Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(освітній рівень)

на тему: "Система захисту WEB-Застосунків на основі двохфакторної автентифікації "

Виконав: студент (ка)

Спеціальності:
125 «Кібербезпека та захист інформації»
(шифр і назва напряму підготовки, спеціальності)
Галанський Ілля Олександрович
підпис (прізвище та ініціали)

Керівник Кульчицький Т.Р.
підпис (прізвище та ініціали)

Нормоконтроль Тимощук Д. І.
підпис (прізвище та ініціали)

Завідувач кафедри Загородна Н.В.
підпис (прізвище та ініціали)

Рецензент
підпис (прізвище та ініціали)

м. Тернопіль – 2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра кібербезпеки
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
«__» _____ 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека та захист інформації
(шифр і назва спеціальності)

Студенту Галанському Іллі Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Система захисту WEB-Застосунків на основі двохфакторної автентифікації

Керівник роботи Кульчицький Тарас Русланович Ph.D в галузі права
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «20» 11 2024 року № 4/7-1112

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи Технічна документація, інтернет-джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. Розділ 1 Аналіз теоретичних аспектів автентифікації у Web-додатках. 1.1 Особливості і необхідність автентифікації в Web-додатках. 1.2 Огляд сучасних методів і технологій перевірки автентичності користувачів в інформаційних системах. 1.3 Дослідження криптографічних протоколів і порівняння різних підходів до автентифікації. Розділ 2 Впровадження та проектування системи двохфакторної автентифікації. 2.1 Вибір інструментів і технологічних рішень для впровадження системи захисту Web-додатків із використання двохфакторної автентифікації. 2.2 Проектування архітектури та алгоритмів функціонування системи.2.3 Реалізація системи подвійної автентифікації та її інтеграція у веб-додатки. Розділ 3 Оцінка Результативності та аналіз функціональності впровадженої системи.3.1Проведення тестувань та аналіз захисних властивостей системи для веб-додатків 3.2Оцінювання ефективності впровадження системи подвійної автентифікації для забезпечення безпеки веб-додатків. Розділ 4 Охорона праці та безпека в надзвичайних ситуаціях. 4.1 Охорона праці 4.2 Безпека в надзвичайних ситуаціях. Висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титулка. 2. Мета, об'єкт дослідження та завдання. 3. Актуальність та основи автентифікації 4. Порівняння методів автентифікації. 5. Вибір технологій для впровадження системи. 6. Архітектура та алгоритм системи 2FA. 7. Функціональність системи 2FA. 8. Тестування захисних властивостей системи. 9. Оцінка ефективності впровадженої системи. 10 Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Г.М., к.т.н., доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М., проректор з адміністративно-господарської роботи та будівництва		

7. Дата видачі завдання 23.09.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	10.10.24	Виконано
2.	Опрацювання джерел про види інформаційних загроз	13.10 – 20.10	Виконано
3.	Вступ та перелік літературних джерел	21.10 – 27.11	Виконано
4.	Оформлення першого розділу	28.10 – 04.11	Виконано
5.	Оформлення другого розділу	05.11 – 12.11	Виконано
6.	Оформлення третього розділу	13.11 – 20.11	Виконано
7.	Оформлення четвертого розділу	21.11 – 28.11	Виконано
8.	Оформлення кваліфікаційної роботи	29.11– 06.12	Виконано
9.	Оформлення висновків	07.12 – 10.12	Виконано
10.	Нормоконтроль	13.12 – 16.12	Виконано
11.	Перевірка на плагіат	16.12 – 17.12	Виконано
12.	Захист кваліфікаційної роботи	23.12.2024	

Студент _____ Галанський І.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Кульчицький Т.Р.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Система захисту WEB-Застосунків на основі двохфакторної автентифікації // ОР «Магістр» // Галанський Ілля Олександрович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБс-61 // Тернопіль, 2024 // С. 78, рис. – 28, табл. – 5 , кресл. – , додат. – 2.

Ключові слова: безпека користувачів, веб-застосунки, 2FA, PHP, OTP, MySql.

Дослідження присвячене розробці системи безпеки веб-застосунків із використанням двофакторної автентифікації (2FA). У роботі детально розглянуто сучасні загрози інформаційній безпеці, методи перевірки автентичності користувачів, а також криптографічні протоколи. Обґрунтовано вибір технологій для розробки, включаючи PHP, MySQL, і протоколи OTP/TOTP.

Запропоновано архітектуру системи захисту, реалізовано базові алгоритми роботи 2FA, включаючи відправлення одноразових паролів через електронну пошту за допомогою бібліотеки PHPMailer. Проведено тестування впровадженої системи, що підтвердило її ефективність у забезпеченні конфіденційності та захисті від несанкціонованого доступу.

ABSTRACT

System for protecting WEB applications based on two-factor authentication // Thesis of educational level "Master"// Illia Halanskyi // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cybersecurity, group СБс-61 // Ternopil, 2024 // P. 78, figs. – 28, tables – 5, drawings – , added. – 2.

Keywords: user security, web applications, 2FA, PHP, OTP, MySql.

The research focuses on the development of a web application security system using two-factor authentication (2FA). The study thoroughly examines modern information security threats, user authentication methods, and cryptographic protocols. The choice of technologies for development, including PHP, MySQL, and OTP/TOTP protocols, is substantiated.

The proposed system protection architecture includes the implementation of basic 2FA algorithms, such as sending one-time passwords via email using the PHPMailer library. The implemented system underwent testing, which confirmed its effectiveness in ensuring confidentiality and protection against unauthorized access.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

OTP	—	One-Time Password
2FA	—	Two-Factor Authentication
MFA	—	Multi-Factor Authentication
SMS	—	Short Message Service
TOTP	—	Time-based One-Time Password
HOTP	—	HMAC-based One-Time Password
ROI	—	Return on Investment
NPV	—	Net Present Value
IRR	—	Internal Rate of Return

ВСТУП

Сьогодні, інтернет став базовою частиною повсякденного життя як для окремих осіб, так і для компаній, забезпечуючи доступ до величезної кількості інформації, послуг і ресурсів. Проте разом із розширенням цифрової присутності зростає і кількість кіберзлочинів, що представляють серйозну загрозу для безпеки особистих даних користувачів. Сфера зламу веб-застосунків є особливо вразливою, оскільки багато користувачів та організацій досі покладаються на застарілі методи автентифікації, такі як паролі. Однак ці стандартні засоби більше не здатні гарантувати надійну безпеку, оскільки кіберзлочинці дедалі частіше знаходять способи обійти їх. Саме тому впровадження двофакторної автентифікації стає актуальним і необхідним заходом для захисту веб-застосунків від несанкціонованого доступу. Цей метод забезпечує додатковий рівень безпеки, знижуючи ризики шахрайства та зловмисних атак.

Актуальність даної роботи обумовлена зростаючими загрозами кіберзлочинності, які можуть завдати значної шкоди як приватним користувачам, так і компаніям. Кількість атак, направлених на отримання доступу до приватної інформації, зростає стрімкими темпами, і їхні наслідки можуть бути руйнівними, включаючи фінансові втрати, репутаційні шкоди і витрати. Оскільки традиційні методи автентифікації, такі як паролі, більше не здатні забезпечити необхідний рівень захисту, необхідно створювати та впроваджувати надійні рішення, такі як двофакторна автентифікація. Це дослідження спрямоване на вирішення цих нагальних проблем кібербезпеки шляхом впровадження ефективної системи захисту. Крім того, зростаюча обізнаність користувачів щодо захисту даних підсилює вимоги до безпеки веб-сервісів, що також робить важливим впровадження сучасних методів автентифікації.

Мета дослідження: впровадження та аналіз системи захисту веб-застосунків на основі двофакторної автентифікації для підвищення їхньої безпеки та захисту конфіденційної інформації користувачів.

Об'єкт дослідження: процес забезпечення безпеки веб-застосунків, який охоплює аналіз потенційних загроз і вразливостей, методів їхньої ідентифікації, виявлення шляхів запобігання несанкціонованому доступу та забезпечення захисту конфіденційної інформації користувачів.

Предмет дослідження: впровадження системи захисту веб-застосунків за допомогою двофакторної автентифікації, зокрема вибір технологій, архітектури та оцінка ефективності запропонованого рішення.

Для досягнення визначеної мети сформульовані такі завдання:

– провести теоретичне дослідження автентифікації в веб-застосунках, включаючи порівняння різних технологій та криптографічних протоколів;
– створити систему двофакторної автентифікації для веб-застосунків із визначенням відповідних технологій та реалізацією;
– провести тестування впровадженої системи, оцінити її ефективність і вплив на рівень безпеки веб-застосунків, а також її економічну доцільність.

Методи дослідження: компіляція (систематизація теоретичної та практичної інформації), практичні дослідження (тестування, експерименти), моделювання (розробка програмних компонентів) та статистичний аналіз (оцінка результатів тестування).

Наукова новизна полягає у розробці архітектури системи двофакторної автентифікації (2FA) на основі протоколу TOTP, що підвищує безпеку веб-застосунків, та інтеграції алгоритмів OTP для зниження ризику компрометації даних.

Практична цінність: впроваджена система може бути використана для посилення безпеки реальних веб-застосунків, надаючи ефективний інструмент для розробників та адміністраторів веб-платформ, які прагнуть забезпечити високий рівень захисту своїх продуктів від сучасних кіберзагроз.

РОЗДІЛ 1 АНАЛІЗ ТЕОРЕТИЧНИХ АСПЕКТІВ АВТЕНТИФІКАЦІЇ У WEB-ДОДАТКАХ

1.1 Особливості і необхідність автентифікації в Web-додатках

Автентифікація – це процедура підтвердження особи користувача або системи, яка намагається отримати доступ до захищених ресурсів чи послуг. Цей механізм дозволяє визначити, чи має суб’єкт відповідні повноваження для доступу до певного ресурсу чи системи [1].

У процесі автентифікації використовуються різні способи підтвердження ідентичності, такі як введення облікових даних (ім’я користувача та пароль), застосування біометричних технологій (наприклад, відбитків пальців чи розпізнавання обличчя), використання токенів, карток доступу чи інших факторів, що підтверджують відповідність заявленій ідентичності.

У сфері інформаційної безпеки аутентифікація відіграє ключову роль у запобіганні несанкціонованому доступу до конфіденційних даних і ресурсів. Для захисту інформації та забезпечення безпеки сучасні системи та веб-застосунки впроваджують різноманітні методи аутентифікації.

Особливості автентифікації в веб-застосунках обумовлені її виконанням через Інтернет, що передбачає обмін даними між користувачем та сервером. Цей процес має враховувати специфіку мережевих протоколів, ризики перехоплення даних, а також необхідність балансування між зручністю користування та рівнем безпеки. Більш детальний аналіз особливостей веб-аутентифікації подано на рисунку 1.1 [1].

У веб-застосунках автентифікація виконується через мережу, де користувач взаємодіє із сервером за допомогою браузера чи іншої клієнтської програми. Цей процес суттєво відрізняється від локальної автентифікації, яка здійснюється безпосередньо на пристрої користувача. Для забезпечення безперервності автентифікації та підтримки сесій веб-застосунки активно використовують cookies. Вони зберігають ідентифікатори сесій і додаткові дані, що дозволяє

користувачеві переміщуватися між сторінками або повторно входити до системи без необхідності повторного введення даних.

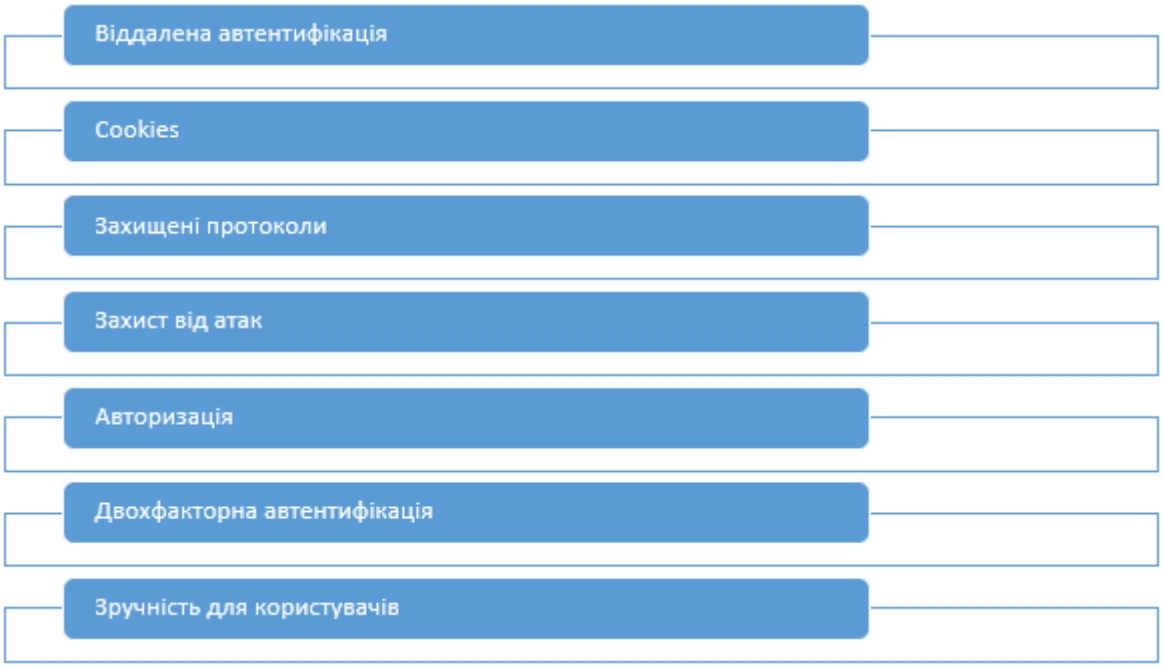

Рисунок 1.1 – Особливості автентифікації в Web-застосунках

Щоб захистити автентифікаційні дані під час їх передачі, застосовуються сучасні протоколи, зокрема HTTPS, які забезпечують шифрування інформації та запобігають її перехопленню. Оскільки веб-застосунки часто стають об’єктом атак, таких як перехоплення сесій, крадіжки паролів або атаки "людина посередник" (man-in-the-middle), впровадження надійних методів автентифікації та дотримання найкращих практик кібербезпеки значно знижують ці ризики.

Після вдалої автентифікації користувачу надається доступ до деяких ресурсів та функцій згідно з його правами й ролями. Для цього важливо, щоб система авторизації була налаштована коректно. Підвищити рівень безпеки допомагає також двофакторна автентифікація, яка передбачає використання двох різних способів підтвердження особи, наприклад, введення пароля та одноразового коду, надісланого на мобільний пристрій.

Однак безпека не повинна суперечити зручності. Процес автентифікації має бути простим і зрозумілим, щоб не відлякувати користувачів. Занадто складні

механізми можуть знижувати залученість до використання веб-застосунку, тому важливо знайти баланс між функціональністю та комфортом для користувача.

Необхідність автентифікації у веб-застосунках зумовлена низкою ключових факторів, які будуть детально проаналізовані на рисунку 1.2 [2].

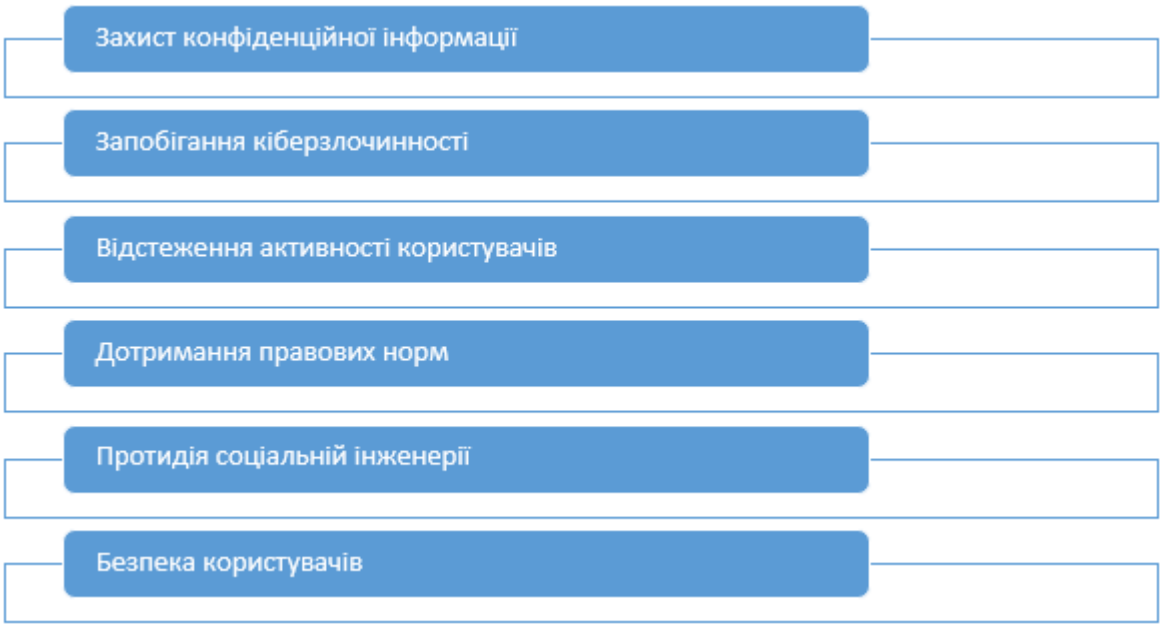

Рисунок 1.2 – Необхідність автентифікації в Web-застосунках

Автентифікація у веб-застосунках має ключову роль у забезпеченні захисту приватної інформації, такої як особисті дані, медичні записи, або ж інформація про фінанси. Вона гарантує, що доступ до цих даних мають лише авторизовані користувачі, що знижує ризик витоку або несанкціонованого використання інформації.

Відсутність належної автентифікації створює сприятливі умови для дій кіберзлочинців, дозволяючи їм отримувати доступ до систем або даних без дозволу. Надійна автентифікація виступає бар’єром проти несанкціонованого доступу та значно знижує ймовірність успішних атак. Водночас, вона також сприяє відстеженню активності користувачів, дозволяючи ідентифікувати їхні дії. Це корисно для аналізу поведінки, ведення логів, а також для виявлення підозрілих чи аномальних активностей, які можуть свідчити про спробу порушення безпеки.

Для веб-застосунків, які працюють із чутливими даними, автентифікація допомагає дотримуватися законодавчих вимог щодо конфіденційності та безпеки. Це особливо важливо в контексті правових норм, які регулюють роботу з особистими або фінансовими даними, таких як GDPR чи HIPAA. Забезпечення відповідності цим нормам гарантує законну діяльність і підвищує довіру до веб-застосунку.

Надійна автентифікація також є ефективним засобом протидії соціальній інженерії. Маніпуляції хакерів, спрямовані на отримання доступу до даних шляхом обману користувачів, стають менш результативними, якщо використовуються сучасні методи, зокрема двофакторна автентифікація. Вона надає додатковий рівень захисту, який ускладнює реалізацію таких атак.

Зрештою, у світі, де кібербезпека стає дедалі важливішою, автентифікація виступає невід'ємним інструментом для надання безпеки користувачеві. Вона сприяє захисту їхніх даних і приватності, що підвищує комфорт і довіру до веб-застосунків, роблячи їх надійними інструментами для щоденного використання.

Автентифікація є критично важливим компонентом веб-застосунків, оскільки забезпечує захист даних, конфіденційність користувачів, протидію кіберзагрозам і відповідність законодавчим стандартам.

Дослідження особливостей та значення автентифікації у веб-застосунках дозволяє виділити низку ключових висновків. Автентифікація виступає основою захисту даних користувачів і ресурсів, забезпечуючи конфіденційність і цілісність інформації. Її надійність напряму впливає на зменшення ризиків, пов'язаних із несанкціонованим доступом, зокрема атак типу крадіжки паролів чи перехоплення сесій.

Обов'язковим елементом безпеки є використання захищених протоколів, таких як HTTPS, які запобігають перехопленню автентифікаційних даних під час їх передачі. У сучасних умовах дедалі більшого поширення кіберзагроз двофакторна автентифікація (2FA) стає необхідною, особливо у випадках роботи з чутливою інформацією. Вона додає подвійний рівень безпеки, роблячи доступ до даних значно складнішим для зловмисників.

Після завершення процесу автентифікації важливу роль відіграє ефективна система авторизації, яка дозволяє належним чином розподілити доступ до функцій та ресурсів веб-застосунку відповідно до прав користувачів. Водночас, автентифікація має бути інтуїтивно зрозумілою і зручною для користувачів, оскільки складність процесу може негативно вплинути на їхній досвід взаємодії.

Високий рівень безпеки автентифікації не лише підвищує довіру користувачів до веб-застосунку, а й забезпечує відповідність законодавчим нормам, що регулюють обробку чутливої інформації. Таким чином, автентифікація залишається ключовим елементом у забезпеченні надійності веб-застосунків, що є важливим як для кінцевих користувачів, так і для розробників.

1.2 Огляд сучасних методів і технологій перевірки автентичності користувачів в інформаційних системах

Рисунок 1.3 – Технології автентифікації користувачів в інформаційних системах

Сучасні системи роботи з інформацією пропонують великий вибір технологій автентифікації, які дозволяють ефективно підтверджувати ідентичність користувачів. Найпоширенішим методом залишаються паролі,

ефективність яких залежить від їхньої складності та унікальності. Однак традиційні паролі дедалі частіше доповнюються або замінюються інноваційними підходами (див. рисунок 1.3). Наприклад, біометричні дані, відбитки пальців, обличчя, голос або радужка ока, дозволяють використовувати унікальні фізичні характеристики людини для ідентифікації. Для їх реалізації потрібні спеціалізовані пристрої [3], що робить метод дуже надійним.

Одноразові паролі (OTP), які генеруються для одноразового використання, є ще одним популярним рішенням, часто застосовуваним у поєднанні з іншими методами, такими як SMS або мобільні додатки. Протоколи на кшталт OAuth і OpenID Connect забезпечують автентифікацію через сторонні платформи, наприклад, Google, спрощуючи процес для кінцевих користувачів.

Фізичні носії, такі як карти доступу чи токени, також широко застосовуються, особливо в корпоративному середовищі. Для забезпечення автентифікації на основі шифрування використовуються цифрові сертифікати та ключі. В організаційних мережах популярним є протокол LDAP (Lightweight Directory Access Protocol), що дозволяє централізовано керувати доступом та автентифікацією.

Двофакторна та багатофакторна автентифікація (2FA/MFA) комбінують різні методи для підвищення рівня безпеки. Наприклад, користувач може одночасно вводити пароль і підтверджувати вхід за допомогою OTP або біометричних даних. Протоколи, як-от SAML (Security Assertion Markup Language), забезпечують єдину автентифікацію між різними додатками.

Нові підходи включають біометричну автентифікацію за допомогою серцевого ритму або фізіологічних характеристик голови, таких як форма вух чи радужка ока. Ці методи перебувають на ранніх етапах впровадження, але обіцяють значно розширити можливості забезпечення безпеки [4, 5, 6]. Додатково існують інноваційні способи введення пароля, наприклад, зворотний пароль, який ускладнює його фіксацію завдяки специфічному способу натискання клавіш.

Таким чином, спектр технологій автентифікації дозволяє підібрати рішення, що найкраще відповідає потребам користувачів і рівню безпеки системи, гармонійно поєднуючи зручність та надійність.

Ці методи можуть працювати як самостійно, так і в поєднанні, залежно від вимог до рівня безпеки. У веб-застосунках найчастіше використовуються паролі, біометрія, OTP, OAuth, двофакторна автентифікація, коди підтвердження та багатофакторна автентифікація. Ці технології надають високий рівень безпеки та зручність для користувачів в онлайн-середовищі.

Пароль — це один із найпоширеніших методів автентифікації, де від користувача потребується ввести секретний текстовий код, який попередньо був збережений в системі.

Переваги:

– зручність для користувачів;

– легкість зміни пароля, якщо він став відомим стороннім особам;

– простота реалізації та використання.

Недоліки:

– вразливість до атак методом перебору;

– потреба в запам'ятовуванні складних паролів, що може бути незручно для користувачів;

– потреба у впровадженні додаткових заходів безпеки, таких як двоетапна перевірка та захист від перехоплення даних (наприклад, за допомогою HTTPS).

Біометричні дані охоплюють унікальні фізичні ознаки користувача, такі як обличчя, відбитки пальців, голос, сканування радужки та інші, які вимірюються і аналізуються за допомогою спеціальних пристроїв або камер.

Переваги:

– забезпечує високий рівень безпеки, оскільки біометричні характеристики не повторюються у різних осіб;

– паролі не потрібно запам'ятовувати.

Недоліки:

– потреба у фізичному обладнанні (сенсори, камери) для зчитування біометричних характеристик;

– неможливість змінити або відновити біометричні дані в разі втрати або компрометації;

– потенційна вразливість до підробки біометричних характеристик.

Одноразові паролі (OTP) — це паролі, що мають обмежений термін дії і можуть бути застосовані тільки один раз. Вони генеруються спеціальними додатками або надсилаються через SMS чи інші засоби комунікації.

Переваги:

– вищий рівень безпеки порівняно з постійними паролями;

– генеруються та використовуються лише в конкретному контексті і обмежений час;

– складно піддаються атакам перебору, оскільки не можна використовувати повторно.

Недоліки:

– потребує інфраструктури для генерації та доставки одноразових паролів;

– необхідність наявності інтернет-з'єднання або мобільного зв'язку для отримання паролів через SMS або додатки.

OAuth — це механізм авторизації, який дає змогу зовнішнім додаткам отримувати доступ до особистих даних користувача на веб-платформах без необхідності вводити пароль. Користувач спочатку автентифікується на спеціальній сторінці, після чого дає дозвіл на доступ до своїх даних, і зовнішній додаток отримує спеціальний токен для подальшої взаємодії з ресурсами користувача. Цей протокол зазвичай застосовується для інтеграції з такими платформами, як соціальні мережі, поштові сервіси та інші онлайн-ресурси[7].

Переваги:

– дає змогу стороннім додаткам отримувати доступ до ресурсів без необхідності ділитися паролем;

– мінімізує ймовірність компрометації пароля.

Недоліки:

– потребує налаштування серверів для обробки авторизації та передачу токенів, що може бути технічно складним;

– вимагає підвищеної уваги до безпеки, оскільки неправильне налаштування може призвести до витоку даних.

OpenID Connect — це додатковий протокол до OAuth, який не лише здійснює авторизацію, а й надає можливість підтвердження особи користувача. Він дозволяє стороннім додаткам отримувати токени, що містять деталі про користувача, як-от ім'я або контактні дані, для забезпечення автентифікації. Цей протокол часто застосовується в системах, які потребують взаємодії з іншими обліковими записами, наприклад, через Google.

Переваги:

– об’єднує авторизацію та автентифікацію в одному протоколі;

– дозволяє безперешкодно використовувати сторонні облікові записи для входу.

Недоліки:

– потребує налаштування з провайдерами OpenID Connect, що може бути складним;

– вимагає підвищеної уваги до безпеки, щоб захистити конфіденційні дані користувача.

Двофакторна автентифікація (2FA) — передбачає використання двох різних методів для підтвердження особистості користувача. Це може бути поєднання чогось, що користувач знає (наприклад, пароль), з чимось, що він має (картка, мобільний пристрій з кодом), або чимось, що його характеризує (біометрія). Після введення пароля, система вимагатиме другого кроку — введення одноразового коду, отриманого через SMS чи генерованого додатком. Такий метод використовується для посилення безпеки, щоб запобігти несанкціонованому доступу, навіть якщо пароль зламали.

Переваги:

– підвищує рівень безпеки, додаючи другий шар підтвердження;

– зменшує шанси на несанкціонований доступ до облікового запису, навіть при зламуванні пароля.

Недоліки:

– може ускладнити процес автентифікації і зайняти більше часу;

– потребує наявності додаткових елементів, таких як мобільний телефон, що можуть бути втрачені або викрадені.

Багатофакторна автентифікація — це метод, що вимагає застосування щонайменше трьох різних факторів для перевірки особистості користувача. Це можуть бути комбінації пароля, одноразового коду, біометрії, токенів чи інших методів. Кожен з етапів включає введення даних з окремого фактору. Цей підхід забезпечує високий рівень захисту та використовується у критичних системах для охорони конфіденційної інформації.

Переваги:

– надання вищого рівня безпеки завдяки багатоступеневій автентифікації;

– ефективний захист важливих даних від несанкціонованого доступу.

Недоліки:

– може вимагати додаткового часу для виконання декількох етапів автентифікації;

– потребує регулярного оновлення та управління кількома методами перевірки.

Коди підтвердження — це унікальні одноразові значення, які вводяться користувачем після основного пароля або інших автентифікаційних даних для завершення перевірки особи. Вони можуть надсилатися через електронну пошту, SMS або створюватися за допомогою спеціалізованих додатків для генерації кодів.

Переваги:

– підвищують безпеку, оскільки потребують додаткового етапу підтвердження;

– простота впровадження та використання для кінцевого користувача;

– можливість додаткового захисту за допомогою інших методів, наприклад, біометрії.

Недоліки:

– залежність від доступності пристроїв або сервісів для отримання коду;

– можливість втрати чи перехоплення коду третіми особами.

Мультифакторна автентифікація (MFA) — це підхід, який передбачає використання кількох незалежних способів підтвердження особи користувача. Цей процес може базуватися на поєднанні інформації, яку користувач знає (наприклад, пароль), володіє (такий як фізичний ключ або смартфон), або є (біометричні дані, включаючи відбитки пальців чи розпізнавання обличчя) [8].

Переваги:

– забезпечує максимальний рівень захисту, завдяки необхідності підтвердження через кілька способів;

– знижує ймовірність несанкціонованого доступу, навіть якщо один з факторів компрометований;

– гнучкість у виборі методів підтвердження залежно від ситуації або потреб.

Недоліки:

– процедура автентифікації може бути складнішою і менш зручною для кінцевого користувача;

– необхідність мати додаткові пристрої чи інструменти, наприклад, токени або мобільні додатки.

Таким чином, було охарактеризовано основні механізми підтвердження ідентичності у веб-застосунках, серед яких використання паролів, біометричних даних, одноразових кодів, протоколів OAuth та OpenID Connect, а також методів двох і багатофакторної автентифікації.

1.3 Дослідження криптографічних протоколів і порівняння різних підходів до автентифікації

Дослідження криптографічних протоколів, які забезпечують ідентифікацію та автентифікацію, має ключове значення для підвищення рівня захисту веб-

застосунків. Нижче наведено огляд основних протоколів, таких як TLS/SSL, Kerberos, OAuth, OpenID Connect, SAML, RADIUS, JWT, з описом їхніх функцій та особливостей (див. рисунок 1.4) [9].

Рисунок 1.4 – Популярні криптографічні протоколи автентифікації

Під час вибору методу автентифікації слід враховувати різноманітні аспекти та параметри, які змінюються залежно від особливостей веб-застосунку та його функціональних вимог. Нижче наведено основні чинники, що впливають на цей вибір:

- рівень забезпечення безпеки;
- комфорт та простота використання для кінцевих користувачів;
- необхідність у додатковій інфраструктурі;
- можливість забезпечення доступу з будь-якої точки;
- потенційні загрози та ризики для безпеки;
- економічні аспекти впровадження;
- зручність налаштування та подальшого обслуговування;
- відповідність нормативним та правовим вимогам;
- сумісність із наявними системами та платформами;
- кількість і тип облікових записів користувачів.

Створена таблиця 1.1, яка містить порівняння різних методів автентифікації за зазначеними критеріями [10].

Таблиця 1.1 – Порівняння різних методів автентифікації

Критерій	Пароль	Біометричні дані	Одноразові паролі	OAuth і OpenID Connect	Двофакторна автентифікація (2FA)	Коди підтвердження	Мультифакторна автентифікація (MFA)
Рівень безпеки	Середній	Високий	Високий	Залежить від реалізації	Високий	Високий	Дуже високий
Зручність для користувачів	Зручний	Залежить від методу	Залежить від засобу	Залежить від реалізації	Залежить від вибору факторів	Залежить від засобу	Залежить від вибору факторів
Потреба в інфраструктурі	Немає	Наявність біометричних сенсорів	Немає	Немає	Наявність мобільного телефону	Немає	Наявність мобільного телефону
Забезпечення доступу здалеку	Так	Так	Так	Так	Так	Так	Так
Загрози безпеці	Підвищені	Залежать від реалізації	Високі	Залежать від реалізації	Залежать від налаштувань	Високі	Високі
Вартість та бюджет	Низька	Залежить від обладнання	Низька	Залежить від реалізації	Низька	Низька	Залежить від вибору факторів
Легкість використання та налаштування	Зручна	Залежить від реалізації	Залежить від засобу	Залежить від реалізації	Залежить від налаштувань	Залежить від засобу	Залежить від вибору факторів
Законодавчі та регуляторні вимоги	Залежить	Залежать від області	Залежать від області	Залежать від області	Залежать від області	Залежать від області	Залежать від області
Інтеграція з існуючими системами	Спрощена	Залежить від інтеграції	Спрощена	Спрощена	Спрощена	Спрощена	Спрощена

Для оцінки та вибору технологій автентифікації слід враховувати кілька важливих аспектів, які впливають на рівень безпеки, зручність використання, потребу в інфраструктурі та відповідність законодавчим вимогам.

Одним із ключових параметрів є рівень безпеки, де найкращі результати демонструє мультифакторна автентифікація (MFA), що вимагає використання кількох факторів для підтвердження особи. Високий рівень захищеності також забезпечують біометричні методи та одноразові паролі. Менш надійними є традиційні паролі та протоколи OAuth/OpenID Connect, однак їх можна посилити додатковими заходами [8].

Другим важливим аспектом є зручність для користувачів. У цьому контексті паролі та OAuth/OpenID Connect виграють завдяки своїй легкості та розповсюдженості. Біометричні методи та OTP зазвичай також зручні, але багато залежить від конкретної реалізації. Водночас MFA та коди підтвердження можуть вимагати більше дій від користувачів, що інколи знижує їхню популярність.

Потреба в інфраструктурі також є значущим фактором. Наприклад, паролі та OAuth/OpenID Connect не вимагають спеціального обладнання, тоді як біометричні методи потребують сенсорів, а MFA або OTP можуть вимагати генераторів одноразових паролів або смартфонів для підтримки мобільних додатків.

У питанні забезпечення віддаленого доступу більшість методів є ефективними. Проте MFA з використанням мобільних додатків вважається одним із найбільш зручних для віддалених користувачів, оскільки дозволяє ефективно підтримувати безпеку на відстані.

Ризики, пов’язані із загрозами безпеці, також варто враховувати. Кожен метод має свої потенційні слабкі місця, і для зменшення цих ризиків потрібні правильне налаштування та контроль.

Вартість та бюджет впливають на вибір методу. Паролі та OAuth/OpenID Connect є найдешевшими у впровадженні. Біометричні системи, навпаки,

потребують значних витрат на обладнання, а MFA або OTP можуть вимагати витрат на мобільні додатки чи апаратні токени.

Ще одним критерієм є легкість використання та налаштування. Прості у впровадженні та налаштуванні паролі та OAuth/OpenID Connect, тоді як біометричні методи та MFA можуть вимагати значних ресурсів для інтеграції.

Не слід забувати і про законодавчі та регуляторні вимоги, які часто визначають вибір технології автентифікації в залежності від галузі діяльності, особливо якщо йдеться про роботу з чутливими даними.

Нарешті, важлива інтеграція з існуючими системами. Найлегше інтегруються паролі, OAuth/OpenID Connect та OTP, тоді як біометрія або MFA можуть потребувати додаткових зусиль для впровадження у наявну інфраструктуру.

У підсумку, кожен із методів має свої переваги й обмеження, тому вибір технології автентифікації залежить від конкретних потреб системи, її користувачів і доступних ресурсів.

РОЗДІЛ 2 ВПРОВАДЖЕНЯ ТА ПРОЕКТУВАННЯ СИСТЕМИ ДВОХФАКТОРНОЇ АВТЕНТИФІКАЦІЇ

2.1 Вибір інструментів і технологічних рішень для впровадження системи захисту Web-додатків із використанням двохфакторної автентифікації

Для впровадження системи захисту веб-додатків із двофакторною автентифікацією доцільно використовувати клієнт-серверну модель і розробити веб-інтерфейс.

Планування системи потребує ретельного вибору технологій, включаючи мову програмування та систему управління базами даних (СУБД), які відповідатимуть поставленим вимогам. Вибір обумовлений необхідністю забезпечення відповідності ключовим критеріям [11].

По-перше, система повинна мати можливість роботи на різних платформах. Це забезпечить інтеграцію з іншими компонентами, які можуть використовувати різноманітні операційні системи чи середовища.

Гнучкість також є важливим фактором, оскільки це дозволяє вносити зміни чи оновлення без значних витрат ресурсів і часу. Особливо це важливо в динамічних умовах, коли вимоги можуть швидко змінюватися.

Захист інформації, як під час обробки, так і при зберіганні, є неймовірно важливим для надання безпеки конфіденційності та цілісності даних. Технології, що використовуються, повинні мати вбудовані механізми шифрування та можливості боротьби із загрозами безпеці.

Мова програмування має бути широкофункціональною, тобто підтримувати різноманітні інструменти та бібліотеки, необхідні для вирішення як поточних, так і потенційних завдань.

Універсальність клієнтської частини системи забезпечує доступ до неї через різні браузери та пристрої, що є важливим для зручності користувачів.

Використання технологій із відкритим вихідним кодом забезпечує спрощення процесів оновлення, виправлення помилок і знижує залежність від сторонніх розробників.

Простота реалізації сприяє швидкому та ефективному розробленню додатка, що особливо важливо у проєктах із обмеженими строками або ресурсами.

Нарешті, економічна доцільність включає в себе не лише початкові витрати на впровадження системи, але й витрати на її подальшу підтримку та оновлення.

Таким чином, вибір технологій для системи повинен базуватися на відповідності вищезазначеним критеріям, що забезпечить створення ефективного, безпечного та економічно вигідного рішення.

Для порівняння мов програмування була створена таблиця за наведеними вище аспектами (див. таблицю 2.1). У ній представлені найпопулярніші мови програмування для Web-додатків станом на 2024 рік, оцінені за п'ятибальною шкалою. PHP виділяється серед інших конкурентних продуктів завдяки наступним перевагам:

- висока продуктивність;
- наявність інтерфейсів для взаємодії з різними системами баз даних;
- безкоштовне розповсюдження;
- наявність вбудованих бібліотек пов'язаних з Web;
- легкість у навчанні та використанні;
- доступність вихідного коду;
- портативність.

Таблиця 2.1 – Порівняння мов програмування

Критерії\Мови	PHP	CSS/HTML	Python	Ruby	JavaScript	Java
Простота програмної реалізації	5	4	4	3	5	4
Кросплатформеність	4	4	5	5	4	5
Гнучкість	5	4	4	3	3	4
Вартість	5	5	3	4	4	3
Безпека	5	4	4	4	5	5
Інтеграція	5	3	4	3	4	4
Відкритість вихідного коду	5	4	4	3	5	4
Універсальність клієнт-додатків	4	5	5	5	3	4
Загальна оцінка	38	33	29	30	33	33

Для впровадження системи захисту веб-додатків із двофакторною автентифікацією, впровадженої у форматі веб-програми, оптимальним вибором є використання мови програмування PHP, а також HTML і JavaScript для реалізації інтерфейсу [12].

До системи керування базами даних висуваються важливі вимоги, які визначають її придатність для використання в межах розроблюваної системи.

Перш за все, СУБД повинна бути сумісною з технологіями, які застосовуються в розробці, наприклад, PHP. Це забезпечує можливість легкої інтеграції та взаємодії між компонентами системи.

Кросплатформність є наступною ключовою вимогою. Система управління базами даних повинна підтримувати роботу на різних операційних системах, що сприятиме її гнучкості та адаптації до різних середовищ.

Зручність у використанні та простота налаштування також мають велике значення. Це дозволяє ефективно впроваджувати систему та знижує витрати на її конфігурування, особливо у проєктах із обмеженими ресурсами.

Популярність і широке поширення СУБД забезпечують доступ до великої кількості документації, інструментів і спільноти розробників, що спрощує її використання та обслуговування. Використання маловідомих або вузькоспеціалізованих СУБД може ускладнити подальше масштабування системи й унеможливити отримання оперативної технічної підтримки.

Нарешті, високий рівень безпеки є обов'язковою вимогою для захисту даних від несанкціонованого доступу та забезпечення їх цілісності.

З урахуванням цих критеріїв вибір СУБД має бути спрямований на забезпечення її ефективної роботи в межах системи, її адаптованості до потреб проєкту та довгострокової стійкості.

Для того щоб полегшити вибір, була створена таблиця яка містить порівняння СУБД за зазначеними критеріями (див. таблицю 2.2) [13].

Таблиця 2.2 – Порівняння СУБД

Аспекти\СУБД	PostgreSQL	MySQL	MS SQL Server	Oracle	DB2
Транзакційна підтримка	3	5	4	4	5
Підтримка зовнішніх ключів	5	4	5	3	5
Точна робота з українською мовою	3	4	5	4	5
Наявність інструменту керування з графічним інтерфейсом	4	5	4	4	4
Здатність доступу до інформації за допомогою мови запитів SQL	3	5	4	3	3
Здатність резервного копіювання БД	5	4	5	5	4
Повна сумісність із обраною сферою розробки (PHP)	4	5	3	2	5
Кросплатформеність СУБД	3	5	5	3	5
Легкість у застосуванні та при впровадженні	3	5	3	4	5
Поширення та популярність СУБД	3	5	5	5	4
Безпека	4	4	4	5	3
Загальна оцінка	40	51	48	42	48

Беручи до уваги всі зазначені критерії, для роботи з базами даних було обрано MySQL — популярну багатокористувацьку систему управління базами даних з відкритим вихідним кодом. Основними перевагами MySQL є її гнучкість, масштабованість і широкий набір функціональностей, включаючи підтримку транзакцій, складних запитів, і розширень. Хоча MySQL може потребувати більше ресурсів, ніж інші СУБД, її стабільність і функціональність роблять її ідеальним вибором для проєктів, які потребують обробки великих обсягів даних.

У поєднанні з PHP та MySQL дозволяє створити потужне і гнучке середовище розробки, забезпечуючи високу продуктивність і підтримуючи кросплатформенність.

2.2 Проектування архітектури та алгоритмів функціонування системи

Двофакторна автентифікація (2FA) є сучасним і ефективним інструментом для забезпечення підвищеної безпеки користувачів у різних інформаційних системах. Її основний принцип полягає у використанні двох незалежних факторів для підтвердження особи: того, що користувач знає (наприклад, пароль), і того, чим він володіє або що він має доступ (наприклад, код чи пристрій). Це зменшує ризики, пов'язані з несанкціонованим доступом, навіть якщо один із факторів буде скомпрометований. Виділимо одні з основних методів 2FA [14].

Перш за все, паролі, прив’язані до часу (TOTP), забезпечують високий рівень захищеності завдяки своїй залежності від даних поточного часу та попередньо встановленого секретного ключа. Такий код генерується динамічно і має короткий термін дії, що робить його непридатним для використання після завершення цього часу.

Біометрична автентифікація є одним із найбільш інноваційних підходів у межах 2FA. Вона ґрунтується на фізичних або поведінкових характеристиках користувача, таких як відбитки пальців, риси обличчя чи голос. Завдяки своїй унікальності цей метод забезпечує надійну перевірку особи.

Коди підтвердження через SMS залишаються поширеним методом 2FA. Після введення пароля користувач отримує одноразовий код на мобільний телефон. Щоб завершити процес авторизації, цей код необхідно ввести в системі. Такий підхід є простим у використанні, хоча й має певні недоліки, пов’язані з можливими ризиками перехоплення повідомлень.

Одноразові паролі (OTP) генеруються спеціальними додатками або апаратними токенами. Ці паролі діють лише протягом обмеженого часу і використовуються для додаткової перевірки. Цей метод забезпечує високий рівень безпеки завдяки унікальності кожного створеного пароля.

Останнім методом є автентифікація за місцезнаходженням, яка обмежує доступ лише до певних географічних зон. Наприклад, система може надати

доступ тільки тоді, коли пристрій користувача перебуває у дозволеній локації, забезпечуючи додатковий рівень контролю.

Кожен із зазначених методів має свої переваги й недоліки, але їх поєднання в межах 2FA створює потужний бар'єр для несанкціонованого доступу, забезпечуючи високу безпеку даних користувачів та системи загалом.

Таблиця 2.3 – Порівняльний огляд різних алгоритмів аутентифікації

Метод аутентифікації	Переваги	Недоліки
SMS-повідомлення	- Зручний для багатьох користувачів. - Простота використання.	- Вразливість до атаки "SIM swapping". Можливість втрати доступу в разі втрати або крадіжки телефону.
Метод одноразових паролів (OTP)	- Високий рівень безпеки, оскільки пароль використовується лише один раз. - Можливість впровадження через різні канали.	- Потребує наявності додаткового пристрою або програмного забезпечення для генерації або отримання OTP. - Потребує внимання.
Біометричні дані	- Унікальність біометричних даних для високого рівня ідентифікації. - Зручний інтерфейс для користувачів.	- Потенційні проблеми з приватністю. - Можливість помилок чи відмови через фізичний стан або технічні аспекти.
Підтвердження на основі часу (TOTP)	- Високий рівень безпеки через зміну паролю в часі. - Зручність використання через мобільні додатки.	- Потребує синхронізації часу між сервером і пристроєм користувача. - Потребує додаткового пристрою або програмного забезпечення.
Підтвердження на основі знаходження	- Можливість підвищення рівня безпеки через геолокаційні дані. - Виявлення "ненормальної" активності.	- Складність в реалізації та потреба у додаткових дозволах користувача. - Проблеми точності геолокації.

Кожен із способів двофакторної перевірки має свої сильні та слабкі сторони, тому вибір конкретного підходу визначається специфічними потребами системи та комфортом для користувачів. Водночас важливо приділити увагу й іншим аспектам безпеки, таким як коректна інтеграція, захист секретних даних і правильна обробка ключів. Ці фактори відіграють ключову роль у забезпеченні

надійності та ефективності 2FA. Порівняння різних алгоритмів аутентифікації показаний в таблиці (див. таблицю 2.3) [15].

У сучасному цифровому світі, де Інтернет відіграє ключову роль у нашому житті та роботі, зростання кіберзагроз стало значною проблемою. Одним із дієвих способів захисту є застосування одноразових паролів (OTP), які забезпечують короткочасний та унікальний доступ до системи.

OTP - це тимчасовий код, створений для підтвердження особи під час виконання транзакцій або входу до облікового запису. Його можна використовувати як самостійний метод захисту або як частину багатофакторної автентифікації. Завдяки своїй обмеженій дії в часі та індивідуальності кожного пароля, OTP значно ускладнює дії зловмисників [16].

Одноразові паролі мають кілька основних видів, кожен із яких використовується для забезпечення додаткового рівня безпеки у веб-застосунках. Перший тип — це OTP, які генеруються системою та надсилаються користувачам через канали зв’язку, такі як SMS або електронна пошта. Вони є зручними для кінцевих користувачів і легко впроваджуються, що робить їх популярним вибором для багатьох систем.

HOTP, або паролі на основі хешу, створюються із застосуванням секретного ключа та лічильника подій. Вони залишаються дійсними до моменту їх використання, що забезпечує їхню надійність у сценаріях, де потрібна прив’язка до певної дії або події.

TOTP, або часові паролі, генеруються на основі поточного часу. Вони використовують комбінацію статичного секретного ключа та змінного часовому параметра, що забезпечує їхню дійсність лише в межах короткого інтервалу часу. Завдяки цьому TOTP є особливо ефективними для захисту від атак типу "повторне відтворення" (replay attacks).

Одноразові паролі є дієвим бар’єром проти багатьох видів атак, таких як фішинг чи перехоплення облікових даних. Вибір конкретного методу OTP залежить від потреб і ресурсів організації, а також рівня безпеки, який необхідно забезпечити.

Рішення щодо вибору алгоритму для побудови системи захисту веб-застосунків з використанням двофакторної автентифікації базується на комплексному аналізі критеріїв безпеки, зручності та сумісності. Одним із найбільш ефективних рішень у цьому контексті є алгоритм одноразових паролів на основі часу (Time-based One-Time Password, TOTP).

Алгоритм TOTP (див. рисунок 2.1) має кілька вагомих переваг для використання у веб-застосунках. Його основна зручність полягає у простоті використання, оскільки OTP автоматично генерується через мобільні додатки, що спрощує автентифікацію та виключає потребу у введенні паролів вручну. Динамічність паролів, яка забезпечується за рахунок використання часового лічильника, сприяє регулярній зміні кодів, роблячи їх тимчасовими і мінімізуючи ризик компрометації.

Додатковий рівень захисту, що надає TOTP, створює подвійний бар’єр безпеки: для доступу необхідно знати пароль і мати доступ до генератора OTP. Завдяки дотриманню стандарту RFC 6238, алгоритм є універсальним і легко інтегрується з різними системами та сервісами. Широка підтримка сучасними інтернет-платформами робить його ефективним інструментом для захисту користувацьких облікових записів.

Окрім цього, TOTP зменшує ризик перехоплення паролів завдяки використанню захищених каналів передачі, адже кожен код є унікальним. Гнучкість алгоритму дозволяє використовувати його як з фізичними токенами, так і з мобільними додатками, забезпечуючи комфорт користувачам і високий рівень безпеки.

Таким чином, алгоритм TOTP є оптимальним вибором для забезпечення надійної автентифікації у веб-застосунках, поєднуючи високий рівень безпеки з простотою впровадження та використання.

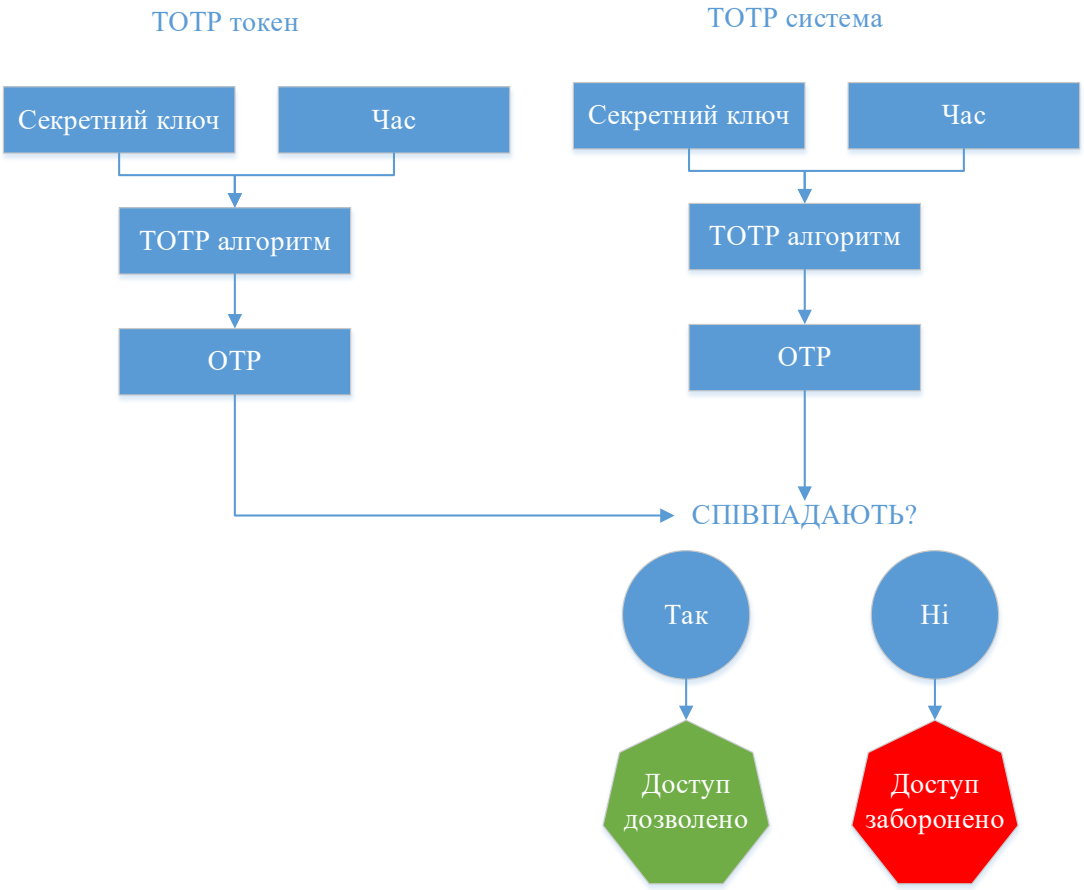

Рисунок 2.1 – Алгоритм роботи системи

2.3 Реалізація системи подвійної автентифікації та її інтеграція у веб-додатки

У впровадженій системі реалізовано структуру бази даних, що складається з однієї таблиці, достатньої для виконання алгоритмів двофакторної автентифікації (див. рисунок 2.2).

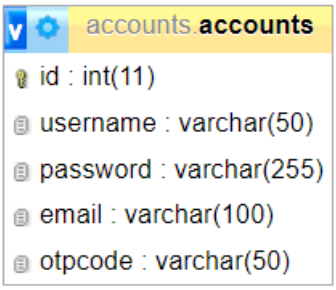

Рисунок 2.2 – База даних системи

Система підтримує кілька основних функцій, спрямованих на забезпечення реєстрації, автентифікації та доступу до персоналізованих даних користувачів. Реєстрація виконується через введення імені, електронної пошти та пароля, при цьому дані перевіряються на відповідність заданим правилам, а пароль шифрується за допомогою хешування для захисту.

Для входу в систему реалізована перевірка введеного імені користувача і пароля. Після успішної автентифікації користувач отримує одноразовий пароль (OTP), який надсилається на вказану під час реєстрації електронну пошту. Введення цього пароля на окремій сторінці підтверджує доступ до системи.

Авторизовані користувачі можуть перейти на особисту сторінку, де відображається персоналізована інформація. Для забезпечення функціональності відправки електронних листів, зокрема OTP, система використовує бібліотеку PHPMailer. Уся інформація зберігається в базі даних, яка включає структуру для запису імен, хешів паролів та електронних адрес.

Система включає кілька основних компонентів, які забезпечують її функціональність. Початковим етапом взаємодії користувача з системою є сторінка index.php, яка відповідає за авторизацію. Користувач вводить свої облікові дані, і, після успішної перевірки, система надсилає одноразовий пароль (OTP) на вказану електронну адресу для додаткової перевірки.

Реєстрація нових користувачів здійснюється через компонент register.php. Цей модуль дозволяє вводити ім’я користувача, електронну пошту та пароль, перевіряє їх на коректність і зберігає в базі даних. Таким чином, він забезпечує створення нових облікових записів.

Для введення одноразового пароля використовується сторінка otp.php. Вона дає можливість користувачу ввести OTP, надісланий на його електронну пошту, та перевіряє його на відповідність. У разі успішної перевірки користувач отримує доступ до своєї особистої сторінки.

Особиста сторінка реалізована в home.php. Вона стає доступною після завершення автентифікації і надає персоналізовану інформацію для кожного

користувача. Цей модуль служить основним інтерфейсом взаємодії із системою після входу.

Надсилання повідомлень, зокрема одноразових паролів і повідомлень про успішну реєстрацію, реалізовано за допомогою сценарію mailer.php, який використовує бібліотеку PHPMailer. Цей компонент відповідає за всю систему електронної пошти в застосунку.

Для зберігання облікових даних користувачів використовується база даних, структура якої описана у файлі accounts.sql. Таблиця включає поля для імені користувача, хешованого пароля та електронної пошти, що забезпечує надійне зберігання даних.

Система двофакторної автентифікації (2FA) реалізує кілька послідовних етапів для забезпечення безпеки доступу. Спочатку користувач створює обліковий запис через форму реєстрації на сторінці register.php, вводячи логін, пароль та адресу електронної пошти. Введені дані перевіряються на відповідність встановленим критеріям, як-от мінімальна довжина пароля чи правильність формату електронної пошти. Успішно пройдена перевірка дозволяє записати інформацію до бази даних (accounts.sql).

На етапі авторизації користувач вводить свої логін та пароль на сторінці index.php. Система перевіряє ці дані, звіряючи їх із записами у базі. Якщо дані коректні, генерується одноразовий код (OTP), який надсилається на електронну пошту користувача за допомогою бібліотеки PHPMailer. Цей код є необхідним для наступного етапу.

Далі користувач вводить отриманий OTP на сторінці otp.php. Система перевіряє правильність коду, і у випадку збігу користувач отримує доступ до системи. Після підтвердження його автоматично перенаправляють на сторінку home.php, де розміщена персоналізована інформація.

Система також забезпечує обробку помилок. Якщо під час реєстрації, авторизації чи підтвердження OTP виникають проблеми, користувач отримує відповідні повідомлення, які допомагають зрозуміти, як виправити ситуацію. Надсилання повідомлень реалізоване через бібліотеку PHPMailer із

використанням налаштувань сервера, які зберігаються у відповідному конфігураційному файлі.

Даний алгоритм гарантує, що доступ до облікового запису отримує лише користувач, який має правильний логін, пароль і одноразовий код, забезпечуючи високий рівень безпеки [17].

Розглянемо файл home.php, який є частиною системи автентифікації веб-додатка. Проаналізуємо його функції та структуру.

Код запускає сесію, що дозволяє використовувати змінні сесії для зберігання даних користувача. Цей рядок необхідний для роботи будь-яких елементів сесії та має бути першим у сценарії. Виконується перевірка наявності змінної authorized у сесії. Якщо вона існує, її значення зберігається в $auth. Інакше змінна залишається порожньою (див. лістинг 2.1).

Лістинг 2.1 – Ініціалізація сесії та перевірка автентифікації

```
<?php
session_start();
?>
<?php
if (isset($_SESSION['authorized']))
$auth = $_SESSION['authorized'];
else
$auth = '';
?>
```

Вказується структура сторінки та підключаються стилі (CSS). Зовнішній файл стилів (style.css) визначає оформлення сторінки. Якщо змінна $auth містить значення, відображається привітальне повідомлення із зазначенням імені користувача (див. лістинг 2.2).

Лістинг 2.2 – Початок HTML-документа та відображення статусу користувача

```
<!DOCTYPE HTML>
```

```
<html>
<head>
<meta charset="utf-8">
<title>Головна сторінка</title>
<link rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.1/css/all.css>
<link href="style.css" rel="stylesheet" type="text/css">
</head>
<body>
<span class="msg">
<?php if ($auth) echo 'Вітаємо, ' . $_SESSION['name'] . '!
Ви успішно авторизувалися.'; ?>
</span>
```

У разі відсутності автентифікації виводиться повідомлення про необхідність повторного входу. Сесія очищується і завершується, щоб гарантувати безпеку даних. Кнопка перенаправляє користувача на сторінку входу (див. лістинг 2.3).

Лістинг 2.3 – Повідомлення про помилку та завершення сесії

```
<span class="er">
<?php if (empty($auth)) echo "Сесія завершена. Будь ласка,
увійдіть повторно."; ?>
</span>
<?php
session_unset();
session_destroy();
?>
<button class="hbtn" type="button"
onclick="location.href='index.php'">Повернутися</button>
```

Узагальнюючи, цей сценарій забезпечує авторизацію користувачів у системі, генерує OTP-код і пересилає його електронною поштою для подальшого

підтвердження. Додатково код включає перевірку введених даних, обробку помилок та перенаправлення. Функція відповідає за зміну поточного маршруту користувача, надсилаючи його на іншу сторінку. Використовується для переходу до наступного етапу, наприклад, на otp.php, після успішної авторизації (див. лістинг 2.7).

Лістинг 2.7 – Перенаправлення користувачів

```
function redirect($url) {
header('Location: ' . $url);
exit();
}
```

Розпочинає нову сесію для відстеження авторизованого користувача. Змінні підготовлені для зберігання повідомлень про помилки чи попередження (див. лістинг 2.8).

Лістинг 2.8 – Ініціалізація сесії та повідомлень

```
session_start();
$usrError = $emailError = $pwError = $formEmpty = '';
```

Підключає зовнішню бібліотеку PHPMailer для роботи з електронними листами. Завантаження виконується через автозавантажувач Composer (див. лістинг 2.9).

Лістинг 2.9 – Імпорт бібліотек

```
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;
require 'vendor/autoload.php';
```

Код перевіряє, чи всі обов’язкові поля форми заповнені. Якщо дані неповні, виводиться повідомлення про помилку (див. лістинг 2.10).

Лістинг 2.10 – Перевірка введених даних

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$username = trim($_POST['username']);
$password = trim($_POST['password']);
if (empty($username) || empty($password)) {
$formEmpty = 'Заповніть усі поля.';
} else {
// Логіка обробки даних
}
}
```

Генерується випадковий шестизначний код, який зберігається в сесії. OTP буде використаний на наступному етапі для підтвердження (див. лістинг 2.11).

Лістинг 2.11 – Генерація OTP

```
$otp = rand(100000, 999999);
$_SESSION['otp'] = $otp;
```

Використовуючи PHPMailer, система надсилає OTP на адресу електронної пошти користувача. У разі помилки виводиться відповідне повідомлення (див. лістинг 2.12).

Лістинг 2.12 – Відправлення електронного листа

```
$mail = new PHPMailer(true);
try {
$mail->setFrom('admin@example.com', 'Система 2FA');
$mail->addAddress($userEmail);
$mail->Subject = 'Ваш OTP-код';
$mail->Body = 'Ваш одноразовий пароль: ' . $otp;
$mail->send();
} catch (Exception $e) {
$emailError = 'Не вдалося надіслати код. Спробуйте ще раз.';
}
```

Під час входу перевіряється відповідність введених даних даним у базі. У разі збігу даних користувача перенаправляють на сторінку введення OTP (див. лістинг 2.13).

Лістинг 2.13 – Перевірка користувача в базі даних:

```
$stmt = $db->prepare('SELECT * FROM users WHERE username = ?');
$stmt->execute([$username]);
$user = $stmt->fetch();
if ($user && password_verify($password, $user['password'])) {
$_SESSION['username'] = $username;
redirect('otp.php');
} else {
$usrError = 'Неправильне ім’я користувача або пароль.';
}
```

Код перевіряє, чи було отримано дані методом POST та чи заповнено обов’язкові поля форми (див. лістинг 2.14).

Лістинг 2.14 – Обробка форми POST

```
if ($_SERVER['REQUEST_METHOD'] === 'POST' &&
isset($_POST['username']) && isset($_POST['password'])) {
// Логіка для подальшої обробки
}
```

Забезпечується підключення до бази даних із перевіркою успішності з’єднання (див. лістинг 2.15).

Лістинг 2.15 – Підключення до бази даних

```
$dbHost = '127.0.0.1';
$dbUser = 'admin';
$dbPassword = 'secure_pass';
$dbName = 'user_system';
$connection = mysqli_connect($dbHost, $dbUser, $dbPassword,
$dbName);
```

```
if (!$connection) {
die('Помилка підключення до бази даних: ' .
mysqli_connect_error());
}
```

Запит до бази даних виконується за допомогою підготовлених інструкцій для уникнення SQL-ін’єкцій (див. лістинг 2.16).

Лістинг 2.16 – Використання підготовлених SQL-запитів

```
$query = $connection->prepare('SELECT id, password_hash, email
FROM users WHERE username = ?');
if ($query) {
$query->bind_param('s', $inputUsername);
$query->execute();
$query->store_result();
}
```

Верифікація пароля здійснюється через функцію password_verify. Генерується випадковий OTP-код для двофакторної аутентифікації (див. лістинг 2.17).

Лістинг 2.17 – Перевірка автентифікації та створення OTP

```
if (password_verify($inputPassword, $storedHash)) {
$otpCode = random_int(100000, 999999);
$_SESSION['otp'] = $otpCode;
}
```

Для надсилання OTP використовується PHPMailer. У разі успіху користувача перенаправляють на сторінку для введення OTP (див. лістинг 2.18).

Лістинг 2.18 – Відправлення OTP через email

```
$mail = new PHPMailer(true);
try {
$mail->setFrom('noreply@site.com', 'Auth System');
```

```
$mail->addAddress($userEmail);
$mail->Subject = 'Ваш код OTP';
$mail->Body = 'Ваш одноразовий код: ' . $otpCode;
$mail->send();
header('Location: otp_verification.php');
} catch (Exception $e) {
$emailError = 'Не вдалося надіслати повідомлення. Спробуйте
знову.';
}
```

Створюється форма входу з обов’язковими полями для введення імені користувача та пароля (див. лістинг 2.19).

Лістинг 2.19 – Форма для входу та повідомлення

```
<form action="<?php echo
htmlspecialchars($_SERVER['PHP_SELF']); ?>" method="post">
<div>
<label>Ім’я користувача</label>
<input type="text" name="username" required>
</div>
<div>
<label>Пароль</label>
<input type="password" name="password" required>
</div>
<button type="submit">Увійти</button>
<a href="register.php">Реєстрація</a>
</form>
```

Цей сценарій ілюструє відправлення електронного листа через SMTP-сервер з використанням бібліотеки PHPMailer.

Включаються основні класи бібліотеки PHPMailer для роботи з електронною поштою (див. лістинг 2.20).

Лістинг 2.20 – Імпорт необхідних класів

```
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;
require 'vendor/autoload.php';
```

Визначаються параметри SMTP-сервера, включаючи ім'я хоста, порт, дані авторизації, а також метод шифрування (див. лістинг 2.20).

Лістинг 2.20 – Створення та налаштування об'єкта PHPMailer

```
$mail = new PHPMailer(true);
try {
// Встановлення режиму SMTP
$mail->isSMTP();
$mail->Host = 'smtp.example.com'; // Адреса сервера SMTP
$mail->SMTPAuth = true;
$mail->Username = 'your-email@example.com'; // Логін
$mail->Password = 'your-password'; // Пароль
$mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
$mail->Port = 587; // Порт для з'єднання
```

Вказується адреса відправника та одержувача (див. лістинг 2.21).

Лістинг 2.21 – Налаштування адреси відправника та одержувача:

```
$mail->setFrom('your-email@example.com', 'Your Name');
$mail->addAddress('recipient@example.com', 'Recipient
Name');
```

Повідомлення може містити HTML-форматування або текстову версію для сумісності (див. лістинг 2.21).

Лістинг 2.21 – Створення повідомлення

```
$mail->isHTML(true);
$mail->Subject = 'Тестове повідомлення через SMTP';
$mail->Body    = '<p>Це приклад електронного листа, створеного
```

```
за допомогою PHPMailer.</p>';
$mail->AltBody = 'Це текстова версія електронного листа для
клієнтів, які не підтримують HTML.';
```

Лист надсилається, а в разі помилки користувач отримає детальне повідомлення (див. лістинг 2.22).

Лістинг 2.22 – Відправлення листа

```
$mail->send();
echo 'Лист успішно надіслано.';
} catch (Exception $e) {
echo "Не вдалося надіслати лист. Помилка: {$mail->ErrorInfo}";
}
```

Цей код демонструє альтернативну реалізацію двофакторної перевірки (2FA) з використанням OTP (одноразового пароля).

Використовується для переадресації користувача на потрібну сторінку (див. лістинг 2.23).

Лістинг 2.23 – Функція для перенаправлення:

```
function navigateTo($destination) {
header('Location: ' . $destination);
exit();
}
```

Запускається сесія для збереження інформації між сторінками. Ініціалізуються змінні для зберігання станів і повідомлень (див. 2.24).

Лістинг 2.24 – Старт сесії та ініціалізація змінних

```
session_start();
$otpError = '';
$successMessage = '';
```

Перевіряє, чи відправлена форма, і обробляє введене значення OTP (див. лістинг 2.25).

Лістинг 2.25 – Обробка форми:

```
if ($_SERVER['REQUEST_METHOD'] === 'POST' &&
isset($_POST['otp'])) {
$enteredOtp = trim($_POST['otp']);
if ($enteredOtp === '') {
$otpError = 'Будь ласка, введіть OTP.';
} else {
// Додаткові дії для перевірки OTP
}
}
```

Використовує підготовлений SQL-запит для перевірки, чи відповідає OTP користувачеві в базі даних (див. лістинг 2.26).

Лістинг 2.26 – Підключення до бази даних:

```
$dbConfig = [
'host' => 'localhost',
'user' => 'root',
'pass' => '',
'name' => 'example_db'
];
$connection = new mysqli($dbConfig['host'], $dbConfig['user'],
$dbConfig['pass'], $dbConfig['name']);
// …
if ($statement = $connection->prepare('SELECT id FROM users
WHERE otp = ? AND username = ?')) {
// …
}
```

Генерує повідомлення про помилки, якщо введений OTP є некоректним або відсутнім. У випадку успішного введення правильного коду відображає відповідне повідомлення (див. лістинг 2.27).

Лістинг 2.27 – Обробка та відображення результатів перевірки:

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
// ...
}
```

Представляє інтерфейс для введення OTP із кнопками для скасування дії або підтвердження введених даних (див. лістинг 2.28).

Лістинг 2.28 – Форма для введення OTP

```
<form class="lfrm" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="post">
<!-- ... (поле для введення OTP) -->
<button class="rbtn" type="reset"
onclick="location.href='index.php'">Cancel</button>
<input class="side" type="submit" value="Submit">
</form>
```

Виводить відповідні повідомлення після обробки введених користувачем даних, розташовані безпосередньо під формою OTP (див. лістинг 2.29).

Лістинг 2.29 – Відображення результатів обробки

```
<span class="er"><?php echo $inc; ?></span>
<span class="er"><?php echo $otppErr; ?></span>
<span class="er"><?php echo $suc; ?></span>
```

Код забезпечує двофакторну аутентифікацію, використовуючи одноразовий пароль (OTP), надаючи можливість перевірити введені дані через форму та отримати повідомлення про результат. Налаштовуються змінні для зберігання

введених значень, повідомлень про помилки, а також успішного виконання дій (див. лістинг 2.30).

Лістинг 2.30 – Оголошення змінних для збереження повідомлень та обробки POST-запитів

```
$usernameErr = $usernameVal = $usernameMatchErr = $emailErr =
$passwordErr = $passwordMatchErr = $passwordVal = $fieldErr =
$regerr = $stmterrmsg = $success = "";
$username = $email = $password = "";
```

Підключення до бази даних та обробка очищених даних, що вводяться користувачем, через функцію test_input() для запобігання потенційним загрозам безпеки (див. лістинг 2.31).

Лістинг 2.31 – Перевірка відправлення форми

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
// ...
}
Перевіряється, чи форма була надіслана за допомогою методу POST.
Підключення до бази даних та обробка введених даних:
$DATABASE_HOST = 'localhost';
$DATABASE_USER = 'root';
$DATABASE_PASS = '';
$DATABASE_NAME = '';
$con = mysqli_connect($DATABASE_HOST, $DATABASE_USER,
$DATABASE_PASS, $DATABASE_NAME);
// ...
function test_input($data) {
// ...
}
$username = test_input($_POST["username"]);
$email = test_input($_POST["email"]);
$password = test_input($_POST["password"]);
```

Реалізація перевірок для забезпечення того, що всі поля заповнені коректно, а введене ім'я користувача не містить неприпустимих символів (див. лістинг 2.32).

Лістинг 2.32 – Перевірка коректності даних що були введені

```
if (empty($username) || empty($password) || empty($email)) {
$fieldErr = "* All fields need to be completed";
}
if (empty($username) && !empty($fieldErr)) {
$usernameErr = "*";
}
if (!empty($username) && !preg_match('/^[A-Za-z0-9]*$/',
$username) && empty($fieldErr)) {
$usernameMatchErr = "*";
$usernameVal = "* The username includes unauthorized
characters!";
}
// ...
```

Перевіряється, чи всі поля коректно заповнені, а також обробляється повідомлення про успішну реєстрацію або помилки, якщо такий користувач уже існує в системі (див. лістинг 2.33).

Лістинг 2.33 – Обробка підтвердження реєстрації та відображення результатів

```
if ($_SERVER['REQUEST_METHOD'] == 'POST' && empty($fieldErr) &&
empty($usernameErr) && empty($emailErr) && empty($passwordErr)
&& empty($usernameMatchErr) && empty($passwordMatchErr)) {
// ...
if (!empty($registered)) {
$regerr = "The username is already taken. Please choose a
different one.
";
}
```

```
// ...
if (!empty($insert)) {
$success = " Registration completed successfully! Please proceed to
log in.";
}
}
```

Створення форми для реєстрації з полями для введення імені, пароля та електронної пошти користувача (див. лістинг 2.34).

Лістинг 2.34 – Форма для введення реєстраційних даних користувача

```
<form class="registration-form" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]); ?>" method="post">
<!-- Поля для введення логіну, паролю та пошти -->
<button class="cancel-btn" type="reset"
onclick="location.href='index.php'">Cancel</button>
<input class="submit-btn" type="submit" value="Registr">
</form>
```

Виведення різних повідомлень про помилки чи успіх після обробки форми, зокрема про порожні поля, некоректні дані або успішну реєстрацію (див. лістинг 2.35).

Лістинг 2.35 – Повідомлення про помилки або успіх виводяться під цією формою

```
<span class="er"><?php echo $fieldErr; ?></span>
<span class="er"><?php echo $regerr; ?></span>
<span class="er"><?php echo $stmterrmsg; ?></span>
<span class="er"><?php echo $usernameVal; ?></span>
<span class="er"><?php echo $passwordVal; ?></span>
<span class="success"><?php echo $success; ?></span>
```

Цей код здійснює процес реєстрації користувача, перевірку введених даних, взаємодію з БД і виведення відповідних повідомлень. Важливо, що паролі повинні бути захищені й хешовані перед їх зберіганням.

РОЗДІЛ 3 ОЦІНКА РЕЗУЛЬТАТИВНОСТІ ТА АНАЛІЗ ФУНКЦІОНАЛЬНОСТІ ВПРОВАДЖЕНОЇ СИСТЕМИ

3.1 Проведення тестувань та аналіз захисних властивостей системи для веб-додатків

Проаналізуємо функціонал впровадженої системи безпеки, що реалізує двохфакторну автентифікацію, з використанням технологій PHP, CSS, JavaScript, та CSS, а також оцінімо її ефективність у контексті її практичного застосування у веб-додатках.

Нижче наведені приклади роботи системи та результати тестування її основних компонентів.

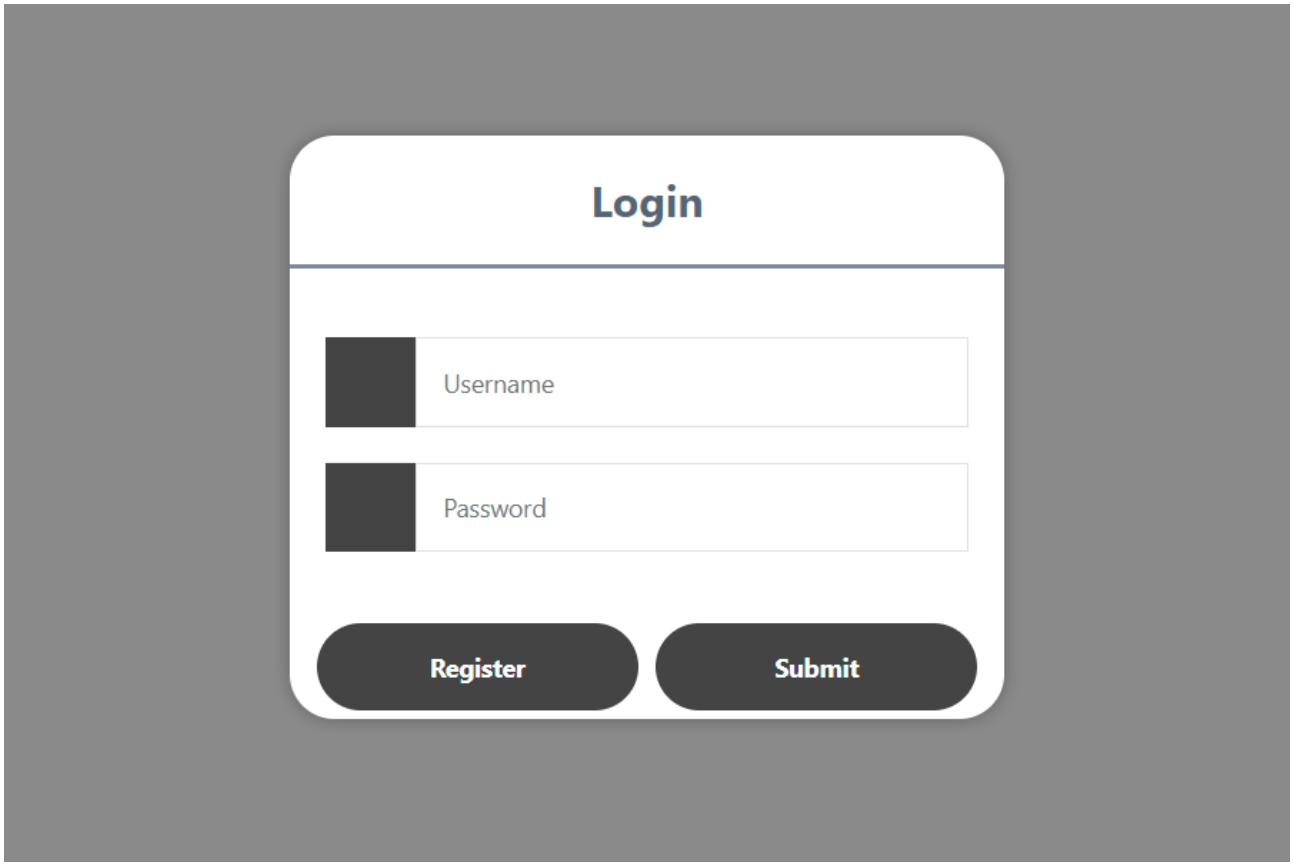

Рисунок 3.1 – Сторінка входу

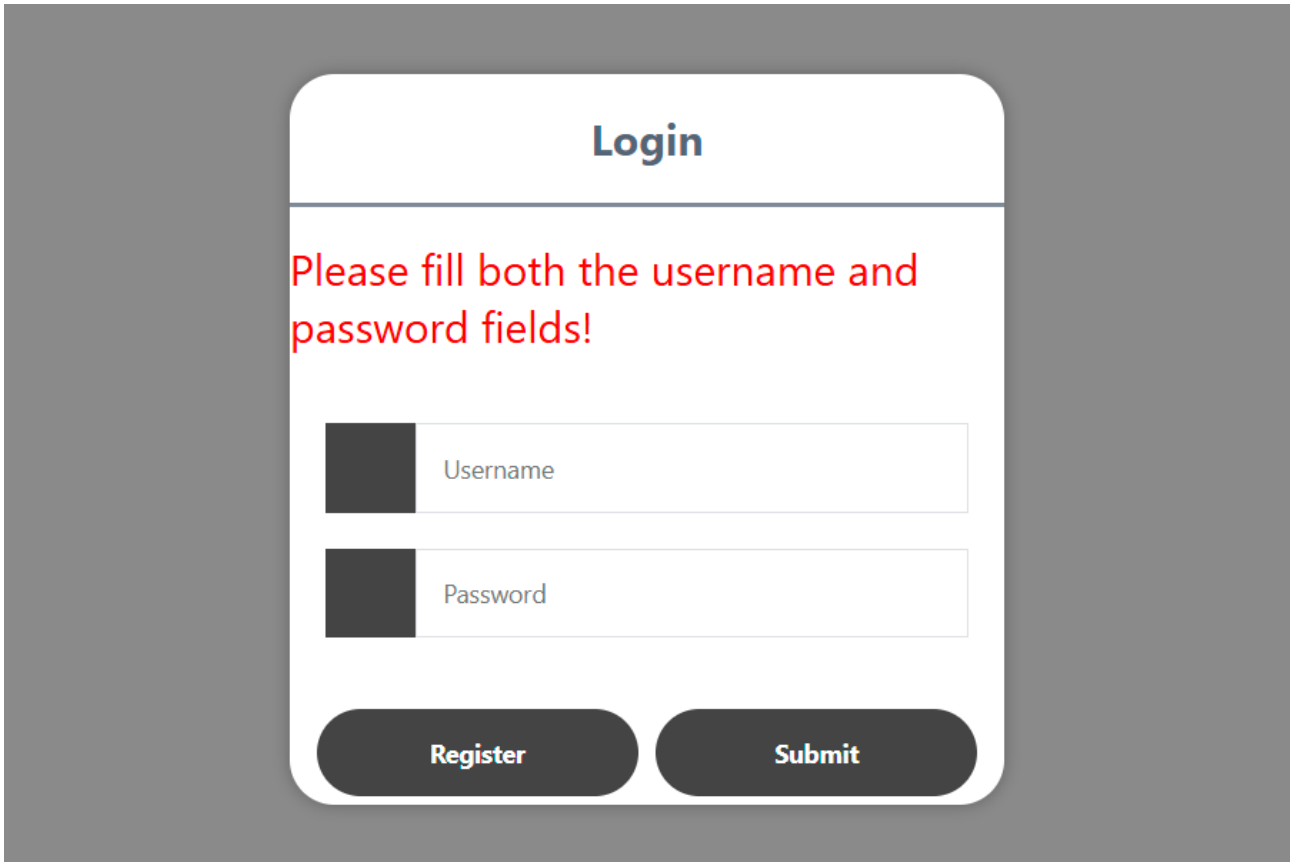

Рисунок 3.2 – Порожні поля входу

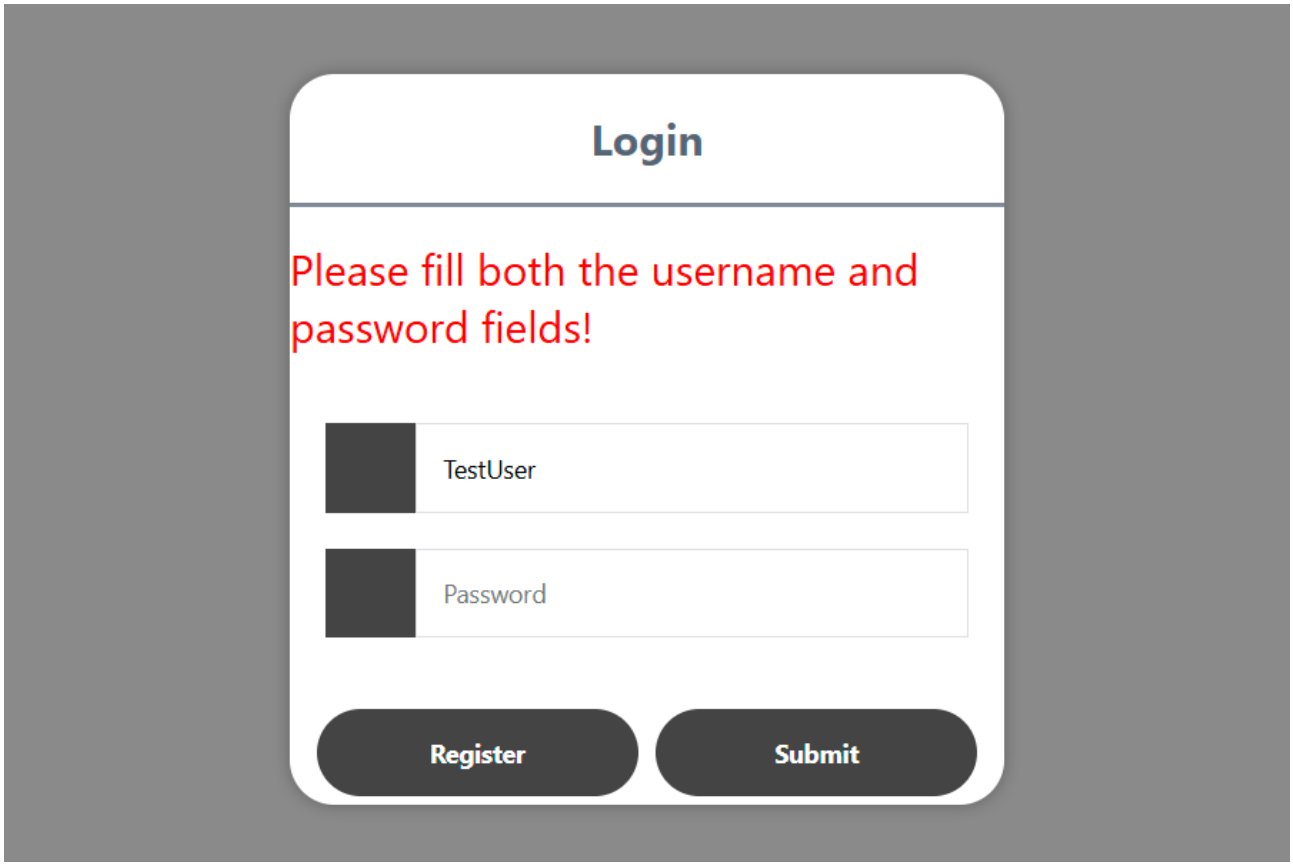

Рисунок 3.3 – Заповнено тільки поле ім’я користувача

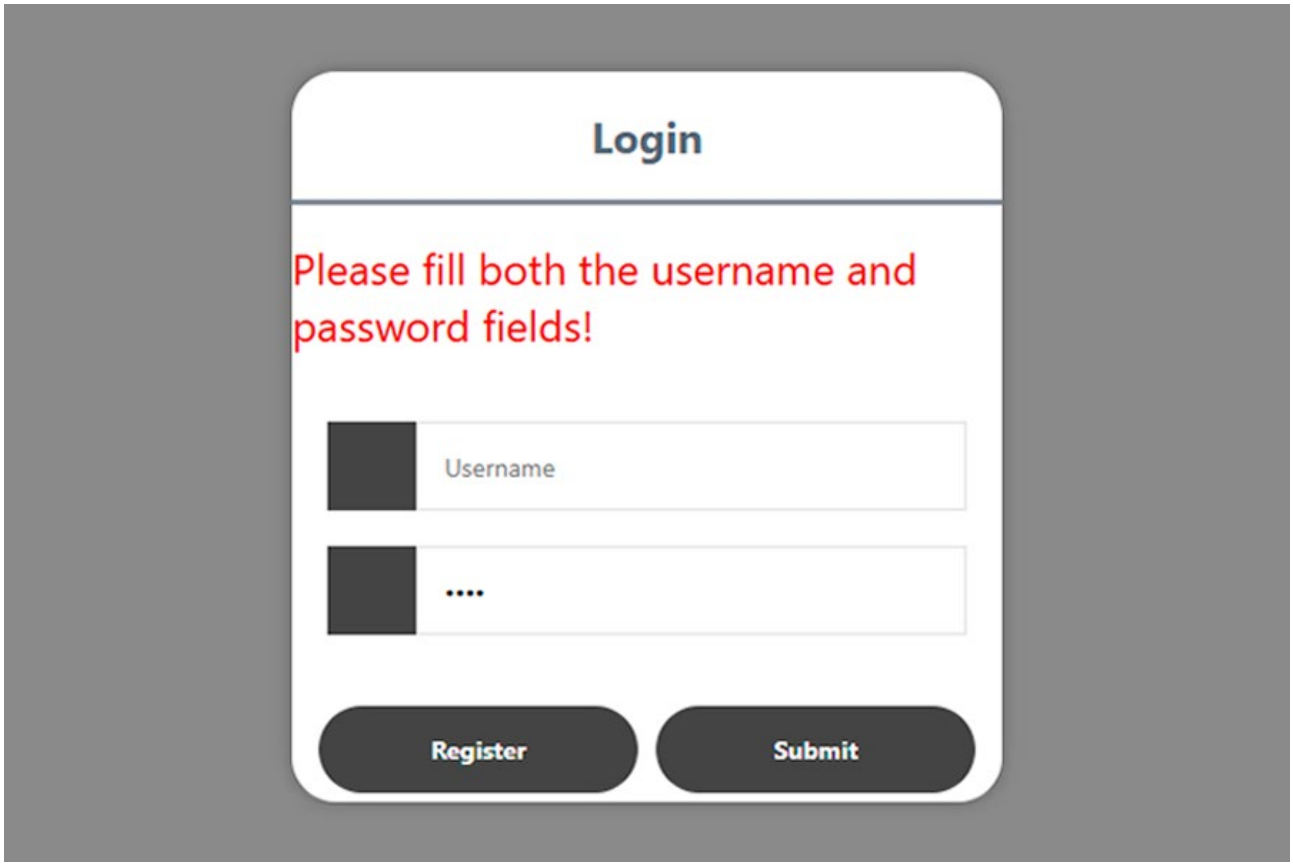

Рисунок 3.4 – Заповнено тільки поле пароля

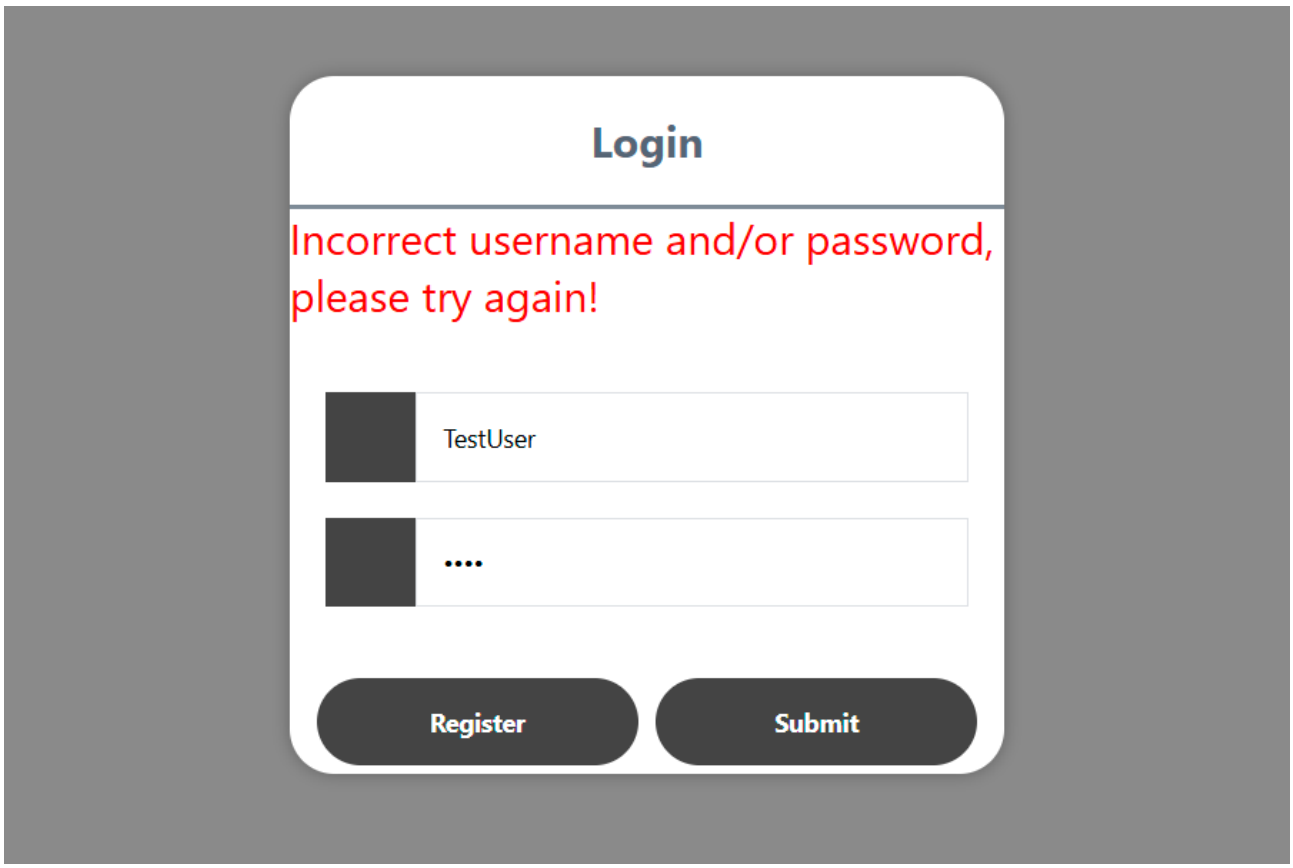

Рисунок 3.5 – Невірне ім’я користувача або пароль

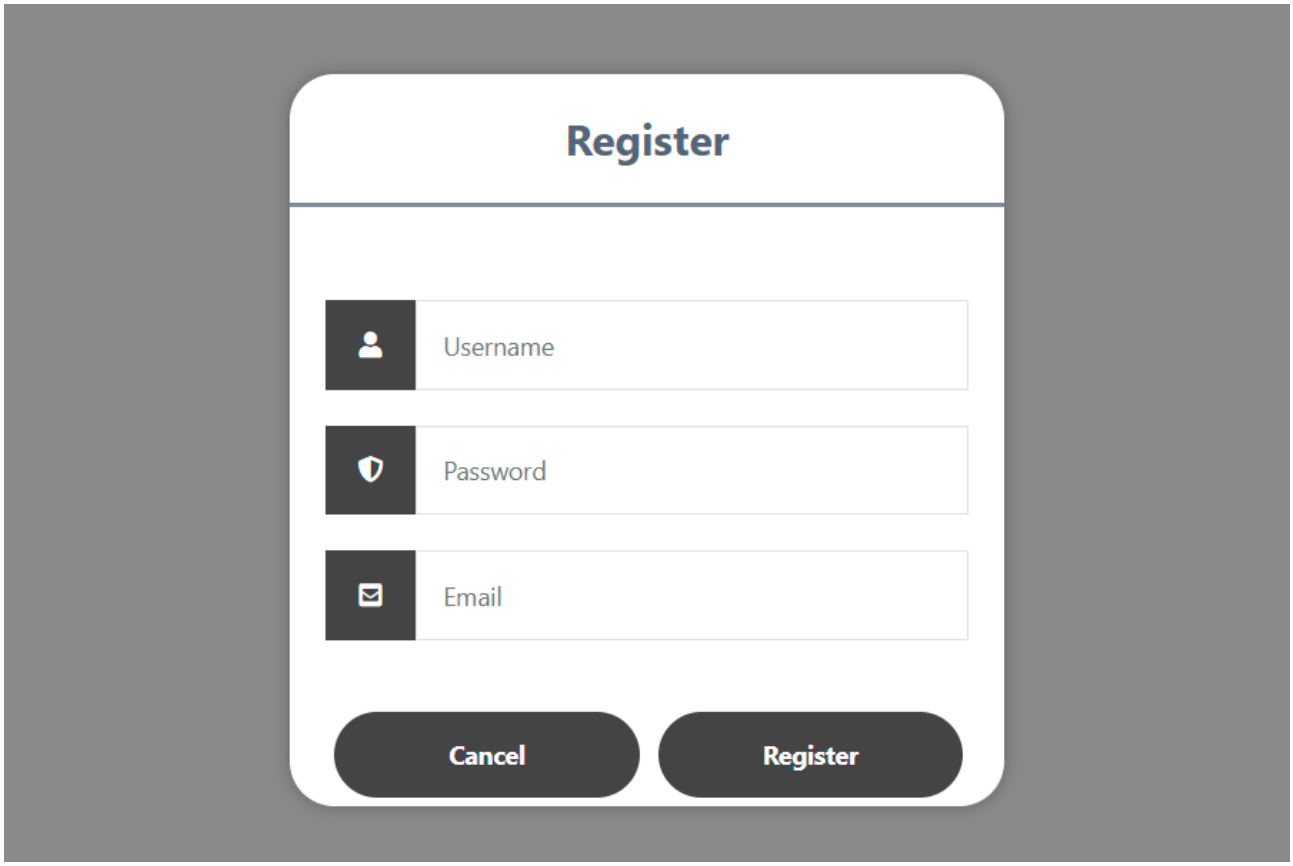

Рисунок 3.6 – Сторінка реєстрації

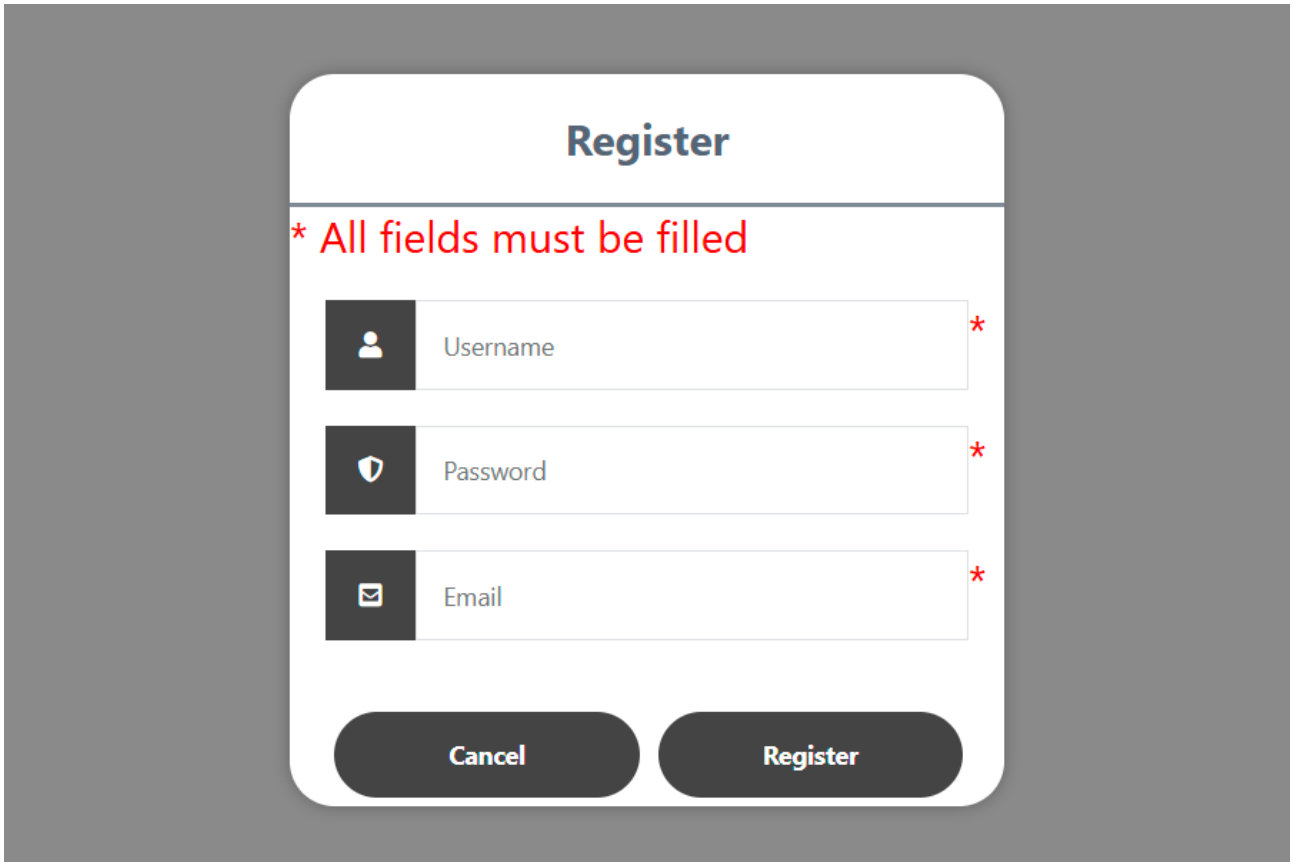

Рисунок 3.7 – Не заповнено жодне поле

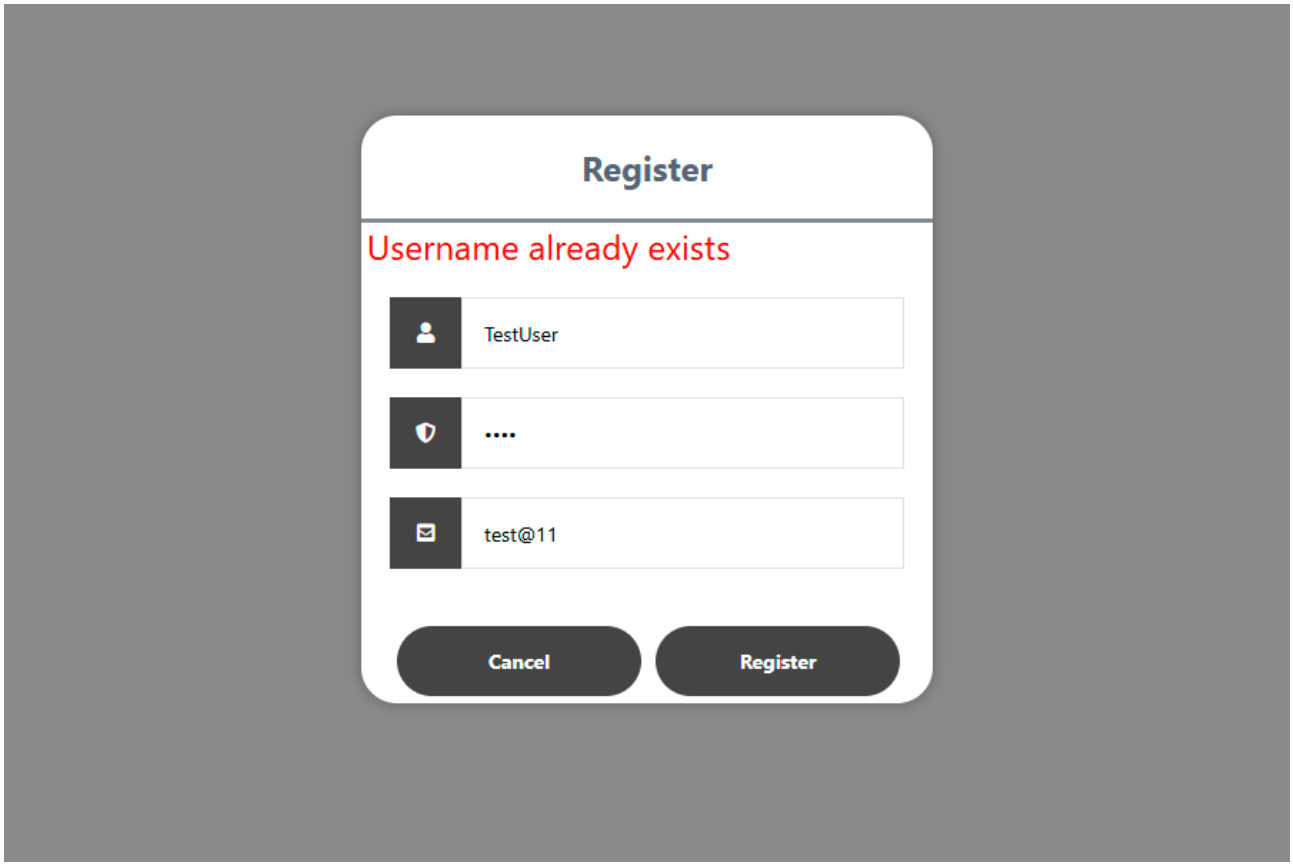

Рисунок 3.8 – Ім’я користувача зайнято

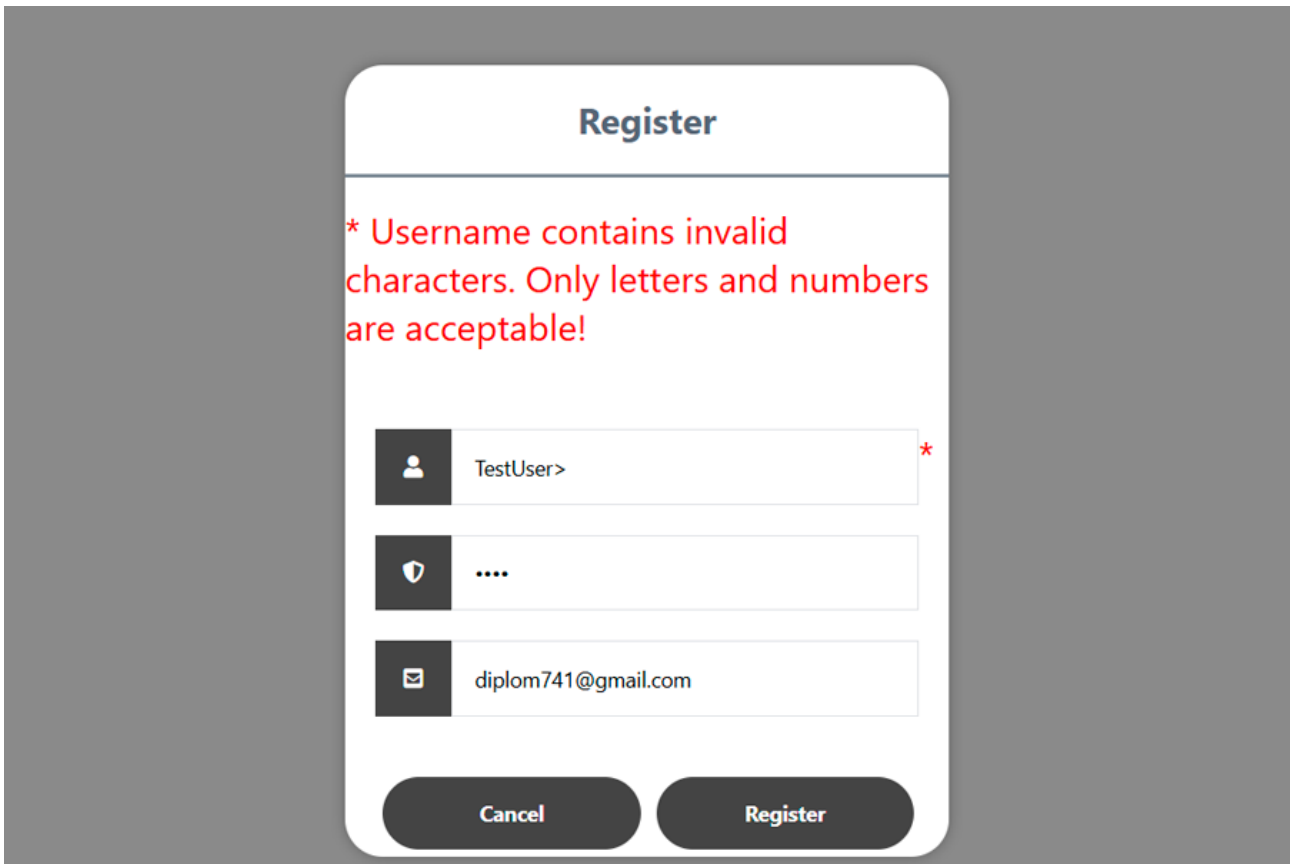

Рисунок 3.9 – Верифікація ім’я

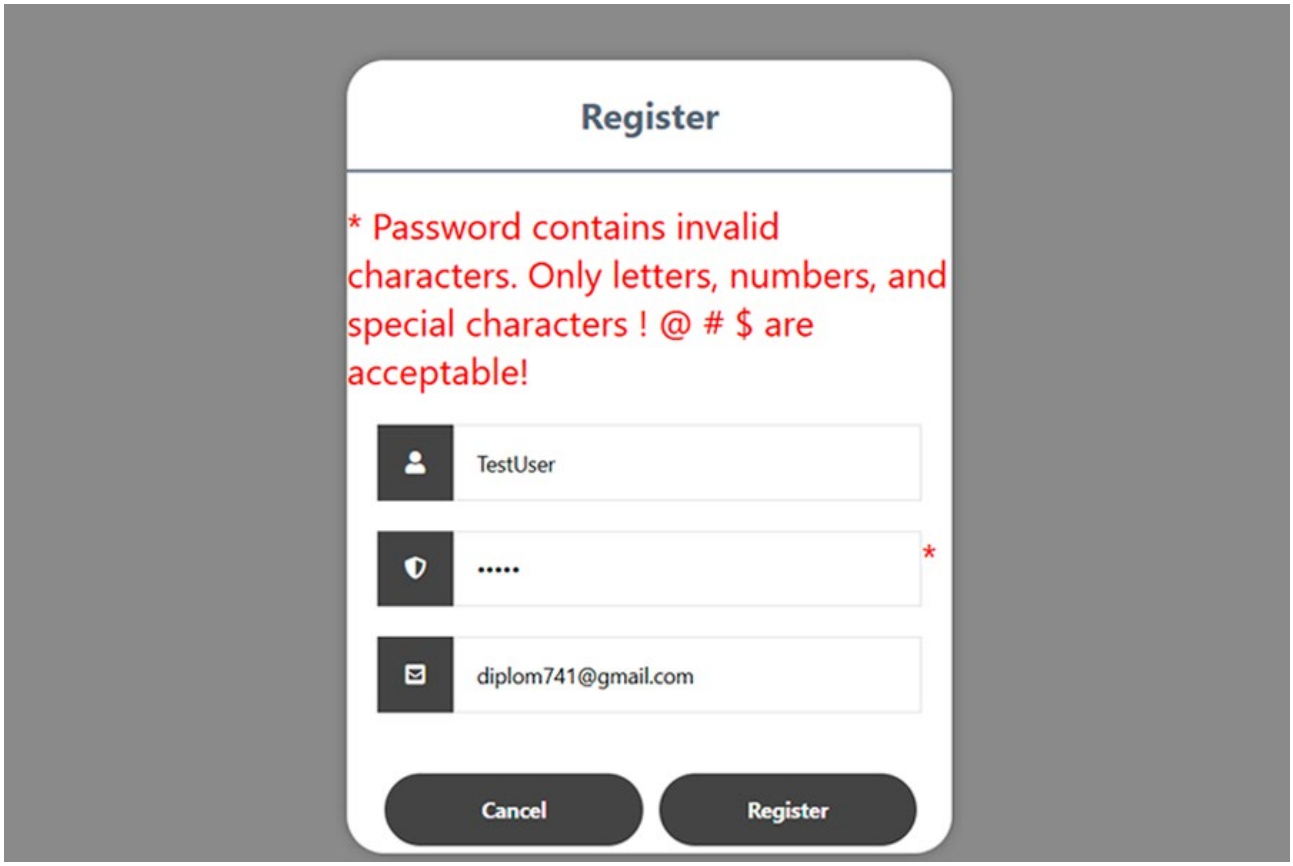

Рисунок 3.10 – Верифікація пароля

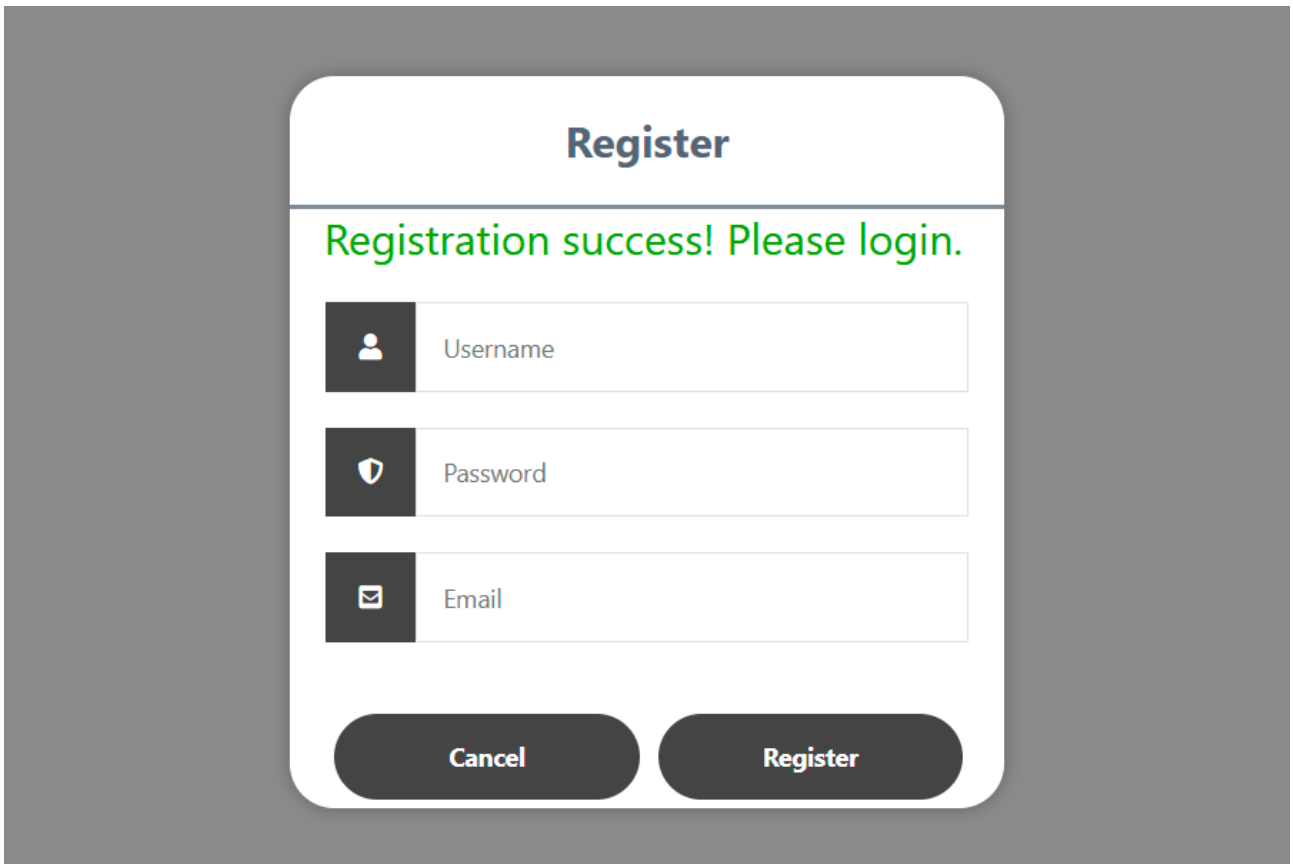

Рисунок 3.11 – Успішна реєстрація

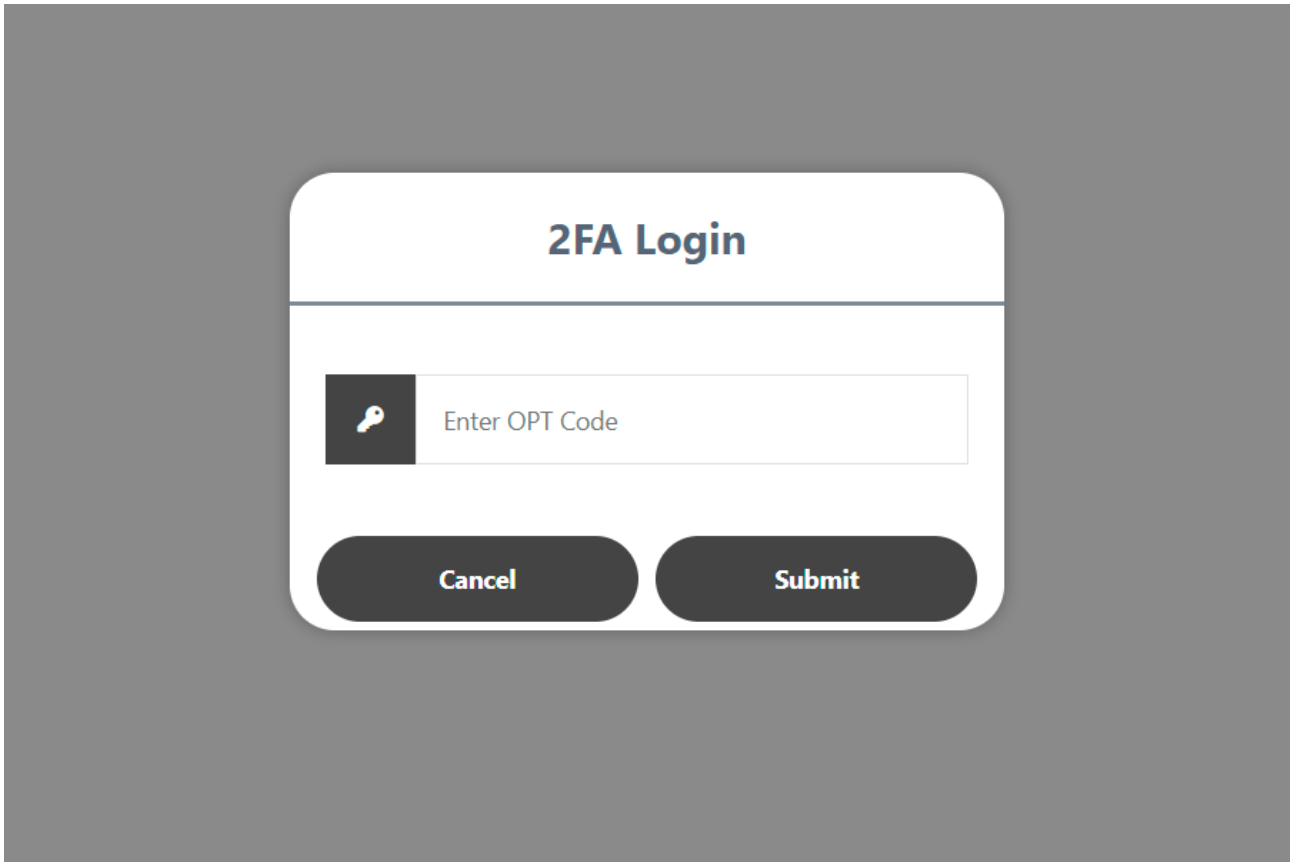

Рисунок 3.12 – Сторінка 2FA

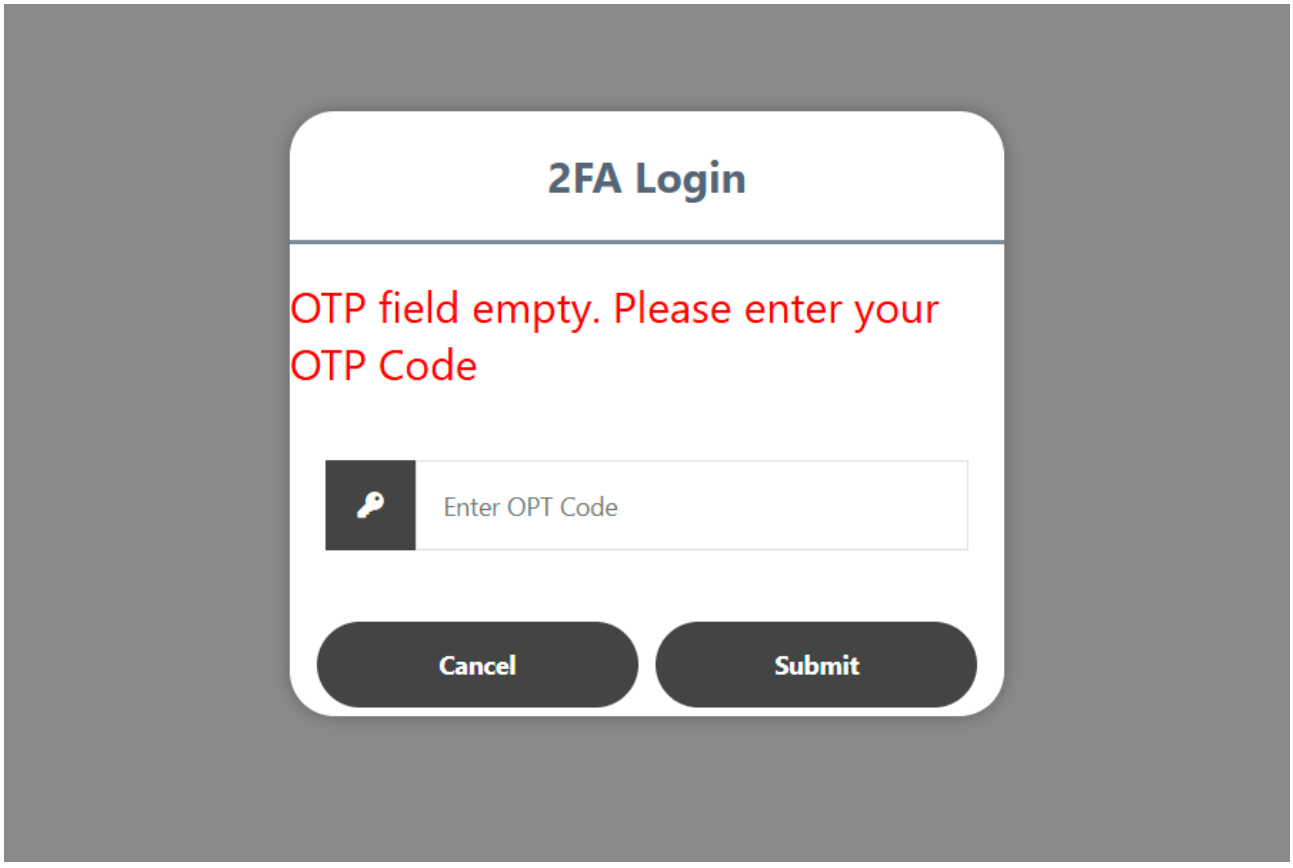

Рисунок 3.13 – Пусте поле вводу

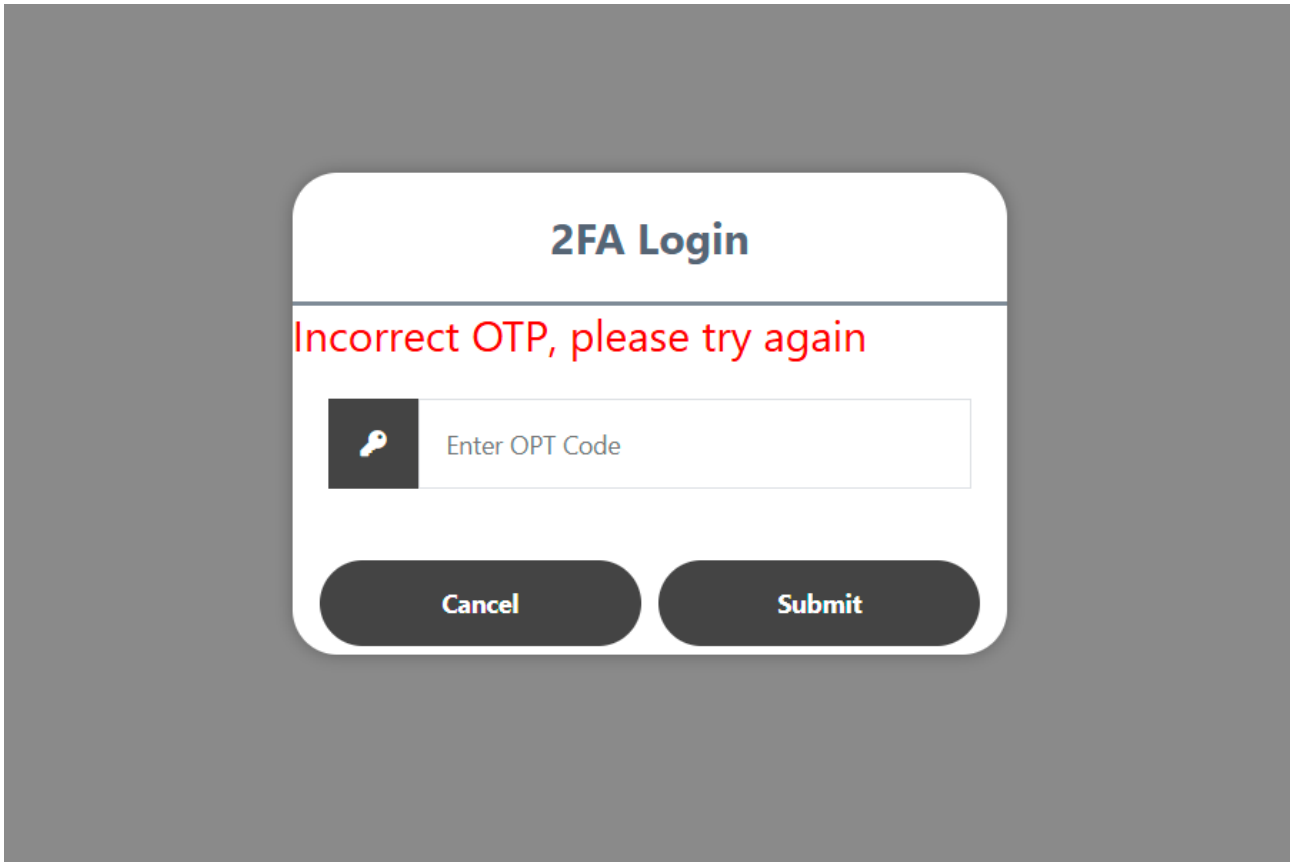

Рисунок 3.14 – Невірно вказаний код

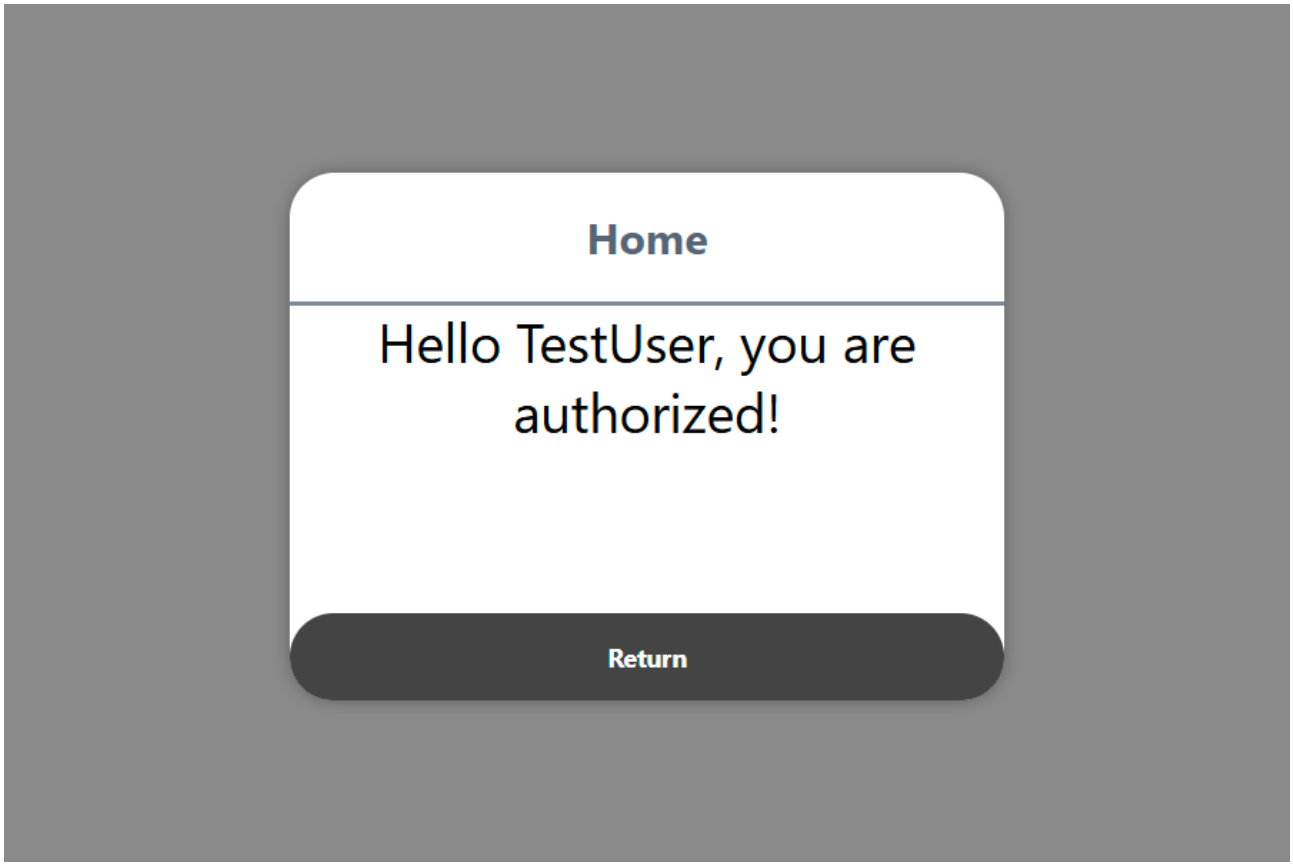

Рисунок 3.15 – Успішна авторизація

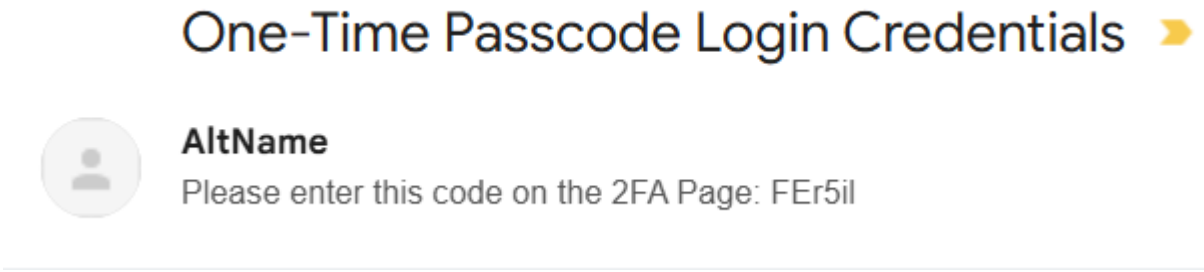

Рисунок 3.16 – Приклад отриманого OTP коду

Оцінка ефективності впровадження двофакторної автентифікації для захисту веб-додатків передбачає всебічний аналіз різних аспектів. Перш за все, досліджується надійність механізмів автентифікації, щоб визначити, наскільки ефективно система протидіє спробам несанкціонованого доступу. Це включає вивчення частоти та успішності атак, таких як фішинг чи brute-force.

Захист конфіденційної інформації є ключовим критерієм, адже збереження даних, пов’язаних із генерацією OTP, має бути організоване таким чином, щоб мінімізувати ризик перехоплення чи компрометації інформації. Крім того, аналізуються криптографічні алгоритми генерації OTP, щоб гарантувати їхню стійкість до атак.

Не менш важливою є зручність системи для користувачів. Оцінка проводиться для того, щоб зрозуміти, чи не створює двофакторна автентифікація зайвих труднощів, які можуть знизити задоволення користувачів або їх готовність працювати із системою.

Іншою складовою є функціональність систем моніторингу та аудиту, які відповідають за виявлення та відстеження підозрілих дій у реальному часі. Журнали подій аналізуються на предмет можливості виявлення спроб порушення безпеки, що дозволяє оцінити, наскільки швидко система може реагувати на інциденти.

Інтеграція з існуючими платформами також вимагає уваги. Це забезпечує сумісність системи автентифікації з різними типами веб-додатків, мінімізуючи ризик виникнення технічних труднощів або несумісностей.

Нарешті, важливим є аналіз готовності до реагування на інциденти. Він передбачає перевірку швидкості та ефективності відновлення після атак і оцінку можливостей мінімізувати шкоду. Загальна безпека веб-додатка після впровадження двофакторної автентифікації оцінюється через порівняльний аналіз рівня захищеності до і після її інтеграції.

Цей підхід дозволяє не лише визначити реальний рівень захисту, але й зрозуміти можливості подальшого вдосконалення системи.

Таблиця 3.1 – Аналіз показників захисту Web-додатків після впровадження системи двохфакторної автентифікації

Показник	Оцінка	Результат аналізу
Рівень аутентифікації	Високий	Система ефективно запобігає несанкціонованому доступу.
Використання секретів	Задовільний	Забезпечена відповідна безпека зберігання та передачі секретів, але існують можливості для покращення.
Міцність алгоритмів аутентифікації	Високий	Використовуються стійкі алгоритми для генерації одноразових паролів.
Прозорість та зручність використання	Задовільний	Користувачі розуміють і використовують систему, але можливості для поліпшення щодо зручності використання.
Моніторинг та журналювання	Високий	Моніторинг та журналювання подій в системі відбуваються ефективно, що дозволяє вчасно виявляти несанкціонований доступ.
Інтеграція з існуючими web-застосунками	Задовільний	Є інтеграція з різноманітними Web-застосунками, але існують області для подальшої оптимізації.
Реагування на інциденти	Високий	Система ефективно реагує на події, пов'язані з безпекою, та швидко відновлюється після інцидентів.
Загальний рівень безпеки	Високий	Загальний рівень безпеки Web-застосунків після впровадження системи двохфакторної автентифікації є високим.

Для більш наочного уявлення результатів тестування буде наведена таблиця з ключовими показниками ефективності системи (див. таблицю 3.1).

Проведений аналіз показав, що впровадження двофакторної автентифікації дало позитивні результати. Система продемонструвала високу ефективність у забезпеченні надійного рівня захисту, використанні сучасних криптографічних алгоритмів і можливості оперативного відстеження подій. Разом із цим, виявлено потенційні напрямки для вдосконалення, зокрема, у спрощенні взаємодії користувачів із системою та оптимізації процесу інтеграції з іншими веб-додатками. Загальна оцінка показує значне підвищення рівня безпеки, що підтверджує ефективність обраного підходу в протидії загрозам безпеки[17].

3.2 Оцінювання ефективності впровадження системи подвійної автентифікації для забезпечення безпеки веб-додатків

Основна проблема безпеки при взаємодії між користувачем і базою даних полягає в тому, що передача даних відбувається безпосередньо, як це показано на схемі (див. рисунок 3.17) [18].

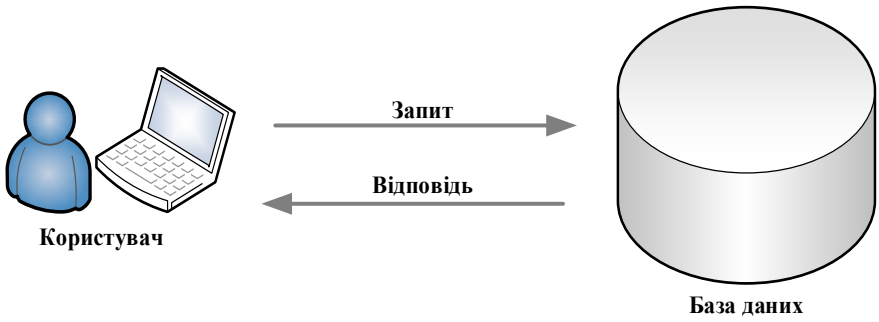

Рисунок 3.17 – Спілкування користувача та бази даних

У разі відсутності належної перевірки введених даних і помилок у реалізації додатка ризик злому суттєво зростає.

Для вирішення цієї проблеми застосовується система двофакторної аутентифікації, що виконує роль посередника. Вона працює як проксі-сервер, який перехоплює весь трафік між клієнтом і сервером бази даних. Користувач

налаштовує параметри, такі як тип бази даних, IP-адресу сервера, порт і режим роботи системи. Сервіс аналізує запити клієнта, виділяючи SQL-інструкції, які зберігаються в логах. У разі виявлення підозрілих дій SQL-запит також записується в лог-файл, але його виконання на сервері бази даних блокується. В інших випадках система очікує наступного запиту клієнта.

Алгоритм функціонування цієї системи подано у вигляді блок-схеми (див. рисунок 3.18).

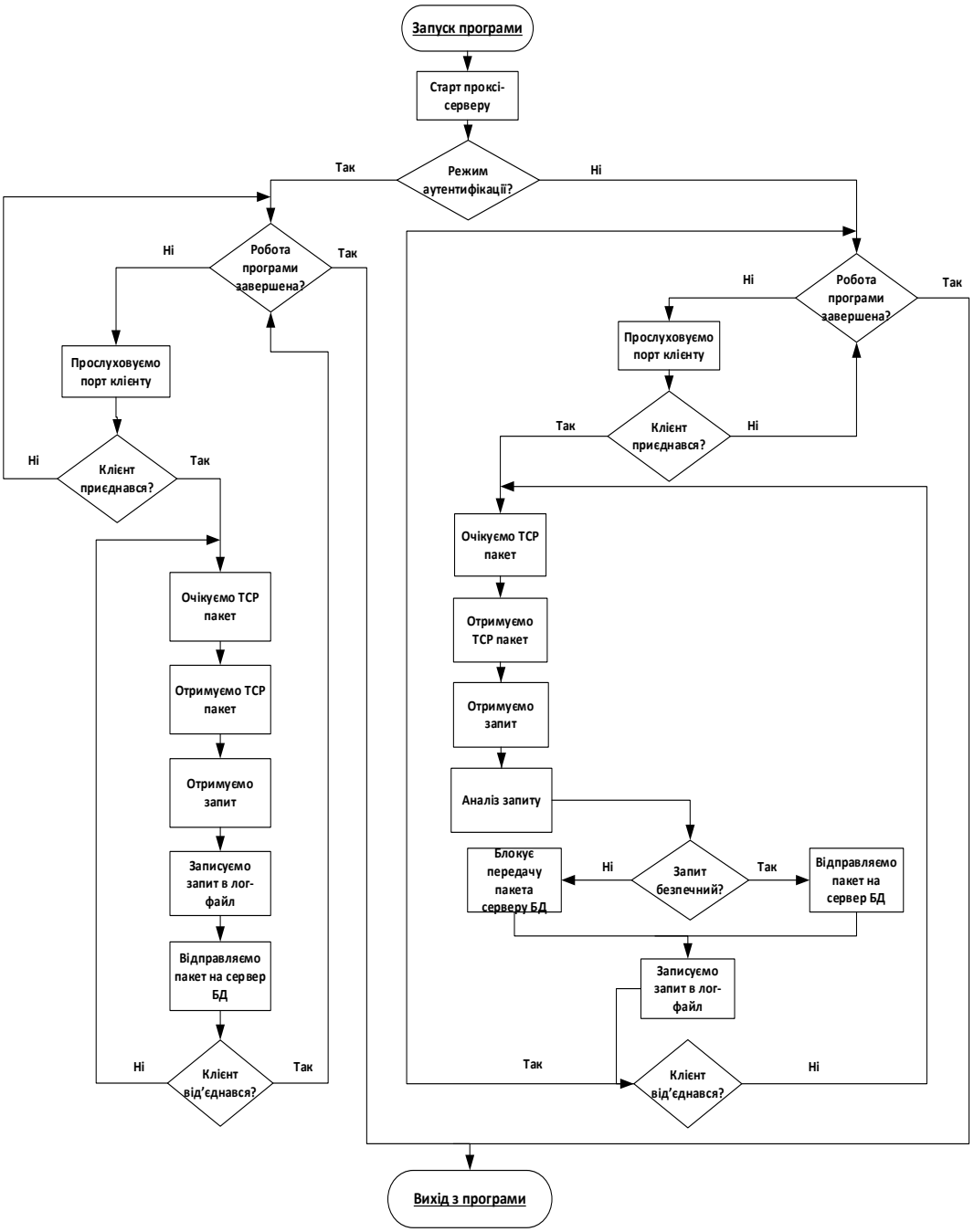

Рисунок 3.18 – Схема функціонування алгоритму для взаємодії з базою даних

Звертаючи увагу на це, взаємодія між користувачем і сервером бази даних здійснюється через проксі-сервер (див. рисунок 3.19). Усі запити аналізуються для виявлення потенційно небезпечних або некоректних включень.

У ході дослідження було здійснено тестування різних способів аутентифікації, під час якого оцінювався середній час виконання операцій. Для різних типів запитів створювалися різні варіанти, а вимірювання проводилися 25 разів для забезпечення точності результатів.

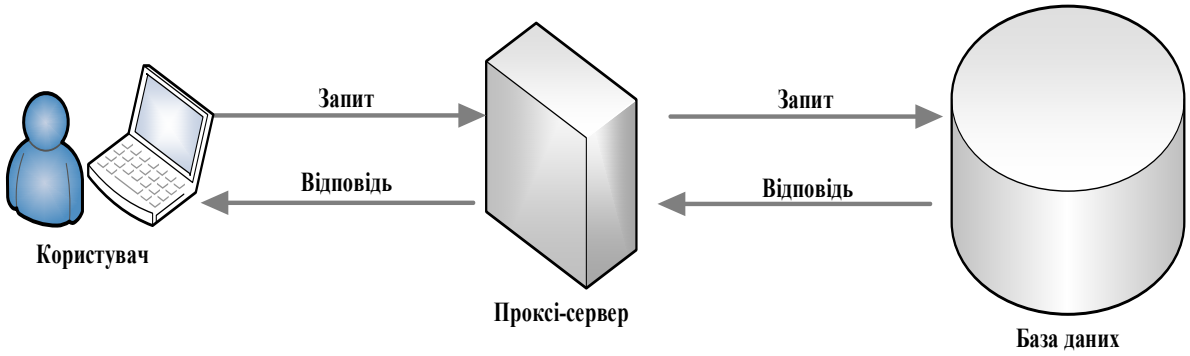

Рисунок 3.19 – Процес комунікації між користувачем та базою даних через проксі сервер

Після закінчення всіх підготовчих етапів розпочинається процес фіксації часу обробки запитів. Всі запити виконуються послідовно 25 разів для кожного з типів баз даних — як при прямому підключенні, так і через систему посередника. У результаті проведених вимірів отримано усереднені дані, які відображені у відповідних таблицях.

На основі зібраних результатів побудовано графіки (див. рисунки 3.20, 3.21, 3.22), що ілюструють середні значення часу виконання для всіх видів запитів.

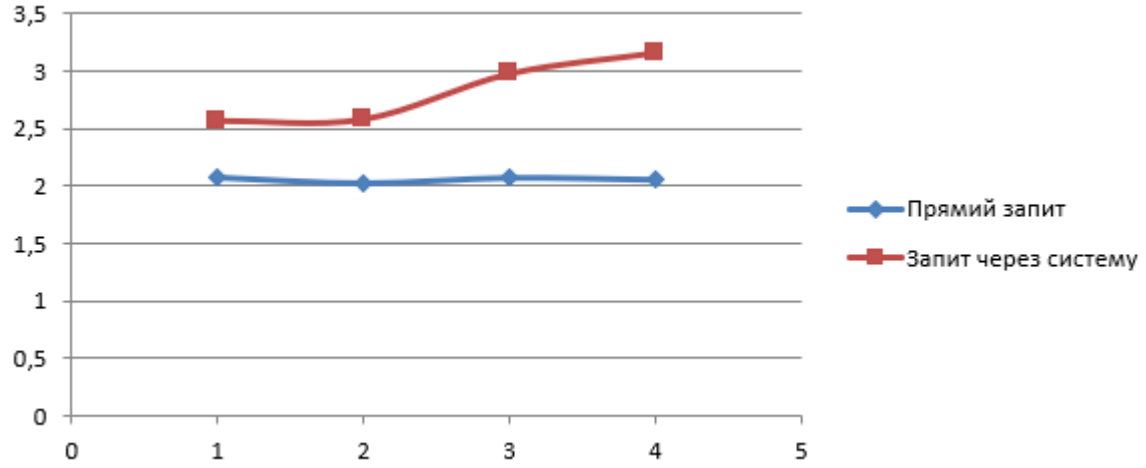

Рисунок 3.20 – Діаграма середніх значень запитів SELECT

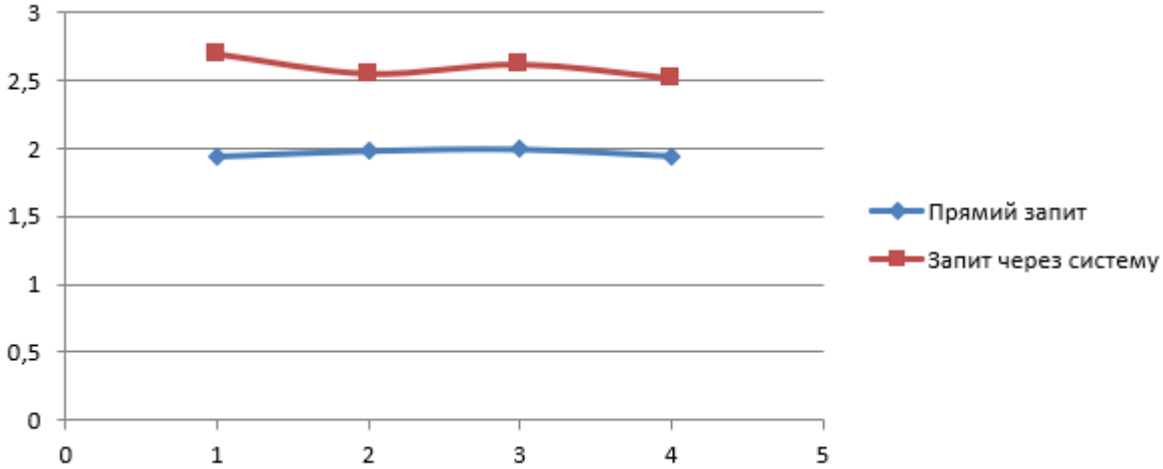

Рисунок 3.21 – Діаграма середніх значень запитів INSERT

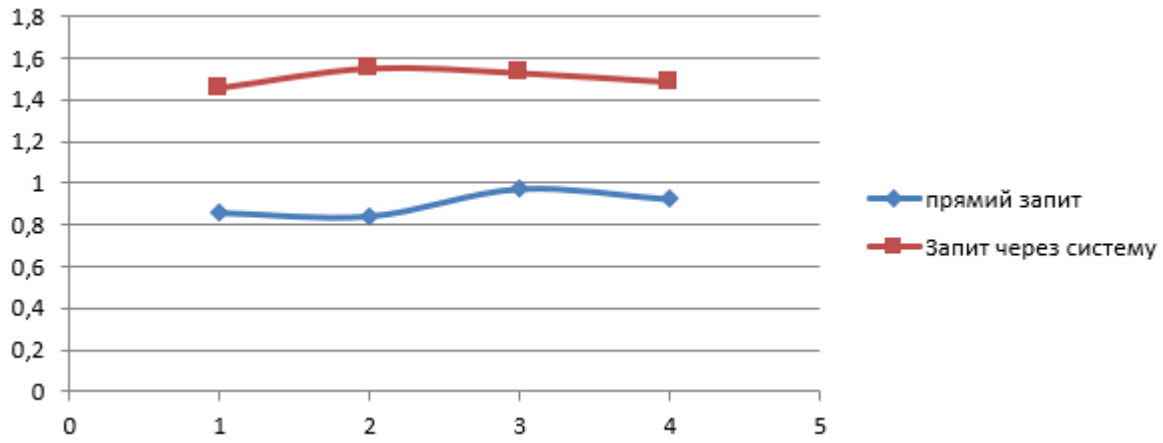

Рисунок 3.22 – Діаграма середніх значень запитів DELETE

Аналіз діаграм демонструє, що виконання запитів через систему супроводжується трохи більшим часом обробки порівняно з прямим доступом. Це пов'язано з додатковим етапом аналізу запитів для забезпечення автентифікації користувачів. Однак затримка є незначною.

Результати тестування виглядають наступним чином:

– затримка перших чотирьох запитів типу SELECT склала 0,49388, 0,54924, 0,908 і 1,10496 мілісекунди відповідно, що в середньому становить 0,76402 мілісекунди;

– затримка перших чотирьох запитів типу INSERT склала 0,75428, 0,56352, 0,6198 і 0,57116 мілісекунди відповідно, із середнім значенням 0,62719 мілісекунди;

– затримка перших чотирьох запитів типу DELETE склала 0,5982, 0,71316, 0,5548 і 0,56088 мілісекунди відповідно, із середнім показником 0,60676 мілісекунди.

Середнє значення затримки виконання запитів через систему для всіх типів склало 0,66599 мілісекунди.

Отже, система додає незначне збільшення часу обробки запитів, що свідчить про її ефективність та збалансованість. Це дозволяє виконувати функції автентифікації без істотного навантаження на базу даних, забезпечуючи при цьому високий рівень захисту.

РОЗДІЛ 4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Охорона праці в Україні регулюється Законом України «Про охорону праці», Кодексом законів про працю України, а також іншими нормативно-правовими актами, які встановлюють обов’язки роботодавців та права працівників щодо створення безпечних умов праці. У сфері інформаційних технологій дотримання цих норм є обов’язковим.

Впровадження системи захисту веб-застосунків передбачає використання різноманітного обладнання та взаємодію з програмними продуктами, тому необхідно забезпечити безпечні умови роботи для всіх учасників процесу. Особливу увагу слід приділити заходам безпеки при роботі з обладнанням, інформаційними системами та забезпеченню протипожежної безпеки [19].

Для уникнення ризиків, пов’язаних із використанням комп’ютерної техніки під час реалізації системи захисту веб-застосунків, працівники повинні дотримуватися правил експлуатації обладнання. Перед початком роботи з серверним обладнанням, мережевими пристроями або іншим технічним забезпеченням необхідно провести діагностику їхнього стану. Важливо уникати перевантаження електромереж, що є особливо актуальним при налаштуванні серверів чи інших енергомістких пристроїв, які забезпечують функціонування системи двофакторної автентифікації. Використання джерел безперебійного живлення (UPS) дозволяє уникнути втрати даних або збою в роботі системи під час аварійного вимкнення електроенергії. Сервери, мережеві комутатори та інші пристрої повинні розташовуватися на поверхнях із антистатичним покриттям, а працівники мають використовувати антистатичні браслети чи заземлення при роботі з чутливими компонентами. Робочі місця працівників, які займаються налаштуванням системи, мають бути обладнані якісними електромережевими фільтрами, заземленими розетками та відповідати вимогам чинних українських

стандартів безпеки. Усі кабелі мають бути розташовані так, щоб уникнути їх пошкодження або створення перешкод для пересування.

Впровадження системи двофакторної автентифікації включає використання серверного обладнання, яке споживає значну кількість енергії та може стати джерелом займання у разі несправності. У приміщеннях, де розміщено сервери або мережеве обладнання, мають бути встановлені вогнегасники, що відповідають стандартам ДСТУ EN 615:2017. Регулярна перевірка справності електропроводки та обладнання, зокрема кабелів, які використовуються для підключення серверів або джерел живлення, є необхідною умовою. Приміщення повинні бути оснащені системами пожежної сигналізації та димовидалення, що дозволяє оперативно виявляти та локалізувати загоряння. Шляхи евакуації мають бути завжди вільними та чітко позначеними, а працівники — регулярно проходити навчання з евакуації та використання протипожежного обладнання. Крім того, слід забезпечити правильне зберігання легкозаймистих матеріалів, наприклад очищувальних засобів для обладнання, якщо такі використовуються, і уникати їхнього розміщення поблизу джерел тепла.

Дотримання вимог охорони праці при впровадженні системи захисту веб-застосунків на основі двофакторної автентифікації є необхідною умовою для забезпечення безпеки робочого процесу. Організація належних заходів безпеки, таких як використання антистатичних матеріалів, джерел безперебійного живлення та засобів пожежогасіння, мінімізує ризики для здоров’я працівників і збереження техніки. Впровадження комплексного підходу до охорони праці сприяє створенню безпечного середовища, забезпечує ефективність роботи та знижує ймовірність аварійних ситуацій. Таким чином, заходи охорони праці є невід’ємною складовою успішної реалізації системи двофакторної автентифікації.

4.2 Безпека в надзвичайних ситуаціях

Забезпечення електробезпеки користувачів персональних комп'ютерів є важливою задачею, що безпосередньо впливає на збереження здоров'я і продуктивності людей у сучасному інформаційному суспільстві. У сьогоднішньому цифровому середовищі комп'ютерні системи та периферійне обладнання є невід'ємною частиною життя більшості користувачів. Проте, зростаюча залежність від технологій вимагає підвищеної уваги до питань безпеки, зокрема, електричної безпеки. Оператори комп'ютерних систем повинні знати про можливі ризики, пов'язані з електричним струмом, і бути готовими до їх мінімізації.

У цьому підрозділі буде розглянуто основні джерела електричних небезпек, методи їх ідентифікації та засоби захисту, які допомагають забезпечити безпеку користувачів під час роботи з персональними комп'ютерами. Розглянуті також заходи управління електробезпекою, спрямовані на зниження ризиків та забезпечення належного рівня захисту [19].

4.2.1 Основні джерела електричних небезпек

Основні джерела електричних небезпек, що можуть загрожувати користувачам персональних комп'ютерів, включають:

1. Несправності електричних мереж**:** Пошкодження кабелів, слабкі контакти розеток, недотримання стандартів електробезпеки під час монтажу електричних мереж можуть призвести до коротких замикань, підвищення напруги та інших проблем, що створюють ризик ураження електричним струмом.
2. Неправильна експлуатація**:** Використання електричних приладів, що не відповідають технічним характеристикам або призначені для іншої напруги, може призвести до пожежі або ураження струмом.

3. Блискавка**:** Ризик ураження блискавкою, що може потрапити в мережу через комп'ютерну техніку або інші електричні пристрої, потребує застосування захисних пристроїв, таких як захисні штепселі та розетки з заземленням.

4. Електромагнітні випромінювання**:** Вплив електромагнітних випромінювань від комп'ютерної техніки може мати негативний вплив на здоров'я користувачів. Це особливо важливо для осіб, що страждають на хронічні захворювання або мають чутливу нервову систему.

4.2.2 Ідентифікація небезпек

Ідентифікація небезпек електричного характеру передбачає проведення перевірок та тестувань електричних мереж, комп'ютерних приладів і техніки на предмет безпеки. Основні методи включають:

1. Аналіз стану електричних мереж**:** Огляд та перевірка якості проводки, розеток, заземлень, використання витратних матеріалів, які відповідають стандартам безпеки.

2. Тестування приладів**:** Перевірка технічного стану комп'ютерних приладів, стабілізаторів, подовжувачів та інших компонентів для виявлення несправностей, що можуть загрожувати безпеці користувачів.

3. Моніторинг параметрів електричних мереж**:** Регулярний моніторинг напруги, струму, частоти та інших параметрів для виявлення відхилень від норми.

4.2.3 Засоби захисту

Для забезпечення електробезпеки користувачів персональних комп'ютерів застосовуються різні засоби захисту, які дозволяють мінімізувати ризики:

1. Захисні пристрої**:** Використання стабілізаторів напруги, РЕЗ (реле захисту) та захисних штепселів для запобігання коротким замиканням і підвищення рівня безпеки.

2. Заземлення**:** Важливість заземлення комп'ютерної техніки для запобігання ураження струмом у випадку пошкодження електричної мережі або блискавки.

3. Захисні пристрої від блискавки**:** Використання блискавковідводів та пристроїв захисту від стрибків напруги для запобігання пошкодження техніки під час гроз.

4. Правильна експлуатація**:** Дотримання інструкцій з експлуатації, використання лише тієї техніки та компонентів, які відповідають технічним вимогам.

4.2.4 Впровадження та управління електробезпекою

Впровадження та управління електробезпекою включає навчання користувачів, регулярні інспекції електричних систем, а також впровадження політик безпеки для мінімізації ризиків:

1. Навчання користувачів**:** Проведення семінарів, тренінгів та інструктажів для користувачів щодо правильного використання електронної техніки та заходів безпеки.

2. Регулярні перевірки**:** Заплановані технічні огляди та перевірки електричних мереж для забезпечення безпечного функціонування техніки.

3. Розробка політик безпеки**:** Впровадження політик безпеки для захисту користувачів від електричних небезпек, таких як заборона використання несправної техніки, встановлення правил для роботи з комп'ютерами.

4.2.5 Висновок

Електробезпека користувачів персональних комп'ютерів є ключовим аспектом забезпечення безпеки роботи з цифровими технологіями. Небезпеки, що пов'язані з електричним струмом, можуть мати серйозні наслідки для здоров'я, включаючи ураження електричним струмом, пожежі та технічні

поломки. Впровадження належних заходів безпеки, регулярні перевірки електричних систем і правильне навчання користувачів є необхідними компонентами стратегії захисту.

Забезпечення електробезпеки включає не лише використання сучасних захисних пристроїв, але й дотримання технічних стандартів та рекомендацій щодо експлуатації комп'ютерної техніки. Основними заходами є встановлення стабілізаторів напруги, застосування захисних розеток, правильне заземлення та використання одноразових паролів для аутентифікації, що знижують ризик виникнення аварійних ситуацій.

Основні джерела небезпек включають несправності електричних мереж, використання несумісної техніки, а також вплив блискавки. Використання засобів захисту, таких як РЕЗ, стабілізатори напруги та пристрої захисту від стрибків напруги, дозволяє мінімізувати ці ризики. Важливим є також впровадження політик безпеки для підтримки належного рівня електробезпеки у всіх цифрових середовищах.

Правильне впровадження цих заходів і постійний моніторинг технічного стану електричних мереж і комп'ютерної техніки допомагають створити безпечне робоче середовище для користувачів персональних комп'ютерів. Використання ефективних методів і сучасних технологій забезпечує стабільну і безпечну роботу в умовах електричних небезпек.

ВИСНОВКИ

Протягом виконання роботи було всебічно досліджено різноманітні аспекти, які стосуються автентифікації, захисту інформації та безпеки веб-додатків. Було проаналізовано сучасні підходи до впровадження систем двофакторної автентифікації, що включало вивчення мов програмування та систем керування базами даних. На основі проведеного аналізу було обрано PHP і MySQL як основні технології для реалізації впровадженої системи захисту, оскільки вони забезпечують оптимальний баланс між функціональністю, доступністю та простотою впровадження.

Особливу увагу було приділено вибору алгоритму двофакторної автентифікації. Результатом дослідження стало впровадження та обґрунтування використання алгоритму OTP, який зарекомендував себе як один із найнадійніших і найбільш придатних для захисту веб-додатків. Було описано структуру впровадженої системи, її ключові функції та алгоритми роботи, а також наведено детальний опис реалізації основних компонентів.

У рамках роботи проведено розгортання системи на локальному сервері, детально описано необхідні інструменти для її роботи, а також інстальовані розширення для забезпечення функціонування всіх компонентів. Під час тестування системи було перевірено її працездатність, яка підтвердилася успішною роботою кожного з реалізованих модулів.

Проведений аналіз безпеки веб-додатків після інтеграції системи двофакторної автентифікації засвідчив її високу ефективність. Система продемонструвала надійний рівень захисту, стійкість до атак завдяки використанню перевірених алгоритмів і високий рівень контролю за подіями. Разом із тим були визначені аспекти, які потребують доопрацювання, зокрема підвищення зручності використання для кінцевих користувачів та оптимізація процесу інтеграції з іншими додатками.

Проведені вимірювання продуктивності системи засвідчили, що вона лише незначно впливає на час виконання запитів. У середньому різниця часу

виконання запитів становила 0,66599 мілісекунди, що є прийнятним компромісом між безпекою та продуктивністю. Система успішно виконує свої функції, не створюючи суттєвого навантаження на базу даних і незначно збільшуючи час виконання запитів.

Отже, можна зробити висновок, що мета роботи була повністю досягнута, а всі поставлені задачі – виконані. Впроваджена система двофакторної автентифікації успішно демонструє свою ефективність як інструмент для підвищення безпеки веб-додатків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. User Authentication Through Mouse Dynamics / Chao Shen та ін. IEEE Transactions on Information Forensics and Security. 2013. Т. 8, № 1. С. 16–30. URL: https://doi.org/10.1109/tifs.2012.2223677 (дата звернення: 04.11.2024).

2. An OWASP Top Ten Driven Survey on Web Application Protection Methods / O. B. Fredj та ін. Lecture Notes in Computer Science. Cham, 2021. С. 235–252. URL: https://doi.org/10.1007/978-3-030-68887-5_14

3. Antal M., Szabó L. Z. Biometric Authentication Based on Touchscreen Swipe Patterns. Procedia Technology. 2016. Т. 22. С. 862–869. URL: https://doi.org/10.1016/j.protcy.2016.01.061 (дата звернення: 08.11.2024).

4. R.S. D. S. Attendance Authentication System Using Face Recognition. Journal of Advanced Research in Dynamical and Control Systems. 2020. Т. 12, SP4. С. 1235–1248. URL: https://doi.org/10.5373/jardcs/v12sp4/20201599 (дата звернення: 22.11.2024).

5. Стасєв Ю. В., Гончаренко К. Г., Мороз В. І. Аналіз методу багатофакторної аутентифікації користувачів інформаційних систем на основі райдужної оболонки ока. Системи обробки інформації. 2023. № 3 (174). С. 63–69. URL: https://doi.org/10.30748/soi.2023.174.09 (дата звернення: 11.11.2024).

6. Recent Advances in User Authentication Using Keystroke Dynamics Biometrics / ред.: Y. Zhong, Y. Deng. Science Gate Publishing P.C., 2015. URL: https://doi.org/10.15579/gcsr.vol2 (дата звернення: 07.11.2024).

7. Lypa, B., Horyn, I., Zagorodna, N., Tymoshchuk, D., Lechachenko T., (2024). Comparison of feature extraction tools for network traffic data. CEUR Workshop Proceedings, 3896, pp. 1-11.

8. Mathew G., Thomas S. A Novel Multifactor Authentication System Ensuring Usability and Security. International Journal of Security, Privacy and Trust Management. 2013. Т. 2, № 5. С. 21–30. URL: https://doi.org/10.5121/ijsptm.2013.2503

9. ТИМОЩУК, Д., & ЯЦКІВ, В. (2024). USING HYPERVISORS TO CREATE A CYBER POLYGON. MEASURING AND COMPUTING DEVICES IN TECHNOLOGICAL PROCESSES, (3), 52-56.

10. A Systematic Literature Review of the Types of Authentication Safety Practices among Internet Users / K. Kovalan та ін. International Journal of Advanced Computer Science and Applications. 2021. Т. 12, № 7. URL: https://doi.org/10.14569/ijacsa.2021.0120792 (дата звернення: 08.11.2024).

11. ТИМОЩУК, Д., ЯЦКІВ, В., ТИМОЩУК, В., & ЯЦКІВ, Н. (2024). INTERACTIVE CYBERSECURITY TRAINING SYSTEM BASED ON SIMULATION ENVIRONMENTS. MEASURING AND COMPUTING DEVICES IN TECHNOLOGICAL PROCESSES, (4), 215-220.

12. Розробка веб-сайтів для початківців: HTML – CSS – JavaScript. Самвидав, 2022.

13. Database Management Systems (DBMS) Comparison: MySQL, Postgr. AltexSoft. URL: https://www.altexsoft.com/blog/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/ (дата звернення: 09.11.2024).

14. Tymoshchuk, D., Yasniy, O., Mytnyk, M., Zagorodna, N., Tymoshchuk, V., (2024). Detection and classification of DDoS flooding attacks by machine learning methods. CEUR Workshop Proceedings, 3842, pp. 184 - 195.

15. Omwoyo R., Kamau J., Mvurya M. A review of Two Factor Authentication Security Challenges in the Cyberspace | International Journal of Advanced Computer Technology. International Journal of Advanced Computer Technology. URL: https://ijact.org/index.php/ijact/article/view/112 (дата звернення: 14.11.2024).

16. RFC 6238: TOTP: Time-Based One-Time Password Algorithm / D. M'Raihi et al. IETF Datatracker. URL: https://datatracker.ietf.org/doc/html/rfc6238 (дата звернення: 15.11.2024).

17. Ahmed S., Mahmood Q. An authentication based scheme for applications using JSON web token. 2019 22nd International Multitopic Conference (INMIC),

м. Islamabad, Pakistan, 29–30 листоп. 2019 р. 2019. URL: https://doi.org/10.1109/inmic48123.2019.9022766

18. World Wide Waste: How Digital Is Killing Our Planet-And What We Can Do about It. Independent Publishing Network, 2020.

19. Заікіна Д., Глива В. Основи охорони праці та безпека життєдіяльності. RS Global S. z O.O., 2019. URL: https://doi.org/10.31435/rsglobal/001 (дата звернення: 27.11.2024).

20. Sureshbabu P., Sakthivadivu M. A Review on Biometrics Authentication System Using Fingerprint. Asian Journal of Computer Science and Technology. 2019. Т. 8, S1. С. 4–6. URL: https://doi.org/10.51983/ajcst-2019.8.s1.2016 (дата звернення: 17.11.2024).

21. Derkach M. V., Khomyshyn V. G., Hudzenko V. O. Testing the security of a web resource using tools for scanning and identifying vulnerabilities. Scientific news of Dahl university. 2023. URL: https://doi.org/10.33216/2222-3428-2023-25-1

22. Trade-off optimal decision of the problem of software system architecture choice / A. Kharchenko та ін. 2015 Xth International Scientific and Technical Conference "Computer Sciences and Information Technologies" (CSIT), м. Lviv, Ukraine, 14–17 верес. 2015 р. 2015. URL: https://doi.org/10.1109/stc-csit.2015.7325465

23. Kulchytskyi, T., Rezvorovych, K., Povalena, M., Dutchak, S., & Kramar, R. (2024). LEGAL REGULATION OF CYBERSECURITY IN THE CONTEXT OF THE DIGITAL TRANSFORMATION OF UKRAINIAN SOCIETY. Lex Humana (ISSN 2175-0947), 16(1), 443-460.

24. T. Lechachenko, R. Kozak, Y. Skorenkyy, O. Kramar, O. Karelina. Cybersecurity Aspects of Smart Manufacturing Transition to Industry 5.0 Model. CEUR Workshop Proceedings, 2023, 3628, pp. 325–329

25. Network Attack Detection Using Machine Learning Methods / N. ZAGORODNA та ін. Challenges to national defence in contemporary geopolitical situation. 2022. Т. 2022, № 1. С. 55–61. URL: https://doi.org/10.47459/cndcgs.2022.7

Додаток А – Публікація

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
імені ІВАНА ПУЛЮЯ
НАУКОВЕ ТОВАРИСТВО ім. ШЕВЧЕНКА

М А Т Е Р І А Л И

XII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

18-19 грудня 2024 року

ТЕРНОПІЛЬ
2024

УДК 001
М34

ПРОГРАМНИЙ КОМІТЕТ

Голова: Приймак Микола – професор кафедри комп'ютерних систем та мереж, д.т.н., професор.
Співголови: Марущак Павло – проректор з наукової роботи, докт. техн. наук, професор.
Баран Ігор – канд. техн. наук, доцент, декан факультету ФІС.
Науковий секретар: Надія Крива– старший викладач.

Члени: Василь Кривень - завідувач кафедри математичних методів в інженерії д.ф.-м.н., професор; Галина Осухівська – завідувач кафедри комп'ютерних систем та мереж, к.т.н., доцент; Микола Карпінський - професор кафедри кібербезпеки, д.т.н., професор; Жанна Баб'як - завідувач кафедри української та іноземних мов, к.пед. н., доцент; Ярослав Литвиненко – професор кафедри комп'ютерних наук, д.т.н., професор; Михайло Петрик - завідувач кафедри програмної іженерії, д.ф.-м.н., професор; Наталія Загородна – завідувач кафедри кібербезпеки, к.т.н., доцент.

ОРГАНІЗАЦІЙНИЙ КОМІТЕТ

Голова: Скоренький Юрій Любомирович – канд. техн. наук, доцент кафедри фізики.
Члени: доцент кафедри комп'ютерних наук, к.т.н. В. Никитюк; доцент кафедри програмної інженерії, к.т.н. Д. Михалик; доцент кафедри кібербезпеки, к.т.н. М. Стадник; доцент кафедри комп'ютерних систем та мереж, к.т.н. Є. Тиш; ст. викладач Л. Джиджора.

М34 Матеріали XII науково-технічної конференції «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя, (Тернопіль, 18–19 грудня 2024 р.). – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2024.

Адреса оргкомітету: ТНТУ ім. І. Пулюя, м. Тернопіль, вул. Руська, 56, 46001,
тел. (0352) 52-41-33, факс (0352) 25-49-83.
E-mail: conffis2024@gmail.com
Редагування, оформлення та верстка: Надія Крива

СЕКЦІЇ КОНФЕРЕНЦІЇ, ЯКІ ПРЕДСТВЛЕНІ В ЗБІРНИКУ

- Математичне моделювання
- Інформаційні системи та технології, кібербезпека
- Комп'ютерні системи та мережі
- Програмна інженерія та моделювання складних розподілених систем
- Новітні фізико-технічні та освітні технології

В збірнику надруковано тези доповідей XII науково-технічної конференції «Інформаційні моделі, системи та технології» (Тернопіль, 18–19 грудня 2024 р.) за такими науковими напрямками: математичне моделювання; інформаційні системи та технології, кібербезпека; комп'ютерні системи та мережі; програмна інженерія та моделювання складних розподілених систем; новітні фізико-технічні та освітні технології.

Розрахований на науковців, викладачів та студентів вузів.

За зміст тез та дотримання норм академічної доброчесності відповідальність несе автор.

УДК 004.056

І. Галанський
Тернопільський національний технічний університет імені Івана Пулюя, Україна

СИСТЕМА ЗАХИСТУ WEB-ЗАСТОСУНКІВ НА ОСНОВІ ДВОХФАКТОРНОЇ АВТЕНТИФІКАЦІЇ

I. Halanskyi

SYSTEM FOR PROTECTING WEB APPLICATIONS BASED ON TWO-FACTOR AUTHENTICATION

Двофакторна автентифікація (2FA) – це сучасний підхід до забезпечення інформаційної безпеки, що значно зменшує ризик несанкціонованого доступу до систем. Вона базується на використанні двох незалежних факторів для підтвердження особи користувача.

Було обрано алгоритм TOTP (Time-based One-Time Password) як ефективний спосіб впровадження другого рівня автентифікації. Цей метод базується на динамічній генерації одноразових кодів, які змінюються з певним часовим інтервалом[1].

У дослідженні розглянуто кілька аспектів впровадження системи двофакторної автентифікації:

1. Аналіз методів автентифікації. Розглянуто традиційні паролі, одноразові паролі (OTP) та біометричні дані. Традиційні паролі мають низький рівень безпеки через можливість їх перехоплення чи підбору. Біометричні методи, хоч і забезпечують високий рівень захисту, є дорогими та складними для інтеграції. Одноразові паролі, зокрема алгоритм TOTP, було обрано як найбільш збалансований варіант за рівнем безпеки, простотою впровадження та доступністю[2].

2. Розробка архітектури системи. Проєктування системи включало вибір технологій, що забезпечують інтеграцію з існуючими веб-додатками. Було обрано мову програмування PHP для серверної частини, MySQL як систему управління базами даних, а також бібліотеку PHPMailer для передачі одноразових паролів через електронну пошту.

3. Реалізація системи. На етапі реалізації було створено програмний модуль, що генерує TOTP-коди, надсилає їх користувачам та перевіряє коректність введених даних. Система забезпечує роботу з базою даних, де зберігаються тимчасові значення кодів та інформація про користувачів.

4. Тестування впровадженого рішення. Проведено тестування на стійкість до атак типу «людина посередник», фішингових атак та перехоплення сесій. Результати показали, що використання TOTP забезпечує високий рівень безпеки, мінімізуючи ризики компрометації облікових даних.

Система двофакторної автентифікації, створена на основі алгоритму TOTP, може бути адаптована для різних сфер застосування, таких як електронна комерція, банківські сервіси чи корпоративні мережі. Її впровадження дозволяє забезпечити високий рівень захисту інформації без значних витрат.

Література

1. RFC 6238: TOTP: Time-Based One-Time Password Algorithm. URL: https://tools.ietf.org/html/rfc6238 (дата звернення 14.11.2024)

2. Two-Factor Authentication (2FA): A Guide. URL: https://www.example.com (дата звернення 12.11.2024)

Додаток Б – Лістинги

index.php

```
<!--This page, index.php is the first page the users see upon accessing the site. Here is the logic for: checking the database for the user and authenticating him/her, generating an alphanumeric unique one-time passcode with the length of 6, sending this OTP to the user via email, and redirecting them to the otp.php page.-->
<?php
/*This function allows us to redirect the user to the next page
(otp.php) only if the user has been authenticated, found within our database, an OTP has been created,
and received a successful email. The header() function sends a raw HTTP header to the client. It must be called before any output is sent. We call exit() so that the script does not continue past the header function.*/
function redirect($url)
//https://www.exchangecore.com/blog/how-redirect-using-php
{
    header('Location: '.$url);
    exit();
}
//This function allows a session to be created. With sessions you can save information from this page and reference it from another.
session_start();
//Here we initialize these vairbales to null/empty. They will hold our error/success messages to be displayed to the user. Note that some of these variables will not be displayed to the user or they are only displayed for debugging purposes.
$usrerr = $emailerr = $emailsuccess = $pwerr = $bothEmpty = '';
//PHPMailer is an API to send emails within PHP applications. Here, we declare our application to use PHPMailer and to use its Exception package for potential errors. https://github.com/PHPMailer/PHPMailer
use PHPMailer\PHPMailer\PHPMailer;
```

```
    use PHPMailer\PHPMailer\Exception;
    //The file structure 'vendor/autoload.php' was generated by composer.
    require 'vendor/autoload.php';
    //This if statement only runs if the user has submitted their response via POST AND the username AND password fields were not submitted with empty values.
    if($_SERVER['REQUEST_METHOD'] == 'POST' && !empty($_POST['username']) && !empty($_POST['password'])) {
    //Here we declare our database (MariaDB/MySQL) information. You should change this to your connection info.
    $DATABASE_HOST = 'localhost';
    $DATABASE_USER = 'root';
    $DATABASE_PASS = '';
    $DATABASE_NAME = '';
    // Try and connect using the info above.
    $con = mysqli_connect($DATABASE_HOST, $DATABASE_USER, $DATABASE_PASS, $DATABASE_NAME);
    if ( mysqli_connect_errno() ) {
     // If there is an error with the connection, stop the script and display the error.
     exit('Failed to connect to MySQL: ' . mysqli_connect_error());
    }
    /* Preparing the SQL statement will prevent SQL injection because parameter values,
    which are transmitted later using a different protocol, need not be correctly escaped. If the original statement
    template is not derived from external input, SQL injection cannot occur.
https://www.w3schools.com/php/php_mysql_prepared_statements.asp */
    if ($stmt = $con->prepare('SELECT id, password, email FROM accounts WHERE username = ?')) {
     // Bind parameters (s = string, i = int, b = blob, etc), in our case the username is a string so we use "s"
     $stmt->bind_param('s', $_POST['username']);
     //Execute the SQL statement
```

```
$stmt->execute();
// Store the result so we can check if the account exists in the database.
$stmt->store_result();
//If the SQL statement returns 1 or more records, retain the id, password and email values where the username matches
if ($stmt->num_rows > 0) {
    $stmt->bind_result($id, $password, $email);
    $stmt->fetch();
    // Account exists, now we verify the password.
    // Password_verify is used to match the password_hash we used in our registration file.
    if (password_verify($_POST['password'], $password)) {
        // Verification success! User has loggedin!
        // Create sessions so we know the user is logged in. They are similar to cookies but remember the data on the server.
        session_regenerate_id();
        //We use this session to verify that the user has successfully logged in to be authorized to visit the otp.php page.
        $_SESSION['loggedin'] = TRUE;
        //We use this session to print the name of the user on the home.php page as well as validate the otp code with the associated username on the otp.php page
        $_SESSION['name'] = $_POST['username'];
        //The characters variable holds 0-9, a-z and A-Z as a string. This is used to create a random one time passcode for each user everytime they log in.
        $characters = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
        //First, we call str_shuffle on our characters variable to generate a random string. Then we call substr() to make our OTP 6 characters long, hence the -6.
        $otp = substr(str_shuffle($characters), -6);
        //Here we prepare the SQL statement to update the optcode column within our database based on the user's username.
```

```
            if ($stmt = $con->prepare('UPDATE accounts SET
otpcode = ? WHERE username = ?')) {
                $stmt->bind_param('ss', $otp,
$_POST['username']);
                $stmt->execute();
                $stmt->store_result();
            }
            //We declare a new PHPMailer object to send an email
to the user.
            $mail = new PHPMailer();
            //Declaring the mail protocol to be used (Simple Mail
Transfer Protocol)
            $mail->IsSMTP();
            //Debug arguement. 0 is false, 1 is true.
            $mail->SMTPDebug  = 0;
            //Authenticate the email before sending. (Your
email/where the email is from)
            $mail->SMTPAuth   = TRUE;
            //Secure method. Can be TLS or SSL
            $mail->SMTPSecure = "tls";
            //Port number for TLS
            $mail->Port       = 587;
            //SMTP server address from your email provider. For
gmail it is smtp.gmail.com
            $mail->Host       = "";
            //The email address used to send emails to users
            $mail->Username   = "";
            //Password of the email account
            $mail->Password   = "";

            //Send the email with HTML formatting.
            $mail->IsHTML(true);
            //Send the OTP email to the email address provided
from the user during registration
            $mail->AddAddress($email, $_POST['username']);
            //Set the from address and the email title
```

```
                $mail->SetFrom("", "AltName");
                //Set the reply-to field and resolve it to the name
"Sym"
                $mail->AddReplyTo("", "AltName");
                //Set the subject of the email
                $mail->Subject    =    "One-Time    Passcode    Login
Credentials";
                //Here is the body of the email. It tells the user to
enter the newly generated OTP code on the otp.php page
                $content = "Please enter this code on the 2FA Page:
" . $otp;

                //Wrap the content variable (body) into HTML
                $mail->MsgHTML($content);
                //Checks if the email did not successfully send
                if(!$mail->Send()) {
                          $emailerr = "Error while sending Email.";
                          var_dump($mail);
                } else {
                     //Email send successfully and redirect the user
to the otp.php page
                       $emailsuccess = "Email sent successfully";
                       redirect('otp.php');
                }

          } else {
                // Incorrect password
                $pwerr = "Incorrect username and/or password, please
try again!"; //Debugging purposes
          }
     } else {
          // Incorrect username
          $usrerr = "Incorrect username and/or password, please try
again!"; //Debugging purposes
     }
```

```
    //Close the databse stmt variable as we are no longer submitting
queries
     $stmt->close();
    }
    //This stmt only prints if you do not redirect the user to
otp.php. Used for debugging purposes.
    echo !empty($emailsuccess);
    }
    //Checks if the form was submitted with POST
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
     //Checks if the username OR password field was submitted empty
     if (empty($_POST["username"]) || empty($_POST["password"])) {
          $bothEmpty = "Please fill both the username and password
fields!";
       }
       //Checks if the email error variable has a value (not null)
      if (!empty($emailerr)) {
       $emailerr = "Error while sending Email.";
     }
     //Checks if the password error variable has a value (not null)
     if (!empty($stmterr)) {
          $pwerr = "     Incorrect username and/or password, please
try again!";
     }
     //Checks if the username error variable has a value (not null)
     if (!empty($usrerr)) {
          $usrerr ="     Incorrect username and/or password, please
try again!";
     }
    }
    ?>

    <!DOCTYPE html>
    <html>
     <head>
          <meta charset="utf-8">
```

```
        <title>Login</title>
        <!--Javascript used for the icons on the form-->
        <script   src="https://kit.fontawesome.com/9d50a4422d.js"
crossorigin="anonymous"></script>
        <!--Style  sheet  used  for  overall  formatting  of  the
webpages-->
        <link href="style.css" rel="stylesheet" type="text/css">
    </head>
    <body>
        <div class="bkg">
            <h1 class="lh1">Login</h1>
            <!--Each span statement is an error message, alerting
the user if the fields are empty or the information is incorrect.
            Not all of these message print because we re-route
the user to the otp.php page.-->
            <span class="er"><?php echo $pwerr;?></span>
            <span class="er"><?php echo $usrerr;?></span>
            <span class="er"><?php echo $bothEmpty;?></span>
            <span class="er"><?php echo $emailerr;?></span>
            <form     class="lfrm"     action="<?php     echo
htmlspecialchars($_SERVER["PHP_SELF"])?>" method="post">
                <label class="label" for="username">
                <i class="fas fa-user-secret"></i>
                </label>
                <!--For the value field, we run a php script that
will echo the user's input if any of the error message variables are
not empty.
                We echo the user's inputs so that they know what
they typed, with the exception of passwords since it is masked.-->
                <input class="ltxt" type="text" name="username"
placeholder="Username" id="username"
                value  =  "<?php  if  (!empty($bothEmpty)  ||
!empty($usrerr) || !empty($pwerr)) echo $_POST['username'];?>">
                <label class="label" for="password">
                    <i class="fas fa-lock"></i>
                </label>
```

```
                    <input       class="lpw"       type="password"
name="password" placeholder="Password" id="password"
                    value   =   "<?php   if   (!empty($bothEmpty)   ||
!empty($usrerr) || !empty($pwerr)) echo $_POST['password'];?>">
                    <!--Here we create the register button that
resets all the field variables and re routes the user to the
register.php page.-->
                    <button       class="side2"       type="reset"
onclick="location.href='register.php'">Register</button>
                    <input        class="side"        type="submit"
value="Submit">
                </form>
                <div id="back"></div>
                <div id="front"></div>
            </div>
     </body>
    </html>

    home.php

    <?php
    session_start();
    //We check if the session generaed on the OTP page is TRUE.
    if (isset($_SESSION['authorized']))
    //We create a new variable named "auth" to be used as conditional
within our form.
    $auth = $_SESSION['authorized'];
    else
    //If the session was not generated, set auth to empty. This can
happend if the user refreshes the page after successfully reaching
the home.php page.
    $auth = '';
    ?>
    <!DOCTYPE HTML>
    <html>
    <head>
```

```
        <meta charset="utf-8">
        <title>Home</title>
        <link                                    rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.1/css/all.css">
          <link       href="style.css"       rel="stylesheet"
type="text/css">
       </head>
    <body>
   <div class="register">
        <h1 class="rh1">Home</h1>
        <!--This is the homepage message that only shows if the
user has been properly authenticated!-->
        <span  class="msg"><?php  if  ($auth)  echo  'Hello  '  .
$_SESSION['name'] . ', you are authorized!';?></span>
        <!--This is the error message that only shows if the user
refreshes the page.-->
        <span  class="er"><?php  if  (empty($auth))  echo  "Opps!
Session has been lost. Please click on the Return button below and
log in again!";?></span>
        <br><br>
        <br><br>
        <!--Upon  clicking  the  return  button,  we  release  and
destroy the sessions saved.-->
        <?php session_unset(); session_destroy();?>
        <button            class="hbtn"            type="reset"
onclick="location.href='index.php'">Return</button>
          </div>
          <div id="back"></div>
        <div id="front"></div>
   </body>
   </html>

   register.php

   <!DOCTYPE HTML>
   <html>
```

```
    <head>
          <meta charset="utf-8">
          <title>Register</title>
          <link                                    rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.1/css/all.css">
            <link        href="style.css"        rel="stylesheet"
type="text/css">
        </head>
     <body>
    <?php

    //We defined variables that hold error messages and set to empty
values
    $usernameErr = $usernameVal = $usernameMatchErr = $emailErr =
$passwordErr =   $passwordMatchErr = $passwordVal = $fieldErr =
$regerr = $stmterrmsg = $success = "";
    //Variables that will hold our POST data from the user
    $username = $email = $password = "";
    //Checks if the user submitted the form
    if($_SERVER['REQUEST_METHOD'] == 'POST') {
    //Here we declare our database information. This would be the
same information provided on the index page. We could have optimized
our application by including this information in a "config.php" file
and call it each time we need database credentials.
    $DATABASE_HOST = 'localhost';
    $DATABASE_USER = 'root';
    $DATABASE_PASS = '';
    $DATABASE_NAME = '';
    // Try and connect using the info above.
    $con    =    mysqli_connect($DATABASE_HOST,    $DATABASE_USER,
$DATABASE_PASS, $DATABASE_NAME);
    if (mysqli_connect_errno()) {
     // If there is an error with the connection, stop the script
and display the error.
     exit('Failed to connect to MySQL: ' . mysqli_connect_error());
    }
```

```
    //This function trims any leading or trailing whitespaces, takes
off the "\" characters, and converts predefined characters such as
"<" to HTML entities such as "&lt"
    function test_input($data) {
      $data = trim($data);
      $data = stripslashes($data);
      $data = htmlspecialchars($data);
      return $data;
    }

    //Here we clean the POST data and assign it to our username,
password, and email variables
    $username = test_input($_POST["username"]);
    $email = test_input($_POST["email"]);
    $password = test_input($_POST["password"]);

    //Here we peform logic statements to make sure our users enters
everything in the correct format.
    //If any of the form fields are empty, the fieldErr vairable
will change from empty to an actual error message.
      if (empty($username) || empty($password) || empty($email)) {
        $fieldErr = "* All fields must be filled";
      }
      //If the username variable is empty and the previous fieldErr
variable isnt set to empty, set the usernameErr to hold an *. This
is echoed later for the user to understand what field the error was
produced in.
      if (empty($username) && !empty($fieldErr)) {
        $usernameErr = "*";
      }
      //Same idea as the previous if statement but for the email
field.
      if  (empty($email) && !empty($fieldErr)) {
        $emailErr = "*";
      }
```

```
      //Same idea as the previous if statement but for the password
field.
      if (empty($password) && !empty($fieldErr)) {
        $passwordErr = "*";
      }
      //Here we double check that the username field is not empty
AND the user has an input that only contains letters & numbers AND
the fieldErr variable is empty (to ensure the user only sees one
error message at a time)
      if     (!empty($username)     &&     !preg_match('/^[A-Za-z0-
9]*$/',($username)) && empty($fieldErr)) {
        //usernameMatchErr is set to hold an * to echo back to the
user where the error is located.
        $usernameMatchErr = "*";
        //usernameVal hold the actual error message as to why the
user is getting the error.
        $usernameVal = "* Username contains invalid characters. Only
letters and numbers are acceptable!";
      }
      //Here we used elseif to make sure the user only sees one error
message at a time. Same logic as the previous if statement except,
the password field can only contain numbers, letters, and the special
characters: ! @ # and $
      elseif    (!empty($password)    &&    !preg_match('/^[A-Za-z0-
9!@#$]*$/',($password)) && empty($fieldErr)) {
        $passwordMatchErr = "*";
        $passwordVal = "* Password contains invalid characters. Only
letters, numbers, and special characters ! @ # $ are acceptable!";
      }

      //This statement checks if the submission from the user is
POST, and if the username, password and email error variables are
empty, signifying that the input is clean enough to enter into our
databse
      //The logic of handling error messages could be optimized to
have all error messages in one variable.
```

```
    if ($_SERVER['REQUEST_METHOD'] == 'POST' && empty($fieldErr) &&
empty($usernameErr) && empty($emailErr) && empty($passwordErr) &&
empty($usernameMatchErr) && empty($passwordMatchErr)) {
    //Here we prepare an SQL statement to be executed against our
database.
      if ($stmt = $con->prepare('SELECT id, password FROM accounts
WHERE username = ?')) {
     // Bind parameters (s = string, i = int, b = blob, etc), hash
the password using the PHP password_hash function.
     $stmt->bind_param('s', $username);
     $stmt->execute();
     $stmt->store_result();
     // Store the result so we can check if the Username exists in
the database.
     if ($stmt->num_rows > 0) {
          //Username already exists. Users cannot create a username
that is already in our database.
        $registered = "Username already exists";
        //Here we check if the registered variable is not empty. We
created a new variable to hold the error message so that we can
simply echo it later within our HTML form.
        if (!empty($registered)) {
          $regerr = "Username already exists";
        }
     } else {
          // Username doesnt exists, insert new account
        if ($stmt = $con->prepare('INSERT INTO accounts (username,
password, email) VALUES (?, ?, ?)')) {
     //We do not want to expose passwords in our database, so we use
password_hash on the password field and use password_verify when a
user logs in on our index.php page.
       $password = password_hash($password, PASSWORD_DEFAULT);
       $stmt->bind_param('sss', $username, $password, $email);
        $stmt->execute();
        //Same idea as the registered logic in lines 87-89. This
indicates that the user has successfully registered.
```

```
        //Note, if the user has a trailing or leading space in their username or password, the registration will still be successful. However, our code trims these spaces so when the user tries to log in with the space character, it will not let them login. They must type their username/password with the characters not including the space. Same thing happens with the "\" character.
        $insert = "Registration success! Please login.";
        if (!empty($insert)) {
          $success = "Registration success! Please login.";
        }
      } else {
     // Something is wrong with the sql statement, check to make sure accounts table exists with all 3 fields.
      $stmterr = "Could not prepare statement within insert stmt";
      if (!empty($stmterr)) {
        $stmterrmsg = "Could not prepare statement";
      }
    }
      }
      //We close our database statement queries
     $stmt->close();
    } else {
     // Something is wrong with the sql statement, check to make sure accounts table exists with all 3 fields.
      $stmterr = "Could not prepare statement";
      if (!empty($stmterr)) {
        $stmterrmsg = "Could not prepare statement";
      }
    }
    //We close the databse connection
    $con->close();
    }
    }

    ?>
```

```
    <div class="register">
          <h1 class="rh1">Register</h1>
          <!--All error and success messages and appear right below the page title and above the form fields. Again, the way our backend logic is written, only one message appears at a time.-->
          <span class="er"><?php echo $fieldErr;?></span>
          <span class="er"><?php echo $regerr;?></span>
          <span class="er"><?php echo $stmterrmsg;?></span>
          <span class="er"><?php echo $usernameVal;?></span>
          <span class="er"><?php echo $passwordVal;?></span>
          <span class="success"><?php echo $success;?></span>

          <!--The php code within the action field sends the submitted form data to the page itself, instead of jumping to a different page. This way, the user will get error messages on the same page as the form. (https://www.w3schools.com/php/php_form_validation.asp)-->
          <form class="rform" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>" method="post">
                <label class="rlabel" for="username">
                     <i class="fas fa-user"></i>
                </label>
            <!--For the value field, we run a php script that will echo the user's input if the fieldErr variable is not empty or if our success variable is still empty. We echo the user's inputs so that they know what they typed, with the exception of passwords since it is masked.-->
                <input class="rinputxt" type="text" name="username" placeholder="Username" id="username" value = "<?php if (!empty($fieldErr) || empty($success)) echo $username?>">
            <!--The php scripts echo the *'s defined eariler in the code.-->
            <span class="er"><?php echo $usernameErr;?><?php echo $usernameMatchErr;?></span>
            <label class="rlabel" for="password">
            <i class="fas fa-shield-alt"></i>
```

```
                    </label>
                    <input    class="rinputpass"    type="password"
name="password" placeholder="Password" id="password" value = "<?php
if (!empty($fieldErr) || empty($success)) echo $password?>">
            <span class="er"><?php echo $passwordErr;?><?php echo
$passwordMatchErr;?></span>
            <label class="rlabel" for="email">
            <i class="fas fa-envelope-square"></i>
                    </label>
                <input       class="rinputemail"       type="email"
name="email"  placeholder="Email"  id="email"  value  =  "<?php  if
(!empty($fieldErr) || empty($success)) echo $email?>">
                <span class="er"><?php echo $emailErr;?></span>
                <!--Here we create the cancel button that resets all
the field variables and re routes the user to the index.php page or
login page.-->
                <button          class="rbtn"          type="reset"
onclick="location.href='index.php'">Cancel</button>
                    <input    class="rinputsubmit"    type="submit"
value="Register">
          </form>
          <!--These two divs are needed to show the moving bubble
background-->
          <div id="back"></div>
               <div id="front"></div>
          </div>
    </body>
    </html>

    mailer.php

    <?php
    //Use this code if your OTP email submission code is not working.
Code from https://github.com/PHPMailer/PHPMailer
    use PHPMailer\PHPMailer\PHPMailer;
    use PHPMailer\PHPMailer\SMTP;
```

```
    use PHPMailer\PHPMailer\Exception;
    require 'vendor/autoload.php';


    $mail = new PHPMailer(true);
    try {
    $mail->SMTPDebug = SMTP::DEBUG_SERVER;
    $mail->IsSMTP();
    $mail->SMTPAuth   = TRUE;
    //Change this if you are not use TLS
    $mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
    //Change this if you are not use TLS
    $mail->Port       = 587;
    //Enter your mail server's address. Example, gmail is
smtp.gmail.com
    $mail->Host       = "";
    //Enter your email address
    $mail->Username   = "";
    //Enter your email password
    $mail->Password   = "";

    $mail->IsHTML(true);
    $mail->AddAddress("receipient-email@email.com",
"ReceipientName");
    $mail->SetFrom("your-email@email.com", "YourName");
    //$mail->AddReplyTo("reply-to-email", "reply-to-name");
    //$mail->AddCC("cc-recipient-email", "cc-recipient-name");
    $mail->Subject = "Test is Test Email sent via ... SMTP Server
using PHP Mailer";
    $mail->Body = "<b>This is a Test Email sent via ... SMTP Server
using PHP mailer class.</b>";

    $mail->send();
    echo "Message has been sent";
    } catch (Exception $e) {
```

```
    echo "Message could not be sent. Mailer Error: {$mail->ErrorInfo}";
    }
    ?>

    otp.php

    <?php
    function redirect($url) //https://www.exchangecore.com/blog/how-redirect-using-php
    {
        header('Location: '.$url);
        exit();
    }
    //We need to declare this function in order to use sessions previously generated in our code.
    session_start();
    //Error/success message variables
    $otpErr = $registered = $otppErr = $inc = $suc = '';
    //We check if the form was submitted and the optcode field isn't empty.
    if($_SERVER['REQUEST_METHOD'] == 'POST' && !empty($_POST['otpcode'])) {
        $DATABASE_HOST = 'localhost';
        $DATABASE_USER = 'root';
        $DATABASE_PASS = '';
        $DATABASE_NAME = '';
        $con = mysqli_connect($DATABASE_HOST, $DATABASE_USER, $DATABASE_PASS, $DATABASE_NAME);
        if (mysqli_connect_errno()) {
            exit('Failed to connect to MySQL: ' . mysqli_connect_error());
        }
        //This if statement finds the account where the optcode and username matches in the database.
```

```
        if ($stmt = $con->prepare('SELECT id FROM accounts WHERE otpcode = ? AND username = ?')) {
            // Bind parameters (s = string, i = int, b = blob, etc)
            //We use the session we created on the index.php to verify that the user logged in can only use the code generated for their account.
            $stmt->bind_param('ss', $_POST['otpcode'], $_SESSION['name']);
            $stmt->execute();
            $stmt->store_result();
            // Store the result so we can check if the account exists in the database.
            if ($stmt->num_rows > 0) {
                // OTP exists
                session_regenerate_id();
                //Here we create a new session called "authorized" and set it to boolean value TRUE to validate that the user has been authorized to see the homepage message that only valid account holders can access!
               $_SESSION['authorized'] = TRUE;
                $registered = "Good"; //Debugging purposes
                redirect('home.php');
            }
            $otpErr = "Incorrect OTP";
        }
    }
    if ($_SERVER["REQUEST_METHOD"] == "POST") {
        //Tells the user that the OTP field is empty
        if (empty($_POST["otpcode"])) {
          $otppErr = "OTP field empty. Please enter your OTP Code";
        }
        //Alerts the user that the OTP they entered was incorrect.
        elseif (!empty($otpErr)) {
            $inc = "Incorrect OTP, please try again";
        }
        if (!empty($registered)) {
```

```
            $suc = "Success OTP"; //Debugging purposes
        }
    }
    ?>
    <!DOCTYPE html>
    <html>
     <head>
          <meta charset="utf-8">
          <title>Login</title>
            <link                                    rel="stylesheet"
href="https://use.fontawesome.com/releases/v5.7.1/css/all.css">
            <link         href="style.css"         rel="stylesheet"
type="text/css">
     </head>
     <body>
          <div class="bkg">
               <h1 class="lh1">2FA Login</h1>
                <span class="er"><?php echo $inc;?></span>
                <span class="er"><?php echo $otppErr;?></span>
                <span class="er"><?php echo $suc;?></span>
               <form      class="lfrm"      action="<?php      echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>" method="post">
                    <label class="label" for="otpcode">
                    <i class="fas fa-key"></i>
                    </label>
                    <input  class="ltxt"  type="text"  name="otpcode"
placeholder="Enter OPT Code" id="otpcode">
                    <!--Here we create the cancel button that resets
all the field variables and re routes the user back to the index.php
page or login page.
                    The user must log in again to navigate back to
the OTP page or they technically could type in the URL to the otp.php
page and enter their OTP code again.-->
                    <button        class="rbtn"        type="reset"
onclick="location.href='index.php'">Cancel</button>
```

```
                    <input        class="side"        type="submit"
value="Submit">

                </form>
                <div id="back"></div>
               <div id="front"></div>
          </div>
     </body>
    </html>
```