Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти бакалавра

зі спеціальності 121 – Інженерія програмного забезпечення

На тему: Розробка інтернет-магазину комп'ютерної техніки з унікальними знижками

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ-20-1
_____ / С. А. Бондарчук /

Керівник
кваліфікаційної роботи _____ / І. О. Доценко
___ /

Завідувач кафедри _____ / А. М. Стрюк /
___

Кривий Ріг
2024

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання та програмного забезпечення

Ступінь вищої освіти: бакалавр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Зав. кафедри
_____ А. М. Стрюк
«___» _____ 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу

студенту групи ІПЗ-20-1 Бондарчуку Сергію Анатоілйовичу

1. Тема: Розробка інтернет-магазину комп'ютерної техніки з унікальними знижками затверджено наказом по КНУ № 275с від «15» квітня 2024 р.
2. Термін подання студентом закінченої роботи: «01» червня 2024р.
3. Вихідні дані по роботі: розроблювана система повинна підтримувати авторизацію, реєстрацію, додавання товару у кошик та генерацію унікальних випадкових знижок .
4. Зміст пояснювальної записки: проаналізувати стан існуючих аналогів та визначитись з метою розробки, вибрати метод реалізації завдання та створити діаграму, здійснити програмну реалізацію розробленої системи, провести тестування розробленої системи.
5. Перелік ілюстративного матеріалу: Діаграма потоків даних, блок-схема розробленого алгоритму, фрагменти коду, демонстрація аналогів.

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання етапів роботи
1	Аналіз професійної області проблем	15.04.2024 – 16.04.2024
2	Аналіз існуючих аналогів	16.04.2024 – 17.04.2024
3	Постановка проблеми	17.04.2024 – 18.04.2024
4	Мета і завдання кваліфікаційної роботи	18.04.2024 – 20.04.2024
5	Вибір технологій розробки	20.04.2024 – 22.04.2024
6	Вибір методів реалізації знижок	22.04.2024 – 23.04.2024
7	Створення діаграми	23.04.2024 – 26.04.2024
8	Опис алгоритмів роботи інтернет-магазину	26.04.2024 – 29.04.2024
9	Бібліотеки та компоненти	29.04.2024 – 01.05.2024
10	Створення бази даних	01.05.2024 – 05.05.2024
11	Генерація випадкової знижки	05.05.2024 – 09.05.2024
12	Реалізація функції скидання всіх знижок	09.05.2024 – 10.05.2024
13	Створення таймеру що веде відлік до оновлення всіх знижок	10.05.2024 – 11.05.2024
14	Реалізація корзини	11.05.2024– 17.05.2024
15	Інтерфейс сторінки та функції реєстрації та авторизації користувачів	17.05.2024 – 22.05.2024
16	Тестування додатку	22.05.2024 – 27.05.2024

Дата видачі завдання: «15» квітня 2024 р.

Студент _____ / С. А. Бондарчук /

Керівник роботи _____ / І. О. Доценко /

РЕФЕРАТ

ІНТЕРНЕТ-МАГАЗИН, КОМП'ЮТЕРНІ СКЛАДОВІ, ГЕНЕРАЦІЯ ЗНИЖКИ, АВТОРИЗАЦІЯ, РЕЄСТРАЦІЯ КОРИСТУВАЧІВ, БАЗА ДАНИХ, СЕРВЕР.

Пояснювальна записка: 72с., 20 рис., 1 дод., 11 джерел.

Мета кваліфікаційної роботи: розробка інтернет-магазину комп'ютерних складових, який забезпечує користувачам зручний та інтуїтивно зрозумілий інтерфейс для купівлі товарів, а також впровадження інноваційної системи персональних знижок.

Предмет розробки: інтернет-магазин комп'ютерних складових для забезпечення зручного та ефективного інтерфейсу для покупців.

Шляхи досягнення мети: аналіз проблемної області, розробка схеми та алгоритмів роботи магазину, створення програмного забезпечення.

Основні конструктивні, технологічні та технікоексплуатаційні показники та характеристики: зручний інтерфейс, швидка робота, можливість реєстрації та авторизації користувачів, генерація випадкових знижок.

Галузь застосування: електронна комерція, онлайн-торгівля комп'ютерними компонентами та аксесуарами.

Висновки: розроблений інтернет-магазин є ефективним та зручним для користувачів. Проект успішно виконав поставлені завдання і відповідає сучасним вимогам електронної комерції.

ABSTRACT

INTERNET SHOP, COMPUTER COMPONENTS, DISCOUNT GENERATION, AUTHORIZATION, USER REGISTRATION, DATABASE, SERVER

Explanatory Note: 72 pages, 20 figures, 1 appendix, 11 sources.

Subject of Development: An online store for computer components to provide a convenient and efficient interface for customers.

Objective: To develop an online store for computer components that offers users a convenient and intuitive interface for purchasing goods, as well as implementing an innovative system of personalized discounts.

Ways to Achieve the Objective: Analysis of the problem area, development of the shop's structure and algorithms, creation of software.

Main Design, Technological, and Technical-Operational Indicators and Characteristics: User-friendly interface, fast performance, user registration and authorization capabilities, random discount generation.

Field of Application: E-commerce, online trade of computer components and accessories.

Conclusions: The developed online store is efficient and convenient for users. The project has successfully achieved its goals and meets the modern requirements of e-commerce.

ЗМІСТ

ВСТУП

У сучасному світі комп’ютери займають важливе місце серед товарів які можна замовити онлайн. Завдяки швидкому темпу технологічного розвитку та постійному оновленню асортименту продукції, цей сегмент на ринку стає особливо динамічним і конкурентноздатним. Тому створення інноваційного інтернет-магазину комп’ютерної техніки є актуальною задачею, яка потребує комплексного підходу та поєднання технологічних та маркетингових розв’язків.

Основною задачею кваліфікаційної роботи є проведення аналізу інтернет-магазинів комп’ютерної техніки, визначення потреб аудиторії, розробка ефективного користувацького інтерфейсу та власне сама розробка онлайн-магазину комп’ютерної техніки, що враховує потреби сучасних користувачів і вирізняється своєю зрозумілістю і простотою в дизайні. Основний крок проекту полягатиме у наданні можливостей для зручного і швидкого придбання обладнання, а також у реалізації унікальних знижок для користувачів. Унікальні знижки сприятимуть популіризації сайту та активізації покупок. Для того, щоб користувач надовго залишався клієнтом онлайн-магазину, знижки мають оновлюватись кожен день.

1 АКТУАЛЬНІСТЬ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1.1 Аналіз професійної області проблеми

У цьому розділі був проведений комплексний аналіз стану ринку комп'ютерної техніки та інтернет-магазинів на сьогодняншній день, що дозволив виявити певні тенденції, проблеми та можливості у цій галузі.

Розглянемо стан ринку комп'ютерної техніки.

Зріс попит на комп’ютерну техніку:

- технологічний прогрес: зі збільшенням кількості нових технологій і технічних пристроїв, таких як потужні комп'ютери, ноутбуки, планшети та інші електронні девайси, зростає зацікавленість споживачів до оновлення своєї техніки;
- робота та навчання з дому: з пандемією COVID-19 та війною суттєво зросла потреба у техніці для роботи та навчання з дому. Це збільшило попит на ноутбуки комп’ютери та інше обладнання.

Конкуренція та насиченість ринку:

- велика кількість конкурентів: на ринку присутні багато компаній, від великих рітейлерів до дрібних спеціалізованих магазинів, це забезпечує широкий вибір для користувачів, але також підвищує конкуренцію;
- зменьшення цін для збільшення попиту: конкуренція часто призводить до зниження цін, це змушує інтернет-магазини шукати інші способи для утримання та залучення нових клієнтів через додаткові послуги та знижки.

1.2 Аналіз існуючих аналогів

Для того, щоб визначитись з найкращими підходами до розробки інтернет-магазину комп'ютерної техніки з унікальними знижками був проведений детальний аналіз існуючих платформ, що мають подібні послуги. Аналіз складався з вивчення їх функціональних можливостей, бізнес-моделей, а також переваг та недоліків цих інтернет-магазинів.

Проведений аналіз існуючих інтернет-магазинів:

- Amazon – один з найбільших світових інтернет-магазинів. Він пропонує широкий асортимент товарів, включаючи книги, електроніку, одяг, іграшки, побутову техніку та багато іншого. Сайт має зручний інтерфейс, швидку доставку та програму лояльності Amazon Prime. Ця програма має такі переваги, як безкоштовна доставка та доступ до стрімінгових сервісів. Amazon також має підримку платформи для інших продавців, це дозволяє підприємцям продавати свої товари на сайті. Завдяки своєму масштабу та інноваціям Amazon став взірцем для інших інтернет-магазинів;

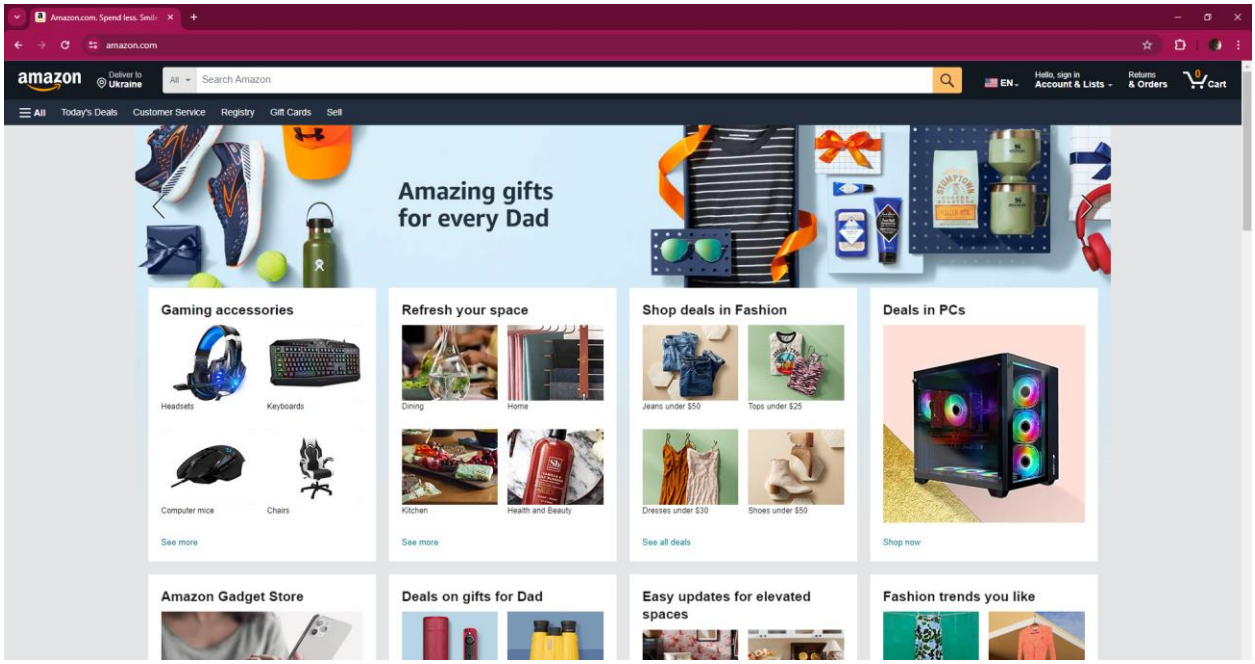

Рисунок 1.1 – Головна сторінка сайту Amazon

- Rozetka – один з найбільших в Україні інтернет-магазинів електроніки, побутової техніки, товарів для дому та офісу. На сайті можна побачити широкий асортимент товарів, зручний інтерфейс для пошуку та вибору товарів, різноманітні способи оплати та доставки, а також підтримується програма лояльності для постійних клієнтів. Rozetka відома своєю

надійністю, конкурентними цінами та високим рівнем обслуговування клієнтів;

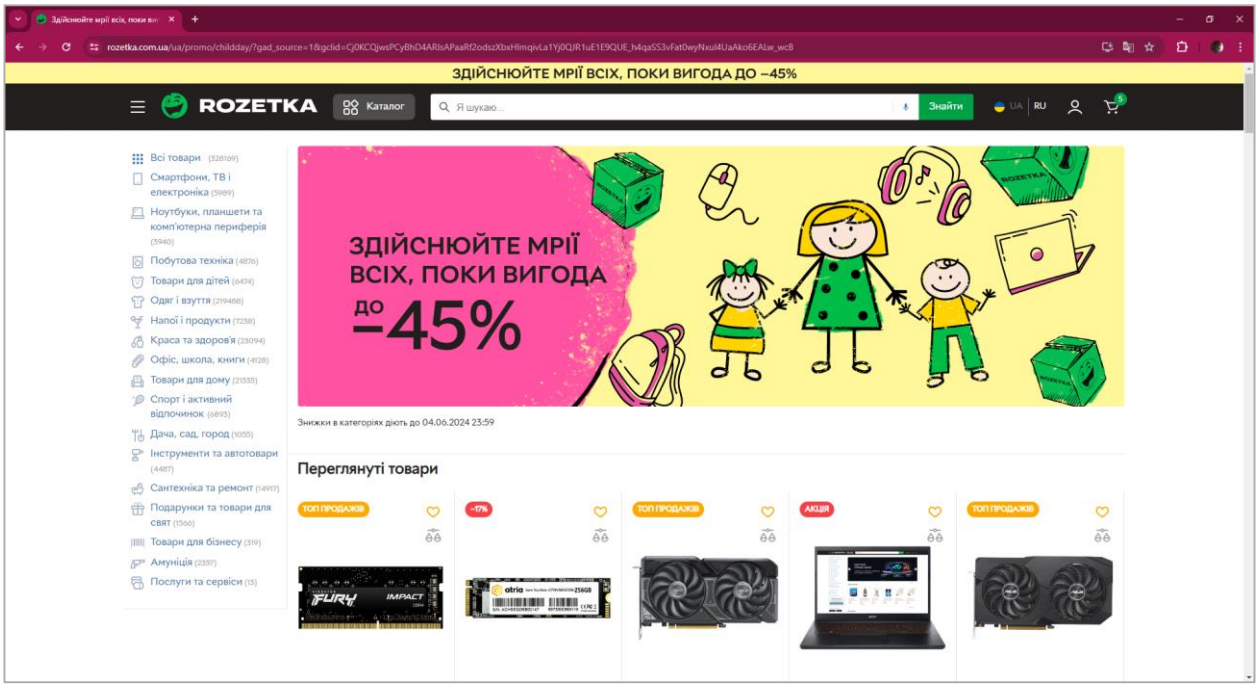

Рисунок 1.2 – Головна сторінка сайту Rozetka

- Citrus – сучасний український інтернет-магазин, що спеціалізується на продажу електроніки, гаджетів та аксесуарів. Цей магазин має широкий асортимент товарів, включаючи смартфони, ноутбуки, планшети, комп’ютерну техніку, аксесуари та побутову техніку. Цитрус відомий своїм інноваційним підходом до торгівлі, регулярними акціями, знижками та високим рівнем обслуговування клієнтів. Магазин також має фізичні пункти продажу у великих містах України, де клієнти можуть безпосередньо забрати свої замовлення та спробувати продукцію перед покупкою.

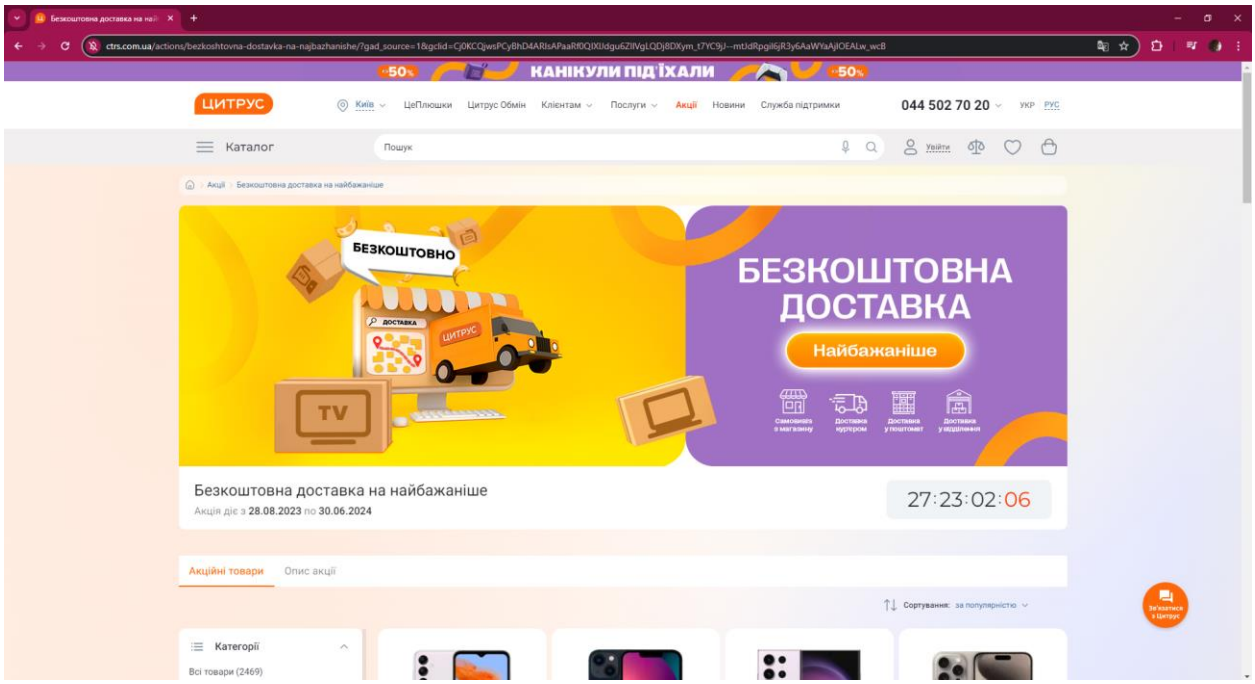

Рисунок 1.3 – Головна сторінка сайту Citrus

Розглянуті інтернет-магазини мають такі переваги:

- широкий асортимент товарів: розглянуті інтернет-магазини пропонують широкий асортимент товарів, які можуть задовольнити різні потреби їхніх клієнтів;
- регулярні знижки та акції: існуючі платформи використовують знижки та спеціальні пропозиції для залучення нових клієнтів і підвищення лояльності до вашого магазину;
- зручна навігація та користувальницький інтерфейс: високий рівень зручності використання забезпечує легкість використання та сприяє покращенню взаємодії з користувачем.

Виявлені недоліки:

- недостатня персоналізація знижок: більшість існуючих платформ пропонують знижки, які не обов'язково враховують особисті переваги та потреби користувачів;
- простота використання для нових користувачів: деякі онлайн-магазини мають надто складну навігацію або захаращений інтерфейс користувача, що може ускладнити пошук необхідної інформації чи продуктів.

1.3 Постановка проблеми

У сучасних умовах швидкого розвитку електронної комерції інтернет-магазини стикаються з серйозною конкуренцією. Залучення нових клієнтів та утримання існуючих є важливими завданнями, які потребують постійного вдосконалення маркетингових стратегій та впровадження інноваційних підходів. Одним з ефективних способів підвищення залученості клієнтів є використання систем персоналізації, зокрема, індивідуальних знижок. Проте, більшість існуючих інтернет-магазинів не використовують можливості, які надають такі системи, або ж реалізують їх недостатньо ефективно.

Основні проблеми, з якими стикаються інтернет-магазини при впровадженні систем знижок, включають:

- недостатня персоналізація: багато інтернет-магазинів пропонують стандартні знижки, які не враховують індивідуальні потреби та інтереси користувачів. Це призводить до зниження ефективності знижкових програм та втрати потенційних клієнтів;
- складність утримання клієнтів: у сучасних умовах користувачі мають доступ до великої кількості інтернет-магазинів, і знижки можуть бути одним з головних факторів, що впливають на їх рішення щодо покупки. Неефективні системи знижок ускладнюють утримання клієнтів та знижують їх лояльність;
- відсутність інноваційних підходів: більшість інтернет-магазинів використовують традиційні методи надання знижок, такі як статичні знижкові коди або одноразові акції. Вони не враховують можливості генерації випадкових персональних знижок, які можуть значно підвищити зацікавленість користувачів та стимулювати повторні покупки.

Враховуючи ці проблеми, виникає необхідність розробки інтернет-магазину, який би забезпечував ефективну систему генерації випадкових персональних знижок. Така система повинна враховувати індивідуальні потреби користувачів, підвищувати їх лояльність та сприяти збільшенню

продажів. Реалізація такої системи потребує ретельного аналізу існуючих методів генерації знижок, вибору відповідних технологій та забезпечення високого рівня безпеки даних.

1.4 Мета і завдання кваліфікаційної роботи

Метою дипломної роботи є розробка інтернет-магазину комп'ютерних складових, який забезпечує користувачам зручний та інтуїтивно зрозумілий інтерфейс для купівлі товарів, а також впровадження інноваційної системи персональних знижок. Основна функція системи знижок полягає у генерації індивідуальних пропозицій для кожного користувача шляхом відкриття віртуального сундука, який щоденно надає випадкову знижку. Це дозволить підвищити залученість користувачів, збільшити лояльність клієнтів та стимулювати повторні покупки.

Об’єктом дослідження є методи та технології розробки інтернет-магазинів, зокрема технології, що забезпечують генерацію та впровадження персональних знижок для користувачів.

Предмет розробки: інтернет-магазин комп'ютерної техніки з унікальними знижками.

Задачі для реалізації інтернет-магазину комп'ютерних складових:

- розробити форму реєстрації користувача;
- розробити форму авторизації користувача;
- розробити відображення списку продуктів з бази даних;
- розробити генерацію випадкових та унікальних знижок;
- розробити кошик для покупок та обробки замовлень;
- розробити відображення вмісту кошика та сумарної вартості;
- розробити функцію яка буде скидувати усі знижки;
- розробити таймер, який буде показувати скільки часу лишилось до оновлення.

2 РОЗРОБКА СХЕМИ І АЛГОРИТМІВ, ВИБІР ІНСТРУМЕНТІВ І МЕТОДІВ

2.1 Вибір технологій розробки

Вибір технологій є ключовим етапом у процесі розробки інтернет-магазину з інноваційною системою генерації випадкових персональних знижок. Розглянемо технології, які можуть бути використані для реалізації проєкту, включаючи мову програмування, фреймворки, бази даних та інші інструменти.

Для розробки інтернет-магазину можна використовувати такі мови програмування:

JavaScript:

Node.js: Використовується для серверної частини додатку. Node.js дозволяє створювати масштабовані веб-додатки та має багато бібліотек для роботи з випадковими числами [1], [2].

React: Популярний фреймворк для створення інтерактивного користувацького інтерфейсу на стороні клієнта [3].

Python:

Django: Потужний фреймворк для швидкої розробки веб-додатків. Django має вбудовану підтримку для роботи з базами даних та дозволяє легко реалізувати функціонал генерації випадкових чисел [4].

Flask: Легкий веб-фреймворк, який підходить для простих додатків і надає більшу гнучкість у виборі компонентів [5].

PHP:

Laravel: Фреймворк з простим і елегантним синтаксисом, який полегшує розробку веб-додатків. Laravel має потужні інструменти для роботи з базами даних та побудови RESTful API [6].

Для кваліфікаційної роботи було вирішено обрати мову програмування Python та фремворк Flask, оскільки я вже маю певний досвід у роботі з ними.

Для зберігання даних інтернет-магазину та інформації про знижки необхідна база даних. Розглянемо основні варіанти:

SQL бази даних:

MySQL: Відкрита реляційна база даних, яка широко використовується завдяки своїй надійності та простоті [7].

PostgreSQL: Потужна реляційна база даних з підтримкою складних запитів та розширень [8].

SQLite: Легка база даних, яка не потребує окремого серверного ПЗ. Ідеально підходить для невеликих проєктів і розробки [9].

NoSQL бази даних:

MongoDB: Документо-орієнтована база даних, яка дозволяє зберігати дані у форматі JSON. MongoDB підходить для динамічних і швидко змінюваних даних [10].

Для кваліфікаційної роботи, я обрав простоту та швидкість розробки: SQLite може бути оптимальним вибором. Вона не вимагає налаштування серверного оточення і легко інтегрується з більшістю веб-фреймворків. SQLite також підходить для зберігання невеликого обсягу даних та швидкого прототипування.

Вибір середовища розробки (IDE).

Для розробки веб-додатку було обрано Visual Studio Code, оскільки це середовище розробки забезпечує зручну та ефективну роботу.

VS Code є легким та швидким редактором коду з підтримкою багатьох мов програмування. Велика кількість розширень дозволяє налаштувати середовище під потреби розробника (наприклад, розширення для Python, JavaScript, підтримка Git). Інтуїтивно зрозумілий інтерфейс та потужні інструменти для відлагодження та тестування коду [11].

2.2 Вибір методів реалізації знижок

У процесі розробки інтернет-магазину з інноваційною системою генерації випадкових персональних знижок важливо проаналізувати різні методи вирішення цієї проблеми.

Метод 1: Використання генераторів випадкових чисел.

Генерація випадкових знижок може бути реалізована за допомогою генераторів випадкових чисел, що створюють випадкові значення, які визначають розмір знижки. Генератори випадкових чисел можуть бути реалізовані за допомогою простих алгоритмів.

Переваги:

- генератори випадкових чисел легко впроваджуються в систему та не потребують значних ресурсів;
- випадкові числа генеруються швидко, що забезпечує ефективну роботу системи знижок;
- генератори дозволяють налаштовувати діапазон знижок, що робить систему гнучкою.

Недоліки:

- повна випадковість може призвести до нерівномірного розподілу знижок серед користувачів, що може бути несправедливо для деяких клієнтів;
- метод не враховує індивідуальні характеристики та поведінку користувачів.

Метод 2: Використання алгоритмів машинного навчання.

Алгоритми машинного навчання можуть використовуватися для аналізу поведінки користувачів та генерації персональних знижок.

Переваги:

- алгоритми дозволяють враховувати індивідуальні характеристики користувачів, що підвищує ефективність знижкових програм;
- моделі можуть налаштовуватись для максимізації продажів та підвищення лояльності клієнтів.

Недоліки:

- використання алгоритмів машинного навчання вимагає значних технічних знань та ресурсів;
- необхідність збору великої кількості даних про користувачів, що може викликати проблеми з конфіденційністю.

Метод 3: Використання комбінованого підходу.

Комбінований підхід включає використання генераторів випадкових чисел для базової генерації знижок з подальшою корекцією результатів за допомогою алгоритмів машинного навчання.

Переваги:

- комбінований підхід дозволяє зберегти елемент випадковості, при цьому враховуючи індивідуальні характеристики користувачів;
- система може бути налаштована для досягнення оптимального балансу між випадковістю та персоналізацією;
- висока ефективність: Комбінований підхід дозволяє максимізувати ефективність знижкових програм та підвищити лояльність клієнтів.

Недоліки:

- реалізація комбінованого підходу потребує значних технічних знань та ресурсів;
- необхідність збору великої кількості даних про користувачів для навчання моделей;
- комбінований підхід може вимагати більше часу на реалізацію та навчання моделей.

Для кваліфікаційної роботи, враховуючи обмежені ресурси та час, доцільно використовувати просту генерацію випадкових чисел. Цей метод є найменш складним у реалізації та не вимагає великих обсягів даних або складних моделей. Крім того, він добре підходить для інтернет-магазину, що спеціалізується на комп'ютерних запчастинах, оскільки користувачі рідко замовляють однакові товари повторно. Наприклад, якщо користувач вже

замовив материнську плату, ймовірність того, що він замовить її знову, дуже низька.

2.3 Створення діаграми потоків даних

Для створення діаграми було використано програму ERwin Process Modeler. Це потужний інструмент для моделювання бізнес-процесів, що дозволяє візуалізувати, аналізувати та оптимізувати процеси.

Діаграма для опису роботи унікальних випадкових знижок (див. рис. 2.1):

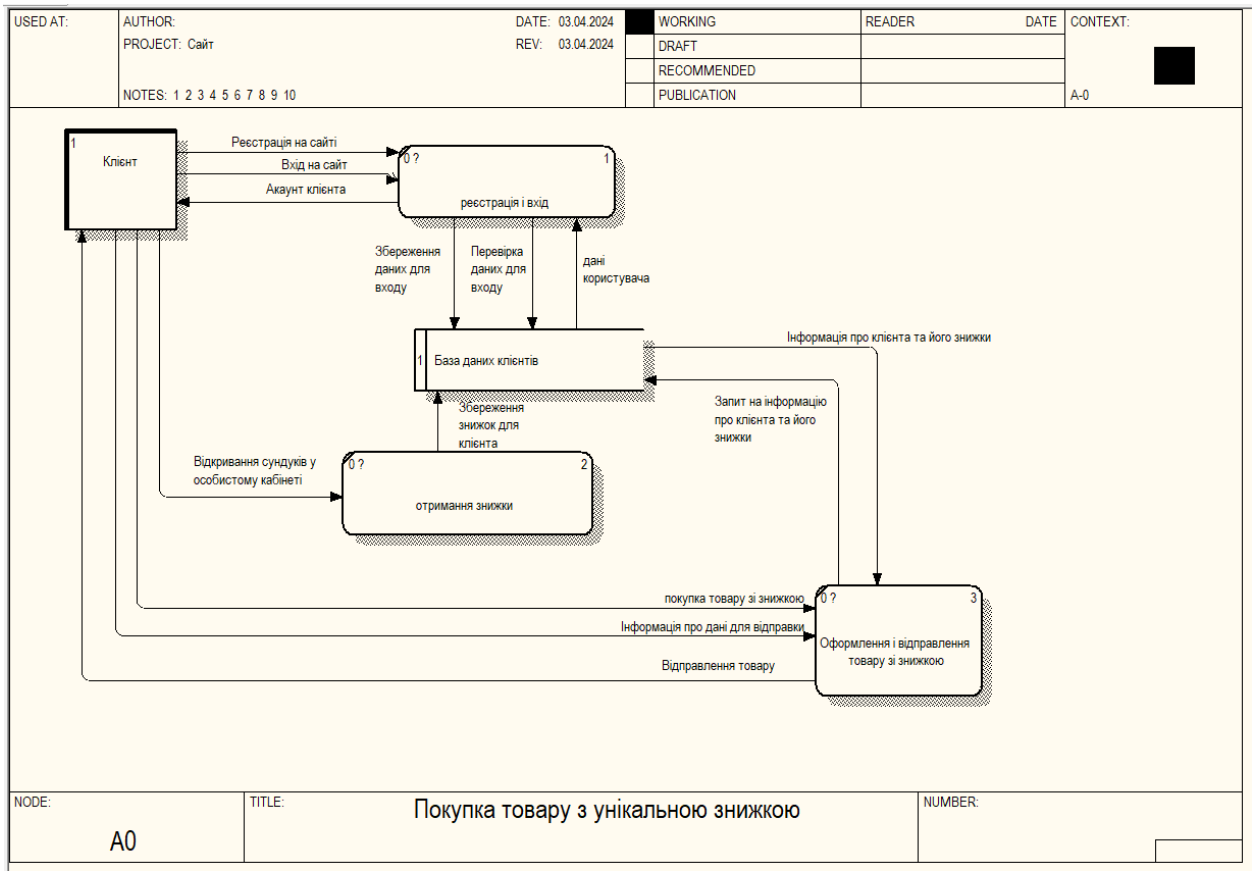

Рисунок 2.1 – Діаграма потоків даних

У моделі процесу генерації випадкових знижок передбачено такі етапи:

- реєстрація користувача: новий користувач реєструється в системі;
- запис у БД: дані користувача зберігаються в базі даних;

- авторизація користувача: користувач входить у систему, використовуючи свої облікові дані;
- дані з БД: витягуються дані користувача з бази даних для подальших дій;
- генерація знижки: система генерує випадкову знижку для користувача;
- запис у БД: згенерована знижка зберігається в базі даних;
- оформлення замовлення: користувач оформлює замовлення, використовуючи надану знижку;
- дані про знижки з БД: витягуються дані про знижки з бази даних для аналізу або звітності.

2.4 Опис алгоритмів роботи інтернет-магазину

Основні алгоритми роботи магазину:

Реєстрація користувача:

- користувач заповнює форму реєстрації;
- перевірка унікальності електронної пошти;
- збереження даних нового користувача в базу даних.

Авторизація користувача:

- користувач вводить електронну пошту та пароль;
- перевірка даних користувача в базі даних;
- авторизація користувача та збереження сесії.

Відображення списку продуктів:

- запит до бази даних для отримання списку продуктів;
- відображення продуктів на відповідних сторінках (материнські плати, процесори, ОЗУ, тощо).

Генерація знижок:

- вибір випадкових продуктів для користувача;
- генерація знижок від 5% до 20%;
- збереження даних про знижки в профілі користувача;
- відображення знижок у профілі користувача;

- отримання даних про знижки користувача з бази даних;
- відображення продуктів зі знижками у особистому кабінеті користувача.

Генератор випадкових знижок:

- вибір трьох випадкових продуктів з бази даних;
- генерація випадкового відсотка знижки для кожного продукту (від 5% до 20%);
- збереження інформації про обрані продукти та їх знижки у профіль користувача.

Реалізація логіки кошика покупок та обробки замовлень:

- додавання товару до кошика при кліканні на кнопку «Додати до кошика»;
- видалення товару з кошика при кліканні на відповідну кнопку;
- перегляд вмісту кошика для коректного відображення обраних товарів та їх сумарної вартості.

Форма для підтвердження замовлення:

- користувач може ввести свої дані для доставки, такі як ім'я, прізвище, номер телефону, електронну пошту, місто та відділення Нової Пошти;
- для міста та відділення Нової Пошти доступна функція автодоповнення, яка дозволяє вибрати потрібні значення зі списку;
- після заповнення форми користувач може натиснути кнопку «Підтвердити замовлення», яка відправить дані на сервер для обробки.

Відображення вмісту кошика та сумарної вартості:

- у правій частині сторінки відображається список товарів, які додані до кошика;
- кожен товар має кнопку «Видалити», за допомогою якої можна видалити товар з кошика без перезавантаження сторінки;
- нижче відображається загальна сума товарів у кошику.
- Скидання всіх знижок:
- заплановане скидання всіх знижок.

Відображення таймеру:

- розрахунок часу до наступного оновлення: Спочатку потрібно визначити час до наступного оновлення даних. Наприклад, якщо наступне оновлення заплановане на 2:00 ночі, а зараз 23:30, то таймер повинен показувати, що до оновлення залишилося 2 години 30 хвилин;
- оновлення відображення таймеру: Після розрахунку часу до наступного оновлення необхідно відобразити цей час на сторінці. Таймер може бути в форматі «години:хвилини:секунди» або «2 год 30 хв 15 сек».

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Бібліотеки та компоненти

Підключення бібліотек та компонентів:

```
from flask import Flask, redirect, render_template, url_for,
request, session, jsonify
from flask_sqlalchemy import SQLAlchemy
import random
import threading
from flask_login import LoginManager, login_user, logout_user,
login_required, current_user, UserMixin
import schedule
import time
import requests
```

- Flask: створює екземпляр додатку;
- redirect: перенаправляє користувача на іншу URL-адресу;
- render_template: рендерить HTML шаблон з переданими даними;
- url_for: генерує URL-адреси для статичних файлів та маршрутів;
- request: дозволяє доступ до даних запиту (GET або POST параметри);
- session: зберігає дані про сесію користувача між запитами;
- jsonify: повертає JSON відповідь;
- SQLAlchemy: створює екземпляр SQLAlchemy для роботи з базою даних;
- random – модуль для генерації випадкових чисел;
- random.randint: генерує випадкове ціле число в заданому діапазоні;
- threading – модуль для роботи з потоками, що дозволяє виконувати кілька завдань одночасно;
- threading.Thread: створює новий потік для виконання функції;
- Flask-Login – розширення для керування авторизацією користувачів;
- LoginManager: створює екземпляр менеджера логінів;
- login_user: авторизує користувача;
- logout_user: виходить з системи;
- login_required: декоратор для захисту маршрутів, які вимагають авторизації;

- current_user: доступ до поточного авторизованого користувача;
- UserMixin: змішує основні методи та властивості для моделей користувачів;
- schedule – модуль для планування завдань;
- schedule.every: встановлює інтервал для виконання завдань;
- schedule.run_pending: виконує заплановані завдання;
- time – модуль для роботи з часом;
- requests – модуль для виконання HTTP-запитів;
- requests.get: виконує GET запит до вказаного URL.

Ці бібліотеки разом забезпечують широкий набір інструментів для створення функціонального веб-додатку з авторизацією користувачів, взаємодією з базою даних, обробкою запитів та іншими можливостями.

3.2 Створення бази даних

База даних створювалась за допомогою додатку Data Base Brouser for SQlite. В цьому додатку були створенні таблиці та зв’язки між ними. Також була заповнена таблиця products.db (див. рис. 3.1).

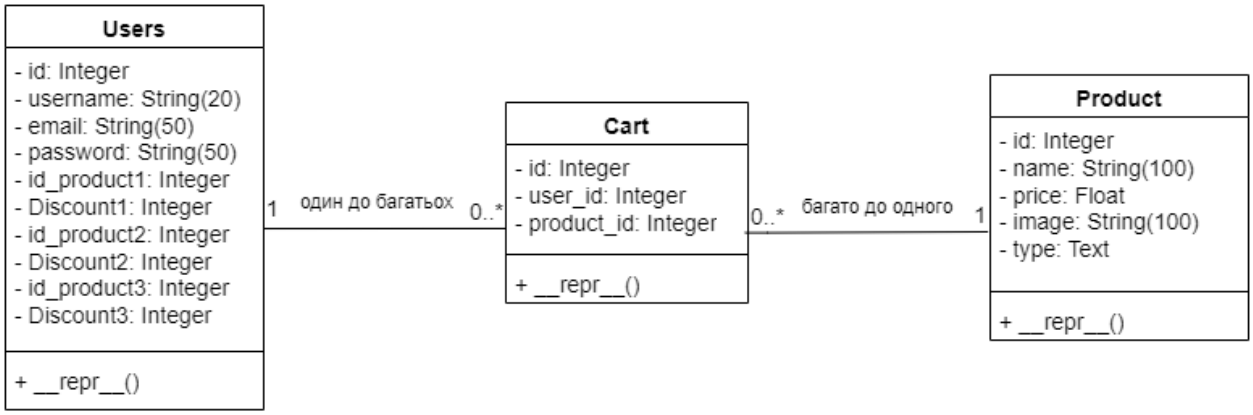

Рисунок 3.1 – Діаграма зв’язків між таблицями у базі даних

Ініціалізація бази даних на сервері:

```
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///products.db'
db = SQLAlchemy(app)
```

Створення класів для таблиць:

Клас користувачів:

```
class Users(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(20), nullable=False)
    email = db.Column(db.String(50), nullable=False,
unique=True)
    password = db.Column(db.String(50), nullable=False)
    id_product1 = db.Column(db.Integer, nullable=True)
    Discount1 = db.Column(db.Integer, nullable=True)
    id_product2 = db.Column(db.Integer, nullable=True)
    Discount2 = db.Column(db.Integer, nullable=True)
    id_product3 = db.Column(db.Integer, nullable=True)
    Discount3 = db.Column(db.Integer, nullable=True)
```

Клас корзини:

```
class Cart(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'),
nullable=False)
    product_id = db.Column(db.Integer,
db.ForeignKey('product.id'), nullable=False)

    user = db.relationship('Users', backref=db.backref('cart',
lazy=True))
    product = db.relationship('Product',
backref=db.backref('cart', lazy=True))

    def __repr__(self):
        return f'<Cart {self.id} - User {self.user_id} - Product
{self.product_id}>'
```

Клас товарів:

```
class Product(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100))
    price = db.Column(db.Float)
    image = db.Column(db.String(100))
    type = db.Column(db.Text)
```

Приклад звернення до бази даних:

```
@app.route('/')
def index():
    products = Product.query.all()
    return render_template("cooling.html", products=products)
```

Ця функція виконує наступні дії:

```
products = Product.query.all()
```

Виконує запит до бази даних для отримання всіх записів з таблиці Product. Це використовує SQLAlchemy для взаємодії з базою даних. Product тут є моделлю, яка представляє таблицю в базі даних.

Результат цього запиту зберігається в змінній products.

Рендеринг шаблону:

```
return render_template("cooling.html", products=products)
```

рендерить HTML-шаблон cooling.html і передає йому змінну products.

Шаблон cooling.html використовує передані дані (products) для динамічного відображення інформації про продукти на веб-сторінці.

3.3 Генерація випадкової знижки

Генерація випадкової знижки знаходиться в особистому кабінеті користувача на сторінці зображений сундук з кнопкою (див. рис. 3.2):

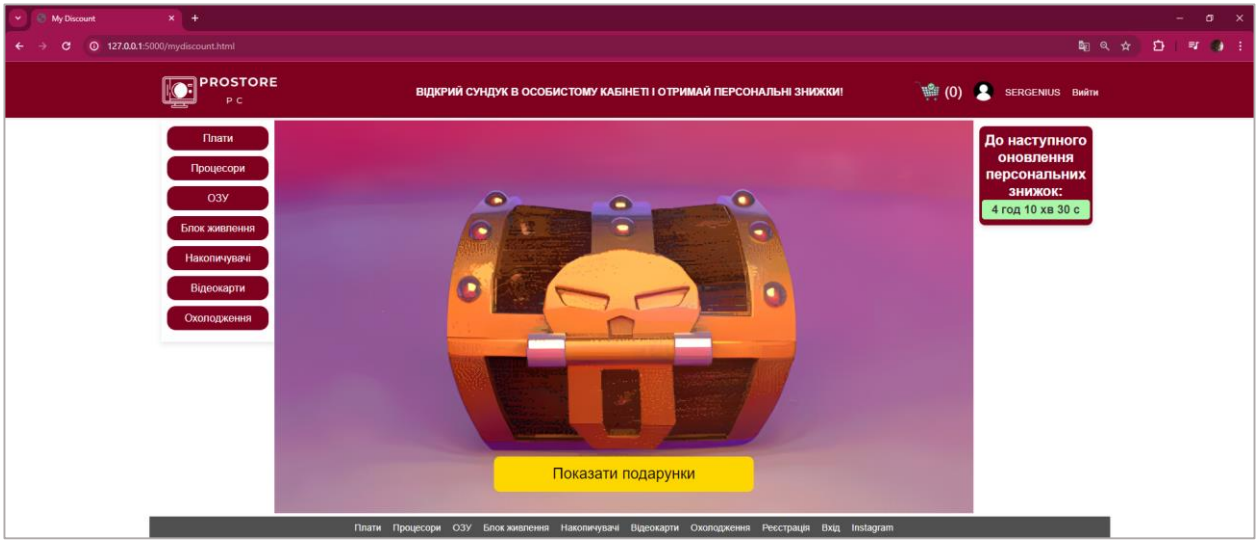

Рисунок 3.2 – Сторінка особистого кабінету

Натиснувши на кнопку «Відкрити сундук» спрацювують функції showGiftModal() та updateUser().

Функція showGiftModal() (див. рис. 3.3).

Всередині цієї функції модальне вікно з ідентифікатором loadingModal відображається, змінюючи його стиль на block, що робить його видимим. Це модальне вікно показує GIF з сундуком.

Використовуючи setTimeout, функція налаштовується так, щоб зачекати 6 секунд перед виконанням наступних дій. Це створює враження відкриття сундуку у режимі реального часу.

Після затримки завантажувальне модальне вікно (loadingModal) приховується, змінюючи його стиль на none.

Модальне вікно з подарунками (giftModal) відображається, змінюючи його стиль на block.

```
function showGiftModal() {
    const loadingModal = document.getElementById("loadingModal");
    loadingModal.style.display = "block";

    setTimeout(function () {
        loadingModal.style.display = "none";
        const giftModal = document.getElementById("giftModal");
        giftModal.style.display = "block";
        updateUser();
    }, 6000); // Зачекати 6 секунд перед показом подарунків
}
```

Рисунок 3.3 – Функція showGiftModal()

Оновлення інформації про користувача:

Схема алгоритму генерації випадкової знижки (див. рис. 3.4).

- викликається функція updateUser() (див. рис. 3.5), яка відправляє POST-запит на серверний маршрут /update_user_info;
- обробка серверу (див. рис. 3.6):

    1) перевіряється, чи поточний користувач аутентифікований за допомогою current_user.is_authenticated. Якщо користувач не аутентифікований, повертається JSON-відповідь з помилкою та статус-кодом 401;

2) отримуються всі продукти з бази даних за допомогою Product.query.all(). Створюється порожня множина selected_product_ids для відстеження вибраних продуктів;

3) для кожного з трьох продуктів виконується цикл while True, який вибирає випадковий продукт з products за допомогою random.choice(products);

4) перевіряється, чи ID цього продукту ще не було вибрано раніше;

5) якщо продукт не було вибрано, його ID додається до selected_product_ids, і цикл while завершується;

6) присвоюється випадкова знижка в межах від 5% до 20% за допомогою random.randint(5, 20);

7) використовується функція setattr для встановлення значень атрибутів id_product{i} та Discount{i} для поточного користувача;

8) у словнику data зберігається інформація про вибрані продукти, їх зображення, назви, ціни та знижки;

9) викликається db.session.commit() для збереження змін у базі даних;

10) до словника data додається ключ success з значенням True;

11) повертається JSON-відповідь зі словником data.

- обробка відповіді від сервера:

1) якщо сервер повертає відповідь з параметром success: true, виконується цикл для оновлення елементів модального вікна з подарунками;

2) цикл проходить через три подарунки (з індексами 1, 2 та 3) і оновлює зображення, назву, оригінальну ціну, знижку та ціну зі знижкою для кожного подарунка, використовуючи дані з відповіді сервера;

3) модальне вікно giftModal тепер показує оновлені дані про подарунки, включаючи зображення, назву продукту, оригінальну ціну, знижку та ціну зі знижкою (див. рис. 3.7).

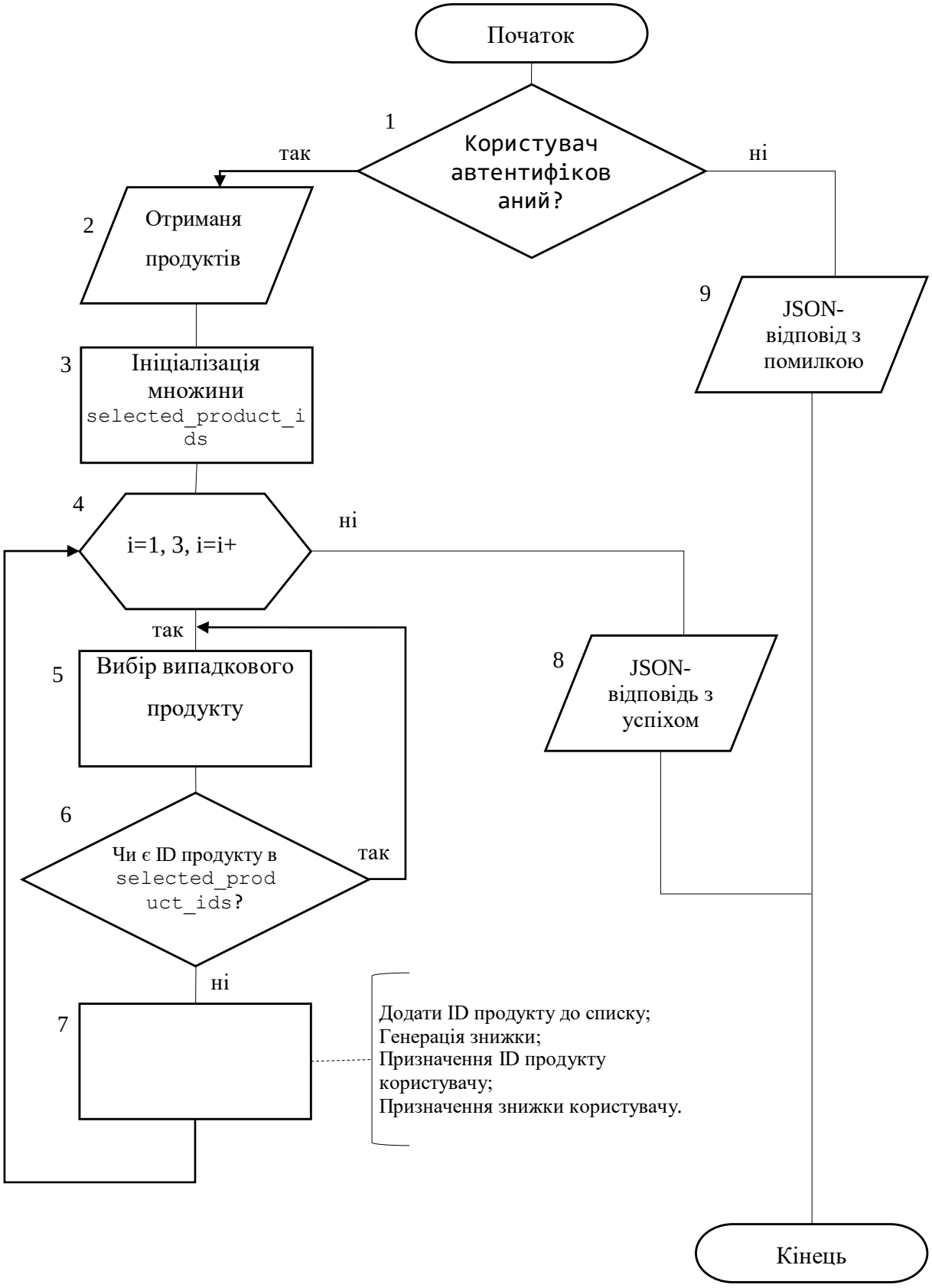

Рисунок 3.4 – Схема алгоритму генерації випадкової знижки

```
function updateUser() {
    fetch("/update_user_info", {
        method: "POST",
        headers: {
            "Content-Type": "application/json",
        },
        body: JSON.stringify({}),
    })
        .then((response) => response.json())
        .then((data) => {
            if (data.success) {
                console.log("User info updated for gifts");
                for (let i = 1; i <= 3; i++) {
                    document.getElementById(
                        `giftImage${i}`
                    ).src = `/static/img/${data[`product_image${i}`]}`;
                    document.getElementById(`giftName${i}`).innerText =
                        data[`product_name${i}`];
                    document.getElementById(`orgPrice${i}`).innerText = `${
                        data[`prise${i}`]
                    }₴`;
                    document.getElementById(
                        `discountPrice${i}`
                    ).innerText = `${(
                        (data[`prise${i}`] * (100 - data[`discount${i}`])) /
                        100
                    ).toFixed(2)}₴`;
                    document.getElementById(
                        `discount${i}`
                    ).innerText = `Знижка ${data[`discount${i}`]}%!`;
                }
            } else {
                console.error("Error updating user info for gifts");
            }
        })
        .catch((error) => {
            console.error("Error:", error);
        });
}
```

Рисунок 3.5 – Функція updateUser()

```python
@app.route('/update_user_info', methods=['POST'])
def update_user_info():
    if not current_user.is_authenticated:
        return jsonify(success=False, error="User not authenticated"), 401

    products = Product.query.all()
    selected_product_ids = set()  # Множина для відстеження вибраних ID продуктів
    data = {}

    for i in range(1, 4):
        while True:
            random_product = random.choice(products)
            if random_product.id not in selected_product_ids:
                selected_product_ids.add(random_product.id)
                break

        discount = random.randint(5, 20)

        setattr(current_user, f"id_product{i}", random_product.id)
        setattr(current_user, f"Discount{i}", discount)

        data[f"product_image{i}"] = random_product.image
        data[f"product_name{i}"] = random_product.name
        data[f"prise{i}"] = random_product.price
        data[f"discount{i}"] = discount

    db.session.commit()

    data["success"] = True
    return jsonify(data)
```

Рисунок 3.6 – Серверна функція /update_user_info

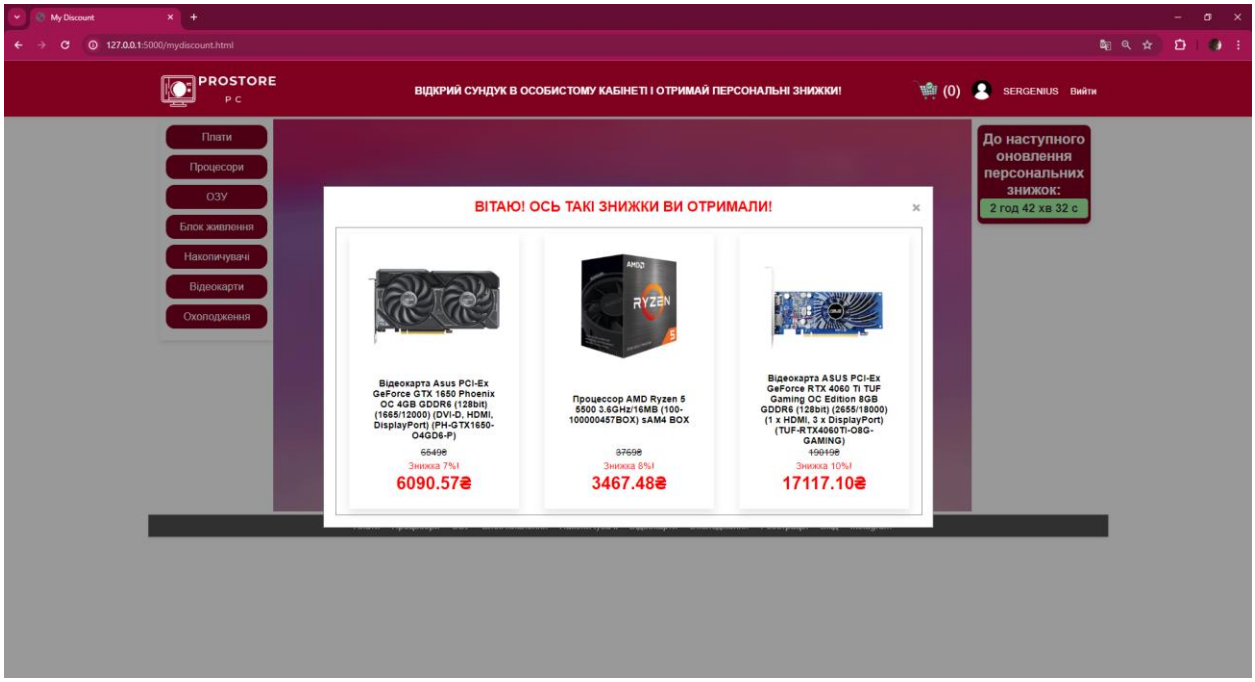

Рисунок 3.7 – Модальне вікно яке показує знижку

3.4 Реалізація функції скидання всіх знижок

Опис функцій для скидання даних користувачів та їх планування:

```
def reset_data():
    with app.app_context():
        users = Users.query.all()
        for user in users:
            user.id_product1 = None
            user.id_product2 = None
            user.id_product3 = None
            user.Discount1 = None
            user.Discount2 = None
            user.Discount3 = None
        db.session.commit()

# Функція для запуску запланованих завдань у окремому потоці
def run_scheduler():
    while True:
        schedule.run_pending()
        time.sleep(1)

# Планування виконання завдання щодня о 2:00 ночі
schedule.every().day.at("02:00").do(reset_data)

# Запуск планувальника у окремому потоці
scheduler_thread = threading.Thread(target=run_scheduler)
scheduler_thread.daemon = True
scheduler_thread.start()
```

Рисунок 3.8 – Функція скидання знижок

Функція reser_data():

- використовується with app.app_context(), щоб виконувати операції у контексті Flask додатка;
- всі користувачі отримуються з бази даних за допомогою Users.query.all();
- для кожного користувача встановлюються значення атрибутів id_product1, id_product2, id_product3, Discount1, Discount2 та Discount3 в None;
- викликається db.session.commit() для збереження змін у базі даних.

Функція run_scheduler():

- функція запускає безперервний цикл while True;

- викликається schedule.run_pending() для виконання всіх запланованих завдань;
- використовується time.sleep(1), щоб зупинити виконання на 1 секунду між перевірками наявності запланованих завдань;
- використовується schedule.every().day.at("02:00").do(reset_data), щоб планувати виконання функції reset_data() кожного дня о 2:00 ночі;
- створюється новий потік scheduler_thread для виконання функції run_scheduler();
- встановлюється scheduler_thread.daemon = True, щоб потік завершувався разом із завершенням головного потоку додатка;
- викликається scheduler_thread.start() для запуску потоку.

3.5 Створення таймеру що веде відлік до оновлення всіх знижок

Реалізація у коді (див. рис. 3.9):

```
<script>
    function calculateTimeUntilNextUpdate() {
        const now = new Date();
        let nextUpdate = new Date();
        nextUpdate.setHours(2, 0, 0, 0); // Встановити час на 2:00:00

        if (now.getHours() >= 2) {
            nextUpdate.setDate(now.getDate() + 1); // Перейти на наступний день, якщо вже після 2 години ночі
        }

        const diff = nextUpdate - now;
        const hours = Math.floor(diff / (1000 * 60 * 60));
        const minutes = Math.floor(
            (diff % (1000 * 60 * 60)) / (1000 * 60)
        );
        const seconds = Math.floor((diff % (1000 * 60)) / 1000);

        return { hours, minutes, seconds };
    }

    function updateCountdown() {
        const { hours, minutes, seconds } =
            calculateTimeUntilNextUpdate();
        document.getElementById(
            "countdown"
        ).innerHTML = `${hours} год ${minutes} хв ${seconds} с`;
    }

    setInterval(updateCountdown, 1000);
    updateCountdown(); // Початковий виклик для негайного відображення таймера
</script>
```

Рисунок 3.9 – Реалізація таймеру

Опис JavaScript функцій для відліку часу до наступного оновлення

Функція calculateTimeUntilNextUpdate() обчислює час до наступного оновлення, яке відбудеться о 2:00 ночі.

- створюється об'єкт now, що представляє поточний час;
- створюється об'єкт nextUpdate, що також представляє поточний час, але буде змінений на час наступного оновлення;
- використовується метод setHours(2, 0, 0, 0), щоб встановити час для nextUpdate на 2:00:00;
- якщо поточний час вже пізніше 2:00, встановлюється наступний день для nextUpdate, збільшуючи поточну дату на 1 (nextUpdate.setDate(now.getDate() + 1));
- різниця в мілісекундах обчислюється як diff = nextUpdate – now;
- перетворення різниці у години, хвилини та секунди:
    1) години: Math.floor(diff / (1000 * 60 * 60));
    2) хвилини: Math.floor((diff % (1000 * 60 * 60)) / (1000 * 60));
    3) секунди: Math.floor((diff % (1000 * 60)) / 1000).
- повертається об'єкт з трьома властивостями: hours, minutes, seconds.

Функція updateCountdown()

Ця функція оновлює відлік часу до наступного оновлення на веб-сторінці.

- виклик функції calculateTimeUntilNextUpdate();
- отримуються обчислені години, хвилини та секунди для наступного оновлення;
- оновлення елемента HTML;
- знаходиться елемент з ідентифікатором countdown і його вміст оновлюється за допомогою innerHTML;
- вміст оновлюється на форматований рядок: ${hours} год ${minutes} хв ${seconds} с.

Виклик функцій:

- викликається setInterval(updateCountdown, 1000) для оновлення відліку часу кожну секунду;
- викликається updateCountdown() одразу після визначення інтервалу, щоб показати початковий відлік без затримки.

Підсумок:

Цей скрипт забезпечує автоматичний відлік часу до наступного оновлення о 2:00 ночі. Функція calculateTimeUntilNextUpdate() обчислює залишок часу до 2:00 наступного дня (або сьогодні, якщо ще не настала 2:00). Функція updateCountdown() оновлює цей відлік на сторінці кожну секунду, забезпечуючи безперервне відображення часу, що залишився.

3.6 Реалізація кошика

Кожен товар має кнопку, яка додає товар до кошика. Після натискання на іконку кошика відображається модальне вікно з товарами (див. рис. 3.10), що додані до кошика. У кошику користувач може побачити загальну суму замовлення та видалити товар, який він передумав купувати.

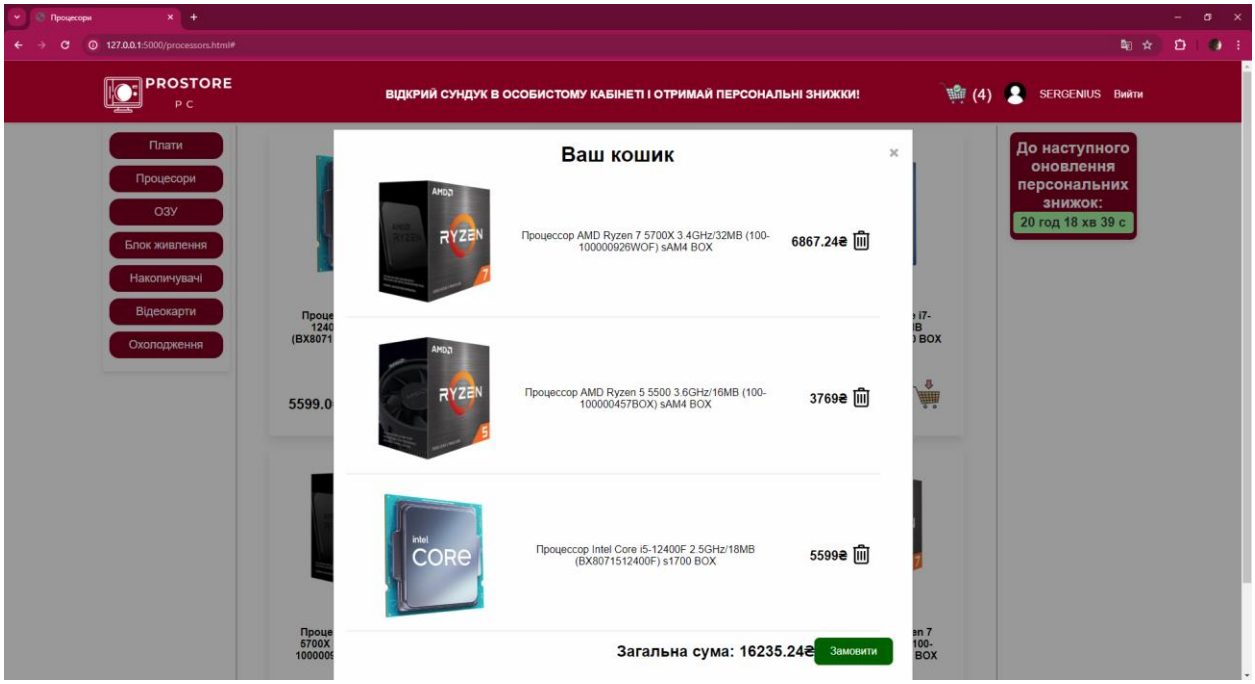

Рисунок 3.10 – Відображення кошика

Функції які забезпечують роботу кошика:

Додавання товару до кошика (див. рис. 3.11):

```python
@app.route('/add_to_cart/<int:product_id>', methods=['POST'])
@login_required
def add_to_cart(product_id):
    product = Product.query.get_or_404(product_id)
    cart_item = Cart.query.filter_by(user_id=current_user.id, product_id=product.id).first()

    new_cart_item = Cart(user_id=current_user.id, product_id=product.id)
    db.session.add(new_cart_item)

    db.session.commit()
    return jsonify(success=True, message="Товар додано до кошика")
```

Рисунок 3.11 – Функція додавання товару у кошик

Опис функції:

- отримання товару за ID;
- створюється новий об'єкт Cart;
- збереження змін у базі даних;
- JSON-відповідь з повідомленням про успішне додавання.

Видалення товару з кошика (див. рис. 3.12):

```python
@app.route('/remove_from_cart/<int:cart_item_id>', methods=['POST'])
@login_required
def remove_from_cart(cart_item_id):
    cart_item = Cart.query.get_or_404(cart_item_id)
    if cart_item.user_id != current_user.id:
        return jsonify(success=False, message="Несанкціоновано"), 403

    db.session.delete(cart_item)
    db.session.commit()
    return jsonify(success=True, message="Товар видалено з кошика")
```

Рисунок 3.12 – Функція видалення товару з кошика

Опис функції:

- отримання товару за ID;
- перевірка, чи користувач є власником товару;
- видалення товару з кошика;

- збереження змін у базі даних;
- JSON-відповідь з повідомленням про успішне видалення.

Перегляд кошика (див. рис. 3.13):

```python
@app.route('/view_cart')
@login_required
def view_cart():
    cart_items = Cart.query.filter_by(user_id=current_user.id).all()
    cart_data = []

    for item in cart_items:
        product = Product.query.get(item.product_id)
        price = product.price
        if item.product_id == current_user.id_product1:
            price -= price * (current_user.Discount1 / 100)
        elif item.product_id == current_user.id_product2:
            price -= price * (current_user.Discount2 / 100)
        elif item.product_id == current_user.id_product3:
            price -= price * (current_user.Discount3 / 100)

        # Округлення ціни до двох знаків після коми
        price = round(price, 2)

        cart_data.append({
            'id': item.id,
            'product_id': product.id,
            'name': product.name,
            'price': price,
            'image': product.image
        })

    total_price = sum(item['price'] for item in cart_data)
    total_price = round(total_price, 2)  # Округлення загальної ціни до двох знаків після коми

    return jsonify(cart_data=cart_data, total_price=total_price)
```

Рисунок 3.13 – Функція перегляду кошика

Опис функції:

- отримання товарів у кошику користувача;
- знижка застосовується до відповідних товарів;
- створення списку cart_data з інформацією про товари;
- сума цін всіх товарів у кошику;
- JSON-відповідь з даними про кошик та загальну вартість.

Серверні функції викликаються за допомогою JavaScript коду, щоб динамічно оновлювати вміст кошика та взаємодіяти з користувачем.

3.7 Інтерфейс сторінки та функції реєстрації та авторизації користувачів

Всі сторінки мають спільний шаблон Base.html в якому в залежності від відкритої сторінки змінюється контент сайту та назва сторінки. Це все реалізовано за допомогою системи шаблонів Jinja2 у Flask, яка дозволяє розширювати базовий шаблон і динамічно змінювати вміст та назву сторінки.

Базовий шаблон base.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>{% block title %}Default Title{% endblock
%}</title>
    <link rel="stylesheet" href="{{ url_for('static',
filename='styles.css') }}">
</head>
<body>
    <header>
            <!-- Навігаційне меню -->
    </header>
    <main>
        {% block body %}{% endblock %}
    </main>
    <footer>
        <!-- Підвал сайту -->
    </footer>
</body>
</html>
```

Шаблон для конкретної сторінки:

```
{% extends 'base.html' %}

{% block title %}
    Системи охолодження
{% endblock %}

{% block body %}
    <!-- Додатковий контент, як-от таблиці, зображення та
інше -->
{% endblock %}
```

Для зміни контенту шапки сайту використовуються сесії Flask для зберігання інформації про користувача після входу на сайт. Це дозволяє динамічно змінювати вміст шапки сайту в залежності від того, чи авторизований користувач.

Інтерфейс сторінки (див. рис. 3.14):

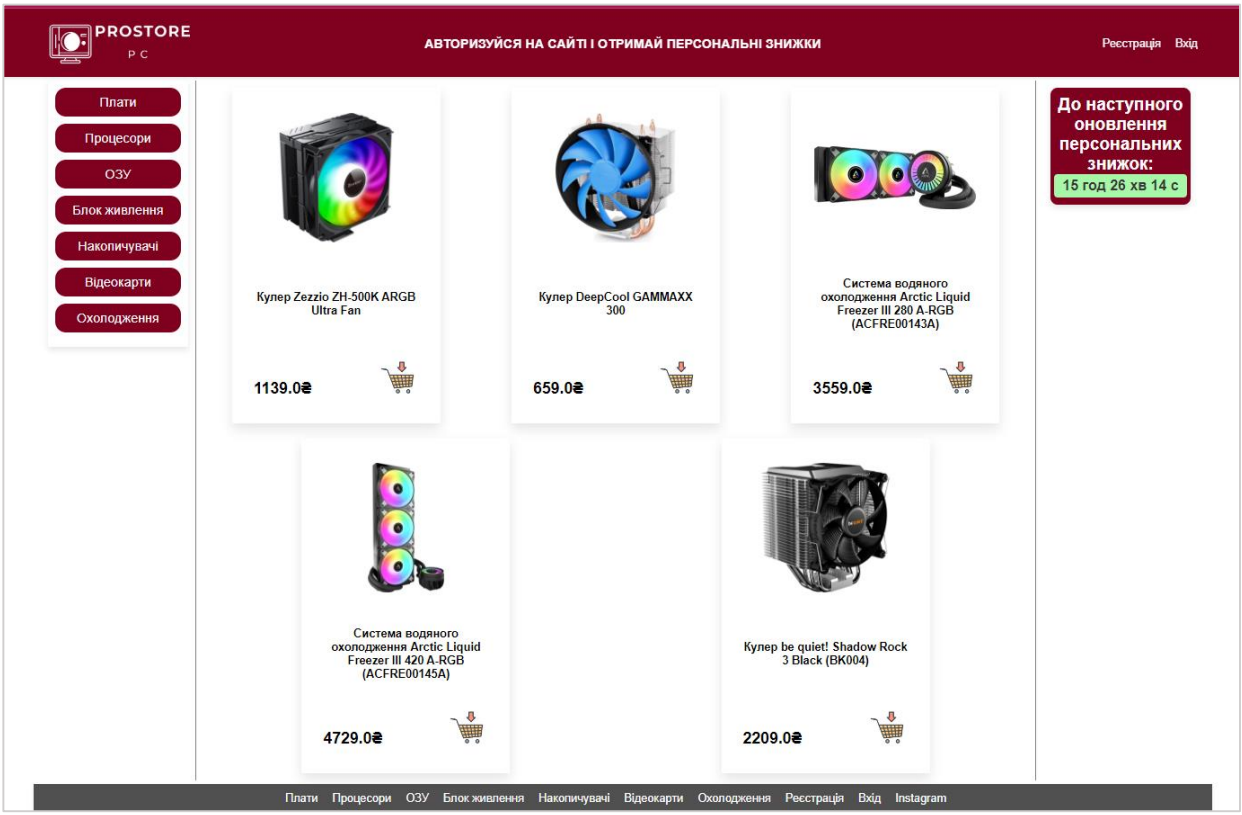

Рисунок 3.14 – Зображення сторінки сайту

Шапка сайту має:

- логотип магазину;
- назва магазину;
- банер який вказує на можливість отримання унікальної знижки після реєстрації;
- кнопки для авторизації та реєстрації користувачів.

Область контенту:

- меню категорій товарів з посиланнями на відповідні розділи сайту;
- таймер відліку до оновлення знижок;

- список товарів, що включає в себе фотографії товарів, назви товарів, ціну товарів та кнопку додання у кошик.

Підвал сайту:

- має посилання на всі сторінки та соціальні мережі.

Реєстрація користувача (див. рис. 3.15).

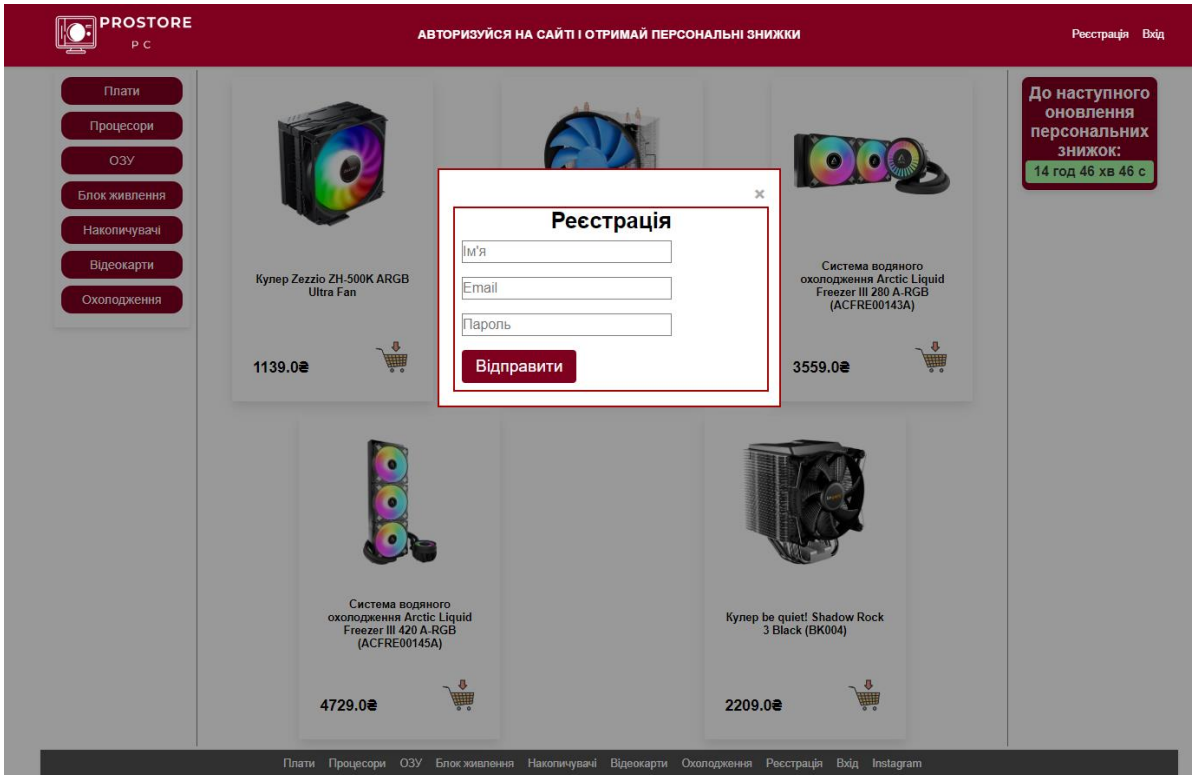

Рисунок 3.15 – Форма реєстрації нового користувача

Функція registration обробляє реєстрацію користувача на веб-сайті. Вона приймає POST запити та виконує наступні кроки:

- збирає дані з форми: ім'я користувача, електронну пошту та пароль;
- перевіряє, чи вже існує користувач з такою електронною поштою:

  1) виконує запит до бази даних, щоб знайти користувача з введеною електронною поштою;

  2) якщо такий користувач існує, повертає повідомлення про те, що ця електронна пошта вже використовується.
- створює новий об'єкт користувача з введеними даними;
- проводить спробу зберегти новий об'єкт користувача в базі даних:

1) якщо збереження пройшло успішно, перенаправляє користувача на головну сторінку;

2) якщо виникла помилка під час збереження, повертає повідомлення про помилку з текстом винятку.

Детальний опис коду:

Ця декорація вказує, що функція registration обробляє запити на URL /registration і приймає POST запити.

```
@app.route('/registration', methods=['POST'])
```

Отримання даних з форми реєстрації.

```
name = request.form['name'], mail = request.form['mail'],
password = request.form['password']
```

Перевірка, чи вже існує користувач з введеною електронною поштою.

```
existing_user = Users.query.filter_by(email=mail).first()
```

Якщо користувач з такою електронною поштою існує, повертається повідомлення про помилку.

```
if existing_user: return «Ця електронна пошта вже
використовується. Будь ласка, введіть іншу адресу».
```

Створення нового об'єкта користувача.

```
user = Users(username=name, email=mail, password=password)
```

Додавання нового користувача до сесії бази даних і збереження змін у базі даних.

```
db.session.add(user), db.session.commit()
```

Перенаправлення на головну сторінку після успішної реєстрації.

```
return redirect(url_for('index'))
```

Обробка винятків під час збереження даних у базі даних і повернення повідомлення про помилку.

```
except Exception as e: return f"Виникла помилка: {str(e)}"
```

Авторизація користувача (див. рис. 3.16):

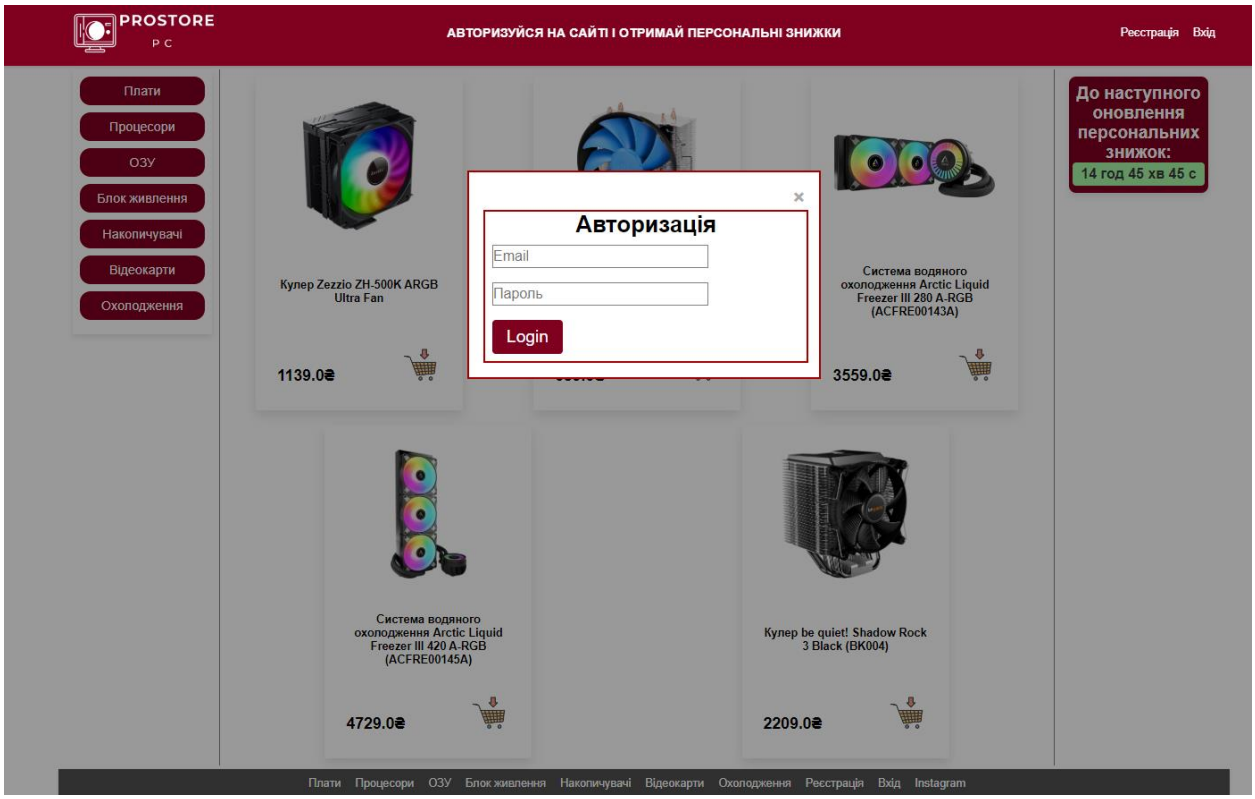

Рисунок 3.16 Авторизація користувача

Функція login обробляє аутентифікацію користувача на веб-сайті. Вона приймає тільки POST запити і виконує наступні кроки:

- збирає дані з форми: електронну пошту та пароль;
- виконує запит до бази даних, щоб знайти користувача з введеною електронною поштою та паролем;
- якщо такий користувач існує, зберігає інформацію про користувача в сесії та викликає функцію login_user, щоб здійснити вхід;
- якщо користувача не знайдено або введені дані невірні, повертає повідомлення про помилку.

Детальний опис коду

Ця декорація вказує, що функція login обробляє запити на URL /login і приймає тільки POST запити.

```
@app.route('/login', methods=['POST'])
```

Отримання даних з форми входу:

```
mail = request.form['mail'], password =
request.form['password']
```

Перевірка, чи існує користувач з введеною електронною поштою та паролем. Виконання запиту до бази даних для пошуку відповідного користувача:

```
user = Users.query.filter_by(email=mail,
password=password).first()
```

Якщо користувача знайдено, збереження інформації про користувача в сесії: ідентифікатор користувача та ім'я користувача.

```
session['user_id'] = user.id, session['username'] =
user.username
```

Виклик функції login_user для здійснення входу користувача:

```
login_user(user)
```

Перенаправлення на головну сторінку після успішного входу:

```
return redirect('/')
```

Якщо користувача не знайдено або введені дані невірні: повернення повідомлення про помилку, якщо електронна пошта або пароль невірні.

```
return "Invalid username or password"
```

Реалізація виходу з аккаунту:

Функція logout обробляє вихід користувача з веб-сайту. Вона видаляє інформацію про користувача з сесії, викликає функцію logout_user, щоб завершити сесію користувача, і перенаправляє користувача на головну сторінку.

3.8 Тестування додатку

Під час розробки веб-сайту було проведено ретельне тестування, яке включало в себе наступні кроки:

Для кожної функціональності системи було розроблено детальні тестові сценарії, що охоплюють усі можливі варіанти використання та граничні випадки.

Кожна функція та алгоритм сайту були протестовані вручну під час його створення. Кожен тестовий сценарій виконувався крок за кроком.

- перевірка успішної реєстрації, обробка помилок при некоректних даних;
- перевірка успішного входу та обробка помилок при некоректних даних;
- тестування відображення списку продуктів, пошуку та фільтрації;
- перевірка генерації знижок та їх відображення на сторінці;
- перевірка функції скидання знижки;
- додавання товарів до кошика: перевірка додавання, перегляду та видалення товарів з кошика.

Оперативно виправлялись всі знайдені помилки.

Після кожного виправлення було проведено повторне тестування для підтвердження виправлень та перевірки, чи не з'явилися нові помилки.

Регресійне тестування виконувалося для перевірки того, що внесені виправлення не вплинули негативно на інші частини системи.

В результаті всіх цих кроків було забезпечено високу якість веб-сайту. Кожна функціональність та алгоритм були протестовані вручну, виявлені помилки були виправлені, а система пройшла через декілька етапів повторного та регресійного тестування.

ВИСНОВОК

Розробка інтернет-магазину комп'ютерних складових була здійснена на основі веб-фреймворку Flask, який надає зручні інструменти для створення веб-додатків. Для забезпечення динамічного вмісту та роботи з базою даних використовувалися технології HTML, CSS, JavaScript, SQLAlchemy та Jinja2.

У результаті розроблено повноцінний інтернет-магазин комп'ютерних складових з такими функціональними можливостями:

- реєстрація та авторизація користувачів;
- відображення каталогу товарів із бази даних;
- генерація унікальних персональних знижок користувачам за допомогою віртуального сундука;
- можливість додавання товарів до кошика, перегляду вмісту кошика;
- функція скидання всіх знижок та підрахунку їх загальної суми;
- відображення таймера, який показує час до наступного оновлення.

Розроблений інтернет-магазин надає користувачам зручний та інтуїтивно зрозумілий інтерфейс для купівлі товарів. Впровадження інноваційної системи персональних знижок дозволяє стимулювати користувачів до покупок, підвищує їхню лояльність та сприяє збільшенню обороту магазину.

Отже, розроблений інтернет-магазин є ефективним інструментом для продажу комп'ютерних складових, який забезпечує зручний та привабливий інтерфейс для користувачів та стимулює їх до покупок за допомогою персональних знижок.

ПЕРЕЛІК ПОСИЛАНЬ

1 Node.js: Server-Side JavaScript Development Basics / freeCodeCamp. [Електронний ресурс]. – Режим доступу: https://www.freecodecamp.org/news/what-is-node-js/. – Дата звернення: 14.06.2024.

2 10+ Top Node.js Frameworks for Building Scalable Web Applications / Flatlogic Blog. [Електронний ресурс]. – Режим доступу: https://flatlogic.com/blog/10-top-node-js-frameworks-for-building-scalable-web-applications. – Дата звернення: 14.06.2024.

3 Getting Started with React / React Official Documentation. [Електронний ресурс]. – Режим доступу: https://reactjs.org/docs/getting-started.html. – Дата звернення: 14.06.2024.

4 Start with Django / Django Official Site. [Електронний ресурс]. – Режим доступу: https://www.djangoproject.com/start/. – Дата звернення: 14.06.2024.

5 Flask Documentation / Flask Official Site. [Електронний ресурс]. – Режим доступу: https://flask.palletsprojects.com/en/2.3.x/. – Дата звернення: 14.06.2024.

6 Laravel Documentation / Laravel Official Site. [Електронний ресурс]. – Режим доступу: https://laravel.com/docs. – Дата звернення: 14.06.2024.

7 MySQL: Відкрита реляційна база даних, яка широко використовується завдяки своїй надійності та простоті. [Електронний ресурс]. – Режим доступу: https://dev.mysql.com/doc/refman/8.0/en/. – Дата звернення: 14.06.2024.

8 PostgreSQL: Потужна реляційна база даних з підтримкою складних запитів та розширень. [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/. – Дата звернення: 14.06.2024.

9 SQLite: Легка база даних, яка не потребує окремого серверного ПЗ. Ідеально підходить для невеликих проєктів і розробки. [Електронний

ресурс]. – Режим доступу: https://www.sqlite.org/docs.html. – Дата звернення: 14.06.2024.

10 MongoDB: Документо-орієнтована база даних, яка дозволяє зберігати дані у форматі JSON. MongoDB підходить для динамічних і швидко змінюваних даних. [Електронний ресурс]. – Режим доступу: https://www.mongodb.com/docs/. – Дата звернення: 14.06.2024.

11 Visual Studio Code: Для розробки веб-додатку було обрано Visual Studio Code, оскільки це середовище розробки забезпечує зручну та ефективну роботу. [Електронний ресурс]. – Режим доступу: https://code.visualstudio.com/docs. – Дата звернення: 14.06.2024.

ДОДАТОК А – КОД ПРОГРАМИ

Серверна частина (app.py)

```
from flask import Flask, redirect, render_template,
url_for, request, session, jsonify
from flask_sqlalchemy import SQLAlchemy
import random
import threading
from flask_login import LoginManager, login_user,
logout_user, login_required, current_user, UserMixin
import schedule
import time
import requests

app = Flask(__name__)
app.secret_key = 'your_secret_key'

app.config['SQLALCHEMY_DATABASE_URI'] =
'sqlite:///products.db'

db = SQLAlchemy(app)
login_manager = LoginManager(app)

class Users(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(20), nullable=False)
    email = db.Column(db.String(50), nullable=False,
unique=True)
    password = db.Column(db.String(50), nullable=False)
    id_product1 = db.Column(db.Integer, nullable=True)
    Discount1 = db.Column(db.Integer, nullable=True)
    id_product2 = db.Column(db.Integer, nullable=True)
    Discount2 = db.Column(db.Integer, nullable=True)
    id_product3 = db.Column(db.Integer, nullable=True)
    Discount3 = db.Column(db.Integer, nullable=True)

class Cart(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer,
db.ForeignKey('users.id'), nullable=False)
    product_id = db.Column(db.Integer,
db.ForeignKey('product.id'), nullable=False)

    user = db.relationship('Users',
backref=db.backref('cart', lazy=True))
    product = db.relationship('Product',
backref=db.backref('cart', lazy=True))

    def __repr__(self):
        return f'<Cart {self.id} - User {self.user_id} -
Product {self.product_id}>'
```

```
class Product(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100))
    price = db.Column(db.Float)
    image = db.Column(db.String(100))
    type = db.Column(db.Text)

API_KEY = '79acafb049e1e10810b25f68c7e0b320'  # Замініть на
ваш API ключ

@app.route('/nova_poshta_cities', methods=['GET'])
def nova_poshta_cities():
    url = "https://api.novaposhta.ua/v2.0/json/"
    payload = {
        "apiKey": API_KEY,
        "modelName": "Address",
        "calledMethod": "getCities",
        "methodProperties": {}
    }
    response = requests.post(url, json=payload)
    data = response.json()
    cities = [city['Description'] for city in data['data']]
    return jsonify(cities=cities)

@app.route('/nova_poshta_departments', methods=['GET'])
def nova_poshta_departments():
    city = request.args.get('city')
    url = "https://api.novaposhta.ua/v2.0/json/"
    payload = {
        "apiKey": API_KEY,
        "modelName": "Address",
        "calledMethod": "getWarehouses",
        "methodProperties": {
            "CityName": city
        }
    }
    response = requests.post(url, json=payload)
    data = response.json()
    departments = [department['Description'] for department
in data['data']]
    return jsonify(departments=departments)


@app.route('/add_to_cart/<int:product_id>',
methods=['POST'])
@login_required
def add_to_cart(product_id):
    product = Product.query.get_or_404(product_id)
    cart_item =
Cart.query.filter_by(user_id=current_user.id,
product_id=product.id).first()
```

```
    new_cart_item = Cart(user_id=current_user.id,
product_id=product.id)
    db.session.add(new_cart_item)

    db.session.commit()
    return jsonify(success=True, message="Товар додано до
кошика")

@app.route('/remove_from_cart/<int:cart_item_id>',
methods=['POST'])
@login_required
def remove_from_cart(cart_item_id):
    cart_item = Cart.query.get_or_404(cart_item_id)
    if cart_item.user_id != current_user.id:
        return jsonify(success=False,
message="Несанкціоновано"), 403

    db.session.delete(cart_item)
    db.session.commit()
    return jsonify(success=True, message="Товар видалено з
кошика")

@app.route('/view_cart')
@login_required
def view_cart():
    cart_items =
Cart.query.filter_by(user_id=current_user.id).all()
    cart_data = []

    for item in cart_items:
        product = Product.query.get(item.product_id)
        price = product.price
        if item.product_id == current_user.id_product1:
            price -= price * (current_user.Discount1 / 100)
        elif item.product_id == current_user.id_product2:
            price -= price * (current_user.Discount2 / 100)
        elif item.product_id == current_user.id_product3:
            price -= price * (current_user.Discount3 / 100)

        # Округлення ціни до двох знаків після коми
        price = round(price, 2)

        cart_data.append({
            'id': item.id,
            'product_id': product.id,
            'name': product.name,
            'price': price,
            'image': product.image
        })

    total_price = sum(item['price'] for item in cart_data)
    total_price = round(total_price, 2)  # Округлення
загальної ціни до двох знаків після коми
```

```
    return jsonify(cart_data=cart_data,
total_price=total_price)


def reset_data():
    with app.app_context():
        users = Users.query.all()
        for user in users:
            user.id_product1 = None
            user.id_product2 = None
            user.id_product3 = None
            user.Discount1 = None
            user.Discount2 = None
            user.Discount3 = None
        db.session.commit()

# Функція для запуску запланованих завдань у окремому
потоці
def run_scheduler():
    while True:
        schedule.run_pending()
        time.sleep(1)

# Планування виконання завдання щодня о 2:00 ночі
schedule.every().day.at("02:00").do(reset_data)

# Запуск планувальника у окремому потоці
scheduler_thread = threading.Thread(target=run_scheduler)
scheduler_thread.daemon = True
scheduler_thread.start()



@login_manager.user_loader
def load_user(user_id):
    return Users.query.get(int(user_id))

@app.route('/')
def index():
    products = Product.query.all()
    return render_template("cooling.html",
products=products)

@app.route('/motherboards.html')
def motherboards():
    products = Product.query.all()
    return render_template('motherboards.html',
products=products)

@app.route('/cooling.html')
def cooling ():
    products = Product.query.all()
```

```
    return render_template("cooling.html",
products=products)

@app.route('/ssd_hdd.html')
def ssd_hdd ():
    products = Product.query.all()
    return render_template("ssd_hdd.html",
products=products)

@app.route('/RAM.html')
def RAM ():
    products = Product.query.all()
    return render_template("RAM.html", products=products)

@app.route('/processors.html')
def processors ():
    products = Product.query.all()
    return render_template("processors.html",
products=products)

@app.route('/powersupply.html')
def powersupply ():
    products = Product.query.all()
    return render_template("powersupply.html",
products=products)

@app.route('/graphicscards.html')
def graphicscards ():
    products = Product.query.all()
    return render_template("graphicscards.html",
products=products)

@app.route('/order')
def order ():
    cart_items = Cart.query.all()
    products = Product.query.all()
    return render_template("order.html", products=products,
cart_items=cart_items)


@app.route('/mydiscount.html')
@login_required
def mydiscount():
    user = Users.query.get(current_user.id)
    products = Product.query.all()
    return render_template("mydiscount.html", user=user,
products=products)

@app.route('/update_user_info', methods=['POST'])
def update_user_info():
    if not current_user.is_authenticated:
        return jsonify(success=False, error="User not
authenticated"), 401
```

```
    products = Product.query.all()
    selected_product_ids = set()  # Множина для відстеження
вибраних ID продуктів
    data = {}

    for i in range(1, 4):
        while True:
            random_product = random.choice(products)
            if random_product.id not in
selected_product_ids:
                selected_product_ids.add(random_product.id)
                break

        discount = random.randint(5, 20)

        setattr(current_user, f"id_product{i}",
random_product.id)
        setattr(current_user, f"Discount{i}", discount)

        data[f"product_image{i}"] = random_product.image
        data[f"product_name{i}"] = random_product.name
        data[f"prise{i}"] = random_product.price
        data[f"discount{i}"] = discount

    db.session.commit()

    data["success"] = True
    return jsonify(data)


@app.route('/login', methods=['POST'])
def login():
    if request.method == 'POST':
        mail = request.form['mail']
        password = request.form['password']
        user = Users.query.filter_by(email=mail,
password=password).first()
        if user:
            session['user_id'] = user.id
            session['username'] = user.username
            login_user(user)
            return redirect('/')
        else:
            # Якщо користувача не знайдено або введені дані
невірні, виводимо повідомлення про помилку
            return "Invalid username or password"

@app.route('/registration', methods=['POST'])
def registration():
    name = request.form['name']
    mail = request.form['mail']
    password = request.form['password']
```

```
    # Перевірка унікальності електронної пошти
    existing_user =
Users.query.filter_by(email=mail).first()
    if existing_user:
        return "Ця електронна пошта вже використовується.
Будь ласка, введіть іншу адресу."

        # Створення нового користувача
    user = Users(username=name, email=mail,
password=password)

    try:
        db.session.add(user)
        db.session.commit()  # Збереження змін у базі даних
        return redirect(url_for('index'))  #
Перенаправлення на головну сторінку після успішної
реєстрації

    except Exception as e:
        return f"Виникла помилка: {str(e)}"  # Відображення
повідомлення про помилку, якщо реєстрація не вдалася




@app.route('/logout')
def logout():
    session.pop('user_id', None)
    session.pop('username', None)
    logout_user()
    return redirect(url_for('index'))

if __name__ == "__main__":
    app.run(debug=True)
```

Шаблон (base.html):

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width,
initial-scale=1" />
        <title>{% block title %}{% endblock %}</title>
        <link
            rel="stylesheet"
            href="{{ url_for('static',
filename='css/main.css') }}"
        />
    </head>
    <body>
```

```
        <header>
            <div class="header-container">
                <div class="header-left">
                    <img
                        src="static/icon/logo.png"
                        alt="Site Logo"
                        class="site-logo"
                    />
                </div>
                <div class="header-center">
                    {% if session['username'] %}
                    <span class="site-name"
                        >ВІДКРИЙ СУНДУК В ОСОБИСТОМУ
КАБІНЕТІ І ОТРИМАЙ
                        ПЕРСОНАЛЬНІ ЗНИЖКИ!</span
                    >
                    {% else %}
                    <span class="site-name"
                        >АВТОРИЗУЙСЯ НА САЙТІ І ОТРИМАЙ
ПЕРСОНАЛЬНІ ЗНИЖКИ</span
                    >
                    {% endif %}
                </div>
                <div class="header-right" class="header-
link">
                    {% if session['username'] %}
                    <a href="#" class="header-link"
id="cart-showCartModal">
                        <img
                            src="\static\icon\cart.png"
                            alt="Add to Cart"
                            class="add-icon"
                        />
                    </a>
                    <div class="cart-counter">
                        (<span id="cart-
cartCount">0</span>)
                    </div>

                    <a href="mydiscount.html"
class="header-link">
                        <img
                            src="\static\icon\profile.png"
                            alt="Add to Cart"
                            class="add-icon1"
                        />
                    </a>
                    <a href="mydiscount.html"
class="header-link"
                        >{{ session['username'] }}
                    </a>
```

```
                    <a href="{{ url_for('logout') }}"
class="header-link"
                        >Вийти</a
                    >
                    {% else %}
                    <a class="header-link"
id="registerButton">Реєстрація</a>
                    <a class="header-link"
id="showLoginModal">Вхід</a>
                    {% endif %}
                </div>
            </div>
        </header>
        <div class="main-container">
            <div class="button-container">
                <a href="motherboards.html"
                    ><button class="btn add-to-cart" data-
product-id="1">
                        Плати
                    </button></a
                >
                <a href="processors.html"
                    ><button class="btn add-to-cart" data-
product-id="2">
                        Процесори
                    </button></a
                >
                <a href="RAM.html"
                    ><button class="btn add-to-cart" data-
product-id="3">
                        ОЗУ
                    </button></a
                >
                <a href="powersupply.html"
                    ><button class="btn add-to-cart" data-
product-id="4">
                        Блок живлення
                    </button></a
                >
                <a href="ssd_hdd.html"
                    ><button class="btn add-to-cart" data-
product-id="5">
                        Накопичувачі
                    </button></a
                >
                <a href="graphicscards.html"
                    ><button class="btn add-to-cart" data-
product-id="6">
                        Відеокарти
                    </button></a
                >
                <a href="cooling.html"
```

```
                    ><button class="btn add-to-cart" data-
product-id="7">
                        Охолодження
                    </button></a
                >
            </div>
            {% block body %}{% endblock %}
            <div class="timer-container">
                <h2>До наступного оновлення персональних
знижок:</h2>
                <div id="countdown"
class="timer">Завантаження...</div>
            </div>
        </div>
        <footer>
            <div class="footer-links">
                <a href="motherboards.html">Плати</a>
                <a href="processors.html">Процесори</a>
                <a href="RAM.html">ОЗУ</a>
                <a href="powersupply.html">Блок
живлення</a>
                <a href="ssd_hdd.html">Накопичувачі</a>
                <a href="graphicscards.html">Відеокарти</a>
                <a href="cooling.html">Охолодження</a>
                <a href="registration.html">Реєстрація</a>
                <a href="login.html">Вхід</a>
                <a
href="https://www.instagram.com/s3rgenius/">Instagram</a>
            </div>
        </footer>

        <div class="modal" id="registerModal">
            <div class="modal-content">
                <span class="close">&times;</span>
                <div class="reglogin-container">
                    <h1>Реєстрація</h1>
                    <form
                        id="myForm"
                        method="post"
                        action="{{ url_for('registration')
}}"
                    >
                        <input
                            type="text"
                            name="name"
                            id="name"
                            class="form-control"
                            placeholder="Ім'я"
                            required
                        />
                        <input
                            type="text"
                            name="mail"
```

```
                                id="mail"
                                class="form-control"
                                placeholder="Email"
                                required
                            />
                            <input
                                type="text"
                                name="password"
                                id="password"
                                class="form-control"
                                placeholder="Пароль"
                                required
                            />
                            <br />
                            <input
                                type="submit"
                                class="btn-success"
                                value="Відправити"
                            />
                            <p class="error"></p>
                        </form>
                    </div>
                </div>
            </div>

            <div class="modal" id="loginModal" style="display:
none">
                <div class="modal-content">
                    <span class="close
closeLoginModal">&times;</span>
                    <div class="reglogin-container">
                        <h1>Авторизація</h1>
                        <form id="loginForm" method="post"
action="/login">
                            <input
                                type="email"
                                name="mail"
                                id="loginMail"
                                class="form-control"
                                placeholder="Email"
                                required
                            />
                            <input
                                type="password"
                                name="password"
                                id="loginPassword"
                                class="form-control"
                                placeholder="Пароль"
                                required
                            />
                            <br />
                            <input
                                type="submit"
```

```
                            class="btn-success"
                            value="Login"
                        />
                        <p class="loginError"></p>
                    </form>
                </div>
            </div>
        </div>

        <div class="cart-modal" id="cart-cartModal"
style="display: none">
            <div class="cart-modal-content">
                <span class="cart-close cart-
closeCartModal">&times;</span>
                <div class="cart-cart-container">
                    <h1>Ваш кошик</h1>
                    <div id="cart-cartItems">
                        {% for item in cart_data %}
                        <div class="cart-cart-item">
                            <img
                                class="cart-cart-image"
                                src="/static/img/{{
item.image }}"
                                alt="{{ item.name }}"
                            />
                            <div class="cart-item-name">
                                <p>{{ item.name }}</p>
                            </div>
                            <p class="cart-price">{{
item.price }}₴</p>
                            <a onclick="removeFromCart('{{
item.id }}')">
                                <img
                                    src="/static/icon/delet
e.png"
                                    alt="Delete"
                                    class="add-icon-cart"
                                />
                            </a>
                        </div>
                        {% endfor %}
                    </div>
                    <div class="cart-footer">
                        <h2>
                            Загальна сума:
                            <span id="cart-totalPrice">{{
total_price }}</span>₴
                        </h2>
                        <button
                            class="cart-btn cart-btn-
success"
                            onclick="window.location.href='
/order'"
```

```
                        >
                            Замовити
                        </button>
                    </div>
                </div>
            </div>
        </div>

        <script>
            function calculateTimeUntilNextUpdate() {
                const now = new Date();
                let nextUpdate = new Date();
                nextUpdate.setHours(2, 0, 0, 0); //
Встановити час на 2:00:00

                if (now.getHours() >= 2) {
                    nextUpdate.setDate(now.getDate() + 1);
// Перейти на наступний день, якщо вже після 2 години ночі
                }

                const diff = nextUpdate - now;
                const hours = Math.floor(diff / (1000 * 60
* 60));
                const minutes = Math.floor(
                    (diff % (1000 * 60 * 60)) / (1000 * 60)
                );
                const seconds = Math.floor((diff % (1000 *
60)) / 1000);

                return { hours, minutes, seconds };
            }

            function updateCountdown() {
                const { hours, minutes, seconds } =
                    calculateTimeUntilNextUpdate();
                document.getElementById(
                    "countdown"
                ).innerHTML = `${hours} год ${minutes} хв
${seconds} с`;
            }

            setInterval(updateCountdown, 1000);
            updateCountdown(); // Початковий виклик для
негайного відображення таймера
        </script>

        <script>
            const loginModal =
document.getElementById("loginModal");
            const showLogin =
document.getElementById("showLoginModal");
            const closeLogin =
document.querySelector(".closeLoginModal");
```

```
            showLogin.addEventListener("click", () => {
                loginModal.style.display = "block";
            });

            closeLogin.addEventListener("click", () => {
                loginModal.style.display = "none";
            });

            document
                .getElementById("loginForm")
                .addEventListener("submit", function
(event) {
                    var email =
document.getElementById("loginMail").value;
                    var password =
                        document.getElementById("loginPassw
ord").value;
                    var error =
document.querySelector(".loginError");

                    error.innerHTML = "";

                    if (email === "" || password === "") {
                        error.innerHTML = "Please fill out
all fields";
                        event.preventDefault();
                        return false;
                    }

                    var emailRegex =
/^[^\s@]+@[^\s@]+\.[^\s@]+$/;
                    if (!emailRegex.test(email)) {
                        error.innerHTML = "Please enter a
valid email address";
                        event.preventDefault();
                        return false;
                    }

                    return true;
                });
        </script>

        <script>
            document
                .getElementById("myForm")
                .addEventListener("submit", function
(event) {
                    var name =
document.getElementById("name").value;
                    var email =
document.getElementById("mail").value;
```

```
                    var password =
document.getElementById("password").value;
                    var error =
document.querySelector(".error");

                    error.innerHTML = "";

                    if (name === "" || email === "" ||
password === "") {
                        error.innerHTML = "Please fill out
all fields";
                        event.preventDefault();
                        return false;
                    }

                    var emailRegex =
/^[^\s@]+@[^\s@]+\.[^\s@]+$/;
                    if (!emailRegex.test(email)) {
                        error.innerHTML = "Please enter a
valid email address";
                        event.preventDefault();
                        return false;
                    }

                    return true;
                });

            const registerModal =
document.getElementById("registerModal");
            const registerButton =
document.getElementById("registerButton");
            const registercloseButton =
document.querySelector(".close");

            registerButton.addEventListener("click", () =>
{
                registerModal.style.display = "block";
            });

            registercloseButton.addEventListener("click",
() => {
                registerModal.style.display = "none";
            });
        </script>
        <script>
            document.addEventListener("DOMContentLoaded",
function () {
                updateCartCount();

                const cartModal =
document.getElementById("cart-cartModal");
                const showCartButton =
```

```
                    document.getElementById("cart-
showCartModal");
                const closeCartButton =
document.querySelector(
                    ".cart-closeCartModal"
                );

                showCartButton.addEventListener("click", ()
=> {
                    fetch("/view_cart")
                        .then((response) =>
response.json())
                        .then((data) => {
                            const cartItemsDiv =
                                document.getElementById("ca
rt-cartItems");
                            cartItemsDiv.innerHTML = "";
                            data.cart_data.forEach((item)
=> {
                                const cartItemDiv =
                                    document.createElement(
"div");
                                cartItemDiv.className =
"cart-cart-item";
                                cartItemDiv.innerHTML = `
                                    <img class="cart-cart-
image" src="/static/img/${item.image}" alt="${item.name}"
/>
                                    <div class="cart-item-
name">
                                        <p>${item.name}</p>
                                    </div>
                                    <p class="cart-
price">${item.price}₴</p>
                                    <a
onclick="removeFromCart(${item.id})">
                                        <img
src="/static/icon/delete.png" alt="Delete" class="add-icon-
cart" />
                                    </a>
                                `;
                                cartItemsDiv.appendChild(ca
rtItemDiv);
                            });
                            document.getElementById(
                                "cart-totalPrice"
                            ).textContent =
data.total_price;
                            document.getElementById(
                                "cart-cartModal"
                            ).style.display = "block";
                        })
```

```
                        .catch((error) =>
console.error("Error:", error));
                });

                closeCartButton.addEventListener("click",
() => {
                    document.getElementById("cart-
cartModal").style.display =
                        "none";
                });

                window.onclick = function (event) {
                    if (event.target == cartModal) {
                        cartModal.style.display = "none";
                    }
                };
            });

            function addToCart(productId) {
                fetch(`/add_to_cart/${productId}`, {
method: "POST" })
                    .then((response) => response.json())
                    .then((data) => {
                        if (data.success) {
                            updateCartCount();
                            updateCartContents(); //
Оновити вміст кошика
                            showNotification();
                        } else {
                            alert(data.message);
                        }
                    })
                    .catch((error) =>
console.error("Error:", error));
            }

            function updateCartContents() {
                fetch("/view_cart")
                    .then((response) => response.json())
                    .then((data) => {
                        const cartItemsDiv =
                            document.getElementById("cart-
cartItems");
                        cartItemsDiv.innerHTML = "";
                        data.cart_data.forEach((item) => {
                            const cartItemDiv =
document.createElement("div");
                            cartItemDiv.className = "cart-
cart-item";
                            cartItemDiv.innerHTML = `
                                <img class="cart-cart-
image" src="/static/img/${item.image}" alt="${item.name}"
/>
```

```
                                    <div class="cart-item-
name">
                                        <p>${item.name}</p>
                                    </div>
                                    <p class="cart-
price">${item.price}₴</p>
                                    <a
onclick="removeFromCart(${item.id})">
                                        <img
src="/static/icon/delete.png" alt="Delete" class="add-icon-
cart" />
                                    </a>
                                `;
                                cartItemsDiv.appendChild(cartIt
emDiv);
                            });
                            document.getElementById("cart-
totalPrice").textContent =
                                data.total_price;
                    })
                    .catch((error) =>
console.error("Помилка:", error));
            }

            function removeFromCart(cartItemId) {
                fetch(`/remove_from_cart/${cartItemId}`, {
method: "POST" })
                    .then((response) => response.json())
                    .then((data) => {
                        if (data.success) {
                            updateCartContents();
                            getCartData();
                            updateCartCount();
                        } else {
                            alert(data.message);
                        }
                    })
                    .catch((error) =>
console.error("Error:", error));
            }

            function updateCartCount() {
                fetch("/view_cart")
                    .then((response) => response.json())
                    .then((data) => {
                        document.getElementById("cart-
cartCount").textContent =
                            data.cart_data.length;
                    })
                    .catch((error) =>
console.error("Error:", error));
            }
```

```
            function showNotification() {
                const notification =
document.createElement("div");
                notification.className = "notification";
                notification.innerHTML = `
                <p>Товар додано до кошика</p>
             `;
                document.body.appendChild(notification);

                // Hide the notification after 3 seconds
                setTimeout(function () {
                    notification.style.display = "none";
                }, 2000);
            }
        </script>
    </body>
</html>
```

Приклад сторінки сайту:

```
{% extends 'base.html' %} {% block title %} Охолодження {%
endblock %} {% block
body %}

<div class="product-container">
    {% for product in products %} {% if product.type ==
"cooling" %}
    <div class="product-tile">
        <div class="product-tile-img">
            <img
                src="\static\img\{{ product.image }}"
                alt="{{ product.name }}"
            />
        </div>
        <div class="product-tile-h2">
            <h2>{{ product.name }}</h2>
        </div>
        {% if product.id == current_user.id_product1 %}
        <div class="product-tile-p2">
            <div class="price-container">
                <p><s>{{ product.price }}₴</s></p>
                <p class="bold-red">
                    {{ product.price * (100.0 -
current_user.Discount1) / 100}}₴
                </p>
            </div>
            <button
                class="add-to-cart-btn"
                onclick="addToCart('{{ product.id }}')"
            >
                <img
                    src="\static\icon\addcart.png"
                    alt="Add to Cart"
```

```
                    class="add-icon"
                />
            </button>
        </div>
        {% elif product.id == current_user.id_product2 %}
        <div class="product-tile-p2">
            <div class="price-container">
                <p><s>{{ product.price }}₴</s></p>
                <p class="bold-red">
                    {{ product.price * (100.0 -
current_user.Discount2) / 100}}₴
                </p>
            </div>
            <button
                class="add-to-cart-btn"
                onclick="addToCart('{{ product.id }}')"
            >
                <img
                    src="\static\icon\addcart.png"
                    alt="Add to Cart"
                    class="add-icon"
                />
            </button>
        </div>
        {% elif product.id == current_user.id_product3 %}
        <div class="product-tile-p2">
            <div class="price-container">
                <p><s>{{ product.price }}₴</s></p>
                <p class="bold-red">
                    {{ product.price * (100.0 -
current_user.Discount3) / 100}}₴
                </p>
            </div>
            <button
                class="add-to-cart-btn"
                onclick="addToCart('{{ product.id }}')"
            >
                <img
                    src="\static\icon\addcart.png"
                    alt="Add to Cart"
                    class="add-icon"
                />
            </button>
        </div>
        {% else %}
        <div class="product-tile-p1">
            <p>{{ product.price }}₴</p>
            <button
                class="add-to-cart-btn"
                onclick="addToCart('{{ product.id }}')"
            >
                <img
                    src="\static\icon\addcart.png"
```

```
                    alt="Add to Cart"
                    class="add-icon"
                />
            </button>
        </div>
        {% endif %}
    </div>
    {% endif %} {% endfor %}
</div>

{% endblock %}
```

Сторінка особистого кабінету з генерацією знижки:

```
{% extends 'base.html' %} {% block title %} My Discount {%
endblock %} {% block
body %} {% if current_user.is_authenticated %} {% if
current_user.id_product1 is
none %}

<div class="chest-container">
    <img class="chest" src="\static\img\chest.jpg" alt="GIF
description" />
    <button class="gold-button" id="showGiftButton"
onclick="showGiftModal()">
        Показати подарунки
    </button>
</div>
{% else %}
<div class="product-container">
    <div class="congrats-content">
        <p class="congrats-text">ВАШІ ПЕРСОНАЛЬНІ
ЗНИЖКИ!</p>
    </div>
    {% for product in products %} {% if product.id ==
current_user.id_product1
    %}
    <div class="product-tile">
        <div class="product-tile-img">
            <img
                src="\static\img\{{ product.image }}"
                alt="{{ product.name }}"
            />
        </div>
        <div class="product-tile-h2">
            <h2>{{ product.name }}</h2>
        </div>
        <div class="product-tile-p2">
            <div class="price-container">
                <p><s>{{ product.price }}₴</s></p>
                <p class="bold-red">
                    {{ product.price * (100.0 -
current_user.Discount1) / 100}}₴
```

```
                    </p>
                </div>
                <button
                    class="add-to-cart-btn"
                    onclick="addToCart('{{ product.id }}')"
                >
                    <img
                        src="\static\icon\addcart.png"
                        alt="Add to Cart"
                        class="add-icon"
                    />
                </button>
            </div>
        </div>
        {% elif product.id == current_user.id_product2 %}
        <div class="product-tile">
            <div class="product-tile-img">
                <img
                    src="\static\img\{{ product.image }}"
                    alt="{{ product.name }}"
                />
            </div>
            <div class="product-tile-h2">
                <h2>{{ product.name }}</h2>
            </div>
            <div class="product-tile-p2">
                <div class="price-container">
                    <p><s>{{ product.price }}₴</s></p>
                    <p class="bold-red">
                        {{ product.price * (100.0 -
current_user.Discount2) / 100}}₴
                    </p>
                </div>
                <button
                    class="add-to-cart-btn"
                    onclick="addToCart('{{ product.id }}')"
                >
                    <img
                        src="\static\icon\addcart.png"
                        alt="Add to Cart"
                        class="add-icon"
                    />
                </button>
            </div>
        </div>
        {% elif product.id == current_user.id_product3 %}
        <div class="product-tile">
            <div class="product-tile-img">
                <img
                    src="\static\img\{{ product.image }}"
                    alt="{{ product.name }}"
                />
            </div>
```

```
        <div class="product-tile-h2">
            <h2>{{ product.name }}</h2>
        </div>
        <div class="product-tile-p2">
            <div class="price-container">
                <p><s>{{ product.price }}₴</s></p>
                <p class="bold-red">
                    {{ product.price * (100.0 -
current_user.Discount3) / 100}}₴
                </p>
            </div>
            <button
                class="add-to-cart-btn"
                onclick="addToCart('{{ product.id }}')"
            >
                <img
                    src="\static\icon\addcart.png"
                    alt="Add to Cart"
                    class="add-icon"
                />
            </button>
        </div>
    </div>
    {% endif %} {% endfor %}
</div>
{% endif %} {% else %}
<p>Please log in to see your discounts.</p>
{% endif %}

<div id="giftModal" class="modal" style="display: none">
    <div class="modal-content-discount">
        <span class="close"
onclick="closeGiftModal()">&times;</span>
        <p class="congrats-text">ВІТАЮ! ОСЬ ТАКІ ЗНИЖКИ ВИ
ОТРИМАЛИ!</p>
        <div class="product-container-discount">
            <div id="giftDetails1">
                <div class="product-tile">
                    <div class="product-tile-img">
                        <img src="" alt="Gift"
id="giftImage1" />
                    </div>
                    <div class="product-tile-h2">
                        <h2 id="giftName1"></h2>
                    </div>
                    <div>
                        <p id="orgPrice1"></p>
                        <p id="discount1"></p>
                        <div class="bold-red">
                            <p id="discountPrice1"></p>
                        </div>
                    </div>
                </div>
```

```
            </div>
            <div id="giftDetails2">
                <div class="product-tile">
                    <div class="product-tile-img">
                        <img src="" alt="Gift"
id="giftImage2" />
                    </div>
                    <div class="product-tile-h2">
                        <h2 id="giftName2"></h2>
                    </div>
                    <div>
                        <p id="orgPrice2"></p>
                        <p id="discount2"></p>
                        <div class="bold-red">
                            <p id="discountPrice2"></p>
                        </div>
                    </div>
                </div>
            </div>
            <div id="giftDetails3">
                <div class="product-tile">
                    <div class="product-tile-img">
                        <img src="" alt="Gift"
id="giftImage3" />
                    </div>
                    <div class="product-tile-h2">
                        <h2 id="giftName3"></h2>
                    </div>
                    <div>
                        <p id="orgPrice3"></p>
                        <p id="discount3"></p>
                        <div class="bold-red">
                            <p id="discountPrice3"></p>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>

<div id="loadingModal" class="modal">
    <div class="modal-content-gif">
        <img
            src="\static\img\chest.gif"
            alt="Loading GIF"
            style="width: 100%"
        />
    </div>
</div>

<script>
    function showGiftModal() {
```

```
        const loadingModal =
document.getElementById("loadingModal");
        loadingModal.style.display = "block";

        setTimeout(function () {
            loadingModal.style.display = "none";
            const giftModal =
document.getElementById("giftModal");
            giftModal.style.display = "block";
            updateUser();
        }, 6000); // Зачекати 6 секунд перед показом
подарунків
    }

    function closeGiftModal() {
        const modal = document.getElementById("giftModal");
        modal.style.display = "none";
        location.reload();
    }

    function updateUser() {
        fetch("/update_user_info", {
            method: "POST",
            headers: {
                "Content-Type": "application/json",
            },
            body: JSON.stringify({}),
        })
            .then((response) => response.json())
            .then((data) => {
                if (data.success) {
                    console.log("User info updated for
gifts");
                    for (let i = 1; i <= 3; i++) {
                        document.getElementById(
                            `giftImage${i}`
                        ).src =
`/static/img/${data[`product_image${i}`]}`;
                        document.getElementById(`giftName${
i}`).innerText =
                            data[`product_name${i}`];
                        document.getElementById(`orgPrice${
i}`).innerText = `${
                            data[`prise${i}`]
                        }₴`;
                        document.getElementById(
                            `discountPrice${i}`
                        ).innerText = `${(
                            (data[`prise${i}`] * (100 -
data[`discount${i}`])) /
                            100
                        ).toFixed(2)}₴`;
                        document.getElementById(
```

```
                            `discount${i}`
                        ).innerText = `Знижка
${data[`discount${i}`]}%!`;
                    }
                } else {
                    console.error("Error updating user info
for gifts");
                }
            })
            .catch((error) => {
                console.error("Error:", error);
            });
    }
</script>

{% endblock %}
```