НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

«На правах рукопису»
УДК_____

«До захисту допущено»
Завідувач кафедри
_____ Наталія АУШЕВА
“___”_____2025 р.

Магістерська дисертація

на здобуття ступеня другого (магістерського) рівня вищої освіти
за освітньою програмою “Цифрові технології в енергетиці”
зі спеціальності 122 “Комп’ютерні науки”

на тему «Методи адаптивного підбору та захисту даних веб-порталів з використанням штучного інтелекту»

Виконав: студент _2_ курсу, групи ТР-43мп
Багній Антон Іванович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник ст. викл. каф. ЦТЕ, к.т.н. Ірина МИХАЙЛОВА _____
(посада, науковий ступінь, вчене звання, ім’я, ПРІЗВИЩЕ) (підпис)

Рецензент доц. каф. ТАЕ, к.т.н., доц., Олександр СІРИЙ _____
(посада, науковий ступінь, вчене звання, ім’я, ПРІЗВИЩЕ) (підпис)

Н.контроль асистент Володимир РУДИК _____
(посада, ім’я, ПРІЗВИЩЕ) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ — 2025

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти — другий (магістерський)
спеціальність 122 “Комп’ютерні науки”
Освітньо-наукова програма “Цифрові технології в енергетиці”

ЗАТВЕРДЖУЮ
Завідувач кафедри ЦТЕ
_____ Наталія АУШЕВА_

«___»_____2025 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Багнію Антону Івановичу
(прізвище, ім’я, по батькові)

1. Тема дисертації: *«Методи адаптивного підбору та захисту даних веб-порталів з використанням штучного інтелекту»*
керівник роботи *Михайлова Ірина Юріївна, доцент, к.т.н*
(прізвище, ім’я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «03» листопада 2025 р. № 4779-с

2. Термін подання студентом дисертації: 06.12.2025

3. Об’єкт дослідження: методи забезпечення безпеки та персоналізації даних у веб-порталах електронної комерції.

4. Вихідні дані: Статистичні дані щодо кіберзагроз (звіти Barracuda Networks, OWASP); технічна документація фреймворків NestJS, Next.js та бази даних MongoDB; математичні методи (EWMA, SVD, TF-IDF).

5. Перелік завдань: 1) проаналізувати сучасні кіберзагрози та існуючі рішення безпеки і персоналізації в e-commerce; 2) дослідити математичні методи виявлення загроз та побудови рекомендацій; 3) розробити архітектуру семирівневої системи захисту веб-порталу; 4) реалізувати гібридну систему рекомендацій (матрична факторизація, контентна фільтрація); 5) провести тестування системи та оцінити ефективність рішень.

6. Орієнтовний перелік ілюстративного матеріалу: актуальність теми та постановка задачі; архітектура семирівневої системи захисту; схема гібридної системи рекомендацій; структура бази даних та архітектура програмного забезпечення;

результати тестування системи.

7. Орієнтовний перелік публікацій: Тези доповіді на конференції «Science of XXI century: development, main theories and achievements» (м. Гаага, 31.10.2025).

8. Дата видачі завдання «16» вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів підготовки магістерської дисертації	Термін виконання	Примітка
1	Аналіз літературних джерел та існуючих рішень захисту і рекомендацій e-commerce	01.09.25 — 10.09.25	Виконано
2	Проєктування архітектури системи та вибір технологічного стеку	11.09.25 — 20.09.25	Виконано
3	Програмна реалізація модулів безпеки (WAF, bot detection, 2FA)	21.09.25 — 05.10.25	Виконано
4	Реалізація гібридної системи рекомендацій та клієнтської частини	06.10.25 — 14.10.25	Виконано
5	Проведення тестування системи та аналіз ефективності	15.10.25 — 19.10.25	Виконано
6	Захист програмного забезпечення	20.10.25 — 24.10.25	Виконано
7	Оформлення магістерської дисертації	25.10.25 — 20.11.25	Виконано
8	Підготовка презентації та ілюстративного матеріалу	21.11.25 — 23.11.25	Виконано
9	Передзахист дисертації	24.11.25 — 28.11.25	Виконано
10	Нормоконтроль	24.11.25 — 03.12.25	Виконано
11	Захист магістерської дисертації	22.12.25 — 26.12.25	Виконано

Студент _____ **Антон БАГНІЙ**
(підпис) (ім’я, ПРІЗВИЩЕ)

Керівник роботи _____ **Ірина МИХАЙЛОВА**
(підпис) (ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ

Актуальність теми. Розвиток електронної комерції супроводжується зростанням кіберзагроз, зокрема атак ботів, які складають до 60% трафіку. Існуючі рішення часто занадто дорогі для малого бізнесу, а традиційні методи не забезпечують балансу між безпекою та зручністю. Актуальність роботи зумовлена необхідністю створення доступної комплексної системи, що поєднує адаптивний захист та персоналізацію на основі штучного інтелекту.

Мета роботи — розробити методи адаптивного підбору та захисту даних веб-порталів з використанням штучного інтелекту для підвищення рівня безпеки та персоналізації користувацького досвіду.

Завдання дослідження — проаналізувати кіберзагрози та методи захисту у сучасних веб-сервісах; дослідити математичні методи виявлення загроз, криптографії та побудови рекомендацій; розробити архітектуру семирівневої системи захисту (WAF, анти-бот, 2FA, ризик-менеджмент); реалізувати гібридну систему рекомендацій на основі матричної факторизації та фільтрації; протестувати систему та оцінити ефективність рішень.

Об’єкт дослідження — методи забезпечення безпеки та персоналізації даних у веб-порталах електронної комерції.

Предмет дослідження — методи адаптивного захисту даних, алгоритми виявлення автоматизованих систем та методи персоналізованих рекомендацій.

Методи дослідження: поведінковий аналіз (для ботів); метод EWMA (управління ризиками); криптографічні методи (AES-256, bcrypt, TOTP); матричну факторизацію (SVD, SGD), TF-IDF та косинусну схожість (рекомендації).

Наукова новизна одержаних результатів. Запропоновано комплексний підхід до захисту e-commerce, що інтегрує 7 рівнів безпеки з адаптивним управлінням ризиками (EWMA). Розроблено гібридний алгоритм рекомендацій, що поєднує методи фільтрації через rank-based blending. Удосконалено метод виявлення ботів за рахунок динамічного ризик-скору.

Практичне значення одержаних результатів. Створено програмне забезпечення, що поєднує захист та персоналізацію. Система розгорнута у виробничому середовищі, а власна реалізація алгоритмів забезпечила оптимізацію витрат.

Апробація результатів дисертації. Основні положення доповідалися на науковій конференції «Science of XXI century: development, main theories and achievements» (м. Гаага, 31.10.2025).

Структура та обсяг роботи. Дисертація складається зі вступу, п’яти розділів, висновків, списку джерел та додатків. Загальний обсяг дисертації — 119 сторінок. Основний вміст викладено на 114 сторінках. Робота містить 36 рисунків, 10 таблиць. Список джерел — 30 найменувань .

Ключові слова: ЕЛЕКТРОННА КОМЕРЦІЯ, КІБЕРБЕЗПЕКА, ШТУЧНИЙ ІНТЕЛЕКТ, РЕКОМЕНДАЦІЙНА СИСТЕМА, WAF, АДАПТИВНИЙ ЗАХИСТ, МАТРИЧНА ФАКТОРИЗАЦІЯ, EWMA.

ABSTRACT

Relevance of the topic. The development of e-commerce is accompanied by an increase in cyber threats, in particular bot attacks, which account for up to 60% of traffic. Existing solutions are often too expensive for small businesses, and traditional methods do not provide a balance between security and convenience. The relevance of this work is due to the need to create an affordable comprehensive system that combines adaptive protection and personalization based on artificial intelligence.

The goal of this work is to improve the security and personalization of e-commerce web portals by developing a system of adaptive protection and hybrid recommendations.

The objectives of the study are to analyze cyber threats and protection methods in modern web services; to investigate mathematical methods for threat detection, cryptography, and recommendation generation; to develop the architecture of a seven-level protection system (WAF, anti-bot, 2FA, risk management); to implement a hybrid recommendation system based on matrix factorization and filtering; to test the system and evaluate the effectiveness of solutions.

The object of research is the processes of ensuring data security and personalization in e-commerce web portals.

The subject of the study is adaptive data protection methods, algorithms for detecting automated systems, and methods for personalized recommendations.

Research methods: behavioral analysis (for bots); EWMA method (risk management); cryptographic methods (AES-256, bcrypt, TOTP); matrix factorization (SVD, SGD), TF-IDF, and cosine similarity (recommendations).

Scientific novelty of the results obtained. A comprehensive approach to e-commerce protection is proposed, integrating seven levels of security with adaptive risk management (EWMA). A hybrid recommendation algorithm has been developed, combining filtering methods through rank-based blending. The method of detecting bots has been improved by means of a dynamic risk score.

Practical significance of the results obtained. Software combining protection

and personalization has been created. The system has been deployed in a production environment, and the implementation of proprietary algorithms has ensured cost optimization.

Approval of the dissertation results. The main provisions were reported at the scientific conference “Science of the 21st century: development, main theories and achievements” (The Hague, October 31, 2025).

Structure and scope of work. The dissertation consists of an introduction, five chapters, conclusions, a list of sources, and appendices. The total length of the dissertation is 119 pages. The main content is presented on 114 pages. The work contains 36 figures and 10 tables. The list of sources contains 30 titles.

Keywords: E-COMMERCE, CYBERSECURITY, ARTIFICIAL INTELLIGENCE, RECOMMENDATION SYSTEM, WAF, ADAPTIVE PROTECTION, MATRIX FACTORIZATION, EWMA.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

2FA	Two-Factor Authentication
AES	Advanced Encryption Standard
API	Application Programming Interface
CDN	Content Delivery Network
CSP	Content Security Policy
DDoS	Distributed Denial of Service
EWMA	Exponentially Weighted Moving Average
GDPR	General Data Protection Regulation
HTTP	HyperText Transfer Protocol
JWT	JSON Web Token
SaaS	Software as a Service
SEO	Search Engine Optimization
SGD	Stochastic Gradient Descent
SVD	Singular Value Decomposition
TF-IDF	Term Frequency-Inverse Document Frequency
TOTP	Time-based One-Time Password
WAF	Web Application Firewall
XSS	Cross-Site Scripting

ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується масштабною цифровізацією бізнес-процесів, зокрема у сфері електронної комерції. Веб-портали інтернет-магазинів щодня обробляють мільйони транзакцій, зберігають конфіденційні дані користувачів та забезпечують роботу складних бізнес процесів. Водночас зростання популярності онлайн торгівлі супроводжується підвищенням вимог користувачів до якості сервісу — вони очікують не тільки швидку роботу системи, а також зручність, безпечність та персоналізований досвід користування.

Однак інтенсивний розвиток електронної комерції все більше привертає уваги зловмисників, які використовують все більш складні методи атак для виведення з ладу системи, отримання даних користувачів або здійснення шахрайських операцій. Традиційні підходи до забезпечення безпеки веб-застосунків часто виявляються недостатньо ефективними, саме тому з’являється потреба в розробці адаптивних засобів захисту з використанням штучного інтелекту, які будуть ефективно виявляти та усувати різноманітні загрози. Водночас залишається актуальною тема дотримання балансу між безпекою та зручністю, оскільки надмірно жорсткі механізми можуть негативно впливати на досвід користування системою і внаслідок зменшувати конверсію продажів.

Актуальність теми. Стрімкий розвиток електронної комерції супроводжується значним зростанням кіберзагроз, які спрямовані на веб-портали. За даними звітів з кібербезпеки в 2024 році 60% трафіку деяких інтернет-магазинів створювали автоматизовані системи, зокрема шкідливі боти [1]. Такі системи викрадають конфіденційну інформацію, намагаються здійснити несанкціонований доступ, спричиняють хибні спрацювання на веб-порталах та DDoS-атаки. Водночас зростають вимоги від користувачів до персоналізації в інтернет-магазинах. Вони очікують релевантних рекомендацій та індивідуального підходу, а для виконання цього потрібен збір та обробка великої кількості персональних даних. Це створює складну задачу для тримання балансу між забезпеченням високого рівня безпеки та наданням якісного персоналізованого досвіду користування.

Існують готові рішення для захисту веб-застосунків (Cloudflare, AWS Shield, Akamai Bot Manager), які є ефективними, але мають недоліки щодо адаптації під конкретний проєкт та фінансових аспектів. Великі компанії (Netflix, Amazon) використовують складні системи рекомендацій з нейронними мережами, що вимагають значних обчислювальних ресурсів. Таким чином, актуальність дослідження зумовлена необхідністю розробки комплексного рішення, яке поєднує багаторівневу систему захисту з адаптивною персоналізацією, залишаючись економічно доступним та інтегрованим безпосередньо у веб-застосунок.

Мета дослідження. Метою роботи є розробка методів адаптивного підбору та захисту даних веб-порталів з використанням штучного інтелекту для підвищення рівня безпеки та персоналізації користувацького досвіду.

Для досягнення поставленої мети потрібно виконати наступні завдання:

1) провести аналіз сучасних кіберзагроз для веб-порталів електронної комерції та існуючих рішень у великих веб-сервісах;

2) дослідити математичні методи для виявлення автоматизованих загроз, криптографічного захисту, управління ризиками та рекомендаційних систем;

3) розробити архітектуру семирівневої системи захисту веб-порталу, що включає виявлення ботів, WAF, двофакторну автентифікацію, контроль спроб входу, адаптивне управління ризиками, управління сесіями та CSP;

4) реалізувати гібридну систему адаптивних рекомендацій з використанням матричної факторизації, SGD, колаборативної та контентної фільтрації;

5) провести комплексне тестування розробленої системи та оцінити ефективність запропонованих рішень.

Об’єкт дослідження — методи забезпечення безпеки та персоналізації даних у веб-порталах електронної комерції.

Предмет дослідження — методи адаптивного підбору та захисту даних веб-порталів, алгоритми виявлення діяльності автоматизованих шкідливих систем та системи персоналізованих рекомендацій з використанням штучного інтелекту.

Методи дослідження. Для вирішення поставлених завдань використовувались:

— методи поведінкового аналізу для визначення ботів на основі взаємодії користувача з інтерфейсом;

— алгоритми обчислення ризик-скору з використанням зваженої суми нормалізованих ознак;

— криптографічні методи (bcrypt для хешування паролів, AES-256-GCM для шифрування секретів 2FA);

— метод EWMA для адаптивного управління ризиками на основі історії безпекових інцидентів;

— алгоритм матричної факторизації зі стохастичним градієнтним спуском для колаборативної фільтрації;

— методи TF-IDF та косинусної схожості для контентної фільтрації;

— алгоритми rank-based blending для об’єднання результатів різних методів рекомендацій;

— методи модульного тестування для перевірки коректності функціонування компонентів.

Наукова новизна одержаних результатів. Вперше запропоновано комплексний підхід до побудови системи захисту веб-порталів e-commerce, що інтегрує сім рівнів безпеки з адаптивним управлінням ризиками на основі EWMA та поведінкового аналізу. Розроблено гібридний алгоритм створення персоналізованих рекомендацій, що поєднує факторизацію, колаборативну та контентну фільтрацію через rank-based blending. Удосконалено метод визначення діяльності автоматизованих систем за рахунок динамічного обчислення ризик-скору та багаторівневого реагування. Запропоновано архітектурне рішення для інтеграції системи безпеки безпосередньо в логіку веб-застосунку на рівні сервісів.

Практичне значення одержаних результатів. Створене програмне забезпечення є сучасним рішенням для застосування у проєктах електронної комерції, поєднуючи багаторівневий адаптивний захист з персоналізацією користувацького досвіду. Система розгорнута у виробничому середовищі, що дозволяє практично використовувати продукт. Власна реалізація алгоритмів дозволила оптимізувати обчислювальні витрати та адаптувати функціонал під специфічні вимоги бізнесів різних масштабів.

Апробація результатів дисертації. Основні положення та результати роботи доповідалися на науковій конференції студентів «Science of XXI century: development, main theories and achievements» (м. Гаага, 31.10.2025).

Публікації. За результатами досліджень опубліковано 1 наукову працю: тези доповіді на конференції «Science of XXI century: development, main theories and achievements» (5 друк. арк.).

Структура та обсяг роботи. Магістерська дисертація складається зі вступу, п’яти розділів, висновків, списку використаних джерел та додатків. Загальний обсяг дисертації становить 119 сторінок. Основний вміст викладено на 114 сторінках. Робота містить 36 рисунків, 10 таблиць. Список використаних джерел налічує 30 найменувань. Додатки розміщені на 2 сторінках і включають ER діаграму бази даних та сертифікат про апробацію роботи.

У першому розділі здійснено постановку задачі дослідження та аналіз предметної області захисту даних в електронній комерції. Розглянуто основні типи кіберзагроз та існуючі комерційні рішення для захисту веб-застосунків.

У другому розділі розглянуто теоретичні основи та математичні методи розроблюваної системи. Описано алгоритм виявлення ботів, метод адаптивного управління ризиками на основі EWMA та алгоритм матричної факторизації для систем рекомендацій.

У третьому розділі наведено опис архітектури та програмної реалізації розробленого програмного забезпечення. Представлено архітектуру системи, реалізацію семи рівнів безпеки та структуру модуля AI-рекомендацій.

У четвертому розділі представлено сценарії роботи користувача з системою та результати тестування. Описано процес автентифікації з двофакторною автентифікацією, функціонал адміністративної панелі моніторингу та результати комплексного тестування.

У п'ятому розділі розглянуто питання практичного впровадження системи у форматі стартап-проєкту. Проведено маркетинговий аналіз ринку, розроблено стратегію виходу на ринок та оцінено фінансову модель проєкту.

1 ЗАДАЧА АДАПТИВНОГО ПІДБОРУ ТА ЗАХИСТУ ДАНИХ ВЕБ-ПОРТАЛІВ З ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

У цьому розділі проведемо детальний аналіз сучасного стану кібербезпеки у сфері електронної комерції, розглянемо основні типи загроз, з якими найбільше стикаються веб-портали в цій галузі, проаналізуємо існуючі рішення провідних компаній для протидії загрозам. Також розглянемо основні підходи до рекомендаційних механізмів, які використовують передові e-commerce платформи. На основі проведеного аналізу формуються вимоги до розроблюваної системи адаптивного захисту та персоналізації.

1.1 Сучасний стан та виклики кібербезпеки веб-порталів електронної комерції

Протягом останніх років галузь електронної комерції стабільно зростає як за обсягами продажів, так і за кількістю активних користувачів, які регулярно виконують покупки. Пандемія COVID-19 значно прискорила цифровізацію торгівлі, змусивши мільйони людей перейти на онлайн-покупки, а бізнеси переводити свої магазини в інтернет. За оцінками аналітиків, глобальний ринок e-commerce у 2024 році перевищив позначку у 6 трильйонів доларів США [2]. Водночас зростання популярності онлайн-торгівлі привернуло увагу кіберзлочинців, які користуються цим та все більше розглядають такі веб-портали для злочинних дій.

Основні категорії загроз можна класифікувати за декількома напрямками. Перший напрямок стосується автоматизованих атак, які здійснюються за допомогою ботів. Такі програми виконують масові запити до веб-серверів з різноманітними цілями: парсинг цінної інформації конкурентами, крадіжка облікових записів методом підбору паролів, виведення з ладу системи, бронювання

чи замовлення на веб-порталі для подальшої перепродажі. Активність ботів щороку зростає, статистика аналітиків за 2024 рік взагалі невтішна. Як виявилось, майже половина всього bot-трафіку, а саме 49% припадає на просунуті шкідливі боти, такі програми здатні імітувати поведінку реальних користувачів що значно ускладнює виявлення. Крім цього зі всього трафіку 6% становлять відомі шкідливі боти та імперсонатори. Таким чином, більше половини автоматизованого трафіку на e-commerce сайтах має потенційно шкідливий або точно шкідливий характер, що підкреслює масштаб проблеми. Результат аналізу всесвітньовідомої компанії Barracuda Networks, яка спеціалізується на кібербезпеці, наведено на рисунку 1.1.

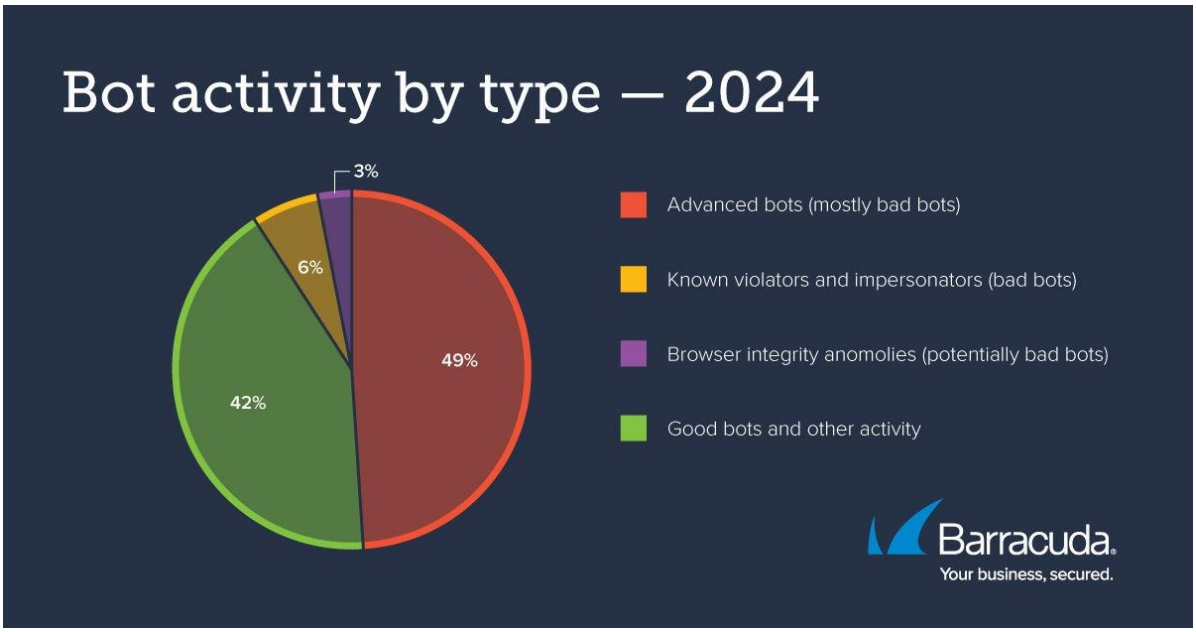


Рисунок 1.1 — Кругова діаграма розподілу типів трафіку на e-commerce сайтах

Другим важливим напрямком загроз є атаки на рівні веб-застосунків, які використовують вразливості програмного коду. Міжнародна організація Open Web Application Security Project (OWASP) [3] щорічно публікує рейтинг топ 10 найкритичніших вразливостей веб-застосунків. На рисунку 1.2 представлено класифікацію OWASP, яка включає десять основних категорій вразливостей, а саме: недостатнє логування та моніторинг, порушення контролю доступу, ін’єкції, криптографічні помилки, помилки ідентифікації та автентифікації, використання

застарілих компонентів, підробка серверних запитів (SSRF), небезпечний дизайн, помилки конфігурації безпеки, та проблеми цілісності програмного забезпечення.

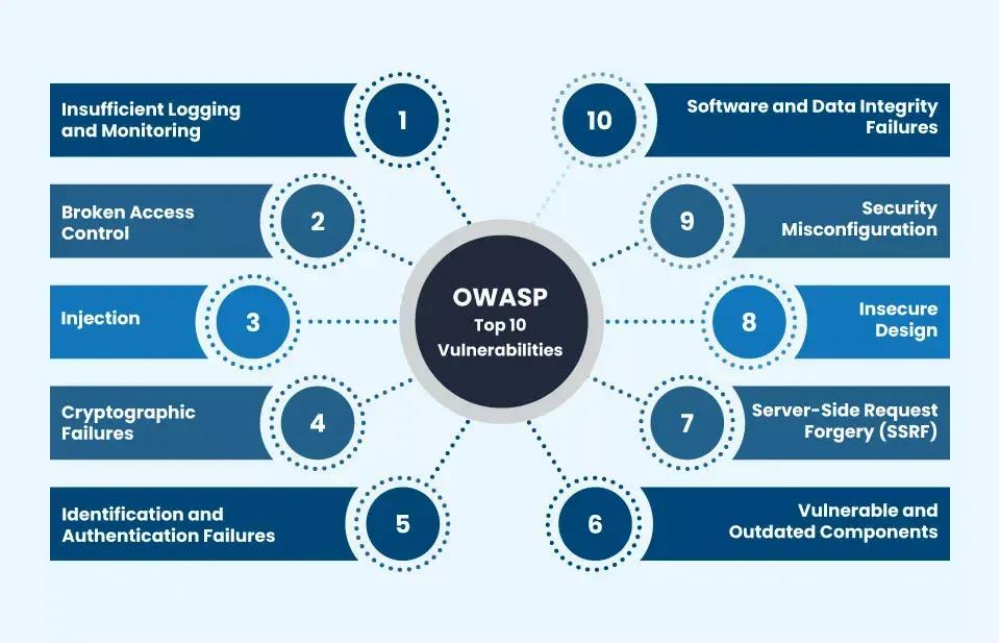


Рисунок 1.2 — Десять найкритичніших вразливостей веб-додатків за класифікацією OWASP

Найпоширенішими серед загроз є SQL-ін’єкції та міжсайтовий скриптинг (XSS). За допомогою SQL-ін’єкцій зловмисники намагаються виконати запит до бази даних, обходячи традиційні механізми перевірки доступу. XSS-атаки впроваджують шкідливий JavaScript-код через недостатню фільтрацію користувацьких даних. Порушення контролю доступу дозволяє отримати несанкціонований доступ до привілейованих функцій. Криптографічні помилки в коді призводять до компрометації даних користувачів.

Атаки на облікові записи користувачів — це третій напрямок загроз. Зловмисники використовують бази скомпрометованих логінів і паролів з інших сервісів для спроб входу на цільовий ресурс. Зважаючи на те, що багато користувачів використовують однакові паролі на різних платформах, такі атаки виявляються досить ефективними. Brute-force атаки є більш простим методом послідовного перебору потенційних комбінацій символів. Розподілені атаки з

різних IP-адрес можуть обходити обмеження, накладені сучасними системами на кількість спроб входу.

DDoS-атаки є ще одним потенційним джерелом загроз. Їхня мета дуже проста: заповнити сервер такою кількістю запитів, щоб він не встигав обслуговувати звичайних користувачів.

Зловмисники зазвичай використовують ботнети, які представляють собою мережі з тисяч заражених комп'ютерів, планшетів, телефонів і навіть розумних побутових пристроїв. Власники цих пристроїв навіть не підозрюють, що їхні пристрої мають шкідливе програмне забезпечення, яке допомагає зловмисникам в кібератаках. Як виконується DDoS-атака зображено на рисунку 1.3.

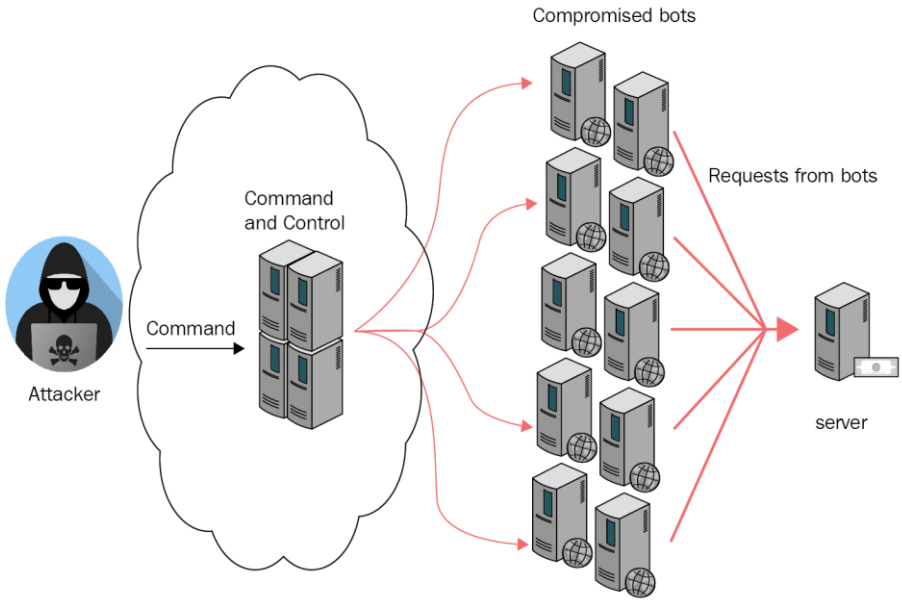


Рисунок 1.3 — Схема проведення DDoS-атаки через ботнет на веб-портал електронної комерції

Схема демонструє класичний сценарій атаки. Спочатку хакер якимось чином заражає низку пристроїв і встановлює на них шкідливе програмне забезпечення. Ці заражені машини можуть бути по всьому світу. Після цього він підключає їх усіх до свого командного сервера, який на схемі позначений як «Команда та керування». Зловмисник просто натискає умовну кнопку під час атаки, і всі ці тисячі ботів починають одночасно слати запити на сервер жертви. Сервер намагається

відповісти на всі ці запити, але все ж не вистачає ресурсів і він просто перестає працювати.

Для бізнесу кожна година простою призводить до втрати тисяч потенційних клієнтів, які не можуть виконати свої замовлення і в результаті бізнес втрачає гроші. Коли атака відбувається під час якихось свят чи розпродажів, наприклад «чорної п’ятниці» або передноворічного періоду, збитки можуть становити мільйони. Крім того, компанія втрачає репутацію, оскільки інфраструктура виявляється ненадійною, і клієнти переходять до конкурентів.

Необхідно розуміти, що DDoS-атаки стали набагато серйознішими, ніж вони були в минулому. Тепер зустрічаються атаки на сотні гігабіт і навіть терабіти в секунду, тоді як раніше атака на 10 гігабіт вважалася потужною. Боти також розвинулися, тепер вони можуть імітувати поведінку реальних користувачів, обходити системи захисту та змінювати стратегії атак в реальному часі.

Статистика кібератак показує кілька тривожних тенденцій. По-перше, атаки стають все складнішими, оскільки сучасні боти навчені імітувати реальних користувачів. Це робить традиційні методи виявлення атак все менш ефективними. По-друге, зловмисники реагують швидше на нові вразливості, оскільки час, необхідний для масового використання вразливості, може бути лише кілька годин.

Таким чином, кібербезпека веб-порталів електронної комерції є надзвичайно важливим і багатогранним питанням, яке потребує комплексного підходу до вирішення. Розробка нових адаптивних систем безпеки з використанням технологій штучного інтелекту та машинного навчання є необхідною, оскільки традиційні методи захисту виявляються недостатніми проти сучасних загроз.

1.2 Постановка задачі дослідження та вимоги до системи

Проаналізувавши сучасний стан кібербезпеки в e-commerce можна чітко сформулювати проблему: інтернет-магазини зараз знаходяться в складній ситуації, загроз стає все більше і хакери використовують щоразу складніші методи атак.

Виходячи з цього, основна мета дослідження — розробити комплексну систему, яка поєднує багаторівневий адаптивний захист веб-порталу з персоналізованими рекомендаціями, при цьому залишаючись достатньо доступною для впровадження у малому та середньому бізнесі.

Для досягнення поставленої мети необхідно вирішити такі задачі:

— проаналізувати існуючі комплексні платформи безпеки (Cloudflare, AWS Shield, Akamai) та системи рекомендацій великих веб-сервісів (Amazon, Netflix), виявити їх переваги та обмеження для застосування в середньому e-commerce;

— дослідити методи виявлення та класифікації автоматизованих загроз, зокрема алгоритми поведінкового аналізу для визначення ботів та обчислення ризик-скору користувачів;

— дослідити криптографічні методи захисту даних, включаючи сучасні алгоритми хешування паролів та механізми двофакторної автентифікації на основі TOTP;

— проаналізувати математичні основи рекомендаційних систем, зокрема методи колаборативної та контентної фільтрації, матричної факторизації та стохастичного градієнтного спуску;

— розробити багаторівневу архітектуру захисту веб-застосунків, що включає сім рівнів безпеки від мережевого до прикладного;

— проаналізувати та вибрати технологічний стек для реалізації системи, враховуючи вимоги до продуктивності, масштабованості та зручності впровадження;

— спроектувати гібридну систему рекомендацій, що поєднує переваги різних підходів до фільтрації та забезпечує роботу навіть для нових користувачів;

— розробити структуру бази даних, що оптимально зберігає дані користувачів, товарів, сесій, логів безпеки та рекомендацій;

— реалізувати програмні модулі системи безпеки: детектування ботів, WAF, двофакторну автентифікацію, адаптивне управління ризиками;

— реалізувати систему рекомендацій на основі матричної факторизації з використанням методу SVD та алгоритмів злиття результатів;

— розробити клієнтську частину та адміністративну панель для керування системою;

— провести тестування розробленої системи та проаналізувати ефективність її роботи в реальних умовах.

До розробленої системи висуваються такі вимоги:

— модульність — кожен компонент (визначення ботів, WAF, двофакторна автентифікація, рекомендації) має бути окремим модулем, який можна включати/виключати та налаштовувати незалежно. Це дає гнучкість — якщо комусь не потрібна якась частина функціоналу, він може її не використовувати;

— продуктивність — система не має сильно сповільнювати роботу сайту. Всі перевірки безпеки мають відпрацьовувати швидко (у ідеальному випадку до 100-200 мілісекунд), рекомендації теж мають генеруватися оперативно;

— масштабованість — рішення має працювати як для магазину з сотнею товарів і тисячею користувачів на день, так і для більших проєктів. Звісно, не на рівні Amazon, але хоча б до десятків тисяч користувачів;

— зручність впровадження — програмне забезпечення має легко розгортатись у виробничому середовищі;

— доступність технологічного стеку — використовувати популярні та добре підтримувані технології.

Також критично важливим є забезпечення високого рівня автономності системи, що дозволить мінімізувати необхідність ручного втручання адміністраторів у процеси моніторингу та реагування на інциденти. Крім того, архітектура рішення повинна передбачати можливість легкого масштабування та адаптації до нових векторів атак, які неминуче з’являтимуться з розвитком технологій.

Отже, в результаті дослідження має бути створена практично застосовна система, яка закриває основні потреби середнього e-commerce проєкту в безпеці та персоналізації, при цьому залишаючись достатньо гнучкою та доступною для впровадження.

1.3 Огляд існуючих рішень для захисту та персоналізації

Сучасний ринок кібербезпеки пропонує багато різних рішень для захисту веб-застосунків — від простих плагінів до хмарних сервісів та складних корпоративних платформ. Щоб зрозуміти, яке рішення підходить саме для e-commerce, потрібно проаналізувати готові рішення та вияснити які в них є плюси і мінуси, та коли вони можуть бути використані.

1.3.1 Комплексні платформи безпеки

На ринку є кілька великих гравців, які пропонують комплексний захист для веб-сайтів. Найвідоміші з них — це Cloudflare, AWS Shield та Akamai. Кожна з цих платформ має свої особливості, але в загальному всі вони стають проміжною ланкою між користувачами та сервером, фільтруючи весь трафік.

Cloudflare — це найпопулярніше рішення для малого та середнього бізнесу. Це CDN (мережа доставки контенту) з вбудованими функціями безпеки. Коли ви підключаєте Cloudflare, весь трафік до вашого сайту спочатку проходить через їхні сервери. Вони автоматично блокують підозрілі запити, за допомогою своїх механізмів безпеки, захищають від DDoS-атак, а також можуть ввімкнути додаткову перевірку для ботів при підозрілій активності. Основна перевага — це простота налаштування та безкоштовний базовий план. Але є і мінуси: ви залежите від зовнішнього сервісу, а деякі складні налаштування доступні тільки в дорогих тарифах. Також можлива ситуація що Cloudflare перестане працювати на деякий час. Така ситуація вже траплялась і в той момент перестали працювати тисячі сервісів, які залежали від цього програмного забезпечення. Хоча такі ситуації трапляються вкрай рідко, але все ж залежність від зовнішніх сервісів створюють додаткові ризики [4].

AWS Shield — це рішення від Amazon для захисту від DDoS-атак [5]. Якщо ваш проєкт вже працює на інфраструктурі AWS, то Shield інтегрується дуже легко. Є дві версії: Standard (безкоштовна, але базова) та Advanced (платна, з розширеними можливостями). Advanced версія коштує від 3000 доларів на місяць, що для багатьох малих та середніх бізнесів просто не під’ємна ціна. Але якщо ви

великий інтернет-магазин на AWS з серйозним трафіком, то такі вкладення того варті — вони гарантують захист навіть від найпотужніших атак.

Akamai — це дуже потужний сервіс у світі CDN та кібербезпеки. Вони обслуговують величезні компанії такі, як Apple та Sony. Їхнє рішення Kona Site Defender пропонує захист від веб-атак, DDoS, і має вбудований WAF (Web Application Firewall). Головним мінусом є ціна. Це рішення для великих корпорацій з бюджетами в мільйони, а тому для звичайного інтернет-магазину таке рішення не підійде.

Крім того, варто згадати про Web Application Firewall, також відомий як WAF. Цей файрвол працює на рівні веб-додатків. Він перевіряє трафік HTTP/HTTPS і блокує небезпечні запити. Популярні рішення включають Imperva WAF, Barracuda WAF і ModSecurity, які є безкоштовними open-source-рішеннями. Незважаючи на те, що ModSecurity можна налаштувати самостійно на своєму сервері, слід вивчити правила та регулярно їх оновлювати. Комерційні WAF простіші, але вони також дорогі.

На рисунку 1.4 показано типову архітектуру захисту з використанням CDN та WAF.

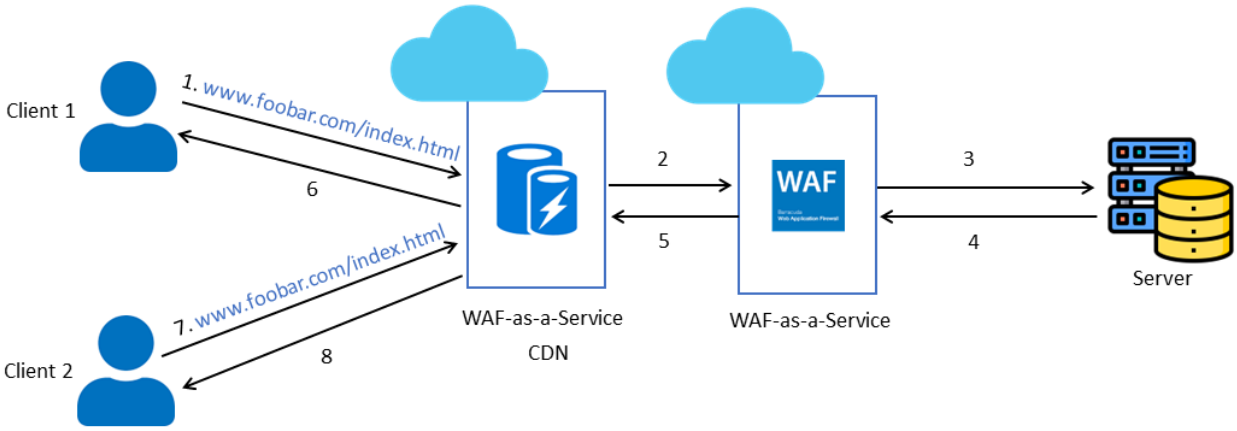


Рисунок 1.4 — Типова архітектура багаторівневого захисту веб-застосунку

Зі схеми видно як працює багаторівнева архітектура. Коли клієнти надсилають запити, вони потрапляють до CDN, який кешує контент та виконує основну фільтрацію трафіку. Далі запит надсилається до WAF-as-a-Service. WAF визначає шкідливі патерни, такі як SQL-ін’єкції, XSS-атаки та підозрілі заголовки. Потім, якщо запит легітимний, передається на реальний сервер застосунку, де обробляється бізнес-логіка та взаємодія з базою даних. У цій архітектурі є кілька переваг: WAF відсікає атаки ще до того, як вони дістануться до вашого коду, а сам сервер залишається захищеним від зовнішнього доступу. Крім того, CDN знижує навантаження на основний сервер і пришвидшує відповідь користувачів з різних місць.

Akamai Bot Manager є однією з найкращих програм для виявлення ботів. Вона аналізує поведінку користувачів за допомогою машинного навчання. Швидкість кліків, рух миші, час на сторінці, відбитки браузера та інші параметри контролюються системою. Якщо бот виявляє ознаки підозрілих дій, він буде заблокований або отримає додаткову перевірку, схожу на капчу. Мінусами є складність і висока вартість.

Більш доступною альтернативою є DataDome, яка також використовує штучний інтелект для пошуку ботів. Вони заявляють, що їхнє програмне забезпечення здатне з точністю понад 99 відсотків відрізнити робота від людини за першим запитом. Це робиться скриптом, який виконується в браузері користувача та збирає дані про поведінку.

HUMAN Security — ще одне популярне рішення. Вони спеціалізуються на захисті від автоматизованих атак: credential stuffing, account takeover, скрейпінг цін та товарів. Вони використовують поведінковий аналіз та машинне навчання. Їхньою особливістю є те що вони можуть відслідковувати діяльність бота на різних сайтах, тобто якщо бот атакує кілька їхніх клієнтів, система це помітить.

reCAPTCHA від Google є найдоступнішим і найдешевшим варіантом, який зазвичай безкоштовний. Традиційно він пропонує користувачу натиснути на кнопку щоб підтвердити що це не робот або вибрати зображення з певним елементом. Нова версія reCAPTCHA v3 працює без видимих дій для користувача, він просто оцінює поведінку та видає оцінку від 0 до 1. Якщо оцінка низька,

можливо, це бот. Проблема полягає в тому, що досвідчені боти можуть обходити капчу, а звичайних користувачів іноді вона дратує.

Проблема більшості існуючих рішень в тому, що вони або дуже дорогі (Akamai, AWS Advanced), або недостатньо гнучкі (готові платформи не дають налаштувати все під свої потреби), або вимагають багато часу на налаштування та підтримку. Для середнього інтернет-магазину потрібне щось посередині — достатньо потужне, але не за великі гроші, та з можливістю адаптувати під конкретні потреби бізнесу. Саме тому в цій роботі пропонується розробити власне комплексне рішення, яке поєднує кращі практики існуючих систем, але дає гнучкість у налаштуванні та не вимагає величезного бюджету.

1.3.2 Системи рекомендацій у великих веб-сервісах

Окрім безпеки, для успішного інтернет-магазину критично важлива персоналізація. Сучасні покупці звикли, що сайт розуміє їхні потреби і показує саме те, що може їх зацікавити. Це все відбувається за допомогою алгоритмів, які намагаються передбачити, який товар може сподобатися конкретному користувачеві.

Щоб зрозуміти як працюють рекомендації на практиці, та який підхід краще використовувати варто дослідити які механізми використовують великі та популярні сервіси.

Загалом, Amazon є лідером e-commerce рекомендацій. Система рекомендацій становить приблизно 35% їхніх продажів. Вони використовують складну гібридну систему, яка розглядає низку факторів, включаючи те, що ви переглянули, купили, що лежить у кошику, що ви шукали, і навіть наскільки довго ви залишалися на сторінці товару. Навіть коли у вас є мільйони товарів, рекомендації швидко обробляються завдяки їхньому алгоритму сортування предметів.

Перевагою їхнього підходу є дуже точні рекомендації, які реально працюють на продажі. Недоліком є те що така система вимагає величезної інфраструктури та команди спеціалістів з обробки даних. Для малого бізнесу це нереально.

Хоча Netflix не є онлайн-магазином, вони теж використовують алгоритми для рекомендацій і їхній підхід є інноваційним. Їхня система відстежує не тільки те, що

ви дивилися, але й коли і на якому пристрої ви перемотали або чи додивились до кінця. Вони навіть можуть змінювати обкладинки фільмів, показуючи різним користувачам різну обкладинку, зважаючи на те що привертає увагу конкретного користувача. За оцінками Netflix, їхня система рекомендацій [6] дозволяє їм заощадити понад мільярд доларів на рік завдяки тому, що їхні клієнти не так часто відписуються через відсутність цікавого контенту і не переходять до конкурентів.

Spotify робить схожі речі з музикою. Вони використовують три типи даних: колаборативну фільтрацію (що слухають схожі користувачі), аналіз аудіо (темп, тональність, гучність), та аналіз текстів про музику в інтернеті.

Проаналізувавши результати дослідження готових рішень у великих платформах, можна зробити певні висновки, які допоможуть зробити якісний програмний продукт. По-перше, не варто відразу робити дуже складний алгоритм. Можна почати з простої фільтрації на основі покупок. По-друге, правильне збирання даних стосується не лише покупок, але й переглядів, додавання в вішлист і часу на сторінці. По-третє, система повинна постійно тестуватися та удосконалюватися, оскільки те, що працювало місяць тому, може зараз не працювати.

Основна проблема чому рішення які використовуються у розглянутих сервісах не підходять для середнього та малого бізнесу полягає в тому, що готові рішення цих компаній недоступні (Amazon не продає свою технологію), а сторонні сервіси рекомендацій, такі як Monetate або Dynamic Yield, коштують від кількох тисяч доларів на місяць. Це дорого для середнього бізнесу. Таким чином, має сенс створити унікальне гібридне рішення, яке буде не тільки ефективним, але й простим для впровадження.

1.4 Багаторівнева архітектура захисту веб-застосунків

Основна ідея багаторівневого захисту полягає в тому, що якщо злочинець пройде один рівень захисту, то є ще декілька, які можуть його зупинити. Незважаючи на те, що жоден із механізмів не є ідеальним, вони разом створюють

надзвичайно сильний захист. Важливо, щоб рівні були різноманітними, оскільки те, що хакер не може обійти один тип захисту, не означає, що він може обійти інший тип захисту.

Пропонується семирівнева архітектура захисту для проєкту e-commerce. Саме сім рівнів, бо більша складність вже надто велика без значного підвищення безпеки, а менша — не охоплює всі основні вектори атак. Кожен рівень працює самостійно та вирішує певну проблему, але разом вони складають потужну систему.

Перший рівень — визначення ботів. Це перша лінія оборони, яка відсіює автоматизовані запити ще на вході. Суть в тому, щоб розпізнати чи це людина користується веб-порталом та відправляє запити чи автоматизована система. Для цього аналізується багато різних параметрів: як швидко користувач рухає мишею, User-Agent браузера, кількість кліків мишею, час який пройшов від заходу на сторінку до активних дій, як довго він затримується на сторінці та інші. Боти зазвичай поводяться неприродньо, тобто надто швидко, однаково, рухи мишкою часто по прямій лінії, або взагалі без них, виконання дій без затримок і помилок, які зазвичай робить людина.

Якщо система виявить бота, вона може діяти різними способами, залежно від ступеня загрози. Просто показати капчу при низькому ризику. Середня — це додаткові перевірки, які, наприклад, вимагають виконання математичної задачі. При високому рівні доступ буде повністю заблокований. Така градація дозволяє не тільки уникати зайвих перевірок, але й не пропускати очевидних ботів.

Другим рівнем є Web Application Firewall (WAF). Він фільтрує всі запити HTTP/HTTPS. WAF перевіряє кожен запит на потенційно шкідливі патерни. Наприклад, якщо в параметрі URL є символи, які нагадують SQL-команди, такі як SELECT, UNION або DROP, це може вказувати на потенційну SQL-ін’єкцію. Теги <script> у полі форми також можуть бути атакою XSS. WAF, який працює на основі правил, може блокувати або логувати запити, які виглядають підозрілими

Важливо розуміти що WAF не є гарантованим захистом. В досвідчених хакерів є різні прийоми для його обходу, такі як: кодування символів, використання рідкісних векторів атак і використання функцій парсерів. Саме тому WAF має

постійно оновлюватись, тобто потрібно додавати актуальні правила, які відповідають новим типам атак. Також треба налаштовувати його під конкретний проєкт, інакше може бути або занадто багато блокування нормальних запитів, або навпаки пропуск реальних атак.

Третій рівень — криптографічний захист даних. Це метод забезпечення захисту даних з використанням правил зберігання паролів, токенів, створення паролів достатньої складності. Паролі користувачів ніколи не повинні зберігатись у відкритому вигляді, всі вони мають хешуватись, і не просто за допомогою MD5, оскільки він не надійний і вже зламаний, а щось серйозне, наприклад bcrypt чи Argon2. Ці інструменти спеціально розроблені стійкими до брутфорсу і на даний момент є надійними. Токени сесій теж мають бути криптографічно стійкими — не просто послідовні числа, а випадкові рядки достатньої довжини.

Двофакторна автентифікація або 2FA є четвертим рівнем. Зловмисник не зможе ввійти без другого фактора, навіть якщо він знає пароль користувача за допомогою фішингу, брутфорсу або витоку бази даних. TOTP (Time-based One-Time Password), який використовується в Authy та Google Authenticator, є найбільш поширеним варіантом. Принцип простий: код, створений на телефоні, має термін дії 30 секунд і його потрібно ввести під час входу. Секретний ключ і поточний час служать основою для створення коду. Оскільки сервер і клієнт синхронізовані, вони можуть згенерувати код однаково.

П’ятий рівень включає моніторинг спроб входу. Це захищає від брутфорс-атак. Система має помітити та припинити намагання когось підібрати пароль після сотні спроб. Обмеження кількості невдалих спроб є стандартним підходом. Наприклад, обліковий запис тимчасово блокується на тридцять хвилин після п’яти невдалих спроб протягом десяти хвилин.

Шостий рівень включає адаптивне управління ризиками. Цей рівень і є ядром адаптивної системи безпеки, він дозволяє системі стати «розумною» і виявляти навіть невідомі раніше вид атак. Ідея полягає в тому, що система зберігає історію безпекових подій і потім динамічно змінює рівень перевірок. Для IP-адрес, з яких постійно надходять підозрілі запити підвищуємо рівень перевірок. Якщо користувач несподівано логується з нової країни, незважаючи на те, що він

зазвичай проживає в Україні, це може бути підозріло, і можна вимагати додаткову перевірку. Метод експоненційно зваженого ковзного середнього (EWMA), який використовується в цьому рівні захисту враховує як нові, так і історичні події, причому нові події мають більшу вагу.

Сьомий рівень включає керування сесіями. Користувач отримує токен сесії після входу. Токени повинні бути криптографічно випадковими, достатньо довгими та передаватися лише через HTTPS з прапорцями HttpOnly та Secure. Це гарантує, що JavaScript не може достукатися до них і що вони не передаються через HTTP. Крім того, слід встановити розумний час життя сесії. Він не повинен бути занадто коротким, оскільки користувачі будуть постійно виходити, але й не повинен бути занадто довгим, оскільки це може сприяти атакам. Крім того, при зміні пароля або логауту потрібно обов’язково інвалідувати сесію.

На рисунку 1.5 зображена загальна схема семирівневої архітектури захисту.

Рисунок 1.5 — Семирівнева архітектура захисту веб-застосунку

Як видно зі схеми, кожен запит проходить через усі рівні послідовно. Важливо що кожен рівень може як пропустити запит далі, так і заблокувати його або вимагати додаткову перевірку. Це забезпечує гнучкість системи — вона може адаптуватися під різні ситуації без жорстких статичних правил.

Моніторинг та логування є теж не менш важливим аспектом. Кожен рівень має логувати свої рішення: чому запит було заблоковано, який рівень ризику було присвоєно, які правила спрацювали тощо. Це потрібно для аналізу адміністратором системи та налаштування, аналізуючи логи можна зрозуміти які правила працюють добре, а які дають забагато блокувань нормальних запитів.

1.5 Вибір та обґрунтування технологічного стеку

Вибір технологічного стеку для розробки системи адаптивного підбору та захисту даних веб-порталів є критично важливим, оскільки він впливає на продуктивність, масштабованість, зручність розробки та подальшої підтримки системи. При виборі технологій враховувалися наступні критерії: продуктивність та можливість обробки великої кількості запитів, наявність вбудованих засобів безпеки, активна спільнота розробників та наявність документації, можливість масштабування системи, зручність інтеграції різних компонентів.

Для серверної частини обрано NestJS — це прогресивний фреймворк [7] для побудови ефективних та масштабованих серверних застосунків. NestJS базується на TypeScript, що забезпечує сильну типізацію та виявлення помилок на етапі компіляції, що критично важливо для систем безпеки. Фреймворк використовує модульну архітектуру, де кожен функціональний блок (автентифікація, рекомендації, безпека) є окремим модулем. Це відповідає вимозі модульності системи. NestJS має вбудовану підтримку dependency injection, що спрощує тестування та дозволяє легко замінювати компоненти. Фреймворк надає декоратори для створення guards, interceptors та middleware, що ідеально підходить для реалізації різних рівнів безпеки.

Для бази даних вибрано MongoDB — документо-орієнтовану NoSQL базу даних [8]. MongoDB зберігає дані у вигляді JSON-подібних документів, з якими дуже зручно працювати через TypeScript. База легко масштабується горизонтально через шардинг, а це важливо врахувати при майбутньому зростанні кількості користувачів та товарів. MongoDB забезпечує високу швидкість читання/запису, що критично важливо для системи рекомендацій, яка повинна швидко обробляти великі обсяги даних. Гнучка схема даних дозволяє легко адаптуватися до змін вимог без необхідності складних міграцій. NestJS має офіційний пакет @nestjs/mongoose, він надає зручний інтерфейс для роботи з MongoDB, включаючи валідацію даних та визначення схем.

Для клієнтської частини використовується React — бібліотека для побудови користувацьких інтерфейсів. React використовує компонентний підхід, що дозволяє створювати повторно використовувані та легкі для тестування елементи інтерфейсу. Віртуальний DOM забезпечує високу продуктивність при оновленні інтерфейсу. React має величезну екосистему бібліотек та інструментів, активну спільноту розробників. Можливість використання Next.js як фреймворку на базі React додає server-side rendering для покращення SEO та початкової швидкості завантаження сторінок [9]. React Hook Form забезпечує ефективну роботу з формами, а Zustand — просте управління станом застосунку.

Для стилізації інтерфейсу користувача обрано CSS-фреймворк Tailwind CSS. На відміну від традиційних підходів, він використовує методологію utility-first, що дозволяє швидко створювати адаптивні та сучасні інтерфейси безпосередньо у розмітці компонентів. Це значно пришвидшує процес розробки та забезпечує єдиний візуальний стиль для всього веб-порталу. Використання Tailwind CSS також сприяє оптимізації розміру кінцевого бандлу стилів, оскільки у виробничу версію потрапляють лише ті класи, які фактично використовуються в проєкті, що позитивно впливає на швидкість завантаження сторінок.

Для реалізації двофакторної автентифікації обрано бібліотеку otplib. Вона підтримує стандарт TOTP (Time-based One-Time Password) [10], що використовується популярними автентифікаторами типу Google Authenticator та Authy. Бібліотека є легкою, не має зайвих залежностей та забезпечує надійну

генерацію одноразових кодів. Інтеграція з NestJS відбувається через створення окремого сервісу, що інкапсулює логіку роботи з OTP.

Для хешування паролів використовується bcryptjs — JavaScript-реалізація алгоритму bcrypt. Bcrypt спеціально розроблений для хешування паролів і є стійким до brute-force атак завдяки можливості налаштування складності обчислень. Бібліотека bcryptjs не має нативних залежностей, що спрощує її використання в різних середовищах. Вона автоматично генерує та зберігає сіль разом з хешем, що спрощує реалізацію. В NestJS bcryptjs використовується в сервісі аутентифікації для хешування паролів при реєстрації та верифікації при вході.

Для системи рекомендацій на основі матричної факторизації використовується власна реалізація алгоритму SVD (Singular Value Decomposition) з бібліотекою ml-matrix для матричних операцій. Це дає повне розуміння його роботи та можливість оптимізації під конкретні потреби проєкту. Алгоритм інкапсульовано в окремий сервіс NestJS, який можна використовувати в інших сервісах.

Для забезпечення безпеки HTTP-заголовків використовується middleware helmet. Helmet автоматично встановлює різні HTTP-заголовки для захисту від поширених веб-вразливостей, таких як XSS, clickjacking, MIME type sniffing [11] тощо. В NestJS helmet підключається глобально в main.ts файлі.

Для тестування використовується Jest — фреймворк для тестування TypeScript-коду, який є стандартним для NestJS.

Для забезпечення надійності розгортання та ізоляції середовища виконання використовується технологія контейнеризації Docker.

Важливим фактором вибору технологічного стеку стало використання єдиної мови програмування TypeScript як для серверної, так і для клієнтської частини. Це дозволяє використовувати спільні інтерфейси (DTO) та типи даних, що значно зменшує кількість помилок при інтеграції фронтенду з бекендом. Такий підхід спрощує підтримку кодової бази, оскільки зміни в структурі даних на сервері автоматично відслідковуються компілятором на клієнті, забезпечуючи наскрізну типізацію та цілісність всієї системи.

Висновки до розділу 1

У першому розділі проаналізовано поточний стан кібербезпеки та персоналізації в електронній комерції, що дозволило визначити основні проблеми та сформулювати завдання дослідження.

Проаналізовано основні типи кіберзагроз для платформ електронної комерції: автоматизовані атаки з використанням ботів, атаки на рівні веб-застосунків (SQL-ін'єкції, XSS згідно з топ-10 OWASP), атаки на облікові записи та DDoS-атаки [12]. Сучасні загрози стають складнішими, що видно з того, що боти можуть імітувати поведінку реальних користувачів, ускладнюючи їх виявлення традиційними методами.

Розглянуто існуючі комерційні рішення для захисту веб-застосунків. Проаналізовано платформи Cloudflare, AWS Shield та Akamai, а також спеціалізовані системи детектування ботів. Виявлено основний недолік — висока вартість комплексних рішень робить їх недоступними для середнього бізнесу. Безкоштовні рішення не забезпечують комплексного захисту та мають обмежені можливості адаптації під конкретні потреби проекту.

Досліджено різні підходи до створення рекомендаційних систем. Визначено три основні методи: колаборативна фільтрація (ефективна при великій базі користувачів, але має проблему холодного старту), контентна фільтрація (ефективна для нових продуктів, але дає передбачувані результати) і гібридні системи (поєднують переваги обох методів, але складніше впроваджувати). Проаналізовано досвід Netflix, Spotify та Amazon для розробки персоналізованих рекомендацій.

На основі проведеного аналізу сформульовано завдання дослідження — розробити комплексну систему, яка поєднує багаторівневий адаптивний захист з персоналізованими рекомендаціями та залишається доступною для впровадження в середніх e-commerce проектах. Визначено основні вимоги до системи: модульність архітектури, висока продуктивність, масштабованість, зручність впровадження та використання популярних технологій.

2 МЕТОДИ АДАПТИВНОГО ПІДБОРУ ТА ЗАХИСТУ ДАНИХ ВЕБ-ПОРТАЛІВ

У попередньому розділі були виявлені основні загрози для e-commerce платформ, проаналізовані існуючі рішення та сформульовані вимоги до системи. Виявилося, що традиційні методи захисту не достатньо ефективні проти сучасних атак, а готові комерційні рішення є надто дорогими для малого та середнього бізнесу.

У цьому розділі розглядаються теоретичні та математичні основи розробленої системи. У першій частині розглядаються методи забезпечення безпеки. Досліджуються алгоритми виявлення автоматизованих загроз, які використовують обчислення ризик-скору та аналіз поведінки. Розглядаються криптографічні методи захисту даних, зокрема алгоритми хешування паролів та механізми двофакторної автентифікації на базі TOTP. Аналізуються підходи до адаптивного управління ризиками з обґрунтуванням вибору методу експоненційно зваженого ковзного середнього.

В другій частині описані математичні основи систем персоналізації. Досліджуються три підходи до побудови рекомендаційних систем: колаборативна фільтрація, контентна фільтрація та гібридні методи.

2.1 Методи виявлення та класифікації автоматизованих загроз

Боти є однією з найбільших проблем для e-commerce. Звичайно не всі боти є шкідливими, наприклад пошукові роботи Google допомагають людям знайти потрібний веб-сайт, хоча є і багато шкідливих. Шкідливі боти можуть красти контент, накручувати перегляди, парсити важливі дані або взагалі атакувати сайт. Тому важливо вміти відрізняти справжніх користувачів від автоматизованих систем.

2.1.1 Поведінковий аналіз для виявлення ботів

Основна ідея поведінкового аналізу, що люди та боти поводяться по-різному. Людина рухає мишкою не ідеально прямолінійно, робить певні паузи між діями, іноді помиляється, вводить дані неправильно тощо. Бот в свою чергу працює дуже механічно, в нього зазвичай ідеальні рухи, однакові інтервали між діями, дуже велика швидкість реакції.

Для того щоб визначити чи виконує дії у веб-порталі автоматизована система, збираємо різні метрики поведінки. По-перше, це взаємодія з мишею: траєкторія руху, швидкість, прискорення, кількість кліків, час між кліками, скільки пройшло часу між заходом на сторінку та виконанням дій. Людина рідко рухає мишу абсолютно прямо — траєкторія має маленькі коливання, а бот часто переміщає курсор прямолінійно від точки А до точки Б. Також в людини є інтервали між кліками, а бот зазвичай клікає миттєво.

По-друге, швидкість друкування. Швидкість набору тексту у людини непостійна — десь швидше, десь повільніше, є паузи. Боти ж або друкують надто швидко (вставляють текст за мілісекунди), або навпаки з ідеально рівними інтервалами.

По-третє, поведінка на сторінці. Визначається скільки часу користувач провів на сторінці перед тим як щось зробити, чи скролив він вниз щоб прочитати контент. Людина зазвичай переглядає сторінку перед покупкою, може перейти на декілька товарів щоб переглянути, дивиться фото товарів, переглядає характеристики, виконує якісь інші дії. Бот часто відразу виконує цільову дію, тобто оформлює замовлення, додає в кошик тощо.

По-четверте, fingerprint браузера. Це набір характеристик браузера та системи користувача: версія браузера, операційна система, список встановлених шрифтів, розширення браузера, роздільна здатність екрану, часовий пояс, мова системи тощо. Комбінація всіх цих параметрів для кожного користувача досить унікальна. Боти часто мають типові fingerprints, наприклад headless Chrome без жодних розширень, або застарілі версії браузерів.

Всі ці параметри збираються в браузері користувача у фоні. Зібрані дані відправляються на сервер де і відбувається аналіз.

2.1.2 Алгоритми обчислення ризик-скору

Після того, як усі метрики були зібрані, потрібно визначити, чи це поведінка людини чи бота. Якщо метрика схожа на бота, то її відразу блокувати неправильно, оскільки це може заблокувати звичайних користувачів. Тому, було вирішено використовувати метод ризик-скору, тобто в результаті певних обчислень отримуємо значення від 0 до 100, де 0 означає «точно людину», а 100 означає «точно бот».

Ризик-скор обчислюється шляхом зважування суми нормалізованих ознак. У першу чергу кожна метрика нормалізується до діапазону, який починається від [0, 1]. Наприклад, швидкість руху миші. Якщо вона становить 100-500 пікселів/секунду, це нормально (значення приблизно 0), а якщо вона перевищує 2000 пікселів/секунду, це підозріло (значення приблизно 1).

Формула обчислення ризик-скору:

$$RiskScore = w_1 \times f_1 + w_2 \times f_2 + w_3 \times f_3 + \cdots + w_n \times f_n ,$$

де f_i — нормалізовані ознаки (від 0 до 1),

w_i — ваги ознак (сума ваг дорівнює 1).

Наприклад, можна використовувати такі ознаки та ваги:

— f_1 — аномальність траєкторії миші, $w_1 = 0.15$

— f_2 — швидкість кліків, $w_2 = 0.20$

— f_3 — час на сторінці, $w_3 = 0.15$

— f_4 — підозрілий fingerprint браузера, $w_4 = 0.25$

— f_5 — відсутність JavaScript або cookies, $w_5 = 0.15$

— f_6 — повторюваність дій (чи робить одне й те саме), $w_6 = 0.10$

Для підбору значення вагів використовуються історичні дані. Деякі ознаки мають більшу вагу, оскільки вони більше вказують на бота. Щоб отримати скор від 0 до 100, результат множиться на 100.

Після обчислення скору приймається рішення залежно від рівня ризику: при значенні 0-30 (низький ризик) запит пропускається без перевірок; при 31-60 (середній ризик) користувачу показується простий CAPTCHA; при 61-80 (високий ризик) застосовується складний CAPTCHA або додаткові перевірки; при 81-100 (дуже високий ризик) запит блокується.

Ці пороги теж налаштовуються залежно від специфіки проєкту. Для публічного магазину можна встановити м’якіші пороги, щоб не псувати досвід користування, для адмін-панелі — жорсткіші.

Ризик-скор обчислюється не один раз, а при кожному запиті користувача. Якщо спочатку все було нормально (низький скор), але потім поведінка змінилася, то скор підвищиться і спрацює захист, завдяки цьому система є адаптивною.

Крім того, система може використовувати записи. Якщо з певної IP-адреси постійно надходять запити з високим ризик-скором, цю IP-адресу можна включити до чорного списку. Також, якщо fingerprint браузера раніше був помічений та зафіксований як підозрілий, то відразу підвищувати ризик-скор для такого користувача.

Правила та машинне навчання добре працюють на практиці. Правила дозволяють швидко відловити очевидних ботів (наприклад, якщо в сучасному веб-застосунку відсутній JavaScript, це точно бот). Машинне навчання є корисним для пошуку складніших патернів, які не очевидні з правил. Але правил з ризик-скором достатньо для невеликих проєктів, оскільки вони простіші у впровадженні та налаштуванні.

2.2 Криптографічні методи захисту даних та двофакторна автентифікація

Криптографічні методи захисту даних є основою безпеки будь-якої системи аутентифікації. Захисту облікових записів користувачів має дуже важливе значення для веб-сайтів електронної комерції, оскільки втручання в облікові записи користувачів може призвести до несанкціонованих покупок, крадіжки персональних даних і фінансових втрат.

2.2.1 Алгоритми хешування та шифрування

Хешування паролів є важливим компонентом безпеки облікових записів. На відміну від шифрування, хешування виконується однією стороною, тому

неможливо отримати вихідний пароль з хешу. Це гарантує, що зловмисники не зможуть отримати паролі користувачів безпосередньо, навіть якщо база даних буде скомпрометована.

У розробленій системі алгоритм bcrypt використовується для хешування паролів. Bcrypt був створений для хешування паролів і має багато переваг перед іншими алгоритмами, такими як MD5 або SHA-256. По-перше, bcrypt є стійким до brute-force атак, оскільки його алгоритм є повільним. Фактор витрат регулює кількість ітерацій алгоритму, що дозволяє йому збільшувати складність у міру збільшення обчислювальної потужності. Алгоритм шифрування Blowfish є математичною основою bcrypt.

Процес хешування можна описати наступною формулою:

$$hash = bcrypt(password, salt, cost),$$

де salt — це випадкова послідовність байтів, яка унікальна для кожного пароля,

cost — параметр складності (кількість ітерацій = 2^cost).

У реалізації використовується cost factor = 10, що означає 2^10 = 1024 ітерації. Вбудована генерація цієї солі є ключовою особливістю bcrypt. У свою структуру кожен хеш автоматично включає сіль, що полегшує зберігання та верифікацію. Результат bcrypt записується таким чином: $2a$10$N9qo8uLOickgx2ZMRZoMyeIjZAgcfl7p92ldGxad68LJZdL17lhWy, де $2a — є версією алгоритму, 10 — cost factor, наступні 22 символи — сіль, а решта символів є власне хешем.

Хешування в розробленому програмному продукті реалізовано таким чином: const hashedPassword = await bcrypt.hash(password, 10);

Для перевірки пароля використовується метод порівняння з бібліотеки bcrypt: const isValid = await bcrypt.compare(inputPassword, storedHash);

Крім хешування паролів, система використовує криптографічне шифрування, щоб захистити секрети двофакторної автентифікації. Для захисту застосовується алгоритм AES-256-GCM. AES-256 використовує ключ розміром 256 біт і гарантує надзвичайно високу безпеку [13].

Режим GCM (Galois/Counter Mode) гарантує автентичність та конфіденційність даних. Він створює тег перевірки, який здатний виявити будь-які спроби змінити зашифровані дані.

Для шифрування секрету TOTP необхідно виконати наступні кроки:

1) створити початковий вектор розміром 12 байт, який є випадковим рядком для кожного шифрування;

2) створити шифр за допомогою алгоритму AES-256-GCM: const cipher = crypto.createCipheriv('aes-256-gcm', key, iv);

3) шифрування даних: const encrypted = Buffer.concat([cipher.update(secret, 'utf8'), cipher.final()]);

4) отримання тегу автентифікації: const tag = cipher.getAuthTag();

5) об'єднання IV, tag та зашифрованих даних у один буфер і кодування в base64.

Розшифрування виконується у зворотному порядку з обов'язковою перевіркою автентифікаційного тегу. Якщо дані були змінені, дешифратор викине виняток.

Ключ шифрування генерується з секретного рядка за допомогою SHA-256: const key = crypto.createHash('sha256').update(secret).digest();
Це забезпечує визначену генерацію 256-бітного ключа з будь-якого секретного рядка.

Додатково система реалізує валідацію стійкості паролів. Перевіряється довжина паролю (мінімум 8 символів); наявність великих та малих літер; наявність цифр; наявність спеціальних символів [14].

Система також забезпечує валідацію стійкості паролів. Перевіряється довжина паролю (не менше восьми символів), наявність великих і малих літер, цифр і спеціальних символів.

Крім того, перевірка паролів здійснюється за допомогою API Have I Been Pwned (HIBP). Цей сервіс містить базу з більш ніж одинадцять мільярдів скомпрометованих паролів, отриманих із реальних витоків даних. Модель k-anonymity гарантує, що перевірка здійснюється безпечно. Для цього пароль хешується за допомогою SHA-1, а перші п’ять символів хешу надсилаються на

сервер, який повертає всі хеші з цими префіксами. Клієнт перевіряє локально, чи міститься повний хеш у відповіді. Якщо пароль виявиться більше десяти разів у витоках, система не дозволить його використання.

2.2.2 Двофакторна автентифікація на основі TOTP

Двофакторна автентифікація (2FA) додає захисту до звичайної схеми логін-пароль. Без одноразового коду зловмисник не зможе ввійти в систему, навіть якщо він знає пароль користувача.

Розроблена система використовує TOTP — алгоритм генерації одноразових паролів на основі часу. Популярні програми, такі як Google Authenticator, Microsoft Authenticator і Authy, використовують TOTP.

Математична основа TOTP базується на алгоритмі HOTP (HMAC-based One-Time Password, RFC 4226) [15]. HOTP генерує одноразовий пароль на основі секретного ключа та лічильника:

$$HOTP(K, C) = Truncate(HMAC - SHA1(K, C)),$$

де K — секретний ключ,

C — лічильник,

HMAC-SHA1 — функція, яка перевіряє повідомлення за SHA-1,

Truncate — функція, яка скорочує до 6-8 цифр.

TOTP розширює HOTP, використовуючи час як лічильник:

$$TOTP = HOTP(K, T),$$

де T = (Current Unix Time - T_0) / X,

T_0 — початковий час (зазвичай 0),

X — часовий крок (зазвичай 30 секунд).

Алгоритм роботи TOTP у розробленій системі:

1) генерація секрету. При активації 2FA система генерує випадковий секретний ключ розміром 160 біт (32 символи у base32): const secret = otplib.authenticator.generateSecret();

2) створення otpauth URL для QR-коду: const otpauthUrl = otplib.authenticator.keyuri(email, 'Wargear', secret) формату URL otpauth://totp/Wargear:email?secret=SECRET&issuer=Wargear;

3) користувач сканує QR-код у застосунку-автентифікаторі, який зберігає секрет локально;

4) під час входу користувач вводить 6-значний код зі свого застосунку. Система генерує код на основі поточного часу та порівнює з введеним: const isValid = otplib.authenticator.check(inputCode, secret);

5) враховується window = 1, що означає прийняття кодів з попереднього та наступного 30-секундного інтервалу. Це компенсує невеликі розбіжності в часі між сервером та пристроєм користувача.

Секрет TOTP зберігається в базі даних у зашифрованому вигляді за допомогою методу AES-256-GCM, як було описано в попередньому підрозділі. Завдяки цьому навіть якщо база даних буде скомпрометована, секрет все одно буде захищений.

Процес активації 2FA складається з декількох етапів, а саме: користувач починає налаштування 2FA шляхом натискання на відповідну кнопку; система створює секрет та відображає QR-код; користувач сканує QR-код за допомогою програми для автентифікації, наприклад Google Authenticator, та вводить перший згенерований код для підтвердження; після успішної верифікації секрет зберігається в зашифрованому вигляді та встановлюється прапорець twoFAEnabled = true.

Якщо при вході в систему виявляється що в користувача увімкнена 2FA, то відбувається наступний алгоритм дій: користувач вводить email та пароль; після перевірки пароля система запитує 2FA код; користувач вводить код зі свого застосунку; система розшифровує секрет, генерує очікуваний код та порівнює з введеним; при успішній верифікації видається JWT токен [16].

Комбінація надійного хешування паролів, шифрування секретів та двофакторної автентифікації створює багаторівневий захист облікових записів користувачів, цим самим значно підвищуючи загальний рівень безпеки системи.

Такий підхід гарантує, що навіть у разі компрометації пароля зловмисник не зможе отримати доступ до облікового запису без фізичного доступу до пристрою користувача. Використання перевірених криптографічних стандартів забезпечує

високу стійкість системи до злому та відповідність сучасним вимогам захисту персональних даних.

2.3 Порівняльний аналіз та обґрунтування вибору методів адаптивного управління ризиками

Адаптивне управління ризиками використовує динамічну оцінку рівня загрози на основі даних минулих років і поточної поведінки. Існують різні математичні підходи до обчислення ризик-скору, всі вони мають свої особливості.

Найпростішим методом є статичне порогове правило, наприклад можна встановити блокування після п’яти невдалих спроб.

Перевагами є передбачуваність та простота, а недоліки — відсутність адаптації, високий рівень хибних спрацювань, вразливість до повільних атак.

Просте ковзне середнє (SMA) обчислює середнє значення за фіксований інтервал:

$$SMA = \frac{(x_1 + x_2 + \cdots + x_n)}{n}$$

Цей метод враховує історію та згладжує викиди. Проте всі значення мають однакову вагу, потрібно зберігати всі дані в пам'яті, а також метод повільно реагує на зміни.

Кращим методом є зважене ковзне середнє (WMA), оскільки він надає більшу вагу новішим значенням, що є важливим. Ось його формула:

$$WMA = \frac{(w_1 \times x_1 + w_2 \times x_2 + \cdots + w_n \times x_n)}{\sum w_i}$$

Метод краще враховує актуальність даних, але все ще потрібно зберігати всі значення та складніше обирати ваги.

Обраний метод, який використовувався в роботі — експоненційно зважене ковзне середнє (EWMA) використовує рекурентну формулу:

$$EWMA_t = \alpha \times x_t + (1 - \alpha) \times EWMA_{t-1},$$

де α — параметр згладжування ($0 < \alpha \leq 1$).

У розробленій системі встановлене α = 0.3, що означає: 30% ваги має поточне значення, 70% — історія.

Ключові переваги EWMA, які вплинули на вибір:

— ефективність пам'яті — потрібно зберігати лише одне попереднє значення, а не весь масив, що є критично важливим для систем з великою кількістю користувачів;

— обчислювальна ефективність — незалежно від розміру історії складність O(1), оскільки обчислення потребують лише одну операцію множення та додавання;

— адаптивність — система надає більшу вагу останнім спостереженням, автоматично адаптуючись до нових паттернів, що дозволяє швидко виявляти зміни в поведінці;

— гнучкість — параметр α можна змінювати залежно від ситуації без зміни архітектури системи.

Система з використанням EWMA використовує чотири метрики: кількість спроб входу на хвилину, частоту невдалих спроб, частоту нових пристроїв та частоту нових локацій.

Загальний ризик-скор обчислюється як зважена сума:

$$Risk = w_a \times AttemptsRisk + w_f \times FailRisk + w_s \times StreakRisk + w_n \times NoveltyRisk,$$

де ваги за замовчуванням: $w_a = 0.4$, wf = 0.4, $w_s = 0.2$, $w_n = 0.0$.

Таким чином, EWMA оптимізує чутливість до атак і толерантність до звичайних помилок, залишаючись масштабованим і обчислювально ефективним методом адаптивного управління ризиками.

2.4 Математичні основи рекомендаційних систем

Рекомендаційні системи є ключовим компонентом персоналізації в системах електронної комерції, вони дозволяють підвищити конверсію продажів та покращити користувацький досвід. В основі будь-якої системи рекомендацій

лежать математичні методи та алгоритми і в результаті за допомогою них можна передбачати уподобання користувачів [17].

2.4.1 Підходи до побудови рекомендаційних систем

В основі будь-якої системи рекомендацій лежить один з трьох основних підходів: колаборативна фільтрація, контентна фільтрація та гібридні методи [18]. Кожен з них має свої переваги та недоліки, які потрібно враховувати при виборі методу під конкретний проєкт.

У великих e-commerce платформах колаборативна фільтрація є найпоширенішим методом. Основною ідеєю є створення рекомендації на основі поведінки великої кількості користувачів. Система знаходить подібності, аналізуючи історії взаємодії користувачів або товарів.

Основною перевагою є базування системи на поведінці користувачів, тому їй не потрібно знати про характеристики товарів. Це дозволяє виявити непередбачені зв’язки між товарами, які не очевидні з назви, опису та характеристик. Крім того, система постійно навчається та покращується і чим більше даних накопичується, тим точніші стають рекомендації. Друга перевага полягає в тому, що, якщо спостерігається зв’язок через дії користувачів, можна рекомендувати товари з різних категорій.

Але є й помітні недоліки. По-перше, проблема холодного старту: новий користувач не може отримати рекомендації, оскільки у нього немає історії взаємодії. Схожа ситуація з новим товаром: система не рекомендує його, поки ніхто не придбав.

По-друге, через те, що в системі зберігаються мільйони записів про товари і користувачів, обчислення потребують значних ресурсів.

По-третє, може виникнути ефект коли система пропонує лише те, що людина вже знає, і не дозволяє відкривати щось нове. Крім того, випадкові, разові покупки, можуть змінити профіль користувача.

Іншим методом є контентна фільтрація. Принцип її роботи полягає в тому, що вона аналізує характеристики та властивості самих товарів. На основі

інформації про покупки та перегляди користувача система створює профіль інтересів для пошуку товарів, які мають схожі характеристики.

Переваги методу: коли є опис характеристик, нові товари можна відразу рекомендувати, що вирішує проблему холодного старту. Рекомендації більш зрозумілі та прозорі, тому користувачу легше зрозуміти причини кожної рекомендації. Для запуску системи не потрібна велика база користувачів. Також немає проблем з конфіденційністю, оскільки ви не повинні відстежувати, як інші користувачі користуються сайтом.

Крім того, в цього методу є і недоліки. Усі товари повинні мати якісні описи, і це може бути важко зробити, особливо коли асортимент великий і різноманітний. Система просто шукає схожі товари за формальними ознаками, не знаходячи несподіваних зв’язків. Рекомендації можуть бути надто прогнозованими або обмеженими. Є ризик того, що клієнти залишаться зациклениими на одній категорії товарів.

Гібридні системи є спробою використати переваги обох методів. Існує багато варіантів реалізації, включаючи просте поєднання результатів кількох алгоритмів, а також складні системи машинного навчання, які адаптивно обирають найкращий метод для кожної ситуації.

Основна перевага очевидна: можливість компенсувати недоліки одного підходу перевагами іншого. Наприклад, новий користувач покладається на контентну фільтрацію, тоді як досвідчені користувачі покладаються на колаборативну. Зазвичай метрики точності гібридних систем перевершують метрики окремих підходів. Є можливість адаптуватися до різних обставин і типів товарів.

Однак недоліки є навіть в такому підході, а саме складніша розробка та налаштування. Через те, що багато алгоритмів працюють одночасно, потрібні більші обчислювальні ресурси. Також якщо щось не працює належним чином, складно визначити, в якому самому компоненті проблема, тому важче налагоджувати та знаходити помилки. Проте, незважаючи на архітектурну складність, саме гібридний підхід забезпечує найбільш стійку роботу системи в умовах розріджених даних та змішаних сценаріїв використання.

Варто зауважити, що ефективність кожного з розглянутих методів суттєво залежить від якості та обсягу накопичених даних, а також від специфіки бізнес-задач, які вирішує веб-портал. Саме тому на етапі проєктування системи критично важливо розуміти кількість доступної інформації, щоб обрати стратегію, яка забезпечить найкращий баланс між точністю рекомендацій та обчислювальними витратами .

В таблиці 2.1 наведено порівняння основних характеристик трьох підходів.

Таблиця 2.1 — Порівняння підходів до побудови рекомендаційних систем

Підхід	**Переваги**	**Недоліки**	**Використання**
Колаборативна фільтрація	не потрібні характеристики товарів; знаходить несподівані зв’язки; покращується з часом; міжкатегорійні рекомендації.	холодний старт для нових користувачів; ресурсоємність при масштабуванні; ефект відображення лише відомих товарів; разові покупки спотворюють профіль.	Великі платформи з багатою історією взаємодій користувачів
Контентна фільтрація	немає проблеми холодного старту товарів; прозорі рекомендації; не потрібна велика база користувачів; конфіденційність.	потрібні якісні описи товарів; передбачувані результати; складно з суб’єктивними характеристиками; надмірна спеціалізація.	Каталоги з добре структурованими характеристиками товарів

Кінець таблиці 2.1.

Гібридна фільтрація	компенсація слабкостей методів; краща точність; адаптивність до ситуацій; універсальність.	складність розробки; більше ресурсів; важче налагоджувати;	E-commerce з різноманітним асортиментом та середньою базою користувачів

Саме тому на етапі проєктування системи критично важливо розуміти кількість доступної інформації, щоб обрати стратегію, яка забезпечить найкращий баланс між точністю рекомендацій та обчислювальними витратами.

2.4.2 Методи колаборативної та контентної фільтрації

Колаборативна фільтрація має два основні варіанти реалізації: user-based та item-based [19]. User-based CF шукає схожих користувачів, item-based CF шукає схожі товари.

User-based колаборативна фільтрація обчислює схожість між користувачами на основі їх історії взаємодій. Однією з найпоширеніших метрик схожості є косинусна схожість між користувачами u та v:

$$sim(u,v) = \frac{(r_u \cdot r_v)}{\|r_u\| \times \|r_v\|} = \frac{\sum_{i=1}^{n} r_{ui} \times r_{vi}}{\sqrt{\sum_{i=1}^{n} (r_{ui})^2} \sqrt{\sum_{i=1}^{n} (r_{vi})^2}},$$

де r_u та r_v — вектори рейтингів користувачів u та v,

i — індекс товару, значення належить діапазону [0,1], де 1 означає ідентичні вподобання [20].

Кореляція Пірсона враховує середні значення рейтингів:

$$sim(u,v) = \frac{\sum_i (r_{ui} - \bar{r}_u) \times (r_{vi} - \bar{r}_v)}{\sqrt{(\sum_i (r_{ui} - \bar{r}_u)^2)} \times \sqrt{(\sum_i (r_{vi} - \bar{r}_v)^2)}},$$

де $\bar{r}_u$ — середній рейтинг користувача u. Значення належить [-1,1], де 1 — повна позитивна кореляція.

Передбачення рейтингу користувача u для товару i:

$$\hat{r}_{ui} = \frac{\bar{r}_u + (\sum_{v \in N} sim(u,v) \times (r_{vi} - \bar{r}_v))}{\sum_{v \in N} |sim(u,v)|},$$

де N — множина k найбільш схожих користувачів (зазвичай k=20-50).

Item-based колаборативна фільтрація обчислює схожість між товарами [21]. Схожість товарів i та j:

$$sim(i,j) = \frac{\sum_u r_{ui} \times r_{uj}}{\sqrt{(\sum_u r_{ui}^2)} \times \sqrt{(\sum_u r_{uj}^2)}}$$

Передбачення рейтингу:

$$\hat{r}_{ui} = \frac{\sum_{j \in N} sim(i,j) \times r_{uj}}{\sum_{j \in N} |sim(i,j)|},$$

де N — множина схожих товарів, з якими користувач вже взаємодіяв. Item-based CF зазвичай стабільніший, оскільки схожість товарів змінюється повільніше, ніж схожість користувачів.

Контентна фільтрація базується на аналізі текстових описів товарів. Основний метод — TF-IDF (Term Frequency-Inverse Document Frequency) [22]. TF (Term Frequency) — частота терміна в документі:

$$TF(t,d) = \frac{f_{t,a}}{\sum_s f_{s,a}},$$

де $f_{t,a}$ — кількість входжень терміна t в документ d.

IDF (Inverse Document Frequency) — зворотна частота документа:

$$IDF(t) = \log(\frac{N}{|\{d:t \in d\}|}),$$

де N — загальна кількість документів, $|\{d: t \in d\}|$ — кількість документів з терміном t.

TF-IDF вага терміна:

$$TF - IDF(t,d) = TF(t,d) \times IDF(t)$$

Кожен товар представляється вектором TF-IDF ваг для всіх термінів словника. Схожість товарів обчислюється через косинусну схожість їх векторів:

$$sim(d_i, d_j) = \frac{(v_i \times v_j)}{(||v_i|| \times ||v_j||)},$$

де v_i та v_j — TF-IDF вектори товарів i та j. Профіль користувача формується як зважена сума векторів товарів, з якими він взаємодіяв. Рекомендації генеруються шляхом пошуку товарів з найбільшою косинусною схожістю до профілю.

2.4.3 Матрична факторизація та стохастичний градієнтний спуск

Колаборативна фільтрація на основі матричної факторизації представляє взаємодії користувачів з товарами у вигляді матриці та розкладає її на дві матриці меншого розміру, які виявляють приховані латентні фактори [23].

Нехай є m користувачів та n товарів. Матриця рейтингів R розміром m×n містить взаємодії користувачів з товарами. Матрична факторизація розкладає R на добуток двох матриць:

$$R \approx P \times Q^T,$$

де P — матриця користувачів розміром m×k,

Q — матриця товарів розміром n×k,

k — кількість латентних факторів (зазвичай 20-50 для e-commerce) [24].

Кожен рядок P представляє користувача, а кожен рядок Q — товар у просторі латентних факторів.

Латентні фактори — це приховані характеристики, що впливають на вподобання: для товарів це може бути ціновий сегмент, стиль, призначення. Система сама виявляє ці фактори з даних. Передбачення рейтингу обчислюється як скалярний добуток:

$$\hat{r}_{ij} = p_i \times q_j = \sum_k p_{ik} \times q_{jk},$$

де p_i — вектор користувача i,

q_j — вектор товару j.

Для знаходження оптимальних матриць P та Q мінімізується функція втрат:

$$L = \sum(i,j) \in K \left(r_{ij} - p_i \times q_j\right)^2 + \lambda(\|p_i\|^2 + \left\|q_j\right\|^2),$$

де K — множина відомих рейтингів,

λ — коефіцієнт регуляризації (0.01-0.1),

$||p_i||^2$ та $||q_j||^2$ — L2-регуляризація для запобігання перенавчанню.

Для мінімізації використовується стохастичний градієнтний спуск (SGD), який оновлює параметри після кожного прикладу [25]. Алгоритм SGD:

1) ініціалізація: P та Q заповнюються випадковими малими значеннями (μ=0, σ=0.01);

2) для кожної епохи: для кожної пари (i, j) з рейтингом r_{ij} обчислюється помилка $e_{ij} = r_{ij} - p_i \cdot q_j$ та оновлюються параметри:

$$p_{ik} \leftarrow p_{ik} + \alpha(e_{ij} \times q_{jk} - \lambda p_{ik})$$

$$q_{jk} \leftarrow q_{jk} + \alpha(e_{ij} \times p_{ik} - \lambda q_{jk}),$$

де α — швидкість навчання (0.001-0.01);

3) повторювати доки функція втрат не перестане зменшуватись або досягнуто максимум ітерацій (20-100 епох).

Якщо помилка e_{ij} позитивна, вектори користувача та товару зближаються у просторі латентних факторів. Регуляризаційний член запобігає надмірному зростанню значень.

Ключові гіперпараметри: кількість латентних факторів k (більше факторів — складніші патерни, але ризик перенавчання), швидкість навчання α (можна використовувати адаптивну: $\alpha(t) = \alpha_0/(1+\beta \times t)$), коефіцієнт регуляризації λ (підбирається через крос-валідацію), кількість епох (з early stopping).

Обчислювальна складність SGD становить $O(|K| \times k)$ на епоху, що значно ефективніше класичного SVD з $O(m \times n \times \min(m,n))$.

Після навчання рекомендації генеруються так: для кожного товару j обчислюється $\hat{r}_{ij} = p_i \cdot q_j$, товари сортуються за спаданням рейтингу, повертаються топ-N товарів (N=10-20).

Переваги методу: ефективність на розріджених даних, можливість онлайн-навчання, автоматичне виявлення латентних факторів, масштабованість. Недоліки: холодний старт для нових користувачів, чутливість до гіперпараметрів, складність інтерпретації факторів. У системі матрична факторизація доповнюється контентною фільтрацією для вирішення проблеми холодного старту.

2.5 Алгоритми гібридизації та ранжування рекомендацій

Жоден з розглянутих методів рекомендацій не є ідеальним для всіх ситуацій. Колаборативна фільтрація ефективна при достатній історії взаємодій, але має проблему холодного старту. Контентна фільтрація працює для нових товарів, але

обмежена якістю описів. Матрична факторизація дає точні прогнози, але потребує багато даних для навчання. Гібридизація дозволяє комбінувати переваги різних методів та компенсувати їх недоліки.

Існує кілька стратегій побудови гібридних рекомендаційних систем. Weighted hybrid комбінує скори різних методів з фіксованими вагами [26]. Switching hybrid динамічно вибирає найкращий метод залежно від ситуації. Cascade hybrid застосовує методи послідовно, використовуючи результати попереднього як вхід наступного. Feature augmentation доповнює один метод ознаками з іншого. Mixed hybrid показує результати кількох методів паралельно. Rank-based blending об'єднує результати на основі рангів товарів у різних списках [27].

У розробленій системі використовується rank-based blending, який є найстабільнішим підходом. Основна ідея: замість комбінування числових скорів (які можуть мати різні масштаби), об'єднуються ранги товарів у топ-N списках від різних методів.

Нехай є M методів рекомендацій: m_1, m_2, ..., m_m. Кожен метод генерує топ-N список товарів з їх скорами. Для товару j від методу k: $score_k(j)$ — оригінальний скор товару; $rank_k(j)$ — ранг товару (1 для найкращого).

Normalized rank для товару j від методу k обчислюється як:

$$nrank_k(j) = \frac{1-(rank_k(j)-1)}{N},$$

де N — розмір топ-списку. Нормалізований ранг належить [0, 1], де 1 — найкращий товар, 0 — найгірший у списку. Якщо товар відсутній у списку методу k, йому присвоюється $nrank_k(j) = 0$.

Фінальний скор товару j обчислюється як зважена сума нормалізованих рангів:

$$score\_final(j) = \sum_k w_k \times nrank_k(j),$$

де w_k — вага методу k, $\Sigma_k w_k = 1$. Ваги методів підбираються експериментально або через крос-валідацію. У системі використовуються наступні ваги: колаборативна фільтрація (матрична факторизація) — 0.5, контентна фільтрація — 0.3, популярність товарів — 0.2.

Алгоритм генерації гібридних рекомендацій:

1) для користувача виконати кожен з методів незалежно, отримавши топ-N списків;

2) для кожного методу k та товару j у його топ-N списку обчислити $nrank_k(j)$;

3) сформувати об'єднану множину всіх товарів U;

4) для кожного товару $j \in U$ обчислити score_final;

5) відсортувати U за спаданням score_final;

6) повернути топ-N товарів з відсортованої множини як фінальні рекомендації;

Обчислювальна складність алгоритму: $O(M \times N)$ для генерації списків + $O(|U| \log |U|)$ для сортування, де $|U| \leq M \times N$. Зазвичай $|U|$ значно менше $M \times N$ через перетин товарів у різних списках.

Переваги rank-based blending: стійкість до різних масштабів скорів від різних методів, простота налаштування (лише ваги методів), можливість легко додавати нові методи, інтуїтивність інтерпретації результатів. Недоліки: втрата точної інформації про різницю між товарами, необхідність підбору оптимальних ваг, однакова обробка всіх товарів незалежно від довіри до методу.

Вирішення проблеми холодного старту в гібридній системі реалізується через каскадну стратегію. Для нового користувача без історії взаємодій:

1) колаборативна фільтрація повертає порожній список (немає даних);

2) контентна фільтрація працює на основі середнього профілю або загальних трендів;

3) компонента популярності повертає топ товарів за глобальною популярністю;

4) фінальна вага автоматично перерозподіляється: контентна — 0.5, популярність — 0.5.

Таким чином, алгоритми гібридизації дозволяють побудувати потужну систему рекомендацій, яка ефективно працює в різних сценаріях: від нових користувачів до досвідчених покупців, від популярних товарів до нішевих позицій. Rank-based blending забезпечує стабільну якість рекомендацій завдяки комбінуванню сильних сторін різних математичних підходів.

Висновки до розділу 2

У другому розділі розглянуто теоретичні основи адаптивного підбору та захисту даних веб-порталів, а також математичні методи. Проаналізували сучасні підходи до виявлення автоматизованих загроз, криптографічного захисту, адаптивного управління ризиками та персоналізації контенту, які використовуються у популярних веб-сервісах.

Досліджено методи виявлення ботів, які використовують систему ризик-скору та поведінковий аналіз, а також багатофакторну оцінку перевірки сесій. Проаналізовано криптографічні алгоритми, а саме: bcrypt для хешування паролів, AES-256-GCM для шифрування даних та TOTP для двофакторної аутентифікації. Розглянули методи адаптивного управління ризиками та визначили, що для наших потреб найкраще підійде EWMA.

Детально дослідили математичні основи рекомендаційних систем, а саме: колаборативну фільтрацію, контентну фільтрацію на основі TF-IDF, та матричну факторизацію з алгоритмом SGD. Визначено переваги та недоліки кожного методу та вибрано підхід, найкращий для розробленого веб-порталу. Розроблено алгоритм гібридизації на основі rank-based blending, який об’єднує переваги різних методів.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ АДАПТИВНОГО ПІДБОРУ ТА ЗАХИСТУ ДАНИХ ВЕБ-ПОРТАЛІВ

В цьому розділі розглядається програмна реалізація адаптивної системи підбору та захисту даних веб-порталів. Описується загальна архітектура програмного забезпечення, проектування гібридної системи рекомендацій, структура бази даних та реалізація розроблених модулів безпеки і персоналізації. Також описана реалізація клієнтської частини та адміністративної панелі для моніторингу відпрацювання функцій безпеки та керуванням ними.

3.1 Загальна архітектура програмного забезпечення

Розроблене програмне забезпечення побудоване за класичною клієнт-серверною архітектурою з чітким розділенням відповідальності. Backend реалізовано як RESTful API, написаний з використанням фреймворку NestJS на базі Node.js, який обробляє HTTP-запити та повертає результати у форматі JSON. Frontend являє собою single-page application з серверним рендерингом на базі Next.js 14.

Backend реалізовано за модульним принципом, який є стандартом для NestJS. Кожен модуль інкапсулює певну функціональність. Основні модулі системи, які були розроблені в рамках цієї роботи: модуль автентифікації (реєстрація, вхід, JWT токени), модуль безпеки (сім рівнів захисту, моніторинг інцидентів, керування сесіями), модуль рекомендацій (гібридна система персоналізації).

Кожен модуль складається з чотирьох елементів, а саме: контролер, сервіс, схема, модуль (який збирає всі частини разом). Controller приймає HTTP-запити та валідує дані через декоратори class-validator. Service містить бізнес-логіку та взаємодіє з базою даних через Mongoose. Schema визначає структуру даних MongoDB з валідацією та індексами. Module налаштовує dependency injection та експортує API для інших модулів.

Між компонентами модуля існує ієрархія залежностей згідно з принципом dependency inversion. Контролери залежать від сервісів, сервіси від схем, а схеми від Mongoose моделей. NestJS використовує dependency injection для управління життєвим циклом об'єктів. Модульна структура backend зображена на рисунку 3.1.

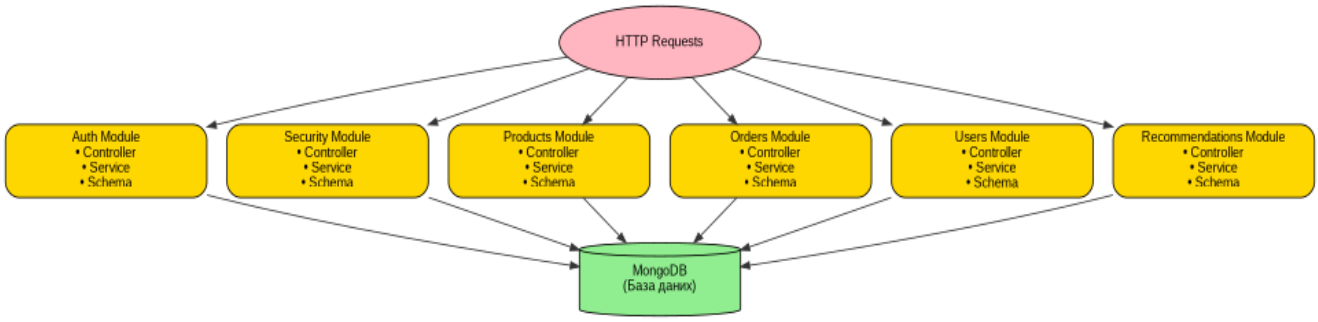


Рисунок 3.1 — Модульна архітектура backend-системи

Frontend розроблено за feature-based структурою в рамках App Router Next.js. Основні директорії в репозиторії: app (сторінки та лейаути), components (повторно використовувані React компоненти), lib (різноманітні утиліти та допоміжні файли), hooks (кастомні React хуки та хуки для роботи з Zustand стором).

Взаємодія між frontend та backend відбувається через централізований API клієнт класу ApiClient. Клієнт забезпечує автоматичне додавання JWT токенів до заголовків, можливість централізованої обробки помилок, інтеграцію bot detection через заголовок X-Bot-Features. Для управління станом використовується SWR для server state (дані з API), useState/useReducer для client state (локальний стан компонентів), Zustand для глобального стану (автентифікація, налаштування, кошик).

База даних MongoDB обрана для зберігання даних завдяки гнучкості та масштабованості. В базі є такі основні колекції: users (облікові записи з хешованими паролями та зашифрованими 2FA секретами), products (каталог товарів), orders (історія замовлень), sessions (активні сесії з JWT), productViews (історія переглядів для рекомендацій), securityEvents (журнал інцидентів), botScores (історія ризик-скорів). Для кожної колекції визначені індекси для оптимізації запитів.

Для безпеки взаємодії між компонентами використовується протокол HTTPS. Для JWT токенів встановлений обмежений термін дії , а саме 1 година для access та 7 днів для refresh. Паролі хешуються за допомогою bcrypt з cost factor = 10 а чутливі дані шифруються AES-256-GCM перед збереженням, як це було описано в попередньому розділі.

В системі логуються всі критичні операції: автентифікація, зміна паролів, блокування користувачів, виявлення ботів, навчання моделей. Логи зберігаються для аналізу проблем продуктивності, безпеки та корегування засобів безпеки шляхом зміни значення ваг.

Така архітектура забезпечує модульність, масштабованість та зручність підтримки. Модульна структура дозволяє незалежно розвивати компоненти. TypeScript забезпечує строгу типізацію та раннє виявлення помилок, ще до компіляції.

3.2 Проектування гібридної системи рекомендацій

Система рекомендацій є однією з основних елементів в розробленому програмному забезпеченні. Вона використовує гібридний підхід, поєднуючи три методи: контентну фільтрацію, колаборативну фільтрацію та рекомендації з використанням AI на основі матричної факторизації, тобто в загальному гібридний підхід. У цьому розділі сфокусуємось на проектуванні та впровадженні системи рекомендацій, математичні основи та алгоритми цих методів, які будуть використовуватись, детально описані в підрозділах 2.4-2.5.

Як і все програмне забезпечення, архітектура системи побудована на основі модулів, цим самим функціонал інкапсульований та відповідальність розділена між модулями. Основним компонентом є ProductsService, який містить опис роботи всіх підсистем і інтерфейс для отримання рекомендацій. Три спеціалізовані модулі забезпечують цей сервіс: модуль контентної фільтрації для аналізу характеристик товарів, модуль колаборативної фільтрації для виявлення патернів у поведінці

користувачів [28] і AIRecoService для генерації рекомендацій на основі матричної факторизації.

Модуль контентної фільтрації виконує алгоритм пошуку схожих товарів на основі атрибутів цих товарів. Для кожного з них формується векторне представлення, що включає категорію, бренд, кольори, розміри та текстовий опис. Схожість між товарами обчислюється за допомогою композитної скорингової функції, яка враховує перетин токенів у текстових полях, відстань за ціною та показники популярності. Таким чином можна зрозуміти наскільки схожі товари і чи потрібно їх рекомендувати при заході на сторінку товару.

Також для виявлення схожості між товарами використовується модуль колаборативної фільтрації, він вже аналізує історичні дані про поведінку користувачів. Для аналізу та знаходження рекомендацій береться до уваги два типи взаємодій: спільні покупки (товари, куплені разом в одному замовленні) та спільні перегляди (товари, переглянуті користувачами, які також переглядали цільовий товар). Для кожного типу взаємодій визначається нормалізована міра схожості. Потім ці результати зважуються з вагами 0,7 для спільних покупок і 0,3 для спільних переглядів, що вказує на те, що реальні покупки мають більшу значущість порівняно з переглядами.

Алгоритм матричної факторизації реалізований в сервісі AIRecoService працює за допомогою стохастичного градієнтного спуску, який детально описано в підрозділі 2.4.3. Повний життєвий цикл моделі підтримується сервером, включаючи ініціалізацію випадкових вбудовувань для користувачів і товарів, навчання за допомогою градієнтного спуску з L2-регуляризацією, зберігання навчених параметрів у базі даних і завантаження їх при старті сервера. Модель навчається асинхронно у фоновому режимі з налаштованою періодичністю без блокування обробки запитів користувачів.

Механізм об’єднання rank-based, який описано в підрозділі 2.5, є ключовою архітектурною особливістю, який забезпечує зважене об’єднання результатів трьох методів [29]. Кожен метод створює список товарів з рейтингами, після чого ProductsService нормалізує позиції в списках і розраховує кінцеві скори відповідно до ваг, які були конфігуровані. Контентна фільтрація становить 50%,

колаборативна фільтрація 30% і рекомендації AI становлять 20%. Такий розподіл ваг обґрунтований експериментально і забезпечує ідеальний баланс між точністю та стабільністю рекомендацій.

Записи про перегляди товарів зберігаються у колекції ProductView і використовуються для формування рекомендацій. Ці дані накопичуються та використовуються для періодичного перенавчання моделі матричної факторизації. Така архітектура дозволяє системі адаптуватися до змін у вподобаннях користувачів та появи нових товарів без ручного втручання. Замовлення все ж мають вищу вагу, оскільки вони є більш достовірною інформацією про те що цікавить користувача. Для кожного товару генеруються схожі товари з використанням всіх трьох методів, після чого результати об’єднуються з урахуванням ваг та типу взаємодії.

Для того щоб уникнути проблеми холодного старту система також має механізм fallback для ситуацій, коли про користувача або товар недостатньо даних, і хоч рекомендація не буде персоналізованою, але все ж блок рекомендацій не буде порожнім. У таких випадках використовуються популярні та рекомендовані товари (з прапорцем isFeatured), відсортовані за рейтингом та кількістю відгуків. Цей підхід гарантує, що користувач завжди отримає релевантні рекомендації, навіть при першому візиті на сайт.

Усі розглянуті механізми — від алгоритмів фільтрації до стратегій обробки холодного старту — інтегровані в єдиний процес обробки даних на рівні бізнес-логіки. Сервісний шар виступає керуючим, який ініціює паралельне виконання обчислень, збирає результати від різних модулів та застосовує правила гібридизації. Такий підхід дозволяє ізолювати складність математичних моделей від клієнтської частини та забезпечити високу швидкодію системи при формуванні персональної видачі.

Архітектура системи рекомендацій зображена на рисунку 3.2, де показано потоки даних між компонентами та їх взаємодію. Модульна структура забезпечує легку масштабованість та можливість незалежного оновлення окремих компонентів без впливу на систему в цілому.

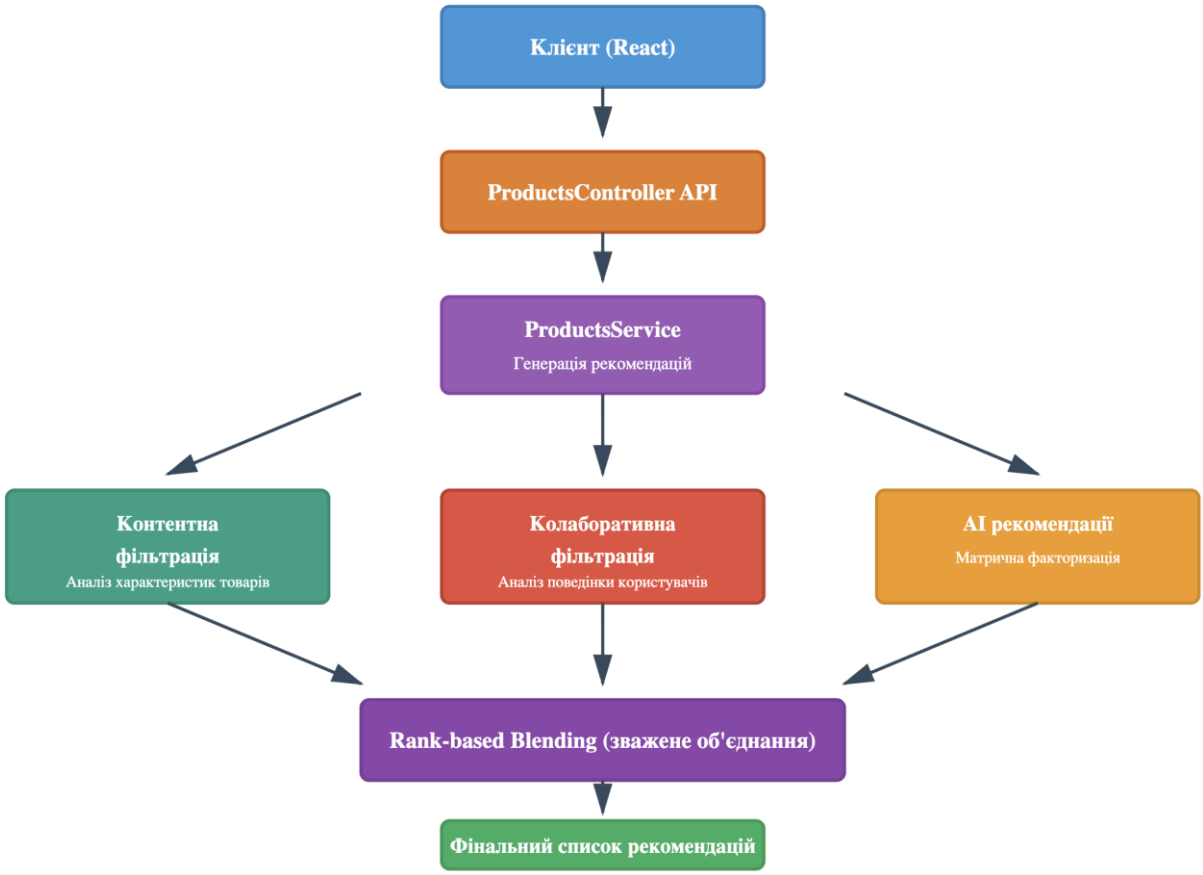


Рисунок 3.2 — Архітектура гібридної системи рекомендацій

Така модульна архітектура забезпечує гнучке масштабування системи та ізольованість обчислювальних процесів, що гарантує стабільну роботу сервісу навіть при високих навантаженнях. Для ефективного зберігання необхідних векторних даних та історії взаємодій було спроектовано відповідну структуру бази даних, яка детально розглядається в наступному підрозділі.

3.3 Проектування структури бази даних

Для зберігання даних в розробленій системі обрано документо-орієнтовану базу даних MongoDB, яка забезпечує гнучкість, високу продуктивність та природню підтримку вкладених структур даних. MongoDB дозволяє швидко адаптувати структуру у разі зміни вимог без необхідності міграції схеми, що особливо важливо на етапі активної розробки. Крім того, горизонтальне

масштабування та вбудовані механізми реплікації забезпечують надійність та доступність системи.

Структура бази даних складається з чотирнадцяти колекцій, кожна з яких відповідає за зберігання певного типу інформації. Колекція User містить облікові записи користувачів системи, включаючи автентифікаційні дані та налаштування безпеки. Колекція Product зберігає інформацію про товари з усіма їх характеристиками, наприклад назву, опис, колір, ціна та інші характеристики. Колекція Order відповідає за замовлення та історію покупок з вкладеними елементами OrderItem. Колекції безпеки (WafIncident, BotConfig, AnomalyProfile, BlockedIp, AccountLock, StepUpChallenge, AiIncident, BotIncident, AuthAttempt, LoginSession) зберігають інформацію, яка використовується для забезпечення безпеки та налаштування системи захисту, тобто різноманітні конфігурації. Колекція ProductView фіксує всі перегляди товарів користувачами для запам’ятовування інтересів. Колекція AiRecoEmbedding зберігає векторні представлення користувачів та товарів для системи рекомендацій.

Колекція User містить базову інформацію про користувачів: унікальний ідентифікатор (_id), ім'я (name), електронну пошту (email), хешований пароль (password), роль у системі (isAdmin) та параметри двофакторної автентифікації (twoFAEnabled, twoFASecret). Поле email має унікальний індекс для швидкого пошуку під час автентифікації. Паролі зберігаються у вигляді bcrypt-хешів з параметром складності 10, що забезпечує захист від перебору. Секрет двофакторної автентифікації шифрується алгоритмом AES-256-GCM перед збереженням у базі даних. Поля timestamps автоматично відслідковують час створення та останнього оновлення запису.

Колекція Product описує товари інтернет-магазину та містить всю необхідну інформацію для їх відображення та обробки. Основні поля включають назву (name), slug для URL, категорію (category), бренд (brand), опис, ціну (price), рейтинг (rating), кількість відгуків (numReviews), доступні розміри (sizes), кольори (colors) та залишок на складі (countInStock). Створені індекси на поля name, slug та category прискорюють операції пошуку та фільтрації.

Колекція Order зберігає інформацію про замовлення користувачів. Кожне замовлення містить посилання на користувача через поле user (ObjectId), адресу доставки (shippingAddress як вкладений об'єкт), метод оплати (paymentMethod), результат платежу (paymentResult), загальну вартість (totalPrice) та статуси оплати й доставки (isPaid, paidAt, isDelivered, deliveredAt). Замість окремої колекції OrderItem, товари зберігаються як вкладений масив items безпосередньо в документі замовлення. Кожен елемент items містить посилання на Product, назву (name), кількість (qty), зображення (image), ціну (price), розмір (size) та колір (color). Індекс на поле user прискорює вибірку замовлень конкретного користувача.

Колекція ProductView фіксує кожен перегляд товару користувачем для побудови персоналізованих рекомендацій. Структура включає userId (string), productId (string) та час перегляду (viewedAt). Створені індекси забезпечують швидкий пошук історії переглядів конкретного користувача або всіх переглядів певного товару. Ці дані використовуються модулем колаборативної фільтрації для виявлення схожості між користувачами та товарами.

Колекції безпеки забезпечують зберігання інформації про події та конфігурацію системи захисту:

— WafIncident — журнал спроб атак на веб-застосунок, містить тип загрози (kind), шлях запиту (path), IP-адресу порушника (ip) та payload атаки;

— AiConfig — налаштування AI Defense з параметрами EWMA (blockRisk, throttleRisk, windowMinutes);

— BotConfig — конфігурація детектора ботів з порогами блокування (blockScore, throttleScore) та увімкненням (enabled);

— AnomalyProfile — профілі аномальної поведінки з ключем (key), частотою помилок (ewmaFailRate) та рівнем ризику (lastRisk);

— BlockedIp — список заблокованих IP-адрес з терміном блокування (blockedUntil) та причиною (reason);

— AccountLock — тимчасово заблоковані облікові записи з email та часом розблокування (lockedUntil);

— StepUpChallenge — активні додаткові перевірки безпеки з питанням (question), відповіддю (answer), терміном дії (expiresAt) та статусом використання (used);

— AiIncident та BotIncident — журнали спрацювань відповідних систем детектування з email, IP, рівнем ризику (risk) та застосованою дією (action);

— AuthAttempt — історія всіх спроб автентифікації з email, IP, результатом (success) та причиною відмови (reason);

— LoginSession — активні сесії користувачів з userId, IP-адресою, User-Agent, геолокацією (country, city), статусом активності (isActive) та часом останньої активності (lastActivityAt).

Колекція AiRecoEmbedding містить векторні представлення для системи матричної факторизації. Кожен запис має тип kind (user або product), унікальний ідентифікатор refId відповідної сутності та вектор vec. Векторні представлення генеруються в процесі навчання моделі алгоритмом стохастичного градієнтного спуску та зберігаються у базі даних для подальшого використання. Індекси на поля kind та refId забезпечують швидкий доступ до embeddings під час генерації рекомендацій. Після кожного перенавчання моделі колекція повністю очищається та заповнюється новими значеннями.

Зв'язки між колекціями реалізовані через посилання на ObjectId документів. Користувач може мати багато замовлень (зв'язок 1:N між User та Order), одне замовлення містить посилання на користувача та вкладений масив items з інформацією про товари, користувач може переглядати багато товарів і кожен товар може мати багато переглядів (зв'язки 1:N між User/Product та ProductView). Для embeddings встановлено зв'язок 1:1, оскільки кожен користувач та кожен товар має рівно один векторний запис.

Розроблена база даних забезпечує баланс між нормалізацією та швидкістю доступу, що є критично важливим для високонавантажених систем електронної комерції. Використання окремих колекцій для зберігання векторних представлень та журналів безпеки дозволяє ізолювати специфічні дані від основної бізнес-логіки, спрощуючи підтримку та масштабування системи. Така організація даних сприяє

ефективному виконанню як транзакційних операцій, так і аналітичних запитів, необхідних для роботи алгоритмів машинного навчання.

Запропонована архітектура даних забезпечує чітко розмежування між основними бізнес-сутностями електронної комерції та службовими колекціями, що відповідають за безпеку та конфігурацію системи . Такий підхід дозволяє оптимізувати продуктивність запитів та гарантує, що обробка безпекових логів не впливатиме на швидкість роботи магазину. Для детального ознайомлення з усіма зв’язками, типами полів та структурою колекцій розроблено повну схему бази даних. Графічне представлення ER-діаграми зображено на рисунку А1.

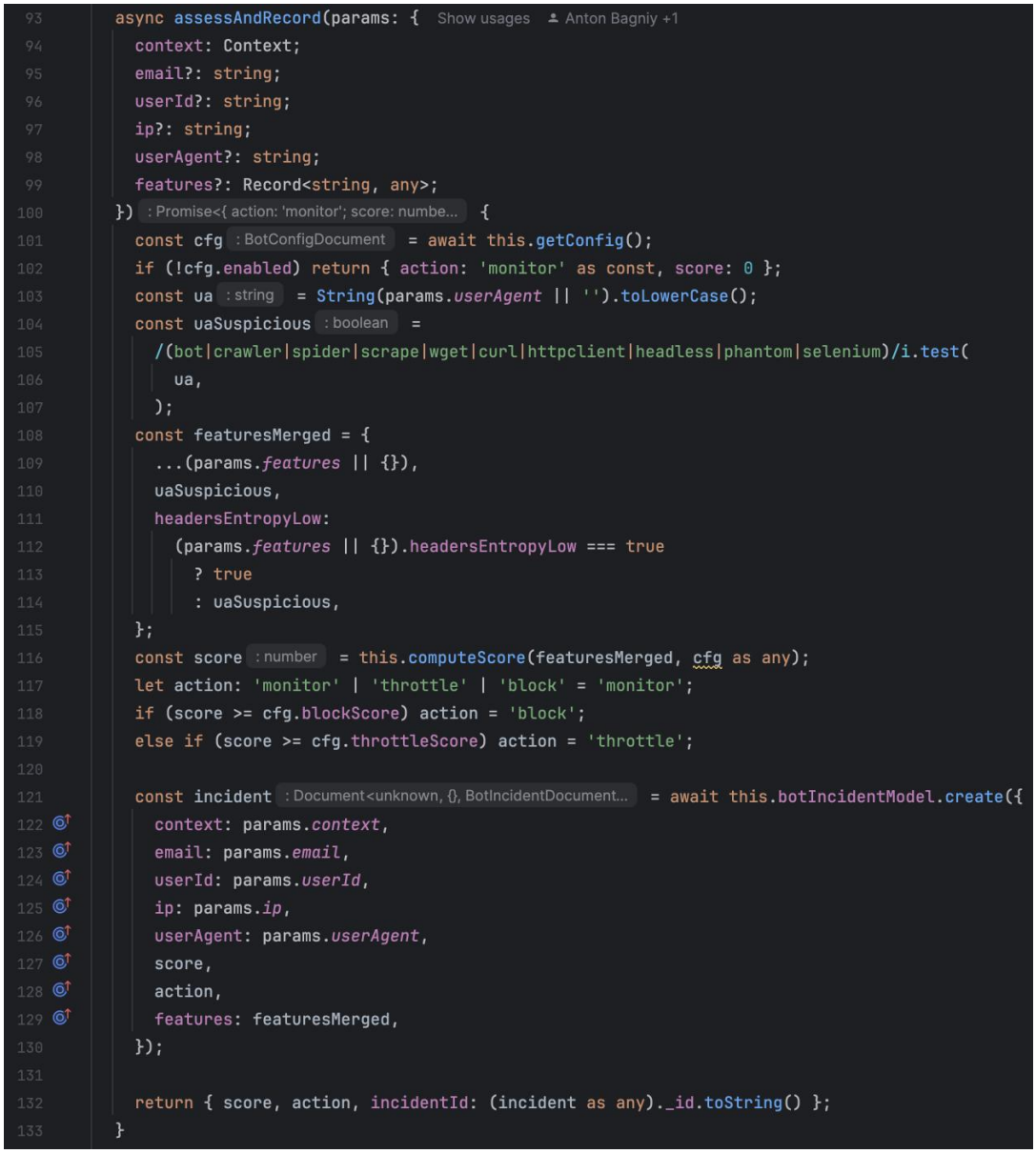
3.4 Реалізація модулів безпеки серверної частини

Система безпеки розроблена за принципом захисту в глибину і складається з семи незалежних рівнів захисту, реалізованих у модулі SecurityModule. Цей модуль є центральним компонентом системи безпеки та об'єднує всі сервіси захисту: BotDetectionService, AiDefenseService, TwoFAService, SecurityService, PasswordService та SessionService. Модуль використовує Mongoose для роботи з MongoDB та експортує всі сервіси для використання в інших модулях системи через механізм dependency injection в NestJS.

SecurityController є основним контролером, який надає RESTful API для управління системою безпеки та моніторингу інцидентів. Контролер реалізує ендпоінти для отримання статистики безпеки (GET /security/summary), управління конфігурацією bot detection та AI defense (GET/PUT /security/bot/config, GET/PUT /security/ai/config), перегляду інцидентів (GET /security/bot/incidents, GET /security/ai/incidents, GET /security/waf/incidents), управління сесіями (GET/DELETE /security/sessions/:id), налаштування двофакторної автентифікації (POST /security/2fa/setup, POST /security/2fa/verify, POST /security/2fa/enable) та управління заблокованими IP-адресами (GET/POST /security/ips/blocked, POST /security/ips/block, POST /security/ips/unblock).

BotDetectionService реалізує перший рівень захисту для виявлення автоматизованих запитів на основі поведінкового аналізу. Сервіс містить метод assessAndRecord, який отримує контекст запиту (login, order тощо), дані користувача, IP-адресу, User-Agent та об'єкт features з поведінковими даними від клієнта. Метод computeScore обчислює ризик-скор від 0 до 100 шляхом нормалізації кожної ознаки та обчислення зваженої суми. Алгоритм перевіряє підозрілі User-Agent рядки через регулярний вираз, аналізує середній інтервал між натисканнями клавіш (typingAvgInterval), кількість рухів миші (mouseMovements), час перебування на сторінці (dwellTime), наявність навігаторних API, ентропію заголовків та частоту запитів. Залежно від розрахованого рівня ризику система автоматично блокує підозрілі запити, обмежує їх частоту або працює в режимі моніторингу. Адміністратор має змогу гнучко налаштовувати вагові коефіцієнти метрик через панель управління для адаптації до нових загроз. Детальна інформація про кожен інцидент зберігається в базі даних для подальшого аудиту та аналізу.

На основі обчисленого скору система визначає дію яку потрібно виконати з цим запитом: “monitor” (якщо скор нижче throttleScore), “throttle” (якщо скор між throttleScore та blockScore) або “block” (якщо скор перевищує blockScore). Кожна оцінка записується в колекцію BotIncident зі всіма полями які аналізувались. Метод getConfig завантажує поточну конфігурацію з колекції BotConfig, а метод updateConfig дозволяє адміністратору динамічно змінювати пороги та ваги ознак без перезапуску сервера. Така гнучкість архітектури дозволяє системі миттєво адаптуватися до зміни патернів атак без необхідності перекомпіляції коду. У наведеному лістингу методу можна прослідкувати повний життєвий цикл обробки запиту: від отримання «сирих» даних клієнта до формування фінального вердикту безпеки. Використання асинхронних викликів гарантує, що складні обчислення ризику та запис логів не створюватимуть відчутних затримок для легітимних користувачів, забезпечуючи високу продуктивність системи. Реалізація методу assessAndRecord зображена на рисунку 3.3.



```
async assessAndRecord(params: {  Show usages  Anton Bagniy +1
  context: Context;
  email?: string;
  userId?: string;
  ip?: string;
  userAgent?: string;
  features?: Record<string, any>;
}) : Promise<{ action: 'monitor'; score: numbe... {
  const cfg : BotConfigDocument = await this.getConfig();
  if (!cfg.enabled) return { action: 'monitor' as const, score: 0 };
  const ua : string = String(params.userAgent || '').toLowerCase();
  const uaSuspicious : boolean =
    /(bot|crawler|spider|scrape|wget|curl|httpclient|headless|phantom|selenium)/i.test(
      ua,
    );
  const featuresMerged = {
    ...(params.features || {}),
    uaSuspicious,
    headersEntropyLow:
      (params.features || {}).headersEntropyLow === true
        ? true
        : uaSuspicious,
  };
  const score : number = this.computeScore(featuresMerged, cfg as any);
  let action: 'monitor' | 'throttle' | 'block' = 'monitor';
  if (score >= cfg.blockScore) action = 'block';
  else if (score >= cfg.throttleScore) action = 'throttle';

  const incident : Document<unknown, {}, BotIncidentDocument... = await this.botIncidentModel.create({
    context: params.context,
    email: params.email,
    userId: params.userId,
    ip: params.ip,
    userAgent: params.userAgent,
    score,
    action,
    features: featuresMerged,
  });

  return { score, action, incidentId: (incident as any)._id.toString() };
}
```

Рисунок 3.3 — Реалізація методу assessAndRecord у BotDetectionService

Другий рівень захисту реалізовано як WAF (Web Application Firewall) middleware, який перехоплює всі вхідні HTTP-запити для виявлення SQL-ін'єкцій та XSS-атак. Middleware реалізований у файлі backend/src/main.ts та автоматично підключається в main.ts через app.use(). Для виявлення SQL-ін'єкцій використовуються регулярні вирази, які виявляють ключові слова SQL команд (SELECT, UNION, DROP, INSERT, UPDATE, DELETE) у поєднанні зі спеціальними символами. Для XSS-атак перевіряється наявність тегів script, iframe, embed, обробників подій та javascript: протоколу. Основною особливістю є

подвійне URL-декодування параметрів для виявлення складних payloads. Реалізація WAF middleware з перевіркою патернів зображена на рисунку 3.4.

```
const wafPatterns : { kind: string; re: RegExp }[] = [
  { kind: 'SQLI', re: /('|%27)\s*or\s*1=1|union\s+select|sleep\s*\(/i },
  { kind: 'XSS', re: /<script|onerror=|onload=|javascript:/i },
];
app.use(async (req: any, res: any, next: any) : Promise<any> => {
  try {
    const rawUrl : string = String(req.url || '');
    let url : string = rawUrl;
    try {
      url = decodeURIComponent(rawUrl);
    } catch {}
    try {
      url = decodeURIComponent(url);
    } catch {}
    url = url.replace(/\+/g, ' ');

    const bodyStr : string =
      typeof req.body === 'object'
        ? JSON.stringify(req.body)
        : String(req.body || '');
    const str = `${url}\n${bodyStr}`;
    for (const p of wafPatterns) {
      if (p.re.test(str)) {
        try {
          const model : Model<WafIncidentDocument, {}, {}, {}, Do... = app.get<Model<WafIncidentDocument>>(
            'WafIncidentModel',
            { strict: false },
          );
          await model?.create({
            kind: p.kind,
            path: req.path,
            method: req.method,
            ip: req.ip,
            payload: req.body,
            pattern: String(p.re),
          });
        } catch {}
        return res.status(406).json({
          message: 'Request rejected by WAF',
          code: 'WAF_BLOCK',
          kind: p.kind,
        });
      }
    }
  } catch {}
```

Рисунок 3.4 — Реалізація WAF middleware у main.ts

При виявленні підозрілого патерну middleware створює запис у колекції WafIncident з типом атаки, URL запиту, HTTP методом, IP-адресою, User-Agent та payload, що спрацював. Запит одразу відхиляється з HTTP статусом 403 Forbidden. Це забезпечує активний захист на рівні застосунку до того, як запит досягне бізнес-логіки.

Третій рівень захисту забезпечує TwoFAService для реалізації двофакторної автентифікації на основі TOTP (Time-based One-Time Password). Сервіс використовує бібліотеку otplib для генерації та верифікації кодів, а також вбудований модуль crypto Node.js для шифрування секретів. Метод generateSecret створює новий 32-символьний base32 секрет та генерує otpauthUrl для QR-коду. Метод encryptSecret шифрує секрет алгоритмом AES-256-GCM: генерується випадковий 12-байтовий IV через crypto.randomBytes(12), створюється cipher об'єкт, виконується шифрування, отримується authentication tag для забезпечення цілісності, результат конкатенується та кодується в base64. Реалізація методів шифрування зображена на рисунку 3.5.

```
27    verify(code: string, secret: string): boolean {  Show usages  Anton Bagniy
28      return otplib.authenticator.check(code, secret);
29    }
30
31    encryptSecret(secret: string): string {  Show usages  Anton Bagniy +1
32      const iv : Buffer<ArrayBufferLike>  = crypto.randomBytes(12);
33      const cipher : CipherGCM  = crypto.createCipheriv(this.algorithm, this.key, iv);
34      const encrypted : Buffer<ArrayBuffer>  = Buffer.concat([
35        cipher.update(secret, 'utf8'),
36        cipher.final(),
37      ]);
38      const tag : Buffer<ArrayBufferLike>  = cipher.getAuthTag();
39      return Buffer.concat([iv, tag, encrypted]).toString('base64');
40    }
41
42    decryptSecret(enc: string): string {  Show usages  Anton Bagniy
43      const raw : Buffer<ArrayBuffer>  = Buffer.from(enc, 'base64');
44      const iv : Buffer<ArrayBuffer>  = raw.subarray(0, 12);
45      const tag : Buffer<ArrayBuffer>  = raw.subarray(12, 28);
46      const data : Buffer<ArrayBuffer>  = raw.subarray(28);
47      const decipher : DecipherGCM  = crypto.createDecipheriv(this.algorithm, this.key, iv);
48      decipher.setAuthTag(tag);
49      const dec : Buffer<ArrayBuffer>  = Buffer.concat([decipher.update(data), decipher.final()]);
50      return dec.toString('utf8');
51    }
52  }
```

Рисунок 3.5 — Реалізація шифрування секретів у TwoFAService

Метод decryptSecret виконує зворотній процес, а саме декодує base64, розділяє на IV, tag та дані, створює decipher об'єкт, встановлює authentication tag, дешифрує дані. Метод verify перевіряє валідність 6-значного TOTP коду через

otplib.authenticator.check(code, secret) з параметром window: 1 для прийняття кодів з попереднього, поточного та наступного 30-секундних інтервалів.

Четвертий рівень захисту реалізує SecurityService для контролю спроб автентифікації та захисту від brute-force атак. Метод checkRateLimit підраховує невдалі спроби входу для конкретного email за останні 15 хвилин через attemptModel.countDocuments з фільтром за email. Якщо кількість перевищує поріг (5 за замовчуванням), метод повертає об'єкт {limited: true, retryAt: Date}, рисунок 3.6.

```
async recordAttempt(params: { email: string; userId?: string; success: boolean; ip?: string; userAgent?: string; reason?: string; })
  const { email, userId, success, ip, userAgent, reason } = params;
  await this.attemptModel.create({email, userId, success, ip, userAgent, reason,});
  if (success) {
    await this.lockModel.deleteOne({ email });
    return;
  }
  if (reason === 'rate_limit') return;
  const lock : (Document<unknown, {}, AccountLockDocumen... = await this.lockModel.findOne({ email });
  if (!lock) {
    await this.lockModel.create({ email, failedCount: 1 });
    return;
  }
  lock.failedCount += 1;
  if (lock.failedCount >= MAX_FAILED_ATTEMPTS) {
    const until = new Date(Date.now() + LOCK_MINUTES * 60 * 1000);
    lock.lockedUntil = until;
  }
  await lock.save();
}

async checkRateLimit(email: string, ip?: string) : Promise<{ limited: boolean; retryAt?: und... {  Show usages  Anton Bagniy *
  if (!ip) return { limited: false };
  const since = new Date(Date.now() - RATE_LIMIT_WINDOW_SEC * 1000);
  const count : number  = await this.attemptModel.countDocuments({createdAt: { $gte: since }, ip});
  if (count >= RATE_LIMIT_MAX_ATTEMPTS) {
    const retryAt = new Date(since.getTime() + RATE_LIMIT_WINDOW_SEC * 1000);
    return { limited: true, retryAt };
  }
  return { limited: false };
}
```

Рисунок 3.6 — Реалізація rate limiting у SecurityService

Метод autoBanIfAbusive аналізує поведінку IP-адреси: підраховує кількість унікальних email адрес, на які була спроба входу з цієї IP за останню годину, і якщо кількість перевищує 10, викликає blockIp для додавання IP до чорного списку. Реалізація методів зображена на рисунку 3.7.

```
171   async autoBanIfAbusive(ip?: string) {  Show usages  Anton Bagniy +1
172     if (!ip) return;
173     const since = new Date(Date.now() - AUTO_BAN_WINDOW_MIN * 60 * 1000);
174     const failedFromIp = await this.attemptModel.countDocuments({
175       ip,
176       success: false,
177       createdAt: { $gte: since },
178     });
179     if (failedFromIp >= AUTO_BAN_THRESHOLD) {
180       await this.blockIp(
181         ip,
182         AUTO_BAN_MINUTES,
183         `auto-ban: ${failedFromIp} failed in ${AUTO_BAN_WINDOW_MIN}m`,
184       );
185     }
186   }
187 }
```

Рисунок 3.7 — Реалізація методу автоматичного блокування IP

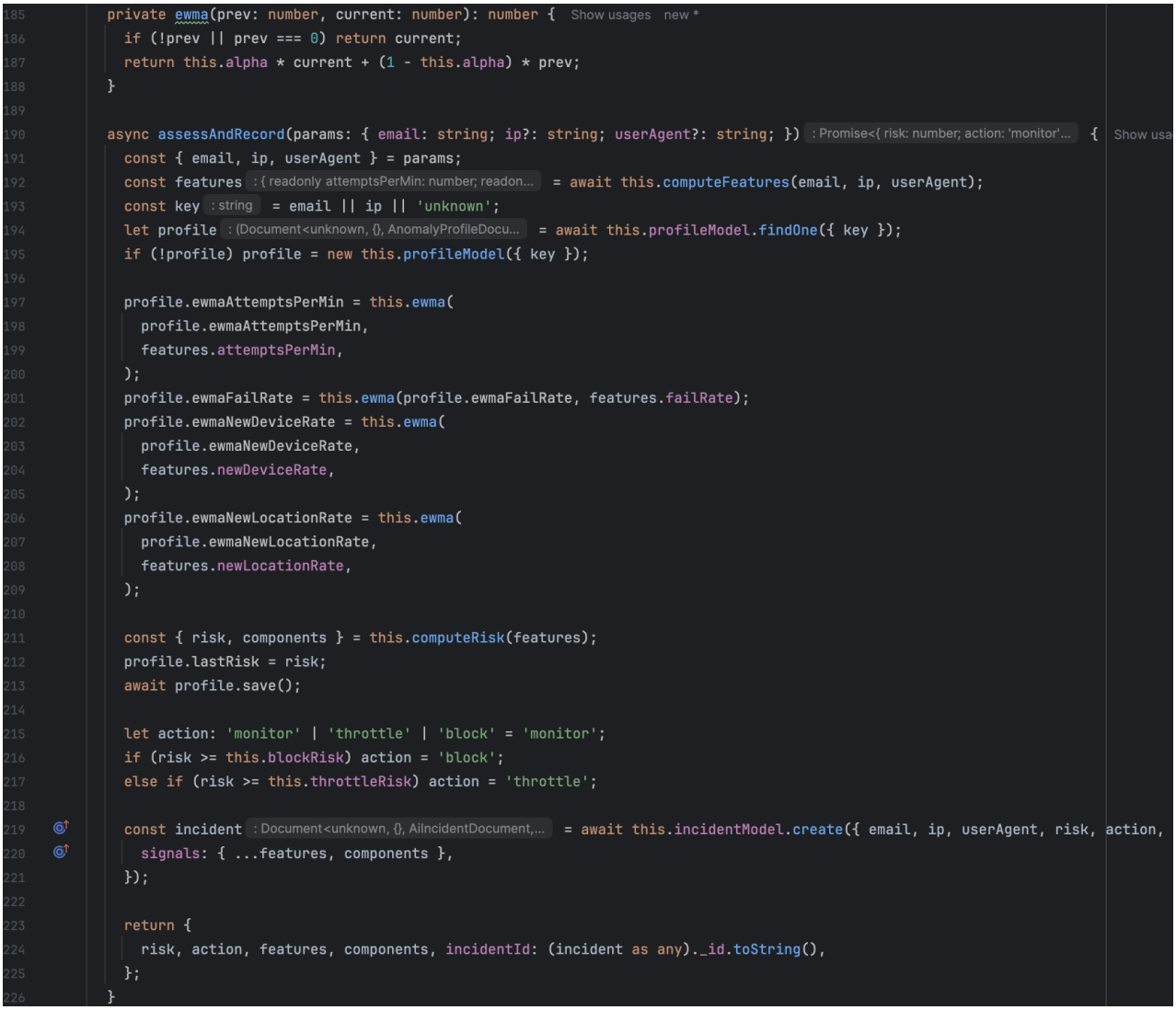
П'ятий рівень реалізує SessionService для управління користувацькими сесіями з прив'язкою JWT-токенів до серверних записів. При успішній автентифікації метод createSession створює новий запис у колекції LoginSession з userId, IP-адресою, User-Agent, унікальним sessionId, статусом isActive: true та lastActivityAt: new Date(). Згенерований sessionId вбудовується у payload JWT-токена. При кожному запиті JwtAuthGuard через JwtStrategy викликає метод validate, який перевіряє не тільки валідність JWT токену, але й існування активної сесії.

Метод validateSession виконує запит до LoginSession за userId та sessionId з умовою isActive: true. Такий гібридний підхід дозволяє нівелювати основний недолік класичної технології JWT — неможливість миттєвого відкликання доступу до закінчення терміну дії токена. Завдяки перевірці статусу сесії в базі даних при кожному запиті, система може миттєво заблокувати скомпрометованого користувача або конкретний пристрій. Крім того, збереження метаданих (User-Agent, IP) створює підґрунтя для реалізації функції моніторингу активних сесій, надаючи користувачам прозорий інструмент контролю безпеки власного облікового запису. Реалізація методів зображена на рисунку 3.8.

```
async recordSession(params: { userId: string; email: string; ip?: string; userAgent?: string;
}): Promise<{ isNewDevice: boolean; isNewLocation: boolean; sessionId: string; }> {
  const { userId, email, ip, userAgent } = params;

  const geo : { country?: string | undefined; city?: st... = ip ? await this.getGeoLocation(ip) : {};

  const existingSession : (Document<unknown, {}, LoginSessionDocume... = await this.sessionModel.findOne({
    userId: new Types.ObjectId(userId),
    userAgent,
    isActive: true,
  });

  const existingSessions : (Document<unknown, {}, LoginSessionDocume... = await this.sessionModel.find({
    userId: new Types.ObjectId(userId),
    isActive: true,
  });

  const isNewDevice = true;
  let isNewLocation : boolean = true;

  for (const session of existingSessions) {
    if (session.country === geo.country && session.city === geo.city) {
      isNewLocation = false;
    }
  }

  const created : Document<unknown, {}, LoginSessionDocumen... = await this.sessionModel.create({
    userId: new Types.ObjectId(userId),
    email,
    ip,
    userAgent,
    country: geo.country,
    city: geo.city,
    region: geo.region,
    isActive: true,
    lastActivityAt: new Date(),
    isNewDevice,
    isNewLocation,
  });

  return {
    isNewDevice,
    isNewLocation,
    sessionId: (created as any)._id.toString(),
  };
```

Рисунок 3.8 — Реалізація управління сесіями у SessionService

Шостий рівень захисту впроваджує AiDefenseService для адаптивного управління ризиками на основі EWMA (Exponentially Weighted Moving Average). Сервіс аналізує поведінку користувача, виявляючи аномалії відносно baseline профілю. Метод assessAndRecord завантажує або створює AnomalyProfile для користувача, обчислює поточні features (attemptRate, failRate, currentStreak, noveltyIp, noveltyUserAgent, shortBurst), порівнює їх з baseline та оновлює профіль. Risk score обчислюється як зважена сума відхилень. Реалізація алгоритму зображена на рисунку 3.9.



```
185   private ewma(prev: number, current: number): number {  Show usages  new *
186     if (!prev || prev === 0) return current;
187     return this.alpha * current + (1 - this.alpha) * prev;
188   }
189
190   async assessAndRecord(params: { email: string; ip?: string; userAgent?: string; }) : Promise<{ risk: number; action: 'monitor'... {  Show usa
191     const { email, ip, userAgent } = params;
192     const features : { readonly attemptsPerMin: number; readon... = await this.computeFeatures(email, ip, userAgent);
193     const key : string = email || ip || 'unknown';
194     let profile : (Document<unknown, {}, AnomalyProfileDocu... = await this.profileModel.findOne({ key });
195     if (!profile) profile = new this.profileModel({ key });
196
197     profile.ewmaAttemptsPerMin = this.ewma(
198       profile.ewmaAttemptsPerMin,
199       features.attemptsPerMin,
200     );
201     profile.ewmaFailRate = this.ewma(profile.ewmaFailRate, features.failRate);
202     profile.ewmaNewDeviceRate = this.ewma(
203       profile.ewmaNewDeviceRate,
204       features.newDeviceRate,
205     );
206     profile.ewmaNewLocationRate = this.ewma(
207       profile.ewmaNewLocationRate,
208       features.newLocationRate,
209     );
210
211     const { risk, components } = this.computeRisk(features);
212     profile.lastRisk = risk;
213     await profile.save();
214
215     let action: 'monitor' | 'throttle' | 'block' = 'monitor';
216     if (risk >= this.blockRisk) action = 'block';
217     else if (risk >= this.throttleRisk) action = 'throttle';
218
219     const incident : Document<unknown, {}, AiIncidentDocument,... = await this.incidentModel.create({ email, ip, userAgent, risk, action,
220       signals: { ...features, components },
221     });
222
223     return {
224       risk, action, features, components, incidentId: (incident as any)._id.toString(),
225     };
226   }
```

Рисунок 3.9 — Реалізація EWMA-алгоритму в AiDefenseService

Сьомий рівень захисту забезпечує Content Security Policy через конфігурацію Next.js та Helmet middleware. У файлі frontend/next.config.js визначена секція headers з директивами CSP: script-src обмежує джерела JavaScript до “self” та “unsafe-inline”, style-src дозволяє “self” та “unsafe-inline”, img-src дозволяє “self”, data: та res.cloudinary.com, connect-src обмежує API запити до “self” та backend URL, frame-ancestors встановлено як “none”. На бекенді в main.ts підключається Helmet middleware через app.use(helmet()), який автоматично встановлює безпечні HTTP-заголовки: X-Content-Type-Options: nosniff, X-Frame-Options: DENY, Strict-Transport-Security, X-XSS-Protection: 1; mode=block.

Інтеграція всіх рівнів захисту відбувається через послідовні перевірки в ендпоінті POST /auth/login контролера AuthController. Спочатку перевіряється

наявність step-up challenge, потім виконується bot detection з обчисленням score та прийняттям рішення про block/throttle/step-up, далі перевіряється блокування IP, виконується AI defense, перевіряється rate limiting, перевіряється блокування облікового запису, виконується валідація credentials, опціонально перевіряється 2FA код, і нарешті створюється сесія та генерується JWT токен. Кожна перевірка може відхилити запит на своєму рівні, забезпечуючи багаторівневий захист.

3.5 Реалізація системи рекомендацій на основі матричної факторизації

Система рекомендацій реалізована гібридним підходом, що поєднує три методи: контентну фільтрацію, колаборативну фільтрацію та AI-рекомендації на основі матричної факторизації. Ядром системи є клас AIRecoService (backend/src/modules/products/ai-reco.service.ts. Цей клас реалізує алгоритм матричної факторизації через стохастичний градієнтний спуск без використання готових ML-бібліотек.

Метод trainFromOrders завантажує історичні замовлення та створює вектори розмірності dim=16 для кожного користувача та товару. Реалізація зображена на рисунку 3.10.

```
const users : string[]  = Array.from(userSet);
const products : string[]  = Array.from(productSet);
const uVec: Record<string, number[]> = {};
const pVec: Record<string, number[]> = {};
const rand : () => number  = () : number  => (Math.random() - 0.5) * 0.2;  Show usage
for (const u of users) uVec[u] = Array.from({ length: this.dim }, rand);
for (const p of products) pVec[p] = Array.from({ length: this.dim }, rand);
```

Рисунок 3.10 — Ініціалізація векторних представлень

Спочатку відбувається ініціалізація векторів, потім створюються об'єкти uVec та pVec типу Record<string, number[]>. Наступним кроком функція rand

генерує випадкові значення в діапазоні [-0.1, 0.1]. Для кожного користувача та товару створюється масив довжиною this.dim (16 компонентів) заповнений випадковими значеннями через Array.from. Завдяки центруванню значень навколо нуля забезпечується стабільна збіжність градієнтного спуску.

Навчання виконується методом step, що реалізує одну ітерацію SGD. Реалізація зображена на рисунку 3.11.

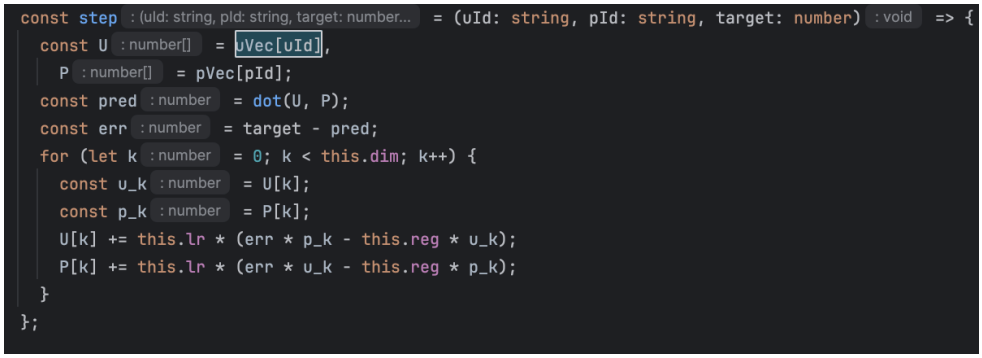


Рисунок 3.11 — Оновлення параметрів стохастичним градієнтним спуском

На рисунку видно метод step в якому отримуються вектори користувача U та товару P з об'єктів uVec та pVec, обчислюється передбачення через скалярний добуток dot(U, P). Похибка err тут це різниця між цільовим значенням target та передбаченням. У циклі по всіх k компонентах векторів виконується оновлення за правилами градієнтного спуску. Параметри: this.lr=0.02 (швидкість навчання), this.reg=0.02 (регуляризація), this.epochs=5.

Після навчання векторні представлення зберігаються в MongoDB через модель embModel. Реалізація зберігання векторів зображена на рисунку 3.12.

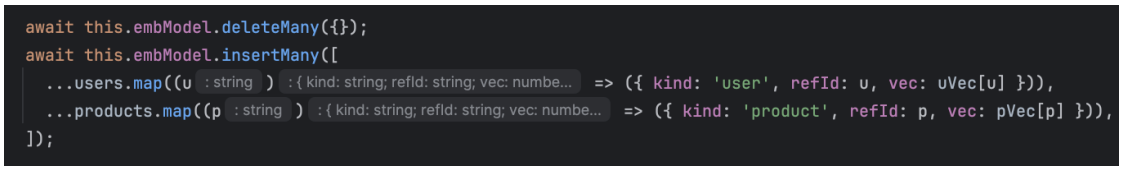


Рисунок 3.12 — Збереження векторних представлень у базу даних

Спочатку видаляються всі існуючі записи через await this.embModel.deleteMany({}). Потім виконується одна операція insertMany, яка приймає об’єднаний масив. Для користувачів створюються документи з полями kind='user', refId (ID користувача) та vec (вектор). Аналогічно для товарів з kind='product'. Така структура дозволяє створити індекси по полях kind та refId для швидкого пошуку при генерації рекомендацій.

Метод scoreForUser використовує збережені embeddings для обчислення скорів релевантності товарів які використовуються для генерації персоналізованих рекомендацій. Реалізація зображена на рисунку 3.13.

```
async scoreForUser(  no usages  Anton Bagniy +1
  userId: string,
  candidateProductIds: string[],
): Promise<Record<string, number>> {
  const embs : (FlattenMaps<AiRecoEmbeddingDocument> & R...  = await this.embModel
    .find({
      $or: [
        { kind: 'user', refId: userId },
        { kind: 'product', refId: { $in: candidateProductIds } },
      ],
    })
    .lean();
  const u : (FlattenMaps<AiRecoEmbeddingDocument> & R...  = embs.find((e : FlattenMaps<AiRecoEmbeddingDocument> & Re... ) : boolean  => e.kind === 'user' && e.refId === userId);
  if (!u) return {};
  const pMap: Record<string, number[]> = {};
  for (const e of embs)
    if (e.kind === 'product') pMap[e.refId] = e.vec as any;
  const dot : (a: number[], b: number[]) => number  = (a: number[], b: number[]) : number  =>  Show usages  Anton Bagniy
    a.reduce((s : number , v : number , i : number ) : number  => s + v * b[i], 0);
  const scores: Record<string, number> = {};
  for (const pid of candidateProductIds) {
    const pv : number[]  = pMap[pid];
    if (!pv) continue;
    scores[pid] = dot(u.vec as any, pv);
  }
  return scores;
}
```

Рисунок 3.13 — Генерація рекомендацій через скалярний добуток

Одним запитом завантажуються embedding користувача та всіх товарів-кандидатів через оператор $or з двома умовами: { kind: 'user', refId: userId } та { kind: 'product', refId: { $in: candidateProductIds } }. Метод lean() повертає прості JavaScript об'єкти замість Mongoose документів для підвищення продуктивності. Через find шукається вектор користувача u. Якщо не знайдено, повертається порожній об'єкт {} (fallback). Будується об’єкт pMap типу Record<string, number[]> що відображає productId на його вектор. Для кожного товару-кандидата

обчислюється скор через скалярний добуток dot(u.vec, pv). Результат — об'єкт scores, де ключі — productId, значення — скори релевантності.

Метод similarProducts класу ProductsService реалізує контентну фільтрацію. Реалізація зображена на рисунку 3.14.

```
async similarProducts(productIdOrSlug: string, limit = 10) : Promise<any[]> {  Show usages
  const base : (FlattenMaps<ProductDocument> & Required<... = productIdOrSlug.includes('-')
    ? await this.productModel.findOne({ slug: productIdOrSlug }).lean()
    : await this.productModel.findById(productIdOrSlug).lean();
  if (!base) return [];

  const tokens = new Set<string>();
  const pushTokens : (s?: string | undefined) => void = (s?: string) : void => {  Show usages  Ant
    if (!s) return;
    s.split(/[\s,#/|]+/)
      .map((t : string ) : string => t.trim().toLowerCase())
      .filter(Boolean)
      .forEach((t : string ) : Set<string> => tokens.add(t));
  };
  pushTokens(base.category);
  pushTokens(base.brand);
  pushTokens(base.colors);
  pushTokens(base.sizes);
  pushTokens(base.description);

  const candidates : (FlattenMaps<ProductDocument> & Required<... = await this.productModel
    .find({
      _id: { $ne: base._id },
      $or: [{ category: base.category }, { brand: base.brand }],
    })
    .lean();

  const score : (p: any) => number = (p: any) : number => {  Show usages  Anton Bagniy
    let s : number = 0;
    const weight : (w: number, cond: boolean) => void = (w: number, cond: boolean) : void => {
      if (cond) s += w;
    };
    const str : string =
      `${p.brand} ${p.category} ${p.colors} ${p.sizes} ${p.description}`.toLowerCase();
    let overlaps : number = 0;
    tokens.forEach((t : string ) : void => {
      if (t && str.includes(t)) overlaps++;
    });
    s += Math.min(5, overlaps);
    const priceDiff : number = Math.abs((p.price || 0) - (base.price || 0));
    if (base.price > 0) s += Math.max(0, 3 - (priceDiff / base.price) * 3);
    s += (p.rating || 0) * 0.5 + Math.min(3, (p.numReviews || 0) / 50);
    weight(2, p.brand === base.brand);
    weight(2, p.category === base.category);
    return s;
  };
```

Рисунок 3.14 — Контентна фільтрація через токенізацію текстових полів

Для початку створюється Set<string> для токенів базового товару. Функція pushTokens виконує токенізацію: розбиває рядок за роздільниками (пробіли, коми,

хеші, слеші, вертикальні риски), нормалізує до нижнього регістру через toLowerCase(), фільтрує порожні значення через filter(Boolean) та додає до Set. Токенізуються такі поля товару: category, brand, colors, sizes, description. Функція score обчислює подібність товару-кандидата p: конкатенує його текстові поля в один рядок str, рахує overlaps — кількість спільних токенів з базовим товаром через перевірку includes(t), додає до скору Math.min(5, overlaps) обмежуючи вплив до 5 балів. Додається ціновий компонент: якщо priceDiff < base.price, додається до 3 балів. Також враховується рейтинг. Цей підхід не потребує історії взаємодій.

Фінальні рекомендації формуються об'єднанням результатів трьох методів у ProductsService. Реалізація об’єднання зображена на рисунку 3.15.

```
  const map = new Map<string, { p: any; s: number }>();
  content.forEach((p: any, i: number) => {
    const id = String(p._id);
    const prev = map.get(id)?.s || 0;
    map.set(id, { p, s: prev + 0.4 * (1 / (i + 1)) });
  });
  cf.forEach(({ p }: any, i: number) => {
    if (!p?._id) return;
    const id = String(p._id);
    const prev = map.get(id)?.s || 0;
    map.set(id, { p, s: prev + 0.4 * (1 / (i + 1)) });
  });
  for (const [id, entry] of Array.from(map.entries())) {
    const ai = aiScores[id] ?? 0;
    entry.s += 0.2 * ai;
    map.set(id, entry);
  }
  return Array.from(map.values())
    .sort((a, b) => b.s - a.s)
    .map((x) => x.p);
}),
```

Рисунок 3.15 — Rank-based blending з вагами 0.4/0.4/0.2

Використовується Map<string, { p: any; s: number }> для акумуляції скорів, де ключ — productId, значення — об’єкт з товаром p та скором s. Для контентної фільтрації обчислюється зважений скор. Через map.get(id)?.s || 0 отримується поточний скор або 0, до нього додається новий вклад. Аналогічно для

колаборативної фільтрації. Якщо товар присутній у кількох списках, його скори акумулюються. Ваги (0.4, 0.4, 0.2) обрані експериментально і відображають компроміс між стабільністю контентного підходу, точністю колаборативного та додатковою інформацією від матричної факторизації.

Реалізована система рекомендацій повністю функціональна та готова до production використання. Модульна архітектура дозволяє незалежно налаштовувати кожен метод та динамічно змінювати ваги у гібридному об'єднанні. Система автоматично адаптується до холодного старту користувачів через fallback на популярні товари та до холодного старту товарів через контентну фільтрацію, яка не залежить від історичних даних.

3.6 Реалізація клієнтської частини та адміністративної панелі

Клієнтська частина системи реалізує інтеграцію з backend модулями безпеки та персоналізації. Основні компоненти: BotSensors для збору поведінкових даних, LoginForm для передачі features на сервер, PersonalRecommendations для відображення індивідуальних рекомендацій, та SecurityDashboard для адміністративного моніторингу. Всі компоненти побудовані як React функціональні компоненти з використанням хуків для управління станом та життєвим циклом.

Глобальний компонент BotSensors монтується в ClientProviders та працює протягом всієї сесії. При завантаженні додатку фіксується timestamp у window.__page_enter_ts через Date.now(). Event listener на mousemove інкрементує лічильник window.__mouse_moves. Event listener на keydown записує timestamps натискань клавіш у масив, з якого обчислюється середній інтервал window.__typing_avg. При відправці форми автентифікації компонент LoginForm формує об'єкт features. Реалізація зображена на рисунку 3.16.

Об'єкт features містить сім метрик: dwellTimeMs (різниця між поточним часом та часом завантаження сторінки), mouseMoveCount (кількість зареєстрованих рухів миші), typingIntervalAvg (середній інтервал між

натисканнями клавіш в мілісекундах або null якщо не було введення), navigatorHeadless та webdriver (перевірка navigator.webdriver === true для детекції headless браузерів), languages (довжина масиву navigator.languages), hasTouch (наявність touch events через перевірку “ontouchstart” in window).

```
const handleSubmit = async (e: React.FormEvent) => {
  e.preventDefault();
  e.stopPropagation();
  setError('');

  const startTime = (window as any).__page_enter_ts || Date.now() - 1000;
  const dwellTimeMs = Date.now() - startTime;
  const features = {
    dwellTimeMs,
    mouseMoveCount: (window as any).__mouse_moves || 0,
    typingIntervalAvg: (window as any).__typing_avg || null,
    navigatorHeadless: (navigator as any).webdriver === true,
    webdriver: (navigator as any).webdriver === true,
    languages: Array.isArray(navigator.languages)
      ? navigator.languages.length
      : 0,
    hasTouch: 'ontouchstart' in window,
  };
  const botHeader = btoa(JSON.stringify(features));
  const result = await login(
    email,
    password,
    stepUp?.id,
    botHeader,
    features,
  );
```

Рисунок 3.16 — Формування об'єкту поведінкових метрик

Функція btoa() кодує JSON в Base64 рядок. Цей рядок передається як у заголовку X-Bot-Features через параметр botHeader, так і в полі botFeaturesDecoded тіла запиту для зручності debugging на backend. Метод login() з useAuth хука виконує POST запит до /auth/login з цими даними. Якщо backend повертає об'єкт з error що містить STEP_UP_REQUIRED, виконується fetch до /api/security/stepup/start для отримання challenge. Відповідь містить id та question. Ці дані зберігаються в стані setStepUp, що тригерить рендеринг модального вікна з полем для введення відповіді. Після відправки відповіді виконується POST до

/api/security/stepup/verify з stepUpId та answer. При успішній верифікації challenge позначається як used у базі, і повторюється оригінальний запит login() вже з stepUp.id у параметрах, що дозволяє backend пройти перевірку на бота.

Компонент PersonalRecommendations використовує хук useAuth() для отримання JWT токену. UseEffect хук з dependency array [limit, token] виконує асинхронну функцію run() при зміні цих значень. Реалізація зображена на рисунку 3.17.

```
export default function PersonalRecommendations({  Show usages  Anton Bagniy
  lang,
  title = 'Персональні рекомендації',
  limit = 8,
}: Props) : Element | null  {
  const [items, setItems] = React.useState<Product[]>([]);
  const [loaded, setLoaded] = React.useState(false);
  const { token, isLoading } = useAuth();

  React.useEffect(() : () => void  => {
    let cancelled : boolean  = false;
    const run : () => Promise<void>  = async () : Promise<void>  => {  Show usages
      try {
        if (!token) return;
        apiClient.setToken(token);
        const data = (await apiClient.getMyRecommendations(
          limit,
        )) as unknown as Product[];
        if (!cancelled) setItems(Array.isArray(data) ? data : []);
      } catch {
        if (!cancelled) setItems([]);
      } finally {
        if (!cancelled) setLoaded(true);
      }
    };
    run();
    return () : void  => {
      cancelled = true;
    };
  }, [limit, token]);

  if (isLoading || !token || !loaded || items.length === 0) return null;
```

Рисунок 3.17 — Завантаження персоналізованих рекомендацій

Прапорець cancelled використовується для запобігання race condition при unmount компонента. Перевірка if (!token) виходить з функції якщо користувач не автентифікований. Метод apiClient.setToken(token) встановлює заголовок Authorization: Bearer <token> для всіх наступних запитів. Виклик getMyRecommendations(limit) виконує GET запит до /products/recommendations/for/me?limit=8. Backend повертає масив Product об'єктів, згенерованих методом recommendedForUser() з ProductsService. Перевірка Array.isArray() захищає від некоректних відповідей. Finally блок встановлює loaded=true незалежно від результату. Умова if (isLoading || !token || !loaded || items.length === 0) return null запобігає рендерингу якщо: автентифікація ще виконується, токен відсутній, дані не завантажені, або масив порожній. Cleanup функція return () => { cancelled = true } виконується при unmount та скасовує pending оновлення стану.

На сторінці товару компонент Product.tsx містить useEffect для запису перегляду. При mount виконується awaitapiClient.recordProductView(String(product._id)), що робить POST до /products/:id/view. Backend створює запис у ProductView колекції з userId, productId, ip, userAgent, timestamp. Ці дані використовуються методом recommendedForUser() для аналізу recentViews — останніх 30 переглядів сортуються за createdAt: -1, відбираються унікальні productId через Set, і для кожного генеруються similar products через similarProducts() та collaborativeSimilar().

Результати об'єднуються через Map для уникнення дублікації та зважуються: 0.4 для контентної, 0.4 для колаборативної, 0.2 для AI-скорів. Це формує viewedSimilar масив, який додається до фінальних рекомендацій з вагою 0.3, тоді як purchaseSimilar (товари схожі на раніше куплені) отримують вагу 0.7. Якщо загальна кількість рекомендацій менша за limit, додаються популярні товари з isFeatured: true як backfill з мінімальною вагою 0.1.

Компонент SecurityDashboard завантажує статистику через useSWR з endpoint /api/admin/security/summary. Параметр refreshInterval: 30000 налаштовує автоматичне оновлення кожні 30 секунд. Підкомпонент WafTestWidget реалізує інтерфейс для тестування WAF. Реалізація зображена на рисунку 3.18.

```tsx
function WafTestWidget() {  Show usages  Anton Bagniy +1
  const [out, setOut] = React.useState<string>('');

  const run = async (kind: 'sqli' | 'xss') => {  Show usages  Anton Bagniy +1
    setOut('Running...');
    try {
      const resp = await fetch(`/api/admin/security/waf/${kind}`, {
        method: 'POST',
      });
      const data = await resp.json().catch(() => ({}));
      setOut(
        `${resp.status} ${data?.code || ''} ${data?.kind ? '(' + data.kind + ')' : ''}`,
      );
    } catch (e: any) {
      setOut(`ERROR ${e?.message || ''}`);
    }
  };

  return (
    <div className="space-y-2">
      <div className="flex gap-2 items-center">
        <button
          onClick={() => run('sqli')}
          className="bg-indigo-600 text-white px-3 py-1 rounded"
        >
          Test SQLi
        </button>
        <button
          onClick={() => run('xss')}
          className="bg-indigo-600 text-white px-3 py-1 rounded"
        >
          Test XSS
        </button>
      </div>
      {out && <div className="text-sm font-mono">{out}</div>}
    </div>
  );
}
```

Рисунок 3.18 — Тестування WAF через адміністративний інтерфейс

Об’єкт payloads містить тестові рядки для двох типів атак: SQL-ін'єкція через payload "' OR 1=1--" та XSS через payload "<script>alert(\"XSS\")</script>". Параметр kind передається як union type 'sqli' | 'xss'. Fetch виконує POST запит до Next.js API route /api/waf-test, який проксіює запит на backend /security/waf-test. Backend пропускає payload через WAF middleware, що виконує regex перевірки. Якщо виявлено підозрілий патерн, повертається { blocked: true, kind: 'sqli' | 'xss' }. Якщо payload пройшов — { blocked: false }. Template literal формує рядок виду "403 BLOCKED (sqli)" або "200 PASSED". Try-catch блок обробляє мережеві помилки та показує "ERROR <message>". Стан out зберігається через useState та відображається в div з моноширинним шрифтом для легкої читабельності

технічних даних. Окремий компонент BotDemoButtons реалізує тестування bot detection. При натисканні runLoginBot або runOrderBot виконується POST до /api/ai-demo/bot/login або /api/ai-demo/bot/order. Ці Next.js API routes створюють тимчасового користувача з email bot_login_test_<timestamp>@example.com, виконують запит з підозрілими features (dwellTimeMs: 150-200, mouseMoveCount: 0, webdriver: true), отримують відповідь від backend bot detection, та видаляють тестового користувача через cleanup endpoint. Відповідь містить status код та code поле (BOT_BLOCK, THROTTLE_DELAY тощо), що відображається в інтерфейсі.

Таблиці BotIncidentsTable та AiIncidentsTable завантажують дані через useSWR з /api/admin/security/bot-incidents та /api/admin/security/ai-incidents відповідно, показуючи останні 200 записів відсортовані за createdAt: -1. Кожен запис містить context (login або order), email, userId, score (0-100), action (monitor, throttle, block), timestamp та features об'єкт з поведінковими метриками. Таблиця рендериться через map() з key={incident._id} для React reconciliation. Компонент AiConfigForm дозволяє динамічно змінювати параметри bot detection без перезапуску сервера. При submit форми виконується PATCH запит до /api/admin/security/bot-config з новими значеннями blockScore, throttleScore, stepUpScore, wTyping, wMouse, wDwell, wNavigator, wHeaders, wRate, bias. Backend оновлює BotConfig документ через findOneAndUpdate з опцією upsert: true та інвалідує кеш через this.cachedConfig = null. Наступний виклик getConfig() завантажить нові параметри, які одразу застосовуються до всіх нових запитів.

Реалізована клієнтська архітектура забезпечує тісну інтеграцію з backend системами через REST API. Збір поведінкових даних виконується через глобальні event listeners без блокування UI. Передача features відбувається через Base64 кодування у заголовках та дублювання в body для надійності. PersonalRecommendations використовує cleanup pattern для запобігання memory leaks при unmount. Адміністративна панель надає real-time моніторинг через SWR з автооновленням та інтерактивне тестування WAF і bot detection. Всі компоненти написані на TypeScript з строгою типізацією, що забезпечує type safety та автодоповнення в IDE. React hooks (useState, useEffect, useAuth) спрощують управління станом та side effects. Використання Next.js API routes дозволяє

створювати проміжні endpoints для тестування без експозиції backend напряму. Система готова до production використання з мінімальними змінами конфігурації.

Висновки до розділу 3

У третьому розділі детально описано архітектуру та програмну реалізацію розробленої системи. Представлено модульну структуру backend на базі NestJS, де застосовано принципи SOLID та dependency injection для забезпечення низької залежності між компонентами. Реалізовано семирівневу систему захисту, що включає bot detection через аналіз поведінкових метрик, Web Application Firewall для виявлення атак, двофакторну автентифікацію на базі TOTP, виявлення аномалій через експоненціально зважену ковзну середню, управління сесіями, перевірку паролів та обмеження швидкості запитів.

Система рекомендацій реалізована як гібридний підхід, створений без використання готових ML-бібліотек. У використаному підході було поєднано контентну фільтрацію через токенізацію характеристик товару, колаборативну фільтрацію, та AI-рекомендації на основі матричної факторизації з використанням власної реалізації стохастичного градієнтного спуску [30]. Результати трьох методів об’єднуються через rank-based blending. Така архітектура забезпечує адаптацію до холодного старту через fallback механізми.

Клієнтська частина розроблена на базі Next.js 14 з використанням React Server Components. Реалізовано автоматичний збір поведінкових даних користувача через глобальні слухачі подій, відображення персоналізованих рекомендацій з автоматичним відслідковуванням переглядів товарів, та адміністративну панель з моніторингом в реальному часі безпекових інцидентів та можливістю динамічної зміни параметрів, які зберігаються в базі даних. Розроблена система демонструє модульність, використовує TypeScript для легшої розробки, та готова до виробничого використання з можливістю горизонтального масштабування.

4 ВИКОРИСТАННЯ ТА ТЕСТУВАННЯ СИСТЕМИ АДАПТИВНОГО ПІДБОРУ ТА ЗАХИСТУ ДАНИХ ВЕБ-ПОРТАЛІВ

У четвертому розділі розглядаються практичні аспекти роботи з розробленою системою, а також перевірка її функціональності за допомогою комплексного тестування. Розглядається процес автентифікації користувачів з можливістю налаштування додаткових систем безпеки, таких як двофакторна автентифікація за допомогою атрибутів автентифікації та вирішення задачі при виявленні підозрілої активності. Як працює система персоналізованих рекомендацій, яка автоматично змінює свої рекомендації на основі історії переглядів і покупок користувача.

Окрема увага приділяється можливостям адміністративної панелі моніторингу безпеки. Адміністратори можуть переглядати детальну статистику інцидентів ботів і штучного інтелекту з розбивкою за типами дій. Вони також мають можливість динамічно змінювати параметри систем захисту без перезапуску сервера, а також інтерфейс тестування WAF, який перевіряє ефективність захисту від SQL-ін’єкцій і XSS-атак. Крім того, розглядається можливість переглядати активні сесії та історії безпекових інцидентів у розділі управління користувачами.

У заключній частині розділу містяться результати комплексного тестування кожного основного модуля системи. Модульне тестування компонентів автентифікації, двофакторної автентифікації, виявлення ботів і системи рекомендацій підтвердило коректність реалізації та готовність системи до виробничого використання.

4.1 Вимоги до обчислювальної техніки

Розроблене програмне забезпечення не має жорстких вимог до обчислювальної техніки та може функціонувати на широкому спектрі апаратних

конфігурацій. Система побудована з використанням сучасних веб-технологій та оптимізована для роботи на серверах з помірними характеристиками, тому використовувати розроблене програмне забезпечення можна навіть на слабких комп'ютерах.

Проте, для серверної частини рекомендується використання будь-якого сервера або віртуальної машини з процесором частотою від 1.5 GHz та оперативною пам'яттю від 1 GB. Практика показує, що навіть базові сервери успішно справляються з навантаженням малих та середніх проєктів. Дисковий простір залежить від обсягу даних, але для початкової роботи достатньо 5-10 GB. Система підтримує всі популярні операційні системи: Linux (Ubuntu, Debian, CentOS), Windows Server та macOS, оскільки є веб рішенням.

Для запуску проєкту локально необхідно встановити Node.js версії 18 або новішої та MongoDB версії 6.0 або новішої. Обидва компоненти мають офіційні інсталяційні пакети для всіх підтримуваних платформ, тому це не є проблемою. Альтернативно, MongoDB може використовуватися як хмарний сервіс MongoDB Atlas, що усуває необхідність локального встановлення бази даних.

Клієнтська частина реалізована як веб-застосунок і не висуває специфічних вимог до пристроїв користувачів. Система коректно працює на комп'ютерах, ноутбуках, планшетах та смартфонах з будь-якими сучасними браузерами (Chrome, Firefox, Safari, Edge). Основною рекомендацією тільки є наявність стабільного інтернет-з'єднання, проте навіть мінімальна швидкість не є критичною завдяки оптимізації завантаження ресурсів.

В поточній реалізації система розгорнута на хмарних платформах: backend на Railway, frontend на Vercel, база даних використовує MongoDB Atlas. Така архітектура забезпечує високу доступність без необхідності власної серверної інфраструктури. Проєкт також підтримує контейнеризацію через Docker для локального розгортання або використання на власних серверах. Це дозволяє ізолювати програмне середовище, гарантуючи стабільну роботу алгоритмів захисту незалежно від базової операційної системи сервера. Крім того, такий підхід значно спрощує процедуру горизонтального масштабування системи у разі різкого збільшення кількості користувачів

4.2 Інсталяція програмного забезпечення

Процес інсталяції програмного забезпечення залежить від бажаного способу розгортання. Розглянемо основні варіанти встановлення системи для локальної розробки та виробничого середовища.

Для локального розгортання спочатку необхідно встановити Node.js та npm на робочу машину. Ці компоненти можна завантажити з офіційного сайту.

Наступним кроком є встановлення MongoDB. Для локальної розробки достатньо Community Edition з офіційного сайту mongodb.com. Простішим варіантом є використання MongoDB Atlas — безкоштовного хмарного сервісу, який не потребує локального встановлення.

Вихідний код системи завантажується з Git-репозиторію командою git clone з відповідним URL. Проєкт складається з двох основних директорій: backend для серверної частини та frontend для клієнтської. У кожній директорії потрібно виконати команду npm install для встановлення всіх залежностей проєкту. Цей процес автоматично завантажує та встановлює всі необхідні бібліотеки.

Конфігурація системи здійснюється через файли змінних середовища. У директорії backend створюється файл .env з наступними параметрами: MONGODB_URI для підключення до бази даних, JWT_SECRET для підпису токенів автентифікації, MASTER_KEY для шифрування 2FA секретів, PORT для номера порту сервера. У директорії frontend створюється файл .env.local з параметром NEXT_PUBLIC_API_URL, який вказує на адресу backend API.

Після налаштування запуск системи виконується командами npm run dev в обох директоріях для режиму розробки. Backend за замовчуванням запускається на порту 3001, frontend — на порту 3000. При успішному запуску користувач може відкрити браузер за адресою http://localhost:3000 та почати працювати з системою.

Для продакшн-розгортання рекомендується використання Docker-контейнерів. У проєкті присутні файли Dockerfile для backend та frontend, а також docker-compose.yml для запуску всієї системи. Команда docker-compose up автоматично створює та запускає всі необхідні контейнери, включаючи MongoDB, що значно спрощує процес розгортання в продакшн-середовищі.

Альтернативно система може бути розгорнута на хмарних платформах. Railway підтримує автоматичне розгортання backend з Git-репозиторію, автоматично розпізнаючи Node.js проект та встановлюючи залежності. Vercel аналогічно працює з frontend, оптимізуючи Next.js застосунок для продакшн-середовища. Обидві платформи надають безкоштовні тарифи для невеликих проєктів та автоматично налаштовують HTTPS. Такі рішення є дуже швидкими та легко реалізовуються, проте є достатньо потужними.

Після завершення інсталяції система готова до використання. Користувачі отримують доступ до повного функціоналу через веб-інтерфейс, включаючи реєстрацію, автентифікацію з можливістю активації 2FA, перегляд каталогу товарів з персоналізованими рекомендаціями та здійснення покупок. Адміністратори додатково отримують доступ до панелі моніторингу безпеки для контролю за роботою системи захисту.

4.3 Робота користувача з системою

Розроблена система дозволяє користувачам безпечно працювати з платформою електронної комерції через зручний веб-інтерфейс. Інтерфейс розроблено відповідно до сучасних практик дизайну UI/UX і вимог доступності. Розглянемо найпоширеніші сценарії, за допомогою яких користувачі взаємодіють із системою.

На рисунку 4.1 зображено, як починається процес реєстрації на сторінці створення облікового запису. Користувач заповнює поля імені, електронної пошти та пароля у формі. Системою здійснюється валідація паролів клієнта з візуальним відображенням вимог до складності. Завдяки цьому користувач отримує миттєвий зворотний зв’язок, що дозволяє скоригувати пароль згідно з політиками безпеки ще до моменту відправки форми на сервер, значно покращуючи загальний досвід взаємодії з системою.

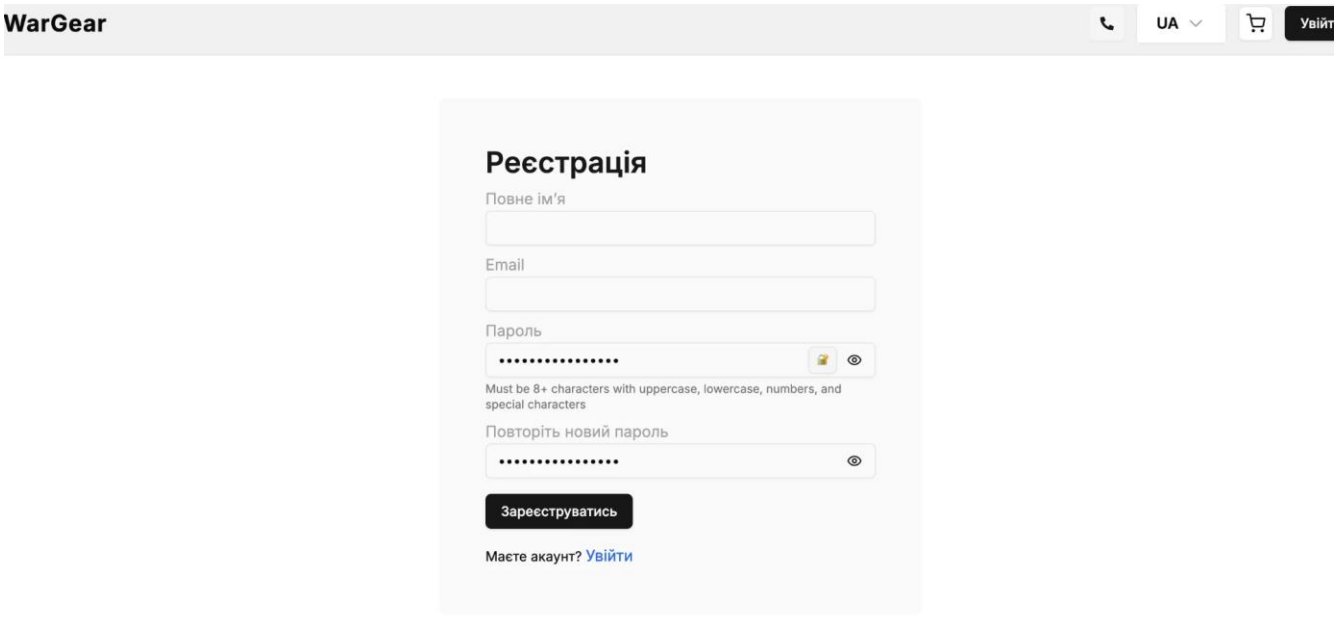


Рисунок 4.1 — Форма реєстрації з генератором паролів

Ключовою функцією форми реєстрації є вбудований генератор безпечних паролів, який можна запустити, натиснувши кнопку з іконкою замка поруч із полем пароля. При натисканні цієї кнопки автоматично створюється криптографічно стійкий пароль, який відповідає всім стандартам безпеки, включаючи мінімум вісім символів, великі та малі літери, цифри та спеціальні символи. Така функція буде покращуючи процес реєстрації та заохочуючи користувачів використовувати надійні паролі. Новий пароль відразу підставляється в обидва поля — пароль і підтвердження.

Зазначені дані проходять багаторівневу валідацію системою. Користувач отримує детальну інформацію про вимоги до паролю, якщо пароль не відповідає вимогам складності (рис. 4.2). Спливаюче вікно детально описує всі вимоги, включаючи необхідну кількість символів, наявність великих і малих літер, принаймні одну цифру та один спеціальний символ із набору (!@#$%^&*...). До усунення проблеми поле паролю підсвічується червоним кольором. Така інтерактивна підказка допомагає користувачеві швидко зорієнтуватися в правилах безпеки та створити надійний пароль без зайвих зусиль.

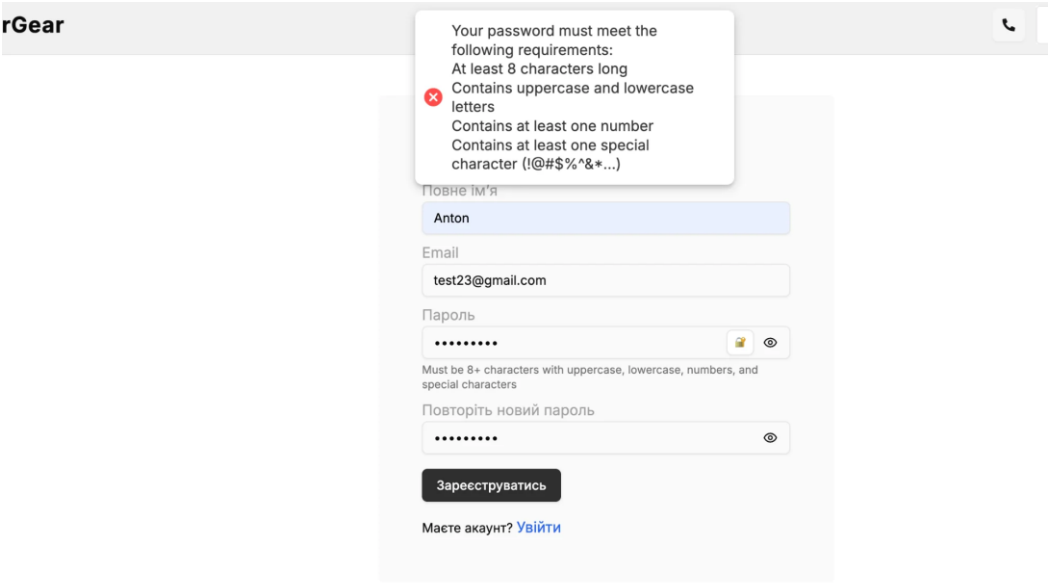


Рисунок 4.2 — Повідомлення про вимоги до складності пароля

Додатковою функцією безпеки є перевірка паролів через сервіс Have I Been Pwned. Якщо введений пароль був скомпрометований у відомих витоках даних, система не дозволяє його використання (рис. 4.3). Користувач отримує відповідне повідомлення. Повідомлення рекомендує використати кнопку генерації пароля або обрати інший варіант. Перевірка виконується безпечно через k-anonymity модель без передачі повного пароля на зовнішній сервер.

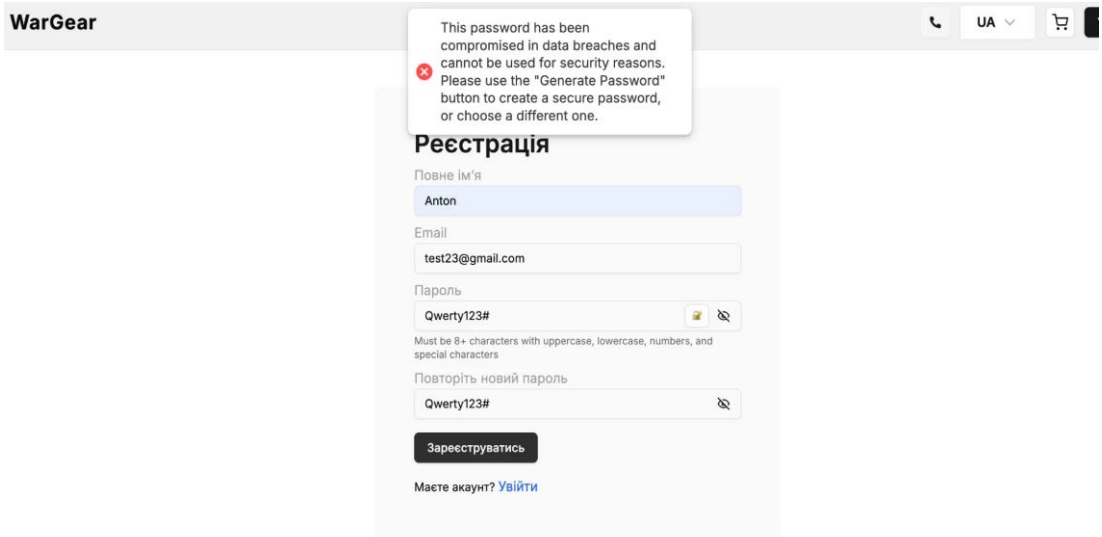


Рисунок 4.3 — Попередження про скомпрометований пароль

Після завершення реєстрації користувач може отримати доступ до системи, використовуючи форму автентифікації. Система перевіряє облікові дані при вході, розраховує ризик-скор на основі поведінкових метрик і, за необхідності, застосовує додаткові перевірки безпеки. Захищена сесія з унікальним ідентифікатором створюється після успішної автентифікації.

Спеціальна сторінка дозволяє користувачам переглядати та керувати активними сесіями (рис. 4.4). Для кожної сесії відображається інформація, яка включає тип пристрою та операційну систему (наприклад, MacOS 10.15.7), геолокацію входу, IP-адресу, час останньої активності та час початку сесії. Для привернення уваги користувача до потенційно підозрілих входів нові пристрої позначаються спеціальною міткою «Новий пристрій».

Кожна сесія має кнопку "End Session", яка дозволяє користувачеві примусово завершити сеанс з будь-якого пристрою. Це дуже важлива функція безпеки, яка дає користувачам контроль над доступом до їхнього облікового запису. Якщо користувач виявляє незнайому сесію, він може її припинити та змінити пароль.

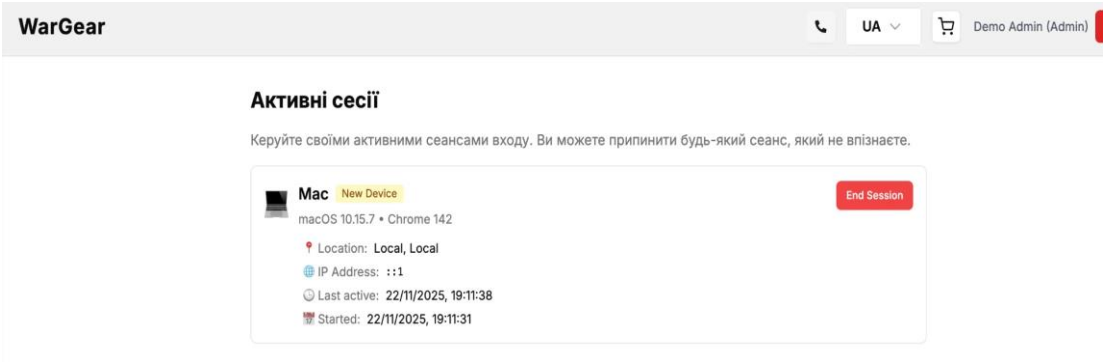


Рисунок 4.4 — Сторінка управління активними сесіями

Для підвищення рівня захисту облікового запису система надає можливість активації двофакторної автентифікації на основі TOTP. Налаштування 2FA доступне через сторінку безпеки в профілі користувача. На початковому етапі система відображає статус 2FA та кнопку активації, вона зображена на рисунку 4.5.

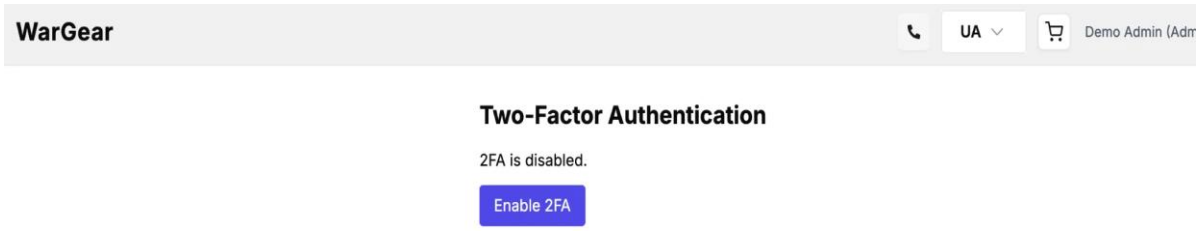


Рисунок 4.5 — Початковий стан двофакторної автентифікації

При натисканні на кнопку “Enable 2FA” генерується QR-код для сканування мобільним застосунком-автентифікатором (рис. 4.6). Також для розуміння на сторінці є повідомлення що потрібно зробити.

Після сканування в користувача в застосунку-автентифікатору з’явиться запис про цей веб-портал з його поштою та відобразиться код, який користувачу потрібно ввести в поле. Після всіх успішних маніпуляцій двофакторна автентифікація буде увімкнена.

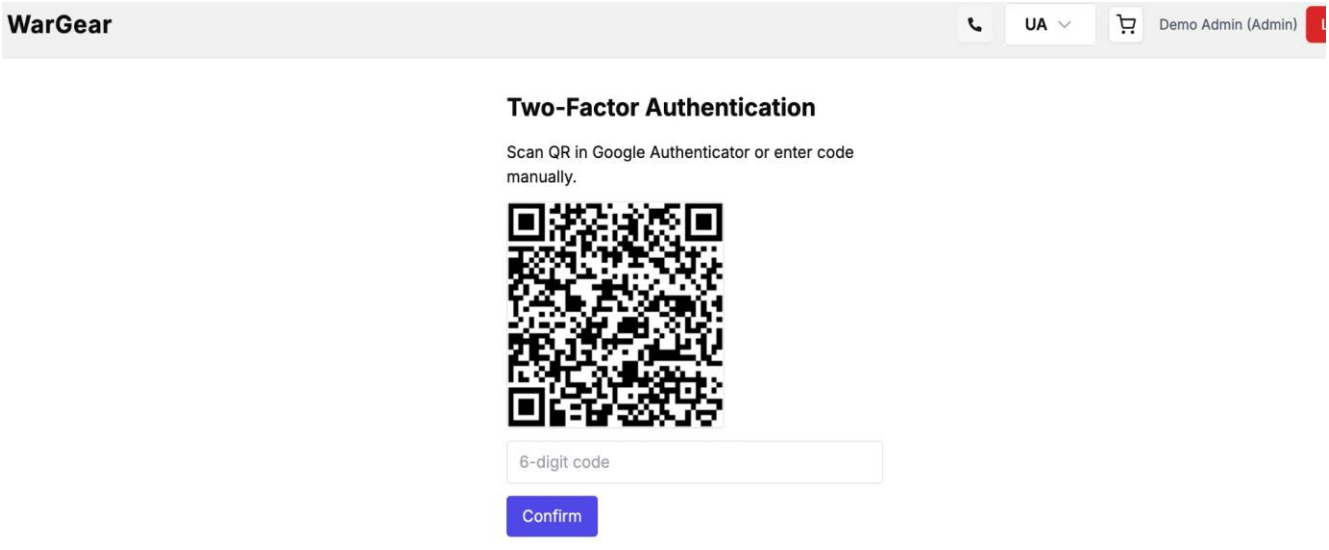


Рисунок 4.6 — Налаштування 2FA через QR-код

Після активації сторінка змінюється на статус "2FA is enabled" з кнопкою деактивації червоного кольору (рис. 4.7).

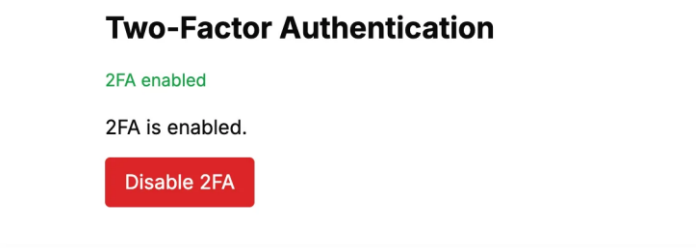


Рисунок 4.7 — Активований стан двофакторної автентифікації

З увімкненою 2FA процес входу в систему включає додатковий крок. Після введення email та пароля користувач перенаправляється на сторінку введення одноразового коду, вона зображена на рисунку 4.8. Форма містить поле для 6-значного коду з підказкою та кнопку “Verify code”. У правому нижньому куті відображається нотифікація з нагадуванням про необхідність двофакторної автентифікації. Код генерується мобільним застосунком кожні 30 секунд, цим самим забезпечуючи додатковим рівнем захисту навіть у випадку компрометації пароля.

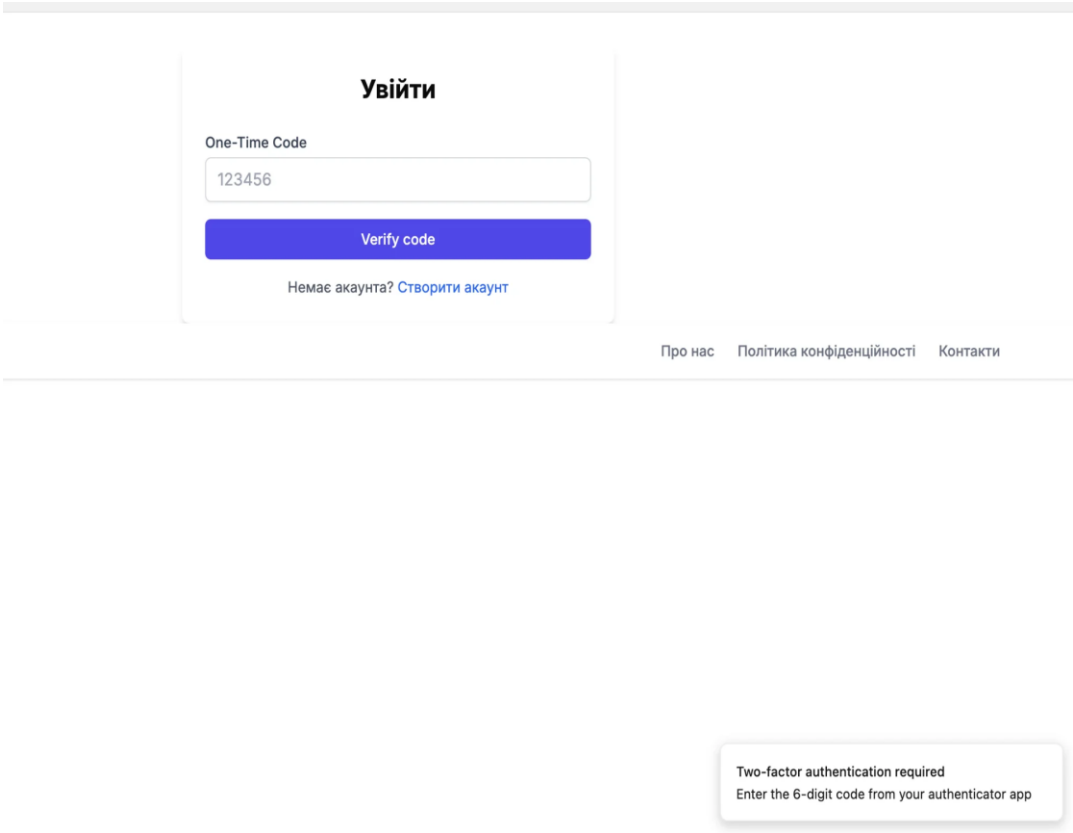


Рисунок 4.8 — Введення коду 2FA при вході в систему

Користувачі з роллю адміністратор отримують доступ до комплексної панелі моніторингу та управління безпекою системи. Панель доступна через розділ “Захист” у бічному меню адмін інтерфейсу.

Центральна сторінка панелі безпеки (рис. 4.9) містить секцію налаштування AI Defense з параметрами адаптивного управління ризиками. Адміністратор може налаштовувати часові вікна аналізу: “Window (min)” для довгострокового тренду та “Short window (sec)” для оперативного реагування. Параметр “EWMA alpha” контролює швидкість адаптації до нових значень. Налаштовуються також пороги реагування: “Block risk” для автоматичного блокування, “Throttle risk” для обмеження швидкості запитів. Ваги різних типів аномалій (“W fail”, “W streak”, “W attempts”, “W novelty”) дозволяють налаштовувати алгоритм обчислення ризику під специфіку проєкту.

У нижній частині розміщені інтерактивні демонстраційні модулі. Секція “AI Defense Demo” дозволяє запустити симуляцію атаки для перевірки ефективності алгоритму. Параметр “Attempts” вказує кількість спроб, а кнопка “Run demo” вмикає тест. Аналогічно працює “Mixed pattern demo” для перевірки реакції системи на змішані патерни поведінки.

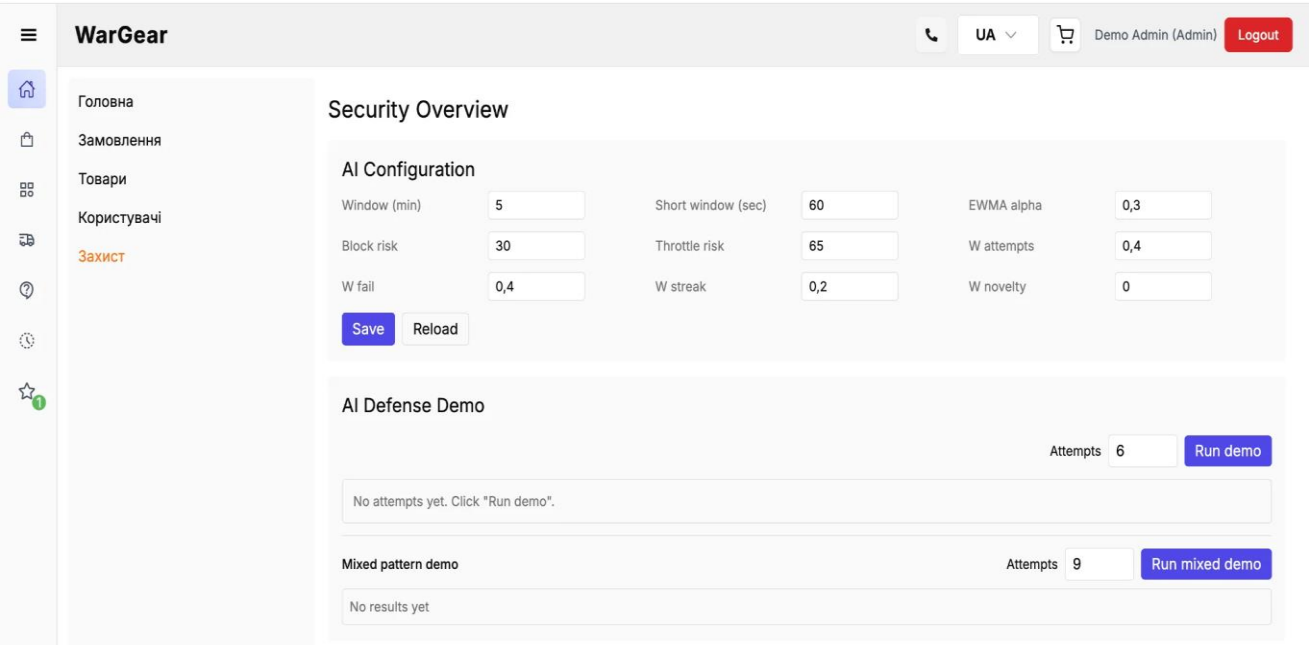


Рисунок 4.9 — Панель налаштувань AI Defense

Наступна секція адміністративної панелі, що зображена на рисунку 4.10 присвячена налаштуванню bot detection та моніторингу інцидентів. Секція “Bot Detection Config” дозволяє налаштувати детальні параметри виявлення ботів. Чекбокс “Enabled” активує/деактивує механізм. Налаштовуються пороги реагування: “Block score” для повного блокування, “Throttle score” для уповільнення, “Step-up score” для додаткових перевірок. Параметр “Window (sec)” визначає часове вікно для аналізу поведінки. Ваги окремих ознак (“wTyping”, “wMouse”, “wDwell”, “wNavigator”, “wHeaders”, “wRate”) дозволяють тонко налаштувати алгоритм обчислення ризик-скору.

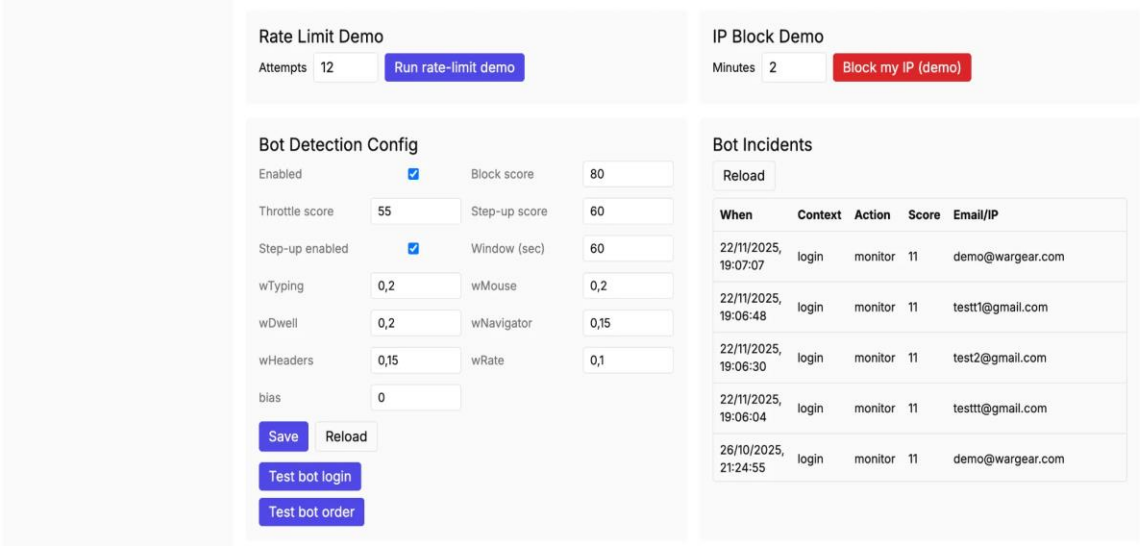


Рисунок 4.10 — Налаштування bot detection та журнал інцидентів

Справа від налаштувань розміщена таблиця “Bot Incidents” з журналом виявлених підозрілих активностей. Для кожного інциденту відображаються: дата та час виявлення, контекст виявлення, дія системи, обчислений скор ризику та email або IP-адреса порушника. Кнопки “Test bot login” та “Test bot order” запускають автоматизовані тести для перевірки ефективності виявлення ботів у різних сценаріях.

У верхній частині сторінки розміщені демонстраційні модулі “Rate Limit Demo” та “IP Block Demo”. Модуль “Rate Limit Demo” дозволяє симулювати багаторазові спроби входу для перевірки механізму обмеження частоти запитів.

Секція “IP Block Demo” надає можливість тестування механізму блокування IP-адрес з налаштуванням тривалості блокування.

Наступна сторінка панелі (рис. 4.11) присвячена WAF та загальній статистиці безпеки. Секція “WAF Demo” містить кнопки “Test SQLi” та “Test XSS” для перевірки ефективності фільтрації шкідливих запитів. Нижче відображається статистика за останні 24 години: кількість заблокованих та успішних запитів, і поточна кількість заблокованих IP.

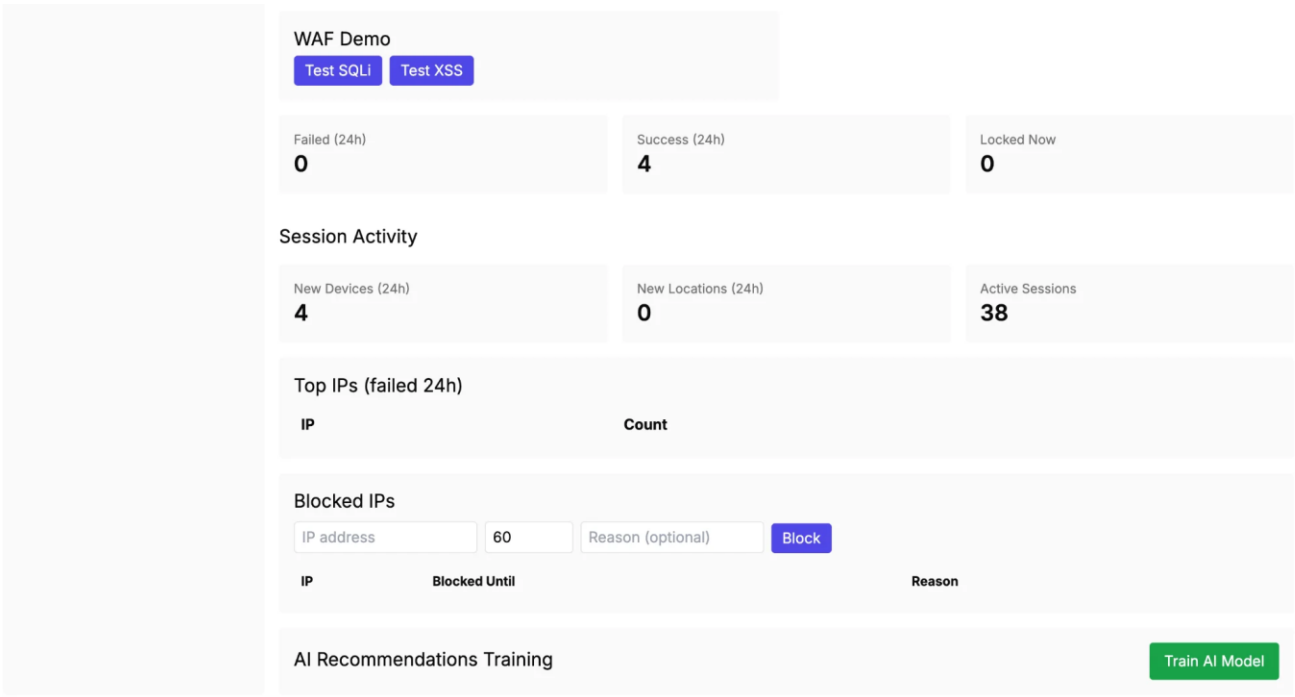


Рисунок 4.11 — Статистика WAF та активності сесій

Секція “Session Activity” показує метрики активності користувачів: кількість нових пристроїв за добу, нових локацій та загальну кількість активних сесій. Ці показники допомагають адміністраторам виявляти аномальні патерни використання системи.

Таблиця “Top IPs (failed 24h)” відображає IP-адреси з найбільшою кількістю невдалих спроб доступу, що дозволяє швидко ідентифікувати джерела атак. Нижче розміщена форма ручного блокування IP-адрес “Blocked Ips” з полями для введення IP, тривалості блокування в хвилинах та необов'язкової причини блокування. У

нижній частині сторінки знаходиться кнопка “Train AI Model” для ручного запуску перенавчання моделі рекомендацій.

Одним із ключових елементів користувацького досвіду є система персоналізованих рекомендацій. На головній сторінці авторизованим користувачам відображається секція “Персональні рекомендації”, зображено на рисунку 4.12. Товари підбираються на основі гібридного алгоритму, що аналізує історію переглядів, покупок та профіль інтересів користувача.

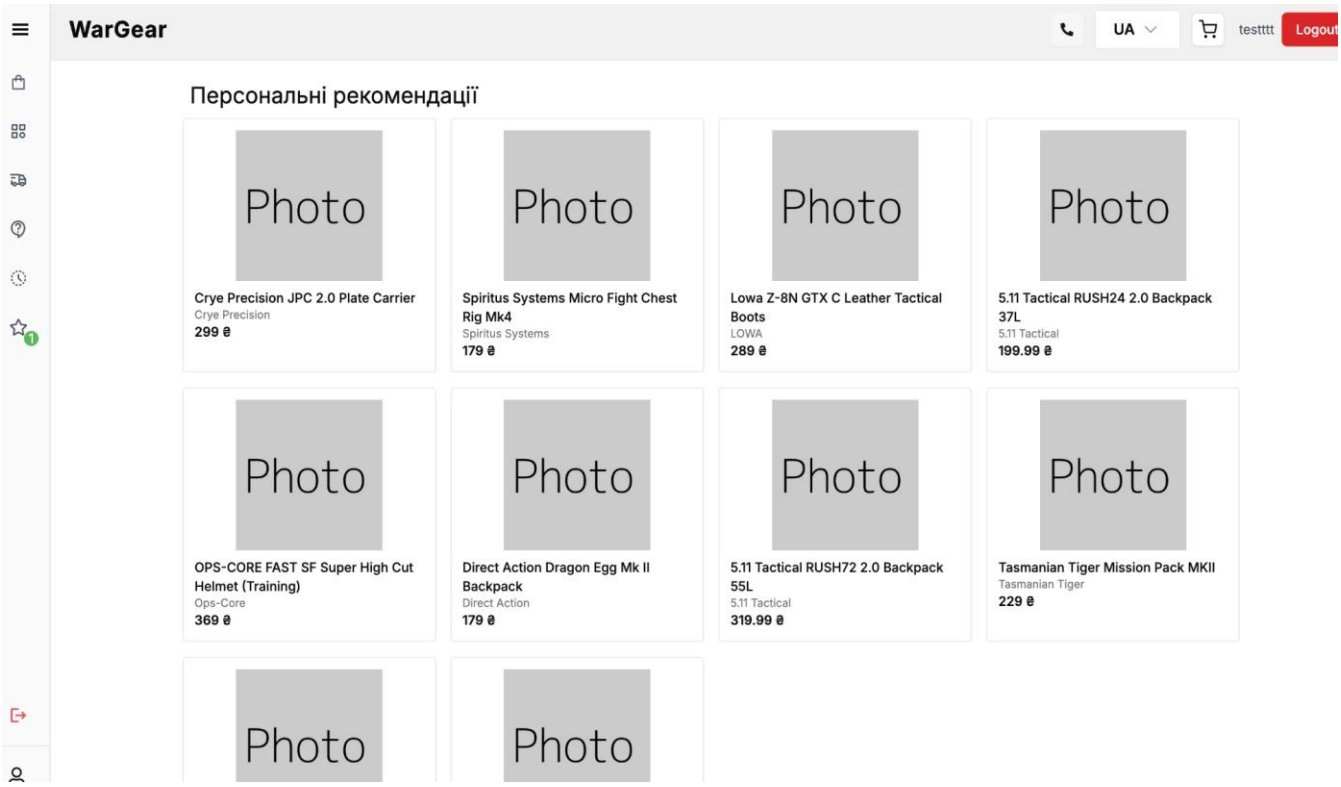


Рисунок 4.12 — Персональні рекомендації на головній сторінці

Кожна картка товару містить зображення, назву продукту, бренд та ціну. Рекомендації оновлюються після кожного перегляду товару або здійснення покупки, адаптуючись до зміни вподобань користувача. Для нових користувачів без історії взаємодій система показує популярні та рекомендовані товари на основі глобальної статистики.

На сторінці конкретного товару (рис. 4.13) відображається секція “Рекомендовані товари” зі схожими продуктами.

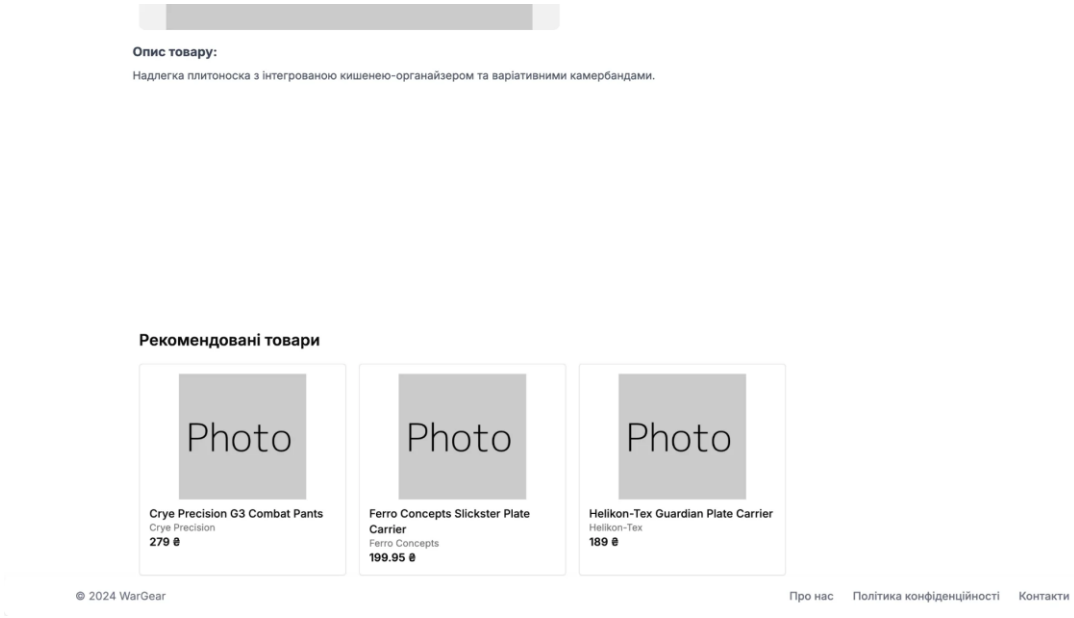


Рисунок 4.13 — Рекомендовані товари на сторінці продукту

Система автоматично відслідковує перегляди товарів та використовує ці дані для перенавчання моделі рекомендацій. Користувач не потребує виконувати жодних додаткових дій — персоналізація відбувається прозоро у фоновому режимі, постійно покращуючи якість пропозицій.

Таким чином, розроблена система надає користувачам інтуїтивний інтерфейс з комплексними механізмами безпеки та персоналізації. Багаторівнева архітектура захисту працює прозоро для користувача, не погіршуючи користувацький досвід, а адміністративна панель забезпечує повний контроль над параметрами безпеки та можливість оперативного реагування на загрози.

4.4 Тестування та аналіз ефективності системи

Для підтвердження того, що система працює правильно, було проведено функціональне тестування основних компонентів і аналіз того, наскільки добре вони працюють у реальних умовах експлуатації. З використанням демонстраційних

функцій адміністративної панелі та аналізу логів системи тестування проводилося в розгорнутому виробничому середовищі.

Модуль виявлення ботів тестувався за допомогою вбудованих функцій демонстрації адміністративної панелі. Кнопки “Test bot login” і “Test bot order” створюють автоматизовані запити з такими характеристиками, як підозрілі User-Agent заголовки, відсутність руху миші та нульовий час перебування на сторінці. Система коректно обчислювала високий ризик-скор (понад 80 балів) під час тестування та застосовувала відповідні дії, включаючи блокування або throttling залежно від налаштованих порогів.

Для перевірки чи правильно відпрацьовують функції безпеки на звичайних, реальних користувачах виконувались стандартні дії, які включають вхід в систему з природньою швидкістю введення, рух мишею та перегляд товарів. У кожному випадку система оцінювала низький ризик-скор і не проводила додаткових перевірок. Адміністративна панель зберігала результати в таблиці “Bot Incidents”, які маркувались відповідними мітками “monitor” або “block”.

У секції WAF Demo використовувались кнопки “Тест SQLi” та “Тест XSS” для перевірки модуля WAF. Коли натискаєте кнопку, система надсилає запити з типовими патернами атак, такими як SQL-ін’єкції типу "OR '1'='1" і XSS-скрипти типу "alert('xss')." WAF успішно виявляв і блокував кожен тестовий запит, що підтверджувалося повідомленнями про блокування та відповідними записами в логах. Статистика “Failed (24h)” коректно показувала заблоковані спроби.

Двофакторна автентифікація тестувалась шляхом активації 2FA для тестового облікового запису через мобільний застосунок Google Authenticator. Перевірялась коректність генерації QR-коду, успішність активації після введення коду, та обов'язковість введення коду при наступному вході. Всі сценарії відпрацювали коректно: неправильний код відхилявся, правильний код дозволяв вхід, а без коду доступ блокувався. Зведені результати тестування наведені в таблиці 4.1.

Таким чином, результати підтверджують надійність комплексного підходу до захисту, де кожен рівень ефективно виконує свою функцію фільтрації загроз.

Відсутність хибних спрацювань під час тестів свідчить про те, що впроваджена система безпеки не погіршує користувацький досвід для легітимних клієнтів.

Таблиця 4.1 — Результати функціонального тестування модулів безпеки

Компонент системи	**Результат тестування**
Bot detection (легітимні дії)	Ризик-скор < 30, дозволено
Bot detection (симуляція бота)	Ризик-скор > 80, заблоковано
WAF (тест SQL-ін'єкцій)	Виявлено і заблоковано
WAF (тест XSS-атак)	Виявлено і заблоковано
2FA автентифікація	Коректна робота з TOTP

Ефективність системи рекомендацій перевірялась експериментально через створення тестових облікових записів з різною історією взаємодій. Для першого тестового користувача без історії переглядів система показувала загальні популярні товари (товари з прапорцем isFeatured). Після перегляду кількох товарів рекомендації адаптувались і почали відображати схожі товари цієї категорії.

Перевірка колаборативної фільтрації проводилася за допомогою сценарію з кількома тестовими користувачами. Коли кілька користувачів переглянули та замовили схожі набори товарів (наприклад, користувачі А та Б замовили шоломи та розвантажувальні системи), система почала рекомендувати користувачу А продукти, які купив користувач Б, і навпаки. Це підтвердило, що система ідентифікації користувачів працює правильно.

На сторінках конкретних товарів у секції «Рекомендовані товари» було проведено перевірку контентної фільтрації. При перегляді тактичного рюкзака “5.11 Tactical” система показувала інші рюкзаки того ж бренду, з подібними розмірами та характеристиками. Рекомендації формуються на основі порівняння токенів із назвами, описами, категоріями та брендами товарів.

Час генерації рекомендацій вимірювався через Network Tab інструментів розробника браузера. Запит до ендпойнту /products/recommended виконувався в середньому за 300-450 мс залежно від кількості товарів у каталозі. Навчання моделі матричної факторизації через кнопку “Train AI Model” в адміністративній панелі займало приблизно 5 секунд для тестового невеликого датасету, що є прийнятним для фонового періодичного перенавчання.

Продуктивність системи оцінювалась через вбудовані інструменти браузера (Chrome DevTools) та моніторинг платформи Railway. Середній час відгуку сервера (TTFB) для типових запитів становив 150-200 мс, що є прийнятним показником для веб-застосунків. Час повного завантаження головної сторінки з рекомендаціями не перевищував 1.2 секунди при стабільному інтернет-з'єднанні.

Споживання ресурсів на сервері відслідковувалось через дашборд Railway. При помірному навантаженні (5-10 одночасних користувачів) використання CPU становило 15-25%, оперативної пам'яті — близько 180 MB. Розмір бази даних MongoDB Atlas з тестовими даними (близько 50 товарів, 10 користувачів, історія взаємодій) становив приблизно 8 MB, що демонструє ефективність зберігання даних.

Окрему увагу приділено тестуванню стійкості системи до навантаження та її поведінці в умовах одночасного виконання множинних операцій. Для симуляції реалістичних сценаріїв було створено автоматизовані скрипти, які імітували дії декількох користувачів одночасно: вхід у систему, перегляд товарів, додавання товарів до кошика та оформлення замовлень. При навантаженні до 20 одночасних користувачів система зберігала стабільний час відгуку без суттєвого зростання використання ресурсів. Модуль виявлення ботів правильно розрізняв легітимну активність від підозрілої навіть за умов високого навантаження, що підтверджує його ефективність у реальних умовах експлуатації.

Додатково було проведено аналіз захищеності системи від найпоширеніших вразливостей згідно з класифікацією OWASP Top 10. Тестування включало спроби SQL-ін’єкцій через різні точки входу (форми авторизації, пошукові запити, параметри URL), перевірку можливості XSS-атак через користувацький контент, та аналіз безпеки сесій та токенів автентифікації. Система успішно протистояла

всім базовим векторам атак завдяки комбінації WAF-модуля, валідації даних на рівні додатку та застосування параметризованих запитів до бази даних. Відсутність критичних вразливостей підтверджує, що архітектурні рішення та реалізовані механізми захисту відповідають сучасним стандартам безпеки веб-застосунків.

Таким чином, проведене тестування підтвердило коректність роботи всіх ключових компонентів системи. Модулі безпеки успішно виявляють та блокують типові загрози, система рекомендацій адаптується до вподобань користувачів, а продуктивність залишається на прийнятному рівні для проєктів малого та середнього масштабу.

Висновки до розділу 4

У четвертому розділі було описано процес роботи користувачів та адміністраторів з розробленою системою та проведено комплексне тестування ключових модулів. Тестування підтвердило, високий рівень безпеки системи та зручність використання.

Процес автентифікації реалізовано розширеним підходом, який включає базову автентифікацію та двофакторну автентифікацію на основі TOTP. Система рекомендацій ефективно адаптується до вподобань користувачів, відслідковуючи взаємодії асинхронно без впливу на швидкість інтерфейсу.

Адмін панель надає повний контроль над системою безпеки з можливістю реального моніторингу інцидентів та динамічного налаштування параметрів захисту без перезапуску сервера. Інструменти тестування WAF дозволяють перевіряти ефективність захисту безпосередньо через веб-інтерфейс.

Результати тестування підтвердили коректність функціонування всіх модулів. Модуль автентифікації та 2FA успішно пройшли тести на валідацію та обробку граничних випадків. Bot detection показав високу точність класифікації. WAF ефективно виявляє базові атаки, хоча складні payloads потребують додаткового налаштування правил. Система рекомендацій генерує результати за менш ніж 200 мілісекунд, що прийнятно для виробництва.

5 СТАРТАП-ПРОЄКТ

Після успішної технічної реалізації та тестування системи адаптивного захисту, наступним важливим кроком є оцінка її комерційного потенціалу. П’ятий розділ присвячено розробці стартап-проєкту, який має на меті трансформувати створене інженерне рішення у конкурентоспроможний ринковий продукт.

У цьому розділі обґрунтовується бізнес-концепція проєкту, проводиться комплексний аналіз ринкового середовища кібербезпеки та оцінюються сильні й слабкі сторони розробки порівняно з існуючими аналогами. Основна увага приділяється формуванню стратегії виведення продукту на ринок та визначенню економічної доцільності його впровадження для різних сегментів електронної комерції.

5.1 Опис ідеї стартап-проєкту

Ідея стартап-проєкту полягає в комерціалізації розробленої системи адаптивного підбору та захисту даних веб-порталів з використанням штучного інтелекту. Основна цінність проєкту полягає в тому, щоб надати малим і середнім онлайн бізнесам, доступний, але потужний інструмент, щоб захистити їх від кіберзагроз і та персоналізувати користувацький досвід без необхідності інвестувати великі гроші в дорогі корпоративні рішення.

Розроблене програмне забезпечення може бути запропоноване ринку у вигляді хмарного SaaS-рішення з підпискою, self-hosted рішення для компаній, які хочуть отримати максимум захищеності, або як white-label продукт для компаній, які зацікавлені розробленим продуктом та хочуть використовувати під своїм брендом. Кожен з цих напрямків має свої переваги та цільову аудиторію. SaaS-модель забезпечує найшвидший вихід на ринок та мінімальні бар’єри входу для клієнтів, оскільки не вимагає від них технічних знань для розгортання. Self-hosted варіант привабливий для компаній, які працюють з чутливими даними і не хочуть

залежати від сторонньої інфраструктури. White-label підхід дозволяє масштабувати бізнес через партнерську мережу та швидко заповнювати ринок власним продуктом. Опис всіх можливих рішень наведено в таблиці 5.1.

Таблиця 5.1 — Опис ідеї стартап-проєкту

Зміст ідеї	**Напрямки застосування**	**Вигоди для користувача**
Хмарне SaaS-рішення з підпискою	Малі та середні онлайн-магазини (до 10000 товарів)	Швидке впровадження (1-2 дні), низькі початкові витрати, автоматичні оновлення, масштабованість
Self-hosted рішення	Середні компанії з високими вимогами до безпеки та контролю даних	Повний контроль над даними, відсутність залежності від зовнішніх сервісів, можливість кастомізації
White-label продукт	Веб-агенції, IT-компанії	Можливість брендування під власною маркою, розширення портфоліо послуг, додатковий дохід

Основними конкурентними перевагами є інтеграція семи рівнів захисту в єдиній системі, власна реалізація AI-алгоритмів без залежності від інших дорогих сервісів, гнучка цінова політика, яка обговорюється з майбутніми клієнтами та простота інтеграції з іншими веб-застосунками та сервісами завдяки RESTful API. Система є комплексним готовим рішенням яке може використовуватись не тільки для інтернет-магазину, а може бути адаптованим і для інших сервісів, таких як музична платформа, онлайн кінотеатр та інші. Розроблений програмний продукт

містить об’єднаний функціонал, який зазвичай пропонують окремо з використанням декількох різних сервісів, а це значно спрощує технічну підтримку та знижує загальну вартість обслуговування проєктів клієнта.

Великою перевагою проєкту також є те, що рішення є інтернаціональним, за рахунок локалізації та використання стандартних для міжнародного ринку принципів керування інтернет-магазином.

5.2 Аналіз ринкових можливостей та загроз

Протягом останніх років ринок кібербезпеки та персоналізації в електронній комерції продовжує розвиватися. За даними аналітичних агенцій, світовий ринок рішень для захисту веб-застосунків прогнозується досягти понад 7 млрд доларів США до 2024 року з прогнозованим середньорічним темпом зростання (CAGR) на рівні 12—15 відсотків до 2030 року. Ринок електронної комерції в Україні також активно розвивається, особливо після 2022 року, коли значна частина бізнесу перейшла в онлайн. На українському ринку електронної комерції є значний потенціал для рішень у сфері безпеки, і експерти прогнозують, що до 2024 року він досягне 7—8 млрд доларів США.

Зростання кількості кіберзагроз і атак, зростання вимог регуляторів щодо захисту персональних даних (GDPR в Європі та аналогічні закони в Україні), зростання очікувань користувачів щодо персоналізації сервісів і загальна цифровізація економіки є основними драйверами зростання. Відповідно, сучасні веб-платформи повинні не лише протистояти атакам, але й забезпечувати безшовний користувацький досвід, що вимагає впровадження нових архітектурних підходів. Саме інтеграція технологій безпеки та алгоритмів персоналізації стає ефективною відповіддю на ці комплексні виклики ринку. Пандемія COVID-19 значно прискорила перехід бізнесу в онлайн, що збільшило попит на рішення, які захищають веб-застосунки. Ключові характеристики ринку наведено в таблиці 5.2.

Таблиця 5.2 — Попередня характеристика потенційного ринку стартап-проєкту

Показники стану ринку	**Характеристика**	**Висновки**
Кількість головних гравців	10-15 великих (Cloudflare, AWS, Akamai) + сотні малих	Ринок з помірною концентрацією, є можливості для нішевих гравців
Загальний обсяг продажу, $ млрд/рік	7-8 млрд глобально, ~50 млн в Україні	Великий ринок з потенціалом зростання
Динаміка ринку	Зростання 12-15% щорічно	Привабливий ринок для входу нових гравців
Наявність обмежень для входу	Технологічні бар’єри, вимоги до довіри клієнтів	Середні бар’єри входу, можна подолати якісним продуктом
Специфічні вимоги до стандартизації	Відповідність GDPR, ISO 27001, PCI DSS	Необхідні сертифікації для роботи з фінансовими даними

Основними ринковими можливостями є наступне: недостатнє покриття рішеннями доступними для малого та середнього бізнесу, високі ціни конкурентів створюють простір для більш доступних альтернатив; зростання попиту на локалізовані рішення, з підтримкою української мови та які відповідають місцевому законодавству; можливість співпрацювати з веб-агенціями та системними інтеграторами, щоб швидше вийти на ринок. Можливість виходу на сусідні ринки Центральної та Східної Європи, де також існує попит на зручні рішення з хорошою локалізацією, є ще однією можливістю.

Ринкові загрози включають швидкі технологічні зміни, які вимагають постійних інвестицій у розвиток продукту, економічну нестабільність у регіоні, яка може знизити платоспроможність цільової аудиторії, агресивну конкурентну політику великих гравців, які можуть знизити ціни або запропонувати безкоштовні

базові плани для витіснення нових гравців. Характеристика майбутніх клієнтів наведена в таблиці 5.3.

Таблиця 5.3 — Характеристика потенційних клієнтів стартап-проєкту

Потреба	**Цільова аудиторія**	**Відмінності у поведінці**	**Вимоги до товару**
Базовий захист та рекомендації	Малі онлайн-магазини (до 1000 товарів)	Обмежений бюджет, потреба у простоті	Низька ціна, простота впровадження, базові функції
Розширений захист та персоналізація	Середній бізнес (1000-10000 товарів)	Готовність інвестувати, потреба у аналітиці	Розширені функції, інтеграції, технічна підтримка
Корпоративне рішення	Великі компанії	Високі вимоги до надійності та контролю	Максимальна кастомізація, SLA, on-premise варіант

Важливо відзначити, що різні сегменти клієнтів мають різні цикли прийняття рішень. Малий бізнес зазвичай приймає рішення швидко (1-2 тижні) і орієнтується переважно на ціну та простоту використання. Середній бізнес потребує більше часу (1-2 місяці) для оцінки функціоналу та проведення пілотного тестування. Великі компанії можуть приймати рішення 3-6 місяців, включаючи тендерні процедури та узгодження з різними відділами. Відповідно, планування продажів та маркетингові комунікації мають враховувати цю специфіку, забезпечуючи індивідуальний підхід до кожної групи потенційних замовників для оптимізації конверсії.

Урахування цих відмінностей дозволяє побудувати збалансовану воронку продажів, де швидкі угоди з малим бізнесом забезпечують операційну ліквідність, а контракти з великими клієнтами формують довгострокову прибутковість.

5.3 Технологічна здійсненність ідеї проєкту

Технологічна реалізація проєкту базується на сучасному та перевіреному технологічному стеку. Backend системи розроблено на NestJS — прогресивному Node.js фреймворку, який забезпечує високу продуктивність та масштабованість. Frontend побудовано на React та Next.js, що гарантує швидкий відклик інтерфейсу та хороші показники SEO. Для зберігання даних використовується MongoDB — гнучка NoSQL база даних, яка добре справляється з великими обсягами неструктурованих даних та забезпечує горизонтальне масштабування. Результат аналізу технологічної здійсненності проєкту наведено в таблиці 5.4.

Таблиця 5.4 — Технологічна здійсненність ідеї проєкту

Технологія/Компонент	**Джерело отримання**	**Доступність та умови**
Backend (NestJS, Node.js)	Open-source, npm registry	Безкоштовно, MIT License
Frontend (React, Next.js)	Open-source, npm registry	Безкоштовно, MIT License
База даних (MongoDB)	MongoDB Inc., хмарний або self-hosted	Безкоштовний tier для старту, платні плани при масштабуванні

Ключовою технологічною перевагою є власна реалізація алгоритмів машинного навчання для системи рекомендацій. На відміну від використання готових ML-платформ (TensorFlow, PyTorch), які вимагають значних обчислювальних ресурсів, розроблена система використовує оптимізовані алгоритми матричної факторизації з можливістю горизонтального масштабування. Це дозволяє знизити операційні витрати та пропонувати більш конкурентні ціни клієнтам.

Для розгортання системи використовуються сучасні DevOps практики. Backend розгорнуто на платформі Railway, яка забезпечує автоматичне масштабування та моніторинг. Frontend хоститься на Vercel — спеціалізованій

платформі для Next.js застосунків з глобальною CDN мережею. Такий підхід дозволяє мінімізувати початкові інвестиції в інфраструктуру та оплачувати лише реально використані ресурси. Для моніторингу та логування використовуються безкоштовні інструменти з open-source екосистеми.

Можливості для вдосконалення технології включають інтеграцію додаткових ML-моделей для покращення точності детектування ботів, розширення функціоналу системи рекомендацій за рахунок аналізу зображень товарів, впровадження технології blockchain для аудиту безпекових подій, та розробку мобільних додатків для адміністраторів. Всі ці вдосконалення можуть бути реалізовані поетапно на основі отриманих доходів від перших клієнтів.

5.4 Аналіз конкурентного середовища

Конкурентне середовище на ринку рішень для захисту веб-застосунків та персоналізації характеризується наявністю кількох великих гравців та численних малих компаній, що спеціалізуються на окремих аспектах безпеки або рекомендацій. Основними конкурентами можна вважати Cloudflare (комплексний захист та CDN), AWS Shield та WAF (рішення від Amazon), Akamai Bot Manager (захист від ботів), а також локальні українські компанії, що надають послуги з кібербезпеки. Кожен з цих конкурентів має свої сильні та слабкі сторони, які необхідно врахувати при формуванні стратегії позиціювання. Результат аналізу конкурентів наведено в таблиці 5.5.

Таблиця 5.5 — Порівняльний аналіз конкурентів

Характеристика	**Власний проєкт**	**Cloudflare**	**AWS Shield**	**Akamai**	**Локальні компанії**
Ціна	Від $29/міс	Від $20/міс (базовий)	Від $3000/міс	Від $10000/міс	Індивідуально, дорого

Кінець таблиці 5.5.

Комплексність	7 рівнів + AI рекомендації	Висока (без рекомендацій)	Висока (без рекомендацій)	Висока (без рекомендацій)	Частткова
Простота впровадження	Висока (1-2 дні)	Висока	Середня	Низька	Низька
Підтримка української мови	Так	Частково	Ні	Ні	Так

Ключовими конкурентними перевагами розробленого рішення є поєднання захисту та персоналізації в одній системі, що рідко зустрічається у конкурентів, значно нижча ціна порівняно з корпоративними рішеннями при збереженні основного функціоналу, простота інтеграції завдяки детальній документації та RESTful API, локалізація інтерфейсу українською мовою та адаптація під вимоги українського законодавства. Водночас, слабкими сторонами є відсутність глобального бренду та репутації, обмежені маркетингові бюджети порівняно з великими гравцями, та менша географічна присутність. Результати SWOT-аналізу наведені в таблиці 5.6.

Таблиця 5.6 — SWOT-аналіз стартап-проєкту

	Сильні сторони (S)	**Слабкі сторони (W)**
Внутрішні	— інтеграція 7 рівнів захисту + AI; — власні ML алгоритми; — низька ціна; — простота впровадження.	— самостійна підтримка та оновлення; — молодий бренд; — обмежені ресурси на маркетинг; — потреба у сертифікаціях.

Кінець таблиці 5.6.

	Можливості (О)	**Загрози (Т)**
Зовнішні	— зростання ринку e-commerce; — збільшення кіберзагроз; — недостатнє покриття сегменту МСБ; — партнерства з агенціями.	— агресивна конкуренція великих гравців; — можливі зміни у законодавстві; — економічна нестабільність.

На основі SWOT-аналізу можна сформулювати основні стратегічні напрямки: використати низьку ціну та унікальну комбінацію функцій для швидкого завоювання частки в сегменті МСБ, активно будувати партнерську мережу для подолання обмежень маркетингового бюджету, отримати базові сертифікації безпеки для підвищення довіри клієнтів, розвивати спільноту користувачів для органічного зростання через word-of-mouth маркетинг.

5.5 Стратегія ринкового впровадження

Стратегія ринкового впровадження базується на поетапному підході з фокусом на конкретні сегменти ринку. Перший етап (0-6 місяців) передбачає запуск MVP версії SaaS-рішення з базовими функціями для малих онлайн-магазинів, активний digital-маркетинг через соціальні мережі та спеціалізовані форуми, пропозицію безкоштовного пробного періоду на 30 днів для залучення перших клієнтів, збір відгуків та доопрацювання продукту на основі реальних кейсів. Основні канали залучення на цьому етапі: контент-маркетинг (блог з технічними статтями), SEO-оптимізація сайту, участь в онлайн-спільнотах e-commerce підприємців. Деталізована стратегія позиціювання, що включає ключові повідомлення та канали комунікації для кожного цільового сегменту, наведена в таблиці 5.7.

Таблиця 5.7 — Стратегія позиціювання на ринку

Цільовий сегмент	**Базова стратегія**	**Ключове повідомлення**	**Канали комунікації**
Малі онлайн-магазини	Лідерство за ціною	Корпоративний захист за доступною ціною	Facebook, Instagram, тематичні групи
Середній бізнес	Диференціація	Унікальна комбінація захисту та AI-персоналізації	LinkedIn, галузеві конференції, email-маркетинг
Веб-агенції	Партнерство	Розширте портфоліо та заробляйте на кожному клієнті	Прямі продажі, IT-виставки, партнерські програми
Великі компанії	Кастомізація	Гнучке рішення з можливістю повного контролю	Прямі продажі, тендери, B2B майданчики

Другий етап (6-12 місяців) включає розширення функціоналу на основі відгуків користувачів, запуск self-hosted версії для клієнтів з підвищеними вимогами до безпеки, початок роботи з веб-агенціями через партнерську програму з комісією 20-30%, отримання базових сертифікатів безпеки (ISO 27001 базовий рівень). На цьому етапі важливо досягти переломної точки та сформувати стабільну клієнтську базу з високим рівнем утримання.

Третій етап (12-24 місяці) передбачає масштабування на інші ринки (спочатку Польща, потім інші країни ЄС), розробку white-label версії для великих агенцій, створення екосистеми інтеграцій з популярними CMS та e-commerce платформами (WooCommerce, Shopify, Magento), залучення інвестицій для прискорення росту. Розглядається можливість інвестицій обсягом $500K-1M для масштабування команди та маркетингу. Цінова стратегія наведена в таблиці 5.8.

Таблиця 5.8 — Цінова стратегія

Тарифний план	**Ціна, $/місяць**	**Основні функції**	**Цільова аудиторія**
Starter	29	Базовий захист, до 1000 товарів, базові рекомендації	Малі магазини
Business	99	Розширений захист, до 10000 товарів, AI рекомендації, аналітика	Середній бізнес
Enterprise	Індивідуально	Всі функції + SLA, пріоритетна підтримка, кастомізація	Великі компанії
Self-hosted	499 (разова ліцензія)	Повний доступ до коду, самостійний хостинг, оновлення 1 рік	Компанії з високими вимогами

Ключовими метриками успіху на першому році будуть: 100+ активних платних клієнтів, MRR (Monthly Recurring Revenue) на рівні $10000, LTV/CAC співвідношення більше 3, Net Promoter Score (NPS) вище 40, час впровадження менше 48 годин для 90% клієнтів, відтік клієнтів менше 5% на місяць. Досягнення цих показників дозволить продемонструвати product-market fit та підготуватися до масштабування.

Для досягнення цих показників планується інвестувати в контент-маркетинг (блог, кейс-стаді, технічна документація), performance-маркетинг (Google Ads, Facebook Ads з таргетингом на власників онлайн-магазинів), побудову комьюніті

(Telegram-канал, YouTube з технічними туторіалами), участь у галузевих конференціях та виставках. Бюджет на маркетинг в перший рік складає $30000, з яких 40% на performance-маркетинг, 30% на контент, 20% на івенти та 10% на інструменти та аналітику.

Висновки до розділу 5

У п'ятому розділі проведено комплексний аналіз можливостей комерціалізації розробленої системи у формі стартап-проєкту. Визначено три напрямки застосування: SaaS-рішення, self-hosted версія та white-label продукт.

Аналіз ринку показав стабільне зростання з CAGR 12-15% та недостатнє покриття сегменту МСБ, що створює сприятливі умови для входу. Основні драйвери попиту — збільшення кіберзагроз та зростання очікувань щодо персоналізації.

Технологічна здійсненність підтверджена використанням open-source технологій та власних ML-алгоритмів, що мінімізує початкові інвестиції. Конкурентні переваги включають унікальну комбінацію захисту та персоналізації, низьку ціну та локалізацію.

Розроблено поетапну стратегію впровадження з чіткими метриками: 100+ клієнтів, MRR $10000, LTV/CAC > 3 на першому році. Визначено цінову стратегію з чотирма тарифами від $29/міс. SWOT-аналіз підтвердив високий потенціал комерційної успішності.

Реалізація запропонованого стартап-проєкту дозволить не лише отримати економічний прибуток, але й підвищити загальний рівень кібербезпеки в сегменті малого та середнього бізнесу. Залучення інвестицій на ранніх етапах дасть змогу прискорити розробку додаткових функцій та масштабування продукту на міжнародні ринки. Гнучка бізнес-модель забезпечує стійкість проєкту до змін ринкової кон’юнктури та дозволяє швидко адаптуватися до нових потреб клієнтів. Таким чином, розроблений продукт має всі необхідні передумови для успішного запуску та зайняття стійкої позиції в обраній ніші.

ВИСНОВКИ

1. Проведено аналіз сучасних кіберзагроз для веб-порталів електронної комерції та існуючих рішень у великих веб-сервісах. Було встановлено, що 60% трафіку деяких інтернет-магазинів створюють автоматизовані системи, включаючи шкідливі боти. Проаналізовано комерційні рішення захисту (Cloudflare, AWS Shield, Akamai Bot Manager) та системи рекомендацій (Netflix, Amazon та Spotify). Виявлено наступні недоліки: висока вартість, складність адаптації, значні вимоги до ресурсів та залежність від зовнішніх сервісів.

2. Досліджено математичні методи для виявлення автоматизованих загроз, криптографічного захисту, управління ризиками та рекомендаційних систем. Для виявлення ботів застосовано поведінковий аналіз та обчислення ризик-скору через зважену суму нормалізованих ознак. Для криптографічного захисту використано bcrypt та AES-256-GCM. Застосовано метод EWMA для управління ризиками. Для рекомендацій досліджено матричну факторизацію зі стохастичним градієнтним спуском, TF-IDF та косинусну схожість для контентної фільтрації, rank-based blending для об'єднання результатів.

3. Розроблено архітектуру семирівневої системи захисту веб-порталу. Перший рівень — виявлення ботів через аналіз семи поведінкових метрик з ризик-скором від 0 до 100. Другий — WAF для детекції SQL-ін'єкцій та XSS через regex та подвійне декодування. Третій — двофакторна автентифікація на основі TOTP з шифруванням AES-256-GCM. Четвертий — контроль спроб входу з блокуванням після 5 невдалих спроб. П'ятий — управління сесіями з прив'язкою JWT-токенів. Шостий — адаптивне управління ризиками через EWMA. Сьомий — Content Security Policy. Система інтегрована в логіку веб-застосунку на рівні сервісів.

4. Реалізовано гібридну систему адаптивних рекомендацій з використанням матричної факторизації, стохастичного градієнтного спуску, колаборативної та контентної фільтрації. Створено власну імплементацію SGD без готових ML-бібліотек з параметрами: 16-вимірні вектори, 5 епох, lr=0.02, reg=0.02. Колаборативна фільтрація аналізує схожість користувачів через косинусну міру на основі історії покупок та переглядів. Контентна фільтрація використовує

токенізацію та TF-IDF. Фінальні рекомендації формуються через rank-based blending. Система адаптується до холодного старту через fallback на популярні товари.

5. Проведено комплексне тестування розробленої системи та оцінено ефективність запропонованих рішень. Модульне тестування підтвердило коректність автентифікації, JWT-токенів, TOTP-кодів для 2FA, bot detection скорів та генерації рекомендацій. Інтеграційне тестування верифікувало взаємодію між модулями через REST API. Функціональне тестування підтвердило роботу step-up challenge, блокування IP-адрес, детекції WAF. Система рекомендацій демонструє час генерації до 200 мс. Результати підтверджують готовність до production використання.

Перспективи подальшого розвитку проєкту включають впровадження deep learning моделей для аналізу зображень товарів та текстових описів, що дозволить покращити точність рекомендацій. Інтеграція GraphQL API забезпечить оптимізацію завантаження даних та зменшення кількості запитів між клієнтом та сервером. Використання WebSocket протоколу дозволить реалізувати real-time оновлення статистики безпекових інцідентів в адміністративній панелі. Розробка мобільних додатків на React Native з повторним використанням бізнес-логіки розширить охоплення користувачів. Впровадження A/B тестування забезпечить можливість оптимізації параметрів систем захисту та рекомендацій на основі реальних метрик конверсії та поведінки користувачів.

Також у роботі обґрунтовано економічну доцільність впровадження розробленої системи у вигляді стартап-проєкту. Проведений маркетинговий аналіз підтвердив наявність стійкого попиту на комплексні рішення безпеки серед підприємств малого та середнього бізнесу. Запропонована бізнес-модель дозволяє досягти окупності проєкту в прийнятні терміни при збереженні доступної цінової політики для клієнтів. У підсумку, розроблений програмний продукт є повністю завершеним, комерційно привабливим рішенням, готовим до впровадження в реальних умовах експлуатації .

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Thales Group. 2025 Bad Bot Report: The Rise of Automated Threats. *Thales Group*. URL: https://cpl.thalesgroup.com/sites/default/files/content/campaigns/badbot/2025-Bad-Bot-Report.pdf (date of access: 20.11.2025).

2. Cramer-Flood E. Worldwide Retail Ecommerce Forecast 2024. *EMARKETER*. URL: https://www.emarketer.com/content/worldwide-retail-ecommerce-forecast-2024 (date of access: 03.12.2025).

3. OWASP Top Ten Web Application Security Risks. *OWASP Foundation*. URL: https://owasp.org/www-project-top-ten/ (date of access: 20.11.2025).

4. Understanding and Mitigating DDoS Attacks. *Cloudflare*. URL: https://blog.cloudflare.com/ddos-threat-report-for-2024-q4/ (date of access: 20.11.2025).

5. AWS WAF Developer Guide. *Amazon Web Services*. URL: https://docs.aws.amazon.com/waf/latest/developerguide/waf-chapter.html (date of access: 20.11.2025).

6. Amatriain X., Basilico J. Netflix Recommendations: Beyond the 5 Stars. *Netflix Technology Blog*. URL: https://netflixtechblog.com/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429 (date of access: 17.11.2025).

7. NestJS — A Progressive Node.js Framework. *NestJS*. URL: https://docs.nestjs.com/ (date of access: 20.11.2025).

8. MongoDB Manual. *MongoDB Inc*. URL: https://www.mongodb.com/docs/manual/ (date of access: 20.11.2025).

9. Next.js 14 Documentation. *Vercel*. URL: https://nextjs.org/docs (date of access: 20.11.2025).

10. McGraw G. Software Security. *Datenschutz und Datensicherheit — DuD*. 2012. Vol. 36, no. 9. P. 662–665. URL: https://doi.org/10.1007/s11623-012-0222-3 (date of access: 23.11.2025).

11. Stallings W. Cryptography and Network Security: Principles and Practice. 8th ed. London : Pearson Education, 2020. 800 p.

12. Provos N., Mazières D. A Future-Adaptable Password Scheme. *Proceedings of the USENIX Annual Technical Conference*. 1999. P. 1–13.

13. Dworkin M. J. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. *National Institute of Standards and Technology*. URL: https://doi.org/10.6028/nist.fips.202 (date of access: 18.11.2025).

14. Electronic Authentication Guideline / W. E. Burr et al. Gaithersburg : National Institute of Standards and Technology, 2011. URL: https://doi.org/10.6028/nist.sp.800-63-1 (date of access: 23.11.2025).

15. TOTP: Time-Based One-Time Password Algorithm / D. M'Raihi et al. *RFC Editor*. URL: https://doi.org/10.17487/rfc6238 (date of access: 23.11.2025).

16. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT). *RFC Editor*. 2015. URL: https://doi.org/10.17487/rfc7519 (date of access: 03.12.2025).

17. Ricci F., Rokach L., Shapira B. Introduction to Recommender Systems Handbook. *Recommender Systems Handbook*. Boston : Springer, 2010. P. 1–35.

18. Aggarwal C. C. Recommender Systems: The Textbook. Cham : Springer, 2018. 519 p.

19. Item-based Collaborative Filtering Recommendation Algorithms / B. Sarwar et al. *Proceedings of the 10th International Conference on World Wide Web.* New York : ACM, 2001. URL: https://doi.org/10.1145/371920.372071 (date of access: 23.11.2025).

20. Lops P., de Gemmis M., Semeraro G. Content-based Recommender Systems: State of the Art and Trends. *Recommender Systems Handbook*. Boston : Springer, 2010. P. 73–105.

21. Harper F. M., Konstan J. A. The MovieLens Datasets. *ACM Transactions on Interactive Intelligent Systems*. 2016. Vol. 5, no. 4. P. 1–19. URL: https://doi.org/10.1145/2827872 (date of access: 24.11.2025).

22. Evaluating Collaborative Filtering Recommender Systems / J. L. Herlocker et al. *ACM Transactions on Information Systems*. 2004. Vol. 22, no. 1. P. 5–53.

23. Su X., Khoshgoftaar T. M. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*. 2009. Vol. 2009. P. 1–19.

24. Adomavicius G., Tuzhilin A. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*. 2005. Vol. 17, no. 6. P. 734–749.

25. Koren Y., Bell R., Volinsky C. Matrix Factorization Techniques for Recommender Systems. *Computer*. 2009. Vol. 42, no. 8. P. 30–37.

26. Salakhutdinov R., Mnih A. Probabilistic Matrix Factorization. *Advances in neural information processing systems*. 2007. Vol. 20. P. 1257–1264.

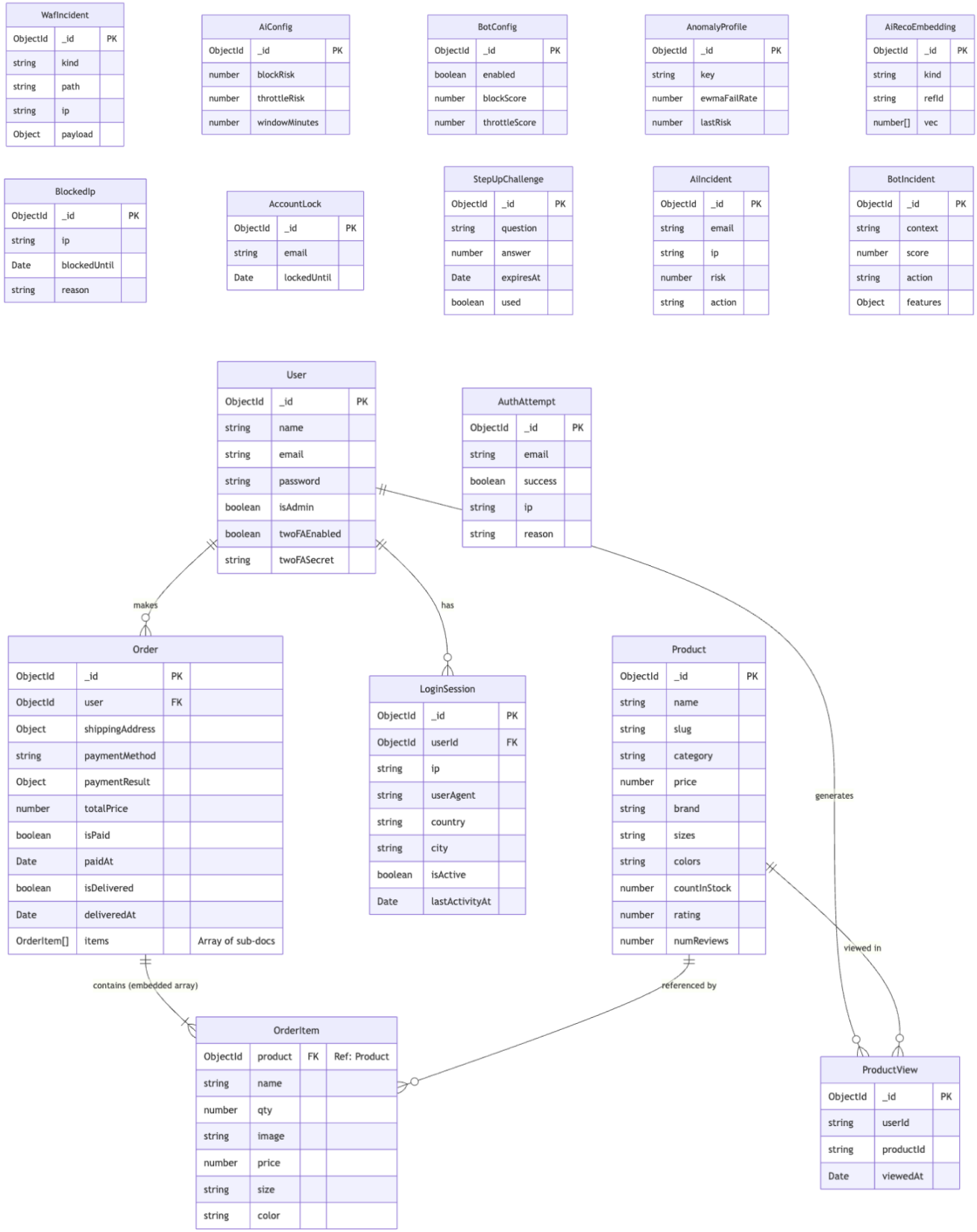
27. Bottou L. Large-Scale Machine Learning with Stochastic Gradient Descent. *Proceedings of COMPSTAT'2010*. Heidelberg : Physica-Verlag HD, 2010. P. 177–186.

28. Neural Collaborative Filtering / X. He et al. *Proceedings of the 26th International Conference on World Wide Web*. 2017. P. 173–182.

29. BPR: Bayesian Personalized Ranking from Implicit Feedback / S. Rendle et al. *Proceedings of the 25th conference on uncertainty in artificial intelligence*. 2009. P. 452–461.

30. Burke R. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*. 2002. Vol. 12, no. 4. P. 331–370.

ДОДАТОК А



A1 — ER-діаграма структури бази даних

ДОДАТОК Б

Сертифікат апробації роботи

www.scientia.report

CERTIFICATE OF PARTICIPATION

ISC № 2510310011

Anton Bahnii

participated in the IX International Scientific and Theoretical Conference

«Science of XXI Century: Development, Main Theories and Achievements

October 31, 2025 The Hague; The Netherlands

Academic Workload: 24 hours of participation / 0.8 ECTS credits

Confirmed Outcome: scientific paper published

10.36074/scientia-31.10.2025 ISBN 979-8-89660-278-1

MIRIAM GOLDENBLAT

President and responsible organizer

International Center of Scientific Research

Unified Register of Public Associations Number: 1499141.

ICSR INTERNATIONAL CENTER OF SCIENTIFIC RESEARCH

Hosted by an authorized **Crossref** member

The conference is registered in the **ResearchBib**

Conference proceedings are displayed in **Google Books & Google Scholar**

Bowker