МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

МАГАЛЯС Роман Сергійович

Алгоритми ідентифікації користувачів веб-сайту / Website user identification algorithms

спеціальність: 123 - Комп'ютерна інженерія
освітньо-професійна програма - Комп'ютерна інженерія
Кваліфікаційна робота

Виконав студент групи КІм-21

Р.С. Магаляс

Науковий керівник:
к.т.н., доцент Л.О. Дубчак

ТЕРНОПІЛЬ – 2023

РЕЗЮМЕ

Кваліфікаційна робота на гему зі спеціальності 123 «Комп’ютерна інженерія» освітнього ступеня «магістр» написана обсягом 86 сторінок і містить 27 ілюстрацій, 8 таблиць, 4 додатка та 50 джерел за переліком посилань.

Метою роботи є розробка алгоритму ідентифікації користувачів веб- сайту на основі реєстрації та авторизації користувачів з покращеною безпекою даних.

Методи досліджень. Для розв’язання поставлених задач у кваліфікаційній роботі використано: методи аналізу алгоритмів їх візуалізації та моделювання, метод емпіричного тестування для аналізу роботи алгоритмів в реальних умовах, а також методи декомпозиції коду алгоритму та розробки програмного забезпечення.

Результати дослідження: алгоритм ідентифікації користувачів веб- сайту з урахуванням можливих покращень безпеки, програмна реалізація алгоритму в вигляді системи реєстрації та ідентифікації веб-сайту.

Результати роботи можуть бути використані при створенні веб- сайтів з алгоритмами ідентифікації.

Орієнтовні напрямки розвитку досліджень: оптимізація часу роботи алгоритмів ідентифікації користувачів з урахуванням вразливостей в безпеці, створення веб-сайтів з реєстрацією та авторизацією за покращеним алгоритмом та розробка засобів захисту від різноманітних атак з боку серверу, вивчення можливих вирішень проблем поведінки користувача які призводять до витоку даних.

КЛЮЧОВІ СЛОВА: АЛГОРИТМ, АНАЛІЗ, ВЕБ, ВЕБ-САЙТ ІДЕНТИФІКАЦІЯ, ПРОГРАМНА СИСТЕМА.

RESUME

The qualification work on the topic of the specialty 123 "Computer Engineering" of the educational degree "Master" is written with a volume of 86 pages and contains 27 illustrations, 8 tables, 4 appendices, and 50 sources in the list of references.

The purpose of the work is to develop an algorithm for user identification on a website based on user registration and authorization with enhanced data security.

Research methods. To solve the tasks set in the qualification work, the following methods were used: methods of analyzing algorithm visualization and modeling, empirical testing method for analyzing the operation of algorithms in real conditions, as well as methods of algorithm code decomposition and software development.

Research results: an algorithm for identifying website users taking into account possible security enhancements, software implementation of the algorithm in the form of a website registration and identification system.

The results of the work can be used in creating websites with identification algorithms.

Potential research directions: optimization of the operation time of user identification algorithms considering vulnerabilities in security, creation of websites with registration and authorization using an improved algorithm, and development of tools to protect against various server-side attacks, studying possible solutions to user behavior problems leading to data leaks.

KEYWORDS: ALGORITHM, ANALYSIS, WEB, WEBSITE, IDENTIFICATION, PROGRAMMING SYSTEM.

ЗМІСТ

ВСТУП

Актуальність роботи. Питання ідентифікації користувачів веб-сайтів залишається важливим в контексті безпеки та персоналізації послуг. З огляду на зростання кількості кіберзагроз і проблеми приватності даних, розробка сучасних та ефективних алгоритмів ідентифікації є необхідною.

Реєстрація та авторизація є одним з найпоширеніших алгоритмів ідентифікації користувачів на веб-сайтах, та залишається актуальним у світі за рядом чисельних причин: персоналізація сайтів, зручність для користувачів, маркетинг, аналітика веб-ресурсів, безпека даних.

Метою проекту є розробка алгоритму ідентифікації користувачів веб-сайту на основі реєстрації та авторизації користувачів з покращеною безпекою даних.

Задачами проекту є дослідити найбільш поширені методи ідентифікації користувачів, зайти можливі вразливості в безпеці та створити алгоритм з покращеною безпекою на основі цих даних.

В процесі досягнення відповідної мети будуть розв’язані такі задачі:

1. Проаналізовані варіанти ідентифікацій користувачів на веб-сайтах.
2. Знайдено та розібрано один з найпоширеніших алгоритмів реєстрації та авторизації користувачів.
3. Досліджено можливі вразливості алгоритму та підібрані відповідні запобіжні методи.
4. Створення власного алгоритму на основі вищезгаданого з урахуванням можливих вразливостей безпеки.
5. Реалізація алгоритму в вигляді коду веб-сайту та тестування його на практиці.

Об’єктом дослідження є веб-сайт з реєстрацією і авторизацією користувачів.

Предметом дослідження є методи і засоби забезпечення безпеки даних користувачів веб-сайтів.

Наукова новизна даного дослідження полягає в розробці алгоритму ідентифікації користувачів веб-сайту.

Практичне значення цього дослідження виявляється у розробці веб-сайту з реєстрацією і авторизацією користувачів та проведенні тестування їх надійності.

Публікація та апробація випускної кваліфікаційної роботи. Результати представлені на VIII Науково-практичній конференції «Інтелектуальні комп’ютерні системи та мережі».

У першому розділі кваліфікаційної роботи проведено аналіз сучасних алгоритмів ідентифікації користувачів на веб-сайтах. Проведено постановку задачі та формування вимог проекту.

У другому розділі розібрано стандартний алгоритм ідентифікації користувачів, проаналізовано його можливі вразливості з боку безпеки та на його основі розроблено покращений алгоритм реєстрації та авторизації. Розраховано його складність.

У третьому розділі розроблено систему ідентифікації користувачів на основі веб-сайту та форм його реєстрації та авторизації. Проведено тестування розробленого алгоритму, підведено підсумки розробки.

1 СУЧАСНІ АЛГОРИТМИ ІДЕНТИФІКАЦІЇ КОРИСТУВАЧІВ ВЕБ-САЙТУ

1.1 Поняття та історія ідентифікації користувачів

Ідентифікація користувачів - це процес визначення та підтвердження особистості персони, яка намагається отримати доступ до певних ресурсів, таких як комп'ютерна система, мережа, акаунт в Інтернеті або інших цифрових служб. Цей процес важливий для забезпечення безпеки і конфіденційності інформації, а також для впровадження правильних управлінських політик та обмежень доступу.

Ідентифікація особистості існує тисячі років. Вона може бути такою простою, як впізнавання обличчя вашого друга та виділення його з товпи, або такою складною, як перевірка ідентичності незнайомця на основі візерунків кровоносних судин на задній стороні їхніх рук.

Одним із варіантів давньої ідентифікації була наявність ювелірних виробів чи інших декоративних предметів. Найстаріші знахідки ювелірних виробів, які використовувались для ідентифікації, - це намистини, які були виявлені в Південній Африці, Алжирі та Ізраїлі. Найстаріші з них налічують приблизно 100 000 років тому. Щодо ідентифікації, ці намистини передавали іншим різноманітну інформацію, включаючи багатство, родинні зв'язки та особисту ідентичність. Навіть зараз у військових цілях використовуються прикраси у вигляді військових жетонів, а також в медичних цілях - медичні браслети для швидкої ідентифікації і класифікації осіб.

Іншим історичним варіантом, який замінив пам'ять, було татуювання. Найдавніші татуювання, виявлені в давній Європі та Єгипті, здавалося, мали терапевтичну мету. Але інші суспільства, такі як маорі Нової Зеландії, розвинули витончені видимі татуювання, які вказували на соціальний статус, походження та членство в певній групі.

Зі зростанням писемної мови та технологій ведення записів ідентифікація еволюціонувала від фізичних символів та шкіряних знаків до писемного слова. Найперший згаданий перепис або урядове збирання особистої інформації громадян датується 3800 р. до н. е. під час Вавилонської імперії, де зафіксовано, що перепис проводився кожні шість-сім років і включав детальні підрахунки як населення, так і ресурсів.

З часом Римська імперія розвинула більше методів збору даних і потребувала більше персоналізованої інформації від громадян, тому було введено різноманітні документи. Серед них були такі, як свідоцтва про народження, документи на власність землі та реєстраційні записи громадянства.

Щодо паспортів, заслугу за їх винайдення можна віддати королю Генріху V Англії в 1414 році, коли він створив документи для англійських громадян, які потребували довести свою ідентичність під час перебування в іноземних країнах. Ці папери потім отримали назву "документи безпечного проводу" і забезпечували безпеку громадянина в сусідній країні, коли їх видають монархом.

У 1829 році британський парламент прийняв реформи Роберта Піла, щоб більше наголосити на друкованих поліцейських записах. Завдяки цій новій увазі, дані могли зберігатися в особистому документальному файлі та пов'язуватися з особами за допомогою унікального числового значення. Це було передвісником більш сучасних урядових баз даних, які пов'язані з ID-картками.

Виходячи із реформ Роберта Піла, Нідерланди розпочали власну децентралізовану систему особистих номерів (PN) у 1849 році, але лише у 1940 році перейшли до видачі особистих ID-карток кожному громадянину. До цього часу Сполучені Штати розпочали розподіляти свої картки з номерами соціального страхування, перша партія роздавалася в 1936 році. Інші країни слідували прикладу, оскільки електронна обробка даних продовжувала проникати в країни та уряди по всьому світу.

У 1858 році сер Вільям Гершель здійснив біометричний прорив. Він успішно використовував чорнильні відбитки пальців як рукописні підписи в заповітах та угодах, роблячи це засобом точної ідентифікації. Ця практика

переросла у Гальтон-Генрі систему класифікації відбитків пальців Scotland Yard і пізніше була автоматизована японцями у 1980-х роках їхньою Автоматизованою системою ідентифікації відбитків пальців (AFIS) та подальше вдосконалена американцями Інтегрованою (AFIS).

Однак ці записи все ще були переважно в паперовому вигляді. Цифрові записи своїх паперових документів у США почали вести лише у 1977 році та створили програму відповідності, здатну перехресно посилати між різними банківськими та урядовими органами. Ця практика в кінцевому рахунку стала загальною.

Ця діджиталізація відкрила шлях для того, що багато людей зараз знають як смарт-карти, які спочатку набули популярності завдяки використанню урядом як національні ID-карти. Серед перших країн, які використовували смарт-карти, були Німеччина, Сінгапур, Чеська Республіка та Іспанія, що розпочали це впровадження наприкінці 1980-х років. Метою цих карток було об'єднання численних необхідних публічних служб в одному місці, включаючи громадянство, охорону здоров'я та фінанси. Карта містила різні дані, від дати народження і цифрових підписів до біометричних даних, таких як відбитки пальців.

Біометричні дані зробили великий стрибок, коли компанії та уряди почали використовувати нові методи для ідентифікації людей. У 2004 році США вперше запустив свої перші штатові автоматизовані бази даних відбитків долонь. Ці бази даних в основному використовувалися правоохоронними органами для зіставлення невідомих відбитків долонь зі списком відомих правопорушників.

Крім цієї технології, інші поліпшення у сфері біометрії включають розвиток розпізнавання мови, розпізнавання райдужок ока, розпізнавання обличчя, послідовність ДНК, геометрія руки та розпізнавання візерунків кровоносних судин, яке ґрунтується на візерунках судин рук.

У 2010 році в Індії вперше запустився найбільший у світі біометричний цифровий система ідентифікації. Система Aadhaar фіксує відбитки пальців та/або сканує іриси і присвоює унікальний 12-значний номер Aadhaar. За даними на

2019 рік, майже 1,2 мільярда людей добровільно зареєструвалися в програмі, яка призначена спростити та прискорити перевірку для урядових програм, а також зменшити ризик шахрайства.

Біометрична перевірка з'явилася на ринку споживчих товарів у 2013 році, коли Apple включила датчик відбитків пальців у iPhone 5S. Інші виробники смартфонів також долучилися до цього після. Пізніше Apple Touch ID було доповнено Face ID в iPhone X у 2017 році. Ці технології забезпечують досить непогані заходи безпеки без потреби в дратівливих паролях. Оскільки вартість такої безпеки продовжує знижуватися, біометрія має всі шанси стати стандартом на мобільних пристроях.

Сучасна ідентифікація користувачів може включати в себе використання різних методів перевірки особистості, таких як:

1. Ідентифікація за знаннями користувача
2. Ідентифікація за предметом
3. Ідентифікація за біометрією
4. Двофакторна аутентифікація (2FA)
5. Мультифакторна аутентифікація (MFA)

Ідентифікація за знаннями користувача: Наприклад, пароль або PIN-код.

Звідси іншим користувачам невідомо, який саме пароль має користувач, тому що він зберігається в зашифрованому вигляді. При авторизації система порівнює введений користувачем пароль із зашифрованою версією паролю, яка зберігається в базі даних. Якщо вони співпадають, користувач отримує доступ до системи.

Важливо зауважити, що безпека системи авторизації, яка використовує паролі, залежить від кількох факторів, таких як довжина та складність паролю, а також методи захисту від атак, наприклад, обмеження кількості невдалих спроб введення паролю (блокування акаунту після певної кількості невдалих спроб) та зберігання паролів в зашифрованому вигляді (наприклад, за допомогою хеш-функцій).

Однак паролі можуть бути вразливі до атак, таких як атаки "брутфорс" (намагання вгадати пароль, перебираючи всі можливі комбінації) та атаки по словнику (спроби вгадати пароль за допомогою популярних паролів або слів з словників). Тому рекомендується використовувати складні, унікальні паролі та змінювати їх регулярно, а також використовувати додаткові методи безпеки, такі як двофакторна аутентифікація, для забезпечення максимального рівня безпеки доступу до систем та акаунтів.

Ідентифікація за предметом: Наприклад, смарт-карта, токен, флеш-карта або інші фізичні об'єкти. Основні прилади та їх опис:

- Смарт-карти: Це картки з вбудованим чипом, який містить інформацію про користувача. Цей чип може містити паролі, ключі або інші конфіденційні дані, які дозволяють отримати доступ до системи або ресурсів. Приклад смарт-карти можна побачити на рисунку 1.1.

Рисунок 1.1 – Смарт-карта

- Токени: Токени - це малі електронні пристрої, які генерують одноразові паролі або коди доступу. Користувач може ввести цей одноразовий пароль, разом зі своїм основним паролем, для двофакторної аутентифікації.
- USB-токени: Подібні до звичайних токенів, але підключаються до комп'ютера через USB-порт. Вони можуть бути використані для генерації

одноразових паролів або для зберігання ключів доступу. Зазвичай виглядають як USB-носії.

- Мобільні пристрої: Сучасні мобільні телефони можуть слугувати засобом ідентифікації за допомогою різних методів, таких як відбиток пальця, розпізнавання обличчя або навіть спеціальні додатки для генерації одноразових паролів.
- RFID-карти: Ці карти містять мікросхему, яка може бути зчитана за допомогою радіочастотного сигналу. Вони часто використовуються для фізичної ідентифікації, наприклад, для відкриття дверей або контролю доступу до певних зон. Приклад RFID-карти, брелка та їх зчитувача можна побачити на рисунку 1.2.

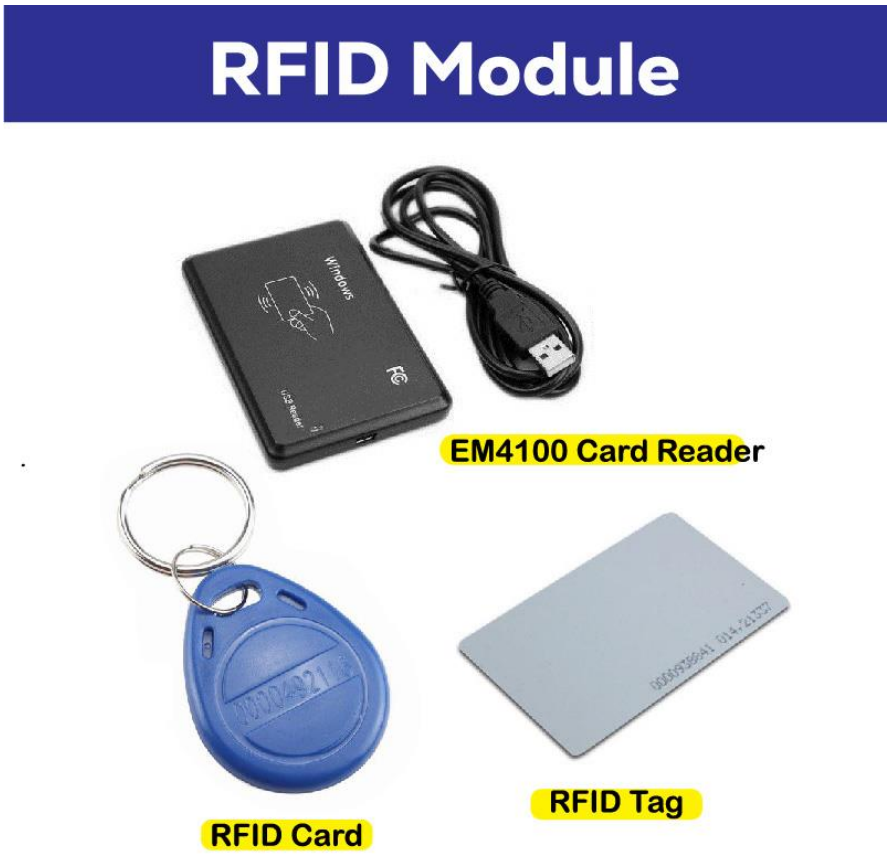

Рисунок 1.2 – RFID-пристрої

Ідентифікація за чимось, що користувач має, зазвичай є безпечнішою, ніж ідентифікація за чимось, що користувач знає, оскільки фізичні об'єкти важко підробити чи вгадати. Однак, важливо забезпечити безпеку цих фізичних об'єктів

і вживати заходів для їх захисту від втрати або крадіжки, щоб запобігти несанкціонованому доступу до систем та ресурсів.

Ідентифікація за біометрією: Наприклад, відбиток пальця, розпізнавання обличчя, розпізнавання голосу. Що значить використання унікальних фізичних рис користувача для підтвердження його ідентичності, опис основних методів:

- Відбиток пальця (біометрична ідентифікація пальця) - цей метод полягає у вимірюванні та аналізі унікальних фізичних характеристик відбитку пальця користувача. Сучасні пристрої, такі як смартфони та сканери входять в склад біометричних пристроїв, можуть зчитувати відбитки пальців і зіставляти їх з зареєстрованими в базі даних для ідентифікації особи.
- Розпізнавання обличчя (біометрична ідентифікація обличчя) - цей метод використовує унікальні риси обличчя, такі як форма очей, рота та носа, для ідентифікації користувача. Алгоритми глибинного навчання можуть аналізувати та порівнювати ці риси для визначення особи.
- Розпізнавання голосу (біометрична ідентифікація голосу) - цей метод використовує унікальні характеристики голосу, такі як тембр, інтонація та швидкість мовлення, для ідентифікації користувача. Голос може бути записаний і порівняний зі зразками в базі даних для підтвердження особи.

Біометричні методи ідентифікації повинні бути дуже надійними, оскільки вони базуються на унікальних фізичних характеристиках кожної особи. Проте вони також мають свої обмеження, такі як можливість неправильного визначення особи у певних умовах освітлення чи шуму в разі розпізнавання обличчя чи голосу. Також важливо забезпечити безпеку зберігання біометричних даних, оскільки вони є дуже особистою інформацією, яка може бути використана для зламу безпеки, якщо потрапляє в ненадійні руки.

На рисунку 1.3 зображено методи біометричної ідентифікації.

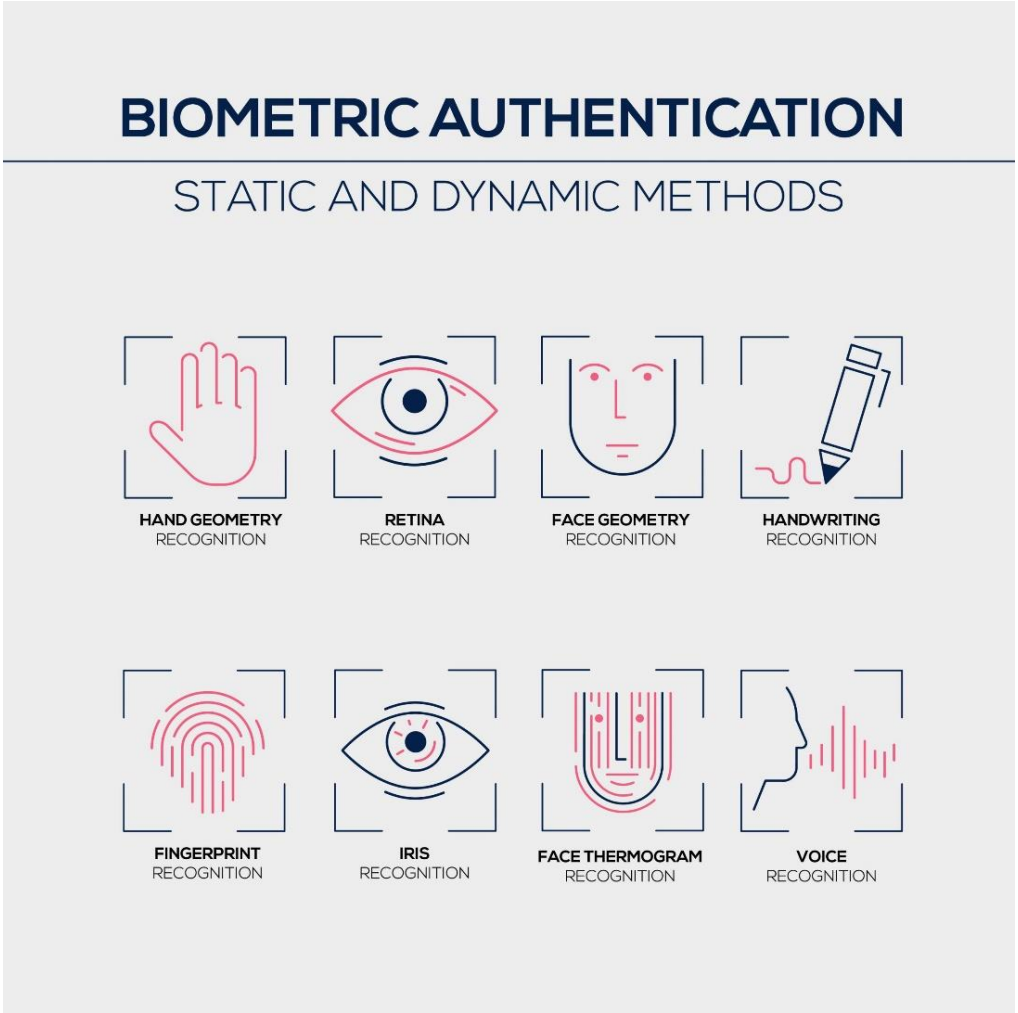

Рисунок 1.3 – Методи біометричної ідентифікації

Двофакторна аутентифікація (2FA): Використання двох або більше методів ідентифікації для забезпечення більшого рівня безпеки. Наприклад, комбінація пароля і підтвердження через мобільний додаток або SMS-повідомлення.

Коли користувач намагається увійти до системи, йому спершу потрібно ввести основний пароль. Після цього система запитує додатковий одноразовий код, який генерується токеном або мобільним додатком. Цей код зазвичай дійсний лише декілька хвилин і не може бути використаний знову.

2FA робить процес аутентифікації набагато безпечнішим, оскільки навіть якщо зловмиснику вдасться отримати доступ до основного пароля, йому все одно буде важко отримати доступ до облікового запису без додаткового одноразового коду або фізичного токену. Це важливо для запобігання несанкціонованому

доступу, особливо в умовах, коли основний пароль може бути викрадений або вгаданий.

Мультифакторна аутентифікація (MFA): Використання трьох або більше методів ідентифікації для забезпечення максимальної безпеки.

Це розширений варіант двофакторної аутентифікації, який використовує три або більше різних методів для перевірки особистості користувача. Ці методи можуть включати в себе те, що користувач знає, те, що він має, і його біометричні данні.

Ідентифікація користувачів є важливою частиною інформаційної безпеки, оскільки вона допомагає уникнути несанкціонованого доступу до чутливої інформації та може запобігти кібератакам та шахрайствам.

1.2 Сучасні алгоритми ідентифікації веб-сайтів

Ідентифікація користувачів на веб-сайтах є важливою для безпеки, персоналізації та надання різних сервісів. Вона дозволяє обмежувати доступ до конфіденційної інформації, захищати облікові записи від несанкціонованого доступу та забезпечувати безпеку користувачів. Ідентифікація також дозволяє персоналізувати взаємодію з сайтом, надаючи користувачам контент, який відповідає їхнім інтересам і потребам. Крім того, вона сприяє зручності користування сайтом, зберігаючи налаштування та уподобання користувачів. Ідентифікація також є важливою для аналітики, оскільки дозволяє веб-сайтам відстежувати користувацьку активність і аналізувати взаємодію з аудиторією. Крім того, вона може сприяти спілкуванню та співпраці між користувачами, створюючи можливості для обміну повідомленнями та співпраці в різних проектах. Ідентифікація користувачів є ключовим елементом в онлайн-середовищі, забезпечуючи якість обслуговування та взаємодії між веб-сайтом і його користувачами.

Багато сайтів використовують ідентифікацію за логіном і паролем для забезпечення безпеки та обмеження доступу до особистої інформації користувачів. Це стандартний метод аутентифікації в інтернеті. Однак, безпека цього методу може залежати від рівня складності паролю, а також від технічних засобів захисту, які використовуються сайтом.

Будь-який веб-сайт, який пропонує особисті облікові записи або доступ до конфіденційної інформації, буде використовувати ідентифікацію за логіном і паролем. Це можуть бути:

1. Соціальні мережі: Facebook, Twitter, Instagram і т.д.
2. Електронна пошта: Gmail, Yahoo Mail, Outlook і інші.
3. Онлайн-магазини: Amazon, eBay, Alibaba тощо.
4. Фінансові установи: Банки, платіжні системи, інвестиційні платформи.
5. Освітні платформи: Університети, онлайн-курси.
6. Закриті форуми та спільноти: Reddit, Stack Exchange і інші.
7. Робочі інструменти: Slack, Microsoft Teams, Google Workspace тощо.
8. Застосунки для знайомств і спілкування: Tinder, Badoo, Bumble.

Це лише кілька прикладів. Практично будь-який веб-сайт, який вимагає від користувачів створення облікового запису або надає особистий доступ до інформації, може використовувати ідентифікацію за логіном і паролем.

В додачу до логіну та пароля більшість сайтів з конфіденційними даними підтримують можливість двофакторної ідентифікації (2FA). Двофакторна ідентифікація важлива через те, що вона впроваджує додатковий шар захисту в інтернеті. Коли користувач вводить свій пароль, система запитує додатковий фактор ідентифікації, такий як спеціальний код, який надсилається на мобільний телефон або іншими способами. Це означає, що, навіть якщо хтось отримає доступ до вашого пароля, він не зможе увійти в обліковий запис без додаткового фактору ідентифікації.

Це дуже важливо в умовах сучасного інтернету, де загрози кібербезпеки стають все більш винахідливими і складними. Атаки фішингу, віруси, шкідливі

програми - усе це може призвести до неправомірного доступу до вашого облікового запису. Додатковий фактор ідентифікації ускладнює завдання зловмисникам, забезпечуючи більш високий рівень безпеки для ваших особистих даних та конфіденційної інформації.

Двофакторна ідентифікація використовує два різних методи аутентифікації для підтвердження особистості користувача. Ось кілька варіантів 2FA:

1. СМС-коди: Система надсилає спеціальний код на мобільний телефон користувача через SMS.
2. Подача голосового дзвінка: Користувач отримує голосовий дзвінок, який містить код підтвердження.
3. Мобільні застосунки: Користувач використовує спеціальні мобільні додатки (наприклад, Google Authenticator, Authy) для отримання часових одноразових кодів.
4. Електронна пошта: Система висилає код аутентифікації на електронну пошту користувача.
5. Апаратні ключі: Користувач користується спеціальним апаратним ключем, який генерує коди аутентифікації.
6. Біометричні дані: Система використовує біометричні дані користувача, такі як відбиток пальця чи скан обличчя, для підтвердження особистості.

Ці варіанти можуть використовуватися окремо або в комбінації для забезпечення додаткового рівня безпеки для облікових записів користувачів.

Також сучасні сайти підтримують ідентифікацію за допомогою Google акаунту або акаунті соціальних мереж. Приклад такого варіанту авторизації можна побачити на рисунку 1.4. При цьому Google або соціальна мережа через яку здійснюється авторизація надає сайту необхідну інформацію і на основі неї формується акаунт всередині сайту. Ідентифікація на сайті за допомогою Google акаунту та акаунтів соціальних мереж має як переваги, так і ризики вони описані в таблиці 1.1.

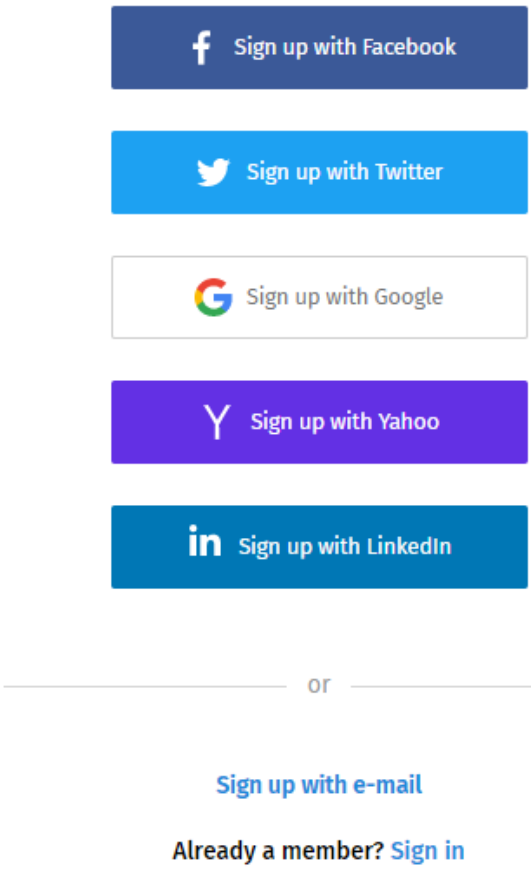

Рисунок 1.4 – Сторінка авторизація через соціальні мережі

Таблиця 1.1 Переваги та ризики авторизації через соціальні мережі

Аспекти	Переваги	Ризики
1	2	3
Зручність для користувачів	Швидкий та простий доступ, відсутність необхідності запам'ятовувати нові логіни і паролі	Можливість витоку особистої інформації, якщо сайт не захищений
Швидкий доступ	Відсутність необхідності вводити логін і пароль	Залежність від сторонніх сервісів, можливий витік даних

Продовження таблиці 1.1

1	2	3
Зменшення ризику витоку паролів	Зменшення ймовірності використання одного пароля на різних сайтах	Можливість компрометації основного акаунту на Google або соціальних мережах
Спам і персоналізована реклама	Якщо сайт має доступ до профілю користувача, може надавати персоналізовані послуги	Спам і надмірна реклама, якщо дані використовуються недобросовісним чином
Залежність від сторонніх сервісів	Зменшення необхідності у власному системі авторизації, менше проблем для користувачів	Втрата доступу в разі проблем або відключення сервісу Google або соціальної мережі
Конфіденційність даних	Потенційно вищий рівень безпеки, якщо сайт забезпечений належними заходами захисту	Ризик витоку конфіденційної інформації, якщо сайт недостатньо захищений

Ця таблиця надає загальний огляд переваг та ризиків ідентифікації на сайті через Google акаунт та акаунти соціальних мереж. Однак потрібно пам'ятати, що безпека в Інтернеті - це спільна відповідальність користувачів та розробників, тому завжди слід бути обережними з невідомими сайтами.

Ще одним способом ідентифікації користувачів на веб-сайтах є кукі-файли (або cookies). Це невеликі текстові документи, які веб-сайти зберігають на комп'ютері користувача чи іншому пристрої для збереження інформації про користувача. Ця інформація може включати в себе дані про сесію, налаштування

сайту, інформацію про авторизацію та інші дані, які допомагають веб-сайтам взаємодіяти з користувачами та пам'ятати їхні вибори.

Кукі використовуються для кількох цілей:

1. Збереження сесії: Кукі можуть зберігати інформацію про сесію, що дозволяє користувачам залишатися залогіненими на веб-сайтах, не вводячи свої дані кожен раз, коли вони переходять з однієї сторінки на іншу.

2. Запам'ятовування налаштувань: Вони можуть зберігати індивідуальні налаштування, наприклад, мову, шрифт чи інші налаштування вигляду веб-сайту, які користувач вибрав.

3. Аналітика: Вони можуть використовуватися веб-сайтами для збору даних про користувачів, таких як відвідувані сторінки, час перебування на сайті, джерело трафіку тощо. Ця інформація може бути корисною для вдосконалення веб-сайту та його вмісту.

4. Рекламні цілі: Кукі можуть використовуватися для показу користувачам спеціалізованої реклами, враховуючи їхні інтереси на основі їхньої активності на веб-сайті.

5. Запам'ятовування товарів в кошику: На інтернет-магазинах кукі можуть використовуватися для того, щоб запам'ятовувати товари, які користувач додав до свого кошика під час покупок.

6. Безпека: Кукі можуть допомагати виявляти та запобігати шахрайській активності, такі як злам сайту чи спроби несанкціонованого доступу до облікових записів користувачів.

Важливо відзначити, що кукі можуть бути використані як зручний інструмент для покращення користувацького досвіду в Інтернеті, але вони також можуть бути використані для відстеження користувачів без їхньої згоди. Чимало веб-сайтів і браузерів надають можливість користувачам керувати тим, які дані зберігаються в кукі-файлах і як вони використовуються.

1.3 Постановка задачі, формування вимог

В даній магістерській роботі розробляється алгоритм ідентифікації користувачів веб-сайту. Він забезпечує безпечну ідентифікацію користувачів веб-сайту. Дозволяє безпечно зберігати користувацькі дані в системі. Ідентифікації користувача в даній роботі реалізовувалась на базі найпоширеніших технологій веб-розробки PHP та MySQL.

PHP (PHP: Hypertext Preprocessor) - це популярна скриптова мова програмування загального вжитку, яка використовується для розробки веб-застосунків. Вона спеціально створена для веб-розробки та може вбудовуватися безпосередньо в HTML-код. PHP легко вивчається і має велику спільноту розробників, що сприяє швидкому розвитку проектів. Переваги та недоліки PHP можна побачити в таблиці 1.2.

Таблиця 1.2 – Переваги та недолі мови програмування PHP.

Характеристика	Переваги	Недоліки
1	2	3
Легкість вивчення та використання	Простий синтаксис, що полегшує вивчення та швидку розробку.	Недосконаліть синтаксису, який може призводити до помилок.
Широкі можливості	Розширена функціональність для взаємодії з базами даних, роботи з формами, обробки зображень тощо.	Деякі функції можуть мати нестабільну роботу або дублюватися.
Загальна популярність	Велика спільнота користувачів і розробників, що означає велику кількість документації та готових рішень.	Періодичні конфлікти через різні версії PHP та використання застарілих практик.

Продовження таблиці 1.2

1	2	3
Інтеграція з великою кількістю серверів та баз даних	Сумісність з багатьма веб-серверами та підтримка багатьох систем управління базами даних (наприклад, MySQL).	Деякі нові технології можуть не мати повної підтримки.
Зручність веб-розробки	Можливість вбудовування PHP-коду безпосередньо в HTML, що полегшує створення динамічних сторінок та шаблонів.	Проблеми з читабельністю коду через велику кількість вбудованого PHP в HTML.

Однією з основних переваг PHP є висока безпека при вірному використанні. Ось декілька аспектів, які сприяють безпеці в PHP:

1. Санітарний код: PHP має вбудовані функції для очищення та перевірки даних, що допомагає у запобіганні атакам типу SQL-ін'єкцій та XSS (Cross-Site Scripting).

2. Захист від взаємодії з файловою системою: PHP забезпечує можливість обмежувати доступ до файлів та директорій на рівні сервера, що допомагає у запобіганні несанкціонованому доступу до файлової системи.

3. Сесії та кукі: PHP дозволяє безпечно управляти сесіями та кукі, забезпечуючи надійну автентифікацію та авторизацію користувачів.

4. Шифрування та хешування: PHP має вбудовані функції для роботи з шифруванням та хешуванням даних, що сприяє безпеці під час зберігання чутливої інформації в базі даних.

У висновку, PHP, при правильному використанні, може забезпечити високий рівень безпеки для веб-застосунків. Важливо дотримуватися кращих практик програмування та використовувати доступні інструменти для захисту від потенційних загроз.

Зважаючи на те, що MySQL є однією з найпопулярніших систем управління базами даних (СУБД) на ринку, нижче наведено таблицю 1.3, яка

представляє переваги та недоліки цієї системи управління базами даних.

Таблиця 1.3 – Переваги та недоліки СУБД MySQL

Характеристика	Переваги	Недоліки
Швидкодія	Висока швидкість роботи завдяки оптимізованому двигуну бази даних.	При великому обсязі даних може спостерігатися втрата швидкодії, яку можна вирішити за допомогою оптимізації запитів і індексів.
Надійність	Надійність та стабільність роботи системи, підтримка транзакцій та відновлення бази даних після збоїв.	Не завжди надійна підтримка гарячої заміни апаратного забезпечення та можливості масштабування горизонтально.
Гнучкість	Широкі можливості для налаштування та конфігурації бази даних, можливість розширення функціоналу за допомогою збережених процедур, тригерів, функцій та інших об'єктів бази даних.	Спостерігається обмежена підтримка деяких сучасних функцій, які пропонують інші СУБД.
Сумісність	Підтримка мов SQL та стандартних інтерфейсів для взаємодії з базою даних, що полегшує інтеграцію з іншими системами та додатками.	Є деякі відмінності у синтаксисі SQL порівняно з іншими СУБД, що може вимагати внесення змін у запити при переносі на інші платформи.
Спільнота	Велика та активна спільнота користувачів та розробників, яка забезпечує широку підтримку, багато додаткових модулів та засобів для вирішення різних завдань.	Офіційна підтримка та оновлення можуть вимагати комерційної підтримки, що може бути вартістю для підприємств і великих проектів.

MySQL також має велику увагу до безпеки даних. Ось кілька способів, якими MySQL забезпечує безпеку:

1. Автентифікація та авторизація: MySQL надає різні методи автентифікації, включаючи використання паролів, ключів та інших методів. Авторизація контролює доступ користувачів до бази даних та її об'єктів.

2. Шифрування даних: MySQL підтримує шифрування даних під час передачі через мережу, що запобігає перехопленню та читанню конфіденційної інформації.

3. Аудит та журналювання: MySQL може вести журнали подій (logs) та аудит всіх активностей користувачів, що дозволяє виявляти та реагувати на потенційні загрози безпеки.

4. Оновлення безпеки: Команда розробників MySQL постійно виправляє виявлені вразливості безпеки через регулярні патчі та оновлення, щоб забезпечити користувачам безпеку їхніх даних.

Ці заходи сприяють створенню безпечних та надійних додатків з використанням MySQL як системи управління базами даних.

PHP і MySQL є надзвичайно сумісними інструментами для веб-розробки. PHP забезпечує можливість створювати динамічний вміст на веб-сайтах, тоді як MySQL дозволяє зберігати та керувати даними. Оскільки PHP і MySQL розвиваються як відкриті технології, вони мають велику спільноту користувачів, яка активно співпрацює над вирішенням проблем та покращенням цих систем.

Ця сумісність забезпечує надійність у розробці веб-додатків, де PHP відповідає за логіку та взаємодію з користувачем, а MySQL - за безпечне та ефективне зберігання даних. Спільне використання цих засобів дозволяє розробникам створювати швидкі, масштабовані та безпечні веб-додатки для користувачів.

1.4 Висновки до розділу 1

У даному розділі проведено аналіз сучасних засобів та методів ідентифікації користувачів веб-сайтів. Здійснено дослідження сучасних алгоритмів ідентифікації, виділено їх переваги та недоліки. А також проведено постановку задачі та формулювання вимог до пропонованого алгоритму ідентифікації користувачів.

2 АЛГОРИТМ ІДЕНТИФІКАЦІЇ КОРИСТУВАЧІВ ВЕБ-САЙТУ

2.1 Стандартний алгоритм ідентифікації користувачів

На сьогодні реєстрація за допомогою електронної пошти та пароля є одним з найпоширеніших методів ідентифікації користувачів на веб-сайтах. Цей процес передбачає створення облікового запису користувача, який буде використовуватися для доступу до різних функцій та сервісів веб-сайту. Це забезпечує безпеку та конфіденційність інформації користувача, оскільки лише власник облікового запису може мати доступ до особистих даних та інших приватних ресурсів сайту.

Першим кроком у процесі реєстрації на сайті є заповнення форми реєстрації, форма реєстрації яка була використана в даному проекті можна побачити на рисунку 2.1. А саме ведення електронної пошти, яка слугує унікальним ідентифікатором користувача. Після цього користувач повинен вибрати пароль, який буде захищати його обліковий запис. Щоб забезпечити безпеку, рекомендується використовувати складний пароль, який містить комбінацію букв, цифр та спеціальних символів. Крім того, деякі веб-сайти також вимагають від користувачів підтвердження паролю, щоб уникнути помилок при його введенні.

Після успішної реєстрації дані користувача, такі як електронна пошта та зашифрований пароль, зберігаються у базі даних веб-сайту. Коли користувач намагається увійти на сайт, він вводить свою електронну пошту та пароль. Система порівнює ці дані з інформацією, збереженою в базі даних. Якщо вони збігаються, користувач отримує доступ до облікового запису і може користуватися всіма можливостями веб-сайту. Якщо ж дані не збігаються, користувач повторно зобов'язаний ввести правильну інформацію або скористатися опцією відновлення пароля, яка забезпечує безпеку та захищає від несанкціонованого доступу до облікового запису користувача.

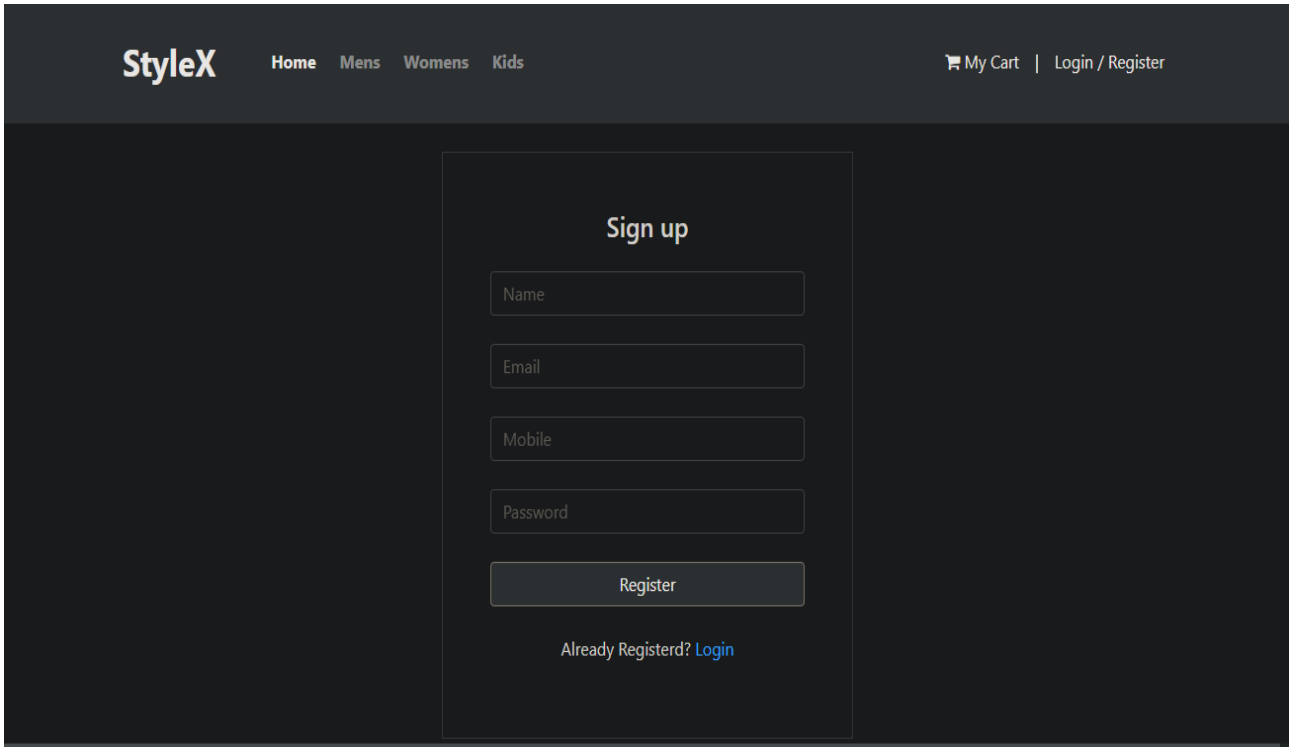

Рисунок 2.1 – Форма реєстрації користувача

Стандартний код алгоритму ідентифікації користувачів можна знайти в додатку 1. Цей алгоритм коду виконує перевірку логіну та пароля, які були введені у форму. Ось поетапний опис його роботи:

1. Перевірка наявності відправленої форми: Код спочатку перевіряє, чи була натискана кнопка "submit" у формі. Це робиться за допомогою isset($_POST['submit']).

2. Отримання та обробка даних з форми: Якщо форма була відправлена, скрипт отримує значення електронної пошти ($email) та паролю ($password), які були введені у форму.

3. Хешування паролю: Пароль хешується за допомогою функції md5(), щоб забезпечити безпеку даних перед збереженням у базі даних.

4. Підготовка та виконання SQL запиту: Запит SQL готується, використовуючи отримані значення електронної пошти та хешованого паролю. Запит шукає користувача в базі даних, який має однакову електронну пошту та пароль.

5. Виконання SQL запиту: Готовий SQL запит виконується за допомогою об'єкта підготовленого запиту.

6. Перевірка результатів: Результати запиту перевіряються за допомогою $query->rowCount(). Якщо знайдено користувача зі співпадаючими електронною поштою та паролем, то:

a. Установка сесійних змінних: Сесійні змінні $_SESSION['login'] та $_SESSION['user'] встановлюються з відповідними значеннями з бази даних.

b. Повідомлення користувачеві: Виводиться повідомлення про успішний вхід за допомогою JavaScript.

c. Перенаправлення користувача: Користувач перенаправляється на сторінку "index.php" за допомогою JavaScript.

7. Обробка невірних даних: Якщо ж знайдено жодного користувача зі співпадаючою електронною поштою та паролем, виводиться повідомлення про невірні дані за допомогою JavaScript.

Цей алгоритм перевіряє введені користувачем дані, шукає відповідний запис у базі даних та надає доступ до системи, якщо введені дані є вірними. Якщо ж дані невірні, користувач отримає повідомлення про помилку. Блок-схему цього алгоритму роботи можна побачити на рисунку 2.2.

Цей код має кілька потенційних проблем з безпекою, які можуть призвести до серйозних вразливостей. Ось деякі з них:

1. Використання md5 для хешування паролю: Функція md5() стара та вже не рекомендується для застосування в системах безпеки, оскільки її можна дуже легко зламати за допомогою атаки в переборі (brute force). Рекомендується використовувати більш сучасні та безпечні алгоритми хешування, такі як bcrypt або Argon2.

2. SQL ін'єкція: Запит SQL побудований за допомогою конкатенації рядків, що може зробити його вразливим до SQL ін'єкцій, якщо вхідні дані не санітаризовані. Рекомендується використовувати підготовлені запити (prepared statements) або об'єктно-орієнтовані методи для взаємодії з базою даних.

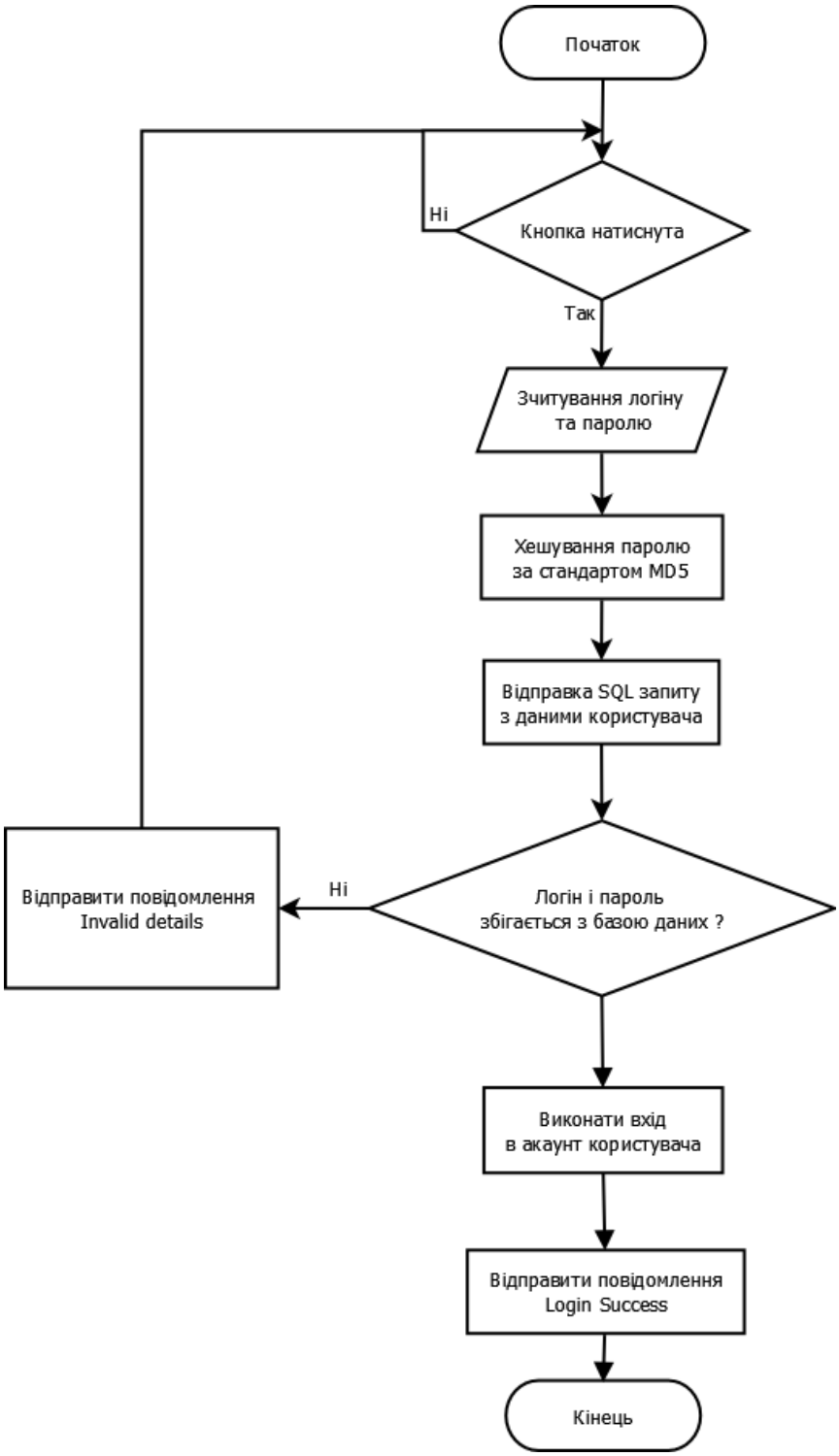

Рисунок 2.2 – Блок-схема стандартного алгоритму ідентифікації користувачів

3. Недостатня валідація вхідних даних: Код не має достатньої валідації вхідних даних перед їх обробкою. Це може призвести до введення неправильних або шкідливих даних в систему.

4. Виведення повідомлень про помилки з JavaScript: Виведення повідомлень про помилки безпеки (таких як "Невірний пароль" або "Невірна електронна пошта") за допомогою JavaScript може надати інформацію злоумисникам і спростити атаки.

5. Брак обробки помилок: Код не має належного обробки помилок під час взаємодії з базою даних. Це може призвести до витоку чутливої інформації або непередбачуваних ситуацій.

6. Відсутність затримки між невдалою аутентифікацією і наступним запитом у формі входу. Це спрощує зловмисникам проводити атаки з перебором, а також збільшує навантаження на сервер та забруднює журнали.

Відповідно до коду логіну було розглянуто код реєстрації користувачів. Сам код можна знайти в додатку 1. Поетапний опис алгоритму коду:

1. Перевірка натискання кнопки "submit":

- Перевіряється, чи було натискання кнопки "submit" у реєстраційній формі.

2. Отримання введених даних з форми:

- Якщо форма була відправлена, отримуються значення ім'я користувача ($name), мобільний номер ($mobile), електронна пошта ($email) та пароль ($password), які були введені у форму.

3. Хешування паролю:

- Пароль хешується за допомогою функції md5(), щоб забезпечити безпеку даних перед збереженням у базі даних.

4. Підготовка та виконання SQL запиту:

- Запит SQL готується для вставки нового користувача в базу даних. Параметри запиту (:name, :email, :mobile, :password) замінюються реальними значеннями.

- Після підготовки запит виконується за допомогою об'єкта підготовленого запиту.

5. Перевірка результату вставки в базу даних:

- Отримується ідентифікатор останнього вставленого запису ($lastInsertId). Якщо вставка в базу даних була успішною:

a. Установка сесійної змінної та виведення повідомлення:

- Сесійна змінна $_SESSION['login'] встановлюється значенням ім'я користувача з форми.
- Виводиться повідомлення про успішну реєстрацію користувача за допомогою JavaScript.

b. Обробка невдалої вставки:

- Якщо вставка в базу даних невдала, виводиться повідомлення про невірно введені дані за допомогою JavaScript.

Цей алгоритм дозволяє реєструвати нових користувачів у системі, перевіряючи правильність їхніх даних, хешуючи пароль та виконуючи вставку в базу даних.

Цей код відповідно до коду форми входу має ті ж проблеми, та навіть збільшує їх число. Наприклад в коді немає перевірки на співпадіння електронної пошти з вже зареєстрованими користувачами.

2.2 Вдосконалений алгоритм ідентифікації користувачів веб-сайту

Відповідно до розділу 2.1 розглянемо код, алгоритм його роботи та покращення які було впроваджено в нього. Код логіну та реєстрації можна знайти в додатку 2.

Найпершою проблемою коду реєстрації було використання шифрування MD5 для хешування паролів. MD5 є одним із найпоширеніших алгоритмів хешування, досі використовується в деяких CMS системах та на сайтах які було

створено на старих версіях цих систем, які являють собою близько 30% від усіх сайтів, але в сфері паролів за сучасними стандартами цей алгоритм вважається застарілим та може бути декодованим за допомогою тунелювання, або застосування веселкових таблиць (колекція загальних паролів і хешів, згенерованих наперед).

Отож в обновленому коді алгоритм MD5 був замінений на сучасний стандарт хешування BCrypt. BCrypt - це адаптивна криптографічна функція формування ключа, що використовується для безпечного зберігання паролів. Для захисту від атак за допомогою веселкових таблиць BCrypt використовує сіль (salt); крім того, функція є адаптивною, час її роботи легко налаштовується і її можна сповільнити, щоб ускладнити атаки перебором. Сіль в контексті шифрування паролів - це додаткові дані, які додаються до паролю перед його хешуванням. Головна мета солі - ускладнити атаки з використанням веселкових таблиць та інших методів атак на хеш-значення пароля.

Коли створюється акаунт або змінюється пароль на веб-сайті, система генерує випадковий рядок (сіль) і додає його до вашого паролю перед тим, як обчислити хеш. Таким чином, навіть якщо два користувачі мають однакові паролі, їхні хеш-значення будуть відрізнятися через використання унікальних солей. Додавання солі ускладнює завдання атакуючим, оскільки їм необхідно заздалегідь знати сіль для кожного пароля.

Також, щоб забезпечити більшу безпеку, було додано умову про мінімальну довжину паролю в 8 символів, а також додаткову перевірку правильності введеного паролю. Створено затримку між невдалими спробами вводу для ускладненого проведення атак з перебором та зменшення навантаження на сервер. Також додано перевірку співпадіння електронної пошти задля запобігання повторної реєстрації користувача. Окрім того повідомлення про помилки та успішну реєстрацію і вхід було винесено на сам сайт замість використання застарілих `alert` повідомлень. На рисунку 2.3 можна побачити оновлену сторінку реєстрації з блоками повідомлень про помилку та успішну реєстрацію.

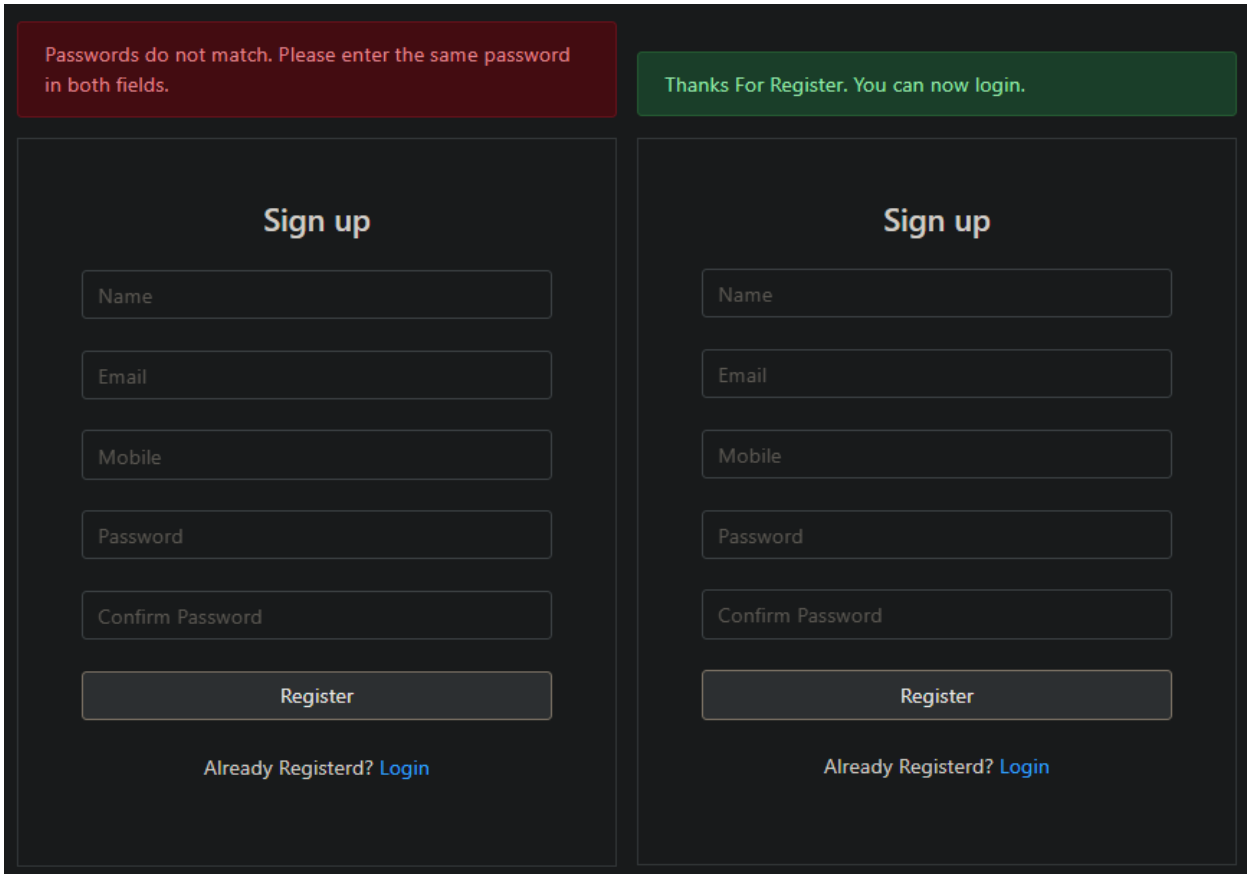

Рисунок 2.3 – Оновлені форми реєстрації з повідомленнями

Окремо можна також виділити покращення безпеки з боку SQL-ін'єкцій. SQL-ін'єкція (SQL injection) - це тип атаки на веб-додатки, яка використовується для вставки зловмисного SQL-коду в запити до бази даних через введення веб-форм або інші входи. Зазвичай SQL-ін'єкція виникає, коли введені користувачем дані не перевіряються на валідність або не екрануються на вході, і потім використовуються в SQL-запитах без належного оброблення. Атакуючий може вставити спеціально сформований SQL-код, який виконується базою даних, що призводить до небезпеки для конфіденційності та цілісності даних. Для запобігань цьому було використано параметризовані запити, що робить його більш захищеним від SQL-ін'єкцій порівняно з простою конкатенацією рядків. Усі зміни які було внесено в код можна побачити нижче у таблиці 2.1.

Таблиця 2.1 – Порівняння оригінального та покращеного алгоритму авторизації

Особливість	Оригінальний алгоритм	Покращений алгоритм
Хешування паролю	Використовує md5	Використовує password_hash і генерує сіль для покращеної безпеки
Перевірка довжини паролю	Немає перевірки	Використовує strlen для мінімальної довжини 8 символів
Сіль (Salt)	Не використовується	Генерує випадкову сіль за допомогою random_bytes та додає до паролю перед хешуванням
Перевірка існування електронної пошти	Не проводиться	Перевіряє, чи існує користувач з вказаною електронною поштою перед реєстрацією
Виведення повідомлень про стан реєстрації	Використовує echo та alert	Використовує змінні $errorMessage та $successMessage для збереження та виведення повідомлень
Іменування змінних	Короткі та загальні	Зрозумілі та специфічні $errorMessage та $successMessage для більшої зрозумілості
Захист від SQL-ін'єкцій	Не використовує підготовлені запити	Використовує підготовлені запити та додатково перевіряє наявність електронної пошти перед вставкою нового користувача
Захист від повторних спроб входу	Відсутній	Запобігає повторним спробам входу протягом 5 секунд, зберігаючи час останньої спроби вхіду в $_SESSION

Також було створено алгоритми роботи відповідні до покращеного коду. Перший алгоритм відповідає за реєстрацію користувачів на веб-сайті. Він

включає в себе перевірку коректності та унікальності введених даних, генерацію хешу паролю та збереження в базі даних. Деталі алгоритму:

1. Початок сесії і підключення конфігураційного файлу: Встановлює сесію та підключає файли конфігурації для роботи з базою даних.

2. Ініціалізація змінних: Створює змінні для зберігання повідомлень про помилки та успіху.

3. Перевірка натискання кнопки "Submit": Перевіряє, чи користувач натиснув кнопку "Submit" на формі реєстрації.

4. Отримання даних з форми: Збирає введені користувачем дані з форми.

5. Перевірка електронної пошти: Форма перевіряє чи в поле пошти було заповнено відповідно то формату.

6. Перевірка заповнення обов’язкових полів в формі.

7. Перевірка паролів: Перевіряє, чи паролі співпадають та чи пароль має достатню довжину.

8. Генерація випадкової солі та хешування паролю: Генерує випадкову сіль та створює хеш паролю для забезпечення безпеки.

9. Перевірка існування електронної пошти в базі даних: Перевіряє, чи введена електронна пошта вже використовувалась для реєстрації.

10. Вставка нового користувача в базу даних: Додає нового користувача до бази даних із захешованим паролем та випадковою сіллю.

11. Виведення повідомлення про успішну реєстрацію або про помилку: Виводить повідомлення про результат реєстрації.

На рисунку 2.4 представлено блок-схему алгоритму роботи реєстрації користувачів.

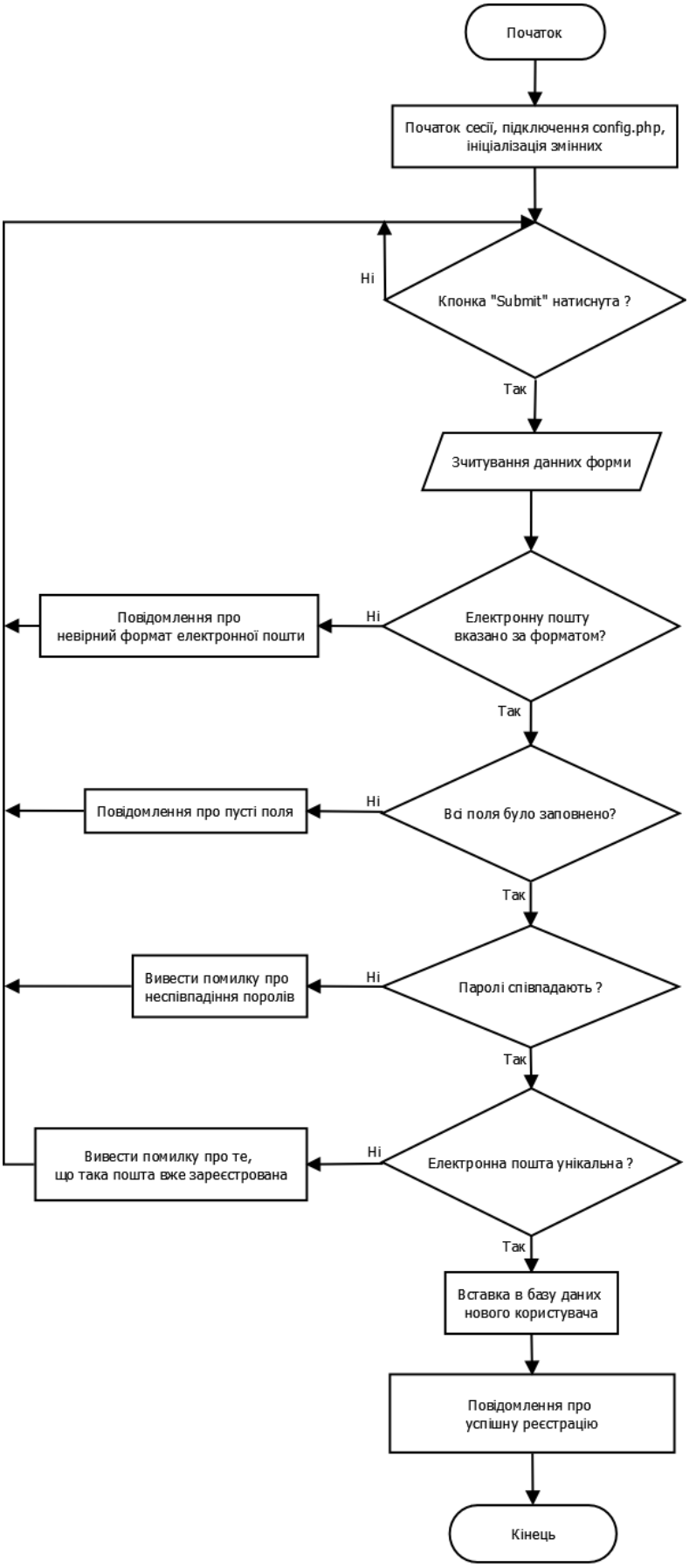

Рисунок 2.4 – Блок-схема оновленого алгоритму реєстрації користувачів

Наступний алгоритм описує роботи форми авторизації користувачів. Нижче подано опис його дій:

1. Початок сесії і підключення конфігураційного файлу: Встановлює сесію та підключає файли конфігурації для роботи з базою даних.

2. Ініціалізація змінних: Створює змінні для зберігання повідомлень про помилки та успіху.

3. Перевірка натискання кнопки "Submit": Перевіряє, чи користувач натиснув кнопку "Submit" на формі авторизації.

4. Отримання даних з форми: Збирає введені користувачем дані (електронна пошта та пароль) з форми.

5. Перевірка часового інтервалу між спробами входу: Перевіряє, чи не відбулася попередня спроба входу протягом останніх 5 секунд. Якщо так, виводить повідомлення про очікування.

6. Виконання авторизації:

- Виконує запит до бази даних, щоб отримати збережений хеш паролю та сіль для вказаної електронної пошти.
- Порівнює введений користувачем пароль з хешем паролю та сіллю з бази даних.
- У разі вдалого входу встановлює сесію для користувача та перенаправляє його на іншу сторінку.

7. Обробка помилок при авторизації: Виводить повідомлення про невірний пароль або електронну пошту, якщо така не знайдена в базі даних.

8. Запис часу спроби входу: Записує час останньої спроби входу в сесію для встановлення часового інтервалу між спробами.

На рисунку 2.5 представлено блок-схему алгоритму роботи ідентифікації користувачів.

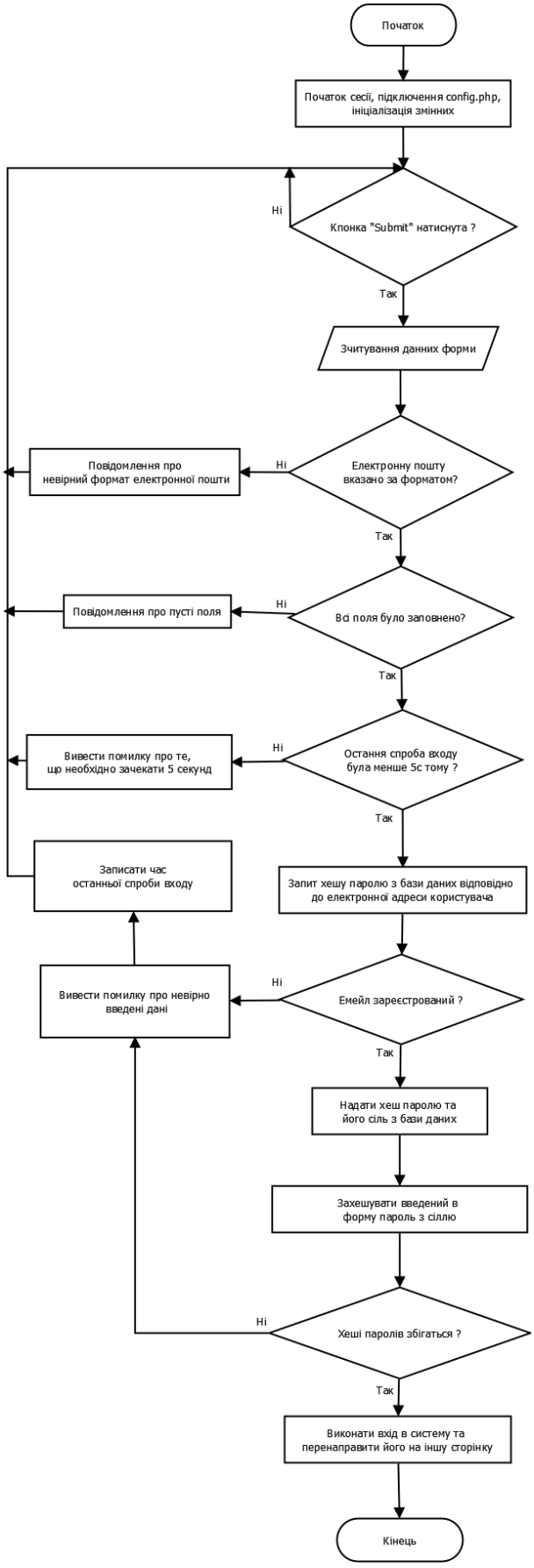

Рисунок 2.5 – Блок-схема оновленого алгоритму ідентифікації користувачів

2.3 Обчислення складності запропонованого алгоритму

Теорія алгоритмів - це галузь науки, що вивчає алгоритми, тобто точні, чітко визначені набори інструкцій, які виконують певні обчислення, розв'язують проблеми або виконують завдання. Ця галузь охоплює дослідження та аналіз різних типів алгоритмів, їхню ефективність, складність та можливість застосування в різних областях, таких як комп'ютерні науки, математика, інженерія та інші. Теорія алгоритмів також вивчає теоретичні межі обчислювальних можливостей, такі як складність задач та обмеження алгоритмів.

Складність алгоритму - це міра ресурсів, необхідних для виконання або виконання алгоритму. Ця концепція може включати в себе кількість операцій, час виконання, кількість пам'яті або інші ресурси, які потрібні для завершення алгоритму.

Складність може бути розглянута як складність у часі (часова складність) - як довго займає виконання алгоритму, або у пам'яті (просторова складність) - скільки пам'яті потрібно для виконання. Іноді також враховується складність за кількістю операцій, що виконуються. Розуміння часової та просторової складності має вирішальне значення для проектування та аналізу алгоритмів. У більшості випадків важливо створювати алгоритми, які будуть ефективні як у часі, так і у використанні пам'яті. Однак часто існує компроміс між часом та обсягом пам'яті, який варто враховувати при розробці алгоритмів.

Часова складність вказує на те, як швидко виконується алгоритм залежно від розміру введених даних. Наприклад, якщо у алгоритму часова складність O(n), це означає, що час його виконання збільшується лінійно з розміром введених даних. У випадку подвоєння розміру вводу, час виконання алгоритму також збільшиться удвічі.

Просторова складність вказує на обсяг пам'яті, який потрібний для виконання алгоритму в залежності від розміру вхідних даних. Наприклад, якщо

просторова складність алгоритму O(n), то обсяг пам'яті, необхідний для виконання алгоритму, зростатиме лінійно з розміром введених даних.

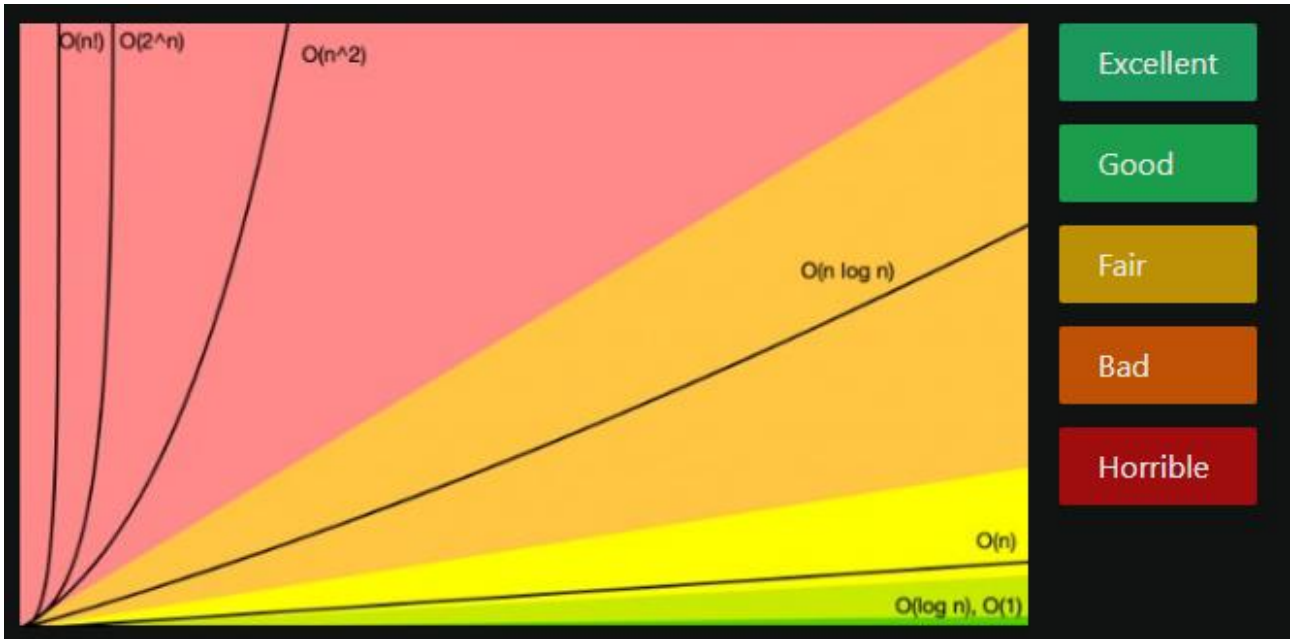

Рисунок 2.6 – Діаграма алгоритмів складності

На рисунку 2.6 можна побачити Діаграму алгоритмів складності та їх розподіл по оцінкам часу. В таблиці 2.2 можна побачити їхню назву та опис. Ця таблиця демонструє різні типи алгоритмів складності в залежності від розміру введених даних. Чим менше значення в складності (наприклад, O(log n) або O(n)), тим ефективніше зазвичай працює алгоритм, у порівнянні з алгоритмами, що мають більшу складність (наприклад, $O(n^2)$ або $O(2^n)$).

Таблиця 2.2 – Типи алгоритмічної складності

Складність	Назва	Опис
1	2	3
O(1)	Константна	Фіксований час виконання без залежності від розміру вводу

Продовження таблиці 2.2

1	2	3
$O(\log n)$	Логарифмічна	Час зростає логарифмічно в залежності від розміру вводу
$O(n)$	Лінійна	Час зростає лінійно від розміру введених даних
$O(n \log n)$	Лінійно-логарифмічна	Час зростає лінійно помножено на логарифм від розміру вводу
$O(n^2)$	Квадратична	Час зростає квадратично від розміру введених даних
$O(2^n)$	Експоненціальна	Час зростає експоненційно від розміру вводу
$O(n!)$	Факторіальна	Час зростає факторіально від розміру введених даних

Відповідно до алгоритмів розроблених в 2.2 було розраховано складність ідентифікації користувачів проекту. Спершу розглянемо складність алгоритму реєстрації користувача. Щоб розрахувати складність цього алгоритму, потрібно розглянути окремі кроки та оцінити їх вплив на часову складність.

Покроковий розрахунок складності алгоритму реєстрації користувачів:

1. Початок сесії і підключення конфігураційного файлу: Зазвичай це фіксований час O(1), якщо вважати, що це просто встановлення з'єднання з сайтом без необхідності для обробки конфігурацій.

2. Ініціалізація змінних: Це також фіксований час O(1), оскільки це просто створення змінних, що не залежать від об'єму даних чи вхідних параметрів.

3. Перевірка натискання кнопки "Submit": Це фіксований час O(1), оскільки це лише перевірка події натискання кнопки.

4. Отримання даних з форми: Часова складність може бути пропорційною кількості даних, введених користувачем, тому це може бути O(n),

де n - кількість даних. Проте в даному випадку кількість даних введених користувачем обмежені формою тому спрощуємо до O(1).

5. Перевірка електронної пошти: Це також може бути фіксований час O(1), оскільки це просто перевірка формату електронної пошти.

6. Перевірка заповнення обов'язкових полів в формі: Це також фіксований час O(1), оскільки це лише перевірка наявності обов'язкових полів.

7. Перевірка паролів: Це зазвичай може бути фіксованим часом O(1) або може залежати від довжини паролю, як O(n), де n - довжина паролю.

8. Генерація випадкової солі та хешування паролю: Зазвичай це також фіксований час O(1) для генерації солі та O(n) для хешування паролю.

9. Перевірка існування електронної пошти в базі даних: Це може мати часову складність, залежну від розміру бази даних, тому це може бути O(log n), де n - кількість записів у базі даних.

10. Вставка нового користувача в базу даних: Це зазвичай має фіксований час O(1) для вставки одного запису в базу даних.

11. Виведення повідомлення про успішну реєстрацію або про помилку: Це також фіксований час O(1), оскільки це просто виведення повідомлення.

Загальна часова складність цього алгоритму буде сумою часових складностей окремих кроків. Наприклад, якщо найбільша часова складність відбувається при перевірці наявності електронної пошти в базі даних O(n), загальна складність буде описана як найбільша з цих складностей серед усіх кроків.

Далі розглянемо складність алгоритму авторизації користувачів. Алгоритм авторизації користувачів має кілька кроків, які можна розглядати з точки зору складності за стандартами O(f(n)).

1. Початок сесії і підключення конфігураційного файлу - цей крок вважається операцією з фіксованим часом, тому його часова складність може бути зазначена як O(1).

2. Ініціалізація змінних - аналогічно, це також фіксований час, тому O(1).

3. Перевірка натискання кнопки "Submit" - також фіксований час O(1).

4. Отримання даних з форми - це операція, яка зазвичай займає час, пропорційний кількості даних, тому часова складність може бути описана як O(n), де n - кількість даних, введених користувачем. Проте в даному випадку кількість даних введених користувачем обмежені формою тому спрощуємо до O(1).

5. Перевірка часового інтервалу між спробами входу - це також фіксований час, тому O(1).

6. Виконання авторизації - цей крок включає доступ до бази даних та порівняння паролів. Зазвичай доступ до бази даних може бути описаний як O(1) або O(log n), де n - кількість записів у базі даних. Порівняння паролів також зазвичай має часову складність O(1) або O(n), де n - довжина паролю.

7. Обробка помилок при авторизації - це також фіксований час O(1).

8. Запис часу спроби входу - це ще одна операція з фіксованим часом, тому O(1).

Загальною складністю алгоритму авторизації користувачів може бути визначена як найбільша часова складність серед усіх кроків, які відбуваються у алгоритмі. У даному найбільша складність буде O(n) при хешуванні та порівнянні паролів, проте оскільки довжина паролю обмежена цю складність можна прирівняти до O(n). Тому коректніше буде використати O(log n). Формула O(log n) застосовується у випадку операцій, які мають логарифмічну часову складність у відношенні до кількості даних, з якими вони працюють. Однак, в контексті запитів до бази даних, складність може варіюватися в залежності від конкретної бази даних, її оптимізації та індексів, використовуваних у запитах.

В деяких сценаріях, при ефективно налаштованих індексах для шуканих полів (як правило, для унікальних значень, які часто використовуються у запитах), пошук по цим індексам може мати логарифмічну складність. Основна ідея полягає в тому, що з кожним подвоєнням кількості записів у таблиці, кількість кроків для пошуку конкретного запису збільшується лише на одиницю.

Проте реальна складність може бути різною залежно від деяких факторів:

1. Структура даних в БД: Використання індексів та їхній тип може впливати на швидкість пошуку.

2. Обсяг даних: Для невеликої кількості даних логарифмічна складність може бути незначною.

3. Оптимізація запитів: Сам запит може бути оптимізований для зменшення часу виконання.

Таким чином, визначення складності запитів до бази даних не завжди однозначно і може залежати від конкретного контексту, структури бази даних та специфіки запитів. Оскільки в даному проекті відбувається хешування користувацьких паролів, які відносно короткі та не забирають багато часу та ресурсів, то найдовшою ланкою складності буде використання бази даних. В цьому випадку використовується база даних MySQL з вбудованою індексацією та оптимізацією відповідно доцільно буде використовувати O(log n), де n – розмір таблиці до якої здійснюється запит.

2.4 Висновки до розділу 2

У другому розділі кваліфікаційної роботи проаналізовано стандартний алгоритм ідентифікації користувачів, що застосовується розробниками веб-сайтів. Запропоновано вдосконалений алгоритм ідентифікації користувачів, що дає можливість побудувати стійкий програмний модуль ідентифікації. Крім того, проведено розрахунок складності запропонованого алгоритму. Що підтвердило його працездатність та конкурентоспроможність.

3 МОДУЛЬ ІДЕНТИФІКАЦІЇ КОРИСТУВАЧІВ ВЕБ-САЙТУ

3.1 Середовище реалізації алгоритму ідентифікації користувачів

Реалізація алгоритму ідентифікації користувачів розроблялась на основі технологічного стеку з такими компонентами як:

- PHP – основна мова програмування на якій базується розроблений алгоритм
- HTML – мова розмітки для створення структури веб-сторінок
- CSS – мова опису стилів для надання вигляду веб-сторінок
- JavaScript – в даному випадку допоміжна мова програмування для реалізації вигляду веб-сторінок
- Git – розподілена система керування версіями
- MySQL (MariaDB) – система управління реляційними базами даних

А також програмними засобами які дозволяють реалізовувати ці технології:

- VS Code – редактор коду з зручними розширеннями
- GitHub - це веб-платформа системи керування версіями Git
- GitHub Desktop – графічний клієнт для роботи з системою контролю версій Git
- XAMPP – збірка веб-серверу для локальної розробки
- Apache – веб-сервер на якому виконується код PHP
- phpMyAdmin – як інтерфейс управління базою даних
- OpenSSL – бібліотека криптографічних функцій
- XAMPP Control Panel – панель керування веб-сервера
- Веб-браузери для тестування (Google Chrome, Microsoft Edge, Opera, Mozilla Firefox)

PHP (Hypertext Preprocessor) - це мова програмування загального призначення, яка використовується переважно для розробки веб-додатків та динамічних веб-сайтів. Вона вбудована безпосередньо в HTML і використовується для генерації вмісту веб-сторінок на сервері.

PHP здатна взаємодіяти з базами даних, керувати сесіями, генерувати веб-сторінки з динамічним контентом, обробляти форми та багато іншого. Вона є однією з найпоширеніших мов програмування для веб-розробки, оскільки дозволяє створювати функціональні та динамічні веб-сайти. Приклад динамічної коду динамічної зміни веб-сторінки можна побачити на рисунку 3.1. На ньому зображено код який виводить привітання в залежності від поточного часу.

Основними особливостями PHP є простота вивчення та використання, велика кількість готових функцій для роботи з базами даних, текстовими рядками, зображеннями, формами тощо. Вона також підтримує багато різних типів баз даних, включаючи MySQL, PostgreSQL, SQLite та інші.

```php
<?php
    $hour = date('G');

    if ($hour < 12) {
        $greeting = 'Доброго ранку!';
    } elseif ($hour < 18) {
        $greeting = 'Доброго дня!';
    } else {
        $greeting = 'Доброго вечора!';
    }
?>

<!DOCTYPE html>
<html>
<head>
    <title>Привітання залежно від часу доби</title>
</head>
<body>

    <h1><?php echo $greeting; ?></h1>

    <p>Зараз на годиннику: <?php echo date('H:i:s'); ?></p>

</body>
</html>
```

Рисунок 3.1 – Приклад коду PHP

PHP зазвичай виконується на веб-сервері, що підтримує PHP, такому як Apache, Nginx або Microsoft IIS, і генерує HTML-код, який потім відсилається браузеру користувача. Вона є важливою складовою для багатьох веб-розробок,

особливо там, де потрібно взаємодіяти з базами даних та створювати динамічні сторінки.

HTML, або "HyperText Markup Language" (мова розмітки гіпертексту), є стандартною мовою розмітки для створення та представлення веб-сторінок. Вона використовується для опису структури та зовнішнього вигляду вмісту веб-сторінки, такого як текст, зображення, відео, посилання і інше.

HTML складається з елементів, які представляють різні частини сторінки і надають їм певні властивості та значення. Елементи розмічаються за допомогою тегів, які оточують вміст і надають йому специфічне значення чи функціональність. Наприклад, тег <p> вказує на початок абзацу, тег <img> використовується для вставки зображення, а тег <a> - для створення посилань. На рисунку 3.2 можна побачити приклад коду з використанням цих тегів.

```
<!DOCTYPE html>
<html>
<head>
    <title>Заголовок сторінки</title>
</head>
<body>

    <h1>Привіт, світ!</h1>
    <p>Це приклад абзацу тексту.</p>

    <img src="зображення.jpg" alt="Опис зображення">

    <a href="https://www.example.com">Посилання на приклад</a>

</body>
</html>
```

Рисунок 3.2 – Приклад HTML-коду

HTML є однією з основних мов для створення веб-сайтів, і вона часто використовується разом з CSS та JavaScript для створення повноцінних та інтерактивних веб-додатків.

CSS, або "Cascading Style Sheets" (кускові таблиці стилів), є мовою стилізації, яка використовується для оформлення і форматування веб-сторінок,

написаних мовою HTML чи іншими мовами розмітки. CSS визначає зовнішній вигляд елементів веб-сторінки, таких як текст, колір, розміри, відступи, межі і інше.

Основні принципи CSS:

1. Вибір елементів: CSS визначає стилі для конкретних елементів чи груп елементів на веб-сторінці. Наприклад, визначення стилів для всіх заголовків <h1> або для конкретного ідентифікатора.

2. Властивості та значення: Кожен стиль має властивість (наприклад, колір чи розмір шрифту) і значення, яке вказується для цієї властивості. Наприклад, color: red; встановлює колір тексту на червоний.

3. Каскадність: "Cascading" вказує на те, що стилі можуть бути успадковані або перезаписані в залежності від їхнього місця в дереві елементів та їхнього пріоритету. Зазвичай, стилі, оголошені більше вище в коді, мають більший пріоритет.

4. Класи та ідентифікатори: Класи та ідентифікатори використовуються для вибору конкретних груп елементів для застосування стилів. Класи вказуються за допомогою крапки (наприклад, .клас), а ідентифікатори - за допомогою хешу (наприклад, #ідентифікатор).

На рисунку 3.3 можна побачити приклад простого CSS коду, який встановлює стилі для заголовків та абзаців.

```css
h1 {
    color: blue;
    font-size: 24px;
}

p {
    color: green;
    font-size: 16px;
    margin-bottom: 20px;
}
```

Рисунок 3.3 – приклад CSS коду

JavaScript - це мова програмування, яка використовується для створення динамічного веб-змісту і взаємодії користувача з веб-сторінками. Вона може контролювати поведінку сторінок, маніпулювати елементами HTML та CSS, обробляти події, виконувати запити на сервер для отримання чи збереження даних, а також створювати різноманітні ефекти та анімації.

На рисунку 3.4 можна побачити приклад простого JavaScript коду, який виводить вітання на основі імені, введеного користувачем у поле введення.

```
<!DOCTYPE html>
<html>
<head>
    <title>Привітання JavaScript</title>
</head>
<body>

    <input type="text" id="nameInput" placeholder="Введіть ваше ім'я">
    <button onclick="greet()">Привітати</button>
    <p id="greeting"></p>

    <script>
        function greet() {
            var name = document.getElementById('nameInput').value;
            var greeting = document.getElementById('greeting');
            greeting.innerHTML = 'Привіт, ' + name + '!';
        }
    </script>

</body>
</html>
```

Рисунок 3.4 – Приклад JavaScript коду

Основні можливості JavaScript:

- Маніпулювання DOM: JavaScript може динамічно змінювати структуру та вміст веб-сторінки, додавати, видаляти та змінювати елементи HTML, CSS та їх атрибути.
- Обробка подій: Вона дозволяє реагувати на події, такі як клік миші, натискання клавіш, завантаження сторінки тощо, і виконувати певні дії відповідно до цих подій.

- Взаємодія з користувачем: JavaScript дозволяє створювати веб-сайти, які відрізняються від статичних, оскільки вони можуть реагувати на введення користувача, валідувати форми, анімувати елементи тощо.
- Асинхронність: JavaScript може виконувати операції асинхронно, що дозволяє здійснювати запити на сервер і отримувати відповіді без блокування виконання інших операцій.
- Робота з файлами кукі та локальним сховищем: Вона дозволяє зберігати дані на боці клієнта, наприклад, у локальному сховищі або у кукі.

Git - це розподілена система керування версіями, що використовується для відстеження змін у програмному коді під час розробки програмного забезпечення. Вона дозволяє розробникам спільно працювати над проектами, відслідковувати зміни, керувати версіями коду, об’єднувати зміни в коді, а також відновлювати попередні стани проекту.

Git зберігає повну історію змін коду, що дозволяє відновлювати будь-яку попередню версію проекту. Кожна зміна коду може бути коментована, що дозволяє зрозуміти, які саме зміни були внесені та чому. Крім того, Git підтримує гілкування, що дозволяє розробникам працювати паралельно над різними функціями чи частинами проекту, не впливаючи одне на одного, та потім злити ці гілки разом.

Git є надзвичайно популярною системою керування версіями у сфері програмування через свою потужність, швидкість та гнучкість. Вона використовується для розробки програмного забезпечення будь-якого розміру - від індивідуальних проектів до великих корпоративних програмних продуктів.

MySQL - це відкрита реляційна система управління базами даних (СУБД), яка використовує мову запитів SQL (Structured Query Language). Вона надає можливості зберігання, організації та управління даними у вигляді таблиць, що мають відносини між собою.

MySQL є однією з найпопулярніших СУБД у світі, використовується для багатьох типів додатків, починаючи від веб-сайтів до великих корпоративних систем. Вона підтримує багато різних типів даних, дозволяє виконувати різні

операції з даними (додавання, видалення, оновлення, вибірка тощо), а також забезпечує можливість використання різних типів індексів для оптимізації швидкості пошуку даних.

MariaDB - це реляційна база даних, яка є гілкою розвитку від відкритої системи управління базами даних MySQL після її придбання компанією Oracle. MariaDB розробляється спільнотою розробників, у тому числі тими, які спершу працювали над MySQL, і вона спрямована на забезпечення відкритого, швидкого та надійного рішення для зберігання та управління даними.

MariaDB відкрита для використання, має багатий набір функцій, які включають у себе підтримку сховищ даних, реплікацію, кластеризацію, транзакції ACID (Atomicity, Consistency, Isolation, Durability) та інші. Вона прагне бути сумісною з MySQL, тому в більшості випадків код, спроектований для MySQL, також може працювати в MariaDB без значних змін.

Visual Studio Code (VS Code) - це безкоштовний та легкий текстовий редактор, розроблений компанією Microsoft. Він призначений розробників програмного забезпечення та надає велику вибірку інструментів для написання коду, відлагодження, редагування та управління проектами.

VS Code підтримує багато мов програмування, включаючи JavaScript, Python, HTML, CSS, C++, Java та інші. Він має широкий вибір розширень, які можна встановлювати для збільшення його можливостей, таких як підтримка нових мов програмування, інтеграція з системами керування версіями, (наприклад Git), автоматична підстановка коду, налагодження тощо.

VS Code славиться своєю швидкістю, гнучкістю та дружнім для користувача інтерфейсом, на рисунку 3.5 можна побачити інтерфейс застосунку. Його можна використовувати для будь-яких типів проектів, від невеликих скриптів до великих програмних розробок, що робить його популярним серед розробників різних спеціалізацій.

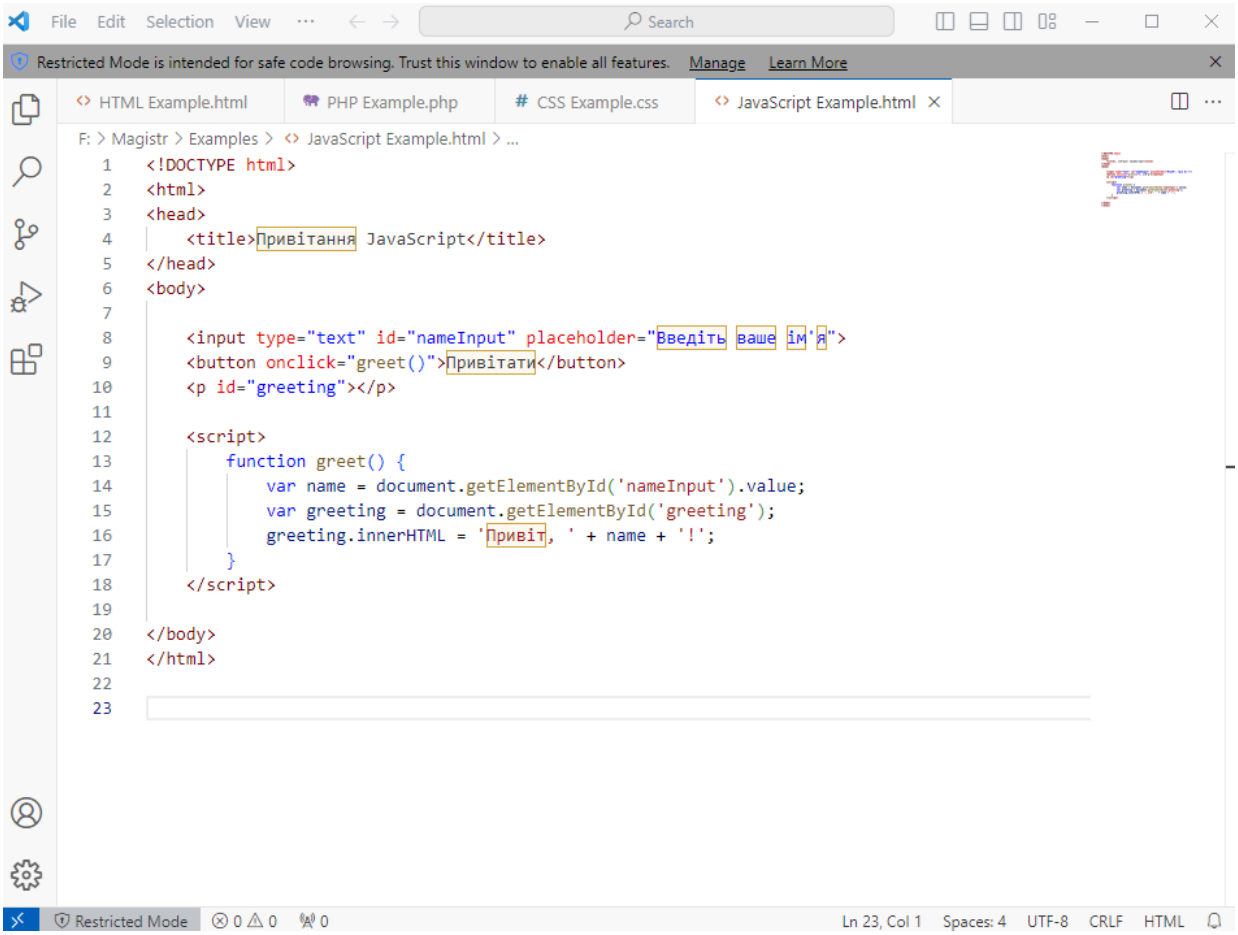

Рисунок 3.5 – Вигляд інтерфейсу VS Code

GitHub - це веб-платформа для хостингу проектів, що використовують систему контролю версій Git. Вона надає можливість розробникам спільно працювати над програмним забезпеченням, відстежувати зміни, керувати версіями коду та співпрацювати в рамках команд для розробки проектів.

GitHub став однією з найпопулярніших платформ для розробки програмного забезпечення, завдяки своїм зручним інтерфейсом, інструментам співпраці та широким можливостям для роботи з відкритим кодом або комерційними проектами.

GitHub Desktop - це графічний клієнт для роботи з системою контролю версій Git та хостингом коду GitHub. Це простий у використанні інструмент, який надає зручний інтерфейс для взаємодії з репозиторіями Git, дозволяючи

розробникам легко керувати своїми проектами, використовуючи Git та інтегруючись з сервісом GitHub.

GitHub Desktop надає можливість клонувати репозиторії з GitHub або будь-якого іншого хостингу, створювати нові гілки, вносити зміни в код, комітувати їх (створення нового стану змін), вирішувати конфлікти злиття, використовувати функції історії змін, а також виконувати злиття змін між гілками.

Цей інструмент особливо корисний для новачків, які тільки починають працювати з Git та GitHub, оскільки він має зрозумілий інтерфейс та спрощує багато операцій, пов'язаних з роботою з версіями коду. В той же час, він також може бути корисним для досвідчених розробників, які шукають зручний інтерфейс для взаємодії з Git та GitHub на рівні робочого столу. Вигляд інтерфейсу GitHub Desktop можна побачити на рисунку 3.6.

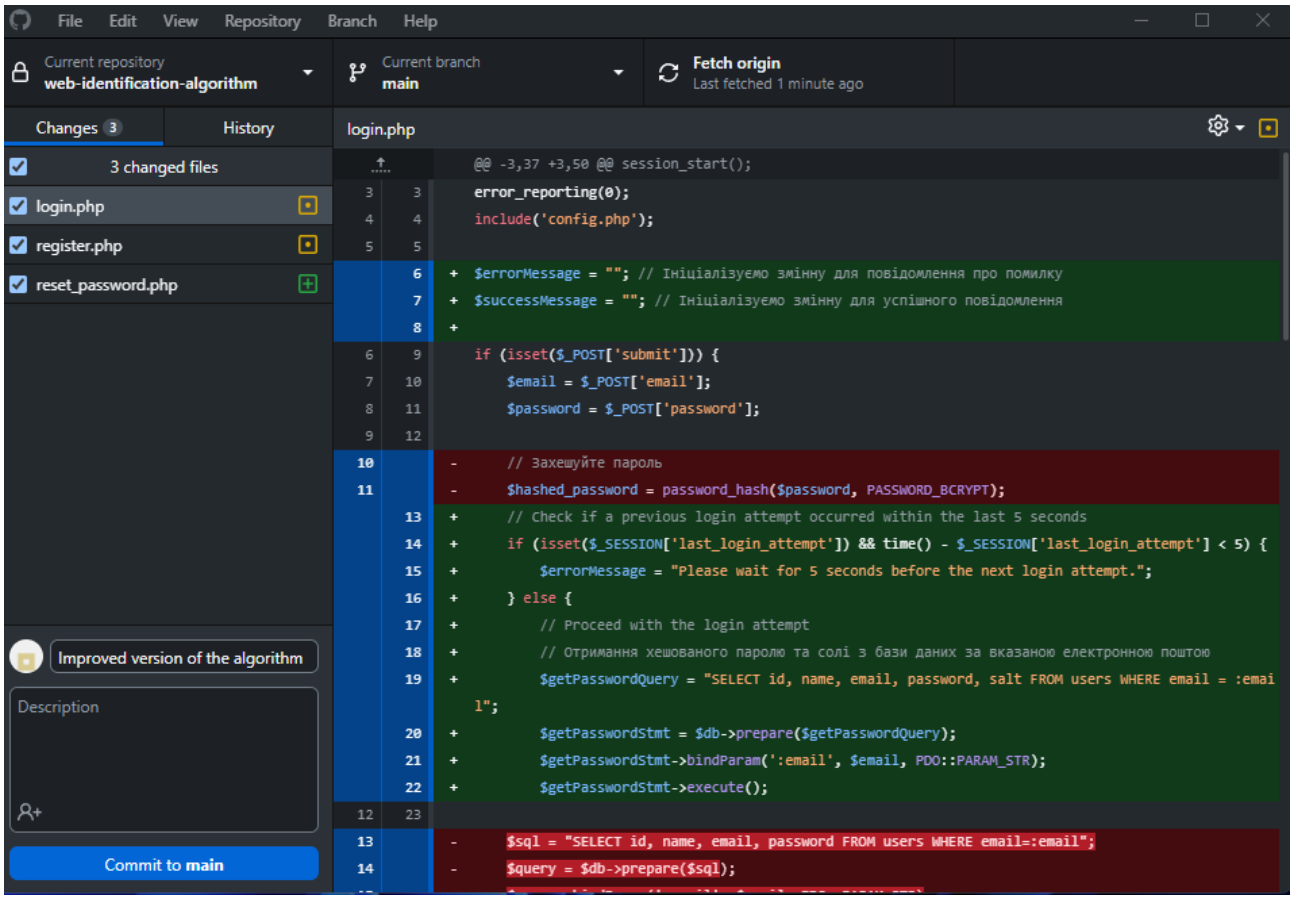

Рисунок 3.6 – Вікно інтерфейсу GitHub Desktop

XAMPP - це безкоштовний набір програм, який включає в себе сервер Apache, базу даних MySQL, мову програмування PHP та інші важливі компоненти. Назва XAMPP походить від перших літер складових частин: "X" - означає будь-яку операційну систему (Windows, Linux, Mac OS X), "A" - Apache, "M" - MySQL, "P" - PHP та "P" - Perl.

XAMPP призначений для локальної розробки та тестування веб-сайтів або веб-додатків перед їх публікацією в Інтернеті. Він дозволяє швидко налаштувати локальне середовище для розробки, що включає веб-сервер Apache для обробки веб-запитів, базу даних MySQL для зберігання і керування даними та мови програмування PHP для розробки динамічних веб-сторінок.

Однією з ключових переваг XAMPP є його простота встановлення та використання. Він дозволяє швидко отримати функціонуюче середовище для розробки веб-додатків без необхідності окремо встановлювати та налаштовувати кожен компонент (Apache, MySQL, PHP) окремо. Це робить його популярним в середовищі розробників, які хочуть швидко розпочати роботу над проектами.

XAMPP Control Panel - це інструмент, який дозволяє керувати серверним середовищем XAMPP, запускати та зупиняти певні його елементи, та надає швидкий доступ до папок та адміністративних функцій програм. Вигляд вікна XAMPP Control Panel зображено на рисунку 3.7.

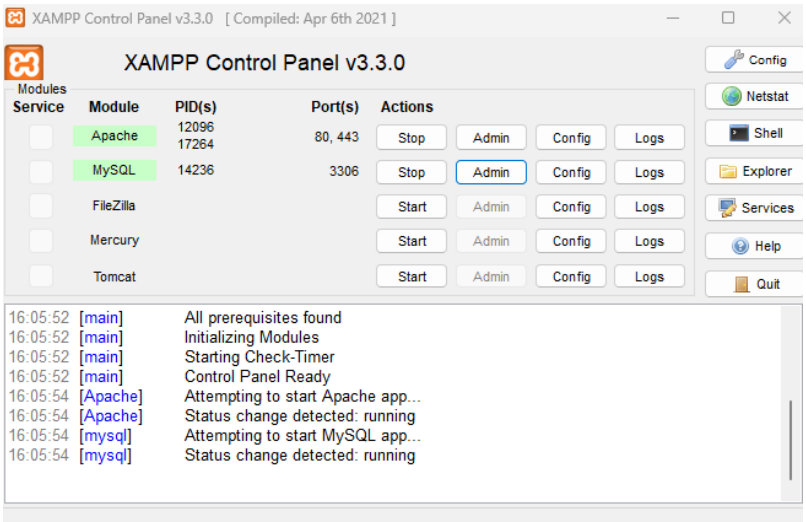

Рисунок 3.7 – Вікно XAMPP Control Panel

Apache - це один з найпопулярніших веб-серверів, який використовується для доставки веб-сторінок і різних веб-ресурсів через протокол HTTP і HTTPS через Інтернет. Apache HTTP Server, часто відомий просто як Apache, є вільним програмним забезпеченням, що розповсюджується під ліцензією Apache.

Apache надає потужний та надійний інструмент для обробки запитів веб-клієнтів, таких як браузери, і доставки вмісту на основі цих запитів. Він підтримує багато функцій, включаючи веб-серверні скрипти, віртуальні хости, SSL / TLS шифрування, автентифікацію та авторизацію користувачів, стиснення вмісту, ведення журналів та багато іншого.

Apache є дуже гнучким і може бути налаштований для великої кількості сценаріїв використання, від невеликих особистих веб-сайтів до великих корпоративних веб-порталів. Він також часто використовується разом з іншими технологіями, такими як PHP, Python, а також з системами управління контентом, наприклад, WordPress або Drupal, для забезпечення веб-сайтів та додатків.

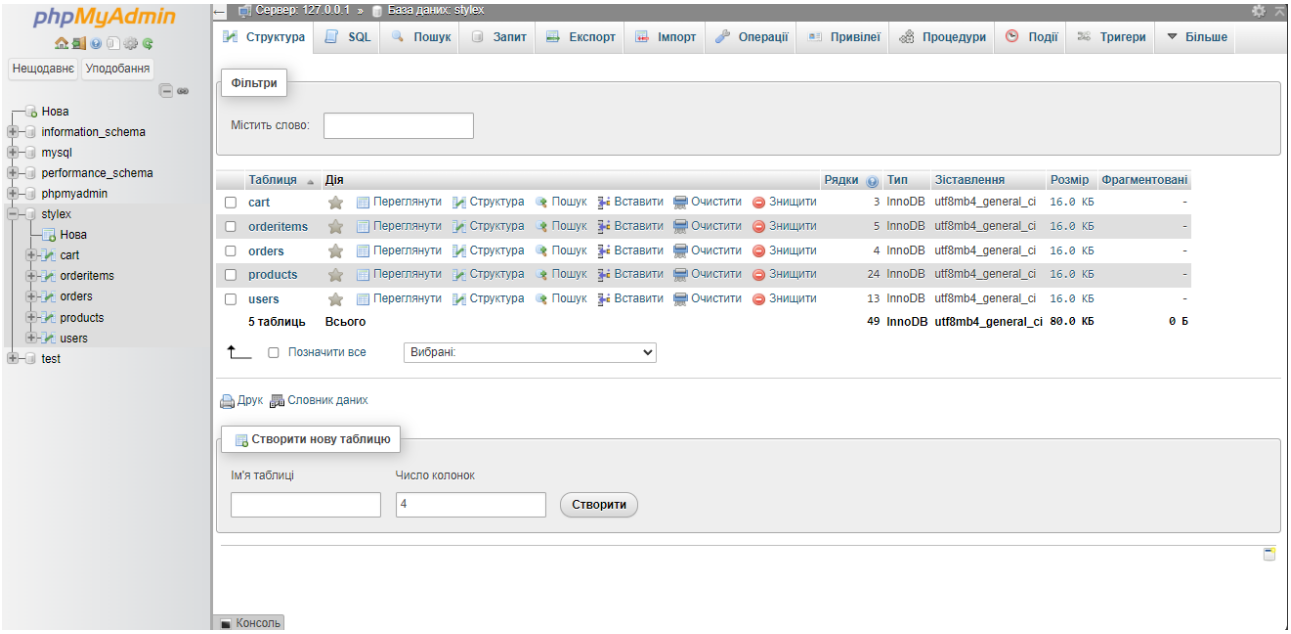

Рисунок 3.8 – Графічний інтерфейс phpMyAdmin

phpMyAdmin - це безкоштовний інструмент у веб-середовищі, який надає графічний інтерфейс для управління базами даних MySQL. Цей інструмент

дозволяє користувачам створювати, видаляти, редагувати бази даних, таблиці, запити SQL, керувати користувачами та їх привілеями і багато іншого. Він особливо корисний для тих, хто не хоче використовувати командний рядок для взаємодії з базами даних, а замість цього бажає використовувати інтуїтивний веб-інтерфейс. Графічний інтерфейс бази даних проекту можна побачити на рисунку 3.8.

OpenSSL - це бібліотека криптографічних функцій, яка забезпечує реалізацію широкого спектру протоколів безпеки, включаючи SSL (Secure Sockets Layer) і TLS (Transport Layer Security). OpenSSL надає інструменти для шифрування та розшифрування даних, підпису та перевірки цифрових підписів, генерації випадкових чисел, керування сертифікатами, аутентифікації і багато іншого.

Бібліотека OpenSSL широко використовується для захисту комунікацій через мережі Інтернет, забезпечення безпеки веб-сайтів (за допомогою HTTPS), розробки криптографічних програм і додатків з відкритим кодом. OpenSSL є одним із найпопулярніших інструментів для роботи з криптографією у веб-розробці та інших областях програмування.

Веб-браузер - це програма, яка призначена для перегляду веб-сторінок в Інтернеті. Вона дозволяє користувачам переходити з однієї веб-сторінки на іншу за допомогою гіперпосилань, переглядати тексти, зображення, відео, аудіо та інші типи контенту, що розміщені в Інтернеті. Браузери також підтримують роботу з різними веб-технологіями, такими як HTML, CSS, JavaScript, що дозволяє відображати та взаємодіяти з різноманітними видами веб-змісту. Наприклад, Google Chrome, Mozilla Firefox, Opera та Microsoft Edge - це деякі з найпопулярніших веб-браузерів.

Порівнюючи Google Chrome, Microsoft Edge, Opera та Mozilla Firefox можна побачити, що вони мають багато спільного, але також і кілька різниць в функціях та екосистемі. Порівняння цих браузерів наведено в таблиці 3.1.

Таблиця 3.1 – Порівняння веб-браузерів

Функція	Google Chrome	Microsoft Edge	Opera	Mozilla Firefox
Швидкодія	Висока	Висока	Висока	Висока
Розширення	Широкий вибір	Обмежений	Обмежений, є унікальні	Широкий вибір
Приватність	Залежить від налаштувань, є проблеми	Захист від слідкування	Вбудований VPN, блокування реклам	Фокус на приватності
Інтеграція з сервісами	Google ecosystem	Microsoft ecosystem	Власна, обмежена	Власна, не прив'язаний до конкретної екосистеми
Унікальні функції	Немає специфічних	Інтеграція з Windows, Cortana	VPN, блокування реклам	Фокус на приватності, відкритий джерела
Кросплатформеність	Підтримується	Підтримується	Підтримується	Підтримується

3.2 Симуляція та верифікація проекту

Проект алгоритму ідентифікації користувачів веб-сайту розроблявся на основі збірки веб-серверу для локальної розробки XAMPP. Специфікацію збірки з версіями компонентів можна побачити в таблиці 3.2.

Таблиця 3.2 – Специфікація збірки XAMPP

Назва	Версія
XAMPP	8.2.12
MariaDB	10.4.32
Apache	2.4.58
PHP	8.2.12
phpMyAdmin	5.2.1
OpenSSL	3.1.3
XAMPP Control Panel	3.2.4

Тестування форм логіну та реєстрації на веб-сайтах включає в себе різні аспекти, від перевірки функціональності до безпеки та зручності використання для користувачів. Тестування можна розділити на такі кроки:

1. Перевірка функціональності:

- Потрібно переконатись в тому що форми логіну та реєстрації працюють правильно. Виконати спробу зареєструватися та увійти, використовуючи різні типи даних (довгі паролі, спеціальні символи тощо).
- Провести перевірку чи правильно обробляються помилки (наприклад, невірний пароль або електронна пошта).

2. Безпека:

- Перевірити наявність захисту від атак, таких як SQL-ін'єкції або перехоплення даних.
- Перевірити, чи застосовані належні методи шифрування паролів (наприклад, хешування з сіллю).

3. Перевірка валідації даних:

- Потрібно впевнитись в тому що форми вимагають правильний формат електронної пошти, паролю тощо.
- Перевірити, чи належним чином обробляються некоректні дані, щоб уникнути потенційних проблем.

4. Тестування на різних пристроях та браузерах:

- Переконатись в тому що форми працюють на різних веб-браузерах та пристроях.

5. Тестування забезпечення:

- Переконатись в тому що не виникає конфліктів з іншими скриптами чи плагінами на сторінці.
- Проведіть тестування навантаження, щоб переконатися, що сервер може обробляти багато запитів на реєстрацію чи вхід одночасно.

6. Документація:

- Створити чітку документацію з зазначенням всіх перевірених аспектів і знайдених проблем.

Для перевірки функціональності було зареєстровано кілька користувачів з різними логінами та паролями, та виконано кілька спроб входу з вірними та хибними даними. Результати невдалої спроби можна побачити на рисунку 3.9. При успішній спробі проходить перенаправлення на головну сторінку, а при невірно введених даних виводиться відповідне повідомлення.

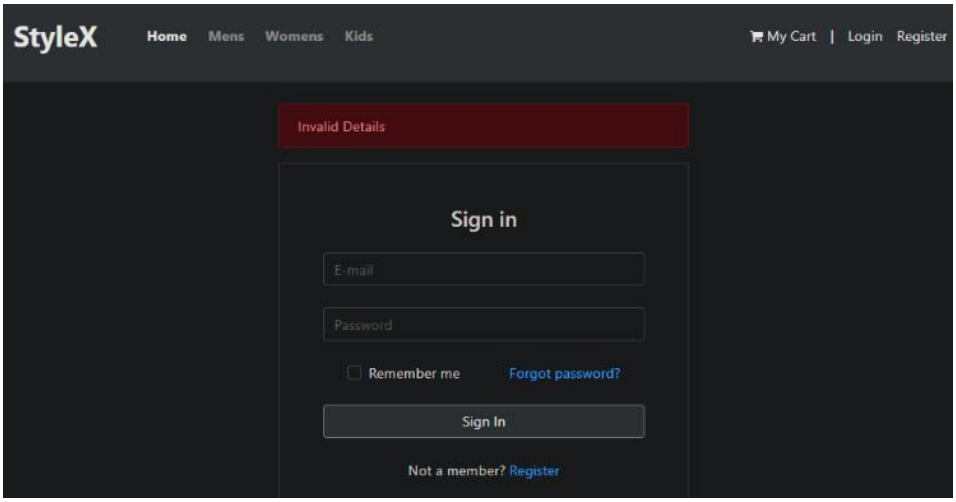

Рисунок 3.9 – Результати тестування входу

Далі було проведено тестування безпеки. Для перевірки захисту від SQL-ін'єкції було зареєстровано користувачів з іменами `DROP TABLE`, SELECT * FROM `users` та тому подібні, ніякі з команд не виконувались завдяки використанню параметризованих запитів. Щоб впевнитись в коректному хешуванні паролів з сіллю було створено два користувачі з ідентичними паролями та зрівняно їх хеші з бази даних. Для кожного з користувачів була згенерована своя сіль тому хеші паролів не співпадають. Рядки з бази даних можна побачити на рисунку 3.10.

id	name	email	mobile	password	created_at	salt
25	HashTest1	HashTest1@mail.com	380661234567	$2y$10$zvb4Da3g/dhi65UD2a0fQ.uRwX57U/b0WSDrcRyrW/S...	2023-11-29 15:19:52	4f9bdd8ce5a9c76c
26	HashTest2	HashTest2@mail.com	380991234567	$2y$10$mLDeskW3xRLsT4CF84AMS.U3Br5.Lg2lbaKjp2bM28v...	2023-11-29 15:20:25	bd0653525420ddbe

Рисунок 3.10 – Рядки користувачів з ідентичними паролями

Тестування валідації даних які вводяться в форму було проведено методом спроби введення невірних даних, проте оскільки HTML теги форм мають відповідні функції тестування пройшло успішно. Також було перевірено форму підтвердження пароля.

Також було проведено тестування кросбраузерності для забезпечення стабільного результату незалежно від платформи. Браузери та їх версії можна знайти в таблиці 3.3. На всіх браузерах реєстрація та авторизація пройшли успішно. Вигляд вінка реєстрації на різних браузерах можна побачити на рисунку 3.11.

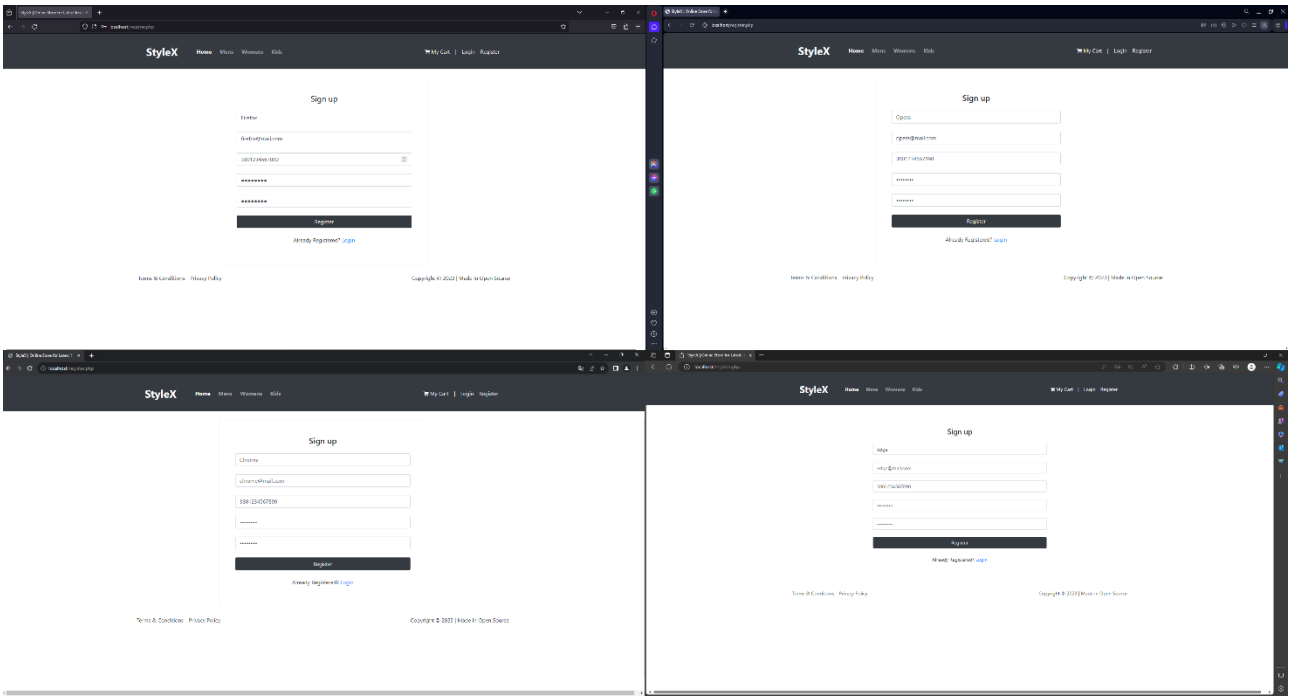

Рисунок 3.11 – Тестування кросбраузерності

Таблиця 3.3 – Версії браузерів тестування

Браузер	Версія
Google Chrome	118.0.5993.159
Opera	105.9.4970.21
Microsoft Edge	119.0.2151.93
Mozilla Firefox	102.10

При тестуванні на локальному сервері немає можливості створити велике навантаження на веб-сайт, тому тестування навантаження не було проведено.

Загалом під час тестування критичних проблем виявлено не було, проте при реальній роботі веб-сайту можуть виникнути проблеми наприклад: атака ботами, атака перехоплення сесій, “брутфорс” атака на сторінку логіну або Denial-of-Service (DoS) або Distributed Denial-of-Service (DDoS) атаки. Навіть якщо ці атаки не здобудуть успіху у отриманні конфіденційної інформації, веб-сайт може некоректно працювати, що в свою чергу, буде перешкоджати використанню функціоналу для реальних клієнтів.

Частина можливих проблем може бути вирішена залежно від вибору веб-хостингу проте деякі з них можна вирішити на рівні сайту. В проекті існує можливість покращення, і як у будь-якого веб-сайту, комплекси безпеки повинні регулярно оновлюватись та відповідати стандартам сучасного часу.

3.3 Результати аналізу роботи алгоритму ідентифікації користувачів

Для демонстрації роботи алгоритму було запущено локальний сервер на базі XAMPP та проведено контрольну реєстрацію та авторизацію. Далі поетапно зображено роботу веб-сайту ідентифікації користувачів з відповідними рисунками.

При переході за посиланням локального серверу ‘localhost’ відкривається головна сторінка веб-сайту, яку можна побачити на рисунку 3.12. На ній видно, що основний функціонал сайту та його сторінки, в верхній частині сайту ‘хедері’ та в правій його частині кнопки корзини, авторизації та реєстрації.

Рисунок 3.12 – Головна сторінка веб-сайту

Далі було відкрито сторінку реєстрації натиснувши на відповідну кнопку справа зверху. Була виконана спроба ввести невірний формат даних та різні паролі. Після чого було створено нового користувача. Результат при успішному створенні користувача можна побачити на рисунку 3.13.

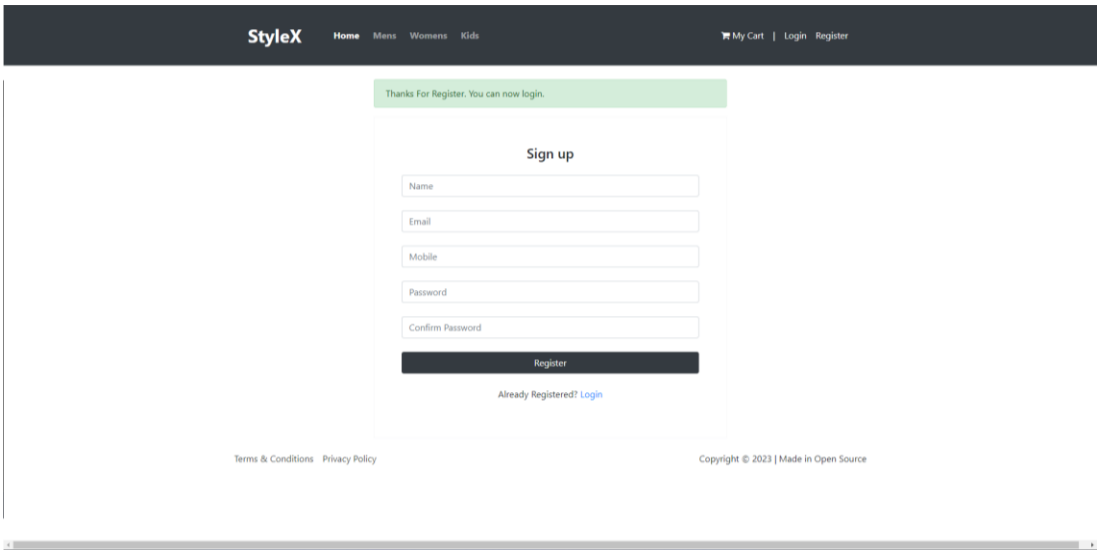

Рисунок 3.13 – Успішна реєстрація користувача на сайті

В свою чергу невдалі спроби реєстрації та повідомлення які при них виникали можна побачити на рисунку 3.14.

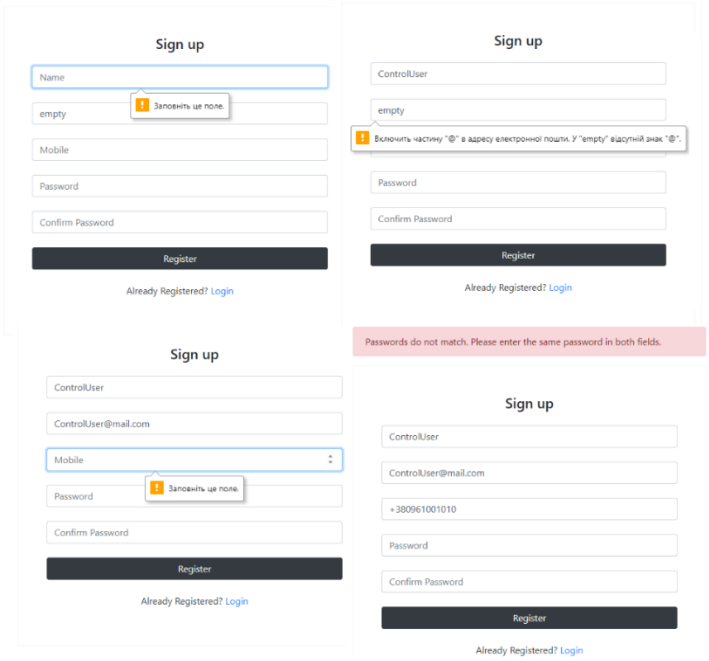

Рисунок 3.14 – Невдалі спроби реєстрації з помилками

Для перевірки коректності запису даних було відкрито та зрівняно введені дані з форми з рядком в таблиці бази даних. Та виконано пошук по таблиці “users” за іменем користувача, результат можна побачити на рисунку 3.15.

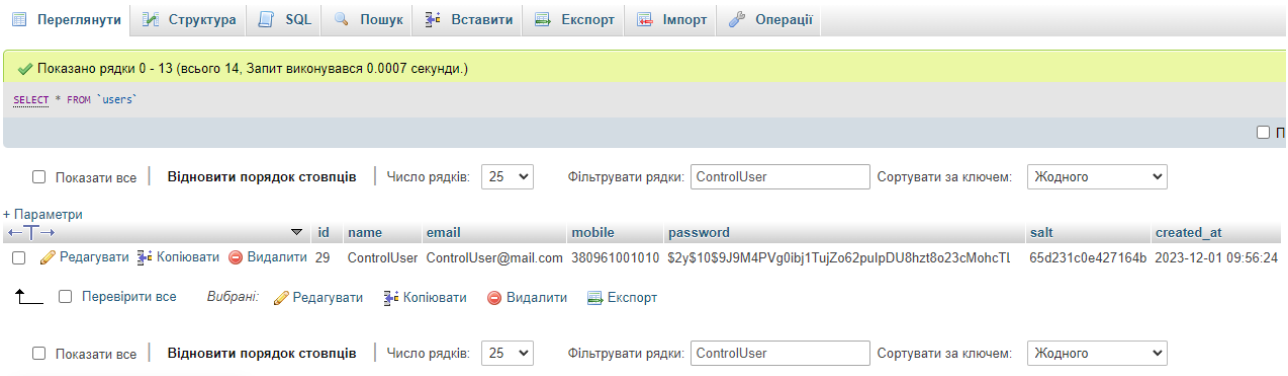

Рисунок 3.15 – Зареєстрований користувач в базі даних

Далі була виконана спроба входу в акаунт через сторінку входу з невірними даними, результат такої спроби можна побачити на рисунку 3.16.

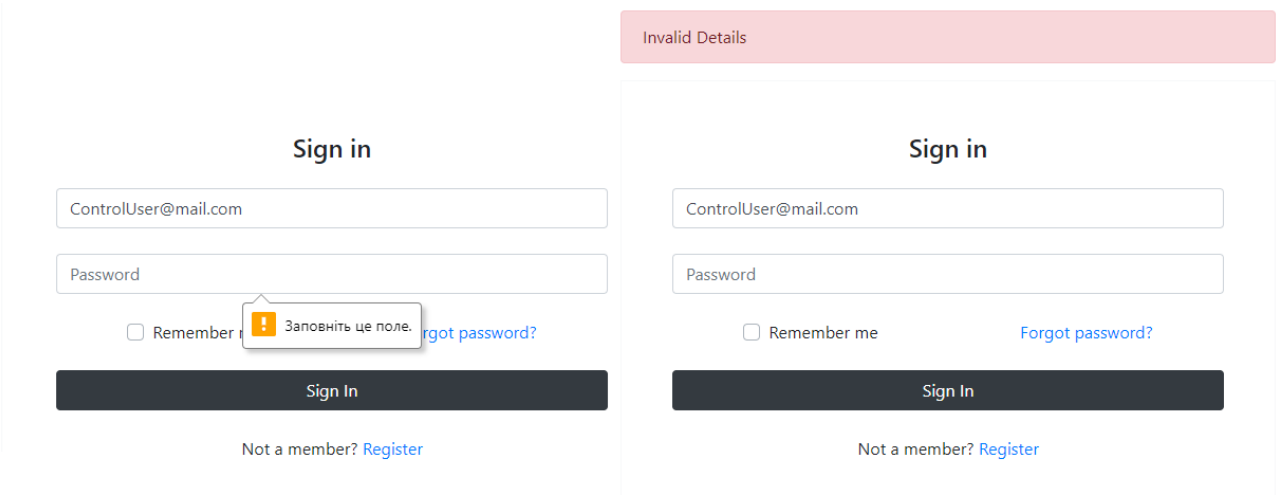

Рисунок 3.16 – Невдалі спроби входу

Після чого було введено правильні дані та успішно переадресовано на головну сторінку, результат успішної спроби можна побачити на рисунку 3.17. В правому верхньому куті можна побачити назву користувача.

Рисунок 3.17 – Успішна авторизація

Отже в результаті контрольного тестування можна побачити що алгоритм працює як задумано, не дає вводити невірні дані та виконує успішні реєстрації та авторизації користувачів на веб-сайті.

В ході розробки проекту було проаналізовано сучасні алгоритми ідентифікації користувачів на веб-сайтах, вибрано найпопулярніший та найбільш вагомий в з них, в вигляді реєстрації та авторизації користувачів. Було знайдено найбільш розповсюджений варіант стандартного вигляду реєстрації та авторизації. Проаналізовано можливі вразливості з боку витоку конфіденційних даних користувачів, та розроблено покращений варіант алгоритму. Здійснено його реалізацію в вигляді коду. В ході тестування підсумки показали, що алгоритму ідентифікації користувачів на веб-сайті, в загальному, працює як задумано та забезпечує підвищених захист конфіденційної інформації користувачів. Однак деякі вразливості, які не залежать від коду алгоритму, можуть з’являтись на інших його рівнях, таких як надійність серверу хостингу чи неуважність користувача.

3.4 Висновки до розділу 3

У третьому розділі здійснено моделювання та верифікацію запропонованого алгоритму ідентифікації користувачів. Здійснено симуляцію роботи змодельованого модуля та досліджено результати його роботи, що підтвердило його працездатність та правильність роботи.

ВИСНОВКИ

У даній кваліфікаційній випускній роботі здійснено наступні етапи:

1. Здійснено аналіз способів ідентифікацій користувачів на веб-сайтах. Проведено порівняння можливих способів ідентифікації та вибрано найпоширеніший з них.

2. Досліджено можливі вразливості реєстрації та авторизації користувачів, способи атак на сайти, що дало можливість підібрати варіанти захисту при подальшій розробці алгоритму.

3. Проаналізовано найпоширеніший варіант алгоритму реєстрації та авторизації користувачів на веб-сайтах, та його реалізацію. На основі цього алгоритму розроблено покращену його версію з урахуванням можливостей різноманітних вразливостей безпеки.

4. Проведено реалізацію цього алгоритму в вигляді веб-сайту та механізму реєстрації та авторизації користувачів. В ході реалізації алгоритм працював як задумано та виконував свої функції.

5. Проведено тестування реалізації алгоритму його функцій та безпеки. В ході тестування було доведено забезпечення покращеної безпеки. Згодом було зроблено висновки про потенційне подальше покращення алгоритму та залишок можливих небезпек на рівні серверу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Адітья Бхаргава. Грокаємо алгоритми. Ілюстрований посібник для програмістів і допитливих. ArtHuss, - 2023. – 256с.
2. The history of ID. [Електронний ресурс]. – Режим доступу: https://www.veriff.com/blog/the-history-of-id
3. Ivancsy, Renata, and Sandor Juhasz. "Analysis of web user identification methods." International Journal of Computer and Information Engineering 1.10. – 2007. - 3049-3056 p.
4. Юрій Когут, Кібербезпека та ризики цифрової трансформації компаній. Сідкон – 2021. – 372с.
5. 100,000 years of identity verification: an infographic history [Електронний ресурс]. – Режим доступу: https://www.trulioo.com/blog/identity-verification/infographic-the-history-of-id-verification
6. Пасічник Наталія Романівна. Математичні моделі відвідуваності веб-сайтів та методи їх ідентифікації. Тернопіль – 2014. – 178с.
7. Богуш В.М., Богуш В.В., Бровко В.Д., Настрадін В.П. Основи кіберпростору, кібербезпеки та кіберзахисту. Ліра-К – 2021. – 554с.
8. Popular Authentication Methods for Web Apps [Електронний ресурс]. – Режим доступу: https://www.baeldung.com/cs/authentication-web-apps
9. Carmagnola, Francesca, and Federica Cena. "User identification for cross-system personalisation." Information Sciences 179. – 2009. – 16-32p.
10. Web Authentication Methods Compared [Електронний ресурс]. – Режим доступу: https://testdriven.io/blog/web-authentication-methods/
11. YANG, Yinghui Catherine. Web user behavioral profiling for user identification. Decision Support Systems. – 2010. – 261-271p.
12. Загрози при роботі в Інтернеті і їх уникнення. [Електронний ресурс]. – Режим доступу: https://naurok.com.ua/zagrozi-pri-roboti-v-interneti-i-h-uniknennya-257244.html

13. Li, Xiaowei, and Yuan Xue. "A survey on web application security." Nashville, TN USA 25.5 - 2011. – 1-14p.

14. Робин Никсон, Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 (6-е издание). Print2print, - 2023. 832с.

15. ТЗ на створення сайту: як правильно його скласти, приклади та помилки, які не варто повторювати. [Електронний ресурс]. – Режим доступу: https://creator-systems.com/ua/news/tz-na-stvorennia-saitu-yak-pravylno-yoho-sklasty-pryklady-ta-pomylky-yaki-ne-varto-povtoriuvaty

16. Васильєв О.М. Програмування мовою PHP. Ліра-К, - 2022. 368с.

17. PHP Підручник. [Електронний ресурс]. – Режим доступу: https://w3schoolsua.github.io/php/index.html#gsc.tab=0

18. І.Л.Бородкіна, Г.О.Бородкін, Web-технології та Web-дизайн : застосування мови HTML для створення електронних ресурсів. Ліра-К, - 2020. 212с.

19. Як захистити свій сайт від злому. Топ-10 розповсюджених варіантів злому. [Електронний ресурс]. – Режим доступу: https://fondy.ua/uk/blog/how-to-protect-website/

20. HTML і CSS довідник українською. [Електронний ресурс]. – Режим доступу: https://html-css.co.ua

21. ALA'A, M. Online registration system. International Journal of Computer Science and Security (IJCSS). – 2010. - 331p.

22. Посібник з PHP [Електронний ресурс]. – Режим доступу: https://php.org.ua/manual/uk/index.md

23. Ерік Фрімен, Елізабет Робсон, Head First. Програмування на JavaScript. Фабула – 2022. – 672с.

24. HTML CSS JavaScript PHP SQL [Електронний ресурс]. – Режим доступу: https://w3schoolsua.github.io/index.html#gsc.tab=0

25. Опановуємо основи алгоритмів, або Як прискорити код з 15 до 1000 запитів за секунду. [Електронний ресурс]. – Режим доступу: https://dou.ua/lenta/articles/why-understanding-algorithms-is-important/

26. Ірина Бородкіна, Теорія алгоритмів. Посібник для студентів вищих навчальних закладів. Центр навчальної літератури – 2019. – 184с.

27. Creating a User Login System with PHP and MySQL [Електронний ресурс]. – Режим доступу: https://www.tutorialrepublic.com/php-tutorial/php-mysql-login-system.php

28. Медовз Донелла, Мистецтво мислити системно. Розв'язання проблем від особистого до глобального масштабу - Медовз Донелла. Vivat – 2023. – 304с.

29. Сучасний підручник з JavaScript [Електронний ресурс]. – Режим доступу: https://uk.javascript.info

30. Secure Login System with PHP and MySQL. [Електронний ресурс]. – Режим доступу: https://codeshack.io/secure-login-system-php-mysql/

31. Daniel Nichter, Efficient MySQL Performance: Best Practices and Techniques. O'Reilly Media – 2022. – 335p.

32. How to Do User Testing - User Testing Definition [Електронний ресурс]. – Режим доступу: https://www.leanlabeducation.org/blog/usability-testing-for-edtech-companies

33. 100 Test Cases For Login Page [Електронний ресурс]. – Режим доступу: https://katalon.com/resources-center/blog/test-cases-for-login-page

34. Топ-8 найкращих браузерів на 2023 рік: порівняння. [Електронний ресурс]. – Режим доступу: https://breezee.com.ua/top-8-naikrashchykh-brauzeriv-na-2023-rik-porivniannia-1697636019

35. Gayathri Mohan, Full Stack Testing. A Practical Guide for Delivering High Quality Software. O'Reilly – 2022. – 406p.

36. Тестування веб-проєктів: основні етапи та поради [Електронний ресурс]. – Режим доступу: https://qalight.ua/baza-znaniy/testuvannya-veb-proektiv-osnovni-etapi-ta-poradi/

37. Scott Chacon, Ben Straub, Pro Git. Apress – 2014. – 440p.

38. About the XAMPP project [Електронний ресурс]. – Режим доступу: https://www.apachefriends.org/about.html

39. Git Guides [Електронний ресурс]. – Режим доступу: https://github.com/git-guides

40. В. О. Кузьміних, О. В. Коваль, Р. А. Тараненко, Управління версіями програмних засобів проекту. Київ КПІ ім. Ігоря Сікорського – 2023. – 114с.