МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ВЕБДОДАТКУ ДЛЯ ВІЗУАЛІЗАЦІЇ ДАНИХ СЕРВІСУ STEAM»**

Виконав: студент 5 курсу, групи 8.1223
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
Д. О. Жеребцов
(ініціали та прізвище)
Керівник доцент кафедри комп'ютерних наук, доцент, к.т.н Матвіїшина Н. В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент доцент кафедри програмної інженерії, доцент, к.т.н. Мухін В. В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти магістр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук,
д.т.н., професор

_____ Шило Г.М.
(підпис)

" 05 " березня 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ СТУДЕНТОВІ

Жеребцову Дмитру Олександровичу

1. Тема роботи Розробка вебдодатку для візуалізації даних сервісу Steam

керівник роботи Матвіїшина Надія Вікторівна, к.т.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » березня 2024 року № 385-с

2. Строк подання студентом роботи 25.11.2024

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної галузі
2. Створення вебдодатку
3. Реалізація проєкту

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 04.03.2024

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етпів кваліфікаційної роботи	Строк виконання етапів роботи	Примітки
1.	Розробка плану роботи	15.03.2024	
2.	Збір вихідних даних	27.03.2024	
3.	Обробка методичних та	10.05.2024	
	теоретичних джерел		
4.	Розробка першого та другого розділів	18.06.2024	
5.	Розробка третього розділу	23.08.2024	
6.	Оформлення та нормокнтроль	30.11.2024	
	кваліфікаційної роботи бакалавра		
7.	Захист кваліфікаційної роботи	12.12.2024	

Студент _____ Д.О. Жеребцов
(підпис) (ініціали та прізвище)

Керівник роботи _____ Н.В. Матвіїшина
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г.Спиця

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка вебдодатку для візуалізації даних сервісу Steam»: 42 с., 8 рис., 6 джерел.

АНАЛІЗ ДАНИХ, ВЕБДОДАТОК, STEAM, FLASK, PYTHON

Мета кваліфікаційної роботи: збір даних із сервісу Steam та їх візуалізація.

Об’єкт дослідження – процес розробки вебдодатку для візуалізації даних.

Предмет дослідження – методи та технології розробки вебдодатку для візуалізації даних Steam.

Метод дослідження – аналіз, описовий, порівняльний.

У кваліфікаційній роботі представлено вебдодаток який збирає, зберігає та візуалізує дані з магазину ігор Steam. Проведено детальний аналіз усіх етапів розробки вебдодатку, починаючи від формування вимог і закінчуючи тестуванням, з метою забезпечення повної функціональності. Використано технології:

Мова програмування Python, фреймворк для веб-розробки Flask, Для парсингу HTML-сторінок Steam та отримання необхідних даних BeautifulSoup.

SUMMARY

Master’s Qualification Thesis «Development of the web application for visualization of Steam service data»: 42 p., 8 figures, 6 references.

DATA ANALYSIS, WEB APPLICATION, STEAM, FLASK, PYTHON

The purpose of the qualification work: to collect data from the Steam service and visualize it.

The object of research is the process of developing a web application for data visualization.

Subject of research – methods and technologies for developing a web application for data visualization of Steam.

Research methods – analysis, descriptive, comparative.

The qualification work presents a web application that collects, stores and visualizes data from the Steam game store. A detailed analysis of all stages of the web application development, from the formation of requirements to testing, was carried out in order to ensure full functionality. Technologies used:

Python programming language, Flask web development framework, BeautifulSoup for parsing Steam HTML pages and retrieving the necessary data.

ЗМІСТ

Завдання на кваліфікаційну роботу…… 2
Реферат 4
Summary 5
Вступ 7
1 Аналіз предметної галузі та постановка задачі 8
1.1 Сервіс Steam: особливості, функціонал 8
1.2 Поняття парсингу 9
1.3 Поняття вебдодатку 10
1.4 Постановка задачі 12
2 Створення вебдодатку 15
2.1 Інструментарій для розробки 15
2.2 Аспекти розробки вебдодатку 16
3 Реалізація проєкту 18
3.1 Архітектура проєкту 18
3.2 Парсинг та збереження даних із вебресурсів 20
3.3 Обробка даних 23
3.3.1 Перетворення даних 24
3.3.2 Зберігання даних 24
3.3.3 Оновлення даних 25
3.3.4 Аналіз даних 26
3.4 Серверна логіка та маршрути 27
3.5 Візуалізація даних та розробка інтерфейсу 31
3.5.1 Головна сторінка 32
3.5.2 Список ігор 33
3.5.3 Популярні ігри 34
3.5.4 Дизайн і адаптивність 35
3.6 Оптимізація 35
3.7 Демонстрація роботи 37
Висновки 41
Перелік посилань 42

ВСТУП

Steam – це популярна платформа яка дає можливість користувачам придбати різні ігри на будь який смак. Важливим аспектом аналізу даних Steam є відстеження змін цін на ігри, що може бути корисним для гравців, розробників та дослідників ринку. Оскільки офіційний API Steam потребує авторизації, для отримання даних було використано методи вебскрейпінгу.

Метою роботи є збирання даних зі сторінок ігор Steam, зокрема інформації про ціни, та збереження цих даних, для подальшої візуалізації змін цін на графіках. Зібрані дані дають змогу аналізувати динаміку цін, виявляти знижки, а також будувати прогнози на основі історії цін.

Для виконання роботи використовувалася мова програмування Python[1]. Дані збиралися за допомогою бібліотек BeautifulSoup[3] і requests, які дозволяють парсити HTML-сторінки і витягувати потрібну інформацію без використання Steam API.

Вебдодаток, створений у рамках роботи, дозволяє користувачам відображати зібрані дані на інтерактивних графіках. Для цього використовувалися бібліотеки Matplotlib і Plotly, які забезпечують гнучкість у візуалізації та аналізі даних. Інтерфейс додатку розроблено з використанням Flask[4], що дає змогу запускати додаток у веббраузері і забезпечує зручний доступ до функціоналу для широкого кола користувачів.

Запропоноване рішення демонструє ефективність використання методів вебскрейпінгу у поєднанні з Python для автоматизації збору даних, їх аналізу та візуалізації. Це дозволяє відстежувати зміни цін на ігри, отримуючи цінну інформацію для ухвалення рішень, наприклад, щодо купівлі чи планування покупок.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Сервіс Steam: особливості, функціонал

Сервіс Steam пропонує доступ до тисяч ігор різних жанрів – від інді-розробок до масштабних блокбастерів. Крім ігор, у Steam можна знайти програмне забезпечення для творчості, навчання та підвищення продуктивності. Усі покупки здійснюються в цифровому форматі, що дозволяє завантажувати ігри будь-коли після придбання, не турбуючись про втрату фізичного носія. Steam славиться своїми регулярними сезонними розпродажами, під час яких користувачі можуть придбати ігри за вигідними цінами.

Однак, у Steam відсутня можливість переглядати історію змін цін на ігри, що часто ускладнює розуміння того, наскільки вигідна пропозиція або знижка. Ця проблема є особливо актуальною під час розпродажів, коли важливо знати, чи була гра дешевшою в минулому.

Саме тому цей проєкт вирішує цю проблему, надаючи можливість аналізувати цінові зміни для обраних ігор. Ця функція дозволяє користувачам бути обізнаними та приймати більш зважені рішення під час покупок.

Steam підтримує кросплатформеність, працюючи на Windows, macOS і Linux, а також має мобільний застосунок для Android та iOS. Завдяки хмарному збереженню дані про прогрес у грі автоматично зберігаються, що дозволяє продовжувати гру з іншого пристрою. У бібліотеці Steam зберігаються всі придбані ігри, зручний пошук і сортування спрощують доступ до них. Ігри автоматично оновлюються до останніх версій, що забезпечує користувачів новими функціями та виправленнями помилок.

Steam підтримує онлайн-гру з друзями та пропонує голосовий і текстовий чат. Завдяки Майстерні Steam гравці можуть створювати,

завантажувати та ділитися власними модифікаціями ігор. Платформа також пропонує досягнення та колекційні картки, які можна обмінювати або продавати. Через спільноту Steam користувачі спілкуються на форумах, пишуть рецензії, читають огляди та створюють групи за інтересами.

1.2 Поняття парсингу

Парсинг вебсторінок – це процес автоматизованого вилучення інформації з HTML-коду вебсторінок. Цей метод широко використовується для збору даних із веб-сайтів, що може включати текст, таблиці, зображення, посилання, мета-дані тощо.

Парсинг[3] є однією з ключових технік у вебскрейпінгу – загальній стратегії автоматизованого збору інформації з Інтернету. Він може бути корисним у різних сферах, таких як аналіз ринку, моніторинг цін, створення баз даних, дослідження тощо. Завдання парсингу можуть включати:

Вилучення тексту та таблиць: Отримання структурованої інформації, такої як ціни, рейтинги, назви продуктів або послуг.

Збір зображень: Завантаження та збереження графічного контенту.

Отримання посилань: Створення карти сайту чи виявлення зовнішніх і внутрішніх посилань.

Аналіз мета-даних: Збір інформації, яка описує сторінку, наприклад, теги meta або заголовки. Одним з найбільш популярних способів розробки ботів є використання мов програмування Python[2], особливо для розробки ботів для чат-платформи Discord.

Як працює парсинг?

Надсилання HTTP-запиту: Скрипт надсилає запит до веб-сервера та отримує HTML-код сторінки.

Обробка HTML: Отриманий код обробляється з метою вилучення необхідної інформації.

Форматування даних: Вилучена інформація може бути структурована у вигляді таблиці, JSON-файлу або бази даних.

Основні інструменти для парсингу на Python.

BeautifulSoup[3]: Бібліотека Python, яка дозволяє легко аналізувати HTML і XML-код.

Requests: Інструмент для надсилання HTTP-запитів і отримання даних із веб-сторінок.

Scrapy: Фреймворк для створення складних і масштабованих парсерів.

Selenium: Використовується для парсингу динамічних сторінок, які генеруються JavaScript.

Lxml: Потужна бібліотека для парсингу та обробки XML/HTML.

У проєкті використовуються BeautifulSoup та Requests.

1.3 Поняття вебдодатку

Вебдодаток це програмне забезпечення, яке працює на сервері та доступне через веб-браузер. Веб-додатки використовуються для виконання інтерактивних завдань і забезпечують користувачів функціональністю, яка часто схожа на звичайні програми, встановлені на комп’ютері або мобільному пристрої.

Основні характеристики веб-додатків:

Доступ через браузер: Користувачам не потрібно встановлювати додаткове програмне забезпечення – вони можуть працювати з веб-додатком за допомогою будь-якого сучасного браузера (Google Chrome, Firefox, Safari тощо).

Платформна незалежність: Завдяки використанню веб-технологій (HTML, CSS, JavaScript) веб-додатки працюють на будь-яких пристроях із доступом до Інтернету.

Централізоване управління: Оновлення та технічна підтримка веб-додатків виконуються на сервері, тому користувачі завжди працюють із актуальною версією.

Інтерактивність: Забезпечується за рахунок динамічного контенту, можливості зберігати дані, а також інтеграції з різними сервісами.

Приклади веб-додатків:

Онлайн-магазини (eBay, Amazon).

Системи управління проєктами (Trello, Asana).

Соціальні мережі (Facebook, Twitter).

Поштові сервіси (Gmail, Outlook).

Онлайн-редактори документів (Google Docs, Office 365).

Компоненти веб-додатку:

Клієнтська сторона (frontend): Відповідає за відображення даних і взаємодію з користувачем. Зазвичай використовує HTML, CSS, JavaScript, а також фреймворки, як-от React, Angular або Vue.js.

Серверна сторона (backend): Обробляє запити користувачів, взаємодіє з базами даних та іншими сервісами. Для розробки серверної частини використовуються мови програмування, такі як Python (Django, Flask), Node.js, Ruby (Ruby on Rails), PHP тощо.

База даних: Використовується для зберігання даних додатку, наприклад, інформації про користувачів чи товари. Популярні системи управління базами даних: MySQL, PostgreSQL, MongoDB.

Типи веб-додатків:

Статичні веб-додатки: Містять фіксований контент, який не змінюється залежно від дій користувача (наприклад, веб-сайти-візитки).

Динамічні веб-додатки: Мають інтерактивний контент, який може змінюватися в залежності від запитів користувача (наприклад, пошукові системи, інтернет-магазини).

Односторінкові додатки (SPA): Завантажуються як єдина сторінка, а контент динамічно оновлюється без перезавантаження.

Прогресивні веб-додатки (PWA): Поєднують функції веб-додатків і звичайних програм, дозволяючи працювати навіть офлайн.

Переваги веб-додатків:

Простота доступу: Не потрібно встановлювати; доступ здійснюється через браузер.

Масштабованість: Можна легко адаптувати для великої кількості користувачів.

Кросплатформеність: Працює на будь-яких пристроях із веб-браузером.

Централізоване оновлення: Усі зміни впроваджуються на сервері, користувачі отримують актуальну версію автоматично.

1.4 Постановка задачі

Основною метою проєкту є створення системи, яка дозволить збирати, аналізувати та візуалізувати дані про ціни на ігри в Steam, надаючи користувачам доступ до історії змін цін. Система повинна допомогти гравцям приймати обґрунтовані рішення щодо купівлі ігор, особливо під час сезонних розпродажів.

Основні пункти реалізації (рис. 1.1).

Збір інформації про ціни на ігри

Використання парсингу веб-сторінок Steam для отримання актуальних цін та знижок.

Застосування бібліотек для роботи з HTML, таких як BeautifulSoup (для структурного аналізу сторінки) і Requests (для завантаження контенту).

Створення бази даних

Замість традиційних баз даних, використано CSV-файли для збереження інформації про ціни та знижки.

Кожен рядок файлу представляє одну гру, з полями для її назви, поточної ціни, знижки та дати останнього оновлення.

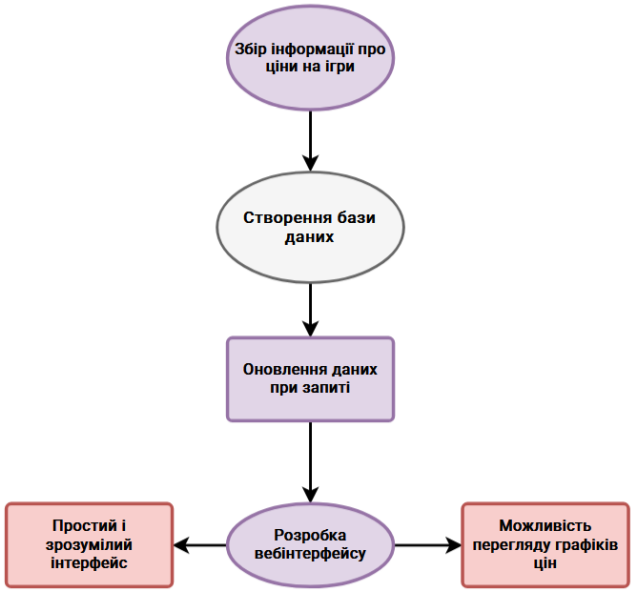


Рисунок 1.1 – Пункти реалізації

CSV-файли зручні для роботи в невеликих проєктах через їх простоту та легкість обробки.

Оновлення даних при запиті

Реалізація логіки, яка оновлює інформацію про ціну гри лише тоді, коли користувач здійснює запит.

Використання механізмів перевірки актуальності даних для уникнення дублювання запитів до серверів Steam.

Розробка вебінтерфейсу

Простий і зрозумілий інтерфейс:

– сикористання фреймворку Flask для створення веб-інтерфейсу;
– створення інтуїтивно зрозумілого дизайну з використанням CSS фреймворків, таких як Bootstrap, для забезпечення адаптивності;
– наявність зрозумілих меню для вибору ігор і навігації між сторінками;

Можливість перегляду графіків цін:

- побудова графіків змін цін за допомогою бібліотек візуалізації, таких як Matplotlib або Plotly;
- додавання функції порівняння цін на різні ігри для наочності.

Таким чином, цей проєкт спрямований на створення інструменту, який не тільки вирішить існуючу проблему з доступом до історії цін на ігри, але й надасть користувачам зручний і функціональний сервіс для планування та оптимізації витрат на покупки в Steam.

2 СТВОРЕННЯ ВЕБДОДАТКУ

2.1 Інструментарій для розробки

Першим кроком є встановлення Python і створення середовища для розробки. Python можна завантажити з офіційного сайту Python.org.

Після цього необхідно встановити основні бібліотеки за допомогою менеджера пакетів pip. Потрібно виконати таку команду у консолі:

pip install flask requests beautifulsoup4 pandas matplotlib flask-bootstrap

Використані бібліотеки:

Flask – це легкий і мінімалістичний веб-фреймворк на Python. Він дозволяє швидко створювати вебдодатки, зберігаючи простоту в реалізації.

Призначення: це мікрофреймворк для створення вебдодатків на Python.

Роль у проєкті:

Налаштування маршрутизації (обробка URL);

Рендеринг HTML-шаблонів;

Requests – це потужна бібліотека для виконання HTTP-запитів із простим API.

Призначення: дозволяє виконувати HTTP-запити).

Роль у проєкті:

Збір даних зі сторінок Steam.

BeautifulSoup (bs4) – це інструмент для парсингу та роботи зі структурованими документами HTML або XML.

Призначення: бібліотека для парсингу HTML і XML.

Роль у проєкті:

Виділення інформації зі сторінок Steam, наприклад, назви ігор, ціни, знижок.

Pandas – це бібліотека для аналізу та обробки табличних даних. Забезпечує зручний інтерфейс для роботи з датасетами.

Призначення: робота з табличними даними.

Роль у проєкті:

Збереження даних про ціни;

Аналіз і сортування даних;

Matplotlib – це популярна бібліотека для створення 2D-графіків та візуалізації даних.

Призначення: створення графіків.

Роль у проєкті:

Візуалізація динаміки цін.

Flask-Bootstrap – це додатковий модуль для Flask, який інтегрує Bootstrap (CSS-фреймворк) у ваш додаток.

Призначення: бібліотека для інтеграції Bootstrap з Flask.

Роль у проєкті:

Забезпечення адаптивного дизайну;

Використання готових компонентів (таблиці, кнопки, форми);

Цей код підключить вашого бота до API Discord та дозволить йому слідкувати за різними подіями, такими як підключення до сервера або приймання повідомлень в чаті. Команда command_prefix встановлює префікс для команд бота, які будуть виконуватися при виклику з цим префіксом.

Після написання коду можна перевірити, чи все працює, відлагодити його та перевірити, чи бот відповідає на всі запити користувачів.

2.2 Аспекти розробки вебдодатку

Одним із головних викликів у проєкті є парсинг динамічного контенту з веб-сторінок Steam, оскільки структура таких сторінок може змінюватися. Це вимагає регулярного тестування і оновлення скриптів для парсингу. Ще одним важливим аспектом є оптимізація запитів до серверів Steam:

використання кешування та обмеження частоти запитів допомагають уникнути надмірного навантаження на сервери.

Інтерфейс користувача, створений за допомогою Flask-Bootstrap, дозволяє легко навігувати між сторінками, вибирати ігри для аналізу та переглядати графіки. Візуалізація даних реалізована таким чином, щоб бути зрозумілою для користувачів із різними рівнями технічної підготовки. Використання графіків та таблиць робить додаток не лише інформативним, а й зручним у використанні.

Оптимізація є важливим аспектом розробки цього вебдодатку, оскільки він працює з великим обсягом даних і виконує регулярні запити до зовнішніх ресурсів. Для забезпечення швидкої роботи додатку використовується кілька технік. По-перше, кешування даних про ціни дозволяє мінімізувати кількість запитів до серверів Steam, зменшуючи затримки і навантаження на систему. Наприклад, дані про популярні ігри можуть оновлюватися один раз на день, а не запитуватися щоразу, коли користувач відкриває сторінку. По-друге, робота з даними оптимізується завдяки використанню Pandas, яка дозволяє ефективно обробляти великі таблиці даних. Ще однією важливою частиною оптимізації є використання асинхронних запитів під час парсингу даних, що допомагає прискорити завантаження інформації з кількох сторінок одночасно.

3 РЕАЛІЗАЦІЯ ПРОЄКТУ

Реалізація вебдодатку, який дозволяє відстежувати ціни на ігри Steam, є центральним етапом у створенні проєкту. На цьому етапі всі елементи, включаючи збір даних, обробку інформації та інтерактивний інтерфейс, об’єднуються в єдине функціональне рішення. Основою проєкту є Python[6] і його бібліотеки, які забезпечують інтеграцію з веб-ресурсами, обробку великих обсягів даних і створення графічного відображення.

Під час роботи над проєктом, використано мову програмування Python.

В процесі розробки розглянуто такі теми:

- архітектура проєкту;
- парсинг даних із веб-ресурсів;
- обробка даних;
- візуалізація даних та розробка інтерфейсу;
- перевірка роботи усіх команд та систем.

3.1 Архітектура проєкту

Архітектура вебдодатку побудована з використанням мікрофреймворку Flask, який забезпечує легку та гнучку розробку вебдодатків. Основною ідеєю архітектури є розділення логіки на окремі функціональні модулі, які відповідають за виконання конкретних завдань. Це забезпечує зручність у підтримці, розширенні та тестуванні проєкту.

Серверна частина реалізована у файлі app.py, який є основним елементом додатку. Flask дозволяє налаштувати маршрути, які пов’язують URL-адреси з відповідними функціями. Наприклад, маршрути /, /games_list і /popular_games відповідають за головну сторінку, список ігор та популярні ігри відповідно. Функції обробляють запити користувачів, виконують

необхідну логіку (наприклад, отримання даних про ціни на ігри) та повертають відповідні HTML-сторінки.

Обробка даних виконується за допомогою бібліотеки Pandas, яка використовується для читання та обробки даних із файлу price_data.csv. Дані про ціни оновлюються функціями update_price_data та load_price_data, що дозволяє зберігати історичну інформацію про зміни цін. Для обробки цієї інформації і побудови графіків використовується бібліотека Plotly, яка дозволяє створювати інтерактивні графіки і вбудовувати їх у веб-інтерфейс.

Збирання даних з сайту Steam виконується за допомогою бібліотеки BeautifulSoup. Вона аналізує HTML-структуру сторінок, витягаючи інформацію про ціни, знижки та назви ігор. Функції get_game_price та fetch_popular_games забезпечують збирання актуальної інформації, яка передається у шаблони для відображення.

Шаблони HTML знаходяться у папці templates і використовують механізм Jinja2 для динамічного відображення даних. Це дозволяє створювати гнучкий інтерфейс, який адаптується до змін даних. Наприклад, сторінка index.html показує графіки для вибраних ігор, а сторінки games_list.html та popular_games.html відображають списки ігор з цінами.

Файли конфігурації та зберігання даних також відіграють важливу роль. JSON-файл games_info.json зберігає базову інформацію про ігри, таку як їхні назви та App ID, що дозволяє легко додавати нові ігри для моніторингу. CSV-файл price_data.csv використовується для зберігання історії змін цін, забезпечуючи можливість аналізу динаміки.

Завдяки чітко продуманій архітектурі, додаток легко підтримувати та розширювати. Наприклад, додавання нового функціоналу, такого як фільтри за категоріями або сортування ігор за ціною, можна реалізувати шляхом додавання нових функцій і невеликих змін у шаблонах.

3.2 Парсинг та збереження даних із вебресурсів

Парсинг даних із вебресурсів є ключовою частиною функціональності вебдодатку. Цей процес дозволяє отримувати актуальну інформацію про ціни, знижки та назви ігор із платформи Steam. Для цього використовується бібліотека BeautifulSoup, яка забезпечує зручні інструменти для аналізу та обробки HTML-коду веб-сторінок.

У проєкті реалізовано кілька функцій для парсингу. Основна функція get_game_price відповідає за отримання інформації про ціну конкретної гри. Вона завантажує сторінку гри з офіційного сайту Steam за допомогою модуля requests, а потім аналізує HTML-код із використанням BeautifulSoup.

Код для функції отримання цін:

```
def get_game_price(app_id):
    url = f"https://store.steampowered.com/app/{app_id}"
    response = requests.get(url)
    if response.status_code != 200:
        return None, None  # Не вдалося завантажити сторінку

    soup = BeautifulSoup(response.text, 'html.parser')

    # Шукаємо поточну ціну гри
    price_div = soup.find("div", class_="game_purchase_price")
    discount_price_div = soup.find("div", class_="discount_final_price")
    discount_percentage_div = soup.find("div", class_="discount_pct")

    if price_div:
        price = price_div.text.strip()
        return price, None  # Ціна без знижки
    elif discount_price_div:
```

```
            discount_price = discount_price_div.text.strip()
            discount_percentage = discount_percentage_div.text.strip() if
discount_percentage_div else "0%"
            return discount_price, discount_percentage
        else:
            # Перевірка на безкоштовні ігри
            free_div = soup.find("div", class_="game_area_free")
            if free_div and "Free" in free_div.text:
                return "Бесплатно", None

        return None, None  # Якщо нічого не знайдено
```

Ця функція шукає елементи HTML, які містять інформацію про ціну (game_purchase_price), знижки (discount_final_price та discount_pct) або статус безкоштовності гри (game_area_free). У разі успіху функція повертає знайдені дані, а у випадку помилки або відсутності інформації – відповідні повідомлення.

Для отримання списку популярних ігор з актуальними цінами використовується функція fetch_popular_games. Вона виконує запит до сторінки Steam із найпопулярнішими іграми та аналізує її HTML-код для вилучення списку ігор.

Код для списку популярних ігор:

```
def fetch_popular_games():
    url = "https://store.steampowered.com/search/?filter=topsellers"
    response = requests.get(url)

    if response.status_code != 200:
        print("Не вдалося завантажити сторінку з популярними іграми.")
```

```
    return []

soup = BeautifulSoup(response.text, 'html.parser')
games = soup.select('.search_result_row')  # Усі ігри на сторінці
popular_games = []

for game in games[:10]:  # Обмежуємо список до 10 ігор
    name = game.select_one('.title').text.strip()  # Назва гри
    link = game['href']  # Посилання на гру в Steam

    # Переходимо на сторінку гри для отримання актуальної ціни
    game_response = requests.get(link)
    if game_response.status_code != 200:
        print(f"Не вдалося завантажити сторінку для гри {name}.")
        price = "Ціна недоступна"
    else:
        game_soup = BeautifulSoup(game_response.text, 'html.parser')

        # Шукаємо ціну на сторінці гри
        price_div = game_soup.select_one('.game_purchase_price')
        if not price_div:
            # Шукаємо ціну у випадку знижки
            price_div = game_soup.select_one('.discount_final_price')

        if price_div:
            price = price_div.text.strip()
        else:
            price = "Ціна недоступна"  # Якщо ціна не знайдена

    # Додаємо гру з ціною до списку
```

```
popular_games.append({
                'name': name,
                'price': price,
                'steam_link': link
            })

        return popular_games
```

Ця функція обробляє список популярних ігор, обмежуючи його до 10 позицій, та виконує додаткові запити для отримання детальної інформації про кожну гру.

Оскільки дані на веб-сторінках можуть змінюватися, функції перевіряють статус відповіді сервера та враховують можливі помилки у структурі HTML-коду. Отримані дані використовуються для оновлення локальної бази (price_data.csv) або для відображення у веб-інтерфейсі додатку.

Використання бібліотеки BeautifulSoup дозволяє адаптувати код до змін у структурі сторінок Steam, забезпечуючи правильну роботи додатку.

3.3 Обробка даних

Обробка даних у вебдодатку є ключовим етапом для забезпечення точності та коректності інформації, яка відображається користувачам. Основна задача цього процесу – підготовка, зберігання, та аналіз отриманих з веб-ресурсів даних, таких як ціни ігор, їх зміни з часом та знижки.

Дані, отримані через парсинг, часто містять зайву інформацію або представлені у непридатному для подальшого використання форматі. Для вирішення цих проблем реалізовано декілька функцій, які дозволяють фільтрувати, перетворювати та аналізувати інформацію.

3.3.1 Перетворення даних

Однією з таких функцій є convert_price, яка перетворює ціни з текстового у числовий формат. Це важливо для подальшого аналізу та порівняння цін.

```
def convert_price(price):
    price = re.sub(r'[^\d.,]', '', price)
    try:
        return float(price.replace(',', '.'))
    except ValueError:
        return None
```

Ця функція видаляє непотрібні символи (наприклад, валюти чи пробіли) з рядка та перетворює його у число з плаваючою крапкою. У разі помилки перетворення функція повертає None, щоб уникнути некоректних даних.

3.3.2 Зберігання даних

Для зберігання історичних даних про зміни цін використовується локальний файл price_data.csv. Функція load_price_data дозволяє завантажувати ці дані у зручний для обробки формат за допомогою бібліотеки pandas.

```
def load_price_data():
    try:
        data = pd.read_csv('price_data.csv')
        data['date'] = pd.to_datetime(data['date'])
        return data
    except FileNotFoundError:
        return pd.DataFrame(columns=['game', 'date', 'price'])
```

У разі відсутності файлу функція повертає порожню таблицю, щоб уникнути збоїв у роботі додатку.

3.3.3 Оновлення даних

```
def update_price_data(selected_game=None):
    games_info = load_games_info()
    if selected_game:
        games_info = [game for game in games_info if game['name'] ==
selected_game]

    data = load_price_data()

    for game in games_info:
        price, _ = get_game_price(game['app_id'])
        if not price:
            print(f"Не вдалося отримати ціну для {game['name']}")
            continue

        price = convert_price(price)
        if price is None:
            print(f"Помилка перетворення ціни для {game['name']}: {price}")
            continue

        today = datetime.datetime.now().strftime('%Y-%m-%d')

        last_record = data[data['game'] == game['name']].iloc[-1:]
```

Для регулярного оновлення інформації про ціни реалізовано функцію update_price_data. Вона перевіряє, чи змінилася ціна для певної гри, і за необхідності додає новий запис у файл price_data.

```
if last_record.empty or last_record['price'].values[0] != price:
    try:
        with open('price_data.csv', 'a', newline='') as csvfile:
            writer = csv.writer(csvfile)
            writer.writerow([game['name'], today, price])
        print(f"Дані про ціну для {game['name']} оновлені: ₴{price}")
    except Exception as e:
        print(f"Помилка запису в файл: {e}")
else:
    print(f"Ціна для {game['name']} не змінилася на {today}. Оновлення пропущено.")
```

Ця функція запобігає дублюванню записів та забезпечує актуальність історичних даних.

3.3.4 Аналіз даних

Для аналізу змін цін у часі використовується функція create_price_plot, яка створює інтерактивний графік за допомогою бібліотеки plotly.

```
def create_price_plot(games_data):
    import plotly.graph_objects as go

    fig = go.Figure()

    for game_data in games_data:
        game_name = game_data['game'].iloc[0] if not game_data.empty else 'Гра'
        fig.add_trace(go.Scatter(
            x=game_data['date'],
            y=game_data['price'],
```

```
            mode='lines+markers',
            name=game_name
        ))

    fig.update_layout(
        title="Графік зміни цін",
        xaxis_title="Дата",
        yaxis_title="Ціна (₴)",
        template="plotly_dark"
    )
    return fig.to_html(full_html=False)
```

Цей графік дозволяє користувачам візуально оцінити динаміку змін цін на обрані ігри, що є важливим для ухвалення рішень щодо їх купівлі.

Таким чином, обробка даних у проєкті забезпечує високу точність, актуальність та зручність використання для кінцевих користувачів.

3.4 Серверна логіка та маршрути

У проєкті для реалізації серверної логіки використовується фреймворк Flask, який забезпечує обробку HTTP-запитів та взаємодію між користувачем і системою. Серверна частина відповідає за отримання даних, їх обробку та передачу у вигляді HTML-сторінок, динамічного контенту та графіків.

Основні функції серверної логіки

Обробка запитів на різні сторінки

- сервер обробляє маршрути для трьох основних сторінок вебдодатка;
- головна сторінка (/);
- сторінка списку ігор (/games_list);
- сторінка популярних ігор (/popular_games).

Обробка введених даних користувача

На головній сторінці користувач може вибирати ігри для порівняння. Сервер отримує ці дані через GET-запити, обробляє їх та генерує відповідні графіки і таблиці.

Побудова графіків

На основі вибраних ігор сервер формує графіки змін цін, використовуючи бібліотеку Plotly. Готові графіки передаються на фронтенд у вигляді HTML-коду.

Взаємодія з базою даних

Сервер отримує актуальні дані про ігри, ціни, знижки та популярність із бази даних, обробляє їх і передає у шаблони для відображення на сторінках.

Реалізація маршруту головної сторінки:

```
@app.route('/')
def index():
    """
    Головна сторінка, де користувачі можуть обрати ігри для відображення графіка.
    """
    selected_game_1 = request.args.get('game_1')
    selected_game_2 = request.args.get('game_2')

    if selected_game_1:
        update_price_data(selected_game_1)
    if selected_game_2:
        update_price_data(selected_game_2)

    data = load_price_data()
    games_info = load_games_info()
    games = [game['name'] for game in games_info]
    plot_html = None
```

```
        price_list_1 = []
        price_list_2 = []

        # Дані для першої обраної гри
        if selected_game_1:
            game_data_1 = data[data['game'] == selected_game_1]
            if not game_data_1.empty:
                price_list_1 = game_data_1[['date',
'price']].sort_values('date').to_dict(orient='records')

        # Дані для другої обраної гри (якщо обрана)
        if selected_game_2:
            game_data_2 = data[data['game'] == selected_game_2]
            if not game_data_2.empty:
                price_list_2 = game_data_2[['date',
'price']].sort_values('date').to_dict(orient='records')

        # Побудова графіка
        if selected_game_1 and selected_game_2:
            games_data = [game_data_1, game_data_2]
            plot_html = create_price_plot(games_data)
        elif selected_game_1:
            games_data = [game_data_1]
            plot_html = create_price_plot(games_data)

        return render_template(
            'index.html',
            games=games,
            plot_html=plot_html,
            price_list_1=price_list_1,
```

```
price_list_2=price_list_2,
                selected_game_1=selected_game_1,
                selected_game_2=selected_game_2,
                is_first_selected=bool(selected_game_1)    # Перевірка, чи обрана
перша гра
            )
```

Реалізація маршруту списку ігор:

```
@app.route('/games_list')
def games_list():
    """
    Сторінка з повним списком ігор та їх цінами.
    """
    games_info = load_games_info()
    game_details = []

    for game in games_info:
        price, discount = get_game_price(game['app_id'])
        game_details.append({
            'name': game['name'],
            'price': price if price else 'Ціна недоступна',
            'discount': discount if discount else 'Без знижки',
            'steam_link':
f"https://store.steampowered.com/app/{game['app_id']}"
        })

    return render_template('games_list.html', game_details=game_details)  #
Сторінка з детальною інформацією про ігри
```

Реалізація маршруту популярних ігор:

```
@app.route('/popular_games')
def popular_games():
    current_date = datetime.datetime.now()
    current_month_year = current_date.strftime("%B %Y")

    # Отримуємо популярні ігри з актуальними цінами
    popular_games = fetch_popular_games()

    return render_template(
        'popular_games.html',
        popular_games=popular_games,
        current_month_year=current_month_year
    )
```

3.5 Візуалізація даних та розробка інтерфейсу

Розробка інтерфейсу та візуалізація даних є важливими аспектами проєкту. Завдяки використанню веб-технологій, таких як HTML, CSS та шаблони Jinja2, було створено інтерактивний і зручний у користуванні вебдодаток.

Інтерфейс складається з трьох основних сторінок:

Головна сторінка (index.html) – для порівняння цін на ігри та відображення графіків.

Список ігор (games_list.html) – для перегляду актуальних цін та знижок.

Популярні ігри (popular_games.html) – для перегляду найпопулярніших ігор поточного місяця.

3.5.1 Головна сторінка

Головна сторінка містить форму для вибору однієї або двох ігор для порівняння. Після вибору ігор відображається інтерактивний графік змін цін, згенерований за допомогою бібліотеки Plotly, а також таблиці з історичними даними про ціни.

Ключовий функціонал:

– динамічний вибір ігор із використанням даних, отриманих через сервер;
– побудова графіка зміни цін;
– таблиці, що відображають історичні дані.

Приклад коду для форми вибору ігор:

```
<form method="get" action="/" class="form-inline mb-4">
  <div class="form-group mr-2">
    <label for="game_1">Оберіть гру 1:</label>
    <select name="game_1" id="game_1" class="form-control">
      <option value=""> Оберіть гру </option>
      {% for game in games %}
        <option value="{{ game }}" {% if game == selected_game_1
%}selected{% endif %}>{{ game }}</option>
      {% endfor %}
    </select>
  </div>
</form>
```

3.5.2 Список ігор

На сторінці games_list.html користувачі можуть переглядати актуальні ціни на всі доступні ігри, включно зі знижками, якщо вони є. У таблиці також є кнопка для переходу до сторінки гри в Steam.

Ключовий функціонал:

– відображення назв ігор, цін, знижок і посилань на Steam;

– сортування ігор у зручному форматі.

Фрагмент таблиці з даними:

```
<table class="table table-dark mt-4">
    <thead>
    <tr>
      <th>Назва</th>
      <th>Ціна</th>
      <th>Знижка</th>
      <th>Посилання на Steam</th>
    </tr>
    </thead>
    <tbody>
    {% for game in game_details %}
      <tr>
        <td>{{ game.name }}</td>
        <td>{{ game.price }}</td>
        <td>{{ game.discount }}</td>
        <td><a href="{{ game.steam_link }}" target="_blank" class="btn
btn-primary">Відкрити</a></td>
      </tr>
    {% endfor %}
    </tbody>
  </table>
```

3.5.3 Популярні ігри

Сторінка popular_games.html відображає найпопулярніші ігри поточного місяця. Інформація представлена у вигляді таблиці, яка включає рейтинг гри, її назву, ціну та посилання на Steam.

Ключовий функціонал:

– динамічне оновлення популярних ігор для поточного місяця;
– відображення інформації у вигляді зрозумілої таблиці.

```
<table class="table table-dark table-hover">
  <thead>
  <tr>
    <th>#</th>
    <th>Название</th>
    <th>Ціна</th>
    <th>Посилання на Steam</th>
  </tr>
  </thead>
  <tbody>
  {% for game in popular_games %}
    <tr>
      <td>{{ loop.index }}</td>
      <td>{{ game.name }}</td>
      <td>{{ game.price }}</td>
      <td><a href="{{ game.steam_link }}"
target="_blank">Відкрити</a></td>
    </tr>
  {% endfor %}
  </tbody>
</table>
```

3.5.4 Дизайн і адаптивність

Для стилізації використовується бібліотека Bootstrap, що забезпечує:

– інтуїтивно зрозумілий вигляд елементів;
– адаптивність для різних розмірів екранів.

Фоновий колір і стилі були налаштовані для створення зручного темного інтерфейсу, який підходить для роботи в умовах низького освітлення.

Таким чином, візуалізація даних та розробка інтерфейсу дозволяють ефективно відображати отриману інформацію, забезпечуючи користувачів усім необхідним функціоналом для аналізу цін на ігри.

3.6 Оптимізація

Оптимізація коду є важливим аспектом створення ефективних та продуктивних додатків. У проєкті реалізовано декілька підходів, які допомагають мінімізувати навантаження на систему, обмежити непотрібну обробку даних та зберегти ресурси. Розглянемо основні моменти оптимізації:

Запис даних у файл лише за необхідності

Функція update_price_data реалізована так, що ціни для гри записуються в CSV-файл лише тоді, коли вони змінюються. Це дозволяє уникнути дублювання даних та зберегти обсяг файлу.

Як це працює:

– дані про останню збережену ціну для гри завантажуються з файлу;
– якщо нова ціна не збігається із збереженою, записується новий рядок із оновленою ціною;
– у разі відсутності змін пропускається оновлення, що економить час і ресурси.

```
last_record = data[data['game'] == game['name']].iloc[-1:]
if last_record.empty or last_record['price'].values[0] != price:
```

Приклад:

```
with open('price_data.csv', 'a', newline='') as csvfile:
    writer = csv.writer(csvfile)
    writer.writerow([game['name'], today, price])
```

Цей підхід дозволяє уникнути постійного перезапису даних, знижуючи кількість операцій вводу-виводу на диску.

Обмеження обробки даних для вибраних ігор

При виборі конкретної гри користувачем для побудови графіка оновлюється лише інформація про ціни для цієї гри. Це дозволяє зменшити обсяг даних, які обробляються:

```
if selected_game_1:
    update_price_data(selected_game_1)
if selected_game_2:
    update_price_data(selected_game_2)
```

Якщо ігри не вибрані, дані не оновлюються, що знижує навантаження.

Попереднє завантаження статичних даних

Функція load_games_info завантажує інформацію про ігри з JSON-файлу. Оскільки цей файл є статичним і не змінюється під час роботи програми, його зчитування виконується лише за необхідності, що знижує зайві операції вводу-виводу.

Перевірка коректності даних

Реалізовано перевірку отриманих даних із Steam, щоб уникнути збереження некоректних цін:

– якщо сторінка гри недоступна або ціна не знайдена, обробка для цієї гри пропускається;

– значення ціни конвертується у числовий формат. У разі помилки конвертації обробка також пропускається.

```
price = convert_price(price)
if price is None:
print(f"Помилка конвертації ціни для {game['name']}: {price}")
continue
```

Обмеження кількості оброблюваних ігор

Для популярних ігор із Steam завантажується лише топ-10. Це дозволяє зменшити час обробки та запити до API Steam:

for game in games[:10]: # Обмеження списку до 10 ігор

Ефективне використання бібліотек

– Pandas: використовується для роботи з даними CSV, що дозволяє зручно фільтрувати, сортувати та обробляти дані;
– Plotly: динамічне створення графіків із мінімальними витратами на обчислення;
– BeautifulSoup: оптимізовано для парсингу сторінок Steam.

3.7 Демонстрація роботи

Демонстрація роботи вебдодатку представлена на рисунках 3.1 – 3.7.

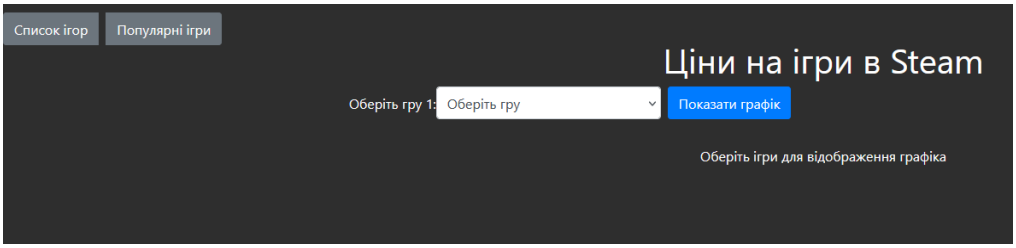


Рисунок 3.1 – Початкова сторінка

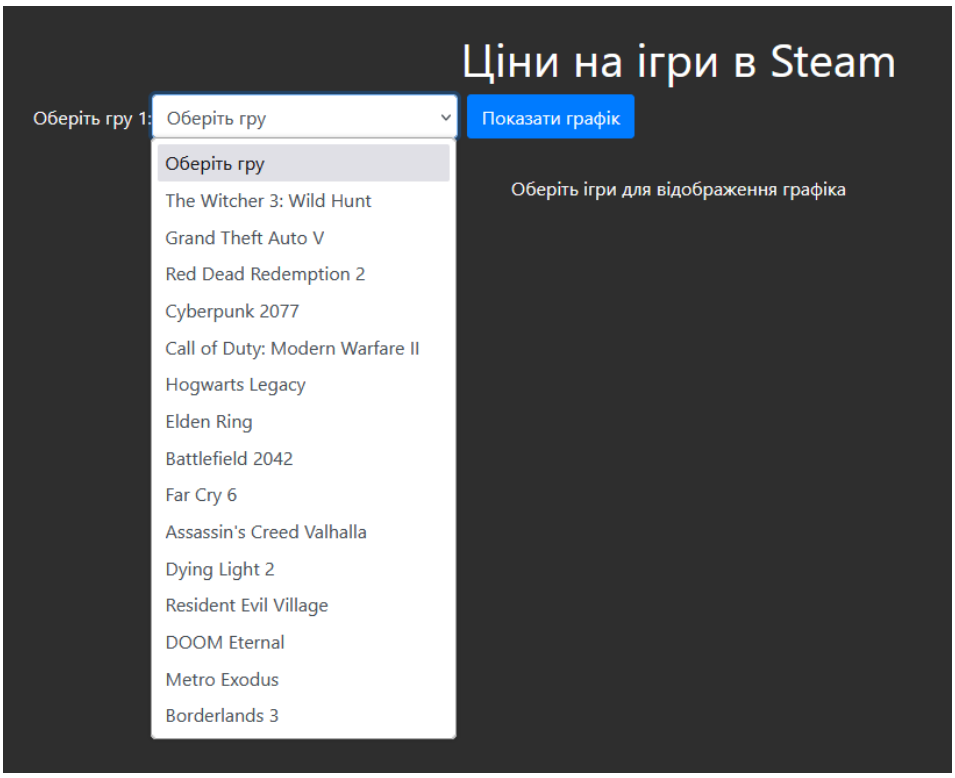


Рисунок 3.2 – Список ігор для вибору

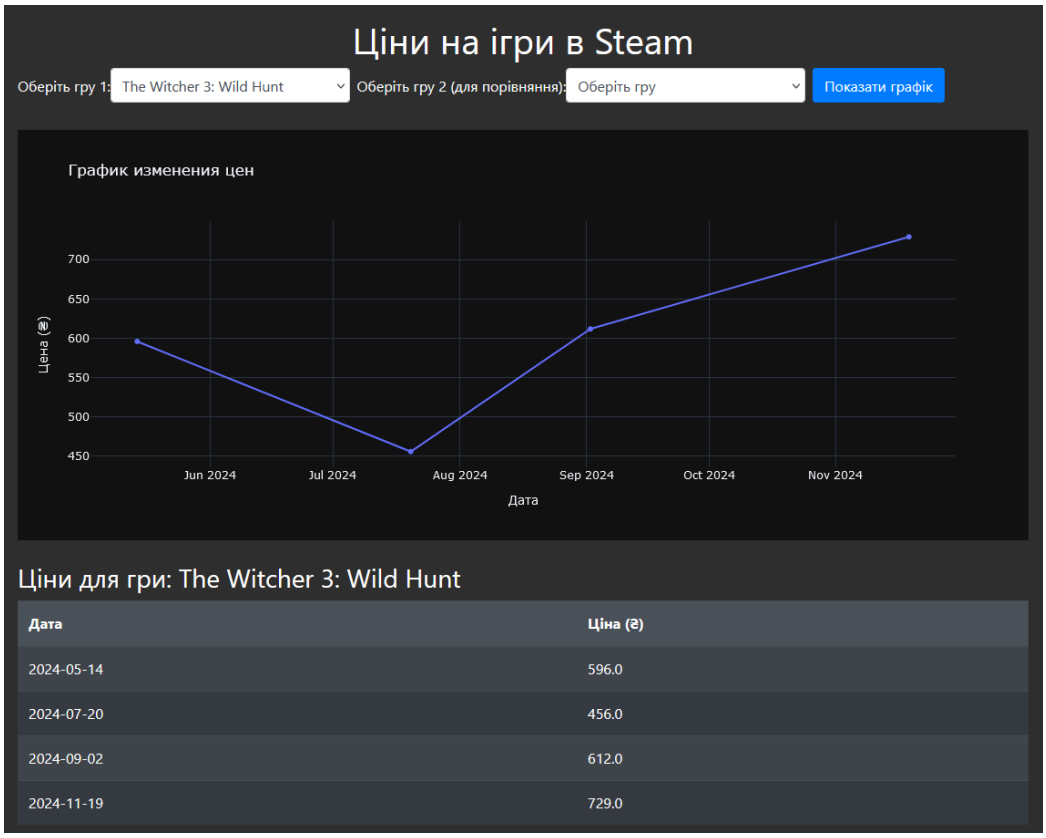


Рисунок 3.3 – Обрано гру

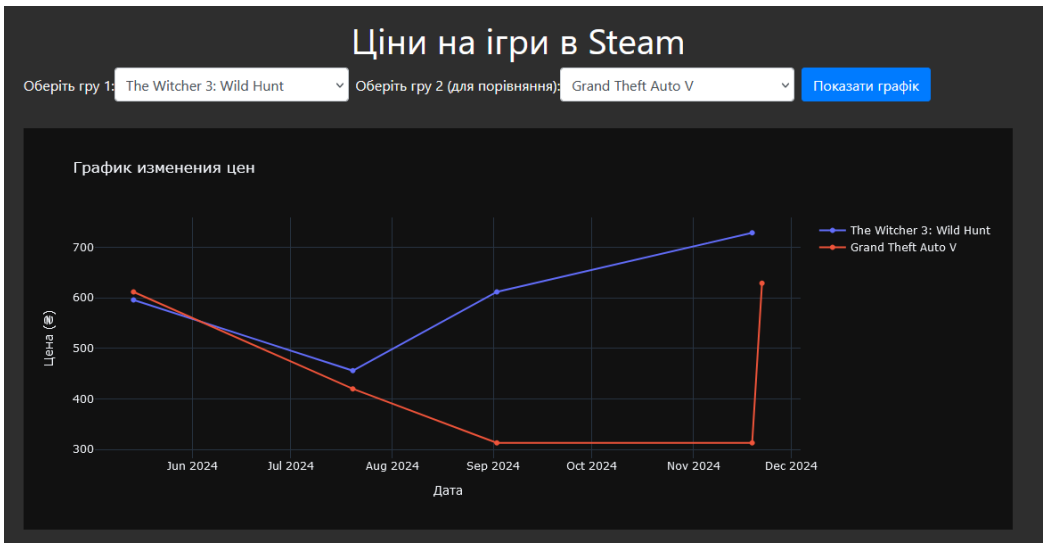


Рисунок 3.4 – Обрано дві гри



Ціни для гри: The Witcher 3: Wild Hunt

Дата	Ціна (₴)
2024-05-14	596.0
2024-07-20	456.0
2024-09-02	612.0
2024-11-19	729.0

Ціни для гри: Grand Theft Auto V

Дата	Ціна (₴)
2024-05-14	612.0
2024-07-20	420.0
2024-09-02	313.0
2024-11-19	313.0
2024-11-22	629.0

Рисунок 3.5 – Історія цін для обраних ігор

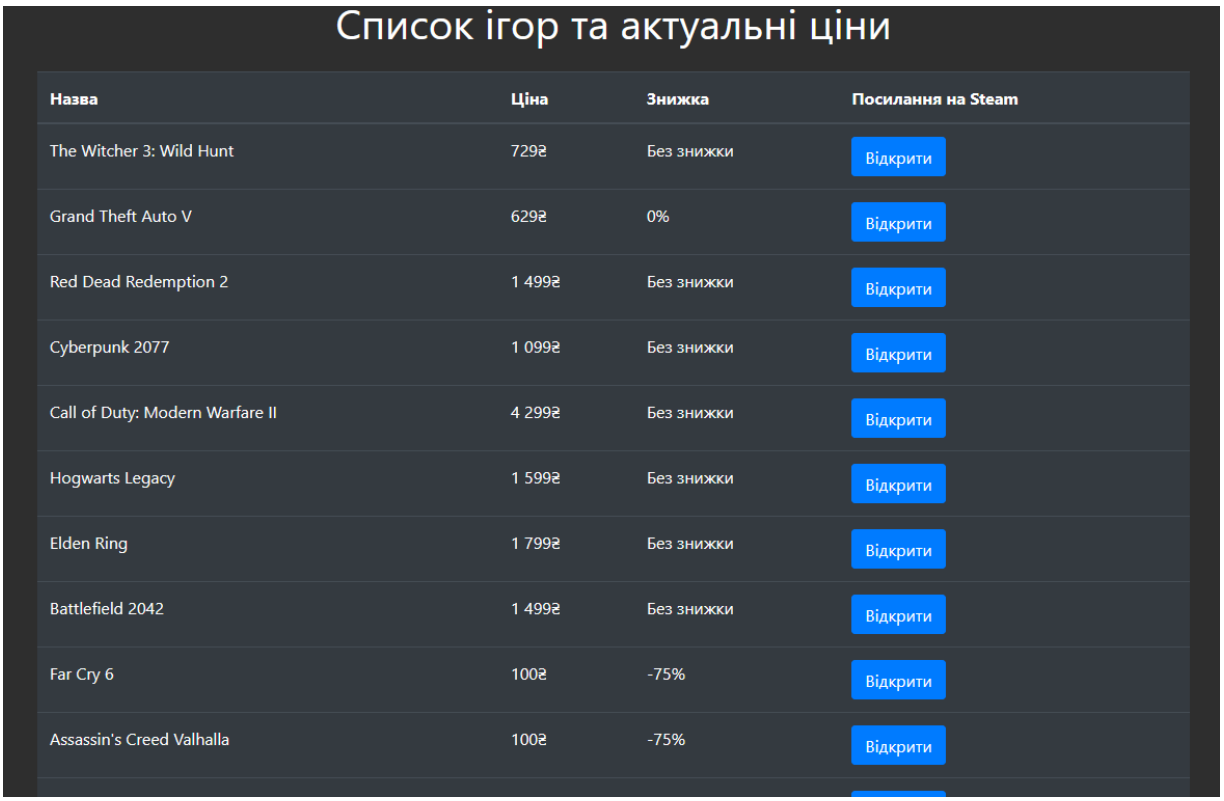


Рисунок 3.6 – Сторінка «Список ігор та актуальні ціни»

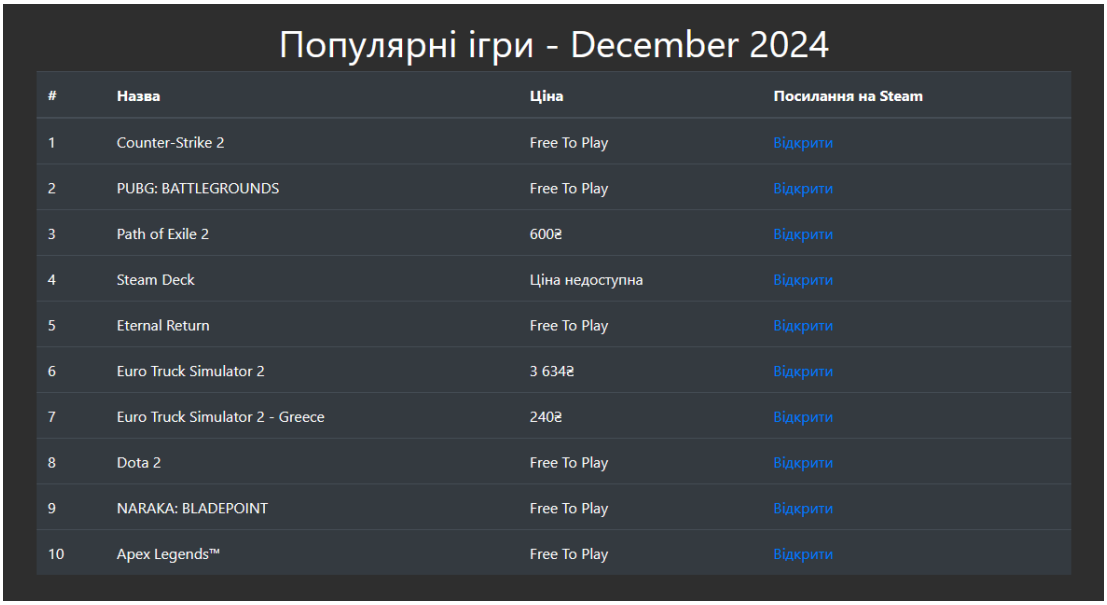


Рисунок 3.7 – Сторінка «Популярні ігри за актуальний місяць»

ВИСНОВКИ

У кваліфікаційній роботі представлено вебдодаток для відстежування змін у цінах на ігри з платформи Steam. У процесі розробки було розглянуто усі кроки які потрібно зробити для того, щоб створити додаток який буде збирати дані з HTML сторінок та візуалізувати їх на графіках.

Результати кваліфікаційної роботи:

– розроблено вебдодаток для візуалізації даних Steam;
– створено різні сторінки з інформацією о цінах;
– реалізовано графіки з відображенням змін цін;
– проведена оптимізація процесів оновлення даних;
– проведено тестування та налагодження проєкту.

Розробка вебдодатку є досить складною задачею, яка потребує уваги до деталей та вміння працювати з різними технологіями та бібліотеками. Проте, результат розробки – інформативний додаток який надає користувачам нову інформацію, якої немає в Steam.

ПЕРЕЛІК ПОСИЛАНЬ

1. Васильєв О. М. Програмування мовою Python. Тернопіль : Навчальна книга – Богдан, 2019. 504 с.

2. Костюченко А. О. Основи програмування мовою Python : навчальний посібник. Чернігів : ФОП Баликіна С. М., 2020. 180 с.

3. Офіційна документація бібліотеки BeautifulSoup URL : https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (дата звернення 15.04.2024).

4. Офіційна документація фреймворку Flask URL : https://flask.palletsprojects.com/en/stable/ (дата звернення 20.06.2024).

5. Яковенко А. В. Основи програмування. Python. Частина 1. Київ : КПІ ім. Ігоря Сікорського, 2018. 195 с.

6. Zelle J. Python Programming: An Introduction to Computer Science, 3rd Ed. Portland : Franklin, Beedle & Associates, 2016. 552 p.