        $ids = $row["reverse_id"];

        $totalMatches1 += intval($conn->query( query: "SELECT COUNT(*) AS count FROM `e-phishing`.open_links
                        WHERE use_id = '$ids'")->fetch_assoc()['count']);

        $totalMatches2 += intval($conn->query( query: "SELECT COUNT(*) AS count2 FROM `e-phishing`.user_data
                         WHERE victim_use_id = '$ids'")->fetch_assoc()['count2']);
    }
} else {$totalMatches1 = 0; $totalMatches2 = 0;}
//=============================================
//Розробнику
$to_dev_max = $conn->query( query: "select COUNT(`to_dev_id`) as max_to_dev
                                    from `e-phishing`.to_developer where `to_dev_login` = '$my_name';")->fetch_assoc()['max_to_dev'];
//=============================================
//Відгуки
$reviews_max = $conn->query( query: "select COUNT(e.`message_id`) as max_reviews
                                    from `e-phishing`.`feedback` e JOIN `e-phishing`.`users` u on e.`user_id` = u.`userid`
                                    where u.`usernames` = '$my_name';")->fetch_assoc()['max_reviews'];
//=============================================
```

Рисунок 3.7.1 – Запити до бази даних

Далі передаємо дані до масиву, покажемо, як приклад, на першій карточці «Відправлені листи» (рис. 3.7.2).

```php
$statistic_array = array(
    array(
        'image' => '/images/statistic-img/letter.webp',
        'title' => 'Відправлені листи',
        'description' => 'Всього відправлено листів:',
        'count' => $max_message,
        'link1' => '/send-mail-page#form-send-mail',
        'link1_text' => 'Відправити лист',
        'link2' => '/my-letters#letters',
        'link2_text' => 'Перелянути відправлені листи',
    ),
```

Рисунок 3.7.2 – Масив «Відправлені листи»

Далі нам потрібно відобразити дані з масиву на веб-сторінці (рис. 3.7.3).

```
foreach ($statistic_array as $stat) {
echo "
    <div class='col-md-4 mt-2'>
        <div class='card mb-4 shadow-max animate-box' data-animate-effect='fadeInUp'>
            <img src='".$stat['image']."' loading='lazy' alt='".$stat['title']."'>
            <div class='card-body text-center'>
                <h5 class='card-title mb-3'>".$stat['title']."</h5>
                <p class='card-text'>".$stat['description']." <span>".$stat['count']."</span></p>
                <a class='btn color-btn' href='".$stat['link1']."'>".$stat['link1_text']."</a>
                <a class='btn color-btn' href='".$stat['link2']."'>".$stat['link2_text']."</a>
            </div>
        </div>
    </div>";
}
```

Рисунок 3.7.3 – Відображення даних з масиву у конструкцію картки

Поглянемо, як дана статистика виглядає на самому сайті (рис. 3.7.4).

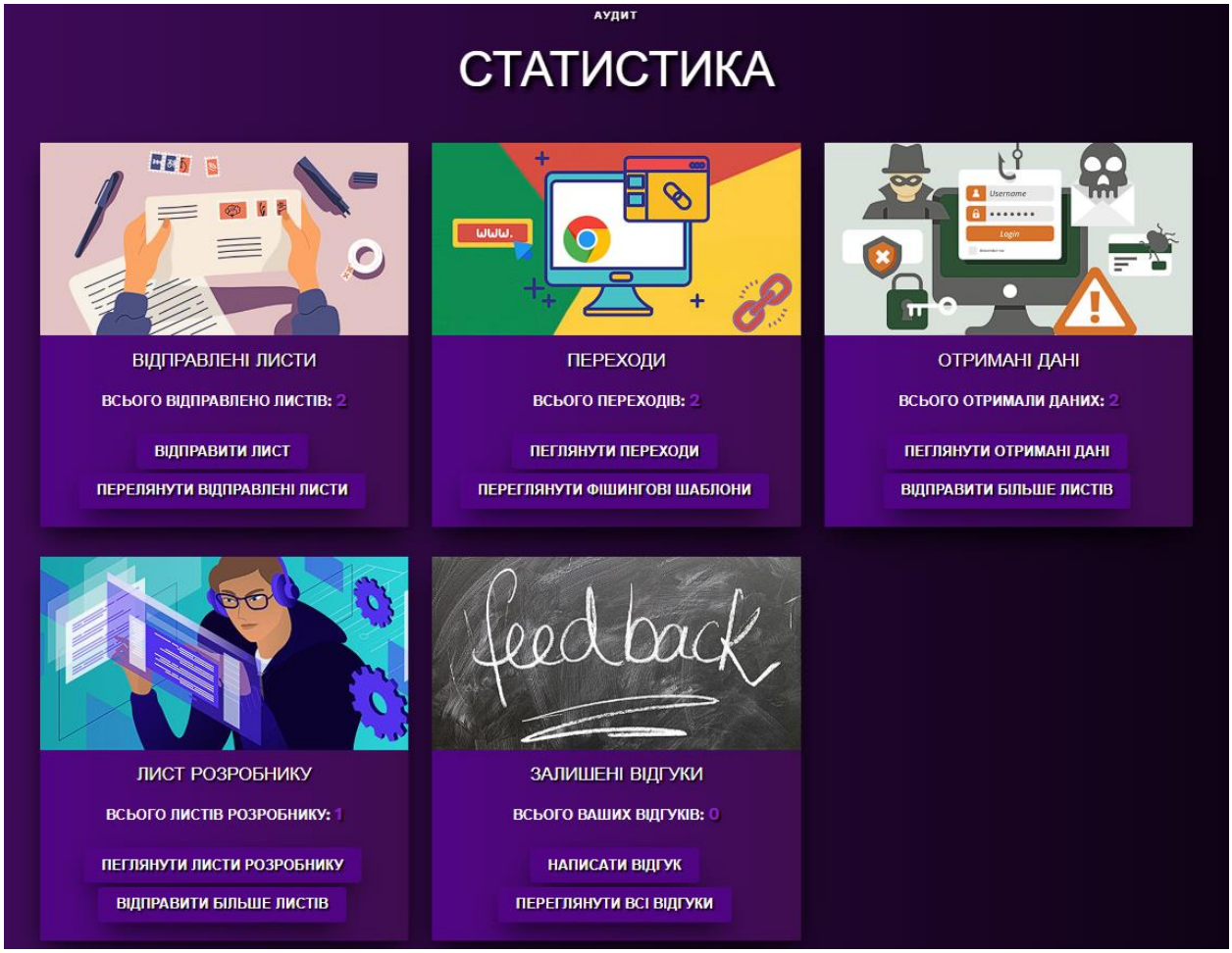


Рисунок 3.7.4 – Відображення статистики на веб-сайті

3.8 БЛОГ – ВІДОМОСТІ ПРО СИСТЕМУ

Блог – у сервісі E-PHISHING відіграє роль карточок з новинами про оновлення сайту, у подальшому буде реалізовано і інший функціонал, який зазвичай притаманний вкладинкам «Блог» на сайтах. Особливістю сторінки «Блог» є те, що кнопка «Створити пост» доступна тільки адміністраторам, всі інші користувачі можуть тільки переглядати карточки з новинами про оновлення сайту. Кнопка «Створити пост» та спливаюче меню з формою у програмному коді реалізовано так (рис. 3.8.1).

```php
<?php
if ($_SESSION['role'] === 'admin') {
    $modal_name = "postModal";
    $svg = "";
    $h5_text = "Створення поста";
    $form = "<form class='bg-section system-background-color px-5 w-100' action='/include/update-service/add-new-post.php' method='post'>
            <label for='post_name'>Тема поста</label>
            <input type='text' id='post_name' name='post_name' class='form-control' style='...' placeholder='Напишіть тему...'>
            <label for='post_description'>Текст поста</label>
            <textarea type='text' id='post_description' name='post_description' class='form-control' style='...'
            placeholder='Напишіть текст...'></textarea>
            <button class='btn color-btn w-100 mt-2'>Опублікувати пост</button>
            <div class='pt-5'></div>
            </form>";
    echo "<button class='btn color-btn' data-toggle='modal' data-target='#$modal_name'><i class='icon-stack-2'></i> Створити пост</button>";
} else {echo "";}
?>
```

Рисунок 3.8.1 – Створення поста (доступно для адміністратора)

Далі, коли адміністратор заповнив всі поля у формі створення поста, дані передаються у файл обробник під назвою add-new-post.php. У цьому файлі перевіряється наявність даних та вони записуються до бази даних (рис. 3.8.2).

```php
$user = $_SESSION['usernames'];
$post_date = date( format: "Y-m-d");

$sql_max_id = "SELECT MAX(post_id) as max_id FROM `e-phishing`.update_service_posts";
$result_max_id = $conn->query($sql_max_id);
$row_max_id = $result_max_id->fetch_assoc();
$max_id = $row_max_id['max_id'];
$new_id = $max_id + 1;

$sql_add_post = "INSERT INTO `e-phishing`.update_service_posts (post_id, post_name, post_description, post_date, post_creator_name)
            VALUES ('$new_id', '$post_name', '$post_description', '$post_date', '$user')";

if ($conn->query($sql_add_post) === TRUE) {
    $_SESSION["success_message"] = "Пост опубліковано!";
} else {
    $_SESSION["error_message"] = "Помилка при публікуванні!";
}
header( header: "Location:/blog#update-service");
$conn->close();
```

Рисунок 3.8.2 – Запис до бази даних інформації про пост

Щоб користувачі могли на сторінці «Блог» побачити дані пости, потрібно циклом вивести всю інформацію, яку ми записували до бази даних (рис. 3.8.3).

```php
$query = "SELECT * FROM `e-phishing`.`update_service_posts`
                                                ORDER BY `post_id` DESC LIMIT $offset, $itemsPerPage";
$result_posts = $conn->query($query);

if ($result_posts) {
    while ($row = $result_posts->fetch_assoc()) {
        $timestamp = strtotime($row["post_date"]);
        echo "<div class='col-md-4 col-sm-4 animate-box user-select-none' data-animate-effect='fadeInUp'>
                <div class='post'>
                    <span class='date'>№{$row['post_id']} | " . date( format: "d.m.Y", $timestamp) . "</span>
                    <h3>{$row['post_name']}</h3>
                    <p>{$row['post_description']}</p>
                    <p class='author'>
                        <cite>
                            <i class='icon-user'></i> {$row['post_creator_name']}
                        </cite>
                    </p>
                </div>
            </div>";
    }
} else {
    echo "<h2 class='text-center'>На даний момент постів немає!</h2>";
}
$result_posts->close();
```

Рисунок 3.8.3 – Відображення постів на сайті

Переглянемо готовий результат, як це бачить користувач на сайті (кнопку «Створити пост» бачить тільки адміністратор) (рис. 3.8.4).

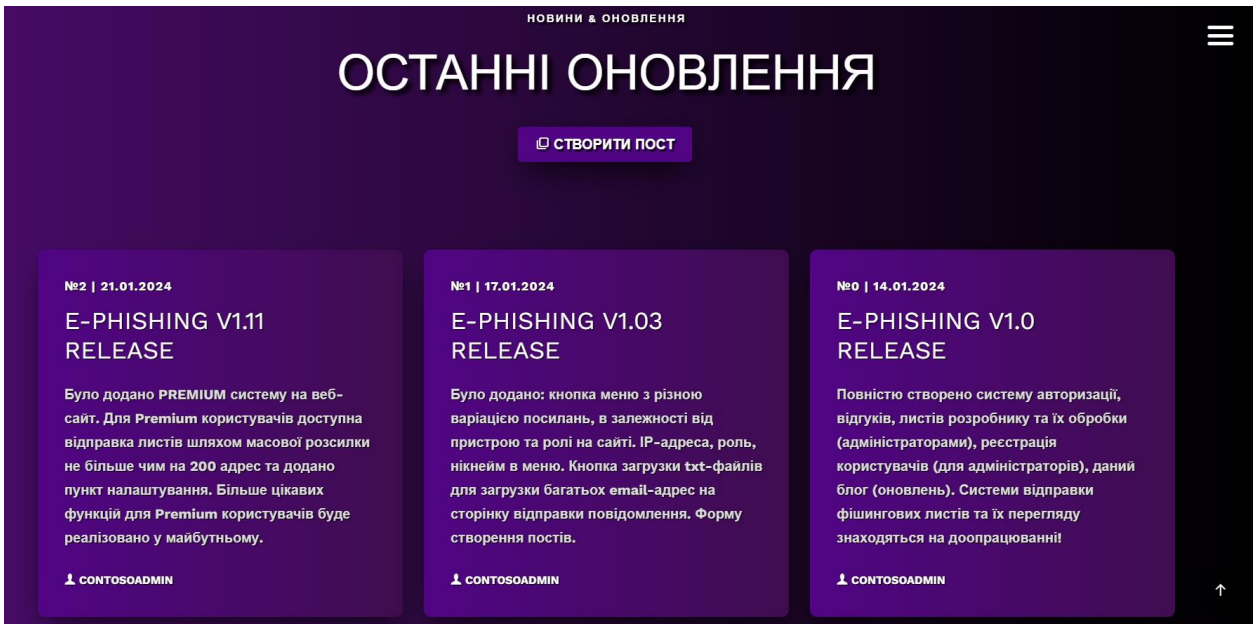


Рисунок 3.8.4 – Відображення постів на сайті

3.9 ВІДГУКИ

Відгуки є невід'ємною частиною будь-якої системи, E-PHISHING не є виключенням. Спочатку створюється форма відправки повідомлення (рис. 3.9.1).

```
<div class='row justify-content-center'>
      <div class='col-md-12'>
        <form id='textarea-reviews' action='/include/comment/add-new-comment.php' method='POST'>
          <label for='comment'>Написати відгук</label>
          <textarea name='comment' id='comment' placeholder='Напишіть відгук про нас...' class='form-control' style='...' required></textarea>
          <br>
          <div class='form-group text-center'>
              <button type='submit' class='btn color-btn mb-3'>ЗАЛИШИТИ ВІДГУК</button>
           </div>
        </form>
      </div>
</div>";
```

Рисунок 3.9.1 – Форма відправки відгуку

Передаємо з форми текст у файл add-new-comment.php та там обробляємо його, визначаємо максимальний ідентифікатор, який уже є в базі, додаємо 1, призначаємо даний ідентифікатор новому коментарю і додаємо в базу даних (рис. 3.9.2).

```
if (!empty($_POST['comment'])) {
    $user = $_SESSION['usernames'];
    $sql_max_id = "SELECT MAX(message_id) as max_id FROM `e-phishing`.feedback";
    $result_max_id = $conn->query($sql_max_id);
    $row_max_id = $result_max_id->fetch_assoc();
    $max_id = $row_max_id['max_id'];

    $sql_user_id = "SELECT userid as user_idd FROM `e-phishing`.users WHERE usernames = '$user'";
    $result_user_id = $conn->query($sql_user_id);
    $row_user_id = $result_user_id->fetch_assoc();
    $user_id = $row_user_id['user_idd'];

    $new_id = $max_id + 1;

    $sql_add_comment = "INSERT INTO `e-phishing`.feedback (message_id, user_id, message)
            VALUES ('$new_id', '$user_id', '$comment')";

    if ($conn->query($sql_add_comment) === TRUE) {
        $_SESSION["success_message"] = "Відгук опубліковано!";
    } else {
        $_SESSION["error_message"] = "Помилка при публікуванні відгуку!";
    }
    header( header: "Location:/comments#testimonial");
    $conn->close();
} else {
    header( header: "Location:/comments#testimonial");
    die($_SESSION["warning_message"] = "Коментар не може бути пустим!");
}
```

Рисунок 3.9.2 – Додавання коментарю до бази даних

Після додавання коментаря маємо його виводити на сайт, тому створюємо дану конструкцію (3.9.3).

```
$current_page = isset($_GET['page']) ? intval($_GET['page']) : 1;
$offset = ($current_page - 1) * $itemsPerPage;
$query = "SELECT f.`message` AS message_rows, f.`message_id`, u.`usernames` FROM `e-phishing`.`feedback` f
    JOIN `e-phishing`.`users` u ON f.`user_id` = u.`userid` ORDER BY f.`message_id`
    DESC LIMIT $offset, $itemsPerPage";
$result_reviews = $conn->query($query);

if ($result_reviews) {
    while ($row = $result_reviews->fetch_assoc()) {
        echo "<div class='testimonial' id='useComment'>
                    <blockquote>
                        <p>“{$row['message_rows']}”</p>
                        <p class='author'> <i class='icon-user'></i>
                            <cite>{$row['usernames']}";
                            if($row['usernames'] == $my_name){
                            echo "<form class='top-right-reload' action='/include/comment/delete-my-comment.php' method='POST'>
                                    <button class='p-1 m-0 fs-6 btn color-btn' type='submit' name='delete-my-message'
                                    value='{$row['message_id']}'>
                                      <i class='icon-trash2'></i> Видалити
                                  </button>
                                  </form>
                            </cite>";
                            }
                            echo "
                        </p>
                    </blockquote>
                </div>";
```

Рисунок 3.9.3 – Вивід коментарів на сайт

Форма з коментарями на сайті виглядає таким чином (рис. 3.9.4).

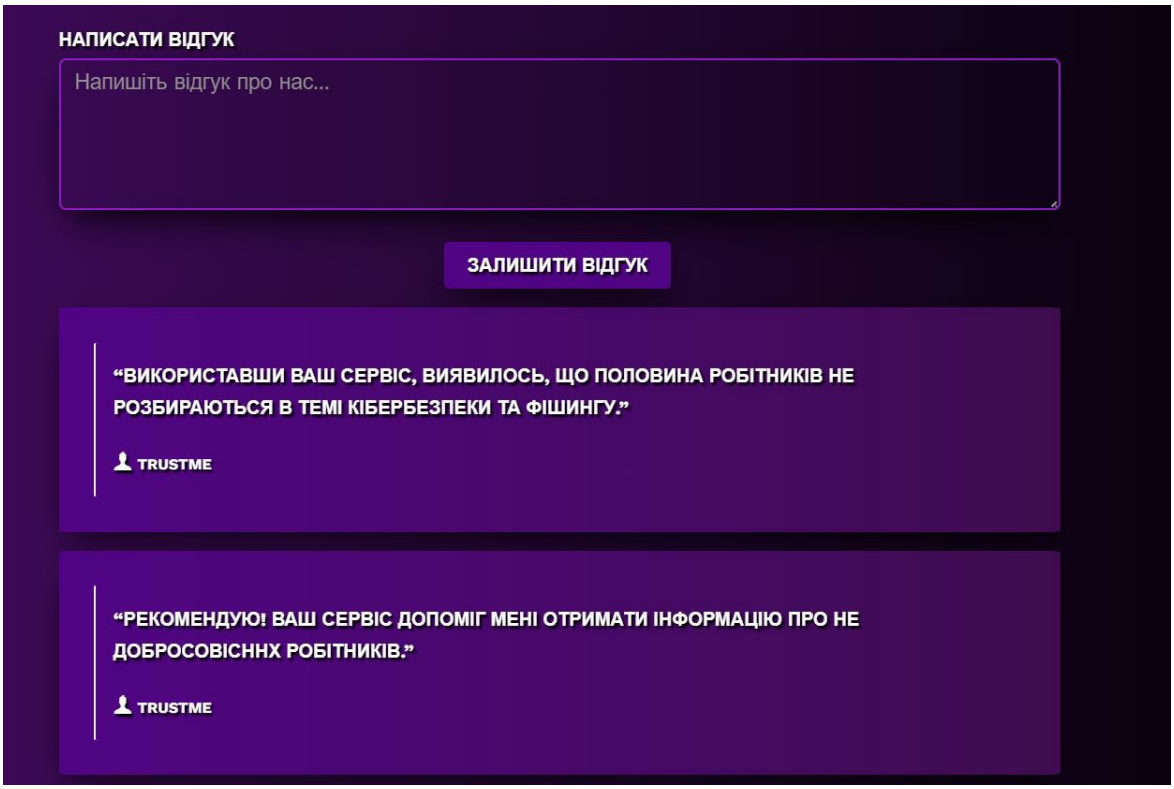


Рисунок 3.9.4 – Частина сторінки «Відгуки»

3.10 ПРИКЛАД РОБОТИ

Створивши повністю систему можна перевірити її функціонал. Перевіримо тільки головний функціонал, тобто відправимо посилання на сторінці «Відправка листа», зімітуємо перехід по ньому, введемо якісь дані та перевіримо на сторінці «Мої Листи». Переглянемо чи в статистику додались відправлені листи, переходи по посиланню та отримані дані.

Почнемо, зареєструємось на сайті та зайдемо в акаунт (рис. 3.10.1-3.10.3).

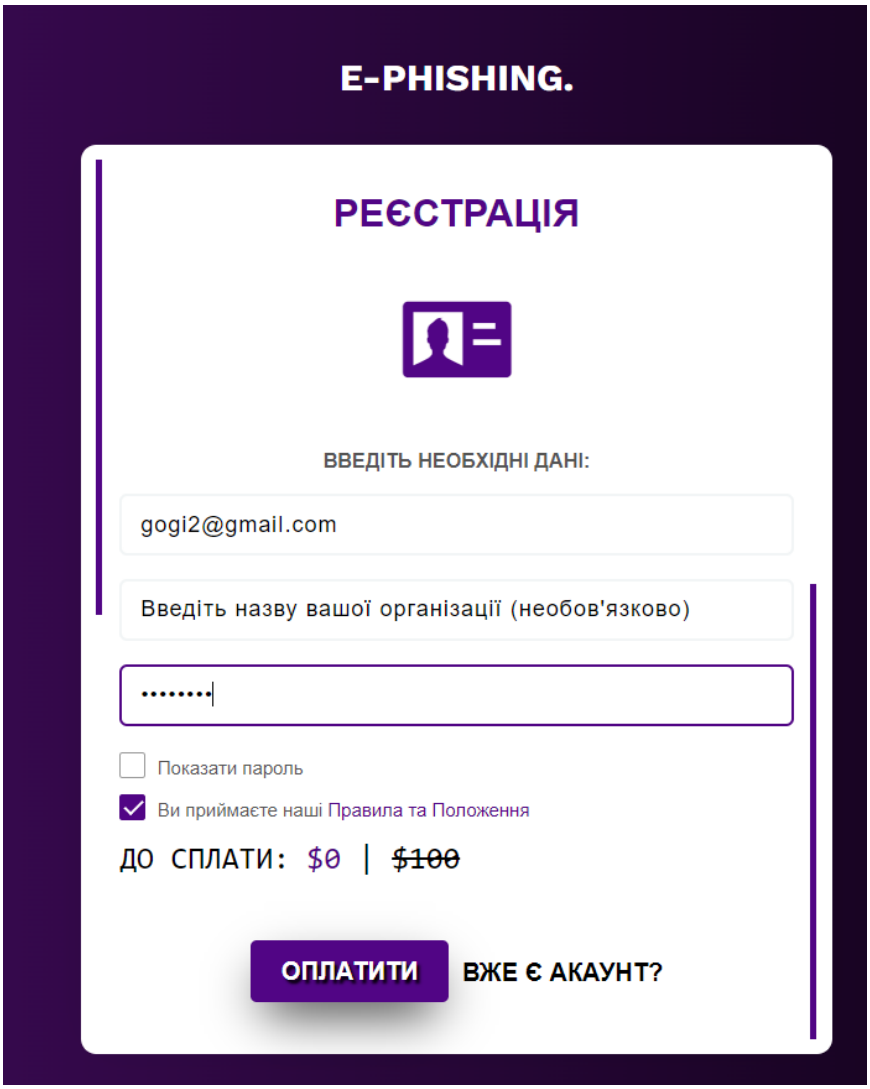


Рисунок 3.10.1 – Реєстрація нового акаунту

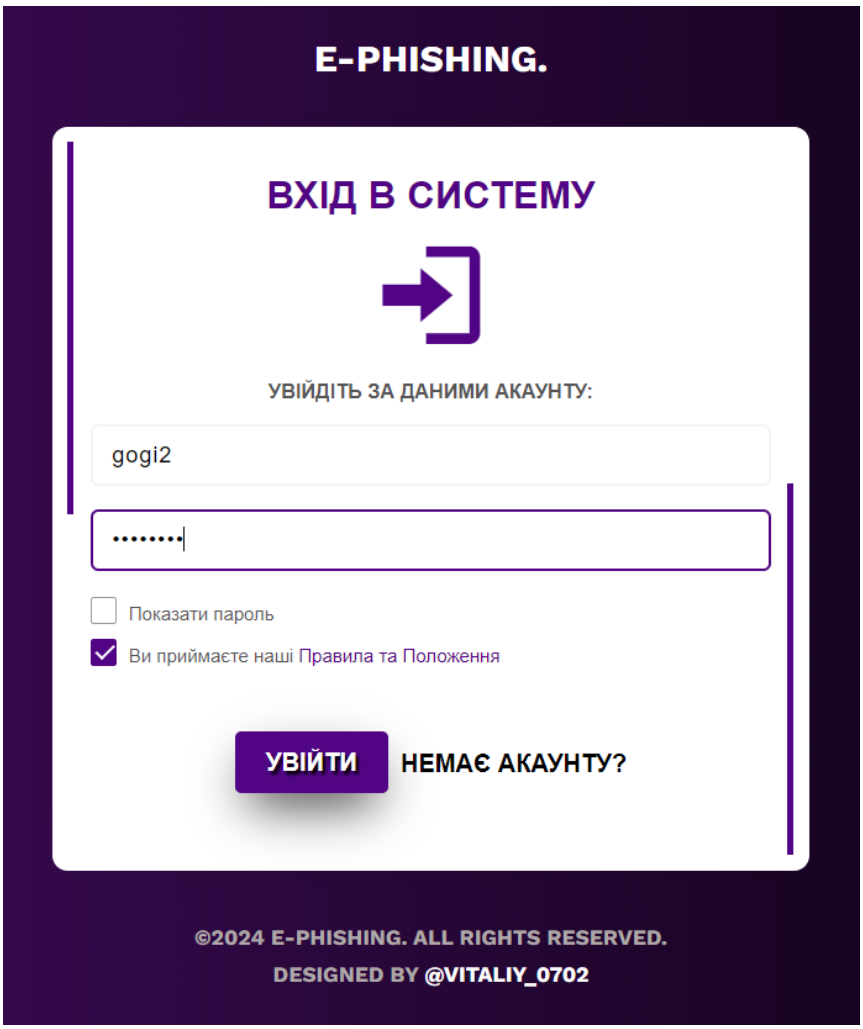


Рисунок 3.10.2 – Вхід в акаунт

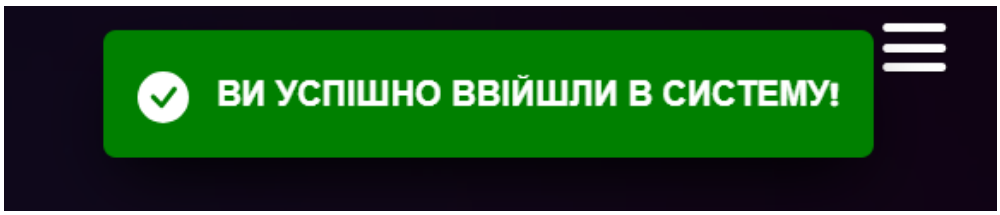


Рисунок 3.10.3 – Підтвердження входу

Далі переходимо на вкладинку «Відправка листа», вибираємо «Кабінет СумДУ» і формуємо посилання (рис. 3.10.4-3.10.5).

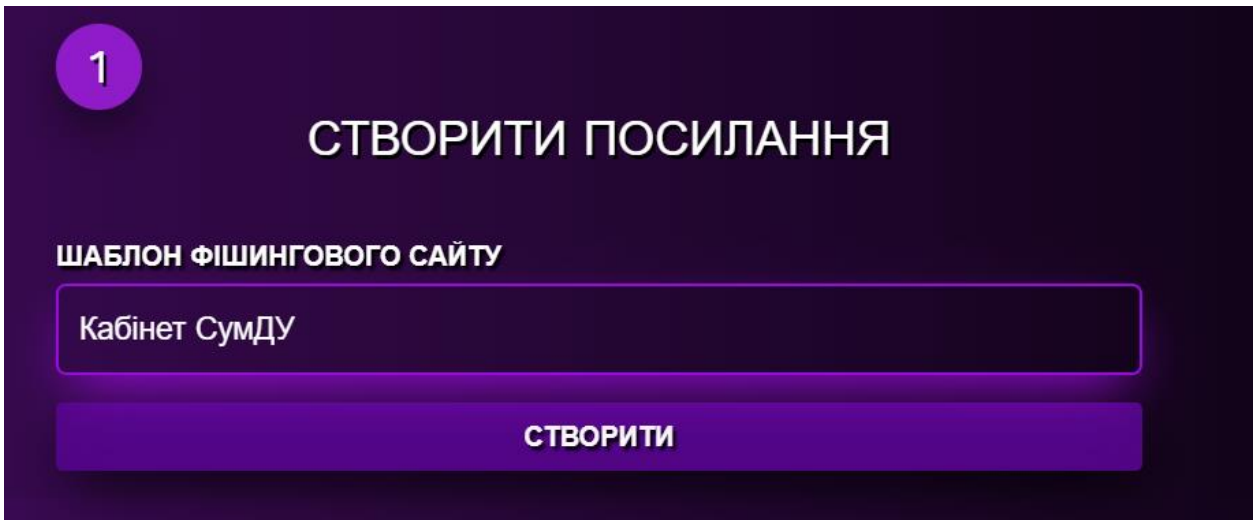


Рисунок 3.10.4 – Створення посилання

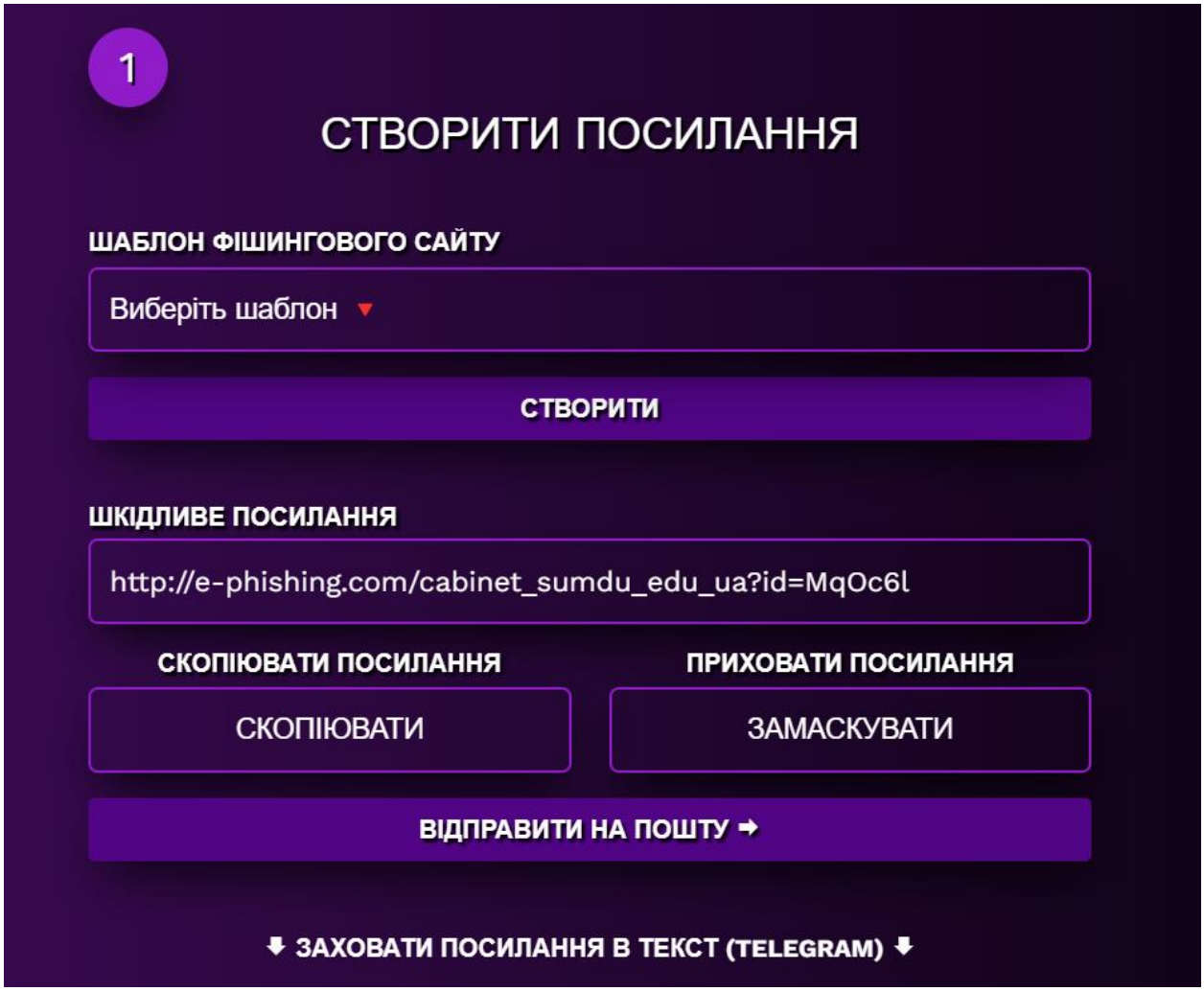


Рисунок 3.10.5 – Посилання створено

Після того як створили посилання, запам’ятовуємо ідентифікатор, що йде в посиланні після id=. Тиснемо кнопку «Відправити на пошту» та вводимо любий email, у даному випадку test@test.com та відправляємо повідомлення (рис. 3.10.6-3.10.7).

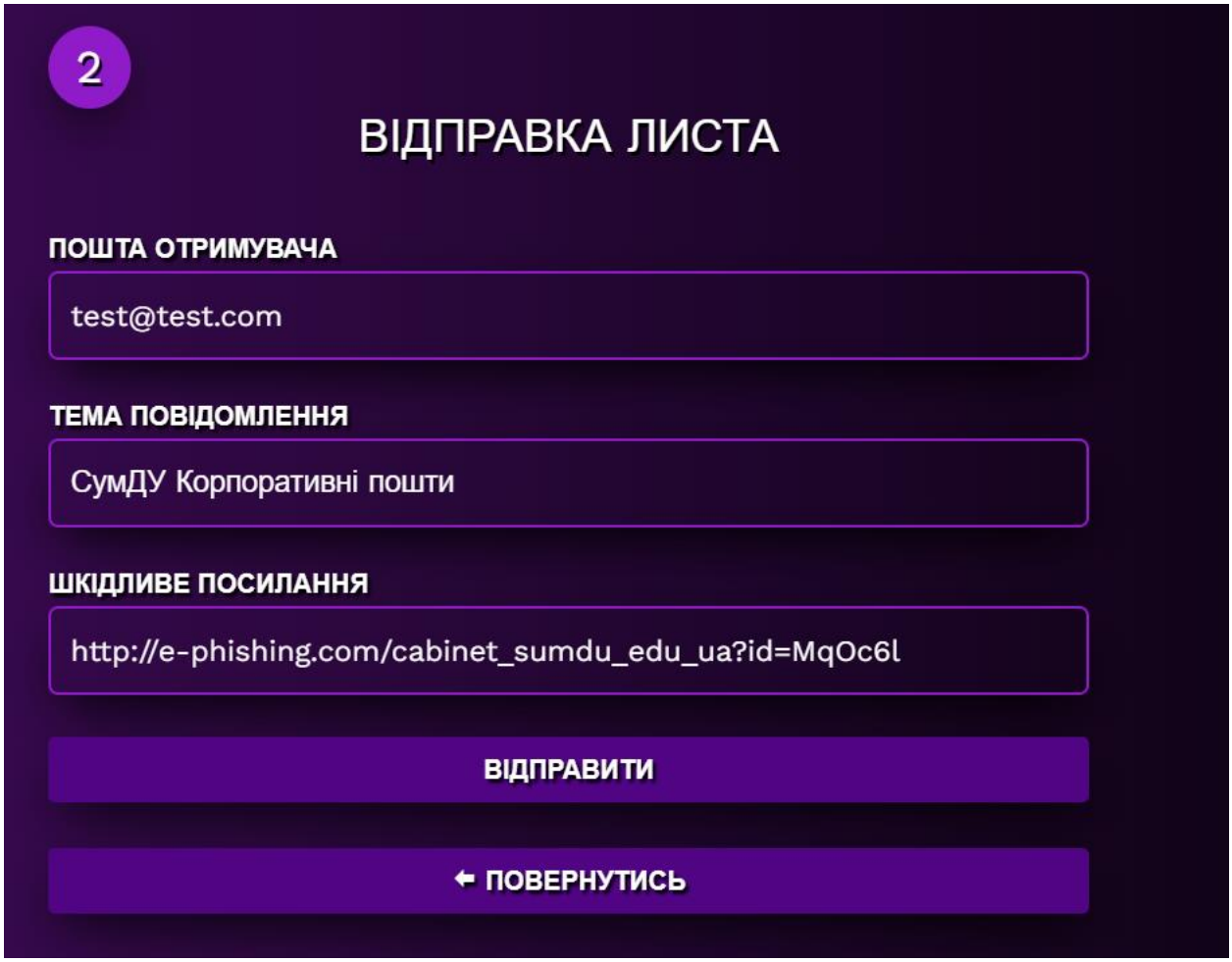


Рисунок 3.10.6 – Відправляємо повідомлення

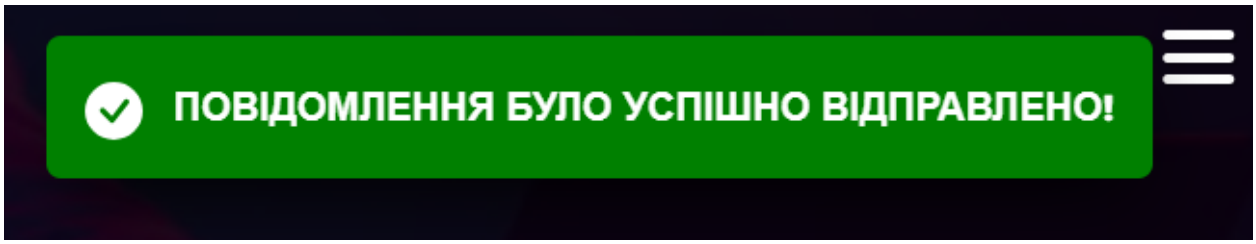


Рисунок 3.10.7 – Підтвердження відправлення повідомлення

Оскільки до сайту не підключений SMTP сервер всі повідомлення зберігаються у файл, перевіримо наявність відправленого повідомлення (рис. 3.10.8).

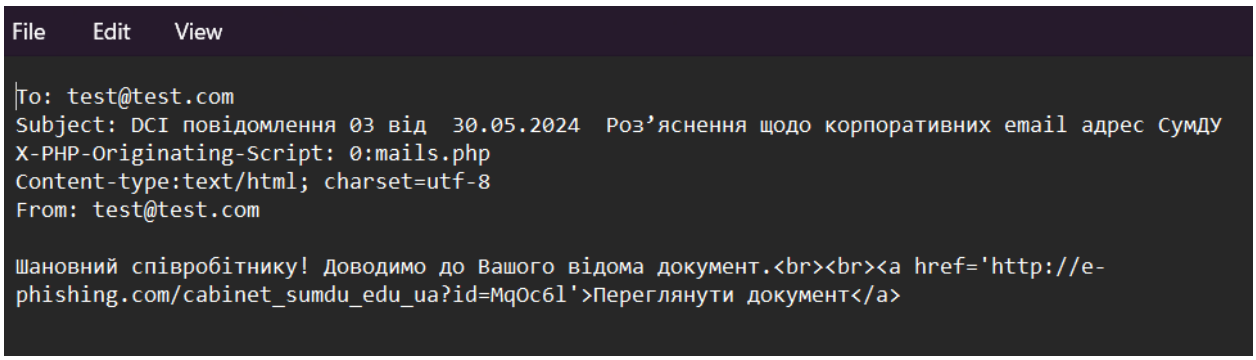


Рисунок 3.10.8 – Повідомлення успішно доставлено

Переходимо на сторінку «Мої Листи», і вибираємо ідентифікатор, який раніше запам’ятовували, дивимось результат (рис. 3.10.9-3.10.10).

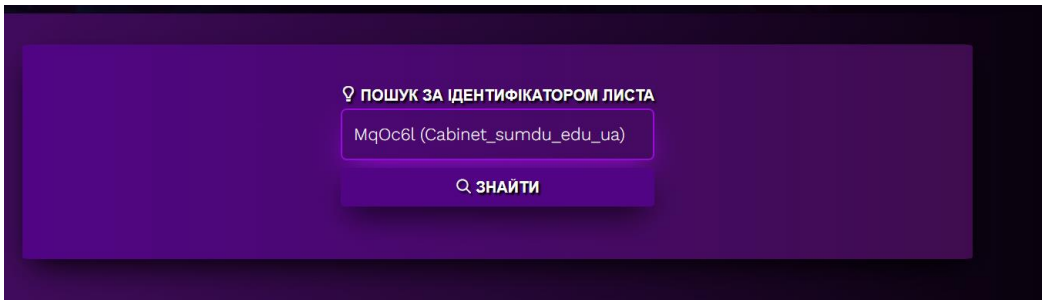


Рисунок 3.10.9 – Вибір ідентифікатору

Рисунок 3.10.10 – Результат пошуку

Як бачимо на рис. 3.10.10 повідомлення ми відправили, але переходів та отриманих даних немає, оскільки ми на локальному сервері і пошта не справжня, імітуємо перехід по посиланню і вводимо тестові дані (рис. 3.10.11).

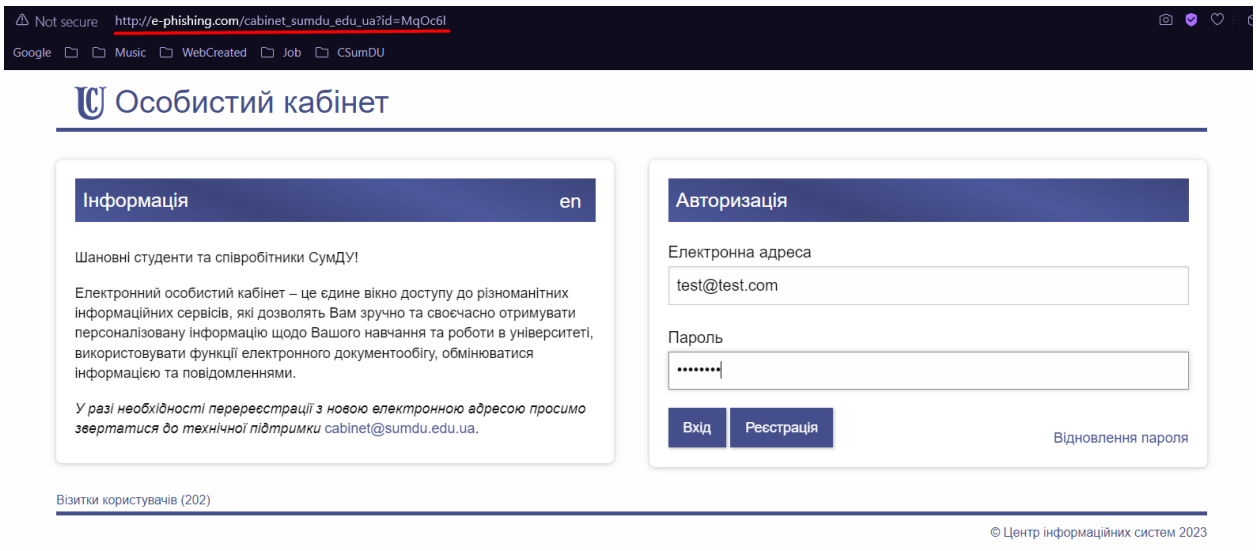


Рисунок 3.10.11 – Перехід по посиланню і введення тестових даних

Після натискання кнопки «Вхід» нас перенаправляє на сторінку справжнього кабінету СумДУ із спробою входу в нього за введеними даними, закриваємо дану сторінку, вона нам не знадобиться. Повертаємось на сервіс E-PHISHING та оновлюємо сторінку (рис. 3.10.12).

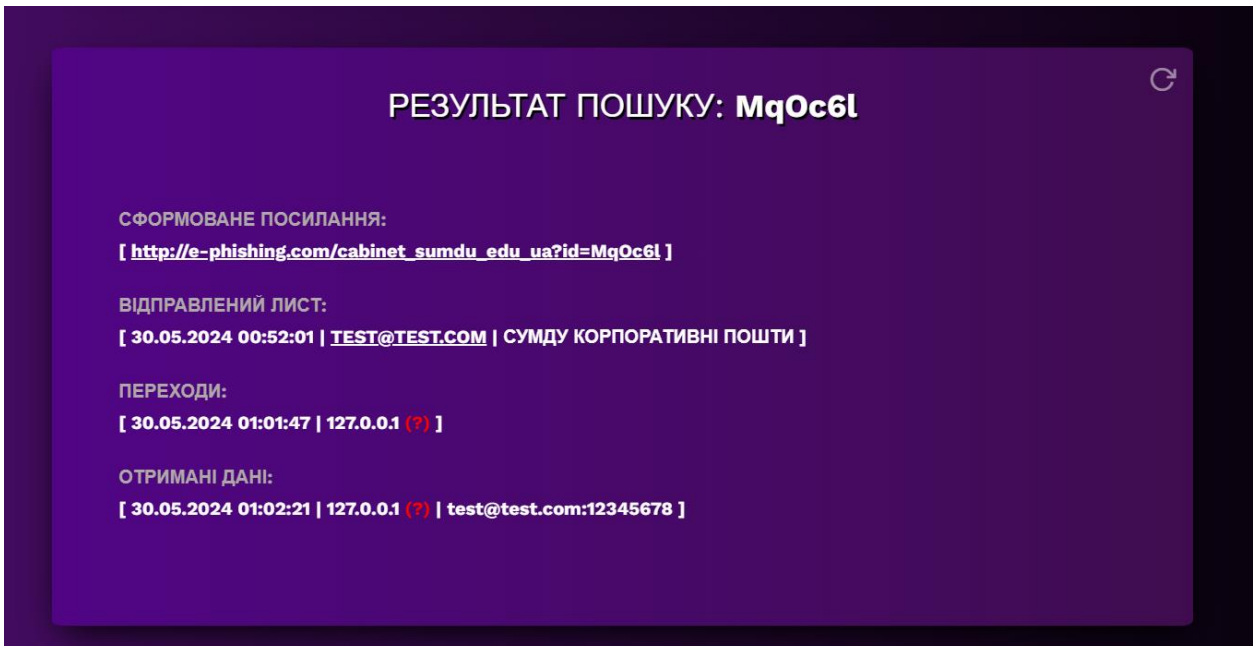


Рисунок 3.10.12 – Переходи і отримані дані зафіксовані

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи бакалавра було розроблено інформаційна система для оцінки рівня кібербезпеки серед робочого персоналу.

Була проведено огляд сучасних технологій та методик рівня оцінки кібербезпеки на підприємствах. У результаті чого було визначено актуальні напрямки та мета роботи.

Було проведено огляд інформаційних систем, які дозволяють проводити оцінку рівня кібербезпеки, та обрано технології для розробки інформаційної системи, а саме: HTML, CSS, PHP та MySQL.

Під час виконання практичної частини роботи було реалізовано інформаційну систему , яка дозволяє провести практичну перевірку рівня кібербезпеки.

Було реалізовано механізми: доступ через веб-інтерфейс, відправка «фішингових» посилань, моніторинг переходів по посиланню, фіксація отриманих даних та формування загальної статистики.

СПИСОК ЛІТЕРАТУРИ

1. PHP: What is PHP? - Manual. PHP: Hypertext Preprocessor. URL: https://www.php.net/manual/en/intro-whatis.php
2. HTML basics - Learn web development | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics
3. Customize. Bootstrap · The most popular HTML, CSS, and JS library in the world. URL: https://getbootstrap.com/docs/5.3/customize/overview/
4. Getting Started with MySQL. MYSQL :: Getting Started. URL: https://dev.mysql.com/doc/mysql-getting-started/en/
5. PHP MySQL Connect to database. W3Schools Online Web Tutorials. URL: https://www.w3schools.com/php/php_mysql_connect.asp
6. Documentation. phpMyAdmin. URL: https://www.phpmyadmin.net/docs/
7. Open Server Panel / Локальний веб-сервер для Windows / Скачать WAMP сервер. Open Server Panel / Локальний веб-сервер для Windows / Скачать WAMP сервер. URL: https://ospanel.io/
8. Perform - performance management для компанії та співробітника. All-in-One HR platform PeopleForce. URL: https://peopleforce.io/uk/products/peopleperform
9. Knowb4. Про продукт, інтеграцію. *Автоматизуй свій бізнес: програмні продукти, інтеграції, інтегратори*. URL: https://hellip.com/ua/product/knowbe4.html
10. Phishing Protection Solutions | Cofense Email Security. *Cofense*. URL: https://cofense.com/
11. Wombat Security Technologies. *ROI4CIO*. URL: https://roi4cio.com/catalog/en/supplier-vendor/wombat-security-technologies

12. homepage. *Terranova Security | Partner of Choice in Security Awareness*. URL: https://www.terranovasecurity.com/
13. Okta - Управління Ідентифікацією | Офіційний Партнер - Cloudfresh. *Cloudfresh*. URL: https://cloudfresh.com/ua/produkty/okta/

ДОДАТОК А. ПРОГРАМНИЙ КОД ВІДПРАВКИ «ФІШИНГОВОГО» ПОВІДОМЛЕННЯ НА ПОШТУ

```php
<?php
require_once __DIR__ . '/../user-audit/user-login.php';
require_once __DIR__ . '/../db/connect.php';

if (!empty($_POST)) {
    $link = $_POST['link'];
    $email = $_POST['email'];
    $theme = $_POST['theme'];

    $link = htmlspecialchars($link, ENT_QUOTES, 'UTF-8');
    $email = htmlspecialchars($email, ENT_QUOTES, 'UTF-8');
    $theme = htmlspecialchars($theme, ENT_QUOTES, 'UTF-8');

    $link = $conn->real_escape_string($link);
    $email = $conn->real_escape_string($email);
    $theme = $conn->real_escape_string($theme);

    $link = urldecode($link);
    $email = urldecode($email);
    $theme = urldecode($theme);

    $link = trim($link);
    $email = trim($email);
    $theme = trim($theme);

    $my_name = $_SESSION['usernames'];

     $sql_user_id = "SELECT userid FROM `e-phishing`.users WHERE usernames =
'$my_name'";
    $result_user_id = $conn->query($sql_user_id);
    $row_user_id = $result_user_id->fetch_assoc();
    $user_id = $row_user_id['userid'];

       $sql_max_id = "SELECT MAX(email_message_id) as max_id FROM `e-
phishing`.email_message";
    $result_max_id = $conn->query($sql_max_id);
    $row_max_id = $result_max_id->fetch_assoc();
    $max_id = $row_max_id['max_id'];
    $new_id = $max_id + 1;

    date_default_timezone_set('Europe/Kiev');
    $currentDateTime = date("Y-m-d H:i:s");

     $sql_type = "SELECT message_id, message_subject, message_text, message_btn
FROM `e-phishing`.message_type WHERE message_name = '$theme'";
    $result_message_type = $conn->query($sql_type);
```

```
    if ($result_message_type) {
        $row_message_type = $result_message_type->fetch_assoc();

        if ($row_message_type) {
                        list($first_subject,  $second_subject)  =  explode('@',
$row_message_type['message_subject']);
            $message_id = $row_message_type['message_id'];
                $message_sub  =  $first_subject  .  "  "  .  date("d.m.Y")  .  "  "  .
$second_subject;
            $message_text = $row_message_type['message_text'];
            $message_btn = $row_message_type['message_btn'];
            $btn = "<a href='$link'>$message_btn</a>";

                    $sql  =  "INSERT  INTO  `e-phishing`.email_message  VALUES
('$new_id','$user_id','$link', '$email', '$message_id', '$currentDateTime')";
            $result_sql = $conn->query($sql);
            if (!$result_sql) {
                $_SESSION["error_message"] = "Помилка при зберіганні даних!";
            }

            $send = mail($email, $message_sub, $message_text . "<br><br>" . $btn,
                "Content-type:text/html; charset=utf-8\r\nFrom: $email");

                  $_SESSION["success_message"]  =  "Повідомлення  було  успішно
відправлено!";
        } else {
            $_SESSION["error_message"] = "За вибраною темою дані відсутні!";
        }
    } else {
        $_SESSION["error_message"] = "Помилка запиту: " . $conn->error;
    }
} else {
    $_SESSION["error_message"] = "Помилка! Немає даних з форми.";
}
header('Location: /send-mail-page#form-send-mail');
die();
```