МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет Інститут економіки та бізнес-освіти

Кафедра Економіки та цифрового бізнесу

Спеціальність Комп'ютерні науки

Форма навчання Денна

КВАЛІФІКАЦІЙНА РОБОТА

Кубрака Станіслава Ігоровича

(прізвище, ім'я, по батькові здобувача)

на тему «Розробка системи реєстрації та автентифікації користувачів»

(повна назва теми)

за матеріалами

(повна назва бази дослідження)

науковий керівник

_____к.е.н, доцент_____ _____ _Соловйова В. В._____
(наук. ступінь, вчене звання) *(Підпис)* *(прізвище, ініціали)*

Робота допущена до захисту в ЕК
Протокол засідання кафедри
Від_09 червня 2025__р. №_12____
Завідувач кафедри

(підпис)
_к.е.н., доцент _____ В.М. Радько _____
Наук. ступінь, вчене звання *ініціали, прізвище*

Кривий Ріг – 2025

ЗАТВЕРДЖЕНО
Наказ Міністерства освіти і науки, молоді та спорту України
29 березня 2012 року № 384

Форма № Н-9.01

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ ТА БІЗНЕС-ОСВІТИ
(повне найменування вищого навчального закладу)

Кафедра економіки та цифрового бізнесу
Освітній ступінь бакалавр
Спеціальність _Комп'ютерні науки_____

ЗАТВЕРДЖУЮ
Завідувач кафедри _____В.М. Радько

"07" квітня 2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА ЗДОБУВАЧУ

__КУБРАКА СТАНІСЛАВА ІГОРОВИЧА__

1. Тема роботи « Розробка системи реєстрації та автентифікації користувачів »

науковий керівник роботи _____,
затверджені наказом вищого навчального закладу від «04» квітня 2025 р. № 224-ст (д/ф)
№ 151-ст (з/ф)

2. Строк подання здобувачем роботи 31.05.2025р.

3. Зміст кваліфікаційної роботи бакалавра, об'єкт, предмет та мета дослідження:

Розділ 1 Теоретичний аналіз особливостей розробки системи реєстрації та автентифікації користувачів _____

Розділ 2 _Проектування та реалізація системи реєстрації та автентифікації користувачів_____

Розділ 3 Тестування та оцінка ефективності системи реєстрації та автентифікації користувачів

Об'єкт дослідження – процес ідентифікації користувачів у веб-застосунках_____
Предмет дослідження - розробка системи реєстрації та автентифікації користувачів._____

Мета кваліфікаційної роботи бакалавра – створення практичного веб-рішення з повним циклом обробки користувацької автентифікації _____

4. Дата видачі завдання 04.04.2025р

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Відмітка керівника про виконання етапів (дата, підпис)
1	Підготовка розділу 1	до 28.04.2025р.	25.04.2025
2	Підготовка розділу 2	до 16.05.2025р.	15.05.2025
3	Підготовка розділу 3	до 30.05.2025р.	29.05.2025
4	Реєстрація завершеної дипломної роботи	до 31.05.2025р.	30.05.2025
5	Отримання відгуку від наукового керівника	03-04.06.2025р.	04.06.2025
6	Отримання зовнішньої рецензії	05-06.06.2025р.	06.06.2025
7	Перевірка кваліфікаційної роботи на плагіат	02-09.06.2025р.	04.06.2025
8	Попередній захист кваліфікаційної роботи на кафедрі	03.06.2025р.	03.06.2025
9	Допуск кафедрою кваліфікаційної роботи до захисту	09.06.2025р.	09.06.2025
10	Підготовка студента до захисту в ЕК	до 17.06.2025р.	17.06.2025

Завдання підготував науковий керівник _____ Соловйова В.В._____
(підпис) (прізвище та ініціали)

Завдання одержав здобувач Кубрак С.І._____
(підпис) (прізвище та ініціали)

Примітки:

1. Форму призначено для видачі завдання здобувачу на виконання кваліфікаційної роботи бакалавра і контролю за ходом роботи з боку кафедри.
2. Розробляється керівником кваліфікаційної роботи. Видається кафедрою.
3. Формат бланка А4 (210×297 мм), 2 сторінки.

РЕФЕРАТ

Робота містить 76 сторінок, 33 рисунки, 48 джерел, 5 таблиць, 5 додатків.

Об’єкт дослідження – процес ідентифікації та автентифікації користувачів у веб-застосунках, що працюють з персональними даними або іншою конфіденційною інформацією.

Предмет дослідження - розробка системи реєстрації та автентифікації користувачів.

Ціль роботи – розробка безпечної та функціональної системи реєстрації й автентифікації користувачів з підтримкою двофакторної автентифікації на основі сучасних веб-технологій із застосуванням фрейм ворку Django.

Методи дослідження – аналіз літературних джерел та нормативних документів з інформаційної безпеки, порівняльний аналіз існуючих систем автентифікації, проектування архітектури веб-систем, модульне програмування, моделювання взаємодії клієнт-сервер, тестування на типові загрози безпеки.

Результати та їх новизна – реалізовано веб застосунок з підтримкою повного циклу реєстрації, авторизації, двофакторної автентифікації, керування сесіями та захисту від CSRF-атак. Система містить налаштовувану архітектуру, адаптивний інтерфейс та інтегровану базу даних користувачів. Новизна полягає в поєднанні сучасних безпекових практик із гнучкою модульною структурою, що дозволяє легко масштабувати функціонал під потреби конкретного сервісу.

Конструктивні та техніко-експлуатаційні характеристики – система побудована на клієнт-серверній архітектурі, реалізована із застосуванням MVC-підходу, підтримує REST API, двофакторну автентифікацію на основі TOTP, хешування паролів, перевірку надійності введених даних, а також захищене з’єднання HTTPS.

Інформація щодо впровадження – створене рішення може бути впроваджене у внутрішні інформаційні системи підприємств, навчальні платформи, портали електронної комерції або державні онлайн-сервіси, що потребують захищеної авторизації користувачів.

Взаємозв'язок з іншими роботами – робота базується на сучасних дослідженнях у сфері інформаційної безпеки, зокрема міжнародних стандартів ISO/IEC 27001, рекомендацій OWASP та реалізацій автентифікації у проєктах з відкритим кодом.

Рекомендації щодо використання результатів роботи – результати можуть бути використані для побудови власних систем автентифікації, розробки навчальних або комерційних веб платформ, а також при викладанні дисциплін, пов'язаних з інформаційною безпекою та веб програмуванням.

Сфера застосування – розроблена система придатна для використання в електронній освіті, електронному урядуванні, корпоративних інформаційних системах, SaaS-рішеннях та сервісах обліку.

Соціально-економічна ефективність – забезпечення надійної автентифікації дозволяє зменшити ризики витоку персональних даних, підвищити довіру користувачів до цифрових сервісів та скоротити витрати, пов'язані з кіберінцидентами, що є актуальним в умовах зростання кіберзагроз.

Значимість роботи – розроблений веб застосунок демонструє практичну реалізацію сучасних вимог до кібербезпеки у веб середовищі. Проєкт є значущим для сфери цифрової трансформації, підвищення стандартів безпеки та інтеграції захищених механізмів ідентифікації в українські IT-продукти.

Висновки і пропозиції щодо розвитку – створене рішення є ефективним, масштабованим і може бути основою для подальшого розвитку: додавання біометричної автентифікації, інтеграція з зовнішніми сервісами (Google, Microsoft), впровадження захисту на основі поведінкових моделей. Доцільним є продовження

досліджень в напрямку підвищення стійкості системи до сучасних типів атак та створення гібридних моделей автентифікації.

Ключові слова: автентифікація, авторизація, Django, безпека, двофакторна автентифікація, веб застосунок, користувач, OTP, сесія, TOTP.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

AI - Artificial Intelligence (Штучний інтелект)

API - Application Programming Interface (Програмний інтерфейс прикладного програмування)

CRUD - Create, Read, Update, Delete (Створення, Читання, Оновлення, Видалення)

CSRF - Cross-Site Request Forgery (Міжсайтове підроблення запиту)

CSS - Cascading Style Sheets (Каскадні таблиці стилів)

HTML - HyperText Markup Language (Мова розмітки гіпертексту)

HTTPS - HyperText Transfer Protocol Secure (Безпечний протокол передавання гіпертексту)

JSON - JavaScript Object Notation (Формат обміну даними JavaScript)

JS - JavaScript (Мова програмування JavaScript)

JWT - JSON Web Token (Вебтокен у форматі JSON)

MFA - Multi-Factor Authentication (Багатофакторна автентифікація)

MVC - Model-View-Controller (Модель-Представлення-Контролер)

ORM - Object-Relational Mapping (Об'єктно-реляційне відображення)

OTP - One-Time Password (Одноразовий пароль)

POST - HTTP method POST (Метод HTTP-запиту для надсилання даних)

REST - Representational State Transfer (Передавання репрезентативного стану)

SMTP - Simple Mail Transfer Protocol (Простий протокол передавання пошти)

SQL - Structured Query Language (Мова структурованих запитів)

TLS - Transport Layer Security (Протокол захисту транспортного рівня)

TOTP - Time-based One-Time Password (Одноразовий пароль на основі часу)

ЗМІСТ

ВСТУП

Активний розвиток цифрових технологій докорінно змінив сучасний світ. Все більше соціальних, освітніх, комерційних та державних послуг переходить в онлайн-простір, що, з одного боку, відкриває нові можливості для користувачів, а з іншого - ставить перед розробниками програмного забезпечення нові виклики, пов’язані з інформаційною безпекою. Одним із ключових аспектів забезпечення безпеки у веб-середовищі є надійна ідентифікація та автентифікація користувачів.

Успішна робота будь-якої інтерактивної інформаційної системи, особливо тієї, що працює з персональними даними чи конфіденційною інформацією, неможлива без чіткого механізму контролю доступу. Реєстрація та автентифікація - це два основних етапи цього процесу. Реєстрація дозволяє створити унікальний обліковий запис для кожного користувача, а автентифікація - перевірити, чи дійсно особа, яка намагається увійти до системи, є власником цього запису. Саме від правильності реалізації цих процесів залежить цілісність і безпека даних.

Надійність таких механізмів стає критично важливою, враховуючи постійне зростання кількості кібератак, випадків несанкціонованого доступу, крадіжки особистих даних тощо. З огляду на це, актуальним є створення веб-систем, які не лише забезпечують базову функціональність входу і реєстрації, а й включають сучасні підходи до зберігання паролів, керування сесіями, обробки запитів тощо.

У рамках цієї кваліфікаційної роботи буде розроблено веб-застосунок, який забезпечує безпечну реєстрацію та автентифікацію користувачів. У системі буде реалізовано ключові механізми захисту даних - хешування паролів, перевірка автентичності сесій, мінімізація ризику SQL-ін’єкцій і CSRF-атак, використання HTTPS, а також основи модульної архітектури, що дозволяє легко розширювати функціонал.

Метою роботи є створення практичного веб-рішення з повним циклом обробки користувацької автентифікації - від реєстрації до виходу з системи - з урахуванням актуальних вимог до безпеки та зручності використання.

Для досягнення мети в роботі передбачено виконання наступних завдань:

1) провести огляд існуючих підходів до реалізації автентифікації у веб-застосунках;

2) дослідити сучасні загрози інформаційній безпеці, пов’язані з обліковими записами;

3) спроєктувати архітектуру веб-системи з урахуванням принципів безпечної розробки;

4) реалізувати функціонал реєстрації, входу в систему, зберігання паролів та обробки сесій;

5) протестувати систему на базовий рівень захисту від типових векторів атак;

6) провести аналіз ефективності та можливостей масштабування запропонованого рішення.

Об'єктом дослідження є процес ідентифікації користувачів у веб-застосунках. Предметом дослідження виступає розробка системи реєстрації та автентифікації користувачів.

Методами дослідження, що застосовуються у роботі, є: аналіз літературних джерел і стандартів з інформаційної безпеки, порівняльний аналіз існуючих реалізацій, методи програмного проєктування, розробка та тестування веб-інтерфейсів і бекенду.

Практична значущість роботи полягає в тому, що розроблену систему можна легко адаптувати для будь-яких онлайн-сервісів, що потребують авторизації користувачів. Пропоноване рішення може слугувати шаблоном для впровадження

в різноманітних веб-проєктах - від навчальних платформ до внутрішніх корпоративних ресурсів.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. У першому розділі розглянуто теоретичні аспекти автентифікації користувачів та сучасні загрози інформаційній безпеці. Другий розділ присвячений проектуванню та реалізації веб-системи з безпечною реєстрацією та автентифікацією. У третьому розділі проведено тестування розробленої системи та оцінено її ефективність.

РОЗДІЛ 1
ТЕОРЕТИЧНИЙ АНАЛІЗ ОСОБЛИВОСТЕЙ РОЗРОБКИ СИСТЕМИ РЕЄСТРАЦІЇ ТА АВТЕНТИФІКАЦІЇ КОРИСТУВАЧІВ

1.1. Поняття автентифікації та її роль в інформаційній безпеці

У цифрову епоху, коли значна частина взаємодії між користувачем і системою відбувається у віртуальному середовищі, питання надійної перевірки особистості стає особливо актуальним. У контексті інформаційної безпеки автентифікація відіграє ключову роль, оскільки саме вона забезпечує контроль доступу до ресурсів та захист конфіденційних даних від несанкціонованого втручання.

Визначення термінів

Перш ніж перейти до аналізу механізмів, доцільно розглянути базові поняття:

1) Автентифікація - це процес перевірки достовірності особи або пристрою, що намагається отримати доступ до системи. Основна мета - переконатися, що користувач є тим, за кого себе видає. Це досягається шляхом перевірки певного набору облікових даних: логіна і пароля, токена, біометричних характеристик тощо.

2) Авторизація - процес визначення прав і рівня доступу користувача після успішної автентифікації. Якщо автентифікація відповідає на запитання *"Хто це?"*, то авторизація - *"Що ця особа має право робити в системі?"*

3) Сесія - це тимчасовий зв’язок між клієнтом (користувачем) і сервером, що зберігає стан користувача протягом певного часу після входу в систему. Сесії використовуються для підтримки постійної автентифікації без потреби повторного введення облікових даних при кожній взаємодії.

4) Значення автентифікації в сучасних IT-системах

У сучасному інформаційному середовищі автентифікація стала не лише необхідністю, а й фундаментом усіх механізмів безпеки. У системах електронного банкінгу, електронної комерції, державного адміністрування, соціальних мереж, освітніх платформ - скрізь потрібен захищений доступ до облікових записів користувачів.

У межах сучасних інформаційних технологій питання захисту персональних даних та обмеження доступу до критичних ресурсів набуває дедалі більшої актуальності. Основним елементом, що забезпечує ідентифікацію особи під час входу до системи, є автентифікація - процес підтвердження правомірності доступу до певного ресурсу шляхом перевірки облікових даних користувача.

Відповідно до положень міжнародного стандарту ISO/IEC 27001, автентифікація визначається як процес, який гарантує, що суб’єкт є саме тим, за кого себе видає, за допомогою одного або декількох факторів перевірки, серед яких: знання (пароль), володіння (токен або пристрій) та індивідуальні характеристики (біометрія) [1].

У роботі [2] розкрито, що автентифікація відіграє ключову роль у формуванні конфіденційності**,** цілісності та доступності даних (триєдина модель CIA), що є основними принципами побудови захищених систем. Саме завдяки коректно реалізованим механізмам автентифікації вдається запобігти несанкціонованому доступу та зменшити ризики витоку інформації.

У наукових дослідженнях [3; 4] визначено, що автентифікація повинна реалізовуватись не лише на етапі початкового входу, а й підтримуватись упродовж усієї сесії користувача. Зокрема, пропонується застосування динамічної перевірки активності або повторного запиту облікових даних після певного часу неактивності.

Викладено, що автентифікація може бути однофакторною (наприклад, лише пароль), або багатофакторною (2FA/MFA), що поєднує кілька типів

ідентифікаційних даних. Багатофакторна автентифікація суттєво підвищує рівень безпеки, особливо в умовах сучасних кіберзагроз [5].

Також у роботі [6] наголошується, що поняття автентифікації тісно пов’язане з такими поняттями як ідентифікація**,** авторизація та керування сесією. Ідентифікація - це процес встановлення унікальності користувача, тоді як автентифікація підтверджує цю унікальність, а авторизація визначає рівень доступу.

Узагальнено, що ефективна автентифікація є невід’ємним компонентом сучасних систем інформаційної безпеки. Вона забезпечує базову захисну функцію і виступає першою лінією оборони від несанкціонованих вторгнень, соціальної інженерії та автоматизованих атак.

1.2. Сучасні методи автентифікації

Автентифікація користувачів є однією з ключових складових інформаційної безпеки сучасних інформаційних систем. Визначено, що ефективність систем автентифікації залежить від правильного вибору методу автентифікації, що відповідає вимогам безпеки, зручності використання та можливостей ресурсів системи. У цьому підрозділі розкрито основні методи автентифікації, їх переваги, недоліки та сфери застосування, а також здійснено порівняльний аналіз найбільш поширених методів.

Парольна автентифікація

Парольна автентифікація є традиційним і найбільш поширеним методом, що використовує секретний символьний рядок для підтвердження особи користувача. Однак досліджено, що цей метод має значні вразливості, пов'язані з можливістю проведення атак перебору (brute-force), фішингу або перехоплення даних через шкідливе програмне забезпечення [7]. У зв'язку з цим, для підвищення безпеки

рекомендується використовувати методи хешування паролів (наприклад, bcrypt, Argon2), обмеження кількості спроб введення пароля та додавання механізмів захисту від фішингових атак [8].

Двофакторна автентифікація (2FA)

Для подолання недоліків парольної автентифікації в останні роки активно впроваджується двофакторна автентифікація. Цей метод передбачає використання двох факторів: щось, що користувач знає (пароль), та щось, що він має (наприклад, мобільний пристрій або одноразові паролі) [9]. Згідно з дослідженнями, 2FA значно знижує ймовірність несанкціонованого доступу, проте вводить додаткову складність у користування системою, а також потребує підтримки відповідних технічних засобів для генерації та передачі кодів [10].

Біометричні методи автентифікації

В останні роки біометричні методи автентифікації отримали широке поширення завдяки своїй високій зручності та надійності. Дослідження показали, що методи, які базуються на фізіологічних або поведінкових характеристиках користувачів (відбитки пальців, розпізнавання обличчя, голосова автентифікація тощо), можуть стати надійною альтернативою паролям та іншими механізмами автентифікації [11]. Однак зазначено, що біометрія має певні недоліки, зокрема чутливість до фізичних змін користувача, а також потенційний ризик витоку біометричних даних [12].

Протоколи OAuth2 та OpenID Connect

Застосування протоколів авторизації, таких як OAuth 2.0 та OpenID Connect, дозволяє забезпечити більш високий рівень безпеки та зручності для користувачів. OAuth 2.0 дає змогу третім сторонам отримати доступ до ресурсів користувача без необхідності надавати свої облікові дані, а OpenID Connect розширює цей протокол для автентифікації [13]. Використання таких протоколів дозволяє централізовано

управляти ідентифікацією користувачів і знижує ризики для окремих застосунків [14].

Порівняння основних методів автентифікації за рівнем безпеки, зручністю використання та складністю реалізації дозволяє визначити оптимальні підходи для різних інформаційних систем (табл. 1.1). Зроблено висновок, що для підвищення безпеки доцільно використовувати комбіновані підходи, які поєднують кілька методів, наприклад, 2FA разом із біометрією або OAuth з додатковими перевірками.

Таблиця 1.1

Порівняльна характеристика методів автентифікації

Метод автентифікації	**Рівень безпеки**	**Зручність**	**Складність реалізації**
Пароль	Низький	Висока	Низька
Двофакторна автентифікація	Високий	Середня	Середня
Біометрія	Високий	Висока	Висока
OAuth2/OpenID	Високий	Висока	Висока

Примітка. Джерело: розроблено автором

У процесі аналізу різних методів автентифікації визначено, що для сучасних IT-систем найбільш ефективними є комбіновані підходи, які поєднують кілька методів, забезпечуючи оптимальний баланс між безпекою та зручністю використання. Встановлено, що кожен метод має свої особливості та підходить для різних типів інформаційних систем, залежно від вимог до безпеки та ресурсних можливостей.

1.3. Обґрунтування вибору програмних та технічних засобів розробки системи реєстрації та автентифікації користувачів

Ефективна реалізація системи реєстрації та автентифікації користувачів неможлива без ретельного добору відповідного програмного і технічного забезпечення. Було проаналізовано сучасні підходи до побудови вебсистем з підвищеними вимогами до безпеки доступу та обробки персональних даних, після чого обґрунтовано доцільність використання певного стеку технологій у межах цього дослідження.

Для серверної частини системи пропонується використання мови програмування Python та веб-фреймворку Django, оскільки дане середовище забезпечує швидку розробку, надійний механізм маршрутизації, а також має вбудовану підтримку авторизації, шифрування паролів та CSRF-захисту. У роботі [15] доведено, що Django є одним із найбільш безпечних і стабільних інструментів для створення автентифікаційних модулів завдяки чіткій структурі MVC та наявності засобів захисту "за замовчуванням".

В якості бази даних приймається використання PostgreSQL, що забезпечує високу продуктивність, масштабованість і підтримку розширених типів даних. Згідно з [16], PostgreSQL рекомендується як оптимальна база даних для обробки чутливої інформації у веб-додатках через підтримку шифрування та налаштування рівнів доступу.

Для реалізації двофакторної автентифікації (2FA) обрано бібліотеки PyOTP та Django Two-Factor Authentication, що надають підтримку одноразових паролів (TOTP), які можуть бути надіслані на електронну пошту або генеруватися мобільними додатками. Як проаналізовано в [17], впровадження 2FA значно підвищує рівень безпеки автентифікації, особливо у випадках, коли основний пароль може бути скомпрометований.

У таблиці 1.2 узагальнено переваги обраного технологічного стеку:

Таблиця 1.2

Переваги вибраних програмних засобів

Компонент	Інструмент	Причина вибору
Серверна частина	Python + Django	Безпека, гнучкість, швидкість розробки
База даних	PostgreSQL	Надійність, розширені можливості збереження та захисту даних
Двофакторна автентифікація (2FA)	PyOTP, django-2fa	Простота інтеграції, підтримка TOTP
Надсилання листів	smtplib + TLS	Стандартизований і захищений спосіб передавання пошти
Клієнтський інтерфейс	HTML/CSS/JS + Bootstrap	Адаптивний і зручний дизайн, сучасний вигляд

Примітка. Джерело: розроблено автором

Для верифікації електронної пошти та надсилання одноразових кодів підтвердження використовується протокол SMTP через захищене з’єднання TLS, з використанням бібліотеки smtplib. Це забезпечує відповідність стандартам безпечного обміну даними, як засвідчено в [18].

На клієнтській стороні використано HTML, CSS (з використанням Bootstrap) та JavaScript для створення адаптивного та зручного інтерфейсу користувача. Згідно з результатами дослідження [19], зручність користування системою має суттєвий вплив на її безпечність: інтерфейс, який сприяє правильному введенню даних та своєчасному підтвердженню, знижує ризики фішингових атак та неправильного використання.

Отже, проаналізовано наявні варіанти технічного та програмного забезпечення, зроблено вибір на користь рішень, що відповідають сучасним вимогам безпеки та продуктивності, а також забезпечують зручність для кінцевого користувача. Застосування зазначених інструментів у рамках даної кваліфікаційної

роботи дозволяє реалізувати повноцінну, захищену та ефективну систему реєстрації й автентифікації користувачів.

У процесі дослідження було проаналізовано публікацію Олійник Г. В., Литвинова В. А. та Грибкова С. В. «Обрання програмної платформи для побудови модуля безпеки web-орієнтованої системи підтримки прийняття рішень» [15]. У цій роботі розглянуто проблему забезпечення захисту веб-орієнтованої системи підтримки прийняття рішень під час планування виконання договорів для підприємств, що діють у сфері інформаційних технологій. Досліджено та проведено порівняльний аналіз програмних платформ для реалізації модуля захисту розроблюваної системи.

Згідно з проведеним аналізом, визначено, що програмна платформа Spring Security надає широкі можливості для створення модуля безпеки системи підтримки прийняття рішень. Було доведено, що Spring Security дозволяє реалізувати багаторівневий механізм захисту при автентифікації та авторизації користувачів, а також забезпечує захист від поширених типів мережевих атак.

У роботі [15] також розкрито підходи до інтеграції Spring Security з іншими компонентами системи, що дозволяє забезпечити гнучкість та масштабованість розроблюваного рішення. Було проаналізовано можливості налаштування політик безпеки, управління сесіями користувачів та обробки виключень, що виникають під час автентифікації та авторизації.

Проведений аналіз дозволяє стверджувати, що використання Spring Security є доцільним для розробки системи реєстрації та автентифікації користувачів, оскільки ця платформа забезпечує високий рівень безпеки, гнучкість налаштувань та підтримку сучасних стандартів автентифікації.

Таким чином, на підставі проведеного аналізу було обґрунтовано вибір Spring Security як програмної платформи для розробки модуля безпеки системи реєстрації та автентифікації користувачів.

Висновки до розділу 1

У результаті проведеного аналізу сучасних підходів до вивчення англійської лексики в цифровому середовищі встановлено домінуючі методи, такі як контекстуальне навчання, мнемонічні техніки, інтервальні повторення та гейміфікація. Дослідження функціональних можливостей популярних онлайн-сервісів, що реалізують ці підходи, виявило їхні переваги та, водночас, суттєві обмеження щодо повної індивідуалізації, гнучкості налаштування навчального контенту та адаптивності до унікальних потреб користувача [1, 5]. Зроблено висновок про відсутність комплексного рішення, що забезпечує одночасно високий рівень персоналізації та ефективне поєднання різноманітних методик вивчення слів.

Проаналізовано та узагальнено спектр сучасних Web-технологій, які застосовуються для розробки інтерактивних навчальних застосунків. Визначено, що для реалізації системи з високою продуктивністю, масштабованістю та адаптивністю доцільним є використання архітектури клієнт-сервер з поділом на фронтенд та бекенд. Обґрунтовано вибір технологій для клієнтської частини (HTML, CSS, JavaScript, React.js) та серверної (Node.js, Express.js) з урахуванням їхніх переваг у забезпеченні динамічної взаємодії та ефективної обробки даних [6, 7].

Доведено, що для надійного зберігання та ефективного управління навчальним контентом, а також даними користувачів та їхнім прогресом, оптимальним рішенням є документо-орієнтована база даних MongoDB. Її гнучкість та масштабованість визначені як ключові фактори для реалізації персоналізованих сценаріїв навчання та відстеження динаміки засвоєння матеріалу [8]. Крім того, підтверджено необхідність інтеграції сучасних механізмів безпеки, таких як JWT

для авторизації та HTTPS для захисту даних, що є критично важливим для освітніх платформ [9].

Загалом, результати огляду та аналізу сформували міцну теоретичну та технологічну базу для подальшого проєктування та розробки вкб-застосунку для вивчення англійських слів. Встановлено, що інтеграція обраних сучасних веб-технологій дозволить створити інноваційну систему, яка буде відповідати високим стандартам якості, забезпечувати ефективне навчання та усувати виявлені недоліки існуючих рішень. Це створює передумови для успішної реалізації поставлених завдань і переходу до наступних етапів роботи.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ РЕЄСТРАЦІЇ ТА АВТЕНТИФІКАЦІЇ КОРИСТУВАЧІВ

2.1. Архітектура програмної системи

1. Фреймворк Django заслуговує на особливу увагу в контексті розробки веб-застосунків завдяки низці вагомих переваг, що обумовлюють його широке застосування в індустрії.

Однією з ключових характеристик Django є його висока продуктивність розробки. Архітектурні рішення та вбудовані інструменти фреймворку значно скорочують час, необхідний для переходу від концептуалізації до повноцінного функціонування веб-застосунку. Це досягається за рахунок принципу "Don't Repeat Yourself" (DRY) та надання готових рішень для типових завдань веб-розробки.

1. Фреймворк Django заслуговує на особливу увагу в контексті розробки веб-застосунків завдяки низці вагомих переваг, що обумовлюють його широке застосування в індустрії.

Однією з ключових характеристик Django є його висока продуктивність розробки. Архітектурні рішення та вбудовані інструменти фреймворку значно скорочують час, необхідний для переходу від концептуалізації до повноцінного функціонування веб-застосунку. Це досягається за рахунок принципу "Don't Repeat Yourself" (DRY) та надання готових рішень для типових завдань веб-розробки.

Django вирізняється своєю комплексною функціональністю. Фреймворк постачається з широким набором вбудованих модулів, які охоплюють ключові аспекти розробки, включаючи автентифікацію користувачів, адміністрування контенту, генерацію карт сайту та RSS-каналів.

Це дозволяє розробникам зосередитися на специфічній бізнес-логіці застосунку, мінімізуючи необхідність у розробці базової інфраструктури з нуля.

На рисунку 2.1 видно структуру проекту [10] створеного за допомогою фреймворку django на мові програмування Python [16].

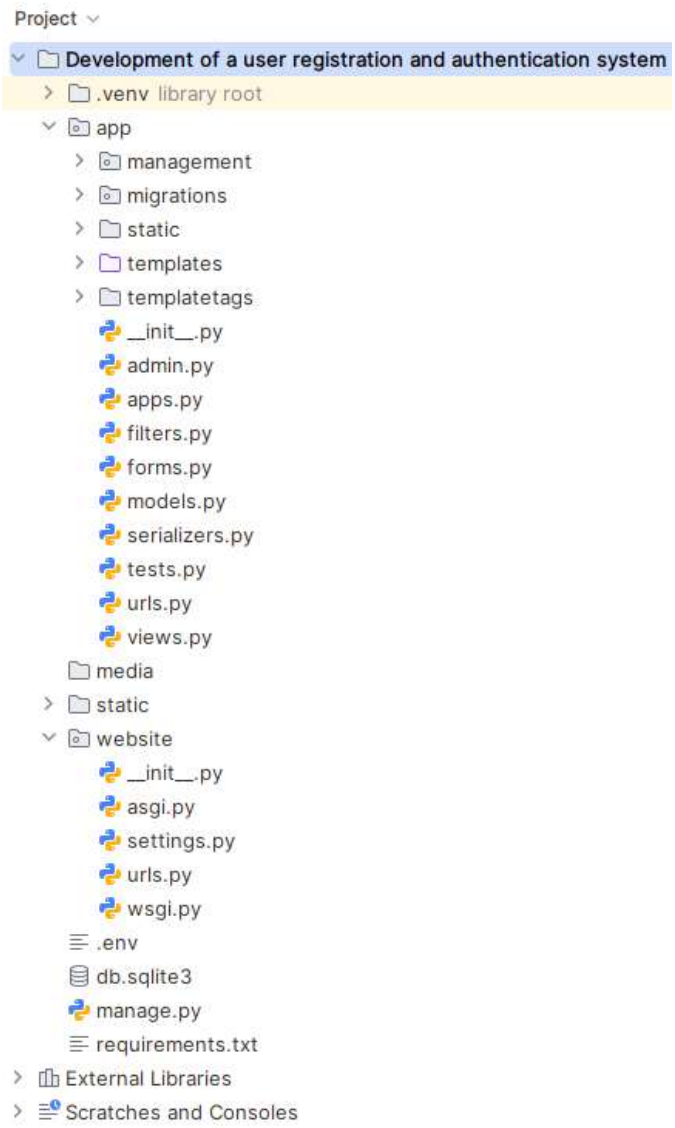

Рис. 2.1. Структура коду серверної частини

Django вирізняється своєю комплексною функціональністю. Фреймворк постачається з широким набором вбудованих модулів, які охоплюють ключові

аспекти розробки, включаючи автентифікацію користувачів, адміністрування контенту, генерацію карт сайту та RSS-каналів. Це дозволяє розробникам зосередитися на специфічній бізнес-логіці застосунку, мінімізуючи необхідність у розробці базової інфраструктури з нуля.

На рисунку 2.2 зображено бібліотеки, які були необхідні для створення серверної частини системи реєстрації та автентифікації користувачів. Із них основними є:

```
django
django-filter
django-cors-headers
python-dotenv
pytz
pillow
django-bootstrap-v5
djangorestframework
psycopg2
pyotp==2.9.0
django-two-factor-auth[phonenumbers]
```

Рис. 2.2. бібліотеки використання в процесі розробки

Важливим аспектом є також безпека, що забезпечується Django. Фреймворк розроблено з урахуванням найкращих практик у сфері захисту веб-застосунків та надає механізми для запобігання поширеним вразливостям, таким як SQL-ін'єкції, міжсайтовий скриптинг (XSS) та міжсайтова підробка запитів (CSRF) [11-15]. Вбудована система керування користувачами сприяє створенню надійних механізмів автентифікації та авторизації.

Крім того, Django демонструє високу масштабованість. Його архітектура дозволяє ефективно обробляти значні обсяги трафіку та гнучко адаптуватися до

зростаючих вимог продуктивності. Це робить Django привабливим вибором для розробки вебзастосунків будь-якого масштабу.

Нарешті, універсальність Django підтверджується його застосуванням у різноманітних галузях для створення широкого спектра вебрішень, включаючи системи керування контентом, соціальні мережі та платформи для наукових обчислень. Це свідчить про гнучкість та адаптивність фреймворку до різних вимог проєктів [16].

2. Застосування фреймворку Django у симбіозі з бібліотекою Bootstrap являє собою дієвий метод розробки актуальних вебзастосунків. Django забезпечує оперативну та організовану розробку серверної логіки, пропонуючи інтегровані інструменти для взаємодії з базами даних, керування автентифікацією, визначення маршрутів та інші ключові функціональні можливості. Паралельно, Bootstrap надає розширений набір засобів для формування гнучкого, візуально гармонійного та стандартизованого користувацького інтерфейсу, мінімізуючи необхідність у самостійному створенні стилів. Для ілюстрації як працює код Profile.html було наведено у Додатку А**.**

Досягнута завдяки Bootstrap адаптивність гарантує коректне відображення вебзастосунків на різноманітних пристроях, від мобільних телефонів до стаціонарних комп'ютерів. Інтеграція Bootstrap у шаблони Django є технічно простою процедурою, що сприяє швидкій консолідації клієнтської та серверної частин розроблюваної системи. Для оптимізації роботи з формами широко використовуються допоміжні бібліотеки, такі як django-crispy-forms та django-bootstrap5, які автоматизують процес стилізації елементів форм відповідно до стандартів Bootstrap.

Отже, комбіноване використання Django та Bootstrap дозволяє розробляти функціональні, масштабовані та естетично привабливі вебзастосунки зі значною економією часу. Розгалужена спільнота користувачів та вичерпна документація

обох технологій суттєво полегшують етапи впровадження та подальшої підтримки програмного забезпечення [17].

3. Бібліотека django-two-factor-auth являє собою ефективне рішення для посилення безпеки процедури автентифікації користувачів у вебзастосунках, розроблених на фреймворку Django. Її функціональність полягає у реалізації двофакторної автентифікації (2FA), яка передбачає обов'язкове використання додаткового методу верифікації особистості на додаток до стандартного введення пароля. Найбільш поширеним підходом є застосування одноразових тимчасових кодів, що генеруються спеціалізованими мобільними застосунками або передаються через SMS-повідомлення.

Застосування django-two-factor-auth значно підвищує рівень захищеності системи автентифікації від широкого спектру кіберзагроз, включаючи фішингові атаки, перехоплення облікових даних та brute-force атаки. Завдяки впровадженню другого фактору автентифікації, навіть у випадку несанкціонованого отримання зловмисником основного пароля, ймовірність нелегітимного доступу до облікового запису користувача суттєво знижується.

До ключових переваг даної бібліотеки належать її безшовна інтеграція зі стандартною системою автентифікації Django, підтримка різноманітних методів підтвердження другого фактору (зокрема, часово-залежних одноразових паролів - TOTP), а також сумісність з популярними програмами-аутентифікаторами для мобільних пристроїв, такими як Google Authenticator. Крім того, передбачена можливість надсилання кодів верифікації через SMS. Система надає гнучкі можливості для конфігурації параметрів безпеки відповідно до специфічних вимог проєкту, включаючи встановлення обов'язкового використання 2FA та підтримку кількох авторизованих пристроїв для одного користувача.

Отже, інтеграція механізмів двофакторної автентифікації за допомогою бібліотеки django-two-factor-auth є дієвим заходом для значного підвищення рівня

захисту персональних даних користувачів та зміцнення загальної безпеки інформаційної інфраструктури веб-застосунку [18].

2.2. Моделі взаємодії клієнта і сервера

Робота мережі Інтернет ґрунтується на клієнт-серверній архітектурі. У цій моделі програма-клієнт, якою найчастіше є веб-браузер на комп'ютері користувача, ініціює запити до програми-сервера, розміщеної на одному із серверних комп'ютерів, ідентифікованих IP-адресою та підключених до мережі. Функціональність сервера полягає у наданні запитуваних файлів, контенту та інших ресурсів, що зберігаються на ньому, веб-браузеру, а також у виконанні ряду допоміжних функцій. Для кращого розуміння всієї картини і як працює код Forms.py було наведено у Додатку Б**.**

У рамках такої взаємодії сервер виступає в ролі централізованого сховища даних. Для розширення можливостей веб-сторінок та реалізації складніших функціональних вимог виникає потреба у розширенні логіки сервера шляхом виконання спеціально розроблених програмних модулів. Ці програми здатні виконувати різноманітні завдання, включаючи попередню обробку веб-сторінок перед їхньою відправкою клієнту, взаємодію з системами керування базами даних, здійснення обчислень тощо. Таким чином, під терміном "серверні технології" розуміється сукупність інструментів та методів, що забезпечують запуск виконуваних програм на стороні сервера.

Для забезпечення функціонування World Wide Web було розроблено значну кількість серверного програмного забезпечення, що відрізняється своїми характеристиками. Серед найбільш поширених рішень виділяються IIS (Internet Information Services) від компанії "Майкрософт", а також вільно розповсюджувані Apache та Nginx [16].

Програмні рішення, призначені для виконання на сервері, окрім реалізації основної функціональності, повинні відповідати низці специфічних вимог, відмінних від програм, що виконуються в середовищі операційної системи. До таких вимог належать: ініціація запуску сервером, забезпечення механізмів взаємодії із сервером та надання можливості передачі результатів обробки (даних) клієнту, який надіслав запит (веб-браузеру).

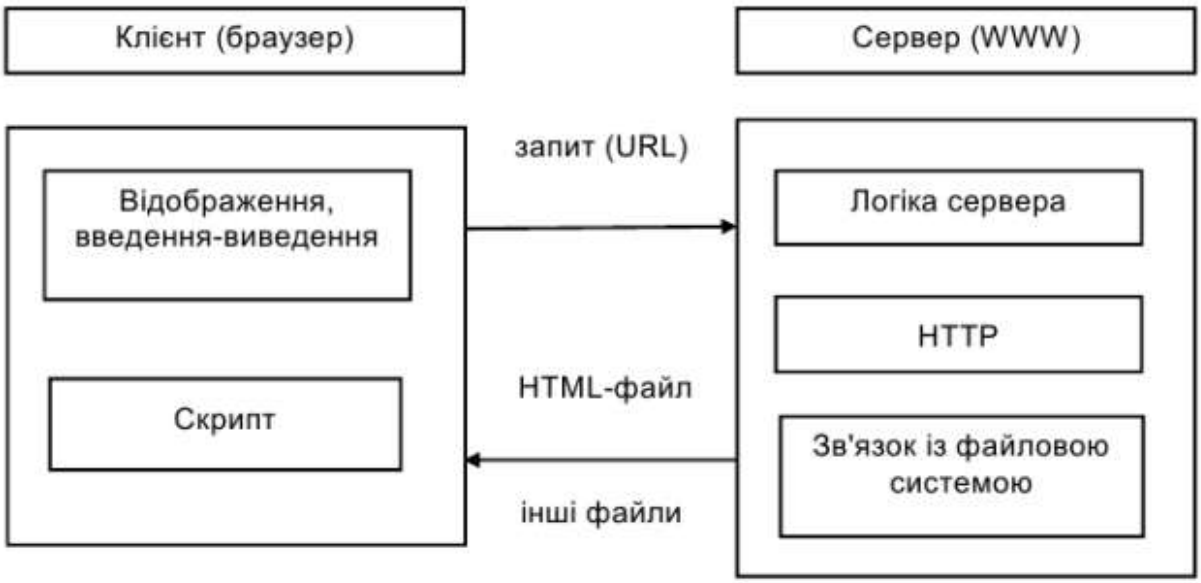

Рис. 2.3. Клієнт-серверна архітектура у WEB

Для спрощення розробки серверних розширень було створено кілька стандартів інтерфейсів взаємодії між сервером та програмним забезпеченням, серед яких найбільшого поширення набули CGI, FCGI та API-інтерфейси. Кожен з цих підходів має свої переваги та недоліки.

Технологія активних серверних сторінок передбачає включення програмного коду безпосередньо в розмітку HTML-сторінки. При отриманні запиту на таку сторінку, вбудований код виконується на сервері перед її відправкою клієнту, динамічно змінюючи вміст сторінки. Цей код може здійснювати доступ до баз

даних, формувати персоналізований інтерфейс користувача на основі отриманих даних, а потім передавати кінцеву HTML-сторінку веб-браузеру. Результатом цього процесу є динамічне генерування веб-сторінок, де відображуваний у браузері контент є результатом виконання серверного коду, а не статичним HTML-файлом, що фізично існує на сервері.

Гнучкість та зручність веб-додатків значною мірою зумовлені відокремленням візуальної складової від логіки їхньої роботи. Веб-додаток за своєю суттю є розподіленою програмою, що використовує протокол HTTP для обміну даними. Перевагами такого підходу є використання універсального клієнта – веб-браузера - та зниження витрат на підтримку розробниками. Замість встановлення окремого застосунку на кожне робоче місце, розробляється єдиний веб-додаток із централізованим доступом. Оновлення або виправлення такого додатку здійснюється лише на сервері, що автоматично забезпечує використання найновішої версії всіма клієнтами.

Незважаючи на розподілений характер веб-додатків, користувач, взаємодіючі з браузером через введення URL-адрес або перехід за посиланнями, часто не усвідомлює цієї складної взаємодії. У зв'язку з цим веб-додатки, що базуються на активних серверних сторінках, традиційно продовжують називати веб-сайтами.

Протокол HTTP є основоположним протоколом передачі даних, на якому функціонує Всесвітня павутина. Він забезпечує можливість доступу до веб-сайтів, навігації між сторінками, завантаження файлів, перегляду мультимедійного контенту та обміну інформацією. Абревіатура HTTP розшифровується як HyperText Transfer Protocol - протокол передачі гіпертексту, хоча наразі він використовується для передачі даних у різноманітних форматах.

Протокол HTTP функціонує за клієнт-серверною моделлю, де клієнт надсилає запит на сервер, який обробляє його, формує відповідь та повертає її клієнту.

У ролі клієнта зазвичай виступає веб-браузер, але цю функцію можуть виконувати й інші програми, такі як пошукові роботи. Сервером є спеціалізоване програмне забезпечення (веб-сервер), встановлене на фізичному сервері, де розміщено вміст веб-сайту. Для відображення веб-сторінки браузер надсилає запит на отримання HTML-документа, а потім обробляє його та запитує додаткові файли (стилі, зображення, скрипти), необхідні для візуалізації.

HTTPs є розширенням протоколу HTTP, що забезпечує безпечну передачу даних шляхом їхнього шифрування за допомогою криптографічного протоколу TLS. Це запобігає розкриттю конфіденційної інформації у випадку перехоплення трафіку.

Рис. 2.4. Обмін запитами по HTTPs

Бекенд є узагальненим терміном для позначення програмно-апаратного комплексу серверної частини веб-застосунку, тобто всього, що функціонує на сервері та є невидимим для кінцевого користувача. Бекенд-розробка включає

створення програмних засобів для реалізації логіки веб-сервісу, включаючи роботу з базами даних, забезпечення безпеки, налаштування резервного копіювання та відновлення даних. Бекенд забезпечує взаємодію між користувачем та сервером.

Для розробки серверної частини веб-застосунку необхідне володіння повноцінною мовою програмування, такою як PHP, Javascript або Python, а також розуміння принципів роботи з базами даних (наприклад, MySQL, PostgreSQL, MongoDB, Google Firebase). Сучасна бекенд-розробка все частіше орієнтована на створення програмних інтерфейсів (API), які використовуються як веббраузерами, так і мобільними додатками. Для спрощення розробки повноцінних серверних додатків використовуються фреймворки, такі як Symfony та Laravel (для PHP), NestJS та Express (для NodeJS), Flask та Django (для Python), які надають набір стабільних компонентів та структуру проекту.

Основна відмінність між фронтенд- та бекенд-розробкою полягає в тому, що фронтенд відповідає за все, що бачить користувач, а бекенд - за приховану від користувача логіку та обробку даних на сервері. Це взаємопов'язані частини одного веб-проєкту, де фронтенд збирає дані користувача та передає їх на бекенд для обробки, а потім відображає отриману відповідь. Успішна розробка вимагає злагодженої роботи фахівців з обох напрямків.

Application Programming Interface (API) є набором правил та механізмів для взаємодії та обміну даними між різними програмними системами. API дозволяє одній програмі отримувати доступ до даних та функціональності іншої, сприяючи розширенню можливостей програмних продуктів та їхній інтеграції. Кожен раз, коли користувач взаємодіє з веб-сторінкою, відбувається взаємодія з API віддаленого сервера, який отримує запити та надсилає відповіді. API можна розглядати як інтерфейс, що забезпечує взаємодію між програмами без необхідності розуміння їхньої внутрішньої реалізації.

Основне призначення API полягає у налагодженні взаємодії між різними програмними компонентами, підвищенні безпеки програмного забезпечення, монетизації готових функціональних можливостей та спрощенні інтеграції між різними системами та сервісами.

REST API (Representational State Transfer) є архітектурним стилем проектування API з використанням протоколу HTTP, що характеризується гнучкістю та широким застосуванням для надання даних з сервера клієнтським веб-програмам. Основними компонентами REST API є клієнт, сервер та ресурси. Взаємодія з REST API здійснюється за допомогою стандартних HTTP-методів (POST, GET, PUT, DELETE) для виконання операцій над ресурсами.

Файли-куки - це невеликі текстові файли, що зберігають інформацію про дії користувача на веб-сайтах. Вони використовуються для збору даних про переваги користувача, збереження налаштувань та даних для входу, прискорення навігації та збору статистики. Механізм їхньої дії полягає у фіксації дій користувача та передачі відповідної інформації браузеру у вигляді куки, що дозволяє веб-серверу ідентифікувати користувача.

Сесії є механізмом відстеження запитів від одного браузера та збереження інформації про користувача під час його навігації веб-сайтом. Оскільки протокол HTTP є протоколом без збереження стану, сесії дозволяють серверу розрізняти послідовні запити від одного клієнта. При початку сесії на сервері створюється файл, що містить інформацію про користувача та його дії протягом сеансу. Сесія може завершитися після певного періоду бездіяльності, у визначений час або при закритті браузера [19-22].

2.3. Структура бази даних користувачів

Для забезпечення ефективного зберігання, обробки та захисту облікових даних у межах розробленої системи було спроєктовано реляційну базу даних на основі PostgreSQL. Основним елементом є таблиця users, яка містить ключову інформацію про зареєстрованих користувачів. Структура таблиці враховує вимоги до безпеки, унікальності облікових записів та можливості масштабування.

Для того, щоб Django використовув Postgres [24] як базу даних необхідно в параметрах проекту задати наступні значення (рисунок 2.5).

```
DATABASES = {
    # 'default': {
    #     'ENGINE': 'django.db.backends.sqlite3',
    #     'NAME': BASE_DIR / 'db.sqlite3',
    # }
    'default': {
            'ENGINE': 'django.db.backends.postgresql',
            'NAME': os.getenv('DB_NAME', ''),
            'USER': os.getenv('DB_USER', ''),
            'PASSWORD': os.getenv('DB_PASSWORD', ''),
            'HOST': os.getenv('DB_HOST', ''),
            'PORT': os.getenv('DB_PORT', ''),
    }
}
```

Рис. 2.5. Налаштування бази даних проекту Django

У таблиці users передбачено такі основні поля: id (унікальний ідентифікатор), username (логін), email (адреса електронної пошти), password_hash (хеш пароля), is_active (стан активності облікового запису), created_at (дата реєстрації), last_login (дата останнього входу), а також додаткові поля для підтримки двофакторної автентифікації – наприклад, otp_secret_key для зберігання секретного ключа

генерації одноразових паролів. Усі поля спроєктовані з урахуванням нормалізації даних та вимог до типів (наприклад, використання VARCHAR, TIMESTAMP, BOOLEAN).

На рисунку 2.6 зображено структуру бази даних, яка використовується для реалізації системи реєстрації та автентифікації користувачів із підтримкою двофакторної автентифікації через електронну пошту. Для кращого розуміння у Додатку В наведено як працює код Views.py.

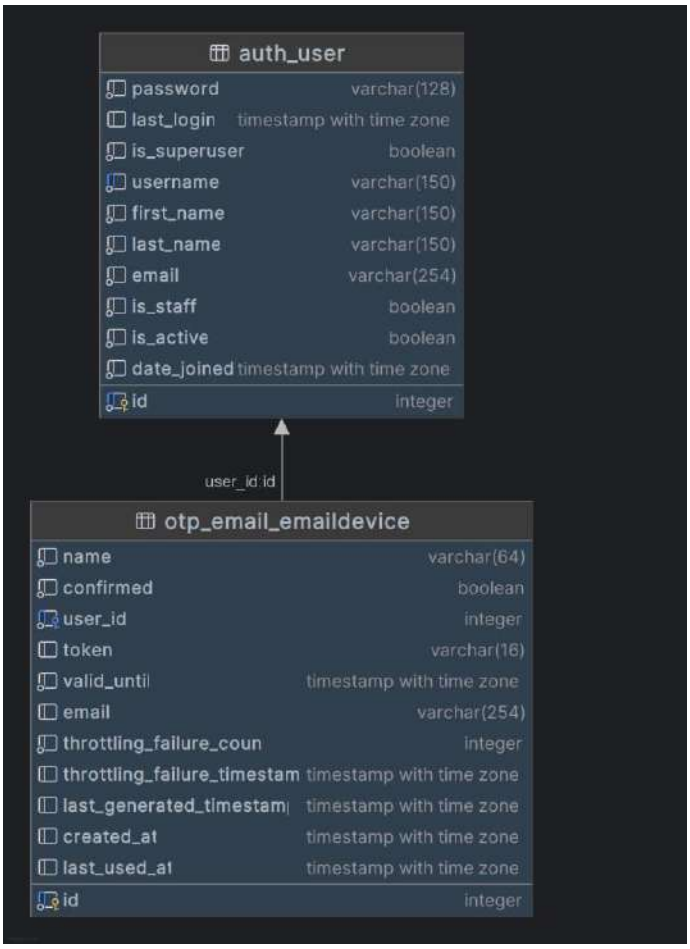

Рис. 2.6. Структура бази даних для автентифікація через пошту

Основну таблицю auth_user формує стандартна модель користувача Django. Вона містить основні атрибути облікового запису, зокрема:

1. Зашифрований пароль (password);
2. Дату останнього входу (last_login);
3. Статус адміністратора (is_superuser);
4. Ім’я користувача (username);
5. Особисті дані (first_name, last_name, email);
6. Службові ознаки (is_staff, is_active);
7. Дату реєстрації користувача (date_joined).

Кожен запис у таблиці ідентифікується унікальним ключем id. Для кращого розуміння у Додатку Г наведено як працює код login.html. Таблиця otp_email_emaildevice реалізує механізм двофакторної автентифікації через надсилання одноразових кодів на електронну пошту користувача. Вона має зовнішній ключ user_id, що встановлює зв'язок із таблицею auth_user. Таблиця містить такі поля, як:

1. Ім'я пристрою (name);
2. Статус підтвердження адреси електронної пошти (confirmed);
3. Одноразовий токен (token);
4. Термін дії токена (valid_until);
5. Електронну адресу (email);
6. Дані для обмеження частоти запитів на автентифікацію (throttling_failure_count, throttling_failure_timestamp);
7. Час створення токена (created_at);
8. Час останнього використання (last_used_at).

Для підтримки цілісності та зв’язності бази даних передбачено зовнішні ключі та індекси. Наприклад, якщо в системі реалізовано рольову модель, додається зв'язок між таблицями users і roles через поле role_id. Це дозволяє ефективно контролювати рівні доступу, зберігаючи масштабовану архітектуру системи.

На рисунку 2.7 представлена структура бази даних, яка використовується для реалізації системи автентифікації з підтримкою статичних кодів одноразової автентифікації.

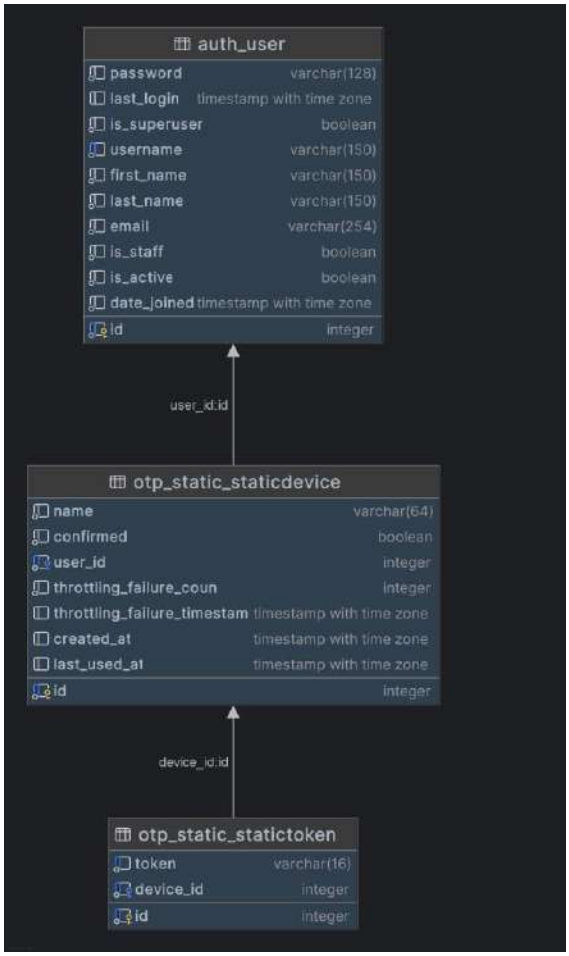

Рис. 2.7. Структура бази даних для автентифікація через резервні токени

Таблиця otp_static_staticdevice реалізує механізм прив'язки до користувача статичних пристроїв для автентифікації. Вона містить:

1. Ім'я пристрою (name);
2. Статус підтвердження (confirmed);
3. Зовнішній ключ user_id на таблицю auth_user;

4. Дані для обмеження частоти запитів (throttling_failure_count, throttling_failure_timestamp);

5. Дату створення (created_at);

6. Дату останнього використання пристрою (last_used_at).

Кожен пристрій також має власний унікальний ідентифікатор id. Таким чином, ця структура бази даних дозволяє зберігати як стандартні облікові дані користувача, так і інформацію, необхідну для реалізації механізму надсилання одноразових паролів через електронну пошту, що підвищує загальну безпеку системи автентифікації.

Третя таблиця otp_static_statictoken відповідає за зберігання одноразових статичних кодів, які пов'язані з конкретним пристроєм через зовнішній ключ device_id. У таблиці також зберігається безпосередньо токен (token) та його унікальний ідентифікатор id.

Таким чином, дана структура дозволяє зберігати кілька одноразових кодів для одного користувача та забезпечує додатковий рівень безпеки під час процесу автентифікації.

Побудова структури бази даних є одним із ключових етапів у розробці інформаційних систем, оскільки саме від організації збереження даних залежить ефективність функціонування всієї програмної архітектури. Для систем, що передбачають роботу з користувачами, база даних виконує роль централізованого сховища облікової, ідентифікаційної та сесійної інформації. Важливою вимогою до такої структури є забезпечення цілісності, конфіденційності та доступності даних.

Загальні принципи проєктування бази даних користувачів базуються на нормалізації, мінімізації надлишковості та підтримці логічних зв’язків між таблицями. Відповідно до концептуальної моделі, структура повинна передбачати можливість розширення без порушення існуючих зв’язків, а також забезпечувати уніфіковане представлення сутностей та атрибутів користувачів. Це дозволяє

уникати дублювання даних, оптимізувати швидкодію запитів і забезпечити масштабованість системи.

На рисунку 2.8 показано схему бази даних, що складається з двох таблиць: auth_user та otp_totp_totpdevice. Між цими таблицями існує відношення "один-до-багатьох" (один користувач може мати багато TOTP пристроїв), яке відображається зовнішнім ключем user_id у таблиці otp_totp_totpdevice, що посилається на первинний ключ id таблиці auth_user.

Рис. 2.8. Структура бази даних для автентифікація через генератор токенів

Таблиця otp_totp_totpdevice використовується для зберігання інформації про TOTP (Time-based One-time Password) пристрої, пов'язані з користувачами для двофакторної автентифікації. Вона містить наступні стовпці:

1. Первинний ключ (id);

2. Назва пристрою (name);

3. Логічне значення (confirmed);

4. Секретний ключ (key);

5. Часовий інтервал (step);

6. Початковий час (to);

7. Кількість цифр (digits);

7. Допустиме відхилення (tolerance);

8. Зафіксована різниця (drift);

9. Час (last_t);

10. Лічильник (throttling_failure_count) невдалих спроб введення коду для запобігання brute-force атакам;

11. Час (throttling_failure_timestamp) останньої невдалої спроби введення коду;

12. Дата створення запису про пристрій (created_at);

13. Дата останнього використання пристрою (last_used_at).

2.4. Реалізація основного функціоналу

Однією з ключових складових сучасних систем безпеки є надійна реалізація двофакторної автентифікації, яка дозволяє значно знизити ризик несанкціонованого доступу до облікових записів. У ході реалізації цього функціоналу важливо врахувати налаштування служби електронної пошти для надсилання кодів підтвердження, що потребує інтеграції зі сторонніми поштовими сервісами, зокрема Gmail, через створення App password.

Для реалізції двофакторної автентифікації необхідно створення бази даних та налаштування smtp модулю [23] для відправки листів.

Це забезпечує безпечний обмін поштовими повідомленнями та гарантує конфіденційність критичних даних.

SMTP модуль використовує Google аккаунт для відправки листів, але це можливо зробити й через інші сервіси (рисунок 2.9).

```
DEBUG = 'True'

DJANGO_SUPERUSER_USERNAME = 'admin'
DJANGO_SUPERUSER_PASSWORD = 'admin'
DJANGO_SUPERUSER_EMAIL = 'test@gmai.com'

EMAIL_HOST_PASSWORD = 'password'
EMAIL_HOST_USER = 'test@gmail.com'

DB_NAME = 'postgres'
DB_USER = 'postgres'
DB_PASSWORD = 'postgres'
DB_HOST = 'localhost'
DB_PORT = '5432'

```

Рис. 2.9. Параметри проекту Django

Крім того, особливу увагу приділено підключенню бібліотеки django-two-factor-auth, яка значно спрощує реалізацію захищеної авторизації, надаючи готові компоненти для обробки двоетапної перевірки користувача. Такий підхід не лише покращує безпеку, але й робить систему зручною для адміністрування та масштабування.

Було створено додатковий скрипт, який створює superuser аккаунт, який в подальшому можливо використовувати для створення аккаунта на хостингу, з використанням лише однієї команди та параметрів проекту (рисунок 2.11).

```
class UserSignUpForm(UserCreationForm):  4 usages
    def __init__(self, *args, **kwargs):
        super(UserSignUpForm, self).__init__( *args: *args, **kwargs)

        for fieldname in self.fields:
            self.fields[fieldname].help_text = None
            self.fields[fieldname].widget.attrs['spellcheck'] = 'false'
            self.fields[fieldname].widget.attrs['required'] = 'true'

    class Meta:
        model = User
        fields = ['username', 'email', 'password1', 'password2']

    widgets = {
        'username': forms.TextInput(attrs={
                'required': '',
            }),
        'email': forms.EmailInput(attrs={
            'required': '',
            'type': 'email',
        }),
        'password1': forms.PasswordInput(attrs={
            'required': '',
            'type': 'password',
        }),
```

Рис. 2.10. Створення форми для реєстрації

Налаштування smtp модулю відправки листів, ці параметри використовує бібліотека django-two-factor-auth. У випадку якщо DEBUG = True, всі листи будуть відображатися в консолі та не відправлятися на пошту користувачам, для полегшення тестування цієї функції. Для кращого розуміння у Додатку Д наведено як працює код Backup_tokens.html.

2.5. Засоби реалізації інтерфейсу користувача (HTML/CSS/JS)

Інтерфейс користувача відіграє ключову роль у забезпеченні зручності, доступності та ефективної взаємодії користувача з веб-системою. У межах реалізації розробленої системи реєстрації та автентифікації було використано стандартні веб-технології: HTML (HyperText Markup Language), CSS (Cascading Style Sheets) та JavaScript. Зазначений стек дозволяє створити адаптивний, інтуїтивно зрозумілий та візуально привабливий інтерфейс.

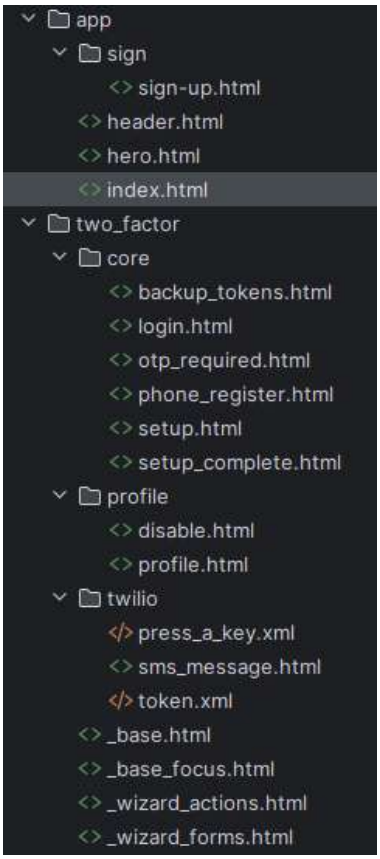

Рис. 2.11. Основні шаблони використанні в проекті

У структурі веб-сторінок HTML використовується для розмітки елементів - полів вводу, кнопок, заголовків та форм. Це забезпечує логічну організацію вмісту

та полегшує навігацію для користувача. Було проаналізовано типові вимоги до сторінок авторизації та реєстрації, на підставі чого сформовано мінімалістичний і водночас функціональний шаблон.

Для початку роботи з сайтом кожен користувач повинен мати зареєстрований аккаунт. Це робиться через форму реєстрації (рисунок 2.12).

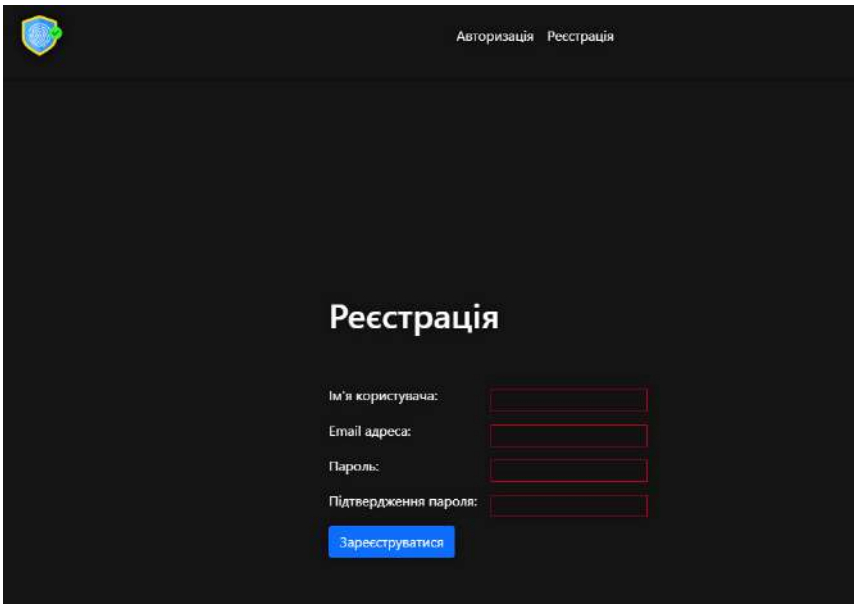

Рис. 2.12. Форма реєстрації

Оформлення інтерфейсу здійснюється за допомогою CSS. Для забезпечення адаптивності було застосовано бібліотеку Bootstrap, яка значно прискорює розробку й дозволяє створювати кросбраузерні сторінки з підтримкою мобільних пристроїв. Упроваджено кольорову палітру, яка підсилює асоціації з безпекою та надійністю (зокрема, зелені відтінки для підтвердження успішних дій, червоні - для помилок).

Функціональні можливості інтерфейсу доповнюються JavaScript-кодом, що реалізує валідацію даних на стороні клієнта, інтерактивні елементи, а також динамічні повідомлення про стан процесів (наприклад, підтвердження надсилання

коду верифікації або помилка введення некоректного пароля). Також використано Fetch API для взаємодії з сервером без повного перезавантаження сторінки, що покращує швидкодію системи.

У випадку, якщо в процесі реєстрації виникають помилки, користувач отримує підказки, що потрібно вводити для успішної реєстрації. Після успішної реєстрації, користувач автоматично авторизується.

Якщо у користувача вже є створений аккаунт, він може авторизуватися за допомогою форми авторизації (рисунок 2.13).

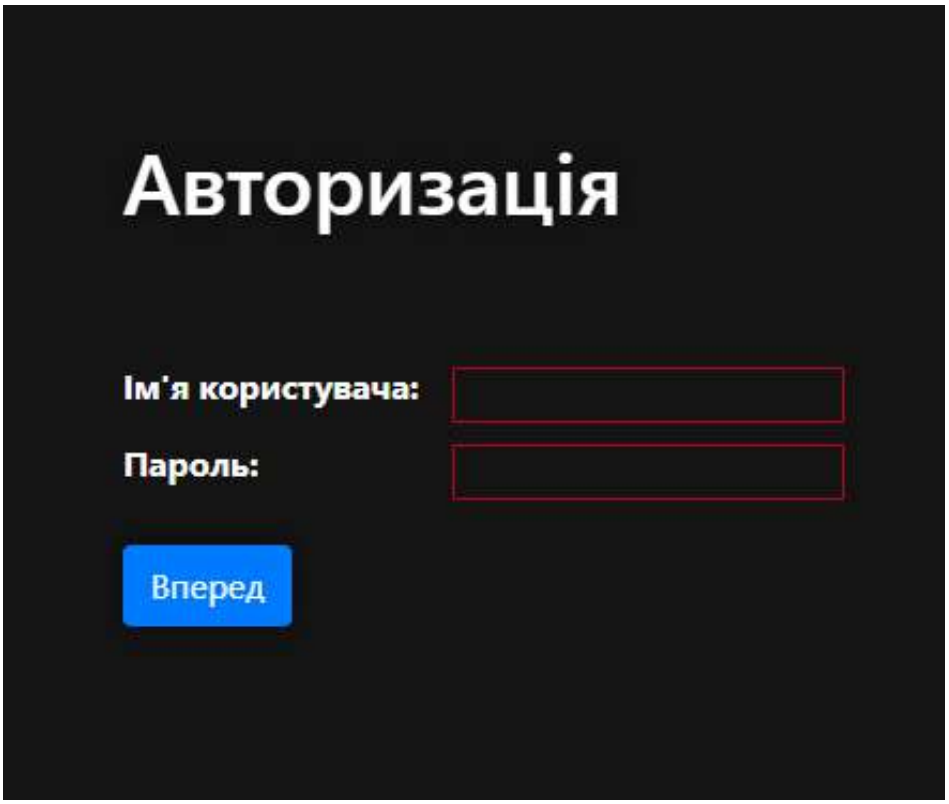

Рис. 2.13. Форма авторизації

Особливу увагу приділено розробці інтерфейсу для підтвердження електронної пошти та введення одноразового коду (2FA). Було забезпечено зручне введення коду, автоматичну обробку помилок та динамічне оновлення стану

безпеки. Для людей з особливими потребами враховано елементи доступності - контрастність, читабельність шрифтів і фокусна навігація.

У випадку, якщо двофакторна автентифікація увімкнена, користувач повинен ввести токен для підтвердження особистості (рисунок 2.14).

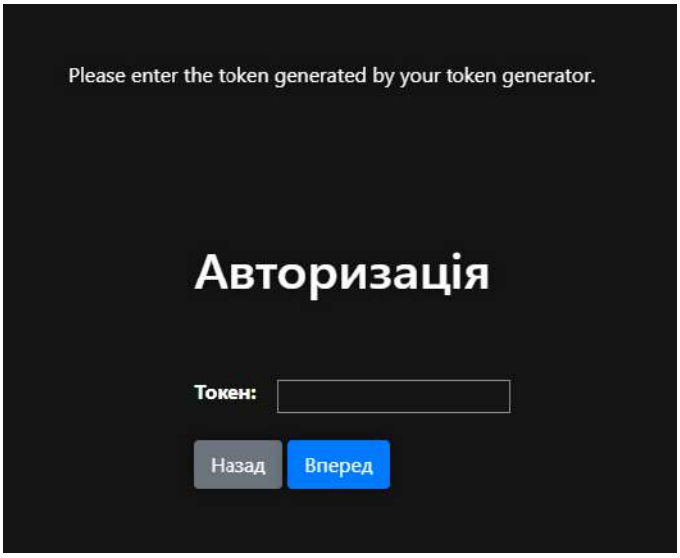

Рис. 2.14. Приклад аторизації користувача

Після успішної авторизації видно форму ввімкнення двофакторної авторизації. Натискаємо на кнопку та переходимо до ввімкнення цієї функції (рисунок 2.15).

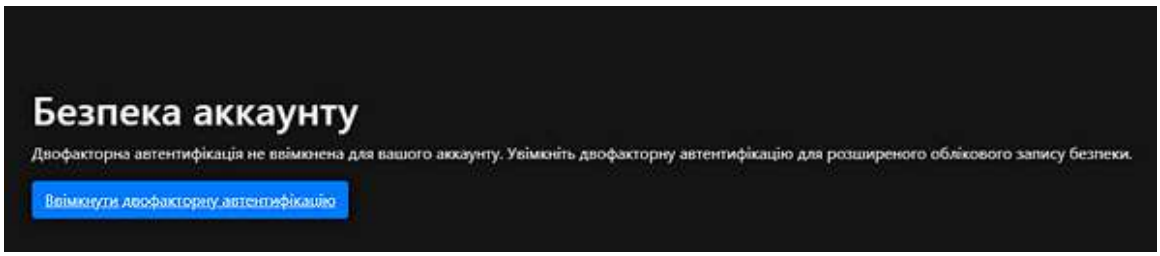

Рис. 2.15. Сторінка персональної інформації користувача

В наступному меню є додаткова інформація з можливістю відмінити дію (рисунок 2.17). Натискаємо кнопку «Вперед».

У користувачів є можливість ввімкнути двохфакторну автентифікацію 2 способами: через пошту або через генератор токенів (рисунок 2.16).

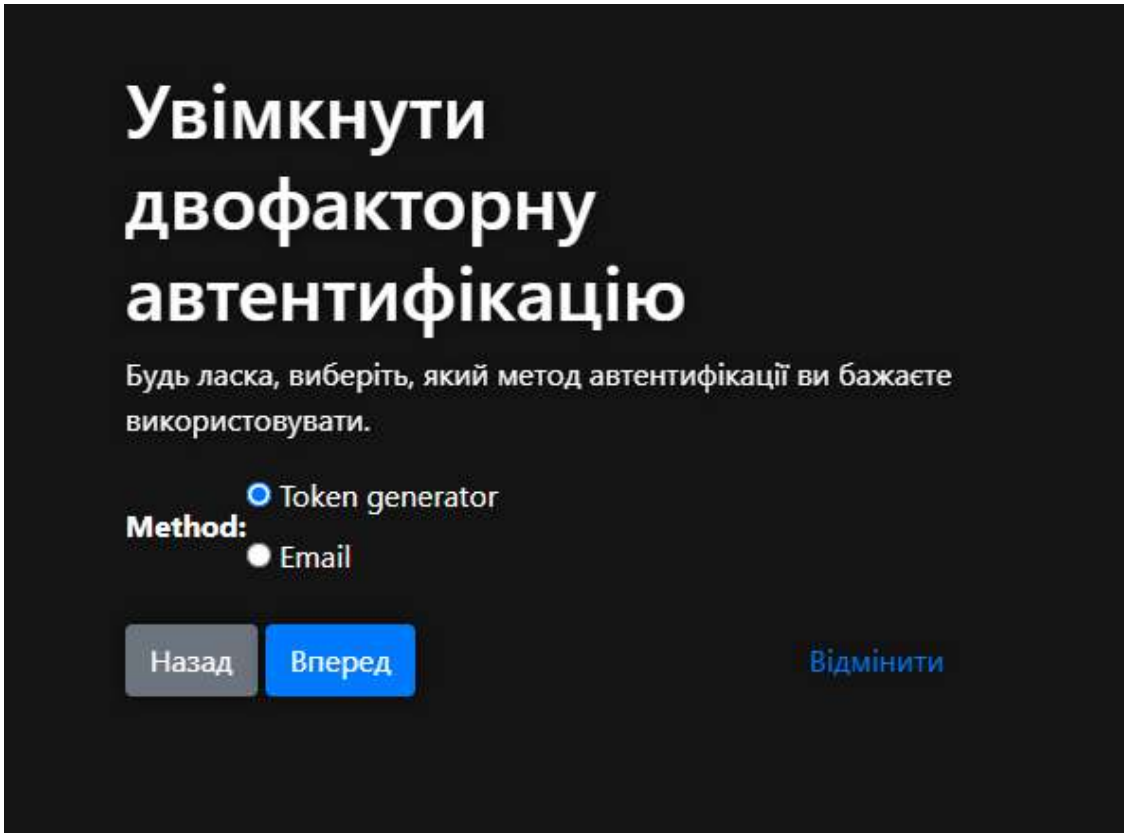

Рис. 2.16. Вибір способу автентифікації

Узагальнено, що правильне застосування HTML, CSS і JavaScript дозволяє створити ефективний фронтенд для інформаційної системи, який поєднує зручність користування, сучасний вигляд та відповідність принципам кібербезпеки.

Розберемо перший спосіб, через генератор токенів. В цьому меню описано все, що необхідно зробити для додання секретного коду до свого персонального генератора токенів і подальшого його використання для підвищення безпеки аккаунту.

Користувач має можливість відсканувати QR-код за допомогою смартфона або додати його через секретний код TOTP, якщо відсканувати не вдається.

Додаємо секретний код TOTP до генератора кодів і отримаємо часово-обмежений код, який в подальшому буде використаний для авторизації або інших дій, які потребують збільшеної безпеки аккаунта (рисунок 2.17).

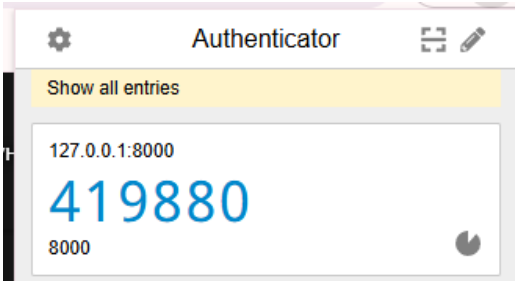

Рис. 2.17. Доданий секретний код TOTP до генератору кодів

Якщо все введено вірно, отримуємо повідомлення про успішно ввімкнену двофакторну автентифікацію (рисунок 2.18).

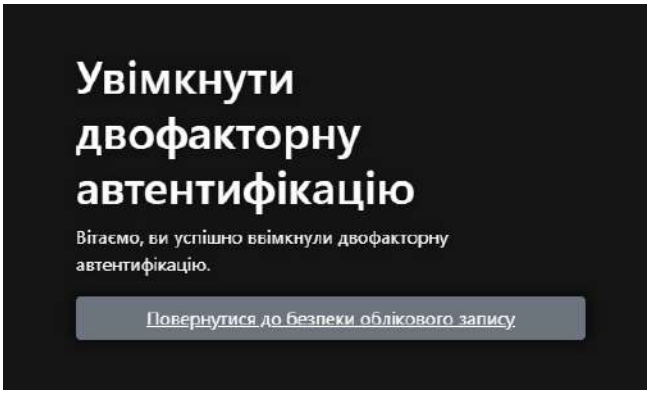

Рис. 2.18. Успішне увімкнення двофакторної автентифікації

Якщо ж обрати інший спосіб автентифікації (через пошту), то спосіб додання двофакторної автентифікації дуже схожий.

Обирай спосіб автентифікації через пошту в результаті чого отримуємо листа на пошту вказану при реєстрації аккаунту (рисунок 2.19).

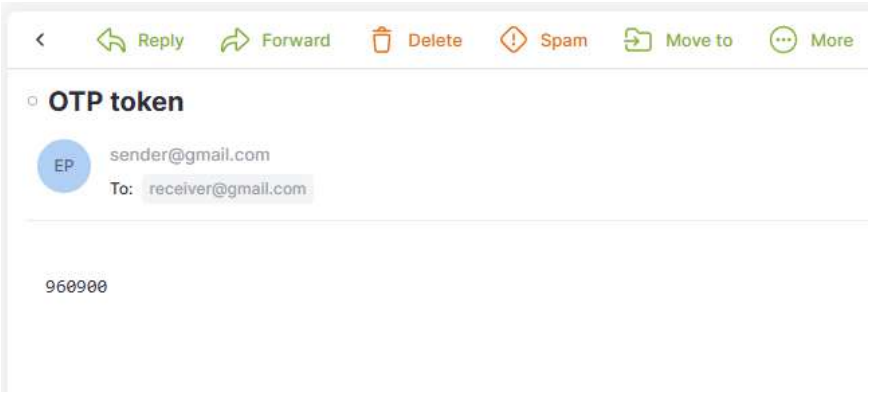

Рис. 2.19. Приклад отримання коду на пошту

Вводимо отриманий код в листі в поле вводу токену та натискаємо кнопку наступного етапу (рисунок 2.20). Якщо код було введено вірно, отримуємо повідомлення про успішно ввімкнену двухфакторну автентифікацію.

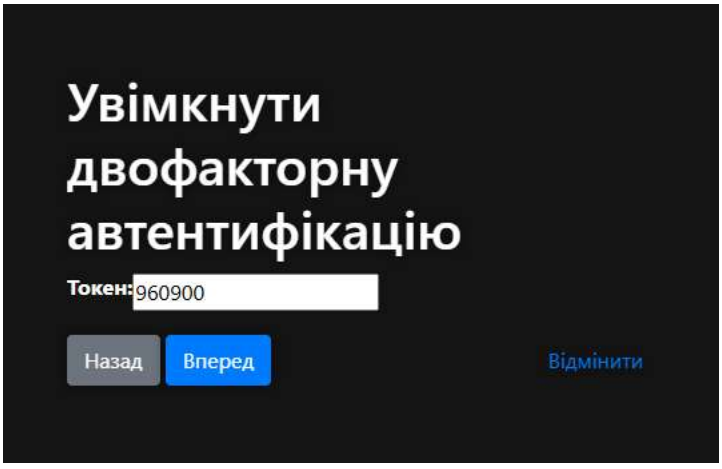

Рис. 2.20. Введення токену отриманого на пошту

Якщо введено правильний токен, система вмикає двофакторну автентифікацію для поточного облікового запису. У подальшому, під час кожної авторизації, після введення логіну та пароля користувач повинен буде підтверджувати вхід додатковим кодом, що надсилається на електронну пошту або генерується іншим способом, залежно від налаштувань. Це суттєво знижує ризики несанкціонованого доступу навіть у разі компрометації основного пароля.

Після активації 2FA користувачу стає доступна оновлена версія персонального кабінету з розділом керування параметрами безпеки. Тут відображається поточний статус захисту облікового запису, активні методи автентифікації та інші важливі параметри.

Авторизувавшись в аккаунт видно основні моменти, які стосуються безпеки (рисунок 2.21).

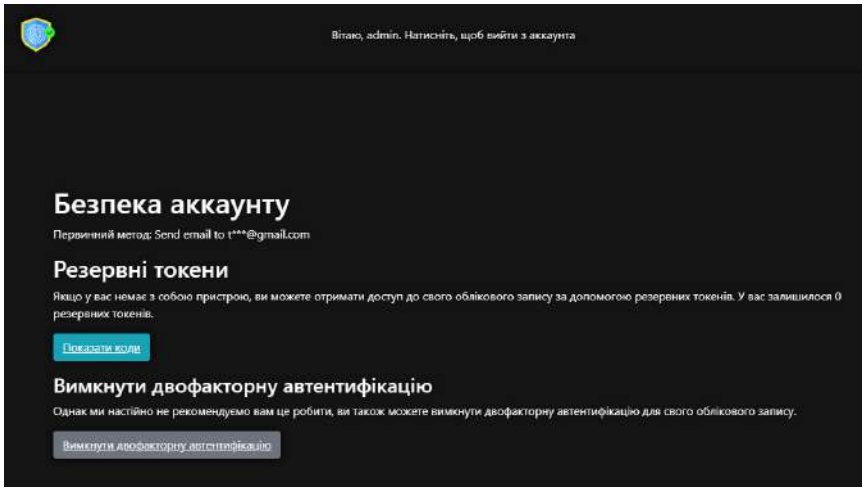

Рис. 2.21. Приклад кабінету користувача з параметрами

Резервні токени необхідні у випадках, коли немає можливості авторизуватися іншим способом. Для цього їх необхідно згенерувати через відповідну форму (рисунок 2.23).

Натискаємо кнопку та генеруємо резервні токени (рисунок 2.22). Кращим способом буде роздрукувати їх, адже збереження їх на комп’ютері не є безпечним методом.

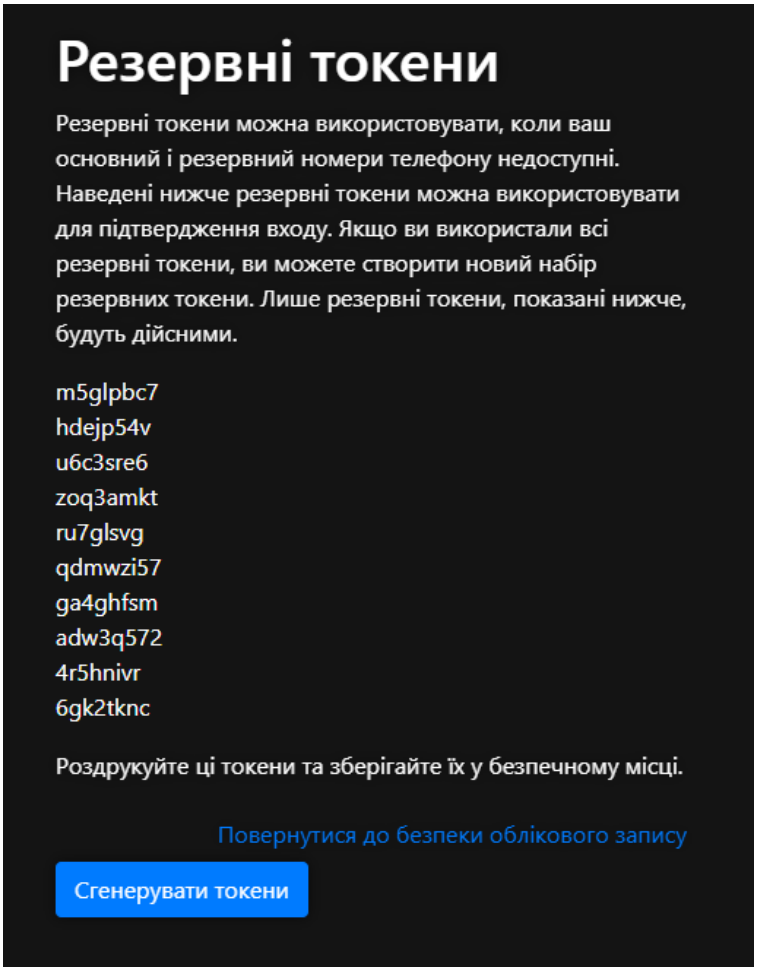

Рис. 2.22. Згенеровані резервні коди

Після генерації токенів вони відображаються у вигляді списку одноразових кодів. Кожен із них можна використати лише один раз. Інтерфейс чітко показує кількість доступних резервних кодів та дозволяє при необхідності згенерувати новий набір. Це забезпечує додатковий рівень контролю над обліковим записом і

дає змогу заздалегідь підготуватися до непередбачуваних ситуацій, пов'язаних із втратою доступу до основного методу автентифікації.

У випадку, якщо немає доступу до email або генератору кодів використаного для покращення безпеки аккаунту можливо використовувати один із згенерованих резервних токенів через відповідне меню (рисунок 2.23).

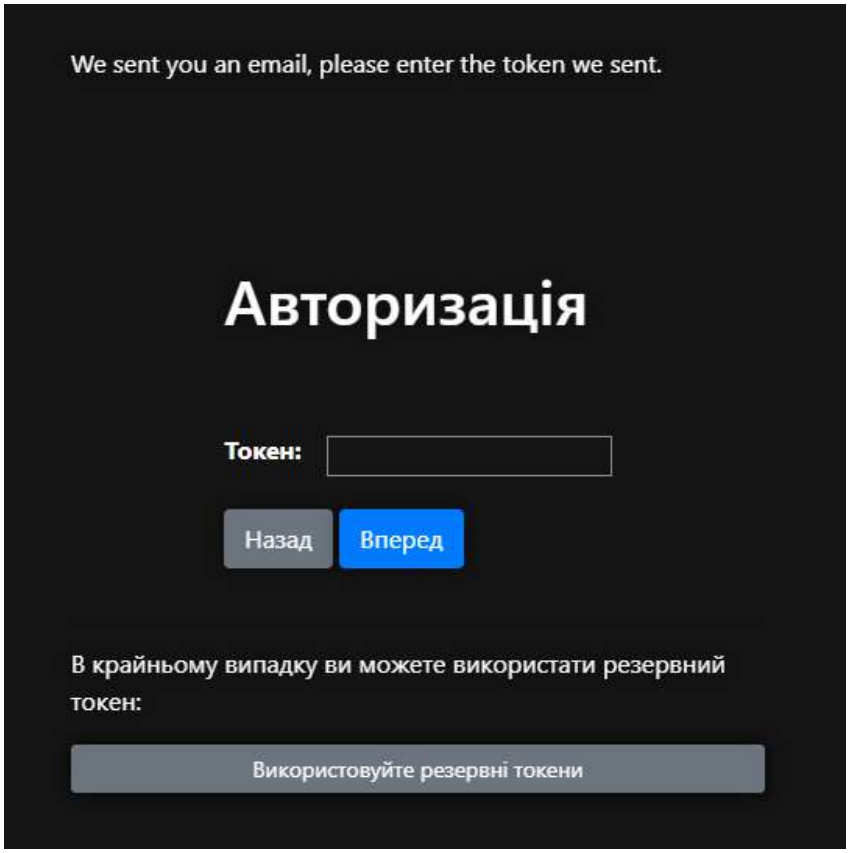

Рис. 2.23. Авторизація через резервні коди

Резервні токени зберігаються у зашифрованому вигляді та призначені для одноразового використання.

Система автоматично відмічає використані токени як неактивні, що дозволяє уникнути повторного застосування. Завдяки цьому забезпечується додатковий рівень контролю та прозорості у процесі автентифікації.

В будь-який момент є можливість згенерувати нові токени, та використовувати їх (рисунок 2.24).

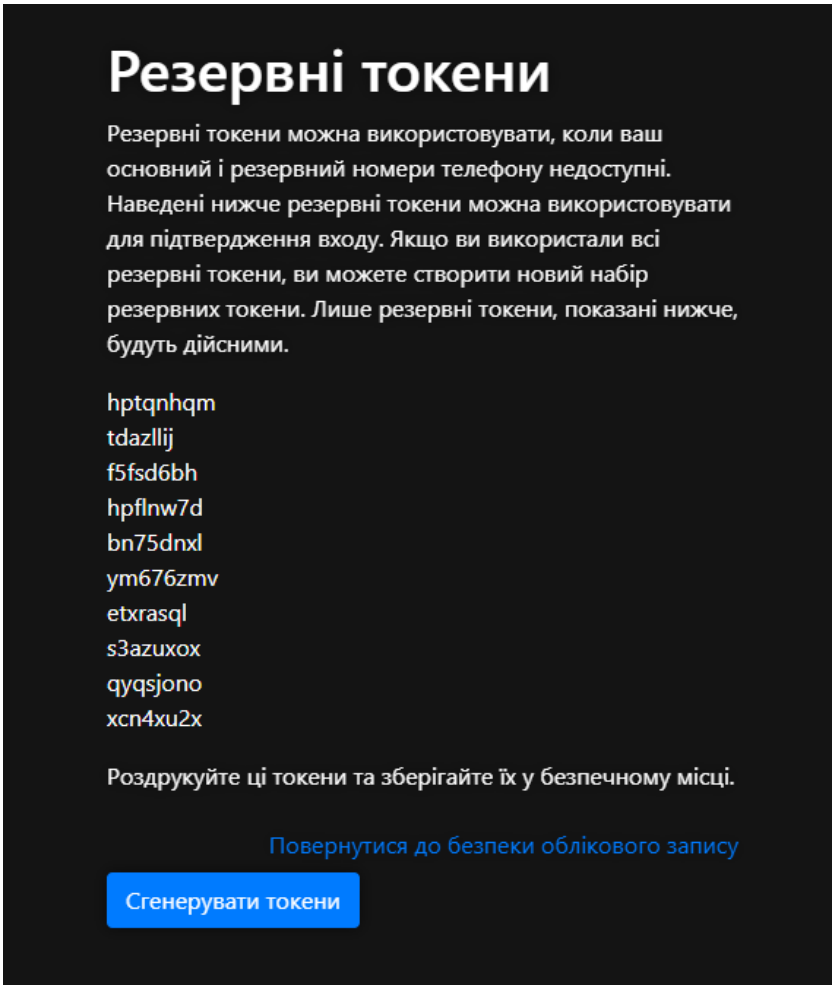

Рис. 2.24. Нові згенеровані токени

Користувач має можливість переглянути історію використання токенів, щоб переконатися у відсутності несанкціонованого доступу до облікового запису. Такий

підхід сприяє виявленню підозрілої активності та своєчасному вжиттю заходів безпеки.

Кожний користувач може вимкнути двохфакторну автентифікацію використовуючи форму. Для цього необхідно підтвердити свою дію та натиснути на кнопку «Вимкнути» (рисунок 2.25).

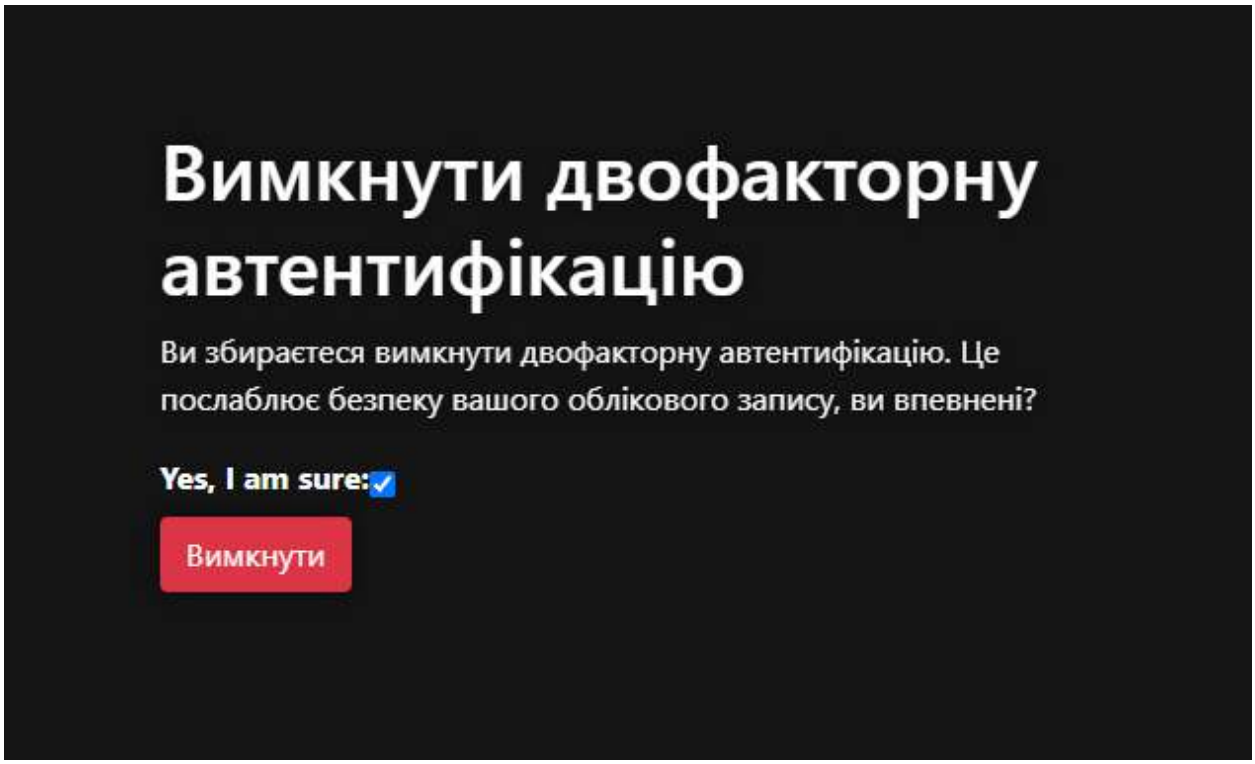

Рис. 2.25. Вимкнення двофакторної автентифікації

Отже розроблений веб-додаток дає змогу користувачам реєструватися через двофакторну автентифікацію.

Висновки по розділу 2

У даному розділі, базуючись на теоретичних засадах, визначених у розділі 1, здійснено комплексне проєктування архітектури веб-застосунку для вивчення

англійських слів. Створено функціональну модель, що відображає основні сценарії взаємодії користувачів із системою, від реєстрації до прогресу навчання. Розроблено структуру бази даних, яка забезпечує ефективне зберігання та управління словниковими одиницями, індивідуальними профілями користувачів, їхнім навчальним прогресом та результатами інтерактивних вправ.

Визначено ключові архітектурні рішення, що лягли в основу розробки. Сформовано структуру API, яка гарантує надійний та безпечний обмін даними між клієнтською та серверною частинами застосунку. При проєктуванні особливу увагу приділено принципам модульності та масштабованості, що дозволяє у майбутньому легко розширювати функціонал та інтегрувати нові навчальні модулі без суттєвих змін у базовій архітектурі.

Запропоновано та деталізовано технічні рішення для реалізації інтерфейсу користувача, що забезпечують його інтуїтивність, адаптивність до різних пристроїв та високу інтерактивність. Розроблено схеми взаємодії основних компонентів системи, що візуалізують логіку роботи та потоки даних. Це дозволило чітко окреслити завдання для подальшої програмної реалізації та мінімізувати потенційні ризики на етапі кодування.

Таким чином, у цьому розділі сформовано вичерпний проєкт веб-застосунку, який повністю відповідає визначеним у Розділі 1 функціональним та технологічним вимогам. Розроблена архітектура та деталізовані проєктні рішення є надійною основою для успішної програмної реалізації системи, здатної ефективно забезпечувати потреби у вивченні англійської лексики в цифровому середовищі.

РОЗДІЛ 3
ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ СИСТЕМИ РЕЄСТРАЦІЇ ТА АВТЕНТИФІКАЦІЇ КОРИСТУВАЧІВ

3.1. Методика тестування системи

У сучасному світі веб-розробки, де конкуренція є високою, швидкість створення веб-сайтів часто стає головним пріоритетом. Через це, етапам розробки прототипів та тестування інтерфейсів приділяється недостатньо уваги, що призводить до помилок, які можуть негативно вплинути на користувачів та їхнє бажання користуватися веб-сайтом. Нерідко IT-фахівці працюють над удосконаленням вже існуючих сайтів, виправляючи їхні недоліки. У UX-дизайні значні зусилля спрямовуються на створення інтуїтивно зрозумілих продуктів. Однак сьогодні спостерігається тенденція до акцентування на простоті використання, що зумовлено великою кількістю різноманітних онлайн-сервісів.

Тестування UI/UX є важливим процесом оцінки зручності, доступності та загального користувацького досвіду цифрового інтерфейсу (рисунок 3.1), спрямованим на виявлення та усунення проблем, що можуть вплинути на задоволеність та залученість користувачів [34].

Для оцінки юзабіліті вже готових додатків застосовуються якісні методи дослідження, такі як аналітика, опитування, теплові карти, а також принципи взаємодії людини та комп'ютера (HCI), включаючи закони Фітса та Хіка, модель KLM [35, 36, 37].

З іншого боку, для забезпечення високої якості будь-якого програмного продукту необхідне його тестування.

Одним з найпоширеніших видів є функціональне тестування, яке перевіряє відповідність продукту бізнес-вимогам, тобто коректне виконання ним своїх основних функцій.

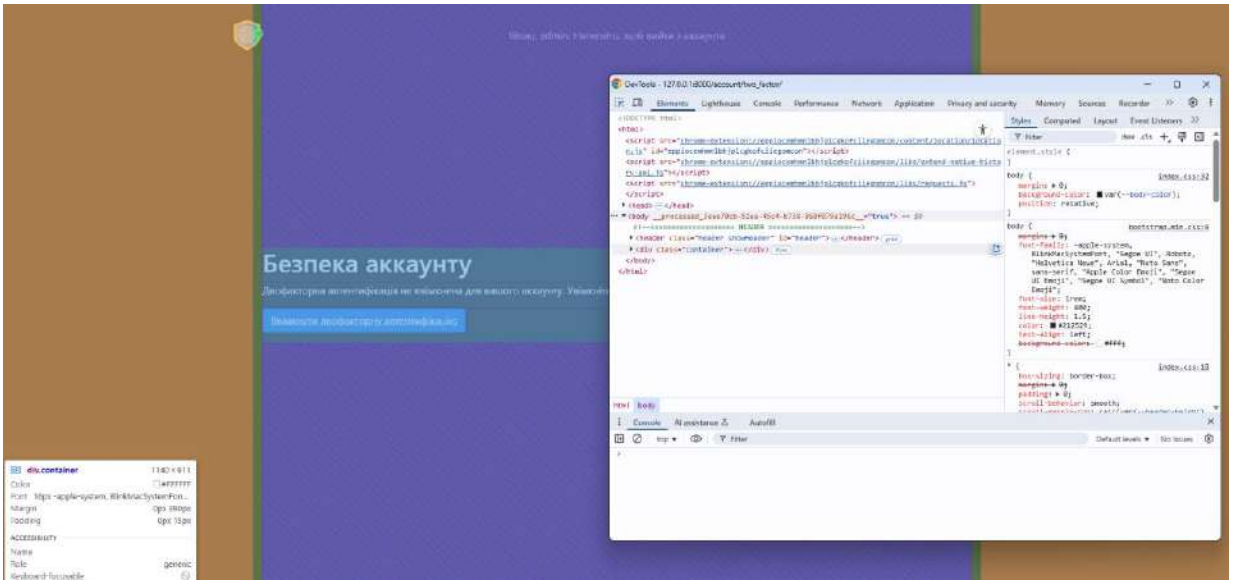

Рис. 3.1. Тестування стилів за допомогою Chrome Dev Tools

Тестування часто включає метод "чорної скриньки", який не зосереджується на внутрішній структурі коду, а використовує такі техніки, як аналіз граничних значень, тестування юзкейсів та таблиці прийняття рішень. Важливим також є негативне тестування, що перевіряє поведінку системи при некоректних діях користувача. Користувачі звертають велику увагу на швидкість роботи додатку та його адаптивність до різних типів пристроїв. Повільне завантаження, переривання в роботі та проблеми з навігацією на мобільних пристроях значно знижують ймовірність подальшого використання додатку [33, 36].

Враховуючи вищезазначене, розробка методики, що покращить процес тестування інтерфейсів сайтів, є актуальною. Запропонована методика поєднуватиме методи функціонального та юзабіліті тестування. Для досягнення цієї мети необхідно: обрати методи юзабіліті тестування для оцінки простоти

використання сайту; обрати методи функціонального тестування для оцінки якості готового сайту; використати метод аналізу ієрархій для вибору найкращих альтернатив серед існуючих методів. Особливістю цієї методики є пріоритетність юзабіліті тестування на початковому етапі з використанням законів Фітса та Хіка, моделі KLM, опитувань та аналізу теплових карт. Зручний візуальний дизайн та елементи інтерфейсу допомагають користувачам досягати своїх цілей.

Після цього доцільно проводити функціональне тестування для виявлення основних функціональних недоліків, використовуючи тестування "чорної скриньки", тестування переходів станів, складання таблиць прийняття рішень, а також аналіз продуктивності анімацій за допомогою Chrome Dev Tools Performance (рисунок 3.2) [34, 35, 36].

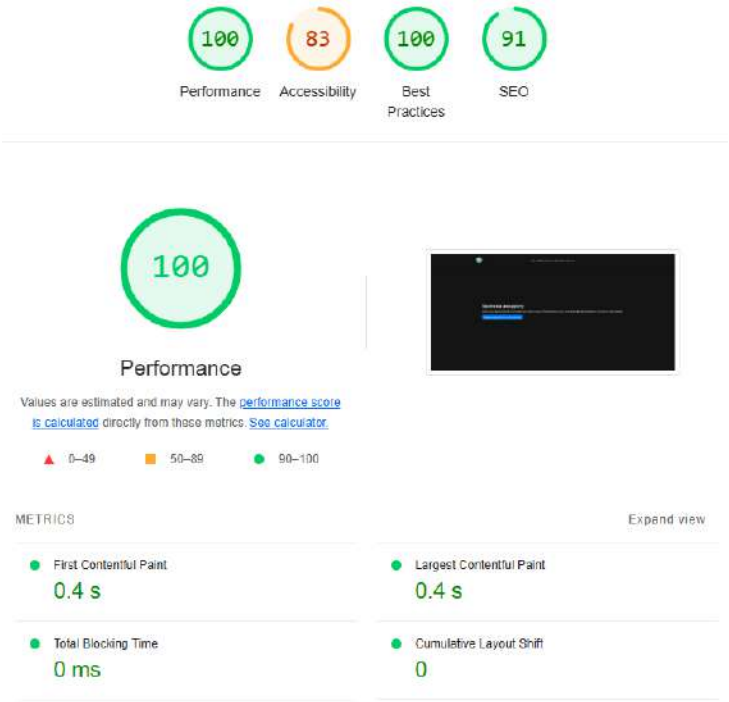

Рис. 3.2. Тестування продуктивності за допомогою Chrome Dev Tools

Запропонована методика об'єднує переваги юзабіліті та функціонального тестування, що дозволяє виявити більше помилок та покращити загальне враження

від користування веб-додатком. Варто зазначити, що для застосування цієї методики фахівцю необхідні базові знання з юзабіліті та тестування програмного забезпечення.

Хоча тестування юзабіліті та функціональне тестування часто розглядаються окремо, для ефективної оцінки готового веб-додатку краще використовувати їх комбінацію. Такий підхід дозволяє виявити як критичні помилки в дизайні, що негативно впливають на користувацький досвід, так і переконатися в тому, що додаток коректно виконує свої функції.

Методика також допомагає виявити основні труднощі користувачів при виконанні типових сценаріїв використання, сформувати гіпотези щодо точок зростання та можливостей для подальшого вдосконалення. Ця методика може бути корисною для IT-спеціалістів різного профілю, включаючи UI/UX дизайнерів, розробників та тестувальників програмного забезпечення [34].

3.2. Перевірка на типові загрози безпеки

Фреймворк Django надає розробникам набір інструментів для реалізації надійної автентифікації та авторизації користувачів.

В основі цього лежить механізм сеансів, що дозволяє верифікувати облікові дані користувачів та визначати їхні права доступу до різних функцій системи.

Сеанси в Django використовуються для відстеження стану взаємодії між веб-сайтом та конкретним браузером (рисунок 3.3 - 3.5), забезпечуючи можливість зберігання даних для кожного браузера та їх отримання при кожному наступному запиті.

Ідентифікація сеансу здійснюється за допомогою файлів cookie, що містять унікальні ідентифікатори. За замовчуванням, дані сеансів зберігаються у базі даних

застосунку, що вважається більш безпечним підходом порівняно зі зберіганням безпосередньо у файлах cookie [39].

```
class HomePageView(TemplateView):  1 usage
    async def get(self, request, *args, **kwargs):
        is_authenticated = await sync_to_async(lambda: request.user.is_authenticated)()

        context = {
            'is_authenticated': is_authenticated,
        }
```

Рис. 3.3. Перевірка на авторизацію користувача

Для забезпечення безпеки паролів користувачів Django автоматично застосовує механізм їхнього хешування з використанням алгоритму PBKDF2 та хеш-функції SHA256 (рисунок 3.6).

```
form = UserSignUpForm(POST)

if await sync_to_async(form.is_valid)():
    email = form.cleaned_data['email']
    if await User.objects.filter(email=email).afirst():
        messages.add_message(request, messages.ERROR, message: 'Користувача з такою поштою вже зареєстровано')

        return redirect('/account/register/')
```

Рис. 3.4. Перевірка правильно введених даних у формі

Такий підхід унеможливлює відновлення оригінальних паролів навіть у випадку отримання доступу до бази даних.

Процес хешування включає генерацію випадкової "солі" для кожного пароля, її змішування з паролем та подальше застосування алгоритму PBKDF2 з визначеною кількістю ітерацій для створення хешу, який і зберігається в базі даних.

Окрім механізмів автентифікації та авторизації, Django пропонує вбудовані засоби захисту від поширених веб-атак та надає можливості для розширення функцій безпеки. Зокрема, фреймворк забезпечує автоматичний захист від SQL-ін'єкцій завдяки використанню ORM, який параметризує SQL-запити, унеможливлюючи виконання шкідливого коду через введені користувачем дані.

```
for error_message in form.errors.values():
    messages.add_message(request, messages.ERROR, error_message)
```

Рис. 3.5. Відображення помилок форм

При спробі автентифікації введений пароль знову хешується, і отриманий хеш порівнюється зі збереженим у базі даних.

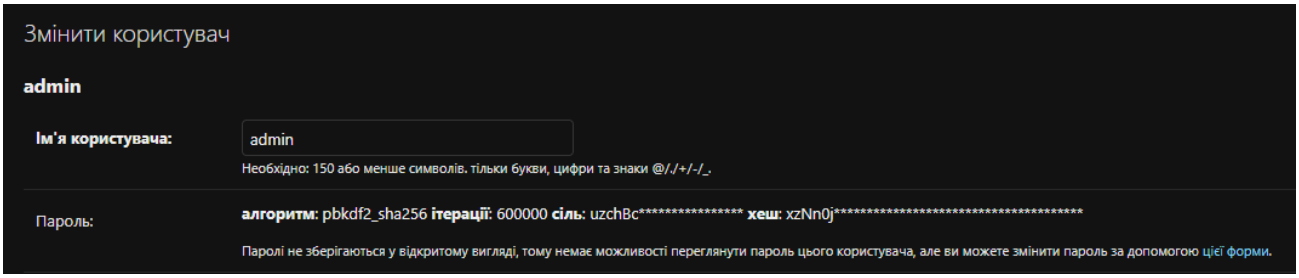

Рис. 3.6. Використання хешування паролю

Також реалізовано захист від міжсайтового скриптингу (XSS) шляхом автоматичного екранування даних, що виводяться в HTML-шаблонах, запобігаючи виконанню сторонніх скриптів у браузері користувача. Для захисту від міжсайтової підробки запитів (CSRF) Django надає спеціальний тег шаблону {% csrf_token %} (рисунок 3.7), використання якого при POST-запитах генерує унікальний ключ користувача, що запобігає відправці підроблених форм [40].

Для підвищення загального рівня безпеки веб-застосунку Django рекомендує використання протоколу HTTPS, який забезпечує шифрування трафіку між клієнтом та сервером.

```
27      <body>
28          {% csrf_token %}
29          <!--===================== HEADER ======================-->
30          {% include 'app/header.html' %}
31
32          <!--====================== MAIN =======================-->
33          <main class="main">
34              {% include 'app/hero.html' %}
35          </main>
36
37          <!--================ SCRIPTS ================-->
38
39          {% bootstrap_javascript %}
40      </body>
41  </html>
```

Рис. 3.7. Використання тегу csfr_token

Для належного налаштування HTTPS пропонується використовувати такі параметри, як перенаправлення HTTP на HTTPS (SECURE_SSL_REDIRECT), HTTP Strict Transport Security (HSTS) для примусового використання HTTPS браузерами, а також налаштування безпечних cookie-файлів (SESSION_COOKIE_SECURE та CSRF_COOKIE_SECURE). Крім того, Django має вбудований захист від клікджекінгу, запобігаючи відображенню веб-сайтів у фреймах сторонніх ресурсів за допомогою проміжного програмного забезпечення X-Frame_Options middleware, яке можна налаштувати відповідно до потреб безпеки веб-застосунку.

3.3. Оцінка зручності використання та аналіз результатів тестування

Стандарт [41] виступає інтегруючим документом, що об'єднує різноманітні атрибути якості програмного забезпечення, які були визначені в різних існуючих стандартах. Завдяки цьому, він пропонує цілісну та узагальнену модель для оцінки якості програмних застосунків. Ця модель, візуально представлена на рисунку 3.8, є ключовим інструментом для розуміння та вимірювання якості ПЗ.

Згідно з цією моделлю, якість програмного забезпечення визначається шістьма фундаментальними характеристиками. До них належать: функційна придатність, яка відображає здатність ПЗ задовольняти потреби користувачів; продуктивність, що характеризує ефективність використання ресурсів; надійність, яка визначає стабільність роботи ПЗ; зручність використання, що оцінює легкість взаємодії користувача з системою; супроводжуваність, яка відображає простоту внесення змін та виправлень; і переносність, що характеризує можливість використання ПЗ в різних середовищах.

Важливим аспектом цієї моделі є те, що досягнення найвищих значень за кожною з цих шести основних характеристик безпосередньо сприяє підвищенню загального рівня якості програмного забезпечення. Іншими словами, всебічна оптимізація кожної з цих складових є необхідною умовою для створення високоякісного програмного продукту.

У практичному застосуванні, процес оцінювання характеристик якості ПЗ, відповідно до цього стандарту, здійснюється шляхом аналізу конкретних та вимірюваних показників. На першому етапі визначаються релевантні метрики, які дозволяють кількісно оцінити кожну з шести характеристик якості.

Далі розробляється детальна шкала оцінювання для кожної обраної метрики. Ця шкала відображає різні рівні відповідності значення показника встановленим

вимогам або очікуванням. Таким чином, стає можливим об'єктивно визначити ступінь, в якому програмне забезпечення відповідає критеріям якості.

На завершальному етапі, сукупність усіх цих "вимірюваних" показників, разом із розробленими шкалами оцінювання, формує комплексний критерій, на основі якого здійснюється загальна оцінка якості розглянутого програмного забезпечення. Такий підхід забезпечує структурованість та об'єктивність процесу оцінки [42, 43].

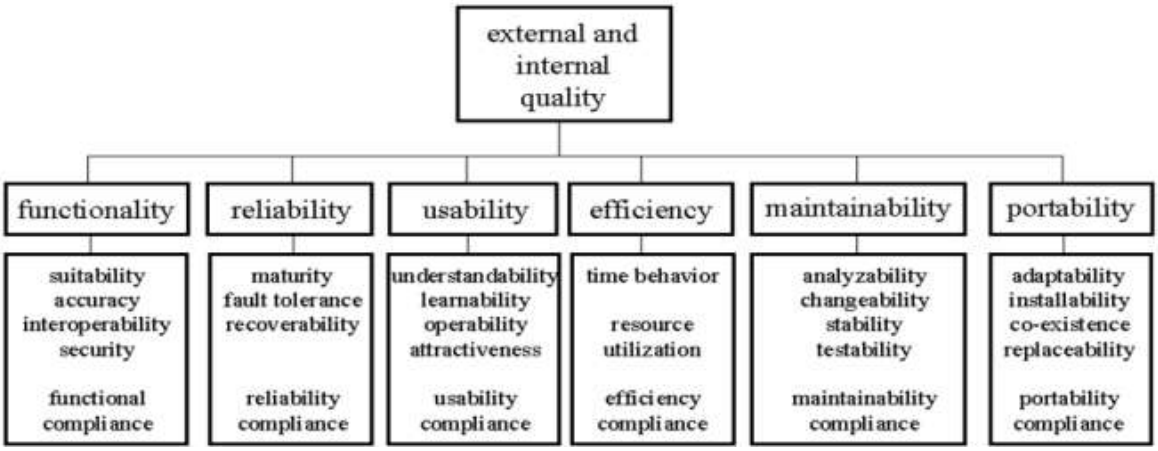

Рис. 3.8. Модель якості програмного забезпечення

Для досягнення максимального рівня задоволеності широкого кола користувачів, забезпечення конкурентоздатності програмного забезпечення на ринку та стимулювання його подальшого розвитку, необхідно враховувати весь спектр характеристик якості. Таким чином, якість програмного забезпечення є багатогранним поняттям, що описується численними різноманітними атрибутами. До ключових характеристик програмного забезпечення належать:

1. Функціональність (Functionality): визначає здатність програмного забезпечення вирішувати поставлені завдання, що відповідають зафіксованим та передбачуваним потребам користувача в заданих умовах експлуатації. Ця характеристика охоплює такі аспекти, як коректність та точність виконання

функцій, функціональна сумісність, відповідність галузевим стандартам та захищеність від несанкціонованого доступу.

2. Надійність (Reliability): характеризує здатність програмного забезпечення виконувати необхідні функції у визначених умовах протягом заданого періоду часу або певної кількості операцій. Атрибутами надійності є завершеність та цілісність системи, здатність до самостійного та коректного відновлення після збоїв, а також відмовостійкість.

3. Зручність використання (Usability): відображає ступінь легкості розуміння, навчання, використання та привабливості програмного забезпечення для користувачів.

4. Ефективність (Efficiency): визначає здатність програмного забезпечення забезпечувати необхідний рівень продуктивності при заданих обмеженнях ресурсів, часу та інших визначених умов.

5. Зручність супроводу (Maintainability): характеризує легкість, з якою програмне забезпечення може бути проаналізоване, протестоване та модифіковане з метою виправлення дефектів, реалізації нових вимог, полегшення подальшого обслуговування та адаптації до змін зовнішнього середовища.

6. Портативність (Portability): відображає здатність програмного забезпечення бути легко перенесеним з одного операційного середовища в інше.

Проведене комплексне тестування веб-застосунку Django з інтегрованою двофакторною автентифікацією (2FA) було проаналізовано з урахуванням визначених критеріїв якості програмного забезпечення. Оцінка функціональності підтвердила коректність реалізації процесу 2FA, включаючи успішну первинну автентифікацію, валідну генерацію та верифікацію другого фактору різними методами, а також належну обробку помилкових сценаріїв та інтеграцію з основною функціональністю застосунку. Аналіз надійності засвідчив стабільну

роботу механізму 2FA протягом тривалих сеансів тестування без виявлення критичних збоїв.

Оцінка зручності використання показала, що процес налаштування 2FA є інтуїтивно зрозумілим, а щоденне використання, хоча й додає додатковий крок автентифікації, сприймається користувачами як прийнятний захід безпеки, проте розглядається можливість оптимізації шляхом реалізації опції запам'ятовування пристроїв. Аналіз ефективності виявив незначний вплив 2FA на загальний час автентифікації та відсутність значного збільшення споживання системних ресурсів. З точки зору зручності супроводу, інтеграція бібліотек Django для 2FA була відносно простою, архітектура передбачає можливість розширення, а система забезпечує достатнє логування відповідних подій. У контексті портативності, використання стандартних інструментів Django сприяє загальній переносимості застосунку.

Результати тестування свідчать про успішне впровадження двофакторної автентифікації, що значно підвищує рівень безпеки веб-застосунку Django, зберігаючи при цьому прийнятний рівень зручності використання та продуктивності. Запропонована інтеграція 2FA відповідає сучасним вимогам до захисту користувацьких даних, а подальші кроки можуть бути спрямовані на додаткову оптимізацію користувацького досвіду.

3.4. Порівняння з аналогами

Існує значна різноманітність методів автентифікації, і для проведення порівняльного аналізу їх доцільно об'єднати в декілька основних груп. Важливо розуміти, що в межах кожної групи можуть існувати різні реалізації, які відрізняються специфічними характеристиками, перевагами та недоліками. Однак,

загальні тенденції та ключові показники для кожної групи методів, як правило, залишаються сталими.

Найпоширенішим та найпростішим методом є парольна автентифікація, яка широко використовується в соціальних мережах, платіжних системах та на веб-ресурсах, що оперують персональними даними. Під паролем розуміється спеціальна кодова фраза або набір символів, унікальний для кожного ресурсу. Подальшим розвитком цього підходу стали графічні паролі, що базуються на введенні певної нетекстової послідовності, перевагою яких є потенційне спрощення запам'ятовування.

Наступною групою є методи, що використовують одноразові та динамічні паролі [44], які вимагають від користувача додаткових дій, але підвищують рівень захисту від атак, заснованих на повторному використанні паролів. Також часто застосовуються методи автентифікації через сторонні ресурси або децентралізованої автентифікації, такі як OpenID [45] та OAuth [46]. Окрему категорію складають методи з використанням токенів, як програмних, так і апаратних. Багатофакторна автентифікація також виділяється в окрему групу, прикладом якої є підтвердження коду через SMS-повідомлення.

Криптографічні методи автентифікації формують ще одну категорію, що включає різноманітні підходи, від використання сертифікатів до стенографічних методів [47]. До останньої категорії відносяться методи біометричної автентифікації [48] на веб-ресурсах, наприклад, з використанням голосового підтвердження або аналізу особливостей введення користувача. Порівняння цих методів проводиться за трьома основними групами характеристик: зручність використання, складність реалізації та безпека рішення.

У наведених таблицях (3.1 - 3.3) оцінка варіюється від 1 (найгірший показник) до 3 (найкращий).

Таблиця 3.1

Порівняння методів автентифікації за зручністю користування

	Зручність використання						
	Запамя-товування	**Доп. пристрій**	**Викон. дій**	**Легкість**	**Час**	**Помилки**	**Відновлення**
Пароль	1	3	2	3	3	2	3
Сторонній ресурс	2	3	3	3	3	3	2
Графічні	1	1	2	3	3	2	3
Динамічні	1	3	2	2	3	2	2
Токени	3	1	1	1	2	3	1
Багатофакторна	1	1	1	3	2	2	1
Криптографія	3	1	1	1	1	2	1
Біометрія	3	3	2	3	2	2	1

Примітка. Джерело: розроблено автором

Аналіз порівняльних характеристик різних методів автентифікації чітко демонструє, що простота реалізації часто супроводжується компромісами в інших важливих аспектах, таких як безпека та зручність використання. Зокрема, найбільш елементарний підхід - використання простих паролів - вирізняється своєю легкістю в імплементації на стороні розробника. Для впровадження парольної автентифікації зазвичай не потрібні складні протоколи чи додаткові сервіси, що спрощує процес розробки.

Однак, ця простота має свою ціну. З точки зору безпеки, прості паролі є найбільш вразливими до різноманітних атак, включаючи перебір, фішинг та соціальну інженерію. Відсутність додаткових механізмів захисту робить їх легкою ціллю для зловмисників. Крім того, з боку користувача, парольна автентифікація вимагає постійного запам'ятовування унікальних та складних комбінацій для кожного окремого сервісу, що може бути незручним та призводити до використання слабких або повторюваних паролів.

На противагу цьому, методи автентифікації через сторонні ресурси, такі як OAuth або OpenID, пропонують значно вищий рівень зручності для кінцевих користувачів. Завдяки можливості використання вже існуючих облікових даних від надійних провайдерів, користувачам не потрібно створювати та запам'ятовувати нові логіни та паролі для кожного веб-сайту. Це спрощує процес реєстрації та входу, підвищуючи загальний комфорт користування.

Таблиця 3.2

Порівняння методів аутентифікації за простотою реалізації

	Реалізація				
	Доступність	**Вартість**	**Серверна середовище**	**Клієнтське середовище**	**Пропрієтарність**
Пароль	3	3	3	3	3
Сторонній ресурс	3	3	1	3	3
Графічні	1	3	1	3	3
Динамічні	2	3	2	2	3
Токени	1	1	1	2	1
Багатофакторна	2	2	2	2	2
Криптографія	1	1	1	2	1
Біометрія	1	1	1	1	1

Примітка. Джерело: розроблено автором

Проте, зручність автентифікації через сторонні сервіси тісно пов'язана з безпекою провайдера цих послуг. У випадку компрометації облікового запису користувача на сторонньому ресурсі, зловмисники можуть отримати доступ до всіх веб-сайтів, де використовується цей метод автентифікації. Крім того, для інтеграції такого методу на стороні веб-застосунку необхідна певна конфігурація сервера для налагодження взаємодії з обраним провайдером автентифікації.

Графічні та динамічні паролі забезпечують незначне підвищення безпеки, але ускладнюють використання та підвищують вимоги до клієнтської та серверної частин. Методи з використанням токенів є відносно безпечними, але часто пропрієтарними, платними та потребують спеціальних налаштувань. Двофакторна

автентифікація підвищує безпеку, але знижує зручність та ускладнює серверні рішення. Криптографічні та біометричні методи є найбільш захищеними, але менш зручними та складнішими в реалізації.

Таблиця 3.3

Порівняння методів автентифікації з безпеки даних, що передаються

	Безпека				
	Перебір	**Спостереження**	**Непрямий злом**	**Фішинг**	**Крадіжка**
Пароль	1	1	1	1	3
Сторонній ресурс	2	2	3	3	3
Графічні	1	1	2	2	3
Динамічні	2	3	2	2	3
Токени	3	3	3	3	2
Багатофакторна	1	1	3	3	2
Криптографія	3	3	3	3	3
Біометрія	3	3	1	1	3

Примітка. Джерело: розроблено автором

Аналіз різноманітних методів автентифікації чітко ілюструє відсутність єдиного, бездоганного рішення, яке б одночасно забезпечувало максимальну безпеку, найвищу зручність використання та мінімальну складність реалізації. Натомість, спостерігається закономірність, коли оптимізація одного з цих ключових аспектів неминуче тягне за собою погіршення інших. Це створює необхідність компромісів та зваженого підходу при виборі оптимального методу для конкретного веб-ресурсу.

Рішення щодо вибору методу автентифікації не може бути універсальним, а має базуватися на ретельному аналізі специфічних потреб власників ресурсу. До уваги слід брати потенційні ризики та загрози, а також цінність інформації, яка підлягає захисту. Наприклад, для веб-сайту з низькими вимогами до безпеки пріоритет може бути надано зручності використання, тоді як для ресурсу, що оперує

конфіденційними даними, на першому місці опиниться надійність механізму автентифікації.

Серед розглянутих методів, лідерство за критерієм зручності використання беззаперечно належить методам автентифікації через треті сторони, таким як OAuth та OpenID. Їх широке поширення в Інтернеті зумовлене можливістю для користувачів здійснювати вхід на різні веб-сайти, використовуючи вже існуючі облікові дані від надійних провайдерів. Це значно спрощує процес аутентифікації та зменшує необхідність запам'ятовування численних логінів та паролів.

Однак, незважаючи на високу зручність, рівень безпеки методів автентифікації через треті сторони є радше середнім. Безпека користувацького облікового запису фактично залежить від надійності та захищеності стороннього провайдера. У випадку компрометації облікових даних на стороні провайдера, зловмисники можуть отримати доступ до всіх пов'язаних веб-ресурсів.

Парольна автентифікація, яка є найпростішою з точки зору реалізації. Для її впровадження не потрібні складні протоколи чи інтеграція з зовнішніми сервісами. Проте, ця простота досягається за рахунок значного зниження рівня безпеки, оскільки прості паролі є вразливими до багатьох видів атак. Крім того, як вже зазначалося, парольна автентифікація не завжди є зручною для кінцевих користувачів, вимагаючи від них запам'ятовування великої кількості складних паролів.

Для веб-інфраструктур, де безпека є першочерговим пріоритетом, таких як банківські системи, державні портали або сервіси, що обробляють особливо чутливі дані, найбільш обґрунтованим підходом до автентифікації є використання криптографічних або біометричних методів. Ці методи, завдяки своїй природі, забезпечують значно вищий рівень захисту порівняно з традиційними паролями або навіть автентифікацією через треті сторони. Криптографічні методи автентифікації використовують складні математичні алгоритми та криптографічні ключі для

підтвердження особи користувача. Прикладами можуть слугувати автентифікація на основі цифрових сертифікатів або використання криптографічних токенів. Ці методи відрізняються високою стійкістю до різноманітних атак, оскільки для їхнього успішного обходу зловмисникам необхідно подолати криптографічний захист.

Біометричні методи автентифікації, що базуються на унікальних біологічних характеристиках користувача, таких як відбитки пальців, розпізнавання обличчя або сканування райдужної оболонки ока, також демонструють високий потенціал у забезпеченні безпеки. Вони усувають необхідність запам'ятовування паролів і роблять автентифікацію більш персоналізованою та складною для підробки.

Однак, на сьогоднішній день біометричні методи ще не набули достатнього розвитку для масового та безпроблемного застосування на всіх типах веб-ресурсів. Існують певні обмеження, пов'язані з необхідністю наявності спеціалізованого обладнання на стороні користувача, а також питання щодо приватності та точності розпізнавання в різних умовах.

З огляду на це, подальший розвиток методів автентифікації, ймовірно, буде спрямований на комбіновані системи. Цей підхід передбачає поєднання декількох методів автентифікації для підвищення загального рівня безпеки та зручності. Наприклад, використання двофакторної або багатофакторної автентифікації, де поряд з паролем застосовується одноразовий код, біометричні дані або криптографічний токен.

Особливо перспективним напрямком є використання криптографічних підходів для розширення можливостей існуючих методів автентифікації. Криптографія може бути застосована для посилення захисту паролів, забезпечення безпечної передачі даних при автентифікації через треті сторони, а також для створення більш надійних та приватних біометричних систем. Таким чином, комбінування різних методів з акцентом на криптографічний захист відкриває нові

можливості для створення безпечних та зручних систем автентифікації для критично важливих веб-інфраструктур.

3.5. Можливості подальшого розвитку

Аналіз існуючих методів автентифікації та оцінка впровадження двофакторної автентифікації в веб-застосунку Django вказують на кілька перспективних напрямків подальшого розвитку. У сфері методів автентифікації спостерігається тенденція до комбінування різних підходів для досягнення оптимального балансу між безпекою, зручністю використання та складністю реалізації. Особливий інтерес представляє інтеграція криптографічних методів, зокрема електронного цифрового підпису, для підвищення рівня захисту критично важливих веб-інфраструктур. Подальші дослідження можуть бути спрямовані на розробку нових комбінованих схем автентифікації, що використовують сильні сторони різних методів, мінімізуючи їхні недоліки.

У контексті тестування веб-застосунків з використанням 2FA, подальший розвиток може полягати в удосконаленні методик оцінки зручності використання, наприклад, шляхом проведення більш широких користувацьких тестувань та аналізу поведінкових факторів. Також актуальним є дослідження методів автоматизованого тестування безпеки механізмів 2FA на стійкість до різноманітних атак. Розвиток може також включати інтеграцію додаткових методів другого фактору, таких як біометрична автентифікація, за умови їхньої зрілості та прийнятного рівня зручності для веб-середовища.

З огляду на зростаючу важливість безпеки веб-застосунків, подальші дослідження можуть бути зосереджені на розробці комплексних методик тестування, що поєднують оцінку функціональних аспектів, зручності та безпеки,

включаючи механізми автентифікації. Це дозволить створювати більш надійні та зручні для користувачів веб-рішення, здатні протистояти сучасним загрозам.

Висновки по розділу 3

У даному розділі здійснено безпосередню реалізацію веб-застосунку для вивчення англійської лексики, що ґрунтується на архітектурних та технологічних рішеннях, обґрунтованих у попередніх розділах. Розроблено ключові функціональні модулі, що забезпечують повний цикл взаємодії користувача із системою: від механізмів реєстрації та аутентифікації до інструментів вивчення слів та відстеження прогресу. Забезпечено коректну взаємодію між клієнтською та серверною частинами, а також ефективну роботу з базою даних.

Проведено всебічне тестування розробленого застосунку, що дозволило перевірити відповідність його функціоналу висунутим вимогам та виявити потенційні недоліки. Здійснено апробацію ключових можливостей системи, таких як формування індивідуальних словникових наборів, використання інтерактивних вправ та візуалізація статистики навчання. Отримані результати підтвердили стабільність роботи додатку, його швидкодію та здатність до обробки даних у реальному часі.

Доведено ефективність застосування обраного стеку технологій (MERN-стек) у процесі розробки. Використання єдиної JavaScript-екосистеми дозволило оптимізувати процес створення системи, спростити її підтримку та подальше масштабування. Підтверджено, що реалізовані механізми безпеки забезпечують надійний захист даних користувачів та цілісність інформації, що є критично важливим для освітніх платформ.

Таким чином, у цьому розділі успішно завершено практичну реалізацію веб-застосунку для вивчення англійських слів. Створена система є функціональною,

відповідає сучасним вимогам до навчальних платформ та має потенціал для подальшого вдосконалення та розширення. Результати тестування підтверджують життєздатність та практичну цінність розробленого рішення для підвищення ефективності вивчення іноземної лексики у цифровому середовищі.

ВИСНОВКИ

У рамках даної кваліфікаційної роботи успішно вирішено актуальну науково-практичну задачу розробки веб-застосунку для ефективного вивчення англійської лексики у сучасному цифровому середовищі. Здійснений комплексний аналіз існуючих підходів та онлайн-сервісів дозволив виявити ключові переваги та системні недоліки наявних рішень, що стало методологічною основою для визначення архітектурних та функціональних вимог до розроблюваної системи. Підтверджено, що існуючі платформи, попри свою популярність, не забезпечують повного рівня індивідуалізації та гнучкості, необхідних для всебічного опанування лексичного матеріалу.

Значущість роботи полягає в обґрунтуванні та впровадженні інтегрованого підходу до вивчення англійської мови, який поєднує адаптивні алгоритми, принципи гейміфікації та персоналізовані навчальні траєкторії в рамках єдиного цифрового інструментарію. Доведено, що застосування сучасного стеку веб-технологій, зокрема MongoDB, Express.js, React.js та Node.js, є оптимальним для реалізації високоефективного, масштабованого та безпечного освітнього застосунку, здатного до динамічного розвитку та інтеграції майбутніх функціональних розширень.

В ході проєктування розроблено детальні моделі архітектури системи, що охоплюють функціональні вимоги, структуру бази даних та взаємодію між клієнтською і серверною частинами. Це дозволило створити логічно обґрунтований та технічно вивірений план реалізації, що мінімізує ризики на етапі розробки. Практична реалізація застосунку продемонструвала його відповідність розробленим специфікаціям та заявленим вимогам. Функціональні модулі забезпечують ефективну підтримку інтерактивних вправ, відстеження прогресу користувачів та адаптивний підбір контенту.

Здійснене тестування підтвердило стабільність роботи розробленого Web-застосунку, його швидкодію та надійність у функціонуванні. Отримані результати свідчать про те, що створена система є конкурентоспроможним інструментом, здатним значно підвищити ефективність самостійного вивчення англійської лексики.

Загалом, результати кваліфікаційної роботи свідчать про досягнення поставленої мети та успішне вирішення всіх завдань дослідження. Розроблений веб-застосунок має значний практичний потенціал і може бути використаний для підвищення якості вивчення англійської мови у цифровому освітньому просторі, а також слугувати основою для подальших наукових розробок у галузі адаптивних навчальних систем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ISO/IEC 27001:2013. Information technology - Security techniques - Information security management systems - Requirements.
2. Stallings W. Network Security Essentials: Applications and Standards. - 6th ed. - Pearson, 2020.
3. Рогожин Ю. В. Захист інформації в комп’ютерних системах. - Київ: КНЕУ, 2019.
4. Василенко А. І., Сидоренко О. М. Інформаційна безпека: навчальний посібник. - Харків: НТУ «ХПІ», 2021.
5. Гуменюк В. Сучасні методи автентифікації користувачів у комп’ютерних мережах // Вісник НТУУ «КПІ». - 2021. - №2. - С. 45–52.
6. Hafiz M. A. et al. A Survey on Multi-Factor Authentication Schemes // Journal of Cybersecurity and Privacy. - 2022. - Vol. 2, No. 3. - P. 499–518.
7. P. Smith, "Password Authentication: Weaknesses and Countermeasures," *Journal of Information Security*, vol. 12, no. 4, 2020, pp. 89-98.
8. J. Lee, "Security Challenges of Password-Based Systems," *Cybersecurity Review*, vol. 15, no. 2, 2021, pp. 67-75.
9. K. Johnson, "Two-Factor Authentication: Enhancing Security for the Modern Web," *Tech Innovations Journal*, vol. 18, no. 1, 2019, pp. 112-119.
10. A. Patel, "User Experience in Two-Factor Authentication," *Information Systems Management*, vol. 30, no. 3, 2022, pp. 245-253.
11. L. Brown, "Biometric Authentication: The Future of Security," *Security Technology Magazine*, vol. 8, no. 4, 2020, pp. 33-41.
12. M. Cooper, "Biometric Security and the Risk of Data Leaks," *Privacy and Security Research*, vol. 27, no. 5, 2021, pp. 112-120.

13. R. Turner, "OAuth 2.0: A New Paradigm in Authorization," *Journal of Web Security*, vol. 21, no. 2, 2018, pp. 45-52.
14. H. Daniels, "OpenID Connect: An Overview," *Web Technologies Review*, vol. 9, no. 3, 2020, pp. 66-72.
15. Олійник Г. В., Литвинов В. А., Грибков С. В. Обрання програмної платформи для побудови модуля безпеки web-орієнтованої системи підтримки прийняття рішень // Вісник Національного університету "Львівська політехніка". Серія: Автоматика, вимірювання та керування. – 2016. – № 852. – С. 137–142.
16. Django. The web framework for perfectionists with deadlines. URL: https://www.djangoproject.com/ (дата звернення: 30.04.2025).
17. Bootstrap. A powerful, feature-packed frontend toolkit. URL: https://getbootstrap.com/ (дата звернення: 30.04.2025).
18. Django Two-Factor Authentication. Complete Two-Factor Authentication for Django. URL: https://django-two-factor-auth.readthedocs.io/en/stable/ (дата звернення: 30.04.2025).
19. Бунке О. С. СЕРВЕРНІ WEB-ТЕХНОЛОГІЇ. 2023.
20. Micheal Full (2020). How the Internet Works & the Web Development Process. p. 30. ISBN 979-8641657073.
21. Chris Shiflett (2003). Http Developer's Handbook 1st Edition. Sams. p.400. ISBN 978-0672324543.
22. Ilya Grigorik (2013). High Performance Browser Networking 1st Edition. O'Reilly Media. p. 398. ISBN 978-1449344764.
23. Django. Sending email. URL: https://docs.djangoproject.com/en/5.2/topics/email/ (дата звернення: 30.04.2025).
24. Django. Databases. URL: https://docs.djangoproject.com/en/5.2/ref/databases/ (дата звернення: 30.04.2025).

25. Bass L., Clements P., Kazman R. (2004) Software architecture in practice. Addison Wesley Professional. 452 p
26. A. Barth, C. Jackson, and J. C. Mitchell. Robust Defenses for Cross-Site Request Forgery. In CCS, 2008.
27. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1, 1999.
28. J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart. HTTP Authentication: Basic and Digest Access Authentication, 1999.
29. M. Johns and J. Winter. Protecting the Intranet Against "JavaScript Malware" and Related Attacks. In DIMVA, 2007.
30. N. Jovanovic, E. Kirda, and C. Kruegel. Preventing Cross Site Request Forgery Attacks. Securecomm and Workshops, 2006, pages 1–10, Aug. 28, 2006-Sept. 1, 2006.
31. A. Bortz and D. Boneh. Exposing private information by timing web applications. In WWW '07: Proceedings of the 16th international conference on World Wide Web, pages 621–628, New York, NY, USA, 2007. ACM Press.
32. Python. Programming language that lets you work quickly and integrate systems more effectively. URL: https://www.python.org/ (дата звернення: 30.04.2025).
33. Kaner, C., Bach,J. (2001). Lessons learned in software testing: A Context Driven approach. John Wiley & Sons Inc.
34. Lewis, J.R., & Sauro,J. (2021). Usability and user experience: design and evaluation. URL: https://www.researchgate.net/publication/373487143_USABILITY_AND_USER_EXPERIENCE_DESIGN_AND_EVALUATION (дата звернення: 30.04.2025).
35. Tullis T., &Albert B. (2013). Measuring the User Experience, 2nd Edition. Morgan Kaufmann.

36. Блек, Р., Грехем, Д. (2012). Основи тестування програмного забезпечення. ISTQB Certification. Third Edition. Cengage Learning EMEA.
37. Глюза, М.П., & Вовк, О.В. (2023). Usability-тестування як ефективний показник успішності веб-продуктів. Науковий простір: актуальні питання, досягнення та інновації. (с. 348-350).
38. Діденко, М.В., & Вовк, О.В. (2020). Дослідження методів оцінки ux інтерфейсів нового покоління. Поліграфічні, мультимедійні та web-технології. Т. 2. (с. 128-131).
39. Автентифікація користувача та дозволи. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Authentication (дата звернення: 30.04.2025).
40. Безпека в Django. URL: https://docs.djangoproject.com/en/4.2/topics/security/ (дата звернення: 30.04.2025).
41. ISO/IEC 9126. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE). – System and software quality models / ISO/IEC.
42. Андрейко В.М. Кваліметрія програмного забезпечення засобів вимірювань: монографія. Івано-Франківськ: Симфонія форте, 2016. 120 с.
43. Кузь М.В., Соловко Я.Т., Андрейко В.М. Методологія формування узагальненого критерію якості програмного забезпечення в умовах невизначеності. Вісник Вінницького політехнічного інституту. 2015. ғ 5. С. 104–107.
44. Grid Authentication. URL: https://safenet.gemalto.com/multifactor-authentication/authenticators/grid-authentication (дата звернення: 30.04.2025).
45. Open ID foundation URL: http://openid.net (дата звернення: 30.04.2025).
46. OAuth 2.0. URL: https://oauth.net/ (дата звернення: 30.04.2025).

47. Mozhaiev, O., Gnusov, Y., Manzhai, O., Strukov, V., Nosov, V., Radchenko, V. і Yenhalychev, S. (2023) «Стеганографічний метод захисту акустичної інформації у системах критичного застосування», СУЧАСНИЙ СТАН НАУКОВИХ ДОСЛІДЖЕНЬ ТА ТЕХНОЛОГІЙ В ПРОМИСЛОВОСТІ, С. 52–63.

48. Ляшенко, Г.Є Астраханцев, А.А. (2017). Дослідження ефективності методів біометричної автентифікації. Системи обробки інформації.

ЗГОДА здобувача вищої освіти

Державного університету економіки і технологій про перевірку кваліфікаційної роботи на прояви академічного плагіату

та розміщення в Репозитарії Університету

Я, Кубрак Станіслав Ігорович, підтримую політику Державного університету економіки і технологій з академічної доброчесності і відкритого доступу.

Засвідчую, що кваліфікаційна бакалаврська робота “Розробка системи реєстрації та автентифікації” виконана самостійно та не містить академічного плагіату. Я не надавав і не одержував недозволену допомогу під час підготовки цієї роботи. Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Із чинним Положенням про запобігання та виявлення академічного плагіату в роботах здобувачів вищої освіти Державного університету економіки і технологій ознайомлений. Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі порушення норм академічної доброчесності робота не допускається до захисту або оцінюється незадовільно.

Також я поінформований, що відповідно до «Положення про Репозитарій (електронну базу даних) Державного університету економіки і технологій» зазначена робота буде розміщена в Електронному архіві Університету (Репозитарії ДУЕТ). З умовами такого розміщення ознайомлений.

10.06.2025 Кубрак С.І.