МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет радіоелектроніки
Центр післядипломної освіти
Кафедра Програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

другий (магістерський)

(рівень вищої освіти)

Дослідження методів кластеризації даних при розробці сайта-агрегатора освітніх онлайн курсів

Виконав:

студент 2 курсу групи ІПЗздм-21-2

Ямпольський Г.В.

(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення

Тип програми Освітньо-наукова

Керівник доц. Назаров О.С.

(посада, прізвище, ініціали)

Допускається до захисту

Зав. Кафедри _____ З.В. Дудар

2023

Харківський національний університет радіоелектроніки

Факультет Комп’ютерних наук

Кафедра Програмної Інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)

Тип програми освітньо-наукова

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Ямпольського Германа Володимировича
(прізвище, ім’я, по батькові)

1. Тема роботи: Дослідження методів кластеризації даних при розробці сайта-агрегатора освітніх онлайн курсів

затверджена наказом університету від «03» 04 2023 р. № 83Стз

2. Термін подання студентом роботи до екзаменаційної комісії «10»05 2023р.

3. Вихідні дані до роботи встановлений календарний план роботи, методичні вказівки до оформлення пояснювальної записки, методи кластеризації та обробки даних.

4. Перелік питань, що потрібно опрацювати в роботі аналіз предметної галузі, огляд наявних математичних моделей, модифікація базових алгоритмів, дослідження можливості прискорення базових моделей, створення плану для подальшого дослідження теми.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз технічного завдання	01.04.2023 - 05.04.2023	виконано
2	Системний аналіз функціонування сайта-агрегатора	06.04.2023 - 10.04.2023	виконано
3	Кластеризація та обробка даних	11.04.2023 - 15.04.2023	виконано
4	Програмна реалізація	16.04.2023 - 20.04.2023	виконано
5	Підготовка пояснювальної записки	21.04.2023 - 25.04.2023	виконано
6	Підготовка презентації та доповіді	25.04.2023 - 30.04.2023	виконано
7	Перевірка роботи на плагіат та нормоконтроль	15.05.2023	виконано
8	Захист кваліфікаційної роботи	24.05.2023	виконано

Дата видачі завдання 01 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Назаров О.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Кваліфікаційна робота магістра містить: 95 с., 37 рис., 3 табл., 7 додатків, 45 джерел.

ДОДАТОК АГРЕГАТОР, КЛІЄНТ-СЕРВЕРНИЙ ДОДАТОК ОСВІТНІ ОНЛАЙН КУРСИ.

Робота присвячена створенню клієнт-серверної програми для агрегації інформації про освітні онлайн курси з різних платформ. Освітні онлайн курси займають лідируюче місце у сфері здобуття віддаленої освіти, що тягне за собою щоденне збільшення кількості, як самих курсів, так і сервісів, що надають можливість їх проходження. З цієї причини агрегацію даних про курси можна віднести до пріоритетних завдань сучасного медіа-простору.

В роботі були проаналізовані підходи популярних онлайн-платформ, що надають можливість проходження курсів. Були виявлені їхні переваги та недоліки, на підставі яких були сформульовані основні вимоги до застосування. Також вивчено теоретичну складову методів класифікації та кластеризації.

Розроблений додаток складається з серверної частини, що реалізує Representational State Transfer архітектуру, та клієнтської частини, що реалізує Single Page Application. Для серверної частини також розроблено програмний модуль для отримання даних про освітні курси з API сторонньої платформи Udemy.

Для розробки серверної частини програми був використаний backend-фреймворк Pyramid для мови програмування Python разом із ORM бібліотекою SQLAlchemy для роботи з СУБД PostgreSQL. Клієнтська частина була розроблена з використанням мікрофреймворку Vue.js, реалізованого мовою JavaScript.

EDUCATIONAL ONLINE COURSES, AGGREGATOR APPLICATION, CLIENT SERVER APPLICATION.

The work is devoted to the creation of a client-server program for aggregating information about educational online courses from various platforms. Online educational courses occupy a leading place in the field of distance education, which entails a daily increase in the number of both the courses themselves and the services that provide the opportunity to complete them. For this reason, the aggregation of data about courses can be attributed to the priority tasks of the modern media space.

The work analyzed the approaches of popular online platforms that provide the possibility of taking courses. Their advantages and disadvantages were identified, on the basis of which the main requirements for application were formulated. The theoretical component of classification and clustering methods was also studied.

The developed application consists of a server part that implements the Representational State Transfer architecture, and a client part that implements the Single Page Application. For the server part, a software module has also been developed to receive data about educational courses from the API of the third-party Udemy platform.

To develop the server part of the program, the Pyramid backend framework for the Python programming language was used together with the SQLAlchemy ORM library for working with the PostgreSQL DBMS. The client part was developed using the Vue.js microframework, implemented in the JavaScript language.

Умови публікації пояснювальної записки

Я, Ямпольський Герман Володимирович
(прізвище, ім’я, по батькові)
студент групи ІПЗздм-21-2 здобувач вищої освіти на другому (магістерському) рівні

кафедра програмної інженерії
(повна назва кафедри)

заявляю: моя кваліфікаційна робота на тему «Дослідження методів кластеризації даних при розробці сайта-агрегатора освітніх онлайн курсів», що буде представлена до ЕК для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлен з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

Освітні курси з відкритим доступом через Інтернет з'явилися на ринку приблизно 2008 року. Популярністю вони почали користуватися до 2012 року, коли найвідоміші майданчики Coursera та Udemy почали отримувати активне інвестування [1].

Зростання популярності було зумовлене тим, що проходження онлайн курсів дозволяє поєднати плюси освіти в установах та самостійної освіти. Учень може навчатися в будь-який зручний для нього час, але при цьому напрям його навчання скоригований професіоналами, які склали онлайн-курс.

Також ключовим фактором зростання популярності освітніх онлайн-курсів стало підвищення зручності при отриманні знань. Онлайн курси надають чітку програму, розбиту на уроки із зазначенням часу проходження у годинах, аналогічно до навчальних курсів у вищих навчальних закладах. До кожного уроку додається лекція, яка вбирає вичавку з матеріалів на тему даного уроку. Під час проходження онлайн курсу учень отримує доступ як до лекцій, складених кураторами курсу, так і до матеріалів, за якими готувався курс, в електронному вигляді. При класичній формі навчання матеріали часто надаються на лекціях, які потребують конспектування, у друкованому вигляді з бібліотек, або в електронному вигляді з обмеженим доступом із внутрішньої мережі навчального закладу.

Спочатку освітні онлайн курси дозволяли лише знайти нові знання, підкріпивши їх проходженням різноманітних тестів з дисципліни.Але згодом освітні майданчики розпочали видачу сертифікатів про проходження курсів, що дозволило підкріплювати своє резюме. Факт визнання успішного проходження онлайн курсів підвищив попит на цю форму навчання.

Зростання попиту над ринком означає зростання пропозицій. На сьогоднішній день по всьому світу функціонують тисячі платформ, що надають доступ до освітніх онлайн курсів. Така різноманітність породжує проблему

вибору платформи для подальшого пошуку курсів на ній. Таким чином, на ринку з'явилися платформи агрегатори освітніх онлайн-курсів. Подібні ресурси зберігають інформацію про курси з різних платформ-постачальників, дозволяючи переглядати інформацію про ці курси. Основним призначенням агрегаторів є ознайомлення користувачів із асортиментом товарів чи послуг, що надаються іншими сервісами. За допомогою агрегаторів користувач здатний зробити виважений вибір у найкоротший термін, оскільки агрегатори дозволяють порівнювати продукцію конкурентів у межах одного ресурсу.

Зараз на ринку існує чимало агрегаторів освітніх курсів, але більша їх частина працює некоректно і не дає постачальнику можливість швидко і зручно керувати курсами. Наявність вад у аналогів підштовхнуло до створення цієї програми.

Основною метою кваліфікаційної роботи магістра є створення програми-агрегатора освітніх курсів. Для досягнення цієї мети було сформульовано такі етапи розробки:

- збір інформації про ситуацію на ринку додатків-агрегаторів освітніх онлайн-курсів;
- виявлення та аналіз загальних недоліків існуючих агрегаторів;
- аналіз API популярних освітніх платформ та методів розробки закритих API;
- реалізація моделі бази даних;
- реалізація серверної частини програми;
- створення програмних засобів отримання даних з API;
- реалізація клієнтської частини програми.

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

Розробити десктопний додаток для сайту агрегатора освітніх онлайн курсів, який реалізує наступні функції:

- RFM-аналіз;
- групування клієнтів;
- попередня обробка даних для аналізу.

Програма, яка буде розроблена в результаті цієї роботи, повинна мати модульну структуру, що дозволяє модифікувати ядро алгоритму аналізу без глибоких знань з програмування, відповідно до побажань користувача (див.рис.1.1).

Рисунок 1.1 – Модульна структура системи

Крім того, він має бути частиною системи модулів, які забезпечують роботу сайту nazya.com. Таким чином, модель системи відповідає вимогам модульного програмування, а сама програма є модулем.

1.1 Аналіз технологій

Десктопний додаток – це програма, яка обробляється на стороні клієнта і виконується як звичайний виконуваний файл на пристрої користувача. Цим пристроєм у більшості випадків є комп'ютер. Розглянемо інструменти, які використовуються для реалізації цього типу додатків, та визначимо допоміжні програми.

1.1.1 Python

Python – це універсальна та сучасна мова програмування високого рівня, перевагами якої є висока продуктивність програмних рішень та структурований і легкий для читання код. Ядро має дуже зручну структуру та широкий спектр інтегрованих бібліотек, які забезпечують привабливий набір корисних функцій та можливостей. Мову програмування можна використовувати як для написання прикладних програм, так і для розробки веб-сервісів.

У науковому середовищі Python активно використовується у поєднанні з NumPy, SciPy та MatPlotLib для виконання різноманітних обчислень. Така комбінація навіть дозволяє модифікувати спеціалізовані комерційні пакети, такі як Matlab.

Python – відносно молода мова. Тому вона містить багато речей з інших мов, які розробники Python вважають найкращими з цих мов. Зокрема, ідеологія відступів для операторів групування та високорівневих структур даних була запозичена з ABC, деякі синтаксичні конструкції, які здавалися менш суперечливими творцеві мови, та інші унікальні особливості інших мов, такі як можливість слідувати обом парадигмам ООП та деякі особливості функціонального програмування.

Python часто порівнюють з Perl та Ruby. Як і Perl, Python можна успішно використовувати для написання сценаріїв.

Як і Ruby, Python є добре розробленою системою для ТОР. У той же час, реалізація ТОР у Python відрізняється від багатьох інших об'єктно-орієнтованих мов. Зокрема:

- на відміну від Ruby, Python не дотримується ідеології "все є об'єктом" і підтримує вбудовані примітивні типи, які не є частиною ієрархії класів. Це спрощує взаємодію між мовами і робить її технічно більш ефективною, хоча ентузіасти об'єктно-орієнтованого програмування можуть вважати її громіздкою;

- на відміну від деяких об'єктно-орієнтованих мов програмування (Java, Object Pascal, Ruby), у Python немає справжнього загального базового класу, від якого всі об'єкти успадковують спільні методи. Хоча в Python новий клас формально успадковує (прямо чи опосередковано) від типу об'єкта, це лише синтаксичний прийом, оскільки спільні для всіх об'єктів методи - id, type, isinstance, issubclass, str, repr, getattr - не успадковуються від об'єкта, а реалізуються як глобальні функції. Наслідком такого рішення є те, що зміна поведінки цих методів відбувається не шляхом перевантаження, а шляхом визначення спеціальних методів класу.

У комерційних додатках швидкість роботи програм на Python часто порівнюють зі швидкістю роботи додатків на Java.

Простота мови дозволяє людині з невеликими знаннями програмування легко додати, замінити або видалити модуль з програми.

1.1.2 LibreOffice

LibreOffice – це потужний офісний пакет, повністю сумісний з 32/64-бітними системами. Його перекладено більш ніж 30 мовами. Він сумісний з найпоширенішими операційними системами, включаючи GNU/Linux, Microsoft Windows і Mac OS X.

LibreOffice є безкоштовним і має відкритий вихідний код, що означає, що його можна завантажити і використовувати безкоштовно. LibreOffice є безкоштовним для індивідуального, шкільного та бізнес використання.

Пакет включає в себе текстовий процесор, електронну таблицю, програму для створення і перегляду презентацій, редактор векторної графіки, систему управління базами даних і редактор формул. Основним форматом файлів, що використовується в програмі, є міжнародний формат OpenDocument, але також можлива робота з іншими поширеними форматами, такими як Office Open XML, DOC, XLS, PPT і CDR.

Офісний пакет поширюється під публічною ліцензією MPL 2.0, що дозволяє вільно встановлювати і використовувати його в домашніх і комерційних

організаціях, а також на персональних комп'ютерах і в освітніх установах.

LibreOffice Calc є похідним від програми електронних таблиць OpenOffice.org. Він розповсюджується під ліцензією Mozilla Public License v2.0.

Можливості електронних таблиць :

Можливість читання/запису OpenDocument (ODF), Excel (XLS), CSV та інших форматів.

Підтримує 1 мільйон рядків/рядків в електронній таблиці, що робить Calc добре придатним для складних наукових або фінансових документів. Кількість стовпців не перевищує 1024, що набагато менше, ніж в Excel (16384 стовпців).

1.1.3 Microsoft Office Excel

Microsoft Excel – це програмний пакет, розроблений компанією Microsoft для використання в середовищі Windows (див.рис.1.2).

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	user_id	назва	ціна	дата	шлях категорії	назва категорії							Індекс товару
2	1765474	3	1000.00	18.03.2023	/taobao/Меблі для будинку на замовлення/Кухонні шафи і акс	Кронштейни і полиці для НВЧ			ТАК	ТАК	НІ	НІ	319010504
3	2478	5	600.00	18.03.2023	/taobao/Все для будинку і Хобі/Ящики для зберігання прилад	Настільні ящики для канцелярії			НІ	ТАК	НІ	ТАК	3050101020804
4	1849413	7	6500.00	18.03.2023	/taobao/Аксесуари, галантерея/Сумки/Гаманці	Гаманці			НІ	ТАК	НІ	НІ	302120101
5	1849410	9	1250.00	18.03.2023	/taobao/Спорт/Наколінники, напульсники	Наколінники, напульсники			ТАК	ТАК	НІ	НІ	21513
6	1849410	11	1250.00	18.03.2023	/taobao/Спорт/Наколінники, напульсники	Наколінники, напульсники			ТАК	ТАК	НІ	НІ	14112
7	1849410	13	1250.00	18.03.2023	/taobao/Спорт/Наколінники, напульсники/Захисне споряджен	Захисне спорядження для футболу			ТАК	ТАК	НІ	НІ	3291801
8	1849410	15	1250.00	18.03.2023	/taobao/Спорт/Наколінники, напульсники/Захисне споряджен	Захисне спорядження для єхоша			ТАК	ТАК	НІ	НІ	329160401
9	1849410	17	1250.00	18.03.2023	/taobao/Спорт/Наколінники, напульсники/Захисне споряджен	Захисне спорядження для футболу			ТАК	ТАК	НІ	НІ	32914040107
10	1849410	19	1250.00	18.03.2023	/taobao/Спорт/Наколінники, напульсники/Захисне споряджен	Захисне спорядження для тренажерно			ТАК	ТАК	НІ	НІ	329160502
11	1849410	21	1250.00	18.03.2023	/taobao/Спорт/Наколінники, напульсники/Захисне споряджен	Захисне спорядження для бейсболу			ТАК	ТАК	НІ	НІ	32914040108
12	1849410	23	1250.00	18.03.2023	/taobao/Спорт/Наколінники, напульсники/Захисне споряджен	Захисне спорядження для бадмінтону			ТАК	ТАК	НІ	НІ	32914050107
13	1849410	25	1250.00	18.03.2023	/taobao/Спорт/Наколінники, напульсники/Захисне споряджен	Захисне спорядження для волейболу			ТАК	ТАК	НІ	НІ	3291501
14	1849410	27	1250.00	18.03.2023	/taobao/Спорт/Наколінники, напульсники/Захисне споряджен	Захисне спорядження для тенісу			ТАК	ТАК	НІ	НІ	329160502
15	1849410	29	1250.00	18.03.2023	/taobao/Спорт/Наколінники, напульсники/Захисне споряджен	Захисне спорядження для екстремаль			ТАК	ТАК	НІ	НІ	3291901
16	1849410	31	1250.00	18.03.2023	/taobao/Спорт/Наколінники, напульсники/Захисне споряджен	Захисне спорядження для баскетболу			ТАК	ТАК	НІ	НІ	3291404109
17	1849410	33	1250.00	18.03.2023	/taobao/Спорт/Наколінники, напульсники/Захисне споряджен	Захисне спорядження для настільного			ТАК	ТАК	НІ	НІ	3291601
18	1849410	35	1250.00	18.03.2023	/taobao/Спорт/Наколінники, напульсники/Захисне споряджен	Захисне спорядження для фітнеса			ТАК	ТАК	НІ	НІ	3291801

Рисунок 1.2 – Приклад даних в програмі

Цей програмний пакет дозволяє автоматизувати інституційну, виробничу або наукову діяльність, пов'язану зі збором, зберіганням, обробкою, передачею та використанням інформації. Багато документів (робочі плани, атестації, довідки, звіти, рахунки, фінансові документи тощо) містять текстові та числові дані в табличній формі для зручності читання. Для автоматизації роботи з ними використовуються спеціальні програми (а точніше програмні пакети), що називаються електронними таблицями (ЕТ), в залежності від типу представлення та обробки даних, які вони використовують, в тому числі і пакет Excel.

Завдяки своїй тривалій присутності на ринку, а також, ймовірно, зручності, MS Excel на сьогоднішній день є найпопулярнішою програмою серед користувачів. Вона призначена для створення електронних таблиць будь-якої складності, виконання математичних розрахунків, вирішення завдань у сфері статистики та фінансового аналізу, а також створення діаграм і таблиць.

Основною вимогою до програми є можливість модернізації алгоритмічного ядра користувачем, який не володіє глибокими знаннями з програмування, наприклад, шляхом заміни алгоритмічного ядра RFM на таке, на яке аналітик може покладатися більше, ніж на те, що використовується в даний момент. Це можливо завдяки модульній структурі програми та використанню Python як мови розробки, оскільки вона має велику кількість бібліотек, як вбудованих, так і тих, що постійно додаються програмістами з усього світу, і є простою у використанні навіть для користувача з невеликими знаннями програмування.

Також було вирішено використовувати MS Office Excel для підготовки даних для завантаження з бази даних, оскільки він є більш звичним і зручним для маркетингових компаній, ніж його безкоштовний аналог LibreOffice Calc.

1.2. Огляд платформ освітніх онлайн курсів

Найвідомішими джерелами освітніх курсів з відкритим доступом через Інтернет є майданчики Coursera, Udemy та AcademyOcean. Саме вони були обрані як основні аналоги розроблюваного агрегатора освітніх курсів.

1.2.1 Coursera

Ресурс Coursera (див.рис.1.3) позиціонує себе як платформу, на якій будь-яка людина в будь-якому місці може навчатися та отримувати дипломи від найкращих університетів світу та постачальників освітніх послуг [2].

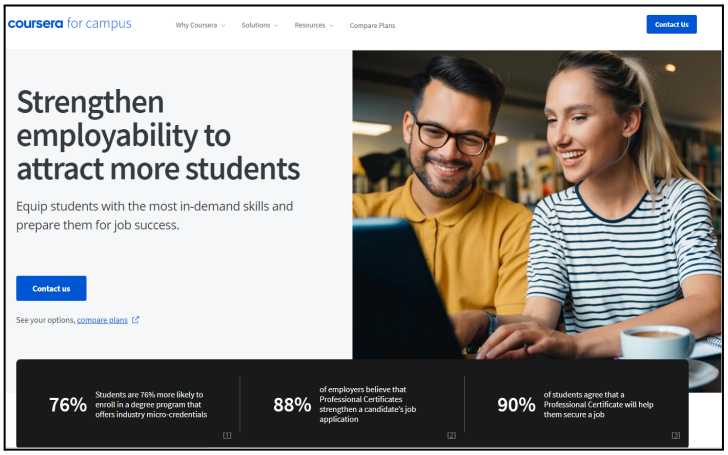

Рисунок 1.3 – Платформа Coursera

Курси включають записані відео лекцій, автоматично оцінені та рецензовані завдання, а також дискусійні форуми спільноти. Однак Coursera не надає можливості стороннім платформам розміщувати свої освітні курси.

1.2.2 Udemy

Udemy (див.рис.1.4) є онлайн-платформою для викладання та навчання [3]. На відміну від академічних програм освітніх онлайн-курсів, які ґрунтуються на традиційних університетських курсах, Udemy розміщує контент від творців онлайн контенту для продажу з метою отримання прибутку.

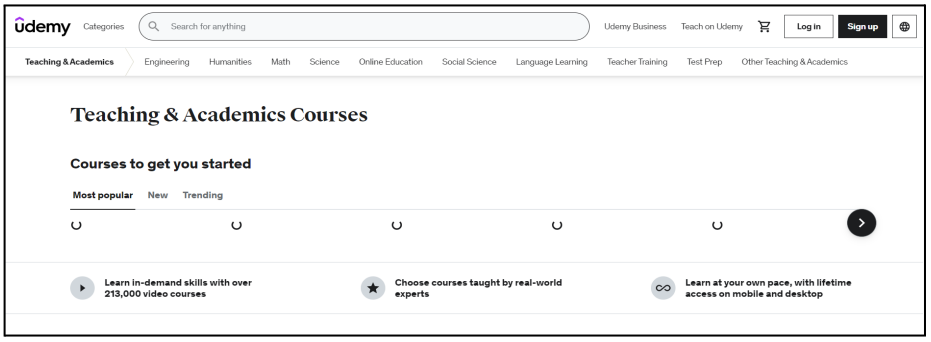

Рисунок 1.4 – Платформа Udemy

Udemy надає користувачам можливість створювати курси, просувати їх та заробляти гроші за навчання. Незважаючи на те, що Udemy дозволяє всім користувачам розміщувати свої онлайн-курси, ресурс не передбачає розміщення освітніх курсів сторонніх платформ.

1.2.3 AcademyOcean

AcademyOcean – це платформа для продажу та проведення навчання [4]. Платформа поєднує у собі інструменти управління навчанням та взаємовідносинами з клієнтами (див.рис.1.5). Як і Udemy, сервіс AcademyOcean не надає можливості взаємодіяти з онлайн-курсами, розміщеними на інших платформах.

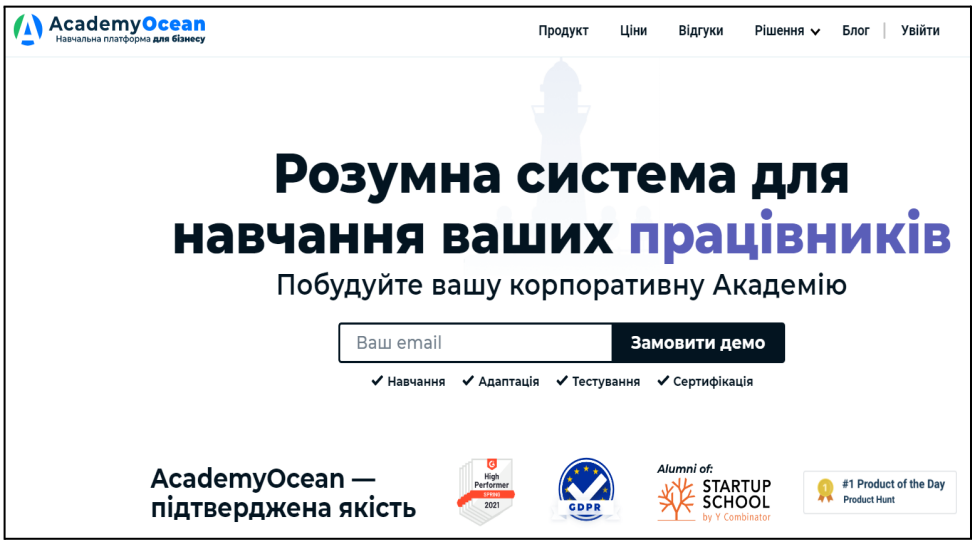

Рисунок 1.5 – Платформа AcademyOcean

Варто зазначити, що жоден із перерахованих сервісів не є повним аналогом розроблюваного агрегатора освітніх курсів, оскільки не надає настільки широкого доступу до розміщення освітніх матеріалів. У таблиці 1.1 містить основні відмінності програми від конкурентів.

У клієнт-серверному додатку, що розробляється, серверна частина відповідає за роботу двох незалежних клієнтських додатків: мобільного та web-клієнта. Мобільний IOS додаток призначений для зручної навігації та пошуку за освітніми курсами.

Таблиця 1.1 – Характерні відмінності програми від конкурентів

	Агрегатор	Coursera	Udemy	Academy Ocean
Генерація особистої збірки	Є	Є	Є	Є
Наявність курсів інших платформ	Є	Ні	Ні	Ні
Спрощене додавання курсів	Є	Ні	Ні	Ні
Виконання завдань курсу	Ні	Є	Є	Є
Перегляд лекцій курсу	Ні	Є	Є	Є

Web-клієнт надає можливість додавати інформацію про курси освітньої платформи та керувати нею з метою поширення курсів поза платформою.

2 СИСТЕМНИЙ АНАЛІЗ ФУНКЦІОНУВАННЯ САЙТА-АГРЕГАТОРА

2.1 Основні поняття системного аналізу

Термін "системний аналіз" або "системний підхід" не отримав загальноприйнятого стандартного тлумачення, незважаючи на те, що людина використовує його вже багато років. Причиною цього є динамічність усіх процесів практично у всіх сферах людської діяльності і, крім того, принципова можливість застосування системного підходу практично до будь-якої проблеми, що вирішується людиною [5].

У всьому світі системний підхід отримав стрімкий розвиток як в технологічному, так і в організаційному управлінні, причому не як абстрактна теорія, а як реальний інструмент управління, що робить працю людини більш продуктивною. Системний підхід використовувався при розробці та реалізації великих військових проектів, програм пілотованих польотів на Місяць і скрізь, де потрібно було планувати і організовувати діяльність сотень компаній з різними власниками і специфікою роботи. Він використовувався для проектування та управління окремими організаціями, як великими, так і малими, і зараз став основою міжнародної мови лідерів, що дозволяє їм розуміти один одного.

Підхід до організаційного розвитку повністю базується на принципах системного мислення. Методи операційного дослідження базуються на ідеї системного підходу до аналізу складних проблем та синтезу інструментів для їх вирішення. Математичні моделі стали засобом виявлення зв'язків між елементами сил і засобів, що використовуються в операціях [6].

Наразі системний аналіз – це прикладна наука, метою якої є виявлення причин реальних труднощів, з якими стикається "власник проблеми", та розробка варіантів їх усунення. Існує багато визначень терміну "системний аналіз", і, узагальнюючи їх, можна дати наступне тлумачення. Системний аналіз – це міждисциплінарна наукова галузь. Об'єктом системного аналізу є концепції та основи постановки і вирішення практичних проблем на основі системного мислення. В якості інструментарію системного аналізу використовується широкий

спектр математичних методів: лінійне та нелінійне програмування, теорія прийняття рішень, імітаційне моделювання.

Об'єкт – це будь-яка реальна сутність, яку можна виокремити за певними характеристиками в нескінченно різноманітному світі [7]. Ще одним фундаментальним поняттям системного аналізу є суб’єктивність. Суб'єкти – це фізичні або юридичні особи, які об'єднують людей у неподільні частини. Яка різниця між предметом і суб'єктом? По-перше, суб'єкт здатний не тільки взаємодіяти з навколишнім середовищем, що є спільним для всіх об'єктів, але й оцінювати свою взаємодію з навколишнім середовищем. По-друге, суб'єкт має здатність переслідувати мету, тобто досягати мети.

Об'єктом системного аналізу є завдання, що застосовуються на різних ієрархічних рівнях, що включає створення та розробку нових систем щодо організації, технологій, фінансів тощо. Це підводить нас до наступного поняття системного аналізу: складна система.

"Система – це засіб для досягнення мети" і "система – це сукупність взаємопов'язаних елементів, відокремлених від середовища і взаємодіючих з ним як єдине ціле". Ф.П. Тарасенко пояснює поняття "система" для цілей системного аналізу шляхом перерахування властивостей системи [8]. Ці властивості повинні задовольняти наступним вимогам:

- вони повинні бути притаманні всім системам;
- знання кожної з цих властивостей необхідне для конкретного кроку процесу розв'язання задачі.

Кількість властивостей – 12, і їх можна класифікувати на три групи: статичні, динамічні та синтетичні властивості.

Однак таке визначення не дає можливості чітко визначити, що є, а що не є системою. Тому поняття системи доповнюється класифікаціями та уточненнями. У літературі зустрічаються різні класифікації:

- за типом поведінки (детерміновані, ймовірнісні, ігрові);
- за характером мети (відкриті та закриті);
- за складністю структури та поведінки (прості та складні);

- за типом науки, що використовується для їх моделювання (математичні, фізичні, хімічні);
- за ступенем організації (добре організовані, погано організовані та самоорганізовані) [9].

Імовірнісні або стохастичні системи – це системи, поведінка яких описується законами теорії ймовірностей. Даних про поточний стан системи та взаємозв'язки між елементами недостатньо для того, щоб з упевненістю передбачити майбутню поведінку системи. Для такої системи існує безліч можливих переходів з одного стану в інший, тобто серія трансформацій стану системи, і кожен сценарій має свою ймовірність. Прикладом стохастичної системи є туристична агенція. Кількість проданих квитків залежить від кількості запитів і поведінки об'єктів.

Наступна класифікаційна ознака базується на типі призначення [10]. Відкриті системи взаємодіють з навколишнім середовищем і обмінюються енергією, масою та інформацією. У закритих системах зворотні зв'язки вивчаються всередині системи, і вони ізольовані від зовнішнього світу. Різниця між відкритими і закритими системами визначається точною чутливістю моделі.

Складні системи по-різному реагують на зовнішні впливи, залежно від свого внутрішнього стану. Крім того, спостерігач не може передбачити поведінку системи. У двох різних випадках система може дати однаковий або зовсім інший результат на одну й ту саму дію. Це відбувається тому, що складна система формує власні закони і правила. Вона характеризується великою кількістю внутрішніх зв'язків.

Проста система має невелику кількість можливих станів, поведінку яких можна легко описати за допомогою математичної моделі. Вона досить передбачувано реагує на зовнішні впливи.

Тому дисципліну системного аналізу можна розділити на три основні напрямки:

- створення моделі об'єкта, що вивчається;
- постановка задачі дослідження;

– розв'язання проблеми.

Розглянемо коротко кожну з цих фаз. Моделювання – це опис процесу мовою математики. Воно детально описує процеси, які цікавлять дослідника. Опис робиться відповідно до вимог дослідника, а якість моделі визначається узгодженістю результатів, отриманих за допомогою моделі, з перебігом спостережуваного процесу або явища. Від якості моделі залежить результат загального аналізу системи.

Етап визначення дослідницького завдання полягає у формулюванні мети аналізу. Мета сама по собі є об'єктом дослідження і повинна бути формалізована. Завдання системного аналізу полягає у виконанні необхідного аналізу невизначеностей, обмежень і формулювань, та у вирішенні оптимізаційної задачі.

Математичні методи повністю використовуються на етапі вирішення проблеми. Слід зазначити, що завдання системного аналізу можуть мати ряд особливостей, які вимагають використання евристичних підходів на додаток до формальних методів.

2.2 Поняття лояльності споживача

Лояльність – це якісна маркетингова характеристика ставлення споживачів до товарів і послуг, що базується на їхній прихильності до певного бренду.

Споживачі належать до однієї з трьох груп лояльності, які наведені на рисунку 2.1.

Рисунок 2.1 – Три групи лояльності клієнтів

Лояльні клієнти – це ті, хто довіряє послугам компанії та користується ними регулярно. Клієнти, що розвиваються, також користуються послугами інших компаній. Клієнти, що зазнали поразки – це ті, хто вже припинив купувати у цільової компанії або незабаром припинить купувати у неї.

У сучасному світі слово "лояльний" перекладається з французької та

англійської як "fidèle" - вірний. Лояльність має щонайменше два значення:

- лояльність до існуючих законів і постанов влади;
- коректне та доброзичливе ставлення до когось або чогось.

Лояльність у маркетингу – це встановлення довгострокових відносин, завдяки яким клієнт доброзичливо ставиться до продукту, бренду чи магазину і стає постійним покупцем.

Маркетинг лояльності ще називають маркетингом відносин.

Поняття лояльності поєднує в собі три ключові елементи:

- довіра;
- прихильність до цінностей;
- довгострокові відносини.

У сучасному світі програми лояльності мають ряд цілей, необхідних для виживання в умовах конкуренції. Розвиток і підвищення лояльності клієнтів допомагає вирішити багато аспектів розвитку комерційної компанії. Цілі лояльності:

- підвищення впізнаваності бренду;
- підвищення лояльності клієнтів до вашої продукції;
- збільшення продаж продукту, який ви просуваєте, в період промо-акції;
- збільшення частоти покупок відвідувача торгової мережі.

Правильно впроваджена програма лояльності має низку переваг перед роздрібними компаніями, які не мають діючих програм лояльності. До переваг впровадження програм лояльності відносяться:

- вигоди;
- зниження витрат на взаємовідносини з клієнтами;
- стабільність грошових потоків.

Існує 4 типи ресурсів, доступних кожному споживачеві:

- часовий ресурс;
- грошовий ресурс;
- когнітивний ресурс (пізнання);
- афективний ресурс (емоційне ставлення).

Основне бажання покупця – придбати необхідний йому товар з найменшими втратами цінних ресурсів: швидше, дешевше, простіше і без емоційного навантаження.

Задоволення головного бажання споживача призводить до його лояльності до ритейлера.

2.3 Завдання збільшення лояльності

Лояльність клієнтів до компанії чи продукту має дві сторони: зовнішню та внутрішню. Внутрішня сторона – це сторона клієнта. Лояльність споживача зростає з кожною досконалою покупкою, доки він відчуває, що економить важливі ресурси, такі як час, гроші та розумові зусилля. Зовнішня сторона - це сторона продавця, тобто комерційної компанії, яка пропонує послугу або продукт. Кожна покупка клієнта збільшує обсяг бізнесу, створеного клієнтом, збільшуючи суму його середнього чека або частоту його покупок. Іншими словами, для компанії лояльність клієнта виражається його купівельними характеристиками: частотою покупок і обсягом створеного бізнесу.

Тому завдання побудови лояльності клієнта можна вирішувати двома способами: за рахунок збільшення частоти покупок клієнта або за рахунок збільшення середнього чека. Іншими словами, показником лояльності покупця є його активність в мережі як активного покупця: чим більше і частіше він робить покупки, тим лояльніше він ставиться до ритейлера (формула 2.1).

Нехай $z_i \in Z$ – запис у базі даних (відповідає дисконтній картці), де Z – множина записів, а $|Z|=n$.

Для кожного $z_i \in Z$:

$$\begin{aligned} F(z_i) &\to \max, \\ M(z_i) &\to \max, \end{aligned} \tag{2.1}$$

де $F(z_i)$ – частота покупок за карткою;

$M(z_i)$ – продаж, здійснений покупцем за карткою.

2.4 Програма збільшення лояльності клієнтів

Програма лояльності – це набір маркетингових заходів, спрямованих на залучення та утримання клієнтів з метою збільшення продажів, продажу більшої кількості товарів чи послуг або підвищення інтересу до корпоративних заходів.

Програми лояльності – це маркетингові методи, призначені для побудови довгострокових відносин із клієнтами та підтримки їх лояльності. Лояльність полягає в розумінні потреб клієнта та розробці послуг, яких він потребує. Програма лояльності спрямована на підвищення задоволеності клієнтів вашою компанією [11].

Місія програм лояльності полягає у створенні стабільної клієнтської бази. Відповідно до закону Парето (80:20), який базується на статистичних дослідженнях, 20% клієнтів генерують 80% прибутку. Щоб утримати ці 20% клієнтів, необхідно розраховувати програми лояльності, адже для компаній залучення нового клієнта в 6-11 разів дорожче ніж утримання існуючого. А найкращий спосіб утримати клієнтів – запропонувати їм вигоду, коли вони купують товар або послугу у вашій компанії.

Добре сплановані програми лояльності можуть стати чудовим способом розширити клієнтську базу. Наприклад, понад 60% канадських домогосподарств беруть участь у програмі Air Miles Canada.

Сьогодні існує багато видів програм лояльності: дисконтна картка, накопичувальна програма, програма бонусів і подарунків, реферальна програма тощо. Беручи участь у програмі, клієнт зазвичай заповнює анкету зі своїми контактними даними, що дозволяє компанії після первинного аналізу даних інформувати його про нові продукти та послуги, які можуть його зацікавити.

Дані про покупки клієнтів дозволяють скласти профіль поведінки клієнта, що дає змогу компаніям краще пізнати своїх клієнтів, їхні інтереси та вподобання, звички та методи здійснення покупок, а також місця, де вони віддають перевагу покупкам.

Найважливішим принципом є правильне використання інформації про клієнтів, щоб підвищити їхній інтерес до ланцюга поставок і тим самим збільшити продажі та загальний оборот ланцюга поставок (див.рис. 2.2).

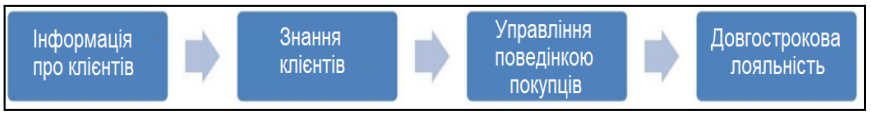

Рисунок 2.2 – Принцип роботи програми лояльності

Перевагами програми лояльності є:

- розширення знань про клієнтів завдяки аналізу даних;
- аналіз купівельного кошика;
- створення профілів клієнтів;
- поділ покупців на групи;
- формування вигідних торгових пропозицій;
- залучення та утримання клієнтів;
- збільшення обсягу продажів;
- закріплення довгострокових ділових відносин з клієнтами.

Одним з найважливіших завдань програми лояльності є аналіз аудиторії. Або визначте групи людей, які, швидше за все, придбають продукт або скористаються послугою. У більшості випадків основним методом аналізу цільових груп є кластеризація: поділ клієнтів на групи зі схожими характеристиками, визначення потреб групи та формулювання пропозиції, адаптованої до цільової групи.

Одне з найвідоміших правил мотиваційного менеджменту звучить так: "Ви можете ефективно управляти тільки тим, що можете виміряти". Зарубіжні експерти стверджують, що не існує такого поняття, як абсолютно лояльний клієнт. Реальною цільовою групою є клієнти, які повертаються протягом тривалого періоду часу.

Лояльність клієнта можна визначити як позитивне ставлення клієнта до продукту, бренду, магазину, послуги тощо, яке, безумовно, є наслідком важливих для клієнта факторів, але часто знаходиться в емоційній сфері.

Розглянемо детальніше модель програм лояльності. Рисунок 2.3 ілюструє принцип взаємодії між відділами компанії.

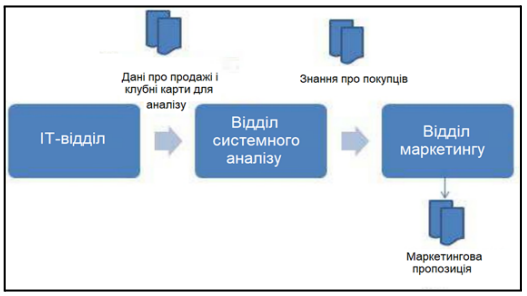

Рисунок 2.3 – Взаємодія відділів компанії

IT-відділ відповідає за надання даних для аналізу відділом системного аналізу. У свою чергу, відділ системного аналізу повинен використовувати дані для отримання знань про клієнтів у спосіб, зручний для маркетологів, і передавати цю інформацію до відділу маркетингу. Залежно від ситуації, мережеві маркетологи формулюють конкретну маркетингову пропозицію для клієнтів і взаємодіють з ними для реалізації цієї пропозиції.

У цій роботі відділ системного аналізу представлений особою, яка проводить дослідження. Відділ системного аналізу відповідає за встановлення специфікацій для IT-відділу для управління первинними даними для аналізу, а також за надання інформації про кінцевого споживача у відповідному форматі маркетологам, щоб після аналізу цих даних персонал відділу маркетингу міг розробити конкретну маркетингову пропозицію для реалізації конкретної акції зі стимулювання збуту для продуктів мережі. Ця маркетингова пропозиція може бути обговорена з працівниками інших відділів, щоб отримати їхню експертну думку щодо переваг промо-акції. Нарешті, після затвердження промо-акції відділ маркетингу розробляє технічне замовлення на реалізацію промо-акції та надсилає його до IT-відділу або, якщо це можливо, зв'язується з замовником самостійно.

На рисунку 2.3 показано узагальнену схему взаємодії між відділами в рамках програми лояльності. Слід зазначити, що всі частини системи, задіяні в розробці програми лояльності, є рівнозначними, і що розвиток клієнтської бази порушується, якщо порушується хоча б один елемент системи.

Зауважимо, що окрім створення та впровадження маркетингових рекомендацій також відстежується відгук клієнтів та динаміку змін у клієнтській базі. Крім того, при формулюванні нових маркетингових пропозицій слід також враховувати минулий досвід динаміки та еволюції клієнтської бази.

Отже, розглянуто основні принципи програми лояльності. Основна частина присвячена здобуттю знань про клієнтів у відділі системного аналізу.

При створенні та розвитку програм лояльності виникає проблема аналізу цільових груп, тобто визначення групи людей, які найчастіше купують товари або користуються послугами. У більшості випадків основним методом аналізу цільових груп є кластеризація, тобто поділ клієнтів на групи зі схожими характеристиками, визначення потреб групи та формулювання пропозиції, адаптованої до цільової групи.

Отже, розглянуто основні методи сегментації даних, їх переваги та недоліки при розробці програми лояльності клієнтів. Також розглянуто основні методи сегментації бази даних. Для кожного з цих методів були проаналізовані їхні підвиди.

Таким чином, було проаналізовано принципи роботи програми лояльності, що дозволить нам поглибити проблему та зосередитися на конкретному аспекті її розвитку: ефективній сегментації та аналізі даних для отримання інформації про клієнтів.

2.5 Інформаційна модель

2.5.1 Модель зберігання даних

Клієнт-серверний додаток «Агрегатор освітніх онлайн курсів» надає інформацію про освітні курси для ознайомлення з їх змістом. Для цього сервіс використовує власну базу даних, де зберігається частина інформації про освітні курси. Повну інформацію про освітній курс можна отримати за посиланням на першоджерело, яке також зберігається в базі даних. Такий підхід дозволяє клієнтській частині швидко здійснювати навігацію та пошук за курсами.

Для роботи програми використовується реляційна база даних SQL.

Інформація зберігається за допомогою моделі даних у третій нормальній формі.

На діаграмі сутностей бази даних, що зображена на рисунку 2.4, описані зв'язки сутностей між собою. Для кожної сутності та many-to-many зв'язку в базі даних існує таблиця.

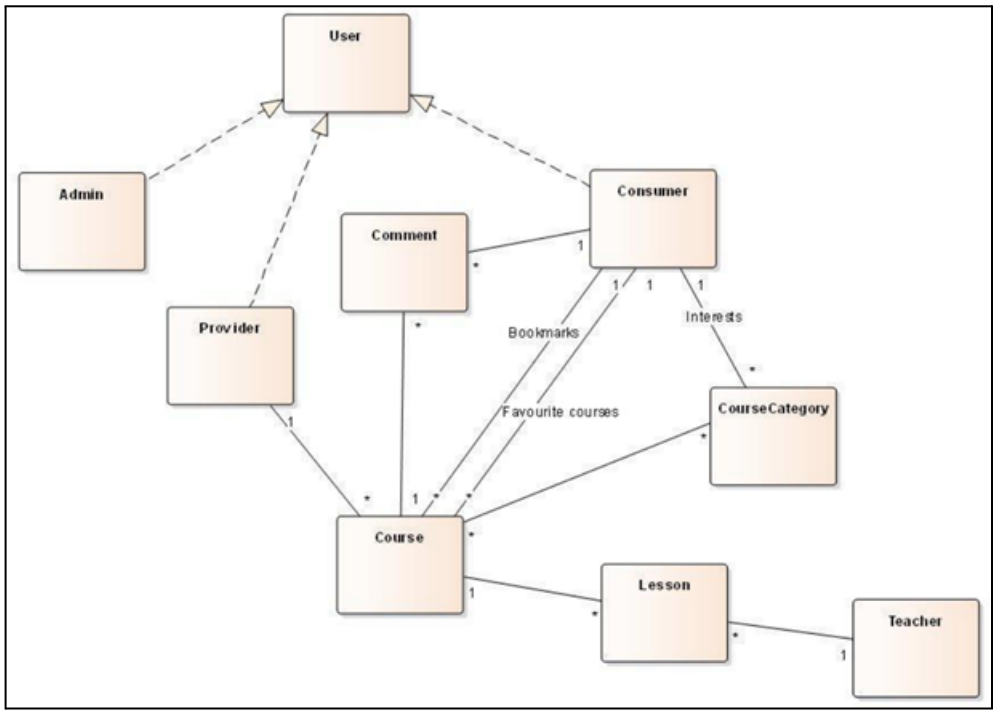

Рисунок 2.4 – Діаграма сутностей бази даних

Таблиці описані у таблиці 2.1.

Таблиця 2.1 – Таблиці бази даних

Таблиця	Опис
user	Користувач програми. Зберігає дані, необхідні для отримання прав доступу та авторизації: email, пароль та роль.
consumer	Користувач-споживач. Зберігає особисті дані споживачів: ім'я, стать, володіння мовами, інтереси, зображення профілю.
provider	Користувач постачальник. Зберігає дані постачальників: ім'я, сайт, інформацію.
admin	Користувач-адміністратор. Зберігає технічні дані адміністраторів ресурсу.

Кінець таблиці 2.1

Таблиця	Опис
course	Освітній курс. Зберігає дані про курси: назву, автор, посилання на оригінал, складність, опис, зображення.
course_category	Категорія курсів. Зберігає дані про категорії курсів: назву, опис.
comment	Коментар до курсу Зберігає дані про коментарі: автор, текст коментаря.
lesson	Урок/лекція курсу. Зберігає дані про урок освітнього курсу: назву, вчитель, курс.
teacher	Вчитель. Зберігає інформацію про вчителя: имя.
course_course_category	Many-to-many зв'язок категорій та курсів.
consumer_fav_courses	Many-to-many зв'язок споживачів та їх обраних курсів.
consumer_bookmarks	Many-to-many зв'язок споживачів та закладок.
consumer_interests	Many-to-many зв'язок споживачів та обраних категорій.

Взаємодія з базою даних здійснюється за допомогою об'єктно-реляційної СУБД PostgreSQL [12]. Цей вибір був зроблений через високу продуктивність цієї СУБД, а також підтримку роботи з великими масивами даних, спадкування, розширюваності.

PostgreSQL підтримує велику кількість типів даних, що конвертуються безпосередньо в типи даних Python за допомогою ORM бібліотеки SQLAlchemy.

У додатку реалізовано другий тип зберігання даних – файлову систему сервера. Цей тип зберігання даних використовується для зберігання файлів (наприклад, зображень). Такий підхід пов'язаний з тим, що зберігання файлів у базі даних сильно навантажує всі операції у ній. Тому в базі даних зберігаються лише назви файлів у файловій системі сервера.

2.5.2 Методи обробки даних

У базу даних інформація може вноситися вручну за допомогою клієнтської програми та за допомогою API сервера. Для збереження великих масивів даних із

сторонніх платформ було реалізовано допоміжний програмний модуль, який взаємодіє з API сторонніх освітніх платформ. На даний момент реалізовано клас для отримання та обробки даних з онлайн-платформи Udemy.

Щоб передавати запити на API Udemy [13], необхідно отримати унікальний ключ, оскільки API є закритим. Для отримання ключа було зареєстровано профіль розробника для API Udemy. Дані із сервісу Udemy отримані на законній основі.

Для роботи агрегатора потрібні не всі дані про курс, тому спеціально розроблений клас UdemyAPI спочатку вибирає необхідні дані, а потім перетворює їх до виду, необхідного для збереження бази даних.

Через особливості API Udemy одним запитом можна отримувати дані не більше ніж про 10000 курсів. Розроблений клас UdemyAPI дозволяє завантажувати дані частинами, фільтруючи курси пошукового рядка. Після отримання дані зберігаються в базі, зображення курсів перетворюються та зберігаються у файловій системі сервера.

Також клас UdemyAPI дозволяє завантажувати коментарі, залишені користувачами Udemy до курсу. На даний момент реалізований метод отримує 20 найпопулярніших текстових коментарів, але це число може бути змінено залежно від потреб сервісу.До методів обробки даних також можна віднести процес збереження файлів на сервері. З файлів зараз на сервері зберігаються лише зображення споживача та курсу. Ці зображення зберігаються у файловій системі каталогу проекту сервера. Для зменшення навантаження на сервер при доступі до файлів усі зображення зберігаються в підкаталогах, названих за першою та другою літерою назви курсу. У цих підкаталогах створюються підкаталоги з ID курсу, де зберігаються всі пов'язані з цим курсом файли. На даний момент там зберігається оригінал збереженого зображення та перетворена версія розміру 256x256 для відображення клієнтської програми. Підкаталоги створюються при необхідності зменшення навантаження. Приклад розбиття підкаталогів показаний на рисунку 2.5.

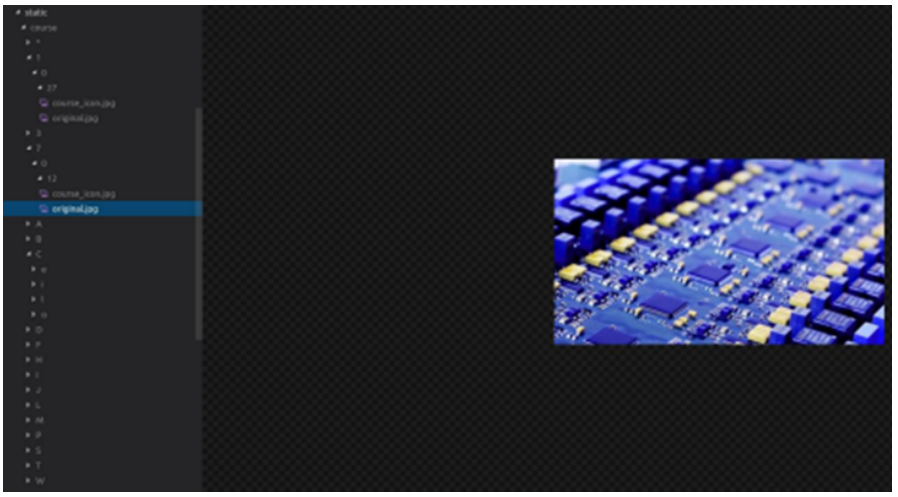

Рисунок 2.5 – Зберігання зображень у файловій системі

Приклад отриманих даних про курс з програмування мовою Python:


```
{
 "_class": "course",
 "id": 1748188,
 "title": "Основи програмування на Python 3",
 "url": "/python3-ua/",
 "is_paid": true,
 "price": "2 800 грн.",
 "price_detail": {
 "amount": 2800,
 "currency": "GRN",
 "price_string": "2 800 грн.",
 "currency_symbol": "грн."
 },
 "visible_instructors": [
 {
 "_class": "user",
 "title": "Julien Duraj",
 "name": "Julien",
 "display_name": "Julien Duraj",
 "job_title": "Software engineer",
 "image_50x50": "https://i.udemycdn.com/user/50x50/49955668_cc3c.jpg",
 "image_100x100": "https://i.udemycdn.com/user/100x100/49955668_cc3c.jpg",
 "initials": "JD",
 "url": "/user/julien-duraj/"
 }
 ],
 "image_125_H": "https://i.udemycdn.com/course/125_H/1748188_f994_4.jpg",
 "image_240x135":
"https://i.udemycdn.com/course/240x135/1748188_f994_4.jpg",
 "is_practice_test_course": false,
 "image_480x270":
"https://i.udemycdn.com/course/480x270/1748188_f994_4.jpg",
 "published_title": "python3-ua"
      }
```

Якщо курс містить у назві невизначений символ, його зображення зберігається у підкаталог під назвою «*».

Зображення формату 256x256 створюється не в момент збереження вихідного зображення, а за першої спроби отримати це зображення з сервера. Це зроблено зменшення навантаження на сервер у момент додавання зображення курсу. Існує можливість перетворення зображення до будь-якого розміру. Для цього необхідно додати потрібний розмір зображення в Enum PictureSize у модулі lib/utils (див. рис. 2.6). І далі спробувати отримати зображення з потрібним розміром із сервера. Якщо такого зображення немає, воно автоматично створиться.

```
19
20
21  class PictureSize(Enum):
22      course_icon = (256, 256)
23      consumer_icon = (512, 512)
24
25
26  def image_resize(path, img_name, resized_name, size):
27      image = Image.open(os.path.join(path, img_name))
28      image = image.resize(size.value, Image.NEAREST)
29      image.save(os.path.join(path, resized_name))
```

Рисунок 2.6 – Модуль lib/utils для роботи із зображеннями

При завантаженні зображень вручну вони перетворюються до розміру 256x256 та будь-якого іншого шляхом розтягування/стиснення. Було вибрано саме цей метод, оскільки він дозволяє привести зображення до необхідного для відображення вигляду без втрати інформації. Але для зображень, отриманих з API Udemy, було обрано інший метод. Зображення курсу мають стандартний розмір 480x270, тому перший метод зробив би всі зображення жахливо розтягнутими. В цьому випадку було вирішено обрізати зображення ліворуч і праворуч до розміру 270x270, а далі стискати до 256x256. Таке рішення було ухвалено після тривалого вивчення зображень курсів на Udemy. На рисунку 2.7 показаний приклад

оригінального та обрізаного зображень. Виявилося, що у більшості випадків зображення спочатку були «квадратної» форми та були підігнані сервісом Udemy під їхній «прямокутний» стандарт, тому обрізання бічних частин не призводило до втрати інформації.

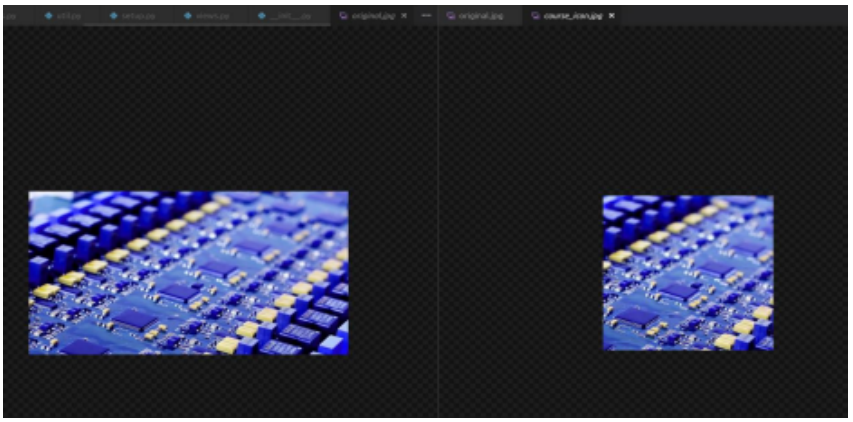

Рисунок 2.7 – Оригінальне та перетворене зображення

Структура функцій дозволяє в майбутньому додати будь-який файл до системи (наприклад, зображення вчителя або голосовий файл для курсу) без зайвих зусиль. Методи зберігання та доступу до файлів уніфіковані та готові до роботи.

3 КЛАСТЕРИЗАЦІЯ ТА ОБРОБКА ДАНИХ

3.1 Кластеризація даних

Одним з основних завдань створення програми лояльності є аналіз цільової групи, тобто виявлення групи людей, які, ймовірно, куплять пропонований товар або послугу. Основним методом аналізу цільової групи є кластеризація, тобто поділ клієнтів на групи зі схожими характеристиками, визначення потреб групи та створення пропозиції, адаптованої до цільового сегмента [14].

Кластеризація даних полягає в поділі заданої множини, що розділяє об’єкти на невеликі групи, які називаються колекціями, і кожна група складається з ідентичних об'єктів, а об'єкти в різних кластерах можна було чітко розрізнити. Завдання кластеризації пов'язане як зі статистичною обробкою, так і з широким класом неосвітніх навчальних завдань. Кластерний аналіз - це багатовимірний статистичний метод, який полягає в зборі даних з інформацією з вибірки об'єктів і подальшої їх класифікації в кластери [15].

Кластер – це група об'єктів, які відрізняються спільною характеристикою, і основною метою кластерного аналізу є пошук груп подібних об'єктів у вибірці. Однак різноманітність його застосувань призвела до появи великої кількості несумісних термінів, методів і підходів, що ускладнює чітке і послідовне застосування та інтерпретацію кластерного аналізу [16].

Кластерний аналіз виконує такі основні завдання [17]:

- розробка типології або класифікації;
- дослідження концептуальних схем, корисних для групування об'єктів;
- формування гіпотези на основі аналізу даних;
- перевірка гіпотези або дослідження з метою визначення того, чи дійсно відокремлені угруповання присутні в наявних даних.

Незалежно від цілей дослідження використання кластерного аналізу передбачає наступні етапи [18]:

- відбір вибірки для кластеризації;
- визначення набору змінних, які будуть використовуватися для оцінки

об'єктів у вибірці;

- розрахунок показників подібності між об’єктами;
- за допомогою методу кластерного аналізу сформувати групи схожих об'єктів;
- перевірити результати кластеризації.

Кластерний аналіз має наступні вимоги до даних [19]:

- показники не повинні порівнюватися між собою;
- показники повинні бути безрозмірними;
- розподіл показників має бути приблизно нормальним;
- показники повинні відповідати вимогам "стабільності", що означає, що на їх значення не впливають випадкові фактори;
- вибірка має бути однорідною і не містити викидів.

Якщо факторний аналіз передує кластерному аналізу, немає необхідності «виправляти» вибірку. Процес факторизації автоматично відповідає наведеним вище вимогам. В іншому випадку модель має бути скоригована [20].

Формальне пояснення задачі кластеризації [21]:

Нехай X – масив елементів і нехай Y – масив номерів стовпців. Задано функцію поверхні об’єкта $\rho(x, x')$. Кількість навчальних об'єктів $X^m = \{x_1, \ldots, x_m\} \subset X$ обмежена. Слід розбити вибірку на підмножини, які називаються кластерами. При цьому кожен кластер містить метрично пов’язані елементи, а елементи в різних категоріях дуже відрізняються один від одного. Кожному елементу $x_i \in X^m$ присвоюється номер кластера y_i.

Алгоритм кластеризації – це функція $a: X \to Y$, яка призначає номер кластера кожному об’єкту. У деяких випадках цей набір Y відомий заздалегідь, але завдання визначення точної кількості кластерів часто базується на певних критеріях якості кластерів [22].

Кластеризація відрізняється від класифікації тим, що початкові ідентифікатори об'єктів уі не фіксуються заздалегідь і їх кількість може бути навіть невідомою Y [23].

Розв'язання задачі кластеризації є принципово неоднозначним, що пов'язано з декількома причинами [24]:

- відсутність чіткого критерію якості кластеризації, тобто існує ряд евристичних критеріїв і навіть ряд алгоритмів, які не мають чітко визначеного критерію, але виконують кластеризацію "за конструкцією" досить обґрунтовано.;
- кількість кластерів зазвичай заздалегідь невідома і визначається за суб'єктивним критерієм;
- результат кластеризації сильно залежить від метрики, вибір якої також зазвичай є суб'єктивним і визначається експертним шляхом.

Існує декілька видів сегментації клієнтської бази даних в контексті аналізу клієнтської бази даних роздрібної торгівлі [25]:

- за профілем клієнта (соціально-демографічна сегментація);
- за даними про поведінку покупців (RFM-аналіз);
- за споживчою цінністю (ABC-аналіз);
- за асортиментними вподобаннями;
- за динамікою та активністю розвитку;
- за допомогою комплексної сегментації.

На жаль, неможливо розглядати будь-які маніпуляції з базою даних, не кажучи вже про кластеризацію, не стикаючись з проблемою надлишкової інформації або обмеженої обчислювальної потужності комп'ютера. Перш ніж приступити до розробки алгоритму кластеризації бази даних клієнтів роздрібної мережі, розглянемо детальніше проблему надмірності даних [26].

3.2 Можливості Data-mining

Методологія Data Mining перекладається як "видобуток даних" або "вилучення даних". У поєднанні з терміном "інтелектуальний аналіз даних" часто використовують терміни "виявлення знань у базах даних" та "видобуток даних". Поява цих термінів пов'язана з новим циклом у розвитку засобів і методів обробки даних [27].

Сфера застосування інтелектуального аналізу даних не обмежена: він має місце скрізь, де є дані. Але перш за все методи видобування даних сьогодні використовуються, м'яко кажучи, комерційними компаніями, які реалізують проекти зі створення сховищ даних. Досвід багатьох компаній показує, що рентабельність інтелектуального аналізу даних може досягати 1000%. Наприклад, щорічна економія в розмірі 700000 доларів. Впровадження інтелектуального аналізу даних на залізничній мережі Великобританії призвело до щорічної економії 700000 доларів [28].

Інтелектуальний аналіз даних є дуже цінним для менеджерів та аналітиків у їхній повсякденній роботі. Компанії усвідомили, що вони можуть отримати відчутні конкурентні переваги, використовуючи методи вилучення даних. Ми коротко опишемо деякі можливі бізнес-застосування інтелектуального аналізу даних [29].

До початку 2000-х років, здавалося, не було особливої потреби переосмислювати цю сферу. Все робилося в контексті галузі, яка називалася прикладною статистикою. Теоретики проводили конференції та семінари, писали вражаючі статті та монографії, рясніли аналітичними результатами [30].

Однак практики завжди знали, що спроби застосувати теоретичні вправи до реальних проблем часто закінчуються невдачею. Досі, однак, мало уваги приділялося проблемам практиків, які здебільшого вирішували свої приватні завдання з обробки невеликих локальних баз даних.

З удосконаленням технологій збору та зберігання даних величезні потоки інформації стали доступними для людей у різних сферах. Діяльність будь-якого підприємства сьогодні супроводжується записом і реєстрацією всіх деталей його діяльності.

Особливості сучасних вимог до обробки даних полягають у наступному [31]:

- обсяг даних необмежений;
- дані є різнорідними (кількісними, якісними, текстовими);
- результати мають бути точними та зрозумілими;

- основні засоби обробки даних повинні бути зручними у використанні.

Традиційна математична статистика, яка вже давно є корисним інструментом для аналізу даних, виявилася дуже складною. Основна причина - концепція вибіркового середнього, яка призводить до роботи з фіктивними величинами. Математичні та статистичні методи виявилися найбільш корисними для перевірки раніше встановлених гіпотез (видобуток даних) і для аналізу «необроблених» даних як основи для онлайн-обробки даних (OLAP) [32].

Сучасна технологія інтелектуального аналізу даних базується на концепції патернів, які відображають фрагменти багатовимірних зв'язків даних. Це характерні закономірності підвимірів даних, які можуть бути виражені в компактній формі, зрозумілій людині. Пошук патернів здійснюється за допомогою методів, які не обмежують апріорні гіпотези про структуру вибірки та тип розподілів значень аналізованих показників.

Загалом, інтелектуальний аналіз даних дуже добре визначає Григорій П'ятецький-Шапіро, один із засновників цього напряму:

«Інтелектуальний аналіз даних – це процес визначення раніше невідомої, корисної, дієвої та інтерпретованої інформації з вихідних даних, необхідної для прийняття рішень у різних сферах людської діяльності».

Сьогодні роздрібні торговці збирають детальну інформацію про кожну покупку за допомогою кредитних карток і комп'ютеризованих систем управління. Типовими проблемами, які можна вирішити за допомогою інтелектуального аналізу даних у роздрібній торгівлі, є наступні:

- аналіз подібності використовується для визначення продуктів, які клієнти хочуть купувати разом. Знання про купівельний кошик необхідне для покращення реклами, розробки стратегії запасу потрібних товарів та їх розміщення в торговельному просторі;
- дослідження часових патернів допомагає маркетологам приймати рішення щодо запасів. Воно відповідає на питання на кшталт: "Якщо клієнт купив відеокамеру сьогодні, чи буде він купувати нові батарейки та плівку завтра?".

Прогнозне моделювання дозволяє маркетологам краще зрозуміти природу потреб різних сегментів споживачів зі специфічною поведінкою, наприклад тих, хто купує у відомих дизайнерів або відвідує магазини. Ці знання необхідні для розробки цілеспрямованих і прибуткових заходів з просування продукції. При розробці заходів щодо просування та розподілу товарів слід залучати висококласних маркетологів-фахівців.

3.3 RFM-аналіз

RFM-аналіз – це інструмент для сегментування клієнтів відповідно до їхньої лояльності на основі їхніх минулих дій та прогнозування їхньої поведінки. RFM - це абревіатура термінів Recency (повторюваність), Frequency (частота) і Monetary (гроші), що означає вартість або інвестиції. Якщо розглянути ці терміни більш детально, то Recency означає ймовірність того, що клієнт повернеться, виходячи з часу, що минув з моменту останньої активності: чим коротший час, тим більша ймовірність того, що клієнт повернеться.

Параметр частоти – це кількість дій, здійснених клієнтом за певний проміжок часу. Вважається, що чим більше клієнт замовляє, тим більша ймовірність, що він замовить знову в майбутньому.

Грошовий параметр "Гроші" характеризується сумою грошей, яку покупець витратив протягом обраного періоду. Знову ж таки, чим більше грошей клієнт витратив, тим більша ймовірність того, що він витратить їх знову. Слід зазначити, що цей елемент часто може бути відсутнім в аналізі, оскільки він тісно пов'язаний з частотою. Грошовий елемент також може бути відсутнім у випадках, коли вигоди, отримані клієнтом, не можуть бути виражені в грошах.

У деяких ситуаціях замість грошей можна використовувати поняття "Тривалість", яке описує загальну тривалість роботи з конкретним клієнтом, тривалість його підписки.

Першим кроком є розгляд параметра Recency. По-перше, необхідно визначитися, що це є основою для бізнес-операцій клієнта. Це може включати покупки, відвідування магазинів і навіть переходи на веб-сайт компанії онлайн.

Все залежить від того, чого компанія хоче досягти за допомогою аналізу, а також від характеру активності, про яку йдеться.

Потім клієнтів слід класифікувати на п'ять груп, залежно від того, як довго вони роблять певні дії.

Найпростіший спосіб зробити таку класифікацію – визначити кілька часових циклів (у цьому випадку 5). Наприклад: дії за останній місяць, 1-2 місяці тому, 2-3 місяці тому, 3-6 місяців тому та 6 місяців-1 рік тому.

Групу, що складається зі споживачів, які лише нещодавно вжили заходів, зазвичай називають номером 5. Номер 1 – це група споживачів, які не вживали заходів протягом найдовшого періоду.

Наступним кроком в аналізі є робота з частотою. Принцип поділу споживачів на 5 груп такий самий, як і для частоти. Однак замість того, щоб ділити їх на періоди, слід визначити кількість транзакцій, яка має бути критерієм віднесення клієнта до тієї чи іншої групи. Приклад: 20 покупок і більше – група 5, 0-2 покупки – група 1. Водночас слід зазначити, що занадто високий або занадто низький поріг входу групи може суттєво вплинути на точність результатів. На рисунку 3.1 показано ідеальну ситуацію для сегментації RFM.

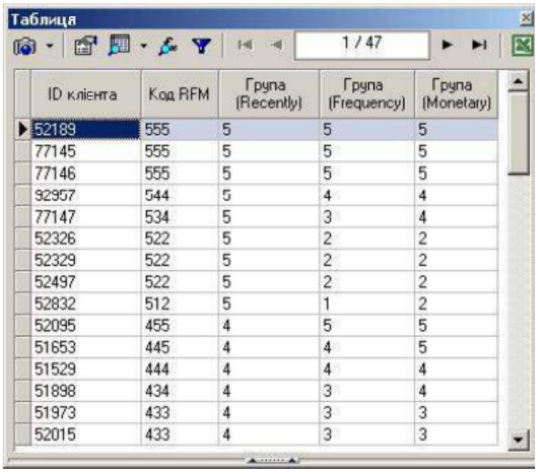

Рисунок 3.1 – Ідеальна ситуація RFM сегментації

Після процедури RFM-аналізу необхідно визначити грошовий параметр. Це те ж саме, що і в попередніх розділах. Умовою віднесення замовника до певної групи є перевищення порогового значення суми витрат.

Таким чином, після аналізу можна сформувати 125 груп від 111 до 555 (цифри – це комбінації номерів груп за кожним показником). Це означає, однак, що до споживачів у кожній групі слід підходити індивідуально. Тут буде корисно виявити різні тенденції в поведінці споживачів, основних клієнтів компанії та тих, хто тільки збирається стати лояльним.

Наприклад, група 555, безсумнівно, є найбільш лояльними клієнтами, яких може мати компанія. Але не думайте, що ви можете "забути про них", тому що вони нікуди не дінуться. Ви повинні показати цим клієнтам, що вони дійсно важливі для компанії, що компанія вдячна їм (шляхом створення спеціальних умов, програм лояльності).

Група 111 – найвідчайдушніші клієнти. Однак важливо розуміти, що вони хоча б раз користувалися послугами компанії. Спробувати залучити їх назад або зрозуміти причину їх низької активності – можливе завдання для маркетингових менеджерів компанії.

Нещодавні 5 клієнтів оцінюють компанію і можуть бути постійними клієнтами. До них можна впевнено звертатися за допомогою поштових розсилок або онлайн-кампаній і заохочувати до покупки.

Групі споживачів, які, як правило, купують невеликі партії, можна запропонувати супутні товари та послуги до їхніх покупок.

Таким чином, за допомогою RFM-аналізу можна отримати поділ основної бази на робочі сегменти. На рисунку 3.2 проілюстровано сегментацію бази даних на робочі сегменти: лояльність, міграція та розвиток.

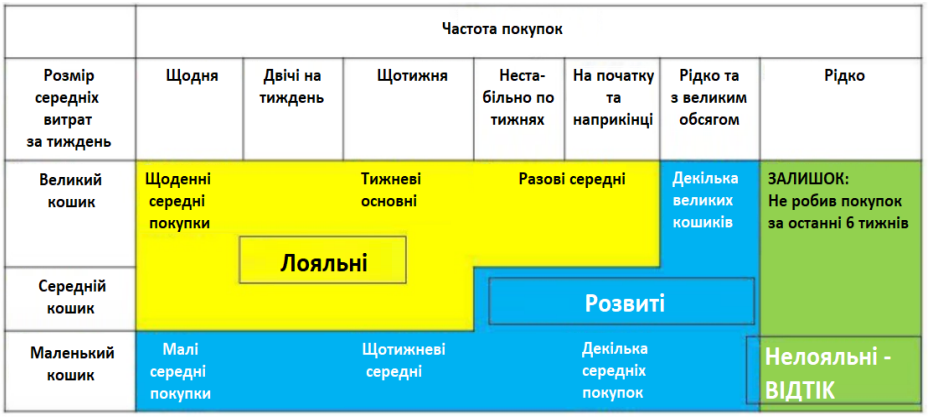

Рисунок 3.2 – Розбиття клієнтської бази на сегменти

Сегментація за допомогою RFM-аналізу полегшує відстеження динаміки сегментів клієнтів. На рисунку 3.3 показано приклад моніторингу динаміки функціональних сегментів RFM.

		Стало у новому періоді					
		1. Новачки	2. Потен-ційний відтік	3. Лояльні клієнти	4. Новачки VIP або касири	5. VIP клієнти	6. Відтік
Було у попередньому періоді	1. Новачки	10%	25%	20%	10%	5%	30%
	2. Потенційний відтік			20%			60%
	3. Лояльні клієнти		15%	80%			5%
	4. Новачки VIP або касири		10%	20%	50%	20%	
	5. VIP клієнти		20%			80%	
	6. Відтік		20%				70%

Рисунок 3.3 – Динаміка робочих сегментів

Сегментація клієнтської бази – одне з найважливіших завдань, яке може постати перед роздрібною мережею. Дослідження показало, що ключовими факторами лояльності покупців є частота покупок, час, проведений в базі даних, і сума грошей, витрачена покупцем. Сегментація на основі цих факторів є ключовим фактором у досягненні мети створення кращої програми лояльності клієнтів. Такий аналіз також полегшує відстеження динаміки лояльних і нелояльних сегментів клієнтів.

Алгоритм RFM-аналізу складається з наступних етапів.

Перший етап – класифікація за параметром рекурентності :

– визначення дати останньої покупки кожного клієнта;

– розрахунок для кожного покупця давності покупки (Recency) як різниці між поточною датою та датою останньої покупки;

– розподіл даних на 5 груп (квантилів). Кожному покупцеві присвоюється ідентифікатор від 1 до 5, залежно від його активності. Тим, хто здійснив нещодавню покупку, присвоюється код R=5, тим, хто не купував нічого протягом тривалого часу, присвоюється код R=1.

Другий етап – частотна класифікація:

– визначте для кожного покупця кількість покупок, здійснених за певний період;

− розділіть дані на 5 груп (квантилів). Клієнти, які зробили більше покупок, кодуються F=5, менш активні клієнти кодуються F=1.

Третій етап – сортування за грошовим параметром:

− визначення грошової суми, витраченої кожним покупцем;

− розподіл даних на 5 груп (квантилів). Клієнти, які витрачають більше, кодуються як M=5, ті, хто витрачає менше, кодуються як M=1.

Четвертий етап – об'єднання результатів, кожному клієнту присвоюється тризначний код RFM.

3.4 Адаптація алгоритму

Результати перших двох кроків – проведення RFM-аналізу та кластеризації RFM-груп – були незадовільними. Було прийнято рішення переглянути алгоритм. Зокрема, необхідно було переглянути часові рамки відбору карт лояльності для аналізу та створити більш наочні діапазони для параметрів Money і Frequency . Параметр R не був включений до RFM-аналізу. Було вирішено, що всі покупці, які відвідали сайт протягом звітного періоду, є цільовою групою, і що програма лояльності буде діяти для них лише протягом одного кварталу. Виключення параметра "Релевантність клієнта" не тільки призводить до додаткових витрат на розрахунки, але й дозволяє представити результати RFM-аналізу у вигляді простої таблиці. Щоб сформувати нові діапазони для параметрів "Гроші" та "Частота", ми провели частотний аналіз даних. Для цього ми розрахували середній чек і середній оборот за кожним рахунком. У таблиці 3.1 наведено середній чек та середню частоту покупок для різних акаунтів.

На жаль, середня частота покупок, здійснених клієнтами, сильно зміщена в бік "купив один-два рази".

Всі акаунти були згруповані в діапазони на основі середньої частоти або середнього чеку. У таблиці 1 показано групування клієнтів за середньою частотою покупок, здійснених клієнтом протягом тижня.

Параметр R (Recency) не враховувався в RFM-аналізі. Було вирішено, що всі покупці, які відвідували сайт протягом досліджуваного періоду, є цільовою

групою, і що програма лояльності застосовується до них лише протягом одного періоду.

Таблиця 3.1 – Середні показники облікових записів

ID акаунту	Ср. чек 1 покупки період, (грн.)	Ср. частота покупок за період
1765474,1,2,1	2874	1
1849413,1,3,1	6519	1
1849410,1,5,1	34398	1
…	…	…
1840687,1,3,1	4902	1

Виключення параметра релевантності клієнтів вивільняє додаткові обчислювальні потужності, необхідні для сегментації бази даних.

Алгоритм RFM:

1. Для кожного $z_i \in Z$:

Визначити $f_i = F(z)$ – частота здійснення покупок по карті.

Визначити $m_i = M(z)$ – товарообіг створюваний клієнтом, який використовує картку.

2. Сформувати множину B виду

$$B = \{\{m_1; f_1\} \ldots \{m_n; f_n\}\}, \tag{3.1}$$

3. Розбити множину B на підмножини

$$S_{fm} \in B, \tag{3.2}$$

так що

$$z_i \in S_{fm}\ , \tag{3.3}$$

якщо

$$f_{\min f} \le f_i \le f_{\max f}, \tag{3.4}$$

$$m_{\min\ m} \le m_i \le m_{\ \max m}, \tag{3.5}$$

де $[f_{\min}; f_{\max}]$ и $[m_{\min}; m_{\max}]$ – межі інтервалів, визначених експертом.

3.5 Кластеризація FM-сегментів

Для того, щоб об'єднати два підходи до сегментації клієнтської бази даних, було вирішено застосувати стандартний метод кластеризації. Попередній крок полягає у групуванні товарів у номенклатурні кластери.

Особливістю вибору методу кластеризації є те, що кількість кластерів, на які слід розбити вибірку сегментів КП, апріорі не відома. Для кластеризації було обрано метод ієрархічної кластеризації.

Планарні алгоритми кластеризації просто розбивають початкову множину об'єктів на кластери, кожен з яких представлений неструктурованою множиною об'єктів. Алгоритми ієрархічної кластеризації не страждають низькою інформаційною потужністю, оскільки для виконання таких алгоритмів потрібно більше часу, набори об’єктів можна представити кількома фіксованими групами, надаючи цінну додаткову інформацію для спеціалістів із обробки інформації про структури даних. Крім того, більшість алгоритмів ієрархічної кластеризації є детермінованими, що корисно в деяких випадках.

Алгоритми ієрархічної кластеризації можна розділити на дві групи:

- алгоритми ієрархічної кластеризації знизу вверх;
- алгоритми ієрархічної кластеризації зверху вниз.

Перша група алгоритмів працює за наступним принципом: окремі об'єкти спочатку об'єднуються в пари тісно пов'язаних об'єктів, а потім ці пари об'єднуються з іншими за тим же принципом до тих пір, поки не буде створено єдиний кластер, що містить всі точки початкової множини об'єктів. Алгоритми низхідної ієрархічної кластеризації працюють за протилежним принципом:

спочатку вибирається найбільший кластер, що містить всі об'єкти вибірки, а потім поступово розбивається на безліч підпорядкованих кластерів.

Об'єкти вибірки представляються у вигляді ієрархії кластерів, яку також називають дендрограмою (див. рисунок 3.4).

Як показано на рисунку 3.4, для введеної вибірки можна створити будь-яку кількість кластерів – тільки об'єкти як кластери включаються на найнижчому рівні. Потім вони з'єднуються один за одним з найближчими кластерами, і якщо задано правило розбиття дендрограми, тобто "відсікання" нижніх гілок, то такий алгоритм кластеризації здатний визначити кількість кластерів, необхідну для мінімізації цільової функції.

Рисунок 3.4 – Приклад дендрограми

Дендрограма зазвичай являє собою дерево, створене на основі матриці приблизних вимірювань. Дендрограма використовується для представлення взаємозв'язку між об'єктами в заданій множині. Для створення дендрограми необхідна матриця подібності для визначення ступеня подібності між парою кластерів. Найчастіше використовуються агломеративні методи.

Для створення матриці подібності необхідно визначити ступінь відстані між двома кластерами. У цій роботі ми використовуємо метод Уорда. На відміну від інших методів кластерного аналізу, використовується дисперсійний аналіз для оцінки відстані між кластерами. Відстань між кластерами – це збільшення суми квадратів відстаней об'єктів від центрів полюсів, отримане з'єднанням об'єктів (формула 3.6):

$$\Delta = \sum_{i} (x_i - \bar{x})^2 - \sum_{x_i \in A} (x_i - \bar{a})^2 - \sum_{x_i \in B} (x_i - \bar{b})^2 \quad (3.6)$$

На кожному кроці алгоритму два кластери об'єднуються для мінімізації дисперсії. Цей метод використовується для задач, де кластери дуже близькі один до одного.

Таким чином, було вивчено та проаналізовано основні методи та способи сегментації клієнтської бази в комерційних компаніях. Більш детально було розглянуто метод RFM-аналізу, який аналізує базу даних клієнтів на основі купівельної поведінки. Метод RFM-аналізу виявився ідеальним для розробленої в компанії програми підвищення лояльності клієнтів. У цьому розділі ми представляємо результати адаптації зарубіжного досвіду RFM-аналізу до реалій торговельної компанії на українському ринку.

Більша частина матеріалу містить інформацію про те, як ефективно поєднувати два підходи до сегментації бази даних, а саме: за поведінкою покупців і за купівельним кошиком. Ефективне поєднання цих підходів забезпечує широкі можливості для подальшого розвитку цільових груп. Найважливішим результатом роботи, представленої в третьому розділі, є рішення відмовитися від R-параметру і використовувати для подальшої ієрархічної кластеризації лише індекс товару, придбаного користувачем, а також F- і M-параметри.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ

Для реалізації програми було обрано мову програмування Python версії 3, оскільки вона була актуальною на момент створення програми, а отже, пропонує найкращу підтримку та найбільш активно розвивається і оновлюється бібліотечна база. Microsoft Office Excel використовується як допоміжний інструмент.

Системні вимоги: Емпірично доведено, що вузьким місцем програми є оперативна пам'ять комп'ютера, на якому вона працює. Для того, щоб програма обробляла дані і не давала збоїв, необхідно не менше 2 ГБ оперативної пам'яті DDR3 на частоті 1333 МГц.

Структура програми являє собою модульний набір інструментів, а саме запропоновану основну функцію для виконання підготовки даних, аналізу RFM та групування категорій за даними аналізу RFM. Структуру програми проілюстровано на рисунку 4.1.

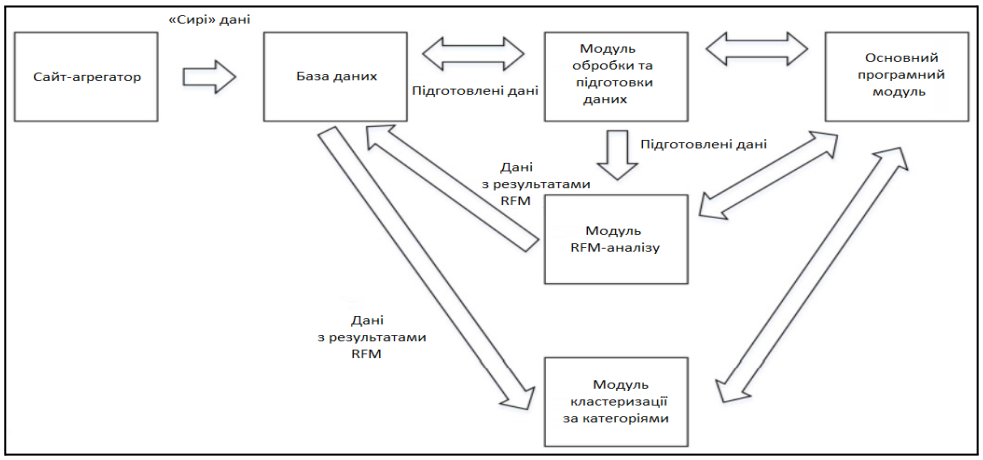

Рисунок 4.1 – Структурна схема програми

Загальний алгоритм роботи програми: Сайт-агрегатор зберігає дані у своїй базі даних. З цієї бази даних завантажуються "сирі" дані, а модуль обробки та підготовки даних приводить їх у форму, необхідну для подальшої роботи, викликаючи відповідну команду в головній програмі та зберігаючи їх поруч з базою даних.

Ці підготовлені дані можуть бути передані до модуля RFM-аналізу або безпосередньо до модуля групування, якщо така мета існує. У свою чергу, модуль

RFM-аналізу аналізує дані і в результаті своєї роботи модифікує підготовлені дані, які вже коректно запитані відповідною командою в головному модулі для передачі модулю кластеризації.

4.1 Модуль обробки та підготовки даних

Більшість алгоритмів кластеризації та аналізу, які є у відкритому доступі в Інтернеті, припускають, що дані, які вони обробляють, задовольняють певні вимоги. Тому було вирішено напівавтоматично адаптувати дані, завантажені з бази даних, до цих вимог:

- файл конвертується в формат .csv ;
- файл конвертується в кодування UTF-8;
- видаляється маркер порядку байт на початку файлу;
- поля рядків розділяються комами: ",";
- перевірити, чи потрібно використовувати символ `.` (крапка) як десяткову крапку в числах; якщо ні, то виправити;
- у полях, що містять інвертовані коми `"`, всі інвертовані коми дублюються;
- у полях, що містять інвертовані коми `,`, подвійні інвертовані коми `` або переноси рядка, всі інвертовані коми беруться в подвійні інвертовані коми ``.

На рисунку 4.2 наведено приклад вхідних даних, оброблених відповідно до вимог алгоритму RFM та для наочності відкритих у Блокноті.

```
order_date,user_id,order_value
2020-03-18,1765474,958.00
2020-03-18,1765474,958.00
2020-03-18,1765474,958.00
2020-03-18,2478,571.00
2020-03-18,2478,571.00
2020-03-18,1849413,6519.00
2020-03-18,1849410,1274.00
2020-03-18,1849410,1274.00
```

Рисунок 4.2 – Приклад попередньо оброблених даних

Такої обробки достатньо для проведення RFM-аналізу, але для ієрархічної кластеризації необхідно перевести дані про товарні категорії з текстового формату в числовий.

Для цього було прийнято рішення проіндексувати товари за категоріями та провести подальшу нормалізацію індексів. Для цього необхідно:

- відокремити категорії одна від одної;
- відсортувати їх за заданою ознакою, в даному випадку за алфавітом;
- пронумерувати базові категорії;
- якщо вкладена категорія належить до тієї ж базової категорії, що й попередня, її номер збільшується, в іншому випадку нумерація починається з початку;
- якщо вкладена категорія закінчується, вона стає базовою, а попередня вкладена категорія стає новою вкладеною категорією;
- повторюйте вищезазначені 2 кроки доти, доки не залишиться більше вкладених категорій.

В результаті індексування ми отримуємо числову інформацію про категорію товару для кожного запису в зовнішній базі даних (див. рисунки 4.3 та 4.4).

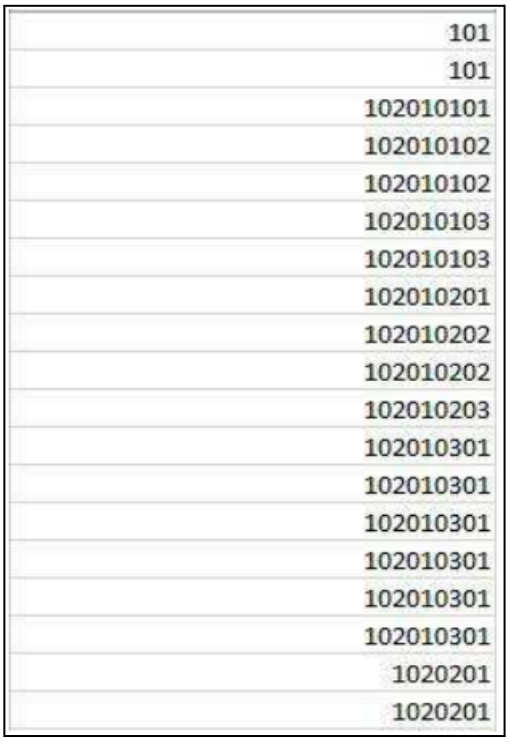

Рисунок 4.3 – Приклад індексів товарів

user_id	назва	ціна	дата	шлях категорій	назва категорії	індекс товару
1765474	Кронштейн	958.00	18 03 23	/taobao/Все для Будинку та Хобі/Будинок та сад/Будівельні	Кронштейни та полиці для НВЧ	2030404020201
1765474	Кронштейн	958.00	18 03 23	/taobao/Меблі для дому на замовлення/Кухонні шафи	Кронштейни та полиці для НВЧ	319010504
1765474	Кронштейн	958.00	18 03 23	/taobao/Будівельні, Оздоблювальні матеріали/Кухня	Кронштейни та полиці для НВЧ	145020201
2478	Настільний	571.00	18 03 23	/taobao/Побутова хімія, засоби гігієни, пристосовані	Настільні ящики для канцелярії	3050101020804
2478	Настільний	571.00	18 03 23	/taobao/Все для Будинку та Хобі /Ящики для зберігання	Настільні ящики для канцелярії	111130404
1849413	Гаманець	6519.00	18 03 23	/taobao/Аксесуари, галантерея/Сумки/Гаманці	Гаманці	302120101
1849410	Наколінник	1274.00	18 03 23	/taobao/Спорт/Наколінники,Напульсники	Наколінники,Напульсники	21513

Рисунок 4.4 – Зведена таблиця з даними про користувача, категорію та її індекс

Алгоритми кластеризації можуть більше працювати з числовою інформацією, ніж з текстовою.

4.2 Модуль RFM-аналізу

Оновлювати модуль RFM-аналізу не потрібно. Алгоритм RFM-аналізу описаний у додатку Е.

Найбільш цінними для подальшої роботи є дані, що зберігаються у файлі _mapping, оскільки він містить інформацію про користувачів та їхню приналежність до того чи іншого сегменту RFM-аналізу.

4.3 Ієрархічна кластеризація

З точки зору цієї роботи, цей елемент є "чорною скринькою", оскільки програма спеціально структурована таким чином, щоб дозволити використання будь-якого модуля алгоритму ієрархічної кластеризації. В даному випадку перевага надається модулю, який використовує перевірений метод Уорда як інструмент для вимірювання відстаней між кластерами. На рисунку 4.5 показано результат формування кластерів на основі приналежності користувача до ГК та придбаного ним продукту.

На рисунку 4.6 показано структуру програмного файлу, який дозволяє обмінюватися модулями.

Таким чином, методи та технології, описані в третьому розділі, а саме: RFM-аналіз, адаптація FM-сегментів, пакування груп BOM та кластеризація FM-сегментів, були реалізовані в програмі.

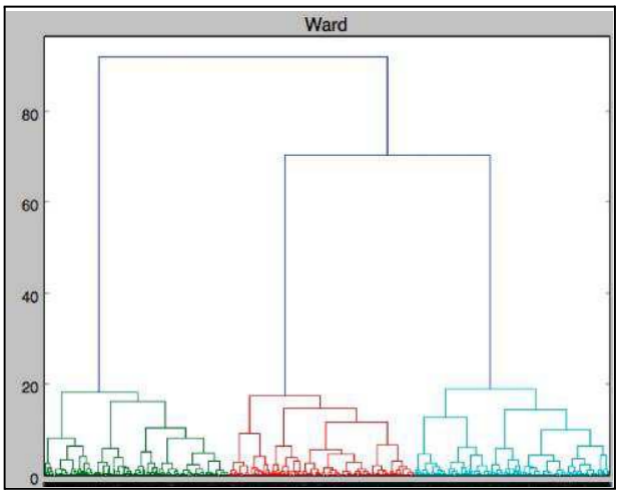

Рисунок 4.5 – Кластеризація користувачів

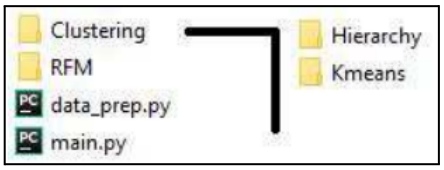

Рисунок 4.6 – Файлова структура програми

Розроблений програмний продукт дозволяє підготувати завантажені дані та розмістити їх у необхідному вигляді для подальшої експлуатації модулів, особливо модуля RFM-аналізу та модуля ієрархічної кластеризації. Крім того, програма дозволяє проводити RFM-аналіз та кластеризацію даних, які вона готує, а модуль кластеризації має можливість візуалізації результатів своєї роботи.

У цьому розділі представлено практичний результат адаптації зарубіжних експериментів з RFM-аналізу до умов торговельної компанії на українському ринку. Проблема полягала в тому, що зарубіжні експерименти не повною мірою відображали специфіку купівельної поведінки, а профіль іноземного покупця не збігався з профілем вітчизняного споживача.

Основним результатом, представленим у цьому розділі, є програма, яка після виконання всіх кроків створює дендрограму, що є поєднанням результату FM-сегментації бази даних Nazya.com та групування користувачів за їхніми товарними вподобаннями. Ця дендрограма представляє великий інтерес для маркетологів, оскільки відображає масу нових знань про клієнтів.

4.4 Опис серверної частини

4.4.1 Архітектура

Серверна частина реалізована відповідно до архітектури REST [33]. Це означає, що сервер працює за принципом закритого API. Взаємодія з клієнтами відбувається незалежно від технологій, за допомогою яких клієнт реалізований. Всі запити клієнта та відповіді сервера надсилають дані у форматі JSON [34]. Виняток становлять зображення – вони передаються у форматі image. Передача окремих файлів за допомогою не JSON запитів не суперечить положенням архітектури REST.

На рисунку 4.7 зображено архітектуру клієнт-серверної програми «Агрегатор освітніх онлайн курсів».

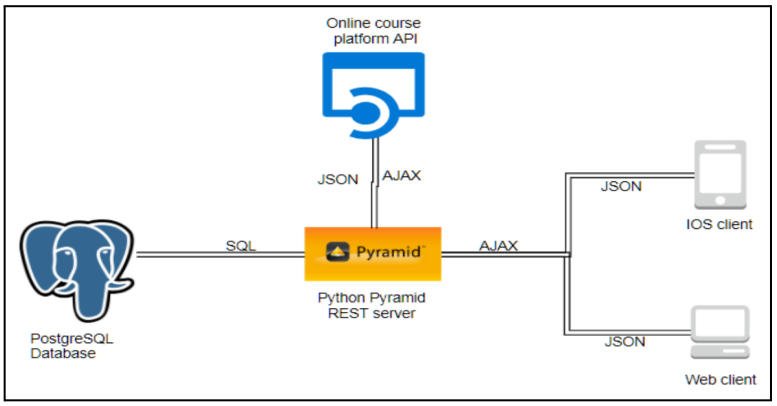

Рисунок 4.7 – Архітектура програми

Серверна частина програми реалізована за допомогою backend-фреймворку Pyramid [35] для мови Python [36]. Особливості структури цього фреймворку дозволяють розширювати прості додатки до більших масштабів. Pyramid має перелік вбудованих функцій, за допомогою яких можна реалізувати робочий сервер за кілька годин. Цей додаток було розроблено за тим самим принципом. Гнучкість системи дозволила з часом додавати новий функціонал та змінювати старий, не порушуючи працездатність серверної частини.

Також з особливостей Pyramid необхідно відзначити просунуту обробку запитів та зручне налаштування сервера з файлу конфігурацій, про які детальніше піде далі.

4.4.2 Взаємодія з базою даних

Для роботи Pyramid сервера з базою даних PostgreSQL було обрано бібліотеку SQLAlchemy [37]. За допомогою ORM ця бібліотека зіставляє класи Python сутності SQL бази даних.

Такий підхід дозволяє створювати SQL таблиці, код, запити мовою ORM бібліотеки, тобто мовою програмування Python. Для реалізації «важких» до виконання завдань є можливість використання нативного SQL, але не у формі виконання єдиної команди execute з єдиним параметром – рядком із SQL кодом, як у інших подібних бібліотеках. У SQLAlchemy можливий виклик всіх SQL функцій, включаючи унікальні функції для PosgreSQL, за допомогою Python коду. Наприклад, для масової вставки таблицю (див. рисунок 4.8) використовується SQL функція insert, куди параметром передається масив об'єктів для вставки. ORM виконав би операцію окремим insert для кожного об'єкта, що виконувалося б значно довше, навантажуючи систему інших транзакцій.

```
if len(to_remove):
    DBSession.execute(consumer_interests.delete()
            .where(and_(consumer_interests.c.consumer_id == consumer.id, consumer_interests.c.course_category_id.in_(to_remove)
if len(to_add):
    DBSession.execute(consumer_interests.insert(), [
        {'consumer_id': consumer.id, 'course_category_id': x} for x in to_add
    ])
```

Рисунок 4.8 – Приклад виконання SQL функцій

4.4.3 Безпека

«Агрегатор освітніх онлайн курсів» - це клієнт-серверний додаток, розрахований на кілька клієнтських програм, де більша кількість запитів до сервера захищена авторизацією. Тому безпеці у цій роботі присвячений окремий пункт.

Авторизація на серверній частині реалізована за допомогою технології JSON Web Token (JWT) [38]. Принцип JWT в автентифікації користувача на сервері за допомогою токена JSON – унікального ключа доступу. Для кожного користувача сервер генерує токен, ґрунтуючись на унікальному ключі користувача

(id, login, email), можливо з іншими JSON полями та алгоритмі шифрування (SHA512). На виході алгоритм видає Base64-URL рядок. Часто отриманий токен передається в тілі авторизованих JSON запитів. Тип Base64-URL дозволяє передавати токен параметром GET запитах. У реалізації цієї програми токен передається в заголовку запиту. Подібна особливість пов'язана з тим, що авторизація була реалізована за допомогою Cookie, але вона була замінена у зв'язку з невідповідністю критеріям безпеки. Для мінімізації масштабів змін було ухвалено рішення передавати токен у заголовку запиту.

У авторизованого користувача на сервері є три рівні прав доступу:

- споживач;
- постачальник;
- адміністратор.

Споживач може переглядати більшу частину інформації на сервісі, додавати коментарі та редагувати особисті дані. Постачальник може переглядати також більшу частину інформації на сервісі, може редагувати особисті дані та дані про свої курси. Адміністратор має право на всі можливі дії. За розподіл прав відповідає клас Root модуля security (див. рис. 4.9).

```
class Root(object):
    __name__ = ''
    __acl__ = [
        (Allow, 'admin', ALL_PERMISSIONS),
        (Allow, 'provider', ['add', 'view']),
        (Allow ,'consumer', ['view'])
    ]

    def __init__(self, request):
        pass
```

Рисунок 4.9 – Клас Root

Це технічний клас, необхідний призначення прав доступу.

Призначені права доступу можна надавати раутам. На рисунку 4.10 показано

раути з правами доступу view та add. До першого доступ отримати зможе будь-який авторизований користувач, а до другого лише постачальник та адміністратор.

```
@view_config(route_name='provider_view', permission='view', renderer='json')
def provider_view(request):
    try:
        provider = DBSession.query(Provider).get(int(request.matchdict['id']))
    except SQLAlchemyError as e:
        request.response.status = 400
        return {'msg' :  MESSAGES['id']}
    if not provider:
        request.response.status = 400
        return {'msg' :  MESSAGES['id']}
    request.response.status = 200
    return provider.to_json()

@view_config(route_name='course_add', permission='add', renderer='json')
@user_id_match
@json_match(schema=course_schema)
def course_add(request):
    course = Course()
```

Рисунок 4.10 – Права доступу до раутів

Сервер відповідає лише на запити із вмістом типу JSON, за що відповідає спеціальний валідатор. Для реалізації валідатора було підключено популярну бібліотеку jsonschema. У реалізації цієї програми бібліотека перевіряє цілісність даних та його типи. Для кожного типу запитів написано свою схему валідації. Наприклад, на рисунку 4.11 зображено схему валідації для додавання або редагування курсу.

```
course_schema = {
    'type' : 'object',
    'properties' : {
        'name' : {'type' : 'string', 'minLength' : 2},
        'author' : {'type' : 'string'},
        'link' : {'type' : 'string'},
        'description' : {'type' : 'string'},
        'language' : {'type' : 'string', 'enum' : LANGUAGES},
        'complexity' : {'type' : ['number', 'null']},
        'course_categories' : {'type' : 'array'}
    },
    'required' : ['name', 'author', 'link', 'description', 'complexity', 'course_categories', 'language']
}
```

Рисунок 4.11 – Схема валідації даних запитів для курсу

Єдиний допустимий тип змісту запиту, окрім JSON – image. Для фільтрації

запитів іншого змісту було створено технічний клас ContentTypePredicate (див. рис. 4.12). Цей клас перевіряє значення поля “content type” у заголовку для кожного запиту. За допомогою регулярних виразів визначається тип переданих даних та перевіряється їх наявність у змінній TYPES. На даний момент дозволено запити типу image з розширеннями png, jpeg, jpg та gif.

```
class ContentTypePredicate(object):
    def __init__(self, val, config):
        self.any_char = val.find('*')
        self.val = val

    TYPES = {
        'image' : ['png', 'jpeg', 'jpg', 'gif']
    }
    def text(self):
        return 'content type = %s' % self.val
    phash = text

    def __call__(self, context, request):
        if self.any_char < 0:
            return request.content_type == self.val
        else:
            p1 = re.compile('([a-zA-Z]+)/*')
            m1 = p1.match(self.val)
            p2 = re.compile('([a-zA-Z]+)/([a-zA-Z]+)')
            m2 = p2.match(request.content_type)
            return m1 and m2 and (m1.group(1) == m2.group(1)) and (m2.group(2) in self.TYPES[m1.group(1)])
```

Рисунок 4.12 – Клас ContentTypePredicate

Для забезпечення більшої безпеки всі отримані паролі користувача зберігаються в базі даних у хешованому вигляді. Рисунок 4.13 містить функції хешування паролів.

```
def hash_password(pw):
    hashed_pw = bcrypt.hashpw(pw.encode('utf-8'), bcrypt.gensalt())
    return hashed_pw.decode('utf-8')


def check_password(expected_hash, pw):
    if expected_hash is not None:
        return bcrypt.checkpw(pw.encode('utf-8'), expected_hash.encode('utf-8'))
    return False
```

Рисунок 4.13 – Хешування паролів

4.4.4 Обробка запитів

Більшість ресурсів сервера йде на обробку запитів. Не дивно, адже це його головна функція. Так само більшість зусиль розробника йде на реалізацію саме цієї складової серверної частини.

З особливостей реалізації обробки запитів даного сервера можна виділити оптимізацію, безпеку та надійність. У багатьох місцях операції ORM замінені виконання SQL коду. При редагуванні списку вибраних категорій курсів споживача стоїть завдання замінити старий список категорій на новий. Запис про кожну обрану категорію лежить у many-to-many таблиці. Для вирішення цього завдання було обрано найбільш оптимальний алгоритм (див. рис. 4.14):

- вибирається різниця множин старих та нових категорій;
- за вибраною множиною ID будується прямий SQL запит на видалення декількох елементів;
- вибирається різниця множин нових і старих категорій;
- за вибраною множиною ID будується прямий SQL запит на додавання кількох елементів.

```
if 'interests' in request.json.keys():
    old_interests = set(map(lambda x: x.id, consumer.interests))
    new_interests = set(request.json['interests'])
    to_remove = old_interests.difference(new_interests)
    to_add = new_interests.difference(old_interests)
    try:
        if len(to_remove):
            DBSession.execute(consumer_interests.delete()
                    .where(and_(consumer_interests.c.consumer_id == consumer.id, consumer_interests.c.course_category_id.in_(to_remove))
        if len(to_add):
            DBSession.execute(consumer_interests.insert(), [
                {'consumer_id': consumer.id, 'course_category_id': x} for x in to_add
            ])
    except SQLAlchemyError as e:
        LOG.exception(e.message)
        request.response.status = 500
        return {'msg' :  MESSAGES['db'], 'err' : e.message}
DBSession.flush()
request.response.status = 200
return {'msg' : MESSAGES['ok']}
```

Рисунок 4.14 – Алгоритм зміни списку обраних категорій

Розв'язання задачі «в лоб» спричинило б виконання спочатку n операцій видалення, а потім m операцій додавання в базу даних. Запропоноване рішення дозволило внести зміни за одну оптимізовану операцію видалення та одну оптимізовану операцію додавання.

Оптимізація стосується не тільки часу виконання коду, а й кількості коду. Величезна кількість перевірок користувача винесено в декоратор перед функціями обробки запитів. На рисунку 4.15 описаний декоратор, який перевіряє чи має авторизований користувач право на доступ до даного рауту.

```
def user_id_match(f=None):
    if f is None:
        return partial(user_id_match)

    @wraps(f)
    def decorated_function(request):
        try:
            user = DBSession.query(User).get(int(request.matchdict['id']))
        except SQLAlchemyError as e:
            request.response.status = 400
            return {'msg' :  MESSAGES['id']}
        if not user:
            request.response.status = 400
            return {'msg' :  MESSAGES['id']}
        if (user.group != 'admin') and (user.id != int(request.jwt_claims['sub'])):
            request.response.status = 403
            return {'msg' :  MESSAGES['access']}
        return f(request)
    return decorated_function
```

Рисунок 4.15 – Декоратор для перевірки відповідності користувача

Наприклад, на рисунку 4.16 споживач може редагувати лише свій профіль.

```
@view_config(route_name='consumer_edit', permission='view', renderer='json')
@user_id_match
@json_match(schema=consumer_schema)
def consumer_edit(request):
    consumer = DBSession.query(Consumer).get(int(request.matchdict['id']))
    if DBSession.query(User).filter(User.email == request.json['email']).filter(User.id != consumer.id).one_or_none():
        request.response.status = 403
        return {'msg' :  MESSAGES['email']}
```

Рисунок 4.16 – Приклад обробки запиту з декораторами

4.4.5 Конфігурація серверної частини

Як було згадано на початку розділу, Pyramid backend-фреймворк дозволяє встановити конфігурацію сервера зі спеціального файлу. У цій реалізації файл називається development.ini. При запуску серверної програми система зчитує всі налаштування файлу і запускається відповідно до них. Отримати доступ до конфігурацій із файлу можна в будь-якій точці коду. На рисунку 4.17 показані конфігурації для URL бази даних, робоча директорія проекту та конфігурація API Udemy із файлу development.ini.

```
sqlalchemy.url = postgresql://user:user@localhost:5432/banner_editor
work.directory = /home/banner_editor

udemy.id = szGL6cZ3fK9nuq40mV7NL0gGDdGzHKxb0UmJLOiQ
udemy.secret = YQqsSxiO53Yh72gXk9p7u9ajNWzMEsm3gziBjPRqDaEFCQpXMjoP4tKfaU8Z72hM4RkNOif2BR5pBhBcAYS
udemy.api.url = https://www.udemy.com/api-2.0/courses/
udemy.url = https://www.udemy.com/
```

Рисунок 4.17 – Файл конфігурацій сервера

4.4.6 Опис класів

У результаті реалізації серверної частини програми були розроблені класи, описані в таблиці 4.1.

Таблиця 4.1 – Таблиця класів

Назва	Успадкування	Опис
PictureSize	Enum	Клас опису розмірів зображень.
UdemyAPI		Клас для роботи з API Udemy.
CorsPreflight Predicate	object	Технічний клас обробки запитів. Необхідний для обробки запитів OPTIONS через протокол HTTP.
Image	object	Анотація клас для всіх SQLAlchemy моделей, для яких необхідно зберігати зображення.
Course	Base, Image	Клас ORM для сутності course.
Lesson	Base	Клас ORM для сутності lesson.
Teacher	Base	Клас ORM для сутності teacher.
CourseCategory	Base	Клас ORM для сутності course_category.
Comment	Base	Клас ORM для сутності comment.
ContentType Predicate	object	Технічний клас обробки запитів. Необхідний для перевірки змісту запиту.

Кінець таблиці 4.1

Назва	Успадкування	Опис
Root	object	Технічний клас для визначення прав доступу користувачів до раутів.
User	Base	Клас ORM для сутності user.
Consumer	User, Image	Клас ORM для сутності consumer.
Provider	User	Клас ORM для сутності provider.
Admin	User	Клас ORM для сутності admin.

4.4.7 Опис API сервера

У додатку Ж описано всі елементи серверної частини, до яких можуть звертатися клієнти. Для кожного наведено тип запиту, параметри, необхідність авторизації, короткий опис.

Таким чином, були докладно описані особливості реалізації серверної частини програми. Особлива увага була приділена опису технічних характеристик серверної частини, яка виконує роль API. Окремо сказано про особливості обробки запитів. Детально були розібрані методи забезпечення безпеки та серверної частини програми.

4.5. Опис клієнтської частини

4.5.1 Архітектура

Архітектура web-клієнта є Single Page Application (SPA). Особливістю цієї реалізації є відмова від оновлення html сторінки. Всі дані відображаються динамічно через AJAX запити або внутрішні обчислення на сторінці. Клієнтська частина виконується як нативний JS додаток у рамках середовища браузера.

На рисунку 4.18 вказано структуру проекту клієнтської частини програми. Можна відзначити, що є лише один html файл, в який динамічно підвантажуються vue-компоненти.

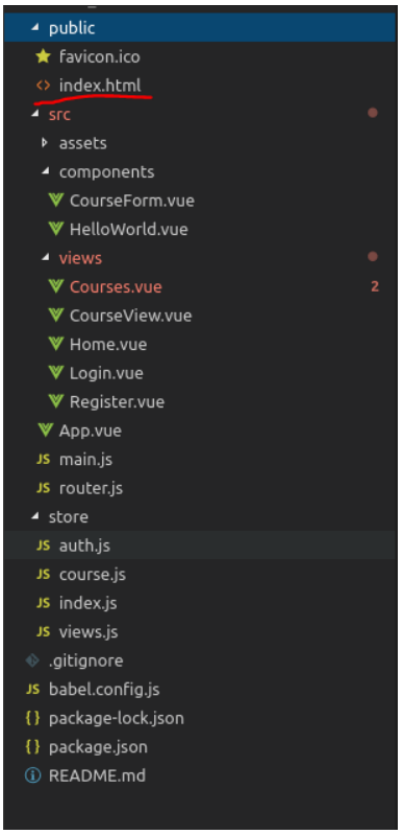

Рисунок 4.18 – Структура проекту клієнтської частини програми

Даний підхід має масу переваг перед класичним. Основною перевагою є підвищення продуктивності. Браузер не витрачає час на відображення подібних елементів під час переходу на інші сторінки, оскільки нові дані відображаються у вже відображених елементах. Це спричиняє зменшення розміру відповідей від сервера – браузер отримує весь клієнт у відповідь перший запит до серверу, а наступні запити лише оновлюють малогабаритні дані у форматі JSON.

Також при використанні SPA як web-клієнт, розробник перекладає безліч завдань на клієнт, зменшуючи навантаження на сервер. SPA можна вважати окремим додатком. Наприклад, якщо формат даних різних клієнтів не збігається, кожен клієнт може самостійно перетворити дані до необхідного вигляду, тоді як при класичному підході серверу необхідно було б генерувати для кожного клієнта унікальну відповідь, що в достатньому обсязі збільшує навантаження при обробці запитів.

Клієнтська частина програми реалізована за допомогою бібліотеки Vue.js

[39]. Бібліотека Vue.js створена подібно до React.js і AngularJS, але набагато менше навантажує систему, не поступаючись у можливостях.

4.5.2 Навігація

Навігація в SPA ведеться за принципом класичного багатосторінкового додатку. Єдина відмінність у тому, що шляхи навігації є «віртуальними» і при зміні шляху лише відбувається якийсь JS сценарій. Наприклад, підвантажується новий компонент або надсилається запит на сервер для оновлення даних.

Для навігації клієнтом використовується бібліотека vue-router.js [40]. Основним завданням бібліотеки є зіставлення необхідних компонентів описаних шляхів. Також до завдань бібліотеки входить механізм переходу між шляхами. На рисунку 4.19 показано параметр “:to” для тега <b-nav>, який вказує зміну шляху при натисканні елемент. Необхідно відзначити, що при роботі з Vue.js можливе вбудовування JS коду в HTML код компонентів.

```
<template>
  <div id="app">
    <b-navbar class="mb-5" toggleable="lg" type="dark" variant="dark">
      <b-navbar-brand to="/home">Home</b-navbar-brand>

      <b-navbar-toggle target="nav-collapse"></b-navbar-toggle>

      <b-collapse id="nav-collapse" is-nav>
        <b-navbar-nav v-if="isLoggedIn">
          <b-nav-item :to="'/courses/page/' + (this.$store.getters['views/coursePage'].page || 1)">Courses</b-nav-item>
        </b-navbar-nav>

        <!-- Right aligned nav items -->
        <b-navbar-nav class="ml-auto">
```

Рисунок 4.19 – Механізм зміни шляху у компоненті App.vue

На рисунку 4.20 показано сторінку за допомогою «/courses/page/1», хоча сторінка не змінювалася жодного разу з моменту завантаження сайту. На цьому прикладі можна відзначити, що віртуальний шлях несе інформаційну функцію, оскільки користувач може зрозуміти, що зараз відображається перша сторінка курсів.

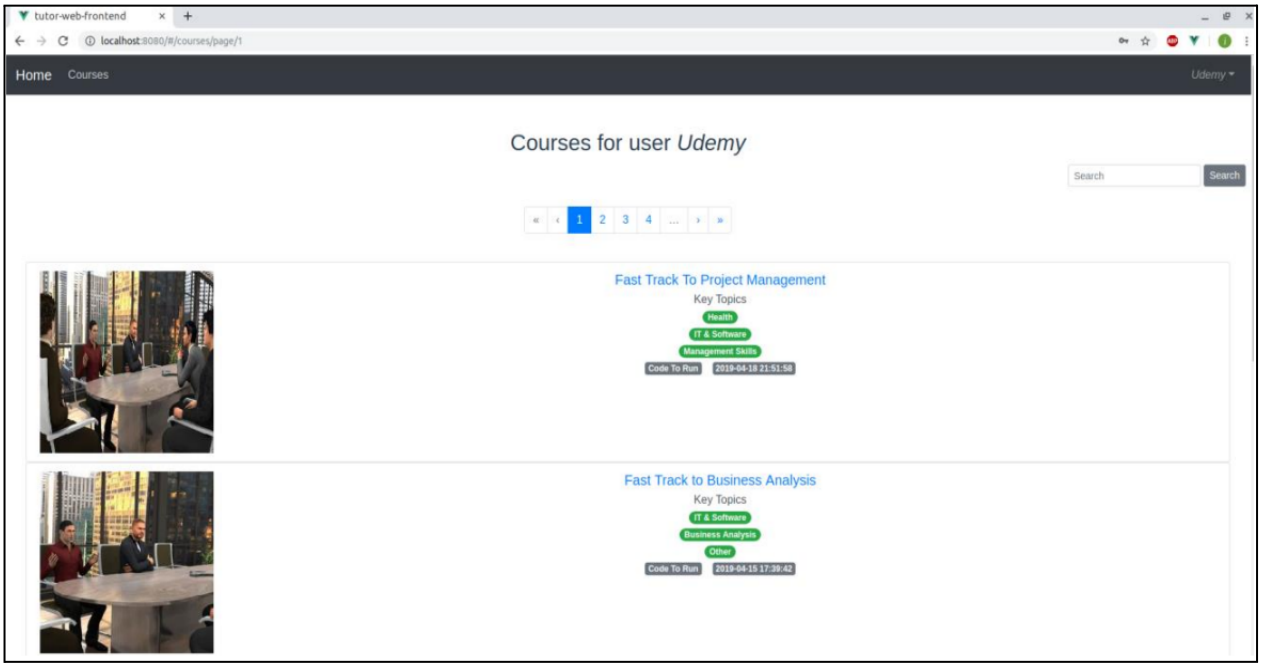

Рисунок 4.20 – Приклад сторінки з віртуальним шляхом

4.5.3 Взаємодія з серверною частиною

Як було описано раніше, клієнтська частина може посилати запити на сервер за певних умов, таких як зміна шляху або натискання на елемент. Для надсилання запитів до сервера API використовується бібліотека Axios [41]. Бібліотека дозволяє надсилати асинхронні запити на сервер. Для забезпечення цілісності даних на клієнті при надсиланні асинхронних запитів використовується бібліотека vuex.js [42]. Ця бібліотека створює склад даних, до яких можна звертатися з будь-якої частини клієнта. Для збереження цілісності даних ця бібліотека пропонує використовувати декілька типів функцій:

- мутації;
- події;
- геттери.

Мутації – це синхронні функції, які змінюють дані клієнта. Це єдині функції, яким можна змінювати дані, на думку цієї бібліотеки.

Події – асинхронні функції, які надсилають запити на сервер та викликають мутації. Через асинхронність цих функцій клієнт може виконувати відразу кілька запитів, але синхронні мутації будуть викликатися чітко без паралельного

втручання в один масив даних. На рисунку 4.21 показаний приклад використання асинхронної функції get_provider, яка викликає синхронні мутації "provider_update" та "error" за допомогою функції commit.

Геттери – функції передачі даних клієнта в компоненти, допоміжні складові, які не входять у клієнт-серверному взаємодії. Принцип використання синхронних мутацій та асинхронних подій схожий на принцип встановлення блокування ресурсу при використанні кількома потоками, що використовується операційними системами, системами керування сховищами тощо.

```
state: {
    course_page: {},
    search_string : '',
    curr_course : {}
},
mutations: {
    course_page_update(state, page) {
        state.course_page = page
    },
    search_string_update(state, search_string) {
        state.search_string = search_string ? search_string : ''
    }
},
actions: {
    get_provider({ commit }, id){
        return new Promise((resolve, reject) => {
            axios.get(`/provider/${id}/view`)
            .then(resp => {
                commit('provider_update', resp.data, {root: true})
                resolve(resp)
            })
            .catch(err => {
                commit('error')
                console.log(err)
                reject(err)
            })
        })
    },
```

Рисунок 4.21 – Синхронні та асинхронні функції

Таким чином, докладно було розглянуто особливості реалізації клієнтської частини програми, що є односторінковим додатком. Була описана архітектура клієнтської частини програми. Окремо було розібрано механізм навігації всередині клієнтської частини, а також методи взаємодії із серверною частиною.

ВИСНОВКИ

Освітні онлайн курси займають одне з провідних місць у сфері здобуття освіти. Кількість онлайн-платформ для проходження онлайн-курсів стрімко зростає з кожним днем. Програми-агрегатори дозволяють збирати інформацію про освітні онлайн курси з різних платформ, полегшуючи завдання користувачів. Основною метою даної роботи було створення клієнт-серверної програми для агрегації інформації про освітні курси з різних платформ зі спрощеним додаванням інформації про курси від постачальників. Мета роботи була успішно виконана за рахунок якісного виконання поставлених завдань.

У ході роботи було проаналізовано ринок постачальників освітніх онлайн-курсів, у тому числі найвідоміші платформи: Coursera, Udemy, AcademyOcean. У ході аналізу були виявлені основні недоліки перерахованих вище платформ, на підставі яких було складено список завдань і план робіт. Було реалізовано клієнт-серверний додаток з використанням бази даних. Серверна частина програми була реалізована за принципами REST архітектури, що дозволяє використовувати її як API. Для отримання даних зі сторонніх платформ було отримано доступ до бази курсів Udemy, а також розроблено програмний модуль для взаємодії з їх API.

В ході дослідження, проведеного в рамках магістерської дослідницької практики, було розроблено модульний інструментарій сегментації бази даних клієнтів на основі стандартних методів кластеризації та поєднання двох підходів до сегментації бази даних клієнтів, а також впроваджено інструментарій для підтримки його застосування. Він дозволяє використовувати різні модулі. Розроблена програма є гнучкою і може бути адаптована, оскільки пропонує можливість заміни алгоритмів на будь-які інші, які, будуть працювати без необхідності доопрацювання програмістом.

Ця робота має кілька можливих напрямів для подальшої розробки, а саме: отримання доступу до API інших освітніх платформ та розробка програмних модулів для взаємодії з ними, розширення можливостей клієнтської частини додатків до можливостей серверної частини, розробка клієнтського Android-додатка для споживачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. State of the MOOC 2017: A Year of Privatized and Open Education Growth // Online Course Report, 2023. URL: https://www.onlinecoursereport.com/ state-of-the-mooc-report/ (дата звернення: 16.04.2023).

2. Coursera // Coursera, 2023. URL: https://www.coursera.org/ (дата звернення: 16.04.2023).

3. Udemy // Udemy, 2023. URL: https://about.udemy.com (дата звернення: 16.04.2023).

4. AcademyOcean // AcademyOcean, 2023. URL: https://academyocean.com/ua (дата звернення: 16.04.2023).

5. Прикладна статистика: навч. посібник / В. О. Костюк; Харк. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків: ХНУМГ ім. О. М. Бекетова, 2015. – 191 с.

6. John P. van Gigch. Applied general systems theory. - Harper & Row, 1974. - 739 p.

7. Аналіз систем розпізнавання образів структури композитів : монографія / Добротвор І.Г., Стухляк П.Д., Микитишин А.Г., Митник М.М. – Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2018. – 224 с.

8. Бахрушин В.Є. Методи аналізу даних : навчальний посібник для студентів / В.Є. Бахрушин. – Запоріжжя : КПУ, 2011. – 268 с.

9. Фінанси бізнесу / [Ситник Н. С., Стасишин А. В., Попович Д.В., Сич О.А. та ін.]; за заг. ред. Н. С. Ситник.- Львів: Видавництво «Апріорі», 2019. - 432 с.

10. Черняк О.І., Захарченко П.В. Інтелектуальний аналіз даних: Підручник. – К.: Знання, 2010. - 840 с.

11. Пістунов І.М., Антонюк О.П., Турчанінова І.Ю. Кластерний аналіз в економіці: Навч. посібник – Дніпропетровськ: Національний гірничий університет, 2008.– 84 с.

12. PostgreSQL: The World's Most Advanced Open Source Relational Database

// PostgreSQL, 2023. URL: https://www.postgresql.org/ (дата звернення: 16.04.2023).

13. Udemy API and Web Service Introduction // Udemy, 2023. URL: https://www. udemy.com/course/api-and-web-service-introduction/ (дата звернення: 16.04.2023).

14. M.S. Aldenderfer, R.K. Blashfield. Cluster Analysis (Quantitative Applications in the Social Sciences). - SAGE Publications, 1984. - 88 p.

15. A.K. Jain R. Algorithms for Clustering Data / R.C. Dubes A.K. Jain. — New Jersey: Prentice Hall, 1988. — 334 p.

16. L. Kaufman. Finding Groups in Data: An Introduction to Cluster Analysis. Wiley Series in Probability and Statistics / L. Kaufman, P. J. Rousseeuw. — New York: John Wiley and Sons, 1990.

17. Albright S. C., Winston W., Zappe C. Data Analysis and Decision Making. Boston : Cengage Learning, 2016. 948 p.

18. Cao L., Philip S. Yu, Zhang C., Zhang H. Data Mining for Business Applications. Springer Science; Business Media, 2008. 402 p.

19. Linoff G. S. Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management. Indianapolis: Wiley, 2011. 888 p.

20. Tryon R. C. Cluster analysis / R. C. Tryon. – London : Ann Arbor Edwards Bros, 1939. – 139 p.

21. Hartigan J. A. Clustering Algorithms / J. A. Hartigan. – New York : Wiley, 1975. – 351 p.

22. Everitt B.S., Landau S., Leese M., Stahl D. Cluster Analysis. 5th Edition. John Wiley & Sons, 2011. 346 p.

23. Kassambara A. Practical Guide to Cluster Analysis in R. New York: STHDA, 2017. 187 p.

24. Romesburg, C. Cluster Analysis for Researchers [Text] / C. Romesburg. – Morrisville , North Carolina : Lulu.com, 2004. – 344 p.

25. Arabie, P. Clustering and classification [Text] / P. Arabie, L. J. Hubert, G. Soete. – Singapore : WorldScientific, 1996. – 490 p.

26. Пістунов І. М. Кластерний аналіз в економіці / І. М. Пістунов, О. П.

Антонюк, І. Ю. Турчанінова. – Д. : НГУ, 2008. – 87 с.

27. Бєлай С.В. Модифікація методу k-середніх кластерного аналізу у задачах прогнозування кризових явищ соціально-економічного походження / С.В. Бєлай, В.Е. Лісіцин // Збірник наукових праць Національної академії Національної гвардії України. 2014. Вип. 2. С. 29-34.

28. Гавриш К.С. Кластерний аналіз підприємств готельного господарства України / К.С. Гавриш // Бізнес Інформ. 2013. № 7. С. 216-224.

29. Забуранна Л.В. Кластерний аналіз підприємств сфери сільського аграрного туризму / Л.В. Забуранна // Ефективна економіка. 2013. № 1. URL: http://nbuv.gov.ua/UJRN/efek_2013_1_17.

30. Куцеконь Л.О. Теоретико-методичні аспекти кластеризації підприємств / Л.О. Куцеконь // Науковий вісник Херсонського державного університету. Сер.: Економічні науки. 2014. Вип. 8(1). С. 100-103.

31. Саричев В.І. Методичні підходи до застосування кластерного аналізу при дослідженні соціально-економічних аспектів людського розвитку / В.І. Саричев // Інтелект XXI. 2014. № 3. С. 97-106.

32. Якимець Р.В. Методи кластеризації та їх класифікація / Р.В. Якимець // Міжнародний науковий журнал. 2016. № 6(2). С. 48-50.

33. Semantic Web Services: A RESTful Approach // GitHub IO, 2023. URL: https://otaviofff.github.io/restful-grounding/ (дата звернення: 16.04.2023).

34. Introducing JSON // JSON, 2023. URL: https://www.json.org/json-en.html/ (дата звернення: 16.04.2023).

35. Pyramid, a Python Web Framework // Pyramid, 2023. URL: https://trypyramid.com/ (дата звернення: 16.04.2023).

36. Python // Python, 2023. URL: https://www.python.org/ (дата звернення: 16.04.2023).

37. SQLAlchemy - The Python SQL Toolkit and Object Relational Mapper // SQLAchemy, 2023. URL: https://www.sqlalchemy.org/ (дата звернення: 16.04.2023).

38. RFC 7519 - JSON Web Token (JWT) // IETF Tools, 2023. URL: https://tools.ietf.org/html/rfc7519 (дата звернення: 16.04.2023).

39. Vue.js - The Progressive JavaScript Framework // Vue.js, 2023. URL: https://vuejs.org/ (дата звернення: 16.04.2023).

40. Vue Router // Vue Router, 2023. URL: https://router.vuejs.org/ (дата звернення: 16.04.2023).

41. Github-actions[bot] and DigitalBrainJS chore(release): v1.3.4 // GitHub, 2023. URL: https://github.com/axios/axios/ (дата звернення: 16.04.2023).

42. What is Vuex? | Vuex // Vuex, 2023. URL: https://vuex.vuejs.org/ (дата звернення: 16.04.2023).

43. Sus, B., Tmienova, N., Revenchuk, I., Bauzha, O., Stirenko, S. Gamification Approach to the Creation of Virtual Laboratory Works and Educational Courses // CEUR Workshop Proceedings, 2020, 2711, pp. 68-78.

44. Dyomina, V., Bilova, T., Pobizhenko, I. An Information System to Support Internal Quality Assurance of Educational Programs // CEUR Workshop Proceedings, 2021, 3013, pp. 173–181.

45. Dudar, Z., Shubin, I., Kozyriev, A. Principles of Creating an Integrated Development Environment for Educational Computer Systems // Lecture Notes in Networks and Systems, 2021, 212 LNNS, стр. 415–435 doi: 10.1007/978-3-030-76343-5_22